FastISS RISC-V VP++: A Simulation Performance Evaluation of RVV Workloads

Manfred Schlägl Daniel Große
Institute for Complex Systems, Johannes Kepler University Linz
manfred.schlaegl@jku.at, daniel.grosse@jku.at

JOHANNES KEPLER UNIVERSITY LINZ

RISC-V SUMMIT EUROPE 2025
12 - 15 May 2025
La Cité des Sciences et de l'Industrie, Paris

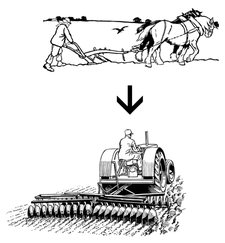
RISC-V®

FastISS Paper | FastISS RVV Paper | RISC-V VP++

ICS

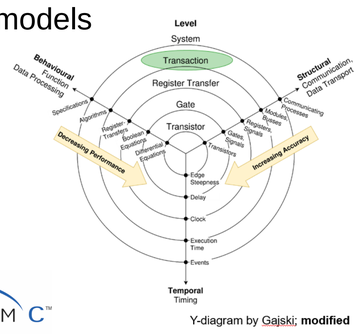
Introduction

RISC-V Vector Extension (RVV):



- ***SIMD***: **S**ingle **I**nstruction, **M**ultiple **D**ata
- **Vector Architecture:** Generic instructions, vector type/length dynamically configurable
- **Acceleration of signal processing, multimedia, machine learning, …**

Virtual Prototypes (VPs): executable software models of a hardware system that run on a host computer.



Y-diagram by Gajski; **modified**

- Modeled at the transaction level (TLM)
- Binary compatible to real/physical hardware
- Widely used by semiconductor global players → Industrial-proven
- Modeled in SystemC (C++ class library, IEEE1666-2023)

SYSTEMC™

Applications: Early design space exploration, parallelization of HW and SW development, and system evaluation and validation, …

RISC-V VP++ [4, 5]

Extensible and configurable SystemC-based, open-source RISC-V VP

- RISC-V 32/64 bit, single/multi-core, **RISC-V Vector Version 1.0** [2, 3]
- Small uC based systems (e.g. bare metal SW, RTOS)
- Complex application processor based systems with virtual memory, graphics, input, mass-storage, network … (e.g. interactive, graphical Linux applications)

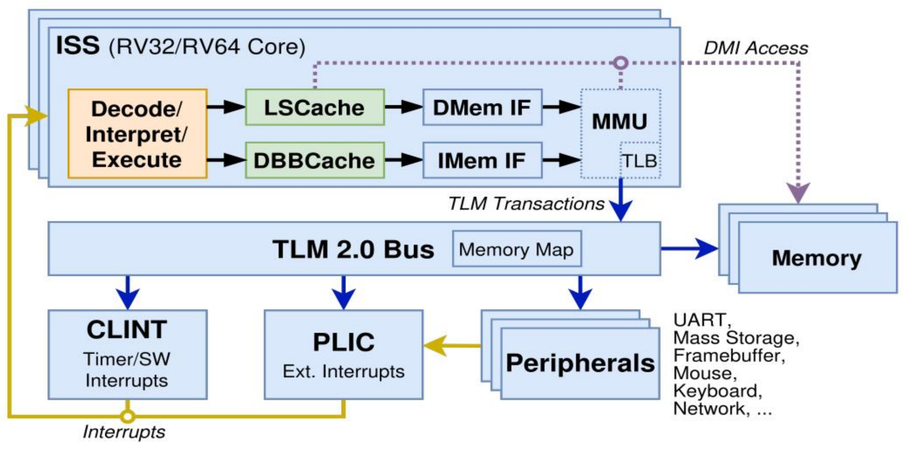
RISC-V Cores → Interpreter-based Instruction Set Simulators (ISS)

→ **Easy to understand:** Fast to create, easy to adapt and extend

→ **Performance very limited:** Fetch, Decode, Dispatch and Execute for every Instruction

Fast Interpreter-Based ISS [1]

Motivation: Improve performance significantly while preserving comprehensibility and adaptability of an interpreter-based ISS



1. Dynamic Basic Block Cache (DBBCache)

→ **Speed up execution** (Fetch, Decode, Dispatch)

