Selective Protection Analysis Using a SEU Emulator: Testing Protocol and Case Study Over the Leon2 Processor

M. A. Aguirre, *Member, IEEE*, J. N. Tombs, *Member, IEEE*, F. Muñoz, V. Baena, H. Guzmán, J. Nápoles, A. Torralba, *Senior Member, IEEE*, A. Fernández-León, F. Tortosa-López, and D. Merodio

***Abstract*—VLSI circuits for space application must be protected by the insertion of massive redundancy. However, this increases silicon area and the production costs, therefore designers can often consider leaving some large, noncritical subcircuits unprotected. This paper presents how FT-UNSHADES, a nonintrusive tool for fault injection on emulated hardware, helps designers to select the proper level of protection in every subcircuit. Using FT-UNSHADES, a test procedure is proposed that provides: 1) information about the quality of the test vectors, 2) a proper estimation of the number of injected faults required to get confidence about the results of a fault injection campaign, and 3) information about the criticality of individual subcircuits by selective fault injection campaigns. In addition, FT-UNSHADES allows the insertion of multi-bit flips. This test procedure has been applied to three different, protected and unprotected, versions of the well-known Leon2 processor, and the results are discussed here.**

***Index Terms*—Fault injection, single event upset (SEU), FPGA-based emulation, FPGA reconfiguration, multi-bit upset, ASIC.**

I. Introduction

ASICs FOR SPACE applications are intended to work in a harsh radiation environment. Specific design techniques are required to protect their memory elements against soft-errors by means of error correction structures. The circuits are hardened by applying some kind of redundancy, like Triple Modular Redundancy (TMR), to their most sensitive subcircuits. If an error in the functional behavior or in the hardening strategy is found, a redesign cycle will be required with the consequence of an increase in the design cost and development time. Furthermore, if due care is not taken, hardware redundancy can be unintentionally removed during the synthesis optimization, leaving unprotected zones hidden to the designer (although log files contain key information about the synthesis process, they are sometimes difficult to interpret or not correctly revised).

Manuscript received October 6, 2006; revised February 20, 2007. This work was supported by the European Space Agency under Contract Number 17540 and by the CELPAE (P05-TIC-00250) Project from the regional government of Andalusia. The European Xilinx SEE consortium provided the XTMR version of the Leon2 processor.

M. A. Aguirre, J. N. Tombs, F. Muñoz, V. Baena, H. Guzmán, J. Nápoles, and A. Torralba are with the Escuela Superior de Ingenieros, Universidad de Sevilla, E41092 Sevilla, Spain (e-mail: aguirre@gte.esi.us.es).

A. Fernández-León, F. Tortosa-López, and D. Merodio are with the Microelectronics Section TEC-EDM / ESTEC / European Space Agency, 2200 AG Noordwijk, The Netherlands.

Digital Object Identifier 10.1109/TNS.2007.895550

In order to check the quality of protection against Single Event Upsets (SEUs), SEU sensitivity is usually assessed by means of dynamic fault injection test. During the test faults are injected into a specific target in the circuit at a given time and the behavior of the faulty circuit is recorded [1]. In the space context a fault is meant as a SEU, or bit-flip, which consists of changing the current state of a flip-flop in a particular clock cycle.

Circuit simulators have been used for fault injection purposes. They provide exhaustive information of the circuit behavior, allowing full access to any internal signal at any time. However circuit simulators are very slow [2], especially for large circuits with a large number of test vectors. On the other hand, circuit emulation is considered to be a good candidate to reduce the time required for a fault injection campaign [3]–[11]. A circuit emulator uses a programmable hardware platform, usually an FPGA, to host the circuit under test. After the synthesis process, the circuit netlist is mapped on the internal resources of the FPGA. To start circuit emulation, the resulting netlist is downloaded into the FPGA where it runs at a high speed. Unfortunately, circuit emulators suffer from limited controllability and observability. In their simplest implementation, the circuit under test can be only controlled by means of their primary input signals and its behavior can be only observed by means of their primary output signals. This limitation is especially frustrating in the case of a fault injection campaign, where the same circuit is emulated thousands of times, with only a bit-flip change in one of its internal registers in a given clock cycle.

