Arm® Cortex®-A65 (MP080)

Software Developer Errata Notice

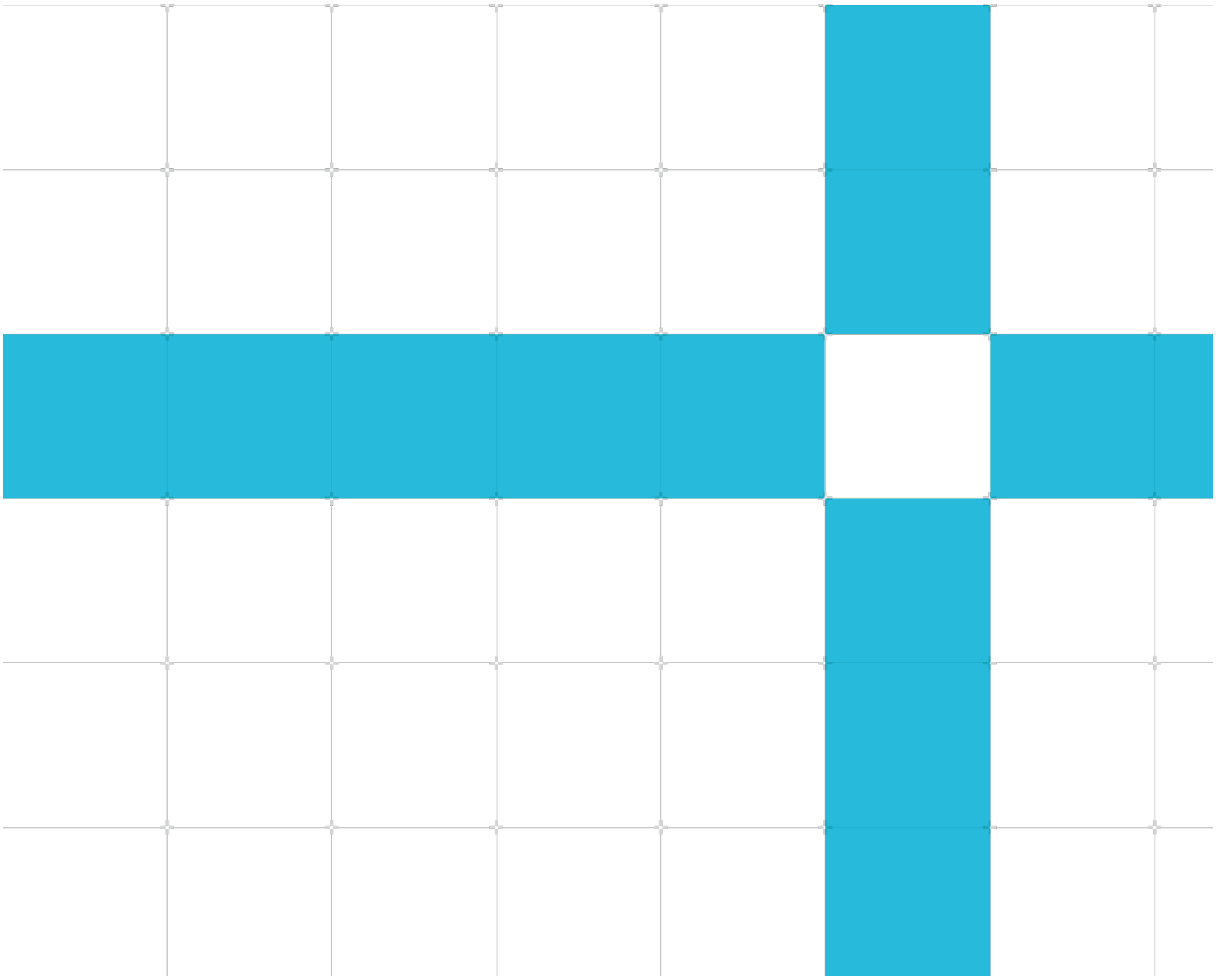
Date of issue: 26-Jul-2022

Non-Confidential

Document version: v8.0

Document ID: SDEN-1065159

This document contains all known errata since the r0p0 release of the product.



Non-confidential proprietary notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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.

Unrestricted Access is an Arm internal classification.

Product status

The information in this document is for a product in development and is not final.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm® Cortex®-A65 (MP080), create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey: **https://developer.arm.com/documentation-feedback-survey**.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

If you find offensive language in this document, please email **terms@arm.com**.

Contents

Introduction

Scope

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

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

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.

Change Control

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 erratum text. Fixed errata are not shown as updated unless the erratum text has changed. The **errata summary table** identifies errata that have been fixed in each product revision.

26-Jul-2022: Changes in document version v8.0

ID	Status	Area	Category	Summary
2599524	New	Programmer	Category B (rare)	Completion of affected memory accesses might not be guaranteed by completion of a TLBI
2599521	New	Programmer	Category C	Atomic store instructions might not report an External abort or System Error

16-Nov-2021: Changes in document version v7.0

No new or updated errata in this document version.

