P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures IV

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich

Fall 2021

2 November 2021

Samsung Function-in-Memory DRAM (2021)

Samsung Develops Industry's First High Bandwidth Memory with AI Processing Power

Korea on February 17, 2021 Audio Share

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

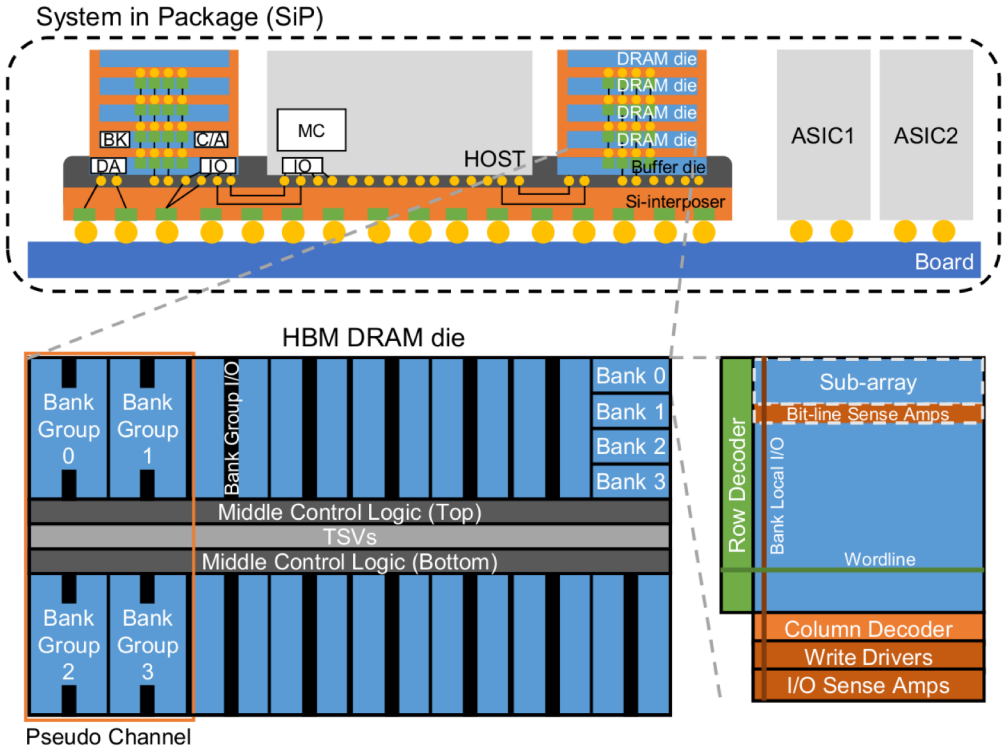
Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, "Our groundbreaking HBM-PIM is the industry's first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications."

https://news.samsung.com/global/samsung-develops-industrys-first-high-bandwidth-memory-with-ai-processing-power

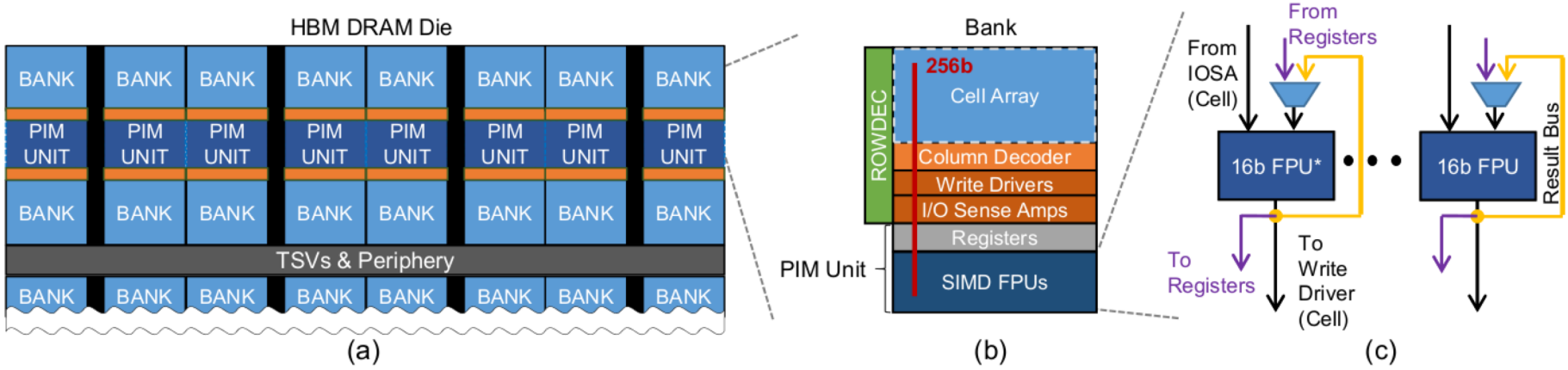
Background: High Bandwidth Memory (HBM)

- HBM stacks DRAM layers and a buffer layer
  - The buffer layer contains I/O circuitry, self-test, test/debug
- DRAM layers and buffer layer communicate using Through Silicon Vias (TSVs)
- The buffer layer is connected to a host processor via a silicon interposer
- 1 HBM2 die comprises 4 pseudo channels (pCHs) each with 4 bank groups
  - An access transfers a 256-bit data block over 4 64-bit bursts over one pCH



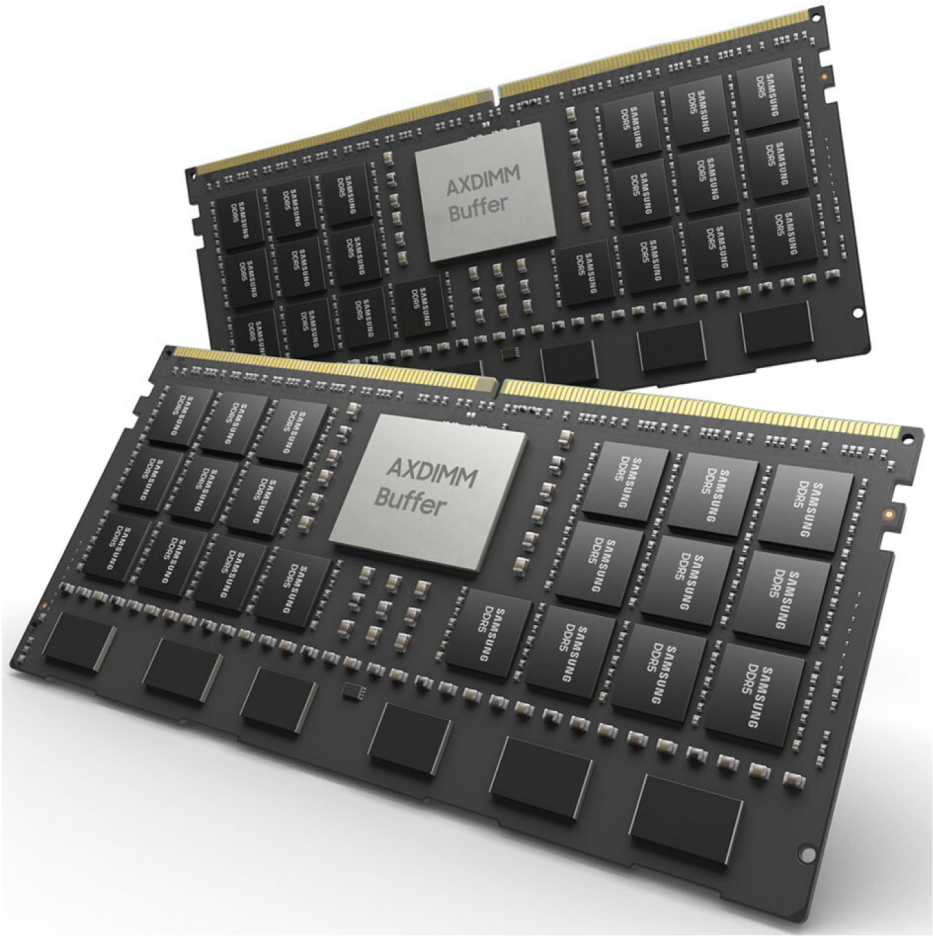
FIMDRAM: System Organization (II)

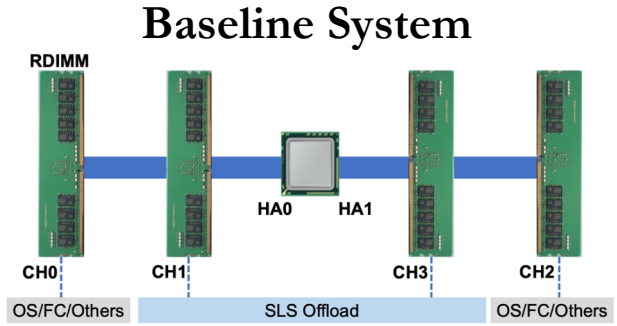
- Design goals:
  - 1. Support DRAM and PIM-DRAM mode for versatility
  - 2. Minimize the engineering cost of redesigning DRAM banks and sub-arrays
- Thus, PIM unit at I/O boundary of bank
  - 1 PIM unit for each 2 banks
  - 16 16-bit SIMD floating-point units (FPUs) per PIM unit

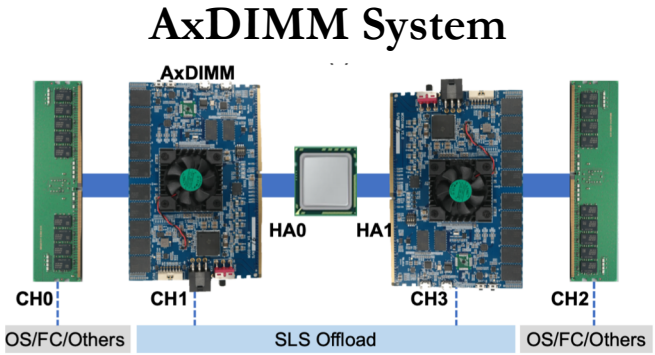


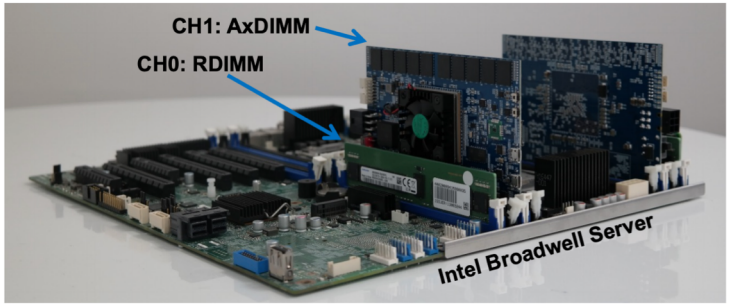
Samsung AxDIMM (2021)

- DDR5-PIM
  - DLRM recommendation system









Ke et al. "Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM", IEEE Micro (2021)

PnM with AxDIMM (IEEE Micro 2021)

Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM

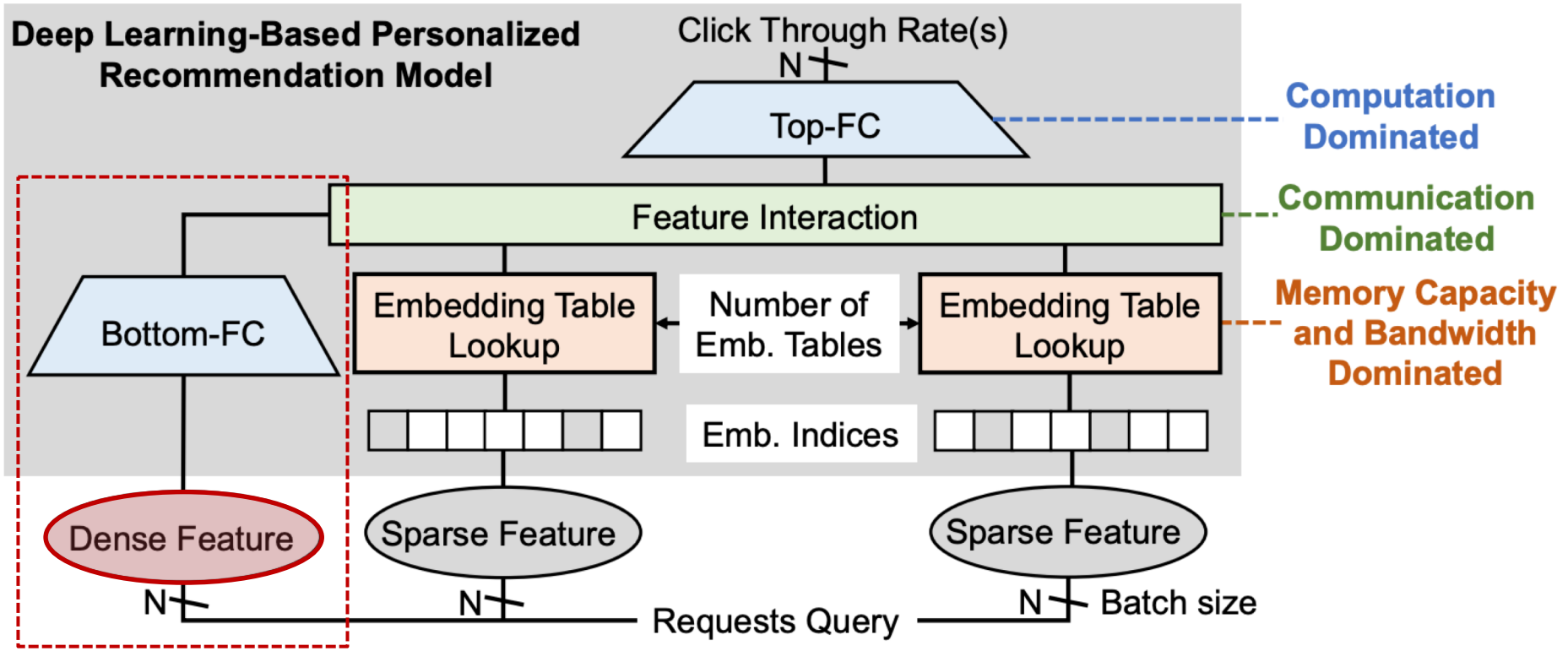
Liu Ke*†, Xuan Zhang†, Jinin So‡, Jong-Geon Lee‡, Shin-Haeng Kang‡, Sukhan Lee‡, Songyi Han‡, YeonGon Cho‡, JIN Hyun Kim‡, Yongsuk Kwon‡, KyungSoo Kim‡, Jin Jung‡, Ilkwon Yun‡, Sung Joo Park‡, Hyunsun Park‡, Joonho Song‡, Jeonghyeon Cho‡, Kyomin Sohn‡, Nam Sung Kim‡, Hsien-Hsin S. Lee*

*Facebook, †Washington University in St. Louis, ‡Samsung

https://doi.org/10.1109/MM.2021.3097700

Overview of Recommendation Models

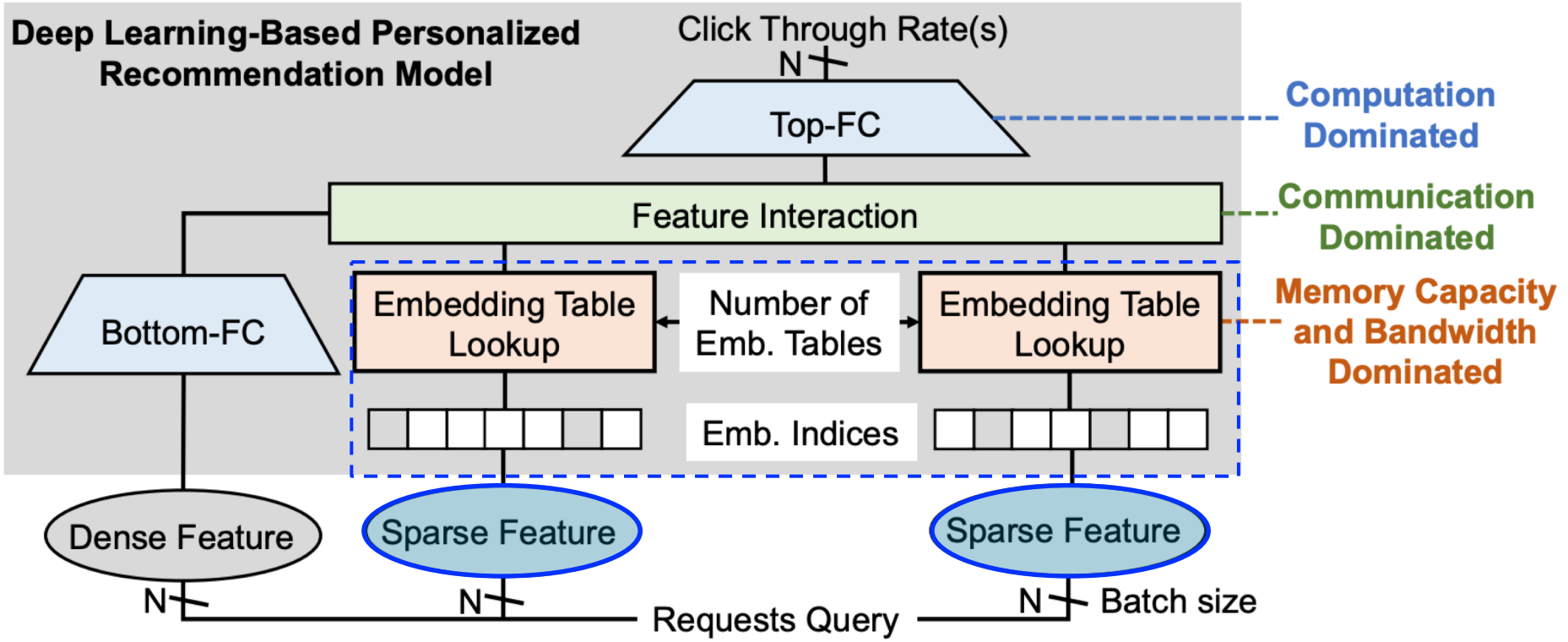
- Personalized recommendation: recommend content to users, e.g., Facebook's DLRM recommendation system



