Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems

Lucas Matana Luza, Daniel Soderstrom, Georgios Tsiligiannis, Helmut Puchner, Carlo Cazzaniga, Ernesto Sanchez, Alberto Bosio, Luigi Dilillo

Title: Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems

Author(s): Lucas Matana Luza, Daniel Söderström, Georgios Tsiligiannis, Helmut Puchner, Carlo Cazzaniga, Ernesto Sanchez, Alberto Bosio and Luigi Dilillo

DOI: 10.1109/DFT50435.2020.9250865

Published: 11 November 2020

Document version: Post-print version (Final draft)

Please cite the original version:
L. M. Luza et al., "Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems," 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2020, pp. 1-6, doi: 10.1109/DFT50435.2020.9250865.

Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems

Lucas Matana Luza[1], Daniel Söderström[2], Georgios Tsiligiannis[3], Helmut Puchner[4], Carlo Cazzaniga[5], Ernesto Sanchez[6], Alberto Bosio[7] and Luigi Dilillo[1]

[1] LIRMM, University of Montpellier, Montpellier, France, *{lucas.matana-luza, dilillo}@lirmm.fr

[2] Department of Physics, University of Jyväskylä, Jyväskylä, Finland, *daniel.p.soderstrom@jyu.fi

[3] IES-UMR UM/CNRS 5214, University of Montpellier, Montpellier, France, *georgios.tsiligiannis@ies.univ-montp2.fr

[4] Cypress Semiconductor, San Jose, United States, *helmut.puchner@cypress.com

[5] Rutherford Appleton Laboratory, Didcot OX11 0QX, United Kingdom, *carlo.cazzaniga@stfc.ac.uk

[6] Politecnico di Torino, Torino, Italy, * ernesto.sanchez@polito.it

[7] Lyon Institute of Nanotechnology, École Centrale de Lyon, Lyon, France, *alberto.bosio@ec-lyon.fr

Abstract

Approximate Computing (AxC) is a well-known paradigm able to reduce the computational and power overheads of a multitude of applications, at the cost of a decreased accuracy. Convolutional Neural Networks (CNNs) have proven to be particularly suited for AxC because of their inherent resilience to errors. However, the implementation of AxC techniques may affect the intrinsic resilience of the application to errors induced by Single Events in a harsh environment. This work introduces an experimental study of the impact of neutron irradiation on approximate computing techniques applied on the data representation of a CNN.

I. Introduction

In the last few years, Approximate Computing (AxC) has become a significant field of research to improve both speed and energy consumption in embedded and high-performance systems [1]. By relaxing the need for fully precise or completely deterministic operations, approximate computing substantially improves energy efficiency and reduces the memory requirement. Various techniques for approximate computing extend the design space by providing another set of design knobs for performance-accuracy trade-offs. For example, the gain in energy between a low-precision 8-bit operation suitable for vision and a 64-bit double-precision floating-point operation necessary for high-precision scientific computation can reach up to 50×, when considering storage, transport, and computation [1]. The gain in energy efficiency (the number of computations per Joule) is even more significant since the delay of basic operations is greatly reduced.

The AxC paradigm fits well for applications characterized by an intrinsic resilience to error/noise [2]. Among these applications, Convolutional Neural Networks (CNNs) have proven to be particularly suited for AxC because of their inherent resilience to errors. Indeed, from a theoretical perspective, CNNs can be considered robust due to their iterative nature and learning process [3].

However, one of the neglected aspects of AxC literature is the impact of the approximation on the reliability of a given hardware implementation. If, from one hand, AxC can reduce the cost of CNN hardware accelerator implementation, it also reduces the inherent resilience of CNN, making them more prone to errors due to an external perturbation (e.g., in harsh environments) or due to aging effects. Indeed, these errors may be amplified by the applied approximation leading to unacceptable outcomes. This point is crucial, especially when CNNs have to be deployed in safety-critical applications, such as autonomous driving [4].

Several works have been done carrying Neural Network reliability, with a significant focus on the analysis of the impact of soft errors on different abstraction levels. In [5], the reliability dependence on three different GPU

This study has been achieved thanks to the financial support of the VAN ALLEN Foundation (Contract No. UM 181387) and the Region Occitanie (Contract No. UM 181386).

This work has been supported by funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 721624.

This work has been partially founded by the projects "IDEX Lyon OdeLe" and "ReAC".

architectures (Kepler, Maxweell, and Pascal) was evaluated executing the Darknet Neural Network [6] when exposed to atmospheric-like neutrons.

