Understanding and Improving GPUs' Reliability Combining Beam Experiments with Fault Simulation

Fernando Fernandes dos Santos
Univ. Rennes, INRIA, France
fernando.fernandes-dos-santos@inria.fr

Luigi Carro
Federal University of Rio Grande do Sul, Brazil
carro@inf.ufrgs.br

Paolo Rech
University of Trento, Italy
paolo.rech@unitn.it

***Abstract*—Graphics Processing Units (GPUs) are being employed in High Performance Computing (HPC) and safety-critical applications, such as autonomous vehicles. This market shift led to significant improvements in the programming frameworks and performance evaluation tools and concerns about their reliability. GPU reliability evaluation is extremely challenging due to the parallel nature and high complexity of GPU architectures. We conducted the first cross-layer GPU reliability evaluation to unveil (and mitigate) GPU vulnerabilities. The proposed evaluation is achieved by comparing and combining extensive high-energy neutron beam experiments, massive fault simulation campaigns at both Register-Transfer Level (RTL) and software levels, and application profiling. Based on this extensive and detailed analysis, a novel accurate methodology to accurately estimate GPUs application FIT rate is proposed. Moreover, by employing the knowledge obtained from the cross-layer reliability evaluation, two novel hardening solutions for HPC and safety-critical applications are proposed: (1) Reduced Precision Duplication With Comparison (RP-DWC), which executes a redundant copy in a reduced precision. RP-DWC delivers excellent fault coverage, up to 86%, with minimal execution time and energy consumption overheads (13% and 24%, respectively). (2) Dedicated software solutions for hardening Convolutional Neural Networks (CNNs) that can correct up to 98% of the CNN errors.**

I. Motivation & Problems addressed

GPUs have evolved from supporting hardware for user applications and graphics rendering to general-purpose accelerators extensively employed in HPC and safety-critical applications such as autonomous vehicles and aerospace markets. The highly parallel architecture of GPUs, in fact, perfectly fits the computational characteristic of most HPC codes and is incredibly efficient in executing matrix multiplication, which is the computing core of CNNs used to detect objects in autonomous vehicles. The most recent GPU architecture advances, such as tensor core and mixed-precision functional units, move toward improving the architecture performances and software flexibility for HPC and deep learning applications.

The market shift of GPUs, from consumer to HPC and safety-critical applications, has suddenly triggered intensive research for GPU hardening that improves GPUs reliability while maintaining high performances. Techniques to improve GPU reliability have been proposed at different levels of the GPU hardware/software, such as in the memory cell [1], Error Correction Code (ECC) [2], [3], and Redundant Multithreading execution [4]. GPU's reliability has become so essential that GPU vendors are working on designing platforms compliant with strict automotive reliability standards such as the ISO26262 [5], [6].

The leading example of GPU usage for safety-critical applications are CNNs. CNNs can exploit GPU's ability to support data and thread-level parallelism while delivering high-accuracy inference results. However, researchers have focused on performance while neglecting other critical aspects, particularly reliability. While performance is vital in these applications, reliability needs to be paramount. It is not possible to tradeoff performance for reliability in safety-critical applications. Modern GPUs have available Error Correction Codes (ECCs) to protect single-bit flips on the main memories. Unfortunately, we have demonstrated that the single-bit flips in the memories often lead to tolerable errors (i.e., errors that do not modify the inference results) [7]. Furthermore, as ECC does not prevent critical errors on CNNs (i.e., errors that modify the inference result) from happening, and duplication approaches reduce the CNNs' performances on GPUs to an unacceptable level, they cannot deliver the needed requirements for safety-critical applications. Thus, this thesis proposes efficient hardening methodologies for CNNs on GPUs.

To be able to evaluate safety-critical and HPC applications reliability on GPUs, the research community has been carefully employing both fault-injection [8]–[13] and beam experiments [3], [14]–[16]. While beam experiments provide a realistic analysis but lack fault propagation visibility, fault simulation allows complete observation of the fault propagation, but it is limited to a subset of the user-accessible resources. Therefore, combining data from beam experiments and fault simulation is an essential missing piece in the GPU reliability evaluation puzzle this thesis intends to find. Additionally, we take advantage of beam experiments and fault simulations to thoroughly validate the proposed new hardening techniques. In the beam and fault simulation experiments, we have considered many novel architectural and software solutions introduced in GPUs, such as reduced precision instructions, GPU dynamic parallelism, high-level machine learning frameworks, and multiple NVIDIA libraries and tools.

In order to **understand** and **improve** the GPU's reliability the following topics are covered in this Ph.D. work:

- Combining beam experiments and fault simulation to deeply understand GPUs' reliability. This thesis presents a detailed comparison between the GPU's Failure In Time

(FIT) rate measured with beam experiments and the FIT rate estimated using fault simulation and kernel profiling.
- This thesis improves the knowledge of GPU's reliability by performing different levels of fault injections (neutron beams, RTL, and software level). Additionally, for the first time for GPUs, a fine grain RTL fault injection (using FlexGripPlus) is combined with the flexibility and efficiency of software fault injection in real GPUs.
- This work advances GPU reliability by characterizing how microarchitecture vulnerabilities in a GPU can undermine a CNN's reliability. Based on this analysis, we propose a novel hardening for CNNs. Most previous works focused only on HPC reliability on GPUs.
- By combining all the knowledge from the previous topics, we propose a new hardening approach for mixed-precision architectures. This thesis goes a step forward in the performance efficiency of Duplication With Comparison (DWC) by presenting Reduced-Precision DWC (RP-DWC), an improvement over the traditional DWC, which consists of executing the replica in a lower precision.

II. Scientific and Technological Excellence

This thesis presents an extensive and accurate reliability analysis on GPUs. It includes the data of *1,200* hours of neutron beam in total, more than *400,000* fault simulations at different levels of abstraction (RTL and software), and detailed application profiling. The experimental beam data account for more than $1.3 \cdot 10^7$ years of terrestrial flux exposure. The neutron-induced error rate is presented for a set of representative HPC and Deep learning codes. Up to 11 applications are evaluated, including 3 CNNs (YOLO, Faster R-CNN, and Resnet). Some codes have been executed using different data types (integer, float-, single-, or half-precision) to understand the impact of mixed-precision on code's reliability. Complementary experiments are presented with ECC ON and OFF to evaluate the efficacy of GPUs built-in reliability solutions and distinguish between the contribution of logic and memory faults to the codes error rate. The FIT rates of the main functional units (including mixed-precision and tensor cores), register file, and shared memory are presented for Kepler and Volta GPUs in order to achieve a fine-grain study of the GPUs vulnerabilities (Sections II-A and II-B). The obtained data highlights that memory errors are not the most critical for GPUs, and thus efficient hardening solutions should focus on the computing and scheduler resources.

