Analyzing hardware accelerators' reliability: from design exploration to fast evaluation with hyperscalers

Analyzing Hardware Accelerators' Reliability: From Design Exploration to Fast Evaluation with Hyperscalers

Juan-David Guerrero-Balaguera[†], Robert Limas Sierra[†], Gustavo Vilar de Farias[†], Sergiu-Mohamed Abed[*†], Ahmet Cagri Bagbaba[*], Josie E. Rodriguez Condia[†]
[†]Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy
[*]Cadence Design Systems, Munich, Germany
{*juan.guerrero, robert.limassierra, gustavo.vilarde, josie.rodriguez*}*@polito.it*; {*sergiu, abagbaba*}*@cadence.com*

***Abstract*—The rising density and complexity of domain-specific accelerators make conventional design space exploration strategies prohibitively expensive. These costs further escalate when non-functional properties, such as resilience, must be assessed for safety-critical applications, highlighting the need for innovative early-stage evaluation frameworks. This work investigates methods for early estimation of design and resilience features in large hardware accelerators for edge computing and machine learning (e.g., *vector processors*, *tensor cores*, and *stereo vision engines*). We first outline design space exploration challenges at the micro-architectural level for configurable vector processors. We then assess the use of open-source hyperscaler frameworks to analyze fine-grained functional and non-functional properties of large accelerators via simulation and emulation. Results show that such frameworks are feasible solutions and provide a viable path to reducing the computational costs of early characterization and evaluation.**

***Index Terms*—Design exploration, Domain-specific accelerator, Fault propagation, Hypercalers, Reliability**

I. Introduction

Currently, technology scaling in semiconductors enhances the development of modern digital designs by enabling the incorporation of a wide range of complex functionalities and specialized, sophisticated operations directly in silicon as System-on-Chip (SoC) devices, including Domain-Specific Accelerators (DSAs), hardware accelerators, and coprocessors [1]–[3]. In particular, the combination of aggressive development schedules, short times to market, and complex hardware functionalities demands the clever use of design, co-design, and development strategies that primarily focus on area, performance, and power budgets. In addition, the strict requirements of resiliency and reliability in safety-critical and functional safety domains exacerbate the need for effective strategies to ensure compliance with safety standards.

One promising approach involves proactively analyzing key system features and conducting early analyses of resilience and fault propagation during the design phase, enabling adjustments that improve robustness and resilience before production. Such a strategy is commonly adopted in software domains through techniques like *"shift-left"* analyses [4]. In particular, preventive analyses on a design aim to accelerate overall development through the early evaluation and analysis of crucial features (e.g., power, performance, functionality), which can guide updates and corrections during the design stages [5]. In [6], the authors employed early analyses to verify IPs during the design of SoCs using advanced analysis frameworks and data analytics, which resulted in considerable computing costs. However, properties of resilience and fault propagation remain partially unexplored. Authors in [7] combined functional and reliability properties to identify tradeoffs in the integration of an in-chip accelerator for GPUs based on micro-architectural versions of the cores. However, such core abstractions might not be available in early design stages.

This work examines the role of early analyses in large digital designs, with an emphasis on assessing design parameters and identifying optimal trade-offs among area, power, and performance, while additionally considering resilience and fault propagation as essential aspects of device integrity.

In particular, we evaluate the impact and benefits of early analyses of three main architectures for hardware accelerators: *i)* vector-based accelerators and coprocessors, *ii)* Dot Product Unit (DPU)-based hardware accelerators, and *iii)* a stream-based DSA for stereo vision. First, we focus on the main advantages and challenges in the early evaluation of functional and non-functional parameters at the low-level micro-architecture of vector processors, highlighting the primary benefits and constraints. Then, we address the challenges for characterizing and evaluating large hardware accelerators that combine data-intensive workloads, high transistor density, and complex operations. In this case, we overview software-based high-level solutions and evaluate the effectiveness of hyperscaler emulation-based platforms in speeding up evaluations while supporting design exploration and validation targets.

