SEUs Mitigation on Program Counter of the LEON3 Soft Processor

Afef KCHAOU
University of Tunis El Manar, Faculty of Sciences of Tunis
Laboratory of Electronics and Microelectronics of Monastir
Monastir, Tunisia

Wajih EL HADJ YOUSSEF
University of Monastir
Laboratory of Electronics and Microelectronics of Monastir
Monastir, Tunisia

Rached TOURKI
University of Monastir
Laboratory of Electronics and Microelectronics of Monastir Monastir, Tunisia

***Abstract*—Analyzing and evaluating the sensitivity of embedded systems to soft-errors have always been a challenge for aerospace or safety equipment designer. Different automated fault-injection methods have been developed for evaluating the sensitivity of integrated circuit. Also many techniques have been developed to get a fault tolerant architecture in order to mask and mitigate fault injection in a circuit. Fault injection mitigation and repair techniques are applied together on LEON3 processor in goal to study the reliability of a soft-core. The so-called NETlist Fault Injection (NETFI+) tool is a fault injection techniques used in this paper. The prediction of Single Event Upset (SEU) error-rates between radiation ground testing and FPGA implementation have been done with good and accurate result. But no functional simulations have been performed. A Triple Modular Redundancy (TMR) is used in this paper as a repair technique versus fault injection. This paper analyses the effectiveness of fault tolerant method on LEON3 soft-core running a benchmark. It starts by evaluating the behavior of LEON3's program counter against Single Event Upset error-rate accuracy between the functional simulation and the FPGA emulation and an analysis of the LEON3 reliability in presence of fault tolerant technique. The objective is to offer, through the new version of NETFI+ with introducing a fault tolerant technique, the possibility to designers to evaluate the benefits of SEUs mitigation for the LEON3 processor on the program counter.**

***Keywords*—NETFI+ ; fault injection; SEUs; LEON3; simulation; emulation; reliability; TMR**

I. INTRODUCTION

Embedded system undergoes several changes across the years, starting from simple mono CPU running applications to a complex system including co-processor, memory, input and output models. Using embedded systems in special applications, safety-critical or mission-critical, allows evaluating their dependability in presence of faults on the circuit or in the implemented application.

Fault injection can be used to evaluate embedded system running its own application [1].

Transient faults and soft errors lead to faults in a system without damaging the system under evaluation. Transient faults are represented by a single or multiple nodes upset directly attributable to excess charge carriers created by an external source of radiation. Soft errors are defined by the impact of a transient fault that can be propagated beyond one clock cycle [2]. It flips one or more bits, modifying the data store of a memory cell, register, flip-flop and latch. SEU and Single Event Transient (SET) are soft errors that affect only one bit, other type of faults are used to modify more than one bit, and it's a Multiple Bit Upset (MBU) [3].

Areoflex Gaisler LEON3 processor has become more used in a critical and safety application, such as in automotive, multimedia system, wireless and more applications which require reliability. Fault injection in LEON3 soft-core is done in many works classified according to the type of faults, the methods used, the block under test, etc. LEON3 is characterized by its complexity and size. It's a reason to be a good design for evaluating the benefits of fault tolerant techniques [4].

In [5], injection of SEU, SET and MBU faults have been done in many components of LEON3, showing that integer unit and multiplier unit are more susceptible against SEU and MBU fault injection.

Emulation-based fault injection in LEON3 is done in [6], allowing a reduction in the experimental time.

In [7], SEU fault injection by FPGA emulation is made by applying an exhaustive fault injection in internal memory of LEON3. The results obtained show that the memory cell containing the data is the most sensitive to SEU.

In [8], a new methodology is proposed to evaluate the real cache 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 with several benchmarks showing that their proposed tool predicted all real errors with little over-estimation. Fault injection is done by radiation and emulation. The result shows that all the cache addresses are sensitive to SEU injection.

In [9], evaluating the effects of single bit errors at the memory and register locations is done using a high level error

injection technique. The results obtained show that this method is inaccurate in comparison with the techniques using a flip-flop error injection.

SEU fault injection, called bit-flip, upset or soft-error propagates in the design depending on the application [10] and it can cause a data corruption or a circuit malfunction. SEUs are random in space and time, they can modify any element on memory location also at any instant time. In [11], the application of a new methodology to attack a Program Counter (PC) of ARM is done by modifying the load-instruction of the PC.

In previous works, the PC of LEON3 was not evaluated in point of sensitivity because to its important function in system security, in addition, based on the work in [5] avowed that the integer unit is the most critical block of LEON3 to SEU and MBU fault injection. The principal goal in this paper is to give the benefits of SEU mitigation for the LEON3 processor on the PC by adding a repair technique of fault tolerant like a TMR in this work.

A new methodology is improved for fault attacks, NETFI+, in order to evaluate the behavior of LEON3 soft-core, the SEU error-rate reliability between the functional simulation and FPGA emulation is done by injecting an SEU fault on the flip-flop of LEON3's IU block precisely on the program counter register according to its importance in the instruction execution process. The principal goal in this paper is to present fault injection approach and analyze countermeasure effectiveness in circuit security.

