IDSL
Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

# Computer Architecture and Memory System Design for Deep Neural Networks

Presenter Hyun Kim | Associate Professor

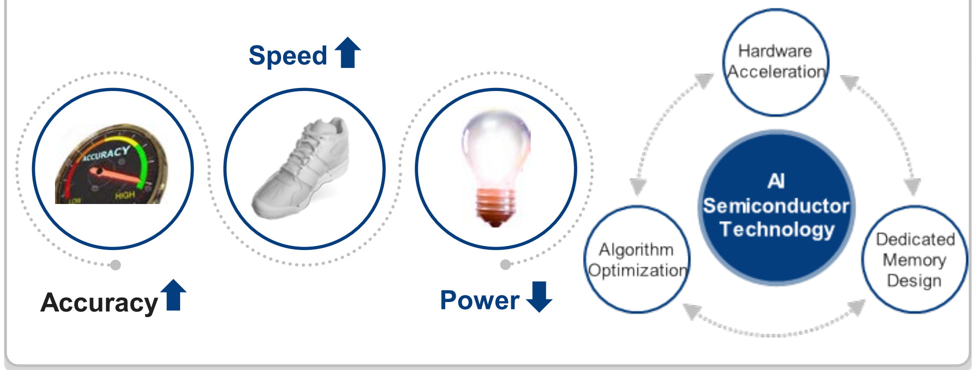
Affiliation Seoul National University of Science and Technology
Electrical and Information Engineering

Contact hyunkim@seoultech.ac.kr / idsl.seoultech.ac.kr

# Key Issue of On-device AI Accelerators: Memory

## Three main goals of AI accelerators

**High accuracy + High speed (Throughput) + Low power (Energy-Efficiency)**



## Necessity of memory-level approach

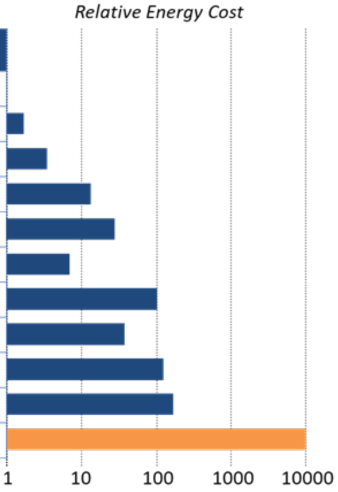
Recent AI models demand a significant amount of data and memory systems are becoming increasingly critical→ **Low-power and high-speed memory platforms** dedicated to AI models lead to power-saving and speed-up of AI accelerators

**Key Point**

**Minimization of memory capacity and access overhead** by reflecting the model structure

## Overhead for memory access compared to operations in processing units

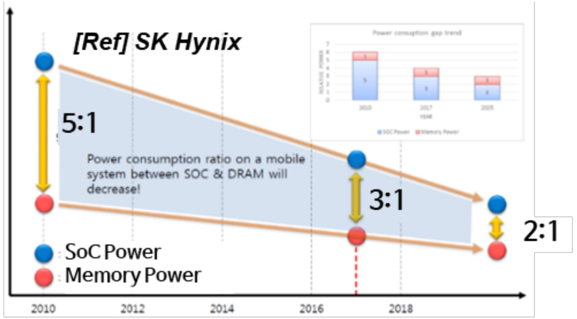
| Operation | Energy (pJ) |
|---|---|
| 8b Add | 0.03 |
| 16b Add | 0.05 |
| 32b Add | 0.1 |
| 16b FP Add | 0.4 |
| 32b FP Add | 0.9 |
| 8b Mult | 0.2 |
| 32b Mult | 3.1 |
| 16b FP Mult | 1.1 |
| 32b FP Mult | 3.7 |
| 32b SRAM Read (8KB) | 5 |
| 32b DRAM Read | 640 |

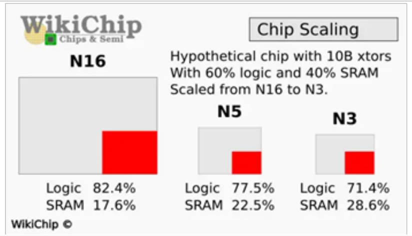


Source : Computing's Energy Problem (and what we can do about it) (ISSCC'14)