In [7], the authors analyze the reliability of a 54-layer DNN (Deep Neural Network) injecting faults in the network weights and input data (images) using an accelerated neutron beam for transients errors, and fault injection test experiments for permanent faults. The inferences were made targeting floating-point values, and the results show that object detection networks have a tendency to generated wrong results when exposed to hardware faults.

An analysis of the error propagation through different abstraction layers is presented in [8]. The authors used an open-source simulator framework (Tiny-CNN) [9] as a base to a DNN simulator, being able to inject faults in specific layers. Furthermore, a different approach is shown in [10], where the authors explore the resilience of a Register-Transfer Level (RTL) model of Neural-Network (NN) accelerator. The work focuses on a High-Level Synthesis approach to characterize its vulnerability, injecting permanent and transient faults on various components in the RTL design. They conclude that the data representation mode, NN data (inputs, weights, or intermediate), NN layers, activation function, and parallelism degree have a significant impact on the severity of faults.

The proposed paper's main contribution is an analysis on the behavior of the CNNs depending on the degree of approximation in the data representation. The analysis is carried out based on the experimental results of a neutron irradiation test campaign.

The paper is organized as follows: Section II presents the context with a description of the applied approximate computing technique; Section III presents the experimental test campaign, test setup, and preliminary results; Section IV presents the next steps of this work and conclusions are given in Section V.

II. Context

Approximate computing techniques can be applied at different levels, where the approach can target software, hardware, or architectural level [11], [12]. Among all the possible techniques, in this work, we consider the data precision reduction, a technique that can be implemented both at the software and architectural levels.

Reducing the data precision of an application (i.e., the number of bits used to represent the data) is a straightforward technique to reduce the memory footprint. Reducing memory usage also reduces energy consumption at the cost of accuracy (i.e., less data have to be transferred from/to the memory). In [13], the authors show that reducing floating point precision on mobile GPUs can bring energy consumption reduction with image quality degradation. This degradation, however, can be acceptable and even imperceptible for the human eye. Reducing the bit-width used for data representation is also a popular approximation method [14]. The way data precision reduction can be used to approximate software and FPGA applications is obvious: it is a matter of code modification. In software, the precision of floating-point units can be easily modified with the use of dedicated libraries or by merely changing the type of the variable. The same can be done at VHDL/Verilog project level: the design can be adapted to process smaller data vectors.

Data precision reduction can bring useful improvements in area and energy costs for hardware projects but frequently does not present significant cost reduction at the software level. For example, fixed-point arithmetic can be used to approximate mathematical functions, such as logarithms, on FPGA implementations providing low area usage [15]. However, at the software level, it can increase the execution time of the application because all the operations and data handling routines are implemented at this level.

III. Experimental Tests

A. Test Facility

The test campaign was carried out at the Rutherford Appleton Laboratories, UK [16], where an atmospheric-like neutron spectrum is delivered in the ChipIR beamline at the second target station of the ISIS neutron source. The neutron flux in the beamline is approximately 10^9 times larger than the atmospheric neutron flux. The ChipIr facility provides a neutron flux of about 5×10^6 n/cm^2/s for energies above 10 MeV [17]. More detailed information about the neutron beam can be found in [17]–[19].

B. Experimental Test Setup

The target CNN is LeNet-5 [20], which is composed of 3 Convolutional Layers (CONV) followed by 2 Fully Connected (FC) layers, and has a total of 61,470 parameters of which 50% are in the convolutional layers. With respect to the original structure, in our own LeNet, we removed the last SoftMax layer in order to bind the last FC layer to the classification output. We trained on the MNIST handwritten digit dataset by using 32 x 32-pixel cropped pictures. The training set contained 48,000 images, with an additional 12,000 for the validation set, and 10,000 for the testing set. The learning rate started at 0.05, with the decay of 5×10^{-4} every $375(*128)$ iterations, and momentum was set to 0.9. The training was done by using the open-source framework N2D2 [21]. The LeNet-5

model description that we used is available in the framework itself. We define as “accuracy” of the CNN the capability to correctly classify the input picture. The accuracy is computed by using the top-1 score [20]. The achieved accuracy over the 10,000 testing images is 99%.

After the training, we exploited [21] in order to export the trained network as C code using three different data representations:

1) 32-bit floating-point: weights are real numbers, no approximation has been applied;
2) 16-bit integer: weights are integer (quantized), data precision reduction approximation has been applied;
3) 8-bit integer: weights are integer (quantized), data precision reduction approximation has been applied.

