# METHODOLOGY AND TOOLS FOR ESTIMATING PROCESSORS SENSITIVITY TO SOFT ERRORS INDUCED BY RADIATIONS


**Said KAROUI**
Electrical Engineering Faculty, University of Sciences and Technology of Oran Mohamed Boudiaf (USTO MB),
BP 1505 El M'naouer 31000 ORAN, Algeria, Phone / Fax 00 213 41 62 71 25, Email sd_karoui@yahoo.com

**Raoul VELAZCO**
TIMA, CNRS - Grenoble Alpes University,
46, avenue Félix Viallet, 38031 GRENOBLE France, Phone / Fax 00 33 4 76 57 45 00, Raoul.Velazco@imag.fr



***Abstract:*** *Energetic particles present in space and in the Earth's atmosphere may produce transient and permanent degradations on integrated circuits. This may lead to critical faults in systems on-board satellites, aircrafts and even in applications operating at ground level such as automotive, medical, etc..... In this paper is presented, a methodology and the required tools for estimating the vulnerability of applications running on microprocessors with respect to bit-flip errors resulting from Single Event Upsets, also called soft errors. The proposed method allows estimating the sensitivity to this effect for any kind of processor running target application programs. Experimental and computational results of SEU tests performed on various complex processors by means of different test systems, as well as a fault emulation approach for predicting error rate will be presented and compared validating the accuracy of the proposed predictive error-rate approach.*



***Key words:*** *Space environment, SEU, particle accelerators, Test program, Fault injection.*


## 1. Introduction

The particles present in the space environment and even in the earth's atmosphere interact with the digital circuits. At the level of the circuit this interaction may result in transient and permanent degradations [1][2]. Among these degradations the singular effects, SEP (Single Event Phenomena) are mainly due to the impact in a circuit sensitive area of single particles such as high energy protons, heavy ions or neutrons with a LET (Linear Energy Transfer) higher than a threshold value. Various means of prevention can be employed to face and significantly mitigate some of these troubles. In case it is considered the occurrence of a SEU (Single Event Upset) phenomenon, also called *upset* or *soft-error*, the consequences of this phenomenon depend on the significance of modified information [3][4] and its random nature both in time and target memory cell makes it critical.

Simulation at ground level, also called *radiation ground testing*, makes possible to study and predict the sensitivity to SEU of components and thus to choose the least vulnerable circuits. It consists in exposing the Device Under Test (DUT), to a radiation representative to that which it will meet in the final application, while it is excited by a suitable sequence of test stimuli. The evaluation of the responses, using an adapted extrapolation, allows predicting its sensitivity to the considered radiation.

Setting up such a test raises the following problems:

- The simulation of the authentic environment,
- The generation of the test stimuli,
- The application of test stimuli to the DUT and the evaluation of results, by means of a test platform.

In case of heavy ions testing, the DUT is placed in a vacuum chamber. To protect the test platform, the DUT and the system must be connected by a shielded cable (Figure 1).

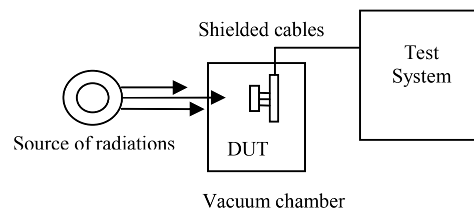


Fig. 1: Set up for radiation ground tests

## 2. Radiation ground test facilities

Data presented in this paper was obtained using three different radiation facilities:

- The Tandem Van de Graaff heavy-ion accelerator of the I.P.N. (Nuclear Physics Institute, Orsay, France). Effective LET values were obtained through varying the beam incidence angle (see eq.1, Table 1).