31-Mar-2020: Changes in document version v6.0

ID	Status	Area	Category	Summary
1595308	Updated	Programmer	Category C	Cache debug Read operations might not correctly return cache data
1599706	Updated	Programmer	Category C	Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation

22-Nov-2019: Changes in document version v5.0

ID	Status	Area	Category	Summary
1541130	New	Programmer	Category B	Speculative AT instruction using out-of-context translation regime might cause subsequent request to generate an incorrect translation
1595308	New	Programmer	Category C	Cache debug Read operations might not correctly return cache data
1599706	New	Programmer	Category C	Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation

25-Mar-2019: Changes in document version v4.0

ID	Status	Area	Category	Summary
1357727	New	Programmer	Category C	TLB Maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs

14-Dec-2018: Changes in document version v3.0

ID	Status	Area	Category	Summary
1227419	New	Programmer	Category B	One thread might incorrectly access the translation of the other thread when CnP is enabled
1243888	New	Programmer	Category B	Transitioning between multi-threaded mode and single-threaded mode when instructions are in progress might cause the core to hang
1252340	New	Programmer	Category C	Loads of mismatched size might not be single-copy atomic
1271812	New	Programmer	Category C	ECC Fatal error is not reported in the presence of lower priority ECC errors
1330121	New	Programmer	Category C	VFP_SPEC and ASE_SPEC PMU events count incorrectly

16-Jul-2018: Changes in document version v2.0

ID	Status	Area	Category	Summary
1185509	New	Programmer	Category A (rare)	A LD3 instruction might result in a spurious linefill request to an unexpected physical address
1092214	New	Programmer	Category B	One thread might incorrectly access the translation of the other thread when CnP is enabled
1156724	New	Programmer	Category B	Transitioning between single-threaded mode and multithreaded mode when instructions are in progress might cause the core to hang or lead to data corruption
1179935	New	Programmer	Category B	A translation table walk request might result in data corruption
1203058	New	Programmer	Category B	A thread might incorrectly trigger a breakpoint even when breakpoint is not enabled
1092588	New	Programmer	Category C	Loads of mismatched size might not be single-copy atomic
1093507	New	Programmer	Category C	Consistently filling a cacheline with transient attributes or a non-temporal hint that is used by the exclusive monitor on the opposite thread of the same core can stop progress of exclusives
1094186	New	Programmer	Category C	Asynchronous interrupts might be delayed when pended Reset Catch debug events are used with debug software step events while software debug is disabled through a hardware debugger
1103552	New	Programmer	Category C	Core might not function correctly in the presence of a persistent error in the TLB or specific cache RAMs
1149497	New	Programmer	Category C	Non-allocating stores might prevent progress of an exclusive loop
1162068	New	Programmer	Category C	Double-bit ECC errors on both lines of a cache line crossing an L1 data load access might cause a livelock
1168856	New	Programmer	Category C	Incorrect parity error reporting when translation table walks from both threads access the TLB RAM in back-to-back cycles
1185809	New	Programmer	Category C	Single-bit parity error is not corrected during boot
1203934	New	Programmer	Category C	Exclusives on one thread do not make progress because of PRFM L2 operations on the other thread that cause L1 data invalidates

12-Feb-2018: Changes in document version v1.0

ID	Status	Area	Category	Summary
1058137	New	Programmer	Category C	Incorrect ETM timestamp value when timestamp and event generation happen on same cycle
1058143	New	Programmer	Category C	Cycle count value in timestamp packet might be incorrect
1073486	New	Programmer	Category C	ECC error observed in an L1 data RAM for a partially filled line might cause future accesses to the same set to be treated as non-cacheable

Errata summary table

The errata associated with this product affect the product versions described in the following table.

ID	Area	Category	Summary	Found in versions	Fixed in version
1185509	Programmer	Category A (rare)	A LD3 instruction might result in a spurious linefill request to an unexpected physical address	r0p0	r1p0
1541130	Programmer	Category B	Speculative AT instruction using out-of-context translation regime might cause subsequent request to generate an incorrect translation	r0p0, r1p0, r1p1, r1p2	Open
1243888	Programmer	Category B	Transitioning between multi-threaded mode and single-threaded mode when instructions are in progress might cause the core to hang	r0p0, r1p0	r1p1
1227419	Programmer	Category B	One thread might incorrectly access the translation of the other thread when CnP is enabled	r0p0, r1p0	r1p1
1203058	Programmer	Category B	A thread might incorrectly trigger a breakpoint even when breakpoint is not enabled	r0p0	r1p0
1179935	Programmer	Category B	A translation table walk request might result in data corruption	r0p0	r1p0
1156724	Programmer	Category B	Transitioning between single-threaded mode and multithreaded mode when instructions are in progress might cause the core to hang or lead to data corruption	r0p0	r1p0
1092214	Programmer	Category B	One thread might incorrectly access the translation of the other thread when CnP is enabled	r0p0	r1p0
2599524	Programmer	Category B (rare)	Completion of affected memory accesses might not be guaranteed by completion of a TLBI	r0p0, r1p0, r1p1, r1p2	Open
2599521	Programmer	Category C	Atomic store instructions might not report an External abort or System Error	r0p0, r1p0, r1p1, r1p2	Open
1599706	Programmer	Category C	Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation	r0p0, r1p0, r1p1, r1p2	Open
1595308	Programmer	Category C	Cache debug Read operations might not correctly return cache data	r0p0, r1p0, r1p1	r1p2

