Investigating and Reducing the Architectural Impact of Transient Faults in Special Function Units for GPUs

Journal of Electronic Testing (2024) 40:215–228
https://doi.org/10.1007/s10836-024-06107-9

# Investigating and Reducing the Architectural Impact of Transient Faults in Special Function Units for GPUs

**Josie E. Rodriguez Condia[1] · Juan-David Guerrero-Balaguera[1] · Edwar J. Patiño Núñez[2] · Robert Limas[1] · Matteo Sonza Reorda[1]**

Received: 30 July 2023 / Accepted: 14 February 2024 / Published online: 21 March 2024


**Abstract**
Ensuring the reliability of GPUs and their internal components is paramount, especially in safety-critical domains like autonomous machines and self-driving cars. These cutting-edge applications heavily rely on GPUs to implement complex algorithms due to their implicit programming flexibility and parallelism, which is crucial for efficient operation. However, as integration technologies advance, there is a growing concern regarding the potential increase in fault sensitivity of the internal components of current GPU generations. In particular, Special Function Unit (SFU) cores inside GPUs are used in multimedia, High-Performance Computing, and neural network training. Despite their frequent usage and critical role in several domains, reliability evaluations on SFUs and the development of effective mitigation solutions have yet to be studied and remain unexplored. This work evaluates the impact of transient faults in the main hardware structures of SFUs in GPUs. In addition, we analyze the main overhead costs and benefits of developing selective-hardening mechanisms for SFUs. We focus on evaluating and analyzing two SFU architectures for GPUs (*'fused'* and *'modular'*) and their relations to energy, area, and reliability impact on parallel applications. The experiments resort to fine-grain fault injection campaigns on an RTL GPU model *(FlexGripPlus)* instrumented with both SFUs. The results on both SFU architectures indicate that *fused* SFUs (in commercial-grade devices) require lower area overhead (about 27%) for their integration in GPUs but are more vulnerable to transient faults (in up to 47% for the analyzed cases) and less power efficient (in up to 36.6%) than *modular* SFUs. Moreover, the reliability estimation shows that *Modular* SFUs are structurally more resilient than *Fused* ones in up to one order of magnitude. Similarly, selective-hardening mechanism based on Triple-Modular Redundancy (TMR) shows that coarse-grain strategies might increase the reliability of the overall SFUs under feasible overhead costs.

**Keywords** Graphics processing units (GPUs) · Fault-tolerance · Reliability evaluation · Special function unit (SFU) · T-Stream core

## 1 Introduction

The programming flexibility and the structural parallelism of Graphics Processing Units (GPUs) boost their vertiginous adoption in several domains, from multimedia and gaming to aerospace, automotive, military, and High-Performance Computing (HPC) applications. In fact, (GPUs) are massively deployed to implement complex algorithms in safety-critical applications, such as those in the automotive and autonomous machines domains (e.g., Deep Neural Networks, *Advanced Driver-Assistance Systems* or *'ADAS'*, and sensor fusion

Responsible Editor: L. M. Bolzani Poehls

✉ Josie E. Rodriguez Condia
josie.rodriguez@polito.it

Juan-David Guerrero-Balaguera
juan.guerrero@polito.it

Edwar J. Patiño Núñez
edwar.patino@uptc.edu.co

Robert Limas
robert.limassierra@polito.it

Matteo Sonza Reorda
matteo.sonzareorda@polito.it

[1] Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin 10129, Italy

[2] Electronics Engineering School, Universidad Pedagogica y Tecnologica de Colombia (UPTC), Av. Central del Norte 39-115, Tunja 150003, Colombia

systems), where device reliability and functional safety are significant concerns. In detail, industrial functional safety standards and norms, such as the ISO 26262 in automotive, demand safety mechanisms and reliability evaluations determining fault effects in a device.

Despite the use of cutting-edge transistor technologies in GPUs to increase performance and reduce power consumption, the *"International Roadmap for Devices and Systems - 2022"* (IRDS) and several independent studies [24, 34] suggest that modern digital devices, such as GPUs, are highly susceptible to *Electromigration* and *Time-Dependent-Dielectric-Breakdown*, both major sources of in-field and accelerated fault effects [25]. In particular, IRDS emphasizes that the lifetime of a device decreases by half at each new manufacturing process generation [25], exacerbating the importance of reliability evaluations and mitigation solutions in GPUs and their internal units. Unfortunately, the limited structural information and the missing architectural details from real devices interfere with deep reliability evaluations (e.g., on the architecture and applications), as well as the exploration and validation of mitigation solutions.

Among the available functional units and cores in GPUs, the *Special Function Units* (SFUs) [46], or *T-Stream* cores [3] are essential accelerators calculating *(in hardware!)* efficient trigonometric and transcendental operations for several domains (e.g., pre-processing, handling, and correlation of images, sensor fusion, and training/inference of Neural Network algorithms). Unfortunately, most of the previous works on GPU's functional units reliability targets Floating-Point Units [39], Integer cores [48], and Tensor units [2, 33, 47], leaving fault effects in SFUs largely unexplored.

Most works in literature analyze the reliability of processor-based systems and hardware accelerators (e.g., CPUs and GPUs) by resorting to three strategies: *i)* Beam experiments, exposing a device to radiation and analyzing their effects on targeted workloads, *ii)* Software-based error injection, representing faults as instruction errors in software, and *iii)*, architectural/functional and low-level microarchitectural simulations, by injecting faults on a functional, RTL- or gate-level implementation of a design [6, 7]. The first two methods employ real devices but can hardly analyze fault effects on focused units. In contrast, the last method provides accurate and fine-grain evaluations when descriptions are available. Authors in [37, 45] analyzed the reliability assessment of the main memory elements in GPU and CPU devices. Their results demonstrate that available low-level structures of a target device increase the accuracy in evaluating reliability. Similarly, authors in [29] exploited functional simulators to evaluate the reliability assessment of multiple GPU architectures in mainly memory hierarchy units. Other works [9, 14–16, 19, 21, 40] evaluated the reliability features of several GPU units (pipeline registers and block schedulers). Unfortunately, most works neglected to evaluate transient fault effects on SFUs. Moreover, some of them are limited by missing structural details of the units, i.e., functional simulators provide acceptable evaluations of memory and data-path units. Still, these can barely describe and evaluate (at fine-grain) functional units, such as SFUs. Authors in [12] analyzed the incidence of SFUs in the application's sensitivity to fault effects. In this case, two versions of the workloads (with and without SFU) are evaluated to observe the workload's impact on transient fault effects injected as instruction errors. This work also introduced a first approach to analyzing the structure of SFUs. Another work [17] provided a first attempt to analyze the effects of faults in SFUs. However, the evaluation was limited to permanent transition path delay faults. To the best of our knowledge, no works in the literature evaluate and analyze the architectural effects of transient faults on the reliability of SFUs for the later exploration of selective hardening solutions.

This manuscript extends a preliminary work [13] that explored the evaluation, analyses, and trade-off among the area, power consumption, and reliability of two SFU architectures. We focused on evaluating the impacts of transient faults (*Single Event Upsets* or SEUs) in the structures of two hardware implementations of SFUs for GPUs: *1)* a *fused* SFU **(SFU1)**, and *2)* a *modular* SFU **(SFU2)**. In detail, *Fused* SFUs are commercial-grade designs exploiting Piece-wise Polynomial Approximations (PPA) [38] to implement highly area-efficient architectures reusing sub-units and process several operations. Moreover, *Modular* SFUs comprise simple, optimized, and independent units (organized in parallel) that implement in hardware compacted algorithms to calculate specific operations [11]. This work extends the reliability analyses on several parallel workloads and micro-benchmarks for SFU cores. Moreover, this work evaluates the impact of the architectural features of both SFUs on their performance operation. In addition, this work proposes, implements, and evaluates the modeling of coarse-grain selective hardening mechanisms for the SFU architectures in GPUs. In particular, we analyze the impact, main benefits, and overhead costs of passive fault-tolerance selective-hardening mechanisms (i.e., based on *Triple Modular Redundancy* or TMR approaches) to mitigate transient fault effects in SFUs.

To evaluate and validate the impacts on the reliability of SFUs and implement the passive selective hardening mechanisms, we resort to one open-source GPU model (*FlexGripPlus*) [10] instrumented with both SFU architectures. We use two available open-source SFUs (with *modular*[1] and *fused*[2] architectures) developed and released in previous works [11, 22]. A total of 20 statistical fault injection campaigns determined the most vulnerable structures in both SFUs and

[1] https://opencores.org/projects/special_functions_unit.

