

# Status update on GR716 Rad-Hard Microcontroller For Space Applications

**Fredrik Johansson (Speaker), Sandi Habinc, Jan Andersson  
Fredrik Sturesson, Nils-Johan Wessman**

*Cobham Gaisler, Kungsgatan 12, SE-411 91, Göteborg, Sweden*

*Tel: +46 31 775 86 50*

*info@gaisler.com*

**Steven Redant, Laurent Berti, Geert Thys**

*IMEC*

*Kapeldreef 75, 3001 Leuven, Belgium*

*Tel: +32 16 28 11 41*

*info@imec.be*

**Claudio Monteleone**

*European Space Agency,*

*Keplerlaan 1, PO box 299, NL-2220AG Noordwijk, The Netherlands*

*Tel: +31 71 56 56791*

*claudio.monteleone@esa.int*

## ABSTRACT

This paper describes the mixed-signal microcontroller GR716 targeting embedded control applications with hard real-time requirements. Prototype devices are currently being tape-out in the activity *Microcontroller for embedded space applications*, initiated and funded by the European Space Agency (ESA).

The presentation and paper will describe the architecture, functionality and simulation results of the device. This abstract describes an on-going development where the devices are in the stage to be taped-out.

## BACKGROUND

Software based data acquisition, data processing and simple control applications are widely used in spacecraft subsystems. They allow implementation of software based control architectures that provide a higher flexibility and autonomous capabilities versus hardware implementations. For this type of applications, where limited processor performance is required, general purpose microprocessors are usually considered incompatible due to high power consumption, high pin count packages, need of external memories and missing peripherals. Low-end microcontrollers are considered more attractive in many applications such as:

- Propulsion system control
- Sensor bus control
- Robotics applications control
- Simple motor control
- Power control
- Particle detector instrumentation

- Radiation environment monitoring
- Thermal control
- Antenna pointing control
- AOCS/GNC (Gyro, IMU, MTM)
- RTU control
- Simple instrument control
- Wireless networking

In these kind of applications the microcontroller device should have a relatively low price, a low power consumption, a limited number of pins and must integrate small amount of RAM and most of the I/O peripherals for control and data acquisition (serial I/Fs, GPIO's, PWM, ADC etc.). The requirements for memory and program length are usually minimal, with no or very simple operating system and low software complexity.

## MICROCONTROLLER APPLICATIONS

Spacecraft subsystem control and monitoring of parameters such as power supply voltages, currents, pressures and temperatures are ideal applications for the LEON3FT microcontroller. Bridges between different communication standards or interface of an equipment towards a higher level controller or the central On Board Computer (OBC) are also ideal applications for the LEON3FT microcontroller.

The LEON3FT microcontroller can perform advanced data handling to offload any higher level controller or the central On Board Computer (OBC). By hiding the data handling details the transmitting data volume can be reduced and simplified functionalities and timing requirements are requested to the higher level controller.



- SRAM/PROM.
- Scrubber with programmable scrub rate for all embedded memories and external PROM/SRAM and SPI memories
- System
  - On-chip voltage regulators for single supply support. Capability to sense core voltage for trimming of the embedded voltage regulator for low power applications
  - Power-on-reset, Brownout detection and Dual Watch Dog for safe operation. External reset signal generation for resetting companion chips
  - Crystal oscillator support
  - PLL for System and SpaceWire clock generation
  - Low power mode and individual clock gating of functions and peripherals
  - Temperature and core voltage sensor
  - External precision voltage reference for precision measurement
  - Four programmable DMA controllers with up to 16 individual channel
  - Timer units with seven 32-bit timers including watchdog
  - Atomic access support for all APB registers (AND, OR, XOR, Set&Clear).
  - Peripheral access control
  - Embedded trace and statistics unit for profiling of the system
- Peripherals
  - SpaceWire with support for RMAP and Time Distribution Protocol
  - Redundant MIL-STD-1553B BRM (BC/RT/BM) interface
  - Multiple CAN 2.0B bus controllers
  - Six UART ports, with 16-byte FIFO
  - Two SPI master/slave serial ports
  - SPI4SPACE. Hardware support for SPI protocol 0,1 and 2 in HW for SPI for SPI4SPACE
  - Two I2C master/slave serial port
  - PacketWire interface
  - PWM with up-to 16 channels. PWM clock support upto 200 MHz
  - Up to 64 General input and outputs (GPIO) with external interrupt capability, pulse generation and sampling
  - Four single ended Digital to Analog Converters (DAC), 12-bit at 3MS/s
  - Four differential or eight single ended Analog to Digital Converters (ADC) 11-bit at 200KS/s with programmable pre-amplifier and support for oversampling. Dual sample and hold circuit integrated for simultaneously sampling
  - External ADC and DAC support up to 16-bit at 1MS/s
- I/O

