Official Release

AndesCore™ AX45MP-1C Errata

Document Number ER080-10
Date Issued 2022-03-25

Copyright Notice

Contact Information

Should you have any problems with the information contained herein, you may contact Andes Technology Corporation through at support@andestech.com

Please include the following information in your inquiries:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of the problem.

General suggestions for improvements are welcome.

Revision History

Rev.	Rev. Date	Revised Content
1.0	2022-03-25	Initial release

Contents

1 Product Revisions

Product	CPU Revision	Release Date
AX45MP	5.0.0	2021-06

2 Severity

The severity of issues can be classified into three categories as follows:

- Level-1: Function issues which cannot be worked around
- Level-2: Function issues which can be worked around
- Level-3: Function limitations or other issues

3 Issue List

3.1 AX45MP 5.0.0

3.1.1 Level-1 Issues

None

3.1.2 Level-2 Issues

ID	Affected Features	Issue Summary
24650	BTB and WFI	A WFI instruction might just behave as NOP, if the location of the WFI instruction was previously a branch instruction and BTB remembers this location as a taken branch.
24642	I-Cache Parity and EXEC.IT	I-Cache parity errors related to execution of EXEC.IT instructions may not be auto-repaired.
24576	StackSafe	Incorrect instruction execution for loading data to the `sp` register when StackSafe is configured.
24569	StackSafe	MUL/DIV/REM instructions update wrong values to the `sp` register when StackSafe is configured.
24560	Writes to Instruction Memory	An incorrect instruction is executed when a branch instruction is replaced by certain instructions after context-switch/self-modifying code.
24490	Cache Coherency	Data sharing races may cause a core to hang or drop data to the shared line.
24156	BTB and EXEC.IT	The processor might fetch wrong instruction data when the instruction memory is modified.
24048	L2-Cache Write-No-Allocate	The processor might hang when a write-no-allocate request follows a L2-Cache register write operation.
23814	JAL Instruction Dependency	A jump-and-link instruction to a bitwise logic-op/branch instruction with read-after-write dependency on the link register might produce an incorrect result.
23754	Virtual Memory	The processor might send speculative/redundant instruction fetch requests with untranslated virtual addresses to the system bus.
23501	CoDense	An exec.it instruction mapping to a JAL instruction might jump to a wrong target.

3.1.3 Level-3 Issues

ID	Affected Features	Issue Summary
24647	D-Cache ECC	ECC Error on D-Cache tag SRAM is not detected when handling probe requests.
24051	Programmable Physical Memory Attribute (PMA)	Static PMA setting for instruction fetch cannot be overridden by programmable PMA entries.
23376	Sv48	Page fault exceptions might not be triggered when a Sv48 virtual memory system is in the Sv39 mode and load effective addresses do not follow the Sv39-mode sign-extension rule.
23353	Store Bus Error	When a store bus error occurs and D-Cache ECC is configured, the next partial store may update D-Cache with corrupted value.
23316	StackSafe	When a stack overflow/underflow exception is taken, the next load/store instruction is not killed.
23199	Misaligned AMO	Store/AMO access fault exceptions are not raised for double-word misaligned AMO instructions.

4 Detailed Issue Descriptions

4.1 23199

Severity: Level-3
Affected Feature: Misaligned AMO
Affected Revisions: 5.0.0

Description:
Store/AMO access fault exceptions are not raised for double-word misaligned AMO instructions.

Workaround:
None.

Recommended Action:
Upgrade to AX45MP 8.0.0 or later revisions.

4.2 23316

Severity: Level-3
Affected Feature: StackSafe
Affected Revisions: 5.0.0

Description:
When a stack overflow/underflow exception is taken, the next load/store instruction is not killed.

Workaround:
None.

Recommended Action:
* Upgrade to 8.0.0 or later revisions.

4.3 23353

Severity: Level-3
Affected Feature: Store Bus Error
Affected Revisions: 5.0.0

Description:
When a store bus error occurs and D-Cache ECC is configured, the next partial store may update D-Cache with corrupted value.

Workaround:
None.

Recommended Action:
* Upgrade to 8.0.0 or later revisions.

