Freescale Semiconductor
Errata

Document Number: MPC7410CE
Rev. 17, 06/2005

Chip Errata for the MPC7410

The MPC7410 is a PowerPC™ microprocessor. This document details all known silicon errata for the MPC7410 and its derivatives. Table 1 provides a revision history for this chip errata document.

Table 1. Document Revision History

Revision Number	Release Date	Significant Changes
17	6/21/2005	Added error 21.
16	2/24/2005	Added error 20.
15	5/2004	Revised error 18.
14	10/2003	Added error 19.
13	9/2003	Added error 18.
12	3/2003	Added error 17.
11	10/2002	Updated projected solution for error 8. Revised error 16.
10	2/2002	Added errors 15 and 16.
0–9	—	Earlier releases of document.

Table 2 describes the devices to which the errata in this document apply and provides a cross-reference to match the revision code in the processor version register to the revision level marked on the part.

Table 2. Revision Level to Part Marking Cross-Reference

MPC7410 Revision Level	Part Marking	Processor Version Register
1.0	A	800C 1100
1.1	B	800C 1101
1.2	C	800C 1102
1.3	D	800C 1103
1.4	E	800C 1104

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which it applies. A ‘Y’ entry indicates the erratum applies to a particular revision level, while a ‘—’ entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Number	Name	Projected Solution	Present in Version:				
			1.0	1.1	1.2	1.3	1.4
1	Cache-inhibited instruction fetches that hit in the L2 direct-mapped space may hang the processor	Instructions in the L2 direct-mapped memory space may not be mapped as cache-inhibited.	—	—	—	—	—
2	L2 may cause data corruption in 32-bit data bus mode	Fixed in Rev. 1.1.	Y	—	—	—	—
3	Speculative instruction stream may cause duplicate data cache tags	Fixed in Rev. 1.1.	Y	—	—	—	—
4	Speculative or noncoherent transactions may cause loss of data	Fixed in Rev 1.1.	Y	—	—	—	—
5	Data stream touch may cause duplicate data cache tags	Fixed in Rev 1.1.	Y	—	—	—	—
6	Incorrect condition code on mismatched LWARX/STWCX pair	Fixed in Rev 1.3.	Y	Y	Y	—	—
7	TLBSYNC may hang in the presence of a DST	Fixed in Rev 1.3.	Y	Y	Y	—	—
8	Queueing six transactions to secondary bus may hang the system	In MPC7410 processors with a revision of 1.2 or older, one of the work arounds listed should be used. In MPC7410 processors with a revision of 1.3 or newer, a Data Transaction Queue (DTQ) entry may be reserved for snoops by setting MSSCR1[18:20] = b’001. The MSSCR1 register is accessed as SPR 1015. The work arounds listed may still be used.	Y	Y	Y	Y	Y

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Number	Name	Projected Solution	Present in Version:				
			1.0	1.1	1.2	1.3	1.4
9	Non-compliant to IEEE Standard 1149.1 (JTAG)	Fixed in Rev 1.2.	Y	Y	—	—	—
10	Consecutive interrupts may be nonrecoverable	Fixed in Rev 1.3.	Y	Y	Y	—	—
11	Disabling L2 cache may hang processor	Fixed in Rev 1.3.	Y	Y	Y	—	—
12	Time base or decrementer may lose accuracy in nap mode	Fixed in Rev 1.4.	Y	Y	Y	Y	—
13	IFTT mode does not identify dcbt/dst instructions as data fetches	Fixed in documentation. IFTT documentation now states that IFTT will not differentiate **dcbt** and **dst** instructions and recommends the given work arounds.	Y	Y	Y	Y	Y
14	TAU reports incorrect temperatures	None. This erratum will not be fixed in future revisions of the MPC7410; use of the TAU on the MPC7410 is not supported.	Y	Y	Y	Y	Y
15	Live-lock when in PLL bypass mode	This erratum will not be fixed in future revisions of the MPC7410. This is a processor limitation for this nonstandard mode of operation (PLL bypass mode).	Y	Y	Y	Y	Y
16	CLK_OUT is driven when a high-impedance state is expected	Fixed in documentation.	Y	Y	Y	Y	Y
17	L2 global invalidate may not clear way 0 after L2 usage	This may be fixed in a future revision.	Y	Y	Y	Y	Y
18	Possible OVDD to AVDD noise coupling in the MPC7410 CBGA package	This may be fixed in a future revision of the MPC7410 CBGA package. This has already been fixed in the HCTE package.	Y	Y	Y	Y	Y
19	COP accesses to memory fail at 6.5x bus to core multiplier	None.	Y	Y	Y	Y	Y
20	L2 Data and/or Parity corruption can occur when the core is soft-stopped via COP	None.	Y	Y	Y	Y	Y
21	Data Stream Prefetch to Guarded memory may hang the processor.	None.	Y	Y	Y	Y	Y

Errata No. 1: Cache-inhibited instruction fetches that hit in the L2 direct-mapped space may hang the processor

Overview:

If a cache-inhibited instruction fetch hits in the L2 direct-mapped memory space while the instruction buffer is full, the processor may not make forward progress.

Detailed Description:

Cache-inhibited instruction fetches are usually not allowed to hit in the L2 cache, except when the L2 interface is configured (at least partially) as direct-mapped memory using L2CR2. If part of this direct-mapped memory space is mapped as cache-inhibited and contains instructions, it is possible for these cache-inhibited instruction fetches to hit. When data is returned from the L2 interface to the instruction cache, this data is not written into the instruction cache. Instead it is forwarded only to the instruction buffer. If the instruction buffer is already full, the instruction cache will request the same data again.