[2] https://opencores.org/projects/special_function_unit_ppa.

provide the impacts at the application levels. Those vulnerable structures are the main targets for the selective hardening analysis in both SFU architectures.

Our results suggest that *modular* SFUs are more structurally resilient to transient faults than *fused* ones by their implicit architecture (workload corruption effects reduced from about 5% to 47%). The multi-functional operation in *fused* SFUs (reusing hardware sub-units) seems to be the main factor in increasing their fault vulnerability. In contrast, using independent units per operation increases the fault resilience in *modular* SFUs. The area and power budget analyses on both SFUs show that *fused* ones demand an additional moderate percentage of power (about 36.6%) in comparison with *modular* ones for the same amount of operations in the complete GPU core. Unsurprisingly, *modular* SFUs are less area efficient than *fused* SFUs (in around 27% of area and resources). The analysis shows that SFU's architecture is vital in its implicit fault vulnerability. Moreover, the association of fault impacts, power budget, operational latency, and area overhead highlights the main benefits and possible disadvantages of each SFU architecture. Then, we modeled and developed selective-hardening solutions for SFUs. For our validation, we employ FPGA-based platforms to evaluate parameters of area and power consumption overhead. Our reliability models suggest that *Fused* SFUs are less structurally reliable than *Modular* ones in up to one order of magnitude.

The document is organized as follows. Section 2 introduces a background of the architectural organization of GPUs and SFUs. Section 3 describes the evaluation approach to characterize fault effects on both SFU architectures. Then, Section 4 reports the fault characterization experiments and their impacts. Section 5 discusses the area and power analyses on both SFU architectures and relates the impacts regarding reliability. Then, Section 6 modulates and evaluates passive selective hardening mechanisms for the vulnerable structures in SFU architectures. Finally, Section 7 draws future works and provides conclusions.

## 2 Background

This section describes the organization and main features of GPUs and SFU cores.

### 2.1 GPU Organization

GPUs are homogeneous arrays of *Parallel Processors* (also known as *Streaming Multiprocessors* or SMs) grouped in clusters to operate one or several parallel tasks exploiting the Multiple-Instruction Multiple-Data (MIMD) paradigm. Each SM implements Single-Instruction Multiple-Data/Thread (SIMD or SIMT) schemes to execute groups of threads (i.e., Warps) in parallel. More in detail, the SM comprises a pipeline of one or more scheduler controllers, a fetch unit, an instruction's decoder, memory controllers, local memories, register files, and several execution units devoted to process arithmetic and logic operations for multiple Warps. Current GPU generations include arrays of Floating-point units (FPUs) in single- (FP32) and double-precision (FP64) sizes, Integer/Streaming cores (INT/SP), and special-purpose accelerators, such as SFUs, which are devoted to performing trigonometric and transient operations, as part of each SM core.

In particular, SFUs are vital units in two main domains: *i)* general purpose computing and *ii)* graphics rendering [30]. In the first case, the SFU cores perform general-purpose operations (e.g., the reciprocal, exponent, logarithm, square root, and trigonometric functions) highly used in CNN's training and the implementation of image processing algorithms (e.g., using *CUDA*). In the second case, the SFUs are a crucial engine of the graph data path in GPUs (i.e., hardware operations of coordinate transformation, perspective division, and vector normalization), which are commonly configured through Graphics *'Application Programming Interface'* (APIs).

### 2.2 Organization of SFU Cores

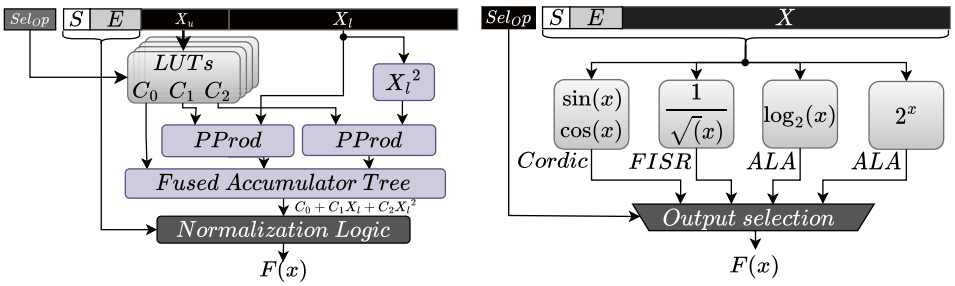
SFUs (or *T-Stream* cores) are crucial in-chip hardware accelerators intended to efficiently execute complex functions. The SFUs in GPUs perform a fast approximation of several transcendental functions, such as ($sin(x)$, $cos(x)$, $\frac{1}{\sqrt{x}}$, $2^x$, and $\log_2(x)$) on real value operands expressed in floating-point IEEE-754 formats.

In hardware, SFUs use a wide variety of approximation algorithms to describe transcendental and special functions directly implying in the final core's structures. The algorithms are classified according to its operation as *i) iterative* when require several steps to provide a result (i.e., Cordic algorithms), and *ii)* non-iterative that compute results using efficient and compacted combinational hardware. Both algorithms can be combined according to different SFU design goals, always looking for a balance among performance, area, precision, and scalability.

Typically, SFUs in commercial products adopt non-iterative approximation algorithms leading to *'Fused'* architectures reusing the same hardware to implement more than one operation [27, 44]. Furthermore, alternative design strategies adopt *Modular* approaches to employ independent and optimized hardware (implementing one or several iterative and non-iterative algorithms) to compute individual operations [11].

#### 2.2.1 Architecture of Fused SFUs

These cores implement the *Piece-wise Polynomial Approximation* (PPA) [27] approach to calculate




(a) Fused SFU (SFU1) (b) Modular SFU (SFU2)

**Fig. 1** A general scheme of the architectures of the *fused* SFU using a PPA structure (**a**) and the *modular* SFU (**b**)

transcendental operations. The PPA approach splits the input value into a set of equal-size sub-segments and evaluates a polynomial expression using per-segment coefficients stored in lookup tables (LUTs) (i.e., Quadratic Polynomial Approximation [44]).

Figure 1 (*left*) depicts the scheme of an SFU employing the polynomial expression $f(x) = C_0 + C_1X_l + C_2X_l^2$. Where $C_0$, $C_1$, and $C_2$ are the segment coefficients indexed by the *Xu* input that describes the segment where the approximation happens, and $X_l$ represents the point inside the segment at which the approximation is made. The general organization of the computation core comprises five main components: a square unit [26], two partial product generators (PProd), a Fused Accumulation Tree, a set of LUTs (one per function to be evaluated), and the normalization and output logic (NL). PPA architectures provide multi-functional operation allowing optimized implementations of an SFU, in terms of area, and latency. Since PPA schemes are highly flexible, several nonlinear functions can be implemented in an SFU by reusing the same hardware and only resorting to specific coefficients in the LUTs per operation. In addition, the PPA strategy is the common base for commercial implementations of SFUs and several works in literature addressed optimization targets by resorting to analyses on their structural parameters to improve the performance of PPA-based SFU cores [41, 4]. In [20], the authors introduce a Dual-Channel Multiplier that focuses on optimizing the hardware multipliers (P Prod) to reduce energy and area. Other strategies include several pipeline stages to improve performance [5], while different approaches focus on compressing and reducing the memory tables (LUTs) through bank partitions [43], bit partitioning [28], and through the adjustment (i.e., assignation of special constraints) of the polynomial coefficients ($C_0$, $C_1$, and $C_2$) of adjacent segments to reduce the overall LUT size [18]. In [31], the authors combine functional units (e.g., ADD and MUL cores) with PPA-based SFU structures to improve the system's data path, as well as reduce the overall area and power of large parallel processors.

#### 2.2.2 Architecture of Modular SFUs

Modular SFUs integrate multi-functional architectures, implementing each function as an individual hardware unit. Each function adopts the most suitable approximation algorithms to guarantee the best balance between accuracy and performance in the core. Figure 1 (*right*) illustrates the scheme of a modular SFU implementing five transcendental functions $sin(x)$, $cos(x)$, $\frac{1}{\sqrt{x}}$, $2^x$, and $\log_2(x)$ resorting to four computational sub-units.

The first sub-unit implements the CORDIC algorithm [49] to evaluate the $sin(x)$ and $cos(x)$ operations. The $\frac{1}{\sqrt{x}}$ operation employs the *Fast Inverse Square Root* algorithm (FISR) implementing an approximation step by evaluating the function $\frac{1}{\sqrt{x}} = 2^{-0.5\times\log_2(x)}$, taking advantage of the logarithmic representation when the bit-wise floating-point operand is interpreted as an integer. Then, a Newton-Raphson iteration refines the output result to reduce the error. The $\log_2(x)$ and $2^x$ functions employ an *Adaptable Logarithm Approximation* (ALA) [1], which is a PPA variation for the execution of exponential and logarithm operations in hardware.

