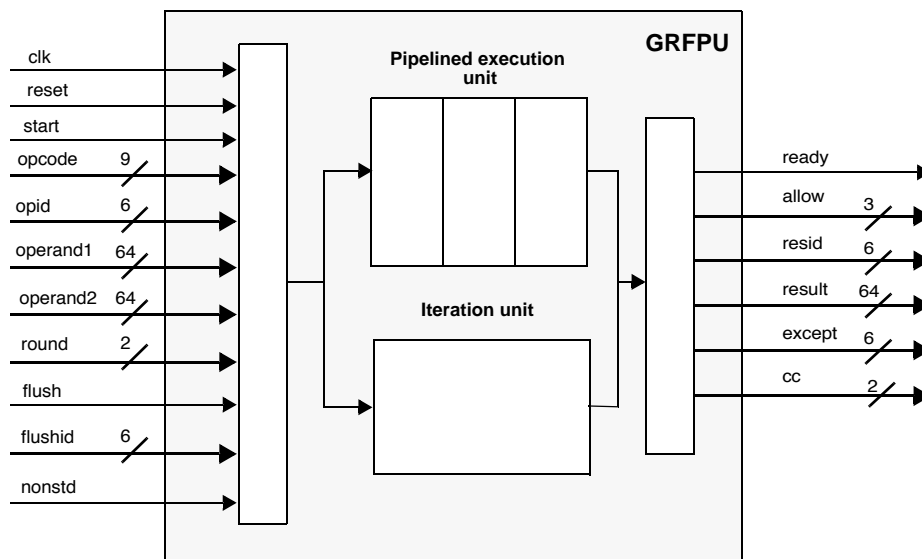


Features

- IEEE Std 754 compliant, supporting all rounding modes and exceptions
- Operations: fully pipelined add, subtract, multiply, divide, square-root, convert, compare, move, abs, negate
- Single and double precision (32- and 64-bit floats) data formats
- Supports all SPARC V8 floating-point instructions
- Fault-tolerant (FT) version available
- Actel CompanionCore
- Support for Fusion, IGLOO, ProASIC3/E, Axcelerator RTAX-S/DSP Product Families

Description

The GRFPU is a high-performance IEEE Std 754 compliant floating-point unit, supporting both single and double precision operands. All operations are IEEE Std 754 compliant, with the exception of denormalized numbers which are flushed to zero. The specified four rounding modes and the detection of exception conditions is fully supported.



Applications

The GRFPU high-performance floating point unit is designed for embedded applications, combining high performance with low complexity and low power consumption. The GRFPU has been designed to interface with the LEON3 SPARC processor.

The fault-tolerant version of the GRFPU in combination with the radiation tolerant Actel RTAX4000S/D FPGA gives a total immunity to radiation effects. This makes it ideally suited for space and other high-rel applications.



CompanionCore

1 Introduction

1.1 Overview

GRFPU is a High-performance Floating Point Unit implementing floating-point operations as defined in IEEE Standard for Binary Floating-Point Arithmetic (IEEE Std 754) and SPARC V8 standard (IEEE Std 1754). Supported formats are single and double precision floating-point numbers.

The GRFPU interfaces the LEON3 SPARC V8 integer unit via the GRFPU Control Unit (GRFPC), as shown in figure 1.

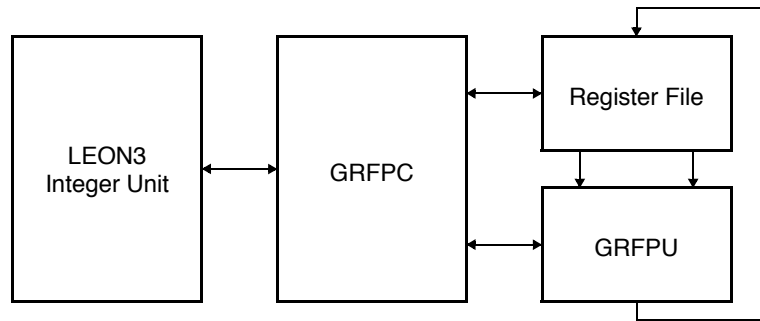


Figure 1. GRFPU integrated with the LEON3 processor via GRFPC Control Unit

The Fault-Tolerant (FT) version of the GRFPU and GRFPC include SEU protection by design. The FPU register file is protected using (32, 7) BCH coding, while all other registers are protected with TMR.

