POLITECNICO DI TORINO
Repository ISTITUZIONALE

ANT-V: A Monolithic RISC-V Vector Processor for Machine Learning at the Edge

Original
ANT-V: A Monolithic RISC-V Vector Processor for Machine Learning at the Edge / Guella, F., Caviglia, A., Caon, M., Carlo, S.D., Masera, G., Martina, M.. - In: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. II, EXPRESS BRIEFS. - ISSN 1549-7747. - (2026), pp. 1-1. [10.1109/tcsii.2026.3733550]

Availability:
This version is available at: 11583/3015893 since: 2026-09-24T17:46:03Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/tcsii.2026.3733550

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

ANT-V: A Monolithic RISC-V Vector Processor for Machine Learning at the Edge

Flavia Guella, Alessio Caviglia, Michele Caon, Stefano Di Carlo, Guido Masera, and Maurizio Martina

***Abstract*—The shift towards edge computing, coupled with the increasing adoption of Machine Learning (ML) for data-intensive tasks, imposes demanding performance requirements on embedded systems, where employing dedicated accelerators is often impractical due to tight power and area budgets. To address these challenges, we present ANT-V, a lightweight RISC-V vector processor designed to efficiently execute data-parallel workloads within the tight resource budgets of edge platforms. ANT-V integrates a selected subset of the `Zve32x` vector extension into the open-source CVE2 core, adopting a monolithic, zero-duplication approach. To minimize hardware overhead, vector instructions execute as controller-driven micro-sequences over the scalar datapath, avoiding duplication of functional units. To eliminate the cost of a dedicated Vector Register File, vector registers are mapped to a software-reconfigurable region of the on-chip SRAM, trading raw bandwidth for area efficiency and runtime flexibility. Post-layout evaluation in $65\,\mathrm{nm}$ CMOS shows negligible system-level area overhead, with up to $17\times$ higher throughput and $16\times$ better energy efficiency compared to the scalar baseline on ML kernels. These results position ANT-V as an effective compromise, enabling faster edge ML inference on general-purpose embedded systems without the additional complexity of specialized hardware.**

***Index Terms*—RISC-V, Vector Processor, Edge Computing, Machine Learning, Embedded Systems.**

I. Introduction

THE growing demand for intelligent Internet-of-Things (IoT) end-nodes, driven by latency, privacy, and energy requirements, places increasing pressure on Micro-Controller Unit (MCU)-class processors operating under tight area and power budgets [1]. Domain-specific accelerators deliver high throughput on fixed dataflows, but their hardware cost, integration overhead, and limited adaptability to evolving Machine Learning (ML) models make them impractical for cost-sensitive edge platforms [2]. In recent years, the RISC-V "V" Vector Extension (RVV) [3] has emerged as an attractive middle ground between general-purpose cores and specialized accelerators, as it efficiently exploits Data-Level Parallelism (DLP), typical of ML kernels, while preserving a flexible, scalable, Vector Length Agnostic (VLA) programming model.

We present ANT-V, a monolithic vector-enhanced RISC-V processor built on the open-source CVE2 core [4]. The key idea is a *zero-duplication* integration in which each vector instruction executes as a short, controller-driven micro-sequence that reuses the scalar pipeline, Load-Store Unit (LSU), and arithmetic units. Furthermore, the Vector Register File (VRF) is mapped to a software-reconfigurable region of the on-chip SRAM, which generally serves as the primary system memory in deeply embedded contexts, rather than being a dedicated hardware structure. The combination of a shared-SRAM VRF with scalar-pipeline reuse is, to the best of our knowledge, unexplored in the RVV landscape and enables vector acceleration at a near-scalar area cost while still complying with the standard `Zve32x` embedded subset. The main contributions are:

- A lightweight, open-source [1] RISC-V vector processor implementing a `Zve32x` subset to enable ML inference at the edge while providing seamless system integration.
- A dynamically adjustable, memory-mapped VRF that exposes capacity/throughput trade-offs at run time, complemented by an optional indirect-addressing extension that enables Direct Memory Access (DMA)-compute overlap at negligible hardware cost.
- Post-layout ($65\,\mathrm{nm}$) and FPGA evaluation showing up to $17\times$ throughput and $16\times$ energy gains over CVE2, and up to $4\times$ over the DSP-extended CV32E40P, with a system-level area overhead below 1%.

