This is a previous version of the article published in IEEE Transactions on Nuclear Science. 2020, 67(7): 1511-1520. doi:10.1109/TNS.2020.2993637

Empirical mathematical model of microprocessor sensitivity and early prediction to proton and neutron radiation-induced soft errors

Alejandro Serrano-Cases[1], Leonardo Maria Reyneri[2], Yolanda Morilla[3], Sergio Cuenca-Asensi[1] and Antonio Martínez-Álvarez[1]✉

***Abstract*—A mathematical model is described to predict microprocessor fault tolerance under radiation. The model is empirically trained by combining data from simulated fault-injection campaigns, and radiation experiments, both with protons (at the CNA facilities, Seville, Spain) and with neutrons (at the LANSCE Weapons Neutron Research facility at Los Alamos, USA). The sensitivity to soft errors of different blocks of commercial processors is identified to estimate the reliability of a set of programs that had previously been optimized, hardened, or both. The results showed a standard error under 0.1, in the case of the ARM processor, and 0.12, in the case of the MSP430 microcontroller.**

***Index Terms*—Fault tolerance, single event upset, proton/neutron irradiation effects, soft errors**

I. Introduction

CRITICAL system developers are increasingly concerned, in view of the continuous reduction in transistor sizes and the consequent effects, because smaller transistors are more prone to errors caused by ionizing particles [1], [2]. Modern electronic devices offer energy-aware and high-performance solutions, however new technological trends focus on low-consumption devices with reduced noise margins, which increase their sensitivity to noise and disruption. For instance, faults caused by ionizing particles [3], [4] could lead to a catastrophic situation in a critical system, installed in a satellite, an airplane or an autonomous vehicle, if the system were to enter a non-operative or an unstable state that can provoke erroneous decisions.

The focus of this study is on radiation-induced transient faults, also known as soft errors. Such faults mainly affect the storage elements of the processors (latches, registers, memory cells,...), provoking temporary data errors and interrupting, for example, the flow of instructions. In this field, the literature offers a large number of techniques designed to increase the reliability of critical systems, which can be classified in accordance with where they are applied, as: hardware, software or hybrid.

Redundancy-based techniques are commonly used for greater reliability. It is known that their exhaustive analysis can locate and identify possible side effects, thereby reducing the other overheads associated with these sorts of techniques, such as: design and fabrication costs, flexibility, energy consumption, performance, fault-coverage and area. In the case of hardware approaches, Triple Modular Redundancy (TMR) [5], [6] is a widely used technique, which shows high error resilience and performance. However, it has high associated costs, such as lengthy development times and high-power consumption. In addition, the resulting custom hardware is targeted at a specific device and therefore a very expensive solution. In contrast, Software-Implemented Hardware Fault Tolerance (SIHFT) [7], [8] is a software programming approach to deal with faults affecting the hardware. These approaches are less expensive and have shorter development timelines than the hardware approaches. However, they are not exempt from overheads, because replication is produced spatially and temporally. Hybrid approaches emerge as a way of improving both aforementioned strategies, by reducing their overheads [9]–[11]. Finally, the literature presents several approaches, such as the work of [12], which do not fit into any of the aforementioned classifications, because neither the hardware nor the software (source code) is modified for greater reliability. Instead, the code generation is tuned to improve reliability, performance and size.

Designing a fault-tolerant system, using one of the above-mentioned techniques, usually requires an analysis of various configurations, considering at all times the requirements of each task, for an ideal combination of hardening parameters (e.g. registers refresh rate, number and location of redundancies, error-checking mechanism, etc.). Each configuration must be evaluated according to the task requirements, looking for the best trade-off between reliability and overheads. In this sense, fault injection campaigns [13]–[15] have been extensively used, because of their flexibility, observability, and low-cost features. However, neither simulation models nor emulation tools possess the fine-grained architectural information and/or accessibility to produce accurate reliability forecasts. Therefore, fault injection is usually consigned to early development stages, while accelerated radiation tests are mandatory in the final stages, prior to the system deployment phase.

Radiation experiments normally involve particle accelerators that expose the devices to realistic conditions where their behavior can be assessed throughout their service life [16]. However, a single test may take several days to be completed,

[1]Dept. of Computer Technology, Ctra. San Vicente del Raspeig s/n, 03690, San Vicente del Raspeig - Alicante, Spain, (✉) e-mail: amartinez@dtic.ua.es

[2]Dept. of Electronics, Politecnico di Torino, Corso Duca d. Abruzzi 24, Turin, Italy, leonardo.reyneri@polito.it

[3]Centro Nacional de Aceleradores (Universidad de Sevilla, CSIC, JA). Avda. Tomás Alba Edison 7, Sevilla, Spain

its repeated use is financially prohibitive, and the particle beam time slots at most facilities are very limited. Under such circumstances, an exhaustive evaluation of the different hardware/software/hardening configurations of the system is impractical.

To alleviate those difficulties, the present work proposes a model to predict the fault tolerance of the application/microprocessor pair under radiation. Our model combines simulated results from fault injection campaigns with results obtained in radiation tests. We used a Multiple Linear Regression approach with a gradient descent algorithm to fit the model parameters. It is intended to help the designers with the exploration of the trade-offs when SIHFT techniques are applied to the programs. To this end, and contrary to other approaches, different cross-sections were considered for different processor blocks, thus adding more flexibility to the model. Furthermore, those cross-sections are not directly measured in one unique radiation test, but empirically adjusted by a number of experimental and simulated campaigns. In this way, the model has a previous training phase where several parameters are fitted to optimize its performance. Once trained, the model can be used to estimate the sensitivity to soft-errors of the different processor blocks and to predict their reliability under radiation.

A preliminary work was presented in [17] that focused on optimizing the application of a High Level SIHFT technique to a reduced number of programs. This work extends that effort in two ways: 1) characterizing two platforms (ARM and MSP430) using the parameters obtained from the trained models; and, 2) analyzing and optimizing the application of three different SIHFT techniques to the benchmark suite: low-level implemented, high-level implemented and non-intrusive.

The remainder of this paper will be organized as follows: related works will be reviewed in Section II; in Section III the model will be described; in Section IV the test boards, facilities, fluxes and benchmarks will be specified; likewise, in Section V, each training step for each model and the model evaluation procedure will be detailed; then, an introduction to the operation of the model for early reliability characterization of applications and hardening techniques will be presented in Section VI. Finally, the conclusions and the contributions of this study will be highlighted in Section VII.

II. Related works

Two main algorithmic approaches have been proposed to overcome the limitations, in terms of observability and economic cost, of radiation experiments: Machine Learning-based (ML-based) and Model-based (M-based).

On the one hand, Machine Learning methods rely on the capability of its algorithms to learn from examples, with no prior knowledge of the relation between the input (i.e. application and processor features) and output variables (i.e. soft-error rate). ML-based methods have been used in several ways. Vishnu et al. [18] proposed a combination of eight ML algorithms, to assess the impact of multi-bit memory errors on HPC applications and to compare their predictions with the fault-injection results. In [19], the authors used linear-regression techniques to backtrack the propagation of soft errors through processes dependent on many-core processor systems. Considering the application of non-intrusive SIHFT methods, the works described in [20] and [21] employed genetic algorithms that select the best compilation flags of certain applications for their reliability optimization. Rocha et al. [22] have recently proposed a *soft-error score*, which combines supervised and unsupervised ML algorithms, to correlate application profiles and processor characteristics with fault-injection results. The score was devised to assist researchers with the identification of the parameters that have most influence on the reliability of the final application.

