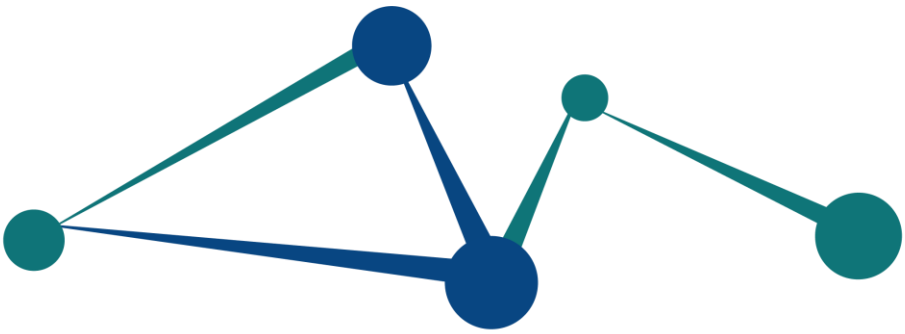


NEUROPULS

Deliverable 5.9

System Architecture Model and Simulation Platform: Iteration 1

Start date of the project: 1st January 2023

Duration 48 months

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

+	Document Classification
Document Title	D5.9 System Architecture Model and Simulation Platform: Iteration 1
Author(s)	P09 – NKUA – George Papadimitriou
Work Package	WP5 – System Architecture Modelling and Simulation
Dissemination Level	PU = Public
Nature	R = Report
Doc ID Code	24_03_29_NEUROPULS_D5.9
Keywords	Simulation, gem5, accelerators modeling, memory hierarchy, security interface

		Document History
2024-01-22	Table of contents, structure defined, abstract and introduction	P09 NKUA – George Papadimitriou
2024-01-30	Section 5, SW-related security aspects	P12 UNIVR – Alberto Lovato, Niccolò Marastoni, Mariano Ceccato
2024-02-12	Introduction, Background (gem5, accelerators), and Section 3	P09 NKUA – George Papadimitriou, Vasileios Karakostas, Dimitris Gizopoulos

Document History

2024-02-26	Section 5, HW-related security aspects	P04 POLITO – Roberta Bardini, Tzamn Carmona, Alessandro Savino
2024-02-28	Section 2.3, and Section 4.1, 4.2, and 4.4	P06 BSC CNS – Mikel Fernandez
2024-03-19	Section 5, HW-related security modeling	P04 POLITO – Roberta Bardini, Tzamn Carmona, Alessandro Savino
2024-03-19	Section 4.3	P01 CNRS – Fabio Pavanelo

Document Validation

Project Coordinator	P1 CNRS – Fabio Pavanello Fabio.pavanello@cnrs.fr
Date	2024-03-29

Document Abstract

This report presents the development of a robust and efficient system architecture modeling and simulation infrastructure, aiming to facilitate the comprehensive evaluation of complete computing systems integrating neuromorphic accelerators and hardware security primitives of the NEUROPULS Horizon Europe project (Grant Agreement n° 101070238) along with electronic CPUs. The simulation infrastructure aims to enable detailed system-level evaluation, encompassing both software and hardware, to analyze the security properties of the computing platform incorporating photonic hardware primitives, and PCM (Phase-Change Memory) memory models (i.e., a type of non-volatile random-access memory).

The first part of this report details the diverse objectives, including the creation of full system simulation tools, exploration of the design space for heterogeneous computing systems featuring photonic neuromorphic accelerators and hardware primitives next to the electronics processing elements, support for benchmarking activities in Use Cases, and the assessment of critical design parameters such as performance, power, resilience, and energy efficiency. The second part shows how the infrastructure enables detailed system-level evaluation, encompassing both software and hardware, to analyze the security properties of the computing platform incorporating photonic hardware primitives.

This deliverable outlines the methodology, tools, and considerations involved in the establishment of the system architecture modeling and simulation infrastructure, highlighting its significance in advancing the understanding and optimization of computing systems.

Table of contents

1. Introduction

In the ever-evolving landscape of computing systems, the integration of neuromorphic accelerators (or other domain-specific accelerators) and hardware security primitives represents a critical frontier. This deliverable introduces the first iteration of a comprehensive system architecture modeling and simulation infrastructure that will be developed in the context of the NEUROPULS project to meet the evolving demands of this paradigm and offer a framework for extensive design space exploration.

The objectives of this endeavor are fourfold: (a) to create efficient full system simulation tools on top and around of the gem5 simulator [2] for modeling and evaluating complete computing systems with neuromorphic accelerators and security primitives, (b) to explore the diverse design space of heterogeneous computing systems employing photonic neuromorphic accelerators and hardware primitives, as developed and characterized in the physical design related Work Packages 2, 3, and 4, (c) to provide support for benchmarking activities within Work Package 6, focusing on Use Cases and benchmarks while assessing essential design parameters at the system scale, including performance, power efficiency and reliability, and (d) to facilitate detailed system-level evaluation of both software and hardware, with a specific emphasis on the security properties of the computing platform leveraging photonic hardware primitives.

The interdisciplinary nature of the NEUROPULS project, and specifically of the Work Package 5 necessitates collaboration between experts in computer architecture, hardware design, neuromorphic computing, photonics, reliability, and hardware security. The simulation infrastructure, which will be built mainly in the context of the WP5, serves as a common ground where researchers from diverse backgrounds can converge, fostering collaboration and knowledge exchange. This cross-disciplinary synergy is essential for holistic advancements, ensuring that innovations in one domain consider the implications and requirements of others.

The creation of such a NEUROPULS simulation infrastructure will not only meet the specific goals outlined above but will also position itself as a crucial enabler for future technological advancements. The capability to simulate complete computing systems, with a focus on neuromorphic accelerators and security primitives, not only aims at the understanding of existing architectures but also serves as a testing ground for novel future ideas and approaches, including performance, power, and reliability evaluations and enhancements. Therefore, creating an advanced simulation system is crucial for steering the direction of research in computing systems.

1.1 Objectives

The primary objectives of this work package (WP5), and thus, of this deliverable, are shown below:

a) **Efficient Full System Simulation Tools Infrastructure:** Our primary goal is to engineer a state-of-the-art infrastructure capable of efficiently simulating complete computing systems. This infrastructure will not only incorporate conventional computing elements (e.g., contemporary CPUs with cache memories, SRAM storage elements, such as register files, TLBs – Translation Lookaside Buffers, etc.), but will also integrate cutting-edge neuromorphic accelerators modeling (e.g., including storage elements, such as ScratchPad memories, register banks, etc.), and hardware security primitives. By doing so, we aim to create a versatile simulation environment that facilitates detailed evaluation and optimization of these innovative components.

b) **Exploration of Heterogeneous Computing Systems:** The advent of photonic neuromorphic accelerators and hardware primitives has opened new frontiers in computing design. We aim to explore the broader design space of heterogeneous computing systems, leveraging the advancements made in Work Packages 2 through 4 and also advancements currently available in the literature. By integrating photonic neuromorphic accelerators, PCMs (Phase-Change Memories) [1], and hardware primitives, we aim to push the boundaries of computing capabilities and assess their potential within a holistic system architecture.

c) **Support for Benchmarking Activities and Assessment of Design Parameters:** Aligned with the objectives of Work Package 6 (Use Cases and Benchmarks), we intend to provide essential support for benchmarking activities. The focus will be on evaluating crucial design parameters at the system scale, with a dual emphasis on performance and power, thereby addressing the pivotal issue of energy efficiency. Through rigorous benchmarking, we aim to establish a comprehensive understanding of how these systems perform under varying workloads.

d) **Facilitation of Detailed System Level Evaluation of Security Properties:** Recognizing the paramount importance of security in modern computing, our infrastructure will enable in-depth evaluations of the security properties of computing platforms. Specifically, the emphasis will be on platforms employing photonic hardware primitives. By conducting thorough evaluations at both software and hardware levels, we aim to contribute valuable insights to the broader discourse on securing advanced computing systems.

1.2 Deliverable Organization

The subsequent sections of this deliverable are organized in a way that provides a comprehensive understanding of the infrastructure development of Work Package 5 (WP5). Each section is designed to unfold a specific aspect of this Work Package, outlining methodologies, approaches, tools, and frameworks used to achieve the previously outlined objectives. More specifically, in Section 2 we provide a comprehensive background for every aspect that WP5 covers, such as domain-specific accelerators, the gem5 simulator (the main simulation tool for the entire WP5), the PCM

memory models, and the PUF models. In Section 3 we delve into the technical challenges of infrastructure development, outlining the methodologies, tools, and frameworks employed to realize the ambitious objectives set forth. Section 4 presents the details of the PCM memory models in the simulation infrastructure, while Section 5 covers the hardware security primitives and the software-related security protocols. Through this deliverable, we aim to contribute not only to the WP5's specific goals but also to the broader discussion on the future of computing architectures.

2. Background & Related Work

2.1 Domain-Specific Accelerators

With Moore's Law slowing down [25], domain-specific accelerators (DSAs) have become increasingly important due to their superior performance and efficiency on specific tasks compared to general-purpose CPUs [26]. Accelerators are specialized hardware engines designed for specific domains, such as graphics [27], deep learning [28], bioinformatics [29], image processing [30], and simulation [31]. They deliver high-performance gains by reducing overheads, offering fast specialized operations, optimized memory systems, and parallelism. General-purpose CPUs perform better at control-intensive tasks but are less efficient than DSAs for specific tasks [32]. As modern computing systems become more complex and heterogeneous, multiple CPUs of different ISAs, and a diverse set of DSAs need to cooperate for optimized performance and energy efficiency [33].

In the realm of electronic accelerators, these solutions are commonly realized through the utilization of customized integrated circuits (ICs) or Field-Programmable Gate Arrays (FPGAs). Their design is geared toward executing computations with optimal efficiency and throughput for targeted applications, complementing general-purpose CPUs in heterogeneous system setups.

However, photonic accelerators exploit photonics principles to expedite information processing. These accelerators utilize photons, as opposed to electrons, for data transmission and processing, presenting advantages in terms of bandwidth, energy efficiency, and diminished heat generation.

Some representative types of accelerators are the following:

1. **Google's Tensor Processing Units (TPUs)**:

   Google's TPUs [34] are domain-specific accelerators designed specifically for accelerating machine learning workloads. They are optimized for matrix multiplication, a common operation in neural network computations. TPUs are used internally in Google's data centers to accelerate AI and ML tasks.

2. **NVIDIA GPUs for Graphics and AI**:

   NVIDIA's Graphics Processing Units (GPUs) [35] are widely used as domain-specific accelerators for graphics rendering. Additionally, NVIDIA GPUs have become popular for accelerating AI (Artificial Intelligence) and ML (Machine Learning) workloads. The parallel processing capabilities of GPUs make them suitable for tasks like deep learning training and inference.

3. **Intel FPGAs for Custom Acceleration**:

   Intel's Field-Programmable Gate Arrays (FPGAs) [36] are reconfigurable devices that allow users to create custom accelerators tailored to specific applications. FPGAs can

be programmed to accelerate a wide range of workloads, from signal processing to encryption, providing flexibility in hardware acceleration.

4. **Optical Co-Processors**:

   In the realm of photonic accelerators, there is ongoing research on the development of optical coprocessors for specific tasks [37], [38]. These coprocessors leverage the properties of light to perform specialized computations, aiming to enhance speed and energy efficiency in information processing.

The design of both electronic and photonic domain-specific accelerators continues to be an active area of research and development as the demand for specialized and efficient computing solutions grows across various industries. NEUROPULS contributes to this domain by developing novel photonic computing architectures and security layers based on photonic PUFs in augmented silicon photonics CMOS-compatible platforms.

2.2 The gem5 Simulator

WP5 aims to build a comprehensive system architecture modeling and simulation infrastructure that incorporates neuromorphic accelerators and hardware primitives, as shown in Figure 1. To achieve this goal, the state-of-the-art computing systems simulator, gem5, will serve as the backbone for the entire WP5. gem5 [13] is a widely used system-level computer architecture simulator that provides a flexible and modular framework for modeling and simulating various aspects of computer systems. The primary goal of gem5 is to enable researchers and developers to explore and evaluate new architectural ideas, system designs, and software optimizations in a simulated environment before implementing them on real hardware. It supports cycle-level simulation of a wide range of computer architectures, including x86, Arm, MIPS, RISC-V, and other older ones, making it a versatile platform for researchers working on diverse computer architectures at the system level (including the microarchitecture, architecture, operating systems, and application layers of the computing stack). gem5 is open-source and has gained popularity in both academic and industrial research communities.

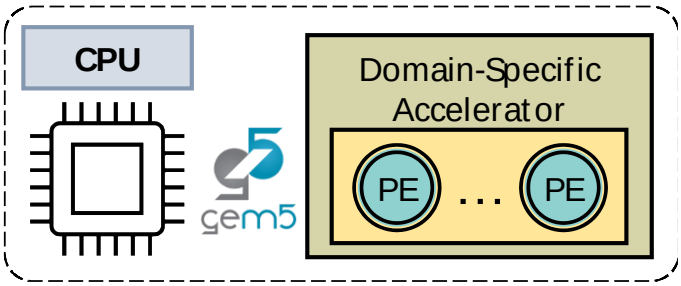


Figure 1: gem5-based system architecture modeling and simulation infrastructure overview.

gem5's design is based on a modular and extensible architecture, allowing users to easily customize and extend the simulator to suit their specific needs. It includes models for various components such as processors, memory hierarchy, caches, interconnects, and peripheral devices, providing a comprehensive simulation environment for studying computer system behavior.

gem5 originated as a successor to the M5 simulator [14]. The transition from M5 to gem5 occurred to address limitations in M5 and to create a more modular and extensible simulation framework. The design philosophy of gem5 centers around modularity and extensibility. It employs a component-based architecture, allowing users to easily combine and modify simulation components to model different aspects of computer systems accurately.