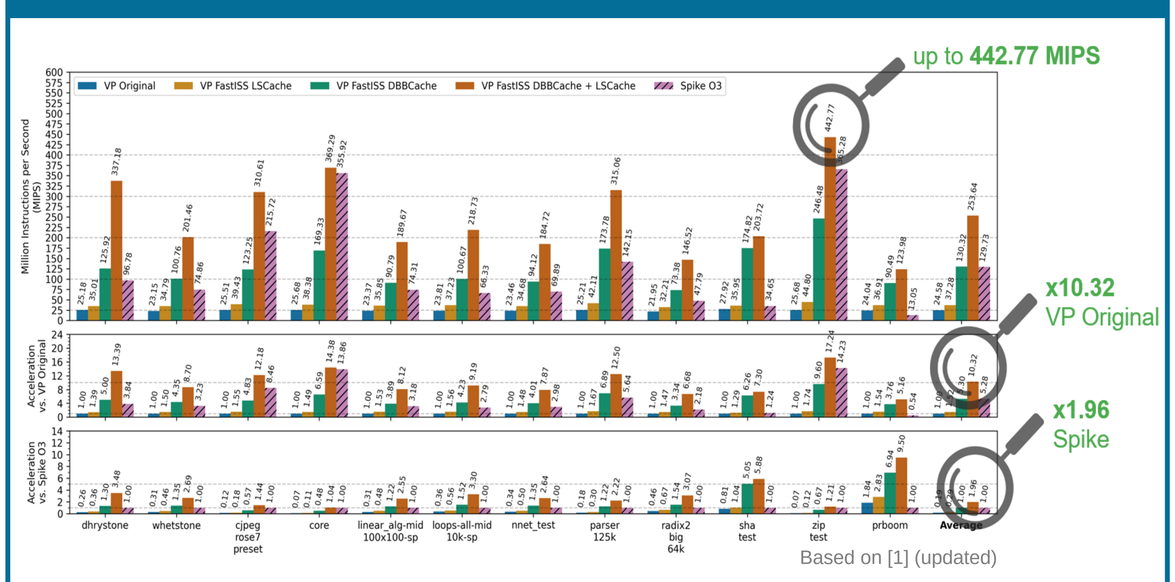
- Extracts Dynamic Basic Blocks (DBBs) at runtime
- Caches as much data as possible for ISS instruction processing (OperationId, Instruction Word, Control flow structure, …)
- Basis for further optimizations: Computed goto, Fast/Slow path, …

2. Load/Store Cache (LSCache)

→ **Speed up memory access**

- Direct mapped cache for 256 in-simulation page addresses
- Direct translation of in-simulation virtual addresses to host system memory addresses

Scalar Simulation Performance [1]



Based on [1] (updated)

Evaluation → No significant negative impact on comprehensibility or adaptability

Vector Simulation Performance

Workload: fbDOOM-RISCV

- Linux port of a classic game from the 1990s
- **optimized for RVV by Semidynamics**
- **Non-vectorized** and (RVV) **Vectorized** variants
- Disabled frame-rate limitation

Measurement:

- Avg. **Frames per Second** (**FPS**) measurement (w.r.t. real host wall-clock time → Simulation Performance)
- Avg. **Million Executed Instructions Per Frame** (**MIPF**)

Results:

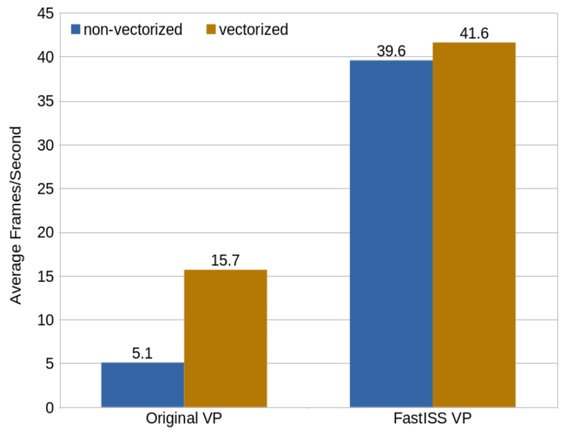


Figure 1: *Average FPS on achieved on* RISC-V VP++

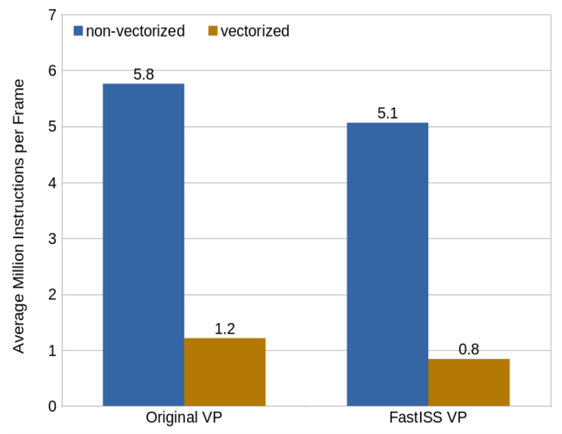


Figure 2: *Average MIPF executed by* RISC-V VP++

- Improved RVV simulation performance by **x2.65** over Original VP (Figure 1)
- Non-vectorized workloads **x7.78** speedup (Figure 1)
  - → Vectorized code uses fewer but more complex instructions → lower **MIPF** for vectorized than non-vectorized (Figure 2)
  - → **FastISS** mainly reduces instruction processing overhead → non-vectorized workloads can benefit more from optimizations

→ **Significant performance improvement for vectorized code (x2.65)**

→ **Non-vectorized workloads can benefit more from optimizations**

References

[1] M. Schlägl and D. Große. "Fast interpreter-based instruction set simulation for virtual prototypes" in DATE, 2025

[2] M. Schlägl and D. Große, "Single instruction isolation for RISC-V vector test failures" in ICCAD, 2024

[3] M. Schlägl, M. Stockinger, and D. Große, "A RISC-V "V" VP: Unlocking vector processing for evaluation at the system level", in DATE, 2024

[4] M. Schlägl, C. Hazott, and D. Große, "RISC-V VP++: Next generation open-source virtual prototype", in OSDA Workshop @ DATE, 2024

[5] M. Schlägl and D. Große, "GUI-VP Kit: A RISC-V VP meets Linux graphics - enabling interactive graphical application development", in GLSVLSI, 2023

This work has been partially supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.