Fault Injection on a Mixed-Signal Programmable SoC with Design Diversity Mitigation

Carlos J. G. Aguilera, Cristiano P. Chenet, Tiago R. Balen

Federal University of Rio Grande do Sul (UFRGS) - Graduate Program on Microelectronics (PGMICRO)
cjgaguilera@inf.ufrgs.br, cristiano.chenet@ufrgs.br, tiago.balen@ufrgs.br

ABSTRACT

This paper presents an approach for runtime software-based fault injection, applied to a commercial mixed-signal programmable system-on-chip (PSoC). The fault-injection scheme is based on a pseudo-random sequence generator and software interruption. A fault tolerant data acquisition system, based on a design diversity redundant scheme, is considered as case study. The fault injection is performed by intensively inserting bit flips in the peripherals control registers of the mixed-signal PSoC blocks, as well as in the SRAM memory of the device. Results allow to evaluate the applied fault tolerance technique, indicating that the system is able to tolerate most of the generated errors. Additionally, a high fault masking effect is observed, and different criticality levels are observed for faults injected into the SRAM memory and in the peripherals control registers..

Index Terms: Fault Injection, Soft-Error, Fault Tolerance, Triple Modular Redundancy, Design Diversity, Mixed-Signal, Single Events, Data Converters, Programmable System-on-Chip.

I. INTRODUCTION

Over the years, the dimensions of the transistors have been progressively reduced, allowing, so far, the continuity of Moore's Law. This paradigm, with the consequent prevalence of deep submicron technologies, allowed the increase in clock rates and circuit miniaturization. On the other hand, such miniaturization implies in higher current leakage, at the same time that make the integrated circuits more sensitive to ionizing radiation effects [1]. In a system that operates in a spacecraft, commercial aircraft or even at ground level, ionizing particle strikes may affect memory blocks, microprocessors, and mixed-signal blocks generating processing errors and potential system failure [2].

Analog-to-digital converters (ADCs) are common blocks in control, instrumentation and communication systems, including the ones adopted in space applications. While much effort has been directed to studies evaluating soft error effects and mitigation techniques in complex digital circuits, such as processors and FPGAs (Field Programmable Gate Arrays), few works deal with this problem on data converters or mixed signal devices [3]–[5].

Fault injection is an evaluation technique used to assess the dependability and fault tolerance degree of electronic and computer systems, by simulating or emulating faults that may occur during the system operation, both in software and hardware. Computers and their applications have increased reliance on electronic systems, in which downtime and failures are not tolerable, such as in safety critical and financial critical applications [6], [7].

A possible fault injection technique, known as software implemented fault injection (SWIFI) consists in emulating faults by modifying the system software in order to include fault injection capability [8]. For systems that are intended to operate in radiation environments, this technique is used for testing the protection and mitigation techniques that are commonly employed in such applications [9]. The use of software interruptions to inject faults in processor based systems was proposed in [10], [11], addressing a technique known as CEU (Code Emulated Upset).

In previous works of our research group [4], [12], we proposed the application of a mitigation strategy based on modular redundancy with design diversity to a data acquisition system, prototyped in a PSoC (Programmable System-on-Chip) device. In the mentioned works, a compiled-time fault injection was carried out, by manually inserting bit-flips in few registers of the architecture, since the purpose was only to validate the spatial-temporal voting scheme proposed in those works.

The first contribution of the current work is the description of a fully automated framework for massive fault injection in the studied PSoC system. The fault injection system is based on a pseudo-random number generator implemented in an auxiliary board, to select the memory and bit positions, to insert the faults, and software interruption to perform the bit-flip injection routine. As a second contribution, the obtained experimental results of fault injection help to understand the criticality levels of faults affecting different parts of the device and the functional implications of such faults in a mixed-signal diversity-based design.

