AMCC 405GPr
PowerPC
Errata

Document Issue 1.00

September 2004

PPC405GPr Embedded Processor Errata

AMCC
APPLIED MICRO CIRCUITS CORPORATION

AMCC reserves the right to make changes to its products, its datasheets, or related documentation, without notice and warrants its products solely pursuant to its terms and conditions of sale, only to substantially comply with the latest available datasheet. Please consult AMCC's Term and Conditions of Sale for its warranties and other terms, conditions and limitations. AMCC may discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information is current. AMCC does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others. AMCC reserves the right to ship devices of higher grade in place of those of lower grade.

AMCC SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Preliminary

PPC405GPr Rev 1.1 (IBM25PPC405GPr-3xBxxxCx)

This document contains errata and design notes that affect designs using the PPC405GPr Rev 1.1 (PVR 0x50910951). Each erratum includes an overview, a description of the system impact and a description of possible work-around(s). Design notes cover items that are not considered errata but need a description beyond what is provided in published documentation. Refer to Tables 1 and 2 for a list of errata and design notes.

Errata are organized by item designator in alphabetical order. The item designator consists of an acronym for the affected functional unit and a numeric value. The numeric values assigned to errata and design notes are not necessarily consecutive.

List of Functional Unit Acronyms:

- CHIP Errata particular to the chip implementation
- CPU PPC405D4V6 processor core
- DMA DMA controller
- DCP Decompression Controller (CodePack™)
- EBC External Peripheral Bus Controller
- EMAC EMAC controller
- GPIO General Purpose I/O controller
- IIC IIC controller
- MAL MAL controller
- OCM On Chip Memory controller
- OPBA On Chip Peripheral Bus Arbiter
- PCI PCI controller
- PLB Processor Local Bus Arbiter
- POB On Chip Peripheral Bus Bridge
- SDRAM SDRAM controller
- UART UART controller
- UIC Universal Interrupt Controller
- ZMII EMAC to PHY interface bridge

Category definitions:

Errata are classified according to system impact and the availability of a work-around.

1. Major impact, no work-around is available. A problem is said to have a major impact if it results in a system crash, a hard failure, an unrecoverable soft failure, significant performance degradation, or the storage of incorrect data.
2. Major impact, work-around is impractical to implement, or a substantial risk of encountering the same or additional problems, including performance issues, exist after the work-around is implemented.
3. Major impact, work-around available. Application of the work-around either eliminates the problem, or reduces it to a minor impact issue.
4. Minor impact, no work-around is available. Minor impact problems result in slight to moderate performance degradation, or are a functional variance from specification.
5. Minor impact, work-around is available. Minor impact problems result in slight to moderate performance degradation, or are a functional variance from specification.
6. Design enhancement.

Errata Summary

Table 1. Errata Summary

Item	Category	Description	Date First Documented	Date Last Updated
CHIP_8	4	The EMAC does not enter sleep mode when the CPC0_ER[EMAC_MM, EMAC_RM, EMAC_TM] bits are enabled.	1/28/02	3/21/02
CPU_121	5	The iccci instruction may errantly cause a Data TLB Exception.	4/19/99	4/19/99
CPU_147	5	Virtual memory marked as non-executable storage (storage attribute EX=0) can be loaded into the instruction cache using the icbt instruction.	8/12/99	4/27/00
CPU_162	3	When in real mode, the 405 core may errantly make speculative instruction fetches from guarded storage.	1/7/00	2/24/00
CPU_197	3	Incorrect real mode attributes may be used when accessing the last instruction in a 128 MB region.	11/17/00	11/17/00
CPU_208	1	icbt instructions executed with data relocation enabled may cause incorrect instruction execution if the icbt misses in the UTLB or does not have permission to access the page.	8/21/01	8/21/01
CPU_213	3	Incorrect data may be flushed from the data cache.	2/21/03	5/15/03
EBC_20	3	No parity generated during a DMA memory to peripheral write.	4/3/03	5/15/03
EMAC_5	4	The EMAC constantly transmits a preamble pattern on the MII interface.	3/25/02	1/29/03
EMAC_7	5	Signal Quality Error (SQE) occasionally reported incorrectly during SQE test.	6/10/02	6/10/02
EMAC_8	5	Soft Reset may not reset all logic in the EMAC	5/7/03	5/15/03
EMAC_9	4	Integrated Flow Control Mechanism may not function correctly	5/7/03	5/15/03
EMAC_10	4	Octet Counter Registers may not record an accurate count	5/7/03	5/15/03
IIC_16	3	Received data in slave mode may be lost.	5/10/02	5/10/02
PCI_18	3	Executing code from PCI address space may hang the CPU.	2/23/01	2/23/01
PCI_24	1	The PCI Bridge Controller does not detect a parity error ($\overline{\text{PCIPErr}}$) asserted by a target during a write cycle.	2/27/01	1/10/03
PCI_25	1	Incorrect address or write data is driven on the PCI bus during a DAC transfer.	11/15/01	1/10/03
PCI_26	5	A parity error generated by outbound PCI reads cannot be masked by the PCIC0_CMD[PER] bit.	3/21/02	8/8/02
PCI_27	3	The internal PCI arbiter unfairly arbitrates when a master removes its request before the transfer.	9/24/02	10/23/02
PLB_4	3	Incorrect data may be stored in the PLB0_ACR during a DCR read after DCR write sequence.	8/21/01	8/21/01
POB_1	3	The PLB-to-OPB Bridge does not wake-up out of sleep mode.	3/11/02	3/11/02
SDRAM_11	1	The SDRAM controller does not consistently enforce the timing parameter tRFC.	9/3/02	10/10/02

