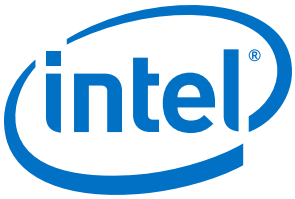

Intel® Stratix® 10 TX Device Errata

Subscribe
Send Feedback

ES-1073 | 2019.10.18
Latest document on the web: PDF | HTML

Contents

ES-1073 | 2019.10.18

Send Feedback

Intel® Stratix® 10 TX Device Errata

This document provides information about known device issues affecting the Intel® Stratix® 10 TX devices.

Intel Stratix 10 TX devices include:

- Intel Stratix 10 TX 400
- Intel Stratix 10 TX 850
- Intel Stratix 10 TX 1100
- Intel Stratix 10 TX 1650
- Intel Stratix 10 TX 2100
- Intel Stratix 10 TX 2500
- Intel Stratix 10 TX 2800

This document includes:

- Intel-Specific Errata for the Intel Stratix 10 TX Devices
- Arm* Cortex*-A53 MPCore* Processor and CoreSight* Errata

Note: To obtain third-party IP errata that applies to the HPS and is under NDA, please contact Intel or your local field representative.

Intel-Specific Errata for the Intel Stratix 10 TX Devices

This section lists the Intel-specific Errata that apply to the Hard Processor System (HPS) and the FPGA of the Intel Stratix 10 TX devices. Each listed erratum has an associated status that identifies any planned fixes.

Note: Currently, the Intel Stratix 10 TX devices have no Intel-specific FPGA errata.

Issue	Affected Devices	Planned Fix
Hard Processor System (HPS)		
HPS stops on the first read request to SDRAM	• Intel Stratix 10 TX 850 • Intel Stratix 10 TX 1100 • Intel Stratix 10 TX 2500 • Intel Stratix 10 TX 2800	No planned fix
FPGA		
None		

Hard Processor System

HPS Stops on the First Read Request to SDRAM

Description

If the Intel Stratix 10 HPS External Memory Interface (EMIF) is enabled in DDR x32 or x16 mode with or without ECC (IO48 Banks 2M/2N), and the IO48 Bank 2L is being used by the core fabric in specific use cases such as:

- LVDS with Dynamic Phase Alignment (DPA)
- LVDS without DPA and with Clock Phase Alignment (CPA) engaged
- Fabric-EMIF
- PHYLite

then the HPS stops on the first read request to SDRAM, and cannot be recovered.

Workaround

When using the Intel Stratix 10 HPS EMIF in DDR x32 or DDR x16 configuration with or without ECC (IO48 Banks 2M, 2N), the IO48 Bank 2L can only be used for LVDS without DPA and without CPA engaged, or GPIO purposes. All other banks have no restrictions and they can be used for LVDS, or Fabric-EMIF, or PHYLite, or GPIO.

Status

Affects:

- Intel Stratix 10 TX 850
- Intel Stratix 10 TX 1100
- Intel Stratix 10 TX 2500
- Intel Stratix 10 TX 2800

Status: No planned fix

Related Information

Intel Stratix 10 High-Speed LVDS I/O User Guide
For more information about LVDS.

Arm Cortex-A53 MPCore and CoreSight Errata

This section lists the Arm Cortex-A53 MPCore and CoreSight errata.

Note: This errata only applies if you are using the Hard Processor System (HPS).

Each listed erratum has an associated category number which identifies the degree of the behavior.

The categories are as follows:

- Category 1: Behavior has no workaround and severely restricts the use of the product in all, or the majority of applications, rendering the device unusable.
- Category 2: Behavior contravenes the specified behavior and might limit or severely impair the intended use of the specified features, but does not render the product unusable in all or the majority of applications.
- Category 3: Behavior that was not the originally intended behavior but should not cause any problems in applications.

Note: This device only contains category 2 and category 3 errata.

