An Experimental Analysis of SEU Sensitiveness on System Knowledge-based Hardening Techniques

O. Ruano, P. Reyes, J.A. Maestro
Universidad Antonio de Nebrija
Madrid, Spain

L. Sterpone
Politecnico di Torino
Torino, Italy

P. Reviriego
Universidad Carlos III de Madrid
Madrid, Spain

Abstract- **Logic Soft Errors caused by radiation are a major concern when working with circuits that need to operate in harsh environments, such as space or avionics applications, where soft errors are traditionally referred as Single Event Effects. In this paper, system knowledge-based hardening techniques using recursive structures for the implementation of moving average filters that provide protection against Single Event Upsets are evaluated through two fault injection systems based on simulation and emulation respectively. Fault injection campaigns show that system knowledge-based redundancy techniques can achieve the same level of dependability as standard redundancy techniques, such as Triple Modular Redundancy, while having optimal cost.**

I. INTRODUCTION

Electronic devices are sensitive to radiation that may happen both in the space environment and at the ground level. The continuous evolution of manufacturing technologies makes integrated circuits more sensitive to radiation effects, such as Single Event Upsets (SEUs). The main causes are related to the shrinking coupled with voltage scaling and high operating frequencies correspond to significantly reduced noise margins, which make circuits more sensitive to radiation, as well as to other phenomena such as crosstalk or internal noise margins that cause transient faults. For these reasons several researchers have done investigations both to develop fault-tolerance methods in order to mitigate SEUs and to analyze their influence in the electronic system deployed for safety or critical mission applications.

Redundancy-based techniques are widely applied to provide protection against SEUs effects. These techniques use additional hardware components or additional computational time to detect the presence of SEUs modifying the expected circuit operations and masking SEUs propagation to the circuit's output. When fault masking is mandatory, designers may resort to Triple Modular Redundancy (TMR) approach. The basic concept of the TMR architecture is that a circuit can be hardened against SEUs by designing three copies of the same circuit and building a majority voter on the outputs of the replicated circuit. Hardening a design through TMR implies severe overheads since all the hardware logic resources needed for the TMR circuitry are triplicated.

In this paper, the effectiveness against SEUs of hardening techniques based on recursive structures are evaluated using two different fault injection platforms: a simulation-based approach working on a VHDL model of the circuit under test, and a new emulation-based fault injection technique able to perform injection campaigns in a fraction of time required by simulation-based approaches, while still supporting most of their positive features.

The main contribution of this paper is composed of two aspects. On one hand, it provides an effective demonstration of the fault tolerance capabilities that the system knowledge-based techniques offer while reducing area overhead versus standard TMR techniques. On the other hand, a novel fault injection approach using SRAM-based FPGA partial reconfiguration is compared with a simulation–based approach showing an increasing of the performance of two orders of magnitude.

Experimental results have been executed running several fault injection campaigns on different versions of the proposed system knowledge-based techniques. The achieved results show that the overhead introduced by the proposed techniques is reduced versus the standard TMR hardening technique, while the same degree of fault tolerance is provided. Furthermore, experimental analysis demonstrated that emulation-based fault injection has a speed-up better than the simulation-based technique.

The paper is organized as follows. Section 2 presents a background on the hardening techniques and fault injection platforms previously developed. Section 3 describes the two platforms whose results are compared. Experimental results about effectiveness and area cost of the evaluated circuit against TMR are illustrated and analyzed in Section 4. Finally, conclusions and future works are exposed in Section 5.

II. BACKGROUND

The problem of radiation on electronic devices has been traditionally addressed in literature.

A classic reference by J.F. Ziegler is offered in [1], where the basic physics of radiation effects is detailed. Different rates of errors at several terrestrial positions are described, providing a quantitative analysis of the radiation effects. One of the factors that measure the sensitivity of circuits to radiation is the error rate. Several works try to provide models for this error rate, in order to foresee the behaviour of the circuit in a particular environment. A Soft Error Rate computation algorithm is presented in [2], which can be applied to combinational circuits. The parametric waveform model is based on the Weibull function. Experiments show that the algorithm is linear in the number of nodes, and results are close to SPICE simulations.

