Benchmarking the Wally RISC-V Processor

Author: Abraham Castaneda | castaa7@unlv.nevada.edu
Mentor: Dr. Sarah Harris
REU Smart Cities Summer Research Program

1. Introduction

Significance:

- A **processor** acts as the **brain of a computer** by running software programs.
  - Examples: Intel Core i9, AMD Ryzen 5000 series, AMD A-10 series
- **RISC-V**, the first widely used, open-source processor architecture, was released in 2015. Like open-source software, its ease of use, lower cost, and shorter development time paved the way for…

The RISC-V Wally Processor:

As an ongoing collaboration between UNLV, HMC, and OSU, we are designing a configurable, low-power, open-source RISC-V processor with the following **features**:

- Pipelined architecture
- Set associative data cache
- Virtual memory
- RV64I and RV32I with RVM (multiply/divide), RVF/D (floating-point/ double) and RVC (compressed) extensions

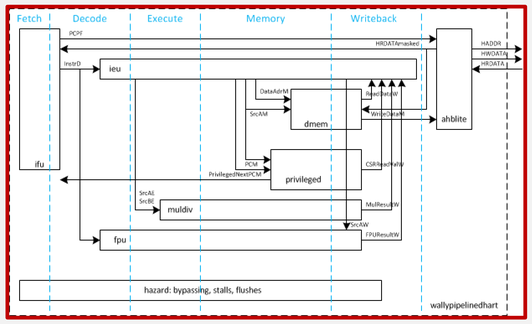

Figure 1. The Wally RISC-V 5-stage pipelined processor

Objective:

- The aim of this project is to use the **CoreMark** benchmark to **evaluate** the Wally RISC-V processor, **compare** it to existing commercial processors, and **improve** its performance.

2. What is CPU benchmarking?

Benchmarks determine processor performance by running **programs that exercise the hardware**. This enables comparison of different processors. **Real-world benchmarks** consist of commercially used programs; whereas **synthetic benchmarks** are code developed for the benchmark itself.

Benchmark	Program Type	Description
Dhrystone	Synthetic	Made to evaluate general purpose performance
CoreMark	Synthetic	Similar to Dhrystone, but improved with more tests
Embench	Real-world	Targeted to embedded processors and runs 19 real programs

Figure 2. A list of sample benchmarks and their characteristics

CoreMark Program Operations:

- Find and sort using **linked lists** to test memory system
- **Matrix manipulations** (matrix addition and multiplication) to test the multiplier/adder.
- **State machine** (determines if an input stream contains valid numbers) to test branch logic.
- **CRC** (cyclic redundancy checks), exercises general processor

3. Method: CoreMark

CoreMark was chosen for its **wide use** among existing commercial processors and its **porting** capabilities (essential for porting to RISC-V).

Running Coremark:

1. Compile program using gcc
2. Create memory file to load into processor
3. Simulate with ModelSim

Challenges:

- Cutting simulation duration down to a reasonable time
- Configuring and debugging counters (MINSTRET and MTIME)

Pros	Cons
❑ Historical data for other processors is widely available ❑ Simple, makes no assumptions about architecture	❑Does not test memory enough ❑Synthetic, so not a completely realistic workload

CoreMark®
An EEMBC Benchmark

4. Benchmarking Results

CoreMark Scoring:

CoreMark runs the benchmark for *N* iterations. The *CoreMark Score* is the number of iterations completed per second. ***CoreMarks/MHz*** (CM/MHz) is the CoreMark Score divided by the processor clock frequency in MHz.

Current Performance: 2.42 CM/MHZ
1.21 Cycles/Instruction (CPI)
Before cache implementation: 1.69 CM/MHZ
1.74 Cycles/Instruction (CPI)

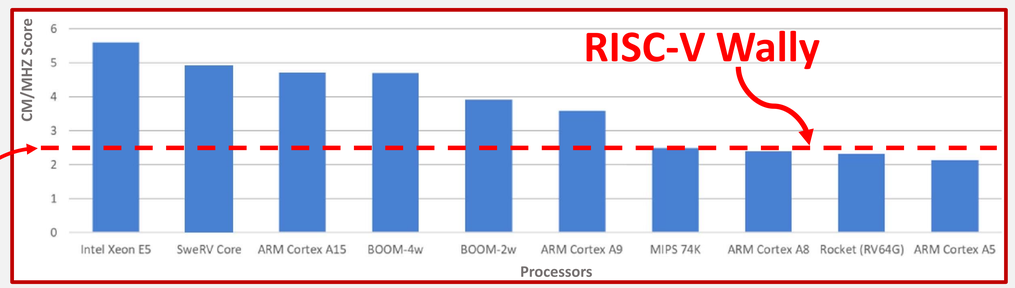

Figure 3. CM/MHz for Wally and other processors

```
# seedcrc           : 0xe9f5
# [0]crclist        : 0xe714
# [0]crcmatrix      : 0x1fd7
# [0]crcstate       : 0x8e3a
# [0]crcfinal       : 0xfcaf
# Correct operation validated. See REA
# ELAPSED TIME: 41
# CoreMark 1.0 : 242 / GCC10.2.0 -O3 -
```

Figure 4. CoreMark Program Output. The program outputs the CoreMark Score (number of iterations per second), which is highlighted. We then calculate 2.42 CM/MHz by dividing the CoreMark Score by the clock frequency (100 MHz). All simulations are run using ModelSim.

5. Conclusions

- We used the CoreMark benchmark to evaluate the performance of the RISC-V Wally processor to **detect bugs** in the system, and to **improve the performance** through bug fixes and microarchitectural optimizations.
- We **successfully ran** the benchmark by fixing bugs, such as those found in the performance counters themselves, in the divide unit, and in the data cache.
- The Wally RISC-V processor currently has a performance of **2.42 CM/MHz**, which is not yet at the performance level of processors such as the SweRV EH1 Core (at 4.9 CM/MHz) or the Andes Technology RISC-V cores (at an average score of 4.42 CM/MHz).
- **Benchmarking will guide** the ongoing development and evaluation of the RISC-V Wally processor.

Future work:

- Analyze **performance bottlenecks** using counters, such as the memory stall performance counter, to improve the processor
- Run **other benchmarks** (for example, Embench)

Embench

Acknowledgements

Dr. Sarah Harris
Dr. David Harris
Dr. James Stine
REU: Smart Cities, Grant #1950872
Elizabeth Hedenburg
Kaveh Pezeshki
Ben Bracker
Ross Thompson