Design Automation Conference 2025 (DAC 2025)

Bank-Split PIM: Enabling Concurrent PIM and Memory Operations for LLM Inference in Heterogeneous Systems

Hyoengjun Cho, Yoonho Jang, Seokin Hong
Sungkyunkwan University
{cho0624ck, jyh8807, seokin}@skku.edu

SUNGKYUNKWAN UNIVERSITY(SKKU)

COMPASSLAB
SKKU Computer Architecture & Systems Laboratory

Background & Motivation

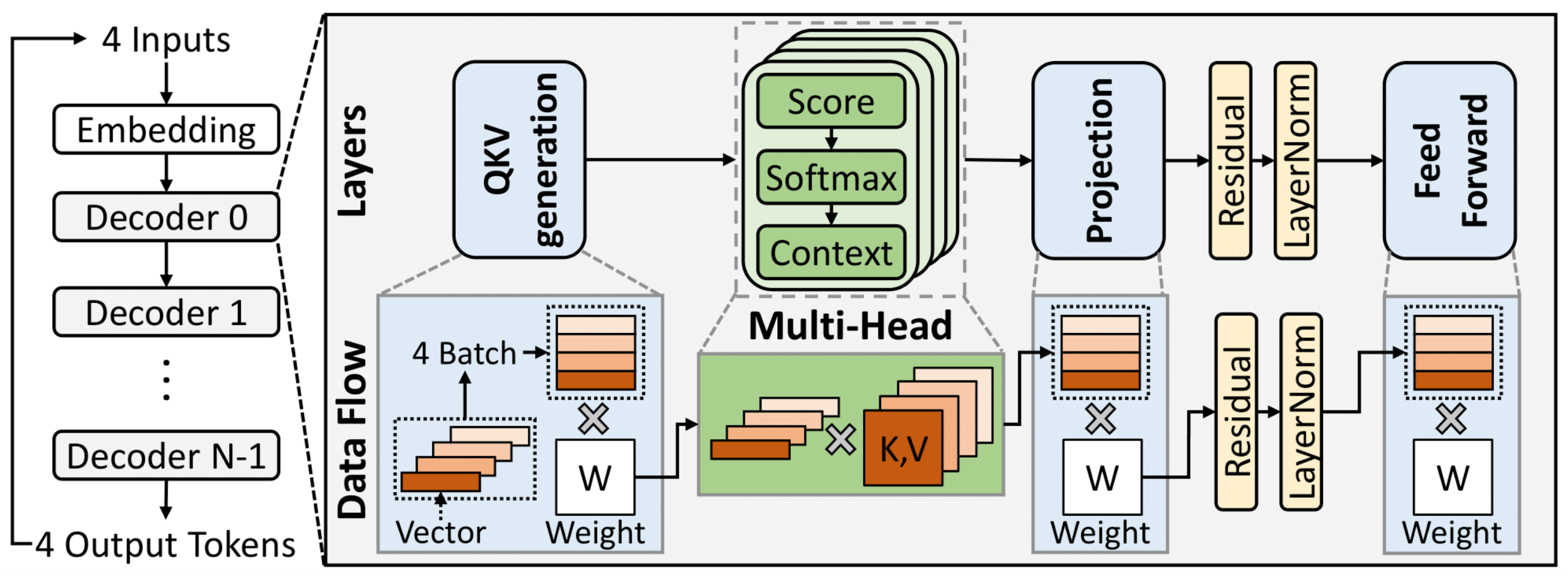
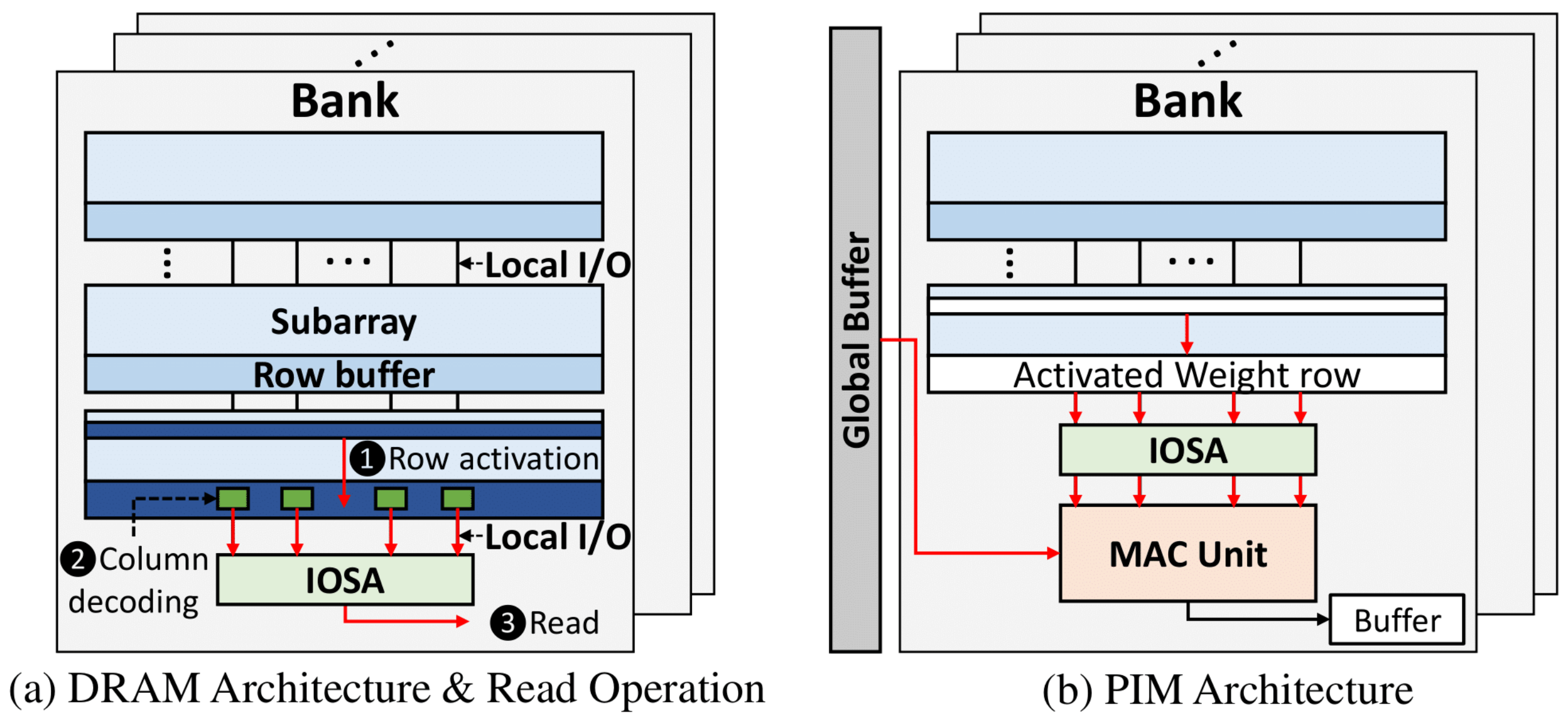


