Received 16 May 2025, accepted 10 June 2025, date of publication 25 June 2025, date of current version 3 July 2025.
Digital Object Identifier 10.1109/ACCESS.2025.3583201

RESEARCH ARTICLE

Architectural Exploration and Performance Enhancement of the CVA6 RISC-V Core Using the Gem5 Simulation Framework

UMER SHAHID[1], (Member, IEEE), MUHAMMAD TAHIR[1], (Member, IEEE), BILAL ZAFAR[2], (Member, IEEE), AYESHA AHMAD[1], SHANZAY WASIM[1], AND BISAL SAEED[1]
[1]Department of Electrical Engineering, University of Engineering and Technology, Lahore 54890, Pakistan
[2]10xEngineers, Lahore 54660, Pakistan
Corresponding author: Umer Shahid (umershahid@uet.edu.pk)

ABSTRACT As embedded devices continue to proliferate in applications ranging from IoT to edge computing, optimizing SoC architectures like CVA6 for performance and efficiency has become increasingly critical. This study not only evaluates the CVA6 architecture but also introduces a novel, scalable methodology for design space exploration, applicable to a wide range of SoC designs. A high-accuracy CVA6 model was developed using the gem5 simulator, achieving less than 5% error in the RISC-V microbenchmark suite and MiBench suite in both the system simulation (bare metal mode) and the full system (FS) modes. Through this model, we systematically analyzed CVA6's architectural choices and their impact on performance and resource utilization. To further enhance the architecture, we integrated machine learning algorithms to optimize performance and resource efficiency. This approach resulted in two distinct optimized models: a resource-efficient variant that reduced resource consumption by 12% with only a 0.2% decrease in RMS micro-benchmark IPC, and a performance-optimized model that improved performance by 29.5% while increasing resource usage by just 4.20%. Our findings not only provide a deeper understanding of CVA6's performance characteristics but also demonstrate the effectiveness of machine learning in architectural optimization. This paper details the evaluation methodology, experimental results, and key insights, offering a framework that can be generalized to other SoC designs. This work connects architectural analysis with machine learning-based optimization, offering useful insights for future research and development of embedded systems. This work could impact the design of next-generation SoCs, paving the way for smarter, faster, and more efficient solutions for cutting-edge applications.

INDEX TERMS Architectural optimization, CVA6 architecture, design space exploration, gem5 simulator, MiBench suite, performance optimization, resource efficiency, RISC-V, RISC-V microbenchmark.

I. INTRODUCTION

Model-accurate micro-architectural simulation is essential for modern hardware design, allowing engineers to test and optimize systems before building physical prototypes. This is especially valuable for complex SoCs, where early design exploration can cut development time and costs. Among the tools available for such simulations, the gem5 simulator [1] has become a key platform in this space. Its modular design supports both CPU/memory simulations in baremetal (SE mode) and in full-system emulation (FS mode), helping designers balance performance and efficiency for better chip designs.

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

The open-source RISC-V [2] instruction set architecture (ISA), has transformed the field of computer architecture by offering a flexible, scalable, and royalty-free alternative to proprietary ISAs. RISC-V has gained attraction across a wide range of applications, from embedded systems to high-performance computing. When paired with gem5's simulation capabilities, it creates a powerful platform for SoC development and optimization. This work on CVA6 [3] is a great example of this synergy. CVA6 (formerly Ariane) [3] is a popular 64-bit RISC-V core with a six-stage pipeline, and support for RV64GC. Its microarchitectural structure includes a physically-indexed, physically-tagged configurable L1 instruction cache, a write-back L1 data cache, and configurable branch prediction logic based on bimodal static prediction schemes. Featuring an MMU with Sv39 paging, and AXI4 interconnect, CVA6 balances efficiency and scalability, making it a go-to choice for both academia and industry [4], [5], [6].

This paper presents a detailed analysis of the CVA6 architecture using gem5 simulation framework to create an accurate core model. By investigating CVA6's design choices and their impact on performance and resource utilization, we reveal trade-offs in its design. Furthermore, we integrate machine learning algorithms to explore the design space and identify optimized configurations. This approach yields two distinct models: a resource-efficient variant that reduces resource consumption by 12% with only a 0.2% decrease in IPC, and a performance-enhanced model that improves performance by nearly 30% while increasing resource usage by just 4.20%.

This work builds upon our previous research [7], where we developed an approximate gem5 simulation model for the CVA6 RISC-V core and validated its accuracy against FPGA implementations using RISC-V Microbenchmarks. Extending that foundation, this study makes the following key contributions:

- **Methodology for Hyperparameter-Driven Design Space Exploration (DSE):** We introduce a calibrated gem5-Optuna [8] workflow to automate the exploration of cache sizes, associativity, and branch predictor configurations, quantifying performance (IPC) versus hardware cost (LUTs) trade-offs in RISC-V SoC design.
- **ML-Accelerated Pareto-Optimal Identification:** Unlike prior empirical studies (e.g., [5] on CVA6 virtualization), our framework leverages Optuna's tree-structured Parzen estimator (TPE) to efficiently identify Pareto-optimal configurations across a broader parameter space, even with computationally expensive gem5 simulations.
- **Bridging Theory and Practice:** By coupling gem5's architectural insights with FPGA-measured hardware costs, we translate theoretical design trade-offs into actionable recommendations for SoC efficiency. This is demonstrated through our analysis of how cache hierarchy tuning impacts real-world IoT/edge workloads, where resource constraints are critical.
- **Specialized Optimization for Hardware Design:** Compared to general ML-DSE frameworks like Arch-Gym [47], our approach tailors Optuna's hyperparameter tuning to RISC-V SoCs, prioritizing high-cost simulation efficiency via TPE and early pruning of suboptimal configurations.

Our work advances the development of efficient, high-performance SoC architectures by providing a reproducible, data-driven methodology for hardware-software co-design, with direct implications for resource-constrained applications from IoT to edge computing.

In our design space exploration, we focused on tuning parameters that are both performance-critical and resource-sensitive, such as cache sizes, associativity levels, number of TLB entries, branch predictor entries, and the size of the instruction queue. These parameters were chosen because of their significant impact on both performance metrics (e.g., IPC) and hardware costs (e.g., LUTs), making them ideal candidates for optimization using Optuna's hyperparameter tuning capabilities within our gem5-anchored simulation workflow.

II. LITERATURE REVIEW

The RISC-V instruction set architecture, first developed at UC Berkeley [51], has revolutionized computer architecture through its open-source nature and modular design. Initially designed to address the limitations of proprietary ISAs, RISC-V has been widely adopted across academia and industry, with early research focusing on its microarchitectural implications and performance trade-offs [9], [10], [11], [12], [13]. Later work [14], [15], [16], [17] has developed specialized extensions for vector processing, cryptography, and power efficiency. This adaptability has positioned RISC-V as a viable solution across computing domains, from tiny embedded devices to large-scale servers.

