Accelerating software development for emerging ISA extensions with cloud-based FPGAs: RVV case study

Marek Pikuła · ORConf 2024 · Gothenburg · 2024-09-14

Samsung R&D Institute Poland

SAMSUNG

→ **Introduction**

Background

Project timeline

What’s next?

INTRODUCTION

Who am I?

①

FPGA Gateware

Experience in developing gateware for specialized equipment involving fast interfaces and soft cores.

②

Platform Software

Tizen OS platform software developer focused on board support, boot-chain, kernel, and system libraries for the RISC-V ecosystem as part of RISE.

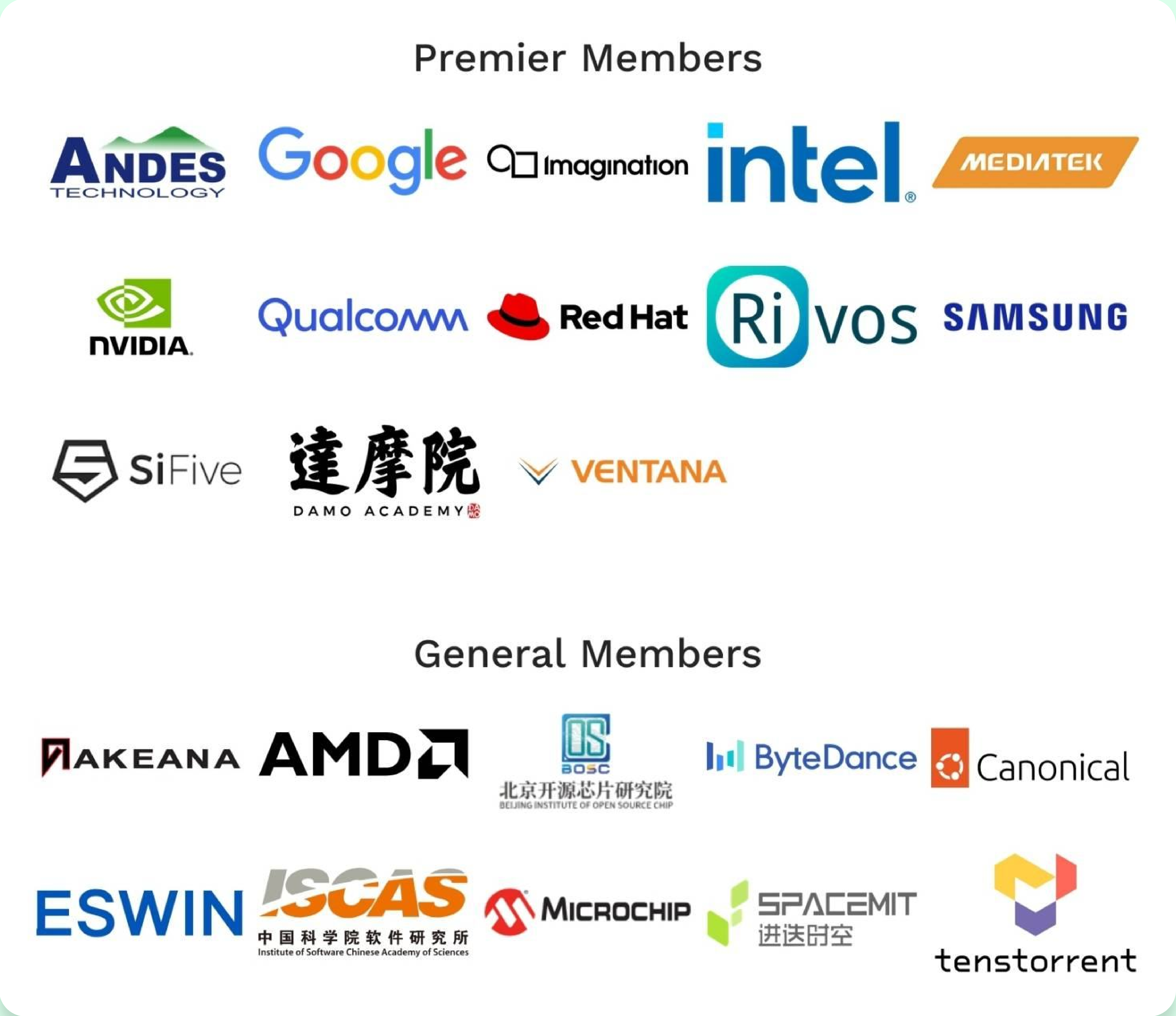
③

CI Workflows

Recently, a lot of effort into multi-platform CI setups for software projects.

(Lightning talk tonight!)

SAMSUNG



Premier Members

General Members

RISE

RISC-V Software Ecosystem

Goal: **Accelerating RISC-V Software Ecosystem development.**

A collaborative effort led by industry leaders with a mission to accelerate the development of open source software for the RISC-V architecture.

Introduction

→ **Background**

Project timeline

What’s next?

