ERR010385

C55 E200Z4 BRANCH DISPLACEMENT ISSUE

AMP APPLICATIONS ENGINEERING

10 AUGUST 2016

REVISION 7

SECURE CONNECTIONS FOR A SMARTER WORLD

Impacted devices (Automotive Power Architecture c55 MCUs)

- Devices impacted by this issue:
  - MPC5744P
  - MPC577xK (e200z4 core only)
  - MPC5777M IOP core (e200z4 core only, e200z7 cores are not impacted)
  - S32R274 (e200z4 core only)

- The following devices are not impacted:
  - MPC574xB/MPC574xC/MPC574xG
  - MPC574xR
  - MPC5777C
  - MPC55xx – no products are impacted
  - MPC56xx – no c90 products are impacted

Errata wording

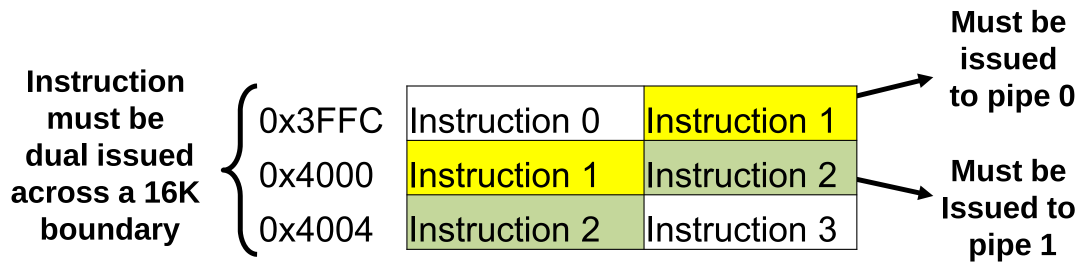
- Title:
  - e200z4: Incorrect branch displacement
- Description:
  - The branch target address will be incorrectly calculated in the e200z4 core, under the following conditions (all conditions must be matched):
    - The first full instruction in a 16 Kbyte section/page of code is a 32-bit long branch with a branch displacement value with the lower 14 bits of the actual displacement exactly 0x3FFE
    - And this branch instruction is located at byte offset 0x0002 in the section/page
    - And the preceding instruction is a 32-bit length instruction which is misaligned across the 16K boundary
    - And both instructions are dual-issued
  - Under these conditions, the branch target address will be too small by 32Kbytes.
- Workaround:
  - After software is compiled and linked, code should be checked to ensure that there are no branch instructions located at address 0x2 of any 16K memory boundary with the lower 14 bits of the displacement equal to 0x3FFE if preceded a 32-bit instruction that crosses the 16K memory boundary. If this sequence occurs, add a NOP instruction or otherwise force a change to the instruction addresses to remove the condition.
  - A tool is available on nxp.com that can be run to examine code for this condition, search for branch_displacement_erratum_10385_checker.

When will this rare case be seen

- A faulty branch displacement calculation will only happen if all of the following conditions all occur at the same time
  - Both instructions must be dual-issued
    - The first instruction must go into pipe 0 and the branch into pipe 1
    - The branch cannot depend on the first instruction results to be dual issued
    - In typical customer code, the majority of instructions are not dual issued
  - A 32-bit length instruction must be misaligned across a 16K boundary.
    - Instruction located at an address offset of 0x3FFE (0x7FFE, 0xBFFE, 0xFFFE).
    - In an 8M flash, there are 496 16K regions in executable space (assuming small blocks are not used for code, 512 if small blocks are also used for code)
  - The next instruction (at address 0x002 of a 16k region) must be a 32-bit branch instruction
  - The branch instruction branch displacement value must have the lower 14 bits of the displacement exactly equal to 0x3FFE

Errata explanation

- An incorrect displacement calculation will occur on the instruction pipe 1 under the following conditions:
  - The PC is at 2 bytes before a 16K bytes boundary (address offset of 0x3FFE and multiples)
  - The instructions at offset 0x3FFE (and multiples) have to be as follows
    - Both instructions must be dual issued at the same time
    - Instruction in pipe 0 must be 32 bits;
    - Instruction in pipe 1 must be a relative branch with the 14 least significant bit displacement value of 0x3FFE
- The calculation is current address (0xXXXX_{3,7,B,F}FFE) + the displacement + 4 (1 32-bit instruction)
  - The calculation could require 2 bits of overflow, but only 1-bit is implemented.