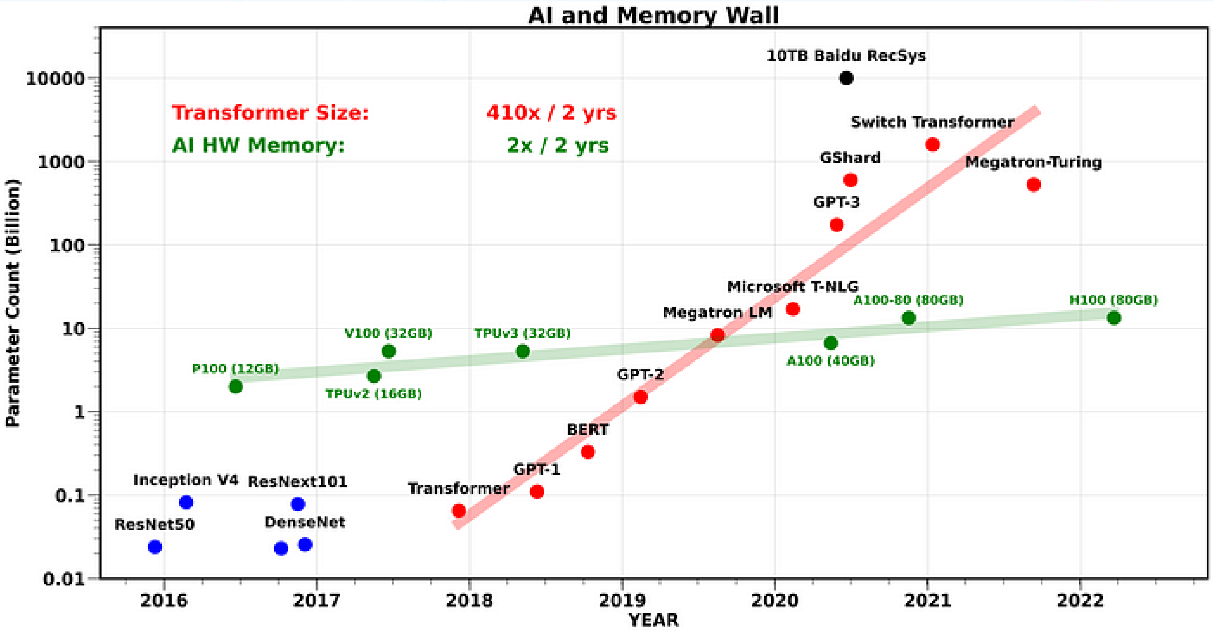
Energy per operation [pJ] 45nm vs 7nm

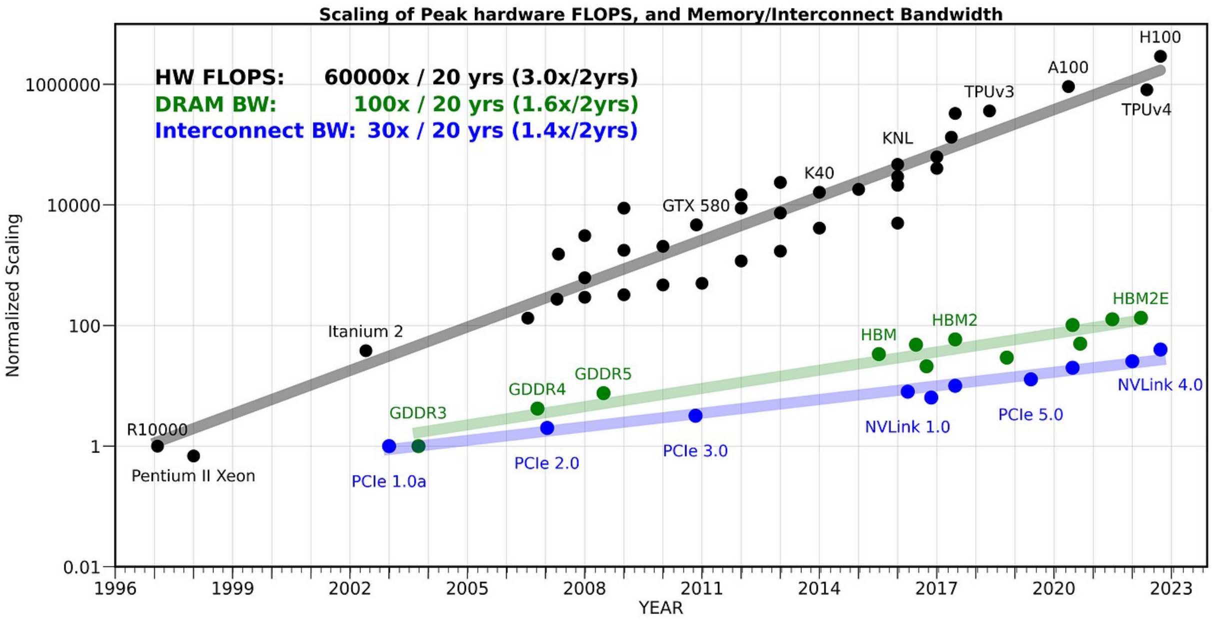
| | Operation | Picojoules per Operation | | |
|---|---|---|---|---|
| | | 45 nm | 7 nm | 45 / 7 |
| + | Int 8 | 0.03 | 0.007 | 4.3 |
| | Int 32 | 0.1 | 0.03 | 3.3 |
| | BFloat 16 | -- | 0.11 | -- |
| | IEEE FP 16 | 0.4 | 0.16 | 2.5 |
| | IEEE FP 32 | 0.9 | 0.38 | 2.4 |
| × | Int 8 | 0.2 | 0.07 | 2.9 |
| | Int 32 | 3.1 | 1.48 | 2.1 |
| | BFloat 16 | -- | 0.21 | -- |
| | IEEE FP 16 | 1.1 | 0.34 | 3.2 |
| | IEEE FP 32 | 3.7 | 1.31 | 2.8 |
| SRAM | 8 KB SRAM | 10 | 7.5 | 1.3 |
| | 32 KB SRAM | 20 | 8.5 | 2.4 |
| DRAM | DDR3/4 | $1300^2$ | $1300^2$ | 1.0 |
| | HBM2 | -- | $250\text{-}450^2$ | -- |
| | GDDR6 | -- | $350\text{-}480^2$ | -- |





# Scaling of Model Size / PU / Memory / Interface



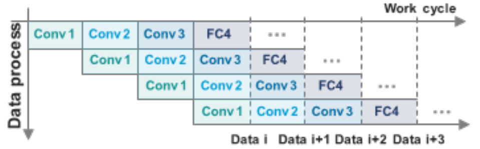


# Overview of On-device AI Accelerators

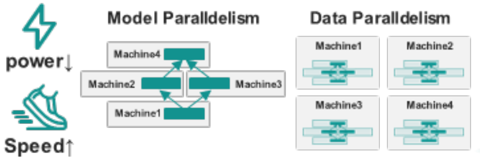
**Mobile Characteristic**

- Both Inference & Training
- Low-Power FPGA/ASIC for Mobile
- Low Precision: 2b/4b/8b (INT)
- Sparse network
- Application-specific accelerator design

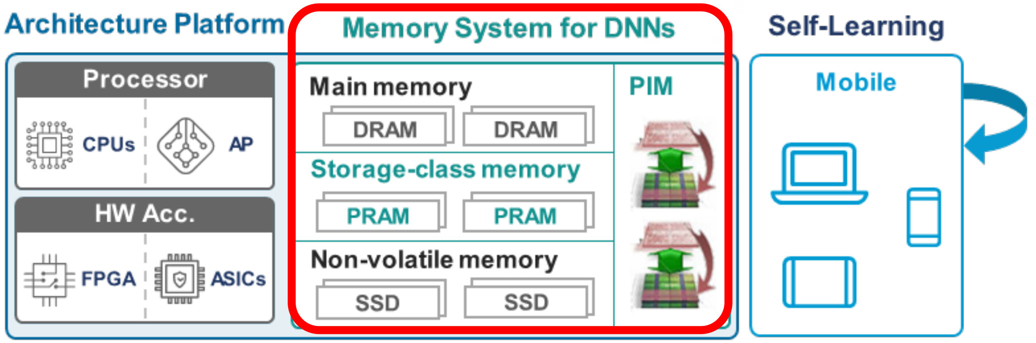
## HW-based low complexity schemes for low-power & speed-up



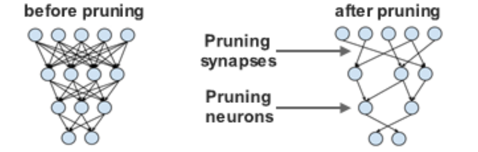
Pipelining



Parallelism



## SW-based low complexity schemes for low-power & speed-up

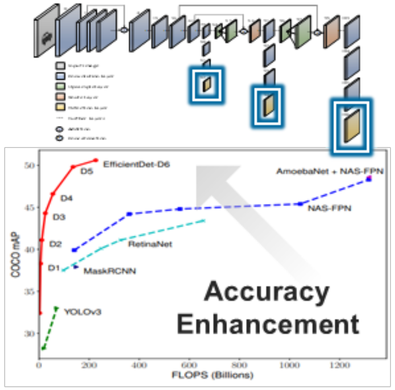


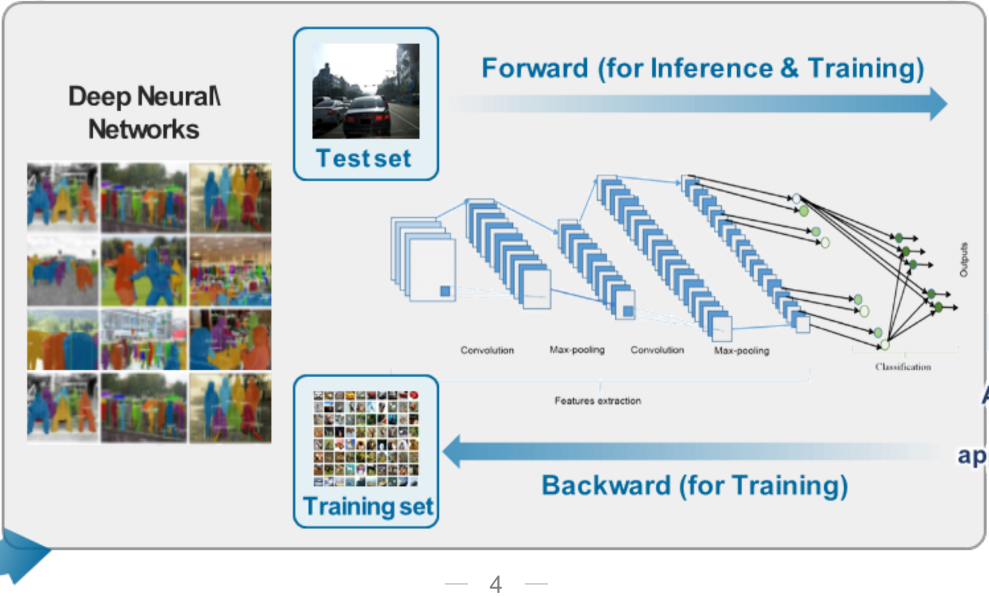
Pruning

$$\begin{array}{|c|c|c|c|}\hline 2.5 & -0.9 & 0.1 & 0.2 \\ \hline 0.2 & -0.1 & 2 & -0.2 \\ \hline 0.5 & -1.9 & -0.2 & -0.1 \\ \hline -0.3 & 0.1 & -1.5 & 1.2 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|}\hline 2^{1} & -2^{0} & 2^{-1} & 2^{-1} \\ \hline 2^{-1} & -2^{-1} & 2^{1} & -2^{-3} \\ \hline 2^{-1} & -2^{1} & -2^{-2} & -2^{-3} \\ \hline -2^{-1} & -2^{-2} & -2^{0} & 2^{0} \\ \hline \end{array}$$

Quantization

## Performance enhancement schemes





Apply to target applications

Autonomous Driving

Robot (Physical AI)

# Low-Power CNN Inference Using Prefetcher-Assisted NVM Systems

## Goal

**Improve the energy efficiency and speed up using prefetch-aware memory controller within the NVM system for low-power on-device AI**

## Motivation 1

- Computation-intensive CNNs have long memory idle time → Huge portion of unnecessary standby energy in main memory

## Motivation 2

- $P_{idle}$ of PRAM is 100 x  lower than DRAM and PRAM has a higher density than DRAM