Table 1. Arm Cortex-A53 MPCore Processor and CoreSight Errata

Errata Listing	Category Number
843819: Memory Locations May be Accessed Speculatively Due to Instruction Fetches When HCR.VM is Set on page 6	Category 2
845719: A Load May Read Incorrect Data on page 7	Category 2
855871: ETM Does Not Report IDLE State When Disabled Using OSLOCK on page 8	Category 2
855872: A Store-Exclusive Instruction May Pass When it Should Fail on page 9	Category 2
711668: Configuration Extension Register Has Wrong Value Status on page 11	Category 3
720107: Periodic Synchronization Can Be Delayed and Cause Overflow on page 12	Category 3
855873: An Eviction Might Overtake a Cache Clean Operation on page 14	Category 3
853172: ETM May Assert AFREADY Before All Data Has Been Output on page 15	Category 3
836870: Non-Allocating Reads May Prevent a Store Exclusive From Passing on page 16	Category 3
836919: Write of the JMCR in EL0 Does Not Generate an UNDEFINED Exception on page 18	Category 3
845819: Instruction Sequences Containing AES Instructions May Produce Incorrect Results on page 19	Category 3
851672: ETM May Trace an Incorrect Exception Address on page 20	Category 3
851871: ETM May Lose Counter Events While Entering WFx Mode on page 21	Category 3
852071: Direct Branch Instructions Executed Before a Trace Flush May be Output in an Atom Packet After Flush Acknowledgment on page 22	Category 3
852521: A64 Unconditional Branch May Jump to Incorrect Address on page 23	Category 3
855827: PMU Counter Values May Be Inaccurate When Monitoring Certain Events on page 24	Category 3
855829: Reads of PMEVCNTR<n> are not Masked by HDCR.HPMN on page 26	Category 3
855830: Loads of Mismatched Size May not be Single-Copy Atomic on page 27	Category 3

ARM Cortex-A53 MPU and CoreSight Errata

843819: Memory Locations May be Accessed Speculatively Due to Instruction Fetches When HCR.VM is Set

Description

The ARMv8 architecture requires that when all associated stages of translation are disabled for the current exception level, software only perform instruction fetches within the same or next translation granule as an instruction which has been or will be fetched due to sequential execution. In the conditions detailed below, this erratum may cause the Cortex-A53 MPCore processor to access other locations speculatively due to instruction fetches.

For this erratum to occur:

- The CPU must be executing at EL3, EL2 or Secure EL1.
- The CPU can be in either AArch32 or AArch64 execution state.
- Address translation is disabled for the current exception level (by clearing the appropriate `SCTLR.M`, `HSCTLR.M` or `SCTLR_ELx.M` bit).
- The `HCR.VM` or `HCR_EL2.VM` bit is set.

Impact

If the above conditions are met, then speculative instruction fetches may be made to memory locations not permitted by the architecture.

Workaround

Because the `HCR.VM` bit default reset value is low, this situation is most likely to occur in power down code, if EL2 or EL3 software disables address translation before the core is powered down. To work around this erratum, software must clear the `HCR.VM` bit before disabling address translation at EL3, EL2 or Secure EL1.

Category

Category 2

845719: A Load May Read Incorrect Data

Description

When executing in the AArch32 state at EL0, the Cortex-A53 MPCore processor may read incorrect data if a load is performed to the same offset within a different 4 GB region as a recent previous load in the AArch64 state.

The following sequence must occur for this erratum to be triggered:

Note: This sequence requires execution in both AArch32 and AArch64, therefore at least one exception level change is required.

1. At EL0 or EL1, in the AArch32 or AArch64 state, the CPU executes a load, store, preload or data- or instruction-cache maintenance by MVA instruction.
2. At EL0 or EL1 in the AArch64 state, the CPU executes a load with:

   ```
   VA[63:32] != 32'h00000000
   ```

3. At EL0 in AArch32 state, the CPU executes a load to the same 4 KB region as the instruction in step 1, with `VA[31:6]` matching the VA from the instruction in step 2.

The erratum does not apply if any of the following occurs between step 1 and step 3 in the sequence:

- A write to any of the following registers:
  - Any `SCTLR`
  - Any `TTBR`
  - Any `TCR`
  - Any `MAIR`
  - `CONTEXTIDR`
- A TLB maintenance instruction is executed, followed by a DSB.

This erratum also does not apply if an address translation instruction executes between steps 2 and 3 in the sequence.

