Ref. Ares(2024)6898123 - 30/09/2024

NEUROPULS

Deliverable 5.1

Accelerators models and components: 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	D5.1 Accelerators models and components: 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	NEUROPULS_D5.1_NKUA
Keywords	Simulation, gem5, accelerators modeling, neural networks

		Document History
2024-07-15	Table of contents, structure defined, abstract and introduction	P09 NKUA – George Papadimitriou
2024-07-30	Background, AI accelerators modeling (Sections 2, 3)	P09 NKUA – George Papadimitriou, Odysseas Chatzopoulos
2024-09-12	Accelerator Design Space Exploration (Sections 4)	P09 NKUA – George Papadimitriou, Odysseas Chatzopoulos

Document History

2024-09-27	Internal Review	Mikel Fernandez, Alessandro Savino, Fabio Pavanello
2024-09-27	Final improvements/edits and quality review	George Papadimitriou, Vasileios Karakostas, Dimitris Gizopoulos

Document Validation

Project Coordinator	P1 CNRS – Fabio Pavanello Fabio.pavanello@cnrs.fr
Date	2024-09-30

This document contains information which is proprietary to the NEUROPULS consortium. The document or the content of it shall not be communicated by any means to any third party except with prior written approval of the NEUROPULS consortium.

Document Abstract

This report presents the progress and the development of several AI (Artificial Intelligence) accelerators, which are built and are evaluated on the robust and efficient system architecture modeling and simulation infrastructure of WP5. We present the comprehensive evaluation of complete computing systems integrating AI and accelerators of the NEUROPULS Horizon Europe project (Grant Agreement n° 101070238) along with RISC-V CPUs. The rise of artificial intelligence (AI) and neuromorphic computing has necessitated the development of specialized hardware accelerators capable of meeting the unique computational demands of these domains. In this context, this deliverable presents a comprehensive overview of our efforts to design, model, and simulate AI accelerators using the gem5 simulation infrastructure. The next iteration of this deliverable (i.e., the deliverable D5.2) will augment this deliverable more AI accelerators modeling and design space exploration, along with neuromorphic accelerators and the NEUROPULS neuromorphic accelerator modeling. This document details our methodologies and experimental results, focusing on performance, area, and reliability assessments of a famous AI accelerator (LeNet5) integrated with RISC-V CPUs. We highlight the implementation and simulation of prominent neural network models, specifically MobileNetV2 and LeNet-5, within our infrastructure.

This report details the diverse objectives, including the execution of accelerators in a full system setup, the exploration of the design space of AI accelerators and the assessment of their critical design parameters such as performance, area, and resilience.

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

Table of contents

1. Introduction

The rise of artificial intelligence (AI) and neuromorphic computing has needed the development of specialized hardware accelerators capable of meeting the unique computational demands of these domains. In this context, this deliverable presents a comprehensive overview of our efforts to design, model, and simulate AI accelerators using the gem5 simulation infrastructure. The detailed gem5-based simulation infrastructure has been presented in the earlier deliverable of the NEUROPULS project, i.e., D5.9. Therefore, this deliverable details our methodologies, evaluations and findings, focusing on performance, reliability, and area estimations of AI accelerators integrated with RISC-V CPUs, which has been developed in this gem5-based simulation infrastructure. Additionally, we highlight the implementation and simulation of prominent neural network (NN) models, specifically MobileNetV2 and LeNet-5, within our infrastructure.

The gem5 simulation framework [2] serves as the foundation for our exploration of AI accelerators. gem5 allows detailed architectural simulations, enabling precise, cycle-level estimations. By integrating AI accelerators with RISC-V CPUs within gem5, we simulate real-world scenarios to evaluate the efficiency and scalability of our designs. Our focus in this iteration of the deliverable “Accelerators models and components: Iteration 1” on AI accelerators includes the design and simulation of components specifically tailored for NN workloads. Neuromorphic accelerators, inspired by the human brain's architecture, offer promising solutions for energy-efficient AI computation, which will be extensively presented in the next iteration (i.e., D5.2).

We model AI accelerators' behaviors, including data flow, memory access patterns, and computational throughput, to understand their operational dynamics. A key aspect of our work involves the rigorous evaluation of a well-established accelerator design, the Lenet-5. Using gem5, we perform detailed simulations to estimate the performance, power consumption, and silicon area of various configurations. These estimations are crucial for optimizing hardware designs to meet the stringent requirements of AI applications.

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

a) **Full System AI Accelerators modeling:** 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 electronic and 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 eases 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 advancements currently available in the literature. By integrating electronic and 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 aims 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 set up 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.

Specifically, deliverable D5.1 focuses on the Accelerator Modeling and Components analysis, along with the evaluation and the design space exploration of cutting-edge electronic AI (Artificial Intelligence) accelerators on the gem5 simulation infrastructure of WP5.

2. Background & Related Work

2.1 Domain-Specific Accelerators

With Moore's Law slowing down [16], domain-specific accelerators (DSAs) or AI (Artificial Intelligence) accelerators have become increasingly important due to their superior performance and efficiency on specific tasks compared to general-purpose CPUs [17]. Accelerators are specialized hardware engines designed for specific domains, such as graphics [18], deep learning [19], bioinformatics [20], image processing [21], and simulation [22]. 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 [23]. 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 [24].

In the realm of electronic accelerators, these solutions are commonly realized using 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 speed up 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.

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.

Modeling AI accelerators in the gem5 simulator alongside RISC-V CPUs is crucial for several reasons. The gem5 simulator is a highly flexible simulator platform that allows for detailed and accurate simulation of complex computing systems. Integrating AI accelerators with RISC-V CPUs within gem5 will enable us to explore the interaction between general-purpose and specialized processors in a controlled environment. This integration is vital for optimizing the performance and energy efficiency of heterogeneous systems (i.e., such as the one considered in the NEUROPULS project). By simulating these systems, we can conduct extensive design space exploration, identifying the most effective configurations for different conditions.

Design space exploration of cutting-edge Convolutional Neural Networks (CNNs) like MobileNetV2 and LeNet-5 is particularly important. These CNNs have different architectural characteristics and computational requirements, providing diverse case

studies for evaluating the reliability, performance and efficiency of AI accelerators. By using gem5 to simulate these CNNs on systems with integrated AI accelerators and RISC-V CPUs, we can gain insights into the optimal design and configuration of hardware for deep learning tasks. This includes understanding how different accelerator designs affect the performance of CNNs, the trade-offs between various architectural choices, and the potential benefits of specific optimizations. These results will empower the following modeling and implementation aspects about photonic accelerators in the gem5 simulation.