Design Note Summary

Table 2. Design Note Summary

Item	Description	Date First Documented	Date Last Updated
1	When the processor is not powered (i.e., Vdd and OVdd = 0), additional precautions are needed if the IIC SDA and SCL are powered.	12/10/99	12/10/99
2	The SDRAM controller specification does not support mixing ECC and non-ECC memory banks.	12/10/99	12/10/99

CHIP_8 Enabling CPC0_ER[EMAC_MM, EMAC_RM, EMAC_TM] bits does not place EMAC asleep.

Category: 4

Overview:

The EMAC consists of three functional blocks: the Management Module (EMAC_MM), Receive Module (EMAC_RX), and Transmit Module (EMAC_TM). The EMAC functional blocks do not enter sleep mode when the CPC0_ER[EMAC_MM, EMAC_RM, EMAC_TM] bits are enabled. Since the EMAC is a class 1 device, it should unconditionally go to sleep once the EMAC_MM, EMAC_RM, and EMAC_TM bits are set in the clock power management enable register (CPC0_ER).

Impact:

The EMAC cannot be placed asleep.

Work-around:

No work-around is available.

CPU_121 The iccci instruction may errantly cause a Data TLB Exception.

Category: 5

Overview:

When data-side relocation (data address translation) is enabled (MSR[DR] = 1), an iccci instruction errantly attempts an access check for the associated page. Since iccci invalidates the entire instruction cache, the effective address it generates is unnecessary.

Impact:

When data-side relocation (data address translation) is enabled, the execution of an iccci may cause a Data TLB miss exception.

Work-around:

There are two possible work-arounds. Work-around 1 avoids this erratum by temporarily disabling data address translation. Work-around 2 describes how to handle this erratum without disabling data address translation.

1. Before executing an iccci instruction, make sure the MSR[DR] is disabled. This can be done using the following pseudo code:

```
mfmsr Rx                    ! Rx is a scratch reg
andi  Ry,Rx,DR_MASK         ! clear MSR[DR] in scratch reg Rx
mtmsr Ry
isync
iccci 0,Rx                  ! The address does not matter.
mtmsr Rx
isync
```

2. When data-side relocation is enabled, ensure that the virtual address generated by the iccci (virtual address = {PID, effective address (RA | 0 + RB)}) has a corresponding page in the TLB.

CPU_147 Virtual memory marked as non-executable storage (storage attribute EX=0) can be loaded into the instruction cache using the icbt instruction.

Category: 5

Overview:

The icbt instruction should execute as a nop (no operation) when the effective address corresponds to a memory page marked as non-executable. Instead, an instruction cache line fill occurs when the effective address of the icbt instruction maps to memory region having the following three characteristics:

1. Marked as non-executable storage (storage attribute EX = 0).
2. Marked as cacheable (storage attribute I = 0).
3. Access is not prohibited by a zone fault (The access control field ZSEL references a ZPR field that does not prohibit access).

Impact:

Touching data belonging to a page marked as non-executable storage into the instruction cache with the icbt instruction can result in unnecessary memory accesses. Note that:

1. Memory marked as non-executable (EX = 0) cannot be executed even if loaded into the instruction cache.
2. icbt instructions are not compiler generated; they are isolated to assembly routines.

Work-around:

1. No work-around is necessary if either:
   a. The translation from virtual to real does not change.
   b. There are no occurrences where an icbt instruction causes a cache line fill of data from a page marked as non-executable storage.
2. Invalidate cache blocks (lines) loaded with data belonging to a page marked as non-executable storage. Use either an icbi or an iccci instruction.

CPU_162 When in real mode, the 405 core may errantly make speculative instruction fetches from guarded storage.

Category: 3

Overview:

In real mode, if instructions (guarded storage or not) and memory mapped I/O (guarded storage) are within 1KB of each other, it is possible for the I/O to be speculatively accessed when the instructions are executed.

Impact:

Memory mapped I/O (MMIO) may be speculatively accessed. An unintentional access to MMIO could result in a loss of data.

Work-around:

Maintain at least 1KB of separation between instructions and memory mapped I/O.

CPU_197 Incorrect real mode attributes may be used when accessing the last instruction in a 128 MB region.

Category: 3

Overview:

When executing instructions in real mode (MSR[IR] = 0), an access to the last instruction in a 128 MB region uses the real mode attributes of the next consecutive 128 MB region under any of the following conditions:

1. The last instruction in a 128 MB region contains a branch target that is non-cacheable or causes a cache miss.
2. The address restored by an rfi or rfci is the last instruction in a 128 MB region and this address is non-cacheable or causes a cache miss.
3. The next to the last instruction in a non-cacheable 128 MB region contains a branch which is predicted taken but is not taken.
4. The next to the last instruction in a non-cacheable 128 MB region contains an isync instruction.

