Coping With SEUs/SETs in Microprocessors by Means of Low-Cost Solutions: A Comparison Study

M. Rebaudengo, M. Sonza Reorda, M. Violante, B. Nicolescu, and R. Velazco

***Abstract*—In this paper, two low-cost solutions devoted to provide processor-based systems with error-detection capabilities are compared. The effects of single event upsets (SEUs) and single event transients (SETs) are studied through simulation-based fault injection. The error-detection capabilities of a hardware-implemented solution based on parity code are compared with those of a software-implemented solution based on source-level code modification. Radiation testing experiments confirmed results obtained by simulation.**

***Index Terms*—Error detection, microprocessor, radiation testing, SET, SEU, simulation-based fault injection.**

I. Introduction

THE increasing popularity of low-cost safety-critical computer-based applications in new areas (such as automotive, biomedical, telecontrol) requires the investigation of new design methods and techniques for guaranteeing dependability. Recently, hardened versions of standard processors were introduced, embedding mechanisms to guarantee the correct program behavior even in the presence of soft errors [1]. Low-cost solutions often adopt error-detection mechanisms (EDMs) based on parity check, while high-end processors are also equipped with more complex methods for detecting and correcting soft errors in combinational logic [2].

The continuous increase in the integration level of electronic systems is making it more difficult than ever to guarantee an acceptable degree of reliability due to the occurrence of faults and soft errors that cannot be modeled and that can dramatically affect the system behavior. As an example, the decrease in the magnitude of the electric charges used to carry and store information increases the probability that alpha particles and neutrons hitting the circuit could introduce errors in its behavior [2]. It is worthwhile to note that when deep submicron technologies are used, both memory elements [3] and combinational gates are sensitive to soft errors [4]. In nanometer technologies, transient pulses induced by soft errors are longer than the propagation delay through a gate. A soft error can thus propagate without masking and can affect the correct circuit behavior. Moreover, the reduction in clock period increases the probability that a soft error could be latched. Due to these trends, it is expected that the sensitivity to single event transients (SETs) of combinational elements in future technologies will no longer be negligible [4]. As a consequence, solutions are required to harden circuits and systems against single event upsets (SEUs) and SETs.

When cost is a major concern, designers tend to adopt commercial hardware even in the case of safety critical designs. In the case of microprocessors, the adoption of a hardened version equipped with EDMs is often too expensive and alternative solutions are required. Modular redundancy is often used and involves the replication of commodity hardware components. The solution is however expensive, since it requires replicated modules and a system overhead to control the operation correctness. In this context, *software implemented fault tolerance* (SIFT) is becoming an attractive solution since it allows the implementation of dependable systems without incurring the high costs associated with the need to design custom hardware or the use of hardware redundancy. Nevertheless, relying on software techniques for obtaining dependability often means accepting additional overhead penalties in terms of code size and reduced performance. However, in many applications memory and performance constraints are relatively loose, and the idea of trading reliability against speed is often acceptable.

Several reports have recently proposed new SIFT techniques [5]–[7] that address SEUs affecting application code and data segments that are complementary to older approaches such as algorithm-based fault tolerance (ABFT) [8] or control flow check [9]. Conversely, evidence of the SIFT capabilities of detecting transient faults affecting the processor internal components, its internal memory elements (i.e., register file, control, and pipeline registers), and its combinational logic, is still missing.

The purpose of this paper is to address the aforementioned issue from an experimental point of view. Fault injection experiments have been performed to evaluate the capabilities of the SIFT technique proposed in [5] of detecting transients faults in the internal memory elements of a processor and in its combinational logic. Therefore, we focused both on SEUs, which result in the modification of the content of a memory cell and on SETs, which result in the modification of the output of a combinational gate within a circuit. These perturbations are the result of the ionization generated by either incident charged particles or daughter particles created by the interaction of energetic particles (i.e., neutrons) and atoms present in the silicon

Manuscript received December 21, 2001; revised March 13, 2002. This work was supported in part by the Italian Space Agency (ASI) and by the Italian Ministry for Education, University, and Research (MIUR) through the CERCOM Research Center.

