Architectural Study, Benchmarking, and Comparative Analysis of In-Memory Computing (IMC) Simulators

Vraj Shah, Shubham Agrawal, Dewansh Singh Chandel, Sawale Sumeet Shivaji

Computer Architecture and Organisation - Term Project

November 12, 2024

Overview

- Introduction to In-Memory Computing (IMC) and its significance
- In-depth Architectural Study for IMC
- Overview of selected IMC simulators
- Architectural overview of Simulators and benchmarking process
- Analysis, insights and conclusion

Introduction

What is In-Memory Computing (IMC)?

- IMC is a computing paradigm that combines memory and processing to reduce data transfer bottlenecks.
- Aims to improve latency, energy efficiency, and throughput by minimizing data movement.
- Applications in AI, ML, and data analytics due to its data-intensive focus.

Role of Simulators in IMC Research

- Simulators provide a platform to test and analyze various IMC architectures before hardware implementation.
- Open-source simulators enable flexibility for exploring IMC concepts and evaluating performance under different configurations.
- This project benchmarks and compares simulators based on architecture, latency, energy consumption, and throughput.

Objectives

Planned Objectives:

1. Selection of appropriate IMC simulators
2. Definition of benchmarking programs
3. Establishment of performance metrics
4. Execution of performance benchmarking
5. Conduct comparative analysis with practical insights

Achieved Objectives:

- Deep understanding of IMC principles and benefits
- Selected suitable open-source simulators for benchmarking
- Conducted architectural study on IMC simulators
- Developed benchmarking programs for both general and IMC-specific workloads
- Defined performance metrics including execution time, memory usage, and scalability
- Drastic failure at benchmarking and comparative analysis

In-Memory Computing (IMC) Overview

- **IMC Paradigm**: IMC is a computing approach designed to address the "memory wall" problem in traditional Von Neumann architectures.
- **Core Idea**: Combines memory and processing, reducing the need for data transfer and enhancing speed and energy efficiency.
- **Applications**: Widely used in AI, ML, and big data analytics, where real-time processing and power efficiency are essential.
- **Advantages**:
  - Lower latency and higher throughput due to reduced data movement.
  - Significant energy savings, especially beneficial for data-intensive applications.
  - Scalability to support large datasets, making it ideal for high-performance and real-time applications.

Key Concepts, Applications, and Challenges in IMC

Key Concepts:

- Energy Efficiency
- Reduced Latency
- Scalability for Data-Intensive Tasks
- Parallelism

Applications:

- Artificial Intelligence (AI) and Machine Learning (ML)
- High-Performance Computing (HPC)
- Edge Computing

Challenges:

- Hardware Limitations
- Programming Complexity
- Thermal Management
- Limited Arithmetic Precision

Types of IMC Architectures

- Overview of In-Memory Computing (IMC) architectures.
- IMC reduces data transfer bottlenecks between memory and processors by performing computation directly within memory.
- Common IMC architectures include:
  - SRAM-based IMC
  - DRAM-based IMC
  - ReRAM-based IMC
  - Flash Memory-based IMC
  - Hybrid IMC

SRAM-based IMC Architecture Basics

- **Structure:** SRAM cells are typically designed with a 6-transistor (6T) configuration.
- **Key Features:**
  - Fast: SRAM cells provide faster access compared to other memory types.
  - Power-efficient: No need for periodic refreshing, unlike DRAM.
  - Stable: Data remains intact as long as power is supplied.
- **Bitwise Operations:** Transistors act as binary switches, enabling rapid operations such as AND, OR, and XOR.
- **Suitability:** SRAM's stable and fast access properties make it ideal for IMC applications requiring real-time computations.

Functioning of SRAM-based IMC

- **Modified Periphery Circuits:**
  - Adds computation-enabling circuits such as sense amplifiers, adders, and summing nodes.
- **Data Encoding:**
  - Binary encoding for bitwise operations.
  - Analog encoding for approximate computations.
- **Row/Column Activation:**
  - Activates specific rows/columns for operations like matrix-vector multiplication.
- **Advantages:**
  - Reduces energy consumption and latency compared to traditional computing.
  - Eliminates the need to move data between CPU and memory.