1.2 Signal overview

The combined GRFPU / GRFPC signals are shown in figure 2 and are directly compatible with the LEON3 processor core. Note that the interface signals are implemented as VHDL records and are not shown in detail.

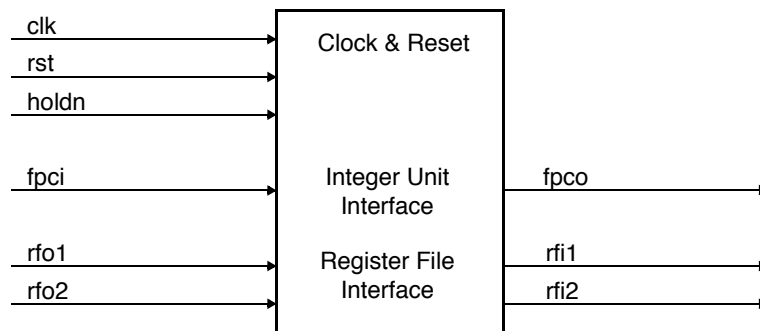


Figure 2. Signal overview

1.3 Implementation characteristics

The GRFPU / GRFPC core is inherently portable and can be implemented on most FPGA and ASIC technologies. Table 1 and 2 show the approximate cell count and frequency for different example configurations on Actel RTAX4000S, RTAX4000D, ProASIC3E and RT ProASIC3 (with and without Triple Modular Redundancy (TMR) for flip-flops).

Table 1. Implementation characteristics ((combinatorial / sequential cells / MATH blocks) / RAM blocks / MHz)

| Core configuration | RTAX4000S-1 | RTAX4000D-Std |
|--------------------|---------------------------------|----------------------------------|
| GRFPU/GRFPC | (20500 / 3200 / 0) / 4 / 25 MHz | (13500 / 3300 / 16) / 4 / 25 MHz |
| GRFPU/GRFPC-FT | (21000 / 3200 / 0) / 8 / 25 MHz | (14000 / 3300 / 16) / 8 / 25 MHz |

Table 2. Implementation characteristics (VersaTile cells / RAM blocks / MHz)

| Core configuration | A3PE3000L-1 | RT3PE3000L-1 | TMR |
|--------------------|--------------------|---------------------|-----|
| GRFPU/GRFPC | 35000 / 8 / 25 MHz | N/A | N/A |
| GRFPU/GRFPC-FT | N/A | 43000 / 16 / 20 MHz | Yes |
| | N/A | 36500 / 16 / 20 MHz | No |

The GRFPU and GRFPU-FT cores are available as pre-synthesized netlists only. The GRFPU(-FT) is only available bundled with the GRFPC(-FT) interface.

2 GRFPU - High-performance IEEE-754 Floating-point unit

2.1 Overview

GRFPU is a high-performance FPU implementing floating-point operations as defined in the IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754) and the SPARC V8 standard (IEEE-1754). Supported formats are single and double precision floating-point numbers. The advanced design combines two execution units, a fully pipelined unit for execution of the most common FP operations and a non-blocking unit for execution of divide and square-root operations.

The logical view of the GRFPU is shown in figure 3.

