Reliability Analysis of an On-Chip Watchdog for Embedded Systems Exposed to Radiation and EMI

C. Oliveira[1], J. Benfica[1], L. M. Bolzani Poehls[1], F. Vargas[1], J. Lipovetzky[2], A. Lutenberg[2], E. Gatti[3], F. Hernandez[4], A. Boyer[5]

[1] Catholic University of Rio Grande do Sul -PUCRS, Porto Alegre, Brazil, vargas@pucrs.br
[2] Universidad de Buenos Aires, Buenos Aires, Argentina, joselipo@gmail.com
[3] Instituto Nacional de Tecnologia Industrial - INTI, Buenos Aires, Argentina, egatti@inti.gob.ar
[4] Universidad ORT, Montevideo, Uruguay, ferhernasan@gmail.com
[5] LAAS-CNRS / Université de Toulouse, Toulouse, France, alexandre.boyer@insa-toulouse.fr

Abstract — **Due to stringent constraints such as battery-powered, high-speed, low-voltage power supply and noise-exposed operation, safety-critical real-time embedded systems are often subject to transient faults originated from a large spectrum of noisy sources; among them, conducted and radiated Electromagnetic Interference (EMI). As the major consequence, the system's reliability degrades. In this paper, we present the most recent results involving the reliability analysis of a hardware-based intellectual property (IP) core, namely Real-Time Operating System - Guardian (RTOS-G). This is an on-chip watchdog that monitors the RTOS' activity in order to detect faults that corrupt tasks' execution flow in embedded systems running preemptive RTOS. Experimental results based on the Plasma processor IP core running different test programs that exploit several RTOS resources have been developed. During test execution, the proposed system was aged by means of total ionizing dose (TID) radiation and then, exposed to radiated EMI according to the international standard IEC 62.132-2 (TEM Cell Test Method). The obtained results demonstrate the proposed approach provides higher fault coverage and reduced fault latency when compared to the native (software) fault detection mechanisms embedded in the kernel of the RTOS.**

Keywords- On-Chip Watchdog, Embedded System, Total Ionizing Dose (TID) Radiation, Electromagnetic Interference (EMI).

I. INTRODUCTION

Nowadays, several safety-critical embedded systems support real-time applications, which have to respect stringent timing constraints. In general terms, real-time systems have to provide not only logically correct results, but temporally correct results as well [1]. In this scenario, Real-Time Operating Systems (RTOSs) have been extensively adopted in order to minimize the design complexity of real-time embedded systems. Typically, these (embedded) systems exploit some important facilities associated to RTOSs' native intrinsic mechanisms to manage tasks, concurrency, memory as well as interrupts. In other words, RTOSs serve as an interface between software and hardware.

At the same time, the environment's always increasing hostility caused substantially by the ubiquitous adoption of wireless technologies represents a huge challenge for the reliability of real-time embedded systems [2]. Note that if these systems are powered by battery or this power is provided by energy harvesting techniques, the yielded reliability is even more fragile. In detail, external conditions, such as Electromagnetic Interference (EMI), Heavy-Ion Radiation (HIR) as well as Power Supply Disturbances (PSD) may cause transient faults on electronic systems [3,4]. Currently, the consequences of transient faults represent a well-known concern in microelectronic systems. The International Technology Roadmap for Semiconductor (ITRS) predicts increasing system failure rates due to this type of fault for future generation of integrated circuits [5]. Therefore, embedded systems based on RTOS are subject to Single Event Upsets (SEUs) causing transient faults, which can affect the application running on embedded systems as well as the RTOS executing the application [6,7]. Affecting the RTOS, this kind of fault can generate scheduling dysfunctions that could lead to incorrect system behavior [1].

Up to now, several solutions have been proposed to deal with the reliability problems of real-time systems [8,9,10]. However, the vast majority of such solutions provide fault tolerance only for the application level and do not consider faults affecting the RTOS that propagate to the application tasks. Typically, these techniques are focused on detecting errors (on the application level) that corrupt data manipulated by the processor and/or induce application illegal control-flow execution [10]. Some of these approaches [8,9] deal with detecting faults degrading RTOS performance, but such approaches are very limited and cover only a few subset of possible faults affecting RTOS activity.