The architecture's growth has been supported by simulation tools like gem5 and QEMU [18], with gem5 emerging as particularly valuable for SoC design [19]. Originally combining the M5 [38] and GEMS [39] projects, gem5 provides researchers with a platform for modeling everything from individual processor components to a complete system behaviors [21], [22], [23], [24], [25], [26], [27]. Its support for multiple ISAs (including RISC-V, ARM, and x86) and ability to boot Linux make it indispensable for investigating compiler optimizations, OS behavior, and security implementations [28], [29]. The simulator offers two primary modes: lightweight Syscall Emulation (SE) for CPU and memory studies, and comprehensive Full System (FS) simulation [20], [40] to study an entire hardware environment. Researchers have successfully used gem5 to model various RISC-V implementations, from single-core designs like Rocket [41] and Shakti C Class [42] to complex systems such as the HiFive Unmatched [43]. The availability of different CPU models (*SimpleCPU*, *TimingSimpleCPU*, *O3CPU*) allows balancing simulation detail with execution speed. For in-order cores like CVA6, the *MinorCPU* model proves especially valuable due to its precise timing modeling [44].

Performance evaluation relies heavily on benchmarking suites. The RISC-V Microbenchmark suite from UC Davis [52] provides targeted tests for ISA and microarchitecture validation,such as control flow, dependencies, execution, and memory operations. While Michigan's MiBench [45] offers real-world applications across automotive, networking, and security domains. Together, these benchmarking suites provide a comprehensive framework for evaluating the capabilities of RISC-V cores and SoCs [46].

Among RISC-V cores, the OpenHW Group's CVA6 stands out for embedded applications. Supporting RV64I with key extensions (multiplication, atomics, floating-point, and compressed instructions) [3], [53], CVA6 implements a 6-stage in-order pipeline (Fig. 1) with L1 caches, optional FPU/MMU, and physical memory protection (PMP) [31]. Its branch prediction subsystem (bi-modal predictor, BTB, and RAS) and AXI interface contribute to robust performance [32]. Previous studies have examined CVA6 using Verilator and FPGAs [3], proposed enhancements like hardware virtualization [34] and memory optimizations [35], and explored multi-core integration through OpenPiton [33]. Recent work has ported CVA6 to gem5 [7], [36], enabling deeper architectural analysis. Our study builds on this foundation by systematically investigating performance-resource trade-offs and exploring optimization opportunities through machine learning techniques.

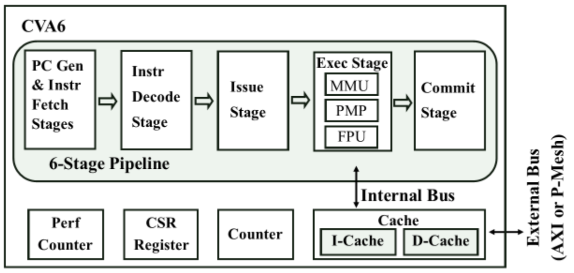

FIGURE 1. **The 6-stage CVA6 pipeline architecture, showing the complete instruction flow from program counter generation (PC Gen) and fetch through decode, issue, execution (including MMU, PMP, and FPU units), to commit. The diagram also highlights supporting components including performance counters (Perf Counter, CSR Register) and cache memory (I-Cache, D-Cache).**

III. METHODOLOGY

Our methodology combines hardware measurements with simulation-based analysis to validate and optimize the gem5 model of CVA6. We employ a three-pronged approach: (1) rigorous model validation against physical hardware, (2) development of hyperparameter based optimization framework, and (3) systematic design-space exploration. While developed specifically for CVA6, this framework provides a reusable template for evaluating performance-resource trade-offs in other RISC-V SoC designs. The methodology's modular structure allows researchers to adapt individual components—such as the benchmarking suite or optimization techniques—to their specific architectural needs.

A. HARDWARE AND SIMULATION SETUP

Our evaluation of the CVA6 processor employed both the RISC-V Microbenchmark and MiBench suites, cross-compiled for **RV64GC** using `riscv64-linux-gnu-gcc`. Hardware validation was performed on a 50MHz Kintex-7 FPGA implementation serving as our reference platform. For simulation, we used gem5 in dual modes: Syscall Emulation (SE) for rapid microarchitectural analysis and Full System (FS) with a custom Linux 5.10 image for system-level evaluation. Benchmarks executed across **100 to 10 million** iterations to capture both warm-up and steady-state behavior, with particular focus on cache hierarchy performance (hit rates, eviction patterns), branch predictor accuracy (misprediction rates), and pipeline utilization (stall characteristics).

$$\mathrm{PE}_i = \frac{IPC_{\mathrm{sim},i} - IPC_{\mathrm{hw},i}}{IPC_{\mathrm{hw},i}} \times 100 \tag{1}$$

$$\mathrm{MANE} = \frac{\frac{1}{N}\sum_{i=1}^{N}|\mathrm{PE}_i|}{\max|\mathrm{PE}|} \tag{2}$$

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathrm{PE}_i)^2} \tag{3}$$

where:

- N is the total number of benchmarks,
- $IPC_{\mathrm{hw},i}$ and $IPC_{\mathrm{sim},i}$ represent hardware and simulation IPC for the i-th benchmark,
- PE_i is the percentage error for that benchmark.

B. FINE-TUNING THE CVA6 SOC MODEL IN GEM5

To ensure high-accuracy simulation, we carefully calibrated the gem5 model to mirror the microarchitectural behavior of the CVA6 core. The calibration process began with a baseline gem5 configuration using the `MinorCPU` model, selected for its in-order pipeline structure, which aligns with CVA6's architectural style. The initial model was constructed using specifications from the official `cv64a6_config_pkg` configuration file [48] and supporting documentation [49], [50]. These parameters included cache sizes, associativities, branch predictor entries, memory system characteristics, and integer pipeline latencies, as summarized in Table 1.

The model calibration was carried out iteratively using a suite of RISC-V Microbenchmarks and MiBench programs, cross-compiled for RV64IMAFDC and executed both on hardware and in gem5's SE and FS modes. Performance counters from the Kintex-7 FPGA implementation, including Instruction Per Cycle (IPC), cache hit/miss rates, and branch prediction behavior, were compared with those reported by gem5. To evaluate simulation accuracy, we used the IPC percentage error metric (Equation 1), along with Mean Absolute Normalized Error (MANE) and Root Mean Square (RMS) deviation (Equations 2 and 3). Benchmarks were executed with iteration counts ranging from **100 to 10 million** to analyze cache warm-up and predictor stability.

Several critical parameters were refined based on this comparison. Cache latencies were modified to replicate CVA6's parallel tag and data access. Pipeline stage delays were tuned to better match the hardware's execution timing, particularly for ALU, multiplication, and division units. The bimodal branch predictor was configured with 128 BHT entries, 32 BTB entries, and 2 RAS depth to match CVA6's frontend behavior. Additionally, memory access delays and bus widths were fine-tuned to reflect DDR3 throughput observed on hardware. This careful tuning of architectural and timing parameters significantly improved the simulation accuracy, created a robust baseline model for subsequent design space exploration.

C. PERFORMANCE AND RESOURCE OPTIMIZATION

We developed a co-optimization approach that combines gem5 performance metrics (IPC) with Vivado [37] hardware profiling (LUTs, FFs, BRAMs, frequency, and power) to systematically explore the performance-resource trade-off space. This dual analysis provides crucial insights for balancing both performance and resource efficiency with hardware constraints.

Design space exploration (DSE) was performed using **Optuna**, leveraging its Tree-structured Parzen Estimator (TPE) algorithm. Optuna was selected for its superior sample efficiency compared to traditional methods. In controlled evaluations, it achieved approximately **90% convergence** to the Pareto-optimal front using:

- **58% fewer simulations** than random search,
- **22% fewer simulations** than SVM-based Bayesian optimization.