Our experimental results suggest that simulation-based analyses may be suitable for focused and small to medium-sized designs. Instead, the evaluation of non-functional parameters

This work was supported by the TIRAMISU project: EU Grant:101169378 HORIZON-MSCA-2023-DN-01, by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing (ICSC), and by the HardNet project through ICSRA - CINECA.

on large accelerators might require hyper-scaler emulation platforms.

II. Challenges in the reliability characterization of hardware accelerators

Transistor miniaturization enables scalable integration of complex functionalities in modern DSAs, supporting advanced applications across Edge Computing, *High-Performance Computing* (HPC), and AI. Moreover, DSAs are key components in several safety-critical applications, including autonomous robots, self-driving cars, healthcare, and aerospace, that are characterized by data-intensive workloads. Unfortunately, the increasing complexity in modern DSAs, mostly based on *Single/Multiple-Instruction Multiple-Data* (SIMD/MIMD) variations, homogeneous arrays of processing elements, or data-flow architectures, impedes the direct adoption of conventional strategies for the design, exploration, and evaluation of the resiliency features [8]. Most strategies and frameworks target the most relevant functional properties (*Performance*, *Power*, and *Area* or 'PPA') while partially neglecting resilience features, e.g., the impact of physical defects in hardware due to temporal or environmental variations and their propagation as errors. Furthermore, modern analyses of large designs require considerable computational power and time, as well as specialized methods and custom frameworks, to enable early reliability evaluations during design phases.

In fault characterization and propagation analyses of large DSAs, the computational and performance costs are related to the DSAs' size (e.g., transistor count), the evaluation benchmarks, and the targeted features to explore (e.g., operational features, configuration, and fault models).

The early modeling and prototyping strategies based on structural and functional descriptions in software are effective in supporting and evaluating conceptual operations and algorithmic behaviors of a targeted system, and can be used to characterize several features, including performance and overall size. However, they might neglect others, including power costs, the fine-grained descriptions in structures, and reliability [9], [10]. In contrast, the availability of micro-architectural descriptions of a DSA allows accurate evaluations. However, reliability characterizations at these levels usually increase computing costs exponentially [11]. In these cases, multi-thread software-based frameworks and emulation-based platforms are the preferred solutions. The emulation frameworks are feasible and can be used to evaluate functional and non-functional properties of large early designs. Unfortunately, the availability of such frameworks is limited, which, in combination with setting up customization for a targeted design, impedes their massive use. Instead, hybrid and multi-layer strategies combine multiple abstraction levels for evaluation [12], [13]. They enable focused analysis of sub-structures but often incur high costs for full-system characterization, as only target components are modeled architecturally while others remain at higher abstraction levels.

III. Characterization of Vector Processing Units

The *Vector Processing Units* (VPUs) are flexible and versatile SIMD-based accelerators commonly included as co-processors or independent engines in modern SoCs. In particular, VPUs reuse their *Processing Elements* 'PEs' (e.g., Adders/multipliers) for the efficient deployment of workloads while allowing flow management in code. Indeed, VPUs exploit hardware multi-threading to enable several cores and internal PEs in the deployment of general-purpose and AI-based applications, making them attractive architectures in DSA development.

This section analyses the positive and negative trade-offs in design space exploration of a fundamental tensor operator for AI (*Matrix Multiplication*) on two representative programmable VPUs: *Klessydra-T* and *Spatz*. In detail, *Klessydra-T* is a programmable coprocessor integrated on an interleaved multi-thread (IMT) RISC-V core for edge computing, with custom vector instructions [14]. The cores employ a custom scratchpad memory to handle inputs, intermediate operands, and results. Moreover, the number of cores and PEs is programmable according to workload needs or resource budget. Instead, *Spatz* is a configurable cluster interconnecting RISC-V cores with Vector Extension support (RVV) [15] [16]. Each core comprises a Vector Register File (VRF) and multi-precision Floating-point PEs, allowing the execution of multiple multiply-and-accumulate (MAC) operations per clock cycle. The cores and PEs are configurable.

