Bit flip injection in processor-based architectures: a case study

G.C. Cardarilli[1], F. Kaddour[2], A. Leandri[1], M. Ottavi[1], S. Pontarelli[1], R. Velazco[2]

[1] Department of Electronic Engineering University of Rome “Tor Vergata”, Italy
[2]TIMA laboratory, 46, Av. Félix Viallet, 38031 Grenoble, France

Abstract

This paper presents the principles of two different approaches for the study of the effect of transient bit flips on the behavior of processor-based digital architectures: one of them based on the on-line “injection” and execution of pieces of code (called CEU codes) using a suitable hardware architecture, while the other is performed using a behavioral level processor description, being based on the so-called “saboteurs” method. Results obtained for benchmark programs executed by a widely used commercial 8-bit microprocessor, allow to validate both approaches which provide inputs for an original error rate prediction methodology. The comparison of predictions to measured error rates issued from radiation ground testing validates the proposed error rate prediction approach.

I. Introduction

The concurrency of both high level integration and electronic system wide diffusion is leading to a constant need of reliability performances improvement for the electronic designs. In fact, while the wide diffusion of electronic systems is permeating also areas where the reliability is an extremely important issue, on the other hand the integration process is leading to the occurrence of transient effects like bit-flips of memory elements due to the interaction with radiation (alpha particles, heavy ions) or to electromagnetic noise. In this section are described techniques of Fault-Injection we applied to the 80C51 microprocessor. The considered faults are limited to changes in the content of internal memory elements. Indeed, this model generally called bit flip, upset or SEU (single event upset) represents presently the most likely fault type due to the operation under radiation of integrated circuits [1]. Moreover, the constant improvements in the manufacturing processes make the circuits sensitive to the effects of neutrons present in the Earth’s atmosphere and constitute a significant challenge to the reliability for electronic equipments of the close future [2]. Therefore, it is of high importance the characterization of electronic systems with respect to their sensitivity to the transient bit flip effects. This characterization can be performed using well-known fault injection techniques aiming at perturbing the normal function of the system under study to evaluate the generated effects. The results of such experiments can provide a useful feedback to the designer of fault tolerant systems in order to implement efficient techniques of fault detection and system recovery or fault masking.

Existing fault injection techniques [3] related with transient faults provoked by radiation can be classified in the following main categories:

1. Radiation ground testing
2. Hardware/Software Implemented Fault Injection
3. Simulated Fault Injection

The first technique consists in exposing the device under test to a suitable radiation beam that causes the real occurrence of the physical events affecting it [4]. This kind of experiment is realized on-line, with the DUT performing an activity supposed to be representative to the one carried out during its operation in the final application. Such radiation ground testing entails thus hardware and software developments that may require a significant and time costly effort. We consider the results of this technique as a reference to evaluate the results obtained with the other ones.

The second technique uses also the physical device but the events are simulated by means of the execution, concurrently with the execution of the application program, of pieces of software that modify the content of memory cells (registers, internal memory) that are the privileged bit flip targets [5][6]. Finally, the third

Proceedings of the Eighth IEEE International On-Line Testing Workshop (IOLTW’02)
0-7695-1641-6/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

technique [7] makes use of an HDL model of the device, for which code modifications are implemented in order to virtually change run-time every part composing the device.

In order to evaluate the impact of the injected fault, a mandatory step of each of the above mentioned techniques is the comparison of the obtained results with expected ones, issued from a fault-free performance of the same activity (the execution of a program in case the device under test is a processor).

The main objective of this work is the confrontation of these three techniques when applied to a 80C51 micro controller running two simple programs. The device chosen to be focused by this study being a processor, it sensitivity to bit flips can be 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. The goal was to evaluate the application cross-section, magnitude generally given to quantify the sensitivity of a device with respect to SEUs, which is the quotient between observed errors and number of injected errors, of both benchmarks.

In section II is described the technique used to inject faults using a VHDL description of the 80C51 micro controller along with simulation results obtained when injecting faults during the execution of test-bench programs. Section III is devoted to briefly present the CEU injection technique and the results of its application using a dedicated hardware. Results of both experiment types are compared in Section IV to results obtained during radiation testing experiments performed using a cyclotron to expose the target circuit to particle beams. Preliminary conclusions and future work are drawn in Section V.

II. Fault injection using a behavioral description

A) Bit-flip injection using a VHDL description

Using behavioral descriptions to study the consequences of faults for complex circuits has been widely proposed in the specialized literature. A review of VHDL-based techniques can be found in [8]. Here we have applied the so-called "saboteurs" technique to a VHDL behavioral model of 80C51 in order to study the effects of bit flips when executing a program. The VHDL model uses an array of 8 bit vectors in order to simulate all the 128 internal RAM bytes and Special Function Registers (SFR) included in the 8051 architecture.
Series of tests were performed where faults were randomly injected both in location of the affected bits inside this array and in time of the SEU occurrence. Note that injected faults did not target the memory bits of program code to be executed by the micro controller, the fault injection being performed by means of suitable modifications added to the VHDL signals within the emulated 8051.

