2025 Design, Automation & Test in Europe Conference (DATE 2025)

DEAR-PIM: Processing-in-Memory Architecture with Disaggregated Execution of All-bank Requests

Jungi Hyun, Minseok Seo, Seongho Jeong, Hyuk-Jae Lee and Xuan Truong Nguyen
Seoul National University
Inter-University Semiconductor Research Center (ISRC)
Email: {hjn0202, sms0121, sh_jeong, hyuk_jae_lee, truongnx}@capp.snu.ac.kr

***Abstract*—Emerging transformer-based large language models (LLMs) involve many low-arithmetic intensity operations, which result in sub-optimal performance on general-purpose CPUs and GPUs. Processing-in-Memory (PIM) has shown promise in enhancing performance by reducing data movement bottlenecks. Commodity near-bank PIMs enable in-memory computation through bank-level compute units and typically rely on all-bank commands, which simultaneously operate the compute units of all banks to maximize internal bandwidth and parallelism. However, activating all banks simultaneously before issuing all-bank commands generally requires high peak power, which may exceed system power limit, when stacking multiple PIM devices for LLM inference. Additionally, under a DRAM power constraint, all-bank commands are only issued after all banks are fully activated through a sequence of single-bank activations, incurring bubble cycles and degrading overall performance.**

To address these shortcomings, this study proposes DEAR-PIM, a novel PIM architecture with *Disaggregated Execution of All-bank Requests*. DEAR-PIM incorporates *disaggregated command queue*, allowing it to buffer all-bank commands and provide them to each bank sequentially without waiting to complete all-bank activations. However, since all banks must finish their disaggregated execution before simultaneous post-processing, synchronization between early-activated and last-activated banks is necessary. To tackle the issue, DEAR-PIM introduces a *column-aware synchronization command* scheme that inserts no-op-like commands into unused columns without modifying the memory controller. Experiments demonstrate that DEAR-PIM achieves a speedup of 2.03-3.33× over an A100 GPU and improves performance by 1.11-1.52× compared to the sequential activation scheme. DEAR-PIM also reduces the peak power consumption by 21.3-41.7% compared to the simultaneous activation scheme.

***Index Terms*—Processing-in-Memory, DRAM Architecture, PIM-DRAM Commands**

I. Introduction

Transformer-based large language models (LLMs) have been showing their superior accuracy and adopted across a wide range of applications such as text summarization and text generation [3]–[5]. LLM inference, however, generally shows sub-optimal performance on general-purpose CPUs and GPUs, especially with a single batch [6]. One of the main challenges to accelerate LLM inference in a single batch is the existence of many low-arithmetic-intensity operations, e.g., general matrix-vector multiplication (GEMV) and element-wise vector operations (Vector Op). For example, fully-connected (FC) layers, which consist of GEMV operations, take a dominant portion of total latency, i.e., 58% to 81%, for various LLM models such as OPT [4] and Llama-3 [5] on a modern A100 GPU [19], as shown in Fig. 1. Therefore, accelerating memory-intensive operations in LLMs is one of the essential tasks to achieve high performance for LLM inference.

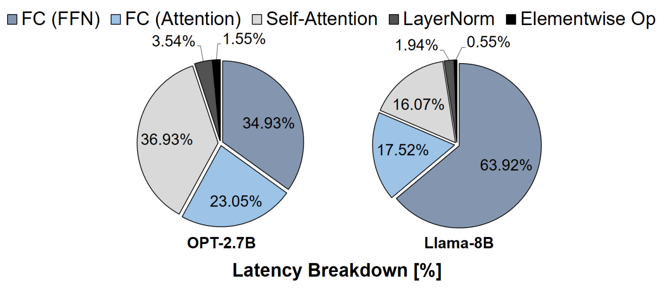


Fig. 1: GPU latency breakdown of LLM inference

Processing-in-Memory (PIM), an effective solution for memory-intensive operations, has recently emerged in modern LLM accelerators [6], [7]. Most PIMs [8], [9], [13] integrate compute units near DRAM banks to perform simple arithmetic operations near memory, enhancing performance and energy efficiency by reducing the massive data movement between a host and a memory. To accommodate large LLM models, modern hardware systems stack multiple PIM devices to increase capacity and computing power. Specifically, commodity near-bank PIMs [8], [9] typically incorporate all-bank commands, which make computing units operate simultaneously to fully utilize internal bandwidth. Before performing an all-bank computation, activation operations to all banks in a channel (so-called *all-bank activations*) are required first to access bank-stored data. However, activating all banks simultaneously (so-called *simultenous activation*) generally incurs a high peak power that is 30-76% higher than that of activating banks sequentially (so-called *seuquential activation*). As a result, simultaneous activation may become less practical for power-constrained DRAM systems.