We believe that the integration of AI accelerators with RISC-V CPUs in the gem5 simulator, as will be presented in this deliverable, is essential for advancing our understanding of heterogeneous computing systems. It allows for comprehensive design space exploration of AI workloads, leading to more efficient and effective hardware designs that can meet the growing demands of modern computing applications.

2.2 AI Accelerators

Our focus on AI accelerators includes the design and simulation of components specifically tailored for NN workloads. Neuromorphic accelerators, inspired by the human brain's architecture, offer promising solutions for energy-efficient AI computation. In this deliverable, we present the modeling of electronic NN accelerators, including data flow, memory access patterns, and computational throughput, to understand their operational dynamics.

To prove the practical applicability of our simulation infrastructure, and to perform design space exploration, we model and evaluate in our gem5-based infrastructure two well-known NN architectures: MobileNetV2 and LeNet-5.

MobileNetV2 [37]: This model is optimized for mobile and embedded vision applications. It employs depth-wise separable convolutions to reduce computational complexity while keeping high accuracy. We simulate MobileNetV2 within gem5, by extending the architecture of [3] to evaluate its performance on our AI accelerator models.

LeNet-5 [38]: A pioneering CNN architecture, LeNet-5 is often used for image classification tasks. By modeling LeNet-5, we provide insights into how traditional NN designs perform on modern AI accelerators.

2.2.1 MobileNetV2

MobileNetV2 represents a significant improvement over its predecessor, MobileNetV1, offering enhanced capabilities for mobile visual recognition tasks, including image classification, object detection, and semantic segmentation. This model is part of the TensorFlow-Slim Image Classification Library [39].

Key Improvements and Features of MobileNetV2

Depth-wise Separable Convolutions

MobileNetV2 continues to use depth-wise separable convolutions, an efficient building block introduced in MobileNetV1. This technique significantly reduces the computational cost and model size by separating the spatial and channel-wise convolutions.

Linear Bottlenecks

A major innovation in MobileNetV2 is the introduction of linear bottlenecks between layers. These bottlenecks capture the model's intermediate inputs and outputs, easing efficient transformations from low-level pixel data to high-level image categories. This structure helps in keeping the compactness of the model while enabling it to learn complex features.

Inverted Residuals and Shortcut Connections

MobileNetV2 introduces inverted residuals with linear bottlenecks. The inverted residual block expands the input before applying depth-wise convolutions and then projects it back to a lower dimension, helping to keep essential information while reducing computation. The shortcut connections between bottlenecks are like residual connections in traditional NNs, which aid in faster training and improved accuracy by allowing gradients to flow more smoothly through the network.

Regarding the technical advantages, MobileNetV2 offers:

Efficiency and Speed

MobileNetV2 models are designed to be faster and more efficient than MobileNetV1. They achieve the same accuracy with only half the number of operations and require 30% fewer parameters. This efficiency translates to a significant speed boost, making the models 30-40% faster.

Improved Accuracy

Despite the reduction in computational cost and model size, MobileNetV2 achieves higher accuracy compared to MobileNetV1. This improvement is due to the more sophisticated architecture that better captures and processes image features.

MobileNetV2 sets a new standard for efficient and accurate mobile visual recognition. By building on the concepts of MobileNetV1 and introducing innovative features like linear bottlenecks and inverted residuals, it achieves superior performance while maintaining a compact and efficient design. This makes MobileNetV2 an ideal choice for deploying advanced machine learning models on edge devices, where computational resources and power consumption are critical constraints [37] [39], and thus an adequate candidate for design space exploration studies in NEUROPULS project.

2.2.2 LeNet-5

LeNet-5 is one of the first CNNs, developed by Yann LeCun and his team in 1998. The details are in their paper "Gradient-Based Learning Applied to Document Recognition," where they used LeNet5 to recognize handwritten digits [38].

As shown in Figure 1 below, LeNet-5 has a five-layer architecture, including two convolutional layers and three fully connected layers. It processes 32x32 grayscale images, meaning it isn't designed for multichannel RGB images.

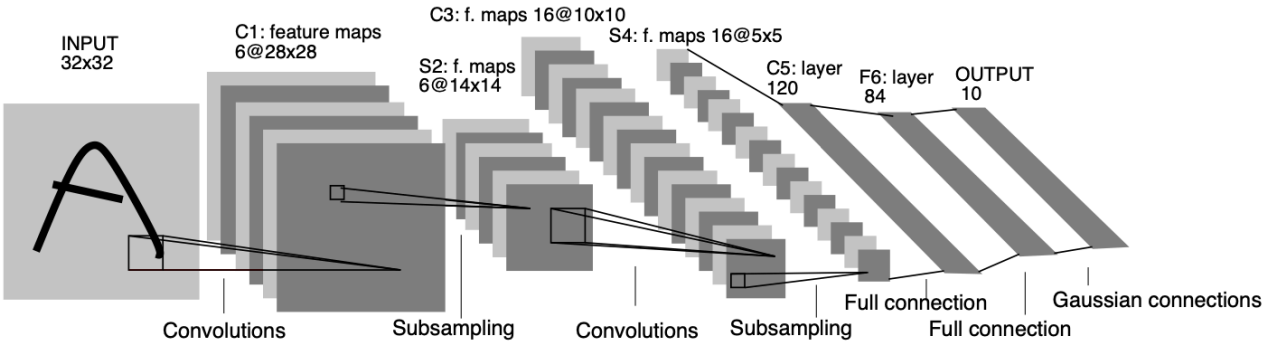

Figure 1. LeNet5 architecture, a Convolutional Neural Network (CNN). (Image Source [38]).

The input layer is followed by a convolutional layer with 6 filters of 5x5 size, reducing the image dimensions and increasing the depth to 28x28x6. This is followed by a pooling layer that halves the size to 14x14x6. Another convolutional layer with 16 filters of 5x5 size is applied, followed by another pooling layer, resulting in a 5x5x16 feature map.

Next, a convolutional layer with 120 filters of 5x5 size flattens the feature map to 120 values. This is followed by a fully connected layer with 84 neurons. The final layer, the output layer, has 10 neurons, corresponding to the 10-digit classes in the MNIST dataset, as shown in Figure 2.

Data Loading

We use the MNIST dataset, which includes 28x28 grayscale images of handwritten digits. It contains 60,000 training images and 10,000 testing images.

