

Memory Systems and Memory-Centric Computing

Lecture 3: Memory-Centric Computing II

Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

17 July 2024

HiPEAC ACACES Summer School 2024

SAFARI

ETH zürich

Agenda For Today

- Memory Systems and Memory-Centric Computing
 - July 15-19, 2024
- Topic 1: Memory Trends, Challenges, Opportunities, Basics
- Topic 2: Memory-Centric Computing
- Topic 3: Memory Robustness: RowHammer, RowPress & Beyond
- Topic 4: Machine Learning Driven Memory Systems
- Topic 5 (another course): Architectures for Genomics and ML
- Topic 6 (unlikely): Non-Volatile Memories and Storage
- Topic 7 (unlikely): Memory Latency, Predictability & QoS
- Major Overview Reading:
 - Mutlu et al., “A Modern Primer on Processing in Memory,” Book Chapter on Emerging Computing and Devices, 2022.

Processing in Memory: Two Approaches

1. Processing **using** Memory
2. Processing **near** Memory

A PIM Taxonomy

- **Nature** (of computation)

- **Using**: Use operational properties of memory structures
- **Near**: Add logic close to memory structures

- **Technology**

- Flash, DRAM, SRAM, RRAM, MRAM, FeRAM, PCM, 3D, ...

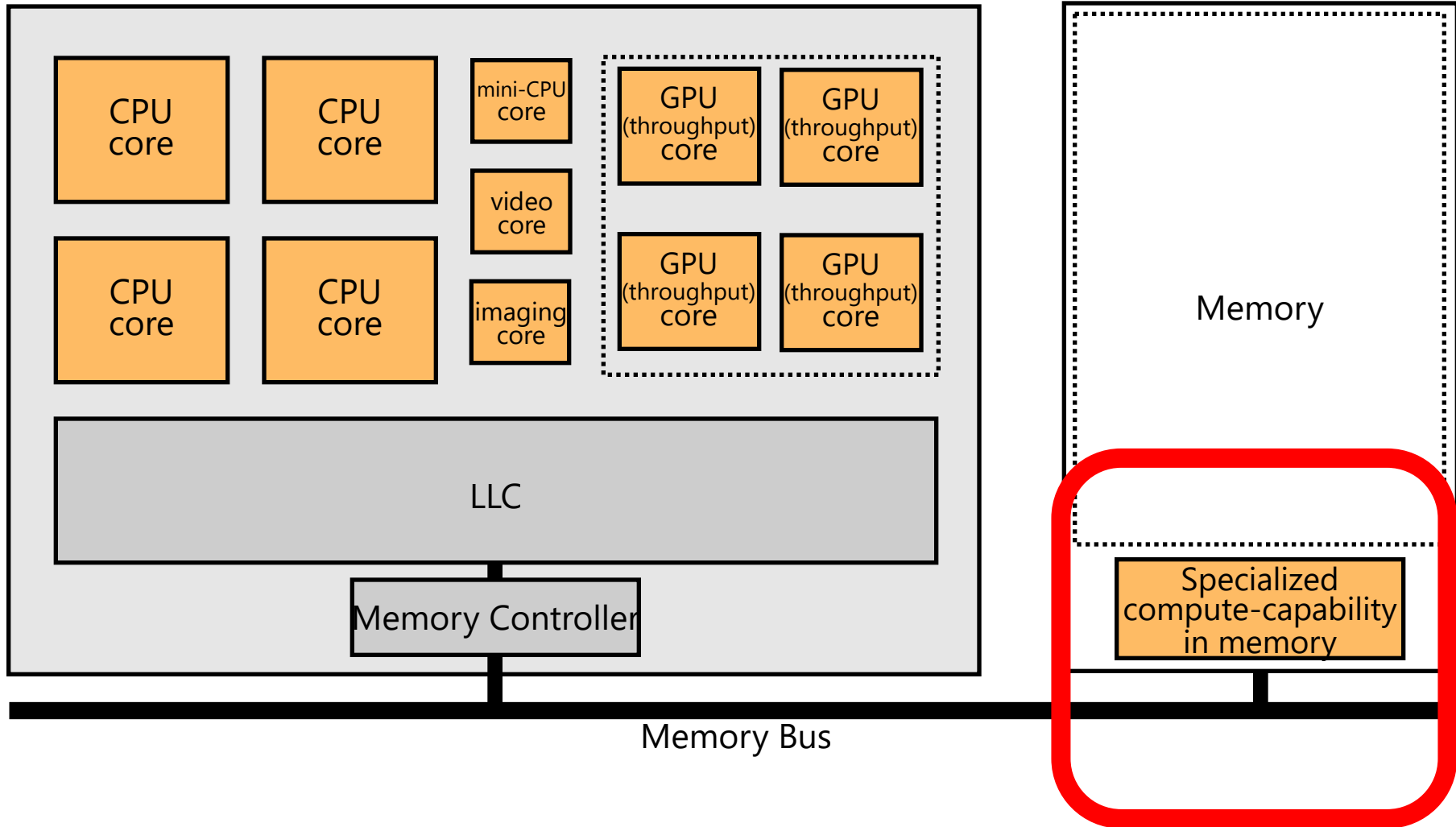
- **Location**

- Sensor, Cold Storage, Hard Disk, SSD, Main Memory, Cache, Register File, Memory Controller, Interconnect, ...

- A tuple of the three determines “PIM type”

- One can combine multiple “PIM types” in a system

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Example PIM Type: Processing using DRAM

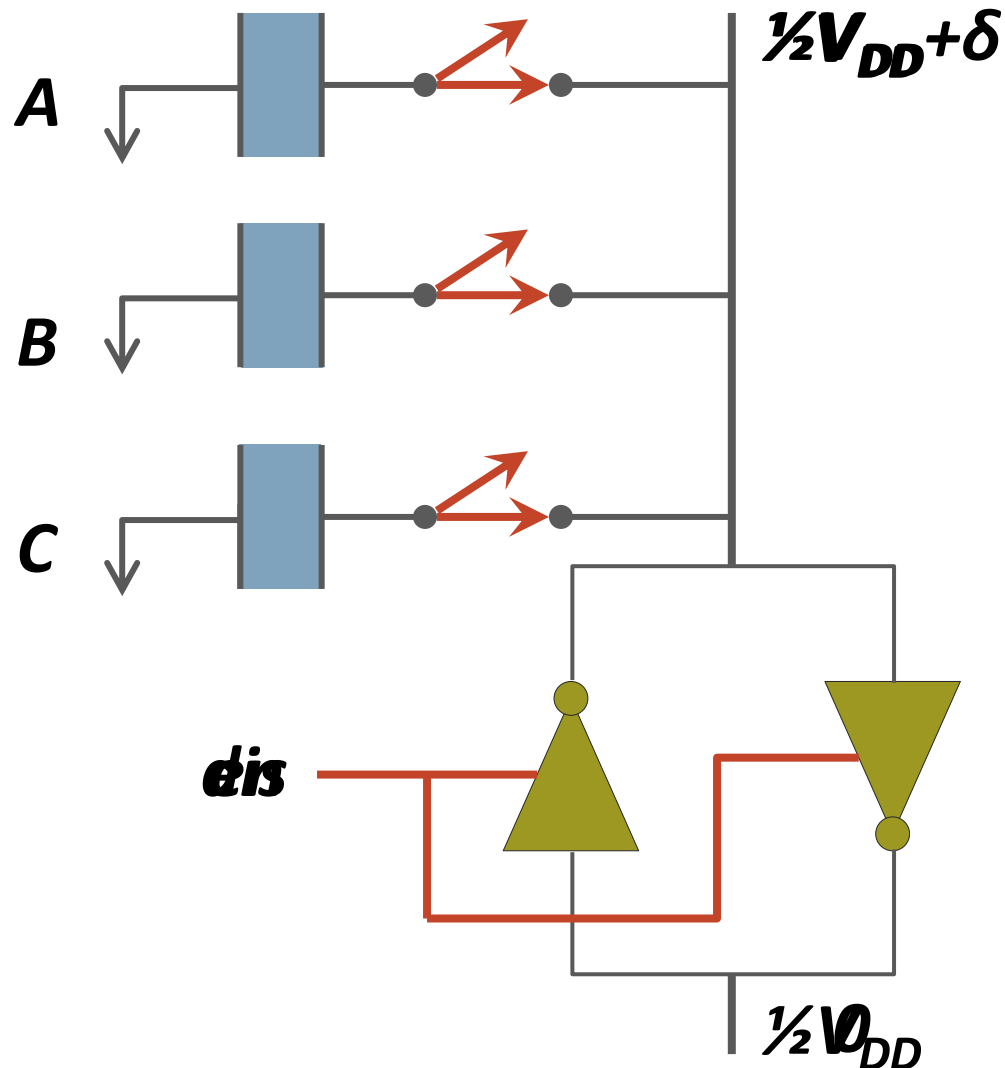
- Nature: Using
- Technology: DRAM
- Location: Main Memory

- Processing using DRAM in Main Memory
 - Seshadri+, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization", MICRO 2013.
 - Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.
 - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
 - Hajinazar+, "SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM," ASPLOS 2021.
 - Oliveira+, "MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Processing," HPCA 2024.

(Truly) In-Memory Computation

- We can support in-DRAM AND, OR, NOT, MAJ
- At low cost
- Using analog computation capability of DRAM
 - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement
 - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
- New memory technologies enable even more opportunities
 - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
 - Can operate on data with minimal movement

In-DRAM AND/OR: Triple Row Activation



Final State
 $AB + BC + AC$

**$C(A + B) +$
 $\sim C(AB)$**

In-DRAM NOT: Dual Contact Cell

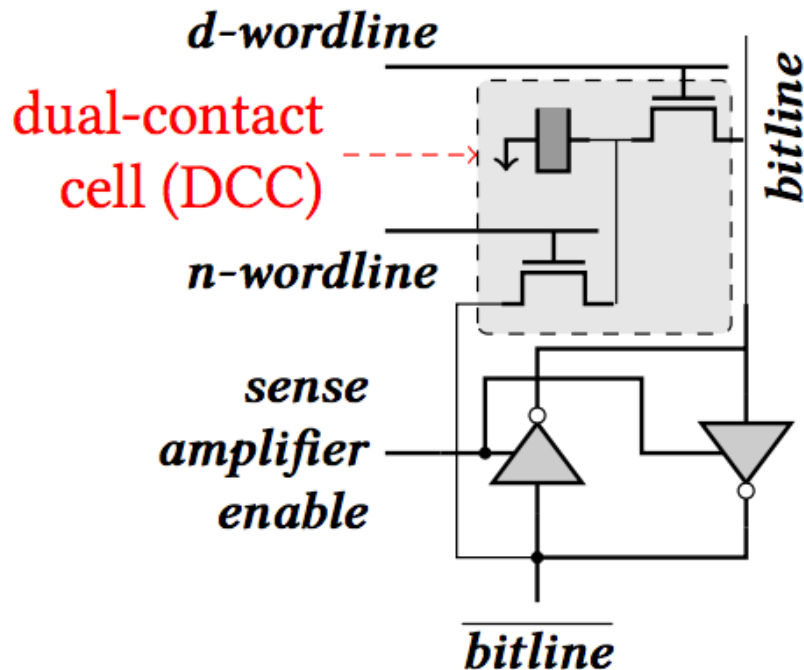


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

Idea:
Feed the
negated value
in the sense amplifier
into a special row

More on Ambit

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[[Slides \(pptx\) \(pdf\)](#)] [[Lightning Session Slides \(pptx\) \(pdf\)](#)] [[Poster \(pptx\) \(pdf\)](#)]

Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri^{1,5} Donghyuk Lee^{2,5} Thomas Mullins^{3,5} Hasan Hassan⁴ Amirali Boroumand⁵
Jeremie Kim^{4,5} Michael A. Kozuch³ Onur Mutlu^{4,5} Phillip B. Gibbons⁵ Todd C. Mowry⁵

¹Microsoft Research India ²NVIDIA Research ³Intel ⁴ETH Zürich ⁵Carnegie Mellon University

In-DRAM Bulk Bitwise Execution

- Vivek Seshadri and Onur Mutlu,
"In-DRAM Bulk Bitwise Execution Engine"
Invited Book Chapter in Advances in Computers, to appear
in 2020.
[Preliminary arXiv version]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch

SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, [**"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"**](#) *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Virtual, March-April 2021.
[[2-page Extended Abstract](#)]
[[Short Talk Slides \(pptx\)](#)] ([pdf](#))
[[Talk Slides \(pptx\)](#)] ([pdf](#))
[[Short Talk Video](#) (5 mins)]
[[Full Talk Video](#) (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

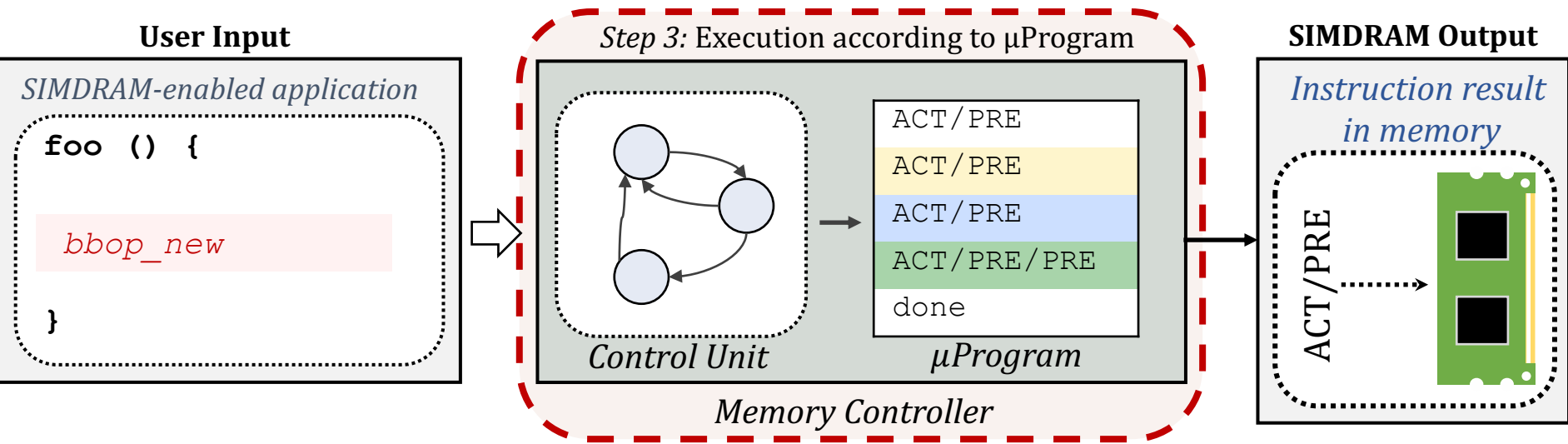
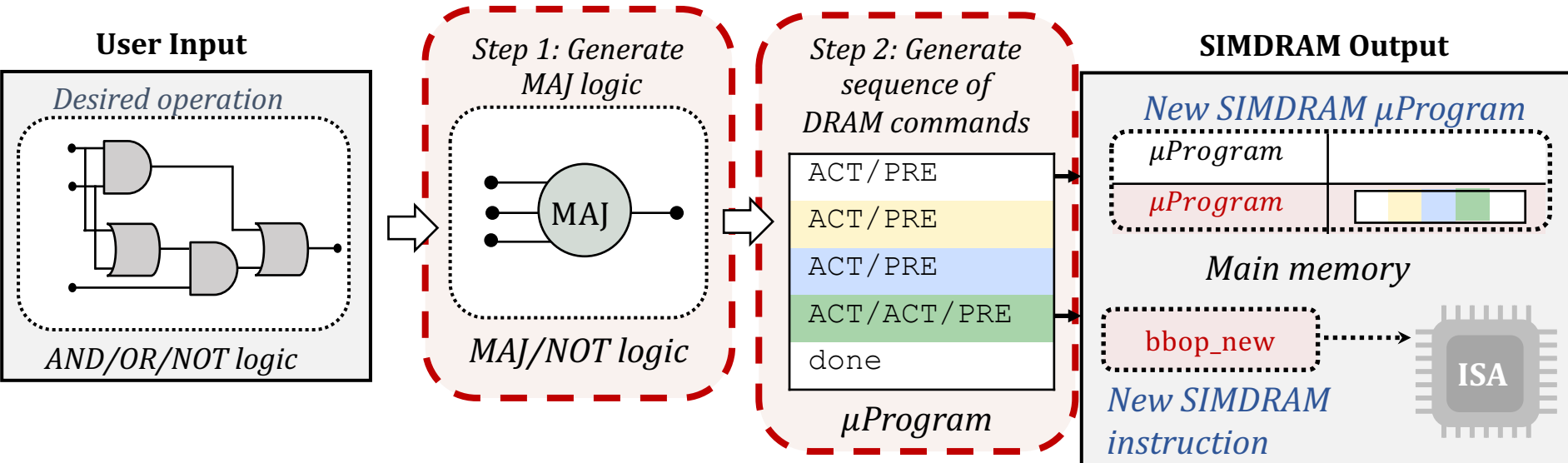
*Nastaran Hajinazar ^{1,2}	*Geraldo F. Oliveira ¹	Sven Gregorio ¹	João Dinis Ferreira ¹
Nika Mansouri Ghiasi ¹	Minesh Patel ¹	Mohammed Alser ¹	Saugata Ghose ³
	Juan Gómez-Luna ¹	Onur Mutlu ¹	

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

SIMDRAM Framework: Overview



SIMDRAM Key Results

Evaluated on:

- 16 complex in-DRAM operations
- 7 commonly-used real-world applications

SIMDRAM provides:

- **88×** and **5.8×** the **throughput** of a **CPU** and a **high-end GPU**, respectively, over **16 operations**
- **257×** and **31×** the **energy efficiency** of a **CPU** and a **high-end GPU**, respectively, over **16 operations**
- **21×** and **2.1×** the **performance** of a **CPU** and a **high-end GPU**, over **seven real-world applications**

SAFARI

More on SIMD RAM

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, [**"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"**](#) *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Virtual, March-April 2021.
[[2-page Extended Abstract](#)]
[[Short Talk Slides \(pptx\)](#) ([pdf](#))]
[[Talk Slides \(pptx\)](#) ([pdf](#))]
[[Short Talk Video](#) (5 mins)]
[[Full Talk Video](#) (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar ^{1,2}	*Geraldo F. Oliveira ¹	Sven Gregorio ¹	João Dinis Ferreira ¹
Nika Mansouri Ghiasi ¹	Minesh Patel ¹	Mohammed Alser ¹	Saugata Ghose ³
	Juan Gómez-Luna ¹	Onur Mutlu ¹	

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

MIMDRAM: More Flexible Processing using DRAM

- **Appears at HPCA 2024** <https://arxiv.org/pdf/2402.19080.pdf>

MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Computing

Geraldo F. Oliveira[†] Ataberk Olgun[†] Abdullah Giray Yağlıkçı[†] F. Nisa Bostancı[†]
Juan Gómez-Luna[†] Saugata Ghose[‡] Onur Mutlu[†]

[†] *ETH Zürich*

[‡] *Univ. of Illinois Urbana-Champaign*

*Our **goal** is to design a flexible PUD system that overcomes the limitations caused by the large and rigid granularity of PUD. To this end, we propose MIMDRAM, a hardware/software co-designed PUD system that introduces new mechanisms to allocate and control only the necessary resources for a given PUD operation. The key idea of MIMDRAM is to leverage fine-grained DRAM (i.e., the ability to independently access smaller segments of a large DRAM row) for PUD computation. MIMDRAM exploits this key idea to enable a multiple-instruction multiple-data (MIMD) execution model in each DRAM subarray (and SIMD execution within each DRAM row segment).*

MIMDRAM: Executive Summary

Problem: Processing-Using-DRAM (PUD) suffers from three issues caused by DRAM's large and rigid access granularity

- Underutilization due to data parallelism variation in (and across) applications
- Limited computation support due to a lack of interconnects
- Challenging programming model due to a lack of compilers

Goal: Design a flexible PUD system that overcomes the three limitations caused by DRAM's large and rigid access granularity

Key Mechanism: MIMDRAM, a hardware/software co-design PUD system

- **Key idea:** leverage fine-grained DRAM for PUD operation
- **HW:** - simple changes to the DRAM array, enabling concurrent PUD operations
 - low-cost interconnects at the DRAM peripherals for data reduction
- **SW:** - compiler and OS support to generate and map PUD instructions

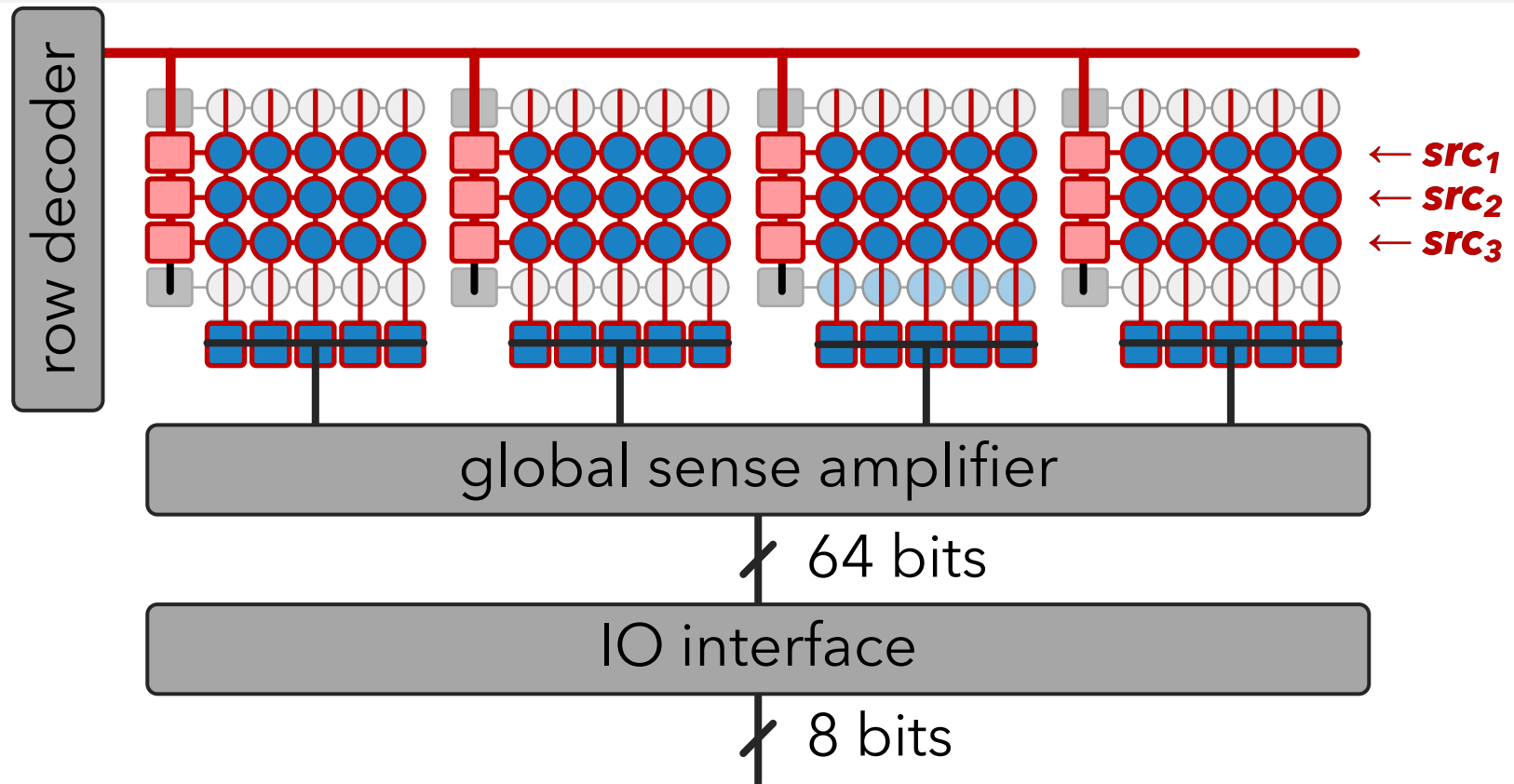
Key Results: MIMDRAM achieves

- **14.3x, 30.6x, and 6.8x** the energy efficiency of state-of-the-art PUD systems, a high-end CPU and GPU, respectively
- Small area cost to a DRAM chip (**1.11%**) and CPU die (**0.6%**)

Background:

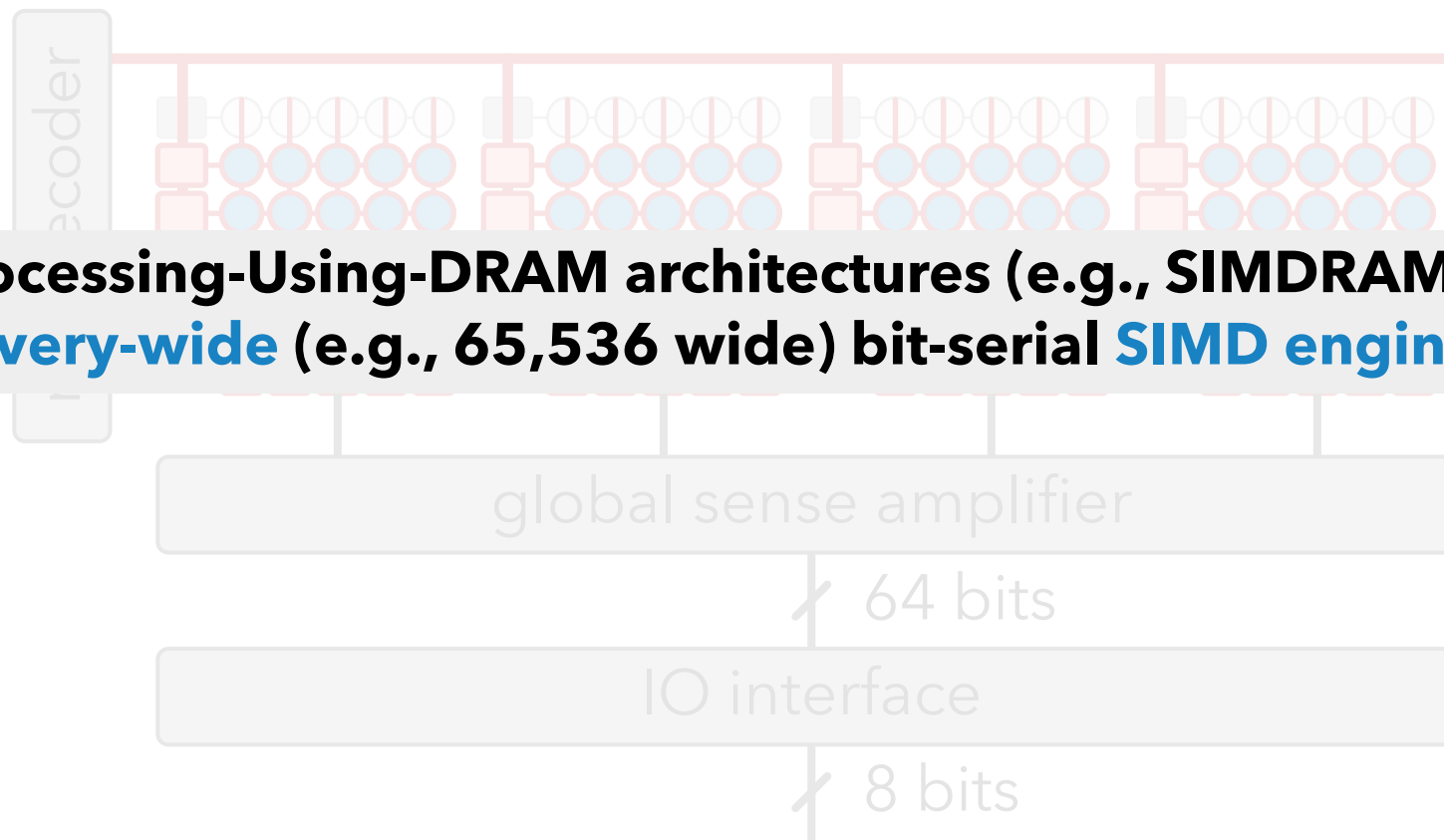
In-DRAM Copy/Init, Majority & NOT Operations

In-DRAM majority is performed by simultaneously activating three DRAM rows



Seshadri, Vivek, et al. " **Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology**," in MICRO, 2017

Background: In-DRAM Majority Operations



Oliveira, Geraldo F., et al. "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM," in ASPLOS, 2021

Limitations of PUD Systems: Overview

PUD systems suffer from **three sources of inefficiency due to the **large** and **rigid** DRAM access granularity**

1 SIMD Underutilization

- due to data parallelism variation within and across applications
- leads to throughput and energy waste

2 Limited Computation Support

- due to a lack of low-cost interconnects across columns
- limits PUD operations to only parallel map constructs

3 Challenging Programming Model

- due to a lack of compiler support for PUD systems
- creates a burden on programmers, limiting PUD adoption

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align
data structures

Just write
my kernel

High-level code for

$C[i] = (A[i] > \text{pred}[i]) ? A[i] + B[i] : A[i] - B[i]$

```
for (int i = 0; i < size ; ++ i){  
    bool cond = A[i] > pred[i];  
    if (cond) C[i] = A[i] + B[i];  
    else C[i] = A[i] - B[i];  
}
```

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align
data structures

Identify
array boundaries

Just write
my kernel

High-level code for

$C[i] = (A[i] > \text{pred}[i]) ? A[i] + B[i] : A[i] - B[i]$

```
for (int i = 0; i < size ; ++ i){  
    bool cond = A[i] > pred[i];  
    if (cond) C[i] = A[i] + B[i];  
    else  C[i] = A[i] - B[i];  
}
```

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align
data structures

Identify
array boundaries

Manually
unroll loop

Map C to
PUD instructions

Just write
my kernel

High-level code for

$C[i] = (A[i] > \text{pred}[i]) ? A[i] + B[i] : A[i] - B[i]$

```
for (int i = 0; i < size ; ++ i){  
    bool cond = A[i] > pred[i];  
    if (cond) C[i] = A[i] + B[i];  
    else  C[i] = A[i] - B[i];  
}
```

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align
data structures

Identify
array boundaries

Manually
unroll loop

Map C to
PUD instructions

Orchestrate
data movement

Just write
my kernel

High-level code for

$C[i] = (A[i] > \text{pred}[i]) ? A[i] + B[i] : A[i] - B[i]$

```
for (int i = 0; i < size ; ++ i){  
    bool cond = A[i] > pred[i];  
    if (cond) C[i] = A[i] + B[i];  
    else  C[i] = A[i] - B[i];  
}
```

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align
data structures

Identify
array boundaries

Manually
unroll loop

Map C to
PUD instructions

Orchestrate
data movement

Just write
my kernel

PUD's assembly-like code for

$C[i] = (A[i] > \text{pred}[i]) ? A[i] + B[i] : A[i] - B[i]$

```
bbop_trsp_init(A , size , elm_size);
bbop_trsp_init(B , size , elm_size);
bbop_trsp_init(C , size , elm_size);

bbop_add(D , A , B , size , elm_size);
bbop_sub(E , A , B , size , elm_size);
bbop_greater(F , A , pred , size , elm_size);
bbop_if_else(C , D , E , F , size , elm_size);
```

Problem & Goal

Problem

Processing-Using-DRAM's large and rigid granularity limits its applicability and efficiency for different applications

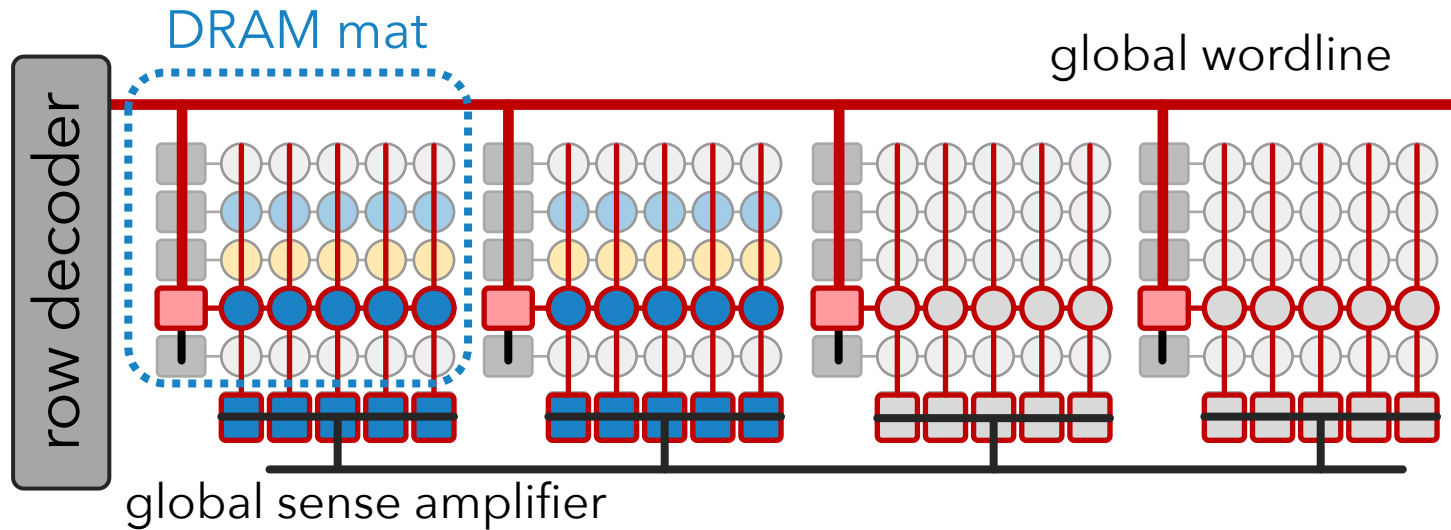
Goal

Design a flexible PUD system that overcomes the three limitations caused by large and rigid DRAM access granularity

MIMDRAM:

Key Idea (I)

DRAM's hierarchical organization can enable fine-grained access



Key Issue:

on a DRAM access, the global wordline propagates across all DRAM mats



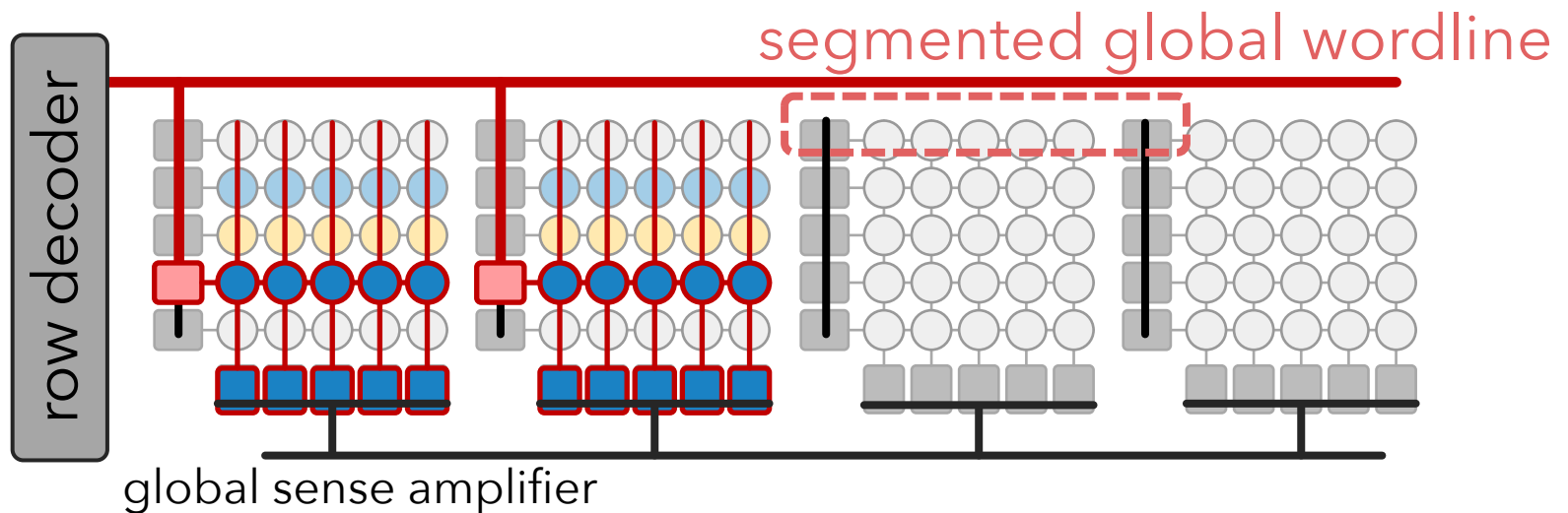
Fine-Grained DRAM:

segments the global wordline to access **individual** DRAM mats

MIMDRAM: Key Idea (II)

Fine-Grained DRAM:

segments the global wordline to access **individual** DRAM mats



Fine-grained DRAM for energy-efficient DRAM access:

[Cooper-Balis+, 2010]: Fine-Grained Activation for Power Reduction in DRAM

[Udipi+, 2010]: Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores

[Zhang+, 2014]: Half-DRAM

[Ha+, 2016]: Improving Energy Efficiency of DRAM by Exploiting Half Page Row Access

[O'Connor+, 2017]: Fine-Grained DRAM

[Olgun+, 2024]: Sectored DRAM

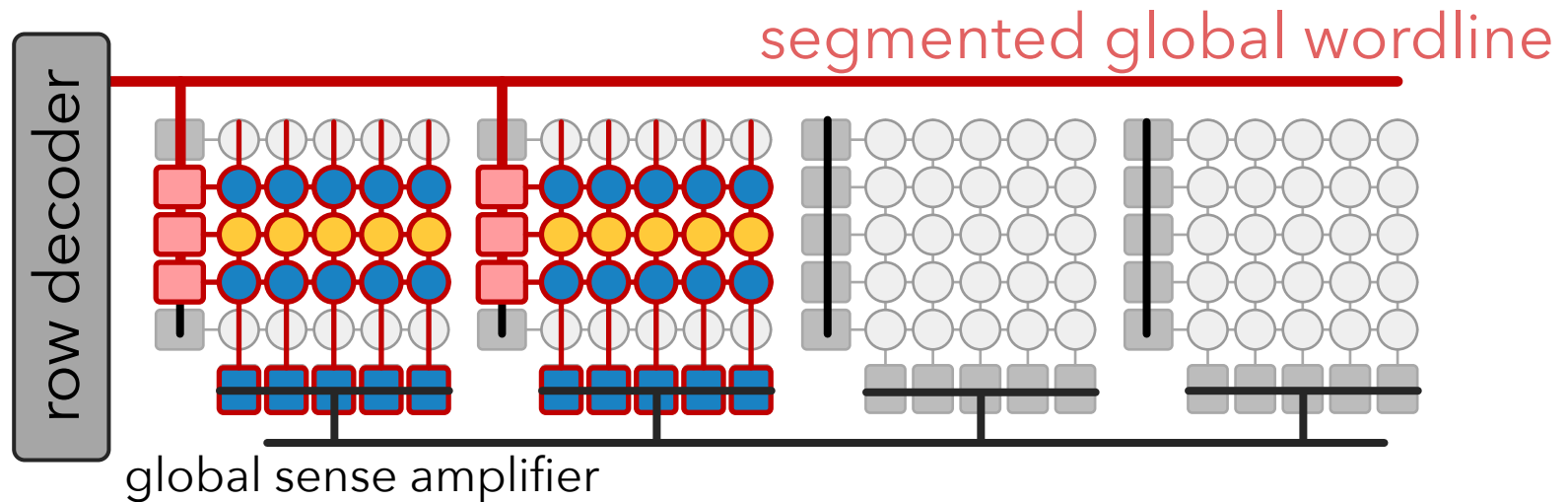
Sectored DRAM

- Ataberk Olgun, F. Nisa Bostanci, Geraldo F. Oliveira, Yahya Can Tugrul, Rahul Bera, A. Giray Yaglikci, Hasan Hassan, Oguz Ergin, and Onur Mutlu, **"Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture"**
ACM Transactions on Architecture and Code Optimization (TACO),
[online] June 2024.
[[arXiv version](#)]
[[ACM Digital Library version](#)]

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun[§] F. Nisa Bostanci^{§†} Geraldo F. Oliveira[§] Yahya Can Tuğrul^{§†} Rahul Bera[§]
A. Giray Yağlıkcı[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

MIMDRAM: Key Idea (III)

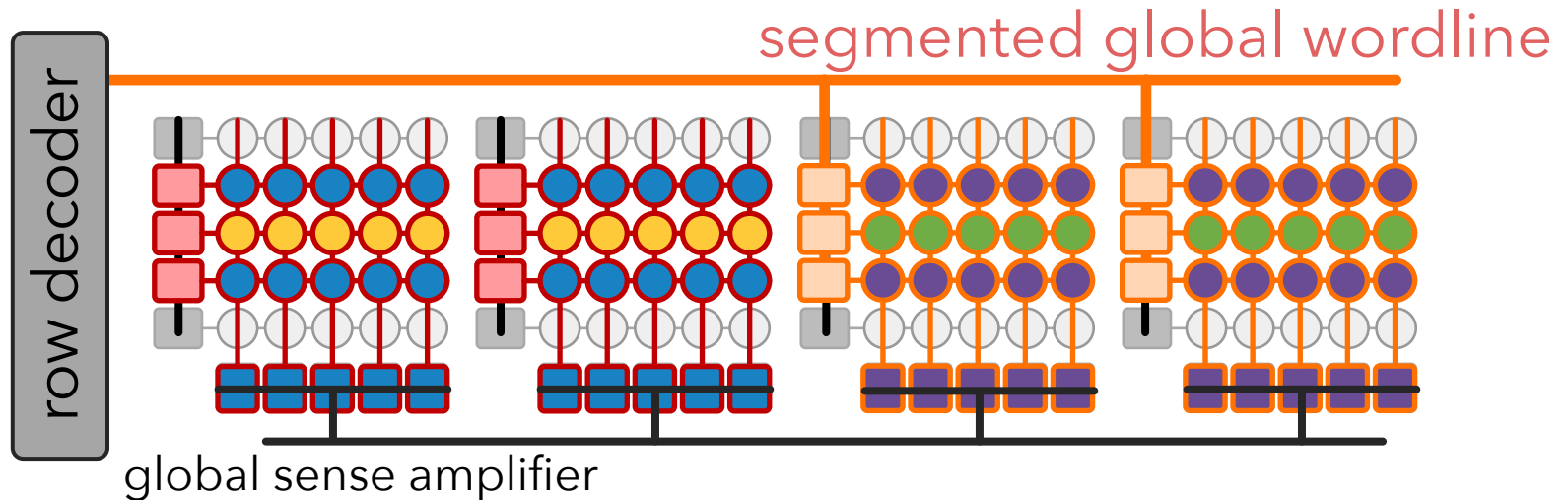


Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data

MIMDRAM: Key Idea (III)

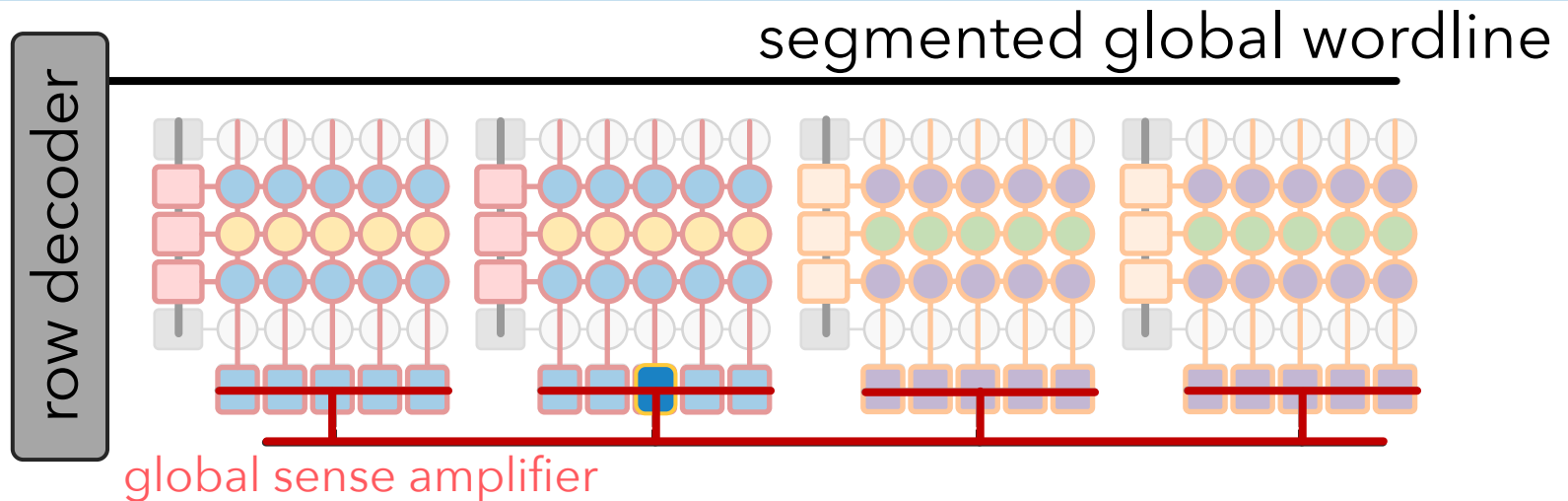


Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data
 - for multiple PUD operations, execute independent operations concurrently
- **multiple instruction, multiple data (MIMD) execution model**

MIMDRAM: Key Idea (III)



Fine-grained DRAM for processing-using-DRAM:

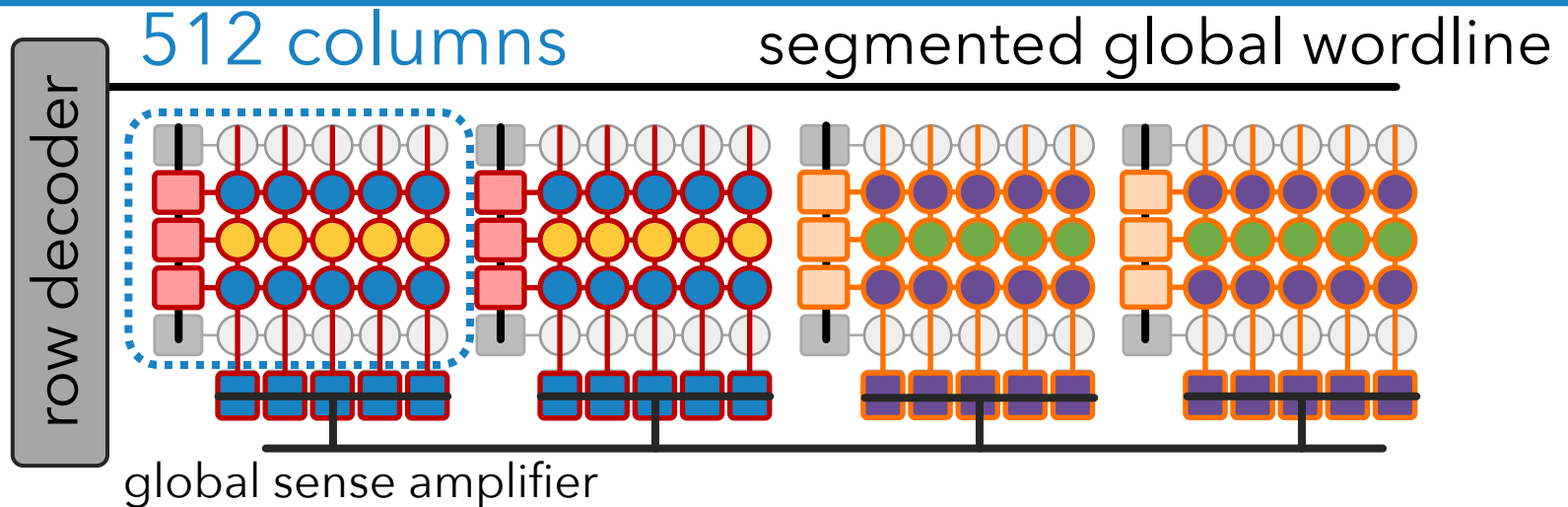
1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data
- for multiple PUD operations, execute independent operations concurrently
→ **multiple instruction, multiple data (MIMD) execution model**

2 Enables low-cost interconnects for vector reduction

- global and local data buses can be used for inter-/intra-mat communication

MIMDRAM: Key Idea (III)



Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data
- for multiple PUD operations, execute independent operations concurrently
→ **multiple instruction, multiple data (MIMD) execution model**

2 Enables low-cost interconnects for vector reduction

- global and local data buses can be used for inter-/intra-mat communication

3 Eases programmability

- SIMD parallelism in a DRAM mat is on par with vector ISAs' SIMD width

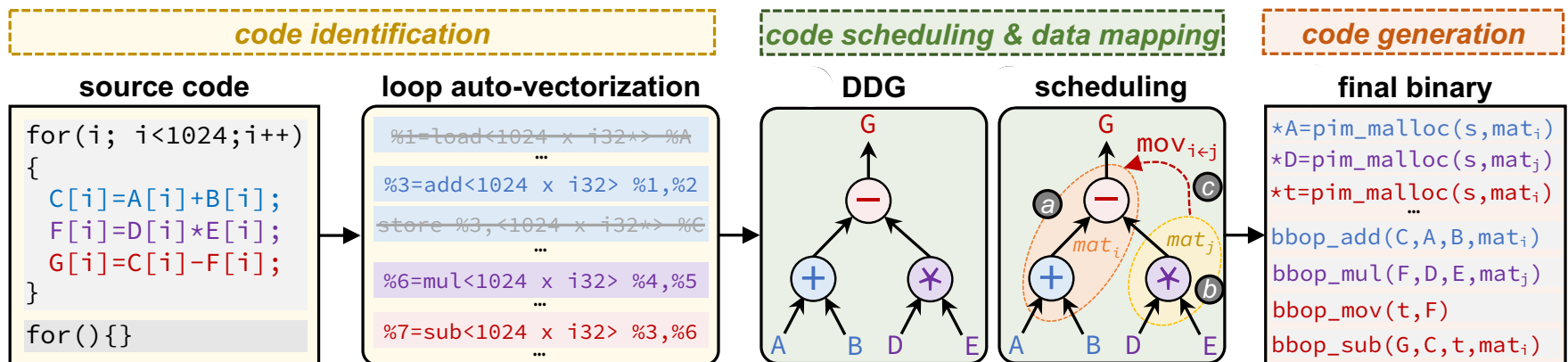
MIMDRAM: Compiler Support (I)

Goal

Transparently:
extract SIMD parallelism from an application, and
schedule PUD instructions while maximizing utilization



Three new LLVM-based passes targeting PUD execution



MIMDRAM: Compiler Support (II)

code identification

source code

```
for(i; i<1024;i++)  
{  
  C[i]=A[i]+B[i];  
  F[i]=D[i]*E[i];  
  G[i]=C[i]-F[i];  
}  
for(){}  

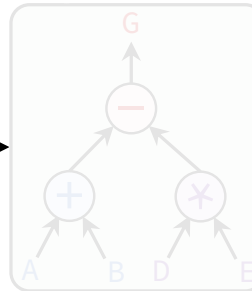
```

loop auto-vectorization

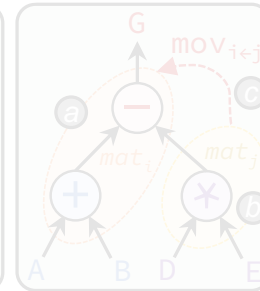
```
%1=load<1024 x i32*> %A  
...  
%3=add<1024 x i32> %1,%2  
store %3,<1024 x i32*> %C  
...  
%6=mul<1024 x i32> %4,%5  
...  
%7=sub<1024 x i32> %3,%6  
...
```

code scheduling & data mapping

DDG



scheduling



code generation

final binary

```
*A=pim_malloc(s,mat_i)  
*D=pim_malloc(s,mat_j)  
*t=pim_malloc(s,mat_i)  
...  
bbop_add(C,A,B,mat_i)  
bbop_mul(F,D,E,mat_j)  
bbop_mov(t,F)  
bbop_sub(G,C,t,mat_i)
```

Goal

**Identify SIMD parallelism, generate PUD instructions,
and set the appropriate vectorization factor**

MIMDRAM: Compiler Support (II)

code identification

source code

```
for(i; i<1024;i++)
{
  C[i]=A[i]+B[i];
  F[i]=D[i]*E[i];
  G[i]=C[i]-F[i];
}
for(){}

```

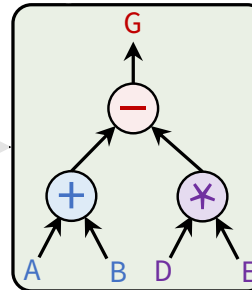
loop auto-vectorization

```
%1=load<1024 x i32> %A
...
%3=add<1024 x i32> %1,%2
store %3,<1024 x i32> %C
...
%6=mul<1024 x i32> %4,%5
...
%7=sub<1024 x i32> %3,%6
...

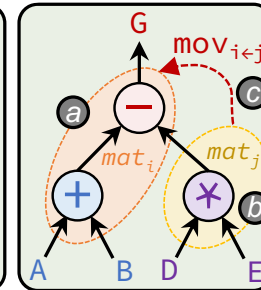
```

code scheduling & data mapping

DDG



scheduling



code generation

final binary

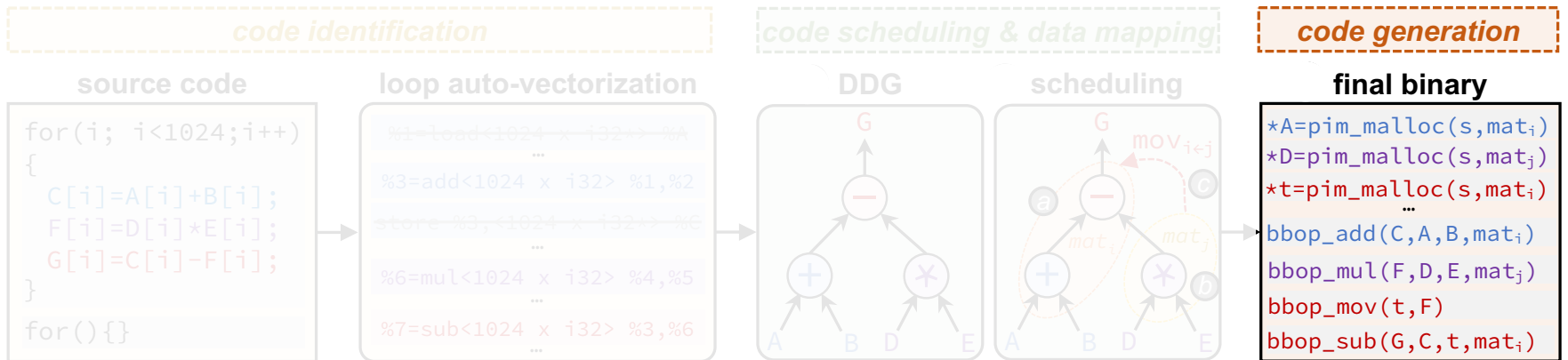
```
*A=pim_malloc(s,mat_i)
*D=pim_malloc(s,mat_j)
*t=pim_malloc(s,mat_i)
...
bbop_add(C,A,B,mat_i)
bbop_mul(F,D,E,mat_j)
bbop_mov(t,F)
bbop_sub(G,C,t,mat_i)

```

Goal Identify SIMD parallelism, generate PUD instructions, and set the appropriate vectorization factor

Goal Improve SIMD utilization by allowing the distribution of independent PUD instructions across DRAM mats

MIMDRAM: Compiler Support (III)



Goal Identify SIMD parallelism, generate PUD instructions, and set the appropriate vectorization factor

Goal Improve SIMD utilization by allowing the distribution of independent PUD instructions across DRAM mats

Goal Generate the appropriate binary for data allocation and PUD instructions

MIMDRAM: System Support

- Instruction set architecture
- Execution & data transposition
- Data coherence
- Address translation
- Data allocation & alignment
- Mat label translation

Evaluation: Methodology Overview

- **Evaluation Setup**

- **CPU:** Intel Skylake CPU
- **GPU:** NVIDIA A100 GPU
- **PUD:** SIMDRAM [Oliveira+, 2021] and DRISA [Li+, 2017]
- **PND:** Fulcrum [Lenjani+, 2020]
- <https://github.com/CMU-SAFARI/MIMDRAM>

- **Workloads:**

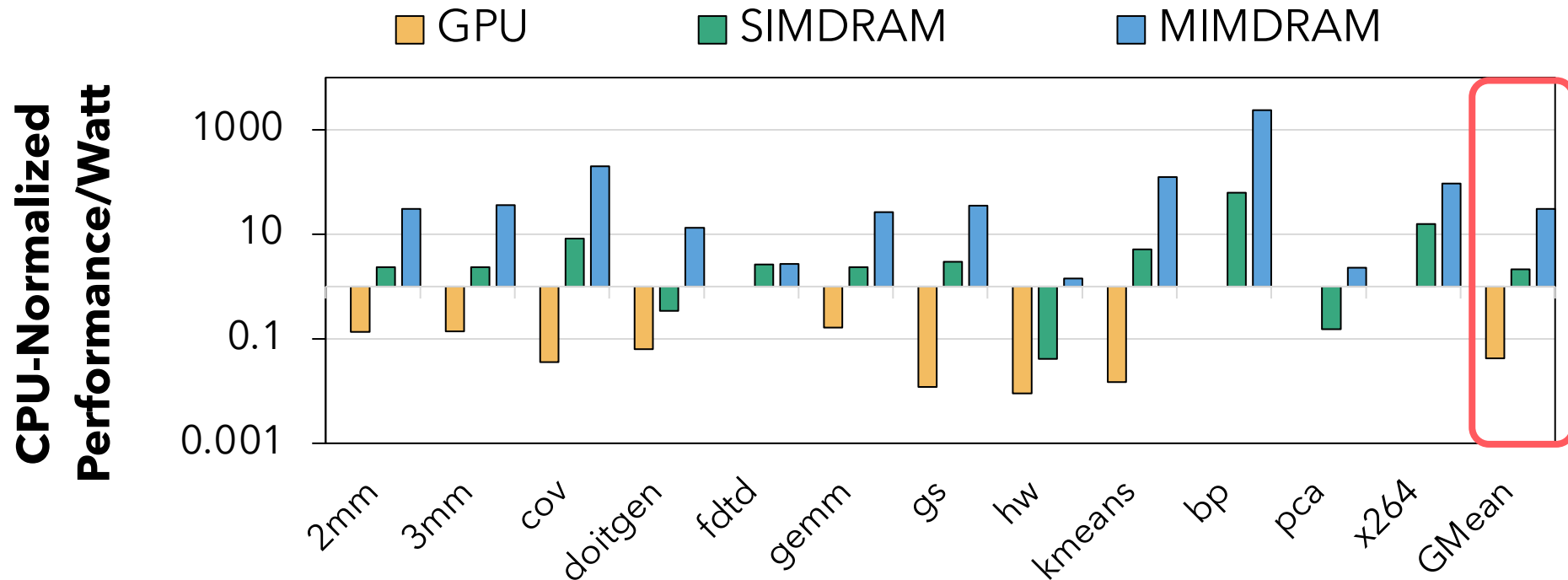
- 12 workloads from Polybench, Rodinia, Phoenix, and SPEC2017
- 495 multi-programmed application mixes

- **Two-Level Analysis**

- **Single application** → leverages intra-application data parallelism
- **Multi-programmed workload** → leverages inter-application data parallelism

Evaluation:

Single Application Analysis - Energy Efficiency

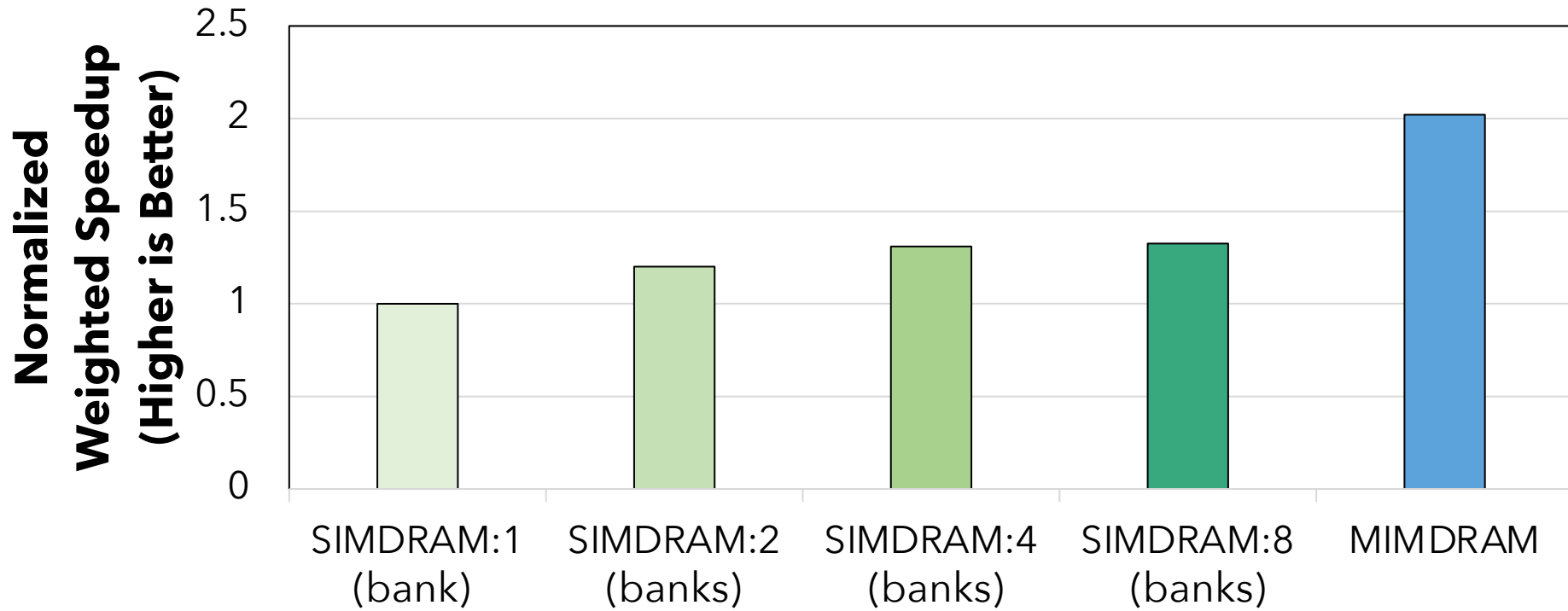


Takeaway

MIMDRAM significantly improves energy efficiency compared to CPU (30.6x), GPU (6.8x), and SIMD (14.3x)

Evaluation:

Multi-Programmed Workload Analysis



Takeaway

**MIMDRAM significantly improves
system throughput (1.68x)
compared to SIMDRAM**

Evaluation:

More in the Paper

- **MIMDRAM with subarray and bank-level parallelism**
 - MIMDRAM provides significant performance gains compared to the baseline CPU (**13.2x**) and GPU (**2x**)
- **Comparison to DRISA and Fulcrum for multi-programmed workloads**
 - MIMDRAM achieves system throughput on par with DRISA and Fulcrum
- **MIMDRAM's SIMD utilization versus SIMD RAM**
 - MIMDRAM provides **15.6x** the utilization of SIMD RAM
- **Area analysis**
 - MIMDRAM adds small area cost to a DRAM chip (**1.11%**) and CPU die (**0.6%**)

MIMDRAM: Summary

**We introduced MIMDRAM,
a hardware/software co-designed processing-using-DRAM system**

- **Key idea:** leverage fine-grained DRAM for processing-using-DRAM operation
- **HW:** - simple changes to DRAM, enabling concurrent instruction execution
- low-cost interconnects at the DRAM peripherals for data reduction
- **SW:** - compiler and OS support to generate and map instructions

Our evaluation demonstrates that MIMDRAM

- **significantly** improves **performance, energy efficiency,** and **throughput** compared to processor-centric (CPU and GPU) and memory-centric (SIMDRAM, DRISA, and Fulcrum) architectures
- incurs **small area cost** to a DRAM chip and CPU die

<https://github.com/CMU-SAFARI/MIMDRAM>

Several Other Works on PIM Programmability

Adoption: How to Ease Programmability? (I)

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, ["Transparent Offloading and Mapping \(TOM\): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"](#)

Proceedings of the [43rd International Symposium on Computer Architecture \(ISCA\)](#), Seoul, South Korea, June 2016.