4.4 23376

Severity: Level-3
Affected Feature: Sv48
Affected Revisions: 5.0.0

Description:
Page fault exceptions might not be triggered when a Sv48 virtual memory system is in the Sv39 mode and load effective addresses do not follow the Sv39-mode sign-extension rule. (Values of bit 63-39 should all equal to that of bit 38.)

Workaround:
None.

Recommended Action:
Upgrade to AX45MP 8.0.0 or later revisions.

4.5 23501

Severity: Level-2
Affected Feature: CoDense
Affected Revisions: 5.0.0

Description:
An exec.it instruction mapping to a JAL instruction might jump to a wrong target. This issue is triggered when all the following conditions are true:

- The pc of the exec.it instruction is larger than or equal to 2MiB.
- The previous instruction before EXEC.IT is stalled by 4 or more cycles.
- The exec.it instruction is dual-issued with the previous instruction.

Workaround:

- Limit the exec.it instruction that maps to a JAL instruction within 2MiB. Please contact Andes for the tool-chain usages.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.0.0 or later revisions.

4.6 23754

Severity: Level-2
Affected Feature: Virtual Memory
Affected Revisions: 5.0.0

Description:
The processor might send speculative/redundant instruction fetch requests with untranslated virtual addresses to the system bus. Instruction fetch unit (IFU) uses a return address stack (RAS) to predict the target address of return instructions. When a return instruction is predicted in M-Mode, the top of the RAS is popped as the address to fetch the next instruction. When the top of the RAS is a virtual address pushed in S-Mode or U-Mode, IFU might send a fetch request using the address without translation in M-Mode. The IFU redirects the return instruction to the correct address in the latter pipeline stages, but the redundant request might cause unexpected side effects.

Workaround:

- Use PMP mechanism to clear the execute permission of memory space that has side effects on reads; or
- Insert the code below in the M-Mode trap handler before the first function call to overwrite all RAS entries with known physical address(clear_ras_done).

```
# workaround for issue 23754 begin
clear_ras:
                li t0, 8
clear_ras_loop:
                beqz t0, clear_ras_done
                addi t0, t0, -1
                jal ra, clear_ras_loop
clear_ras_done:
# workaround for issue 23754 end
```

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.0.0 or later revisions.

4.7 23814

Severity: Level-2
Affected Feature: JAL Instruction Dependency
Affected Revisions: 5.0.0

Description:
A jump-and-link instruction to a bitwise logic-op/branch instruction with read-after-write dependency on the link register might produce an incorrect result.

This issue requires the following code sequence to trigger:

- A jump-and-link instruction (JAL, JALR, C.JAL, C.JALR)
- The target of the jump-and-link instruction is a bitwise logic-op/branch instruction, with read-after-write dependency to the link register of the jump-and-link instruction.
- The list of affected instructions are:
  - AND, OR, XOR, ANDI, ORI, XORI, C.AND, C.OR, C.XOR, C.MV BNE, BEQ, C.BEQZ, C.BNEZ, BEQC, BNEC

Workaround:

- This sequence is not a typical sequence that any compiler would generate.
- Insert an NOP instruction as the destination of the jump-and-link instruction for hand-crafted assembly codes.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.1.0 or later revisions.

4.8 24048

Severity: Level-2
Affected Feature: L2-Cache Write-No-Allocate
Affected Revisions: 5.0.0

Description:
The processor might hang when a write-no-allocate request follows a L2-Cache register write operation. The request sequence is as follows:

1. A first request whose response is not ready until the third request is processed,
2. A second request writing to any L2-Cache control register,
3. A third request that is a write-no-allocate request.

The response of the third request may be dropped and cause the processor to hang.

Workaround:
Avoid the write-no-allocate request. Perform all the following operations:

- Disable write-around by clearing the `mcache_ctl.DC_WAROUND` bit; and
- Avoid using PMA MTYPE 8/9; and
- Avoid using `L1D_VA_LOCK` cctl command to lock all ways of D-Cache; and
- Avoid sending requests with awcache=7 to IOCP.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.2.0 or later revisions.

4.9 24051

Severity: Level-3
Affected Feature: Programmable Physical Memory Attribute (PMA)
Affected Revisions: 5.0.0

Description:
Static PMA setting for instruction fetch cannot be overridden by programmable PMA entries. A region could be statically assigned as Device Region by assigning its PMA setting at configuration time of the processor hardware and the setting should be allowed to be overridden with settings in the programmable PMA entries. However, instead of honoring the settings in the Programmable PMA, the processor incorrectly treats such a region as a Device Region for instruction fetches. The implication of this issue is a performance one: such regions will not be cached by I-Cache.