The C code was ported to a Xilinx Zynq-7000 based system. This device is a System-on-Chip (SoC), which provides an ARM Cortex™A9 processor attached to a 28 nm Artix®7 FPGA. The CNN application was ported to this embedded system with the use of two external memories:

- two units of the MT41K128M16JT-125, a 2 Gb SDRAM DDR3L from Micron Technologies, and
- one S27KS0642GABHI020, a 64 Mib HyperRAM™ Self-Refresh DRAM manufactured by Cypress Semiconductor.

To characterize the relative radiation response of the hardware implementations carrying the AxC techniques on the applications, we target the irradiation on the HyperRAM memory. This memory was already characterized by the authors under thermal neutrons and presented different types of errors, such as Single Bit Upsets (SBUs), stuck-at bits, and block errors that affect up to 2048 sequential memory addresses [22].

Since the memory layout for an application is divided in sections, which generally are:

- *text*: executable instructions,
- *data*: constants and statically allocated variables,
- *heap*: dynamically allocated variables,
- *stack*: store parameters for function calls, return addresses, and local variables.

the memories sections were split into the two memory devices. The HyperRAM hosted the *data* and *heap* sections, while the DDR3L allocated the *text* and *stack* sections [23]. This division was made in order to isolate the source of errors, where the main affected portion of the application are the weights and the processed image data.

The testing set was processed by the three versions of the network. Depending on the degree of approximation, the accuracy of the CNN may present a slight degradation compared to the precise CNN (i.e., the one using 32-bit floating-point representation). Tab. I summarizes the impact of the approximation on the CNN. The first column shows the CNN’s data representation, while the second column provides the obtained accuracy. The last two columns report the memory footprint of each CNN. It is interesting to point out that the accuracy degradation is really minor and affects only the *8-bit integer* data representation. This was expected because of the intrinsic error resilience of the CNN. On the other hand, the column (.rodata) shows a significant reduction of the used memory thanks to the approximation (up to 4x memory reduction).

TABLE I
SUMMARY OF PRECISION AND MEMORY FOOTPRINT FROM THE THREE DIFFERENT DATA REPRESENTATIONS.

Data Representation	Accuracy	Memory Usage (.rodata)	Memory Usage (HyperRAM)
32-bit float	99.07%	505,812 bytes	56.69%
16-bit integer	99.07%	273,176 bytes	53.70%
8-bit integer	99.05%	154,112 bytes	52.17%

In order to increase the reliability of the system, the SoC configuration memory (CRAM) was monitored by the commercial Xilinx scrubber, the Soft Error Mitigation (SEM) core, which reports detected SBUs, and, when possible, corrects them [24]. Fig. 1 depicts the top-level diagram of the system.

During the irradiation campaign, for the first slot of the tests, four boards were mounted in the setup in a configuration that exposed only the HyperRAM memories to the beam.

Concerning the systems that were running the CNN applications, we used two different portions of 2,000 images from the training set, where the output result during the irradiation was compared with a golden result to enable the identification of faulty runs.

At the same time, one of the memory samples was characterized under the radiation beam by applying dynamic and static test algorithms. The dynamic test algorithms perform patterns of write and read operations in the memory, emulating a real application and enabling the identification of functional faults, and they are commonly used at production level as presented in [25], [26]. In the static test mode, a known data pattern is written in the memory

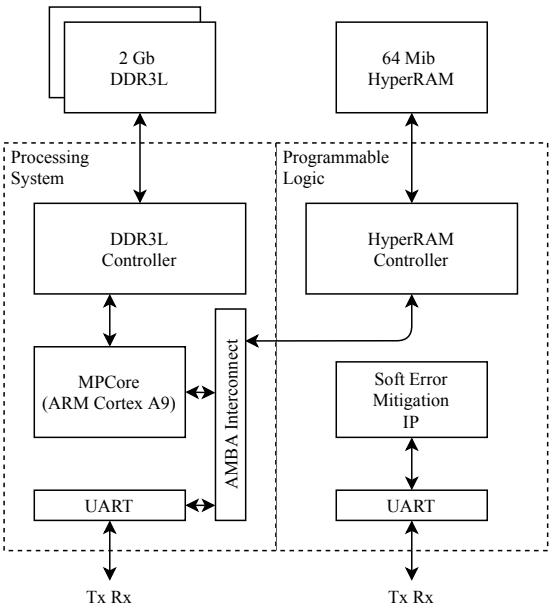

Fig. 1. System top level diagram.

before the irradiation. Then the device is exposed to the beam for a specific fluence; when the beam is stopped, a read-back operation is performed to identify corrupted bits [22], [27], [28].