The L2 interface contains a queue (the L2DTQ) of transactions that are waiting to access the L2 SRAMs. Normally this queue allows newer read operations to reorder around older, non-conflicting write operations. However, if the instruction cache is continually requesting instructions that the instruction buffers have no room for, L2DTQ may not make forward progress for older, pending write operations. This can cause the L1OPQ and DRLT to back up, which backs up the load/store unit and prevents the instruction buffer from draining.

Projected Impact:

Systems that map cache-inhibited instructions into the L2 direct-mapped memory space

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Do not map instructions in the L2 direct-mapped memory space as cache-inhibited.
2. Set L2DTQ_RDIS (bit 13) in MSSCR1. This prevents the L2DTQ from reordering newer read operations around older write operations.

Projected Solution:

Fixed in documentation. Instructions in the L2 direct-mapped memory space may not be mapped as cache-inhibited.

Errata No. 2: L2 may cause data corruption in 32-bit data bus mode

Overview:

When operating in 32-bit data bus mode, the L2 may return data to the data reload table in the wrong order. This causes register corruption in the destination register of a load as well as data corruption in the L1 data cache. In addition, **dcbst** instructions to odd word addresses (that is, address bit 29 = b‘1’) that hit to modified (M=b‘1’) or dirty (D=b‘1’) in the data cache and hit in the L2 (cache or direct-mapped space) may cause data to be written to the SRAMs in the incorrect order.

Detailed Description:

When operating in 32-bit data bus mode, the L2 interface controller must sometimes zero out the low order L2 address bit (bit 18). This is necessary when bit 29 of the real address is non-zero, and the access is a burst read or write. This will force the L2 interface to start the burst transfer at an aligned double word. This is necessary due to the interface between the L2 bus and the processor core. However, when single beat writes (such as those generated by write-through stores) access the L2 interface, L2 address bit 18 should not be zeroed out. The equation that differentiates single-beat operations from burst operations contains a term that is not fully qualified, and in some cases incorrectly signals a single-beat transaction for a burst read using the L2DTQ bypass path. This causes the L2 interface to return data to the core in the incorrect order, causing register corruption in the destination register of the load as well as data corruption in the L1 data cache.

A similar problem exists for burst write operations that start at odd word addresses (address bit 29 = b‘1’). Zeroing the low order L2 address bit causes the data to be written in the incorrect order to the L2 SRAMs.

Projected Impact:

Systems using the L2 interface in 32-bit data bus mode

Work Around:

Set L2BDIS (bit 12) of MSSCR0 to prevent read transactions from bypassing the L2DTQ, and set DCBST (bit 23) of MSSCR0 to force **dcbst** to be treated like **dcbf**.

Projected Solution:

Fixed in Rev. 1.1.

Errata No. 3: Speculative instruction stream may cause duplicate data cache tags

Overview:

A canceled speculative load on a mispredicted path causes a reload of the data cache in the same cycle that a store to the same cache line accesses the cache, creating duplicate data cache tags.

Detailed Description:

A store to cache line A is dispatched, becomes the oldest instruction in the machine, reads data, and ends up in the completed store queue. This store is followed by a conditional branch that is mispredicted taken. The instruction at the target of the mispredicted branch is a load to non-overlapping bytes of cache line A. The load correctly bypasses the store, misses in the cache, and allocates a data reload table entry. The reload request from the load hits in the L2.

In cycle 0, the L2 completely returns the last beat of data for the speculative load request, and the data is written into the data reload data buffer. The branch is also resolved in this cycle.

In cycle 1, the speculative load is marked as ready to reload the dL1, a speculative cancel is issued due to the mispredicted branch, and the store starts an access at the dL1.

In cycle 2, the reload for the speculative load starts an access at the dL1. The speculative cancel in the previous cycle causes the needs_reload bit for the speculative load data reload table entry to be reset. The store has progressed to the second stage of the dL1 pipeline and checks all reload table entries for an address collision. Because the needs_reload bit for the speculative reload has been reset, the address collision test fails and the store allocates a new data reload table entry.

In cycle 3, the second half of the speculative reload finishes. Cache line A is now valid in the cache and in the data reload table.

A subsequent access to the same cache line will identify the data reload table or data cache duplicate tag.

Projected Impact:

May be present in any system.

Work Arounds:

Any of the following will serve as a work around to this erratum:

1. Set the L1OPQ_SIZE bits (bits 7–8) in MSSCR1 to ‘10.’ Note that this will slow down store accesses to the DL1 to a maximum of one every other cycle.
2. Set the L1WDB_SIZE bits (bits 9–10) in MSSCR1 to ‘10.’ Note that this will slow down store accesses to the DL1 to a maximum of one every other cycle. Because L1 write data buffer entries remain active for a longer period of time than L1 OP Queue entries, this work around is expected to result in lower performance
3. Set the DRLT_SIZE bits (bits 3–4) in MSSCR1 to ‘10.’ Note that this will slow down load and store accesses to the DL1 to a maximum of one every other cycle. This work around is expected to result in the lowest performance of the three options.

Projected Solution:

Fixed in Rev. 1.1.

Errata No. 4: Speculative or noncoherent transactions may cause loss of data

Overview:

Speculative transactions or coherency transactions mapped to noncoherent cacheable space may be dropped in MPX bus mode, leading to an undesirable processor state. The problem does not occur in 60x mode.

