RISC-V ISA Extensions with Hardware Acceleration for Hyperdimensional Computing

Rocco Martino[1], Marco Angioli[1], Antonello Rosato[1], Marcello Barbirotta[1]
Abdallah Cheikh[1] and Mauro Olivieri[1]

[1]Dept. of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, Italy

Abstract

Hyperdimensional Computing (HDC) leverages high-dimensional distributed representations called hypervectors (HVs) and simple arithmetic operations, making it an ideal paradigm for learning tasks on resource-constrained devices. This work introduces the first RISC-V Instruction Set Architecture (ISA) extension specifically designed to execute all fundamental arithmetic operations of HDC directly through dedicated instructions, which, when appropriately combined, enable a variety of learning tasks by efficiently encoding and processing information. This extension is coupled with a specialized hardware acceleration unit, integrated into the Klessydra-T03 RISC-V core, to perform computations on binary HVs efficiently. The proposed solution enables a seamless trade-off between execution time and hardware resource utilization through both synthesis-time configurability and runtime programmability. The custom ISA extension is fully integrated into the RISC-V GCC toolchain, allowing software developers to exploit its capabilities via intrinsic function calls. Benchmarking on an FPGA platform demonstrates significant performance improvements across a wide range of HDC tasks, from basic arithmetic operations to real-world classification problems.

Introduction

Hyperdimensional Computing (HDC), also known as Vector Symbolic Architectures (VSA), is a symbolic computing paradigm that represents information through distributed high-dimensional representations called hypervectors (HVs) [1]. These are processed using three fundamental arithmetic operations and one comparison operation—bundling, binding, permutation, and similarity—which define the mathematical space where information is encoded and manipulated. By appropriately combining these operations, HDC enables various learning tasks, including classification, clustering, and regression.

We present the first RISC-V instruction set architecture (ISA) extension specifically designed to cover the direct execution of fundamental HDC operations in hardware. The new instructions, listed in Table 1, are exposed to the programmer via intrinsic functions and fully integrated into the GCC toolchain.

The ISA extension controls the Hyperdimensional Coprocessor Unit (HDCU), a configurable hardware accelerator integrated into the execution stage of the Klessydra T03 RISC-V [2] core (Figure 1). The highly configurable architecture allows customization at both synthesis time and runtime. At synthesis time, users can define hardware parallelism, memory size, and supported operations to balance resource utilization and performance. At runtime, the size of HVs and the number of operations can be dynamically configured via dedicated Control Status Registers (CSRs), enabling the same hardware to adapt to varying workloads.

Description of the Solution

Building on the RISC-V ISA extension, the HDCU executes HDC operations in hardware, optimizing performance and energy efficiency, with each instruction controlling a dedicated functional unit (FU) to accelerate operations such as bundling, binding, and permutation. Integrated into the Klessydra-T03 core, the HDCU operates in parallel with the Arithmetic Logic Unit (ALU) and Load Store Unit (LSU), leveraging lightweight multi-threading for efficient workload distribution. Upon fetching an HDC instruction, the core checks unit availability and delegates execution if idle, enabling concurrent operation of different FUs. The HDCU employs Single Instruction Multiple Data (SIMD) parallelism, allowing multiple HV elements to be processed in a single cycle. The SIMD width, configurable at synthesis time, balances execution speed and hardware resources (i.e. SIMD=256 processes 256 HV elements concurrently.)

The custom RISC-V ISA extension ensures seamless interaction between software and hardware. High-level intrinsic functions translate into low-level instructions, offloading computationally intensive tasks to the HDCU with minimal processor intervention. To enhance efficiency, hardware loops process HV elements iteratively without redundant instruction fetches. Controlled by CSRs, these loops adjust execution based on HV size and parallelism, reducing overhead and enabling efficient execution of complex tasks like encoding and associative search.

Hypervectors are stored in dedicated local Scratch-

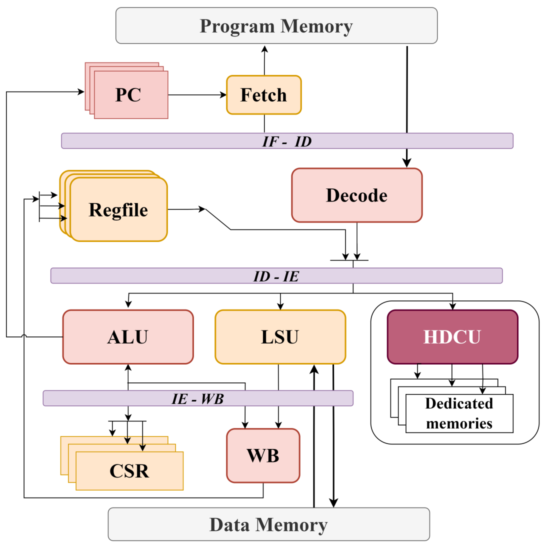

Figure 1: *Integration of the HDCU in the Klessydra core. The accelerator works in parallel with the ALU and the LSU during the execution stage, acting like a coprocessor.*

pad Memories (SPMs) for high-bandwidth and low latency access. Data transfers between main memory and SPMs are managed by dedicated load/store instructions.

