Dear Chair man,

I would like to submit the paper titled “Injecting Multiple Upsets in a SEU tolerant 8051 Micro-controller” for an oral presentation at the Latin America Test Workshop 2002.

Thank you,
Sincerely,
Fernanda

ABSTRACT
This paper investigates the behavior of a SEU tolerant 8051-like micro-controller protected by single error correction Hamming Code in the presence of multiple upsets. Single event upsets (SEUs) and multiple bit upsets (MBUs) were analyzed, since they are more likely to occur in nano-metric technologies under high-energy heavy-ions. Upsets were randomly injected in all sensitive parts of the design, such as registers, flip-flops and memory, at a random time during the micro-controller application program execution. The experiment was emulated in a Virtex FPGA platform. Results evaluate the robustness of the tolerant 8051 in a multiple upsets environment.

CONTACT AUTHORS
Contact Author and presenter:
Fernanda Gusmao de Lima
Address: Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Campus do Vale - Bloco IV
Bairro Agronomia - Porto Alegre - RS -Brasil
CEP 91501-970 Caixa Postal: 15064
Phone: 55 (51) 33167036
Fax: 55 (51) 33167308
e-mail: fglima@inf.ufrgs.br

Luigi Carro
Address: Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Campus do Vale - Bloco IV
Bairro Agronomia - Porto Alegre - RS -Brasil
CEP 91501-970 Caixa Postal: 15064
Phone: 55 (51) 33167036
Fax: 55 (51) 33167308
e-mail: carro@iae.ufrgs.br

Raoul Velasco
Address: TIMA Laboratory
46, Av. Félix Viallet
38031, Grenoble - France
e-mail: Raoul.Velasco@imag.fr

Ricardo Reis
Address: Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Campus do Vale - Bloco IV
Bairro Agronomia - Porto Alegre - RS -Brasil
CEP 91501-970 Caixa Postal: 15064
Phone: 55 (51) 33167036
Fax: 55 (51) 33167308
e-mail: reis@inf.ufrgs.br
Injecting Multiple Upsets in a SEU tolerant 8051 Micro-controller

F. Lima\(^1\), L. Carro\(^1\), R. Velazco\(^2\), R. Reis\(^1\)

\(^1\)Federal University of Rio Grande do Sul
Av. Bento Gonçalves 9500, Bloco IV, Porto Alegre, RS, Brazil
<fglima, carro, reis>@inf.ufrgs.br

\(^2\)TIMA Laboratory
46, Av. Félix Viallet, 38031, Grenoble - France
Raoul.Velazco@imag.fr

ABSTRACT

This paper investigates the behavior of a SEU tolerant 8051-like micro-controller protected by single error correction Hamming Code in the presence of multiple upsets. Single event upsets (SEUs) and multiple bit upsets (MBUs) were analyzed, since they are more likely to occur in nano-metric technologies under high-energy heavy-ions. Upsets were randomly injected in all sensitive parts of the design, such as registers, flip-flops and memory, at a random time during the micro-controller application program execution. The experiment was emulated in a Virtex FPGA platform. Results evaluate the robustness of the tolerant 8051 in a multiple upsets environment.

I. INTRODUCTION

Integrated circuits operating in space applications are susceptible to transient faults called Single Event Upset (SEU), which are caused by charged particles generated by the Sun activity [1]. SEUs are a major concern in space environment, and have also been observed on the earth atmosphere as the result of interactions of neutrons [2]. SEUs are responsible for bit flips in MOS memory elements, and consequently, the SEU sensitivity is strongly correlated with the amount of storage elements present in the design, e. g. registers, memory bits, flip-flops, etc. The consequences of SEU depend on the nature of the perturbed information, ranging from erroneous results to system crashes.