- Configurable I/O selection matrix with support for mixed signals, internal pull-up/pulldown resistors
- LVDS transceivers for SpaceWire or SPI4SPACE
- Dedicated SPI boot ROM support for configuration
- Supply
  - Single 3.3V±0.3V supply or separate Core Voltage 1.8V±0.18V, I/O voltage 3.3V±0.3V

### Processor performance and determinism

In order to improve determinism, the LEON3FT microcontroller contains only a local instruction and data static RAM with fixed response times. All EDAC units in the system have the same latency and behaviour in the corrected as in the uncorrected case. This also applies to the CPU, so dynamic SEU handling schemes such as the LEON3FT pipeline restart on error options is not used.

Local instruction RAM tightly coupled to the LEON3FT CPU will be the main memory to execute the software. Due to its direct connection to the CPU, the execution of the software will be deterministic. For applications where full cycle-level determinism is not needed, it will also be possible to execute software from an external SRAM.

The local instruction memory will be implemented using dual-port RAM. The memory's second port will be connected to the main system AHB. This will allow modifying of the local instruction RAM without the intervention of the CPU. The contents of this memory will be protected against SEU errors with EDAC and scrubbing.

If the DMA peripherals and the processor are connected to a shared single-port memory, or to a memory via the same bus, and try to simultaneously access the shared resource then the DMA activity will have an effect on the execution time. On the other hand DMA activity will have no impact on SW execution time by using a dual-port on-chip data RAM and a separate bus for the DMA peripherals. This means that there is a separate access path for the CPU core to local instruction and data RAMs that is unaffected by concurrent DMA activity.

For applications demanding determinism on nested interrupts, a special interrupt handling scheme will be implemented in software where nested interrupts are allowed to occupy one additional register window. The number of levels of nested interrupts that can be handled without additional timing penalty depends on the complexity of the software implementation.

In the architecture, deterministic interrupt latency will be achieved by:

- Running software (including interrupt handlers)

from local RAM and accessing any data needed during the interrupt handling through port separate from AMBA ports.

- Adapting the register window usage (using a flat model) structure to avoid unexpected window over/underflow traps. This is done in the compiler and application code, and most OS code does not need modification.
- The alternate window pointer feature from the SPARC V8E extension to allow window over/underflow handlers to run with traps enabled.
- Register file partitioning to allow partitioning of the register file (the windows) to different “contexts”. Contexts can for example be threads to speed up context switching and/or interrupt contexts to dedicate windows to ISRs.

### LEON/REX and Runtime improvements

The new LEON/REX alternate window pointer feature (AWP) support and the improved interrupt single vector trap handler (SVT) have been tested and characterized in a series of measurements running on prototype hardware.

By delaying a timer interrupt N clocks into an overflow or underflow trap handling the interrupt latency and interrupt latency jitter as a result of SAVE/RESTORE can be quantified.

Five different software runtime configurations were benchmarked:

- Current BCC SVT
- Improved BCC SVT
- Improved BCC SVT with AWP
- Current BCC MVT
- Current BCC MVT with AWP

In order to understand where the latencies comes from the time from the interrupt is asserted to the time the ISR is reached is split up in three parts presented in the plots below:

- Interrupt assert to acknowledge (assert to first instruction of trap executed)
- Acknowledge to the Interrupt Service Routine (first instruction of trap to first instruction of ISR)
- Total latency (Assert to ISR first instruction)

The worst case interrupt latencies seen when an interrupt is asserted on top of a window underflow/overflow handler are presented in the table below. The highlighted rows are estimates results that

can achieved in the LEON/REX environment.

Latency / Config	Assert to Acknowledge		Acknowledge to ISR		Total – Assert to ISR	
	Overflow	Underflow	Overflow	Underflow	Overflow	Underflow
CWP, SVT	134	143	539	281	673	424
CWP, new SVT	70	62	296	166	366	228
AWP, new SVT	35	34	202	166	207	200
CWP, MVT	60	52	262	152	322	204
AWP, MVT	25	24	188	152	193	176

Table 1: Worst case latencies measured

The new LEON/REX architecture also improves the “context” switching by allowing partitioning of the register file (the windows) to different “contexts”. By assigning windows to software threads or interrupts the software execution don't have to wait for the LEON3FT processor to store used windows on the stack.