TABLE I
Qualitative area, bandwidth, and flexibility trade-offs of data-parallel integration strategies.

Architecture	Area cost	Vector BW	Flexibility[a]	Main trade-off
Coprocessor [5], [6]	High	High (ded. VRF)	High	BW/perf. vs. area
Tightly coupled [7], [8]	Med.–High	High (ded. VRF)	High	BW vs. HW cost
Pipeline reuse [9], [10]	**Low**	**Low–Med. (shared)**	**High**	**Area vs. BW**
Fixed-SIMD [11], [12]	Low	Low–Med. (scalar)	Low	Area vs. flexibility

[a] Flexibility refers to the ability to support a programmable vector execution model across different vector lengths and data widths.

II. Related Work

Vector architectures can first be distinguished by how vector execution is integrated with the host processor; the related trade-offs are summarized in Table I. Decoupled coprocessors, such as Ara [5] and Spatz [6], employ dedicated vector datapaths and multi-ported VRFs, providing high operand bandwidth at the cost of substantial hardware overhead and integration complexity. Tightly coupled designs [7], [8] reduce this separation while retaining dedicated vector resources and therefore a similar bandwidth advantage. Pipeline-reused architectures instead exploit existing scalar resources [9], [10],

Flavia Guella, Guido Masera and Maurizio Martina are with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Torino, Italy (e-mail: name.surname@polito.it). Alessio Caviglia and Stefano Di Carlo are with the Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, 10129 Torino, Italy (e-mail: name.surname@polito.it). Michele Caon is with the Embedded Systems Lab, EPFL, Lausanne, Switzerland (e-mail: michele.caon@epfl.ch).
Flavia Guella and Alessio Caviglia contributed equally to this work.

[1] https://github.com/edge-group-polito/ANT-V

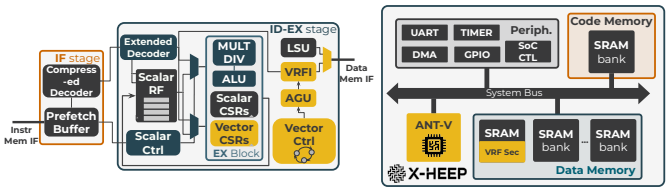


Fig. 1. ANT-V datapath (left) and integration in X-HEEP MCU (right). Yellow blocks are newly added, while blue blocks are modified pre-existing units.

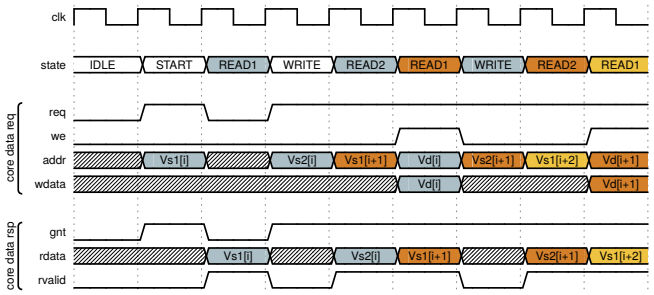


Fig. 2. Core-data interface requests/responses for a two-operand vector instruction. After the fill cycle, the first read of iteration $i{+}1$ overlaps with the write of iteration i. The state row shows the corresponding FSM sequencing.

substantially reducing area at the cost of lower vector-state bandwidth; ANT-V follows this approach, thereby minimizing additional hardware. Fixed-SIMD solutions [11], [12] provide inexpensive DLP but do not offer the VLA abstraction of RVV.

A second distinguishing feature is the organization of the vector state. Existing compact designs either retain a dedicated VRF [9], reuse the scalar register file [10], or reduce the physical VRF through caching techniques [13]. ANT-V removes the dedicated physical VRF altogether and directly maps the architectural vector registers onto a software-reconfigurable region of the system SRAM. This enables a large, software-configurable `VLEN` without a corresponding increase in core-local storage. Although this organization shares with some near-memory architectures the objective of dynamically using memory for both storage and vector state [14] [15], the two approaches are fundamentally different: near-memory systems typically rely on dedicated or specialized memory resources tightly coupled with computation, whereas ANT-V reuses the unmodified system SRAM and performs all computation in the scalar datapath. The resulting architecture therefore preserves a conventional `Zve32x`-compatible programming model.

