Robust Convolutional Neural Networks in SRAM-based FPGAs: a Case Study in Image Classification

F. Benevenuti¹, F. Kastensmidt¹, A. Oliveira¹, N. Added², V. Aguiar², N. Medina², M. Guazzelli³

¹Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil

²Universidade de São Paulo (USP), São Paulo, Brazil

³Centro Universitário da FEI, São Bernardo do Campo, Brazil

e-mail: fglima@inf.ufrgs.br

Abstract— This work discusses the main aspects of vulnerability and degradation of accuracy of an image classification engine implemented into SRAM-based FPGAs under faults. The image classification engine is an all-convolutional neural-network (CNN) trained with a dataset of traffic sign recognition benchmark. The Caffe and Ristretto frameworks were used for CNN training and fine-tuning while the ZynqNet inference engine was adopted as hardware implementation on a Xilinx 28 nm SRAM-based FPGA. The CNN under test was generated using an evolutive approach based on genetic algorithm. The methodologies for qualifying this CNN under faults is presented and both heavy-ions accelerated irradiation and emulated fault injection were performed. To cross validate results from radiation and fault injection, different implementations of the same CNN were tested using reduced arithmetic precision and protection of user data by Hamming codes, in combination with configuration memory healing by the scrubbing mechanism available in Xilinx FPGA. Some of these alternative implementations increased significantly the mission time of the CNN, when compared to the original ZynqNet operating on 32 bits floating point number, and the experiment suggests areas for further improvements on the fault injection methodology in use.

Index Terms— Deep Learning; Neural Networks; Reliability; Single-event Effects; SRAM-based FPGA.

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are flexible components that can be customized and reconfigured to implement a large variety of designs. The large amount of computational resources available on reconfigurable devices such as FPGAs associated with microprocessors makes programmable devices attractive to demanding applications. FPGAs are attractive devices for use in automotive industry, aircrafts and satellites systems due to their high density and capability of integrating many designs into a single chip and still achieving high performance due to the process parallelism. Real-time image processing and classification using neural networks is one of the applications present in many of ones cited above.

However, integrated circuits operating in radiation environments are well known to be susceptible to errors due to particle ionization, known as soft errors. Soft errors may occur when a single ionizing particle strikes a sensitive region at the electronic device creating a current pulse and they have a

transient effect. When this current pulse occurs in a sequential logic gate near the drain of a transistor in off-state mode it may flip the value stored in the memory cell, known as Single Event Upset (SEU). When the pulse occurs in a combinational logic gate, it may be propagated through the logic, known as Single Event Transient (SET).

Circuits operating at space and at higher altitudes are subject to a complex environment that includes a large spectrum of particles of different mass each with a different energy range. The radiation experienced in space near Earth comes from several different sources [1], including protons and heavier ions from solar flares and cosmic rays. The space environment includes also protons and electrons trapped in the Van Allen Belts and heavy ions trapped in the Earth's magnetosphere. Interactions of high energy protons with higher atmosphere can generate neutrons. The interaction of those neutrons inside electronics can generate secondary ionizing particles, becoming an issue for electronics at avionics altitude. Most of these neutrons will decay on the atmosphere and reach ground level as thermal neutrons but some high-energy neutrons still can reach electronics at ground level. Both thermal neutrons and high energy neutrons can interact with electronics generating secondary ionizing particles, meaning that even circuits operating at sea level can experience transient faults caused by the interaction of low and high-energy neutrons with the materials on the integrated circuit [2].

Standard grade commercial off-the-shelf (COTS) SRAM-based FPGAs are mainly susceptible to SEUs in their configuration memory bits and embedded memory cells. SRAM-based FPGAs contains millions of memory cells and, consequently, a primary challenge for using such FPGA devices in a spacecraft or in mission critical or safety critical applications at ground level that require high reliability is to address the effects of radiation on the device operation.

Depending on the architecture of the design implemented into SRAM-based FPGAs, more or less configuration bits are used and more or less susceptible bits may be responsible for provoking an error in the design output. However, it is not only the number of used bits that determine the sensitivity, the masking effect of the application algorithm plays an important role and for that there are tradeoffs in the architecture such as

area, performance, execution time in clock cycles and types of resources utilized that may directly contribute to the soft error rate analysis in FPGAs.

When considering the design of neural networks embedded into SRAM-based FPGAs challenges reside in the search for the best neural network topology, or its structure, including the number of layers, the dimension and type of processing on each layer, the training strategy, and so on, matching the requirements of a specific task with best accuracy and resource usage in terms of time, power and area.

Strategies for automated design space exploration (DSE) have been proposed, some of them adopting the bio-inspired evolutionary engineering, including the use of genetic algorithms. These iterative and evolutionary strategies are particularly useful whenever the coverage of the whole solution space is not feasible. The genetic algorithms (GA) were already used in the search of solutions on software, hardware and scheduling, as well as in the design of neural networks [3, 4].

In this paper, we present a case study convolutional neural network (CNN) topology covering a 43 traffic sign classes dataset generated using an evolutionary approach based on genetic algorithm. The Berkeley's Caffe [5] framework enhanced with Ristretto [6] was used for CNN training and fine-tuning. The ETH Zürich's ZynqNet [7] inference engine, generated by high-level synthesis (HLS), was adopted for hardware implementation.

A CNN topology includes aspects such as the number of layers and type and properties of each layer. An important aspect of this work is that the CNN topology was not defined by hand by a human expert but evolved from scratch using a genetic algorithm process.

The ultimate goal of this approach is to use the genetic algorithm process to evolve a CNN with good fault tolerance properties on the presence of radiation. To accomplish this goal both metrics of classification accuracy and reliability must be part of the evolutionary process. A candidate to provide the reliability metric to this process is the use of fault injection on the configuration memory of the SRAM-based FPGA [8], that is to insert fault injection testing of the CNN inside each cycle of the evolutionary process.

Hence, an important contribution of this research is the comparison and cross validation of fault injection and radiation results aiming the use of fault injection as one of the qualification steps to calculate some reliability metric for the evolutionary process.

Different implementations of the CNN were evaluated under radiation and fault injection. The reliability curves obtained from cumulative frequency histogram of failure events and the metric of mission time (MT) were used in the analysis.