Figure 1. LLM batched inference

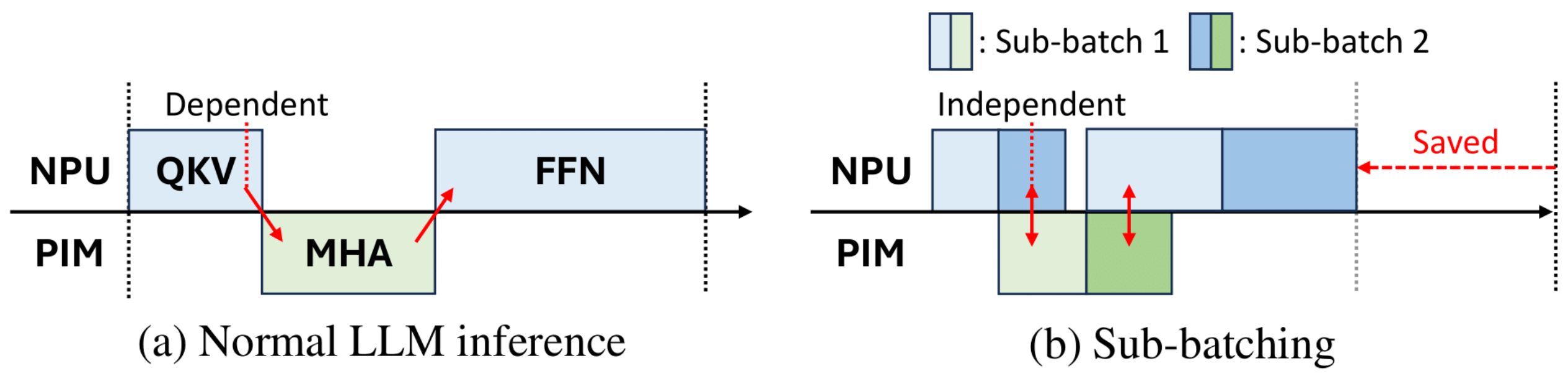
- **LLM inference in Heterogeneous system**

As depicted in Figure 1, batched Large Language Model (LLM) inference comprises both compute-intensive General Matrix Multiplication (GEMM) operations (blue blocks) and memory-intensive General Matrix-Vector (GEMV) operations (green blocks). PIM-PU Heterogeneous systems, integrating a NPU or GPU for efficient GEMM and PIM for high internal bandwidth to handle memory-intensive workloads, can effectively accelerate LLM inference.