Impact

If the above conditions are met, data corruption can occur. The load at EL0 can access data written at EL1 for which it does not have read permissions; however, a load in non-secure state cannot access secure data.

Workaround

If one of the exception conditions listed after the sequence above is applied between steps 2 and 3, this erratum does not occur. Because there must be an exception return from AArch64 to AArch32 between positions 2 and 3 in the above sequence, the recommended workaround is to insert a write of the `CONTEXTIDR_EL1` register into operating system exception return sequences.

Category

Category 2

855871: ETM Does Not Report IDLE State When Disabled Using OSLOCK

Description

The OS Lock feature in the Embedded Trace Macrocell (ETM) allows software running on the Cortex-A53 MPCore to disable external debug access and save the register state before powering down the ETM. Software must follow a defined sequence to ensure that the register state is stable and all trace data is output before the system is powered-down. Because of this erratum, when the OS Lock mechanism is used, the ETM never indicates to software that it is safe to be powered-off. This erratum occurs when:

- The ETM is enabled (`TRCPRGCTLR.EN = 1`)
- The OS Lock feature disables the ETM (`TRCOSLAR.OSLK = 1`)

Impact

When the OS lock feature is enabled and software polls `TRCSTATR.IDLE`, it remains high even after the ETM is disabled and drained of trace data. If the ETM is already disabled by clearing the `TRCPRGCTLR.EN` bit then a software save and restore sequence behaves correctly.

Workaround

To execute the disable sequence properly:

1. Disable the ETM by setting the `TRCOSLAR.OSLK` bit.
2. Save the state of the `TRCPRGCTLR.EN` bit.
3. Clear the `TRCPRGCTLR.EN` bit.

Category

Category 2

855872: A Store-Exclusive Instruction May Pass When it Should Fail

Description

The Cortex-A53 MPCore processor implements an internal exclusive monitor to manage load-exclusive, store-exclusive, and clear-exclusive instructions. Because of this erratum, a load-exclusive instruction to cacheable memory may set the monitor to the exclusive state when the processor does not have exclusive access to the line. A subsequent store-exclusive instruction may pass when it should fail.

This erratum affects all load-exclusive and store-exclusive instructions, including load-acquire exclusive and store-release exclusive instructions.

The conditions that trigger this erratum are listed below:

- A core executes a store to memory that is marked as both inner-writeback and outer-writeback.
- The store is not a store-exclusive (or a store-release exclusive) or a store-release instruction.
- The store is not followed by a DMB SY or DSB.
- The store misses in the L1 data cache.
- The store does not trigger a linefill. This requires one or more of the following to be true:
  - The core is in read-allocate mode
  - The memory is marked as no-write-allocate
  - The memory is marked as transient
  - The store is a STNP instruction
  - The store is triggered by a DC ZVA instruction
- The core starts a linefill to the same address as the store. The linefill is started for one of the following:
  - A PRFM, PLD, or PLDW instruction
  - An automatic data prefetch
  - A pagewalk
- The core executes a load-exclusive (or a load-acquire exclusive) instruction to the same address as the store
- The store data is forwarded to the load-exclusive instruction
- If a core starts a linefill to the same address as the store, the load-exclusive instruction retires before the linefill is serialized.

Impact

If the above conditions are met then the Cortex-A53 MPCore processor may set the internal exclusive monitor. This behavior is incorrect because the processor is not guaranteed to have exclusive access to the line. If another master executes a load-exclusive instruction to the same address then both masters may gain access to an exclusive region of code at the same time.

Workaround

The only workaround is to avoid the conditions described above. Disabling read-allocate mode by setting CPUACTLR.RADIS to 0x3 degrades write-stream performance. Therefore, the preferred workaround is to use an appropriate store-exclusive (or a store-release exclusive) or a store-release instruction or DMB instruction.

Category

Category 2

711668: Configuration Extension Register Has Wrong Value Status

Description

The Program Trace Macrocell (PTM) implements several read only registers that provide a mechanism for tools using the PTM to determine which features are present in a specific implementation. The `Configuration Code Extension` register (0x7A, address 0x1E8) has an incorrect value of 0x000008EA. The correct value is 0x00C019A2.