SRAM-based IMC: Operations

- **Bitwise Operations:**
  - Utilize the inherent binary nature of SRAM cells.
  - Sense amplifiers configured to detect voltage patterns for logical operations:
    - AND: Low voltage output indicates $A \wedge B$.
    - OR: High voltage output indicates $A \vee B$.
    - XOR: Exclusive high values detected as $A \oplus B$.
  - Rows/columns activated simultaneously for parallel processing.
- **Matrix-Vector Multiplication:**
  - Data stored in rows (matrix) and broadcast across columns (vector).
  - Parallel row activation computes dot products simultaneously.
  - Read currents summed at bitlines for analog accumulation.
- **Analog Operations:**
  - Encode data as varying voltages instead of binary states.
  - Multiple rows activated; their read currents summed to approximate operations like addition.
  - Noise-tolerant, suitable for neural networks and approximate computing.

DRAM-based IMC Architecture Basics

- **Memory Cells:**
  - Each cell consists of a capacitor (stores charge) and a transistor (access control).
- **Key Components:**
  - Wordlines: Activate rows of memory cells.
  - Bitlines: Read/write data from/to columns.
  - Sense Amplifiers: Detect charge levels for read operations.
- **Challenges:**
  - Requires periodic refreshing to retain data.
  - Slower access times compared to SRAM.

Functioning of DRAM-based IMC

- **Modified Periphery Circuits:** Enhancements include:
  - Sense amplifiers for logical operations.
  - Charge-sharing mechanisms for computation.
- **Row Activation:**
  - Single-row activation for simple read/write operations.
  - Multi-row activation for bitwise computations (AND, OR, XOR).
- **Charge Accumulation:**
  - Summing multiple rows' charges enables analog computations like matrix multiplications.
- **Advantages:**
  - High data density for large-scale tasks.
  - Energy-efficient for memory-bound applications.

DRAM-based IMC: Operations

- **Bitwise Operations:**
  - Dual/multi-row activation enables logic operations.
  - Sense amplifiers detect cumulative charge levels:
    - AND: High charge indicates $A \wedge B$.
    - OR: Detects charge from any active row for $A \vee B$.
    - XOR: Charge levels differentiate exclusive high states $A \oplus B$.
- **Matrix-Vector Multiplication:**
  - Rows mapped to matrix, vector values applied as voltages.
  - Charge-sharing across bitlines enables summation of partial products.
  - Sense amplifiers detect accumulated charge for final dot product results.
- **Analog Operations:**
  - Data represented as varying charge levels on capacitors.
  - Multiple rows activated; combined charge approximates operations.
  - Effective for applications like AI models tolerating approximation and noise.

ReRAM-based IMC Architecture Basics

- **Structure:**
  - Crossbar array with ReRAM cells at wordline/bitline intersections.
- **Key Features:**
  - Multi-level cells (MLCs) store multiple bits per cell.
  - Low power consumption for storage and computation.
- **Operation:**
  - Binary encoding (high/low resistance states) for precision.
  - Analog encoding (varied resistance levels) for approximate computing.

Functioning of ReRAM-based IMC

- **Voltage Manipulation:** Controls bitwise operations like AND, OR, XOR.
- **Row Activation:** Activates multiple rows to perform current summation.
- **Analog Operations:**
  - Resistance levels represent weights.
  - Summed currents enable matrix-vector multiplications.
- **Applications:**
  - AI workloads like neural networks.
  - Approximate computations in machine learning.

ReRAM-based IMC: Operations

- **Bitwise Operations:**
  - Voltage manipulation across ReRAM cells enables logic:
    - AND: Low-resistance states produce high currents for $A \wedge B$.
    - OR: Any low-resistance cell results in $A \vee B$.
    - XOR: Exclusive patterns detected by sense amplifiers.
  - Parallel processing across rows accelerates computations.
- **Matrix-Vector Multiplication:**
  - Rows represent matrix elements, input vector encoded as applied voltages.
  - Ohm's and Kirchhoff's laws govern current summation:

$$\text{Output Current} = \sum (V \cdot G)$$

  - Sense amplifiers process summed currents for dot product results.
- **Analog Operations:**
  - Data encoded as resistance levels, representing weights.
  - Current accumulation on bitlines approximates summation or addition.
  - Suitable for approximate computing in machine learning and AI.