(a) DRAM Architecture & Read Operation (b) PIM Architecture

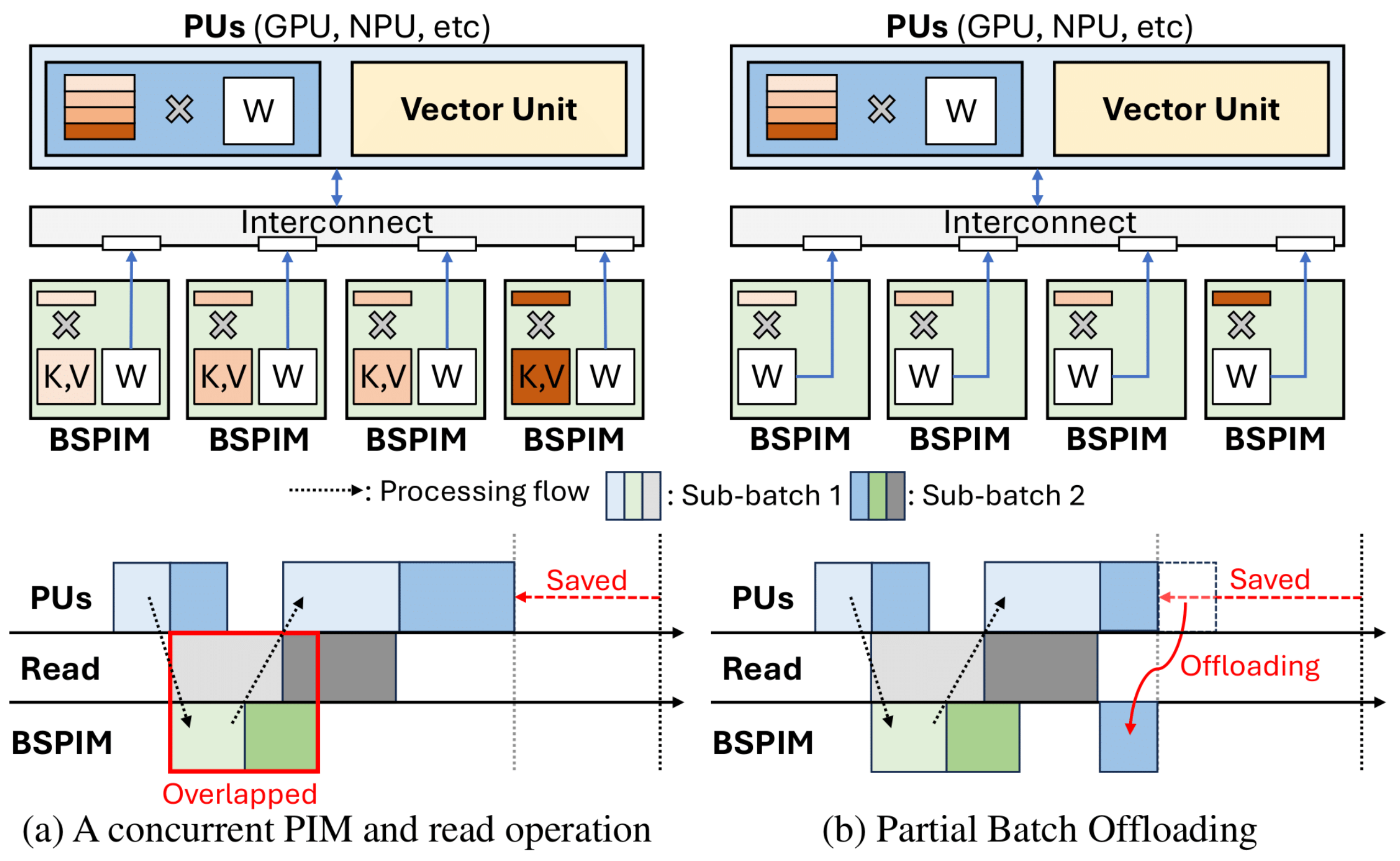
Figure 2. DRAM and PIM architecture



(a) Normal LLM inference (b) Sub-batching

Figure 3. LLM batched inference flow in a heterogeneous system

As illustrated in Figure 2, both DRAM read operations and PIM's MAC operations share bank resources, including local I/O wires, row buffers, and sense amplifiers. This resource contention prevents them from operating concurrently. Such a limitation poses a significant constraint on recently proposed techniques like sub-batching for concurrent execution (as shown in Figure 3(b)), where resource conflicts permit the simultaneous operation of the PU and PIM.



(a) A concurrent PIM and read operation (b) Partial Batch Offloading

Figure 4. Overview of LLM process and timeline in BSPIM

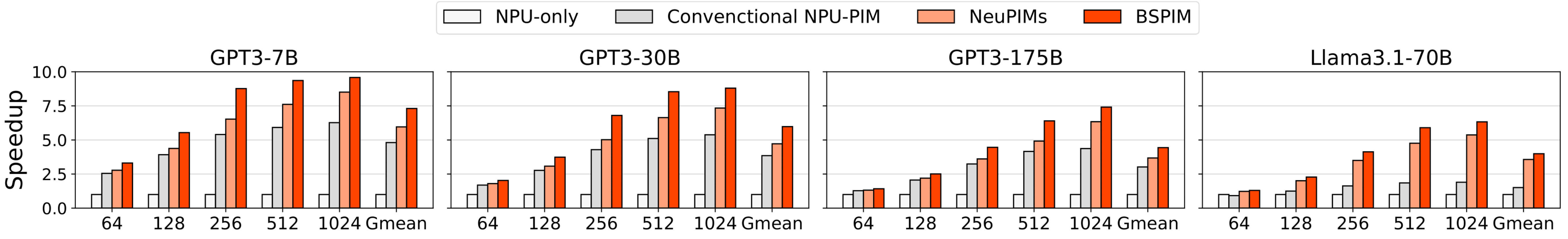


Figure 7. Normalized performance of NPU-only, conventional NPU-PIM, NeuPIMs and BSPIM with batch sizes of 64, 128, 256, 512 and 1024. Evaluated with various Models: GPT3-7B, GPT3-30B, GPT3-175B, Llama3.1-70B

