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

• 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

Generative AI services

“Inference Costs Eclipses Training Costs Over Time”

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

Cost*** ($)

Chat  Code  Translation  Search  Q&A

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

(**) TCO (Total Cost of Ownership) ~ CapEx + 3 * OpEX

(*** Inference cost ~ #users, assumed to be growing over months

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

(**) 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

Transformer Architecture

# parameters (per layer)

Input Embedding + Positional Encoding

Feed-Forward

Multi-head Attention

$2.7B$ $6.7B$ $13B$ $175B$

LLM model size

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

© SK hynix Inc. This material is proprietary of SK hynix Inc. and subject to change without notice.
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.

- High capacity
- Cost efficient

Managed DRAM Solution**

AiM: “Move compute, not Data”

- High Bandwidth
- Cost Efficient
- Energy Efficient

(*) CME: Capacity Memory Expansion
(**) [ISSCC ‘19] A 512GB 1.1V Managed DRAM Solution with 16GB ODP and Media Controller

$ / GB

Bandwidth

DDR5
LPDDR5
GDDR6
HBM3

Computing Memory
Mobile Memory
Graphics, Al (Inference)
Al (Training), HPC

Managed DRAM Solution**

CXL-CME*

Generative AI, LLMs
Accelerator-in-Memory (AiM)

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

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

(*) [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 by 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.

MAC\[0\] \* V[0] + MAC\[1\] \* V[1] + ... + MAC\[15\] \* V[15]

- 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.

Global Buffer

Weight Matrix

Bias

Activation Vector

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.
## AiM-specific Memory Commands

All-bank operations via single command for high compute efficiency

<table>
<thead>
<tr>
<th>Multi-Bank Activation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>4, 16 banks</td>
</tr>
<tr>
<td>ACTAF</td>
<td>4, 16 banks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-Bank Compute</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>1, 4, 16 banks</td>
</tr>
<tr>
<td>AF</td>
<td>16 banks</td>
</tr>
<tr>
<td>EWMUL</td>
<td>1, 4 bank groups</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Transfer</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RDCP</td>
<td>Global Buffer</td>
</tr>
<tr>
<td>WRCP</td>
<td>Global Buffer</td>
</tr>
<tr>
<td>WRGB*</td>
<td>Global Buffer</td>
</tr>
<tr>
<td>RDMAC*</td>
<td>MAC REG</td>
</tr>
<tr>
<td>WRMAC*</td>
<td>MAC REG</td>
</tr>
<tr>
<td>RDAF*</td>
<td>AF REG</td>
</tr>
<tr>
<td>WRBK</td>
<td>16 banks</td>
</tr>
</tbody>
</table>

(*) 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

Model configurations from GPT-3, LLaMA, and OPT

(*) Assumed 8*AiM packages per each AiM card
(**) Capacity required to store model weights in 16b precision, excluding Key and Value histories
Scale-out AiM for Large Language Models

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

\[ N \times \#\text{AiM packages} = N \times 2 \times \#\text{AiM channels} = N \times 2 \times 16 \times \#\text{PUs (or #Banks)} \]

Scale-out AiM

- (4 AiM system, 64 AiM channels, 32GB, 1024 PUs)
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
- Output Dimension: unit of parallelism (bank/channel)

**Row-first tile scheduling**

- Tile 0, Tile 1, Tile 2, Tile 3
- Tile 4, Tile 5, Tile 6, Tile 7
- Tile 8, Tile 9, Tile 10, Tile 11
- Tile 12, Tile 13, Tile 14, Tile 15

**Column-first tile scheduling**

- Tile 0, Tile 1, Tile 2, Tile 3
- Tile 4, Tile 5, Tile 6, Tile 7
- Tile 8, Tile 9, Tile 10, Tile 11
- Tile 12, Tile 13, Tile 14, Tile 15

**Column-first (using 2* MAC_REGS)**

- Tile 0, Tile 1, Tile 2, Tile 3
- Tile 4, Tile 5, Tile 6, Tile 7
- Tile 8, Tile 9, Tile 10, Tile 11
- Tile 12, Tile 13, Tile 14, Tile 15
## (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
• **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**

**Other Optimization techniques**
- Constant (beta, gamma) folding
- AiM-specific fusion for residual-cut
- Special instruction for “Value” vector append using masked DRAM write command (for Multi-head Attention).
- Refresh during Communication

*Scheduling All channel/bank AiM refresh operations during communications*
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

- **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**) (**) 1/6 of peak data rate of GDDR6, 6Gbps (or 1TB/s)
- **Scale out**: chip2chip interconnect (QSFP28)
- **Thermal Cooling**: Passive

(*) Xilinx Virtex Ultrascale + (VU9P)

---

AiM-centric system

- PCIe EP
- Instuction Sequencer
- Local Mem.
- ALUs
- Multicast Interconnect
- C2C Router

---

© SK hynix Inc. This material is proprietary of SK hynix Inc. and subject to change without notice.
AiM Software Stack for Scale-out AiM System

ML Application (GPT, LLaMA, OPT ...)

Deep Learning Framework

PyTorch

AiM Extension

AiM Execution Provider

User Space

AiM Runtime Library

Memory Management

AiM OP Kernel

Partition Manager

AiM Code Generator

Tile scheduler

Kernel Space

AiM Device Driver

Memory Allocator

Tile allocator

Multi-Device Scheduler

PCle/DMA Device Driver

Performance Profiler

Performance Model

Software Emulator

Function Simulator

Function Model

Memory/Instruction Debugger, ...

LLM inference (Text Generation) Demonstration GUI

Text

Performance

Text Performance

OPT, ...

+ 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

Currently under

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

Anticipating your inputs

Looking forward to your proposals!

SKhynix_PIM@skhynix.com

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
Thank You