M. Rebaudengo, M. Sonza Reorda, and M. Violante are with the Dipartimento Automatica e Informatica, Politecnico di Torino, Torino I-10129, Italy (e-mail: reba@polito.it; sonza@polito.it; violante@polito.it).

B. Nicolescu and R. Velazco are with the TIMA Laboratory, Grenoble F-38031, France (e-mail: Bogdan.Nicolescu@imag.fr; Raoul.Velazco@imag.fr).

Publisher Item Identifier S 0018-9499(02)05831-8.

substrate. When traditional very large scale integration (VLSI) technologies are used, SETs can be neglected since their effects are usually filtered by the propagation delay of combinational gates. However, SET effects cannot be ignored in deep submicron technologies [4]. In our experiments, we adopted the technique described in [5] because the application we studied is not suitable for the adoption of other SIFT approaches (such as ABFT or control flow check).

In our experiments, we also studied the error-detection capabilities offered by a hardened processor whose internal memory elements have been protected against SEUs by means of parity codes. This study allows us to better understand the SIFT error-detection capabilities by comparing it with a well-known hardware-based approach for guaranteeing safety.

All our experimental data were obtained by means of a simulation-based fault injection tool that supported the injection of SEUs and SETs in the gate-level model of a processor running a program.

The models for SEUs and SETs are known to have a good correlation with the effects of real particles hitting a circuit [10], [11], provided that suitable circuit descriptions are available. In order to fruitfully exploit the fault models, we need processor descriptions that capture as many details of the corresponding hardware as possible. In our experiments we adopted the gate-level model of an 8051 processor core obtained by synthesizing a soft-core implementing the whole 8051 instruction set. The model we are using thus embeds detailed structural information about the analyzed processor.

The figures we gathered show that the SIFT approach, even if not yet able to achieve complete fault coverage, effectively detects most of the transients induced by charged particles in a processor-based system.

II. Benchmark Application

The application we considered is part of an industrial system and is based on an Intel 8051 processor running a communication protocol. The processor receives information from an analog to digital conversion system, applies a simple scaling algorithm and sends the results to a master controller with a bit rate of 50 kbit/s over an I^2C bus. The I^2C (*interintegrated circuit*) bus [12], developed by Philips, allows integrated circuits to communicate directly with each other via a simple bi-directional two-wire bus including a serial clock line (SCL) and a serial data line (SDA). Nowadays, the I^2C bus is becoming a standard bus system that is used in consumer electronics, telecommunications, and industrial electronics.

The code implementing the communication protocol is written in standard C language and amounts to about 110 lines of code. A gate-level model of an 8051 processor running at 30 MHz and equipped with 32 kbyte of program memory is exploited, which guarantees a maximum bit rate of 120 kbit/s.

We considered three versions of the system; they are described in the following subsections.

A. Unhardened System

The system is composed of an unhardened 8051 processor executing the unhardened code of the I^2C communication protocol. The system does not embed any EDM.

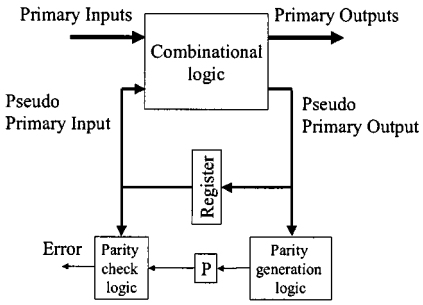

Fig. 1. Parity-based hardening mechanism.

B. Hardware-Implemented EDM: Parity Codes

The system is composed of a hardened 8051 processor executing the unhardened code implementing the I^2C communication protocol. Fig. 1 presents the architecture of the parity-based detection mechanism of the hardened 8051 processor: for every register, a flip-flop P stores the parity bit computed by the *parity generation logic*, and an *error* signal is computed by the *parity check logic* on the basis of the P flip-flop and the content of the hardened *register*.