Workaround:
None.

Recommended Action:
Avoid assigning regions as Device Regions in the configuration time for regions to be used as cacheable instruction memories.

4.10 24156

Severity: Level-2
Affected Feature: BTB and EXEC.IT
Affected Revisions: 5.0.0

Description:
The processor might fetch wrong instruction data when the instruction memory is modified.

This issue occurs when a branch instruction is replaced by a non-branch instruction. It is triggered when a BTB entry needs to be invalidated for the new instruction and the operation collides with the following events simultaneously:

- EXEC.IT execution; and
- Another branch redirection

Please note that the following scenarios are also counted as instruction modification for the purpose of this issue:

- The debugger loads a new program; or
- The operation system performs a context switch to a new process with the same instruction address space.

There are two instruction layout that could trigger this issue and they are described below, assuming that a branch at address A is replaced by a non-control-flow instruction:

- Scenario 1:
  - A branch or jump instruction is located between A+2 to A+12.
  - An EXEC.IT instruction is after the branch or jump instruction, and the address is in A+4 to A+48.
- Scenario 2:
  - A branch or jump instruction is located between A-2 to A-12.
  - An EXEC.IT instruction is located between A+2 to A+44.

Under these scenarios, the branch prediction information for the replaced branch has to be trained into BTB as a taken branch and corrected to a non-taken branch before it is being replaced to trigger this issue.

Workaround:

- Reset the processor before executing another program in the same memory space, or
- Avoid using EXEC.IT instructions:
  - This could be achieved by not using -Os, -Os2, and -Os3 compiler optimization flag.
  - Note that Linux environment does not use EXEC.IT instructions by default.
- Perform BTB flush sequence to flush all BTB entries after instruction memory modification. Please contact Andes for the sample code of the BTB flush sequence.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.2.0 or later revisions.

4.11 24490

Severity: Level-2
Affected Feature: Cache Coherency
Affected Revisions: 5.0.0

Description:
Data sharing races may cause a core to hang or drop data to the shared line.

This issue requires at least three store instructions writing to the same cache line, the first store instruction misses D-Cache, and the last store instruction writes to the same XLEN-bit naturally aligned region of any of the earlier stores. Furthermore, there should be one load instruction executed immediately before the last store instruction in back-to-back cycles. The load instruction encounters stall conditions and the last store instruction encounters a cancellation event. Because of the cache miss of the first store, it will take a while before the cache line data returns and valid entries in the store buffer are written back to D-Cache. This issue is triggered when this cache line is accessed by other processor cores/IOCP in this period.

The load instruction could be stalled when it is a device/noncacheable load or when non-blocking mode is off. And the last store instruction could be canceled by branch misprediction or interrupts. Furthermore, since the processor is a dual-issue machine, load and store instructions could be executed in back-to-back cycles when they are next to each other or separated by one extra instruction in the dynamic program sequence.

The complete event sequence in chronological order is described below:

- One of the processor core has a store miss to cache line A, with multiple subsequent stores to the same cache line before the cache miss returns
- Followed by a load instruction and a store instruction in back-to-back cycle, also before the write-miss to line A returns:
  - the load instruction needs to encounter stall conditions:
    - either because it is a device/non-cacheable load, or
    - or because it is a cacheable load but non-blocking load is not enabled.
  - the store hits one of the earlier stores to line A in the store buffer (e.g., hitting entry K of the store buffer)
  - the store gets killed (e.g., branch mis-prediction or interrupts)
- Another processor core (or IOCP) writes to line A and trigger a probe request to the above core and hit entry K before it is written back to D-Cache

Workaround:

- Turn on non-blocking mode; and
- For all accesses to device/non-cacheable loads, if there are subsequent stores to any shared data region, insert a FENCE or two NOP instructions between the load and the store instruction to ensure that these two instructions cannot be executed in back-to-back cycles.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.3.0 or later revisions.

4.12 24560

Severity: Level-2
Affected Feature: Writes to Instruction Memory
Affected Revisions: 5.0.0

Description:
An incorrect instruction is executed when a branch instruction is replaced by certain instructions after context-switch/self-modifying code. An instruction is fetched and executed incorrectly because the residual state for the branch instruction in BTB is not repaired correctly.