Note:

1. Real mode attributes are specified by the ICCR, SU0R, and SLER registers.
2. All 128 MB regions have the same storage attributes after reset. Therefore, a branch instruction at the reset vector 0xFFFFFFFC is not affected by this erratum after a core, chip, or system reset.
3. In real mode, the storage regions wrap making the last storage region (0xF8000000 - 0xFFFFFFFF) and the first storage region (0x00000000 - 0x07FFFFFF) consecutive.

Impact:

The table below lists the impact of encountering one of the conditions listed above. The first column contains the real mode storage attribute of the 128 MB region being accessed and the second column contains the real mode storage attribute of the next consecutive 128 MB region.

Table 3. Description of Impact for Item CPU_197

Request from First Region	Request assumes the attribute of Second Region	Impact
Non-cacheable	Cacheable	An instruction cache line fill from the non-cacheable storage region may occur.
Cacheable	Non-cacheable	A desired instruction cache line fill from the cacheable storage region may not occur.
Big Endian	Little Endian	The request from the big endian storage region is treated as little endian storage. A program exception may be generated, or an incorrect instruction may be executed.
Little Endian	Big Endian	The request from the little endian storage region is treated as big endian storage. A program exception or machine check exception may be generated.
Non-compressed Storage Region	Compressed Storage Region	The request from the non-compressed storage region is treated as compressed storage. A program exception may be generated.
Compressed Storage Region	Non-compressed Storage Region	The request from the compressed storage region is treated as non-compressed storage. A program exception may be generated.

CPU_197 Continued

Work-around:

1. No work-around is required if any of the following apply:
   a. Code does not exist in the last two word locations of a 128 MB region.
   b. The real mode storage attributes of any 128 MB region containing executable code are identical to the attributes for the next contiguous 128 MB region.
   c. The last two words of a 128 MB region are only accessed while in virtual mode (MSR[IR] = 1).
2. Do not place code in the last word locations of a 128 MB region.
3. When performing a soft reset, branch directly to the entry point of the application code. Do not branch to the reset vector (0xFFFFFFFC). Unlike a core, chip or system reset, a soft reset does not guarantee that all 128 MB aligned regions have the same storage attributes. The branch at the reset vector in the last storage region (0xF8000000 - 0xFFFFFFFF) will obtain the storage attributes of the first storage region (0x00000000 - 0x07FFFFFF). If these storage attributes differ, unexpected results are possible.

CPU_208 icbt instructions executed with data relocation enabled may cause incorrect instruction execution if the icbt misses in the UTLB or does not have permission to access the page.

Category: 1

Overview:

icbt instructions executed with data relocation enabled (MSR[DR] = 1) may cause incorrect instruction execution if the icbt misses in the UTLB or does not have permission to access the page.

For the icbt to cause the instruction cache to deliver incorrect instructions to the fetcher a number of events must line up.

Conditions:

1. Data relocation is enabled (MSR[DR] = 1).
2. Either condition 2a or 2b is true.
   a. The TLB page referenced by the icbt instruction is not found in the UTLB.
   b. The TLB page referenced by the icbt instruction is marked protected by its ZPR settings.
3. A cache line fill completes while the execute logic is requesting the instruction cache unit to perform an icbt.
4. The fetcher has to be requesting an address for a new cache line followed by a request for the previous cache line. This condition occurs when the fetcher re-requests data thrown away because there was no room in the fetch queue.
5. In the same cycle as conditions 3 and 4, the icbt must be presented to the instruction cache. The instruction cache must not accept the fetch request this cycle, but does accept the icbt.

Impact:

An incorrect instruction may be executed after execution of an icbt instruction.

Note, this erratum does not affect compiler generated code. The icbt instruction is not a compiler generated instruction.

Work-around:

Perform one of the following:

1. Ensure data relocation is disabled (MSR[DR] = 0) when executing an icbt instruction.

or

2. When data relocation is enabled, ensure the TLB page referenced by an icbt instruction is in the UTLB and not protected by access control settings.

or

3. When data relocation is enabled, execute an isync instruction after every icbt instruction.

CPU_213 Incorrect data may be flushed from the data cache.