The simulator includes detailed modeling of memory hierarchies, including caches, main memory, and storage, and support for modeling on-chip and off-chip interconnects. Researchers can analyze the impact of different memory configurations and the effects of different communication architectures on system performance.

Finally, gem5 is widely adopted in both academic research and industrial settings. It is used for a range of studies, including microarchitecture exploration, performance analysis, software development, and validation of new architectural ideas before actual hardware implementation.

Some related works that complement or are often used with gem5 in the context of computer architecture research are shown below:

1. **SPEC and MiBench Benchmarks**:

   The benchmark suites offered by the Standard Performance Evaluation Corporation (SPEC), including SPEC CPU 2017 [15], or other popular suites such as MiBench [48] are extensively utilized for assessing computer system performance. gem5 is commonly utilized by researchers to simulate the execution of SPEC benchmarks on CPU(s), enabling the evaluation of the effects of architectural modifications on practical workloads [16].

2. **DRAMSim2**:

   DRAMSim2 [17] is a memory system simulator that focuses specifically on modeling dynamic random-access memory (DRAM) behavior. Researchers integrate DRAMSim2 with gem5 [18] to study memory subsystem performance, investigate memory hierarchy designs, and analyze the impact of different DRAM architectures on overall system performance.

3. **Sniper**:

   Sniper [19] is another cycle-level simulator that complements gem5 in the realm of computer architecture research. It provides detailed simulation capabilities for multicore processors and supports various performance analysis tools. Researchers frequently utilize gem5 for system-level simulations and turn to Sniper for faster microarchitectural studies, especially given the absence of system-level support in Sniper [20].

4. **Pin**:

   Pin, developed by Intel [21], is a dynamic binary instrumentation tool. Researchers employ Pin for the instrumentation and analysis of binary programs, subsequently utilizing gem5 for simulations at the system level [22]. This pairing enables a thorough evaluation of behaviors at both the application and system levels.

5. **gem5-gpu**:

   gem5-gpu [23] is a simulator designed to emulate tightly integrated CPU-GPU systems. It builds upon gem5, a modular full-system CPU simulator, and GPGPU-Sim, a detailed GPGPU simulator [24]. This approach enables the simulation of diverse system configurations, including those with coherent caches and a unified virtual address space between the CPU and GPU, as well as systems maintaining separate physical address spaces for the GPU and CPU. Notably, gem5-gpu supports the execution of the most unmodified CUDA 3.2 source code, allowing applications to launch non-blocking kernels for simultaneous CPU and GPU processing.

These related works and tools provide additional perspectives and capabilities to researchers using gem5, allowing them to conduct comprehensive studies across various levels of abstraction and dimensions of computer architecture. The combination of gem5 with these tools enhances the versatility and depth of architectural exploration in both academia and industry. Through the NEUROPULS project, our objective is to establish a novel System-on-Chip (SoC) infrastructure built upon the gem5 simulator. Leveraging its simulation effectiveness, we aim to deliver a new simulation infrastructure that supports not only CPU modeling but also accelerator designs within an SoC framework integrated with gem5.

2.3 PCM Memory Models

Conventional memory technologies like SRAM and DRAM fall short in meeting the escalating memory demands due to limitations in technology scaling, including low density, high standby power, and inadequate reliability. Phase-Change Memory (PCM) emerges as one of the most promising non-volatile memories (NVM) solutions. PCM cells utilize chalcogenide alloys, such as Ge2–Sb2–Te5 (GST), to store information by modulating the resistance transition between amorphous (high-resistance) and crystalline (low-resistance) states of the material.

The process involves a SET operation (representing 1) where the phase change material undergoes crystallization induced by heating through an electrical pulse or laser irradiation, resulting in a transition to the crystalline state when the temperature exceeds its crystallization threshold. Conversely, a RESET operation (coding 0) entails applying a different electrical pulse followed by a quick cessation, or a brief exposure to lower laser heat, causing the cell to revert to the amorphous state.

Adopting NVMs as a replacement for the current DRAM-based main memory architecture can yield substantial benefits, including reduction in total energy

consumption while maintaining the same capacity. However, with the continuous reduction in feature size, supply voltage, and increased on-chip density, computer systems are expected to become more susceptible to both hard errors and transient errors.

Replacing SRAM-based on-chip caches with Non-Volatile Memories (NVMs) holds the potential to significantly enhance system performance owing to their larger capacity and reduce power consumption due to their zero-standby leakage feature.

State-of-the-art NVM simulation methods must account for NVM-specific characteristics, allowing user applications to tailor them for customized interfaces. However, such customization imposes higher demands on developers. Architectural-level NVM simulators (e.g., NVSIM, NVMain, CACTI) focus on hardware details such as memory cells and transistor size in physical designs.

2.4 Physical Unclonable Function (PUF) Models

Physical Unclonable Functions (PUFs) stand at the forefront of hardware-based security, providing a foundational layer characterized by ease of manufacture and inherent resistance to replication. This unique attribute positions PUFs as ideal candidates for generating and safeguarding cryptographic keys, particularly in device authentication. Operating through a Challenge-Response Pair (CRP) mechanism, PUFs introduce a robust process involving two crucial stages: enrollment and verification.

In the enrollment stage, the distinctive physical characteristics of a device undergo capture and transformation into a digital representation. This intricate process exposes the device to challenges manifested as specific input signals or stimuli. The device responds to these challenges by leveraging its intrinsic physical properties, generating responses that collectively form the CRP.

The captured responses are a digital fingerprint, giving each device a unique identity. This identity arises from inherent variations and imperfections within the physical components, rendering accurate replication nearly impossible. The primary objective of the enrollment stage is to establish a reliable and distinct signature for each device, ensuring that no two devices share identical CRPs.

The verification stage employs the stored CRP to confirm the device's identity in subsequent interactions. The device responds when presented with a new challenge, and the generated output is compared to the stored CRP. A successful match between the response and the digital fingerprint affirms the device's identity, granting authentication. This verification process is a crucial security measure, permitting access or authentication only to legitimate devices with the correct physical characteristics. Any attempts to clone the device or employ fraudulent responses lead to mismatches during verification, thereby fortifying the system's overall security.

Beyond the enrollment and verification stages, PUFs exhibit diverse classifications based on material fabrication and security attributes. Material categorizations include distinctions such as silicon and non-silicon, electronic and non-electronic. Security classifications delineate PUFs into categories such as Weak and Strong PUFs. The NEUROPULS project specifically focuses on photonic PUFs, which rely on the random splitting of a laser beam interacting with multiple resonant devices.

The NEUROPULS project aims to leverage the same technology used for the accelerator and the photonic PUF. In particular, the photonic PUF will be coupled with the accelerator to provide a signature also of the accelerator. The utilization of PUFs created from silicon photonics has demonstrated efficacy, as evidenced by diverse studies and author contributions [53] [55].

The application of photonic PUFs, especially within silicon photonics, showcases promising results and aligns with ongoing research endeavors, such as the innovative NEUROPULS project. Two different PUFs are available in the literature, and the following subparagraph will describe them to better grasp their peculiarities.

2.4.1 Weak PUFs

A Weak PUF is characterized by a limited number of challenges, often as few as one fixed challenge. In contrast to Strong PUFs, which prioritize intricate challenge-response behavior, Weak PUFs utilize their limited set of challenges to derive a classical binary secret key. While Weak PUFs may be advantageous regarding invasiveness and individualization during production, their susceptibility to side-channel attacks, such as power consumption or emanation analysis, poses a challenge. Unlike Strong PUFs, they lack complexity and resistance to numerical prediction, relying on error correction for their limited challenges. Examples include the SRAM PUF, Butterfly PUF, and Coating PUF, each tailored for specific cryptographic applications focusing on simplicity in key derivation. In essence, weak PUFs serve as a specialized form of secret key storage suitable for various cryptographic schemes but necessitates considering their susceptibility to certain attacks.

2.4.2 Strong PUFs

Strong PUFs are designed to provide heightened security in the face of potential adversarial threats. PUFs, which generate unique digital fingerprints through CRPs, serve as identifiers for electronic devices. Strong PUFs set themselves apart by exhibiting a complex and intricate relationship between challenges and responses, creating a formidable barrier for adversaries attempting to predict or imitate their behavior. The requirement for a vast and practically unmanageable number of possible challenges adds a layer of security, making it computationally infeasible for attackers to determine all challenge-response pairs within a limited timeframe. Strong PUFs resist cloning by ensuring that responses are extremely difficult to replicate, leveraging manufacturing

variations and structural disorders beyond the control of the original manufacturer. Notably, the challenge-response behavior of Strong PUFs, exemplified by the Optical PUF using random optical reflection patterns, finds application in key establishment and identification protocols. The inherent complexity of Strong PUFs makes numerical prediction challenging for adversaries, enhancing the unpredictability and overall security of these specialized devices. In summary, Strong PUFs play a crucial role in fortifying the security of electronic devices by incorporating advanced features that resist adversarial attempts to compromise their identification mechanisms.

3. Simulation Infrastructure for Neuromorphic Accelerators Models

This section delves into the heart of the simulation infrastructure — the system architecture simulation framework, which is called gem5-MARVEL [33]. It outlines the underlying principles, methodologies, and innovations embedded in the gem5-MARVEL framework. Special emphasis is placed on its ability to accommodate neuromorphic accelerators and design space exploration using the NEUROPULS simulation infrastructure, ensuring a holistic representation and exploration of modern computing systems.

3.1 gem5-MARVEL

3.1.1 gem5-based Accelerator Modeling

Recently, several efforts have been made to integrate domain-specific accelerator (DSA) models with the state-of-the-art gem5 simulation environment. These efforts include, among others, gem5-Aladdin [39] and PARADE [40]. Additionally, SystemC support was added to gem5 [41] enabling the potential of cycle-accurate modeling of hardware structures including accelerator datapaths. Therefore, existing works for modeling domain-specific accelerators rely either on pre-RTL [39], [40] or RTL-based solutions [41] (e.g., C/C++-based models).

However, all these options have significant disadvantages. On one hand, both gem5-Aladdin and PARADE, although they are both pre-RTL frameworks and thus, comprehensive, they provide limited support for design space exploration due to their restrictive simulation semantics. In addition, they suffer from low simulation fidelity when data availability, parallelism, and timing are not decoupled from the input dataset. On the other hand, while SystemC support offers the potential of highly accurate modeling, being an RTL-based alternative, it requires considerably higher design effort and eventually provides lower throughput than the other two pre-RTL frameworks. Although low-level simulators may provide accurate fault effects, their simulation throughput is extremely low to be affordable and cannot model long-running workloads with OS activity (RTL simulation is several orders of magnitude slower than cycle-level microarchitectural simulation [42]-[44]. To this end, gem5-MARVEL relies on microarchitecture-level simulation using the latest version of the gem5 simulator [13], [45], which allows (i) deterministic, (ii) end-to-end, (iii) cycle-level execution of (iv) large workloads (v) on top of an operating system; this combination is impossible at lower levels. To this end, we are based on a new pre-RTL framework, which overcomes these limitations and provides flexibility, and excellent tradeoffs among performance, simulation fidelity, and ease-of-use.

3.1.1.1 gem5-SALAM

gem5-MARVEL is an extension of gem5-SALAM [46] that uses an advanced dynamic graph execution engine based on LLVM [47]. gem5-SALAM instruments the LLVM IR (Intermediate Representation) to model DSAs using C descriptions of their functionality. Its main advantages are:

- It provides accurate representation of the accelerator datapath based on analysis of the LLVM intermediate representation of the accelerated algorithm.
- It provides cycle-level modeling through the dynamic LLVM-based runtime execution engine.
- It decouples the datapath and memory components to aid design space exploration and system optimization.
- gem5's tight integration enables seamless and intricate interaction between the accelerator and other system modules, including the CPU and the memory subsystem. Its high level of integration within the gem5 allows for complex interaction between the accelerator and other system modules, such as the CPU and the memory subsystem.

For the above reasons, we believe that gem5-SALAM is the ideal candidate for integration into our framework to complement the CPU side of the system with the accelerators side. This lets us evaluate the performance and reliability of many accelerator architectures, ranging from loosely coupled multi-accelerator configurations to tightly coupled coprocessors.

3.1.1.2 Architecture Overview

The gem5-based simulation infrastructure comprises two core components: the Compute Unit and the Communications Interface. The Compute Unit represents the custom accelerator's datapath, while the Communications Interface facilitates memory access, control, and synchronization through memory access ports, Memory-Mapped Registers (MMRs), and interrupt lines. The memory access ports allow parallel access to different memory types like scratchpad memories (SPMs) and register banks (these two types of memories occupy the largest part of the area of many accelerators). MMRs consist of configurable status, control, and data registers, enabling low-level device configuration and facilitating communication between the accelerator and the host, and between multiple accelerators in a cluster. By treating the accelerator as a memory-mapped device, the host can utilize the provided interrupt signals for synchronization without the need for constant polling.