Flash Memory-based IMC Architecture Basics

- **Structure:**
  - Floating-gate transistors for charge storage.
- **Key Types:**
  - NAND flash: High density, slower read speeds.
  - NOR flash: Faster reads, suitable for random access.
- **Advantages:**
  - Supports both binary and analog operations.
  - High data storage density.

Functioning of Flash Memory-based IMC

- **Threshold Detection:**
  - Logic states represented by different threshold voltages.
- **Row/Column Activation:**
  - Enables parallel processing across rows and columns.
- **Sense Amplifiers:**
  - Detect combined voltage levels for analog computations.
- **Applications:**
  - Matrix-vector multiplication.
  - Cryptography and data analytics.

Flash Memory-based IMC: Operations

- **Bitwise Operations:**
  - Logic states encoded as threshold voltages.
  - Sense amplifiers interpret cumulative charge states:
    - AND: Combined threshold from high-charge cells.
    - OR: Any charged cell activates output.
    - XOR: Exclusive high threshold levels detected.
  - Parallel processing enabled by row/column activation.
- **Matrix-Vector Multiplication:**
  - Matrix mapped to rows, vector applied as input voltages.
  - Threshold levels combined along bitlines; sense amplifiers detect results.
  - Effective for neural network layers and large-scale computations.
- **Analog Operations:**
  - Multi-level cells (MLCs) encode data as varied threshold voltages.
  - Cumulative charge across bitlines approximates analog summation.
  - Ideal for approximate tasks like AI inference or image processing.

Hybrid IMC

- Combines SRAM, DRAM, and ReRAM to maximize strengths of each type.
- Benefits:
  - High performance (SRAM).
  - High density (DRAM).
  - Non-volatility (ReRAM).
- Applications:
  - Tasks requiring a balance of speed, density, and persistence.

Functioning of Hybrid IMC

- **Adaptive Operations:** Utilizes the strengths of each memory type.
- **Matrix Operations:** Distributes computational tasks based on memory strengths.
- **Analog Summation:** Leverages DRAM and ReRAM for energy-efficient summation techniques.

Hybrid IMC: Operations

- **Bitwise Operations:**
  - SRAM: High-speed logical operations (AND, OR, XOR) via rapid row activation.
  - DRAM: Slower operations on larger datasets.
  - ReRAM: Adjustable resistance states enable approximate bitwise logic.
  - Parallel processing across memory types for diverse tasks.
- **Matrix-Vector Multiplication:**
  - SRAM: High-speed matrix-vector operations for real-time tasks.
  - DRAM: Handles larger matrices with moderate speed.
  - ReRAM: Multi-bit cells approximate results for large datasets.
  - Tasks distributed across memory types based on performance requirements.
- **Analog Operations:**
  - DRAM and ReRAM support analog summation techniques.
  - SRAM handles precise operations requiring high speed.
  - Hybrid designs optimize performance and energy efficiency for mixed workloads.

Simulators for In-Memory Computing (IMC)

- Thorough investigation into open-source IMC simulators.
- Goals:
  - Identify simulators widely recognized in research.
  - Assess ability to simulate IMC architectures effectively.
- Simulators Studied:
  1. PUMA
  2. PIM-SIM
  3. Cross-SIM
  4. MN-SIM
  5. Multi-SIM
  6. CIM-SIM

PUMA: Processing Using Memory Architecture

- Designed for IMC-based machine learning workloads.
- **Core Features:**
  - Integrates SRAM for both storage and computation.
  - Reduces latency and power consumption.
  - Analog computations (e.g., matrix-vector multiplications) performed within memory arrays.
- **Processing Elements (PEs):**
  - Arrays of PEs with local accumulators and shift registers.
  - Handle intermediate results locally, minimizing memory traffic.

PUMA: Advanced Features

- **Analog-Digital Partitioning:**
  - Analog computations for efficiency.
  - ADCs/DACs enable smooth interaction with digital operations.
  - Multi-level cells (MLCs) support high-density analog storage and computation.
- **Data Flow Optimization:**
  - Hierarchical interconnects (local/global buses) reduce congestion.
  - Parallelism and pipelining optimize large-scale neural network computations.
- **Neural Network Acceleration:**
  - Efficient execution of MAC operations for convolutional and fully connected layers.
  - Can handle multiple layers simultaneously with reduced overhead.