## 3 Methodology for Evaluating, Analyzing and Reducing the Impact of Transient Faults in SFUs

Our evaluation is divided into three stages: *i)* the evaluation and analysis of transient fault effects in SFUs and their relation with their internal structures. *ii)* A combined analysis of the area, power, performance, and reliability of both SFU architectures. *iii)* The exploration, modeling, and evaluation of coarse-grain selective-hardening mechanisms for SFUs. Figure 2 depicts a general scheme of the method to characterize fault effects and explore selective-hardening mechanisms for both SFU architectures in GPUs. For our evaluation, two versions of the *FlexGripPlus* GPU have been created, each including a different SFU implementation (**GPU1** with the **SFU1**, and **GPU2** using **SFU2**). The following subsections describe the primary targets for each stage of the evaluation.

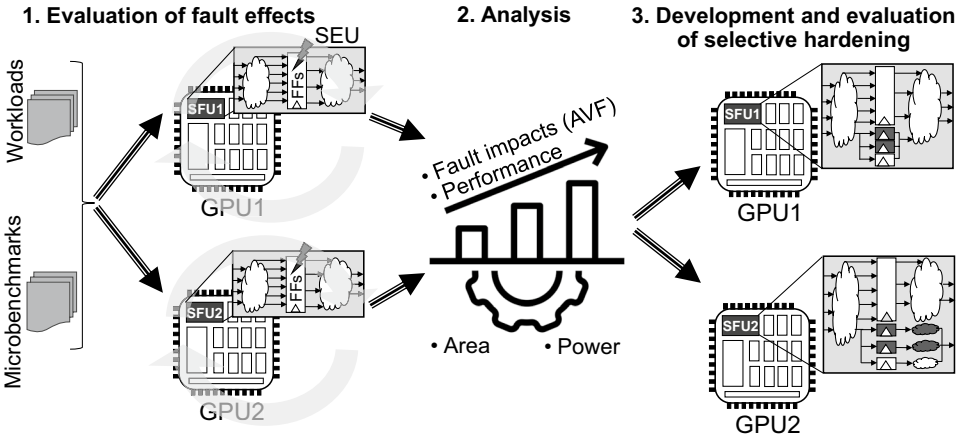



**Fig. 2** A general scheme of the method used to characterize fault effects, analyze their impacts on the architecture of SFUs and develop selective hardening mechanisms

### 3.1 Reliability Evaluation of the SFU's Architecture

The characterization of fault effects on the SFU architectures exploits an statistical-based fault injection approach that comprises fault-injection campaigns determining the *Architectural Vulnerability Factor* (AVF) [36] on both GPUs (**GPU1** and **GPU2**). Each injection campaign involves several logic faulty simulations that exhaustively target all available flip-flops (FFs) in both SFUs. In detail, every campaign randomly inject (in time) an individual *Single Event Upset* (SEU) on one targeted fault site and then a complete simulation is executed. This procedure is exhaustively repeated for each fault site in the SFU core. In modern generation devices, the SEU fault model represents state changes in the system's structures caused by one single ionizing particle (e.g., ions, electrons, photons) striking a sensitive node. Since, these changes temporarily affect and modify the content of memory cells or storage elements (e.g., FFs) in a system, we represent a SEU as the bit-flip on one targeted site (flip-flop) of an SFU. Then, we observe the hardware fault effects at the output of the GPU system, considering the fault propagation and the corruption on a running application. We employ an RT Level description of the GPU and SFU units for the experiments.

We used two application types as input workloads for the fault characterization: *1)* Representative GPU applications employing SFUs (i.e., from the Rodinia tool suite and NVIDIA SDK samples), and *2)* carefully designed micro-benchmarks to address individual SFU operations (*FSIN*, *FCOS*, *RSQRT*, *EXP2*, and *LOG2*). Each micro-benchmark includes exclusive instructions for every operation and resorts to a considerable amount of input data operands to excite the SFU's sub-units and propagate faults.

For the experimental evaluation, we adapted a custom fault injection environment [9] to target each flip-flop in the SFUs of both GPUs. Our approach takes advantage of the operative times of the SFUs on the parallel workloads an only inject faults on these operative intervals, so reducing the overall simulation times. In particular, our environment randomly selects a fault-injection time (clock cycle) according to the active execution times of the SFUs per application (i.e., only when executing SFU instructions/operations) [50]. Then, a fault site is targeted and the fault is placed. The simulation resumes and continues until it is finished. It must be noted that preliminary fault-free profiling executions, on the parallel workloads, provide the active intervals of the SFU cores that support the selection of the injection times (clock cycles) to be used during the fault injection campaigns. The output results (from the GPU's memory) are collected and retrieved for later evaluation and fault classification.

Faults are classified according to the output effect on the applications as: *i) Detected Unrecoverable Error* (DUE) that is caused when the fault hangs or crashes the execution of the application and results are not available, *ii) Silent Data Corruption* (SDC) when the impact of a fault is propagated to the outputs of the applications and corrupts the results, and *iii) masked* when the fault effect does not affect the application's operation and the module's functionality in the GPU.

### 3.2 Evaluation of Area, Power, and Performance in SFU's Architectures

To evaluate the cost of area, power, and performance of both SFU architectures, we consider the SFU gate-level implementations in two cases: *i)* stand-alone evaluation (i.e., determining their individual architectural features) and *ii)* evaluation when integrated with the complete GPU core (SM cores instrumented with each SFU).

We perform the logic synthesis on both SFUs, using the same technology library for the units inside the GPU cores and targeting the same operative performance (e.g., maximum operative frequency). In the evaluation, we employ the instrumented GPUs (**GPU1** and **GPU2**) to evaluate the architectural features. The power consumption analysis considers the 50% of switching activity and the maximum

obtained operative frequency per unit. Similarly, we compare the relative area cost of each SFU inside a GPU core.

As a result of the comparisons, we correlate four main parameters: the relative area size, the power budget, the operational latency, and the fault vulnerability for both SFU architectures to analyze the best trade-off of both SFUs for GPUs.

### 3.3 Exploring and Evaluating Selective hardening mechanisms for SFUs

Our evaluation and analysis of fault-tolerance structures aim at identifying internal structures and crucial targets to increase the reliability of an SFU unit, considering their internal organization. For this purpose, this stage explores and evaluates hardware-based hardening mechanisms for SFUs by resorting to one passive hardening strategy (i.e., *Triple Modular Redundancy* or TMR).

First, we characterize the structures of the sub-units in both SFU cores. Then, we identify the primary and alternative hardening configurations following coarse-grain schemes according to the SFU's internal structures and the results from the reliability evaluation performed in the first stage, see Subsection 3.1. Consecutively, we implement each hardening configuration to evaluate each hardening configuration's structural features (e.g., area, power, and performance). Finally, we characterize, model, and evaluate the reliability features of each hardening configuration by resorting to reliability functions of probability and *Reliability Block Diagram* (RBD) [23] analyses. As a reference for comparison, we apply the complete passive hardening on both SFU architectures.

## 4 Reliability Evaluation of SFUs

This section describes the experiments and the result analyses of the reliability evaluation on SFU architectures. We consider the workloads and their impact on the activity of the targeted operation inside the GPU. In our experiments, the configuration of the two instrumented GPUs (**GPU1** and **GPU2**) includes one SM cluster, one SM per SM cluster, 32 parallel cores, and 4 SFUs per SM. Each SFU accounts for a total number of flip-flops (FFs) equal to 134 and 720 in **SFU1** and **SFU2**, respectively, which are the targets during the fault injection campaigns. The reliability evaluation experiments are performed on a server of 12 Intel Xeon CPUs running at 2.5 GHz and with 256 GB of RAM.

We employ five representative parallel applications (*NN*, *Back Propagation* or 'BP,' *Euler3D*, *Gaussian*, and *Image Denoising* or 'ImDen') from the NVIDIA Samples SDK and the Rodinia Tool suites [8]. Each application includes one or several instructions explicitly addressing the SFUs. Similarly, we encoded five micro-benchmarks to excite specific structures performing each operation. More in detail, we applied a set of 2,048 sample operands following their operational ranges (i.e., *FCOS* and *FSIN* use operands in range $[0, \pi/2]$, *FEXP* employs values in the range [0, 1), *FRSQRT* with values in range [1, 4), and *FLG2* with values in range [1, 2)). During the evaluation procedures, the selected operational ranges skip the dependency and use of additional operations and their associated hardware (e.g., range reduction operations or *RRO* instructions). The kernel configuration of each micro-benchmark exploits the maximum number of concurrent threads (1,024) per SM to excite each SFU. It is worth noting that we distribute the sample values to apply the same operands among the 4 SFUs per SM. Thus, a total of 8,192 threads are submitted per micro-benchmark to operate the sample values in the available SFUs per SM.