III. ANT-V Architecture

ANT-V preserves the two-stage in-order pipeline of CVE2 and its system interface and is conceived as a *drop-in replacement* for the scalar core. Vector support, as depicted in Figure 1 (left), is implemented within the existing datapath rather than added as a parallel coprocessor, while scalar instruction execution remains unmodified and does not suffer any cycle-level penalty from the microarchitectural modifications.

A. Monolithic, Zero-Duplication Integration

Upon decoding a vector instruction, the scalar front-end is stalled and the vector controller takes over the pipeline for the entire micro-sequence. After latching the vector configuration, the controller and Address Generation Unit (AGU) iterate over vector slices, scheduling source and destination VRF accesses, driving the existing EX unit, and committing each result. Since scalar and vector execution are serialized and VRF accesses are statically scheduled, no dynamic hazard or write-back arbitration is required. The supported Instruction Set Architecture (ISA) is a `Zve32x` subset comprising bitwise, min/max, integer arithmetic, fused Multiply-Accumulate (MAC), permutation/sliding, and load/store instructions; resource-hungry operations infrequent in ML kernels [16], such as gather/scatter, are omitted.

B. Memory-Mapped, Reconfigurable VRF

Rather than instantiating a dedicated, multi-ported register file, the 32 architectural vector registers of ANT-V are mapped to a contiguous region of the system memory whose base address is software-configurable through a dedicated Control and Status Register (CSR). This choice (i) removes the area cost of a dedicated VRF, which dominates compact RVV implementations, (ii) lets software trade VRF capacity for general-purpose memory depending on the workload, and (iii) keeps ANT-V compatible with any embedded MCU that exposes its data memory through the existing core data interface, with no extra bus port. Accordingly, workloads that do not exercise vector code can reclaim the full SRAM for scalar use, while compute-intensive phases can expand the VRF up to the available `VLEN`. The main drawback of this organization is the limited bandwidth of the single-port VRF. To mitigate the latency associated with sequential vector element accesses, the vector controller implements an interleaved access protocol (Figure 2): instead of executing each 32-bit slice with a strict read–read–compute–write schedule, the first-operand read of iteration $i{+}1$ is performed in the idle cycle between the second read and write-back of iteration i. After a single pipeline-fill cycle, this hides one memory-read cycle per iteration, bringing the steady-state throughput of two-operand vector instructions to one 32-bit slice every three cycles.

C. LSU Reuse and Packed-SIMD Datapath

Vector load/store instructions reuse the scalar LSU: the vector controller issues one 32-bit-aligned request per cycle, while the AGU computes the next element address with an optional stride. Because the LSU and the VRF interface share the same single data-memory port and accesses are serialized by the controller FSM (Section III-A), core-internal conflicts cannot occur and the arbitration collapses to combinational multiplexing. The controller handles back-pressure from the system bus by stalling the vector pipeline, allowing ANT-V to operate correctly across arbitrary memory hierarchies.

The scalar arithmetic units are upgraded to partitioned, packed-Single Instruction Multiple Data (SIMD) operators. The 32-bit adder is built from four 9-bit slices whose inter-slice carries are gated by a small `SEW`-controlled multiplexer, as shown in Figure 3. With all three carries broken, the unit produces four independent 8-bit sums; with only the

central carry broken, it produces two 16-bit sums, and with all carries chained it produces a single 32-bit sum. The 32×32-bit multiplier is partitioned along the same byte boundaries, with the partial products selectively suppressed or combined according to the active `SEW`, and can be pipelined with the adder for MAC operations. A vector CSR block complements the scalar CSR unit to maintain `vtype`, `vl`, `vstart` and the element-width configuration, which the controller propagates to the partitioned units.

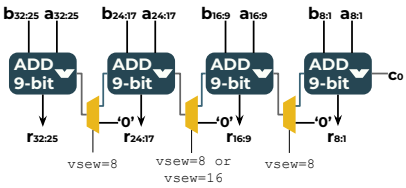


Fig. 3. Block diagram of the partitioned adder for packed-SIMD.

D. Indirect-Addressing Extension

At negligible hardware cost, requiring only additional decode logic, ANT-V optionally supports a custom vector ISA extension, inspired by the near-memory extension proposed in [14], in which the source/destination vector register indices are read from scalar registers instead of being encoded in the instruction. This unlocks a matrix-oriented kernel style in which large portions of the memory-mapped VRF are addressed by a single parametric kernel body, eliminating the need for loop unrolling and, combined with a programmable DMA engine, enabling double-buffered kernels in which VRF refill is overlapped with computation.

