Exploiting RISC-V Hint Instructions for Lightweight VLIW Execution

Leo Marek[1]*, Gregory Chekler[1]*, Jing Jing Chen[1]*, Elliot Chae[1]*,
Ethan Carr[1]* and Ray Simar[1]
[1]Department of Electrical and Computer Engineering, Rice University, Houston, USA
Email: {lnm7, gsc8, jc165, ejc12, ehc6, ray.simar}@rice.edu
*Equal Contribution

***Abstract*—The pursuit of high Instruction-Level Parallelism (ILP) at lower power has renewed interest in Very Long Instruction Word (VLIW) architectures. Yet, conventional VLIW designs often face challenges such as code density and lack of binary compatibility. This paper introduces a novel hint-based VLIW implementation built on the RISC-V Instruction Set Architecture (ISA). Our proposal utilizes architecturally reserved HINT instructions to encode static scheduling decisions, enabling parallel execution without the need for complex hazard detection hardware.**

We evaluated our architectural extensions using the Google MPACT simulator and implemented them at the Register Transfer Level (RTL) by modifying the OpenHW Group CVW (Wally) core. The design was verified using Questa and Verilator and is being synthesized and deployed on an FPGA platform. Performance of the design was assessed using a suite of handwritten DSP benchmarks—specifically FFT, IIR, FIR, and Dot Product kernels—representing a subset of the Embench DSP test suite. Results show that this approach delivers substantial speedup on DSP workloads while preserving full binary cross compatibility meaning libraries can be written for STARBUG and flexibly linked directly with other open ended code.

I. Introduction

Modern embedded processors face a difficult trade-off between performance and energy efficiency. Superscalar architectures extract high Instruction-Level Parallelism (ILP) through dynamic hardware scheduling, but the required speculation, dependency checking, and hazard detection introduce substantial area and power overheads. In this work, we propose a hybrid approach that leverages the RISC-V "HINT" instructions. These instructions are interpreted as NOPs on unmodified RISC-V cores but can carry explicit dependency information for our modified microarchitecture, enabling VLIW-style parallel issue without incurring the complexity of traditional dynamic scheduling.

II. Methodology

Our design methodology progressed from high-level ISA modeling to hardware realization on FPGA.

A. ISA Validation

We began by modeling the proposed HINT-based extensions using Google's MPACT simulator. This allowed us to validate the decoding logic and bundle formation strategy before committing to RTL.

B. RTL Implementation

For the hardware implementation, we utilized the OpenHW Group CVW (Wally) core as our baseline. We modified the fetch and decode stages of Wally to recognize the scheduled bundles. We implemented a 12 output 4 input register file to enable 4 integer datapaths and thus 4 wide VLIW bundles.

C. Verification and Emulation

The modified RTL was simulated using both Questa and Verilator to ensure functional correctness. Following simulation, the design is being synthesized and deployed on FPGA.

III. Evaluation

To assess the efficacy of the proposed architecture, we focused on Digital Signal Processing (DSP) workloads which traditionally benefit from VLIW execution and are typically found in embedded applications. We utilized a set of handwritten assembly benchmarks optimized for our hint-based scheduler. We compared these to standard RISC-V performance with GCC compilation. These benchmarks represent a critical subset of the Embench DSP test suite, specifically:

- Fast Fourier Transform (FFT)
- Infinite Impulse Response (IIR) Filters
- Finite Impulse Response (FIR) Filters
- Dot Product Operations

A. Results

Preliminary simulator measurements indicate successful parallel issue of the DSP kernels, resulting in reduced cycle counts compared to the baseline scalar Wally core and speedup ranging from 1.5-4x.

IV. Conclusion

STARBUG is a unique method to bring VLIW efficiency to the RISC-V ecosystem. We have established a robust flow for exploring hint-based parallelism that allows for extensibility and compatibility with the RISC-V ecosystem.

ACKNOWLEDGMENT

The authors would like to acknowledge David Harris and the OpenHW Group for the open-source CVW core and Tor Jerammiasen and other contributors to Google MPACT simulator.

REFERENCES

[1] A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: User-Level ISA*, Version 2.2, May 2017.
[2] D. Harris et al., *The RISC-V Wally Processor*, OpenHW Group.