Sequential activation in PIMs, on the other hand, comes with a DRAM timing issue in bank activations. Specifically, most DRAMs rely on sequential bank activation to comply with power-related timing constraints, such as the Four Activation Window (tFAW) and Row-to-Row Delay (tRRD) [2], [16]. Under these constraints, it incorporates a sequence of single-bank activations to activate all banks, which takes considerable latency. More importantly, existing PIM architectures only enable all-bank commands after the completion of all-bank activations, leading to unnecessary bubble cycles and thereby degrading overall performance.

To address these shortcomings, this paper proposes **DEAR-PIM**, a novel PIM architecture with ***D****isaggregated* ***E****xecution of* ***A****ll-bank* ***R****equests*, with the following contributions.

- **Methodology:** DEAR-PIM introduces a simple yet effective concept of disaggregated execution of all-bank requests. Unlike aforementioned schemes, our method enables early execution of a bank request before the completion of all-bank activations under a power constraint. Consequently, it relieves a peak power, reduces unnecessary bubble cycles, and enhances overall performance.
- **Microarchitecture and Software Stack:** DEAR-PIM incorporates *disaggregated command queue* that buffers all-bank commands and provides them to each bank with a different delay. However, since all banks must finish their execution before simultaneous post-processing, synchronization between all banks must be ensured. To address this issue, DEAR-PIM introduces a *column-aware synchronization command* scheme that inserts no-op-like commands into unused columns without modifying the memory controller.
- **Implementation and Evaluation:** DEAR-PIM is implemented on the top of HBM-PIM [8] and evaluated on a microbenchmark and LLM models. Experiments demonstrate that DEAR-PIM achieves a speedup of 2.03-3.33× over an A100 GPU [19] and improves performance by 1.11-1.52× compared to the sequential activation scheme. DEAR-PIM also reduces the peak power by 21.3-41.7% compared to the simultaneous activation scheme.

II. Background

A. Memory-Intensive Operations and PIMs

Deep learning frameworks generally make use of many APIs built on the basic linear algebra subprograms (BLAS). Level-3 BLAS (matrix-matrix operations) allows for significant data reuse within the on-chip cache, leading to many operations per byte. As a result, its performance is primarily governed by the system's computational capabilities. However, Level-1 BLAS (scalar-vector and vector-vector operations) and Level-2 BLAS (matrix-vector operations) - fundamental operations in modern LLMs [3]–[5] - have low arithmetic intensity, which generally requires frequent off-chip memory accesses to perform a few computations. PIMs [8], [9], [13] place compute units in memory, which enable performing arithmetic operations near DRAM banks, avoiding off-chip memory accesses, and thereby enhancing performance and energy efficiency.

B. DRAM/PIM Organization and Commands

DRAM consists of multiple banks, each organized into a cell array where individual cells store binary data (0 or 1) in capacitors. Before accessing DRAM cells, it requires *bank activation* that opens a specific row within a bank and buffers the data for access. Bank activation is widely known to be

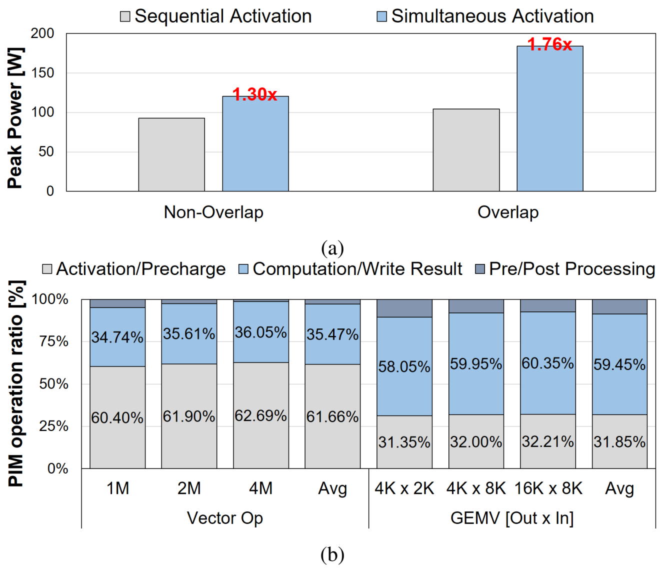


Fig. 2: PIM profiling results (a) Peak power requirements by sequential and simultenous activation schemes, (b) Latency breakdown of PIM operations in sequential activation scheme

power-intensive due to DRAM cell charging. Meanwhile, because DRAM uses destructive read operations—where reading the data disrupts its stored state—a precharge operation is needed to restore the data, preparing the DRAM for the next activation. Activation and precharge are essential for reliable data access and maintenance in DRAM, and *sequential bank activation* is favorable and used frequently in DRAMs due to the power-intensive nature of activation.