In the evaluation, we performed a total of 20 fault injection campaigns on both versions of the GPUs (accounting for the number of GPUs × number of workloads). Our evaluation considers the exhaustive fault injection of transient faults (SEUs) in all available sites (FFs) of one of the available SFUs in the GPU core, following the evaluation approach described in Subsection 3.1, so considering all possible fault impacts as the product of the architectural features on the evaluated SFUs.

We employ the approach described in [32] to determine the minimal amount of faults to be evaluated per fault campaign on a given workload, considering an interval of confidence of at least 95% for each evaluated workload. In practice, the total number of faults in a campaign is proportional to the number of faults injected per site across the execution time of the workloads. Thus, we injected, on average, a set of 26 faults per hardware site, representing a total of 3,484 fault injections in **SFU1** and 18,720 fault injections in **SFU2** (per evaluated application), and stem from more than $2.15 \times 10^5$ injected and characterized faults in both SFUs. The fault injection campaigns provide the reliability assessment of each flip-flop on both SFUs, as well as the fault effects on the running workloads. It is worth noting that each fault campaign considers a random injection time targeting only those intervals when the SFUs are active.

We perform two evaluation on the SFU cores: *1)* Structural evaluation of the SFUs and *2)* Application level impact effects from faulty SFUs. First, we determined the impact affect of transient faults on the structures of the SFUs for each GPU. In this case, our main target is to analyze the micro-architecture vulnerability of each SFU architecture. Figure 3 reports the normalized AVF results for both SFU architectures, considering the number of identified error effects divided by the total number of injected faults. In

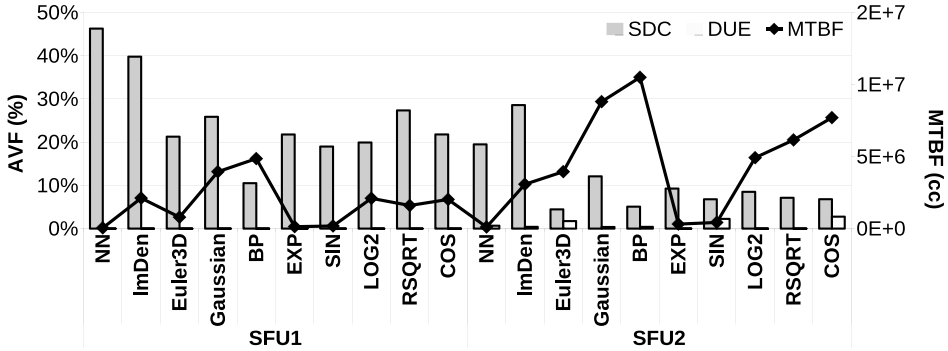



**Fig. 3** AVF and MTBF for the evaluated workloads in both SFU architectures

general and for all workloads, the reported results demonstrate that the internal structures (in particular, the associated FFs) of *fused* SFUs (**SFU1**) are more vulnerable to faults than those in *modular* SFUs (**SFU2**). In some cases, the normalized percentage of SDCs increases from about 5% to 47%. Our exhaustive evaluation of each FF, in both SFUs, suggests that faults affecting one of the input registers highly promote their propagation to the primary outputs and the result's corruption.

In detail, the reiterated use of the same hardware structures in **SFU1** to calculate different operations promotes equivalent fault effects for each operation. Furthermore, faults corrupting sites near the output ports in **SFU1** directly corrupt the results. In contrast, faults in a *modular* SFU (**SFU2**) are mainly related to the type of an executed operation since each sub-unit processes different operations, so only those faults inside the hardware sub-units are prone to impact the result. In fact, the micro-benchmark results show that only faults affecting any hardware site used for the execution of a given operation are propagated and produce corruption effects. A deep analysis of the corrupted results and their fault source reveals that the 'Output selector logic' (OSL) sub-unit (near the primary outputs) is highly vulnerable to faults (from 15% to 25% of observed faults for all workloads). Furthermore, the identified DUEs in **SFU2** (from 1% up to 4%) are the product of faults affecting the internal controllers (e.g., controller status, control signals, and iteration counters) in the implementation of an iterative CORDIC sub-unit for *SIN* and *COS* operations.

To observe the impact effects of faulty SFU at the application and system level, we calculated the *Mean Time Between Failures* (MTBF) [42], considering a constant flux as *1/application_time(cc)*, and the cross-section of each SFU as the ratio between the total number of identified SDCs and the total amount of injected faults. The MTBF combines the timing effects from each evaluated application with reliability assessment parameters. In particular, we consider those faults that propagate across the application and cause corruptions on the results (*SDCs*). In general, the experimental results, show that on most of the applications (*BP*, *Gaussian*, *Euler3D*, *ImDem*, *LOG2*, *RSQRT* and *COS*) using a *modular* SFU (SFU2) clearly have more operative time between failures (i.e., more reliable), in terms of clock cycles or (cc), than the same applications using a fused SFU (SFU1). These results suggest that applications are less susceptible to faults in a *modular* SFU architecture than in a *fused* one, so supporting the idea that **modular** SFU architectures can be considered as feasible reliable alternatives for SFU integration in GPU architectures. In particular, the frequent use of the SFU cores by several of the analyzed parallel workloads (*BP*, *Gaussian*, *Euler3D*, and *ImDem*) seems to be a key factor for the propagation of fault effects on the results from an SFU affected by transient faults. Interestingly, we also observed that some micro-benchmarks (*LOG2*, *RSQRT*, and *COS*), which are focused on specific SFU operations, show equivalent rises in the execution time between failures. Thus, these preliminary experimental results indicate that the architecture of the SFU plays a crucial role on the activation and propagation of faults for heterogeneous applications (i.e., using several GPU resources and instructions), as well as in fully embarrassingly parallel applications devoted to use the targeted SFU cores. We also observed that some micro-benchmarks (e.g., *EXP* and *SIN*) show a minimal rise in the operative time between failures (MTBF). A detailed analysis on both benchmarks show that these are encoded and described as the others (e.g., using the same amount of machine instructions and number of operands). However, it seems that the analyzed data workload (uniformly distributed for the operative ranges on both workloads) affects the activation and propagation of faults effects. Although the difference of MTBF among SFUs is minimal for both micro-benchmarks, in comparison with other applications, the results still support the idea that modular SFU architectures are feasible alternatives to improve the execution time between failures on applications.

An additional analysis was performed on the NN workload. In particular, this application presented a constant behavior of MTBF for both SFU architectures. Interestingly, the micro-architecture results show a considerable percentage of faults producing SDCs (46% in SFU1 and 19.5% in SFU2). However, the overall execution time (cc) of the application during the experiments reduced the structural

**Table 1** A comparison of the relative size of SFUs and other functional units and the GPU core

| | Area ($\mu m^2$) | Area w.r.t. a SP core (%) | Area w.r.t. a FP32 core (%) | Area w.r.t. a GPU core (%) |
|---|---|---|---|---|
| **SFU1** | 3,651.4 | 317,5 | 37,5 | 4,6 |
| **SFU2** | 5,095.5 | 443,1 | 52,3 | 6,4 |
| **SP** | 1,149.9 | 100.0 | - | - |
| **FP32** | 9,735.0 | - | 100.0 | - |
| **GPU core** | 315,347.9 | - | - | 100.0 |

impact of the SFU architecture when affected by transient faults. Our analysis indicates that the particular encoding of the application, as well as the limited amount of SFU instructions in the parallel application's algorithm are the main factors masking the structural impacts of SFUs at the application level.

Our experimental results indicates that embarrassingly parallel micro-benchmarks on SFUs (that represent fragments from large parallel applications) and heterogeneous parallel workloads, which use several GPU resources (e.g., SFUs, SPs, and FP32 cores) and their associated instructions, are more resilient to transient faults on modular SFU architectures than on fused ones. Furthermore, we observed that in some cases (e.g., NN application) the code description contributes to mask effects at the application level from soft-errors (i.e., transient faults) arising on the SFUs.