Category: 3

Overview:

The CoreConnect™ architecture based PPC405GPr consists of masters and slaves interconnected via the Processor Local Bus (PLB). The PPC405 CPU features separate PLB master interfaces for the instruction cache unit (ICU) and the data cache unit (DCU). Any CPU load, store, or data cache control instruction may result in the CPU issuing one or more transactions through its DCU PLB interface. Whether or not a given data operation results in a PLB transaction depends on factors including the cacheability of the referenced address and, in the case of a cacheable access, the state of the data cache. Once the DCU initiates a PLB read or write request the transaction proceeds with timing that is dependent on both the PLB slave responsible for servicing the given address and PLB activity that may be occurring between other masters and slaves. If the PPC405 DCU interface performs the following specific sequence of PLB operations incorrect data may be flushed from the data cache:

1. Data cache flush of line1 (line write): This flush is a pre-conditioning operation and is not tightly coupled with the sequence of operations defined by steps 2-5. It is only required that this flush be the last data side write to the PLB prior to steps 2-5.
2. Data cache fill of line2 (line read): This fill replaces a dirty line and causes the flush in step 5.
3. Data cache fill of line3 (line read): This fill is to a different congruence class than the line2 data cache fill. It may or may not cause data to be replaced or flushed.
4. Non-cacheable or write-through write of word4 (word, halfword or byte write): The timing of the request on the PLB for this write must occur within a narrow window immediately after the CPU receives the last data item from the data cache fill of line3.
5. Data cache flush of line2 (line write): This flush is due to the data cache fill of line2 in step 2. The data written to memory is incorrect.

For this erratum to occur the PLB slave servicing the data cache fill of line3 must return the cache line data in four consecutive PLB cycles. Since the PLB data buses are 64-bits wide and all of the slaves in the PPC405GPr feature 32-bit external buses, line fill data is in most cases returned every other PLB cycle. For data to be returned in consecutive PLB cycles the slave must have accepted the line fill request from the CPU as a pipelined transfer, and read and buffered the line contents from external memory while at the same time another PLB transaction between a different master and slave maintained ownership of the on-chip PLB read data bus.

An instruction sequence that can cause the above series of data side PLB operations is as follows:

1. A load or store instruction that misses in the data cache and causes a fill to a line with the Least Recently Used (LRU) tag pointing to a valid dirty line, or a dcbf to a line with valid dirty data: This instruction causes the flush of line1.
2. A load or store instruction that misses in the data cache and causes a line fill to a line with its LRU pointing to a valid dirty line: This instruction causes both the fill of line2 and a subsequent flush of the dirty contents of line2 that were replaced by the fill.
3. A store instruction that misses in the data cache and causes a line fill: This instruction causes the fill of line3 and may or may not result in a line flush.
4. A store instruction to a memory address designated as either non-cacheable or write-through: This instruction causes the non-cacheable write of word4.
5. A load instruction requiring data from either or both of the last two 64-bit doublewords returned for the cache line fill of line3. This load does not cause a PLB operation, but is required to create the conditions necessary to cause this erratum.

CPU_213 Continued

These instructions need not execute with a particular timing relative to each other. Rather, they may be separated by other unrelated instructions and operations including branches, interrupts, instruction cache misses, and additional loads or stores that hit in the data cache.

Impact:

Incorrect data may be flushed (written) to memory from the data cache.

Work-around:

Perform one of the following work-arounds:

- Set CCR0 reserved bits 1 and 3. When these bits are set and there is a line fill pending or in progress for a data cache miss the data cache is prevented from servicing other CPU load or store requests. Because this work-around blocks data cache accesses during line fills it may affect performance.
- Mark all data memory as write-through using the DCWR register or W storage attribute of each TLB entry.

EBC_20 No parity generated during a DMA memory-to-peripheral write.

Category: 3

Overview:

Setting the parity enable bit in the DMA controller register does not produce the expected results during DMA memory-to-peripheral writes. The EBC drives all 0's on PerPar0:3 instead of odd parity.

Impact:

DMA memory-to-peripheral mode cannot be used if parity is required. Work-around requires the use of an EBC bank.

Work-around:

The recommended work-around is to use hardware device paced memory-to-memory DMA transfers. This type of transfer is still initiated using DMAReq, but as stated in the DMA section of the user's manual, a chip select is used in place of the DMAAck. An unused EBC bank is required since a chip select is needed. The EBC bank must be properly configured for accessing memory to perform memory-to-memory DMA transfers and the bank's parity enable bit (EBC0_BnAP[PEN]) for must be set. The EBC address pins can be ignored. The only requirement when the DMA is programmed is the source/destination address registers match the address range configured in the EBC bank.

EMAC_5 The EMAC constantly transmits a preamble pattern on the MII interface.

Category: 4

Overview:

The minimum value specified for the TLR (EMAC0_TMR1[TLR]) is wrong. The correct equation is:

min(TLR) = (MAL Burst Limit / 2) + 1

Since the MAL Burst Limit is 16, the correct minimum value for TLR is 9.

Impact:

An incorrect TLR setting (below the minimum) can cause the EMAC to constantly transmit a preamble pattern on the MII interface which blocks anything else from being transmit. This error does not generate an error indication such as an interrupt or a status bit.

Work-around:

Set the TLR value to 9 (EMAC0_TMR1[TLR]=9).

EMAC_7 Signal Quality Error (SQE) occasionally reported incorrectly during SQE test.

Category: 5

Overview:

Occasionally while running the Signal Quality Error (SQE) test, errors may be reported even though none have occurred. The SQE test is only used when operating an MII interface at 10Mbs in half-duplex mode, and is used to verify the Collision Detect logic in the PHY is operating properly.

Impact:

This is a minor problem and has no affect on data transfers with the EMAC.