Detailed Description:

Transactions that are canceled after being written into the L2 miss queue (L2MQ) of the bus interface unit (BIU) may leave the L2MQ in an undesirable state. Transaction cancellations can occur in one of two ways: in the speculative cancellation case, the transactions are loads from two separate mispredicted branches and in the coherency cancellation case, the transactions are consecutive stores that are converted to coherency-only KILL transactions due to store miss merging.

The sequence and timing of events needed to sensitize the errata are quite complicated and rely on two critical timing windows.

A transaction is written into the first entry of the L2MQ. Two more transactions are written into the second and third L2MQ entries and are subsequently canceled, leaving two empty holes in the L2MQ. One more transaction is written into the fourth entry of the L2MQ. The first transaction then starts and completes an address tenure.

In the first of the two critical timing windows, the fourth transaction starts an address tenure, is canceled, and is eliminated from the queue before completing the address tenure. $\overline{\text{ARTRY}}$ is asserted for this address tenure that was just eliminated from the L2MQ. The queue is now in an undesirable state.

In the second of the two critical timing windows, another transaction is dispatched to the L2MQ during the last core cycle of the $\overline{\text{ARTRY}}$ bus cycle indicated above. This is allocated the L2MQ entry previously allocated by the first transaction. Due to the state of the L2MQ at the time, this transaction will not immediately make a bus request. Any further transactions that are queued to the L2MQ may go to the system bus out of program order or they may hang the processor.

Projected Impact:

May be present in MPX bus systems.

Work Arounds:

Any of the following will serve as a work around to this erratum:

1. Set the speculative disable bit (bit 22) in HID0 to ‘1.’ This will disable data cache and instruction cache speculation. Also, set the non-global broadcast enable bit (bit 22) in MSSCR0 to ‘1.’ This will force all noncoherent transactions to the system data bus.
2. Set the L2MQ_SIZE bit (bit 14) in MSSCR1 to ‘1.’ This will limit the L2 miss queue to one entry and avoid the scenario described above.
3. Run the processor in 60x mode. The problem is not present in 60x mode.

Projected Solution:

Fixed in Rev 1.1.

Errata No. 5: Data stream touch may cause duplicate data cache tags

Overview:

A data stream touch instruction that accesses the data cache in the same cycle that a canceled speculative load causes a reload of the same line to the data cache may create duplicate data cache tags.

Detailed Description:

A data stream touch (DST, DSTT) or data stream touch for store (DSTST, DSTSTT) instruction to cache line A is dispatched and sets up a data stream engine. This data stream touch is followed by a conditional branch that is mispredicted taken. The instruction at the target of the mispredicted branch is a load to non-overlapping bytes of cache line A. The load misses in the cache and allocates a data reload table entry. The reload request from the load hits in the L2.

In cycle 0, the L2 completely returns the last beat of data for the speculative load request, and the data is written into the data reload data buffer. The branch is also resolved in this cycle.

In cycle 1, the speculative load is marked as ready to reload the dL1, a speculative cancel is issued due to the mispredicted branch, and the data stream touch starts an access at the dL1.

In cycle 2, the reload for the speculative load starts an access at the dL1. The speculative cancel in the previous cycle causes the needs_reload bit for the speculative load data reload table entry to be reset. At this point, the data stream touch instruction has progressed to the second stage of the dL1 pipeline and checks all reload table entries for an address collision. Because the needs_reload bit for the speculative reload has been reset, the address collision test fails, and the data stream touch allocates a new data reload table entry.

In cycle 3, the second half of the speculative reload finishes. Cache line A is now valid in the cache and in the data reload table.

A subsequent access to the same cache line will identify the data reload table or data cache duplicate tag.

Projected Impact:

May be present in any system that uses the DST(T)/DSTST(T) instructions.

Work Arounds:

Any of the following will serve as a work around to this erratum:

1. Set the NOPDST bit (bit 30) in HID0 to ‘1.’ This will effectively no-op all DST(T)/DSTST(T) instructions.
2. Set the DRLT_SIZE bits (bits 3-4) in MSSCR1 to ‘01.’ This will set the data reload table to two entries. The data stream engine will only access the cache if the data reload table has both entries free.
3. Identify the problem by setting the EIEC bit (bit 13) in HID0 to ‘1’ and enabling machine check interrupts. Schedule a hardware flush of the cache on identification of the problem and continue. This solution may only be appropriate in systems where there is no snooping traffic to the affected cache line.

Projected Solution:

Fixed in Rev 1.1.

Errata No. 6: Incorrect condition code on mismatched LWARX/STWCX pair

Overview:

A STWCX instruction may be performed without setting the condition code if the store hits in the L2, and the LWARX instruction that set the reservation is to another coherency granule.

Detailed Description:

The errata requires that cache line A be resident in the L2 but not the data L1. This can happen if the line was cast out of the data L1 into the L2 for pure data accesses or if the line was loaded into the L2 at the time of the instruction fetch due to instruction/data cache line sharing.

First, a LWARX instruction sets a reservation to cache line B.

Next, a STWCX instruction to cache line A misses in the L1 and accesses the L2.

Then, a snoop RESERVATION KILL (rwitm, rwitm atomic, rclaim, wr w/kill, wr w/flush, wr w/flush atomic, kill) transaction that matches the reservation address (cache line B) kills the reservation.

If the above conditions are met, there is a one cycle window in which the STWCX can successfully perform the store and yet report an unsuccessful condition code. Use of mismatched LWARX/STWCX pairs is highly discouraged.