The processor is thus able to detect every SEU affecting its internal memory elements.

C. Software-Implemented EDM: SIFT

The system is composed of an unhardened 8051 processor executing the hardened code of the I^2C communication protocol.

The hardened version of the I^2C code is obtained by applying the methodology described in [5]. This SIFT approach is based on introducing data and code redundancy according to a set of transformation rules. Code transformations were designed to be applied on high-level code and to detect upsets affecting both data and code. The first idea implemented by the proposed transformation rules is that any variable used by the program must be duplicated and the consistency between the two copies must be verified after any read operation on the variable. In this way, any fault affecting the storage elements containing the program data is detected. The second idea is that any operation performed by the program must be replicated and the results of the two executions must be verified immediately for coherency. In this way it is possible to detect faults affecting the storage elements containing the program code or the processor executing the program. Finally, some rules are proposed to verify that the execution flow is the expected one.

A major novelty of this strategy relies in the fact that being based on a set of simple transformation rules, their implementation on any high-level code can be completely automated. This frees the programmer from the burden of guaranteeing the application robustness against errors and drastically reduces the costs for program hardening.

The transformations are applied to the I^2C in a fully automated manner exploiting a tool we wrote for this purpose [13]. The binary code of the hardened program (obtained from the compilation of 200 lines of the hardened C code program) fits in the 8051 internal memory space, so in the studied application the method does not introduce any hardware overhead.

As already observed in [5], the SIFT approach may degrade performance: in this case, this means reducing the maximum bit rate the program is able to guarantee, which falls to 60 kbit/s. This speed still satisfies the required data transfer rate and, thus, no performance loss is observed.

III. Simulation-Based Fault Injection Environment

When defining the fault injection environment, we were driven by two constraints: the need for easily accessing the processor gates and memory elements and the need for easily introducing parity codes in the 8051. We addressed these issues by developing a simulation-based fault injection environment, where an in-house developed event-driven parallel fault simulator is used to simulate a gate-level model of the 8051, as proposed in [14]. The fault simulator has been instrumented in order to support the injection of SEUs and SETs.

The tool classifies fault effects according to the following categories.

1) *Wrong answer*: The fault is not detected by any EDM, but the result is different from the expected one.
2) *Latent*: The fault does not affect the program behavior, but at the end of the experiment, the value of at least one memory element inside the processor is different from the expected one.
3) *Effectless*: The injected fault does not affect the program behavior nor the final value of memory elements.
4) *Detected*: The fault triggers an EDM.

The fault injection tool amounts to about 10 000 lines of C code, including the fault simulator and has been developed under the Solaris operating system.

IV. Experimental Results

In order to evaluate the effectiveness of the SIFT approach in dealing with transient faults affecting a processor during the execution of a program, we first performed a set of fault injection experiments whose results are summarized in Section IV-A.

We then performed additional experiments in order to compare the error-detection capabilities of the SIFT approach with respect to the parity-based mechanism we described in Section II-B. The results we obtained are reported in Section IV-B.

Finally, we performed some radiation testing experiments in order to validate the results gathered by means of simulations. The obtained results are reported in Section IV-C.

A. Evaluating the SIFT Technique

In order to evaluate the effectiveness of the SIFT approach, we used the simulation-based fault injection tool described in Section III to perform two sets of experiments.

A first set targeted SEUs in the processor internal memory elements. Faults were randomly selected both in space and time. In particular, the fault location was identified by randomly selecting one of the memory elements among those inside the 8051 (register file, status register, register of the control unit). Moreover, injection time was randomly selected during the whole program execution time.

TABLE I
Fault Injection in the 8051

	SEU		SET	
Fault Effects	Unhardened system	SIFT	Unhardened system	SIFT
TOTAL	1,000	1,000	1,000	1,000
Wrong answer	234	39	104	11
Latent	8	63	8	27
Effect-less	758	768	888	914
Detected	0	130	0	48

A second set of experiments targeted SETs in the processor combinational logic. As in the previous case, faults were randomly selected both in time and space, but this time, gates instead of memory elements were selected.