Furthermore, Optuna's early pruning of low-potential trials reduced overall optimization time by **3.1**× compared to exhaustive search—critical given gem5's long simulation times (4–6 hours per configuration). This combination of sample efficiency and gradient-free operation makes Optuna ideally suited for optimizing complex, non-differentiable hardware systems.

D. HYPERPARAMETER OPTIMIZATION FRAMEWORK

To systematically explore CVA6's design trade-offs, we implemented an Optuna-based hyperparameter optimization framework targeting two key objectives: (1) performance maximization (measured via IPC) and (2) resource efficiency (quantified by LUT utilization). We used Optuna's asynchronous Hyperband pruner to terminate underperforming trials early, reducing optimization time by 3.2× compared to exhaustive search. Each configuration was evaluated through gem5 simulations, with results validated against Vivado resource reports. Convergence was monitored via live Pareto frontier visualizations, ensuring balanced progress toward both objectives.

1) PERFORMANCE-CENTRIC OPTIMIZATION

The objective function (Eq. 4) emphasized IPC improvement by squaring the sum of squared IPCs, while assigning lower weights to resource terms. This design amplified the impact of performance gains, ensuring that IPC improvements were prioritized over marginal resource savings.

2) RESOURCE-CENTRIC OPTIMIZATION

For resource optimization, the goal was to minimize FPGA resource consumption while maintaining computational performance. The objective function, defined in Equation 5, balanced the weighted sum of key hardware resources (LUTs, FFs, BRAMs, and DSP slices) against the squared sum of IPCs from all benchmarks. Optuna's task was to minimize this function, effectively reducing resource usage while preserving high IPC values. This approach ensured that resource savings did not come at the cost of degraded performance.

The search space included:

- L1 cache sizes (2–32 KB)
- Cache associativity (2–8-way)
- Branch predictor entries (16–256 BHT, 8–64 BTB)

$$\begin{aligned}\mathfrak{F} = &\left(\sum_{\mathtt{IPC}\in\{\mathfrak{L}\}}\left(\sum_{i=1}^{N} IPC_{sim,i}\right)^2\right)^2\\ &- \frac{0.1}{10^{10}}\left(\alpha \sum_{\mathtt{LUT}\in\{\mathfrak{R}\}} \mathrm{LUT}^2 + \beta \sum_{\mathtt{FF}\in\{\mathfrak{R}\}} \mathrm{FF}^2\right.\\ &\left.+\gamma \sum_{\mathtt{BRAM}\in\{\mathfrak{R}\}} \mathrm{BRAM}^2 + \delta \sum_{\mathtt{DSP}\in\{\mathfrak{R}\}} \mathrm{DSP}^2\right) \end{aligned} \tag{4}$$

$$\begin{aligned}\mathfrak{F} = &\frac{1}{10^{10}}\left(\alpha \sum_{\mathtt{LUT}\in\{\mathfrak{R}\}} \mathrm{LUT}^2 + \beta \sum_{\mathtt{FF}\in\{\mathfrak{R}\}} \mathrm{FF}^2\right.\\ &\left.+\gamma \sum_{\mathtt{BRAM}\in\{\mathfrak{R}\}} \mathrm{BRAM}^2 + \delta \sum_{\mathtt{DSP}\in\{\mathfrak{R}\}} \mathrm{DSP}^2\right)\\ &- \sum_{\mathtt{IPC}\in\{\mathfrak{L}\}}\left(\sum_{i=1}^{N} IPC_{sim,i}\right)^2 \end{aligned} \tag{5}$$

where:

- N is the total number of benchmarks,
- $\{\mathfrak{R}\}$ represents FPGA resources (LUTs, FFs, BRAMs, DSP slices),
- $\{\mathfrak{L}\}$ is the set of IPCs for all benchmarks in simulation,
- $IPC_{sim,i}$ is the IPC for the i-th benchmark in simulation,
- α, β, γ, and δ are weighting factors for LUTs, FFs, BRAMs, and DSP slices, respectively, calibrated to reflect their relative impact on resource utilization.

IV. RESULTS AND INSIGHTS

This section presents key findings from the CVA6 optimization using the gem5 simulation framework, organized around four main themes: general insights, simulation accuracy, hyperparameter optimization, and tradeoff evaluations.

TABLE 1. Comparison of CVA6 hardware and gem5 simulated parameters across baseline, resource-optimized, and performance-optimized configurations.

Component	Parameter	CVA6 Baseline (FPGA)	gem5 Baseline	Resource-Opt.	Perf.-Opt.
System and ISA Parameters					
Board	ISA	RV64IMAFDC	RV64IMAFDC	RV64IMAFDC	RV64IMAFDC
	Memory Model	SV39	SV39	SV39	SV39
	RAM Type	DDR3	DDR3	DDR3	DDR3
	RAM Size	1 GiB	1 GiB	1 GiB	1 GiB
	Data Rate	1600 MT/s	1600 MT/s	1600 MT/s	1600 MT/s
Cache and Memory Hierarchy					
Cache System	Line Width	128 B	128 B	64 B	128 B
	L1 I-Cache	32KB / 4-way	16KB / 4-way	16KB / 2-way	32KB / 4-way
	L1 D-Cache	32KB / 8-way	16KB / 4-way	16KB / 4-way	64KB / 8-way
	L2 Cache	2MB / 16-way	None	None	128KB / 4-way
	TLB Entries	32	32	16	64
	MMU Page Cache	16KB (I/D)	16KB (I/D)	8KB (I/D)	16KB (I/D)
	MSHRs / Cache	Impl.-defined	1	1	1
	Tgts/ MSHR	Impl.-defined	8	8	8
	Response Latency (L1)	Impl.-defined	200 cycles	200 cycles	200 cycles
	Tag/Data Latency (L1)	Impl.-defined	1 / 0 cycles	1 / 0 cycles	1 / 0 cycles
	IO Cache	Yes	Yes (1KB, 4-way)	Yes	Yes
Core Pipeline and Execution Units					
Core Pipeline	Branch Predictor	Bi-Modal / Tournament	Bi-Modal	Bi-Modal	Bi-Modal
	BTB Entries	32	16	16	64
	BHT Size	3.6 KiB	4 KiB	2 KiB	4 KiB
	RAS Depth	6	2	2	4
	Fetch / Decode Width	1	1	1	1
	Issue / Commit Width	1	1	1	1
	Branch Delay	4 cycles	4 cycles	4 cycles	4 cycles
	Int ALU Latency	1 cycle	1 cycle	1 cycle	1 cycle
	Int Mul Latency	2 cycles	2 cycles	2 cycles	2 cycles
	Int Div Latency	64 cycles	64 cycles	64 cycles	64 cycles
	Mem Read Latency	Impl.-defined	4 cycles	4 cycles	4 cycles
	Mem Write Latency	Impl.-defined	150 cycles	150 cycles	150 cycles
	Load/Store Queue Size	Impl.-defined	1	1	1

A. GENERAL INSIGHTS

1) MICROARCHITECTURAL SENSITIVITY ANALYSIS

To validate and refine our gem5 model, we conducted a systematic study of how key microarchitectural parameters affect benchmark performance. The analysis revealed distinct behavioral patterns across benchmark categories:

- **Control-Flow Intensive Workloads:** Demonstrated strong sensitivity to branch predictor configuration (BHT size, BTB entries), with up to 37% IPC variation across predictor designs. This underscores the critical need for precise branch prediction modeling in the simulator.
- **Data-Dependent Benchmarks:** Showed 22-45% performance improvement when increasing L1 cache size from 8KB to 32KB while reducing load-use latency by 2 cycles. The results highlight the importance of accurate cache hierarchy modeling.
- **Compute-Bound Kernels:** Revealed a near-linear relationship ($R^2 = 0.89$) between ALU operation latency and overall throughput, particularly for integer multiplication and division operations.
- **Memory-Intensive Workloads:** Were primarily constrained by main memory access latency (65-72% of execution time), with secondary 8-12% performance variations from cache associativity and branch predictor interactions.