[[Slides \(pptx\) \(pdf\)](#)]

[[Lightning Session Slides \(pptx\) \(pdf\)](#)]

Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim^{*} Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA ^{*}KAIST [§]ETH Zürich

Adoption: How to Ease Programmability? (II)

- Geraldo F. Oliveira, Alain Kohli, David Novo, Juan Gómez-Luna, Onur Mutlu,
“DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures,”
in *PACT SRC Student Competition*, Vienna, Austria, October 2023.

DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures

Geraldo F. Oliveira*

Alain Kohli*

David Novo‡

Juan Gómez-Luna*

Onur Mutlu*

**ETH Zürich*

‡*LIRMM, Univ. Montpellier, CNRS*

Adoption: How to Ease Programmability? (III)

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
"SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT), Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen¹ Juan Gómez-Luna¹ Izzat El Hajj² Yuxin Guo¹ Onur Mutlu¹
¹ETH Zürich ²American University of Beirut

Adoption: How to Keep It Simple?

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoun Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015. [[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoun Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[†]Carnegie Mellon University

Real DRAM Chips
Are Already Quite Capable:
FC-DRAM & SiMRA

Recall: DRAM Testing Infrastructure



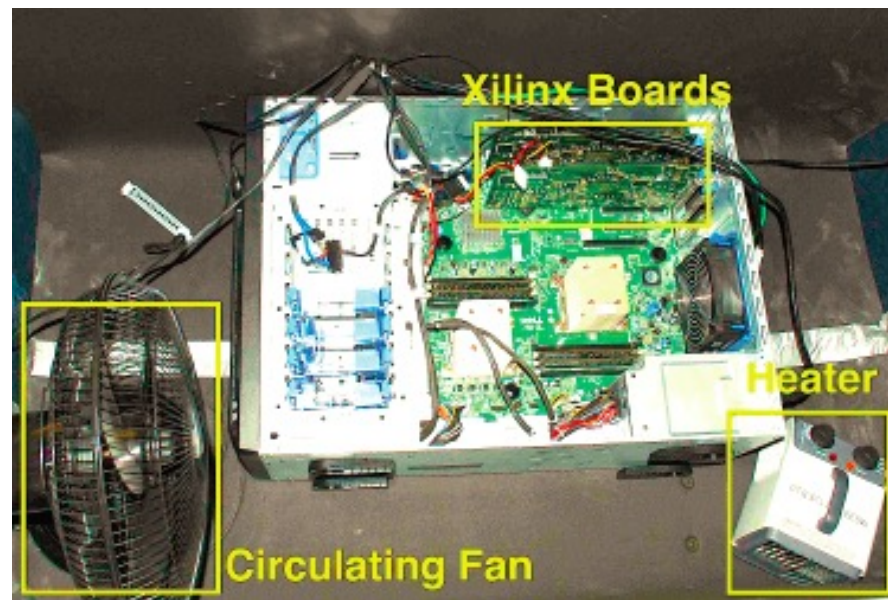
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

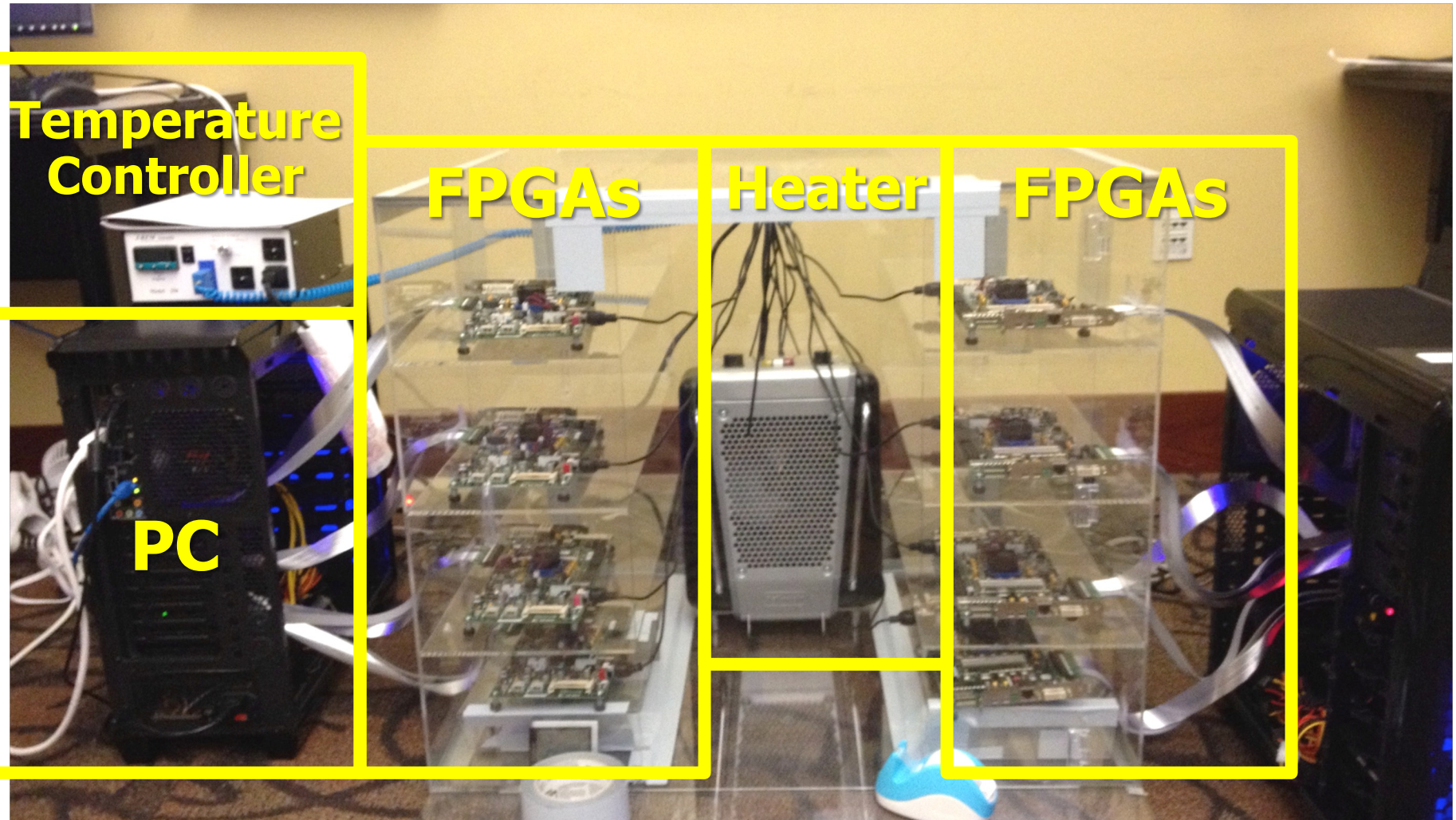
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)

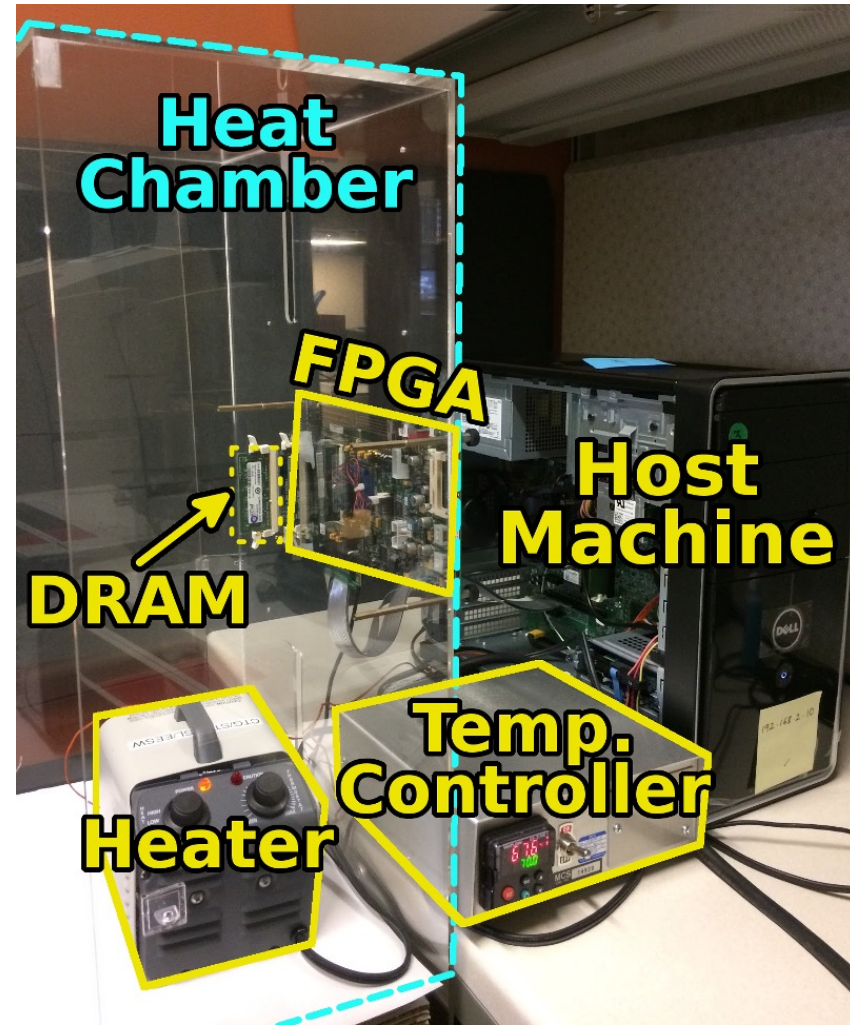


Recall: DRAM Testing Infrastructure



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



SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan, Nandita Vijaykumar, Samira Khan, Saugata Ghose, Kevin Chang, Gennady Pekhimenko, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,

"SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies"

Proceedings of the 23rd International Symposium on High-Performance Computer Architecture (HPCA), Austin, TX, USA, February 2017.

[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]

[Full Talk Lecture (39 minutes)]

[Source Code]

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan^{1,2,3} Nandita Vijaykumar³ Samira Khan^{4,3} Saugata Ghose³ Kevin Chang³
Gennady Pekhimenko^{5,3} Donghyuk Lee^{6,3} Oguz Ergin² Onur Mutlu^{1,3}

¹ETH Zürich ²TOBB University of Economics & Technology ³Carnegie Mellon University
⁴University of Virginia ⁵Microsoft Research ⁶NVIDIA Research

DRAM Bender

- Ataberk Olgun, Hasan Hassan, A Giray Yağlıkçı, Yahya Can Tuğrul, Lois Orosa, Haocong Luo, Minesh Patel, Oğuz Ergin, and Onur Mutlu,
"DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips"
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2023.
[[Extended arXiv version](#)]
[[DRAM Bender Source Code](#)]
[[DRAM Bender Tutorial Video](#) (43 minutes)]

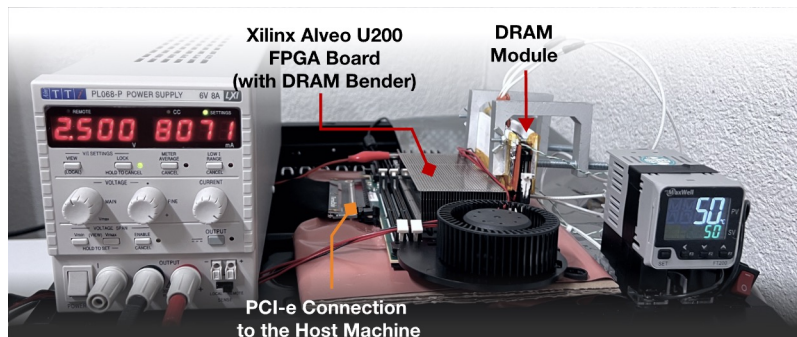
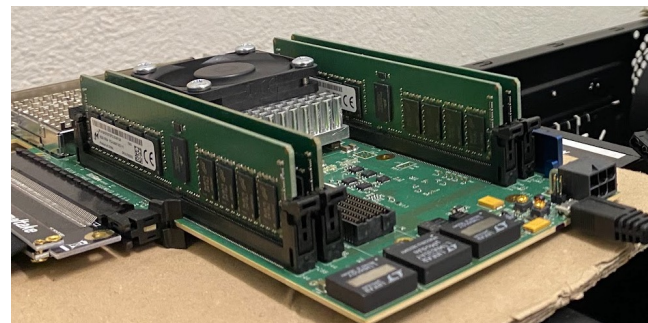
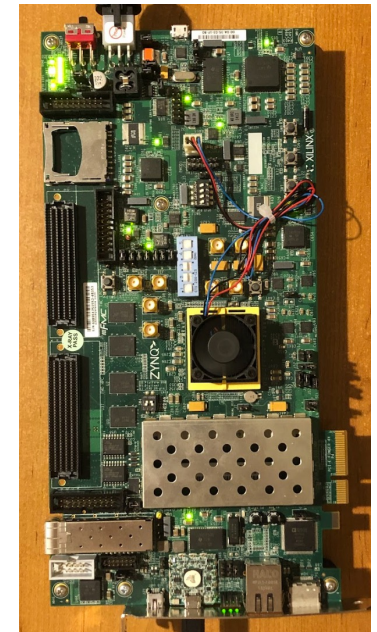
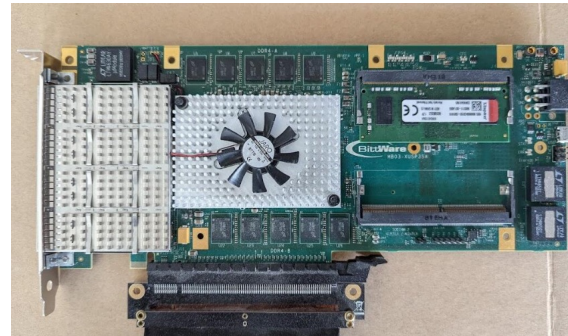
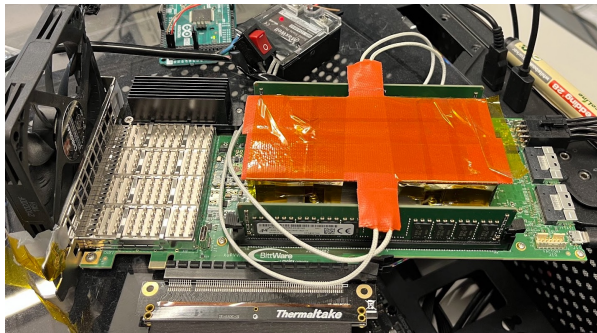
DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun[§] Hasan Hassan[§] A. Giray Yağlıkçı[§] Yahya Can Tuğrul^{§†}
Lois Orosa^{§⊙} Haocong Luo[§] Minesh Patel[§] Oğuz Ergin[†] Onur Mutlu[§]
 [§]*ETH Zürich* [†]*TOBB ETÜ* [⊙]*Galician Supercomputing Center*

DRAM Bender: Prototypes

Testing Infrastructure	Protocol Support	FPGA Support
SoftMC [134]	DDR3	One Prototype
LiteX RowHammer Tester (LRT) [17]	DDR3/4, LPDDR4	Two Prototypes
DRAM Bender (this work)	DDR3/DDR4	Five Prototypes

Five out of the box FPGA-based prototypes



DRAM Chips Are Already (Quite) Capable!

- **Appears at HPCA 2024** <https://arxiv.org/pdf/2402.18736.pdf>

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

We experimentally demonstrate that COTS DRAM chips are capable of performing 1) functionally-complete Boolean operations: NOT, NAND, and NOR and 2) many-input (i.e., more than two-input) AND and OR operations. We present an extensive characterization of new bulk bitwise operations in 256 off-the-shelf modern DDR4 DRAM chips. We evaluate the reliability of these operations using a metric called success rate: the fraction of correctly performed bitwise operations. Among our 19 new observations, we highlight four major results. First, we can perform the NOT operation on COTS DRAM chips with 98.37% success rate on average. Second, we can perform up to 16-input NAND, NOR, AND, and OR operations on COTS DRAM chips with high reliability (e.g., 16-input NAND, NOR, AND, and OR with average success rate of 94.94%, 95.87%, 94.94%, and 95.85%, respectively). Third, data pattern only slightly

DRAM Chips Are Already (Quite) Capable!

- <https://arxiv.org/pdf/2312.02880.pdf>

PULSAR: Simultaneous Many-Row Activation for Reliable and High-Performance Computing in Off-the-Shelf DRAM Chips

Ismail Emir Yuksel Yahya Can Tugrul F. Nisa Bostanci Abdullah Giray Yaglikci Ataberk Olgun
Geraldo F. Oliveira Melina Soysal Haocong Luo Juan Gomez Luna Mohammad Sadrosadati
Onur Mutlu

ETH Zurich

We propose PULSAR, a new technique to enable high-success-rate and high-performance PuM operations in off-the-shelf DRAM chips. PULSAR leverages our new observation that a carefully-crafted sequence of DRAM commands simultaneously activates up to 32 DRAM rows. PULSAR overcomes the limitations of existing techniques by 1) replicating the input data to improve the success rate and 2) enabling new bulk bitwise operations (e.g., many-input majority, *Multi-RowInit*, and *Bulk-Write*) to improve the performance.

DRAM Chips Are Already (Quite) Capable!

- **Appears at DSN 2024**



Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel¹ Yahya Can Tuğrul^{1,2} F. Nisa Bostancı¹ Geraldo F. Oliveira¹
A. Giray Yağlıkçı¹ Ataberk Olgun¹ Melina Soysal¹ Haocong Luo¹
Juan Gómez-Luna¹ Mohammad Sadrosadati¹ Onur Mutlu¹

¹*ETH Zürich* ²*TOBB University of Economics and Technology*

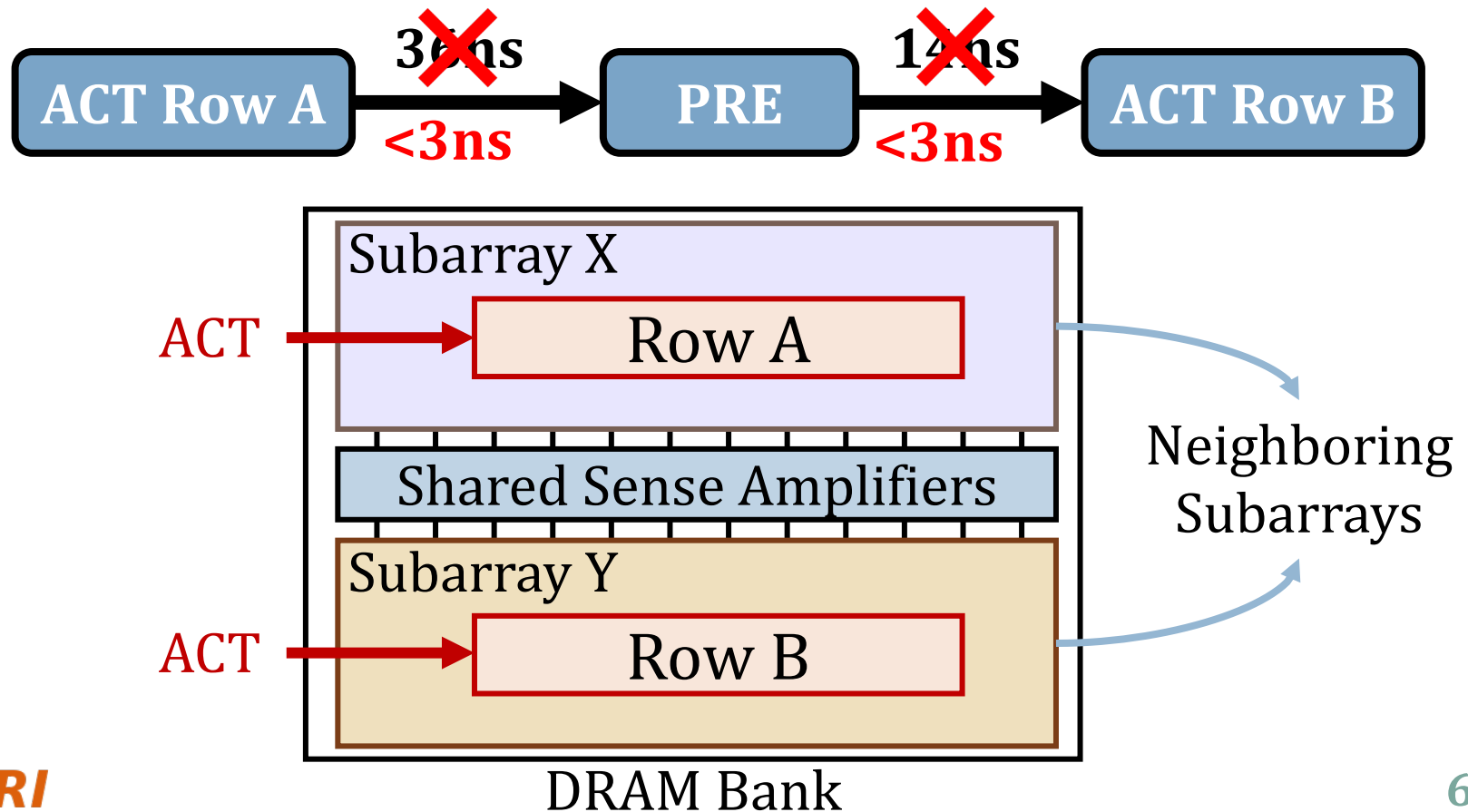
The Capability of COTS DRAM Chips

We demonstrate that COTS DRAM chips:

- 1 Can simultaneously activate up to 48 rows in two neighboring subarrays
- 2 Can perform **NOT operation** with up to **32 output operands**
- 3 Can perform up to **16-input AND, NAND, OR, and NOR** operations

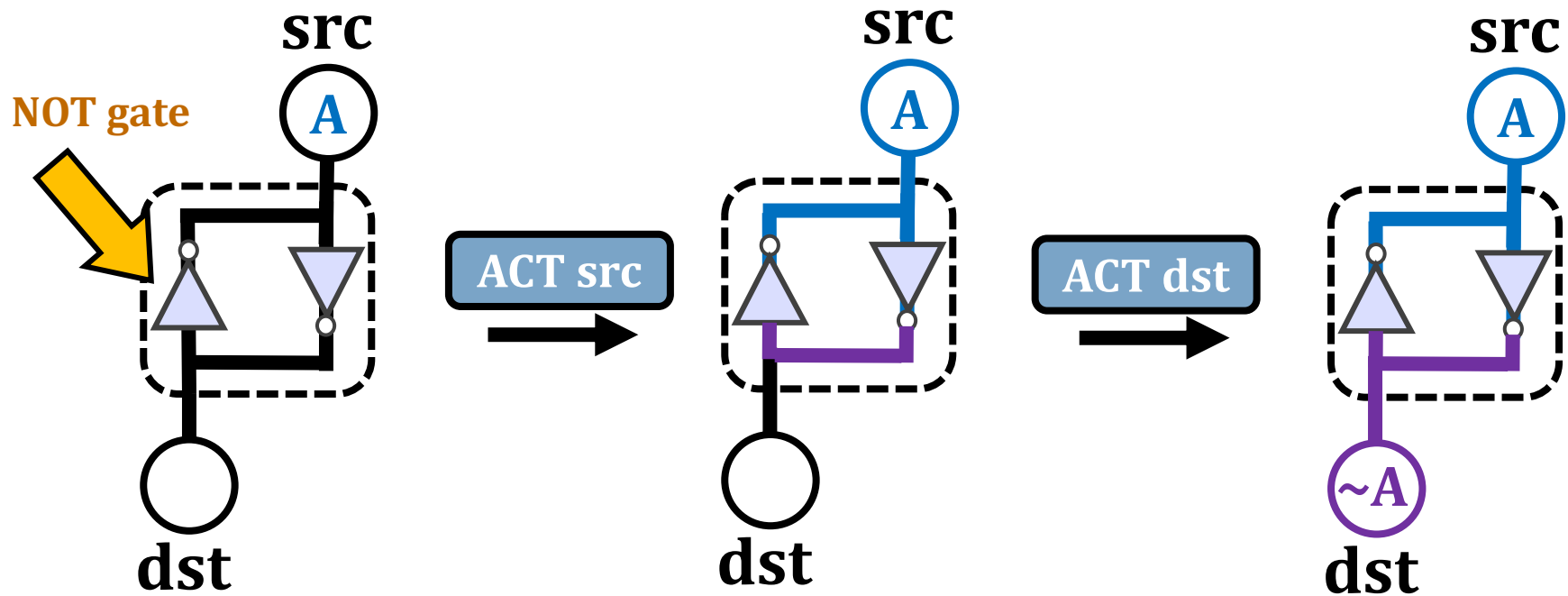
Finding: SiMRA Across Subarrays

Activating two rows in **quick succession**
can **simultaneously** activate
multiple rows in neighboring subarrays



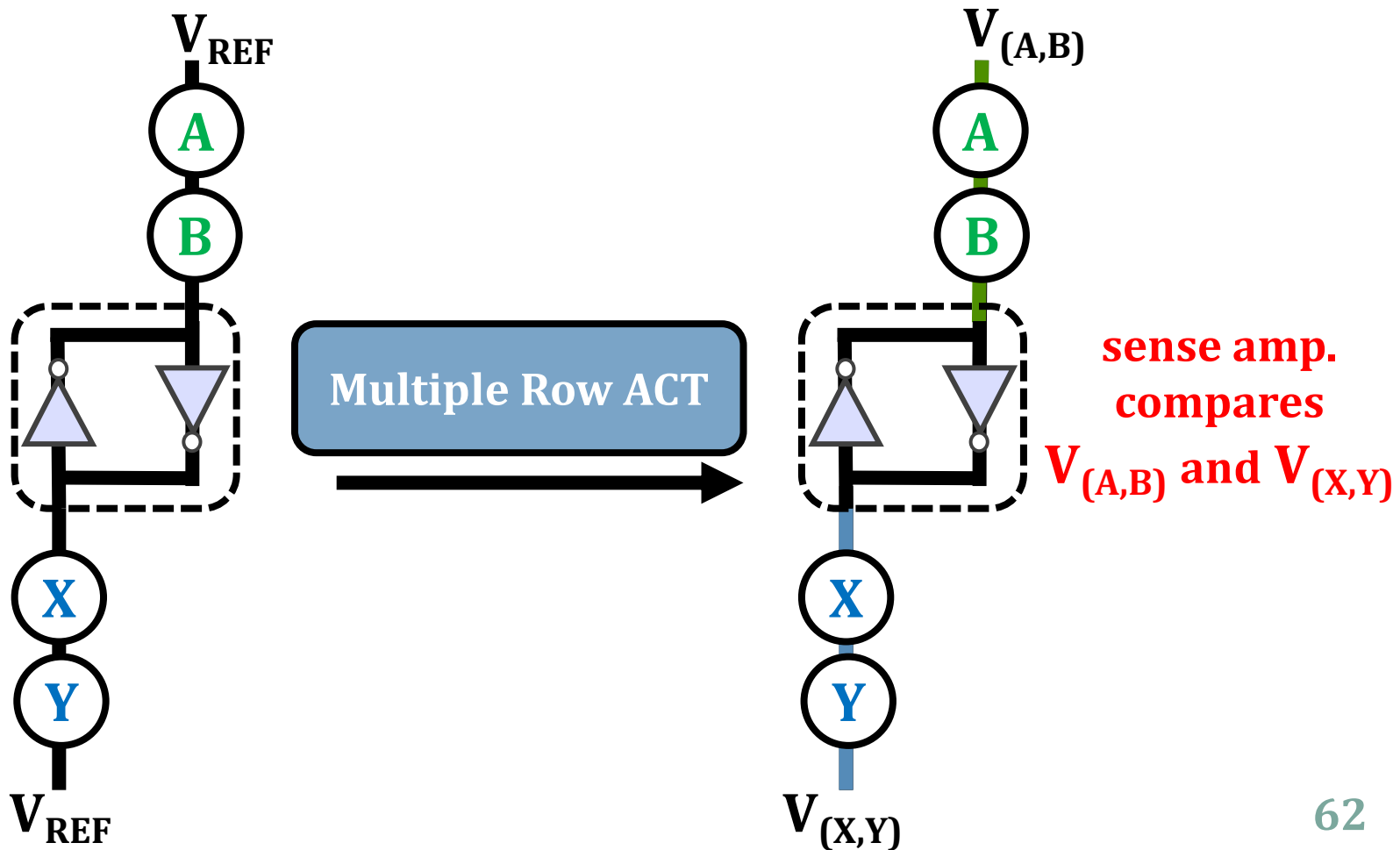
Key Idea: NOT Operation

Connect rows in neighboring subarrays through a **NOT gate** by simultaneously activating rows

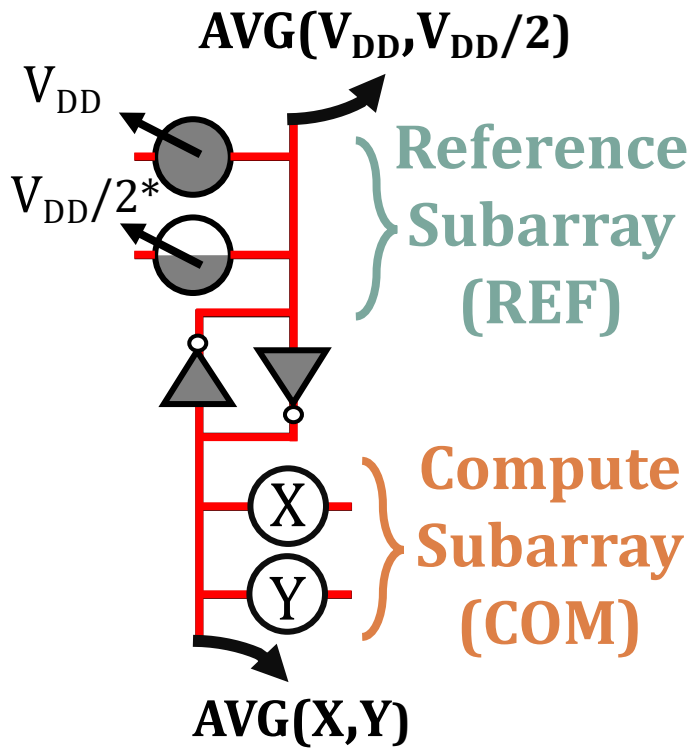


Key Idea: NAND, NOR, AND, OR

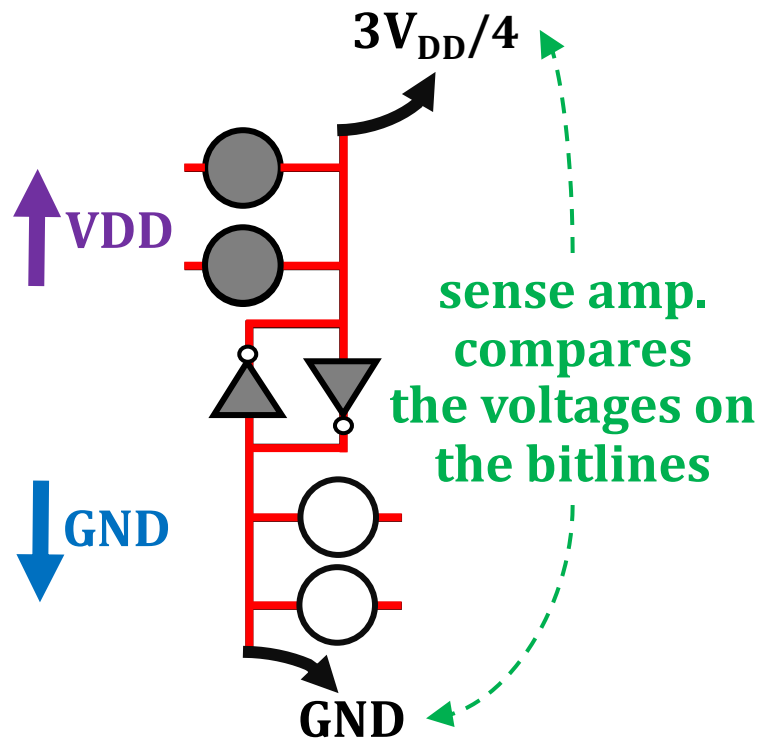
Manipulate the bitline voltage to express
a wide variety of functions using
multiple-row activation in neighboring subarrays



Two-Input AND and NAND Operations



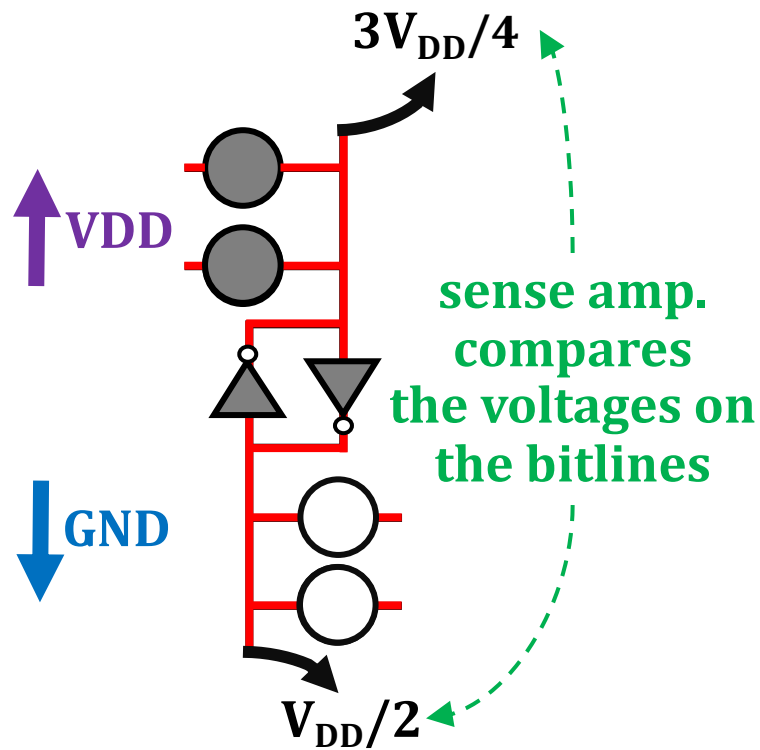
Two-Input AND and NAND Operations



$V_{DD}=1$ & $GND=0$

X	Y	COM	REF
0	0	0	1

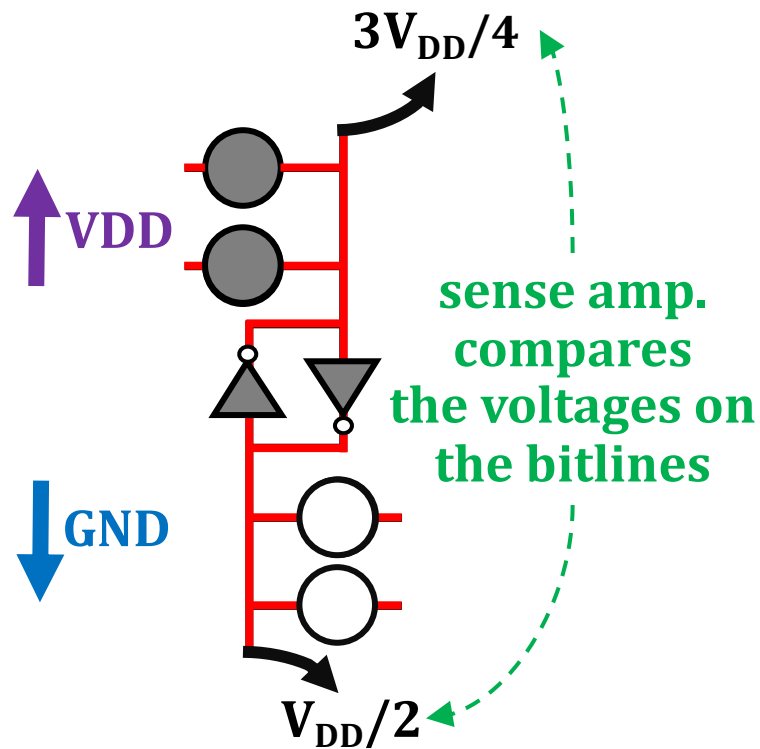
Two-Input AND and NAND Operations



$V_{DD}=1$ & $GND=0$

X	Y	COM	REF
0	0	0	1
0	1	0	1

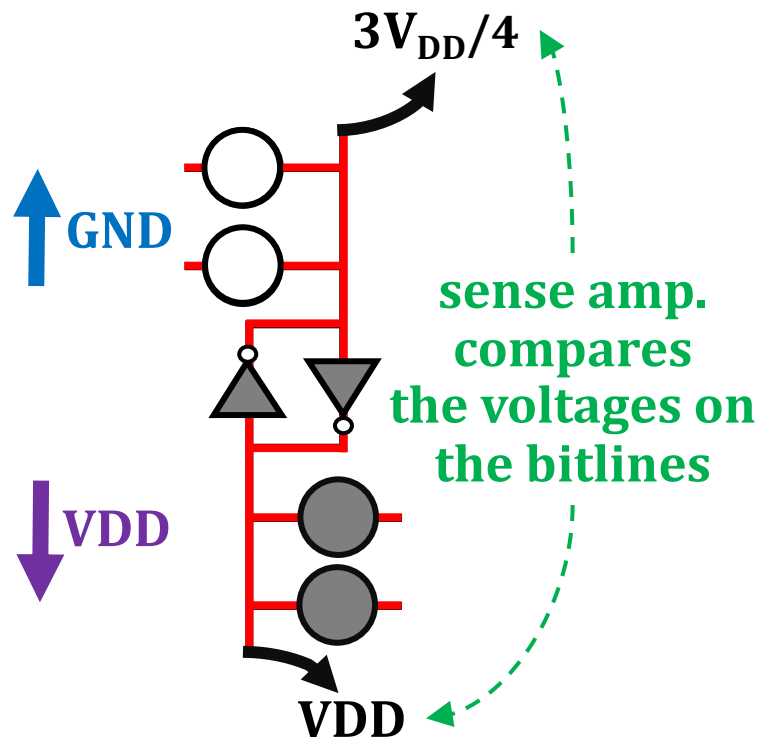
Two-Input AND and NAND Operations



$V_{DD}=1$ & $GND=0$

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1

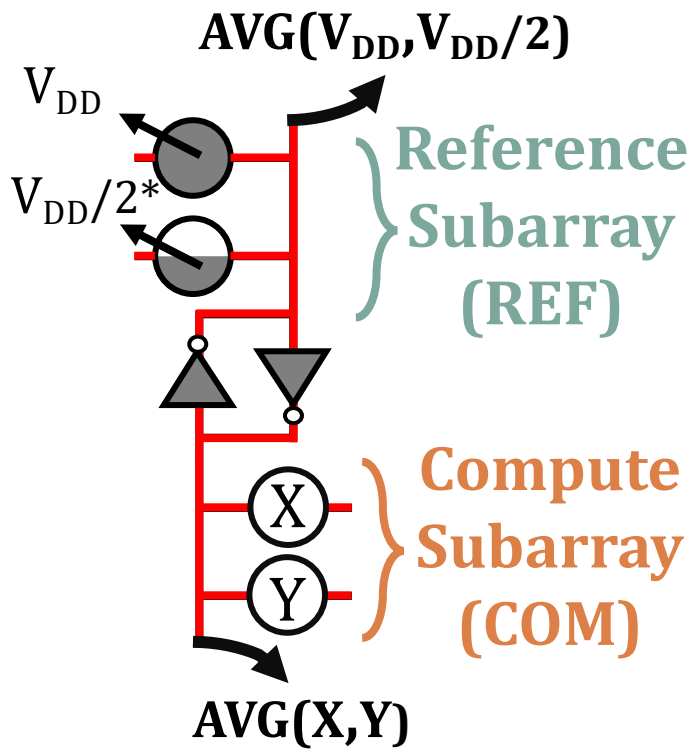
Two-Input AND and NAND Operations



$V_{DD}=1$ & $GND=0$

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Two-Input AND and NAND Operations



$V_{DD}=1$ & $\text{GND} = 0$

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
		AND	NAND

Many-Input AND, NAND, OR, and NOR Operations

We can express **AND**, **NAND**, **OR**, and **NOR** operations by carefully manipulating the **reference voltage**

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

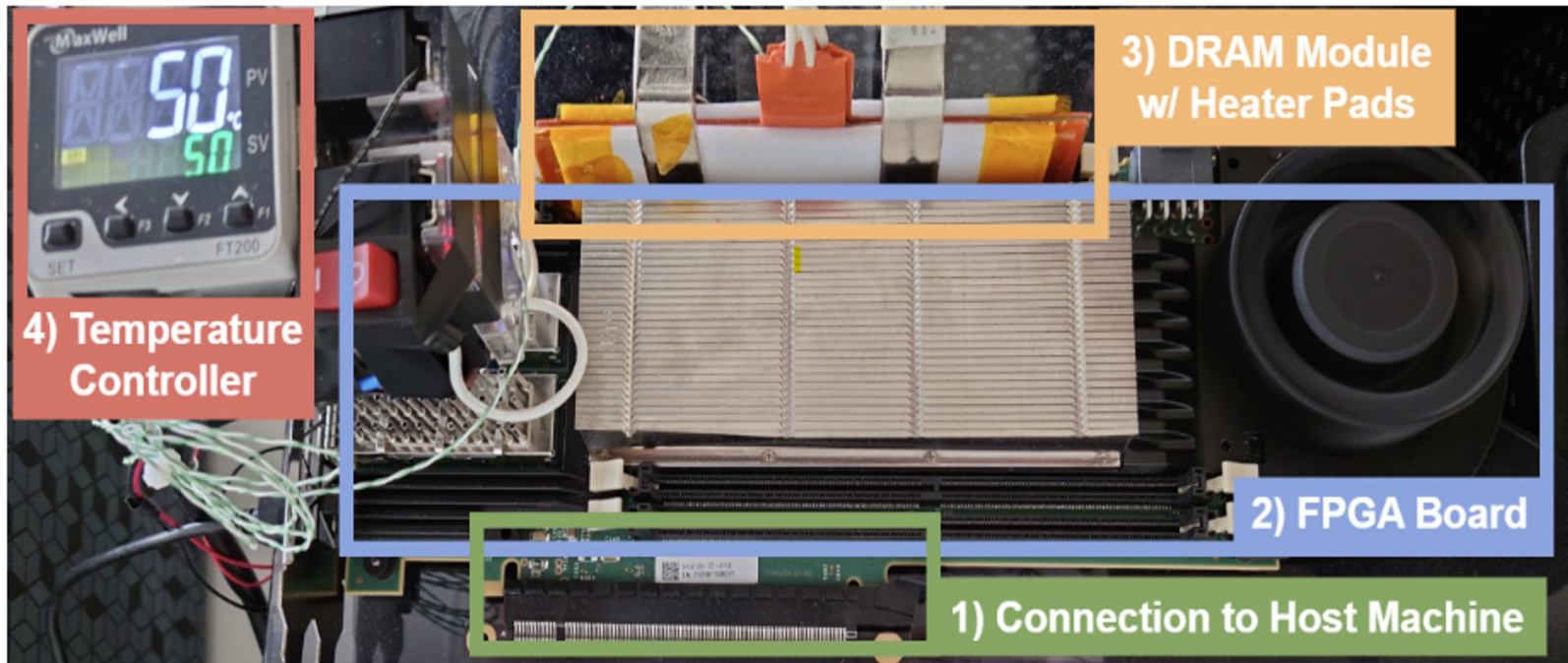
ETH Zürich

(More details in the paper)

<https://arxiv.org/pdf/2402.18736.pdf>

DRAM Testing Infrastructure

- Developed from [DRAM Bender \[Olgun+, TCAD'23\]*](#)
- **Fine-grained control** over DRAM commands, timings, and temperature

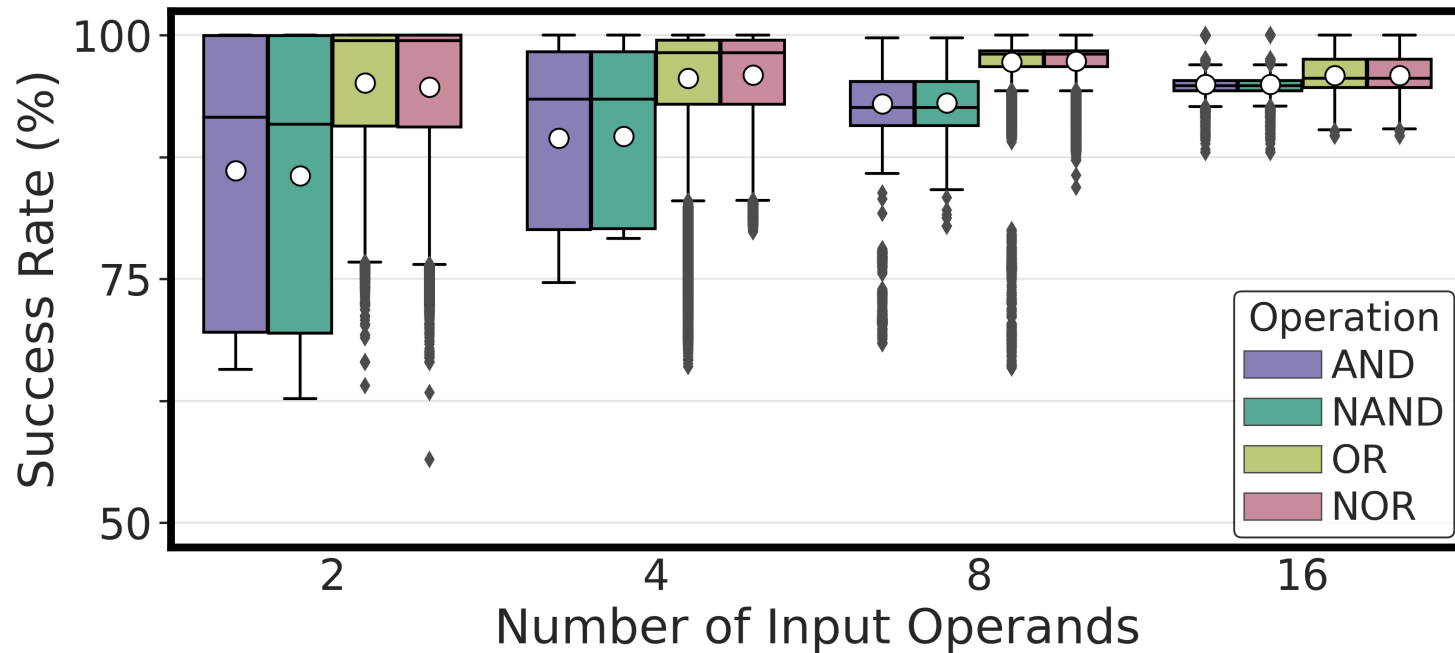


DRAM Chips Tested

- 256 DDR4 chips from two major DRAM manufacturers
- Covers different die revisions and chip densities

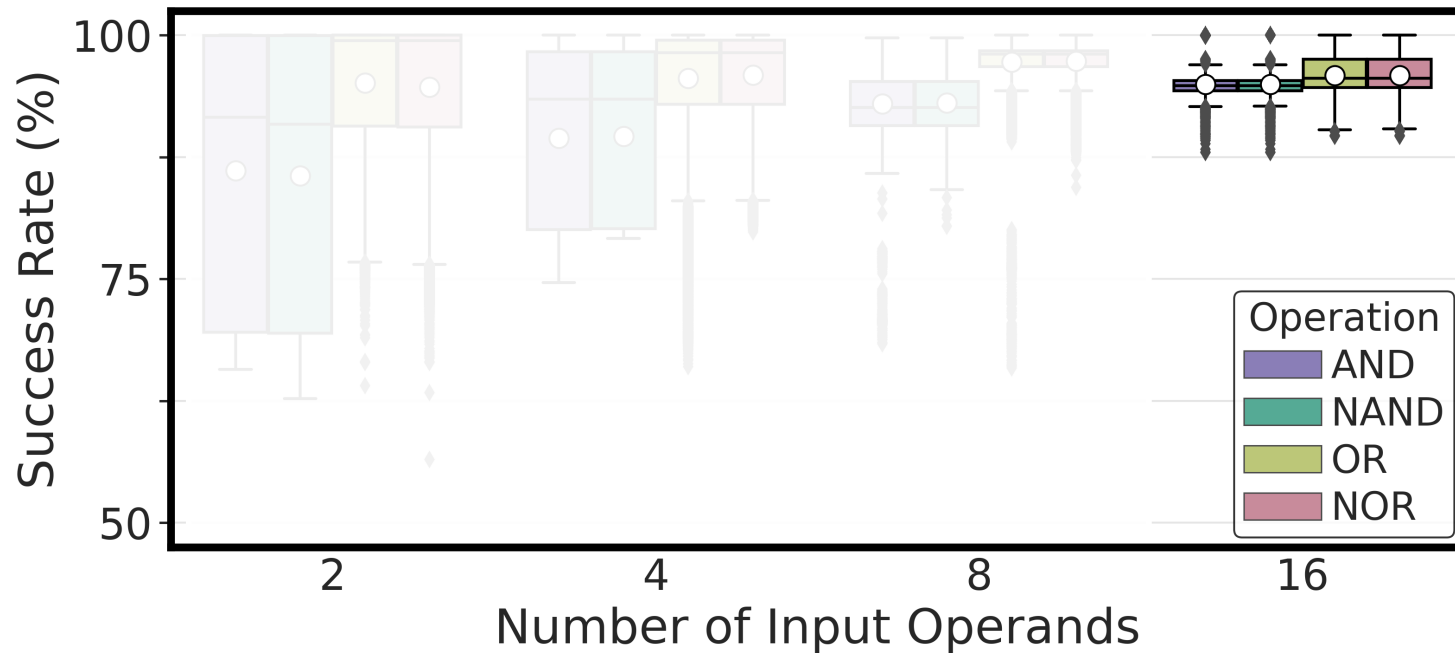
Chip Mfr.	#Modules (#Chips)	Die Rev.	Mfr. Date ^a	Chip Density	Chip Org.	Speed Rate
SK Hynix	9 (72)	M	N/A	4Gb	x8	2666MT/s
	5 (40)	A	N/A	4Gb	x8	2133MT/s
	1 (16)	A	N/A	8Gb	x8	2666MT/s
	1 (32)	A	18-14	4Gb	x4	2400MT/s
	1 (32)	A	16-49	8Gb	x4	2400MT/s
	1 (32)	M	16-22	8Gb	x4	2666MT/s
Samsung	1 (8)	F	21-02	4Gb	x8	2666MT/s
	2 (16)	D	21-10	8Gb	x8	2133MT/s
	1 (8)	A	22-12	8Gb	x8	3200MT/s

