



Arm Cortex-X925 (MP210)

Software Developer Errata Notice

Date of issue: October 01, 2024

Non-Confidential

Document version: 9.0

Copyright © 2024 Arm® Limited (or its affiliates). All rights reserved.

Document ID: SDEN-2864216

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



This document is Non-Confidential.

Copyright © 2024 Arm® Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted Arm's Proprietary notice found at the end of this document.

This document (SDEN_2864216_9.0_en) was issued on October 01, 2024.

There might be a later issue at <http://developer.arm.com/documentation/SDEN-2864216>

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Cortex-X925 (MP210), 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>.

Contents

rOp0 implementation fixes	5
Introduction	6
Scope	6
Categorization of errata	6
Change Control	7
Errata summary table	10
Errata descriptions	13
Category A	13
2926479 The core might deadlock or data corruption might occur under certain micro-architectural conditions	13
2937969 Data corruption can occur under certain micro-architectural conditions when the PE executes an SVE store instruction	14
Category A (rare)	14
Category B	15
2921199 Execution of STG instructions in close proximity might cause loss of MTE allocation tag data	15
2922378 Branch prediction history not suppressed when switching from low to high EL	16
2933290 MTE Tag checking may not be performed if an access that crosses a 16-byte boundary encounters a poisoned Allocation tag	18
2963999 Incorrect virtualization of reads to MPIDR_EL1 and MIDR_EL1	19
2982189 PE executing DRPS during Debug Halt under Double Fault condition will not execute properly	21
3007699 SPE might write to pages which lack write permission at Stage-1 or Stage-2	22
3043240 Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock	24
3324334 MSR PSTATE.SSBS to 0 is not fully self-synchronizing	26
3701747 Read of ICH_VMCR_EL2.VBPR1 might return incorrect data based on SCR_EL3.NS	27
Category B (rare)	29
2917970 PE might incorrectly detect a Watchpoint debug event instead of a Data Abort exception on a page crossing memory access, resulting in errant entry to Debug state or routing the Data Abort exception to an incorrect Exception level	29
Category C	31
2871705 Noncompliance with prioritization of Exception Catch debug events	31
2871706 MPAM value associated with instruction fetch might be incorrect	33

2874250	Accessing a memory location using mismatched Shareability attributes when MTE tag checking is enabled might cause data corruption	34
2901777	PMU event MEM_ACCESS_CHECKED_WR incorrectly counts aborted or inactive stores in MTE precise mode	35
2910965	L2D_CACHE_WB_CLEAN overcounts	36
2911068	SPE latency counters are corrupted under certain conditions	37
2925547	Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported	38
2927450	PE might report an unexpected SEA or SError on a read access by a load instruction	39
3061575	TagMatch responses with error indication do not generate a SError abort	40
3384220	PMU event L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW count incorrectly	41
3563818	Incorrect decoding of SVE version of PRFH scalar plus vector (gather) instructions	42
3604846	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative	43
3605029	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed	44
3627351	PMU event STALL_SLOT_FRONTEND counts when instruction fetch is stalled for PCRF availability	46
3633453	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception	47
3640926	SPE operation type is corrupted under certain conditions	48
3694318	LS misses RAR hazard on case with clean critical beat and poisoned final response with ECC disabled	49
3694450	FFR might not capture the lowest faulting memory element	50
3699919	PE might fail to log a RAS error for L2 data RAM ECC errors	51
3705902	PMU events are mis-categorized by not considering the effect of "Taken locally"	52
3707125	Parity Error in L2BTB can cause failure in breakpoint detection	53
Proprietary notice		54
Product and document information		56
Product status		56
Product completeness status		56
Product revision status		56

rOp0 implementation fixes

Note the following errata might be fixed in some implementations of rOp0. This can be determined by reading the REVIDR_EL1 register where a set bit indicates that the erratum is fixed in this part.

REVIDR_EL1[0]	2926479 The core might deadlock or data corruption might occur under certain micro-architectural conditions
REVIDR_EL1[1]	2937969 Data corruption can occur under certain micro-architectural conditions when SVE store instruction is executed by PE

Note that there is no change to the MIDR_EL1 which remains at rOp0 but the REVIDR_EL1 is updated to indicate which errata are corrected. Software will identify this release through the combination of MIDR_EL1 and REVIDR_EL1.

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.

October 01, 2024: Changes in document version v9.0

ID	Status	Area	Category	Summary
3043240	New	Programmer	Category B	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock
3701747	New	Programmer	Category B	Read of ICH_VMCR_EL2.VBPR1 might return incorrect data based on SCR_EL3.NS
3627351	New	Programmer	Category C	PMU event STALL_SLOT_FRONTEND counts when instruction fetch is stalled for PCRF availability
3633453	New	Programmer	Category C	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception
3640926	New	Programmer	Category C	SPE operation type is corrupted under certain conditions
3694318	New	Programmer	Category C	LS misses RAR hazard on case with clean critical beat and poisoned final response with ECC disabled
3694450	New	Programmer	Category C	FFR might not capture the lowest faulting memory element
3699919	New	Programmer	Category C	PE might fail to log a RAS error for L2 data RAM ECC errors
3705902	New	Programmer	Category C	PMU events are mis-categorized by not considering the effect of "Taken locally"
3707125	New	Programmer	Category C	Parity Error in L2BTB can cause failure in breakpoint detection

May 29, 2024: Changes in document version v8.0

ID	Status	Area	Category	Summary
3563818	New	Programmer	Category C	Incorrect decoding of SVE version of PRFH scalar plus vector (gather) instructions
3604846	New	Programmer	Category C	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative
3605029	New	Programmer	Category C	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed

April 26, 2024: Changes in document version v7.0

ID	Status	Area	Category	Summary
3324334	New	Programmer	Category B	MSR PSTATE.SSBS to 0 is not fully self-synchronizing
3384220	New	Programmer	Category C	PMU event L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW count incorrectly

October 06, 2023: Changes in document version v6.0

ID	Status	Area	Category	Summary
3061575	New	Programmer	Category C	TagMatch responses with error indication do not generate a SError abort

