Towards an Industrial-Grade Open-Source FPU for RISC-V Vector Processors

Enis Mustafa*, Michael Platzer*, Domenic Wüthrich* and Florian Zaruba*

Axelera AI

Abstract

Hardware floating-point units (FPUs) are crucial for HPC, ML, and embedded applications, yet no stand-alone RISC-V FPU fully supports vector extensions. We have enhanced CVFPU into a production-ready solution by optimizing timing and power (31% leakage reduction, up to 48% peak power savings), adding vfrec7/vfrsqrt and symmetrical widening adds, and improving verification by fixing seven bugs. These contributions make CVFPU a more viable choice for performance-critical applications while strengthening the overall open-source RISC-V ecosystem.

Introduction

The rise of RISC-V as an open Instruction Set Architecture (ISA) has driven innovation in hardware design. While open-source software is widely adopted, open-source hardware remains niche, with limited commercial uptake.

Adopting fully open-source processor cores is challenging due to the high development cost and the reluctance of companies to share their work. Beyond Register Transfer Level (RTL) code, full processor integration requires extensive verification and EDA tooling, areas where open-source alternatives are still developing. As a result, academic institutions lead much of the open-source hardware work, though often without commercial intent.

However, while full core designs remain difficult to open-source, sub-components like execution units offer practical opportunities. Open-source hardware components benefit smaller companies by providing well-validated designs to build upon.

This report presents a case study on how Axelera AI enhanced an open-source floating-point unit (CVFPU, formerly FPnew) [1] and integrated it into an ultra-wide RISC-V vector processor.

Contributions

CVFPU is an open-source, highly parameterizable multi-format floating-point unit (FPU) used in several RISC-V cores, including the popular CVA6 [2]. It supports SIMD operations, allowing multiple operands to be processed simultaneously, making it well-suited for vector workloads.

However, we identified several gaps in its support for the 70+ vector floating-point instructions in the RISC-V V extension. Key missing features included symmetrical widening addition and LUT functions, which we implemented. Additionally, large-scale vector processing places extreme timing pressure on the fused multiply-add (FMA) block, particularly without automatic retiming. To address this, we introduced parallel summation with a leading zero anticipator to improve timing for strict cycle constraints (see Figure 1).

Furthermore, some operations natively supported by CVFPU but less used in scalar RISC-V - such as widening and narrowing - had undergone limited verification. In our work, we discovered and fixed seven hardware bugs, all of which have been upstreamed.

This report presents our modifications, along with a detailed analysis of power, area, and timing, demonstrating how open-source hardware can be extended to benefit the broader RISC-V community.

CVFPU already supported an asymmetric widening add operation, where one narrow operand and one wide operand are added to produce a wide result. To support RISC-V V extension, a symmetric widening add, where two narrow operands are added to form a wide result, is required.

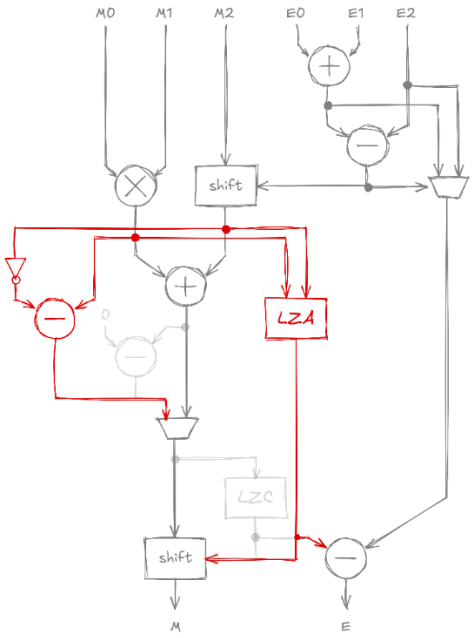

Figure 1: Our contributions (highlighted in red) to the FMA Block of CVFPU. The previous design had the summation, the conditional negation, and the Leading Zero Count (LZC) in sequence. With our changes, these three operations are done in parallel, with a Leading Zero Anticipator (LZA) replacing the LZC.

* All authors contributed equally to this work.

Symmetric Widening Add

Fortunately, the flexible design of CVFPU allowed us to add support for such a symmetric widening add with very little overhead, leveraging the existing shifting logic for the mantissa of the second addend to perform the proper pre-widening.

Improvements to the FMA data path

The long data path and high complexity of the FMA block create timing issues, worsened by parallel instantiations of multiple CVFPU units. Hence, we attempted to shorten the length of paths through this block.