The paper is organized as follows. In Section II a description of the NETFI+ principle will be done, overview of LEON3's integer unit is presented in Section III. Next section presents the NETFI+ flow, the analysis of the reliability of the LEON3's program counter by evaluating the SEU error-rate reliability between simulation and FPGA emulation is presented in Section V. Section VI provides a presentation of a repair technique used in order to evaluate the SEU mitigation for the LEON3 soft processor on the program counter. A conclusions and perspectives will be presented in the last section.

II. NETLIST FAULT INJECTION PRINCIPLE

Fault injection approaches are based on injecting faults that can induce errors. Many researches separate between the methods of fault injection, it can be classified depending on two techniques based on hardware and software fault injection.

A software fault injection is presented by using a software program to inject faults in a physical model. Simulated fault injection can be observed and controlled while the system is simulated using HDL simulator.

A hardware fault injection allows evaluating a behavior of a system based on Commercial off-the-shelf (COTS) processor [12]. It's widely used and it can be classified on three categories [13]:

- Logical fault injection using debugging facilities: This type of injection allows to the processor logic resources to access their internal blocks and to add bit-flips.
- Physical fault injection: this method is accomplished using laser beams, electromagnetic interferences or a radiation in goal to induce faults in integrated circuits. This method offers actual hardware faults on real systems. It requires expensive material and the number of faults injected is limited, also a deep knowledge of the actual layout of the circuit [14].
- Logical fault injection: it can be made by circuit simulation using hardware description languages (HDL models) simulator or by circuit emulation using hardware emulation platforms. In simulation-based fault injection, the system under test is simulated in another system, while the emulation-based fault injection facilitates the injection on complex models by reducing time spent by a simulation-based fault injection.

The fault injection type used in this paper is the last one: Logical fault injection. Simulation-based fault injection allows the fault injection in high level models. In general, fault injection is presented by a bit-flip fault model when the content of a memory cell is inverted. It permits to evaluate the behavior of fault tolerance mechanisms [14].

The principle idea of the logical fault injection is the injection at the bit-flip model by inverting the content of a memory cell at the instant injection. Studying the reliability of an embedded system is a principal goal to define the capability of the system to run its function in abnormal condition for a given period of time [15]. Soft errors disturbing memory cells and registers in embedded system are called SEU was analyzed to evaluate the soft error-rate [13].

Sensitivity of the LEON3's integer unit against soft errors was estimated through two fault-injection campaigns. A first one was performed in a simulation in order to analyse a random SEU injection. A second one was performed in FPGA emulation to accelerate the fault injection campaign and mainly to evaluate the SEU error-rate reliability in the simulation campaign also the validation of a NETFI+ tool.

The NETFI+ tool used in this paper is based on netlist fault injection. It allows to inject SEU, MBU and SET faults in circuits at Register Transfer Level implemented on FPGA. This method enables to inject faults in all memory cell and at any clock cycle, exhaustive or randomly in time and location. In this paper, the principle idea is to study the reliability of LEON3's PC against SEU fault injection.

The HDL source code of the circuit is synthesized to get the correspondent netlist [16]. In next step, a MODNET (MODify NETlist) tool, described in [17], will be used to choose the type of faults which can be injected and give a modified netlist.

A NETFI+ tool in this work is improved to inject faults in all the memory cell of LEON3 also to inject all type of faults SEU, MBU and SET.

