Vicuna 2.0 : A Configurable RISC-V Embedded Vector Hardware Platform

J. Parker Jones[1]*, Philipp van Kempen[2], and Daniel Müller-Gritschneder[1]

[1]Embedded Computing Systems, Technical University of Vienna
[2]Chair of Electronic Design Automation, Technical University of Munich

Abstract

Vicuna 2.0 is an open-source SystemVerilog implementation of the `Zve32x`, `Zve32f`, and `Zvfh` extensions built upon the previous work of the Vicuna project[1]. Vicuna 2.0 is extremely configurable, allowing for detailed analysis and evaluation of vector unit configurations targeted for specific workloads as demonstrated by a design space exploration in this work. Vicuna 2.0 is believed to be the first open-source implementation of the `Zve32f` and `Zve32f_Zvfh` subsets of the RISC-V Vector Extension.

Background

The RISC-V Vector (`V`) Extension, ratified in 2021, also includes extension subsets targeted at lower cost embedded devices [2]. Specifically, the `Zve32x` Extension and the `Zve32f` Extension, which provide support for basic vector integer and IEEE 754 single-precision floating-point operations. The `Zvfh` extension is also included in the specification and provides support for IEEE 754 half-precision floating-point operations. Together, these extensions allow for the addition of different levels of vector support and provide a very large design space for embedded vector processors when combined with the customization options already present in the `V` Extension.

The original Vicuna project by Platzer and Puschner is a configurable, timing-predictable vector co-processor implementing a subset of the `Zve32x` Extension in SystemVerilog[1]. The co-processor implements the eXtension InterFace (XIF) provided by the OpenHW Group [3], allowing for connection to any core such as the CV32E40X[4]. The configuration features allow for users to define their own configurations, with variable vector register width (`VREG_W`), number and configuration of vector pipelines, datapath width of each pipeline (`VLANE_W`), as well as many other features of the design. The chosen configuration can then be simulated using Verilator[5].

Improvements Over the Original Vicuna Project

Vicuna 2.0 provides multiple upgrades when compared to the original Vicuna project:

- Addition of a vector division unit. Completes support for the `Zve32x` extension.
- Optional support for the `Zve32f` and `F` extensions. Includes the addition of a vector floating-point unit and a scalar floating-point co-processor both based on the CV-FPU (Formerly known as FP_NEW)[6].
- Optional support for the `Zvfh` and `Zfh` extensions. Upgrades floating-point units to support half-precision operations.
- Allow for scalar execution of benchmarks for comparison purposes.
- Improved memory interface to increase efficiency of vector loads and stores.
- Expanded data collection during Verilator simulations.
- Various RTL bug fixes.
- CMake build system for user-friendly configuration and addition of new tests and benchmarks.

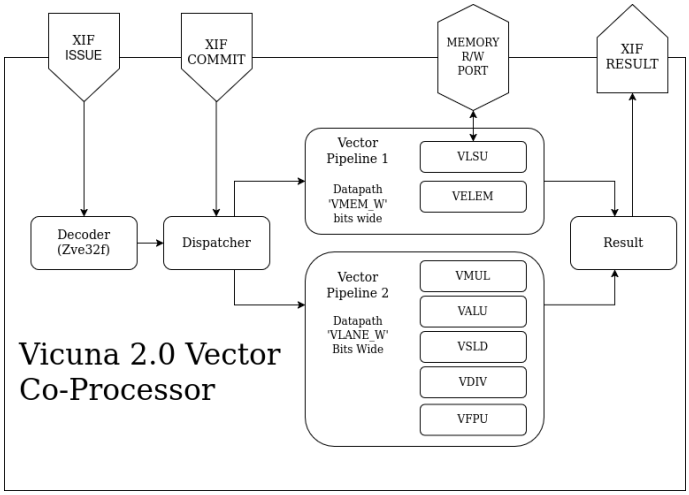

Figure 1: *Block Diagram of the Vector Co-Processor in the 'Dual Pipeline' Configuration*

*Corresponding author: `jefferson.jones@tuwien.ac.at`

Demonstration of Vicuna 2.0

In order to demonstrate the capabilities of Vicuna 2.0 as a platform, a design space exploration was performed for the anomaly detection benchmark (*toycar*) from the MLPerf Tiny benchmark suite for embedded machine learning inference[7]. Using the provided trained models and input data, INT8, FP32, and FP16 quantized versions of the benchmark were made. Tensorflow Lite for Microcontrollers (TFLM) was used to deploy these models [8]. As TFLM does not natively support operations with FP16, new kernels were included to allow for these operations. LLVM autovectorization was used to generate vectorized code for these benchmarks.