## 5 Evaluation of Performance, Area, and Power Analysis of Architectures in SFUs

The first evaluation targets the individual implementation of each SFU (**SFU1** and **SFU2**) considering a logic synthesis of 15nm technology library [35] targeting a frequency of 500MHz.

Table 1 shows the relative percentage of area occupied by each SFU unit compared to other functional units (SP and FP32 cores) and the complete logic of a GPU core for the 15nm logic synthesis. As the base for the area comparison, the synthesis of the GPU cores includes 8 FP32, and 8 SP cores. Thus, SFU cores are excluded from the GPU cores logic, and the obtained percentages represent the overhead cost of including SFUs from each architecture. Despite the relatively low area of SFUs in comparison to a complete GPU core (4.6% in **SFU1** and 6.4% in **SFU2**), SFU units are crucial cores of fundamental importance. In particular, SFU1 cores might be feasible to improve area usage in large GPU designs. Moreover, the comparison of SFUs with other functional units shows that SFUs are comparable in area to SP cores (from three to more than four times the area) and FP32 units (almost third or half the size).

For the individual evaluation of performance, cells and area sizes, and power consumption of the SFUs, Table 2 reports the obtained results of the 15nm synthesis of both SFUs targeting an operative frequency of 500MHz. To calculate the performance effect of each architecture, we analyzed the longest path for both circuits. The results unsurprisingly suggest that *modular* SFUs are more costly in terms of size (area and used resources) than *fused* SFUs. In fact, as initially anticipated, *fused* SFUs are more area efficient than *modular* SFUs (in around 27% of area and resources). Moreover, the performance of *fused* SFUs is higher than the *modular* ones, which is mainly caused by bottlenecks on the iterative units for trigonometric operations (e.g., *CORDIC* algorithm). Interestingly, both implementations show that *modular* implementations are slightly more power efficient than *fused* SFUs (in around 36.6%). In the modular SFU, the used core is the only active (triggered) to perform a given operation, while the others remain inactive.

To analyze, correlate and compare the complete features of both SFU architectures, we associate four main features for comparison purposes: *i)* the relative size (*RSize*) of SFUs calculated as the ratio between each SFU unit and the total size of the complete GPU core, using the results from the logic synthesis implementation; *ii)* the power consumption (*PWC*), from the gate level implementation, *iii)* the Operational Latency (*OPL*), as a normalized average of the number of clock cycles required to execute each operation (*SIN*, *COS*, *EXP2*, *LOG2*, *RSQRT*) in the SFUs, and *iv)* the fault impact produced by each SFU architecture, and calculated as a preliminary average *AVF* ($AVF_{AVG}$) from the analyzed applications (see Fig. 4).

The observed trends on both SFUs allow us to determine each unit's possible advantages and constraints when integrated into a GPU. In particular, from the normalized behaviors, it can be observed that *modular* SFUs, see Fig. 4

**Table 2** Main features of Size, power, and performance of both gate-level SFUs implemented at 15nm

| | Frequency (*MHz*) | Size | | Power (*mW*) | Performance (*ns*) |
|---|---|---|---|---|---|
| | | Area ($\mu m^2$) | Cells | | |
| **SFU1** ***(Fused)*** | 500.0 | 3,651.4 | 11,423 | 756.1 | 1.7 |
| **SFU2** ***(Modular)*** | 500.0 | 5,095.5 | 13,170 | 554.4 | 1.9 |

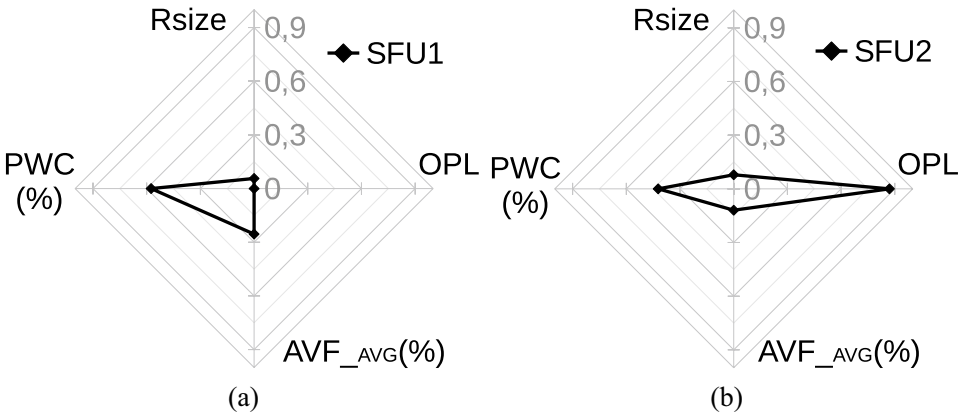



**Fig. 4** A comparison of relative size (*RSize*), relative power consumption (*PWC*), operational latency (*OPL*), and average Architectural Vulnerability Factor ($AVG_{AVF}$) for the **a** *Fused* (**SFU1**) and **b** *Modular* (**SFU2**) architectures

(right), are less vulnerable to faults but increase a GPU's relative area cost and power consumption. In contrast, *fused* SFUs, see Fig. 4 (left), are more area and energy efficient but more vulnerable to propagate fault effects. In addition, these architectures introduce minimal operational latency in the execution of the intended operations (i.e., better performance). Current design approaches focus on performance, area, and power consumption, and the same applies to SFUs. Interestingly, our results suggest that GPU designs focused on reliability might consider alternative SFU architectures with better reliability features and feasible power budgets, such as *modular* architectures. Unfortunately, the operational latency (OPL) in the *modular* SFU is higher than in the *fused* one, mainly due to the iterative sub-units (Cordic core). Thus, competitive *modular* SFUs might require advanced and non-iterative algorithms to replace the Cordic code and reduce the overall operational latency of the SFU unit. Similarly, *Fused* SFUs might exploit schemes of sub-unit gating approaches to reduce energy consumption.

## 6 Fault Mitigation on SFUs: Evaluating Selective Hardening Approaches

In this Section, we explore and evaluate hardware-based hardening mechanisms for SFUs. First, we analyze the architecture of both SFUs (SFU1 and SFU2), revealing the primary sub-unit in both designs. The identification of the sub-units of each SFU considers the structural sources for most identified errors during the reliability characterization in Section 4. Hence, for our exploration, *fused* SFU comprise *i)* the ROM-tables (LUTs), *ii)* the square unit ($X^2$), *iii)* the array of partial products and fused accumulator (PPFAs), and *iv)* the normalization logic (NL), which was identified as a significant source or data corruption, see Fig. 1. Similarly, *modular* SFU includes *i)* individual operational cores (e.g., Cordic, ALA, and FISR units) and *ii)* the output selector logic (OSL) that is highly sensitive to fault propagation. Figure 5 illustrates the occupied area for each sub-structure in both SFU cores.

According to the internal organization and occupancy of the sub-units in both SFUs, we define several targets to explore and estimate coarse-grain selective hardening. The complete hardening of the *fused* SFU1 is defined as the **R11** configuration for our analyses. A second hardening scheme considers the Rom-tables, the square unit, and the array of partial product units is **R12**. Moreover, the third hardening scheme (**R13**) focuses only on the square unit and the array of partial product units. Similarly, we determine the complete hardening of the *modular* core (SFU2) as **R21**. One selective hardening scheme targets the operational cores only as **R22**. Finally, a third configuration targets the hardening of the output selector logic only as **R23**.

We implement each selective hardening configuration (**R12**, **R13**, **R22**, and **R23**) and the complete hardening schemes (**R11** and **R21**) on the RT-level descriptions of both SFUs (SFU1 and SFU2). Then, each hardened configuration is verified and validated using an FPGA platform (*Intel DE2-115*, Cyclone IV EP4CE115F29). Table 3 reports the used area (in terms of *Logic Elements* or LEs), the *Total Thermal Power Dissipation* or TTPD, and the performance impacts for each hardening configuration.