Regarding faults affecting the RTOS that propagate to application tasks, previous works indicate that 21% of them lead to application failure [1] and then, are liable to be detected by approaches monitoring application-level execution. Generally, these faults tend to miss their deadlines and to produce incorrect output results. Moreover, the work presented in [6] demonstrates that about 34% of the faults injected in the processor's registers led to scheduling dysfunctions. Indeed, about 44% of these dysfunctions led to system crashes, about 34% caused real-time problems and the remaining 22% generated incorrect system output results. To conclude, the fault tolerance techniques proposed up to now represent feasible solutions, but they do not guarantee detection of faults affecting the RTOS task scheduling process.

In this context, authors proposed in [11] an on-chip watchdog (RTOS-G) to monitor the RTOS's execution flow in order to detect scheduling misbehavior. RTOS-G provides detection of faults that change the tasks' execution flow in embedded systems for critical applications. In the first version, the on-chip watchdog was prototyped in systems running RTOSs based on the *Round-Robin* scheduling algorithm. As a further development, authors published [12] a more complex version of the watchdog devoted to monitor the scheduling activity of *Preemptive* RTOSs.

In the present work, we analyze the fault detection capability of such on-chip watchdog. This reliability analysis is performed by means of a case-study implementation based on the soft-core processor Plasma (opencores.org) and the RTOS-G watchdog that were prototyped into the Xilinx Virtex4 FPGA. This IC was aged under exposition to total ionizing dose (TID) radiation and the system performance analyzed under TEM-cell test method conditions [13].

The remainder of the paper is organized as follows: Section II presents a background with respect to the theory of RTOSs. Section III briefly describes the on-chip watchdog to monitor preemptive RTOSs. Section IV presents the case-study, describes the TID radiation and EMI experiments setup and discusses the obtained results. Finally, Section V draws the final conclusions of this work.

II. BACKGROUND

RTOSs represent a vital component to many embedded systems and provide a software platform upon which to build applications. An RTOS is a program that schedules execution in a timely manner, manages system resources, and provides a consistent foundation for developing application code. Basically, RTOSs can be classified in hard-RTOSs and soft-RTOSs. The main difference between the two categories is that a soft-RTOS can tolerate latencies and responds with decreased service quality while the hard-RTOS has to respect its deadlines, otherwise its tasks fail execution. In general terms, RTOSs provide four basic services to the application service: (1) time management, (2) interrupt handling, (3) memory management and (4) device management.

In computing, a *process* is an instance of a computer program that is being executed. A *computer program* is a passive collection of instructions; a *process* is the actual execution of those instructions. Several processes may be associated with the same program; for example, opening up several instances of the same program often means more than one process is being executed. Processes are often called "tasks" in embedded operating systems. The sense of "process" (or task) is "something that takes up time", as opposed to 'memory', which is "something that takes up space". In this context, a task is a set of program instructions that are loaded in memory.

A *deadline* is the time instant at which a task must finish its execution. The period of a task is the time interval between initiating two successive executions of such task. Generally, a task can be in one of the following three states: *blocked*, *ready* or *running*. Further, the transfer of CPU execution from one task to another one is called Context Switch (CS).

Every RTOS has a wide range of facilities (namely, system resources), which simplify the design of real-time applications by offering native mechanisms to manage tasks, concurrency, memory, time as well as interrupts. In comparison to other (not real-time) operating systems, the efficient use of the CPU is considered the more critical and the more important issue in real-time RTOSs. For instance, upon accessing a given blocked embedded system resource during the execution of a task, real-time RTOSs might force the task to wait for a semaphore release or some other external event before proceeding executing such task. In this context, RTOSs perform a CS to force the CPU to execute another task that is labeled *ready* to run and therefore, guarantee a more efficient usage of the CPU. If there is more than one task ready to run, the decision will be taken on the basis of task priorities.

Most RTOSs use scheduling algorithms based on the *Round-Robin* algorithm, which assigns equal Time Slices (TSs) to each task and executes them without priority in circular order. However, in a typical real-time application, there will be tasks with a shorter response time than others. Considering this situation, RTOSs usually implement a *Preemptive* algorithm with priority support. This results in a dynamic scheduling order and ensures time consistency for critical tasks.

It is important to point out that preemption will take place only if a task with higher priority than the executing one is ready to run. However, if all ready tasks have the same priority, the *Tick* signal will divide the CPU time between these tasks in equal parts (TSs) and no preemption can take place.

III. THE PROPOSED ON-CHIP WATCHDOG

This section describes the IP core RTOS-G. This IP is an on-chip watchdog that monitors tasks' execution flow according to the RTOS *preemptive* algorithm [11,12]. Figure 1 depicts the RTOS-G functional block diagram.

