Combining Fault Simulation and Beam Data for CNN Error Rate Estimation on RISC-V Commercial Platforms

Fernando Fernandes dos Santos, Marcello Traiola, Angeliki Kritikakou

Combining Fault Simulation and Beam Data for CNN Error Rate Estimation on RISC-V Commercial Platforms

Fernando Fernandes dos Santos, Marcello Traiola, and Angeliki Kritikakou
Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France

***Abstract*—Thanks to the RISC-V open-source Instruction Set Architecture, researchers and developers can efficiently propose new solutions at a low cost and low power consumption. RISC-V-based architectures can then be customized to run Machine Learning (ML) algorithms efficiently and inserted in safety and mission-critical domains, where the execution must be reliable. However, a fault in the hardware resources can compromise the system's ability to operate correctly. Thus, it is necessary to characterize the ML applications' vulnerabilities on RISC-V processors and how errors in those operations impact the Convolutional Neural Network (CNN) misclassification rate. In this research paper, we assess the error rate induced by neutrons on the basic operations of a CNN running on a RISC-V-based processor (GAP8) and how each operation contributes to the entire CNN error rate. Our findings indicate that memory errors are the primary contributors to the system's error rate. Furthermore, we present a case study demonstrating how the CNN microbenchmarks can be used to estimate the error rate of an entire CNN. By combining data from fault simulation and beam experiments, our error rate estimation led to a result that closely matches those obtained solely from beam experiments.**

I. Introduction

Recent advances in ML algorithms, such as quantization, reduced precision training, and weight pruning, enabled tiny CNN models to be adopted in low-power consumption processors and accelerators, such as TPUs, tiny ARM CPUs, and RISC-V SoCs. Compared to other architectures, RISC-V processors have the advantage of implementing an open-source Instruction Set Architecture (ISA), allowing designers to propose new customized architectures with smaller non-recurring engineering costs. RISC-V processors are today adopted in several domains, including end-user applications [1], High-Performance Computing (HPC) [2], and safety-critical applications [3], [4]. Such a market trend became a promising option for CNN-based safety-critical applications where power consumption, real-time execution, and reliability are mandatory.

Researchers have been focused on improving CNN's performance and power consumption on recent RISC-V architectures [5], [6]. However, for RISC-V processors to be employed on safety-critical applications, their reliability must be thoroughly characterized to define how faults can impact the system's correct functioning. Faults that disrupt a system's operation can be generated by different events, such as environmental perturbations, ionizing radiation, software errors, process, temperature, or voltage variations [7], [8]. Ionizing radiation is particularly dangerous for safety-critical applications as it leads to very high error rates [9]. Terrestrial neutrons can perturb the state of a transistor or change memory cells' values, leading to soft errors. Note that the device is not permanently damaged (a new operation or memory write will replace the faulty value), but the error may propagate to the output and possibly lead to critical errors.

To evaluate a device radiation-induced error rate, researchers perform radiation experiments with CPUs [10], GPUs [11]–[13], FPGAs [14], [15], and Tensor Processing Units [16]. The error rate of soft RISC-V processors synthesized on FPGAs, or ASICs, has been measured when exposed to heavy ions and neutrons [17]–[20]. In this paper, we enrich the state of the art by not only evaluating the error rate of an ASIC RISC-V processor but also performing a detailed reliability evaluation of the most basic CNNs operations on a Parallel Ultra Low Power (PULP) RISC-V System on Chip (SoC), GAP8 from GreenWaves [5] (**Section IV**). Moreover, we perform fault simulation and correlate the results with beam experiments' results (**Section V**). We first estimate the error probability of each operation as the product of its hardware cross-section – obtained from the beam experiments – and the fault propagation probability (or Architectural Vulnerability Factor, AVF) – obtained from fault simulation (**Section V-A**). Finally, we estimate the cross-section of the whole code running on the device by summing all operations' error probabilities (**Section V-B**).

GAP8 includes an 8-cluster core that supports standard RISC-V instructions and a set of Single Instruction Multiple Data (SIMD) instructions. SIMD instructions are extremely interesting for CNNs as they can improve performance without an excessive increase in power consumption. We expose the GAP8 to a neutron beam and measure its error rate while running the main CNNs operations (Convolution, Fully connected Linear, and MaxPooling). As CNNs are both computationally- and memory-demanding and GAP8 does not have memory error protection, we evaluate the error rate and criticality of the main GAP8's memories. We discuss how memory errors can impact the error rate of the main CNN operations.