To a broader comparison of radiation and fault injection the CNN was analyzed in the presence of different mitigation strategies, including the association of configuration memory scrubbing, implemented natively by Xilinx's hardware, with data storage protected with Hamming code and processing on fixed-point arithmetic with reduced precision.

II. SOFT ERRORS IN SRAM-BASED FPGAS

On SRAM-based FPGAs, the configuration memory cells are responsible for configuring the design in the programmable logic blocks (CLBs), the arithmetic processing blocks (DSPs) and the switch boxes routing signals between the FPGA elements.

A SEU in the configuration memory can change the configuration of a routing connection or the configuration of a look-up table (LUT) or flip-flop in the CLBs or the configuration of the BRAMs (Fig. 1). SEUs at the configuration memory have a persistent effect but can be corrected by the reconfiguration of the FPGA, for instance by reloading the application bitstream into the FPGA. These SEUs can also be corrected by a process named memory scrubbing, for instance by a periodic scan of the configuration memory and the use of redundant information in the form of error correction codes.

Other embedded memory cells present on FPGAs are the flip-flops and memory blocks (BRAMs) that are used mostly to store user data at the application level, possibly volatile data generated dynamically by the application. Compared to the configuration memory, SEUs at flip-flops and BRAM are less persistent because they may be discarded if a new value is written to the memory, and are subject to temporal masking, as the corrupted value may be never read by the application. Upsets in the configuration memory bits and in the BRAM can be corrected by loading an entire or partial bitstream following by resetting the design.

One of the popular SRAM-based FPGA families combining hardcore embedded processors and configurable FPGA matrix is the Zynq®-7000 series from Xilinx® [9]. These system of chip (SoC) devices are manufactured in 28 nm technology and are composed of two main parts: the processor system part (PS) that features an Arm® Cortex®-A9 hardcore processor surrounded by a set of peripherals and memory interfaces while the programmable logic part (PL) consists of a Xilinx 7 Series SRAM-based FPGA.

Besides all the possible SEUs at the FPGA side, the use of SoC devices adds to the equation the SEU susceptibility of memory cells at the processor system (Fig. 2). Elements of the microprocessor subject to SEUs include the embedded on-chip-memory (OCM), processor register file, L1 and L2 memory caches and intermediate storage on the processor control and data path (Fig 3). Effects on the hardcore embedded processor can lead to data-silent corruption and

Fig. 1. SEUs at the logical elements of SRAM-Based FPGAs.

Fig. 2. Possible SEU effects in Zynq-7000 SoC.

Fig. 3. Possible SEU at the microprocessor domain.

control flow errors. Refreshing the memories and resetting the processor can be techniques to correct some of these errors.

III. QUALIFICATION UNDER SOFT ERRORS

A. Qualification parameters and metrics

There are many parameters used to characterize a design into SRAM-based FPGA under soft errors. The first group of parameters includes Area and Performance. The area of an implemented design can be expressed in terms of the number of used resources such as LUTs, flip-flops, BRAM blocks, DSP blocks, etc. Also, it is possible to express the area in terms of configuration frames. A configuration frame is the smallest addressable memory segment of the configuration memory (bitstream). Since each frame is related to a specific resource and position in the floorplan [10], it is possible to calculate the number of configuration frames used by a design. The performance of a design can be expressed in terms of the execution time, operational frequency and the processed workload. The execution time can be defined by the number of clock cycles to perform the operation. According to the FPGA and design architecture, a maximum clock frequency is achieved. Another important parameter is the workload processed by the design. The workload is the amount of data computed in one execution. In terms of reliability, the performance information is helpful to know how much time the design is exposed to errors during the execution of the implemented function.

The second group of parameters includes the Essential bits, Error and Critical bits. The essential bits are defined by Xilinx [11], and they refer to the amount of configuration bits associated to a design mapped in a certain FPGA. Essential bits are a subset of the total configuration bits, and they depend on the area of the implemented design. The errors are defined as any deviation from the expected behavior. They can be classified as Silent Data Corruption (SDC) errors and hang

or timeout errors. The main source of errors in SRAM-based FPGA is SEUs in the configuration memory. The critical bits are defined as the amount of configuration bits that once flipped they cause an error in expected design behavior (SDC or hang). The critical bits are a sub-set of the essential bits.

The third group of parameters includes radiation measurements such as static and dynamic cross sections. The static cross section (σ_{static}) is an intrinsic parameter of the device usually expressed in terms of area (usually $\text{cm}^2/\text{device}$ or cm^2/bit) and is related to the minimum susceptible area of the device to a particle species (e.g. neutron, proton, heavy ion, etc.). The expression to obtain the static cross section per bit of a device is known particle flux ϕ and time t :

$$\Phi_{particles} = \left(\phi \left[\frac{particles}{\text{cm}^2 \cdot s} \right] \right) \cdot (t [s]) \left[\frac{particles}{\text{cm}^2} \right], \quad (1)$$

$$\sigma_{static-device} = \frac{N_{SEU}}{\Phi_{particles}} [\text{cm}^2], \quad (2)$$

where N_{SEU} is the number of SEU in the configuration memory bits, $\Phi_{particles}$ is the particle fluence. The fluence is measured by particle per cm^2 , and it is calculated by multiplying the particle flux by the time the device has been exposed to that flux. The cross section per bit (Eq. 3) is calculated by dividing the static cross section by the total number of bits in the device (N_{bit}). In SRAM-based FPGAs the most important static cross section per bit is the one from the configuration memory and the user embedded memory (BRAM).

$$\sigma_{static-bit} = \frac{N_{SEU}}{\Phi_{particles} \cdot N_{bit}} \left[\frac{\text{cm}^2}{\text{bit}} \right] \quad (3)$$

The dynamic cross section ($\sigma_{dynamic}$) is defined as the probability that a particle generates a functional failure in the design. The expression to obtain the dynamic cross section is:

$$\sigma_{dynamic} = \frac{N_{events}}{\Phi_{particles}} [\text{cm}^2], \quad (4)$$

where N_{events} is the number of events observed in the design behavior and $\Phi_{particles}$ is also the particle fluence.

