Estimation of Hardware Task Reliability on Partially Reconfigurable FPGAs

Reza Ramezani[1], Juan Antonio Clemente[2], Yasser Sedaghat[1], Hortensia Mecha[2]
reza.ramezani@stu.um.ac.ir, ja.clemente@fdi.ucm.es, y_sedaghat@um.ac.ir, horten@dacya.ucm.es
[1] DDEmS Lab, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[2] Computer Architecture Department, Universidad Complutense de Madrid, Madrid 28040, Spain

***Abstract*- This paper presents a novel statistical model to estimate the reliability and number of errors of hardware tasks running on partially reconfigurable FPGAs in harsh environments. The proposed model has been validated by means of fault injection. The obtained results endorsed by the 95% confidence interval reveal the high accuracy of the proposed reliability model.**
Keywords: Reliability Model, Soft Error Rate, Fault Injection, FPGAs.

I. Introduction and Related Work

New-generation safety-critical embedded systems demand high performance, reliability, efficiency and flexibility. These characteristics are especially dominant in fields such as avionics and aerospace. Modern SRAM-based FPGAs are a good solution to cope with these requirements, thanks to their ability to modify part of their functionality on-the-fly, which is known as dynamic partial reconfigurability.

However, space computing systems are exposed to high doses of radiation in comparison to those operating at ground level [1]. The negative effect of said radiation is the occurrence of the so-called Single Event Upsets (SEUs), which is a broad category of events by which a single particle strike eventually causes memory cell upsets [2]. Therefore, designers should simultaneously consider reliability and performance of such systems.

This paper presents a mathematical model that estimates the reliability of hardware tasks running on partially reconfigurable devices (in particular, FPGAs). The presented model incorporates some parameters including task computation time, task size, the percent of task sensitive bits, and the Soft Error Rate (SER) of the environment. The validity of the proposed reliability model has been examined by the fault injection platform NESSY [3, 4] on a number of actual hardware tasks running on Xilinx™ Virtex-5 XC5VLX110T FPGA. The obtained results endorsed by the 95% confidence interval reveal the high accuracy of the proposed model, especially in harsh environments, in which the discrepancy between estimations and the experiments is less than 0.5%.

II. Related Work

Many studies in the literature have focused on assessing the reliability of SRAM-based FPGA designs. For example, Héron et al. [5] have introduced a micro-view reliability model that, by combining soft error sensitivity and physical reliability, divides a given design into multiple partitions and basic elements. Finally the reliability of these basic elements is estimated to derive that of the entire design. In a similar work, the reliability of an FPGA design is estimated by calculating the error propagation probability of the low-level gates [6].

The model presented by Ostler et al. [7] is another study in this area that considers coincident upsets and estimates the reliability of a given FPGA design protected by TMR and scrubbing. In that work, reliability is a design-specific parameter that is obtained by fault injection or accelerated radiation experiments, rather than relying on theoretical models. In a similar way, many studies analyze the SEU sensitivity of SRAM-based FPGAs by emulating SEUs through a fault injection mechanism into the device configuration files [8, 9].

The work by Edmonds et al. [10] is another example of reliability estimation in which the probability of a system failure during a single clock cycle is derived to obtain an equation which calculates the system error rate.

The reliability model presented in this work employs a macro-view approach that, without requiring any fault-injection or radiation-accelerated experiment, pays attention to the occurrence of one or more upsets during task execution. This model uses simple parameters of task and environment to estimate the reliability of the hardware tasks.

III. System Model

Since the size of the running tasks depends on the number of occupied basic reconfigurable elements and the architecture of the underlying FPGA, in this section both hardware tasks and the target FPGA have been modeled.

A) Task Model

In this paper, each hardware task τ is modeled as follows:

$$\tau = \{CT_\tau, CC_\tau, TS_\tau, SB_\tau\} \qquad (1)$$

where CT_τ is task computation time, CC_τ is task size in terms of basic reconfigurable elements, TS_τ is task size in the configuration memory (number of configuration bits), and SB_τ is the percent of sensitive bits of the task.

Any bit flip in a sensitive bit leads to the task failure [11]. The number of sensitive bits of a design can be estimated by means of fault injection, fault emulation or even radiation-ground experiments [8]. To the best of our knowledge, 35% is the highest value of sensitive bits reported in the literature [12]. Anyway, the sensitive bits of a task can be pessimistically estimated as 100% to obtain a lower bound of the task reliability.