Different solutions have been proposed to increase the controllability and observability of circuit emulators. Many of them modify the circuit under test introducing extra hardware to change the state of the internal registers during the fault injection campaign (and, optionally, to trace fault propagation) [6], [7], [10]. These methods are known as *instrumentation*, and the extra circuitry introduced to control the fault injection process is known as *the instrument*. Instrumentation is a powerful method to increase the controllability, (and, optionally, the observability), of circuit emulators, and provides a high speed when compared to circuit simulators. However, instrumentation is an intrusive technique which modifies the design under test in its high level description, producing a model for testing that can, eventually, be different to the original design.

A second approach is to increase controllability and observability in FPGA-based circuit emulators taking advantage of the partial or total reconfiguration of the Xilinx FPGAs [8], [9].

FT-UNSHADES uses a unique approach, based on dynamic reconfiguration techniques, that allows the user to maintain the same Register Transfer Level (RTL) structure in the final ASIC and in the circuit emulated on the FPGA. Using commercial tools like Formality,[1] it is possible to maintain and guarantee an equivalence between ASIC and FPGA netlists. In FT-UNSHADES a fault is modeled as a change of the information stored in a register at a given clock cycle. To solve the controllability and observability problems, FT-UNSHADES takes advantage of the configuration mechanism inside every Xilinx FPGA, called *Capture* and *Readback*.[2] Faults are injected into the FPGA flip-flops by means of read-modify-write operations of the FPGA configuration memory. Using partial reconfiguration, every bit of the current state of a circuit can be low level manipulated in such a way that it is not necessary to make any changes in the circuit netlist. As a consequence, FT-UNSHADES is a nonintrusive approach.

Other authors have proposed similar techniques for the fault analysis of FPGAs in space applications, focusing on the effects of damage to the FPGA configuration memory rather than the flip-flop state of the design [13].

When compared to other existing tools, FT-UNSHADES incorporates extra features which provide additional capabilities for the analysis of the results of a fault injection campaign. They allow the designer to restrict the fault injection to selective parts of a hierarchical design, which is of interest to determine the sensitivity of each different subcircuits, and thus decide their proper level of protection. FT-UNSHADES also permits the testing of multi-SEUs. Particle hits or radiation can produce more than one simultaneous bit-flip in registers, especially if they are close in the layout or the error is produced by a transient pulse in a combinational network that feeds multiple registers (Single Event Transient). Higher clock speeds and smaller technologies suggest that robustness against multi-SEU injection will likely be a requirement for space applications in the near future.

This paper is organized as follows. Section II introduces the FT-UNSHADES system. Section III describes the fault injection process in FT-UNSHADES and discusses the proposed testing procedure. Section IV presents a fault injection campaign of the well-known Leon2 processor using FT-UNSHADES with the testing procedure defined in Section III. Three different protected and unprotected versions of this processor have been used. The results obtained with the fault injection campaigns are discussed showing the additional features provided by FT-UNSHADES. Finally, some conclusions are drawn in Section V.

II. FT-Unshades

FT-UNSHADES [11] is a platform based on Xilinx Virtex FPGAs. It is a version of the UNSHADES hardware and software co-design and co-debug system, intended for fault injection test [12].