Impact

This erratum has no impact on the generation of trace or the configurations that can be enabled. Tools that read this register in order to determine the capabilities of the PTM detect fewer features than are actually present.

The missing bits include:

- Bit[23] - Return stack implemented.
- Bit[22] - Timestamp Implemented.
- Bit[12] - Reserved, for compatibility with the ETM architecture

Bits[10:3] are incorrect and should indicate 52 external inputs.

Workaround

Tools using the PTM-A9 can read the `peripheral ID` registers (at offsets 0xFE0 to 0xFEC) to determine the version of the PTM-A9 and the features which are implemented, rather than relying on the `Configuration Code Extension` register.

Category

Category 3

720107: Periodic Synchronization Can Be Delayed and Cause Overflow

Description

The Program Trace Macrocell (PTM) is required to insert synchronization information into the trace stream at intervals in order to allow a partial trace dump to be decompressed. The synchronization period can be controlled either external to the PTM or by a counter that by default, requests synchronization every 1024 bytes of trace.

Synchronization does not need to be inserted precisely at a regular interval, so allowance is made to delay synchronization if the PTM is required to generate other trace packets, or if there is not sufficient space in the FIFO. To guarantee that some synchronization packets are always inserted regardless of the conditions, a 'forced overflow' mechanism shuts off trace if a new synchronization request occurs before the previous request was satisfied. This forced overflow mechanism is minimally intrusive to the trace stream, but ensures that synchronization is inserted after no more than 2x the requested interval.

Due to this erratum, some specific sequences of instructions prevent the PTM from being able to insert any synchronization into the trace stream while that instruction sequence continues. Typically, there is just a short delay which may not be noticed and the synchronization is inserted once the particular pattern of waypoints changes. It is possible that overflows are generated in the trace regardless of the utilization of the FIFO in the PTM. In this scenario, typically only a single byte of trace (up to 5 waypoints) is lost.

The scenario does not correspond to sustained high rates of trace generation which could genuinely cause the FIFO to become full.

Scenarios that cause this erratum are rare, and are limited to code iterating around a loop many times. The loop would contain several branches and be dependent on memory accesses completing.

This erratum can occur under the following conditions:

1. Tracing is enabled.
2. Branch broadcasting is disabled.
3. Cycle accuracy is disabled.
4. The processor executes tight loops of repeating code, lasting longer than the configured synchronization period.

Impact

This erratum reduces the frequency of periodic synchronization and potentially causes trace overflows where some trace is lost. In the case of an overflow, trace following the overflow can be correctly decompressed. This erratum is more noticeable if the periodic synchronization requests are more frequent.

Workaround

The most appropriate workaround depends on the use-case for the trace:

- If the trace buffer is large enough, consider increasing the synchronization period by setting the ETMSYNCFR to a higher value.
- If no trace must be lost (and periodic synchronization does not have to be guaranteed), set `bit[0]` of ETMAUXCR to disable the forced overflow function. Synchronization is inserted at the earliest opportunity, dependent on the executed instruction stream.

Category

Category 3

855873: An Eviction Might Overtake a Cache Clean Operation

Description

The Cortex-A53 MPCore processor supports instructions for cache clean operations. To avoid data corruption, the processor must ensure correct ordering between evictions and cache clean operations for the same address.

Because of this erratum, the processor can issue an eviction and an L2 cache clean operation to the interconnect in the wrong order. The processor can also issue transactions that become outstanding in the interconnect at the same time. This situation violates the AXI Coherency Extensions (ACE) protocol specification.

This erratum occurs when the following conditions are met:

1. One or both of the following are true:
   - `L2ACTLR[14]` is set to 1. This setting allows WriteEvict transactions on the ACE interface when the processor evicts data that it holds in the UniqueClean state.
   - `L2ACTLR[3]` is set to 0. This setting allows Evict transactions on the ACE interface when the processor evicts clean data.
2. A CPU executes a cache clean-by-address operation for a line that is present and dirty in the L2 cache.
3. A CPU performs any type of memory access to the same set. Memory access types can include a pagewalk, instruction fetch, cache maintenance operation or data access.
4. An instruction fetch to the same set triggers an L2 cache eviction.
5. A present and dirty cache line that is selected for L2 cache eviction at the same time it is targeted for a cache clean operation.