The results we gathered are reported in Table I, where numbers for the unhardened system are also reported for comparison purpose.

For both fault models a high percentage of the injected faults do not modify the processor behavior. This may be due to the following effects:

1) SEU is injected in a register whose value is overwritten before the register is first used;
2) combinational logic inhibits the propagation of the SET toward the processor memory elements and the processor outputs.

As far as the unhardened system is concerned, we note that most of the remaining faults produce wrong answers. The SIFT approach is able to detect them no matter which fault model is used. It is able to detect both SEUs and SETs, thus showing a high degree of flexibility.

A few faults escape the SIFT EDM. They are faults that cause the processor to execute unexpected branches that are not covered by the SIFT EDM.

Finally, we observed an increase in the number of latent faults for the adopted SIFT technique. This result is mainly due to the fact that this technique checks the integrity of each variable before use and there are faults that affect some memory elements after their last usage by the program.

B. Comparing Alternative EMD Approaches

In order to better understand the effectiveness of the SIFT approach, we compared it with a very simple hardware-implemented EDM that consists in hardening every memory element of the processor with a parity code as described in Section II-B.

The parity-based mechanism only hardens memory elements against SEUs, while SETs in the combinational logic cannot be detected. This can be explained by noting that SETs injected in the combinational logic affecting the pseudoprimary output lines (Fig. 1) are considered by the *parity generation logic* as valid data and, thus, the fault escapes the detection mechanism.

When making the comparisons between the SIFT and the parity-based approach, we thus considered only the SEU fault model. We performed a new series of fault injections in the system implementation described in Section II-B by exploiting our simulation-based fault injection tool. As in the previous case, faults were randomly selected in both time and space.

TABLE II
FAULT INJECTION IN THE 8051 INTERNAL STORAGE ELEMENTS

Fault effects	Unhardened system	Parity hardened	SIFT
TOTAL	1,000	1,000	1,000
Wrong answer	234	0	39
Latent	8	0	63
Effect-less	758	0	768
Detected	0	1,000	130

TABLE III
ERROR RATE

f_{SEU} [SEUs/day]	Unhardened system [1/day]	Parity hardened [1/day]	SIFT [1/day]
5.51E-3	1.29E-3	0	2.15E-4
2.33E-2	0.54E-2	0	9,09E-4

Table II reports the results for the three-system implementations we described in Section II.

As expected, the parity-based detection mechanism is the most effective. It is able to correctly detect all the faults injected in the 8051 memory elements, even if most of them do not modify the program execution in the unhardened system.

Given the figures of Table II, it is possible to compute the error rate [10] for the three-system implementations. The error rate can be computed as follows:

$$ER = f_{\mathrm{SEU}} \cdot P_{\mathrm{WA}} \tag{1}$$

where f_{SEU} is the frequency of occurrence of SEUs, while P_{WA} is the probability for a SEU to produce a wrong answer. We computed the latter term from Table II as the number of wrong answers over the number of injected faults. As far as the frequency of occurrence of SEU is concerned, we adopted the figures reported in [16].

By exploiting (1), we obtained the figures reported in Table III.

Table III shows that the SIFT approach reduces the error rate of the system by a factor of six; even if the approach is still far from being exhaustive, as in the case of the parity hardened system, it is nevertheless able to significantly improve the system safety. As an example, the number of errors falls from one error every two years (for 5.51E-3 SEUs/day) to one error every 13 years.

To compare the detection capabilities of the parity hardened system with that of SIFT we also computed the corresponding detection rate as follows:

$$DR = f_{\mathrm{SEU}} \cdot P_D \tag{2}$$

where f_{SEU} is the frequency of occurrence of SEUs, while P_D is the probability for a SEU to trigger the EDM the system embeds. It is computed as the number of detected faults over the total number of injected faults.

