Security of PIM Systems

Onur Mutlu

omutlu@gmail.com

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

30 November 2023

Dagstuhl MAD (Microarchitectural Attacks & Defenses)

SAFARI   **ETH**zürich

Problem
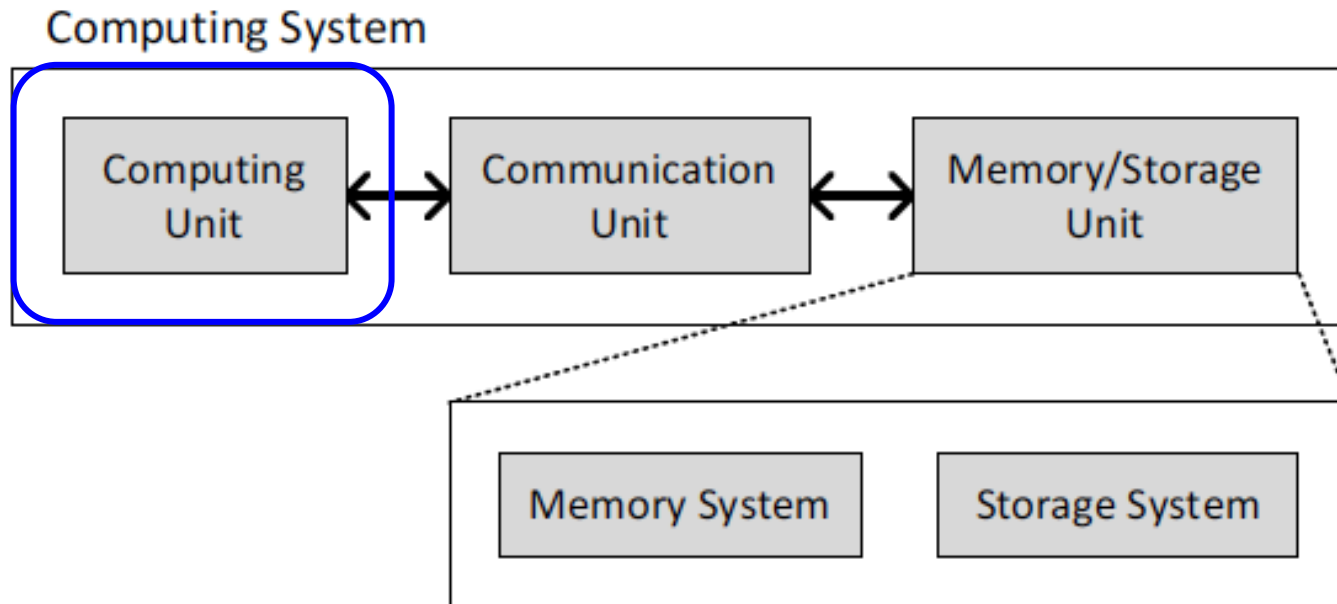
Computing

is Bottlenecked by Data

Today's Computing Systems

- Processor centric

- All data processed in the processor → at great system cost

Computing System

It's the Memory, Stupid!

- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)
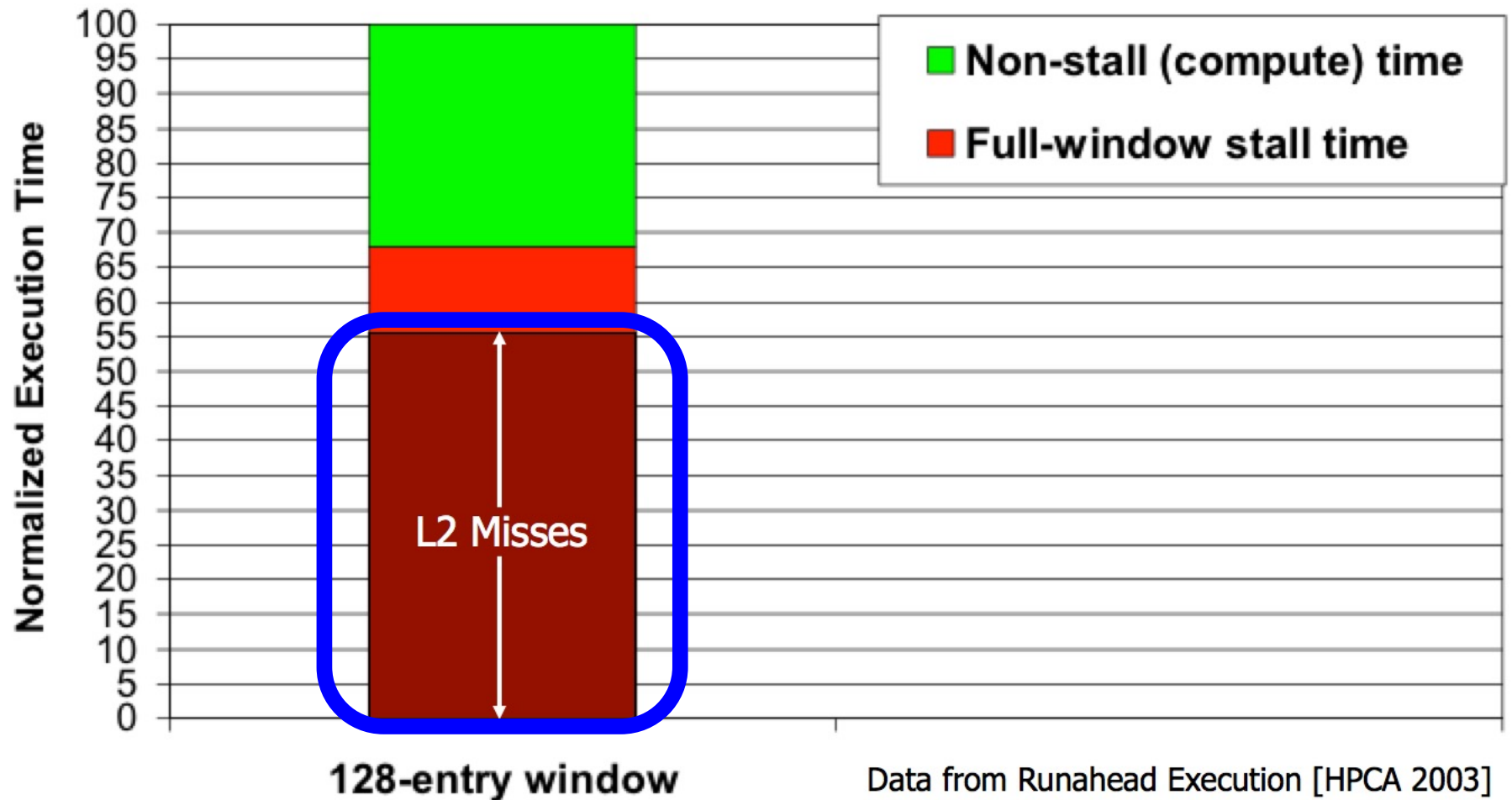
RICHARD SITES

It's the Memory, Stupid!

When we started the Alpha architecture design in 1988, we estimated a 25-year lifetime and a relatively modest 32% per year compounded performance improvement of implementations over that lifetime (1,000× total). We guestimated about 10× would come from CPU clock improvement, 10× from multiple instruction issue, and 10× from multiple processors.
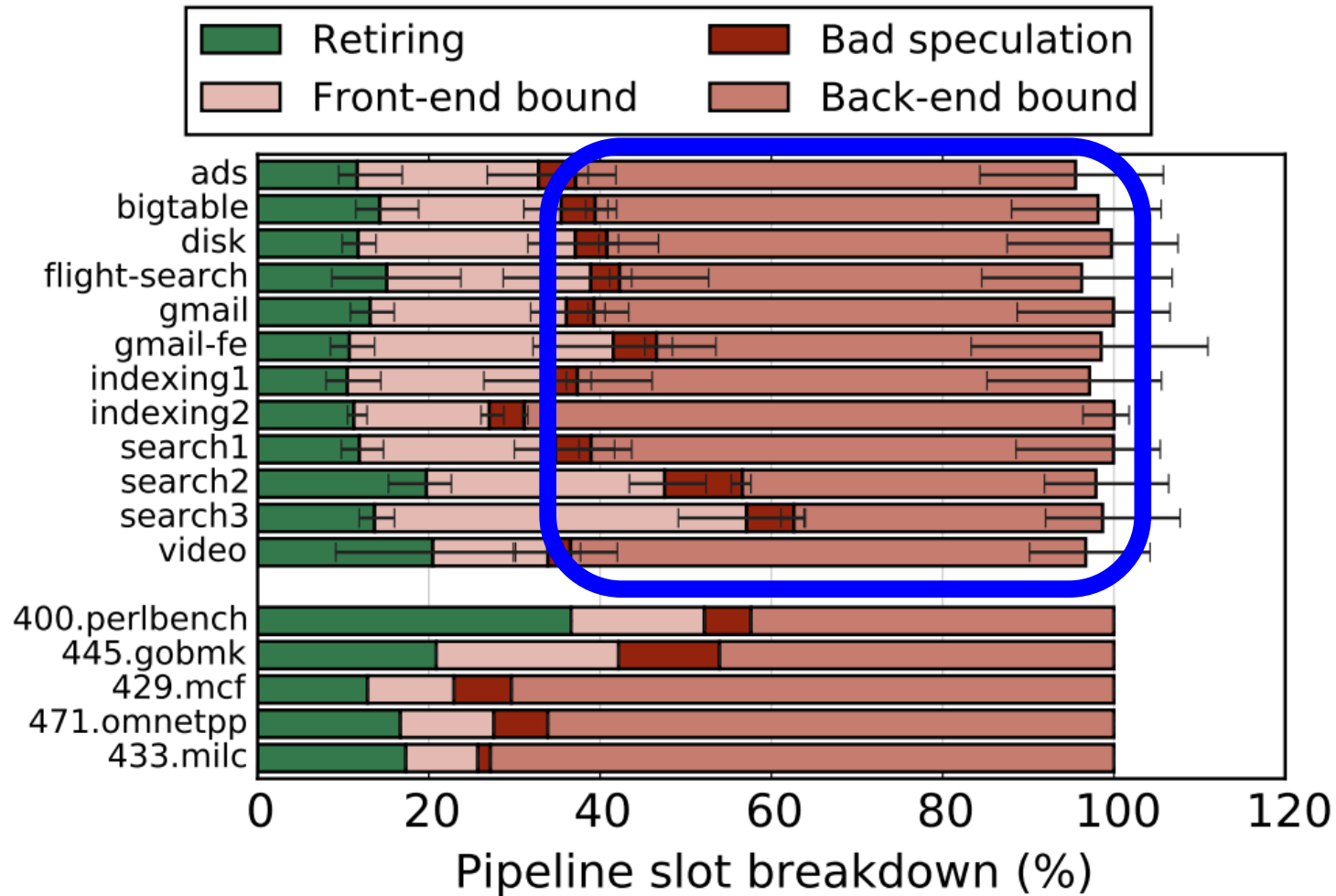
5, 1996 ◇ MICROPROCESSOR REPORT

I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.
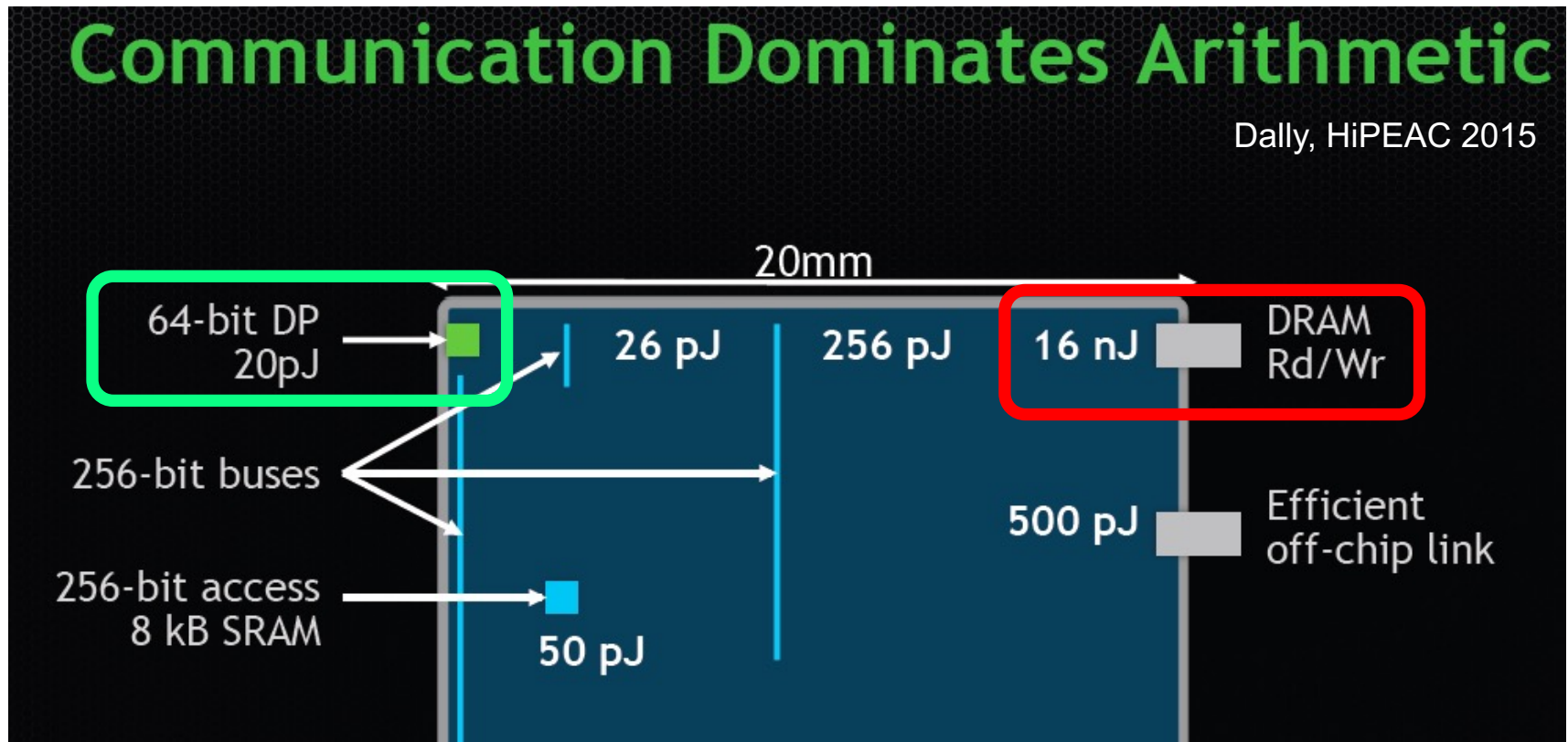
Processor-Centric System Performance

128-entry window

Data from Runahead Execution [HPCA 2003]

Mutlu+, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors," HPCA 2003.

Processor-Centric System Performance

- All of Google's Data Center Workloads (2015):

Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

Data Movement vs. Computation Energy

Communication Dominates Arithmetic

Dally, HiPEAC 2015

- 64-bit DP 20pJ
- 256-bit buses
- 256-bit access 8 kB SRAM

20mm

26 pJ   256 pJ   16 nJ — DRAM Rd/Wr

500 pJ — Efficient off-chip link

50 pJ

A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy

Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.
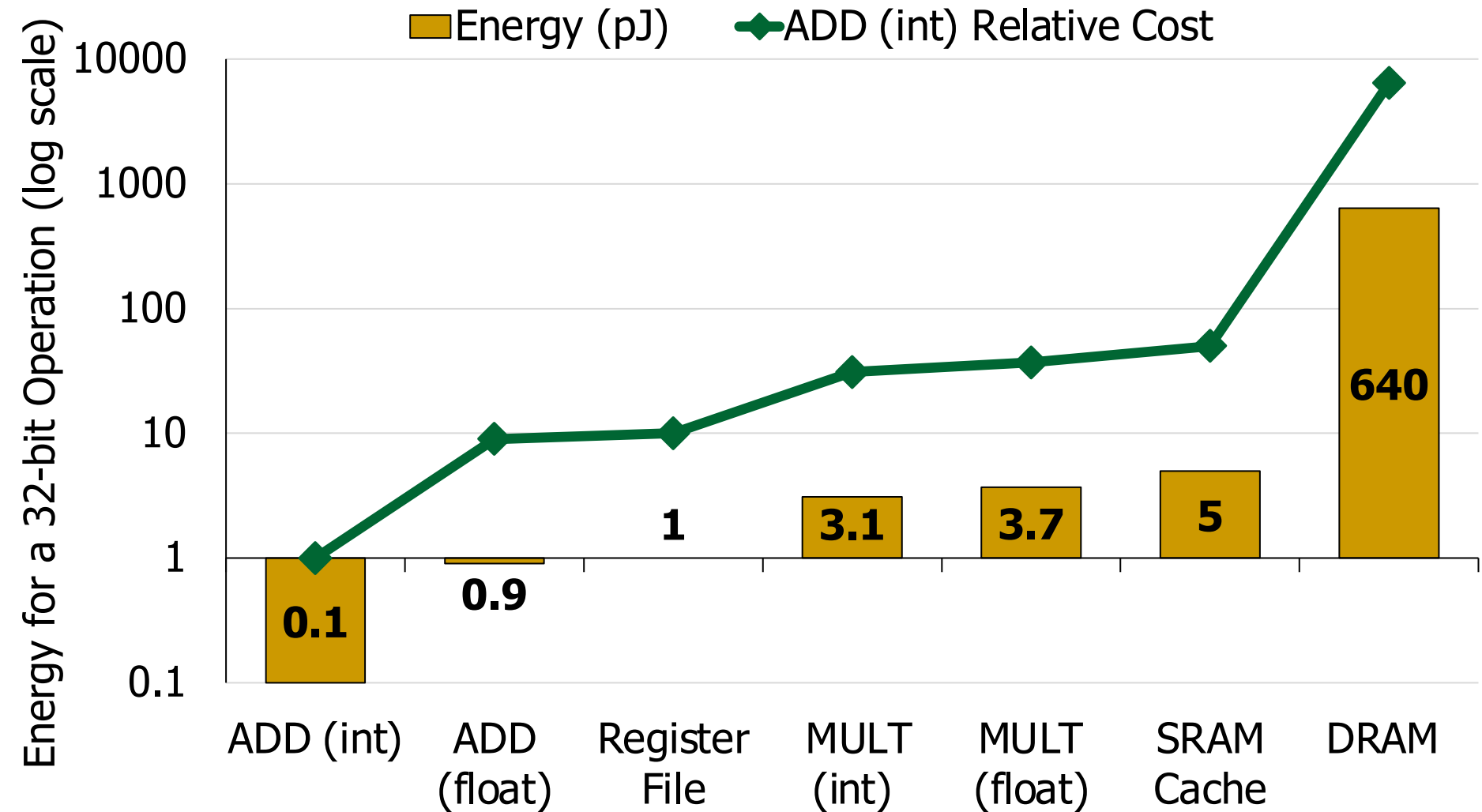
Data Movement vs. Computation Energy
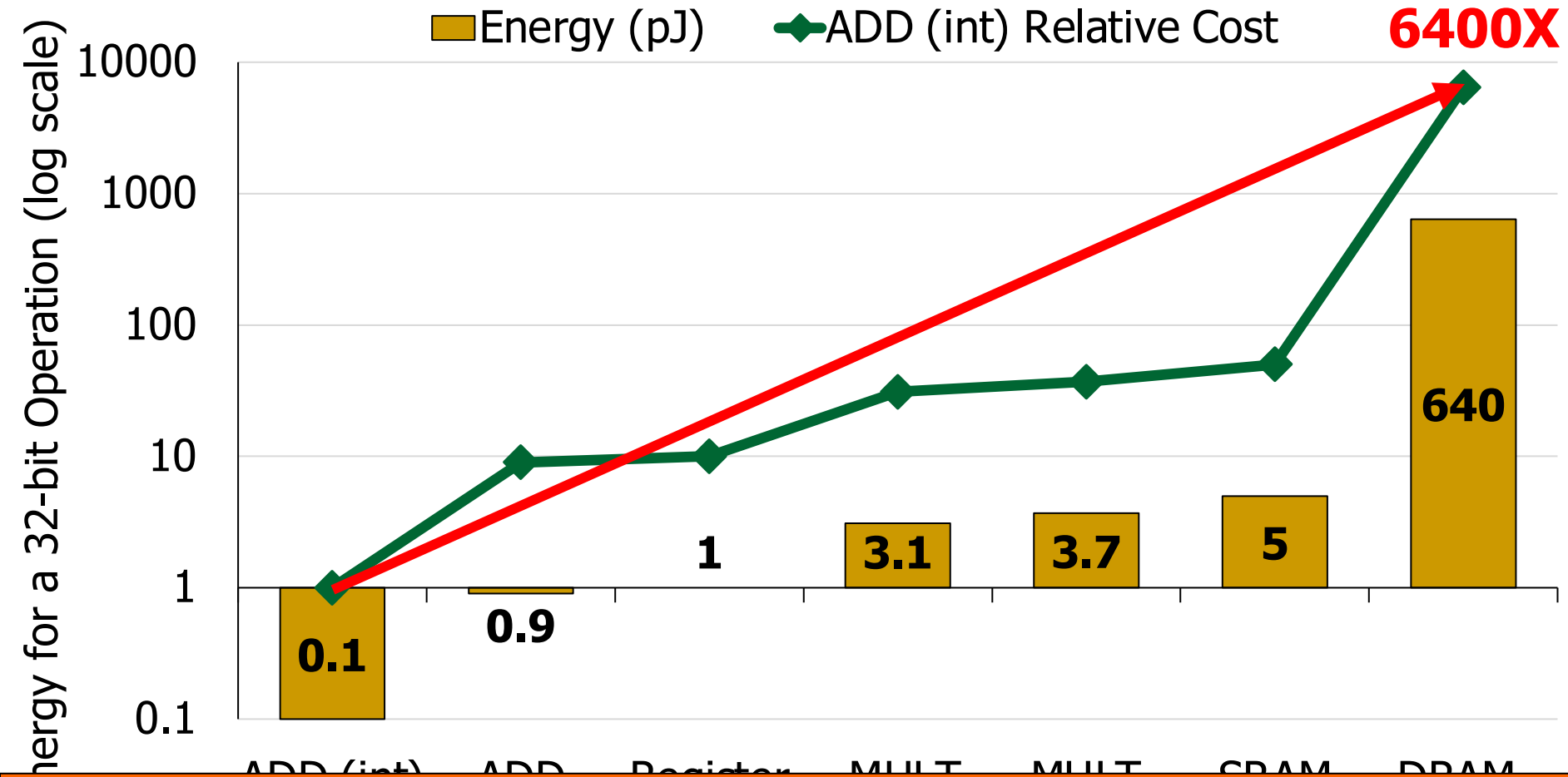
A memory access consumes 6400X the energy of a simple integer addition

Energy Waste in 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.

62.7% of the total system energy
is spent on **data movement**

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1]   Saugata Ghose[1]   Youngsok Kim[2]

Rachata Ausavarungnirun[1]   Eric Shiu[3]   Rahul Thakur[3]   Daehyun Kim[4,3]

Aki Kuusela[3]   Allan Knies[3]   Parthasarathy Ranganathan[3]   Onur Mutlu[5,1]

SAFARI

Energy Waste in Accelerators

- 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)]

> 90% of the total system energy is spent on memory in large ML models

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[⋆†]

[†]*Carnegie Mellon Univ.*   [◇]*Stanford Univ.*   [‡]*Univ. of Illinois Urbana-Champaign*   [§]*Google*   [⋆]*ETH Zürich*
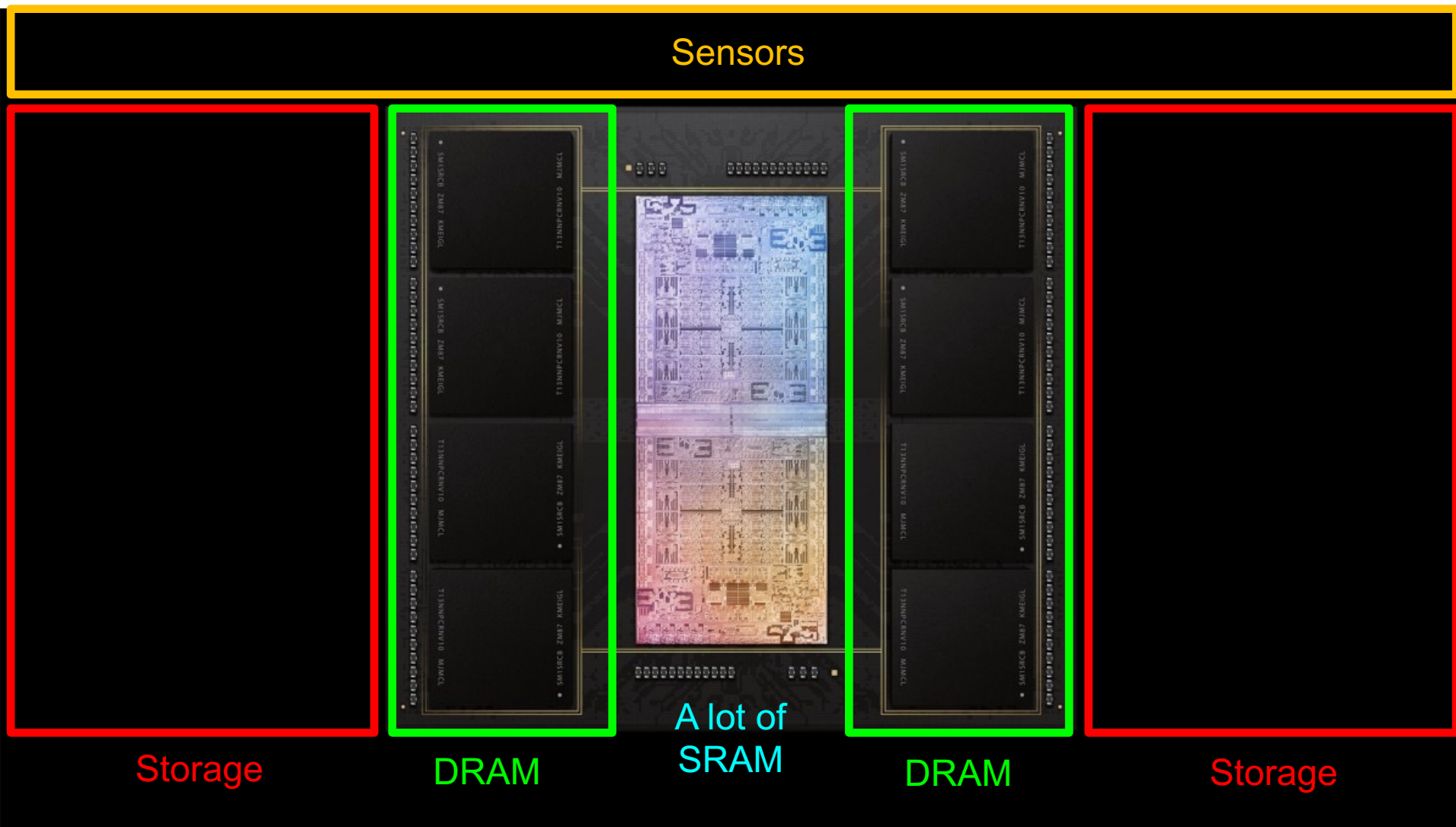
SAFARI

Fundamental Problem

Processing of data
is performed
far away from the data
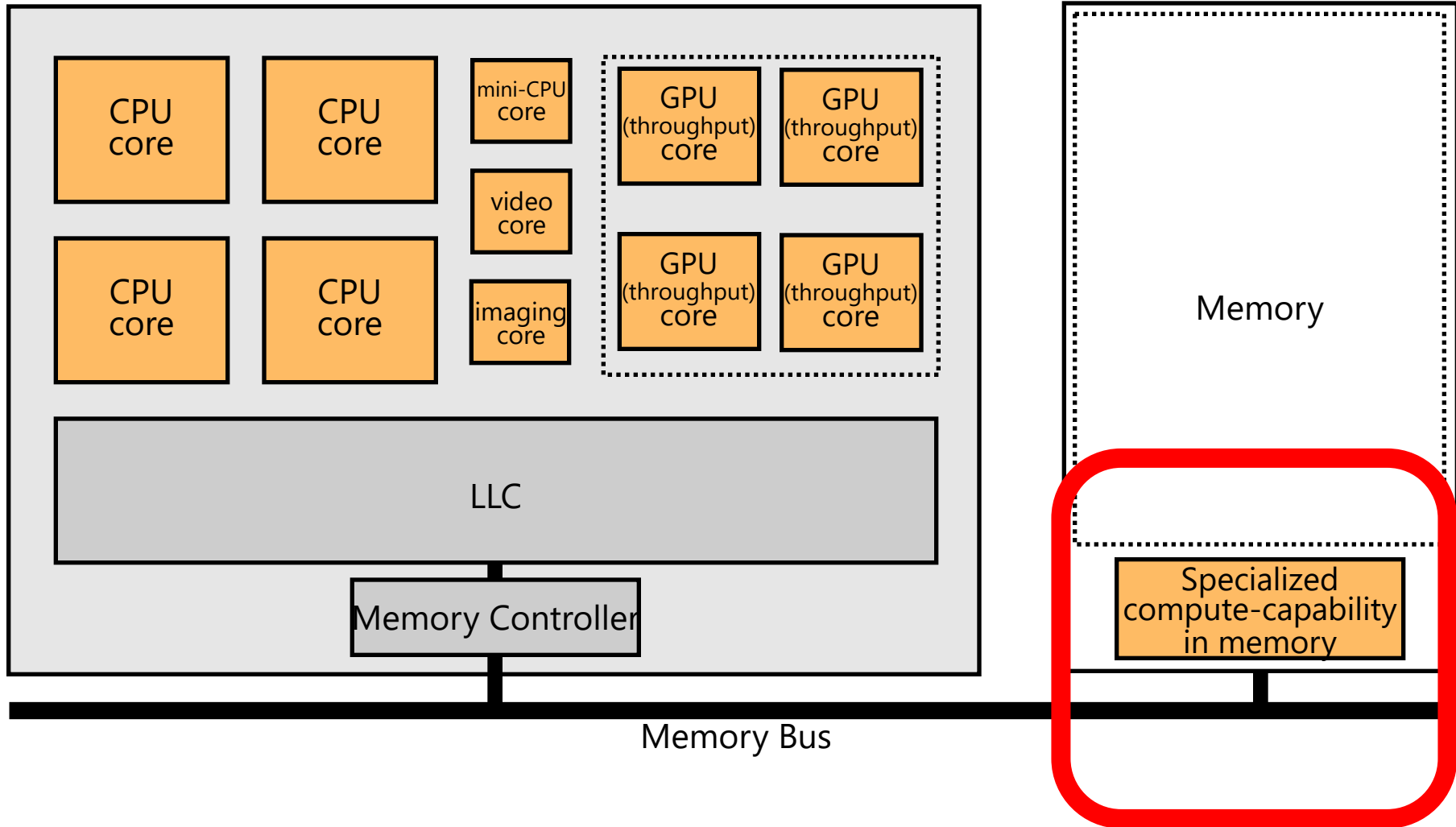
We Need A **Paradigm Shift** To …

- Enable computation with minimal data movement

- Compute where it makes sense (where data resides)

- Make computing architectures more data-centric
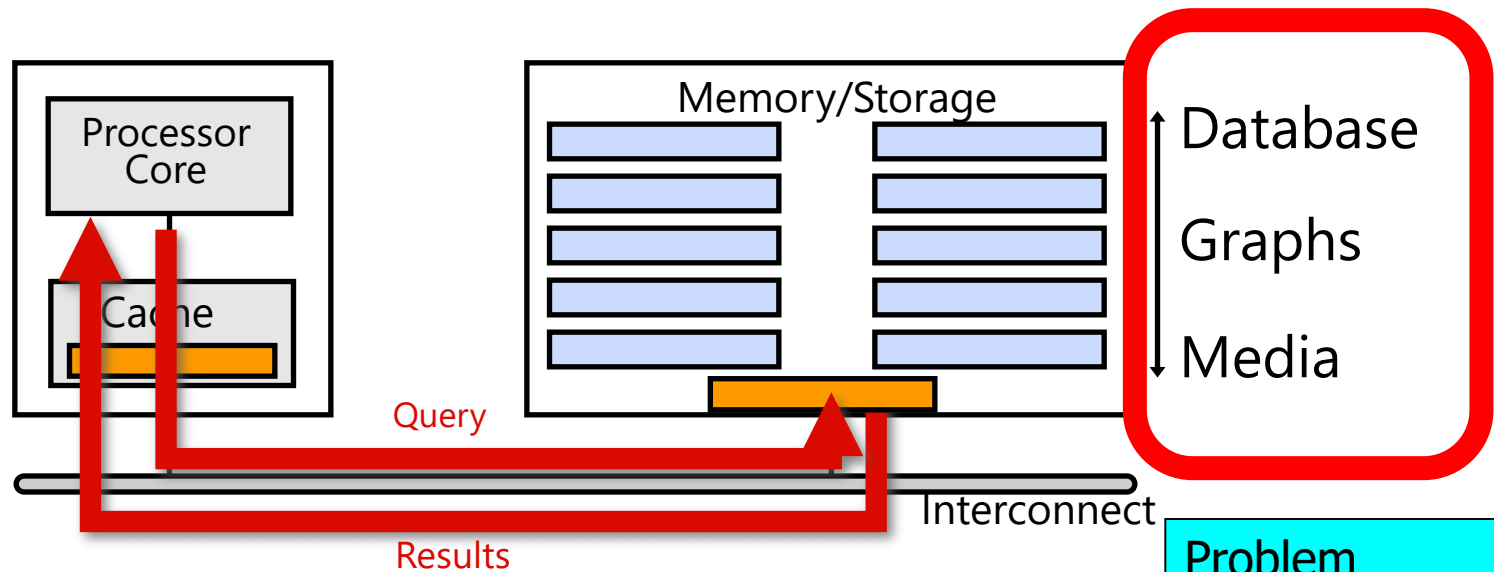
Process Data Where It Makes Sense

Sensors

Storage

DRAM

A lot of SRAM

DRAM

Storage

Apple M1 Ultra System (2022)

Memory as an Accelerator

Memory similar to a "conventional" accelerator

Goal: Processing Inside Memory/Storage
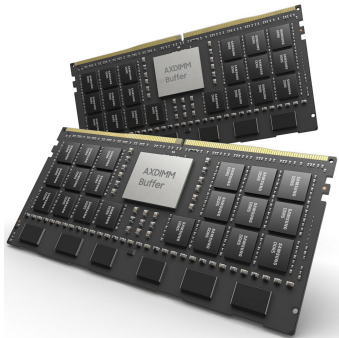

- Many questions … How do we design the:
  - compute-capable memory & controllers?
  - processors & communication units?
  - software & hardware interfaces?
  - system software, compilers, languages?
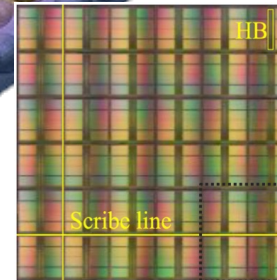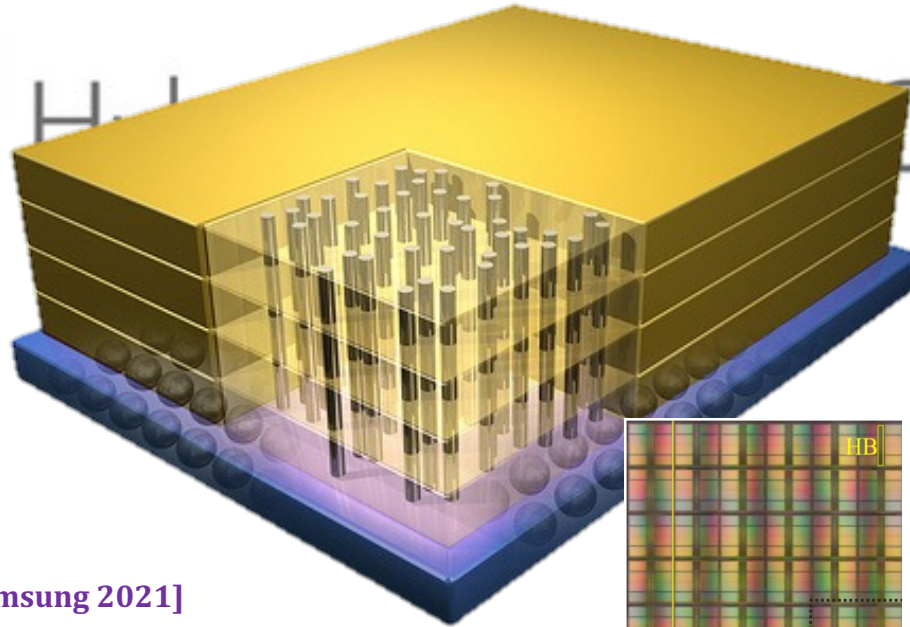 - algorithms & theoretical foundations?