Work-around:

Disable the SQE test in the EMAC.

EMAC_8 Soft Reset may not reset all logic in the EMAC

Category: 5

Overview:

Soft Reset (EMAC0_MR0[SRST]) may not reset all logic in the EMAC.

Impact:

Once Soft Reset has completed, and the receive channel is re-enabled, an extra data double word may be added at the beginning of the first received packet without any bad status indication on this packet when forwarded to the MAL.

Work-around:

Before Soft Resetting the EMAC, first disable the receive channel (EMAC0_MR0[RXE]=0), then wait for the RX MAC Idle bit to be set (EMAC0_MR0[RXI]=1). Once idle, enable the Soft Reset bit (EMAC0_MR0[SRST]=1).

EMAC_9 Integrated Flow Control Mechanism may not function correctly

Category: 4

Overview:

The integrated flow control mechanism is used only in Full-duplex mode in order to avoid receive FIFO overrun. Two watermarks are set on the receive FIFO. When the high watermark is crossed upwards, an urgent pause packet is internally assembled and transmitted causing the other end's transmitter to stop. When the low watermark is crossed downwards, a zero pause packet is internally assembled and transmitted, thus causing the other end to resume transmission.

There are three problems with the integrated flow control mechanism:

1. If a request is issued, and in the meanwhile the opposite watermark is crossed, the request initiator could take the request back. A request is taken back by de-asserting the request signal without waiting for acknowledge. This condition can lead to a deadlock in the handshake protocol, since the request initiator cannot know if the transmitter received the request (or if it is going to acknowledge the request).
2. The integrated flow control mechanism cannot issue two consecutive urgent pause packet requests.
3. When a request for a zero pause packet is issued, and until an acknowledge is accepted, the receive FIFO can theoretically reach the high watermark. The issue pause mechanism is not aware of this high watermark crossing, and thus does not send an urgent pause packet.

Impact:

The impact of the three problems list above is as follows:

1. Total failure of the issue pause mechanism preventing future pause control frames from being issued as a response to watermark crossings.
2. FIFO overflow - A user may want to set the Pause Timer Register to a small value that causes only a short stop, which would not necessarily cause the receive FIFO to fall below the low watermark.
3. FIFO overflow - The FIFO overflow occurs because the high watermark crossing does not cause an immediate request for non-zero pause packet transmission.

Work-around:

No work-around is available. Integrated Flow Control should be disabled (EMAC0_MR1[EIFC]=0).

EMAC_10 Octet Counter Registers may not record an accurate count

Category: 4

Overview:

The Transmitted Octet Count Register (EMAC0_OCTX) and Received Octet Count Register (EMAC0_OCRX) contain the total number of bytes transmitted and received respectively on the media (including bad packets). After a packet is transmitted or received, the transmitter or receiver sends the number of frame bytes to the count register logic, where it is added to the proper counter register. Since the transmit and receive logic are on a different clock domain from the register logic, synchronization across the clock boundary is necessary. In the process of crossing the clock boundary, the count from the transmitter or receiver may be incorrectly sampled, therefore, causing the counter registers to record inaccurate values.

Impact:

The EMAC0_OCTX and EMAC0_OCRX registers may not contain an accurate count.

Work-around:

No work-around is available.

IIC_16 Received data in slave mode may be lost.

Category: 3

Overview:

Any data received when the slave data buffer is full (i.e. contains four bytes of data) is lost.

Impact:

Data received in slave mode can be lost.

Work-around:

Do not wait until the slave data buffer is full before reading it.

PCI_18 Executing code from PCI address space may hang the CPU.

Category: 3

Overview:

As with all high-performance processors, the PPC405 CPU features an instruction pipeline designed to provide the execution unit with a steady stream of instructions. To optimize throughput, the instruction fetcher attempts to keep the pipeline full by requesting instructions along the most probable path of code execution. Whenever an instruction fetch cannot be fulfilled by the instruction cache, either a 4-word or 8-word line fill request is forwarded via the on-chip Processor Local Bus (PLB) to either the SDRAM, PCI, or peripheral bus controller. In the case of instruction fetches from PCI address space, the PCI controller obtains and internally buffers the requested instructions from the PCI target before returning them to the CPU. If during this period of time an instruction such as a conditional branch or an interrupt alters the code flow such that the pending instruction fetch from PCI address space is no longer necessary, the CPU may abort the transfer.

Impact:

If the CPU aborts an instruction fetch from PCI address space and then subsequently performs a data read from PCI address space, the read may hang the PLB bus. This hang condition will persist until the CPU fetches another instruction from PCI address space.

Work-around:

If the CPU is not fetching instructions from PCI address space no work-around is required. Otherwise, ensure that the CPU is the only master in the PPC405GPr accessing PCI address space and do all of the following:

1. Completely disable instruction caching. If operating in real mode, MSR[IR] = 0, set ICCR = 0. If running in virtual mode, MSR[IR] = 1, ensure that all TLB entries that allow execution, TLBLO[EX] = 1, also inhibit instruction caching: TLBLO[I] = 0.
2. Program CCR0[24] = 1. Setting this undocumented bit ensures that the CPU cannot continuously satisfy instruction requests from the instruction cache fill buffer.
3. Program PCIL0_PMMnHA = 0x0 for all PLB to PCI mappings containing executable code. This prevents the PCI Bridge from generating dual address cycles on the PCI bus.