On the other hand, Model-based predictors are employed to compensate the traceability limitations of the errors observed in the radiation experiments with the vulnerability estimations offered by fault injection. Rezgui et al. [23] were the first to propose the prediction of the Single Event Upset (SEU) program cross-section, both from the static SEU cross-section derived from radiation testing and the error rates noted in fault-injection campaigns. According to this model, the sensitivity of a program to SEUs can be computed as the product between the cross-section of the underlying processor technology and the fraction of upsets with consequences (errors) when running the program. Faults were emulated on real devices (microcontrollers) with a limited amount of memory and resources. That model was extended to multi-core and many-core processors [24]. Two additional *moderating factors*, which took account of input from shared and cache memories, computed while running the program, were proposed in the same study.

Recently, the original model has also been used by other authors in [25], to predict the Failures in Time (FIT) of an ARM processor executing a code. In this case, fault injection campaigns were performed on a large benchmark suite running on a cycle-accurate simulator. The contributions of the main processor blocks (cache data and instruction memories, TLBs and register file) were estimated and summed up by a global vulnerability factor. In a similar way to [23], the technology-dependent sensitivity, raw FIT in this case, was measured on an L1 cache with neutron beam experiments and used in the model as representative of the Cortex-A9 implemented in Xilinx Zynq devices.

III. Mathematical Model and Reliability Figure

The key idea of our model is to modulate the intrinsic cross-section of the different processor blocks with the error-masking effect that is generated by the piece of code they are running. It is well known that any soft error induced on a bit can be masked, depending on its use by the software. Furthermore, the inclusion of some sort of code redundancy can improve the masking effect, such as those used in the SIHFT techniques [26]. However, when a piece of code is proposed for hardening, it is important to analyze all possible configurations, to assess whether the overhead that is introduced is worthwhile. Therefore, not only is it important to analyze the configuration parameters that affect program performance (e.g. data replication, consistence checks, error masking, resource optimizations, resource mapping, ...), but it is also important to analyze the radiation sensitivity of the solution.

In practical experiments, simulated fault-injection campaigns are used as low-cost solutions for fast checking the effects of these techniques, to identify the overhead that has been introduced and to estimate the application fault coverage. For instance, the relaxation of some constraints, such as permitting few overheads throughout the execution time of an application, can only be tolerated when a significant improvement in reliability or resource availability compensates the overhead incorporated in the system. However, costly irradiation campaigns are still necessary to ensure the correctness of the solution, because in most cases there is no clear correlation between simulation and irradiation results [27]. The main problem is the way in which real irradiation results can be accurately predicted with simulated fault injection campaigns on complex devices.

Modern microprocessors are usually composed of several blocks the contribution of which to final device reliability is somewhat uneven. The sensitivity of each block is dependent on its own technological implementation and may show slight differences from one block to another. In our approach, block sensitivity is a parameter of a Multiple Linear Regression model. Multiple block-sensitivity parameters are fitted by experimental results, which offer an estimation of the static cross-section, and fault injection results, which integrate the software masking effect. Once trained, the model helps the designer to estimate the reliability of each piece of code on a given processor via simple simulations. It also helps with speedy evaluation of the different trade-offs in simulation (away from radiation environments) and limits the real radiation measurements to the single (maybe, two or three) most suitable configuration(s) with enhanced reliability capabilities.

A. Microprocessor blocks decomposition

In a program, two major blocks can be relatively easily identified: storage and control. A storage block is composed of the *Register File* (**R**) and memory subsystem, which usually follows a specific resource mapping scheme that is dependant on the platform and can involve several resources with different technological implementations. General purpose compilers usually split the programs following a standard set of blocks, which are used to segment the code and to map them easily on a specific platform:

- *Program memory*, (**P**): where the program instructions are written and, depending on processor and compilation flags, the instructions can be executed from a Read-Only Memory (ROM) or a Random-Access Memory (RAM) (after initial loading from the ROM). The ROM itself can have several implementations, from true ROM (rare) to FLASH, which are often considered immune to SEUs, although the controller and the initialization and writing codes are not necessarily immune.
- *Data memory* (**D**): usually implemented in static or dynamic RAM, some processors use ferroelectric RAMs, which have, it is claimed, greater immunity to radiation effects [28], ie. MSP430.
- *Constants*: depending on the compiler and the optimization flags that are used, constants will either form part of the *code memory* (P) or will be stored in the *data RAM* (D) and initialized on program start-up. (Included on D or P depending on the compiler decisions).
- *Program stack*, (**S**): depending on the processor, the program stack can be part of the RAM data or can be separately implemented.

The *control block*, (**X**), is the processor part that processes instruction fetching, opcode decoding, internal timing, peripheral controls and interrupts. This part is seldom simulated or the behavior is approximated in high level instruction-accurate or cycle-accurate simulators. Therefore, the effects within the control block, which are often independent of the specific code such as the essential platform configuration bits, can be estimated as a constant parameter. In contrast, low-level simulators are capable of simulating this part, such as Register-transfer level (RTL) simulators. However, they are too slow for effective trade-offs within the optimization process.

B. Model Description

As made clear in the previous subsection, our model takes account of the following i blocks/sections ($i \in \{P, D, B_1, B_2, B_3, ..., S, R, X\}$) of the test processor. They must be separately considered, because they could have different hardware implementations, and they are used in different ways by different applications. They therefore have their own *cross section* (σ_i) [29]. Our model also follows a Multiple Linear Regression (MLR) [30] approach where the coefficients are σ_i and the intercept terms integrates the contribution of the non simulable blocks. A gradient descent algorithm [31] is used to fit the parameters, instead of the conventional *least squares* method. The gradient descent approach optimizes a target function through a systematic selection of values within a range that will either minimize or maximize the function result. In this sense, the estimated SEU events, i.e. *Expected Number of Single Events* (SDC for an erroneous output or HANG for non-response states), can be modeled as follows:

$$\begin{aligned}
\hat{P}(\mathtt{SD}) &= \Phi \cdot \mathtt{T} \cdot \sum_{\mathtt{i}} \sigma_{\mathtt{i}} \cdot \overline{\mathtt{SD}}_{\mathtt{i}} = && (1)\\
&= \Phi \cdot \sigma^{*} \cdot \mathtt{N}_{\mathtt{Exec}} \left\{ \mathtt{T}_{\mathtt{E}} \left(\sum_{\mathtt{i}} \mathtt{K}_{\mathtt{i}} \cdot \overline{\mathtt{SD}}_{\mathtt{i}} + \overline{\mathtt{CSD}}_{\mathtt{X}} \right) \right\} && (2)\\
\hat{P}(\mathtt{HG}) &= \Phi \cdot \mathtt{T} \cdot \sum_{\mathtt{i}} \sigma_{\mathtt{i}} \cdot \overline{\mathtt{HG}}_{\mathtt{i}} = && (3)\\
&= \Phi \cdot \sigma^{*} \cdot \mathtt{N}_{\mathtt{Exec}} \left\{ \mathtt{T}_{\mathtt{E}} \left(\sum_{\mathtt{i}} \mathtt{K}_{\mathtt{i}} \cdot \overline{\mathtt{HG}}_{\mathtt{i}} + \overline{\mathtt{CHG}}_{\mathtt{X}} \right) \right\} && (4)
\end{aligned}$$

where, Φ is the radiation flux ($particles/s/cm^2$); σ_i is the relative cross section per byte of each block, ($cm^2/byte$) expressed as $\sigma_i = \mathtt{K}_{\mathtt{i}} \cdot \sigma^*$; where, $\mathtt{K}_{\mathtt{i}}$ represents the block sensitivity that modulates a unique cross section, σ^*, shared by the whole platform. $\mathtt{T}$ is the total exposure to radiation flux, which is also expressed as ($\mathtt{T}_{\mathtt{E}} \cdot \mathtt{N}_{\mathtt{Exec}}$), where $\mathtt{T}_{\mathtt{E}}$ is the nominal execution time (s) and $\mathtt{N}_{\mathtt{Exec}}$ is the number of executions or runs performed during the radiation tests. $\overline{\mathtt{SD}}_i$ ($\overline{\mathtt{HG}}_i$) is the *average sensitivity* to SDC (HANG) of each block. $\overline{\mathtt{CSD}}_X$ ($\overline{\mathtt{CHG}}_X$) modulates the influence of the non-simulated part.

