Journal of Engineering Science and Technology
Vol. 13, No. 1 (2018) 058 - 068

AN EXHAUSTIVE ANALYSIS OF SEU EFFECTS IN THE SRAM MEMORY OF SOFT PROCESSOR

AFEF KCHAOU[1,2,*], WAJIH EL HADJ YOUSSEF[2], RAOUL VELAZCO[3], RACHED TOURKI[2]

[1]University of Tunis El Manar, Faculty of Sciences of Tunis, Tunisia
[2]Electronics and Micro-Electronic Laboratory (LEME), Faculty of Sciences of Monastir, Tunisia
[3]University of Grenoble-Alpes, TIMA Labs, INPG, Grenoble, France
*Corresponding Author: kchaouafef@gmail.com

Abstract

The Embedded system design is characterized by its daily complexity. It integrates a hardware and software parts together on a common platform. These parts may be defective by a spurious signal, subsequently found to be two types of errors. The software and hardware errors can attack the embedded system. In this paper an exhaustive analysis of the effects of Single Event Upset into the Static Random Access Memory occupied area of Aeroflex Gaisler LEON3 processor is presented. It is a soft core pipeline processor that is part of the GRLIB IP library based on Scalable Processor Architecture, SPARC V8, implemented in Virtex-5 FPGA. A new software methodology allowing fault injection is explored and illustrated in order to classify the defective behaviours while executing several benchmarks. This investigation is done by an exhaustive fault injection campaign (More than 200000 transient faults) into SRAM memory of LEON3 considered as a processor. The proposed method makes error rate predictions more accurate compared to other techniques.

Keywords: LEON3, SEU, Software fault injection, Benchmark, FPGA.

1. Introduction

The complexity of embedded systems is constantly increasing due to the tight constraints on area use, size, power consumption and performance. Another constraint is the time to-market deadlines. An embedded system integrates both hardware and software parts on a common platform to perform a specific application. It is characterized by their energy, real-time, security and performance [1]. Using soft-core processors an embedded system can help the designer. Soft-core processors are flexible, simply customized and have an independent technology

Abbreviations

AES	Advanced Encryption Standard
AHB	Advanced High-speed Bus
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
FIPS	Federal Information Processing Standards
FFT	Fast Fourier Transform
MBU	Multiple Bit Upset
SEU	Single Bit Upset
SET	Single Event Transient
SPARC	Scalable Processor ARChitecture
SRAM	Static Random Access Memory
TCL	Tool Command Language

that can be synthesized for any given target Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Arrays (FPGA) technology. The architecture of soft-core processors is described at a higher abstraction level using an HDL [2].

There are many soft-core processors like NIOS II, Altera, Pico-Blaze, LEON3, etc. Each core has different performance characteristics and features that are suitable for specific applications. LEON3 processor is a highly configurable open source code developed by Aeroflex Gaisler Research, designed for embedded systems [3]. It supports power-down mode and clock gating with robust and fully synchronous single-edge clock design [4]. LEON3 soft core processor can be implemented on ASIC and FPGA thanks to its best performance and most configurable processor. Many studies show that LEON3 is the most efficient processor by comparison with other processors like the Xilinx Micro Blaze soft-core processor and hard-core on-chip PowerPC processor [5]. Faults injection mechanism is a method to understand the behaviour of a microprocessor under soft errors. There are many methods of faults injection that can be implemented on both Software and Hardware. Soft errors are transient faults that can modify values stored in memory elements operation under radiation environment in space or Earth [6]. The modification in memory of our microprocessor can be done at the state of memory bit or flip-flop, this is named a soft error [7]. Such faults can modify the results of executing application or continue executing the application in a loop without stopping [8]. In the prototype-based fault injection, we inject faults into the system to:

- Study the processor behaviour in the presence of fault injection,
- Evaluate the effectiveness of fault injection mechanisms on the system's dependability, and indicating the advantages and disadvantages of the method.

The principal idea is to analyse behaviour of multicore LEON3 in the presence of fault injection. In this paper, a validation of fault injection in one core of LEON3 is carried out.

