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

DOTS: DRAM-PIM Optimization for Tall and Skinny GEMM Operations in LLM Inference

Gyeonghwan Park, Sanghyeok Han, Byungkuk Yoon, and Jae-Joon Kim
Electical and Computer Engineering
Seoul National University, Korea
[gyeonghwan.park, hansh778, bkyoon, kimjaejoon]@snu.ac.kr

***Abstract*—For large language models (LLMs), increasing token lengths require smaller batch sizes due to increase in memory requirement for KV caching, leading to under-utilization of processing units and memory bandwidth bottleneck in NPUs. To address the challenge, we propose DOTS, a new DRAM-PIM architecture that can handle both GEMV and GEMM efficiently, even outperforming NPUs in GEMM operations when batch sizes are small. The proposed DRAM-PIM reduces power consumption and latency caused by frequent DRAM row activation switching in conventional DRAM PIMs with negligible hardware overhead. Simulation results show that our proposed design achieves throughput improvements of 1.83x, 1.92x, and 1.7x over GPU, NPU, and heterogeneous NPU/PIM systems, respectively, for models as large as or larger than OPT-175B.**

I. Introduction

For transformer-based Large Language Models (LLMs), as model sizes and input/output token sizes increase, applying multi-batching becomes more challenging due to memory capacity limitations imposed by KV caching, which scales linearly with batch size. In these scenarios, GEMM operations involve the tall and skinny input matrix (Tall and Skinny GEMM; TS-GEMM) with small batch sizes, making them memory-intensive and leading to low utilization of processing units in NPUs.

To address the issues, we propose DOTS, a DRAM-PIM architecture for TS-GEMM. When the batch size is not large enough to efficiently utilize resources in NPUs, dedicating all decoding stages to PIM can be faster, leaving the NPU to handle only the prefill stages. However, existing PIMs [1]–[3] are primarily optimized for GEMV operations, which has led to the use of input stationary (IS) dataflow that maximizes input feature vector reuse. To improve efficiency in GEMM operations, we adopt a weight stationary (WS) approach to take advantage of the weight reuse chance and minimize the number of row activations, a power- and time-consuming operation in DRAM.

However, adopting the WS approach in conventional DRAM-PIM architectures introduces new challenges that are not encountered in IS dataflow. First, updating the input feature matrix before each MAC operation leads to performance degradation. Second, the accumulated results need to be extracted from the result latch after each MAC operation, causing significant read-write transition overhead. To overcome these challenges, we propose a double-buffering technique for the global buffer and the use of multiple result latches with minimal overhead.

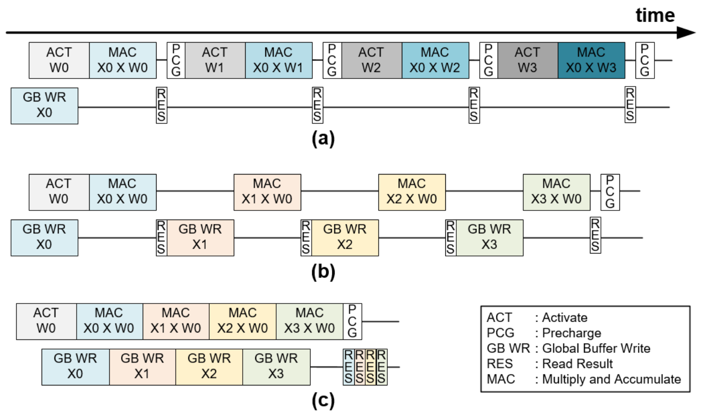

Fig. 1: Comparison of timing diagrams for (a) IS dataflow in conventional DRAM-PIMs, WS dataflows in (b) conventional DRAM-PIMs, and (c) the proposed DOTS.

II. DOTS Architecture

Preference for weight stationary dataflow Typically, existing PIM systems employ Input Stationary (IS) dataflow to maximize input vector reuse, because GEMV operations have no weight reuse opportunity. However, Using IS dataflow for GEMM operations in DRAM-PIM misses every weight reuse chance and thus leads to large DRAM activation overhead. As shown in Fig. 1a, DRAM row activation overhead for weight loading degrades the GEMM performance in IS dataflow. We propose using a Weight Stationary (WS) dataflow for GEMM operations in DRAM-PIM instead of IS dataflow. WS dataflow minimizes row switching and increases PE utilization in PIM by maximizing weight reuse.

Challenges in WS dataflow implementation Although WS dataflow involves lower activation overhead as shown in Fig. 1b, it introduces new issues. First, bringing different input vectors in every computation requires extra overhead for writing to the global buffer(GB). Second, outputs for different input vectors are not automatically accumulated, requiring the results to be read out after each computation. Especially, the interleaving of reads and writes on data bus introduces additional latency to avoid bus conflict. To address these challenges, we propose a new DRAM-PIM architecture with minimal hardware modifications from conventional DRAM-PIM. By eliminating these overheads, we expect a performance enhancement, as shown in Fig. 1c. Our design implements *double buffering in GB* to overlap GB writing

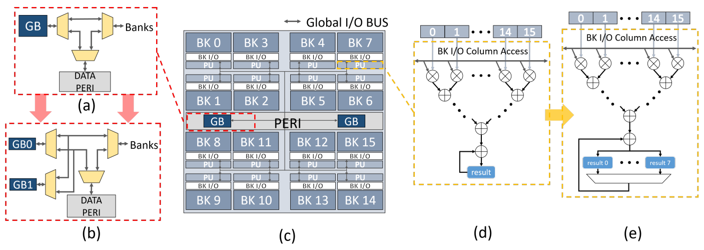

Fig. 2: (a) Baseline GB and I/O interconnect. (b) Proposed GB with double-buffering. (c) DRAM-PIM organization (d) Baseline MAC unit. (e) Proposed MAC unit with multiple result latches.

with MAC operations, and introduces *multiple result latches* to read out multiple outputs simultaneously.