Table IV reports the detection rate for the two hardened systems as well as the error rate of the unhardened system.

In order to obtain a safe system, we have to devise an EDM whose detection rate is at least equal to the error rate of the unhardened system. By analyzing the detection rate reported in Table IV, we can observe the following.

1) The parity hardened system has a detection rate much higher than the required one. As a result, most of the detected faults are likely to have no effect. The impact of this effect on the system performance depends on the error recovery strategy. If the procedure activated after such an error is detected is time consuming, the system is likely to waste a significant amount of time in recovering.
2) The SIFT system shows a detection rate lower than the computed error rate. On the one hand, this implies that the approach has to be improved before being adopted in critical systems. On the other hand, it is able to classify fault more efficiently than the parity based approach. It is thus able to minimize the time spent for error recovering in case of time-consuming error handling strategy.

TABLE IV
ERROR RATE VERSUS DETECTION RATE

f_{SEU} [SEUs/day]	Error rate of the unhardened system [1/day]	Detection rate of the parity hardened [1/day]	Detection rate of the SIFT [1/day]
5.51E-3	1.29E-3	5.51E-3	0.72E-3
2.33E-2	0.54E-2	2.33E-2	0.30E-2

TABLE V
RADIATION TESTING RESULTS

	Unhardened system	SIFT
Flux [$\frac{particles}{cm^2 s}$]	5,000	5,000
Number of SEUs	53	68
Detected	0	60
Wrong answer	53	8

C. Radiation Testing Experiments

To get more confidence on the results coming from simulations, we performed some radiation testing with the Cyclotron facility available at Louvain-La-Neuve, Belgium, using the THESIC platform [15] for the hardware and software test experiment setup. Argon heavy ions (LET 14.1 MeV/mg/cm^2) were used during radiation testing.

Due to the lack of the parity-hardened 8051 device, we only tested the unhardened and the SIFT system implementations, which exploit a standard (unhardened) 8051: during the experiments an Atmel 89C52 device has been used. The gathered measures for the two systems are summarized in Table V, where the flux to which the processor was exposed, the total number of SEUs observed and the fault effects classification are reported. The number of SEUs is measured by analyzing the content of the processor memory elements at the end of the program execution. As a result, this figure corresponds to the number of SEUs producing some observable effects on the processor behavior.

The figures reported in Table V confirm the effectiveness of the SIFT approach in dealing with transient errors originated inside processor memory elements due to the effects of impinging

TABLE VI
RADIATION TESTING RESULTS VERSUS SIMULATION RESULTS

	Radiation testing		Simulation	
Fault effects	Unhardened system [%]	SIFT [%]	Unhardened system [%]	SIFT [%]
Wrong answer	100.0	11.8	100.0	23.1
Detected	0.0	88.2	0.0	76.9

particles: SIFT is indeed able to detect 88.2% of the upsets observed in the processor, while the escape rate (e.g., SEUs that provoke unexpected behaviors) is 11.8%.

The analysis of Table V validates the results simulation provides, too. Table VI reports the percent number of wrong answers and detected faults for the unhardened system and the SIFT one when radiation testing and simulation are considered. Note that these figures are computed as the number of faults classified as "detected" or "wrong answer" over the number of SEUs, which have some effect on the processor behavior. As far as the simulation is concerned, this number is computed as the total number of injected SEUs minus the number of effectless faults.

Table VI shows that a good correlation exists between radiation testing and simulation.

V. CONCLUSION

In this paper, the effects of SEUs and SETs in a microprocessor executing an industrial application were evaluated. In particular, internal memory elements and combinational logic have been considered as possible fault sources. Two alternative hardened versions of a system implementing a communication protocol have been evaluated by means of simulation-based fault injection.

The experiments showed that even if a simple parity-based hardening approach would provide complete coverage of the injected faults, this solution has limited capabilities for identifying fault effects. As a result, in case of time-consuming error recovery strategies, the system could waste time to recover from effectless faults reducing the system availability. Moreover, the parity-based hardening approach is not able to handle SETs in combinational logic.

