Domain Specific Architectures

Processor-In-Memory (PIM) Architectures

Guest Lecturer

MD Arafat Kabir
Summer Graduate, PhD
Advisor: Dr. David Andrews
Dissertation: Deep-learning FPGA accelerator using PIM architecture

Earliest Publications

IEEE Transactions on Computers, 19

Cellular Logic-in-Memory Array

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be "programmed" to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.

digital computers and informat
Advantages can be obtained by
cuitry in the form of a cellular
sional iterative configuration of
which contains both logic and st
mainly to its immediate neighbo
therefore, has the form of a me
hanced with logic at each digit
tures of these cellular logic-in-m
are described. By way of an out
balance of this paper then descri
cal circuitry, operation, and use o
a cellular sorting array, that has
versatile and efficient than seve
lular logic-in-memory arrays un
array can be employed not only
ing memory which keeps in order
it, but also as a pushdown mer
memory, a content-addressed m

Computer, 1995

Processing in Memory: The Terasys Massively Parallel PIM Array

Maya Gokhale
David Sarnoff Research Center*

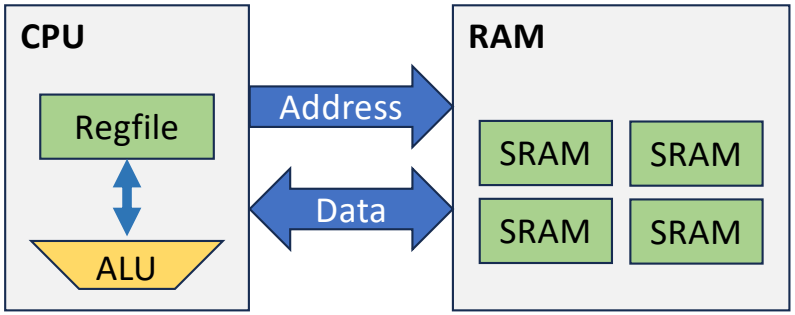
Bill Holmes and Ken Iobst
Supercomputing Research Center

**The work reported here was done while the author was at the Supercomputing Research Center, Bowie, Maryland.*

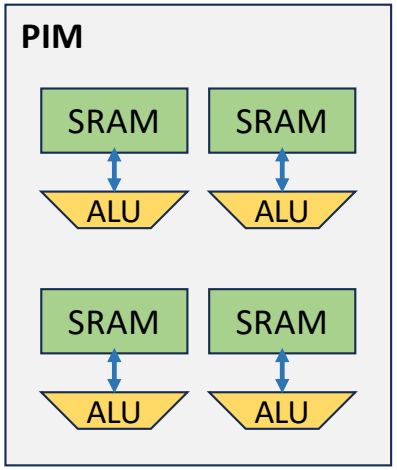
SIMD processor arrays provide superior performance on fine-grained massively parallel problems in which all parallel threads do the same operations most of the time. However, this fine-grained synchrony limits the application space of SIMD (single instruction, multiple data) machines. If there are many alternative data-dependent actions among the parallel threads, the total execution time is the *sum* of the alternatives rather than the maximum single-thread execution time. Additionally, if the application is not inherently load-balanced, performance can degrade seriously: Most of the processors finish their work quickly and become idle, while a few processors end up with the lion's share of the work. Thus, the economics of purchasing a high-performance computer often dictate giving up peak performance on a small application set (massively parallel SIMD) in favor of more modest improvement over a larger range of applications (general-purpose

What is Processor-in-Memory (PIM)

- Main idea: Put memory and processor together (same chip)