The fourth group of parameters include the metrics used based on the radiation measurements and estimations in the laboratory. They are Soft Error Rate (SER) and Mean Workload Between Failures (MWBF). SER can be obtained multiplying the dynamic cross section with the particle flux ϕ_{env} in the targeted radiation environment, as in Eq. 5, and is expressed in Failure in Time (FIT) units, being defined as the expected number of errors per 10^9 device operation hours in a determined radiation environment.

$$SER = \sigma_{dynamic} \cdot \phi_{env} [FIT]. \quad (5)$$

If the qualification was conducted in the target environment and operation conditions, it simplifies to:

$$SER = \frac{N_{events}}{t} [FIT]. \quad (6)$$

The metric MWBF was proposed in [12], and it evaluates the amount of data (workload) processed correctly by the design before the appearance of an output error. For a repairable system with negligible mean time to repair (MTTR) where we can simplify mean time between failures as mean time to failure (MTTF), we can compute:

$$MTBF = \frac{1}{\sigma_{dynamic} \cdot \phi} [h], \quad (7)$$

$$MEBF = \frac{MTBF}{t_{exec}} [executions], \quad (8)$$

$$MWBF = MEBF \cdot w = \frac{w}{\sigma_{dynamic} \cdot \phi \cdot t_{exec}} [data], \quad (9)$$

where w is the workload processed in one execution, $\sigma_{dynamic}$ is the dynamic cross section, ϕ is the particle flux and t_{exec} is the execution time of a single processing cycle of the design or the period between processing cycles.

The metrics presented above are useful to describe the design reliability but are inadequate compare different choices of implementation, especially in designs with the presence of parallelism, redundancy or mitigations techniques where the failure rate changes over time. A better alternative in this case may be to look at the reliability, or survival, curves of the design.

Given a set of N_{errors} observed failure events for which are known the time $t_{elapsed,i}$ of occurrence of each failure i , we can compute the empirical cumulative distribution function, or failure curve, $F(t)$ as:

$$F(t) = \frac{1}{N_{errors}} \sum_{i=1}^{N_{errors}} \mathbf{1}_{\{t_{elapsed,i} \leq t\}}, \quad (10)$$

where

$$\mathbf{1}_{\{P\}} = \begin{cases} 1 & \text{if } P \\ 0 & \text{if } \neg P \end{cases}. \quad (11)$$

Conversely, known the particle flux ϕ of the experiment we can rewrite Eq. 10 in terms of particle fluence, that is $F(\Phi)$. Being the reliability the reciprocal of failure, we can then compute the reliability curves $R(t)=1-F(t)$ and $R(\Phi)=1-F(\Phi)$.

The analysis of the reliability curves allows the comparison of alternative implementations of the design in zones of higher reliability, for instance where $R(\Phi) \geq 90\%$ or $R(\Phi) \geq 99\%$, where the behavior of the design may be very different from its behavior at lower reliability zones characterized by the MTTF and derived metrics.

It is noteworthy that, known the particle flux ϕ_{env} in the targeted radiation environment, the reliability curve $R(\Phi)$ can be translated to $R(t)$ at that specific environment.

B. Beam Testing

The accelerated radiation method consists in exposing the device in front of a particle accelerator beam and monitor the design behavior. This method is employed to obtain the device sensitivity to a determined range of particles and also to obtain a soft error rate of a circuit. In the case of FPGAs, the test can be static or dynamic. The basic procedure consists on

programming the FPGA with a known (golden) bitstream, irradiate the device and continuously readback the bitstream from the FPGA, to compare to the golden bitstream and to count the number of bit-flips. From the static test, it is possible to calculate the static cross section of the FPGA configuration memory. The static cross section can be given per bit. In case the target FPGA has already been tested for a particular particle, the static cross section can be found in papers or in datasheets. The static cross section per bit for many Xilinx FPGAs can be obtained from the its reliability report [13].

The dynamic test analyzes the design output mapped into the FPGA, and from this test is obtained the dynamic cross section and soft error rate. In this case, the expected soft error rate is much lower than the obtained from the static test. In this work, we compare the dynamic cross section with estimates obtained by the method proposed in [14].

The most common way to qualify integrated circuits for SEEs is by means of accelerated radiation testing [15]. In accelerated radiation tests, the devices are exposed to a specific radiation source whose intensity is much higher than the ambient levels of radiation that the device would normally experience. This induces the occurrence of SEUs, allowing useful data to be obtained in a fraction of the time, such as hours or days, instead of weeks, months, or even years, in case of real-time tests. Accelerated tests are performed at accelerator facilities, which accelerate specific particle species, such as neutrons, protons, and heavy ions. Neutron facilities are generally used for testing parts destined for terrestrial and avionic applications, where neutrons are the main result product of the interaction of cosmic rays with the Earth's atmosphere.

In such facilities, SEEs recorded will be due primarily to high energy neutrons (higher than 10 MeV and in the average of 14 MeV). Two of the main neutrons facilities are the Los Alamos National Science Center (LANSCE) in the United States and the Rutherford Appleton Laboratory (RAL/ISIS) in the United Kingdom. SEEs can also be induced by protons. Proton facilities are very useful because they easily reach very higher energies (usually between 50 MeV and 200 MeV) than neutron facilities. Furthermore, they are capable of generating protons with sufficient energy to simulate solar flares and Earth's proton belt conditions. One of the main proton facilities is located at the Paul Scherrer Institut (PSI) in Switzerland.

Heavy ions facilities are generally used for testing parts destined for space orbit, where primary cosmic rays can cause significant damage to electronic devices. The most important difference among heavy ion experiments and both neutron and proton experiments is related to the dosimetry. The energy measurement unit of heavy ion experiments is the Linear Energy Transfer (LET), which describes the amount of energy lost per unit length of track. In other words, it describes the action of radiation upon matter, or how much energy an ionizing particle transfers to the material traversed per unit distance. Thus, the LET depends on the nature of the radiation as well as on the material traversed. Brazil has a very useful heavy ion facility, which is located at the Universidade de São

Paulo (USP), an 8 MV Pelletron accelerator. There is also the case in which electronic components are exposed to a very high flux of different particle species. This is the reality of the CERN's accelerators chain, where electronic components can be exposed to high-energy hadrons (protons, neutrons, pions), heavy ions, and other particles, at the same time. Aiming to simulate such complex environment, in 2015 CERN started operating a new and unique mixed-field radiation test facility, the CERN High Energy Accelerator Mixed-field (CHARM), located at CERN, Switzerland. At CHARM, it is possible to have particles with energies near 10 GeV.