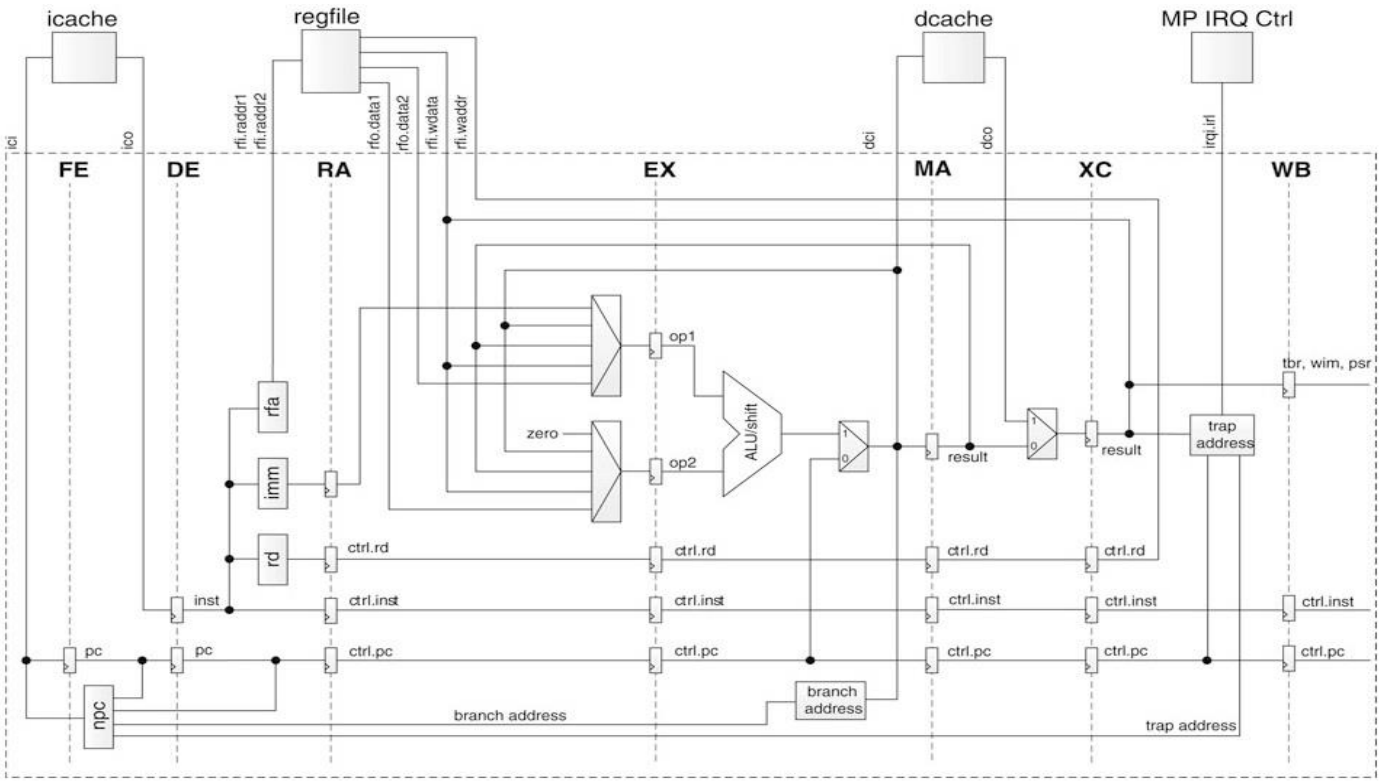

Fig. 1. The micro-architecture of the LEON3's Integer Unit [18].

III. Overview of LEON3's Integer Unit

The LEON3 Integer Unit (IU) is fully compliant with the SPARC V8 standards. SPARC is a CPU instruction set architecture derived from RISC. It comprises an integer unit (IU), an optional Floating-point unit (FPU) and a coprocessor (CP).

The IU executes the arithmetic instructions, computes memory addresses (load/store), maintains the Program Counter (PC) and controls the instruction execution for the FPU and the CP. "Fig. 1" shows the pipeline of the IU which consists of seven-stages with Harvard architecture.

IU integrates seven stage of pipeline, the FE stage (FEtch) fetches the instruction from the instruction cache through its address given by a PC. DE stage (DEcode) decodes the instruction. In the RA stage (Register Access), all operands are read from the register file or from the internal data bypasses and stored in EX stage (EXecute). ME stage (MEmory) stores the results and communication between IU and the other peripherals components which can be done. In XC stage (eXCeption) all traps and interrupts are resolved. In WR stage (WRite), a data not sent to the register file will be stored [19], [9].

Integer Unit controls, in general, all the operation of the processor and it includes two types of register: general-purpose registers and control/status registers. Whose General-purpose registers is a 32-bit registers, called r register.

An instruction can access the 8 global registers and a 24 registers window into r register. The register window contains 8 in and 8 local registers of a particular register set. The 8 in registers are addressable from the current window, the out registers.

The IU control/status registers include Processor State Register (PSR), Window Invalid Mask (WIM), Trap Base Register (TBR), Multiply/divide Register (Y), Program Counters (PC), Implementation-dependent Ancillary State Registers (ASRs) and Implementation-dependent IU Deferred-Trap Queue.

IV. Fault Injection Flow

The emulation of SEU faults is done in the PC which is the overall security of any embedded system, in this case the LEON3 processor. PC gives the address of the instruction currently being executed by the IU.

Only 30-bit of PC will be used in the six stages of pipeline (FE, DE, RA, EX, ME, XC) because the LSB two bits of the PC are not used in the configuration, its implementation will cause a debug of the HDL model and an area waste in synthesis. A NETFI+ method allows injecting SEUs faults in a Flip-Flop of PC in all the stage of pipeline, in total 180 FF will be used to control the PC. "Fig. 2" illustrates the workflow adopted in the NETFI+.

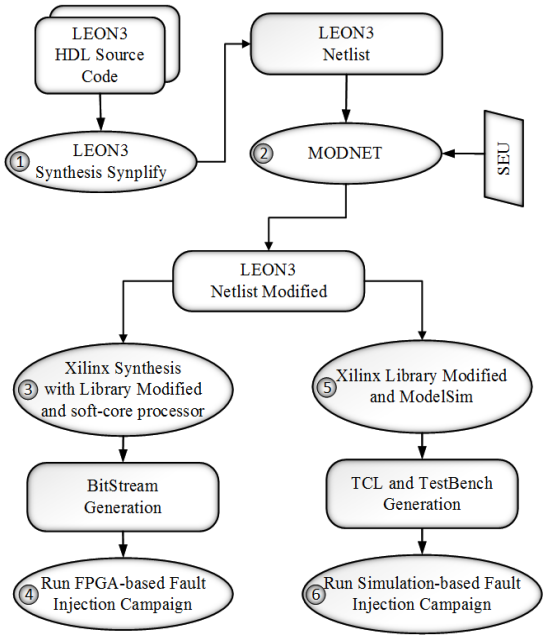

Fig. 2. Fault injection flowchart.