## Solution/ Contribution

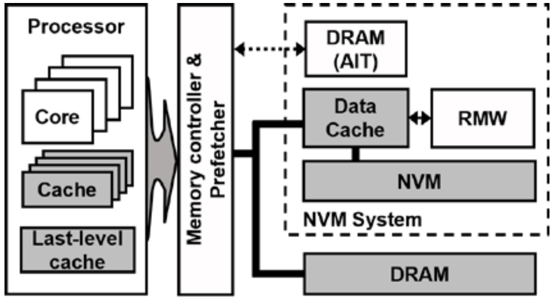
### 1 Adaptive prefetcher for CNN inference

Record and predict using block addresses in main memory through hot/cold data frequency and temporal and spatial locality

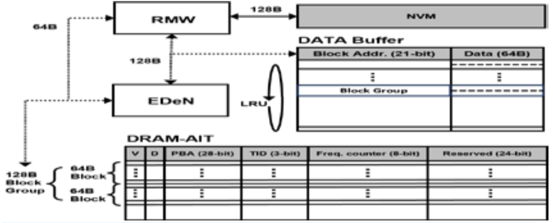
### 2 Prefetcher-aware memory controller within the NVM system:

36-bit of DRAM-AIT is reserved to be used as counter for prefetcher and data management & Proposed architecture uses 128B data as a block group to eliminate additional data latency in RMW

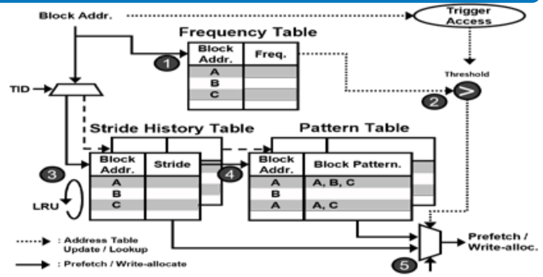
## Overview of the EDeN Platform



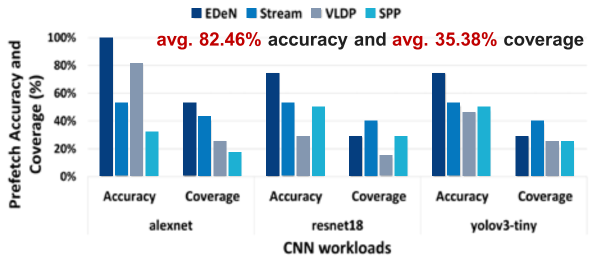
## Internal Architecture of EDeN Prefetcher



## Control Flow of EDeN prefetcher

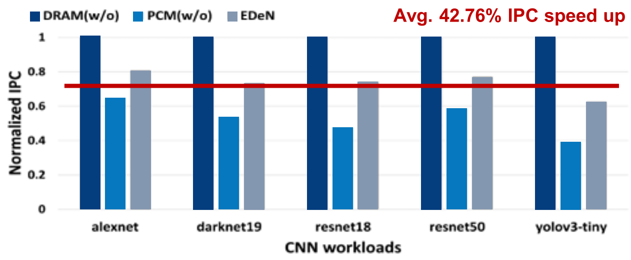


## Prefetcher accuracy and coverage comparison



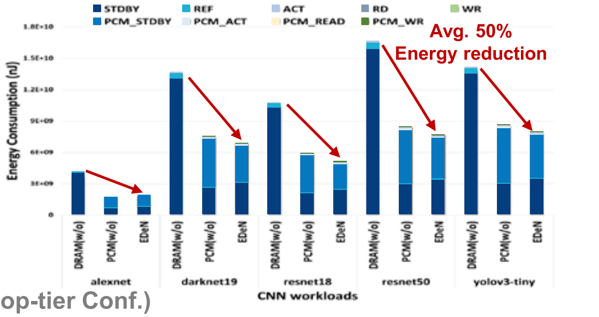
avg. 82.46% accuracy and avg. 35.38% coverage

## Normalized IPC comparison



Avg. 42.76% IPC speed up

## Energy consumption comparison



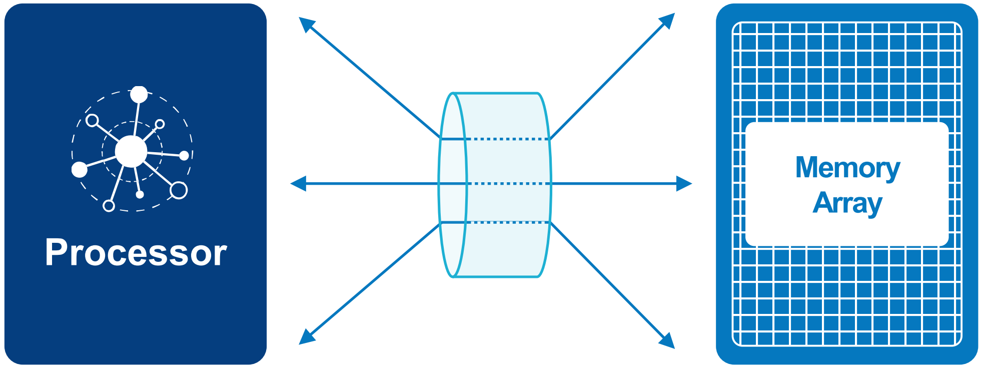
Avg. 50% Energy reduction

"EDeN: Enabling Low-Power CNN Inference on Edge Devices Using Prefetcher-Assisted NVM Systems," ISLPED 2024 (BK21 Top-tier Conf.)

# Processing-in-Memory (PIM)

## Motivation of PIM

### Memory Bottleneck or Memory “Wall”



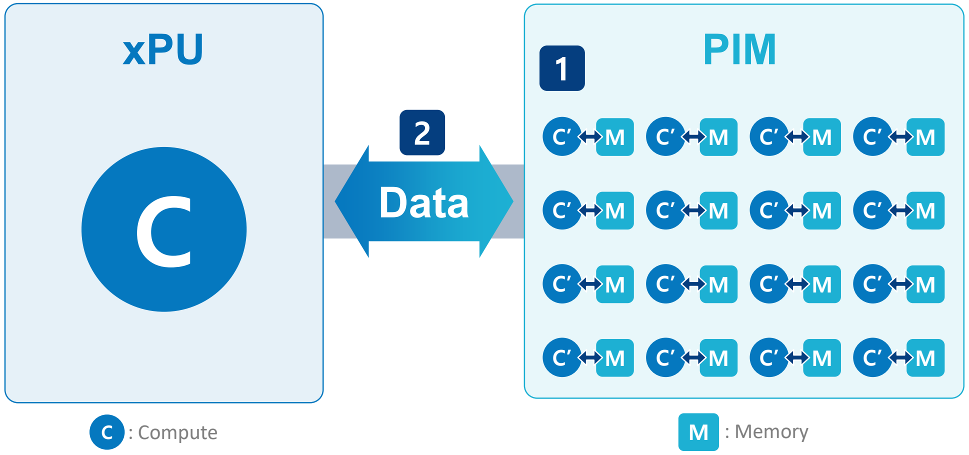
Von Neumann architecture

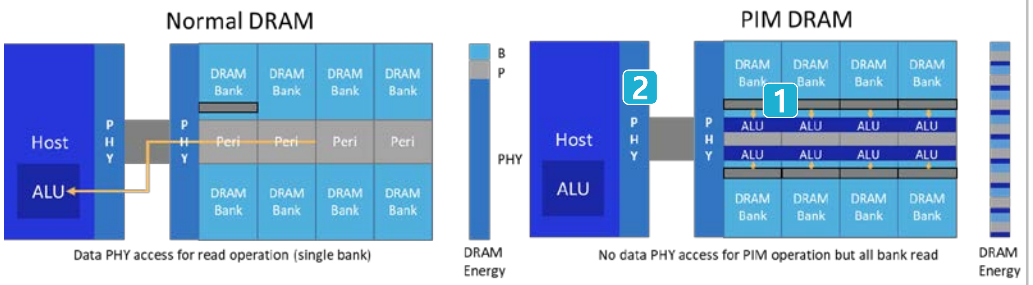
**Memory bottleneck** (or Memory wall) due to serial communication method between processor and memory area

→ **PIM** is a technology to overcome the memory bottleneck (wall)

→ **Energy-efficiency** and **high-speed** are essential!