BACKGROUND

Project background

① RVV system library support

Internal porting effort for open source Linux packages used in Tizen OS for RISC-V vector extension (RVV1.0).

Selected **pixman** as a starting point.

② No RVV targets (a year ago)

A year ago, no hardware targets with RVV1.0 were available on the market, and internal targets were WIP.

③ QEMU is no good for RVV

QEMU is unsuitable for RVV benchmarking as it doesn't represent a concrete hardware implementation.

We wanted to learn how to write optimal code for a new vector platform.

Project requirements

1. Full RVV1.0 support ⟶ main goal of the project.
2. Linux support (MMU) ⟶ Linux software and libraries.
3. FPGA-compatible ⟶ performance requirement.
4. Easy to use and deploy ⟶ targeted for software developers.
5. Adjustable configuration ⟶ low to high-end configurations.

Introduction

Background

→ **Project timeline**

What next?

PROJECT TIMELINE

Research

Research into CPU cores with RVV1.0

1. **Tenstorrent Ocelot** – failed implementation on FPGA.
2. **PULP Spatz** – not designed to be fully RVV1.0-compliant.
3. **CHIPS Alliance T1** – no plan to implement MMU support.
4. **PULP Ara** – potentially the best candidate.
5. **Others** – not yet released at the start of the project.

Selected CVA6 with PULP Ara

1. OpenHW **CVA6** – open source, high-quality application RV64 core.
2. Ara claims full support for **RVV1.0**.
3. Supports **multiple configurations** of VLEN and lane count.
4. Work-in-progress **MMU** support.

Research into the hardware platform

1. **COTS FPGA** boards capable of fitting a large RVV core are costly and unsuitable for an experimental project.
2. Alternative: **AWS EC2 F1** cloud instances with Xilinx VU9P FPGAs.
3. **FireSim**, along with the **Chipyard** project, provide an easy-to-use framework for simulating a complete SoC at speeds near the FPGA prototype level in the cloud or on-premises.

Integration and testing

Integrating PULP Ara into Chipyard

Benchmarking RVV code on the target

1. The base was the existing **CVA6 integration**.
2. (At the time) the upstream PULP Ara implementation lacked **MMU support**, but a working patch set existed.
3. Experiments with different RVV configurations. First iteration:
   - 2 lanes, VLEN = 2048
   - fmax = 80 MHz
   - 31% LUT, 12% FF, 19% BRAM, 5% URAM, 2% DSP

1. The initial rvv-bench suite run was unsuccessful due to bugs in PULP Ara.
2. Subsequent rvv-bench instruction test confirmed erroneous behavior for some vector instructions.
3. Development of several pixman algorithms and comparison with scalar versions.

PROJECT TIMELINE

Benchmark results

rvv-bench instruction test

145/188 tests were successful. Others resulted in either explicit instruction error, or a complete processor hang.

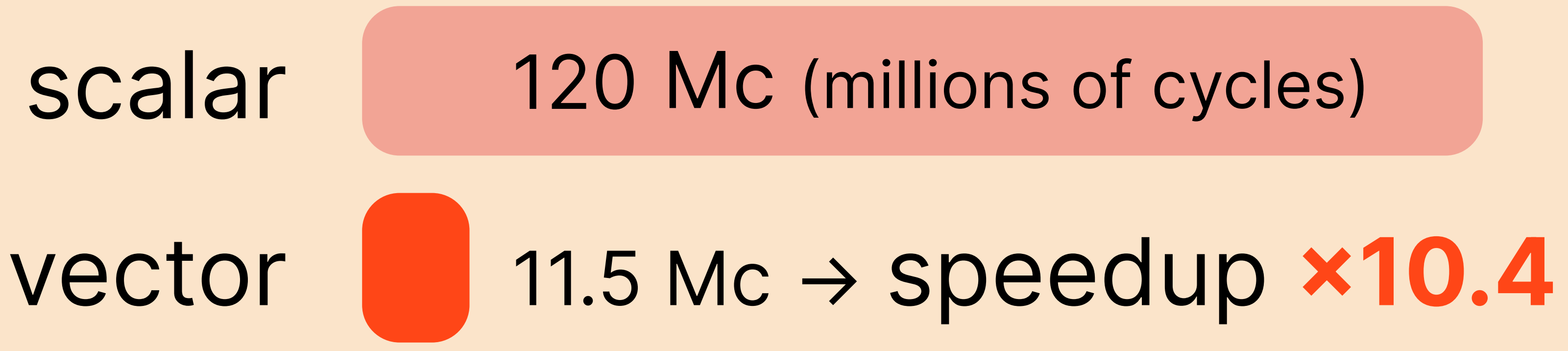
Not tested with the revised version yet!

instruction	e8m1	e8m2	...	e64m4	e64m8
`vadd.vv v8,v16,v24`	16	32	...	64	129
`vadd.vv v8,v16,v24,v0.t`	28	50	...	73	140