All the data gathered with the extensive beam experiments, detailed GPU codes profiling, and large fault simulation campaigns at different levels of abstraction (RTL and software) enabled a precise and cross-layer analysis of the GPUs vulnerabilities. Then, to merge all the knowledge obtained in each layer of abstraction, a novel methodology that **combines data from beam experiments, fault simulation, and profiling** has been presented, allowing an accurate GPU error rate estimation. This new FIT estimation methodology is validated by comparing the programs' FIT rates measured with beam experiments with the failure rates estimated from fault injections using two NVIDIA fault injectors (SASSIFI [9] and NVBitFI [11]), evaluating at which level and under which assumptions fault simulation can provide a realistic reliability evaluation for GPUs (Section II-B (details at Section II-B).

Based on the knowledge from the multilevel evaluation, correlated with an algorithm analysis, novel experimentally-tuned, efficient and effective, hardening solutions are designed. This thesis advances GPU's reliability by characterizing how microarchitecture vulnerabilities in the hardware can undermine GPU reliability and proposes novel fault tolerance techniques. For instance, with efficient software-level hardening techniques, detecting up to 98% of the SDCs on CNNs executing on a real GPU is possible. In addition, to not be limited to a specific type of algorithm, a more general fault tolerance methodology for HPC has been proposed. The novel hardening takes advantage of mixed-precision GPU hardware and moves a step forward in the performance efficiency of DWC by presenting **RP-DWC**. RP-DWC is an improvement over the traditional DWC approach, which consists of executing the replica in a lower precision. The results show that RP-DWC achieves excellent coverage (up to 86%) with minimal overheads. The time overhead can be as low as 10%, while the energy consumption overhead can be as low as 24% (details at Sections II-C and II-D). RP-DWC builds on ideas from previous works that have proposed approximating the algorithm or hardware for efficient fault tolerance [17]–[20].

A. GPU reliability evaluation

1) Concept and Approach: This thesis's first contribution is evaluating the neutron-induced error rate of two NVIDIA GPUs, Tesla V100 and Tesla K40, Volta and Kepler architectures, respectively. The evaluation of the error rate is performed through neutron beam experiments. A set of 11 codes running on GPUs are exposed to a flux of neutrons. Beam experiments are the most effective way to measure the FIT rate of code running on a computing device. By dividing the number of observed errors by the received particles fluence η ($neutrons/cm^2$) it is possible to calculate the *cross section* ($\sigma[cm^2] = \frac{\#errors}{\eta}$). The cross-section ($cm^2$) represents the circuit area that will generate an output error if hit by a particle. The higher the number of computation resources, the higher the cross-section, and the higher the probability for an impinging particle to generate an error. All the experiments performed for this thesis follow the JEDEC standard [21].

When multiplied with the expected neutron flux at which the device will operate ($13 \times 10^9 neutrons/(cm^2 \cdot h)$ at sea level), the cross-section estimates the realistic error rate, expressed in FIT, i.e., errors per 10^9 hours of operation. To not reveal business-sensitive data, all the FIT rates are reported normalized by a constant factor.

The experiments are performed at the ChipIR facility of the Rutherford Appleton Laboratory, UK, and at the LANSCE facility of the Los Alamos National Laboratory, USA. Figure 1 shows the setup mounted in the ChipIR facility. Both facilities deliver a beam of neutrons with a spectrum of energies that resembles the atmospheric neutron one [22], the probability of

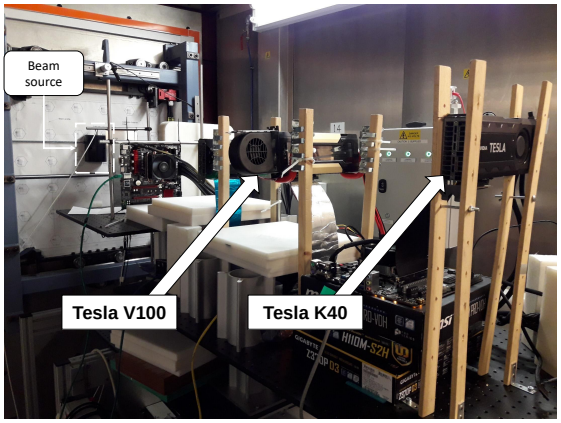


Fig. 1: Experiment setup at ChipIR. More than 1,000 hours of beam experiments are performed for Kepler and Volta GPU architectures. The large beam experiments campaigns allowed obtaining the error rates for 11 HPC codes and 3 CNNs.

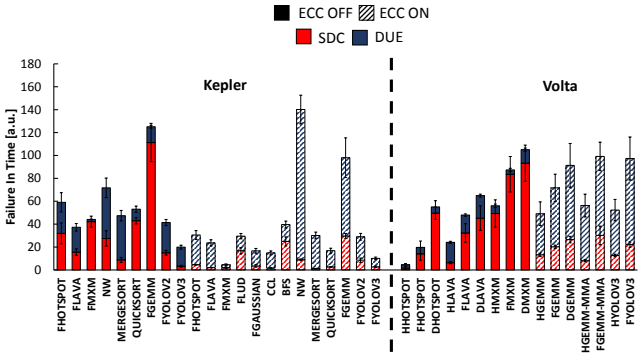


Fig. 2: Normalized FIT rates for Kepler and Volta. It is clear that ECC can reduce the SDC rate while increasing the DUE rate. Additionally, for mixed-precision architectures, the smaller the precision, the lower the overall FIT rate.

generating an error of a neutron produced in the experimental facilities is similar to a terrestrial neutron one.

Neither the cross-section nor the FIT rates depend on the execution time but only on the number of resources used for computation, their sensitivity (fault probability to occur), and criticality (probability for the fault in the resource to affect the calculation). If the same amount of memory is exposed for a given time t or $2 \times t$, its FIT rate will not change. In fact, in $2 \times t$, it is expected twice the error and twice the neutrons ($2 \times t$fluence). Similarly, under the correct assumption that at most one fault can affect the GPU during code execution (the natural flux is very low), executing x *sequential* ADDs or $2 \times x$ sequential ADDs does not change the probability of having one ADD corrupted by neutrons. However, what can change is the probability of the error in one of the ADDs propagating to the output of the sequence of the operations (i.e., the Architecture Vulnerability Factor (AVF)). If the additional x ADDs are executed in *parallel* with the original sequence, the FIT rate is expected to double (same execution time, same fluence, but $2 \times t$ error rate). These observations are used in Section II-B to account for GPU parallelism management in the FIT rate estimation based on fault injection.

2) Obtained results: Figure 2 shows the measured SDC and DUE normalized FIT rates for the GPUs executing the codes with ECC OFF and ON. Values are reported with 95% confidence intervals considering a Poisson distribution.

For Kepler, the average SDC FIT rate with ECC OFF is up to 21× higher than with ECC ON. Not surprisingly, the ECC reduces the SDC FIT rate significantly. For Volta, it is not possible to test the same codes with ECC ON and OFF due to beam time restrictions. The DUE FIT rate increases by up to 5× when ECC is ON. The DUE increase is exacerbated for NW and FGEMM because of the high number of kernel calls (NW calls multiple kernels concurrently) and access to the main memory (FGEMM uses lots of global memory and highly utilizes the memory bandwidth).

