Simulation of Bit-Serial Digital Processing-In-Memory Systems

Siyuan Ma[1], Jiajun Hu[2], Lizy John[1], Aman Arora[2]

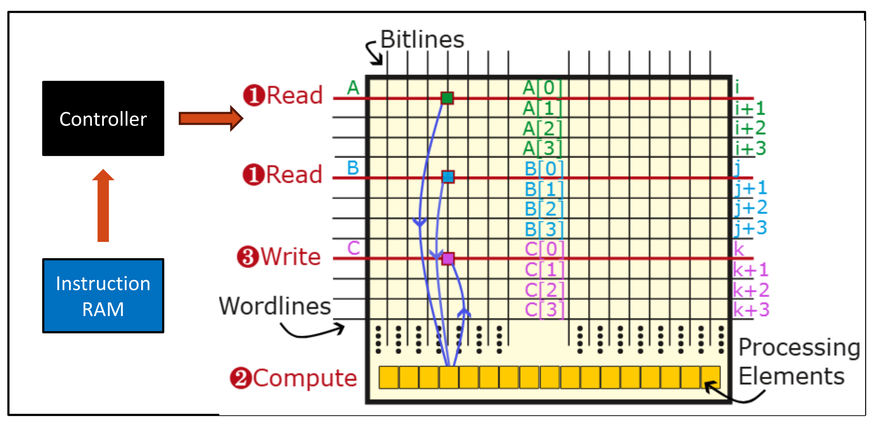
[1]Univrersity of Texas at Austin, [2]Arizona State University

Introduction

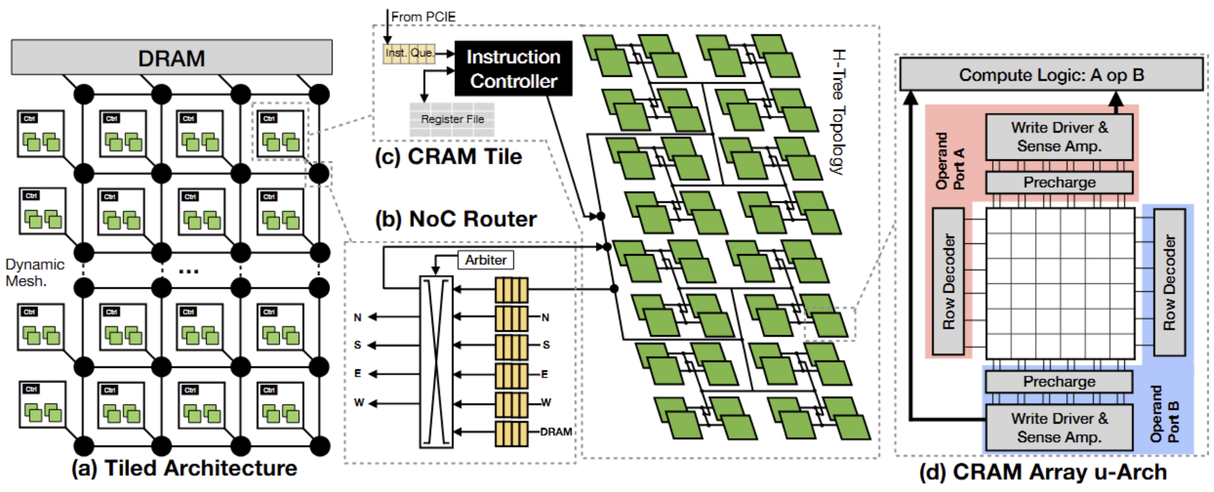
- Many modern applications are memory-bound. Processing-in-Memory (PiM) Systems address the memory bottleneck issue.
- PiM categories: Digital, Analog, Bit-Parallel, Bit-Serial, In-Memory, Near-Memory etc.
- Bit-Serial Digital PiM:
  - Store data and perform computation across RAM rows.
  - + Flexibility in data precision, Massive parallelism, Low area overhead
  - - Unconventional data layout, complex data communication, lack of simulator/modelling framework
- This work introduce two modelling framework for SRAM and DRAM based PiM systems