SEE characterization using particle beams (heavy ions, protons, or neutrons) is a global approach, since the entire device is irradiated. Such test provides a number of events for a particular fluency, without any information about the detected faults location and the time they happened. In this context, laser beams can be used as an efficient complementary tool for accelerated tests in order to evaluate the sensitivity of complex electronic components exposed to radiation and to distinguish different effects. Laser tests provide a high level of accessibility to locate the circuit elements where faults are injected. The small laser spot and precise beam localization characteristics allow sensitive device nodes to be pinpointed with submicron accuracy. Therefore, taking into account the complexity of modern devices, laser testing is especially useful since it can provoke charges with spatial localization and temporal precision that is mandatory for analyzing faults that can be easily masked in particle accelerator beam tests. Laser has the disadvantage of having its beam reflected by metallization layers, thus complex circuits must be irradiated from the backside. The National Research Nuclear University MEPhI in Russia has several dedicated laser facilities for testing electronics components.

When radiation experiments are used for design qualification, the metrics of interest are typically the cross section σ and the MTTF, and its derived metrics such as SER and MWBF, that can be combined with the respective metrics of other components to obtain a systemwide figure of reliability. However, when radiation experiments are used for comparison and choice between alternative implementations of a design, given a set of reliability requirements, the reliability curves $R(\Phi)$ and $R(t)$ provide better basis for engineering decisions. In this case, the empirical curves can be fitted to well-known probability distribution functions, such as Exponential or Weibull, providing a compact representation by a small set of parameters and an easier form for analysis and extraction of additional reliability metrics.

C. Fault injection by Emulation

Fault injection (FI) by emulation is a well-known method to analyze the reliability of a design implemented in an SRAM-based FPGA in the laboratory. In this case the FPGA bitstream is modified by a circuit or a computer tool by flipping bits to emulate an SEU in the memory cells.

In some fault injection tools, the bitstream file is modified to emulate an SEU before it is loaded into the FPGA. Other fault injectors can take advantage of dynamic partial

reconfiguration capabilities of SRAM-based FPGAs to reduce the time to inject bit-flips. In this case, the original bitstream configured into the FPGA is modified from inside the FPGA.

The output of the design under test (DUT) can be constantly monitored to analyze the effect of the injected fault into the design. If functional failure is detected, that can be a computation error or a functional interruption, hang or crash, this means that a configuration memory bit that is critical to the design was affected. The fault injection method can provide us the error analysis, such as the magnitude of the error in the computed result, classification of the type or severity of the functional failures, and the number or list of critical bits that may be used as a reliability metric or to suggest fault mitigation approaches.

In one type of fault injection methodology, faults are injected sequentially over all the FPGA configuration memory in a region of interest, one bit at a time. With this exhaustive method it is possible to obtain a complete list of all the critical bits on the design. The number of critical bits gives an overall indication of the design reliability while list with the memory address of each critical bits may be mapped back to the design logical modules to indicate which modules are candidates for implementation of fault tolerance techniques to mitigate failures and improve the design reliability.

With this method, however, the entire fault injection campaign can take from a few hours to weeks depending on the amount of bits that are going to be flipped in the region of interest of the FPGA memory, the type of connection and communication speed between the FPGA and the fault injection control, the design processing time and the processing result communication and diagnosis time.

Another fault injection methodology focuses less in the exhaustive list of critical bits and more on the expected overall behavior of the design under radiation. The occurrence of radiation effects on the electronic device may follow random patterns, for instance SEUs may occur randomly at any position of the FPGA memory, distributed uniformly over the whole device, and the SEUs may occur randomly at any time of the design processing cycle. Additionally, depending on the type, energy and incidence angle of the particle, radiation may affect multiple bits simultaneously, which may hinder the efficacy of failure mitigation techniques such as fault tolerance by modular redundancy with co-located logic or memory recovery by error correction codes that can detect or correct only single bit SEUs. Finally, instead of one SEUs at a time, the radiation effects accumulate over time until the FPGA is reconfigured or a memory correction such as periodic memory scrubbing takes place. Consequently, a fault injection method that emulate radiation effects during the normal operation of the design can be randomized and bit flips can accumulate over time.

Notwithstanding, randomized fault injection can also be seen, in some sense, as a sampling of the critical bits, although functional failures may also occur due to the cumulative effect of latent faults. The ratio of functional failures observed with a single accumulated fault injected over the total number of memory bits in the region of interest can be used to estimate

of the total number of critical bits, comparable to the number of critical bits obtained from exhaustive fault injection, but the randomized method cannot provide a complete list of memory addresses of the critical bits.

As an advantage, the functional failure observations, diagnostic and reliability metrics obtained from randomized and accumulated fault injection can be analyzed with the same statistical tools used in the analysis of the results of beam testing, meaning that from this method of fault injection we can obtain estimates of metrics similar to the dynamic cross section (σ) and mean time to failure (MTTF), as well as estimates of soft error rate (SER) and mean workload between failure (MWBf).

IV. CONVOLUTIONAL NEURAL NETWORKS (CNN)

Image classification using CNN can be used both on aerospace and ground level applications. Examples include environment, weather and climate monitoring onboard earth observation satellites and computer vision onboard Advanced Driver Assistance Systems (ADAS) and autonomous vehicles.

The subject of traffic sign recognition was adopted as benchmark in this work due to its complexity comparable to real world problems for neural networks.

Traffic sign recognition is also an application that may require safety critical behavior, thus reinforcing the importance of a fault tolerant CNN topology, although, in that case, neutron radiation on ground level would be more relevant than heavy ions radiation.

The German Traffic Sign Recognition Benchmark (GTSRB) [16] was chosen for training and test of the neural network. This dataset includes thousands of images labelled in 43 classes. Sample image classes covered by the dataset is presented in Fig. 4.

No data augmentation was applied to the dataset. Despite being originally unbalanced, two balanced subsets of images were taken from the dataset, including a similar number of images for each of the 43 classes, to be used at the training and test steps during the training stage. Another subset of images was selected, now unbalanced and preserving the same proportions of the original dataset, to be used during fault injection and radiation experiments.

The convolutional neural network (CNN) inference engine used in this work is the ZynqNet [7], originally designed by researchers from ETH Zürich to run on a Xilinx Zynq-7000 device, from which took its name.

Originally ZynqNet was both a neural network topology, that is the ZynqNet CNN, and an embedded inference engine, that is the *ZynqNet FPGA Accelerator*.