Double buffering in GB When applying the WS dataflow in PIM, the GB must be filled at the start of each loop. In conventional DRAM-PIM architectures (Fig. 2a), both writing to the GB from the DATA PERI block for external I/O and feeding the input vector to the processing units (PU) in DRAM-PIM bank share the same GB, preventing the overlap of these two operations. Additionally, while GB writes can be performed during row activation in IS dataflow, WS does not require activating another row for the next PIM tile operation, making it impossible to overlap both operations. This issue can be resolved by implementing double buffering in the GB and separating the I/O bus for read and write operations, as shown in Fig. 2b. Instead of adding extra buffers, the memory is divided into two sets: a read buffer and a write buffer, which alternate roles. While the read buffer transfers data to the banks, the write buffer receives data from the DATA PERI, and then the roles switch, effectively hiding the latency of GB writes.

Multiple result latches The WS dataflow in conventional DRAM PIM leads to frequent read operations, since different output features are computed without being accumulated. This limitation arises from having only one result latch per bank, as illustrated in Fig. 2d. The result read path uses the global I/O bus, meaning that other operations, such as computation or GB writing, cannot be executed simultaneously. Additionally, before reading from the result latch, the entire adder tree pipeline must be cleared, which introduces additional stall cycles. To address this, we propose incorporating multiple result latches per PU in each bank, reducing the need to frequently offload results to the processor for subsequent MAC operations.

III. Evaluation

Performance Fig. 3 presents a runtime and throughput simulation results of the decode stage across various model sizes and input/output token sizes. We configure our system as a heterogeneous architecture, combining state-of-the-art NPUs, TPUv4i [4] with DOTS memory devices and compared DOTS system against GPU, TPU, NPU-PIM heterogeneous system [5] and conventional PIM-only system.

For models like the 66B, where batch sizes are relatively large, using PIM becomes less effective due to the compute-

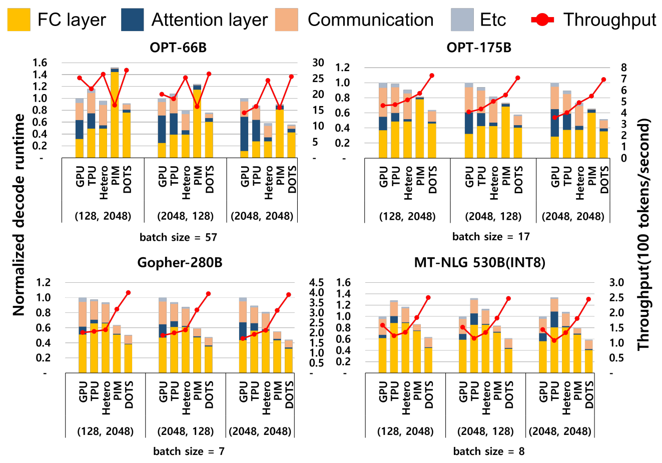

Fig. 3: Decode stage runtime and throughput comparison for OPT models with varying model sizes and token counts. Runtime (normalized to GPU) is shown on the left y-axis, while throughput is displayed on the right y-axis. Batch sizes for each model were selected to prevent out-of-memory (OOM) errors.

intensive fully connected (FC) layers. However, as model sizes increase, the batch size needs to be reduced due to the limitation in memory capacity. The proposed DOTS architecture consistently outperforms all other systems across all model sizes. It efficiently addresses challenges posed by large models and small batch sizes by leveraging PIM effectively, coupled with architectural optimizations such as double buffering and multiple result latches. We observe performance improvements of 1.75x, 1.64x, 1.42x, and 1.27x over GPU, TPU, heterogeneous, and PIM-only systems, respectively, for the OPT-175B model. For models larger than 175B, the improvements are 1.87x, 2.06x, 1.84x, and 1.36x, respectively.

Acknowledgement: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00208606, NeuroHub+: Scheduler and Simulator for General In-Memory Neural Network Accelerators) (Corresponding Author: Jae-Joon Kim).

References

[1] M. He *et al.*, “Newton: A dram-maker’s accelerator-in-memory (aim) architecture for machine learning,” in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 372–385.

[2] 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 *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[3] S. Lee *et al.*, “Hardware architecture and software stack for pim based on commercial dram technology : Industrial product,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 43–56.

[4] N. P. Jouppi *et al.*, “Ten lessons from three generations shaped google’s tpuv4i : Industrial product,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 1–14.

[5] J. Park *et al.*, “Attacc! unleashing the power of pim for batched transformer-based generative model inference,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2024, pp. 103–119.