Previously, the FMA multiplied the mantissa of the two multiplicands, then added the product to the shifted mantissa of the addend. Under certain conditions, the resulting sum needs to be negated, which was done in a third step. Finally, the result was normalized by counting the leading zeroes of the result mantissa and shifting it accordingly.

We improved upon this design by performing three of these four operations in parallel, rather than having all of them in sequence. A negated sum is computed in parallel with the positive sum, avoiding the need for a separate negation step. A leading zero anticipator (LZA), which anticipates the leading zero count of the sum based on the addends, replaces the LZC on the result.

Vector Support for RISC-V DV

RISC-V DV[†] is a popular random instruction generator, designed to provide extensive coverage of the RISC-V base instruction set. However, it currently lacks support for the latest RISC-V Vector ISA (version 1.0). To address this gap, we contributed an updated implementation that ensures compliance with the most recent specification.

By leveraging RISC-V DV with our extended vector ISA support, we successfully identified and resolved seven bugs in the floating-point unit (FPU), all of which have been fixed and upstreamed to benefit the broader RISC-V community.

Evaluation

For the evaluation, we physically-aware synthesized the CVFPU with and without the proposed changes in a 5nm process node using the official Process Design Kit (PDK) and standard cell libraries. We verified the added features and bug fixes using our RISC-V DV implementation. The CVFPU was configured as in the default CVA6 setup, with three pipeline stages through the FMA, but set to merged unit mode. For our experiments the synthesis was constrained with an input/output delay of 100ps and a setup clock uncertainty of 5% of the cycle period.

We ran two experiments: one with fixed pipeline stages and another with retiming enabled, allowing the synthesizer more flexibility in balancing the stages. In both cases, physical-aware synthesis was performed for the original design (Original) and the enhanced design (LZA). The first key observation is that both the Original and LZA designs perform significantly better when retiming is enabled.

As shown in Figure 2, the LZA-enhanced design exhibits a more moderate increase in area for both cases, indicating that the design is under less pressure and that the enhancements effectively improve timing.

For the retimed case, we also ran a power simulation for both designs, showing a reduction in leakage power of over 30%, from 2.36mW (Original) to 1.79mW (LZA). This reduction is further reflected in the Vt mix, where the usage of low-Vt cells decreases from 27% to 20%, with the remainder being regular-Vt cells.

For dynamic power simulation, we evaluated the design using mixed arithmetic operations over a 2000ns period in a stand-alone testbench. We dumped toggle activity and back-annotated 75% of it into Fusion Compiler for power estimation. The results show an average power reduction of 7% over the entire estimation period and up to a 48% reduction in peak power.

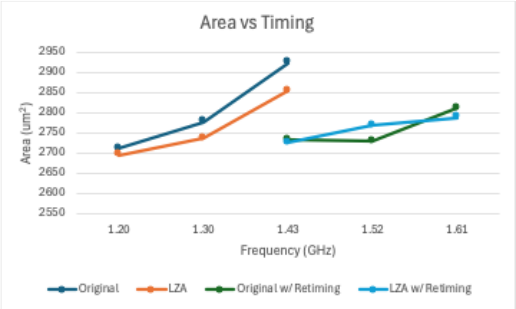

Figure 2: Area vs. Timing plot for the original design and the LZA enhanced design (with retiming).

Related Work and Conclusion

Several open-source FPUs exist, such as Rocket's hard-float and Xiangshan's Fudian (both in Chisel), which pose integration challenges in Verilog-based ASIC flows. The Walley FPU, while well-documented with a solid architecture, is less parameterizable than CVFPU.

Our work advances CVFPU from a research prototype to a production-ready IP, showcasing the potential of open-source hardware. To sustain progress, expanding on maintainers and investing in shared CI/regression infrastructure will be key to fostering a mature open-source IP ecosystem.

References

[1] Mach, Stefan, et al. "FPnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing." *IEEE Transactions on Very Large Scale Integration Systems* 29.4 (2020): 774-787.

[2] Zaruba, Florian, and Luca Benini. "The cost of application-class processing: Energy and performance analysis of a Linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (2019): 2629-2640.

[3] Asanovic, Krste, et al. "The rocket chip generator." *EECS Department, University of California, Berkeley.* Rep. UCB/EECS-2016-17 4 (2016): 6-2.

[†] https://github.com/chipsalliance/riscv-dv