[1]Synopsys Verify Tools. http://www.synopsys.com

[2]Xilinx XTMRtool. http://www.xilinx.com

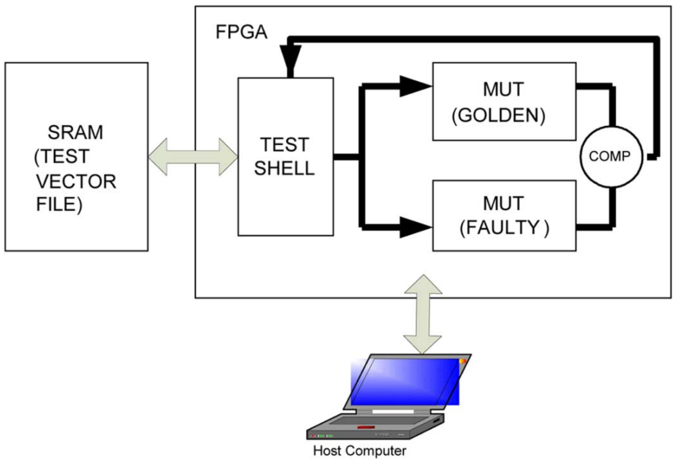

Fig. 1. FT-UNSHADES basic scheme.

FT-UNSHADES is based on a dedicated hardware platform and a specific software tool devoted to the control of the test injection campaign.

The hardware platform is based on two FPGAs. The first one, called Control FPGA or CFPGA, provides communication between the hardware platform and the system host. The second one, called System FPGA or SFPGA (a Xilinx XC2V8000 device in the current version of FT-UNSHADES), hosts the system under test. In addition, three 24-MB SRAM memories are used to store the input test vectors. The exact FPGA and family is not important. A Virtex II device is used for its large capacity.

On the software side, a system framework has been developed using a simple test language which defines the test injection campaign and helps in the analysis of the results. After a fault injection campaign, a fault database is produced containing the necessary information to analyze the fault activity. In addition, the designer can select a step-by-step execution of the experiments stored in the fault database, making possible to trace the fault propagation path.

The circuit to be tested is prepared using software routines integrated into the Xilinx standard design flow. Three actions are required:

- Input test vectors are obtained from an HDL simulator. Then they are properly formatted and compressed.
- The circuit to be emulated is synthesized using the Xilinx Virtex II design flow to obtain the Module Under Test (MUT) using any available synthesis tool. After synthesis, formal verification checks are performed in order to guarantee RTL matching between Xilinx and ASIC vendor netlists.
- The MUT is encapsulated to build the so-called Emulation Block (EB).

The structure of an EB is schematized in Fig. 1: two identical copies of the MUT (named Golden and Faulty, respectively) are instantiated as black box components along with a control unit called the Test Shell. During the fault injection campaign a comparator will detect discrepancies (if any) between the corresponding outputs at the Golden and Faulty instances.

The EB is implemented for the target FPGA and downloaded into the SFPGA, to start the emulation.

The Test Shell is a simple hardware which performs four tasks:

1) It allows the loading of the (compressed) input test vector file to the internal SRAM memories of the FT-UNSHADES hardware platform.
2) It decompresses and synchronously presents to the MUTs the test vectors.
3) Once the emulation process starts, it controls the emulation clock until the clock cycle for the fault injection is reached. Then, after the fault is injected it allows cycle-by-cycle propagation of the fault until a discrepancy is observed between the corresponding outputs of the Golden and Faulty instances, or until the test vector set is exhausted.
4) Finally, in every instance, it controls the dialog between the FT-UNSHADES platform and the host.

III. Injection Process

A. Fundamentals of the Approach

FT-UNSHADES injects faults in a very interactive and non intrusive way; by means of partial reconfiguration techniques, the selection of the target clock cycle and register can be freely specified. FT-UNSHADES injects faults by read-modify-write techniques using the configuration bits of the SFPGA [12].

