# Workload-Driven Optimization of RISC-V Core Configurations for Embedded Applications

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

2rd Wangzhi Zhan
*Bradley Dept. of ECE*
*Virginia Tech*
Blacksburg, USA
wzhan24@vt.edu

***Abstract*—General purpose proecssors often take up too much space and too much power for specific embedded applications that require only a fraction of the provided resources. This project utilizes a new design pipeline to systematically test the RISC-V Berkeley Out-of-Order Machine (BOOM) on specific embedded workloads. The BOOM core's parameters are changed on a granular level, manipulating frontend widths, scalar widths, reorder buffer sizes, cache size and associativity, translation lookaside buffer, and branch predictors. The four workloads tested are Dijkstra's Algorithm, Advanced Encryption Standard (AES), Fast Fourier Transform (FFT), and Huffman Encoding.**

**The key metric analyzed is Instructions per Cycle (IPC) over Gate Count, which represents performance per chip area that is relevant for embedded applications. IPC is calculated using Chipyard and Verilator, and Gate Count is calculated using Yosys synthesis. Our results indicate that larger chip sizes have diminishing return, and that a smaller, more specific configuration has the best performance per area. This shows that targeted hardware design can lower chip area while maintaining similar performance.**

***Index Terms*—embedded, BOOM, Yosys, IPC, Gate Count**

## I. Introduction

Embedded devices are becoming ever more prevelent in the modern age. These devices often have unique workloads, and the size and power efficiency of the chips for these devices is often just as important as the actual performance. General purpose processors typically focus less on power and space, opting for more advanced branch predictors, larger issue widths, and larger caches. However, these processors can result in extra resources that are not fully utilized by more simple embedded workloads. A processor with more silicon area typically correlates to greater power consumption, which can be detrimental to a constrained embedded environment.

The modularity of the RISC-V BOOM core presents an opportunity to analyze the space efficiency of specific chip configurations. Designing the most opitmal design is challenging because of the interconnected dependencies of the core parameters. Increasing one aspect of the chip has cascading effects on how well the other resources are used.

In this project, we evaluate the trade-offs between different chip configurations and their impact on the area efficiency. We use UC Berkeley's BOOM core and Yosys to systematically evaluate performance using IPC and are using Gate Count. This metric informs our decision on the most area efficient configuration for embedded workloads.

### A. Workloads

- **Dijkstra's Algorithm**: Used in robotics and autonomous vehicles for pathfinding and decision making.
- **Advanced Encryption Standard (AES)**: Used in security and cryptography for secure data transmission.
- **Fast Fourier Transform (FFT)**: Used in digital signal processing for sensor data analysis.
- **Huffman Encoding**: Used to compress data before transmission between devices

### B. Contributions

- **Systematic Design Analysis**: Seven different core parameters are manipulated and the resulting change in performance and area is analyzed.
- **Pipeline Automation**: Candidate design configurations are easily created and supported by a modular scala file and Yosys synth file.
- **Workload-Driven Optimization**: Specific workloads are used to validate performance for different embedded uses.

## 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. [7]. Machine learning algorithms can also benefit from vectorization, demonstrated by V. Titopoulos et al. [8]. 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. [9] propose a systolic–vector hybrid accelerator combining systolic arrays and vector processors to handle diverse DNN workloads efficiently. Bhattacharjee et al. [10] 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. [11] 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. [12] develop X-SET, a graph mining accelerator that mitigates irregular memory access patterns, while Y. Xiao et al [13] 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 and Test in Europe Conferencea and 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.