Workload-Driven Optimization of RISC-V Core Configurations for Emerging Applications

1st Ryan Frost
Bradley Dept. of ECE
Virginia Tech
Blacksburg, USA
rfrost26@vt.edu

2nd Enrique Maldonado
Bradley Dept. of ECE
Virginia Tech
Blacksburg, USA
enriquem@vt.edu

3rd Wangzhi Zhan
Bradley Dept. of ECE
Virginia Tech
Blacksburg, USA
wzhan24@vt.edu

***Abstract*—The large-scale use of highly repetitive algorithms in fields such as machine learning and signal processing creates an opportunity to improve efficiency with processor cores designed for specific types of workloads rather than the traditional general purpose processor core. This project addresses the systematic categorization of applications based on execution needs and connects those needs to specific processor configurations for optimal performance. Using RISC-V instruction set architecture and Verilator simulations, we will analyze a number of trial configurations for each defined category of workloads, and we will optimize one specific configuration for each category. The effectiveness of these optimized configurations will be tested by running both category-specific benchmarks and generic benchmarks. This will show how optimizing for specific workloads will improve efficiency for those categories over a generic core.**

***Index Terms*—architecture, RISC-V, Verilator, optimization, benchmarks.**

I. Introduction

The increasing diversity of computational workloads drives a need for more specific processor cores tailored to a specific category of application. From machine learning to signal processing, new algorithms are being developed that have drastically different execution needs. The typical generic processor is becoming increasingly outdated for these specific tasks. The performance and power metrics for generic chips are falling behind newly designed specialized chips, as chip designers and software engineers try to extort every bit of efficiency.

A. Project Goal

The main justification for this project is the need for a systematic method to connect new and emerging applications and the many different configurable parameters of a processor core. Optimizing for generic benchmarks is not enough anymore, as they fail to represent the unique execution patterns used in specific use cases.

This project seeks to address this challenge. By systematically categorizing new algorithms from a variety of different areas, we will create specific classes based on exception needs. For each category, we will narrow down a fine-tuned configuration that will work best for that specific type of application. This will highlight the importance of blending software design with hardware design, and show how processor parameters can make a big difference in the latest algorithm research.

II. Related Work

Newly proposed algorithms across the software engineering fields have yet to be analyzed on specific processor configurations. Existing algorithms are also showed to drastically depend on processor configurations for the best performance. One area of recent algorithm innovation is in real time systems.

A. Novel Real Time Algorithms

Real time algorithms are used by machines and autonomous devices to make quick and accurate decisions. The time sensitive nature of these applications means that processor cores with specific parameters for this task is highly valuable. More executions per second means more complex algorithms can be run in the same amount of time. There is also often an element of energy efficiency in real time systems, as they may be run on off-grid robots or in remote areas. Reducing main memory pulls reduces the energy demand from the main memory bus, resulting in a more energy efficient application.

Four potential workloads related to real time robotics are collision detection, path finding, Fast Fourier transform (FFT), and finite impulse response filters (FIR). A novel collision detection algorithm is proposed and analyzed against existing algorithms by Wang et al. [1]. An extention to Dijkstra's path finding algorithm is proposed for 3D path finding by Luo et al. [2]. Two foundation signal processing algorithms, FFT and FIR, are expanded upon by Zhao et al. and Zhang & Jiang, respectively [3] [4]. These unique workloads may be used to develop processors for better real time robot performance.

B. Vectorization

In addition to newly proposed algorithms, existing algorithms may be adjusted to better use vectorization. The vector extension for RISC-V ISA opens new opportunities for research and implementation. Some main features include the operation of arithmetic/logic and load/store instructions to operate on sets of vectors instead of individual data items, and having a vector register file that can hold a large number of elements. Vector architectures can also include multiple pipelines, leading to advantages in performance and scalability. The vector engine increases the amount of instruction level parallelism by performing functions such as vector renaming, issuing order and queues for instructions, pipeline interconnections, and receiving and managing instructions in the vector memory unit, detailed by C. Ramirez et al. [5]. The RISC-V Vectorized Benchmark suite offers a variety of benchmarks that can be used to test the performance of the architecture implementation.