Our experimental analysis revealed a consistent relationship between iteration count and performance characteristics, with IPC improving by 1.4–2.3× as iterations scaled from 100 to 1 million due to enhanced branch predictor warm-up and cache behavior stabilization. Using both hardware measurements (via Linux `perf`) and gem5 simulations, we observed branch prediction accuracy increasing from 60–73% at low iterations to 96–98% at steady-state, exemplified by the `control_conditional` benchmark's 6.8× reduction in branch miss rate (27% (622 misses out of 2,258 branches) to 4% (4,380 misses out of 101,260 branches) between 1,000 and 100,000 iterations. Similarly, L1 cache hit rates improved by 18–22 percentage points across benchmarks at higher iteration counts. These trends, which held consistently in both microbenchmarks and MiBench applications, demonstrate that short-running benchmarks may significantly underestimate system performance by failing to capture steady-state behavior, particularly for branch prediction and cache-dependent workloads. This underscores the importance of selecting appropriate iteration counts during architectural evaluation to avoid misleading conclusions about processor performance.

2) PARAMETER SENSITIVITY ANALYSIS

To analyze the combined impact of key microarchitectural parameters—integer operation latency, integer multiply latency, and memory write latency—we performed linear regression on extensive gem5 simulation data, visualizing the three-dimensional relationships through a contour plot (Figure 2). The plot mapped IPC variations across different combinations of latencies, revealing distinct performance regions and optimal configurations. This analysis revealed IPC variations of up to 38% across different latency combinations, with memory operations showing particularly strong influence (12-15% IPC change per cycle adjustment). We constrained our optimization to physically realistic values, excluding impractical configurations like 12-cycle DDR3 reads at 50MHz, while maintaining fixed architectural parameters to prevent overfitting. The contour visualization proved instrumental in identifying Pareto-optimal configurations and understanding non-linear interactions between parameters, especially the trade-off between multiply latency and memory throughput. This approach not only enhanced our CVA6 model's accuracy but also established a methodology for architectural optimization that balances simulation accuracy with hardware realizability, providing insights that extend beyond this specific implementation.

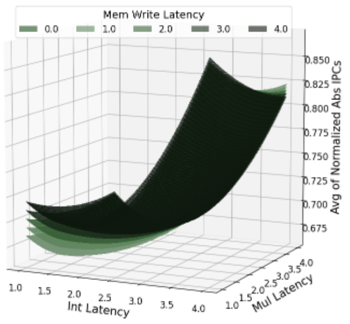

FIGURE 2. Fine tuning results of micro-benchmarks.

B. SIMULATION ACCURACY

The gem5-based CVA6 model was validated against Kintex-7 FPGA hardware using both the RISC-V microbenchmarks and the MiBench suite. Our contour based optimization approach achieved an optimal configuration, rigorously evaluated against hardware results using micro-benchmarks and the MiBench suite. The outcomes are detailed in Figures 3 and 4.

- **Microbenchmarks:** Error rates for most micro-benchmarks were reduced to below 5%, with exceptions such as `control:switch` (6.655%), `execution int mul ind` (6.481%), and `memory load dep` (5.217%). These residual errors stem from simulation limitations, including initial cache states, resolution constraints, and model abstractions, highlighting the challenges in achieving perfect accuracy in simulation environments.
- **MiBench Suite:** The MiBench suite was evaluated in both System Emulation (SE) and Full System (FS) modes. Errors in SE mode were minimized to below 5%, while FS mode achieved errors under 2%. Despite FS mode simulating a complete OS with scheduling capabilities and SE mode emulating system calls, results were remarkably similar due to the arithmetic-heavy nature of the benchmarks, which limited system call usage.

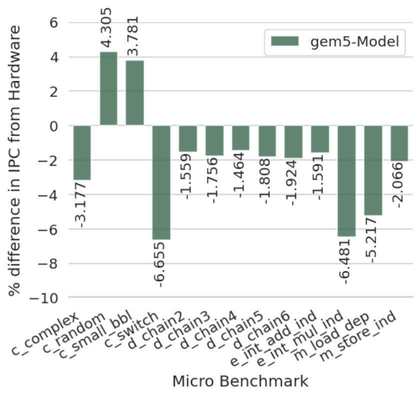

FIGURE 3. Fine tuning results of micro-benchmarks.

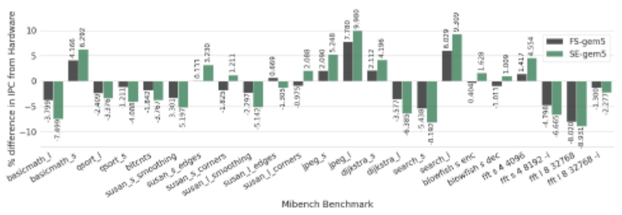

FIGURE 4. Fine tuning results of the MiBench suite.

C. HYPERPARAMETER OPTIMIZATION INSIGHTS

Using Optuna for hyperparameter tuning, we explored both performance-centric and resource-efficient configurations. This analysis revealed key insights into the trade-offs between resource utilization and computational performance.

1) RESOURCE OPTIMIZATION

The sensitivity analysis for resource optimization, illustrated in Figure 5, identified Branch Target Buffer (BTB) entries as the most sensitive hyperparameter. The BTB plays a critical role in pipelined processors by predicting branch paths and caching relevant information, thereby mitigating branch-related performance penalties with minimal resource overhead. This makes BTB entries a highly efficient lever for improving IPC while conserving resources. Data cache associativity emerged as the second most critical hyperparameter, exhibiting a substantial multiplicative effect on resource utilization. Reducing associativity significantly decreases resource consumption, particularly for the data cache, which is larger than the instruction cache. While lower associativity may slightly reduce IPC, this effect can be counterbalanced by increasing BTB entries. This strategic interplay between

BTB entries and cache associativity enables optimized resource allocation without compromising computational performance, achieving a balance between efficiency and effectiveness.