Interestingly, in the case of SFU1, the reported results show that the complete hardening configuration (**R11**) affects the performance and reduces its maximum operative frequency by up to 15.5%. Moreover, **R11** represents an overhead of 77.6% in area and 5.5% in additional power consumption in the FPGA implementation. In contrast, the complete hardening of SFU2 (**R21**) increases the area and power consumption overhead at 76.4% and 6.03%, respectively, while affecting the performance at around 8.1%. A direct comparison of the relative impact in area and performance shows that the overhead in the area and power consumption are similar for both SFUs. However, our results show that the evaluated implementation of **R21** produces lower effects in performance than the equivalent hardening on SFU1 (**R11**). In contrast, the evaluation of the selective hardening configurations shows that the overhead costs for the SFU1 cases (**R12** and **R13**) are relatively slower costs

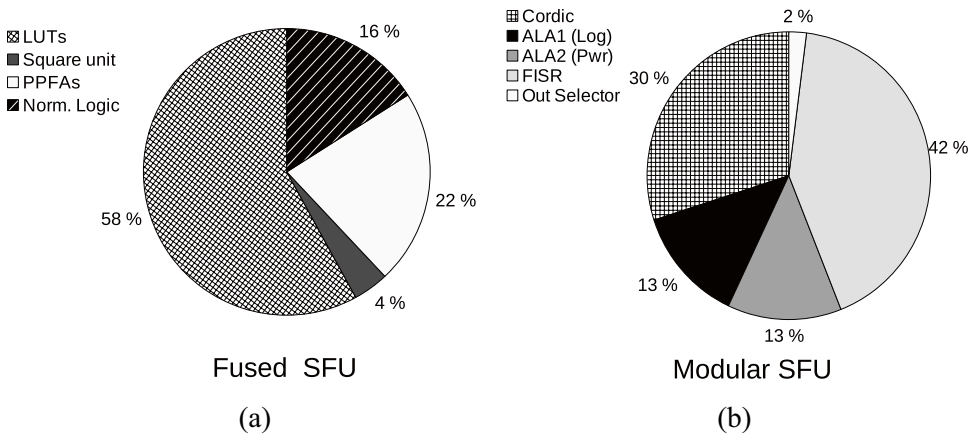



**Fig. 5** Percentage distribution of occupied FPGA's area by the sub-modules of the *fused* SFU (**a**) and *modular* SFU (**b**)

than those for the selective hardening version of SFU2 (**R22** and **R23**). In particular, **R13** and **R23** configurations cost less than 10% of the additional area on both SFUs. On the other hand, aggressive selective hardening solutions, such as **R12** and **R22** increase the area and power costs by up to 75.0% and 3.9%, respectively, while affecting the performance in up to 11.1%.

In particular, **R12** configuration includes the LUTs as part of the hardening in SFU1, which are the main ones responsible for the considerable overhead costs. Alternative methods for memory hardening, such as Error Correcting Codes (ECCs), would be more effective in the LUTs and can contribute to reducing the area overhead in this configuration. In contrast, the observed area overhead in the **R22** can hardly be reduced since custom logic for each operation is mainly involved.

To evaluate the impact on reliability and fault-tolerance of each hardened configuration, we estimate individual reliability functions based on the probability of correct operation of the units in combination with *RBD* analysis to include the structural composition of each SFU as part of our reliability model.

Since the operation of SFU1 requires the serial execution of several sub-units, we define the probability of correct execution as a serial relation of the probabilities for each sub-unit, as expressed in Eq. 1.

$$R_{SFU1} = R_{LUTs} \cdot R_{X^2} \cdot R_{PPFAs} \cdot R_{NL} \tag{1}$$

where $R_{LUTs}$, $R_{X^2}$, $R_{PPFAs}$, and $R_{NL}$ are the probability functions of the ROM-tables, square unit, array of partial products and fused accumulator, and normalization logic, respectively. Thus, the probability function representing the TMR hardening of the complete SFU ($R_{11}$) is described in Eq 2.

$$R_{11} = 3R_{SFU_1}^2 - 2R_{SFU_1}^3 \tag{2}$$

Similarly, we determine the probability functions of reliability when hardening the ROM tables, square unit, and the array of partial products ($R_{12}$), as well as the probability function for the square unit and the array of partial products ($R_{13}$), which are depicted in Eqs. 3 and 4, respectively.

$$R_{12} = \left(3\left(R_{LUTs} \cdot R_{X^2} \cdot R_{PPFAs}\right)^2 - 2\left(R_{LUTs} \cdot R_{X^2} \cdot R_{PPFAs}\right)^3\right) \cdot R_{NL} \tag{3}$$

$$R_{13} = R_{LUTs} \cdot \left(3R_{X^2}^2 - 2R_{X^2}^3\right) \cdot \left(3R_{PPFAs}^2 - 2R_{PPFAs}^3\right) \cdot R_{NL} \tag{4}$$

As represented in $R_{12}$ and $R_{13}$, the targeted units for hardening affect the computation of the equivalent probability function of reliability.

In the case of SFU2, we follow a similar procedure to determine the probability functions of reliability for the complete ($R_{20}$) and the selective hardening configurations ($R_{21}$ and $R_{22}$). In particular, the organization of the sub-units in a parallel and serial fashion implies that

**Table 3** Performance and overhead results for the hardening configurations

| Config | LEs | TTPD (mW) | Max. Frequency (MHz) | Config | LEs | TTPD (mW) | Max. frequency (MHz) |
|---|---|---|---|---|---|---|---|
| ***SFU1 (Fused)*** | 6,754 | 160.65 | 10.3 | ***SFU2 (Modular)*** | 8,639 | 158.61 | 9.9 |
| ***R11*** | 11,996 | 169.49 | 8.7 | ***R21*** | 15.243 | 168.18 | 9.1 |
| ***R12*** | 10,273 | 166.96 | 9.7 | ***R22*** | 15.124 | 164.09 | 8.8 |
| ***R13*** | 7,245 | 163.57 | 10.1 | ***R23*** | 9.075 | 159.88 | 9.7 |

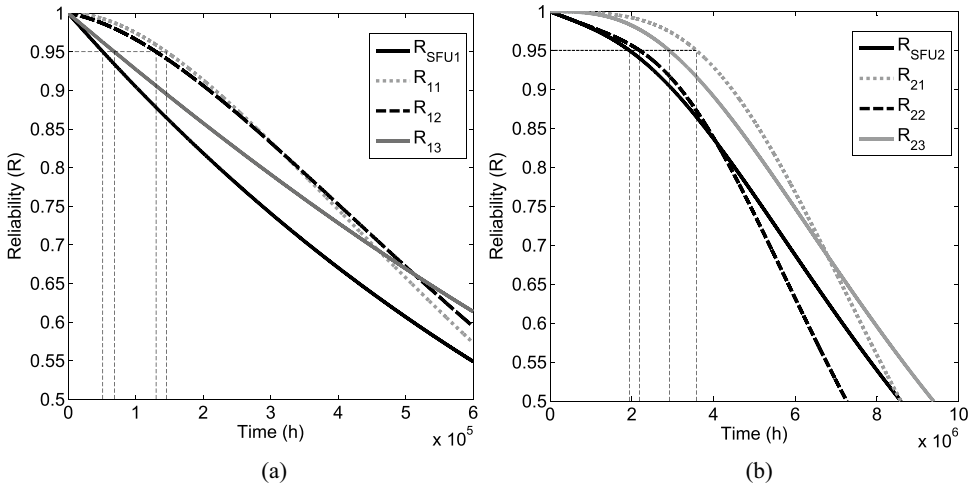



**Fig. 6** Impact in the reliability of the different selective and complete hardening configurations for **a** SFU1 and **b** SFU2

the operation of the SFU directly depends on the targeted operation (and its particular hardware unit) and the OSL unit. Equation 5 represents the probability function of reliability for the SFU2, where $R_{Cordic}$, $R_{FISR}$, $R_{ALA1}$, $R_{ALA2}$, and $R_{OSL}$ represent the probability of correct operation of the Cordic, FISR, ALA1 (*logarithmic*), ALA2 (*power*), and OSL units, respectively.

$$R_{SFU_2} = \{1 - [(1 - R_{Cordic}) \cdot (1 - R_{FISR}) \cdot (1 - R_{ALA1}) \cdot (1 - R_{ALA2})]\} \cdot R_{OSL} \quad (5)$$

The reliability functions for the complete TMR hardening of SFU2 ($R_{21}$) is equal to the expression in Eq. 2. Furthermore, Eqs. 6 and 7 describe the probability functions for the reliability of the selective hardening targeting the operational units and the output selector logic (OSL), respectively.

$$R_{22} = R_{OSL} \cdot (1 - [1 - (3R^2_{Cordic} - 2R^3_{Cordic})] \cdot [1 - (3R^2_{FISR} - 2R^3_{FISR})] \cdot [1 - (3R^2_{ALA1} - 2R^3_{ALA1})] \cdot [1 - (3R^2_{ALA2} - 2R^3_{ALA2})]) \quad (6)$$

$$R_{23} = 1 - [(1 - R_{Cordic}) \cdot (1 - R_{FISR}) \cdot (1 - R_{ALA1}) \cdot (1 - R_{ALA2})] \cdot (3R^2_{OSL} - 2R^3_{OSL}) \quad (7)$$

To analyze and validate the main benefits in the reliability of the different selective hardening configurations, we evaluate each probability function replacing the probability function for the typical function on time: $R = e^{-\lambda t}$. We employ a typical rate of failures in time of $10^{-6} faults/h$. and the area occupation of each sub-unit in the SFUs, see Fig. 5, to calculate the individual probability function of the sub-units. Thus, in SFU1, we determine $\lambda_{LUTs} = 5.8x10^{-7}$, $\lambda_{X^2} = 4.0x10^{-8}$ , $\lambda_{PPFAs} = 2.2x10^{-7}$ , $\lambda_{NL} = 1.6x10^{-7}$ . Similarly, for SFU2, we determine $\lambda_{Cordic} = 3.0x10^{-7}$, $\lambda_{FISR} = 4.2x10^{-7}$, $\lambda_{ALA1} = 1.3x10^{-7}$, $\lambda_{ALA2} = 1.3x10^{-7}$, and $\lambda_{OSL} = 2.0x10^{-8}$.