Built on the top of DRAMs, PIMs [8], [9], [13] place compute units close to the DRAM bank array, using a local data bus to retrieve data from nearby banks. Given a command from a host, most PIMs [8], [9] use an all-bank command, a PIM command to enable compute units on all banks in a channel. To perform an all-bank command, it requires all banks to be activated beforehand (so-called *all-bank activations*). Under a peak power limit, PIMs/DRAMs often rely on sequential bank activations to comply with power-related timing constraints, such as tFAW and tRRD [2], [14], [16].

III. Motivation

A. Peak Power Issue in All-bank Activations

Emerging PIM-based accelerators for LLMs [6], [7] stack multiple PIM devices to store large model parameters. To fully utilize their in-memory bandwidth and computing power, it generally requires to use all-bank commands. Unfortunately, activating all banks, a must-do task before all-bank commands, may require a considerable peak power. Fig. 2a reports the peak power of two all-bank activation schemes: (1) *simultaneous activation*, and (2) *sequential activation*. In the figure, 'Overlap'[1] refers to an option that allows the next bank-side activation to overlap with the current bank-side computation, enhancing performance and energy efficiency.

[1]This overlap option can be utilized in specific PIM architectures where a compute unit is shared by multiple banks, such as Samsung's HBM-PIM [8].

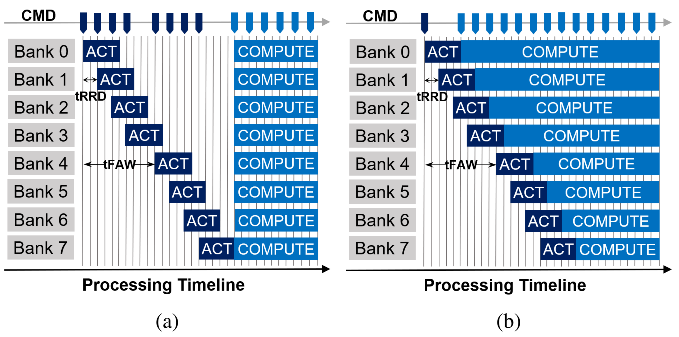


Fig. 3: PIM execution scheme (a) Sequential activation, (b) DEAR-PIM

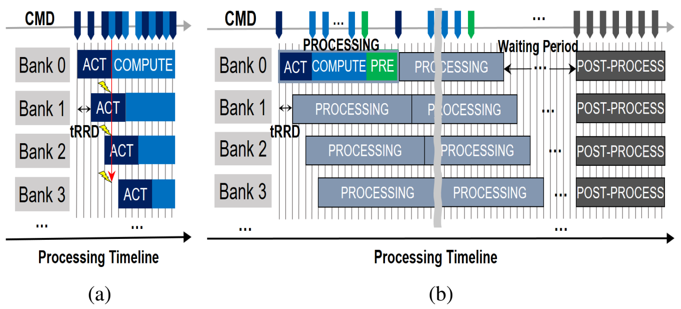


Fig. 4: DEAR-PIM design challenges (a) All-bank computation before the completion of sequential activations, (b) Waiting period required between the last bank and others for synchronized execution

However, this option can lead to higher peak power, as it involves activating and utilizing multiple bank sides simultaneously. It is observed that simultaneous activation may increase the peak power consumption by 30-76% compared to sequential activation, showing simultaneous activation worsens peak power especially in the overlap case. This comparison suggests a strong message that simultaneous activation may become less practical for power-constrained DRAM systems.

B. Performance/Timing Issues with All-bank Activations

Performance. Under a DRAM power constraint, sequential activation becomes more favorable and frequently used for all-bank activations. Unfortunately, sequential activation generally incurs a large performance degradation in PIMs. Fig. 2b reports the latency breakdown of PIM operations for a microbenchmark with various GEMV and Vector Op (See Table I for detailed settings). The profiling results show that activation and precharge operations account for an average of 31.85% of execution time in GEMV and 61.66% in Vector Op. It suggests that a long latency by sequential activation is likely to degrade performance, emphasizing the need for a new execution scheme to minimize this overhead.