September 07, 2023: Changes in document version v5.0

ID	Status	Area	Category	Summary
2926479	Updated	Programmer	Category A	The core might deadlock or data corruption can occur under certain micro-architectural conditions
2937969	Updated	Programmer	Category A	Data corruption can occur under certain micro-architectural conditions when the PE executes an SVE store instruction
2921199	Updated	Programmer	Category B	Execution of STG instructions in close proximity might cause loss of MTE allocation tag data
2922378	Updated	Programmer	Category B	Branch prediction history not suppressed when switching from low to high EL
2933290	Updated	Programmer	Category B	MTE Tag checking may not be performed if an access that crosses a 16-byte boundary encounters a poisoned Allocation tag
2963999	Updated	Programmer	Category B	Incorrect virtualization of reads to MPIDR_EL1 and MIDR_EL1
2982189	Updated	Programmer	Category B	PE executing DRPS during Debug Halt under Double Fault condition will not execute properly
3007699	New	Programmer	Category B	SPE might write to pages which lack write permission at Stage-1 or Stage-2
3043240	New	Programmer	Category B	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock
2917970	Updated	Programmer	Category B (rare)	PE might incorrectly detect a Watchpoint debug event instead of a Data Abort exception on a page crossing memory access, resulting in errant entry to Debug state or routing the Data Abort exception to an incorrect Exception level
2871706	Updated	Programmer	Category C	MPAM value associated with instruction fetch might be incorrect
2871705	Updated	Programmer	Category C	Noncompliance with prioritization of Exception Catch debug events
2874250	Updated	Programmer	Category C	Accessing a memory location using mismatched Shareability attributes when MTE tag checking is enabled might cause data corruption
2901777	Updated	Programmer	Category C	PMU MEM_ACCESS_CHECKED_WR incorrectly counts aborted or inactive stores in MTE precise mode
2910965	Updated	Programmer	Category C	L2D_CACHE_WB_CLEAN overcounts
2911068	Updated	Programmer	Category C	SPE latency counters are corrupted under certain conditions
2925547	Updated	Programmer	Category C	Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported
2927450	Updated	Programmer	Category C	PE might report an unexpected SEA or SError on a read access by a load instruction

July 07, 2023: Changes in document version v4.0

ID	Status	Area	Category	Summary
2921199	New	Programmer	Category B	Execution of STG instructions in close proximity might cause loss of MTE allocation tag data
2922378	New	Programmer	Category B	Branch prediction history not suppressed when switching from low to high EL
2963999	New	Programmer	Category B	Incorrect virtualization of reads to MPIDR_EL1 and MIDR_EL1
2982189	New	Programmer	Category B	PE executing DRPS during Debug Halt under Double Fault condition will not execute properly
2917970	New	Programmer	Category B (rare)	PE might incorrectly detect a Watchpoint debug event instead of a Data Abort exception on a page crossing memory access, resulting in errant entry to Debug state or routing the Data Abort exception to an incorrect Exception level
2911068	New	Programmer	Category C	SPE latency counters are corrupted under certain conditions
2925547	New	Programmer	Category C	Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported

May 25, 2023: Changes in document version v3.0

ID	Status	Area	Category	Summary
2937969	New	Programmer	Category A	Data corruption can occur under certain micro-architectural conditions when the PE executes an SVE store instruction
2933290	New	Programmer	Category B	MTE Tag checking may not be performed if an access that crosses a 16-byte boundary encounters a poisoned Allocation tag

May 12, 2023: Changes in document version v2.0

ID	Status	Area	Category	Summary
2926479	New	Programmer	Category A	The core might deadlock or data corruption can occur under certain micro-architectural conditions
2901777	New	Programmer	Category C	PMU MEM_ACCESS_CHECKED_WR incorrectly counts aborted or inactive stores in MTE precise mode
2910965	New	Programmer	Category C	L2D_CACHE_WB_CLEAN overcounts
2927450	New	Programmer	Category C	PE might report an unexpected SEA or SError on a read access by a load instruction

April 21, 2023: Changes in document version v1.0

ID	Status	Area	Category	Summary
2871706	New	Programmer	Category C	MPAM value associated with instruction fetch might be incorrect
2871705	New	Programmer	Category C	Noncompliance with prioritization of Exception Catch debug events
2874250	New	Programmer	Category C	Accessing a memory location using mismatched Shareability attributes when MTE tag checking is enabled might cause data corruption

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
2926479	Programmer	Category A	The core might deadlock or data corruption can occur under certain micro-architectural conditions	r0p0	r0p1
2937969	Programmer	Category A	Data corruption can occur under certain micro-architectural conditions when the PE executes an SVE store instruction	r0p0	r0p1
2921199	Programmer	Category B	Execution of STG instructions in close proximity might cause loss of MTE allocation tag data	r0p0	r0p1
2922378	Programmer	Category B	Branch prediction history not suppressed when switching from low to high EL	r0p0	r0p1
2933290	Programmer	Category B	MTE Tag checking may not be performed if an access that crosses a 16-byte boundary encounters a poisoned Allocation tag	r0p0	r0p1
2963999	Programmer	Category B	Incorrect virtualization of reads to MPIDR_EL1 and MIDR_EL1	r0p0	r0p1
2982189	Programmer	Category B	PE executing DRPS during Debug Halt under Double Fault condition will not execute properly	r0p0	r0p1
3007699	Programmer	Category B	SPE might write to pages which lack write permission at Stage-1 or Stage-2	r0p0	r0p1
3043240	Programmer	Category B	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock	r0p0, r0p1	Open
3324334	Programmer	Category B	MSR PSTATE.SSBS to 0 is not fully self-synchronizing	r0p0, r0p1	Open
3701747	Programmer	Category B	Read of ICH_VMCR_EL2.VBPR1 might return incorrect data based on SCR_EL3.NS	r0p0, r0p1	Open
2917970	Programmer	Category B (rare)	PE might incorrectly detect a Watchpoint debug event instead of a Data Abort exception on a page crossing memory access, resulting in errant entry to Debug state or routing the Data Abort exception to an incorrect Exception level	r0p0	r0p1

