ECE-718: An Extensible Architecture for Hyperdimensional Computing

1st Graham Power
Electrical & Computer Engineering
McMaster University
Hamilton, Canada
powerg@mcmaster.ca

I. Introduction

Hyperdimensional computing is an encoding which stores data in very high dimensional vectors, on the order of thousands to tens of thousands of dimensions. While there are many different ways to represent these vectors [1], called encodings, these encodings generally support three fundamental operations. Similarity, a method for comparing two vectors to determine their elements' similarity, generally produces a number bounded of [-1,1]. Bundling, which takes two or more vectors and produces a vector similar to all the inputs, as defined by the similarity metric. Binding, which takes two vectors and produces a vector dissimilar to all the inputs, again as defined by the similarity metric. Usually, binding is commutative and invertible, though there are some exceptions. In VTB [2], MBAT [3], and BSDC-S [4], binding is not commutative. In HRR [5], MBAT [3], and VTB [2] binding is only pseudo-invertable, and in BSDC-CDT [4] binding is entirely non-invertable.

A. The Problem

While many previous works have implemented HDC accelerators [6]–[8], they have all been restricted to a single encoding, mainly binary. This project aims to create an HDC accelerator that supports multiple hypervector encodings and is built in a way that can be easily extended to support additional encodings in the future. If time allows, an AXI interface will be implemented, allowing the accelerator to be used by a processor.

II. Proposed Architecture

The accelerator will be broken down into sub-kernels for easier implementation. Each kernel will be capable of performing a similarity, bundling, or binding operation on hypervectors one dimension at a time. The accelerator will also contain hypervector memories for direct access to hypervectors without going through the AXI bus.

A. Kernel-Level Architecture

Three core kernel types will be built for this accelerator: similarity, bundling, and binding. All kernels will be given two dedicated ports to the hypervector memory whose bit width is equal to the width of a single dimension. One port will be a dedicated read port, the other a dedicated write port.

The similarity kernel consists of three stages. First, a single dimension of A is loaded into a register from memory, then B is read from memory and used directly with a to calculate a single-dimension similarity metric. This metric is then loaded into a similarity accumulation register, and the process is repeated for every dimension in the hypervector. Finally, the accumulated similarity is formatted (ex., the final square root calculation in cosine similarity) and written back to the memory.

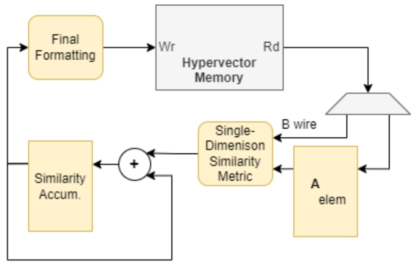

Fig. 1. Architecture of a Single Similarity Kernel

The bundling kernel consists of two stages. The initial dimension is first loaded into the bundled element register. Then, for every addition hypervector being bundled, it is loaded from memory, bundled, and accumulated into the bundled element register. When complete, the final bundled vector is written back to the memory.

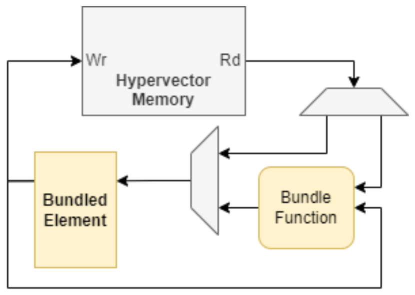

Fig. 2. Architecture of a Single Bundle Kernel

The binding kernel consists of two stages. The initial hypervector is first loaded from the memory into the A_elem register. Then, the second hypervector is loaded from memory and used with the first in the bind function before being written back to the memory.