2. Literature Review

Several works used a fault injection in LEON2/3 processor. Cho et al. [9] evaluated the effects of single-bit errors at the register and memory locations by

applying a variety of high level error injection techniques, the results obtained show that the technique used is highly inaccurate when its compared to flip-flop error injection techniques. Other types of fault injection, multiple faults, and security evaluation are brought up in [10]. Aguirre et al. [11] presented the effects of SEUs and the time needed for error recovery in LEON2 processor. Authors presented a test procedure applied to three different, protected and unprotected versions of the well-known LEON2 processor. Houssany et al. [12] proposed a new methodology of emulation based fault injection to evaluate the cache memories sensitivity for a given application, and to calculate a more accurate SER. The methodology, based on monitoring the memory accesses, is applied to the LEON3 soft-core with several benchmarks. They show that their proposed tool precise that all the addresses are sensitive to SEU fault injection.

Da Silva and Sanchez [13] presented a virtual platform with fault injection where the faults are injected in the RAM memory. The results obtained show that the zone storing the instruction code is the most critical one. For most existing processors, SEU can be injected by software means concurrently with the execution of a program at a desired instant and dedicated sequence of instructions. Typically, the SEU fault injection mechanism consists in injecting the fault at a general purpose register or at a directly addressable internal or external memory location.

In this paper, a fault injection by emulation method, so-called SEUs, is used to emulate the consequence of errors on the internal memory of LEON3 processor. LEON3 is set to execute several benchmarks. The benchmarks are intended to provide correct results even if a failure occurs in the middle of the execution. In each address of the internal memory, the exclusive-or logic operation is affected with each byte during a single clock cycle. It consists in changing the current state of a flip-flop in a particular clock cycle. A script program was created based on Tool Command Language (TCL) to modify the internal memory of LEON3.

3. Fault Injection and Micro Architecture of LEON3 Processor

3.1. Fault injection

There are two types of fault injection, a hardware and software, it depends on the type of faults and the effort required creating them. With a hardware injector, you can control the location of the fault. But in the case of data corruption, a software injector might suffice. Other faults like bit-flips (Single Event Upsets) in memory cells can be injected by both methods. Fault injection permits the evaluation of how designs perform inside injection environments. They evaluate the action of errors using the bit-flip model: SEUs are emulated as the change in the value of a circuit function, single event transients (SETs) are emulated to change the register content, and multi bit upsets (MBUs) are emulated to change the content of paired registers dependent on the layout of the circuit. These cases are called SEEs because they are a result of a single particle impact against a device [14]. Now, researchers are taking more interest in developing software implemented fault injection tools because it doesn't require expensive hardware. Software methods are convenient for directly producing changes at the software-state level like memory and register. This method is less expensive but it also incurs a higher perturbation overhead because it executes software on the target system [15].

3.2. Micro architecture of LEON3 processor

Aeroflex Gaisler research has made freely available a collection of open source IP cores and a design configuration environment for developing the LEON3 based FPGA designs, collectively referred to as GRLIB IP library [16]. The block diagram of LEON3 is shown in Fig. 1.

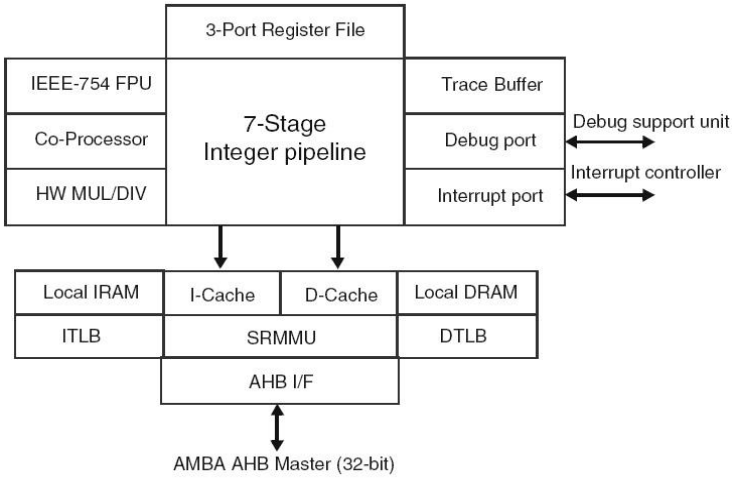


Fig. 1. Block Diagram of LEON3 [16].