IV. Experimental Results

ANT-V is integrated into the open-source X-HEEP MCU [17], as shown in Figure 1 (right), synthesized with Synopsys Design Compiler® 2020.09, and placed-and-routed with Cadence Innovus® 22.33 using a low-power 65 nm CMOS library under worst-case conditions. Power is extracted with Synopsys PrimePower® 2020.09 using switching activity from post-layout simulation. FPGA validation is performed on a Xilinx Zynq UltraScale+ XCZU9EG. All software is compiled with GNU Compiler Collection (GCC) 15.2.0 (`-march=rv32imczve32x_zicsr`, `-O2`), with auto-vectorization disabled to allow assembly-level optimization of vector kernels.

A. System-Level Evaluation and Benchmarking

CVE2, the DSP-extended CV32E40P [12], and ANT-V are integrated into X-HEEP with 32 kB of instruction memory and four 32 kB data banks; for ANT-V, 16 kB of the last bank is reserved for the memory-mapped VRF at run time (`VLEN`=4096). Each System–on–Chip (SoC) is placed-and-routed at 200 MHz including SRAM macros.

ANT-V shows an area overhead of 21% compared to CVE2, mostly due to the additional decoding and operand multiplexing logic, while the vector controller and the VRF interface account for 7% of the total area, as shown in Figure 4. The only module influenced by the selected `VLEN` is the AGU accounting for less than 1% of the area. The system-level area overhead is limited to ≈1%, confirming the drop-in nature of the integration: since the VRF is part of the system memory, no on-chip storage is sacrificed beyond what software allocates.

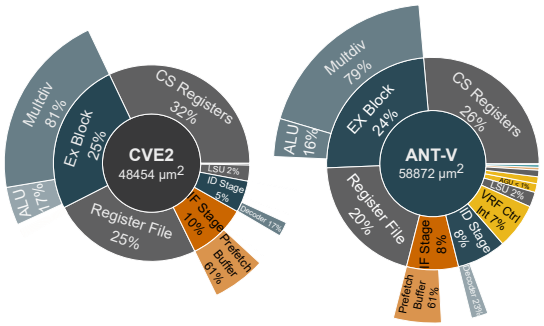


Fig. 4. Post-synthesis area split of baseline CVE2 (left) and ANT-V (right), @200 MHz. Percentages show the relative area occupation with respect to the parent module. Added blocks in yellow, and modified blocks in blue.

TABLE II
System-level throughput and energy comparison (post-layout) on X-HEEP SoC @200 MHz, 65 nm

Benchmark	Bit-Width	CVE2 (Baseline)		ANT-V Improvement	
		Cycles/Out	Energy/Out[b]	Throughput	Energy
Matrix Multiplication[a]	8-bit	151.2	9.69 nJ	**17.3×**	**16.6×**
	16-bit	151.2	9.85 nJ	9.7×	9.3×
	32-bit	145.2	9.64 nJ	5×	4.8×
Conv2D 3x3[a]	8-bit	143.17	9.30 nJ	**11.4×**	**11.4×**
	16-bit	147.23	9.48 nJ	6.6×	6.3×
	32-bit	157.20	10.31 nJ	3.7×	3.6×
ReLU[a]	8-bit	18.08	1.12 nJ	**8.2×**	**7.6×**
	16-bit	17.10	1.11 nJ	4.6×	4.4×
	32-bit	15.10	1.02 nJ	2.2×	2.2×

[a] $A[8,8] \times B[8,p]$, $p = \{64, 96, 128\}$. [b] Energy is estimated considering CPU, memory, and bus power contributions.

TABLE III
Comparison between CV32E40P and ANT-V in terms of computational efficiency.

Metric	CV32E40P	ANT-V
Peak throughput [GOPS][a]	0.1	**0.45**
Energy efficiency [GOPS/W] [b]	20.4	**150**
Area efficiency [GOPS/mm^2][b]	1.35	**7.6**

[a] 8-bit MAC operations are considered; each MAC is counted as two elementary operations; [b] Core-only.