Each Device Under Test (DUT) was used for a different application. However, for the second irradiation slot, the setup was changed to use only two DUTs, where the "HyperRAM 1" kept running the characterization tests, and the "HyperRAM 2" was performing interleaving runs using the three versions of the CNN, following the description given in Table II.

TABLE II
SUMMARY OF SPECIMEN AND TYPE OF TESTS USED DURING THE TEST CAMPAIGN.

Test Slot	DUT	Test Type
Slot 1	HyperRAM 1	Dynamic and Static tests
	HyperRAM 2	CNN with 32-bit floating-point
	HyperRAM 3	CNN with 16-bit integer
	HyperRAM 4	CNN with 8-bit integer
Slot 2	HyperRAM 1	Dynamic and Static tests
	HyperRAM 2	CNN with: - 32-bit floating-point - 16-bit integer - 8-bit integer

C. Analysis of Results

Based on the characterization tests, we identified three types of errors. The results are in line with the ones that we presented in [22], but with a soft-error rate a little bit higher since that work concerned thermal neutrons (implying lower energy). The first failure mode consists of SBUs. The second one recalls the stuck-at bit fault, which appears as permanent or temporary. The stuck-at bit fault results in memories cells that had their retention time affected by a particle interaction resulting in a cell that always returns the same value. The third type of error that causes a higher concern consists of block errors.

Fig. 2 presents a run of a dynamic test named mMats+ (Eq. 1)

$$\uparrow (w0); \{\uparrow (r0, w1); \uparrow (r1, w0)\} \tag{1}$$

which is a modified version of the March Mats+ algorithm [28]. In Fig. 2, the dots represent a faulty word detected during an element of the test. The faults detected during the $\uparrow (r0, w1)$-element are depicted in blue, and the ones detected during the $\uparrow (r1, w0)$-element are in orange. It is possible to identify several stuck bits appearing as