Vectorization benefits various computational workloads. Secure hash algorithms like SHA-3 functions are used in a number of applications, requiring computational intensive consumption, as demonstrated by H. Li et al. [6]. This is also true for astrophysical applications such as Octo-Tiger, as demonstrated by P. Diehl et al. [?]. Machine learning algorithms can also benefit from vectorization, demonstrated by V. Titopoulos et al. [?]. This is expanded upon in the next subsection.

C. Machine Learning Applications

Recent studies on AI-related applications show that processor design and configuration strongly influence the efficiency of machine learning workloads. J. Kim et al. [?] propose a systolic–vector hybrid accelerator combining systolic arrays and vector processors to handle diverse DNN workloads efficiently. Bhattacharjee et al. [?] evaluate ML inference workloads on RISC-V systems using gem5 and an MLIR-based toolchain, showing that architecture parameters like cache size and pipeline type greatly affect performance. Gómez-Luna et al. [?] analyze memory-centric systems, finding that memory-bound ML algorithms such as decision trees and K-Means gain large speedups when data movement is minimized. C. Xu et al. [?] develop X-SET, a graph mining accelerator that mitigates irregular memory access patterns, while Y. Xiao et al [?] introduce GAHLS, a compiler-assisted hardware synthesis framework that maps LLVM IR graphs into heterogeneous accelerators for AI and graph analytics. Together, these works demonstrate that AI and graph workloads have diverse compute and memory demands, reinforcing our focus on exploring parameter-level optimization on general-purpose RISC-V cores for different application categories.

III. Proposed Method

Our method systematically explores how key architectural parameters influence the performance of different application types and identifies optimized configurations for each category. We begin with the baseline RISC-V Rocket core and vary seven architectural parameters: cache size and associativity, pipeline depth, floating-point unit configuration, virtual memory settings, TLB size, branch predictor type, and clock frequency balance.

First, we vary these seven parameters individually for each application to measure how each one impacts performance metrics such as CPI and cache hit rate. This analysis will reveal which parameters are most critical for each application's efficiency. Next, we categorize the applications based on their sensitivity to these parameters. Applications that show similar performance trends under parameter changes will be grouped into the same category.

For each category, we will design a specific configuration that combines the most beneficial parameter settings for that group of workloads. We will then run these specialized configurations across all applications within the category to evaluate their performance and consistency. Afterward, we will analyze the systematic results from all applications to identify trends and confirm the robustness of each configuration. Based on this analysis, we will derive a final optimized configuration that performs best on average within each category.

Finally, we will compare all category-specific configurations against the baseline using benchmark programs such as matrix multiplication. This comparison will quantify the benefit of specialization, demonstrating how tuning architectural parameters for specific workloads can yield higher performance and efficiency than a one-size-fits-all processor design.

IV. Evaluation Plan

This study will focus on the RISC-V Rocket Configuration and how it can be optimized to have better performance on different high performance computing applications. A wide range of applications were chosen including: Machine Learning, Signal Processing, Graph Analytics, Robotics, Astrophysics and Real-Time System Applications. Seven core configuration parameters will be fine-tuned in order to determine the best overall architecture for six applications. These parameters include cache size and associativity, pipeline architecture, virtual memory, and floating point unit, translation lookup buffer and branch predictor configurations. We will test the optimized configurations using the matrix multiplication benchmark. The evaluation metrics will be Cycles Per Instruction (CPI) and Cache Hit-Rate.

V. Timeline

The timeline for the project begins November 5th and ends December 17th. There are 9 milestones identified to complete the project on time. A review period is included to receive feedback for a more polished final draft. Changes to the timeline will be documented for review in the final paper.