Processing in Memory: Two Types
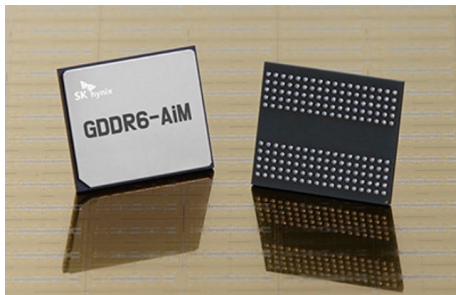
1. Processing **near** Memory
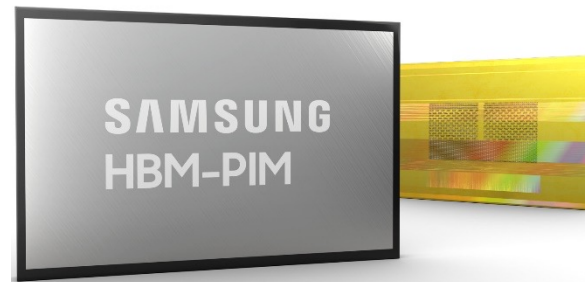2. Processing **using** Memory
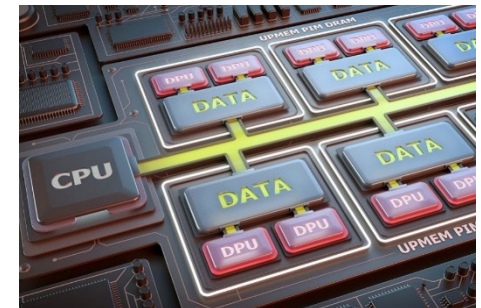
Processing-in-Memory Landscape Today

[Samsung 2021]

[Alibaba 2022]

[SK Hynix 2022]

[Samsung 2021]

[UPMEM 2019]

And, many other experimental chips and startups

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.

Potential Security Issues & Benefits (I)

- Can PIM worsen security?
  - Worsened or easier-to-induce physical issues (e.g., RowHammer)?
  - Worsened or new side channels?
  - Hardware bugs?
  - New threat models?
  - …

- Can PIM enhance security?
  - Less exposure of data (& keys?)
  - In-memory encryption & cryptographic hashing
  - Execution of security functions; trusted execution in memory
  - Support for security primitives (TRNGs, PUFs, encryption, …)
  - More or better isolation?
  - …

Potential Security Issues & Benefits (II)

- Security analysis of PIM Systems
  - Different types of PIM: PnM vs. PuM
  - Different locations: cache, MC, DRAM, NVM, storage, remote, …
  - General-purpose vs. special-purpose PIM?
  - Multi tenancy vs. single workload?
  - Concurrent host and PIM access?
  - Memory bus protection; memory wire(s) protection?
  - Robustness issues like RowHammer, RowPress, …
  - …

- Can PIM support (more) secure execution of workloads?
  - What is needed to do so?
  - Secure PIM enclaves?
  - …

SAFARI

Processing in Memory: Evaluation Methods

Simulators (Open Source)

- Ramulator 2.0 & Ramulator-PIM

- DAMOVSim

- UPMEMSim (UPMEM)

- AiMSim (SK Hynix)

- ...

SAFARI

Ramulator + Gem5

- 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

SAFARI   **https://github.com/CMU-SAFARI/ramulator2**

Opportunity: 3D-Stacked Logic+Memory
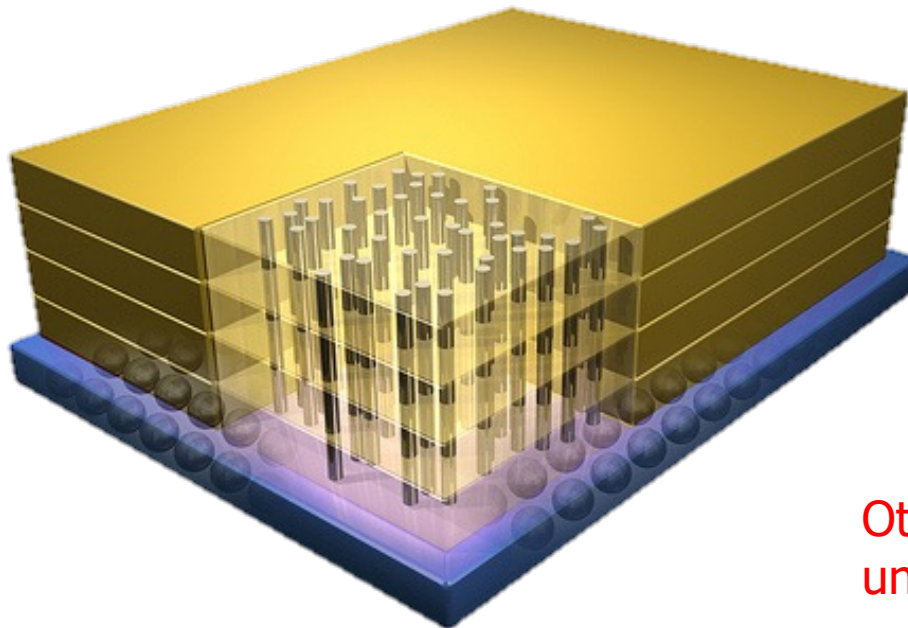
Memory

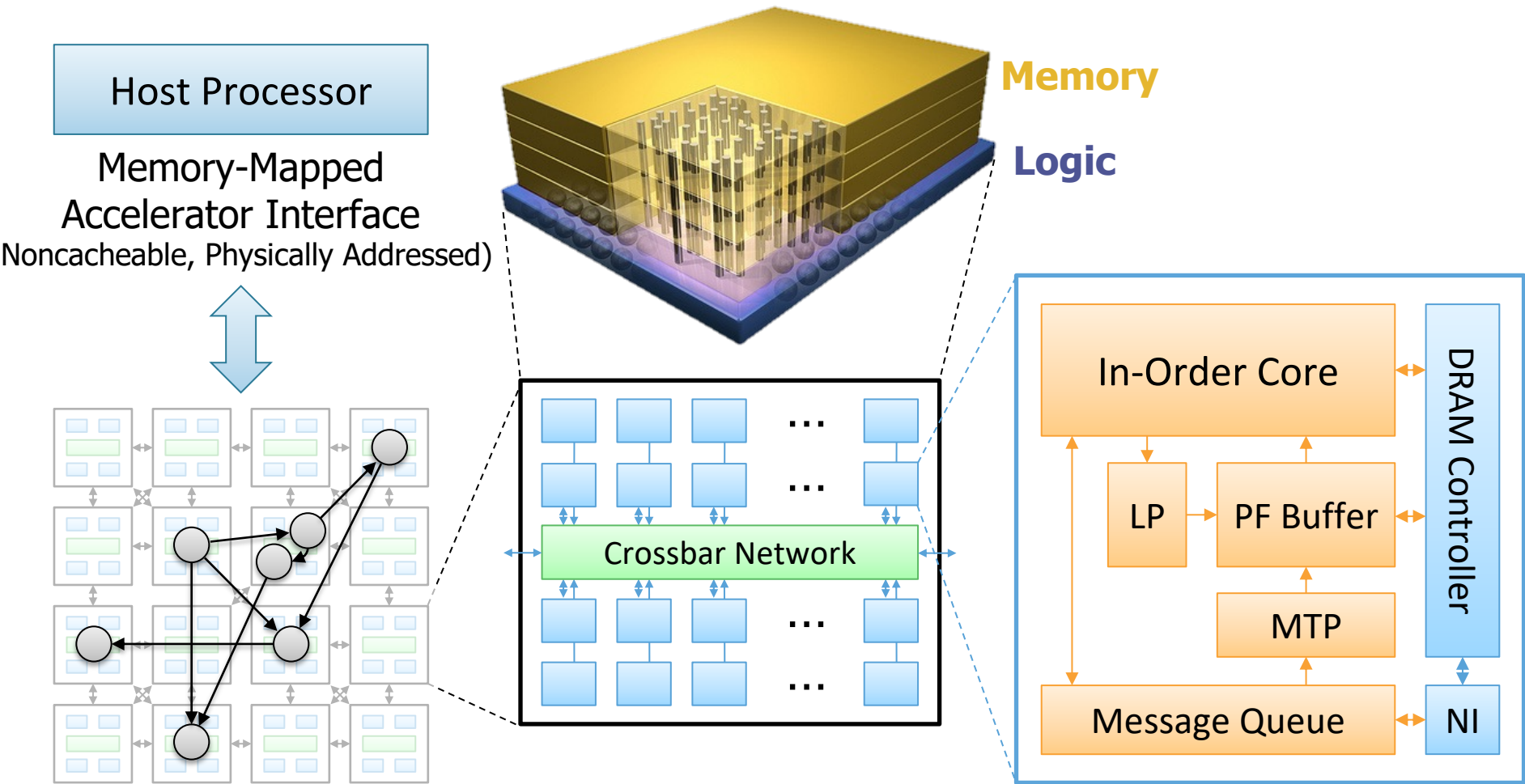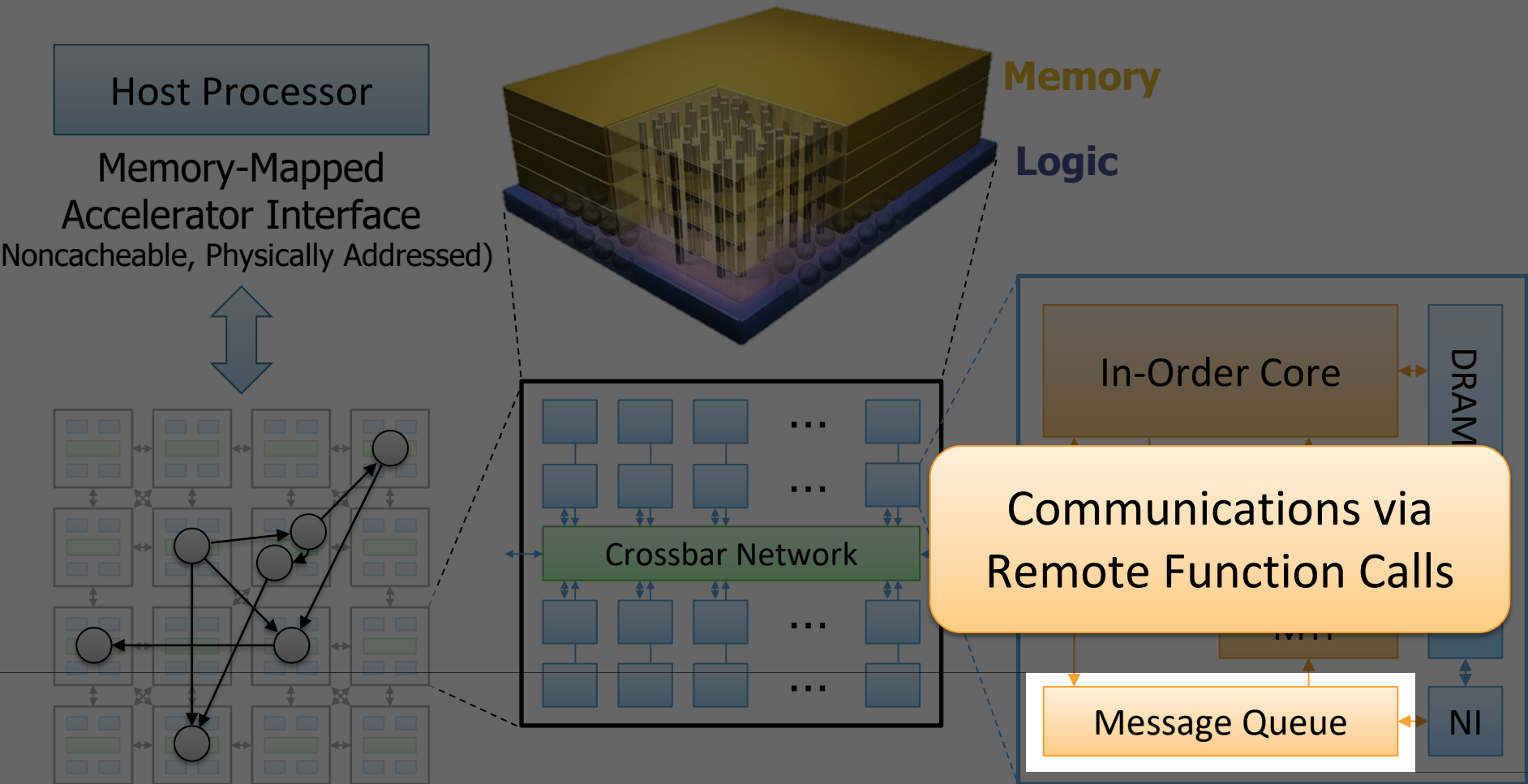**Logic**

Other "True 3D" technologies under development

Tesseract System for Graph Processing

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

Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

In-Order Core

DRAM Controller

LP

PF Buffer

MTP

Message Queue

NI

Tesseract System for Graph Processing

Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

In-Order Core

DRAM

Crossbar Network

...
...
...
...

Communications via
Remote Function Calls

Message Queue

NI

Tesseract System for Graph Processing

Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

...
...
...
...

Prefetching

LP

PF Buffer

MTP

DRAM Controller

Message Queue

NI

Simulated Systems

DDR3-OoO   HMC-OoO   HMC-MC   **Tesseract**

102.4GB/s   640GB/s   640GB/s   **8TB/s**

Tesseract Graph Processing Performance

>13X Performance Improvement

On five graph processing algorithms

13.8x

11.6x

9.0x

+56%

+25%

Speedup

16
14
12
10
8
6
4
2
0

DDR3-OoO   HMC-OoO   HMC-MC   Tesseract   Tesseract-LP   Tesseract-LP-MTP

Tesseract Graph Processing System Energy

Legend: Memory Layers, Logic Layers, Cores

Chart axis values: 1.2, 1, 0.8, 0.6, 0.4, 0.2, 0

Categories: HMC-OoO, Tesseract with Prefetching

> 8X Energy Reduction

More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung 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[†]   Kiyoung Choi

junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
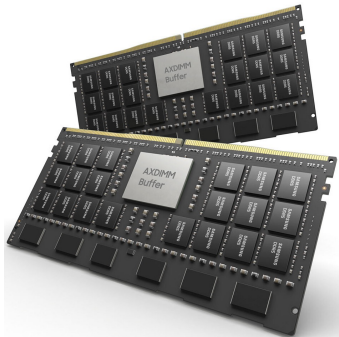
Seoul National University   [§]Oracle Labs   [†]Carnegie Mellon University

PIM Prototypes
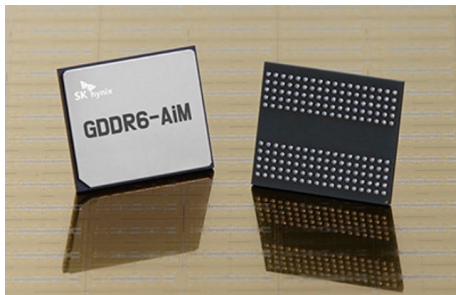
Processing-in-Memory
in the Real World
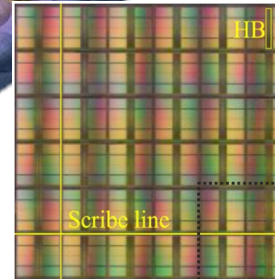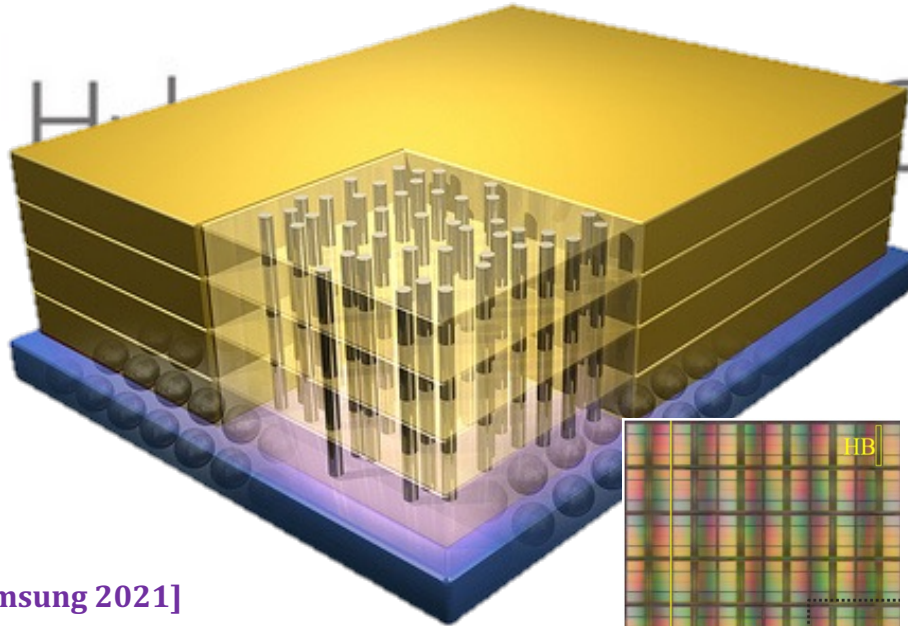
SAFARI