In fact, a PSoC device was already tested under fault injection in a related work [10]. However, the experiment was directed to a device from the first generation of the PSoC family (comprising a simple 8-bit processor). The target application in that work was purely digital (matrix multiplication), to which no mitigation technique was applied. In this work, the studied device pertains to the third generation of PSoC family from Cypress semiconductor (comprising a 32 bit ARM processor). Additionally, the application is a fault tolerant mixed-signal system based on design diversity, comprising three ADCs, besides digital hardware and software resources for controlling the converters and the direct memory access, as well as to perform the voting. Therefore, experimental results also allows to validate the previously proposed mitigation technique [12].

II. BACKGROUND

A. Radiation Effects and Soft Errors

Radiation effects on electronic systems may be classified as Total Ionizing Dose (TID), Displacement Damage (DD) and Single Event Effects (SEE) [13]. TID is a long term cumulative effect, which degrades some electrical properties of circuits due to the build up of trapped charges in the integrated circuits oxides [14]. Displacement damage are defects created into the crystalline structure of the semiconductor due to nonionizing energy loss of incident particles (usually heavy ions and neutrons) [15]. Single event effects occur due to the impact of strongly ionizing particles in sensitive areas of integrated circuits, inducing current pulses that can disturb the circuit operation [16]–[20]. If the SEE generates a bit inversion in a memory element it is called SEU (Single Event Upset) [21]. On the other hand, a temporary current pulse induced by an SEE that may propagate into the signal path (either in digital or analog circuits) is known as Single Event Transient (SET) [22]. The system level effects of bit inversions in digital systems, caused by SEEs, are also known as soft errors. If the system stop working due to an SEE the event is classified as Single Event Functional Interrupt (SEFI).

B. Fault Tolerance with Design Diversity

Safety critical systems exposed to ionizing radiation must employ some hardening strategy, depending on the application and the required degree of radiation tolerance. There are several mitigation techniques that are carried out since the early stages of a system or IC (Integrated Circuit) development. Mitigation to soft errors is usually obtained by adding some degree of redundancy, usually hardware or information redundancy (as, for example, error correcting codes). Triple Modular Redundancy (TMR) is a popular technique that consists in triplicating the hardware (or part of it) and voting upon the results of the computation done by each TMR copy. This way, a reduction in soft error rate is obtained, for systems operating under ionizing radiation incidence [23]. The drawbacks of this technique are the increase in area and power consumption.

An improved TMR technique is the Diversity Triple Modular Redundancy (DTMR). In this approach the hardware and software elements used to perform the multiple computations are not copies, but are independently designed to meet the system requirements [24]. Design diversity can enhance the system reliability to common-mode faults because each module may have different levels of resilience, therefore the probability of multiple domain faults may be reduced. The main objective of this technique is to avoid multiple errors that may arise due to commonalities among the system copies, by using different hardware devices, different clock frequencies, and different software implementations [12].

Redundancy with diversity is used by designers and integrators of electronic systems for critical applications, such as space missions, avionics and military applications. Examples of applications of Design Diversity techniques in aircrafts from NASA, Airbus and Boeing can be found in [25]–[28]. In some recent works, assessment of fault tolerance is performed for various diversitary architectures, specifically in digital and FPGA-based architectures [29], [30].

In [4], the application of DTMR to mixed-signal (MS) circuits was addressed, identifying the possible modes of diversity implementation (time, domain, level and architecture) and the drawbacks of applying this technique do MS systems. One of the case studies of the mentioned work is a data acquisition system implementing the DTMR technique with hardware and time diversity. In that work the

system was validated by a limited set of manually injected faults. For this reason, the same case study circuit is considered in this work, to be target of massive fault injections as described in the following sections.

III. FAULT INJECTION SETUP

A. Programmable SoC

The case study circuit, a Data Acquisition System (DAS), was fully implemented in a commercial programmable SoC (PSoC 5LP from Cypress Semiconductor) manufactured in a 130nm CMOS technology. The general architecture of the PSoC is presented in Figure 1 [31]. The PSoC has a 32-bit ARM Cortex-M3 CPU (up to 80 Mhz), 256 kBytes of flash memory, 64 kBytes of SRAM memory, 2 kBytes of EEPROM memory and 24 channels of DMA (Direct Memory Access). The device also comprises digital peripherals such as communication interfaces and PLDs (Programmable Logic Devices), based on UDBs (Universal Digital Blocks) which provide the implementation of various functions such as timers, counters, and others. Also, analog peripherals such as a sigma-delta AD converter, two SAR (Successive Approximation Register) converters, digital to analog converters, comparators, operational amplifiers and configurable analog blocks may also be used to implement several analog functions [31].