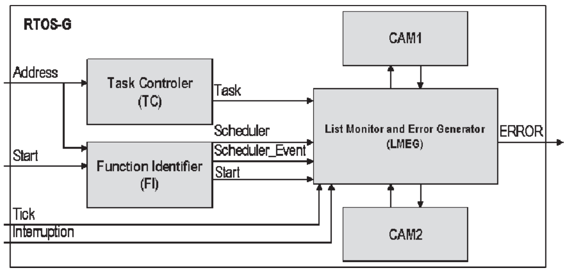

Figure 1. Functional block diagram of the RTOS-G.

The RTOS-G is connected to the embedded system's bus in order to monitor the following information: *Start*, *Tick* and

Interrupt signals as well as the RAM addresses accessed during the execution of the application code. In more detail, the RTOS-G is composed of five functional blocks. The *Task Controller* (TC) identifies the task in execution based on the address accessed by the microcontroller during the application's execution. At every clock cycle, the TC compares the address on the bus with the addresses associated to each task. If the accessed address is related to a task, the signal *Task* receives the corresponding task's number. The *Function Identifier* (FI) analyses the functions executed during the task scheduling process in order to check the scheduling process execution order. Finally, the FI identifies the event that triggered the scheduling process based on that order (e.g., occurrence of a *Tick* signal, IO request or semaphore acquisition). The block named *List Monitor and Error Generator* (LMEG) receives the *Scheduler_Event* signal and the *Task* in execution. Based on this information the LMEG classifies all tasks in two separate lists, *ready tasks* and *blocked tasks*, each one organized according to their state and priority. The LMEG implements the scheduling algorithm and indicates errors when a scheduling misbehavior is detected. As the last blocks, the two *Content-Addressable Memories* (CAM1 and CAM2) save the lists generated by the LMEG module. The tasks labeled *ready* are stored in CAM1 while the tasks labeled as *blocked* are saved in CAM2.

To implement preemption, the algorithm with priority support keeps a list of all tasks labeled *ready* (ready-list). The tasks are sorted by their priority. Therefore, every time a *CS* takes place and a scheduling event is performed, the (*ready*) task marked with the highest priority is executed. The complexity of monitoring this kind of behavior relies on keeping track of the *ready*-list: its elements must not have any pending IO requests or semaphore objects still to be acquired. In order to acomplish this task (keeping track of the *ready*-list), the RTOS-G should monitor not only the task addresses, but also the addresses related to the kernel synchronization, including: *SemaphoreLock()* and *SemaphoreUnlock()*. These functions lock and unlock a previously created synchronization object which is passed by parameter to the related functions. However, it is not possible for the RTOS-G to monitor the parameters of function calls; only the addresses of the functions are captured by the RTOS-G. Consequently, the described solution does not monitor all possible fault conditions. To counteract this limitation, an execution flow analysis is adopted as solution, since the function parameters remain unknown. In this solution, the RTOS-G observes the *order* in which the functions are being called to infer the *ready*-list constraints. To illustrate this mechanism, Figure 2 shows a situation where *Task1* is running and tries to acquire a semaphore. The system call is performed and the RTOS kernel realizes that the semaphore is already locked. In order to prevent the system from going into a deadlock as well as to increase the CPU usage, the kernel performs a *CS* calling another task into execution. The resulting execution flow for an already locked semaphore consists of: *SemaphoreLock()* and *ReSchedule()*. When the RTOS-G detects this flow, it will infer that *Task2* is *running* and therefore is taken out from the *ready*-list. A similar analysis can be performed for other situations, always concentrating all efforts in keeping the detection algorithm generic enough for any RTOS or processor. As further positive effect, this type of analysis has rendered dispensable the *Tick* signal. In more detail, the RTOS-G detects the *Tick* by recognizing the following execution flow: *Interrupt()* and *ReSchedule()*.

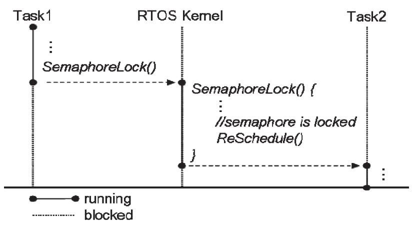

Figure 2. Typical task context switching activity executed under the control of an RTOS, which is monitored by the on-chip watchdog, RTOS-G.

