Memory-Centric Computing with SK hynix's Domain-Specific Memory

Yongkee Kwon, Guhyun Kim, Nahsung Kim, Woojae Shin, Jongsoon Won, Hyunha Joo, Haerang Choi, Byeongju An, Gyeongcheol Shin, Dayeon Yun, Jeongbin Kim, Changhyun Kim, Ilkon Kim, Jaehan Park, Chanwook Park, Yosub Song, Byeongsu Yang, Hyeongdeok Lee, Seungyeong Park, Wonjun Lee, Seongju Lee, Kyuyoung Kim, Daehan Kwon, Chunseok Jeong, John Kim, Euicheol Lim and Junhyun Chun, SK hynix inc.

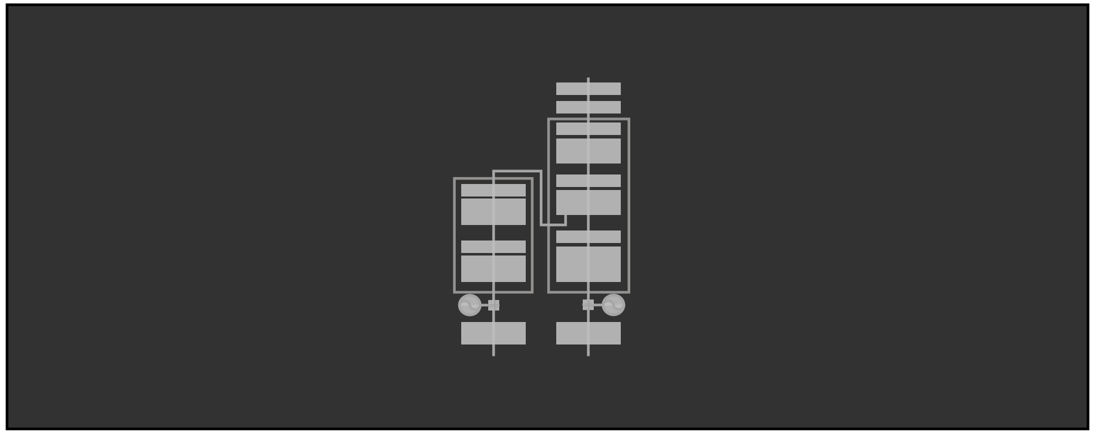
- Memory-centric Computing for LLMs
- Accelerator-in-Memory (AiM)
- Efficiently Scaling AiM for LLM inferences
- System Analysis: Proof-of-Concept, Performance Analysis and System Deployment
- Conclusion

Generative AI and Inference Cost

Generative AI

"This new technology can help people everywhere improve their lives"*

Large Language Models (LLMs)
behind the generative AI boom



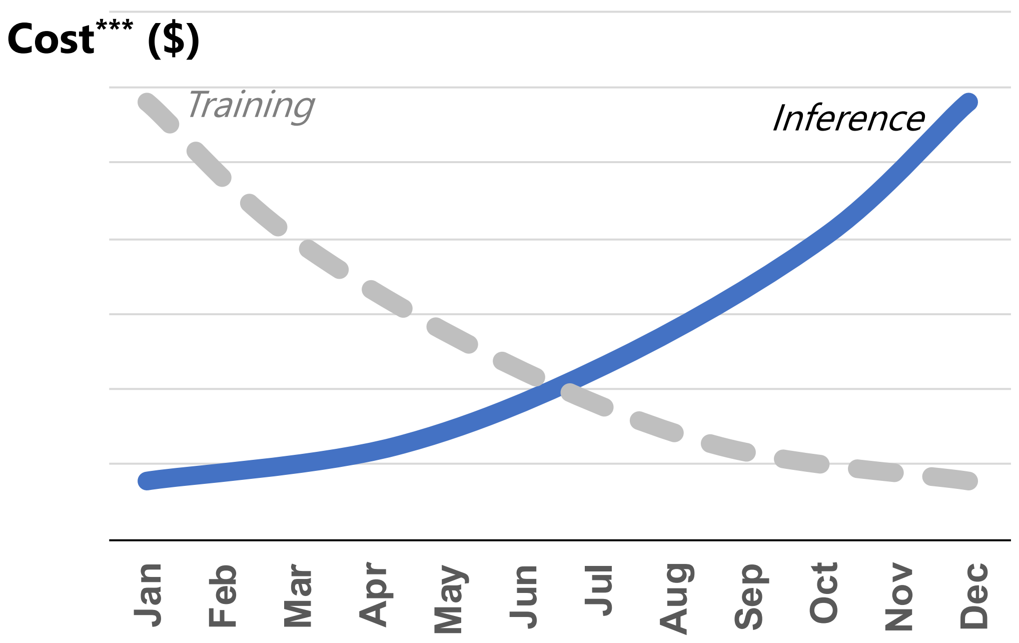
Chat **Code** **Translation** **Search** **Q&A**

(*) "The Age of AI has begun", Bill Gates, March,2023

Generative AI services

"Inference Costs Eclipses Training Costs Over Time"

Inference is all about efficiency
- Performance, Cost**, and Energy -

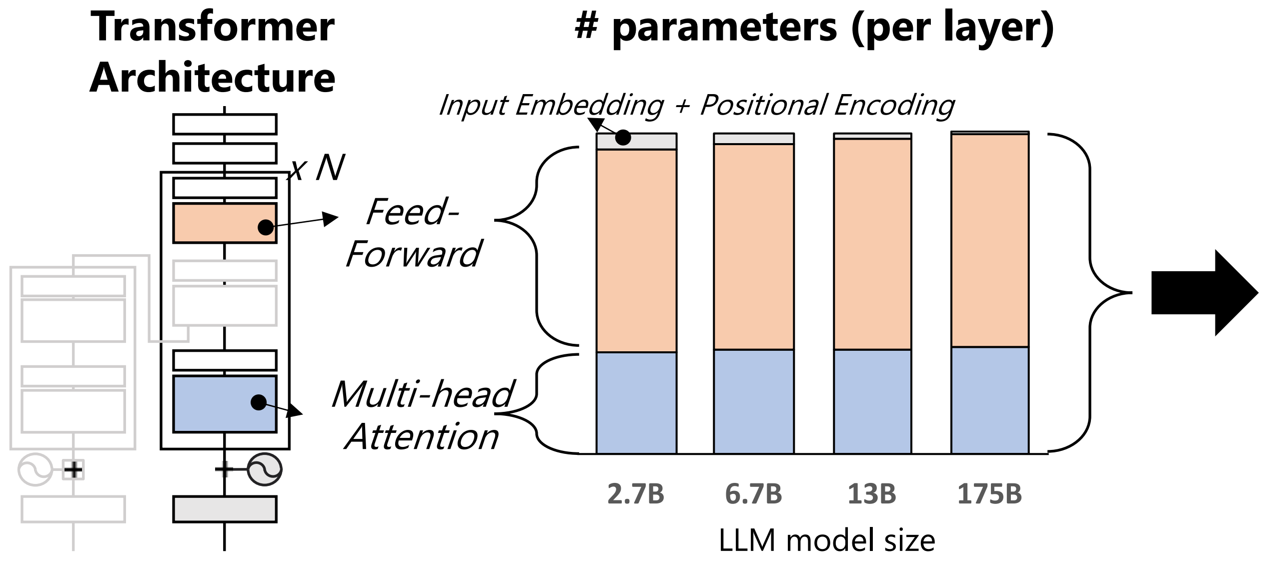


*(**) TCO (Total Cost of Ownership) ~ CapEx + 3 * OpEX*
(***) Inference cost ~ #users, assumed to be growing over months

Large Language Model: “It’s the Memory, …”

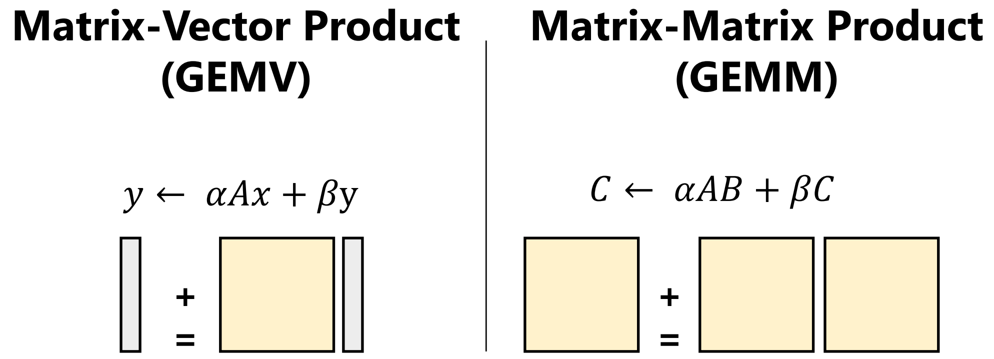
Transformer model
Fundamental Building block of all LLMs

- Transformer autoregressive decoder*: many large matrix-vector multiplications (or GEMV)



Matrix-Vector Multiplication
All about moving matrices

- GEMV: memory BW-bound with low arithmetic intensity
- GEMM: compute-bound with sufficient reuses