Bank-Split PIM

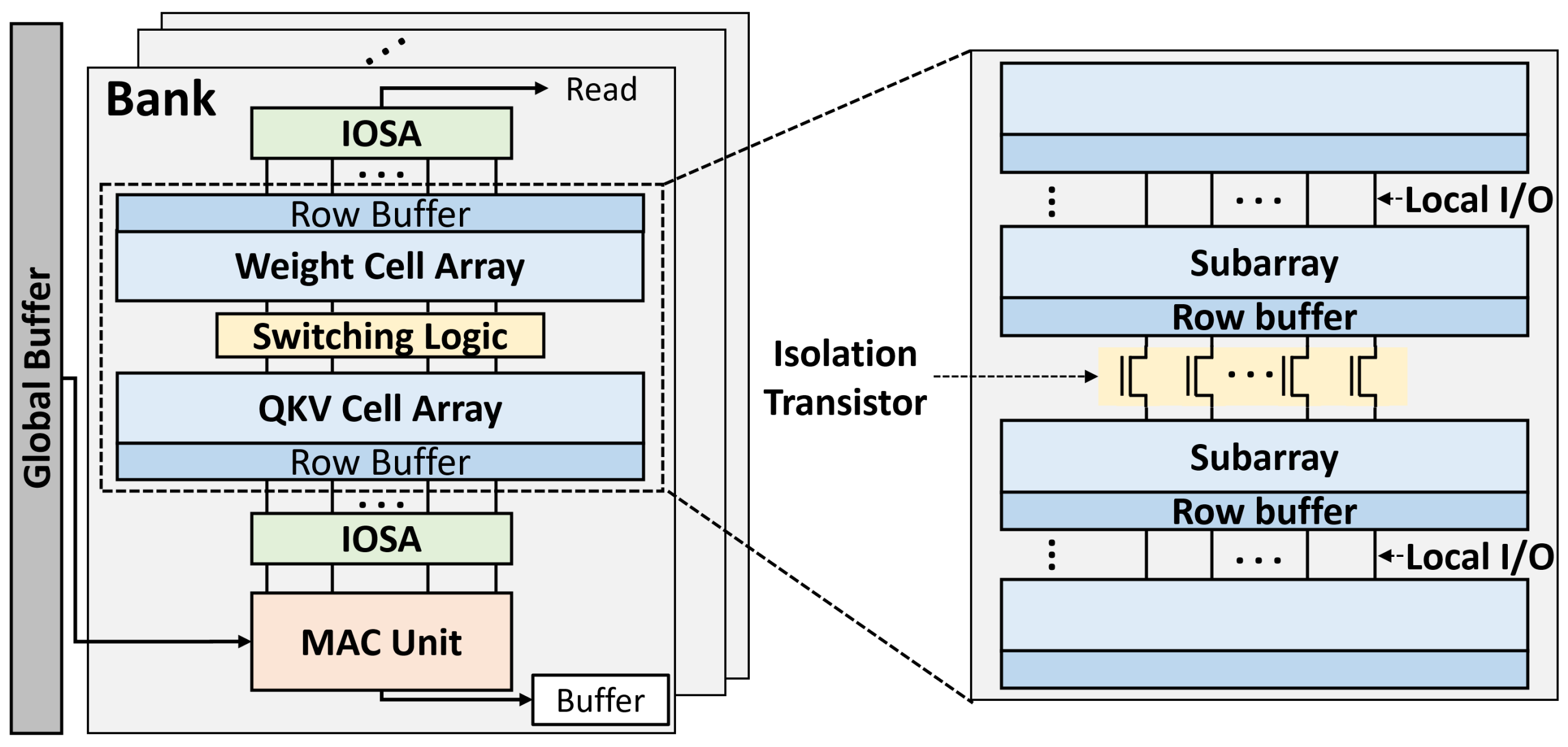