You can see some of the samples of images below:

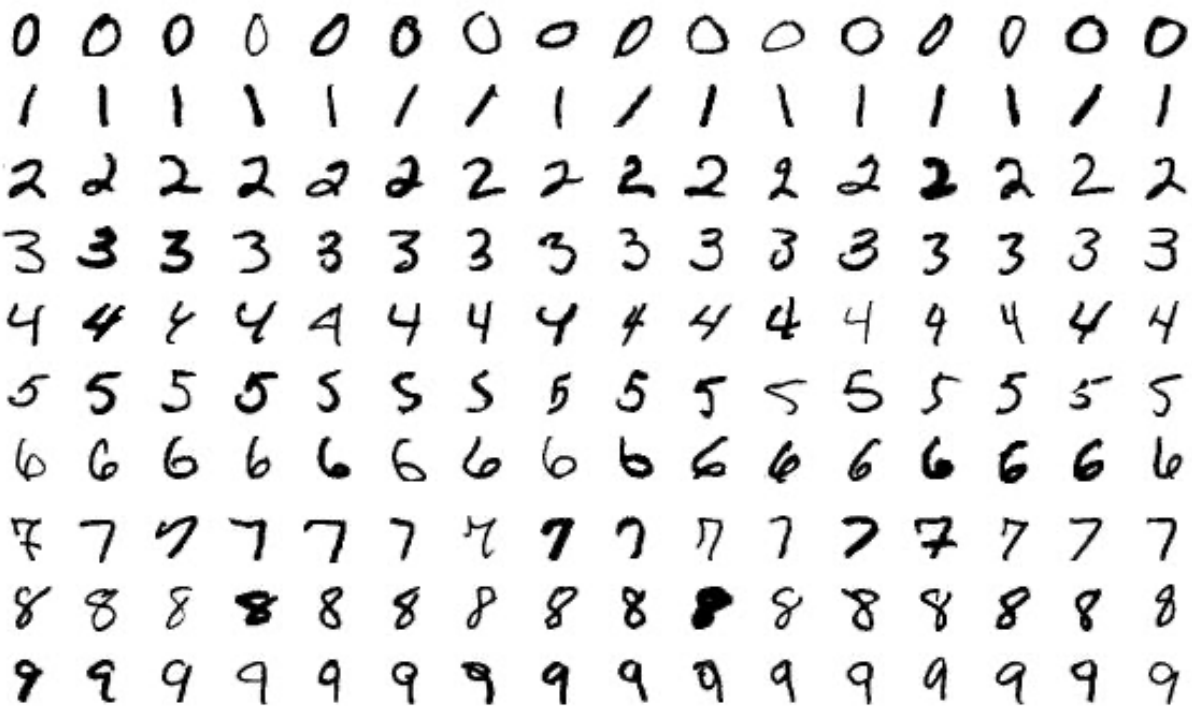

Figure 2. The MNIST database (Modified National Institute of Standards and Technology database) – Image Source from [40].

2.3 The gem5 Simulator

WP5 aims to build a comprehensive system architecture modeling and simulation infrastructure that incorporates neuromorphic accelerators and hardware primitives. To achieve this goal, the state-of-the-art computing systems simulator, gem5, will serve as the backbone for the entire WP5. gem5 [2] 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.

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 [4]. 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 [5], or other popular suites such as MiBench [34] 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 [6].

2. **DRAMSim2 and DRAMSim3**:

   DRAMSim2 [7] and DRAMSim3 [8] is a memory system simulator that focuses specifically on modeling dynamic random-access memory (DRAM) behavior. Researchers integrate them with gem5 [9] to study memory subsystem performance, investigate memory hierarchy designs, and analyze the impact of different DRAM architectures on overall system performance.

3. **Sniper**:

   Sniper [10] 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 [11].

4. **Pin**:

   Pin, developed by Intel [12], 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 [13]. This pairing enables a thorough evaluation of behaviors at both the application and system levels.

5. **gem5-gpu**:

   gem5-gpu [14] 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 [15]. 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.

3. Simulation Infrastructure for AI and Neuromorphic Accelerators Models

This section delves into the heart of the simulation infrastructure — the system architecture simulation framework, which is called gem5-MARVEL [24]. 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

In this section, we present a summary of our gem5-based modeling implementation. Recently, several efforts have been made to integrate DSA models with the state-of-the-art gem5 simulation environment. These efforts include, among others, gem5-Aladdin [25] and PARADE [26]. Additionally, SystemC support was added to gem5 [27] 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 [25], [26] or RTL-based solutions [27] (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 [28]-[30]. To this end, gem5-MARVEL relies on microarchitecture-level simulation using the latest version of the gem5 simulator [2], [31], 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.

gem5-MARVEL is an extension of gem5-SALAM [32] that uses an advanced dynamic graph execution engine based on LLVM [33]. 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.

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

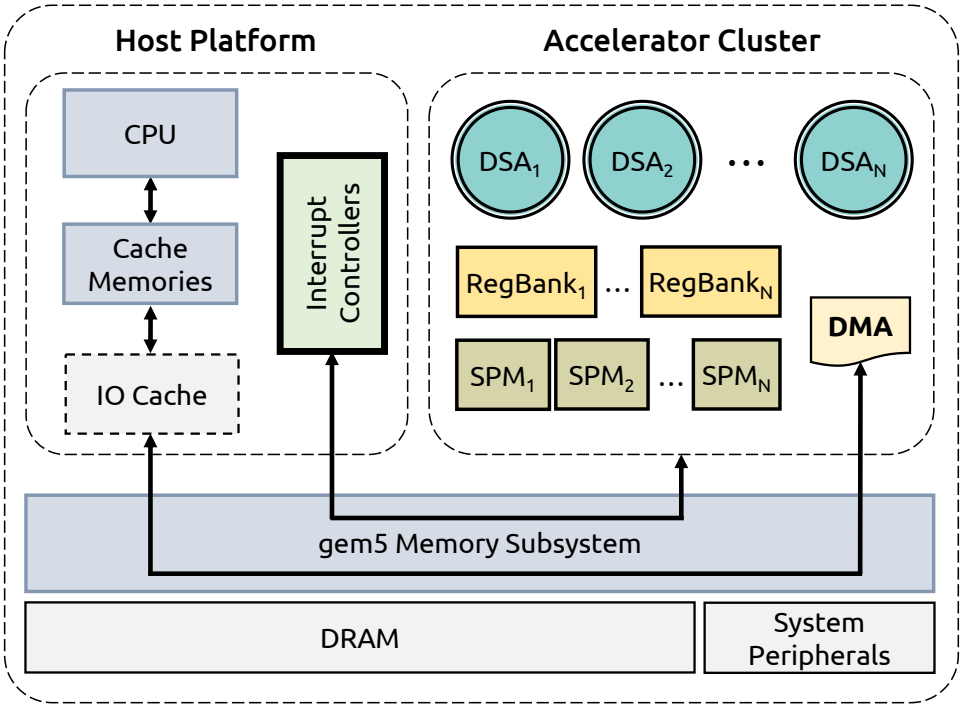

Figure 3: gem5-based SoC architecture and interconnection.

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.