II. Radiation effects on RISC-V devices

Terrestrial high-energy neutrons that interact with the hardware may generate soft errors in the system. The striking

Contact e-mail:{fernando.fernandes-dos-santos, marcello.traiola, angeliki.kritikakou}@inria.fr

TABLE I: Characteristics for the evaluated codes on GAP8 RISCV (memory and CNN).

	SoC usage	Cycles [KB]	Mem size
Linear	8 Cores+Vector	2.2	1024x16 bytes
Maxpool	8 Cores+Vector	3.8	112x112 bytes
Convolution	8 Cores+Vector	22.8	112x112+5x5 bytes
Mem. L1	1 Core	–	62KB
Mem. L2	1 Core	–	448KB
CNN	8 Cores+Vector	1013.9	293KB

particle can generate single or multiple-bit flips on a memory resource, such as caches, registers, or buffers, or even corrupt the value of a functional unit inside a processing core. If the incorrect value is used as part of the algorithm computation, the error will be propagated through the code running on the device. At the application level, the incorrect value may lead to the following outcomes: **No effect on the program output**: the fault is masked, and the program output is unaffected. **Silent Data Corruption (SDC)**: the program finishes, but the output is incorrect, and no error flags are raised. **Detected Unrecoverable Error (DUE)**: the system stops working, forcing it to be rebooted or power cycled. A DUE can result from uncorrectable memory events, crashes, or errors generating an infinite loop (program hangs).

Prior works have studied the reliability of RISC-V cores synthesized on FPGAs with neutrons and heavy ions experiments [17]–[19]. As many RISC-V processors share an open-source concept, developers can modify the architecture to apply Full or Partial Modular Redundancy to increase the processor reliability and evaluate the hardened architecture on beam experiments [17], [21]. Commercial RISC-V processor's error rate and criticality have been characterized on a neutron beam experiment to determine the impact of neutron-induced faults on a set of applications [18], [20], [22]. However, none of the prior works have focused the analysis on common CNN layers and used this information to estimate the cross-section of a CNN. In this paper, we propose three main contributions:

- we evaluate the most basic CNN operations running on an ASIC commercial multicore RISC-V platform (GAP8) on a neutron beam and extract the realistic cross-section;
- we measure the memory cross-section and fault model of the main GAP8 memories and correlate the results with the cross-section observed on CNN operations.
- we perform fault simulation and combine the results with data from neutron beam experiments for the basic CNN operations and memory. By combining data from both neutron beam and fault simulations, we are able to estimate the cross-section with reasonable accuracy.

III. Metrics and Evaluation Methodology

In this Section, we describe the beam experiments methodology for measuring GAP8 RISC-V SoC's cross-section, the codes we evaluate, and the metrics we use to support the evaluation in the results section.

A. Device under test and evaluated codes

Device Under Test: For the beam experiments, we consider a commercial RISC-V SoC named GAP8. GAP8 is a multicore RISC-V platform from GreenWaves technologies based on PULP architecture. GAP8 is built with $55nm$ TSMC 55LP CMOS technology, and it consists of a cluster of 8 RISC-V cores connected by a Logarithmic Interconnect and a RISC-V Fabric Controler (FC) processor to manage the cluster. The FC core operates at 250MHz, while each cluster core operates at 175MHz. The FC core also has 16KB of data and 1KB of instruction cache. Both the FC and the cluster can access an L1 memory of 64KB and an L2 memory of 512 KB. Both L1 and L2 are scratchpad memories, i.e., the programmer must allocate and manage the memories. None of the SoC memories have an Error Correction Code (ECC) nor protection against Single Events Upsets. GAP8 supports integer and fixed-point arithmetic. During our experiments, we set the voltage and frequencies of GAP8 SoC to the maximum optimal values according to the GAP8 manual (1.2 volts with 250MHz for the FC and 175MHz for the cluster). This configuration set has a power consumption of 75mW [5].

```
1  /**Memory test code**/
2  while no memory error is observed {
3      /**Set all the memory with AAAAAAAA**/
4      for i in every 4 bytes of the memory {
5          M[i] ⇐ 0xAAAAAAAA
6      }
7      sleep for 1s
8      /**Compare the memory values**/
9      errors ⇐ 0
10     for i in every 4 bytes of the memory {
11         errors += (M[i] != 0xAAAAAAAA)
12     }
13     /**If there is error(s)
14     return to the host and log**/
15     if errors != 0
16         break & log on the host
17 }
18
```

Fig. 1: Behavioral algorithm for L1 and L2 memory test.