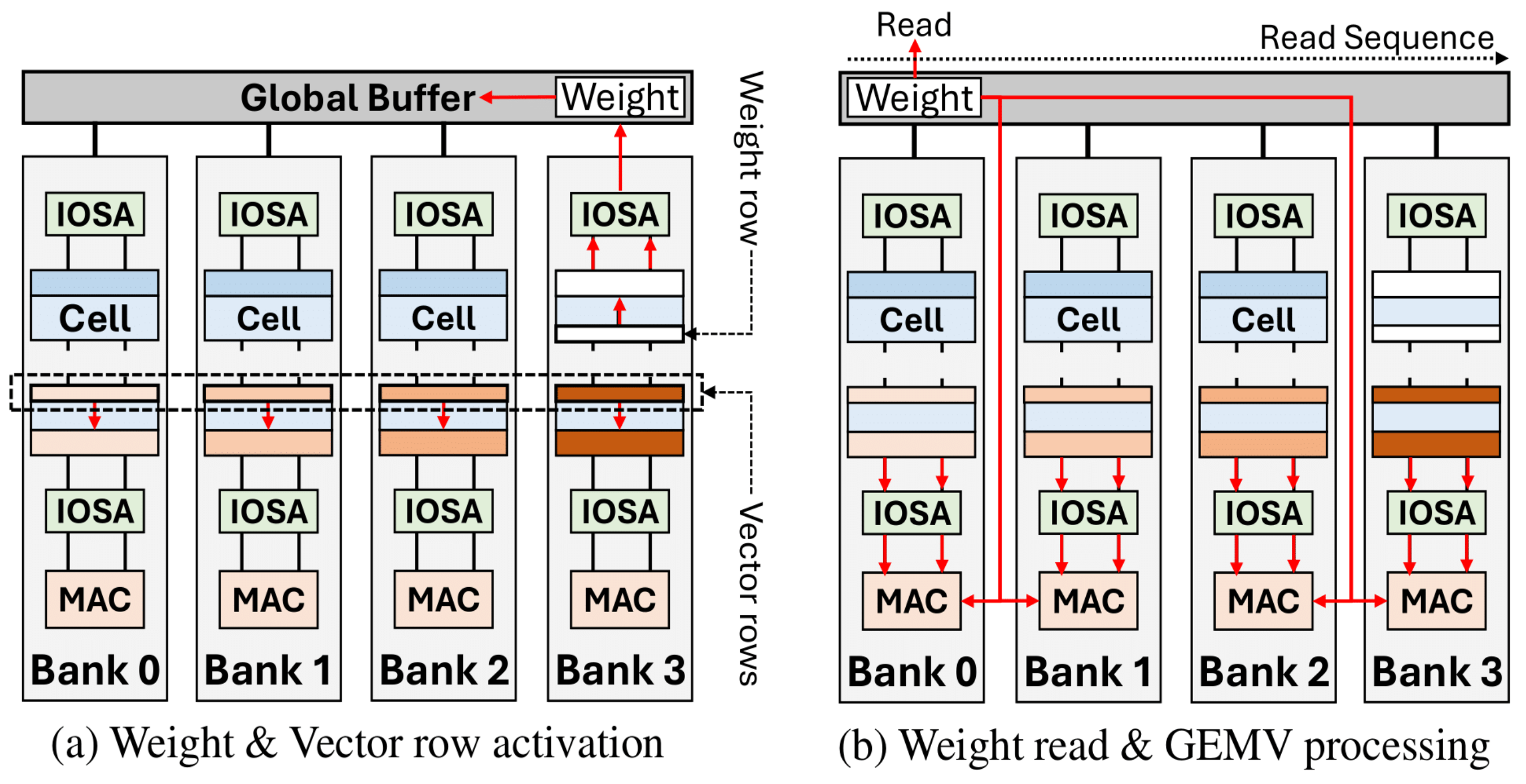