The optimized architecture of both VPUs supports the analysis of functional parameters (PPA) on the tensor operator workload through software-based simulation strategies within an affordable timeframe.

For each VPU configuration and workload, we evaluated the PPA features after logic synthesis. From the experiments, we computed the workload performance on each configuration, adapting the matrix multiplication according to each VPU, so *Klessydra-T* used integer 16x16 matrices, while *Spatz* worked on 64×64 double-precision floating point matrices.

Table I report the power, performance (in clock cycles or 'CC') and area costs per VPU configuration on *Klessydra-T* and *Spatz*.

As observed in both VPUs, performance improvements, power consumption costs, and area overhead are proportional to the number of cores and PEs in a system. In detail, the results in *Klessydra-T* indicate that the workload size is crucial for identifying the optimal amount of PEs per core. Indeed, a small workload in a large VPU may lead to an increase in area without a substantial performance improvement. For example, as shown in table I, for multi-thread applications, the increase from 4 to 8 PEs/Core leads to a greater power consumption without any performance gain due to the fixed amount of time used for memory operations.

Regarding *Spatz*, looking at table I at the configurations with 2 core and 4 PEs and 4 cores with 2 PEs per core, we see that the former case achieves better performance at lower area and power consumption while having the same total number

TABLE I
KLESSYDRA-T AND SPATZ POWER, PERFORMANCE AND AREA (PPA) RESULTS

VPU	Cores	PEs/Core	Power (mW)	Performance (CC)	Area (μm)
Klessydra-T	**1**	**2**	1.420	3,613	4,909
		4	2.397	3,100	8,767
		8	4.319	3,095	16,366
	3	**2**	4.164	1,401	14,386
		4	7.118	1,230	25,892
		8	12.777	1,230	48,623
	6	**2**	8.388	615	28,734
		4	14.144	573	51,632
		8	25.673	573	97,217
Spatz	**2**	**2**	601.277	72,000	2.00×10^6
		4	948.490	39,400	2.86×10^6
	3	**2**	826.454	55,800	2.84×10^6
		4	1,312.390	31,200	4.09×10^6
	4	**2**	1,032.842	39,600	3.65×10^6
		4	1,568.021	23,200	5.34×10^6

of PEs, which indicates that the extra control logic coming from the latter case leads to less efficient usage of the PEs. Looking at the power and performance metrics when doubling the PE counts on the three core counts considered for Spatz, we see a performance improvement between 41% to 45% and a power increase of 52% to 59%.

According to our results, it seems that the ideal number of cores is limited by the application parallelism capability and hardware constraints, as observed in the performance results.

Clearly, the evaluation and design space exploration of several VPU configurations and workloads is affordable when considering limited data sizes. Unfortunately, the computational costs increase exponentially as the dimensions of the matrices to operate on grows. The size of the VPUs also plays a crucial role in the feasibility of this type of analysis. For example, *Klessydra-T*'s bigger configuration has about 200 thousand gates, while *Spatz* with four cores and four PEs per core has a total number of gates of approximately 3.03 million, which hinders the study of fault tolerance, exacerbating the need for clever methods for such analysis.

IV. ACCELERATING FAULT CHARACTERIZATION THROUGH HYPERSCALE COMPUTING

The continuous growth in architectural complexity of DSAs, particularly those employed in multimedia, image processing, and AI, renders the assessment and characterization of properties, such as reliability and fault tolerance, increasingly intractable due to the prohibitive computational power involved.