Initially, a Hardware Description Language (HDL) of the LEON3 is synthesized by Synplify tools to get the Verilog netlist in Step 1. This first step does not require any modification to the original design.

In Step 2, the first netlist resulting will be used as input for the MODNET tools, which adds a new signal "INJ" to all the Flip-Flop (FD and FDE) components used in the block of IU. After that, the new netlist obtained is then synthesized, by Synplify tools, in Electronic Design Interchange Format (EDIF) using a modified version of the sensitive components, which includes "INJ" signals to access them to fault injection. "Fig. 3" exhibits the addition of the 'INJ' signal in the design.

In Step 2, two possibilities of test injection can be applied, the first one is by FPGA emulation, steps 3 and 4, and the second one is by a simulation campaign, steps 5 and 6.

The FPGA emulation campaign is performed in Steps 3 and 4. In Step 3, the EDIF file obtained in Step 2 is then attached to the soft-core processor and the last synthesis is performed to generate a bitstream based on the target FPGA. Finally, in Step 4 the experiment is executed in hardware-based FPGA platform.

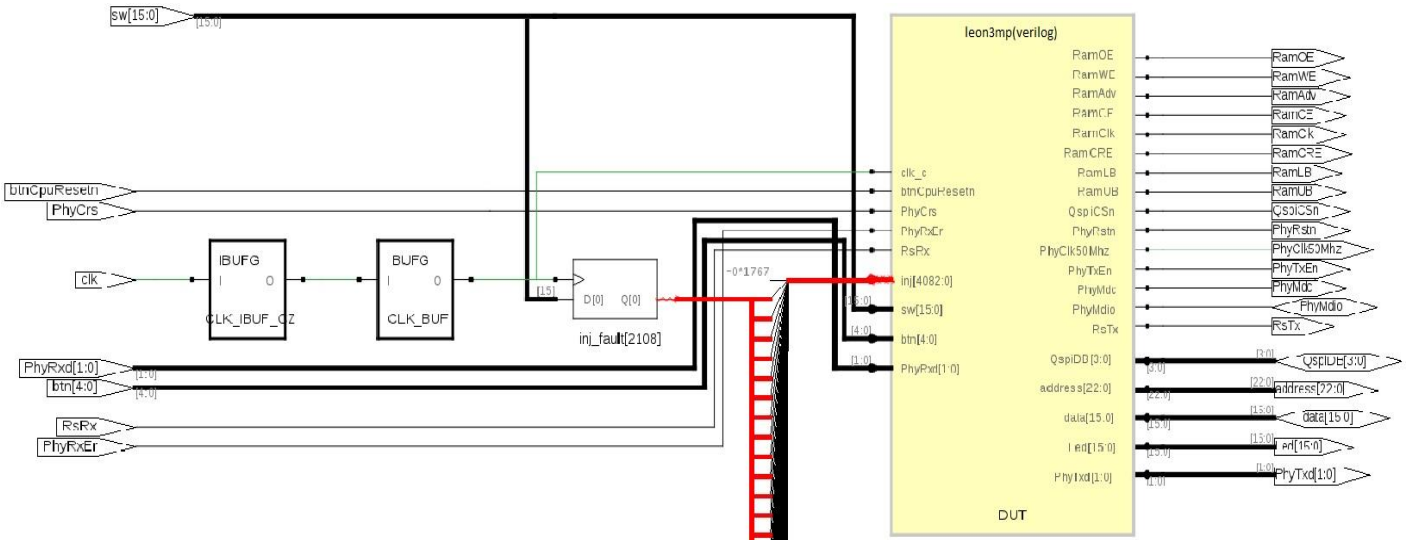

Fig. 3. Architecture of LEON3 blog diagram which includes "INJ" signal.

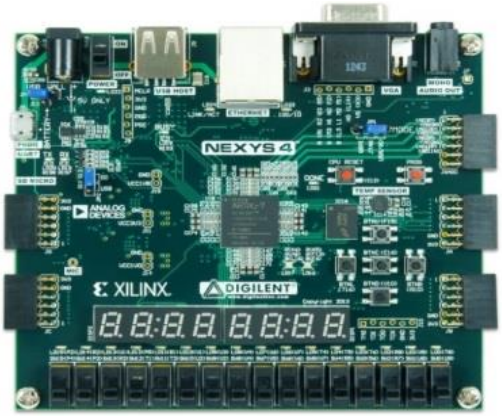

Fig. 4. The Nexys4 board.

In this work, a Nexys4 board, equipped with Xilinx Artix-7 XC7A100T-CS324, is used which is a complete circuit board. As shown in "Fig. 4", the board is occupied by a diverse I/O, development connectors that allows a connection with the LEON3 implemented.

The setup and control of the fault injection experiments are performed by a Soft-Core Fault Injection Processor (SCFIP), embedded in the FPGA, and by a Tool Command Language (TCL) script in a personal computer, as can be seen in "Fig. 5".

The SCFIP is used as the controller in charge of randomly selecting the time and in which registers will inject the faults. The results of LEON3 execution are sent to a personal computer connected through the UART interface.