Projected Impact:

May be present in any system that uses mismatched LWARX/STWCX address pairs.

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Avoid STWCX instructions that do not match the current reservation address or force a DCBF to the STWCX address before a STWCX that does not match the current reservation address.
2. Turn off the L2. The problem will not occur if the L2 is disabled.

Projected Solution:

Fixed in Rev 1.3.

Errata No. 7: TLBSYNC may hang in the presence of a DST

Overview:

The MPC7410 may not make forward progress if a DST instruction has caused an MMU tablewalk, that MMU tablewalk was marked by a TLBIE instruction, and a TLBSYNC instruction is pipelined the cycle after the MMU tablewalk accesses the dL1 cache.

Detailed Description:

The errata requires that a DST engine be active during a TLBSYNC instruction. The DST engine must also have been activated in the same context as the TLBSYNC instruction, meaning that it was initiated while the processor was in the privileged state.

First, the DST engine makes a request for an address that initiates an MMU tablewalk.

Then, a TLBIE instruction marks the MMU tablewalk instruction because of a potential change in translation. A mark will identify all memory requests that were translated before the TLBIE instruction was executed. Because all TLBIE instructions are snooped from the processor bus regardless of the processor that initiated the transaction, the source of the TLBIE instruction is not relevant.

Finally, a TLBSYNC instruction must follow the MMU tablewalk back-to-back in the pipeline. Because the TLBSYNC is always high priority in the L1OPQ, this opens a one-cycle window where the TLBSYNC may pass the marked MMU tablewalk at arbitration to the L2. Once the TLBSYNC has passed the marked tablewalk, it will progress to the system bus. This processor will then infinitely self-$\overline{\text{ARTRY}}$ the TLBSYNC instruction because of the marked MMU tablewalk instruction. The system will be in a live-lock condition.

Projected Impact:

Any system that has an active DST engine while executing a TLBSYNC instruction in a privileged context

Work Around:

Insert a DSSALL instruction before a TLBSYNC instruction.

Projected Solution:

Fixed in Rev 1.3.

Errata No. 8: Queueing six transactions to secondary bus may hang the system

Overview:

Queueing six transactions from a single MPC7410 to a secondary bus may use all data transaction queue resources and hang the system if forward progress cannot be made by allowing the MPC7410 to complete at least one outstanding transaction.

Detailed Description:

This errata details a scenario where a system may not make forward progress. The errata requires a series of queued transactions to a secondary bus. For this situation to occur, the MPC7410 must also snoop a transaction from another device connected to a secondary bus. The errata can affect both uniprocessor and multiprocessor implementations.

First, the MPC7410 initiates, and the system chipset logic accepts six transactions to a secondary bus. These transactions do not need to be consecutive, although all six transactions need to have completed their address tenure without completing a data tenure or a tenure on the secondary bus. The six transactions fill up the six-entry data transaction queue (DTQ) in the MPC7410, making it impossible for the processor to issue or snoop any more transactions until at least one of its DTQ entries is freed by the completion of an outstanding (pipelined) data tenure.

Then, the system chipset requires that a memory transaction that was already accepted or posted to the secondary bus make an address transaction on the processor system bus. Because the DTQ of the processor is full, there is no room for a data tenure should the MPC7410 need to push modified data in response to the snooped transaction. The MPC7410 will assert $\overline{\text{ARTRY}}$ to the snooped transaction until one of the queued transactions finishes a data tenure, allowing the processor to complete the snoop copy back.

If the system logic is unable to allow a data tenure for any of the queued transactions from the processor until the snooped write has completed, neither the MPC7410 nor the secondary bus can proceed until the other is complete. This could result in a deadlock scenario.

Projected Impact:

Any system that allows six outstanding transactions from a single processor and that has a secondary bus with the characteristics detailed above. While few system logic designs would have such characteristics, those that cannot retry transactions from devices connected to the secondary bus might show susceptibility to this errata.

Another possible design that might be affected would have an arbitration scheme such that the device the MPC7410 is snooping is always given higher priority than the target device(s) of the MPC7410 queued transactions.

Finally, any system in which a snooped transaction will prevent the execution of any other transactions until it completes can also be impacted.

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Limit the number of outstanding transactions to a secondary bus from a single processor to five in system logic.
2. Mark the memory space on the secondary bus as guarded and avoid DST(ST)(T) and LMW instructions.

Projected Solution:

In MPC7410 processors with a revision of 1.2 or older, one of the work arounds listed above should be used.

In MPC7410 processors with a revision of 1.3 or newer, a Data Transaction Queue (DTQ) entry may be reserved for snoops by setting MSSCR1[18:20] = b’001. The MSSCR1 register is accessed as SPR 1015. The work arounds listed above may still be used.

Errata No. 9: Non-compliant to IEEE Standard 1149.1 (JTAG)

Overview:

Control of many pins under IEEE Standard 1149.1 (JTAG), including TDO, is disrupted when the processor is configured in the PLL OFF state.

Detailed Description:

When the PLL_CFG(0:3) pins are placed into the PLL OFF state (PLL_CFG = ‘1111’b), not all pins can be guaranteed to transition from a ‘0’ to a ‘1’ at all specified voltage levels. The part is non-compliant to IEEE Standard 1149.1 in this respect only.

The PLL_CFG pins should have static formats in any test environment. Do not use return-to-high or surround-by-complement formats for these pins since this can also cause the unintentional PLL OFF state to occur.