## PIM Features

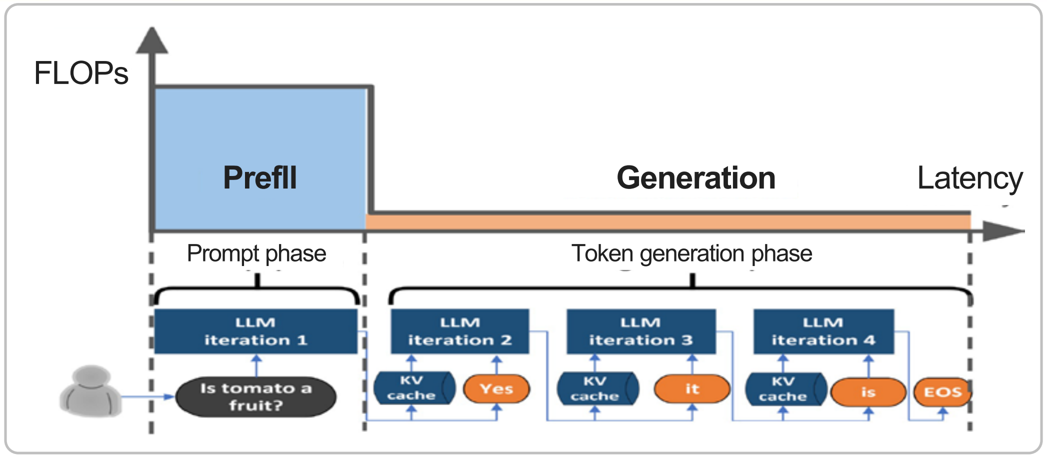


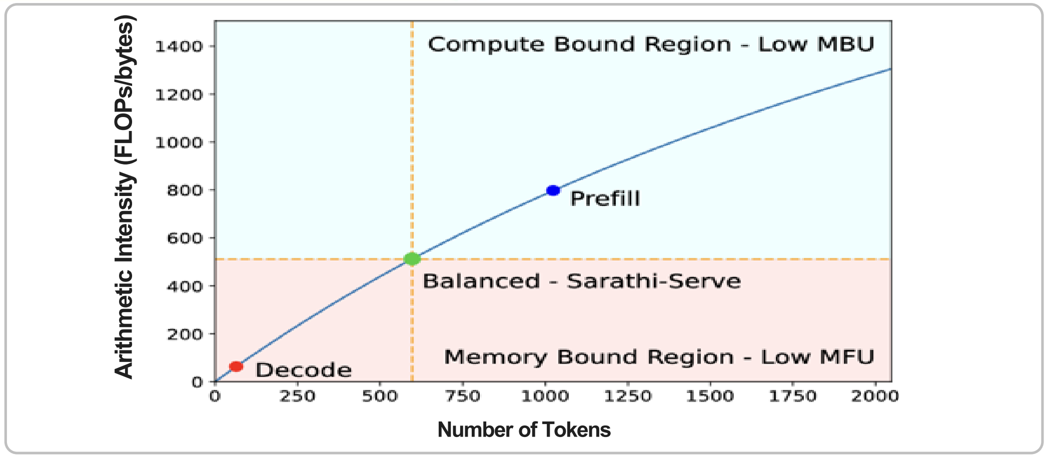

Normal DRAM
Host
ALU
PHY
DRAM Bank
Peri
Data PHY access for read operation (single bank)
B
P
PHY
DRAM Energy
PIM DRAM
2
1
ALU
No data PHY access for PIM operation but all bank read
DRAM Energy

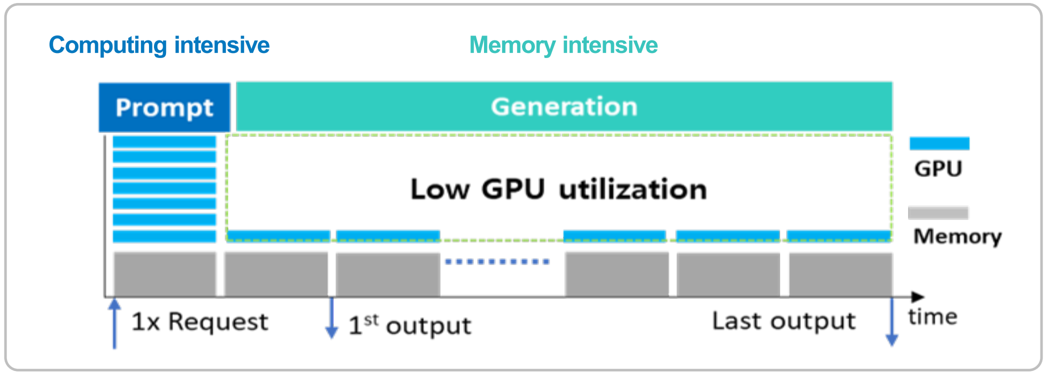
1 **Performance Improvement**
By utilizing the higher Bandwidth inside the memory

2 **Energy Efficiency Improvement**
By minimizing data movement between host and memory

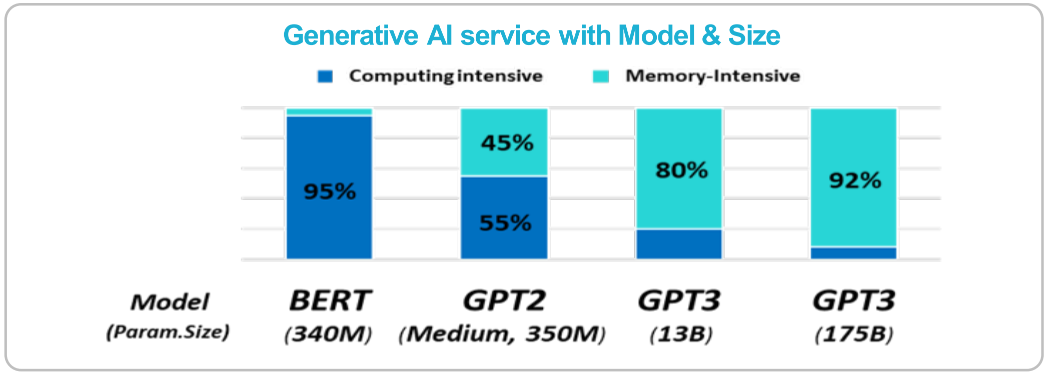
# LLM: Prefill + Generation (Decode)







Generative AI service with Model & Size



## LLM operation | Compute-bound+Memory-bound

01 LLM Inference consists of input processing (Prompt, Prefill) and answer generating (Generation, Decode)

02 Prefill: Compute-bound / Generation: Memory-bound

03 The larger the model, the more memory intensive function (GEMV), so memory bandwidth for GEMV has a greater impact on system performance than the processor

04 Good motivation for Hybrid NPU-PIM Acceleration Platform

# A Versatile and Cycle-Accurate Simulator for DRAM PIM Systems

## Goal

**A cycle-accurate and scalable PIM-CPU simulator that can faithfully model CPU-only, PIM-only, and hybrid DRAM-based PIM-CPU configurations, thereby enabling system-level exploration of emerging memory-centric architectures**

## Motivation

The lack of versatile and cycle-accurate simulation frameworks significantly limits the evaluation and optimization of diverse PIM designs → Existing in-house tools are narrowly tailored to specific architectures, lacking generality and extensibility

## Solution/ Contribution

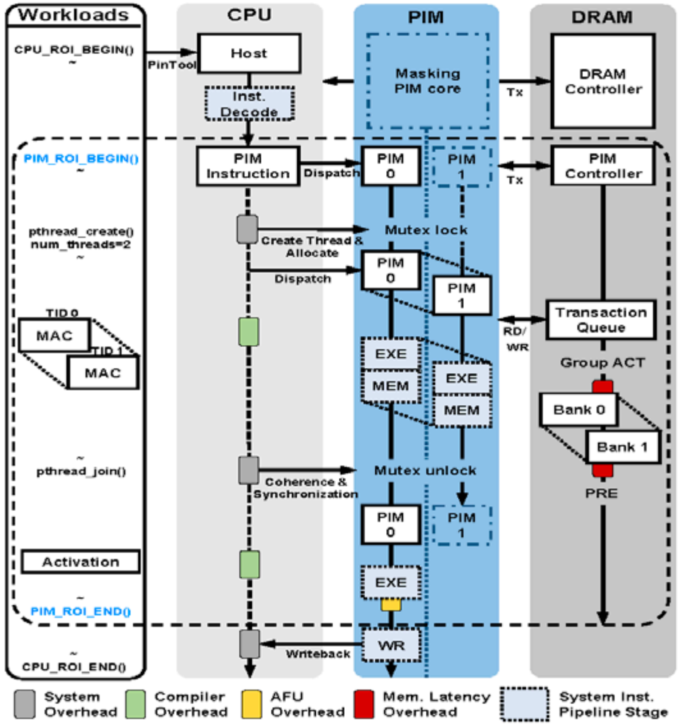
### 1 Comprehensive PIM-CPU configuration support

