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

Background

The RISC-V Vector (`V`) Extension provides many levels of vector support for embedded devices [2].

- `Zve32x` - embedded vector integer
- `Zve32f` - embedded vector single-precision floating-point
- `Zvfh` - vector half-precision floating point

The original Vicuna project [6] provides an incomplete implementation of the `Zve32x` extension as a co-processor using the eXtension InterFace (XIF) [4]. Vicuna is highly configurable, allowing users to configure the vector register width (`VREG_W`), number and configuration of vector pipelines, and datapath width of each pipeline (`VLANE_W`).

Improvements with Vicuna 2.0

- Addition of a vector division unit. Completes support for the `Zve32x` extension.
- Optional support for the `Zve32f` extension. 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) [5].
- Optional support for the `Zvfh` extension. Upgrades floating-point units to support half-precision operations.
- 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, addition of new tests and benchmarks, and integration into other projects.

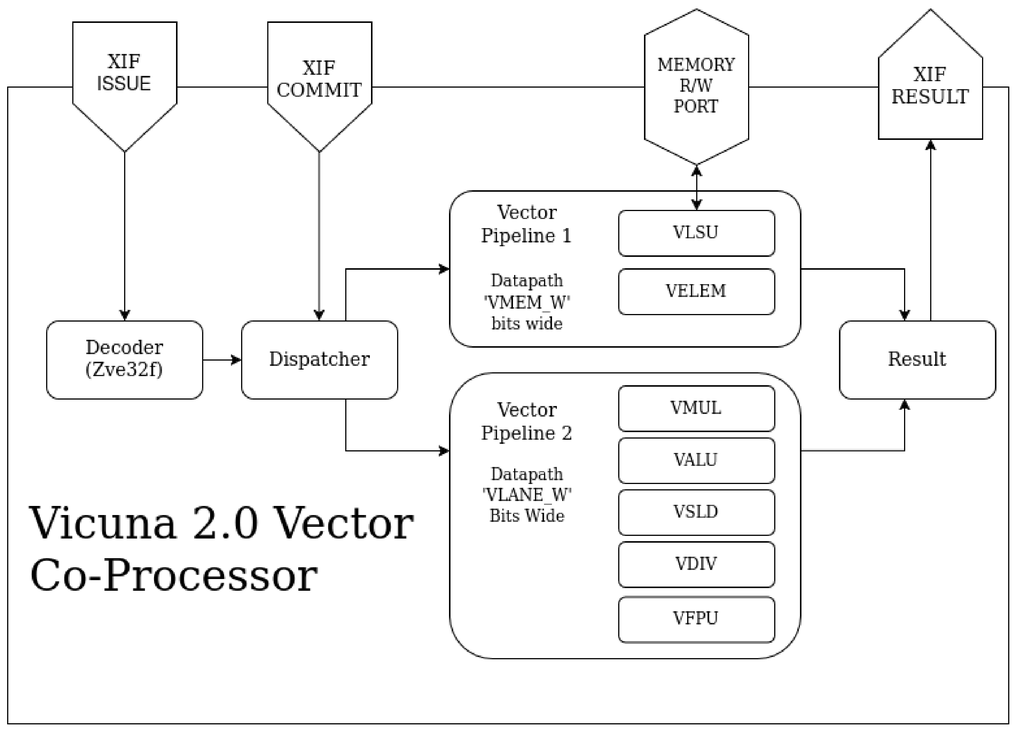

Figure 1: Block Diagram of the Vector Co-Processor in the 'Dual Pipeline' Configuration for `Zve32f`

Demonstration of Vicuna 2.0 with Machine Learning Inference

In order to demonstrate the stability of Vicuna 2.0 and its use as a design testing platform, a design space exploration was performed for the anomaly detection benchmark (*toycar*) from the MLPerf Tiny benchmark suite for embedded machine learning inference [1]. The testing environment was set up as follows:

- INT8, FP32, and FP16 quantized versions of the benchmark were made from provided trained model.
- Tensorflow Lite for Microcontrollers (TFLM) was used to deploy these models [3]. New kernels were added to provide FP16 support.
- LLVM autovectorization was used to generate vectorized code.
- Vicuna was configured in the 'Dual-Pipeline' configuration shown in Figure 1. `VMEM_W` was fixed to 32 bits.
- Vicuna was connected to the CV32E40X scalar core, with an additional scalar floating-point co-processor included if required.
- `VREG_W` was varied from 128 to 4096 bits and all valid values of `VLANE_W` were tested.
- Results for each vectorized benchmark were compared their respective scalar execution.
- The Verilator simulation environment was used to collect cycle, instruction, and stall data.

Design Space Exploration Results

	`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
% Stall Cycles at Memory	34.1%	43.2%	56.2%

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

From the results in Table 1, the floating-point configurations perform better than the integer configuration. From the instruction usage data, this can be attributed to the sign extension operations necessary for the integer-only configuration, as the floating-point configurations can perform operations with the data source format.

As a result of this lesser load on the computational functional unit, both of the floating-point benchmarks are more memory bound, implying they would benefit from a larger `VMEM_W`. Running these tests again with `VMEM_W` = 64 bits results in a speedup of 1.19x and 1.11x for the `Zve32f` and `Zvfh` systems respectively.

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 at `https://github.com/vproc`, and any contributions are welcome.

References

[1] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau, U. Thakker, A. Torrini, P. Warden, J. Cordaro, G. D. Guglielmo, J. Duarte, S. Gibellini, V. Parekh, H. Tran, N. Tran, N. Wenxu, and X. Xuesong. Mlperf tiny benchmark, 2021.

[2] R.-V. Collaboration. The risc-v instruction set manual, vol 1, unpriveleged architecture, version 20240411.

[3] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems, 2021.

[4] O. 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.

[5] S. Mach, F. Schuiki, F. Zaruba, and L. Benini. Fpnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing, 2020.

[6] M. Platzer and P. Puschner. Vicuna: A Timing-Predictable RISC-V Vector Coprocessor for Scalable Parallel Computation. In *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, 2021.