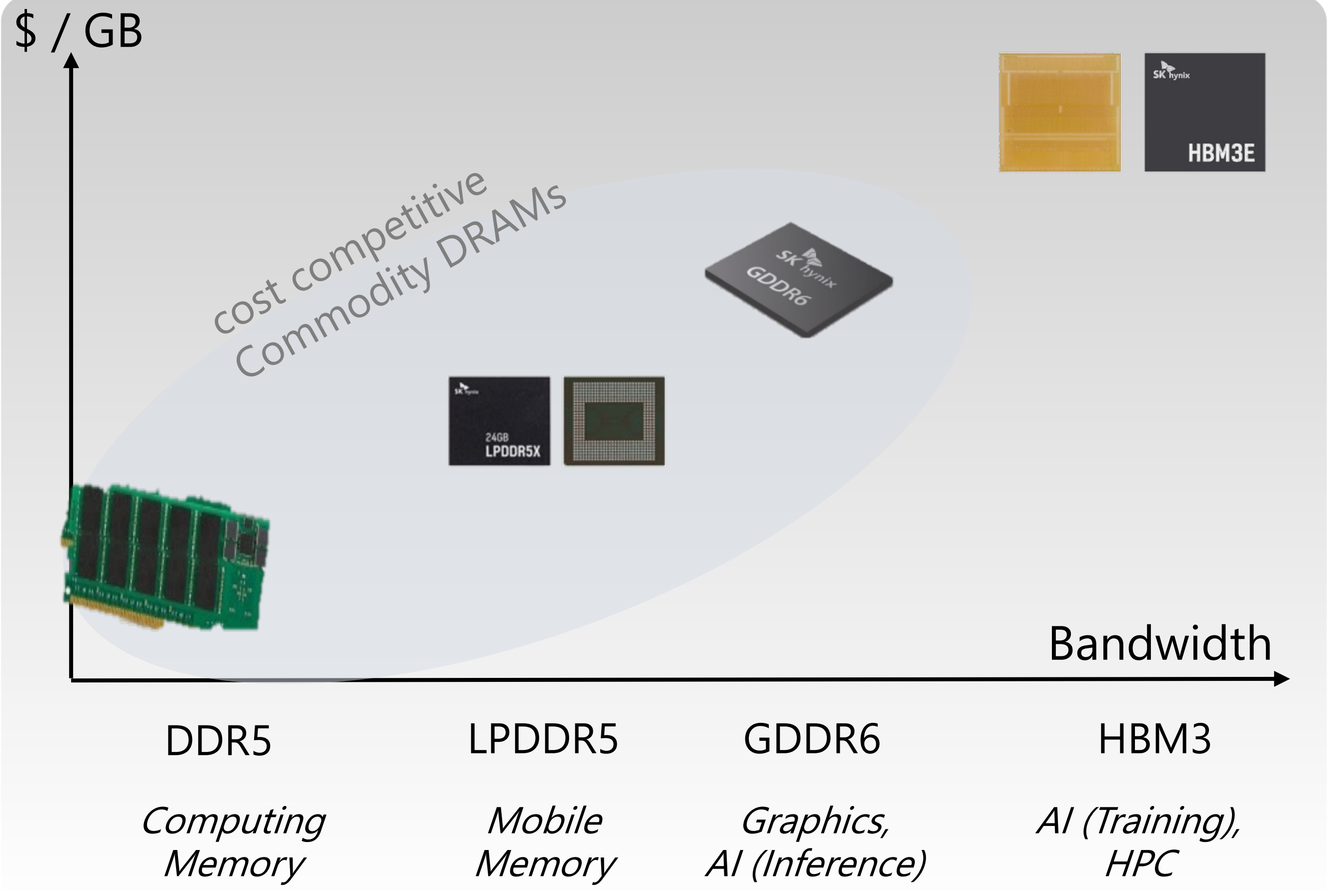

Matrix-Vector Product (GEMV)
$y \leftarrow \alpha Ax + \beta y$
=
+
Matrix-Matrix Product (GEMM)
$C \leftarrow \alpha AB + \beta C$
=
+

∴ Memory-centric computing for efficient LLM inferences

(*) Assumptions: batch1 inference during output token generation phase

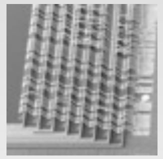
Why Domain-Specific Memory?

SK hynix's AiM (Accelerator-in-Memory): Domain specific memory for High-Bandwidth / Low-Cost / Low-Power to enable highly efficient memory-centric computing for LLM



Domain Specific Memory

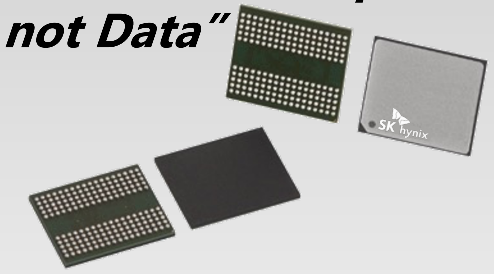
Managed DRAM Solution**



- High capacity
- Cost efficient

*CXL-CME**

AiM: "Move compute , not Data"



- **High Bandwidth**
- **Cost Efficient**
- **Energy Efficient**

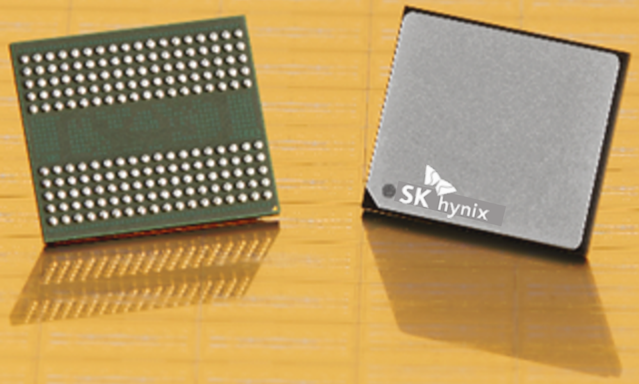
Generative AI, LLMs

(*) CME: Capacity Memory Expansion

(**) [ISSCC'19] A 512GB 1.1V Managed DRAM Solution with 16GB ODP and Media Controller

Accelerator-in-Memory (AiM)

- True All-bank Parallelism
- End-to-end GEMV Acceleration in Memory
- AiM-specific Memory Commands

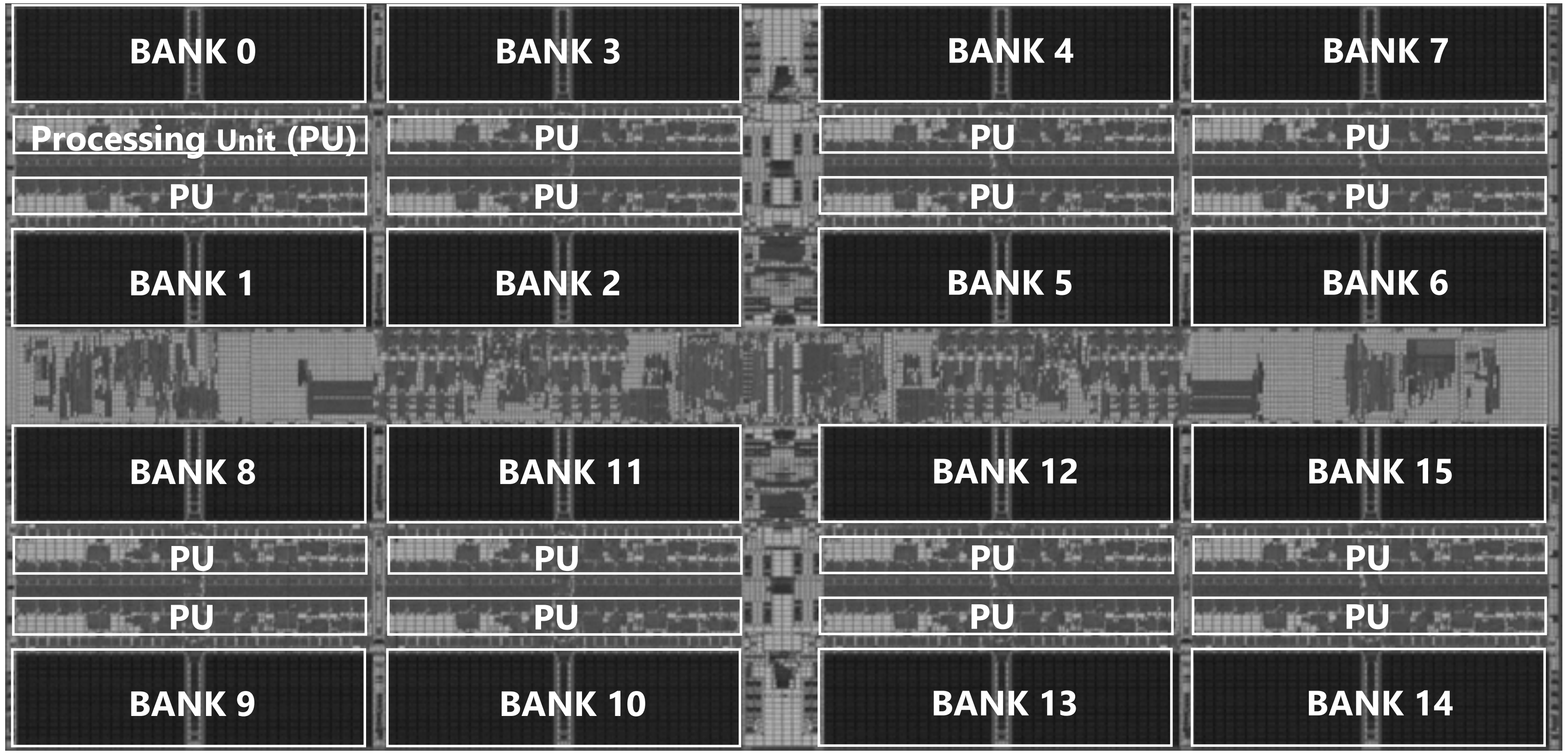

SK hynix

Accelerator-in-Memory: "True All-Bank Parallelism"

SK hynix's First GDDR6-based Processing-in-Memory Product Sample

Design Goals: No Compromise in Parallelism (Performance= Bandwidth)

GDDR6-AiM Die Photograph



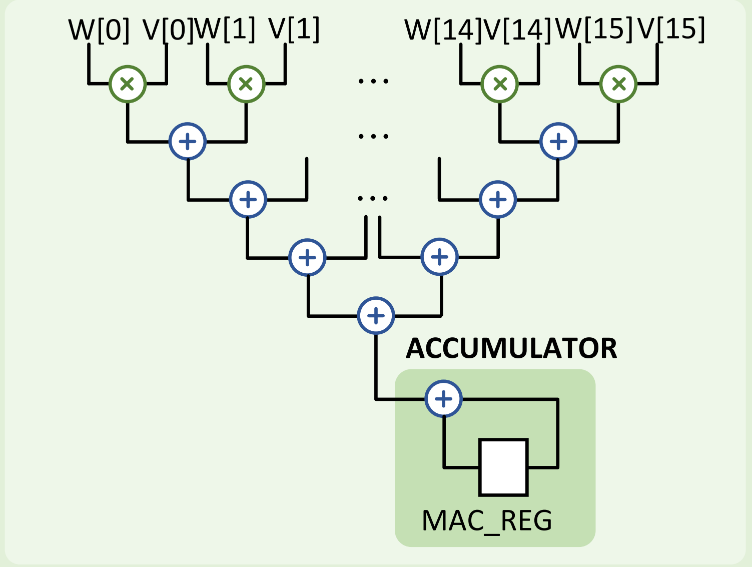
GDDR6-AiM* (per die)	
DRAM Type	GDDR6
Process Technology	1y
Memory Density	4Gb
Organization	X16
IO Data rate	16 Gbs/pin (@1.25V)
(External) Bandwidth**	32 GB/s
Operating Speed	1 GHz
Processing Unit (PU)	16 PU/die
Compute Throughput**	512 GFLOPS
Internal Bandwidth**	512 GB/s
Numeric Precision	BF16
Activation Function support***	Sigmoid, tanh, GELU, ReLU, Leaky ReLU, …