On the other hand, the experiments showed that the SIFT approach, even if still not exhaustive as far as the error-detection capability is concerned, is able to efficiently classify fault effects. As a result, effectless faults are neglected, while faults potentially leading to wrong answers are detected and the appropriate error recovery procedure is invoked. The automated implementation allows reducing the overall implementation costs thanks to the adoption of cheaper and not-hardened parts and to the reduction of the system delivering time. Moreover, the considered SIFT approach showed a very high degree of flexibility, being able to effectively detect both SEUs and SETs.

In order to make the SIFT approach more widely applicable, we are currently investigating further improvements to reduce performance degradation and to add correction capabilities.

REFERENCES

[1] J. Gaisler, "Evaluation of a 32-bit microprocessor with built-in concurrent error detection," in *Proc. 27th Int. Symp. Fault-Tolerant Computing*, Seattle, WA, June 1997, FTCS-97, pp. 42–46.

[2] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proc. 17thVLSI Test Symp.*, Dana Point, CA, Apr. 1999, pp. 86–94.

[3] L. W. Massengill, "Cosmic and terrestrial single-event radiation effects in dynamic random access memories," *IEEE Trans. Nucl. Sci.*, vol. 43, pp. 576–593, Apr., pt. 1 1996.

[4] L. Anghel and M. Nicolaidis, "Cost reduction of a temporary faults detecting technique," in *Proc. Design, Automation, and Test Eur. Conf.*, Paris, France, Mar. 2000, pp. 591–598.

[5] P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 2231–2236, Dec. 2000.

[6] P. P. Shirvani, N. Saxena, and E. J. McCluskey, "Software-implemented EDAC protection against SEUs," *IEEE Trans. Reliability*, vol. 49, pp. 273–284, Sept. 2000.

[7] A. Benso, S. Chiusano, P. Prinetto, and L. Tagliaferri, "A C/C++ source-to-source compiler for dependable applications," in *Proc. Int. Conf. Dependable Systems and Networks*, New York, June 2000, pp. 71–78.

[8] K. H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. C-33, pp. 518–528, Dec. 1984.

[9] Z. Alkhalifa, V. S. S. Nair, N. Krishnamurthy, and J. A. Abraham, "Design and evaluation of system-level checks for on-line control flow error detection," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, pp. 627–641, June 1999.

[10] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 2405–2411, Dec. 2000.

[11] L. W. Massengill, A. E. Baranski, D. O. Van Nort, J. Meng, and B. L. Bhuva, "Analysis of single-event effects in combinational logic-simulation of the AM2901 bitslice processor," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 2609–2615, Dec. 2000.

[12] [Online]. Available: http://www-us.semiconductors.com/i2c/

[13] M. Rebaudengo, M. Sonza Reorda, M. Torchiano, and M. Violante, "An experimental evaluation of the effectiveness of automatic rule-based transformations for safety-critical applications," in *Proc. Int. Symp. Defect and Fault-Tolerance in VLSI Systems*, Yamanashi, Japan, Oct. 2000, pp. 257–265.

[14] H. Cha, E. M. Rudnick, J. Patel, R. K. Iyer, and G. S. Choi, "A gate-level simulation environment for alpha-particle-induced transient faults," *IEEE Trans. Comput.*, vol. 45, pp. 1248–1256, Nov. 1996.

[15] R. Velazco, P. Cheynet, A. Bofill, and R. Ecoffet, "THESIC: A testbed suitable for the qualification of integrated circuits devoted to operate in harsh environment," in *Proc. IEEE Eur. Test Workshop*, Barcelona, Spain, May 1998, pp. 89–90.

[16] R. Velazco, P. Cheynet, and R. Ecoffet, "Effects of radiation on digital architectures: One years results from a satellite experiment," in *Proc. XII Symp. Integrated Circuit and System Design*, Natal, Brazil, Sept. 1999, pp. 164–169.