Items 1 & 2 in this work-around ensure that any CPU data request from PCI address space will be followed by an instruction fetch, thus preventing the PCI Bridge from becoming stuck in a hang condition. Following an aborted instruction fetch, the PCI Bridge may fail to complete a delayed read. Therefore, the PCI target providing instructions to the PPC405GPr may be unresponsive until its delayed read discard timer expires after 215 PCI clocks. Also, the target may detect this situation as an error condition. The system must be tolerant of these events.

> **Note:** If the PCI target is another PPC405GPr, its delayed read discard timer will expire, allowing forward progress, and no error will be detected.

Since executing directly from PCI address space is relatively slow, and because data in PCI address space cannot be cached, (the PCI Bridge does not support cache line write operations) it is highly recommended that systems boot from PCI and load their application into SDRAM memory. Boot code executing from PCI space should set CCR0[24] = 1 immediately after the branch from address 0xFFFFFFFC. Once the load process completes and execution is from SDRAM address space, enable instruction caching and clear CCR0[24].

PCI_24 The PCI Bridge Controller does not detect a parity error ($\overline{\text{PCIPErr}}$) asserted by a target during a write cycle.

Category: 1

Overview:

The PCI Bridge Controller fails to detect a parity error ($\overline{\text{PCIPErr}}$) asserted by a target during a write cycle under the following conditions:

1. The PCI Bridge is operating in Asynchronous Mode.
2. The PCI Bridge is running an outbound (PLB-to-PCI) write.
3. The PCI target asserts $\overline{\text{PCIPErr}}$ in response to detecting a data parity error.
4. The parity error is for the last beat of a burst or for a single beat.
5. The PCIClk frequency is within the following range: 0.7*SyncPCIClock ≤ PCIClk ≤ 1.1*SyncPCIClock

   PCIClk ranges affected by this erratum for typical clock configurations:

   - PLB Clk = 100MHz, SyncPCIClock = 50MHz, 35MHz ≤ PCIClk ≤ 51MHz
   - PLB Clk = 133MHz, SyncPCIClock = 66.7MHz, 46.7MHz ≤ PCIClk ≤ 66.7MHz
   - PLB Clk = 133MHz, SyncPCIClock = 44.4MHz, 31.1MHz ≤ PCIClk ≤ 46MHz

   Note: The SyncPCIClock is an internal clock derived from the PLB Clk and PCIClk is supplied by the PCI bus.

Impact:

A parity error may not be detected during a write cycle.

Work-around:

Select a PCIClk outside the failing frequency range such that PCIClk < 0.7*SyncPCIClock or PCIClk > 1.1*SyncPCIClock.

PCI_25 Incorrect address or write data is driven on the PCI bus during a DAC transfer.

Category: 1

Overview:

When the PCI Bridge Controller is the initiator of a DAC (Dual Address Cycle) transfer in Asynchronous Mode, it may drive incorrect address and/or write data values on the PCI bus. This error can occur if the internal SyncPCIClock and the externally supplied PCIClk have clock periods within ±2ns of one another. The following are PCIClk frequency ranges for typical clock configurations that are affected by this erratum:

- PLB Clk = 100MHz, Sync PCI Clk = 33.3MHz, 31MHz ≤ PCIClk ≤ 36MHz
- PLB Clk = 100MHz, Sync PCI Clk = 50MHz, 45MHz ≤ PCIClk ≤ 55MHz
- PLB Clk = 133MHz, Sync PCI Clk = 66.7MHz, 59MHz ≤ PCIClk ≤ 66.7MHz

Note:

(1) DAC cycles are used to address PCI targets in the address space above 4 Gigabytes.
(2) The SyncPCIClock is an internal clock derived from the PLB Clk and the PCIClk is supplied by the PCI bus.

Impact:

Address and data values driven on the PCI bus are not reliable during a DAC transfer.

Work-around:

Use one of the following clock configurations: PLB Clk/SyncPCIClock/PCIClk = 100MHz/100MHz/66MHz, 100MHz/50MHz/33MHz, 133MHz/44MHz/33MHz.

PCI_26 A parity error generated by an outbound PCI read cannot be masked by the PCIC0_CMD[PER] bit.

Category: 4

Overview:

The Parity Error Response bit (PCIC0_CMD[PER]) determines whether a PCI error is recorded by the PLB Bus Error Assertion Event status bit (PCIC0_ERRSTS[MEAE]) and reported to the PLB master. However, the PCIC0_CMD[PER] does not reliably mask a parity error generated by an outbound PCI read. (Outbound traffic is generated when the PCI Bridge is a PLB slave and a PCI Master.) The table below list all scenarios where PCIC0_CMD[PER] fails to correctly mask outbound parity errors.

Note: The PCIC0_CMD[PER] correctly masks the PCIC0_STATUS[DPE] for outbound reads.