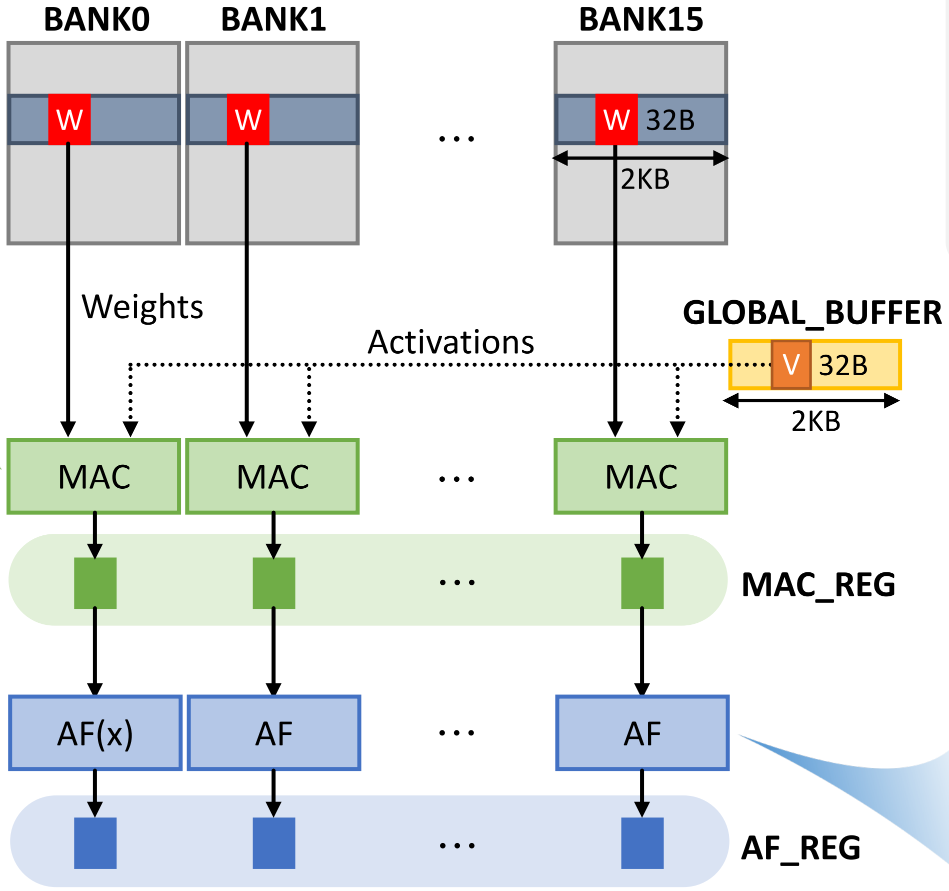
(*) [ISSCC'22] A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications"
(**) Defined as a peak during burst operations
(***) Any customized function may apply with limitation in accuracy bu using internal lookup table and linear interpolation unit.

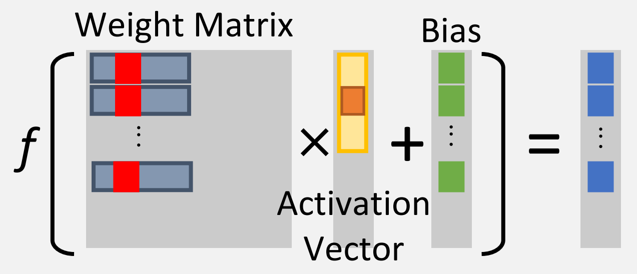
End-to-end GEMV Acceleration in Memory

Multiply-And-Accumulate (MAC)

- Performs MAC operation on **sixteen** BF16 weight matrix and vector elements (corresponds to a single DRAM column access, i.e. 32B).
- Computation results are stored in a dedicated **MAC_REG** set and can be later accessed by the user.

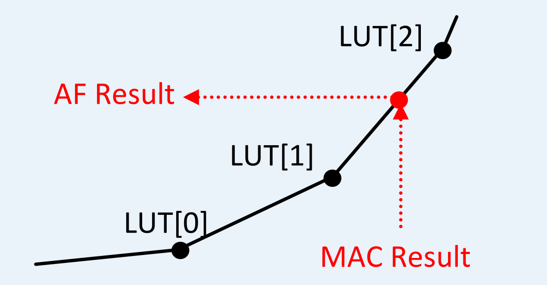






Activation Function Module

- Performs **Activation Function (AF)** computation by linearly interpolating pre-stored AF template data using MAC calculation results.
- Interpolation results are stored in a dedicated **AF_REG** set and can be later accessed by the user.



- **MAC** and **Activation Function** operations can be performed in all banks in parallel.
- **Weight** matrix data is sourced from **Banks**; **Vector** data is sourced from the **Global Buffer**.
- **MAC** results are stored in latches collectively referred to as **MAC_REG**.
- **Activation Function** results are stored in latches collectively referred to as **AF_REG**.

AiM-specific Memory Commands

All-bank operations via single command for high compute efficiency

Multi-Bank Activation		
ACT	4, 16 banks	Activate four/sixteen banks in parallel
ACTAF	4, 16 banks	Activate rows storing Activation Functions LUTs in four/sixteen banks in parallel
Multi-Bank Compute		
MAC	1, 4, 16 banks	Perform MAC (Multiply-and-Accumulate) in one/four/sixteen banks in parallel
AF	16 banks	Compute Activation Function (Non-linear function) in all banks
EWMUL	1, 4 bank groups	Perform element-wise multiplication
Data Transfer		
RDCP	Global Buffer	Copy data from a bank to the Global Buffer
WRCP	Global Buffer	Copy data from the Global Buffer to a bank
WRGB*	Global Buffer	Write to Global Buffer *(often Activation vector data)*
RDMAC*	MAC REG	Read from MAC result register
WRMAC*	MAC REG	Write to MAC result register *(or WRBIAS as often BIAS data is written)*
RDAF*	AF REG	Read from Activation Function result register
WRBK	16 banks	Write to all activated banks in parallel

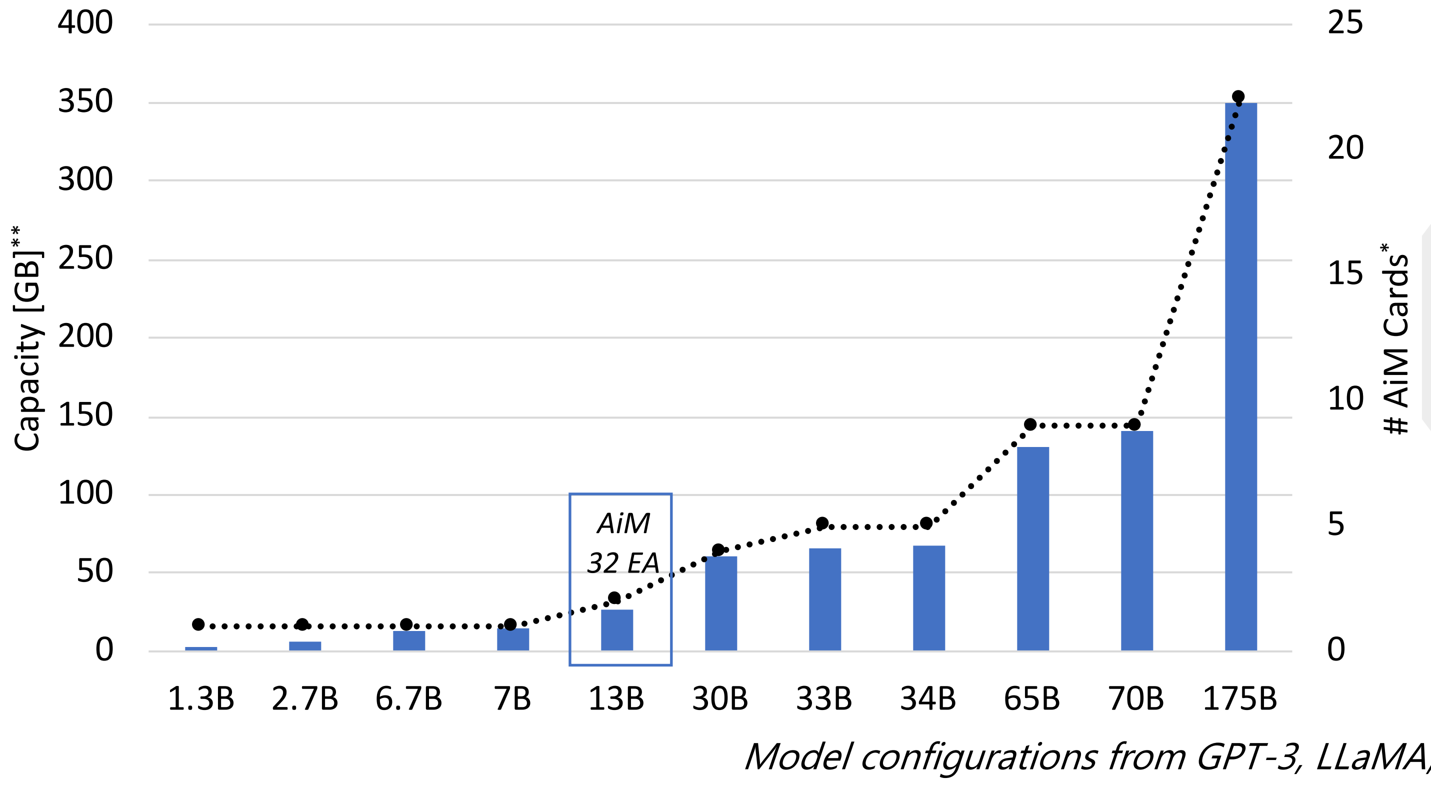
(*) Commands marked as CMD require a special Mode Register set to be recognized.*

Efficiently Scaling AiM for LLM Inferences

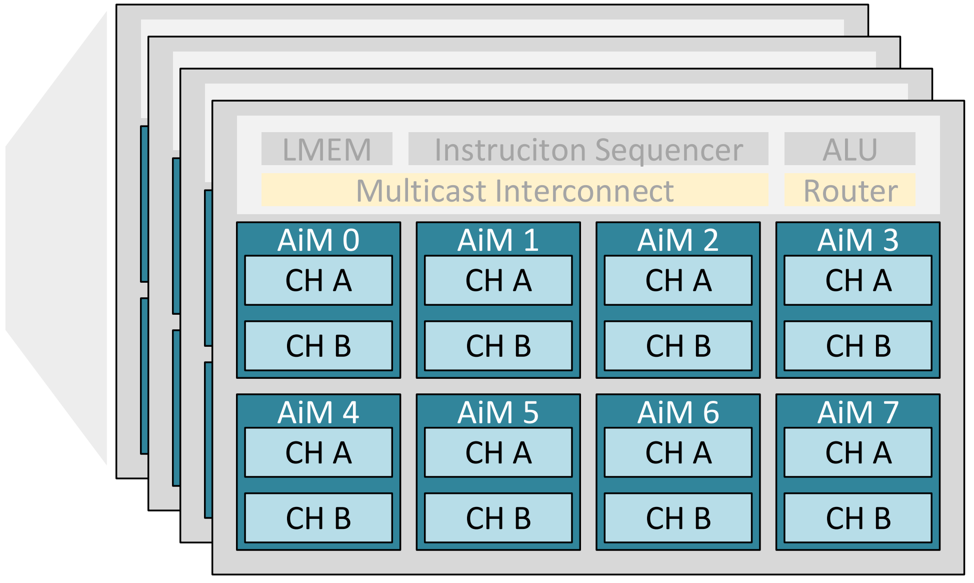
- Practical yet Efficient Mapping: AiM-specific Tiling
- Scalable AiM-based System Architecture
- Matrix-Vector Accumulate Instruction (exploiting AiM Tiling)