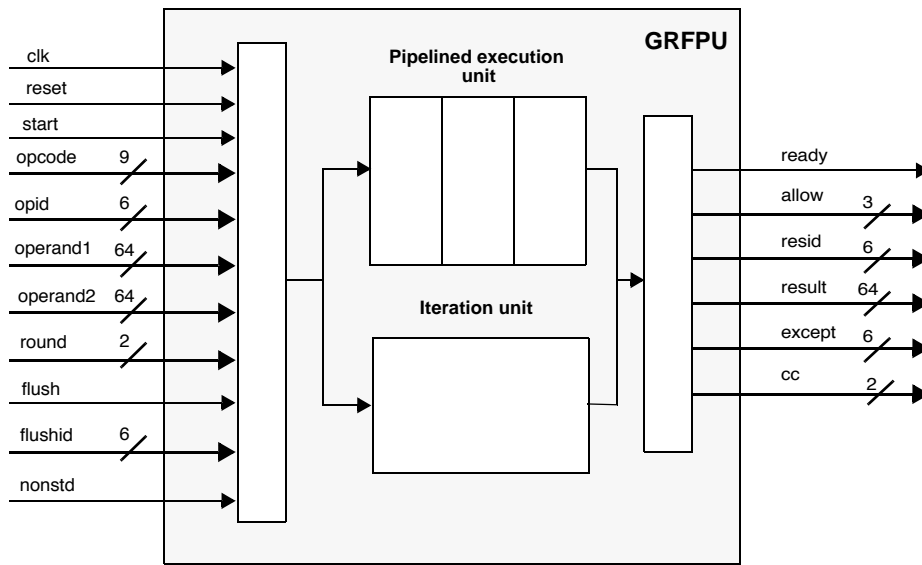


Figure 3. GRFPU Logical View

This document describes GRFPU from functional point of view. Chapter “Functional description” gives details about GRFPU’s implementation of the IEEE-754 standard including FP formats, operations, opcodes, operation timing, rounding and exceptions. “Signals and timing” describes the GRFPU interface and its signals. “GRFPU Control Unit” describes the software aspects of the GRFPU integration into a LEON processor through the GRFPU Control Unit - GRFPC. For implementation details refer to the white paper, “GRFPU - High Performance IEEE-754 Floating-Point Unit” (available at www.gaisler.com).

2.2 Functional description

2.2.1 Floating-point number formats

GRFPU handles floating-point numbers in single or double precision format as defined in the IEEE-754 standard with exception for denormalized numbers. See section 2.2.5 for more information on denormalized numbers.

2.2.2 FP operations

GRFPU supports four types of floating-point operations: arithmetic, compare, convert and move. The operations implement all FP instructions specified by SPARC V8 instruction set, and most of the operations defined in IEEE-754. All operations are summarized in table 3, with their opcodes, operands, results and exception codes. Throughputs and latencies are shown in table 3.

Table 3. : GRFPU operations

| Operation | OpCode[8:0] | Op1 | Op2 | Result | Exceptions | Description |
|---------------------------------|-------------------------------------|----------------|----------------|----------------|--|---|
| Arithmetic operations | | | | | | |
| FADDS FADDD | 001000001 001000010 | SP DP | SP DP | SP DP | UNF, NV, OF, UF, NX | Addition |
| FSUBS FSUBD | 001000101 001000110 | SP DP | SP DP | SP DP | UNF, NV, OF, UF, NX | Subtraction |
| FMULS FMULD FSMULD | 001001001 001001010 001101001 | SP DP SP | SP DP SP | SP DP DP | UNF, NV, OF, UF, NX UNF, NV, OF, UF, NX UNF, NV, OF, UF | Multiplication, FSMULD gives exact double-precision product of two single-precision operands. |
| FDIVS FDIVD | 001001101 001001110 | SP DP | SP DP | SP DP | UNF, NV, OF, UF, NX | Division |
| FSQRTS FSQRTD | 000101001 000101010 | - - | SP DP | SP DP | UNF, NV, NX | Square-root |
| Conversion operations | | | | | | |
| FITOS FITOD | 011000100 011001000 | - | INT | SP DP | NX - | Integer to floating-point conversion |
| FSTOI FDTOI | 011010001 011010010 | - | SP DP | INT | UNF, NV, NX | Floating-point to integer conversion. The result is rounded in round-to-zero mode. |
| FSTOI_RND FDTOI_RND | 111010001 111010010 | - | SP DP | INT | UNF, NV, NX | Floating-point to integer conversion. Rounding according to RND input. |
| FSTOD FDTOS | 011001001 011000110 | - | SP DP | DP SP | UNF, NV UNF, NV, OF, UF, NX | Conversion between floating-point formats |
| Comparison operations | | | | | | |
| FCMPS FCMPD | 001010001 001010010 | SP DP | SP DP | CC | NV | Floating-point compare. Invalid exception is generated if either operand is a signaling NaN. |
| FCMPES FCMPED | 001010101 001010110 | SP DP | SP DP | CC | NV | Floating point compare. Invalid exception is generated if either operand is a NaN (quiet or signaling). |
| Negate, Absolute value and Move | | | | | | |
| FABSS | 000001001 | - | SP | SP | - | Absolute value. |
| FNEGS | 000000101 | - | SP | SP | - | Negate. |
| FMOVS | 000000001 | | SP | SP | - | Move. Copies operand to result output. |