ID	Area	Category	Summary	Found in versions	Fixed in version
1357727	Programmer	Category C	TLB Maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs	r0p0, r1p0	r1p1
1330121	Programmer	Category C	VFP_SPEC and ASE_SPEC PMU events count incorrectly	r0p0, r1p0, r1p1, r1p2	Open
1271812	Programmer	Category C	ECC Fatal error is not reported in the presence of lower priority ECC errors	r0p0, r1p0	r1p1
1252340	Programmer	Category C	Loads of mismatched size might not be single-copy atomic	r0p0, r1p0	r1p1
1203934	Programmer	Category C	Exclusives on one thread do not make progress because of PRFM L2 operations on the other thread that cause L1 data invalidates	r0p0	r1p0
1185809	Programmer	Category C	Single-bit parity error is not corrected during boot	r0p0	r1p0
1168856	Programmer	Category C	Incorrect parity error reporting when translation table walks from both threads access the TLB RAM in back-to-back cycles	r0p0	r1p0
1162068	Programmer	Category C	Double-bit ECC errors on both lines of a cache line crossing an L1 data load access might cause a livelock	r0p0	r1p0
1149497	Programmer	Category C	Non-allocating stores might prevent progress of an exclusive loop	r0p0	r1p0
1103552	Programmer	Category C	Core might not function correctly in the presence of a persistent error in the TLB or specific cache RAMs	r0p0	r1p0
1094186	Programmer	Category C	Asynchronous interrupts might be delayed when pended Reset Catch debug events are used with debug software step events while software debug is disabled through a hardware debugger	r0p0	r1p0
1093507	Programmer	Category C	Consistently filling a cacheline with transient attributes or a non-temporal hint that is used by the exclusive monitor on the opposite thread of the same core can stop progress of exclusives	r0p0	r1p0
1092588	Programmer	Category C	Loads of mismatched size might not be single-copy atomic	r0p0	r1p0
1073486	Programmer	Category C	ECC error observed in an L1 data RAM for a partially filled line might cause future accesses to the same set to be treated as non-cacheable	r0p0	r1p0

ID	Area	Category	Summary	Found in versions	Fixed in version
1058143	Programmer	Category C	Cycle count value in timestamp packet might be incorrect	r0p0, r1p0, r1p1, r1p2	Open
1058137	Programmer	Category C	Incorrect ETM timestamp value when timestamp and event generation happen on same cycle	r0p0, r1p0, r1p1, r1p2	Open

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

1185509
An LD3 instruction might result in a spurious linefill request to an unexpected physical address

Status

Fault Type: Programmer Category A (Rare)
Fault Status: Present in r0p0. Fixed in r1p0

Description

Executing an LD3 Multiple (Q=0) instruction with certain combinations of encodings and base address alignments might result in a spurious linefill request to an unexpected and UNPREDICTABLE physical address in the memory system.

Configurations Affected

This erratum affects all configurations.

Conditions

1. An LD3 instruction executes and accesses memory locations within the last 16 bytes of a page (A) without overlapping the following page (B).
2. Accesses to addresses within page B generate a translation fault.

Implications

If the above conditions are met, then the core might make a linefill request to an unexpected and UNPREDICTABLE physical address. If the system is unable to respond to this spurious request with either a valid data return or an error response, then the core might hang. The linefill request might also occur to a read-sensitive location, which might cause other unexpected or UNPREDICTABLE behaviors.

Workaround

No workaround is available for sample silicon.

Category B

1541130
Speculative AT instruction using out-of-context translation regime might cause subsequent request to generate an incorrect translation

Status

Fault Type: Programmer Category B
Fault Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

A Speculative Address Translation (AT) instruction translates using registers that are associated with an out-of-context translation regime, and caches the resulting translation in the TLB. A subsequent translation request, generated when the out-of-context translation regime is current, uses the previous cached TLB entry producing an incorrect virtual to physical mapping.

Configurations Affected

This erratum affects all configurations.

Conditions

1. A Speculative AT instruction performs a table walk translating a virtual address to a physical address using registers that are associated with an out-of-context translation regime.
2. The address translation data that is generated during the walk is cached in the TLB.
3. The out-of-context translation regime becomes current and a subsequent memory access is translated using previously cached address translation data in the TLB, resulting in an incorrect virtual to physical mapping.

Implications

If the above conditions are met, then the resulting translation is incorrect.

Workaround

When context-switching the register state for an out-of-context translation regime, system software at EL2 or above must ensure that all intermediate states during the context switch report a Level 0 translation fault in response to an AT instruction that targets the out-of-context translation regime. Note that a workaround is only required if the system software contains an AT instruction as part of an executable page.

1243888
Transitioning between multi-threaded mode and single-threaded mode when instructions are in progress might cause the core to hang

Status

Fault type: Programmer Category B
Fault status: Present in r0p0, r1p0. Fixed in r1p1

Description

The core supports transitioning between multithreaded mode and single-threaded mode when instructions are in progress for a thread that stays active while another thread transitions between active and inactive state. Under rare conditions, a hardware access flag update initiated by the thread that is being de-activated or activated, might not complete and then cause the core to hang.

Configurations affected

All configurations are affected.

Conditions

This erratum occurs if one of the following scenarios is true:

Scenario 1:

1. Thread A initiates a powerdown sequence with the MMU on. The last instruction in the sequence is a WFI. A prefetch for instructions following the WFI initiates a translation table walk which requires a hardware access flag update.
2. Thread A powerdown completes and the core goes to single-threaded mode before the translation table walk of Thread A completes.
3. The translation table walk descriptor fetch returns for Thread A. The access flag update is initiated at this point but cannot complete, causing the core to hang.

Scenario 2:

1. Thread A gets re-activated.
2. Thread A initiates a translation table walk which requires a hardware access flag update or a hardware dirty bit atomic update.
3. The translation table walk descriptor fetch returns for Thread A. The access flag update is initiated at this point but cannot complete, causing the core to hang.

Implications

If one of the two scenarios occurs, then hardware management of the access flag or dirty state might not complete and the core might hang after transitioning between single-threaded mode and multithreaded mode.