Validated cycle-level fidelity (≤ 6 % latency & power error) across CPU-, PIM- and hybrid modes versus Newton and PIM-HBM baselines

### 2 Multi-threaded PIM operation compiler

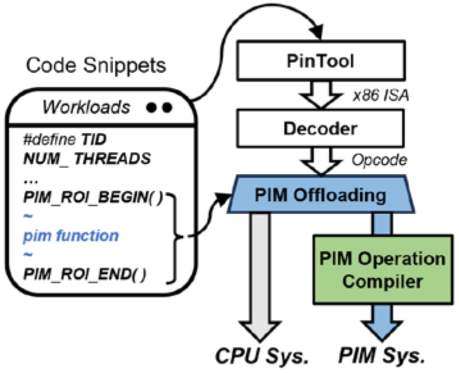
Automatically maps x86 micro-ops in PIM_ROI to various PIM opcodes (MAC, MAXPOOL, ReLU, Softmax), delivering 46.6× speed-up and 15.2× energy efficiency on Llama2-7B

## Simulation flow of HyDRASim

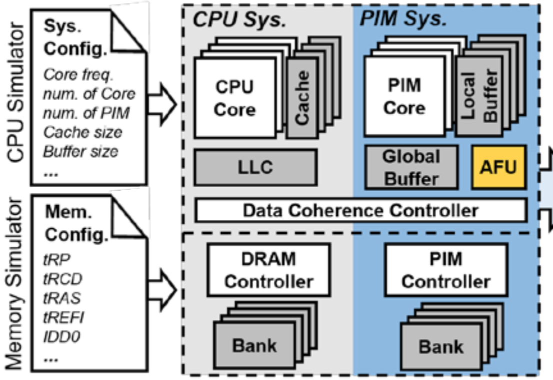


## Overview of HyDRASim

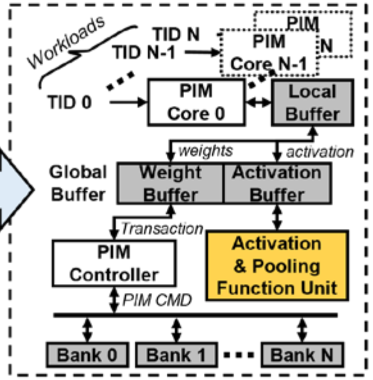
- Application-Level



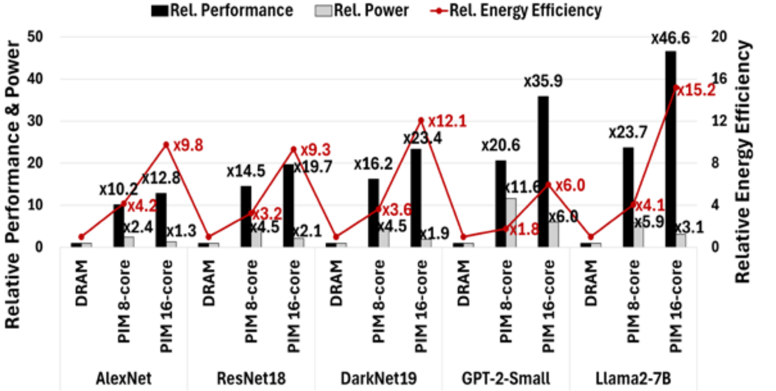
- Architecture-Level



- PIM Structure for PNM Mode



## Relative performance & power consumption



Negligible increase in computations

"HyDRASim: A Versatile and Cycle-Accurate Simulator for Hybrid DRAM PIM-CPU Systems," MASCOTS 2025 (BK21 Top-tier Conf.)

# Emulation Frameworks Supporting Adaptive Configurations in DRAM PIMs

## Goal

**A Fast, cycle-accurate, and scalable FPGA-based PIM emulation framework**

## Motivation 1

- CPU-based software-level simulators limit throughput

## Motivation 2

- Limited FPGA BRAM capacity for PIM emulation

## Solution/ Contribution

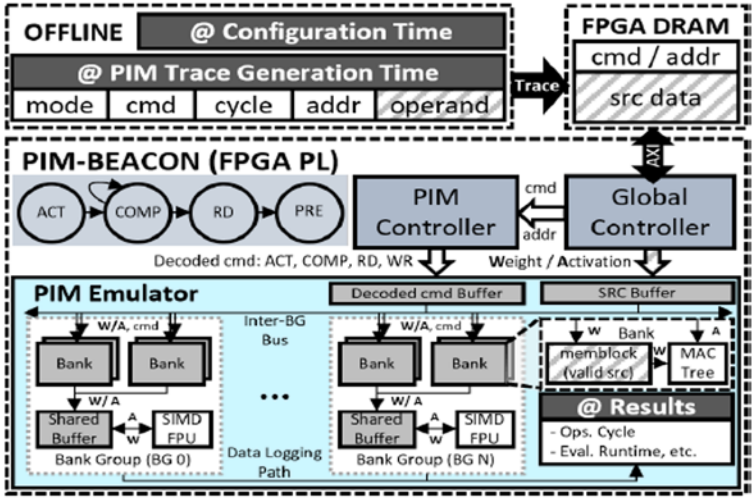
### 1 Trace-driven, cycle-accurate FPGA emulation

**FSM-based controller** decouples macro-command scheduling & **Offline traces** provide reproducible macro-commands & **Modular design** provides per-cycle accuracy and simplifies exploration of various DRAM-PIM architectures

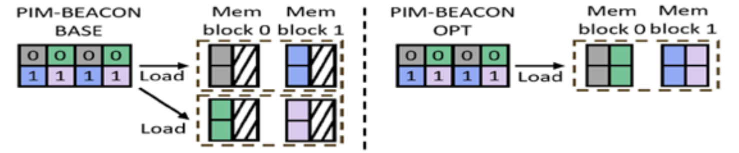
### 2 Memblock-based memory management & Optimizations

**Memblock** caches active DRAM regions reducing BRAM usage & **Hit/Miss logic** avoids redundant reloads & **Dynamic latency** hiding overlaps operations & **Adaptive memblock utilization** optimizes efficiency via BRAM remapping

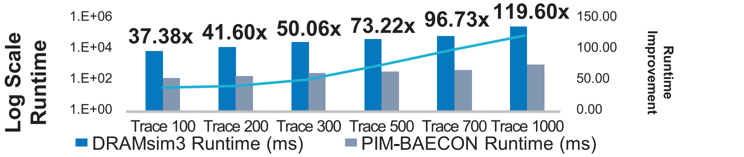
## Overview of proposed PIM emulation framework



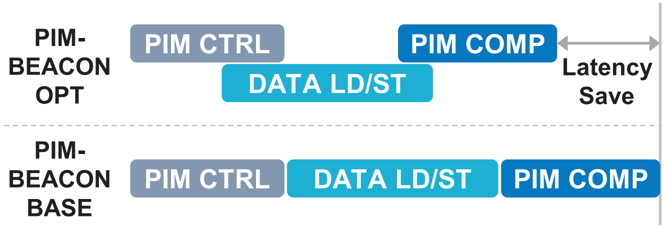
## Proposed adaptive memblock utilization



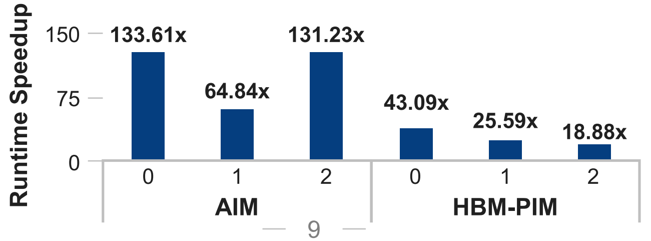
## Runtime speedup evaluation (vs DRAMsim3)



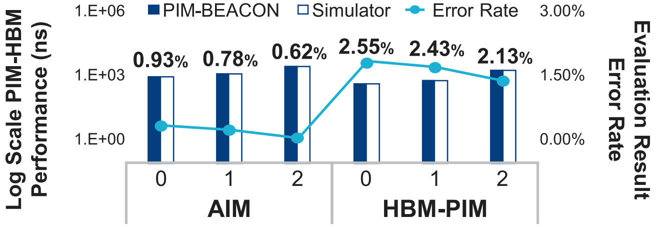
## Timing diagram of the proposed dynamic latency hiding method



## Runtime speedup evaluation (vs in-house PIM simulators)



## Performance reliability evaluation (vs in-house PIM simulators)



"PIM-BEACON: A Benchmarking and Emulation Framework Supporting Adaptive Configurations in Processing-in-Memory Systems," ISPASS 2025 (BK21 Top-tier Conf.)