Additionally, the gem5-based infrastructure includes Direct Memory Access (DMA) devices and custom memories that can be seamlessly integrated into accelerator designs, enhancing its versatility. Figure 2 shows the SoC architecture, including gem5-SALAM's features, on which our fault injector is built. Specifically, the accelerator designs that we tested are loosely coupled and communicate with the host CPU via MMRs and DMA transactions. The CPU writes the input and output memory addresses to the accelerator MMRs and directs the accelerator to start the computation. The accelerator transfers the data to its SPMs or Register Banks via DMA, performs the required

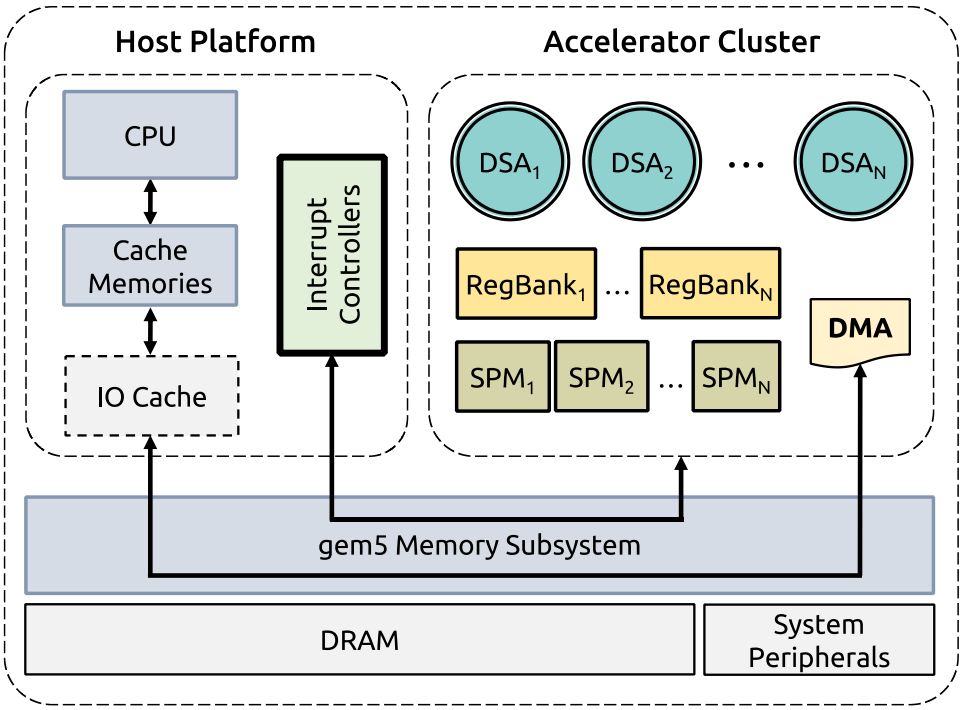


Figure 2: gem5-based SoC architecture and interconnection.

calculations, and transfers the data back to the system memory. After task completion, it notifies the host via a pre-defined interrupt.

3.1.2 Adding RISC-V Support

Currently, gem5-SALAM only supports the Arm ISA (Instruction Set Architecture) when it comes to the processor cores of the simulated system. However, the tremendous growth of the RISC-V ecosystem in the past few years, and its rapid adoption in both academia and industry, motivated us to port gem5-SALAM to also support the RISC-V ISA and system configuration.

The recent introduction of RISC-V full-system execution support into gem5 was highly beneficial to this endeavor. Nonetheless, the main challenge of extending the framework to also support the RISC-V ISA is to identify the Arm-specific components (i.e., the ISA dependencies) and translate them into the corresponding RISC-V ones. We summarize below the major components that had a strong dependency on the Arm platform. Specifically:

- The interrupt system used by gem5-SALAM hardware components that employed the Arm General Interrupt Controller (GIC) for posting interrupts to the host CPU,
- The automatic gem5 configuration script generator used an Arm gem5 configuration script as a template.

Next, we summarize these two ISA-dependent features, and how we converted them to enable RISC-V support into gem5-SALAM.

3.1.2.1 From (Arm) GIC to (RISC-V) PLIC

gem5-SALAM hardware components, use the Arm GIC (Generic Interrupt Controller) to send and receive interrupts to and from the CPU, aiding the synchronization between accelerator and host and removing the overheads of constant polling. We translated these functions at both hardware and software levels to the Platform Level Interrupt Controller (PLIC) present in the current gem5 RISC-V model. To make the transition from GIC to PLIC possible into the gem5-MARVEL we had to change the interrupt interface of the gem5 objects modeling the accelerator design to work with the RISC-V PLIC. We also had to modify the setup code and interrupt handlers that were specific to GIC, and alter them to the corresponding of the PLIC interface, as it is defined in the RISC-V ISA. After a lengthy debugging process, we managed to identify a bug in the gem5 PLIC implementation that resulted in the incorrect memory mapping of the interrupt claim space. After rectifying this issue, we were able to use interrupts to properly synchronize accelerator functions with the host CPU.

3.1.2.2 Automatic Configuration Script Generator

gem5-SALAM uses an automatic gem5 configuration script generator to simplify the development of accelerator-rich SoCs. This allows for complex configuration scripts to be generated by parsing a single YAML file that contains a description of the simulated system. To port the generator to RISC-V ISA and in the latest version of gem5, we swapped the Arm-specific script template to an already existing RISC-V full-system configuration script, made modifications to initialize the gem5-SALAM components, and added the accelerator memory-mapped addresses to the address ranges of the RISC-V platform.

3.1.3 Fault Injection

3.1.3.1 Overview

gem5-MARVEL is also a fault injection framework, that operates at the microarchitecture-level and supports transient and permanent fault injections to all hardware structures of the CPU and for the three prevailing ISAs (x86, Arm, RISC-V). The fault injection feature was implemented in the simulation framework to support the reliability aspect of the NEUROPULS project through the WP5.

Every fault injection campaign consists of a series of faults to be injected, which constitute the statistical sample, the simulations for every fault, the output results, and their parsing to estimate the vulnerability (or other desired metrics). As shown in Figure 3, depending on the parameters of the system (i.e., the microarchitectural details, the size, and structure of the hardware components, etc.), a different component of gem5-MARVEL is in charge of producing these fault mask files for a fault injection campaign

❶. Running scripts that serve as a campaign controller and are in charge of carrying out each step necessary for a full SFI campaign are used by gem5-MARVEL to manage fault injection campaigns. These scripts compose the library of the gem5-MARVEL. For injecting a single or multiple faults during system simulation, each fault injection simulation requires a fault mask input file ❷. The fault injection simulations come next ❸, in which the controller supplies the necessary inputs for the simulation and stores copies of the results. Multiple systems and/or CPU cores can be used to speed up the assessment, turning the time consumption problem to an infrastructure scale one. gem5-MARVEL can be configured to spawn multiple workers for a fault injection campaign to achieve 100\% utilization of the available hardware resources.

Each gem5-MARVEL instance runs an independent simulation, which corresponds to one injected fault (single or multiple), and once the simulation finishes, it produces several output files and logs ❹, with any required information about the specific fault injection run. Once the simulation is complete, all generated outputs and system states (including simulation statistics, output files, terminal I/O, operating system verbose (faults, messages, exceptions, etc.) are stored for post-processing. These are used to determine what the result of the injected fault on this simulation was. Finally, the results are parsed ❺. Figure 3 shows the preparation (also known as campaign initialization), simulation, and parsing phases of this process. gem5-MARVEL contains a pool of hardware configurations and benchmarks from which the user may choose to do the AVF or HVF assessment to make usage simpler and more user-friendly. Additional hardware configurations or benchmarks can be added to the pool at any time. The last step, after the parsing phase, is the final AVF (or other metric, e.g., HVF) estimation ❻.

3.1.3.2 Measuring AVF and HVF

gem5-MARVEL evaluates both the AVF and the HVF and provides accurate evaluation results using statistical fault injection for both methodologies. The Hardware Vulnerability Factor (HVF) of a hardware structure is the fraction of faults in the structure that are either activated within the hardware layer or exposed to a higher layer [50]. A

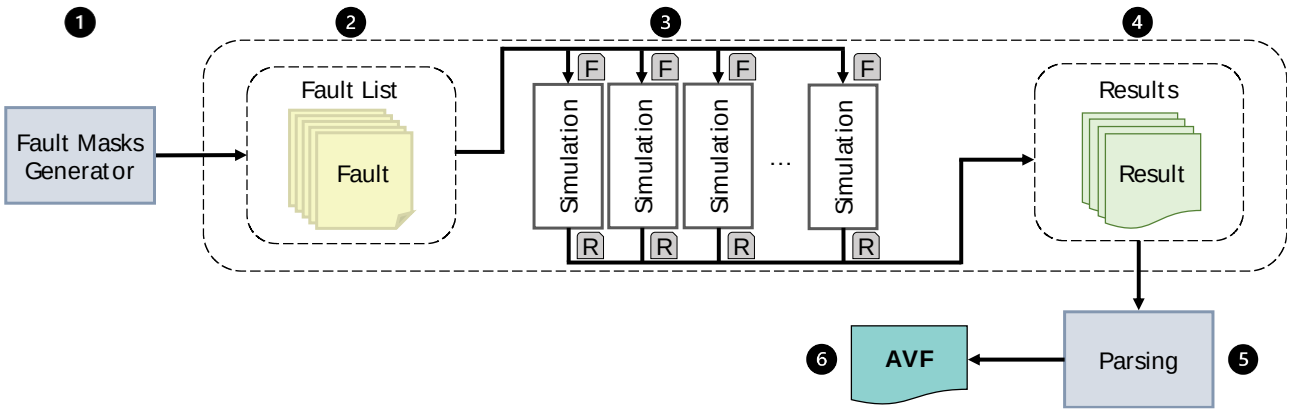


Figure 3: Fault injection campaign high-level layout with parallel workers.

hardware-visible fault is exposed to the user program once it reaches a software (or architecture visible) resource [51].

gem5-MARVEL employs two vulnerability evaluation methodologies of different layers: the HVF assessment [50] and the AVF assessment, providing the partial microarchitecture-dependent vulnerability and the full cross-layer vulnerability, respectively. As shown in Figure 4(a), the microarchitecture-dependent evaluation (HVF) focuses on the effects of hardware faults only until they first "touch" the software layer and stops at that point. Apparently, for accelerator designs, where the faults target the scratchpad memories of each design, the HVF and AVF analysis are identical. The reason is that the architecture of a domain-specific accelerator is totally different from the architecture of a general-purpose CPU.

In an accelerator design, any fault is eventually visible, unless the fault hits an invalid or unused cell of the scratchpad memory. In that case, the fault is characterized as masked. The HVF analysis considers as Benign faults, those faults that eventually get masked by a microarchitectural operation (e.g., a misprediction), and thus, the fault occurrence never reaches the commit stage of an out-of-order microprocessor (i.e., the fault is not architecturally visible). On the other hand, any fault that reaches the commit stage (i.e., architecturally visible), is considered as a corruption and participates in the total HVF measurement.

AVF (see Figure 4(b)) on the other hand, considers the entire program's execution, an architecturally visible fault may affect the program's operation or may get masked. A fault that either reaches the software and gets masked by a program's operation (e.g., the corrupted register value is never be used) or it is benign (it does not reach the software), it is a Masked fault for the AVF classification, since it does not affect the program. On the other hand, a fault that eventually reaches the software layer and subsequently affects the program's operation, it is classified either as an SDC or as a Crash. This is the process that gem5-MARVEL follows for the HVF and the AVF assessments.

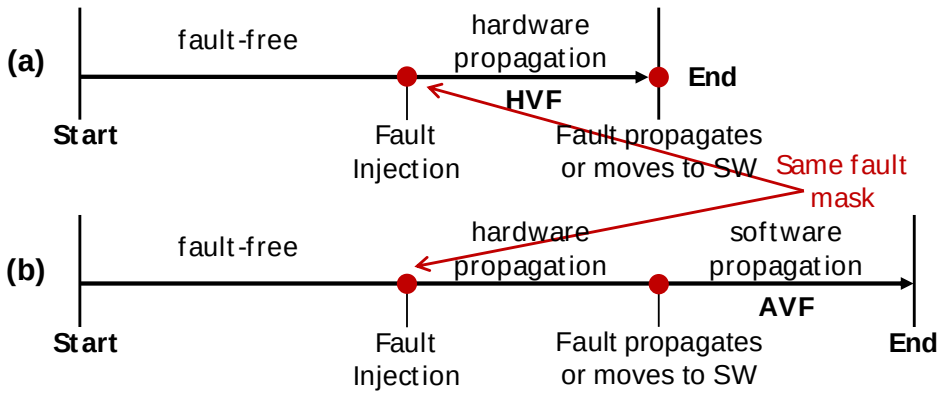


Figure 4: (a) HVF evaluation process; (b) AVF evaluation process, which may also consider the HVF evaluation.

3.1.3.3 Implementation of gem5-MARVEL

The main objectives of the gem5-MARVEL fault injector are as follows:

(1) gem5-MARVEL ensures accuracy in the vulnerability (AVF and HVF) reports it delivers. To achieve this objective, applications, benchmarks, or accelerator design models, are executed until completion unless a fault is earlier identified to be masked (e.g., the fault is injected into an invalid cache line, or it is overwritten before being read). This allows for capturing the final program impact of faults.

(2) gem5-MARVEL increases the speed of fault injection campaigns, especially for transient faults evaluations. To this end, multiple workstations can be utilized, and the gem5-MARVEL is optimized to terminate a fault injection run immediately in scenarios where a fault is inserted in an invalid or unused entry of a hardware structure or when a faulty entry is overwritten before ever being read. These optimizations provide significant speedup for individual runs across all benchmarks and structures, resulting in significant savings in injection campaign time.