Projected Impact:

Any system that places the PLL_CFG(0:3) pins in the PLL OFF state while doing IEEE Standard 1149.1 interconnect testing

Work Around:

Designate any single PLL_CFG pin as a compliance enable pin. The MPC7410.R1B BSDL designates PLL_CFG1 pin to be a compliance pin. The boundary latch for this pin is declared an internal latch since a compliance pin cannot have a boundary latch. Constraining the PLL_CFG1 pin to a ‘0’ will prevent the PLL OFF state from triggering the non-compliant state.

Projected Solution:

Fixed in Rev 1.2.

Errata No. 10: Consecutive interrupts may be nonrecoverable

Overview:

Consecutive performance monitor, decrementer, or thermal management interrupts may become nonrecoverable.

Detailed Description:

The performance monitor, decrementer, and thermal management interrupts are all gated by the external interrupt enable bit of the MSR (bit 16). Once processing of one of these interrupts is begun, the hardware will reset MSR(EE) = 0 to prevent taking a second interrupt during the processing of the first interrupt. The process of resetting the MSR(EE) bit takes two processor clocks.

The priorities for these interrupts are as follows:

PFMPerformance monitor interrupt

3. DEC Decrementer interrupt
4. TMI Thermal management interrupt

If a higher priority interrupt request occurs during the first cycle of lower priority interrupt processing, the machine may appear to start processing the lower priority interrupt and then switch to processing the higher priority interrupt. In addition, the SRR0 register will contain the address of the lower priority interrupt vector, and the SRR1 register will contain the MSR values normally seen during interrupt processing, including MSR(EE) = 0 and MSR(RI) = 0. This results in the loss of the last instruction address, effectively making the interrupt nonrecoverable.

Note that other interrupts maskable by the MSR(EE) bit, including the system management interrupt and the external interrupt, are not affected by this errata.

Projected Impact:

Any system that enables a combination of performance monitor, decrementer, or thermal management interrupts at the same time.

Work Around:

Avoid enabling any combination of performance monitor, decrementer, or thermal management interrupts at the same time.

Projected Solution:

Fixed in Rev 1.3.

Errata No. 11: Disabling L2 cache may hang processor

Overview:

The MPC7410 may hang if the L2 cache is disabled during an outstanding instruction fetch.

Detailed Description:

The problem centers around the interaction between the instruction cache and the L2 cache as the L2 cache is disabled. The scenario is as follows:

1. An instruction fetch misses in the instruction cache and allocates a reload table entry.
2. When the instructions return from the BIU, they are forwarded around the instruction cache and dispatched, as well as written, to the IRLDQ.
3. One of these instructions is a **mtspr** targeting the L2CR. This instruction disables the L2.
4. When all beats of data return to the IRLDQ, the IRLT arbitrates to reload the L2. Because the L2 is now disabled, it does not expect reload requests from the IRLT.
5. The unexpected reload request is mishandled by the L2 and passed to the BIU as an instruction fetch miss.

Projected Impact:

Any system that executes code to disable the L2 cache.

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Mark the code that disables the L2 cache as cache-inhibited.
2. Preload the code that disables the L2 cache into the instruction cache before execution. This requires the code be structured in such a way that the instruction fetch be completed before the **mtspr** is executed. For example:

```
offset   instruction

1C       branch to offset 3C              /* preload mtspr without executing */
20       mtspr 1017, r?                   /* disable L2 */
24       sync
28       isync
2C       branch to offset 60
3C       branch to offset 40

40       sync
44       isync
48       branch to offset 20

60       next sequential fetch
```

Projected Solution:

Fixed in Rev 1.3.

Errata No. 12: Time base or decrementer may lose accuracy in nap mode

Overview:

The time base counter or decrementer may lose accuracy when transitioning from the nap power saving mode.

Detailed Description:

The nap power saving mode is entered when the processor asserts $\overline{\text{QREQ}}$ and the system logic responds with $\overline{\text{QACK}}$. The time base counter and decrementer switch from being clocked by the normal gclk distribution tree to the pll gclk2 distribution tree. The normal gclk distribution tree is shut off to conserve power.

When the nap power saving mode is exited, the gclk distribution tree is re-started, and the time base counter and decrementer switch back to being clocked by the normal gclk distribution tree. The nap power saving mode can be exited in one of the following ways:

1. Deasserting $\overline{\text{QACK}}$ to allow the processor to snoop
2. Asserting $\overline{\text{INT}}$, $\overline{\text{SMI}}$, or $\overline{\text{MCP}}$
3. Taking a decrementer, performance monitor, or thermal management interrupt
4. Asserting hard or soft reset

If any of the first three categories of events happens in a six processor-clock cycle window after entering nap state, the time base counter and decrementer may miss clock ticks for the remainder of the six processor-clock cycle window. This will not cause harm to the operation of the processor but may cause a loss of accuracy in the time base counter or decrementer.

Projected Impact:

This may be an issue for any system that uses the nap power saving mode and requires absolute accuracy from the time base counter or decrementer. For instance, if a time base is maintained across two processors in a multiprocessing system, there can be scenarios where one processor may appear to drift with respect to the other processor.

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Avoid the use of nap mode.
2. Avoid using the time base counter or decrementer for highly accurate measurements.

Projected Solution:

Fixed in Rev 1.4.

Errata No. 13: IFTT mode does not identify dcbt/dst instructions as data fetches

Overview:

The instruction fetch transaction type (IFTT) encoding differentiation mode does not correctly identify **dcbt** or **dst** instructions as data fetches.