ID	Area	Category	Summary	Found in versions	Fixed in version
2871705	Programmer	Category C	Noncompliance with prioritization of Exception Catch debug events	r0p0, r0p1	Open
2871706	Programmer	Category C	MPAM value associated with instruction fetch might be incorrect	r0p0, r0p1	Open
2874250	Programmer	Category C	Accessing a memory location using mismatched Shareability attributes when MTE tag checking is enabled might cause data corruption	r0p0, r0p1	Open
2901777	Programmer	Category C	PMU MEM_ACCESS_CHECKED_WR incorrectly counts aborted or inactive stores in MTE precise mode	r0p0	r0p1
2910965	Programmer	Category C	L2D_CACHE_WB_CLEAN overcounts	r0p0	r0p1
2911068	Programmer	Category C	SPE latency counters are corrupted under certain conditions	r0p0	r0p1
2925547	Programmer	Category C	Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported	r0p0	r0p1
2927450	Programmer	Category C	PE might report an unexpected SEA or SError on a read access by a load instruction	r0p0	r0p1
3061575	Programmer	Category C	TagMatch responses with error indication do not generate a SError abort	r0p0, r0p1	Open
3384220	Programmer	Category C	PMU event L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW count incorrectly	r0p0, r0p1	Open
3563818	Programmer	Category C	Incorrect decoding of SVE version of PRFH scalar plus vector (gather) instructions	r0p0, r0p1	Open
3604846	Programmer	Category C	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative	r0p0, r0p1	Open
3605029	Programmer	Category C	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed	r0p0, r0p1	Open
3627351	Programmer	Category C	PMU event STALL_SLOT_FRONTEND counts when instruction fetch is stalled for PCRF availability	r0p0, r0p1	Open

ID	Area	Category	Summary	Found in versions	Fixed in version
3633453	Programmer	Category C	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception	r0p0, r0p1	Open
3640926	Programmer	Category C	SPE operation type is corrupted under certain conditions	r0p0, r0p1	Open
3694318	Programmer	Category C	LS misses RAR hazard on case with clean critical beat and poisoned final response with ECC disabled	r0p0, r0p1	Open
3694450	Programmer	Category C	FFR might not capture the lowest faulting memory element	r0p0, r0p1	Open
3699919	Programmer	Category C	PE might fail to log a RAS error for L2 data RAM ECC errors	r0p0, r0p1	Open
3705902	Programmer	Category C	PMU events are mis-categorized by not considering the effect of "Taken locally"	r0p0, r0p1	Open
3707125	Programmer	Category C	Parity Error in L2BTB can cause failure in breakpoint detection	r0p0, r0p1	Open

Errata descriptions

Category A

2926479

The core might deadlock or data corruption might occur under certain micro-architectural conditions

Status

Fault Type: Programmer Category A

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain micro-architectural conditions the *Processing Element* (PE) might deadlock or cause data corruption while executing an instruction that uses PSTATE.{N,Z,C,V} conditional flags.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs if the following conditions apply:

- The PE executes an instruction that updates the PSTATE.{N,Z,C,V} conditional flags.
- The PE later executes another instruction which consumes PSTATE.{N,Z,C,V} conditional flags.

Implications

If the previous conditions are met, under certain micro-architectural conditions the PE might deadlock, or data corruption might occur.

Workaround

There is no workaround for this erratum.

2937969

Data corruption can occur under certain micro-architectural conditions when the PE executes an SVE store instruction

Status

Fault Type: Programmer Category A

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain micro-architectural conditions, the *Processing Element* (PE) might corrupt data while executing an SVE store instruction.

Configurations affected

This erratum affects all configurations.

Conditions

The PE executes SVE instructions.

The PE later executes an SVE store instruction.

Implications

If the above conditions are met, under certain micro-architectural conditions, the PE might corrupt data.

Workaround

There is no workaround.

Category A (rare)

There are no errata in this category.

Category B

2921199

Execution of STG instructions in close proximity might cause loss of MTE allocation tag data

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain rare microarchitectural conditions, two or more STG instructions that access the same cache line but different 32-bytes might not write the MTE allocation tag to memory.

Configurations Affected

This erratum affects all configurations where the BROADCASTMTE pin is HIGH.

Conditions

This erratum occurs under the following conditions:

1. Memory tagging is enabled.
2. Two or more STG instructions are executed in close proximity to the same cache line but different 32-byte locations, interleaved with one or more STGs to a different cache line.

Implications

If the previous conditions are met, then under specific microarchitectural conditions, one of the STG instructions might not write the MTE allocation tag to memory.

Workaround

This erratum can be avoided by setting CPUACTLR5_EL1[14] to 1.

2922378

Branch prediction history not suppressed when switching from low to high EL

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Branch prediction history from an attacker in lower *Exception Level* (EL) is not properly suppressed when switching to a victim at higher EL. This causes the victim to unexpectedly speculate to a section of its own code that contains instructions that cause a side effect (such as a cache miss) which is later observable by the attacker.

Configurations affected

This erratum affects all configurations.

Conditions

When switching from lower EL to higher EL and the following CPUACTLR4 bits are configured as follows:

- Bit 11: BHB_SUPPRESS_AT_VEC_RESTART_DIS is set to 0.
- Bit 10: BHB_FLUSH_AT_VEC_RESTART_EN is set to 0.

Implications

An attacker running at lower EL might affect the behavior of a victim at higher EL, causing the victim to incorrectly (unexpectedly) speculate to one of its targets, which in turn can cause a side effect (such as a cache miss) observable by the attacker. A carefully crafted attack might result in confidential or sensitive information being leaked by the victim.

Workaround

The recommended hardware workaround is to disable BHB suppress, and to enable BHB flush. This can be done via CPUACTLR4 as follows:

- Set bit 11: BHB_SUPPRESS_AT_VEC_RESTART_DIS to 1.
- Set bit 10: BHB_FLUSH_AT_VEC_RESTART_EN to 1.

Using the above combination, the history register will be cleared on low to high EL transitions, precluding the attack, but with a negligible performance impact.