Processing-in-Memory Landscape Today

[Samsung 2021]

[Alibaba 2022]

[SK Hynix 2022]

[Samsung 2021]

[UPMEM 2019]

SAFARI

And, many other experimental chips and startups
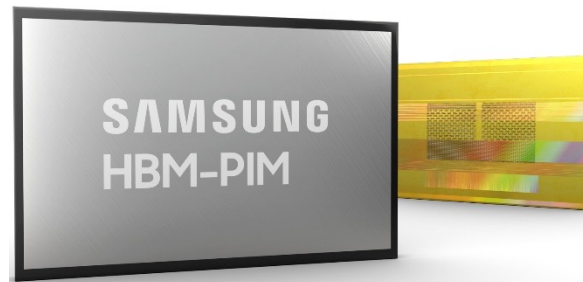
Processing-in-Memory Landscape Today

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], Euiseok 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)
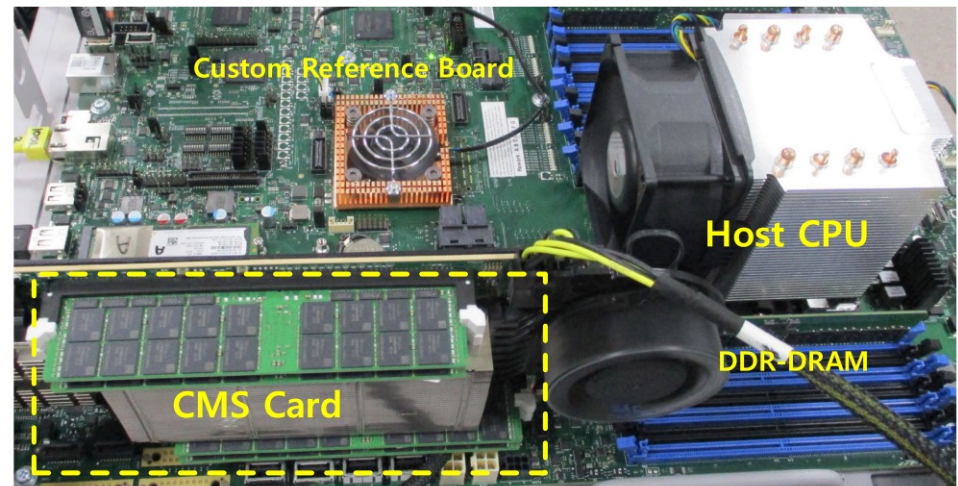
Fig. 6. FPGA prototype of proposed CMS card.

SAFARI

Processing-in-Memory Landscape Today

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

Samsung AxDIMM (2021)

- **DDRx-PIM**
  - DLRM recommendation system

Baseline System

AxDIMM System

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

SAFARI

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%

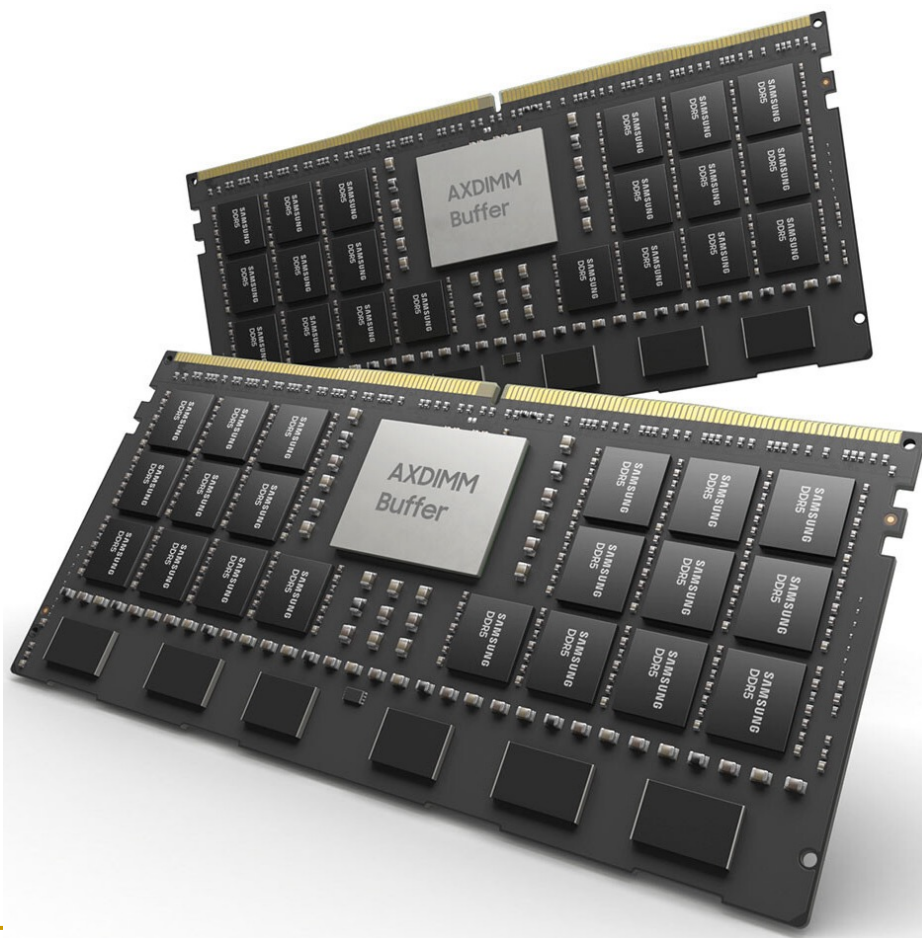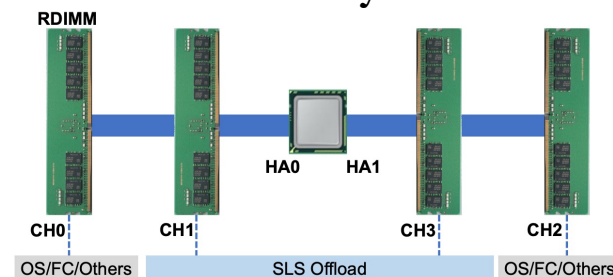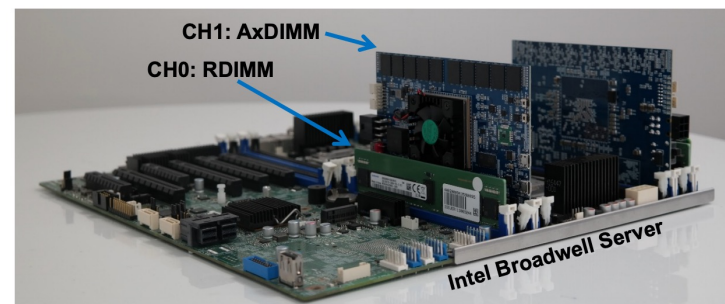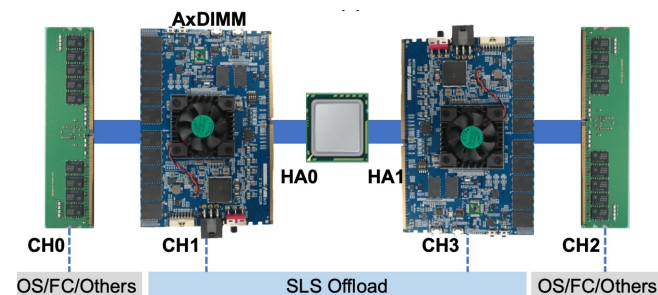Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing 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**

SID1
Core-die
(HBM2)

SID0
Core-die
(FIMDRAM)

Buffer-die →
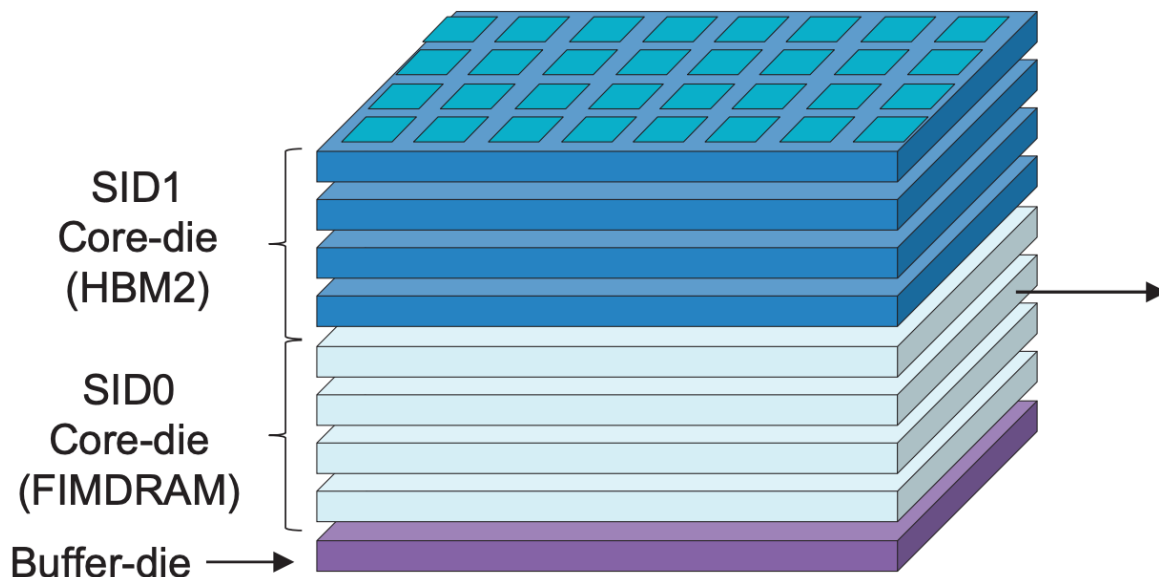
[3D Chip Structure of HBM with FIMDRAM]

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)**

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1],
Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1],
Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1],
Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1],
Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1],
David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3],
Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

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]

40

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

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1],
Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1],
Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1],
Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1],
Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1],
David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3],
Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
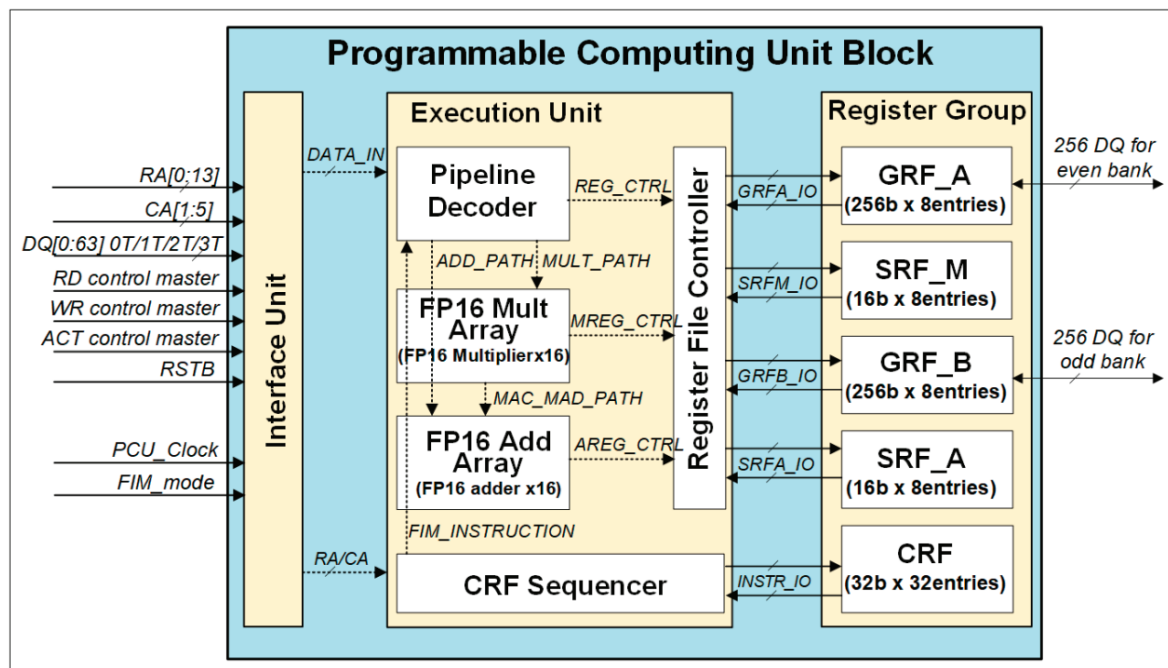[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

41

Chip Implementation

- **Mixed design methodology to implement FIMDRAM**
  - Full-custom + Digital RTL
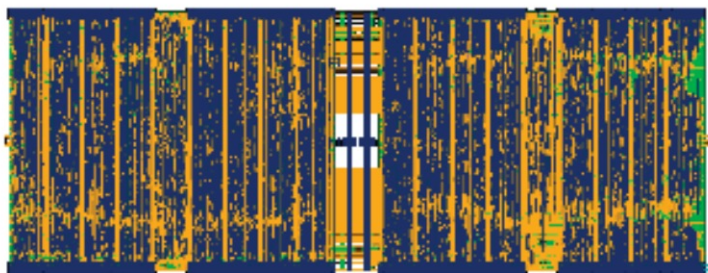
[Digital RTL design for PCU block]

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1],
Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1],
Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1],
Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1],
Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1],
David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[1], JoonHo Song[3],
Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
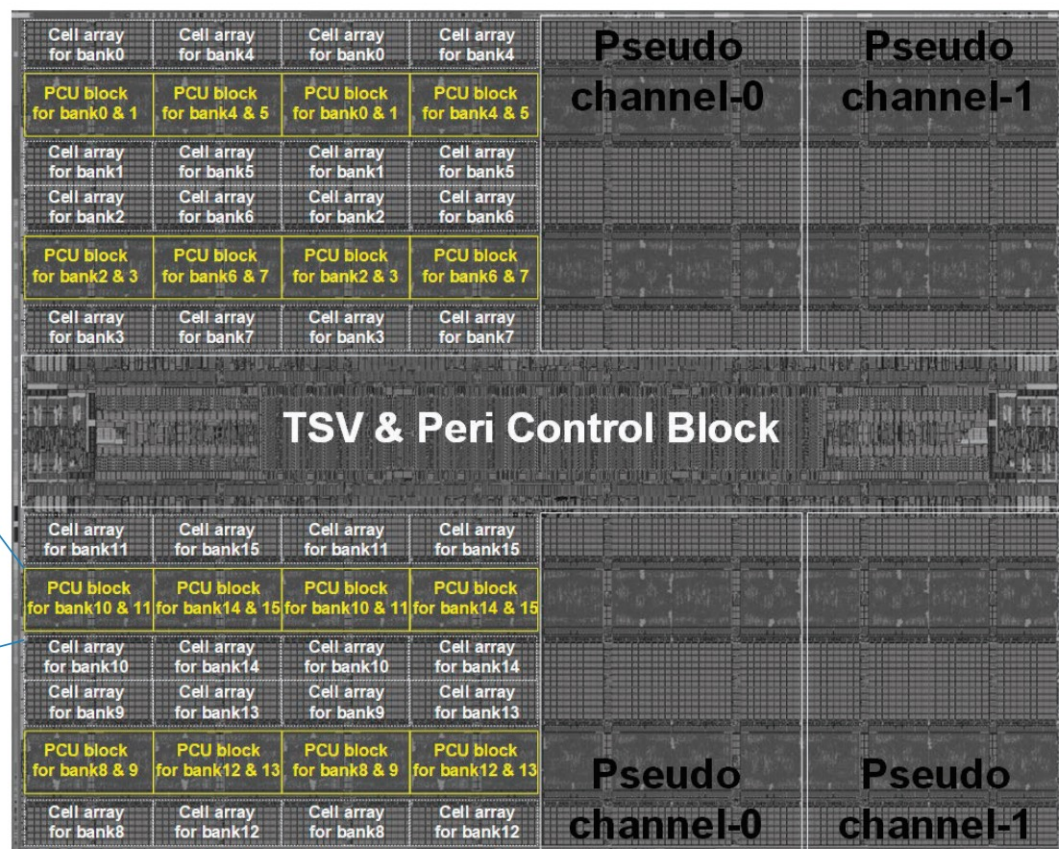[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

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.
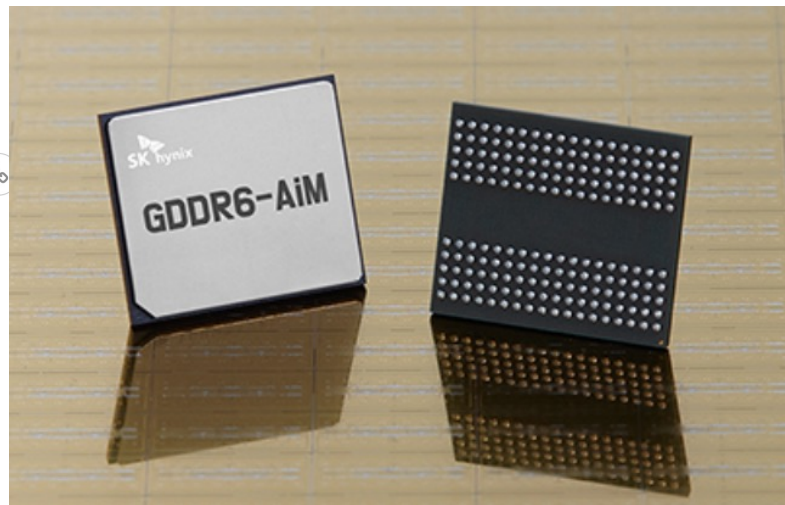
11.1 A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications

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.