- A Californium 252 fission-decay source equipment available at the military and space research center ONERA/CERT/DERTS (Toulouse, France).
- The Heavy Ion Facility (HIF) cyclotron accelerator of Louvain Catholic University (UCL, Belgium). Table 2 depicts the ions used during the experiments.

$$LET_{\text{eff}} = \frac{LET_{\text{i}}}{\cos \theta_{\text{i}}} \qquad (1)$$

Table 1
I.P.N. Accelerator Ion Energies and LET

| Ion | Energy(MeV) | LET(MeV/mg/cm$^2$) |
|---|---|---|
| 12C | 84 | 1.7 |
| 19F | 111 | 4.0 |
| 35Cl | 153 | 12.7 |
| 58Ni | 179 | 27.0 |

Table 2
H.I.F. Accelerator Ion Energies and LET

| Ion | Energy(MeV) | LET(MeV/mg/cm$^2$) |
|---|---|---|
| 12Ar | 372 | 10.1 |
| 17Ni | 500 | 21.9 |
| 25Kr | 756 | 32.4 |

## 3. Test system design

Concerning the test system design, two representative approaches available in the literature are presented in references [5][6]. They can be distinguished by the methods used for the test stimuli application and the response evaluation.

### 3.1. Stimuli application

The application of the test stimuli can be realized according to two principles:

- *Dominated control*: the test stimuli are stored in the tester memory as binary values. The tester converts them to electrical signals and separately applies them at the DUT inputs. The sequencing of these stimuli is totally controlled by the tester. When the DUT is a microprocessor, the tester provides the data and instructions at the proper timing.

- *Assisted control*: the DUT is in a "natural" environment, the stimuli are stored in the tester memory as a sequence of instructions. These instructions are executed by the DUT itself (if it is a microprocessor) or by an associated microprocessor (for none microprocessor DUTs). The DUT, or its associated microprocessor, accedes normally the memory to fetch data and instructions.

### 3.2. Response evaluation

The evaluation of the DUT responses is made by comparison to a reference. Two solutions used are:

- *Well-known responses*: the reference values are established before the test, either by the application of the test stimuli to a "good" circuit or by simulation or computation.
- *Golden chip method*: the reference values are the output responses of a "good" chip not exposed to radiation or an emulator responses.

### 3.3. Choice of a test system

Testers implementing the *dominated control* strategy allow performing efficient parametric and logic tests for all kind of circuit types. Commercially available testers are generally designed using this strategy. Nevertheless, using these testers requires a precise analysis of the DUT timings to prepare the test sequences and to analyze the responses when a precise diagnosis is looked for. Moreover this strategy leads to bulky equipment.

An *assisted control* strategy allows working in microprocessor native languages. The test developer can think more naturally and efficiently. The test meaning is also clear to others who may need to update the test program. It leads to low cost, compact and portable testers. Nevertheless a specific interface board has to be developed for every new DUT.

Concerning the response evaluation, the use of the *golden chip* method needs complex hardware developments (comparators, tri-states encoding, etc.), particularly to enable the comparison only when the signals are stable. The reference must have a high quality, to avoid uncovering of a similar design or masking flaws in both the reference and the DUT.

As the *well-known responses* can be correctly established (emulation for instance) and easily updated, the *well-known responses* strategy is generally preferred.

Several testers (FUTE16, THESIC, and ASTERICS) were designed and realized at TIMA Laboratory according to the DUT *assisted control strategy with well-known response evaluation* [7].

### 3.4. Test systems architectures

The test systems were developed at TIMA with the collaboration of the French Space Agency (CNES), to cope with both functional tests and vulnerability studies against radiation in space environment.

In these testers the DUT operates in its natural environment (interfaced to a typical architecture). When the DUT is a processor, the interface is composed of memories to store the test program, glue logic, power and clock circuitry. The result of the execution of the test program is stored in a memory for comparison with a reference data.

These testers are mainly composed of:

A main system (motherboard) responsible of performing the following tasks: control all operations related with the DUT, i.e. power on/off, current consumption control, test stimuli download, starting and stopping test cycles, and communicating data to and from the computer host using a serial link.

A daughterboard (application specific card) implementing a suitable functional environment, where the DUT will be exercised by specific test stimuli.

A computer host for user interface (on-line monitoring of the test execution, result displaying) and mass memory purposes (storing the experiment logs for later analysis).

The features of these testers can be summarized as follows:

- Versatility: it allows testing ranges of digital circuit types (processors, peripherals, memories).
- Portability: it is autonomous, compact, and monitored by a terminal or a PC compatible.
- Flexibility: a specific daughter board must be realized for each new DUT, the test program can be developed on the tester directly as an object code file or in a high level language on a host (PC computer).
- Low cost.

