# GRFPU Floating-point controller: Missing FDIV/FSQRT Result

Technical note
Doc. No GRLIB-TN-0013
Issue 1.3

2017-12-21

# CHANGE RECORD

| Issue | Date | Section / Page | Description |
|---|---|---|---|
| 1.2 | 2017-11-17 | | First public release |
| 1.3 | 2017-12-21 | Section 4.2 | Overview of compiler workaround and list of toolchains with the workaround available. |
| | | | |

# TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Scope of the Document

This document describes a corner case present in the GRFPC floating point controller, which is the hardware used to interface the GRFPU floating point unit with the LEON3/FT and LEON4/FT processors. The corner case described in this document can cause a FDIV/FSQRT operation result not to be committed to the floating point register file. The issue manifests if;

- two instructions exist in the instruction flow between two FDIV/FSQRT instructions, and
- at least one of them is a floating point operation, and
- the processor encounters certain number of hold cycles after the first FDIV/FSQRT operation reaches execute stage of the pipeline.

## 1.2 Affected Components

Cobham and Aeroflex components that are affected are:

- GR712RC
- GR740 (first silicon revision used for prototypes, silicon revision 1 is NOT affected)
- UT699
- UT699E
- UT700

All GRLIB versions up to (including) b4206 are affected.

LEON3FT-RTAX components that are unaffected since they do not make use of the GRFPU unit and instead are implemented using the GFPU-lite that has a different floating-point controller.

## 1.3 Distribution

GRLIB users are free to use the material in this document in their own documents and to redistribute this document. Please contact Cobham Gaisler for inquires on other use.

### 1.3.1 Contact

For questions on this document, please contact Cobham Gaisler support at support@gaisler.com. When requesting support include the part name if the question is a specific device or the full GRLIB IP library package name if the question relates to a GRLIB IP library license.

## 1.4 How to check if a LEON3/FT or LEON4/FT design is affected

The GRLIB build ID is present in the AMBA plug&play area. The build ID is also reported by the GRMON debug monitor when connecting to the device.

If you have licensed GRLIB for use in your own FPGA or ASIC design, the GRLIB version can be seen in the file name of the downloaded release package, in the directory name after unpacking the release, and in the file lib/grlib/stdlib/version.vhd in the release file tree (constant grlib_build).

If the GRLIB build ID is smaller than 4207 then the GRLIB version is affected.

# 2 TECHNICAL DESCRIPTION

This section provides design details in order to explain the origins of the issue. Readers only interested in the software impact and possible workarounds may skip to section 3.

## 2.1 Pipelining

The following diagram shows an overview of the LEON3/FT, LEON4/FT integer unit pipeline and floating point controller (FPC) pipeline:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Integer pipeline: | Fetch | Decode | Regfile | Execute | Memory | Exception | Write-back | |
| FPC pipeline: | | Decode | Regfile | Execute | Memory | Exception | Write-back | Division |

The floating-point controller (FPC) contains an instruction pipeline that is synchronized with the processor pipeline, and in the typical case instructions will flow through both pipelines in lockstep. A typical (FADD, FMUL, etc) FPU operation is started when the instruction enters the execute stage and will be finished by the time the operation has reached the last (write-back) stage of the FPC pipeline. Because the FPC pipeline is synchronized to the processor pipeline it may be stalled during cache misses, but the actual floating-point unit (GRFPU) performing the operation is free running and does not stall. The FPU operation may finish before the operation has reached the last stage in the FPC and the FPC has result holding registers in every stage between execute and writeback to manage this condition.

