HAL open science

Experimental evaluation of neutron-induced errors on a multicore RISC-V platform

Fernando Fernandes dos Santos, Angeliki Kritikakou, Olivier Sentieys

To cite this version:

Fernando Fernandes dos Santos, Angeliki Kritikakou, Olivier Sentieys. Experimental evaluation of neutron-induced errors on a multicore RISC-V platform. IOLTS 2022 - 28th IEEE International Symposium on OnLine Testing and Robust System Design, Sep 2022, Torino, Italy. pp.1-7. hal-03697265

HAL Id: hal-03697265

https://inria.hal.science/hal-03697265v1

Submitted on 16 Jun 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Experimental evaluation of neutron-induced errors on a multicore RISC-V platform

Fernando Fernandes dos Santos, Angeliki Kritikakou, and Olivier Sentieys
Univ. Rennes, INRIA, Rennes, France

***Abstract*—RISC-V architectures have gained importance in the last years due to their flexibility and open-source Instruction Set Architecture (ISA), allowing developers to efficiently adopt RISC-V processors in several domains with a reduced cost. For application domains, such as safety-critical and mission-critical, the execution must be reliable as a fault can compromise the system's ability to operate correctly. However, the application's error rate on RISC-V processors is not significantly evaluated, as it has been done for standard x86 processors. In this work, we investigate the error rate of a commercial RISC-V ASIC platform, the GAP8, exposed to a neutron beam. We show that for computing-intensive applications, such as classification Convolutional Neural Networks (CNN), the error rate can be 3.2× higher than the average error rate. Additionally, we find that the majority (96.12%) of the errors on the CNN do not generate misclassifications. Finally, we also evaluate the events that cause application interruption on GAP8 and show that the major source of incorrect interruptions is application hangs (i.e., due to an infinite loop or a racing condition).**

I. Introduction

Thanks to the open-source Instruction Set Architecture (ISA), RISC-V based processors are today adopted in several domains, including end-user applications [1], High Performance Computing (HPC) [2], and safety-critical applications [3], [4]. As RISC-V architecture allows full customization, it enables systems design with reduced cost. Consequently, RISC-V processors became a promising option for safety-critical and mission-critical systems, where power consumption, real-time execution, and reliability are of highest importance. However, the majority of RISC-V works mainly focus on design for performance and power consumption, often neglecting reliability. RISC-V architectures need to be evaluated in order to characterize the errors that can reduce their reliability and potentially prevent the system from meeting its mandatory constraints.

The sources of such errors can be environmental perturbations, ionizing radiation, software errors, process, temperature, or voltage variations [5], [6]. It has been demonstrated that the faults caused by radiation have the highest error rates [7]. A terrestrial neutron strike may perturb a transistor's state or generate bit-flips in memory or current spikes in logic circuits that, if latched, lead to an error. Neutron-induced events usually are soft because the device is not permanently damaged. A new write operation will correctly store the value on the struck memory cell, and consequently, a new operation using the struck logic gate will provide the correct result. To evaluate the device reliability under radiation, researchers expose devices such as CPUs [8], GPUs [9]–[11], FPGAs [12], [13], and Tensor Processing Units [14] to a radiation flux and measure the error rate. Recently, the error rate of soft RISC-V processors synthesized on FPGAs has been measured, when exposed to heavy ions and neutrons [15]–[17]. However, as far as we know, none of the existing works considered RISC-V processors physically implemented as an ASIC.

This work is the first to characterize the error rate of a commercial ASIC RISC-V platform, i.e., the ultra-low-power GAP8 platform from GreenWaves [18] with a RISC-V master core and a cluster of 8 RISC-V cores, measured on neutron beam experiments. Beam experiments provide a realistic error rate that can be directly scaled to terrestrial flux, and thus, our results can be used to estimate the real error rate of the GAP8 RISC-V platform. Additionally, to effectively investigate the error rate, we classify the fault outcome of each incorrect execution. We, then, can distinguish, for instance, if the execution hangs or crashes due to a memory error.

We report results from beam testing campaigns that assess the reliability of the GAP8 platform, while running a representative set of codes, and measure the error rates and the Mean Execution Between Failures (MEBF). The beam data covers accelerated testing that represents a total of more than 570,000 years of the platform operation. The main contributions of this work are as follows:

- Provide experimental data and findings on the impact of the neutron-induced errors on GAP8 RISC-V platform.
- Reliability evaluation of 5 representative codes from different domains, from signal processing to machine learning.
- Analysis of the fault outcomes for the erroneous executions observed in the experiments using the GAP8 Software Development tool Kit (SDK).
- We measure the MEBF to estimate the number of executions that GAP8 devices would perform before experiencing a failure.