The considered LEON3 processor is an open source core synthesizable VHDL model of 32-bit processor based on the SPARC-V8 architecture with support for multiprocessing configurations. The processor is used to generate a smaller or faster implementation, which made its popularity in the space community. All hardware configurations for LEON3 processor were built using the GRLIB implementation tool Version provided by Gaisler and a Cygwin tool, which provides LEON3 template designs for many FPGA boards. The model is highly configurable and uses the AMBA 2.0 Advanced High-Speed Bus (AHB) interfacing with other IP core. The template design is represented in Fig. 2.

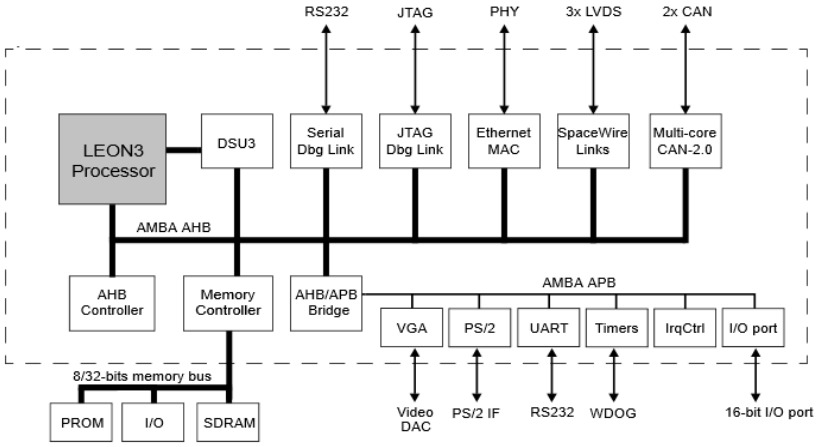


Fig. 2. LEON3 Template Design [17].

The connection design is based on the AMBA AHB, to which LEON3 processor and other high-bandwidth devices are connected. Access to the external memory is made through a combined PROM/IO/SRAM/SDRAM memory controller. The on chip peripheral devices include three Space Wire links, Ethernet 10/100 Mbit MAC, Dual CAN-2.0 interface, serial and JTAG debug interfaces, two UARTs, Interrupt Controller, Timers and an I/O port. A debug monitor, GRMON, provides a quick hardware and software validation. LEON3 application can be loaded or debugged using a command line interface or a graphical user interface by GRMON.

4. Methodology of SEU fault injection

In order to get the quality of protection against SEU sensitivity is usually assessed by means of dynamic fault injection test. Faults are injected into a specific target in the circuit at a given time and the behaviour of the faulty circuit is then recorded [11]. The consequence of the injected SEU may be classified as follows [18]: Silent fault when the injected fault does not have any consequence on the program results. This type of fault touches the register or data not used or not yet used by the processor. Another possibility is that the algorithm provides a correct result despite the fault, or the results of the program are not the expected ones and a Timeout result is obtained when the execution time of the benchmark exceeds the limit execution time. The proposed SEU strategy consists in injecting SEU faults in the SRAM occupied area when running a benchmark application under test. Before starting a fault-injection, it is necessary to determine the first address of loading the data of the selected application and the contents of the memory address map. To inject a fault, a single bit modification is done at each run. In all these cases, one SEU per execution cycle is injected. The proposed fault-injection mechanism is depicted in Fig. 3.

