

# A portable fault-tolerant microprocessor based on the SPARC V8 architecture

Jiri Gaisler

European Space Research and Technology Centre (ESTEC)

2200 AG Noordwijk, The Netherlands

jgais@ws.estec.esa.nl

voice: +31-715654880, fax: +31-715654295

**Abstract.** *This paper presents the LEON processor. The processor is based on the SPARC V8 architecture and implemented as a synthesisable VHDL model. A portable design style has been adapted to allow simple migration to new semiconductor technologies. Software transparent SEU-protection is implemented using TMR registers and cache parity. The processor has integrated SRAM and DRAM support, as well as a master/slave PCI interface.*

## 1 Introduction

In October 1997, an internal project (denoted LEON) was launched at ESTEC to study and develop a high-performance 32-bit processor for European space missions beyond year 2000. This paper presents the rational, design goals and architectural details of the project.

## 2 Background

With the development of the ERC32 (1997) and ERC32SC (1999), European space industry have unrestricted access to a radiation-tolerant 32-bit processor with a performance of up to 25 MIPS / 4 MFLOPS. Nevertheless, to stay competitive in the commercial space market and to meet the requirements of future applications, higher performance to a lower cost will be needed. The following requirements are anticipated for future missions:

- High performance (100 MIPS / 25 MFLOPS)
- Low component *and* system cost
- Efficient use of *cheap-of-the-shelf* (COTS) software development tools
- Unrestricted long-term availability of both components and software tools

Processor performance is typically a combination of architecture and process technology. Using state-of-the-art commercial technology, processors are today capable of more than 1,000 MIPS. By reusing commercial architectures and investing heavily in radiation-tolerant process technology, several high-performance processor for space applications are now available or under development in U.S. (RH32+, PowerPC, Pentium). A solution for European space industry could therefore be to use an U.S.-provided processor in demanding applications. However, this solution is not without problems. Radiation-tolerant processors are subject to (changing) U.S. export regulations, placing restriction on usage and re-exportation. Proposals have also been made to limit the availability of NASA technology in order to improve the competitiveness of U.S. companies on the commercial space market. In some cases, the processor is not available as a component, but only as a complete board or subsystem.

Looking at a European source for a future high-performance processor, the problems lie more on the technical side. Compared to U.S., European radiation-tolerant process technology is not as advanced, primarily due to lower level of funding. Also, most of the widely used processor architectures are developed and owned by U.S. companies. Even the processor design used in ERC32 is licensed from U.S. and cannot (easily) be modified or improved in Europe.

Long-term availability is a problem area for both European and U.S. solutions. So far, all used processors have been proprietary designs, manufactured and supported by a single company. If the company for some reason decides to discontinue the manufacturing of the processor, little can be done. Moreover, processor designs

tend to be specially adapted for a particular process or cell library, and difficult to port even if the design was made available.

The LEON project was started in late 1997 to study and develop a high-performance processor to be used in European space projects. The objectives for the project were simple: to provide an open, portable and non-proprietary processor design, capable to meet the identified requirements for performance, software compatibility and low system cost. The following design goals were defined for the processor:

- **Portability.** To take advantage of the latest semiconductor technology, the processor should be easily portable without requiring manual redesign or special cell-library development.
- **Modular design.** The processor design should be modular to simplify modifications and improvements, and to be used as a core in SOC designs.
- **Standard interfaces.** The processor should have standardised interfaces to simplify system integration and to reuse commercial cores, components and tools.
- **Software compatibility.** The processor should be compatible with both the currently used software development tools and COTS software packages.
- **Use of *single-event-upset* (SEU) sensitive semiconductor process.** To reduce cost and increase performance, it should be possible to implement the processor on SEU sensitive semiconductor processes while still meeting the reliability requirement of space applications.
- **Scalability.** The processor should be usable in both low- and high-end applications with minimum hardware and software overhead.