SP - single precision floating-point number

CC - condition codes, see table 6

DP - double precision floating-point number

UNF, NV, OF, UF, NX - floating-point exceptions, see section 2.2.3

INT - 32 bit integer

Arithmetic operations include addition, subtraction, multiplication, division and square-root. Each arithmetic operation can be performed in single or double precision formats. Arithmetic operations have one clock cycle throughput and a latency of four clock cycles, except for divide and square-root operations, which have a throughput of 16 - 25 clock cycles and latency of 16 - 25 clock cycles (see table 4). Add, sub and multiply can be started on every clock cycle, providing high throughput for these common operations. Divide and square-root operations have lower throughput and higher latency due to complexity of the algorithms, but are executed in parallel with all other FP operations in a non-blocking iteration unit. Out-of-order execution of operations with different latencies is easily handled through the GRFPU interface by assigning an id to every operation which appears with the result on the output once the operation is completed (see section 3.2).

Table 4. : Throughput and latency

| Operation | Throughput | Latency |
|--|---------------|---------------|
| FADDS, FADDD, FSUBS, FSUBD, FMULS, FMULD, FSMULD | 1 | 4 |
| FITOS, FITOD, FSTOI, FSTOI_RND, FDOI, FDOI_RND, FSTOD, FDTOS | 1 | 4 |
| FCMPS, FCMPE, FCMPEP, FCMPEPD | 1 | 4 |
| FDIVS | 16 | 16 |
| FDIVD | 16.5 (15/18)* | 16.5 (15/18)* |
| FSQRTS | 24 | 24 |
| FSQRTD | 24.5 (23/26)* | 24.5 (23/26)* |

* Throughput and latency are data dependant with two possible cases with equal statistical possibility.

Conversion operations execute in a pipelined execution unit and have throughput of one clock cycle and latency of four clock cycles. Conversion operations provide conversion between different floating-point numbers and between floating-point numbers and integers.

Comparison functions offering two different types of quiet Not-a-Numbers (QNaNs) handling are provided. Move, negate and absolute value are also provided. These operations do not ever generate unfinished exception (unfinished exception is never signaled since compare, negate, absolute value and move handle denormalized numbers).

2.2.3 Exceptions

GRFPU detects all exceptions defined by the IEEE-754 standard. This includes detection of Invalid Operation (NV), Overflow (OF), Underflow (UF), Division-by-Zero (DZ) and Inexact (NX) exception conditions. Generation of special results such as NaNs and infinity is also implemented. Overflow (OF) and underflow (UF) are detected before rounding. If an operation underflows the result is flushed to zero (GRFPU does not support denormalized numbers or gradual underflow). A special Unfinished exception (UNF) is signaled when one of the operands is a denormalized number which is not handled by the arithmetic and conversion operations.