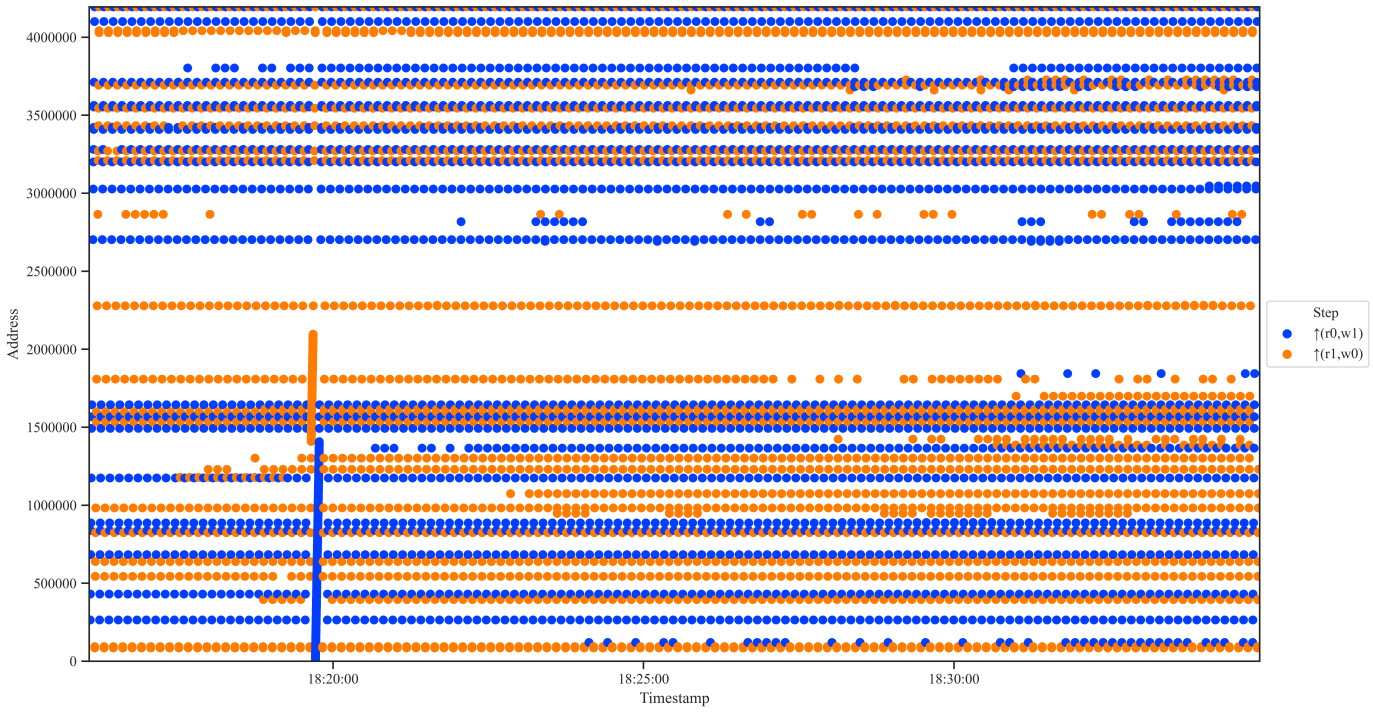

Fig. 2. Errors during a mMats+ test run. The dots represent a faulty word detected during the different operations of the algorithm.

temporary and permanent during the test run (horizontal sequences of dots on the graph), and also a block error (appearing as two vertical sequences of dots) affecting 2048 addresses.

Fig. 3 depicts a logical bitmap of the memory related to a part of an irradiation test run. In this figure, it is possible to identify the block error appearing in a vertical shape. The logical bitmap is built dividing the memory addresses into two parts, with odd rows in the left and even rows in the right. Each pixel represents a bit cell, where the black pixels represent a bit that was read with an error. The grey lines are used to limit the memory region, and a rectangle zoom-in is added to increase the visibility of the block event.

To evaluate the memory sensitivity to the presented errors, the failure types were divided into SBUs, stuck-at bits, and block errors. We calculated the event cross section (σ) as

$$\sigma = \frac{N}{F \times M} \tag{2}$$

where N is the number of events, F is the cumulative fluence in particles/cm^2, and M is the number of bits [29]. Also, the Soft-Error Rate (SER) is defined as

$$SER = \sigma \times (1024 \times 1024) \times 10^9 \times j \tag{3}$$

where 1024×1024 (bits) is the Mb coefficient, 10^9 is the FIT definition, and j (13 particles/cm^2/h) is the neutron energies' ($>$ 10 MeV) flux at New York (sea level) outdoors for a mean solar activity defined in JEDEC JESD89A [30], [31]. Table III presents the acquired results.

The experimental tests using the CNN were made in parallel with the characterization test of the memory device, in which the output of the algorithm is composed of a frame that indicates if the image was correctly recognized, the cumulative success rate, and the amount of time needed to make the inference.

We identified slightly different results comparing the default (32-bit floating-point) CNN and the ones that apply the approximate computing techniques. In total, 20 runs were performed using the *Float 32* version, 24 runs for the *Integer 16*, and 23 runs for the *Integer 8* version. In this document, a "run" is defined as the inference of a set of 2,000 images. Some "runs" did not achieve the end of its execution since a functional interruption occurred, which did not affect the DUT (HyperRAM), but other parts of the computation system. In these cases, we consider for calculations only the processed images within a "run". The number of runs was limited due to the available beam time and the limited capacity of the embedded system to run the CNN.