(3) gem5-MARVEL is highly configurable, as discussed in section 3.1.3.1.

gem5-MARVEL is composed of a modified version of gem5 that permits fault injection and includes instrumentation for controlling and executing simulation campaigns, through a library of automated scripts. It supports various types of fault models, including transient and permanent faults, in single or multiple configurations or any combination. The framework incorporates fault masks that specify the injection of faults, which can contain one or multiple faults for the simulation.

It has enough information to accurately target one or multiple components at a certain time or period. Each fault is described by: (i) thread ID (i.e., simulated CPU ID), (ii) microarchitectural component, (iii) position within the component (bit granularity), (iv) fault model, (v) clock cycle of the occurrence, and (vi) mask effect (bit flip, stuck-at 1, stuck at 0). Moreover, gem5-MARVEL uses configuration presets. Each configuration preset consists of attributes, such as the ISA, memory configuration, CPU core (in-order, out-of-order, etc.), multicore setup, system setup, disk images, Linux kernel versions, etc., along with gem5-MARVEL attributes and a list of supported hardware structures for injections. New configuration presets can be easily added to cover different requirements. In such a way, it is straightforward for any user to add any desired design parameters to support 100s of different purposes.

gem5-MARVEL provides high flexibility and ease of expansion, which makes it suitable for any reliability evaluation study and any new microprocessors or accelerator design. gem5-MARVEL has a modular design, which employs inherent checkpointing features of gem5 to ensure that the faults affect only the execution of the program being studied. To accelerate the injection process, we have incorporated additional functionality into gem5's checkpointing mechanism to preserve both the microarchitectural and architectural states (gem5 preserves only the architectural state). Consequently, it becomes possible to study long-running workloads (without long warm-up periods) and initiate the analysis from any desired time point while maintaining the accurate microarchitectural state, including the correct data in cache memories.

3.2 Accelerator Designs and Design Space Exploration

Building on the foundation described in the previous section, this section explores the diverse design space of heterogeneous computing systems. It details the integration of diverse photonic neuromorphic accelerators found in the literature, showcasing how the infrastructure facilitates exploration and optimization of future designs.

3.2.1 Configurations, Components, Benchmarks & Accelerator Designs

gem5-MARVEL supports all dominant 64-bit ISAs, i.e., Arm, x86, and RISC-V, and for the CPU side evaluations we present in the following sections, we model the same out-of-order microarchitecture for every ISA (microarchitectural modifications can of course be arbitrarily implemented in gem5), as shown in Table 1. This is a commonly used configuration for modern mid-end commercial CPUs.

gem5-MARVEL targets tens of hardware structures for both CPUs and DSAs. gem5-MARVEL supports various CPU microarchitectural components as fault-injection targets and performance analysis, including integer and floating-point physical register files, memory cache levels, load and store queues, reorder buffer, TLBs, register renaming unit, etc. We focus on five major CPU structures for fault injection: (1) Integer Physical Register File, (2) L1 Instruction Cache, (3) L1 Data Cache, (4) Load Queue, and (5) Store Queue. For DSA designs, fault injection and performance analysis results are reported for scratchpad memories and register banks, since these are the available DSA hardware structures.

Scratchpad memories are high-speed internal memories located close to accelerator functional units (see Figure 2}; DSAs consist of functional units). They serve as temporary

Table 1: Major Simulator Configurations for each ISA.

Parameter	Value
ISA	RISC-V / Arm / x86
Pipeline	64-bit OoO (8-issue)
L1 Instruction Cache	32KB, 64B line, 128 sets, 4-way
L1 Data Cache	32KB, 64B line, 128 sets, 4-way
L2 Cache	1MB, 64B line, 2048 sets, 8-way
Physical Register File	128 Int; 128 FP
LQ/SQ/IQ/ROB entries	32/32/64/128

storage for ongoing calculations and data manipulation, tailored to the specific needs of the accelerator design. In our accelerator configurations, scratchpad memories play a crucial role in handling input, output, and intermediate data for accelerated algorithms. Input data are transferred from the CPU to accelerators via DMA, and the results are DMA'd back to the CPU after processing. Register banks fulfill a comparable role to SPMs but function as slower and less complex components, showing a delta delay between the moment a register is written to and when the data becomes available for read operations.

We employ a comprehensive and diverse set of 15 workloads from the MiBench benchmarks suite [48] for all 3 different CPU ISAs. For the DSAs evaluation, we present results for fault injections in the two largest types of accelerator memory structures: the scratchpad memories and the register banks of the designs. We employ 8 MachSuite accelerator designs [49] (see Table 2) and we measure their AVF when running full-system simulations based on the RISC-V ISA for the CPU side. For each structure, 1,000 single-bit faults are randomly generated following the uniform distribution as defined in [52], resulting in nearly 250,000 fault injection runs in total for all benchmarks and

Table 2: Target Injection Components for each Accelerator Design.

Accelerator	Component	Memory Size (Bytes)	Memory Type
BFS	EDGES	16,384	Register Bank
	NODES	2,048	Register Bank
FFT	IMG	8,192	Scratchpad Mem
	REAL	8,192	Scratchpad Mem
GEMM	MATRIX 1	32,768	Scratchpad Mem
	MATRIX 2	32,768	Scratchpad Mem
MD_KNN	NLADDR	16,384	Scratchpad Mem
	FORCEX	2,048	Scratchpad Mem
MERGESORT	MAIN	8,192	Scratchpad Mem
	TEMP	8,192	Scratchpad Mem
SPMV	VAL	13,328	Scratchpad Mem
	COLS	6,664	Scratchpad Mem
STENCIL2D	ORIG	32,768	Scratchpad Mem
	SOL	32,768	Scratchpad Mem
	FILTER	360	Register Bank
STENCIL3D	ORIG	65,536	Scratchpad Mem
	SOL	65,536	Scratchpad Mem
	C_VAR	8	Register Bank

accelerator designs, 3 different 64-bit ISAs, and all hardware components for CPU and DSAs. We follow the widely adopted formulation of [52] for the statistical fault sampling calculations; our 1,000 faults correspond to 3% error margin with 95% confidence level.

3.3 Performance and Reliability Evaluation Metrics

This section provides insights into the assessment of critical design parameters. It illustrates the methodologies that will be employed to evaluate performance and power efficiency of CPUs and accelerator designs, showcasing how the NEUROPULS simulation infrastructure contributes to the energy-efficient design of computing systems.

3.3.1 Domain-Specific Accelerators Vulnerability Evaluation

In this subsection, we report the AVF results from fault injection on eight different DSA designs, targeting their large on-chip SRAMs: scratchpad memories (SPMs) and register banks (RegBanks). These components store input, output data, and intermediate results of accelerated algorithms. For each DSA, we select representative SPMs and RegBanks for independent fault injection campaigns to assess their AVF, as shown in Table 2. Figure 5 presents the breakdown of SDC and Crash fault effects, which together constitute the complete AVF, for all designs.

The *BFS* DSA design uses two distinct RegBanks for accessing the EDGES and the NODES of the input graph and does not use any SPMs. The AVF of the EDGES RegBank is 35%, while the AVF of the NODES RegBank is 20%, and as shown in Figure 5, nearly all fault effects of *BFS* are Crashes. This is due to data from both RegBanks being used as

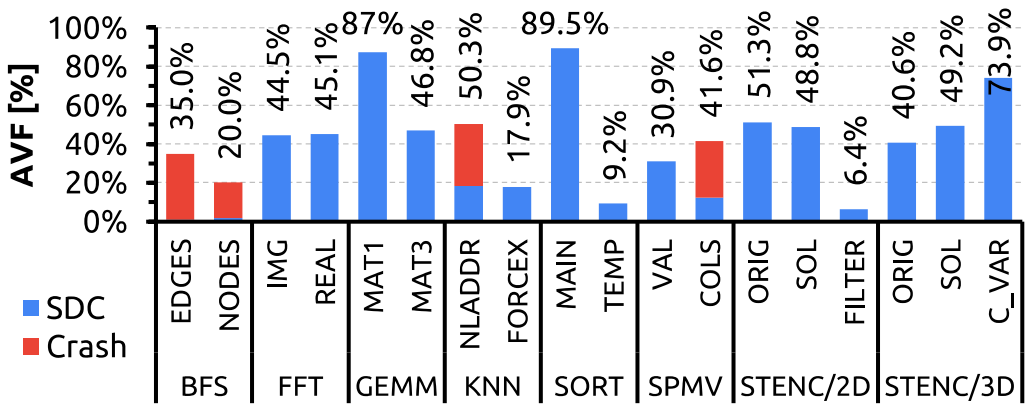


Figure 5: AVF for different accelerator designs with different injection targets.

indices for graph traversals by the accelerator hardware. As a result, faults in any RegBank lead to either excessively long execution times or out-of-bounds memory accesses that surpass the size of the system's physical memory.

The *FFT* design utilizes two SPMs to store the imaginary (IMG) and REAL components of the algorithm's output. The IMG SPM has an AVF of 44.5%, while the REAL SPM has an AVF of 45.1%. These AVFs are quite similar because a fault in either the imaginary or real part of the *FFT* result has an equal probability of corrupting the accelerator output. Interestingly, as shown in Figure 5, all faulty runs result in SDCs, since the SPM data is not utilized by any accelerator control logic or used as indices for memory accesses.

The same pattern is also observed in the *GEMM* and *MERGESORT* designs. GEMM holds the input data of one of the matrices to be multiplied in one SPM and the result of the matrix multiplication in another SPM, while MERGESORT uses two SPMs to store the main array data and temporary intermediate values. As shown in Figure 5, the output SPM (MATRIX3) of *GEMM* has significantly lower AVF than the input SPM. This can be attributed to the injected faults in the output SPM being overwritten much more often because the input SPM gets written to only once by the DMA device when the accelerator is initialized, whereas the output SPM gets written to for the entire accelerator runtime.

For *MERGESORT*, the TEMP SPM has significantly lower AVF than in MAIN SPM, which can be attributed to the overwriting of numerous faults due to the continuous stream of memory writes to the SPM. Similar observations are valid also for the remaining DSA designs, which are omitted due to space limitations.

Observation: Most accelerator designs result in very high SDC rates in the presence of faults.

Architectural Implication: Most accelerators are datapath-heavy, with few control dependencies on their input data. This characteristic enables accelerators to efficiently execute data-intensive tasks by prioritizing parallel processing of large data volumes. However, it also amplifies the probability of SDCs in the presence of transient faults. Therefore, fault mitigation strategies for accelerators should primarily focus on data corruptions and not the control flow.

3.3.2 Performance-Aware Comparison

We explore how different computing systems can be fairly compared using gem5-MARVEL regarding reliability and show how vulnerability measurements can be combined with system performance. We showcase our methodology with a test case scenario by comparing the reliability of two different computing systems: a standalone RISC-V CPU and a standalone DSA. For a fair comparison, 4 algorithms are properly implemented to run and are modeled in both computing systems. These are a Matrix Multiplication Algorithm (i.e., GEMM), BFS, FFT, and KNN algorithms (as described in Table 2). AVF is a pure reliability metric that provides no information about system performance.

AVF alone cannot provide any insights on the tradeoff between performance and reliability of a chip. To this end, gem5-MARVEL is also able to compute a new simple reliability metric named *Operations per Failure (OPF)*. OPF is the number of times a workload is executed before a system failure happens and is computed using the following formula: OPF = OPS / AVF, where OPS (Operations per Second) is the number of operations (i.e., tasks) that the compute unit can perform during 1 second. Assume, for example, the Matrix Multiplication algorithm, which performs 2 x N^3 operations, where N is the size of the matrices. Thus, OPS = 2 x N^3 / Exec_Time.

The OPF metric enables a combined analysis of performance and reliability into a single metric. For the same workload that runs on different platforms (a CPU or an accelerator in our example), larger OPF values indicate a better tradeoff between reliability and performance (larger number of correct executions over time).

Figure 6 showcases the pure reliability evaluations against the new proposed metric for the 4 algorithms, which considers the performance of the platform and presents the tradeoff between performance and reliability in a single metric. As Figure 6 demonstrates, while the AVF (left graph) shows that all 4 algorithms running on the accelerator (the DSA labels in the x-axis) are significantly more vulnerable than running on a RISC-V CPU, the combined vulnerability and performance metric OPF (right graph) shows that the same algorithms can be executed in the accelerator significantly more times than in the CPU before observing a system failure (i.e., better tradeoff between performance and reliability for the accelerator design).

Observation: Although the accelerator design is more vulnerable, it demonstrates a better tradeoff between performance and reliability.

Architectural Implication: The higher OPF value suggests that the accelerator design offers increased resilience and can maintain stable operation for a more extended period, making it more suitable for executing the algorithm in real-world scenarios where reliability is crucial. Although AVF highlights its higher vulnerability, the advantages in OPF emphasize the benefits of using the accelerator design to achieve improved performance while maintaining an acceptable level of reliability.

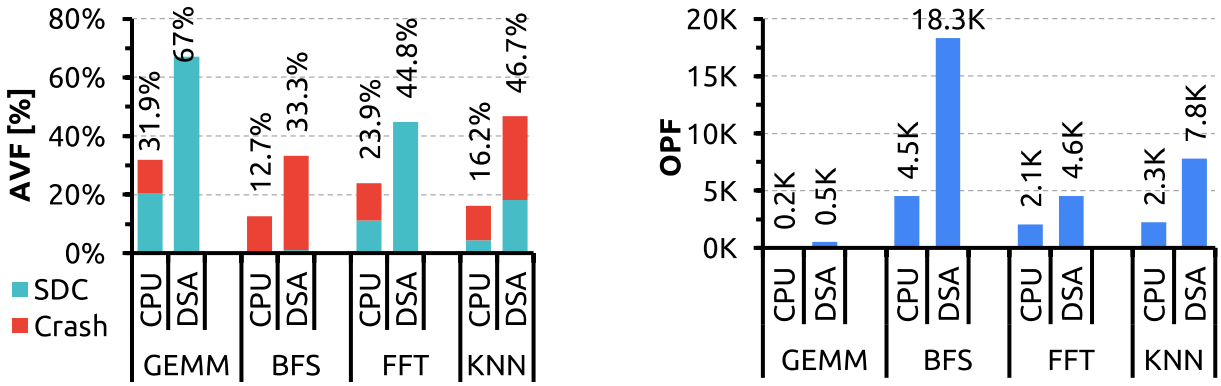


Figure 6: Breakdown of SDC and Crash AVF of 4 algorithms for both CPU and accelerator (left graph), and the OPF for CPU and accelerator (right graph).