Benchmark on prototype systems shows a large reduction of software execution time of switching “context”

### Flat register model for low response time

Very low interrupt response time can be achieved by utilising LEON3FT microcontrollers large number of register windows. Low interrupt response time can be achieved by using a FLAT register window model. The FLAT register window model eliminates window underflow/overflow-trap and the number of cycles executed with trap disables.

A real-time application compiled with the single register window model (FLAT) using GCC does not issue any SAVE or RESTORE instructions so it executes inside a single window. When a trap is taken (interrupt trap), the current window is decremented with one.

Each unique trap handler is to execute each level of the interrupt nest hierarchy in its own window. Since the FLAT ABI preserves local and input registers as needed, there is no need for the interrupt trap handler to store these. The output registers become the next interrupts input registers and are thus "protected" by the ABI. Only the global registers have to be stored/restored when jumping between interrupt nest levels. There is room to temporarily store the global registers in the local registers of the trap window.

Examples provided with software environment version 2.0.2 demonstrates that the total latency from interrupt assert to ISR can be as low as less than 28 clock cycles.

Benefits from using FLAT register model:

- Interrupt ack to ISR is 22 cycles (constant)
- Interrupt exit is 15 cycles
- No registers stored to memory in interrupt trap
- No registers loaded from memory in interrupt trap (except to get ISR handler)

- Supports interrupt nesting
- No application specific considerations

Limitations of FLAT register model:

- If SVT is used, then 11 cycles have to be added for the ack to ISR cycles
- Requires one register window per nested interrupt request level

### **Programmable DMA controller**

Cobham Gaisler has developed a DMA controller able to perform concurrent programmable sequences of data transfers between any on-chip peripherals in the AMBA address space. The DMA controller is able to transfer data both between peripherals, between peripherals and memory and between memory areas. If the accessed memory is internal or external does not matter, as long as the memory is mapped into AMBA address space reachable from the AHB bus where the core is mapped.

The DMA controller has been specifically designed to offload the CPU and provide DMA capabilities to peripherals that do not have an internal DMA engine. The CPU is offloaded by the fact that the peripheral event is directly routed to the DMA controller. By routing events directly to the DMA controller or even directly between peripherals, these interrupts are in effect offloaded from the CPU. These reduce also the number of concurrent interrupts the CPU must handle and that may erode the system determinism.

### **Pin-multiplexing**

The device shall be an attractive solution for a wide range of applications. Because of the small package and high number of interfaces, the functionality of the pins must be configurable and the pins must be shared between several peripherals. The number of configurable user pins has been chosen to be 64.

### **Clocking, reset and maximum frequency**

The maximum operating frequency for the AMBA system is 50 MHz. The device can have separate clock signal inputs for system, SpaceWire and MIL-STD-1553B interfaces. The clocks signals can also be derived from single source via clock multipliers and dividers inside the device.

In order to avoid problems with reset sequencing, the device has one single reset input that is sequenced internally to provide reset signals to the different clock domains within the device.

### **SUPPORT FOR PROFILING AND DEBUGGING**

The device provides debug interfaces via the JTAG and

UART. The dedicated Debug bus allows non-intrusive debugging since the DSU, trace buffers and performance counters can be accessed without causing traffic on the Processor AHB bus.

The design also supports filtering for both the AHB and instruction trace buffers.

The LEON3 statistics unit provides performance counters, with support for filtering, for a large number of events, including:

- Data write buffer hold
- Branch prediction miss
- Total/Integer/Floating-point instruction count
- Total execution count
- AHB bus statistics for Processor AHB bus and Master I/O AHB bus

The interrupt controller in the design supports interrupt time stamping with time stamps interrupt line assertion and processor interrupt acknowledge.

### **SUPPORT FOR PROM-LESS APPLICATIONS**

The device provides an easy access for systems that want to avoid having a boot-PROM connected to the device and prefer to upload software remotely.

The device can be accessed and remotely configured via SpaceWire, SPI, UART and I2C.

### **TARGET TECHNOLOGY AND PACKAGE**

The technology used is UMC 180 nm, using the DARE library from IMEC, and the package is a 132 pin CQFP

### **SOFTWARE SUPPORT**

The architecture is already supported by all operating systems and tool-chains provided by Cobham Gaisler.

### **CONCLUSION**

The device in development is a SPARC V8 microcontroller that is based on the well known LEON architecture. The device is a prototype for a possible future device targeted at microcontroller applications and will have several new features that are not found in contemporary LEON devices. This includes architectural features to improve determinism, availability of the device in a low pin-count package, and support for the reduced instruction set.