Exploratory Study about the Use of New Reconfigurable FPGAs in Space

Rafal Graczyk
Space Research Center of Polish Academy of Sciences, Astri Polska
rgraczyk@cbk.waw.pl

Marcin Stolarski
Space Research Center of Polish Academy of Sciences, Astri Polska
mstolars@cbk.waw.pl

Patrick Cormery
EADS Astrium
patrick.cormery@astrium.eads.net

Abstract

The paper will present an exploratory study about the potential use of reconfigurable FPGAs in the context of Fault Tolerance Computers and digital systems for space systems. It will provide an analysis of potential applications of these new techniques, starting from classical architecture principles. The purpose of this analysis is to evaluate the feasibility of these approaches, to show the potential benefits for space systems and identify critical points and difficulties to be solved towards an operational deployment. After a theoretical overview of the architecture solutions trade-off using the reconfiguration principles, representative industrial case studies derived from current launchers and space vehicle applications will be presented.

1. Introduction

Avionics solutions based on various level of Equipment redundancy, to comply with the safety criticality or availability requirements of the whole the system, have some drawbacks. Redundant hardware systems are in contradiction with the constraints of space system on mass, power and volume whereas today's "System On Chip" technology trend is bringing new perspectives for highly integrated avionics architectures. Furthermore, for the execution of the flight software (e.g. flight control, the vehicle and mission management functions) within the On-Board Computers, the use of space specific technologies has often been a very strong constraint for the implementation of on-board functionalities. For the launchers or other space systems, the On-Board Computer has traditionally been implemented in the form of a Printed Circuit Board (PCB), built around a space specific microprocessor chip, with constraining limitation of processing power. The level of performances offered by high integrated circuits can give potentialities for the implementation of complex functions required for future space missions, as well as reconfiguration mechanisms. However, robust fault detection isolation and recovery must be demonstrated with respect to these new approaches.

The presentation will focus on the application of these concepts to a Fault Tolerant, On-Board Computer set and will address the following aspects:

- Application of architecture principles (potential evolutions derived from current operational systems) to launchers and space vehicle case studies
- Fault detection and Isolation: feared events, error or malfunction detection in Processing Unit, Dealing with malfunctioning Processing Unit
- Recovery mechanisms: assessment of the reactivity at application level (in a representative condition of load of the computer set), Synchronizing Processing Unit state after repair process

2. System overview

The main objective of research system described in paper is to derive key characteristics of avionic systems chosen as a base of study, implement them as a models, implement fault detection layers, fault isolation measures and recovery mechanism, and test what levels of system availability and fault tolerance could be achieved. Test system architecture design takes into account various needs that are considered in preparatory phase, and can be summarized in three high level requirements. First, the dynamic reconfiguration capability of designed System-on-Chip (SoC), as a key point of undertaken study, is a very important hardware requirement. Second, the software platform, shall be compatible with current and further avionics developments. Third, the hardware platform, shall be large enough to host several instances (at least three) of software platform, fault detection management layer and external interfaces for communication with test environment. As a good practice, logic resources for test benches shall be kept as well. Four, test system shall utilize redundancy schemes

978-1-4577-0599-1/11/$26.00 ©2011 IEEE

to raise reliability and availability and to bear a significant resemblance to existing avionics systems. The use of FPGA devices as a base for new fault tolerant space borne computers has two main reasons. First, it is much more effective to delegate complex computational tasks on reconfigurable logic devices than most processors, especially when parallelism or pipelining could be used. Therefore, more functionality could be placed in one system. Processing power/ Watt coefficient is also better for well designed FPGA systems than processors. Second, Xilinx FPGAs come in different qualification grades and can be, as its heritage shows, a base for wide range of avionic system applications - from low-cost microspace approach to very demanding, classical high reliability space missions. It is worth mentioning that using FPGA devices reduces impact of component obsolescence in systems that are going to be manufactured for around two decades or more [1].