The remainder of the paper is organized as follows. Section II presents the background on radiation effects on computing devices and the current advances in RISC-V reliability. Section III presents the tested device, selected codes, and describes the evaluation methodology. The experimental results are presented in Section IV, and finally Section V concludes the paper.

II. Radiation effects on RISC-V devices

The effects of radiation has been studied for CPUs [8], GPUs [9]–[11], FPGAs [12], [13], and Deep Neural Networks accelerators [14], ARM CPUs [19], and memories [20]–[22] on beam experiments. When the device is exposed to the beam of accelerated particles, faults will be induced on the hardware and may manifest as output errors. Beam experiments are the standard methodology for injecting faults on an electronic device. As the chip is equally exposed to the beam of particles the faults will not be limited only to a subset of the device resources. Consequently, beam experiments provide a realistic device error rate running an application.

As RISC-V processors have been employed in several application domains, including space exploration [3], [23], HPC [2], and safety-critical systems [4], it becomes imperative to study RISC-V reliability against neutron-induced errors. The reliability of soft RISC-V cores synthesized on FPGAs exposed to neutrons and heavy ions has been studied in past works [15]–[17]. Researchers applied Triple Modular Redundancy for RISC-V and evaluated its reliability in beam experiments [15]. In fact, as RISC-V is an open-source ISA, developers can easily modify the architecture to add fault tolerance and evaluate the hardened architecture on beam experiments. However, the fault model and error rate measured on beam experiments are significantly hardware-dependent. As a result, the fault model and error rate generated for soft cores synthesized on FPGAs do not correspond to the fault model and error rate for ASIC processors, and thus, the evaluation of soft cores synthesized on FPGAs is not representative of applications running on ASIC RISC-V processors. Note that RISC-V processors built as ASIC are preferable over simulated cores, especially for ultra low power applications, as the use of an FPGA may add an unnecessary power consumption.

To the best of our knowledge, this is the first paper that: (1) evaluates an ASIC commercial mutlicore RISC-V platform (GAP8) on a neutron beam and extracts the realistic error rate for five representative codes; (2) evaluates the observed outcomes on the beam experiments. Furthermore, we keep a trace of each incorrect execution on the beam experiments in order to further analyse it, providing useful insights.

A. Fault propagation on GAP8

The natural flux of high-energy neutrons at sea level is about 13 $neutrons/(cm^2 \times h)$ [24]. When iterating with the hardware, high-energy neutrons may generate soft errors in the system. Figure 1 shows the error propagation on GAP8 ultra-low-power RISC-V platform. The striking particle can generate single or multiple bit flips on a memory resource such as caches, registers, buffers, or even corrupt the value of a functional unit inside a processing core. If the value is used as part of the algorithm computation, the incorrect value will be propagated by the code running on the device. At the application level, the fault may manifest as the following outcomes: (1) **No effect on the program output**: The fault is masked, the program output is not affected or the circuit functionality is not affected. (2) **Silent Data Corruption (SDC)**: The program finishes, but the output is not correct, and no flag nor indication is raised. (3) **Detected Unrecoverable Error (DUE)**: The system stops working, forcing it to be rebooted or power cycled. A DUE can result from events such as uncorrectable memory events, crashes, or an error that generates an infinite loop (the program hangs).

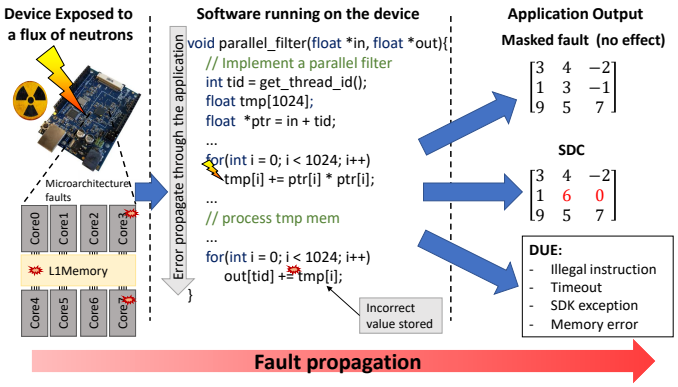


Fig. 1: Fault propagation on a multicore RISC-V platform (GAP8) when exposed to a flux of neutrons. The fault is generated at the hardware level as an incorrect functional unit computation on the parallel cores or bit flips in the memory resources. The fault is then propagated on the application execution and manifested as an SDC, DUE, or no observed effect at the application output.