Dense features: continuous inputs in vectors and matrices are processed by typical DNN layers (e.g., fully connected layers)

Overview of Recommendation Models

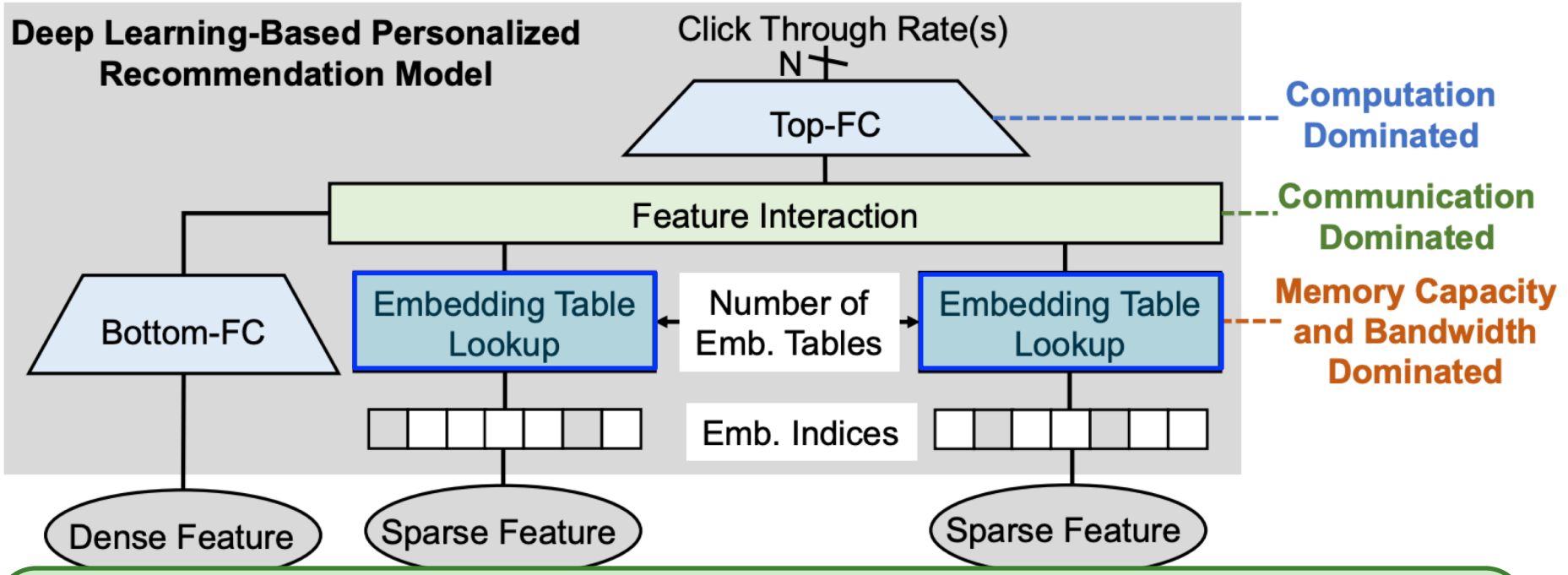
- Personalized recommendation: recommend content to users, e.g., Facebook's DLRM recommendation system



Sparse features: for categorical inputs;
processed by indexing large embedding tables

Overview of Recommendation Models

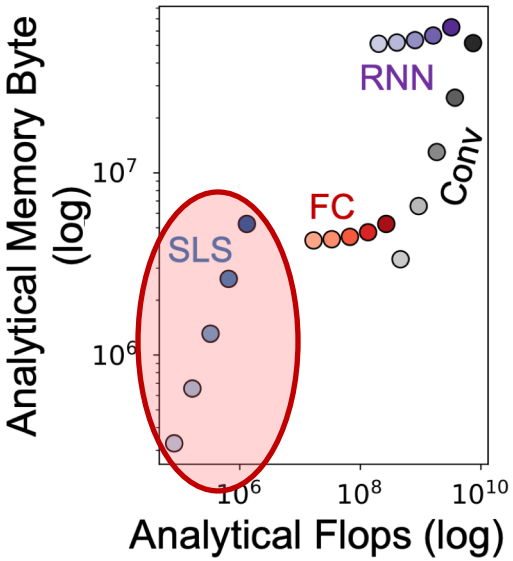
- Personalized recommendation: recommend content to users, e.g., Facebook's DLRM recommendation system

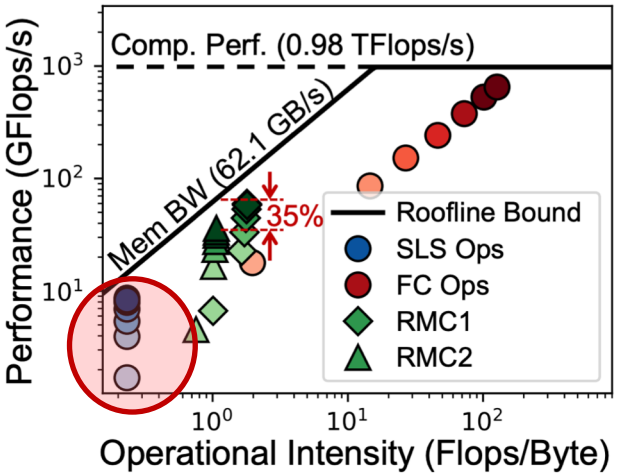


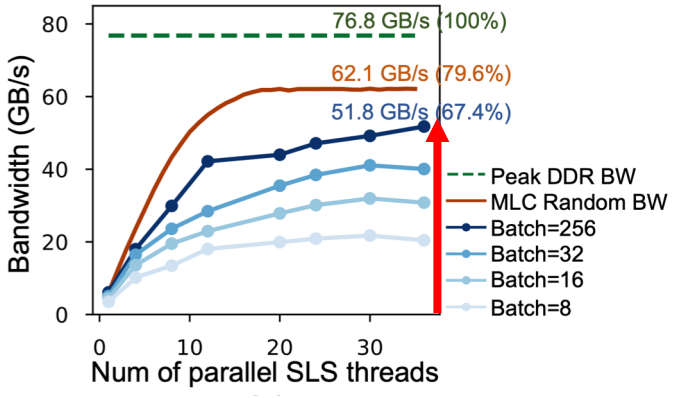
Embedding tables are organized as a set of potentially millions of vectors: lookup and pooling operations represent sparse features learned during training and generally exhibit Gather-Reduce pattern, via Caffe2's SparseLengths (SLS) operators

DLRM Performance Characterization

- Identifying key performance bottlenecks for the DLRM system





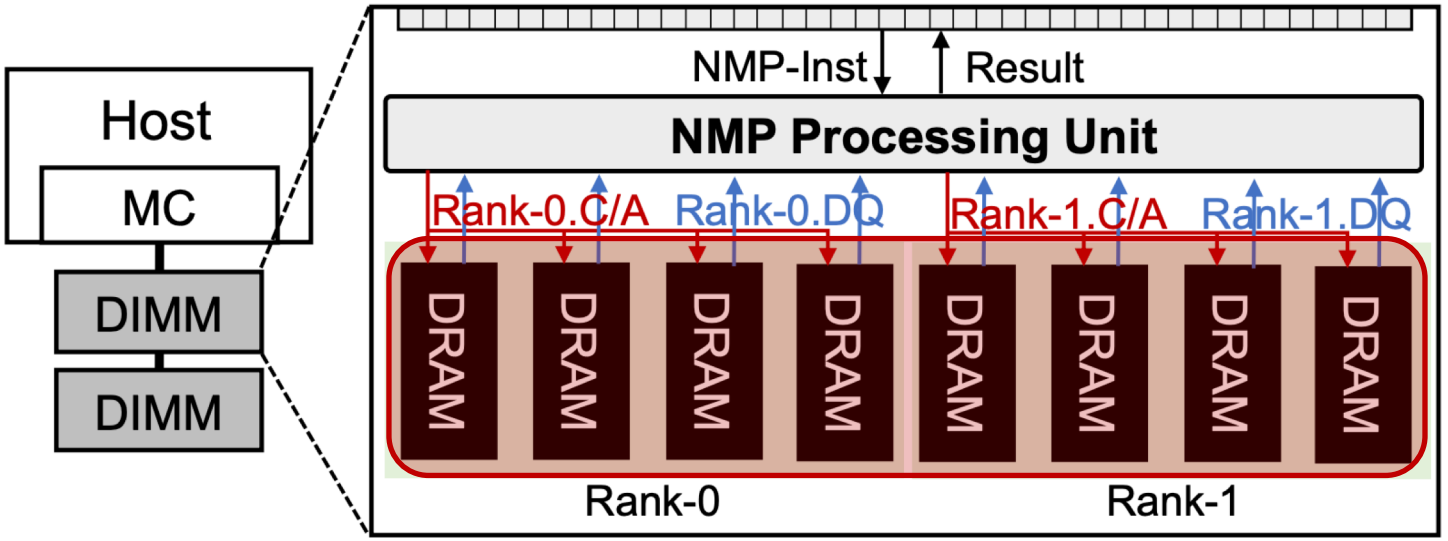


SparseLengths (SLS) operators:
- Low FP intensity
- Larger batch size:
  - Higher memory footprint
  - Higher memory intensity

The memory bandwidth can easily be saturated by embedding operations especially as both the batch size and the number of threads increase

RecNMP Architecture

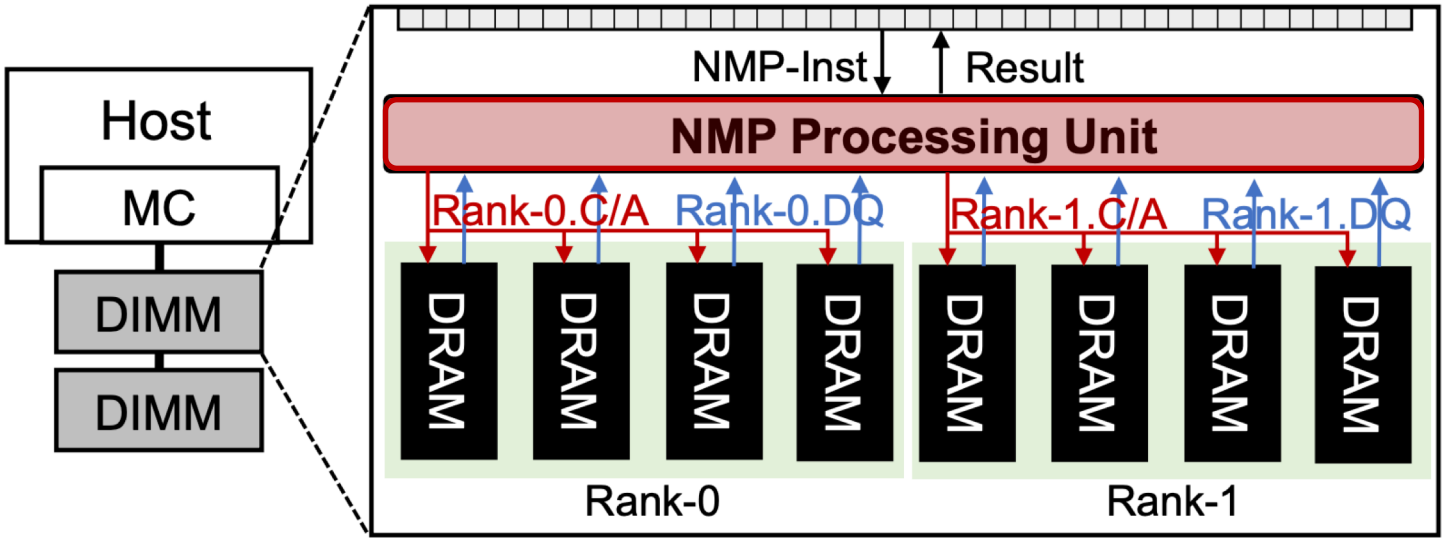
- DIMM-based NMP architecture for recommendation systems
  - Multiply the bandwidth by exploiting rank-level parallelism



Embedding entries are fetched from the concurrently activated ranks

RecNMP Architecture

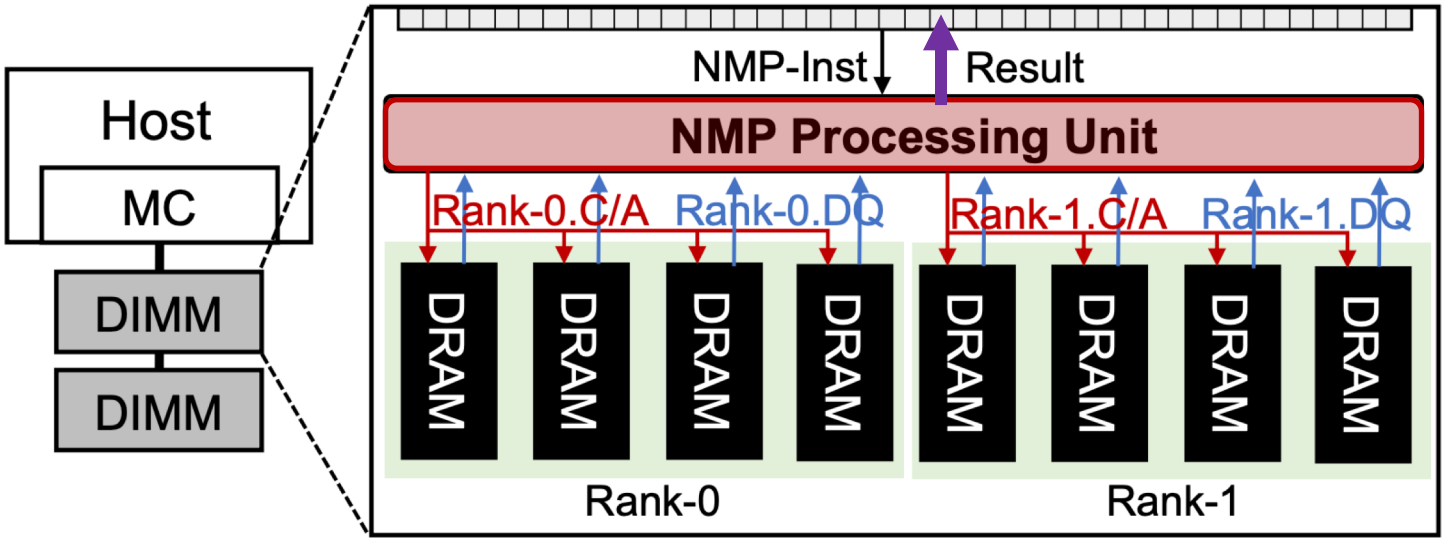
- DIMM-based NMP architecture for recommendation systems
  - Multiply the bandwidth by exploiting rank-level parallelism



The NMP PU performs the local embedding lookup and pooling functions at memory-side, producing the general Gather-Reduce execution pattern

RecNMP Architecture

- DIMM-based NMP architecture for recommendation systems
  - Multiply the bandwidth by exploiting rank-level parallelism