2933290

MTE Tag checking may not be performed if an access that crosses a 16-byte boundary encounters a poisoned Allocation tag

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain microarchitectural conditions, an MTE Tagged store that crosses a 16-byte boundary may not report a tag check fail if one of the Allocation tags is poisoned and a tag check fail is detected on refill of the cache line.

Configurations affected

This erratum affects all configurations where the BROADCASTMTE pin is HIGH.

Conditions

1. Memory tagging is enabled.
2. The PE executes a store which accesses an MTE Tagged page such that the access crosses a 16-byte boundary and reads two Allocation tags.
3. One of the Allocation tags accessed is poisoned. Tag check passes for the other granule.
4. The cache line is snooped out of the L1 and when the line is fetched again, the tag check for the granule that was initially poisoned fails, but the PE fails to report the error.

Implications

If the above conditions are met, then the PE may not report a tag check fail as required by the architecture.

Workaround

This erratum can be avoided by setting CPUACTLR5_EL1[42] to 1. Arm does not expect any performance implications from applying this workaround.

2963999

Incorrect virtualization of reads to MPIDR_EL1 and MIDR_EL1

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

In EL2/EL3, reads of MPIDR_EL1 and MIDR_EL1 might incorrectly virtualize which register to return when reading the value of MPIDR_EL1/VMPIDR_EL2 and MIDR_EL1/VPIDR_EL2, respectively.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs under the following conditions:

1. An exception entry to EL2 or EL3 occurs
2. No context synchronizing event (such as an ISB) has occurred since the last exception entry
3. An MRS instruction is executed to read either MPIDR_EL1 or MIDR_EL1

Implications

If the previous conditions are met, then the core might not correctly choose what it should return:

- when executing a read to MPIDR_EL1, it might return either MPIDR_EL1 (correctly) or VMPIDR_EL2 (incorrectly)
- when executing a read to MIDR_EL1, it might return either MIDR_EL1 (correctly) or VPIDR_EL2 (incorrectly)

Workaround

This erratum can be avoided by inserting an ISB prior to an MRS read to either MPIDR_EL1 and MIDR_EL1. Performance impact is expected to be negligible in real systems. This sequence can be implemented through execution of the following code at EL3 as soon as possible after boot:

```
// add ISB before MRS reads of MPIDR_EL1/MIDR_EL1
LDR x0,=0x0
MSR S3_6_c15_c8_0,x0 // MSR CPUPSELR_EL3, X0
LDR x0,=0xd5380000
MSR S3_6_c15_c8_2,x0 // MSR CPUPOR_EL3, X0
```

```
LDR x0,=0xFFFFFFFF40
MSR S3_6_c15_c8_3,x0 // MSR CPUPMR_EL3, X0
LDR x0,=0x000080010033f
MSR S3_6_c15_c8_1,x0 // MSR CPUPCR_EL3, X0
ISB
```

2982189

PE executing DRPS during Debug Halt under Double Fault condition will not execute properly

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

When a DRPS instruction is executed in Debug Halt state, a double fault should cause implicit ESB according to the *Arm Architecture Reference Manual for A-profile architecture* when (SCR_EL3.EA == '1' && SCR_EL3.NMEA == '1' && PSTATE.EL == **EL3**). However, the *Processing Element* (PE) will only execute part of the instruction for this case.

Configurations affected

This erratum affects all configurations with double fault extension.

Conditions

This erratum occurs under the following conditions:

1. The PE is in Debug Halt state.
2. Software is currently executing at EL3 Exception level.
3. SCTLR_EL3.IESB == '0'
4. SCR_EL3.EA == '1' && SCR_EL3.NMEA == '1' indicating double fault.

Implications

The DRPS instruction is not executed correctly.

Workaround

When executing a DRPS instruction in EL3, set SCTLR_EL3.IESB to override double fault. Doing this will force the correct DRPS execution sequence to occur.

3007699

SPE might write to pages which lack write permission at Stage-1 or Stage-2

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r0p1.

Description

The *Statistical Profiling Extension* (SPE) uses the Stage-1 translation regime of the owning exception level in the owning Security state. Due to this erratum, the SPE might write to memory which lacks write permission at Stage-1 and/or Stage-2 of the owning exception level's translation regime, without raising a fault.

Configurations affected

This erratum affects all configurations that support SPE.

Conditions

This erratum occurs under the following conditions:

1. The SPE buffer is enabled.
2. Registers PMBPTR_EL1 and PMBLIMITR_EL1 are configured to include a virtual address VA_X.
3. A valid Stage-1 translation exists for the virtual address VA_X.
4. If Stage-2 is enabled, a valid Stage-2 translation exists for the intermediate physical address IPA_X for the virtual address VA_X.
5. At least one of the following conditions is true:
 - a. The Stage-1 translation for VA_X lacks write permission.
 - b. The Stage-2 translation for IPA_X lacks write permission.
6. None of the following apply:
 - a. Stage-1 hardware dirty bit management is enabled.
 - b. Stage-2 is enabled, and Stage-2 hardware dirty bit management is enabled.

Implications

The SPE might write to VA_X rather than generating a fault. This might allow malicious software with control over SPE to corrupt memory for which it is not intended to have write access to.

Workaround

No hardware workaround is available.

A hypervisor at EL2 should not give virtual machines control of SPE unless the hypervisor can handle writes to any pages mapped at Stage-2.

An OS kernel at EL1 or EL2 should not configure the SPE buffer to contain any page which might lack write permission at Stage-1.

No current software is expected to have this problem.

3043240

Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0 and r0p1. Open.

Description

Under certain conditions, changing block size without break-before-make or mis-programming the contiguous bit can lead to an interruptible livelock in violation of FEAT_BBM level 2 requirements until TLB maintenance is performed.

Configurations affected

This erratum affects all configurations.