Military, avionics and aerospace applications utilize advanced electronic systems with microprocessors in order to meet spacecraft requirements such as physical volume, weight, power and cost. Fault-tolerance and high-reliability have always been essential attributes of these systems, to keep them operational in such hostile environment. In [3], a SEU tolerant 8051-like micro-controller protected by Single Error Correction (SEC) Hamming code technique was proposed. The protected micro-controller has been analyzed in the presence of single event upsets by fault injection emulation in [4]. The results have shown the efficiency of the Hamming code technique to correct one SEU per application execution time.

This work continues the reliability evaluation of the SEU tolerant 8051 protected by SEC Hamming Code in the presence of upsets. Previous experiments were based on random injection of a single SEU per application execution, covering all registers, flip-flops and internal memory addresses. However, some space applications located in a high flux environment can be susceptible to more than one upset per micro-controller execution time. The goal of this work is to analyze whether upsets can be accumulated in the proposed tolerant architecture, when the time between upsets (TBU) is relative reduced. The results presented in this paper emulate the fault injection of two SEU per application run in four different TBU.

In addition, when a single high-energy charged particle hits the silicon in a high-density device, it can also provoke multiple bit upsets (MBUs) in adjacent cells. If multiple upsets are located in the same Hamming coded word, the mitigation technique will not be able to correct the upsets. The idea is to analyze the percentage of multiple upsets that can actually cause an error in the application. The results should be similar to the fault injection results obtained in [4], when single upsets were injected in the non-protected 8051 micro-controller.

The fault injection system proposed in [4] was improved in order to emulate more than one single upset per application execution time, and multiple upsets generated by the same charged particle. The approach injects transient bit flips on a selected target in a processor-like core, described in VHDL, concurrently with the execution of the application program. The full experiment was emulated in a Virtex FPGA platform, in order to speed up the fault injection process. Thousand of faults were injected in just some seconds, avoiding long simulation times.

This paper is organized as follows. The SEU tolerant 8051-like micro-controller architecture and the hamming code implementation are described in section II. Section III discusses different types of upsets that can be observed in the space environment and their effects on the 8051 architecture. Section IV presents an overview of the fault injection emulation method and its main improvements. Section V shows the fault injection results. Conclusions and future works are commented in section VI.

II. SEU TOLERANT 8051-LIKE MICRO-CONTROLLER

An 8051-like micro-controller developed at UFRGS and presented in [5, 6] was protected against SEU by using Hamming Code technique [3]. The microprocessor description is divided into six main blocks, illustrated in fig. 1. These units are Finite State Machine (FSM), Control Unit, Instruction Unit, Datapath, Data and Program memories. The FSM block generates 24-clock cycles in order to synchronize each instruction in the circuit operation. It has a very simple combinational logic and a set of flip-flops. The Control Unit generates some control signals for the datapath, and it is basically composed of multiplexors and registers. The Instruction Unit decodes the microcode word for each instruction, and it is also basically composed of multiplexors and logic gates. The Datapath includes an Arithmetic Logic Unit (ALU), logic gates, multiplexers and registers.

The mitigation technique used to correct single bit flips in the storage elements (registers or data memory cells) consists on codifying each register and memory element with Hamming code [7], and to correct the stored word by
decodifying every time this register or memory address is accessed. This technique can be applied to any sequential circuit described in VHDL, where the storage elements can be easily identified by the designer.

Fig. 1. General Scheme of the available 8051

The four mentioned structures in the 8051 VHDL description have been protected by single error correction Hamming Code. These blocks contain memory cells that can be affected by SEU radiation effects. The Instruction unit is a completely combinational structure and in principle it is not subjected to bit flips, requiring no protection schemes for it. The program memory usually is implemented in a ROM block that is immune to these faults, since its content can not be modified. In short, 128 bits from registers and 2048 bits from internal memory were protected by Hamming code.