**3.4.1. FUTE16 tester**

The architecture of the first tester FUTE16 (Functional Upset TEster) is organized around a VME bus (Figure 2).

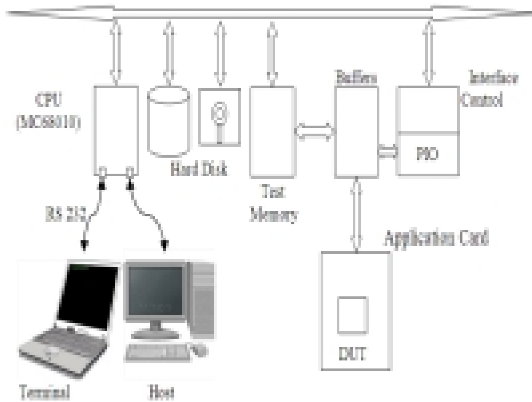


Fig. 2: Architecture of the FUTE16tester

A specific interface daughter board has to be realized for every new DUT. Its role is to produce, through the parallel IO port, a DUT minimum environment. Different mechanisms (watch-dog, address control, illegal op-code detection, etc.) have been implemented to detect some critical errors.

The test program is written in a high-level language on the host, the obtained object code is transmitted to the tester's mass-memories. The object code is then downloaded at the tester RAM.

Test stimuli application is achieved by the execution of the test program by the DUT (or the associated microprocessor, in case of non-processor circuit) after an assertion of the “reset” signal by the CPU via an input/output port (PIO). During the test program execution, the test results are stored in the tester RAM. A “test end” signal is then sent by the DUT to the CPU via the PIO. Finally, to detect the potential errors the CPU compares the results to predetermined reference values.

It is important to note that the DUT current consumption must be permanently controlled in order to protect the DUT against Single Event Latchup (SEL). Indeed, SELs are destructive faults resulting from the impact of a single particle provoking, a short circuit between Power and Ground which leads to thermal destruction of the circuit.

The software, written in C language, takes over the control and the management of all operations related to the test. The user interface allows an easy manipulation of the machine. It is realized by a high-level commands organized as a hierarchical menus.

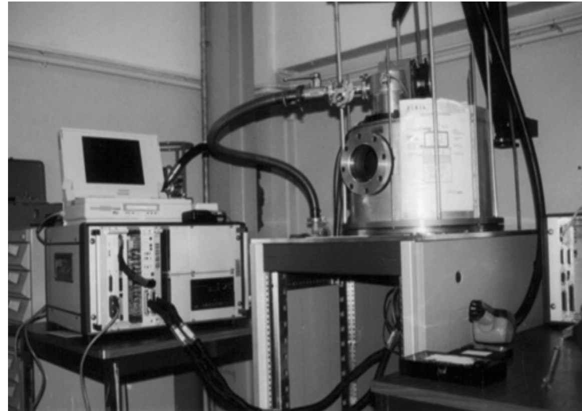

Fig. 3: FUTE16 at ONERA Californium facility

**3.4.2. ASTERICS tester**

The growing complexity of the DUTs lays down to improve the performances of the testers in order to meet the short timings, high frequency requirements of the last digital circuits available on the market. The main idea of the ASTERICS (Advanced System for the Test under Radiation of Integrated Circuits and Systems) [6] was to implement the digital environment needed by the DUT by means of an FPGA whose configuration is obtained from

compiling the description of this environment in a hardware description language such as VHDL. In this way, the user work is limited to wiring the DUT signals to the ones of the tester.

The architecture of ASTERICS takes into account the performance's requirement of the latest processors. It is built around two Xilinx Virtex4 FPGAs (see Figure 4):

- *Control FPGA*: a Xilinx Virtex4 FX, embedding a PowerPC processor, is in charge of both the control of the test, and the communication between the user and the tester via a gigabit Ethernet link. The PowerPC runs at 300MHz allowing a good data rate transfer.
- *Chipset FPGA*: a Xilinx Virtex4 LX, optimized for logic resources, embeds a design allowing connecting the DUT to the required digital/analog resources available in the motherboard and needed by the DUT. In case of a processor, this chipset acts as a memory controller. This FPGA allows operating frequencies of about 200MHz.