Table II reports the average cycles- and energy-per-output measured by post-layout simulation, for the systems featuring CVE2 and ANT-V as host cores on three representative ML kernels. Across data widths, ANT-V delivers 2.2×–17.3× higher throughput and proportional energy improvement, with the largest gains on 8-bit kernels where packed-SIMD is fully exploited. The Rectified Linear Unit (ReLU) shows the smallest, yet still above 2×, improvement due to its simple computation, making it better suited for fusion with the preceding layer. To identify the architectural sources of these gains, we performed an ablation study on matrix multiplication (matmul). Disabling the interleaved VRF protocol reduces the

TABLE IV
COMPARISON WITH EXISTING RISC-V-BASED VECTOR PROCESSORS.

	Spatz [6]		**RS5 + vec** [7]		**ANT-V** (this work)
Integration	Coprocessor		Tightly-Coupled		Pipeline-reuse
VRF type	Multi-port, multi-bank (latch)		Multi-port, multi-bank		Single-port, shared SRAM bank
VRF size, #lanes, `VLEN` (bit)	1 kB, 2 lanes, 256		256 B, 2 lanes, 64		1×16 kB, 1 lane, 4096
ALU features	$2\times$ 32-bit SIMD ALU		$2\times$ 32-bit SIMD ALU		$1\times$ 32-bit SIMD ALU
Supported operations	`Zve32x` subset		`Zve32x` subset		`Zve32x` subset + custom
Technology (nm)	22	65[a]	28	65[a]	65
Area [μm^2] [b]	$64 \cdot 10^3$	$561 \cdot 10^3$	$37 \cdot 10^3$	$201 \cdot 10^3$	$\mathbf{170 \cdot 10^3}$
Nominal freq. [MHz]	500	200[c]	500	200[c]	200
Peak throughput[d] [GOPS]	1.92 [e]	**0.768** [e]	1	0.4	0.45
Energy eff. [GOPS/W]	126.6 [e]	**76.9** [e,f]	N/A	N/A	48 [g]
Area eff. [GOPS/mm^2]	30 [e]	1.37 [e]	27.02	2	**2.65**

[a] Area at 65 nm is computed from 22 nm and 28 nm using the most conservative standard-cell scaling factor between $(\frac{65}{tech})^2$ and [18]. [b] Area includes the VRF of the reported size. [c] Frequency is rescaled to our constraints, which should represent a worst-case assumption for our design. [d] 8-bit MACs are considered, unless otherwise specified, each MAC is counted as two elementary operations. [e] Throughput and energy on 32-bit MACs estimated in [19]. [f] Energy is rescaled according to the smallest factor between standard technology scaling and [18]. [g] Power extracted from post-layout simulation on a matmul kernel.

speedup from $5\times$ to $\approx 3.9\times$, corresponding to a $1.28\times$ contribution from interleaving. Packed-SIMD provides a further $3.5\times$ throughput gain, below the ideal $4\times$ due to control and vector load/store overheads. We further quantified the impact of serialized scalar–vector execution, which becomes negligible for longer vectors: the potential benefit of overlap drops from $1.25\times$ at 8 elements to $\approx 1.01\times$ at 128 elements. Finally, the system-level power increase is kept below 5% across all evaluated benchmarks, confirming that ANT-V is suited for deployment in power-constrained devices.

Compared with CV32E40P running matmul kernels optimized with dot product built-ins, ANT-V achieves $4\times$ peak throughput, $7.3\times$ better energy efficiency, and $5.6\times$ better area efficiency, as reported in Table III, showing that the VLA model scales with input size while fixed-width packed-SIMD saturates (Figure 5). When considering the contribution of the 16 kB VRF, which represents a worst-case estimate, the efficiency gain remains above $2\times$.

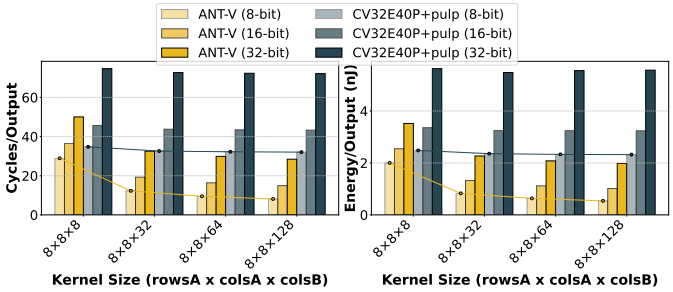


Fig. 5. Cycles/output and energy/output for *matmul* of reported sizes on ANT-V and CV32E40P with `XPulpV2` extensions. Lower is better.