# General-Purpose Programmable PIM with a Dual-Tier ISA

## Goal

Develop a **general-purpose PIM architecture** that bridges **functionality** and **programmability** gaps for end-to-end AI acceleration

## Motivation

- Existing PIMs fail to support the end-to-end execution of modern heterogeneous AI workloads due to critical gaps in functionality and programmability

## Solution/ Contribution

### 1 Synergistic Heterogeneous Architecture

Co-locating 128 In-Bank PEs (for bandwidth) and 8 Logic-Die Clusters (for compute) within a single HBM3 stack

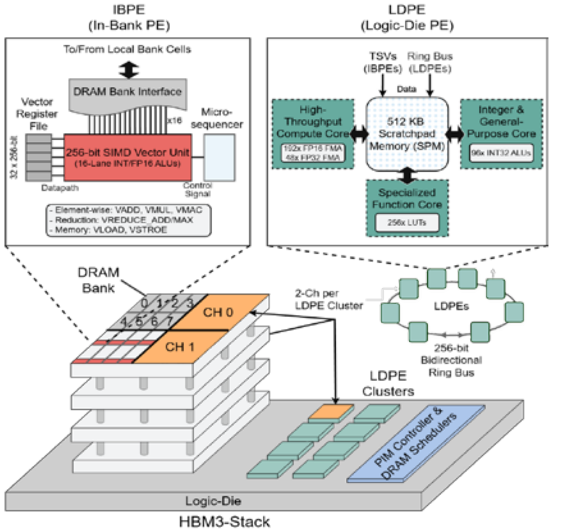
### 2 Dual-Tier Fused ISA

A unified instruction set that fuses complex subgraphs into single commands, minimizing host interaction and data movement

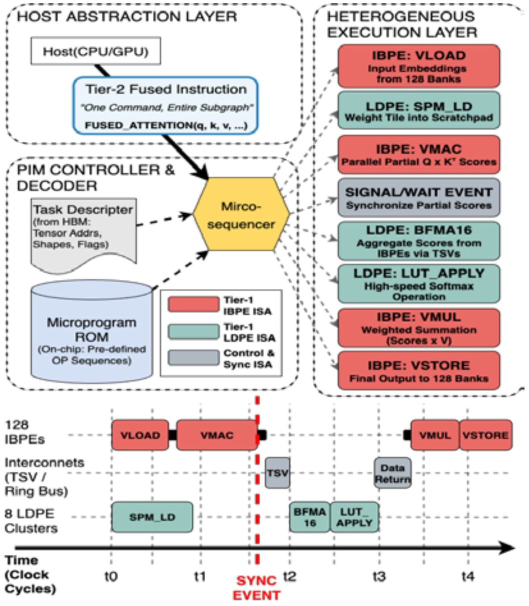
### 3 HW/SW Co-Design & Performance

Minimize off-chip traffic by up to 83% and achieves 19.1x higher energy efficiency and up to 43.5x speedup (LLaMA3) compared to NVIDIA A100

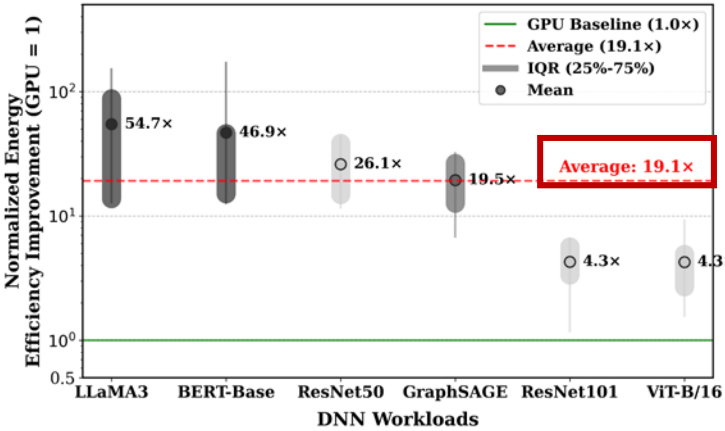
## Overall architecture of proposed GPPIM



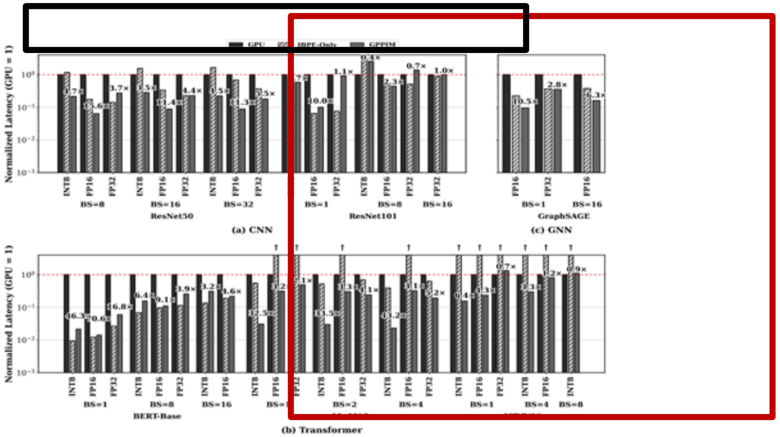
## The fused ISA execution model


HOST ABSTRACTION LAYER
Host(CPU/GPU)
Tier-2 Fused Instruction
"One Command, Entire Subgraph"
FUSED_ATTENTION(q, k, v, ...)
PIM CONTROLLER & DECODER
Task Descripter
(from HBM: Tensor Addrs, Shapes, Flags)
Mirco-sequencer
Microprogram ROM
(On-chip: Pre-defined OP Sequences)
Tier-1 IBPE ISA
Tier-1 LDPE ISA
Control & Sync ISA
HETEROGENEOUS EXECUTION LAYER
IBPE: VLOAD
Input Embeddings from 128 Banks
LDPE: SPM_LD
Weight Tile into Scratchpad
IBPE: VMAC
Parallel Partial Q x K' Scores
SIGNAL/WAIT EVENT
Synchronize Partial Scores
LDPE: BFMA16
Aggregate Scores from IBPEs via TSVs
LDPE: LUT_APPLY
High-speed Softmax Operation
IBPE: VMUL
Weighted Summation (Scores x V)
IBPE: VSTORE
Final Output to 128 Banks
128 IBPEs
Interconnets (TSV / Ring Bus)
8 LDPE Clusters
VLOAD
VMAC
VMUL
VSTORE
TSV
Data Return
SPM_LD
BFMA 16
LUT_ APPLY
Time (Clock Cycles)
t0
t1
SYNC EVENT
t2
t3
t4

## Energy efficiency improvement of GPPIM over GPU (A100)



## Performance speedup of GPPIM over GPU and Baseline PIM (IBPE-Only)



"GPPIM: General-Purpose Programmable PIM with a Dual-Tier ISA for End-to-End Semantic Fusion," ASPLOS 2026, Submitted

# Heterogeneous NPU-PIM Execution

## Goal

**To alleviate decode-phase memory bottlenecks and low-latency LLM/VLM inference by dynamically rebalancing between an NPU and PIM on a unified memory subsystem with co-execution**

## Motivation 1

- LLM and VLM inference is phase heterogeneous, with compute-bound prefill and memory-bound decode, and KV-cache traffic that scales with sequence length stresses memory bandwidth

## Motivation 2

- Static accelerator mappings and blocking PIM execution are not robust to variation in batch size or sequence length, producing low utilization and unstable latency

## Solution/ Contribution

### 1 Unified memory and programming model

A unified address space with memory-mapped control enables the RISC-V core, NPU, and PIM to share buffers without copies and access memory concurrently

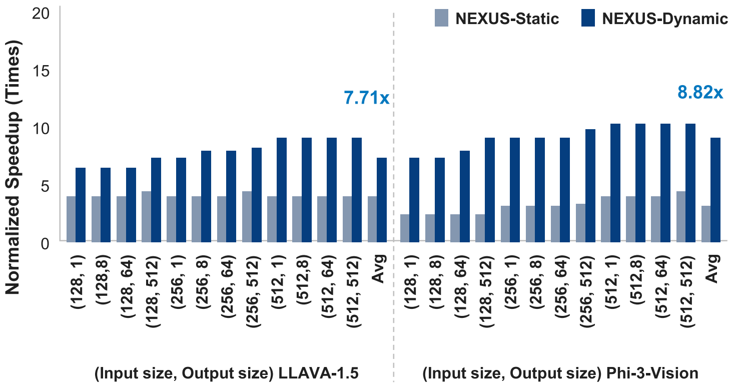
### 2 Asynchronous co-execution

A callback-driven nonblocking runtime overlaps NPU compute and DMA with in-memory PIM operations to improve concurrency and reduce stalls