$\overline{\mathtt{SD}}_i$ ($\overline{\mathtt{HG}}_i$) are calculated by multiplying the frequency of a given event (τ_i) by the block size in bytes (S_i), taking into account that larger blocks are more prone to SEUs events. Simulated fault injection campaigns are used to obtain the frequency of a given event (τ_i), calculated as the number of events divided by the number of faults that are injected ($\mathtt{R_i}$) per block.

$$\overline{\mathtt{SD}}_\mathtt{i} \triangleq \tau_{\mathtt{SDC}} \cdot \mathtt{S_i} = \frac{\#\mathtt{SD_i}}{\#\mathtt{R_i}} \cdot \mathtt{S_i} \tag{5}$$

$$\overline{\mathtt{HG}}_\mathtt{i} \triangleq \tau_{\mathtt{HANG}} \cdot \mathtt{S_i} = \frac{\#\mathtt{HG_i}}{\#\mathtt{R_i}} \cdot \mathtt{S_i} \tag{6}$$

A relevant factor in the model is the bracketed expression that represents the *size-time figures*, χ_{SD} and χ_{HG}, of the given program, which can be used to compare the overheads of different solution strategies and to identify the blocks where the hardening efforts should be focused. This way, the equations can be rewritten as:

$$\frac{\hat{P}(\mathtt{SD})}{\mathtt{N_{Exec}}} = \Phi\cdot\sigma^*\cdot\chi_{SD}\,;\;\; \chi_{SD} = \mathtt{T_E}\cdot\left(\sum_i \mathtt{K_i}\cdot\overline{\mathtt{SD}}_i + \overline{\mathtt{CSD}}_\mathtt{X}\right) \tag{7}$$

$$\frac{\hat{P}(\mathtt{HG})}{\mathtt{N_{Exec}}} = \Phi\cdot\sigma^*\cdot\chi_{HG}\,;\;\; \chi_{HG} = \mathtt{T_E}\cdot\left(\sum_i \mathtt{K_i}\cdot\overline{\mathtt{HG}}_i + \overline{\mathtt{CHG}}_\mathtt{X}\right) \tag{8}$$

Our model proposes that the error probability depends on three independent factors:

- The radiation flux (Φ)
- The shared processor cross section (σ^*)
- The intrinsic configuration of the program (χ_{SD} and χ_{HG})

As a consequence, one can concentrate on guiding the hardening and optimization efforts by looking exclusively at the easy-to-obtain *Size-Time figures* (χ_{SD} and χ_{HG}).

Additionally, the model can be used to estimate the well known metrics *Mean Work To Failure* (MWTF), first introduced by Reis at [32], and *Mean Time To Failure* (MTTF). The MWTF metric captures the average work that an application can perform before an event is produced (*number of executions*). The MTTF metric captures the average time between two failures during a given sample time (T_{sample}).

$$\mathtt{MWTF} = \begin{cases} \frac{1}{\hat{P}(\mathtt{SD})/\mathtt{N_{Exec}}} & = \frac{1}{\Phi\cdot\sigma^*\cdot\chi_{SD}} \quad \text{for SDC} \\ \frac{1}{\hat{P}(\mathtt{HG})/\mathtt{N_{Exec}}} & = \frac{1}{\Phi\cdot\sigma^*\cdot\chi_{HG}} \quad \text{for HANG} \end{cases} \tag{9}$$

$$\mathtt{MTTF} = T_{sample} \cdot \mathtt{MWTF} \tag{10}$$

IV. Experimental Setup

Two devices (MSP430 and ARM) were evaluated in two different radiation test facilities, using several benchmarks, in order to assess the model described in the previous section. They were classified into three groups according to whether either optimization or hardening techniques or both were applied.

A. DUTs - Devices Under Test - ARM & MSP430

Two Devices Under Test were proposed for the irradiation experiment: an ARM microprocessor and a MSP430 micro-controller.

The first DUT is a ZYBO board, equipped with a 28nm CMOS *Xilinx ZYNQ XC7Z010* System On Chip (SoC) [33]. This SoC is divided into two parts, an FPGA area (Programmable Logic – PL) and the dual-core 666MHz 32-bit ARM cortex A9 microprocessor (Processing System – PS). This processor has a 13-stage instruction pipeline that includes a branch prediction block and support for two levels of cache. The processor also has a built-in On Chip Memory (OCM), where a bootloader or the program under test can be loaded. The simulated fault injected campaigns of this device were performed over a *OVPsim* simulator from *Imperas*, an instruction-accurate processor, which can be extended using custom plugins to perform non-intrusive injections [21].

The second DUT is the MSP430F5529 from Texas Instrument [34], a 130nm micro-controller widely used in micro-satellites (CubeSats). This device is a 16-bit RISC CPU running at 25MHz, equipped with 128KiB of flash memory and 8KiB of RAM. The register file is composed of 5 control registers and 11 general purpose registers. The simulated fault injection campaigns of this device were performed on the NAKEN simulator, which also was extended to provide non-intrusive fault-injection capabilities [35].

Software Simulators can be simply altered to provide non-intrusive fault injection capabilities based on the bit-flip model. Therefore, they can be used to obtain the fault coverage statistics, in terms of Silent Data Corruption (SDC), when an erroneous output is produced, and in terms of HANG when the platform stops working [29]. In particular, simulators such as [22], [36], [37] can estimate, given a program and its configuration, both the SDC and the HANG values. In both cases, the simulated fault injection was separately performed on each register and the memory was also injected, taking into account the program data, the program code and the stack section.

During the radiation test, both DUTs were configured, to send a signal to an external computer every 5 seconds reporting the internal status in the absence of errors. Any discrepancy between the calculated and the golden results for the DUT will signal an SDC error, which is immediately communicated to the external computer. If there is no communication within 10 seconds, the DUT is considered to have reached the HANG state and it is restored by the external computer.

B. Facilities - LANSCE & CNA

Two radiation test campaigns were performed, to develop the model and for its validation.

The first test campaign was performed in mid-2018 at the National Centre for Accelerators, in Spain [38]. Irradiation tests were performed using the external beam line, installed in the cyclotron laboratory. The compact system emits a unique proton beam (18 MeV), without any degrader films. The beam is emitted through the window, a 125μm Mylar® foil, and the air distance up to the DUT (device under radiation) position.

In this particular case, the DUT was placed at 53.5 cm from the exit nozzle, so that the final energy at the surface was 15.2 MeV, with an estimated spread in the order of 300 KeV. The device package was not removed, as the energy range of incident protons in the silicon active area (8 to 10 MeV) was sufficient to produce events, which has previously been validated by the CNA group in similar campaigns conducted with these types of boards [7]. The final energy of the incident beam on the surface and in the active area was obtained with energy-loss data calculated with the SRIM2013 code [39]. A medium flux value was calculated at the base of the pulses, registered by a counter on the 10 pA sensitive scale. The flux value was monitored within a 5% fluctuation during each run. Finally, the fluence at the DUT was calculated using the exposure time for each run (10^9) to an accuracy of 10%. The radiation field, defined by a mask on the device, covered the active area with a uniformity that was higher than 90%.

