ACCELERATING SOFTWARE DEVELOPMENT FOR EMERGING ISA EXTENSIONS WITH CLOUD-BASED FPGAS: RVV CASE STUDY

The RISC-V Vector Extension (RVV) promises an enhanced performance and power efficiency across various complex computational tasks. However, the efficient utilization of RVV demands careful consideration of the optimization approach. This project examines strategies for accelerating this process. Key challenges include assessing performance differences among algorithmic approaches and overcoming initial hardware constraints. FireSim provides a comprehensive solution by offering advanced software and hardware simulation capabilities. Utilizing FireSim, we started the process of enhancing source code with RVV instructions (called vectorization) for the pixman project. Our experience outlines the efficacy of a cloud-based FPGA simulation in expediting software development for emerging ISA extensions. Overall, FireSim facilitates faster iteration cycles and informed design decisions, benefiting individual developers and fostering collaboration in remote teams.

BACKGROUND

- Support porting effort for open source Linux packages used in Tizen OS for RISC-V vector extension (RVV1.0) – chosen **pixman**, which is used in many GUI applications.
- At the start of the project, there were **no hardware targets** with RVV1.0 available on the market, and there were no release dates of potential candidates.
- **QEMU is not suitable** for RVV benchmarking, as it doesn't implement a concrete hardware implementation, and rather translates RVV to host's SIMD code.

REQUIREMENTS

- **Full RVV1.0 support** to enable software engineers to utilize the full capabilities of the extension in the ported software.
- **Linux support** because ported software is Linux-based.
- **MMU support** required to run full Linux.
- **Possible to run on an FPGA** – performance requirement to perform meaningful benchmarks.
- **Easy to use and deploy** – the goal is to provide software developers with a ready-made development and testing environment.
- **Support for *perf* profiling** – for comparing different implementations.
- **Option to adjust microarchitecture** to benchmark code on low- to high-end configurations (e.g., adjustable VLEN, lane count, etc).

PROJECT TIMELINE

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.

Selected PULP Ara

1. Vector coprocessor for OpenHW Group **CVA6** – a quality, open source, application class RV64 core.
2. Claimed 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 not suitable for an experimental project.
2. Alternative: **AWS EC2 F1** cloud instances with Xilinx VU9P FPGAs.
3. **FireSim**, along with **Chipyard** project, provide an easy-to-use framework to simulate a complete SoC at near-FPGA-prototype speeds on cloud or on-premises FPGAs.

Integrating PULP Ara into Chipyard

1. The base was the existing **CVA6 integration**.
2. The upstream Ara implementation lacks **MMU support**, but working community efforts exist.
3. Experiments with different RVV configurations. First iteration:
   - 2 lanes, VLEN = 2048
   - f_{max} = 80 MHz
   - 31% LUT, 12% FF, 19% BRAM, 5% URAM, 2% DSP

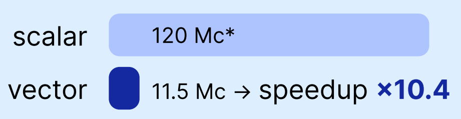
Benchmarking RVV code on the target

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.

BENCHMARKING RESULTS

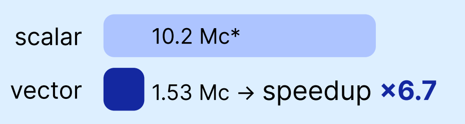
RGB565 to RGB888 conversion

Initial implementation was hand optimized after reviewing benchmarks:
- 24 → 18 instructions
- 1383 → 1099 cycles

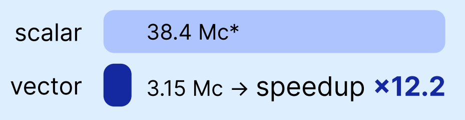


```
asm("vsetvli %0, %1, e16, m4, ta, ma" : "=r"(vlen) : "r"(len_left));
asm("vle16.v v16, %0" :: "m"(*cur_s));

// Handle R and B channels.
asm("vand.vx v20, v16, %0" :: "r"(0×F800));
asm("vand.vx v28, v16, %0" :: "r"(0×001F));
asm("vwmulu.vx v0, v20, %0" :: "r"(1 << 8));
asm("vwmaccu.vx v0, %0, v28" :: "r"(1 << 3));
asm("vsetvli x0, x0, e32, m8, ta, ma");
asm("vsrl.vi v8, v0, 5");
asm("vor.vv v0, v0, v8");
asm("vand.vx v0, v0, %0" :: "r"(0×FF00FF));

// Handle G channel.
asm("vsetvli x0, x0, e8, m2, ta, ma");
asm("vnsrl.wi v8, v16, 5");
asm("vsll.vi v8, v8, 2");
asm("vsrl.vi v10, v8, 6");
asm("vwaddu.vv v16, v8, v10");
asm("vsetvli x0, x0, e16, m4, ta, ma");
asm("vwmaccu.vx v0, %0, v16" :: "r"(1 << 8));
asm("vse32.v v0, %0" : "=m"(*cur_d));
```

UN8_rb_MUL_UN8



UN8×4_MUL_UN8×4_ADD_UN8×4_MUL_UN8



CONCLUSIONS

Our experience with FireSim demonstrates the potential of the cloud-based FPGA simulation for accelerating software development for emerging ISA extensions like RVV. By providing access to cost-effective, scalable hardware resources and comprehensive simulation capabilities, FireSim enables faster iteration cycles and more informed design decisions. This approach not only benefits individual developers but also facilitates collaboration in remote teams, bridging the gap between hardware and software development efforts.

MAREK PIKUŁA
Samsung R&D Institute Poland
m.pikula@partner.samsung.com

Embedded developer by day, DevOps engineer by night. Marek creates high-quality, well-tested and documented solutions in established technologies while actively exploring the new and shiny. He feels the best in complex projects requiring system-level and in-detail perspectives, connecting multiple domains from hardware through gateware and firmware up to the software running in the cloud.

MAREK SZYPROWSKI
Samsung R&D Institute Poland
m.szyprowski@samsung.com

Marek is a Linux kernel developer at Samsung R&D in Warsaw, Poland. He specializes in embedded systems. His ongoing effort is to provide better support for Samsung SoC in the Linux kernel. In recent years he focused on the day-to-day testing of the Linux kernel project.

*Mc – millions of cycles

SAMSUNG