Timing. The root-cause of the performance bottleneck in sequential activation scheme is a DRAM timing issue in bank activations. Following the conventional DRAM specifications [2], [16], banks are activated sequentially with power-related timing constraints. Then, all-bank commands are invoked to maximize parallelism of computing units, as illustrated in Fig. 3a. In this case, although the first bank is activated earlier, the requests on this bank must be delayed until the completion of the last bank's activation. Consequently, this scheme incurs considerable bubble cycles, and thereby degrades performance. This finding suggests an opportunity to gain performance by avoiding these bubble cycles.

IV. DEAR-PIM Design

A. Design Methodology

This section presents DEAR-PIM, a PIM architecture with Disaggregated Execution of All-bank Requests. In DEAR-PIM's execution scheme, as shown in Fig. 3b, each bank independently executes all-bank commands after power-related timing constraints. In this way, unlike the sequential activation, DEAR-PIM removes the need to wait for other banks to activate, minimizing performance degradation while adhering to power constraints.

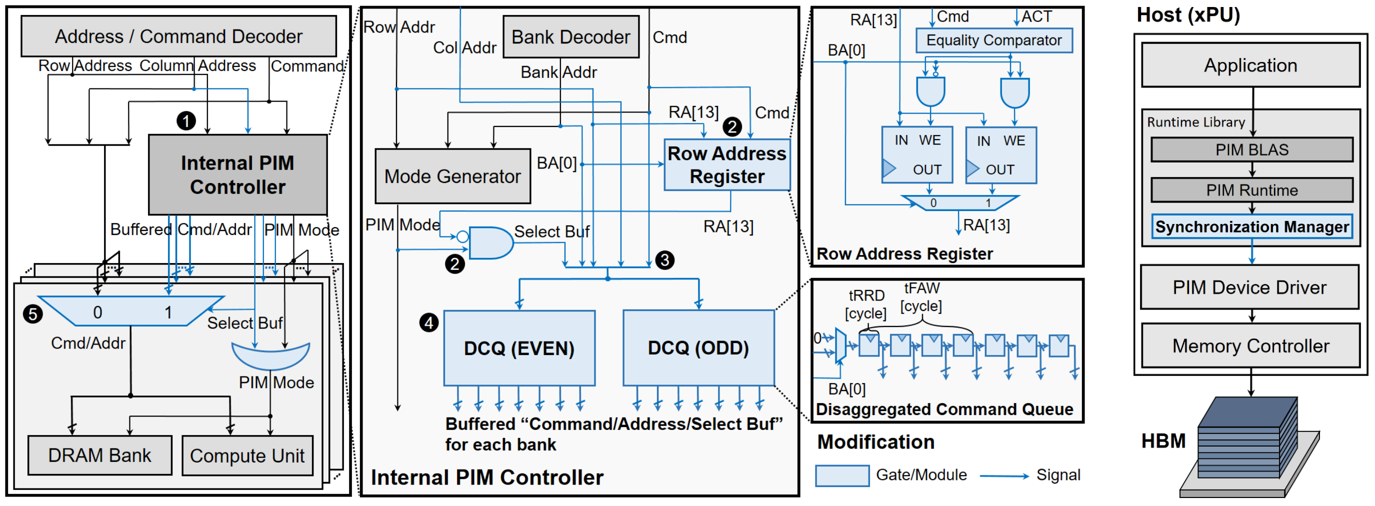
Despite these advantages, disaggregated execution scheme presents two new challenges. The first challenge is to ensure a proper execution at each bank with all-bank commands. Given a shared command bus within the same channel, an all-bank command from a host affects to all banks in the channel. Therefore, issuing an all-bank command before the sequential activation completes, as illustrated in Fig. 4a, can result in some banks receiving the command before their activation finishes or even starts, causing unexpected behavior in the bank array. To solve this challenge, DEAR-PIM introduces new *disaggregated command queue* (DCQ) that buffers all-bank commands and provides them to each bank with a different delay, enabling an early execution for each bank without the completion of all-bank activations as shown in Fig. 3b.