The setup of the experiment was therefore a simulation setup. In fact, we generated a VHDL schematic with the instantiation of the studied micro controller and other needed blocks. The main blocks are:

- 8051
- SRAM 64k
- ROM 4k

These blocks were simulated using a commercial VHDL simulator [9]. The only modification we made to the VHDL model was to add a *saboteur* process able to inject bit flips inside the registers within the 8051. The VHDL simulator, concurrently with the normal processes emulating the 8051 micro controller, executes the *saboteur* process. Therefore the activation of fault injection, performed by means of the two extra signals IND and BIT, is totally independent and asynchronous to the state of the 8051.
In fig. 1 is shown a schematic description of the fault injection strategy in a general case. The VHDL behavioral description of the 8051 can be seen as a set of concurrent processes, each one implementing a function of the micro controller (ALU, PC incrementer, Watchdog, etc.) and the communication between these processes is provided by a set of internal signals visible to all the processes. Some of these signals have physical counterparts like SFR, RAM, PC and others. The *saboteur* is a special process that runs concurrently to the other processes and is activated, in our case, by two external commands IND and BIT. When normal operation (without fault injection) is carried out the *saboteur* is in a stand-by mode, when the

Proceedings of the Eighth IEEE International On-Line Testing Workshop (IOLTW'02)
0-7695-1641-6/02 $17.00 © 2002 IEEE
IEEE COMPUTER SOCIETY

fault injection is activated, the saboteur modifies an internal signal inverting its value. In this way the *saboteur* provides an asynchronous SEU injection on any internal signal of the VHDL description.

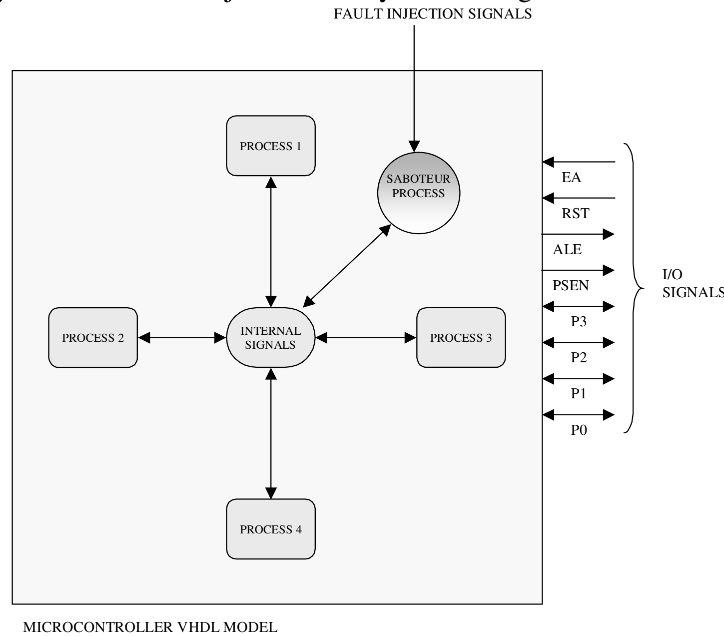

Figure 1: "Saboteur" process

It has to be noticed that, in order to have a realistic behavior of the micro controller, the injection must be performed only on those signals representing physical registers. Assuming that the SEUs mainly affect the internal registers rather than combinatorial logic, we isolated the signals representing these registers and made them our SEU injection target.

In fig. 2 is shown the external 8051-module instantiation with fault injection capability. The IND and BIT signals appear as two additional ports of the 8051 and are driven by the simulator executing a macro file. IND signal indicates the address of the internal byte to be affected by the injected fault and BIT indicates the specific bit to flip. The instant of the injection is forced by the macro together with these two addresses.
The fault injection technique is composed of the following steps. First is defined the time width of injection zone relatively to the program that must be tested, secondly is defined the number of logical targets that must be used in order to inject error in all internal register (in this case 152). Once both these ranges are determined, the macro routine is generated for executing test using a suitable C++ program. The C++ program is based on recursive algorithm, the first step of each cycle is the generation of time variable for error injection (into the predefine injection zone interval time), the second step is the generation of logical targets of injection, both byte and bit addresses. Finally the C++ program writes the correct macro file commands for fault injection.