Matrix multiplication (naive MxM or optimized GEMM) is Kepler and Volta's code with the highest SDC FIT rate. The SDC FIT rate of matrix multiplication is particularly significant when ECC is OFF (2 to 3× higher than other codes). Matrix multiplication heavily relies on FMA operation, which, according to the data in Section II-B, is among the most vulnerable functional units. Moreover, as the code is easily parallelizable, most GPU functional units are used for computation, exacerbating the fault probability. Additionally, according to data from fault simulation, matrix multiplication has the highest AVF. As a result, the higher FIT rate of matrix multiplication is caused by the use of highly sensitive functional units, the parallel use of most of the available units in parallel, and the high probability for a fault in one unit to affect the result. For CNN's, as YOLOV2 and YOLOV3, more than 75% of the operations are matrix multiplication related [23]. CNN shares with matrix multiplication the problem of using a high amount of the most sensitive functional units.

For 3 CNNs (YOLO, Faster RCNN, and Resnet), we have demonstrated that ECC can reduce one order of magnitude the errors that do not modify the inference result (i.e., *Tolerable errors*) [7]. However, ECC has shown poor performance in correcting errors that modified the inference result in a CNN (i.e., *Critical Errors*). Even with ECC ON, the percentages of critical errors for YOLO, Faster RCNN, and Resnet CNNs are 61%, 25%, and 16%, respectively (details at [7]). We used this information to improve the reliability of CNNs in Section II-C.

The new GPU hardware dedicated to mixed float precisions in all NVIDIA devices after Volta architecture provides outstanding performance for various applications. Hence, for Volta GPU, we focus on comparing the FIT rates of codes executed with different precisions (double, single, and half). This comparison serves as a baseline for future mixed-precision applications. For all the codes, independently of the ECC status, increasing the precision increases the code FIT rate. A higher precision functional unit has a higher area and, thus, a higher probability of being hit by a neutron (see Figure 3). When ECC is OFF, the trend is exacerbated by the fact that higher precision implies a higher number of bits to store data, which has a linear dependence on the FIT rate.

3) Novelty and foundational character: The FIT rates of 11 codes obtained from beam experiments have been presented for HPC and CNNs codes. **Thanks to the large amount of data collected from radiation experiments for NVIDIA GPUs, we are able to observe different behaviors and outcomes not reported before in prior works.** A significant advance in GPU testing and fault injection. All the data presented in this section gives a strong motivation and highlights peculiar vulnerabilities of the GPU architectures to pave the analysis that will be considered in the following sections. Data presented in Figure 2 allowed the observations, such as the ECC impact on the code's reliability and the benefits of using smaller precisions on the error rate. Additionally, the data extracted from the experiments demonstrated that for certain types of applications, such as CNNs, the ECC is not as efficient in preventing critical errors on GPUs. This indicates that memory errors are not as critical as logic errors for CNNs. These observations are only possible thanks to beam experiments performed with real neutron beams.

B. Failure In Time estimation

1) Concept and Approach: We need to consider fault propagation to understand the impact of hidden GPU resources and identify the code/architecture characteristics/metrics that mainly impact the GPU error rate. To fully evaluate the GPU vulnerabilities, different levels of fault injection are performed, physical fault injection on beam experiments, RTL, and software level injections. Based on these experiments, it is possible to determine that for specific codes like CNNs, the critical errors are not generated on the primary memory resources but can be a product of faults in resources such as functional units, parallelism management, scheduler, dispatcher, and queues. Combining the information obtained in the different levels of evaluation, it is, then, possible to propose a methodology for FIT rate estimation by combining kernel profiling, fault injection, and instruction error rate.

A device's probability of being corrupted by a neutron is equal to the sum of the probabilities of having a neutron-induced corruption in one of its resources. Thus, the Cross-Section of a code (informally the FIT rate) is the sum of the probabilities of having a neutron-induced fault in each of the resources used for its computation multiplied by the fault probability in that resource to propagate and manifest at the output (the resource AVF). In principle, knowing the AVF and FIT rate of every resource used for computation would allow a perfect estimation of a code's FIT rate. Unfortunately, even if each GPU resource were accessible by the user, it would be unfeasible to measure the FIT and AVF of each resource since the GPU is a very complex device. We decided to limit this study to the contribution of GPUs' main functional units and memories. Beam experiments measured the FIT rates of most common functional units (arithmetical micro-instructions), register file, and shared memory. Then, the probability of a fault in each micro-instruction or used memory to affect the code output is calculated through fault injection.

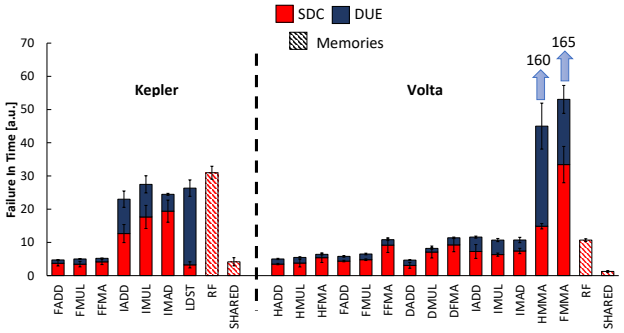


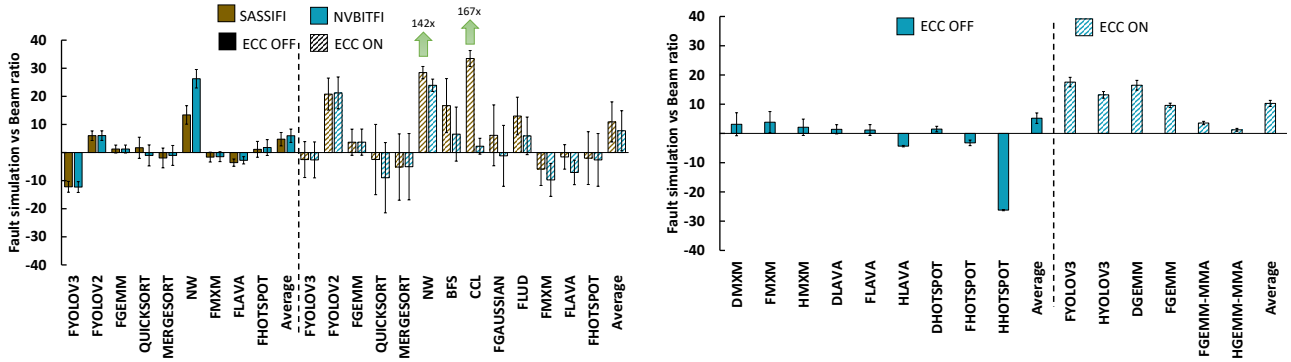
Fig. 3: Micro-benchmarks experimental FIT rates, normalized to each device's lowest measured value: FADD's DUE on Kepler, HFMA's DUE on Volta. Memories values are also normalized by the minimum memory FIT for each architecture.

We can estimate the FIT rate of a code (†FIT) by adding the expected error probability contribution of each micro-instruction $\sum P(E_{INST_i})$ and memory level $\sum P(E_{MEM_i})$ to the code error rate. The contributions to the code' FIT rate, $P(E_{INST_i})$ and $P(E_{MEM_i})$ depend on the number of resources used for computation, the probability of a fault to be generated (the resource cross-section), and the probability for the fault in that resource to affect the computation (AVF). The $P(E)$ is then calculated by the product of resource usage by the AVF by the cross-section of the resource. Moreover, as the percentage of instructions of a given type in the code directly impacts the probability of the fault propagating through the code [24], it has been chosen as a resource usage metric.