Workaround

Software can execute instructions to do the following:

- When de-activating a thread:

```
MRS X0, TCR_ELx
MOV X1, #1
BIC X0, X0, X1, LSL #Y // see Note below for details on the value of Y

MSR TCR_ELx, X0
ISB
MOV X0, #0
TLBI VALE1,X0
DSB
WFI

//Note:
// 1.    "TCR_ELx" is the Translation Control Register corresponding to the
//       system state in which the thread will be/is re-activated
// 2.    "Y" is the value of the bit position of the HA field (Hardware Access //
flag update field) in TCR_ELx.
//       - The bit position of the HA field in TCR_EL1 is 39.
//       - The bit position of the HA field in TCR_EL2 depends on the
//         value of HCR_EL2.E2H.
//       - The bit position is 21 when HCR_EL2.E2H == 0,
//         and is 39 when HCR_EL2.E2H == 1.
//       - The bit position of the HA field in TCR_EL3 is 21.
```

- When re-activating a thread, before re-enabling hardware management of the access flag:

```
MOV X0, #0
TLBI VALE1,X0
DSB
```

1227419
One thread might incorrectly access the translation of the other thread when CnP is enabled

Status

Fault type: Programmer Category B
Fault status: Present in r0p0, r1p0. Fixed in r1p1

Description

When CnP is enabled for a translation scheme, a thread might incorrectly use a translation from a different translation scheme for the other thread.

Configurations affected

All Helios configurations are affected.

Conditions

1. Multithread mode is on.
2. One of the following scenarios occurs:
a. The threads run at different Exception levels. The thread that accesses the translation has a different context from the other thread for the same Exception level. The Exception level of the other thread allows sharing of the translation between threads.
b. The threads run at the same Exception level, but their translation scheme differs.

Implications

An incorrect translation might be used.

Workaround

Disable cross-thread translation sharing in the instruction uTLB by setting bit[51] of CPUACTLR_EL1 (CPU Auxiliary Control Register) to '1.

1203058
A thread might incorrectly trigger a breakpoint even when breakpoint is not enabled

Status

Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r1p0

Description

When one thread is enabling breakpoint while it is not idle and it is waiting for a translation table walk, the other thread can generate a breakpoint even when breakpoint is not enabled.

Configurations affected

This erratum affects all configurations.

Conditions

1. Multithread mode is on.
2. One thread is enabling breakpoint while its instruction fetch is waiting for a translation table walk.
3. The other thread does not have breakpoint enabled and hits in the L1 TLB in the same cycle as the breakpoint is enabled for the opposite thread.

Implications

An unexpected breakpoint might be triggered.

Workaround

No workaround is required for silicon samples.

1179935
A translation table walk request might result in data corruption

Status

Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r1p0

Description

Under specific internal timing conditions, a translation table walk request made to an L1 cacheable location while there is a store in progress to that same location, might result in corruption of the translation.

Configurations Affected

This erratum affects all configurations.

Conditions

1. A store is in progress to memory location A.
2. An L1 linefill is in progress to memory location A.
3. A translation table walk request is made to memory location A.

Implications

If the above conditions are met along with specific internal timing conditions, then the core might process a portion of the translation table walk with the pre-store data while another portion proceeds with the post-store data. This might lead to corruption of the translation as well as corruption of the information cached in the TLB for the corresponding page(s).

Workaround

The erratum can be avoided by preventing translation table walks from allocating lines into the L1 cache. This has a negligible impact on performance when an L2 cache is present. To do this, set CPUACTLR(_EL1)[49] to 1.

1156724
Transitioning between single-threaded mode and multithreaded mode when instructions are in progress might cause the core to hang or lead to data corruption

Status

Fault type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r1p0

Description

The core supports transitioning between single-threaded mode and multithreaded mode when instructions are in progress. Under rare conditions:

- A hardware access flag update that occurs while the core is transitioning between single-threaded mode and multithreaded mode might be lost, which might cause the core to hang.
- New stores executing after a transition might be lost.

Configurations affected

All configurations are affected.

Conditions

One of the following conditions is true:

- The core transitions between single-threaded mode and multithreaded mode in the same cycle as a hardware access flag update occurs.
- After a transition between single-threaded mode and multithreaded mode, the active thread performs a new store instruction before the store buffer is drained.

Implications

If one of the above conditions is met, then data might be lost and the core might hang after transitions between single-threaded mode and multithreaded mode.

Workaround

For silicon samples, do not put threads in deactivated mode. An active thread is not able to use the full hardware resources available and therefore does not gain full advantage of single-threaded mode.

1092214
One thread might incorrectly access the translation of the other thread when CnP is enabled

Status

Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r1p0

Description

When CnP is enabled for a translation scheme, a thread might incorrectly use a translation from a different translation scheme for the other thread.

Configurations affected

This erratum affects all core configurations.

Conditions

1. Multithread mode is on.
2. One of the following scenarios occurs:
    a. The threads run at different Exception levels. The thread that accesses the translation has a different context from the other thread for the same Exception level. The Exception level of the other thread allows sharing of the translation between threads.
    b. The threads run at the same Exception level, but their translation scheme differs as the threads do not use the same settings for SCTLR_ELx.[0] and HCR_EL2.[0].

Implications

An incorrect translation might be used.

Workaround

Disable the usage of CnP by setting TTBR0_ELx[0] and TTBR1_ELx[0] to 0.