On an end-to-end 8-bit autoencoder anomaly-detection model [20], [21], comprising ten dense layers interleaved with ReLU activations, ANT-V reaches $8.3\times$ throughput and $9\times$ energy improvement over CVE2, and $4\times$/$4.6\times$ over CV32E40P. The gap with respect to the kernel-level results in Table II is due to the prevalence of vector–matrix products in the dense layers for which our kernel scales less favorably than for matrix–matrix products.

B. FPGA Characterization

The complete X-HEEP SoC is emulated on the XCZU9EG with AMD Vivado® 2024.2 across a diverse benchmark suite spanning ML kernels and general vectorized workloads, confirming correct execution across multiple ANT-V configurations and vector-length parameterizations. Post-implementation characterization on the standalone core yields $f_{\max} = 130$ MHz, with a resource footprint of 5336 CLB LUTs, 2877 registers, and 14 DSPs. At 130 MHz, ANT-V delivers a peak throughput of 0.34 GOPS on 8-bit matmul, demonstrating viable performance portability beyond the primary ASIC target.

C. Memory-Access Overhead Analysis

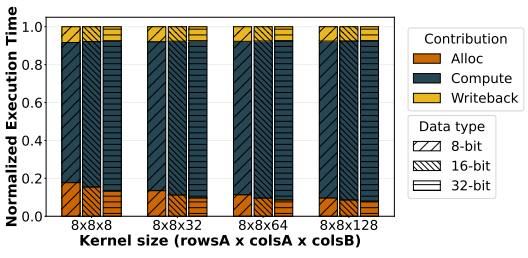


Fig. 6. Relative breakdown of *matmul* kernel cycles into allocation, computation and write-back, for varying matrix sizes, and `SEW`.

To evaluate the impact of the limited bandwidth offered by the memory-mapped VRF on the kernel execution, Figure 6 breaks down the cycle count of the matmul kernel into three phases: *allocation*, which loads operands from the memory region into the VRF; *computation*, which performs vector arithmetic on VRF-resident data; and *write-back* (stores results back to memory). Across all evaluated matrix sizes and data

types, the allocation phase stays below 20% of the kernel cycles and the write-back below 5%, leaving more than 75% to actual computation. The relative cost of memory transfers further decreases for larger matrices and narrower data types, as transfer overhead is increasingly amortized. This confirms that the interleaved access protocol of Sec. III-B is sufficient to amortize the single-port VRF, and motivates the indirect-addressing extension of Sec. III-D as a complementary mechanism to overlap allocation with computation through DMA.

D. Comparison with RISC-V Vector Processors

Table IV compares ANT-V with representative RISC-V vector designs targeting embedded devices and ASIC implementations. Notably, ANT-V integrates a significantly larger logical VRF than the other designs. Although peak performance does not require the full `VLEN`, the reported results include the full VRF footprint, representing a strict worst-case condition for area and energy efficiency. ANT-V exceeds the peak throughput of the tightly coupled RS5. With the smallest VRF configuration preserving the estimated peak throughput, corresponding to $\mathrm{VLEN} = 1024$, the area efficiency increases to 4 GOPS/mm^2 ($2\times$ that of RS5). Spatz serves as a reference decoupled implementation, achieving the highest throughput through a latch-based, multi-port VRF and two execution lanes, at over $3\times$ the area of ANT-V. While Spatz favors parallel bandwidth, ANT-V minimizes hardware cost through datapath reuse and a shared VRF.

Among FPGA-only pipeline-reuse designs, ANT-V requires approximately $1.6\times$ more LUTs than RVVe [10], reflecting RVVe's narrower custom ISA and simpler element-counter-based operand sequencing. Despite a lower clock frequency (130 vs. 200 MHz), ANT-V achieves higher peak throughput (0.34 vs. 0.2 GOPS). Compared with the fixed-SIMD implementation in [11], ANT-V achieves $12\times$ higher throughput at under half the LUT cost. Overall, these results validate ANT-V's intended design point: sacrificing peak vector bandwidth in exchange for an extremely compact, drop-in vector processor retaining a standard VLA programming model.