Since the validity of the presented model has been verified on a Virtex-5 FPGA, in this case the basic reconfigurable elements are *Configurable Logic Blocks* (CLBs). However, this idea can be easily ported to any other FPGA in the market, since all of them feature any sort of basic reconfigurable elements, such as *Logic Elements* (LE) in Altera architectures.

B) Partially Runtime Reconfigurable FPGA Model

The target FPGA features partial runtime reconfigurability and it includes an array of basic reconfigurable elements (CLBs in case of the FPGA used in this paper), so that hardware tasks are synthesized and mapped on a subset of them. Typically, in modern FPGAs, a single basic element is not the minimum addressable segment of the device that can be reconfigured separately. Instead, a set of them must be reconfigured at the same time, which in this paper is referred to as *CLB group*. Thus, the target FPGA-based computer RC is modeled as:

$$\mathrm{RC} = (RO, CO, GS, CB) \qquad (2)$$

978-1-5090-4366-8/16/$31.00 ©2016 IEEE

where RO and CO indicate row count and column count of the FPGA, respectively. GS stands for CLB group size (in terms of CLB count in each group), and finally CB is the number of configuration bits that a CLB group contains.

The presented reliability model is built upon the 2D area model. The area model describes the way in which the hardware tasks are arranged within the FPGA. In 2D area model, tasks are modeled as rectangles that can span a rectangular subset of resources within the FPGA [13]. Task size (TS_τ) –in terms of configuration bits– depends on the number of occupied basic reconfigurable elements and the FPGA area model. In order to estimate this parameter, in the 2D area model, let RO_τ and CO_τ be the number of rows and columns that task τ occupies, respectively. TS_τ is obtained as:

$$TS_\tau = CO_\tau * \left\lceil \frac{RO_\tau}{GS} \right\rceil * CB \qquad (3)$$

IV. Reliability Model and its Validation

A) The Proposed Reliability Model

The model presented in this paper assumes that upsets accidentally occur in hardware tasks running on a SRAM-based FPGA, according to a given SER. Then, using simple equations, it estimates the reliability and number of errors (#Errors) that are expected to observe in said tasks. As indicated by [1], different altitudes above the Earth surface have different SERs, which can be measured *per bit per time unit*. Reliability of a task τ (denoted as R_τ) is the probability that the task executes from its start time to its finish time without any failure, with the condition that the task had no error when starting its execution.

The proposed model assumes that at most one upset occurs at a time, but one or more upsets might occur during task execution. In addition, since not all bits of a hardware task are sensitive, the model only takes the occurrence of upsets in sensitive bits into account.

Upsets caused by radiations can be regarded as independent and random statistical events. Thus, it is reasonable to make the following assumptions:

(1) The number of upsets during a given interval of time depends only on the length of the interval and not on the past history of the system.

(2) For any small time interval $(t, t + \delta t)$, the probability of a single upset is $\mu * \delta t$. μ is a constant, while the probability of more than one occurrence is negligible.

Based on these assumptions, the bit flips caused by radiations, follow the Poisson distribution. Thus, the probability of, at least, one bit flip in the sensitive bits of task τ, given j upsets, can be obtained as:

$$P(F_{\tau,j}) = e^{-v_\tau} \frac{{v_\tau}^j}{j!} \qquad (4)$$

where

$$v_\tau = \mu * (TS_\tau * SB_\tau) * \alpha CT_\tau \qquad (5)$$

in which μ is the SER of environment expressed *per bit per time unit*, and α is used for task computation time relaxation (which by default is 1).

Let $P(F_\tau)$ indicate the probability of failure of task τ given j upsets, j ranging from 1 to ∞ during task execution. Therefore we have:

$$P(F_\tau) = \sum_{j=1}^{\infty} P(F_{\tau,j}) \qquad (6)$$

By having $P(F_\tau)$, the reliability of task τ is obtained as:

$$R_\tau = 1 - P(F_\tau) \qquad (7)$$

Once the task reliability R_τ is achieved, the Mean Time to Failure (MTTF, [7]) is calculated from:

$$MTTF_\tau = \frac{CT_\tau}{1 - R_\tau} \qquad (8)$$

Finally, the number of errors ($\#Errors_\tau$), observed in the task, during the operating time of the system ($LifeTime$) is estimated as:

$$\#Errors_\tau = \frac{LifeTime}{MTTF_\tau} \qquad (9)$$

B) Reliability Model Validation

The reliability model described in Subsection IV.A has been validated by means of simulation-based experiments and fault injection. The validation procedure takes two distinct statistical events into account. The first event is *the occurrence of j upsets during the execution of task τ among its configuration bits*. The second event is *the probability of, at least, one upset in sensitive bits of task τ given j upsets scattered in all its configuration bits*.