While a CNN topology, the ZynqNet CNN is related to SqueezeNet [17] and AlexNet [18] topologies. While the GTSRB dataset adopted in this work used only 43 image categories, AlexNet, SqueezeNet and ZynqNet CNN topologies targeted a different task of image classification aiming at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset covering 1000 image categories. Hence, most aspects of the ZynqNet CNN topology is not relevant to this work and, in fact, the topology of the CNN

Fig. 4. Sample GTSRB image classes: (a) speed limit signs, (b) prohibitory signs, (c) derestriction signs, (d) mandatory signs, (e) danger signs and (f) other unique signs [16].

Fig. 5. Neural network implementation and experimental setup for heavy ions.

used in this work was derived from scratch autonomously by the genetic algorithm process.

For comparison, Fig. 6 presents a simplified diagram of the aforementioned CNN topologies, designed for the ILSVRC dataset, and the evolved CNN tested in this work, targeting the GTSRB dataset. A distinctive feature of SqueezeNet and ZynqNet CNN topologies is the absence of Perceptron layers, also known as fully-connected (FC) layers, typically used as the last layers in many neural network topologies. The ZynqNet CNN also avoids the use of intermediate pooling layers and have other optimizations focusing the implementation on FPGA.

On the other side, there is the ZynqNet FPGA Accelerator, an embedded inference engine coded as software in C++ language. The ZynqNet code is annotated with Xilinx High-Level System (HLS) directives for automatic generation of optimized hardware implementation on FPGA. This code can be compiled to execute on computers.

The ZynqNet FPGA Accelerator is designed as a set of processing elements specialized for convolution computation, exploring the DSP resources on Xilinx FPGA, and a set of local memory buffers that hold both input data and weights for the convolution layer currently being processed (Fig. 5). The absence of Perceptron layers in the CNN topology, that is an all-convolutional CNN, enables the entire neural network computation to be offloaded the ZynqNet FPGA Accelerator and makes negligible the remaining load executing as software on the microprocessor.

On the original ZynqNet implementation, software executing on the Arm Cortex-A9 processor available at the Xilinx Zynq-7000 device family coordinates data movement and processing. Notwithstanding, the ZynqNet hardware uses the Arm AMBA™ Advanced eXtensible Interface (AXI™) both for data access and control, allowing it to be wired to others microprocessors through the appropriate interface bridges,

Fig. 6. Neural network layers for different CNN topologies.

including soft-cores such as Arm Cortex-M0 and RISC-V completely implemented on FPGA.

Only the ZynqNet inference engine and its approach of all-convolutional neural network were adopted in this work while a new CNN topology targeting the GTSRB task was grown from scratch, trained with the Caffe framework [5], along an evolutive process based on genetic algorithms. The original GTSRB images were preprocessed including image resizing, color space conversion and image equalization. However, even the choice of best image size and preprocessing parameters of the solution dealt by the evolutive process.

V. EVOLUTIVE ENGINEERING PROCESS

Due to the higher dimensionality of the problem, it is not feasible to cover all the possibilities in the solution space for the design of complex system on chip for machine learning applications with fault tolerance requirements. To cope with this, the proposed evolutive engineering process integrates tools and techniques from bio-inspired algorithms, expert systems, cluster computing, high-level synthesis, SEU mitigation on hardware and software, emulated fault injection, classification metrics and reliability metrics (Fig. 7).

In this process a set of rules defines the constraints inherent to the ZynqNet inference engine, such as the allowed kernel sizes and types of CNN layers implemented, constraints inherent to the target FPGA device, such as number of DSP and BRAM blocks available on the device, and constraints associated to the target application such as input image dimensions, types of image preprocessing and number of output classes.

Based on these rules and following the genetic algorithms approach, an initial population of individuals is generated randomly. Each individual on the population will be a different candidate CNN topology associated with additional configuration parameters such as image preprocessing approach, inference engine implementation strategy and,

Fig. 7. Proposed reliability aware evolutive engineering process.

occasionally, a set of fault tolerance techniques to be implemented.

On each evolutionary cycle each new individual on the population must be trained as a CNN using the target dataset. The Caffe and Ristretto frameworks are used in this step. Upon convergence on the training process each individual will have achieved a level of accuracy that will depend mostly on its topology.

As this process is based on genetics algorithms, it does not evolve one single design solution to the target task. Instead, the process maintains and evolves a set of candidate solutions that are iteratively modified, combined or discarded according to the genetics algorithms operations of mutation, crossover and selection. Naturally, best performing individuals should be kept in the population while discarding individuals with lower performance.

However, for fault tolerant applications, the performance should be measured not only in terms of classification accuracy but also in terms of reliability. For instance, we may have a CNN topology with high accuracy that may instantly stop working in the presence of radiation, or we may have a CNN topology not so good in terms of accuracy but able to sustain that level of accuracy for a long time even in the presence of radiation.

For this kind of analysis, it is proposed that reliability metrics would be provided by fault injection, as highlighted in Fig. 4. However, it is required that fault injection results are consistent with radiation before allowing fault injection metrics to drive

the evolutionary process. Hence, up to this stage of development the fault injection metrics were not integrated into the evolutionary process as we are still analyzing the consistency between radiation and fault injection.

Meanwhile, the evolutionary process is being driven only by accuracy metrics. In this aspect, different accuracy metrics could be used and we adopted the metrics named F-1 Score and Top-1 Accuracy, describe on the following.

VI. ACCURACY METRICS

The image classification engine takes an image as input and gives as output the predicted image class for that image. For any given image class, comparing the predicted image class computed by the classification engine to the true image class provided at the dataset, we can determine the number of images of this class that were correctly classified, that is the true predictions, the number of images of this class that were incorrectly classified to another class, that is the missed predictions, and the number of images of other classes that were incorrectly classified to this class, that is the false predictions.