43

https://news.skhynix.com/sk-hynix-develops-pim-next-generation-ai-accelerator/

SK Hynix Accelerator-in-Memory (2022)

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

AliBaba PIM Recommendation System (2022)

ISSCC 2022 / February 24, 2022 / 8:30 AM
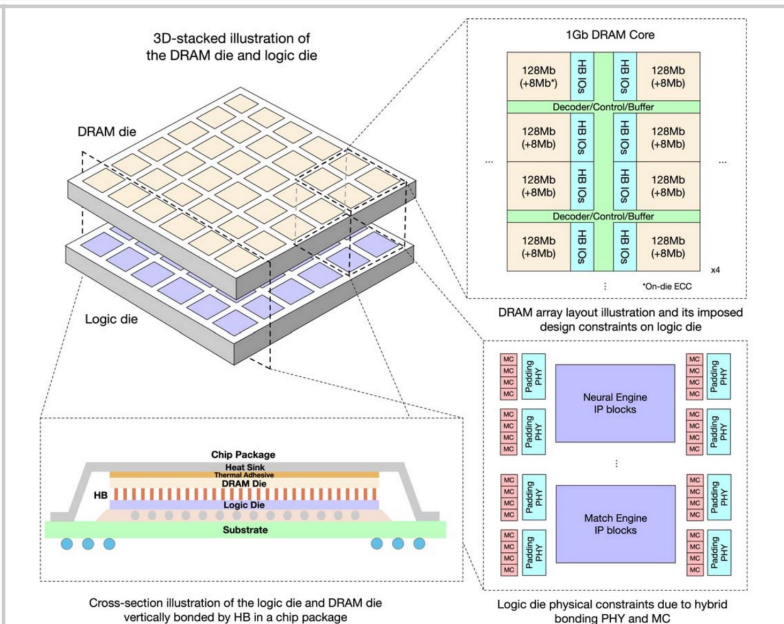
Figure 29.1.2: Illustration of 3D-stacked chip, cross-illustration of package, DRAM array layout and design blocks on logic die.
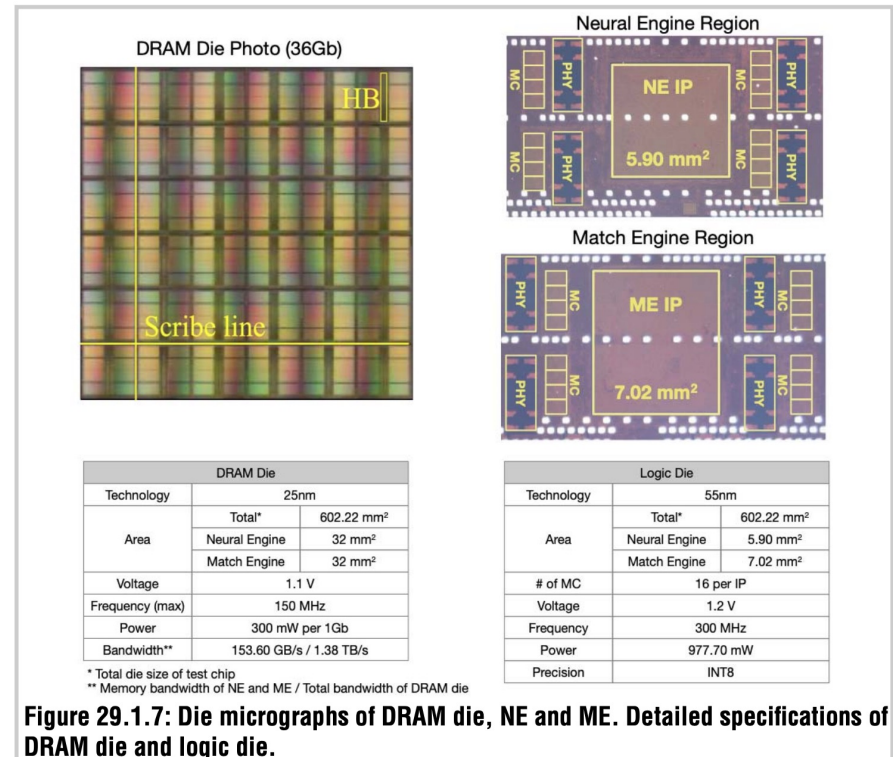
Figure 29.1.7: Die micrographs of DRAM die, NE and ME. Detailed specifications of DRAM die and logic die.

DRAM Die		
Technology	25nm	
Area	Total*	602.22 mm²
	Neural Engine	32 mm²
	Match Engine	32 mm²
Voltage	1.1 V	
Frequency (max)	150 MHz	
Power	300 mW per 1Gb	
Bandwidth**	153.60 GB/s / 1.38 TB/s	

Logic Die		
Technology	55nm	
Area	Total*	602.22 mm²
	Neural Engine	5.90 mm²
	Match Engine	7.02 mm²
# of MC	16 per IP	
Voltage	1.2 V	
Frequency	300 MHz	
Power	977.70 mW	
Precision	INT8	

* Total die size of test chip
** Memory bandwidth of NE and ME / Total bandwidth of DRAM die

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

Dimin Niu[1], Shuangchen Li[1], Yuhao Wang[1], Wei Han[1], Zhe Zhang[2], Yijin Guan[2], Tianchan Guan[3], Fei Sun[1], Fei Xue[1], Lide Duan[1], Yuanwei Fang[1], Hongzhong Zheng[1], Xiping Jiang[4], Song Wang[4], Fengguo Zuo[4], Yubing Wang[4], Bing Yu[4], Qiwei Ren[4], Yuan Xie[1]

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

8GB/128xDPU PIM R-DIMM Module

CPU (x86, ARM, RV...) ⟷ DDR Data bus

UPMEM PIM chip ... x N
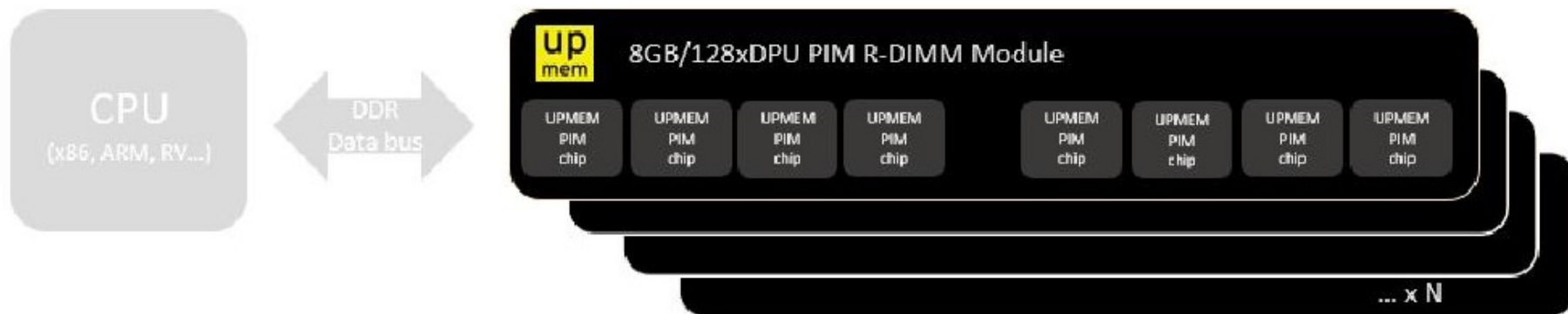
UPMEM Memory Modules

- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz

SAFARI
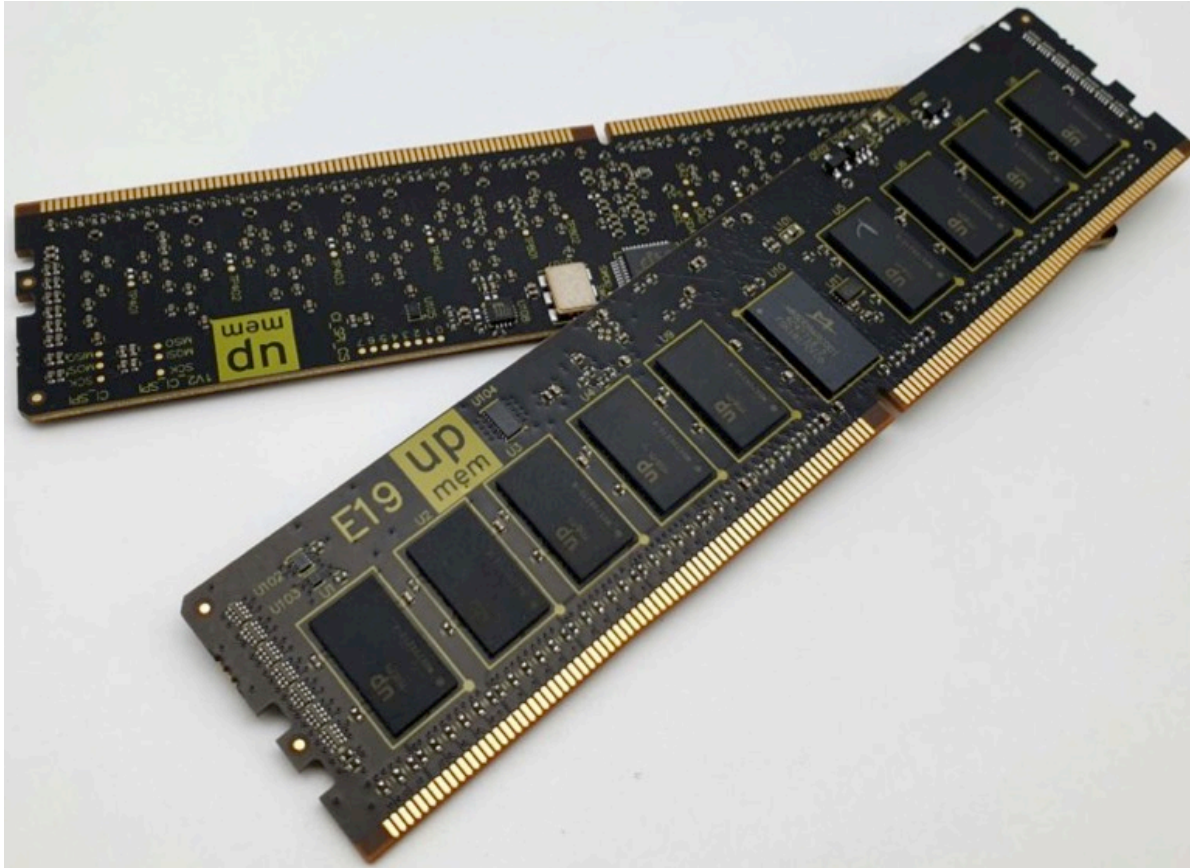
2,560-DPU Processing-in-Memory System

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*).

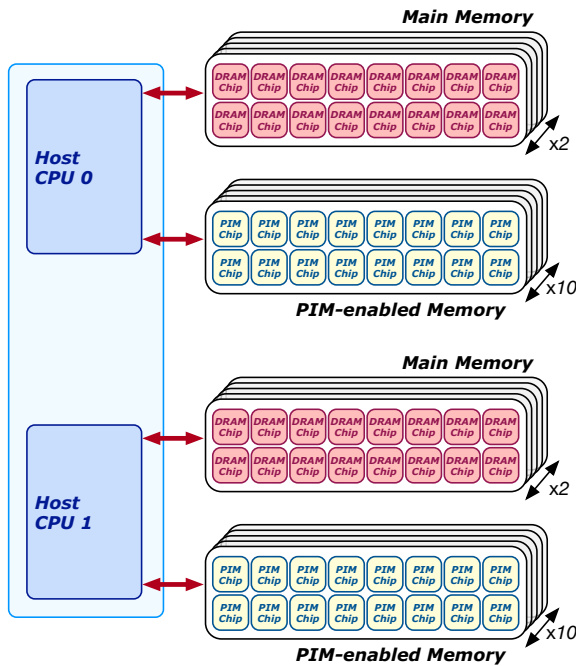Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware 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.

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

More on the UPMEM PIM System
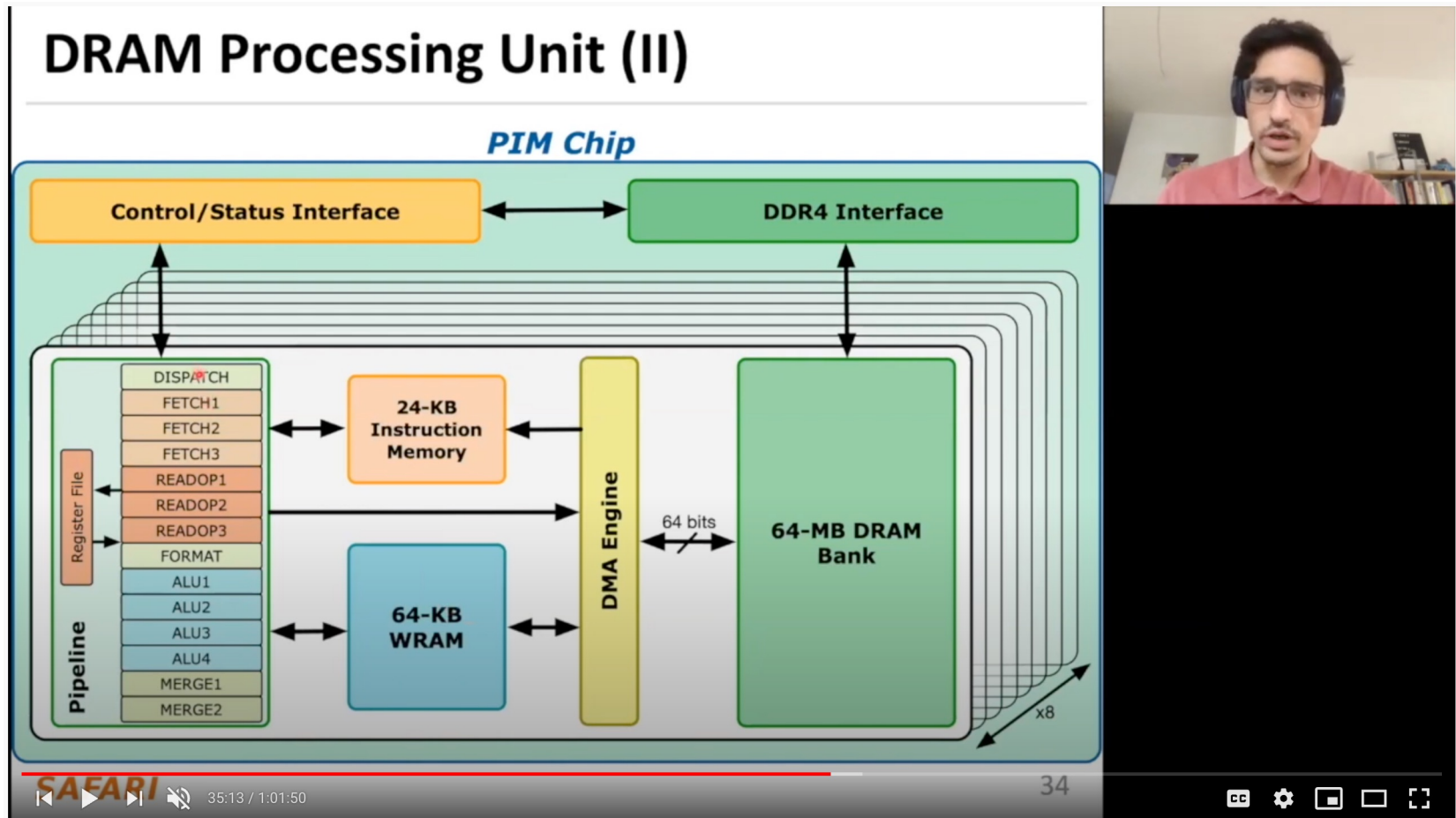
Experimental Analysis of the UPMEM PIM Engine

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.