Bit-Serial PiM Systems

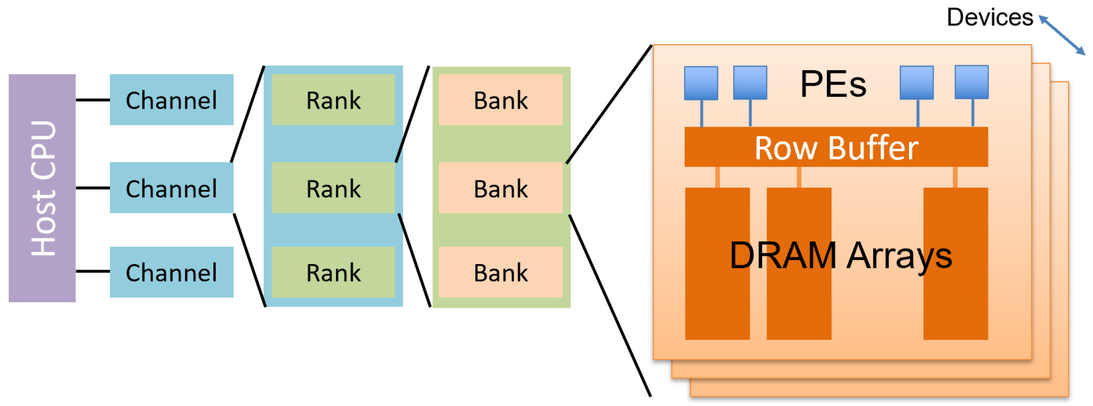
- Bit-Serial Computation in Compute-RAM (CRAM)[1]. Two rows in CRAM are activated to perform a bit-wise operation. The result bit is stored to the third row. This operation is performed on all bits of operands sequentially, and parallelly across columns



- SRAM PiM System[2]. CRAMs are organized by H-Tree to form a tile. Tiles are connected by packed-switched mesh. The system stores data using on-device DRAM, similar to GPU HBM.



- DRAM PiM System. Processing Elements (PE) are added at row buffer within banks to form Compute RAMs. DRAM hierarchies (channel, rank, bank, device) provides massive parallelisms



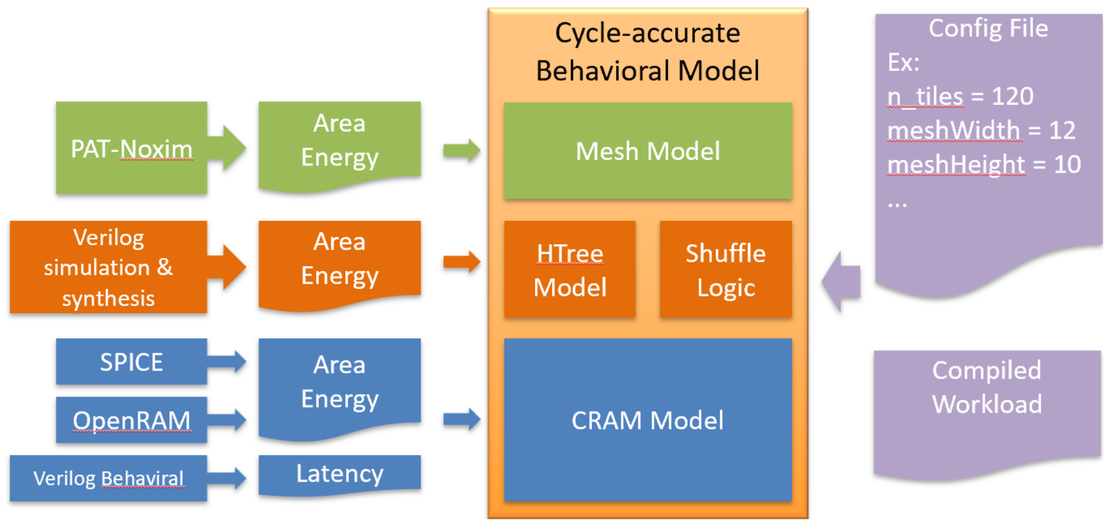
References:

[1] A. Arora et al., "COMEFA: Deploying Compute-in-Memory on FPGAs for Deep Learning Acceleration," ACM Transactions on Reconfigurable Technology and Systems, vol. 16, no. 3, pp. 1–34, Jun. 2023, doi: 10.1145/3603504.

[2] S. Ma *et al.*, "PIMSAB: A P rocessing- I n- M emory System with S patially- A ware Communication and B it-Serial-Aware Computation," *ACM Transactions on Architecture and Code Optimization*, Sep. 2024, doi: 10.1145/3690824.