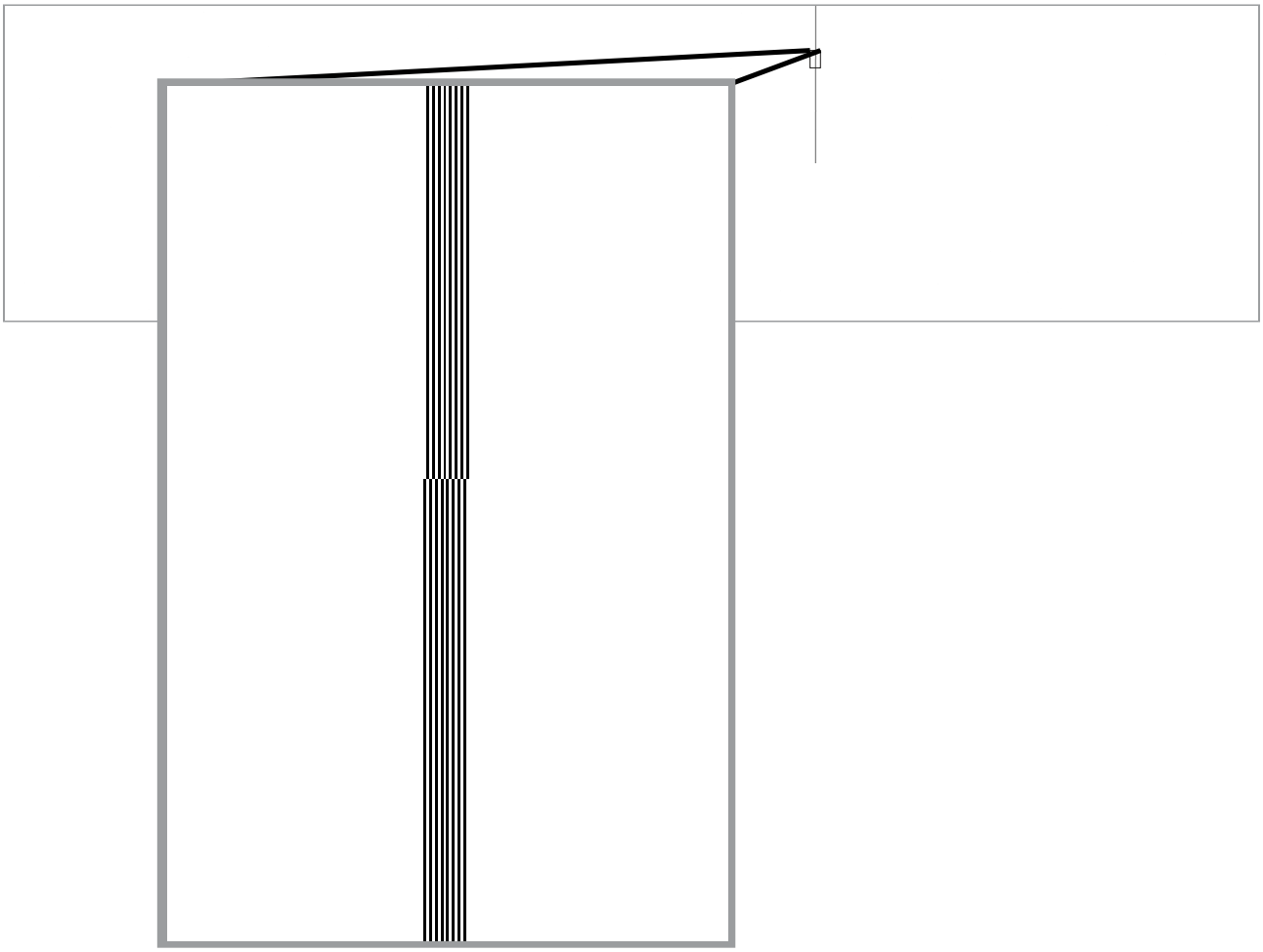

Fig. 3. Bitmap obtained from a read operation during a dynamic test. Each pixel represents a bit; each bit that was identified with an error appears in black. The gray lines are used to limit the region. Zoom-in is added to increase the visibility of the block event.

TABLE III
ESTIMATED CROSS SECTION WITH 95% CONFIDENCE INTERVALS USING A FLUENCE UNCERTAINTY OF 10%, AND SER FOR THE FAILURE TYPES IDENTIFIED IN THIS STUDY. THE VALUES WERE CALCULATED USING THE EQ. 2 AND 3

Failure type	σ **(cm²/bit)**	**Lower limit** σ **(cm²/bit)**	**Upper limit** σ **(cm²/bit)**	**SER (FIT/Mb)**
SBU	$2.86 * 10^{-17}$	$2.53 * 10^{-17}$	$3.19 * 10^{-17}$	$3.90 * 10^{-1}$
Stuck bit	$1.48 * 10^{-17}$	$1.30 * 10^{-17}$	$1.66 * 10^{-17}$	$2.02 * 10^{-1}$
Block error	$6.68 * 10^{-19}$	$4.74 * 10^{-19}$	$9.23 * 10^{-19}$	$9.10 * 10^{-3}$

The outputs of the *Float 32* version did not suffer any impact during the irradiation, with all the performed runs always returning results consistent with the golden one. For the *Integer 16* version, two runs returned fault results, affecting the capability to recognize the images. The first faulty run gave a success rate of 25.8 %, where the expected result was 99.15 %. The second one was only able to provide correct results in 0.69 % of the cases, in this case, the expected result was 98.85 %. The two different expected results are related to the use of two different portions (2,000 images each) of the inputs set.

Regarding the *Integer 8* experiments, we identified 4 faulty runs. In details, the first run resulted in a success rate of 72.9 %, where the gold result is 99.15 %, in which, one of the control variables in a *for loop* interaction was affected, and the algorithm did not stop the loop, leading to wrong inferences of more than 2,000 images.