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

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
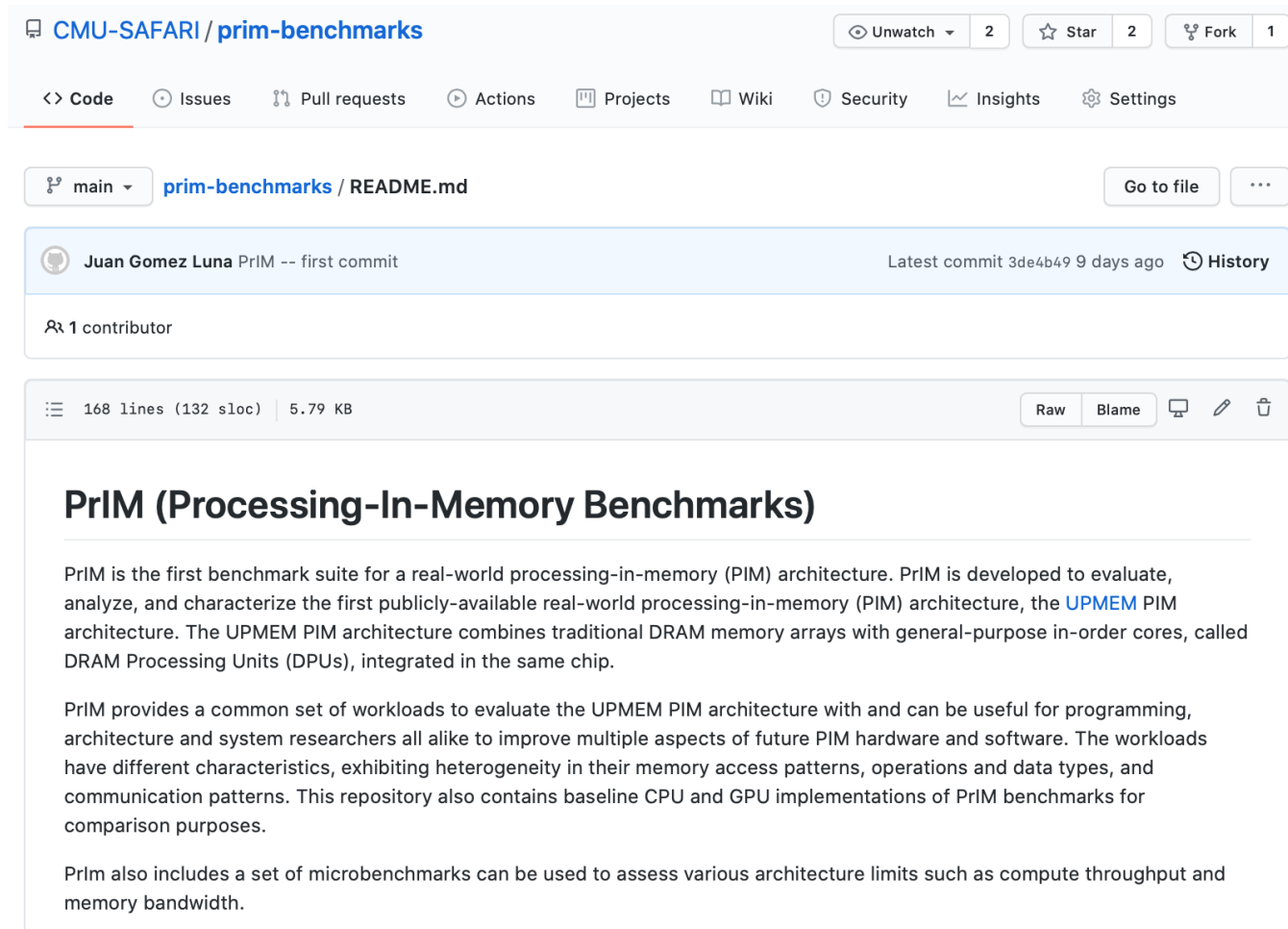
Onur Mutlu
ETH Zürich

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

Understanding a Modern PIM Architecture

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

JUAN GÓMEZ-LUNA[1], IZZAT EL HAJJ[2], IVAN FERNANDEZ[1,3], CHRISTINA GIANNOULA[1,4], GERALDO F. OLIVEIRA[1], AND ONUR MUTLU[1]

[1] ETH Zürich
[2] American University of Beirut
[3] University of Malaga
[4] National 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

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for <u>P</u>rocessing-<u>i</u>n-<u>DRAM</u>

Ataberk Olgun[§†]   Juan Gómez Luna[§]   Konstantinos Kanellopoulos[§]   Behzad Salami[§*]
Hasan Hassan[§]   Oğuz Ergin[†]   Onur Mutlu[§]

[§]ETH Zürich   [†]TOBB ETÜ   [*]BSC

https://arxiv.org/pdf/2111.00082.pdf
https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing Using Memory Prototype

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing Using Memory Prototype

≡ README.md   ✎

Building a PiDRAM Prototype

To build PiDRAM's prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR's fpga-zynq repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM's memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPGAConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPGAConfig -j<number of cores>`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (system_top.bit) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx PHY IP we use in PiDRAM's memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:

1- Open IP Catalog
2- Find "Memory Interface Generator (MIG 7 Series)" IP and double click

https://arxiv.org/pdf/2111.00082.pdf
https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Microbenchmark Copy/Initialization Throughput

In-DRAM Copy and Initialization improve throughput by 119x and 89x

More on PiDRAM

- Ataberk Olgun, Juan Gomez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"
  ACM Transactions on Architecture and Code Optimization (**TACO**), March 2023.
  [arXiv version]
  Presented at the 18th HiPEAC Conference, Toulouse, France, January 2023.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (40 minutes)]
  [PiDRAM Source Code]

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun§   Juan Gómez Luna§   Konstantinos Kanellopoulos§   Behzad Salami§

Hasan Hassan§   Oğuz Ergin†   Onur Mutlu§

§ETH Zürich   †TOBB University of Economics and Technology

Eliminating the Adoption Barriers

How to Enable Adoption of Processing in Memory

SAFARI

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

Potential Security Issues & Benefits (I)

- Can PIM worsen security?
  - Worsened or easier-to-induce physical issues (e.g., RowHammer)?
  - Worsened or new side channels?
  - Hardware bugs?
  - ...

- Can PIM enhance security?
  - Less exposure of data (& keys?)
  - In-memory encryption & cryptographic hashing
  - Execution of security functions; trusted execution in memory
  - Support for security primitives (TRNGs, PUFs, encryption, ...)
  - ...

SAFARI

Potential Security Issues & Benefits (II)

- Security evaluation of PIM Systems
  - Different types of PIM: PnM vs. PuM
  - Different locations: cache, MC, DRAM, NVM, storage, remote, …
  - General-purpose vs. special-purpose PIM?
  - Multi tenancy vs. single workload?
  - Concurrent host and PIM access?
  - Memory bus protection; memory wire(s) protection?
  - Robustness issues like RowHammer, RowPress, …
  - …

- Can PIM support (more) secure execution of workloads?
  - What is needed to do so?
  - Secure PIM enclaves?
  - …

SAFARI

Security of PIM Systems

Onur Mutlu

omutlu@gmail.com

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

30 November 2023

Dagstuhl MAD (Microarchitectural Attacks & Defenses)

SAFARI   **ETH**zürich

Processing in/near Memory: An Old Idea

- Kautz, "Cellular Logic-in-Memory Arrays", IEEE TC 1969.

IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be "programmed" to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.

Fig. 1. Cellular sorting array I.

$\hat{x}$ leads return to X-register)

CELL EQUATIONS:
$\hat{x} = \bar{w}x + wy$
$s_y = wcx, \quad r_y = wc\bar{x}$
$\hat{z} = M(x, \bar{y}, z) = x\bar{y} + z(x + \bar{y})$

Processing in/near Memory: An Old Idea

- Stone, "A Logic-in-Memory Computer," IEEE TC 1970.

A Logic-in-Memory Computer
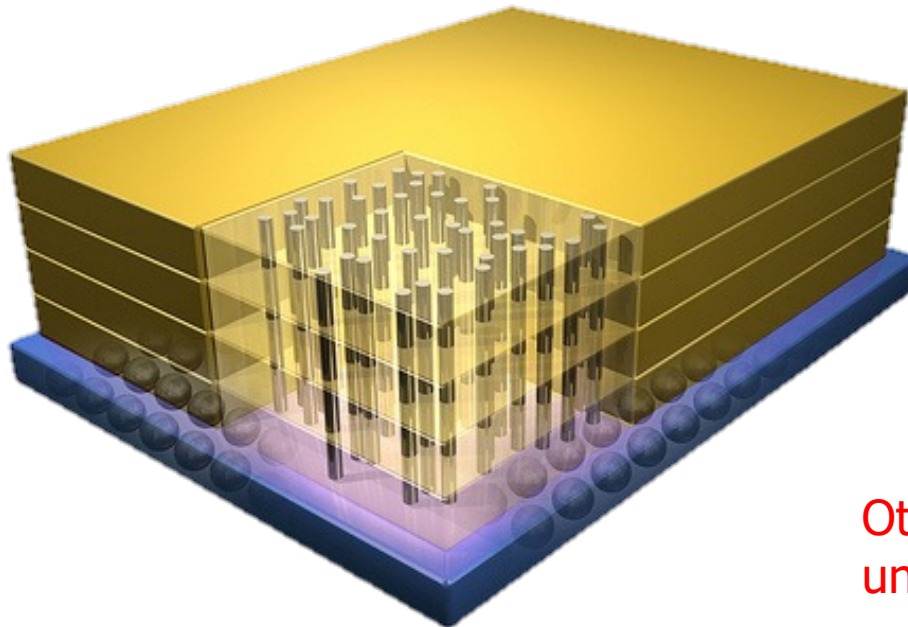
HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.

Processing-near-Memory Systems

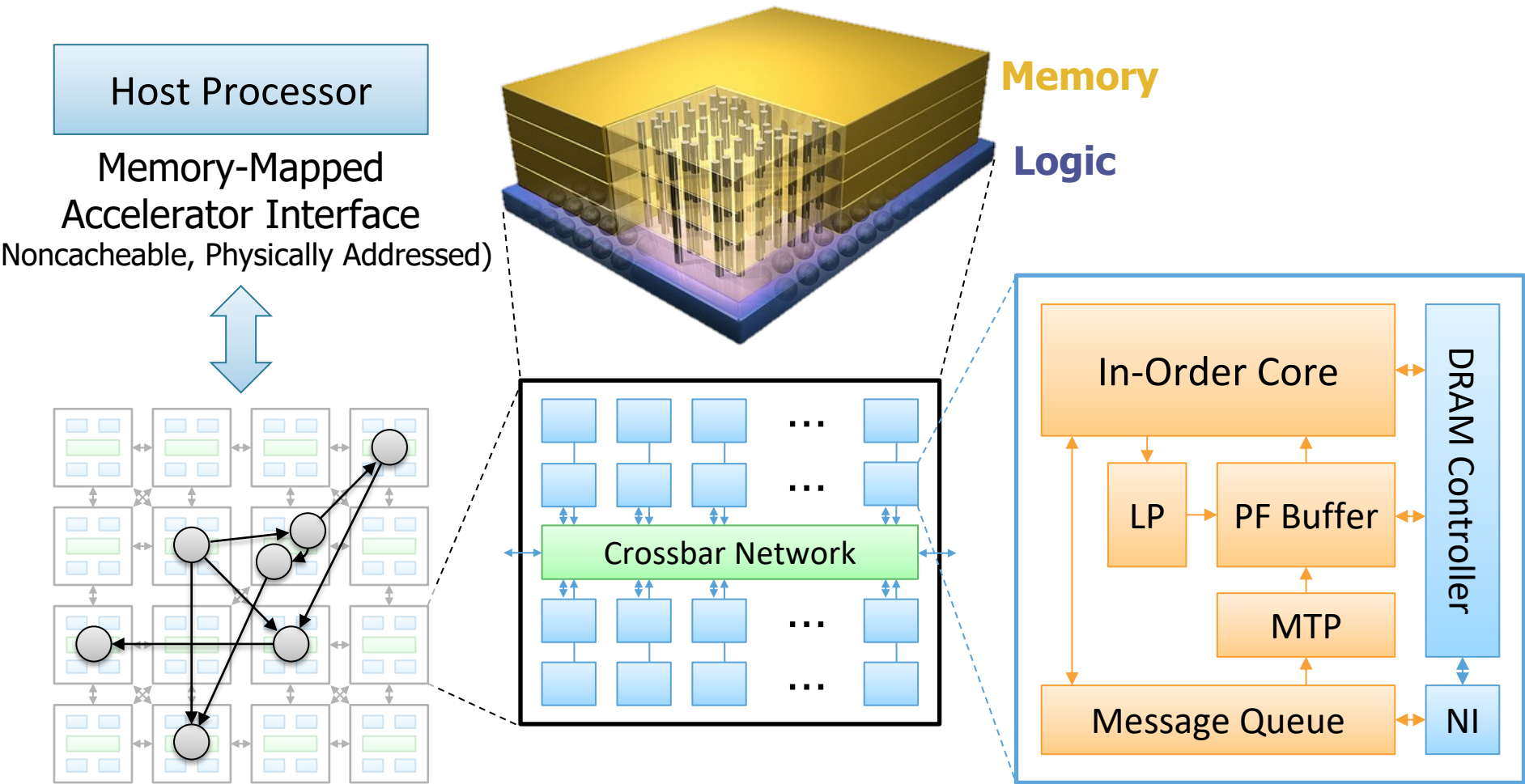# Opportunity: 3D-Stacked Logic+Memory

Hybrid Memory Cube CONSORTIUM
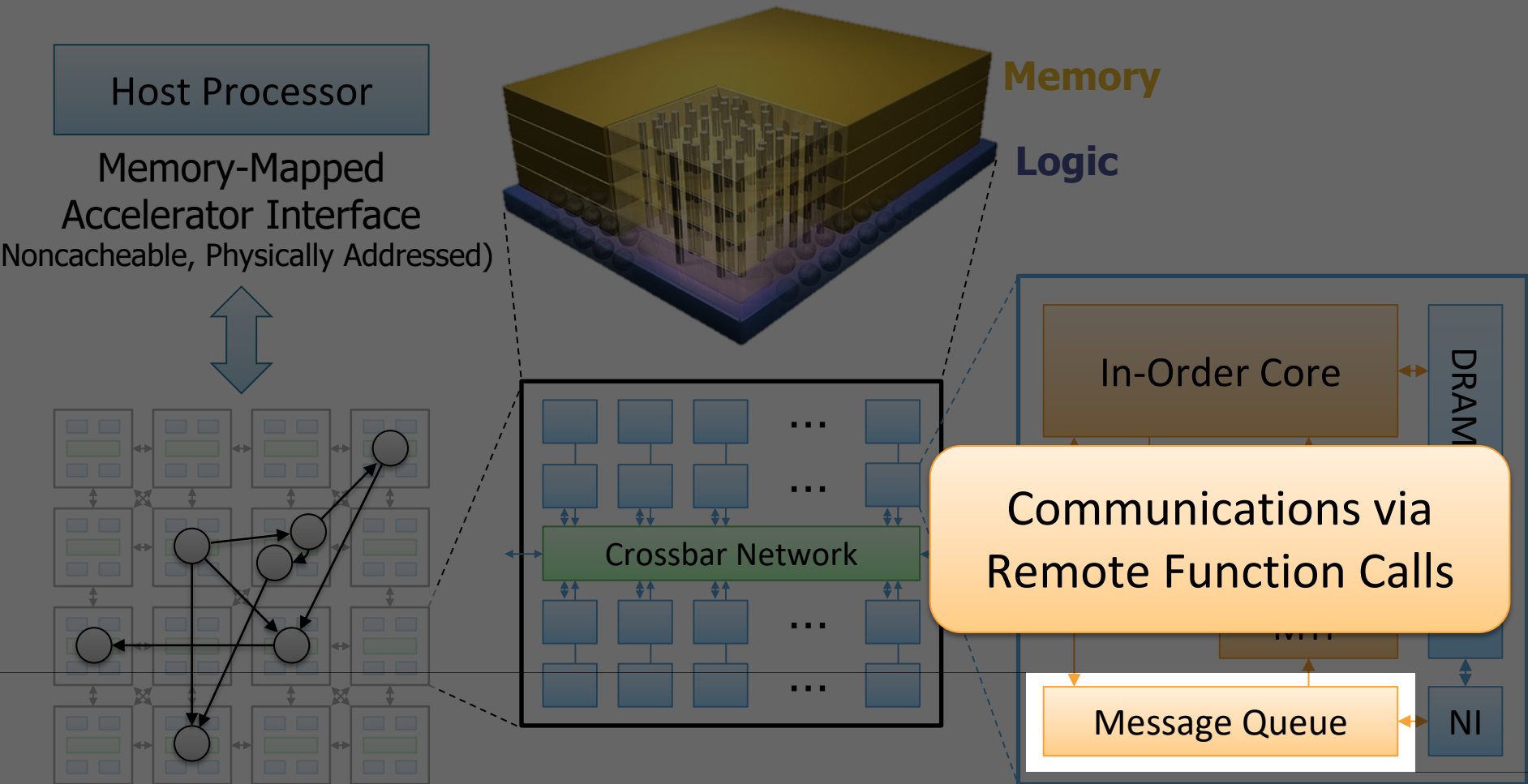
Memory

Logic

Other "True 3D" technologies under development
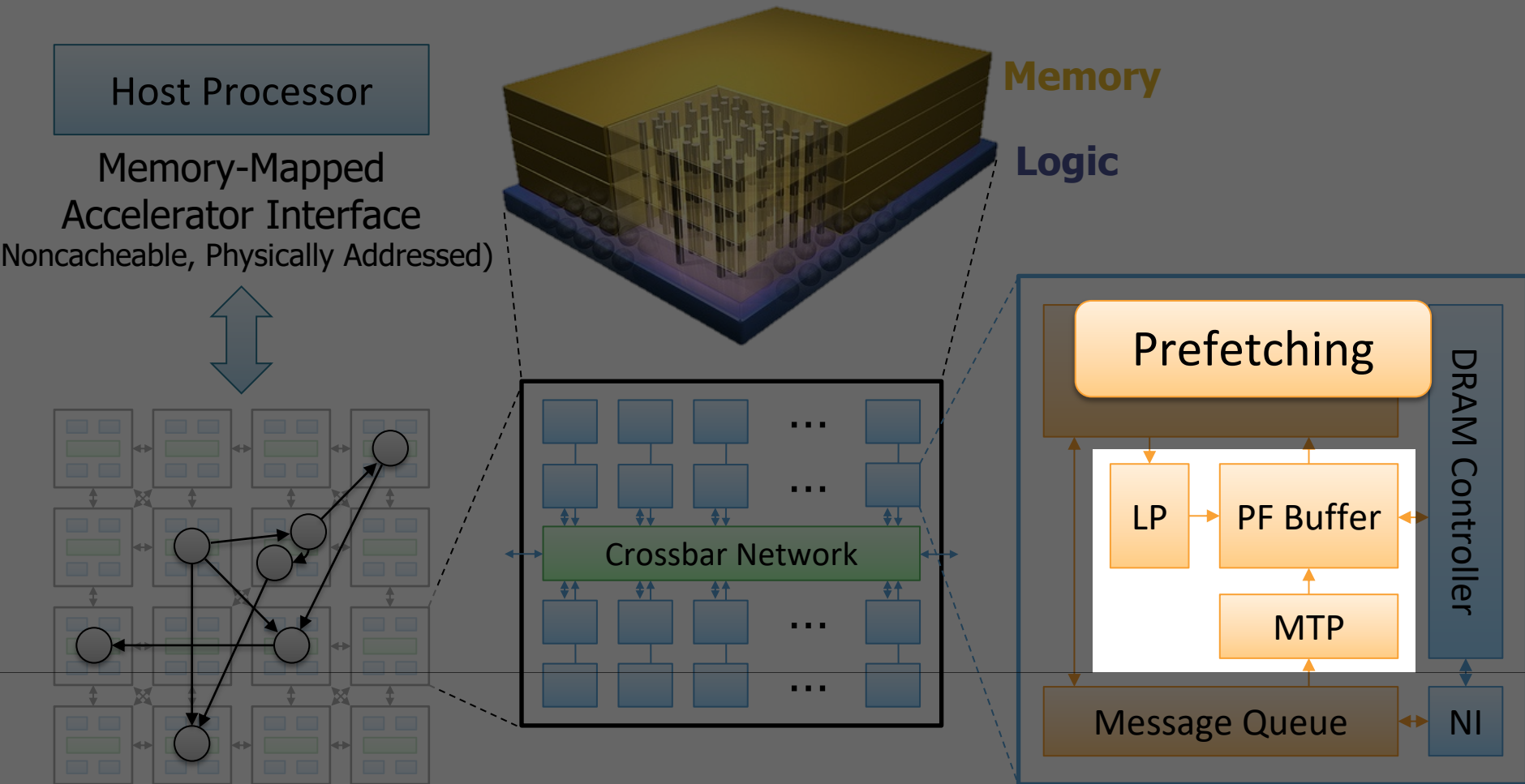
SAFARI

Tesseract System for Graph Processing

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

Host Processor

Memory-Mapped
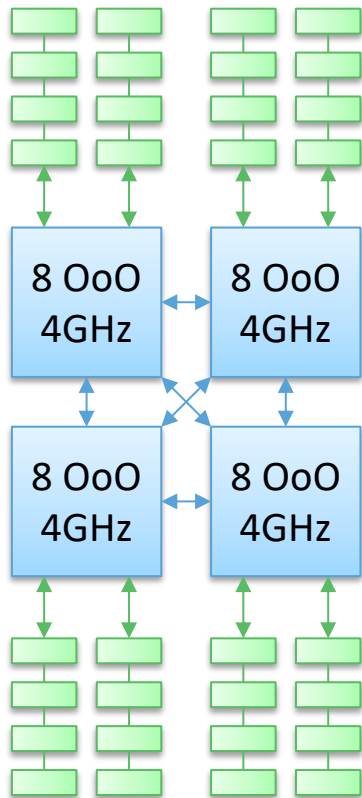Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

In-Order Core

LP   PF Buffer

MTP

Message Queue

DRAM Controller

NI

Tesseract System for Graph Processing

Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

In-Order Core

DRAM

Crossbar Network

**Communications via
Remote Function Calls**

Message Queue

NI

Tesseract System for Graph Processing

Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

...
...
...
...