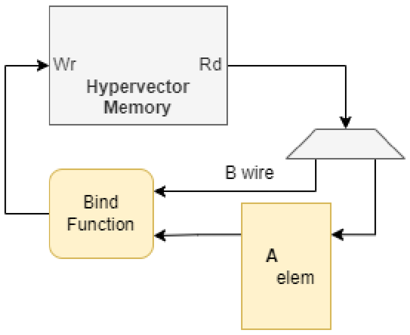

Fig. 3. Architecture of a Single Bind Kernel

B. System Architecture

All kernels will share a single hypervector memory. This is both to reduce FPGA area and to allow hypervectors to be bound, bundled, and compared without transferring entire hypervectors between kernel-specific memories. The similarity kernels will also share a dedicated similarity cache to store the results of similarity comparisons. This independent cache was chosen because the result of a similarity comparison is always a single number, not an entire hypervector, and may require a different bit-width than required to store the dimensions of a hypervector.

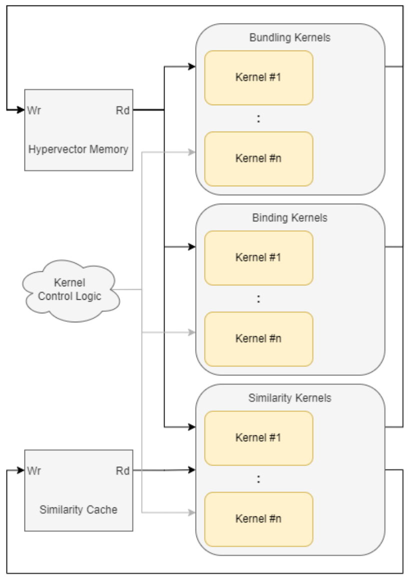

Fig. 4. High-Level Archictecture of the Hyperdimensional Computing Accelerator

C. AXI Interface

The AXI interface will expose four registers to the processor. One register will be for loading/storing hypervectors from/to the internal hypervector memory. There will also be registers to initiate hypervector bind, bundle, and similarity operations. Finally, there will be a register to check the status of previously initiated operations. This will be used to ensure the result of an operation only gets used or read elsewhere after it is completed.

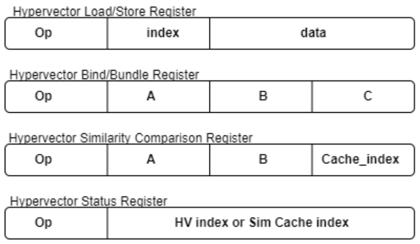
Hypervector Load/Store Register
Op
index
data
Hypervector Bind/Bundle Register
Op
A
B
C
Hypervector Similarity Comparison Register
Op
A
B
Cache_index
Hypervector Status Register
Op
HV index or Sim Cache index

Fig. 5. Registers Exposed to the Processor by the AXI Interface

References

[1] K. Schlegel, P. Neubert, and P. Protzel, "A comparison of vector symbolic architectures," *Artificial Intelligence Review*, vol. 55, no. 6, p. 4523 – 4555, 2022, cited by: 16; All Open Access, Green Open Access, Hybrid Gold Open Access.

[2] J. Gosmann and C. Eliasmith, "Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks," *Neural Comput.*, vol. 31, no. 5, p. 849–869, may 2019.

[3] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Comput.*, vol. 25, no. 8, p. 2038–2078, aug 2013.

[4] D. A. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Trans. on Knowl. and Data Eng.*, vol. 13, no. 2, p. 261–276, mar 2001.

[5] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, p. 623 – 641, 1995, cited by: 329; All Open Access, Green Open Access.

[6] F. Taheri, S. Bayat-Sarmadi, and S. Hadayeghparast, "Risc-hd: Lightweight risc-v processor for efficient hyperdimensional computing inference," *IEEE Internet of Things Journal*, vol. 9, no. 23, p. 24030 – 24037, 2022, cited by: 0.

[7] M. Imani, S. Salamat, B. Khaleghi, M. Samragh, F. Koushanfar, and T. Rosing, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," 2019, Conference paper, p. 190 – 198, cited by: 25.

[8] M. Imani, J. Morris, J. Messerly, H. Shu, Y. Deng, and T. Rosing, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," 2019, Conference paper, cited by: 27; All Open Access, Bronze Open Access.