FT-UNSHADES allows a selective fault injection campaign, restricting the fault injection to a given subcircuit of a hierarchical design. To this end it takes advantage of the back-annotation information generated by the Xilinx design flow. Using this information the register candidate to be flipped can be identified by its hierarchical path name within the design structure. Hence, it is possible to associate the high level description of the flip-flop to its geographical position in the SFPGA. For example, the Xilinx standard design flow generates the *logic allocation* file which contains information like

$$\text{Bit 0x005a0400 5358 Block} = \text{SLICE\_X84Y60 Latch} = \text{YQ}$$
$$\text{Net} = \text{SEU\_MUT/U\_ALL/U\_RAM/sfr\_dph(1)}.$$

The above data states a relationship between a register location and its net name, indicating that flip-flop $\text{SEU\_MUT/U\_ALL/U\_RAM/sfr\_dph(1)}$ was mapped to the internal flip-flop *labelled Y*, at *row 84* and *column 60*. The numbers in the line provide the necessary information to access this particular register content through the configuration circuit of the SFPGA. Note that the net name is represented by its post synthesis hierarchical path, given by the chain of instance labels which identifies the register in the high level description of the circuit. Classifying the nets by their hierarchical path, it is possible to selectively attack a subset of flip-flops that belong to the same subcircuit within the design structure.

B. Proposed Testing Procedure

A testing procedure is now proposed to assess, in a systematic way, the effectiveness of the inserted protections against SEUs, and the quality of the test vectors (QTV). The QTV to be defined below provides information about how the test vectors can propagate the effect of a SEU to the primary outputs.

The proposed testing protocol can be outlined as follows:

1) Define a fault injection campaign over the unprotected version of the design under test. Faults are randomly applied over the complete set of registers. Some faults will propagate to the primary outputs of the design. The percentage of detected faults (defined here as the QTV) represents the circuit dynamic sensitivity to SEUs with the current test vectors. The user might change the test vectors with the objective of increasing this percentage as much as possible. Systematic fault injection in all registers and cycles is possible, but circuit size and test duration would normally make this prohibited. A systematic campaign will last F*C*C clock cycles, where F is the number of registers, and C is the number of test vector clock cycles.
2) In the second stage of the testing procedure, internal registers are clustered into subcircuits. Applying the test vectors obtained in the first stage of the testing procedure, once again, to the unprotected version of the circuit under test, the percentage of detected faults per subcircuit is obtained, which represents the subcircuit sensitivity to SEUs. This information may be used to decide which subcircuits in the design should be protected.
3) In the third stage, a fault injection campaign is applied to the protected version of the circuit to detect errors in the protection scheme. The results will also reveal possible improper collapses of redundancies made by the synthesis tools.
4) Finally, the FT-UNSHADES platform allows multi-bit fault injections, that is, simultaneous bit-flips (modeling multi-bit upsets or MBUs) at different registers.

In space, radiation or particle hits may produce simultaneous bit-flips in neighbor registers, which can be modeled as a MBU. The ability of FT-UNSHADES to keep the hierarchical structure of a design allows it to identify neighbor registers which may be excited by a MBU in a fault injection campaign. Not only MBUs robustness will likely be a requirement for space applications in the future, the design sensitivity to MBUs also reveals the presence of internal structures which were not properly protected. For example, a MBU injection campaign reveals that a pure straightforward pipeline protected with TMR is less robust against radiation when it is only voted at the end. This result cannot be extracted from a conventional SEU injection campaign.

IV. LEON2 Case Study

A. Case Study Definition

FT-UNSHADES has been used in a fault injection campaign over three versions of the well-known Leon2 processor.[3] The first version is the Leon2 netlist produced by the toolbox provided by Gaisler Research. The second version is the same processor of version 1 protected with the XTMRTool . The third version is the original Leon2-FT licensed by the European Space Agency, which was protected using several *ad hoc* techniques. The Leon2 core has been defined with the same configuration options for the three versions.