The GPU' FIT rate has the peculiarity of varying significantly based on the code degree of parallelism and how the GPU scheduler can allocate the available functional units. Consequently, the probability of a neutron corrupting an operation inside a thread depends on how many threads are active and how many parallel operations are executed. The higher the number of instructions a thread is allowed to schedule or the higher the number of active threads in an SM, the higher the number of functional units that can be corrupted. Hence, we profile the codes to measure code parallelism and to understand how many computing resources are exposed. We consider two metrics extracted from NVIDIA profiling tools on the FIT estimation, the GPU kernel Achieved Occupancy (AO) and the Instruction Per Cycle (IPC). Equation 1 is used to accurately estimate the $\dagger FIT$ rate on GPUs. High occupancy and a high IPC indicate that many resources are employed for computation. The lower the occupancy and the IPC, the lower the resources used for computation. When ECC is ON, the $P(E_{MEM_i})$ can be assumed to be zero.

$$\dagger FIT = AO \cdot IPC \cdot \sum P(E_{INST_i}) + \sum P(E_{MEM_i}) \quad (1)$$

2) Obtained Results: To estimate the FIT rate using Equation 1, the FIT_{INST_i} (micro-instruction error rate) and FIT_{MEM_i} (memory resources error rate) are needed. We designed seven classes of synthetic microbenchmarks that, 99%



(a) Kepler (b) Volta

Fig. 4: Comparison between the SDC FIT rate measured with the beam and predicted with fault injection.

of the time, execute only one type of GPU micro-instruction to measure the FIT rate of the functional units and main atomic instructions of Kepler and Volta architectures. The microbenchmarks can also be used to compare the reliability of the different functional units that compose GPUs architecture.

Figure 3 shows the microbenchmarks SDC and DUE normalized FIT rate measured with beam experiments on Kepler and Volta. FMA, ADD, MUL, and MAD are tested, both with ECC ON and OFF and produced similar error rates (differences lower than 20%). These microbenchmarks use only a few registers. Results show that for all the tested float instructions (FADD, FMUL, FFMA) on Kepler, both SDC and DUE rates are very similar. When the instructions are executed using INT32, the FIT rate is, on average, 4× higher than FP32. This is probably because the integer operations are executed in the same hardware as the FP32 operations with evident lower efficiency that can increase the vulnerability.

As mixed precision float operation is a crucial novelty on modern GPUs, we focus Volta's microbenchmarks analysis on how radiation can impact different precisions operations. The differences in the FIT rates between int, double, float, and half-precision operations in Figure 3 rely on the different Volta mixed-precision cores' complexities. Since a multiply requires more resources than an addition, its FIT rate is expected to be higher, and FMA (fused multiply and addition) is expected to have a FIT rate higher than ADD and MUL, which follows the results. Additionally, the higher the operation precision, the higher the FIT rate (higher precision implies more resource utilization). It is worth noting that, dissimilar to Kepler, integer operations on Volta are executed on dedicated cores. The FIT rate depends on the complexity of the hardware resources.

For Volta architecture, the FIT rate of Matrix Multiplication and Addition (MMA) micro-benchmark (*Tensor Core*) is also presented. While being more sensitive, the MMA core performs, in one operation, the equivalent of $4x4$ FMAs and the loop control variables needed to implement MxM in software. The FIT of each HMMA and FMMA is 9× and 12× higher than a FMA (FMMA uses the HMMA core after a cast). As 64 MMA instructions are required to multiply two 16×16 matrices, and for each warp-wide MMA instruction that replaces a warp of 32 FMAs, it is possible to deduce that the use of MMA is 2× (64/32, where 32 is the number of threads in a warp) more reliable than the combination of operations needed to execute a software MxM. MMA eliminates repeated fetches of the multiply-and-add operations and reduces activity in instruction memory and pipelines.

Beam vs Fault injection: The codes' FIT rates measured with beam experiments (Section II-A) are compared with those predicted with fault simulation and profiling following Equation 1. This comparison's main scope is to evaluate at which level a reliability analysis based on fault simulation can be considered realistic. Methodology details can be found at [25]. Two NVIDIA fault injectors are used for this work, SASSIFI (only supported for Kepler GPU) and NVBitFI.

Figure 4 compares the codes SDC FIT rate measured with beam experiments and estimated with fault injection. To ease the comparison visualization, for each code, the highest SDC FIT rate between the one measured with beam experiments and the one estimated is divided by the lowest SDC FIT rate between the two. Whenever the fault injection SDC FIT rate is higher than the beam one, the value is represented as negative, positive otherwise. For instance, on the Kepler with ECC OFF, executing FYOLOv3 fault simulation estimates a FIT rate 7× higher than the one experimentally measured.

A promising result is that despite the simplifications fault simulation introduces, in most cases, the SDC FIT prediction is reasonably close to the SDC FIT measured with the beam. The *absolute average* difference between fault simulation and beam on Kepler is 5× for SASSIFI and 6× for NVBITFI, with ECC OFF. When ECC is ON, on Kepler, the average difference is 11× for SASSIFI and 8× for NVBITFI. On Volta, the average is 5× when ECC is OFF and 10× when ECC is ON. As we compare completely different evaluation strategies, we consider these differences to be extremely promising.

For 25 out of 38 configurations, the fault injection underestimates the SDC FIT rate. One limitation of the model is that not all resources are accessible for fault simulation. Only the most common micro-instructions are contemplated, as testing all 20 instruction types is unfeasible. The most common micro-instructions are measured using NVIDIA profiling tools. While

the considered micro-instructions cover more than 70% of instructions that compose the codes, it is still possible that some errors in the unconsidered micro-instructions generate errors, and this would only count in beam experiments. When ECC is OFF, the prediction model will consider the memory error rate, already shown to dominate GPUs' FIT rate [26]. On average, when ECC is OFF, fault injection can better predict the beam SDC FIT rate, as the contribution to the FIT rate of the not modeled functional units and instructions is much smaller than the memory contribution.

For some outliers (NW and CCL on Kepler, HHotspot on Volta), the fault-injection-based estimation is very different from the beam. The kernels used for NW and CCL are not well-suitable for GPUs, and they underuse the available resources and have poor memory access patterns. These inefficiencies may reduce the possibility of having corruptions in functional units and increase the error rate due to other sources of errors, like threads and memory management. The proposed model still needs to consider these sources of errors, resulting in a poor underestimation of not well-parallelized codes.

We can use the method to derive insights for DUEs as well. DUEs can be caused by many factors, including interrupts triggered by ECC, corruption on device-host synchronizations, illegal memory accesses, hardware scheduler corruption, or fault-induced deadlocks. We mainly characterize the arithmetical functional units, memories, and Load/Store instructions of GPUs with beam experiments. Thus, only a subset of the causes for DUEs are included in the prediction model, and a significant underestimation of the code DUE FIT rate is to be expected. On average, the estimated DUE FIT rate is up to 629× for Kepler and 46,700× for Volta. This considerable divergence attests that many DUEs do not come from arithmetic micro-instructions and that modeling micro-instructions and memories are insufficient to predict the GPU DUE rate.