IV. Experimental Results

The fault detection capability of the RTOS-G watchdog with respect to the RTOS native fault detection mechanisms has been evaluated by: (1) aging FPGAs containing the RTOS-G and a microprocessor. Aging was performed by exposing the FPGAs to total ionizing dose (TID) radiation; and (2) in the sequence, the FPGAs were exposed to radiated EMI according to the IEC 62.132-2 standard (TEM Cell Method) [13].

A. Case Study

To evaluate the proposed approach we adopted a case study composed of a Von Neumann 32-bit RISC Plasma microprocessor running an RTOS (www.opencores.org). The Plasma microprocessor is implemented in VHDL and has, with exception of the load/store instructions, an instruction set compatible to the MIPS architecture. Moreover, the Plasma's RTOS adopts the *preemptive scheduling* algorithm with priority support composed of the following three states: *blocked*, *ready* and *running*. The Plasma's RTOS provides a basic mechanism able to monitor the task's execution flow and manage some particular situations when faults cause misbehavior of the RTOS's essential services, such as stack overflow and timing violations. This mechanism is implemented by a function named *assert()*. Generally, when the argument of the *assert()* function is false, the RTOS sends an error message through the standard output.

For the fault injection experiments, we developed two different benchmarks that exploit great part of the resources offered by the Plasma's RTOS (i.e., the use of message queues, semaphores and interrupts). Figure 3 shows the block diagram associated to the two benchmarks implementing the following tasks:

- *BM1:* 8 tasks access and update the value of a global variable, which is protected by a semaphore. Indeed, another global variable is accessed by an interrupt. The 8 tasks are assigned to the following different priorities: 1, 2, 3, 4, 1, 2, 3 and 4, respectively. The interrupt has the maximum priority.

- *BM5:* this benchmark is the most complex of the two experiments and consumes the largest amount of RTOS resources. In this benchmark, an *interrupt* communicates with a task (T1) via a message queue. Then, T1 communicates with Tasks T2 and T3 via two other message queues, which in turn send messages to Tasks T4, T5, T6, T7 T8, T9 and T10, respectively, by means of queue resources as well, as depicted in Figure 3. In this scenario, Task 1 has priority equal to 1, Tasks 2 and 3 equal to 2, Tasks 4 and 5, equal to 3, Tasks 6 to 9 equal to 4, and finally, Task 10 holds the highest priority: 5.

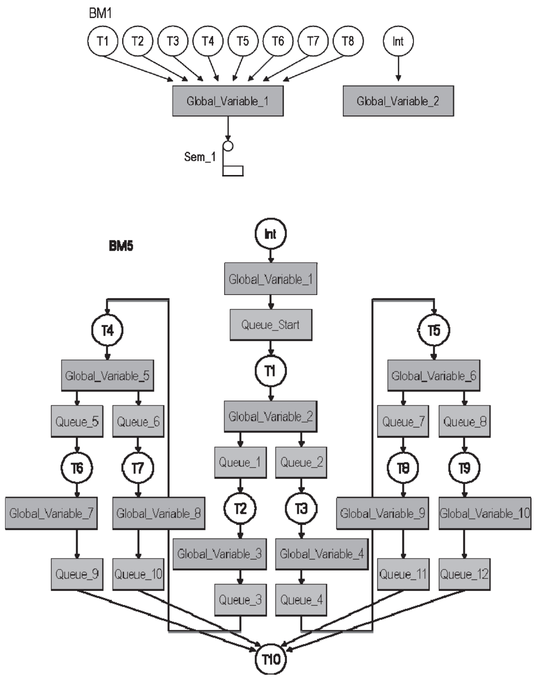

Figure 3. Functional block diagrams of the two benchmarks.

B. Experiment Setup

The first part of the experiment was dedicated to age four Virtex-4 (XC4VFX12-10SF363) FPGAs with two different levels of ^{60}Co TID radiation: two FPGAs belonged to the same fabrication lot (Lot 1), while the other two FPGAs belonged to another lot (Lot 2). Lot 1 FPGAs received a total dose of 160 krads, whereas Lot 2 FPGAs received 336 krads.

The main effect of TID radiation on CMOS devices is the reduction of the threshold voltage (V_{th}) of pMOS transistors and increase of leakage currents due to trapped charge in insulating layers. As consequence, these transistors tend to cut-off. Therefore, the "0 to 1" response of gates along with the circuit is delayed, while the "1 to 0" remains approximately the same since nMOS devices are less sensitive to this type of radiation. Thus, it can be observed the increased occurrence of "delay faults" as long as the circuit becomes old.