This allows to define, for a given class k , the metrics for Precision and Recall as:

$$Precision_k = \frac{True\ Predictions_k}{True\ Predictions_k + False\ Predictions_k}, \quad (12)$$

$$Recall_k = \frac{True\ Predictions_k}{True\ Predictions_k + Missed\ Predictions_k}. \quad (13)$$

For the whole set of images tested covering a number N of classes, we have the macro-averaged metrics as:

$$Macro\ Precision = \frac{\sum_1^N Precision_k}{N}, \quad (14)$$

$$Macro\ Recall = \frac{\sum_1^N Recall_k}{N}. \quad (15)$$

As the macro-averaged metrics are unweighted means, it is expected to be less biased by the imbalance between the 43 classes of traffic signs present on the GTSRB dataset. That is, all classes are dealt as equally important and must be correctly recognized despite the occurrences of some traffic signs being significantly higher than other traffic signs.

Finally, the metric F-1 Score, used in this work, can be expressed as the harmonic mean of macro-averaged precision and recall, computed as:

$$F_1\ Score = 2 \times \frac{Macro\ Precision \times Macro\ Recall}{Macro\ Precision + Macro\ Recall}. \quad (16)$$

Another metric considered is the Top-1 Accuracy that is simply the ratio total number of true predictions over the total number of images tested.

For generality, the Top- k Accuracy considers as success if the true class of the image is among the k best hypotheses. Top- k Accuracy may be relevant to critical systems as a more conservative system may take into account not only the output class with best score, but also consider the course of action in

the case of the output with second and third-best scores. A driver assistance system, for instance, may acknowledge a second-best hypothesis that a traffic sign is a speed limit of 30 km/h event if the best hypothesis indicates that the traffic sign is an 80 km/h.

Although the different metrics can be combined to obtain the fitness metric for each individual during the evolutionary process, to avoid excessive specialization of the CNN it is convenient that the Top- k accuracy have a lower weight than the metric F-1 Score that combines both precision and recall.

The results presented in this work are from the best individual taken from a population of 50 individuals after evolved through 50 generations. The evolution of F-1 Score along the several generations on the evolutionary process is presented in Fig 8. Comparative information about the 5 best individuals on the population after 50 generations are summarized on Table I.

For the CNN topology selected for tests, when considering the subset of images selected for tests on FPGA and all the 43 traffic sign classes, the probability that the known correct class is the output of the classification engine with higher score (Top-1 Accuracy) is around 81%. On the other hand, the probability that the correct class is among the 5 best scored outputs (Top-5 Accuracy) rises to around 93%. The input to this CNN topology is a grayscale image scaled to 32×32 pixels and pre-processed with adaptive histogram equalization (CLAHE). Also, this CNN has eight processing stages (Fig. 6.d), comprising seven convolutional layers and one average pooling, leading to a total of 1,618,432 multiply-accumulate (MACC) operations.

The CNN implementation was tested in eight different configurations varying the use of configuration memory scrubbing and the data representation for CNN weights, inputs and outputs, as summarized in Table II.

TABLE I
BEST FITTED INDIVIDUALS IN POPULATION AFTER 50 GENERATIONS

Individual Identity Hash	CNN Processing Stages	Input Image Format	Input Image Preprocessing	CNN Training Iterations	MACC Operations on CNN Execution	F-1 Score
...1a28	8	32x32 grayscale	CLAHE	2000	1,618,432	0.536
...049c	9	32x32 RGB	CLAHE Y (YUV)	2000	17,743,360	0.525
...75ae	9	64x64 grayscale	CLAHE	2000	5,553,920	0.522
...7090	8	64x64 grayscale	CLAHE	2000	4,878,336	0.516
...0a36	8	64x64 grayscale	CLAHE	2000	6,332,672	0.516

TABLE II

CNN CONFIGURATIONS TESTED

Design variant	Memory scrubbing	Data representation	Design essential bits	Device utilization ^(a)	
Float	Inactive	Standard IEEE 754	32 bits	6,258,430	24%
Float S	Active	floating-point number			
Q16	Inactive	Fixed-point Q15:16 packed		4,200,499	16%
Q16 S	Active	into 32 bits			
Q10	Inactive	Fixed-point Q9:10 packed		3,538,035	14%
Q10 S	Active	into 32 bits			
Q10 H	Inactive	Fixed-point Q9:10 +		5,981,011	23%
Q10 H+S	Active	Hamming code + TMR of signal and 2 MSB bits packed into 32 bits			

^a Estimated by Xilinx Vivado tool, relative to a total of 25,697,632 configuration bits.

Fig. 8. Progress of F-1 Score metric across evolutionary generations.

In the case of processing with fixed-point arithmetics, the CNN already trained for floating-point processing using the standard Caffe framework [5] was further tuned to fixed-point using the Ristretto [6] enhancements for Caffe. The precision in the fixed-point arithmetic was chosen so that there is no negative impact in the classification engine accuracy. Thus, the same CNN classification accuracy is expected for the eight configurations selected for test.

For all the eight configurations tested the classification engine was able to process 11 image frames per second. The design implementation did not use 100% of all the FPGA resources, presenting relatively low resources density, as exemplified in Fig. 9. The FPGA area usage by a CNN design is basically limited by the number of DSP and memory available in a certain FPGA device. Although the use of fixed-point arithmetics diminished the demand for DSP blocks, the BRAM blocks usage was near 100% and remained as a major constraint.

VII. RESULTS

A. Fault Injection Campaign

The emulated fault injection instrumentation used in this work is based on Xilinx's Internal Configuration Access Port (ICAP) available on the Xilinx Zynq-7000 device. In Xilinx 7 Series FPGAs, the configuration memory is organized in frames of 101 word of 32 bits and the ICAP hardware module allows each frame to be read and write individually. An additional module [19] is integrated to the design allowing the communication with fault injection campaign controller that coordinates the initial FPGA programming, fault injection, diagnostic data collection and FPGA board reset and reconfiguration in case of failure.

Faults were injected randomly [8] over the whole configuration memory of the FPGA occupied by the ZynqNet hardware modules. The rate in which faults were injected was adjusted to be lower than the rate of the configuration memory scrubbing and similar to bit-flips rate observed in radiation experiments, measured by static tests, so that results from fault injection and irradiation have similar conditions.

As fault injection is to be integrated into the iterative process of evolutive engineering to evaluate each individual

Fig. 9. Example floorplan, for the Float CNN implementation.

member of the population, the fault injection campaigns time was reduced by truncating each fault injection cycle in up to 150 faults injected. Thus, after each FPGA reconfiguration, faults are injected, and accumulated in the absence of memory scrubbing, until a functional interruption is identified by timeout or the limit of 150 faults is reached. The CNN classification is executed continuously during fault injection and the results are recorded aside the number of faults injected.

