Chapter 2
The LEON3 Processor

This section contains a brief description of the LEON3 SPARC V8 processor implementation developed by Gaisler Research, with an emphasis on information relevant to the derived UTLEON3 microthreaded processor. The LEON3 processor is part of the *GRLIB* package that is under continuous development; the following description is based on *GRLIB* Version 1.0.17. The description provided in this section is organized in a top-down manner, starting with a view of the whole IP library *GRLIB* and going down to the LEON3 integer pipeline.

2.1 The GRLIB Library

GRLIB is a library of VHDL source codes of IP cores for designing a complete system on chip centered around the LEON3 processor [4,5]. The library is structured into directories that group IP cores according to the contributor's company. There are more subdirectories with complete IP cores organized in packages in the directories. These directories contain VHDL source codes of packages for simulation and synthesis and files with package configurations. A package configuration is propagated to VHDL entities via generics. The library is managed by an automated tool based on *GNU make* that manages configurations and compilation of packages for simulation and synthesis.

2.1.1 Two-Process Coding Style

The majority of the VHDL code in the GRLIB library is coded using the two-process method developed by Gaisler [2], where one process forms the combinational part of the logic and uses variables to compute the results, and the other process implements register updates. All buses are coded as records; this reduces the coding effort when adding or removing interface signals.

M. Daněk et al., *UTLEON3: Exploring Fine-Grain Multi-Threading in FPGAs*,
DOI 10.1007/978-1-4614-2410-9_2, © Springer Science+Business Media, LLC 2013

The advantage of this coding method is shorter simulation time, but the price paid is lower readability of the code for designers not familiar with this coding style (namely due to C-like sequential dependencies between variable assignments in the combinational process in contrast to the usual parallel nature of VHDL signal assignments). Newer versions of *doxygen* can be used to extract documentation from the VHDL sources.

2.1.2 GRLIB Directory Structure

The LEON3 processor consists of several parts. The main parts of the processor implementation are placed in the $*GRLIB*/*lib*/*gaisler*/*leon*3 directory. VHDL files with the SPARC V8 instruction codes and support functions for instruction disassembly during simulation are placed in the $*GRLIB*/*lib*/*grlib*/*sparc* directory. User designs with top-level files that instantiate complete SoC designs are located in the $*GRLIB*/*designs* directory.

2.2 LEON3

LEON3 is an implementation of the SPARC V8 architecture with several specific features. Figure 2.1 shows a block diagram of the LEON3 processor with optional parts (e.g. DIV32, MUL32, DSU, MMU).

The minimal configuration of the LEON3 processor consists of an integer pipeline, 3-port register file, instruction cache and data cache. A more powerful configuration can contain a memory management unit, 32-bit integer multiplication and division unit, 16-bit integer multiply-accumulate unit, debug interface and trace buffers according to its configuration. LEON3 has two extension interfaces, usually used by a floating-point unit and a co-processor.

More information about the SPARC V8 architecture and the LEON3 implementation are in [3, 5, 7, 15]. One of our initial design requirements was that the new UTLEON3 processor be binary compatible with the original LEON3 processor. To better understand the design decisions we took for UTLEON3 we describe the relevant parts of LEON3 in the following text.

2.2.1 Pipeline Operation

The LEON3 processor execution is controlled by the integer pipeline. The pipeline is implemented in seven independent stages that use delayed branches and several forwarding paths to achieve high performance. A schematic view of the pipeline is shown in Fig. 2.2.