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 4, 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 responsible for producing these fault mask files for a fault injection campaign ❶. Running scripts that serve as a campaign controller and are responsible for 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 single or multiple faults during system simulation, each fault injection simulation requires a fault mask input file e 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

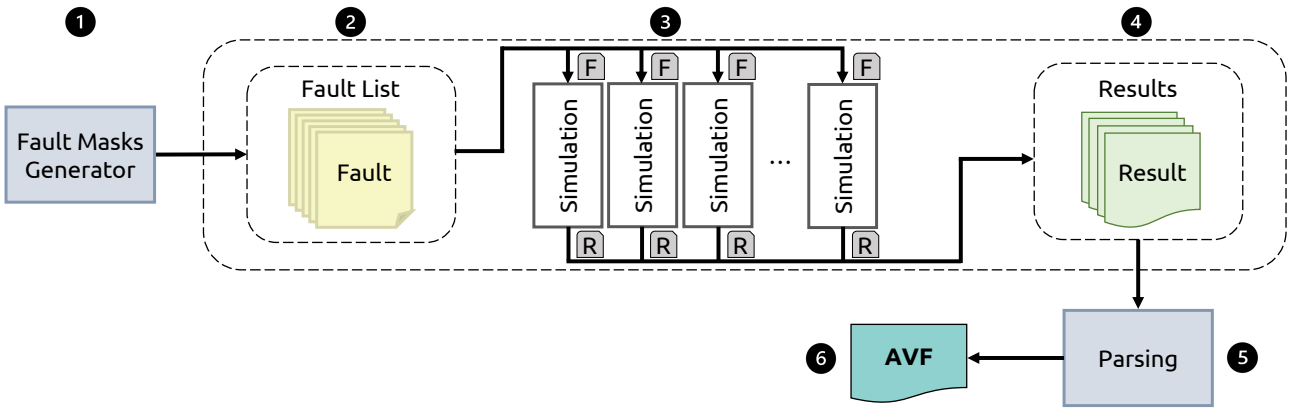

Figure 4. Fault injection campaign high-level layout with parallel workers.

determine what the result of the injected fault on this simulation was. Finally, the results are parsed ❺. Figure 4 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 (Architectural Vulnerability Factor) or HVF (Hardware Vulnerability Factor) 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 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 [35]. A hardware-visible fault is exposed to the user program once it reaches a software (or architecture visible) resource [36].

gem5-MARVEL employs two vulnerability evaluation methodologies of different layers: the HVF assessment [35] and the AVF assessment, providing the partial microarchitecture-dependent vulnerability and the full cross-layer vulnerability, respectively. As shown in Figure 5(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

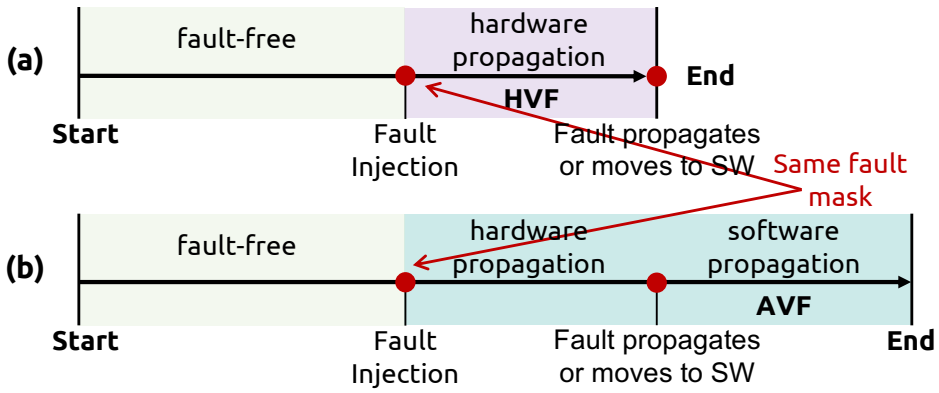

Figure 5. (a) HVF evaluation process; (b) AVF evaluation process, which may also consider the HVF evaluation.

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 5(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.

4. Accelerator Designs and Design Space Exploration

While the NEUROPULS accelerators are still in development and not yet integrated into gem5, we have successfully implemented well-established neural network accelerators. This provides valuable performance insights and sets the foundation for future comparisons and design space exploration with the final NEUROPULS accelerator model. In this section, we conduct an extensive exploration of the design space for LeNet-5, thoroughly analyzing various architectural configurations and their performance implications. Additionally, we detail the modeling steps taken to successfully implement MobileNetv2, explaining the considerations made in adapting its structure to specific design goals.

4.1 LeNet-5 Design Points

Leveraging the capabilities of our advanced accelerator modeling infrastructure, we thoroughly examine **two distinct topologies** of an accelerator implementing the **LeNet-5 CNN**, extending the architecture of [3] and exploring **three different levels of parallelism**. In the following subsections, we provide an in-depth analysis of the various design points we considered, and delve into the trade-offs between performance, area, and reliability assessment. This detailed exploration allows us to uncover the strengths and weaknesses of each approach, offering valuable insights into the impact of parallelism on system efficiency and dependability.

We investigate two distinct hardware topologies, referred to as **Naive** and **Massive**, which demonstrate how designers can quickly explore different architectures using our infrastructure. These topologies serve as examples to showcase the flexibility and efficiency of our system in evaluating various design choices. For these topologies, we have identified six key internal parameters for exploration: output height, output width, kernel height, kernel width, and the input/output channel depth. These kernel and channel depth parameters play a crucial role in defining the internal loop structure of the system. By adjusting these loops, we can unroll them to enhance datapath parallelism. This flexibility allows us to develop three hardware variants that we benchmark on each architecture.

The first hardware variant, which we label as "**No Unroll**," does not feature any loop unrolling. It relies solely on temporal compute parallelism within the accelerator. The second variant, termed "**Input Unroll**," fully unrolls the kernel's height, width, and input channel dimensions for each layer of the network. For instance, in the case of Conv1, which has a 5x5x6 convolutional window, the input window is fully unrolled, resulting in

a parallelism factor of 150. This unrolling process ensures that the feature map and weight scratchpad porting are aligned with the unroll factor.

The third and most aggressive variant, called "**Output Unroll**," fully unrolls not only the kernel height and width but also both the input and output channel dimensions. For example, in the Conv1 layer, this approach would unroll across a set of loops covering 16x5x5x6 dimensions, resulting in a much larger parallelism factor of 2400. These three variants—No Unroll, Input Unroll, and Output Unroll—illustrate the varying degrees of parallelism that can be achieved by adjusting this critical design knob, showcasing how designers can tune different aspects of a given architecture for optimal performance.