Performing AND, NAND, OR, and NOR



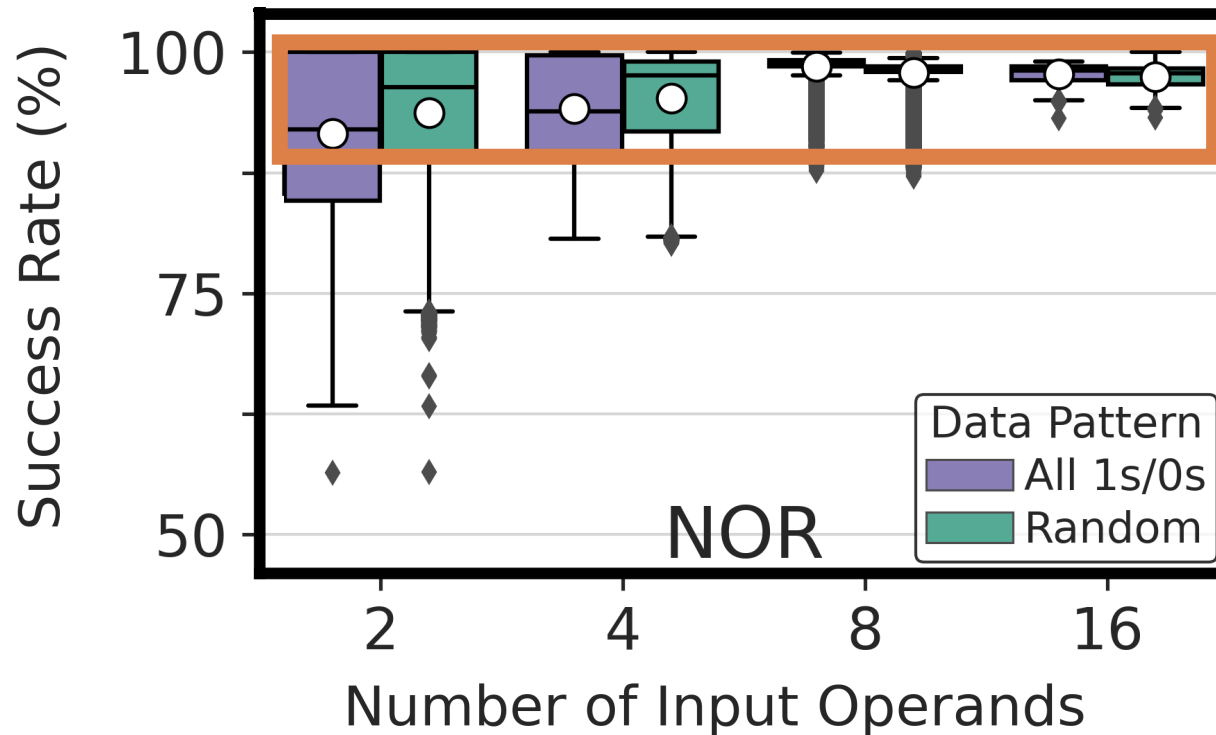
COTS DRAM chips can perform {2, 4, 8, 16}-input AND, NAND, OR, and NOR operations

Performing AND, NAND, OR, and NOR



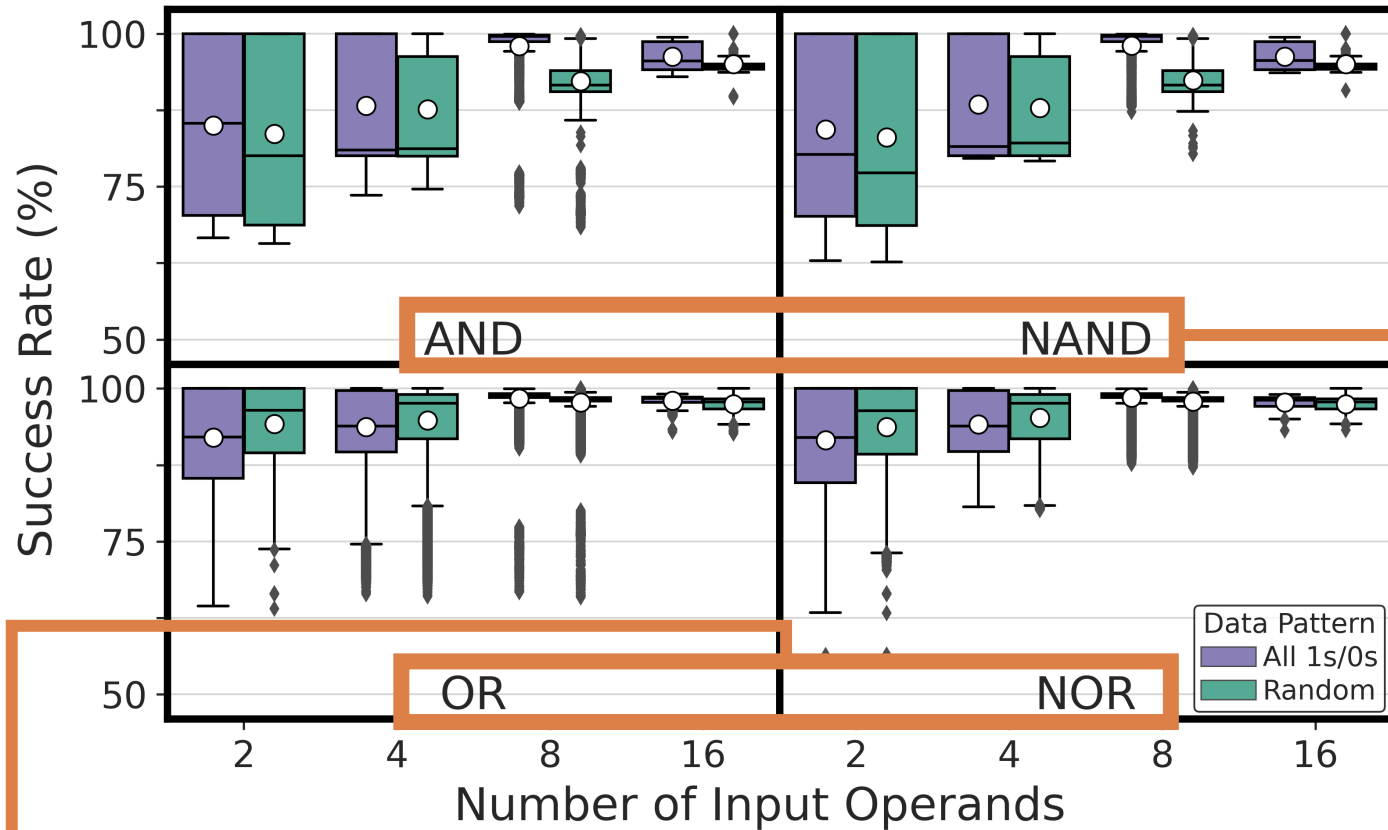
**COTS DRAM chips can perform
16-input AND, NAND, OR, and NOR operations
with very high success rate (>94%)**

Impact of Data Pattern



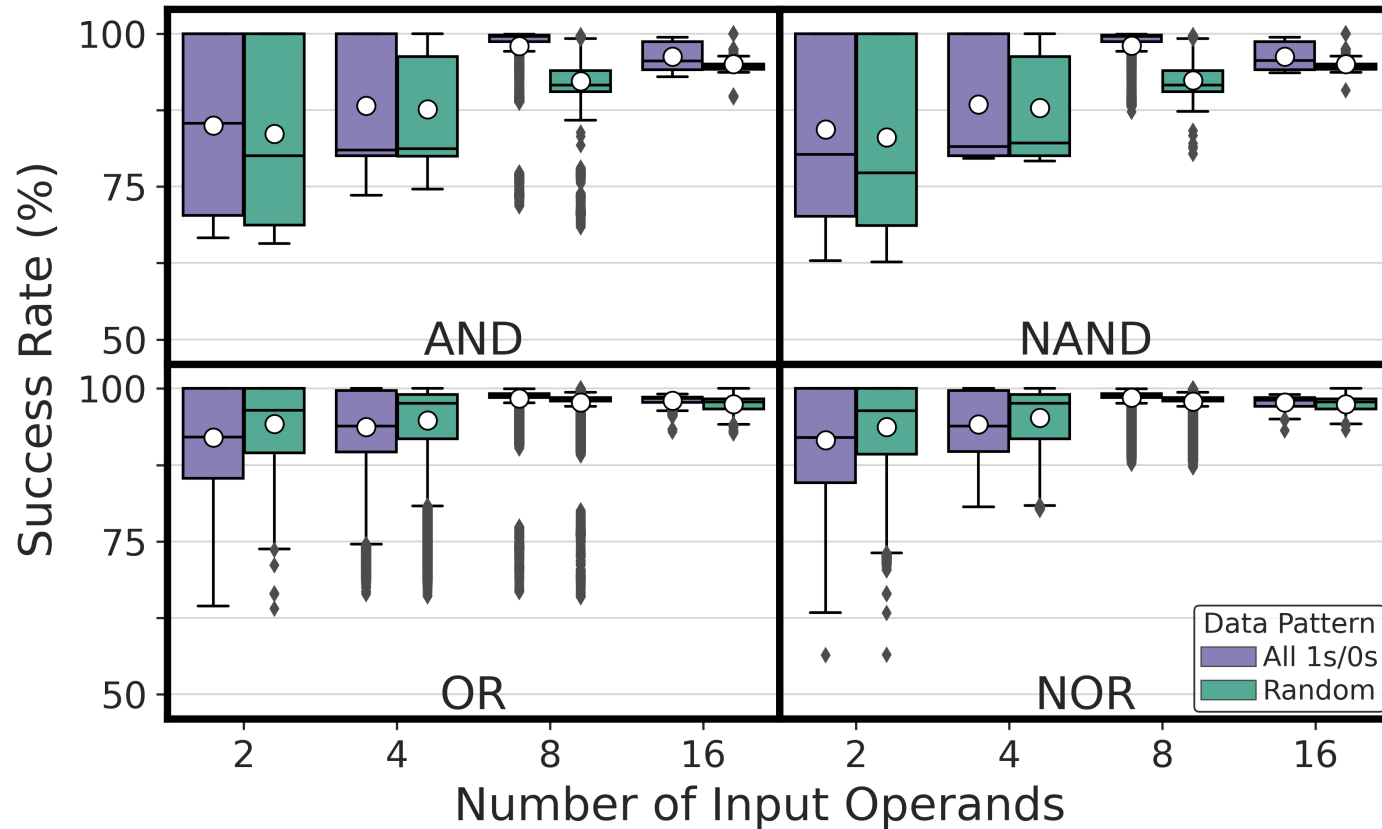
**1.98% variation in average success rate
across all number of input operands**

Impact of Data Pattern



Impact of data pattern is **consistent** across all tested operations

Impact of Data Pattern



Data pattern slightly affects the reliability of AND, NAND, OR, and NOR operations

More in the Paper

- Detailed hypotheses & key ideas to perform
 - NOT operation
 - Many-input AND, NAND, OR, and NOR operations
- How the reliability of bitwise operations are affected by
 - The location of activated rows
 - Temperature (for AND, NAND, OR, and NOR)
 - DRAM speed rate
 - Chip density and die revision
- Discussion on the limitations of COTS DRAM chips

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

Processing-using-DRAM (PuD) is an emerging paradigm that leverages the analog operational properties of DRAM circuitry to enable massively parallel in-DRAM computation. PuD has the potential to significantly reduce or eliminate costly data movement between processing elements and main memory. A common approach for PuD architectures is to make use of bulk bitwise computation (e.g., AND, OR, NOT). Prior works experimentally demonstrate three-input MAJ (i.e., MAJ3) and two-input AND and OR operations in commercial off-the-shelf (COTS) DRAM chips. Yet, demonstrations on COTS DRAM chips do not provide a functionally complete set of operations (e.g., NAND or AND and NOT).

We experimentally demonstrate that COTS DRAM chips are capable of performing 1) functionally-complete Boolean operations: NOT, NAND, and NOR and 2) many-input (i.e., more than two-input) AND and OR operations. We present an extensive

systems and applications [12, 13]. Processing-using-DRAM (PuD) [29–32] is a promising paradigm that can alleviate the data movement bottleneck. PuD uses the analog operational properties of the DRAM circuitry to enable massively parallel in-DRAM computation. Many prior works [29–53] demonstrate that PuD can greatly reduce or eliminate data movement.

A widely used approach for PuD is to perform bulk bitwise operations, i.e., bitwise operations on large bit vectors. To perform bulk bitwise operations using DRAM, prior works propose modifications to the DRAM circuitry [29–31, 33, 35, 36, 43, 44, 46, 48–58]. Recent works [38, 41, 42, 45] experimentally demonstrate the feasibility of executing data copy & initialization [42, 45], i.e., the RowClone operation [49], and a subset of bitwise operations, i.e., three-input bitwise majority (MAJ3) and two-input AND and OR operations in unmodified commercial off-the-shelf (COTS) DRAM chips by operating beyond

<https://arxiv.org/pdf/2402.18736.pdf>

Summary

- We experimentally demonstrate that **commercial off-the-shelf (COTS)** DRAM chips can perform:
 - **Functionally-complete** Boolean operations: NOT, NAND, and NOR
 - **Up to 16-input** AND, NAND, OR, and NOR operations
- We characterize **the success rate** of these operations on **256 COTS DDR4 chips** from **two major manufacturers**
- We highlight **two key results**:
 - We can perform **NOT** and **{2, 4, 8, 16}-input AND, NAND, OR, and NOR** operations on COTS DRAM chips with **very high success rates (>94%)**
 - **Data pattern** and **temperature** only slightly affect the reliability of these operations

We believe these empirical results demonstrate the promising potential of using DRAM as a computation substrate

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips

Experimental Characterization and Analysis



İsmail Emir Yüksel

Yahya C. Tuğrul F. Nisa Bostancı Geraldo F. Oliveira

A. Giray Yağlıkçı Ataberk Olgun Melina Soysal Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI

ETH zürich

Executive Summary

Motivation:

- **Processing-Using-DRAM (PUD)** alleviates **data movement bottlenecks**
- Commercial off-the-shelf (COTS) DRAM chips can perform **three-input majority (MAJ3)** and **in-DRAM copy** operations

Goal: To experimentally analyze and understand

- The **computational capability** of COTS DRAM chips beyond that of prior works
- The **robustness** of such capability under various **operating conditions**

Experimental Study: 120 DDR4 chips from two major manufacturers

- COTS DRAM chips can perform **MAJ5, MAJ7, and MAJ9** operations and **copy** one DRAM row to **up to 31 different rows** at once
- Storing **multiple redundant copies** of MAJ's input operands (i.e., input replication) drastically increases **robustness** (>30% higher success rate)
- **Operating conditions** (temperature, voltage, and data pattern) **affect** the robustness of in-DRAM operations (by up to 11.52% success rate)

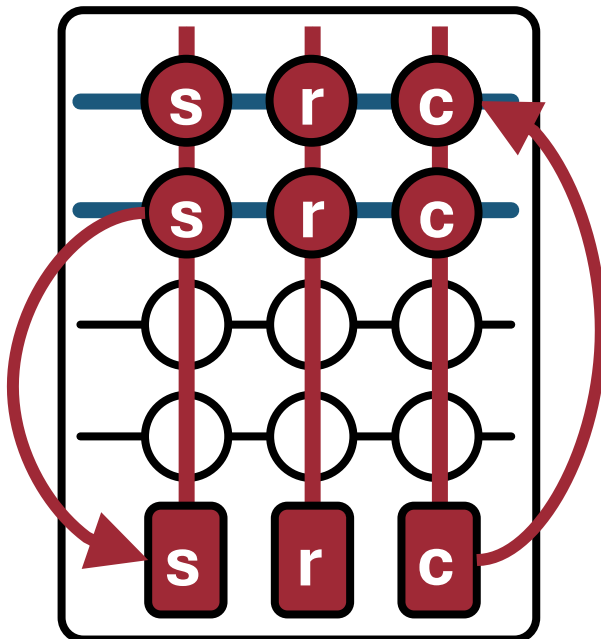
Leveraging Simultaneous Many-Row Activation

- 1** Perform **MAJX** (where $X > 3$) operations
- 2** Increase the **robustness** of MAJX operations
- 3** Copy **one row's content** to **multiple rows**

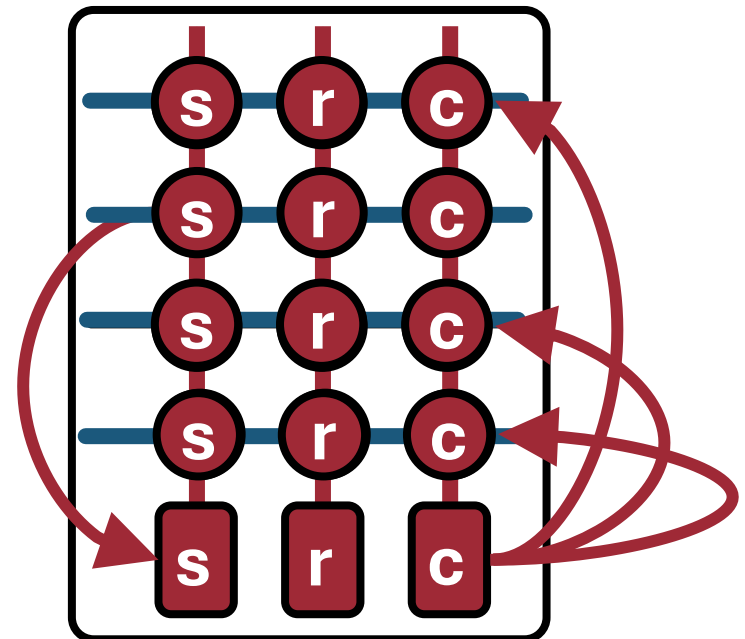
In-DRAM Multiple Row Copy (Multi-RowCopy)

Simultaneously activate many rows to copy **one row's content** to **multiple destination rows**

RowClone



Multi-RowCopy



Key Takeaways from Multi-RowCopy

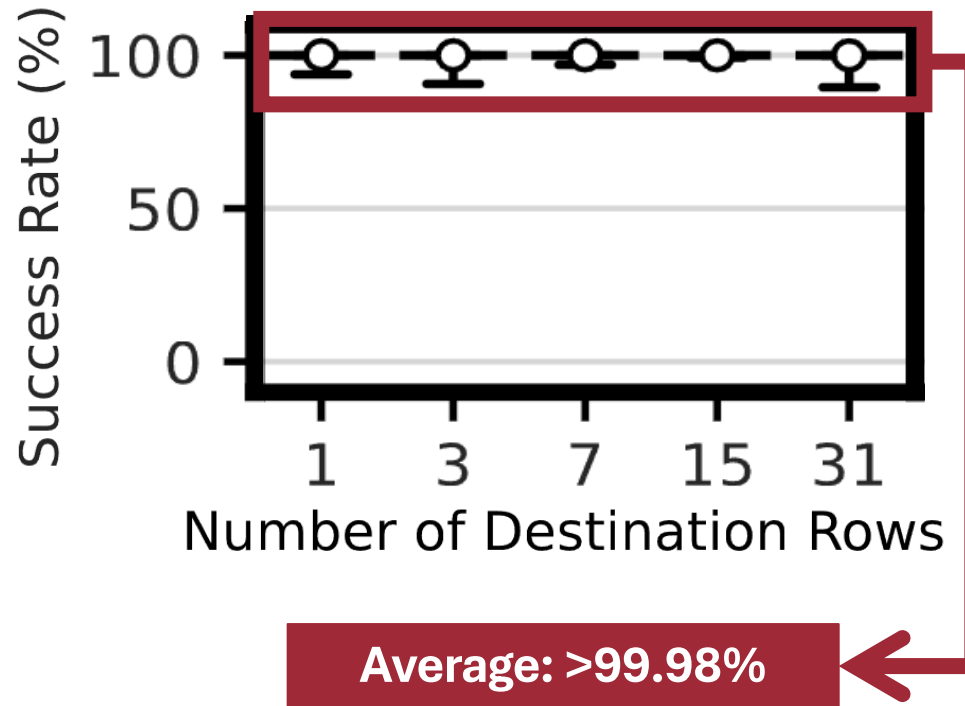
Key Takeaway 1

COTS DRAM chips are capable of copying one row's data to 1, 3, 7, 15, and 31 other rows at very high success rates

Key Takeaway 2

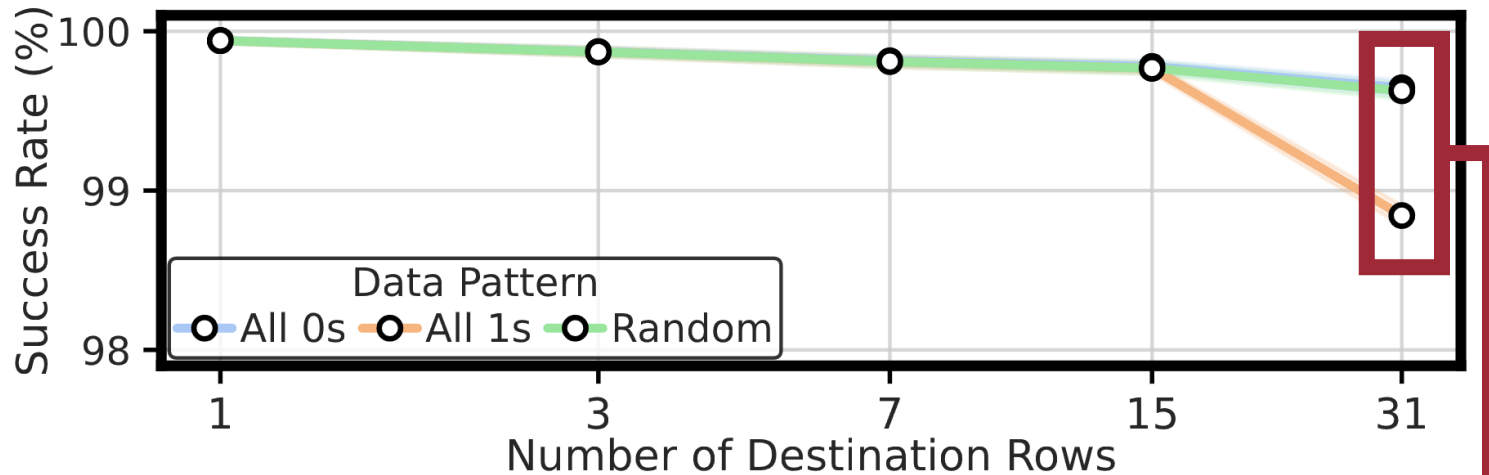
Multi-RowCopy in COTS DRAM chips is highly resilient to changes in data pattern, temperature, and wordline voltage

Robustness of Multi-RowCopy



COTS DRAM chips can copy one row's content to up to 31 rows with a very high success rate

Impact of Data Pattern

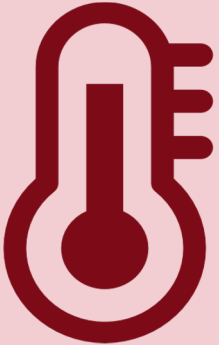


**At most 0.79% decrease in
average success rate**

**Data pattern has a small effect
on the success rate of the Multi-RowCopy operation**

Also in the Paper: Impact of Temperature & Voltage

Temperature



50 °C → 90 °C

Increasing temperature up to 90°C
has a very small effect on
the success rate of the Multi-RowCopy operation

Wordline Voltage



2.5V → 2.1V

Reducing the wordline voltage
only slightly affects
the success rate of the Multi-RowCopy operation

More in the Paper

- Detailed hypotheses and key ideas on
 - Hypothetical row decoder circuitry
 - Input Replication
- More characterization results
 - Power consumption of simultaneous many-row activation
 - Effect of timing delays between ACT-PRE and PRE-ACT commands
 - Effect of temperature and wordline voltage
- Circuit-level (SPICE) experiments for input replication
- Potential performance benefits of enabling new in-DRAM operations
 - Majority-based computation
 - Content destruction-based cold-boot attack prevention
- Discussions on the limitations of tested COTS DRAM chips



Code
Reproducible



Dataset
Reproducible

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel¹ Yahya Can Tuğrul^{1,2} F. Nisa Bostancı¹ Geraldo F. Oliveira¹
A. Giray Yağlıkçı¹ Ataberk Olgun¹ Melina Soysal¹ Haocong Luo¹
Juan Gómez-Luna¹ Mohammad Sadrosadati¹ Onur Mutlu¹
¹ETH Zürich ²TOBB University of Economics and Technology

*We experimentally analyze the computational capability of commercial off-the-shelf (COTS) DRAM chips and the robustness of these capabilities under various timing delays between DRAM commands, data patterns, temperature, and voltage levels. We extensively characterize 120 COTS DDR4 chips from two major manufacturers. We highlight four key results of our study. First, COTS DRAM chips are capable of 1) simultaneously activating up to 32 rows (i.e., simultaneous many-row activation), 2) executing a majority of X (MAJX) operation where $X > 3$ (i.e., MAJ5, MAJ7, and MAJ9 operations), and 3) copying a DRAM row (concurrently) to up to 31 other DRAM rows, which we call **Multi-RowCopy**. Second, storing multiple copies of MAJX's input operands on all simultaneously activated rows drastically increases the success rate (i.e., the percentage of DRAM cells that correctly perform the computation) of the MAJX operation. For example, MAJ3 with 32-row activation (i.e.,*

A subset of PIM proposals devise mechanisms that enable PUM using DRAM cells for computation, including data copy and initialization [67, 72, 77, 78, 89, 104, 127], Boolean logic [56, 64–66, 68, 70, 72, 76, 79, 122, 127–129], majority-based arithmetic [64, 66, 69, 72, 91, 127, 130, 131], and lookup table based operations [82, 106, 107, 132]. We refer to DRAM-based PUM as *Processing-Using-DRAM (PUD)* and the computation performed using DRAM cells as PUD operations.

PUD benefits from the bulk data parallelism in DRAM devices to perform bulk bitwise PUD operations. Prior works show that bulk bitwise operations are used in a wide variety of important applications, including databases and web search [64, 67, 79, 130, 133–140], data analytics [64, 141–144], graph processing [56, 80, 94, 130, 145], genome analysis [60, 99, 146–149], cryptography [150, 151], set operations [56, 64], and hyper-dimensional computing [152–154].

<https://arxiv.org/pdf/2405.06081>


Our Work is Open Source and Artifact Evaluated










Code
Reproducible





Dataset
Reproducible



 **SiMRA-DRAM** Public






 Edit Pins  Watch 4  Fork 0  Starred 6



 main  1 Branch  0 Tags

 Add file  Code

About





 **unrealismail** Update README.md a51abfa · last month  5 Commits




 DRAM-Bender	initial comit	last month
 analysis	initial comit	last month
 experimental_data	initial comit	last month
 LICENSE	initial comit	last month
 README.md	Update README.md	last month

 **README**  License

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

Source code & scripts for experimental characterization and demonstration of 1) simultaneous many-row activation, 2) up to nine-input majority operations and 3) copying one row's content to up 31 rows in real DDR4 DRAM chips. Described in our DSN'24 paper by Yuksel et al. at <https://arxiv.org/abs/2405.06081>

 Readme  View license  Activity  Custom properties

 6 stars  4 watching  0 forks

Report repository

<https://github.com/CMU-SAFARI/SiMRA-DRAM>

What Else Can We Do Using DRAM?

In-DRAM True Random Number Generation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu,
"D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"
Proceedings of the 25th International Symposium on High-Performance Computer Architecture (HPCA), Washington, DC, USA, February 2019.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Full Talk Video](#) (21 minutes)]
[[Full Talk Lecture Video](#) (27 minutes)]
Top Picks Honorable Mention by IEEE Micro.

D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim^{‡§} Minesh Patel[§] Hasan Hassan[§] Lois Orosa[§] Onur Mutlu^{§‡}
[‡]Carnegie Mellon University [§]ETH Zürich

In-DRAM True Random Number Generation

- Ataberk Olgun, Minesh Patel, A. Giray Yaglikci, Haocong Luo, Jeremie S. Kim, F. Nisa Bostanci, Nandita Vijaykumar, Oguz Ergin, and Onur Mutlu,
"QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"
Proceedings of the 48th International Symposium on Computer Architecture (ISCA), Virtual, June 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Short Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (25 minutes)]
[[SAFARI Live Seminar Video](#) (1 hr 26 mins)]

QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

Ataberk Olgun^{§†}

Minesh Patel[§]

A. Giray Yağlıkçı[§]

Haocong Luo[§]

Jeremie S. Kim[§]

F. Nisa Bostanci^{§†}

Nandita Vijaykumar^{§⊙}

Oğuz Ergin[†]

Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB University of Economics and Technology

[⊙]University of Toronto

In-DRAM True Random Number Generation

- F. Nisa Bostanci, Ataberk Olgun, Lois Orosa, A. Giray Yaglikci, Jeremie S. Kim, Hasan Hassan, Oguz Ergin, and Onur Mutlu,

"DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators"

Proceedings of the 28th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, April 2022.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Short Talk Slides \(pptx\)](#) ([pdf](#))]

DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators

F. Nisa Bostanci^{†§}

Ataberk Olgun^{†§}

Lois Orosa[§]

A. Giray Yağlıkçı[§]

Jeremie S. Kim[§]

Hasan Hassan[§]

Oğuz Ergin[†]

Onur Mutlu[§]

[†]*TOBB University of Economics and Technology*

[§]*ETH Zürich*

In-DRAM Physical Unclonable Functions

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,
"The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"
Proceedings of the 24th International Symposium on High-Performance Computer Architecture (HPCA), Vienna, Austria, February 2018.
[[Lightning Talk Video](#)]
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]
[[Full Talk Lecture Video](#) (28 minutes)]

The DRAM Latency PUF:

Quickly Evaluating Physical Unclonable Functions

by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim^{†§}

Minesh Patel[§]

Hasan Hassan[§]

Onur Mutlu^{§†}

[†]Carnegie Mellon University

[§]ETH Zürich

In-DRAM Lookup-Table Based Execution

João Dinis Ferreira, Gabriel Falcao, Juan Gómez-Luna, Mohammed Alser, Lois Orosa, Mohammad Sadrosadati, Jeremie S. Kim, Geraldo F. Oliveira, Taha Shahroodi, Anant Nori, and Onur Mutlu,

"pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables"

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

[[Slides \(pptx\)](#)] ([pdf](#))

[[Longer Lecture Slides \(pptx\)](#)] ([pdf](#))

[[Lecture Video](#) (26 minutes)]

[[arXiv version](#)]

[[Source Code](#) (Officially Artifact Evaluated with All Badges)]

Officially artifact evaluated as available, reusable and reproducible.



pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables

João Dinis Ferreira[§]

Gabriel Falcao[†]

Juan Gómez-Luna[§]

Mohammed Alser[§]

Lois Orosa^{§∇}

Mohammad Sadrosadati[§]

Jeremie S. Kim[§]

Geraldo F. Oliveira[§]

Taha Shahroodi[‡]

Anant Nori^{*}

Onur Mutlu[§]

[§]ETH Zürich

[†]IT, University of Coimbra

[∇]Galicia Supercomputing Center

[‡]TU Delft

^{*}Intel

What About Other Types of Memories?

In-Flash Bulk Bitwise Execution

- Jisung Park, Roknoddin Azizi, Geraldo F. Oliveira, Mohammad Sadrosadati, Rakesh Nadig, David Novo, Juan Gómez-Luna, Myungsuk Kim, and Onur Mutlu,
"Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory"
Proceedings of the 55th International Symposium on Microarchitecture (MICRO),
Chicago, IL, USA, October 2022.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Longer Lecture Slides \(pptx\)](#)] [[pdf](#)]
[[Lecture Video](#) (44 minutes)]
[[arXiv version](#)]

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

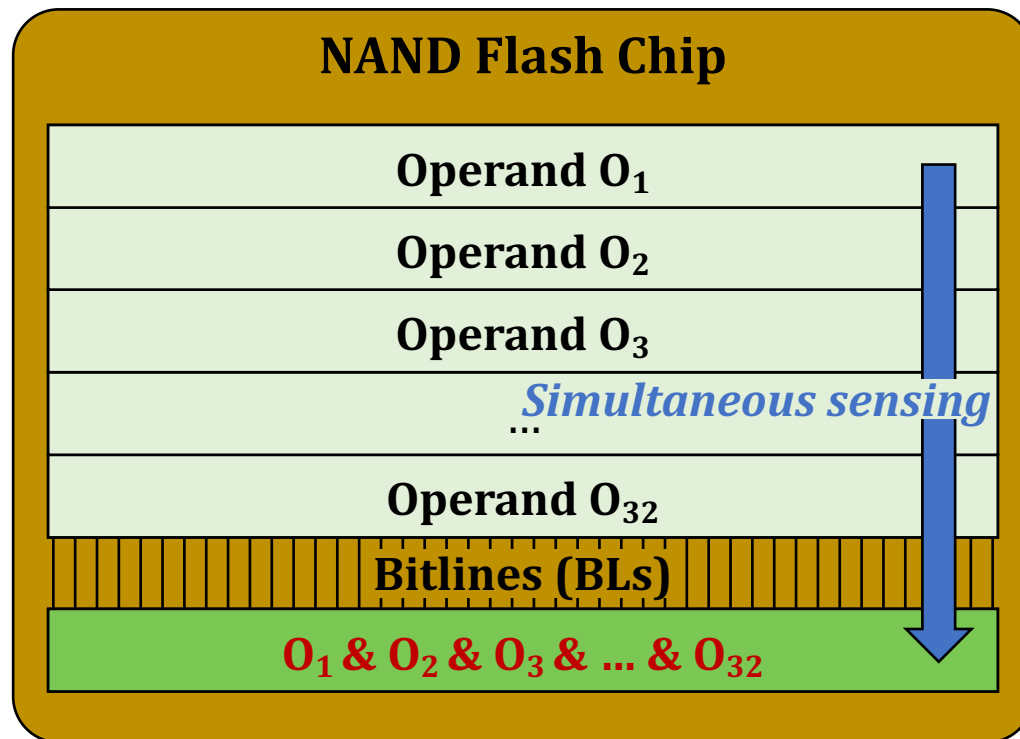
Jisung Park^{§∇} Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

[§]ETH Zürich [∇]POSTECH [†]LIRMM, Univ. Montpellier, CNRS [‡]Kyungpook National University

Flash-Cosmos: Basic Ideas

▪ Flash-Cosmos enables

- Computation on multiple operands with a single sensing operation
- Accurate computation results by eliminating raw bit errors in stored data



Multi-Wordline Sensing (MWS): Bitwise AND

■ Intra-Block MWS:

Simultaneously activates multiple WLs in the same block

→ Bitwise AND of the stored data in the WLs

A bitline reads as '**1**' only when all the target cells store '**1**'
→ Equivalent to the bitwise AND of all the target cells

*Operate
as a resistance (1)
or an open switch (0)*

WL₂
WL₃
WL₄

Result: 0

0

0

0

Multi-Wordline Sensing (MWS): Bitwise AND

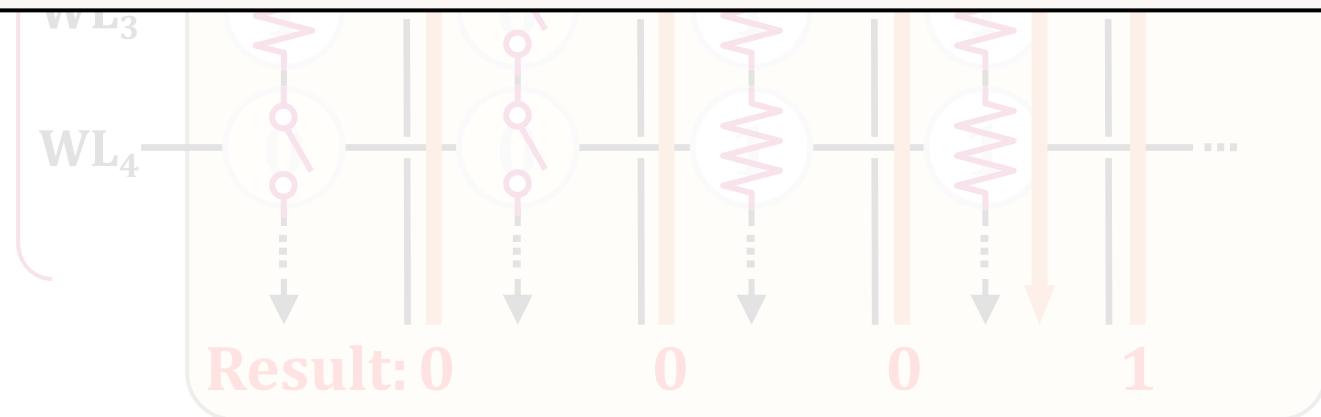
- Intra-Block MWS:

Simultaneously activates multiple WLs in the same block

→ Bitwise AND of the stored data in the WLs



Flash-Cosmos (Intra-Block MWS) enables bitwise AND of multiple pages in the same block via a single sensing operation



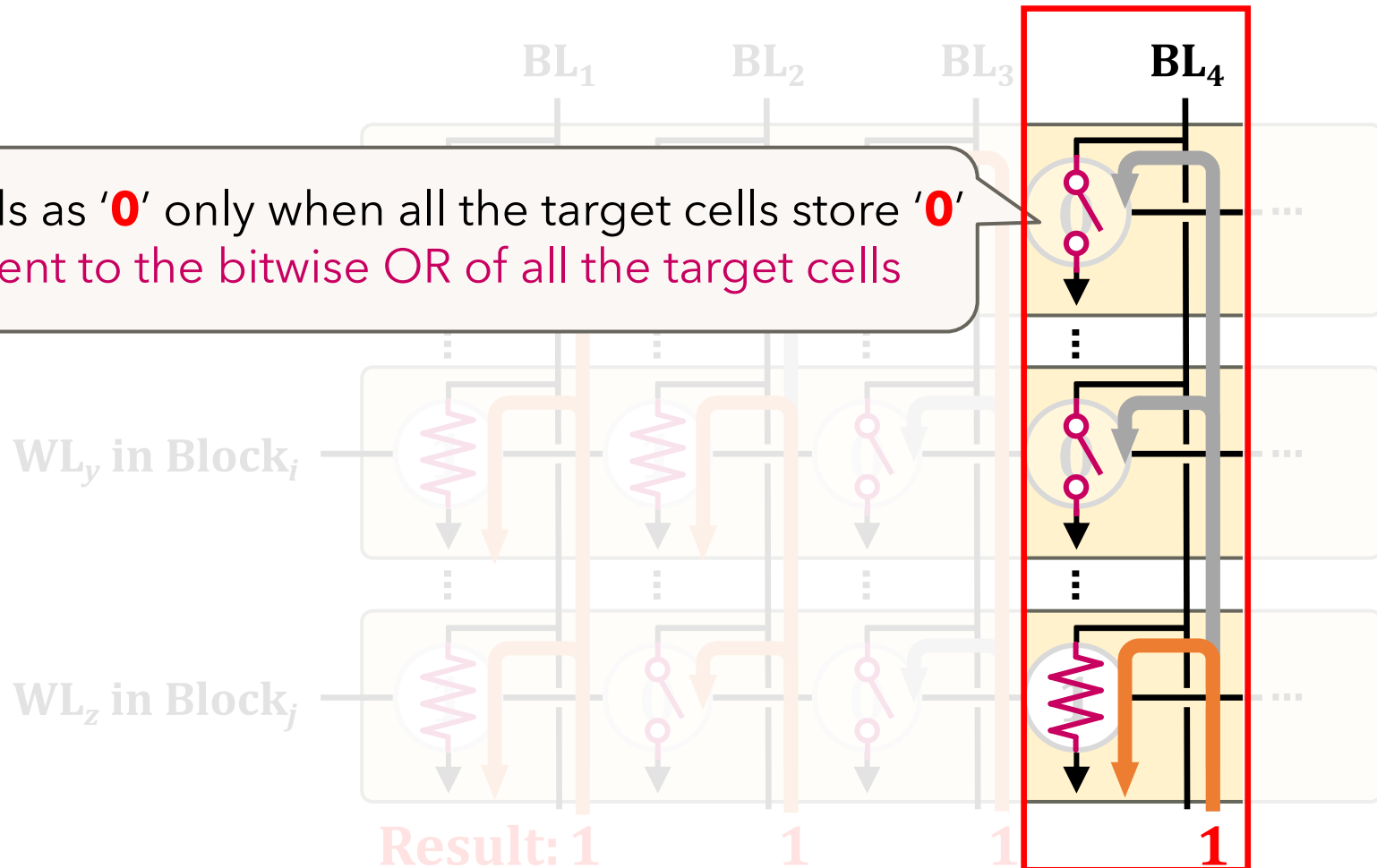
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS:**

Simultaneously activates multiple WLs in different blocks

→ Bitwise OR of the stored data in the WLs

A bitline reads as '**0**' only when all the target cells store '**0**'
→ Equivalent to the bitwise OR of all the target cells

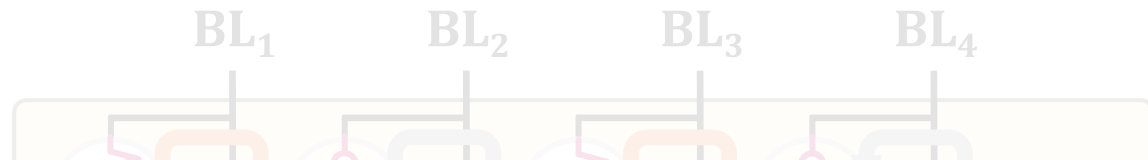


Multi-Wordline Sensing (MWS): Bitwise OR

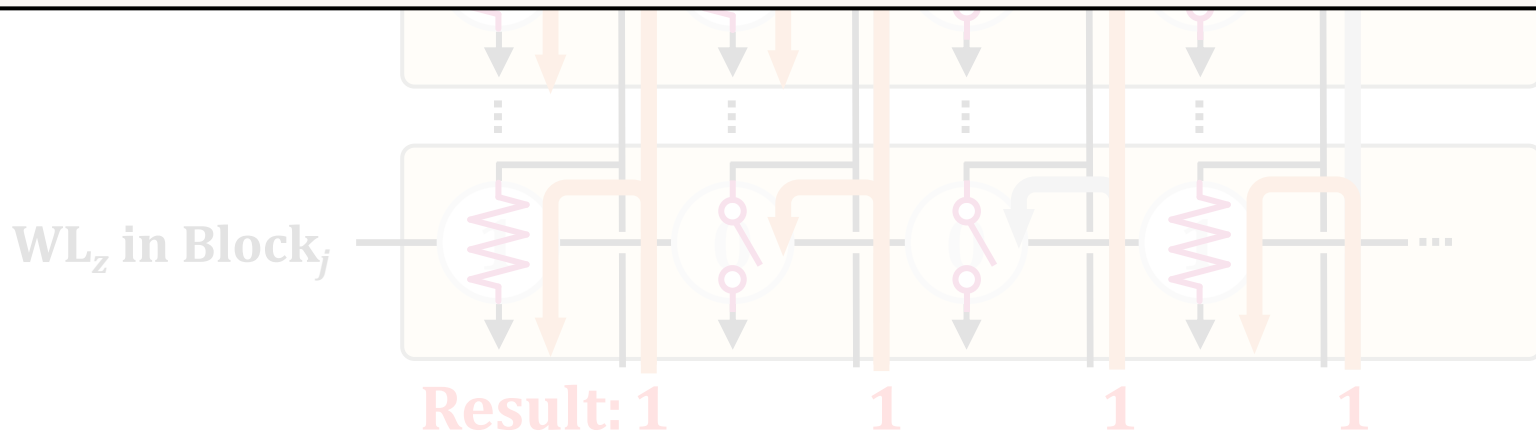
■ Inter-Block MWS:

Simultaneously activates multiple WLs in different blocks

→ Bitwise OR of the stored data in the WLs



Flash-Cosmos (Inter-Block MWS) enables
bitwise OR of multiple pages in different blocks
via a single sensing operation



Other Types of Bitwise Operations

Flash-Cosmos also enables
other types of bitwise operations
(NOT/NAND/NOR/XOR/XNOR)
leveraging **existing features** of NAND flash memory

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

Jisung Park^{§∇} Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

[§]*ETH Zürich* [∇]*POSTECH* [†]*LIRMM, Univ. Montpellier, CNRS* [‡]*Kyungpook National University*



<https://arxiv.org/abs/2209.05566.pdf>

Enhanced SLC-Mode Programming (ESP)

- **Goal:** eliminate raw bit errors in stored data (and computation results)
- **Key ideas**
 - Programs only a single bit per cell (SLC-mode programming)
 - Trades storage density for reliable computation
 - Performs more precise programming of the cells
 - Trades programming latency for reliable computation

Maximizes the reliability margin
between the different states of flash cells

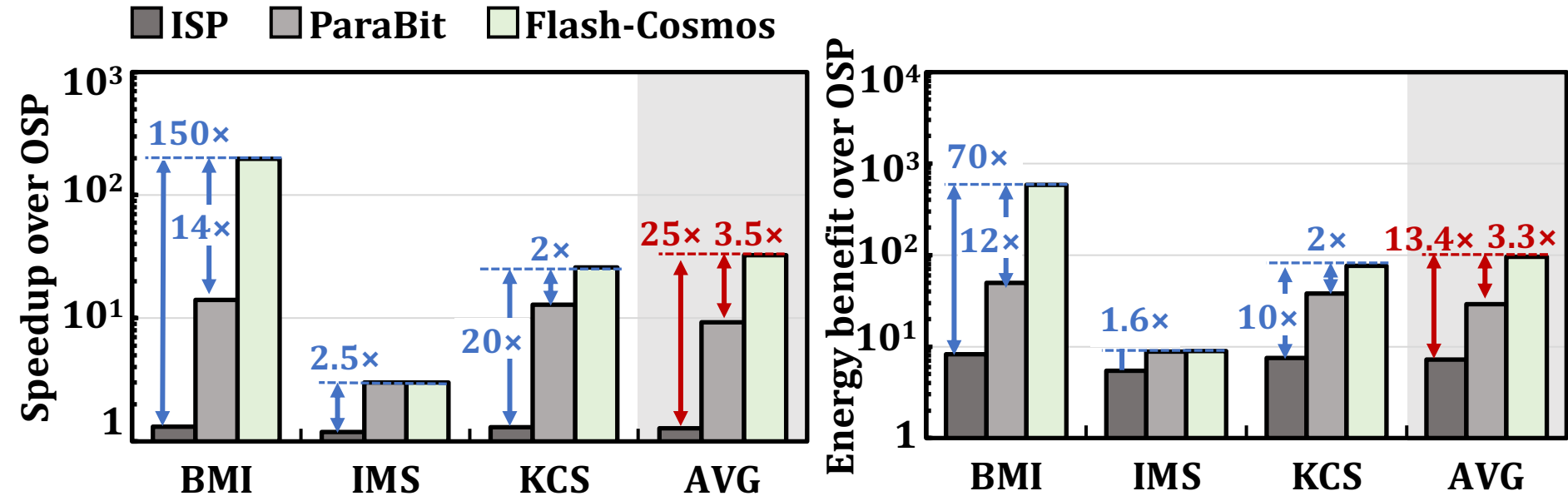
Results: Real-Device Characterization

No changes to the cell array
of commodity NAND flash chips

Can have many operands
(AND: up to 48, OR: up to 4)
with small increase in sensing latency ($< 10\%$)

ESP significantly improves
the reliability of computation results
(no observed bit error in the tested flash cells)

Results: Performance & Energy



Flash-Cosmos provides **significant performance & energy benefits** over all the baselines

The larger the number of operands,
the higher the performance & energy benefits

Pinatubo: RowClone and Bitwise Ops in PCM

Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li^{1*}, Cong Xu², Qiaosha Zou^{1,5}, Jishen Zhao³, Yu Lu⁴, and Yuan Xie¹

University of California, Santa Barbara¹, Hewlett Packard Labs²

University of California, Santa Cruz³, Qualcomm Inc.⁴, Huawei Technologies Inc.⁵
{shuangchenli, yuanxie}@ece.ucsb.edu¹

Pinatubo: RowClone and Bitwise Ops in PCM

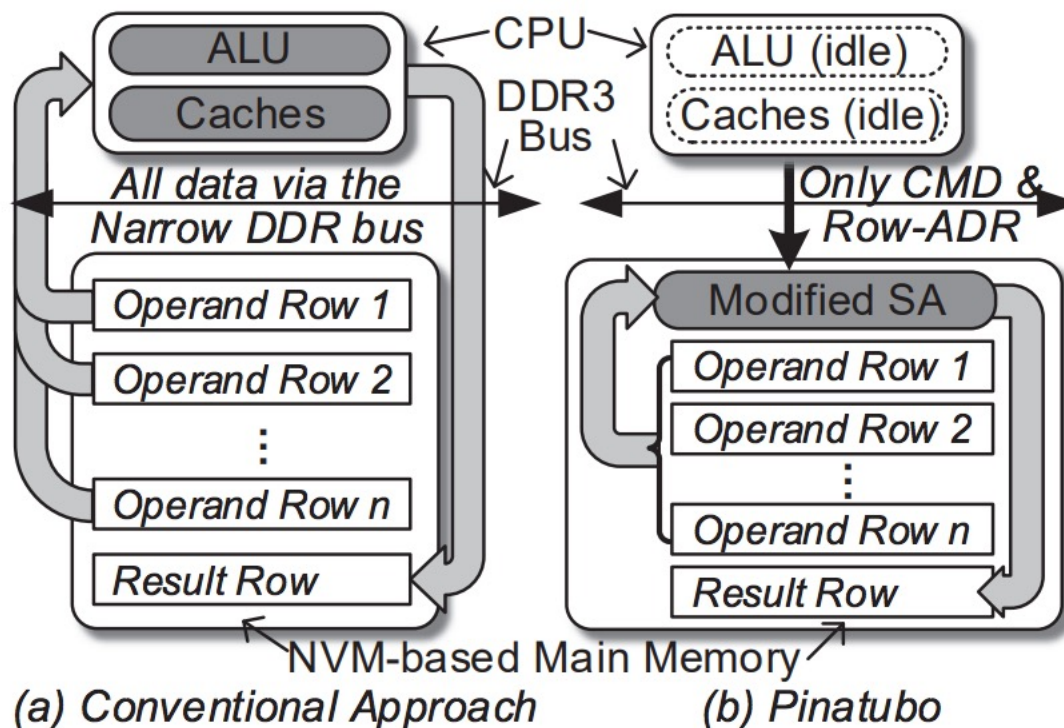
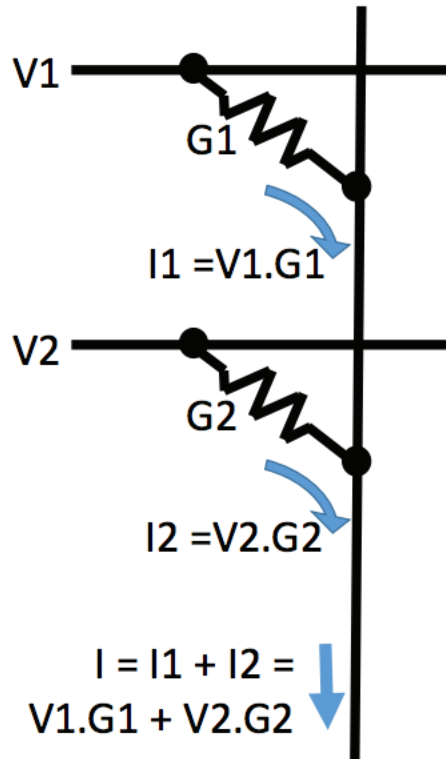


Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n -row bitwise operations inside NVM in one step.

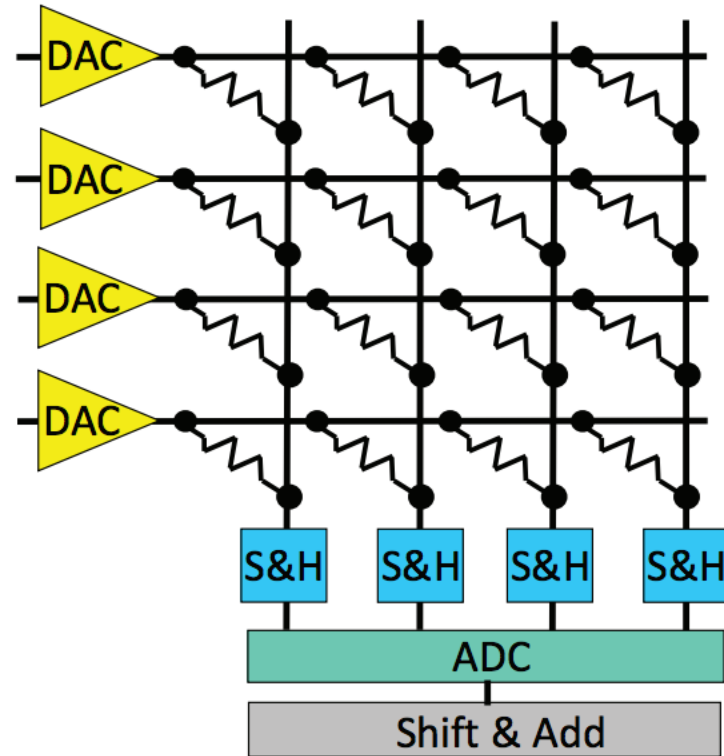
In-Memory Crossbar Array Operations

- Some emerging NVM technologies have crossbar array structure
 - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
- Crossbar arrays can be used to perform dot product operations using “analog computation capability”
 - Can operate on multiple pieces of data using Kirchhoff's laws
 - Bitline current is a sum of products of wordline $V \times (1 / \text{cell } R)$
 - Computation is in analog domain inside the crossbar array
- Need peripheral circuitry for $D \rightarrow A$ and $A \rightarrow D$ conversion of inputs and outputs

In-Memory Crossbar Computation



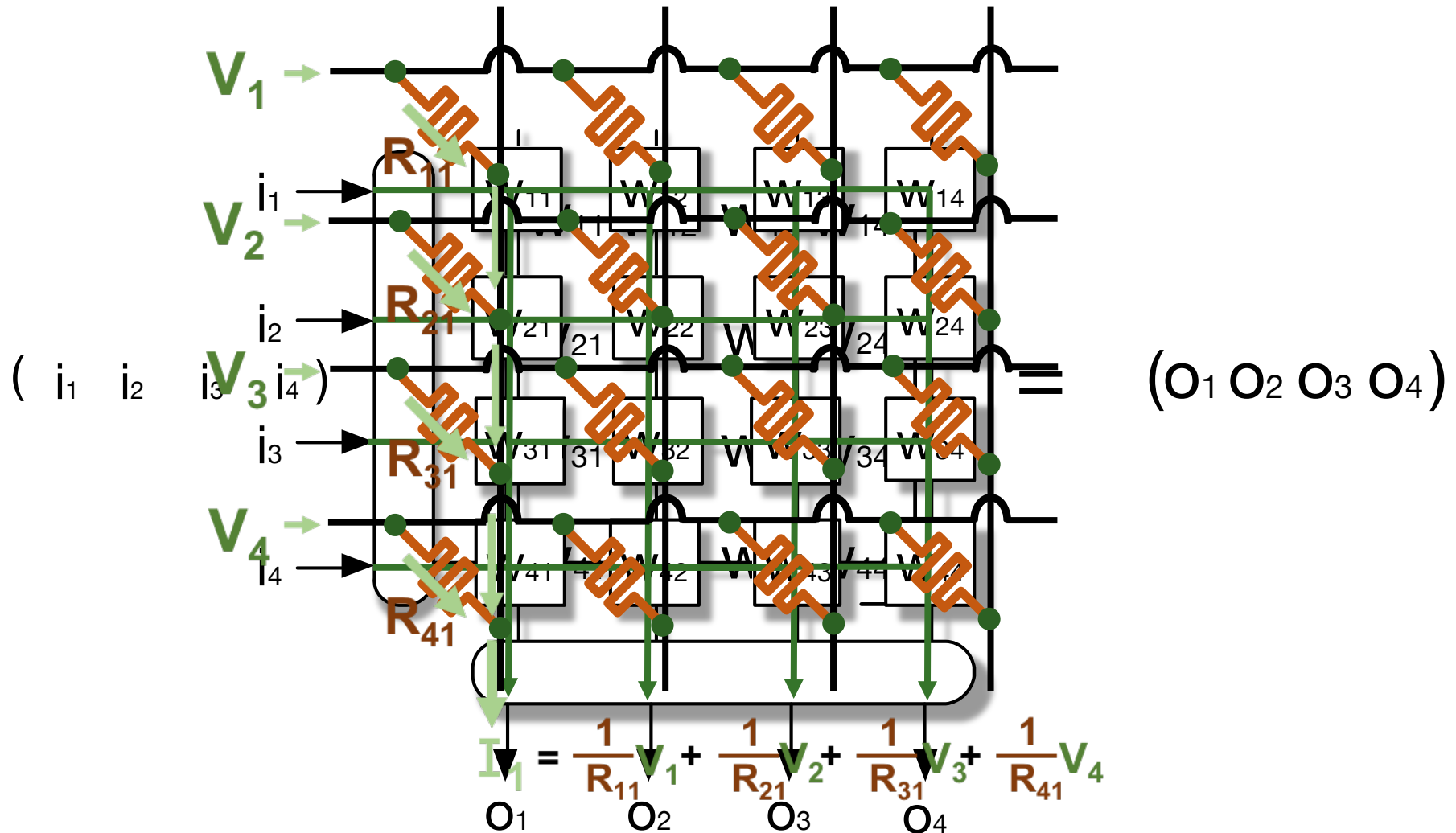
(a) Multiply-Accumulate operation



(b) Vector-Matrix Multiplier

Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

In-Memory Crossbar Computation



Other Readings on Processing using NVM

- Shafiee+, “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars”, ISCA 2016.
- Chi+, “PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory”, ISCA 2016.
- Prezioso+, “Training and Operation of an Integrated Neuromorphic Network based on Metal-Oxide Memristors”, Nature 2015
- Ambrogio+, “Equivalent-accuracy accelerated neural-network training using analogue memory”, Nature 2018.