Avionic systems that are used as a reference and are modeled are Ariane 5 OBCs (On-Board Computers) and ATV (Autonomous Transfer Vehicle) DPS (Data Processing System). Obviously, due to differences between those systems there is definitely a need to create two basic models, and integrate them on separate units of test hardware. Typical availability figure, related to described systems is a maximum 80-100ms of single downtime for a mission duration of around 2-3 hours of A5 to several weeks for ATV.

A5 OBCs are different than ATV DPS. The first is a very high availability system with tight (fast) control loops, exposed to many asynchronous events . The latter, is less time constrained, but requires higher processing powers, and has more levels of redundancy due to, possibly, life threatening results of permanent guidance failure during rendezvous phase of approach to International Space Station.

As written before, avionics models consists of functional modules, which relate to high level functions in terms of processing power needs (i. e. instruction count) and procedure triggers. Functional modules are interconnected with data flows which model the real data exchange in terms of volume and frequency. Functions and data flows are, at least, constrained by maximum allowed execution time and acceptable data loss.

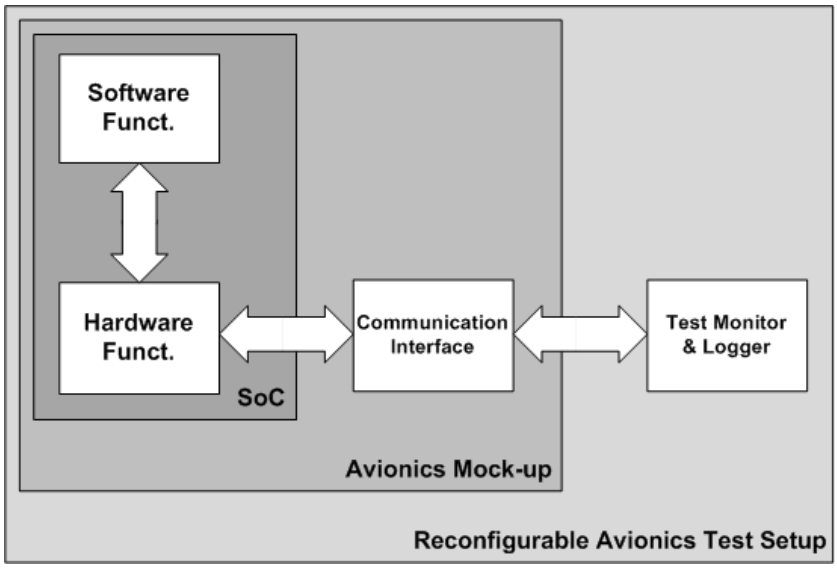

Figure 1 Reconfigurable Avionics Test Setup

Functional modules are built in software, where data flows both in software in hardware. Whole system is fed with input vectors and external triggers by external interfaces which provide communication and system management links to test monitor computer (Figure 1). A first step of the study to be presented (on-going on the first half of 2011), is to review the avionics architecture solutions based on the use of reconfigurable FPGAs. Within these architectures, a basic building block implementing a fault tolerant processing unit (regarding especially the space environment radiation effects) will be defined and implemented on a mock-up for experimentation.

Analysis of key requirement and constrains led to baseline system design. System-on-Chip is implemented in Virtex5 FPGA, understood later as a hardware platform. Virtex 5 itself is among three other families of Xilinx FPGA that support dynamic, partial reconfiguration and has development tools that really work, which might not be case for other hardware vendors. Virtex5, additionally, comes in qualified military and space grade components which is also of value. The exact devices used in test are (smaller) V5FX70T and (larger) V5FX130T (in standard development and in-house custom boards) and for the rest of a study only non space qualified devices will be used. The radiation tolerant versions are taken into account for future developments, perhaps closer to implementation in real space system, although restrictions like ITAR have to be seriously considered as important factor driving system design.