Each of these unrolling techniques provides a unique approach to parallelism. The "No Unroll" variant maintains simplicity, focusing on temporal parallelism. "Input Unroll" offers more parallelism by extending across input channels and kernel dimensions, while "Output Unroll" maximizes parallelism across both input and output channels, offering the most comprehensive parallel structure. These variations help demonstrate the impact of unrolling on performance and resource utilization across different hardware architectures.

4.1.1 Naïve LeNet-5 topology

The Naive Design serves as a baseline implementation against which other architectural features are compared. In this system, data is transferred directly between scratchpad memories using DMA, and each layer of the network is processed sequentially, as illustrated in Figure 6.

The Naive design represents a simple and straightforward architecture for the given CNN. At a high level, the design consists of separate accelerators that access memory through a shared DMA. A central controller, referred to as the Top, manages all memory transfers to and from these accelerators and ensures synchronization between them. Due to the way memory is handled, there is no overlap in the execution of the accelerators, as each subsequent accelerator must wait for the complete output feature map from the previous layer before it can begin processing. This approach simplifies synchronization, as the Top controller processes each network layer one after the other and manages the corresponding memory transfers accordingly.

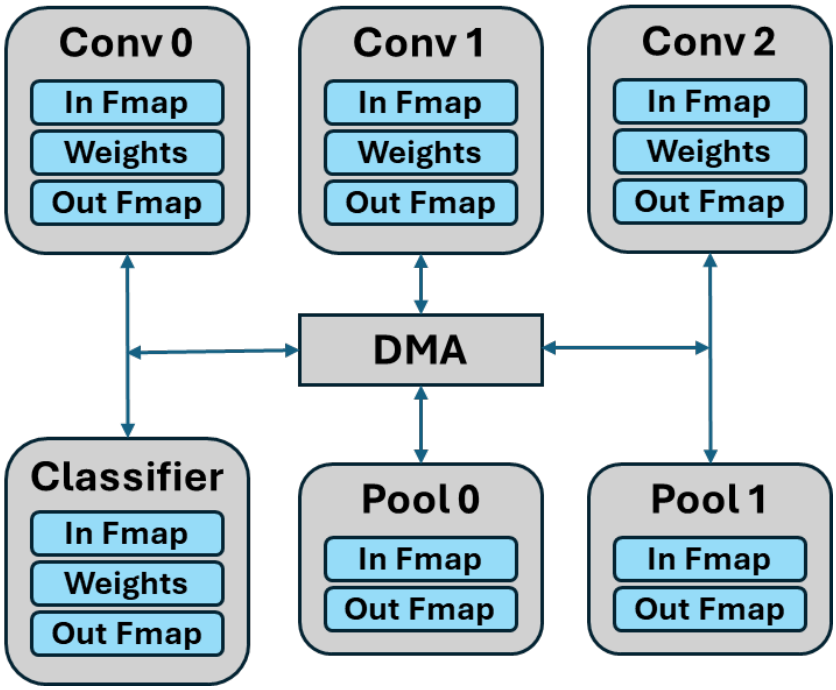

Figure 6. The Naive architecture design of LeNet-5.

4.1.2 Massive LeNet-5 topology

In the Massive architecture, accelerators are connected in a streaming-like configuration through scratchpad memories, with a dedicated "Data Sync" accelerator responsible for coordinating data transfers, as illustrated in Figure 7. This architecture allows for self-synchronization, significantly reducing control overhead and enabling overlapping execution of tasks.

Streaming in CNNs presents challenges due to the imbalance between data production and consumption. While data is typically written once, it is read multiple times because of the shifting convolutional windows. In the Naive design, separate input and output memories are used, and data is moved through a DMA, but this approach does not fully optimize the use of data across layers, missing opportunities for increased parallelism during execution.

To address this, the Massive architecture employs the Data Sync accelerator, which is specifically designed to manage data movement between layers. Unlike traditional DMA approaches, the Data Sync accelerator is aware of the data access patterns in the network, allowing it to transfer data between layers as soon as it becomes available. This transforms the input and output scratchpads into a highly efficient stream buffer, enabling a single write with multiple reads.

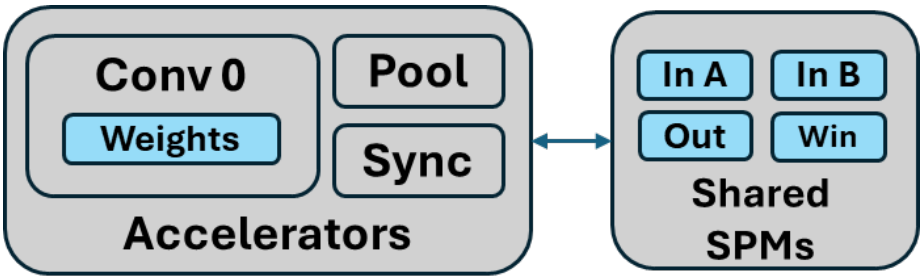

Figure 7. The Massively Parallel architecture design of LeNet-5.

This synchronization method allows each accelerator to execute operations based on data availability, without relying on a central controller, improving parallelism and efficiency across the architecture.

4.2 LeNet-5 Area & Performance Trade-Off

In Figure 8, we present the relative area estimates for the six design points considered in our design-space exploration study. The analysis reveals intriguing insights into the area footprints associated with different topologies and levels of parallelism. Notably, the Naïve and Massive topologies, which utilize only temporal parallelism without incorporating spatial parallelism (such as Input Unroll and Output Unroll), exhibit comparably small area footprints. However, it is worth noting that the area footprint of the Massive topology is approximately 25% larger than that of the Naïve topology.

As we introduce increased levels of parallelism—thereby engaging more functional units—we observe a substantial increase in area. Specifically, the Naïve topology that incorporates Input Unroll demonstrates an area that is eight times larger than its counterpart without any parallelism. This trend becomes even more pronounced with the addition of Output Unroll, where the area expands to a staggering 32 times greater than the base case. The Massive topology displays a similar pattern of growth in area with increased parallelism. In this case, the area of the Massive topology with Input Unroll is ten times larger than that of the configuration without any form of unrolling. When we consider Output Unroll, the area escalates dramatically, reaching 50 times the size of the no-Unroll configuration.

