Design and Evaluation of a Hardware on-line Program-Flow Checker for Embedded Microcontrollers

M. Ottavi*, S. Pontarelli, A. Leandri, A. Salsano
Dipartimento di Ingegneria Elettronica
Università di Roma "Tor Vergata", Italy
**Electrical and Computer Engineering Department*
Northeastern University Boston (MA) USA
{ottavi, pontarelli, salsano}@ing.uniroma2.it

Abstract

This paper investigates the effects of a class of transient faults, the so-called Single Event Upsets, on the execution of programs in typical microcontroller architecture as can be found on a system on chip for embedded applications. It is observed that the consequences of targeting the registers used in the control flow can cause unexpected jumps of the program and consequent heavy effects on the results or the freeze of the microcontroller. A novel hardware based control flow checker is then introduced and implemented on an FPGA test bed together with the microcontroller core and fault injection circuitry. The FPGA implementation allows to dynamically and quickly injecting faults on the microcontroller whereas the results of the fault injection campaign allow to evaluate the fault coverage of the proposed method with a high degree of flexibility.

Keywords: Microcontrollers, Fault Injection, Program-flow checking, FPGA, Partial reconfiguration

1. Introduction

In recent years the continuous trend of increasing the integration to a system level has led to the introduction of more and more complex devices. New devices proposed by many suppliers are today implementing different functional cores such as microcontroller, FPGA, memory blocks (RAM, ROM, EEPROM, Flash), analog interfaces and more, and are commonly referred to as "system on a chip" (SoC). SoCs have typical application in embedded systems and therefore it is worth to consider their reliability with respect to the occurrence of possible faults. In particular a class of transient faults also known as Single Event Upset (SEU) is becoming more and more important with the reduction of the feature size of the transistors and the consequent reduction of their critical charge. In particular it has been shown that radiation at sea level can induce effects on a device modeled in most of the cases as an inversion of a logic value stored in a memory element and called "bit-flip". [18] notes that starting from the 0.13 micron process it will be observed a bit flip every 10 days for a 128 Mbyte SRAM at sea level.

The problem of reliability of SoC microprocessors with respect to the SEUs is closely related to what already studied for applications in harsh radioactive environments, such as space or nuclear plants and in particular to the usage of Commercial Off the Shelf Components for these applications [19]. However if the interest into using Commercial Off The Shelf (COTS) components in space application could be an interesting alternative to rad-hard components due to their low cost and high performances, in embedded computing the usage of rad-hard technologies would just be unfeasible for its uneconomical impact.

Although we are focusing on the transient faults it should be noticed that in general the effects of radiations are divided in permanent and transient faults, where the permanent faults can be considered an accelerated ageing and are related to the total time that a device spends under radiation (TID: Total Ionizing Dose) while the transient faults are instantaneous and are

Proceedings of the 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'06)
0-7695-2706-X/06 $20.00 © 2006 IEEE

related to the energy and charge of the impacting particles (SEU), also for this randomness of their occurrence in time SEU represent presently one of the most interesting fault types due to the radiation effects on integrated circuits [1][2].

Obtaining high reliability without the expensive rad-hard fault avoidance techniques requires the introduction of fault-tolerant techniques and therefore hardware and/or software overhead to obtain error detection and system reconfiguration. An approach based on hardware redundancy like a typical NMR (N-Modular Redundancy) could be straightforward; however its consequences on power consumption and area occupation and the consequent manufacturing cost induced us to pursuing the use of on-line testing methodology that is discussed in this paper with specific focus on control flow check.

Well-known fault injection techniques allow perturbing the normal function of the system under study to evaluate the SEU effects. The results of such experiments are an indispensable feedback to the designer of fault tolerant systems in order to implement efficient techniques of fault detection and system recovery or fault masking.

We applied fault injection techniques to an 8051 microcontroller and evaluated the kind of errors obtained during the execution of a set of test programs. The choice of the 8051 is in line with a very well established previous literature [3][4][14][15][17] and allows a common ground of comparison for the results, moreover the 8051 is a rather simple but at the same time diffused architecture and the studies on it can be easily extended to other microcontrollers. Since we did not target any specific SoC architecture, we simulated it on an FPGA platform that implemented the IP core of the 8051, the proposed control flow checking hardware and the specific fault injection circuitry. The usage of an FPGA based fast prototyping platform introduces advantages both in the speed of the simulations and in the high degree of accessibility to the inner hardware registers of the IP core as compared to software based fault injection strategies [17].