Evaluated codes: We focus our analysis on the reliability of the core operations of CNNs. We perform 3 types of experiments, one of which is to characterize the memories, the CNN's most common operations, and a full CNN inference.

1) Memory microbenchmark:

As CNNs are known for being resource-demanding in terms of memory and computational resources, and as GAP8's main memories do not have ECC, we investigate the GAP8's memory cross-section and correlate it with the results for CNN operations. We design a microbenchmark to measure L1 and L2 memories cross-section. Figure 1 depicts a behavioral code for the memory microbenchmark. The code consists of a loop that runs while no memory errors are observed (line 2). The L1 or L2 memories of GAP8 are populated with a specific pattern (chunks of 4 bytes with $0xAA$) (lines 4 to 6). After the writing, the code sleeps for 1s (line 7). Finally, the microbenchmark reads all the memory values and logs on the host if there are errors (lines 9 to 16).The memory microbenchmarks are designed to use as much memory as possible, utilizing 87.5% of L1 memory and 96.8% of L2 memory. Despite our attempts, we could not allocate 100% of the memories for these benchmarks as they kept crashing.

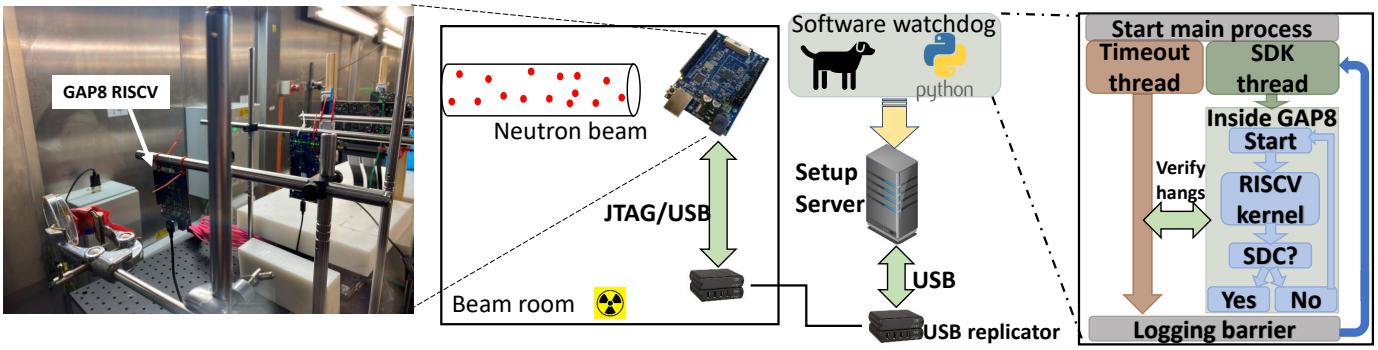

Fig. 2: ChipIR setup overview. The GAP8 is exposed to the neutron beam. The communication is performed through JTAG/USB by a USB replicator. Outside the beam room, a software watchdog server monitors the events inside the SoC.

However, the memory cross-section is calculated per byte, so this limitation does not impact the results.

2) Common CNN operations:

Convolution layer is the main operation performed on a CNN. The selected code consists of a 5×5 filter convoluted into a matrix of 112×112. The Convolution layers also have the characteristics to be the most resource-demanding procedure of a CNN. Consequently, convolution layers have been demonstrated to be one of the most critical parts of CNNs [23]–[25]. As discussed in Section IV, the convolution operation has the highest error rate.

Fully connected Linear layer is the class of algorithm that performs the last step of the inference on a CNN. The standard fully connected neural networks receive input and, based on the neuron's activation, produce an inference that is reduced to a smaller set of values. In our experiments, the Linear layer operates over an input of 1024 bytes and produces 16 values, which correspond to probability values in a real CNN.

MaxPooling layer is a specific ML algorithm explicitly created to avoid over-fitting on CNNs. MaxPooling is placed after a certain number of layers to reduce the amount of data by filtering the values based on a filter block. We selected a MaxPooling layer that applies a 2×2 filter to a matrix of size 112×112. For each 2×2 block, only the highest value is propagated. As proposed by prior works [23]–[25], MaxPooling is expected to mask at least 75% of the faults.

3) 5-layer CNN:

As part of our study, we chose a 5-layer CNN as the baseline, in addition to the basic operations of CNN. We selected this CNN to classify handwritten digits (MNIST dataset) as a representative example of embedded machine learning. The CNN we chose is a quantized (16-bit integer) version of LeNet CNN [26], which comprises two convolutional layers, two MaxPooling layers, and a Linear layer.