A methodology to compute the effects of charged particle inducing delay errors (Soft Delay Errors) is presented in [3]. The different node sensitivity is computed in order to

1-4244-1161-0/07/$25.00 ©2007 IEEE

employ node hardening techniques, and therefore, increase the reliability of CMOS circuits. Techniques to detect and correct errors are very common too. The goal of such techniques is to mitigate the effects of radiation, both by detecting errors when they happen, and by trying to correct them, thus getting rid of the negative effect. In [4], the problem of Concurrent Error Detection (CED) is discussed in Burst-Mode machines. An enhanced duplication process is proposed in order to give a solution to this problem, showing an interesting saving in hardware. A technique to minimize the impact of soft errors in circuits is presented in [5]. Through the use of complementary pass transistor devices, those gates affected by SEUs are isolated, and therefore their negative effect is removed. This is achieved with limited area, delay and power overheads.

In the memory research field, a new BIRA (built-in-redundancy-analysis) algorithm is presented in [6], in order to allocate 2D redundancy using 1D local bitmap. The proposed experiments offer a high repair rate, close to the existing optimal algorithms. In [7], the problem of sub-65nm designs is described. Since it is stated that classical fault-tolerance techniques for soft error detection are expensive, a recently developed Built-In-Soft-error-Resilience (BISER) technique is proposed, which seems effective for soft error blocking or detection. When the validation of the effectiveness of electronic circuits hardening techniques against SEUs is considered, several approaches should be mentioned.

Several works have already explored the use of FPGAs for speeding-up fault injection of permanent faults [8][9]. In [10] the extension to the injection of transient fault is proposed, where an instrumented model of the system under analysis is exploited. Although very efficient in reducing the CPU time needed for evaluating high numbers of faults, it mandates the introduction of fault-injection-oriented features in the model, and therefore it cannot be exploited in those applications where intellectual property (IP) cores coming from third parties are used, for which the model's source code is not available.

The authors of [11] and [12] proposed alternative approaches to inject faults while emulating the system using FPGA devices, where partial reconfiguration is employed to perform the injection of SEUs. The most important benefit stemming from these approaches is that the source model of the system under analysis is not needed, while only a netlist suitable for being placed in an FPGA is required. On the one hand, the intellectual property of the IP core is preserved; on the other hand, the SEE analysis is performed on the very same model that will be deployed in the final system, differently from [11] where the model has to be changed to insert the fault-injection-oriented features. The major drawback of [12] is the speed: being based on JBits [14] and on a slow communication interface between the board carrying the emulated system, and the host computer managing the experiment, the number of faults that may be injected is quite limited. The authors reported that about 100 msec are needed for injecting, and classifying the effect of one SEU.

The major drawback of [13] is the portability: a specific custom-developed board is needed to perform fault injection. The system is very time-efficient (44 msec are needed for injecting and classifying the effect of one SEU), but it may be quite expensive to implement.

III. Overview on the Fault-Injection platform

In this section, the descriptions of the SEUs simulation and emulation platforms used to study the effectiveness of the electronic circuits are exposed.

A. *Simulation-based platform*

The purpose of this platform [15] is to help designers to predict and explore potential weak points on ICs sensitive to hazards in the early design cycles without the need of developing a prototype. It can also be used to assess the effectiveness of a given protection technique. The platform is composed of the SST simulator developed by the ESA Data Systems Division and Matlab. A commercial HDL simulator (i.e. ModelSim) is used to run the simulations. For a given circuit under test, we would generate a number of testcases in terms of the corresponding input and output data using Matlab. We would also generate a number of test configurations in terms of the soft errors inserted using the SST. The testcases would be designed to fully test the circuit functionality and performance while the test configurations would ideally reflect the soft error environment that is expected for the device operating conditions. Then, combinations of both can be easily tested by just selecting the appropriate input and output data files and test configuration file. In fact, with a simple script, the testing of all relevant combinations can be easily automated.

As it can be seen in figure 1 three independent modules make up the platform.