While software-based **Fault Simulation** (FSim) remains a powerful technique for analyzing hardware fault effects, the escalating complexity of modern DSAs implies prohibitively long execution times (in some cases spanning years), thereby underscoring the need for efficient acceleration strategies. In this regard, **Fault Emulation** strategies have been developed for speeding up the fault evaluation and reliability assessment of computational hardware using FPGA devices. Although fault emulation alleviates the time required for a fault injection campaign, its implementation still necessitates significant effort to set up a proper evaluation environment. Thus, it is crucial to develop adequate tools that facilitate the deployment of a fault injection campaign using emulation strategies. Furthermore, *Hyperscale Computing* systems provide the necessary computational resources for accelerating complex and time-consuming workloads (e.g., fault injections).

TABLE II
TIME PER FAULT COMPARISON BETWEEN SHADOWFI AND A COMMERCIAL LOGIC SIMULATOR.

Target core	SHADOWFI		Comm. Simulator	FI Campaign	
	FPGA	Verilator		FPGA	Com-Simulator
TCU	1.7 (s)	2.8 (s)	49 (s)	11.3 (h)	326.7 (h)
SVC	1.8 (s)	14.3 (s)	600 (s)	12.0 (h)	4,000.0 (h)

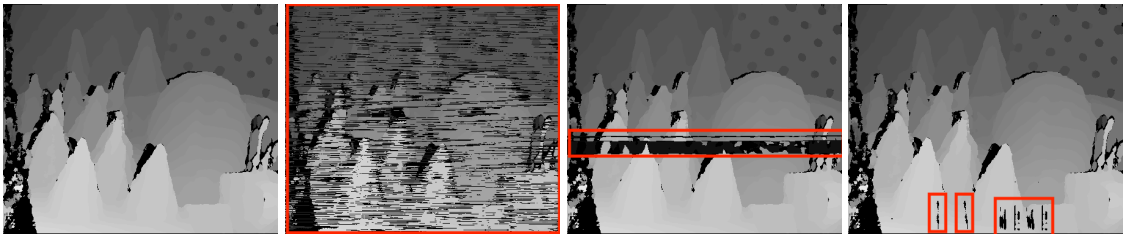
In particular, several initiatives from industry and academia have developed different FPGA-based hyperscale computing systems, bringing the flexibility of reconfigurable computing at scale. Such computational infrastructures are the perfect match for implementing and accelerating fault emulation frameworks.

In this regard, **SHADOWFI** has emerged as an open-source fault emulation framework that leverages the power of hyperscale computing, providing outstanding FI acceleration when targeting complex DSAs. Specifically, the framework offers two fault injection workflows. The first comprises a simulation-based approach powered by HPC infrastructures, while the second one utilizes fault emulation using FPGA cluster infrastructures. Both fault injection strategies consist of netlist-based fault instrumentation, which inserts saboteur circuits that can model both permanent and transient faults. **SHADOWFI** is publicly available at https://github.com/divadnauj-GB/shadowfi.git.

This section exploits the advantages of **SHADOWFI** to assess the impact of permanent and transient faults on two DSAs. The first one corresponds to a *Stereo Vision Core* (SVC) that computes the disparity map in stereo vision applications (e.g., autonomous navigation systems). The SVC accelerator implements a deep pipeline streaming architecture for the real-time execution of stereo images. The second DSA corresponds to a pipeline *Tensor Core Unit* (TCU) used in GPUs and processors for accelerating Matrix Multiplications (MMs) in AI.

For the experiments, **SHADOWFI** performed fault injection campaigns on both DSAs using a statistical sampling strategy [17], with a confidence level of 95% and 2% of error margin, accounting for 24K faults on each hardware target. Table II reports the execution times per fault for both experiments when using software-based **SHADOWFI** simulation (in Verilator), **SHADOWFI** emulation (with FPGAs), and a commercial logic simulator. The results show that **SHADOWFI** emulation is around 600X faster than a commercial logic simulator for a single fault. Likewise, software-based **SHADOWFI** simulation is 40X faster w.r.t. the commercial logic simulator, especially for the SVC.