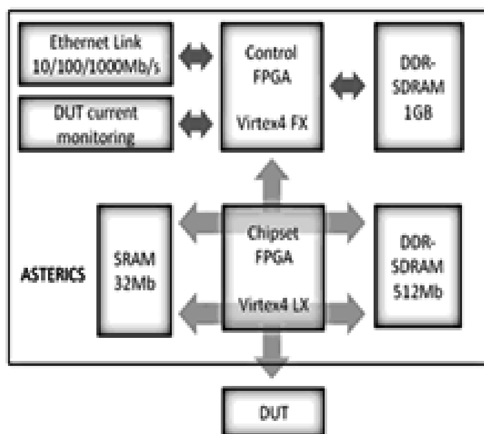


Fig. 4: Architecture of the ASTERICS tester

The DUT is connected to the tester via a high-speed connector. Up to 180 IOs are available with a voltage range from 1.2V to 3.3V selectable by software. Obviously, different kinds of memories (SRAM and DDR-SDRAM) are available for the user to cope with the different data rate requirements. A current monitoring circuit protects the DUT against latch-ups. Figure 5 depicts the ASTERICS tester.

An API embedding a set of functions (memory read/write, setting the current consumption limit, resetting the DUT etc) was developed. The user has just to call these functions. The API is available as a DLL for Windows and will be soon ported to Linux.

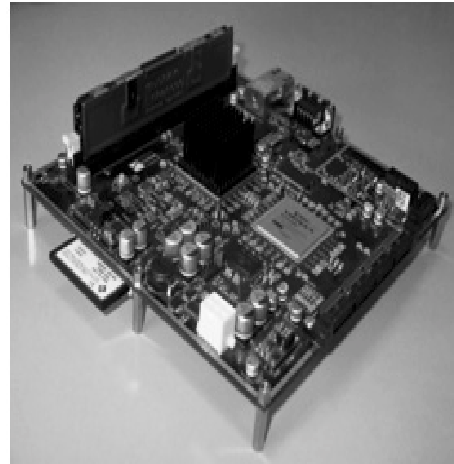

Fig. 5: The ASTERICS tester

The Ethernet link of the tester allows the user to remotely control the experiment anywhere in the world whenever an internet connection is available.

## 4 Radiation ground testing

The strategy to characterize the behavior of a DUT in a radiation environment consists of exposing the DUT to a radiation similar to those of the final environment, while it executes a test sequence.

### 4.1. The test stimuli

Because of the increasing complexity of current microprocessors, designing and performing radiation tests on them becomes a non-trivial task. As different programs activate in a different way the sensitive parts of a DUT, the total upset cross-section (i.e. average error rate par particles) will strongly depend on the executed test program. Therefore, there is no standard way to test microprocessors and no absolute value for the SEU cross-section. Published data on the upset sensitivity of various processors [8][9][10] has been generally obtained from radiation ground testing in which the test program executed by the DUT consists of checking sequentially in a continuous loop until an error is detected, each of the sensitive memory element accessible to the user. Such test programs, generally called "register test ", intensively exercise the microprocessor registers using a reduced subset of the instruction set. The final application software will exercise the processor registers with a given unknown rate, activating probably other sensitive elements. The question addressed is: how significant is the sensitivity issued from register test program if compared to the one of the final application?

In order to provide an objective feedback about these two strategies, both a classic register test program and a set of application-like programs or benchmarks were used to perform SEU testing. The benchmarks are programs implementing complex operations such as FFT computation, sort of an array of elements, matrix inversion, and CRC computation.

### 4.2. Radiation Ground testing results

The setup was used to test two RISC SPARC MHS90C601 and 602. The cross sections obtained with the Tandem Van de Graaff heavy-ion accelerator of the IPN, are plotted in Figure 6.

The various benchmarks programs show very close sensitivities. In addition it is noted that, as expected, the use of a registers test program type led to a significant over-estimating, of at least one order of magnitude, of the DUT sensitivity.

