Arm® Cortex®-A65AE (MP082)

Software Developer Errata Notice

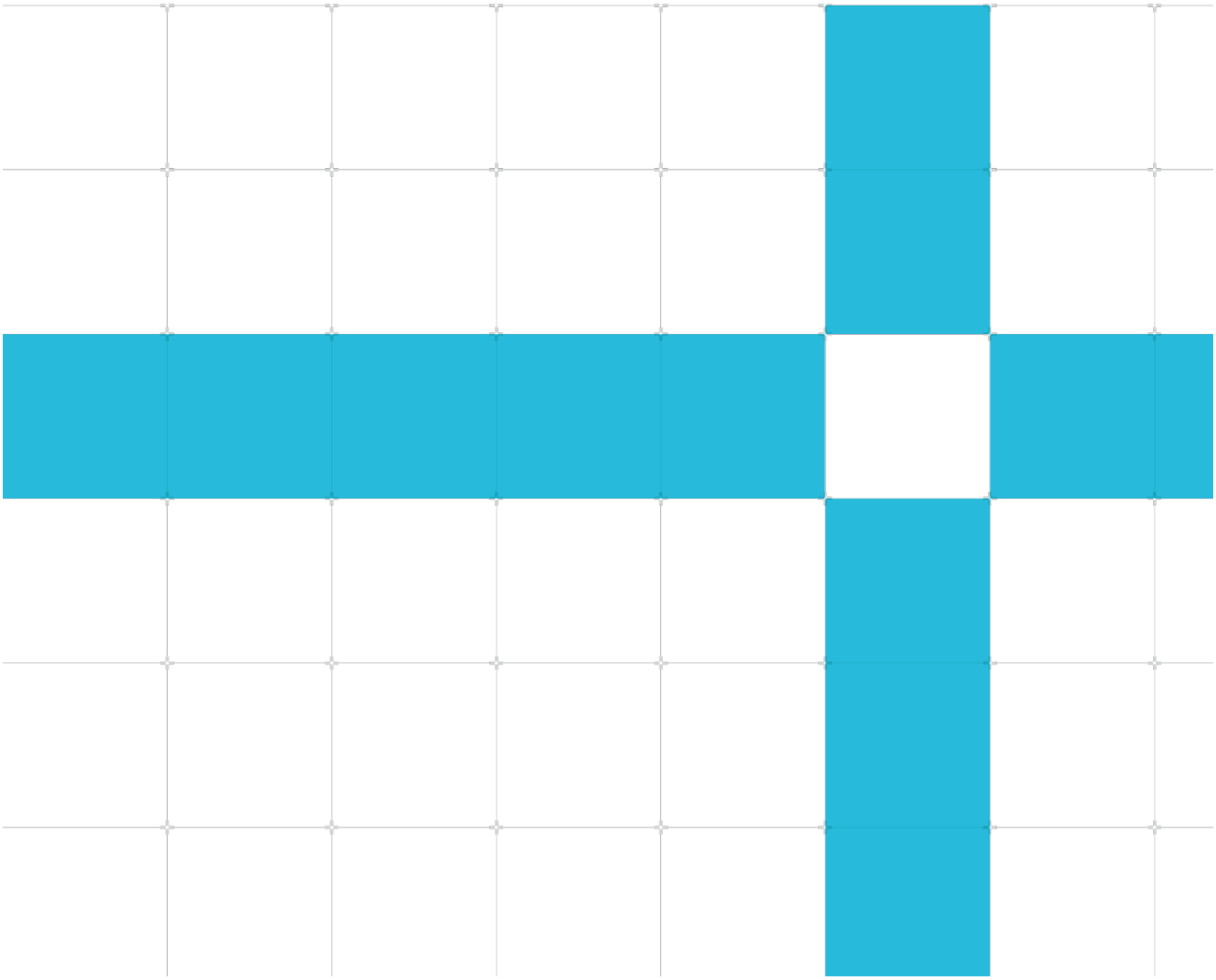
Date of issue: 01-Apr-2022

Non-Confidential

Document version: v6.0

Document ID: SDEN-1344564

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®-A65AE (MP082), 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.

01-Apr-2022: Changes in document version v6.0

ID	Status	Area	Category	Summary
2599525	New	Programmer	Category B (rare)	Completion of affected memory accesses might not be guaranteed by completion of a TLBI
2599522	New	Programmer	Category C	Atomic store instructions might not report an External abort or System Error

16-Nov-2021: Changes in document version v5.0

No new or updated errata in this document version.

28-Feb-2020: Changes in document version v4.0

ID	Status	Area	Category	Summary
1638563	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 v3.0

ID	Status	Area	Category	Summary
1638571	New	Programmer	Category B	Speculative AT instruction using out-of-context translation regime might cause subsequent request to generate an incorrect translation
1638563	New	Programmer	Category C	Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation
1638565	New	Programmer	Category C	Cache debug Read operations might not correctly return cache data

29-Mar-2019: Changes in document version v2.0

ID	Status	Area	Category	Summary
1344572	Updated	Programmer	Category C	VFP_SPEC and ASE_SPEC PMU events count incorrectly
1415199	New	Programmer	Category C	TLB Maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs
1443418	New	Programmer	Category C	Cycle count value in timestamp packet might be incorrect
1443622	New	Programmer	Category C	Incorrect ETM timestamp value when timestamp and event generation happen on same cycle

07-Dec-2018: Changes in document version v1.0

ID	Status	Area	Category	Summary
1344569	New	Programmer	Category C	Loads of mismatched size might not be single-copy atomic
1344572	New	Programmer	Category C	VFP_SPEC and ASE_SPEC PMU events count incorrectly

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
1638571	Programmer	Category B	Speculative AT instruction using out-of-context translation regime might cause subsequent request to generate an incorrect translation	r0p0, r1p0, r1p1	Open
2599525	Programmer	Category B (rare)	Completion of affected memory accesses might not be guaranteed by completion of a TLBI	r0p0, r1p0, r1p1	Open
2599522	Programmer	Category C	Atomic store instructions might not report an External abort or System Error	r0p0, r1p0, r1p1	Open
1638565	Programmer	Category C	Cache debug Read operations might not correctly return cache data	r0p0	r1p0
1638563	Programmer	Category C	Cache debug data register Read operations result in UNDEFINED exception when following the mnemonics specified in product documentation	r0p0, r1p0, r1p1	Open
1443622	Programmer	Category C	Incorrect ETM timestamp value when timestamp and event generation happen on same cycle	r0p0, r1p0, r1p1	Open
1443418	Programmer	Category C	Cycle count value in timestamp packet might be incorrect	r0p0, r1p0, r1p1	Open
1415199	Programmer	Category C	TLB Maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs	r0p0	r1p0
1344572	Programmer	Category C	VFP_SPEC and ASE_SPEC PMU events count incorrectly	r0p0, r1p0, r1p1	Open
1344569	Programmer	Category C	Loads of mismatched size might not be single-copy atomic	r0p0	r1p0

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

1638571
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. 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.

Category B (rare)

2599525
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. 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

2599522
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. 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.

1638565

Cache debug Read operations might not correctly return cache data

Status

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

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.

1638563

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. 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.

1443622
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. 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.

1443418
Cycle count value in timestamp packet might be incorrect

Status

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

1415199
TLB maintenance operations might not invalidate TLB entries in the presence of a persistent error in the TLB RAMs

Status

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

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.

1344572
VFP_SPEC and ASE_SPEC PMU events count incorrectly

Status

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

1344569
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 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.