B. Case Study Implementation

The simplified block diagram of the case study circuit is depicted in Figure 2. The DAS is composed by three ADCs operating in parallel: two SAR converters and a sigma-delta converter. Besides the hardware diversity implementation, due to different ADC architectures, temporal diversity is achieved due to the different sampling rate of the SAR ADCs (740 ksps and 74 ksps, where sps stands for "samples per second"). The system also comprises two voters: one main spatial voter and a temporal voter, which also do the coarse synchronization of the DAS.

In addition to the ADCs and voters the implemented system also comprises three sample-and-hold blocks, three channels of Direct Memory Access (DMA) and a synchronizer block, needed to accurately synchronize the voting cycles, since the conversion times are different for the triplicated converters. Besides that, a fault injection block is also necessary. Additionally, a status register (composed by 5 circular buffers) monitors the output of the three ADCs and both voters, sending its content to an external computer whenever a fault is detected by the voters. The buffer size is such that at least two complete periods of the converted analog signal is stored (one cycle before and another after the error detection). The fault injection block also communicates to an auxiliary equipment (AE), responsible for controlling the fault injection procedure. Figure 3 shows the overall system block diagram of the DUT (Device Under Test).

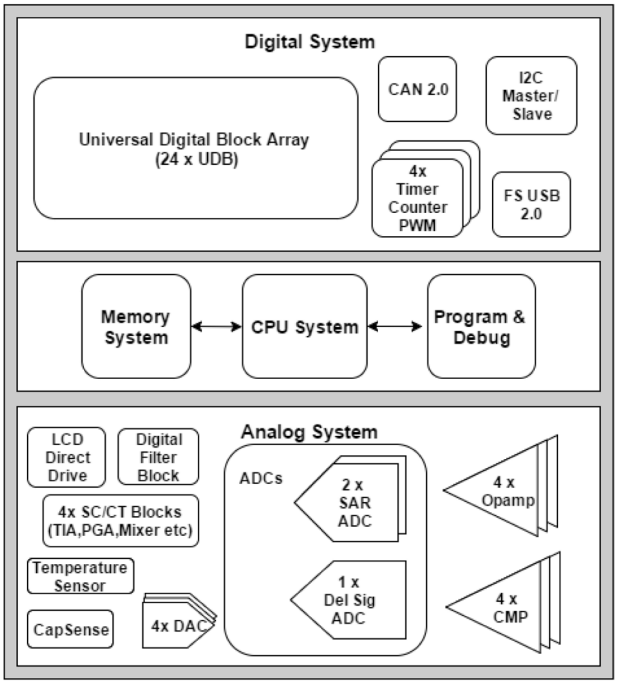

Figure 1. Architecture of PSoC 5LP [31].

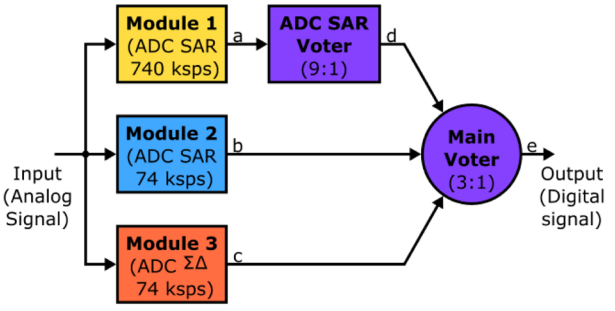

Figure 2. DAS scheme, based on a Diversity TMR technique

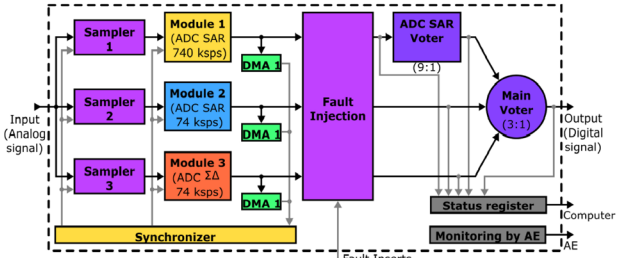

Figure 3. Details of the full implementation of DAS in the PSoC device.