V. Conclusion

We presented ANT-V, a monolithic RISC-V vector processor that augments the CVE2 core with a `Zve32x` subset at near-scalar hardware cost. By executing vector instructions as controller-driven micro-sequences over the reused scalar pipeline and by mapping the VRF to a software-reconfigurable region of the on-chip SRAM, ANT-V achieves up to $17\times$ higher throughput and $16\times$ better energy efficiency than the scalar baseline, with a system-level area overhead below 1%. ANT-V shows the smallest area footprint among the compared designs. These results show that RVV-based extensions can be added to MCU-class cores as a compact drop-in solution, enabling efficient ML execution at the edge without committing to a fixed-function accelerator. Future work will target higher throughput with limited area overhead by introducing dedicated interfaces to a multi-ported VRF.

References

[1] V. Tsoukas *et al.*, "A Review on the emerging technology of TinyML," *ACM Comput. Surv.*, vol. 56, no. 10, Jun. 2024.

[2] P. Gao *et al.*, "OASIS: A Commercial High Performance Terminal AI Processor Supporting RISC-V Tensor Extension Instructions," in *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '25, New York, NY, USA, 2025, p. 1264–1283.

[3] RISC-V International, *RISC-V Instruction Set Manual, Volume I. Unprivileged Architecture. RISC-V "V" Vector Extension, Version 1.0*, Nov 2021. [Online]. Available: https://lists.riscv.org/g/tech-vector-ext/attachment/691/0/riscv-v-spec-1.0.pdf

[4] P. D. Schiavone *et al.*, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[5] M. Perotti *et al.*, "Ara2: Exploring Single- and Multi-Core Vector Processing With an Efficient RVV 1.0 Compliant Open-Source Processor," *IEEE Transactions on Computers*, vol. 73, no. 7, p. 1822–1836, Jul. 2024.

[6] ——, "Spatz: Clustering Compact RISC-V-Based Vector Units to Maximize Computing Efficiency," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 7, pp. 2488–2502, 2025.

[7] W. A. Nunes *et al.*, "Accelerating machine learning with risc-v vector extension and auto-vectorization techniques," in *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2025, pp. 1–5.

[8] C. Wang *et al.*, "SPEED: A Scalable RISC-V Vector Processor Enabling Efficient Multiprecision DNN Inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 1, pp. 207–220, 2025.

[9] M. Stanic *et al.*, "Towards low-power embedded vector processor," in *Proceedings of the ACM International Conference on Computing Frontiers*, ser. CF '16, New York, NY, USA, 2016, p. 339–342.

[10] P. Schmidt *et al.*, "RVVe: A Minimal RISC-V Vector Processor for Embedded AI Acceleration," in *2024 IEEE 37th International System-on-Chip Conference (SOCC)*, 2024, pp. 1–6.

[11] M. S. Bonet *et al.*, "SPARROW: A Low-Cost Hardware/Software Co-designed SIMD Microarchitecture for AI Operations in Space Processors," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1139–1142.

[12] M. Gautschi *et al.*, "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 10, p. 2700–2713, Oct. 2017.

[13] V. Titopoulos *et al.*, "Register Dispersion: Reducing the Footprint of the Vector Register File in Vector Engines of Low-Cost RISC-V CPUs," in *Proceedings of the 22nd ACM International Conference on Computing Frontiers*, ser. CF '25, New York, NY, USA, 2025, p. 3–11.

[14] M. Caon *et al.*, "Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 3, pp. 1003–1018, Jul. 2025.

[15] X. Liu *et al.*, "Hnmc: Hybrid near-memory computing circuit for neural network acceleration," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 11, pp. 4763–4767, 2024.

[16] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," 06 2018, pp. 2704–2713.

[17] P. D. Schiavone *et al.*, "X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, ser. CF '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 379–380.

[18] S. Sarangi *et al.*, "DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era," in *2021 IEEE International Symposium on Circuits and Systems*, 2021, pp. 1–5.

[19] M. Cavalcante *et al.*, "Spatz: A compact vector processing unit for high-performance and energy-efficient shared-l1 clusters," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3508352.3549367

[20] mlcommons, "Mlperf tiny," 2020, accessed: March 7, 2026. [Online]. Available: https://github.com/mlcommons/tiny

[21] DCASE2020, "Unsupervised detection of anomalous sounds for machine condition monitoring," 2020, accessed: March 7, 2026. [Online]. Available: https://dcase.community/challenge2020/task-unsupervised-detection-of-anomalous-sounds