Vicuna was configured with two vector pipelines, one containing the Vector Load/Store unit and the other containing all other functional units. The first pipeline containing most functional units has a datapath width of `VLANE_W` bits. The width of second pipeline containing the vector load/store unit, `VMEM_W`, was fixed to 32 bits to represent an embedded system with a limited memory interface. The vector unit was connected to the 4-stage CV32E40X scalar core.

For the experiments, the `VREG_W` was varied from 128 to 4096 bits. All valid values of `VLANE_W` were tested. The top level Verilator simulation was used to gather data during simulations, such as total cycles and total instructions. In addition, the opcode of each executed and stalled instruction was logged.

Design Space Exploration Results

Table 1 shows the configuration of Vicuna with fastest execution time (in total number of cycles) for the `Zve32x`, `Zve32f`, and `Zvfh` implementations. In addition, the CPI and speedup factor compared to the respective scalar unit are also reported. An interesting observation is that the `Zvfh` system has an optimal `VREG_W` that is half that of the `Zve32f` system, caused by the use of half-precision floating-point values as intermediates.

The instruction stall data provides more insights, mainly that the floating-point systems are considerably more memory-bound than the integer one. Memory loads account for 52.1% of the total runtime for `Zve32f` and 56.2% for `Zvfh`. This is due to the use of their storage formats for calculations, which removes the need for the conversion operations used by the Zve32x system. This implies that increasing `VMEM_W` could drastically improve performance. As expected, running the experiment again with a `VMEM_W` of 64 bits results in a speedup of 1.19x and 1.11x for the `Zve32f` and `Zvfh` systems respectively when compared the the systems in Table 1.

	`Zve32x`	`Zve32f`	`Zvfh`
`VREG_W` Bits	1024	512	256
`VLANE_W` Bits	512	256	128
Total Cycles	1298593	1124928	865799
CPI	2.15	2.07	3.15
Speedup vs. Scalar	2.67x	3.00x	3.30x

Table 1: *Fastest Configuration of Vicuna for INT8, FP32, and FP16* `toycar`

Conclusion and Future Work

Vicuna 2.0 provides a significant upgrade to the original Vicuna project, completing support for `Zve32x` and adding support for `Zve32f` and `Zvfh`. As demonstrated here, it can be used as a platform for design space exploration while making use of the Verilator simulation environment to gather detailed information on the state of the vector unit for further analysis.

Future work for this project involves implementation of Vector Cryptography Extensions, support for RTL synthesis, investigation into pipelining functional units, and vector pipeline visualization efforts.

Vicuna 2.0, along with all dependencies and programs used for the experiments are open-source[1], and any contributions are welcome.

References

[1] Michael Platzer and Peter Puschner. "Vicuna: A Timing-Predictable RISC-V Vector Coprocessor for Scalable Parallel Computation". In: *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*. 2021. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13932.

[2] RISC-V Collaboration. *The RISC-V Instruction Set Manual, Vol 1, Unpriveleged Architecture, Version 20240411.*

[3] OpenHW Group. *OpenHW Group Specification: CORE-V eXtension interface (CV-X-IF)*. https://docs.openhwgroup.org/projects/openhw-group-core-v-xif/en/latest/. Accessed: 2025-02-07.

[4] OpenHW Group. *OpenHW Group CORE-V CV32E40X RISC-V IP*. https://github.com/openhwgroup/cv32e40x. Accessed: 2025-02-07.

[5] Verilator Collaboration. *Verilator*. https://www.veripool.org/verilator/. Accessed: 2025-02-07.

[6] Stefan Mach et al. *FPnew: An Open-Source Multi-Format Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing*. 2020. arXiv: 2007.01530 [cs.AR]. URL: https://arxiv.org/abs/2007.01530.

[7] Colby Banbury et al. *MLPerf Tiny Benchmark*. 2021. arXiv: 2106.07597 [cs.LG]. URL: https://arxiv.org/abs/2106.07597.

[8] Robert David et al. *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*. 2021. arXiv: 2010.08678 [cs.LG]. URL: https://arxiv.org/abs/2010.08678.

[1] https://github.com/vproc