3) Novelty and foundation character: **This thesis is the first work that presented reliability analysis for 3 different levels of abstraction for NVIDIA GPUs.** We have considered physical fault injection with neutron beams, low-level GPU abstraction with RTL fault simulations, and assembly-level injections. As a consequence of the cross-layer evaluation, a methodology to provide a FIT estimation methodology that allows, for the large majority of the codes, an accurate SDC rate prediction has been proposed. Even considering the outliers, the SDC prediction average stayed up to 12×. Also, a new fault model based on the syndromes observed on physical and RTL fault injections is proposed (details at [27]). As a final remark on the SDC estimation, it would be good to emphasize that once the microarchitectural model is available, the proposed model could then be applied to predict the fault rate of codes executed in future GPUs. Despite the model's intrinsic limitations, it can successfully provide a good FIT rate estimation on average, even when ECC is ON.

C. Fault Tolerance for Convolutional Neural Networks

1) Concept an approach: Additionally, prior work shows that ECC does not mask all the faults as an error in computing

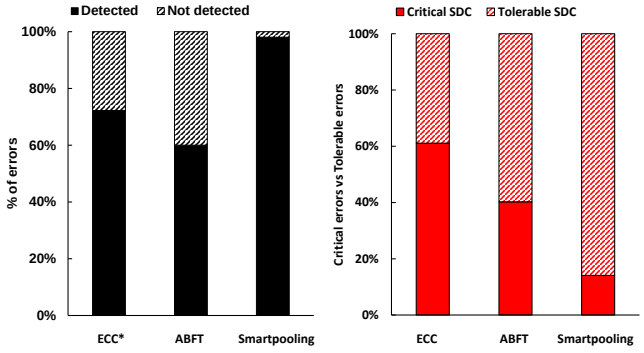


Fig. 5: Detected vs. Undetected errors, and Critical vs. Tolerable of undetected SDCs for YOLOv1

elements could propagate to the output [28]. The beam tests provide the realistic probability of experiencing an SDC when ECC is ON or OFF, which is the only way to evaluate ECC's effectiveness. Consequently, due to the increasing importance of CNNs for safety-critical applications, this work shows two hardening techniques for CNNs and compares them with available ECC for GPUs.

GEMM ABFT: This work leverages ABFT to protect YOLOv1, protecting matrix multiplication operations, as described by Huang et al. [29], and extended by Rech et al. [30] for GPUs. For FFT-based convolutions, a promising ABFT able to detect more than 80% of faults has been described by Pilla et al. [31]. Each convolutional layer of YOLOv1 is protected using ABFT to detect errors. Using an ABFT to harden a CNN arises from the observation that 67% of GPU processing in YOLO, 82% in Faster R-CNN, and 80% in ResNet is spent in matrix multiplication-related operations.

Reliable Max-pooling: Current CNNs use FP32 or smaller floating point formats due to the small intrinsic values represented in a CNN. A preliminary evaluation with two datasets (Caltech and VOC2012) shows that all the fault-free maximum absolute values for elements entering the YOLOv1 maxpool layers are up to 21.15. That is, the profiled values are extremely small, considering the full range of FP16 or FP32 that can be represented and that radiation can produce. This work then proposes a more reliable maxpool layer that evaluates if the value of the max element is greater than a threshold (to be conservative, the threshold is set to be $10x$ the max value of a fault-free execution) and, if so, halt the processing of the current frame and move to the next frame. This solution will detect faults in GPUs that affect multiple elements that impact the final inference most. As the maxpool layer is intrinsically imprecise, propagating the second-highest value does not significantly undermine detection accuracy [32], [33]. The overhead introduced to implement detection/correction is limited to 4 variables that hold the thresholds for each layer and a conditional for each thread.

2) Obtained results: Figure 5 shows the percentages of the detection for each fault tolerance tested on YOLOv1 running on a Kepler GPU. The beam experiments are performed using the same methodology as in Section II-A. Figure 5 also shows

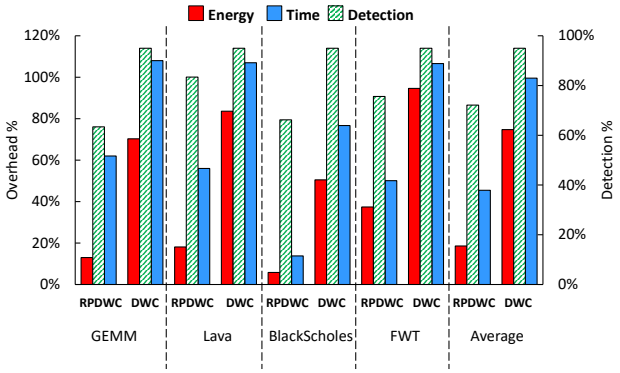


Fig. 6: Error detection and overhead of DWC and RP-DWC.

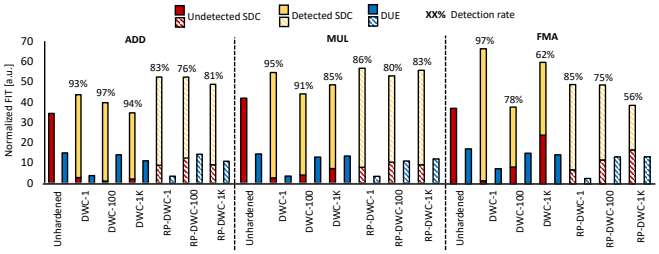


Fig. 7: Beam experimental data for the unhardened, the traditional DWC, and RP-DWC versions of the micro-benchmarks. Dashed lines are used just to highlight the RP-DWC versions.

the percentage of Critical and Tolerable SDCs produced for each technique. For comparison, in figure 5, the ECC detection efficacy is extracted using the relative difference between the execution under beam with ECC ON vs. ECC OFF for YOLOv1. The ABFT procedure does not significantly affect YOLOv1's SDCs or Crash error rate (details at [7]).

It is clear that ABFT can correct about 60% of the SDCs. If considering only the critical SDCs, an ABFT-protected YOLOv1 is more resilient than an ECC-protected version. This is because ABFT corrects all the detected errors that affect GEMM computation. As noted earlier, on GPUs, the ABFT code is run in parallel and executed in linear time. However, since ABFT performs multiplication/accumulation, it can be implemented using the same hardware used for matrix multiplication. Minimal hardware changes are required to implement the proposed hardening. ECC, on the contrary, has a logarithmic memory cost.

Additionally, Figure 5 reports the percentage of SDCs detected and undetected with the proposed max pool layer. Data was obtained with beam experiments. Smartpool detected 98% of SDCs under the beam. The most promising result is that the radiation experiments demonstrate that only 2% of the SDCs remain undetected, which is extremely close to the 99% detection limit ASIL-D imposes for self-driving vehicles.