3.3.3 Accelerator Design Space Exploration

The accelerator design side of gem5-MARVEL is capable of modeling data-dependent control accelerator execution by independently evaluating the static and dynamic elements of the system, which allows for more configuration options facilitating design space exploration. While it provides a default hardware profile that creates a 1-to-1 mapping of instructions to functional units, the user can also specify constraints on individual hardware resources to enforce functional unit reuse. The gem5-MARVEL ecosystem utilizes additional parameters in the "config" and "hardware profile" to fine-tune the system. The hardware resource model is generated dynamically from YAML configuration files that define a hardware profile.

This change enables the user to create and define hardware profiles at the granularity of individual accelerators within the cluster. The hardware model generated from this profile defines how functional units and instructions are handled during runtime simulation. This allows users to redefine any parameter within the hardware model, including creating and linking customized instructions and functional units, and begin a new simulation without needing to recompile or rebuild the system [46].

In this subsection, we showcase how the gem5-MARVEL can leverage the inherent features of both gem5 and explore the accelerator design space from the reliability point of view. As a case study, we show the reliability evaluation of the MachSuite GEMM accelerator for different degrees of parallel processing, i.e., amount of parallel functional units. Figure 7(a) shows the AVF of SPM1 (holds the data of one of the input matrices) for the different accelerator configurations. We can see that as the parallel functional units are reduced, the AVF (the vulnerability) increases significantly. This may be attributed to the slower SPM access rate that allows more faults to propagate to the output without being masked. %This observation applies to this kind of memory since the data occupy the entire SPM from the beginning of the processing.

Note that the data occupies the entire SPM from the beginning of processing. Along the same lines, in Figure 7(b) we can see the differences of performance and area for the GEMM accelerator design. It is clear from these graphs that based on the outcomes of gem5-MARVEL we can find the optimal tradeoff between reliability, area, and performance.

Observation: The AVF increases significantly when the number of parallel functional units in DSAs is reduced.

Architectural Implication: With fewer parallel functional units, faults have a higher chance of affecting the output data due to the data occupying the entire SPM from the beginning of processing. This implies that designs with fewer parallel functional units may be more susceptible to transient faults, potentially leading to incorrect results during computation. To enhance fault resilience, future designs could consider optimizing the access rates to the SPM and implementing suitable fault-tolerance mechanisms.

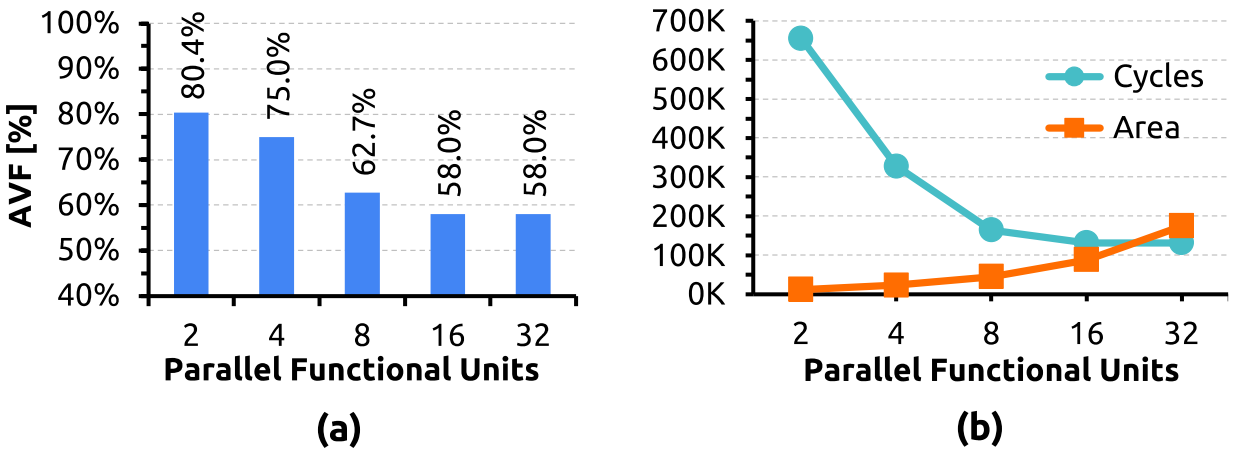


Figure 7: Gemm accelerator design for five different configurations of available functional units, showing (a) the AVF for different functional units, and (b) the performance and area statistics.

3.4 Reliability Assessment of the Heterogeneous Architecture

gem5-MARVEL is a state-of-the-art microarchitecture-level fault injection framework that targets heterogeneous SoC architectures, including both CPUs of different ISAs and DSAs. Such a framework can provide numerous invaluable insights into the overall systems' resilience. We group the fundamental insights that gem5-MARVEL can deliver:

- **CPU ISA comparison**: By injecting faults into CPUs with different ISAs, gem5-MARVEL can compare fully unprotected designs or any error detection or correction mechanisms at the software or hardware layer. This analysis reveals which ISA performs better under fault conditions and which requires stronger protection.
- **Accelerator impact**: Fault injection in DSAs helps understand how resilient they are to different fault scenarios, which is critical in several application domains.
- **System-level resilience**: gem5-MARVEL allows analyzing the overall system's resilience. By injecting faults at different locations in the system, it can be observed how the CPUs and accelerators interact and recover from faults. This knowledge is valuable for designing next-generation fault-tolerant systems at scale.
- **Error recovery mechanisms**: gem5-MARVEL can shed light on the effectiveness of error protection (detection and correction) mechanisms in both the CPUs and accelerators. This insight can guide improvements in the system's error handling and fault recovery strategies.
- **Power and performance trade-offs**: gem5-MARVEL can help the concurrent assessment of the complex tradeoffs among power consumption, performance,

and resilience. It provides insights into how protection mechanisms can impact performance and power efficiency.

4. Memory Hierarchy Models

The emergence of novel memory technologies offers significant advantages over traditional ones and provides designers with opportunities for extensive design space explorations. These emerging memory technologies have the potential to revolutionize system architecture and performance.

The focus of this task is the modeling of the system memory hierarchy for systems that employ neuromorphic accelerators hardware (including those considered in the Use Cases of the project). The novel aspect of this task is the insertion of PCM memory models (with parameters coming from the design and characterization work in WP2-WP4) at different levels of the memory hierarchy and the evaluation of its contribution to the main aspects of the system: performance and power consumption which combined determine the energy efficiency of the system.

4.1 NVMain

This task focuses on identifying and configuring the simulator infrastructure needed to validate and extrapolate the optic chip behavior. In this sense, first it was needed to identify from the different available simulators those that can satisfy the project requirements. NVMain [54] was selected, see Figure 8 The main reason is because it is a specialized framework tailored for Non-Volatile Memory (NVM) systems, which introduces two pivotal features that enhance its appeal for NVM memory modeling. Firstly, its modular design facilitates integration and toggling of various techniques aimed at addressing endurance and write-related challenges, as well as hybrid memory configurations, enabling thorough exploration of architectural design spaces. Secondly, NVMain stands out as the first main memory simulator to incorporate data values efficiently, which proves invaluable in assessing application-level implications of NVM designs, particularly those based on multi-level cell architectures. In the figure below, an overview of the NVMain architecture is described.

NVMain configuration involved 2 steps: integration into the gem5 simulator and the identification of the parameters that need to be tailored to imitate the chip behavior. Regarding the integration, NVMain was originally integrated by its creators into gem5 on early versions to analyze how the PCM memories such us Intel Optane had on systems where they were integrated. That integration was abandoned on a fork of the NVMain project, leading to the complete obsolescence of the integration due to hard changes on the simulator insights. Nowadays, the NVMain simulator integrated with gem5 can only be used on containers targeting that older version, which is incompatible with the new architectures and improvements included on the newer versions of the gem5 simulator.

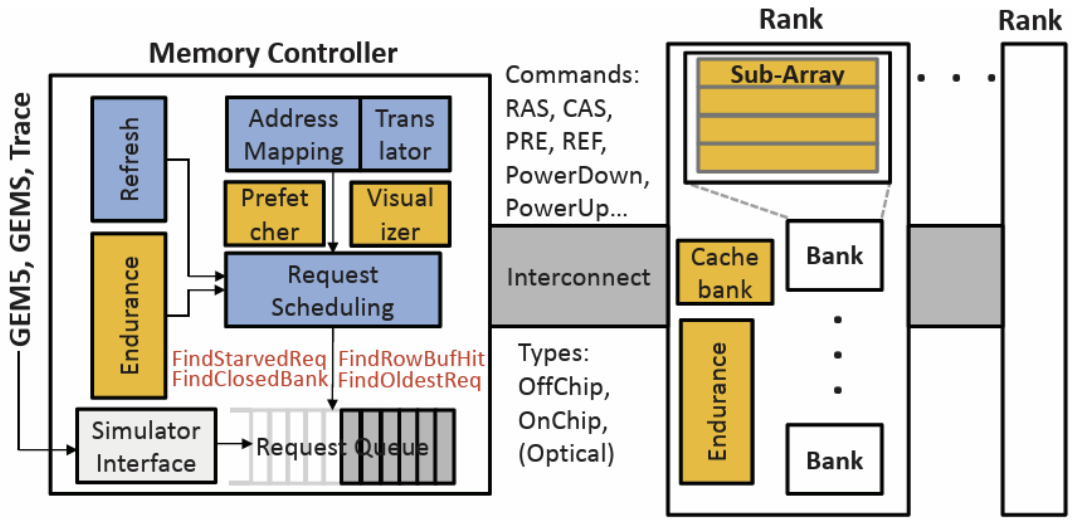


Figure 8: Overview of NVMain Architecture. Only one memory controller with one channel is shown [54].

To address this, we made an effort to reintegrate again NVMain on the latest version of gem5. That is, we performed an integration from the older gem5 version into gem5 22.0.0.2. To this end, the latest common commit between NVMain and gem5 main branch has been tracked. Patches have been applied on the source code to solve the integration problems in that old commit. Since the last integration commit it has been found that it is not possible to apply a clean merge on gem5 due to code refactor changes inside the memory component, cache subsystem and their own cache protocol (based in ruby, which has been discontinued). In this sense, the structural changes inside the gem5 code have been tracked to fully re-integrate NVMain.

Furthermore, NVMain uses Python version 2, which is not compatible with gem5 as it uses Python version 3. An effort has been made to make NVMain compatible with Python 3 to allow integration into gem5.

Regarding the parameters, we identified the configuration parameters that can imitate similar behaviors on NVM to those that may be required within the project. In this way, from the integration of the NVMain onto the gem5 simulator is expected to analyze the behavior of the optical non-volatile memories (PCM) used in this project, due to NVMain simulator highly configurable interface, i.e., setting timings or energy needed to perform an operation. In this way, we can compare the effect on performance or power consumption compared with other state of the art technologies.

4.2 NVMinterface

NVMInterface is a gem5 component which allows instantiation and accurate simulation of NVM memories. It is instantiated in gem5 as any other memory controller, and it is assumed that a *MemCtrl* object with a NVM Interface will be used. Its main difference with respect to NVMain is that it is readily available in gem5 version 22.0.0.2, and it is designed to simulate the main system memory. To that end, it provides different configurable parameters as well as statistics to accurately model the device and collect metrics of its use. For example, some of these metrics include read or write bursts per bank, which only make sense in the context of DRAM but not for caches.

Although NVMInterface is already supported in gem5 and can be configured as the memory type of the system, an effort has been made to test it with some configurations like those that may be required within the project, to ensure the component is ready to be used. To that end, different configurations and instantiation contexts have been explored:

- Different configurations of NVM were tested, with varying number of ranks.
- The component can be used directly as a memory controller or as part of a HybridMemory component, where it could be instantiated alone or with a DRAMInterface if needed.

4.3 Approaches for Operating PCM Memories and related modeling

PCM-based memories have attracted considerable interest over the last two decades due to the several advantages that they offer such as high switching speeds (order of ns or lower), medium-to-high endurance, high re-programmability (>= 10^6), good scalability and integration with CMOS compared to flash memories [56]. Initially used for rewritable DVDs, they have been considered more recently in integrated photonic applications ranging from in-memory and neuromorphic computing to pure photonic DRAM [57] [58] [59].

One of the key advantages of an optical read-out approach consists of being able to achieve very low latency, but also very low optical signals (limited by SNR and photodetector sensitivities) which translates into overall low-power consumption. However, the writing phase poses a series of trade-offs from an architectural point of view to operate PCM memories.

In the next two sections, we will discuss two different ways to operate PCM memories in terms of the writing phase, i.e., optically and electrically, while we will focus solely on an optical read-out approach. Modeling strategies will also be discussed.

4.3.1 Optical writing approach

In an optical writing approach, the PCM patch located above the photonic waveguide is stimulated by short optical pulses which gradually change its crystalline (and amorphous) fraction, thereby encoding data directly in its phase.

