IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, Issue 3, Ver. V (May - Jun. 2014), PP 48-54
www.iosrjournals.org

Performance Estimation of a LEON 3FT Processor Based Design for Spacecraft Applications

Shruthi N[#1], Prashant Kulshreshtha[*2], Dinakaran E[*3], Dr. Girish V Attimarad[#4], Mr. Subramanya Udupa[*5],
[#]*Dayananda Sagar College of Engineering, Department of Electronics & Communication Bangalore, INDIA*
**Control Electronics Group, ISRO Satellite Centre Bangalore, INDIA*

Abstract: *The content of this paper is intended to highlight the performance of the 32-bit LEON 3FT processor in terms of execution speed in comparison with the currently used 16-bit processor. Therefore, the work related to this paper is considered to be an upgrade over the previous implementation. The proposal for such an enhancement has been materialized successfully by means of a LEON 3FT processor based design on a protoboard, extensively supported by suitable simulator and debugger tools and environment. A set of selected benchmark programs have been executed on the superior processor mainly to track the execution times. The benchmark suite is selected in order to suit, support and verify the functionality of the possible list of real-time tasks in the spacecraft application. Taking advantage of this enhancement, the end user can time critical, real-time tasks efficiently. The results of this paper show a remarkable giant leap in system performance in terms of execution speed.*

Index Terms: *LEON 3FT, performance analysis, benchmark programs, computational capabilities, execution speed*

I. Introduction

It is evident that the developments in the field of processors and technology are enormous. There is advancement towards miniaturization, performance and operation control of the processors being used in real-time systems. Keeping in mind the possible drawbacks of the current processors being used, a designer often looks out for much more superior processors, processors whose performance can rule its predecessors. In an similar attempt to implement a much better processor in the design of CPU for spacecraft systems, this paper proposes to implement a 32-bit LEON 3FT processor in competition with the previous 16-bit MIL-STD-1750 based processor, which is becoming nearly obsolete. The overall performance of the spacecraft is expected to take a leap, by utilizing the extended features of the 32-bit processor. One can expect a leap in execution speed, throughput, and better power saving features, enhanced data handling capability and an appreciable reduction in chip size and hence weight. There is definitely a need for such an enhancement. Section II highlights the features of the 32-bit processor, which provide extensive support to the design being implemented later in Section IV. The primary objective of the design is upgrade and modernization of the existing CPU design of spacecraft applications.

A detailed discussion about the factors influencing the performance of a system is taken care of in Section III. Section V discloses the test environment followed by the test results in Section VI. There is a mention of the conclusion of this paper and a proposal to continue future work in the field of Fault tolerance, Pipeline, Caching, Interrupt Handling and Memory Interfacing supported by Worst-Case Execution time (WCET) analysis.

II. Insight Into The Leon 3ft Processor

The LEON family of processors was originally designed by the European Space Research and Technology Centre (ESTEC), under the European Space Agency (ESA), in the year 1997, but was later handed out to Gaisler Research (Aeroflex Gaisler), which is suitable for system-on-a-chip (SoC) designs. However, the fault tolerant version of LEON, the LEON 3FT aims at providing supports to work in harsh, highly variant space environments. The LEON 3FT is a monolithic, SPARC V8 processor which follows the RISC based Harvard architecture and projects a 7-stage instruction pipeline with a predominant fault-tolerant feature. Full access to all processor registers and cache memory is provided through the debug support unit (DSU). User can set instruction breakpoints and perform single stepping through the DSU. The DSU acts as an AHB slave and can be accessed by any one of the following AHB