[3]http://www.gaissler.com

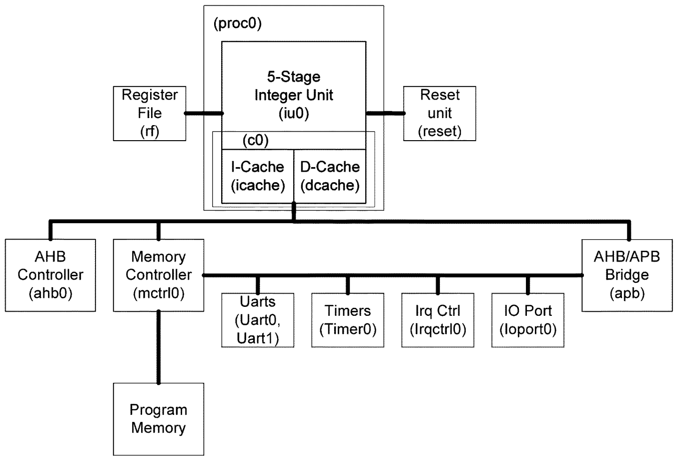

Fig. 2. Structure of the Leon2.

TABLE I
RESULTS OF THE SYNTHESIS PROCESS FOR EACH LEON2 VERSION

	n FF	4LUT	Multipl.	clk rate (ns)	BRAM
Leon2	5222	15249	2	23.748	32
Leon2Xtmr	13854	49845	6	23.916	84
Leon2-FT	13370	22542	2	23.820	34
Test-Shell	1118	2312	0	118.765	6

TABLE II
DETECTED FAULTS WITH THE UNPROTECTED VERSION OF LEON2

INJECTED FAULTS	100	500	1000	5000	10000	2000000
% OF DETECTED FAULTS	24	20	18.7	17.0	17.0	16.9

Fig. 2 shows the hierarchical structure of the Leon2 processor. It is a 32-bit synthesizable processor core based on the SPARC V8 architecture. The core is highly configurable, and particularly suitable for system-on-a-chip (SOC) designs. The Leon2 processor was designed under contract from the European Space Agency and the full source code is available under the GNU LGPL license. A description of the internal structure of the Leon2 and its primary inputs and outputs can be found in .

Table I represents the results of the synthesis process of each Leon2 version, showing how size increases in the protection process. (In Table I, LUT stands for Look-Up Table and BRAM for Block RAM.)

The set of test vectors used for the circuit is based on the ROM content for the production test, which is distributed with the processor netlist. It guarantees that every sub-block inside the processor will be activated. The test has 275 000 vectors.

B. Testing Results

Following the proposed testing procedure, in the first step a fault injection campaign is applied to the unprotected version of Leon2. Table II shows the percentage of detected faults.

According to Table I, a constant percentage of detected faults is reached at approximately 5000 injections, showing that the number of visible faults with the current test vectors is only 17%, out of a theoretical 100%. It is a representative figure for the QTV. This index may be used to predict how effective the radiation test would be with these test vectors. The designer could decide to improve the QTV, if he considers that this figure is not good enough. At the same time, this table provides the approximate number of injection cycles required to reach a good level of confidence in the results produced by a fault injection campaign.

In the second step of the testing procedure, different subcircuits are selectively subjected to a fault injection campaign in order to identify sensitive zones of the design. Table III shows the results obtained by systematically testing every internal subcircuit of the design. Ten thousand faults were injected in every subcircuit (except for the *reset* module, where the number of injected faults was reduced to 1000). This table shows the number of registers of the subcircuit (second row), the number of detected faults (third row), the time spent in the campaign in seconds (fourth row), the mean and max number of clock cycles elapsed between the fault injection and its detection (fifth and sixth row, respectively), and the primary output where the fault was detected (seventh and subsequent rows, one per primary output. Primary outputs are described in).