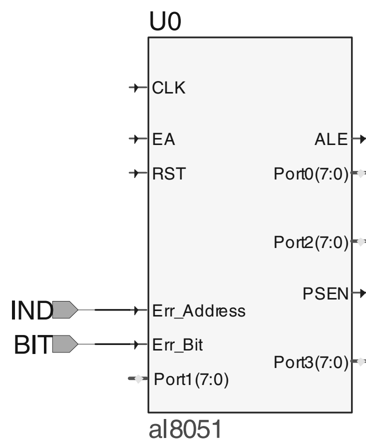

Figure 2: Structure of VHDL 8051 model with fault injection capability

0-7695-1641-6/02 $17.00 © 2002 **IEEE**

IEEE COMPUTER SOCIETY

B) Setup of a VHDL fault injection Campaign

Setting up a fault injection campaign requires the following steps:

- Analysis of the VHDL model
- Set up of the Saboteur process
- Set up of the Macro generator Program

In principle these steps can be applied to any VHDL description of a digital system including memory elements. For instance, can be corrupted by this method the content of bits of a Finite State Machine status register, as well as the internal registers of a micro controller description. The first step requires the analysis of the VHDL description to find the targets suitable for bit-flip injection. The second step is done adding the saboteur VHDL process described above, capable of modification of the value of the selected target. The third step is the generation of the macro file that drives the simulator engine giving both the stimuli to the VHDL description of the device under test (like clock reset etc) and the randomly generated activation stimuli for the saboteur process.
The first step is the more important change to be afforded when a new device VHDL description must be tested. Once the targets of the VHDL code are identified, the modifications related to the second step consist in the connection of the saboteur to the selected target. The stimuli for the macro generator to the saboteur are almost the same while the stimuli for the DUT VHDL description are strictly related to its functions.
Therefore, once the setup phase is performed the fault injection campaign can be carried out in batch mode, the length of the simulation depending on the complexity of the VHDL model.
In the following paragraphs are described the results obtained when applying this injection strategy to the 8051 VHDL description while running two test bench applications.

B.1) Matrix multiplication

The matrix multiplication program operates in four phases (fig. 3), in the first phase, some internal registers are set in order to initialize the system, in the second phase, the 6x6 matrixes are generated and stored in the internal RAM, in the third phase the 6x6 product matrix is generated (this is the more time consuming phase), in the last phase, the result matrix is compared to the expected one and the incorrect results are stored in the external RAM. We define the "Test" as the union of all these four phases, and define the "Injection Zone" as the union of "generate matrix" and "product of matrix" phases. At the end of each test the system provides the dump signal that generates a report of errors revealed during the result control phase.

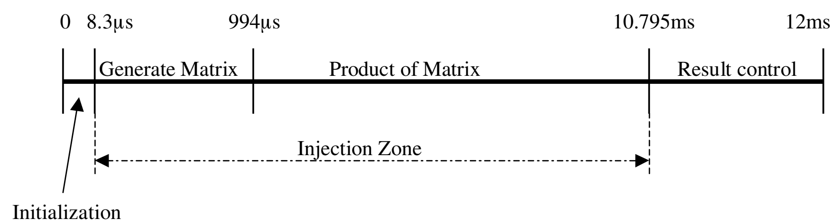

Figure 3: Phases of Test for the matrix multiplication program

We assume for this study that only single bit flips may occur during the execution of the multiplication program (the injection of multiple bit flips is also possible). The bit flip injection will arise only within the zone called "Injection Zone" with a random (uniform) distribution in time and in location.
The whole test lasts about 12 ms. The obtained results, in terms of percentages of errors with respect the total number of injected bit flips, are reported in table 1 classified as tolerated errors, result errors and lost of sequences. "*Tolerated errors*", correspond to those bit flips injected in memory elements which do not cause any effect at the outputs of the program. "*Results errors",* gathers cases where obtained results are different from the expected ones. Finally, the cases where we do not get any answer from the processor are classified in the loss of sequence group. The consequences of injected bit flips belonging to this last malfunction type

IEEE COMPUTER SOCIETY

are unrecoverable, needing to restart program execution. The results were analyzed in detail to determine the number of wrong elements in the result matrix obtained in each faulty execution.