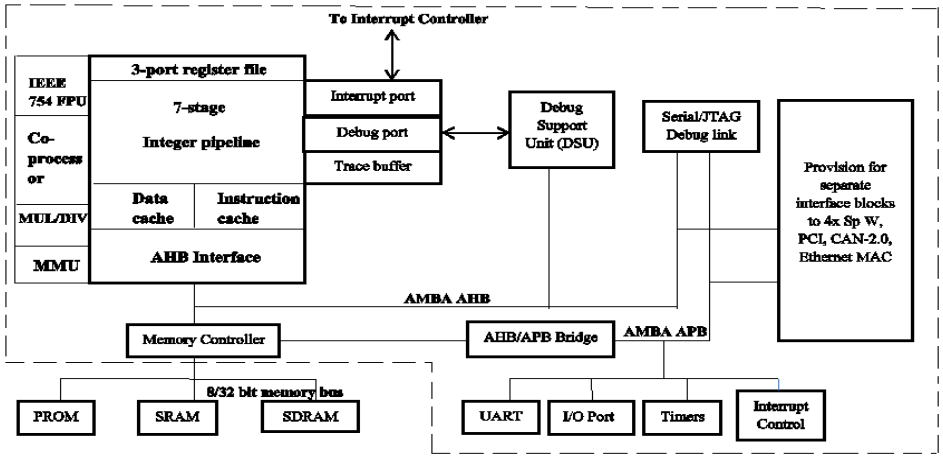


Figure 1 – Functional Block Diagram of LEON 3FT processor

masters: the debug UART, the JTAG port, the PCI port, or a Space Wire link using RMAP. A block diagram of LEON3 architecture is shown in Figure 1.

III. Performance Analysis

With the advent of a new, superior processor being brought into the spacecraft application, it becomes a mandatory exercise to analyse the processor's performance in terms of clock cycles for instruction execution as well as the overall execution time. Reducing or simplifying the instruction set is not the goal of any architectural modification. Instead, each of the architectures ultimately aims at the *IMPROVEMENT OF OVERALL SYSTEM PERFORMANCE*. We, in this paper have made an attempt to highlight the fact that the RISC supported LEON 3FT processor is a better performer when compared to the CISC supported MA 31750. The performance of a processor is defined by the time requires by the processor to complete a specific task, task may be a program, an algorithm or a benchmark. The three factors determining the performance of a processor are related as:

Performance or Time per task = C * T * I

where C - Cycles per instructions, T - Time per cycle and I - Instruction per Task. It is evident from the above relationship that reducing any of these three factors improves the performance of the system. CISC systems tend to reduce the 'I' factor in order to reduce the Timer per Task whereas RISC designs reduce the 'C' and 'I' factors effectively to enhance system performance.

A. Time per Instruction (CT) **-** The 'C' and 'T' factors are complementary i.e. the product of 'C' and 'T' yield the Time per Instruction factor. If 'T' is reduced (increasing clock speed), the time for one cycle reduces which in turn reduces the amount of work that can be accomplished in a single cycle. In most of the processors, the designer is not very much concerned whether it is a matter of faster clock rates (short cycle times) and more instruction cycles or slower clock rates (longer cycle times) and fewer instruction cycles – it is the total Time per Instruction that is more important. The execution time shoots up as the number of cycles increase.

B. Cycles per Instruction (C) **-** The strength of RISC processors lies in reduction of cycles per instruction and Time per Cycle. RISC design incorporates the "Pipeline" feature to reduce the 'C' factor by a factor equal to the depth of the pipeline. While pipeline can improve the execution speed a program, there might also arise some problems. All the stages of a pipeline process may not take the same time slice to execute. This makes it much harder to synchronise the various stages of the different instructions. Also, some instructions may even depend on the results of some earlier instructions. The pipeline on the MA 31750 always holds two 16-bit words. On the other hand, the LEON 3FT integer unit uses a single instruction issue pipeline with seven stages with the Harvard architecture implementation. The stages are **FE** (Instruction Fetch), **DE** (Decode), **RA** (Register Access), **EX** (Execute), **ME** (Memory), **XC** (Exception) and **WR** (Write).

C. Time per Cycle (T) **-** The time required to perform a machine cycle (factor 'T') is determined by **instruction decode time, instruction operation time**, **instruction access time** and a**rchitectural simplicity**.