Commercial devices like GAP8 often offer a SDK that allows the developers to profile and trace events on the application running on the device. Researchers have successfully monitored neutron-induced events using the SDK or the operating system for ARM [25], FPGAs [26], and GPUs [11], [27]. The event tracing helps us understand the sources of the errors and the syndromes caused by the radiation in the experiments. On GAP8, we can classify the DUE observed in each code execution into four main sub-classes:

- **Illegal instruction**: The GAP8 SDK returned a crash code that represents an invalid instruction being executed. Illegal instructions can happen when a neutron-generated error corrupts the instruction's opcode;
- **SDK exception**: The GAP8 SDK is responsible for initializing and configuring the device before executing the application for each iteration. If the SDK cannot pre-set the memory or execute the application on the device, it will throw an exception to the user. When the SDK exception happens multiple times sequentially, we perform a power cycle on the board;
- **Timeout**: A Single Event Effect (SEE) on GAP8 may lead to an infinite loop inside the application. To avoid the application hanging indefinitely, we set a *Timeout* limit to stop the running code and start it again;
- **Memory error**: When the application running on GAP8 cannot allocate memory, the API will generate an error reporting a memory allocation error.

III. Experimental methodology

In this section, we describe the GAP8 RISC-V processor, the codes we characterize, the metrics adopted, and how they are measured for GAP8.

TABLE I: Used codes characteristics. *The execution time is extracted using on the GAP8 SDK.

	Domain	Complexity	Used input	Exec. time* [s]
Finite Impulse Response	Signal processing	$O(n)$	32KB array	6.4
Matrix Addition	Linear algebra	$O(n^2)$	1KB array	3.3
Bilinear Resize	Image processing	$O(n^2)$	322x242 Matrix	9.1
Matrix Mul	Linear algebra	$O(n^3)$	M1(70x200) M2(150x70)	3.7
MNIST (CNN)	Machine learning	$O(l * n^3)$	MNIST frame	4.9

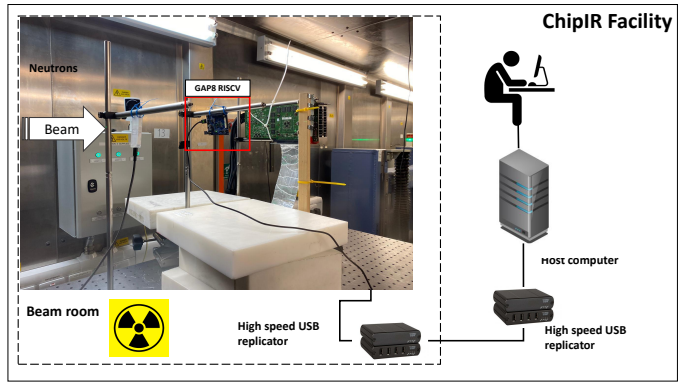


Fig. 2: GAP8 setup for the neutron beam experiments.

A. Device under test and evaluated codes

Evaluated platform: We consider the GAP8, a multicore RISC-V platform from GreenWaves technologies. The device under test is built with $55nm$ TSMC 55LP CMOS technology. The SoC comprises a cluster of 8 RISC-V cores connected by a Logarithmic Interconnect crossbar, and a RISC-V main processor to manage the cluster. GAP8 has an L1 memory of 64KB and an L2 memory of 512KB, and both memories are shared between the cores of the cluster. Each core of the cluster operates on a maximum frequency of 175MHz. It supports only integer and fixed-point arithmetic. GAP8 is an ultra low power device, and it can process 22.65GOPS with a power consumption of 96mW [18].

Evaluated codes: We chose the five representative codes listed in Table I from signal processing to machine learning domains. The selected codes vary in their complexity and implementation characteristics. Codes such as *MNIST Convolutional Neural Network (CNN)*, *Matrix Addition*, and *Matrix Multiplication* are highly parallelizable, easily distributed between GAP8 eight cores, and computing-intensive. That is, for those codes, the memory instructions are reduced, and most of the instructions are arithmetic ones. On the other hand, *Finite Impulse Response (FIR)* and *Bilinear Resize* also have computing intensive parts but also have a high number of memory instructions and synchronizations between the main core and cluster.