The experiment was performed following the standardized 1019.8 method for TID radiation testing of the MIL-STD-883H standard. The experiment was carried out at room temperature at a 155.5 rads/s dose rate, in a PISI industrial plant. Basically, the PISI plant allows a uniform irradiation of the samples when they are physically placed on known isodose surfaces. Radiocromatic perspex amber dosimeters were employed to measure TID. The exposition time for Lot 1 was 18 minutes, while for Lot 2 it was 37 minutes. In order to have a better understanding of how much this radiation level represents in terms of aging, it is worth noting that authors published a work [12] where similar FPGAs were exposed up to 420 krads working properly. It was observed that above this radiation level the component was permanently damaged, i.e, it was no more able to properly run the application. Figure 4 gives an example of the effects of aging on the FPGA electrical parameters. In this figure, 1v2 is the core *current* (resp. *voltage*), whereas 2v5 and 3v3 are the second and third periphery *currents* (resp. *voltages*), respectively. As previously mentioned, the primary consequence of TID deposition on the IC is the increase of signal propagation delay. Thus, reducing V_{DD} to the minimum possible value for system fault-free operation increases even more the signal propagation delay. As consequence, it is more difficult for the system to respect stringent timing constraints. However, it is curious to observe that the FPGA sensitivity to power supply reduction is not proportional to the aged state of the IC. In more detail, the aged FPGAs from Lot 2 (336 krads) presented the same order of magnitude of sensitivity to V_{DD} reduction as the aged FPGAs from Lot 1 (160 krads).

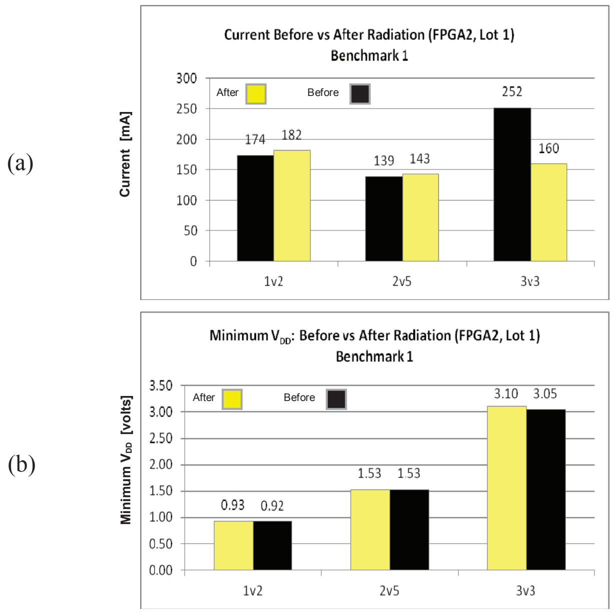

Figure 4. Electrical parameters measurements for FPGA2 (Lot 1) before and after TID deposition (160krads): (a) Static Current (I_{DDq}) consumption; (b) Minimum (V_{DD}) voltage for the FPGA to operate properly.

After aging the four FPGAs as described above, these components were characterized to radiated electromagnetic immunity (TEM Cell Method) according to the IEC 61.132-2 standard [13]. This method was adapted to a GTEM in order to have a larger cell volume to perform the experiment. Thus, it was possible to irradiate the whole set formed by the shielding box containing the board under test (whose test side was placed outwards the box). Figure 5 presents the test environment. The TEM cell test method conditions under which the four FPGAs were characterized to radiated electromagnetic immunity are respectively:

- EM field range*:* from 10 to 220 V/m (volts/meter);
- Radiated signal frequency range*:* [150kHz - 1GHz];
- Signal modulation format*:* AM/FM 80%.

C. Results' Discussion

Figure 6 compares the fault detection capability of the

RTOS-G watchdog against the native fault detection mechanisms embedded in the kernel of the Plasma RTOS before and after aging the FPGAs with TID radiation. This comparison was performed for the fresh and aged FPGAs operating in a radiated electromagnetic environment according to the IEC 61.132-2 standard for a TEM-cell test method.

The terms depicted in Figure 6 mean respectively: "Only RTOS": faults detected exclusively by the RTOS native mechanisms (these faults escaped RTOS-G detection); "Both": faults simultaneously detected by the RTOS native mechanisms and by the RTOS-G watchdog; "RTOS": faults detected by the RTOS native mechanisms (note that part of these faults might eventually be detected by the RTOS-G as well); "Only RTOS-G": faults detected exclusively by the RTOS-G watchdog (these faults escaped RTOS native mechanisms detection); and "IP RTOS-G": faults detected by the RTOS-G watchdog (part of these faults might eventually be detected by the RTOS native mechanisms as well).