### 3 Definition and architecture

#### 3.1 Processor architecture

To maintain software compatibility with ERC32 and the current line of software development tools, the SPARC architecture was selected for the LEON processor. It could be argued that other architectures (e.g. x86, PowerPC, MIPS, ARM) are supported by more COTS software tools, however, a major advantage of the SPARC architecture is that is completely “open”, well documented, and under control of a non-profit organisation (SPARC International). Re-implementing proprietary architectures often bring legal problems in form of alleged patent or copyright violations. A well-known case is INTEL versus AMD regarding the x86 architecture, but other cases also exist (e.g. MIPS versus LEXRA, INTEL versus NEC).

#### 3.2 Cache memory

Cache memories have not been used in space applications due to the problems of calculating the worst-case execution time in hard real-time systems. To increase the performance and reduce power consumption, it is however necessary to use on-chip cache memories. The cache memories must be implemented and used in a way that will make it possible to accurately calculate the worst-case execution time for a code segment. Cache simulators and other tools can be used to predict cache hit-rates and processor performance, but cannot predict the effects of asynchronous events (interrupts). Since an interrupt can occur anywhere in the program, it is difficult to predict which impact it will have on the cache and the overall execution time. To accurately calculate the worst-case execution time for a program in the presence of asynchronous interrupts, the interrupt rate as well as the impact on the cache has to be determined and bounded.

Cache locking has often been used to guarantee the execution time of a particular code segment, but this technique lowers the overall cache hit rate and can only be used on small code segments. The important aspect in real-time systems is predictability rather than maximum performance, and the developed solution for LEON is to automatically disable the cache when an interrupt is taken. The interrupt handling routine will not evict any cache lines and the cache will maintain the state it had before the interrupt. The cache is re-enabled upon the return from the interrupt handler. The worst-execution time can be determined by adding the nominal execution time with the execution time of the interrupt handler. The nominal execution time can be calculated using a cache simulator or similar tool, while the execution time of the interrupt handler can be calculated directly since it does not use the cache. The maximum interrupt rate has to be determined from the application.

The LEON cache memory is implemented as a separate instruction and data cache. Both caches are direct-mapped and the data cache implements write-through policy. A cache with higher associativity (multi-set) would yield higher hit rates but require custom RAM cells which would affect portability. The simple direct-mapped caches can be implemented with ordinary single-port static RAM cells, which exist in most cell libraries.

#### 3.3 Memory interface

A fast and efficient memory interface is a key parameter to reach high performance. Commercial processors are using dynamic RAM (DRAM) in conjunction with cache memories to find a compromise between cost and memory bandwidth. In space applications, static RAM (SRAM) have been exclusively used due to the poor radiation characteristics of DRAM. An increasing number of space applications are now requiring memory sizes above

8 Mbyte, making it difficult to use SRAM. These applications often operate in low-earth orbits, where the radiation environment is less severe. To reduce cost and weight of the applications, DRAM will most likely be used in the future. The LEON processor therefore implements a flexible memory interface allowing both SRAM and DRAM to be attached without additional external logic. The memory is organised in 32-bit words with a 7-bit SEC/DED BCH code for error-detection and correction. To be able to use the processor in low-end applications, 8- and 16-bit devices can also be attached.

### 3.4 I/O interface

The choice of I/O bus has a large impact on overall system design. Due to the continuously increasing device integration, more and more I/O devices are placed on the computer main board, rather than as earlier on the backplane bus. To standardize the interconnection between complex I/O devices and the CPU, Intel defined the PCI bus. The PCI bus is a synchronous multi-master bus with transfer rates of up to 132 Mbyte/s in 32-bit mode and 528 Mbyte/s with the 64-bit and 66 MHz extensions. The PCI bus has also been extended to an backplane bus called CompactPCI. Due to the wide acceptance, numerous I/O devices with PCI interface are available on the market and at least 10 different companies provide PCI cores for ASIC design. For space applications, the PCI bus have several positive characteristics; it includes a parity bit for error-detection, it has low power consumption and it can be implemented with device technologies available for space use. In addition, commercial PCI devices can be used during prototyping to reduce cost. The availability of PCI cores will also reduce the development cost of new, space-specific, I/O devices.