Postprocessing of CNN output results allows the analysis of these results in terms of computation errors and classification failures. It allows the organization of failure events in the form of a cumulative frequency histogram representing failure distribution $F(f)$ computed in the same manner as in Eq. 10. Reliability under fault injection can then be computed as its complement ($R(f)=1-F(f)$). For comparability, the same procedure can be applied to fault injection and radiation experiments.

One could consider any computation error in the numeric value of the CNN output score as a functional failure. The task at hand, however, is not a classical control task and the semantic value of the classification is more meaningful than the numeric value computed. Moreover, as the CNN classification hypotheses are ranked by score, as far as the numerical errors do not affect the order of the best hypothesis, the output class will not be changed.

In some sense, for machine learning applications, the reliability metrics fuses seamlessly with accuracy metrics and this can be further extended to metrics such as Precision (as in Eq. 14), Recall (as in Eq. 15) and Top-5 Accuracy. In brief, the reliability curve can be seen as how the intelligence in the neural network is decaying over time.

B. Beam Testing Campaign

The heavy ions accelerated irradiation experiments were performed using the SAFIIRA [20] test setup dedicated to electronic devices qualification at the tandem electrostatic 8 MeV Pelletron accelerator of the LAFN-USP facility, Brazil. In this setup, the beam is produced by multiple collimations, defocusing and scattering by thin gold foils, achieving high uniformity, large area and very low particle fluxes in the range from 10^2 to 10^5 particles/cm²/s.

The experiments were conducted with ^{28}Si and ^{16}O beams. For ^{28}Si , the beam with energy of 80 MeV arriving to the experimental chamber with an estimated effective energy of 73.5 MeV after passing through the gold foils on the scattering chambers. The Linear Energy Transfer (LET) was estimated as 12 MeV/mg/cm² at the surface. The average beam flux measured during the experiments ranged from 3.2×10^2 to 1.2×10^3 particles/cm²/s. For ^{16}O the beam with energy of 44 MeV had estimated effective energy of 38.4 MeV and LET of 5.2 MeV/mg/cm² at the surface. The average beam flux of ^{16}O ranged from 4.1×10^2 to 2.3×10^3 particles/cm²/s.

A Zynq Evaluation and Development board (ZedBoard) with a delidded Xilinx Zynq-7000 was used as test vehicle for radiation experiments (Fig. 10). The Zynq-7000 device was irradiated in-vacuum with the beam at normal incidence (0°).

The heavy ions irradiation was limited to 30 seconds after each FPGA board reconfiguration. As occurred at fault injection campaigns, it was irradiated until a functional interruption was detected or the time limit was reached. Each CNN configuration was irradiated for at least 1 hour, reaching fluence in the order of 10^6 particles/cm² and no less than 100 reconfiguration cycles.

C. Comparisons Fault Injection vs. Beam Testing

In further developments of the evolutive process the metrics extracted from reliability curve from fault injection may be combined with the CNN accuracy metrics for the population ranking and selection of best solutions driven by genetic algorithm.

To allow the use of fault injection inside the evolutionary process, aside accuracy metrics, to drive the CNN evolution towards a higher fault tolerance, reliability metrics obtained from fault injection shall be consistent to metrics obtained from radiation experiments.

In the case of radiation experiments, given a total particle fluence (Φ) and the number of failure events (N_{events}) observed for the design under test, from Eq. 4, the Mean Fluence to Failure (MFTF) can also be computed as in Eq. 17, meaning the average number of particles per unit area required to experience a functional failure.

$$MFTF = \frac{\Phi_{particles}}{N_{events}} = \frac{1}{\sigma_{dynamic}} \quad (17)$$

Conversely, under fault injection, given total number of accumulated injected faults (f) and the number of failure events (N_{events}) similar metrics of error rate under fault injection ($\tau_{failure}$) [14] and Mean Time to Failure (MTTF), in terms of number of accumulated injected faults, can be calculated as in Eq. 18 and Eq.19, respectively.

$$\tau_{failure} = \frac{N_{events}}{f} \quad (18)$$

$$MTTF = \frac{f}{N_{events}} = \frac{1}{\tau_{failure}} \quad (19)$$

Finally, given a target reliability requirement, we can look at the reliability curves obtained from fault injection ($R(f)$) or from radiation experiments ($R(\Phi)$) to identify the maximum time the design under test can be exposed to faults or radiation

Fig. 10. Development board with delidded Zynq-7000 deployed on SAFIIRA experimental chamber.

while sustaining that target reliability. This metric we called as Mission Time (MT) for the given reliability requirement, where the time, in our case, is given in terms or accumulated faults injected or in terms of particle fluence.

Here, the MT was obtained directly from the empirical reliability curves $R(\Phi)$ and $R(f)$, obtained from radiation experiments and fault injection, respectively, but it could also be obtained from the equation of a fitted model of probability distribution function. For instance, for a reliability curve described by a Weibull distribution parameterized by scale parameter α and a shape parameter β in the form:

$$R(t) = e^{-\left(\frac{t}{\alpha}\right)^\beta}, \quad (20)$$

the MT sustaining a target reliability $R(t) \geq r$ could be estimated as:

$$t = \alpha(-\ln r)^{\frac{1}{\beta}}. \quad (21)$$

The charts in Fig. 11 to Fig. 14 presents the Mission Time for the design under test for a target reliability chosen arbitrarily as 90%.

The metric is presented considering two criteria. In the first case (Fig. 11 and Fig. 13), any computation error is considered as a functional error, that is any numeric difference on the CNN output. On the second case (Fig. 12 and Fig. 14), it is considered a functional failure only in the case of an incorrect image classification, that is the same criteria adopted in the analysis using the reliability curves.

Despite the difference in scale for the number of faults injected and particle fluence, we can observe in MT a strong improvement by the use of configuration memory healing based on Xilinx scrubbing mechanism. Also, we can observe in MT that the effect of memory scrubbing is stronger when we tolerate computation errors that do not cause an incorrect image classification. This means that memory scrubbing cannot avoid all computation errors but can minimize the errors such that they are not sufficiently high to cause an incorrect classification on the CNN output. This effect is observed both in fault injection and radiation.

