ARM CoreLink™ Level 2 Cache Controller (L2C-310 or PL310)

r0 releases

Software Developers Errata Notice

ARM®

ARM UAN 0010B (ID102712)

ARM CoreLink Level 2 Cache Controller (L2C-310 or PL310)
Software Developers Errata Notice

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
14 June 2012	A	Non-confidential	First release for r0 releases, limited distribution
20 September 2012	B	Non-confidential	Second release for r0 releases

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Contents
ARM CoreLink Level 2 Cache Controller (L2C-310 or PL310) Software Developers Errata Notice

Chapter 1
Introduction

This chapter introduces the errata notices for ARM CoreLink Level 2 Cache Controller L2C-310 (PL310).

1.1 Scope of this document

This document describes errata categorized by level of severity. Each description includes:

- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

This document describes errata that may impact anyone who is developing software that will run on implementations of this ARM product.

1.2 Categorization of errata

Errata recorded in this document are split into the following levels of severity:

Table 1-1 Categorization of errata

Errata type	Definition
Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A(rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error, or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B(rare)	A significant error, or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

1.3 Errata summary

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the text of the erratum Description, Conditions, Implications or Workaround. Fixed errata are not shown as updated, unless the erratum text has changed.

Table 1-2 Changes in Document v1

Status	ID	Area	Cat	Rare	Summary of erratum
	504326	Prog	B		Prefetching can be done on a wrong address or on a wrong memory area
	540921	Prog	B		Parity must be disabled in exclusive cache configuration
	588369	Prog	B		Clean and Invalidate maintenance operations do not invalidate clean lines
	588371	Prog	B		Potential data corruption when receiving an AXI locked access while treating a prefetch
	769419	Prog	B		No automatic Store Buffer drain, visibility of written data requires an explicit Cache Sync operation
	738415	Prog	B	Rare	A cacheable read with address bits [20:5] equal to 0x0000 can be stalled by non-cacheable read traffic targeting other address region
	501023	Prog	C		Address Filtering register content cannot be read
	754670	Prog	C		A continuous write flow can stall a read targeting the same memory area
	765569	Prog	C		Prefetcher can cross 4KB boundary if offset is programmed with value 23

Chapter 2
Errata Descriptions

This chapter includes the errata descriptions for ARM CoreLink Level 2 Cache Controller L2C-310 (PL310).

2.1 Category A

There are no errata in this category.

2.2 Category A (Rare)

There are no errata in this category.

2.3 Category B

This section describes Category B errata.

2.3.1 (504326) Prefetching can be done on a wrong address or on a wrong memory area

Status

Affects: product PL310 Level-2 Cache Controller

Fault Type: Cat B

Fault Status: Present in: r0p0, Fixed in r1p0. Unchanged in this document.

Description

When the Instruction or Data Prefetch feature is enabled, each PL310 slave port is able to initiate linefills based on the explicit read transactions it receives. This allows the cache controller to issue transactions to L3 in advance of the stream sent by the L1 in order to maximize the L2 hit rate.

Prefetch transactions are initiated a few cycles after the slave port receives a cacheable read and target the next cache line. Furthermore, it is not permitted to cross a 4KB boundary when generating prefetch transactions.

The Prefetch feature is not correctly implemented and prefetch transactions initiated by the slave port(s) can cross a 4KB boundary and target non-cacheable memory.

Conditions

This problem occurs when the following conditions are met:

1. Instruction or Data Prefetch is enabled.
2. In a slave port, a read slot dealing with a read transaction to address A creates a new prefetch transaction (normally targeting address A + 1 line) at the same time as it sends its last data back to L1.
3. Before this prefetch transaction is handled:
   - The slot captures a new read transaction at address B if one was pending on AXI.

     OR
   - The slot increments its address pointer if no transaction is pending on AXI.
4. The prefetch transaction is sent to a master port targeting address B + 1 line with cacheable attributes of transaction B, or address A + 2 lines respectively.