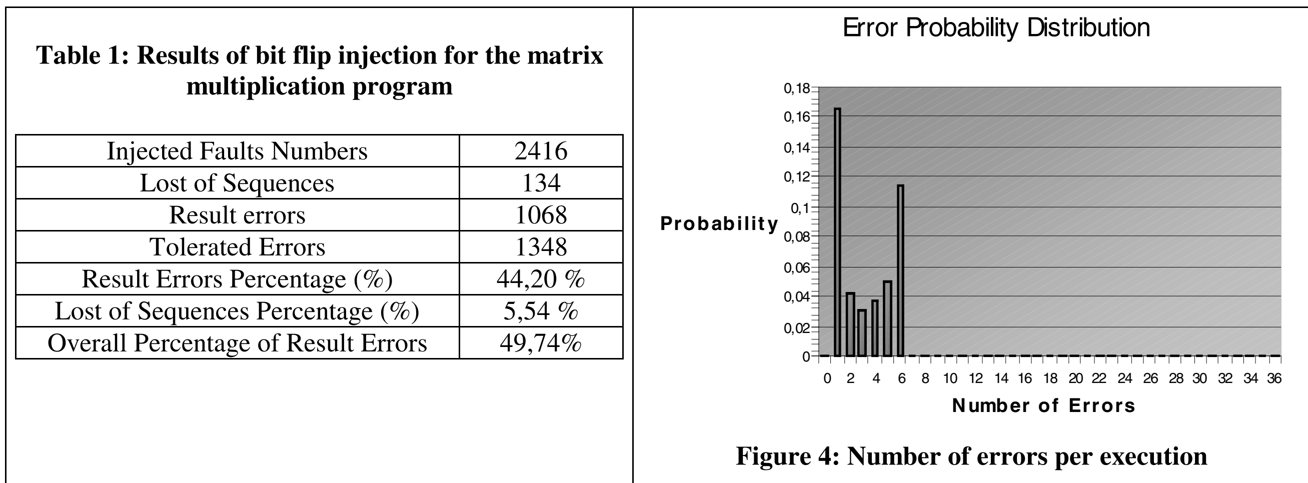

Table 1: Results of bit flip injection for the matrix multiplication program

Injected Faults Numbers	2416
Lost of Sequences	134
Result errors	1068
Tolerated Errors	1348
Result Errors Percentage (%)	44,20 %
Lost of Sequences Percentage (%)	5,54 %
Overall Percentage of Result Errors	49,74%

Figure 4: Number of errors per execution

In figure 4 is reported a histogram obtained from table 1 and normalized to the total number of tests, representing the probability distribution of the error within a single test. An explanation of these results could be that the propagation of an injected SEU is related to the fault injection instant. In particular the experiments leading to six errors suggest that was corrupted a value of one of the 2 matrixes before they were multiplied. The latency of these unelaborated data inside the micro is quite long; in fact, if we inject the bit flip during the generation of the matrixes the incorrect value of the multiplicand matrix will generate 6 incorrect values on the result matrix. Also during the elaboration the SEU injection could affect unelaborated data but with a lower probability while the elaboration proceeds.

B.2) Vector sorting program

The second experiment was made using a vector-sorting program. This program operates in five phases as shown in Fig. 5. In the first phase, some internal registers are set in order to initialize the system and a 30 element vector is generated and stored in the micro controller internal RAM. In the second phase, the original unsorted vector is stored in the external RAM. In the next phase the vector stored in the internal RAM of 8051 micro controller is sorted, this is the more time consuming phase. In the fourth phase the sorted vector is copied in the external RAM, this is the last phase of injection zone. In the last phase, the result of sorting algorithm is tested and the incorrect results are stored to the external RAM. Note that in this case the number of maximum possible errors is 30.

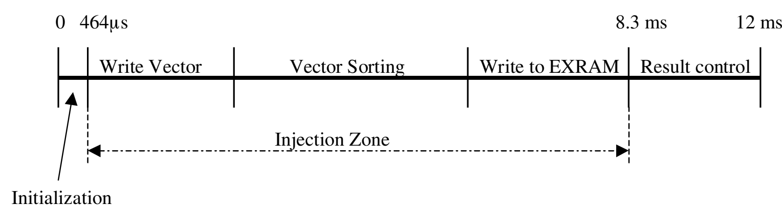

Figure 5: Simulation phases of Vector sorting program

In this case we define the “Test” as the union of all these five phases, and define the “Injection Zone” as union of “Write vector”, “Vector sorting” and “Write to exRAM” phases. At the end of each test the system provides the dump signal that generates a report of errors revealed during the result control phase. The test follows the same steps of the previous one and the fault injection and result analysis are made in the same way.

IEEE COMPUTER SOCIETY

Like in the matrix product test, the injection of bit flip occurs only within the zone called "Injection Zone". The whole test lasts about 12ms and the percentages of result errors, tolerated bit flips and lost of sequences are reported in table 2. Moreover, we analyzed the results with details to evaluate how many wrong results in the vector were detected in each faulty test.