The second test campaign, the neutron SEE campaign, was performed at the Los Alamos Neutron Science Center (LANSCE) in September 2018 [40], [41]. The experiments were performed at the Weapons Neutron Research Facility (WNR), using Target 4 Flight Path 30L (ICE I). The neutron beam was produced from a tungsten spallation source at approximately 30 degrees to the left of the main beam. The shape of the neutron spectrum was very similar to the one produced in the atmosphere by cosmic rays. During the campaign, the DUT remained at 23 m from the neutron source, and the beam was collimated, so that a spot with a diameter in the order of 30 mm was obtained. This size covers the active area with a higher uniformity than 90%. The LANSCE dosimetry data yielded a constant neutron flux of $1.7 \cdot 10^5 n/(s \cdot cm^2)$, above 10MeV. Taking into account the times to complete each run, the total fluence was calculated with an accuracy of 10%.

C. Benchmarks

During the experiment, several programs from ”Benchmarks for Energy Measurements on Embedded Platforms” (BeeBs) [42] were hardened and evaluated using simulated fault injections. All the benchmarks were evaluated, by injecting 1000 faults per resource (registers and memory sections) that constituted the platform under simulation. For the sake of simplicity, only the most significant program versions were proposed for radiation testing at LANSCE and CNA:

- **Bubblesort (BB):** A sorting algorithm that swaps contiguous elements of a vector until the vector is shortened.
- **Dijkstra (DK):** A shortest path finder between two points, given an adjacency matrix.
- **NDES (ND):** A block encryption algorithm with which the ”key” matrix must be preserved, in order to cypher and to decypher the messages.

All the aforementioned benchmarks were hardened using several strategies, based on SIHFT techniques applied at different levels from the assembly level to sophisticated structures present in a general purpose programming language such as *C++*.

- **MOOGA** (**M**) [21]: a *genetic algorithm* guided by a *multi-objective optimization algorithm* was used to optimize an application by means of the compiler flags, which is able to generate a huge number of equivalent applications with improved features such as size, time, and fault coverage altogether.
- **SHE** (**S**) [43] *Software Hardening Environment*: assembly to assembly compiler capable of adding automatic protection to the MSP430 designs, by using the S-SWIFT-R technique. The compiler has the capability of producing selective register hardening, in order to reduce the time overheads of replica computation.
- **HData** (**H**) [17] - *Hardened Data*: classes templates on *C++* that replace the basic language types, in order to implement automated and transparent TMR protection.

Benchmarks were coded using the following name convention: first the benchmark name, then the hardening technique applied, and finally the version number. For instance, BB-M-10 corresponds to the 10th version of Bubblesort algorithm hardened using MOOGA. In addition, those versions matching with the standard optimization flags are indicated using parentheses (e.g. BB-M-3 (O3)).

V. Model Training and Evaluation

All the benchmarks were evaluated and characterized prior to the radiation experiment, to analyze the impact of hardening techniques on program reliability. Once the measures had been obtained following radiation, the model was adjusted to set the sensitivity parameters. Finally, the model was evaluated against a control group of versions not used in the training phase.

A. Programs evaluation ($\overline{\mathtt{SD}}_\mathtt{i}$ and $\overline{\mathtt{HG}}_\mathtt{i}$)

Table I summarizes the average sensitivity of the different blocks present in the algorithms selected for radiation. The table classifies them by the DUTs under evaluation and the exposed particles.

The LANSCE ARM-n^0 applications showed that the reference version of the Dijkstra algorithm (DK-M) achieved better results in all blocks with the exception of the $\overline{\mathtt{SD}}_\mathtt{D}$ (RAM), which was improved 25× by DK-H. On the contrary, Bubblesort offered improvements in most of the blocks under analysis, with the exception of $\overline{\mathtt{HG}}_\mathtt{S}$ (stack) and $\overline{\mathtt{HG}}_\mathtt{P}$ (Program) where the hardening technique had an adverse influence, due to block-size growth.

The CNA ARM-p^+ and the *ARM-p^+ applications provided a clear view of the best version of the program (11 and 14), which corresponded with the hardened version of the Bubblesort algorithm. This version achieved the best results in almost all block and SEU events, which implies a program of greater reliability. In contrast, the NDES algorithm provided no clearly reliable version, due to the strengths and weaknesses of each version.

No clear indicator was found to ascertain which version among the LANSCE MSP430-n^0 applications offered major reliability enhancements, however the effects of each hardening technique were visible. For instance, SHE builds are focused on enhancing the register file reliability, an objective achieved by both. As the HData technique is focused on data

TABLE I
SIMULATED $\overline{\mathtt{SD}}_{\mathtt{i}}$ AND $\overline{\mathtt{HG}}_{\mathtt{i}}$ VALUES FOR ALL BENCHMARKS IRRADIATED. BOLD BENCHMARK NAMES ARE USED AS REFERENCE BUILD FOR COMPARATIVE PURPOSES. LOWER VALUES ARE BETTER (HIGHLIGHTED).

ARM-n^0	Register		RAM		STACK		Program	
Training	$\overline{\mathtt{SD}}_{\mathtt{R}}$	$\overline{\mathtt{HG}}_{\mathtt{R}}$	$\overline{\mathtt{SD}}_{\mathtt{D}}$	$\overline{\mathtt{HG}}_{\mathtt{D}}$	$\overline{\mathtt{SD}}_{\mathtt{S}}$	$\overline{\mathtt{HG}}_{\mathtt{S}}$	$\overline{\mathtt{SD}}_{\mathtt{P}}$	$\overline{\mathtt{HG}}_{\mathtt{P}}$
BB-M-16	2.1	12.1	896	14.1	4.9	**5.9**	0	**318**
BB-H-1	**0.2**	**9.1**	**3.8**	**7.3**	**0**	17.7	0	373
DK-M-17	**3.6**	**9.9**	401	**339**	**11.8**	**18.7**	**128**	**1282**
DK-H-2	6.3	19.1	**16.0**	540.6	13.9	21.2	244	2929
*ARM-p^+	Register		RAM		STACK		Program	
Control	$\overline{\mathtt{SD}}_{\mathtt{R}}$	$\overline{\mathtt{HG}}_{\mathtt{R}}$	$\overline{\mathtt{SD}}_{\mathtt{D}}$	$\overline{\mathtt{HG}}_{\mathtt{D}}$	$\overline{\mathtt{SD}}_{\mathtt{S}}$	$\overline{\mathtt{HG}}_{\mathtt{S}}$	$\overline{\mathtt{SD}}_{\mathtt{P}}$	$\overline{\mathtt{HG}}_{\mathtt{P}}$
BB-M-10	2.6	15.6	397	14.2	0	5.4	56.8	375
BB-H-11	**0.24**	**8.8**	**4.1**	**8.0**	0	11.1	**0**	504
BB-M-12	22.2	12.9	401	13.8	0	5.1	263	323
BB-M-13	1.6	11.3	270	151	1.7	6.0	115	435
BB-H-14	**0.2**	**8.8**	**4.1**	**8.0**	0	11.1	**0**	504
BB-M-15	1.6	11.3	270	152	1.7	6.0	115	435
ARM-p^+	Register		RAM		STACK		Program	
Training	$\overline{\mathtt{SD}}_{\mathtt{R}}$	$\overline{\mathtt{HG}}_{\mathtt{R}}$	$\overline{\mathtt{SD}}_{\mathtt{D}}$	$\overline{\mathtt{HG}}_{\mathtt{D}}$	$\overline{\mathtt{SD}}_{\mathtt{S}}$	$\overline{\mathtt{HG}}_{\mathtt{S}}$	$\overline{\mathtt{SD}}_{\mathtt{P}}$	$\overline{\mathtt{HG}}_{\mathtt{P}}$
BB-M-1	22.2	12.9	400	13.8	0	5.12	263	323
BB-M-2	1.5	12.6	256	144	3.4	**3.13**	95.4	371
BB-M-3	2.3	16.2	395	24.8	0	4.27	335	345
BB-M-4	1.6	11.3	270	151.6	1.7	5.97	115	435
BB-M-5	1.8	15.6	377	14.3	0	5.97	61.7	**288**
ND-M-6	**0.7**	13.9	**31.9**	168	**6.8**	29.9	1295	**534**
ND-M-7	26.2	**9.8**	186	**0.8**	20.5	**12.5**	2891	702
ND-M-8	0.8	14.0	37.3	176	**6.8**	29.0	**1202**	641
ND-M-9	0.8	14.0	37.3	176	**6.8**	29.0	**1202**	641
MSP430-n^0	Register		RAM		STACK		Program	
Training	$\overline{\mathtt{SD}}_{\mathtt{R}}$	$\overline{\mathtt{HG}}_{\mathtt{R}}$	$\overline{\mathtt{SD}}_{\mathtt{D}}$	$\overline{\mathtt{HG}}_{\mathtt{D}}$	$\overline{\mathtt{SD}}_{\mathtt{S}}$	$\overline{\mathtt{HG}}_{\mathtt{S}}$	$\overline{\mathtt{SD}}_{\mathtt{P}}$	$\overline{\mathtt{HG}}_{\mathtt{P}}$
BB-M-18	5.9	4.5	99.1	0	**0**	**1.2**	256	**0**
BB-S-5	**0**	**3.1**	**0**	0	**0**	1.3	**0**	1.4
BB-H-3	0.3	5.0	**0**	0	0.2	3.9	59.4	204
DK-M-19	3.7	7.2	158	**44.8**	**1.2**	6.0	**61.03**	**261**
DK-S-4	**0.1**	**3.1**	177	61.6	2.9	3.3	110.5	413
DK-H-4	1.0	3.9	**0**	116	2.1	**3.1**	671	1208