PIM-SIM: Processing-In-Memory Simulator

- Focuses on integrating computation directly into memory for energy-efficient processing.
- **Memory-Processing Integration:**
  - Uses PIM-enabled DRAM modules for in-memory computation.
  - Processing units like ALUs are embedded within memory banks.
- **Parallelism and Local Computation:**
  - Distributed processing across memory banks.
  - Tasks like vector addition and matrix multiplication are offloaded to memory.
- **Benefits:**
  - High data-level parallelism.
  - Reduced data movement between CPU and memory.
  - Energy-efficient for memory-bound tasks.

PIM-SIM: Key Features

- **Efficient Interconnects:**
  - High-speed local interconnects for intra-bank operations.
  - Cross-bank communication via high-bandwidth networks for matrix computations.
- **Task Specialization:**
  - Ideal for repetitive tasks like filtering and matrix operations.
  - Offloading reduces latency and enhances throughput for machine learning workloads.
- **Scalability:**
  - Parallel operations across memory banks enable scalability for large datasets.

Cross-SIM: Flexible IMC Simulation Tool

- High-performance simulation for complex architectures.
- **Core Capabilities:**
  - Modular components for memory, processors, I/O, and network.
  - System-level modeling of heterogeneous architectures.
- **IMC Simulation:**
  - Models processing-in-memory (PIM) systems.
  - Task-level and memory-parallelism reduce data movement bottlenecks.
- **Key Applications:**
  - Benchmarking emerging IMC technologies.
  - Evaluating performance, energy efficiency, and scalability.

Cross-SIM: Advanced Features

- **Support for Emerging IMC Technologies:**
  - 3D-stacked memory and near-memory processing simulations.
  - High-bandwidth, low-latency architectures.
- **Energy Profiling:**
  - Detailed energy consumption analysis.
  - Identifies power-efficient configurations for IMC.
- **Fault Tolerance:**
  - Simulates error correction mechanisms (e.g., ECC).
  - Evaluates system reliability under fault conditions.

MN-SIM: Memory Network Simulator

- Specialized for memory-centric computing architectures.
- **Core Features:**
  - Integrates PEs within memory for local computations.
  - High-bandwidth interconnects enable efficient cross-bank communication.
- **Local and Global Data Processing:**
  - Local operations like vector additions occur within memory banks.
  - Global operations supported by distributed memory systems.
- **Applications:**
  - Machine learning, data analytics, and large-scale memory-bound tasks.

MN-SIM: Advanced Features

- **Parallelism:**
  - Concurrent operations across memory banks.
  - Optimized data locality reduces access delays.
- **Energy Efficiency:**
  - In-memory computations reduce power consumption.
  - Avoids frequent data transfers to external processors.
- **Hardware-Software Co-Design:**
  - Models interaction between hardware and software optimizations.
  - Enables task scheduling and parallelization strategies.

Multi-SIM: Electronics Simulation Tool

- General-purpose tool for analog and digital circuit simulation.
- **Component-Based Design:**
  - Simulates custom memory units with embedded processing elements.
  - Models PIM architectures for in-memory computations.
- **Parallelism:**
  - Simulates parallel memory accesses for low-latency operations.
- **Applications:**
  - Useful for early-stage design and testing of hybrid IMC systems.

Multi-SIM: Advanced Features

- **Energy Modeling:**
  - Simulates power draw and current usage for memory-integrated systems.
- **Fault Tolerance:**
  - Models error-checking components like voltage fluctuations.
  - Useful for evaluating system reliability.

CIM-SIM: Compute-In-Memory Simulator

- Specialized for simulating memory-integrated computation.
- **Core Capabilities:**
  - Modular simulation of memory, processing, and networks.
  - Supports large-scale heterogeneous IMC systems.
- **IMC Focus:**
  - Task-parallelism and memory-parallelism.
  - Simulates advanced technologies like 3D-stacked memory.

CIM-SIM: Advanced Features

- **Energy Efficiency:**
  - Profiles power consumption for IMC configurations.
- **Fault Tolerance:**
  - Models error detection and correction mechanisms.
- **High-Performance Simulations:**
  - Distributed simulations for data-intensive workloads.

