LEON SRMMU Behaviour

Technical note 2015-10-27

Doc. No GRLIB-TN-0002

Issue 1.0

CHANGE RECORD

Issue	Date	Section / Page	Description
1.0	2015-10-27		First issue under this document name. This document replaces TN-LEON-SRMMU.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope of the Document

This document describes behaviour for the LEON/GRLIB Memory Management Unit (MMU). The behaviour of the memory management unit has changed starting at GRLIB build ID 4152 and this document explains the reason for the changes and the effects on software for the old and new behaviour.

1.2 Distribution

LEON3, LEON3FT, LEON4 and LEON4FT 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 distribution.

1.2.1 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.

1.2.1.1 Checking GRLIB version

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 are licensing GRLIB for use in your own FPGA or ASIC design, this 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).

1.3 Reference Documents

[SPARCv8] The SPARC Architecture Manual, Version 8, Revision SAV080SI9308, SPARC International Inc

2 SRMMU BEHAVIOUR

2.1 Overview

The LEON MMU is implemented based on the SRMMU description in appendix H of [SPARCv8]. Sections H.5 and H.6 of [SPARCv8] describes the behaviour of the Fault Status Register and the Fault Address Register. LEON processors implemented from versions prior to build 4152 of the GRLIB library have a guard for the FSR and FAR registers where one invalid address error prevents new invalid address errors from overwriting the registers. This has effects on the software handling of MMU faults.

This document assumes that the reader is familiar with the SPARC V8 reference MMU architecture. Please see [SPARCv8] appendix H before proceeding to the next sections.

2.2 Description of SRMMU behaviour before and after build 4152

LEON processors implemented with build IDs lower than 4152 (applies to Cobham GR712, UT699, UT699e, UT700) will only overwrite (overwrite here means updating FSR and FAR while FSR.FAV='1') an invalid address error when the MMU receives an AMBA ERROR response during page table traversal. In addition to this, these implementations will signal invalid address error for errors described in [SPARCv8] as translation errors. The table below describes differences in behaviour between the two versions:

State		New event	Behaviour of versions prior to build 4152	Behaviour build 4152 and higher
FAR.FAV	**FSR.FT**			
Don't Care	Don't care	AMBA ERROR response	FSR and FAR updated. FSR.FT=internal error	FSR and FAR updated. FSR.FT=translation error
0	Don't care	Any fault	FSR and FAR updated	FSR and FAR updated
0	Don't care	PTD is found in a level-3 page table, or a PTE has ET=3	FSR and FAR updated with invalid address error	FSR and FAR updated with translation error
1	Invalid address error	Any, except AMBA ERROR response (see first row for AMBA ERROR case)	No change	FSR and FAR updated according to [SPARCv8] section H.5.
1	Invalid address error	PTD is found in a level-3 page table, or a PTE has ET=3	No change	FSR and FAR updated with translation error

Table 1: Fault type reporting and overwrite conditions for different LEON MMU versions

2.3 Effects

The MMU behaviour prior to LEON/GRLIB build 4152 requires additional software handling to resolve MMU faults in cases where a MMU lookup alters the state of the MMU FSR and FAR registers but does not cause a trap where the registers are read to clear the FSR.FAV bit. In this case, implementations prior to build 4152 may contain old information in the FSR and FAR registers for a subsequent trap.

Three cases where this happens are described in the subsections below.

2.3.1 Branch prediction causing MMU faults

In LEON processors with branch prediction, the processor will assume that branches are always taken and will speculatively fetch the instructions at the branch target address. Consider the code sequence below:

```
...
nop
nop
cmp %g1, %g2
bne btarget
nop
nop
nop
...
```

In the above sequence, the processor in LEON/GRLIB implementations with build ID lower than 4152 will always fetch the instructions at the branch target *btarget*, regardless of if the branch is later taken or not. If *btarget* is an invalid address (MMU page table entry PTE.V='0') then the MMU FSR and FAR registers will be updated with information on the fault. For LEON/GRLIB implementations with build ID 4152 and higher, the branch prediction behaviour can be configurable via register %ASR17.

For LEON/GRLIB implementations with build ID lower than 4152, this will mean that a subsequent MMU fault will not update the FSR and FAR registers. A load or store to an invalid address, following the sequence above, will cause an MMU trap. But the FSR and FAR registers will contain information from the speculative instruction fetch.

In LEON implementations with build ID 4152 and higher, the latest access will update the FSR and FAR registers during normal operation.

2.3.2 Load and branch to invalid addresses

In LEON processors, prior to build 4152, the sequence below will cause a trap on the load instruction while the MMU FAR and FSR registers will contain information about the btarget address. This is assuming that both the *dtarget* and *btarget* addresses are invalid (PTE.ET=0).

```
...
nop
nop
set dtarget, %g2
ld [%g2], %g0
ba btarget
nop
nop
nop
...
```

In the above sequence, the processor will follow the branch and fetch instructions from the *btarget* address. The lookup for *btarget* is done before the lookup for load address *dtarget*. A lookup is then performed for *dtarget* and the processor will, correctly trap, on the load instruction. The MMU FSR and FAR registers will contain information on the invalid address error for the instruction fetch.