Conditions

1. The contiguous bit is mis-programmed for a set of contiguous Stage-1 or Stage-2 translation table entries.
2. A load or store crosses a page boundary within a contiguous address range such that an access for one page is translated by a translation table entry with the contiguous bit set and an access for another page is translated via a translation table entry with the contiguous bit clear.

or

1. A Stage-1 or Stage-2 translation table entry is modified without break-before-make such that a VA or IPA which was previously translated by a Page or Block entry is subsequently translated via a larger Block entry.
2. No TLB maintenance is performed to remove TLB entries for the stale Page or Block entry.
3. A load or store crosses a page boundary such that accesses for either page could be translated via the new block entry, and at least one access could have been translated by a distinct Page or Block entry prior to modification.

Implications

When the previous conditions are met, the load or store instruction will stall indefinitely without raising a fault. During the stall, the load or stall can be interrupted.

Workaround

Where software which manages the translation tables cannot ensure that it is not subject to the stall conditions, or where stalling is unacceptable, software which manages the translation tables should ignore **ID_AA64MMFR2_EL1.BBM** and always follow a break-before-make approach.

Where software which manages the translation tables can ensure that it is not subject to the stall conditions, and it is acceptable to transiently stall lower privileged software, software which manages the translation tables should minimize the period for which the contiguous bit is mis-programmed and minimize the period between modifying a translation table entry and invalidating TLB entries for the previous translation table entry.

3324334

MSR PSTATE.SSBS to 0 is not fully self-synchronizing

Status

Fault type: Programmer Category B
Fault status: Present in r0p0 and r0p1. Open.

Description

When PSTATE.SSBS is written to 0, the Arm Architecture specifies that side-effects are guaranteed to be visible to later instructions in the Execution stream. However, for a window of time during speculative execution of **MSR PSTATE.SSBS**, speculative store data bypassing might still occur.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if the following condition applies:

MSR PSTATE.SSBS executes, setting PSTATE.SSBS to 0.

Implications

Security sensitive code executed shortly after **MSR PSTATE.SSBS** to 0 might not be fully protected by the *Speculative Store Bypass Safe* (SSBS) feature.

Workaround

Software at EL3, EL2, and EL1 should follow writes to the SSBS register with a *Speculation Barrier* (SB) instruction to ensure that the new value of PSTATE.SSBS affects subsequent instructions in the execution stream under speculation.

A kernel at EL1 or EL2 should not advertise the presence of MRS/MSR instructions to read/write the SSBS register from ELO. Arm expects that kernels provide system calls for ELO software to modify PSTATE.SSBS when the SSBS register is not implemented and that ELO software will use this when the presence of the SSBS register is not advertised.

3701747

Read of ICH_VMCR_EL2.VBPR1 might return incorrect data based on SCR_EL3.NS

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

When ICH_VMCR_EL2.VBPR1 is written in Secure state (SCR_EL3.NS==0) and then subsequently read in Non-secure state (SCR_EL3.NS==1), a wrong value might be returned. The same issue exists in the opposite way: write in Non-secure state and read in Secure state. ICH_VMCR_EL2.VBPR1 is an alias of ICV_BPR1_EL1 which is architecturally defined as NOT banked. The RTL erroneously has this register implemented as two separate registers (secure and non-secure copies) banked by SCR_EL3.NS.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs if all the following conditions apply:

1. The *Processing Element* (PE) is executing at EL3
2. SCR_EL3.NS == 1 or 0
3. The PE executes an MSR ICH_VMCR_EL2.VBPR1 instruction
4. SCR_EL3.NS == 0 or 1 (the opposite value from when the MSR occurred)
5. The PE executes an MRS <dst>, ICH_VMCR_EL2.VBPR1 instruction

Implications

If the previous conditions are met, the MRS <dst>, ICH_VMCR_EL2.VBPR1 instruction will erroneously return the value that was last written to this field with the opposite SCR_EL3.NS value from which it was read (or the reset value if it was never written in that security state).

Workaround

The workaround is for EL3 software that performs context save/restore on a change of Security state to use a value of SCR_EL3.NS when accessing ICH_VMCR_EL2 that reflects the Security state that owns the data being saved or restored. For example, EL3 software should set SCR_EL3.NS to 1 when saving or restoring the value ICH_VMCR_EL2 for Non-secure (or Realm) state. EL3 software should clear SCR_EL3.NS to 0 when saving or restoring the value ICH_VMCR_EL2 for Secure state.

Category B (rare)

2917970

PE might incorrectly detect a Watchpoint debug event instead of a Data Abort exception on a page crossing memory access, resulting in errant entry to Debug state or routing the Data Abort exception to an incorrect Exception level

Status

Fault Type: Programmer Category B (Rare)

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain conditions, the *Processing Element* (PE) might incorrectly detect a Watchpoint debug event instead of a Data Abort exception when a memory access spans multiple pages. The Data Abort is detected for the first page and the Watchpoint debug event is associated with the second page. The Watchpoint debug event detection might route the Data Abort to the incorrect target Exception level or cause the PE to enter Debug state.

Note the contents of the ESR and FAR registers capture the information associated with the Data Abort.

Configurations affected

This erratum affects all configurations.

Conditions

1. Watchpoints are enabled.
2. The PE executes a page split access that generates a Data Abort on the first page and a Watchpoint match on the second page.
3. The PE executes a younger load instruction that generates an external abort which coincides with a 1 cycle window when processing the Data Abort and Watchpoint debug event.

Implications

If the previous conditions are met and EDSCR.HDE is set (enables Halting Debug on Watchpoint debug event), then the PE will enter Debug state rather than taking a Data Abort exception.

If EDSCR.HDE is not set, the PE might route the abort to the incorrect Exception level:

- If MDCR_EL2.TDE == 0, a stage 2 Data Abort might result in a Data Abort exception taken erroneously to EL1.

- The rarity of PE internal timings required to exhibit this bug is comparable to *Reliability, Availability, and Serviceability* (RAS) error FIT rates. Expected outcome is a kernel panic that will kill the process.
- If `MDCR_EL2.TDE == 1`, a stage 1 Data Abort might result in a Data Abort exception taken erroneously to EL2.
 - This scenario is containable within a hypervisor via the software workaround outlined below.