On Serving Large Language Models

Large number of AiMs (or memories) required for serving LLMs



AiM card (with 8* AiM packages)

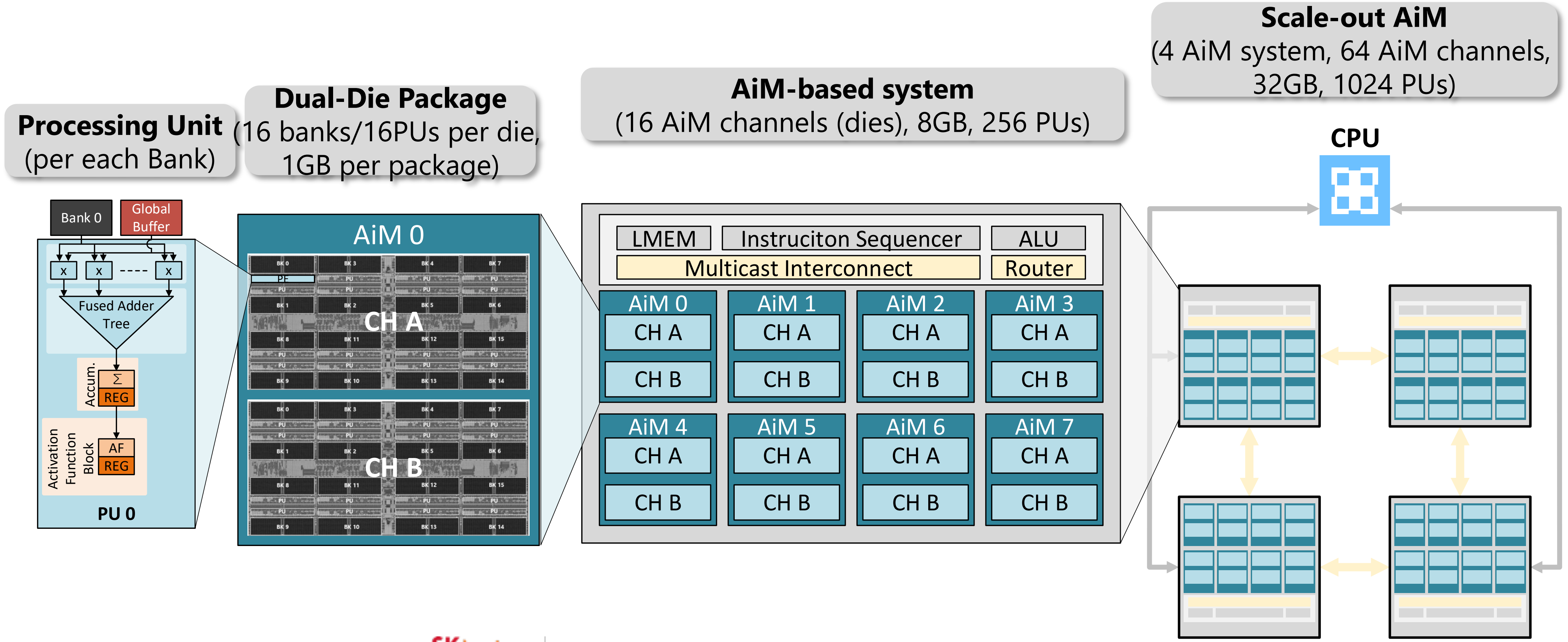


*(*) Assumed 8*AiM packages per each AiM card*
(**) Capacity required to store model weights in 16b precision, excluding Key and Value historys

Scale-out AiM for Large Language Models

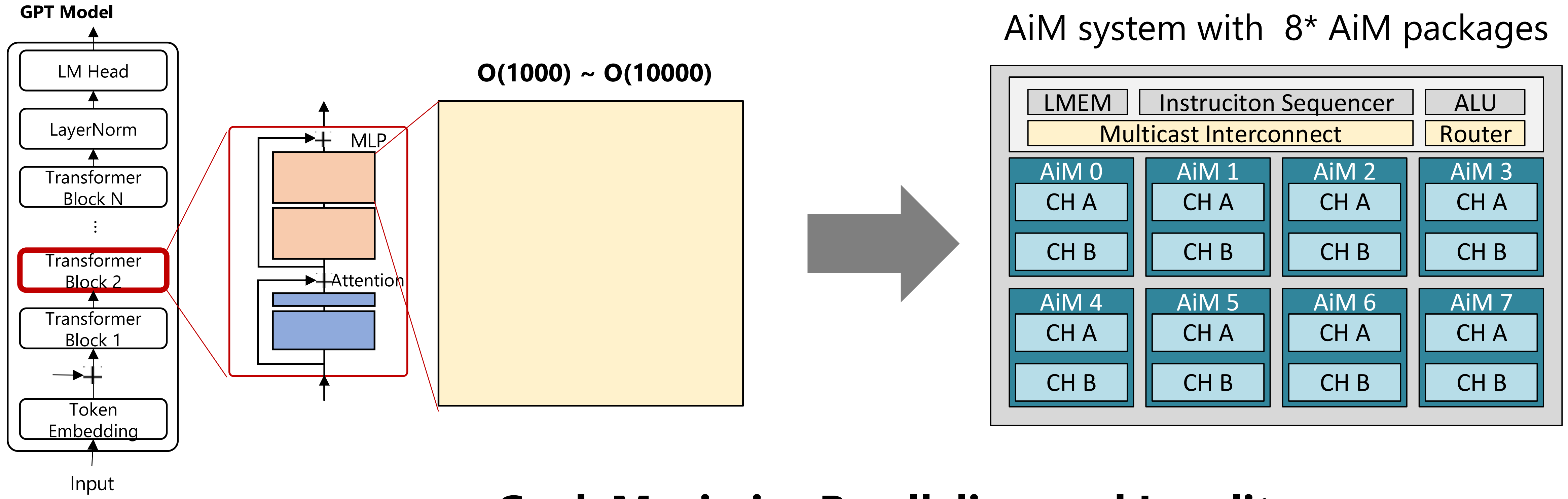
Large number of AiMs (or memories) required for serving LLMs

N * #AiM packages = N * 2 * #AiM channels = N * 2 * 16 * #PUs (or #Banks)



Key Techniques for Efficient Scaling AiMs for LLMs

1. **(Software) Practical yet Efficient Mapping based on AiM-specific Tiling for data (matrix) partitioning**
2. **(Hardware) Scalable AiM-centric System Architecture**
3. **(SW-HW interface) Matrix-Vector Accumulate (with AiM-specific Tiling)**

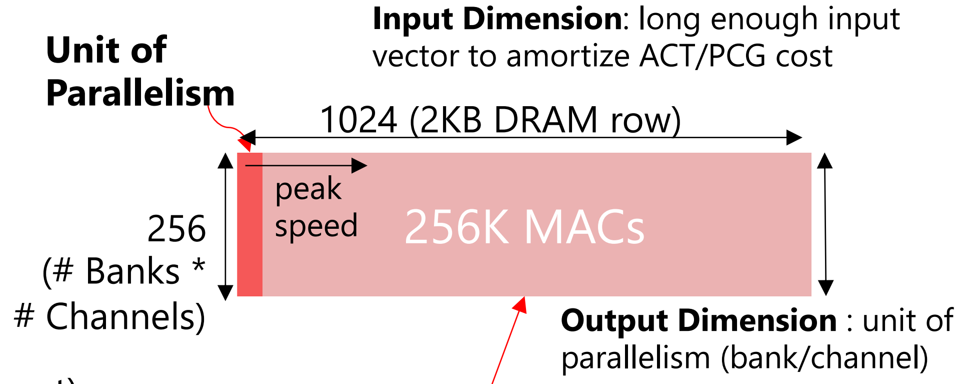


Goal: Maximize Parallelism and Locality

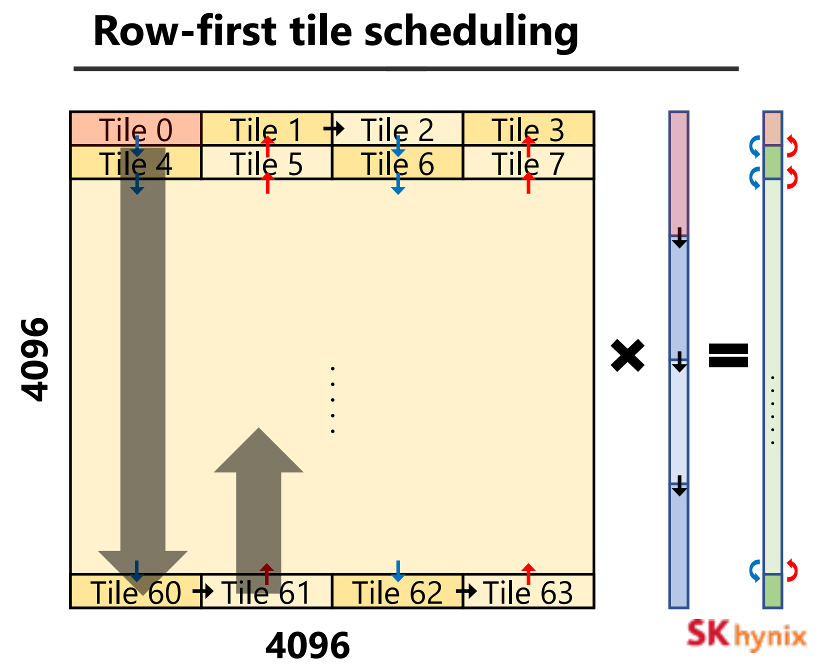
(1) Practical/Efficient Mapping: AiM-specific Tiling

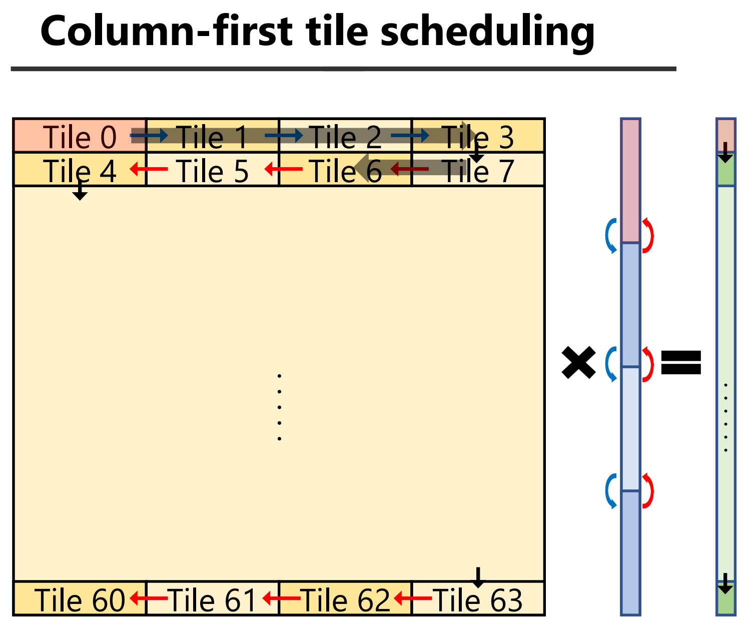
- **Partitioning Large Matrices by Tiling for Parallelism and Locality**
- **Tile Scheduling for Locality**