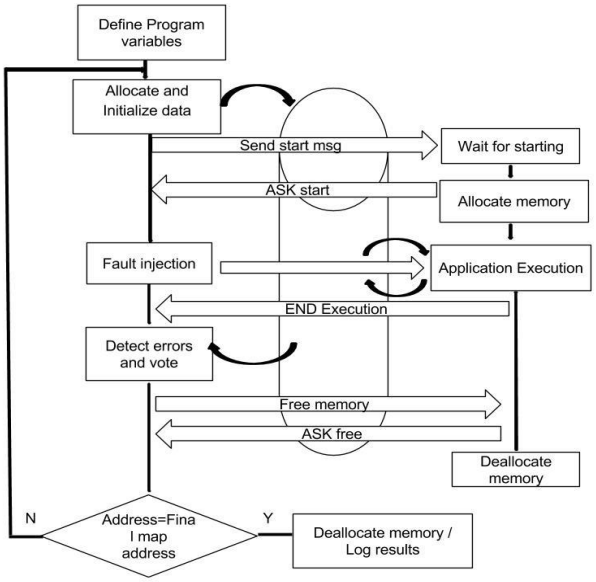


Fig. 3. The proposed software fault injection strategy.

To implement a fault injector, two scripts will be used, the first one is employed to allocate the memory address map when executing the target application and the second one is used for throwing the fault injection. The first step is the definition of the first address of data, this is done by a process created by a script to execute the benchmark to allocate the data memory and then a second script will be used to inject a fault.

At the instant of fault injection, the fault injector reads the information about the memory address map (allocates memory) provided by the first script and then it injects the fault using a specific function. For each address of SRAM memory there are 128 bits, the principal idea of the SEU fault injection is to affect for each bit an XOR logic operation during one clock cycle. After a memory bit modification is done, the execution of the benchmark is accomplished, the fault injector sends END EXEC signals to analyse the consequences of the fault injection. Since the fault injection cycle is reached, the fault injector masks the selected bit and creates a set of erroneous inputs. In the same way, SEU Fault-injection strategy is applied for all selected 128 bit memory address content. Hence, SEU is applied for the allocated memory address map to be tested. If the fixed final memory address map is reached, the memory will not be allocated and the result will be saved in a log file, if not, the memory address contents will be allocated again.

5. Results and Discussion

5.1. Benchmark programs

Three benchmarks were chosen in this fault injection experiments to test the variation in sensitivity on executed instructions: Advanced Encryption Standard, AES (59.23 Kb), Matrix Multiplication (62.59 Kb) and Fast Fourier Transform, FFT (62.28 Kb). Each benchmark executes out of the internal SRAM memories and uses only internal memory during their execution.

- *Advanced Encryption Standard*: The AES algorithm was selected as a data encryption standard by the National Institute of Standards and technology (NIST) in 1997 [16]. In our case, we considered the AES 128-bit benchmark (800000 instructions, total execution time: 12 ms) a famous and strong encryption algorithm which has several advantages in data ciphering, security applications, simplicity of implementation and low memory requirements, but it suffers from some drawbacks like high computations, pattern in ciphered images, and hardware requirement [19].
- •*A standard Matrix Multiplication* (MulMatrix): this algorithm multiplies two integer matrices and stores the result at a specified memory location. It is widely used for solving scientific problems. The matrix multiplication requires a large data processing with only a few loops and therefore uses mostly the data-path from the microprocessor, being ideal to verify the coverage of the techniques in detecting errors affecting data. In this work 30x30 input matrices were used in order to maximize the memory occupation to increase the probabilities of observing SEU consequences.
- •*Fast Fourier transform* (FFT): FFT is one of the most widely used operations in digital signal processing algorithms. Fourier analysis has many

scientific applications in physics, signal processing, imaging, probability theory, statistics, cryptography, numerical analysis, acoustics, geometry and other areas [20].

The programs are coded in C and have been compiled resorting to the GNU C/C++ GCC compiler, which is able to generate an executable software application by LEON3 [21].