3) Novelty and foundation character: **This thesis is the first to evaluate the error rate, criticality, and fault tolerance for Convolutional Neural Networks running on GPUs and to experimentally validate the proposed hardening solutions**. Prior work have also considered how to improve the reliability of neural networks. Most available solutions rely on partial or total duplication (or even triplication) of operations, and some techniques require specific hardware modifications. Due to processing overhead and added costs, these approaches are less than ideal for real-time object detection, a task commonly performed in automotive applications. The techniques proposed in this thesis are optimal for CNNs and can be implemented without much effort on modern GPUs.

D. Reduced Precision Duplication With Comparison

1) Concept and approach: We propose a more generic fault tolerance method for GPU named RP-DWC. The traditional DWC is a generic system modular redundancy with a result comparison at the end. RP-DWC consists of duplicating the instruction flow for execution with lower precision. Reduced Precision DWC has three main benefits compared to a traditional DWC: (1) **Smaller overhead**, as the redundant copy is always in a reduced precision (FP64 to FP32 for this work), the overhead is always smaller when compared with standard redundant techniques; (2) **The replica has a lower probability of being corrupted**, as reducing precision reduces the code error rate (see Section II-A); (3) **Diversity of the copies**, since the replicated operations will execute in a different precision and use different processing resources, reducing the chances for a fault to have the same impact in both copies.

To implement RP-DWC, five steps must be taken: **Step 1.** Casting the inputs to Reduced Precision, in this work, from FP64 to FP32; **Step 2.** Executing the original and reduced-precision instructions. As modern GPUs have dedicated units to execute different types of precisions, a block of independent instructions for different precisions can execute in parallel; **Step 3.** Casting the high-precision result to reduced precision. Once both instructions complete their execution, FP64 is cast to FP32 for the comparison; **Step 4.** Performing the error detection operation. To compare the FP64 cast to FP32 and FP32, it is possible to consider reinterpreting them as unsigned integers (UINT32) and subtracting them. By subtracting the two 32 bits representations interpreted as UINT, we have a fast (and accurate) valuation of the magnitude of the difference between the two numbers. The higher the result of the subtraction, the more significant the difference between the two representations; **Step 5.** Comparing and taking action on the result based on the magnitude of the UINT result.

We can implement the correctness check different granularities, at each instruction or after a block of instructions. In a coarse-grained RP-DWC, a sequence of FP64 instructions is duplicated with a series of FP32 instructions. The replicated sequence receives the FP64 input cast to FP32. The two sequences are executed in parallel without interacting until the correctness check is reached. A longer block of instructions could increase the two copies' intrinsic difference. A larger copies difference implies a higher number of undetectable errors. However, as the error propagates in the sequence of instruction, it may increase in magnitude, thus becoming detectable even with a larger copy difference (details at [34]).

2) Obtained results: A software fault injection is performed to evaluate the efficiency of DWC and RPDWC. Random

single-bit flips are injected in the code execution, and the application's output is compared with a golden version to check correctness. Figure 6 shows the efficiency and efficacy of RP-DWC by showing the percentage of detected errors and the imposed overheads (execution time and energy consumption). The detection rate and the overheads of a traditional DWC are also included to ease the comparison with RP-DWC. Four codes are shown, MXM, Lava, FWT, and BlackScholes.

The slightly lower detection rate compared to traditional DWC (over 95%) or state-of-the-art DWC from previous work [14], [17]–[20], [35], [36] is not surprising, since, as any faults hitting or propagating to the less-significant bits of an FP64 number are not detectable. Even though this limitation of RP-DWC provides an upper bound of erroneous bits coverage it can achieve, these wrong least significant bits are the ones that will provide a smaller impact to the application output.

RP-DWC has a higher detection capability than previous work that approximates the algorithm or proposes approximated hardware for error detection ([18], [20], [37]), which is 55%-76% for RP-DWC and 20%-40% for previous work. This data attests that approximating the algorithm might not be as effective as approximating the hardware. The higher error detection of RP-DWC compared to the use of dedicated approximated hardware could be caused by the higher approximation chosen in [17], [18] and by intrinsically more reliable hardware designed by NVIDIA.

Data presented in Figure 6 shows that the execution time overhead of RP-DWC (35% in the worst case of a fine-grain RP-DWC) is much lower than traditional DWC (70%-90%) and of recent efficient DWC (39% in [36]). Our version of conventional DWC has a lower overhead compared to some previous studies that showed an overhead of 2x [14], as it duplicates operations inside a thread rather than threads or blocks of threads. If FP64 cores are available, the GPU could then schedule some of the two FP64 copies in parallel.

Figure 6 shows that the energy overhead of RP-DWC is significantly reduced. The overhead can be as low as 24% to 32%. BlackScholes reaches an even lower energy overhead (13.8%). Such a low overhead is not solely justified by RP-DWC but also by the simplicity of the code (few global memory access, no shared memory utilization, and executes only simple operations) [38]. For the same reasons, the energy overhead of the traditional DWC is lower for BlackScholes than the other codes. The energy consumption overhead of RP-DWC is comparable to the traditional DWC only for the fine-grain implementation (1 check every operation for RP-DWC vs. 1 check at the end of the application for traditional DWC). The energy consumption overhead of a fine grain RP-DWC is, then, higher than 100%. This is justified because it is actually executing 3x the instructions of the unhardened version (FP64 and FP32 copies plus the error detection operation). Traditional DWC has, even with the check only at the end of the computation, a higher energy consumption overhead, which ranges from 111% to 124%. This favorable energy consumption result of RP-DWC is achieved by leveraging the FP32 cores to execute the redundant copy in parallel.

Neutron Beam Experiments: To have an even more realistic evaluation of the effectiveness of RP-DWC and a direct comparison with traditional DWC, the GPUs were exposed to accelerated neutron beams running microbenchmarks implemented in the same fashion as the ones discussed in Section II-B. Figure 7 reports the beam experiment results for the microbenchmark in the unhardened version (no duplication) and protected with a traditional DWC and with RP-DWC, in three different granularity (correctness check after 1, 100, or 1,000 operations). The detection rates of DWC and RP-DWC are explicit in the Figure to ease comparing the effectiveness of the two techniques. The dashed columns differentiate the RP-DWC results from the traditional DWC ones.

Figure 7 shows that the SDC FIT rate for the unhardened version is lower than the FIT rate of the protected versions (considering the combination of detected and undetected SDCs). This is expected, as the check operation introduces a computation and memory overhead that can increase the protected versions error rate. The increased FIT rate is higher when the check is performed at each instruction (more instructions executed in parallel). RP-DWC has a higher increase in the SDC rate for all configurations but FMA. This is because the cast and error detection operations are computationally more costly than ADD and MUL, but not of FMA. Nevertheless, DWC and RP-DWC detect most SDCs, resulting in a much lower undetected SDC rate than unhardened versions. As observed with fault injection, and for the same reason, the detection rate of the traditional DWC is always higher than the RP-DWC. On average, under the beam, the detection rate of RP-DWC is 9% lower than DWC. The use of RP-DWC results in a slightly lower error detection but makes both overheads much smaller than traditional DWCs, as reported in [4], [36].