Depending on the SNR that can be achieved from a writing as well as reading point of view, a certain bit resolution will be achieved. While this approach is the most promising for rapidly writing PCM memories (below ns for good heat evacuation and suitable material properties), it comes with 2 major limitations: (i) the optical waveguides for read and write operations shall be separated – this is related to the fact that optical writing signals shall not modify multiple memory elements and (ii) to achieve low optical losses different waveguide materials might need to be used e.g., Si (for read-out signals) versus SiN (for writing signals) due to two-photon absorption effects which may take place for high-power pulses. Other strategies relying on crossbar architectures leveraging optical crossing geometries will also be explored for matrix-matrix multiplication.

4.3.2 Electrical writing approach

Another approach that is under investigation concerns the possibility of writing the PCM memories with short electrical pulses, similarly to what is done in D4.1 for the neuromorphic architectures, to set the weights of the accelerator. While this approach does not require to increase the number of waveguides, it does increase the number of routing wires. However, given the possibility of using multiple interconnect layers for wiring, this approach simplifies the routing schemes that are required. However, it comes with a major caveat that electrical contacts cannot come too close to an optical device. Therefore, the writing speed is limited by the thermal constant of the dielectric distribution around the Si waveguide, which is hundreds of ns from D3.4. Power consumption is also increased in this scenario due to the absence of local heating of the PCM patch as instead the case for the optical writing scenario.

4.3.3 Benchmarking strategies and modeling

To benchmark the PCM memories, we will be looking at how they perform under classical tasks such as matrix-matrix multiplication in the optical domain where, e.g., the PCM elements store the matrix elements for one of the matrices to multiply, while the other one is encoded in the incoming optical signals in a crossbar architecture.

We will also investigate typical read/write operations for memories under real computing flow scenarios. Parameters such as power consumption, latency, speed/throughput and distributions will be obtained to provide input to the gem5 platform.

To extract these parameters, compact behavioral models such as the one presented in D3.4 based on a transfer matrix approach for PCM-based Mach-Zehnder interferometers (written by electrical pulses) are currently being explored.

4.4 Integration of Different Approaches in the Simulator – Future Directions

As described in the previous section, PCM memories have a very different nature. This presents a challenge in their integration in the simulator, which is required to accommodate these different types. The use of PCM within the simulator is a two-dimensional problem: on the physical side, PCM memories may have very different features, including differences in voltages, operation timing, among others. On the purpose side, PCM memories may be used as part of the processor caches, which present a particular set of challenges and simulation requirements, and as the DRAM, which may present different challenges and requirements.

To solve the physical challenge, we have worked on collecting a list of configurable parameters for the simulator components. This task has resulted in analyzing a series of NVMain configuration files, understanding the meaning of the configurable parameters, and classifying them into a category. To perform the classification, we have developed a taxonomy where each parameter is classified as one of the following classes: General Memory, DRAM Energy, Energy parameters, General Geometry parameters, Endurance, Power Down, Simulation Control, Device Timing, Memory Controller, and MLC.

Regarding the two different purposes of PCM memories, we have identified and described two gem5 components which perfectly match the intended purposes of the PCM. On the one hand, NVMain covers its use as cache memories, while NVMInterface covers its use as DRAM memory. On the other hand, the configuration parameters and the statistics generated by each component match their purpose and provide a flexible way of integrating the PCM into different levels of the memory hierarchy of the simulator.

5. Hardware/Software Security Models and interfaces

This deliverable section outlines the progress and collaborative efforts dedicated to modeling Hardware Security Models within the project's first year. The structure is divided into two key subsections, each addressing critical aspects of our endeavors: Hardware Security Primitives and Simulation Modeling Approach/Decisions.

5.1 Hardware Security Primitives (definition, interface, basic blocks)

In this subsection, we explore the integral components of hardware security primitives, focusing on their definition, interface, and the basic blocks that constitute the foundation of our project.

- **Definition**: We delve into the conceptual underpinnings of PUFs, highlighting their significance in enhancing hardware security through their inherent unpredictability and uniqueness. This part clearly explains PUFs' role and potential applications in cryptographic systems.
- **Interface**: The interface design ensures that PUFs can seamlessly integrate with conventional digital devices. This involves detailing the interaction protocols between the PUFs and other system components, particularly emphasizing compatibility with the gem5 simulator for effective simulation and integration.
- **Basic Blocks**: Identifying and modeling the basic blocks of PUFs form the core of our project. This includes the detailed exploration of the photonic components of PUFs, outlining the specific modeling requirements essential for their integration into digital systems and ensuring their functional efficacy in a simulated environment.

NEUROPULS comprises several hardware modules, with the Application-Specific Integrated Circuit (ASIC) playing a pivotal role. The ASIC serves dual functions as the primary AI processor, handling complex computations, and as a Physically Unclonable Function (PUF) for bolstering security in various applications. This makes it the primary hardware security primitive of the NEUROPULS project. The ASIC's intricate design incorporates, on the PUF side, essential components such as the Photonic Circuit (and with Modulators, Photodetectors, and Power Drivers) and a set of Electrical components, contributing to its overall functionality. This is depicted in Figure 9.

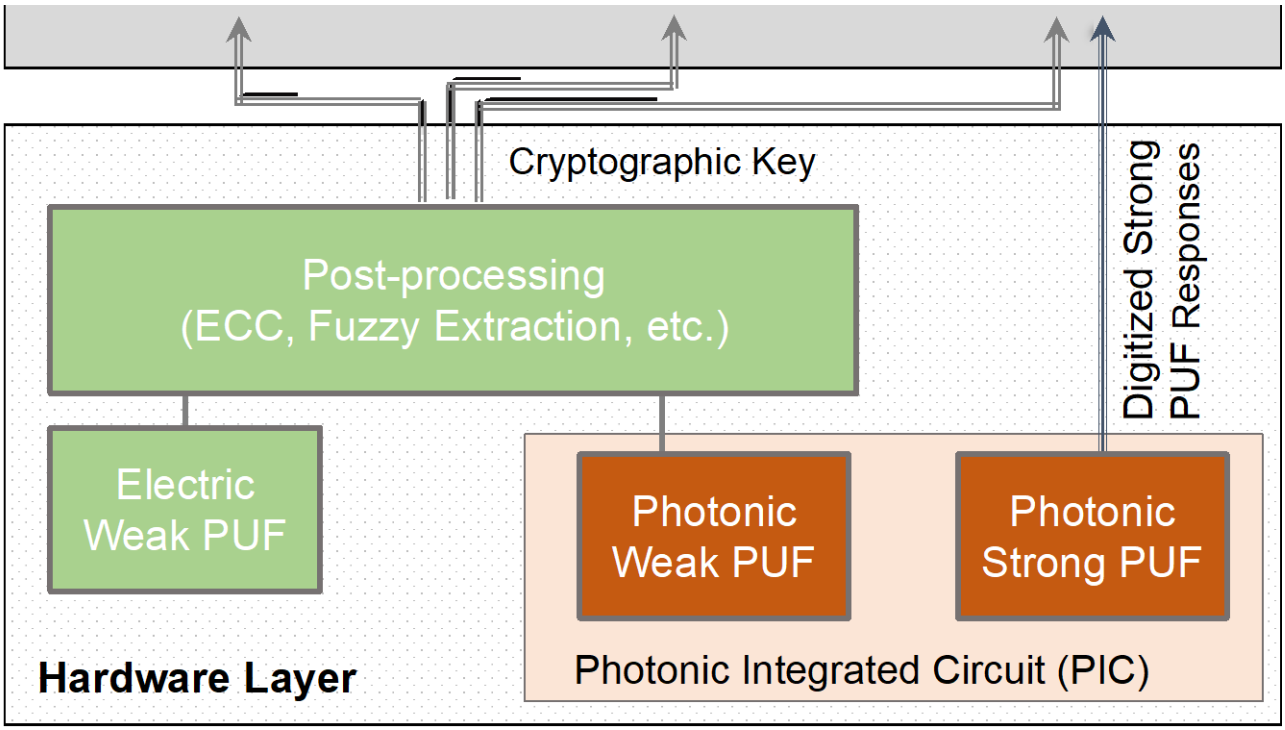


Figure 9: Hardware Description of NEUROPULS PUF Components.

Precise modeling should include all in the communication part as foreseen in D4.3 and D6.1, covering critical elements like the Analog-to-Digital Converter (ADC), Digital-to-Analog Converter (DAC) along with all the photonic components. In real hardware, these components work seamlessly in tandem, facilitating efficient communication and collaboration between the main processor and the ASIC. The high-level model definition starts by identifying the necessary modules:

- **PIC (Photonic PUF part)**: It will be the primary processing unit for generating the CPRs. The ASIC contains:
    - **PUF architecture** will mimic the architecture described in D4.3 regarding single components and their connection, serving as a source of natural randomness. The main components will be a multi-mode interference (MMI) device and resonators modeled with a floating-point single precision for maximum computational speed. While the modeling follows the mathematics behind each component behavior, requiring some extra time for computation, this modeling approach paves the way for accurate accounting of what contributes to a PUF creation in the real hardware. Moreover, it allows for precise time and power estimation. This architecture will be complemented with a purely functional component in which the users will be required to fill a table of known CRPs for faster emulation of the functionality without carrying an accurate time and power estimation.

  - **Modulator**: it will model the behavior of devices that control the intensity of laser light, influencing data transmission through variations in light amplitude. As accurate modeling will be provided here, in floating point precision, it will be feasible to explore later if variations in modulator behavior introduce additional sources of randomness, contributing to the uniqueness of the PUF signatures as it does in the real hardware.
  - **Photodetector**: it will model the devices that measure the power of light, converting optical signals into electrical signals. This modeling follows a pure analog flow, assessing the light power at the end of the photonic circuit, capturing and converting output data.
- **Power Drivers**: The power drivers are required to model the components that change the voltage within the photonic circuit, effectively programming weights in the circuit. They will be primarily necessary for the accelerator modeling to decouple programming and inference precisely. Still, they might serve the PUF modeling to explore the future adoption of the accelerator itself as a source of CRPs.
- **GPIO Interface**: Modeling the GPIO interface is crucial for time-accurate modeling of the general photonic interaction as it accounts for a lower time frequency in the programming of the Photonic PIC. Without it, a general model of a DMA transfer will not be able to distinguish between the operation modes (inference, programming, operation selection, CRPs request, etc.) operating the correct time and power modeling.
- **DAC (Digital-to-Analog Converter)**: The DAC converts digital signals to analog signals for input to laser modulators. In the context of the PUF, the modulation of laser light intensity, controlled by the DAC, contributes to introducing variability in the Photonic Circuit, a key aspect of PUF functionality.
- **ADC (Analog-to-Digital Converter)**: This component is vital for the PUF as it converts analog signals from photodetectors to digital signals for processing. The analog signals from the photodetectors, capturing variations in light power at the end of the photonic circuit (part of the PUF), are translated into digital form by the ADC. This digital data is then used for further processing, contributing to the overall PUF functionality.
- **TIA (Transimpedance amplifier):** The TIA needs to convert from the current signal (photocurrent) of the photodetector to a suitable voltage which optimally spans the dynamic range of the ADC.
- **Electric Weak PUF**: As described in Figure 9, the complete PUF architecture at this stage of the project couples the Photonic PUFs with an Electric weak PUF.

5.2 Simulation Modeling Approach & Design Decisions

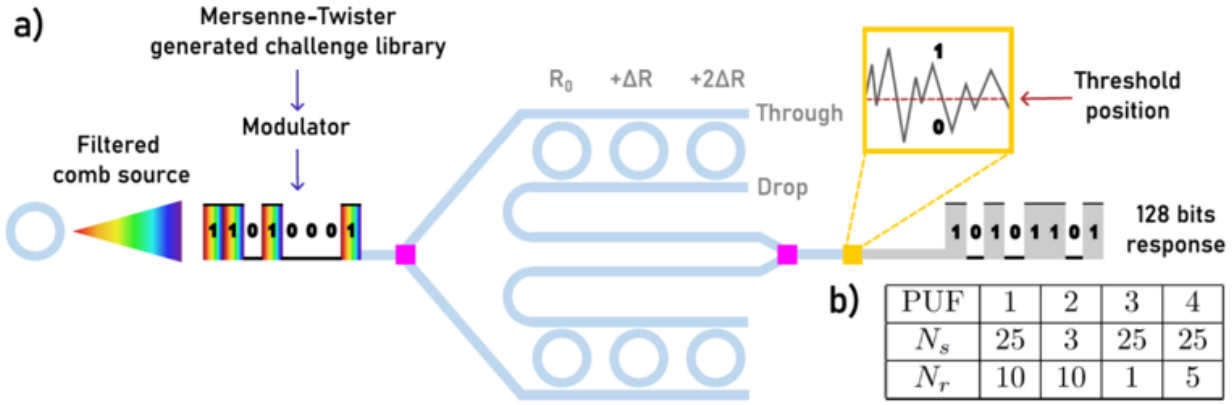
This subsection addresses the strategies and considerations underlying our simulation modeling approach on gem5-MARVEL, including the pivotal design decisions that guide our simulation efforts.

- **Modeling Approach**: A comprehensive review of the simulation tools and methodologies employed to model PUFs is provided. This includes examining the capabilities and limitations of current simulation tools to accurately replicate the complex behavior and security features of PUFs.
- **Design Decisions**: Critical design decisions are highlighted, focusing on the functional requirements for effectively exploiting PUFs within simulated environments. This includes the integration of PUFs with standard digital devices and the adaptation of simulation models to support ongoing developments in security architecture.
- **Integration and Functional Requirements**: The integration of PUFs into broader system architectures is discussed alongside the functional requirements that guide this process. This involves detailing the efforts to synchronize the physical device modeling with digital simulation platforms, ensuring that the PUFs' unique attributes are effectively captured and utilized within the simulation framework.