The validation procedure is described in detail in Algorithm 1. It receives three inputs: the hardware task τ, the lifetime of the system upon which task τ will run ($LifeTime$), and the maximum number of upsets that might occur during the execution of task τ to simulate Eq. (6) ($MaxUpsets$). It is important to note that this equation converges very rapidly, which makes it possible to use this upper bound for the infinite summation of said equation. Since the configuration and execution times of tasks are usually in the order of milliseconds [14], in this paper, millisecond has been used as time unit.

The experiment environment runs task τ for $LifeTime$ time units, therefore task τ is executed for $LifeTime/CT_\tau$ times. In this algorithm, for all these individual runs (Line 1), and for all the possible upsets (Line 2), task τ is firstly initialized in NESSY (Line 3). Then, for all the time units in τ (Line 4), the algorithm firstly determines if j upsets occur. For this purpose, a metric named *Milliseconds to Upsets* is used, which stands for the time that it takes (on average) to flip j bits of configuration data of task τ during its execution. The value of $Milliseconds\ to\ Upsets\ (\tau, j)$ is obtained from Eq. (10), but since all bits of task τ might be flipped, in order to calculate $P(F_{\tau,j})$ in this equation, it is assumed all bits of task τ are sensitive (i.e. $SB_\tau = 1$).

$$Milliseconds\ to\ Upsets\ (\tau, j) = \frac{CT_\tau}{P(F_{\tau,j})} \qquad (10)$$

In order to determine if j upsets occur during task execution (CT_τ), a random number (RND) is generated between 1 and $Milliseconds\ to\ Upsets\ (\tau, j)$ (Line 5). If RND = 1, it means the occurrence of j upsets (Line 6). Note that, by increasing j, $Milliseconds\ to\ Upsets\ (\tau, j)$ increases as well, which makes this event (i.e., RND = 1) less probable to happen. When j upsets occur, they are emulated by NESSY fault injection tool [3, 4]: the task runs until the current time unit (Line 7), and the upsets are injected in the task randomly in any of its configuration bits (line 8). This process continues for all the time units of τ.

Once this happens, τ finishes its execution in NESSY (Line 11) and its final output vector is compared with the "golden" one (which had been obtained without fault injection). If they do not match, at least, one of the upsets injected throughout its execution has flipped sensitive bits which causes task failure. Thus, the errors count is updated and the time elapsed between

Algorithm 1 – Procedure of the reliability validation

Input task $\tau = \{CT_\tau, CC_\tau, TS_\tau, SB_\tau\}$
Input MaxUpsets
Input LifeTime
1. **for** i = 1 **to** $LifeTime/CT_\tau$ {
2. **for** j = 1 **to** $MaxUpsets$ {
3. **Initialize** τ in NESSY;
4. **for** k = 1 **to** CT_τ {
5. RND = **Random** (1, Millisecond to Upsets(τ, j));
6. **if** (RND = 1) {
7. **Run** τ in NESSY **until** k time units;
8. **Toggle** j random bits of τ configuration data;
9. }
10. }
11. **Run** τ **in** NESSY **until** CT_τ;
12. **if** (output vector ≠ "golden" output vector) {
13. $\#Errors_\tau$++;
14. **Calculate** TBF_τ;
15. }
16. **Restore** task with "golden" configuration data;
17. }
18. }
19. **return** R_τ, $MTTF_\tau$, $\#Errors_\tau$;

this failure and the previous one is recorded in the variable *Time Between Failures (TBF)* (Lines 12-15). Finally, the golden configuration of τ is restored for the next task execution (Line 16). When all the executions of τ are completed, $MTTF_\tau$ is calculated as follows:

$$MTTF_\tau = \frac{\sum TBF_\tau}{\#Errors_\tau} \qquad (11)$$

Finally, following Eq. (8), the reliability of task τ is calculated as:

$$R_\tau = 1 - \frac{CT_\tau}{MTTF_\tau} \qquad (12)$$

C) The NESSY Fault Injection Tool

The validation procedure has used the fault injection tool NESSY [3, 4] to inject upsets and manage the task executions. The microprocessor-based hardware system has been implemented in the FPGA, where the hardware task under test is also placed. NESSY tool has been used to inject the bit-flips in the configuration bits and to compare results of the task execution. NESSY and the hardware task are placed in disjoint partially reconfigurable regions, thereby guaranteeing non-intrusiveness. For the sake of efficiency, the microprocessor-based hardware system uses the Internal Configuration Access Port (ICAP) for bit-flip injection. Afterwards, the hardware task is executed and its results are compared with the "golden" result to check whether the injected faults lead to a failure in the normal operation of the hardware task under test or not.