Note that there is no scrub structure in order to avoid the accumulation of upsets. The Hamming decoder corrects a single upset in the registers and the correct value is used in the design. However, the correct result will be updated in the register just in the next write cycle. An upset in the memory will remain there until the address is re-written by the application. Although this implementation may not be totally efficient in a high particle flux where upsets may accumulate in registers or in the data memory, it presents some advantages, such as reduced area overhead and minimal performance penalty.

The scrubbing in the memory can be easily implemented by a dual port memory, an address counter and an extra logic to avoid writing conflicts. Scrubbing in the registers can be implemented by adding a new input path and an extra logic to avoid writing conflicts. The scrubbing rate should be higher that the upset rate in the design, synchronized by the system clock or its division.

A. Hamming Code Implementation

Hamming code is an error-detecting and error-correcting binary code that satisfies the equation \( d + p + 1 \leq 2^p \), where \( d \) is the number of data bits and \( p \) is the number of parity bits. Following this equation the Hamming code can correct all single-bit errors on \( d \)-bit words and detect double-bit errors when an overall parity check bit is used [7]. The steps to create the code word are described as follow. First mark all bit positions that are powers of two as parity bits (1,2,4,8,16, etc). All other bit positions are for the data to be encoded (3,5,6,7,9, etc). Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips. Position \( 1 \) checks 1 bit and skips 1 bit (1,3,5,7,9,11, etc). Position \( 2 \) checks 2 bits and skips 2 bits (2,3,6,7,10,11, etc). Position \( 4 \) checks 4 bits and skips 4 (4,5,6,7,12,13, etc). Position \( 8 \) checks 8 bits and skips 8 bits (8-15,24-31, etc).

The applied coding method is recommended for systems with low probabilities of multiple errors in the same coded word. For an 8-bit word, the SEC Hamming code represents 50% of more memory cells and extra combinational logic. The (12, 8) Hamming code calculation is illustrated in fig. 2.

![Hamming Code (12, 8) Calculation Schematic](image)

Fig. 2. Hamming Code (12, 8) Calculation Schematic

Hamming code can also correct double bit errors (SEC-DEC), however the algorithm is much more complicated because not all-double bit errors are deterministic, two different bit patterns can generate the same sums. The solution in this case is iterative. To find two bits errors we assume one bit in error and then solve for the second error as a single bit error. The procedure is to interactively move through the bits of the coded word changing each bit state. The new sums are calculated for the modified coded word. Then the single bit calculation above determines if this is the correct solution. If not, the bit is restored and the next bit is tried. Solutions in order to reduce the complexity can be studied as double bit errors in adjacent cells are represented as a subset of double errors in the Hamming coded word and consequently may reduce the area overhead in the upset correction.

III. UPSETS ANALYSIS IN THE 8051 ARCHITECTURE

The constant improvement in the technology process in the past few years has remarkably reduced the transistor geometry and power supply levels in the integrated circuits. In high-density circuits operating in low voltage, the memory cells are able to store information with less capacitance, which means that less charge or current is required to store the data. Unfortunately, a direct consequence is the increase in the device vulnerability to radiation, as charged particles that were once negligible are now much more likely to produce upsets [8]. Recent studies show that memory cells composed of transistors with lengths smaller than 0.25 \( \mu m \) and combinational logic composed of transistors with length smaller than 0.13 \( \mu m \) may be subjected to transient upsets either operating in the space environment or in atmosphere [9].
When a single charged particle strikes the silicon, it loses its energy via the production of electron-hole pairs, resulting in a dense ionized track in the local region. The energy transferred to the device is called Linear Energy Transfer, and it is measured by the incremental energy per unit length (MeV*cm²/mg). The ionization causes a transient current pulse. The SEU targets are drains of transistor in the OFF state. The minimum charge that a particle must deposit in a transistor drain in order to cause an SEU is denoted by critical charge. The critical charge must be bigger than the collector charge, which in turn is based on transistor parameters such as capacitance and voltage.