Impact

Because of this erratum transactions are erroneously reordered in the interconnect, resulting in data corruption.

Workaround

You can workaround this erratum by changing the cache clean-by-address operations to cache clean-and-invalidate operations. To enable these operations, set `CPUACTLR.ENDCCASCI` to 1.

Category

Category 2

853172: ETM May Assert AFREADY Before All Data Has Been Output

Description

When the AFVALID signal on the Advanced Trace Bus (ATB) interface asserts, the ETM immediately outputs all buffered trace. The ETM must assert the AFREADY output one cycle after all trace that was buffered on the cycle in which AFVALID was first asserted.

Because of this erratum, the AFREADY signal may be asserted before all the necessary trace has been output.

Impact

Because of this erratum, the ETM may contain trace data that was generated before the flush request.

Workaround

The system can ensure that all trace has been drained from the ETM by disabling it. You can disable the ETM by clearing the TRCPRGCTLR.EN bit. Next, system software must poll the TRCSTATR.IDLE bit until it reads as 1. This value indicates the ETM is idle and all trace that was generated before the write to TRCPRGCTLR has been output.

Category

Category 3

836870: Non-Allocating Reads May Prevent a Store Exclusive From Passing

Description

A Cortex-A53 MPCore processor executing a load and store exclusive instruction in a loop is allowed to repeatedly fail under certain conditions. An example of an allowed condition is when another processor repeatedly writes to the same cache line.

Because of this erratum, a non-allocating load from another processor may cause the store-exclusive instruction to repeatedly fail.

The failure occurs under these conditions:

- One CPU executes a loop containing a load-exclusive and a store-exclusive instruction. The loop continues until the store-exclusive instruction succeeds.
- Another CPU or master in the system repeatedly executes a load to the same L1 data cache index as the store-exclusive instruction.
- The cache line containing the address of the load is present in the L1 cache of the CPU, and is not present in the L2 cache.
- The load does not cause an allocation into the cache of the CPU or the master executing the load. This response results in a snoop arriving at the CPU holding the cache line every time the load is performed.
- The load is repeated at exactly the same frequency as the load- and store-exclusive loop, such that every time around the loop, the snoop arrives at the same time as the store exclusive instruction is executing.

A load can be non-allocating in Cortex-A53 MPCore processor in the following conditions:

- A load instruction executes on a CPU, while the `CPUACTLR.DTAH` bit is not set, with one of the following conditions:
  - The memory address is marked as writeback cacheable, with no read allocate, in the page tables.
  - The memory address is marked as writeback cacheable, transient, in the page tables.
  - The memory address is marked as writeback cacheable, and an `LDNP` non-temporal load instruction is used.
- A read transaction is made on the ACP interface.

Impact

If a CPU or other master is polling a location to determine when the value changes, then it can prevent another CPU from updating that location, causing a software livelock. Most polling routines, however, use memory that can be allocated into their cache. For improved efficiency it is recommended to use a load-exclusive and `WFE` instruction to avoid repeated polling. Because the frequency of the polling loop must match the frequency of the load store-exclusive loop, the conditions that cause this erratum are unlikely. Any other disturbance in the system, such as an interrupt or other bus traffic can easily alter the frequency of either loop sufficiently to break the livelock.

Workaround

If the repeating load is being executed on a Cortex-A53 CPU, then this erratum can be worked around by setting the CPUACTLR.DTAH bit. Note this version of the Cortex-A53 MPCore processor sets this bit by default.

If the repeating load is being executed by another master in the system connected to the ACP interface, then the erratum can be worked around by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop.

If the repeating load is being executed by another master in the system connected through the coherent interconnect, then the erratum can be worked around either by ensuring that the other master allocates the line into its cache, or by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop. Note that the Cortex-A53 MPCore processor always allocates the line into either the cache or a temporary buffer, and so does not require any workaround for typical polling loops.

Category

Category 3

836919: Write of the JMCR in EL0 Does Not Generate an UNDEFINED Exception

Description

When EL0 is using AArch32 register width and a write is performed in EL0 to the Arm Jazelle® Main Configuration Register (JMCR), the write should be UNDEFINED. Because of this erratum, the write is permitted but ignored.