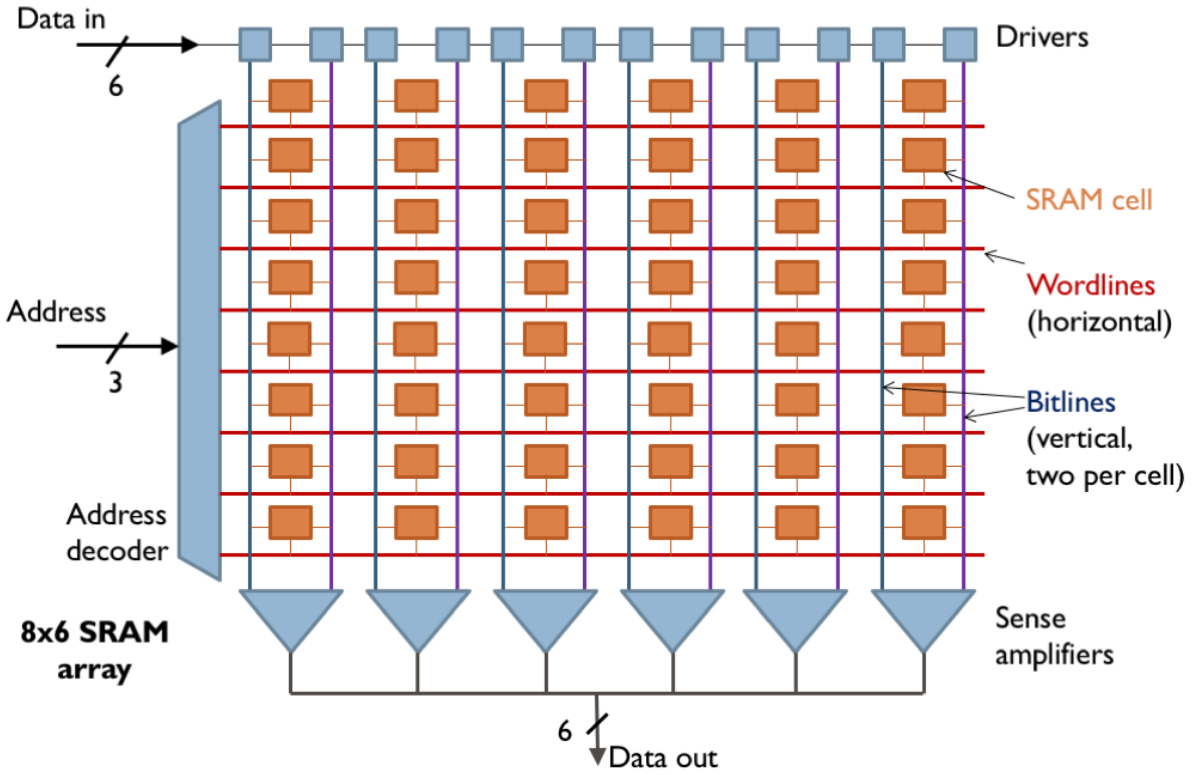
Classic von Neumann Architecture



PIM Architecture

SRAM Array

- Image source: Google search



A Practical PIM Design: Cache-based

- C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "**Neural Cache**: Bit-Serial in-Cache Acceleration of Deep Neural Networks," in 2018 ACM/IEEE 45Th annual international symposium on computer architecture (ISCA), 2018, pp. 383–396.

Why Do We Need PIM Architectures?

- What happens in the CPU when you add two vectors?
  - C = A+B; A,B,C are vectors of 1000 elements

- `result[i] = array1[i] + array2[i];`
  - `read array1[i] to register R1`
  - `read array2[i] to register R2`
  - `add R1, R2 and save to R3`
  - `Write R3 to result[i]`

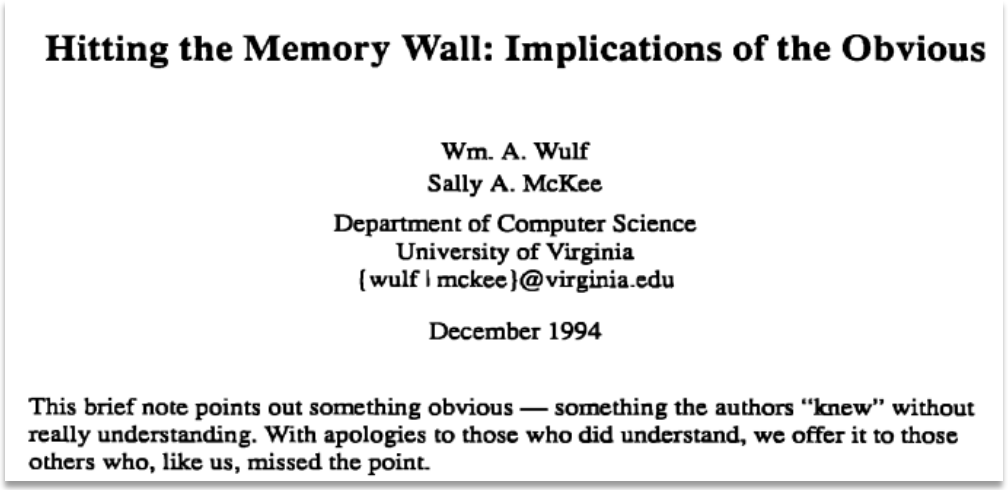
- `for (int i = 0; i < size; i++)`
  - `increment i (R4)`
  - `read size to register R5`
  - `compare R4, R5`
  - `make a conditional jump`

Most of the instructions (90%) are spent on control sequence!

```
void add_vector(int array1[], int array2[],
                int result[], int size)
{
    for (int i = 0; i < size; i++) {
        result[i] = array1[i] + array2[i];
    }
}
```

Why Do We Need PIM Architectures?