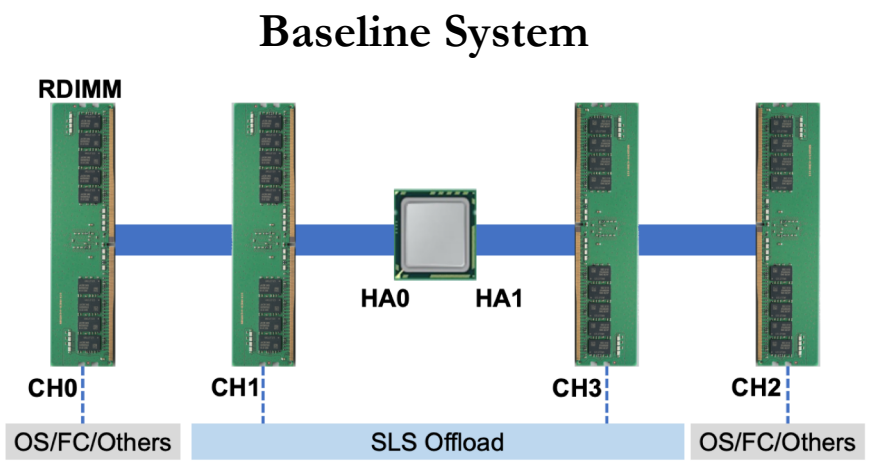


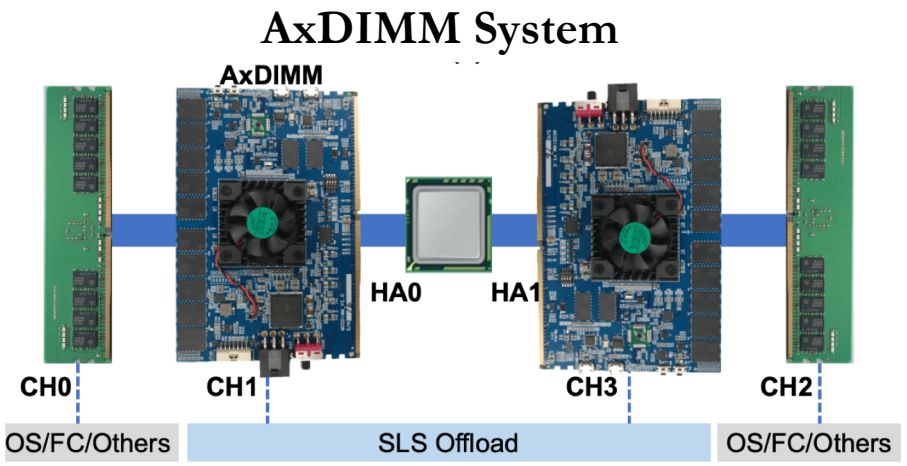
Element-wise summation of the embedding entries is performed inside the NMP PU, and the final pooling result is transferred back to host

AxDIMM Design: Overview

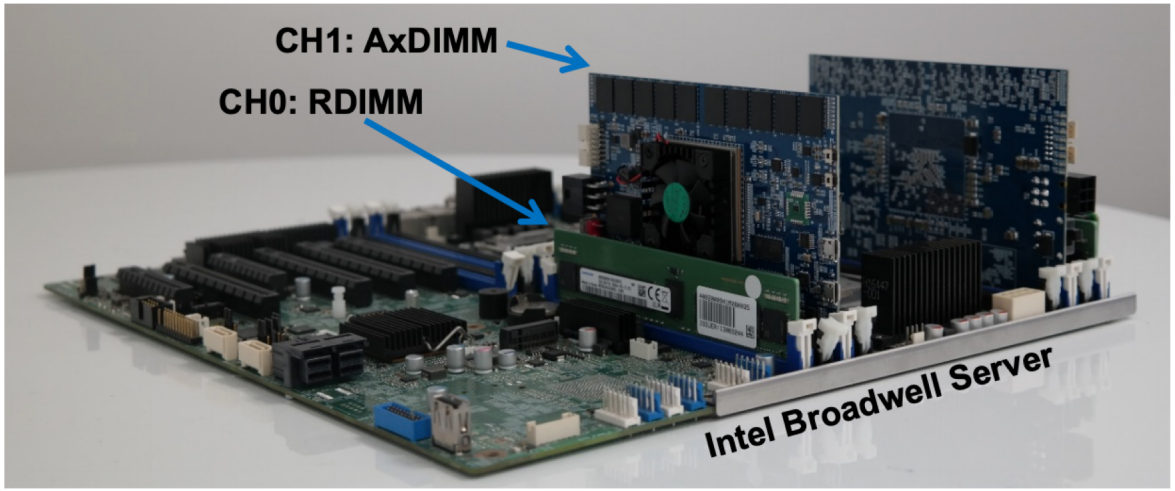
- Accelerator DIMM (AxDIMM)
  - DDR4-compatible FPGA-based platform with standard memory interfaces
- AxDIMM can potentially
  - support both in-order general-purpose processor and specialized accelerator modules
  - be an ideal prototyping platform for near-memory processing
- RecNMP case study, including:
  - hardware implementation
  - software-stack support

AxDIMM System



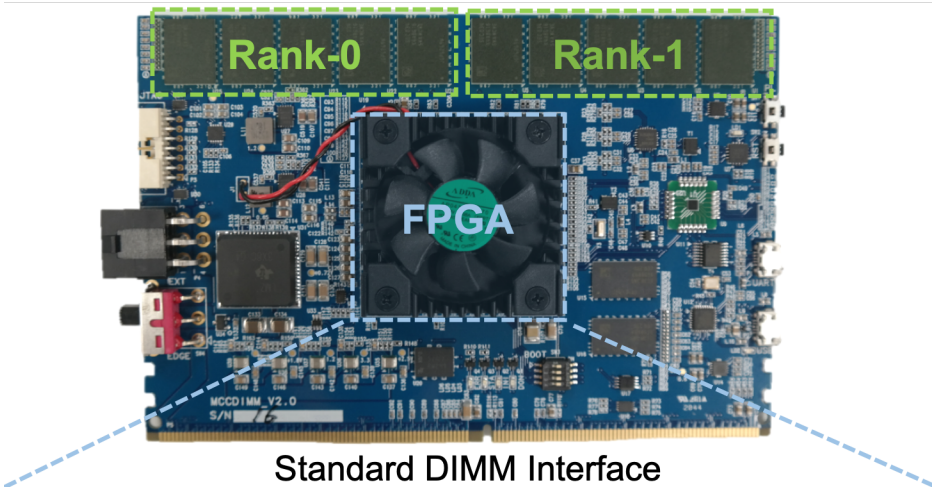


FPGA: Xilinx XCZU19EG FPGA



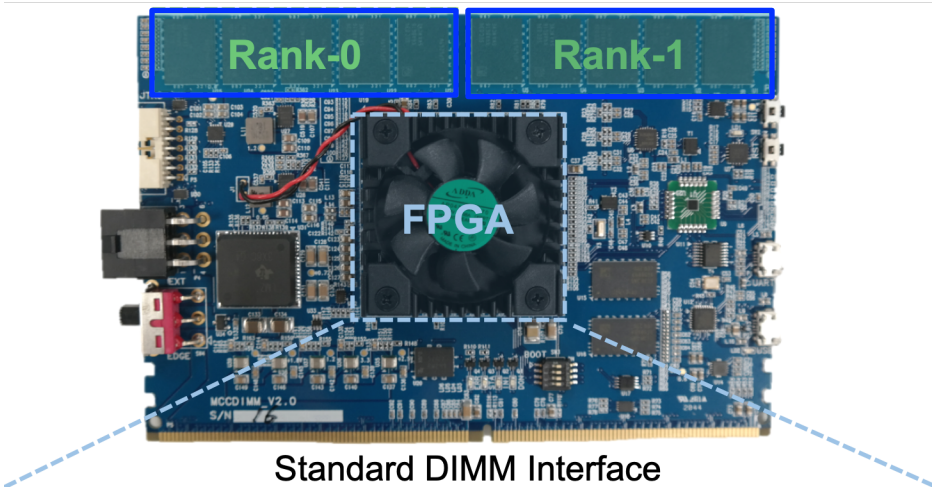
System was slowed down (1/3 of normal DDR4 memory channel speedup; CPU went from 3.2 GHz to 1.2 GHz) to keep up with the FPGA IO speed

AxDIMM Design: Hardware Architecture



FPGA board with standard DIMM interface:
It serves as a real-system
near-memory processing implementation

AxDIMM Design: Hardware Architecture

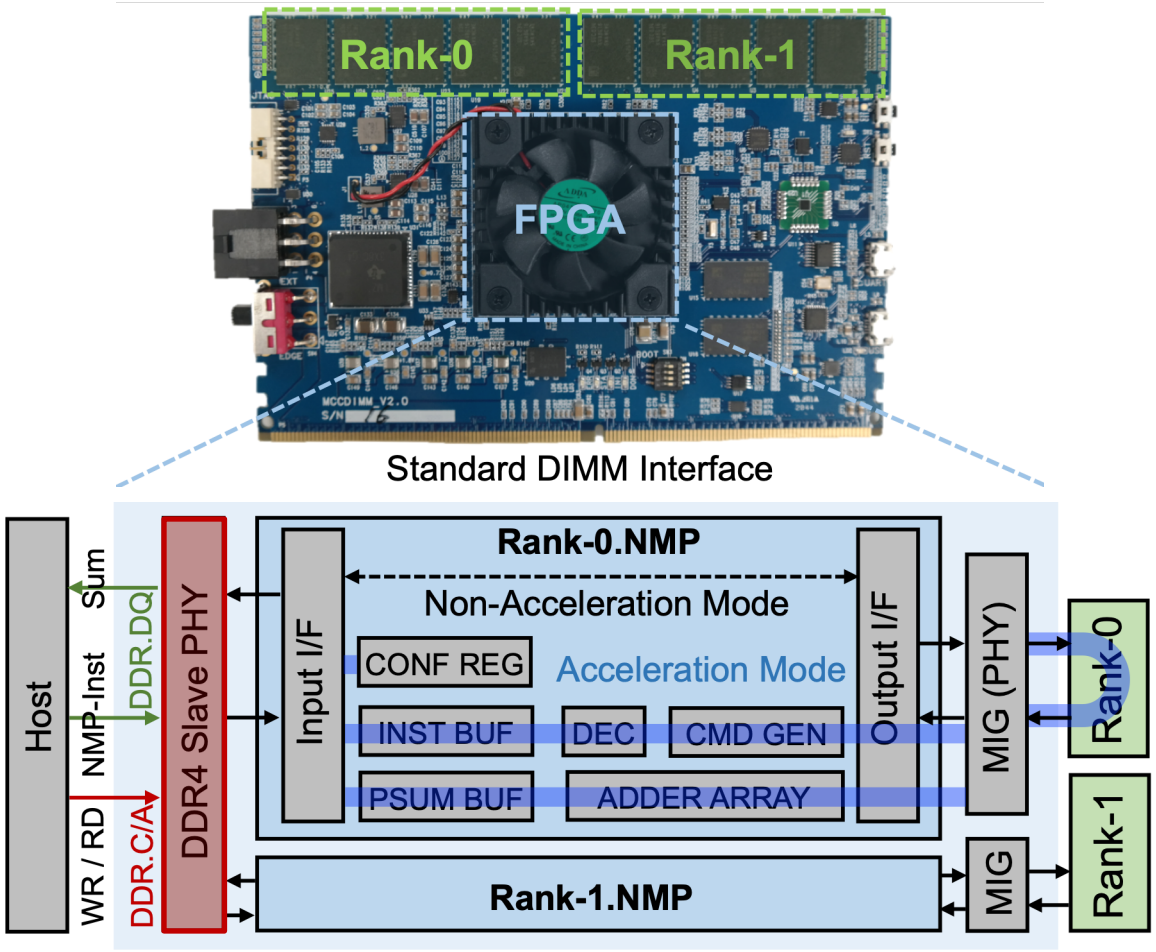


Standard DIMM Interface

Rank-level parallelism:
Two DRAM ranks are activated in parallel
to load embedding entries from memory

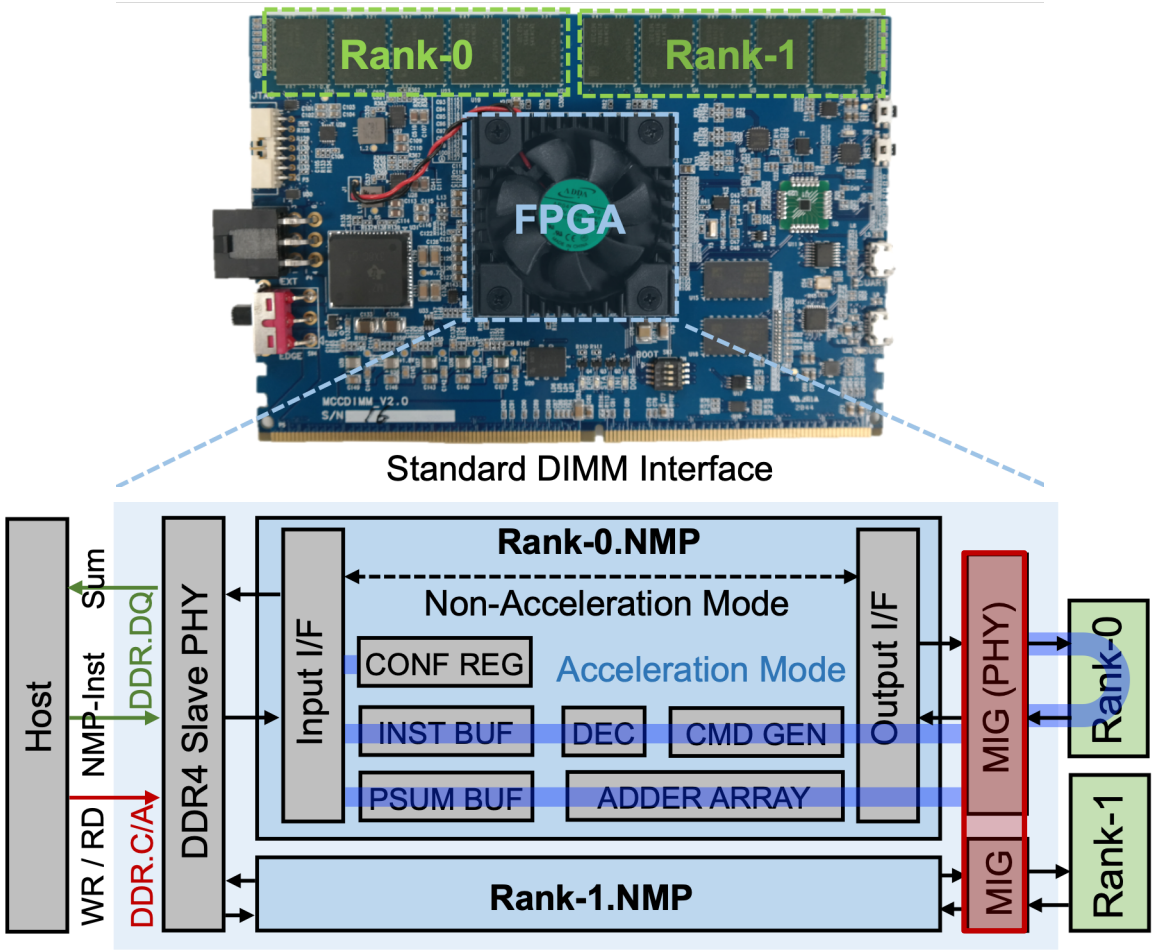
Element-wise summation
is performed inside the FPGA module