Processing in Memory: Two Approaches

1. Processing using Memory
2. **Processing near Memory**

PIM Review and Open Problems

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

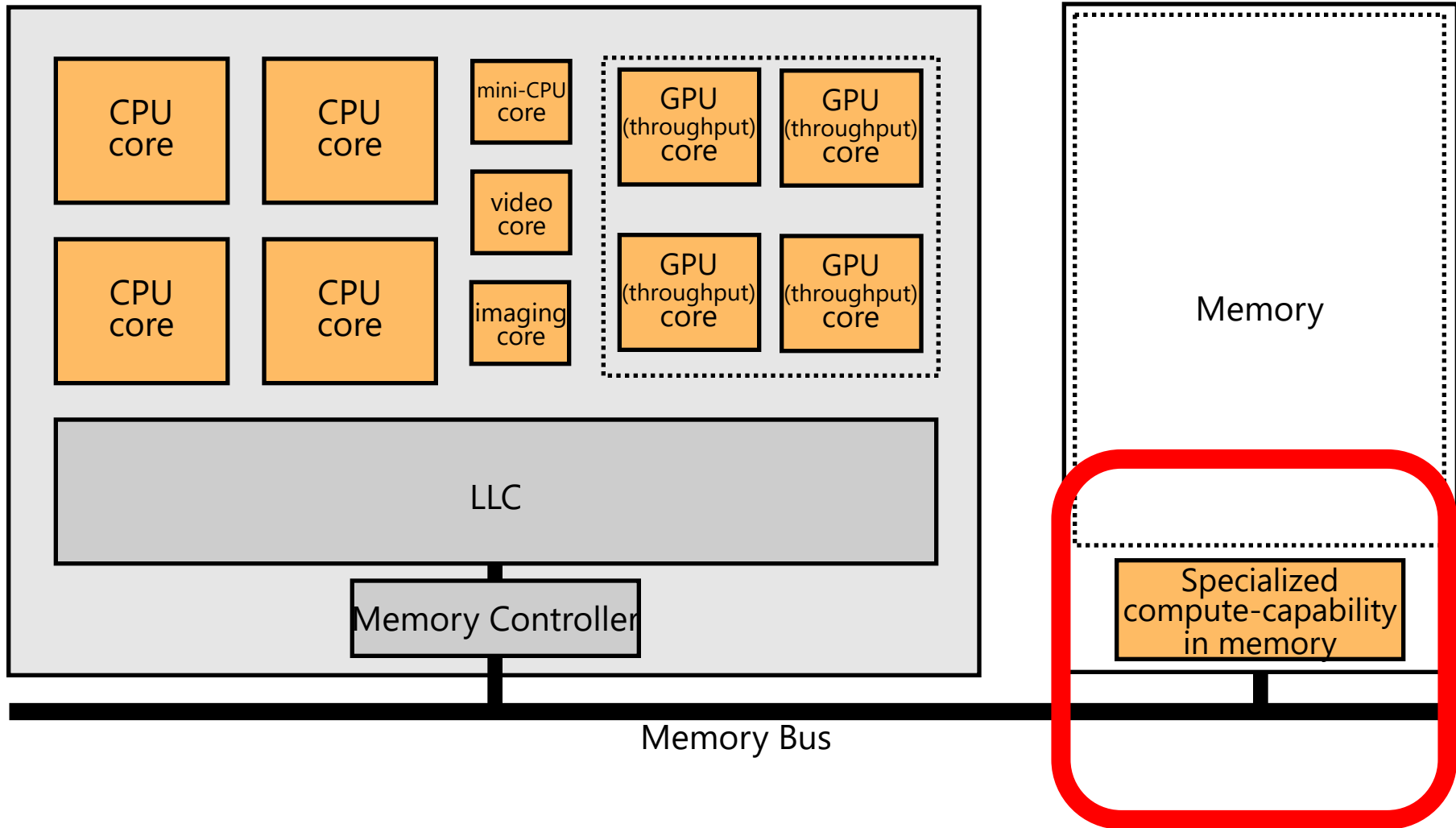
^d*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

"A Modern Primer on Processing in Memory"

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Accelerating In-Memory Graph Analytics

- Large graphs are everywhere (circa 2015)



36 Million
Wikipedia Pages



1.4 Billion
Facebook Users

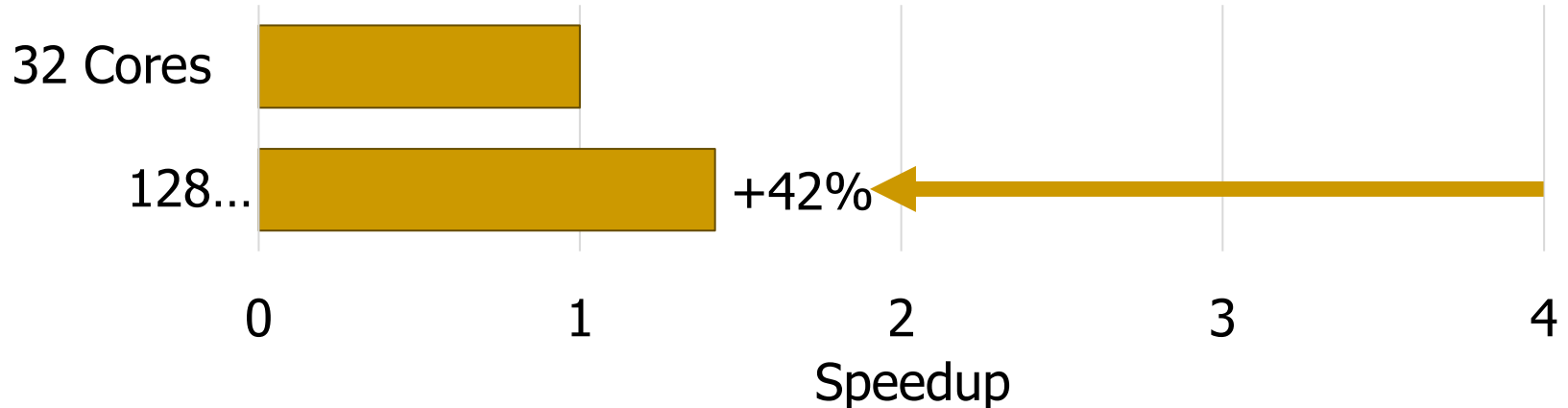


300 Million
Twitter Users



30 Billion
Instagram Photos

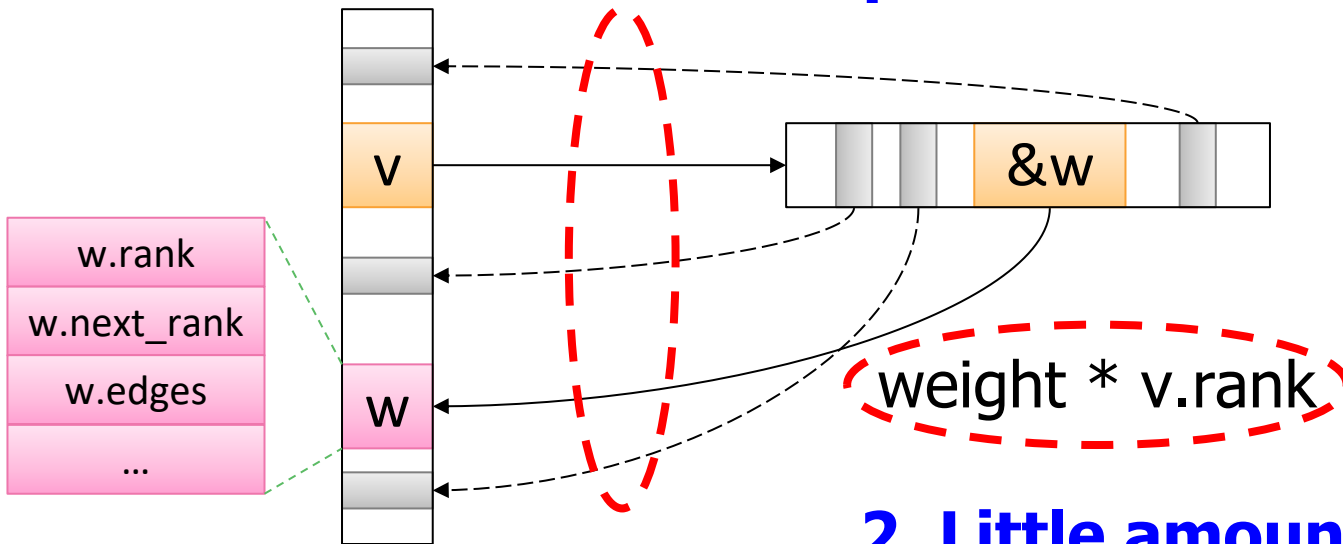
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

1. Frequent random memory accesses

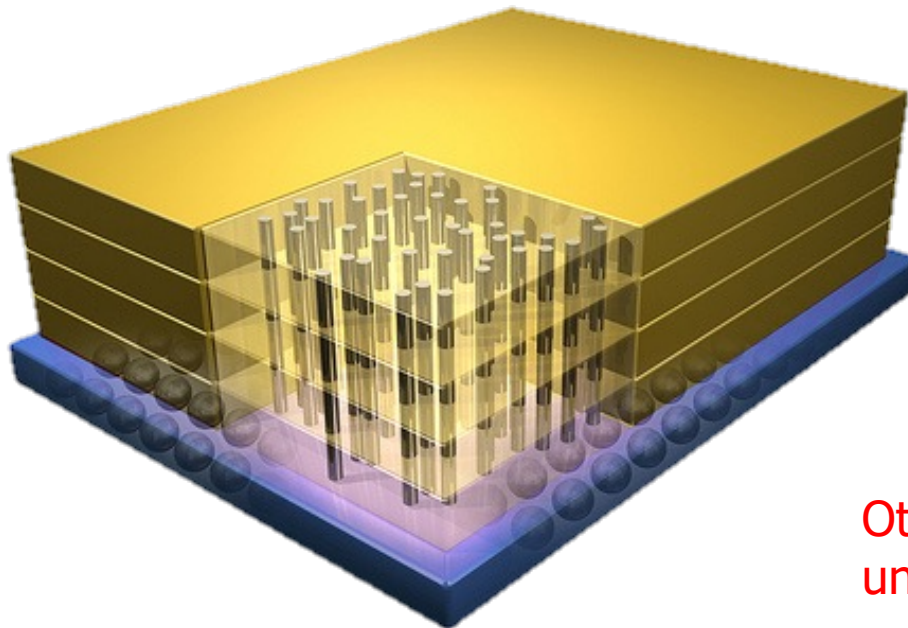


2. Little amount of computation

Opportunity: 3D-Stacked Logic+Memory



Hybrid Memory Cube
C O N S O R T I U M



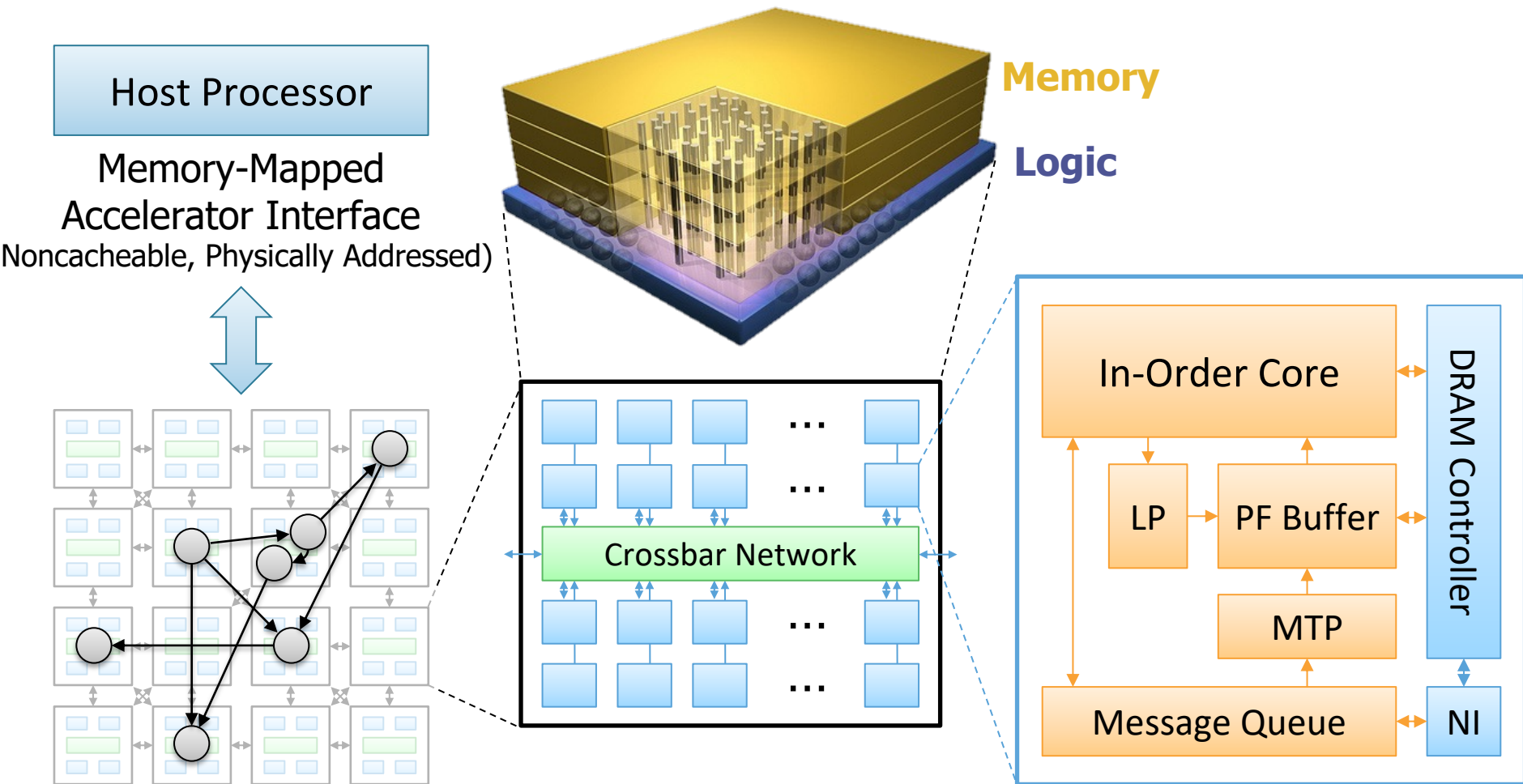
Memory

Logic

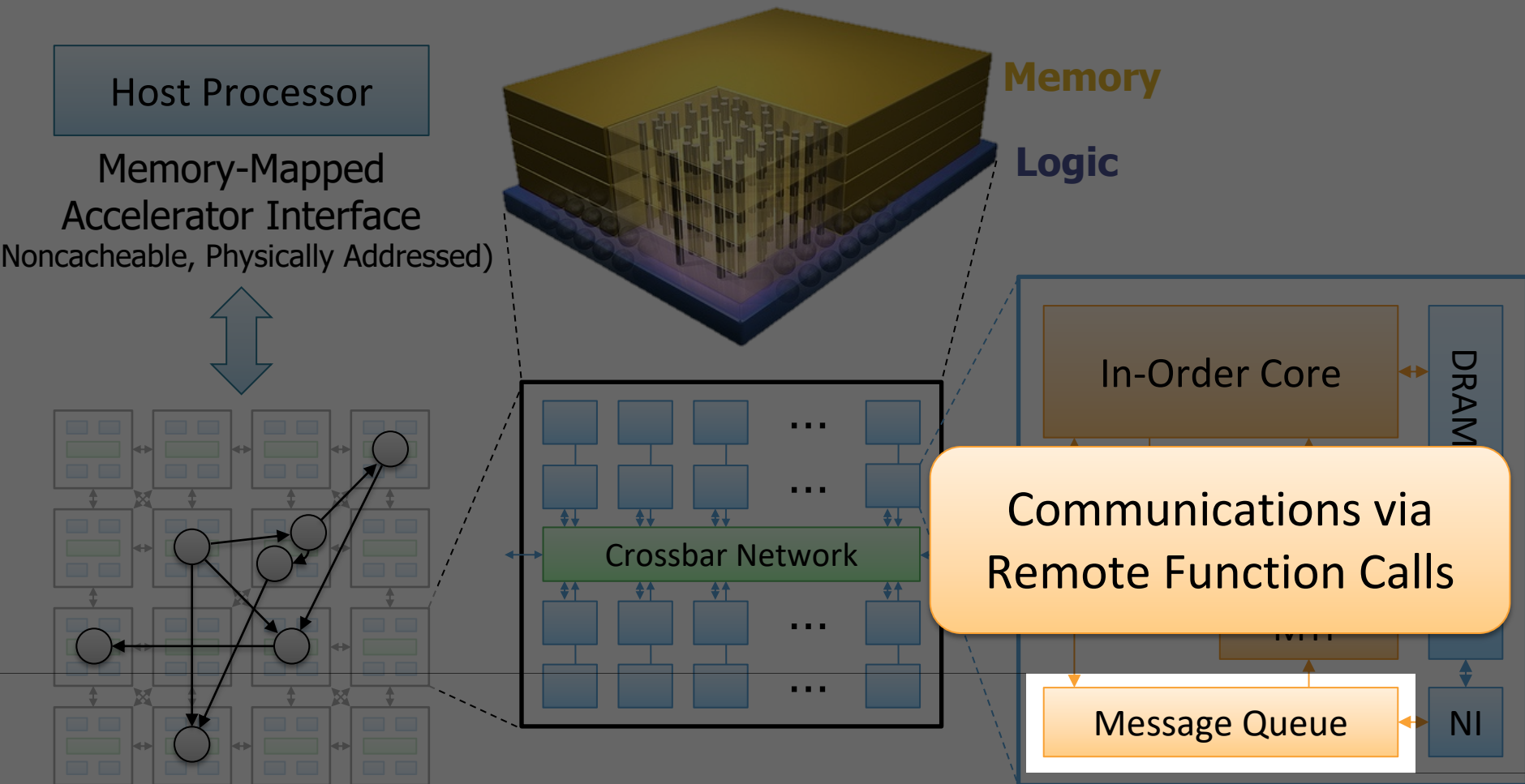
Other "True 3D" technologies
under development

Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores

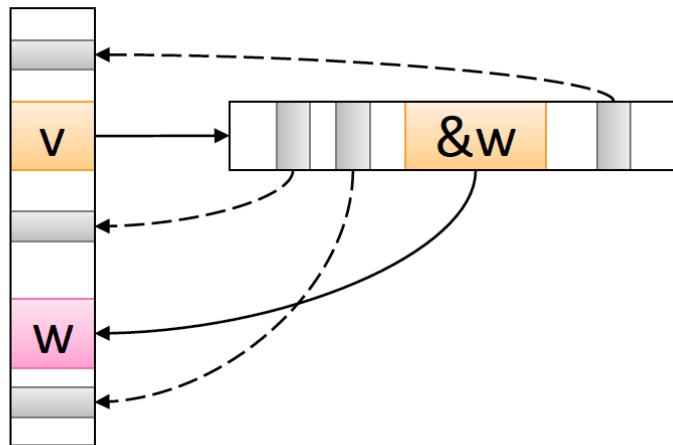


Tesseract System for Graph Processing



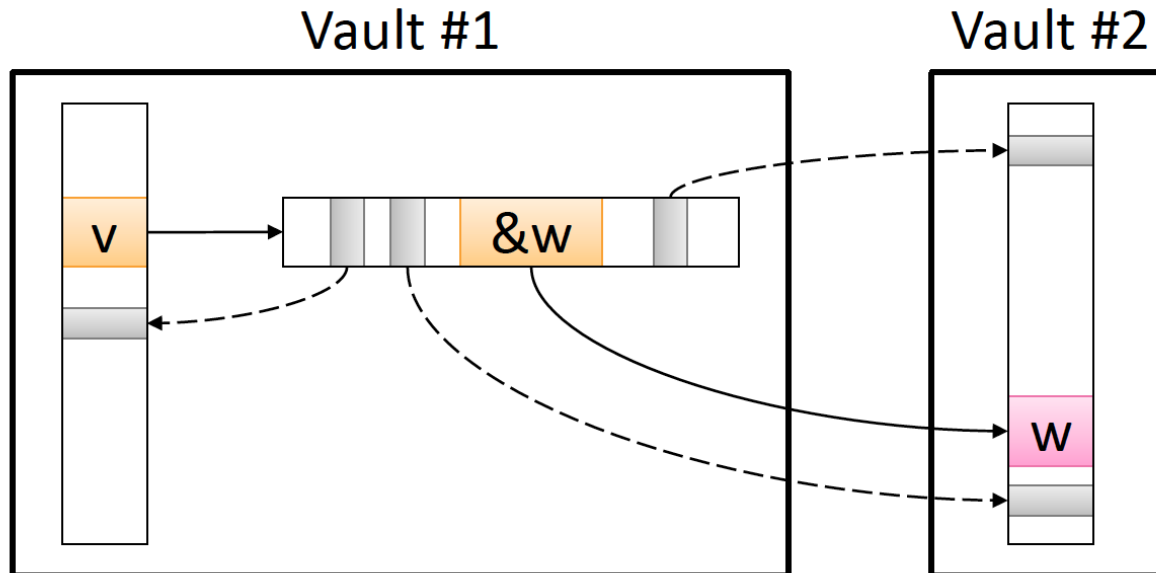
Communications In Tesseract (I)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```



Communications In Tesseract (II)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

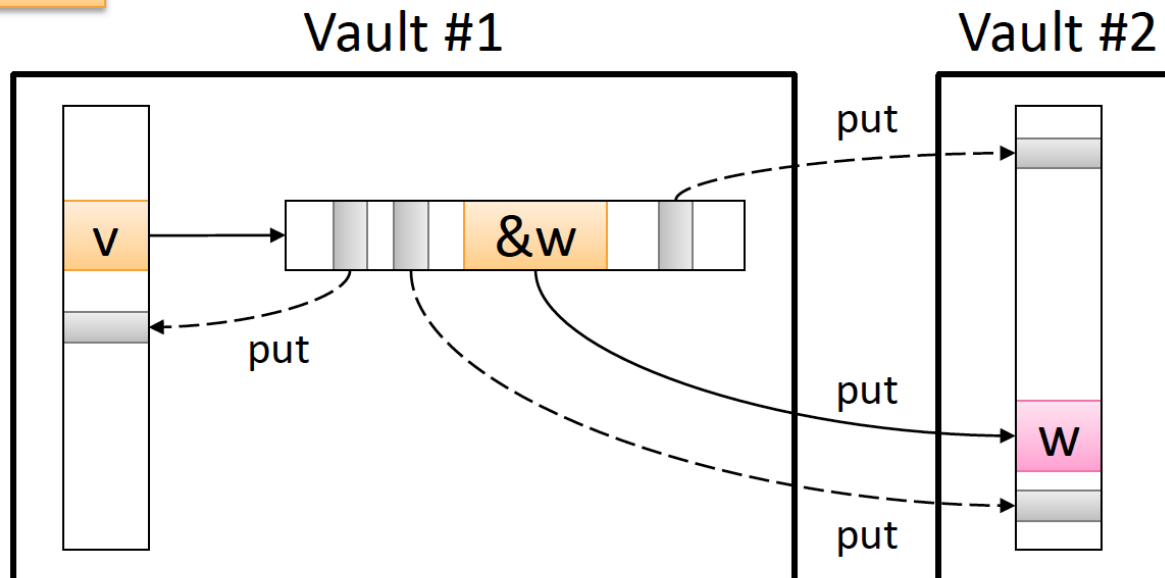


Communications In Tesseract (III)

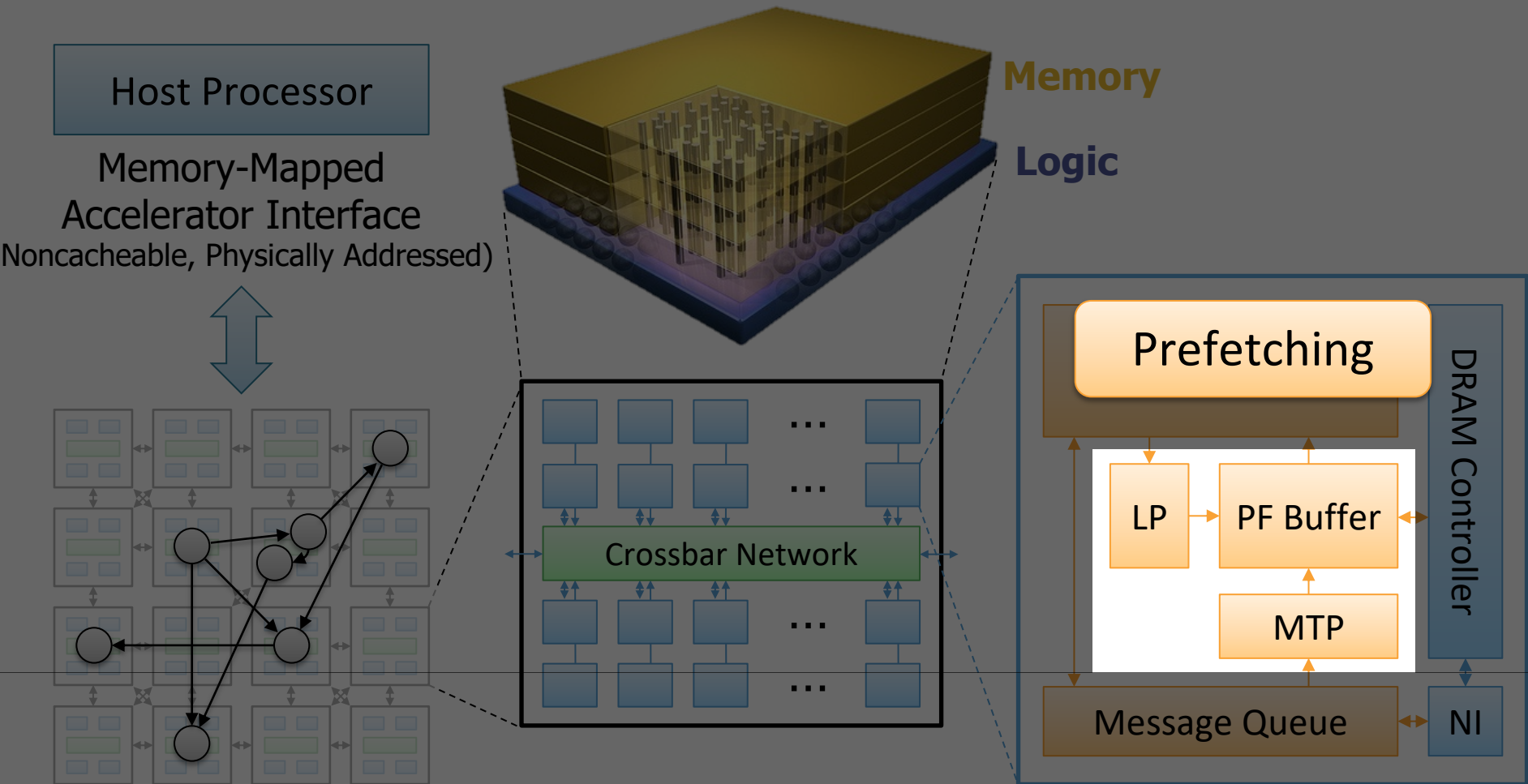
```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    put(w.id, function() { w.next_rank += weight * v.rank; });  
  }  
}  
barrier();
```

Non-blocking Remote Function Call

Can be **delayed**
until the nearest barrier

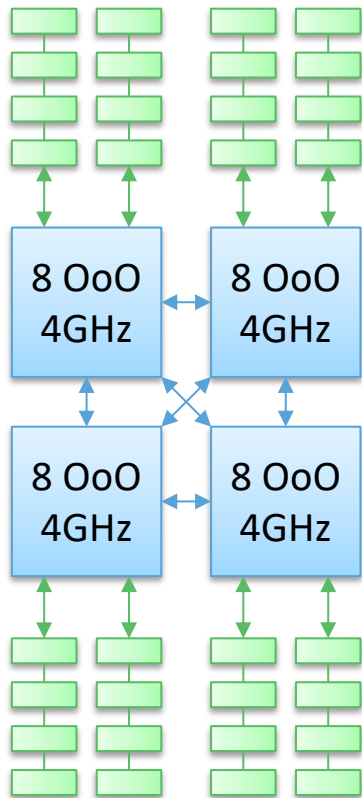


Tesseract System for Graph Processing



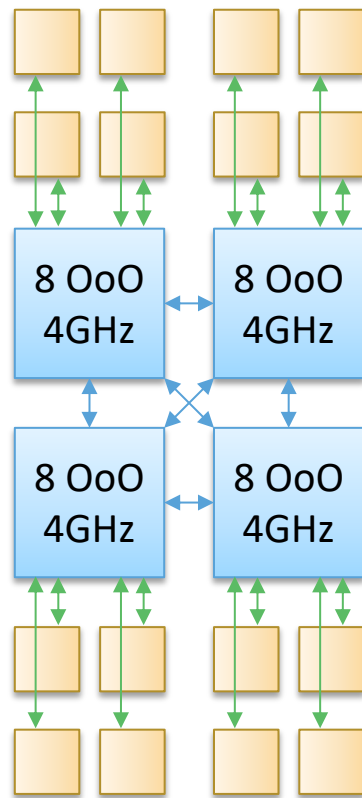
Evaluated Systems

DDR3-OoO



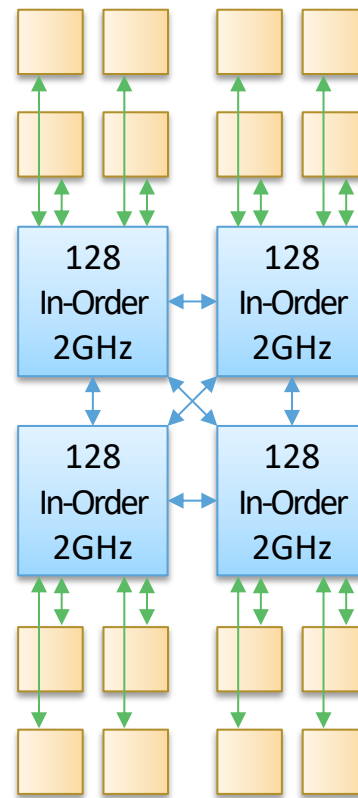
102.4GB/s

HMC-OoO



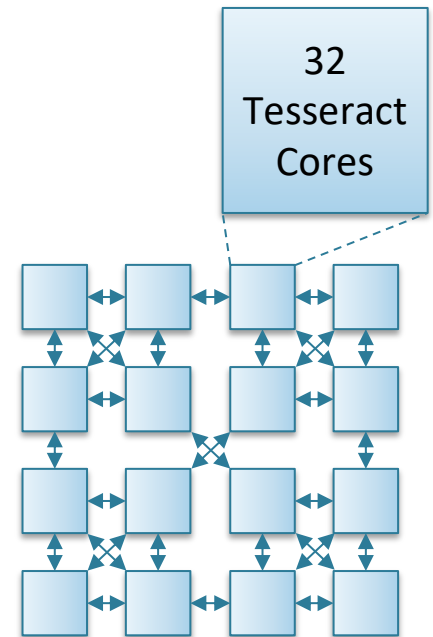
640GB/s

HMC-MC



640GB/s

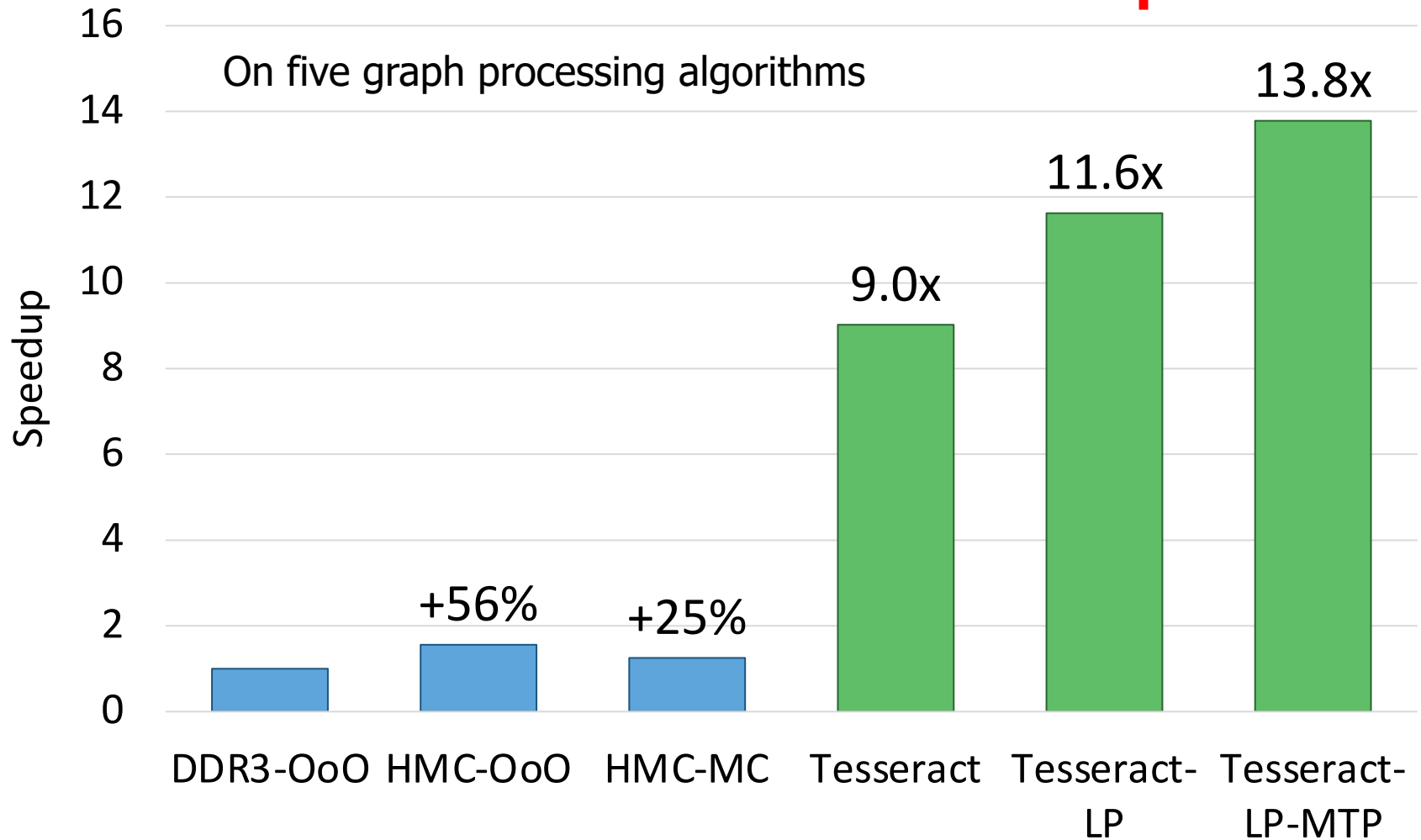
Tesseract



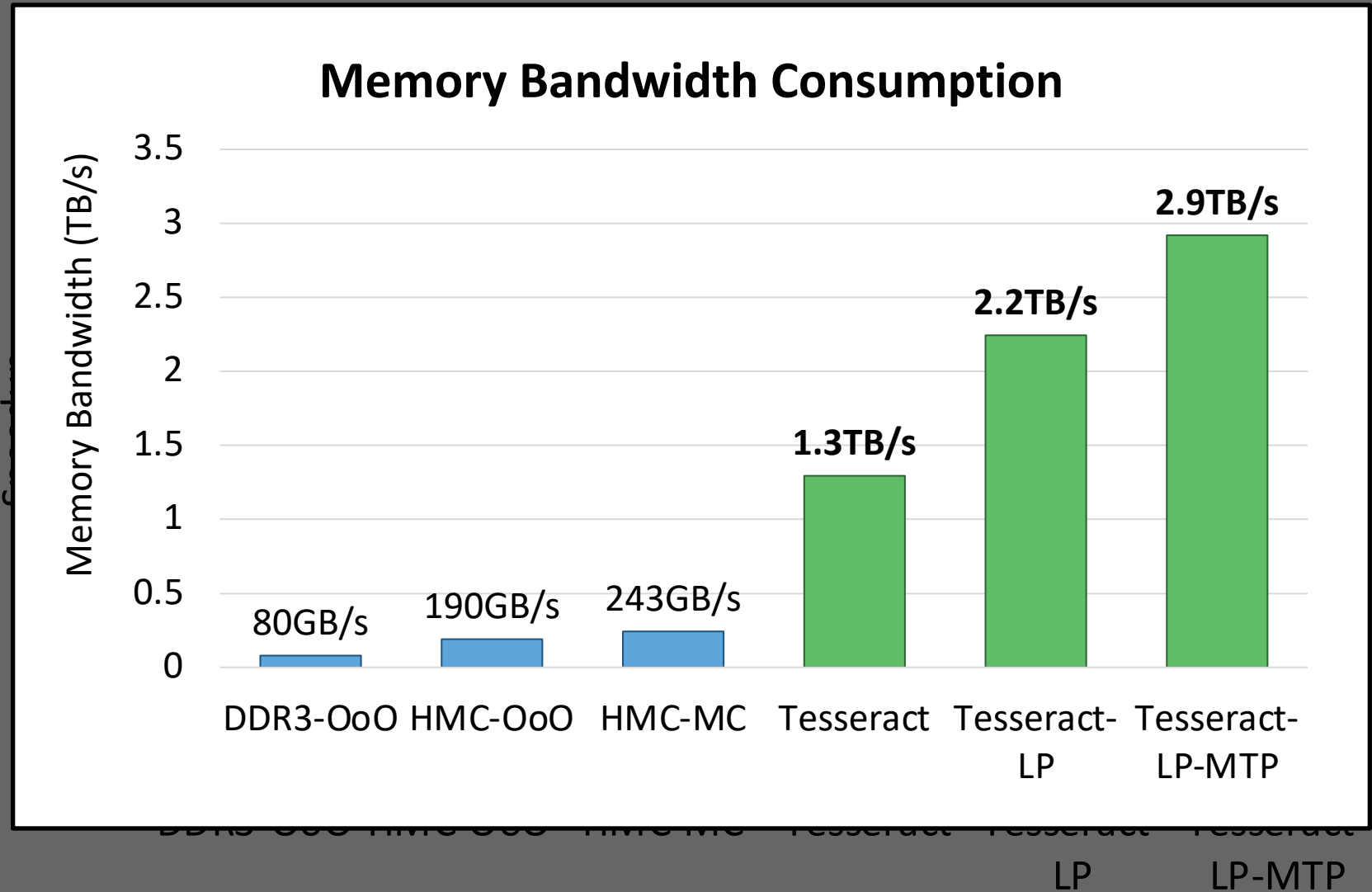
8TB/s

Tesseract Graph Processing Performance

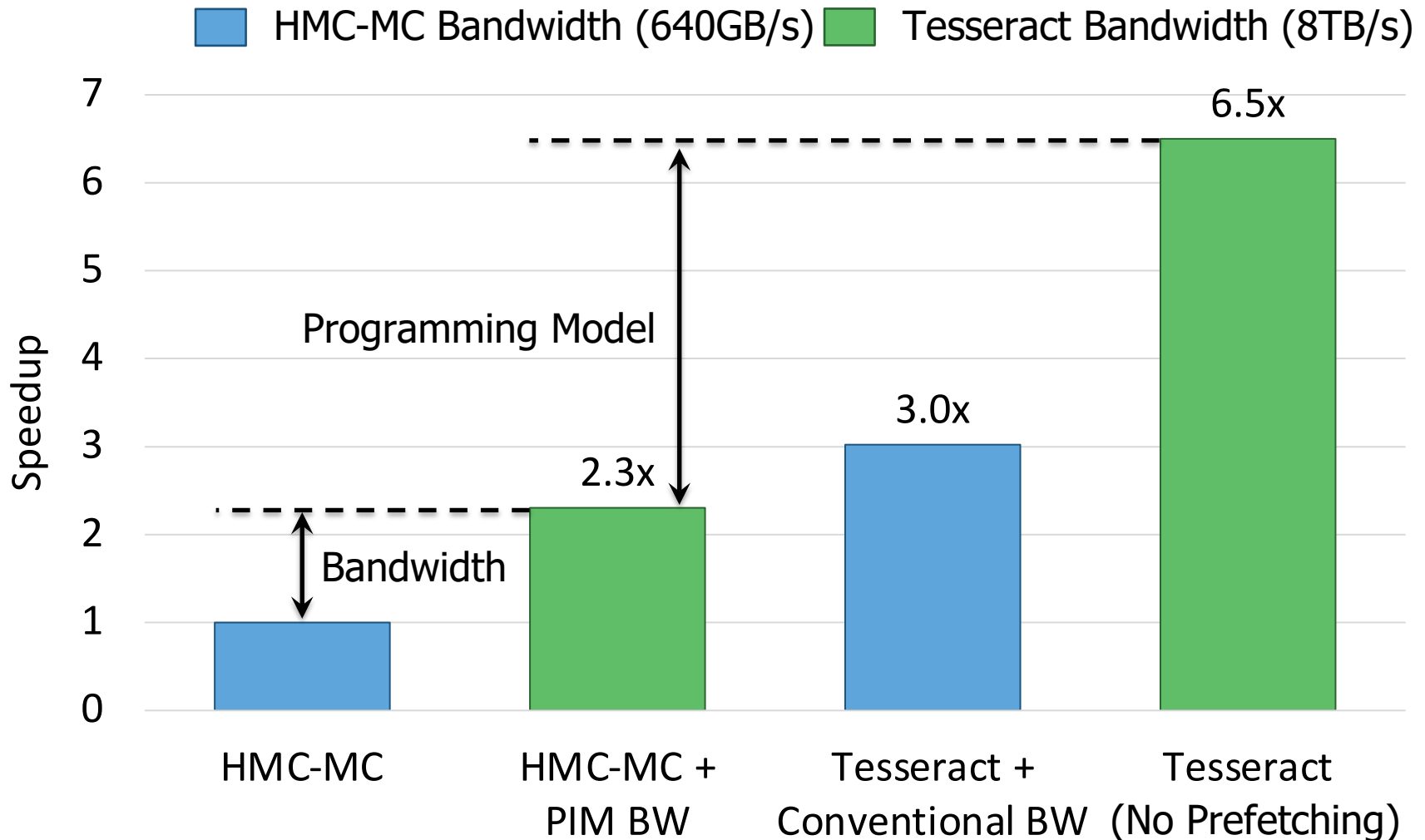
>13X Performance Improvement



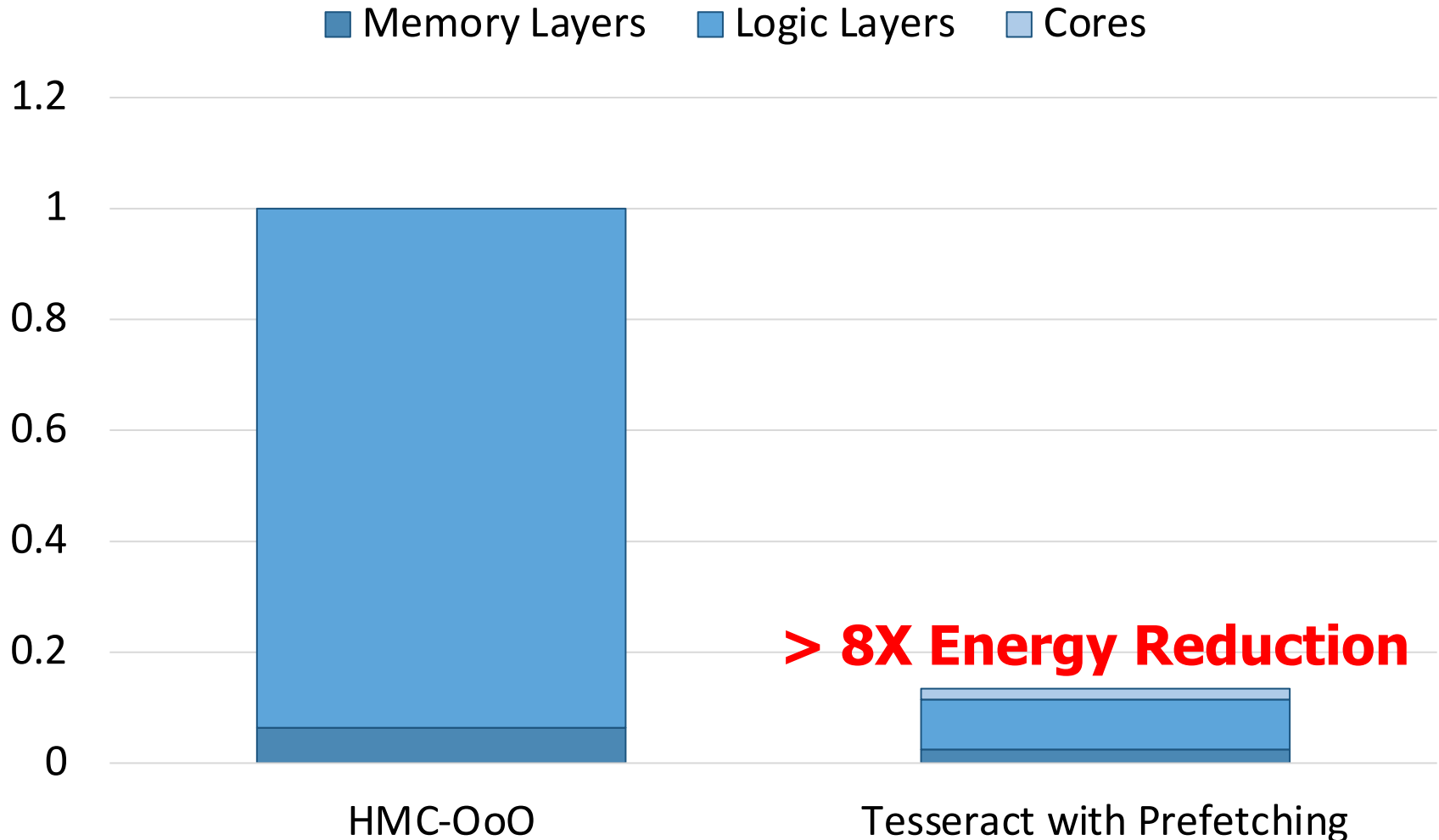
Tesseract Graph Processing Performance



Effect of Bandwidth & Programming Model



Tesseract Graph Processing System Energy



More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoun Choi,
"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
Proceedings of the 42nd International Symposium on Computer Architecture (ISCA), Portland, OR, June 2015.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
Top Picks Honorable Mention by IEEE Micro.
Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn Sungpack Hong[§] Sungjoo Yoo Onur Mutlu[†] Kiyoun Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[§]Oracle Labs

[†]Carnegie Mellon University

A Short Retrospective @ 50 Years of ISCA

Retrospective: A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn[†] Sungpack Hong[‡] Sungjoo Yoo[▽] Onur Mutlu[§] Kiyoung Choi[▽]
[†]Google DeepMind [‡]Oracle Labs [§]ETH Zürich [▽]Seoul National University

Abstract—Our ISCA 2015 paper [1] provides a new programmable processing-in-memory (PIM) architecture and system design that can accelerate key data-intensive applications, with a focus on graph processing workloads. Our major idea was to completely rethink the system, including the programming model, data partitioning mechanisms, system support, instruction set architecture, along with near-memory execution units and their communication architecture, such that an important workload can be accelerated at a maximum level using a distributed system of well-connected near-memory accelerators. We built our accelerator system, Tesseract, using 3D-stacked memories with logic layers, where each logic layer contains general-purpose processing cores and control logic for each core using a message-passing programming model. Cores could be specialized for graph processing (or any other application to be accelerated).

To our knowledge, our paper was the first to completely design a near-memory accelerator system from scratch such that it is both generally programmable and specifically customizable to accelerate important applications, with a case study on major graph processing workloads. Existing work in academia and industry showed that similar approaches to system design can greatly benefit both graph processing workloads and other applications, such as machine learning, for which ideas from Tesseract seem to have been influential.

This short retrospective provides a brief analysis of our ISCA 2015 paper and its impact. We briefly describe the major ideas and contributions of the work, discuss later works that built on it or were influenced by it, and make some educated guesses on what the future may bring on PIM and accelerator systems.

I. BACKGROUND, APPROACH & MINDSET

We started our research when 3D-stacked memories (e.g., [2–4]) were viable and seemed to have promise for building effective and practical processing-near-memory systems. Such near-memory systems could lead to improvements, but there was little to no research that examined how an accelerator could be completely (re-)designed using such near-memory technology, from its hardware architecture to its programming model and software system, and what the performance and energy benefits could be of such a re-design. We set out to answer these questions in our ISCA 2015 paper [1].

We followed several major principles to design our accelerator from the ground up. We believe these principles are still important: a major contribution and influence of our work was in putting all of these together in a cohesive full-system design and demonstrating the large performance and energy benefits that can be obtained from such a design. We see a similar approach in many modern large-scale accelerator systems in machine learning today (e.g., [5–9]). Our principles are:

1. *Near-memory execution* to enable/exploit the high data access bandwidth modern workloads (e.g., graph processing) need and to reduce data movement and access latency.
2. *General programmability* so that the system can be easily adopted, extended, and customized for many workloads.
3. *Maximal acceleration capability* to maximize the performance and energy benefits. We set ourselves free from backward compatibility and cost constraints. We aimed to completely re-design the system stack. Our goal was to explore the maximal performance and energy efficiency benefits we can gain from a near-memory accelerator if we had complete freedom to change things as much as we needed. We contrast this approach to the *minimal intrusion* approach we also explored in a separate ISCA 2015 paper [10].
4. *Customizable to specific workloads*, such that we can maximize acceleration benefits. Our focus workload was graph

analytics/processing, a key workload at the time and today. However, our design principles are not limited to graph processing and the system we built is customizable to other workloads as well, e.g., machine learning, genome analysis.

5. *Memory-capacity-proportional performance*, i.e., processing capability should proportionally grow (i.e., scale) as memory capacity increases and vice versa. This enables scaling of data-intensive workloads that need both memory and compute.

6. *Exploit new technology (3D stacking)* that enables tight integration of memory and logic and helps multiple above principles (e.g., enables customizable near-memory acceleration capability in the logic layer of a 3D-stacked memory chip).

7. *Good communication and scaling capability* to support scalability to large dataset sizes and to enable memory-capacity-proportional performance. To this end, we provided scalable communication mechanisms between execution cores and carefully interconnected small accelerator chips to form a large distributed system of accelerator chips.

8. *Maximal and efficient use of memory bandwidth* to supply the high-bandwidth data access that modern workloads need. To this end, we introduced new, specialized mechanisms for prefetching and a programming model that helps leverage application semantics for hardware optimization.

II. CONTRIBUTIONS AND INFLUENCE

We believe the major contributions of our work were 1) complete rethinking of how an accelerator system should be designed to enable maximal acceleration capability, and 2) the design and analysis of such an accelerator with this mindset and using the aforementioned principles to demonstrate its effectiveness in an important class of workloads.

One can find examples of our approach in modern large-scale machine learning (ML) accelerators, which are perhaps the most successful incarnation of scalable near-memory execution architectures. ML infrastructure today (e.g., [5–9]) consists of accelerator chips, each containing compute units and high-bandwidth memory tightly packaged together, and features scale-up capability enabled by connecting thousands of such chips with high-bandwidth interconnection links. The system-wide rethinking that was done to enable such accelerators and many of the principles used in such accelerators resemble our ISCA 2015 paper's approach.

The “memory-capacity-proportional performance” principle we explored in the paper shares similarities with how ML workloads are scaled up today. Similar to how we carefully sharded graphs across our accelerator chips to greatly improve effective memory bandwidth in our paper, today's ML workloads are sharded across a large number of accelerators by leveraging data/model parallelism and optimizing the placement to balance communication overheads and compute scalability [11, 12]. With the advent of large generative models requiring high memory bandwidth for fast training and inference, the scaling behavior where capacity and bandwidth are scaled together has become an essential architectural property to support modern data-intensive workloads.

The “maximal acceleration capability” principle we used in Tesseract provides much larger performance and energy improvements and better customization than the “minimalist” approach that our other ISCA 2015 paper on *PIM-Enabled Instructions* [10] explored: “minimally change” an existing

system to incorporate (near-memory) acceleration capability to ease programming and keep costs low. So far, the industry has more widely adopted the maximal approach to overcome the pressing scaling bottlenecks of major workloads. The key enabler that bridges the programmability gap between the maximal approach favoring large performance & energy benefits and the minimal approach favoring ease of programming is compilation techniques. These techniques lower well-defined high-level constructs into lower-level primitives [12, 13]; our ISCA 2015 papers [1, 10] and a follow-up work [14] explore them lightly. We believe that a good programming model that enables large benefits coupled with support for it across the entire system stack (including compilers & hardware) will continue to be important for effective near-memory system and accelerator designs [14]. We also believe that the maximal versus minimal approaches that are initially explored in our two ISCA 2015 papers is a useful way of exploring emerging technologies (e.g., near-memory accelerators) to better understand the tradeoffs of system designs that exploit such technologies.

III. INFLUENCE ON LATER WORKS

Our paper was at the beginning of a proliferation of scalable near-memory processing systems designed to accelerate key applications (see [15] for many works on the topic). Tesseract has inspired many near-memory system ideas (e.g., [16–28]) and served as the de facto comparison point for such systems, including near-memory graph processing accelerators that built on Tesseract and improved various aspects of Tesseract. Since machine learning accelerators that use high-bandwidth memory (e.g., [5, 29]) and industrial PIM prototypes (e.g., [30–41]) are now in the market, near-memory processing is no longer an “eccentric” architecture it used to be when Tesseract was originally published.

Graph processing & analytics workloads remain as an important and growing class of applications in various forms, ranging from large-scale industrial graph analysis engines (e.g., [42]) to graph neural networks [43]. Our focus on large-scale graph processing in our ISCA 2015 paper increased attention to this domain in the computer architecture community, resulting in subsequent research on efficient hardware architectures for graph processing (e.g., [44–46]).

IV. SUMMARY AND FUTURE OUTLOOK

We believe that our ISCA 2015 paper's principled rethinking of system design to accelerate an important class of data-intensive workloads provided significant value and enabled/influenced a large body of follow-on works and ideas. We expect that such rethinking of system design for key workloads, especially with a focus on “maximal acceleration capability,” will continue to be critical as pressing technology and application scaling challenges increasingly require us to think differently to substantially improve performance and energy (as well as other metrics). We believe the principles exploited in Tesseract are fundamental and they will remain useful and likely become even more important as systems memory access and computation demands of future workloads. We also project that as hardware substrates for near-memory acceleration (e.g., 3D stacking, in-DRAM computation, NVM-based PIM, processing using memory [15]) evolve and mature, systems will take advantage of them even more, likely using principles similar to those used in the design of Tesseract.