V. Experimental Results

A) Experimental Setup

The proposed model has been validated by a system consisting of a PC and a Xilinx™ Virtex-5 XC5VLX110T FPGA. The generation of upsets has been simulated on the PC by generating random numbers. Once the upsets have been generated, our NESSY fault injection tool is used to emulate upsets in the hardware task under test.

Table 1 - Estimated SER for different orbits and solar conditions

	Upset/bit/Day (μ)		
Sol. Cond. → / Orbit ↓	**Solar Max**	**Worst Day**	**Peak 5-Min**
GEO	6.09E-08	3.35E-04	1.29E-03
GPS	6.09E-08	2.89E-04	1.10E-03
Polar	2.25E-07	7.99E-05	2.97E-04

To have realistic SERs, in this work, we have focused on the following three "harsh" orbits: geostationary Earth orbit (GEO), global positioning system (GPS), and Polar [7]. SERs are estimated in each orbit for the worst day, the peak five minutes, and the solar max conditions of a Solar Energetic Particle (SEP) event (see Table 1).

As noted in [15], ITC'99 is one of the best task sets to assess radiation effects on hardware tasks. In this paper, tasks b01 to b13 from this task set have been used for the experiments. The underlying FPGA has operated at a frequency of 100MHz and the hardware tasks have been executed for different clock cycles to obtain different execution times in the range of [10 ... 500] *ms*. The sensitivity of the tasks have been obtained from the NESSY tool [3, 4].

For each experiment, the lifetime of the system has been set to one year. Each experiment has then been repeated 10 times and the average of the results has been calculated as the final result.

B) Results

The experiments have been done for all orbits and solar conditions. The results of the experiments have been presented in Figure 1, but due to the limited space, only four different values from the SERs in Table 1 have been chosen. In this case, only GEO peak 5-min, GPS peak 5-min, GEO worst day, and Polar worst day have been examined. This figure shows the #Errors of the presented model and the experiments (model validation) in logarithmic scale.

As the experiments are stochastic and also they are based on different trials, it is mandatory to calculate confidence margins. For these experimental results, the well-known 95% confidence interval has been used, as explained in [16]. Confidence intervals have also been added to the experimental results of Figure 1 as vertical lines.

When #Errors is more than 50, the relative 95% confidence interval of the #Errors can be approximated with the Normal distribution as shown by Eq. (13). In the other cases, when #Errors is less than 50, the Poisson error bars for absolute 95% confidence intervals reported in [17] can be used.

$$Relative\ 95\%\ Confidence\ Interval = \pm\frac{2\sqrt{\#Errors}}{\#Trials} \qquad (13)$$

As the obtained results show, a lower #Errors is observed for the lowest SERs, which leads up to 22.5% of discrepancies between the estimated and the experimental results. In spite that, with few #Errors, this discrepancy is significant, the value of the estimations are still acceptable since, in all the cases, they lie in the 95% confidence interval. Of course, as the SER increases, #Errors increases as well. In this case, the discrepancy of the results is as less as 0.05%. This shows the high accuracy of the proposed model.

Last but not the least, it is noteworthy to state that the presented model estimates the reliability of hardware tasks by simple equations. In contrast, the fault injection tools take hours or even days, depending on the size and execution time of a task, to estimate the hardware tasks reliability.

