



# 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, 1968

## Cellular Logic-in-Memory Arrays

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 information processing. Advantages can be obtained by circuitry in the form of a cellular array. A cellular array is a two-dimensional iterative configuration of identical elementary networks, or cells, which contains both logic and storage. The array is primarily to its immediate neighbors. Therefore, the array has the form of a memory enhanced with logic at each digit. Figures of these cellular logic-in-memory arrays are described. By way of an outline, balance of this paper then describes the cellular circuitry, operation, and use of a cellular sorting array, that has been shown to be versatile and efficient than several other cellular logic-in-memory arrays. The sorting array can be employed not only as a sorting memory which keeps in order the data, but also as a pushdown memory, a content-addressed 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.

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

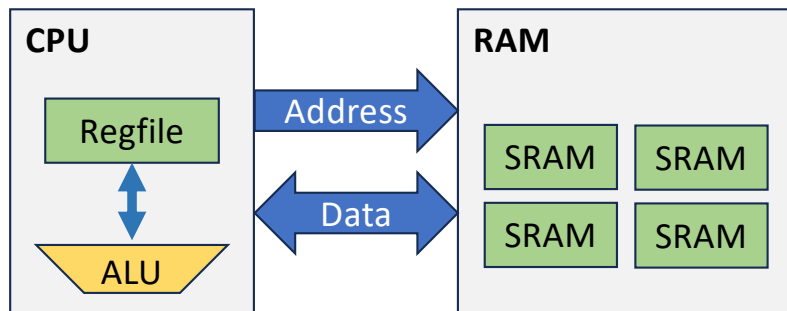
\*The work reported here was done while the author was at the Supercomputing Research Center, Bowie, Maryland.

**S**IMD 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

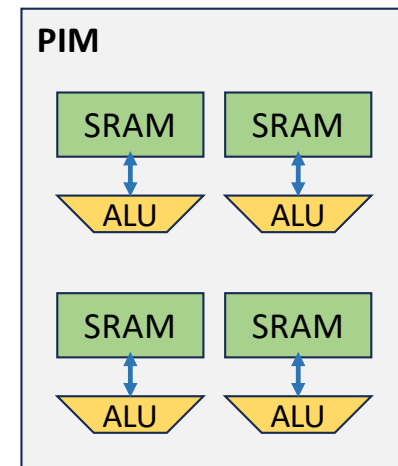
# What is Processor-in-Memory (PIM)



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



Classic von Neumann Architecture

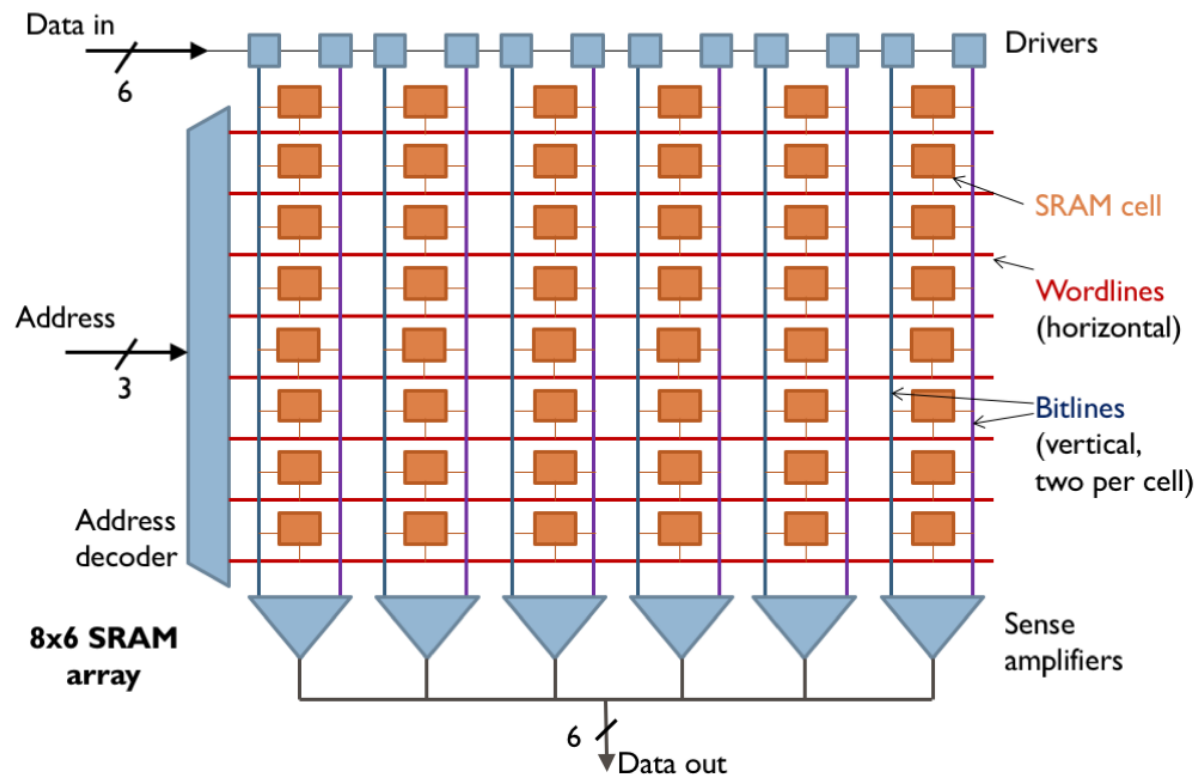


PIM Architecture

# 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.