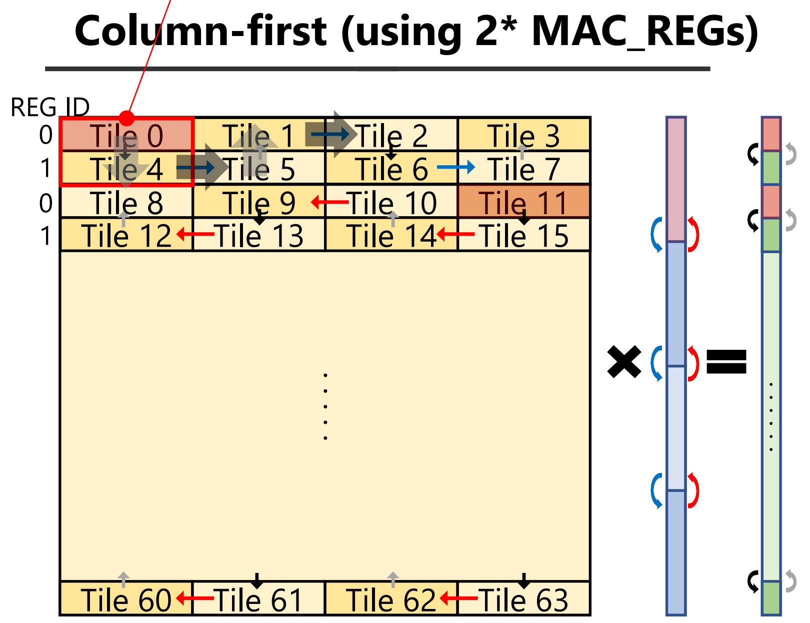
- **Tiling** [shape: #floats per DRAM rows (1024)] x [#banks (16) * # channels (i.e., 16)]
  - to maximize **parallelism** (available across multiple channels and banks)
  - to exploit row-buffer **locality** (amortizing DRAM row switching cost)
- **Tile scheduling** (4094 x 4096 matrix examples)
  - to improve locality (~ maximize vector reuse = minimize offchip data movement)



Unit of Parallelism
Input Dimension: long enough input vector to amortize ACT/PCG cost
1024 (2KB DRAM row)
peak speed
256K MACs
256 (# Banks * # Channels)
Output Dimension : unit of parallelism (bank/channel)



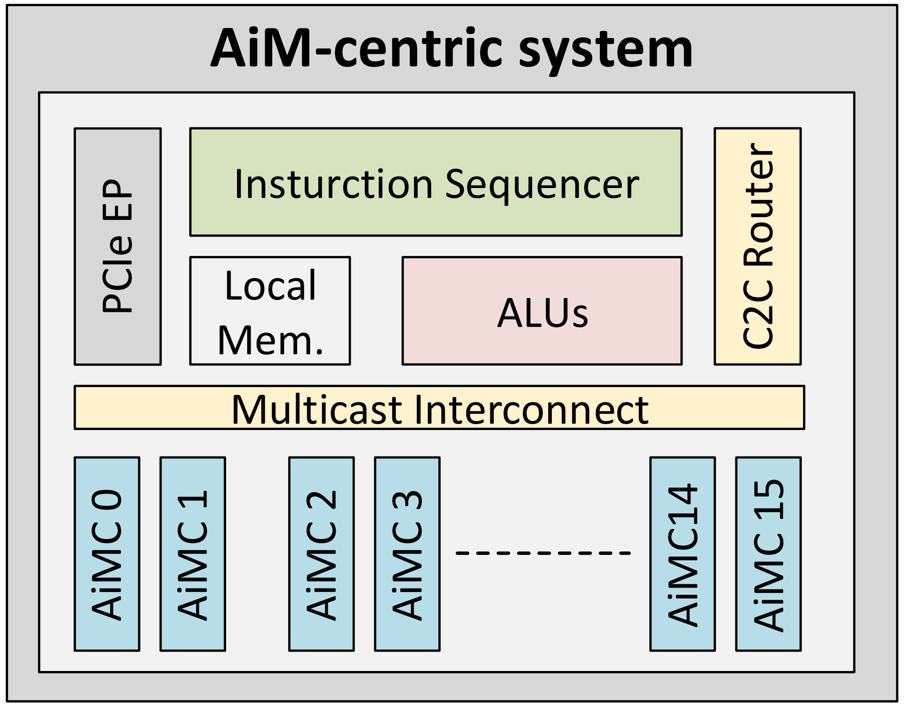




(2) Scalable AiM-based System Architecture

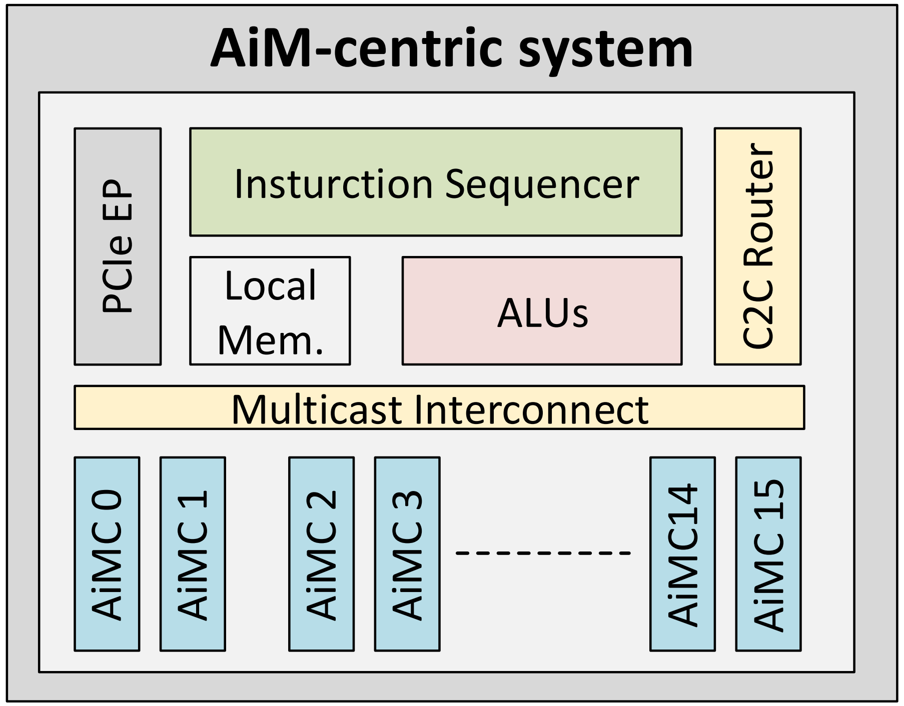
Architecture goals

- **Efficient Scaling (unit of scaling)**
  - Orchestrate multiple AiMs within each AiM-system (scale-up) and across multiple AiM-systems (scale-out)
- **High performance**
  - Maximize compute throughput for a set of AiM devices by keep all AiMs as busy as possible in parallel
  - Exchange/manipulate data (vectors) between storage elements efficiently
- **Easy of programming**
  - Minimize software stack overhead



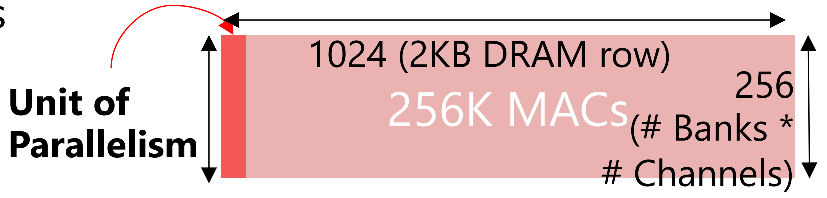
(2) Scalable AiM-based System Architecture

- **AiM Controller:** Generates and schedules low-level AiM and typical DRAM commands.
- **Scalable Multicasting Interconnect:** Enables efficient workload distribution through flexible instruction parallelism. Supports unicast, multicast, and broadcast modes
- **Router:** Custom card-to-card (or chip-to-chip) interconnect for scale-out
- **Compute Unit (ALU):** Layer Normalization, SoftMax, Element-wise addition (residual cut)
- **Instruction Sequencer:** Decodes AiM instructions generated by software and provides direct memory access for the host



(3) Matrix-Vector Accumulate (exploiting AiM tiling)

- **Matrix Vector Accumulate Instruction (Channel mask, Start DRAM address (row, column), #iters, MAC reg ID):**
  - CISC-like instruction decoded/broadcasted to multiple AiM channels
  - Corresponding to unit tile (up to #channels * 16banks * 1024 MACs)