The second challenge is synchronization between the first and last banks at the end of processing. Typically, PIM requires post-processing tasks, such as setting up a next input in a compute unit's register or changing PIM modes [8], [10], once processing is complete. Since all banks must finish their computation before post-processing begins, it must ensure timing synchronization between all banks, as shown in Fig. 4b. To address this issue, DEAR-PIM introduces a *column-aware synchronization command* (CSC) that prevents other commands to be scheduled during this period.

The following Sections IV-B and IV-C present the detailed architectures and a modification in the PIM software stack to address the aforementioned challenges.

B. Microarchitecture

To implement DCQ for the disaggregated execution scheme, two key design considerations must be addressed. The first is that the same all-bank command should be issued to each bank with a different delay. While per-bank DCQs can introduce unique delays for each bank, they limit the potential of hardware reuse across banks. As a result, we opt for a per-channel DCQ, a shared DCQ that can provide buffered commands after a predefined cycle delay for each bank. The second consideration is the need for a mechanism to deliver buffered commands to each bank array. Since the host sends all-bank commands via a shared command bus, a multiplexer is required to select whether the host command or the buffered



(a) (b)

Fig. 5: DEAR-PIM design (a) Microarchitecture, (b) PIM software stack

command is propagated to the bank array. Therefore, an select signal for the buffered command is needed to indicate whether it is an all-bank command, allowing the multiplexer to select and deliver the command to the bank array.

In this part, we describe the implementation of the microarchitecture as shown in Fig. 5a. We integrate it into the HBM-PIM architecture [8], [10], where each channel has an internal PIM controller responsible for managing mode changes during all-bank execution. We utilize this space to integrate two per-channel DCQs, one for each bank side, as two banks (EVEN and ODD) share a compute unit in HBM-PIM. Before buffering the command/address in the DCQ, an select signal (i.e., 'Select Buf' signal in Fig. 5a) is required to verify whether the current command is an all-bank command targeting the real bank array in all-bank mode. Given that HBM-PIM reduces the bank array size by half [10], the 13th bit of the row address (RA[13]) is set to 0 for all-bank commands aimed at the bank array. Therefore, the combination of PIM mode and RA[13] signal can be used to generate the select signal. However, since the RA[13] bit is only available with the activation command, a 2-bit row address register (1 bit per bank side) is added to generate the select signal for subsequent commands. This signal is then sent to the per-bank multiplexer to select between the host command/address and the buffered command/address. This approach ensures that the buffered command/address is delivered to the bank array only during all-bank command execution. The overall flow of DEAR-PIM operates as follows.

The decoded command/address is first sent to the internal PIM controller (❶), where the PIM mode signal from the existing mode generator and the RA[13] signal from the row address register generate the multiplexer's select signal (❷). Next, the command/address and select signal are sent together to the DCQ module (EVEN or ODD) (❸). Within the DCQ, the command/address is propagated for different amounts of time for each bank, allowing individual delays to be applied (❹). Finally, the buffered command/address, along with the

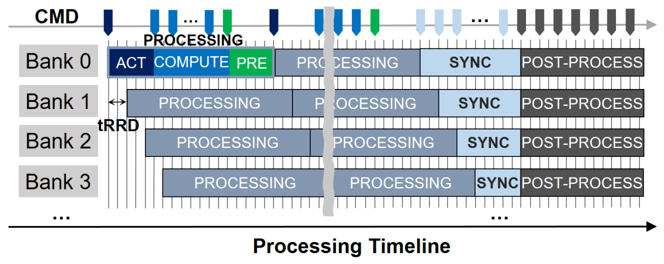


Fig. 6: Synchronization at the end of processing using column-aware synchronization commands.

buffered select signal, is passed to a multiplexer, which sends the buffered command/address to the bank array when the command is an all-bank command (❺).

C. PIM Software Stack

Although the microarchitecture of DEAR-PIM supports the disaggregated execution scheme, a challenge remains. Since the host memory controller issues all-bank commands for the first-activated bank, it may issue post-processing commands while the buffered all-bank commands are still being executed on later-activated banks. As these post-processing commands need to be executed simultaneously due to their potential impact on bank behavior (e.g., mode change), synchronization is required before post-processing to ensure correct execution.