## 2.2 Missing FDIV/FSQRT Result

For high-latency floating-point division and square-root operations (FDIV, FSQRT), the FPC pipeline is not long enough to guarantee catching the result out from the FPU before the instruction leaves the FPC pipeline. Therefore a special "Division" stage at the end of the FPC pipeline keeps track of the unfinished operation and writes the result when it has arrived. The Division stage can only hold one operation, and therefore there are interlocks in the FPC to guarantee that a second division will not reach the Divison stage before the previous one has completed. However, a corner case exists where this guarantee is violated, and a division/square-root operation can be lost because it reaches the division stage when a previous division/square-root has not written the result to the floating-point register file. Hence causing the current FDIV/FSQRT operation not being able to allocate the Division stage. Triggering this condition requires both a specific sequence and a specific combination of pipeline stall cycles to get the FPU results at a specific stage in the FPC. For the corner case to be triggered, the following conditions must be met:

1. Result of an FDIV/FSQRT instruction arrives at the write-back stage of the FPC pipeline (can be caused by pipeline stalls due to cache misses)
2. Two or three instructions after the FDIV/FSQRT in 1. is another FDIV/FSQRT operation.
3. At least two instructions between FDIV/FSQRT instructions, explained in the previous step, is a floating point operation that writes to floating-point register file apart from FDIV/FSQRT and these floating point operations do not have any data dependency to the FDIV/FSQRT instruction at 1.
4. By the time the second FDIV/FSQRT mentioned in 2. reaches write-back stage the result is not arrived yet from FPU.

If all the above conditions are met then the result of the second FDIV/FSQRT mentioned in 4. will not be written to register file. It should be noted that reproducing this problem can be very hard due to the result of first FDIV/FSQRT should arrive when the instruction is in the write-back stage. Since FDIV/FSQRT operation cycles depend on the operand values, the required stall cycles after the operation reaches access stage will vary.  In addition, that specific number of stall cycles might only be generated under certain circumstances. For example the stall cycles incurred by an instruction cache miss will depend on the current activity on the AHB bus and the memory controller behaviour.

# 3 FUNCTIONAL IMPACT

The design issue described in section 2 can result in incorrect software execution for instruction sequences described in this section.

## 3.1 Vulnerable Sequences

Vulnerable sequence that can trigger the corner case described in errata is:

1. FDIV/FSQRT instruction
2. Two or three instructions in which at least two of them is a FPop1 type operation (apart from FDIV/FSQRT) or a load operation to the floating point register file. In addition source or destination operands of all the FPop1 or load to floating point register file instructions in this slot must not depend on the destination register of the FDIV/FSQRT operation in 1. In addition, the source operands of all FPop2 type operations and store from floating point register file in this three instruction slot should not depend on the destination register of the FDIV/FSQRT in 1. The two instructions executed right after FDIV/FSQRT in 1. must not be a FDIV or FSQRT instruction.
3. FDIV/FSQRT instruction in which source and destination registers do not depend on the FDIV/FSQRT instruction in 1.

Note:

- All the instructions described above must be back-to-back in the instruction flow without any intervening instructions. This condition can be met if there is a PC-relative control-transfer instruction in any of the three instructions described in 2.
- For FPop1 type operations see SPARC Architecture Manual : Version 8, Table F-5
- For FPop2 type operations see SPARC Architecture Manual : Version 8, Table F-6
- A PC-relative control-transfer instruction includes branch on integer condition codes instructions (see SPARC Architecture Manual: Version 8, section B.21), Branch on floating-point condition code instructions (see SPARC Architecture Manual : Version 8, section B.22) and CALL instruction (see SPARC Architecture Manual : Version 8, section B.24).
- The CALL instruction mentioned in the previous point only includes the real CALL instruction (CALL label). The pseudo CALL instruction (call with register relative addressing) do not cause the erroneous behaviour described in this document.
- When checking dependencies between floating point operations the double and single property of the instructions must be taken into account. For example a floating point double operation which writes to register %f04 actually writes to both register %f04 and %f05. Hence an upcoming floating point single operation that uses register %f05 as source operand has a dependency. Or if a floating point single instruction writes to register %f05 and an upcoming floating double instruction has %f04 as a source it has a dependency since it reads both registers %f04 and %f05 as a source.

If all the conditions described above is met, the result of the FDIV/FSQRT operation in 3. may in rare cases not be written to floating point register file and this may cause a software malfunction.