In the second faulty run, after image number 429 had been processed, the output frame was sent with all the variables with the value 0, showing that a block error occurred in the irradiated memory in the address range that was storing the concerned variables.

The third faulty run returned all the output frames with a corrupted value. In this case, a direct correlation between a memory fault and wrong data output is clear, since it likely happened that a block error event affected the memory portion with the output frame information.

The fourth faulty run resulted in a success rate of 16.75 %, where the expected result was 98.8 %. The first fault recognition was in the image 169, and after this point, the CNN was able to keep recognizing the images, but just once in a while between several faulty inferences.

The errors occurred in two different ways, affecting the software execution (when the output frame is corrupted), and affecting the CNN inference (when the network is not able to return the expected result). It is important to underline that after each execution of the CNN, the entire processor system was reprogrammed, preventing the Soft Error accumulation. Thus, this faulty behaviour is not due to stuck bits accumulated in the memory along with the irradiation, which was not able to impact the network, since it does not lead to a gradual degradation of inference capability. On the other hand, considering the calculated error cross section, which is relatively low, and taking into account that the memory usage does not correspond to 100 % of the memory storage capacity, we may correlate the faulty runs mainly to the effects of block errors in the memory.

As a general outcome of the experiments, these results confirm that approximation on the data representation affects the intrinsic resilience of the application because the amount of wrong outputs tends to increase with the applied level of approximation.

Also, in order to complete the analysis of the results, we consider the failures that concerned the FPGA during the runs. The SEM was able to identify SBUs, MBUs, and uncorrectable errors in the FPGA configuration memory, but these faults did not have any correlation with the problematic runs.

IV. Future Work

The group's target aims at evaluating the radiation-induced effects on a real embedded architecture, exploiting the experimental data on the impact of radiation on the hardware.

Alongside the studies on effects on different memory devices, which is a regular target of the group, the characterization tests made on the HyperRAM provide valuable information on the occurrence frequency of the failure types and how it affects the memory operation. These experimental data will be used to deeply investigate the correlation between the failure modes occurring within the memory device used to store the weights and data to be processed by the CNN, and the effects on the execution of the algorithms in terms of results quality and operational resilience. In particular, based on the experimental data, the fault injection will use realistic failures, taking into account the fault models and cross sections. This will enable the injection of errors in different regions of the network, allowing the identification of the critical parts.

V. Conclusion

Ionizing particles induce soft errors that impact the memory (HyperRAM in this case) in different ways with SBUs, stuck-at bits, and block errors. These different types of failures can lead to a wrong execution of the CNN. Experimental results show that, on one side, the amount of wrong outputs in CNN execution tends to increase when the approximation computing technique is stronger. On the other side, the types of errors seem to affect the three different versions of the network in the same way with similar failure modes.

For the next steps of the study, the use of a fault emulator will provide us complementary data to produce a deeper analysis, where it would be possible to define better the correlation between the occurrence of faults in the memory and the failure in the execution of the CNN algorithms along with the application of the different approximation computing techniques.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016, doi: 10.1145/2893356.

[2] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–9, doi: 10.1145/2463209.2488873.

[3] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *CoRR*, vol. abs/1511.06488, Nov. 2015.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2722–2730, doi: 10.1109/ICCV.2015.312.

[5] F. F. dos Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux, and P. Rech, "Evaluation and mitigation of soft-errors in neural network-based object detection in three GPU architectures," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Aug. 2017, pp. 169–176, doi: 10.1109/DSN-W.2017.47.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[7] A. Lotfi *et al.*, "Resiliency of automotive object detection networks on GPU architectures," in *2019 IEEE International Test Conference (ITC)*, Nov. 2019, pp. 1–9, doi: 10.1109/ITC44170.2019.9000150.

[8] G. Li *et al.*, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2017, pp. 1–12, doi: 10.1145/3126908.3126964.

[9] Tiny-CNN. (2016) [Online]. Available: https://github.com/nyanp/tiny-cnn.

[10] B. Salami, O. Unsal, and A. Cristal, "On the resilience of RTL NN accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Sept. 2018, pp. 322–329, doi: 10.1109/CAHPC.2018.8645906.

[11] G. Rodrigues, F. Lima Kastensmidt, and A. Bosio, "Survey on approximate computing and its intrinsic fault tolerance," *Electronics*, vol. 9, no. 4, p. 557, Mar. 2020, doi: 10.3390/electronics9040557.