Every fault affecting the hardware can have a particular level of severity on the system's functionality. Hence, it is crucial to define adequate metrics that enable the classification of the faults according to their effects on the functionality of a particular DSA. In particular, we classified the faults into three main groups: *Masked*, *Silent Data Corruption (SDC)*, and *Detected Unrecoverable Error (DUE)*. Further details about



Fault-free $PSNR < 12dB$ $12dB < PSNR < 24dB$ $PSNR > 24dB$

Fig. 1. Exemplification fo different fault effects categorized according the PSNR impact on the final disparity map for the SVC.

these classifications can be found in [18]. In addition, the SDCs can be further classified into two or more categories, e.g., some faults induce negligible or almost imperceptible errors at the system's output, which do not pose a system threat (SDC-Safe); whereas other faults seriously jeopardize the system and can potentially lead to failure (SDC-Critical). This classification is application dependent, requiring the definition of a threshold or behaviors that allow for the discrimination of whether a fault falls into one or several SDC classes.

This work defines metrics that allow for differentiating SDC faults based on the effect they produce at the output of the application. In particular, we adopted the *Peak-to-Noise-Ratio* (PSNR) and the relative L_∞ norm to measure the fault impact on the SVC and the TCU accelerators, respectively. The PSNR enables the precise measurement of the impact of faults on the SVC accelerator by considering the deviations between the golden and faulty disparity maps (i.e., lower PSNR values mean higher severity due to the faults). Figure 1 depicts three visual examples of fault effects along their PSNR results. When faults result in a PSNR below 24dB, their visual effects exhibit significant noise that can completely or partially distort the disparity map. Such faults should be considered as (*SDC-critical*) since their effects can cause a complete system failure due to the impossibility of properly distinguishing the distance of objects in the scene. Instead, faults with a PSNR above 24dB have almost negligible impact on the output disparity map; these faults are not a serious threat to the system, since their effects are mostly attenuated by any image denoise filtering and can be classified as (*SDC-safe)*.

In contrast, the impact of faults on the TCU are quantified using the relative L_∞ norm, defined as $\frac{\|F-G\|_\infty}{\|F\|_\infty}$, where F and G corresponds to the faulty and golden output matrices from the TCU. This metric is typically employed to measure the error between approximate and exact matrices [19], making it suitable for the fault impact evaluation in AI DSAs. In this work, we define *SDC-safe* faults when the induced relative L_∞ norm is lower than 1%; otherwise, faults are labeled as *SDC-critical*.

Figure 2 presents the distribution of faults based on their impact for each DSA. In the case of the SVC, 21% of faults had no impact on the DSA's functionality, while 79% resulted in errors in the DSA's outputs (SDCs). Notably, only 10.7% of faults are classified as SDC-safe due to their minimal effect on the generated disparity map, whereas 68.2% of faults pose a significant threat to the application's performance. In contrast, the results indicate that 28% of faults do not affect the TCU's

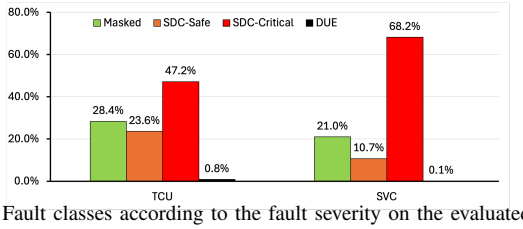


Fig. 2. Fault classes according to the fault severity on the evaluated DSAs.

operation, and 23.6% have a relative L_∞ norm error of less than 1%. However, approximately 47.2% of faults lead to errors in the TCU results, which could seriously jeopardize the application. These findings would not have been possible without the use of **SHADOWFI**, reducing a FI campaign of 24,000 faults to less than 12 hours, while the usage of conventional simulators would have taken 327 hours for the TCU and 4,000 hours for the SVC. Hence, the results enable us to identify hardware locations within the DSAs that are more vulnerable, providing guidelines for the development of effective hardening solutions and countermeasures to mitigate the effects of corruption.