B. Physical Fault Injection With Neutron Beam

In order to effectively evaluate the reliability of GAP8, we expose it to a neutron beam and measure its cross-section. Similar to [5], [20], we use the GreenWaves Software Development Toolkit (SDK) to build and run the experiments. Our experiments are performed at the ChipIR facility of the Rutherford Appleton Laboratory, UK. The available neutron flux was about $3.5 \times 10^6 n/(cm^2/s)$, ~8 orders of magnitude higher than the terrestrial flux at sea level [27]. The facility delivers a beam of neutrons with a spectrum of energies that resembles the atmospheric neutron one [28]. As a metric to characterize the error rate, we calculate the *cross-section* by dividing the number of observed errors ($\#errors$) by the received particle fluence (η), i.e., Equation 1. The fluence is obtained by multiplying the average neutron flux of the test facility ($neutrons/(cm^2 \cdot s)$) by the effective code execution time in seconds.

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

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.

Figure 2 shows an overview of the experiments. We created a software watchdog consisting of Python scripts that run on the setup server computer outside the beam room. The watchdog controls GAP8 by monitoring, executing the programs, logging events, and recovering from device hangs. The program is killed and relaunched if it stops responding in a predefined interval. We set each code timeout individually depending on the code execution time, up to $3\times$ the expected execution time. The code inside GAP8 executes the same kernel for a predetermined number of iterations. We set the maximum number of internal SoC iterations to 65KB for a good tradeoff between time wasted with communication and proper kernel executions. After each iteration inside the SoC, the output of the kernel is compared with a constant golden value. If there is a mismatch between both outputs, the main process will log the useful information and restart again. It is worth noting that only errors happening on the kernel output are considered for the error rate analysis.

C. Fault Simulation With GVSoC

The GVSoC is an event-based simulator that simulates all PULP platform instruction set [29]. The GVSoC allows the simulation of the same codes we evaluated under a neutron beam with the exact input sizes and without modifications. We have modified the GVSoC simulator in order to be able

TABLE II: Memory SDC Cross Section observed on the neutron beam experiments. L1 and L2 memories are evaluated.

	Total cross-section	Byte cross-section
L1	$8.13 \times 10^{-9} \pm 7.4 \times 10^{-10}$	$1.28 \times 10^{-13} \pm 1.2 \times 10^{-14}$
L2	$4.21 \times 10^{-8} \pm 3.9 \times 10^{-9}$	$9.18 \times 10^{-14} \pm 8.6 \times 10^{-15}$

to simulate faults at the instruction output (**Inst**) and in memory addresses (**Mem**). The fault simulation has 4 steps: Application profiling, fault site selection, fault simulation, and post-injection.

Application profiling: We have modified the GVSoC simulator to enable generating tracing for each instruction executed and memory address written. During the profiling stage for instruction injection, we store information such as the instruction label, the core where the instruction is executed, and the output register. For memory fault injection, we record each written memory address during the simulation. After the profiling is finished, the traces will be saved in a file for the fault site selection.

Fault site selection: After extracting profiling data, we choose a subset of fault sites to simulate. These sites are randomly selected from the traces using a uniform distribution of fault sites. For each fault model (Inst and Mem), we simulated 10,000 for the MNIST CNN, 20,000 in total.

Fault simulation: For instruction output injection, we modify the output register with a fault mask. The fault mask is generated based on the findings from [30]. Specifically, for 95% of the faults, only one bit in the instruction output is modified by the fault mask, while for the remaining 5%, random bits of the register are flipped. For memory fault injection, we select the target address and then perform SBU/2-MCUs based on the results presented in Section IV. This means that SBUs and 2-MCUs will be injected with the same percentages (and distances for MCUs) observed in the beam experiments for each memory level.

Post injection: After simulating faults on GVSoC, we classify them into SDCs, DUEs, and masked.

The GVSoC fault simulations allow the extraction of the Architectural Vulnerability Factor (AVF) for the CNN operations and the full CNN. The AVF is the probability of a fault propagating to the code output once injected in an instruction output or memory address [31].

IV. CNN Basic Components Cross-section

We start our analysis by presenting the cross-section observed on the memory and basic CNN Operations microbenchmarks. It paves the way for the evaluation of the CNN-based operation codes. We then compare the cross-section of various CNN operations. All results (CNN common operations and memory benchmark) are reported with 95% confidence intervals considering a Poisson distribution.