The AE is implemented in secondary board, comprising another PSoC device, in which a pseudo-random sequence generator (PRSG) is programed. The PRSG defines the address and the bit position that will be flipped, both in the peripherals control registers and in the SRAM memory, during the fault injection. A predefined clock of the AE activates the PRSG. The generated random number is sent to the DUT through a parallel connection, then, the software interruption is activated in the DUT to inject the fault. The interruption routine is responsible for inserting the fault, by performing an XOR operation with the content of the selected register and a mask corresponding to the faulty bit position. The implemented fault injection scheme, is shown in Figure 4. The AE also operates like a watchdog in which a counter is reset whenever a transition occurs in an "alive" signal sent by the DUT. If the alive signal remains silent for more than 30 seconds, the watchdog resets the DUT. This is done to deal with possible SEFIs (Single Event Functional Interrupts) on the DUT

Due to the memory organization of the PSoC and due to the fact that part of the SRAM is devoted to the control registers of the peripherals, the fault injection is performed in distinct ways, for the peripherals and for the CPU SRAM memory.

1) *Fault Injection in Peripherals*: The registers of the PSoC responsible for controlling the peripherals are composed of 8 bits. The nominal addresses of the peripherals registers range from 0x40004000 to 0x5FFFFFFF. However, there are only 13000 physical registers within this address range [31]. Therefore, a fully random fault injection is not allowed, since most of the address in this range are not targeted to a physical memory position. To overcome this issue, a vector with the valid addresses of the peripherals registers is stored into the fault injection block into the DUT, in a way that the PRSG will randomly select one of the valid addresses to inject a fault. Due to limitations in memory to store all 13000 addresses, a subset of 7.892 registers was selected, prioritizing the ones responsible for controlling the analog and mixed-signal peripherals. A clock signal provided by the AE, activates the system interruption in order to inject the fault into the DUT. The clock frequency defines the number of injected faults per second. In this work a 1 Hz clock frequency is employed.

In this part of the experiment, a "semi-permanent" model for fault injection into the peripherals is used. This means that the fault is injected and after 500 ms it is removed. This time, while the bit-error remains active, comprises several millions of clock cycles. This is done since the fault may remain latent and an error may manifest several clock cycles after the fault injection, during the program execution. Faults are then removed to prevent the error pile up, in order to have a more precise interpretation of the results.

2) *Fault Injection in SRAM*: The PSoC SRAM memory consists of two banks of 32 kB, as shown in Table I. From the base addresses 0x1FFF8000 and 0x20000000 it is possible to reach any address of the banks by adding to them a 15- bit number. This way, such 15-bit number is generated by the pseudo random sequence generator from AE, so that a fully random address selection is achieved for the SRAM fault injection. A permanent fault model for injecting faults in the SRAM is adopted, which means that a fault inserted is never removed by the injector block.

C. Test Setup

The overall test setup for the fault injection campaign is shown in Figure 5. Besides the AE, the DUT is connected to a computer which stores the fault detection logs. The DAS input is a 120Hz sinewave signal from an external signal generator. An oscilloscope is also used to monitor the real-time operating status of the DAS, by means of the "alive" signal.