The choice of diverse codes increases the quality of the obtained results, which are extendable to different applications [28]. We select a CNN to classify handwritten digits as a representative case of embedded machine learning algorithms. The CNN is a quantized (16-bit integer) version of LeNet CNN [29] and has two convolutional layers, each one followed by a ReLU and a MaxPooling layer. The last layer that performs classification is a Linear layer. For the CNN error evaluation, we separate the errors observed on the classification output by their criticality. That is, the errors that are observed on the evaluated CNN can be separated into *Tolerable SDCs* (i.e., the ones that do not affect the inference result) and *Critical SDCs* (i.e., the ones that modify the classification result).

B. Beam Experiment Setup

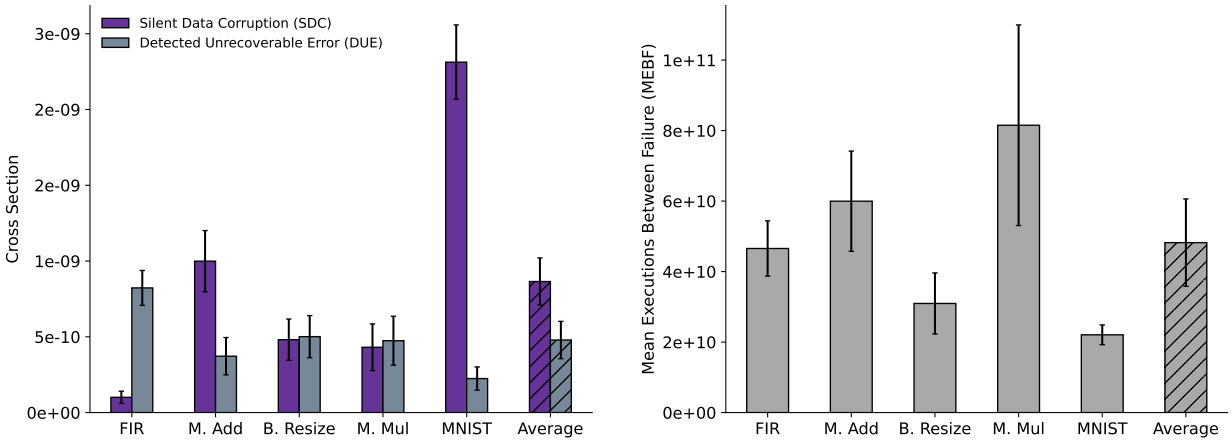
Using beam experiments, we can effectively evaluate the reliability of the GAP8 platform. We expose the GAP8 to a neutron beam to evaluate its error rate. We use the default GAP8 SDK to build and run the experiments. Our experiments are performed at the ChipIR facility of the Rutherford Appleton Laboratory, UK. Figure 2 shows the setup of the experiments. The facility delivers a beam of neutrons with a spectrum of energies that resembles the atmospheric neutron one [30]. The available neutron flux was about $3.5 \times 10^6 n/(cm^2/s)$, $\sim$8 orders of magnitude higher than the terrestrial flux at sea level [24]. Since the terrestrial neutron flux is low, it is improbable to see more than a single corruption during program execution in a realistic application. We have carefully designed the experiments to maintain this property (observed error rates were lower than 1 error per 2,000 executions). Experimental data can then be used to estimate the error rate in the terrestrial radioactive environment.

We created a software watchdog that runs on the GAP8's host computer (see Figure 2) to monitor and perform recovery from device hangs. The software watchdog controls the application under test by executing the applications and monitoring if the execution does not return a result inside a time limit. Then, the program is killed and relaunched if it stops responding in a predefined time interval. We set each code timeout individually depending on the code execution time, up to 6$\times$ the expected execution time. The software watchdog also logs if the application crashes during the execution of the code. When the application hangs, the software watchdog uses an ethernet-controlled switch to perform a power cycle on the USB replicator, rebooting the GAP8 board. The board power cycle is necessary to detect when the GAP8 stops responding.

From beam experiments, we can calculate the *cross-section* by dividing the number of observed errors by the received particles fluence η $(neutrons/cm^2)$, as show in Equation 1.

$$\sigma[cm^2] = \frac{\#errors}{\eta} \tag{1}$$

The fluence η is obtained by multiplying the average neutron flux provided by the test facility $(neutrons/(cm^2 \cdot s)$ by the effective execution time. The cross-section (cm^2) represents the circuit area that will generate an output error (SDC or DUE) if hit by a particle. The higher the number of computation resources, the higher the cross-section, and the higher the probability of an impinging particle generating an error.