To address this without altering the memory controller, we leverage DRAM commands for synchronization. However, general DRAM commands are unsuitable for this purpose, as they trigger read/write operations that could lead to unexpected behavior. We observed that some columns with specific rows remain unused in HBM-PIM, particularly rows reserved for post-processing, due to the removal of half of the cell array [10]. This discovery led us to develop a column-aware synchronization command (CSC): a read command with an unused address that acts as a no-op. By inserting a sufficient number of CSCs, we can prevent post-processing commands

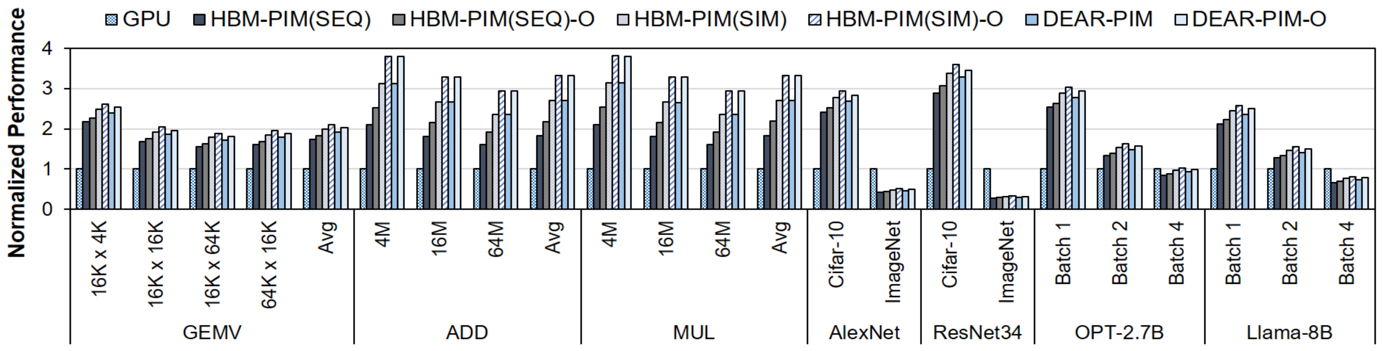


Fig. 7: Performance evaluation result. Note that 'O' refers to the option of permitting the overlap of the computation and activation.

from being scheduled during the waiting period, as illustrated in Fig. 6.

These CSCs are inserted by a synchronization manager, integrated into the PIM software stack as shown in Fig. 5b. Once the PIM runtime generates the PIM execution kernel, which can be later translated into several DRAM commands, the manager calculates the required number of CSCs based on DRAM timing parameters and inserts them into the kernel before sending the commands to the PIM. Through this mechanism, DEAR-PIM can be integrated into the existing PIM software stack without requiring changes to the host memory controller.

V. Evaluation

A. Methodology

Methodology. We implemented DEAR-PIM on the top of HBM-PIM [8], [10] by modifying PIMSimulator [1], an open-source cycle-accurate Samsung HBM-PIM simulator. Following [8], [10], the memory system configurations are summarized in Table I. We use four PIM stacks with the total capacity of 24 GB that sufficiently accommodates the parameters of Llama-3 8B [5] with half-precision floating point. For performance evaluation, we compare DEAR-PIM with the NVIDIA A100 80GB GPU [19], HBM-PIM with sequential activation scheme and simultaneous activation scheme (i.e., HBM-PIM(SEQ) and HBM-PIM(SIM)). Notably, A100 accommodates HBM2e which is compatible with HBM2 in PIMSimulator [1]. For energy efficiency evaluation, we evaluate DEAR-PIM with the HBM-PIM with both schemes.

TABLE I: Memory system configuration

HBM-PIM x64 configuration	
# of bank per pCH	16 banks
# of pCH per die	4 pCHs
# of die per rank	4 dies
# of rank per stack	2 ranks
# of stack	4 stacks (24GB)
Timing parameters [tCK]	tRCD_RD=14, tRCD_WR=10, tFAW=16, tRRD_S=4, tRP=14, tCCD_L=4
Clock frequency	1.2GHz

Benchmarks. Similar to [8], [10], our experiments are conducted on a microbenchmark that includes various GEMV and Vector Op settings. Furthermore, we evaluate DEAR-PIM and its counterparts with well-known PIM offloadable operations (e.g., FC, CONV layers, and element-wise operations) in popular DNN models AlexNet [17] and ResNet [18] and modern LLMs OPT [4] and Llama-3 [5]. Following [6], [8], our evaluation mainly targets LLM inference with a single batch, considering latency-sensitive LLM serving scenario. We also report the results with a small batch, e.g., batch = 2 or 4, for better understanding.

Power. We apply parameters from previous works [2], [10], [20] to the Micron power model used in DRAMsim3 [15]. The model with this setting results in energy consumption of 2.51 pJ/bit, which closely aligns with the 2.75 pJ/bit reported in the HBM-PIM paper [10].