VI. Conclusions and Future Work

This paper has presented a novel mathematical model to estimate the reliability of hardware tasks running on reconfigurable devices, without requiring any fault-injection or radiation-accelerated experiment. For that purpose, this model incorporates some features of the hardware tasks including task computation time, task size, and the percent of the task sensitive bits, and then it estimates the reliability of the tasks

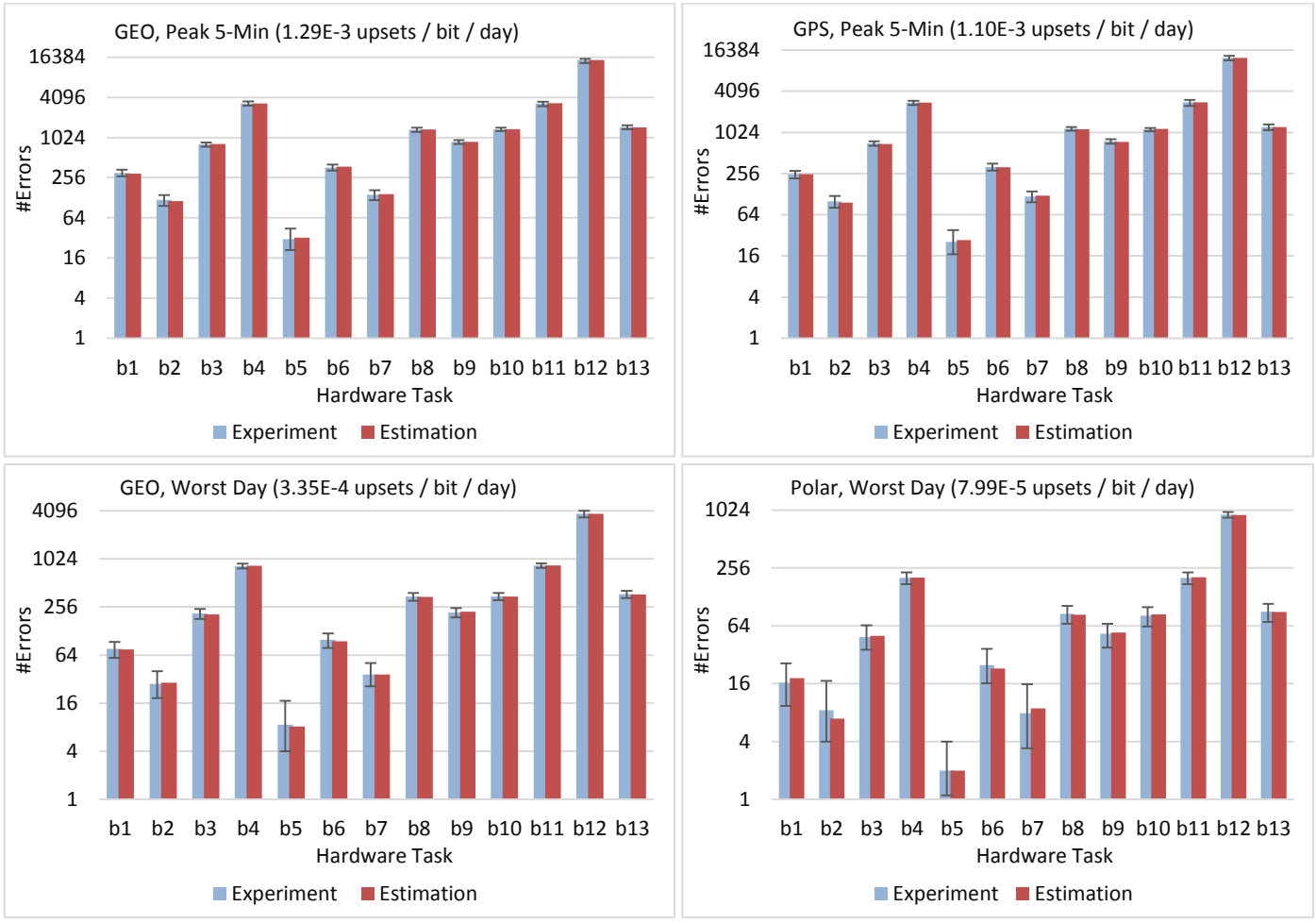

Figure 1 - #Errors obtained from Estimations of the proposed reliability model and from the experiments (logarithm 2 scale)

under a given SER. By using this reliability model, other researchers can estimate the number of errors that are expected to occur in a given hardware task, for harsh environments.

The proposed reliability model has been evaluated by means of fault injection. The obtained results revealed the high accuracy of the proposed model, especially in harsh environments.

For future work we wish to pay attention to other resources of modern FPGAs, as well as taking MBUs into account.

ACKNOWLEDGEMENT

This work was supported in part by the Spanish Ministry of Science and Innovation (MCINN) through grant TIN2013-40968-P.