AxDIMM Design: Hardware Architecture



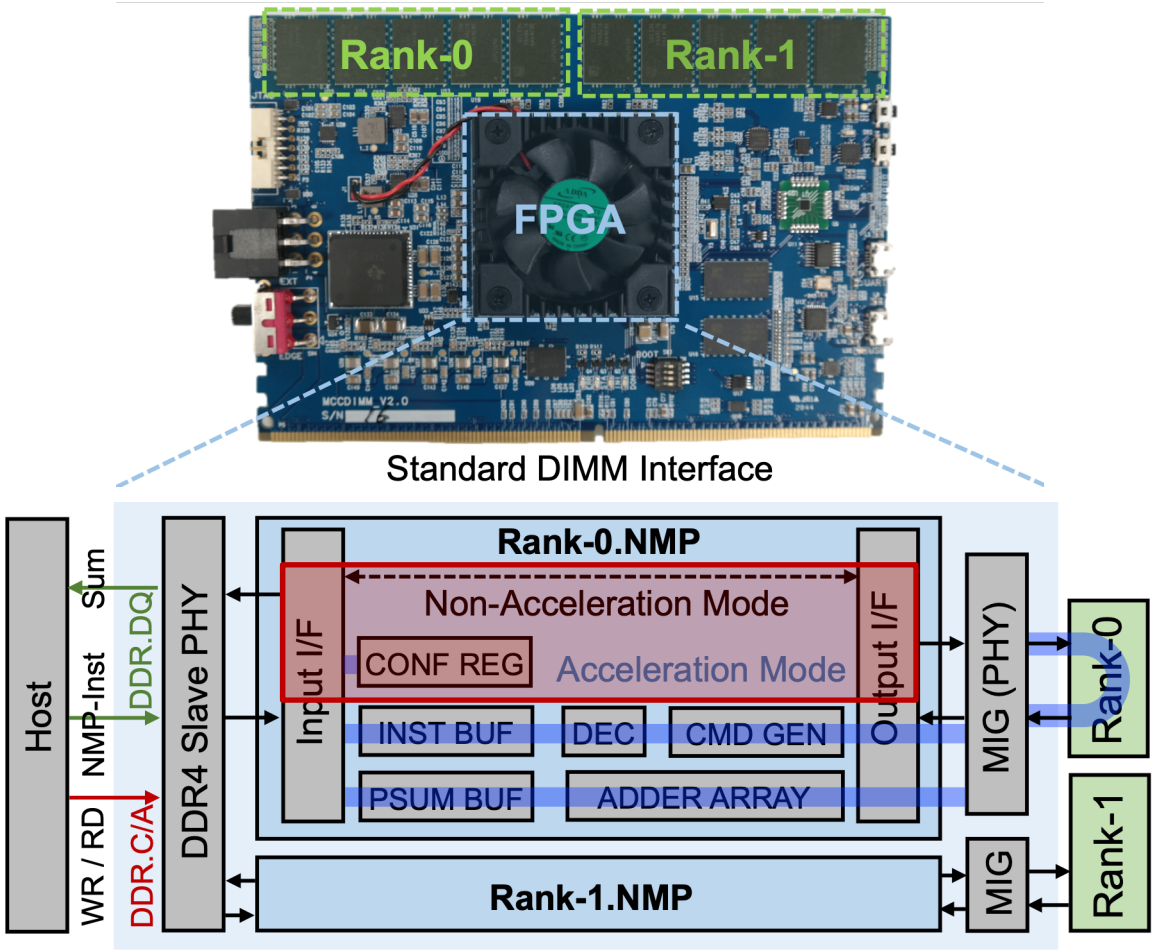
DDR4 slave PHY receives DRAM commands and NMP instructions (via DQ pins) from the host side

AxDIMM Design: Hardware Architecture



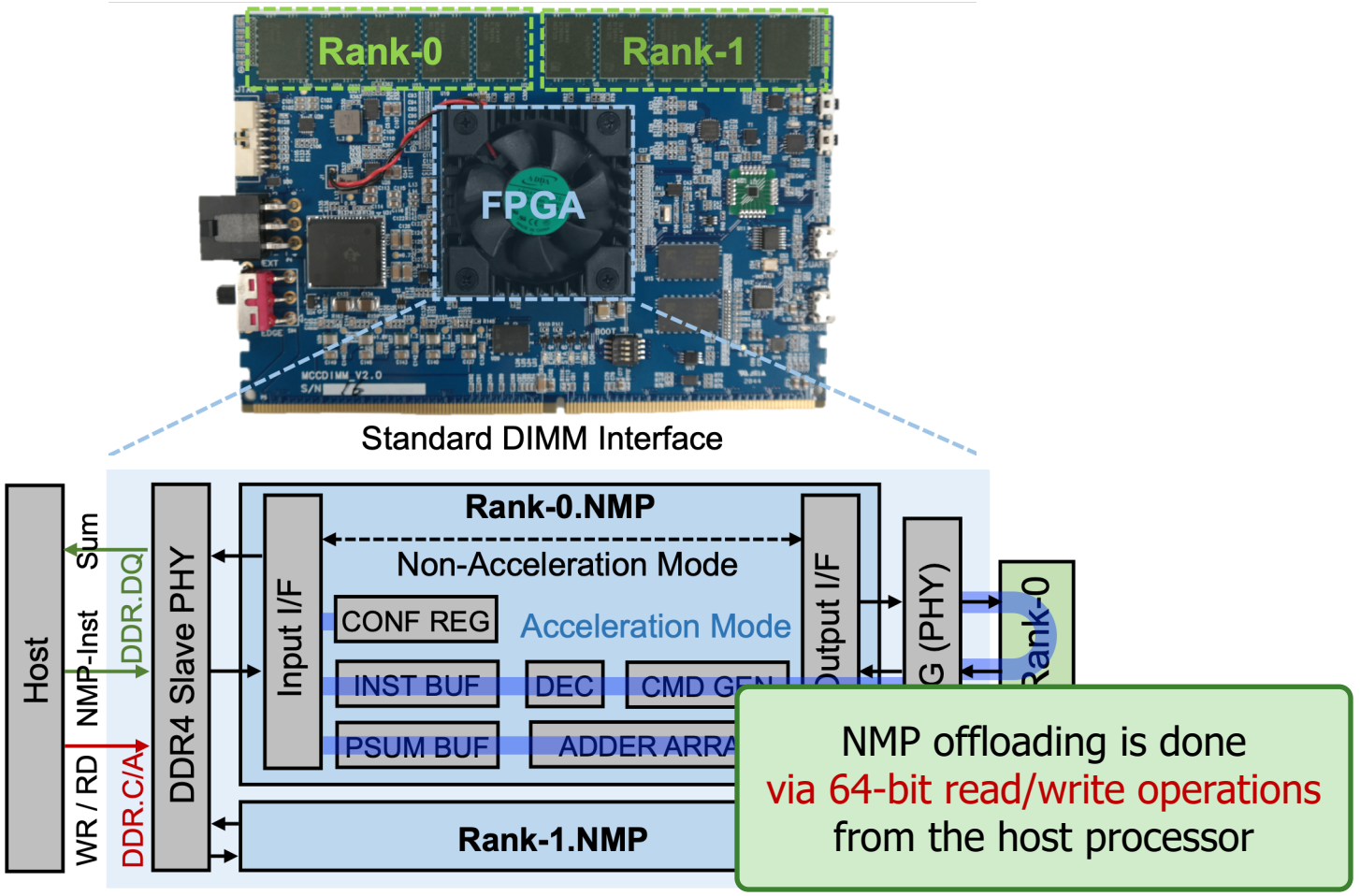
The memory interface generator (MIG) supports the internal rank accesses between Rank-NMP and the DRAM device

AxDIMM Design: Hardware Architecture



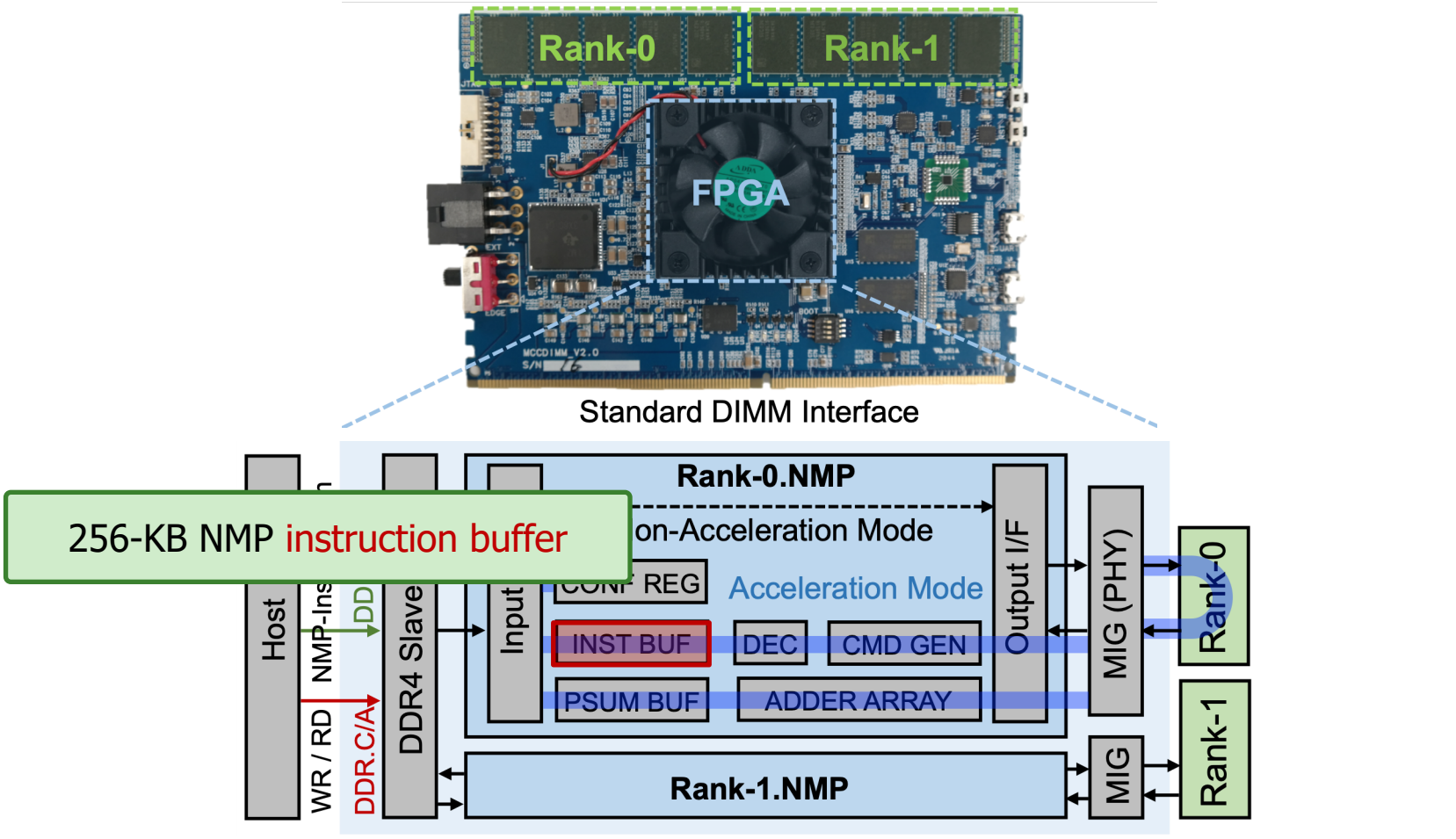
Two execution modes:
(1) non-acceleration mode
(2) acceleration mode (blocking)

AxDIMM Design: Hardware Architecture



NMP-Inst (64 bits)	OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
	2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

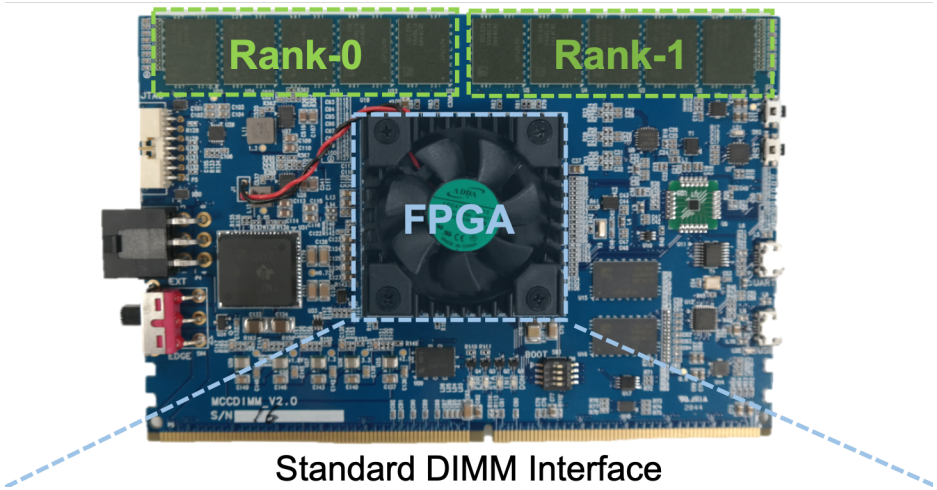
AxDIMM Design: Hardware Architecture



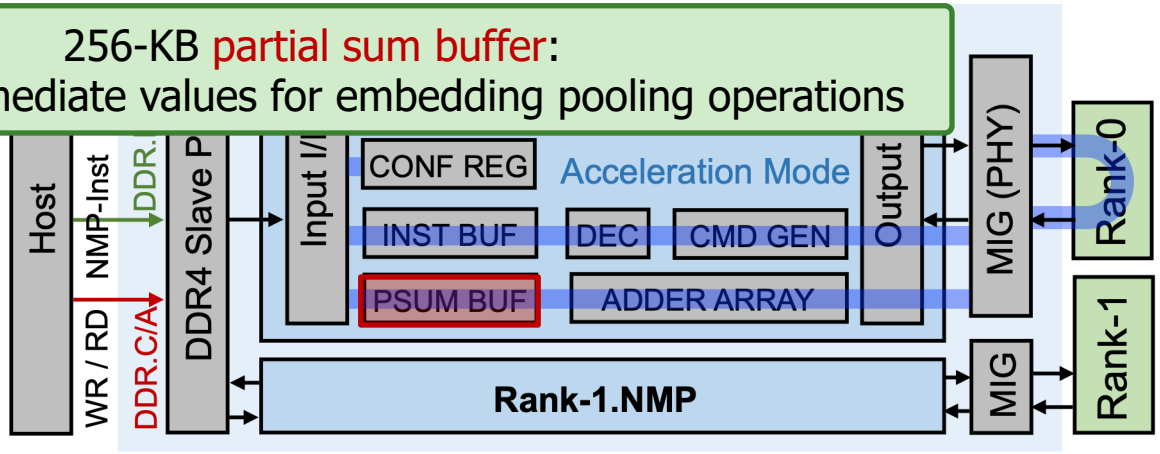
NMP-Inst (64 bits)

OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

AxDIMM Design: Hardware Architecture



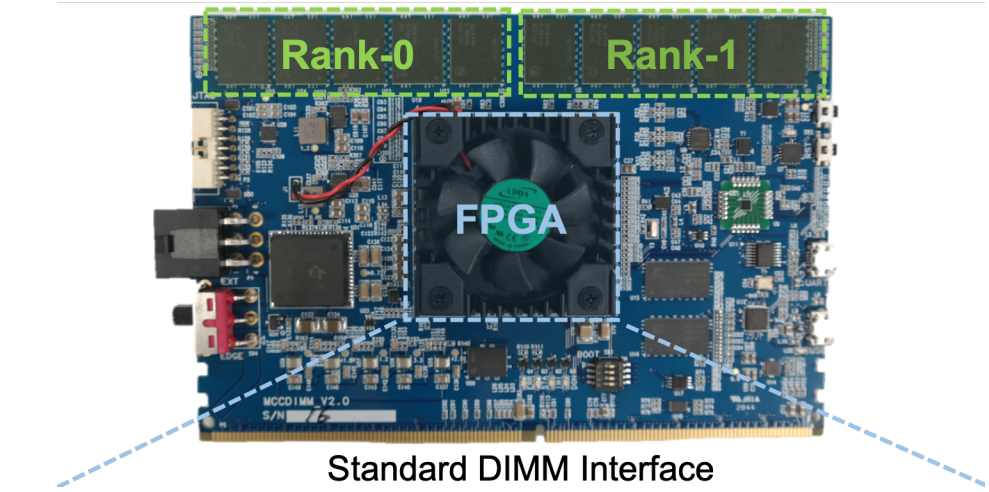
256-KB partial sum buffer:
It stores intermediate values for embedding pooling operations