Results and Impact

The Klessydra-T03 core, including the HDCU, was synthesized and tested on a Xilinx Zynq UltraScale+ ZCU106 (EK-U1-ZCU106-G) FPGA board to evaluate its performance on key HDC tasks. Benchmarking results, obtained using a custom testing library, show substantial speedups, defined as the ratio of cycles required for software execution with standard C code to those using dedicated hardware.

As shown in Table 2, the HDCU provides substantial acceleration across core operations, achieving up to 2665× speedup for HV sizes of 8192 with SIMD=1024. This demonstrates the architecture's scalability and efficiency for computationally intensive tasks.

In real-world scenarios, such as classification, the HDCU significantly reduced execution time across the HDC pipeline, including encoding, training, and inference. For instance, it achieved 297× speedup on the CARDIO dataset [3] and 2438× on the EMG dataset [4] during inference. These results underscore its adaptability for diverse applications, from low-power edge devices to high-performance systems.

Conclusion

This work presented the Hyperdimensional Coprocessor Unit (HDCU), a flexible hardware accelerator for Hyperdimensional Computing (HDC) integrated into the Klessydra-T03 core via a custom RISC-V ISA extension. The HDCU efficiently executes fundamental HDC operations while ensuring software programmability.

Benchmarking showed up to 2665× speedup in core operations and over 2438× in real-world classification tasks. Its configurable design, with synthesis-time parallelism tuning and runtime adaptability via CSRs, balances execution speed and hardware resource use.

Beyond computational efficiency, the HDCU offers a versatile framework for HDC, bridging domain-specific acceleration and flexibility. Future work will refine memory hierarchy, expand ISA capabilities, and support additional learning paradigms for broader adaptability.

Table 1: *Custom RISC-V Instructions for HDC*

Instruction	Description
`hvbundle(rd, rs1, rs2)`	Bundle the N-bit HV in `rs1` with the binary HV in `rs2`, storing the result in `rd`.
`hvbind(rd, rs1, rs2)`	Bind HVs in `rs1` and `rs2`, producing a new HV in `rd`.
`hvperm(rd, rs1, rs2)`	Permute the HV in `rs1` by `rs2` positions, storing the result in `rd`.
`hvsim(rd, rs1, rs2)`	Compute Hamming distance between HVs in `rs1` and `rs2`, storing the similarity in `rd`.
`hvclip(rd, rs1, rs2)`	Binarize the HV in `rs1` using threshold `rs2`, storing the result in `rd`.
`hvsearch(rd, rs1, rs2)`	Compare the HV in `rs1` with all class HVs in `rs2`, storing the closest match in `rd`.
`hvmemld(rd, rs1, size)`	Load an HV from memory location `rs1` into the SPM at `rd`, for `size` bytes.
`hvmemstr(rd, rs1, size)`	Store an HV from the SPM at `rs1` to memory location `rd`, for `size` bytes.

Table 2: *Hardware utilization and speedup factors for HDC operations across different degrees of parallelism (SIMD), showing the scalability of the proposed HDCU design with HV sizes ranging from 32 to 8192.*

SIMD [bit]	#LUTs	#FFs	f[MHz]	Bind	Perm.	Bundle	Sim.
32	1030	450	234	26.64×	18.80×	120.56×	157.38×
64	2040	651	221	50.49×	35.17×	237.69×	298.26×
128	3243	868	218	91.39×	62.28×	462.23×	539.90×
256	4016	1101	215	153.62×	101.37×	875.97×	907.49×
512	13811	3731	160	232.91×	147.71×	1585.62×	1375.88×
1024	34189	7478	140	313.92×	191.48×	2665.20×	1854.44×

References

[1] Denis Kleyko et al. "Vector symbolic architectures as a computing framework for emerging hardware". In: *Proceedings of the IEEE* 110.10 (2022), pp. 1538–1571.

[2] Abdallah Cheikh et al. "Klessydra-T: Designing Vector Coprocessors for Multithreaded Edge-Computing Cores". In: *IEEE Micro* 41.2 (2021), pp. 64–71. DOI: `10.1109/MM.2021.3050962`.

[3] D. Campos and J. Bernardes. *Cardiotocography*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51S4N. 2010.

[4] A. Rahimi. *EMG Dataset*. Version 1.0. Jan. 2024. URL: `https://github.com/abbas-rahimi/HDC-EMG/blob/master/dataset.mat`.