We observed that SEU events had different effects on the system depending on the time and the location in which they occur [3], we also observed that some bit-flips have no consequences on the program execution, because they occur before a valid value is written, while a bit-flip occurring in the data registers can give consequences on a little portion of the outputs of the test programs. However we also observed some bit-flips can cause critical effects on the test programs resulting in a complete freeze of the system where the 8051 is embedded. These errors are caused by bit-flips in the memory elements involved in the control of the program flow. Therefore, the checking of the control flow of a microcontroller assumes a very important role in order to avoid the total loss of the functions performed by the system.

The increasing diffusion of integrated IPs suggested to opt for hardware based flow control check as opposed to the more widely studied software based flow control check [15][16]. Hardware solutions with embedded IP cores of microcontrollers and FPGAs are now a very common product offered by vendors. The benefits of hardware based flow control check are both in performances and ease of implementation as with its introduction the flow control checker does not cause any perturbation on the normal execution of the program by introducing modification to the executed code. Moreover it will be shown that also the hardware overhead introduced with hardware checkers can be kept low by introducing signature analysis to obtain output compaction.

The goal of this paper is to introduce an innovative hardware flow control checker for embedded microcontrollers and to test its performances.

To expedite the analysis and to obtain a high degree of flexibility we also introduce and describe an ad-hoc developed hardware fast prototyping platform implemented on a Xilinx Virtex FPGA together with the 8051. The proposed solution is validated injecting SEU-like faults during the execution of the test programs.

The paper is organized as follows. In section 2 we discuss the SEU effects on program execution flow. Section 3 describes the structure of an instruction level signature analysis checker. Section 4 describes the FPGA implementation of an 8051 and its checker. In section 5 we show the test-bed used for the validation of the methodology and the obtained values of fault coverage are presented in section 6. Finally, conclusions are drawn in Section 7.

2 SEU effects on program execution flow

The SEU effect on the execution of a set of programs running on the 8051 microcontroller has been evaluated in terms of their sensitivity to the bit-flip injected [3] while the actual sensitivity of the 8051 hardware memory elements has been measured through the execution of a static test under radiation [4]. From the tests under radiation is derived the so called cross-section (eq. 1) which gives a probabilistic estimation of the number of particles needed to get a bit flip on a particular target.

$$\sigma_{SEU} = \#\ detected\ errors\ /\ particle\ fluency \qquad \text{(eq. 1)}$$

Instead, from the fault injection experiments it can be estimated the program error rate as a function of the number of injected bit flips leading to error in the execution of a given program normalized by the total number of injected bit flips (eq. 2).

$$\tau = \#\ detected\ errors\ /\ \#\ injected\ errors \qquad \text{(eq. 2)}$$

Therefore combining he previous two equations, the sensitivity to SEU of a program can be calculated by the product of the cross-section by the error rate to injected bit flips (eq.3):

$$\tau_{SEU} = \sigma_{SEU} * \tau \qquad \text{(eq. 3)}$$

The multiplication of the static cross-section by the error rate derived from fault injection represents a good estimator of the rate of detected errors normalized by the number of particles, which is the definition of the error rate under radiation for a given application. In this way, once the SEU static cross-section is measured from a suitable radiation ground testing experiment, the evaluation of error rates for different application programs can be done without exposing the circuits to radiation, significantly saving time and economic efforts. In [3] we proposed an initial estimation of τ obtained by using two fault injection methods that considered as targets of injection all the Special Function Registers of the microcontroller and the 128 bytes internal memory registers.
The first considered method injects faults by executing suitable pieces of code and is called CEU (Code Emulating an Upset) technique [5] [6] the second method instead uses an HDL model of the device [7], for which code modifications are implemented in order to virtually change run-time every part composing the device. The two models were in accordance (Table 1) and showed that the sensitivity to bit flips resulted strongly related to the executed program. Bit flip faults were injected run-time during the execution of two benchmark software applications: a bubble sort of an integer vector and a 6x6 matrix multiplication program.