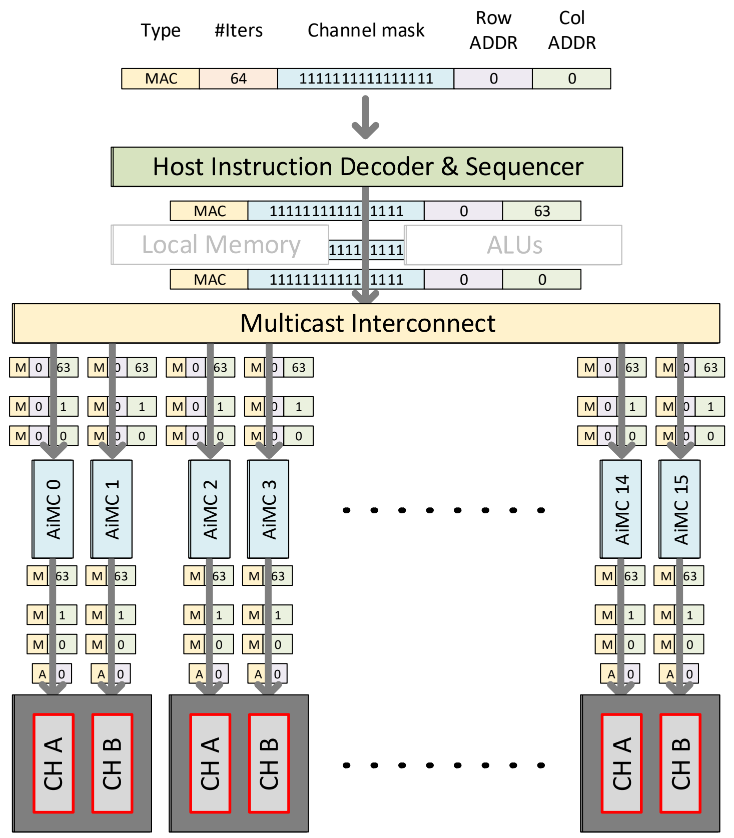
① CISC-like instruction sent from host

② Instruction is further decoded into multiple micro-instructions

③ Micro-instructions are multicasted into each channel

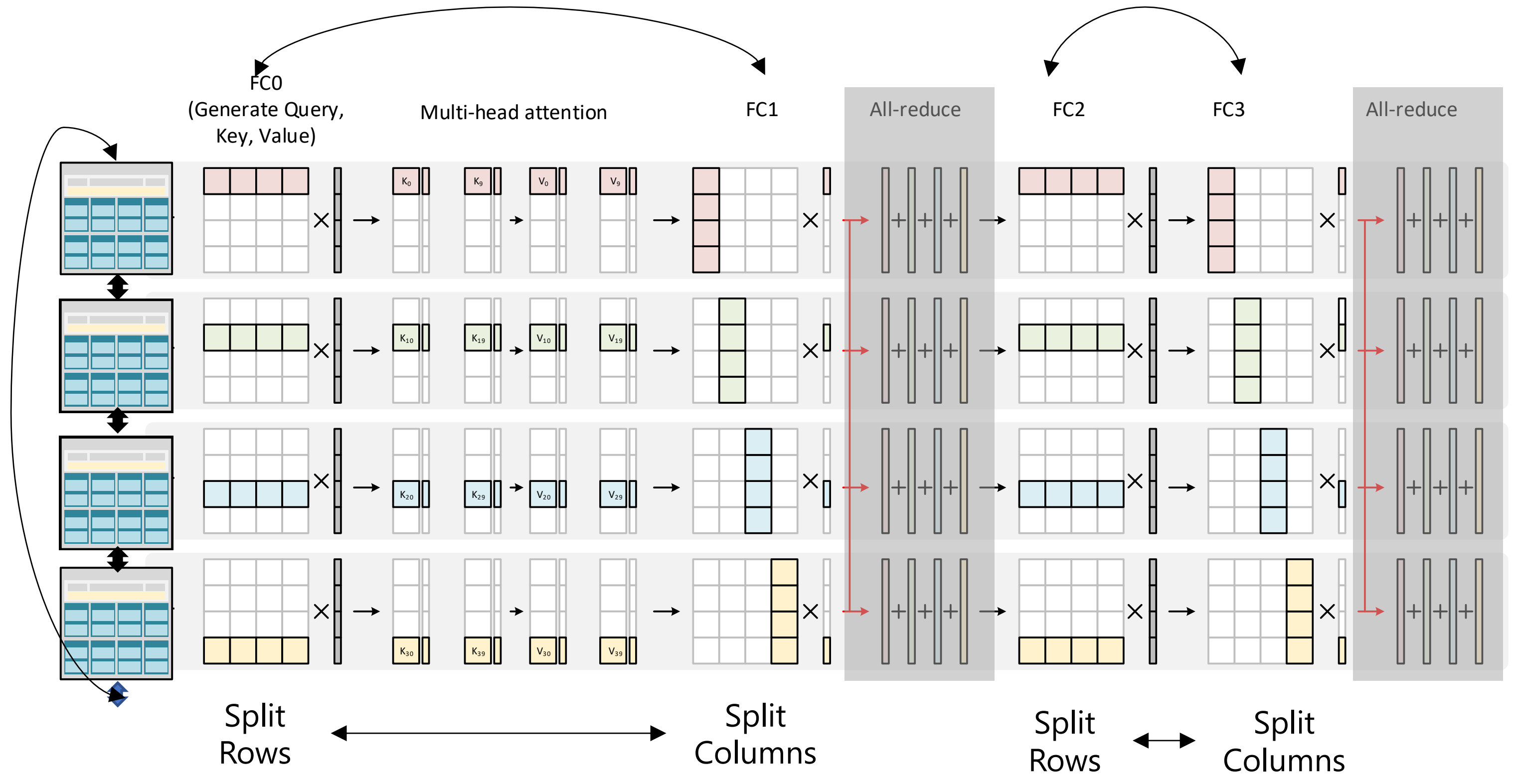
④ Micro-instructions are translated into AiM (DRAM) commands

⑤ Simultaneous AiM operations across multiple channels!



Optimization Techniques for Scale-out Architecture

Pair-wise (row → column) partitioning for two adjacent fully-connected layers



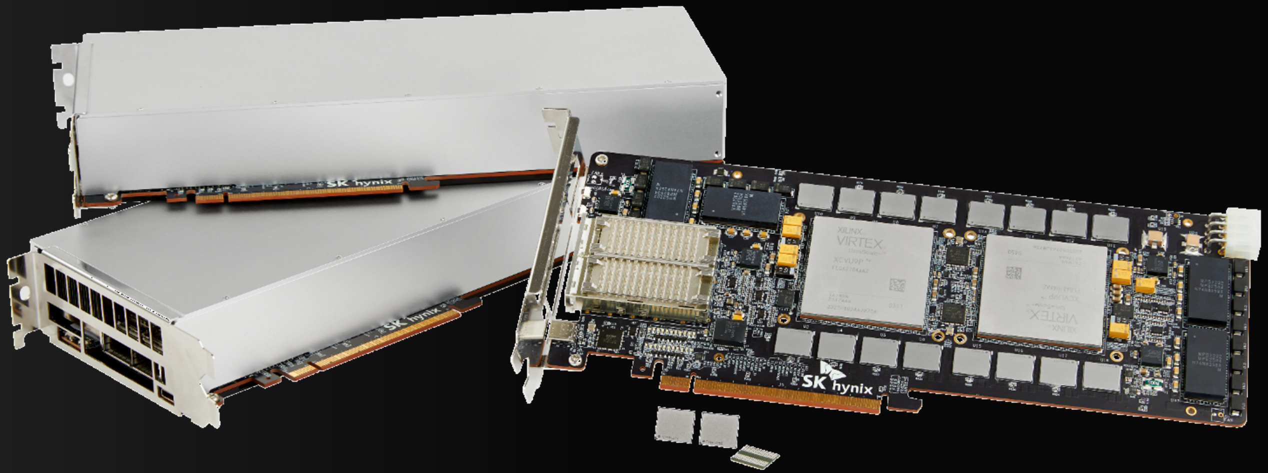
Scheduling All channel/bank AiM refresh operations during communications

Other Optimization techniques

- Constant (beta, gamma) folding
- AiM-specific fusion for residual-cut
- Special instruction for "Value" vector append using masked DRAM write c ommand (for Multi-head Attention).
- Refresh during Communication

System Analysis

- Proof-of-Concept: AiM-centric Accelerator Prototype
- Performance Analysis and Estimate
- System Deployment Options

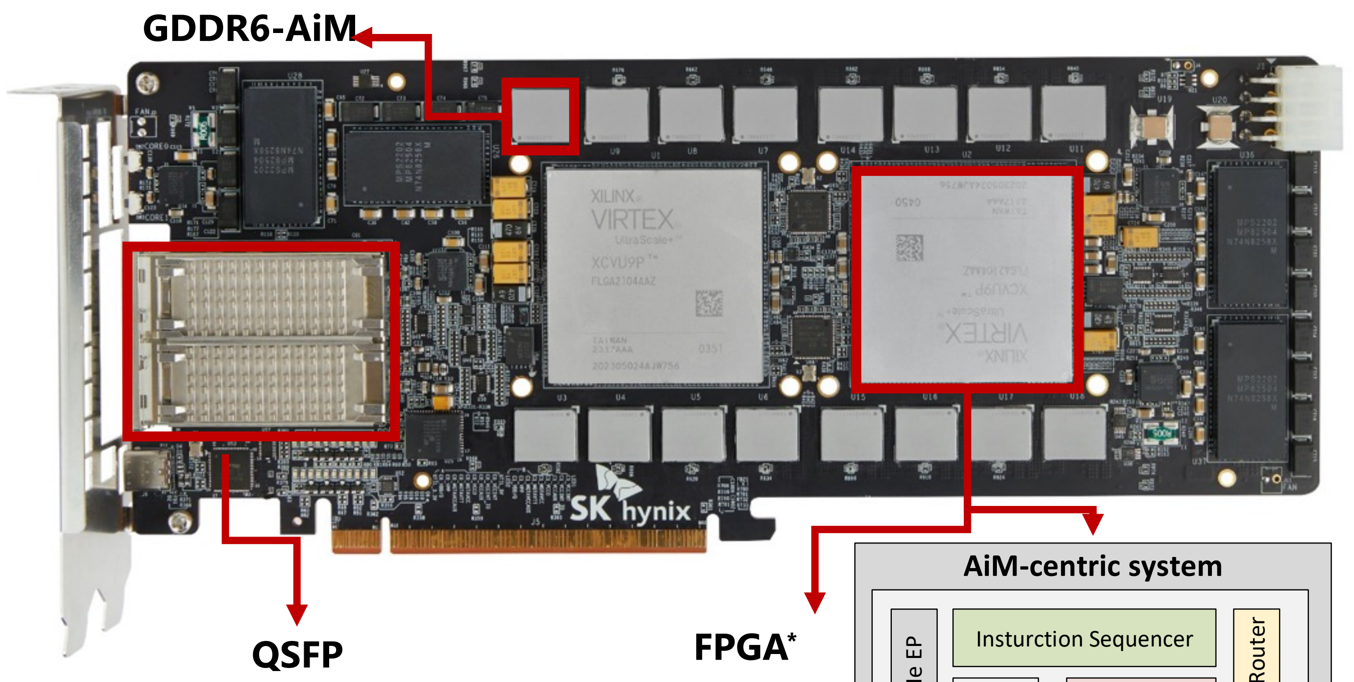


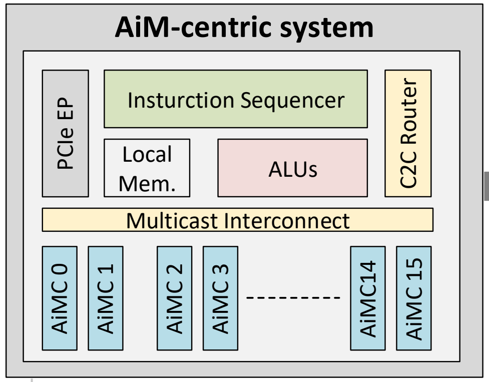
Proof-of-Concept for Scale-out AiM System