2.2.4 Rounding

All four rounding modes defined in the IEEE-754 standard are supported: round-to-nearest, round-to-+inf, round-to--inf and round-to-zero.

2.2.5 Denormalized numbers

Denormalized numbers are not handled by the GRFPU arithmetic and conversion operations. A system (microprocessor) with the GRFPU could emulate rare cases of operations on denormals in software using non-FPU operations. A special Unfinished exception (UNF) is used to signal an arithmetic or conversion operation on the denormalized numbers. Compare, move, negate and absolute value operations can handle denormalized numbers and do not raise the unfinished exception. GRFPU does not generate any denormalized numbers during arithmetic and conversion operations on normalized numbers. If the infinitely precise result of an operation is a tiny number (smaller than minimum value representable in normal format) the result is flushed to zero (with underflow and inexact flags set).

2.2.6 Non-standard Mode

GRFPU can operate in a non-standard mode where all denormalized operands to arithmetic and conversion operations are treated as (correctly signed) zeroes. Calculations are performed on zero operands instead of the denormalized numbers obeying all rules of the floating-point arithmetics including rounding of the results and detecting exceptions.

2.2.7 NaNs

GRFPU supports handling of Not-a-Numbers (NaNs) as defined in the IEEE-754 standard. Operations on signaling NaNs (SNaNs) and invalid operations (e.g. inf/inf) generate the Invalid exception and deliver QNaN_GEN as result. Operations on Quiet NaNs (QNaNs), except for FCMPEs and FCMPEd, do not raise any exceptions and propagate QNaNs through the FP operations by delivering NaN-results according to table 5. QNaN_GEN is 0x7fffe00000000000 for double precision results and 0x7fff0000 for single precision results.

Table 5. : Operations on NaNs

| | Operand 2 | | | |
|-----------|-----------|----------|----------|----------|
| Operand 1 | | FP | QNaN2 | SNaN2 |
| | none | FP | QNaN2 | QNaN_GEN |
| | FP | FP | QNaN2 | QNaN_GEN |
| | QNaN1 | QNaN1 | QNaN2 | QNaN_GEN |
| | SNaN1 | QNaN_GEN | QNaN_GEN | QNaN_GEN |

2.3 Signal descriptions

Table 6 shows the interface signals of the core (VHDL ports). All signals are active high except for RST which is active low.

Table 6. : Signal descriptions

| Signal | I/O | Description |
|----------------------------------|-----|---|
| CLK | I | Clock |
| RST | I | Reset |
| START | I | Start an FP operation on the next rising clock edge |
| NONSTD | I | Nonstandard mode. Denormalized operands are converted to zero. |
| OPCODE[8:0] | I | FP operation. For codes see table 3. |
| OPID[7:0] | I | FP operation id. Every operation is associated with an id which will appear on the RESID output when the FP operation is completed. This value shall be incremented by 1 (with wrap-around) for every started FP operation. If flushing is used, FP operation id is 6 -bits wide (OPID[5:0] are used for id, OPID[7:6] are tied to "00"). If flushing is not used (input signal FLUSH is tied to '0'), all 8-bits (OPID[7:0]) are used. |
| OPERAND1[63:0] OPERAND2[63:0] | I | FP operation operands are provided on these one or both of these inputs. All 64 bits are used for IEEE-754 double precision floating-point numbers, bits [63:32] are used for IEEE-754 single precision floating-point numbers and 32-bit integers. |
| ROUND[1:0] | I | Rounding mode. 00 - rounding-to-nearest, 01 - round-to-zero, 10 - round-to-+inf, 11 - round-to--inf. |
| FLUSH | I | Flush FP operation with FLUSHID. |
| FLUSHID[5:0] | I | Id of the FP operation to be flushed. |
| READY | O | The result of a FP operation will be available at the end of the next clock cycle. |
| ALLOW[2:0] | O | Indicates allowed FP operations during the next clock cycle. ALLOW[0] - FDIVS, FDIVD, FSQRTS and FSQRTD allowed ALLOW[1] - FMULS, FMULD, FSMULD allowed ALLOW[2] - all other FP operations allowed |
| RESID[7:0] | O | Id of the FP operation whose result appears at the end of the next clock cycle. |
| RESULT[63:0] | O | Result of an FP operation. If the result is double precision floating-point number all 64 bits are used, otherwise single precision or integer result appears on RESULT[63:32]. |
| EXCEPT[5:0] | O | Floating-point exceptions generated by an FP operation. EXC[5] - Unfinished FP operation. Generated by an arithmetic or conversion operation with denormalized input(s). EXC[4] - Invalid exception. EXC[3] - Overflow. EXC[2] - Underflow. EXC[1] - Division by zero. EXC[0] - Inexact. |
| CC[1:0] | O | Result (condition code) of an FP compare operation. 00 - equal 01 - operand1 < operand2 10 - operand1 > operand2 11 - unordered |