Implications

The implications of this erratum when the conditions above are met are:

- Prefetch can be done to a Non-Cacheable, Device or Strongly Ordered memory area, in the case of the pending transaction B. In this case, the master port deadlocks.
- Transaction to address A + 2 lines may be done on a new 4KB page of memory space, regardless of page attribute settings in L1 MMU.

Workaround

The only workaround is to keep the Instruction and the Data Prefetch features disabled. This is the default behavior.

2.3.2 (540921) Parity must be disabled in exclusive cache configuration

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat B

Fault status: Present in: r0p0, r1p0. Fixed in r2p0. Unchanged in this document.

Description

When exclusive cache configuration is enabled, a read hit in the L2 cache leads to the invalidation of the corresponding cache line. A dirty line must be evicted before being replaced. However, it is still possible to hit in the line until the eviction actually happens.

If parity is implemented and enabled, and if such a second hit happens, the Tag RAM parity bit is incorrectly inverted. This then generates a false parity error when the line is subsequently accessed. This error is propagated to the CPU as an abort or an interrupt.

Conditions

This problem occurs when the following conditions are met:

1. Parity is implemented (RTL option) and enabled (bit [21] of the Auxiliary Control Register).
2. Exclusive cache configuration is enabled (bit [12] of the Auxiliary Control Register).
3. A read access hits in the L2 cache.
4. The cache line (CL) is invalidated (Tag RAM valid bit reset) and its Tag RAM parity bit is inverted.
5. CL is dirty.
6. Another read hits in CL before it gets evicted.
7. CL is invalidated (no effect) and its parity is inverted again.
8. The Tag RAM parity bit does not correspond to the actual parity of the Tag RAM content for CL.
9. A parity error is generated when CL is further accessed, for example for eviction or lookup purpose.

Implications

Exclusive cache configuration is only supported by the ARM Cortex-A9 and ARM11 MPCore processors. Therefore the erratum only affects systems where these processors are attached to PL310. Furthermore, the erratum only affects systems where parity is implemented in PL310. This is an RTL option that needs to be defined before synthesis.

When all conditions above are met, the Tag RAM content is corrupted and the cache line content can no longer be accessed. Moreover, false parity errors are generated and are propagated to the CPU as aborts or interrupts.

Workaround

In systems affected by the erratum, a simple workaround is to make sure that exclusive cache configuration and parity are not enabled at the same time.

2.3.3 (588369) Clean and Invalidate maintenance operations do not invalidate clean lines

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat B

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0. Unchanged in this document.

Description

The PL310 L2 cache controller implements three types of Clean and Invalidate maintenance operations:

- by Physical Address (offset `0x7F0`)
- by Index/Way (`0x7F8`)
- by Way (`0x7FC`).

They are architecturally defined to behave as the execution of a clean operation followed immediately by an invalidate operation, both performing to the same memory location. This functionality is not correctly implemented in PL310 because clean lines are not invalidated as a result of these operations.

Conditions

This problem occurs when the following conditions are met:

1. A Clean&Invalidate by Physical Address, by Index/Way or by Way operation is executed in the PL310 L2 cache controller.
2. The operation targets a cache line, the status of which is valid and clean.
3. The cache line remains valid at the end of the operation.

Implications

This erratum has implications when a Clean and Invalidate operation has been executed for either of the following reasons:

- to guarantee that a specific memory region is not present in the L2 cache
- to guarantee that all or part of the L2 cache is invalidated at the end of a process.

Because of this erratum, valid cache lines might remain in the L2 cache, potentially causing unexpected failures.

Clean and Invalidate operations usually serve two purposes:

- They provide a useful combined function when shutting down the L2 cache, particularly if the cache is disabled.
- They provide a combination of “Clean” and “Invalidate”, which can, in principle, be used for ensuring coherency from inside outwards (clean functionality - any change in the L2 cache is made visible to L3) and from outside inwards (invalidate functionality - any change at L3 is made visible to the L2 cache).

Workaround

In the low power mode case mentioned above, the Clean and Invalidate operation can clearly be replaced by a sequence of “a Clean operation followed by an Invalidate operation”.