Scale-out AiM realization for proof-of-concept

to demonstrate and analyze end-to-end performance, power, and scalability

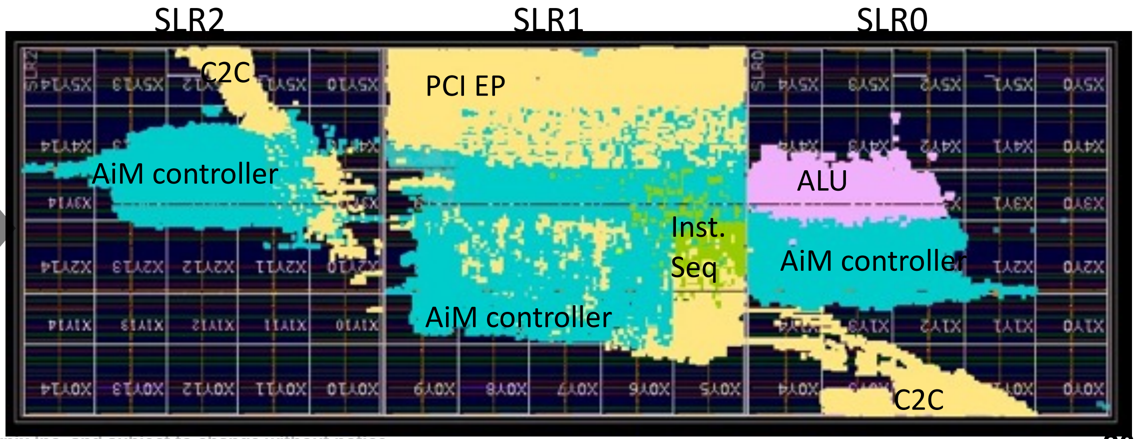
AiM-centric accelerator prototype





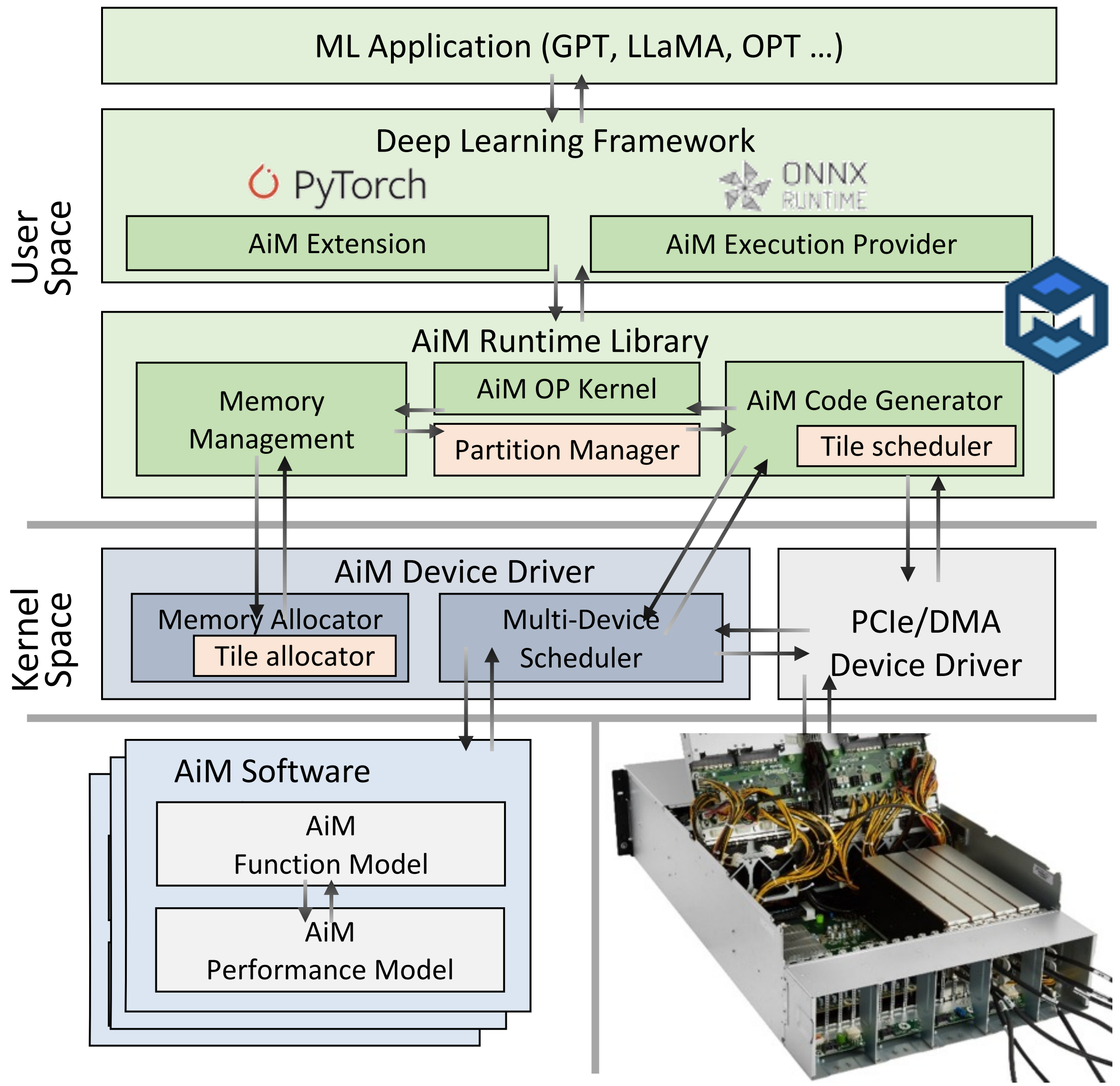
Specification

Host Interface		PCIe Gen3 x8x8 (bifurcated)
Form Factor		FHFL (A100/A30 compatible)
Configuration		2 FPGA* x 16 AiM package
AiM	**Capacity**	16 GB
	Bandwidth	170 GB/s (@2.67Gbps**)
Scale out		chip2chip interconnect (QSFP28)
Thermal Cooling		Passive

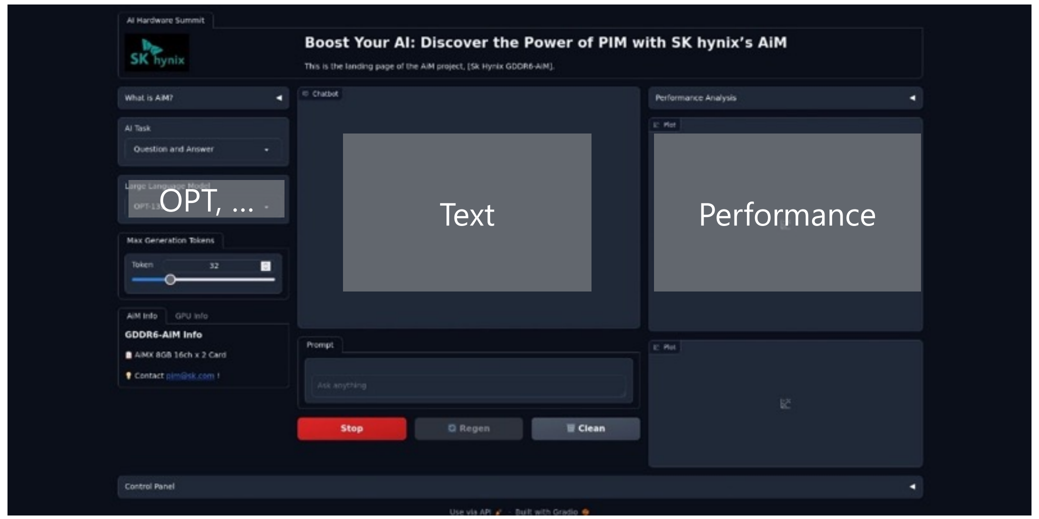


(*) Xilinx Virtex Ultrascale+ (VU9P)
(**) 1/6 of peak data rate of GDDR6, 6Gbps (or 1TB/s)

AiM Software Stack for Scale-out AiM System



LLM inference (Text Generation) Demonstration GUI


AI Hardware Summit

Boost Your AI: Discover the Power of PIM with SK hynix's AiM

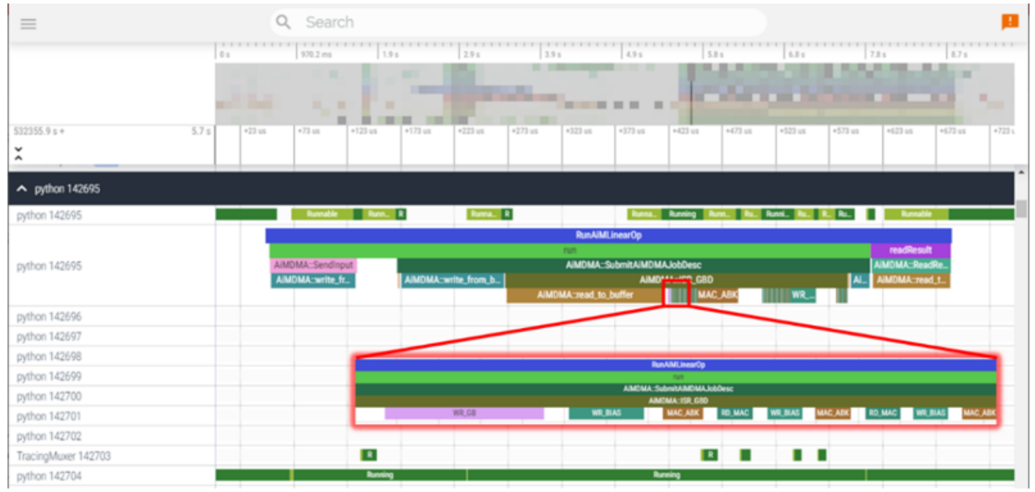
This is the landing page of the AiM project, [Sk Hynix GDDR6-AiM].

GDDR6-AiM Info

AiMX 8GB 16ch x 2 Card

Contact pim@sk.com !