triplication, $\overline{\mathtt{SD}}_{\mathtt{D}}$ showed improvements in both cases. However, both techniques introduce new vulnerabilities into the program code, which may not be taken into consideration, because the program code block is immune to radiation, as will be explained in connection with the model.

B. Model Training and Validation

A limited set of applications for irradiation was proposed for training the model, due to time constraints on access to the facilities and fluxes. The model coefficients were obtained using the gradient descent method to adjust equations 7 and 8. The error function (equation 11) was used to optimize the model, during the gradient descent training process. ϵ needs to be minimized to obtain values close or equal to 1, which meant that the model estimations exactly matched the real irradiation measurements. The equation penalizes the higher differences between the expected number of faults provided by the model ($\hat{P}(\mathtt{Event})$) and the real irradiation measurements ($P(\mathtt{Event})$).

$$\epsilon = 10^{\mathtt{mean}\left(\left|\log_{10} P(\mathtt{SD})-\log_{10} \hat{P}(\mathtt{SD})\right|^3+\left|\log_{10} P(\mathtt{HG})-\log_{10} \hat{P}(\mathtt{HG})\right|^3\right)} \tag{11}$$

Table II shows the different $\mathtt{K_i}$ sensitivity parameters per block and the common shared processor cross section (σ^*) obtained from the training phase. Also, an analysis of the results is presented in Table II by measuring the goodness of the correlation between both the real and the predicted events for the different DUTs and the particles: i.e. the respective correlations using a Simple Linear Regression model (being *m* the slope of the regression line and *Std* the standard error) and Pearson's coefficient (R^2).

TABLE II
TRAINED MODEL SENSITIVITY PARAMETERS AND CORRELATION METRICS FOR DIFFERENT DUTS AND PARTICLES.

	$\sigma^*(cm^2/byte)$	$\mathtt{K_R}$	$\mathtt{K_D}$	$\mathtt{K_S}$	$\mathtt{K_P}$	$\overline{\mathtt{CSD}}_{\mathtt{X}}$	$\overline{\mathtt{CHG}}_{\mathtt{X}}$	m	*Std*	R^2
ARM-n^0	$6.66{\cdot}10^{-14}$	23	1	1	1	44	0	1.04	0.06	0.91
ARM-p^+	$2.27{\cdot}10^{-14}$	23	1	1	1	44	0	0.87	0.08	0.91
MSP430-n^0	$7.90{\cdot}10^{-13}$	5	1	1	0	4	0	1.32	0.12	0.86

TABLE III
CORRELATION MODEL RESULTS FROM CONTROL GROUP

	m	*Std*	R^2
*ARM-p^+	0.88	0.07	0.89

An overview of the trained sensitivity parameters showed that all memory blocks sharing the same technology also shared the same $\mathtt{K_i}$ (see RAM $\mathtt{K_D}$ and stack $\mathtt{K_S}$). The model highlighted that the MSP430 Program code block was technologically protected (flash chip) and therefore immune to radiation, although ARM may affect the effectiveness of the SIHFT techniques whenever there are code overheads. The control block, (X), was less prone to errors in MSP430 than in ARM, because the logic complexity is higher in ARM than in the MSP430 microcontroller. The $\mathtt{K_R}$ associated with the register file showed similar sensibility to the memory ($\mathtt{K_D}$) in the MSP430, which meant that both blocks shared similar technology. In contrast, the higher sensitivity of the ARM device was due to the higher complexity of the register file subsystem.

The correlation analysis revealed that the model for ARM presented the highest correlation, with a Pearson's coefficient of 0.91, while the MSP430 R^2 coefficient was 0.86. Focusing on the case of ARM-n^0, the line that best fitted the relation between the real measures and the predictions ($m = 1.04$) showed that the model predictions perfectly fitted the measurements with a reduced *Std* of 0.06. In contrast, the models for ARM-p^+ and for MSP430-n^0 showed under- and over-estimation, respectively, defined by slopes of 0.87 and 1.32 with *Std* of 0.08 and 0.12 respectively.

TABLE IV

MEASURES AND PREDICTIONS UNDER NEUTRON AND PROTON IRRADIATION. $P(\texttt{Events})$ IS THE NUMBER OF EVENTS MEASURED (95% OF CONFIDENCE INTERVALS ARE SHOWN AS SUBSCRIPT AND SUPERSCRIPT). $\hat{P}(\texttt{Events})$ IS THE NUMBER OF EVENTS ESTIMATED BY OUR MODEL. $\texttt{T(s)}$ IS THE TOTAL RADIATION TIME IN SECONDS.

ARM-n^0	T	Measured		Predicted	
	(10^5s)	P(SD)	P(HG)	$\hat{P}$(SD)	$\hat{P}$(HG)
BB-M-16	3.97	12_{6}^{21}	4_{1}^{10}	15	9
BB-H-1	4.00	1_{0}^{6}	8_{3}^{16}	1	9
DK-M-17	4.09	18_{11}^{29}	36_{25}^{51}	10	29
DK-H-2	4.11	13_{7}^{23}	42_{30}^{58}	7	62

ARM-p^+	T	Measured		Predicted	
Training	(10^5s)	P(SD)	P(HG)	$\hat{P}$(SD)	$\hat{P}$(HG)
BB-M-1	0.43	112_{85}^{139}	14_{8}^{24}	90	47
BB-M-2	0.66	76_{58}^{97}	40_{28}^{56}	46	86
BB-M-3	0.38	79_{61}^{101}	36_{25}^{51}	56	50
BB-M-4	0.24	69_{52}^{89}	45_{32}^{61}	24	43
BB-M-5	0.31	65_{49}^{85}	38_{26}^{53}	38	48
ND-M-6	0.16	72_{55}^{93}	35_{24}^{50}	53	40
ND-M-7	0.81	91_{71}^{115}	15_{8}^{25}	74	19
ND-M-8	0.22	80_{62}^{102}	33_{22}^{47}	61	54
ND-M-9	0.16	82_{63}^{114}	29_{19}^{42}	45	40