D. Instructions per Task (I) **–** For RISC designs, the 'I' factor is a major disadvantage among the C, T and I factors. A possible set of solutions is being provided in RISC designs to reduce the number of instructions per task. Solutions include optimization of compilers and operating system

support. It is possible to make extremely fast memory but this is only practical for small amounts of memory for cost, power and signal routing reasons. The solution is to provide a small amount of very fast memory known as a CPU cache which holds recently accessed data. Cache memories were not used much in space applications due to the problems of calculating the WCET in hard real-time systems. To increase the performance and reduce power consumption, it is however necessary to use on-chip cache memories and also monitoring the WCET. The paper [2] briefing the "Prototype Execution-time Analyser for LEON" (PEAL) project funded by ESA/ESTEC and executed in 2006 clearly states that a cache-equipped processor puts forward the below factors which may affect the execution time in real-time. (a) The total size of the program code and data in relation to the cache size, (b) location of the program code and data, (c) history of code and data addresses accessed by the program in the past, and (d) interrupts and pre-emptions occurred during program execution.

The LEON cache memory is implemented as a separate instruction and data cache. Apart from some of the crudest methods to find a solution like disabling caches or by using caches fully and freely, systematic methods yielding safe upper bound on the WCET were proposed. Some of the methods are (i) static cache-aware WCET analysis [3, 4]; and (ii) measurement-based WCET analysis [5, 6] with some systematic way to specify and measure the test coverage to ensure a sufficiently reliable result.

IV. Hardware Implementation

The design of the CPU system for spacecraft applications is implemented on a protoboard, the four major modules of the protoboard being the I/O & Power Supplies, Processor, 1553 communication and Memories. The features of the then proposed design were as follows: (1) Usage of high density EEPROMs for software storage and minimal PROM for Boot Code. (2) Boot code can perform EEPROM update by telecommands in addition to BOOT functions. (3) Program execution from SRAM for faster execution, EEPROM being slow and more susceptible to upsets. (4)16 bit level translators to interface to external +5V I/O bus – to achieve +3.3 V for I/O & +2.5 V for core. (5) RAM-transceiver-arbitration logic integrated - Mil 1553B controller. (6) Software tool set that supports both Ada & C (cross-compilation).

The design implementation diagram is shown below in Figure 2 followed by a brief explanation.