Table 4. List all affected PCIC0_CMD[PER] and PCIC0_ERREN[MEAE] bit combinations

PCI data bus Parity Error	PCIC0_CMD[PER]	PCIC0_STATUS[DPE]	PCIC0_ERREN[MEAE]	PCIC0_ERRSTS[MEAE]	Error Reported to PLB Master
Outbound Read	0	0	0	May report an error regardless of the PCIC0_CMD[PER]	No error reported
Outbound Read	0	0	1	May report an error regardless of the PCIC0_CMD[PER]	May report an error
Outbound Read	1	1	1	May not report an error	May not report an error

Impact:

Outbound read parity errors are unreliably masked by the PCIC0_CMD[PER] bit.

Work-around:

1. To prevent a PCI Parity error from being reported to a PLB master, mask PCI parity errors and disable error reporting by setting PCIC0_CMD[PER]=0 and PCIC0_ERREN[MEAE]=0. Note, clearing the PLB Error Assertion Enable (PCIC0_ERREN[MEAE]=0) prevents any bus error from being reported to a PLB master.
2. There is no work-around to guarantee a PCI parity error is reported to a PLB master when PCI parity errors are unmasked and error reporting is enabled, PCIC0_CMD[PER]=1 and PCIC0_ERRSTS[MEAE]=1.

PCI_27 The internal PCI arbiter unfairly arbitrates when a master removes its request before the transfer.

Category: 3

Overview:

The internal PCI arbiter does not track fairness in the situation given in *Table 5* and described below.

Cycle 1. Master A performs a read arbitration request (via one of $\overline{\text{PCIReq1:5}}$) to the PPC405GPr (inbound).
Cycle 2. Master A is granted the bus.
Cycle 4. Master B performs a read arbitration request to the PPC405GPr (inbound).
Cycle 5. The PPC405GPr retries the request from Master A (by asserting $\overline{\text{PCIStop}}$ and deasserting $\overline{\text{PCITRDY}}$). The read is handled as a delayed transaction.
Cycle 6. When Master A is retried, Master B deasserts its read request for one clock cycle.
Cycle 7. Master B is granted the bus and begins a read cycle on the bus.
Cycle 9. Master B is retried by the PPC405GPr.
Cycle 11. One PCI clock cycle after the retry in cycle 9, Master B detects the bus is idle with the grant going away and immediately starts a new read cycle.
Cycle 13. Master B is again retried because the PPC405GPr has read data available for Master A.

Cycles 11 through 14 repeat. With the early removal of the Master B read request in cycle 10, the arbiter fairness algorithm does not block Master B and Master A does not receive grant long enough to obtain the bus.

Table 5. PCI Tenure Sequence demonstrating erratum PCI_27

Cycle	Master	REQ_A	GNT_A	REQ_B	GNT_B	FRAME	IRDY	DEVSEL	STOP
1		0	1	1	1	1	1	1	1
2		0	0	1	1	1	1	1	1
3	A	0	0	1	1	0	1	1	1
4		0	0	0	1	1	0	1	1
5		0	1	0	0	1	0	0	0
6		1	1	1	0	1	1	1	1
7	B	1	1	0	1	0	1	1	1
8		0	1	0	0	1	0	1	1
9		0	1	0	0	1	0	0	0
10		0	1	1	0	1	1	1	1
11	B	0	1	1	1	0	1	1	1
12		0	0	0	1	1	0	1	1
13		0	1	0	0	1	0	0	0
14		0	1	1	0	1	1	1	1
15	B	0	1	1	1	0	1	1	1
16		0	0	0	1	1	0	1	1
17		0	1	0	0	1	0	0	0
18		0	1	1	0	1	1	1	1

PCI_27 Continued

Master B removes its read request in cycle 10 before the arbiter records which master received the grant. Subsequent requests by Master B are treated as its first. The arbiter repeatedly grants the bus to Master B and does not give Master A an opportunity to complete the delayed read. This situation continues until either the delayed-read discard timer expires or the PPC405GPr generates an outbound request. The delayed-read discard timer expires after 2^{15} PCI clocks.

Impact:

The PCI arbiter repeatedly grants the bus to only one master resulting in a live lock condition.

Work-around:

Perform one of the following if a PCI Master behaves as Master B described in the overview:

1. Delay the $\overline{\text{PCIReq}}$ from the PCI Master by one PCI clock by inserting a flip-flop between PCI Master $\overline{\text{PCIReq}}$ pin and the PCI arbiter (arb_req# = flip_flop_Q; flip_flop_D = master_req#, flip_flop_CLK = PCI_CLK)

   If timing can be met, the output of the flip-flop can additionally be AND'ed with the input for better performance (arb_req# = flip_flop_Q & flip_flop_D; flip_flop_D = master_req#, flip_flop_CLK =PCI_CLK)

2. Use an external PCI arbiter.

PLB_4 Incorrect data may be stored in the PLB0_ACR during a DCR read after DCR write sequence.

Category: 3

Overview:

If a DCR write operation (mtdcr) to the PLB Arbiter Control Register (PLB0_ACR) is immediately followed by a DCR read operation (mfdcr) of the same register, the PLB0_ACR will latch in 0s instead of the intended DCR write data.