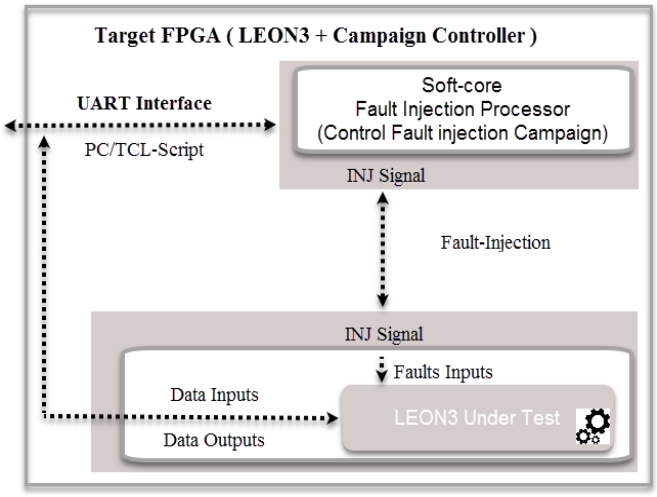

Fig. 5. FPGA fault injection strategy.

The simulation fault injection campaign following Step 5 and 6 of "Fig. 2". In Step 5, the files obtained in step 2 are then attached to the Xilinx modified library in the ModelSim Mentor Tool [20]. Finally, in Step 6 the simulation, setup and the control of the fault injection campaign are performed by a TestBench in a dedicated server equipped with two Intel Xeon CPU E5-2620 and 64GB of RAM memory.

V. Program Counter Reliability

The NETFI+ tool described on Section 2 will create a variety of SEU influence on LEON3. The first campaign of injection is done by a simulation fault injection. A benchmark, MulMatrix (Matrix product), is used to execute a simulation and to compare the results obtained with the standard results (Golden Results).

The benchmark used in this test is selected because it does not take enough time in RTL simulation also the number of repetitive instructions involved by its execution enables to guarantee a wrong behavior in presence of fault injection.

A random SEU fault injection in the PC register at the stages of pipeline is done. In total, 1080000 faults are injected in sensible FF of LEON3's PC during one period, 100% of the flip-flop of FE and DE stage are sensitive to SEU, the FF of the other stages are non-sensitive.

The results obtained can be classified in four categories shows in "Table 1": overwritten Faults, Failure results, Timeout and Stopped Execution.

The faults can be overwritten in some cases, it explains that the error is masked and cannot modify the result, this explains that at the moment of the injection, the PC does not point to the instruction used at the instant of simulation (about 100% faults undetected in RA, EX, ME and XC stage).

Failure results is procured when the simulation is done but giving a false result, this explains that at the moment of SEU injection, the PC does not point in the correct address of the currently instruction used in simulation (6.66% in FE stage and 3.33% in DE stage).

TABLE I. SEU Fault Injection Analysis

Type of Result	Stages of pipeline					
	FE	*DE*	*RA*	*EX*	*ME*	*XC*
%Overwritten	-	-	100	100	100	100
%Failure result	6.66	3.33	-	-	-	-
%Timeout	3.33	-	-	-	-	-
%Stopped execution	90	96.66	-	-	-	-

The simulation can exceed the approximate time of simulation (4365 us in this case) like a 3.33% in FE stage, this type of faults is named Timeout, the PC in this case stop incrementing and this explains a cause an infinite loop in simulation. Also the execution can be stopped (90% in FE stage and 96.66% in DE stage). In another way, the simulation cannot be finished normally, and the execution stops just after the moment of injection. In other words, the PC does not contain any address to be pointed.

The benefits of the simulation-based fault injection that it allows a fine-grained analysis through the assembly code of fault injection campaign. The assembly code contains all the instructions of simulation. SEU injection in PC occur some traps.

The results obtained are resumed in "Table 2". The traps can be more detailed [21]:

- Illegal instruction:

When the simulation ends before a normal time of execution, an attempt is made to execute the instruction with an unimplemented opcode or an UNIMP instruction (Assembly code: unimp (trapped)). Other reason can be responsible for the stopped execution that the instruction would result an illegal processor state at the decode stage (Assembly code: save %sp, -0x0060, %sp (trapped)).

- Privileged instruction:

At the fetch stage, the PC stops to increment and remains constant and in another case, the PC does not contain any address to be pointed. The assembly code shows that an attempt was made to execute a privileged instruction (Assembly code: ldub [%o4], %o5 [0x0000XXXX]).

- Window overflows:

A SAVE instruction is responsible for this trap because at this instant the Current Window Pointer (CWP) will point to a window marked not valid in the WIM.

- Window underflows:

It's caused by a RETT or a RESTORE instruction attempted in this case when the CWP would point to a window marked invalid in the WIM. In fact, RESTORE instruction has the same role of ADD instruction, it allows to increment by '1' the CWP and to compare it with WIM, if the WIM bit corresponding to the new CMP is '1', a window underflow trap is then generated.

TABLE II. TRAPS ANALYSIS

Type of Trap	Stages of pipeline	
	FE	***DE***
Illegal_instruction	49.5%	51.33%
privileged_instruction	14.4%	6.33%
Window_overflow	12.5%	-
Window_underflow	23.5%	42.33%

The NETFI+ tool applied in fault injection by simulation takes a lot of time, an FPGA implementation based fault injection is done to validate a NETFI+ tool and to evaluate the results obtained by simulation.