Detailed Description:

When the HID0[IFTT] bit is set, the MPC7410 processor can differentiate instruction fetches from data fetches using their TT encoding on the system bus. Data fetches are identified as READ ATOMIC (TT = 11010), while instruction fetches are identified as READ (TT = 01010).

Touch-for-load instructions including **dcbt** and **dst** are identified as READ (TT = 01010) regardless of the setting of the HID0[IFTT] bit. These instructions will perform as expected from a processor perspective regardless of their TT encoding on the system bus.

Projected Impact:

This can be an issue for any system that depends on IFTT to differentiate instruction from data fetches.

Work Arounds:

Any of the following will serve as a work around to this erratum:

1. Replace **dcbt** instructions with **dcbtst** and replace **dst** instructions with **dstst**. Note that this could have the detrimental effect of lowering performance because loads cannot fold to touch-for-store instructions. See Section 3.5.3.2 of the *MPC7410 RISC Microprocessor User's Manual* for a caution on touch-for-store instructions.
2. Set HID0[NOPTI] and HID0[NOPDST] to no-op all touch instructions.
3. Remove **dcbt** and **dst** instructions from the code.

Projected Solution:

Fixed in documentation. IFTT documentation now states that IFTT will not differentiate **dcbt** and **dst** instructions and recommends the above work arounds.

Errata No. 14: TAU reports incorrect temperatures

Overview:

The thermal assist unit (TAU) reports temperatures between 35 to 55 degrees lower than expected.

Detailed Description:

This erratum only affects customers using the TAU. If a trip temperature is programmed into the sensor's control registers, the output interrupt is never received even if temperatures exceed the expected set point by up to 55 degrees (even after calibration). A control application is not alerted of excessive temperatures, which can lead to damage of the part.

Projected Impact:

Programmed trip temperatures do not trigger output interrupts even if temperatures exceed the expected setpoint by up to 55 degrees.

Work Arounds:

None.

Projected Solution:

None. This erratum will not be fixed in future revisions of the MPC7410; use of the TAU on the MPC7410 is not supported.

Errata No. 15: Live-lock when in PLL bypass mode

Overview:

When running in PLL bypass mode, when a self-snoopable transaction (TLBIE, ICBI, SYNC, TLBSYNC) is self-$\overline{\text{ARTRY}}$ed for any reason, it will be self-$\overline{\text{ARTRY}}$ed forever.

Detailed Description:

The MPC7410 was designed to perform all snooping operations in core to bus clock ratios of 2:1 or greater. These two cycles are needed to collect all the necessary snoop responses from around the chip and drive the $\overline{\text{ARTRY}}$, $\overline{\text{SHD0}}$, and $\overline{\text{SHD1}}$ pins as necessary.

In PLL bypass mode, the core and system clocks are matched, which results in a single internal snoop cycle per system clock. The problem centers around a hold latch in the $\overline{\text{ARTRY}}$ logic that is normally reset in the second internal snoop cycle. Once this latch is set for a self-snooped transaction, there is no way to clear it.

The value in this hold latch will then propagate to the pins, causing a false single-cycle assertion on the $\overline{\text{ARTRY}}$ pin. The part will then react as if it had asserted $\overline{\text{ARTRY}}$ again, causing an infinite cycle of $\overline{\text{ARTRY}}$ assertions for the self-snooped transaction. Note that this false single-cycle assertion of $\overline{\text{ARTRY}}$ could result in problems for other processors in an MP system.

This problem can also occur with the $\overline{\text{SHD0}}$ and $\overline{\text{SHD1}}$ pins, although it is not expected to result in system issues.

Projected Impact:

This can be an issue for any system that uses the nonstandard, PLL bypass mode of operation and self-snoopable transactions (TLBIE, ICBI, SYNC, TLBSYNC) that may be self-$\overline{\text{ARTRY}}$ed.

In PLL bypass mode, a self-snoopable transaction that is self-$\overline{\text{ARTRY}}$ed for any reason will be self-$\overline{\text{ARTRY}}$ed forever. In addition, the false single-cycle assertion of $\overline{\text{ARTRY}}$ caused by this issue could also result in problems for other processors in an MP system.

Work Around:

Avoid using TLBIE, ICBI, SYNC, TLBSYNC in PLL bypass mode, or implement a synchronization routine such that these instructions never have a reason to be retried.

Projected Solution:

This erratum will not be fixed in future revisions of the MPC7410. This is a processor limitation for this nonstandard mode of operation (PLL bypass mode).

Errata No. 16: CLK_OUT is driven when a high-impedance state is expected

Overview:

CLK_OUT is driven with engineering debug information when HID0[ECLK] = 0 and HID0[BCLK] = 0. The expected state of CLK_OUT in this mode is high impedance.

CLK_OUT is driven with engineering debug information when HID0[ECLK] = 0 and HID0[BCLK] = 1. The expected state of CLK_OUT in this mode is SYSCLK/2.

Detailed Description:

The CLK_OUT pin provides a PLL clock output for PLL testing and monitoring. The configuration of the HID0[ECLK] and HID0[BCLK] bits determines the state of the CLK_OUT signal. When HID0[ECLK] = 0 and HID0[BCLK] = 0, the expected state of CLK_OUT is high impedance; however, in this mode the CLK_OUT signal always outputs engineering debug information. When HID0[ECLK] = 0 and HID0[BCLK] = 1, the expected state of CLK_OUT is SYSCLK/2; however, in this mode the CLK_OUT signal always outputs engineering debug information.