Software platform is based on LEON3 processor, standard, GPL version. As LEON3 is SPARC derivative, compatibility requirement is fulfilled as both new A5 OBCs and ATV DPS are based on ERC32 processor which follows SPARC V7 specification. LEON3 in Virtex5 can be clocked up to 150 MHz which shall provide necessary processing power for functional modules, with having large margin for maintenance and self-control or self-repair activities [2]. Bear in mind that 150 MHz on LEON3 yields at least 6 times more processing power that ERC32 solutions on A5 or ATV - this fact alone signifies that all timing and data restrictions for mock-up system shall be easy to meet. As for logic resources needs, LEON3 in standard configuration requires around 10% of logic of V5FX70T. Assuming that with fault management and recovery layers, LEON3 needs 15 % of these logic resources, there is still a lot of spare logic and routes when triple-redundant system is implemented (around 45% of logic). For FX130T stats are even better [3].

System architecture is build around LEON3 (CPU unit) together with IO devices and other peripherals (AUX unit) which forms an Processing Element (PE), then multiplied in several instances to build redundancy. AUX provides interface to system bus and to External Interface (EXT I/F). All interconnects between CPU and AUX, and among PEs are secured by Fault Detection and Isolation System (FADIS). Processing Element (or its part) recovery is managed by Reconfiguration and Synchronization Controller (RSC). PEs are put in Triple Module Redundancy configuration which is classic, well described and very effective hot redundancy scheme. In systems requiring even higher order of availability, Quadruple Module Redundancy (QMR) is considered as an option. QMR protects system in case of consecutive failures in two PEs as long as their occurrence happens in larger time span than recovery time of one PE.

The TMR (or QMR) scheme is implemented on module

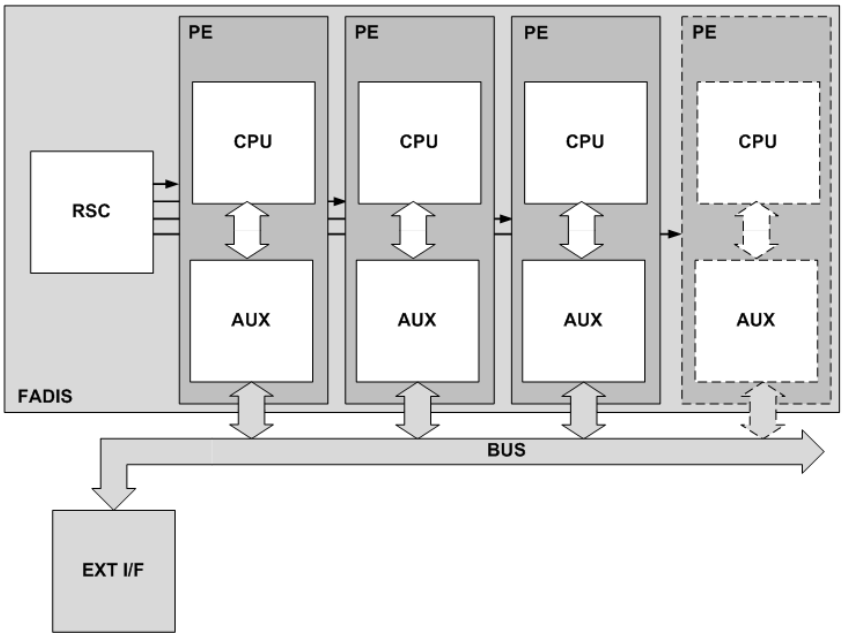

Figure 2 Proposed TMR (QMR) system architecture

not on register level. As functional units modeled in this study are complex and utilize vast amount of logic resources, the presumptive effectiveness of register level approach is superseded by technical feasibility of implementation of module level redundancy [4, 5]. Architecture is shown for clarity on Figure 2.

3. Dynamic reconfiguration

Dynamic reconfiguration (or dynamic partial reconfiguration) of FPGA is not a new feature. Technically, it has been present in some devices for last ten years (most commonly in SRAM FPGA, by Xilinx and Atmel). Nevertheless it didn't spread among logic design community due to very weak tools supporting hardware capabilities. It has changed about year ago, when Xilinx introduced new design flow in its tool chain, aware of dynamic and partial reconfiguration capabilities and supported it actively.