In the coherency case, the additional requirement is to make sure that no cache line is modified and marked dirty between the individual clean and invalidate operations. This is guaranteed by temporarily disabling write-back and cache allocation. You can use the following pseudo-code routines as workarounds for each different Clean and Invalidate operation type:

- Clean and Invalidate by PA

  ```
  Disable Write-Back and Cache Linefill (set bits [1:0] of the Debug Control Register)
  Clean by PA (0x7B0)
  Invalidate by PA (0x770)
  Re-enable Write-Back and Cache Linefill (reset bits [1:0] of the Debug Control Register)
  ```

- Clean and Invalidate by Way

  ```
  Disable Write-Back and Cache Linefill (set bits [1:0] of the Debug Control Register)
  loop_label
  Clean by Index/Way (0x7B8)
  Increment Index/Way to cover all targeted ways
  B loop_label if not all indexes/ways are covered
  Invalidate by Way (0x77C) + poll for completion ; in Non-secure world, consider a loop on all
  targeted ways (see erratum 539766)
  Re-enable Write-Back and Cache Linefill (reset bits [1:0] of the Debug Control Register)
  ```

- Clean and Invalidate by Index/Way

  PL310 does not implement an Invalidate by Index/Way operation. As a consequence, the workaround is to replace the by Index/Way operation with a by Way operation, as described in the previous bullet.

If PL310 is integrated in a system where TrustZone is implemented, the workarounds described above and involving the Debug Control Register involve calling the Secure world to change the values in the Debug Control Register, because this register is only accessible in Secure world.

2.3.4 (588371) Potential data corruption when receiving an AXI locked access while treating a prefetch

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat B

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0. Unchanged in this document.

Description

In the PL310 L2 cache controller, address hazards can occur between read transactions in the slave port(s) and writes in the store buffer (STB). When such a hazard occurs, the read must be stalled until the write completes. The implementation of this behavior is not fully correct if the read is a prefetch and if an AXI locked access is received by one slave port, causing a potential corruption of the cache line targeted by the prefetch and the write.

Conditions

This problem occurs when the following conditions are met:

1. Instruction or Data Prefetch is enabled (bits [29:28] of the Auxiliary Control Register).
2. A prefetch to a cacheable memory location is initiated by a read transaction in one PL310 slave port.
3. There is an address hazard between this prefetch and a slot of the STB.
4. Because of this hazard, the prefetch is stalled while the STB slot is asked to drain.
5. An AXI locked access is received by one PL310 slave port.
6. This AXI locked access unstalls the prefetch, which can then proceed.
7. The memory location targeted by the two requests is not in the L2 cache.

Implications

The race between the prefetch and the write resulting from the conditions listed above can lead to:

- a double allocation of the cache line targeted by the prefetch and the write

  OR

- an address hazard between a read and a write on the PL310 AXI master interfaces.

This results in unpredictable behavior and in the corruption of the memory location targeted by the prefetch and the write.

Workaround

ARM processors can only generate an AXI locked access when a SWP instruction is executed. This particular instruction is deprecated. Do not use it in software dedicated to ARM processors.

If PL310 is driven by AXI masters consisting of either ARM CPUs with software free of SWP instruction or other masters not able to issue AXI locked accesses, there is no need for a workaround. If necessary, a workaround is to keep prefetch disabled (bits [29:28] of the Auxiliary Control Register), which is the default behavior.

2.3.5 (769419) No automatic Store Buffer drain, visibility of written data requires an explicit Cache Sync operation

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat B

Fault Status: Present in: r0p0, r1p0, r2p0, r3p0, r3p1. Fixed in r3p2. Unchanged in this document.

Description

Before revision r3p2, the PL310 Store Buffer did not have any automatic draining mechanism. Any data written to one of these devices would consequently remain in the buffer, invisible to the rest of the system.

If an L3 external agent keeps on polling this memory location, waiting to see the update of the written data to make any further progress, then a system livelock might happen.

Conditions

The erratum can only happen on Normal Memory regions.