In the absence of memory scrubbing, the results from fault injection and radiation present a similar relative ranking of the design variants in terms of Mission Time, except in the case when Hamming code is used (Q10 H), where improvements of reliability observed under radiation were not captured by fault

Fig. 11. Mission Time under fault injection, any computation error is a functional failure.

Fig. 12. Mission Time under fault injection, only incorrect image classification is a functional failure.

injection. As the fault injection mechanism in use is based on Xilinx ICAP module, it cannot reach the data content of BRAM memory. In consequence, fault injection on configuration memory cannot put in evidence the benefits of Hamming code because it cannot inject faults on data content of memory blocks. This is one source of discrepancy between fault injection using only ICAP module and radiation experiments and is an area of improvement for the fault injection methodology.

Meanwhile, there is also discrepancies in many design variants when using memory scrubbing. Here, another source of discrepancy is the potential interference of the fault injection mechanism with the Xilinx memory scrubbing hardware. Both fault injection and memory scrubbing are related to the same hardware module on Xilinx devices and the activation of the ICAP module inhibits temporarily the memory scrubbing.

Despite having numeric values in different scales for the number of emulated faults injected and the particle fluence, the tendency observed in both cases is similar. The first aspect to be observed is the same distinctive behavior of the configurations where memory scrubbing is active, when compared to scrubbing inactive, which is seen both in fault injection and radiation. There is discrepancy, however, when we compare the relative order of the CNN configurations. The configuration using Hamming code is seen as the best configuration under radiation, but not under fault injection. This may be due the fact that the emulated fault injection at the configuration memory does not affect the memory blocks (BRAM) and, thus, does not put in evidence the benefits from redundancy on data storage.

VIII. CONCLUSIONS

This work described experimental evaluation of a CNN for image classification implemented on Xilinx Zynq-7000

Fig. 13. Mission time under heavy ions (^{16}O), any computation error is a functional failure.

Fig. 14. Mission time under heavy ions (^{16}O), only incorrect image classification is a functional failure.

SRAM-based FPGA, using emulated fault injection and heavy-ions irradiation.

Results obtained for the eight CNN configurations tested under radiation demonstrates that memory scrubbing represents an important improvement on mission time. Meanwhile, the fault injection mechanism based on Xilinx ICAP module is less consistent with radiation in the presence of memory scrubbing. Fault injection results also diverges from radiation experiments when mitigation techniques are implemented to protect data content on BRAM memory, as ICAP fault injection is unable to put in evidence the benefits of such mitigations.

At least on some design variants the results are promising and help in corroborating the use of emulated fault injection as a cheap and fast source for reliability metrics driving the iterative engineering process.

However, results also suggest enhancements required in fault injection to cover other relevant parts of the FPGA besides the configuration memory and further developments on this evolutive engineering process must tackle the fault injection on memory blocks.

ACKNOWLEDGEMENTS

The authors acknowledge financial support from the Brazilian funding agencies: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001; Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

REFERENCES

- [1] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005. [Online]. Available: <https://doi.org/10.1109/TDMR.2005.853449>

- [2] E. Normand, "Single event upset at ground level," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, 1996. [Online]. Available: <https://doi.org/10.1109/23.556861>
- [3] K. O. Stanley and R. Miiikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: <https://doi.org/10.1162/106365602320169811>
- [4] L. Xie and A. Yuille, "Genetic CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 1388–1397. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.154>
- [5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia (MM '14)*. New York, NY, USA: ACM, 2014, pp. 675–678. [Online]. Available: <http://doi.acm.org/10.1145/2647868.2654889>
- [6] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018. [Online]. Available: <https://doi.org/10.1109/TNNLS.2018.2808319>
- [7] D. Gschwend, "ZynqNet: An FPGA-accelerated embedded convolutional neural network," Master Thesis, ETH, Zürich, Aug. 2016. [Online]. Available: <https://github.com/dgschwend/zynqnet>
- [8] F. Benevenuti and F. L. Kastensmidt, "Comparing exhaustive and random fault injection methods for configuration memory on SRAM-based FPGAs," in *2019 IEEE 20th Latin-American Test Symposium (LATS)*, Santiago, Chile, March 2019, pp. 87–92.
- [9] Xilinx, Inc., "Zynq-7000 All Programmable SoC: Technical reference manual," San Jose, CA, USA, April 2021, User Guide, UG585 (v1.13). [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf
- [10] Xilinx, Inc., "7 Series FPGAs configuration," San Jose, CA, USA, August 2018, User Guide, UG470 (v1.13.1). [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf
- [11] R. Le, "Soft error mitigation using prioritized essential bits," San Jose, CA, USA, April 2012, Application Note, XAPP538(v1.0). [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp538-soft-error-mitigation-essential-bits.pdf
- [12] P. Rech, L. Pilla, P. Navaux, and L. Carro, "Impact of GPUs parallelism management on safety-critical and HPC applications reliability," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 455–466. [Online]. Available: <https://doi.org/10.1109/DSN.2014.4>
- [13] Xilinx, Inc., "Device reliability report: Second half 2020," San Jose, CA, USA, April 2021, User Guide, UG116(v10.14). [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug116.pdf
- [14] R. Velazco, G. Foucard, and P. Peronnard, "Combining results of accelerated radiation tests and fault injections to predict the error rate of an application implemented in SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3500–3505, 2010. [Online]. Available: <https://doi.org/10.1109/TNS.2010.2087355>
- [15] JEDEC, "Measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices," Arlington, VA, October 2006, JEDEC Standard, JESD89A. [Online]. Available: <https://www.jedec.org>
- [16] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for trafficsign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012, Selected Papers from IJCNN 2011. [Online]. Available: <https://doi.org/10.1016/j.neunet.2012.02.016>
- [17] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2017. [Online]. Available: <https://github.com/DeepScale/SqueezeNet>
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: <http://doi.acm.org/10.1145/3065386>
- [19] J. Tonfat, L. Tambara, A. Santos, and F. Kastensmidt, *Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors*. Cham: Springer International Publishing, 2016, pp. 132–143. [Online]. Available: https://doi.org/10.1007/978-3-319-30481-6_11
- [20] V. A. P. Aguiar, et al., "SAFIIRA: A heavy-ion multi-purpose irradiation facility in Brazil," *Review of Scientific Instruments*, vol. 91, no. 5, p. 053301, 2020. [Online]. Available: <https://doi.org/10.1063/1.5138644>