Prefetching

LP PF Buffer

MTP

DRAM Controller

Message Queue

NI

Evaluated Systems

DDR3-OoO	HMC-OoO	HMC-MC	Tesseract
102.4GB/s	640GB/s	640GB/s	**8TB/s**

Tesseract Graph Processing Performance

>13X Performance Improvement

On five graph processing algorithms

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

Tesseract Graph Processing System Energy

Legend: ■ Memory Layers ■ Logic Layers □ Cores

> 8X Energy Reduction

X-axis: HMC-OoO, Tesseract with Prefetching
Y-axis: 0, 0.2, 0.4, 0.6, 0.8, 1, 1.2

More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung 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[†]   Kiyoung 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

Processing-using-Memory Systems

SAFARI

Processing using DRAM

- We can support
  - Bulk bitwise AND, OR, NOT, MAJ
  - Bulk bitwise COPY and INIT/ZERO
  - True Random Number Generation; Physical Unclonable Functions
  - Lookup Table based more complex computation
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating (multiple) rows performs computation

- 30-77X performance and energy improvement
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
  - Seshadri+"RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

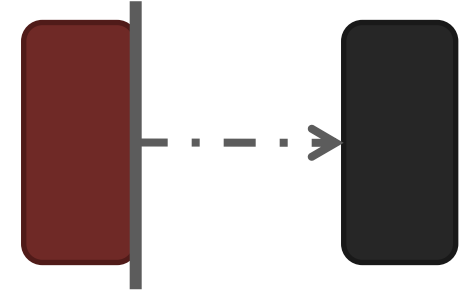# Starting Simple: Data Copy and Initialization

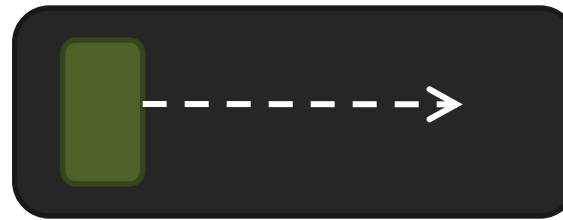*memmove & memcpy:* 5% cycles in Google's datacenter [Kanev+ ISCA'15]
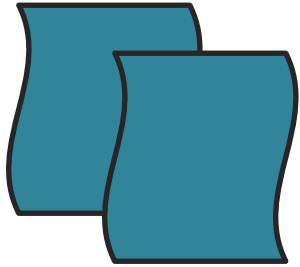
Forking
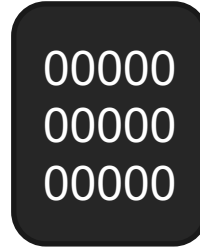
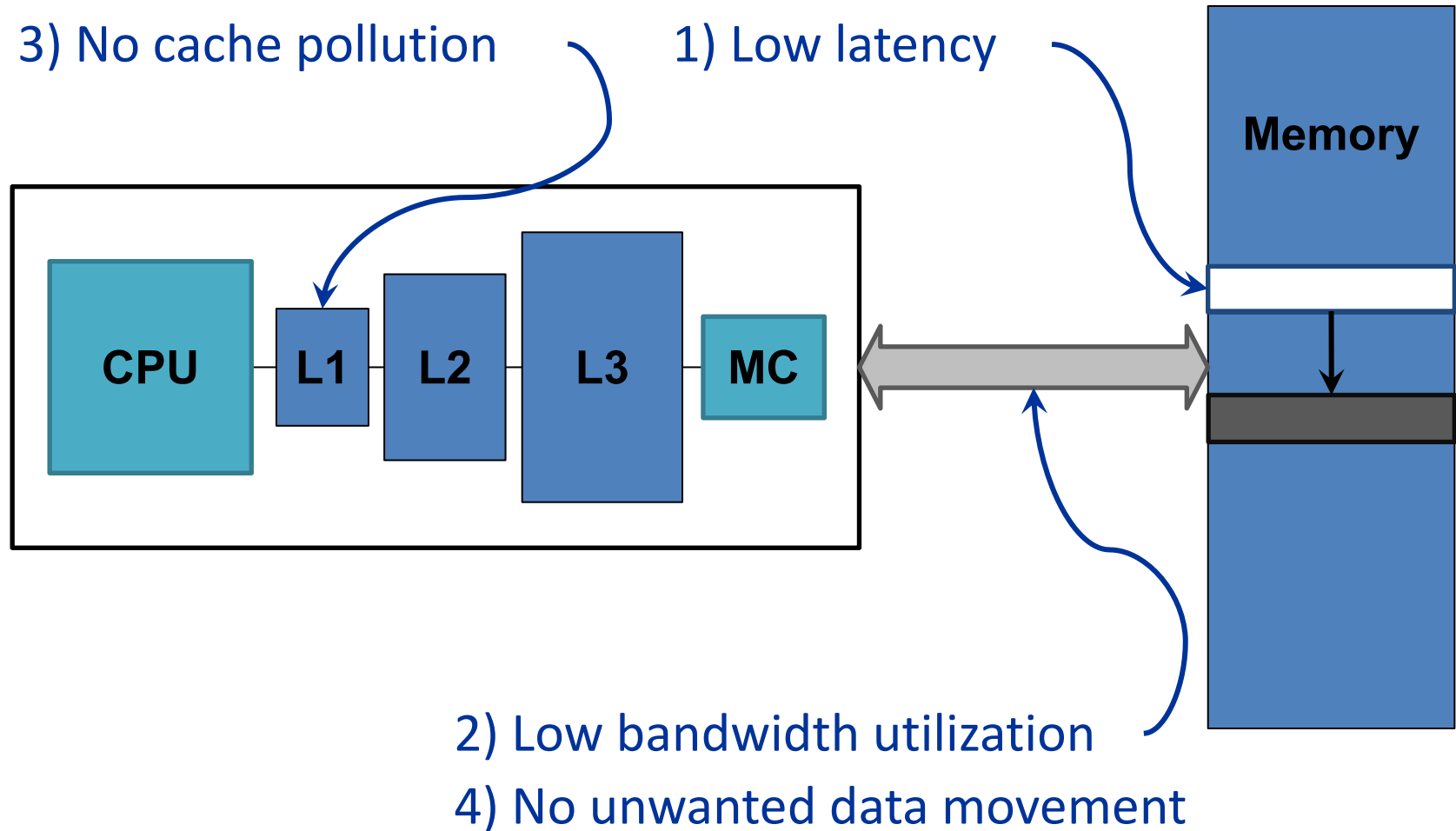**Zero initialization
(e.g., security)**

Checkpointing

**VM Cloning
Deduplication**

Page Migration

Many more

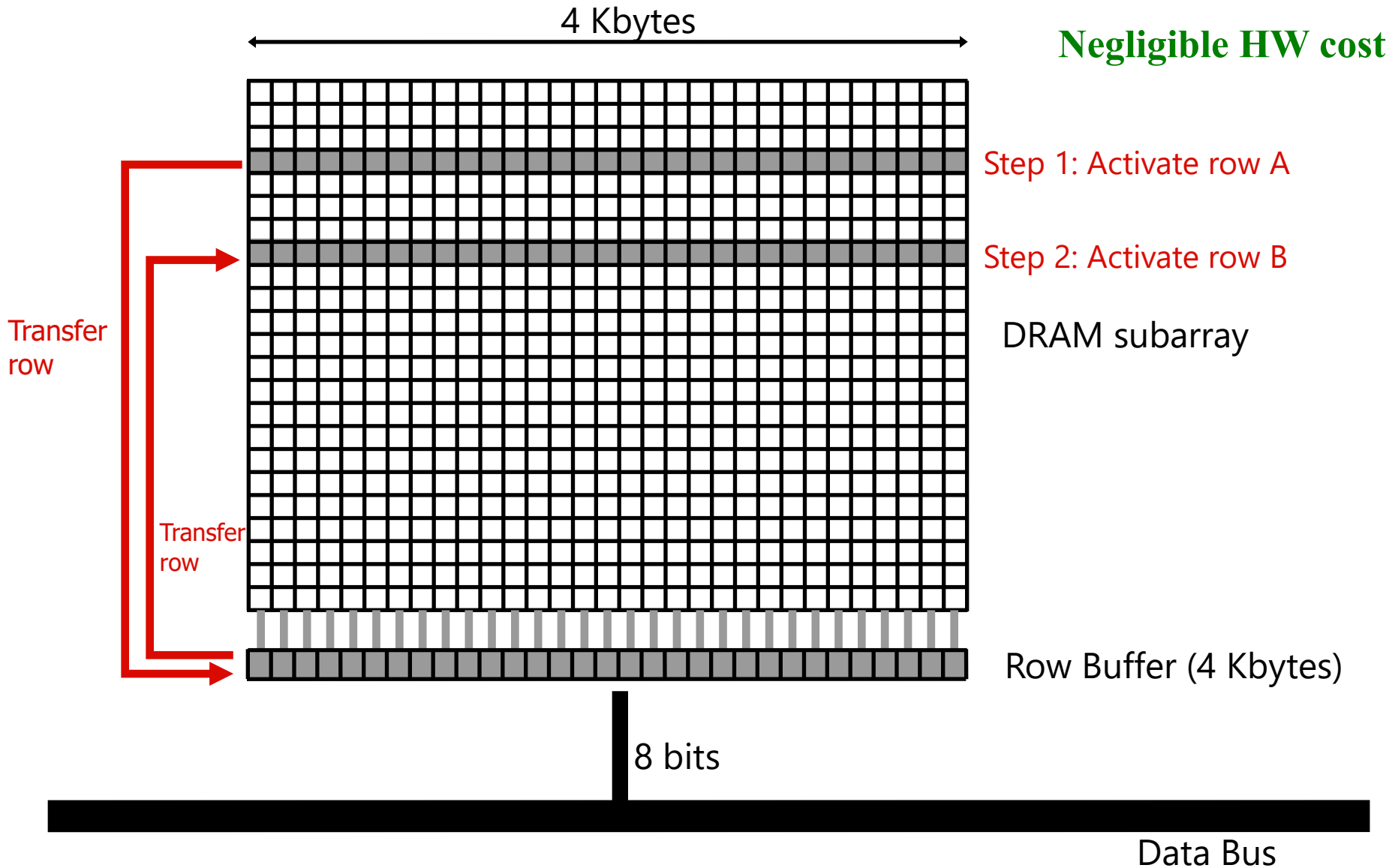**SAFARI**

Future Systems: In-Memory Copy

3) No cache pollution

1) Low latency

Memory

CPU **L1** **L2** **L3** **MC**

2) Low bandwidth utilization

4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ

RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost

4 Kbytes

Step 1: Activate row A

Step 2: Activate row B

DRAM subarray

Transfer row

Transfer row

Row Buffer (4 Kbytes)

8 bits

Data Bus

RowClone: Latency and Energy Savings
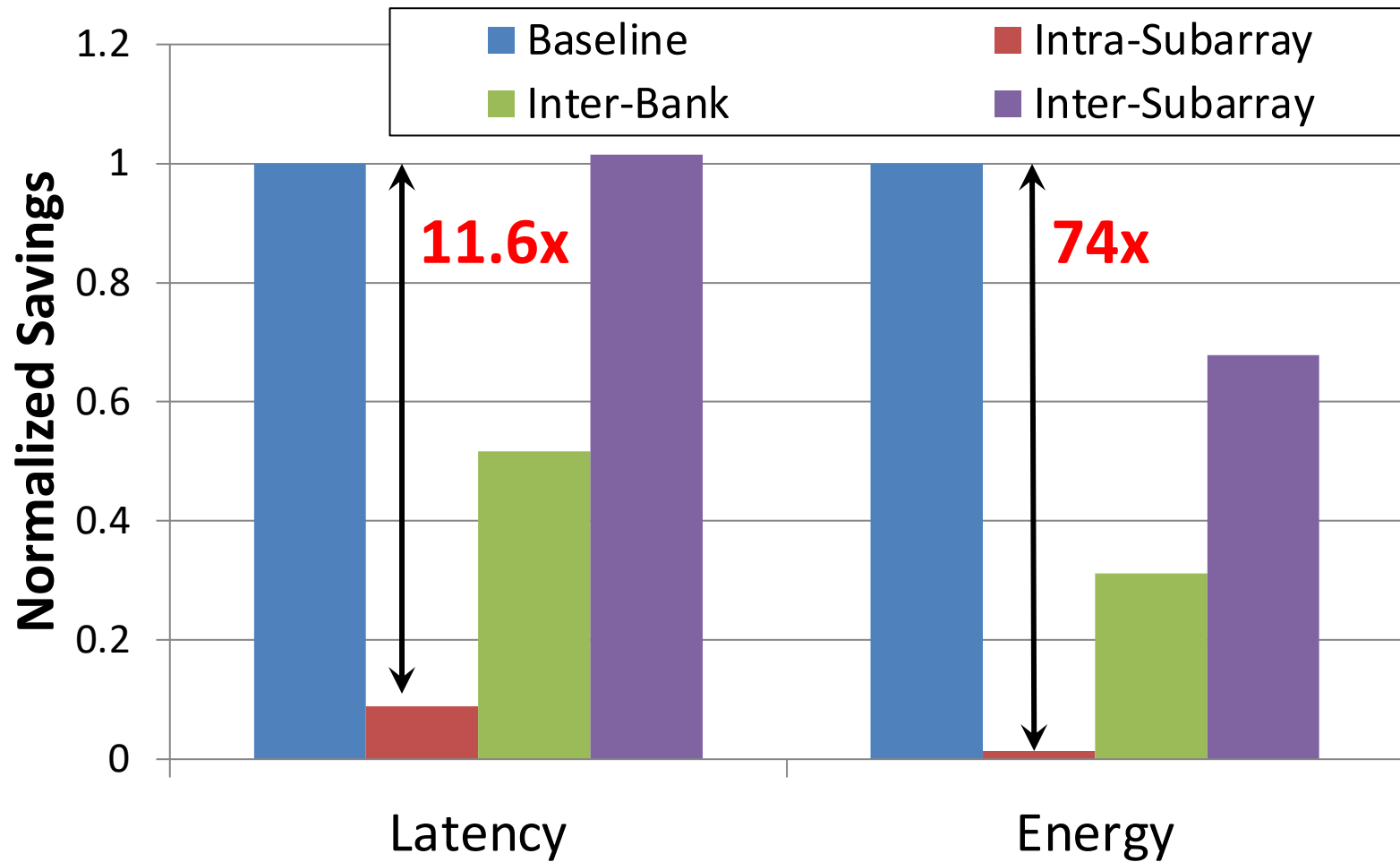
Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

81

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,
  "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"
  Proceedings of the 46th International Symposium on Microarchitecture (**MICRO**), Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri
vseshadr@cs.cmu.edu

Yoongu Kim
yoongukim@cmu.edu

Chris Fallin*
cfallin@c1f.net

Donghyuk Lee
donghyuk1@cmu.edu

Rachata Ausavarungnirun
rachata@cmu.edu

Gennady Pekhimenko
gpekhime@cs.cmu.edu

Yixin Luo
yixinluo@andrew.cmu.edu

Onur Mutlu
onur@cmu.edu

Phillip B. Gibbons†
phillip.b.gibbons@intel.com

Michael A. Kozuch†
michael.a.kozuch@intel.com

Todd C. Mowry
tcm@cs.cmu.edu

Carnegie Mellon University   †Intel Pittsburgh

RowClone in Off-the-Shelf DRAM Chips

- Idea: Violate DRAM timing parameters to mimic RowClone

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

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

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

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

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

83

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for <u>P</u>rocessing-<u>i</u>n-<u>DRAM</u>