Figure 5. A Bank Architecture of BSPIM

- **Bank-Split PIM (BSPIM) Architecture**

The Bank-Split PIM (BSPIM) architecture addresses the resource conflict by splitting a single DRAM bank into two independent partitions, each with its own dedicated Local I/O wires. Each partition has own I/O Sense Amplifier, enabling simultaneous operation shown in Figure 4(a): the PIM partition can perform MAC operations while the DRAM partition concurrently handles read requested by the host.

In Figure 5, BSPIM allocates QKV data to be stored in the PIM partition (below the switching logic) and processed by the MAC unit. Concurrently, the DRAM partition (above the switching logic) can read necessary weights for the PU.



(a) Weight & Vector row activation (b) Weight read & GEMV processing

Figure 6. PBO operation

- **Partial Batch Offloading (PBO)**

To maximize PIM utilization and balance workloads in heterogeneous systems, BSPIM implements Partial Batch Offloading (PBO). When PIM is idle during GEMM-heavy workloads, it offloads operations by concurrently receiving broadcast weights from the PU and processing them. Conversely, if the PU is idle during GEMV-heavy workloads, PIM simultaneously performs MAC operations and transfers QKV data to the PU for MHA execution. BSPIM runtime scheduler dynamically profiles PU and PIM latencies to determine optimal offload batch sizes, ensuring balanced execution throughout inference.

Experimental Result

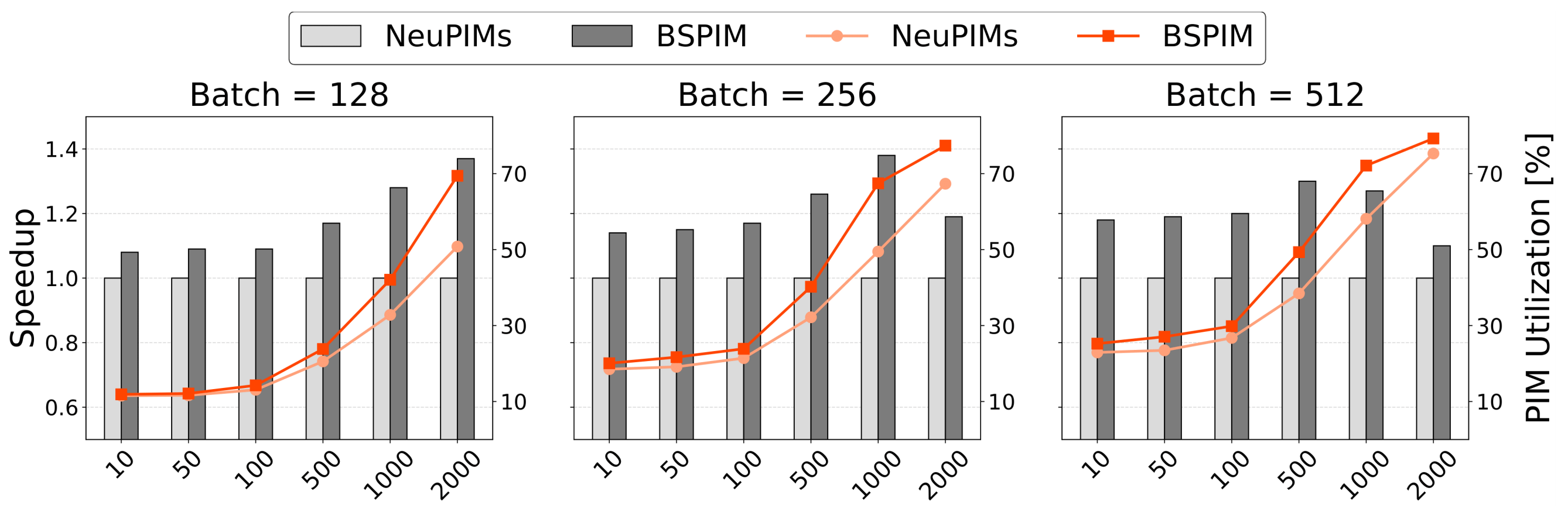


Figure 8. Normalized performance and PIM utilization following different input sequence lengths

- **Performance and Utilization**

Figure 7 compares normalized speedups of BSPIM against an NPU-only baseline, a conventional NPU–PIM design, and the NeuPIMs state-of-the-art across GPT3 and Llama3.1 model. BSPIM delivers up to 7.3× acceleration on GPT3-7B and consistently outperforms NeuPIMs by around 20% on average across all batch sizes and model scales.