Figure 6 depicts the changes in reliability in time (Failures in Time or FIT) for each selective hardening configuration in SFU1 and SFU2, respectively. As depicted in both cases, the complete hardening extends across the time the probability of correct operation of the SFUs. In general, The observed reliability degradation on SFU1 is associated with the structural organization of the fused SFU core. In this case, the probability of correct execution depends on the number of sub-units serially connected to process an operation and provide a result. Since SFU1 requires the proper operation of most of the units inside the core (four sub-units), its probability of correct operation *(Reliability)* is influenced by each sub-unit and behaves almost linearly for the observed time interval. Moreover, the probability of correct operation of SFU1 is lower than the probability of correct operation of SFU2, which only involves two serially connected sub-units for its correct operation.

Regarding the selective-hardening mechanisms for SFU1, **R11** and **R12** behave in similar proportions indicating that the latter could be a feasible configuration to provide equivalent reliability benefits to the complete hardening on the SFU. On the other hand, for SFU2, its clear that **R23** (protecting the OSL unit) provides more reliability benefits than **R22** (protecting the individual operational cores) since all operations in the SFU employ the OSL structures and faults arising on this units can directly compromise the output results. Moreover, the minimal area overhead in **R23** configuration is a feasible candidate for selective hardening of the SFU2 core.

In our evaluation, we define several coarse-grain selective hardening configurations for both SFUs. As expected, our results suggest that the structural organization plays a crucial role in the reliability of SFUs. In fact, each sub-units in both SFUs impact differently the reliability of each core. In our exploration of selective hardening configurations, we focused on several units that are critical for the operation of the SFUs. In some cases, the protected units massively increased the overhead costs (e.g., area) with moderate reliability benefits (e.g., LUTs in SFU1). Moreover, our analyses targeted critical units, such as the OSL structures, employed in each SFU2 operation. In this case, modeling results demonstrate an increase in the reliability benefits with minor overhead costs.

Although we mainly focused our evaluation on the reliability of SFU architectures as a vital non-functional property, our results in Figs. 4 and 6, determine the importance of evaluating and modeling the reliability in SFUs as a complementary instrument and parameter for the design and integration of modern systems. Interestingly, our results suggest that *fused* SFUs are adequate solutions in terms of performance and size. However, other emerging design alternatives, such as *modular* SFUs, might become feasible solutions when considering reliability features. In fact, a comparison between the reliability features of both SFUs ($R_{SFU1}$ and $R_{SFU2}$), see Fig. 6 shows that the probability functions for the *modular* SFUs behave better in time and increase the reliability of the unit in up to one order of magnitude.

## 7 Conclusion and Future Work

This work focused on evaluating and investigating the incidence of the structural features of two SFU architectures for GPUs and the impacts of transient faults effects on reliability. According to the results, the fault characterization and evaluation shows that *fused* SFU architectures (base of commercial devices) are adequate solutions in terms of area and performance, but these architectures are more vulnerable to fault effects than *modular* SFUs. The multi-functional use of the internal structures in *fused* SFUs seems to be the main factor increasing the sensitivity to faults.

A comparison of area, power, and operational latency in relation to the complete GPU core suggests that *fused* SFUs are more area and performance efficient, but demand more power budget than *modular* SFUs. The outcomes of our analyses are intended to include reliability features of the architecture as a relevant design parameter, such as area, power, and performance, in the design of functional units for hardware accelerators, such as GPUs, for the safety-critical domain.

Our exploration, modeling, and evaluation of selective hardening solutions for SFUs show that *modular* architectures behave better in time in up to one order of magnitude when compared to *fused* ones. Furthermore, the evaluated hardening configurations show that some sub-units directly affect the overall reliability of an SFU, so aiming at more effectively protecting the unit against faults with minor overhead costs.

In the future, we plan to evaluate the fine-grain reliability of other crucial structures in hardware accelerators, such as the Tensor Core units for the proposal of multi-level error models and hardware-based hardening solutions.

**Funding** Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. This work has been supported by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing.

**Data Availibility** The low-level micro-architecture SFU cores (*Fused* and *Modular*) that were used in the current study are open-source and available following the links included in manuscript. The datasets generated during the experiments of the current study are available from the corresponding author upon reasonable request.

## Declarations

**Conflicts of Interests/Competing Interests** The authors declare no relevant financial or non-financial conflict interests. This work was partially sponsored and funded by the National Resilience and Recovery Plan (PNRR) under the Fondazione ICSC Centro Nazionale di Ricerca in High Performance Computing, Big Data e Quantum Computing (National Center for HPC, Big Data and Quantum Computing) - SPOKE 1 FUTURE HPC & BIG DATA.



## References

1. Bariamis D, Maroulis D, Iakovidis DK (2010) Adaptable, fast, area-efficient architecture for logarithm approximation with arbitrary accuracy on fpga. J Signal Process Syst 58(3):301–310
2. Basso PM, Santos FFd, Rech P (2020) Impact of tensor cores and mixed precision on the reliability of matrix multiplication in gpus. IEEE Trans Nucl Sci 67(7):1560–1565
3. Bayoumi A, Chu M, Hanafy Y et al (2009) Scientific and engineering computing using ati stream technology. Comput Sci Eng 11(6):92–97
4. Bellal R, lamini ES, Belbachir H, et al (2019) Improved affine arithmetic-based precision analysis for polynomial function evaluation. IEEE Trans Comput 68(5):702–712
5. Chang Y, Wei J, Zhao G et al (2013) A novel architecture of special arithmetic function unit for area-efficient programmable vertex shader