The frequency of single event upsets in the silicon depends of the charged particle flux. Trapped particles (protons, electrons and heavy ions) located in the Van Allen Belts present a flux ranging from 1 to 10⁶ particles/cm²/s. The electron energy is up to few MeV while the proton energy is up to hundred of MeV. Satellite orbits pass constantly to the Van Allen Belts. For long spacecraft missions, Galactic Cosmic rays and Solar Flare must be taken into account. Usually, they present a low flux but high-energy particles ranging from MeV to GeV [1].

According to the flux, it is possible to observe single or multiple SEUs per application execution. Multiple SEUs, either caused by a single charged particle or by two charged particles with certain time between upsets, can provoke error in the system protected by the single error correction Hamming cod. The error is due to the possibility of accumulation of upsets in the same coded word, which invalids the hamming code correction. This event is related to the hamming code implementation, the architecture and the application by itself.

Although SEU is the major concern in space application, multiple bit upsets (MBU) starts to be also a matter to be addressed nowadays because of the nano-metric technologies when a single high-energy ion can pass through the silicon energizing two adjacent memory cells [10]. MBUs can be induced by direct ionization or nuclear recoil, as it is presented in fig. 3 [11]. In [12], experiments in memories under proton and heavy ions fluxes have shown multiple upsets provoked by a single ion. MBUs were observed for all angles of incidence for LET greater than 25 MeV*cm²/mg. Based on the references, the majority of multiple upsets located in adjacent cells are provoked by a single particle. There is a very low probability of more than one charged particle interacting in adjacent cells, provoking upsets in a period smaller than 1 s.

![Two adjacent cells](image)

![Heavy Ion](image)

![Proton / Neutron](image)

![Nuclear recoil](image)

Fig. 3. MBU provoked by a single particle

Table 1 summarizes single and multiple upset events generated by a single charged particle in a SEC Hamming code structure. By definition, a single upset in a hamming coded word will always be corrected (tolerated upset). Two or more upsets in the same coded word will not be properly corrected. The time between upsets should be enough for the system to recover the upset, and so to avoid accumulated upsets in the same coded word. However, multiple upsets can be tolerated if that register or memory data is not being used at the moment by the application.

Case 1 in table 1 defines one single bit upset per application, which is always corrected by the SEC Hamming code. Case 2 defines two single upsets per application in different coded words, which is also always corrected by the Hamming code implementation. Case 3 shows two single upsets per application happening in the same coded word, with an interval between them (TBU). In this case, the upset may or may not be accumulated in the coded word, depending on the TBU and the refreshing rate of the design. Case 4 defines one multiple bit upset at different coded words. In this case, an ion hits two adjacent cells that belong to different words (scrambled layout). Because in the end, there is just one upset per coded word, the SEC Hamming code is able to correct the upsets. Case 5 addresses multiple bit upsets in the same coded word. In this case, the SEC Hamming code is not able to correct the double bit error. Consequently, the two phenomenon where the single error correction hamming code does not assure 100% reliability, case 3 and 5 in table 1, were tested in this work.

<table>
<thead>
<tr>
<th>Source</th>
<th>Upset</th>
<th>Time</th>
<th>Location</th>
<th>Case</th>
<th>Single Error Correction Hamming Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single charged particle</td>
<td>Single</td>
<td>One per application</td>
<td>Coded word</td>
<td>1</td>
<td>Design hardened</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Two or more per application at different time (high flux)</td>
<td>Different coded words</td>
<td>2</td>
<td>Design hardened</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Same coded word</td>
<td>3</td>
<td>Very low probability to cause an error, it depends to the time between upset and the design architecture ability to avoid accumulation of upsets.</td>
</tr>
<tr>
<td>Multiple</td>
<td>One per application</td>
<td>Different coded words</td>
<td>Same coded word</td>
<td>4</td>
<td>Design hardened</td>
</tr>
<tr>
<td></td>
<td>(high-energy particles, nano-technologies)</td>
<td></td>
<td></td>
<td>5</td>
<td>High probability to cause an error, it depends to the time occurrence in the application execution and design architecture.</td>
</tr>
</tbody>
</table>