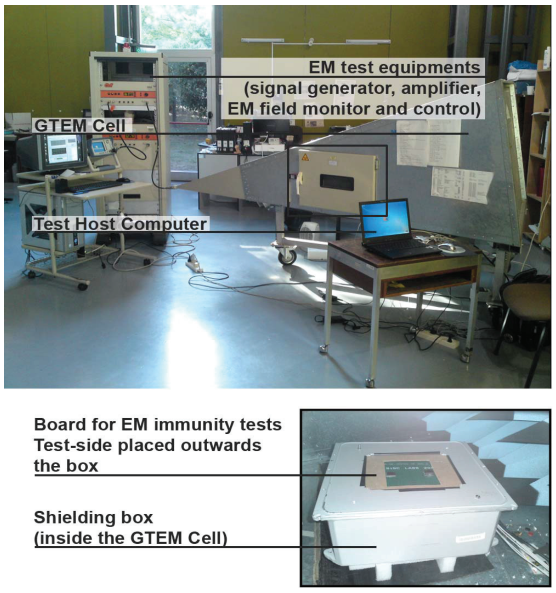
EM test equipments
(signal generator, amplifier,
EM field monitor and control)
GTEM Cell
Test Host Computer
Board for EM immunity tests
Test-side placed outwards
the box
Shielding box
(inside the GTEM Cell)

Figure 5. Environment Setup for TEM-Cell Method Measurements.

Analyzing the experimental results shown in Figure 6, one can conclude the following reasoning:

a) For any of the test cases, the fault detection capability of the RTOS-G watchdog is at least double of the one of the RTOS native mechanisms. For instance, 44.73% and 92.28% (Fig. 6a) and 17.60% and 89.28% (Fig. 6b).

b) After aging the FPGAs, this difference is even greater. For instance, 2.93% and 93.15% (Fig. 6c) and 10.98% and 97.58% (Fig. 6d).

c) After aging, it is clear the dramatic reduction of the fault detection capability of the RTOS native mechanisms: from 44.73% and 17.60% (Figs. 6a and 6b) to 2.93% and 10.98% (Figs. 6c and 6d).

d) The RTOS-G fault detection capability is independent of the aging state of the FPGAs: 92.28% and 89.28% (Figs. 6a and 6b for the fresh FPGAs) while 93.15% and 97.58% (Figs. 6c and 6d for the aged FPGAs).