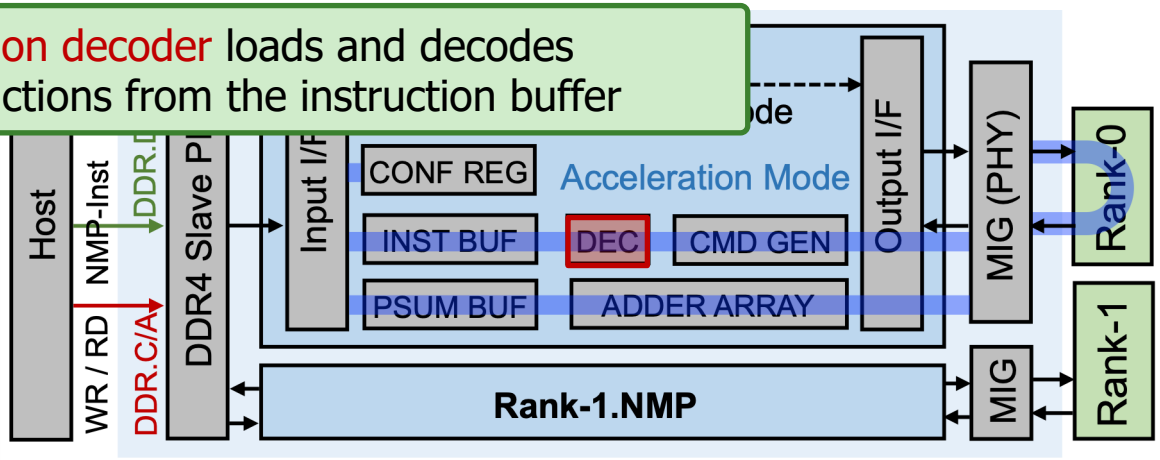
NMP-Inst (64 bits)

OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

AxDIMM Design: Hardware Architecture



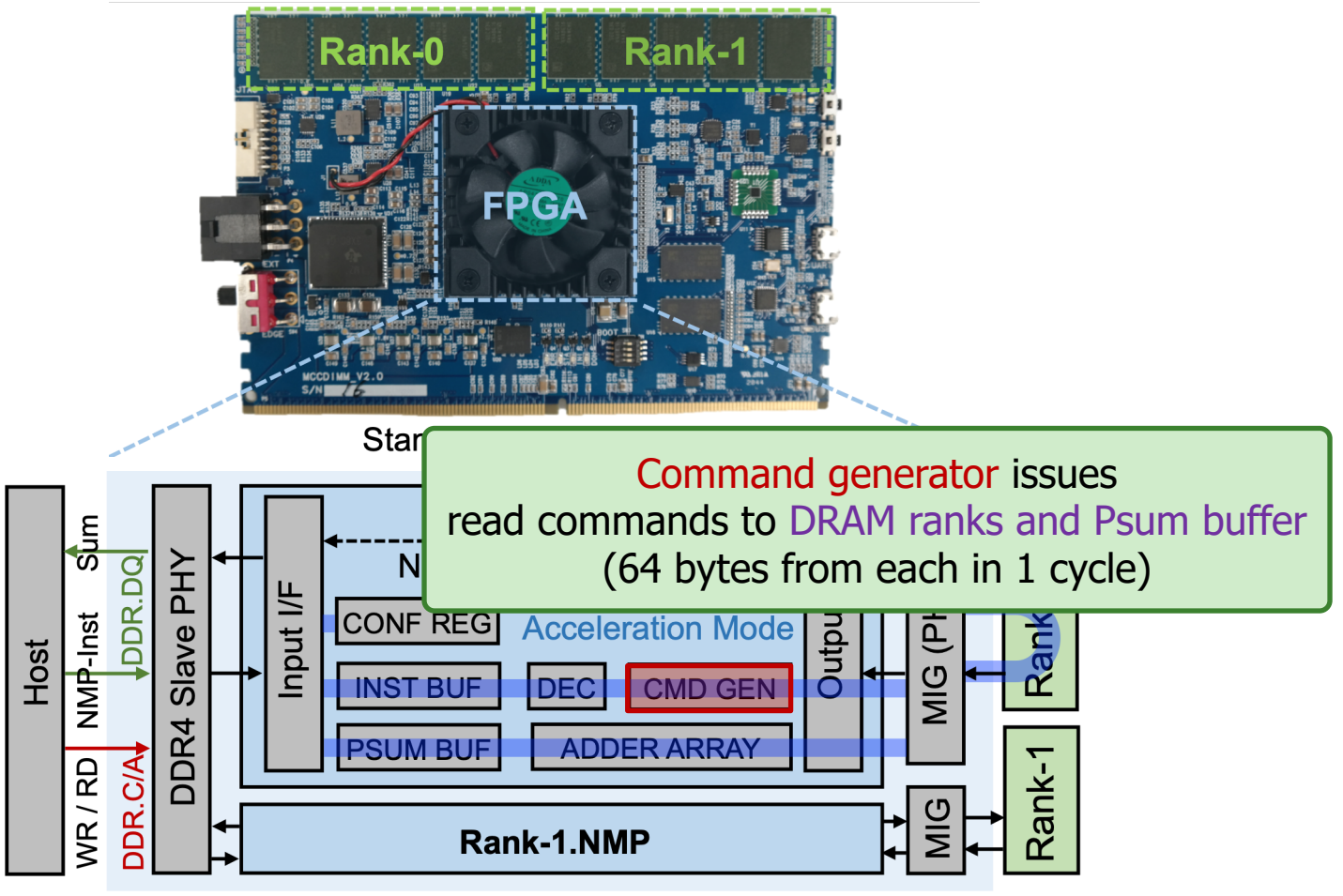
Instruction decoder loads and decodes
NMP instructions from the instruction buffer



NMP-Inst (64 bits)

OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

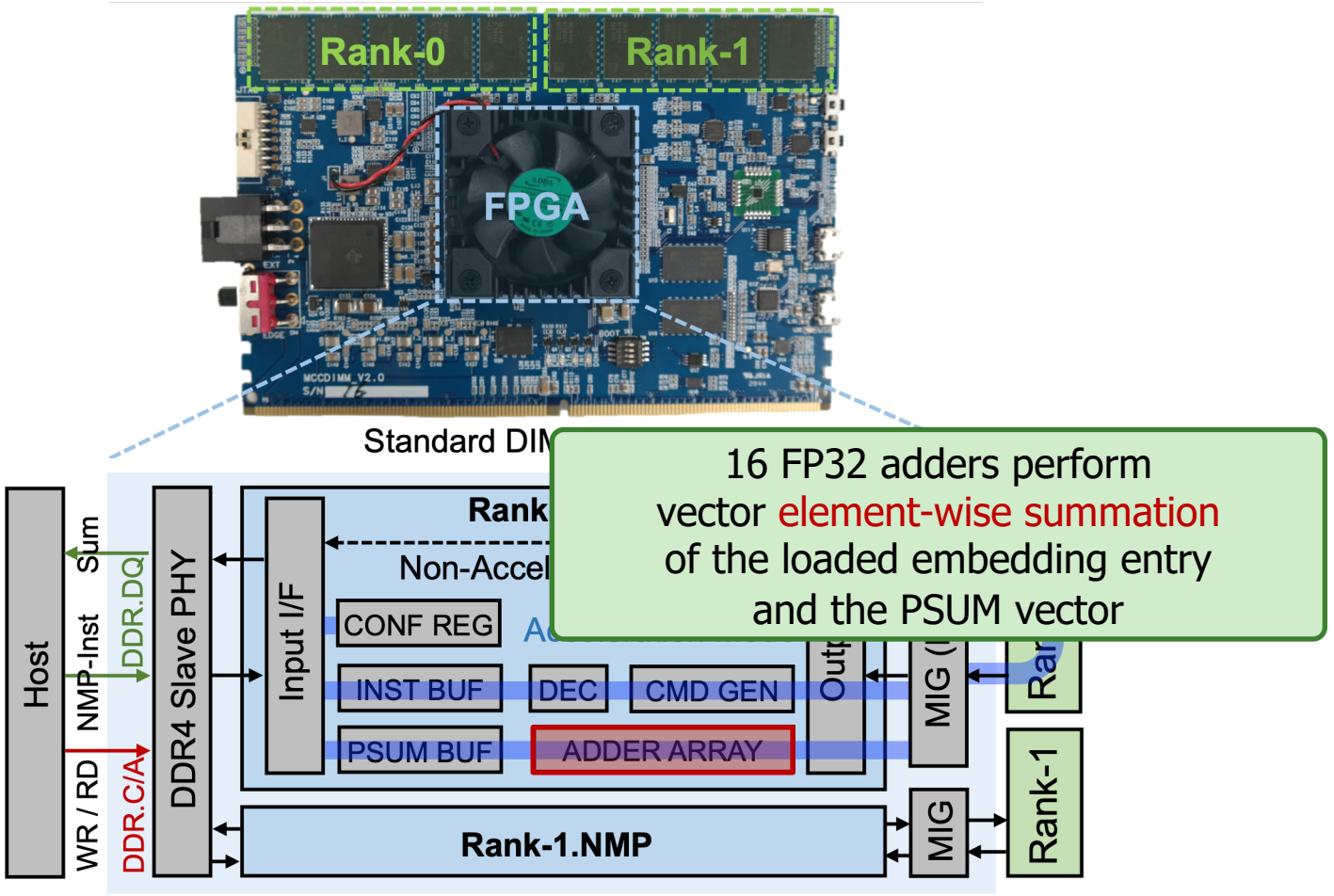
AxDIMM Design: Hardware Architecture



NMP-Inst (64 bits)

OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

AxDIMM Design: Hardware Architecture

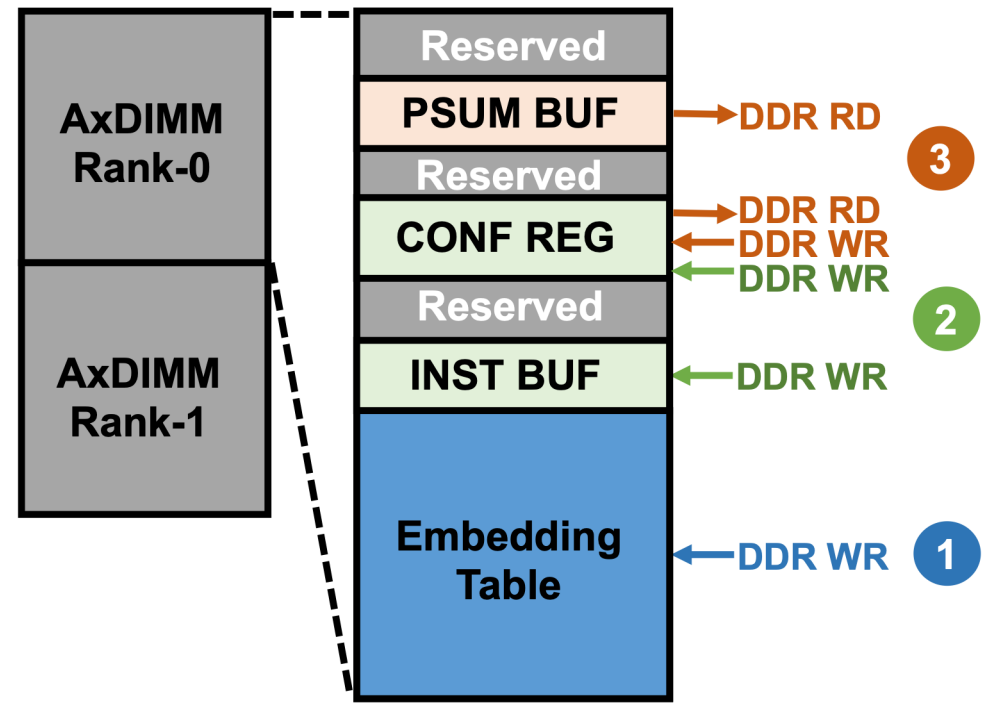


NMP-Inst (64 bits)

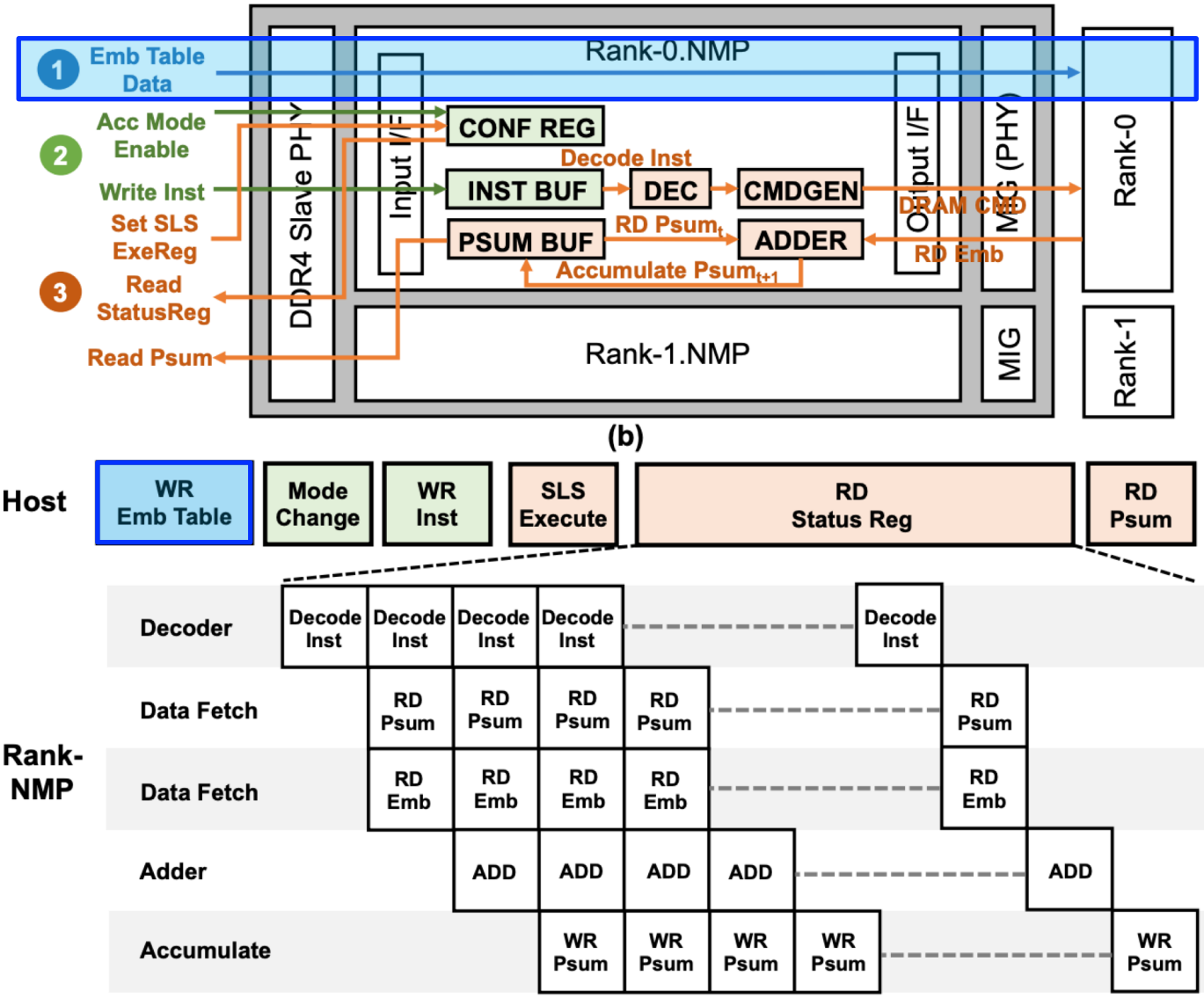
OpCode	Locality	PSUM Tag	Trace End	Reserved	Row Addr	BG	BA	Col Addr
2 bit	1 bit	12 bit	1 bit	17 bit	17 bit	2 bit	2 bit	10 bit