	CEU Injection	VHDL Injection
Matrix Multiplication	48,8%	48,71%
Vector Sorting	26 %	22,14%

Table 1: Error rates factors obtained by fault injection for the two benchmark programs

The error rates reported could be considered as the set of all SEUs that, during test bench, produced a difference on an output port of the design, i.e. the failure set as in [8] [10]
A more detailed analysis of the results showed that the errors detected during the experimental phase can be related to different causes.
In particular we noticed that the occurrence of bit flips in the memory locations devoted to the program execution control like: Program Counter register, Data Pointer Transfer Register, Stack Pointer Register, Program Status Word Register, Stack Area, caused random jumps in the program flow execution. The effects of these random jumps depend on the time and on location in which the bit-flips occurred.
In general we deduced that the injection of bit flips in the registers devoted to the program control will cause more sensitive consequences as opposed to isolated effects on result integrity. Since the effects of errors in the flow control are more catastrophic and span from some erroneous results (result errors) to the total freeze of the microcontroller and the consequent lost of synchronization with the rest of the system it is embedded into (lost of sequence) we focused our attention to design a control flow checker and to perform further fault injection campaigns targeted to the specific memory registers involved in the control flow.

3. Proposed Architecture of Control Flow Checker

The proposed control flow checker is based on a hardware signature analysis technique to verify on-line the correctness of the operations executed by the microcontroller.
Signature analysis check is a well-known technique present in literature [12] [13]. The main feature of the checker is its complete independence from the system similarly to [20] but with a simple ad-hoc architecture instead of a watchdog processor. The major advantage of the architecture is that it does not require modifications to the code executed by the microcontroller nor to the microcontroller itself. To obtain this result with pursued the following approach:

1. We considered the microcontroller as a “black box” whose external observable signals are the address bus and the other I/O pins.
2. We modeled the behavior of the system analyzing the flow diagram of the executable code.

The signature analysis is applied to control the behavior of this “black box”. A hardware signature checker, which controls address bus and synchronization signals of the microcontroller, physically performs the error detection.
Figure 2 shows a typical flow diagram of a program executed by the microcontroller. In the flow diagram of a generic assembler program there are two kinds of instructions: sequential instructions and conditional instructions. We assume that all the conditional instructions and some particular sequential instruction (such as the first one) correspond to the flow diagram checkpoints. Therefore, the decomposition of this flow diagram can be done considering all the consequential instructions between two checkpoints.
During the execution of the program, the microcontroller generates a sequence of addresses to read the program code stored in the program memory. Each sequence of instructions is mapped into a sequence of addresses on the address bus; therefore, we can associate to each sequence of instructions a signature vector that is computed based on the sequence of addresses produced on the program address bus.
The calculation of the signature is obtained using Linear Feedback Shift Registers (LFSR) applied to the program address bus.

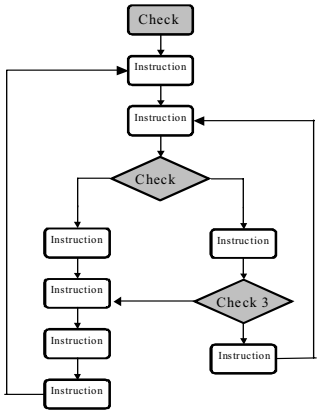

Figure 2: Generic assembler program flow diagram.

The hardware signature analysis system (Figure 3) is intended to perform the hardware check of the signature with the above-defined approach. The checker is pre-loaded with all possible signatures and checkpoints associated with the program executed by the microcontroller. The occurrence of a fault in the microcontroller generates an abnormal sequence that is revealed by the signature checker and communicated to the error handler. The error handler can reset the active microcontroller or substitute it with a spare one if the failure persists after the reset.

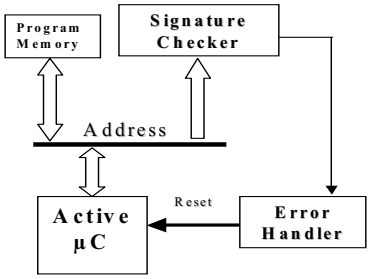

Figure 3: Hardware signature analysis system.

In Figure 4 we show the internal structure of the signature analysis checker. This system is composed of five main elements.