Category B (rare)

2599524
Completion of affected memory accesses might not be guaranteed by completion of a TLBI

Status

Fault Type: Programmer Category B (Rare)
Fault Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

The core might not guarantee completion of all memory accesses after completion of a TLB Invalidate (`TLBI`) instruction affecting those accesses on another core.

Configurations affected

This erratum affects all configurations.

Conditions

1. Another PE in the system executes a `TLBI` or Instruction Cache (`IC`) instruction, followed by a Data Synchronization Barrier (`DSB`) instruction.
2. The core executes a store to a memory location A.
3. Another PE in the system modifies the descriptor used by the store to memory location A, using a break-before-make sequence.
    - The break-before-make sequence will include a `TLBI` instruction, followed by a `DSB` instruction.
4. Rare, timing-sensitive, microarchitectural conditions occur.

Implications

The `DSB` used after the `TLBI` as part of the break-before-make sequence might not guarantee the completion of the store to memory location A under very rare and unlikely timing conditions. For most systems and applications, the latency of the break-before-make sequence and time until later reuse is very likely to exceed the latency required to naturally complete the store.

Workaround

Given the rarity of the conditions needed to trigger this erratum, a workaround is not expected to be needed in most systems.
If a workaround is required, then the `TLBI`, `DSB` sequence from the break-before-make sequence can be repeated. After repeating the `TLBI`, `DSB` sequence, all memory accesses that use a translation changed by the break-before-make sequence will have completed.

Category C

2599521
Atomic store instructions might not report an External abort or System Error

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

If the core receives read data from the interconnect where some beats of the data indicate an error, then, for certain types of atomic instruction, the core might not report the error.

Configurations Affected

This erratum affects all configurations.
However, if the system interconnect supports poisoning, then it is unlikely to meet the other conditions required.

Conditions

1. The CPUECTLR_EL1.ATOMIC field is set to 0b01, which forces all store atomics to be performed in the L1 cache.
2. A core executes an atomic store instruction to Inner Write-Back, Outer Write-Back cacheable memory.
3. The address accessed by the instruction is not present in any cache in the cluster, so the DSU sends a read transaction to the system interconnect.
4. The interconnect returns data that contains an error.
    - On CHI, this means some data flits indicate DERR or NDERR.
    - On ACE, this means some data transfers indicate SLVERR or DECERR.

Implications

If these conditions are met, then the core will correctly discard the data from the atomic store and the data from the interconnect, however it will not report an External abort or System Error interrupt. Therefore, the software will be unaware that the data was discarded.
The software is unlikely to use an atomic store as the first instruction to access this address. Therefore, if the error response from the interconnect is because of a permanent fault such as the address not being mapped to any peripheral, then an abort would have been reported when the software initially wrote to that address with a normal store instruction.
Systems using a CHI interface and configured with ECC support would be expected to poison data that got an uncorrectable error, rather than returning a DERR or NDERR. These systems using poison would not be impacted by this erratum.
In systems that meet the configurations described, this erratum might cause a negligible increase in overall system failure rate.

Workaround

For many systems that have not changed the default value of CPUECTLR_EL1.ATOMIC, no workaround is necessary. The software should not set the CPUECTLR_EL1.ATOMIC field to 0b01.

1599706

Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

The core provides a mechanism to read the internal memory that is used by the L1 cache and TLB structures through IMPLEMENTATION DEFINED system registers. The memory and location are selected using a Write operation which dumps the data to into various data registers (CDBGDR0_EL3, CDBGDR1_EL3, and CDBGDR2_EL3) and the data is accessed using a Read operation of the data registers.

Cache debug data register Reads performed as specified in the Core Technical Reference Manual results in an UNDEFINED exception.

Additionally, MRS <xd>, s3_3_c15_c0_3 should return and UNDEFINED exception but does not.

Configurations Affected

This erratum affects all configurations.

Condition(s)

Read the cache debug data register using the mnemonics specified in the Core Technical Reference Manual.

Implications

The cache debug data register Read results in an UNDEFINED exception instead of reading the contents into the destination register. The cache debug data register Read operations are mapped to different locations with the Op1 field set to 6 instead of 3 and are accessible in EL3 only.

Workaround

The Op1 field for Read access of the three data registers should be 3 instead of 6.

That is, the following opcodes are to be used:
CDBGDR0_EL3: MRS <Xd>, S3_3_c15_c0_0
CDBGDR1_EL3: MRS <Xd>, S3_3_c15_c0_1
CDBGDR2_EL3: MRS <Xd>, S3_3_c15_c0_2

Additionally, software should not be using MRS <xd>, s3_3_c15_c0_3 and expect an UNDEFINED exception.

1595308

Cache debug Read operations might not correctly return cache data

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0, r1p0, and r1p1. Fixed in r1p2

Description

The core provides a mechanism to read internal memory contents from the L1 instruction cache, L1 data cache, and TLB structures through IMPLEMENTATION DEFINED system registers. In some cases, the core might not provide correct instruction cache and TLB RAM read results and does not provide data cache data/tag/dirty RAM read data.

Configurations Affected

This erratum affects all configurations.

Conditions

For data cache data Read operations and data cache tag Read operations:

1. Perform a cache debug operation, either a data cache data Read operation or a data cache tag Read operation, as specified in the Technical Reference Manual and in the errata notice.
2. Read the cache debug data registers associated with the prior cache debug read operation.