5.2.1 ASIC Modeling

The foundation of ASIC modeling relies on using C++, primarily due to its pivotal role as the principal language in the gem5-Marvel simulation framework. Throughout this process, each module of the ASIC is crafted and replicated in C++ to closely emulate the underlying hardware components and replicate their physical characteristics.

The hardware modules within the ASIC model reproduce the functional behavior of each component, enabling the accurate modeling of data flow through the ASIC. Those components are not only functional replications; they also simulate material variations. This feature significantly contributes to achieving a more comprehensive and realistic representation of the hardware, a crucial aspect in security applications where the robustness of the ASIC board's embedded Physical Unclonable Function (PUF) holds paramount importance. The basic element of the Photonic PUF is shown in Figure 10.



PUF	1	2	3	4
N_s	25	3	25	25
N_r	10	10	1	5

Figure 10: NEUROPULS Photonic PUF [55].

Components and Current State:

- **Photonic Integrated Circuit** (PIC):
  - **Optical Modulators**: To be implemented.
  - **MMI:** (Multimode Interference): Implemented frequency analysis using scattering parameters for characterization. The model also includes other parameters enabling the simulation of physical imperfections in the device, such as imbalances and losses. Incorporation of time domain analysis is planned.
  - **Photodetectors:** Developed to remove nonlinearities, currently computing the optical power of the input signal.
  - **Micro-ring Resonators:** Yet to be developed.
  - **Photonic Waveguides:** Implemented for frequency analysis. Incorporation of time domain analysis is planned.

Other Components to be Implemented:

- Electric PUF/ SRAM weak PUF: To be developed and studied
- TIA
- ADC
- DAC
- Power drivers
- GPIO

The development and incorporation of these components are essential for achieving a comprehensive ASIC model compatible with gem5-MARVEL. Further efforts will be directed towards implementing missing components, integrating time domain analysis, and exploring the functionality and performance of additional modules such as the Electric PUF, TIA, ADC, DAC, power drivers, and GPIO interface. These endeavors are crucial for advancing the simulation capabilities and ensuring the accuracy and reliability of the ASIC model within the gem5-Marvel framework.

5.3 Interfacing

NEUROPULS hinges on two core components: the ASIC/PUF and the RISC-V ecosystem. The crux of the matter lies in establishing a smooth data flow between these components, necessitating the creation of interfaces. These interfaces are split into Hardware and Software, with Hardware being physically integrated into the device and Software acting as the intermediary between the user/application and the Accelerator. Hardware interfaces are the tangible connections within the device, crucial for direct communication between the ASIC/PUF and the system.

- Hardware Interfaces: They are discussed in the security model primitives.
- Software interfaces:
    - **User Interface**: Currently analyzing how applications will be launched, either in a bare-metal environment or through an Operating System.
    - **PUF and the photonic accelerator Interface**: Planned to be achieved through memory mapping, wherein specific memory portions will be designated for the dedicated parts required for the PUF and the photonic accelerator.
    - **Direct Memory Access (DMA)**: Given that communication with the accelerator uses memory mapping, the role of DMA is essential for reading all configurations provided by the user and then transferring them to the Application-Specific Integrated Circuit (ASIC) without the intervention of the RISC-V microprocessor.
    - **Security Interfaces**: NEUROPULS aims to provide encryption mechanisms (e.g., AES), Key Management Systems (KMS), Hash generators (e.g., SHA-2), Message Authentication (e.g. HMAC), and other security-related services. This will complement the PUFs with a proper NEUROPULS Root-of-Trust (RoT), as depicted in Figure 11 At this stage, such RoT is the target architecture to support the protocols described later. On top of the actual PUFs modeling, they can be built as pure software components, e.g., C libraries exploiting the PUFs by DMA protocol access, or they can be further hardware blocks simulated in gem5-MARVEL.

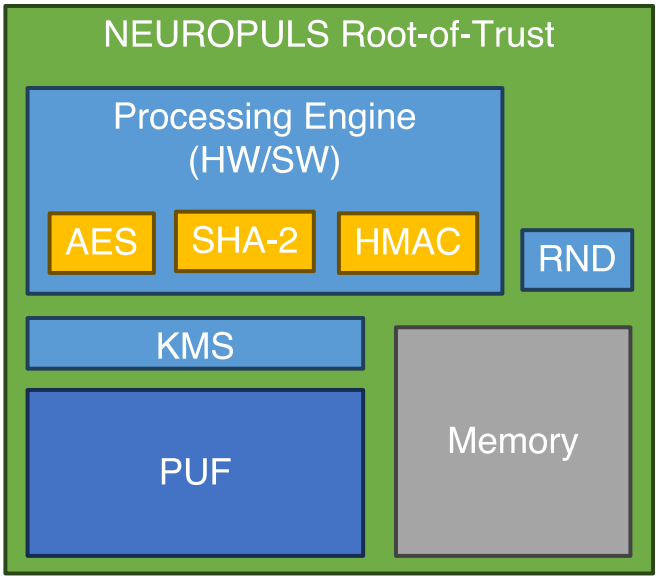


Figure 11: High-level preliminary NEUROPULS Root-of-Trust components.

5.4 Protocols for Mutual Authentication, Software Attestation and Encryption

5.4.1 Mutual authentication

Authentication is the first step in secure communication, which consists of verifying the identity of a participant (e.g., a device) before exchanging sensitive data with another participant. The context of NEUROPULS includes two main roles: the device to be verified and an external entity acting as the verifier. This context imposes two requirements: First, the authentication shall be mutual, i.e., considering that sensitive data travel in both directions, the device and the verifier should verify each other's identity. Second, the authentication process shall be lightweight because the resources on the device are constrained.

Existing authentication strategies based on PUFs require the verifier to store a large database of CRPs for each device, as first described in seminal works on PUFs [3], [4] and detailed by Suh et al. [5], or to exploit heavier protocols [6], [7]. However, to meet the lightweight requirement, we decided to adopt a different strategy, that is, HSC-IoT, the authentication procedure proposed by Hossain et al. [8]. Their idea is to use only a single CRP as a shared secret between the device and the verifier to support mutual authentication and to update it after each use with a fresh CRP.

Practically, the first CRP is shared at manufacturing time and is meant to support the first actual authentication session. At each actual authentication session, the device uses a fresh CRP that is based on the response of the previously used CRP. The new response is sent to the verifier in encrypted form and, if mutual authentication succeeds, the current CRP is updated on both the device and the verifier.

Figure 12 shows a UML sequence diagram showing messages exchanged between the device and the verifier according to the mutual authentication protocol.

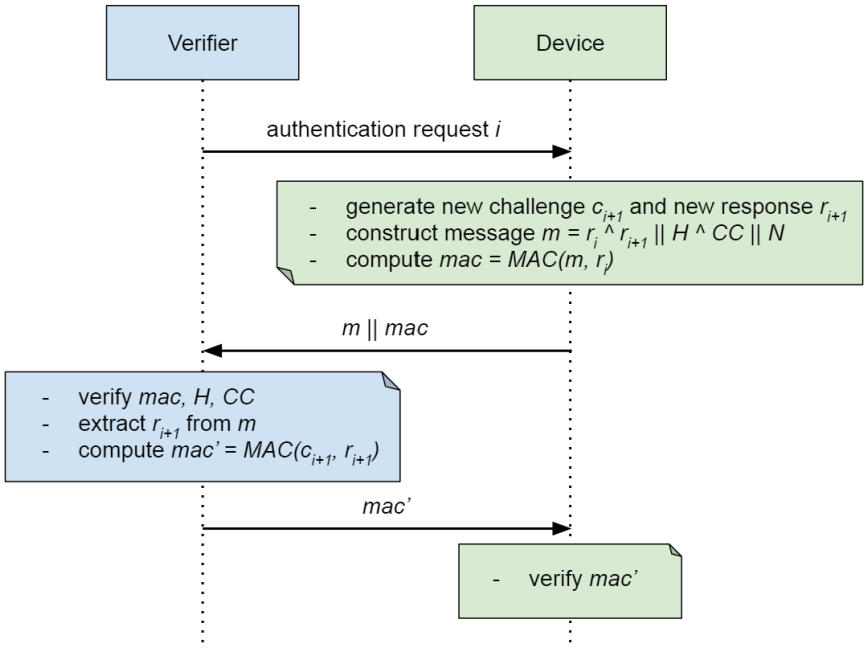


Figure 12: Mutual authentication protocol between the device and the verifier.

The protocol starts with the authentication request from the verifier. The device derives the new challenge c_{i+1} from the current response r_i, using it as a seed for a pseudo-random number generation (RNG) function known to both participants, c_{i+1} = RNG(r_i). The device computes a message m containing the new response r_{i+1}, XORed with the current response r_i. In the figure, the operator ^ denotes the XOR operation and || denotes concatenation. The message may also contain proof of the integrity of the software, such as the hash of the memory H and a clock count CC that represents the time needed to perform a given task. The message can contain a nonce N for freshness. This message m is then sent to the verifier along with its MAC signature, calculated using the MAC(data, key) function, whose first argument is the data to sign and the second is the key, r_i in this case. Now, the verifier can authenticate the device by checking the message signature using the secret r_i. The new response r_{i+1} is derived from m and stored in the verifier to be used as a shared secret in the next authentication session. Finally, the

verifier authenticates the device by demonstrating that it knows the new secret r_{i+1}, which is used to sign c_{i+1} (generated using r_{i+1}) through the MAC function again.

This protocol only needs one CRP to be known by the verifier at any point, which is more scalable than other solutions that require a large database of CRPs. In addition, CRPs are kept confidential because they are never exchanged in clear text. Although this protocol is very lightweight, it is designed to resist many attacks, as discussed by Hossain et al. [8].

5.4.2 Software attestation

Remote software attestation validates the integrity status of remote computing devices without relying on secure hardware components such as Trusted Platform Modules (TPMs). Such specialized components are often unsuitable for devices with limited resources [9], [10]. This approach enables remote systems to detect malicious or unintended changes in the firmware, software, or hardware that operates on these devices.

Attestation mechanisms generally send a hash of the device's memory to the verifier to prove that the device is not compromised [11]. An example of this is given in the previous section; during mutual authentication, the algorithm can include a hash of the memory, which provides some proof of the device's integrity. In this section, we describe a more powerful approach that, however, imposes stronger assumptions, i.e., it leverages an ideally reliable strong PUF and on a PUF model available to the verifier. To avoid attacks where a device hides its compromised memory regions from verification (e.g., by moving these regions around while the algorithm hashes the uncompromised memory), attestation protocols often employ temporal constraints that guarantee the unfeasibility of these attacks [12]. An important part of these protocols is the root of trust, a part of the system proven not to be compromised, on which the protocol can base its operations. Without being able to use secure hardware modules (i.e., TPMs) as a root of trust, many modern protocols have started adopting PUFs as an alternative. In our framework, we leverage the photonic PUF (pPUF) embedded in the neuromorphic accelerator to generate many CRPs that can then be used to hash different areas of the device's memory.

The verifier starts the attestation by crafting a message, the attestation request, that contains a timestamp t and a challenge c_1. The message is then sent to the device, which then promptly starts the attestation process by using the issued challenge to compute a response r_1 in the pPUF. The response is combined with the timestamp as the seed for an RNG that generates the random walk-in memory: $m_1,...,m_n = RNG(r_1 + t)$. A secure hash algorithm is then used with the initial chunk of memory m_1 on the device and r_1, generating the first hash $h_1 = HASH(m_1, r_1)$, while r_1 is used simultaneously as the next challenge for pPUF, as $r_1 = pPUF(r_1)$. All subsequent hashes depend on the previously calculated ones: $h_{i+1} = HASH(m_{i+1}, r_{i+1}, h_i)$. The inherent speed of the pPUF (at least 5 Gb/s) guarantees that the constant challenge and response generation never slows down the protocol, so the temporal constraints of our approach can be stricter than those found in previous work. After exhausting all memory regions, the final hash, h_n, is sent to the

verifier. This protocol minimizes the network load by having a very small footprint in both the attestation initiation and its finalization, which allows our temporal constraints to be mostly focused on the speed of the iterative hash function. The verifier has a copy of the uncompromised device memory and a model of the pPUF, so it can start to calculate h_n right after generating the attestation request. After receiving h_n from the device, the verifier checks that its value is correct and that the attestation did not exceed its temporal constraints; if both requirements are satisfied, the attestation is successful. Otherwise, a new request is issued, and the protocol restarts with a new timestamp and challenge.

5.4.3 Neural network configuration and data encryption

Another important security requirement is the confidentiality of the actual data processed by the accelerator and the confidentiality of the neural network configuration. To meet this confidentiality requirement, encryption encodes the data in all communications with external parties and all the software running on the device. The NN configuration, the input data to the device, and the computation output are encrypted using the secret keys described in Section II. This key is never exposed to the software layer, but encryption and decryption occur on the hardware in the implementation of the security primitives shown here:

Function name	*Parameters*	*Results*
load_network	ciphered_network	
execute_network	ciphered_input	ciphered_output

load_network receives the neural network configuration in encrypted form. The configuration is decrypted in hardware and loaded in the accelerator. execute_network takes the input as a decrypted parameter and fed to the accelerator. Then, the computation result is encrypted and returned by this function. As specified by the function signatures, data are never exposed in plaintext to the software. Data are decrypted internally by the device using primitives that never leave plaintext in the memory after execution, thus preserving confidentiality even against an internal attacker capable of reading the RAM.