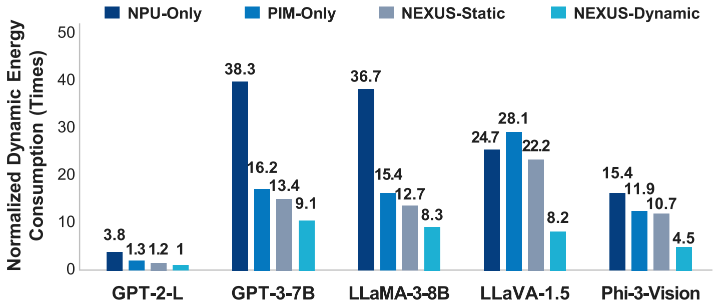
### 3 Dynamic partitioning with tile alignment

Runtime counters drive adaptive NPU-PIM offloads per operator that stay within a stable range and align to NPU tiles, with residual rows directed to the PIM

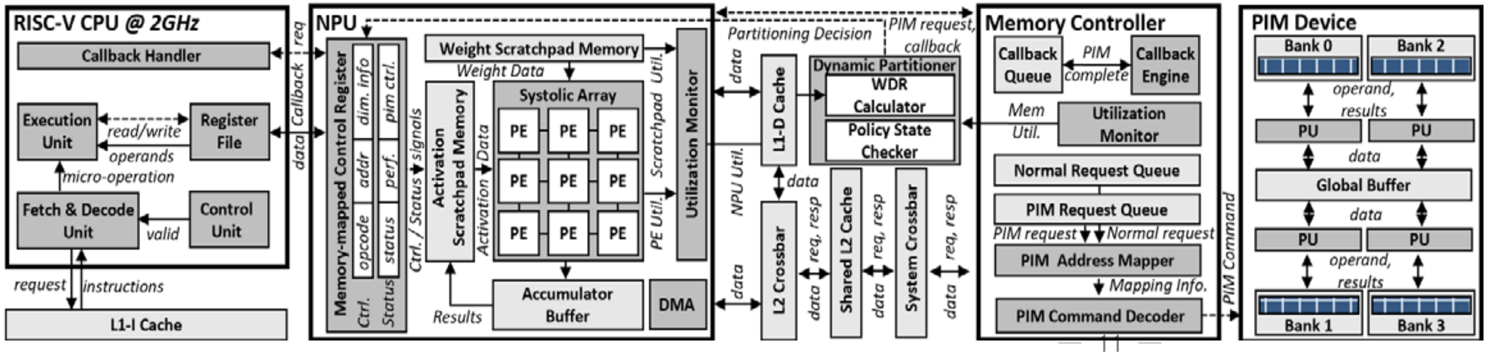
## VLM performance scalability analysis of NEXUS



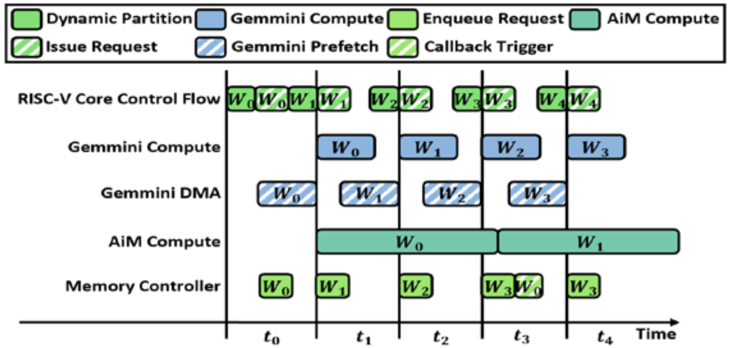
## Energy Efficiency Analysis of NEXUS



## An overall architecture of NEXUS



## Asynchronous execution of timeline of NEXUS



"NEXUS: Heterogeneous NPU-PIM EXecution with Unified Memory Subsystem for Dynamic Workload Partitioning," ASPLOS 2026 (BK21 Top-tier Conf.), Submitted

IDSL Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

# PIM-Aware Efficient Compression for Embedding Layers in sLLMs

## Goal

To **enhance the efficiency of embedding layer processing in sLLM environments through PIM based optimizations**

## Motivation 1

- In quantized sLLMs, embedding layers constitute a significant share of the overall model parameters

## Motivation 2

- Embedding layer computations exhibit low arithmetic intensity (operations per byte), rendering them memory-bound

## Solution/ Contribution

### 1 XOR-based masking compressor (XMC)

A lossless compression scheme tailored for sLLM embedding layers, achieving 1.49× higher compression ratio

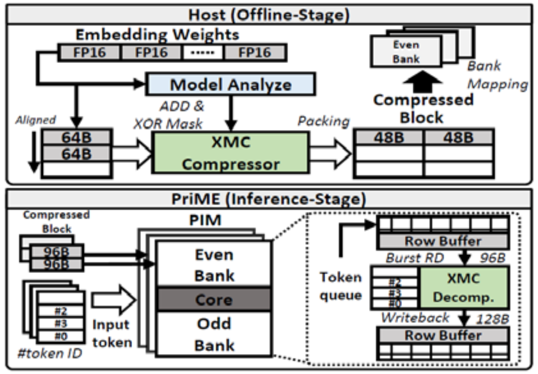
### 2 Hardware-friendly decompressor

Achieves a 3-cycle decompression latency with only 0.05% area overhead compared to conventional HBM

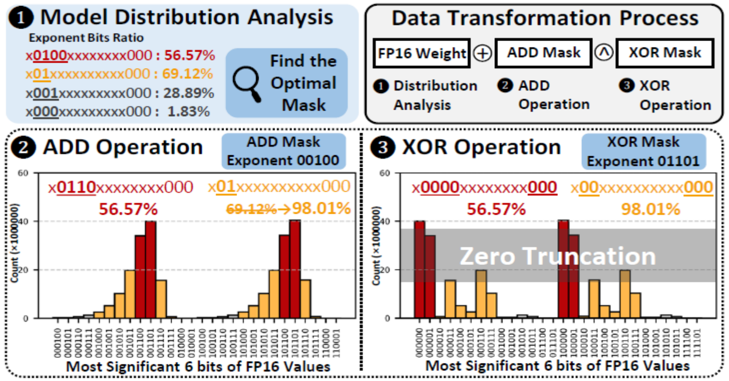
### 3 PIM optimization with XMC

Embedding-layer-aware dataflow optimization enables a 3.95× speedup over GPUs and an 11.5× improvement over conventional HBM-PIM

## Overview of proposed PriME architecture



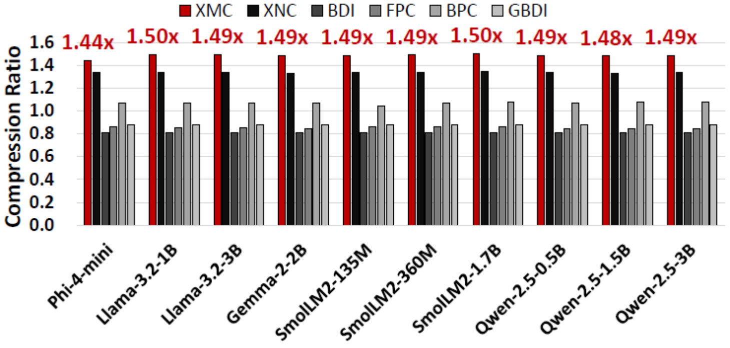
## Data transformation process /w optimal mask value



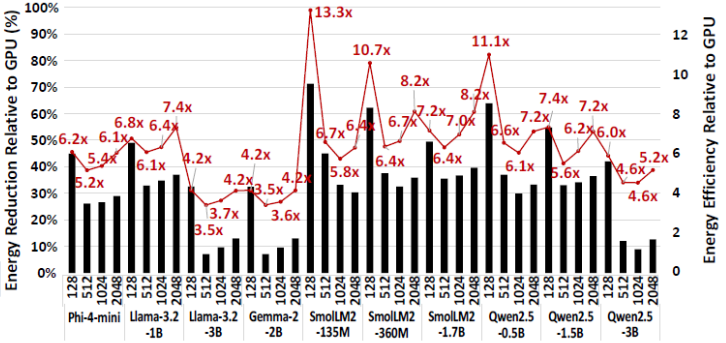
## HW overhead & performance of PriME Decomp.