In LEON implementations with build ID 4152 and higher, the data access will update the FSR and FAR registers and the MMU FSR and FAR registers will contain information for the trapped instructions. When this has been resolved, a new trap will be generated for the branch to *btarget*.

2.3.3 Interrupt before branch instruction

In LEON processors prior to build 4152, the FAR and FSR registers may reflect old information if an interrupt occurs before a branch instruction:

```
...
nop
nop
nop <--- IRQ happens here
b btarget
...
```

The *btarget* address instruction fault will get stored. If the IRQ handler returns to the same address then the situation will be resolved since we trap again trap again but if it returns somewhere else (for example timer IRQ leading to task switch) then there could be an unhandled error data in the MMU.

3 REQUIREMENTS ON SOFTWARE

3.1 Overview

LEON implementations with build ID 4152 and higher do not need special consideration. MMU handlers conforming to the SPARCv8 SRMMU description will function correctly.

LEON implementations with a build ID lower than 4152 need to implement the following for handling MMU faults:

- When the processor traps on a load or store instruction, or due to the execution flow entering a new memory page, the handler needs to be aware that the FSR and FAR registers may contain information about a previous, non-handled, instruction fault.
- When the processor traps with data_access_exception, the MMU FSR and FAR registers may instead of the data access causing the trap, contain information about a previous mis-predicted instruction fault. It may also contain information about an instruction fault about to happen within 4 instructions after the trapping instruction.
- When the processor traps with any other exception, the MMU FSR and FAR registers may similarly contain information about a previous mis-predicted instruction fault or an instruction fault about to happen within 4 instructions after the trapping instruction. If this is not cleared then it will remain in the MMU until the next MMU-related trap.

For implementations with branch prediction, the fault above may be caused by branch prediction and would in this case not need to be handled. For implementations without branch prediction the fault is due to an instruction fault and is optional to handle (the fault will be regenerated after the data fault has been correctly handled).

When the MMU contains stale information from an old fault it is enough to read the FSR register, to clear the FSR.FAV flag, and then re-execute the failing instruction. This will regenerate the trap and, since FSR.FAV='0', will update FSR and FAR with the latest required information. This does not solve the case where the failing instruction is followed by a branch that causes the FSR and FAR register to be updated with an instruction fault.

The only known workaround for both cases is to:

1) make the data_access_exception trap handler capable of handling paging of instructions as well as data. (connect the instruction_access_exception and data_access_exception to the same handler and check the MMU FSR Access Type field instead).

2) clear the MMU error state whenever changing context (ensures instruction addresses are always valid for the current process).

3.2 Behaviour of different software versions

- Linux: The on-demand paging design of Linux minimizes the work performed by only mapping in accessed regions. MMU faults are part of the normal operation of Linux kernel and the trap handlers analyse the content of the FAR/FSR MMU registers. A workaround for the issues described in this document has been implemented to the 3.10 linux kernel, available from release 3.10.58-1.0.4 and onwards. The workaround is always enabled forall LEON targets.
  Linux versions prior to release 3.10.58-1.0.4 do NOT correctly handle he SRMMU behaviour in devices with build ID lower than 4152. As long as the FAR register contains addresses that are valid for the current process, there will be no malfunction. This appears to be the typical case since no user reports about this issue have been received during the 10+ years that the problem has existed. There is a possible performance degradation due to the uncecessary MMU page table traversal and all users are recommended to upgrade or patch their Linux kernels as there is a risk for malfunctions being triggered.

- VxWorks: The default LEON VxWorks MMU fault trap handling is to shutdown or stop when the kernel or Real-Time Process (RTP) causes an MMU instruction or data fault exception. The MMU fault is not analysed. Thus, a workaround is not applicable for VxWorks. The user can override the default behaviour with a custom trap handler or VxWorks exception hook. User MMU fault trap handlers should take this errata into account if FAR/FSR are being analysed. The VxWorks operating system maps all valid address ranges when an Real-Time Process (RTP) is created. VxWorks thereby ensures that MMU faults only happen when there is an unexpected fault. The real-time design of VxWorks avoids MMU faults since it would have be hard to predict the impact on an application's real-time behaviour by pausing the application while handling MMU page faults.

- No other operating systems distributed by Cobham Gaisler have MMU support.

4 FAQ

4.1 I am using a system with MMU enabled without problems. Do I need to do anything?

If your device has a build ID less than 4152 then your MMU will update FSR and FAR according to the old behaviour described in this document. Note that you may not notice this behaviour unless you have code sequences described in this document under the same conditions (load/store and branch to invalid address in sequence). Users of all devices prior to build ID 4152 should verify that their MMU trap handlers are equipped to handle stale values in FSR and FAR.