B. Performance/Energy Results

Fig. 7 presents the performance evaluation results for the GPU, DEAR-PIM, and HBM-PIM baselines. For GEMV, DEAR-PIM achieves an average speedup of 1.92× over the GPU and 1.11× over the sequential activation baseline, increasing to 2.03× and 1.12× with overlapping. Additionally, for Vector Op, it shows a 2.70-3.33× improvement over the GPU and 1.48-1.52× over the sequential baseline. Above result demonstrates that DEAR-PIM can reduce sequential activation latency through multiple disaggregated all-bank commands, thereby enhancing the performance of DEAR-PIM. Meanwhile, DEAR-PIM experiences a slight performance decrease of 3.4-4.3% for GEMV and less than 0.1% for Vector Op compared to the simultaneous activation baseline. This overhead primarily results from the synchronization required at the end of processing. Nevertheless, DEAR-PIM operates within DRAM power constraints, effectively managing peak power more evenly.

For CNN workloads, DEAR-PIM's performance decreases as image size increases, ranging from 2.84-3.45× speedup over the GPU for Cifar-10 to 0.32-0.49× performance for ImageNet. This result suggests that compute-bound convolutional layers with larger images are more efficiently executed on GPUs, as in a previous study [14]. In contrast, for LLM workloads, DEAR-PIM achieves 2.78-2.94× better performance for OPT-2.7B and 2.36-2.49× better performance for Llama-8B compared to the GPU in a single batch scenario. This improvement is primarily driven by PIM's ability to leverage

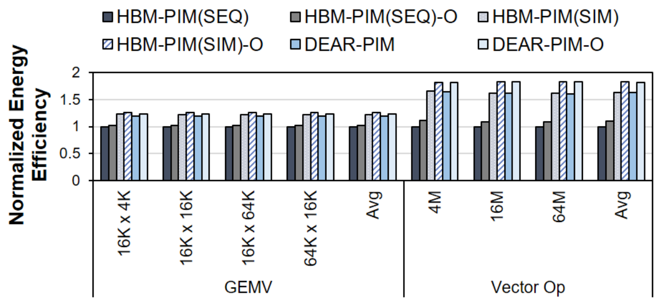


Fig. 8: Energy efficiency evaluation result. Note that 'O' refers to the option of permitting the overlap of the computation and activation.

higher internal memory bandwidth compared to the external memory bandwidth of GPU (i.e., 4.8 TB/s vs 2 TB/s) [2], [19]. However, as batch size increases, this advantage diminishes because GEMV operations become compute-intensive GEMM operations, and DEAR-PIM's peak throughput is much lower than that of the A100 GPU (i.e., 4.8 TFLOPS vs 312 TFLOPS) [10], [19]. Meanwhile, across all workloads, DEAR-PIM exhibits a performance trend similar to that observed in the microbenchmark when compared to HBM-PIM with both schemes.

To compare energy efficiency, we conducted experiments using a microbenchmark. The results, presented in Fig. 8, show that DEAR-PIM achieves 1.20-1.21× and 1.63-1.66× higher energy efficiency compared to the sequential activation baseline for GEMV and Vector Op, respectively. As energy efficiency is closely tied to execution time, DEAR-PIM exhibits slightly lower energy efficiency than the simultaneous activation baseline, with a decrease of 2.2-2.4% for GEMV and 0.2% for Vector Operations.

C. Peak Power Analysis

As shown in Table II, compared to the simultaneous activation baseline, DEAR-PIM reduces peak power by roughly 21.3-41.7%, depending on whether non-overlap or overlap options are considered. While DEAR-PIM slightly increases peak power compared to sequential activation baseline, the difference is minimal, about 2.8-3.8%. This result shows that DEAR-PIM can effectively minimize peak power overhead while offering better performance compared to sequential activation scheme.

TABLE II: Peak power result

	Non-Overlap	**Overlap**
Sequential Activation	92.62 W	104.43 W
Simultaneous Activation	120.58 W	184.01 W
DEAR-PIM	96.12 W	107.31 W

D. DEAR-PIM Implementation

To measure the area and power overhead of DEAR-PIM, we used Synopsys Design Compiler [12] with a 14nm library, scaling the results to 20nm fabrication, the technology used in Samsung's HBM-PIM. We also accounted for the differences in density between DRAM and logic fabrication, given that the DRAM process is 10× less dense than the logic process [13]. After scaling, our evaluation in Table III shows a total area overhead of 0.138 mm², which is less than 1% of the HBM-PIM die area (84.4 mm²) [8]. The power overhead is measured at 92 mW, which is also less than 1% of the HBM-PIM's average power consumption of 60.5 W.