The results obtained using a Californium fission-decay source facility of the ONERA/CERT/DERTS follow similar tendency.

This over-estimation can be explained by the fact that this circuit has a great number of over exposed general purpose registers during the static registers test. For the other programs, conceived to fulfill particular functions, only some registers are sensitive during very short times. Their contents are often used and thus reallocated.

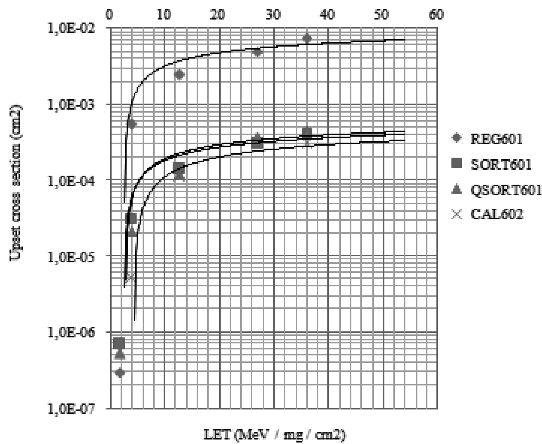


Fig. 6: MHS90C601upset cross-sections

The only detected errors are the upsets occurred on the memory elements containing values which will be exploited in the continuation of the program. There is an effect of upset masking.

The PPC7448 tests were performed at the HIF cyclotron accelerator of the UCL using the ASTERICS platform.

In order to ensure a good measure of the static cross-section (the sensitivity of the different memory cells accessible by the instruction set), the execution flow of the program was split into 2 phases.

The first is the startup phase where the processor is configured. The second is the running phase where the memory cells are filled with a pattern, irradiated and dumped into an external memory. These two phases are measured with counters implemented into the FPGA. If a counter value is different of the one expected, the associated run is discarded. This method allows knowing the real fluency seen by the processor; indeed, the particles hitting the DUT during result checking should not be taken in consideration.

Fig. 7 depicts the measured dynamic cross-section obtained when testing with a real space application, AOCS (Attitude and Orbit Control System), provided by the CNES and compared to a register test.

Obtained results show the high impact of the Data cache in the AOCS application sensitivity and the significant overestimation issued from registers test programs.

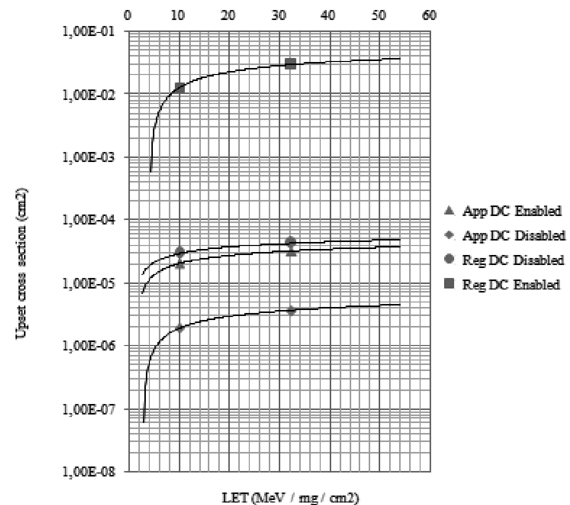


Fig. 7: Cross-sections obtained for the PPC7448

### 5. Cross-section estimation

To confirm the disparity of the sensitivities obtained using the two types of test program (registers test and application type), we calculated the sensitivity of the circuit for both kind of programs as proposed in [11].

In fact, the sensitivity of a circuit for a given program is function of the occupancy rate called "duty factor" of the significant elements (i) used by this program. The occupancy rate is defined as the sum of the percentages of times (compared to the

total duration of the program) during which the element is sensitive to radiations and where upsets on this element can be observed there (by causing noticeable application errors).