The erratum occurs under the following conditions:

- The processor is executing in AArch32 user mode.
- A write to the JMCR is executed, using the instruction MCR `p14,7,<Rt>,c2,c0,0`.

Impact

Rather than treating the MCR instruction as UNDEFINED, a write to the JMCR is ignored.

Workaround

No available workaround.

Category

Category 3

845819: Instruction Sequences Containing AES Instructions May Produce Incorrect Results

Description

When the Cortex-A53 MPCore processor is executing in the AArch64 state, certain sequences of instructions that include AES instructions may cause incorrect results.

There are two code sequences that can cause this erratum in the AArch64 execution state:

- Code sequence 1:
  1. The CPU executes an AESE instruction.
  2. The CPU executes a USQADD instruction.
     - Both the Vn and Vd of this instruction must be the same register as Vd for the AES instruction.
     - The size field for this instruction must be '00', indicating byte-sized elements.
     - The USQADD instruction can be in either vector or scalar form.
- Code sequence 2:
  1. The CPU executes a SUQADD instruction.
     - The size field for this instruction must be 00, indicating byte-sized elements.
     - The SUQADD instruction can be in either vector or scalar form.
  2. The CPU executes an AESMC or AESIMC instruction.
     - Both the Vn and Vd of this instruction must be the same register as the Vd for the SUQADD instruction.

For both these sequences, the two instructions listed must be executed consecutively, with no other instructions or exceptions between them. If either of these sequences is met, the final result of the affected instruction sequence may be incorrect.

Impact

The sequences of instructions described in the conditions above are not expected to occur in real code because they do not perform useful computation. Therefore, there is no impact expected to real systems.

Workaround

Because the code sequences for this erratum are not expected to occur in real code, no workaround is required.

Category

Category 3

851672: ETM May Trace an Incorrect Exception Address

Description

The address in an exception packet is the preferred exception return address. Because of this erratum, the address may be equal to the target address of the exception. The trace stream is not corrupted, and decompression can continue after the affected packet.

The following sequence is required for this erratum to occur:

1. The ETM must start tracing instructions because of one of the following:
   a. `ViewInst` starts.
   b. The security state changes indicating trace is now permitted.
   c. The values of the external debug interface (`DBGEN`, `SPIDEN`, `NIDEN`, `SPNIDEN`) change to indicate trace is now permitted.
   d. The CPU exits debug mode.
2. Before the CPU executes any other behavior that causes a trace to be generated, it executes a direct branch instruction, which may or may not be taken.
3. The next instruction is a load or store that causes a data abort or watchpoint exception.

After this sequence, provided certain timing specific conditions are met, the address in the exception packet may be incorrect.

Impact

The trace decompressor may incorrectly infer execution of many instructions from the branch target to the provided address.

Workaround

The trace decompressor can detect that this erratum has occurred by checking if the exception address is in the vector table and identifying if the branch was not expected to be taken to the vector table. A decompressor can infer the correct address of the exception packet. The target of the preceding branch (if the branch was taken), or the next instruction after the branch (if the branch was not-taken) provides the correct address of the exception packet.

Category

Category 3

851871: ETM May Lose Counter Events While Entering WFx Mode

Description

If the ETM resources become inactive because of a low-power state, there is a one-cycle window during which the counters and the sequencer may ignore counter-at-zero resources.

The following sequence is required for this erratum to occur:

1. The core executes a `WFI` or `WFE` instruction.
2. The ETM enters a low-power state.
3. In a one-cycle window around this point, either:
   a. A counter in self-reload mode generates a counter-at-zero resource.
   b. A counter in normal mode gets a `RLDEVENT` on the cycle in which it has just transitioned to zero.
4. A counter or sequencer is sensitive to the counter-at-zero resource.

Impact

Counters sensitive to a counter-at-zero resource may not reload or decrement. If the sequencer is sensitive to a counter-at-zero resource, it may not change state, or may change to an incorrect state.

Workaround

The ETM can be prevented from entering low-power mode by setting the `LPOVERRIDE` bit of Trace Event Control 1 (`TRCEVENTCTL1R`) register. This workaround is only needed if there is a counter or sequencer sensitive to a counter-at-zero resource.