For instruction cache tag read operations, instruction cache data Read operations, TLB tag Read operations, and TLB data Read operations:

1. On a given thread, perform one of the following cache debug operations as specified in the Technical Reference Manual and errata notice:
   a. Instruction cache tag Read.
   b. Instruction cache data Read.
   c. TLB tag Read.
   d. TLB data Read.
2. On the opposite thread, perform any cache debug read operation.
3. Read the cache debug data registers associated with the first cache debug operation.

Implications

If the above conditions are met, then the data associated with the cache debug read operations might not appear correctly in the cache debug data registers associated with the cache debug read operation.

Workaround

There is no workaround for data cache data Read operations and data cache tag Read operations.

For all other cache debug Read operations, a software synchronization mechanism can be used to guarantee exclusive access to the cache debug Read operations and the associated cache debug data Read registers.

1357727
TLB maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0, r1p0. Fixed in r1p1

Description

Software should ensure that changes to the translation table are reflected correctly in the TLB RAMs through appropriate TLB maintenance operations. However, in the presence of a persistent error in the TLB tag/data RAMs, a TLB maintenance operation that encounters a parity error might not invalidate the required entries in the TLB.

Configurations affected

This erratum affects all configurations.

Conditions

1. A persistent (non-transient) parity error exists in the TLB tag/data RAM.
2. A TLB maintenance operation encounters a transient parity error.
3. The transient parity goes away on the entry associated with the maintenance operation.

Implications

There is still substantial benefit being gained from the ECC logic. There might be a negligible increase in overall system failure rate because of this erratum.

Workaround

No workaround is required.

1330121
VFP_SPEC and ASE_SPEC PMU events count incorrectly

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

The core implements several architectural and IMPLEMENTATION DEFINED performance monitor events. Because of this erratum, the partitioning of instructions to count the performance event monitors VFP_SPEC and the ASE_SPEC is different from what is specified in the architecture.

Configurations Affected

All configurations with NEON_FP enabled are affected.

Conditions

PMU event counters are configured to count the 0x0074, ASE_SPEC and/or 0x0075, VFP_SPEC events.

Implications

Each ASE_SPEC and VFP_SPEC PMU event counts a collection of SIMD floating-point instructions as well as scalar floating-point instructions, however the sum of these two events should still be correct and count as if these event counters were implemented correctly.

Workaround

There is no workaround.

1271812
ECC fatal error is not reported in the presence of lower priority ECC errors

Status

Fault type: Programmer Category C
Fault status: Present in r0p0, r1p0. Fixed in r1p1

Description

A cache lookup request that encounters a single-bit ECC error in an L1 data RAM and persistent error in the L1 dirty RAM simultaneously, does not report the single-bit ECC error. Software might not be aware of the existence of a persistent error, even though the core observes the effects of the error. Persistent ECC errors cause subsequent cacheable access to the set to be treated as non-cacheable, causing loss of coherency.

Configurations affected

All configurations are affected.

Conditions

1. The data needed for the snoop is in the L1 cache.
2. A persistent ECC error is detected in the dirty RAM by the snoop.
3. A single-bit ECC error is present in the data requested by a previous snoop.

Implications

There is still substantial benefit being gained from the ECC logic. There might be a negligible increase in overall system failure rate because of this erratum.

If the above conditions are met along with specific timing of the snoop requests, then a fatal ECC error might not be reported for the persistent ECC error The single-bit error is reported. Software is not expected to continue operating normally following a persistent error.

Workaround

No workaround is required.

1252340
Loads of mismatched size might not be single-copy atomic

Status

Fault type: Programmer Category C
Fault status: Present in r0p0, r1p0. Fixed in r1p1

Description

A core executing a cacheable store followed some time later by a cacheable load of a larger size to the same address might not meet single-copy atomicity requirements.

Configurations affected

All configurations are affected.

Conditions

1. On one PE (PE0), a non-release store instruction which is required to be single-copy atomic is executed.
2. On a second PE (PE1), the following sequence must occur:
   a. A cacheable store instruction is executed. This store must have a smaller access size than the store on first PE (PE0), and must be to an address with the bytes accessed by the first PE (PE0).
   b. A cacheable load instruction that crosses a 128-bit boundary and requires single copy atomicity is executed. The load must have a larger access size than the store from the same PE, and at least one byte of the load must be to the same address as the store.

The Arm architecture requires that the load is single-copy atomic. However, in the conditions described, the load might observe a combination of the two stores, indicating that the PE0 store was ordered first. However, if the load is repeated a second time, then it might see only the data from the PE0 store indicating that the PE0 store was ordered second.

Implications

Concurrent, unordered stores are not common in multithreaded code. In the C11 standard, they are restricted to the family of "relaxed" atomics. In addition, using different size load and store instructions to access the same data is unusual. Therefore, most multithreaded software is not going to meet the conditions for this erratum.

Workaround

Most multithreaded software is not expected to meet the conditions for this erratum and therefore does not require a workaround. If a workaround is required, then the store on the second PE should be replaced with a store release instruction.

1203934
Exclusives on one thread do not make progress because of PRFM L2 operations on the other thread that cause L1 data invalidates

Status

Fault type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

A PRFM on one thread targeting the L2 to the same Physical Address (PA) as an exclusive sequence on the other thread can cause a cacheline invalidation that forces an exclusive code sequence to fail.

Configurations Affected

This erratum affects all configurations.

Conditions

1. The exclusive line is brought into the L1 data cache.
2. A PRFM targeting the L2 executes before the LDXR on the opposite thread.
3. The LDXR executes and hits the L1 data cache, and sets the exclusive monitor.
4. A snoop caused by the PRFM to enforce the L1/L2 cache exclusivity causes an invalidate and clears the exclusive monitor before the STXR executes.