- "The Memory Wall"
  - Memory operations (read/write) are significantly slower than CPU operations.
- Cycle latency from Wikipedia
  - Corei9: 0.16 ns
  - DDR5: 14 ns



Hitting the Memory Wall: Implications of the Obvious

Wm. A. Wulf
Sally A. McKee

Department of Computer Science
University of Virginia
{wulf | mckee}@virginia.edu

December 1994

This brief note points out something obvious — something the authors "knew" without really understanding. With apologies to those who did understand, we offer it to those others who, like us, missed the point.

Why Do We Need PIM Architectures?

- Memory access cost
  - 100x slower
  - 100x more energy
- Narrow bus connection
  - Limited bus frequency
  - Limited number of pins
  - One word at a time
- Application requirements
  - Different access patterns

Domain-specific PIMs,

- Smaller arrays can be faster
- PIM can minimize energy cost
- Not limited by bus frequency or pin count
- Match application access pattern
- Massive parallelism

Key Takeaway

PIM architectures can help with memory-intensive applications.

Domain Specific Architectures

Case Study: PIM Architectures for FPGAs

Guest Lecturer

MD Arafat Kabir
Summer Graduate, PhD
Advisor: Dr. David Andrews
Thesis: Deep-learning FPGA accelerator using PIM architecture

A Modern FPGA Internals

- What is an FPGA?

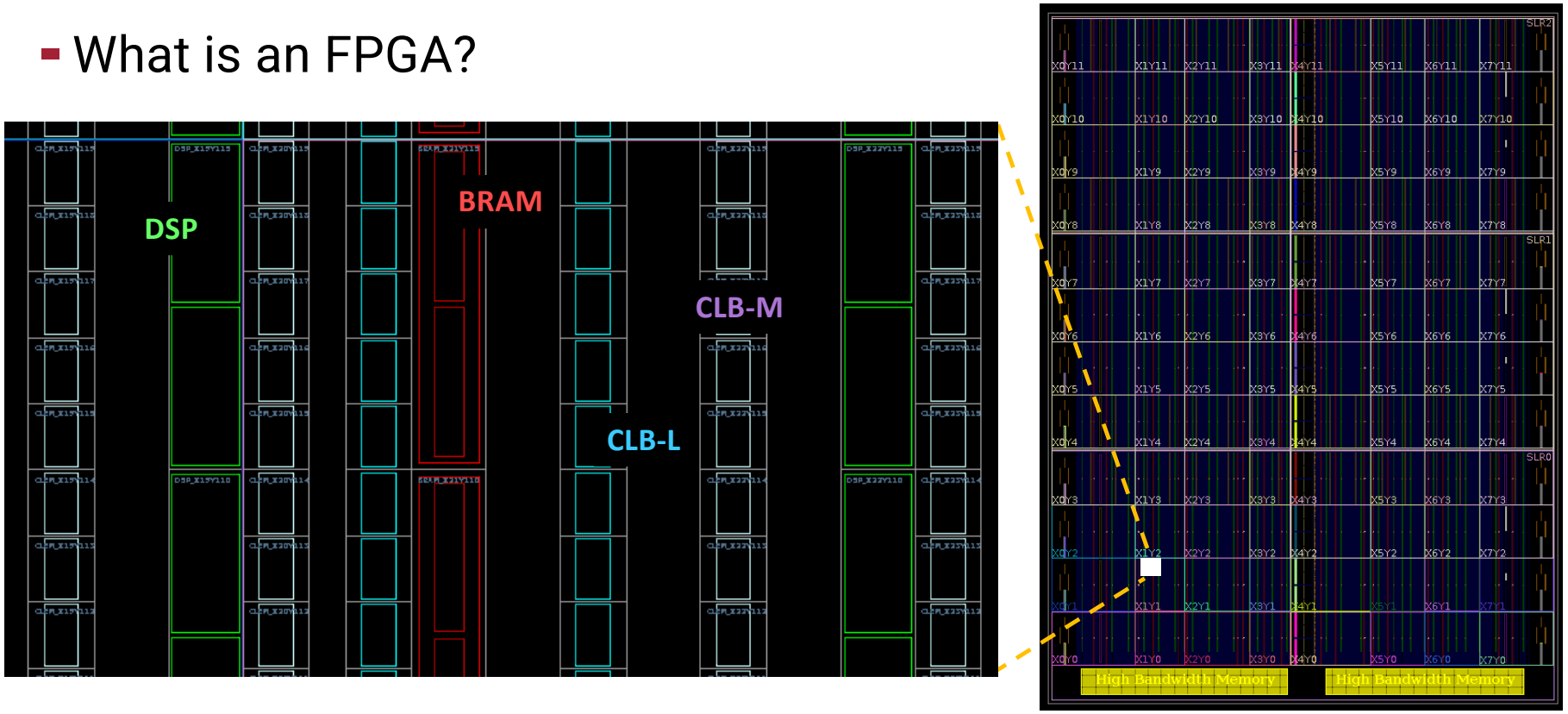
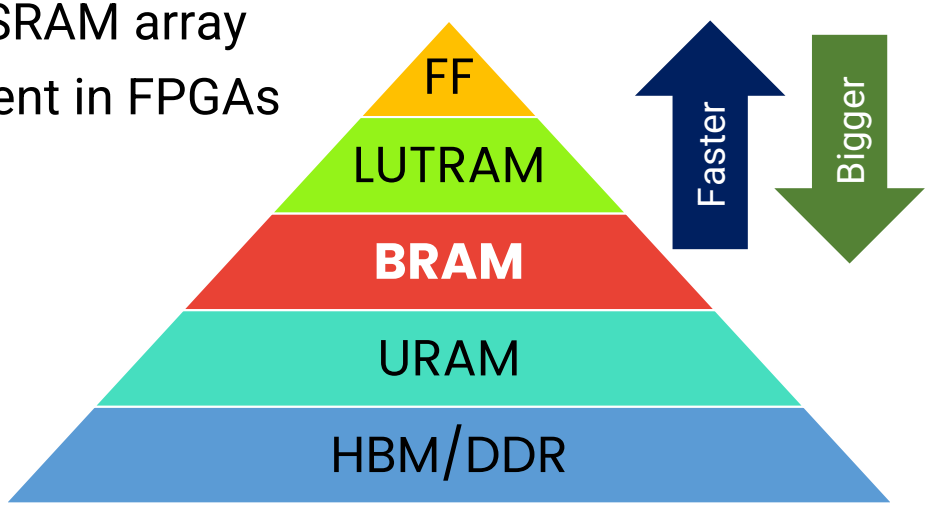