⋮

RGB565 to RGB888

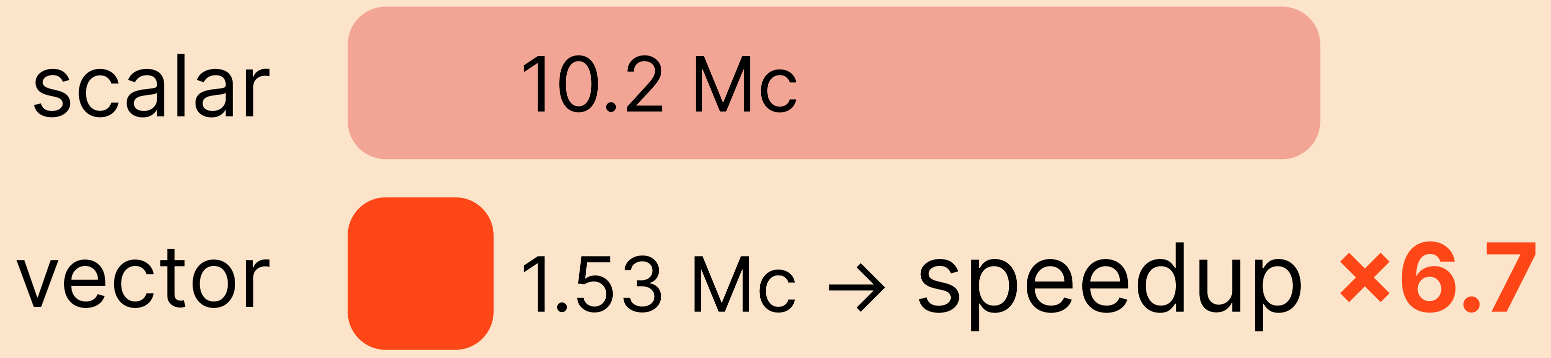
Hand-optimized after reviewing benchmarks:
24 → 18 instruction, 1383 → 1099 cycles



scalar 120 Mc (millions of cycles)

vector 11.5 Mc → speedup ×10.4

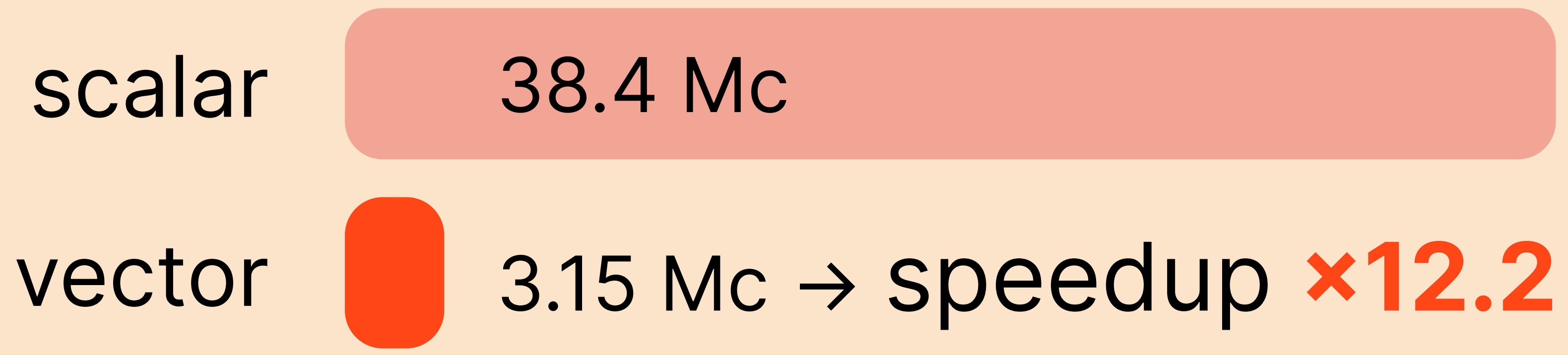
UN8_rb_MUL_UN8



scalar 10.2 Mc

vector 1.53 Mc → speedup ×6.7

UN8x4_MUL_UN8x4_\ ADD_UN8x4_MUL_UN8



scalar 38.4 Mc

vector 3.15 Mc → speedup ×12.2

Introduction

Background

Project timeline

→ **What’s next?**

WHAT'S NEXT?

What's next for the project?

① Rebase on upstream

To use the official MMU support and incorporate fixes.

② Support more configurations

Different VLENs and lane counts to provide more benchmarking targets.

③ Streamline deployment

So that it is easy for a regular developer to spin up a target and test their code.

④ Upstream changes

To have official support for PULP Ara in Firesim.

What's next for RISC-V?

How to make the evaluation, testing,
and adoption of new extensions
easier for software developers?

Thank you!

Questions?

Project repository

m.pikula@partner.samsung.com

linkedin.com/in/marek-pikula

github.com/MarekPikula

SAMSUNG