A. Memory cross-section

Table II shows the L1 and L2 microbenchmarks SDC cross sections observed on the beam experiments. We present the total cross-section that considers the cross-section of the entire evaluated memory (based on the memory sizes from Table I) and the cross-section per byte.

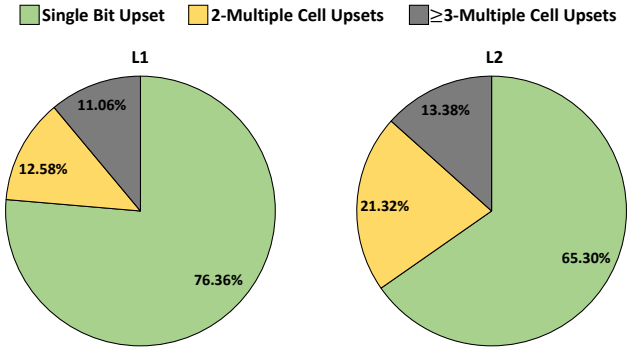

Fig. 3: Distribution of observed Single-Bit Upsets (SBUs) and Multiple-Cell Upsets (MCUs) on L1 and L2 microbenchmarks.

The memory cross-section per byte is similar to values measured in prior works with similar technologies [32], [33]. Although L1 and L2 are similar memories, the L2 cross-section per byte is 29% smaller than the L1 cross-section. L1 and L2 are differently organized on the SoC and, consequently, have different latencies and bus connections. More precisely, L1 is a shared memory among the cores in the cluster, while L2 is a larger memory divided into 4 blocks.

Figure 3 depicts the Single-Bit Upsets (SBUs) and Multiple-Cell Upsets (MCUs) for both L1 and L2 memories. The results show that the SBUs on L1 and L2 memories are the most common events observed in the experiments, i.e., 76.4% and 65.3%, respectively. Contrarily, MCUs have a lower occurrence when compared to SBUs. For L1, 12.6% for 2 MCUs and 11.1% for 3 or more MCUs. For L2, 21.3% for 2 MCUs and 13.4% for 3 or more MCUs.

We observe different MCU patterns for the two GAP8 levels of memory. In 0.9% of the cases for L1 memory, we observe large MCUs with 10 to 18 bits. In 1.1% of the cases for L2, the MCUs have 7 or 9 bits. The maximum size of observed MCU was 18 for L1 and 9 for L2. GAP8 uses a tightly coupled data memory to increase performance and save energy, and, as observed in prior works, the MCUs can be spatially random SRAM memories depending on the cell organization and the access type [32]. A single particle can corrupt many bits of a memory array. Note that no adjacent bits (Multiple Bit Upset, MBU) were observed in a word. As most modern memories have bit-interleaving, the occurrence of MBUs is rare.

While less frequent, MCUs are much harder to be masked on computation and can increase the fault propagation probability by at least 50% [34]. Additionally, it has been shown that the patterns of the MCUs can range in the number of bits flipped [32], [35]. That is, the same particle may modify the value of 16 bits in the memory. Based on the high cross-section and high criticality observed on the GAP8's memory reliability analysis, we expect memory errors to significantly impact the overall code cross-section (details in Section V). In the next section, we show how the code's cross-section changes based on the code's memory usage.

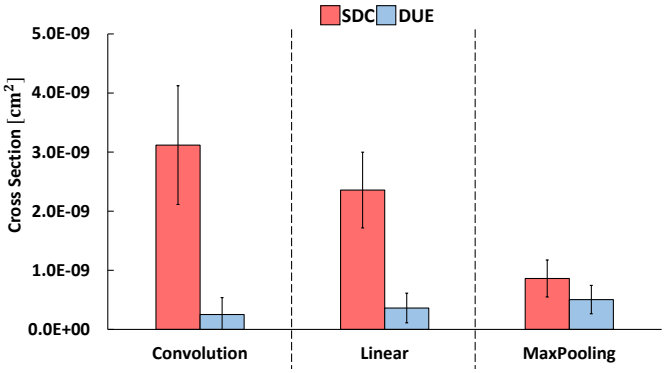

Fig. 4: Experimentally measured Silent Data Corruption (SDC) and Detectable Unrecoverable Errors (DUE) Cross Sections.

B. CNN operations cross-section

Figure 4 depicts the experimentally measured SDC and DUE cross sections. The y-axis shows the cross-section, and the x-axis shows the type of operation (Convolution, Linear, & MaxPooling).