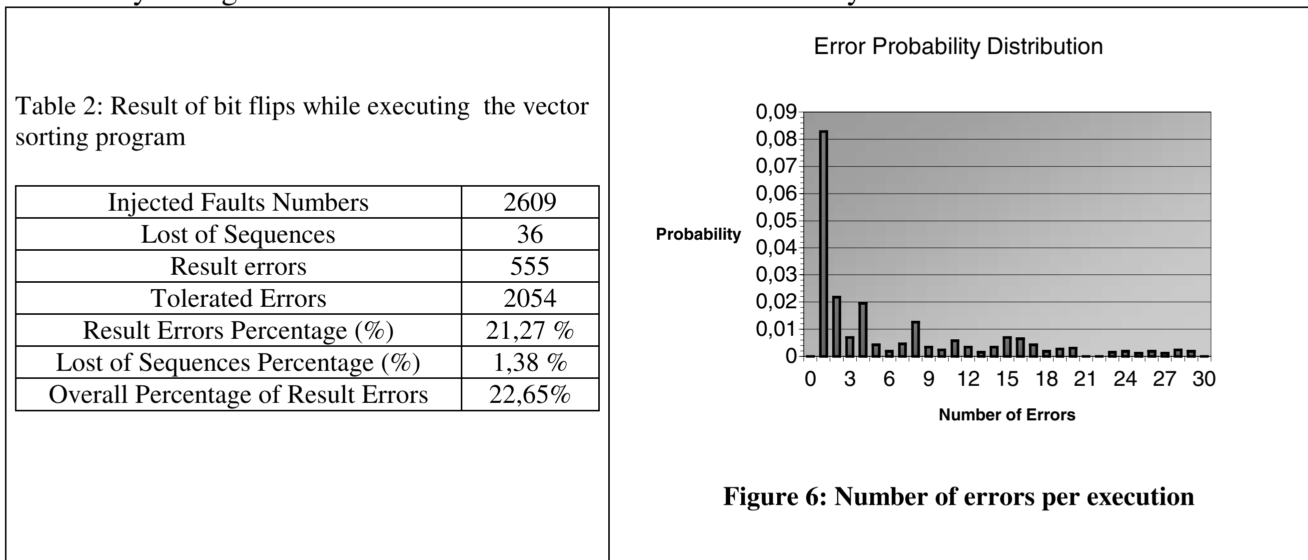

Table 2: Result of bit flips while executing the vector sorting program

Injected Faults Numbers	2609
Lost of Sequences	36
Result errors	555
Tolerated Errors	2054
Result Errors Percentage (%)	21,27 %
Lost of Sequences Percentage (%)	1,38 %
Overall Percentage of Result Errors	22,65%

Figure 6: Number of errors per execution

Fig. 6 clearly shows that the errors are more distributed for the vector sorting program than in the case of matrix product case. In table 2 it can be seen that most of the injected faults had no effect at the sorted vector level. In fact, since the vector sorting program uses less memory locations than the matrix product program, the fault injection in the vector sorting experiment provides a lower error rate.

III. C.E.U injection approach

A) Principle

For processor-based architectures, bit flip injection can be performed at system level using an existing hardware architecture including the circuit to be tested (commercially available emulator boards for instance) or a specific hardware (dedicated testers). The main idea is here the injection of faults as the consequence of the execution of suitable pieces activated at desired instants by an interrupt-like signal (available in most of the processor-based architectures). Such a technique, so-called CEU (Code Emulating an Upset) injection and proposed initially in [10], has been applied to various processors devoted to space applications. It efficiency was proved for different circuits by comparing radiation data and CEU based predictions [11]. In the following the C.E.U injection technique is briefly presented, details of its implementation to complex processors can be found in [11].

Three steps are needed to set up a C.E.U injection experiment for a given processor:
- the first step concerns the identification of bit flip targets, called *CEU targets* in the following, which is accomplished through the identification of those memory elements (registers, internal memories, etc...) of the processor which are accessible (read or write operation) via the instruction set. It defines the sensitive area where upsets can be injected through the CEU mechanism.
- the second step consists in determining an instruction sequence, called *CEU code*, whose execution allows the inversion of the content of a selected bit for each of the CEU targets. In fact, the CEU targets are classified into families according to the type of associated CEU code, which is composed, in the simplest case, of a single instructions performing the XOR between the target content and a suitable mask, followed by a return from interrupt instruction.
- the last step requires the development of an external program, for the automated hardware/software monitoring of fault injection experiments. Indeed, to accomplish the injection of bit flips, randomly in time and location, needs an external hardware to trigger the interrupt procedure, automate the memory loading

0-7695-1641-6/02 $17.00 © 2002 IEEE
IEEE COMPUTER SOCIETY

with data corresponding to the desired CEU code, and to compare the outputs to expected results and monitor the application program execution time to detect possible sequencing faults (sequence losses) resulting from injected upset.

The architecture of a dedicated test system, called THESIC (Testbed for Harsh Environment Studies on Integrated Circuit) developed at TIMA laboratory as a generic platform for radiation ground testing purposes [12], offered an attractive infrastructure for the CEU approach and was used to implement all the needed mechanisms.