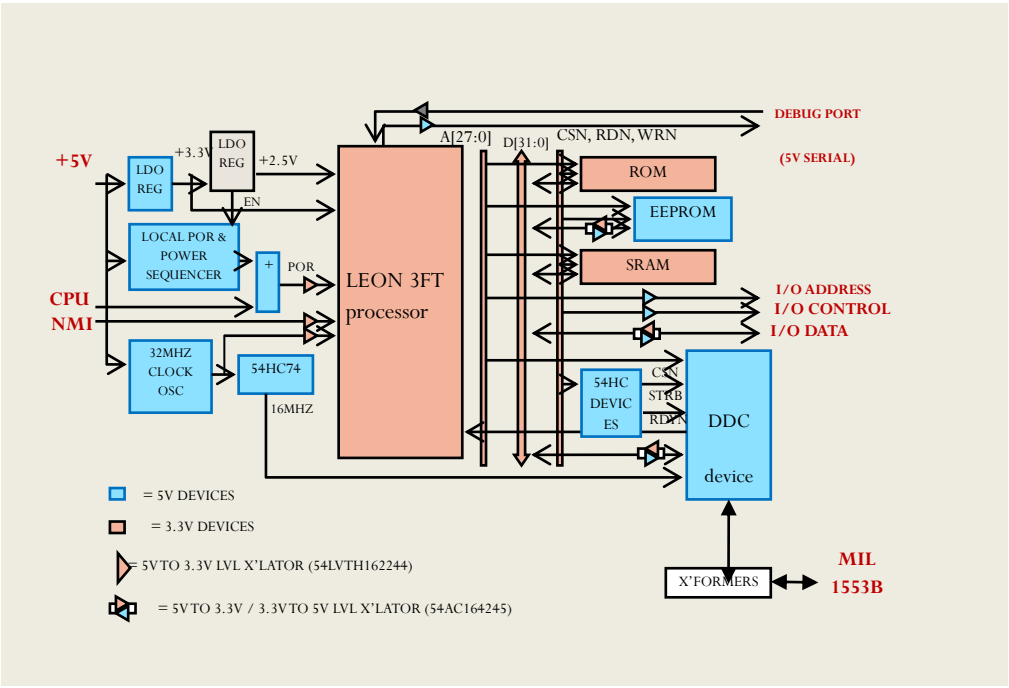


Figure 2 – Design Implementation Diagram

a) Bus - The external memory address bus is 28-bit wide whereas the data bus is 32- bit wide with 8 check bits (6 CBs in MA 31750). The control bus comprises of dedicated chip select, read and write signals for various memory blocks.

b) Memory – Memory Controller acts as the interface between memory (PROM, EEPROM and SRAM) and the AHB bus. Requirement is 32K × 32 of boot PROM, 512K × 32 of EEPROM & 1.5M × 32 of SRAM and all three can be EDAC protected using (39, 7) BCH code, nevertheless, PROM does not require EDAC implementation. PROM is 3.3V operated. It is slower than SRAM and has the boot program residing on it. We

observe that even though the OBC PROM requirement is only 32K × 32, the configuration is 32K × 40, since EEPROM requires additional 8 bits (512K × 40) for implementing the EDAC logic and both EEPROM & PROM share the same memory bank with respect to the processor; we extend a common configuration of 40 for both PROM & EEPROM. EEPROM is slowest & more vulnerable to disturbances. Therefore, there arises a need to load the program from EEPROM to SRAM. One can interface up to 1GB SRAM externally through the Memory Controller. The SRAM area is divided into five RAM banks and the size of each bank can be set as varying from 8Kbyte to 256 Mbyte through the MCFG2 configuration register. A SRAM read constitutes two data cycles and 0-3 wait states.

PROM is fabricated with QML-qualified radiation-hardened technology and is designed for use in systems operating in radiation environments. PROM operates over the full military temperature range, requires a single 3.3 V ± 5% power supply, and is available with TTL-compatible I/O. Power consumption is typically 15mW/MHz in operation and is less than 10mW/MHz in the low power- enabled mode. The PROM operation is fully asynchronous, with access time < 60 ns.

c) Level Translators – 5V to 3.3V translators are implemented using 54LVTH162244. On the other hand, 5V to 3.3V and vice versa translations are implemented using 54AC164245.

d) *Power on Reset Generation & Power Sequencing* - The first start-up is the I/O (requiring 3.3V) and next is the Core (requiring 2.5V). Proper power sequencing of the processor is achieved by bringing up V_{DD} to its recommended minimum operating voltage of 3.0V, and then delaying t_{VCD} clock cycles before bringing up the V_{DDC} supply. If power is applied to the V_{DDC} supply pins while V_{DD} is less than 3.0V, excessive current or damage to the device could occur. Power sequencing is needed when various types of electronic equipment must be powered up or down in groups, rather than all simultaneously. The design was proposed to implement suitable delays using RC based circuits. The delay values in the design for I/O, core signals and the RESET signals in this design are 10ms, 15ms and 25ms.

e) Clock – The 16-bit processor competed with a frequency of 12 MHz, the current 32-bit processor promising a theoretical capability of nearly 48 MHz But, at higher frequencies, power consumption increases and also, board design is complex. So, a maximum frequency of operation of 32MHz is chosen.

f) *Low Dropout Regulators* - Low dropout regulators are used to provide regulated voltages to I/O and the core of the processor. LDOs improve transient response. The advantages of a low dropout voltage include a lower minimum operating voltage, higher efficiency operation and lower heat dissipation. LDO regulator is a DC linear voltage regulator which can operate with a very small input–output differential voltage and provides output voltages of 1.5V, 1.8V, 2.5V & 3.3V with 1.21V reference voltage. The only disadvantage of LDOs is their weight. Normal regulators cannot be used.