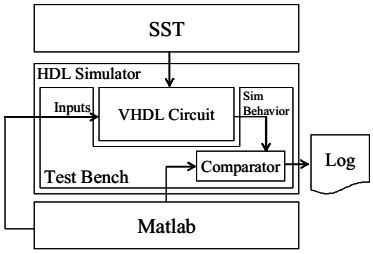

Fig. 1 Scheme of the Simulation-based platform used

1. *SEUs Simulation Tool (SST)*: This component consists of a set of modules used to prepare the environment to generate soft errors in both sequential and combinational logic.
2. *HDL Simulator*: This module is in charge of holding the circuit to test, and performing a simulation at the design stage. The description of the environment is divided into two parts: the circuit and the test bench. In particular, the test bench will produce the different test scenarios for the circuit (based on the input values provided by the Matlab module), will capture the circuit outputs, and will compare them with the expected results (also provided by Matlab). In case both are different, that will indicate an error, which will be logged in the system for further study. This

environment is generic (independent of the circuit behaviour) for circuits devoted to signal processing (or at least a significant part of them). It is also flexible in the way that it is straightforward to generate different input signals to test the circuit operating in several environments. For other kinds of circuits (e.g., controllers), another application rather than Matlab would be designated to hold the golden data.

3. *Matlab*: This module compares the theoretically correct behaviour of the system with the actual outputs produced by the HDL simulator. It has the advantage that the Matlab code does not need to reflect the actual circuit implementation, it only needs to be functionally equivalent. This facilitates the use of a single Matlab model to explore different implementation alternatives. The difference between both behaviours will indicate the presence of a SEU, what will trigger the mechanism to detect the source of such an error.

B. Emulation-based platform

The emulation-based platform is composed of the following modules: a host computer; an FPGA board equipped with a Virtex II-Pro device, and a serial communication link to the host computer. The host computer is primarily used for configuring the Virtex-II Pro and for the generation of a fault location list. However, during the execution of the fault injection experiment, its only purpose is to provide a user-friendly interface to run the fault-injection experiments and to collect the results in terms of fault-effect classification.

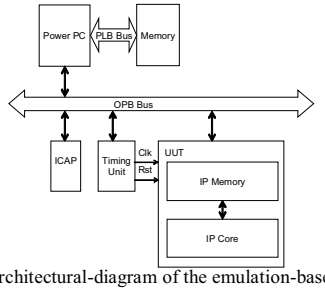

Fig. 2 Architectural-diagram of the emulation-based fault injection approach

The FPGA board is the core of the fault-injection system and its layout is depicted in Fig. 2. It is composed of four components interconnected by an On-chip Peripheral Bus (OPB):

- Timing Unit: it drives the UUT clock and reset. The clock of the UUT has the same frequency of the FPGA device layout. A port connected to the OPB Bus defines its functionality.
- Unit Under Test (UUT): it is the circuit under test and it may consist of an IP core and an own memory. The IP core's input and output ports are connected to the OPB Bus while the reset and clock signals are connected to the Timing Unit.
- ICAP: it is the Internal Configuration Access Port provided by last generations of Xilinx FPGAs. It allows the access to the FPGA configuration memory through an internal port in order to perform partial reconfiguration without the support of an external hardware. For the purpose of this work, we configured the ICAP in such a way that it is able to access to all the memory elements (such as Flip-Flops or Latches) of the UUT IP core.
- PowerPC microprocessor: it is hardwired in the FPGA device and it has two functionalities. At first, it performs the fault injection of SEUs within the memory elements of the UUT IP core through the execution a fault-injector algorithm. Latter, it communicates the fault-injection experiment results to the host computer through a serial communication link.

The serial communication link is supported by a RS-232 cable that connects the FPGA board to the host computer.