6. Chatzidimitriou A, Kaliorakis M, Gizopoulos D et al (2017) Rt level vs. microarchitecture-level reliability assessment: Case study on arm(r) cortex(r)-a9 cpu. In: Proceeding of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), p 117–120
7. Chatzidimitriou A, Gizopoulos D (2016) Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy. In: Proceeding of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), p 69–78
8. Che S, Boyer M, Meng J et al (2009) Rodinia: A benchmark suite for heterogeneous computing. In: Proceeding of the 2009 IEEE International Symposium on Workload Characterization (IISWC), p 44–54
9. Condia JER, Azambuja JR, Sonza Reorda M et al (2020) Analyzing the sensitivity of gpu pipeline registers to single events upsets. In: Proceeding of the 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), p 380–385
10. Condia JER, Du B, Sonza Reorda M et al (2020) Flexgripplus: An improved gpgpu model to support reliability analysis. Microelectron Reliab 109(113):660
11. Condia JER, Guerrero-Balaguera J, Moreno-Manrique C et al (2020) Design and verification of an open-source sfu model for gpgpus. In: Proceeding of the 17th Biennial Baltic Electronics Conference (BEC 2020), p 1–6
12. Condia JER, Guerrero-Balaguera JD, Patiño Núñez EJ et al (2023) Evaluating the prevalence of sfus in the reliability of gpus. In: Proceeding of the 2023 IEEE European Test Symposium (ETS), p 1–6
13. Condia JER, Guerrero-Balaguera JD, Patiño Núñez EJ et al (2023) Analyzing the architectural impact of transient fault effects in sfus of gpus. In: Proceeding of the 2023 IEEE 24th Latin American Test Symposium (LATS), p 1–6
14. Condia JER, Faggiano R, Sonza Reorda M (2022) Microarchitectural reliability evaluation of a block scheduling controller in gpus. In: Proceeding of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), p 26–31
15. Condia JER, Rech P, Santos FFD et al (2022) An effective method to identify microarchitectural vulnerabilities in gpus. IEEE Trans Device Mater Reliab 22(2):129–141
16. Condia JER, Santos FFd, Sonza Reorda M et al (2021) Combining architectural simulation and software fault injection for a fast and accurate cnns reliability evaluation on gpus. In: Proceeding of the IEEE 39th VLSI Test Symposium (VTS), p 1–7
17. Condia JER, Sonza Reorda M (2023) Evaluating the impact of transition delay faults in gpus. In: Proceeding of the 36th International Conference on VLSI Design and 22nd International Conference on Embedded Systems (VLSID), p 353–358
18. De Caro D, Petra N, Strollo AGM (2009) High-performance special function unit for programmable 3-d graphics processors. IEEE Trans Circuits Syst I Regul Pap 56(9):1968–1978
19. Du B, Condia JER, Sonza Reorda M et al (2019) On the evaluation of seu effects in gpgpus. In: Proceeding of the 2019 IEEE Latin American Test Symposium (LATS), p 1–6
20. Ellaithy DM, El-Moursy MA, Zaki A et al (2019) Dual-channel multiplier for piecewise-polynomial function evaluation for low-power 3-d graphics. IEEE Trans Very Large Scale Integr (VLSI) Syst 27(4):790–798
21. Gonçalves M, Saquetti M, Azambuja JR (2018) Evaluating the reliability of a gpu pipeline to seu and the impacts of software-based and hardware-based fault tolerance techniques. Microelectronics Reliability / Proc of the 29th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF 2018) 88-90:931–935
22. Guerrero-Balaguera JD, Condia JER, Sonza Reorda M (2021) On the functional test of special function units in gpus. In: Proceeding of the 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), p 81–86
23. Guo H, Yang X (2007) A simple reliability block diagram method for safety integrity verification. Reliab Eng Syst Safety 92(9):1267–1273. Critical Infrastructures
24. Hamdioui S, Gizopoulos D, Guido G et al (2013) Reliability challenges of real-time systems in forthcoming technology nodes. In: Proceeding of the 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), p 129–134
25. Hennessy JL, Patterson DA (2012) Computer architecture: a quantitative approach - Fifth Edition, 4th. Chapter - Data-Level Parallelism in Vector, SIMD, and GPU Architectures
26. Hsiao SF, Li SY, Tsao KH (2015) Low-power and high-performance design of opengl es 2.0 graphics processing unit for mobile applications. In: Proceeding of the 2015 IEEE International Conference on Digital Signal Processing (DSP), p 110–114
27. Hsiao SF, Wen CS, Chen YH et al (2017) Hierarchical multipartite function evaluation. IEEE Trans Comput 66(1):89–99
28. IEEE (2022) The international roadmap for devices and systems: 2022. In: Institute of Electrical and Electronics Engineers (IEEE)
29. Jayashree Basu (1976) On binary multiplication using the quarter square algorithm. IEEE Trans Comput C–25(9):957–960
30. Kim YJ, Chung K, Kim LS et al (2009) Bank-partition and multi-fetch scheme for floating-point special function units in multi-core systems. In: Proceeding of the IEEE International Symposium on Circuits and Systems (ISCAS), p 1803–1806
31. Kim YJ, Kim HE, Kim SH et al (2012) Homogeneous stream processors with embedded special function units for high-utilization programmable shaders. IEEE Trans Very Large Scale Integr (VLSI) Syst 20(9):1691–1704
32. Leveugle R, Calvez A, Maistri P et al (2009) Statistical fault injection: Quantified error and confidence. In: Proceeding of the Design, Automation & Test in Europe Conference & Exhibition (DATE), p 502–506
33. Limas Sierra R, Guerrero-Balaguera JD, Condia JER et al (2023) Analyzing the impact of different real number formats on the structural reliability of tcus in gpus. In: Proceeding of the IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC), p 1–6
34. Limas Sierra R, Guerrero-Balaguera JD, Condia JER et al (2024) Exploring hardware fault impacts on different real number representations of the structural resilience of tcus in gpus. Electronics 13(3)
35. Martins M, Matos JM, Ribas RP et al (2015) Open cell library in 15nm freepdk technology. In: Proceeding of the 2015 Symposium on International Symposium on Physical Design (ISPD '15), p 171-178
36. Mukherjee S, Weaver C, Emer J et al (2003) A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In: Proceeding of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36), p 29–40
37. Osudin D, Child C, He YH (2019) Rendering non-euclidean space in real-time using spherical and hyperbolic trigonometry. Computational Science - ICCS 2019. Springer International Publishing, Cham, p 543–550
38. Papadimitriou G, Gizopoulos D (2021) Demystifying the system vulnerability stack: Transient fault effects across the layers. In: Proceeding of the ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pp 902–915
39. Pineiro JA, Oberman S, Muller JM et al (2005) High-speed function approximation using a minimax quadratic interpolator. IEEE Trans Comput 54(3):304–318
40. Qoutb AEG, El-Gunidy AM, Tolba MF et al (2014) High speed special function unit for graphics processing unit. In: Proceeding of the 9th International Design and Test Symposium (IDT), p 24–29

41. Rech P, Navaux P, Carro L (2013) Neutron sensitivity of integer and floating point operations executed in gpus. In: Proceeding of the 2013 14th Latin American Test Workshop (LATW), p 1–6
42. Santini T, Rech P, Nazar G et al (2014) Reducing embedded software radiation-induced failures through cache memories. In: Proceeding of the 19th IEEE European Test Symposium (ETS), p 1–6
43. Santos FFd, Condia JER, Carro L et al (2021) Revealing gpus vulnerabilities by combining register-transfer and software-level fault injection. In: Proceeding of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), p 292–304
44. Santos FFd, Navaux P, Carro L et al (2019) Impact of reduced precision in the reliability of deep neural networks for object detection. In: Proceeding of the 2019 IEEE European Test Symposium (ETS), p 1–6
45. Schulte M, Swartzlander E (1994) Hardware designs for exactly rounded elementary functions. IEEE Trans Comput 43(8):964–973
46. Strojwas AJ, Doong K, Ciplickas D (2019) Yield and reliability challenges at 7nm and below. In: Proceeding of the 2019 Electron Devices Technology and Manufacturing Conference (EDTM), p 179–181
47. Tselonis S, Gizopoulos D (2016) Gufi: A framework for gpus reliability assessment. In: Proceeding of the 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), p 90–100
48. Vallero A, Tselonis S, Gizopoulos D et al (2018) Multi-faceted microarchitecture level reliability characterization for nvidia and amd gpus. In: Proceeding of the IEEE 36th VLSI Test Symposium (VTS), p 1–6
49. Walther S (1971) A unified algorithm for elementary functions. In: Proceeding of the International Workshop on Managing Requirements Knowledge, p 379-385
50. Ziade H, Ayoubi RA, Velazco R (2004) A survey on fault injection techniques. Int Arab J Inf Technol 1(2):171–186



**Josie E. Rodriguez Condia** received the M.Sc. degree in electronics engineering from Universidad Pedagogica y Tecnologica de Colombia (UPTC), Colombia in 2017, and the Ph.D. degree in Computer Engineering from Politecnico di Torino, Turin, Italy in 2021. He is now an Assistant professor at the same institution. His research interests include functional testing, parallel architectures, Graphics Processing Units, and embedded system design.

**Juan-David Guerrero-Balaguera** is currently pursuing a PhD in the Department of Control and Computer Engineering at Politecnico di Torino, Torino, Italy. His research interests include functional tests, artificial intelligence, and parallel architectures. Guerrero-Balaguera has an MSc in electronics from the Universidad Pedagógica y Tecnológica de Colombia (UPTC), Sogamoso, Colombia.

**Edwar J. Patiño Nuñez** received a BSc in electronics from the Universidad Pedagógica y Tecnológica de Colombia (UPTC), Tunja, Colombia. His research interests include digital design, computer arithmetic circuit design, and testing.

**Robert Limas** is currently pursuing a PhD in the Department of Control and Computer Engineering at Politecnico di Torino, Torino, Italy. He received a MSc in electronics from the Universidad Pedagógica y Tecnológica de Colombia (UPTC), Sogamoso, Colombia. His research interests include artificial intelligence, High-Performance Computing and parallel architectures.

**Matteo Sonza Reorda** received the M.Sc. degree in electronics in 1986 and the Ph.D. degree in computer engineering in 1990, respectively, both from Politecnico di Torino, Italy. Currently he is a full professor in the Department of Control and Computer Engineering of the same Institution. His research interests include test of SoCs and fault tolerant electronic system design.