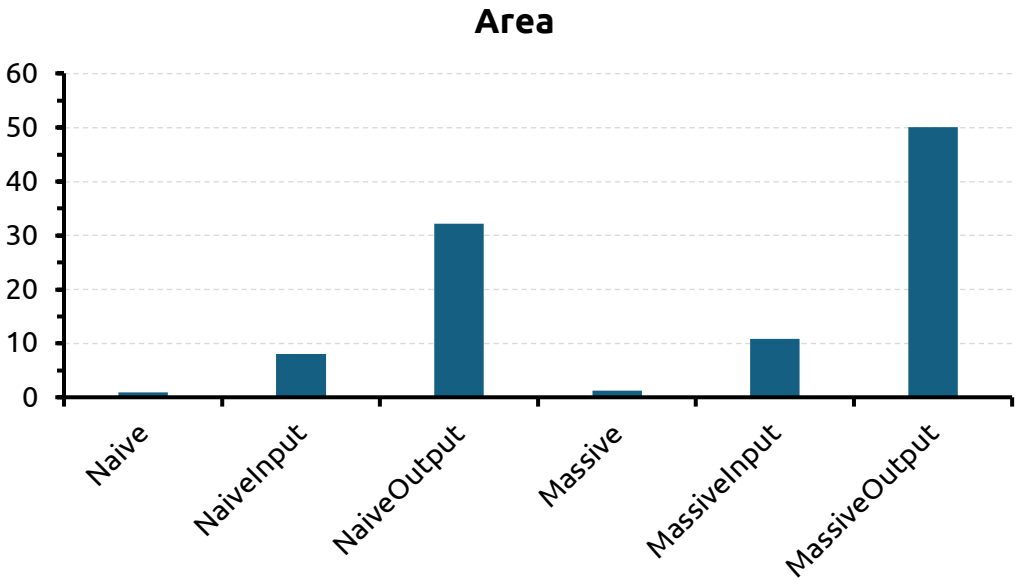

Figure 8. Relative area estimations for six design points.

These findings underscore the significant impact that varying degrees of parallelism can have on area utilization, highlighting the trade-offs involved in design decisions within the context of architectural exploration.

Figure 9, we present the relative performance measurements for the six design points that we have examined in our study. The Naïve implementation without any form of unrolling serves as the baseline for comparison. This baseline allows us to assess the performance improvements associated with different unrolling strategies and topologies.

The Naïve implementation with Input and Output unrolling variations shows modest yet notable performance gains. Specifically, the Naïve Input Unroll achieves an impressive performance enhancement of 8 times that of the baseline, while the Naïve Output Unroll variation offers a slightly better performance improvement of 19 times the baseline.

In contrast, the Massive implementation demonstrates significantly greater performance advantages when compared to the Naïve implementation. The Massive topology provides a baseline speedup of 4 times over the Naïve approach. However, the true potential of the Massive implementation is realized when we consider the effects of unrolling. With the introduction of Input Unroll, the performance skyrockets to an astounding 33 times faster than the baseline. Even more striking is the performance

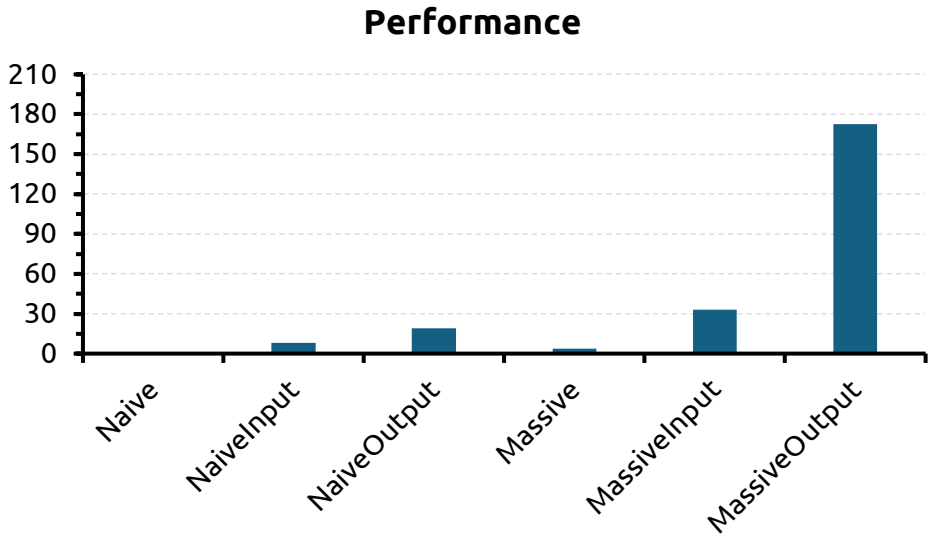

Figure 9. Relative performance assessment for the six design points.

increase observed with the Output Unroll variant, which reaches an impressive 170 times the baseline performance.

These results illustrate not only the effectiveness of different unrolling strategies in enhancing performance but also highlight the considerable advantages of the Massive implementation compared to the Naïve approach. This analysis underscores the importance of optimizing design points to achieve significant performance improvements in computational tasks.

4.3 Reliability Assessment of Heterogenous Architecture

To assess the reliability of the various design points considered in our study, we conducted statistical fault injection tests on the Scratchpad Memories (SPMs) of each accelerator configuration. The SPMs represent the largest array structures within the accelerator cluster, making them particularly susceptible to soft errors caused by cosmic particle radiation. Given the critical role that these memories play in the overall functioning of the accelerators, understanding their vulnerability to such errors is essential for ensuring reliable operation.

Our analysis reveals that the fully connected layer SPMs, especially those containing the weights of the layers, exhibit the highest vulnerability, with AVFs exceeding 80%, as shown in Figure 10. This elevated vulnerability can be attributed to the high density of these layers and their proximity to the output, where any errors could have a significant impact on the final results. The tightly packed nature of the weights in fully connected layers increases the likelihood of errors propagating to the output, potentially compromising the result of the computations being performed.

When examining the differences in AVF among the various unroll levels and the two topologies, we find only minor variations. These differences are primarily due to the distinct access patterns of the memory over time, which influence how susceptible each design point is to faults. In the Naïve topology, we observe that the output SPM of the first convolutional layer (conv0Output) also demonstrates a significant AVF of approximately 13%. This value, while considerably lower than that of the fully connected layer SPMs, still indicates a notable susceptibility to errors.

Overall, the findings from our reliability assessment underscore the critical importance of considering soft error vulnerabilities in the design and implementation of NN accelerators. By identifying the specific SPMs that exhibit the highest risk, engineers and researchers can develop more robust error mitigation strategies, ultimately leading to more reliable and resilient architectures capable of performing effectively even in radiation-prone environments.