The execution phase of the used fault injection approach is performed by several procedures included within the fault-injector algorithm as illustrated in the Figure 3.

```
/*Pre-running*/
FI_initialization()
{CC,GO}=Golden_Run_UUT()
/*Campaign*/
for number of injected faults NF
{
reset_UUT()
{FT,FL} = random (CC,FL)
run_UUT(FT)
Value= read_value(FL)
Inject_SEU(FL,!Value)
FCL=monitor_UUT(CC,GO)
}
/*Fault Injection Results*/
communication_host(FCL)
```

Fig. 3 The emulation-based fault injection algorithm

The algorithm is executed by the PowerPC and it consists of three parts: pre-running, campaign and fault injection results. The Pre-running starts the fault injection experiment. At first, it loads within the PowerPC memory the test patterns that will be applied to the UUT and initializes the UUT IP memory (i.e. if the IP core is a processor the UUT IP memory will be loaded with the desired program). Secondly, it performs a golden run of the UUT storing the total number of Clock Cycle (CC) and the Golden Output (GO) produced. The Campaign performs the fault injection of the selected number of faults (NF). The following steps are executed for the injection of each SEU:

1. The procedure reset_UUT() resets the UUT and configures the Timing unit in such a way that it sends a reset to the UUT.
2. A fault injection time (FT) and a fault location (FL) are randomly selected considering the number of clock cycle CC and the set FL available.
3. The procedure *run_UUT*(FT) stars the execution of the UUT until the clock cycle FT is reached. This operation is performed by configuring a Timing Unit's terminal counter at the FT value.
4. The procedure *read_value*(FL) reads the value of the fault location FL. This procedure reads directly the content of the flip-flop or latches from the configuration memory through the usage of the ICAP port.
5. The procedure *Inject_SEU* (FL,!Value) partially reconfigures the bitstream of the FPGA writing the opposite value within the content of the flip-flop or latches identified by FL. Therefore a SEU is injected in the considered fault location.

6. The procedure *monitor_UUT*(CC,GO) continues the execution of the UUT until CC is reached. During the execution, it monitors the UUT output ports reading the data on the RS-232 interface and comparing their value with the UUT golden outputs. It finally updates a fault classification list (FCL) with the results obtained by the fault injection and classifying each injected SEU as silent, if the output produced by the UUT are equal to the GO; wrong answer, if a mismatch was detected.

When the fault injection campaign is concluded, the PowerPC communicates the fault injection results to the host computer returning the FCL through the procedure *communication_host* (FCL).

IV. Experimental Results

In this section, we first give an overview of the evaluated system knowledge-based techniques based on FIR filters used to compare the protection effectiveness results provided by the simulation and emulation-based platforms. Finally, fault injection results are presented and commented.

A. The case study: system knowledge-based hardening techniques

To come up with optimal Single Event Effects (SEEs) protection techniques, we need to take the requirements of the application in which the circuit is used into account. For the specific case of moving average filters in recursive form, the case study of this paper, a further explanation of the protection techniques tested can be found in [16]. These developed techniques offer a set of possible alternative solutions to TMR and they are specifically designed for moving average filters.

Moving average filters are one special type of FIR filter which shows some interesting properties for implementation and they are used in many applications such as industrial controls or automotive. They perform the following operation [16].

$$y[n] = \frac{1}{N}\sum_{i=0}^{N-1} x[n-i] \quad (1)$$

Where x[n] is the input signal, y[n] the output and N is the number of input samples that are processed each time to compute each y[n] sample and, normally, its value is a power of two, so that the division can be implemented with a shift operation. In this case, the filter needs only adders.

A more efficient implementation can be derived by rewriting (1) as follows:

$$y[n] = y[n-1] + \frac{1}{N}(x[n] - x[n-N]) \quad (2)$$

In this case, only two adders are needed irrespective of the value of N. In fact, this implements the FIR filter using an Infinite Impulse Response (IIR) structure.

A first look at the effect of SEUs on both structures shows that in the case of the more efficient IIR implementation, SEUs in the delay line or in the accumulator can cause errors in the output that will persist until the filter is reset. Therefore, SEEs can influence the choice of the implementation structures for digital filters, and suggests the interest of smart protection techniques. Depending on the application requirements a number of protection techniques have been proposed in [17].

The first one consists in adding protection through a decimated filter. If the application can tolerate occasional errors on the output of the filter, the computation of the output can be done in parallel by another structure added to the filter, for the sake of comparison. Obviously, if this added structure is a replica of the filter itself, we would be doubling the complexity of the system. To avoid this situation, this parallel structure will be implemented with a *decimated* filter, which has a structure simpler than a regular filter, with the drawback that it only computes the right output one out of N cycles.