The SDC rate results show that the higher the memory resource usage, the higher the SDC rate. As the Convolutional layer is the most resource-demanding, it has the highest error rate. The same happens for Linear and MaxPooling operations. MaxPooling layers also can mask most of the faults, as 75% of the values are discarded as expected (details Section III).

On the contrary, the DUE cross-sections are much lower than the SDC ones. The SDC cross-section is on average 2.1×10^{-9}, while the DUE cross-section is on average 3.4×10^{-10}. DUEs are caused mainly by events unrelated to arithmetic calculations, e.g., illegal instructions, hangs due to a fault-induced infinity loop, fault-induced deadlocks, incorrect addresses for jump and branch instructions, and illegal memory accesses. Not surprisingly, the highest DUE cross section is the MaxPooling operations (5.1×10^{-10}), which consists of multiple *max* instructions on memory blocks of the input.

V. Cross-section Estimation

In this section, we show how we estimate the entire CNN SDC cross-section with reasonable accuracy by combining the simulation results with the results from beam experiments.

A. AVF

The cross-section of the microbenchmarks was obtained using synthetic versions of the CNN main operations. Although the kernels used are the same as those utilized in actual CNNs, they are not applied to the entire CNN. Additionally, the CNN operations cross-section represents the probability of a fault in the hardware impact on CNN basic operation output. In order to estimate the cross-section of an entire CNN, we need to compute the probability of a fault in one of the CNN layers propagating to the output (i.e., the AVF of the layer). We then computed the AVF for each layer of the 5-layer CNN used in our evaluations (Convolution 1 and 2, MaxPooling 1 and 2, and Linear) and presented the result in Table III.

TABLE III: AVF for each layer per fault model, and the entire CNN. Note that the AVF of each layer was obtained as part of the entire CNN. We injected a single fault at $layer_i$ and then propagated it throughout the entire CNN until its output.

	AVF SDC		AVF DUE		Op. %
	Inst	Mem	Inst	Mem	
Convolution 1	3.56%	2.82%	11.52%	0.30%	1.08%
Maxpool 1	0.17%	0.44%	4.53%	0.46%	49.10%
Convolution 2	27.69%	16.21%	0.03%	0.01%	46.41%
Maxpool 2	0.08%	0.16%	0.01%	0.01%	2.73%
Linear	0.05%	0.14%	0.03%	0.06%	0.68%
Full CNN	51.29%		16.94%		100%

Table III displays the AVF of each layer during a complete inference of the CNN. The table also includes the memory usage of each layer, including input, output, filters, and bias, as a reference. To calculate the AVF, we injected 10,000 faults for each fault model, i.e., Inst and Mem.

The SDC AVF is directly proportional to the size of the layer and the operation being performed. That is, Convolutions 1 and 2 have the highest SDC AVF. As seen in the results of the synthetic microbenchmarks cross-section (Fig. 4), MaxPooling and Linear have the highest masking factor, which leads to SDC AVFs ranging from 0.05% to 0.16%.

The DUE AVF is predominantly high on the first layer of the CNN. All the memory allocation and data move instructions execute before the first layer. On GAP8, L1 and L2 memories are scratchpad memories, meaning the program must manage all data. As a result, many instructions are highly sensitive to faults and can lead to DUEs. Although on a lower scale, a similar behavior happens on the Linear layer. Memory deallocation occurs after the Linear layer, increasing the chances of DUEs occurring after the last layer. Finally, for MaxPooling 1, a large portion of addresses are being accessed as it operates over a large portion of the memory, many DUEs are expected to occur on the MaxPooling 1.

B. Cross-section Estimation

The cross-section of a complex code running on a device could be estimated by summing each system component error probability (component hardware cross-section) and the fault propagation probability (AVF). Knowing the AVF and cross-section of every resource used for computation, in principle, would allow a perfect estimation of the cross-section of a code. However, in complex systems, the number of resources to be evaluated, even in a small RISCV core like GAP8, is considerably high. Consequently, it would be infeasible to measure each resource cross-section and AVF given that the devices today integrate many resources on a single chip.