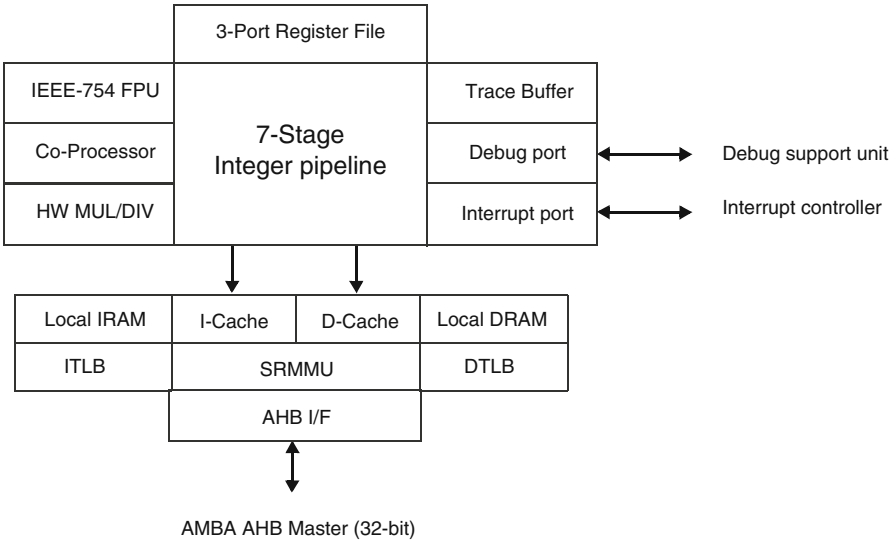

Fig. 2.1 Structure of the LEON3 processor

The integer pipeline is coded in the file *iu3.vhd*. The state of the whole *iu3* pipeline is captured in one signal, named *r*. This signal is a record of records, where each level-1 record corresponds to each pipeline stage. The level-2 records are tailored to the requirements of each pipeline stage, some items propagate through several pipeline stages. The structure of the pipeline data structure is included in Appendix A.

2.2.2 Instruction Set

Instructions in the LEON3 processor are encoded in three 32-bit formats. Instruction formats and an excerpt from the instruction table is included in Appendix B. The LEON3 integer pipeline implements the full SPARC V8 standard, including hardware multiply and divide instructions, and in addition it implements hardware multiply-accumulate instructions.

Instructions can be divided in six categories: load/store, integer arithmetic, control transfer, read/write control registers, floating-point operations and co-processor operations. There are several free places in the *Format 3* $(\text{op} = 2)$ instruction set table. The load/store instructions can use alternate address spaces defined by the 8-bit field *ASI* in the instruction code. LEON3 uses only a few address spaces, and there are a lot of spaces for possible extensions. There are three empty places for new branch (flow control) instructions.