Category

Category 3

852071: Direct Branch Instructions Executed Before a Trace Flush May be Output in an Atom Packet After Flush Acknowledgment

Description

The Embedded Trace Macrocell (ETMv4) architecture requires that when a trace flush is requested on the AMBA Trace Bus (ATB), a Cortex-A53 MPCore processor must complete any packets that are in the process of being encoded and output them prior to acknowledging the flush request. When trace is enabled, the Cortex-A53 MPCore processor attempts to combine multiple direct branch instructions into a single atom packet. If a direct branch instruction is executed, and an atom packet is in the process of being generated, the processor does not force completion of the packet prior to acknowledging the flush request. This erratum is a violation of the ETMv4 architecture.

The following conditions are required for this erratum to occur:

1. ETM is enabled.
2. Instruction tracing is active.
3. The processor executes one or more direct branch instructions.
4. An atom packet is in the process of being encoded but is not complete.
5. A trace flush is requested on the AMBA ATB.

Impact

When the above conditions occur, the ETM must complete encoding and output the atom packet prior to the trace flush request being acknowledged. Because of this erratum, the atom packet is output after the flush is acknowledged. Therefore, it appears to software monitoring the trace that the direct branch was executed after the requested flush.

Workaround

Enabling the timestamp by setting the TS bit in the Trace Configuration (`TRCCONFIGR`) register resolves the erratum because the atom packets complete when timestamp behavior is enabled.

Category

Category 3

852521: A64 Unconditional Branch May Jump to Incorrect Address

Description

When executing in AArch64 state with address translation disabled, unconditional immediate branch instructions might jump to an incorrect address.

The following conditions are required for this erratum to occur:

1. The processor is executing in AArch64 state.
2. The `SCTLR_ELx.M` bit for the current exception level is 0.
3. The `HCR_EL2.VM` bit is 0.
4. A B or BL instruction from the unconditional branch (immediate) encoding class is executed.
5. This branch has an `imm26` field of `0x1FFFFFF`, encoding a branch target of `{pc}+0x7FFFFC`.

Impact

If these conditions are met, then the processor might incorrectly branch to the target `{pc}-0x8000004` instead of `{pc}+0x7FFFFC`.

Workaround

The workaround for this erratum is to avoid the conditions described.

Category

Category 3

855827: PMU Counter Values May Be Inaccurate When Monitoring Certain Events

Description

The Cortex-A53 MPCore processor implements a Performance Monitor Unit (PMU). The PMU allows programmers to gather statistics on the operation of the processor during runtime. Because of this erratum, software may read inaccurate PMU counter values when monitoring certain events. Specifically:

- The INST_RETIRED event counts architecturally executed instructions. Because of this erratum, it may count more instructions than were architecturally executed.
- The ST_RETIRED event does not count store-exclusive instructions that fail. Because of this erratum, it does count these instructions.
- The UNALIGNED_LDST_RETIRED event counts loads and stores that fail their alignment check. Because of this erratum, it may also count LDRD and STRD instructions that pass their alignment check.
- The EXC_TAKEN and EXC_RETURN events are filtered precisely according to the exception level/security state they were executed in. Because of this erratum, they are filtered according to the destination exception level/security state.

This erratum is present when:

- A performance counter is enabled and configured to count an INST_RETIRED , ST_RETIRED, UNALIGNED_LDST_RETIRED, EXC_TAKEN, or EXC_RETURN event.
- The following conditions occur during the event capture:
  - INST_RETIRED: An immediate branch instruction is executed.
  - ST_RETIRED: A store-exclusive instruction is executed and fails.
  - UNALIGNED_LDST_RETIRED: A LDRD or STRD instruction is executed that is word aligned but not double word aligned.
  - EXC_TAKEN: An exception is taken.
  - EXC_RETURN: An exception return is executed.
- The INST_RETIRED, ST_RETIRED, and UNALIGNED_LDST_RETIRED filter settings for the performance counter are configured so that an event must be counted if it occurs.
- The EXC_TAKEN and EXC_RETURN filter settings are configured to one of the following descriptions:
  - Events must be counted in the original exception level and security state, but must not be counted in the exception level and security state following the exception or the exception return
  - Events must not be counted in the original exception level and security state, but must be counted in the exception level and security state following the exception or the exception return.