If we consider a program which at the instant (t1) loads a register by a value, uses it after some instructions at the instant (t2), and does not use it later on, this register is sensitive during all the duration (T) of the program, but indeed it does contribute to the sensitivity of the circuit, only between the instants (t1) and (t2). The occupancy rate of this register for this program is thus:

$$f_r = \frac{t_2 - t_1}{T} \qquad (2)$$

For a realistic characterization of the circuit for a given application program it is necessary:

- to determine the global cross sections of the circuit using the register test program,
- to determine the occupancy rates of the various elements, imposed by the application program,
- finally, to calculate the total sensitivity of the circuit as being the sum of the sensitivities of the various elements times the respective occupancy rates:

$$\sigma_{PROG} = \sum_{i} \sigma_i \times f_i \qquad (3)$$
$$f_i \leq 1$$

The equation (3) can be replaced by the following equation, if the cross section per bit ($\sigma_{bit}$) is established:

$$\sigma_{PROG} = \sum_{i} \sigma_{bit} \times N_i \times f_i \qquad (4)$$
$$f_i \leq 1$$

Where $N_i$ is the number of bits of the element.

The cross section per bit is the ratio of the global cross section per the number of the circuit total number of memory bits. The equation (4) becomes:

$$\sigma_{PROG} = \frac{\sigma_G}{N_t} \sum_{i} N_i \times f_i \qquad (5)$$
$$f_i \leq 1$$

Where $N_t$ is the total number of memory bits.

The total number of memory bits of processor SPARC 601, provided by the manufacturer, is 5247. The number of bits known for the user (general purpose registers and control and status registers) is 4512. Then the difference between these two numbers is the unknown number of bits, estimated to 735.

The floating-point calculation among the various benchmark programs giving very close sensitivities. In addition to the relative simplicity of the instructions flow of this program, another advantage offered by the use of this processor is the fact that most instructions require one or two clock cycles.

We determined occupancy rates, fixed by the used test program, of the various sensitive elements of the considered processor. For that we followed the flood of instructions, during a simulated execution, and determined the occupancy rates of the various known registers for the user for the floating-point calculation program. The calculation of the sum of the occupancy rates of the various known memory bits, for the considered program, gives a value of 225,6. The cross section calculated for the considered program, becomes:

$$\sigma_{PROG} = \frac{\sigma_G}{N_t} \sum_{i} ( 225.6 + N_u \times f_u ) \qquad (6)$$
$$f_i \leq 1$$

where $N_u$ and $f_u$ are respectively the circuit number of unknown memory bits and the occupancy rates of the unknown memory bits.

Owing to the fact that the occupancy rates of these hidden memory bits cannot be precisely given, for the calculation we considered the two extreme cases of use of these points, 0 and 1. Two values are thus obtained, one 6 times and the other 25 times smaller than that measured using a test of registers. The real sensitivity of the circuit for the application should be framed by these two limits.

Figure 8 presents the cross sections measured during radiation tests while using the registers test program as well as the higher and lower limits of the calculated cross sections. It is noted that the cross section measured using the application program is well framed between the two extremes. This shows that the user unknown memory bits are mandatory for the calculation of the circuit sensitivity and that the sensitivity measured using a registers test program could lead to a significant over-estimate.

As stated before the goal is to evaluate the circuit sensitivity to considered radiation while it executes the final application program. Unfortunately this software is available only several months after the testing phase.

The evaluation with the method proposed in [11] will lead to a range of estimation between a minimum and a maximum limits values.

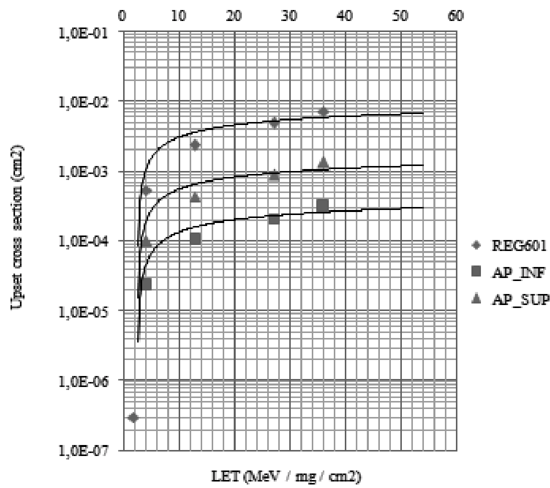


Fig. 8: Upset cross-section limits

The high costs related with the use of particle accelerators facilities to simulate the radiation environment make this kind of testing difficult to achieve. Moreover, any change in the software executed by the processor implies the realization of extra radiation ground testing campaigns, increasing the development costs. We have developed and validated a fault injection method in which the errors are simulated by hardware/software means, to accurately predict the error rate with respect to SEU faults for any application without redoing radiation ground testing experiments in which the final application is executed [16].