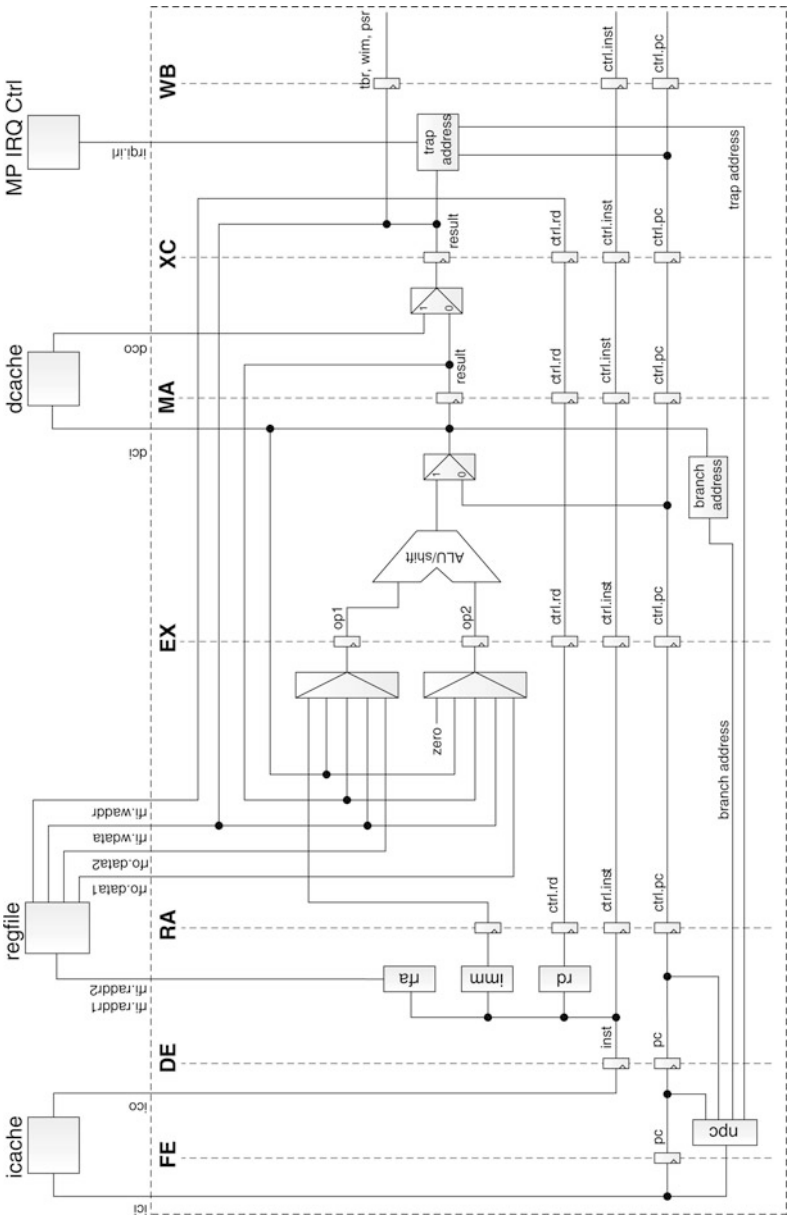

Fig. 2.2 The LEON3 integer pipeline – a simplified view

All instructions can behave differently in the user and privileged modes, and they can cause an exception if they are called in an improper mode. Load/store data from/to an improper address space can cause the invalid memory access exception.

2.2.3 Registers

The LEON3 processor includes all the necessary registers as defined in the SPARC V8 specification. It contains general-purpose registers (the *%r* registers) and control/status registers (*%psr*, *PC*, etc.) as shown in Appendix C.

The general-purpose registers are included in a register file that is connected to the processor instruction pipeline through three ports. The register file is organized as 8 global registers and an implementation dependent number of 16-register sets. Instructions can access all 8 global registers (*global*), 16 registers of the current window (*local*, *in*) and the 8 *in* registers of the next window (*out*). The current window is changed only by the *SAVE* and *RESTORE* instructions or when entering or returning from a trap. The SPARC V8 specification recommends to implement 8 register sets for compatibility reasons. Four *r* registers have a special function. The register *%r*15 (*%o*7) is used for saving the program counter of a *CALL* instruction. When a trap occurs, the *pc* and *npc* are copied into registers *%r*17 and *%r*18 (*%l*1 and *%l*2) of the trap's new register window. Register *%r*0 (*%g*0) behaves in a specific way; if *%r*0 is a source operand, the constant value 0 is read. When *%r*0 is used as a destination operand, the result of the operation is discarded.

2.2.4 Exceptions

LEON3 implements the general SPARC trap model. The table of the implemented traps and their individual priorities is in the *GRLIB IP User's Manual* [5]. Trapping is enabled by setting the bit *ET* in processor status register. LEON3 supports *single vector trapping* (*SVT*) that can reduce the code size of trap handlers. In the *SVT* mode all traps use a handler to reset a trap, and the trap type is indicated in the field *TT* in the register *TBR*.

2.2.5 Software Tools

The software tools needed for building a microthreaded processor from the LEON3 VHDL source codes are fully described in the *GRLIB User's Manual* [4]. These tools don't have to be modified.

Programs for the LEON3 processor can be processed using a modified GNU toolchain for the SPARC architecture. The source code of these tools is available on the *Gaisler Research* web site. The minimal setting requires at least the GNU assembler to program the processor.

2.3 From LEON3 to UTLEON3

The internal structure of the UTLEON3 processor is derived from LEON3, but certain parts are different. These are namely:

Instruction decoder	– new instructions that support microthreaded execution.
Integer pipeline	– new instructions, new program flow control mechanism (thread switch).
Register file	– new registers, new register state bits, new register allocation scheme.
Traps	– new traps due to error states in the microthreaded mode.
Caches	– new cache controllers that support delayed memory accesses (R/W), independent register file updates, and reference counting.
Program flow	– a distributed call stack equivalent stored in the register file, thread scheduler and instruction cache.

UTLEON3 maintains the standard SPARCv8 instruction formats and opcode table. New microthread management instructions have been implemented over specific cases of the SPARCv8 RDASR and WRASR instructions.

http://www.springer.com/978-1-4614-2409-3

UTLEON3: Exploring Fine-Grain Multi-Threading in FPGAs

Daněk, M.; Kafka, L.; Kohout, L.; Sýkora, J.; Bartosiński, R.

2013, XVIII, 222 p., Hardcover
ISBN: 978-1-4614-2409-3