[12] L. Anghel, M. Benabdenbi, A. Bosio, M. Traiola, and E. I. Vatajelu, "Test and reliability in approximate computing," *Journal of Electronic Testing*, vol. 34, no. 4, pp. 375–387, May 2018, doi: 10.1007/s10836-018-5734-9.

[13] C. C. Hsiao, S. L. Chu, and C. Y. Chen, "Energy-aware hybrid precision selection framework for mobile GPUs," *Computers & Graphics*, vol. 37, no. 5, pp. 431–444, Aug. 2013, doi: 10.1016/j.cag.2013.03.003.

[14] C. Rubio-González *et al.*, "Precimonious: Tuning assistant for floating-point precision," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Nov. 2013, pp. 1–12, doi: 10.1145/2503210.2503296.

[15] J. G. Pandey, A. Karmakar, C. Shekhar, and S. Gurunarayanan, "An FPGA-based fixed-point architecture for binary logarithmic computation," in *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*, Dec. 2013, pp. 383–388, doi: 10.1109/ICIIP.2013.6707620.

[16] G. Tsiligiannis *et al.*, "SEE response evaluation of robotic systems for the nuclear decommissionning," Mar. 2020, doi: 10.5286/ISIS.E.RB2010438.

[17] C. Cazzaniga and C. D. Frost, "Progress of the Scientific Commissioning of a fast neutron beamline for Chip Irradiation," *Journal of Physics: Conference Series*, vol. 1021, p. 012037, May. 2018, doi: 10.1088/1742-6596/1021/1/012037.

[18] C. Cazzaniga, M. Bagatin, S. Gerardin, A. Costantino, and C. D. Frost, "First tests of a new facility for device-level, board-level and system-level neutron irradiation of microelectronics," *IEEE Transactions on Emerging Topics in Computing*, Nov. 2018, doi: 10.1109/TETC.2018.2879027.

[19] D. Chiesa *et al.*, "Measurement of the neutron flux at spallation sources using multi-foil activation," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 902, pp. 14–24, Sept. 2018, doi: 10.1016/j.nima.2018.06.016.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998, doi: 10.1109/5.726791.

[21] CEA-LIST. N2D2. [Online]. Available: https://github.com/CEA-LIST/N2D2.

[22] L. Matana Luza *et al.*, "Effects of thermal neutron irradiation on a Self-Refresh DRAM," in *IEEE 15th International Conference on Design & Technology of Integrated Systems in Nanoscale Era*, Apr. 2020, pp. 1–6, doi: 10.1109/DTIS48698.2020.9080918.

[23] L. D. Pyeatt, *Modern Assembly Language Programming with the ARM Processor*. Newnes, 2016, doi: 10.1016/C2015-0-00180-0.

[24] "PG036 - Soft Error Mitigation Controller v4.1 Product Guide," Xilinx, 2018, [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/sem/v4_1/pg036_sem.pdf.

[25] V. G. Mikitjuk, V. N. Yarmolik, and A. J. Van De Goor, "RAM testing algorithms for detection multiple linked faults," in *Proceedings ED&TC European Design and Test Conference*, Mar. 1996, pp. 435–439, doi: 10.1109/EDTC.1996.494337.

[26] G. Tsiligiannis *et al.*, "Dynamic test methods for COTS SRAMs," *IEEE Transactions on Nuclear Science*, vol. 61, no. 6, pp. 3095–3102, Dec. 2014, doi: 10.1109/TNS.2014.2363123.

[27] A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, *Advanced Test Methods for SRAMs*. Boston, MA: Springer US, 2010, doi: 10.1007/978-1-4419-0938-1.

[28] D. Niggemeyer, M. Redeker, and J. Otterstedt, "Integration of non-classical faults in standard March tests," in *Proceedings. International Workshop on Memory Technology, Design and Testing (Cat. No.98TB100236)*, Aug. 1998, pp. 91–96, doi: 10.1109/MTDT.1998.705953.

[29] E. Petersen, *Single Event Effects in Aerospace*. Hoboken, NJ, USA: John Wiley & Sons, 2011, doi: 10.1002/9781118084328.

[30] *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices*, JEDEC Solid State Technology Association Std. 89A, Oct. 2006, [Online]. Available: https://www.jedec.org/standards-documents/docs/jesd-89a.

[31] G. Tsiligiannis *et al.*, "SRAM soft error rate evaluation under atmospheric neutron radiation and PVT variations," in *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, Jul. 2013, pp. 145–150, doi: 10.1109/IOLTS.2013.6604066.