Projected Impact:

For systems concerned with EM emissions, this signal should be terminated.

Work Arounds:

None

Projected Solution:

Fixed in documentation.

Errata No. 17: L2 global invalidate may not clear way 0 after L2 usage

Overview:

The L2 cache global invalidate function may not clear way 0 of the L2 cache after L2 cache usage. The L2 cache global invalidate function operates correctly if it is performed before the L2 cache has been enabled for the first time.

Detailed Description:

The MPC7410 supports global (not flash) invalidation of the L2 cache through the L2CR[L2I] parameter. Setting L2CR[L2I] causes a global invalidation of the L2 cache. A global invalidation is performed by automatically sequencing through the L2 cache tags and clearing all bits of the tag (tag data bits, tag status bits, and FIFO bit) for each of the two ways (way 0 and way 1).

When sequencing through the L2 cache tags for way 0 and way 1, the L2 global invalidate way select is masked by an internal latch. Depending on the state of the internal latch, the L2 global invalidate function may not clear the tags for way 0 of the L2 cache. The L2 global invalidate function will always correctly clear way 1 of the L2 cache.

The state of this internal latch is always correct after the assertion of $\overline{\text{HRESET}}$. Setting L2CR[L2I] after an $\overline{\text{HRESET}}$ assertion, but prior to enabling the L2 cache, will correctly invalidate all of the L2 tags. Using the L2 global invalidate after enabling the L2 may not invalidate all of the L2 tags.

Projected Impact:

This can be an issue for any system that uses the L2 global invalidate function after the L2 cache has been enabled for the first time.

Work Arounds:

Either of the following will serve as a work around to this erratum:

1. Limit use of the L2 global invalidate function to before the L2 cache has been enabled for the first time.
2. Use L2 cache hardware flush. Using L2CR[L2HWF] will correctly invalidate all of the L2 tags, but this feature will maintain coherency. The system should be able to handle possible castouts of modified data. Once the L2 cache has been enabled and used, there is no way to guarantee the invalidation of the L2 tags without maintaining coherency.

Projected Solution:

This may be fixed in a future revision.

Errata No. 18: Possible OV_{DD} to AV_{DD} noise coupling in the MPC7410 CBGA package

Overview:

On systems that use the MPC7410 CBGA package, it is possible for noise on OV_{DD} to couple into the PLL supply voltage, AV_{DD}, internal to the package.

Detailed Description:

On systems that implement the PLL filter at the AV_{DD} input pin (as shown in Figure 1-1), if the system exhibits a high amount of noise on OV_{DD}, it is possible for the noise on OV_{DD} to couple into the PLL supply voltage (AV_{DD}) internal to the MPC7410 microprocessor through the CBGA package. This issue is unique only to the AV_{DD} input of the MPC7410 CBGA package and is more likely at high system bus frequency and heavy bus traffic, that is, high likelihood of worse case I/O buffer switching. Note that this error has no impact on the $L2AV_{DD}$ input of the MPC7410. System designs should still implement the filter show in Figure 1-1 on the $L2AV_{DD}$ input.

When this high frequency OV_{DD} noise is coupled onto AV_{DD} it can adversely affect the switching threshold of the internal clock distribution buffer and cause it to fail to switch, resulting in a clock glitch visible on CLK_OUT. The PLL remains locked and attempts to adjust phase for the missing or malformed clock pulse, inducing minor jitter, but quickly relocks when the synchronous clock is restored on the next pulse.

It is possible to monitor the CLK_OUT output of the MPC7410 for disturbances by setting HID0[BCLK, ECLK] = b'01 to output the core frequency after $\overline{\text{HRESET}}$ is deasserted. A disturbance on CLK_OUT indicates a disturbance on the internal core clock of the processor, which can affect any clocked logic in the device. Note that no disturbance was found on the L2 output clocks (L2CLK_OUTA, L2CLK_OUTB, L2SYNC_OUT).

It is quite possible for systems with the MPC7410 CBGA package that implement the original filter, shown in Figure 1-1, to never experience this error. The root cause of the error is a combination of internal package sensitivity to OV_{DD} noise due to I/O switching and customer board-dependent sensitivity. Customer board-dependent sensitivities due to board design include: trace length/routing, noise on power supplies, and loading from other devices.

This error has been seen on very few customer systems. Measurements on those systems show system errors coincident with noise measurements at the AV_{DD} pin around 500 mV peak-to-peak (600 mV max) with the original AV_{DD} filter while running heavy system and L2 bus traffic. With the new filter, noise at the AV_{DD} pin increased to around 800 mV. This AV_{DD} noise was not visible on the V_{DD} side of the AV_{DD} filter circuit.

The recommended AV_{DD} filter, shown in Figure 1-1, was designed to keep high-frequency noise from entering the chip from the V_{DD} supply but actually works against suppressing the high frequency OV_{DD} noise coupled onto AV_{DD} internally. By depopulating the filter capacitors, the external supply can meet the instantaneous current requirements of the PLL circuitry and improve noise margin. If the filter capacitors are depopulated, the series resistor should be increased to 51 Ω. This increased resistance provides better damping for the inevitable inductance in the AV_{DD} path from board to package to silicon.

It is the recommendation of Freescale that systems that exhibit a high amount of noise on OV_{DD} implement the AV_{DD} filter fix shown in Figure 1-2. Both filter options should be qualified by the customer, and the solution providing the most robust margin should be implemented. It is also recommended that the pads for the removed capacitors be left in the design to provide for the possible re-introduction of the filter in Figure 1-1. This would be necessary in order to migrate between the two filters, or in the case that there is a planned transition to the HCTE package of the MPC7410. Designs using the HCTE package of the MPC7410 should implement the filter shown in Figure 1-1.