Fig. 3 shows a graph derived from Table III. The sensitivity against SEUs for a given subcircuit is defined as the percentage of injected faults detected in the fault injection campaign applied to that subcircuit. In Fig. 3, the sensitivity of every subcircuit is shown along with its relative size in terms of number of registers. According to Fig. 3, the *reset* subcircuit presents the highest sensitivity to SEUs. Therefore, for a safe operation of the circuit, despite its small relative area, the reset subcircuit should be considered as critical and, therefore, it should be protected. On the other hand, the subcircuits called *ahpb* and *timers s*how the lowest sensitivity to SEUs. With this information designers can decide about their proper level of protection, if any. The activity of the output signals is also valuable information: if an output is not affected by the injected faults, its corresponding driving circuitry could also be left unprotected. This is not the case with the Leon2 processor, as shown in Table III.

In the third phase of the testing procedure the same fault injection campaign was launched for the Leon2-XTMR and the Leon2-FT. As expected, these circuits did not show any malfunctioning at their output signals, showing that both circuits are properly protected.

In the fourth stage of the testing procedure the fault injection campaign was repeated in the two protected versions of the Leon2 processor but, in this case, each injected fault was a simultaneous multi-bit flip in randomly selected registers. Both designs did not show any malfunctioning in their primary outputs for a fault injection campaign with three simultaneous bit-flips. When the campaign was repeated with five simultaneous bit-flips, some faults propagated to the primary outputs where they were detected as shown in the graph shown in Fig. 4. After a detailed analysis of every faulty experiment, all the detected faults were found to be caused by a simultaneous attack to the same FF but at different clock domains, which is coherent to a TMR protection scheme.

From the results of this case study, we conclude that the XTMR and the FT versions of the Leon2 processor were properly protected. The case study also showed that the test vectors,

TABLE III
RESULTS OF THE FAULT INJECTION CAMPAIGN FOR THE ORIGINAL LEON2 PROCESSOR WITHOUT PROTECTIONS.
(THE DESCRIPTION OF THE LEON2 SUBCIRCUITS AND PRIMARY OUTPUTS CAN BE FOUND IN)

Subcircuit	top	ahpb	c0	ic	ioport	iu	mctrl	proc	reset	rf	timers	uarts
#Registers	2279	84	266	166	90	980	131	1447	8	64	127	217
time (s)	2590.33	2592.44	4109.76	2634.06	2632.17	2950.76	2569.48	2516.93	255.96	3424.66	2638.15	2620.16
Faults Det	1704	102	2780	2038	215	1717	3089	1704	809	0	119	1700
Mean (cyc)	26690.8	8.5	1152.8	62.5	178404	26063.1	93762.5	14594.0	3	0	171082.3	168239.0
Max (cyc)	362337		224519	12545	360357	325256				0	363594	364879
Errorn	11	0	0	0	0	19	0	14	0	0	0	0
Address	566	86	1472	1781	3	762	592	839	419	0	99	2
Datao	797	15	1263	149	1	923	1401	833	28	0	20	864
Dataen	37	0	0	0	0	0	629	0	8	0	0	0
Ramon	252	83	415	377	1	322	678	329	508	0	98	0
Ramoen	240	83	397	377	1	314	478	321	481	0	98	0
Rwen	36	0	0	0	0	0	610	0	2	0	0	0
Román	15	0	33	24	0	37	5	22	6	0	0	0
Iosn	11	0	0	0	0	0	246	0	0	0	0	0
Oen	202	83	365	327	1	253	273	290	485	0	98	0
Read	9	0	31	1	0	11	77	15	14	0	0	0
Writen	26	0	0	0	0	0	492	0	2	0	0	0
Pioo	152	1	0	0	211	0	0	0	0	0	0	834
Pioen	65	0	0	0	0	0	0	0	9	0	0	0