3) Novelty and foundation character: **RP-DWC is the only approach that is, at the same time, generic, software-implemented (no hardware changes), and leverages the existing redundant mixed-precision hardware** for reduced performance and energy overheads. RP-DWC requires no prior knowledge of the algorithm and can be automatically inserted by the compilation toolchain. Previous to other software-implemented instruction replication, RP-DWC exploits the dedicated mixed-precision functional units available in modern GPUs to execute the replicated dataflow, leveraging these units that would otherwise be idle to increase the parallelism and reducing the execution time of the hardened software.

Compared to approaches that propose approximate hardware for fault detection, the RP-DWC approach is implemented entirely in software and targeted towards already existing and future architectures. A key advantage of RP-DWC is that the extra hardware (already paid in the form of mixed-precision processing elements) can be used for performance improvements (when reliability is not an issue) and for approximate fault detection. Moreover, software implementations can be parameterized for different trade-offs between error detection and overheads for INT64, INT32, INT16, INT8, and floating-point computations FP64, FP32, FP16, and BF16/TF32.

III. Impact

In this thesis, I presented a detailed reliability analysis of a broad domain of applications from HPC to machine learning. With a multi-level evaluation, I proposed fault tolerance for HPC and safety-critical domains that significantly beat, in terms of efficiency and efficacy, existing techniques such as ECC and DWC. The data and discussions presented in this work are helpful for future research and industry innovation as they pave a path for detailed reliability evaluation of next-generation GPUs and showcase how to design efficient and effective hardening solutions.

I conducted the most complete and extensive study on GPU reliability, focusing on attacking one of the leading issues on HPC and autonomous vehicles nowadays, single event effects on computing devices, i.e., SDCs and DUEs. It is estimated that a large supercomputer like Oak Ridge National Laboratory Titan has a functional stop every ≈ 40 hours due to faults caused by terrestrial neutrons [39]. As GPUs are the heart of the main supercomputers today, the results presented in this thesis can strengthen industrial competitiveness, growth, and sustainability in HPC research and industry.

Thanks to the extensive beam experiments performed for this thesis, we can accurately understand DUEs occurrences since fault simulation acts at a higher level of abstraction. Our detailed evaluation of DUEs on GPUs, which includes measuring the error rate and the error sources, has a direct market and societal impact. This knowledge can help researchers and engineers solve the problem of the high rate of supercomputer nodes crashing due to faults caused by terrestrial neutrons. Avoiding supercomputer interruptions can save time and energy, and increase productivity, as the cost of having a supercomputer rebooting is exceptionally high.

The study presented in this thesis is not limited to HPC. With the proposed cross-layer analysis, we can also tackle other problems that arise from radiation effects on GPUs, such as the fault tolerance of Machine Learning (ML) applied to safety-critical applications. GPUs are today the standard hardware for ML, allowing developers to create and deploy new algorithms without relying on hardware changes. However, autonomous vehicles driven by embedded GPUs can only be employed on a large scale if the reliability of current and future machine learning algorithms is improved.

The software-level techniques proposed for CNNs enable the development of reliable commercial systems with a much faster approach. For example, it is possible to cite the new *NVIDIA Drive Thor* platform for self-driving cars composed of multiple embedded GPUs. Our hardening method allows future developers to improve the final system's reliability without changing the hardware platform, leading to a more reliable system without compromising productivity and budget.

It is worth noting that a plethora of open questions arise from this thesis. We can cite at least two that will have a high impact in the future: (1) The impact of compilers and micro-instructions on the fault probability must be deeply studied. Simpler architectures commonly used on safety-critical applications have a well-developed ecosystem that provides compilers and evaluation tools for engineers to estimate the reliability of an application before deploying the system. Such tools and methods are not yet available for complex hardware like GPUs; (2) Take advantage of the machine learning nature by projecting ML algorithms that "learn" to be reliable against errors at the training phase. By understanding the architecture fault model, as we did for GPUs, the researchers may develop ML algorithms that "know" how to keep the accuracy at the inference phase even in the presence of faults.

During my Ph.D. studies, I have collaborated with many research institutions and industries. I participated in two internships in the largest radiation test facilities in the world, at the

TABLE I: Scientific production through the years of the Ph.D study

International journal papers published as main author	
Analyzing and increasing the reliability of convolutional neural networks on GPUs	**Transactions on Reliability 2018**
Kernel and layer vulnerability factor to evaluate object detection reliability in GPUs	**IET Computers & Digital Techniques 2019**
Reduced precision DWC: an efficient hardening strategy for mixed-precision architectures	**Transactions on Computers 2021**
Experimental Findings on the Sources of Detected Unrecoverable Errors in GPUs	**Transactions on Nuclear Science 2022**
International journal papers published as a co-author	
Impact of tensor cores and mixed precision on the reliability of matrix multiplication in GPUs	**Transactions on Nuclear Science 2020**
Thermal neutrons: a possible threat for supercomputer reliability	**The Journal of Supercomputing 2021**
Physical stress, Book chapter: Cross-Layer Reliability of Computing Systems	**IET Digital Library 2020**
Conference papers	
Analyzing the criticality of transient faults-induced SDCS on GPU applications	**ScalA 2017**
Evaluation and mitigation of soft-errors in neural network-based object detection in three GPU architectures	**DSN-W 2017**
Radiation-induced error criticality in modern HPC parallel accelerators	**HPCA 2017**
Code-dependent and architecture-dependent reliability behaviors	**DSN 2018**
Reliability evaluation of mixed-precision architectures	**HPCA 2019**
Impact of reduced precision in the reliability of deep neural networks for object detection	**ETS 2019**
Reduced-Precision DWC for Mixed-Precision GPUs	**IOLTS 2020**
An Overview of the Risk Posed by Thermal Neutrons to the Reliability of Computing Devices	**DSN-S 2020**
Demystifying gpu reliability: comparing and combining beam experiments, fault simulation, and profiling	**IPDPS 2021**
Revealing GPUs Vulnerabilities by Combining Register-Transfer and Software-Level Fault Injection	**DSN 2021**
Protecting GPU's Microarchitectural Vulnerabilities via Effective Selective Hardening	**IOLTS 2021**
Combining Architectural Simulation and Software Fault Injection for a Fast and Accurate CNNs Reliability Evaluation on GPUs	**VTS 2021**

Rutherford Appleton Laboratory in 2018 and the Los Alamos National Laboratory Radiation Effects Summer School in 2019. As a result, we have published papers in collaboration with many researchers from different institutions, including NVIDIA, Los Alamos National Laboratory, Rutherford Appleton Laboratory, Politecnico di Torino, Northeastern University College of Engineering, and Ecole Centrale de Lyon.

As a Ph.D. student, I was invited to become a reviewer of different journals, transactions, and conferences, such as Microelectronics Reliability, IEEE Transactions on Nuclear Science, Journal of Systems Architecture, SELSE Workshop, and IEEE IOLTS conference. Also, while a Ph.D. student, I was a co-advisor on a bachelor's final work and guided undergrad and master students on everyday research tasks.