According to recent research [36]–[39], a reasonable estimate of a complex system's error rate can be obtained by analyzing the error rate contributions of its main functional units and memories. However, characterizing many functional units can be daunting, so *we simplified the estimation process by focusing on the most basic operations of CNNs, i.e., Convolution, MaxPooling, and Linear*. As presented in Sections IV and V-A, we have determined the hardware fault probability (CNN operation cross-section) and propagation for each of

these operations (CNN operation AVF). Using this information, we estimate a complete code's cross-section ($\hat{\sigma}_{[cm^2]}$) by summing the expected contribution of each CNN operation error probability $P(E_{op_i})$ and each CNN operation memory error probability $P(E_{MEM_{op_i}})$, as shown in Equation 2.

$$\hat{\sigma}_{[cm^2]} = \sum_{i=1}^{n} P(E_{op_i}) + \sum_{i=1}^{m} P(E_{MEM_{op_i}}) \quad (2)$$

The contributions to the cross-section, such as $P(E_{op_i})$ and $P(E_{MEM_{op_i}})$, depend on various factors, such as the number of resources used for computation, the probability of generating a fault (the resource cross-section) and the likelihood of that fault affecting the computation (AVF). This relationship is formalized in Equation 3.

$$P(E_{op_i}) = \bar{S}_{op_i} \cdot AVF_{op_i} \cdot \sigma_{op_i} \quad (3)$$

Where $\bar{S}_{op_i}$ denotes the % of operations of a particular layer, AVF_{op_i} represents the AVF of the layer, and σ_{op_i} is the cross-section of the microbenchmark. It should be noted that the same approach is taken to calculate $P(E_{MEM_{op_i}})$, with the only difference being that $\bar{S}_{op_i}$ is replaced by $Mem_{byte_{size}}$, and AVF_{op_i} corresponds to the memory injections AVF, and σ_{op_i} is the memory byte cross-section.

Figure 5 shows the estimated cross-section considering the contributions from the Memory (only the right side of Equation 2), CNN operations (only the left side of Equation 2), and on CNN Operations + Memory (the whole Equation 2).

The SDC cross-section estimated from the memory contribution is 20.1% higher than the measured beam cross-section. This outcome is expected given that the memory has a high SDC cross-section, as detailed in Section IV, and the AVF of injections on memory in the main CNN layers is also high. In contrast, predictions based on CNN operations yield a SDC cross-section that is 89% of the measured beam cross-section. Compared to the memory demands of CNN, the functional units occupy less space, resulting in a smaller SDC cross-section. Ultimately, when the CNN operations and memory contributions are considered, the estimated cross-section is 30.7% higher than that measured with the beam.

We compared the SDCs that caused misclassification in the beam experiments and fault simulations. The results show that 2.27% of the SDCs caused misclassification in the beam experiments, while 1.84% caused misclassification in the GVSoC FI simulations. The GVSoC FI misclassification rate was 18.9% less than the beam experiment results. The misclassification rate depends on various factors, like the type of faults and how they affect the CNN weights and layer outputs. Recent research has shown that the fault space increases as the size of the CNN increases [40]. Although it may not be practical to simulate all necessary faults, our estimation with GVSoC FI, using a reduced number of faults (10,000 per fault model), generated a misclassification rate reasonably close to the one obtained from the beam experiments.

The CNN op+Memory, CNN Op, and Memory estimations resulted in DUE cross-sections that are 3.27×, 3.49×, and 61.66× lower than the beam DUE cross-section, respectively. It is important to note that DUE sources can extend beyond

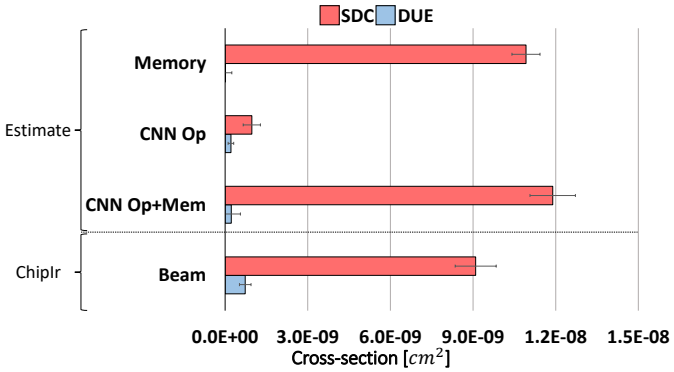

Fig. 5: Comparison of estimated cross-sections with observations from neutron beam experiments

faults injected in instruction output and memory values. In the fault injection analysis, 95% of the DUEs were attributed to invalid memory accesses in the CNN model. In contrast, in the beam experiments, most DUEs resulted from system hangs (i.e., the device stopped responding). Such events pose a significant challenge to simulate in GVSoC, an event-based simulator with a constrained level of detail. This discrepancy in simulation accuracy for DUEs has also been highlighted in recent studies on CPUs [37], [38] and GPUs [36], with observed differences reaching up to 62× for CPUs [37] and 600× for GPUs [36].