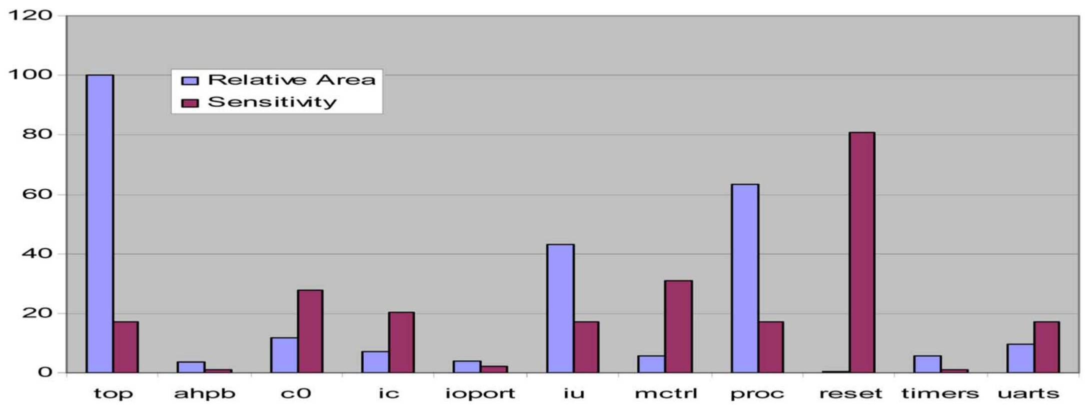

Fig. 3. Relative area and Sensitivity against SEUs of the subcircuits of the unprotected Leon2 processor.

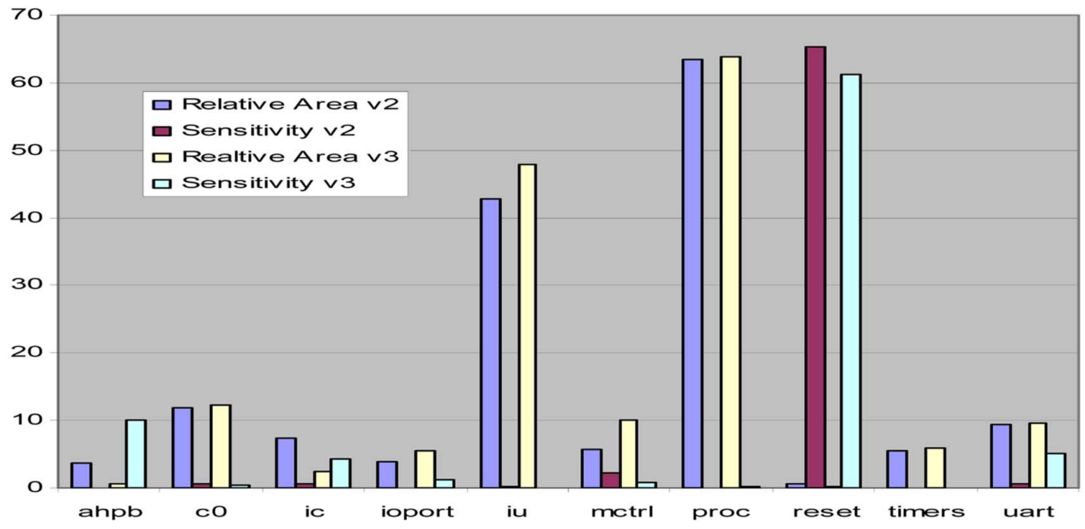

Fig. 4. Relative area and sensitivity for 5 simultaneous faults with the Leon2-Xtmr(v2) and the Leon2-FT(v3) versions.

although adequate for production test, would lead to a low percentage of detected faults in a radiation test, pointing out that a more exhaustive test vector database would be required for this case. Finally, the study also showed that different subcircuits inside the Leon2 processor could be designed with different levels of protection.