Implications

An exclusive sequence might not be able to make progress because of a constant stream of operations that perform PRFM L2 operations that invalidate the cache from the opposite thread.

Workaround

A denial of service can be avoided if the OS sets up a thread-specific timer-based interrupt source to interrupt each thread periodically.

1185809
Single-bit parity error is not corrected during boot

Status

Fault type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

A parity error seen on an instruction cache data or tag RAM, before error detection is enabled, might result in a corrupted instruction.

Configurations affected

This erratum affects all configurations.

Conditions

1. The Error Record Error Detection bit is not set.
2. A single-bit error occurs on an instruction cache data or tag RAM entry.

Implications

A corrupted instruction might be executed, leading to corrupted data or unexpected aborts.

Workaround

No workaround is needed for sample silicon to be resilient to parity errors in this scenario.

1168856
Incorrect parity error reporting when translation table walks from both threads access the TLB RAM in back-to-back cycles

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

When back-to-back translation table walk RAM accesses occur and one has a parity error, an incorrect reporting of the error might occur on one of the accesses.

Configurations Affected

This erratum affects all configurations.

Conditions

1. Both threads have active translation table walks.
2. Both threads access the TLB RAM in back-to-back cycles.
3. One or both of the accesses have a parity error.

Implications

An error is reported but the index might not be accurate. The error count might be over-reported.

Workaround

No workaround is required for silicon samples.

1162068
Double-bit ECC errors on both lines of a cache line crossing an L1 data load access might cause a livelock

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

Cache line crossing loads that encounter a double-bit ECC error on both cache lines do not allow either to be reported. The mechanism to handle double-bit errors causes a livelock.

Configurations Affected

This erratum affects all configurations.

Conditions

1. A load must cross a cache line.
2. The cache line crossing load must have an ECC error on both cache lines.

Implications

A livelock might occur and ECC errors might not be reported. For silicon samples, multi-bit errors might not be handled correctly.

Workaround

No workaround is required.

1149497
Non-allocating stores might prevent progress of an exclusive loop

Status

Fault type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

If non-allocating stores are included in a loop with a load and store exclusive sequence, then they might prevent the exclusive sequence from passing.

Configurations affected

All configurations are affected.

Conditions

1. A loop including a load and store exclusive instruction is executed.
2. The loop contains additional store instructions before the load exclusive, which get executed every time around the loop, including when the loop is re-executed because the store exclusive is failing.
3. The additional store instructions are at least two stores to the same cache line, and both stores are cacheable but do not allocate into the cache. This means that they are to memory marked as either cacheable No Write-Allocate or Transient, or that the stores are non-temporal. The stores can be to a different cache line from the load and store exclusive.

Implications

The store exclusive sequence might continually fail, causing a software livelock. Interrupts can still be taken so the OS or hypervisor can break out of the loop.

Workaround

It is not expected that non-allocating stores would be present in typical exclusive loops. Such loops should be rewritten to use allocating stores.

1103552
Core might not function correctly in the presence of a persistent error in the TLB or specific cache RAMs

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

The core should report persistent errors correctly and should not livelock because of a single persistent error in either:

- The TLB RAM.
- The instruction cache data and tag RAMs.
- The data cache data, tag, and dirty RAMs.
- The L2 cache data and tag RAMs.
- The L2 data buffer RAMs.

Because of this erratum, there are scenarios in sample silicon where the error might cause incorrect error reporting or a livelock.

Configurations affected

This erratum affects all core configurations.

Conditions

A persistent error exists in one of:

- The TLB RAM.
- The instruction cache data and tag RAMs.
- The data cache data, tag, and dirty RAMs.
- The L2 cache data and tag RAMs.
- The L2 data buffer RAMs.

Implications

The core might incorrectly report an error or livelock in the presence of a persistent error.

Workaround

No workaround is required for silicon samples.

1094186
Asynchronous interrupts might be delayed when pended Reset Catch debug events are used with debug software step events while software debug is disabled through a hardware debugger

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

When pended Reset Catch debug events are used with debug software step events, and software debug is dynamically disabled through a hardware debugger, asynchronous interrupts might be delayed until a future Halting Step debug event or software step event is taken.

Configurations Affected

This erratum affects all configurations.

Conditions

1. Reset catch is configured.
2. Warm reset is deasserted. However at this time, halting is not allowed as **SPIDEN** is set to 0, causing the Reset catch to pend.
3. Meanwhile, the core executes code, which enables software debug. MDSCR_EL1[0] is set to 1.
4. A context synchronization event occurs, for example ERET or ISB, which enables halting. This is interpreted as both a Reset Catch debug event and a software step event. The Reset Catch debug event has a higher priority and the thread enters halting mode.
5. While in halting mode, the APB programs the OS Lock bit, which disables software debug.
6. When exiting halting mode from the Reset Catch debug event, the core does not take the asynchronous interrupt that is presented.

Implications

If the above conditions occur, then asynchronous interrupts are blocked until the next Halting Step or Software Step event is completed. However, disabling self-hosted debug while in halting mode is not expected to be a typical scenario.

Workaround

Do not disable self-hosted debug through the debugger when in halting mode.

1093507
Consistently filling a cacheline with transient attributes or a non-temporal hint that is used by the exclusive monitor on the opposite thread of the same core can stop progress of exclusives

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

A load that uses the non-temporal hint or transient attribute to the same cacheline as an exclusive from the opposite thread might cause the exclusive line to be evicted from the L1 cache. The eviction causes a clear of the exclusive monitor before the corresponding store exclusive can execute, causing the exclusive to fail indefinitely.