6. Conclusion: Towards Future Innovations

The NEUROPULS full system simulation infrastructure for heterogeneous electronic/photonic systems provides a dynamic environment for the project partners, and generally for researchers and engineers, to iterate through various design iterations rapidly. By offering a simulation platform based on state-of-the-art microarchitecture-level simulators, such as gem5, for simulating complete computing systems, including neuromorphic accelerators and security primitives, the NEUROPULS infrastructure will become a playground for future innovation. NEUROPULS partners and researchers can experiment with novel architectures, algorithms, and configurations.

In the realm of heterogeneous computing systems, in which diverse components need to seamlessly interact, the NEUROPULS simulation infrastructure aims to make informed decisions at the system level. It allows for the exploration of intricate design spaces, facilitating a more in-depth understanding of how different components may impact the overall system performance and security. This knowledge is invaluable for architects and engineers seeking to optimize systems for specific use cases or performance criteria.

7. References

[1] M. Le Gallo, A. Sebastian, "An overview of phase-change memory device physics," Journal of Physics D: Applied Physics, Vol. 53, Issue 21, March 2020, DOI: 10.1088/1361-6463/ab7794.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood., "The gem5 simulator," SIGARCH Comput. Archit. News 39, 2 (May 2011), 1–7, DOI: 10.1145/2024716.2024718.

[3] *Physical One-Way Functions.* Pappu, Ravikanth, et al. 2002, Science, Vol. 297, p. 2026-2030.

[4] *Silicon physical random functions.* Gassend, Blaise, et al. [edited by] Vijayalakshmi Atluri. s.l. : ACM, 2002. Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002. p. 148–160.

[5] *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* Suh, G. Edward and Devadas, Srinivas. s.l. : IEEE, 2007. Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, 2007. p. 9–14.

[6] *HRP: A HMAC-based RFID mutual authentication protocol using PUF.* Jung, Seung Wook and Jung, Souhwan. s.l. : IEEE Computer Society, 2013. The International Conference on Information Networking 2013, ICOIN 2013, Bangkok, Thailand, January 28-30, 2013. p. 578–582.

[7] *Mutual authentication in wireless body sensor networks (WBSN) based on Physical Unclonable Function (PUF).* Lee, Young-Sil, Lee, Hoon-Jae and Alasaarela, Esko. [edited by] Roberto Saracco, et al. s.l. : IEEE, 2013. 2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013, Sardinia, Italy, July 1-5, 2013. p. 1314–1318.

[8] *HSC-IoT: A Hardware and Software Co-Verification Based Authentication Scheme for Internet of Things.* Hossain, Md. Mahmud, Noor, Shahid Al and Hasan, Ragib. s.l. : IEEE Computer Society, 2017. 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2017, San Francisco, CA, USA, April 6-8, 2017. p. 109–116.

[9] *FPGA-Based Remote-Code Integrity Verification of Programs in Distributed Embedded Systems.* Basile, Cataldo, Di Carlo, Stefano and Scionti, Alberto. 2012, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 42, p. 187-200.

[10] *HAtt: Hybrid remote attestation for the Internet of Things with high availability.* Aman, Muhammad Naveed, et al. s.l. : IEEE, 2020, IEEE Internet of Things Journal, Vol. 7, p. 7220–7233.

[11] *AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS.* Feng, Wei, et al. s.l. : Elsevier, 2018, Computer Networks, Vol. 134, p. 167–182.

[12] *ATT-Auth: A hybrid protocol for industrial IoT attestation with authentication.* Aman, Muhammad Naveed and Sikdar, Biplab. s.l. : IEEE, 2018, IEEE Internet of Things Journal, Vol. 5, p. 5119–5131.

[13] *J.* Lowe-Power, et al., The gem5 Simulator: Version 20.0+, arXiv, 2020, https://arxiv.org/abs/2007.03152.
[14] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems," in IEEE Micro, vol. 26, no. 4, pp. 52-60, July-Aug. 2006, doi: 10.1109/MM.2006.82.
[15] A. Limaye and T. Adegbija, "A Workload Characterization of the SPEC CPU2017 Benchmark Suite," 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Belfast, UK, 2018, pp. 149-158, doi: 10.1109/ISPASS.2018.00028.
[16] A. Sandberg, S. Diestelhorst, W. Wang, "Architectural Exploration with gem5", ASPLOS 2017, https://www.gem5.org/assets/files/ASPLOS2017_gem5_tutorial.pdf
[17] P. Rosenfeld, E. Cooper-Balis and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," in IEEE Computer Architecture Letters, vol. 10, no. 1, pp. 16-19, Jan.-June 2011, doi: 10.1109/L-CA.2011.4.
[18] P. Karunamurthy, S. Alhady, A. Wahab, W. Othman, "Integration of gem5 and DramSim2 for DDR4 Simulation," International Journal of Advanced Trends in Computer Science and Engineering, 2020, doi: 10.30534/ijatcse/2020/ 99912020.
[19] T. E. Carlson, W. Heirman and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, Seattle, WA, USA, 2011, pp. 1-12, doi: 10.1145/2063384.2063454.
[20] A. Akram and L. Sawalha, "×86 computer architecture simulators: A comparative study," 2016 IEEE 34th International Conference on Computer Design (ICCD), Scottsdale, AZ, USA, 2016, pp. 638-645, doi: 10.1109/ICCD.2016.7753351.
[21] C. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," In Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation (PLDI '05), 2005, doi: 10.1145/1064978.1065034.
[22] A. Sabu, H. Patil, W. Heirman and T. E. Carlson, "LoopPoint: Checkpoint-driven Sampled Simulation for Multi-threaded Applications," 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, Republic of, 2022, pp. 604-618, doi: 10.1109/HPCA53966.2022.00051.
[23] J. Power, J. Hestness, M. S. Orr, M. D. Hill and D. A. Wood, "gem5-gpu: A Heterogeneous CPU-GPU Simulator," in IEEE Computer Architecture Letters, vol. 14, no. 1, pp. 34-36, 1 Jan.-June 2015, doi: 10.1109/LCA.2014.2299539.
[24] A. Jain, M. Khairy, and T. G. Rogers, "A quantitative evaluation of contemporary gpu simulation methodology," Proceedings of the ACM on Measurement and Analysis of Computing Systems - SIGMETRICS, vol. 2, no. 2, p. 35, 2018, doi: 10.1145/3224430.
[25] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," Commun. ACM, vol. 62, no. 2, p. 48–60, jan 2019. [Online]. Available: https://doi.org/10.1145/3282307.
[26] J. Cong, V. Sarkar, G. Reinman, and A. Bui, "Customizable domain-specific computing," IEEE Design & Test of Computers, vol. 28, no. 2, pp. 6–15, 2011.

[27] Z. Jia, M. Maggioni, J. Smith, and D. P. Scarpazza, "Dissecting the nvidia turing t4 gpu via microbenchmarking," 2019. [Online]. Available: https://arxiv.org/abs/1903.07486.
[28] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16), 2016, p. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30.
[29] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomicsco-processor provides up to 15,000x acceleration on long read assembly," in Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18), 2018, p. 199–213. [Online]. Available: https://doi.org/10.1145/3173162.3173193.
[30] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: Balancing efficiency & flexibility in specialized computing," in Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13), 2013, p. 24–35. [Online]. Available: https://doi.org/10.1145/2485922.2485925.
[31] P. Agrawal and W. Dally, "A hardware logic simulation system," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 9, no. 1, pp. 19–29, 1990.
[32] G. Moore, "Cramming more components onto integrated circuits," Proceedings of the IEEE, vol. 86, no. 1, pp. 82–85, 1998.
[33] O. Chatzopoulos, G. Papadimitriou, V. Karakostas, and D. Gizopoulos, "gem5-MARVEL: Microarchitecture-Level Resilience Analysis of Heterogeneous SoC Architectures," in IEEE International Symposium on High-Performance Computer Architecture (HPCA 2024), 2024.
[34] https://cloud.google.com/tpu/docs/intro-to-tpu
[35] J. Huang, Accelerating AI with GPUs: A New Computing Model, 2016, https://blogs.nvidia.com/blog/accelerating-ai-artificial-intelligence-gpus/
[36] Intel, leverage a Source-Accessible Framework to Ease Custom FPGA-based Acceleration Platform Development, https://www.intel.com/content/dam/www/central-libraries/us/en/documents/open-fpga-stack-product-brief.pdf
[37] J. Hayes, Optical processors light the path to warp factor computing, 2023, https://eandt.theiet.org/2021/09/17/optical-processors-light-path-warp-factor-computing
[38] K. Wiggers, LightOn researchers explain how they trained an AI model on an optical co-processor, 2020, https://venturebeat.com/ai/lighton-researchers-explain-how-they-trained-an-ai-model-on-an-optical-co-processor/
[39] Y. S. Shao, S. L. Xi, V. Srinivasan, G.-Y. Wei, and D. Brooks, "Co-designing accelerators and soc interfaces using gem5-aladdin," in 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016, pp. 1–12.
[40] J. Cong, Z. Fang, M. Gill, and G. Reinman, "Parade: A cycle-accurate full-system simulation platform for accelerator-rich architectural design and exploration," in 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015, pp. 380–387.

[41] C. Menard, J. Castrillon, M. Jung, and N. Wehn, “System simulation with gem5 and systemc: The keystone for full interoperability,” in 2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2017, pp. 62–69.
[42] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, “Demystifying Soft Error Assessment Strategies on ARM CPUs: Microarchitectural Fault Injection vs. Neutron Beam Experiments,” in 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), June 2019, pp. 26–38, doi: 10.1109/DSN.2019.00018.
[43] P. R. Bodmann, G. Papadimitriou, R. L. R. Junior, D. Gizopoulos, and P. Rech, “Soft Error Effects on Arm Microprocessors: Early Estimations versus Chip Measurements,” IEEE Transactions on Computers, vol. 71, no. 10, pp. 2358–2369, 2022, doi: 10.1109/TC.2021.3128501.
[44] H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham, and S. Mitra, “Quantitative evaluation of soft error injection techniques for robust system design,” in Proceedings of the 50th Annual Design Automation Conference (DAC ’13), 2013, doi: 10.1145/2463209.2488859.
[45] “gem5 GitHub Repository,” https://github.com/gem5/gem5, accessed: 2024-02-10.
[46] S. Rogers, J. Slycord, M. Baharani, and H. Tabkhi, “gem5-salam: A system architecture for llvm-based accelerator modeling,” in 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 471–482.
[47] S. Rogers, J. Slycord, R. Raheja, and H. Tabkhi, “Scalable llvm-based accelerator modeling in gem5,” IEEE Computer Architecture Letters, vol. 18, no. 1, pp. 18–21, 2019.
[48] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” in Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538), 2001, pp. 3–14,. doi: 10.1109/WWC.2001.990739.
[49] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, “Machsuite: Benchmarks for accelerator design and customized architectures,” in 2014 IEEE International Symposium on Workload Characterization (IISWC), 2014, pp. 110–119.
[50] V. Sridharan and D. R. Kaeli, “Using Hardware Vulnerability Factors to Enhance AVF Analysis,” in Proceedings of the 37th Annual International Symposium on Computer Architecture, ser. ISCA ’10. New York, NY, USA: ACM, 2010, pp. 461–472, doi: 10.1145/1815961.1816023.
[51] G. Papadimitriou and D. Gizopoulos, “Anatomy of On-Chip Memory Hardware Fault Effects Across the Layers,” IEEE Transactions on Emerging Topics in Computing, pp. 1–12, 2022. [Online]. Available: https://doi.org/10.1109/TETC.2022.3205808.
[52] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, “Statistical fault injection: Quantified error and confidence,” in 2009 Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 502–506.
[53] F. Pavanello et al., "NEUROPULS: NEUROmorphic energy-efficient secure accelerators based on Phase change materials aUgmented siLicon photonicS," 2023 IEEE European Test Symposium (ETS), Venezia, Italy, 2023, pp. 1-6, doi: 10.1109/ETS56758.2023.10173974.

[54] M. Poremba, T. Zhang, Y. Xie, “NVMain 2.0: A User-friendly Memory Simulator to Model (Non-)Volatile Memory Systems”, IEEE Computer Architecture Letters Volume 14 Issue 2 July-Dec. 2015 , DOI 10.1109
[55] Jiménez, P., Cardoso, R., de Queiroz, M.G., Abdalla, M., Zrounba, C., Rührmair, U., Marchand, C., Letartre, X., & Pavanello, F. (2023). Photonic Physical Unclonable Function Based on Symmetric Microring Resonator Arrays. Frontiers in Optics + Laser Science 2023 (FiO, LS).
[56] Milad Mohseni, Ahmad Habibized Novin, A survey on techniques for improving Phase Change Memory (PCM) lifetime, Journal of Systems Architecture, Volume 144, 2023, 103008, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2023.103008.
[57] J. Feldmann et al., “Parallel convolutional processing using an integrated photonic tensor core,” Nature, vol. 589, no. 7840, pp. 52–58, Jan. 2021, doi: 10.1038/s41586-020-03070-1.
[58] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, “All-optical spiking neurosynaptic networks with self-learning capabilities,” Nature, vol. 569, no. 7755, pp. 208–214, May 2019, doi: 10.1038/s41586-019-1157-8.
[59] C. Ríos et al., “Integrated all-photonic non-volatile multi-level memory,” Nature Photon, vol. 9, no. 11, pp. 725–732, Nov. 2015, doi: 10.1038/nphoton.2015.182.