Proposal to Optimize RISC-V BOOM Core for Specific Applications

Aniket Adhikari
Department of Computer Science
Virginia Tech
Virginia, U.S.

***Abstract*—Given the varying needs for instruction sets in different industries, there are a number of ways in which their implementations need to be parameterized. The Berkley-Out-of-Order Machine (BOOM) core is an implementation of the RISC-V ISA that offers a number of different parameters to be altered. Other papers talk about altering parameters for different purposes like efficiency or performance. This paper proposes a project to fine-tune a well-rounded configuration for a BOOM core by evaluating CPI, throughput, and memory access time upon benchmarks like Quick Sort and Dhrystone.**

***Index Terms*—RISC-V, Computer Organization, Out-of-Order Execution**

I. Introduction

Global demand for microprocessors continues to increase, likely as a result of growing acceptance of smartphones and tablets as well as implementation of cloud computing. Additionally, industries such as automotive, banking, aerospace and defense, medicine, and others are contributing to this growing demand [5].

New ISAs are created and implemented, such as RISC-V, to fit the requirements of various industries, whether it be cost, energy, or performance. The Berkeley Out-of-Order Machine (BOOM) is a high-performance, synthesizable superscalar core implementation of the RISC-V ISA. There are a number of different parameters that can be manipulated, such as branch prediction, number of instructions fetched (or decoded) per cycle, number of registers for different operations, and more [7] [8], giving it strong flexibility for optimization. The parameters of the BOOM core would be altered for reasons such as improved performance, efficient resource usage, improved scalability, cost reduction, or reduced energy consumption. However, the requirements for optimization generally need to be holistic, meaning it has a mix of improvements.

Overall, It is critical to modify configurations of ISA cores, such as the BOOM core, in order to meet specific requirements of diverse computing applications and environments. For my project, I am proposing to create an optimized BOOM core configuration for general applications through fine-tuning of parameters and settings.

II. Related Work

A. Creation and Implementation of the BOOM Core

The technical report for the BOOM core [3] explains why the superscalar core is necessary for advancement in computer architecture research as well how it will be used for education, research, and industry. Through use of the RISC-V ISA and Chisel hardware construction language, the BOOM core is industry-competitive in terms of branch prediction and can be parameterized for different workflows. As we know in ECE 5504, UC Berkeley has provided the open-source Rocket-chip System-on-a-chip (SoC) generator so that we can use BOOM and other cores in our class. Towards the end of the report, there is a section about future work that talks about users and classes like ours being able to configure BOOM in millions of unique ways. However, it is important to be careful when actually deciding on what design configurations to pursue because of how expensive it can be to determine configurations for end-users. They also mention how performance improvements in BOOM can be explored, such as data-prefetching or more aggressive instruction fetch (IF) unit.

B. BOOM Core Micro-architecture Design Space Exploration Framework

This paper [2] begins by talking about the struggle of fine-tuning micro-architecture designs in order to achieve a balance between power and performance. It proposes BOOM-Explorer, which leverages an active learning algorithm called MicroAL to search for Pareto-optimal micro-architecture designs of BOOM and decide which design effectively balances power and performance needs.

Essentially, this paper automates the process of what I'm proposing to do, which is to find an optimal configuration of BOOM through modification of its many parameters. They modify various parameters such as the number of physical integer registers (IntPhyRegister), associativity of the L1 cache (ICacheWay), and much more. This paper provides a great starting point for my project since they give information about parameters in BOOM and pseudo-code for their BOOM-Explorer algorithm. 25 benchmarks are used to thoroughly compare performance and power gains going from a generic BOOM configuration to Pareto-optimal design. An interesting point was brought up about how time-consuming it can be to design, implement, test, and verify different configurations of the BOOM processor. This is one of my main concerns for this project, but I think this paper as well as my knowledge about certain micro-architecture design choices can help aid my process in finding an optimal configuration.

C. *Energy-Focused Instruction Sets*

Jong-eun Leet, Kiyoung Choit, and Nikil D. Dutt [6] propose a technique to enhance energy-efficiency of Application Specific Instruction-set Processors (ASIPs) through the optimization of instruction encoding, while considering instruction width and count. In their study, they approached synthesis of the instruction set strictly from the lens of energy consumption, which leads to a trade-off of overall performance. For embedded systems like household appliances and devices, power consumption is a key metric since consumers might be interested in reducing financial burden. On the other hand, it would not be useful for something like a desktop PC users require high performance output.

III. Proposed Method

This sections talks about how I plan to leverage techniques in class lectures and assignments to optimize the BOOM core for general applications.