During the experiment, 23815 faults were injected into the peripheral control registers. This is near 3 times the number of registers stored in the valid addresses vector. Into the SRAM, 130236 faults were in-

Table I. PSoC SRAM organization

Address range	Purpose
0x1FFF 8000 - 0x1FFF FFFF	Up to 32 kB SRAM in code region
0x2000 0000 - 0x2000 7FFF	Up to 32 kB SRAM in SRAM region

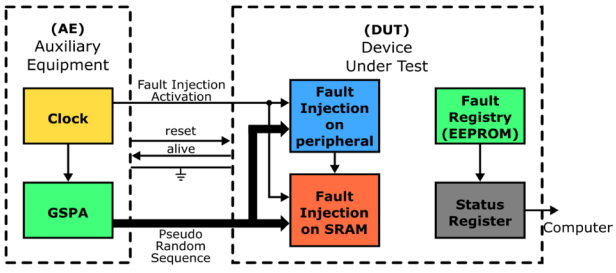

Figure 4. Main blocks of the AE and DUT needed for the fault injection procedure.

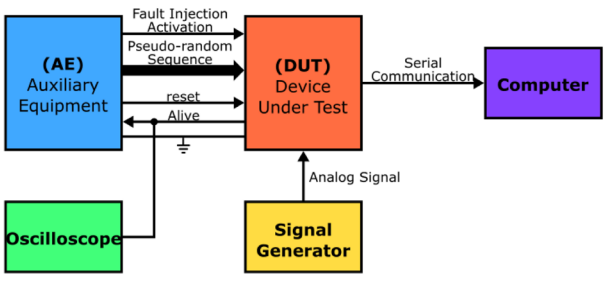

Figure 5. Test setup for the fault injection campaign.

jected, which corresponds to near two injected faults per SRAM byte (64 kBytes). The overall usage rate of the SRAM memory, according to the compiler report for the DUT, is 36.8%.

IV. EXPERIMENTAL RESULTS

The experimental results are divided in two groups, in order to better detail the errors observed either due to injected faults in the peripherals and in the SRAM.

A. Test Setup

Errors detected in the peripherals were classified into two groups: i) errors detected by the system voters, and ii) errors due to SEFIs. The later class comprises the errors that were detected by the watchdog, or by the experiment operator (manual reset). Table II shows the results of fault injection in the peripherals. From the 23815 injected faults, 848 generated detected errors, corresponding to 3.56% of the total injected faults. From the total amount of errors 2.22% (528) were detected by the voters.

In the group of observed errors due to SEFIs, 4 different types of errors were identified. The communication between the DUT and the monitoring computer is made through a serial RS232 interface. Therefore, one class of observed SEFI is related to an error on the serial interface, manifested by incoherent writing in the test report, possibly generated by a change in the interface speed, which may be due to faults injected into the system PLLs and oscillators. However, the system was able to continue running, though the test data results were not generated. For this type of error, a manual reset had to be triggered to continue the experiment. The same procedure had to be applied to the second class of SEFI error: lost of functionality of fault injector blocks of the DUT. This is probably due to a failure on the interruption scheme of the microprocessor. However, as in the previous mentioned case, the DAS system remained working correctly. The third class of observed SEFIs, for the peripherals fault injection, is a permanent error at the output of one of the converters. These cases also demanded a manual reset. These 3 classes of errors, comprise 63 of the 320 observed SEFIs, and are not the most critical ones, since the system is still functional, even when one of the converter fails (due to the DTMR scheme). Completing the set of SEFI faults there are the faults that caused a system halt and, for this reason, they were detected by the AE watchdog. This way, the total number of critical SEFIs observed is 257 (1.08% of injected faults).

Table II. Results of fault injection in the peripherals

Total injected faults	**23815**	**100%**
Total observed errors	**848**	**3,56%**
-Total errors detected by voters	528	2,22%
-Total SEFIs	320	1,34%

Figure 6 shows examples of errors detected by the voters on the ADCs, resulting from fault injection in the peripherals. Since single fault injection is performed at each injection, in all cases the overall DAS was able to tolerate the fault, as expected for a TMR system. The discontinuities observed in the curves are related to the time needed by the fault injection routine, since, during the fault injection, some analog samples are lost, and there is a time discontinuity as well, though not evidenced in the plots. Figures 6 (a) and (b) show an error on the SAR ADC operating at 740 kps which is recovered on the next voting cycle. A similar case is observed in figure 6 (d) on the SAR ADC operating at 74 ksps. Figures 6 (c), (e) and (f) show some cases in which a permanent error is observed at the converters.