TABLE III: DEAR-PIM area and power overhead

	Total Overhead	**Original HBM-PIM**
Area	0.138 mm²	84.4 mm²
Power	0.092 W	60.5 W

VI. Related Work

AiM [9], [11] is a commodity PIM that utilizes all 16 banks, each equipped with a compute unit. To support efficient use of all-bank commands, it enables simultaneous activation of 4 or 16 banks, but requires additional reservoir capacitors, occupying 26% of the area in each compute unit. There is a study [14] that introduces the Single Instruction Long Data (SILD) command to mitigate sequential activation overhead in all-bank command schemes. SILD command reads the data of entire columns with a single command, thereby minimizing command bus congestion. Another study [21] divides PIM execution into two phases: a memory phase that fetches data using per-bank commands and a computation phase that performs simultaneous computation across banks using all-bank commands. Our solution, DEAR-PIM, offers an architectural approach for the disaggregated execution scheme that supports all-bank commands under DRAM power constraints.

VII. Conclusion

In this work, we present DEAR-PIM, a novel architecture designed to overcome the limitations of previous PIM execution schemes, particularly the challenges associated with both simultaneous and sequential bank activation. By incorporating a disaggregated command queue within the PIM controller and a column-aware synchronization command for synchronization, DEAR-PIM can support disaggregated execution scheme for all-bank commands, effectively reducing peak power while enhancing performance.

Acknowledgement

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2023-00228255, PIM-NPU Based Processing System Software Developments for Hyper-scale Artificial Neural Network Processing, 50% / RS-2023-00256081, Artificial Intelligence Semiconductor Support Program to Nurture the Best Talents, 20%) and in part by the Technology Innovation Program (or Industrial Strategic Technology Development Program) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) (No. 20014490, Development of Technology for Commercializing Lv.4 Self-driving Computing Platform Based on Centralized Architecture, 30%). Xuan Truong Nguyen is the corresponding author.

References

[1] Samsung Advanced Institute of Technology, "PIMSimulator," https://github.com/SAITPublic/PIMSimulator.

[2] JEDEC, "High Bandwidth Memory (HBM) DRAM," JESD235, 2013.

[3] A. Radford *et al.*, "Language Models are Unsupervised Multitask Learners," *arXiv*, 2019.

[4] S. Zhang *et al.*, "OPT: Open Pre-trained Transformer Language Models," *arXiv*, 2022.

[5] A. Dubey *et al.*, "The Llama3 Herd of Models," *arXiv*, 2024.

[6] M. Seo *et al.*, "IANUS: Integrated Accelerator based on NPU-PIM Unified Memory System," in *ASPLOS*, 2024.

[7] J. Park *et al.*, "AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference," in *ASPLOS*, 2024.

[8] S. Lee *et al.*, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology," in *ISCA*, 2021.

[9] M. He *et al.*, "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning," in *MICRO*, 2020.

[10] Y. Kwon *et al.*, "A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications," in *ISSCC*, 2021.

[11] S. Lee *et al.*, "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," in *ISSCC*, 2022.

[12] Synopsys, "Design Compiler," https://www.synopsys.com.

[13] F. Devaux *et al.*, "The true Processing In Memory accelerator," *IEEE Hot Chips 31 Symposium (HCS)*, 2019.

[14] H. Kal *et al.*, "AESPA: Asynchronous Execution Scheme to Exploit Bank-Level Parallelism of Processing-in-Memory," in *MICRO*, 2023.

[15] S. Li *et al.*, "DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator," *IEEE CAL*, 2020.

[16] T. Zhang *et al.*, "Half-DRAM: A High-bandwidth and Low-power DRAM Architecture from the Rethinking of Fine-grained Activation," in *ISCA*, 2014.

[17] A. Krizhevsky *et al.*, "ImageNet Classification with Deep Convolutional Neural Networks," in *NeurIPS*, 2012.

[18] K. He *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[19] NVIDIA, "Nvidia A100 Tensor Core GPU," https://www.nvidia.com/en-us/data-center/a100.

[20] K. Chen *et al.*, "CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory," in *DATE*, 2012.

[21] Y. Paik *et al.*, "Achieving the Performance of All-Bank In-DRAM PIM With Standard Memory Interface: Memory-Computation Decoupling," *IEEE Access*, 2022.