VI. Conclusions

We have discussed how faults affect the core operations of CNNs on a commercial RISC-V SoC when exposed to a beam of neutrons. We evaluated the cross-section of the main memories of GAP8 RISC-V to investigate the impact of unprotected memories on the code's reliability. After evaluating the CNN operations and memories, we used an instruction set simulator to perform fault simulations.

To combine the data from the beam experiments and the fault simulation, we proposed an approximation method to estimate the cross-section of an entire CNN. Our results showed that our approximation method is reasonably close to the expected cross-section even when using the CNN layers as basic blocks in the estimation calculation. As a future direction, we will measure the cross-section of more deep neural networks to measure how much our idea can be generalized.

Acknowledgment

This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 899546 with the support of the Brittany Region and partially funded by RADNEXT [41] under grant No 101008126. It is funded by ANR FASY (ANR-21-CE25-0008-01) and ANR RE-TRUSTING (ANR-21-CE24-0015-02). ChipIR provided beam time (DOI 10.5286/ISIS.E.RB2220502). We acknowledge the researchers who helped with neutron experiments, Drs. Christopher Frost, Maria Kastriotou, and 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] 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.

[6] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "Pulp-nn: accelerating quantized neural networks on parallel ultra-low-power risc-v processors," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2164, p. 20190155, 2020. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2019.0155

[7] 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–.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] F. F. Dos Santos, A. Kritikakou, and O. Sentieys, "Experimental evaluation of neutron-induced errors on a multicore risc-v platform," in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 1–7.

[21] A. E. Wilson, M. Wirthlin, and N. G. Baker, "Neutron radiation testing of risc-v tmr soft processors on sram-based fpgas," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2023.

[22] L. A. Tambara, J. Andersson, Á. B. de Oliveira, F. Abouzeid, M. Hjorth, and P. Roche, "Radiation evaluation of leon5ft/noel-vft demonstrator on stm 28nm-fdsoi technology," in *2023 European Data Handling & Data Processing Conference (EDHPC)*. IEEE, 2023, pp. 1–6.

[23] 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

[24] 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.

[25] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin, and J. Brunhaver, "How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on fpgas," *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 865–872, 2021.

[26] 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.

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

[28] 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

[29] N. Bruschi, G. Haugou, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "Gvsoc: a highly configurable, fast and accurate full-platform simulator for risc-v based iot processors," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*. IEEE, 2021, pp. 409–416.

[30] A. Kritikakou, O. Sentieys, G. Hubert, Y. Helen, J.-F. Coulon, and P. Deroux-Dauphin, "Flodam: cross-layer reliability analysis flow for complex hardware designs," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 819–824.

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

[32] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *2011 International Reliability Physics Symposium*, 2011, pp. 5B.4.1–5B.4.7.

[33] Xilinx, "Device reliability report second half 2021 (ug116)," June 2021. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/user_guides/ug116.pdf

[34] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets vulnerability analysis of modern microprocessors," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 119–130.

[35] J. Suh, M. Annavaram, and M. Dubois, "Macau: A markov model for reliability evaluations of caches under single-bit and multi-bit upsets," in *IEEE International Symposium on High-Performance Comp Architecture*, 2012, pp. 1–12.

[36] F. F. dos Santos, S. K. S. Hari, P. M. Basso, L. Carro, and P. Rech, "Demystifying gpu reliability: comparing and combining beam experiments, fault simulation, and profiling," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2021, pp. 289–298.

[37] P. R. Bodmann, G. Papadimitriou, R. L. R. Junior, D. Gizopoulos, and P. Rech, "Soft error effects on arm microprocessors: Early estimations versus chip measurements," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2358–2369, 2021.

[38] P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "The impact of soc integration and os deployment on the reliability of arm processors," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2021, pp. 223–225.

[39] K. Ito, H. Itsuji, T. Uezono, T. Toba, M. Itoh, and M. Hashimoto, "Constructing application-level gpu error rate model with neutron irradiation experiment," in *2022 22nd European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, 2022, pp. 1–6.

[40] J. Guerrero, M. S. Reorda, and J. Aribido, "Assessing convolutional neural networks reliability through statistical fault injections," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[41] R. G. Alía, A. Coronetti, K. Bilko, M. Cecchetto, G. Datzmann, S. Fiore, and S. Girard, "Heavy ion energy deposition and see intercomparison within the radnext irradiation facility network," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2023.