1. The **"Signature generator"** block receives in input the addresses of the instructions that the microcontroller provides during the execution of a sequence of instruction and computes the related signature.
2. The **"Test point detector"** block that is pre-loaded with all the addresses of the test points, detects the start and end of each sequence.
3. The **"System Handler"** block controls the signature generation system. The "System Handler" receives the normal synchronizations signals generated by the microcontroller and schedules the signature computing operations.
4. **"Signature comparator"**: once a test point is detected the present sequence is terminated and the signature vector is evaluated by the signature comparator. If the signature provided is not belonging to the pre-calculated set of signatures, the comparator signals the occurrence of a failure to the error handler that starts the system recovery procedure. Otherwise, if the signature is a valid one, the signature generator starts the signature computation of the next sequence

5. **"Watchdog timer"**: this block handles the occurrence of faults that could cause a freeze of the microcontroller and the consequent impossibility of detecting the fault with signature analysis. A timer is set to zero on each start of sequence and if the time measured becomes longer than the longest possible sequence an error signal is issued.

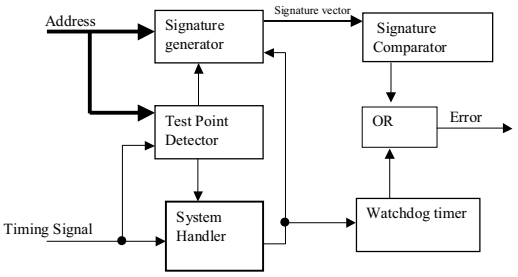

Figure 4: Internal structure of signature checker block.

4. FPGA Implementation of an 8051 with control flow checker

The above introduced methodology has been tested and verified on hardware by using a free synthesizable 8051 IP model [11] (with specific modifications) on a Xilinx Virtex XCV1000 FPGA mounted on the Xilinx AFX BG560-100 development board.
The model we synthesized contains both the program ROM and the RAM of the microntroller. The main characteristics of the model are:

- Instruction set compatible to the industry standard 8051 microcontroller
- Optimized architecture enabling fast one to four clocks per OP code
- No multiplexed I/O ports
- 256 bytes internal RAM
- Up to 64 Kbytes ROM and up to 64 Kbytes external RAM

We then implemented the hardware checker above described that introduces a low hardware overhead. Finally, to test the fault coverage of the proposed checking, we also added some suitable modules (so-called "saboteurs") to inject SEU in the memory locations of the 8051 devoted to the program execution. The area occupied by the microcontroller, with 4 Kbyte of external RAM and 64 Kbyte ROM, together with the checker is summarized in Table 2, as it can be noticed the introduction of the HW checker accounts for a very small overhead of about 9% of the total design.

	# of SLICEs	FPGA occupation	Design occupation
8051	2475	20%	90.9%
HW checker	247	2 %	9.1%
Total	2722	22%	100%

Table 2: Area Occupation of the 8051 microcontroller and the HW flow checker

5. Fault Injection Test Bed

To test the effectiveness of the signature analysis checker we performed a fault injection campaign on the 8051 implemented on the FPGA together with the checker. Fault injection has key role in the design flow of self-checking digital circuits and the FPGA implementation is fundamental to obtain a rapid prototyping of digital systems. Several FPGA fault injection methodologies have been proposed in literature [9] [10]. The speed-up of injection campaigns on a FPGA implementation is very high respect to VHDL simulated methods [3] being the length each run about 60 ms.

The execution of a fault injection campaign can be described as the repetition of the following steps:

1) The test program is partially executed for a random number of clock cycles until occurrence of the bit-flip.
2) A SEU fault is injected into a random register of the set of target registers. The injection is performed using one of the methods proposed in literature.
3) The test program is executed until its end.
4) The RAM and the 8051 registers are read back and compared with the fault-free results (golden run).

We choose to perform the fault injection using some minor modifications to the VHDL code of the system [10]. These modifications allow changing the content of one or more bit of the registers when external signals are asserted. The use of this method is better suited to our purpose with respect to the use of partial reconfiguration of the FPGA [9].
In fact, while the use of partial reconfiguration seems to be very attractive because of its flexibility, we observed that its application in our case presented two major drawbacks:

1) The partial reconfiguration needs a certain time to perform and in this period the operation of the microcontroller must be stopped.
2) In a complex system, like the synthesized microcontroller, the identification of the targets is not straightforward and the operation is time consuming

The approach we used allowed us to quickly perform a large number of injection runs with a very low time overhead. The set-up of the fault injection campaign is sketched in Figure 5: a C program running on a workstation exchanges data and controls with the system mapped on the FPGA board using a parallel cable: the fault injection mask is executed runtime during the program execution without stopping the microcontroller whose clock is controlled by the .
The fault injection is made with a suitable saboteur subsystem operating on the above-mentioned Special Function Registers of the 8051 microcontroller.