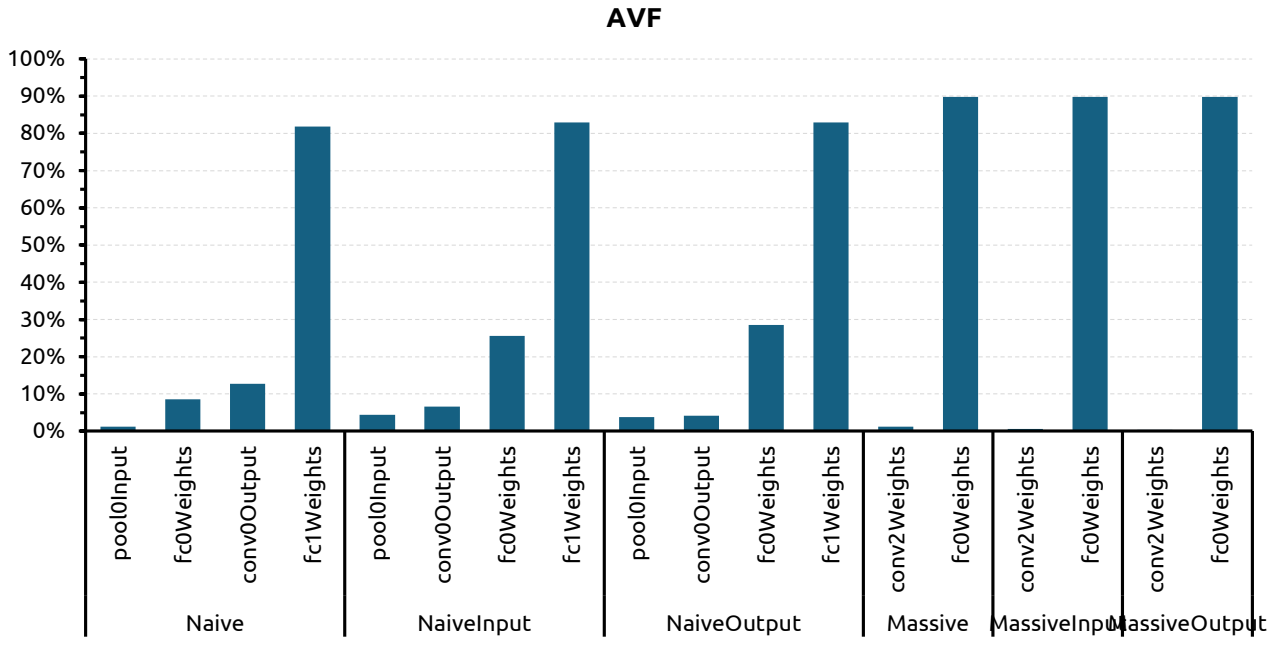

Figure 10. AVF for the target SPMs of each design point.

In Figure 11 we showcase the breakdown of fault effects (i.e., Silent Data Corruptions – SDC, and Masked faults) of the statistical injected faults. Due to the dataflow nature of the accelerators that we use in this study, the only possible outcomes of injected faults are either SDCs (i.e. output corruptions that are not detected by any mechanism) or masked faults (i.e. faults that are never propagated to the output). Masking can either be temporal or spatial. Temporal masking happens when an error bit is overwritten by a value and thus never used. Spatial masking can happen due to storage elements in an array being not fully utilized, and thus a corruption in an unused cell does not affect the computation being performed.

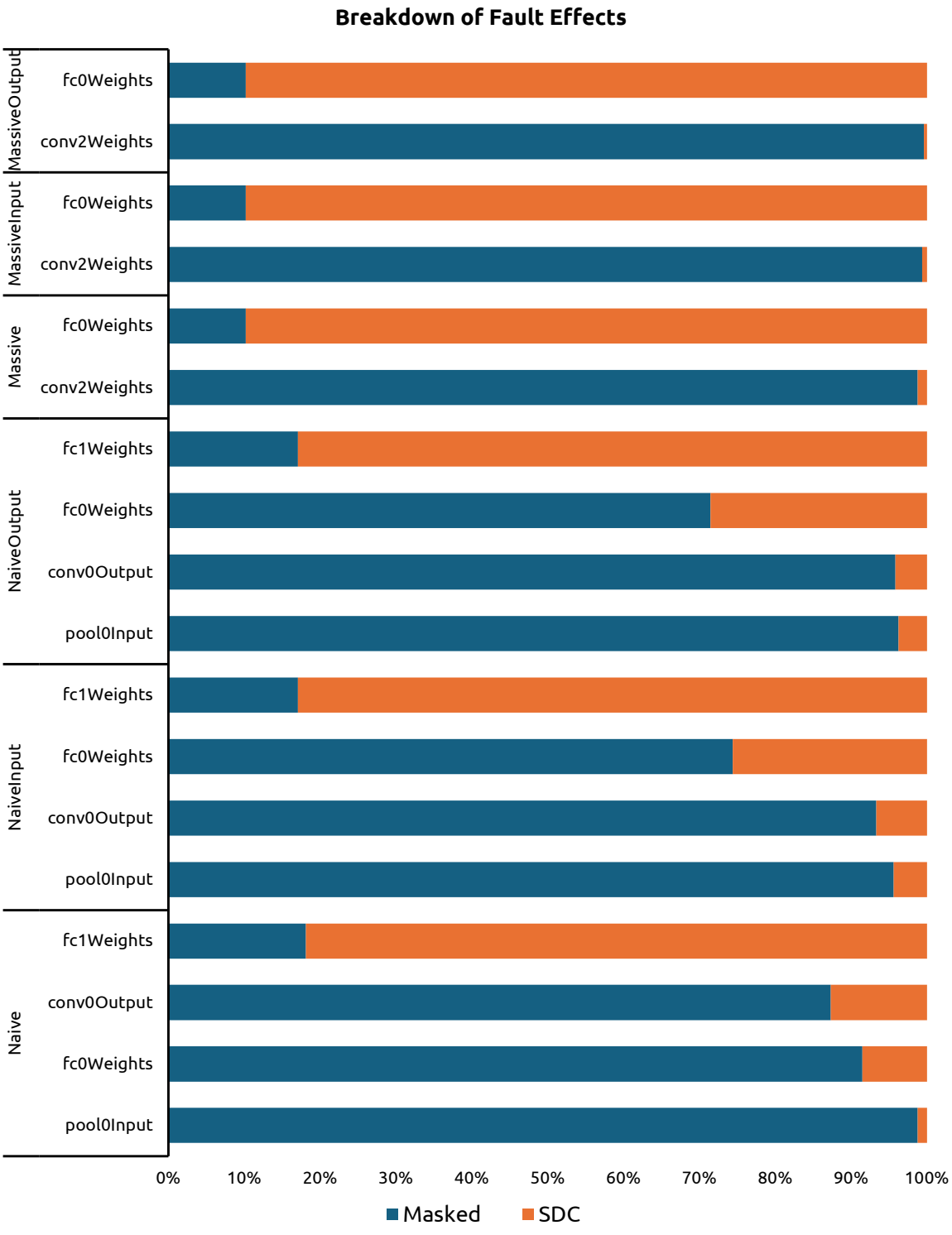

Figure 11: Fault effect breakdown of injected faults.