AxDIMM Design: Address Map

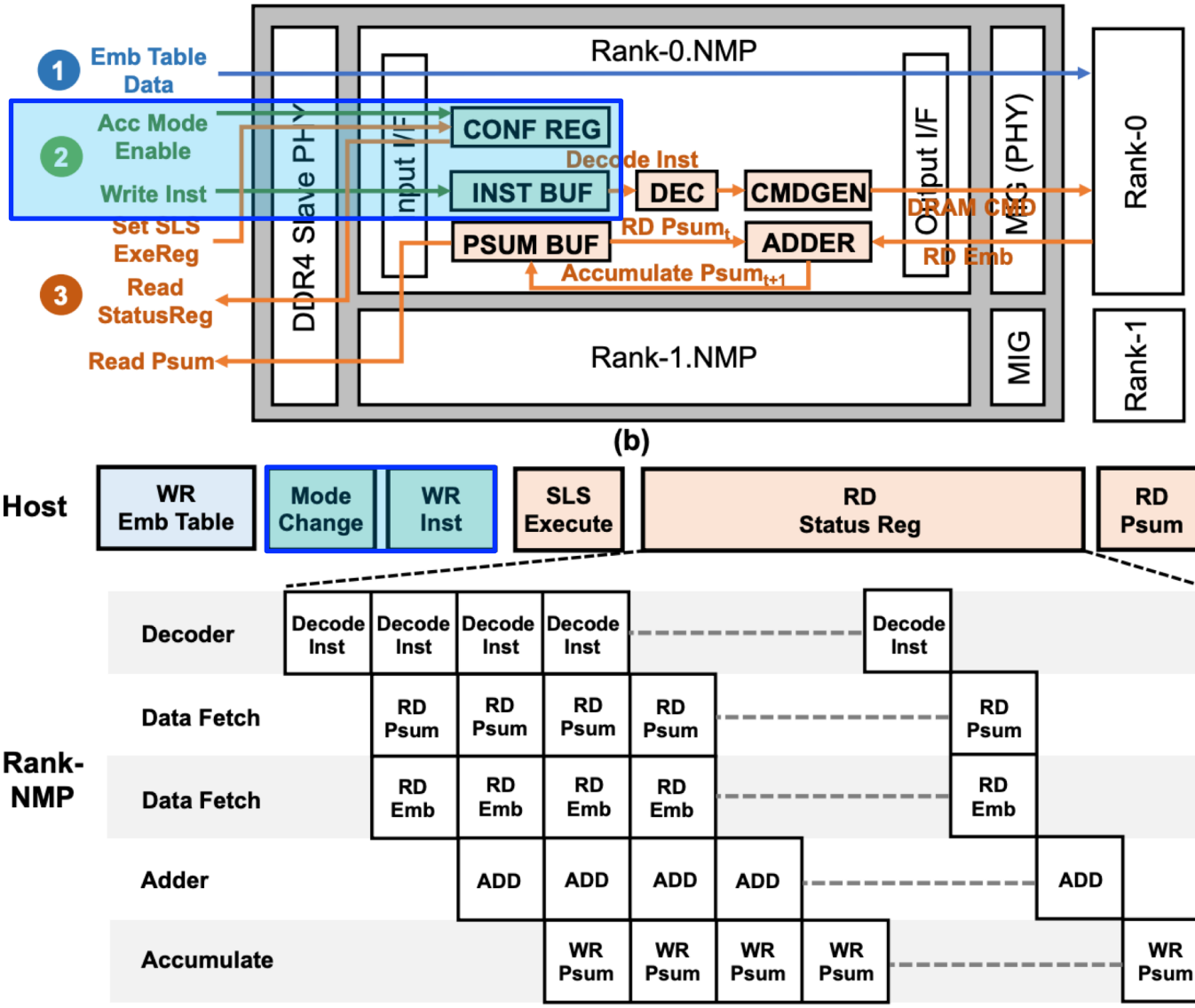
- Memory map of AxDIMM



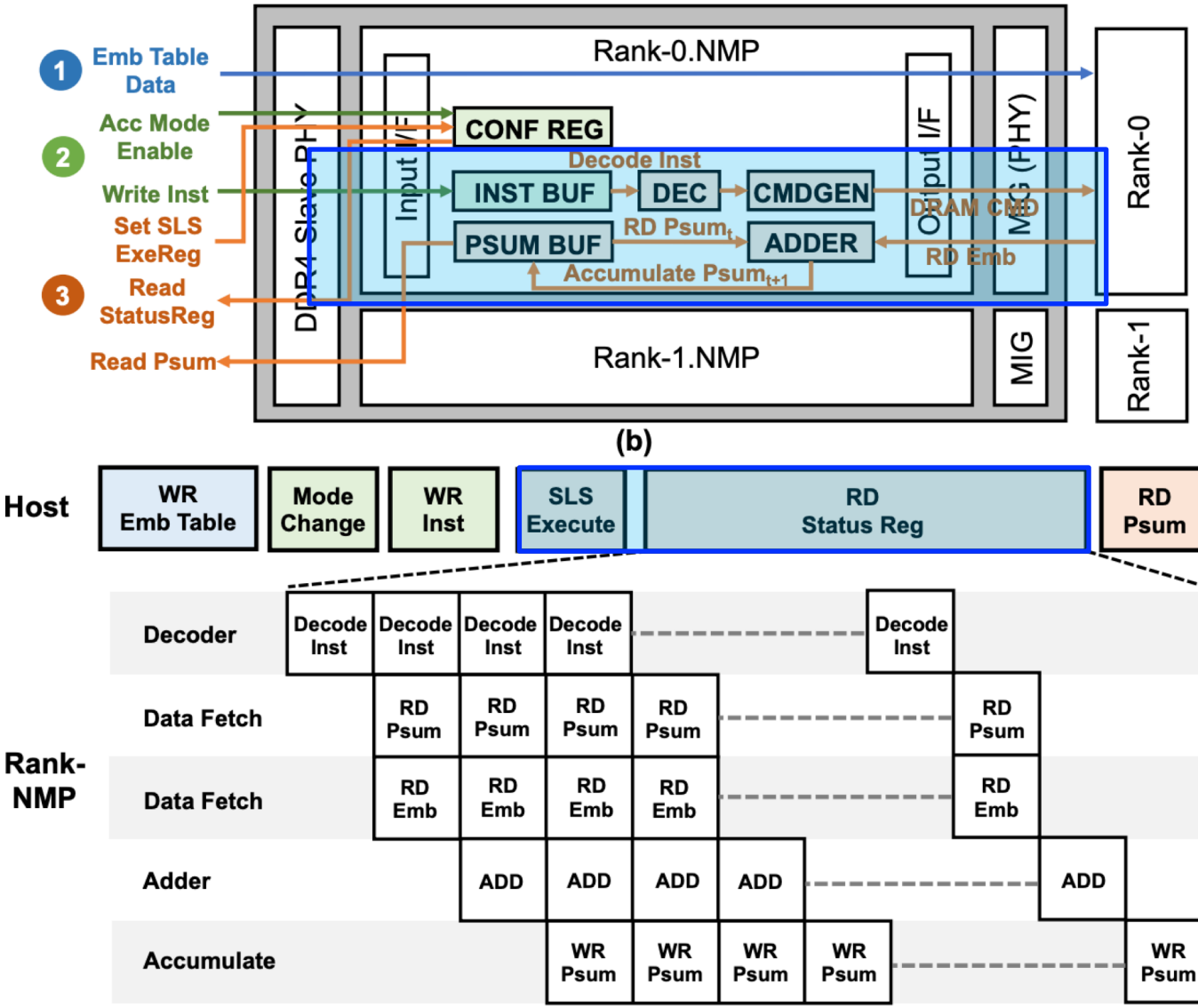
AxDIMM Design: Execution Flow



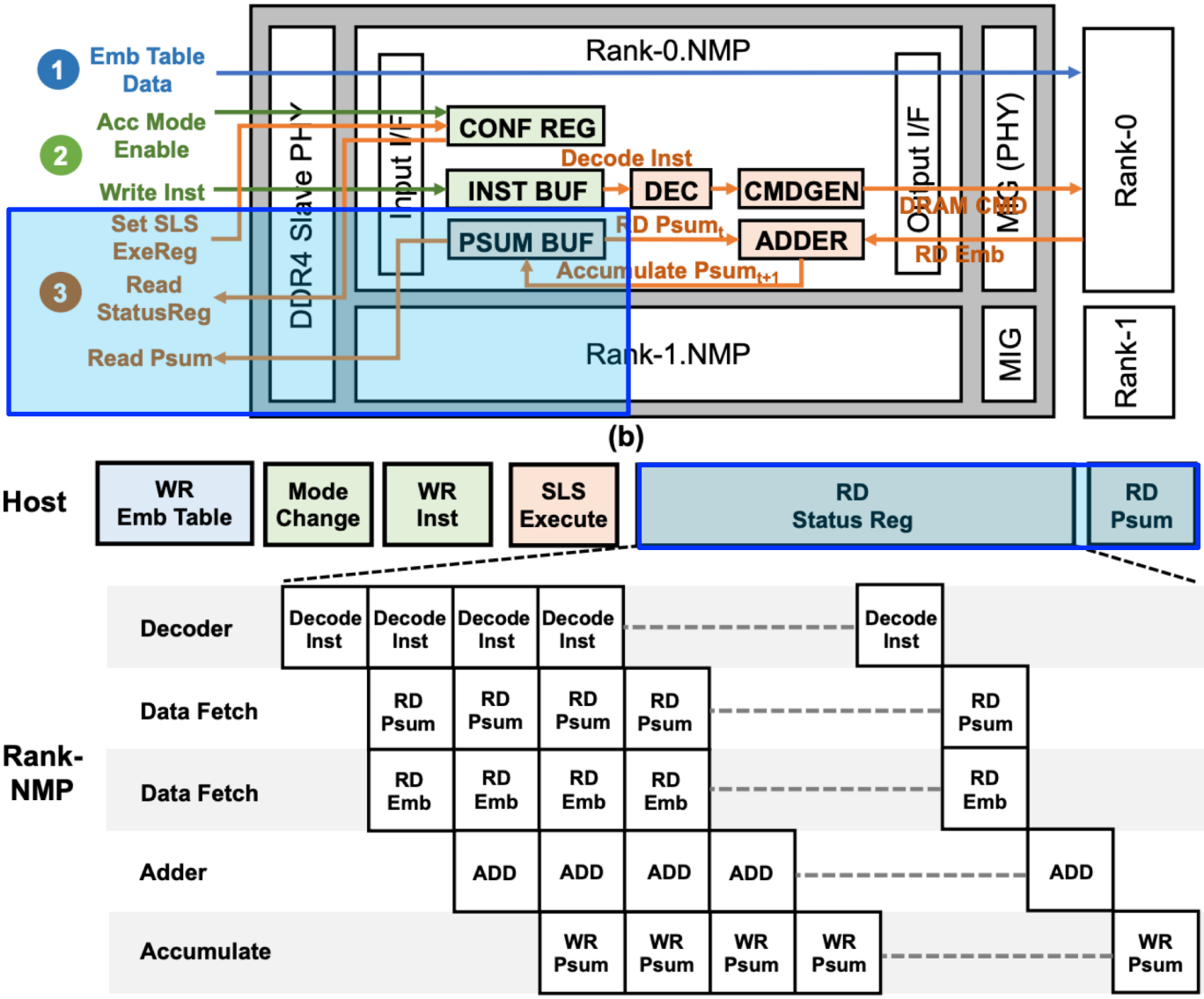
AxDIMM Design: Execution Flow



AxDIMM Design: Execution Flow



AxDIMM Design: Execution Flow



PnM with AxDIMM (IEEE Micro 2021)

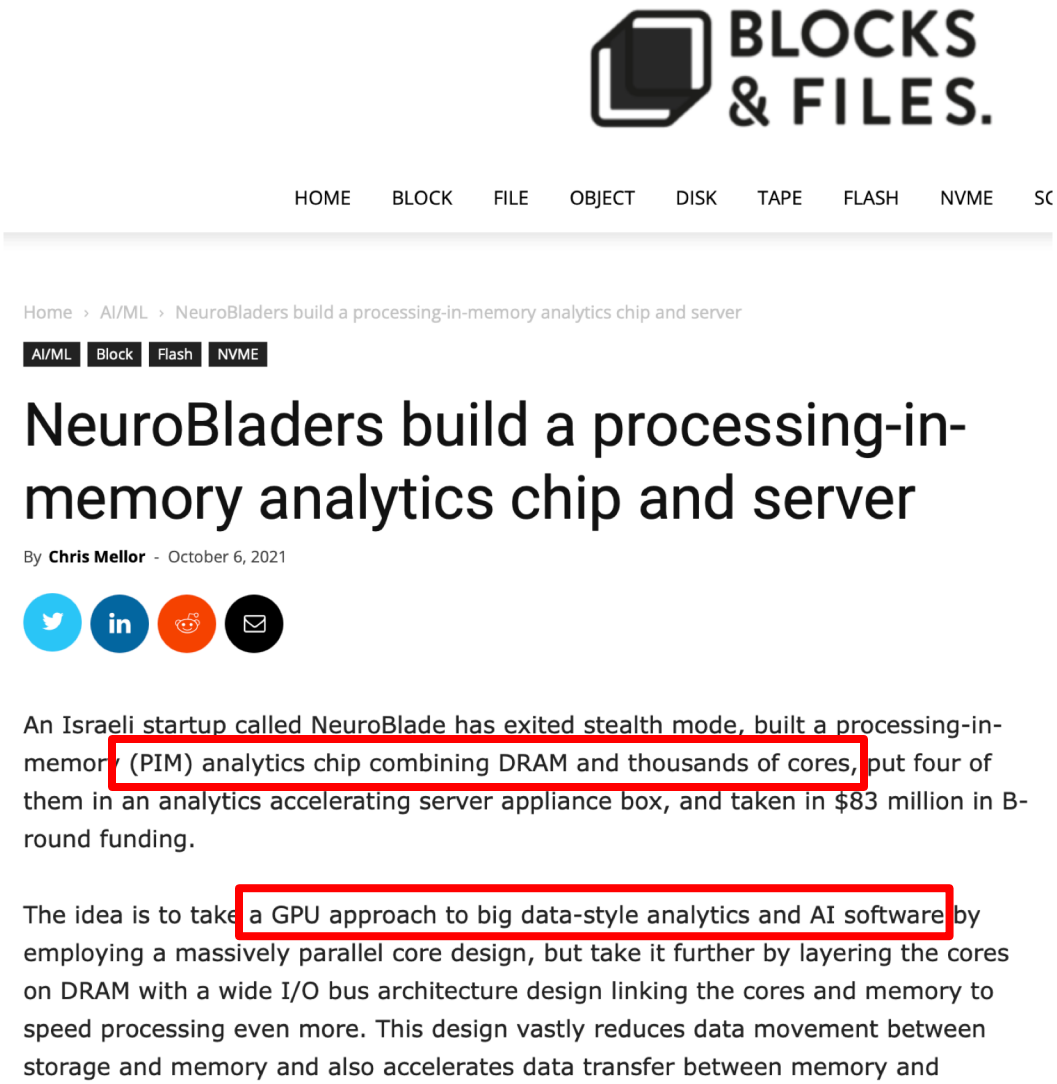
Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM

Liu Ke*†, Xuan Zhang†, Jinin So‡, Jong-Geon Lee‡, Shin-Haeng Kang‡, Sukhan Lee‡, Songyi Han‡, YeonGon Cho‡, JIN Hyun Kim‡, Yongsuk Kwon‡, KyungSoo Kim‡, Jin Jung‡, Ilkwon Yun‡, Sung Joo Park‡, Hyunsun Park‡, Joonho Song‡, Jeonghyeon Cho‡, Kyomin Sohn‡, Nam Sung Kim‡, Hsien-Hsin S. Lee*

*Facebook, †Washington University in St. Louis, ‡Samsung

https://doi.org/10.1109/MM.2021.3097700

More Real-World PIM to Come



BLOCKS & FILES.

HOME BLOCK FILE OBJECT DISK TAPE FLASH NVME SC

Home › AI/ML › NeuroBladers build a processing-in-memory analytics chip and server

AI/ML Block Flash NVME

NeuroBladers build a processing-in-memory analytics chip and server

By **Chris Mellor** - October 6, 2021

An Israeli startup called NeuroBlade has exited stealth mode, built a processing-in-memory (PIM) analytics chip combining DRAM and thousands of cores, put four of them in an analytics accelerating server appliance box, and taken in $83 million in B-round funding.

The idea is to take a GPU approach to big data-style analytics and AI software by employing a massively parallel core design, but take it further by layering the cores on DRAM with a wide I/O bus architecture design linking the cores and memory to speed processing even more. This design vastly reduces data movement between storage and memory and also accelerates data transfer between memory and processing cores.

https://blocksandfiles.com/2021/10/06/neurobladers-build-a-processing-in-memory-analytics-chip-and-server/

NeuroBlade Patent (I)

(12) **United States Patent**
Sity et al.