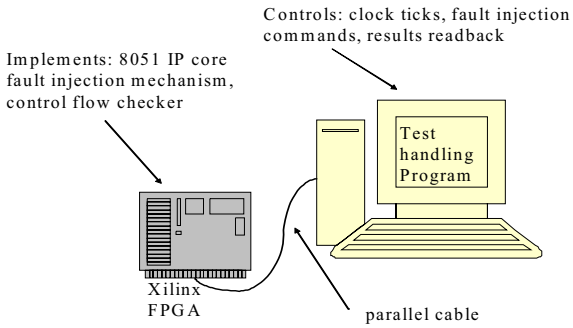

Figure 5 Fault Injection System setup

In particular, our main goal was to perform bit flip injection on the most sensitive registers of the microcontroller with respect to the lost of sequence in the control flow and to evaluate the fault coverage of the proposed architecture. To focus on errors caused by the program flow, we choose as a target test program the vector sorting: the lower sensitivity of this program to general fault injection as shown in Table 1 can be explained because this program makes a lesser use of memory to store the values, therefore more sensitive to errors cased by faults affecting the registers of the control path.
We loaded a simple vector sorting test bench program on the microcontroller ROM [3] and, for each bit flip injection run, we evaluated the correctness of the results with respect to the "golden run". We therefore choose a subset of the register space that we expected to reveal more errors in the program execution. The subset is composed of the following registers:

Program Counter	**PC**
Stack Pointer	**SP**
Data Pointer transfer Register	**DTPR**
Program Status Word	**PSW**
Accumulator	**ACC**
B register	**B**

When the read back is performed the workstation acquires both the results of the computation performed by the 8051 and the signals generated by the control flow checker. In this way we were able to perform the performance evaluation of the control flow checker.

6. Simulation Results

We performed a large number of fault injection runs in three series of randomly chosen injection times and targets, the results of the simulations are reported in Table 3.

	# Runs	#Result Errors	% Result Errors	#Lost of sequence	%Lost of sequence	# Detected Errors	#Detected Lost of Sequence	Result Errors coverage	Lost of Sequence coverage
First	40000	6078	15.19	4969	12.42	3110	4908	51.16%	98.77%
Second	40000	5948	14.87	4864	12.16	3066	4797	51.54%	98.62%
Third	40000	5692	14.90	4785	11.96	2964	4746	52.07%	99.18%
Average	40000	5912.67	14.99	4872.67	12.18	3046,67	4817	51.53%	98.86%

Table 3: Fault Injection Simulation result

As mentioned above, we focused our attention on detecting the errors caused by the faults in the control flow by injecting bit flips in specifically targeted registers. The results show that targeting these registers causes a very similar amount of lost of sequence faults and result errors (12.18% and 14.99% respectively) whereas when targeting all the registers the result errors are much more relevant (see [3]).
We expected that the checker would have a high coverage on the lost of sequence errors as these are closely related to faults that affect the control path of the microcontroller, the results of the simulations confirmed our expectations showing that the checker provided and average 98.86% of coverage. We assumed that a lost of sequence occurred if the program did not terminate in a time 150% of the normal execution time then we checked if the signature analysis checker detected the error. The high coverage allows protecting the microcontroller from freezes that would isolate it from the rest of the system. We also found an interesting simulation result in the correlation between result errors and control flow error with an average of above 50%, however, the improvement of the correctness in the results values should be obtained using appropriate methods for the data path that are not considered in this paper.

7. Conclusions

The program flow checking of the microcontroller was implemented with a signature analysis checker connected to the ROM address bus. The performance evaluation of the system was made running a simple test bench program on the SoC microcontroller for embedded applications synthesized on an FPGA. The choice of the targets for the injections was made

on the Special Function Registers because of their particularly strong impact on the overall microcontroller reliability. The results showed that the checker provides a high coverage with respect to the faults affecting the control flow of the microcontroller allowing a very high level of protection against freezes and showing a 50% correlation between control flow errors and wrong computation results.

8. References

[1] T. Ma, P. Dressendorfer, Ionizing Radiation Effects in MOS Devices and Circuits, Wiley Eds., New York, 1989.