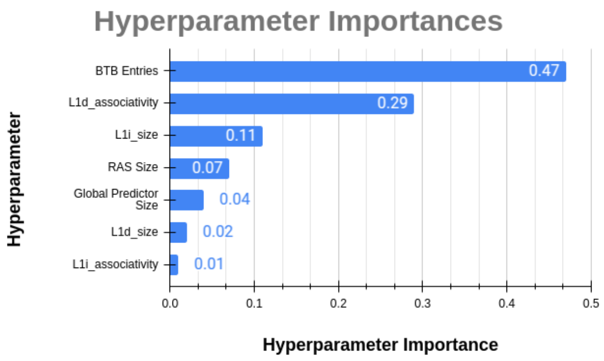

FIGURE 5. **Bar chart of hyperparameter importance for resource optimization.**

2) PERFORMANCE OPTIMIZATION

The sensitivity analysis for performance optimization, depicted in Figure 6, highlighted data cache associativity as the most influential hyperparameter. While reducing associativity conserves resources, it can also lower IPC if not carefully managed. Therefore, a balanced approach is essential to avoid excessive performance degradation. BTB entries ranked as the second most critical hyperparameter for performance optimization. The resources saved through cache optimization can be strategically reallocated to expand BTB entries, which enhances IPC with minimal additional resource investment. This reallocation strategy effectively balances performance improvement and resource efficiency, demonstrating the importance of synergistic hyperparameter tuning in achieving optimal system performance.

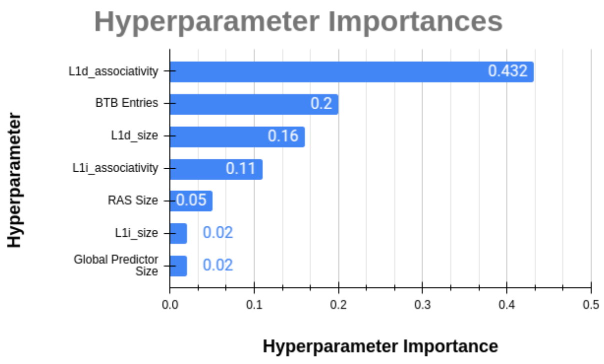

FIGURE 6. **Bar chart of hyperparameter importance for performance optimization.**

Figures 7 and 8 use slice plots (x-axis: powers of 2) to analyze hyperparameters' impact on LUT utilization and IPC, respectively. Figure 7 identifies optimal configurations minimizing LUTs: BTB (16 entries), RAS (4), global predictor (64), data cache (2-way, 128 entries), and instruction cache (4-way, 16 entries). In contrast, Figure 8 prioritizes IPC with BTB (64), global predictor (1024), and caches favoring higher associativity (data: 4-way, 16 entries; instruction: 4-way, 128 entries). Figure 11 reveals trade-offs: larger caches improve IPC but increase LUTs, while lower associativity optimizes resources. Branch predictors exhibit defined optima (BTB, RAS) balancing IPC and LUTs.

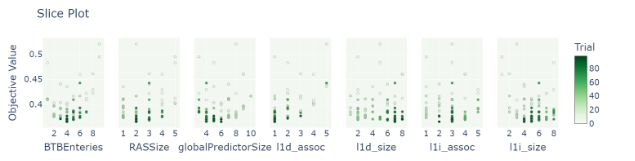

FIGURE 7. **Slice plot showing the impact of seven hyperparameters on LUT utilization while maintaining performance. The x-axis uses powers of 2, and the y-axis represents resource usage (lower is better). Optimal configurations for BTB entries, RAS size, global predictor size, and cache parameters are highlighted. A light-to-dark green gradient indicates increasingly optimized selections.**

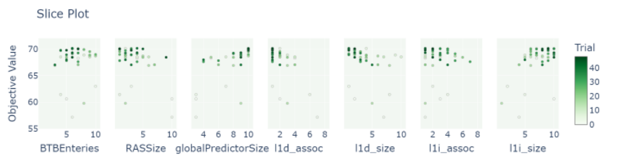

FIGURE 8. **Slice plot showing the effect of seven hyperparameters on IPC improvement while minimizing LUT overhead. The x-axis uses powers of 2, and the y-axis represents IPC (higher is better). Optimal configurations for BTB entries, RAS size, global predictor size, and cache parameters are highlighted. A light-to-dark green gradient indicates increasingly optimized selections.**

Figure 11 presents a detailed analysis of the interplay between cache configurations, branch predictor parameters, and their impact on processor performance and resource utilization. The 3D scatter plots illustrate the relationship between Instructions Per Cycle (IPC) and overall resource usage, highlighting optimization patterns across different architectural configurations. Larger L1 instruction and data cache sizes (`l1i_size` and `l1d_size`) generally improve IPC; however, they introduce diminishing returns and increased LUT consumption. In contrast, cache associativity (`l1i_assoc` and `l1d_assoc`) offers an optimal balance, where lower associativity significantly reduces LUT usage while maintaining competitive IPC. Branch predictor parameters (BTB size, RAS size, and global predictor size) exhibit well-defined optimal ranges, ensuring maximum IPC without excessive LUT utilization. The color gradient from light to dark green in Figure 11 represents progressive optimization iterations, with darker points indicating more refined configurations.

D. TRADEOFF EVALUATION

Tuning cache and branch predictor parameters significantly improves performance, especially in benchmarks with nested control flows. These optimizations reduce cache misses and branch mispredictions, leading to higher instruction throughput. Among micro-benchmarks, branch-heavy workloads like `control_complex` show notable IPC gains due to their sensitivity to predictor accuracy and cache efficiency. In contrast, execution- and dependency-focused micro-benchmarks exhibit negligible changes, as their performance hinges on compute and data dependencies rather than fetch behavior. Memory-bound benchmarks see only marginal improvements, constrained by inherent memory latency rather than cache hit rates.

Benchmarks with simple loops, such as `Susan, JPEG,` and `Bitcount (MiBench)`, show stable IPC regardless of tuning, while those with nested control structures—`Basicmath, Dijkstra, String Search`, and `Qsort`—benefit significantly from improved prediction and caching. These enhancements reduce control and memory stalls, driving substantial IPC gains. Figures 9 and 10 highlight the impact of these architectural optimizations, particularly for workloads with complex control flow and memory access patterns.

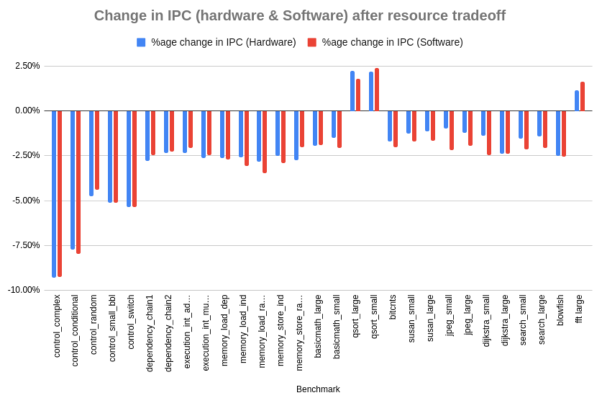

FIGURE 9. **Bar chart showing the percentage change in IPC after integrating the optimized predictor and cache parameters in gem5 model and in hardware using resource optimization objective function.**

TABLE 2. **Hardware resource changes after tradeoff optimizations.**

Hardware Parameter	Resource Tradeoff			Performance Tradeoff		
	Before	After	% Change	Before	After	% Change
Slice LUTs	78768	69316	-12.00%	78768	81919	4.01%
LUT as Logic	76127	66912	-12.11%	76127	79173	4.01%
LUT as Memory	2641	2324	-12.01%	2641	2747	4.02%
LUT as DRAM	2004	1758	-12.28%	2004	2142	6.89%
LUT as Shift Register	637	560	-12.09%	637	663	4.09%
Slice Registers	51558	49854	-3.31%	51558	53621	4.01%
Register as Flip Flops	51542	49840	-3.31%	51542	53604	4.01%
On Chip Power	2.908	2.832	-2.62%	2.908	3.18	9.36%
Dynamic Power	2.724	2.643	-2.98%	2.724	3.01	10.50%
WNS	0.125	0.125	0.00%	0.125	0.125	0.00%