(a) GAP8 Cross section

(b) GAP8 Mean Executions Between Failure (MEBF)

Fig. 3: Figures 3a and 3b shows the cross-sections and the Mean Executions Between Failures (MEBF) extracted from the neutron beam experiments for each evaluated code, respectively. The hatched bars represent the average cross-section and MEBF, respectively. It is worth noting that the MEBF is calculated using the execution time provided by GAP8 SDK.

As present in previous works, the cross-section alone does not contain information on the execution time [31], [32]. Thus, to better evaluate the reliability of tested device and codes, we need to correlate error rates with performance. Hence, we evaluate the Mean Executions Between Failures (MEBF) rate for each tested code. The MEBF is defined as the number of correct executions completed before experiencing a failure [32]. We calculate the MEBF by dividing the Mean Time Between Failures (MTBF) rate by the code's execution time. The MTBF is the inverse of the cross-section multiplied by the terrestrial flux ($1/(\sigma \times flux_{terrestrial})$). The MEBF rate is then directly proportional to the tested setup resilience. A higher MEBF rate means more results could be computed without experiencing errors.

IV. Beam experiments results

In this section, we present the results of the first evaluation of a commercial parallel ultra-low-power processing processor based on the RISC-V architecture. We compare the cross-section of the various evaluated codes and the Mean Execution Between Failures (MEBF). Additionally, we discuss the erroneous execution outcomes observed in the beam experiments for each code.

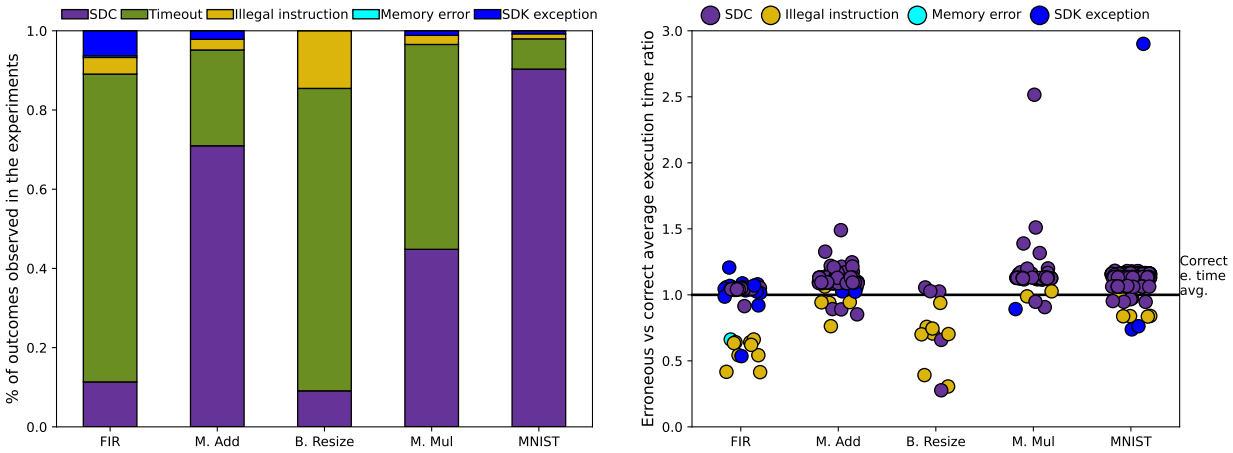
A. GAP8 error rate

Figure 3a shows the experimental cross-section measured in the beam experiments (y-axis) for the five codes we evaluated and the average cross-section. We present the error rate split into two categories, Silent Data Corruption (SDC) and Detected Unrecoverable Error (DUE).

The results show that the DUE error rate varies less than the SDC rate for the evaluated configurations. The highest DUE rate variations come from FIR (72% higher than the average) and MNIST (54% lower than the average). The FIR code is split into steps interleaved with operations sequentially performed by the master core. The majority of FIR execution is composed of sequential code and managing memory on the main core, leading to less efficient utilization of GAP8 resources. Consequently, an incorrect synchronization between the cluster of cores and the main core controller is expected to generate a DUE. On the contrary, MNIST CNN is highly optimized to extract the maximum performance of the cluster of cores and reduces any unnecessary management performed by the main core controller and reduce the Direct Memory Access (DMA) operations, leading to a lower DUE rate than the other codes. FIR, Bilinear Resize, and Matrix Mul operate over large arrays, which leads to more DMA requests and synchronization actions performed by the main core, increasing the DUE rate.