VII. REFERENCES

[1] J. Zhang, Y. Guan, and C. Mao, "Optimal Partial Reconfiguration for Permanent Fault Recovery on SRAM-based FPGAs in Space Mission," *Advances in Mechanical Engineering,* vol. 5, 2013.

[2] M. Niknahad, O. Sander, and J. Becker, "Fine Grain Fault Tolerance—A Key to High Reliability for FPGAs in Space," in *Aerospace Conference, IEEE*, 2012, pp. 1-10.

[3] V. Alaminos, F. Serrano, J. A. Clemente, and H. Mecha, "NESSY: An Implementation of a Low-cost Fault-injection Platform on a Virtex-5 FPGA," in *the Conference on Radiation and its Effects on Components and Systems (RADECS)*, 2012.

[4] F. Serrano, J. A. Clemente, and H. Mecha, "A Methodology to Emulate Single Event Upsets in Flip-Flops Using FPGAs through Partial Reconfiguration and Instrumentation," *Nuclear Science, IEEE Transactions on,* vol. 62, pp. 1617-1624, 2015.

[5] O. Héron, T. Arnaout, and H.-J. Wunderlich, "On the Reliability Evaluation of SRAM-based FPGA Designs," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2005, pp. 403-408.

[6] H. Asadi, M. B. Tahoori, B. Mullins, D. Kaeli, and K. Granlund, "Soft Error Susceptibility Analysis of SRAM-based FPGAs in High-performance Information Systems," *Nuclear Science, IEEE Transactions on,* vol. 54, pp. 2714-2726, 2007.

[7] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou, P. S. Graham, K. S. Morgan, B. H. Pratt, H. M. Quinn, and M. J. Wirthlin, "SRAM FPGA Reliability Analysis for Harsh Radiation Environments," *Nuclear Science, IEEE Transactions on,* vol. 56, pp. 3519-3526, 2009.

[8] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator Validation of an FPGA SEU Simulator," *Nuclear Science, IEEE Transactions on,* vol. 50, pp. 2147-2157, 2003.

[9] P. Sundararajan, S. McMillan, B. Blodget, C. Carmichael, and C. Patterson, "Estimation of Single Event Upset Probability Impact of FPGA Designs," in *Military and Aerospace Programmable Logic Devices Conference (MAPLD)*, 2003.

[10] L. D. Edmonds, "Analysis of Single-event Upset Rates in Triple-modular Redundancy Devices," NASA Jet Propulsion Laboratory, 2009.

[11] I. Herrera-Alzu and M. Lopez-Vallejo, "System Design Framework and Methodology for Xilinx Virtex FPGA Configuration Scrubbers," *Nuclear Science, IEEE Transactions on,* vol. 61, pp. 619-629, 2014.

[12] J. S. Monson, M. Wirthlin, and B. Hutchings, "A Fault Injection Analysis of Linux Operating on an FPGA-embedded Platform," *International Journal of Reconfigurable Computing,* vol. 2012, 2012.

[13] C. Steiger, H. Walder, and M. Platzner, "Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-time Tasks," *Computers, IEEE Transactions on,* vol. 53, pp. 1393-1407, 2004.

[14] X. Iturbe, K. Benkrid, T. Arslan, I. Martinez, and M. Azkarate, "ATB: Area-time Response Balancing Algorithm for Scheduling Real-time Hardware Tasks," in *International Conference on Field-Programmable Technology (FPT)*, 2010, pp. 224-232.

[15] H. Quinn, W. H. Robinson, P. Rech, M. Aguirre, A. Barnard, M. Desogus, L. Entrena, M. Garcia-Valderas, S. M. Guertin, and D. Kaeli, "Using Benchmarks for Radiation Testing of Microprocessors and FPGAs," *Nuclear Science, IEEE Transactions on,* vol. 62, pp. 2547-2554, 2015.

[16] R. Velazco, J. A. Clemente, G. Hubert, W. Mansour, C. Palomar, F. J. Franco, M. Baylac, S. Rey, O. Rosetto, and F. Villa, "Evidence of the Robustness of a COTS Soft-error Free SRAM to Neutron Radiation," *Nuclear Science, IEEE Transactions on,* vol. 61, pp. 3103-3108, 2014.

[17] H. Quinn, "Challenges in Testing Complex Systems," *Nuclear Science, IEEE Transactions on,* vol. 61, pp. 766-786, 2014.