REFERENCES

- [1] J. Ahn *et al.*, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing,” in *ISCA*, 2015.
- [2] Hybrid Memory Cube Consortium, “HMC Specification 1.1,” 2013.
- [3] J. Jeddell and B. Keeth, “Hybrid Memory Cube: New DRAM Architecture Increases Density and Performance,” in *VLSIT*, 2012.
- [4] JEDEC, “High Bandwidth Memory (HBM) DRAM,” Standard No. JESD235, 2013.

- [5] N. Jouppi *et al.*, “TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embedding,” in *ISCA*, 2023.
- [6] J. Fowers *et al.*, “A Configurable Cloud-Scale DNN Processor for Real-Time AI,” in *ISCA*, 2018.
- [7] S. Lie, “Cerebras Architecture Deep Dive: First Look Inside the Hardware/Software Co-Design for Deep Learning,” in *IEEE Micro*, 2023.
- [8] E. Talpes *et al.*, “The Microarchitecture of DIO, Tesla's Exa-Scale Computer,” in *IEEE Micro*, 2023.
- [9] A. Ishii and R. Wells, “NVLink-Network Switch - NVIDIA's Switch Chip for High Communication-Bandwidth SuperPODs,” in *Hot Chips*, 2022.
- [10] J. Ahn *et al.*, “PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture,” in *ISCA*, 2015.
- [11] R. Pope *et al.*, “Efficiently Scaling Transformer Inference,” in *MLSys*, 2023.
- [12] D. Lepikhin *et al.*, “GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding,” in *ICLR*, 2021.
- [13] S. Wang *et al.*, “Overlap Communication with Dependent Computation via Decomposition in Large Deep Learning Models,” in *ASPLOS*, 2023.
- [14] Ahn *et al.*, “AIM: Energy-Efficient Aggregation Inside the Memory Hierarchy,” *ACM TACD*, vol. 13, no. 4, 2016.
- [15] O. Mutlu *et al.*, “A Modern Primer on Processing in Memory,” *Emerging Computing: From Devices to Systems*, 2021, <https://arxiv.org/abs/2012.03192>.
- [16] M. Zhang *et al.*, “GraphR: Reducing Communication for PIM-Based Graph Processing with Efficient Data Partition,” in *HPCA*, 2018.
- [17] L. Song, “GraphR: Accelerating Graph Processing Using ReRAM,” in *HPCA*, 2018.
- [18] Z. Zhuo *et al.*, “GraphQ: Scalable PIM-Based Graph Processing,” in *MICRO*, 2019.
- [19] G. Dai *et al.*, “GraphA: A Processing-in-Memory Architecture for Large-Scale Graph Processing,” in *IEEE TCAD*, 2018.
- [20] G. Li *et al.*, “GraphIA: An In-situ Accelerator for Large-scale Graph Processing,” in *MEMSYS*, 2018.
- [21] S. Rheindt *et al.*, “NEMESIS: Near-Memory Graph Copy Enhanced System-Software,” in *MEMSYS*, 2019.
- [22] L. Belayneh and V. Bertacco, “GraphVine: Exploiting Multicast for Graph Analytics,” in *DATE*, 2020.
- [23] N. Challaipalle *et al.*, “Gauss: Graph Analytics Accelerator Supporting Sparse Rank-Level Parallelism using Crossbar,” in *ISCA*, 2020.
- [24] M. Zhou *et al.*, “Ultra Efficient Accelerator for De Novo Genome Assembly by Near-Memory Computing,” in *ISCA*, 2021.
- [25] X. Xie *et al.*, “SpaceA: Sparse Matrix Vector Multiplication on Processing-in-Memory Accelerator,” in *HPCA*, 2021.
- [26] M. Zhou *et al.*, “HyGraph: Accelerating Graph Processing with Hybrid Memory-Centric Computing,” in *ISCA*, 2021.
- [27] M. Lenjani *et al.*, “Gearbox: A Case for Supporting Accumulation Dispatching and Hybrid Partitioning in PIM-based Accelerators,” in *ISCA*, 2022.
- [28] M. Orenes-Vera *et al.*, “Dalorex: A Data-Local Program Execution and Architecture for Memory-Bound Applications,” in *HPCA*, 2023.
- [29] J. Choquette, “Nvidia Hopper GPU: Scaling Performance,” in *Hot Chips*, 2022.
- [30] F. Devaux, “The True Processing in Memory Accelerator,” in *Hot Chips*, 2019.
- [31] J. Gómez-Luna *et al.*, “Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System,” *IEEE Access*, 2022.
- [32] J. Gomez-Luna *et al.*, “Evaluating Machine Learning Workloads on Memory-Centric Computing Systems,” in *ISPASS*, 2023.
- [33] S. Lee *et al.*, “Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology: Industrial Product,” in *ISCA*, 2021.
- [34] Y.-C. Kwon *et al.*, “25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2 Tbps Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications,” in *ISSCC*, 2021.
- [35] L. Ke *et al.*, “Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDimM,” in *IEEE Micro*, 2021.
- [36] D. Lee *et al.*, “Improving In-Memory Database Operations with Acceleration DIMM (AxDimM),” in *DaMoN*, 2022.
- [37] S. Lee *et al.*, “A 1nm 1.25 V 8Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1Tbops MAC Operation and Various Activation Functions for Deep-Learning Applications,” in *ISSCC*, 2022.
- [38] D. Niu *et al.*, “184QPS/W 64Mb/mm² 3D Logic-to-DRAM Hybrid Bonding with Process-Near-Memory Engine for Recommendation System,” in *ISSCC*, 2022.
- [39] Y. Kwon, “System Architecture and Software Stack for GDDR6-AIM,” in *HCS*, 2022.
- [40] G. Singh *et al.*, “FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications,” *IEEE Micro*, 2021.
- [41] G. Singh *et al.*, “Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric,” *ACM TACD*, 2021.
- [42] S. Hong *et al.*, “PGX-D: A Fast Distributed Graph Processing Engine,” in *SC*, 2015.
- [43] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” in *ICLR*, 2017.
- [44] L. Nai *et al.*, “GraphPIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks,” in *HPCA*, 2017.
- [45] A. Besta *et al.*, “NISA: Semi-centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems,” in *MICRO*, 2021.
- [46] T. J. Ham *et al.*, “Graphicionado: A High-Performance and Energy-Efficient Accelerator for Graph Analytics,” in *MICRO*, 2016.

Accelerating Graph Pattern Mining

- 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.

[[Slides \(pdf\)](#)]

[[Talk Video](#) (22 minutes)]

[[Lightning Talk Video](#) (1.5 minutes)]

[[Full arXiv version](#)]

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

Maciej Besta¹, Raghavendra Kanakagiri², Grzegorz Kwasniewski¹, Rachata Ausavarungnirun³, Jakub Beránek⁴, Konstantinos Kanellopoulos¹, Kacper Janda⁵, Zur Vonarburg-Shmaria¹, Lukas Gianinazzi¹, Ioana Stefan¹, Juan Gómez-Luna¹, Marcin Copik¹, Lukas Kapp-Schwoerer¹, Salvatore Di Girolamo¹, Nils Blach¹, Marek Konieczny⁵, Onur Mutlu¹, Torsten Hoefler¹

¹ETH Zurich, Switzerland
Thailand

²IIT Tirupati, India

³King Mongkut's University of Technology North Bangkok,
⁴Technical University of Ostrava, Czech Republic

⁵AGH-UST, Poland

PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,

"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"

Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Williamsburg, VA, USA, March 2018.

[[Slides \(pptx\) \(pdf\)](#)] [[Lightning Session Slides \(pptx\) \(pdf\)](#)] [[Poster \(pptx\) \(pdf\)](#)]

[[Lightning Talk Video](#) (2 minutes)]

[[Full Talk Video](#) (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹

Saugata Ghose¹

Youngsok Kim²

Rachata Ausavarungnirun¹

Eric Shiu³

Rahul Thakur³

Daehyun Kim^{4,3}

Aki Kuusela³

Allan Knies³

Parthasarathy Ranganathan³

Onur Mutlu^{5,1}

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

SAFARI

Carnegie Mellon

Google



SEOUL
NATIONAL
UNIVERSITY

ETH zürich

Consumer Devices



Consumer devices are everywhere!

**Energy consumption is
a first-class concern in consumer devices**



Popular Consumer Workloads



Chrome

Google's web browser



TensorFlow Mobile

Google's machine learning
framework

VP9



Video Playback

Google's **video codec**

VP9

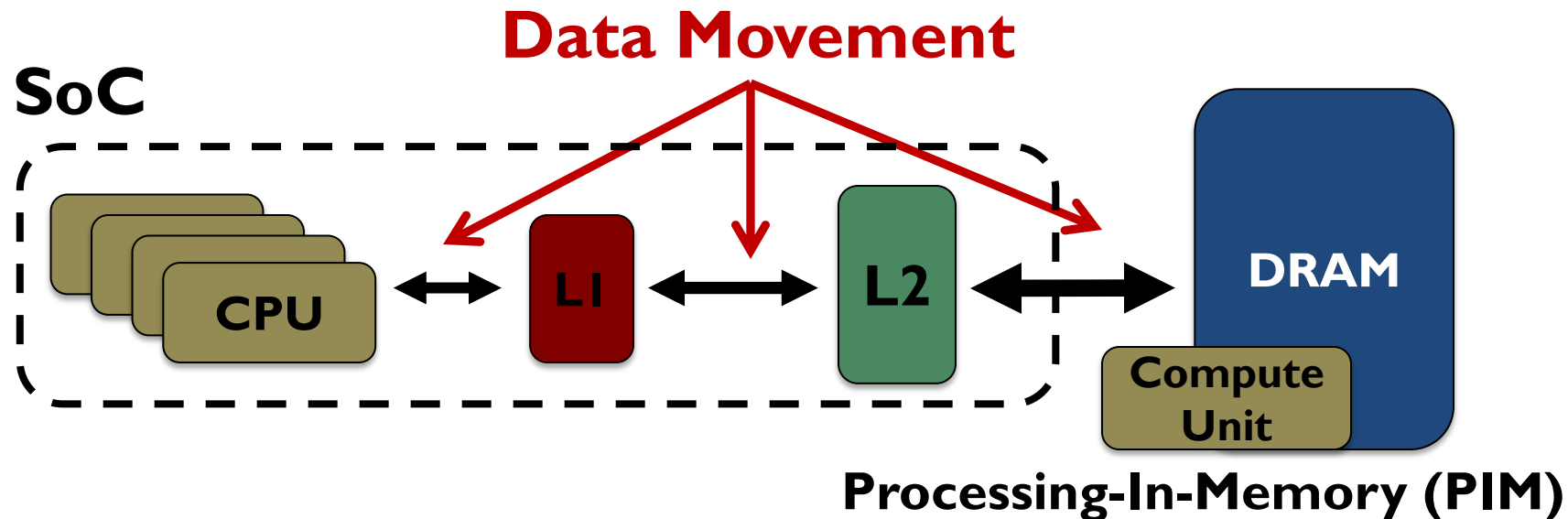


Video Capture

Google's **video codec**

Energy Cost of Data Movement

1st key observation: **62.7%** of the total system energy is spent on **data movement**



Potential solution: move computation **close to data**

Challenge: limited area and energy budget

Using PIM to Reduce Data Movement

2nd key observation: a significant fraction of the **data movement** often comes from **simple functions**

We can design lightweight logic to implement these simple functions in **memory**

Small embedded
low-power core



Small fixed-function
accelerators



Offloading to PIM logic reduces energy and improves performance, on average, by 2.3X and 2.2X

Workload Analysis



Chrome

Google's web browser



TensorFlow Mobile

Google's machine learning
framework



Video Playback

Google's **video codec**

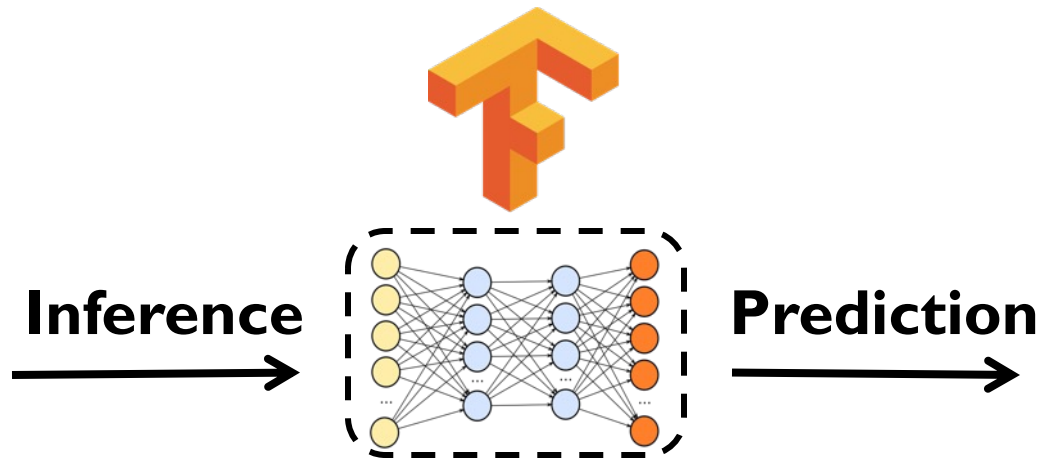


Video Capture

Google's **video codec**

SAFARI

TensorFlow Mobile

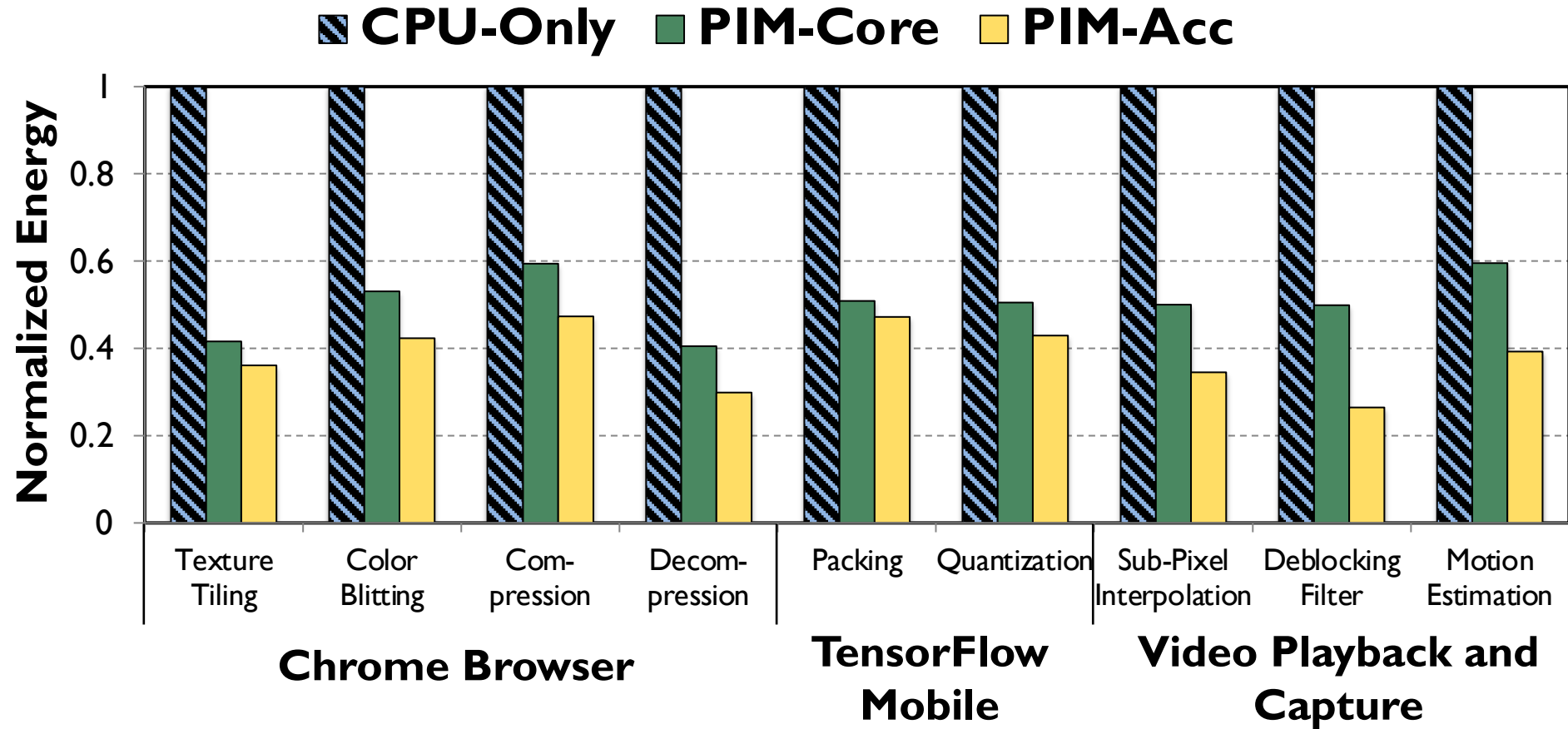


57.3% of the inference energy is spent on data movement



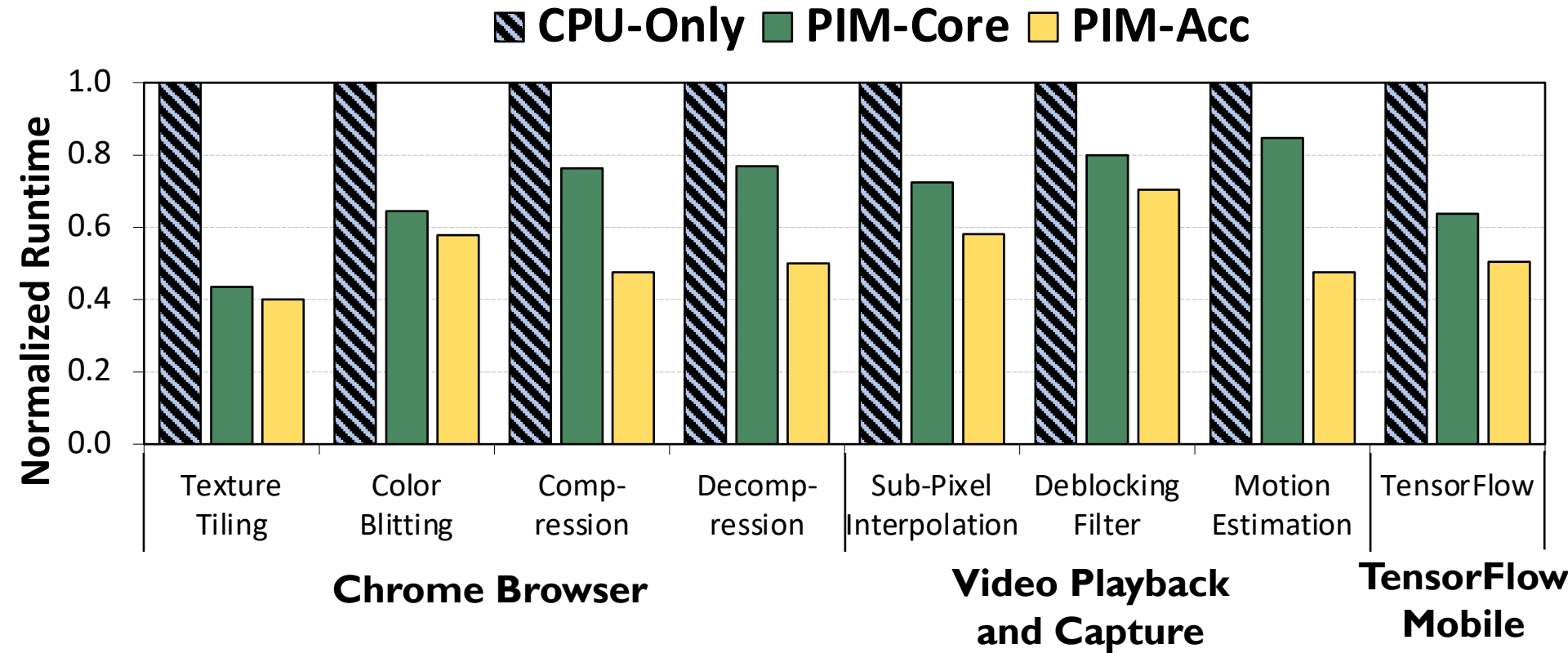
54.4% of the **data movement** energy comes from packing/unpacking and quantization

Normalized Energy



PIM core and PIM accelerator reduce
energy consumption on average by 49.1% and 55.4%

Normalized Runtime



Offloading these kernels to **PIM core** and **PIM accelerator** reduces **program runtime** on average by **44.6%** and **54.2%**

More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,

"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"

Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Williamsburg, VA, USA, March 2018.

[[Slides \(pptx\) \(pdf\)](#)] [[Lightning Session Slides \(pptx\) \(pdf\)](#)] [[Poster \(pptx\) \(pdf\)](#)]

[[Lightning Talk Video](#) (2 minutes)]

[[Full Talk Video](#) (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹

Saugata Ghose¹

Youngsok Kim²

Rachata Ausavarungnirun¹

Eric Shiu³

Rahul Thakur³

Daehyun Kim^{4,3}

Aki Kuusela³

Allan Knies³

Parthasarathy Ranganathan³

Onur Mutlu^{5,1}

Accelerating Neural Network Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Virtual, September 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand^{†◇}

Geraldo F. Oliveira^{*}

Saugata Ghose[‡]

Xiaoyu Ma[§]

Berkin Akin[§]

Eric Shiu[§]

Ravi Narayanaswami[§]

Onur Mutlu^{*†}

[†]*Carnegie Mellon Univ.*

[◇]*Stanford Univ.*

[‡]*Univ. of Illinois Urbana-Champaign*

[§]*Google*

^{*}*ETH Zürich*

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand

Saugata Ghose

Berkin Akin

Ravi Narayanaswami

Geraldo F. Oliveira

Xiaoyu Ma

Eric Shiu

Onur Mutlu

PACT 2021

SAFARI

Carnegie Mellon



UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN



ETH zürich

Executive Summary

Context: We extensively analyze a state-of-the-art edge ML accelerator (Google Edge TPU) using 24 Google edge models

- Wide range of models (CNNs, LSTMs, Transducers, RCNNs)

Problem: The Edge TPU accelerator suffers from **three challenges:**

- It operates **significantly below** its peak throughput
- It operates **significantly below** its theoretical energy efficiency
- It **inefficiently** handles memory accesses

Key Insight: These shortcomings arise from **the monolithic design** of the Edge TPU accelerator

- The Edge TPU accelerator design does not account for **layer heterogeneity**

Key Mechanism: A new framework called **Mensa**

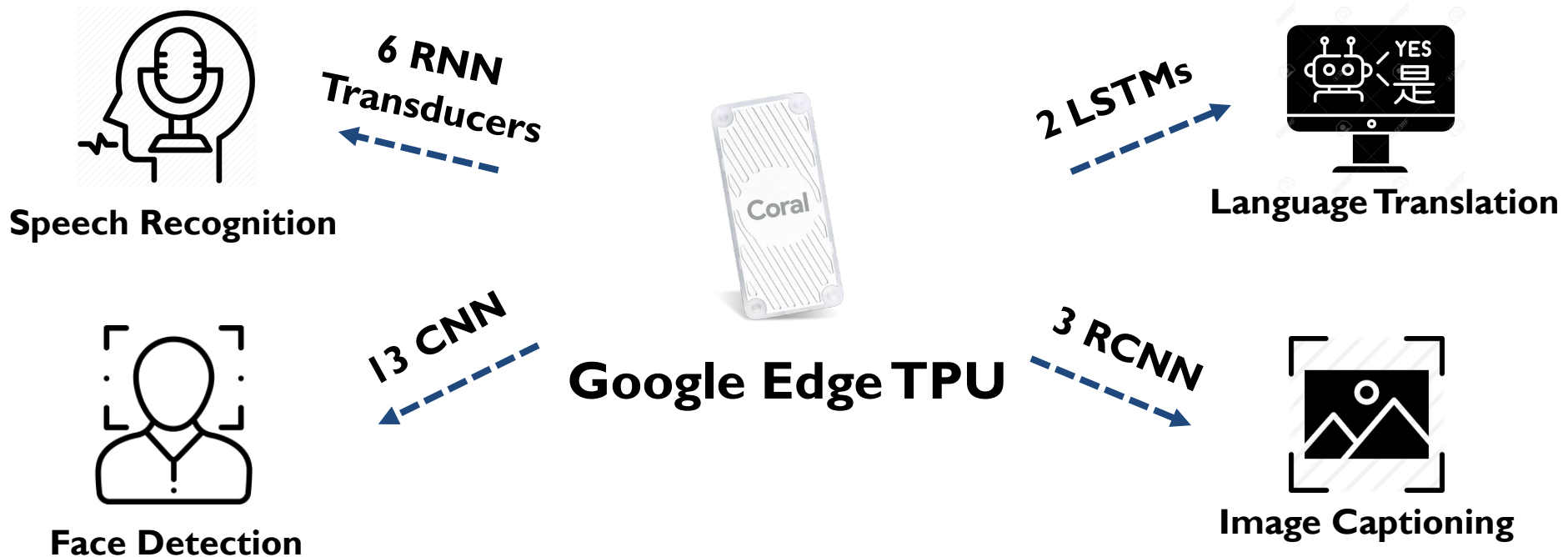
- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Key Results: We design a version of Mensa for Google edge ML models

- Mensa improves performance and energy by **3.0X** and **3.1X**
- Mensa reduces cost and improves area efficiency

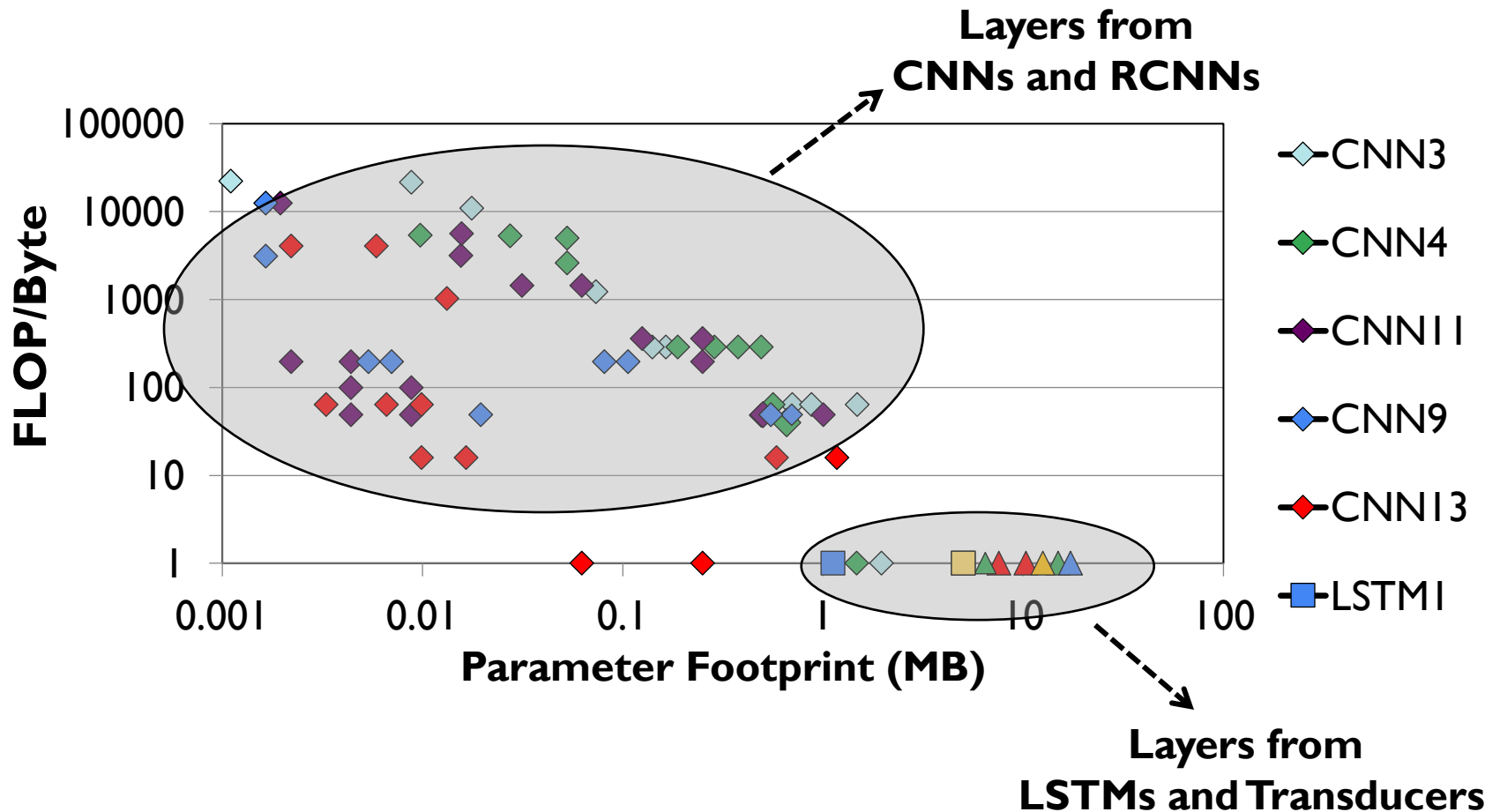
Google Edge Neural Network Models

We analyze inference execution using 24 edge NN models



Diversity Across the Models

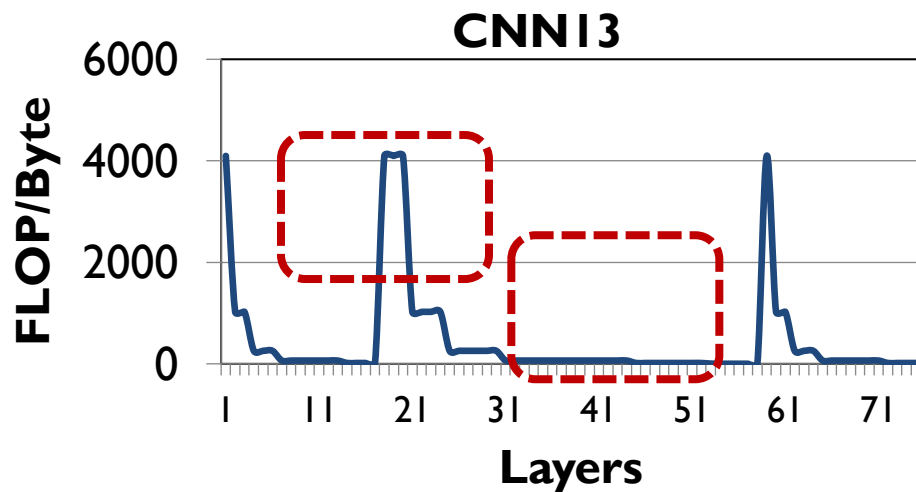
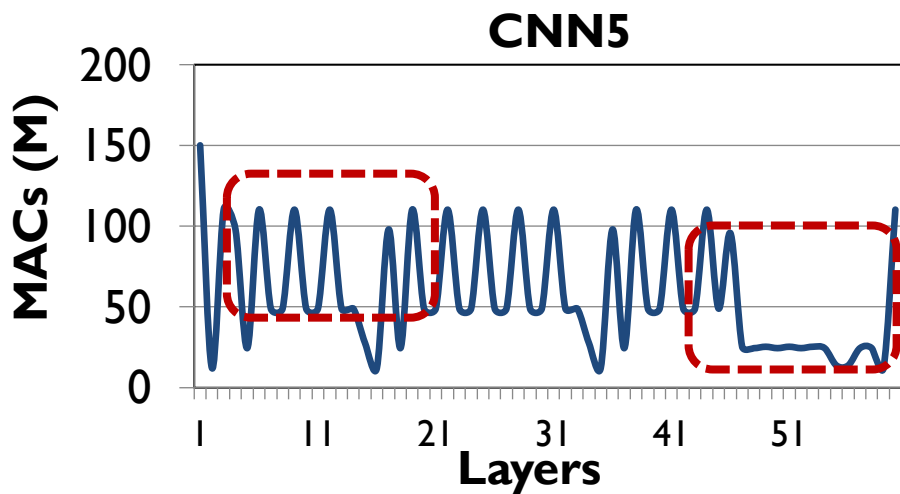
Insight I: there is **significant variation** in terms of layer characteristics **across the models**



Diversity Within the Models

Insight 2: even **within** each model, layers exhibit **significant variation** in terms of layer characteristics

For example, our analysis of edge **CNN** models shows:



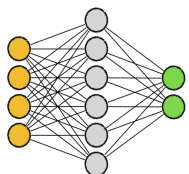
Variation in **MAC intensity**: up to **200x** across layers

Variation in **FLOP/Byte**: up to **244x** across layers

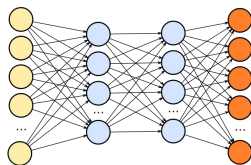
Mensa High-Level Overview

Edge TPU Accelerator

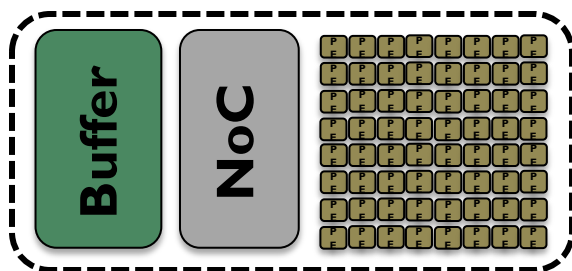
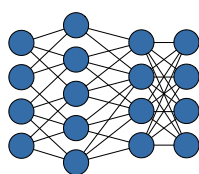
Model A



Model B



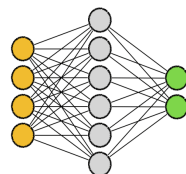
Model C



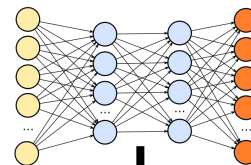
Monolithic Accelerator

Mensa

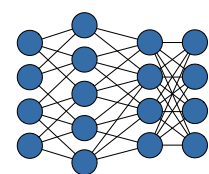
Model A



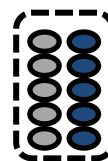
Model B



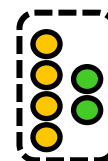
Model C



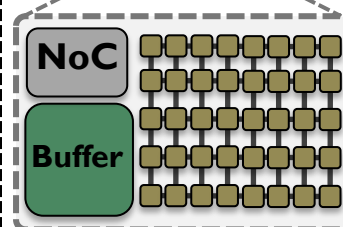
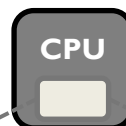
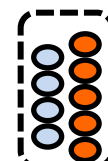
Family 1



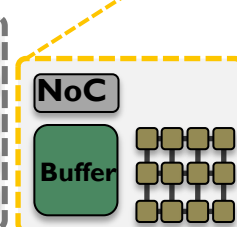
Family 2



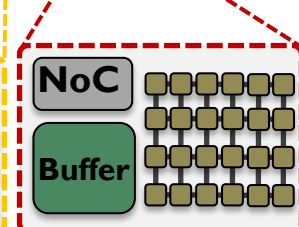
Family 3



Acc. 1



Acc. 2

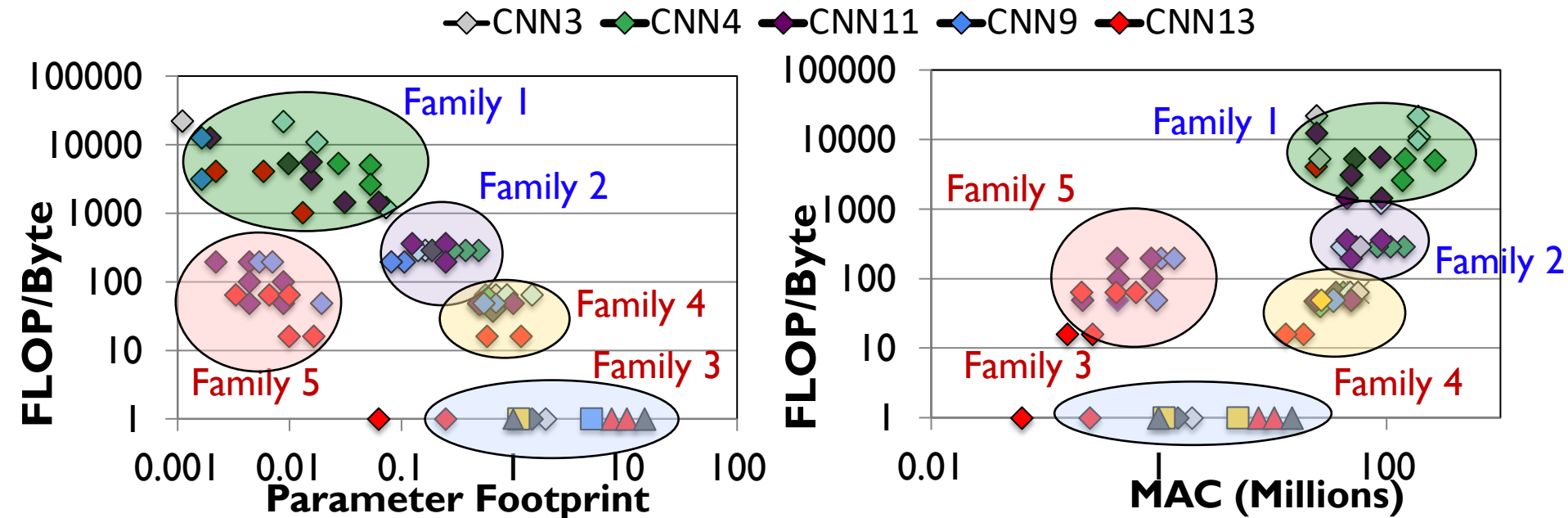


Acc. 3

Heterogeneous Accelerators

Identifying Layer Families

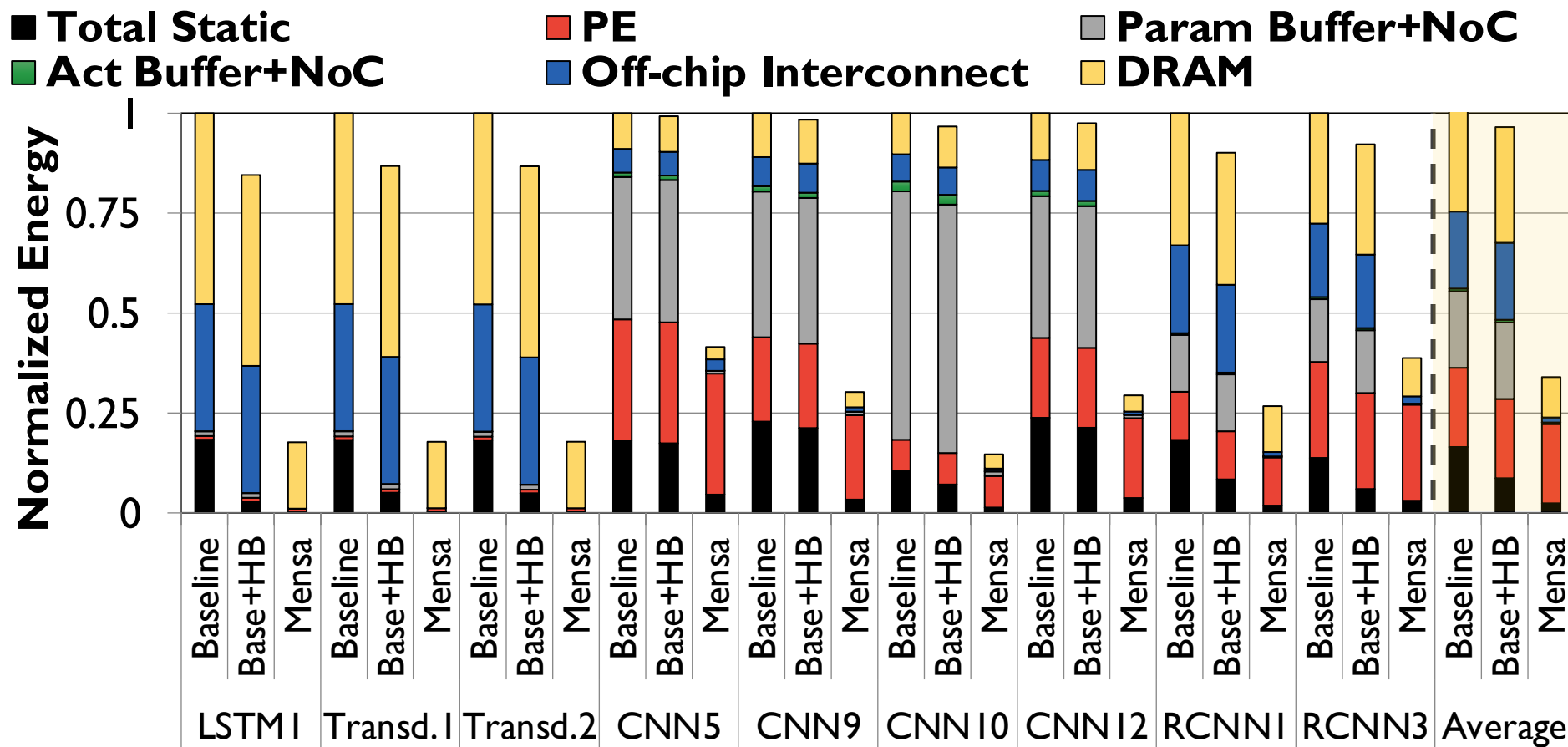
Key observation: the majority of layers group into a small number of layer families



Families 1 & 2: low parameter footprint, high data reuse and **MAC intensity**
→ compute-centric layers

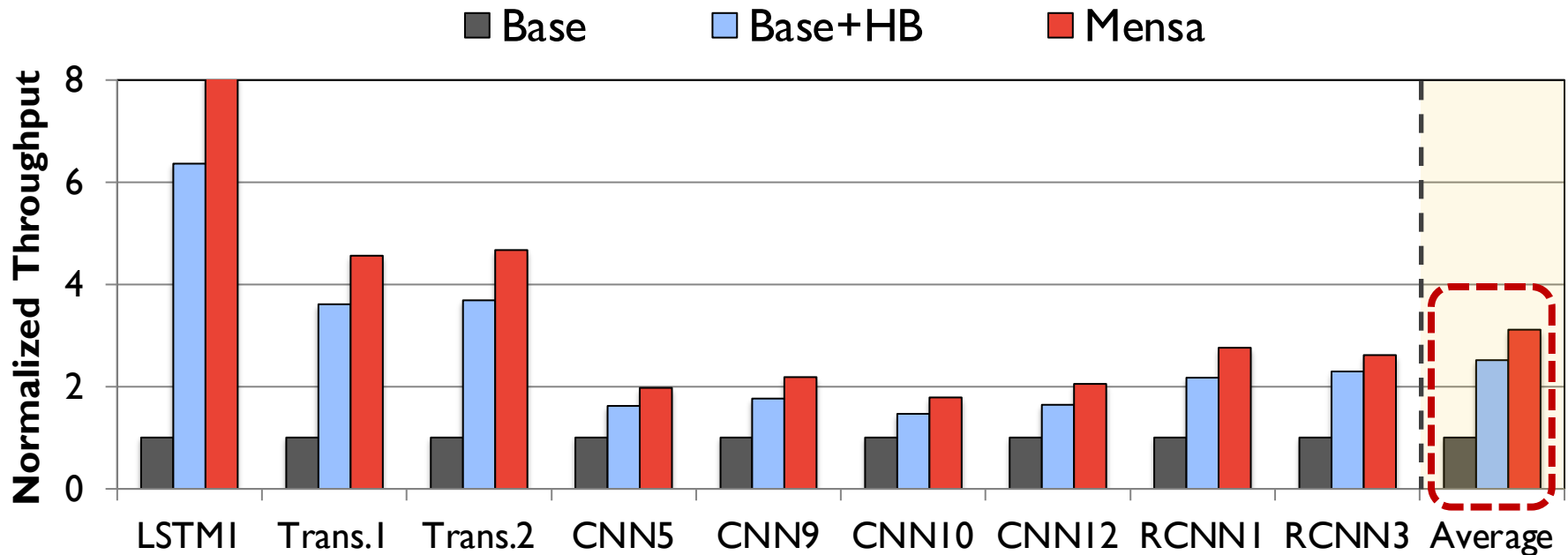
Families 3, 4 & 5: high parameter footprint, low data reuse and **MAC intensity**
→ data-centric layers

Mensa: Energy Reduction



Mensa-G reduces energy consumption by 3.0X
compared to the baseline Edge TPU

Mensa: Throughput Improvement



Mensa-G improves inference throughput by 3.1X
compared to the baseline Edge TPU

Mensa: Highly-Efficient ML Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Virtual, September 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand^{†◇}

Geraldo F. Oliveira^{*}

Saugata Ghose[‡]

Xiaoyu Ma[§]

Berkin Akin[§]

Eric Shiu[§]

Ravi Narayanaswami[§]

Onur Mutlu^{*†}

[†]*Carnegie Mellon Univ.*

[◇]*Stanford Univ.*

[‡]*Univ. of Illinois Urbana-Champaign*

[§]*Google*

^{*}*ETH Zürich*

In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Virtual, February-March 2022.
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Video](#) (90 seconds)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi¹ Jisung Park¹ Harun Mustafa¹ Jeremie Kim¹ Ataberk Olgun¹
Arvid Gollwitzer¹ Damla Senol Cali² Can Firtina¹ Haiyu Mao¹ Nour Almadhoun Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto

Genome Sequence Analysis

Data Movement from Storage



Alignment

Storage
System

Main
Memory

Cache

Computation
Unit
(CPU or
Accelerator)

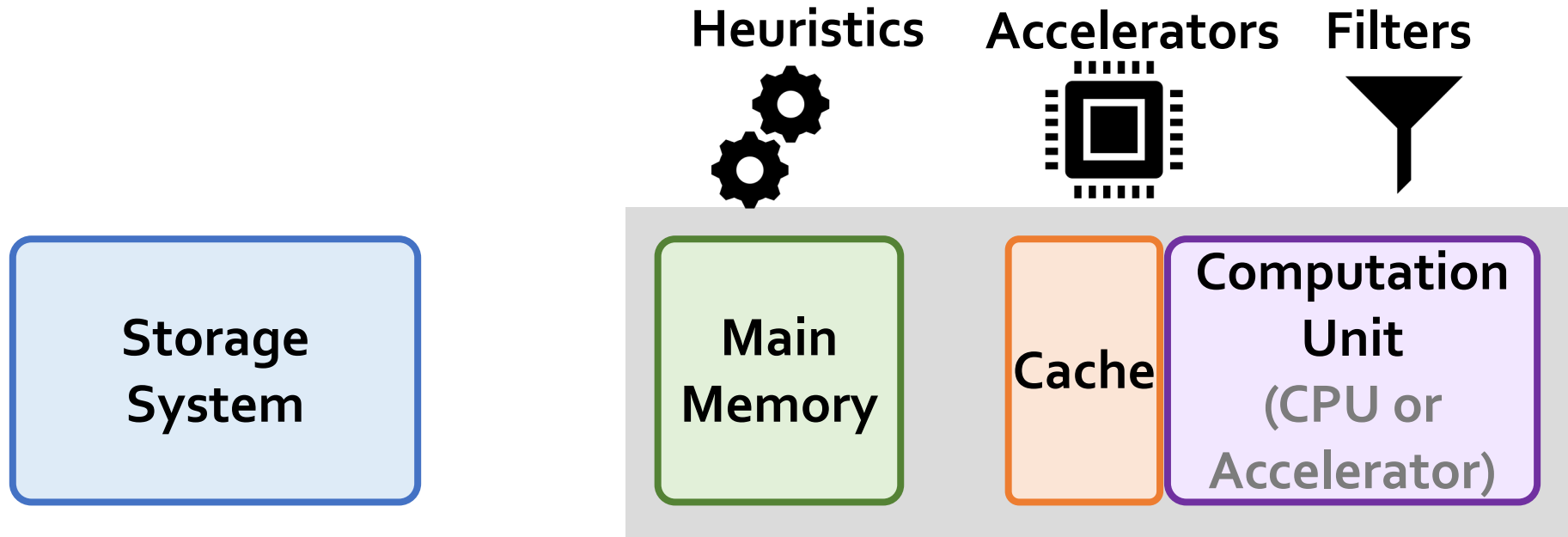


Computation overhead



Data movement overhead

Compute-Centric Accelerators



Computation overhead

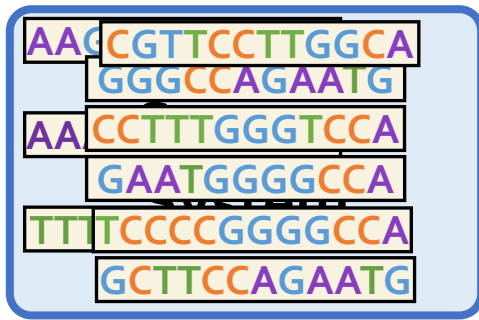


Data movement overhead

Key Idea: In-Storage Filtering



*Filter reads that do **not** require alignment inside the storage system*



Filtered Reads

**Main
Memory**

Cache

**Computation
Unit
(CPU or
Accelerator)**

Exactly-matching reads

Do not need expensive approximate string matching during alignment

Non-matching reads

Do not have potential matching locations and can skip alignment

GenStore



*Filter reads that do **not** require alignment
inside the storage system*

GenStore-Enabled
Storage
System

Main
Memory

Cache

Computation
Unit
(CPU or
Accelerator)



Computation overhead



Data movement overhead

GenStore provides significant speedup (1.4x - 33.6x) and
energy reduction (3.9x - 29.2x) at low cost

In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Virtual, February-March 2022.
[[Lightning Talk Slides \(pptx\)](#)] ([pdf](#))
[[Lightning Talk Video](#) (90 seconds)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi¹ Jisung Park¹ Harun Mustafa¹ Jeremie Kim¹ Ataberk Olgun¹
Arvid Gollwitzer¹ Damla Senol Cali² Can Firtina¹ Haiyu Mao¹ Nour Almadhoun Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto

Also See: In-Storage Metagenomics [ISCA 2024]

- Nika Mansouri Ghiasi, Mohammad Sadrosadati, Harun Mustafa, Arvid Gollwitzer, Can Firtina, Julien Eudine, Haiyu Mao, Joel Lindegger, Meryem Banu Cavlak, Mohammed Alser, Jisung Park, and Onur Mutlu,

"MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing"

Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, July 2024.

[[Slides \(pptx\)](#)] [[pdf](#)]

[[arXiv version](#)]

MegIS: High-Performance, Energy-Efficient, and Low-Cost Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi¹ Mohammad Sadrosadati¹ Harun Mustafa¹ Arvid Gollwitzer¹
Can Firtina¹ Julien Eudine¹ Haiyu Mao¹ Joël Lindegger¹ Meryem Banu Cavlak¹
Mohammed Alser¹ Jisung Park² Onur Mutlu¹
¹ETH Zürich ²POSTECH

We Covered Until Here
in Lecture 3

Memory Systems and Memory-Centric Computing

Lecture 3: Memory-Centric Computing II

Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

17 July 2024

HiPEAC ACACES Summer School 2024

SAFARI

ETH zürich

Backup Slides

(To Be Covered in Lecture 4)

DAMOV Analysis Methodology & Workloads

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland

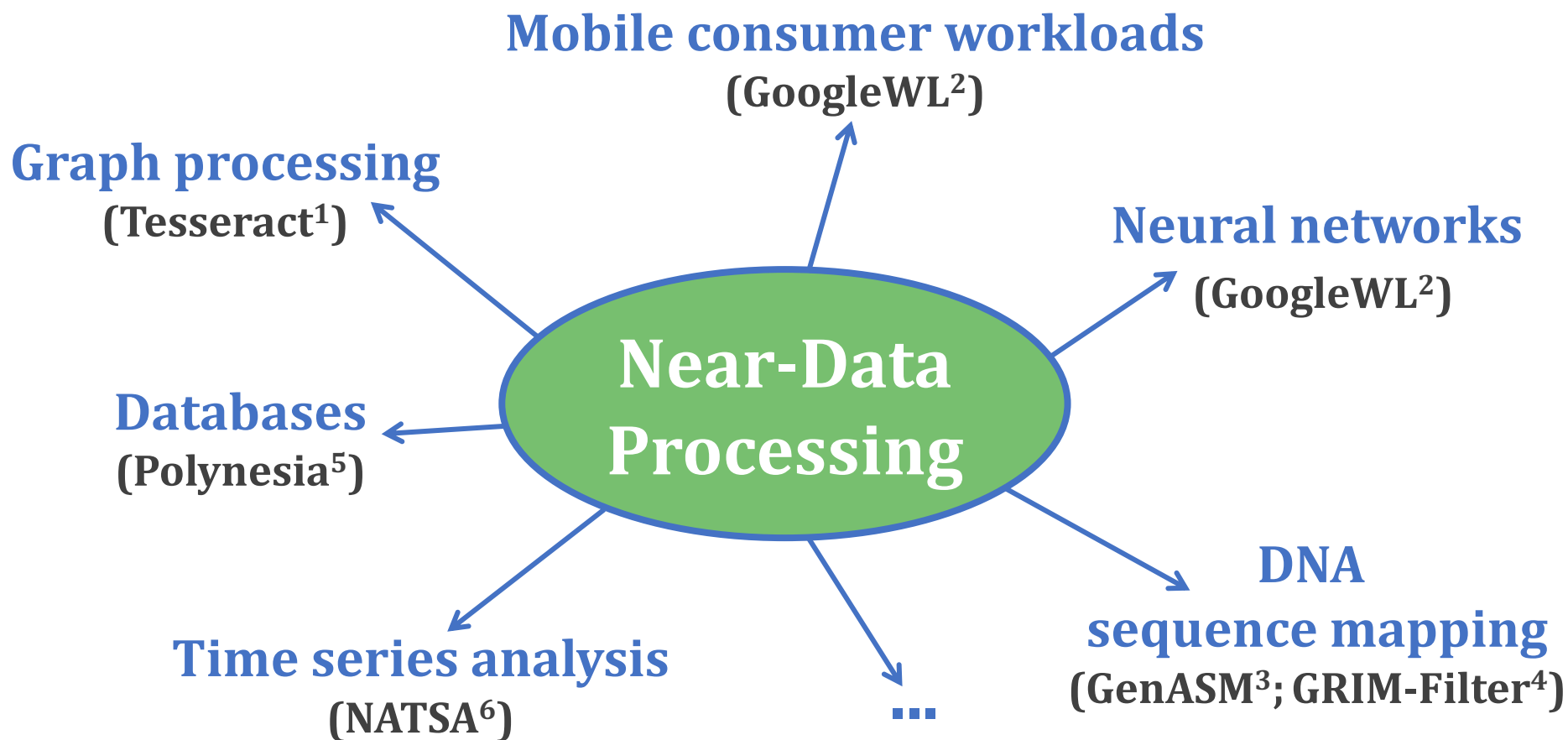
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order obstacle against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at <https://github.com/CMU-SAFARI/DAMOV>.

When to Employ Near-Data Processing?



[1] Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," ISCA, 2015

[2] Boroumand+, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS, 2018

[3] Cali+, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," MICRO, 2020

[4] Kim+, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies," BMC Genomics, 2018

[5] Boroumand+, "Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design," arXiv:2103.00798 [cs.AR], 2021

[6] Fernandez+, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis," ICCD, 2020

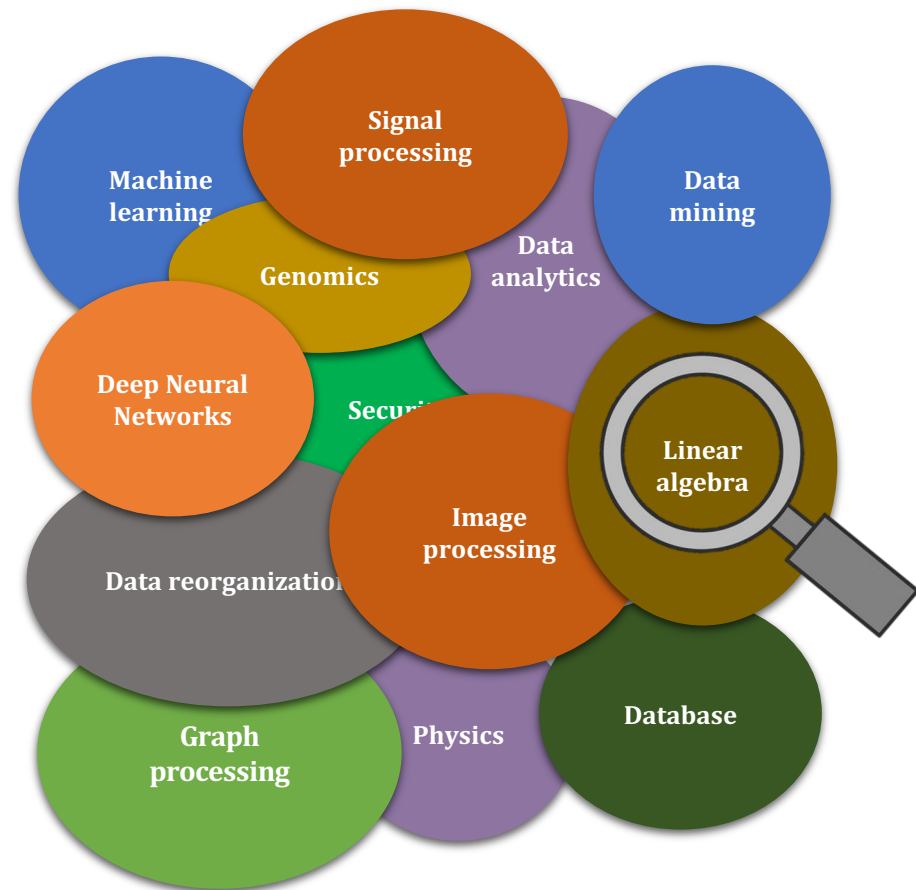
Step 1: Application Profiling

- We analyze 345 applications from distinct domains:

- Graph Processing
- Deep Neural Networks
- Physics
- High-Performance Computing
- Genomics
- Machine Learning
- Databases
- Data Reorganization
- Image Processing
- Map-Reduce
- Benchmarking
- Linear Algebra

...