Figure 3a shows that the SDC rate is directly related to how the evaluated codes use the functional units. As most of the GAP8 resources are underutilized during FIR execution, the SDC rate is expected to be much lower than the average ($4.5\times$ lower than the average SDC rate). The Matrix Addition performs the summation of two linear arrays, which allows the code to be highly parallelized and equally distributed among the cores of the cluster. An error in a core's functional unit or an unprotected memory resource is likely to generate an SDC. On the contrary, Bilinear Resize and Matrix Multiplication operations can also stress the GAP8 computing units. However, to perform the calculations on large memory arrays that occupy or exceed the L1 memory (See Table I), a high number of memory operations is required, which reduces the usage of the functional units compared to the other codes. As a result, the SDC rate is reduced.

The MNIST CNN stresses all the computational and memory resources of the GAP8 platform, leading to the highest observed error rate ($3.2\times$ the average SDC rate). When used for image classification, CNN produces a tensor of values representing the probabilities of classified objects in the frame. Then, after the last layer of the CNN, the probability values



(a) (b)

Fig. 4: Figure 4a shows the erroneous executions outcomes percentages. Figure 4b shows the execution time ratio between the faulty and the fault-free execution time. Timeout executions are not shown as they are always higher than $3\times$ the expected execution time. The execution time is measured considering the SDK device setting step.

will be ranked, and the highest value will be selected. Although a fault can propagate to the last layer of the CNN, it may not change the classification.

To evaluate the criticality of the SDCs on MNIST CNN we separate the SDC into *Tolerable SDCs* (a SDC that does not change the classification) and *Critical SDCs* (a SDC that changes the classification). In our experiments, 96.12% of the SDCs would be classified as *Tolerable SDCs*. Our findings are aligned with previously published works that measured the criticality of the errors on GPUs and TPUs [14], [33], [34]. In fact, it has been demonstrated that CNNs can support variations on the output values and still produce the correct classification [33], [34]. However, Critical SDCs are a non-neglectable threat, especially for safety-critical and mission-critical systems. Therefore, such errors should be identified and analyzed in order to design hardened techniques for low-power devices, like the GAP8 platform, to at least detect critical errors that may undermine the system's reliability.

B. Mean Executions Between Failures

The cross-section is a metric that considers the probability of error given the resources used by the application. However, the cross-section does not give us any insight into how the performance can impact the reliability. Thus, we must use the MEBF to analyze the application's reliability and performance impact. Figure 3b shows the MEBF for the evaluated codes. It is worth noting that we calculated the MEBF using the sum of the SDC and DUE cross-section for this evaluation. Then, we can evaluate the MEBF for all the radiation induced-events on the GAP8 platform.

Regarding the MEBF, the differences between the evaluated codes change compared to the cross-section evaluation. MNIST CNN has the highest error rate and, consequentially, even if the code does not have the highest execution time, it has the lowest MEBF ($2.2x10^{10}$ MEBF) among the executed codes. Contrarily, Matrix Multiplication has neither the highest error rate nor highest execution time, leading to the highest MEBF ($8.2x10^{10}$ MEBF). It is worth noting that even if Bilinear Resize does not have a high cross-section, it has an MEBF that is only 40% higher than MNIST due to the high execution time. On the other hand, FIR and Matrix Addition have an MEBF near the average.

The cross-section and the MEBF can be directly used to estimate the device's failure rate using the obtained cross-section multiplied by terrestrial flux (13 $neutrons/(cm^2 \times h)$). If we consider the average cross-section from Figure 3a, the sum of SDC and DUE cross-sections ($\approx 1.34x10^{-9}$) would yield a failure after each $\approx 6.66x10^7$ hours of operation. However, this analysis is valid only when one operating device is considered. The scenario changes when we extend the analysis to multiple devices working in parallel. When considering a million GAP8 based-systems executing simultaneously, the interval of faults would be reduced to 66.6 hours, i.e., one fault after each ≈2.8 days. This is an alarming result, since many IoT applications on smart home appliances are expected to be in charge of controlling critical systems, such as heating and cooling systems and high voltage mechanisms.