Dynamic partial reconfiguration (DPR) is a way to change a part of logic circuit implemented in FPGA while other parts of circuit operate actively and undisturbed. The intuitive benefit of DPR is possibility to fit a larger design into smaller FPGA, and change functionality over time according to user or processing needs. Such approach saves power and space, and, in the same time, extends design flexibility and, perhaps, life time.

Other, more complex benefit of DPR is possibility to repair a malfunctioning design. In case a FPGA configuration memory suffers from Single Event Events (SEEs) - upsets (SEU) or other functional interrupts (SEFI) – both in soft (temporary) and firm (destructive to logic fabric) types.

In case of configuration memory soft error, DPR allows to rewrite chosen part of configuration memory and therefore remove soft error. In case of firm configuration memory error – possible countermeasure is to physically move logic modules from destroyed location to other location on chip – or in other words – disable malfunctioning location and partially configure spare resources with contents from disabled region.

System-on-Chip utilizing DPR capabilities must be divided into static part and a various number of reconfigurable regions (slots) which are prepared for given reconfigurable modules. DPR process can be of external or internal origin. In both cases, there must be dedicated control unit a reconfiguration manager, feeding the configuration memory with partial bit stream. External origin of DPR process mean that reconfiguration manager resides off-chip, and accesses FPGA through standard means (e. g. JTAG). Internal origin needs reconfiguration manager to be placed on-chip and accesses configuration memory via ICAP (Internal Configuration Access Port). In latter one, FPGA device is dependent only on storage device containing partial bit streams.

Obviously DPR has some drawbacks. One of the most important is the lack of global design optimization. Optimization is done separately in static part and in each of reconfigurable modules.

It worth mentioning that DPR process itself is also prone to errors and some additional countermeasures might be necessary to raise reconfiguration reliability (e. g. verification, checksums).

Taking into account architectural considerations it makes sense to put CPU and AUX of each Processing Element into reconfigurable regions. Such decision limits effort (and time) needed for recovery. If CPU fails, then AUX remains unchanged, and CPU is reconfigured and synchronized. If AUX fails, then AUX has to be reconfigured, and CPU needs only to be synchronized with rest of system (it was separated from system bus during AUX reconfiguration) [6,7,8].

4. Fault Detection and Isolation System

Fault Detection and Isolation System is a crucial part of presented technology. It is supervisory layer that detects faults and raises dedicated countermeasures. FADIS first objective is to detect fault occurrence.

Faults, understood mainly as radiation induced errors or SEE, can affect different parts of FPGA device. Both configuration memory and logic fabric. There are known and effective solutions as Triple Module Redundancy (TMR), but implementing them on gates or flip-flop levels of each instance of PE might be extremely hard, not mentioning heavy effect on device performance. Moreover, bearing in mind that in given FPGA around 75-90% (depending on device) of transistors are used for routing resources rather than logic, it is much more likely that SEE will be a functional interrupt rather that a flip-flop switch. As a conclusion of statements above, it is reasonable to look for signs of fault where the function or operations results occur. It means that outputs, status registers, flag registers of given PE, shall be constantly monitored by FADIS and compared against results from other Processing Elements [9, 10].

Memory elements of CPU and AUX are implemented with special care, protected by Error Correction Coding (ECC) mechanisms (e. g. Hamming algorithm). It is relatively low cost (in terms of system performance and resources utilization) method of assuring very high reliability to memory elements, especially when compared to TMR.

FADIS second objective is to isolate fault. It is done by halting faulty unit and turning down all the related out buffers. Next step, is to start reconfiguration processes

followed by synchronization process. Whole system is supposed to remain operational, but performance (due to synchronization needs) might be affected. FADIS itself has to be also TMR protected.