Table 1 – Hamming Code Reliability and Upset Analysis in Space Environment
IV. FAULT INJECTION EMULATION EXPERIMENT

Fault injection is an attractive technique for design evaluation due to its high flexibility in terms of spatial and temporal information. Some works have been done in order to inject faults in high level descriptions [13, 14, 15]. The process involves inserting faults into particular targets in a system at a determined time in the process and monitoring the results to define its behavior in response of a fault. It has also a reduced turnaround time and evaluation cost compared to traditional radiation ground testing. Moreover, it has been shown that injecting fault in a programmable logic platform, after synthesizing the full system in a Field Programmable Gate Array (FPGA) can speed up the process in many orders of magnitude [4].

A fault injection system that emulates single event upsets in memory related components (single flip-flops or latches, registers and memories), in processor-like circuits was proposed in [4]. Because the proposed technique is completely parameterizable, it was possible to perform modifications in the VHDL code in order to inject as many upsets as wanted per execution. This new version takes into account two upsets at the same time in the same hamming coded word, and two upsets per application time also at the same hamming coded word, evaluating the time between upsets (TBUs).

The method does not insert faults in combinational logic. The whole system is a run-time fault injection mechanism that is performed during the prototyped execution without interrupting the micro-controller application. The system is divided into three main design blocks:

- **Fault injection Control block**: generates all the fault enable signals to all register and memories.
- **Device Under Test (DUT) core**: fault injection paths are added to the design in order to inject bit flips in all SEU sensitive parts.
- **Monitor block**: responsible to monitor the results of the DUT core in order to analyze the effects of each inserted fault.

The fault injection block has been modified in order to fit the new requirements. The system is synthesized in a single Virtex FPGA [14] in order to speed up the emulation process. In addition, the system takes advantage of the Virtex family embedded features, such as high performance, large amount of configuration logic blocks (CLBs) and dedicated internal RAM blocks, called SelectRAMs, with 4096 bits each, that can be configured as single or dual-port. The system also uses the Xilinx embedded logic analyzer called Chipscope [15] that is synthesized together with the design in the VIRTEX FPGA and monitored by a host computer.

A. Fault Injection System

The Fault Injection Control block is responsible to generate the time when a fault will happen and its specific location in the design. The method was designed to be as flexible as possible, in order to emulate the upsets occurring in the space environment. It pseudo-randomly generates 5 parameters per application execution, which are stored in parameterized size registers: time of occurrence, memory and register location, bit to be flipped and the choice between memory and registers. The parameters are generated according to a pseudo random sequence.

The Fault Injection Control block has been changed in order to inject multiple upsets. The first modification provides two upset fault injections at the same time at the same coded word by changing the mask process. In this case, the mask flips two adjacent bits instead of only one presented in the previous approach. The second modification permits two subsequent upsets in the same coded word separated in time by a random period. In this case there is two timers, one for the first upset and other for the second upset and two flip bit masks. The time between upsets is adjustable. We report results based on four different time between upsets.

In order to inject faults in all storage elements of a design, it is necessary to add special paths in the high-level description to assure fault controllability. In the case of processor-like circuits, all storage elements can be classified as registers and memory. The proposed method replaces the normal single port memory block (SelectRAM) by a dual-port memory, as illustrated in Fig. 4.a. Port B performs the fault injection in order to avoid interruption in the normal microprocessor operation. The write signal from the microprocessor is compared to the fault injection write signal, to avoid conflicts in the memory.

The internal registers receive one extra multiplexor and xor logic gate, as shown in Fig. 4.b. In order to perform fault injection in registers, it is necessary to write the new corrupted value in the register by the fault enable signal. Results have shown no impact in performance.