## 6. Fault injection for cross-section prediction

The tests were performed at the HIF cyclotron of the UCL. The device under test was the PPC7448. Its architecture includes two levels of caches and a huge register file making this processor potentially highly sensitive to SEUs. The test platform used for fault injection was ASTERICS tester.

### 6.1. Fault injection method

Three main approaches based on hardware or software strategies can be used for the simulation of SEUs. Among software ones, depending on the DUT available description level, three approaches can be used:

- SPICE level fault injection [12]
- VHDL/Verilog level [13]
- Use of a Instruction Set Simulator [14]

Emulating SEU faults at the hardware level for processors can be done by means of:

- FPGA: the RTL description of the target circuit is implemented with some modifications allowing the alteration of memory cells [15]
- The CEU (Code Emulated Upsets) method [16]

The CEU approach is based on the use of interrupt-like signals available in most of the processors. Indeed, the program code associated to the interruption routine can be used to alter a memory cell, randomly selected among those accessible through the instruction set. The randomness of the SEU's instants of occurrence can be easily implemented by random assess of the considered interrupt signal using a counter for instance.

A significant contribution of this method is related with the prediction of error rate combining static cross-sections issued from radiation ground tests and fault-injection results. Indeed, the static cross-section (eq.7) provides an estimation of the average number of particles needed to provoke an upset.

$$\sigma_{\text{STATIC}} = \frac{\#\,upsets}{\#\,particles} \quad (7)$$

From SEU fault-injection experiments performed while the target processor executes an application, can be derived τ which is the average number of upsets required to provoke a faulty behavior of the application (erroneous output, timeout, etc.).

$$\tau_{\text{INJ}} = \frac{\#\,errors}{\#\,upsets} \quad (8)$$

From equations (7) and (8) can be derived the cross- section of the application:

$$\sigma_{\text{SEU}} = \sigma_{\text{STATIC}} \times \tau_{\text{INJ}} \quad (9)$$

The final goal of the present work is to apply the CEU strategy to predict the error rate of a complex application, AOCS in our case, which, as told in section 4.2, is a CNES real flight software devoted to the orbital control of a satellite and adapted to run a complex processor using the ASTERICS platform.

### 6.2. Fault injection experimental results

Fault injection sessions were performed using the same test platform. Around 150000 faults were injected to get an evaluation of $\tau_{inj}$. From obtained $\tau_{inj}$ and static cross-sections was estimated the application $\sigma_{SEU}$ error rate using the eq. (9). Table 3 (respectively 4) shows the error-rate prediction versus the one obtained under radiation with data cache disabled (respectively enabled).

As shown in Table 3 and Table 4, the predictions are very close from the measures. It is important to note that owing to the high cost of beam time, often the measures under radiation are stopped after

reaching a few tens of errors; while error rates issued from CEU fault injection approach can be estimated from the results issued from very huge number of injected SEUs, thus exploring in a more extensive way the time-location fault space.

Table 3
Predicted vs. Measured Error Rate (Cache Disabled)

| Ion | σ Predicted($cm^2$) | σ Measured($cm^2$) |
|---|---|---|
| Argon | 1.96E-06 | 1.84E-06 |
| Krypton | 3.82E-06 | 3.56E-06 |

Table 4
Predicted vs. Measured Error Rate (Cache Enabled)

| Ion | σ Predicted($cm^2$) | σ Measured($cm^2$) |
|---|---|---|
| Argon | 2.12E-05 | 2.04E-05 |
| Krypton | 3.24E-05 | 3.17E-05 |

## 7. Conclusions and future work

This paper deals with the estimation of the vulnerability to soft-errors induced by ionizing radiation on processors. A generic approach is described. It combines the results issued from radiation tests aiming at the experimental evaluation of the sensitivity of the processor's various memory elements, to those issued from fault injection. This approach allows evaluating the studied processor vulnerability for any kind of target application program. An experimental setup was developed and used both for heavy ions testing and fault injection.