Impact:

Incorrect data may be stored in the PLB0_ACR.

Work-around:

Do not read the PLB0_ACR register immediately after writing it. Separate a DCR read from a DCR write of the PLB0_ACR with a DCR read of any register other than PLB0_ACR.

POB_1 The PLB-to-OPB Bridge does not wake-up out of sleep mode.

Category: 3

Overview:

When sleep mode is enabled the PLB to OPB Bus Bridge may not wake from sleep mode if a PLB transfer is issued to the Bridge.

Impact:

The PLB to OPB Bus Bridge cannot be waked.

Work-arounds:

Do not enable sleep mode for the PLB to OPB Bus Bridge. This block of logic is very small so the power consumption impact is minimal.

SDRAM_11 The SDRAM controller does not consistently enforce the timing parameter tRFC.

Category: 1

Overview:

When entering and exiting self refresh mode, the SDRAM controller, if operating at 133 MHz, does not meet the tRFC timing requirement for physical bank 3. Part of the sequence required for entering and exiting self refresh mode is an auto refresh. The tRFC timing parameter is the time between auto refresh commands. Typical SDRAM requires a tRFC of 60 ns; the SDRAM controller provides 52.5 ns under the following conditions:

1. Memory attached to physical bank 3 i.e. $\overline{\text{BankSel3}}$.
2. Entering and exiting self refresh. Self refresh mode is enabled by the SDRAM0_CFG[SRE] bit field.
3. The SDRAM interface is operating at 133 MHz.

Impact:

Data integrity cannot be guaranteed if tRFC is violated. The exact impact of this erratum depends on how the memory responds to a tRFC of 52.5 ns.

Work-arounds:

Avoid operating under the conditions described in the overview by performing one of the following:

1. Do not use self refresh.
2. Do not use bank 3.
3. Operate the SDRAM interface at 100 MHz.
4. Use SDRAM modules that have a tRFC less than 52.5 ns when operating the SDRAM at 133 MHz.

Design Notes:

1. When the processor is not powered (i.e., Vdd and OVdd = 0), additional precautions are needed if the IIC SDA and SCL are powered. Under this circumstance, the processor is in a high-impedance state and may be damaged and/or experience a degradation in reliability. (Note: The IIC SDA and SCL I/Os are 5V tolerant which implies that the voltage on these I/Os may be as high as 5.5V.) The following critical items must be considered:
   a. If the input voltage to the PPC405xx/NPe405x exceeds the ESD protection diode drop and is supplied with a current in excess of 147mA, the IIC SDA and SCL I/Os will be permanently damaged.
      i (Vsource – (4*0.8)/(Rsource)
      ii Vsource = source voltage = 5.5V maximum
      iii Rsource = source resistance = Pull-up Resistor for 0.147 = (5.5 – 3.2)/Rsource or Rsource = 2.3/0.147 = 15.65 ohms (minimum)

      Raising the value from this minimum of 15.65 ohms and/or reducing Vsource from 5.5 V will give added margins of safety against the I/O being permanently damaged.
   b. A sustained (i.e. several hours) maximum current greater than 8.1 mA will weaken or degrade the reliability of the IIC SDA and SCL I/Os. To improve reliability choose a pull-up resistor, Rsource, greater than 284 ohms, 0.0081 = (5.5 – 3.2)/Rsource = 284 ohms. (The recommended value for Rsource in the processor data sheet is 3K ohms. 284 ohms is the minimum for protection against damaging currents and 3K ohms is the maximum that guarantees a logic 1/high with a pull-up.)
   c. Use a series resistor to ensure that the voltages applied to SDA and SCL inputs do not exceed a 3.2 V drop across the ESD diodes when Vdd = OVdd = 0 V. The choice of series resistor depends on the pull-up resistor used and the required voltage levels at the inputs of the other devices on the IIC bus (Vih, min = 2.4 V).
2. The SDRAM controller does not support mixing ECC and non-ECC memory banks.

Revision Log

Rev	Contents of Modification
August 20, 2002	Errata document version 3.0.0 • Items added: EMAC_5, EMAC_7, IIC_16, POB_1 • Changed item numbers for the following items: Old Item Number → New Item Number CPU_9 → CPU_147 CPU_10 → CPU_162 CPU_15 → CPU_121 CPU_65 → CPU_197 CPU_75 → CPU_208 EMAC_80 → CHIP_8 PCI_68 → PCI_18 PCI_81 → PCI_26 PLB_79 → PLB_4 IIC_1 → 1 SDRAM_2 → 2
November 8, 2002	Errata document version 4.0.0 • Items added: PCI_27, SDRAM_11 • Modified introduction (page 1)
January 31, 2003	Errata document version 5.0.0 • Modified: EMAC_5 • Items added: PCI_24, PCI_25
May 15, 2003	Errata document version 6.0.0 • Items added: CPU_213, EBC_20, EMAC_8, EMAC_9, EMAC_10

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

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

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

405xx_600_errata_.fm.6.0.0
May 15, 2003