5.2. Results of benchmark experiment

The emulation of SEU faults in data SRAM memory was done at each benchmark execution. The SRAM data of LEON3 loads the data in the register, this explains that the fault injection in the internal memory of LEON3 is faster than the injection in the register, but via this method, we can't modify the Program Counter and Stuck Pointer register. The SEUs were injected only in SRAM area. It is well-known that SEUs do not permanently damage a circuit, but they can cause faulty behaviours. In the worst case, the processor will have to be reset or reconfigured to overcome the effects of a soft error. The design is synthesized by using Xilinx ISE; the implementation of LEON3 is made by a CYGWIN tool works. During a compilation of executable file of benchmark program, to analyse the behaviour of LEON3, in each clock one error is sent to the internal memory. Obviously, SEUs do not permanently infect a circuit; however faulty circuit behaviour can be provoked. Worst case scenario, the processor should be reset or reconfigured to defeat the effects of a soft error [22]. The consequence of the injected SEU in the internal memory of LEON3 processor can be classified as follows:

- •False Result: the algorithm provides a false result when the errors are injected in the data SRAM.
- •Silent fault: the algorithm provides a correct result despite the fault.
- •Timeout: the application does not terminate normally. The workload execution is not completed after a predefined amount of time and the simulation is halted externally, which explains a power down mode of LEON3 processor or a DSU error in our case.
- •Stopped mode: the application terminates abnormally with error indication. The algorithm does not provide any result despite the fault; in this case it needs a reset trap of LEON3, memory access violation or invalid instruction.
- •No response: when the application does not produce any result.

Table 1 summarizes an analysis of the effects of SEU fault injection in SRAM occupied area of LEON3 processor.

When we increase the number of SEUs faults injection in the internal memory of LEON3, the error percentage changes by decreasing in some type of errors, but a new behaviour of LEON3 is created so that the execution duration exceeds a limit.

For a number of 62719 injected faults, we note that 18.735% of stopped mode is produced; the application is achieved abnormally with error indication. This percentage decreases with the augmentation of the number of errors. This analysis is done by injecting over 200000 errors into the internal memory of the

considered processor. The behaviour of LEON3 processor against the injected faults is reported and has shown that about 0.034% of faults are observed in simulation time. With this sort of error the results of the program are not the predicted ones. As expected too, 93.340 % of injected faults did not have any consequence on the AES program's results (silent-fault). A number of 5.186% of injected faults produced a stopped-mode, 0.0008% of timeouts errors and about 1.437% of no-response-mode where observed. The lower sensibility of the LEON3 processor can be explained for the reason that of no sufficient registers were used when variables needed to be duplicated. This can be interpreted by SEUs injected in variables used to initialize the AES's counters. This processor behaviour is noticed especially when the stack pointer and program counter were not aimed by the injected SEUs.

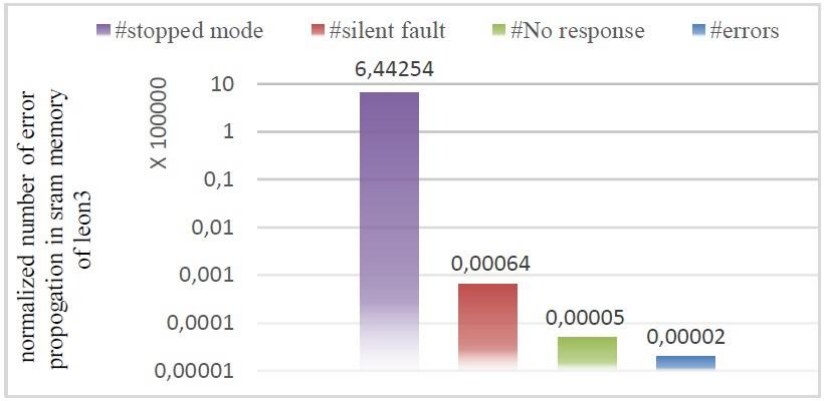
Table 1. Results of fault injection experiments in SRAM occupied area of LEON3 processor.

#injection	%error	%silent-fault	%Timeout	%stopped-mode	%no-response
17020	0.464	99.2	0.011	0.217	0.099
62719	0.125	75.943	0.003	18.735	5.191
128255	0.061	88.235	0.001	9.162	2.538
161023	0.049	90.629	0.001	7.297	2.022
226559	0.034	93.340	0.0008	5.186	1.437

To make LEON3 processor more sensible for SEUs faults, complex benchmarks will be used to perform a fault injection campaign. Two benchmarks were chosen to test the sensitivity of our processor: MulMatrix and FFT.