V. Conclusion

Fault injection is the standard procedure to assess the quality of the protection of a circuit against single event upsets (SEUs). For large circuits only emulated fault injection based on an FPGA provide test results in a reasonable time. This paper presents FT-UNSHADES, a hardware emulator which uses partial reconfiguration techniques on a Xilinx FPGA to provide fast, nonintrusive, fault injection emulation. Using FT-UNSHADES, a testing procedure has been proposed to help the designer: 1) to determine the quality of the test vectors, 2) to determine an adequate number of injected faults, 3) to assess the quality of the protection against SEUs, 4) to assess the level of criticality of different subcircuits and, finally, 5) to asses the quality of the protection against multi-bit flips. As a result, FT-UNSHADES helps designers to select which parts of the design could be left unprotected. FT-UNSHADES has been applied to three different versions of the Leon2 processor. It has shown that the reset subcircuit of the Leon2 processor is the most sensitive part of the design. It has also shown that the test vectors used for production test are not adequate for radiation test. Finally, the two protected versions of the Leon2 processor showed to be properly protected.

References

[1] J. A. Clark and K. Pradhan, "Fault injection – a method for validating computer system dependability," *IEEE Computer*, vol. 28, no. 6, pp. 47–56, Jun. 1995.

[2] J. Boué, P. Pétillon, and Y. Crouzet, "MEFISTO-L: A VHDL-based fault injection tool for the experimental assessment of fault tolerance," in *Proc. Fault-Tolerant Computing Symp.*, 1998, pp. 168–173.

[3] L. Antoni, R. Leveugle, and B. Feher, "Using run-time reconfiguration for fault injection in hardware prototypes," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Syst.*, 2000, pp. 405–413.

[4] R. Leveugle, "Fault injection in VHDL descriptions and emulation," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Syst.*, 2000, pp. 414–419.

[5] R. Leveugle, "A low-cost hardware approach to dependability validation of IPs," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Syst.*, 2001, pp. 242–249.

[6] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "FPGA-based fault Injection techniques for fast evaluation of fault tolerance in VLSI circuits," in *Proc. 11th Int. Conf. Field Programmable Logic and Applications, FPL 2001*, Belfast, U.K., Aug. 2001, pp. 493–502.

[7] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante, "Exploiting circuit emulation for fast hardness evaluation," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 6, pp. 2210–2216, Dec. 2001.

[8] L. Antoni, R. Leveugle, and B. Fehér, "Using run-time reconfiguration for fault injection in hardware prototypes," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Syst.*, 2002, pp. 245–253.

[9] A. Parreira, J. P. Teixeira, A. Pantelimon, M. B. Santos, and J. T. Sousa, "Fault simulation using partially reconfigurable hardware," in *Proc. 13th. Int. Conf. Field-Programmable Logic and Applications (FPL)*, Sep. 2003, pp. 839–848.

[10] C. López-Ongil, M. García-Valderas, M. Portela-García, and L. Entrena, "Techniques for fast transient fault grading based on autonomous emulation," in *Proc. Design and Test in Europe Conf. 2005*, Munich, Germany, Feb. 2005, pp. 308–309.

[11] M. A. Aguirre, J. N. Tombs, V. Baena, F. Muñoz-Chavero, A. Torralba, A. Fernández-León, and F. Tortosa, "FT-UNSHADES: A new system for SEU injection, analysis and diagnostics over post synthesis netlist," in *Proc. NASA Military and Aerospace Programmable Logic Devices, MAPLD*, Washington, D.C., Sep. 2005.

[12] M. A. Aguirre, J. N. Tombs, V. Baena, J. L. Mora, J. M. Carrasco, A. Torralba, and L. G. Franquelo, "Microprocessor and FPGA interfaces for in-system co-debugging in field programmable hybrid systems," *Microprocess. Microsyst.*, vol. 29, no. 2–3, pp. 75–85, 2005.

[13] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2147–2157, Dec. 2003.

[14] M. French, P. Graham, M. Wirthlin, and L. Wang, "Cross functional design tools for radiation mitigation and power optimization of FPGA circuits," presented at the 6th Annu. NASA Earth Sci. Technol. Conf., ESTC2006, University of Maryland, College Park, MD, Jun. 27–29, 2006.