Benchmarking Programs Listed

- Matrix Multiplication (MatMul)
- Neural Network Inference
- Convolutional Neural Networks (CNNs)
- Long Short-Term Memory (LSTM) Networks
- Sparse Matrix-Vector Multiplication (SpMV)
- DNN Training (Backpropagation)

Performance Metrics Listed

- Execution Time
- Memory Usage
- Latency
- Throughput
- Energy Efficiency
- Scalability

Benchmarking Process

- Encountered significant compatibility and setup issues, limiting execution of planned experiments.
- **Successful Runs:**
  - MNSIM: AlexNet, LeNet, ResNet18.
  - PUMA: LSTM, MLP.
  - PIMSIM: ResNet18.
- **Unsuccessful Runs:**
  - Compiler issues and dependency conflicts.
  - Outdated tools and simulator-specific requirements.
- Highlights the urgent need for standardized benchmarking tools and frameworks.

Results: Comprehensive Metrics Table

Framework	Program	Latency (ms)	Energy (nJ/J)	Power (W)	Simulation Time (s)
MNSIM	AlexNet	0.8248	313829.11 nJ	108.1671	0.3910
	LeNet	0.35549	7906.19 nJ	3.2541	0.1142
	ResNet18	3.08743	935655.54 nJ	498.5991	1.5233
	VGG8	5.3307	3827427.15 nJ	338.6245	0.9770
	VGG16	3.59392	1923761.69 nJ	534.9752	1.6312
PUMA	LSTM	NA	0.0041 J	42637.46	9.5583e-5
	MLP	NA	0.000124 J	4543.86	2.7274e-5
PIMSIM	ResNet18	25	1647884735 pJ/it	0.0659 mW	0.1

Results: Analysis

- Compatibility issues prevented running standardized programs across all simulators.
- **Key Insights:**
  - Latency: Significant variations observed, with MNSIM performing better for AlexNet and LeNet.
  - Energy Efficiency: PIMSIM showcased high energy efficiency for ResNet18 tasks.
  - Power Consumption: MNSIM demonstrated higher power requirements for complex models like VGG16.
- Highlights the lack of interoperability among IMC simulators and the need for standardized testing frameworks.

Results: Architectural Comparison

Simulator	**Memory Type**	**Key Features**	**Applications**
PUMA	SRAM	Analog-digital integration, MLP	Neural networks
PIMSIM	DRAM	Processing-in-memory, ONNX support	Data analytics, AI/ML
MNSIM	SRAM/Hybrid	Bitwise ops, matrix multiplication	General-purpose workloads
CIM-SIM	Flash/ReRAM	Non-volatile memory, analog ops	Low-power AI/ML
Multi-SIM	SRAM	Customizable processing elements	Analog/digital circuit design
Cross-SIM	Hybrid	Modular components, scalability	HPC, heterogeneous systems

Challenges Faced

- **Compatibility Issues**:
  - PUMA's reliance on Python 2.7 required a virtual environment, adding complexity.
  - PIMSIM model conversions had limitations, only successfully compiling a subset of models.
- **Simulator Selection Constraints**:
  - Limited availability of open-source simulators with desired functionality and support.
- **Program Support and Consistency**:
  - Inconsistencies across simulators made it challenging to run a uniform set of programs for benchmarking.

Open Issues

- **CrossSim Limitations**:
  - Lacks analog accelerator attributes like energy and area, and requires CUDA, limiting compatibility with AMD systems.
- **Documentation Gaps**:
  - CIMulator's limited documentation impedes effective implementation.
  - MultiSim lacks a repository, affecting accessibility and usability.
- **Standardization Challenges**:
  - Absence of standardized benchmarking tools limits comparability across simulators.

Key Takeaways

- **Significance of Open-Source IMC Simulators**:
  - Critical tools for exploring new computing architectures, but hindered by inconsistent support and limited documentation.
- **Benchmarking Challenges**:
  - Running a common set of programs across simulators is essential for reliable comparison, but currently infeasible.
- **Recommendations for Future Research**:
  - Future work should enhance cross-platform compatibility, improve documentation, and standardize benchmarking protocols for meaningful evaluations.

The END

Thank You!