The Monitor Block observes the application results by reading all the internal memory serially. In the fault injection emulation, the micro-controller is always reset between application executions. When the system resets the microcontroller, all registers are initialized, however the memories that are implemented using the Virtex SelectRAM blocks can receive their initial values just once when the device is customized. In order to assure that no previous upset is going to remain in the memory in each new execution, the Monitor Block clears the SelectRAM memory in the beginning of each execution.

B. Fault Injection Platform

The fault injection platform consists in a “golden”, or reference chip method, where two identical microprocessors are operating in tandem, one exposed to the fault injection and the other not. An external controller continuously compares the outputs indicating any occurrence of error or lost of sequence. All logic is located in the same FPGA. Fig. 5.a presents the fault injection system schematic. The fault injection test platform used in the 8051-like micro-controller experiment, shown in Fig. 5.b, is made from one AFX V300PQ240-100 daughter card, a JTAG cable used as an interface to a host PC, and a control panel. The system operates in conjunction with a host PC in order to collect the data in the Chipscope analyzer tool.
IV. FAULT INJECTION RESULTS

The 8051 micro-controller application used in the fault injection experiment is a 6x6 matrices multiplication program. This application performs the multiplication by using shift and add algorithm. The prototyped machine in the Virtex FPGA can run at 10MHz. This means that, to run the matrices multiplication program, one needs roughly 5ms.

A total of 24,576 random single or double upsets were injected in the SEU tolerant 8051 micro-controller protected by SEC Hamming code, using the matrices multiplication benchmark. According to the consequences of injected upsets, obtained results were classified into the three following types: tolerated upsets; errors in the memory and lost of sequence.

Tolerated upsets correspond to those single upsets injected on memory elements that were corrected by the hamming code. Also, this set includes those multiple upsets injected on memory elements whose content was not relevant for the application. For instance, this could be a register that is not used in the application, or a register which will be written after the fault occurrence, thus overwriting the fault, or a memory address that will be written after the fault occurrence.

Errors in the memory consist of any discrepancy between the reference core memory value and the DUT memory value. These errors will happen every time a multiple fault is inserted in a register that is being used for the application or in a memory address that the hamming code is not able to correct.

Finally, a lost of sequence is the case when the reference processor finishes its application before the DUT. These malfunctions are unrecoverable, needing a hardware reset to restart program execution. The main reasons for the occurrence of lost of sequence are multiple upsets in registers that are counting for loops, multiple upsets in the program counter, multiple upsets in the interruption logic or multiple upsets in the operators stored in the memory. When an upset affects the operators stored in the memory, it might provoke an extended loop in the multiplication routine, which is composed by addition and shifts.

Tables II summarizes the obtained experimental results in the SEU tolerant 8051 protected by SEC Hamming code in presence of single and double upsets. The experiment 1 emulates single bit upsets, one per application. SEU were injected in registers and memory in the proportion shown in the table. Because all sensitive parts of the 8051 are protected by Hamming code, the results show that all upsets were tolerated.

Experiments 2 and 3 report the injection of two bit flips in two adjacent cells, located in the same Hamming coded word, at the same time, in order to emulate MBUs. Two upsets were injected per application at the same time in two adjacent cells located in the same coded word. Experiment 2 emulates MBUs just in the internal memory, while experiment 3 emulates MBUs in the full micro-controller. The results match to the fault injection values obtained in [4], when single upsets were injected in the non-protected 8051 micro-controller. The tolerated errors are from upsets in those elements whose content was not relevant for the application or value was updated before being used.
TABLE II – Fault Injection Results in the SEU Tolerant 8051-like Micro-Controller (upsets in R= registers, M= memory)