*ARM-p^+	T	Measured		Predicted	
Control	(10^5s)	P(SD)	P(HG)	$\hat{P}$(SD)	$\hat{P}$(HG)
BB-M-10	0.79	15_{8}^{25}	51_{37}^{68}	20	74
BB-H-11	0.54	3_{1}^{9}	99_{75}^{124}	13	107
BB-M-12	0.30	32_{21}^{46}	69_{51}^{88}	89	50
BB-M-13	0.19	9_{4}^{17}	44_{30}^{59}	9	29
BB-H-14	0.18	1_{0}^{6}	88_{57}^{96}	5	79
BB-M-15	0.22	5_{2}^{12}	45_{32}^{61}	11	36

MSP430-n^0	T	Measured		Predicted	
	(10^5s)	P(SD)	P(HG)	$\hat{P}$(SD)	$\hat{P}$(HG)
BB-M-18	3.05	27_{17}^{40}	5_{2}^{12}	26	5
BB-S-5	22.2	1_{0}^{6}	7_{3}^{15}	1	3
BB-H-3	46.2	1_{0}^{6}	2_{0}^{7}	1	6
DK-M-19	35.5	43_{30}^{59}	16_{9}^{26}	34	16
DK-S-4	106	81_{62}^{103}	9_{4}^{17}	35	15
DK-H-4	269	2_{0}^{7}	15_{8}^{25}	2	27

Several irradiated applications were used as a control group (*ARM-p^+), to verify the model that was trained for the ARM proton campaign (ARM-p^+). The control group (applications excluded from the training set) was formed with several applications with a deactivated cache and the RAM block forced outside the beam, to highlight the relevance of the register file. Within this control group, two new applications were found, one of which was hardened with the HData technique, which presented 2 versions (11 and 14), which were configured so that the former was outside and the latter inside the radiation flux respectively. Table III shows the regression analysis performed over the control group, using the parameters obtained after training the model. This analysis revealed a high model consistency, because the trend line slope was preserved (0.88 ± 0.07) and the R^2 showed little variation (0.89).

Finally in Table IV, the expected values of the model are shown alongside the real irradiation measurements and confidence intervals calculated using [16]. Also, Figure 1 shows the model predictions for the versions of Bubblesort used as control group and the measurements obtained from the radiation campaign. As can be seen, most of the predictions are within the confidence intervals. The model tends to overestimate the SDC events slightly, with the only notable exception of BB-M-12, which exceeds this behavior due possibly to an inaccuracy in the radiation experiments. In addition, to a certain extent the model overestimation of SDC is compensated with the underestimation of the HANG number. The program versions are arranged in order of increasing number of events, in this way, it can be observed that the predictions follow the same trend as the radiation results. Therefore, the model can identify the most sensitive versions (BB-M-12 to SDC and BB-H-11 to HANG) and the less sensitive versions (BB-H-14 to SDC and BB-M-13 to HANG).

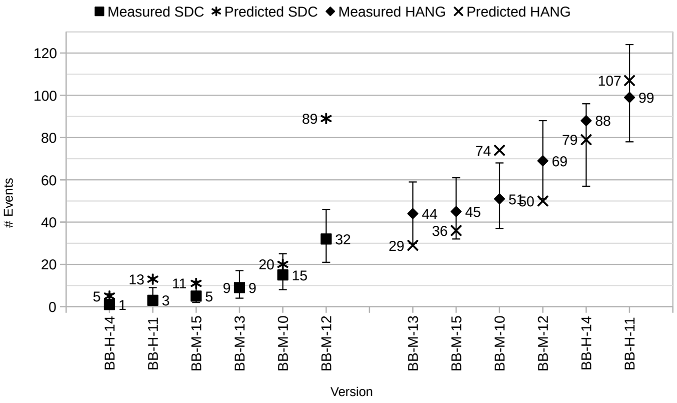

Fig. 1. SDC and HANG events for the applications of the control group. The figure shows the experimental measures obtained with 95% confidence interval and the number of events predicted by the model.

VI. Early Program Sections Sensitivity Characterization

The operation of the model for early reliability characterization of applications and hardening techniques is presented in this section. First, the parameters obtained during the model training are used to evaluate the contribution of the different processor blocks to the program reliability. Then, the *size-time figures* (χ_{SD} and χ_{HG}) are calculated taking into account not only the sensitive area but also the program exposed time under the different configurations. Finally, the MWTF and MTTF metrics, obtained from the *size-time figures*, help to estimate the real impact of different hardening techniques under real irradiation.

For the sake of simplicity, only some Bubblesort versions are discussed for both architectures (ARM and MSP430).

1) Block contribution to the Size-time figures: the processor blocks sensitivity to SDC and HANG jointly with the execution times of each version are shown in Figures 2 and 3. For comparison purposes all numbers are normalized to a baseline version compiled with -O0 flag (without any optimization). The evaluation of each build and optimization can be performed quite reasonably, by taking account of each technique and the time overhead that it introduces.

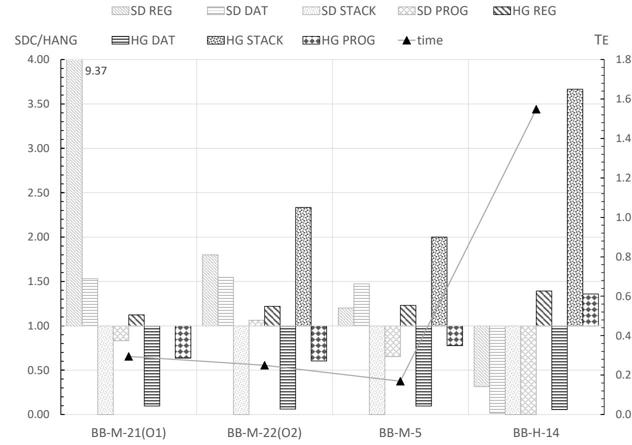

Fig. 2. Normalized execution time (triangles) and contribution to the size-time figures (SDC: gray bars; HANG: dark bars) of the ARM processor blocks. Baseline version -O0.

Related to ARM processor (figure 2), only MOOGA optimization and HData technique were applied to the benchmarks. It can be observed that MOOGA achieved a meaningful reduction in execution time up to 5× (0.18×), because of aggressive performance optimizations, while HData produced an increase by around 2×, due to the extra calculations needed to perform the TMR.

Regarding the individual contribution of the processor's block, the Register file and the Data section achieved an increase at the SDC in the optimized versions obtained by the MOOGA. Due to the extensive use of those resources, the increase in their susceptibility to failure was very relevant in the case BB-M-21, which increased by almost 10×. In contrast, it was the HData version BB-H-14 that managed to achieve an important reduction because of the registers and variables replication. Both techniques achieved reductions in the Program block, especially in the HData version BB-H-14 where is reduced to 0.

Finally, the Stack block was, in all cases (including the baseline) of little interest from the point of view of SDC, because of its negligible susceptibility to this kind of faults. However, it is remarkable the rise in its HANG sensitivity that increased by 2.3× with MOOGA (B-M-22) and up to 3.7× with HData (BB-H-14). The Register file contribution to HANG was also, but to a lesser extent, negatively affected by the hardening techniques. For instance, BB-M-5 presents a sensitivity 1.23× greater than the baseline and 1.4× the BB-H-14 version. On the contrary, the Data section susceptibility to HANG was minimized to almost zero in all the versions, and also the contribution of the Program block is reduced, by MOOGA optimization, between 0.60× (BB-M-22) and 0.78× (BB-M-5). Only HData version increased this figure by 1.36×.