Impact

If the erratum conditions are met, the performance counter may erroneously increment or fail to increment. Specifically:

- INST_RETIRED: The counter may erroneously increment by two when only one instruction executes.
- ST_RETIRED: The counter may erroneously increment for the failed store-exclusive instruction.
- UNALIGNED_LDST_RETIRED: The counter erroneously increments for the LDRD or STRD instruction.
- EXC_TAKEN: The counter may erroneously increment or erroneously fail to increment.
- EXC_RETURN: The counter may erroneously increment or erroneously fail to increment.

Workaround

For the EXC_TAKEN and EXC_RETURN events, you can workaround the erratum by changing the filter settings for the performance counter to monitor the destination exception level and security state instead of the exception level and security state in which the exception or exception return are executed.

There are no workarounds for the other PMU events.

Category

Category 3

855829: Reads of PMEVCNTR<n> are not Masked by HDCR.HPMN

Description

Software executing at EL2 can set the HDCR.HPMN and MDCR_EL2.HPMN field to restrict non-secure EL0 and EL1 software from accessing a subset of the performance counters. If n > HPMN, then Performance Counter n (PMEVCNTR<n>) register is not accessible to non-secure EL0 or EL1. Because of this erratum, Cortex-A53 reads of inaccessible registers might not return zero as expected, but instead might read the actual contents of the counter registers.

For this erratum to be present:

1. The processor is executing at non-secure EL1 or EL0.
2. HDCR.HPMN and MDCR_EL2.HPMN are set to a value n, where n < 6.
3. A read of PMEVCNTR<n> and PMEVCNTR<n>_EL0 is executed.

Impact

If the conditions described above are met, then non-secure EL0 and EL1 software can read the values of performance counters reserved for EL2.

Workaround

Software executing at EL2 can set the HDCR.TPM and MDCR_EL2.TPM bit. This configuration causes all non-secure EL0 and EL1 accesses to performance monitor registers to trap to EL2. The EL2 software can then emulate accesses as required, and can mask out accesses to reserved registers.

Category

Category 3

855830: Loads of Mismatched Size May not be Single-Copy Atomic

Description

The Cortex-A53 MPCore processor supports single-copy atomic load and store accesses as described in the ARM architecture documentation. However, in some unusual code sequences, this erratum can cause the CPU executing a store, and later a load, to the same address but with a different access size to load data that does not meet the requirements of a single-copy atomic load.

For this erratum to be present, a store instruction must execute on one CPU. This instruction can be any halfword, word, or double word store instruction that is not a store release and the address must be aligned to the access size. On a second CPU, the following sequence must occur:

1. A store instruction executes. This store must be a smaller access size than the store on the first CPU, and must address bytes accessed by the first CPU. The address must also be aligned to the access size.
2. The store instruction does not allocate into the cache because of any one of the following conditions:
   - The memory address is marked as transient
   - The write allocate hint in the translation table is not set or the memory is marked as non-cacheable
   - The CPU recently executed a stream of stores and has subsequently dynamically switched into a no-write allocate mode
3. A load instruction executes. The load must be a larger access size than the store from the same CPU, and at least some bytes of the load must be to the same address as the store. The address must also be aligned to the access size.

The Arm architecture requires that the load is single-copy atomic. However, in the conditions described, the load may observe a combination of the two stores, indicating that the store on the initial CPU was serialized first. If the load is repeated, it might only see the data from the first CPU's store, indicating that the store on the first CPU was serialized second.

Impact

Concurrent, unordered stores are uncommon in multi-threaded code. In the ISO/IEC 9899:2011 (C11) standard, they are restricted to the family of relaxed atomics. Using different size load and store instructions to access the same data is also uncommon. For these reasons, the majority of multi-threaded software does not meet the conditions for this erratum.

Workaround

Most multi-threaded software does not satisfy the conditions of this erratum and therefore, does not require a workaround . If a workaround is required, then replace the store on the second CPU with a store-release instruction.

Category

Category 3

Intel Stratix 10 TX Device Errata Revision History

Document Version	Changes
2019.10.18	Initial release.