Figure: AMD (Xilinx) Alveo U55 Datacenter Card (Virtex UltraScale+)

Terminologies and Acronyms

- **Accelerator:** custom hardware for a specific application
- **PIM, CIM, IMC:** Processing-in-Memory, In-Memory Computing
- **PE:** Processing Element
- **Overlay:** Virtual architecture on top of reconfigurable fabric of FPGA
- **BRAM:** Block-RAM, 18K/20K/36K bit SRAM array
- **LUT:** A lookup-table based logic element in FPGAs



Case Study: CCB/RIMA

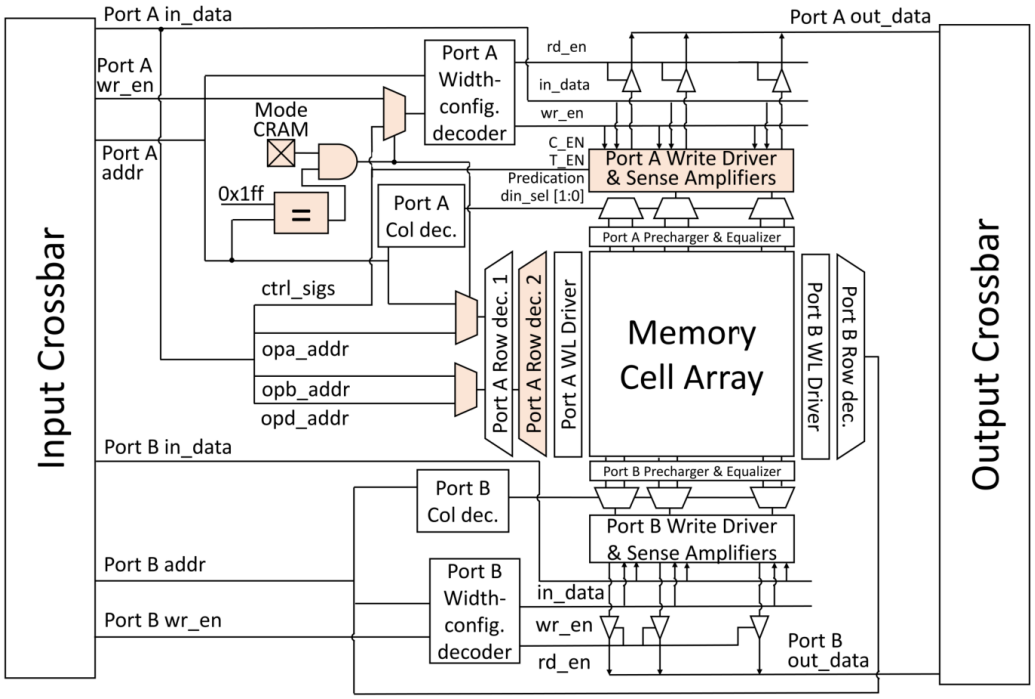
- Motivations:
  - **C**ompute **C**apable **B**RAM (CCB)
  - Modifying BRAMs to implement CCB (PIM)
  - The first proposal of custom-BRAM PIM for modern FPGAs
  - **R**econfigurable **I**n-**M**emory **A**ccelerator (RIMA) using CCB