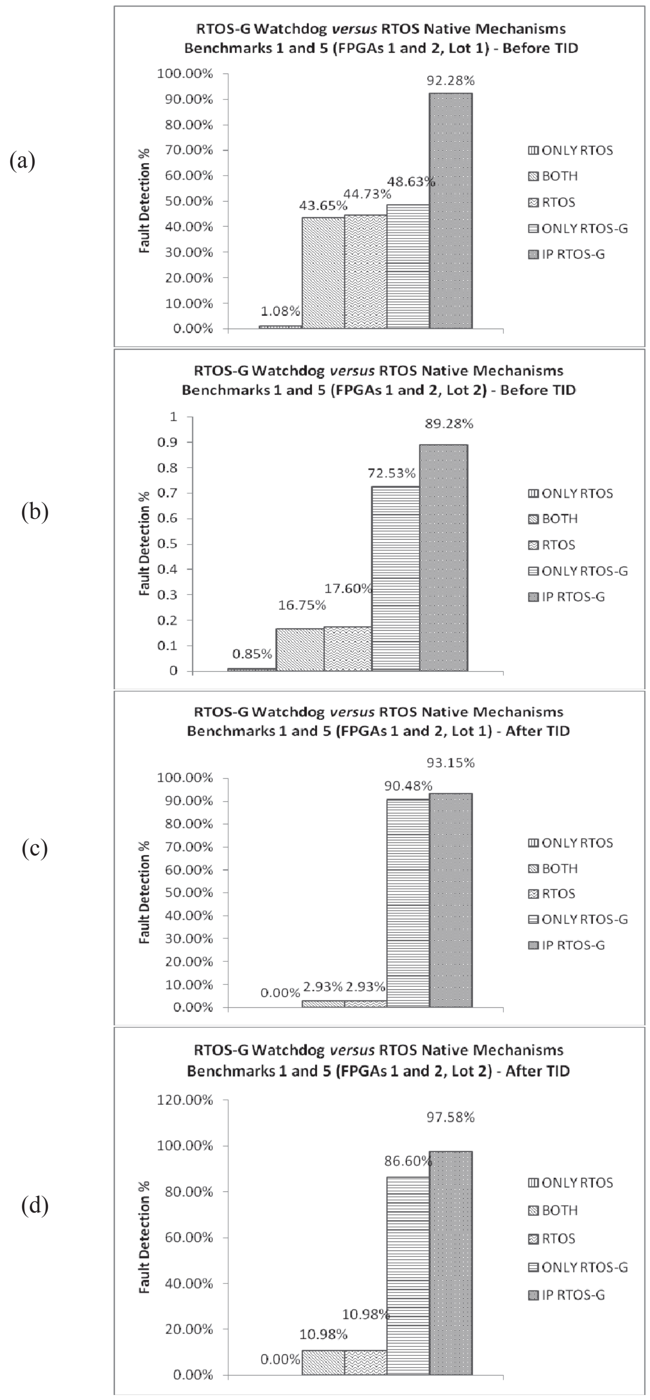

Figure 6. Comparison between the fault detection capability of the RTOS-G watchdog against the native fault detection mechanisms embedded in the kernel of the Plasma RTOS for fresh and aged FPGAs operating in an EMI-exposed environment.

We concluded that the much higher fault detection capability of the RTOS-G watchdog and its reduced sensitivity to the aging of the FPGAs is due to the following facts:

a) The watchdog is much smaller and logically less complex than the Plasma processor. Thus, rendering it intrinsically more robust to delay increase, which is the first electrical parameter to be degraded in an aged integrated circuit.

b) The processor executes applications under restrict timing control of the RTOS. Thus, the processor becomes very sensitive to delay and transient faults induced by aging the FPGAs with TID radiation.

c) As in a previous work [12], the fault detection latency of the RTOS-G watchdog was measured with the help of ChipScope (ISE-Xilinx Design Framework) and it was verified to be negligible when compared to the one of the RTOS native mechanisms. More precisely, the watchdog is capable of signaling system errors in average of 2% of the time required by the RTOS to output the same error indication. This reduced latency facilitates fault detection by the watchdog since typically, the RTOS takes several thousands of clock cycles to indicate error occurrence. Note that the processor is particularly sensitive to additional faults, if they rise up during the long time interval during which the processor is running under the control of the RTOS. If such faults occur, the system may be permanently corrupted before an error indication is yielded by the processor under the control of the RTOS.

	Frequencies or [frequency ranges] at which the system failed (MHz)	
	Pre-radiation	Post-radiation (160 krad)
FPGA 1 (Lot 1)	391	[284 – 311]
	492	[436 – 450]
	740	

	Frequencies or [frequency ranges] at which the system failed (MHz)	
	Pre-radiation	Post-radiation (336 krad)
FPGA 2 (Lot 1)	492	212
	512	[419 – 441]
	747	[492 – 507]
		522
		[538 - 543]
		560
		740

Figure 7. Frequencies at which the system failed, for fresh and aged FPGAs during the TEM-cell test method.

Figure 7 presents the frequencies where it was observed system failure during the IEC 62.132-2 TEM-cell test method. Intuitively (and as observed in this figure), the probability of coupling between an electromagnetic wave and board tracks and IC package is increased for frequencies around 600 MHz. In this case, we estimated that on-board geometries measuring 2.5 cm could behave as an efficient antenna ($\lambda/20$).

We performed this experiment by varying the electric field from 10 to 220 volts/meter. Approximately from 170 volts/meter, we started observing the occurrence of transient faults on the FPGA. As observed, the aged FPGAs failed at a larger number of frequencies when compared to the fresh FPGAs (before TID-exposure). Furthermore, the aged FPGAs failed mostly along with a whole frequency range (instead of specific frequency points). For instance, when irradiating the fresh system inside the GTEM cell, we observed that it failed precisely on the frequency of 391 MHz (we repeated several times the test execution when we observed a failure, to be sure that we were measuring the good numbers). On the other hand, when we irradiated the aged system, we observed that it failed not only on the 391 MHz frequency, but in a large frequency range: [294 - 311] MHz (Fig. 7, FPGA 1 / Lot 1).