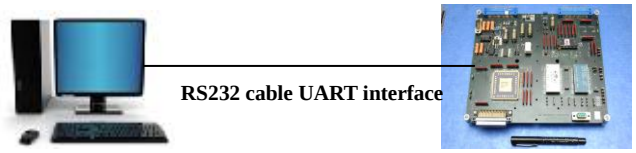
g) *1553B Protocol communication* –1553B is used for intra subsystem communication as well as to communicate with the external world. Shared RAM & arbitrary logic are both required, but not as a separate implementation in a FPGA because the DDC device which has in-built shared RAM & arbitrary logic is used. This reduces cost, production time & size, hence, weight.

V. Test Environment

This section intends to provide a brief description of the various tool sets and utilities as well as the development environment in which the tests related to this paper were carried out. All tests were carried out on the LEON 3FT core, SPARC V8 compatible processor. The compilation and simulation support is provided by GNAT Pro from AdaCore. GNAT Pro is a compiler and software development tool set for Ada programming language in a cross- compilation environment. By default, it assumes Ada 95 however it may be implemented in ADA 83/2005/2012 as well. All testing tools are installed on a LINUX workstation; GRMON is a general debug monitor for the LEON processor, and for SOC designs based on the GRLIB IP library. The monitor connects to a dedicated debug interface on the target hardware, through which it can perform read and write cycles on the on-chip bus (AHB).

The debug interface can be of various types: the LEON2 processor supports debugging over a serial UART and 32-bit PCI, while LEON3 also supports JTAG, Ethernet and SpaceWire (using the GRESB Ethernet to SpaceWire bridge) debug interfaces. On the target system, all debug interfaces are realized as AHB masters with the debug protocol implemented in hardware. GRMON can operate in two modes: command-line mode and GDB mode. In command-line mode, GRMON commands are entered manually through a terminal window. In GDB mode, GRMON acts as a GDB gateway and translates the GDB extended-remote protocol to debug commands on the target system. In our test setup, GDB gateway has been utilized. The test set up is represented

diagrammatically as in Figure 3.



PC running GRMON **Protoboard containing LEON 3FT processor**

Figure 3 – Test Setup

ADA codes get executed in the same environment. The compiler and debugger actions are carried out using the following set of commands. Command for Target initialization to establish connectivity between GRMON and the processor is mentioned below.

grmon –gdb

The programmer is expected to obtain the dump file using the command:

leon3-elf-objdump -d filename | tee filename.dump

Attaching to GDB and debugging through the GDB protocol and code execution:

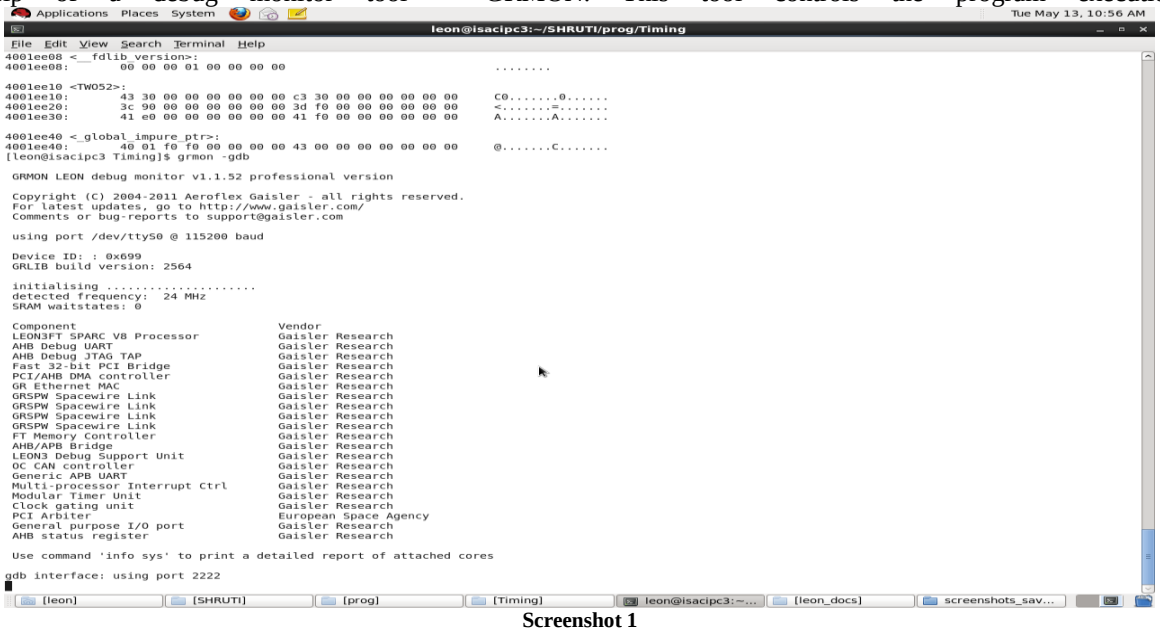
leon3-elf-gdb filename

target remote: portaddress,

load filename

continue

The dump file reveals the generated assembly level code from which one can calculate the total execution time of a specific computer program. The compile, link and fuse files were loaded on to the proto board with the help of a debug monitor tool – GRMON. This tool controls the program execution



Screenshot 1

on the proto board for LEON processors. GRMON communicates with the LEON processor through a gateway section, the non-intrusive debug support unit (DSU) of the LEON system. The GRMON architecture shows the command layer as an integration of both Basic commands and GDB protocol. This facilitates the programmer with two ways to provide inputs to GRMON. User can either provide a set of GRMON commands from the terminal or remotely connect to the GNU debugger. Screenshot 1 depicts the target connectivity using GRMON.

VI. Results Of The Practical Benchmark Test Suite

We understand that the results of the entire set-up can be obtained when a related software code runs and executes fine on the hardware support established. The current space application certifies the workability of the protoboard after executing a predefined benchmark ADA code on it. The ADA code, in itself contains all vital space application logics. As discussed before, GRMON debug monitor supports the system by loading this ADA code on to the processor residing on the protoboard. Prior to the implementation of the 32-bit LEON 3FT processor, the same ADA code was even executed on the previously used 16-bit MA 31750 processor.

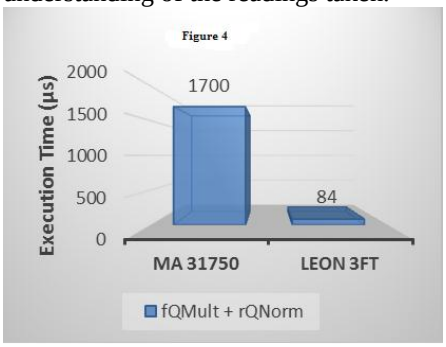
For the purpose of simplicity, logics covering quaternion multiplication and norm calculations were only executed instead of the entire program module and a main program calls these logics. The logics may also be

called within multiple looping constructs for the purpose of testing complex looping times. This main program is a simple toggle generating code, henceforth one can expect to see the output in the form of a periodic square wave on the CRO. Either the positive pulse width (Ton) or the negative pulse width (Toff) is the execution time .The same logics were executed on the new test bed to obtain comparative readings. With an objective to tabulate a fine set of readings for many more computational logics that may possibly be used in the spacecraft application, we intend to propose to execute each of the logics as separate entities and note the execution time and the possible number of instruction cycles consumed by each of the functions when executed on the LEON 3FT processor. The tabulation of readings is displayed in Table 1.