- Publication

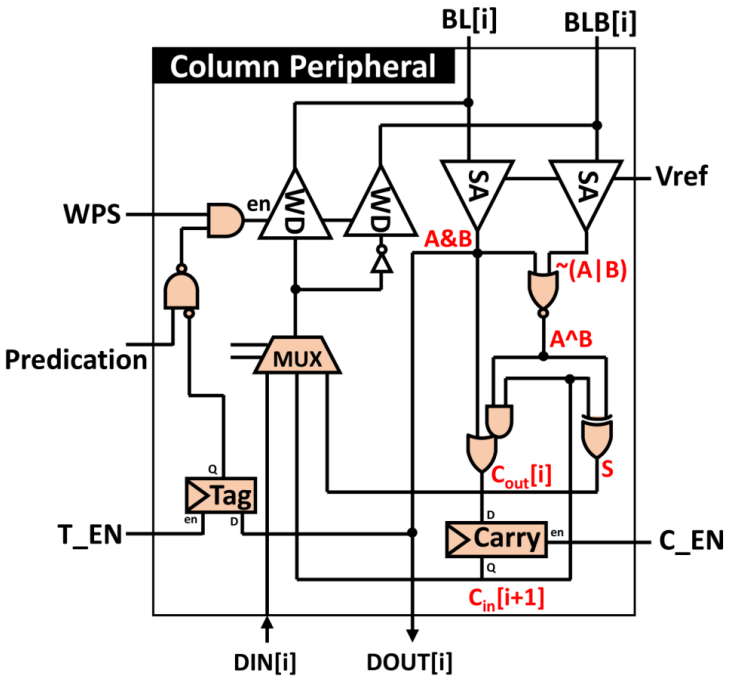
- X. Wang, V. Goyal, J. Yu, V. Bertacco, A. Boutros, E. Nurvitadhi, C. Augustine, R. R. Iyer, and R. Das, **"Compute-Capable Block RAMs for Efficient Deep Learning Acceleration on FPGAs,"** 2021 IEEE 29th Annual International Symposium on Field-Programmable CustomComputing Machines (FCCM), pp. 88–96, **2021**.

CCB Architecture

- Based on Neural-cache



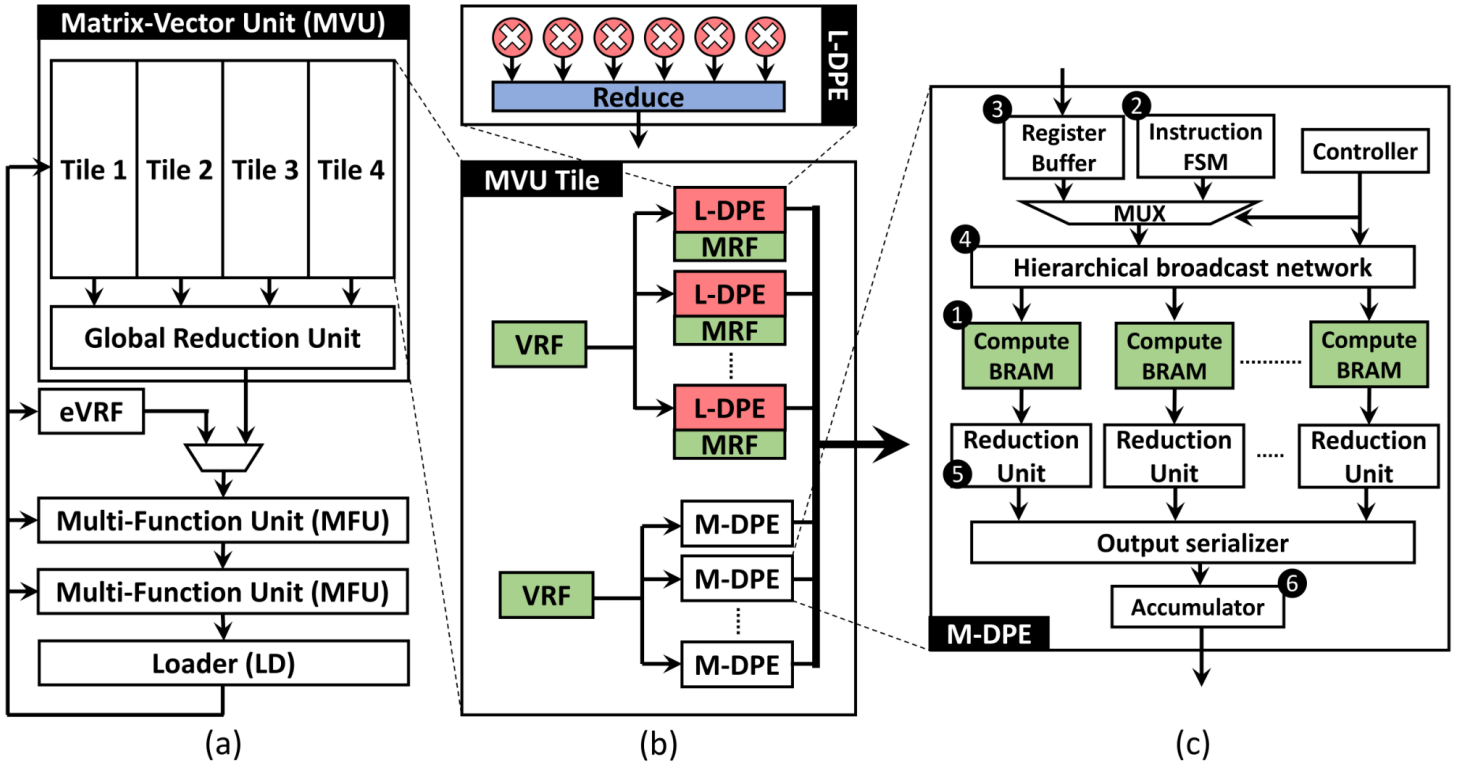
(a)