1) RESOURCE OPTIMIZATION TRADE-OFFS

Our resource optimization (parameter list in Table 1) achieved a 12% reduction in LUT usage, with only a 2.07% decrease

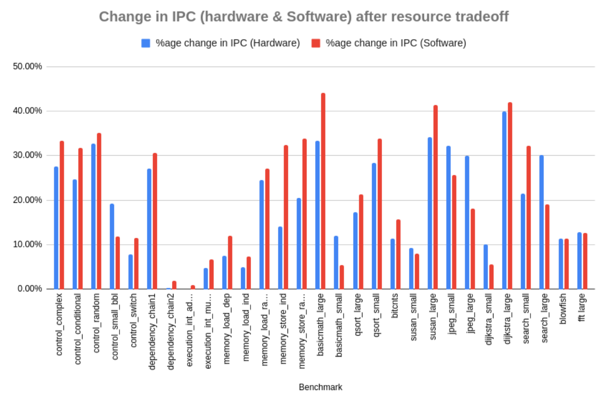

FIGURE 10. **Bar chart showing the percentage change in IPC after integrating the optimized predictor and cache parameters in gem5 model and in hardware using performance optimization objective function.**

in the RMS of benchmark IPCs. Specifically, LUT as Logic and LUT as Memory were reduced by 12.11% and 12.01%, respectively, indicating more efficient logic packing and memory usage. Reductions in LUT as DRAM and LUT as Shift Register (both over 12%) further enhanced area efficiency. Slice Register usage declined by 3.31%, reflecting improved register allocation. Power efficiency also improved, with on-chip and dynamic power consumption decreasing by 2.62% and 2.98%, respectively. Despite these optimizations, WNS remained stable at 0.125 ns, confirming that timing constraints were met.

Table 2 summarizes these results, while Figure 9 shows that control micro-benchmarks experienced the largest IPC drop—up to 9.4%—resulting in an overall 2.67% RMS IPC reduction. However, MiBench workloads such as Dijkstra, Qsort, and FFT exhibited IPC gains that offset losses in other benchmarks, yielding a net 3.37% increase in RMS IPC for MiBench.

2) PERFORMANCE OPTIMIZATION GAINS

Our performance optimizations (parameter list in Table 1) led to a substantial 20.48% average performance improvement across benchmarks (Figure 10), validated on both FPGA hardware and GEM5 simulations, with only a modest 4.2% increase in LUT usage (Table 2). Leveraging the Optuna framework, we explored configuration spaces (Figure 11) that yielded significant IPC gains—exceeding 30% in some cases. Control-oriented benchmarks showed the most notable improvements: `control_random` increased by 32.69%, `control_complex` by 27.59%, and `control_conditional` by 24.61% on hardware. Enhanced cache efficiency enabled `memory_store_random` to gain 32.46% (hardware) and 33.84% (software), while `memory_load_random` improved by 24.57% in hardware due to reduced memory conflicts.

In the MiBench suite, `Dijkstra` accelerated by nearly 22% across both platforms, benefiting from lower memory latency. `Blowfish` (11.34%) and `FFT large` (12.78%)

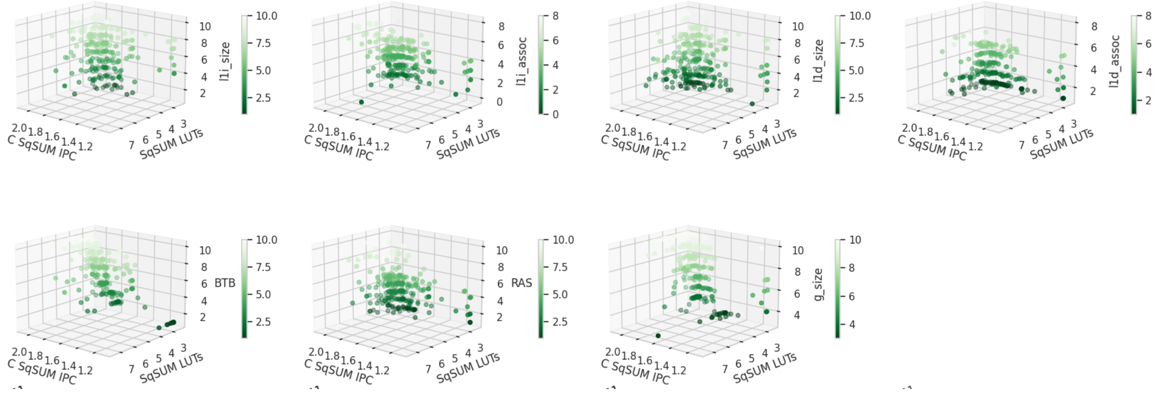

FIGURE 11. 3D scatter plots illustrating the relationships between various cache and branch predictor parameters and their impact on processor performance (IPC) and resource utilization (LUT usage). The top row of plots focuses on cache parameters (L1 instruction and data cache sizes and associativities), while the bottom row examines branch prediction parameters (BTB size, RAS size, and global predictor size). Color gradients represent optimization trials, highlighting key trade-offs between IPC improvement and LUT consumption.

also saw performance boosts, driven by faster bitwise execution and improved branch prediction.

3) POTENTIAL FOR FURTHER OPTIMIZATION

While additional tweaks could boost control benchmarks by up to 29.5% and overall performance by 8.085%, this would require using 9.6% more LUTs—pushing the hardware close to its limits. For instance, `Basicmath` could gain 46% IPC and `Blowfish` 2.5%, raising the average MiBench improvement to 14.18%. However, the high LUT cost makes this impractical. Figure 11 shows the trade-off between performance gains and resource usage. On the Kintex-7 FPGA, we capped optimizations at a 31.88% performance increase to stay under the 30% LUT limit. Final results still delivered strong improvements: `Basicmath` jumped 29.95%, and overall MiBench performance rose 11.76%, though `StringSearch` dipped slightly (1.5%) due to its unique workload demands.

V. CONCLUSION

This paper presents a detailed analysis of the CVA6 System-on-Chip (SoC) architecture and introduces a scalable methodology for design space exploration, leveraging machine learning for performance and resource optimization. Through high-accuracy modeling in gem5, we systematically evaluated CVA6's architectural choices and their impact on efficiency. The integration of machine learning techniques enabled the development of two optimized variants: a resource-efficient model reducing resource usage by 12% with minimal performance loss and a performance-optimized model achieving a 29.5% performance gain with a modest 4.2% increase in resource consumption. These results highlight the potential of machine learning in architectural optimization, providing a framework that can be extended to other SoC designs. The insights from this study contribute to the development of more efficient and high-performance embedded systems, paving the way for next-generation SoCs in IoT, edge computing, and other emerging applications.

REFERENCES

[1] 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," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, May 2011, doi: 10.1145/2024716.2024718.

[2] A. Waterman, Y. Lee, R. Avizienis, H. Cook, D. Patterson, and K. Asanovic, "The RISC-V instruction set," in *Proc. 25th IEEE Hot Chips Symp. (HCS)*, Stanford, CA, USA, Aug. 2013, p. 1.

[3] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2629–2640, Nov. 2019, doi: 10.1109/TVLSI.2019.2926114.