Reconfiguration and Synchronization Controller supervises all signals and processes taking place in FADIS layer. RSC, as crucial unit for functioning of whole system, has to be tripled as well to prevent single point of failure. RSC has to scrub configuration memory of FADIS layer and RSC itself to possibly cover all SEFIs in functional units responsible for detection of system / module level faults . SEU in JTAG or ICAP circuits cannot be mitigated and in case of their permanency a reconfiguration of whole device is necessary.

There is a special case of destructive events that may cause a permanent failure to part of configuration memory or logic resource. For such case there is a technical possibility to move a functional block or larger module into different physical location. But so far there is no need for further exploration because of two facts. First, statistics show that it is extremely unlikely for a qualified Xilinx device to suffer from destructive events [11, 12]. Second, technology requirements impose a need to prepare a dedicated reconfigurable region for units to be moved there from other parts of FPGA fabric and to have configuration bitstreams for given functions prepared and stored.

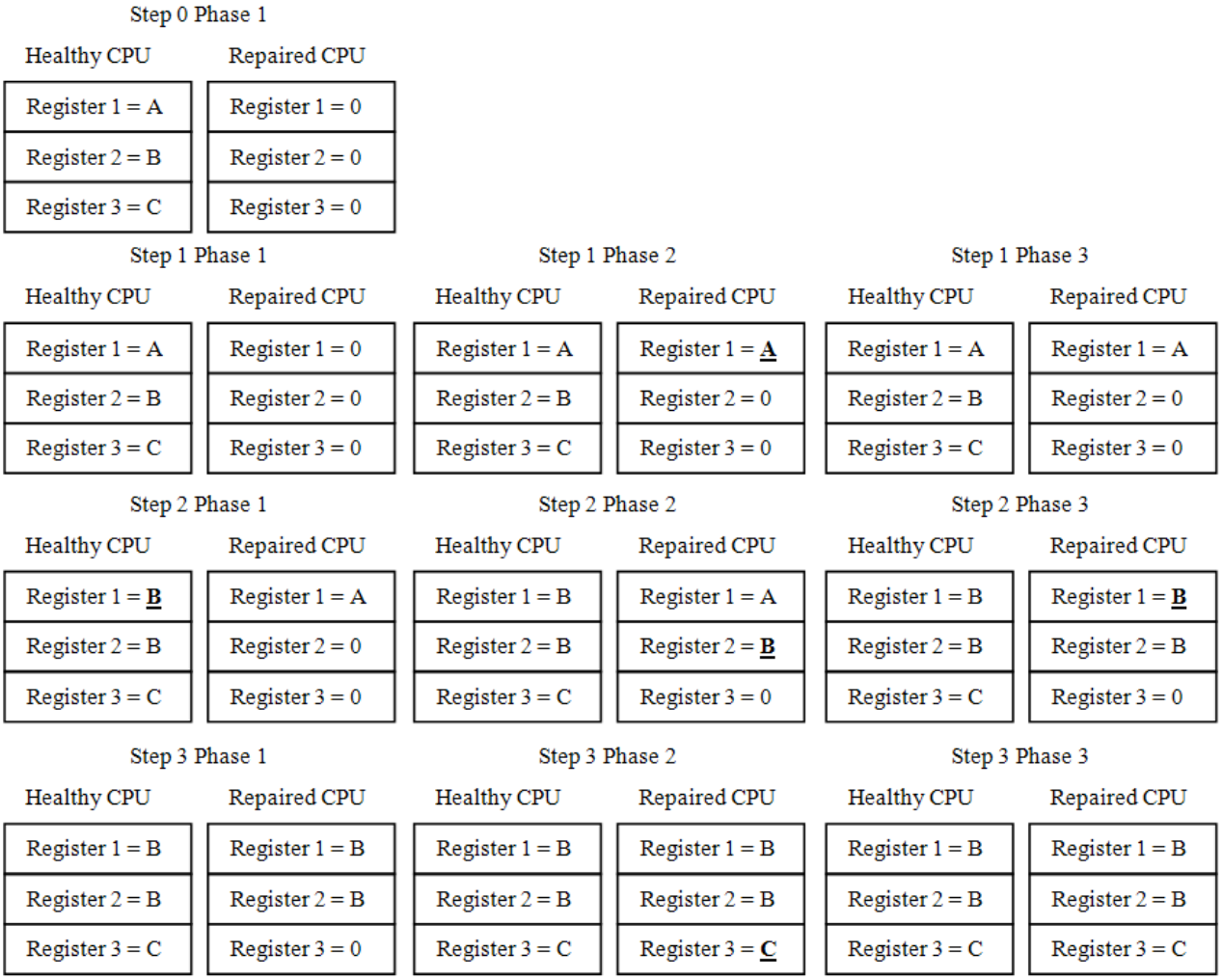

Figure 3 Synchronization algorithm example

5. Synchronization

After reconfiguration, the newly placed device is completely reset and has to be synchronized with rest of redundant devices. There are at least four synchronization strategies which are feasible for implementation in reconfigurable avionics test platform.
Easiest approach is to reset other redundant systems. This is fastest solution but renders whole system as not available for certain amount of time, not mentioning possibility of data and control loss. So this approach is suitable for low availability, low criticality systems.
Second approach assumes use of an operating system with process synchronization capability. Most of systems with preemptive multitasking technology have mechanisms for synchronization of a microprocessors and their cache. Every task preemption event (changing register and cache contents of every operating PE) could be also used to fill reconfigured PE with new, most up-to-date contents. Reconfigured PE will be halted till the first preemption, an in that time will decrease system reliability. This approach is very software or operating system dependent.