(b)

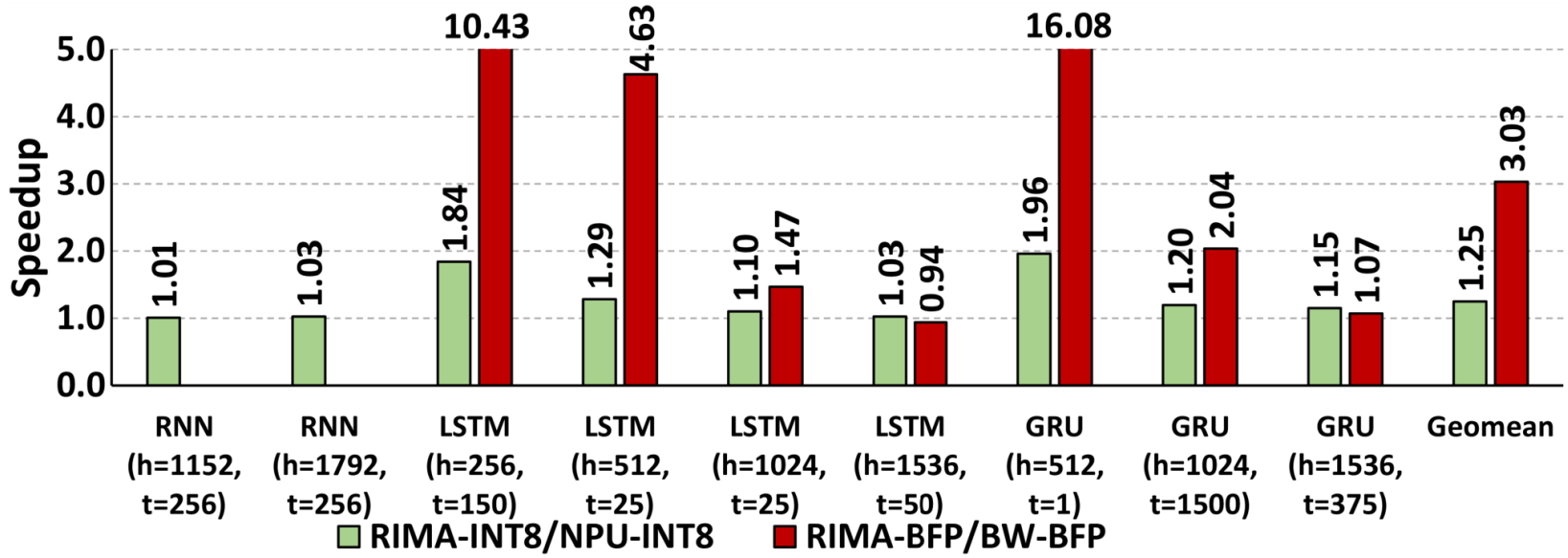
RIMA Architecture

- GEMV followed by non-linear function (activation)
- $Fn(A \cdot B) = Fn(a_1.b_1 + a_2.b_2 + \ldots + a_n.b_n)$



RIMA: Key improvements

- Compared with Intel's NPU and Microsoft's Brainwave architectures
  - Average speedup of 1.25x and 3x respectively
- Roughly 10x faster than Nvidia Titan V GV100 GPU