[4] A. Dörflinger, M. Albers, B. Kleinbeck, Y. Guan, H. Michalik, R. Klink, C. Blochwitz, A. Nechi, and M. Berekovic, "A comparative survey of open-source application-class RISC-V processor implementations," in *Proc. 18th ACM Int. Conf. Comput. Frontiers*, May 2021, pp. 12–20, doi: 10.1145/3457388.3458657.

[5] B. Sá, L. Valente, J. Martins, D. Rossi, L. Benini, and S. Pinto, "CVA6 RISC-V virtualization: Architecture, microarchitecture, and design space exploration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 11, pp. 1713–1726, Nov. 2023.

[6] M. Ijaz, F. Saleem, U. Shahid, S. Waheed, and J.-R. Coulon, "Implementation and performance evaluation of bit manipulation extension on CVA6 RISC-V," in *Proc. 20th ACM Int. Conf. Comput. Frontiers*, May 2023, pp. 385–386, doi: 10.1145/3587135.3591439.

[7] U. Shahid, A. Ahmad, and S. Wasim, "Gem5-based evaluation of CVA6 SoC: Insights into the architectural design," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, May 2024, pp. 298–300, doi: 10.1109/ISPASS61541.2024.00037.

[8] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Jul. 2019, pp. 2623–2631, doi: 10.1145/3292500.3330701.

[9] D. Kanter, "RISC-V offers simple, modular ISA," Tech. Rep., 2016, pp. 1–5. [Online]. Available: https://riscv.org/wp-content/uploads/2016/04/RISC-V-Offers-Simple-Modular-ISA.pdf

[10] R. Holler, D. Haselberger, D. Ballek, P. Rossler, M. Krapfenbauer, and M. Linauer, "Open-source RISC-V processor IP cores for FPGAs—Overview and evaluation," in *Proc. 8th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2019, doi: 10.1109/MECO.2019.8760205.

[11] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gurkaynak, and L. Benini, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7864441

[12] T. Marena, "RISC-V: High performance embedded SweRV core microarchitecture, performance and CHIPS alliance," Tech. Rep., Jan. 2019. [Online]. Available: https://riscv.org/wp-content/uploads/2019/04/RISC-V_SweRV_Roadshow-.pdf

[13] O. Chatzopoulos, G. M. Fragkoulis, G. Papadimitriou, and D. Gizopoulos, "Towards accurate performance modeling of RISC-V designs," 2021, *arXiv:2106.09991*.

[14] C. R. Lazo, C. A. Hernández, O. Palomar, O. S. Unsal, M. A. Ramírez, and A. Cristal, "A RISC-V simulator and benchmark suite for designing and evaluating vector architectures," 2021, *arXiv:2111.01949*.

[15] E. Tehrani, T. Graba, A. S. Merabet, and J.-L. Danger, "RISC-V extension for lightweight cryptography," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2020, pp. 222–228, doi: 10.1109/DSD51259.2020.00045.

[16] T. Lu, "A survey on RISC-V security: Hardware and architecture," 2021, *arXiv:2107.04175*.

[17] M. Johns and T. J. Kazmierski, "A minimal RISC-V vector processor for embedded systems," in *Proc. Forum Specification Design Lang. (FDL)*, Feb. 2020, doi: 10.1109/FDL50818.2020.9232940.

[18] *QEMU*. [Online]. Available: https://www.qemu.org/docs/master/about/index.html

[19] P. Adelt, B. Koppelmann, W. Mueller, and C. Scheytt, "A scalable platform for QEMU based fault effect analysis for RISC-V hardware architectures," in *Proc. Methods Description Lang. Model. Verification Circuits Syst. (MBMV)*, Mar. 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9094540

[20] O. Chatzopoulos, G. Papadimitriou, V. Karakostas, and D. Gizopoulos, "Enabling design space exploration of risc-v accelerator-rich computing systems on gem5," in *Proc. RISC-V Summit Eur.*, 2023, pp. 1–2. [Online]. Available: https://riscv-europe.org/summit/2023/media/proceedings/posters/2023-06-08-Odysseas-CHATZOPOULOS-poster.pdf

[21] F. A. Endo, D. Couroussé, and H.-P. Charles, "Micro-architectural simulation of in-order and out-of-order ARM microprocessors with gem5," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Model., Simul. (SAMOS XIV)*, Jul. 2014, pp. 266–273, doi: 10.1109/SAMOS.2014.6893220.

[22] A. A. Abudaqa, T. M. Al-Kharoubi, M. F. Mudawar, and A. Kobilica, "Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with Gem5 simulator," in *Proc. 5th Int. Conf. Electr. Electron. Eng. (ICEEE)*, May 2018, pp. 317–322, doi: 10.1109/ICEEE2.2018.8391354.

[23] I. Wang, P. Chakraborty, Z. Y. Xue, and Y. F. Lin, "Evaluation of gem5 for performance modeling of ARM Cortex-R based embedded SoCs," *Microprocessors Microsyst.*, vol. 93, Sep. 2022, Art. no. 104599, doi: 10.1016/j.micpro.2022.104599.

[24] Q. Huppert, T. Evenblij, M. Perumkunnil, F. Catthoor, L. Torres, and D. Novo, "Memory hierarchy calibration based on real hardware in-order cores for accurate simulation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 707–710. [Online]. Available: https://hal-lirmm.ccsd.cnrs.fr/lirmm-03084343/document

[25] J. V. A. Vieira, M. A. Souza, and H. C. D. Freitas, "Performance evaluation of Intel and AMD memory hierarchies using a simulation-driven approach with Gem5," in *Proc. Anais Estendidos do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD Estendido)*, Oct. 2023, pp. 17–24. [Online]. Available: https://sol.sbc.org.br/index.php/sscad_estendido/article/view/26444/26267

[26] K. P. Srinivasan, "Creating a PCI express interconnect in the gem5 simulator," Ph.D. dissertation, Univ. Illinois Urbana, Champaign, IL, USA, Jul. 2018. [Online]. Available: https://hdl.handle.net/2142/101569

[27] X. Li, Z. Dong, S. Li, S. Gao, J. Jiang, G. He, and Z. Mao, "MUG5: Modeling of universal chiplet interconnect express (UCIe) standard based on gem5," in *Proc. IEEE 15th Int. Conf. ASIC (ASICON)*, Oct. 2023, pp. 1–4. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/2646/1/012026/pdf

[28] C. Gu, "A study on impacts of hardware changes on performance of different multithreading libraries based on Gem5," *J. Phys., Conf. Ser.*, vol. 2646, no. 1, Dec. 2023, Art. no. 012026. [Online]. Available: https://www.proquest.com/docview/2907796679?sourcetype=Scholarly%20Journals

[29] Q. Forcioli, J.-L. Danger, and S. Chaudhuri, "A gem5 based platform for micro-architectural security analysis," in *Proc. 12th Int. Workshop Hardw. Architectural Support Secur. Privacy*, Oct. 2023, pp. 91–99. [Online]. Available: https://dl.acm.org/doi/10.1145/3623652.3623674

[30] PULP Platform. *CVA6 RISC-V Core*. [Online]. Available: https://github.com/openhwgroup/cva6

[31] *CVA6 Requirement Specification—CVA6 Documentation*. [Online]. Available: https://docs.openhwgroup.org/projects/cva6-user-manual/02_cva6_requirements/cva6_requirements_specification.html

[32] *AXI4 Interface Protocol*. [Online]. Available: https://developer.arm.com/

[33] Piton Project. *OpenPiton: An Open Source Manycore Research Framework*. [Online]. Available: http://parallel.princeton.edu/papers/openpiton-asplos16.pdf

[34] B. Costa, L. Valente, J. Martins, D. Rossi, L. Benini, and S. Pinto, "CVA6 RISC-V virtualization: Architecture, microarchitecture, and design space exploration," 2023, *arXiv:2302.02969*.