Workaround

There is no complete workaround for this erratum. A partial software workaround addresses the more serious scenario of a stage 1 Data Abort resulting in a Data Abort exception taken erroneously to EL2 without updating `HPFAR_EL2`.

EL2 can protect against this case as follows:

- Reserve one bit of IPA space so that `VTCTR_EL2.PS` is never the maximum supported.
- Write all 1's to `HPFAR_EL2[63:0]` before entering EL1 or EL0.
- Exceptions to EL2 due to this erratum that should have set `HPFAR_EL2` will instead use an out of range IPA. The guest should be restarted as the conditions for this erratum are rare and are not likely to be encountered again.

Category C

2871705

Noncompliance with prioritization of Exception Catch debug events

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

ARMv8.2 architecture requires that Debug state entry due to an Exception Catch debug event (generated on exception entry) occur before any asynchronous exception is taken at the first instruction in the exception handler. An asynchronous exception might be taken as a higher priority exception than Exception Catch and the Exception Catch might be missed altogether.

Configurations Affected

This erratum affects all configurations.

Conditions

1. Debug Halting is allowed.
2. EDECCR bits are configured to catch exception entry to ELx.
3. A first exception is taken resulting in entry to ELx.
4. A second, asynchronous exception becomes visible at the same time as exception entry to ELx.
5. The second, asynchronous exception targets an Exception level ELy that is higher than ELx.

Implications

If the above conditions are met, the core might recognize the second exception and not enter Debug state as a result of Exception Catch on the first exception. When the handler for the second exception completes, software might return to execute the first exception handler, and assuming the core does not halt for any other reason, the first exception handler will be executed and entry to Debug state via Exception Catch will not occur.

Workaround

When setting Exception Catch on exceptions taken to an Exception level ELx, the debugger should do either or both of the following:

1. Ensure that Exception Catch is also set for exceptions taken to all higher Exception Levels, so that the second (asynchronous) exception generates an Exception Catch debug event.
2. Set Exception Catch for an Exception Return to ELx, so that when the second (asynchronous) exception handler completes, the exception return to ELx generates an Exception Catch debug event.

Additionally, when a debugger detects that the core has halted on an Exception Catch to an Exception level ELy, where $y > x$, it should check the ELR_ELy and SPSR_ELy values to determine whether the exception was taken on an ELx exception vector address, meaning an Exception Catch on entry to ELx has been missed.

2871706

MPAM value associated with instruction fetch might be incorrect

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

Under some scenarios, the MPAM value associated with an instruction fetch request might be incorrect when context changes.

Configurations Affected

This erratum affects all configurations.

Conditions

1. An Instruction fetch request is attempted before a context switch but is not completed until after a context switch.

Implications

The MPAM value associated with the instruction fetch request might be incorrect.

Workaround

There is no workaround.

2874250

Accessing a memory location using mismatched Shareability attributes when MTE tag checking is enabled might cause data corruption

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

A PE accessing a same physical memory location with mismatched Shareability attributes and requiring a read of *Memory Tagging Extension* (MTE) tags might result in data corruption.

Configurations affected

This erratum affects all configurations.

Conditions:

This erratum occurs under the following conditions:

1. PE accesses a physical memory location using cacheable and Non-shareable attributes.
2. PE accesses the same physical address using cacheable and shareable attributes with MTE checking enabled.

Implications

If the previous conditions are met, the PE might expose stale data from the PE caches established by a Non-shareable access. This data might become visible to shareable observers in the same Shareability domain, even if the PE performs the required cache maintenance for ensuring ordering and coherency when aliasing Shareability.

Workaround

Arm expects that operating systems do not use mismatched Shareability attributes for aliases of the same memory location for tagged pages.

2901777

PMU event MEM_ACCESS_CHECKED_WR incorrectly counts aborted or inactive stores in MTE precise mode

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

The MEM_ACCESS_CHECKED_WR PMU events increment incorrectly when accessing a tagged page, although the write is aborted.

Configurations affected

This erratum affects configurations with BROADCASTMTE = 1.

Conditions

This erratum occurs if the following conditions apply:

1. A store accesses an MTE tagged page in MTE precise mode.
2. The write is either aborted or inactive due to SVE predication.

Implications

If the previous conditions are met, the PMU event might increment inaccurately.

Workaround

This erratum has no workaround.

2910965

L2D_CACHE_WB_CLEAN overcounts

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

Counting of the L2D_CACHE_WB_CLEAN event includes transfer of data directly to another PE using the AMBA CHI Direct Cache Transfer mechanism.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs under the following conditions:

1. The PE processes a forwarding snoop from the DSU or HN-F and sends data directly to another PE using a CompData message.

Implications

If the previous condition is met, the PE will count the L2D_CACHE_WB_CLEAN event contrary to the architectural specification of this event.

Workaround

No workaround is required for this erratum.

2911068

SPE latency counters are corrupted under certain conditions

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain conditions, the dispatch to issue and dispatch to completion latency counters for certain Statistical Profiling samples might be corrupted.

Configurations affected

This erratum affects all configurations.

Conditions

1. Statistical profiling is enabled at the appropriate Exception level.
2. The first instruction sampled is one of the following instructions:
 - FADDA
 - BFMMLA
 - FDIV
 - FSQRT
3. The sample gets flushed under certain micro-architectural conditions.
4. The next sample of one of the above instructions might capture incorrect latency values.

Implications

If the above conditions are met, the dispatch to issue and dispatch to completion counts for certain samples of FADDA, BFMMLA, FDIV, or FSQRT in the *Statistical Profiling Extension* (SPE) buffer might be corrupted.

Workaround

There is no workaround.

2925547

Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Reads of PMSIDR_EL1.CountSize incorrectly report 0x2 (12-bit) when 0x3 (16-bit) should be reported. The *Processing Element* (PE) implements 16-bit saturating counters, thus 0x3 should be reported.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs under the following conditions:

1. Any read of PMSIDR_EL1.CountSize

Implications

Software dependent on PMSIDR_EL1.CountSize will receive the incorrect counter width.

Workaround

If software needs to know the counter size, it should treat it as 16-bits and ignore the PMSIDR_EL1.CountSize value.