This scenario demonstrates experimentally that aged FPGAs are more sensitive than fresh FPGAs to transient faults induced by electromagnetic environments like the one promoted by the TEM-cell test method. Therefore, the use of the proposed approach yields even more benefits as time pass by during system lifetime.

V. Conclusions

We analyzed the fault detection capability of an on-chip watchdog (RTOS-G) designed to detect transient faults affecting the task scheduling process of real-time operating systems (RTOSs). The reliability analysis was performed by means of a case-study implementation based on the soft-core processor Plasma (opencores.org) and the RTOS-G watchdog that were prototyped into the Xilinx Virtex4 FPGA. This IC was aged under exposition to total ionizing dose (TID) radiation and the system performance analyzed under TEM-cell test method conditions. The obtained results for the fresh (pre-radiation) and aged (post-radiation) experiments indicate that the RTOS-G watchdog presents a much higher fault detection capability when compared with the conventional RTOS native fault detection mechanisms. This scenario is even more evident when the system is aged, since it is more sensitive to transient faults in a larger frequency spectrum.

References

[1] N. Ignat, B. Nicolescu, Y. Savari, G. Nicolescu, "Soft-Error Classification and Impact Analysis on Real-Time Operating Systems", IEEE Design, Automation and Test in Europe, 2006.

[2] S. Ben Dia, R. Ramdani, E. Sicard, "Electromagnetic Compatibility of Integrated Circuits – Techniques for Low Emission and Susceptibility", Springer, 2006.

[3] J. Freijedo, L. Costas, J. Semião, J. J. Rodríguez-Andina, M. J. Moure, F. Vargas, I. C. Teixeira, and J. P. Teixeira, "Impact of power supply voltage variations on FPGA-based digital systems performance", Journal of Low Power Electronics, vol. 6, pp. 339-349, Aug. 2010.

[4] J. Semião, J. Freijedo, M. Moraes, M. Mallmann, C. Antunes, J. Benfica, F. Vargas, M. Santos, I. C. Teixeira, J. J. Rodríguez-Andina, J. P. Teixeira, D. Lupi, E. Gatti, L. Garcia, F. Hernandez, "Measuring Clock-Signal Modulation Efficiency for Systems-on-Chip in Electromagnetic Interference Environment". 10th IEEE Latin American Test Workshop (LATW'09), March 2009.

[5] http://public.itrs.net

[6] D. Mossé, R. Melhelm, S. Gosh, "A non-preemptive real-time scheduler with recovery from transient faults and its implementation", IEEE Trans. on Software Engineering, Vol. 29, N°. 8, pp. 752-767, August, 2003.

[7] B. Nicolescu, N. Ignat, Y. Savaria, G. Nicolescu, "Analysis of Real-Time Systems Sensitivity to Transient Faults Using MicroC Kernel," IEEE Transactions on Nuclear Science, Vol. 53, N° 4, August 2006.

[8] S. Gosh, R. Melhem, D. Mossé, J. Sarma, "Fault-tolerant Rate Monotonic Scheduling", Journal of Real-time Systems, Vol. 15, N° 2, Sept. 1998.

[9] P. Mejia-Alvarez, D. Mossé, "A responsiveness approach for scheduling fault-recovery in real-time systems", 5th Real-Time Technology and Applications Symposium, pp. 83-93, 1999.

[10] Ph. Shirvani, R. Saxena. E. J. McCluskey, "Software-implemented EDAC protection against SEUs", IEEE Trans. on Reliability, Vol. 49, N° 3, pp. 273-284, Sept. 2000.

[11] J. Tarrillo, L. Bolzani, F. Vargas, "A Hardware-Scheduler for Fault Detection in RTOS-Based Embedded Systems", IEEE 12th EUROMICRO Conference on Digital System Design, 2009.

[12] Benfica, J.; Bolzani Poehls, L. M.; Vargas, F.; Lipovetzky, J.; Lutenberg, A.; Garcia, S. E.; Gatti, E.; Hernandez, F. Evaluating the Effects of Combined Total Ionizing Dose Radiation and Electromagnetic Interference. IEEE Transactions on Nuclear Science, Vol. 59 ,Issue 4, pp. 1015-1019, Aug. 2011.

[13] "IEC 62.132-2, Ed. 1: Integrated Circuits - Measurements of Electromagnetic Immunity, 150KHz to 1GHz - Part 2: Measurement of radiated immunity - TEM-Cell Method. 2004-07". (www.iec.ch)