The simulation based fault injection offers the analysis of the reliability of any circuit, such as a microprocessor pipeline or cache memory using all the types of faults. The term exhaustiveness can be done only by this technique, but it consumes a long time. Emulation based fault injection allows to inject a high number of faults by winning the time of the fault injection process.

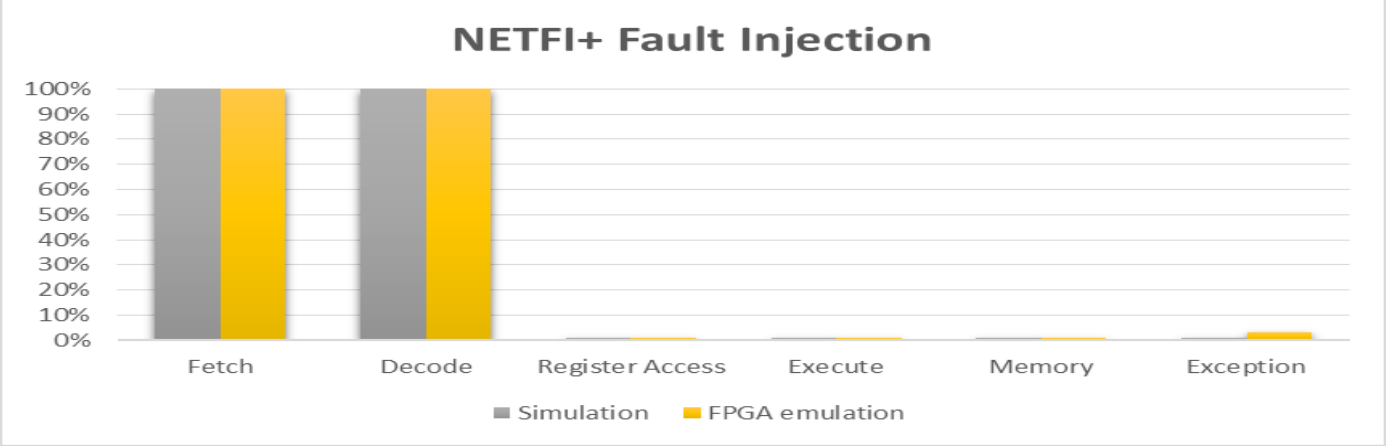

Fig. 6. NETFI+ analysis simulation versus FPGA emulation.

A NETFI+ tool is validated by simulation and FPGA implementation when a FF of FE and DE stage are 100% sensitive to SEU (“Fig. 6”). For the flip-flop non-sensitive of the other stages, the result remains the same except for the flip-flop of the XC stage when 0.02% is sensible to SEU.

The NETFI+ tool is limited in number of faults injected in FPGA emulation but it is faster than the fault injection by simulation. The analysis of the results obtained in simulation shows the benefits of the NETFI+ tool in simulation in its accuracy and the large number of faults which can be injected.

VI. LEON3 SEU MITIGATION EVALUATION

A. Principle of TMR

TMR is the most commonly used as a mitigation technique against SEUs for FPGA designs, used in radiation environments. The principle of TMR technique is done by triplicating a design and voting on the outputs of the three modules triplicated. TMR can be implemented on the latest commercial FPGA technologies, but it is costly in terms of area and power. It makes the circuit fault tolerant by masking and reducing the faults. It protects the design from errors propagated in LUT, internal state and control signals. “Fig. 7” shows the principle idea of the integer unit redundant with single voter.

The single voter with the triplicated logic will mask logic and errors created by SEUs. While two or three redundant copies of the design work correctly, errors will be masked and the output of the block will be correct [4].

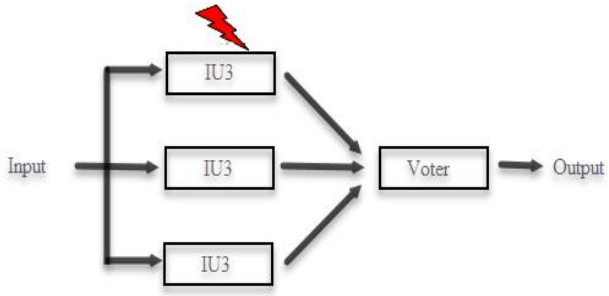

Fig. 7. TMR flow.

In [22], many approaches are used to detect a laser SEU faults for LEON3 on SRAM-based FPGA with the integration of several fault countermeasure techniques, the results obtained show that the modular triplication with single voter is the best one to mask errors. In [23], authors announced that a TMR presents a portable and robust solution.

TMR is generally used as a mitigation technique against a radiation fault injection. In [4], diverse repair techniques have been used to improve the SEUs mitigation for the LEON3 processor using two different approaches: Fault injection and Neutron radiation test. The results evince that using TMR with both CRAM (configuration memory) scrubbing and BRAM (internal block memory) scrubbing demonstrates that the reliability improvement is about 51.30x which used fault injection, and about 48.85x using Neutron radiation test.