The list of affected instructions are:

- CSR write instructions, or
- MDU/LOAD instructions with rd being sp.

Any half-word of a branch instruction overwritten by the affected instructions would cause this issue.

Workaround:

- Reset the processor before executing another program in the same memory space, or
- Perform BTB flush sequence to flush all BTB entries after instruction memory modification. Please contact Andes for the sample code of the BTB flush sequence.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.3.0 or later revisions.

4.13 24569

Severity: Level-2
Affected Feature: StackSafe
Affected Revisions: 5.0.0

Description:
MUL/DIV/REM instructions update wrong values to the `sp` register when StackSafe is configured. The affected instructions perform division-by-zero or multiplication-by-zero operations with the following operand details:

- DIV/REM instruction:
  - the `sp` register is the destination register
  - the second operand (rs2) is the `zero` register (x0)
- MUL instruction, when the radix type is selected for the hardware multiplier:
  - the `sp` register is the destination register
  - the second operand (rs2) is the `zero` register (x0)

Workaround:

- These instructions are not typical instructions that any compiler would generate.
- For hand-crafted assembly codes, replace the affected instruction with two instructions
  - The first instruction uses a temp register as the destination register
  - The second instruction moves the temp register to the `sp` register.

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.3.0 or later revisions.

4.14 24576

Severity: Level-2
Affected Feature: StackSafe
Affected Revisions: 5.0.0

Description:
Incorrect instruction execution for loading data to the `sp` register when StackSafe is configured.

The affected instructions are C.LWSP and C.LDSP from the "C" extension and LBGP, LBUGP, LHGP, LHUGP, LWGP, LWUGP and LDGP from the Andes performance extension. In both cases, `sp` will be updated to a wrong data. Furthermore, for C.LWSP and C.LDSP, the program counter will be incorrectly updated by 4 instead of 2.

Workaround:

- These instructions are not typical instructions that any compiler would generate.
- For hand-crafted assembly codes, replace the affected instruction with two instructions:
  - The first instruction uses a temp register as the destination register
  - The second instruction moves the temp register to the `sp` register

Recommended Action:

- Apply the workaround; or
- Upgrade to 8.3.0 or later revisions.

4.15 24642

Severity: Level-2
Affected Feature: I-Cache Parity and EXEC.IT
Affected Revisions: 5.0.0

Description:
I-Cache parity errors related to execution of EXEC.IT instructions may not be auto-repaired. The issue occurs when all of the following conditions are true:

- `mcache_ctl.IC_ECCEN` is 0x3 (Generate exceptions on both repairable and unrepairable parity/ECC errors), and
- An EXEC.IT instruction is fetched, and
- The EXEC.IT table entry for this EXEC.IT instruction is in I-Cache and has TAG/Data parity errors.

The EXEC.IT table is a 4KiB data pointed to by the `uitb` CSR.

When `mcache_ctl.IC_ECCEN` is 0x3, the processor is required to repair the corresponding entry before taking exceptions. However, it enters the trap handler without correcting the parity errors under the above conditions.

Workaround:

- Execute a FENCE.I instruction in the trap handler

Recommended Action:

- Apply the workaround; or
- Upgrade to 12.0.0 or later revisions.

4.16 24647

Severity: Level-3
Affected Feature: D-Cache ECC
Affected Revisions: 5.0.0

Description:
ECC Error on D-Cache tag SRAM is not detected when handling probe requests.

Workaround:
None.

Recommended Action:

- Upgrade to 11.0.0 or later revisions.

4.17 24650

Severity: Level-2
Affected Feature: BTB and WFI
Affected Revisions: 5.0.0

Description:
A WFI instruction might just behave as NOP, if the location of the WFI instruction was previously a branch instruction and BTB remembers this location as a taken branch. This issue could occur when a branch is overwritten with a WFI instruction under context-switch/self-modifying code.

Workaround:

- The WFI instruction is defined by the RISC-V privilege specification to be a hint and NOP is a legal implementation. A proper usage should place WFI in an idle loop and no workaround should be required.

Recommended Action:

- Apply the workaround; or
- Upgrade to 12.0.0 or later revisions.