Figure 4 shows the distribution of the fault-injection consequences in the tested applications when targeting memory locations. After applying the proposed technique over the programs, a fault injection campaign on both was performed where 644422 faults were injected in MulMatrix and 85765 faults in FFT benchmarks. The analysis of the obtained results can be summarized as follows:

- •As would be expected in LEON3 processor configuration, fewer number of fault simulations generate wrong outputs (0.0003%) and no response mode (0.0007%) when injected in MulMatrix. No erroneous output neither no-response mode was observed.
- •0.0099% of fault injections in MulMatrix benchmark (resp.0.259% in FFT) finish without any effects.
- •99.988% of SEU fault injection produces stopped-mode errors when executing the MulMatrix program, this rate is about 4.047% for FFT algorithm.
- •As shown 95.692% of injected SEUs produced an integer unit error (IU in error mode) when FFT code is targeted and the program is halted. However no kind of error was noticed for MulMatrix. This error can be caused by a synchronous trap, in this case the IU enters into an error mode and remains in that mode until it is reset by an external logic. As shown in Fig. 4, LEON3 processor sensitivity strongly depends on the software used. The processor is more and more sensitive to SEUs when targeting complex benchmark.



(a) MulMatrix Benchmark.

(b) FFT Benchmark.

Fig. 4. Fault injection consequences when targeting memory locations.

6. Conclusions

In this work, we have estimated the sensitivity of SRAM memory of soft processor LEON3 in the presence of SEU fault injection by emulation.

This paper detailed the analysis of SEU propagation through the internal memory of LEON3 processor implemented in Virtex-5 FPGA and gives its behaviour. A new methodology for the SEU fault injection has been presented. Only one SEU was injected when executing a benchmark application under test and the SRAM occupied area was exhaustively explored.

The obtained results of fault analysing will be used in the future work to propose the fault-tolerant LEON3 processor. To implement an SEU-tolerant processor on a non-hardened semiconductor process, a variety of low-cost error-detection and correction techniques, such as TMR registers, on-chip EDAC, BCH, parity, pipeline restart, and forced cache miss, might also improve reliability and reduce area costs. While we focused on soft errors in this paper, future work must address other architecture multicore LEON3 and change the component that can be injected, like

registers. Other type of fault injection can be used in the future work like a Single Event Transient and a Multi Bits Upsets to inject faults in the logic gate or in multi registers at the same time.

References

1. Jiang, W.; Zhenlin, G.; Yue; M.; and Sang, N. (2013). Measurement based research on cryptographic algorithms for embedded real-time systems. *Journal of Systems Architecture*, 59(10), 1394-1404.
2. Jason, G.T.; Ian, D.L.A.; and Mohammed, A.S.K. (2006). Soft-core processors for embedded systems. *Proceedings of International Conference on Microelectronics ICM '06*, Dhahran, Saudi Arabia, 170- 173.
3. FIPS_PUB_197 (2001). Advanced encryption standard (AES). November 26, 2001, National Institute of Standards and Technology, U.S. Department of Commerce. Retrieved January 17, 2017, from http://csrc.nist.gov/ publications/ fips/ fips197/fips-197.pdf.
4. Hasamnis, M.; Chimankar, P.; and Limaye, S.S. (2012). Comparative analysis of LEON3 and NIOS II processor development environment: case study rijindael's encryption algorithm. *International Journal of Electronics and Computer Science Engineering*, 1(3), 1308-1314.
5. Ahmad, N.; and Rezaul Hasan, S.M. (2013). Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using novel XOR gate. *Integration the VLSI Journal*, 46(4), 333-344.
6. Casagrande, L.G.; and Kastensmidt, F.L. (2016). Soft error analysis in embedded software developed with & without operating system. *Proceedings of Latin-American Test Symposium LATS'16*, Foz do Iguacu, Brazil, 147- 152.
7. Harward, N.A.; Gardiner, M.R.; Hsiao, L.W.; and Wirthlin, M.J. (2015). Estimating soft processor soft error sensitivity through fault injection. *Proceedings of Annual International Symposium on Field-Programmable Custom Computing Machines FCCM'15*, Vancouver, British Canada, 143-150.
8. Chielle, E.; Rosa, F.; Rodrigues, G.S.; Tambara, L.A.; Kastensmidt, F.L.; Reis, R.; and Cuenca-Asensi, S. (2015). Reliability on ARM processors against soft errors by a purely software approach. *Proceedings of European Conference on Radiation and Its Effects on Components and Systems RADECS'15*, Moscow, Russia, 1-5.
9. Cho, H.; Mirkhani, S.; Cher, C.Y.; Abraham, J.A.; and Mitra, S. (2013). Quantitative evaluation of soft error injection techniques for robust system design. *Proceedings of ACM/EDAC/IEEE Design Automation Conference DAC'13*, ACM, Austin, TX, USA, 1-10.
10. Campagna, S.; Hussain, M.; and Violante, M. (2010). Hypervisor-based virtual hardware for fault tolerance in COTS processors targeting space applications. *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems DFT'10*, Kyoto, Japan, 44 - 51.
11. Aguirre, M.A.; Tombs, J.N.; Munoz, F.; Baena, V.; Guzmán, H.; Napoles, J.; Torralba, A.; Fernandez-Leon, A.; Tortosa-Lopez, F.; and Merodio, D. (2007). Selective protection analysis using a SEU emulator: testing protocol