2927450

PE might report an unexpected SEA or SError on a read access by a load instruction

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain micro-architectural conditions, a load executing on a *Processing Element* (PE) might incorrectly consume data poison or DErr/NDErr that was meant for an instruction fetch or descriptor fetch for an unrelated translation table walk.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs if the following conditions apply:

1. The PE executes a load instruction.
2. An instruction fetch or descriptor fetch for an unrelated translation table walk returns poisoned data or generates a DErr/NDErr.

Implications

If the previous conditions are met, then the PE might incorrectly signal SEA or SError on the load instruction, but the data returned by the load will be correct.

Workaround

There is no workaround for this erratum.

3061575

TagMatch responses with error indication do not generate a SError abort

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

When tag checks are performed outside of the *Processing Element* (PE), the AMBA CHI protocol returns a TagMatch response that indicates whether or not the tag check succeeded or failed. If an error condition occurred while performing the tag check, the system might return the TagMatch response with an error indication. If this occurs, the PE should report a SError abort, but fails to do so.

Configurations affected

This erratum affects all configurations with the BROADCASTMTE pin asserted

Conditions

This erratum occurs under the following conditions:

1. PE has *Memory Tagging Extension* (MTE) enabled in asynchronous checking of stores
2. PE performs tag checked stores
3. Write streaming causes the PE to send the stores to the interconnect as write transactions
4. While performing the tag check operation for the write, the interconnect encounters an error condition while reading the tag value

Implications

If the conditions are met, the interconnect might return a TagMatch response with an error indication, but the PE might not generate a SError abort. If the TagMatch response indicates a tag check failure (Resp=Fail), TFSR_ELx bits will still be updated.

Workaround

No workaround is required for this erratum.

3384220

PMU event L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW count incorrectly

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

Counting of the L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW PMU events are incorrect.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs whenever the L3D_CACHE_ALLOCATE, L3D_CACHE, or L3D_CACHE_RW PMU events (numbers 0x29, 0x2b, and 0x8150 respectively) are selected for counting.

Implications

The PMU events L3D_CACHE_ALLOCATE, L3D_CACHE, and L3D_CACHE_RW count unpredictably and cannot be used for useful analysis.

Workaround

This erratum has no workaround.

3563818

Incorrect decoding of SVE version of PRFH scalar plus vector (gather) instructions

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

The 32-bit unpacked scaled Scalar plus Vector offset form of gather PRFH instruction might not prefetch from the correct address.

Configurations affected

This erratum affects all configurations.

Conditions

A 32-bit unpacked scaled Scalar plus Vector offset form of gather PRFH instruction is executed from any EL, when *Scalable Vector Extension* (SVE) is enabled and not trapped.

Implications

The affected instruction is a software prefetch which does not affect architectural state in any way (including suppression of any translation faults). Thus, this erratum will not affect the functional operation of the CPU.

Software that relies on explicit prefetches to increase performance might not realize the benefit of the requested prefetches and might instead experience additional cache pollution when executing this instruction targeting incorrect addresses.

Workaround

No workaround is expected to be necessary for this erratum.

3604846

PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

When software directly writes PSTATE.PAN or PSTATE.UAO with an MSR instruction, the Arm Architecture specifies that side-effects are guaranteed to be visible to later instructions in the Execution stream. However, for a window of time prior to the execution of MSR PSTATE.{PAN,UAO}, instructions following the MSR might speculatively execute with the old context, prior to re-executing non-speculatively under the new, expected context.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if the following condition applies:

- MSR PSTATE.{PAN or UAO} executes

Implications

Speculative execution of instructions using stale PSTATE.{UAO,PAN} context could in theory present a window of opportunity for a security attack. However, Arm security team has evaluated the practical risk to be very low, given the use-cases of the bits in question and the complexity involved in exploiting.

Workaround

A workaround is not expected to be required.

3605029

Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

A hardware generated prefetch operation or a PRFM instruction might indicate a L1D_TLB_REFILL_RD event leading to an incorrect count.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if all the following conditions apply:

1. PMU counters are configured to count event 0x004C.
2. A hardware generated prefetch or PRFM instruction might encounter a L1D TLB miss, resulting in a refill operation and triggering event 0x004C.

Implications

If the previous conditions are met, the count indicated by event 0x004C will not reflect the conditions specified in the Arm Architecture Reference Manual. Furthermore, this event is used in calculating the "Attributable Level 1 TLB refill rate, read" metric which by extension will not reflect an accurate rate.

Workaround

No workaround is required unless PMU event 0x004C is required. If a workaround is needed, this erratum can be avoided by counting three separate PMU events in place of event 0x004C:

- Event 0x0005 (L1D_TLB_REFILL)
- Event 0x004D (L1D_TLB_REFILL_WR)
- Event 0x10E. (L1D_TLB_REFILL_RD_PF)

These events can be used to calculate an Effective event 0x004C as follows:

Effective Event 0x004C = Event 0x0005 - Event 0x004D - Event 0x010E

Effective event 0x004C can be used in place of event 0x004C in calculation of "Attributable Level 1 TLB refill rate, read" to provide an accurate rate calculation.

Arm Architecture Reference Manual relevant events:

Mnemonic	Number
L1D_TLB_REFILL	0x0005
L1D_TLB_REFILL_RD	0x004C
L1D_TLB_REFILL_WR	0x004D
L1D_TLB_RD	0x004E

Implementation Defined relevant event:

Mnemonic	Number
L1D_TLB_REFILL_RD_PF	0x010E

Arm Architecture Reference Manual relevant metric:

"Attributable Level 1 TLB refill rate, read" (Event 0x004C / Event 0x004E)

3627351

PMU event STALL_SLOT_FRONTEND counts when instruction fetch is stalled for PCRF availability

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

When instructions are not available to be dispatched due to Program Counter Register File (PCRF) fullness, they are counted by the STALL_SLOT_FRONTEND PMU event instead of the STALL_SLOT_BACKEND PMU event.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs whenever instruction fetch is stalled due to PCRF fullness and the PMU is configured to count the STALL_SLOT_FRONTEND or STALL_SLOT_BACKEND events.