2.4 Timing

An FP operation is started by providing the operands, opcode, rounding mode and id before rising edge. The operands need to be provided a small set-up time before a rising edge while all other signals are latched on rising edge.

The FPU is fully pipelined and a new operation can be started every clock cycle. The only exceptions are divide and square-root operations which require 16 to 26 clock cycles to complete, and which are not pipelined. Division and square-root are implemented through iterative series expansion algorithm. Since the algorithms basic step is multiplication the floating-point multiplier is shared between multiplication, division and square-root. Division and square-root do not occupy the multiplier during the whole operation and allow multiplication to be interleaved and executed parallelly with division or square-root.

One clock cycle before an operation is completed, the output signal RDY is asserted to indicate that the result of an FPU operation will appear on the output signals at the end of the next cycle. The id of the operation to be completed and allowed operations are reported on signals RESID and ALLOW. During the next clock cycle the result appears on RES, EXCEPT and CC outputs.

Table 4 shows signal timing during four arithmetic operations on GRFPU.

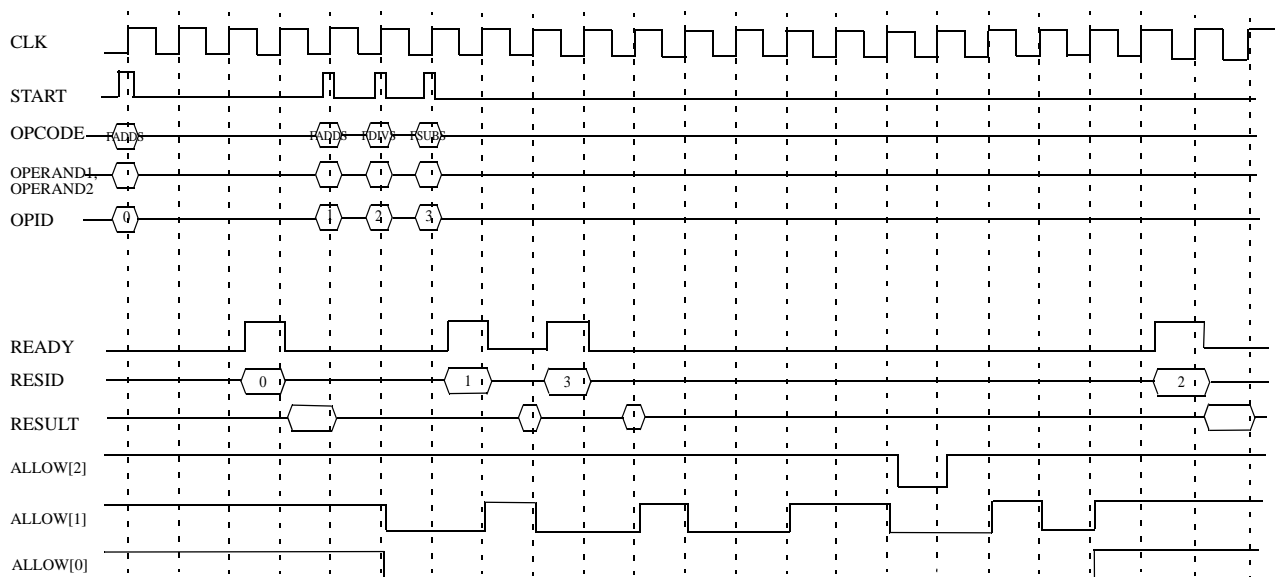


Figure 4. Signal timing

2.5 Shared FPU

In multi-processor systems, a single GRFPU can be shared between multiple CPU cores providing an area efficient solution. In this configuration, the GRFPU is extended with a wrapper. Each CPU core issues a request to execute an FP operation to the wrapper, which performs fair arbitration using the round-robin algorithm.

In shared FPU configuration, GRFPU uses an 8 bit wide id for each operation. The three high-order bits are used to identify the CPU core which issued the FP operation, while the five low-order bits are used to enumerate FP operations issued by one core. FP operation flushing is not possible in shared FPU configuration.

3 GRFPC - GRFPU Control Unit

The GRFPU Control Unit (GRFPC) is used to attach the GRFPU to the LEON integer unit (IU). GRFPC performs scheduling, decoding and dispatching of the FP operations to the GRFPU as well as managing the floating-point register file, the floating-point state register (FSR) and the floating-point deferred-trap queue (FQ). Floating-point operations are executed in parallel with other integer instructions, the LEON integer pipeline is only stalled in case of operand or resource conflicts.

In the FT-version, all registers are protected with TMR and the floating-point register file is protected using parity coding.