Based on the effort and importance of this thesis, the reliability, testing, and radiation effects communities have extensively recognized our work. The impact of the work performed in my Ph.D. is demonstrated by the number of international journal publications (7 journals published) and by the number of citations (409 in total, with an h-index of 9, according to Google Scholar). Additionally, I am very honored for the awards this thesis was recognized:

1) **CAPES award:** Best 2021 thesis on Computer Science of the whole country by CAPES. CAPES is the Brazilian government agency for research funding.
2) **Paul Phelps Award:** In 2021, I was awarded the *Paul Phelps Continuing Education Award* from the IEEE Nuclear & Plasma Sciences society.
3) ***Cum Laude*:** My thesis was evaluated with *Cum Laude* by a board of reviewers composed of Dr. Timothy Tsai (NVIDIA), Dr. Dimitris Gizopoulos (National and Kapodistrian University of Athens), and Dr. Evgenia Smirni (College of William & Mary).
4) **Best paper runner up:** In the 2018 edition of IEEE/IFIP DSN, a paper that I have co-authored was one of the runner-ups for best paper award.
5) **Best of SELSE:** In the 2018 edition of SELSE, a paper that I have co-authored won the best paper award.

IV. List of scientific output

Table I presents all scientific output generated in the Ph.D. I have authored and co-authored 19 papers during my Ph.D.

References

[1] P. Rech *et al.*, "Measuring the Radiation Reliability of SRAM Structures in GPUS Designed for HPC," in *IEEE 10th SELSE*, 2014.
[2] S. K. S. Hari *et al.*, "Low-cost program-level detectors for reducing silent data corruptions," in *IEEE/IFIP DSN*, 2012.
[3] M. B. Sullivan *et al.*, "Characterizing And Mitigating Soft Errors in GPU DRAM," in *54th Annual IEEE/ACM MICRO*, 2021.
[4] J. Wadden *et al.*, "Real-World Design and Evaluation of Compiler-Managed GPU Redundant Multithreading," in *41st ISCA*, 2014.
[5] ISO, "ISO 26262-9:2011 Preview Road Vehicles Functional Safety." https://www.iso.org/standard/51365.html, 2011.
[6] NVIDIA, "NVIDIA Anounces World's First Functionally Safe AI Self-Driving Platform," 2018.
[7] F. F. d. Santos *et al.*, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," *Trans. on Reliability*, 2019.
[8] J. Wei *et al.*, "Quantifying the Accuracy of High-Level Fault Injection Techniques for Hardware Faults," in *IEEE/IFIP DSN*, 2014.
[9] S. K. S. Hari *et al.*, "SASSIFI: An architecture-level fault injection tool for GPU application resilience evaluation," in *IEEE ISPASS*, 2017.
[10] S. Tselonis *et al.*, "GUFI: A framework for GPUs reliability assessment," in *IEEE ISPASS*, 2016.
[11] T. Tsai *et al.*, "NVBitFI: Dynamic Fault Injection for GPUs," in *IEEE/IFIP DSN*, 2021.
[12] L. Yang *et al.*, "Practical Resilience Analysis of GPGPU Applications in the Presence of Single-and Multi-Bit Faults," *IEEE Transactions on Computers*, 2021.
[13] M. Kaliorakis *et al.*, "Differential Fault Injection on Microarchitectural Simulators," in *IEEE Int. Symp. on Workload Characterization*, 2015.
[14] D. A. G. Goncalves de Oliveira *et al.*, "Evaluation and Mitigation of Radiation-Induced Soft Errors in Graphics Processing Units," *IEEE Transactions on Computers*, 2016.
[15] K. Ito *et al.*, "Analyzing DUE errors on GPUs with neutron irradiation test and fault injection to control flow," *IEEE Trans. Nucl. Sci.*, 2021.
[16] J. M. Badia *et al.*, "Reliability evaluation of lu decomposition on gpu-accelerated system-on-chip under proton irradiation," *IEEE Transactions on Nuclear Science*, 2022.
[17] K. Seetharam *et al.*, "Applying Reduced Precision Arithmetic to Detect Errors in Floating Point Multiplication," in *IEEE 19th PRDC*, 2013.
[18] M. Maniatakos *et al.*, "Exponent monitoring for low-cost concurrent error detection in FPU control logic," in *29th IEEE VTS*, 2011.
[19] M. B. Sullivan, *Low-cost duplication for separable error detection in computer arithmetic*. PhD thesis, The Univ. of Texas at Austin, 2015.
[20] G. Rodrigues *et al.*, "Approximate TMR based on successive approximation and loop perforation in microprocessors," *Micro. Reliability*, 2019.
[21] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," *JEDEC Standard*, 2006.
[22] C. Cazzaniga *et al.*, "Progress of the Scientific Commissioning of a fast neutron beamline for Chip Irradiation," *Journal of Physics*, 2018.
[23] J. Redmon *et al.*, "YOLO9000: Better, Faster, Stronger," *arXiv preprint arXiv:1612.08242*, 2016.
[24] G. Li *et al.*, "Modeling Input-Dependent Error Propagation in Programs," in *IEEE/IFIP DSN*, 2018.
[25] F. F. d. Santos *et al.*, "Demystifying GPU Reliability: Comparing and Combining Beam Experiments, Fault Simulation, and Profiling," in *IEEE IPDPS*, 2021.
[26] I. S. Haque *et al.*, "Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU," in *2010 10th CCGRID*, 2010.
[27] F. F. d. Santos *et al.*, "Revealing gpus vulnerabilities by combining register-transfer and software-level fault injection," in *IEEE DSN*, 2021.
[28] G. Li *et al.*, "Understanding Error Propagation in GPGPU Applications," in *SC16*, 2016.
[29] K.-H. Huang *et al.*, "Algorithm-Based Fault Tolerance for Matrix Operations," *IEEE Transactions on Computers*, 1984.
[30] P. Rech *et al.*, "An Efficient and Experimentally Tuned Software-Based Hardening Strategy for Matrix Multiplication on GPUs," *IEEE Transactions on Nuclear Science*, 2013.
[31] L. L. Pilla *et al.*, "Software-Based Hardening Strategies for Neutron Sensitive FFT Algorithms on GPUs," *IEEE Trans. Nucl. Sci.*, 2014.
[32] M. Riesenhuber *et al.*, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, 1999.
[33] D. Scherer *et al.*, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in *ICANN Part III*, 2010.
[34] F. F. dos Santos *et al.*, "Reduced Precision DWC: An Efficient Hardening Strategy for Mixed-Precision Architectures," *IEEE Transactions on Computers*, 2022.
[35] P. Kudva *et al.*, "Low-Cost Concurrent Error Detection for Floating-Point Unit (FPU) Controllers," *IEEE Transactions on Computers*, 2013.
[36] A. Mahmoud *et al.*, "Optimizing Software-directed Instruction Replication for GPU Error Detection," in *SC18*, 2018.
[37] Y. Zhang *et al.*, "Reduced Precision Checking to Detect Errors in Floating Point Arithmetic," *CoRR*, 2015.
[38] A. Yazdanbakhsh *et al.*, "AxBench: A Multiplatform Benchmark Suite for Approximate Computing," *IEEE Design Test*, 2017.
[39] D. Tiwari *et al.*, "Understanding GPU Errors on Large-scale HPC Systems and the Implications for System Design and Operation," in *ACM HPCA*, 2015.