C. Analysis of the incorrect execution outcomes

Thanks to the setup presented in Section III, we can distinguish the outcomes of each incorrect execution (i.e., executions that do not finish and/or produce an incorrect output). Figure 4a shows the percentages of each outcome for each evaluated code in the vertical axis, i.e., SDC, Timeout, Illegal Instruction, Memory error, and SDK exception.

The executions that finished with SDCs or Timeout are the majority of the incorrect executions observed in the experiments. On average, the sum of SDCs and Timeout represents 92% of the incorrect executions. In fact, the probability of a loop control variable, stored in a memory or a register, is

modified and thus leads to a Timeout execution, is higher for codes with more instructions such as branch, memory, and synchronization between main and cluster cores, such as FIR and Bilinear Resize. Furthermore, for FIR and Bilinear Resize codes, more errors from Illegal Instructions, Memory errors, and SDK exceptions are observed compared to codes composed of more arithmetic instructions, Matrix Add, Matrix Mul, and MNIST CNN.

Additionally, in Figure 4b, we report the ratio between the expected average execution time (without faults) and the execution time when an incorrect execution is observed (vertical axis). We draw a black line crossing the y-axis at 1 to represent each code's error-free average execution time. The most exciting result is that most executions finishing with an SDC have an execution time near the expected average, but some executions are far from the expected execution time. In fact, even if the SDCs are only errors observed at the application's output, the neutron-induced events may perturb the control flow or increase/reduce the loop iterations, generating a mismatch between the expected output and the observed one. Additionally, the algorithm number of iterations modified by the error may reduce or increase the execution time of the code.

V. Conclusions

In this paper, we have evaluated the realistic error rate of the GAP8 RISC-V multicore processor implemented as an ASIC exposed to a flux of neutrons. We have considered five representative codes for our experiments, including a quantized Convolutional Neural Network. The error rate of the device extracted from the beam experiments is directly related to how the code uses the processor resources. That is, codes that have to perform more synchronization actions between the main core and the cluster of core and more memory operations, like FIR, have a higher DUE rate. Contrarily, more compute-intensive codes that stress more the functional units and have fewer memory instructions, like the MNIST CNN, have a higher SDC rate.

Using GAP8 SDK, we have been able to trace the outcomes and the execution time of the incorrect executions observed in the radiation experiments. The results show that a fault can reduce or increase the execution time of an instance of a code that finished with SDC. These results can be used to tune the building and deployment of smart cities, HPC, and safety-critical applications.

Even if the beam experiments are the most realistic way to evaluate an electronic device, they still lack fault propagation visibility. In the future, we will inject faults in GAP8 microarchitecture to simulate and evaluate fault propagation. Thanks to RISC-V open-source ISA, we can perform realistic and exhaustive fault simulations in different levels of abstraction, such as RTL models and cycle-accurate simulators. Then, the evaluation of the fault propagation will be an essential step in proposing fault tolerance for parallel multicore RISC-V processors.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 899546 and with the support of the Brittany Region, and partially funded by ANR FASY (ANR-21-CE25-0008-01) and ANR RE-TRUSTING (ANR-21-CE24-0015-02). Neutron beam time was provided by ChipIR (DOI:10.5286/ISIS.E.RB2200004-1). We acknowledge the researchers that supported and helped with the neutron experiments, Dr. Paolo Rech, Dr. Christopher Frost, and Dr. Carlo Cazzaniga.

References

[1] L. Lu, M. Zhang, and D. He, "Design and implementation of a smart home system based on the risc-v processor," in *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, 2020, pp. 300–304.

[2] F. Ficarelli, A. Bartolini, E. Parisi, F. Beneventi, F. Barchi, D. Gregori, F. Magugliani, M. Cicala, C. Gianfreda, D. Cesarini, A. Acquaviva, and L. Benini, "Meet monte cimone: Exploring risc-v high performance compute clusters," in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, ser. CF '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 207–208. [Online]. Available: https://doi.org/10.1145/3528416.3530869

[3] D. A. Santos, L. M. Luza, C. A. Zeferino, L. Dilillo, and D. R. Melo, "A low-cost fault-tolerant risc-v processor for space systems," in *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2020, pp. 1–5.

[4] A. Ruospo, R. Cantoro, E. Sanchez, P. D. Schiavone, A. Garofalo, and L. Benini, "On-line testing for autonomous systems driven by risc-v processor design verification," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2019, pp. 1–6.