<table>
<thead>
<tr>
<th>Case</th>
<th># Faults per application</th>
<th>Time</th>
<th>Total number of application program executions</th>
<th>Errors in the memory</th>
<th>Lost of sequence</th>
<th>Tolerated Upsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>3072, upsets in R (192), M (2880)</td>
<td>0</td>
<td>0</td>
<td>3072 (100%)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>T1; T2=T1</td>
<td>3072, upsets in R (0), M (3072)</td>
<td>1891 (62%)</td>
<td>362 (12%)</td>
<td>819 (26%)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>T1; T2=T1</td>
<td>3072, upsets in R (212), M (2860)</td>
<td>1782 (58%)</td>
<td>384 (13%)</td>
<td>906 (29%)</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>T1'; T2=T1'+TBU</td>
<td>12288, upsets in R (745), M (11543)</td>
<td>6109 (50%)</td>
<td>811 (7%)</td>
<td>5368 (43%)</td>
</tr>
<tr>
<td></td>
<td>8191 &lt; TBU ≤ 16656</td>
<td></td>
<td>5635, upsets in R (374), M (5261)</td>
<td>2683 (48%)</td>
<td>366 (7%)</td>
<td>2586 (45%)</td>
</tr>
<tr>
<td></td>
<td>4095 &lt; TBU ≤ 8191</td>
<td></td>
<td>3052, upsets in R (178), M (2874)</td>
<td>1694 (55%)</td>
<td>208 (7%)</td>
<td>1150 (38%)</td>
</tr>
<tr>
<td></td>
<td>1023 &lt; TBU ≤ 4095</td>
<td></td>
<td>2286, upsets in R (145), M (2141)</td>
<td>1286 (56%)</td>
<td>161 (7%)</td>
<td>839 (37%)</td>
</tr>
<tr>
<td></td>
<td>0 &lt; TBU ≤ 1023</td>
<td>811, upsets in R (48), M (763)</td>
<td>446 (55%)</td>
<td>76 (10%)</td>
<td>289 (35%)</td>
<td></td>
</tr>
</tbody>
</table>

It is important to notice the great influence of the memory in the results, as the values do not significantly change when MBUs are injected only in the memory or in the memory and registers. The results are very similar to the fault injection results in the non-protected 8051. Based on previous results [4], we expect more than 96% of the MBUs to be tolerated when a SEC-DEC Hamming code is implemented in the internal memory.

Experiment 4 emulates two single bit upsets in sequence per application at the same coded word. In this case, the time between upset (TBU) is taken into account. The experiments were tested with a TBU ≤ 16656 clock cycles using a frequency of 10 MHz. All observed errors and lost of sequences are due to upsets in the memory. The results show that the refresh of the memory is important during the application program execution in order to avoid accumulation of upsets. Using refresh in the memory, no errors or lost of sequence were observed.

V. CONCLUSION

This paper presented experimental results in the SEU tolerant 8051 protected by SEC Hamming code in presence of upsets. Results show the 8051 robustness to single event upsets when upsets occur once per application. However, we demonstrated that the SEU tolerant micro-controller is not robust to MBUs, or multiple SEU per application, when scrubbing in the memory is not used. Experiments 2 and 3 in table II show the high memory sensitivity in the whole design reliability. Solutions must be studied in order to avoid MBUs.

One solution to avoid errors in presence of MBUs is to apply SEC-DEC Hamming code in the memory and registers. Another option is to design scrambled memory cells in the layout floorplanning, avoiding the placement of two adjacent cells from the same Hamming coded word. Both solutions can introduce penalties in the area and performance, and studies must be done on this matter. Assuming that MBUs can be more frequent in memories than registers because of their high-density designs, SEC Hamming code can be applied in the registers while SEC-DEC Hamming code can be applied in the memory. In addition, a simplified version of the SEC-DEC Hamming code can be used in assumption of only two adjacent errors. Future work aims at analyzing the advantages and drawbacks of these techniques in the SEU tolerant 8051 micro-controller.

ACKNOWLEDGMENTS

The authors would like to thank Xilinx and CNPq Brazilian Agency for their support of this work.

REFERENCE

Controller Tolerant to Transient Faults”, JETTA, 2000.