For this to cause a livelock, the transient load or store must consistently initiate the fill of the line before the load exclusive from the other thread. If the load exclusive can execute before the transient load and initiate the linefill, then it will force the cacheline to be non transient and progress will be made. The filling of the line as transient and executing a second load that evicts the line acts as a denial of service to the exclusive sequence on the opposite thread.

Configurations Affected

This erratum affects all configurations.

Conditions

1. A load or store with transient attributes or a non-temporal hint to the same cacheline as an exclusive on the opposite thread fills the line into the cache.
2. All ways to the same index are valid in the cache.
3. A subsequent load to the same index is performed before the exclusive store executes.

Implications

An exclusive sequence can be blocked from making progress by a constant stream of transient or non-temporal loads or stores to the same cacheline on the opposite thread.

Workaround

For sample silicon, do not allow transient attributing or non-temporal loads or stores to the cacheline used by the exclusive monitor on the opposite thread.

1092588
Loads of mismatched size might not be single-copy atomic

Status

Fault type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r1p0

Description

A core executing a store followed some time later by a load to the same address, but with a larger access size, might not meet single-copy atomicity requirements.

Configurations affected

This erratum affects all configurations.

Conditions

1. On one PE (PE0), a non-release store instruction which is required to be single-copy atomic is executed.
2. On a second PE (PE1), the following sequence must occur:
   a. A non-release store instruction is executed. This store must have a smaller access size than the store on PE0, and must be to an address with the bytes accessed by PE0.
   b. A load instruction which requires single copy atomicity is executed. The load must have a larger access size than the store from the same PE, and at least some bytes of the load must be to the same address as the store.

The Arm architecture requires that the load is single-copy atomic. However, in the conditions described, the load might observe a combination of the two stores, indicating that the PE0 store was ordered first. However, if the load is repeated, then the second time it might see only the data from PE0 store indicating that PE0 store was ordered second.

Implications

Concurrent, unordered stores are not common in multithreaded code. Using different size load and store instructions to access the same data is unusual. Therefore, most multithreaded software is not going to meet the conditions for this erratum.

Workaround

Most multithreaded software is not expected to meet the conditions for this erratum and therefore will not require a workaround. If a workaround is required, then the store on PE1 should be replaced with a store release instruction.

1073486
ECC error observed in an L1 data RAM for a partially filled line might cause future accesses to the same set to be treated as non-cacheable

Status

Fault Type: Programmer Cat C
Fault Status: Present in r0p0. Fixed in r1p0

Description

A load request encountering an ECC error in an L1 data RAM when a linefill to the requested address is ongoing might be incorrectly treated as a persistent error and reported as deferred error (DE) in the error record primary status register.

Configurations Affected

This erratum affects all configurations.

Conditions

1. A line is in partially filled state in the L1 data RAM.
2. A read access hits the partially filled line. The data needed for the read operation is in the L1 cache.
3. An ECC error is present in the data requested by the read.
4. The completion of the outstanding linefill takes an unusual long time from the point where the read request encountered the ECC error.

Implications

An ECC error might be incorrectly reported as a persistent error, causing subsequent cacheable accesses to the same L1 data cache set to be treated as non-cacheable. Software is not expected to continue operating normally following the reporting of a deferred error (DE) in the error record primary status register, so this erratum results in a small increase in the failure in time (FIT) rate.

Workaround

To work around this erratum, a configuration register bit can be set as follows to disable persistent error handling:

```
DSB
ISB
MRS X0, CPUACTLR_EL1
ORR X0, X0, #0x0004000000000000
MSR CPUACTLR_EL1, X0
DSB
```

```
ISB
```


1058143
Cycle count value in timestamp packet might be incorrect

Status

Fault Type: Programmer Category C
Fault: Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

When a timestamp packet is generated in the trace, the timestamp packet can indicate the cycle count between the previous cycle count element and the element the timestamp is associated with. This can be used to infer the alignment between cycle count elements and the global timestamp. Timestamp packets can be inserted in the trace stream some time after the element the timestamp is associated with gets traced. Because of this erratum, the cycle count indicated in the timestamp packet will be determined by the time when the packet is inserted in the trace stream.

Configurations Affected

All configurations are affected.

Conditions

- Timestamping must be enabled, with TRCCONFIGR.TS== 1.
- Cycle counting must be enabled, with TRCCONFIGR.CCI== 1.

Implications

The cycle count for the timestamp packet will indicate a time after the element which was used to capture the timestamp. If there has been a gap in the tracing of elements which can be timestamped, this error can be large. This does not affect the incremental cycle count values which are related to instructions being committed.

Workaround

There is no workaround for this erratum.

1058137
Incorrect ETM timestamp value when timestamp and event generation happen on same cycle

Status

Fault Type: Programmer Category C
Fault: Status: Present in r0p0, r1p0, r1p1, r1p2. Open

Description

When an event packet and timestamp packet are generated on the same cycle, the value of the timestamp packet should be sampled at the time of that event. Because of this erratum, the value of the timestamp is sampled on the previous atom or event packet.

Configurations Affected

All configurations are affected.

Conditions

- Timestamping is enabled.
- Event packet generation is enabled.
- An event packet and a timestamp packet must be generated on the same cycle.

Implications

The timestamp value might be slightly out of date. In a typical system, atom packets and event packets are frequent relative to the global timestamp signal increment, which means that the amount of errors will be small.

Workaround

There is no workaround for this erratum.