B. Results of fault injection in SRAM

Only 46 errors were detected from the 130236 faults injected in the SRAM memory, corresponding to 0.03% of faults detected as errors, as can be seen in Table III. It can be noticed from the table that 60.87% of the observed errors, were detected by the voters, and 39.12% generated SEFIs. The critical SEFIs (system halt) represents 13.04% of the observed errors. This shows a high fault masking effect in this memory. Figure 7 shows some examples of the ADCs and voter outputs for faults that were detected by the voters.

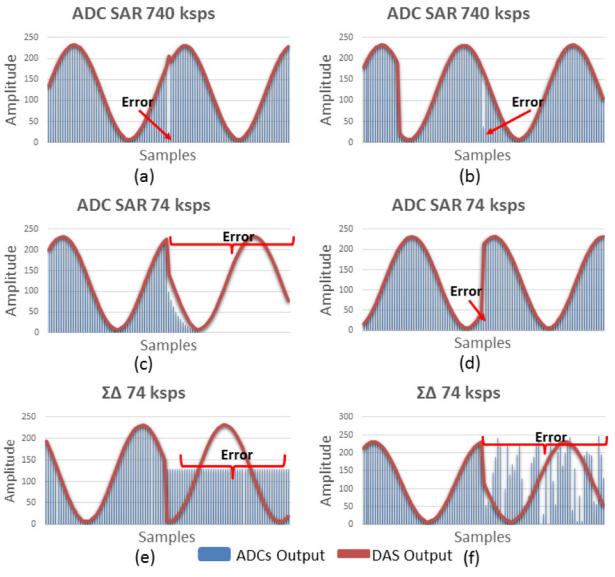

Figure 6. Examples of soft error effects on the ADCs, for faults injected into the peripherals.

Table III. Results of fault injection in the SRAM

Total injected faults	**130236**	**100%**
Total observed errors	**46**	**0,03%**
-Total errors detected by voters	28	60,87%
-Total SEFIs	18	39,12%

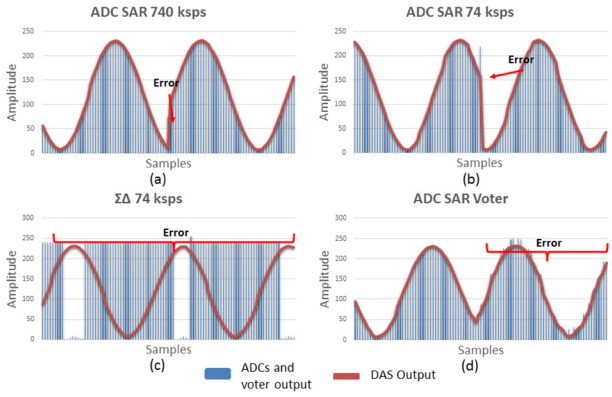

Figure 7. Examples of soft error effects on the ADCs and voters, for faults injected into the SRAM memory.

Figures 7 (a) and (b) show errors observed at the SAR ADCs, immediately after the injection of the fault. It is possible that, in these cases, the fault was injected in the address in which the quantized sample was stored in the SRAM, directly modifying its value.

Figure 7 (c) shows errors in the Sigma-Delta ADC. It should be noticed that, in this case, errors were detected after the time of the fault injection, since no discontinuity of the corrected voted signal is observed in the curve. One of the possible causes of this error can be attributed to the inadequate performance of either the buffer, which stores the converted data, the voters or the status register block. Figure 7 (d) shows errors occurring at the output of the temporal voter (SAR ADC voter), which start to occur after the injection of the fault and remain on subsequent voting cycles. This behavior suggests that a modification in the voter may have occurred generating a small error at its output.