(10) **Patent No.: US 10,762,034 B2**
(45) **Date of Patent: Sep. 1, 2020**

(54) **MEMORY-BASED DISTRIBUTED PROCESSOR ARCHITECTURE**

(71) Applicant: **NeuroBlade, Ltd.**, Hod-Hashron (IL)

(72) Inventors: **Elad Sity**, Kfar Saba (IL); **Eliad Hillel**, Kfar Saba (IL)

(73) Assignee: **NeuroBlade, Ltd.**, Hod-Hashron (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/512,590**

(22) Filed: **Jul. 16, 2019**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,837,747 A * 6/1989 Dosaka G11C 8/12 365/189.05

5,155,729 A 10/1992 Rysko et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA 2 149 479 C 5/2001

OTHER PUBLICATIONS

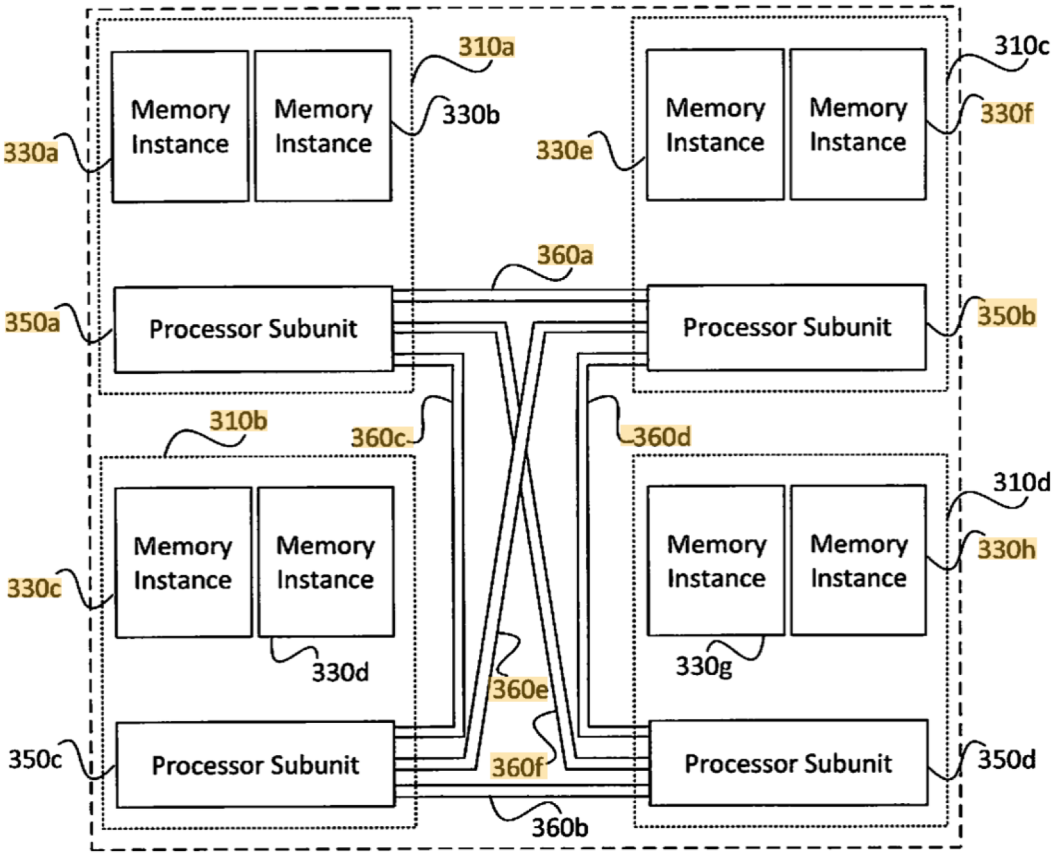
Ahn et al., "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," ISCA '15 (Jun. 13-17, 2015), pp. 105-117.

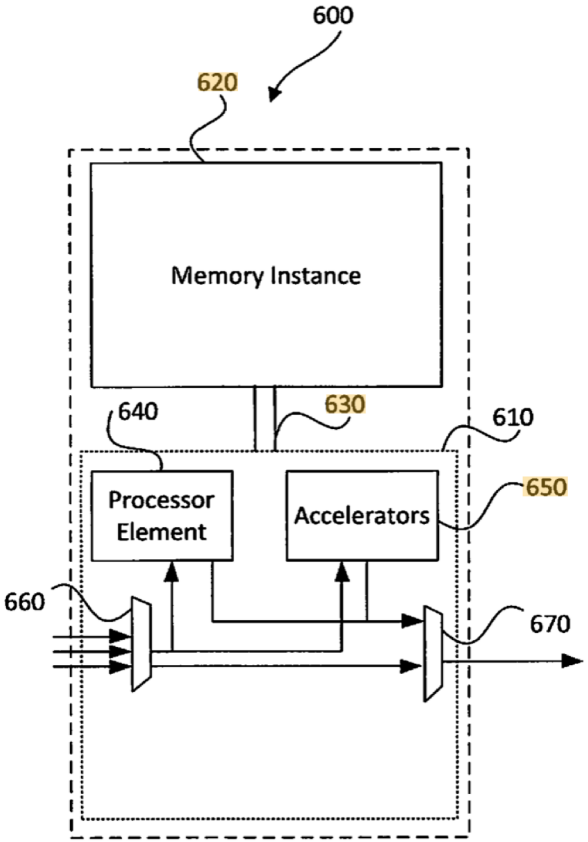
(57) **ABSTRACT**

Distributed processors and methods for compiling code for execution by distributed processors are disclosed. In one implementation, a distributed processor may include a substrate; a memory array disposed on the substrate; and a processing array disposed on the substrate. The memory array may include a plurality of discrete memory banks, and the processing array may include a plurality of processor subunits, each one of the processor subunits being associated with a corresponding, dedicated one of the plurality of discrete memory banks. The distributed processor may further include a first plurality of buses, each connecting one of the plurality of processor subunits to its corresponding, dedicated memory bank, and a second plurality of buses, each connecting one of the plurality of processor subunits to another of the plurality of processor subunits.

Sity et al., "Memory-based Distributed Processor Architecture," US 10,762,034 B2

NeuroBlade Patent (II)





Sity et al., “Memory-based Distributed Processor Architecture,” US 10,762,034 B2

Similarities and Differences among Current PIM Systems

- Similarities
  - Current real-world processing-in-memory architectures follow a processing-near-memory approach

- Differences
  - Near-bank (UPMEM, FIMDRAM) vs. near-chip (AxDIMM)
  - General-purpose (UPMEM) vs. special-function (FIMDRAM)
  - FGMT (UPMEM) vs. SIMD (FIMDRAM, AxDIMM)
  - Natively integer (UPMEM) vs. floating point (FIMDRAM)
    - FP16 (FIMDRAM) vs. FP32 (AxDIMM)
  - DDR4 (UPMEM) vs. HBM2 (FIMDRAM) vs. DDR5 (AxDIMM)

Processing-using-Memory in Real DRAM Chips

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao
feig@princeton.edu
Department of Electrical Engineering
Princeton University

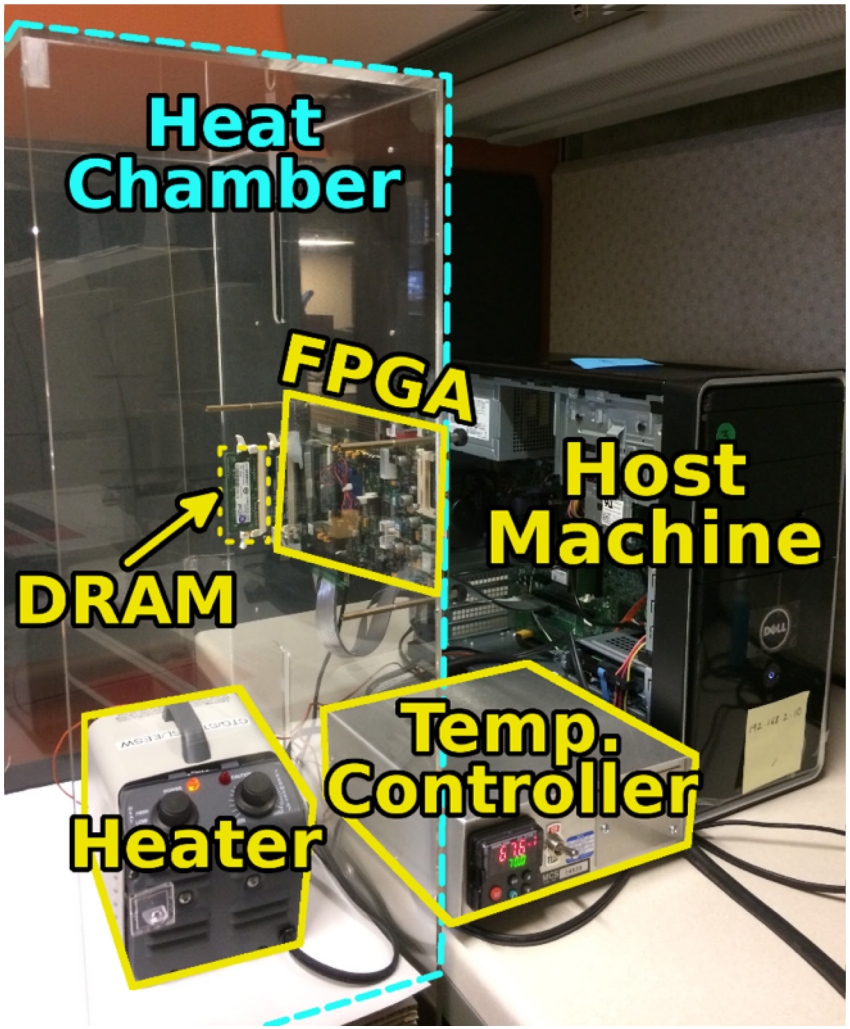
Georgios Tziantzioulis
georgios.tziantzioulis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlaff
wentzlaf@princeton.edu
Department of Electrical Engineering
Princeton University

https://parallel.princeton.edu/papers/micro19-gao.pdf

SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan et al., "**SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies**," HPCA 2017



- Flexible
- Easy to Use (C++ API)
- Open-source
  github.com/CMU-SAFARI/SoftMC

RowClone & Bitwise Ops in Real DRAM Chips

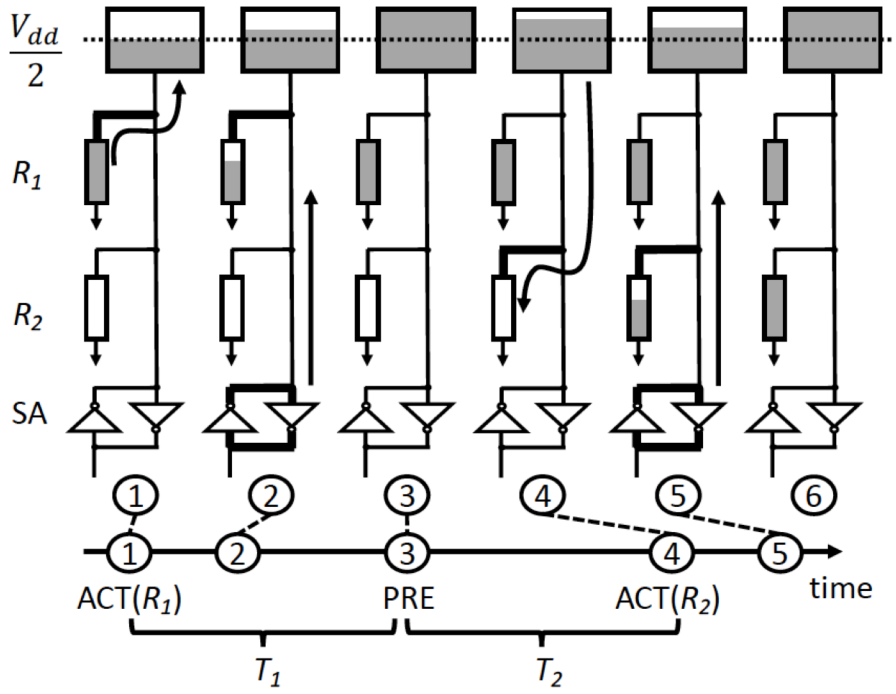


Figure 4: Timeline for a single bit of a column in a row copy operation. The data in R_1 is loaded to the bit-line, and overwrites R_2.