3.1 Floating-Point register file

The GRFPU floating-point register file contains 32 32-bit floating-point registers (%f0-%f31). The register file is accessed by floating-point load and store instructions (LDF, LDDF, STD, STDF) and floating-point operate instructions (FPop).

3.2 Floating-Point State Register (FSR)

The GRFPC manages the floating-point state register (FSR) containing FPU mode and status information. All fields of the FSR register as defined in SPARC V8 specification are implemented and managed by the GRFPU conforming to the SPARC V8 specification and the IEEE-754 standard. Implementation-specific parts of the FSR managing are the NS (non-standard) bit and *ftt* field.

If the NS (non-standard) bit of the FSR register is set, all floating-point operations will be performed in non-standard mode as described in section 2.2.6. When the NS bit is cleared all operations are performed in standard IEEE-compliant mode.

Following floating-point trap types never occur and are therefore never set in the *ftt* field:

- *unimplemented_FPop*: all FPop operations are implemented
- *hardware_error*: non-resumable hardware error
- *invalid_fp_register*: no check that double-precision register is 0 mod 2 is performed

GRFPU implements the *qne* bit of the FSR register which reads 0 if the floating-point deferred-queue (FQ) is empty and 1 otherwise.

The FSR is accessed using LDFSR and STFSR instructions.

3.3 Floating-Point Exceptions and Floating-Point Deferred-Queue

GRFPU implements the SPARC deferred trap model for floating-point exceptions (*fp_exception*). A floating-point exception is caused by a floating-point instruction performing an operation resulting in one of following conditions:

- an operation raises IEEE floating-point exception (*ftt* = *IEEE_754_exception*) e.g. executing invalid operation such as 0/0 while the NVM bit of the TEM field is set (invalid exception enabled).
- an operation on denormalized floating-point numbers (in standard IEEE-mode) raises *unfinished_FPop* floating-point exception
- sequence error: abnormal error condition in the FPU due to the erroneous use of the floating-point instructions in the supervisor software.

The trap is deferred to one of the floating-point instructions (FPop, FP load/store, FP branch) following the trap-inducing instruction (note that this may not be next floating-point instruction in the pro-

gram order due to exception-detecting mechanism and out-of-order instruction execution in the GRFPC). When the trap is taken the floating-point deferred-queue (FQ) contains the trap-inducing instruction and up to seven FPop instructions that were dispatched in the GRFPC but did not complete.