The following examples illustrates vulnerable sequences:

Example-1:

```
        fdivd %f12,%f10,%f16
        fmuld %f10,%f6,%f8
        fmuld %f4,%f6,%f2
        fdivd %f10,%f4,%f24
```

Example-2:

```
        fdivd %f14,%f20,%f16
        ldd [%l3],%f8
        ldd [%l3] 8,%f10
        ldd [%l3] 12,%f12
        fdivd %f2,%f4,%f24
```

Example-3:

```
        fdivd %f12,%f10,%f16
        fbne L1
        fmuls %f4,%f6,%f2
        …
        …

L1:
        fmuls %f4,%f8,%f26
        fdivs %f10,%f4,%f24
```

Example-4:

```
        fdivs %f12,%f10,%f16
        nop
        fmuls %f10,%f6,%f8
        fmuls %f4,%f6,%f2
        fdivs %f10,%f4,%f24
```

# 4 WORKAROUND / MITIGATION

## 4.1 Workaround

Missing FDIV/FSQRT results can be avoided by avoiding the sequence described in section 2.3. This can be accomplished by ensuring that at least two instructions exist after an FDIV/FSQRT operation which is not a FPop1 type operation. If not, insert an extra NOP instructions after the FDIV/FSQRT until the vulnerable sequence is avoided.

## 4.2 Toolchain versions with workaround

Errata workarounds are available for BCC, RCC, and VxWorks. The workarounds include compiler fixes as well as patched assembly code for the system libraries. The following release versions and later will include the workarounds in toolchain and OS/libraries:

- BCC 1.0.50
- BCC 2.0.2
- RCC 1.2.22
- RCC 1.3-RC3
- VxWorks 6.7 1.0.21 – Toolchain 1.0.15
- VxWorks 6.9 2.0.8 – Toolchain 1.0.4
- VxWorks 7 – Toolchain 7.2.0.0
- MKPROM 2.0.63

The compiler workaround will do the following to prevent vulnerable sequences from being generated:

- Prevent FDIV and FSQRT instructions from being placed in a delay slot.
- Insert NOP instructions to prevent the sequence (FDIV/FSQRT) → (two or three floating point operations or loads) → (FDIV/FSQRT).
- Insert NOP instructions to prevent (FDIV/FSQRT) followed by a branch.
- Not insert NOP instructions if any of the floating point operations have a dependency on the destination register of the first (FDIV/FSQRT).
- Not insert NOP instructions if one of the floating point operations is a (FDIV/FSQRT).

Note that the compiler workaround will not make any modifications to inline assembly code in the user application. Any sensitive assembly instruction sequence will have to be modified manually.

The compiler workaround is activated by compiling with the "-mfix-gr712rc", “-mfix-ut699”, “-mfix-ut700”, or “-mfix-tn0013” flag. For BCC1, RCC 1.2, and VxWorks 6.7 the “-mtune=ut699” flag is used instead of “-mfix-ut699”.

The patches for the compiler workaround are listed below. They are included in the official GCC distribution for version 7.3.0 and later.

[SPARC] Errata workaround for GRLIB-TN-0013
https://gcc.gnu.org/viewcvs/gcc?view=revision&revision=255237
* config/sparc/sparc.c (sparc_do_work_around_errata): Use mem_ref
https://gcc.gnu.org/viewcvs/gcc?view=revision&revision=255393

## 4.3 Scan script

A scan script is provided to allow scans of disassembled programs for code that can trigger the issue described in this document. The script is available from:

http://www.gaisler.com/notes

Please see the header in the script file for usage instructions.

## 4.4 Risk of software errors

The risk of encountering this issue depends on the instruction sequences in the software running on the device. In addition several requirements on timing need to be fulfilled for the issue to manifest. At the time of writing Cobham Gaisler has not received any reports of the issue being triggered on customer system. The problem was found during internal tests.

Information furnished by Cobham Gaisler is believed to be accurate and reliable. However, no responsibility is assumed by Cobham Gaisler for its use, or for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Cobham Gaisler.

All information is provided as is. There is no warranty that it is correct or suitable for any purpose, neither implicit nor explicit.