The selected solution for LEON is to provide a 32-bit PCI interface directly on-chip. The interface can work both as PCI host and PCI target. However, "traditional" I/O devices can still be attached to the memory bus.

### 3.5 On-chip peripherals

To simplify system integration, a set of standard peripheral functions are included in the LEON processor. These include timers, UARTs, watchdog, I/O ports and interrupt controller. Figure 1 below shows the processor block diagram.

## 4 Error-detection and fault-tolerance

One important objective for the LEON development is to be able to use a SEU-sensitive semiconductor process. To maintain correct operation in the presence of SEU errors, extensive error-detection and error-handling functions are needed. The goals for LEON is to detect and tolerate one error in any on-chip register without software intervention, and to suppress effects from SEU errors in combinational logic.

To meet the fault-tolerance goals, all on-chip registers can be implemented using *triple-modular redundancy* (TMR), i.e. three registers in parallel and a voter selecting the majority result. The benefit of such a scheme is that error-masking and error-removal is implicit, and than no glitch is produced on the output when an SEU occurs. The register file can be provided with a 32-bit SEC/DED EDAC which will check and correct used register data. On-chip cache memories (tags and data) can be provided with parity bits which are checked when the tag or data are used. A parity error will translate into a cache miss,

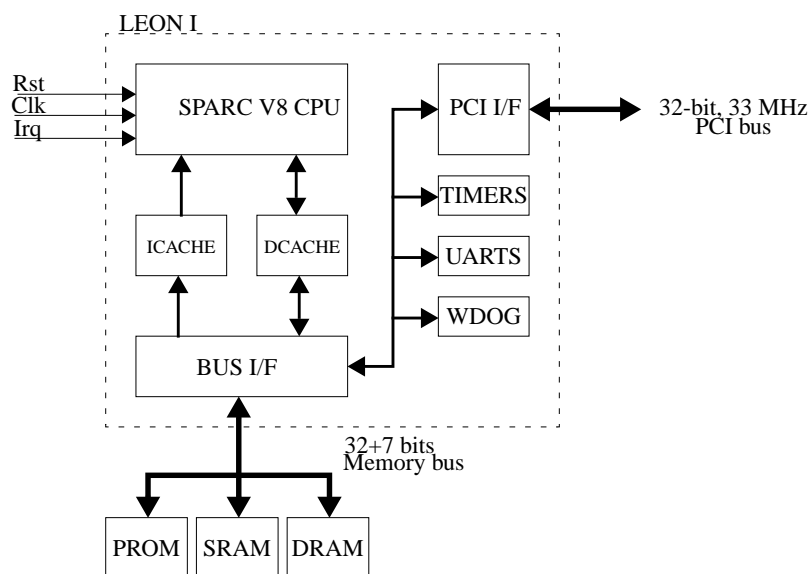


Figure 1: LEON block diagram

causing the cache to be updated with the correct data from the main memory. Since the data cache implements a write-through policy, valid data can always be loaded from the main memory.

## 5 VHDL model

To achieve portability, the LEON is implemented as a synthesisable VHDL model (figure 2). A full-custom design style would have resulted in both higher performance and smaller area, but would be tied to a particular process and difficult to modify and port. In addition, the VHDL model can also be used for low-cost prototyping using FPGA devices and for board-level system simulation.

The LEON VHDL model does not require any custom macro-cells apart from static RAM which are used for the caches and register file. The model is extensively configurable: the number of register windows, cache size and organisation, fault-tolerance options, clocking scheme and speed/area trade-off can be defined through a single configuration file. The model is written at highest possible RTL level to allow designers to quickly understand the model if modifications are necessary. Each entity typically consists of only two processes, one implementing the combinational logic and one sequential elements (registers). This scheme also benefits simulation speed; on an ordinary PC (pentium @ 350 MHz) approximately 15 instructions/s is reached using the

ModelSim-5.2 simulator. To simplify debugging, a built-in disassembler can list executed instructions in the simulator window.