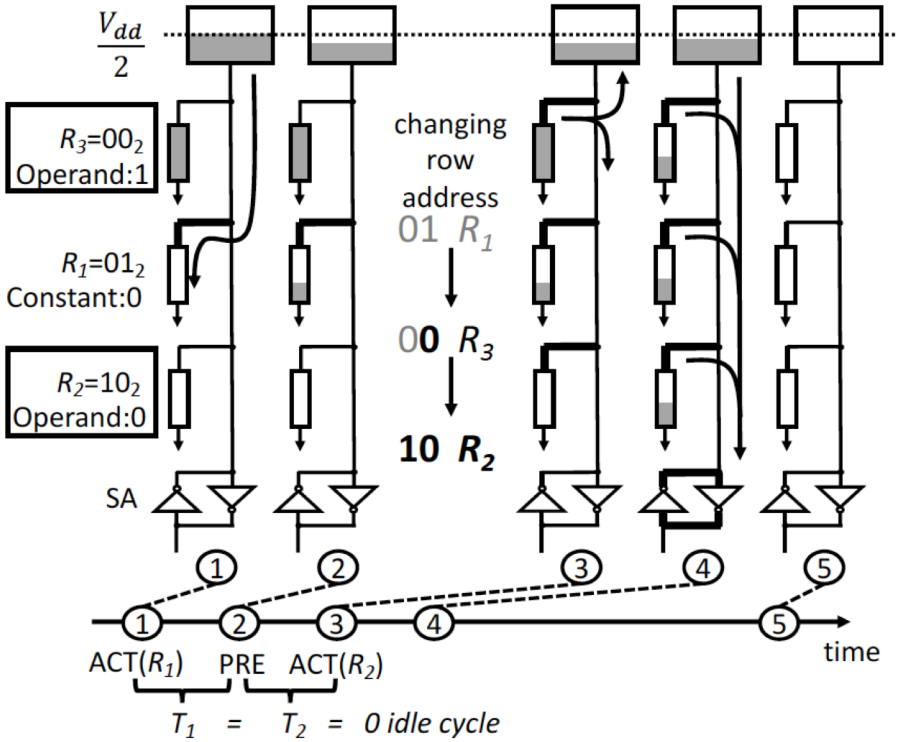
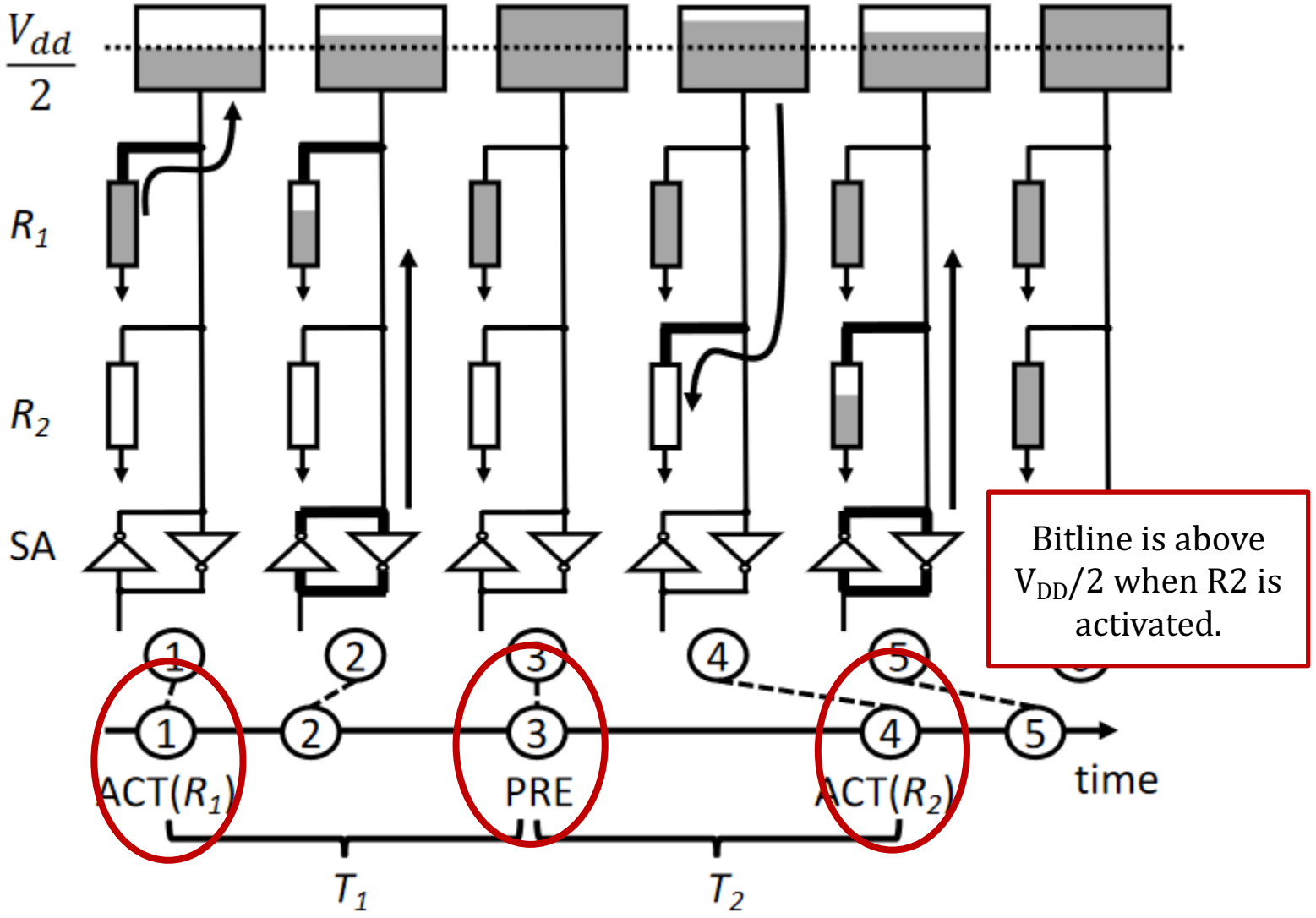
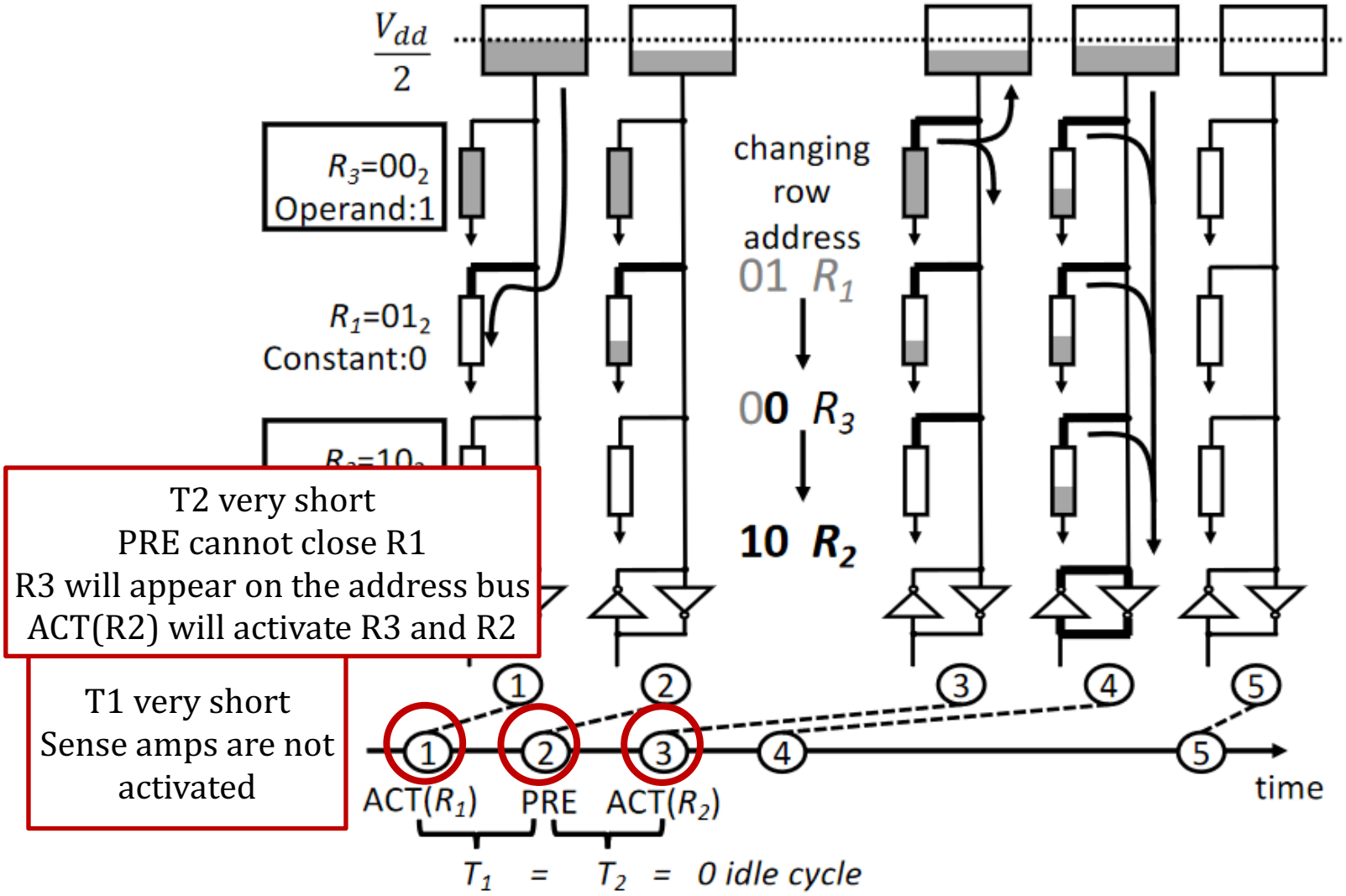


Figure 5: Logical AND in ComputeDRAM. R_1 is loaded with constant zero, and R_2 and R_3 store operands (0 and 1). The result ($0 = 1 \wedge 0$) is finally set in all three rows.

Row Copy in ComputeDRAM



Bitwise AND in ComputeDRAM



Experimental Methodology (I)

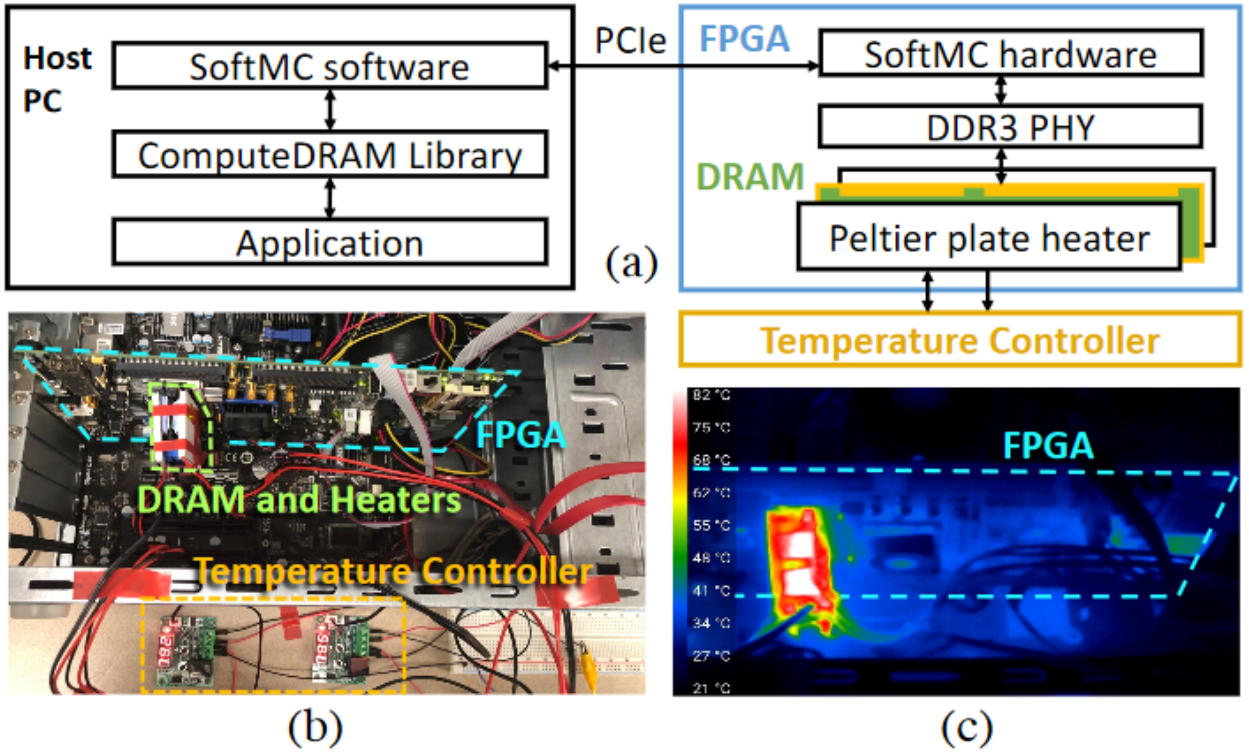


Figure 9: (a) Schematic diagram of our testing framework. (b) Picture of our testbed. (c) Thermal picture when the DRAM is heated to 80 °C.

Experimental Methodology (II)

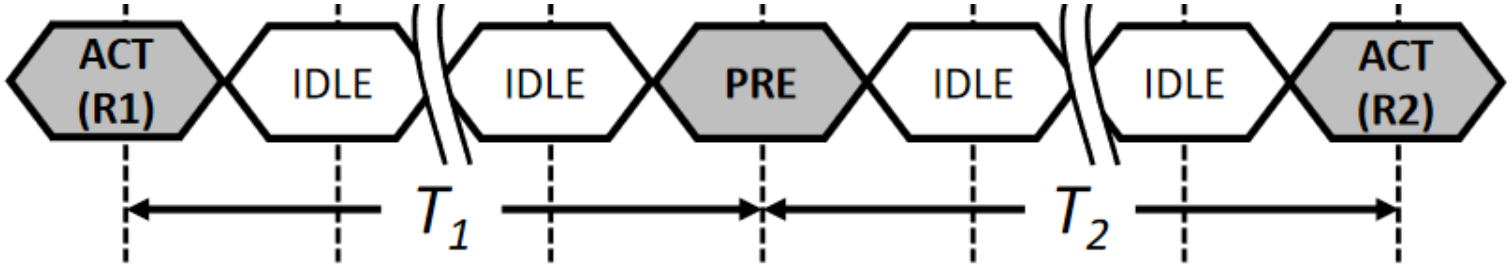
Table 1: Evaluated DRAM modules

Group ID: Vendor_Size_Freq(MHz)	Part Num	# Modules
SKhynix_2G_1333	HMT325S6BFR8C-H9	6
SKhynix_4G_1066	HMT451S6MMR8C-G7	2
SKhynix_4		2
SKhynix_4		4
SKhynix_4		2
Samsung_4		2
Samsung_4		2
Micron_2G		2
Micron_2G		2
Elpida_2G_1333	EBJ21UE8BDS0-DJ-F	2
Nanya_4G_1333	NT4GC64B8HG0NS-CG	2
TimeTec_4G_1333	78AP10NUS2R2-4G	2
Corsair_4G_1333	CMSA8GX3M2A1333C9	2

32 DDR3 Modules
~256 DRAM Chips

Proof of Concept (I)

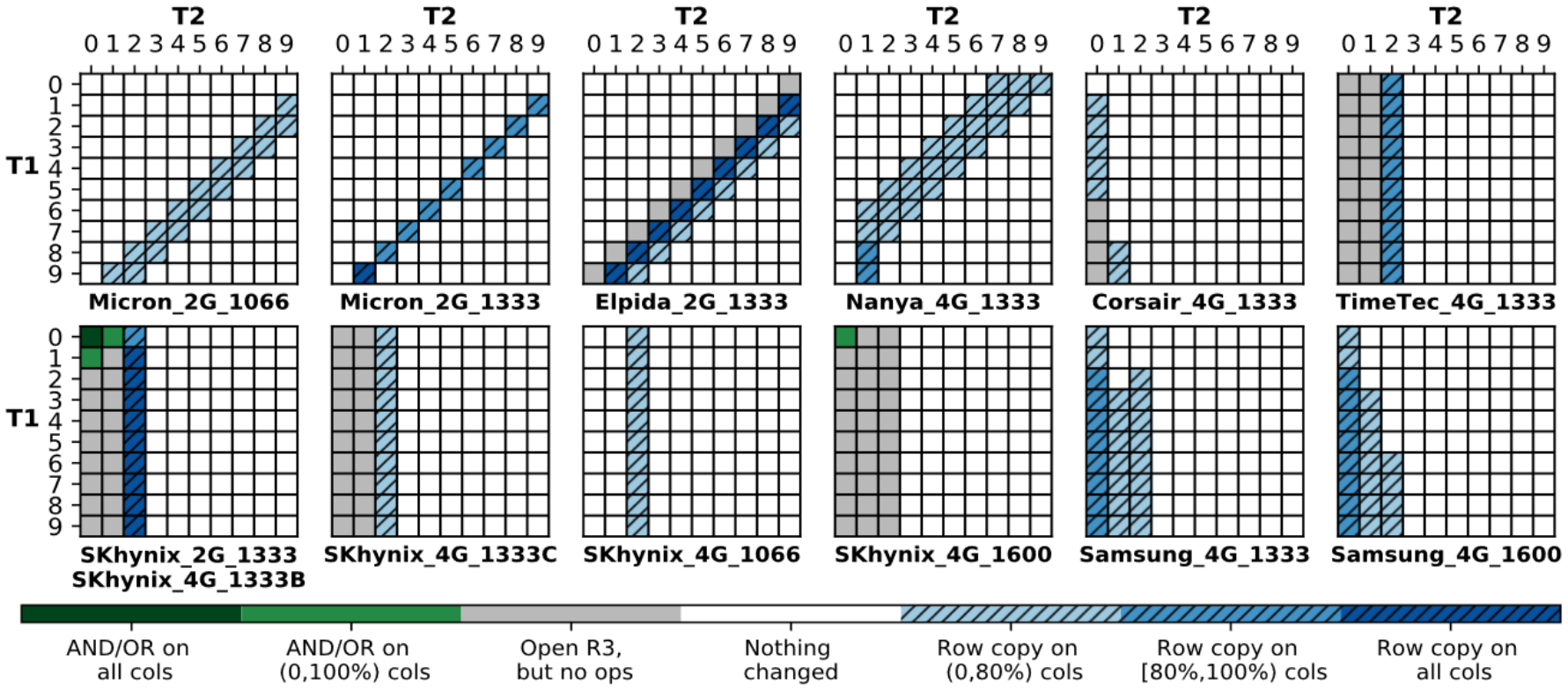
- How they test these memory modules:
  - Vary T_1 and T_2, observe what happens.



SoftMC Experiment

1. Select a random subarray
2. Fill subarray with random data
3. Issue ACT-PRE-ACTs with given T_1 & T_2
4. Read out subarray
5. Find out how many columns in a row support either operation
   - Row-wise success ratio

Proof of Concept (II)



- Each grid represents the success ratio of operations for a specific DDR3 module.

Processing-using-Memory in Real DRAM Chips

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao
feig@princeton.edu
Department of Electrical Engineering
Princeton University

Georgios Tziantzioulis
georgios.tziantzioulis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlaff
wentzlaf@princeton.edu
Department of Electrical Engineering
Princeton University

https://parallel.princeton.edu/papers/micro19-gao.pdf

PnM and PuM Working Synergistically

- Maciej Besta, Raghavendra Kanakagiri, Grzegorz Kwasniewski, Rachata Ausavarungnirun, Jakub Beránek, Konstantinos Kanellopoulos, Kacper Janda, Zur Vonarburg-Shmaria, Lukas Gianinazzi, Ioana Stefan, Juan Gómez-Luna, Marcin Copik, Lukas Kapp-Schwoerer, Salvatore Di Girolamo, Nils Blach, Marek Konieczny, Onur Mutlu, and Torsten Hoefler, **"SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems"**
  *Proceedings of the 54th International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2021. [Older arXiv version]

SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems

Maciej Besta[1], Raghavendra Kanakagiri[2], Grzegorz Kwasniewski[1], Rachata Ausavarungnirun[3], Jakub Beránek[4], Konstantinos Kanellopoulos[1], Kacper Janda[5], Zur Vonarburg-Shmaria[1], Lukas Gianinazzi[1], Ioana Stefan[1], Juan Gómez Luna[1], Marcin Copik[1], Lukas Kapp-Schwoerer[1], Salvatore Di Girolamo[1], Marek Konieczny[5], Onur Mutlu[1], Torsten Hoefler[1]

[1] *ETH Zurich* [2] *IIT Tirupati* [3] *King Mongkut's University of Technology North Bangkok*
[4] *Technical University of Ostrava* [5] *AGH-UST*

Upcoming Lectures

- End-to-end PuM system integration
- Workload characterization for PIM suitability
- Programming an UPMEM-based PIM system

P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures IV

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich

Fall 2021

2 November 2021