Table 1

Function	Description	Execution Time (in µs)	Number of cycles (24 MHz clock)
fQMult	Quaternion Multiplication	**15.2**	**≈ 365**
fLSqrt	Square root of number of type Quaternion	**6.8**	**≈ 164**
rQNorm	Norm value of number of type Quaternion	**19.5**	**≈ 468**
bopr1	Bit Operations	**2.35**	**≈ 57**
rMem_Cp1	Copy source and destination address	**2.27**	**≈ 55**
rRead_In_Ports	Read data from multiple input ports with additional data handling/manipulation logic	**2.31**	**≈ 56**
Arith	Basic mathematical operations on arithmetic values	**5.2**	**≈ 125**
flt_pnt_oper	Basic mathematical operations on floating-point values	**5.33**	**≈ 128**

The execution time then recorded was 1.7ms and now a tremendous dip of just 80µs, which indeed is a very significant performance elevation in terms of execution speed. The same has been depicted in the form of a bar graph, Figure 4 to enhance the visual understanding of the readings taken.



VII. Conclusion & Future Work

In this paper, the design and implementation of CPU for spacecraft applications using a 32-bit processor is presented. It is basically an enhancement over the existing design which implements a 16-bit processor for operation. The superseding processor indeed advertises an enhancement in terms of execution speed, operation in HiRel environments, fault tolerance, multiple options for external communication, power and timing constraints, size and hence weight and many more. But, the scope of this paper mainly revolves around the execution speed. The successful workability of the protoboard bearing our design is verified by the benchmark suite. A remarkable leap in execution speed has been achieved with the LEON 3FT compatible superior processor. LEON 3FT processors have already joined the league of super processors. The reduction in chip size and a 53 MIPS throughput via 66 MHz base clock frequency (as of LEON 3FT) as against a 1.5 MIPS throughput via a 12 MHz base clock frequency (as of MA 31750) are also added advantages of using the 32-bit processor. Nevertheless, few of the negativities of LEON 3FT have been brought to light and worked upon for betterment in LEON4. Efforts have been made to overcome some of the limitations of this processor in the next version of LEON namely LEON4 processor. LEON4 improvements [7] over the LEON3 processor include Branch Prediction, 64-bit pipeline with single cycle load/store and 128-bit wide L1 cache. Due to the 128-bit wider AHB bus and single cycle load and store instructions, there is a relatively 4x performance increase in terms of cache line fills.

Most on-board software is subject to real-time deadlines, so that the important figure is not (only) the average execution time, but the WCET or, in practice, a trustworthy upper bound or estimate of the WCET. But, the multiple-stage pipeline architecture, independent data and instruction caches and a memory management unit for the shared memory make a timing analysis difficult. Therefore, one can possibly go ahead with the WCET analysis of the current application to fulfil real-time deadlines. On the other hand, the LEON 3FT

features like Fault tolerance, 7-stage Pipeline, Caching, Interrupt Handling and Memory Interfacing can be explored and demonstrated with small code snippets. A detailed analysis of the above features will definitely highlight the superiority of the 32-bit processor we have implemented in this application.

References

[1] LEON3FT SPARC V8 Micro Processor, Functional manual, August 2010.
[2] Considerations on the LEON cache effects on the timing analysis of on-board applications - G. Bernat, A. Colin, J. Esteves, G. Garcia, Moreno, N. Holsti, T. Vardanega and M. Hernek
[3] "Fast and efficient cache behaviour prediction for real-time systems", by C. Ferdinand and R. Wilhelm. Real-Time Systems 17, 131–181, 1999.
[4] aiT Worst-Case Execution Time Analyzers. http://www.absint.com/ait/, AbsInt Angewandte Informatik GmbH.
[5] "WCET analysis of probabilistic hard real-time systems", by G. Bernat, A. Colin and S.M. Petters. In Proceedings of the 23rd Real-\Time Systems Symposium RTSS 2002, Austin, Texas, USA, 279–288.
[6] RapiTime. http://www.rapitasystems.com/wcet.html, Rapita Systems Ltd. the SPARC V8 Architecture - Jiri Gaisler, IEEE, 2002
[7] Next Generation Multipurpose Microprocessor – Jan Andersson, Jiri Gaisler and Roland Weigand.