[5] J. C. Laprie, "Dependable computing and fault tolerance : Concepts and terminology," in *Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years., Twenty-Fifth International Symposium on*, Jun 1995, pp. 2–.

[6] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 86–94.

[7] R. Baumann, "Soft errors in advanced computer systems," *2005 IEEE Design Test of Computers*, 2005.

[8] G. P. Dávila, D. Oliveira, P. Navaux, and P. Rech, "Identifying the most reliable collaborative workload distribution in heterogeneous devices," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1325–1330.

[9] D. A. G. Goncalves de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 791–804, 2016.

[10] J. M. Badia, G. Leon, J. A. Belloch, M. Garcia-Valderas, A. Lindoso, and L. Entrena, "Comparison of parallel implementation strategies in gpu-accelerated system-on-chip under proton irradiation," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2021.

[11] K. Ito, Y. Zhang, H. Itsuji, T. Uezono, T. Toba, and M. Hashimoto, "Analyzing due errors on gpus with neutron irradiation test and fault injection to control flow," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1668–1674, 2021.

[12] H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, "Radiation-induced multi-bit upsets in sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2455–2461, 2005.

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

[14] R. L. Rech Junior, S. Malde, C. Cazzaniga, M. Kastriotou, M. Letiche, C. Frost, and P. Rech, "High energy and thermal neutron sensitivity of google tensor processing units," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 567–575, 2022.

[15] A. E. Wilson and M. Wirthlin, "Neutron radiation testing of fault tolerant risc-v soft processor on xilinx sram-based fpgas," in *2019 IEEE Space Computing Conference (SCC)*, 2019, pp. 25–32.

[16] A. B. de Oliveira, L. A. Tambara, F. Benevenuti, L. A. C. Benites, N. Added, V. A. P. Aguiar, N. H. Medina, M. A. G. Silveira, and F. L. Kastensmidt, "Evaluating soft core risc-v processor in sram-based fpga under radiation effects," *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1503–1510, 2020.

[17] D. A. Santos, L. M. Luza, M. Kastriotou, C. Cazzaniga, C. A. Zeferino, D. R. Melo, and L. Dilillo, "Characterization of a risc-v system-on-chip under neutron radiation," in *2021 16th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1–6.

[18] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "Gap-8: A risc-v soc for ai at the edge of the iot," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2018, pp. 1–4.

[19] P. Bodmann, G. Papadimitriou, R. L. Rech Junior, D. Gizopoulos, and P. Rech, "Soft error effects on arm microprocessors: Early estimations vs. chip measurements," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[20] R. Naseer, Y. Boulghassoul, J. Draper, S. DasGupta, and A. Witulski, "Critical charge characterization for soft error rate modeling in 90nm sram," in *2007 IEEE International Symposium on Circuits and Systems*, 2007, pp. 1879–1882.

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

[22] M. B. Sullivan, N. Saxena, M. O'Connor, D. Lee, P. Racunas, S. Hukerikar, T. Tsai, S. K. S. Hari, and S. W. Keckler, *Characterizing And Mitigating Soft Errors in GPU DRAM*. New York, NY, USA: Association for Computing Machinery, 2021, p. 641–653. [Online]. Available: https://doi.org/10.1145/3466752.3480111

[23] J. Andersson, "Development of a noel-v risc-v soc targeting space applications," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 66–67.

[24] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," JEDEC Standard, Tech. Rep. JESD89A, 2006.

[25] M. Peña Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in arm microprocessors," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, 2019.

[26] B. Du and L. Sterpone, "Online monitoring soft errors in reconfigurable fpga during radiation test," in *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2017, pp. 1–5.

[27] F. F. dos Santos, S. Malde, C. Cazzaniga, C. Frost, L. Carro, and P. Rech, "Experimental findings on the sources of detected unrecoverable errors in gpus," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 436–443, 2022.

[28] H. Quinn, "Challenges in testing complex systems," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 766–786, 2014.

[29] 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.

[30] 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. [Online]. Available: https://doi.org/10.1088/1742-6596/1021/1/012037

[31] G. Reis, J. Chang, N. Vachharajani, and S. Mukherjee, "Design and evaluation of hybrid fault-detection systems," in *Proceedings of the 2005 International Symposium on Computer Architecture, ISCA'05*. IEEE Press, 2005, pp. 148–159.

[32] 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.

[33] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "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*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126964

[34] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2019.