[35] C. Allart, J. R. Coulon, A. Sintzoff, O. Potin, and J.-B. Rigaud. (2023). *Performance Modeling of CVA6 With Cycle-Based Simulation*. [Online]. Available: https://riscv-europe.org/summit/2023/media/proceedings/posters/2023-06-07-C%C3%B4me-ALLART-poster.pdf

[36] P. Ravenel, A. Perais, B. De Dinechin, and F. Pétrot, "A gem5-based CVA6 framework for microarchitectural pathfinding," presented at the RISC-V Summit Eur., Jun. 2023. Accessed: Aug. 28, 2023. [Online]. Available: https://riscv-europe.org/media/proceedings/posters/2023-06-06-Pierre-RAVENEL-abstract.pdf

[37] Xilinx. (2019). *Vivado Design Suite: User Guide*. [Online]. Available: https://www.xilinx.com

[38] 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," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul. 2006, doi: 10.1109/MM.2006.82.

[39] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *ACM SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 92–99, Nov. 2005, doi: 10.1145/1105734.1105747.

[40] S. H. Nikounia and S. Mohammadi, "Gem5v: A modified gem5 for simulating virtualized systems," *J. Supercomput.*, vol. 71, no. 4, pp. 1484–1504, Feb. 2015, doi: 10.1007/s11227-014-1375-7.

[41] T. Ta, L. Cheng, and C. Batten, "Simulating multi-core RISC-V systems in gem5," presented at the 2nd Workshop Comput. Archit. Res. RISC-V, Los Angeles, CA, USA, Jun. 2018. Accessed: Aug. 25, 2023. [Online]. Available: https://www.csl.cornell.edu/cbatten/pdfs/ta-gem5-riscv-carrv2018.pdf

[42] *Shakti Core-model-gem5 GitLab*. [Online]. Available: https://gitlab.com/shaktiproject/tools/core-models-gem5

[43] Y. S. Shao and D. Brooks, "ISA-independent workload characterization and its implications for specialized architectures," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2013, pp. 245–255. [Online]. Available: https://ieeexplore.ieee.org/document/6517333

[44] A. Saidi, J. Hestness, D. Wentzlaff, D. Black-Schaffer, D. Benedict, D. Barton, and D. Barton, "Enabling reproducible and agile full-system simulation," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2020, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/document/982917

[45] *MiBench*. [Online]. Available: https://github.com/embecosm/mibench

[46] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. 4th IEEE Int. Workshop Workload Characterization*, Dec. 2001, pp. 3–14. [Online]. Available: https://vhosts.eecs.umich.edu/mibench/Publications/MiBench.pdf

[47] S. Krishnan, A. Yazdanbakhsh, S. Prakash, J. Jabbour, I. Uchendu, S. Ghosh, B. Boroujerdian, D. Richins, D. Tripathy, A. Faust, and V. J. Reddi, "ArchGym: An open-source gymnasium for machine learning assisted architecture design," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, Jun. 2023, pp. 1–16.

[48] GitHub. (Jul. 19, 2023). *Cv64a6_imafdc_sv39_config_pkg.sv*. [Online]. Available: https://github.com/openhwgroup/cva6/blob/716d21c4243c8796f7ff7722d01f1a6c65ad40fa/core/include/cv64a6_imafdc_sv39_config_pkg.sv

[49] N. N. Kabylkas. (Oct. 2022). *Improving Effectiveness and Productivity of Microprocessor Verification*. Accessed: Aug. 28, 2023. [Online]. Available: https://escholarship.org/content/qt83t7t9hk/qt83t7t9hk_noSplash_312ff4620c77cc3b71b75872f00d1c3d.pdf
[50] *OpenHW Group*. Accessed: Aug. 28, 2023. [Online]. Available: https://docs.openhwgroup.org/_downloads/cva6-user-manual/en/cv32a6_v0.1.0/pdf/
[51] *RISC-V Foundation*. [Online]. Available: https://riscv.org/about/history/
[52] M. A. Z. Alves, C. Villavieja, M. Diener, F. B. Moreira, and P. O. A. Navaux, "SiNUCA: A validated micro-architecture simulator," in *Proc. IEEE 17th Int. Conf. High Perform. Comput. Commun. 7th Int. Symp. Cyberspace Saf. Secur., IEEE 12th Int. Conf. Embedded Softw. Syst.*, Aug. 2015, pp. 605–610.
[53] *CVA6 Requirements Specification*, OpenHW Group.

UMER SHAHID (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the University of Engineering and Technology (UET), Lahore, Pakistan, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree. From 2015 to 2017, he was a Teaching Assistant with the Department of Electrical Engineering, Lahore. Since 2019, he has been a Lecturer with the Department of Electrical Engineering, UET. He is the author of seven articles. His research interests include RISC-V compliance testing, performance enhancement of RISC-V based SoCs. He is a Member Technical Staff (Level-II) with 10xEngineers. He is the Vice-Chair of Special Interest Group of Architecture Compliance (SIG-ARCH) with RISC-V Internationals. He has been the Volunteer Manager with HotChips Conference, since 2023.

MUHAMMAD TAHIR (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Illinois Chicago, Chicago, IL, USA, in 2008. He was a Visiting Research Scholar with the University of Illinois Chicago, in the summer of 2014. He is currently a Professor with the Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan. He has more than 60 international publications to his credit. He was a principal investigator of numerous funded research projects related to embedded systems and intelligent surveillance networks. His research interests include distributed resource optimization for wireless networks, real-time wireless multimedia networks, and computer architecture. He is a reviewer of numerous IEEE journals and conferences.

BILAL ZAFAR (Member, IEEE) received the B.Sc. degree in electronics engineering from the GIK Institute of Engineering Sciences and Technology, and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles. He is the Co-Founder and CEO of 10xEngineers, a design and verification services company based, Lahore, Pakistan. 10xEngineers specialized in design and verification of RISC-V processors, IPs and SoCs, and image processors hardware. Started in 2021, 10xEngineers today has a team of 80 engineers. 10xEngineers is a RISC-V International Development Partner and is the first official RISC-V Laboratory Partner. Previously, he was a Principal Engineer with Qualcomm, where he co-led a world-wide team of about 60 engineers working on developing custom & semi custom IPs.

AYESHA AHMAD is currently pursuing the bachelor's degree in electrical engineering, from the University of Engineering and Technology, Lahore. Her research interests include embedded programming and computer hardware synthesis.

SHANZAY WASIM is currently pursuing the bachelor's degree in electrical engineering, from the University of Engineering and Technology, Lahore. Her research interests include embedded programming and computer hardware synthesis.

BISAL SAEED is currently pursuing the bachelor's degree in electrical engineering, from the University of Engineering and Technology, Lahore. Her research interests include embedded programming and computer hardware synthesis.