- Note: many instruction sequences, such as a branch that depends on the result of the preceding instruction, will not be dual issued.

Tools

- NXP has developed a tool (branch_displacement_erratum_10385_checker.exe) that can be used to search compiled code (ELF, SRECORD, and Intel Hex format files) to look for the conditions that may not be executed properly. The file format is automatically detected.
  - This utility identifies possible issues by displaying the assembly code listing with the address of the instruction and the branch, showing the actual displacement. The user must manually determine the source code that must be modified. The user should add instructions or rearrange functions to address the issue.
  - The utility does not check to see if the branch is dependent on the previous instruction. It also does not check to whether the branch can be dual issued with the previous instruction.
  - In ELF files, only sections marked executable are checked.
  - This utility now supports SRECORD and Intel Hex files. In SRECORD and Hex files, all areas are checked and there is no distinction between code and data segments. In data sections, it is possible that false matches could be identified if the data matches an affected code sequence. It is the user's responsibility to determine whether a positive detection of a issue can be safely ignored because it was triggered by data, not executable code.
    - An option to ignore false positives is available to manually ignore false matches. This allows for a PASSED response, as well as returning a zero from the utility.
  - Keep in mind that even if this utility identifies possible issues, the code may execute properly. The dual issue must occur with the misaligned 32-bit instruction (over a 16K memory region) issuing to pipe 0 and the branch with the 14-bit displacement of 0x3FFE must issue to pipe 1.
- This tool does not make any changes to the ELF, SRECORD, or Hex files, it opens the files as READ ONLY and identifies if the specific code sequence occurs in the file. It is the user’s responsibility to make code changes to their source files to avoid this issue.

Secondary tool

- NXP developed a second tool that is written in TCL and was developed independently from the first tool.
- This second tool only supports elf files
- The program will detect and report errata violations in the <elf_file> to the console. If any failures are detected, the scripts exits with a non-zero value to the system, and a distinctive "FAIL" string to the console. If no failures are detected, the script exits with 0 to the system and a distinctive "PASS" string to the console output.
- This version does not show the offending code, it only identifies where a 32-bit branch instruction occurs on a 16K+0x2 boundary and has a displacement with the lower 14-bits of 0x3FFE
- It does list 16K section (that exist in the file) that do not have a branch that matches the criteria for the erratum to exist.
- This tool does not make any changes to the ELF file, it opens the files as READ ONLY and identifies if the specific code sequence occurs in the file. It is the user’s responsibility to make code changes to their source files to avoid this issue.

Compilers

- NXP will work with compiler companies to modify the compiler and linker to automatically avoid the instruction sequence. However, this would require changing to newer versions of the tool.
  - NXP's S32 Design Studio for Power v1.1 has been updated to automatically not generated the impacted code sequence.
    - GNU Build Tools for e200 processors (support VLE and BookE ISA, based on gcc 4.9.2, binutils 2.24 and gdb 7.8.2)
  - NXP is still working with Green Hills Software (Multi Compiler), Wind River (Diab Compiler), and HighTec (Development Platform). Contact the compiler vendor for availability.

Branch_displacement_erratum_10385_checker Run example

```
c:\branchcheck>branch_displacement_erratum_10385_checker.exe test.elf

test.elf:     file format elf32-powerpc

Checking section . isrvectbl:
Checking section .text:

VIOLATION: Issue was detected in function ' check_test ':
 at:
 0x01003FFE: 1C 63 8C 24  e_add16i r3,r3,-29660
 0x01004002: 79 FD BF FF  e_bl    0x01008000 <test_func>

FAILED: Detected 1 total e200z branch violations, 0 violations ignored

c:\branchcheck>
```

- This example shows an example of the code that could fail to execute properly.
- The instruction at address 0x01027FFE is a 32-bit instruction
- Followed by a branch at address 0x01004002 to the test_func function at address 0x01008000
- 0x1008000 - 0x01004002 = 0x3FFE

Branch_displacement_erratum_10385_checker Run example (pass)

```
c:\branchcheck>branch_displacement_erratum_10385_checker.exe
core0_slave.elf

core0_slave.elf:     file format elf32-powerpc

Checking section .start:

Checking section .init:

Checking section .fini:

Checking section .text:

Checking section .text_vle:


PASSED: No e200z branch violations detected
```

- This example shows an example of an elf file that does not contain branches that meet the conditions of the erratum.