Case Study: PiCaSO/IMAGine

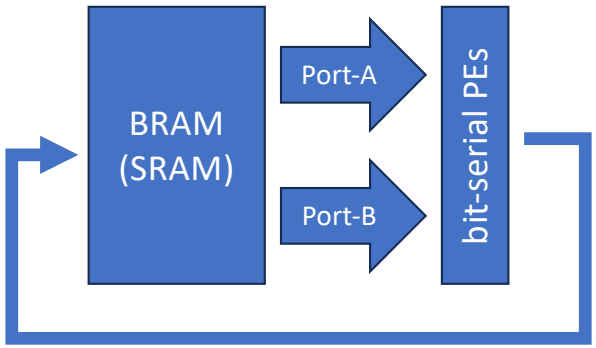
- Motivations:
  - Achieving the fastest clock: BRAM Fmax
  - Achieving linear scaling of peak-performance: 100% BRAM -> PIM

- Publications

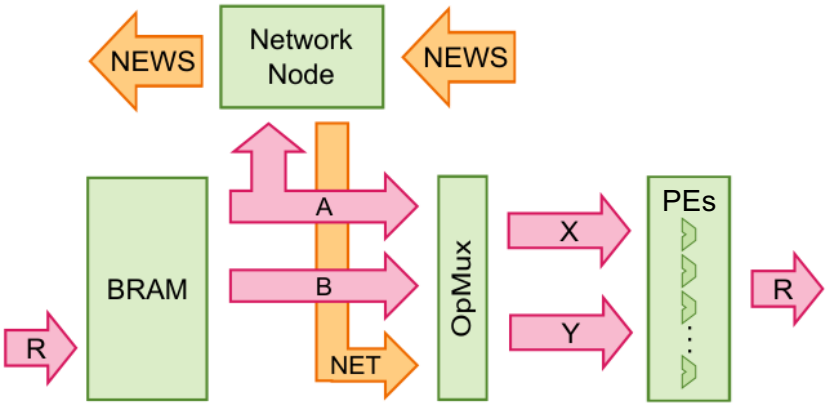
- M. A. Kabir, E. Kabir, J. Hollis, E. Levy-Mackay, A. Panahi, J. Bakos, M. Huang, and D. Andrews, **"FPGA Processor In Memory Architectures (PIMs): Overlay or Overhaul ?"** in 2023 33rd International Conference on Field-Programmable Logic and Applications (FPL). Gothenburg, Sweden: IEEE, Sep. **2023**, pp. 109–115.
- M. A. Kabir, T. Kamucheka, N. Fredricks, J. Mandebi, J. Bakos, M. Huang, and D. Andrews, **"IMAGine: An In-Memory Accelerated GEMV Engine Overlay"** in 2024 34th International Conference on Field-Programmable Logic and Applications (FPL). Turin, Italy: IEEE, Sep. **2024**.

PiCaSO Architecture

- **P**rocessor **i**n/near Memory S**ca**lable and Fa**s**t **O**verlay
- Existing PIMs: BRAM -> PE -> BRAM
- PiCaSO: Src (BRAM, Stream) -> OpMux -> PE -> BRAM



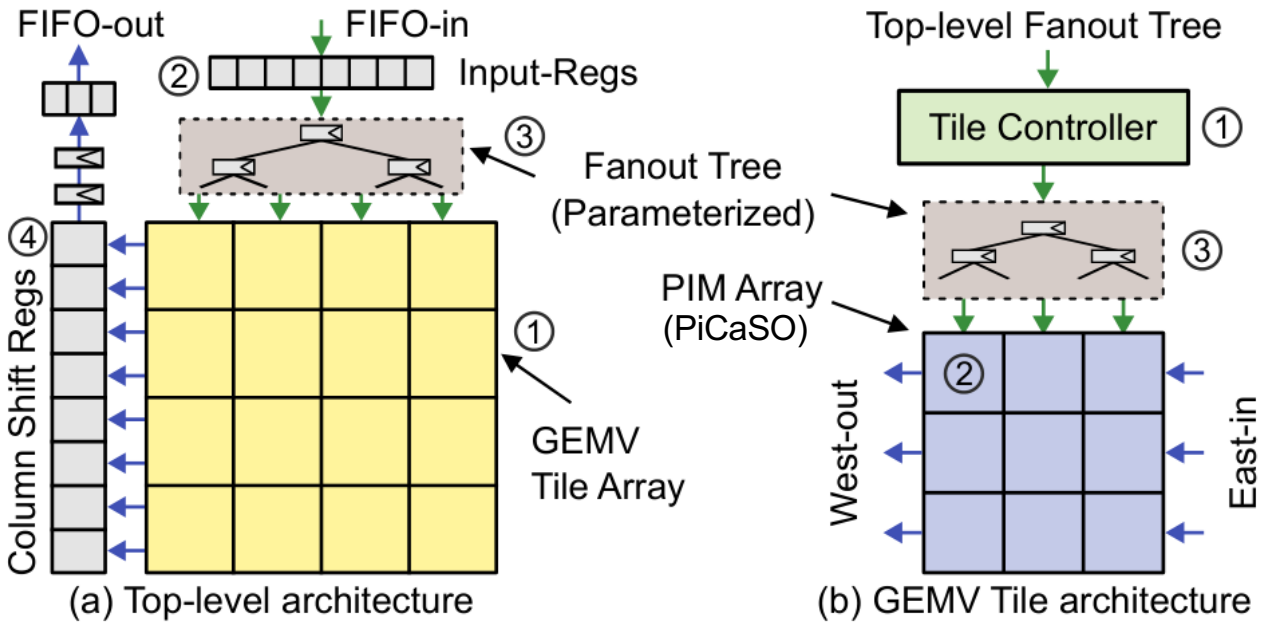
General architecture of existing PIM designs