A. *Adjust Micro-architecture Parameters*

Manipulating parameters such as issue width, size of instruction window, number of functional units, and more can potentially improve the BOOM core's ability to handle multiple out-of-order instructions. The BOOM core documentation lists the top-level parameters that can be manipulated. We've looked at the BOOM core configurations in Part II of Homework 6 when we compared cycles per instruction (CPI) of the RocketConfig, SmallBoomConfig, and LargeBoomConfig.

B. *Configure Memory Hierarchy*

Configuration of cache size, associativity and replacement policies in the memory hierarchy can minimize memory access latency and/or miss rate. This is something we've already touched on in Part II of Homework 5 when we compared CPI and hit rate of unique cache size and replacement policy configurations. Additionally, we experimented with cache parameters and analyzing the impact on the performance in part II of Homework 3

C. *Configure Branch Prediction*

Configuring the branch predictor to accurately predict outcomes from branch instructions can minimize the impact on CPI that result from mispredicted branches. This is something we're currently talking about in Homework 7, as we configure the Small BOOM core with and without custom branch predictors, then subsequently measuring their respective CPI on previously used benchmarks such as Qsort and Dhrystone.

IV. Evaluation Plan

Below I have included the benchmarks I want to use as well as their overall purpose in the context of the project.

A. *Benchmarks*

1) Qsort: Qsort, or Quick Sort, is a function that implements a sorting algorithm that compares pairs of elements [1]. It will be used to evaluate the performance of the configuration by measuring the time taken to sort an arbitrary set of data.

2) Dhrystone: Dhrystone is a benchmark intended to measure the efficiency and performance of computer systems, specifically on integer computing capabilities [9]. This benchmark has a place in this project since it is readily available to us as students but it being almost 40 years old forces me to consider other benchmarks in addition to Dhrystone.

3) CoreMark: CoreMark is a free-to-use benchmark that measures the performance of micro-controllers and CPUs [4]. It tests operations that commonly appear in different applications such as list processing, matrix manipulation, state machine, and cyclic redundancy check.

B. *Metrics*

Additionally, I will be using the following measurements to ensure that the configurations are well-rounded in terms of latency and bandwidth.

1) Cycles per instruciton (CPI): CPI is a crucial performance indicator to evaluate the efficiency of a processor in executing instructions. Low CPI generally equates to better performance. I will be comparing CPI of different configurations of the BOOM core.

2) Throughput: Throughout measures the amount of work done in a specific period. It looks at the number of instructions executed per unit of time, which helps in evaluating the overall processing capacity of the system

3) Memory Access Time: Memory access time refers to how long it takes to perform read or write operations to the system's memory hierarchy.

V. Timeline

A. *Week 1 (Nov 1 - Nov 7)*

The first week of the project will be spent evaluating feedback from instructors. Additionally, I will attend office hours to ask the professor about potential alterations or additions to my evaluation metrics. Following this, I will research parameters of ISAs that effect my evaluation metrics.

B. *Week 2-3 (Nov 8 - Nov 21)*

The next two weeks will be spend applying configurations in code and running it on my local machine. Code will be stored in a GitHub repository, where milestones and issues will also be tracked. I also have to make sure that the benchmarks I listed earlier function in Visual Studio Code.

C. *Week 4 (Nov 22 - Nov 28)*

The fourth week will be spend running my configurations against the benchmarks and making observations.

D. *Week 5-6 (Nov 29 - Dec 12)*

I will spend the last two weeks on the final presentation and paper.

References

[1] QSort Function | CPlusPlus Reference.

[2] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong. BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, Munich, Germany, Nov. 2021. IEEE. ISSN: 1558-2434.

[3] C. P. Celio. *A Highly Productive Implementation of an Out-of-Order Processor Generator*. PhD thesis, EECS Department, University of California, Berkeley, Dec. 2018.

[4] S. Gal-On and M. Levy. Exploring CoreMark™ – A Benchmark Maximizing Simplicity and Efficacy.

[5] J. John. Global Microprocessor Market Will Reach USD 8,894 Million By 2025: Zion Market Research. May 2019.

[6] J.-e. Lee, K. Choi, and N. Dutt. Energy-efficient instruction set synthesis for application-specific processors. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.*, pages 330–333, Seoul, South Korea, 2003. IEEE.

[7] R. of University of California. Welcome to RISCV-BOOM's documentation! — RISCV-BOOM documentation, 2019.

[8] R. of University of California. RISC-V BOOM, 2020.

[9] A. R. Weiss. Dhrystone Benchmark: History, Analysis, "Scores" and Recommendations, Oct. 2002.