## 6 Current status

At present (April 1999), the integer unit, cache memory, peripheral functions and memory interface have been implemented. The integration of a floating-point unit (FPU) and PCI interface is planned to be completed in Q3 1999. The current implementation has been successfully synthesised into an Altera 10K130E-1 FPGA, achieving 15 MIPS (@ 15 MHz). Synthesis runs targeting Temic MG2 (0.5  $\mu\text{m}$ ) processes yielded a clock frequency above 50 MHz, while 100 MHz was reached using a commercial 0.35  $\mu\text{m}$  processes.

The overall complexity of the processor is limited; the integer unit is approximately 20,000 gates (exclusive register file), the cache controller and memory interface 25,000 gates (exclusive cache RAM) and the peripheral functions 20,000 gates. The complexity of the FPU and PCI interface have not yet been determined, but are estimated to 30,000 respectively 20,000 gates. A complete processor would then have a complexity of 115,000 gates, or approximately 8  $\text{mm}^2$  on a typical 0.35  $\mu\text{m}$  process. In a SOC design, the integer unit and cache/memory controller would consume 45,000 gates, or 3  $\text{mm}^2$ . If the fault-tolerance functions are enabled (TMR and cache parity), the overall area increase is 35%, while the timing impact is 10% (approximately).

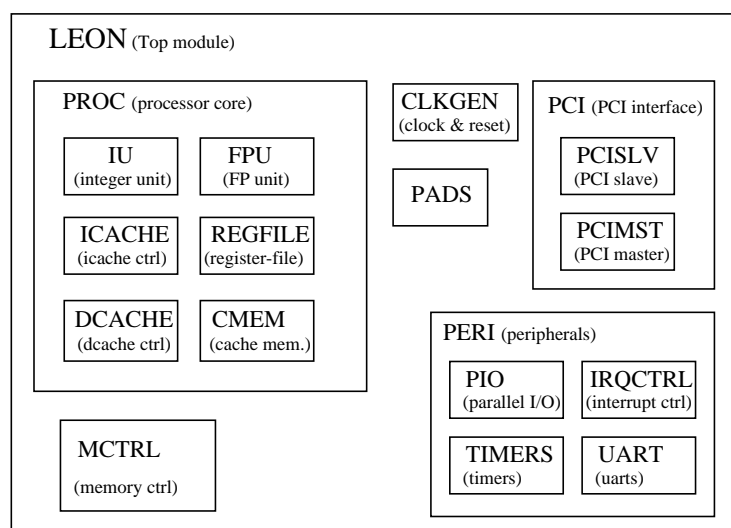


Figure 2: LEON VHDL model block diagram

## 7 Availability and future development

The LEON VHDL model will be made available in two versions: a basic, fully-synthesisable model will be freely available, while an enhanced model incorporating the fault-tolerance features, PCI interface and floating-point unit can be licensed by European space companies. The basic model is planned to be released in September 1999, and the enhanced model in December 1999.

By freely distributing the basic LEON design, thereby potentially increasing the user base, it is believed that improvements and correction to the design will be contributed in much the same way as with *open-source* software. This has indeed been the case for other freely distributed models such as ERC32 and EVI32.

When the LEON development is completed, it is planned to manufacture a prototype device on a commercial process for further evaluation. After the evaluation, a radiation-tolerant device suitable for space applications could then be manufactured on the best available process at that time.

## 8 Conclusion

Considering the anticipated requirements for a future processor, the LEON design meets them as follows:

- **High performance:** portable design ensures the use of best available process; fault-tolerance to enable use of fast but SEU-sensitive processes.
- **Lower component and system cost:** integration of necessary peripheral functions; integrated SRAM and DRAM controller; integrated PCI interface; usable as core in SOC designs; fault-tolerance to enable use of SEU-sensitive processes without having to develop SEU-hardened cell libraries.
- **Software compatibility and use of COTS software:** fully compatible with ERC32 software tools (Aonix & VxWorks); fault-tolerance features completely software transparent; compatible with free software tools such as GCC, GNAT, RTEMS, GDB, SIS and DDD.
- **Long-term availability:** non-proprietary, portable design; open architecture.