PiCaSO Architecture

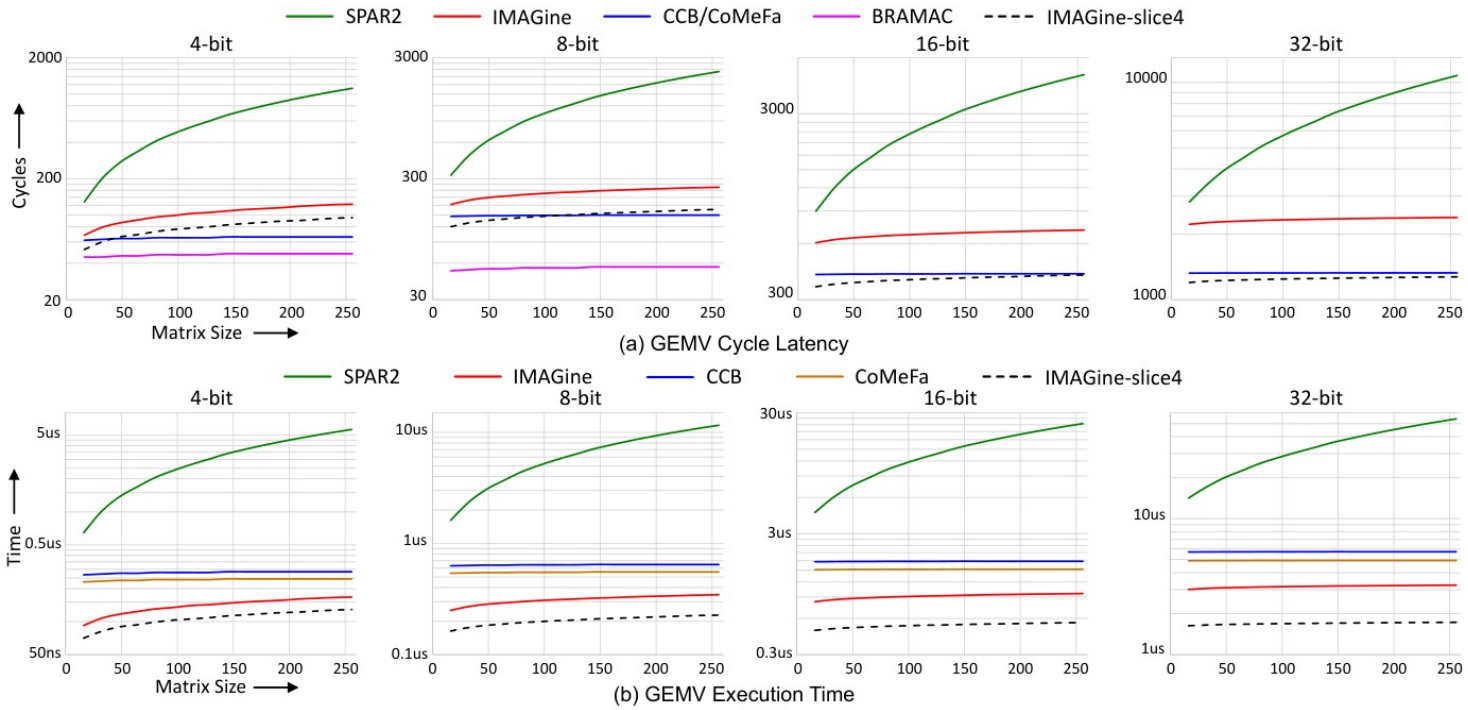
IMAGine Architecture

- **I**n-**M**emory **A**ccelerated **G**EMV eng**ine** overlay



IMAGine: Key Improvements

- Faster clock than TPU v1 & v2 with equal or more MACs on Alveo U55
  - Clock: 737 MHz (TPU 700MHz); MAC: 64K (TPU v1 64K, v2 16K)
- Faster GEMV compared to existing FPGA PIM accelerators



Key Takeaway

PIM architecture for FPGA is an active research topic with room for **significant** contributions!