B) Performing CEU injection experiments on a 8051 micro controller

The CEU injection approach was used to derive the error rates of benchmark programs. The CEU targets of the chosen micro controller are: the 128 bytes of internal RAM, the Special Function Registers (SFRs), the Program Counter and general purpose registers. It must be noted the existence in the 8051 architecture, of a few memory elements potentially sensitive to upsets (UAL registers, control part flip-flops, latch registers, etc…) which cannot be affected by the CEU injection approach. The amount of these inaccessible targets was estimated as being 7% of the whole bit flip sensitive area.
Details about the determination of CEU codes for each of the CEU targets as well as the analysis of particular cases obtained from CEU injection experiments can be found in [13]. We limit here to report the results obtained for the two mentioned benchmark programs, in order to compare the two approaches studied in this work.
Various automatic experiments were performed during which thousands of bit flips were injected, randomly in both time occurrence and location, concurrently with execution of the studied programs. These experiments aimed at quantifying the rate of "effective" upsets for the tested programs, in order to derive realistic figures for the expected error rate in the final environments.
Results given in the Table 3 following the same classification than in preceding section were obtained using pseudo-random CEU injection sessions, by running the matrix multiplication and the bubble-sort vector programs.

Table 3: Results of CEU injection for two benchmark programs

	Matrix Multiplication	Bubble-Sort Vector
Injected Faults	12245	15632
Tolerated Errors	6117	11792
Results Errors	5784	3484
Sequence Loss	344	356
Error Rate (%)	50,04 %	24,56 %

As in the case of fault injection performed using VHDL descriptions, in order to provide the worst-case conditions in terms of exposing the circuit to the effects of SEUs, the 6×6 matrix multiplication program was designed in such a way both the operand and result matrixes are stored within the internal SRAM, occupying most of this memory. Thus, only 20 bytes (10 SFRs and 10 bytes of it internal memory) were not used. Note that in this case, practically half of the total number of injected faults caused errors on the results, while only 2,8% of them caused sequence loss. This last case was mainly observed when injecting bit flips in PC counter.

Results of CEU injection experiments performed on the vector sort program, gave an error rate approximately the half than the one obtained for matrix multiplication program. This lower sensitivity was expected for this program, as it only uses 30 bytes of the internal memory that corresponds to around 24% of the all internal memory (while 77% of it was used for matrix multiplication application).

IV. Using fault injection result to predict error rates

A) Radiation ground testing set up

Radiation testing experiments, in which the 80C51 processor was exposed to beams of several heavy-ion species, were performed the CYCLONE cyclotron available at the UCL (Université Catholique de Louvain-la-Neuve, Belgium). Further details about the Cyclone facility and the main characteristics of the heavy ions, to which the studied processor was exposed are provided in [14].

As an illustration of a typical experimental set-up used while performing such kind of testing, in fig. 7 is depicted a photo of THESIC tester within the vacuum chamber of CYCLONE cyclotron devoted to heavy ion radiation testing.

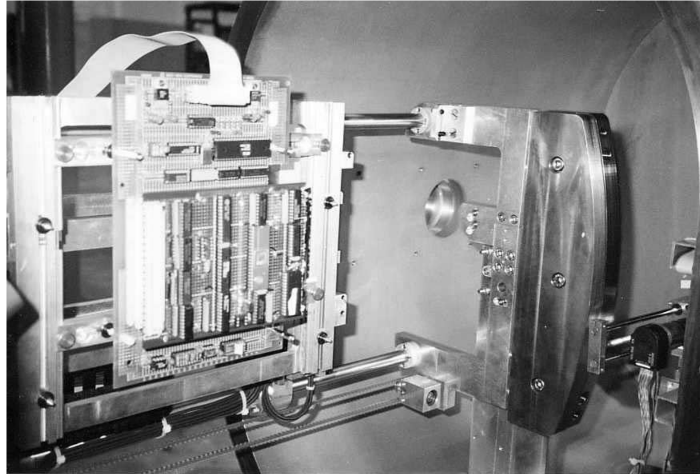

Figure 7: The 8051 hardware set-up for radiation ground testing in the CYCLONE vacuum chamber

These experiments of ground testing allowed measuring the static SEU cross-section of 8051 micro controller by executing a memory-like test pattern (static strategy) while exposing the circuit to the heavy ion beams. Such an experiment provides statistical evaluations of the number of particles needed to flip a bit of a given memory element. The resulting cross-section curve gives for each of used particle species (identified by the energy they deposit in Silicon measured by the LET or Linear Energy Transfer) the number of detected bit flips normalized by the number of hitting particles (eq. 1).

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

Measured cross-section curve for the 8051 is depicted in fig. 8.