It can be noticed from the fault injection experiments, that, for the tested application, the faults at the memory addresses related to the control of the peripheral of the PSoC are more critical than faults occurring in the CPU SRAM. In this implemented system, the application is highly dependent on the peripherals, to perform the sampling and conversions of the analog signal. Although 36.8% of the SRAM is used, as the voting task and the general control of the system are performed by software, faults in SRAM have a lower impact in the performed application.

V. CONCLUSIONS

In this paper a software-based fault injection campaign was applied to a commercial programmable mixed-signal system-on-chip. Faults are emulated by means of SW interruption, which modifies the memory content, inserting bit-flips in memory positions randomly selected by a pseudo-random number generator. More than 154000 faults were injected in a data acquisition system programmed into the PSoC. The case study design implements a DTMR fault tolerance technique with spatial and temporal voting schemes.

Fault injection was divided in two experiments, one to inject fault in the PSoC peripherals and other directed to the CPU SRAM. Results showed that the faults injected at the peripheral blocks present higher criticality to the application than the faults injected into the SRAM, since only 0.03% of faults in SRAM generated errors, whereas for the peripherals this number is 3.56%.

Results show that most of the observed errors were tolerated by the DTMR system (62% of total errors). Additionally, most of the observed SEFIs are related to malfunction on the fault injection and monitoring blocks, whereas only 0.19% of the total injected errors generated critical SEFIs

As general conclusions, the applied fault injection methodology showed suitable to intensively test the soft error effects on the complex mixed-signal SoC under study. Finally, with the fault injection results, the previously proposed mixed-signal DTMR technique was proved to be effective to tolerate soft errors.

ACKNOWLEDGMENT

This research is supported in part by Brazilian National Council for Scientific and Technological Development (CNPq), under grant number 56947/2014-0.

REFERENCES

[1] H. J. Barnaby, “Total-ionizing-dose effects in modern cmos technologies,” IEEE Transactions on Nuclear Science, December 2006.

[2] A. Johnston, C. Lee, B. Rax, and D. Shaw, “Using commercial semiconductor technologies in space,” pp. 175–182, 1995, radiation and its Effects on Components and Systems - RADECS.

[3] A. J. Lanot and T. R. Balen, “Analysis of the effects of single event transients on an sar-adc based on charge redistribution,” 2014, 15th IEEE Latin-American Test Workshop.

[4] C. P. Chenet, T. R. Balen, F. L. Kastensmidt, L. A. Tambara, G. M. Borges, and M. S. Lubaszewski, “Exploring design diversity redundancy to improve resilience in mixed-signal systems,” 2015, microelectronics Reliability.

[5] T. L. Turflinger, “Single-event effects in analog and mixed-signal integrated circuits,” IEEE Transactions on Nuclear Science, pp. 594– 602, April 1996.

[6] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, “Fault injection techniques and tools,” Computer, pp. 75–82, April 1997.

[7] J. A. Clark and P. K. Dhiraj, “Fault injection a method for validating computer-system dependability,” Computer, pp. 47–56, June 1995.

[8] H. Ziade, R. Ayoubi, and R. Velazco, “A survey on fault injection techniques,” The International Arab Journal of Information Technology. Vol 1, No. 2, pp. 171–186, July 2004.

[9] J. V. Carreira, D. Costa, and J. G. Silva, “Fault injection spot-checks computer system dependability,” IEEE Spectrum, pp. 50–55, August 1999.

[10] W. Mansour, R. Velazco, H. Ziade, R. Ayoubi, and W. E. Falou, “Seu simulation by fault injection in psoc device: Preliminary results,” 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), pp. 330–333, 2012.

[11] R. Velazco, S. Rezgui, and R. Ecoffet, “Predicting error rate for microprocessor-based digital architectures through c.e.u. (code emulating upsets) injection,” IEEE Transactions on Nuclear Science (Vol.47, no 6), pp. 2405 – 2411, December 2000.

[12] C. P. Chenet, A. J. Lanot, and T. R. Balen, “Design diversity redundancy with spatial - temporal voting applied to data acquisition systems,” IEEE, March 2014, IATW, 2014 15th Latin American.