Figure 8 evaluates the impact of input sequence length on throughput and PIM utilization across three batch sizes (128, 256, 512). As sequence length increases, both BSPIM and NeuPIMs demonstrate performance gains. However, BSPIM consistently achieves substantially higher PIM utilization from under 20% at short sequences to over 70% at length 2000 which attributed to its ability to overlap read and compute operations.

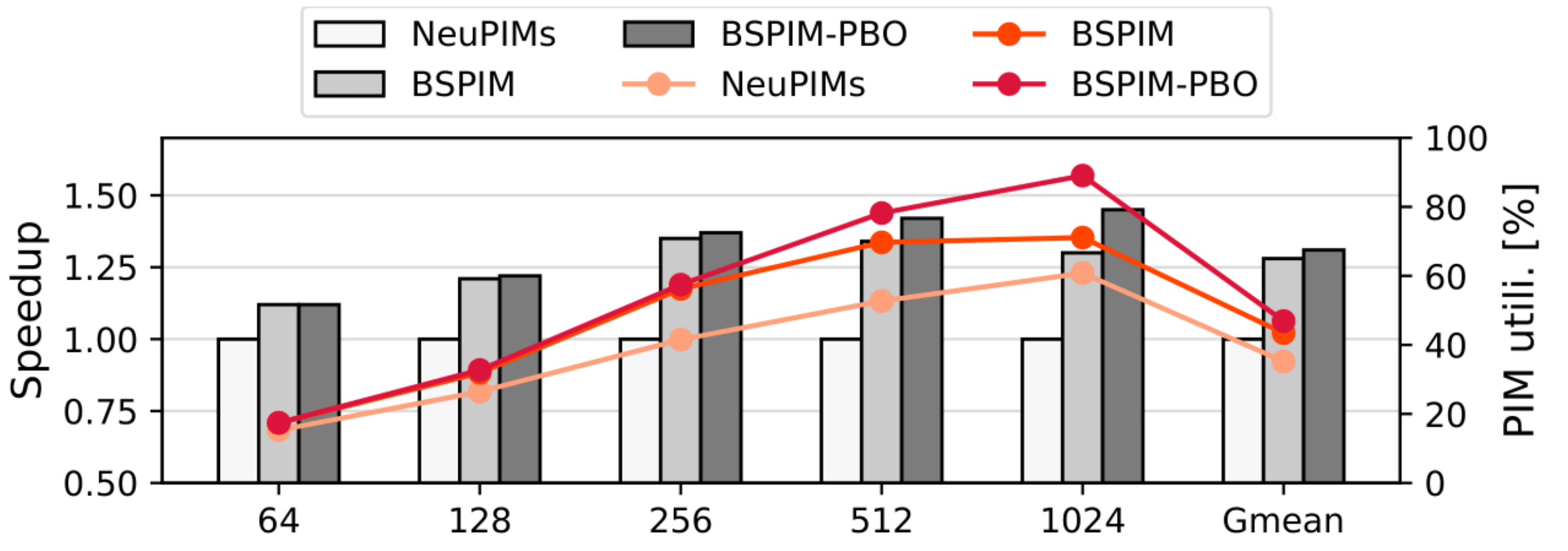


Figure 9. Normalized speedup and PIM utilization when adopting Partial Batch Offloading (PBO)

- **Impact of PBO on performance and utilization**

Figure 13 isolates the benefit of Partial Batch Offloading by comparing BSPIM with and without PBO to NeuPIMs on GPT3-30B. With PBO enabled, BSPIM achieves an additional 1.45× speedup at batch 1024 and boosts PIM utilization by up to 28%, confirming that offloading partial batch to idle device effectively mitigates imbalances and keeps both devices busy.

COMPASSLAB
SKKU Computer Architecture & Systems Laboratory

62