SAFARI



Step 3: Memory Bottleneck Analysis

**Six classes of
data movement bottlenecks:**

each class \leftrightarrow data movement
mitigation mechanism

Memory Bottleneck Class

1a: *DRAM
Bandwidth*

1b: *DRAM Latency*

1c: *L1/L2
Cache Capacity*

2a: *L3 Cache
Contention*

2b: *L1 Cache
Capacity*

2c: *Compute-Bound*

DAMOV is Open Source

- We open-source our **benchmark suite** and our **toolchain**

CMU-SAFARI / DAMOV

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code

About



DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing. Described by Oliveira et al. (preliminary version at <https://arxiv.org/pdf/2105.03725.pdf>)

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



omutlu Update README.md

ce1b4ea 17 days ago 5 commits

simulator	Cleaning	19 days ago
README.md	Update README.md	17 days ago
get_workloads.sh	DAMOV -- first commit	19 days ago

README.md

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including [BWA](#), [Chai](#), [Darknet](#), [GASE](#), [Hardware Effects](#), [Hashjoin](#), [HPCC](#), [HPCG](#), [Ligra](#), [PARSEC](#), [Parboil](#), [PolyBench](#), [Phoenix](#), [Rodinia](#), [SPLASH-2](#), [STREAM](#).

DAMOV-SIM

DAMOV
Benchmarks

SAFARI

DAMOV is Open Source

- We open-source our [benchmark suite](#) and our [toolchain](#)

CMU-SAFARI / DAMOV

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code

About

DAMOV is a benchmark suite and a

Get DAMOV at:

<https://github.com/CMU-SAFARI/DAMOV>

README.md

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including [BWA](#), [Chai](#), [Darknet](#), [GASE](#), [Hardware Effects](#), [Hashjoin](#), [HPCC](#), [HPCG](#), [Ligra](#), [PARSEC](#), [Parboil](#), [PolyBench](#), [Phoenix](#), [Rodinia](#), [SPLASH-2](#), [STREAM](#).

Readme

Releases

No releases published
[Create a new release](#)

Packages

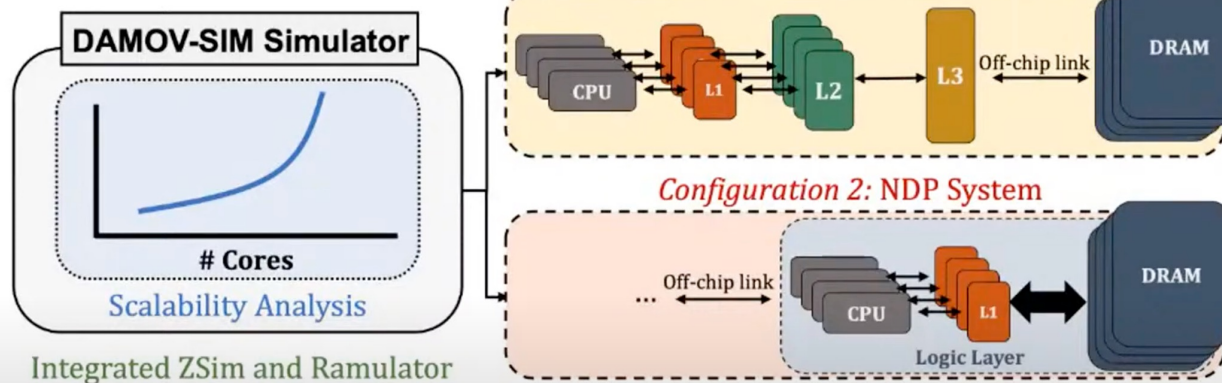
No packages published
[Publish your first package](#)

Languages

More on DAMOV Analysis Methodology & Workloads

Step 3: Memory Bottleneck Classification (2/2)

- **Goal:** identify the specific sources of data movement bottlenecks



- **Scalability Analysis:**
 - 1, 4, 16, 64, and 256 out-of-order/in-order host and NDP CPU cores
 - 3D-stacked memory as main memory

SAFARI DAMOV-SIM: <https://github.com/CMU-SAFARI/DAMOV> 30

SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks

352 views • Streamed live on Jul 22, 2021

18 0 SHARE SAVE ...



Onur Mutlu Lectures
17.7K subscribers

ANALYTICS

EDIT VIDEO

https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9&index=3

More on DAMOV Methods & Benchmarks

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
"DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
IEEE Access, 8 September 2021.
Preprint in arXiv, 8 May 2021.
[[arXiv preprint](#)]
[[IEEE Access version](#)]
[[DAMOV Suite and Simulator Source Code](#)]
[[SAFARI Live Seminar Video](#) (2 hrs 40 mins)]
[[Short Talk Video](#) (21 minutes)]

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland

MOHAMMAD SADROSADATI, ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

Ramulator 2.0 for PIM Systems

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
"Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
*Preprint on **arxiv**, August 2023.*
[[arXiv version](#)]
[[Ramulator 2.0 Source Code](#)]

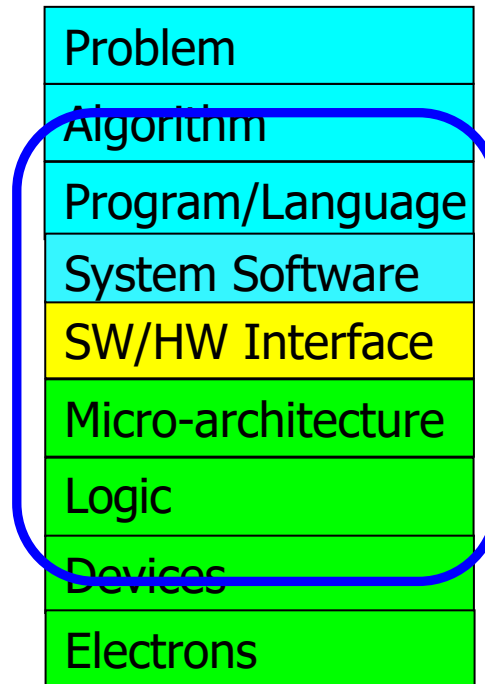
Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

<https://arxiv.org/pdf/2308.11030.pdf>

We Need to Revisit the Entire Stack

- With a **memory-centric mindset**



We can get there step by step

PIM Review and Open Problems

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

^d*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

"A Modern Primer on Processing in Memory"

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

PIM Review and Open Problems (II)

A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose[†] Amirali Boroumand[†] Jeremie S. Kim^{†§} Juan Gómez-Luna[§] Onur Mutlu^{§†}

[†]*Carnegie Mellon University*

[§]*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

"Processing-in-Memory: A Workload-Driven Perspective"

Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.

[Preliminary arXiv version]

Processing in Memory: Adoption Challenges

1. Processing **using** Memory
2. Processing **near** Memory

How to Enable Adoption of Processing in Memory

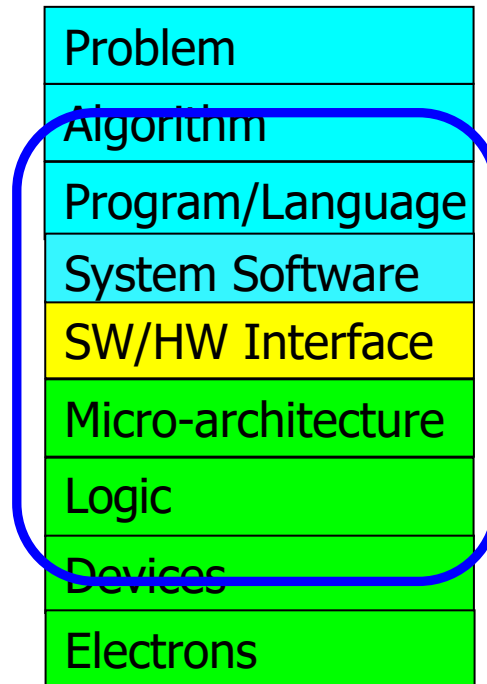
Potential Barriers to Adoption of PIM

1. **Applications & software** for PIM
2. Ease of **programming** (interfaces and compiler/HW support)
3. **System** and **security** support: coherence, synchronization, virtual memory, isolation, communication interfaces, ...
4. **Runtime** and **compilation** systems for adaptive scheduling, data mapping, access/sharing control, ...
5. **Infrastructures** to assess benefits and feasibility

All can be solved with change of mindset

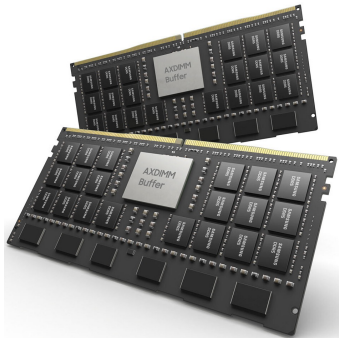
We Need to Revisit the Entire Stack

- With a **memory-centric mindset**

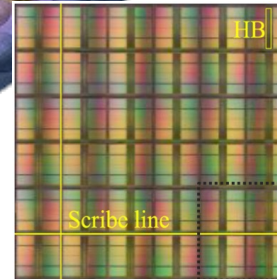
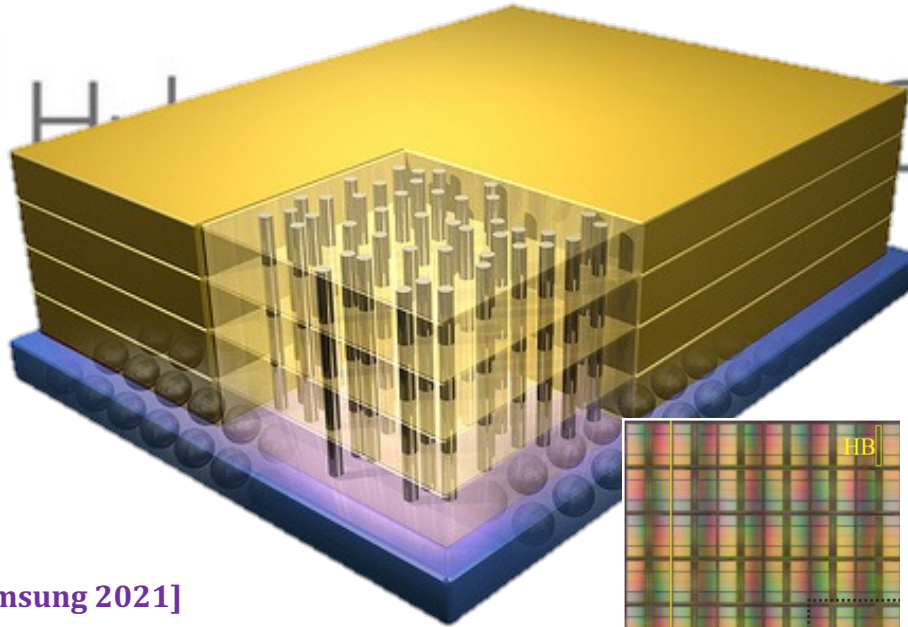


We can get there step by step

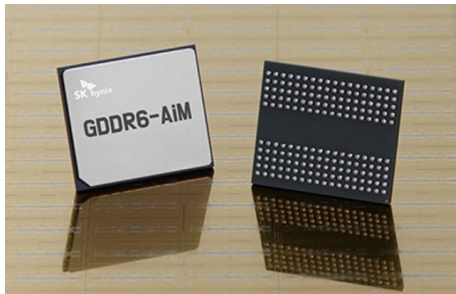
Processing-in-Memory Landscape Today



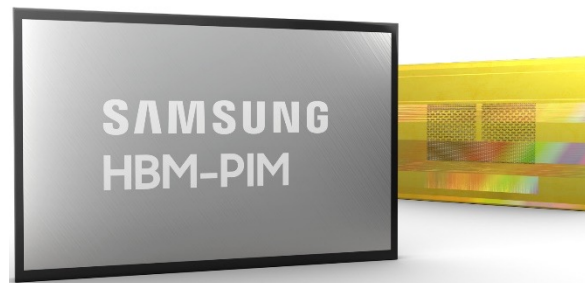
[Samsung 2021]



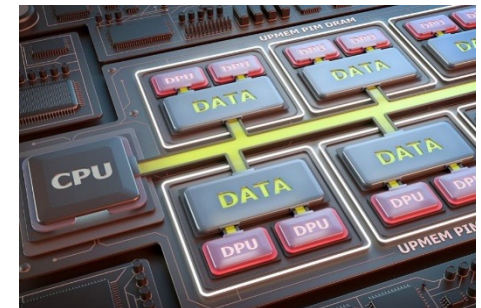
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]



[UPiM 2019]

Adoption: How to Keep It Simple?

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015.
[[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoungh Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

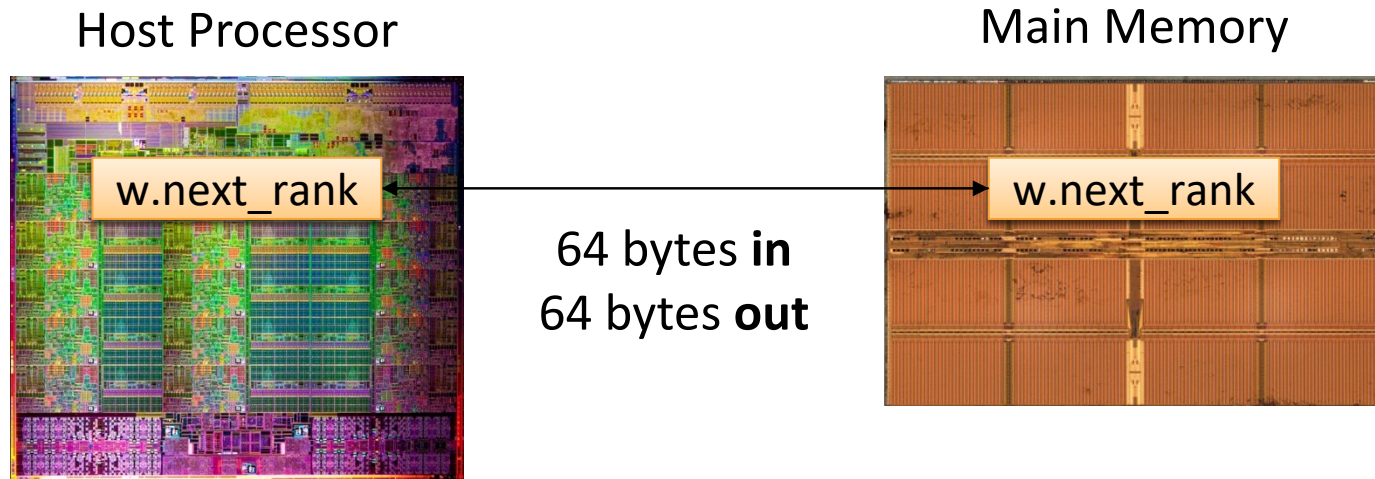
[†]Carnegie Mellon University

PEI: PIM-Enabled Instructions (Ideas)

- **Goal:** Develop mechanisms to get the most out of near-data processing with **minimal cost, minimal changes to the system, no changes to the programming model**
- **Key Idea 1:** Expose each PIM operation as a **cache-coherent, virtually-addressed host processor instruction** (called PEI) that operates on **only a single cache block**
 - e.g., `__pim_add(&w.next_rank, value) → pim.add r1, (r2)`
 - No changes sequential execution/programming model
 - No changes to virtual memory
 - Minimal changes to cache coherence
 - No need for data mapping: Each PEI restricted to a single memory module
- **Key Idea 2:** **Dynamically decide where to execute a PEI** (i.e., the host processor or PIM accelerator) based on simple locality characteristics and simple hardware predictors
 - Execute each operation at the location that provides the best performance

Simple PIM Operations as ISA Extensions (II)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {  
        w.next_rank += value;  
    }  
}
```



Conventional Architecture

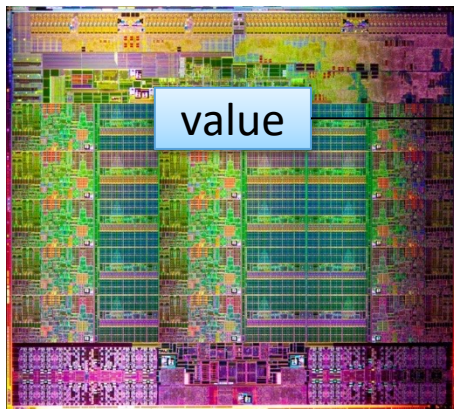
Simple PIM Operations as ISA Extensions (III)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {  
        __pim_add(&w.next_rank, value);  
    }  
}
```

pim.add r1, (r2)

__pim_add(&w.next_rank, value);

Host Processor



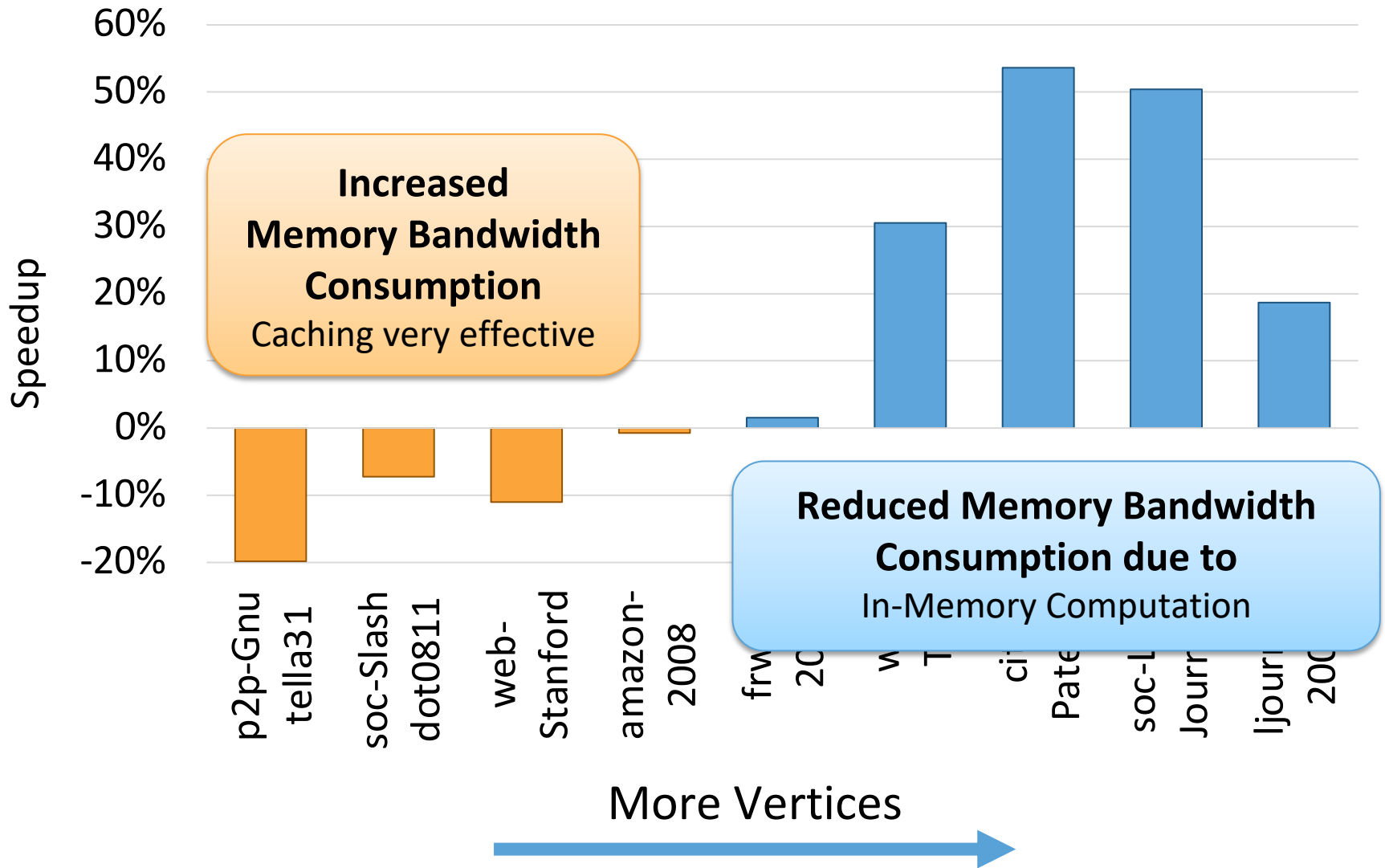
Main Memory



8 bytes in
0 bytes out

In-Memory Addition

Always Executing in Memory? Not A Good Idea



PEI: PIM-Enabled Instructions (Example)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {
```

pim.add r1, (r2)

```
        __pim_add(&w.next_rank, value);  
    }
```

```
pfence();  
}
```

pfence

Table 1: Summary of Supported PIM Operations

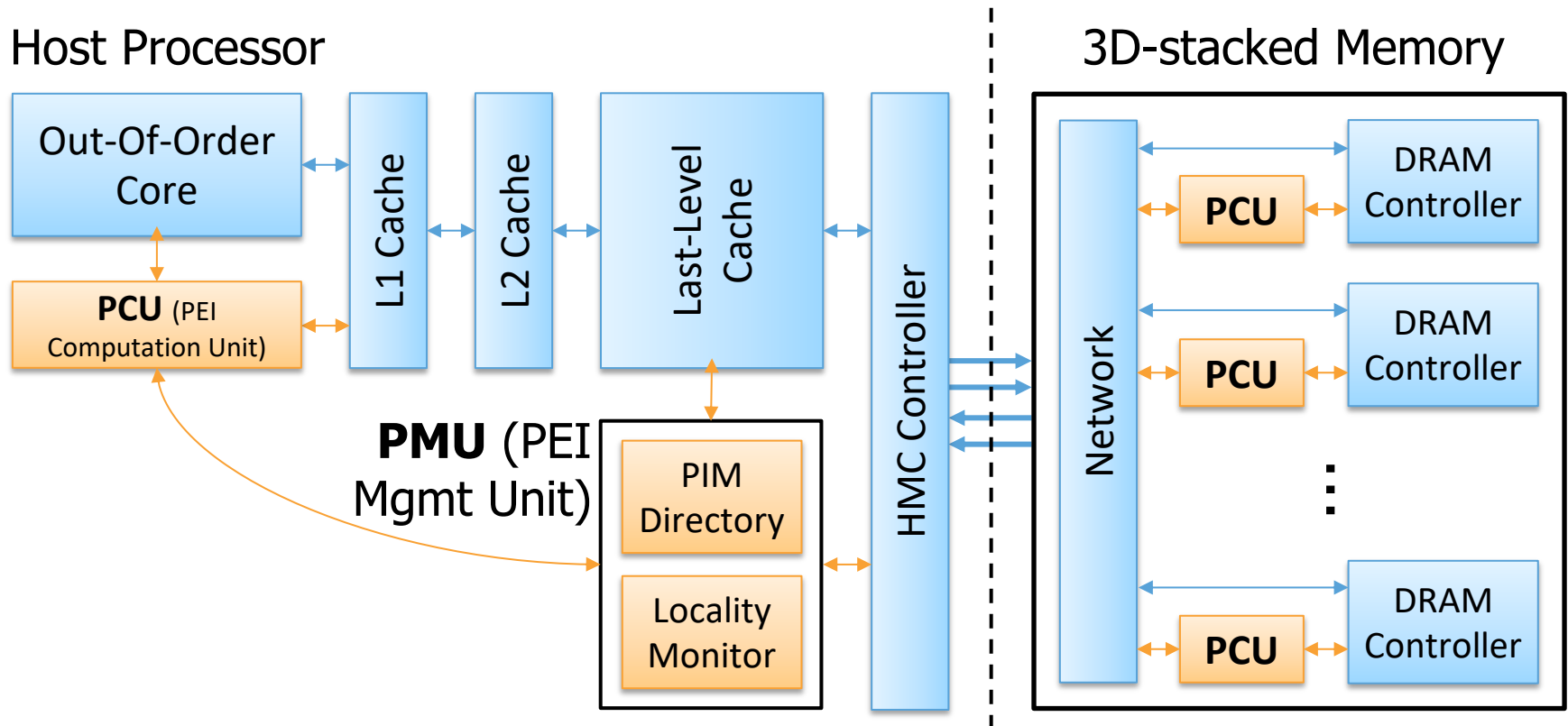
Operation	R	W	Input	Output	Applications
8-byte integer increment	O	O	0 bytes	0 bytes	AT
8-byte integer min	O	O	8 bytes	0 bytes	BFS, SP, WCC
Floating-point add	O	O	8 bytes	0 bytes	PR
Hash table probing	O	X	8 bytes	9 bytes	HJ
Histogram bin index	O	X	1 byte	16 bytes	HG, RP
Euclidean distance	O	X	64 bytes	4 bytes	SC
Dot product	O	X	32 bytes	8 bytes	SVM

- Executed either in memory or in the processor: dynamic decision
 - ❑ Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- *Not* atomic with normal instructions (use *pfence* for ordering)

PIM-Enabled Instructions

- Key to practicality: **single-cache-block restriction**
 - **Each PEI can access *at most one last-level cache block***
 - Similar restrictions exist in atomic instructions
- Benefits
 - **Localization**: each PEI is bounded to one memory module
 - **Interoperability**: easier support for cache coherence and virtual memory
 - **Simplified locality monitoring**: data locality of PEIs can be identified simply by the cache control logic

Example (Abstract) PEI uArchitecture



Example PEI uArchitecture

PEI: Initial Evaluation Results

- Initial evaluations with **10 emerging data-intensive workloads**
 - ❑ Large-scale graph processing
 - ❑ In-memory data analytics
 - ❑ Machine learning and data mining
 - ❑ Three input sets (small, medium, large) for each workload to analyze the impact of data locality
- Pin-based cycle-level x86-64 simulation
- **Performance Improvement and Energy Reduction:**
 - 47% average speedup with large input data sets
 - 32% speedup with small input data sets
 - 25% avg. energy reduction in a single node with large input data sets

Table 2: Baseline Simulation Configuration

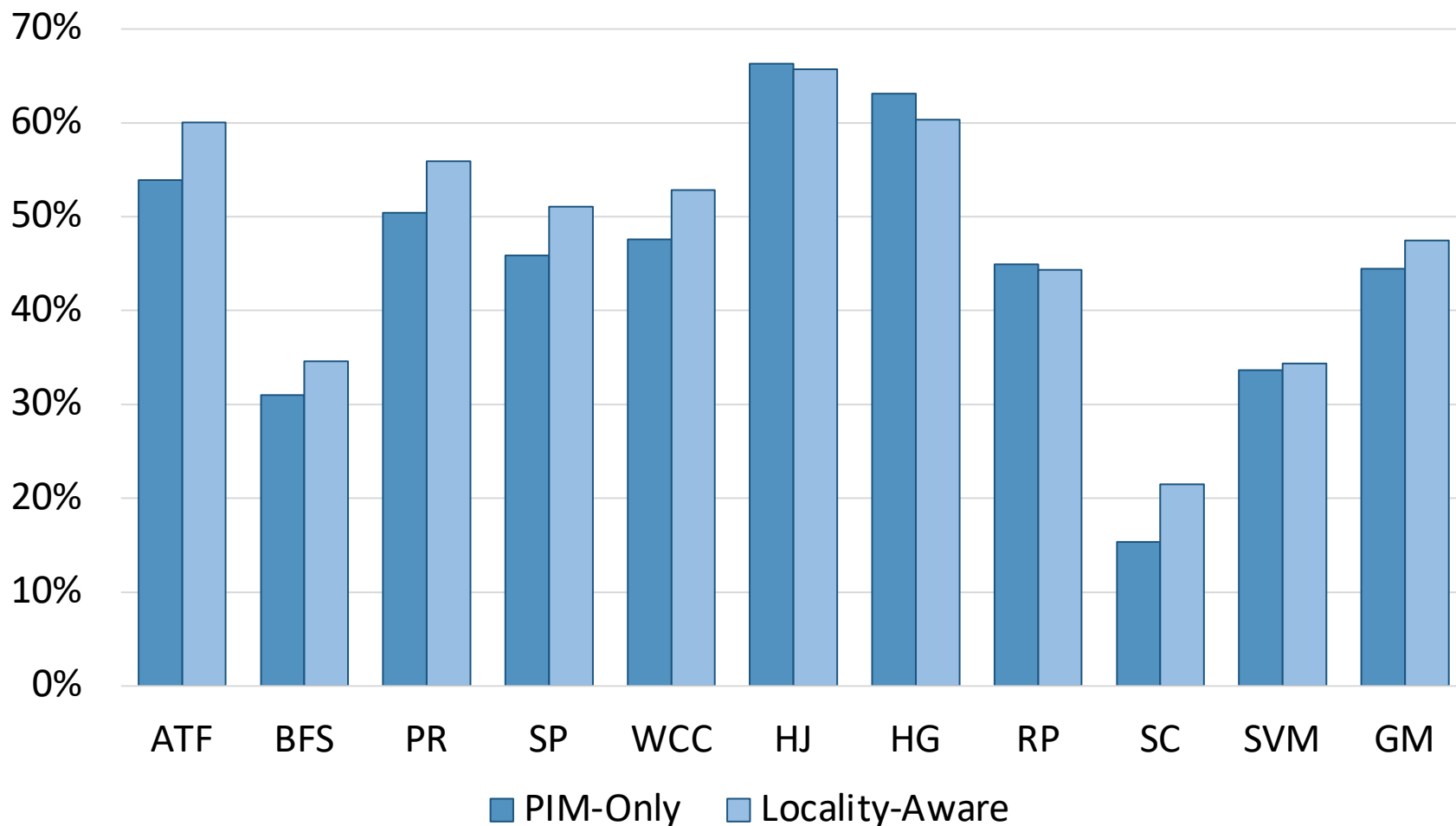
Component	Configuration
Core	16 out-of-order cores, 4 GHz, 4-issue
L1 I/D-Cache	Private, 32 KB, 4/8-way, 64 B blocks, 16 MSHRs
L2 Cache	Private, 256 KB, 8-way, 64 B blocks, 16 MSHRs
L3 Cache	Shared, 16 MB, 16-way, 64 B blocks, 64 MSHRs
On-Chip Network	Crossbar, 2 GHz, 144-bit links
Main Memory	32 GB, 8 HMCs, daisy-chain (80 GB/s full-duplex)
HMC	4 GB, 16 vaults, 256 DRAM banks [20]
– DRAM	FR-FCFS, tCL = tRCD = tRP = 13.75 ns [27]
– Vertical Links	64 TSVs per vault with 2 Gb/s signaling rate [23]

Evaluated Data-Intensive Applications

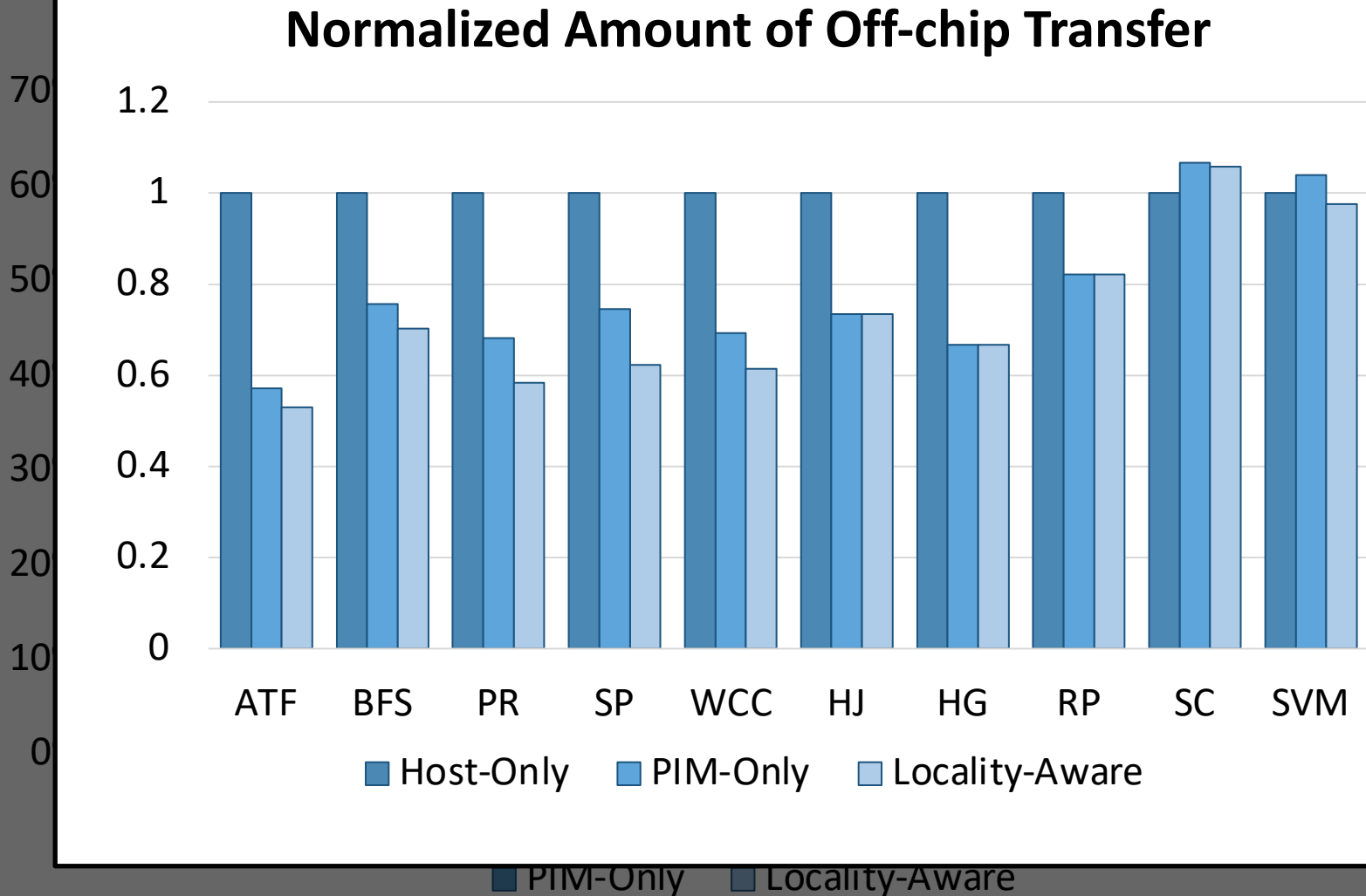
- Ten emerging data-intensive workloads
 - Large-scale graph processing
 - Average teenage follower, BFS, PageRank, single-source shortest path, weakly connected components
 - In-memory data analytics
 - Hash join, histogram, radix partitioning
 - Machine learning and data mining
 - Streamcluster, SVM-RFE
- Three input sets (small, medium, large) for each workload to show the impact of data locality

PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: Host-Only)

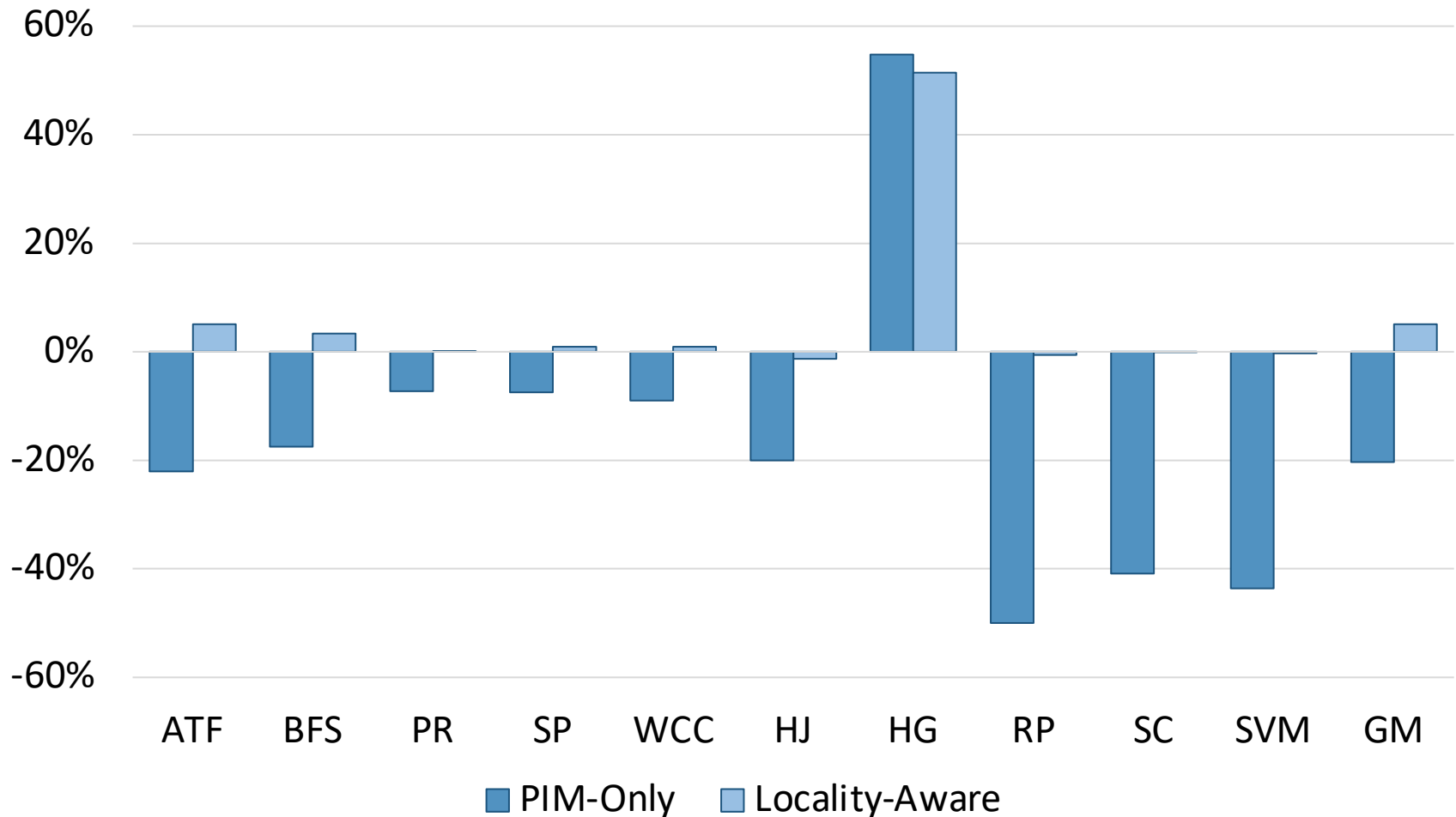


PEI Performance: Large Data Sets

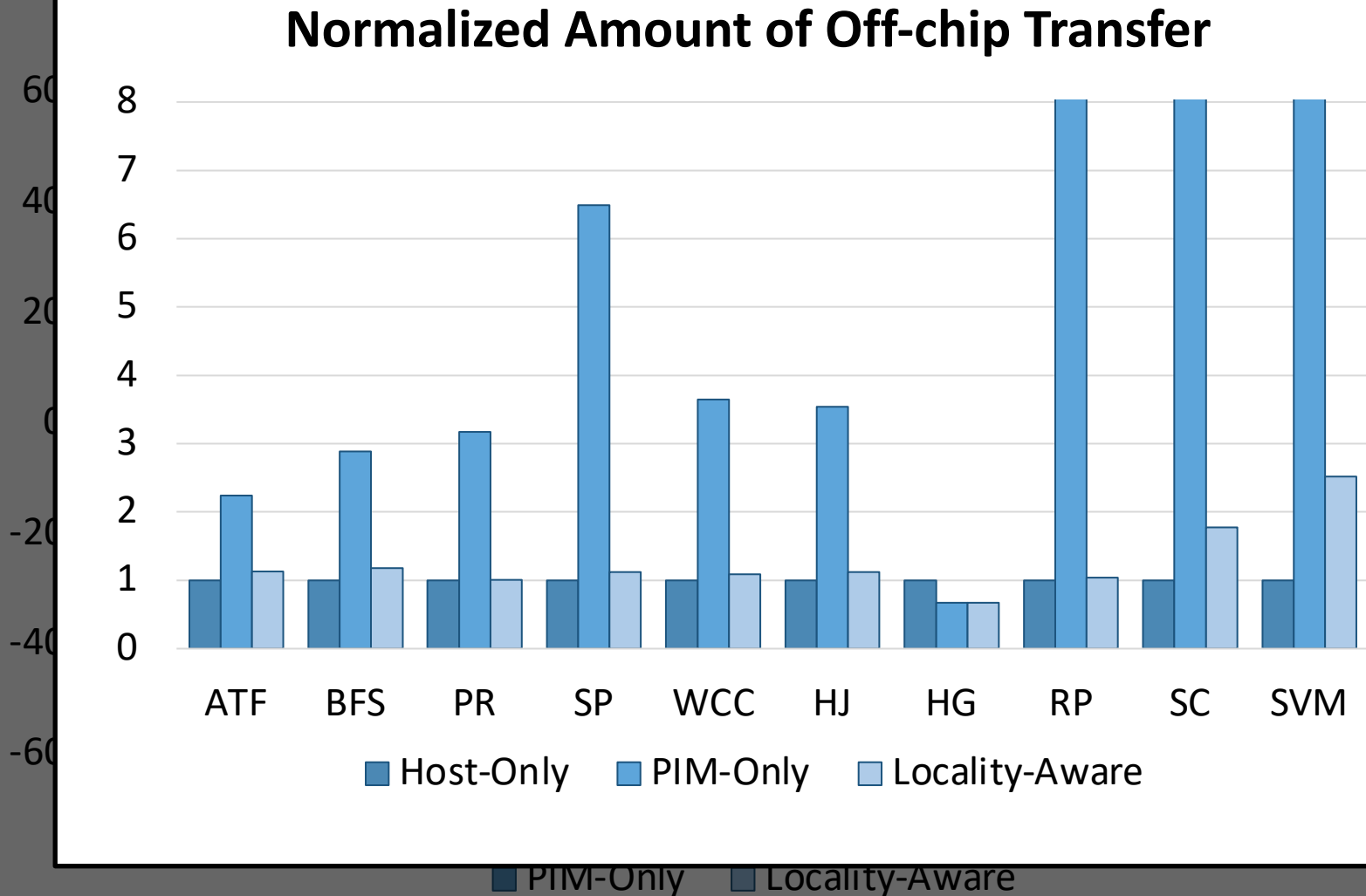


PEI Performance Delta: Small Data Sets

(Small Inputs, Baseline: Host-Only)

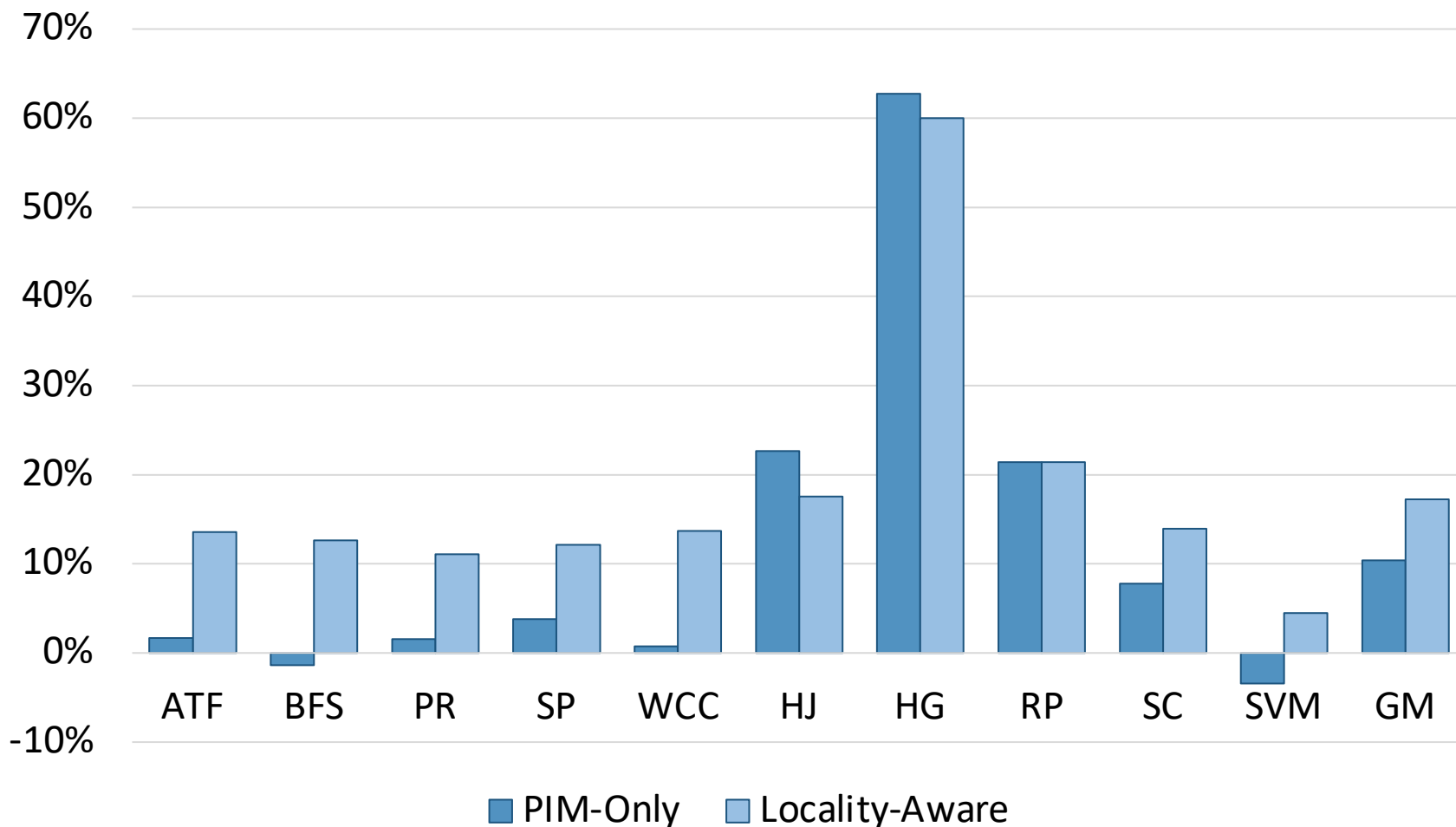


PEI Performance: Small Data Sets

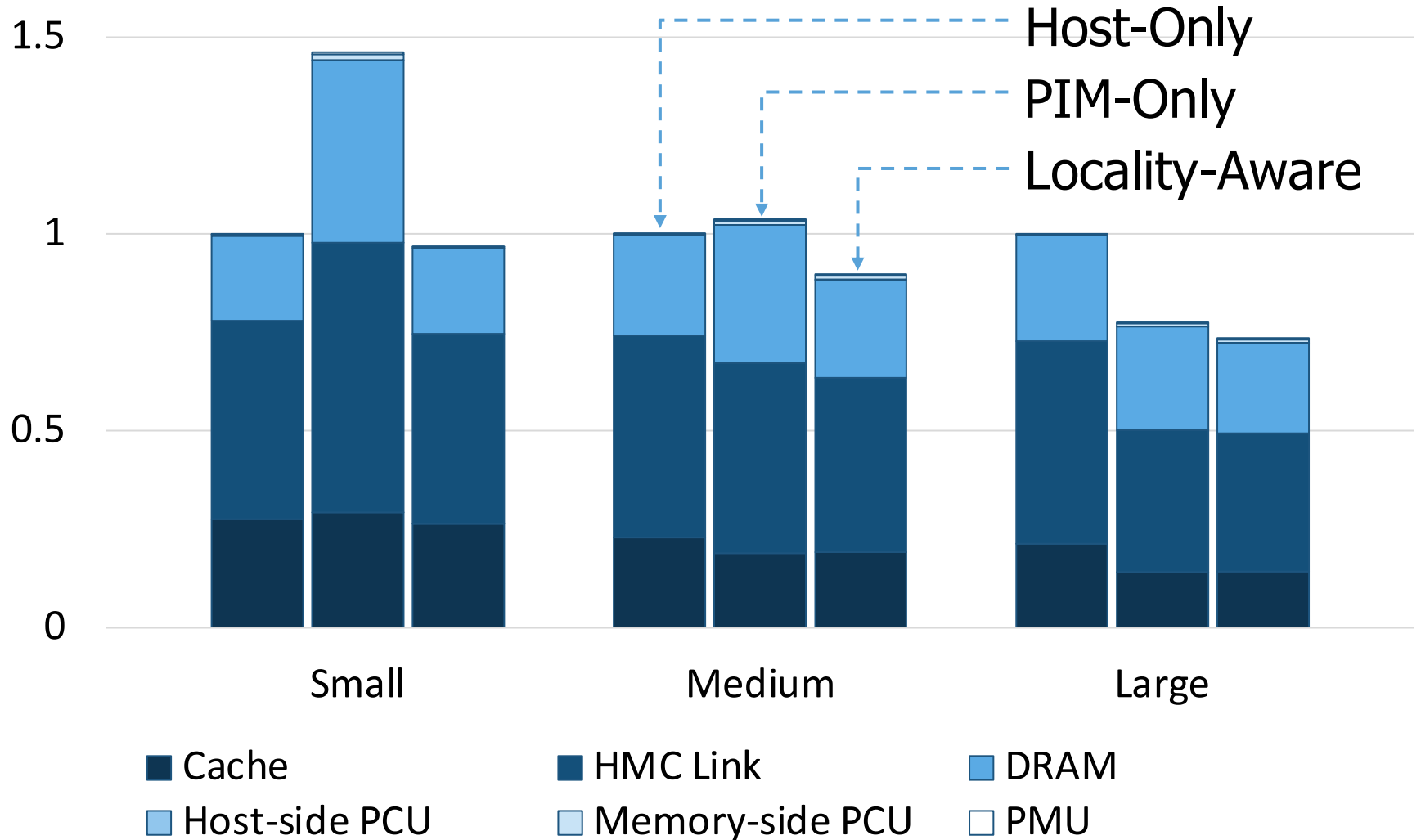


PEI Performance Delta: Medium Data Sets

(Medium Inputs, Baseline: Host-Only)



PEI Energy Consumption



PEI: Advantages & Disadvantages

■ Advantages

- + Simple and low cost approach to PIM
- + No changes to programming model, virtual memory
- + Dynamically decides where to execute an instruction

■ Disadvantages

- Does not take full advantage of PIM potential
 - Single cache block restriction is limiting

Adoption: How to Keep It Simple?

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015.
[[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoungh Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[†]Carnegie Mellon University

Adoption: How to Ease Programmability? (I)

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, ["Transparent Offloading and Mapping \(TOM\): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"](#)

Proceedings of the [43rd International Symposium on Computer Architecture \(ISCA\)](#), Seoul, South Korea, June 2016.

[\[Slides \(pptx\) \(pdf\)\]](#)

[\[Lightning Session Slides \(pptx\) \(pdf\)\]](#)

Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

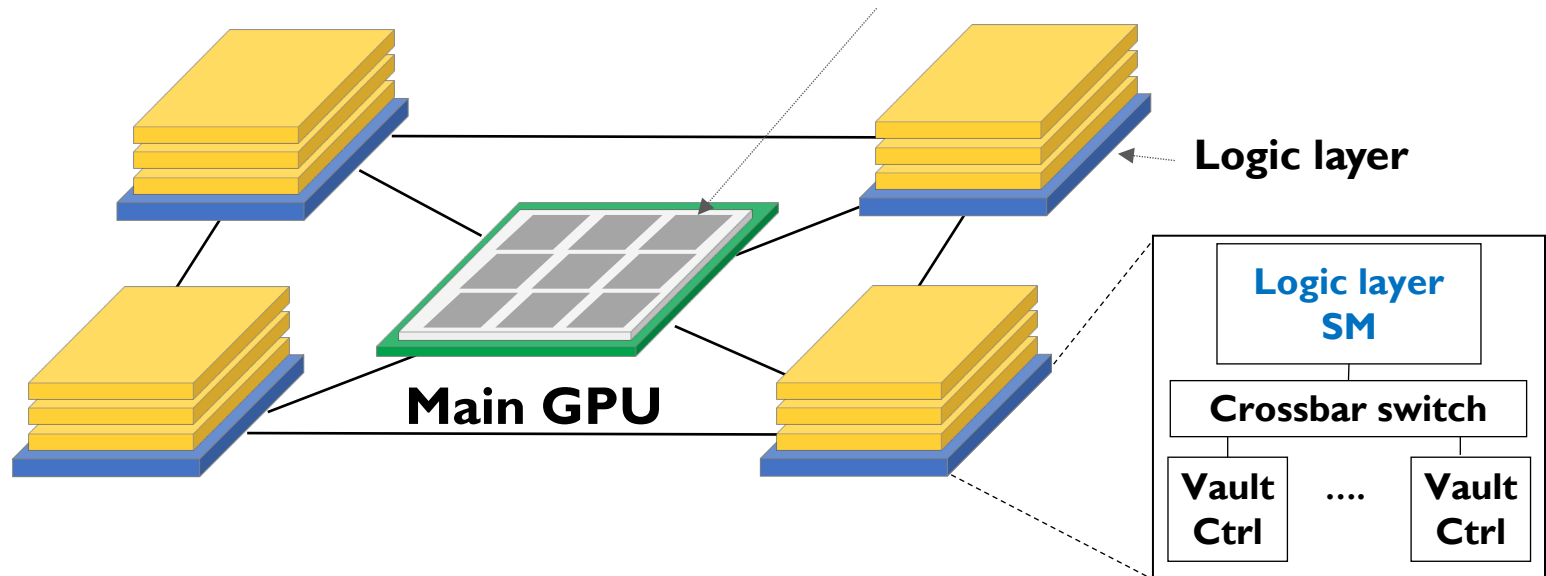
Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim^{*} Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA ^{*}KAIST [§]ETH Zürich

Truly Distributed GPU Processing with PIM

**3D-stacked memory
(memory stack)**

SM (Streaming Multiprocessor)