In summary MOOGA technique reduces the exposure time of the application but increases the sensitivity to SDC of the Register file and Data and the sensitivity to HANG of the Stack. In addition, MOOGA achieves an almost full mitigation of HANG events in the Data memory and a minor reduction on Program memory. On the other hand, HData produces a high time overhead but achieves an efficient protection of all the blocks against the SDC events. Its main drawback is the increase of HANG susceptibility, mainly of the Stack section.

In a similar way, Figure 3 shows the normalized values of some representative versions hardened by MOOGA, HData and SHE techniques. As evidenced by the model parameters, the Program section that is implemented in FLASH memory was unaffected by radiation. Therefore its contribution, zero in all the versions including the baseline, is not shown in the figure.

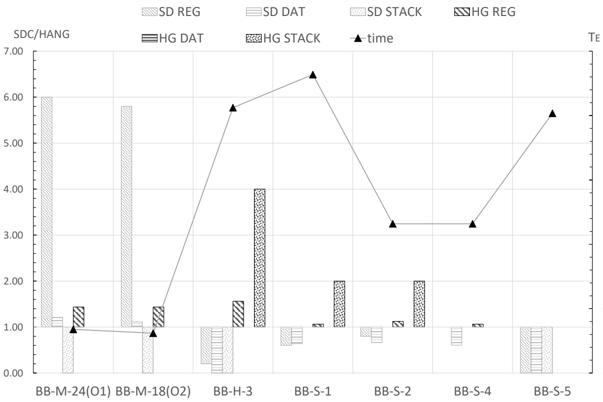

Fig. 3. Normalized execution time (triangles) and contribution to the size-time figures (SDC: gray bars; HANG: dark bars) of the MSP430 processor blocks. Baseline version -O0.

As can be seen, versions optimized with MOOGA (BB-M-24 and BB-M18) present the same behavior described for the corresponding versions in the ARM processor. In this case, the contribution of the Data section to the HANG sensitivity is negligible and remains stable along all the versions. The HData version (BB-H-3) also follows the same trend than previously, and corroborates the efficiency of the technique for mitigating SDC faults. The SHE technique was applied selectively to the registers of the Register file. It produced versions with different time overheads, from 1.2× (BB-S-4) to 2.32× (BB-S-1), depending on the level of use of the protected resources. As the figure shows, the technique improves the mitigation of SDC faults, mainly by reducing the contribution of the Register file and the Data section, but increasing the sensitivity of the Stack to HANG faults by 2.0× (BB-S-1 and BB-S-2). Selecting carefully the registers by their vulnerability, the protection can be improved and extended to the Stack block without worsening the HANG sensitivity (see the BB-S-5 version).

2) χ_{SD}/χ_{HG} and MWTF/MTTF performance metrics: Despite the good estimations that were offered by both SDC and HANG on each technique and its impact on the fault coverage of each test application, the execution time was not taken into account. The χ_{SD}/χ_{HG} metrics shown in Figures 4 and 5 captured this impact and evaluated whether the execution time overhead of the technique was worthwhile.

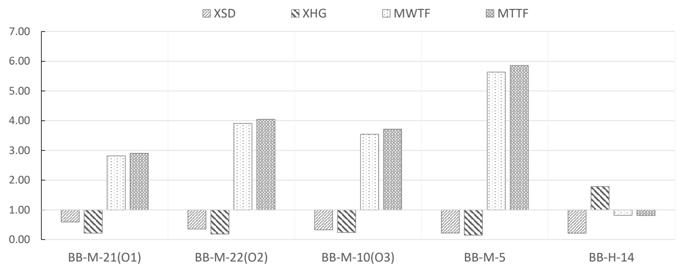

Fig. 4. Normalized χ_{Event} figures, MWTF and MTTF metrics for ARM calculated using the model parameters for protons ($\sigma^* = 2.27 \cdot 10^{-14} cm^2/byte$), a radiation flux of $10^9 p/cm^2 \cdot s$; and a sample time of $20ms$. Baseline version -O0.

As can be seen in Figure 4, the reference version (O0) showed the worst results in terms of χ_{SD} and χ_{HG}, with the exception of the BB-H-14 version, which achieved the highest rates of χ_{HG}. Focusing on this HData version, even though it was a build with better fault coverage for SDC, the execution time and the Program memory size overheads were decisive in the lower rate of MWTF and MTTF. MOOGA version BB-M-5, in contrast, achieved the highest results, even presenting an important HANG overhead in Stack and Data sections, however the reduced execution time was decisive in the amount of work it was capable of completing before a SEU event was produced.

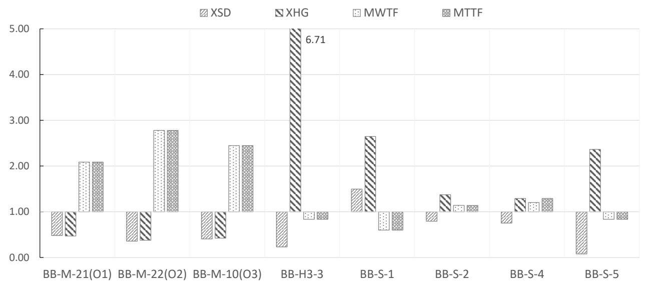

Fig. 5. Normalized χ_{Event} figures, MWTF and MTTF metrics for MSP430 calculated using the model parameters for protons ($\sigma^* = 2.27 \cdot 10^{-14} cm^2/byte$), a radiation flux of $10^9 p/cm^2 \cdot s$; and a sample time of $20ms$. Baseline version -O0.

Figure 5 shows the expected results for MSP430 under a neutron campaign.

It shows how the hardening techniques achieved the most similar results to the reference version (O0) and, in several cases, they showed a worsening. The overhead time was decisive for achieving comparable improvements to the optimized applications, which could process 3× more information before an SEU was detected.

VII. Conclusion

In this study, an empirical model has been presented that is capable of predicting the effects of radiation on different DUTs under real irradiation campaigns (protons and neutron). The model has been trained and tested with several applications of interest under different irradiation campaigns at the CNA and the LANSCE facilities. The results showed a good correlation between the predictions and the real results in both cases, the training and the control group of versions.

A case of use has also been presented, where the sensitivity of different program blocks has been estimated for several programs of interest. Likewise, several hardening proposals and optimized applications have been evaluated. The evaluation results have not only emphasized the importance of fault coverage obtained from simulated fault injections for increasing the reliability of an application, but also the importance of reducing the sizes and the performance overheads introduced by these techniques.

Acknowledgment

This work has been supported in part by the projects *ESP-2015-68245-C4-3-P* and *ESP-2015-68245-C4-4-P (Spanish MINECO)*. The authors would like to convey their gratitude to Los Alamos Neutron Science Center, and especially to its staff for their support during the test campaign.

References

[1] J. M. Benedetto, P. H. Eaton, D. G. Mavis, M. Gadlage, and T. Turflinger, "Digital single event transient trends with technology node scaling," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3462–3465, Dec. 2006.

[2] D. M. Fleetwood, "Evolution of total ionizing dose effects in MOS devices with Moore's law scaling," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1465–1481, Aug. 2018.

[3] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[4] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in CMOS processes," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. 128–143, Apr. 2004.

[5] L. A. C. Benites, F. Benevenuti, A. B. De Oliveira, F. L. Kastensmidt, N. Added, V. A. P. Aguiar, N. H. Medina, and M. A. Guazzelli, "Reliability Calculation With Respect to Functional Failures Induced by Radiation in TMR Arm Cortex-M0 Soft-Core Embedded Into SRAM-Based FPGA," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1433–1440, Jul. 2019.

[6] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A triple core lock-step (TCLS) ARM® cortex®-r5 processor for safety-critical and ultra-reliable applications," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE, Jun. 2016, pp. 246–249.