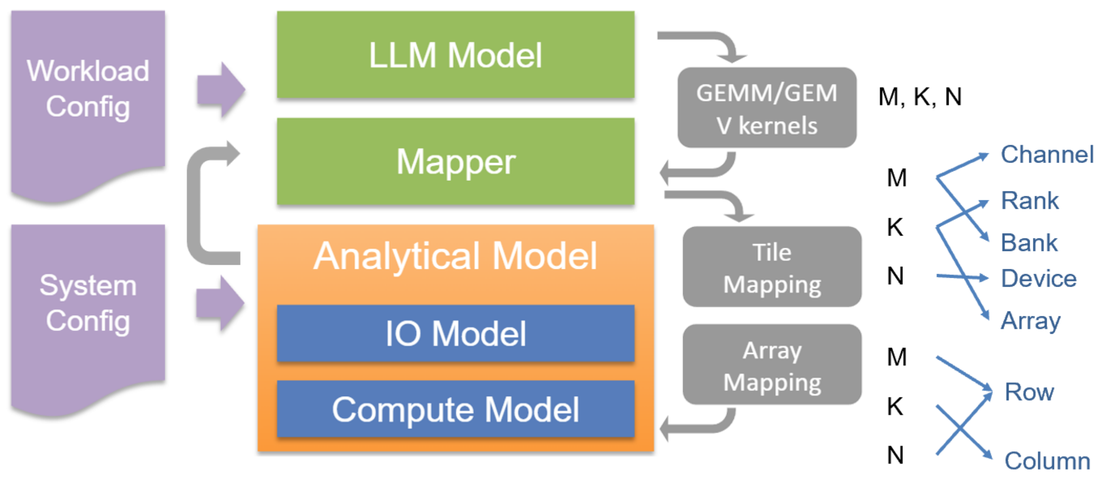
SRAM PiM Modelling Framework

- Area, energy and latency parameters of system components are simulated using PAT-Noxim, Verilog, SPICE and OpenRAM.
- Configuration file specifies system setup.
- Workload is compiled to PiM instructions, including CRAM instructions, IO instructions and data movement instructions.
- Behavioral model simulates system components cycle-by-cycle, provides cycle counts and energy consumptions.



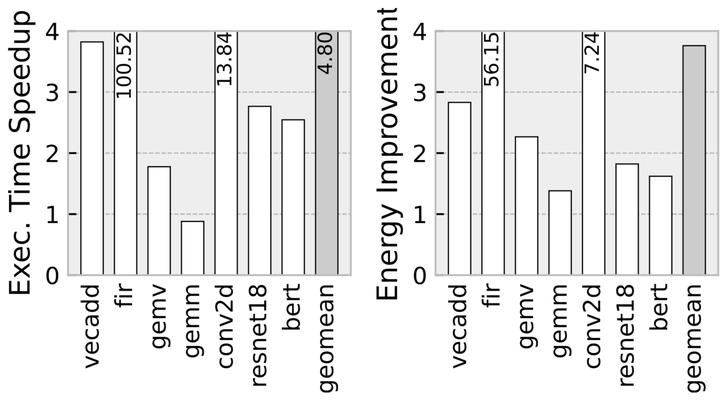
DRAM PiM Modelling Framework

- Along the line of Calculon/LLMCompass, this framework models LLM end-to-end performance for DRAM PiM system.
- Timing parameters are taken from DDR specs. Workload configuration specifies type of LLM transformers and hyper-parameters. System configuration specifies number of Channel, Rank etc.
- Internal LLM model decomposes transformer block to GEMM/GEMV kernels with sizes on M, K, N dimensions
- Mapper searches for mapping strategies to map workload to various parallelism levels. Heuristics and brute-force search are used.
- Analytical model reports cycle counts based on compute cycles and IO cycles

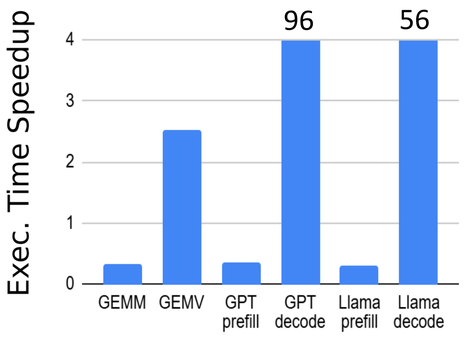


Results

- SRAM PiM achieves 4.8x speedup and 3.88x energy reduction compared to A100
- DRAM PiM achieves 2.5x speedup on GEMV, and 96x, 56x speedup on GPT and Llama decoding compared to A100.
- A100 is faster than both PiM systems on GEMM and GEMM-heavy workloads like LLM prefill due to TensorCores.



SRAM PiM vs. A100 GPU



SRAM PiM vs. A100 GPU