Projected Impact:

May be present in any system using the MPC7410 CBGA package.

Work Around:

Implement the AV_{DD} filter shown in Figure 1-2.

Projected Solution:

This may be fixed in a future revision of the MPC7410 CBGA package. This has already been fixed in the HCTE package.

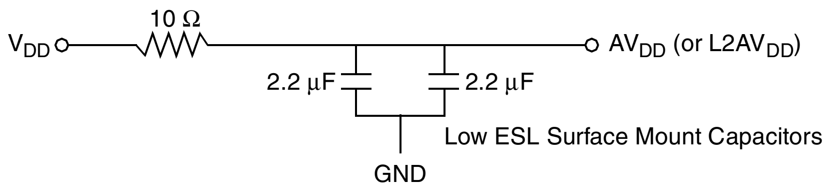

Figure 1-1. Original PLL Power Supply Filter Circuit

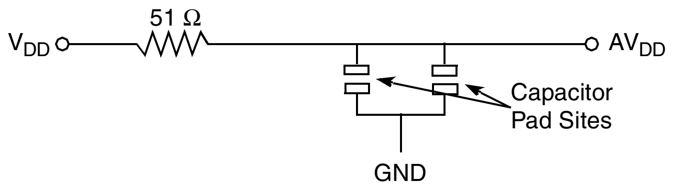

Figure 1-2. PLL Power Supply Filter Circuit Work Around

Errata No. 19: COP accesses to memory fail at 6.5x bus to core multiplier

Overview:

COP accesses to memory return undefined data when the bus to core multiplier is set to the 6.5x configuration.

Detailed Description:

COP accesses to memory return undefined data when the bus to core multiplier is set to the 6.5x configuration.

Functional operation is not impacted by this issue. The 6.5x configuration is enabled when the PLL_CFG(0:3) pins are set to b‘0101.

Projected Impact:

This problem will occur in debug situations where the COP is used to access memory. Functional operation is not impacted by this issue.

Work Around:

Avoid using the 6.5x bus to core multiplier when accessing memory through the COP.

Projected Solution:

None.

Errata No. 20: L2 Data and/or Parity corruption can occur when the core is soft-stopped via COP

Overview:

L2 data and/or parity can be corrupted when the core is soft-stopped via COP if the L2 is enabled.

Detailed Description:

When the core is soft-stopped via COP while the L2 is enabled, the following may occur:

1. If the L2CR[L2PE] bit is cleared, L2 data may be corrupted in the L2 SRAMs. When the core is allowed to resume via COP, the core may read and use corrupted data from the L2 SRAMs.

2. If the L2CR[L2PE] bit is set, then both L2 data and parity may be corrupted in the L2 SRAMs. When the core is allowed to resume via COP, the core may read the corrupted data and parity back from the L2 SRAMs. With the L2CR[L2PE] bit set, the result of read will be the core taking the Machine Check exception if the MSR[ME] bit is set. If the MSR[ME] bit is clear, the core will enter the checkstop state. If the Machine Check exception is taken for bad L2 parity, the error flag in SRR1[11/L2DP] will be set.

Emulator tools use soft-stop regularly for debugging purposes. The emulator tools use soft-stop in the following cases:

1. To stop the execution of code.
2. To single step code.
3. To set up an instruction breakpoint in code.
4. To set up a data breakpoint in code.

All of the above actions cause the processor to go into a stop mode. If the L2 is enabled when the processor goes into a stop mode, the L2 data and/or parity may be corrupted.

Projected Impact:

This error may occur on any system that utilizes the backside L2 interface when the core is soft-stopped via COP.

Work Around:

None.

Projected Solution:

None.

Errata No. 21: Data Stream Prefetch to Guarded memory may hang the processor

Overview:

Data Stream Prefetch using dst or dstst instructions to Guarded memory (WIMG=xxx1) may hang the processor.

Detailed Description:

The Data Stream Prefetch engines and the dst and dstst instructions are provided to allow for software directed prefetch of regular data structures in memory.

These engines can generate memory touch requests for a long time after the original dst or dstst instruction has completed in the machine and exited the completion buffer.

The problem occurs when the parent dst or dstst instruction to Guarded memory is speculative with respect to the other instructions in the machine at the time it is dispatched to the prefetch engine. The engine will begin to generate individual child touch requests to Guarded memory. These child touch requests should be marked as speculative if the parent dst or dstst instruction is still speculative with respect to the other instructions in the machine. Once the parent dst or dstst instruction becomes non-speculative, all outstanding and future children should be marked as non-speculative.

Speculative transactions to Guarded memory will wait in the miss queue and refrain from accessing the system bus until they are marked as non-speculative. If a child touch request misses the transition of its parent from speculative to non-speculative, the child touch may stay speculative and block the miss queue forever, deadlocking the machine.

Projected Impact:

This error may occur on any system that maps dst/dstst instructions to Guarded memory.

Work Around:

1. Avoid dst/dstst instructions to Guarded memory.

OR

2. Set HID0[30] NOPDST to nop dst/dstst instructions.

Note that setting HID0[22] SPD will force all memory transactions to appear as if they are mapped Guarded. If this setting is used, follow Work Around #2 above.

Projected Solution:

None.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

Document Number: MPC7410CE
Rev. 17
06/2005