```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
                             uint8_T const * const in, const double *factor,
                             size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
                      sliceIdx*numRows*numCols;
```

Adoption: How to Ease Programmability? (II)

- Geraldo F. Oliveira, Alain Kohli, David Novo, Juan Gómez-Luna, Onur Mutlu,
“DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures,”
in *PACT SRC Student Competition*, Vienna, Austria, October 2023.

DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures

Geraldo F. Oliveira*

Alain Kohli*

David Novo‡

Juan Gómez-Luna*

Onur Mutlu*

**ETH Zürich*

‡*LIRMM, Univ. Montpellier, CNRS*

Adoption: How to Ease Programmability? (III)

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
"SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT), Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen¹ Juan Gómez-Luna¹ Izzat El Hajj² Yuxin Guo¹ Onur Mutlu¹
¹ETH Zürich ²American University of Beirut

Adoption: How to Ease Programmability? (IV)

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
"DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
IEEE Access, 8 September 2021.
Preprint in arXiv, 8 May 2021.
[[arXiv preprint](#)]
[[IEEE Access version](#)]
[[DAMOV Suite and Simulator Source Code](#)]
[[SAFARI Live Seminar Video](#) (2 hrs 40 mins)]
[[Short Talk Video](#) (21 minutes)]

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland

MOHAMMAD SADROSADATI, ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

Adoption: How to Maintain Coherence? (I)

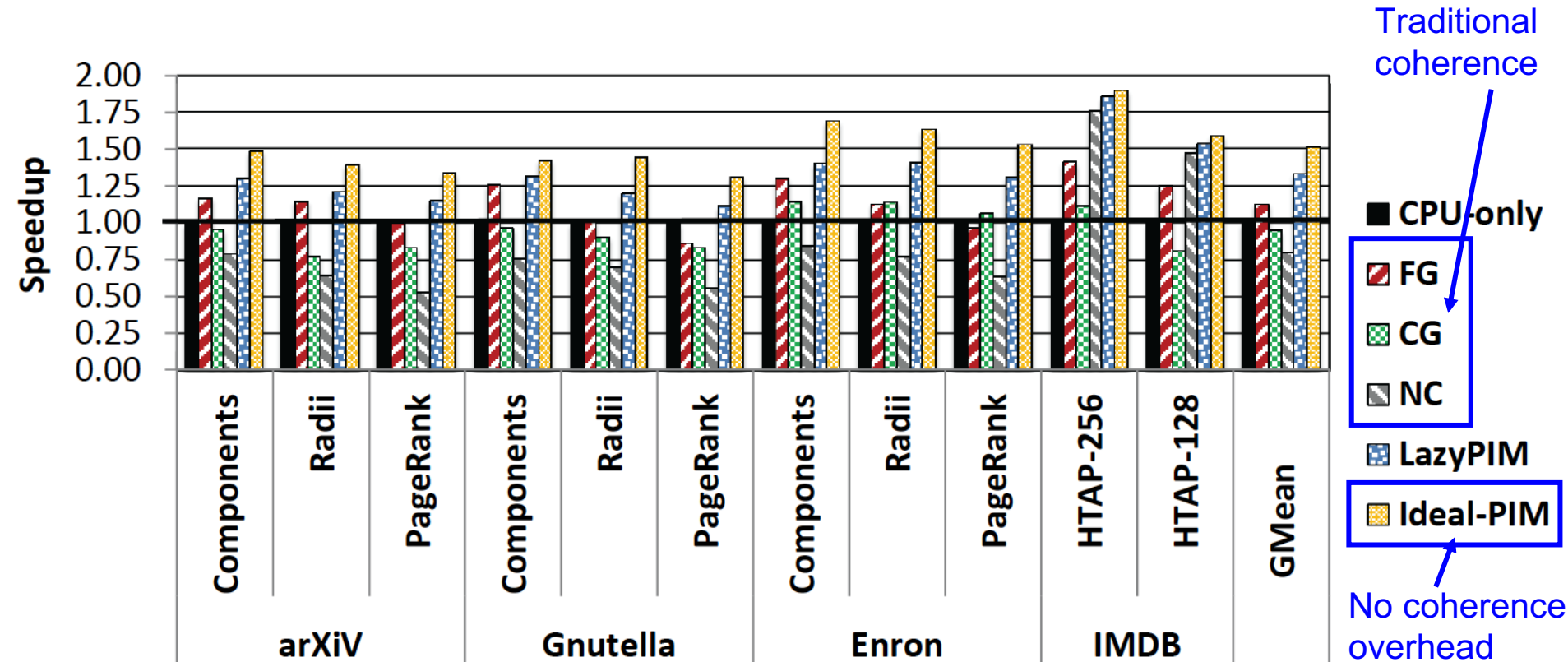
- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
"LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"
IEEE Computer Architecture Letters (CAL), June 2016.

LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand[†], Saugata Ghose[†], Minesh Patel[†], Hasan Hassan^{†§}, Brandon Lucia[†],
Kevin Hsieh[†], Krishna T. Malladi^{*}, Hongzhong Zheng^{*}, and Onur Mutlu^{††}

[†]Carnegie Mellon University ^{*}Samsung Semiconductor, Inc. [§]TOBB ETÜ [‡]ETH Zürich

Challenge: Coherence for Hybrid CPU-PIM Apps



Adoption: How to Maintain Coherence? (II)

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"

Proceedings of the 46th International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, June 2019.

CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand[†]

Saugata Ghose[†]

Minesh Patel[★]

Hasan Hassan[★]

Brandon Lucia[†]

Rachata Ausavarungnirun^{†‡}

Kevin Hsieh[†]

Nastaran Hajinazar^{◇†}

Krishna T. Malladi[§]

Hongzhong Zheng[§]

Onur Mutlu^{★†}

[†]Carnegie Mellon University

[★]ETH Zürich

[‡]KMUTNB

[◇]Simon Fraser University

[§]Samsung Semiconductor, Inc.

Adoption: How to Support Synchronization?

- Christina Giannoula, Nandita Vijaykumar, Nikela Papadopoulou, Vasileios Karakostas, Ivan Fernandez, Juan Gómez-Luna, Lois Orosa, Nectarios Koziris, Georgios Goumas, Onur Mutlu, **"SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures"**
Proceedings of the 27th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, February-March 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Short Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (21 minutes)]
[[Short Talk Video](#) (7 minutes)]

SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures

Christina Giannoula^{†‡} Nandita Vijaykumar^{*‡} Nikela Papadopoulou[†] Vasileios Karakostas[†] Ivan Fernandez^{§‡}
Juan Gómez-Luna[‡] Lois Orosa[‡] Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[†]*National Technical University of Athens* [‡]*ETH Zürich* ^{*}*University of Toronto* [§]*University of Malaga*

Adoption: How to Support Virtual Memory?

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
["Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"](#)
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh[†] Samira Khan[‡] Nandita Vijaykumar[†]
Kevin K. Chang[†] Amirali Boroumand[†] Saugata Ghose[†] Onur Mutlu^{§†}
[†]*Carnegie Mellon University* [‡]*University of Virginia* [§]*ETH Zürich*

Adoption: Evaluation Infrastructures

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
"Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
*Preprint on **arxiv**, August 2023.*
[[arXiv version](#)]
[[Ramulator 2.0 Source Code](#)]

Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

<https://arxiv.org/pdf/2308.11030.pdf>

Methodologies, Workloads, and Tools for Processing-in-Memory: Enabling the Adoption of Data-Centric Architectures

Geraldo F. Oliveira and Onur Mutlu

geraldofojunior@gmail.com

<https://geraldofojunior.github.io/>

SAFARI

ETH zürich

Processing-in-Memory: Challenges

To fully support PIM systems, we need to develop:

- 1 **Workload characterization methodologies and benchmark suites targeting PIM architectures**
- 2 **Frameworks that can facilitate the implementation of complex operations and algorithms using PIM primitives**
- 3 **Compiler support and compiler optimizations targeting PIM architectures**
- 4 **Operating system support for PIM-aware virtual memory, memory management, data allocation and mapping**
- 5 **End-to-End System-on-Chip Design Beyond DRAM**

The lack of tools and system support for PIM architectures limit the adoption of PIM systems

An Example: SimplePIM Framework

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
"SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT), Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen¹ Juan Gómez-Luna¹ Izzat El Hajj² Yuxin Guo¹ Onur Mutlu¹
¹ETH Zürich ²American University of Beirut

Programming a PIM System

- PIM programming is challenging

- Manage data movement between host DRAM and PIM DRAM
 - Parallel, serial, broadcast, and gather/scatter transfers
- Manage data movement between PIM DRAM bank and scratchpad
 - 8-byte aligned and maximum of 2,048 bytes
- Multithreaded programming model
- Inter-thread synchronization
 - Barriers, handshakes, mutexes, and semaphores

Our Goal

Design a **high-level programming framework** that abstracts these hardware-specific complexities and provides **a clean yet powerful interface** for ease of use and **high program performance**

The SimplePIM Programming Framework

- SimplePIM provides standard abstractions to build and deploy applications on PIM systems
 - Management interface
 - Metadata for PIM-resident arrays
 - Communication interface
 - Abstractions for host-PIM and PIM-PIM communication
 - Processing interface
 - Iterators (`map`, `reduce`, `zip`) to implement workloads

Productivity Improvement (I)

- Example: Hand-optimized histogram with UPMEM SDK

```
... // Initialize global variables and functions for histogram
int main_kernel() {
    if (tasklet_id == 0)
        mem_reset(); // Reset the heap
    ... // Initialize variables and the histogram
    T *input_buff_A = (T*)mem_alloc(2048); // Allocate buffer in scratchpad memory

    for (unsigned int byte_index = base_tasklet; byte_index < input_size; byte_index += stride) {
        // Boundary checking
        uint32_t l_size_bytes = (byte_index + 2048 >= input_size) ? (input_size - byte_index) : 2048;
        // Load scratchpad with a DRAM block
        mram_read((const __mram_ptr void*)(mram_base_addr_A + byte_index), input_buff_A, l_size_bytes);
        // Histogram calculation
        histogram(hist, bins, input_buff_A, l_size_bytes/sizeof(uint32_t));
    }
    ...
    barrier_wait(&my_barrier); // Barrier to synchronize PIM threads
    ... // Merging histograms from different tasklets into one histo_dpu
    // Write result from scratchpad to DRAM
    if (tasklet_id == 0)
        if (bins * sizeof(uint32_t) <= 2048)
            mram_write(histo_dpu, (__mram_ptr void*)mram_base_addr_histo, bins * sizeof(uint32_t));
        else
            for (unsigned int offset = 0; offset < ((bins * sizeof(uint32_t)) >> 11); offset++) {
                mram_write(histo_dpu + (offset << 9), (__mram_ptr void*)(mram_base_addr_histo +
                    (offset << 11)), 2048);
            }
    return 0;
}
```

Productivity Improvement (II)

- Example: SimplePIM histogram

```
// Programmer-defined functions in the file "histo_filepath"
void init_func (uint32_t size, void* ptr) {
    char* casted_value_ptr = (char*) ptr;
    for (int i = 0; i < size; i++)
        casted_value_ptr[i] = 0;
}

void acc_func (void* dest, void* src) {
    *(uint32_t*)dest += *(uint32_t*)src;
}

void map_to_val_func (void* input, void* output, uint32_t* key) {
    uint32_t d = *(uint32_t*)input;
    *(uint32_t*)output = 1;
    *key = d * bins >> 12;
}

// Host side handle creation and iterator call
handle_t* handle = simple_pim_create_handle("histo_filepath", REDUCE, NULL, 0);

// Transfer (scatter) data to PIM, register as "t1"
simple_pim_array_scatter("t1", src, bins, sizeof(T), management);

// Run histogram on "t1" and produce "t2"
simple_pim_array_red("t1", "t2", sizeof(T), bins, handle, management);
```

Productivity Improvement (III)

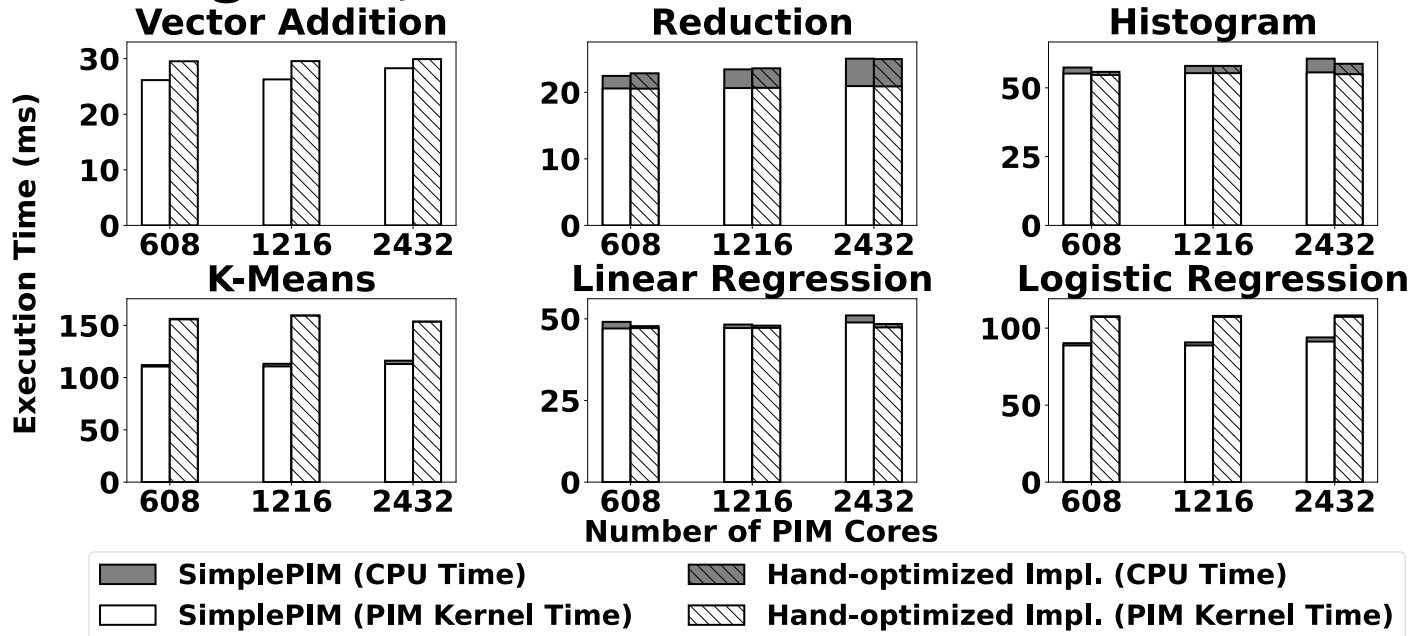
- Lines of code (LoC) reduction

	SimplePIM	Hand-optimized	LoC Reduction
Reduction	14	83	5.93×
Vector Addition	14	82	5.86×
Histogram	21	114	5.43×
Linear Regression	48	157	3.27×
Logistic Regression	59	176	2.98×
K-Means	68	206	3.03×

SimplePIM reduces the number of lines of effective code by a factor of 2.98× to 5.93×

Performance Evaluation

- Weak scaling analysis



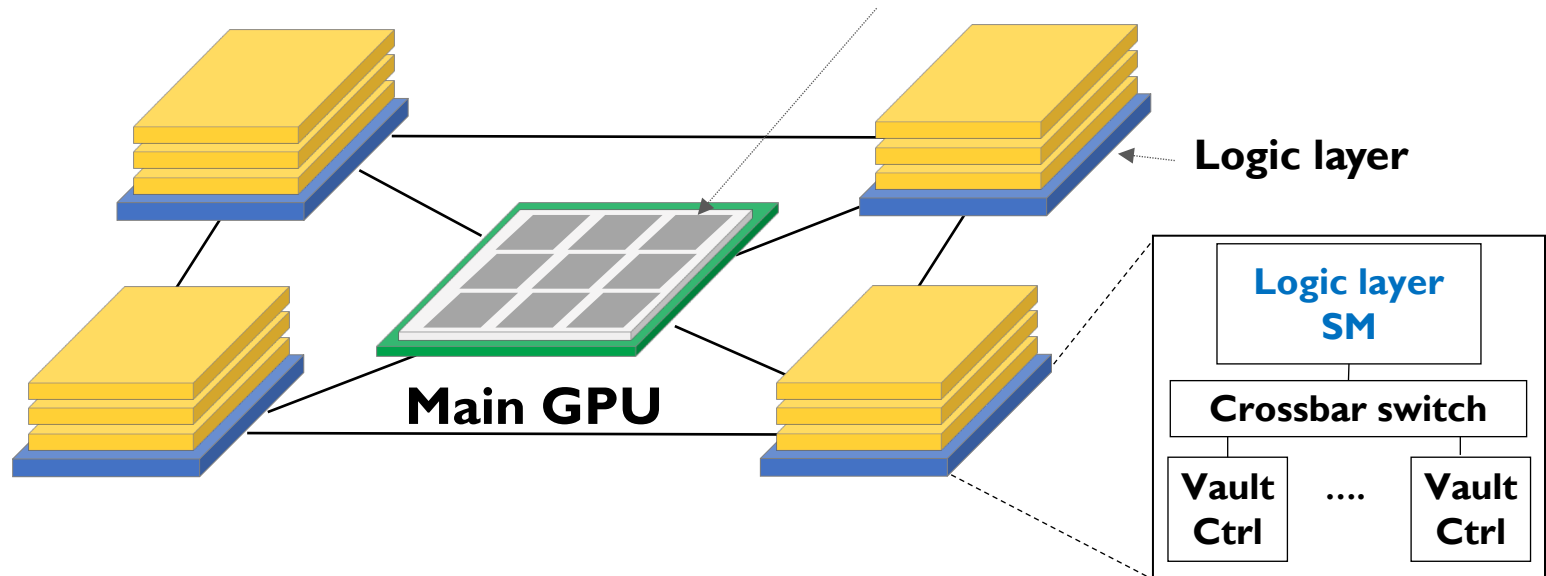
SimplePIM achieves **comparable performance** for reduction, histogram, and linear regression

SimplePIM **outperforms hand-optimized implementations** for vector addition, logistic regression, and k-means by 10%-37%

Truly Distributed GPU Processing with PIM

**3D-stacked memory
(memory stack)**

SM (Streaming Multiprocessor)