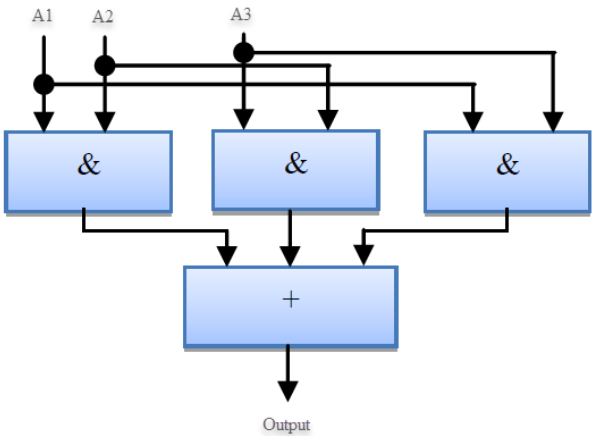

Fig. 8. Logic bloc of the voter.

The principal idea of TMR used in this work is by triplicating the integer unit of LEON3 and comparing the outputs with the golden operation (without countermeasure techniques) by voting the results of three redundant copies of the design to mask SEUs.

The voter is the important element in TMR technique. The importance of reliability in a majority voter is attributed to its application in both conventional fault-tolerant design and new Nano-electronic systems.

"Fig. 8" shows the logic bloc of the voter that masks faults in a single block of IU, such as A1, A2 and A3 which represent the output of the first, the second and the third copy of IU, respectively.

B. Analysis of SEU mitigation on LEON3's Program Counter

In order to evaluate the behavior of LEON3 facing SEUs, fault injection is performed in the Program Counter register of the IU over its stage of pipeline. The proposed strategy of fault injection campaign NETFI+ is used and validated using two different campaign methods: simulation and FPGA emulation. The result obtained shows that 100% of FF in Fetch and Decode stage are sensitive to random SEUs fault injection by simulation and FPGA emulation. In goal to mitigate SEUs faults in the PC of LEON3, hardware integration of TMR fault tolerant technique in LEON3, by redundant the integer unit, is done. Only the bits of the PC register of Fetch and Decode stage will be used to mitigate de SEUs faults.

This section summarizes the TMR fault tolerant testing on LEON3 for SEUs fault injection. The SEU mitigation for the LEON3 processor on the program counter is shown in "Table 3", two LEON3 design variations is shown in a table, the first design without TMR fault tolerant technique (unmitigated) and the second design with TMR.

The simulation of the design with TMR is made with success. Improvement is represented by a terms of sensitivity mitigation for fault injection.

The improvement in design sensitivity, according to the baseline design is enhanced when integrated a TMR module and injecting SEUs faults in one copy of IU, whose the TMR mitigates all the SEU injected to given about 100% of faults undetected.

TABLE III. SEUS FAULT INJECTION RESULTS WHICH INCLUDES TMR

Description	**Unmitigated**	**One copy of IU affected**	**Two copies of IU affected**	**Three copies of IU affected**
#faults injected	#18000	#18000	#36000	#108000
%sensitivity	100%	0%	0.1%	0.06%
#sensitive bits	#9000	#0	#2700	#3450
Improvement	1.00x	-	3.33x	2.60x

While the injection in three copies of IU demonstrates about 2.60x improvement over the injection within two copies of IU, the SEU mitigation technique used in this work, TMR, provides an important improvement in design sensitivity over the unmitigated baseline design. The percentage of sensitive bits within three copies of IU is about 3.19%, this is significant that the SEUs mitigation attained reaches about 96.81%.

The results reveal that using TMR in SRAM-based FPGA without scrubbing [5], the percentage of the sensitive bits is about 4.65% while in this paper using a TMR in IU, the sensitive bits represent 3.19% of the total bits.

VII. CONCLUSIONS AND PERSPECTIVES

In this paper, an extensive fault injection campaigns are done to evaluate the robustness of soft-core LEON3 processor against Single Event Upset. A new fault injection approach was improved in order to evaluate the susceptibility to soft errors, SEU, in LEON3's program counter.

Two approaches to evaluate the sensitivity of integrated circuits to Single Event Upsets provoked by energetic particles present in the environment (space, Earth's atmosphere) were explored. The first one is based on RTL simulations allowing evaluating IC sensitivity against SEU at early design phases, while the second one focuses on FPGA emulation which enables to obtain results closer to the ones of the hardware IC.

The accuracy/limitations of both approaches are studied by the analysis of experimental results. Fault injection based RTL simulation can be applied at early design phase, allowing fine-grained analysis also efficient local solution but it requires very important simulation time. For the fault injection based FPGA emulation is faster than fault injection based RTL simulation but it does not give detail information on reporting.

The results in this work put in evidence the importance of increasing the robustness of LEON3's Program Counter register against soft-errors for critical applications. A hardware integration of a countermeasure unit is done in this work to give back the design fault tolerant. Analyzing the results shows that SEUs mitigation on the PC at the sensitive stage of pipeline, Fetch and Decode stages is improved about 99.979% using a repair technique, TMR.

Future work must be addressed to other types of faults like SET in the principal unit of LEON3, the integer unit, which including a SETs mitigation technique, TMR module.

REFERENCES

[1] M. Murciano, M. Violante, "Validating the Dependability of Embedded Systems through Fault Injection by means of Loadable Kernel Modules," IEEE International High Level Design Validation and Test Workshop, 2007, pp 179-186.