The second scheme consists in adopting a double parity architecture. One alternative to using TMR in all registers consist of computing a two-dimensional parity, where for each input value and each bit position it is computed a parity on two bits. These two sets of parity bits, form the accumulated parity of the circuit, which is constantly being updated. Dynamically, each time a new value reaches the circuit, this parity bits are re-checked and compared with the accumulated values. Therefore, single SEUs will be undoubtedly identified and corrected [17].

B. Fault Injection Results

In this section we present the experimental results obtained from the fault injection campaigns on several version of a FIR filter implemented with IIR structure (see Eq, 2).

Four different FIR filters have been implemented:

1. *IIR_Basic*: it is the plain version of the Infinite Impulse Response filter.
2. *IIR_RedTec*: it is the FIR Filter hardened using a decimated filter.
3. *IIR_SystemKnowledge*: it is the FIR Filter hardened adopting a two-dimensional parity architecture.
4. *IIR_TMR*: it is the FIR Filter hardened adopting standard Triple Modular Redundancy (TMR).

In order to compare the simulation-based versus the emulation-based fault-injection we synthesized the circuits using an ASIC and FPGA –oriented synthesizers.

The characteristics of the implemented FIR Filters on the SRAM-based FPGA are illustrated in Table I where the number of used Flip-Flops (FFs), Slices and Look-Up Tables (LUTs) are reported. The recursive solutions developed need less resources than the standard TMR technique. In particular, the IIR_SystemKnowledge technique drastically reduce the number of needed FFs while introducing a minimal overhead of used LUTs.

TABLE I
FPGA-based Synthesis Characteristics of the Implemented FIR Filters

Circuit	Slices [#]	FFs [#]	LUTs [#]
IIR_Basic	82	140	43
IIR_RedTec	108	158	212
IIR_SystemKnowledge	178	196	287
IIR_TMR	290	420	260

The equivalent results in terms of FFs and total gates

obtained using an ASIC synthesizer with a 0.25µm TSMC library are shown in Table II. As it can be noticed, the number of FFs is exactly the same as the one obtained for the FPGA-based synthesis. Moreover, the area overhead introduced by the recursive solution is drastically reduced if compared with the standard TMR technique.

TABLE II
TSMC 0.25 µM AND 50-MHZ SYNTHESIS

Circuit	FFs	Total gates
IIR_Basic	140	866
IIR_RedTec	158	1300
IIR_SystemKnowledge	196	2151
IIR_TMR	420	2962

In order to implement the emulation-based fault injection platform described in section 3.2, we used a Xilinx Virtex-II Pro Platform SRAM-based FPGA [18] embedding a PowerPC 405 [19]. The fault injection campaigns have been performed injecting randomly SEUs within the FFs used by the circuits. The workload for each injection was of 1,000 input stimuli.

TABLE III
FAULT GRADING

Circuit	Injected SEU	Wrong Answers	Golden Outputs
IIR_Basic	50,000	50,000	0
IIR_RedTec	50,000	21	49,979
IIR_SystemKnowledge	50,000	0	50,000
IIR_TMR	50,000	0	50,000

The obtained fault classification is shown in Table III, where the number of injected faults and the fault grading are reported. It can be noticed that the IIR_SystemKnowledge solution provides complete protection against SEUs affecting FFs of the delay line, as the TMR solution. Comparing this results with the ones obtained using software-based fault injection (see [16] for more details) the next points can be concluded:

For the RedTec protection technique, the average cycles used for the filter to correct its behavior after the SEU injection is around 16 cycles.

For the SystemKnowledge Technique, the filter is 100% effective againts SEUs, similar to TMR.

We have contrasted the emulation-based fault injection platform with the SST simulator tool [15] in order to double-check the efficiency of the protection techniques and compare the results obtained with both fault injection platforms.

The SST tool corroborates the protection efficiency obtained with the emulation-based platform, what proves that the different redundancy techniques are valid.