The following scenario is an example that can exhibit the erratum, where an L3 agent might loop infinitely waiting for the notification from the CPU for an unbounded amount of time:

- An L3 agent is waiting for notification from CPU before making progress
- CPU attached to PL310 issues the necessary notification using a write access, which stays in PL310 store buffer
- No additional activity forcing the store buffer to drain is received by PL310.

Implications

Because of the erratum, a livelock situation might occur in the system.

Workaround

If a write access needs to be made visible to an L3 external agent, the workaround for this erratum consists of using a Cache Sync operation in order to force the PL310 Store Buffer to drain. This is illustrated in the following pseudo-code sequence:

```
STR // to be made visible to L3
DSB
CACHE_SYNC
```

Revision r3p2 implements a counter so that slots are automatically drained after 256 cycles of presence in the store buffer.

2.4 Category B (Rare)

This section describes Category B rare errata.

2.4.1 (738415) A cacheable read with address bits [20:5] equal to `0x0000` can be stalled by non-cacheable read traffic targeting other address region

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat B (rare)

Fault Status: Present in: r0p0, r1p0, r2p0. Fixed in r3p0. Unchanged in this document.

Description

When a cacheable read transaction issues its lookup to the L2 cache, address hazard checking is performed between the cacheable read and the active address slots in the PL310 master port(s).

- On r0p0 and r1p0, this address hazard checking is done by only comparing bits[20:5] of the address.
- On r2p0, there is an RTL configuration (`define pl310_FULL_ADDR_HAZARD in pl310_defs.v) that controls the number of bits involved in address hazard checking:
  - If the RTL configuration is not implemented, the behavior is the same as on r0p0 and r1p0. The erratum applies to this case.
  - If the RTL configuration is implemented (this is the recommended configuration), bits[31:5] are taken into account for address hazard checking. In this case, this erratum does not apply; refer to defect #719601.

When non-cacheable reads are active in the master port(s), the corresponding slots are marked as valid but their address is always considered as 0x0 even if they target another address range.

As a result, if the cacheable read with address bits [20:5] is equal to `0x0000`, the address hazard can be detected with non-cacheable reads active in the master port(s).

Conditions

This problem occurs when the following conditions are met:

1. The revision of PL310 is r0p0, r1p0, or r2p0 with pl310_FULL_ADDR_HAZARD configuration not implemented.
2. A cacheable read with address bits [20:5] equal to 0x0000 is received by one PL310 slave port.
3. A continuous flow of non-cacheable reads targeting another address region is serviced by PL310 such that there is always at least one outstanding non-cacheable request that has been issued by PL310.

   In this context, non-cacheable can be Strongly Ordered, Device or Normal Memory Non-Cacheable.

Implications

Under the conditions described, a cacheable read can be unnecessarily stalled by non-cacheable read requests. The cacheable read can only be serviced when there are no outstanding non-cacheable read transactions active in PL310. This is most likely to be an issue where the L2 cache is servicing requests from multiple cores, and so the number of non-cacheable accesses is unbounded.

If there is no dependency between the cacheable read and the non-cacheable traffic, the stall can potentially take a long time. This will not result in functionally incorrect operation.

If there is a dependency between the cacheable read and the non-cacheable traffic, this could in principle result in deadlock. For example, this could occur when the continuous non-cacheable traffic is actually coming from multiple polling loops each of which are waiting for the end of a process and the cacheable read itself is part of the process. The requirements for such multiple polling loops to non-cacheable space dependent on the forward progress of cacheable transactions are thought to be extremely unlikely to occur in general code, but might occur in very specific cases, for example during the booting of a multiprocessor operating system.

Workaround

If necessary, the WFE/SEV workaround can be implemented as follows:

- the primary processor that is responsible for updating a status location at the end of the process does this by executing the following pseudo-code sequence:

```
STR [SO location]
DSB
SEV
```

- the secondary processors that poll the status location do this by executing the following pseudo-code sequence:

```
loop
LDR [SO location]
CMP
WFENE
BNE loop
```

2.5 Category C

This section describes Category C errata.

2.5.1 (501023) Address Filtering register content cannot be read

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat C

Fault Status: Present in: r0p0. Fixed in r1p0. Unchanged in this document.