[2] A. Spilla, I. Polian, J. M¨uller, M. Lewis, V. Tomashevich, B. Becker, W. Burgard, "run-time Soft Error Injection and Testing of a Microprocessor using FPGAs," University of Frelburg, Jun 2013.

[3] G.S. Rodrigues, F. Rosa, Á. Barros de Oliveira, F. Lima Kastensmidt, L. Ost, R. Reis, "Analyzing the Impact of Fault Tolerance Methods in ARM Processors under Soft Errors running Linux and Parallelization APIs, " IEEE Transactions on Nuclear Science, 2017, pp. 1-7.

[4] A M. Keller, M J. Wirthlin, "Benefits of Complementary SEU Mitigation for the LEON3 Soft Processor on SRAM-based FPGAs," IEEE Transactions on Nuclear Science, 2016, pp.519-528.

[5] H. Abbasitabar, H. Zarandi, R. Salamat, "Susceptibility Analysis of LEON3 Embedded Processor Against Multiple Event Transients and Upsets," IEEE 15th International Conference on Computational Science and Engineering, 2012, pp. 548-553.

[6] K. Chibani, M. Portolan, R. Leveugle. "Fast Register Criticality Evaluation in a SPARC Microprocessor," 10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2014, pp 1-4.

[7] A. Kchaou, W. El Hadj Youssef, R. Tourki, F. Bouesse, P. Ramos, R. Velazco, "A deep Analysis of SEU Consequences in the Internal Memory of LEON3 processor," 17th IEEE Latin-American Test Symposium, 2016, pp 178.

[8] S. Houssany, N. Guibbaud, A. Bougerol, R. Leveugle, F. Miller, N. Buard, "Microprocessor Soft Error Rate Prediction Based on Cache Memory Analysis," IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 59, NO. 4, 2012, pp 980-987.

[9] H. Cho, S. Mirkhani, C-Y. Cher, J-A. Abraham, S. Mitra, "Quantitative Evaluation of Soft Error Injection Techniques for Robust System Design," 50th Annual Design Automation Conference, May 29 - June 07, 2013, ACM, Austin, TX, USA, pp: 1-10.

[10] P. Peronnard, R. Ecoffet, M. Pignol, D. Bellin, R. Velazco, "Predicting the SEU Error Rate through Fault Injection for a Complex Microprocessor," IEEE International Symposium on Industrial Electronics, 2008, pp 2288-2292.

[11] N. Alimi, Y. Lahbib, M. Machhout, R. Tourki, "An RTOS-based Fault Injection Simulator for Embedded Processors," International Journal of Advanced Computer Science and Applications, Vol. 8, No. 5, 2017, pp 300-306.

[12] A. Clough, "Commertial Off-The-Shelf (COTS) Hardware and Software of Train Control Applications: System Safety Considerations," The national onformation service, Springfield, Virginia 22161. April 2003.

[13] M. Portela-Garcia, C. Lopez-Ongil, M. Garcia Valderas, L. Entrena, "Fault Injection in Modern Microprocessors Using On-Chip Debugging Infrastructures," IEEE Transactions on dependable and secure computing, Vol.8, No.2; March-April 2011, pp 308-314.

[14] J I. Gonzalez, A S. Cases, A M. Alvarez, S C. Asensi, H G. Miranda, M A. Aguirre, " Contrast of a HDL model and COTS version of a microprocessor for SOFT-ERROR Testing," 18th IEEE Latin American Test Symposium, 2017, pp. 1-6.

[15] S. Ahmad, S. Sardar, B. Noor, A. Asar, "Analyzing Distributed Generation Impact on the Relaibility of Electric Distribution Network," International Journal of Advanced Computer Science ans Applications, Vol.7, No.10, 2016, pp 217-221.

[16] W. Mansour, R. Velazco, "SEU fault-injection in VHDL-based processors: a case study," 13th Latin American Test Workshop (LATW), 2012, pp 1-5.

[17] W. Mansour, R. Velazco, " An automated SEU fault-injection method and tool for HDL-Based Designs," IEEE Transactions on Nuclear Science, 60(4):2728-2733, Aug 2013.

[18] M. Danek, L. Kafka, L. Kahout, J. Sykora, R. Bartosinski, "UTLEON3: Exploring Fine-Grain MultiThreading in FPGAs," Springer-Verlag New York, 2013, pp. 1-222.

[19] J. Gaisler, M. Isomäki, "LEON3 GR-XC3S-1500 Template Design," Gaisler Research Based on GRLIB, October 2006.

[20] "ModelSim Advanced Verification and Debugging," SE Tutorial, Version 6.0b, 15 November 2004.

[21] Gaisler groups,"The SPARC Architecture Manual Version 8," SPARC International, 1991-1992.

[22] F. Benevenuti, F L. Kastensmidt, " Evaluation of Fault Attack Detection on SRAM-based FPGAs," 18th IEEE Latin American Test Symposium, 2017, pp. 1-6.

[23] P. Rech, C. Aguiar, C. Frost, and L. Carro, "An efficient and experimentally tuned software-based hardening strategy for matrix multiplication on gpus," IEEE Transactions on Nuclear Science, vol. 60, no. 4, Aug 2013, pp. 2797–2804.