```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
                             uint8_T const * const in, const double *factor,
                             size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
                      sliceIdx*numRows*numCols;
```

Accelerating GPU Execution with PIM (I)

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**

Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim* Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA ^{*}KAIST [§]ETH Zürich

Accelerating GPU Execution with PIM (II)

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das,
"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"
Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT), Haifa, Israel, September 2016.

Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik¹ Xulong Tang¹ Adwait Jog² Onur Kayiran³
Asit K. Mishra⁴ Mahmut T. Kandemir¹ Onur Mutlu^{5,6} Chita R. Das¹

¹Pennsylvania State University ²College of William and Mary
³Advanced Micro Devices, Inc. ⁴Intel Labs ⁵ETH Zürich ⁶Carnegie Mellon University

Accelerating Linked Data Structures

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
["Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"](#)
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh[†] Samira Khan[‡] Nandita Vijaykumar[†]
Kevin K. Chang[†] Amirali Boroumand[†] Saugata Ghose[†] Onur Mutlu^{§†}
[†]*Carnegie Mellon University* [‡]*University of Virginia* [§]*ETH Zürich*

Accelerating Dependent Cache Misses

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**

Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.

[Slides (pptx) (pdf)]

[Lightning Session Slides (pptx) (pdf)]

Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi*, Khubaib[†], Eiman Ebrahimi[‡], Onur Mutlu[§], Yale N. Patt*

**The University of Texas at Austin [†]Apple [‡]NVIDIA [§]ETH Zürich & Carnegie Mellon University*

Accelerating Runahead Execution

- Milad Hashemi, Onur Mutlu, and Yale N. Patt,
"Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"
Proceedings of the 49th International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, October 2016.
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pdf\)\]](#) [\[Poster \(pptx\) \(pdf\)\]](#)
Best paper session.

Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi*, Onur Mutlu[§], Yale N. Patt*

**The University of Texas at Austin* [§]*ETH Zürich*

Accelerating Climate Modeling

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, September 2020.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (23 minutes)]
Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh^{a,b,c} Dionysios Diamantopoulos^c Christoph Hagleitner^c Juan Gómez-Luna^b
Sander Stuijk^a Onur Mutlu^b Henk Corporaal^a
^aEindhoven University of Technology ^bETH Zürich ^cIBM Research Europe, Zurich

Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zülal Bingöl, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**
Proceedings of the 53rd International Symposium on Microarchitecture (MICRO), Virtual, October 2020.
[[Lighting Talk Video](#) (1.5 minutes)]
[[Lightning Talk Slides \(pptx\)](#) ([pdf](#))]
[[Talk Video](#) (18 minutes)]
[[Slides \(pptx\)](#) ([pdf](#))]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali^{†⋈} Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim^{◇†}
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose^{*†} Onur Mutlu^{◇†▽}
[†]Carnegie Mellon University [⋈]Processor Architecture Research Lab, Intel Labs [▽]Bilkent University [◇]ETH Zürich
[‡]Facebook [⊙]King Mongkut's University of Technology North Bangkok ^{*}University of Illinois at Urbana-Champaign

Accelerating Sequence-to-Graph Mapping

- Damla Senol Cali, Konstantinos Kanellopoulos, Joel Lindegger, Zülal Bingöl, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie Kim, Nika Mansouri Ghiasi, Gagandeep Singh, Juan Gomez-Luna, Nour Almadhoun Alserr, Mohammed Alser, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
"SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping"
Proceedings of the 49th International Symposium on Computer Architecture (ISCA), New York, June 2022.
[[arXiv version](#)]

SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

Damla Senol Cali¹ Konstantinos Kanellopoulos² Joël Lindegger² Zülal Bingöl³
Gurpreet S. Kalsi⁴ Ziyi Zuo⁵ Can Firtina² Meryem Banu Cavlak² Jeremie Kim²
Nika Mansouri Ghiasi² Gagandeep Singh² Juan Gómez-Luna² Nour Almadhoun Alserr²
Mohammed Alser² Sreenivas Subramoney⁴ Can Alkan³ Saugata Ghose⁶ Onur Mutlu²

¹Bionano Genomics ²ETH Zürich ³Bilkent University ⁴Intel Labs
⁵Carnegie Mellon University ⁶University of Illinois Urbana-Champaign

Accelerating Basecalling + Read Mapping

- Haiyu Mao, Mohammed Alser, Mohammad Sadrosadati, Can Firtina, Akanksha Baranwal, Damla Senol Cali, Aditya Manglik, Nour Almadhoun Alserr, and Onur Mutlu,
"GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping"
Proceedings of the 55th International Symposium on Microarchitecture (MICRO),
Chicago, IL, USA, October 2022.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Longer Lecture Slides \(pptx\)](#)] [[pdf](#)]
[[Lecture Video](#) (25 minutes)]
[[arXiv version](#)]

GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping

Haiyu Mao¹ Mohammed Alser¹ Mohammad Sadrosadati¹ Can Firtina¹ Akanksha Baranwal¹
Damla Senol Cali² Aditya Manglik¹ Nour Almadhoun Alserr¹ Onur Mutlu¹
¹*ETH Zürich* ²*Bionano Genomics*

Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quisiant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,
"NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
Proceedings of the 38th IEEE International Conference on Computer Design (ICCD), Virtual, October 2020.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (10 minutes)]
[[Source Code](#)]

NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez[§]

Ricardo Quisiant[§]

Christina Giannoula[†]

Mohammed Alser[‡]

Juan Gómez-Luna[‡]

Eladio Gutiérrez[§]

Oscar Plata[§]

Onur Mutlu[‡]

[§]*University of Malaga*

[†]*National Technical University of Athens*

[‡]*ETH Zürich*

Accelerating Graph Pattern Mining

- 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.

[[Slides \(pdf\)](#)]

[[Talk Video](#) (22 minutes)]

[[Lightning Talk Video](#) (1.5 minutes)]

[[Full arXiv version](#)]

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

Maciej Besta¹, Raghavendra Kanakagiri², Grzegorz Kwasniewski¹, Rachata Ausavarungnirun³, Jakub Beránek⁴, Konstantinos Kanellopoulos¹, Kacper Janda⁵, Zur Vonarburg-Shmaria¹, Lukas Gianinazzi¹, Ioana Stefan¹, Juan Gómez-Luna¹, Marcin Copik¹, Lukas Kapp-Schwoerer¹, Salvatore Di Girolamo¹, Nils Blach¹, Marek Konieczny⁵, Onur Mutlu¹, Torsten Hoefler¹

¹ETH Zurich, Switzerland
Thailand

²IIT Tirupati, India

³King Mongkut's University of Technology North Bangkok,
⁴Technical University of Ostrava, Czech Republic

⁵AGH-UST, Poland

Accelerating HTAP Database Systems

- Amirali Boroumand, Saugata Ghose, Geraldo F. Oliveira, and Onur Mutlu,
"Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design"
Proceedings of the 38th International Conference on Data Engineering (ICDE),
Virtual, May 2022.
[[arXiv version](#)]
[[Slides \(pptx\)](#) ([pdf](#))]
[[Short Talk Slides \(pptx\)](#) ([pdf](#))]

Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

Amirali Boroumand[†]
[†]*Google*

Saugata Ghose[◇]
[◇]*Univ. of Illinois Urbana-Champaign*

Geraldo F. Oliveira[‡]
[‡]*ETH Zürich*

Onur Mutlu[‡]

Mensa: Highly-Efficient ML Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Virtual, September 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand^{†◇}

Geraldo F. Oliveira^{*}

Saugata Ghose[‡]

Xiaoyu Ma[§]

Berkin Akin[§]

Eric Shiu[§]

Ravi Narayanaswami[§]

Onur Mutlu^{*†}

[†]*Carnegie Mellon Univ.*

[◇]*Stanford Univ.*

[‡]*Univ. of Illinois Urbana-Champaign*

[§]*Google*

^{*}*ETH Zürich*

Accelerating Data-Intensive Workloads

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoun Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"**
Proceedings of the 42nd International Symposium on Computer Architecture (ISCA), Portland, OR, June 2015.
[[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoun Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[†]Carnegie Mellon University

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu, ["FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"](#)
IEEE Micro (**IEEE MICRO**), 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] Mohammed Alser[◇] Damla Senol Cali[✕]

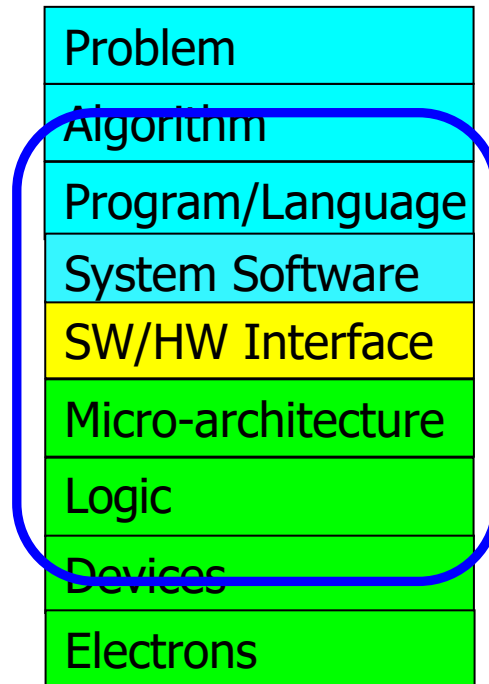
Dionysios Diamantopoulos[▽] Juan Gómez-Luna[◇]

Henk Corporaal^{*} Onur Mutlu^{◇✕}

[◇]*ETH Zürich* [✕]*Carnegie Mellon University*

^{*}*Eindhoven University of Technology* [▽]*IBM Research Europe*

We Need to Revisit the Entire Stack



We can get there step by step

Processing-in-Memory in the Real World

PIM Tutorial at ISCA 2024

ISCA 2024 Memory-Centric Computing Systems Tutorial

Saturday, June 29, Buenos Aires, Argentina

Organizers: Geraldo F. Oliveira, Dr. Mohammad Sadrosadati, Ataberk Olgun, Professor Onur Mutlu

Program: <https://events.safari.ethz.ch/isca24-memorycentric-tutorial/>

Overview of PIM | PIM taxonomy
PIM in memory & storage
Real-world PNM systems
PUM for bulk bitwise operations
Programming techniques & tools
Infrastructures for PIM Research
Research challenges & opportunities




<https://www.youtube.com/watch?v=KV2MXvcBgb0>

<https://events.safari.ethz.ch/isca24-memorycentric-tutorial>

PIM Tutorials [MICRO'23, ISCA'23, ASPLOS'23, HPCA'23, ISCA'24]

■ Lectures + Hands-on labs + Invited talks



ISCALogos

ISCALogos 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • start

Real-world Processing-in-Memory Systems for Modern Workloads

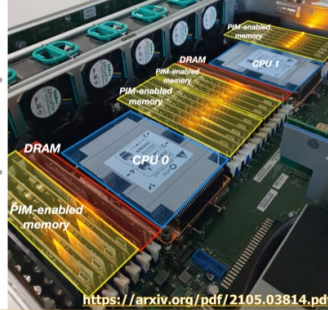
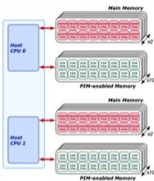
Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).

2,560-DPU Processing-in-Memory System



<https://arxiv.org/pdf/2105.03814.pdf>

Table of Contents

- Real-world Processing-in-Memory Systems for Modern Workloads
- Tutorial Description
- Organizers
- Agenda (June 18, 2023)
- Lectures (tentative)
- Hands-on Labs (tentative)
- Learning Materials

PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3) develop programming frameworks and tools that can lower the learning curve and ease the adoption of PIM.

This tutorial focuses on the latest advances in PIM technology, workload characterization for PIM, and programming and optimizing PIM kernels. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hand-on labs about important workloads (machine learning, sparse linear algebra, bioinformatics, etc.) using real PIM systems, and (4) shed light on how to improve future PIM systems for such workloads.

<https://www.youtube.com/live/GIb5EqSrWk0>

<https://events.safari.ethz.ch/isca-pim-tutorial/>

Real PIM Tutorial [ISCA 2023]

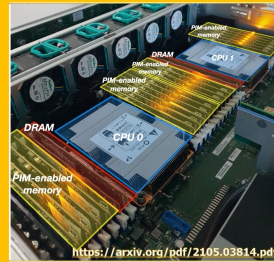
■ June 18: Lectures + Hands-on labs + Invited talks

ISCA 2023 Real-World PIM Tutorial Sunday, June 18, Orlando, Florida

Organizers: Juan Gómez Luna, Onur Mutlu, Ataberk Olgun
Program: <https://events.safari.ethz.ch/isca-pim-tutorial/>



Overview PIM | PNM | UPMEM PIM |
PNM for neural networks |
PNM for recommender systems |
PNM for ML workloads |
How to enable PIM? | PUM prototypes
Hands-on Labs: Benchmarking |
Accelerating real-world workloads



International Symposium on Computer Architecture (ISCA)

Real-world Processing-in-Memory Systems for Modern Workloads

<https://www.youtube.com/live/GIb5EgSrWk0?feature=share>

Room: Magnolia 16
Marriott World Center Orlando
Orlando, FL, USA
July 18th, 2023

SAFARI zoom

ISCA 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads

Onur Mutlu Lectures
33.9K subscribers

Subscribed

57

Share

Download

Clip

...

1,687 views Streamed live on Jun 18, 2023 Livestream - Data-Centric Architectures: Fundamentally Improving Performance and Energy (Spring 2023)

[https://www.youtube.com/
live/GIb5EgSrWk0](https://www.youtube.com/live/GIb5EgSrWk0)

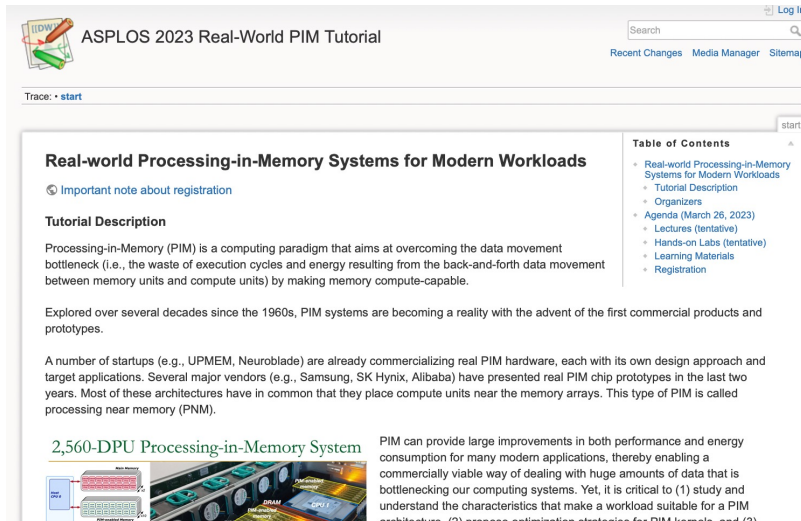
[https://events.safari.ethz.ch/
isca-pim-tutorial/](https://events.safari.ethz.ch/isca-pim-tutorial/)

Tutorial Materials

Time	Speaker	Title	Materials
8:55am-9:00am	Dr. Juan Gómez Luna	Welcome & Agenda	(PDF) (PPT)
9:00am-10:20am	Prof. Onur Mutlu	Memory-Centric Computing	(PDF) (PPT)
10:20am-11:00am	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures / Programming General-purpose PIM	(PDF) (PPT)
11:20am-11:50am	Prof. Izzat El Hajj	High-throughput Sequence Alignment using Real Processing-in-Memory Systems	(PDF) (PPT)
11:50am-12:30pm	Dr. Christina Giannoula	SparseP: Towards Efficient Sparse Matrix Vector Multiplication for Real Processing-In-Memory Systems	(PDF) (PPT)
2:00pm-2:45pm	Dr. Sukhan Lee	Introducing Real-world HBM-PIM Powered System for Memory-bound Applications	(PDF) (PPT)
2:45pm-3:30pm	Dr. Juan Gómez Luna / Ataberk Olgun	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components / PUM Prototypes: PiDRAM	(PDF) (PPT) (PPT)
4:00pm-4:40pm	Dr. Juan Gómez Luna	Accelerating Modern Workloads on a General-purpose PIM System	(PDF) (PPT)
4:40pm-5:20pm	Dr. Juan Gómez Luna	Adoption Issues: How to Enable PIM?	(PDF) (PPT)
5:20pm-5:30pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	(Handout) (PDF) (PPT)

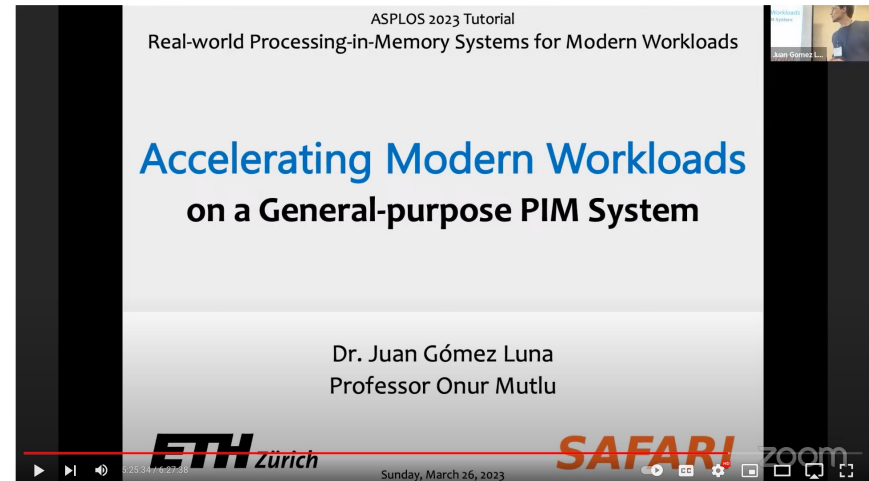
Real PIM Tutorial [ASPLOS 2023]

■ March 26: Lectures + Hands-on labs + Invited talks



Tutorial Materials

Time	Speaker	Title	Materials
9:00am-10:20am	Prof. Onur Mutlu	Memory-Centric Computing	(PDF) (PPT)
10:40am-12:00pm	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures Programming General-purpose PIM	(PDF) (PPT)
1:40pm-2:20pm	Prof. Alexandra (Sasha) Fedorova (UBC)	Processing in Memory in the Wild	(PDF) (PPT)
2:20pm-3:20pm	Dr. Juan Gómez Luna & Ataberk Olgun	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components	(PDF) (PPT) (PDF) (PPT)
3:40pm-4:10pm	Dr. Juan Gómez Luna	Adoption issues: How to enable PIM? Accelerating Modern Workloads on a General-purpose PIM System	(PDF) (PPT) (PDF) (PPT)
4:10pm-4:50pm	Dr. Yongkee Kwon & Eddy (Chanwook) Park (SK Hynix)	System Architecture and Software Stack for GDDR6-AiM	(PDF) (PPT)
4:50pm-5:00pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	(Handout) (PDF) (PPT)



ASPLOS 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads

Onur Mutlu Lectures
32.1K subscribers

Subscribed

33

Share

Clip

Save

...

views Streamed 7 days ago Livestream - Data-Centric Architectures: Fundamentally Improving Performance and Energy (Spring 2023)

LOS 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads

<https://events.safari.ethz.ch/asplos-2023/>

<https://www.youtube.com/watch?v=oYCaLcT0Kmo>

<https://events.safari.ethz.ch/asplos-pim-tutorial/>

Real PIM Tutorial [HPCA 2023]

February 26: Lectures + Hands-on labs + Invited Talks

HPCA 2023 Real-World PIM Tutorial

Trace: - start

Real-world Processing-in-Memory Architectures

Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade, Mythic) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Allbaba) have presented real PIM chip prototypes in the last two years.

2,560-DPU Processing-in-Memory System

Most of these architectures have in common that they place compute units near the memory arrays. But, there is more to come: Academia and Industry are actively exploring other types of PIM by, e.g., exploiting the analog operation of DRAM, SRAM, flash memory and emerging non-volatile memories.

PIM can provide large improvements in both performance and energy consumption, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to examine and research adoption issues of PIM using especially learnings from real PIM systems that are available today.

This tutorial focuses on the latest advances in PIM technology. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hand-on labs using real PIM systems, and (4) shed light on how to enable the adoption of PIM in future computing systems.

Goal: Processing Inside Memory

Processor Core

Memory

Database

Graphs

Media

Query

Results

Interconnect

- Many questions ... How do we design the:
 - compute-capable memory & controllers?
 - processors & communication units?
 - software & hardware interfaces?
 - system software, compilers, languages?
 - algorithms & theoretical foundations?

HPCA 2023 Tutorial: Real-World Processing-in-Memory Architectures

Onur Mutlu Lectures

32.1K subscribers

1.8K views Streamed 1 month ago Livestream - P&S Data-Centric Architectures: Fundamentally Improving Performance and Energy (Fall 2022)

HPCA 2023 Tutorial: Real-World Processing-in-Memory Architectures

<https://events.safari.ethz.ch/real-pi...>

Time	Speaker	Title	Materials
8:00am-8:40am	Prof. Onur Mutlu	Memory-Centric Computing	PDF PPT
8:40am-10:00am	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures Programming General-purpose PIM	PDF PPT
10:20am-11:00am	Dr. Dimin Niu	A 3D Logic-to-DRAM Hybrid Bonding Process-Near-Memory Chip for Recommendation System	
11:00am-11:40am	Dr. Christina Giannoula	SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures	PDF PPT
1:30pm-2:10pm	Dr. Juan Gómez Luna	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components	PDF PPT
2:10pm-2:50pm	Dr. Manuel Le Gallo	Deep Learning Inference Using Computational Phase-Change Memory	
2:50pm-3:30pm	Dr. Juan Gómez Luna	PIM Adoption Issues: How to Enable PIM Adoption?	PDF PPT
3:40pm-5:40pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	Handout PDF PPT

<https://www.youtube.com/watch?v=f5-nT1tbz5w>

<https://events.safari.ethz.ch/real-pim-tutorial/>

Real PIM Tutorial [MICRO 2023]

■ October 29: Lectures + Hands-on labs + Invited talks

Real-world Processing-in-Memory Systems for Modern Workloads

Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).

2,560-DPU Processing-in-Memory System

PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3) develop programming frameworks and tools that can lower the learning curve and ease the adoption of PIM.

This tutorial focuses on the latest advances in PIM technology, workload characterization for PIM, and programming and optimizing PIM kernels. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hands-on labs about important workloads (machine learning, sparse linear algebra, bioinformatics, etc.) using real PIM systems, and (4) shed light on how to improve future PIM systems for such workloads.

<https://arxiv.org/pdf/2105.03814.pdf>

2,560-DPU Processing-in-Memory System

<https://arxiv.org/pdf/2105.03814.pdf>

MICRO 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads

Onur Mutlu Lectures
34.6K subscribers

<https://www.youtube.com/watch?v=ohUooNSIxOI>

<https://events.safari.ethz.ch/micro-pim-tutorial>

Agenda (Tentative, October 29, 2023)

Lectures

1. Introduction: PIM as a paradigm to overcome the data movement bottleneck.
2. PIM taxonomy: PNM (processing near memory) and PUM (processing using memory).
3. General-purpose PNM: UPMEM PIM.
4. PNM for neural networks: Samsung HBM-PIM, SK Hynix AiM.
5. PNM for recommender systems: Samsung AxDIMM, Alibaba PNM.
6. PUM prototypes: PiDRAM, SRAM-based PUM, Flash-based PUM.
7. Other approaches: Neuroblade, Mythic.
8. Adoption issues: How to enable PIM?
9. Hands-on labs: Programming a real PIM system.

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu, ["FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"](#) *IEEE Micro* (**IEEE MICRO**), 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] Mohammed Alser[◇] Damla Senol Cali[✕]

Dionysios Diamantopoulos[▽] Juan Gómez-Luna[◇]

Henk Corporaal^{*} Onur Mutlu^{◇✕}

[◇]*ETH Zürich* [✕]*Carnegie Mellon University*

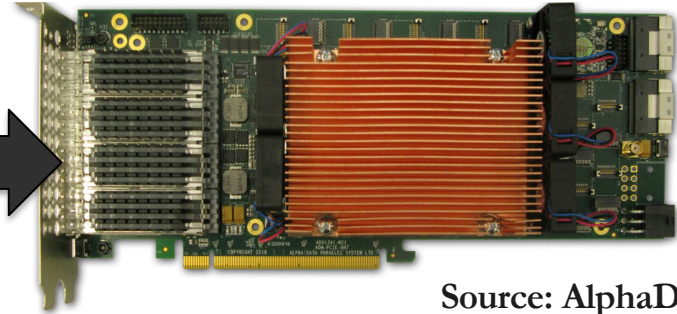
^{*}*Eindhoven University of Technology* [▽]*IBM Research Europe*

Near-Memory Acceleration using FPGAs



Source: IBM

IBM POWER9 CPU



Source: AlphaData

HBM-based FPGA board

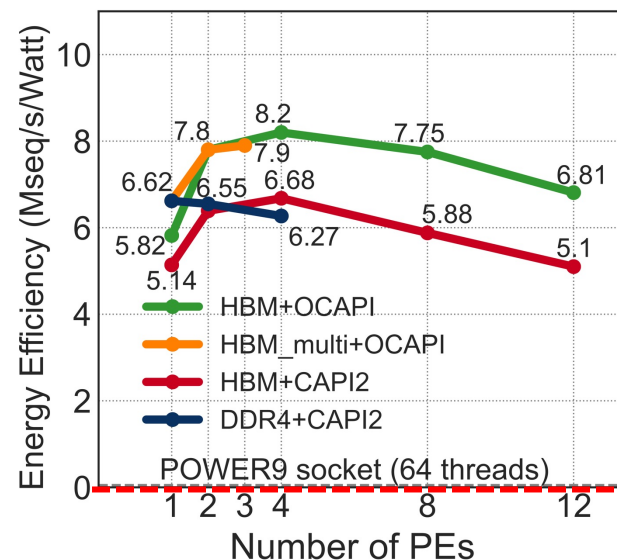
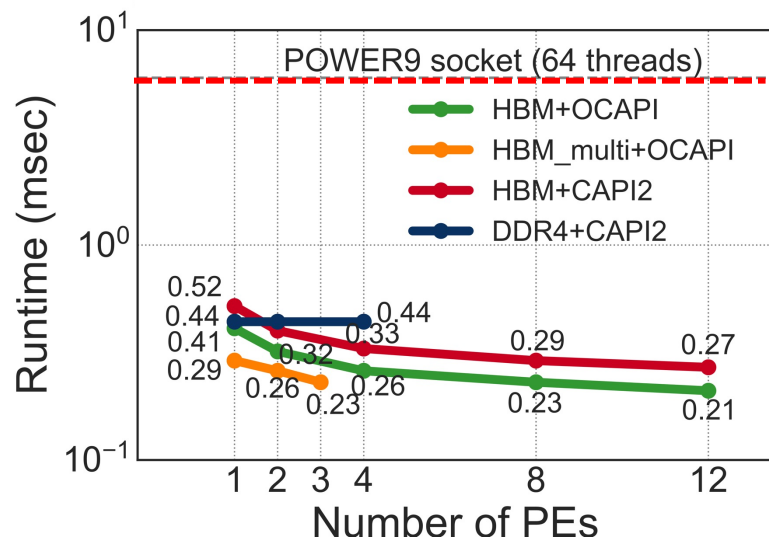
Near-HBM FPGA-based accelerator

Two communication technologies: CAPI2 and OCAPI

Two memory technologies: DDR4 and HBM

Two workloads: Weather Modeling and Genome Analysis

Performance & Energy Greatly Improve



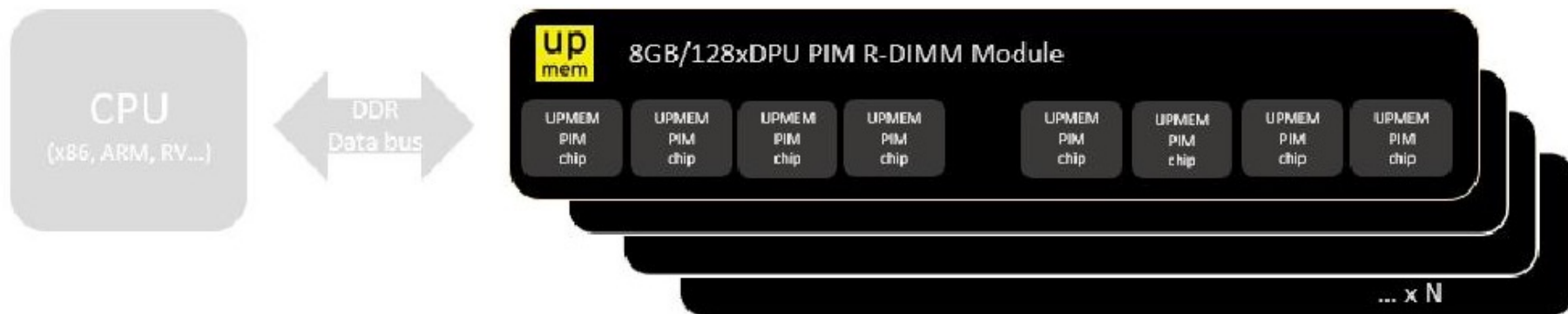
5-27× performance vs. a 16-core (64-thread) IBM POWER9 CPU

12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

HBM alleviates memory bandwidth contention vs. DDR4

UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
 - DDR4 R-DIMM modules
 - 8GB+128 DPUs (16 PIM chips)
 - Standard 2x-nm DRAM process
 - **Large amounts of** compute & memory bandwidth



Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

IZZAT EL HAJJ, American University of Beirut, Lebanon

IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

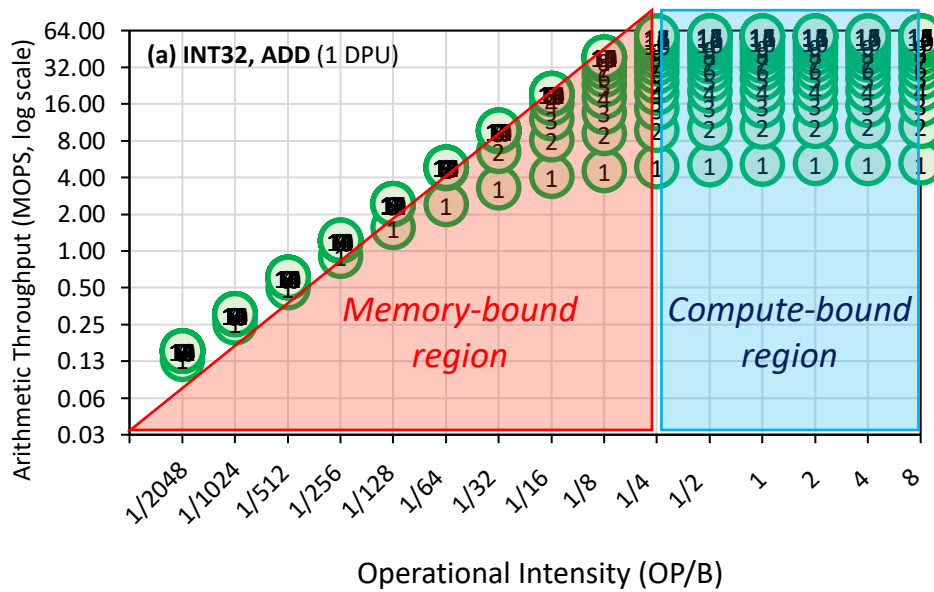
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this *data movement bottleneck* requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as *processing-in-memory* (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called *DRAM Processing Units* (DPUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM* (*Processing-In-Memory benchmarks*), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

Key Takeaway 1



The throughput saturation point is as low as $\frac{1}{4}$ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

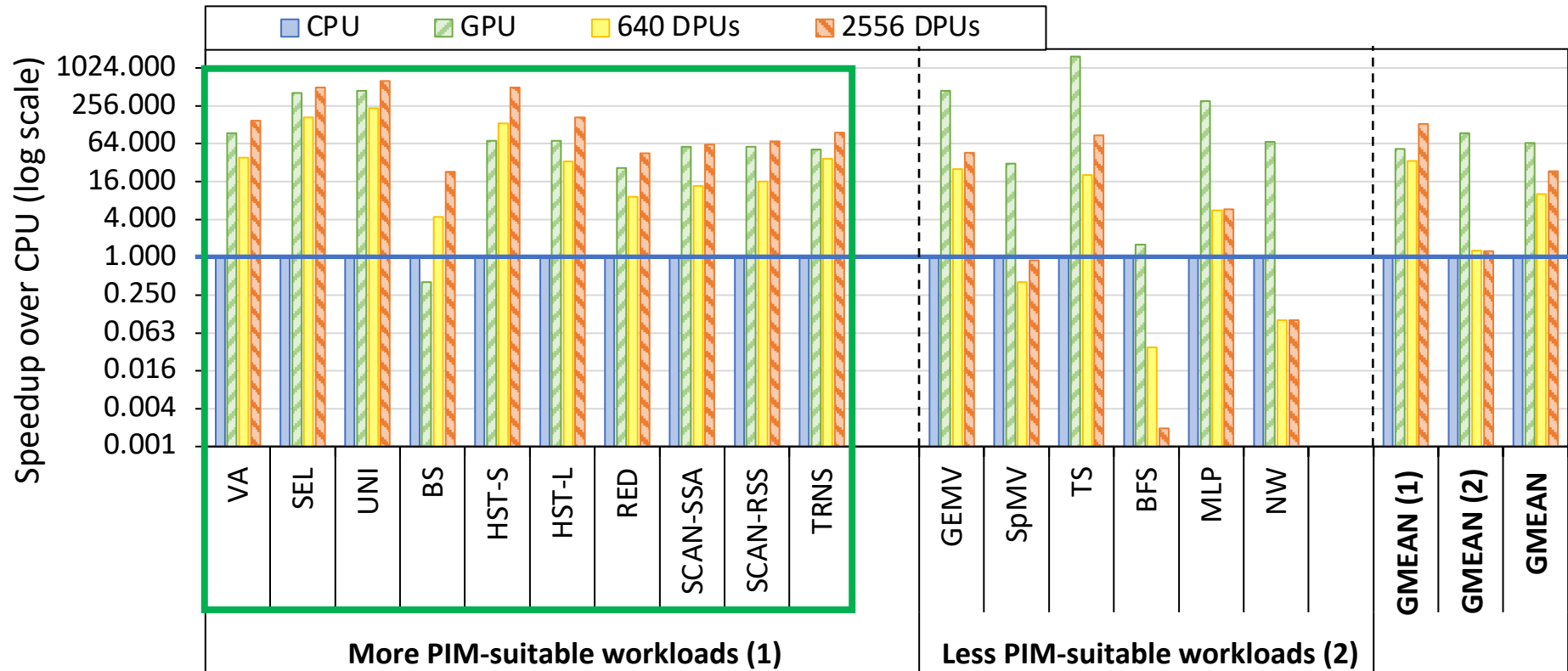
The UPMEM PIM architecture is fundamentally compute bound. As a result, the most suitable workloads are memory-bound.

Key Takeaway 2

Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS*	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556 ⁹	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W [†]
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W [†]

* Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.
⁹ Estimated TDP = $\frac{\text{Total DPU}}{\text{DPU}_{\text{chip}}}$ × 1.2 W/chip [199].



KEY TAKEAWAY 2

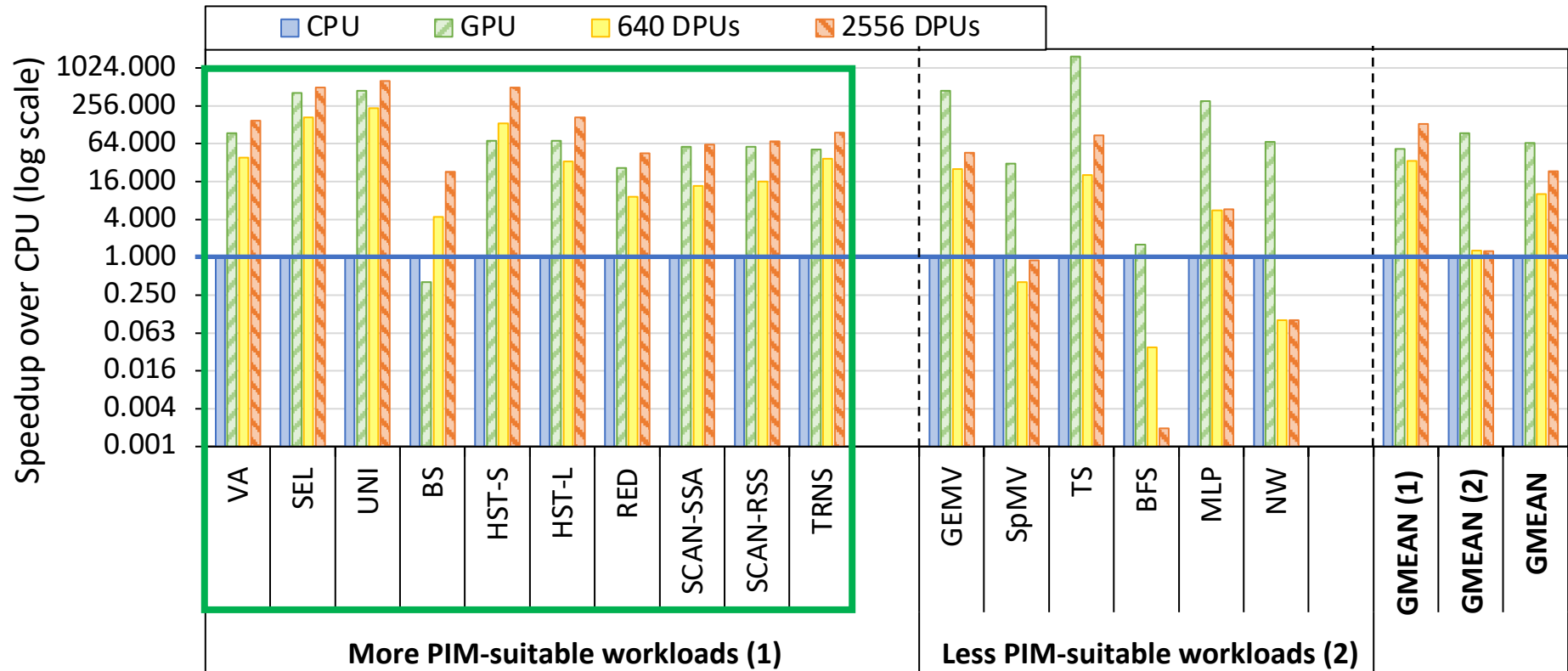
The most well-suited workloads for the UPMEM PIM architecture use no arithmetic operations or use only simple operations (e.g., bitwise operations and integer addition/subtraction).

Key Takeaway 3

Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS*	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556 [†]	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W [†]
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W [†]

*Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.
[†]Estimated TDP = $\frac{\text{Total DPU}}{\text{DPU}_1/\text{chip}} \times 1.2 \text{ W/chip}$ [199].



KEY TAKEAWAY 3

The most well-suited workloads for the UPMEM PIM architecture require little or no communication across DPUs (inter-DPU communication).

UPMEM PIM System Summary & Analysis

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"
*Invited Paper at Workshop on Computing with Unconventional Technologies (**CUT**), Virtual, October 2021.*
[[arXiv version](#)]
[[PrIM Benchmarks Source Code](#)]
[[Slides \(pptx\)](#) ([pdf](#))]
[[Talk Video](#) (37 minutes)]
[[Lightning Talk Video](#) (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna
ETH Zürich

Izzat El Hajj
*American University
of Beirut*

Ivan Fernandez
*University
of Malaga*

Christina Giannoula
*National Technical
University of Athens*

Geraldo F. Oliveira
ETH Zürich

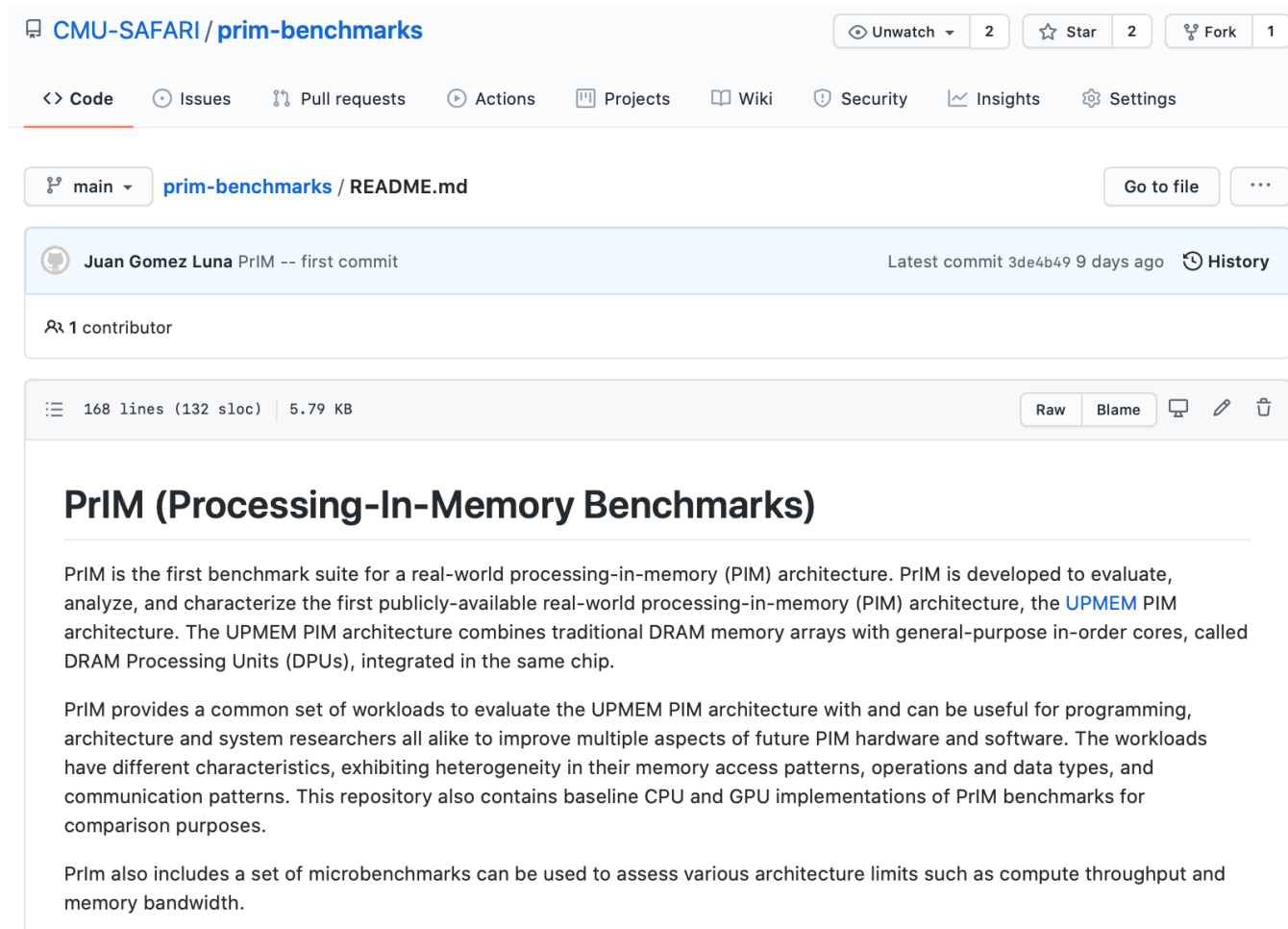
Onur Mutlu
ETH Zürich

PrIM Benchmarks: Application Domains

Domain	Benchmark	Short name
Dense linear algebra	Vector Addition	VA
	Matrix-Vector Multiply	GEMV
Sparse linear algebra	Sparse Matrix-Vector Multiply	SpMV
Databases	Select	SEL
	Unique	UNI
Data analytics	Binary Search	BS
	Time Series Analysis	TS
Graph processing	Breadth-First Search	BFS
Neural networks	Multilayer Perceptron	MLP
Bioinformatics	Needleman-Wunsch	NW
Image processing	Image histogram (short)	HST-S
	Image histogram (large)	HST-L
Parallel primitives	Reduction	RED
	Prefix sum (scan-scan-add)	SCAN-SSA
	Prefix sum (reduce-scan-scan)	SCAN-RSS
	Matrix transposition	TRNS

PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- <https://github.com/CMU-SAFARI/prim-benchmarks>



CMU-SAFARI / **prim-benchmarks** Unwatch 2 Star 2 Fork 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main prim-benchmarks / README.md Go to file ...

Juan Gomez Luna PrIM -- first commit Latest commit 3de4b49 9 days ago History

1 contributor

168 lines (132 sloc) 5.79 KB Raw Blame

PrIM (Processing-In-Memory Benchmarks)

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the [UPMEM](#) PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

PrIM provides a common set of workloads to evaluate the UPMEM PIM architecture with and can be useful for programming, architecture and system researchers all alike to improve multiple aspects of future PIM hardware and software. The workloads have different characteristics, exhibiting heterogeneity in their memory access patterns, operations and data types, and communication patterns. This repository also contains baseline CPU and GPU implementations of PrIM benchmarks for comparison purposes.

PrIm also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.

Understanding a Modern PIM Architecture

Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System

**JUAN GÓMEZ-LUNA¹, IZZAT EL HAJJ², IVAN FERNANDEZ^{1,3}, CHRISTINA GIANNOULA^{1,4},
GERALDO F. OLIVEIRA¹, AND ONUR MUTLU¹**

¹ETH Zürich

²American University of Beirut

³University of Malaga

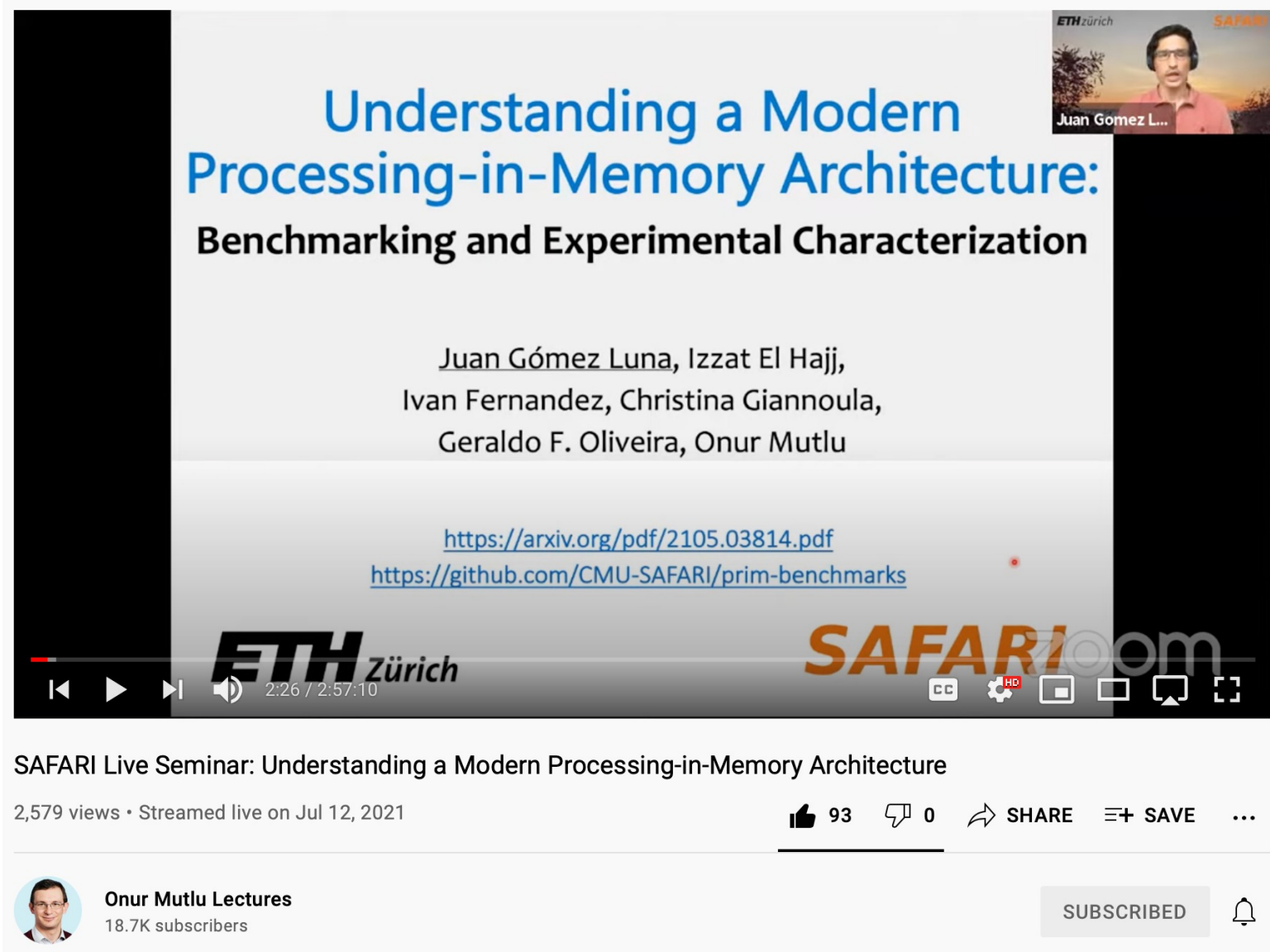
⁴National Technical University of Athens

Corresponding author: Juan Gómez-Luna (e-mail: juang@ethz.ch).

<https://arxiv.org/pdf/2105.03814.pdf>

<https://github.com/CMU-SAFARI/prim-benchmarks>

Understanding a Modern PIM Architecture



The image shows a YouTube video player interface. The video title is "Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization". The speaker is Juan Gómez Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu. The video is from the "SAFARI Live Seminar" series. The video player shows a progress bar at 2:26 / 2:57:10. The video is from the channel "Onur Mutlu Lectures" with 18.7K subscribers. The video has 2,579 views and was streamed live on Jul 12, 2021. The video player includes a video player with a title, speaker names, a link to the arXiv paper, a link to the GitHub repository, and a video player with a progress bar and controls. The video is from the channel "Onur Mutlu Lectures" with 18.7K subscribers. The video has 2,579 views and was streamed live on Jul 12, 2021.

Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez Luna, Izzat El Hajj,
Ivan Fernandez, Christina Giannoula,
Geraldo F. Oliveira, Onur Mutlu


<https://arxiv.org/pdf/2105.03814.pdf>
<https://github.com/CMU-SAFARI/prim-benchmarks>

ETH Zürich SAFARI

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

2,579 views • Streamed live on Jul 12, 2021

93 0 SHARE SAVE ...

 **Onur Mutlu Lectures**
18.7K subscribers

SUBSCRIBED

ML Training on a Real PIM System

Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna¹ Yuxin Guo¹ Sylvan Brocard² Julien Legriel²
Remy Cimadomo² Geraldo F. Oliveira¹ Gagandeep Singh¹ Onur Mutlu¹

¹ETH Zürich ²UPMEM

An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna¹ Yuxin Guo¹ Sylvan Brocard² Julien Legriel²
Remy Cimadomo² Geraldo F. Oliveira¹ Gagandeep Singh¹ Onur Mutlu¹

¹ETH Zürich ²UPMEM

Short version: <https://arxiv.org/pdf/2206.06022.pdf>

Long version: <https://arxiv.org/pdf/2207.07886.pdf>

<https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s>

ML Training on a Real PIM System

- Need to optimize data representation
 - (1) fixed-point
 - (2) quantization
 - (3) hybrid precision
- Use **lookup tables (LUTs)** to implement complex functions (e.g., sigmoid)
- Optimize data placement & layout for **streaming**
- Large speedups: **2.8X/27X vs. CPU, 1.3x/3.2x vs. GPU**

ML Training on Real PIM Talk Video

Comparison to CPU and GPU (III)

• Decision tree and K-means with Criteo 1TB dataset

(a) Decision Tree

Architecture	PIM Kernel	CPU-PIM	Inter PIM	PIM-CPU
DTR	~10,000	~10,000	~10,000	~10,000
CPU	~100,000	~100,000	~100,000	~100,000
GPU	~10,000	~10,000	~10,000	~10,000

(b) K-means

Architecture	GPU-CPU	CPU-GPU	GPU Kernel
KME	~10,000	~10,000	~10,000
CPU	~100,000	~100,000	~100,000
GPU	~10,000	~10,000	~10,000

PIM version of DTR is **62x** faster than the CPU version and **4.5x** faster than the GPU version

PIM version of KME is **2.7x** faster than the CPU version and **3.2x** faster than the GPU version

ETH zürich SAFARI
Juan Gomez Luna

Machine Learning Training on Memory-centric Computing Systems, Juan Gómez-Luna for ISPASS 2023



Onur Mutlu Lectures
32.9K subscribers

Analytics

Edit video

9

Share

Download

Clip

Save

...

242 views 11 days ago Livestream - Data-Centric Architectures: Fundamentally Improving Performance and Energy (Spring 2023)
Evaluating Machine Learning Workloads on Memory-centric Computing Systems

ML Training on Real PIM Systems

- Juan Gómez Luna, Yuxin Guo, Sylvan Brocard, Julien Legriel, Remy Cimadomo, Geraldo F. Oliveira, Gagandeep Singh, and Onur Mutlu,
"Evaluating Machine Learning Workloads on Memory-Centric Computing Systems"
Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, North Carolina, USA, April 2023.
[[arXiv version](#), 16 July 2022.]
[[PIM-ML Source Code](#)]
Best paper session.

An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna¹ Yuxin Guo¹ Sylvan Brocard² Julien Legriel²
Remy Cimadomo² Geraldo F. Oliveira¹ Gagandeep Singh¹ Onur Mutlu¹
¹ETH Zürich ²UPMEM

<https://github.com/CMU-SAFARI/pim-ml>

SpMV Multiplication on Real PIM Systems

- Appears at SIGMETRICS 2022

***SparseP*: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems**

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece

IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

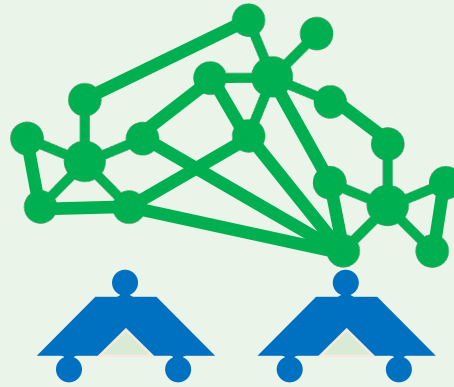
NECTARIOS KOZIRIS, National Technical University of Athens, Greece

GEORGIOS GOUMAS, National Technical University of Athens, Greece

ONUR MUTLU, ETH Zürich, Switzerland

<https://arxiv.org/pdf/2201.05072.pdf>

<https://github.com/CMU-SAFARI/SparseP>



SparseP

Towards Efficient Sparse Matrix Vector Multiplication
on Real Processing-In-Memory Architectures

Christina Giannoula

Ivan Fernandez, Juan Gomez-Luna,

Nectarios Koziris, Georgios Goumas, Onur Mutlu

SAFARI **ETH** zürich



UNIVERSIDAD
DE MÁLAGA

SparseP: Key Contributions

1. Efficient SpMV kernels for current & future PIM systems

- SparseP library = 25 SpMV kernels
 - Compression, data types, data partitioning, synchronization, load balancing

SparseP is Open-Source

SparseP: <https://github.com/CMU-SAFARI/SparseP>

2. Comprehensive analysis of SpMV on the first commercially-available real PIM system



- 26 sparse matrices
- Comparisons to state-of-the-art CPU and GPU systems
- Recommendations for software, system and hardware designers

Recommendations for Architects and Programmers

Full Paper: <https://arxiv.org/pdf/2201.05072.pdf>

SparseP Talk Video



The screenshot shows a YouTube video player with a presentation slide. The slide features the SparseP logo, which consists of a green network graph above two blue stylized human figures. Below the logo, the title "SparseP" is written in large blue letters, followed by the subtitle "Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures" in smaller blue letters. The presenter's name, "Christina Giannoula", is listed in red, followed by the names of other contributors: "Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, Onur Mutlu". At the bottom of the slide, there are logos for SAFARI, ETH zürich, CSLab, and several university seals. A small video inset in the top right corner shows the presenter, Christina Giannoula, with her name "Christina Gian..." visible below it. The video player interface includes a progress bar at 0:02 / 55:25, various control icons, and a list of logos at the bottom.

Processing-in-Memory Course: Lecture 11: SpMV on a Real PIM Architecture - Spring 2022

149 views • Streamed live on May 19, 2022

👍 12 🗑 DISLIKE ➦ SHARE ⬇ DOWNLOAD ✂ CLIP ≡+ SAVE ...



Onur Mutlu Lectures
25K subscribers

ANALYTICS

EDIT VIDEO

More on SparseP

Christina Giannoula, Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, and Onur Mutlu,

"SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures"

*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Mumbai, India, June 2022.*

[\[Extended arXiv Version\]](#)

[\[Abstract\]](#)

[\[Slides \(pptx\) \(pdf\)\]](#)

[\[Long Talk Slides \(pptx\) \(pdf\)\]](#)

[\[SparseP Source Code\]](#)

[\[Talk Video \(16 minutes\)\]](#)

[\[Long Talk Video \(55 minutes\)\]](#)

SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece

IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

NECTARIOS KOZIRIS, National Technical University of Athens, Greece

GEORGIOS GOUMAS, National Technical University of Athens, Greece

ONUR MUTLU, ETH Zürich, Switzerland

[**https://github.com/CMU-SAFARI/SparseP**](https://github.com/CMU-SAFARI/SparseP)

Transcendental Functions on Real PIM Systems

- Maurus Item, Juan Gómez Luna, Yuxin Guo, Geraldo F. Oliveira, Mohammad Sadrosadati, and Onur Mutlu,
"TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems"
Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, North Carolina, USA, April 2023.
[[arXiv version](#)]
[[Slides \(pptx\)](#)] [[pdf](#)]
[[TransPimLib Source Code](#)]
[[Talk Video](#) (17 minutes)]

TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems

Maurus Item
Geraldo F. Oliveira

Juan Gómez-Luna
Mohammad Sadrosadati

Yuxin Guo
Onur Mutlu

ETH Zürich

<https://github.com/CMU-SAFARI/transpimlib>

Sequence Alignment on Real PIM Systems

- Safaa Diab, Amir Nassereldine, Mohammed Alser, Juan Gómez Luna, Onur Mutlu, and Izzat El Hajj,
"A Framework for High-throughput Sequence Alignment using Real Processing-in-Memory Systems"
Bioinformatics, [published online on] 27 March 2023.
[[Online link at Bioinformatics Journal](#)]
[[arXiv preprint](#)]
[[AiM Source Code](#)]

A Framework for High-throughput Sequence Alignment using Real Processing-in-Memory Systems

Safaa Diab¹ Amir Nassereldine¹ Mohammed Alser² Juan Gómez Luna²
Onur Mutlu² Izzat El Hajj¹

¹American University of Beirut ²ETH Zürich

<https://github.com/CMU-SAFARI/alignment-in-memory>



Summary

- Sequence alignment on traditional systems is limited by the **memory bandwidth bottleneck**
- **Processing-in-memory (PIM)** overcomes this bottleneck by placing cores near the memory
- Our framework, **Alignment-in-Memory (AIM)**, is a PIM framework that supports multiple alignment algorithms (NW, SWG, GenASM, WFA)
 - Implemented on UPMEM, the first real PIM system
- Results show **substantial speedups over both CPUs (1.8X-28X) and GPUs (1.2X-2.7X)**
- AIM is available at:
 - <https://github.com/CMU-SAFARI/alignment-in-memory>

Homomorphic Operations on Real PIM Systems

- Harshita Gupta, Mayank Kabra, Juan Gómez-Luna, Konstantinos Kanellopoulos, and Onur Mutlu,
"Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System"
Proceedings of the 2023 IEEE International Symposium on Workload Characterization Poster Session (IISWC), Ghent, Belgium, October 2023.
[[arXiv version](#)]
[[Lightning Talk Slides \(pptx\)](#) ([pdf](#))]
[[Poster \(pptx\)](#) ([pdf](#))]

Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System

Harshita Gupta* Mayank Kabra* Juan Gómez-Luna Konstantinos Kanellopoulos Onur Mutlu
ETH Zürich

Accelerating ML Training on Real PIM Systems

- <https://arxiv.org/pdf/2404.07164>

Analysis of Distributed Optimization Algorithms on a Real Processing-In-Memory System

Steve Rhyner¹ Haocong Luo¹ Juan Gómez-Luna² Mohammad Sadrosadati¹
Jiawei Jiang³ Ataberk Olgun¹ Harshita Gupta¹ Ce Zhang⁴ Onur Mutlu¹
¹ETH Zurich ²NVIDIA ³Wuhan University ⁴University of Chicago

Accelerating ML Training on Real PIM Systems

- <https://arxiv.org/pdf/2404.07164>

8. Conclusion

We evaluate and train ML models on large-scale datasets with centralized parallel optimization algorithms on a *real-world* PIM architecture. We show the importance of carefully *choosing* the distributed optimization algorithm that fits PIM and analyze tradeoffs. We demonstrate that *commercial* general-purpose PIM systems can be a viable alternative for many ML training workloads on large-scale datasets to processor-centric architectures. Our results demonstrate the necessity of adapting PIM architectures to enable inter-DPU communication to overcome scalability challenges for many ML training workloads and discuss decentralized parallel SGD optimization algorithms as a potential solution.

Accelerating GNNs on Real PIM Systems

- <https://arxiv.org/pdf/2402.16731>

Accelerating Graph Neural Networks on Real Processing-In-Memory Systems

Christina Giannoula^{*†}, Peiming Yang^{*}, Ivan Fernandez Vega^{§†}, Jiacheng Yang^{*}, Yu Xin Li^{*},
Juan Gomez Luna^{¶†}, Mohammad Sadrosadati[†], Onur Mutlu^{†‡}, Gennady Pekhimenko^{*||}

^{*}University of Toronto [†]ETH Zürich [§]Barcelona Supercomputing Center

[¶]NVIDIA [‡]Stanford ^{||}CentML

Accelerating GNNs on Real PIM Systems

- <https://arxiv.org/pdf/2402.16731>

Abstract—Graph Neural Networks (GNNs) are emerging ML models to analyze graph-structure data. Graph Neural Network (GNN) execution involves both compute-intensive and memory-intensive kernels, the latter dominates the total time, being significantly bottlenecked by data movement between memory and processors. Processing-In-Memory (PIM) systems can alleviate this data movement bottleneck by placing simple processors near or inside to memory arrays. In this work, we introduce PyGim, an efficient ML framework that accelerates GNNs on real PIM systems. We propose intelligent parallelization techniques for memory-intensive kernels of GNNs tailored for real PIM systems, and develop handy Python API for them. We provide hybrid GNN execution, in which the compute-intensive and memory-intensive kernels are executed in processor-centric and memory-centric computing systems, respectively, to match their algorithmic nature. We extensively evaluate PyGim on a real-world PIM system with 1992 PIM cores using emerging GNN models, and demonstrate that it outperforms its state-of-the-art CPU counterpart on Intel Xeon by on average $3.04\times$, and achieves higher resource utilization than CPU and GPU systems. Our work provides useful recommendations for software, system and hardware designers. PyGim will be open-sourced to enable the widespread use of PIM systems in GNNs.

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



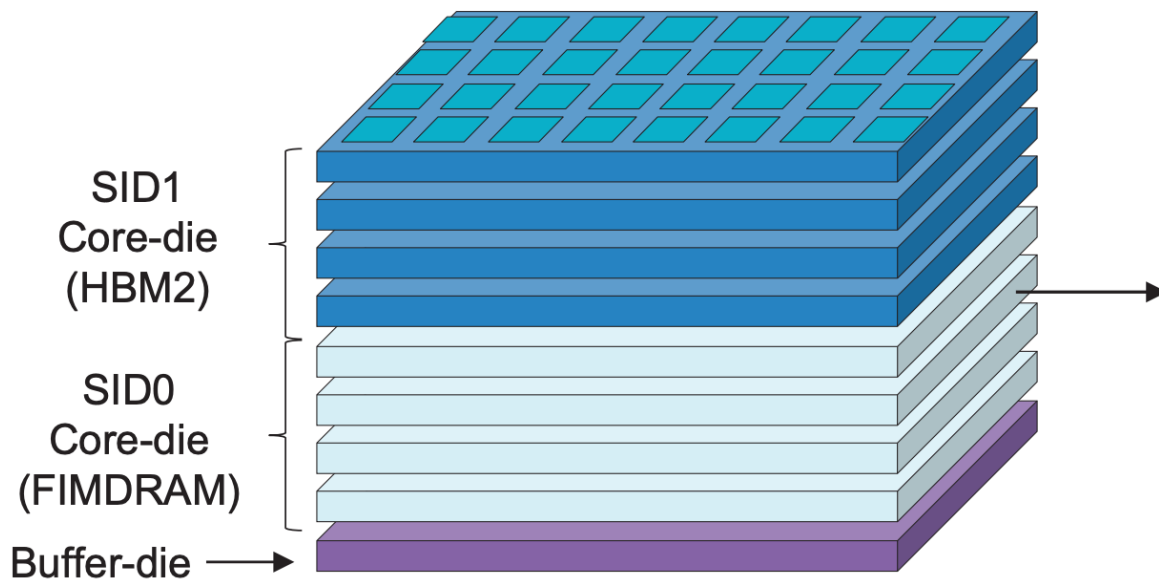
The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

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."

Samsung Function-in-Memory DRAM (2021)

■ FIMDRAM based on HBM2



[3D Chip Structure of HBM with FIMDRAM]

Chip Specification

128DQ / 8CH / 16 banks / BL4

32 PCU blocks (1 FIM block/2 banks)

1.2 TFLOPS (4H)

**FP16 ADD /
Multiply (MUL) /
Multiply-Accumulate (MAC) /
Multiply-and- Add (MAD)**

ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 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

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyun-Sung Shin¹, Jin Kim¹, BengSeng Phuah¹, HyoungMin Kim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang², Shinhaeng Kang¹, Yuhwan Ro³, Seungwoo Seo³, JoonHo Song³, Jaeyoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwaseong, Korea

²Samsung Electronics, San Jose, CA

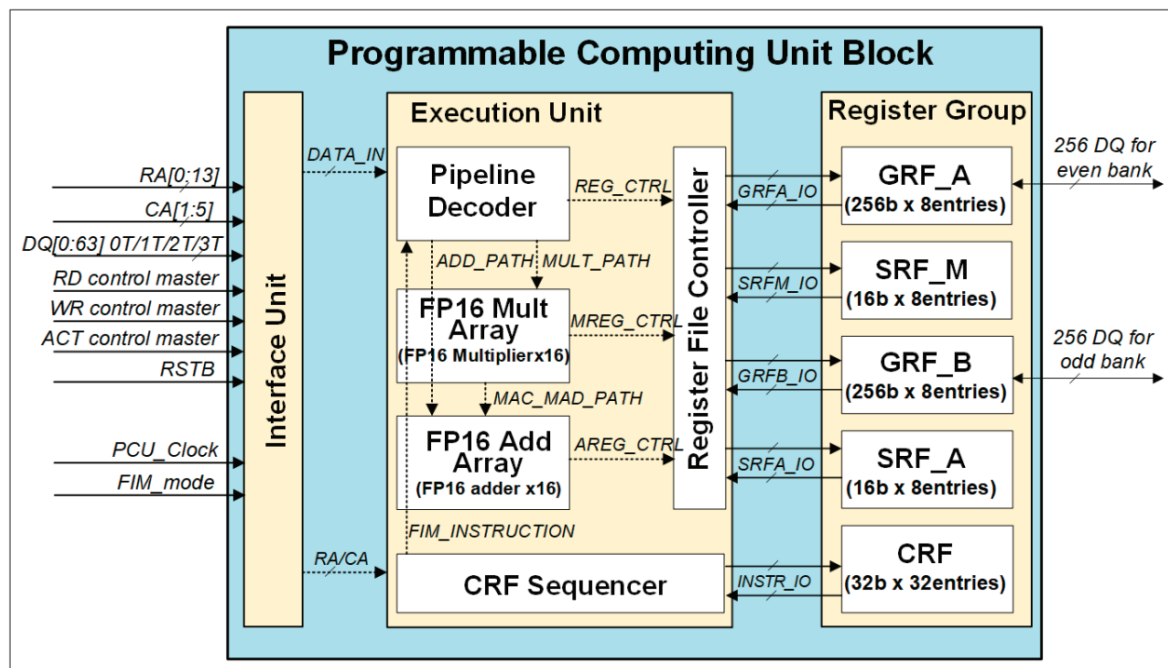
³Samsung Electronics, Suwon, Korea

Samsung Function-in-Memory DRAM (2021)

Programmable Computing Unit

■ Configuration of PCU block

- Interface unit to control data flow
- Execution unit to perform operations
- Register group
 - 32 entries of CRF for instruction memory
 - 16 GRF for weight and accumulation
 - 16 SRF to store constants for MAC operations



[Block diagram of PCU in FIMDRAM]

ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 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

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyun-Sung Shin¹, Jin Kim¹, BengSeng Phuah¹, HyungMin Kim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang², Shinhaeng Kang³, Yuhwan Ro³, Seungwoo Seo³, JoonHo Song³, Jaeyoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwaseong, Korea

²Samsung Electronics, San Jose, CA

³Samsung Electronics, Suwon, Korea

Samsung Function-in-Memory DRAM (2021)

[Available instruction list for FIM operation]

Type	CMD	Description
Floating Point	ADD	FP16 addition
	MUL	FP16 multiplication
	MAC	FP16 multiply-accumulate
	MAD	FP16 multiply and add
Data Path	MOVE	Load or store data
	FILL	Copy data from bank to GRFs
Control Path	NOP	Do nothing
	JUMP	Jump instruction
	EXIT	Exit instruction

ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 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

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyun-Sung Shin¹, Jin Kim¹, BengSeng Phuah¹, HyungMin Kim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang², Shinhaeng Kang¹, Yuhwan Ro¹, Seungwoo Seo¹, JoonHo Song¹, Jaeyoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwaseong, Korea

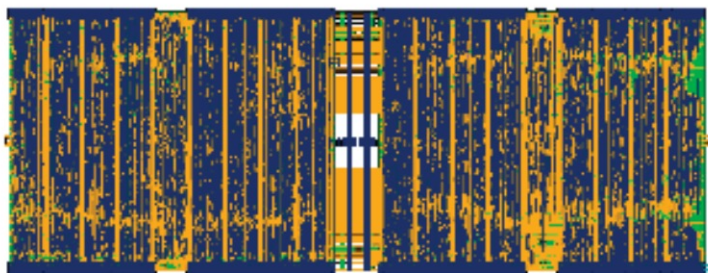
²Samsung Electronics, San Jose, CA

³Samsung Electronics, Suwon, Korea

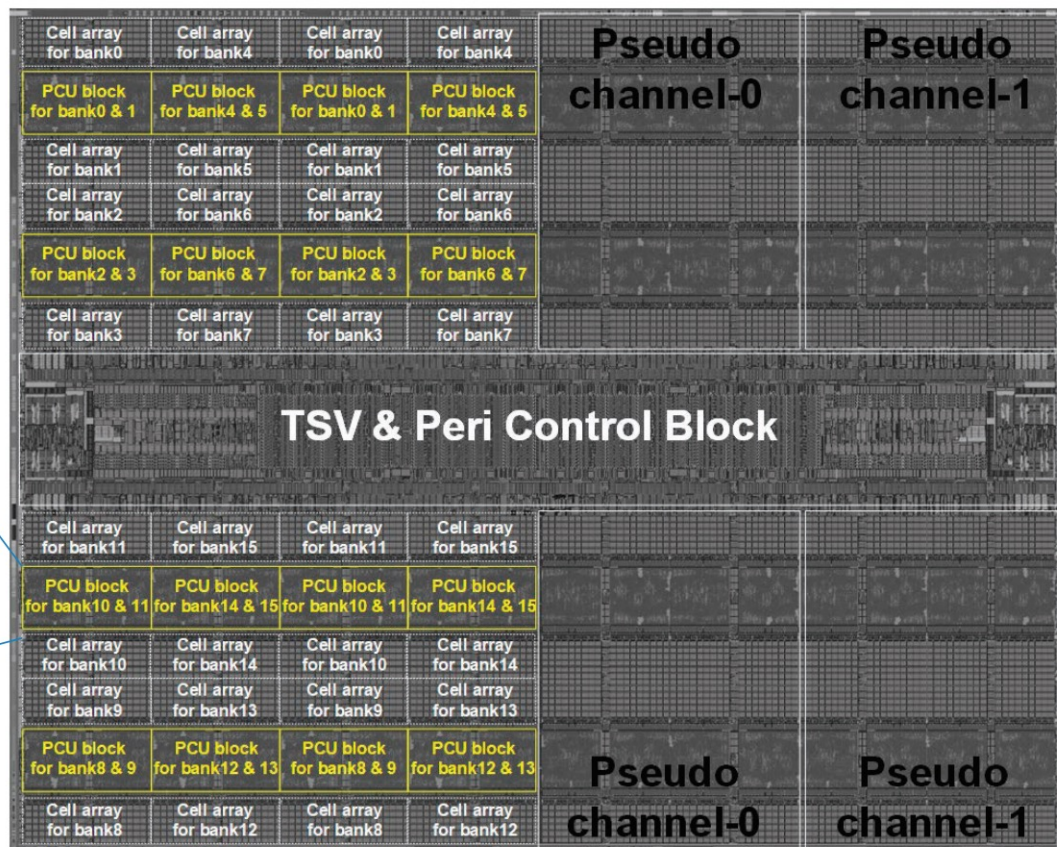
Samsung Function-in-Memory DRAM (2021)

Chip Implementation

- Mixed design methodology to implement FIMDRAM
 - Full-custom + Digital RTL



[Digital RTL design for PCU block]



ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 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

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyeon Choi¹, Hyun-Sung Shim¹, Jin Kim¹, BengSeng Phuah¹, HyounMin Kim¹, Myeong Jun Song¹, Ahn Chai¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang², Shinfraeng Kang³, Yulwan Ro³, Seungwoo Seo³, JoonHo Song³, Jaeyoun Yoon¹, Kyomin Sohn¹, Nam Sung Kim¹

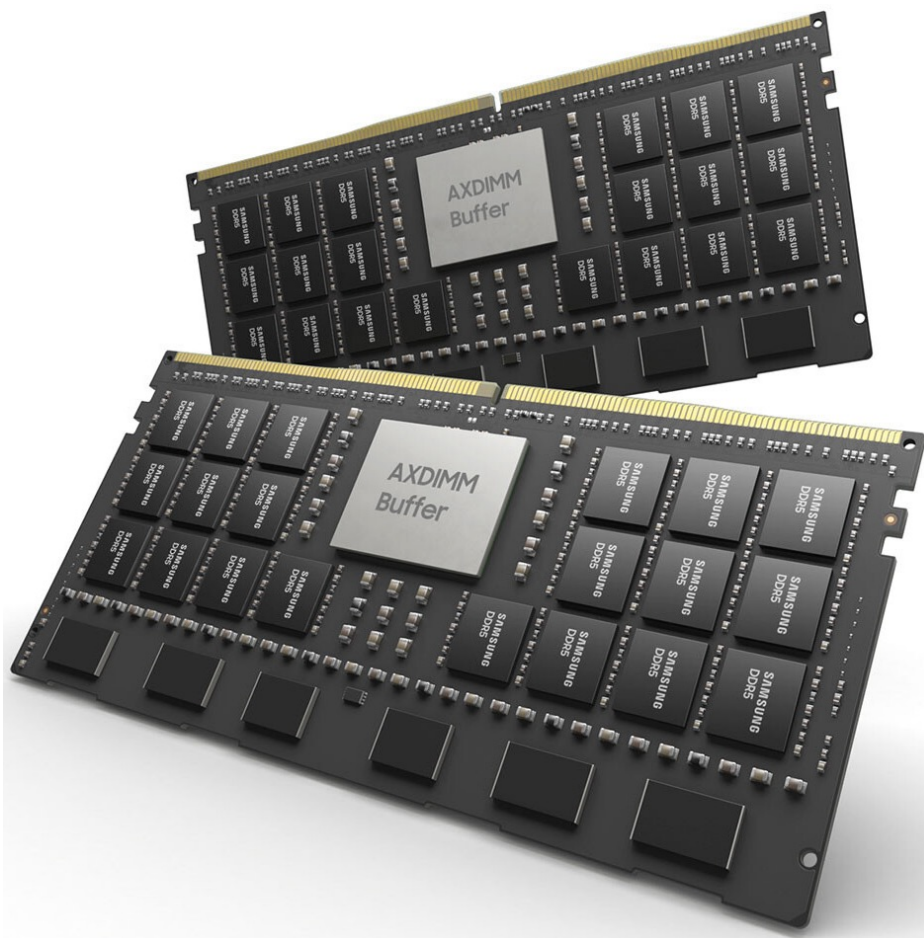
¹Samsung Electronics, Hwaseong, Korea

²Samsung Electronics, San Jose, CA

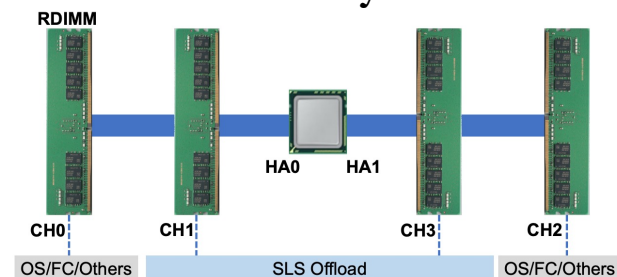
³Samsung Electronics, Suwon, Korea

Samsung AxDIMM (2021)

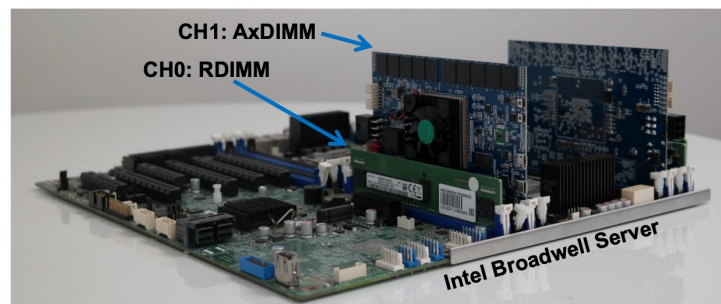
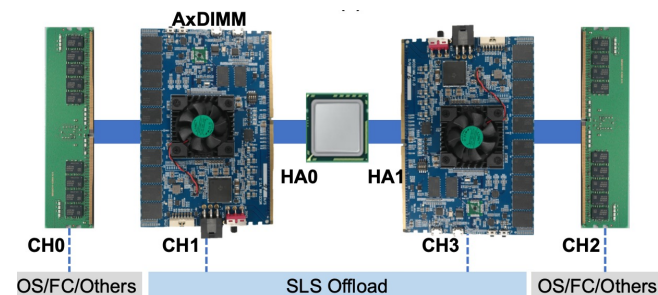
- DDRx-PIM
 - DLRM recommendation system



Baseline System



AxDIMM System



SK Hynix Accelerator-in-Memory (2022)

SK hynix Develops PIM, Next-Generation AI Accelerator

February 16, 2022



Seoul, February 16, 2022

SK hynix (or “the Company”, www.skhynix.com) announced on February 16 that it has developed PIM*, a next-generation memory chip with computing capabilities.

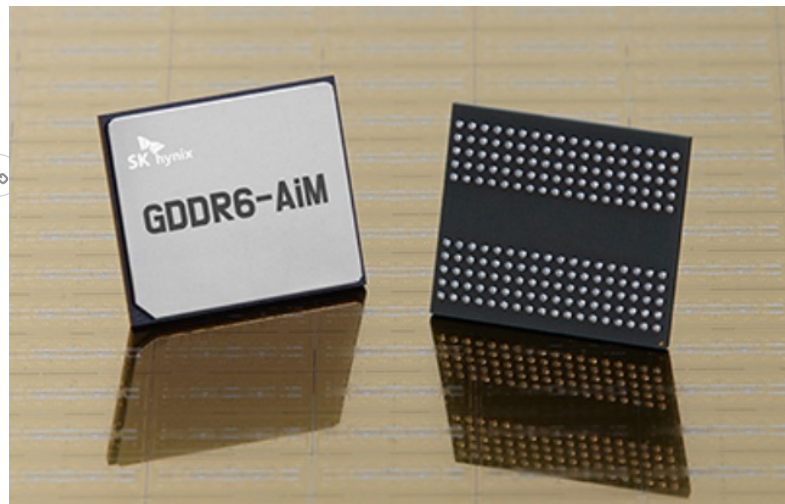
**PIM(Processing In Memory): A next-generation technology that provides a solution for data congestion issues for AI and big data by adding computational functions to semiconductor memory*

It has been generally accepted that memory chips store data and CPU or GPU, like human brain, process data. SK hynix, following its challenge to such notion and efforts to pursue innovation in the next-generation smart memory, has found a breakthrough solution with the development of the latest technology.

SK hynix plans to showcase its PIM development at the world’s most prestigious semiconductor conference, 2022 ISSCC*, in San Francisco at the end of this month. The company expects continued efforts for innovation of this technology to bring the memory-centric computing, in which semiconductor memory plays a central role, a step closer to the reality in devices such as smartphones.

**ISSCC: The International Solid-State Circuits Conference will be held virtually from Feb. 20 to Feb. 24 this year with a theme of “Intelligent Silicon for a Sustainable World”*

For the first product that adopts the PIM technology, SK hynix has developed a sample of GDDR6-AiM (Accelerator* in memory). The GDDR6-AiM adds computational functions to GDDR6* memory chips, which process data at 16Gbps. A combination of GDDR6-AiM with CPU or GPU instead of a typical DRAM makes certain computation speed 16 times faster. GDDR6-AiM is widely expected to be adopted for machine learning, high-performance computing, and big data computation and storage.



11.1 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

Seongju Lee, SK hynix, Icheon, Korea

In Paper 11.1, SK Hynix describes an 1ynm, GDDR6-based accelerator-in-memory with a command set for deep-learning operation. The 8Gb design achieves a peak throughput of 1TFLOPS with 1GHz MAC operations and supports major activation functions to improve accuracy.

SK Hynix Accelerator-in-Memory (2022)

System Architecture and Software Stack for GDDR6-AiM

Yongkee Kwon and Chanwook Park
SK hynix inc.

5:42 / 6:27:38

SK hynix

ASPLOS 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads



Onur Mutlu Lectures
32.1K subscribers

Analytics

Edit video

33



Share

Download

Clip

Save



1,146 views Streamed live on Mar 26, 2023 Livestream - Data-Centric Architectures: Fundamentally Improving Performance and Energy (Spring 2023)

ASPLOS 2023 Tutorial: Real-world Processing-in-Memory Systems for Modern Workloads

<https://events.safari.ethz.ch/asplos-...>

<https://www.youtube.com/watch?v=oYCaLcT0Kmo>

AliBaba PIM Recommendation System (2022)

ISSCC 2022 / February 24, 2022 / 8:30 AM

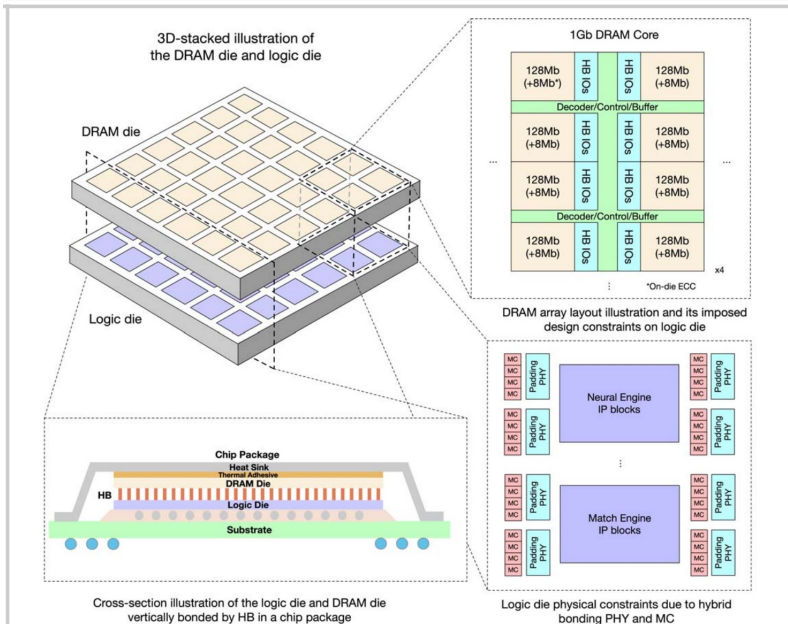


Figure 29.1.2: Illustration of 3D-stacked chip, cross-illustration of package, DRAM array layout and design blocks on logic die.

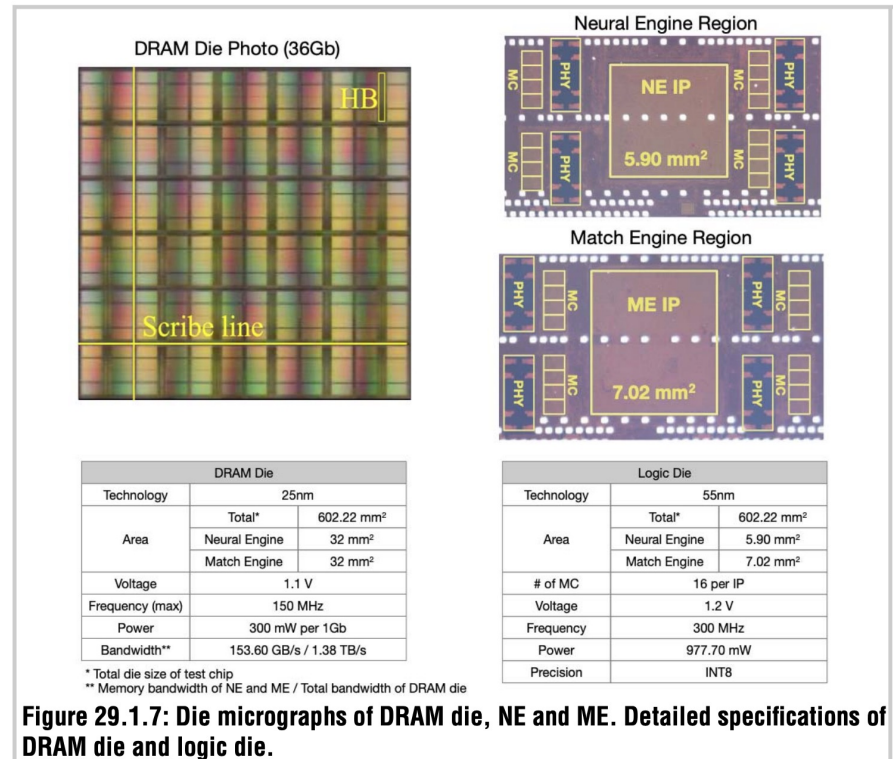


Figure 29.1.7: Die micrographs of DRAM die, NE and ME. Detailed specifications of DRAM die and logic die.

29.1 184QPS/W 64Mb/mm² 3D Logic-to-DRAM Hybrid Bonding with Process-Near-Memory Engine for Recommendation System

Dimin Niu¹, Shuangchen Li¹, Yuhao Wang¹, Wei Han¹, Zhe Zhang², Yijin Guan², Tianchan Guan³, Fei Sun¹, Fei Xue¹, Lide Duan¹, Yuanwei Fang¹, Hongzhong Zheng¹, Xiping Jiang⁴, Song Wang⁴, Fengguo Zuo⁴, Yubing Wang⁴, Bing Yu⁴, Qiwei Ren⁴, Yuan Xie¹

SK Hynix CXL Processing Near Memory (2023)

IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 22, NO. 1, JANUARY-JUNE

Computational CXL-Memory Solution for Accelerating Memory-Intensive Applications

Joonseop Sim^{ID}, Soohong Ahn^{ID}, Taeyoung Ahn^{ID},
Seungyong Lee^{ID}, Myunghyun Rhee, Jooyoung Kim^{ID},
Kwangsik Shin, Donguk Moon^{ID},
Euseok Kim, and Kyoung Park^{ID}

Abstract—CXL interface is the up-to-date technology that enables effective memory expansion by providing a memory-sharing protocol in configuring heterogeneous devices. However, its limited physical bandwidth can be a significant bottleneck for emerging data-intensive applications. In this work, we propose a novel CXL-based memory disaggregation architecture with a real-world prototype demonstration, which overcomes the bandwidth limitation of the CXL interface using near-data processing. The experimental results demonstrate that our design achieves up to $1.9\times$ better performance/power efficiency than the existing CPU system.

Index Terms—Compute express link (CXL), near-data-processing (NDP)

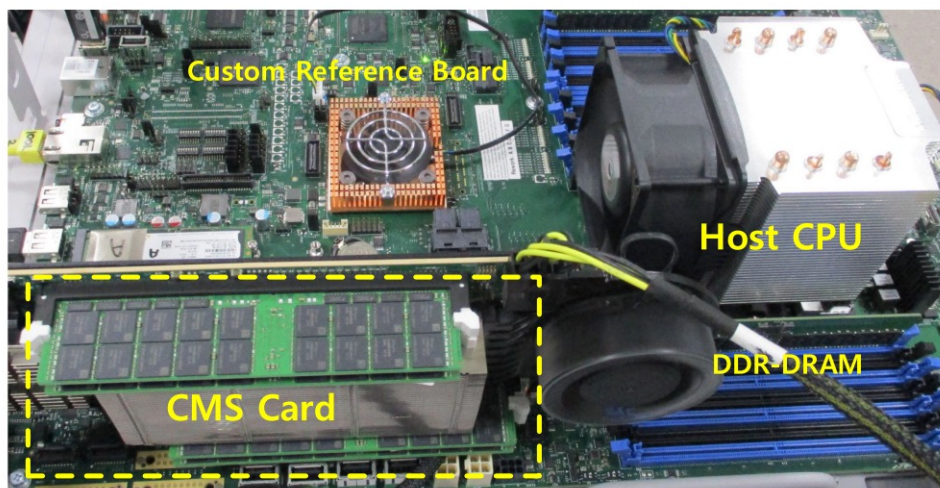


Fig. 6. FPGA prototype of proposed CMS card.

Samsung CXL Processing Near Memory (2023)

Samsung Processing in Memory Technology at Hot Chips 2023

By Patrick Kennedy - August 28, 2023



Samsung PIM PNM For Transformer Based AI HC35_Page_24

Concluding Remarks

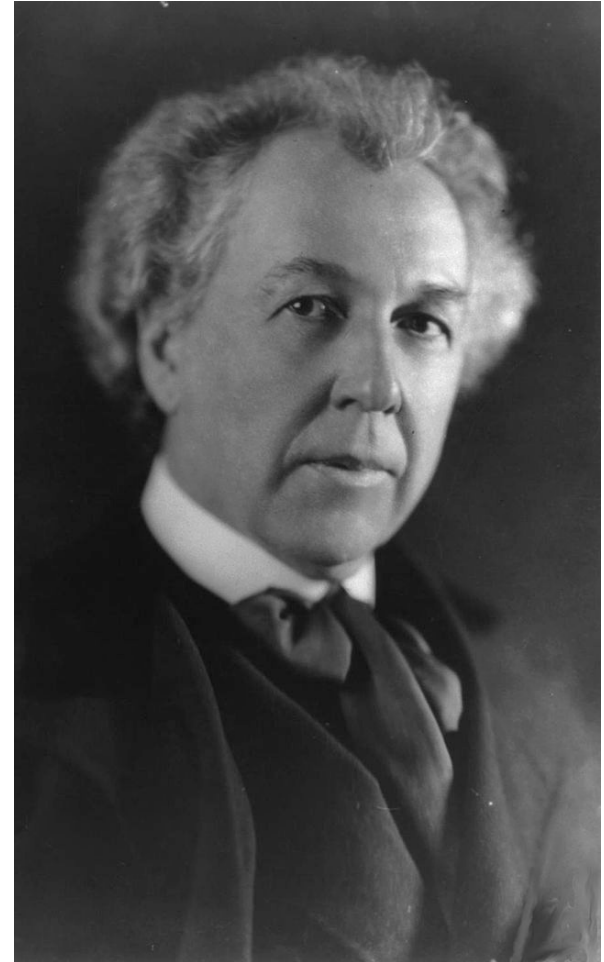
Fundamentally Energy-Efficient **(Data-Centric)** Computing Architectures

Fundamentally High-Performance **(Data-Centric)** Computing Architectures

Computing Architectures with Minimal Data Movement

A Quote from A Famous Architect

- “architecture [...] based upon **principle**, and not upon **precedent**”



Precedent-Based Design?

- “architecture [...] based upon **principle**, and not upon **precedent**”



Principled Design

- “architecture [...] based upon **principle**, and not upon **precedent**”





The Overarching Principle

Organic architecture

From Wikipedia, the free encyclopedia

Organic architecture is a [philosophy](#) of [architecture](#) which promotes harmony between human habitation and the natural world through design approaches so sympathetic and well integrated with its site, that buildings, furnishings, and surroundings become part of a unified, interrelated composition.

A well-known example of organic architecture is [Fallingwater](#), the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy dialog with the rushing water and the steep site. The horizontal striations of stone masonry with daring [cantilevers](#) of colored beige concrete blend with native rock outcroppings and the wooded environment.

Another Example: Precedent-Based Design



Principled Design



Another Principled Design



Source: By Martín Gómez Tagle - Lisbon, Portugal, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=13764903>

Source: <http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/>

Another Principled Design



Principle Applied to Another Structure



Source: By 準建築人手札網站 Forgemind ArchiMedia - Flickr: IMG_2489.JPG, CC BY 2.0

Source: <https://www.dezeen.com/2016/08/29/santiago-calatrava-oculus-world-trade-center-transportation-hub-new-york-photographs-hufton-crow/>

The Overarching Principle

Zoomorphic architecture

From Wikipedia, the free encyclopedia

Zoomorphic architecture is the practice of using animal forms as the inspirational basis and blueprint for architectural design. "While animal forms have always played a role adding some of the deepest layers of meaning in architecture, it is now becoming evident that a new strand of **biomorphism** is emerging where the meaning derives not from any specific representation but from a more general allusion to biological processes."^[1]

Some well-known examples of Zoomorphic architecture can be found in the **TWA Flight Center** building in **New York City**, by **Eero Saarinen**, or the **Milwaukee Art Museum** by **Santiago Calatrava**, both inspired by the form of a bird's wings.^[3]

Overarching Principles for Computing?



Concluding Remarks

- It is time to design **principled system architectures** to solve the **memory problem**
- We must design systems to be **balanced, high-performance, and energy-efficient** → **memory-centric**
 - Enable computation capabilities in memory
- This can
 - Lead to **orders-of-magnitude** improvements
 - **Enable new applications & computing platforms**
 - **Enable better understanding of nature**
 - ...
- Future of **truly memory-centric computing** is bright
 - We need to do research & design across the computing stack

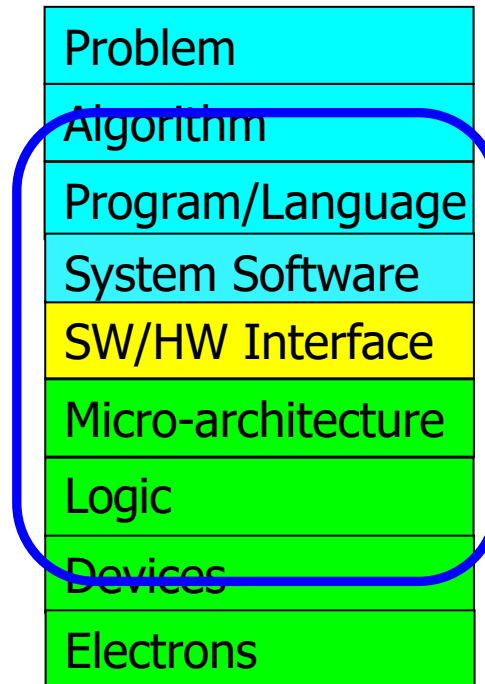
Data-centric

Data-driven

Data-aware

We Need to Revisit the Entire Stack

- With a **memory-centric mindset**



We can get there step by step

PIM Review and Open Problems

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

^d*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

"A Modern Primer on Processing in Memory"

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

Open Source Tools: SAFARI GitHub



SAFARI Research Group at ETH Zurich and Carnegie Mellon University

Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

👤 440 followers 📍 ETH Zurich and Carnegie Mellon U... 🔗 <https://safari.ethz.ch/> ✉ omutlu@gmail.com

🏠 Overview 📁 Repositories 98 📁 Projects 📦 Packages 👤 People 13

📁 ramulator Public

A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the...

● C++ ☆ 532 🍴 206

📁 prim-benchmarks Public

PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publ...

● C ☆ 126 🍴 47

📁 MQSim Public

MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implement...

● C++ ☆ 268 🍴 143

📁 rowhammer Public

Source code for testing the Row Hammer error mechanism in DRAM devices. Described in the ISCA 2014 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_isca14.pdf.

● C ☆ 211 🍴 42

📁 SoftMC Public

SoftMC is an experimental FPGA-based memory controller design that can be used to develop tests for DDR3 SODIMMs using a C++ based API. The design, the interface, and its capabilities and limitatio...

● Verilog ☆ 120 🍴 27

📁 Pythia Public

A customizable hardware prefetching framework using online reinforcement learning as described in the MICRO 2021 paper by Bera et al. (<https://arxiv.org/pdf/2109.12021.pdf>).

● C++ ☆ 109 🍴 34

<https://github.com/CMU-SAFARI/>

Referenced Papers, Talks, Artifacts

- All are available at

<https://people.inf.ethz.ch/omutlu/projects.htm>

<https://www.youtube.com/onurmutlulectures>

<https://github.com/CMU-SAFARI/>

Funding Acknowledgments

- Alibaba, AMD, ASML, Google, Facebook, Hi-Silicon, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware, Xilinx
- NSF
- NIH
- GSRC
- SRC
- CyLab
- EFCL
- SNSF

Thank you!