Performance Profiler

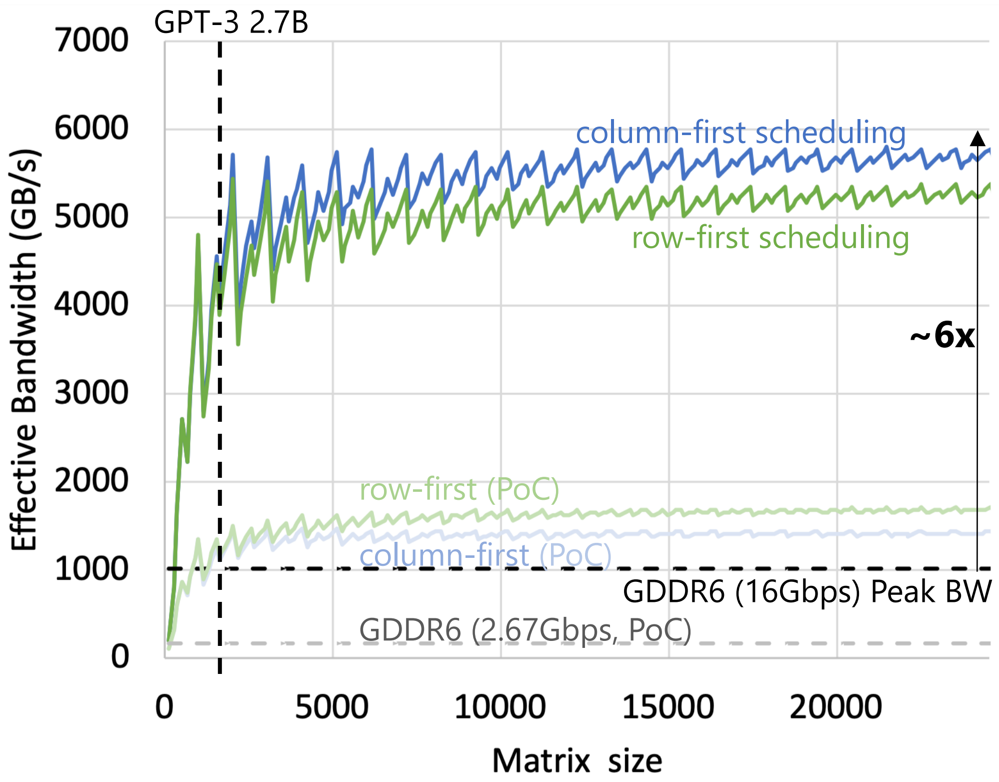


+ Memory/Instruction Debugger, …

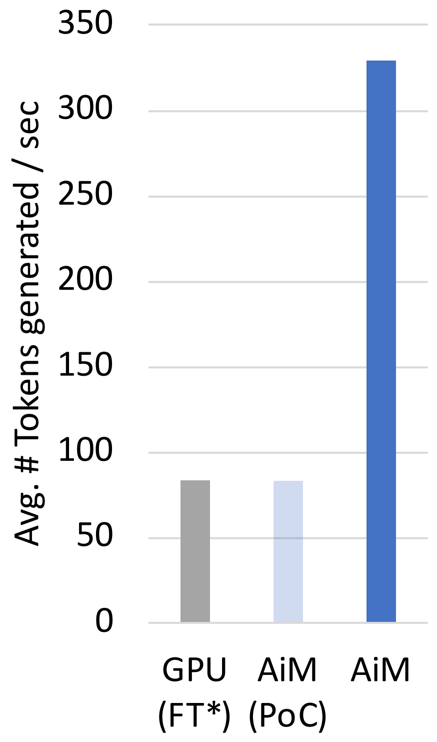
Performance Analysis

Effective bandwidth of GDDR6-AiM (@16Gbps) during GEMV expected to approach 6TB/s (6x baseline GDDR6 peak bandwidth) or higher with optimized tiling strategies

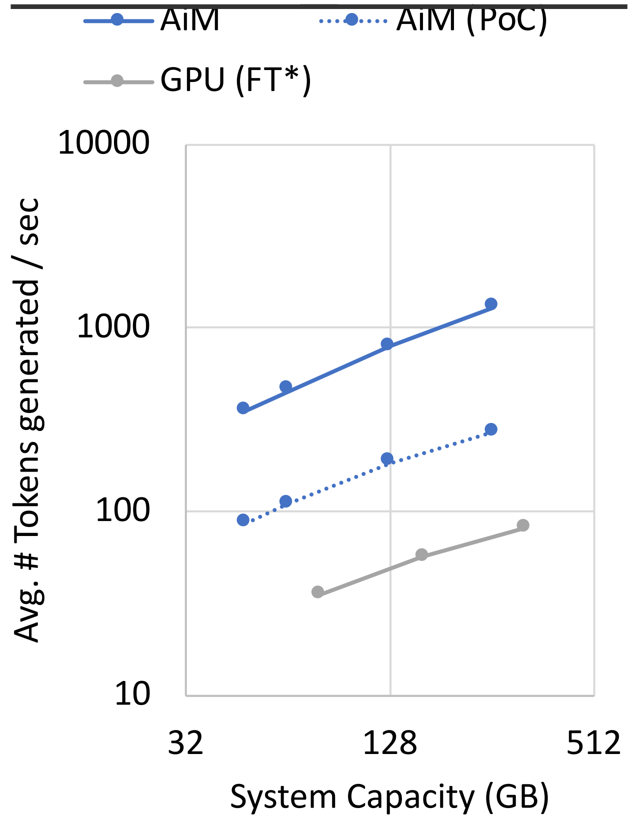
Bandwidth Achieved (GEMV)



Single Card (GPT-3 6.7B)



Multi-card Scaling (20B)

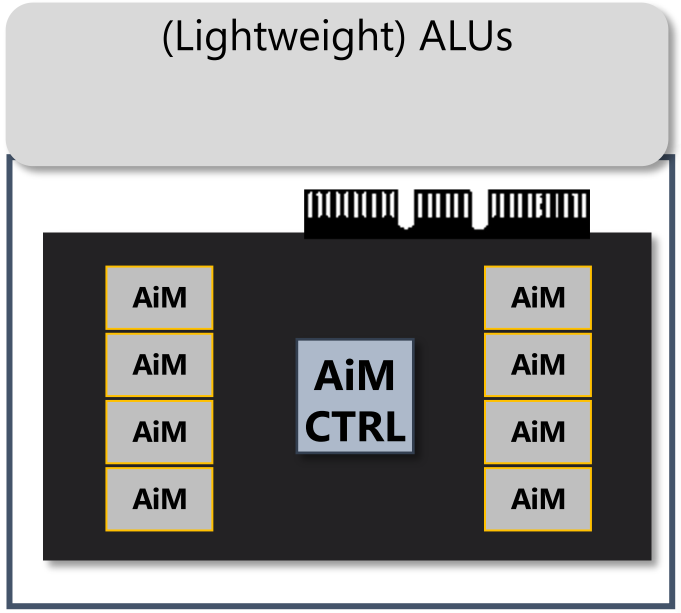


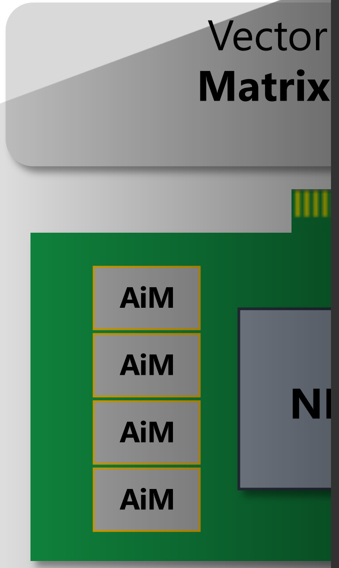
Note: Performance analysis of AiM (@16Gbps) is based on AiM cycle-accurate performance model, verified with performance measurements of AiM PoC (@2 ~ 2.67Gbps).

(FT) FasterTransformer: A100 GPU (SXM), batch1 performance*

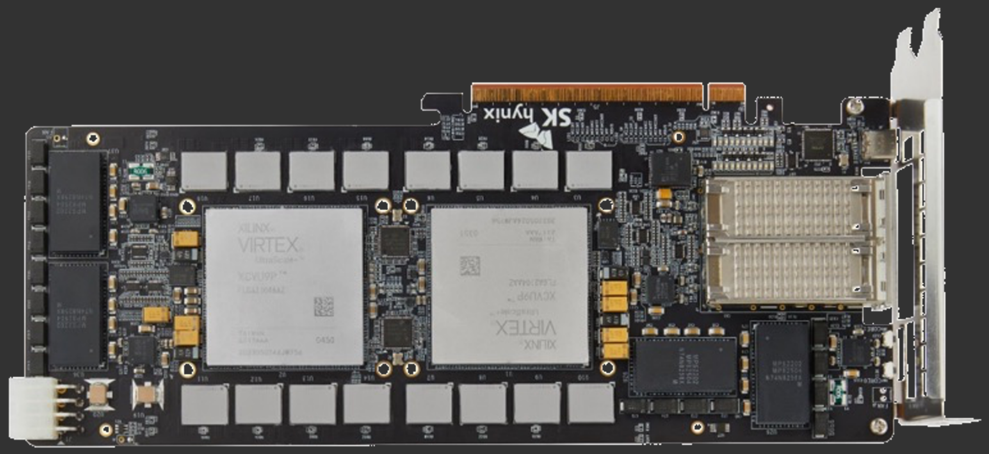
System Integration Options

“ Possibilities are ENDLESS “





Domain Specific for LLM inference
High Performance
Cost & Energy Efficient

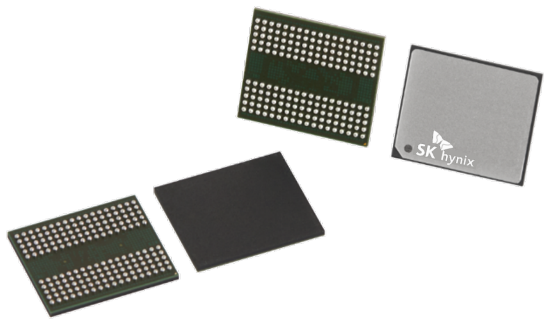


SK hynix
AiM-centric Accelerator Card

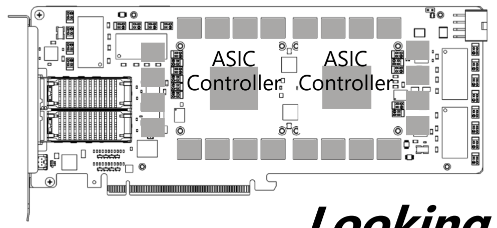
What's Next?

AiM 1st generation: GDDR6-AiM for High Performance Solution

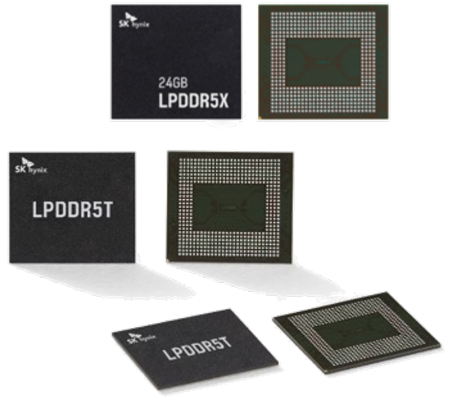
Exploring AiM next gen. for High Capacity Solution for Even Larger Models



Hypothetical yet feasible layout with ASIC controllers



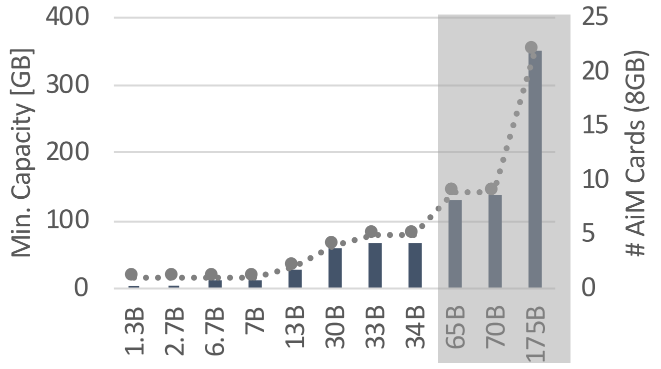
Looking forward to your proposals!



Currently under

- Architecture exploration
- Performance analysis
- Power/thermal analysis
- Cost analysis
- …

Anticipating your inputs



SK hynix NEWSROOM

SK hynix Starts Mass Production of Industry's First 24GB LPDDR5X DRAM, Aug. 11, 2023
SK hynix's LPDDR5T Verified as World's Fastest Mobile DRAM Using MediaTek's Next-Gen Mobile Platform, Aug. 10, 2023
SK hynix Develops World's Fastest Mobile DRAM LPDDR5T, Jan. 24, 2023

SKhynix_PIM@skhynix.com

Thank You