Implications

Correlation of STALL_FRONTEND and STALL_SLOT_FRONTEND telemetry might be impacted when the PCRF is often full, because the STALL_FRONTEND PMU event will not count under the same PCRF full conditions.

Workaround

This erratum has no workaround.

3633453

EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

When a Load-Exclusive instruction is executed with Halting Step enabled, EDSCR.STATUS is not updated if the Load-Exclusive instruction causes a synchronous exception.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs under the following conditions:

1. In Debug state, the debugger enables Halting Step
2. Debug state is exited and a Load-Exclusive instruction (LDX*/LDAX*) is stepped
3. The Load-Exclusive generates a synchronous exception while executing

Implications

If the conditions are met, EDSCR.STATUS will not be updated.

Workaround

There is no workaround.

3640926

SPE operation type is corrupted under certain conditions

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

The FP field (Floating Point) of the operation type header in a *Statistical Profiling Extension* (SPE) record, might not be set correctly for certain *Scalable Vector Extension* (SVE) samples. The affected opcodes are FDIV, FDIVR and FSQRT.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs under the following conditions:

1. SPE sampling is enabled.
2. SPE samples one of the following instructions:
 - FDIV
 - FDIVR
 - FSQRT

Implications

If the previous conditions are met, then the FP bit information in the SPE buffer might be inaccurate for the previous mentioned samples.

Workaround

There is no workaround.

3694318

LS misses RAR hazard on case with clean critical beat and poisoned final response with ECC disabled

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

When PE is configured with ERROCTL.R.ED = 0, a load instruction that received data on the CPU AMBA CHI interface with some words marked Poisoned can violate internal visibility requirement.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if all the following conditions apply:

1. PE is configured with ERROCTL.R.ED = 0, disabling Error detection and correction
2. Data requested by a load instruction is received on the CPU AMBA CHI interface with some words marked Poisoned, indicating an uncorrected error has been detected in the system
3. Load consumes non-poisoned words from the returned data.
4. Another PE performs a write to one or more of the bytes consumed by the load

Implications

When the above conditions are met, load instruction might read stale data violating memory ordering requirements.

Workaround

No workaround is expected to be necessary for this erratum.

3694450

FFR might not capture the lowest faulting memory element

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

Under certain unusual micro-architectural conditions, the *Processing Element* (PE) executing a *Scalable Vector Extension* (SVE) First-fault or Non-fault vector load instruction that fails *Memory Tagging Extension* (MTE) tag check or reads poisoned data might not capture the correct faulting element in the *First Fault Register* (FFR).

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if all of the following conditions apply:

1. PE executes an SVE First-fault load instruction with first active element to device memory.
2. PE executes a younger SVE First-fault or Non-fault vector load instruction to normal memory where active element of the Non-fault vector load instruction or non-first active element of the First-fault vector load instruction fails MTE tag check or reads poisoned data.
3. Unusual micro-architectural conditions occur.

Implications

When the above conditions are met, FFR lane corresponding to the lowest faulting memory element might not be set to False.

Workaround

Arm does not expect this issue to occur in realistic code sequences, so no workaround is needed. Please contact Arm for more details.

3699919

PE might fail to log a RAS error for L2 data RAM ECC errors

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

Under specific circumstances, the L2 cache might fail to log a corrected or uncorrected ECC error in the PE ERXSTATUS/MISC/ADDR registers.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if all the following conditions apply:

1. Error correction is enabled with ERROCTL.ED set to 1.
2. PE is performing simultaneous memory reads to both Device or Normal Non-cacheable and Normal-WriteBack memory.
3. Specific timing conditions occur.
4. PE detects an ECC error in the L2 data RAM.

Implications

If the specified conditions occur, the PE might not report the ECC error detected by the L2.

Note that there is no silent data corruption - any consumers of the data will receive a poison indication along with the data. The issue is a failure to report the error to the RAS error log.

Workaround

No workaround is necessary for this erratum.

3705902

PMU events are mis-categorized by not considering the effect of "Taken locally"

Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r0p1. Open.

Description

FEAT_VHE establishes broad use of "Taken locally" as a qualifier that determines which instances of an exception are counted by particular PMU events.

PMU events are mis-categorized by failing to consider "Taken locally", specifically resulting in mis-categorizations between PMU events EXC_UNDEF and EXC_TRAP_OTHER, as well as between PMU events EXC_SVC and EXC_TRAP_OTHER.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum can occur if one of the following conditions apply:

- When the effective value of HCR_EL2.{E2H,TGE} **is** {1,1}, an exception can increment PMU event 0x008D EXC_TRAP_OTHER, when the exception should instead increment PMU event 0x0081 EXC_UNDEF.
- When the effective value of HCR_EL2.{E2H,TGE} is **NOT** {1,1}, an exception can increment PMU event 0x0081 EXC_UNDEF, when the exception should instead increment PMU event 0x008D EXC_TRAP_OTHER.
- When the effective value of HCR_EL2.{E2H,TGE} is **NOT** {1,1}, executing an SVC instruction can increment PMU event 0x0082 EXC_SVC, when that SVC instruction should instead increment PMU event 0x008D EXC_TRAP_OTHER.

Implications

When the previous conditions are met, PMU event counts might be inaccurate for events 0x0081, 0x0082, and 0x008D.

Workaround

There is no workaround.

3707125

Parity Error in L2BTB can cause failure in breakpoint detection

Status

Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

A parity error in L2BTB lookup that has multiple entries in block can cause a breakpoint detection fail for that specific lookup.

Configurations affected

This erratum affects all configurations.

Conditions

The erratum occurs if the following sequence of conditions apply:

1. A breakpoint is set to specific instruction A
2. Instruction A is fetched
3. Instruction A lookup contains parity error in L2BTB

Implications

The breakpoint will be missed, and program execution will errantly continue beyond that breakpoint.

Workaround

There is no workaround.

Proprietary notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(PRE-1121-V1.0)

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

The rxpy identifier indicates the revision status of the product described in this manual, where:

rx

Identifies the major revision of the product.

py

Identifies the minor revision or modification status of the product.