4.4 MobileNetv2 Modeling

We divide the MobileNetV2 architecture into four primary computational blocks, each assigned to a dedicated cluster. These clusters have distinct network structures, with the head, tail, and classifier functioning as single-use clusters. Thanks to the dynamic reconfigurability of the accelerators in our infrastructure, we can create a reusable, unified body cluster. Figure 12 illustrates the system-wide architecture, which extends the architecture of [3]. Although the Classification cluster is omitted due to its straightforward design, it is part of the network implementation. The DW functional unit addresses the producer-consumer imbalances in CNNs. In this functional unit, we introduce an internal Im2col accelerator for DW convolution, which processes data from an input FIFO stream to prepare the convolution window for the main computation accelerator. The PW functional unit has a distinct design to accommodate the specific data access patterns of the PW operation. This PW functional unit operates independently, handling both data management and computation. The "normal" convolution, is essentially the DW functional unit processing a 3-channel RGB image. Each functional unit writes its output to a FIFO stream that feeds into the next unit in sequence. Since the pointwise accelerator doesn't require input data reordering, it manages its own convolution window and processing. Using these functional units as building blocks, we construct four clusters, interconnecting them with FIFO buffers. Each cluster also includes Direct Memory Access (DMA) modules for main memory access and a top-level accelerator to handle memory transfers and initialization. Network inputs and outputs are managed via FIFO buffers accessed through a Stream DMA, configured by the top-level system. Weights and quantization parameters are transferred to their respective Scratch-Pad Memories (SPMs) before the accelerators begin computation.

The Head cluster is responsible for processing the network's first two layers, which include a standard convolution layer and a unique inverted residual block (IRB). The Body cluster is where the majority of the network's computations occur. The Body contains IRBs and support for residual connections. The Tail cluster extracts features for the classifier and performs average pooling. Finally, the Classifier cluster runs the fully connected layer that completes the network's computation.

Note that the modeling process for this design is still ongoing; therefore, we have not provided any design space exploration results in this version of the deliverable. However, we aim to include comparisons and a detailed assessment in the next iteration of the deliverable, alongside other electronic and neuromorphic accelerator designs.

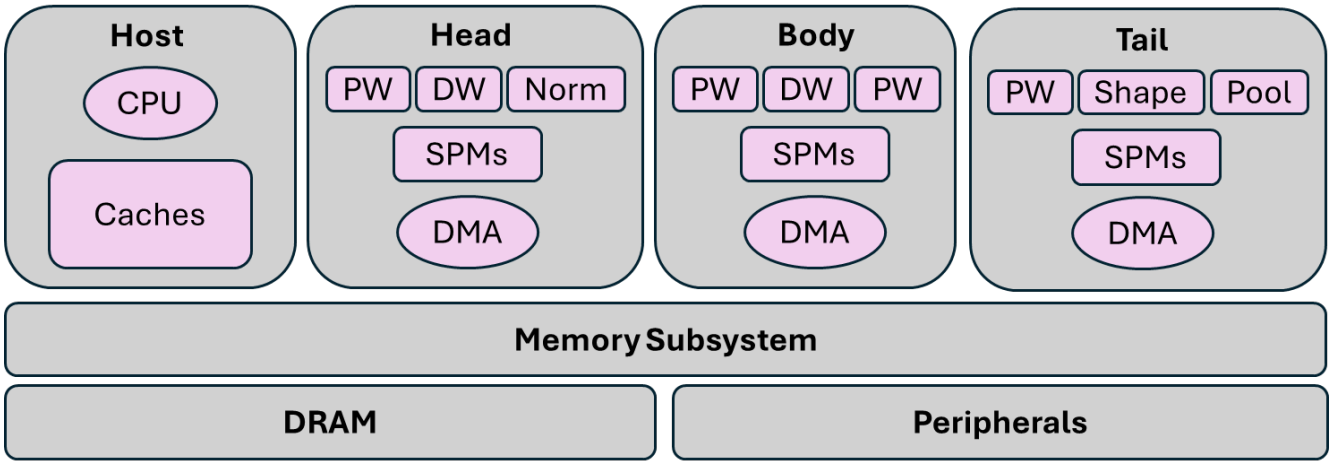

Figure 12. MobileNetV2 Accelerator Architecture.

5. Conclusion

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.

This deliverable lays the groundwork for the detailed modeling and evaluation of AI accelerators within the gem5 simulation framework, with special focus on electronic designs. By simulating NN workloads, such as Lenet-5, and analyzing critical performance metrics, we have taken significant steps toward understanding the operational dynamics and design trade-offs of these accelerators. The integration of photonic neuromorphic accelerators in the next iteration of the deliverable will further broaden the scope of our exploration, positioning our work at the forefront of heterogeneous computing system design. As we continue to refine our models and expand our benchmarking efforts, the next deliverable iteration (D5.2) will delve deeper into neuromorphic accelerator architectures and their potential for energy-efficient AI computation. Ultimately, this work contributes to a versatile simulation environment that supports comprehensive system-level evaluations, fostering innovation in AI hardware design.

6. 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] *J.* Lowe-Power, et al., The gem5 Simulator: Version 20.0+, arXiv, 2020, https://arxiv.org/abs/2007.03152.

[3] Z. Spencer, S. Rogers, J. Slycord, H. Tabkhi, "Expanding hardware accelerator system design space exploration with gem5-SALAMv2," Journal of Systems Architecture, Volume 154, 2024, https://doi.org/10.1016/j.sysarc.2024.103211.

[4] 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.

[5] 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.

[6] A. Sandberg, S. Diestelhorst, W. Wang, "Architectural Exploration with gem5", ASPLOS 2017, https://www.gem5.org/assets/files/ASPLOS2017_gem5_tutorial.pdf

[7] 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.

[8] S. Li, Z. Yang, D. Reddy, A. Srivastava and B. Jacob, "DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator," in *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106-109, 1 July-Dec. 2020, doi: 10.1109/LCA.2020.2973991.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] 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.
[22] 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.
[23] G. Moore, "Cramming more components onto integrated circuits," Proceedings of the IEEE, vol. 86, no. 1, pp. 82–85, 1998.
[24] 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.
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] 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.
[31] "gem5 GitHub Repository," https://github.com/gem5/gem5, accessed: 2024-02-10.
[32] 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.
[33] 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.
[34] 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.
[35] 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.
[36] 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.
[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.
[38] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
[39] M. Sandler, A. Howard, "MobileNetV2: The Next Generation of On-Device Computer Vision Networks", https://research.google/blog/mobilenetv2-the-next-generation-of-on-device-computer-vision-networks/
[40] https://paperswithcode.com/dataset/mnist