Branch_displacement_erratum_10385_checker Run example (ignore)

```
c:\branchcheck>branch_displacement_erratum_10385_checker.exe test.elf
--ignore-address 0x01003FFE


test.elf:     file format elf32-powerpc


Checking section . isrvectbl:
Checking section .text:


IGNORED: Issue was detected in function ' check_test ':
 at:
 0x01003FFE: 1C 63 8C 24  e_add16i r3,r3,-29660
 0x01004002: 79 FD BF FF  e_bl    0x01008000 <test_func>


PASS: Detected 1 total e200z branch violations, 1 violations ignored


c:\branchcheck>
```

- This example shows an example with 1 issue ignored.
- The instruction at address 0x01027FFE is a 32-bit instruction
- Followed by a branch at address 0x01004002 to the test_func function at address 0x01008000
- However, the –ignore-address is set to 0x01003FFE and is therefore ignored

TCL script output

- c:\branchcheck>tclsh8.5.exe e200z4_branch_displacement_e10385.tcl test.elf
- Checking core2branchexample.elf -- Processing segment from 0x1000000 .. 0x1000020
- Checking core2branchexample.elf -- Processing segment from 0x1001000 .. 0x1002004
- Checking core2branchexample.elf -- Processing segment from 0x1002100 .. 0x1002206
- Checking core2branchexample.elf -- Processing segment from 0x1002210 .. 0x1002256
- Checking core2branchexample.elf -- Processing segment from 0x1002260 .. 0x100325e
- Checking core2branchexample.elf -- Processing segment from 0x1003260 .. 0x1003290
- Checking core2branchexample.elf -- Processing segment from 0x1003290 .. 0x1003296
- Checking core2branchexample.elf -- Processing segment from 0x1003ff0 .. 0x100400e
- FAIL - illegal branch e_b at 0x1004002
- Checking core2branchexample.elf -- Processing segment from 0x1008000 .. 0x1008014
- Checking core2branchexample.elf -- Processing segment from 0x1008014 .. 0x1008030

SECURE CONNECTIONS
FOR A SMARTER WORLD

Notes

- 32-bit branch instructions can have a 15 bit branch displacement or 24-bit displacement, both conditional and non-conditional branches are impacted
  - 15-bit displacement values would be 0x3FFE, 0x7FFE, 0xBFFE, and 0xFFFE
  - 24-bit displacement: any multiple of the 15-bit displacement
- 16-bit branch instructions can only have an 8 bit displacement (therefore are not impacted)
- Branches that depend on the result of the preceding instruction cannot be dual-issued. There are other pairs of 32-bit instructions that cannot be dual issued.

- e200z4 single issue only cores are not impacted

Presentation Revision History

- June 14, 2016
  - Initial customer release
- June 23, 2016
  - Updated tool failing case slide to show the updated output message
  - Added tool passing case slide
  - Added statement to Notes “both conditional and non-conditional branches are impacted”
- June 27, 2016
  - Added warning that only sections marked executable in the elf file are checked.
  - Clarified – relative branches are impacted.
- June 28, 2016
  - Redrew memory map for clearer understanding.
  - Corrected address typographical error in description on slide 6.
- July 12, 2016
  - Added discussion of the new SRECORD option.
  - Add current status of compilers (as of July 11,2016)
  - Added example of ignoring a match case.
  - Corrected typo in address on the examples.
- July 18, 2016
  - Support was added to the utility to support Intel Hex format files. Presentation updated accordingly. File format auto detected added.
  - Added comment to contact the compiler vendor.
- August 10, 2016
  - 2 slides added for a secondary tcl script that will also identify erratum conditions. 1 a description of the tcl script and a second that shows the usage of the script. Added statement that the screened files are not modified (read only).