Since the results from the two platforms are similar and comparable, we can conclude that any of them could be used to perform a fault injection analysis of a given system. As a time estimation, the developed emulation platform needs, on average, 3.95msec versus 671msec of the SST simulation-based approach.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have evaluated the fault tolerance capabilities of recursive-oriented hardening techniques versus the standard Triple Modular Redundancy (TMR) technique, using two different fault injection platforms: a simulation-based approach working on a software model of the circuit under test and a novel emulation-based fault injection approach. Experimental results demonstrated that the system knowledge-based redundancy techniques may achieve the same level of fault tolerance as the standard TMR approach while optimizing the circuit area overhead. Furthermore, as the protection effectiveness results from the two platforms give similar conclusions, it could be extrapolated that the two fault injection platforms are equally useful in order to perform fault injection analysis on any given type of systems, although an initial time comparison puts in perspective that hardware emulation is faster than software simulation.

As future work we plan to investigate the fault tolerance capability of recursive-oriented techniques applied on SRAM-based FPGAs when affected by SEUs within their configuration memory.

REFERENCES

[1] J.F. Ziegler, "Terrestrial cosmic rays", IBM Journal of Research and Development, vol 40, #1, 1996.

[2] R.R. Rao, K. Chopra, D. Blaauw, D. Sylvester, "An Efficient Static Algorithm for Computing the Soft Error Rates of Combinational Circuits", Proceedings of Design Automation and Test Conference, 2006.

[3] B. S. Gill, C. Papachristou, F. G. Wolff, "Soft Delay Error Analysis in Logic Circuits", Proceedings of Design Automation and Test Conference, 2006.

[4] S. Almukhaizim, Y. Makris, "Concurrent Error Detection in Asynchronous Burst-Mode Controllers", Proceedings of Design Automation and Test Conference, 2005.

[5] J. Kumar, M.B. Tahoori, "Use of pass transistor logic to minimize the impact of soft errors in combinational circuits", Workshop on System Effects of Logic Soft Errors (SELSE), 2005.

[6] T.-W. Tseng, J.-F. Li, D.-M. Chang, "A Built-In Redundancy-Analysis Scheme for RAMs with 2D Redundancy Using 1D Local Bitmap", Proceedings of Design Automation and Test Conference, 2006.

[7] S. Mitra, T. Karnik, N. Seifert, M. Zhang, "Logic Soft Errors in Sub-65nm Technologies Design and CAD Challenges", Proceedings of Design Automation Conference, 2005.

[8] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, "Fault Injection into VHDL Models: the MEFISTO Tool", Proc. FTCS-24, 1994, pp. 66-75

[9] S. A. Hwang, J. H. Hong, C. W. Wu, "Sequential circuit fault simulation using logic emulation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: Vol. 17, No. 8, Aug. 1998, pp. 724 -736

[10] K. T. Cheng, S. Y. Huang, W. J. Dai, "Fault emulation: A new methodology for fault grading", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No. 10, Oct. 1999, pp. 1487 -1495

[11] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, "Exploiting Circuit Emulation for Fast Hardness Evaluation", IEEE Transactions on Nuclear Science, Vol. 48, No. 6, December 2001, pp. 2210-2216

[12] L. Antoni, R. Leveugle, B. Fehér, "Using Run-time Reconfiguration for Fault Injection in Hardware Prototypes", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2000, pp. 405-413

[13] J. Tombs, M. A. Aguirre, "FT-UNSHADES", Microelectronics Presentation Day, 2004

[14] JBits 2.8, Xilinx, San Jose, CA, 2001

[15] D. Gonzalez-Gutierrez, "Single Even Upset Simulation Tool Functional Description", ESA Report TEC-EDM/DCC-SST2, July 2004.
[16] P. Reyes, P. Reviriego, J.A. Maestro and O.Ruano, "New Protection Techniques against SEUs for Moving Average Filters in a Radiation Environment" , IEEE Transactions on Nuclear Science, 2007 (in press)
[17] A.V. Oppenheim and R.W. Schafer, "Discrete Time Signal Processing", Prentice Hall 1999. ISBN: 0137549202.
[18] Xilinx Product Specification, "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet" DS083 v4.5, October 10, 2005
[19] Xilinx Reference Guide, "PowerPC Processor", EDK 6.1, September 2, 2003.