[2] J. H. Elder, J. Osborn, W.A. Kolasinsky, R. Koga, A method for characterizing microprocessor's vulnerability to SEU, IEEE Trans. on Nucl. Sci., vol 35, N° 6, pp. 1679- 1681, Dec. 1988

[3] G.C. Cardarilli, F. Kaddour, A. Leandri, M. Ottavi, S. Pontarelli, R. Velazco, "Bit flip injection in processor-based architectures: a case study", 8th IEEE International On-Line Testing Workshop, IOLTW, France, July 2002.

[4] R. Velazco, Ph. Cheynet, A. Bofill, R. Ecoffet, THESIC: A testbed suitable for the qualification of integrated circuits devoted to operate in harsh environment, IEEE European Test Workshop (ETW'98), Sitges, (Espagne), pp. 89-90, 27-29 Mai 1998.

[5] R. Velazco, S. Rezgui and R. Ecoffet "Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulation Upset) Injection", IEEE Trans. on Nuclear Sci. , Vol 47, Dec 2000 pp. 2405 – 2411.

[6] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodríguez, J.R. Mingo, A New Methodology for the Simulation of Soft Errors on Microprocessors : A Case Study, MAPLD 2000 Military and Aerospace of Programmable Devices and Technologies, Laurel, Maryland (USA), Vol. 1, Session B, 26-28 Sept. 2000

[7] J. Garcia, J. C. Baraza, D. Gil, P.J. Gil "Comparison and application of different VHDL-based fault injection Techniques", Defect and Fault Tolerance in VLSI Systems, 2001. DFT '01. Int. Symposium on , 2001 pp. 233 – 241

[8] Berrojo, L.; Gonzalez, I.; Corno, F.; Sonza Reorda, M.; Squillero, G.; Entrena, L.; Lopez, C. Design, New techniques for speeding-up fault-injection campaigns Automation and Test in Europe Conference and Exhibition, 2002. Proceedings , 2002 Page(s): 847 -852

[9] L. Antoni, R. Leveugle, B. Feher "Using run-time reconfiguration for fault injection in hardware prototypes", Defect and Fault Tolerance in VLSI Systems, 2000 DFT '00. Int. Symposium on.

[10] Civera, P.; Macchiarulo, L.; Rebaudengo, M.; Sonza Reorda, M.; Violante, M. "Exploiting FPGA-based techniques for fault injection campaigns on VLSI circuits" Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on , 2001 Page(s): 250 -258

[11] http://www.oregano.at/services/8051.htm

[12] Wilken, K.; Shen "Continuous signature monitoring: low-cost concurrent detection of processor control errors", J.P. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 9 Issue: 6 , June 1990 Page(s): 629 –641

[13]. Mahmood, A.; McCluskey E.J, "Concurrent error detection using watchdog processors-a survey" Computers, IEEE Transactions on , Volume: 37 Issue: 2 , Feb. 1988 Page(s): 160 -174

[14] Ammari, A.; Hadjiat, K.; Leveugle, R.; "On combining fault classification and error propagation analysis in RT-Level dependability evaluation" Proceedings. 10th IEEE International On-Line Testing Symposium, 2004. IOLTS 2004. 12-14 July 2004 Page(s):227 – 232

[15] Nicolescu, B.; Peronnard, P.; Velazco, R.; Savaria, Y.; "Efficiency of transient bit-flips detection by software means: a complete study" Proceedings. 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2003. 3-5 Nov. 2003 Page(s):377 – 384

[16] Fazeli, M.; Farivar, R.; Miremadi, S.G. "A software-based concurrent error detection technique for power PC processor-based embedded systems" 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 3-5 Oct. 2005 Page(s):266 – 274

[17] Gil, D.; Gracia, J.; Baraza, J.C.; Gil, P.J "Analysis of the influence of processor hidden registers on the accuracy of fault injection techniques" Ninth IEEE International High-Level Design Validation and Test Workshop, 2004. Volume , Issue , 10-12 Nov. 2004 Page(s): 173 – 178

[18] Nicolaidis, M. "IP for embedded robustness" Proceedings Design, Automation and Test in Europe Conference and Exhibition, 2002. 4-8 March 2002 Page(s):240 – 241

[19] G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, A. Salsano, "A Fault-Tolerant Solid State Mass Memory for Space Applications" in IEEE Transactions on Aerospace and Electronic Systems, vol. 41, n. 4, pp: 1353-1372, October 2005

[20] T. Michel, R. Leveugle, G. Saucier "A new approach to control flow checking without program modification" 21st Symposium on Fault-Tolerant Computing (FTCS), 1991, pp. 334-341