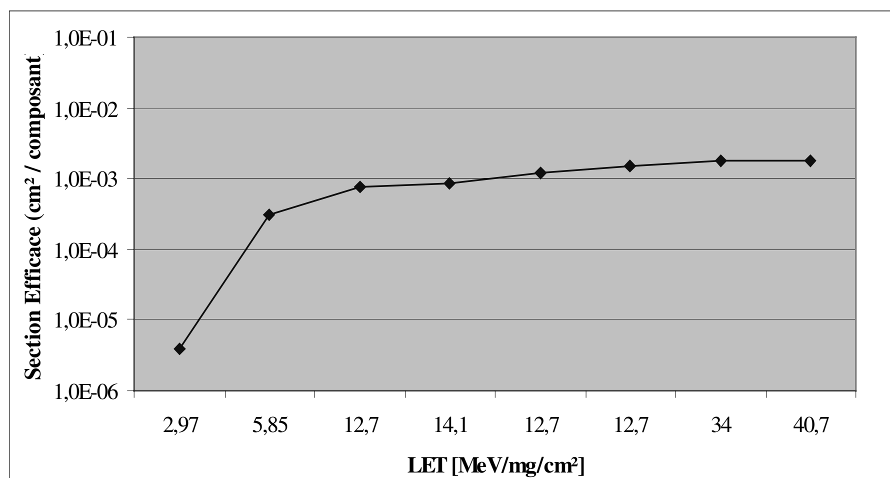

Figure 8: Static cross-section obtained for the 8051 micro controller

IEEE COMPUTER SOCIETY

B) Predicting program error rates from fault injection experiments

In [12] was presented an approach to predict error rate for an application while executing a given program under radiation, based on fault injection results and a limited radiation ground testing. The idea is to measure the bit flip sensitivity of the DUT memory elements by execution of a static test under radiation. From the derived cross-section, is known a probabilistic estimation of the number of particles needed to get a bit flip on a particular target. From fault injection experiments presented in sections II and III, it can be estimated τ_{CEU} which is the program error rate give as the number of injected bit flips leading to error in the execution of a given program normalized by the total number of injected errors (eq. 1).

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

The sensitivity to SEU of the program can be calculated by the product of the cross-section by the error rate to CEU:

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

Indeed, multiplying the static cross-section by the error rate derived from fault injection constitutes an 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 the case of the circuit and programs evaluated in this study, the error rates obtained by the two fault injection methods are very close, the resulting predictions for program error rate will thus be also very similar. In table 4 are summarized the error rate factors derived for the two programs. Indeed, to get the estimated error rate for a particular heavy ion, this figures should be multiplied by the corresponding cross-section measure, reducing in our case the difference to a few percents.

Table 4: Error rates factors obtained by fault injection for the two benchmark programs

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

Aiming at comparing the validity of prediction based on fault injection sessions radiation ground testing were performed with the UCL cyclotron. Owing to the cyclotron facility planning constraints, we could only expose to radiation the 8051 while executing the matrix multiplication From the formula 2 and the underlying SEU cross-sections, the error rates of the matrix multiplication program have been estimated. In fig. 9 are depicted curves corresponding to both the measured and predicted error rates. Notice that owing to the fact that the vertical axis is a logarithmic one, no differences are visible for the two prediction curves, only one being thus represented. The comparison of predicted and measured error rate curves, put in evidence the excellent correlation obtained from the prediction technique based on fault injection. In fact, the two curves are practically superposed except for the neon ions (LET =5,85 [MeV/mg/cm²]) where the difference is still negligible. These results show clearly the excellent efficiency of this technique to predict error rates, at least for the simple studied processor (the 80C51), where bit flips can be injected in a large part of internal sensitive zones of this micro controller, corresponding approximately to 93% of the whole sensitive area.

IEEE COMPUTER SOCIETY

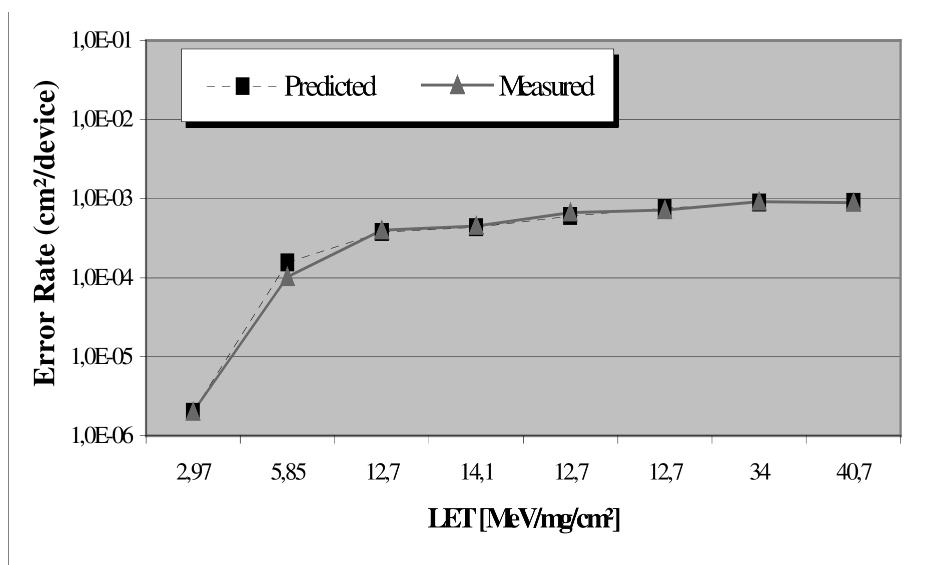

Fig. 9: Predicted and measured error rates under different heavy ions for the 80C51
Exposed Program: a Matrix Multiplication