Description

There are two configuration registers linked to the Address Filtering feature:

- Address Filtering Start Register, offset `0xC00`
- Address Filtering End Register, offset `0xC04`.

The PL310 Cache Controller Technical Reference Manual documents these registers as being read-write. This behavior is not correctly implemented in the cache controller and the registers are always read as zero whatever their actual content.

Conditions

This problem occurs when the following conditions are met:

1. The Address Filtering registers have been configured and no longer have their default values.
2. A read transaction targeting the PL310 register file is received by one slave port with the offset `0xC00` or `0xC04`.
3. Data `0x0` is returned.

Implications

The Address Filtering registers are only supposed to be written in systems using the Address Filtering feature. In such systems, this erratum has implications when the content of these registers must be known. One common situation is a low power mode entry sequence during which the content of all registers must be saved before powering down the L2 cache.

Workaround

A workaround for this erratum is to hold the content of the Address Filtering registers in another memory area. During a low power mode exit sequence, this permits restoration of the correct register values by reading them from this other memory area.

2.5.2 (754670) A continuous write flow can stall a read targeting the same memory area

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat C

Fault Status: Present in: all revisions. Unchanged in this document.

Description

In PL310 r3px (x=0, 1, 2), and previous revisions (r0 to r2) with the pl310_FULL_ADDR_HAZARD RTL option implemented, hazard checking is done on bits [31:5] of the address.

In PL310 r0 to r2 with the pl310_FULL_ADDR_HAZARD RTL option NOT implemented, hazard checking is done on bits [20:5] of the address.

When PL310 receives a read with Normal Memory (cacheable or not) attributes, hazard checking is performed with the active writes of the store buffer. If an address match is detected, the read is stalled until the write completes.

Because of this erratum, a continuous flow of writes can stall a read that targets the same memory area.

Conditions

This problem occurs when the following conditions are met:

1. PL310 revision is r3, or r0 to r2 with the pl310_FULL_ADDR_HAZARD RTL option implemented.
2. PL310 receives a continuous write traffic targeting the same address marked with Normal Memory attributes.
3. While treating this flow, PL310 receives a read that targets the same 32-byte memory area.

OR

1. PL310 revision is r0 to r2 with the pl310_FULL_ADDR_HAZARD RTL option NOT implemented.
2. PL310 receives a continuous write traffic with Normal Memory attributes and targeting addresses with identical [20:5] bits.
3. While treating this flow, PL310 receives a read with the same address bits [20:5].

Implications

When the conditions above are met, the read might be stalled until the write flow stops.

Note that this erratum does not lead to any data corruption.

Note also that normal software code is not expected to contain long write sequence like the one causing this erratum to occur.

Workaround

There is no workaround and there is unlikely to be a need for a workaround for this erratum.

2.5.3 (765569) Prefetcher can cross 4KB boundary if offset is programmed with value 23

Status

Affects: product PL310 Level-2 Cache Controller.

Fault Type: Cat C

Fault Status: Present in: all revisions. Unchanged in this document.

Description

When the prefetch feature is enabled (bits[29:28] of the Auxiliary or Prefetch Control Register set HIGH), the prefetch offset bits of the Prefetch Control Register (bits[4:0]) configure the advance taken by the prefetcher compared to the current cache line. Refer to the PL310 Cache Controller Technical Reference Manual for more information. One requirement for the prefetcher is to not go beyond a 4KB boundary. If the prefetch offset is set to 23 (5'b10111), this requirement is not fulfilled and the prefetcher can cross a 4KB boundary.

Conditions

This problem occurs when the following conditions are met:

1. One of the Prefetch Enable bits (bits [29:28] of the Auxiliary or Prefetch Control Register) is set HIGH.
2. The prefetch offset bits are programmed with value 23 (5'b10111).

Implications

When the conditions above are met, the prefetcher can issue linefills beyond a 4KB boundary compared to the original transaction. This can cause system issues because those linefills can target a new 4KB page of memory space, regardless of page attribute settings in L1 MMU.

Workaround

A workaround for this erratum is to program the prefetch offset with any value except 23.