and case study Over the Leon2 processor. *IEEE Transactions on Nuclear Science*, 54(4), 951-956.

12. Houssany, S.; Guibbaud, N.; Bougerol, A.; Leveugle, R.; Miller, F.; and Buard, N. (2011). Microprocessor soft error rate prediction based on cache memory analysis. *Proceedings of European Conference on Radiation and Its Effects on Components and Systems RADECS'11*, Sevilla, Spain, 412-419.
13. Da Silva, A.; and Sanchez, S. (2010). LEON3 ViP: a virtual platform with fault injection capabilities. *Proceedings of Euromicro Conference on Digital System Design: Architectures*, Methods and Tools DSD'10, Lille, France, 813-816.
14. Abbasitabar, H.; Zarandi, H.R.; and Salamat, R. (2012). Susceptibility analysis of LEON3 embedded processor against multiple event transients and upsets. *Proceedings of International Conference on Computational Science and Engineering ICCSE'12*, Nicosia, Cyprus, 548-553.
15. Natella, R.; Cotroneo, D.;Duraes, J.A.; and Madeira, H.S. (2013). On fault representativeness of software fault injection. *IEEE Transactions on Software Engineering*, 39(1), 80-96.
16. Daněk, M.; Kafka, L.; Kohout, L.; Sýkora, J.; and Bartosiński, R. (2013). UTLEON3: exploring fine-grain multi-threading in FPGAs-the LEON3 processor, *Springer-Verlag New York*, 9-14.
17. Gaisler, J.; and Isomaki, M. (2006). LEON3 GR-XC3S-1500 template design, *Copyright Gaisler Research*, 1-153.
18. Mansour, W.; and Velazco, R. (2013). SEU fault-injection in VHDL-based processors: a case study. *Journal of Electronic Testing: Theory and Applications*, 29(1), 87- 94.
19. Daemen, J.; and Rijmen, V. (2002). The design of rijndael AES - the advanced encryption standard - the advanced encryption standard process, *Springer-Verlag Berlin Heidelberg*,1-8.
20. Spinean, B.; and Gaydadjiev, G. (2012). Implementation study of FFT on multi-lane vector processors. *Proceedings of Euromicro Conference on Digital System Design DSD'12*, Izmir, Turkey, 815-822.
21. Gaisler, J. (2002). A portable and fault-tolerant microprocessor based on the SPARC v8 architecture. *Proceedings of International Conference on Dependable Systems and Networks DSN'02*, Washington, DC, USA, 409-415.
22. Cotroneo, D.; and Madeira, H. (2013). Innovative technologies for dependable OTS-based critical systems –introduction to software fault injection, *Springer-Verlag Italia*, 1-16.