V. Conclusions

This work overviews two strategies for the design, exploration, and evaluation of Domain-Specific hardware accelerators. First, we review the use of low-level microarchitecture abstractions for the exploration and evaluation of functional features of VPUs. Then, we used hypercaler-based emulation framework to speed up the evaluation of complex accelerators.

The results highlight the importance of exploring the VPU parameters for optimal PPA utilization in accordance with workload requirements. However, due to growing area and complexity in hardware, the fault-sensitivity study is not always feasible together with power, performance, and area analysis. Hence, the use of a framework such as **SHADOWFI** has been proven essential to accelerate fault simulation.

In conclusion, while PPA analysis can still be performed in complex architectures, the study of fault tolerance needs more efficient strategies, such as emulation frameworks (**SHADOWFI**), to allow reliability studies preserving accuracy under affordable times. In the future, we plan to apply **SHADOWFI** in transistor-dense accelerators and SoCs.

References

[1] Y.-J. Mii, "Semiconductor industry outlook and new technology frontiers," in *IEEE Int. Electron Devices Meeting (IEDM)*, 2024, pp. 1–6.

[2] K. Zhang, "1.1 semiconductor industry: Present future," in *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 67, 2024, pp. 10–15.

[3] B. Dally, "Hardware for deep learning," in *2023 IEEE Hot Chips 35 Symposium (HCS)*. IEEE Computer Society, 2023, pp. 1–58.

[4] S. A. Vaddadi *et al.*, "Shift left testing paradigm process implementation for quality of software based on fuzzy," *Soft Computing*, pp. 1–13, 2023.

[5] X. Zhuang *et al.*, "Enhancing product robustness in reliability-based design optimization," *Reliability Engineering System Safety*, vol. 138, pp. 145–153, 2015.

[6] H. J. Chakraborty and D. K. Acharya, "A shift-left approach in qualification of digital ips for socs by applying advanced automation and data analytics," in *Int. Symp. on VLSI Design and Test (VDAT)*, 2024, pp. 1–5.

[7] J. E. Rodriguez Condia *et al.*, "Investigating and reducing the architectural impact of transient faults in special function units for gpus," *Journal of Electronic Testing*, vol. 40, no. 2, pp. 215–228, 2024.

[8] J. Leng *et al.*, "Asymmetric resilience: Exploiting task-level idempotency for transient error recovery in accelerator-based systems," in *Int. Symp. on High Performance Computer Architecture (HPCA)*, 2020, pp. 44–57.

[9] D. Gnad *et al.*, "Reliability and security of ai hardware," in *European Test Symp. (ETS)*, 2024, pp. 1–10.

[10] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 902–915.

[11] J. E. R. Condia *et al.*, "A multi-level approach to evaluate the impact of gpu permanent faults on cnn's reliability," in *Int. Test Conf. (ITC)*, 2022, pp. 278–287.

[12] F. F. d. Santos *et al.*, "Revealing gpus vulnerabilities by combining register-transfer and software-level fault injection," in *51st Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*, 2021, pp. 292–304.

[13] M. H. Ahmadilivani *et al.*, "Special session: Reliability assessment recipes for dnn accelerators," in *2024 IEEE 42nd VLSI Test Symposium (VTS)*, 2024, pp. 1–11.

[14] A. Cheikh *et al.*, "Klessydra-t: Designing vector coprocessors for multithreaded edge-computing cores," *IEEE Micro*, vol. 41, no. 2, 2021.

[15] M. Cavalcante *et al.*, "Spatz: Clustering compact risc-v-based vector units to maximize computing efficiency," Sep. 2023.

[16] "Risc-v vector extension 1.0." https://github.com/riscvarchive/riscv-v-spec, 2022.

[17] R. Leveugle *et al.*, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 502–506.

[18] J.-D. Guerrero-Balaguera *et al.*, "Understanding the effects of permanent faults in gpu's parallelism management and control units," in *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–15.

[19] B. Tran and V. Vu, "Fast exact recovery of noisy matrix from few entries: the infinity norm approach," *arXiv preprint arXiv:2501.19224*, 2025.