- 11/05 - 11/14 Get applications in a runnable state
- 11/05 - 11/14 Create pipeline for testing multiple configurations for multiple applications at once
- 11/14 - 11/19 Systematically categorize the applications based on parameter impact
- 11/19 - 11/21 Design multiple trial configurations for each identified category
- 11/21 - 12/03 Analyze trial configurations results and design most optimal configuration for each specific category
- 12/03 - 12/08 Compare with standardized benchmarks for each category and document overall performance between final configurations
- 12/08 - 12/10 Write paper draft
- 12/10 - 12/14 Receive feedback on draft
- 12/14 - 12/17 Finalize paper draft

References

[1] B. Wang, Z. Ferguson, T. Schneider, X. Jiang, M. Attene, and D. Panozzo. 2021. A Large-scale Benchmark and an Inclusion-based Algorithm for Continuous Collision Detection. ACM Trans. Graph. 40, 5, Article 188 (October 2021), 16 pages. https://doi.org/10.1145/3460775

[2] M. Luo, X. Hou and J. Yang, "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm," in IEEE Access, vol. 8, pp. 147827-147838, 2020, doi: 10.1109/ACCESS.2020.3015976.

[3] X. Zhang and S. Jiang, "Application of Fourier Transform and Butterworth Filter in Signal Denoising," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2021, pp. 1277-1281, doi: 10.1109/ICSP51882.2021.9408933.

[4] S. Zhao, Y. S. Shmaliy, C. K. Ahn and F. Liu, "Self-Tuning Unbiased Finite Impulse Response Filtering Algorithm for Processes With Unknown Measurement Noise Covariance," in IEEE Transactions on Control Systems Technology, vol. 29, no. 3, pp. 1372-1379, May 2021, doi: 10.1109/TCST.2020.2991609.

[5] Cristóbal Ramírez, César Alejandro Hernández, O. Palomar, Osman Ünsal, Mario Alberto García-Ramírez, and A. Cristal, "A RISC-V Simulator and Benchmark Suite for Designing and Evaluating Vector Architectures," ACM Transactions on Architecture and Code Optimization, vol. 17, no. 4, pp. 1–30, Nov. 2020, doi: https://doi.org/10.1145/3422667.

[6] H. Li, N. Mentens and S. Picek, "Maximizing the Potential of Custom RISC-V Vector Extensions for Speeding up SHA-3 Hash Functions," 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2023, pp. 1-6, doi: 10.23919/DATE56975.2023.10137009.

[7] P. Diehl et al., "Preparing for HPC on RISC-V: Examining Vectorization and Distributed Performance of an Astrophysics Application with HPX and Kokkos," ARXIV, pp. 1656–1665, Nov. 2024, doi: https://doi.org/10.1109/scw63240.2024.00207.

[8] V. Titopoulos, K. Alexandridis, and G. Dimitrakopoulos, "Vectorized FlashAttention with Low-cost Exponential Computation in RISC-V Vector Processors," arXiv.org, 2025. https://arxiv.org/abs/2510.06834 (accessed Oct. 29, 2025).

[9] J.-H. Kim, S. Yoo, S. Moon, and J.-Y. Kim, "Exploration of Systolic-Vector Architecture with Resource Scheduling for Dynamic ML Workloads," arXiv.org, 2022. https://arxiv.org/abs/2206.03060

[10] D. Bhattacharjee, Anmol, T. Marinelli, K. Pathak, and P. Kourzanov, "Full-stack evaluation of Machine Learning inference workloads for RISC-V systems," arXiv.org, 2024. https://arxiv.org/abs/2405.15380 (accessed Oct. 29, 2025).

[11] J. Gómez-Luna et al., "Evaluating Machine LearningWorkloads on Memory-Centric Computing Systems," 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, NC, USA, 2023, pp. 35-49, doi: 10.1109/ISPASS57527.2023.00013.

[12] C. Xu, T. Shi, S. Sun, J. Zhai, and X. Chen, "X-SET: An Efficient Graph Pattern Matching Accelerator With Order-Aware Parallel Intersection Units," ACM Digital Library, pp. 1505–1519, Oct. 2025, doi: https://doi.org/10.1145/3725843.3756112.

[13] Y. Xiao, S. Nazarian, and P. Bogdan, "GAHLS: an optimized graph analytics based high level synthesis framework," Scientific reports, vol. 13, no. 1, Dec. 2023, doi: https://doi.org/10.1038/s41598-023-48981-x.