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

Marek Pikuła[1], Marek Szyprowski[1]

[1]Samsung R&D Institute Poland

Abstract

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 article 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.

Introduction

RISC-V Vector Extension (RVV) is making its way towards mainstream use. It opens up exciting avenues for software vectorization on RISC-V, promising improved performance and power efficiency across various computational tasks ranging from image processing to AI workloads. However, realizing its full potential demands careful evaluation and optimization of software implementations. This article delves into the nuances of accelerating software development for new ISA extensions, with a spotlight on the RVV extension.

Approaches to Vectorization

One of the key challenges in the software RVV vectorization is assessing the performance difference between different algorithmic approaches. Since the ratification in 2021, RVV support has gradually come to the upstream toolchains. Although the development progress of auto-vectorizers in both GCC and LLVM is promising, some use cases will always require a manual approach. Unfortunately, not many reliable sources give a "golden standard" guide to follow. Two notable exceptions are RVVRadar[1] and RVV benchmark from Camel Coder[2], but both require physical hardware or manual performance calculations.

Initial Hardware Constraints

At the outset of any project involving emerging ISA extensions like RVV, there is often a lack of readily available hardware for testing and validation. While tools like QEMU provide means to check the validity of the algorithms, they may not accurately reflect performance characteristics. When comparing scalar and RVV implementations of the same algorithm, the recently released QEMU 8.2 performs worse on the latter. This can lead to suboptimal design choices based on incomplete or inaccurate performance assessments.

Role of Simulation Tools

Simulation tools play a crucial role in software development for emerging ISA extensions. While the Spike ISA simulator offers robust tracing and debugging functionality, it lacks in the area of microarchitectural considerations, which are particularly important for such a large and complex ISA extension like RVV.

Software HDL simulation tools provide detailed profiling capabilities but suffer from slow performance, especially for complex designs. Classical FPGA prototyping offers a middle ground but comes with high initial costs and limited tracing capabilities.

FireSim Simulation

To address these challenges, FireSim[3] offers a comprehensive solution. FireSim extends the ChipYard[4] project with advanced software and hardware simulation capabilities, making it an ideal platform for developing and testing RISC-V core implementations, along with external or internal specialized accelerators.

Key Features

FireSim provides several key features tailored to software development for emerging ISA extensions:

- Ready-to-use processor and accelerator cores with pre-built *FireMarshal* Linux environment.
- *TracerV*: In-hardware tracing with out-of-band Flame Graph visualization.
- Support for automatic ILA insertion, assertion synthesis, and out-of-band performance counters.
- Multicore and multi-SoC setups (so-called *supernode* simulation) with on-chip network connected to the host for complex workload scenarios.

Cloud Deployment

One of the notable advantages of FireSim is its support for cloud deployment, specifically on AWS EC2 F1 instances with Xilinx UltraScale+ VU9P FPGA cards. This approach offers a low entry barrier on both technical and financial levels, allowing teams to provision FPGA resources quickly and pay only for the compute time used. It also significantly decreases the risk of the entire operation. There is no need to buy expensive FPGA hardware or license Vivado, as everything comes in a pay-by-hour manner, which is especially important for software-focused teams.

This approach might also appeal to multidisciplinary, remote teams. For example, the hardware team can develop an SoC on local hardware they already have on-site (FireSim also supports local targets) and immediately pass the work to a remote software team. Overall, this can dramatically increase the speed of evaluation and integration and allow for more dynamic, end-to-end product development.

RVV Case Study

We utilized FireSim to develop and optimize the RVV port for the *pixman*[1] project. First, we evaluated readily available RVV-capable cores. We started with the *Tenstorrent Ocelot*[5] project (a RISCV-BOOM core with integrated RVV accelerator), which already had ChipYard integration. Unfortunately, the implementation was not directly synthesizable in the FireSim setting. The second choice was *PULP Ara*[6], a coprocessor for the CORE-V CVA6 core. We smoothly adapted the existing ChipYard CVA6 wrapper and successfully built and simulated the SoC on AWS node, running a provided Fedora image by following the comprehensive and, importantly, complete documentation of the aforementioned projects, even though we had no prior experience with AWS tools.

Development Process

In the initial phase of the pixman vectorization, we focused on the correctness of the approach and not on performance. For this we used QEMU environment, because, even with its suboptimal RVV implementation, it greatly outperforms any other emulation solution. Another important benefit is the possibility of working on a local machine without the need to provision AWS resources.

Once we worked out the general solution, we started profiling the code in the FireSim environment. This way, we could reliably compare between different optimizations and with the base scalar implementation. To do this, we used detailed TracerV reports along with performance counter measurements. We also experimented with different microarchitectural configurations to see the implications for overall performance, which allowed us to make informed decisions regarding algorithmic approaches – optimal in both low- and high-end configurations.

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.

References

[1] Lucas Klemmer, Manfred Schlaegl, and Daniel Große. "RVVRadar: A Framework for Supporting the Programmer in Vectorization for RISC-V". In: *ACM Great Lakes Symposium on VLSI*. 2022.

[2] Camel Coder. "Vectorizing Unicode conversions on real RISC-V hardware". In: (2022). URL: https://camel-cdr.github.io/rvv-bench-results/articles/vector-utf.html.

[3] Sagar Karandikar et al. "FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud". In: *ISCA@50 Retrospective: 1996-2020*. Ed. by José F. Martínez and Lizy K. John. June 2023.

[4] Alon Amid et al. "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs". In: *IEEE Micro* 40.4 (2020), pp. 10–21. ISSN: 1937-4143. DOI: 10.1109/MM.2020.2996616.

[5] Srikanth Arekapudi and Dongjie Xie. "Ocelot: Open Source Vector Unit". In: *RISC-V Summit North America 2022*. 2022. URL: https://github.com/tenstorrent/riscv-ocelot/.

[6] Matheus Cavalcante et al. "Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor With Multiprecision Floating-Point Support in 22-nm FD-SOI". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.2 (2020), pp. 530–543. DOI: 10.1109/TVLSI.2019.2950087.

[1] https://gitlab.freedesktop.org/pixman/pixman