| Metrics | Area ($\mu m^2$) | Power (mW) | Throughput (GB/s) | Delay (cycles) |
|---|---|---|---|---|
| Decompressor | 40110 | 4.7944 | 10.67 | 3 |

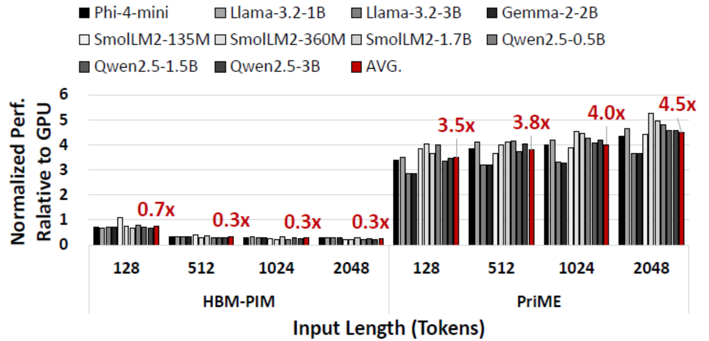
## Comp. ratio comparison of each method



## Energy reduction/efficiency Comparison



## GPU normalized perf. PriME & HBM-PIM



"PriME: PIM-Aware Efficient Compression for Memory-Bound Embedding Layers in sLLMs," ICCD 2025 (BK21 Top-tier Conf.)

# Sparsity-Preserving Attention Replacement for PIM-based LLM Inference

## Goal

**Accelerating LLM scaled dot-product attention inference (SDPA) by leveraging dynamic sparsity and bank-level parallelism in HBM-based PIM**

## Motivation

- Because SDPA causes severe bottlenecks in NPUs, there is a strong need for PIM that effectively exploits sparsity

## Solution/ Contribution

### 1 Elemax: A Highly Parallelizable Softmax Alternative

Introduce Elemax, a ReLU-based and highly parallelizable alternative to Softmax
→ reduce attention latency eliminating expensive operations such as exponential, division

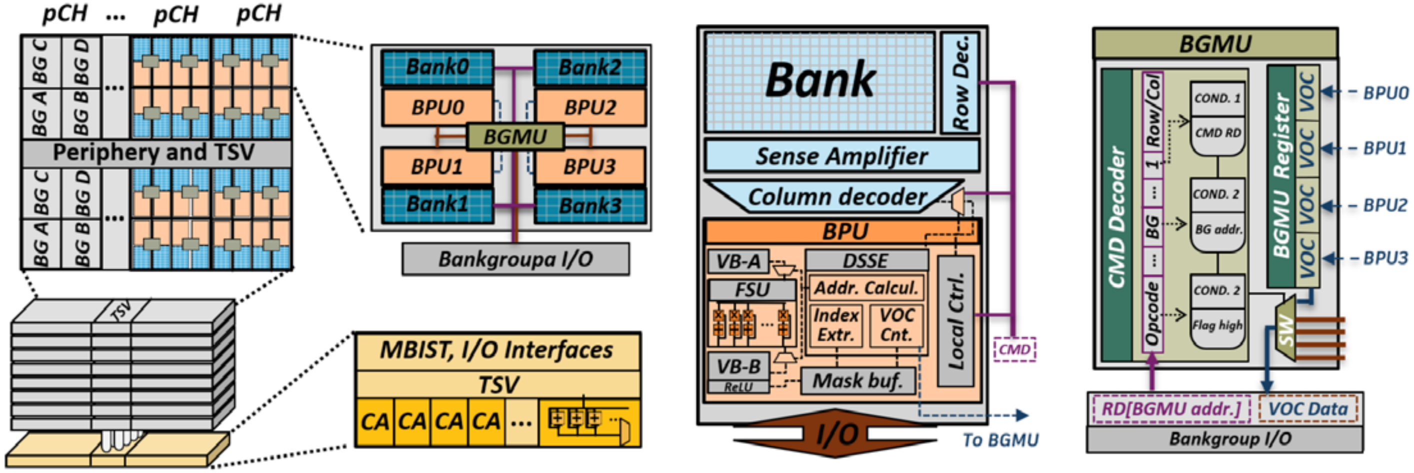
### 2 SPARE-PIM: Dynamic-Sparsity-Aware PIM Architecture

Design SPARE-PIM, a dedicated PIM architecture for Elemax → hide the latency of sparsity detection, exploit it to prune unnecessary operations, and maximize parallelism

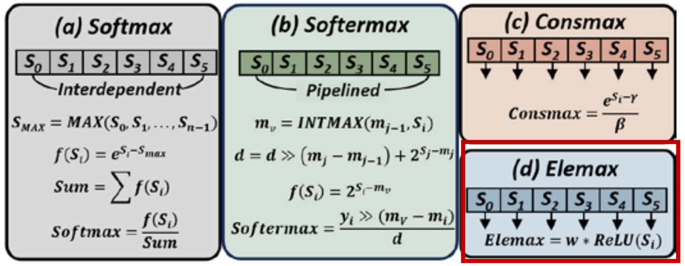
### 3 Single-Stream SDPA Acceleration Dataflow

Develop a single-stream SDPA acceleration dataflow → skip bank writes for intermediate activation results and reuse data in place to improve PIM throughput

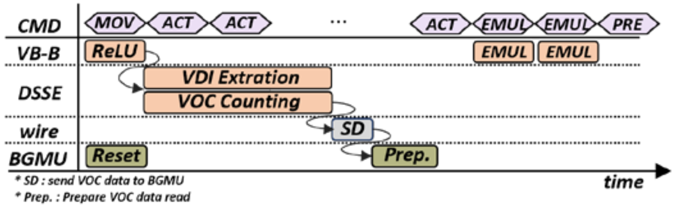
## Overall architecture of SPARE-PIM for exploiting dynamic sparsity



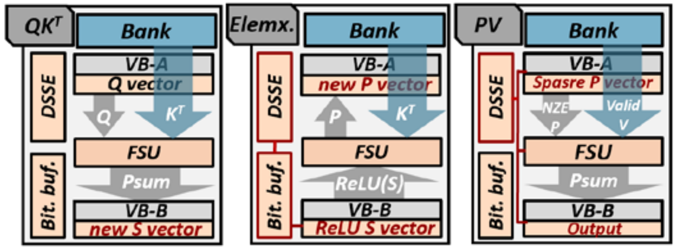
## Comparison of Softmax Replacements


(a) Softmax
$S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$
Interdependent
$S_{MAX} = MAX(S_0, S_1, \dots, S_{n-1})$
$f(S_i) = e^{S_i - S_{max}}$
$Sum = \sum f(S_i)$
$Softmax = \frac{f(S_i)}{Sum}$
(b) Softermax
$S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$
Pipelined
$m_v = INTMAX(m_{j-1}, S_i)$
$d = d \gg (m_j - m_{j-1}) + 2^{S_j - m_j}$
$f(S_i) = 2^{S_i - m_v}$
$Softermax = \frac{y_i \gg (m_V - m_i)}{d}$
(c) Consmax
$S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$
$Consmax = \frac{e^{S_i - \gamma}}{\beta}$
(d) Elemax
$S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$
$Elemax = w * ReLU(S_i)$

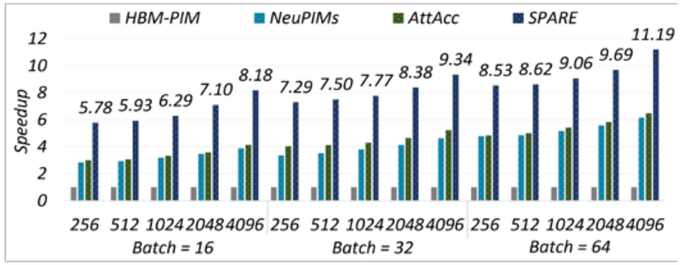
## Timing Diagram of Sparsity Detection



## Core Dataflow of Accelerating SDPA Inference



## Comparison with Existing PIM-Based Accelerators



"SPARE: Sparsity-Preserving Attention Replacement for Efficient PIM-based LLM Inference," IEEE Transactions on Computers, Submitted (IF 3.8)