The vulnerability evaluation of processors using a registers test strategy over-exposes the registers during radiation tests. This led to an over-estimate of the processor final application sensitivity. In case of processors, a generic approach for SEU fault-injection so-called CEU was used. It is based on the activation at random instants of an interrupt signal to emulate the occurrence of a SEU in a randomly selected DUT memory cell. Fault injection results combined to static-cross sections issued from radiation ground testing provided error-rates estimations very close to those issued from running the same application under heavy-ion beams.

Future work aims at applying this approach to a very advanced many-core processor which may candidate to be part of an OBS (On Board Computer) of a nanosatellite which will be launched in the close future.


## Acknowledgements

Special thanks to all members of CNES, ONERA-CERT, IPN, and HIF for providing support for performing radiation ground testing. Thanks also to e2v staff for providing support on the PPC7448.


## References


1. R. Velazco and al, *Radiation effects on embedded system* (Ed Springer, 2007).
2. S. Karoui, *Vulnerability study of complex circuits to natural space radiations*, Ph.D *Thesis INPG*, Grenoble, December 1993.
3. D. Binder, E. C. Smith and A. B. Holman, *Satellites anomalies from galactic cosmic rays*, IEEE TNS, Vol. 32, December 1975.
4. C. S. Guenzer, A. B. Campbell and P. Shapiro, *Single event upsets in NMOS microprocessors and logic devices*, IEEE TNS, Vol. 32, December 1981.
5. R. Koga, W. A. Kolasinski, M. T. Marra and W. A. Hanna, *Techniques of microprocessors testing and SEU rate prediction*, IEEE TNS, Vol. 32, December 1985.
6. F.Faure, P. Peronnard, and R. Velazco, *Thesic+: A flexible system for see testing*, RADECS, 2002.
7. R. Huston, *Microprocessor testing: a testing turnaround - smart DUT runs the tester*, Fairchild Systems, Technical Bulletin*, n° 5*, 1975.
8. J. Cusik et al, *SEU vulnerability of the ZILOG Z-80 and NSC-800 microprocessors*, IEEE TNS, Vol. 32, No. 6, December 1985.
9. R. H. Sorensen et al, *The SEU risk assessment of the Z80, 8086 and 80C86 microprocessors intended for use in low altitude polar orbit*, IEEE TNS, Vol. 32, No. 6, December 1986.
10. R. Koga et al, *Heavy-ion induced upsets of microcircuits: a summary of the Aerospace Corporation test data*, IEEE TNS, Vol. 32, No. 6, December 1984.
11. H. Elder and al, *A method for characterizing a microprocessor's vulnerability to SEU*, IEEE TNS, Vol. 35, No. 6, December 1988.
12. L. W. Massengili, A. E. Baranski, D. O. van Nort, J. Meng, and B. L. Bhuva, *Analysis of single event effects in combinational logic - simulation of the ani2901 bit slice processor*, IEEE TNS, vol. 47, pp. 2609-2615, December 2000.
13. R. Velazco, A. Corominas, and P. A. Ferreyra, *Injecting bit-flips by means of a purely software approach: a case studied,* Defect and Fault Tolerance in VLSI Systems, 2002.
14. P. Fouillat, V. Pouget, D. Lewis, S. Buchner, and D. McMorrow, *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices, (*R. Schrimpf and D. Fleetwood, Eds. World Scientific), 2004, pp. 43-56.
15. R. Velazco, S. Rezgui, and R. Ecoffet, *Predicting error rate for microprocessor-based digital architectures through c.e.u. (code emulating upsets) injection*, IEEE TNS, vol. 47, pp. 2405-2411, December 2000.
16. S. Rezgui, R. Velazco, R. Ecoffet, S. Rodriguez, J.R. MINGO, *Estimating error rates in processor-based architectures*, IEEE TNS., vol. 48, pp. 1680-1687, October 2001.