[13] R. Velazco, P. Fouillat, and R. Reis, Radiation Effects on Embedded Systems. Springer Dordrecht, The Netherlands, 2007.

[14] M. Turowski, A. Raman, and R. Schrimpf, “Nouniform total-doseinduced charge distribuition in shallow-trench isolation oxides,” IEEE Transactions on Nuclear Science, vol. 51, no 6, pp. 3166–3171, 2004.

[15] J. Srour, C. Marshall, and P. Marshall, “Review of displacement damage effects in silicon devices,” IEEE Transactions on Nuclear Science, vol. 50, no 3, pp. 653–670, 2003.

[16] D. Binder, E. Smith, and A. B. Holman, “Satellite anomalics from galactic cosmic rays,” IEEE Transactions on Nuclear Science, v.22, n.6, pp. 2675–2680, December 1975.

[17] T. May and M. Woods, “A new physical mechanism for soft errors in dynamic memories,” Reliability Physics Symposium, pp. 33–40, 1978.

[18] J. Ziegler and W. Lanford, “Effect of cosmic rays on computer memories,” Science, v.206, n.4420, pp. 776–788, November 1979.

[19] F. Wang and V. Agrawal, “Single event upset: an embedded tutorial,” IEEE International Conference on VLSI Design, pp. 429–434, 2008.

[20] R. Ecoffet, S. Duzellier, P. Tastet, C. Aicardi, and M. Labrunee, “Observation of heavy ion induced transients in linear circuits,” Radiation Effects Data Workshop, IEEE, pp. 72–77, July 1994.

[21] C. S. Gunzer, E. A. Wolicki, and R. G. Allas, “Single event upset of dynamic rams by neutrons and protons,” IEEE Transactions on Nuclear Science, v.26, n. 6, pp. 5048–5052, December 1979.

[22] M. P. Baze and S. P. Bucbner, “Attenuation of single event induced pulses in cmos combinational logic,” IEEE Transactions on Nuclear Science, v.44, n. 6, pp. 2217–2223, December 1997.

[23] F. L. Kastensmidt, F. Almeida, S. Pagliarini, L. Entrena, A. Lindoso, E. S. Millan, and E. Chielle, “Single event induced charge sharing effects in tmr with differente levels of granularity,” 2012, iV Werice Aeroespacial - Workshop on the Radiation Effects on Electronic and Photonic Devices for Aerospace Appplications, Vol 1, p.67-72, out/2012, Sao Jose dos Campos. IEAv.

[24] A. Avizienis and J. Kelly, “Fault tolerance by design diversity: Concepts and experiments,” Computer, vol. 17, no. 8, pp. 67–80, 1984.

[25] J. Lala and R. Harper, “Architectural principles for safety-critical realtime applications,” Proc. IEEE 82 (1), p. 25?40, January 1994.

[26] K. Szalai and et al., “Digital fly-by-wire flight control validation experience,” NASA Tech. Memo. 72860, December 1978.

[27] R. Riter, “Modeling and testing a critical fault-tolerant multi-process system,” FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on Fault- Tolerant Computing, June 1995.

[28] D. Briere and P. Traverse, “Airbus a320/a330/a340 electrical flight controls a family of fault-tolerant systems,” FTCS-23. Digest of Papers., Twenty-Fifth International Symposium on Fault- Tolerant Computing, p. 616?623, June 1993.

[29] Z. Wang, L. Yang, and A. Chattopadhyayy, “Architectural reliability estimation using design diversity,” 2015, 16th International Symposium on Quality Electronic Design.

[30] R. J. Rizwan A. Ashraf, Ouns Mouri and R. F. DeMara, “Design-fordiversity for improved fault-tolerance of tmr systems on fpgas,” 2011, international Conference on Reconfigurable Computing and FPGAs.

[31] C. Semiconductor, PSoC 5LP Architecture TRM, 2015, technical Reference Manual http://www.cypress.com/?docID=46050.