Third approach is to stop healthy units, copy register and cache contents from healthy ones to newly configured. This approach is relatively easy to implement, but, obviously, system will stop responding until registers are copied. It is much faster that whole system boot, but it might not be possible to afford absolute system halt.
Last proposed approach is sharing of system's processing performance to a normal activities processes and a synchronization process. For example, one third of a system processing power could be used for synchronization process and rest for normal activities. In such case, the three operation phases can be identified: 1 – normal operation phase, 2 – loop copy phase, 3 – changes annotation phase. So in first phase the healthy CPU executes normal operations code. In second phase (loop copy) the repaired CPU copied all the registers from healthy one. In third phase, if in normal operations phase the healthy CPU changed any of internal register or caches, then the repaired CPU has to annotate changes that took place in meanwhile. All phases are looped and are switched according to system clock. Example of how described algorithm works is shown on Figure 3.
The synchronization algorithm description:

- Step 0 (system state after repairing). The system use the CPU with three registers. The healthy CPU registers are filled with data: A, B, C. The repaired CPU has only zeros in the registers (after reset). During the next steps the flight software will execute instructions: NUL (do nothing), MOV 2 1 (copy register 2 to register 1), NUL.
- Step 1 Phase 1. The healthy CPU execute NUL instruction.
- Step 1 Phase 2. The repaired CPU copy register 1 from healthy CPU.
- Step 1 Phase 3. Nothing to do.
- Step 2 Phase 1. The healthy CPU execute MOV 2 1 instruction (copy register 2 to register 1).
- Step 2 Phase 2. The repaired CPU copies register 2 from healthy CPU.
- Step 2 Phase 3. The repaired CPU copies (changed in phase 1) register 1 from healthy CPU.
- Step 3 Phase 1. The healthy CPU execute NUL instruction.
- Step 3 Phase 2. The repaired CPU copies register 3 from healthy CPU.
- Step 3 Phase 3. Nothing to do.
- After Step 3 Phase 3 the repaired CPU shall be fully synchronized (have the same contents as healthy CPU) with rest of system and could start executing normal mode software.