V. CONCLUSIONS AND FUTURE WORK

Two different techniques for fault injection of bit flips in processor-based architectures were exposed and experimented on a 8051 micro controller. The first one is based on behavioral description and uses a "saboteur" VHDL process, whose activation concurrently with the simulation of a target program, induces the injection of a bit flip in one of the memory elements used by the description. The second one works at the hardware system level, injecting faults by means of the execution of particular pieces of code activated asynchronously during the execution of the studied program. Both approaches were applied to inject faults "on-line" concurrently with the execution of simple benchmark programs. The obtained results were very similar, in terms of both the number of injected faults provoking deviation in the program results and the type of deviation provoked. This can be explained by the fact that both approaches focused the same memory elements target set: the internal registers and memory of the 8051. The only difference was the instant of fault injection: synchronous with the clock for the behavioral approach and limited to instants of interrupt assertion for the CEU injection method. These results prove that this limitation, which could constitute a serious obstacle, is not a major drawback for the CEU injection approach, at list for the studied case.

Radiation testing was performed on the 8051 to compute its underlying SEU cross-section and to get measures of the error rate of a simple program. Error rates of the 8051 when executing a program under radiation were predicted from fault injection experiment results combined with underlying cross–section. Prediction and measures were in very good agreement validating both the injection methods and the prediction approach.

One of the main advantages of using VHDL based fault injection technique compared to the CEU approach, is that being based on a behavioral model of a target circuit, it allows, in principle, to perform fault injection in all the targets of bit flips provoked by radiation. Even in the case presented in this work, for which fault injection was limited to a subset of memory elements, obtained results were very close to those issued from physical fault injection performed by radiation testing. The second advantage relies in the possibility to inject faults during each of the clock cycles, simulating in a quite good way the asynchronous occurrence of real faults due to particle ionization. Therefore the VHDL injection seems a suitable fault injection technique also for characterizing behavioral models of not yet implemented devices. The VHDL technique can also be applied, without substantial modifications, to structural models to obtain a wider range of injection targets.
On the other hand, behavioral fault injection needs the availability of a suitable model of target circuit, which is not always the case. In terms of simulation time the VHDL injection is significantly slower compared with the CEU injection technique. But this should not be a major drawback considering that fault injection sessions can be done in laboratory, and do not need expensive installation like particle accelerators used for radiation ground testing.

Future work includes performing the same kind of research on a more complex processor, to study the difficulties related with the existence of pipeline blocks and cached memories.

IEEE COMPUTER SOCIETY

V. REFERENCES

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

[2] E. Normand, Single-Event Effects in Avionics, IEEE Trans. on Nuclear Science, Vol. 43, n° 2, pp. 461-474, April 1966.

[3] M. C. Hsueh, T. K Tsai and R. K. Iyer "Fault Injection Techniques and Tools" Computer , Volume: 30 Issue: 4 , April 1997, pp. 75 –82

[4] 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

[5] J. Carreira, H. Madeira, J. G. Silva, Xception : a technique for the experimental evaluation of dependability in modern computers, IEEE Transactions in Software Engineering, Vol. 24, N° 2, pp. 125-136, February 1988.

[6] A. Benso, P.L. Civera, M. Rabudengo, M. Sonza Reorda, A low cost programmable board for speeding up fault injection in microprocessor-based systems, Annual Reliability and Maintainability Symposium, 1999, pp. 171-177.

[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] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, Fault injection into VHDL Models: the MEFISTO tool, Proc. of 24th International Fault Tolerant Computing (FTCS-24) , pop. 66-75, Austin, Texas (USA), June 1994.

[9] Aldec, Inc "Active-HDL, VHDL Reference Guide" April 2001.

[10] 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.

[11] 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. To be published at JSR (Journal of Space Rockets).

[12] 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.

[13] R. Velazco, S. Rezgui, H. Ziade, Assessing the soft error rate of digital architectures devoted to operate in radiation environment: a case studied, *accepté pour présentation à LATW 2001 (IEEE Latin-American Test Workshop), Cancun (Mexique)*, 11-14 février 2000

[14] G. Berger, G. Ryckewaert, R. Harboe-Sorensen, L. Adams, Cyclone - a multipurpose Heavy Ion, Proton and Neutron SEE Test Site, RADECS Radiation and its effects on Components and Systems, 1997.

IEEE COMPUTER SOCIETY