[7] A. Lindoso, M. García-Valderas, L. Entrena, Y. Morilla, and P. Martín-Holgado, "Evaluation of the suitability of neon simd microprocessor extensions under proton irradiation," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1835–1842, Aug. 2018.

[8] J. A. Blome, S. Gupta, S. Feng, and S. Mahlke, "Cost-efficient soft error protection for embedded microprocessors," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES '06. New York, NY, USA: ACM, 2006, pp. 421–431.

[9] P. Bernardi, L. B. Poehls, M. Grosso, and M. S. Reorda, "A hybrid approach for detection and correction of transient faults in SoCs," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 439–445, Oct. 2010.

[10] A. Martínez-Álvarez, F. Restrepo-Calle, S. Cuenca-Asensi, L. M. Reyneri, A. Lindoso, and L. Entrena, "A Hardware-Software Approach for On-Line Soft Error Mitigation in Interrupt-Driven Applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 502–508, Jul. 2016.

[11] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, M. García-Valderas, L. Entrena, A. Martínez-Álvarez, and S. Cuenca-Asensi, "Dual-core lockstep enhanced with redundant multithread support and control-flow error detection," *Microelectronics Reliability*, vol. 100-101, no. 113447, pp. 1–5, Sep. 2019.

[12] M. Demertzi, M. Annavaram, and M. Hall, “Analyzing the effects of compiler optimizations on application reliability,” in *2011 IEEE International Symposium on Workload Characterization (IISWC)*, Nov. 2011, pp. 184–193.

[13] P. Yuste, J. C. Ruiz, L. Lemus, and P. Gil, “Non-intrusive software-implemented fault injection in embedded systems,” in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 23–38.

[14] J. Frtunikj, J. Fröhlich, T. Rohlfs, and A. Knoll, “Qualitative evaluation of fault hypotheses with non-intrusive fault injection,” in *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Nov. 2015, pp. 160–167.

[15] D. Ferraretto and G. Pravadelli, “Simulation-based fault injection with QEMU for speeding-up dependability analysis of embedded software,” *Journal of Electronic Testing*, vol. 32, no. 1, pp. 43–57, Jan. 2016.

[16] European Space Agency, ESA, “ESCC Basic Specification No. 25100: Single event effects test method and guidelines, Issue 2,” *ESA, Noordwijk, Netherlands*, Oct. 2014.

[17] L. M. Reyneri, A. Serrano-Cases, Y. Morilla, S. Cuenca-Asensi, and A. Martínez-Álvarez, “A Compact Model to Evaluate the Effects of High Level C++ Code Hardening in Radiation Environments,” *Electronics*, vol. 8, no. 6–653, pp. 1–13, Jun. 2019.

[18] A. Vishnu, H. V. Dam, N. R. Tallent, D. J. Kerbyson, and A. Hoisie, “Fault modeling of extreme scale applications using machine learning,” in *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, 2016, pp. 222–231.

[19] R. A. Ashraf, R. Gioiosa, G. Kestor, R. F. DeMara, C. Cher, and P. Bose, “Understanding the propagation of transient errors in hpc applications,” in *SC ’15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2015, pp. 880–891.

[20] N. Narayanamurthy, K. Pattabiraman, and M. Ripeanu, “Finding resilience-friendly compiler optimizations using meta-heuristic search techniques,” in *2016 12th European Dependable Computing Conference (EDCC)*, Sep. 2016, pp. 1–12.

[21] A. Serrano-Cases, Y. Morilla, P. Martin-Holgado, S. Cuenca-Asensi, and A. Martinez-Alvarez, “Nonintrusive automatic compiler-guided reliability improvement of embedded applications under proton irradiation,” *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1500–1509, Jul. 2019.

[22] F. R. da Rosa, R. Garibotti, L. Ost, and R. Reis, “Using machine learning techniques to evaluate multicore soft error reliability,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2151–2164, Jun. 2019.

[23] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodriguez, and J. R. Mingo, “Estimating error rates in processor-based architectures,” *IEEE Transactions on Nuclear Science*, vol. 48, no. 5, pp. 1680–1687, Oct. 2001.

[24] V. Vargas, P. Ramos, V. Ray, C. Jalier, R. Stevens, B. Dupont De Dinechin, M. Baylac, F. Villa, S. Rey, N. Zergainoh, J. Méhaut, and R. Velazco, “Radiation experiments on a 28 nm single-chip many-core processor and seu error-rate prediction,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 483–490, Jan. 2017.

[25] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, “Demystifying soft error assessment strategies on arm cpus: Microarchitectural fault injection vs. neutron beam experiments,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2019, pp. 26–38.

[26] N. Oh, P. Shirvani, and E. McCluskey, “Error detection by duplicated instructions in super-scalar processors,” *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 63–75, Mar. 2002.

[27] F. M. Lins, L. A. Tambara, F. L. Kastensmidt, and P. Rech, “Register file criticality and compiler optimization effects on embedded microprocessor reliability,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 8, pp. 2179–2187, Aug. 2017.

[28] S. Gerardin and A. Paccagnella, “Present and future non-volatile memories for space,” *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3016–3039, Dec. 2010.

[29] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor,” in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, Dec. 2003, pp. 29–40.

[30] J. D. Jobson, *Multiple Linear Regression*. New York, NY: Springer New York, 1991, pp. 219–398.

[31] J. A. Snyman and D. N. Wilke, *Practical Mathematical Optimization: Basic Optimization Theory and Gradient-Based Algorithms*. Springer International Publishing, 2018.

[32] G. A. Reis, J. Chang, N. Vachharajani, S. S. Mukherjee, R. Rangan, and D. I. August, “Design and evaluation of hybrid fault-detection systems,” in *32nd International Symposium on Computer Architecture (ISCA’05)*, Jun. 2005, pp. 148–159.

[33] Xilix, UG585, “Zynq-7000 all programmable SoC: Technical reference manual,” 2016.

[34] T. Instruments, “MSP430x5xx and MSP430x6xx Family User’s Guide (SLAU208M).”

[35] A. Serrano-Cases, J. Isaza-Gonzalez, S. Cuenca-Asensi, and A. Martinez-Alvarez, “On the influence of compiler optimizations in the fault tolerance of embedded systems,” in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, Jul. 2016, pp. 207–208.

[36] E. Carlisle and A. George, “Dynamic robust single-event upset simulator,” *Journal of Aerospace Information Systems*, vol. 15, no. 5, pp. 282–296, May. 2018.

[37] F. Rosa, F. Kastensmidt, R. Reis, and L. Ost, “A fast and scalable fault injection framework to evaluate multi/many-core soft error reliability,” in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*. IEEE, Oct. 2015, pp. 211–214.

[38] C. N. de Aceleradores, “Seville, Spain,” http://www.cna.us.es, Last visited: July 10th.

[39] J. Z. et al., “SRIM the stopping and range of ions in matter. SRIM co.” http://www.srim.org, 2010.

[40] S. A. Wender and P. W. Lisowski, “A white neutron source from 1 to 400 MeV,” *Nuclear Inst. and Methods in Physics Research, B*, vol. 24-25, no. PART 2, pp. 897–900, 1987.

[41] P. W. Lisowski and K. F. Schoenberg, “The Los Alamos Neutron Science Center,” *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 562, no. 2, pp. 910–914, 2006.

[42] J. Pallister, S. J. Hollis, and J. Bennett, “BEEBS: open benchmarks for energy measurements on embedded platforms,” *CoRR*, vol. abs/1308.5174, 2013.

[43] A. Martinez-Alvarez, S. Cuenca-Asensi, F. Restrepo-Calle, F. R. P. Pinto, H. Guzman-Miranda, and M. A. Aguirre, “Compiler-directed soft error mitigation for embedded systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 159–172, Mar. 2012.