The strategy described in detail has largest utilization of system resources but it is least intrusive and most transparent for the software (no pauses in executing software, only temporary, predictable performance loss). This properties are necessary for real time, fault hardened systems like avionics. Synchronization process described in this section can be considered as a progress, when related to older techniques, like lockstep. The latter one is based on executing the same application code on two identical microprocessors, first is normally connected to the system components like RAM and IO. Second can only read data but can't write data to system. Outputs of two microprocessors are connected to comparison system which raises error flag after detection of difference in microprocessor outputs. TMR with synchronization is a step further – it utilizes more functional units making them resistant to one error and allowing detection of two faults [13, 14]. With dynamic FPGA reconfiguration incorporated in system, single faults can be continuously repaired, rendering system reliability almost constant in time (with short periods of lowered reliability during fault detection and recovery).

6. Summary

Exploratory study of use of new reconfigurable FPGAs in space is a joint project of EADS Astrium, Astri Polska and Space Research Center of Polish Academy of Sciences (SRC PAS). Project main objective is to build avionics mock-ups utilizing mechanisms for raising systems reliability and availability, hot redundancy,

dynamic partial reconfiguration and processor resynchronization.

Dynamic reconfiguration capability although present in FPGA devices for at least a decade has not been utilized in satisfactory level so far, mainly, due to lack of real support from tools. As this situation started to change recently (by Xilinx efforts) it is right moment to start the research and development works related to Dynamic Reconfiguration used for raising system reliability and availability in avionics.

Presented paper describes first assumptions and requirements derived in RecAv project hosted in Astri and SRC PAS.

Analysis of Ariane 5 and ATV avionics has been performed, deriving key functionalities, data flows and time (synchronous and asynchronous) constraints for construction of reference systems. In the same time, study of dynamic reconfiguration and synchronization techniques has been started. As a result, first architectures, fault mitigation and processing elements synchronization mechanisms has been outlined.

7. References

[1] R. C. Ferguson, R. Tate USE OF FIELD PROGRAMMABLE GATE ARRAY TECHNOLOGY IN FUTURE: SPACE AVIONICS, NASA Johnson Space Center, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.g

[2] F. Koebel, J. -F. Coldefy, SCOC3: a space computer on chip, Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2010

[3] SEU Strategies for Virtex-5 Devices, www.xilinx.com

[4] S. Thompson, A. Mycroft, G. Brat, A. Venet, Automatic In-Flight Repair of FPGA Cosmic Ray Damage, Proc. 1st Disruption in Space Symposium, 2005

[5] S. Habermann, R. Kothe, T. Vierhaus, Built-in Self Repair by Reconfiguration of FPGAs, 12th IEEE International On-Line Testing Symposium, 2006

[6] M.Platzner, J. Teich, N. When, Dynamically Reconfigurable Systems, Springer, 2010

[7] D. Candiloro, Management and Analysis of Bitstreams Generators for Xilinx FPGAs, Verlag Dr. Mueller, 2009

[8] K. Compton, S. Hauck, Reconfigurable Computing: A Survey of Systems and Software, ACM Computing Surveys Vol. 34, 2002

[9] C. Carmichael, E. Fueller, F. Fabula, F. De Lima, Proton Testing of SEU Mitigation Methods for the Virtex FPGA, MAPLD Conference, 2001

[10] C. Yui, G. Swift, C. Carmichael, R. Koga, J. George, "SEU Mitigation Testing of Xilinx Virtex II FPGAs," data workshop paper, Nuclear and Space Radiation Conference (NSREC), 2003

[11] C. Carmichael, E. Fueller, P. Blain, M. Caffrey, SEU Mitigation Techniques for Virtex FPGAs in Space Applications, http://china.xilinx.com/esp/mil_aero/collateral/presentations/SEU_mitigation_technique.pdf

[12] Xilinx Radiation Effects and Mitigation Overview http://www.xilinx.com/esp/mil_aero/collateral/presentations/radiation_effects.pdf

[13] IBM, PowerPC 750GX Lockstep Facility, www.ibm.com

[14] Harn Hua Ng, PPC405 Lockstep System on ML310, www.xilinx.com