Ataberk Olgun[§†]   Juan Gómez Luna[§]   Konstantinos Kanellopoulos[§]   Behzad Salami[§*]
Hasan Hassan[§]   Oğuz Ergin[†]   Onur Mutlu[§]

[§]ETH Zürich   [†]TOBB ETÜ   [*]BSC

https://arxiv.org/pdf/2111.00082.pdf
https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing-using-Memory Prototype

https://arxiv.org/pdf/2111.00082.pdf
https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing-using-Memory Prototype

README.md

Building a PiDRAM Prototype

To build PiDRAM's prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR's fpga-zynq repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM's memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPGAConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPGAConfig -j<number of cores>`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (system_top.bit) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx PHY IP we use in PiDRAM's memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:
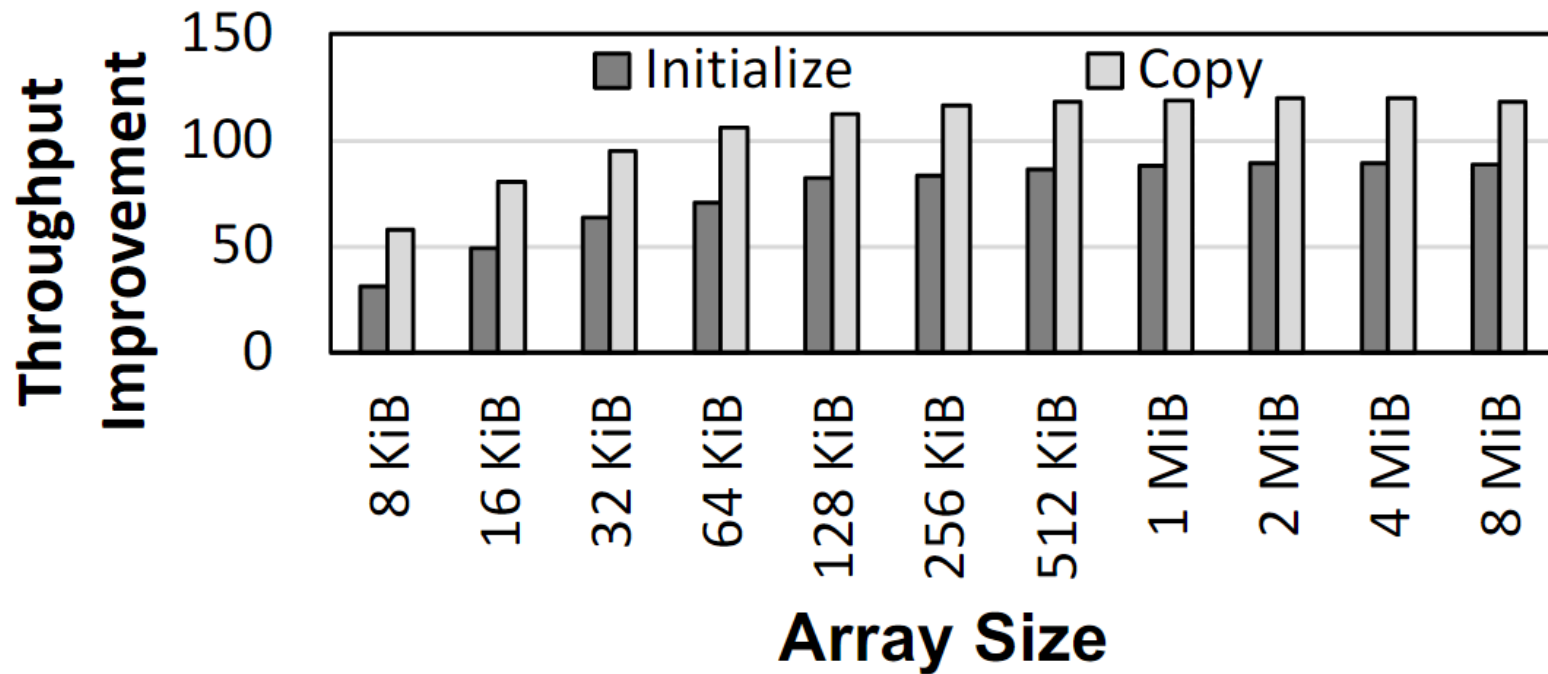
1- Open IP Catalog
2- Find "Memory Interface Generator (MIG 7 Series)" IP and double click

https://arxiv.org/pdf/2111.00082.pdf
https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Microbenchmark Copy/Initialization Throughput

In-DRAM Copy and Initialization
improve throughput by 119x and 89x

More on PiDRAM

- Ataberk Olgun, Juan Gomez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"
  ACM Transactions on Architecture and Code Optimization (**TACO**), March 2023.
  [arXiv version]
  Presented at the 18th HiPEAC Conference, Toulouse, France, January 2023.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (40 minutes)]
  [PiDRAM Source Code]

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun[§]   Juan Gómez Luna[§]   Konstantinos Kanellopoulos[§]   Behzad Salami[§]
Hasan Hassan[§]   Oğuz Ergin[†]   Onur Mutlu[§]

[§]ETH Zürich   [†]TOBB University of Economics and Technology

Lecture on RowClone & Processing using DRAM

Seminar in Computer Arch. - Meeting 3: RowClone: In-Memory Data Copy and Initialization (Fall 2021)

292 views • Streamed live on Oct 7, 2021

👍 21   👎 0   ↗ SHARE   ≡+ SAVE   ...

Onur Mutlu Lectures
19.1K subscribers   SUBSCRIBED   🔔

https://www.youtube.com/watch?v=n6Pwg1qax_E&list=PL5Q2soXY2Zi_7UBNmC9B8Yr5JSwTG9yH4&index=4

89

(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

$\frac{1}{2}V_{DD}+\delta$

A

B

C

dis

$\frac{1}{2}V_{DD}$
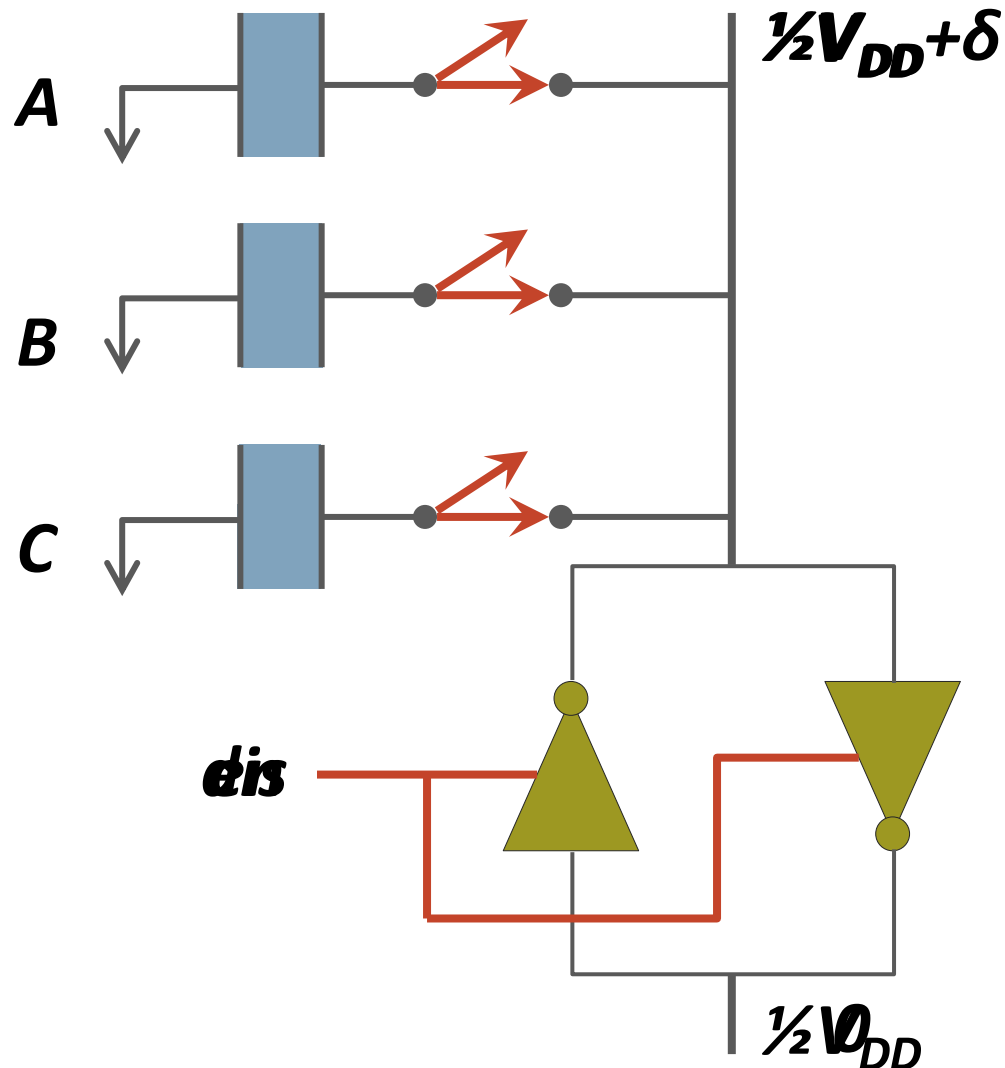
Final State
AB + BC + AC

C(A + B) +
~C(AB)

Bulk Bitwise Operations in Workloads

Bitmap indices
(database indexing)

BitWeaving
(database queries)

BitFunnel
(web search)

Bulk Bitwise Operations

Set operations

DNA sequence mapping

Encryption algorithms

...

[1] Li and Patel, BitWeaving, SIGMOD 2013
[2] Goodwin+, BitFunnel, SIGIR 2017

SAFARI
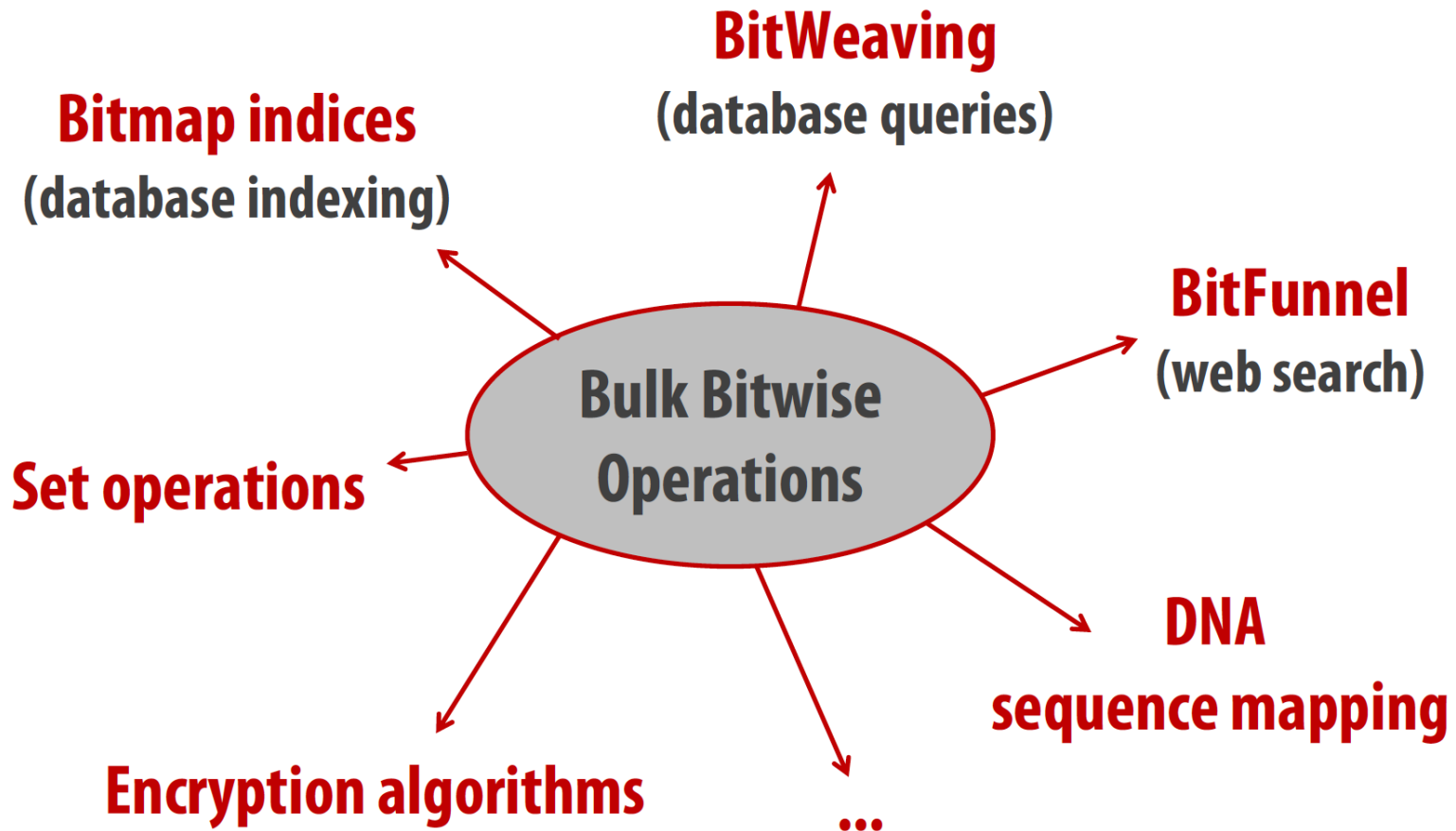
In-DRAM Acceleration of Database Queries

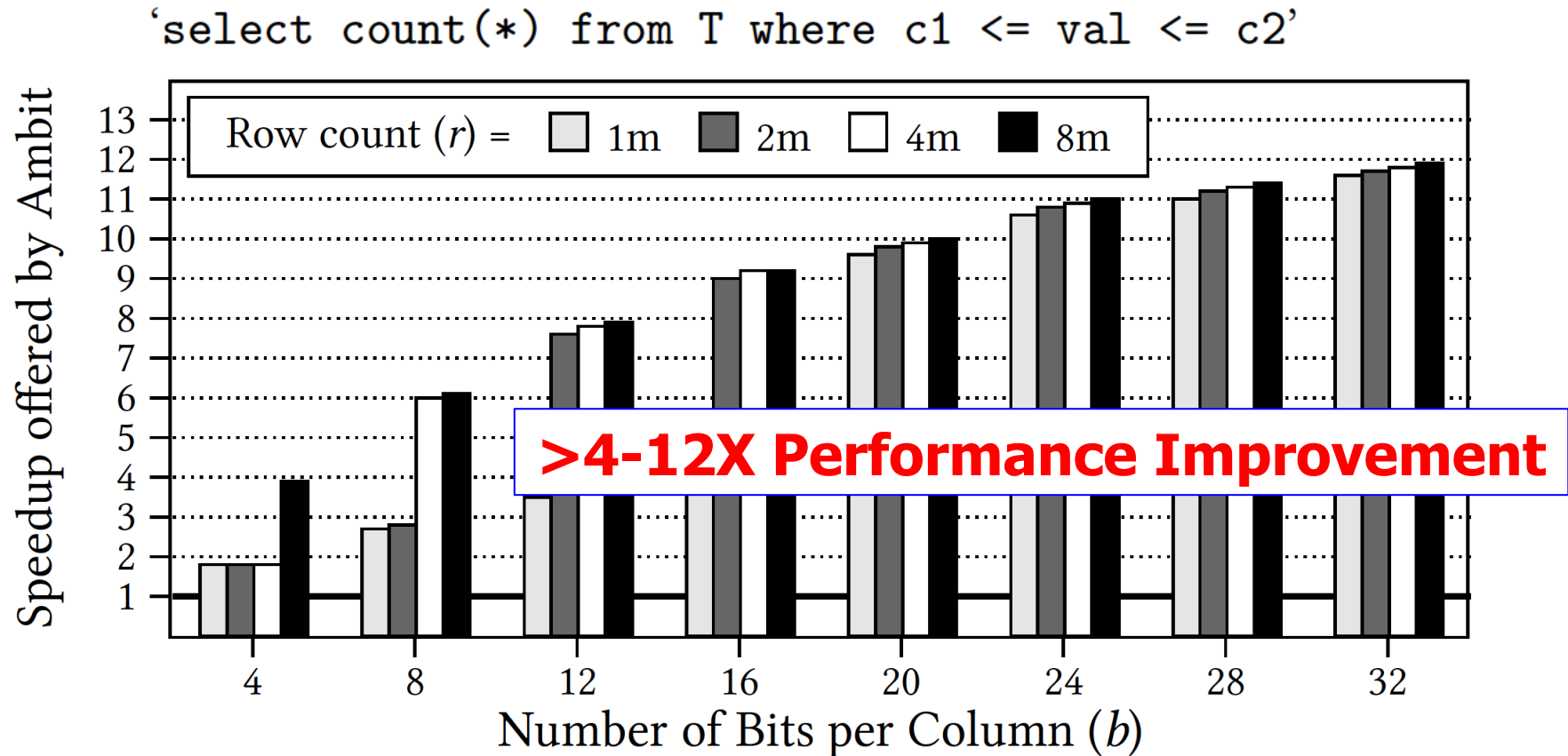**Figure 11:** Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

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 Hassan[4]   Amirali Boroumand[5]
Jeremie Kim[4,5]   Michael A. Kozuch[3]   Onur Mutlu[4,5]   Phillip B. Gibbons[5]   Todd C. Mowry[5]

[1]**Microsoft Research India**   [2]**NVIDIA Research**   [3]**Intel**   [4]**ETH Zürich**   [5]**Carnegie 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]

[1]ETH Zürich   [2]Simon Fraser University   [3]University of Illinois at Urbana–Champaign