After the trap is taken the *qne* bit of the FSR is set and remains set until the FQ is emptied. The STDFQ instruction reads a double-word from the floating-point deferred queue, the first word is the address of the instruction and the second word is the instruction code. All instructions in the FQ are FPop type instructions. The first access to the FQ gives a double-word with the trap-inducing instruction, following double-words contain pending floating-point instructions. Supervisor software should emulate FPods from the FQ in the same order as they were read from the FQ.

Note that instructions in the FQ may not appear in the same order as the program order since GRFPU executes floating-point instructions out-of-order. A floating-point trap is never deferred past an instruction specifying source registers, destination registers or condition codes that could be modified by the trap-inducing instruction. Execution or emulation of instructions in the FQ by the supervisor software gives therefore the same FPU state as if the instructions were executed in the program order.

4 Reference documents

- [AMBA] AMBA Specification, Rev 2.0, ARM IHI 0011A, 13 May 1999, Issue A, first release, ARM Limited
- [GRLIB] GRLIB IP Library User's Manual, Aeroflex Gaisler, www.aeroflex.com/gaisler
- [GRIP] GRLIB IP Core User's Manual, Aeroflex Gaisler, www.aeroflex.com/gaisler
- [SPARC] The SPARC Architecture Manual, Version 8, Revision SAV080SI9308, SPARC International Inc.
- [IEEE] IEEE Standard for Binary Floating-Point Arithmetic, IEEE Std 754-1985

5 Ordering information

Ordering information is provided in table 7 and a legend is provided in table 8.

Table 7. Ordering information

| Product | Source code | Netlist | Technology |
|----------------|-------------|-----------|------------|
| GRFPU/GRFPC | N/A | EDIF/VHDL | Any |
| GRFPU/GRFPC-FT | N/A | EDIF/VHDL | RTAX |

Table 8. Ordering legend

| Designator | Option | Description |
|-------------------|----------------|---|
| Product | GRFPU/GRFPC | Floating Point Unit and Control Unit |
| | GRFPU/GRFCP-FT | Fault-Tolerant Floating Point Unit and Control Unit |
| Netlist | EDIF | EDIF gate-level netlist |
| | VHDL | VHDL gate-level netlist |
| Technology | AX | Axcelerator |
| | RTAX | Fault-Tolerant Axcelerator |
| | PROASIC3 | ProASIC3 |
| | PROASICE | ProASICE |
| | FUSION | Fusion |
| | IGLOO | IGLOO |

Table of contents

1 Introduction 2

 1.1 Overview 2

 1.2 Signal overview 2

 1.3 Implementation characteristics 3

2 GRFPU - High-performance IEEE-754 Floating-point unit 4

 2.1 Overview 4

 2.2 Functional description 4

 2.2.1 Floating-point number formats 4

 2.2.2 FP operations 5

 2.2.3 Exceptions 7

 2.2.4 Rounding 7

 2.2.5 Denormalized numbers 8

 2.2.6 Non-standard Mode 8

 2.2.7 NaNs 8

 2.3 Signal descriptions 9

 2.4 Timing 10

 2.5 Shared FPU 10

3 GRFPC - GRFPU Control Unit 11

 3.1 Floating-Point register file 11

 3.2 Floating-Point State Register (FSR) 11

 3.3 Floating-Point Exceptions and Floating-Point Deferred-Queue 11

4 Reference documents 13

5 Ordering information 13

Information furnished by Aeroflex Gaisler AB is believed to be accurate and reliable.

However, no responsibility is assumed by Aeroflex Gaisler AB for its use, nor 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 Aeroflex Gaisler AB.

| | |
|---------------------|--------------------------|
| Aeroflex Gaisler AB | tel +46 31 7758650 |
| Kungsgatan 12 | fax +46 31 421407 |
| 411 19 Göteborg | sales@gaisler.com |
| Sweden | www.aeroflex.com/gaisler |



Copyright © September 2009 Aeroflex Gaisler AB.

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