Storage-Centric Computing

Enabling Fundamentally-Efficient Computers

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

1 December 2024

CCF China Storage Keynote Talk

SAFARI

ETH zürich

Problem

Computing

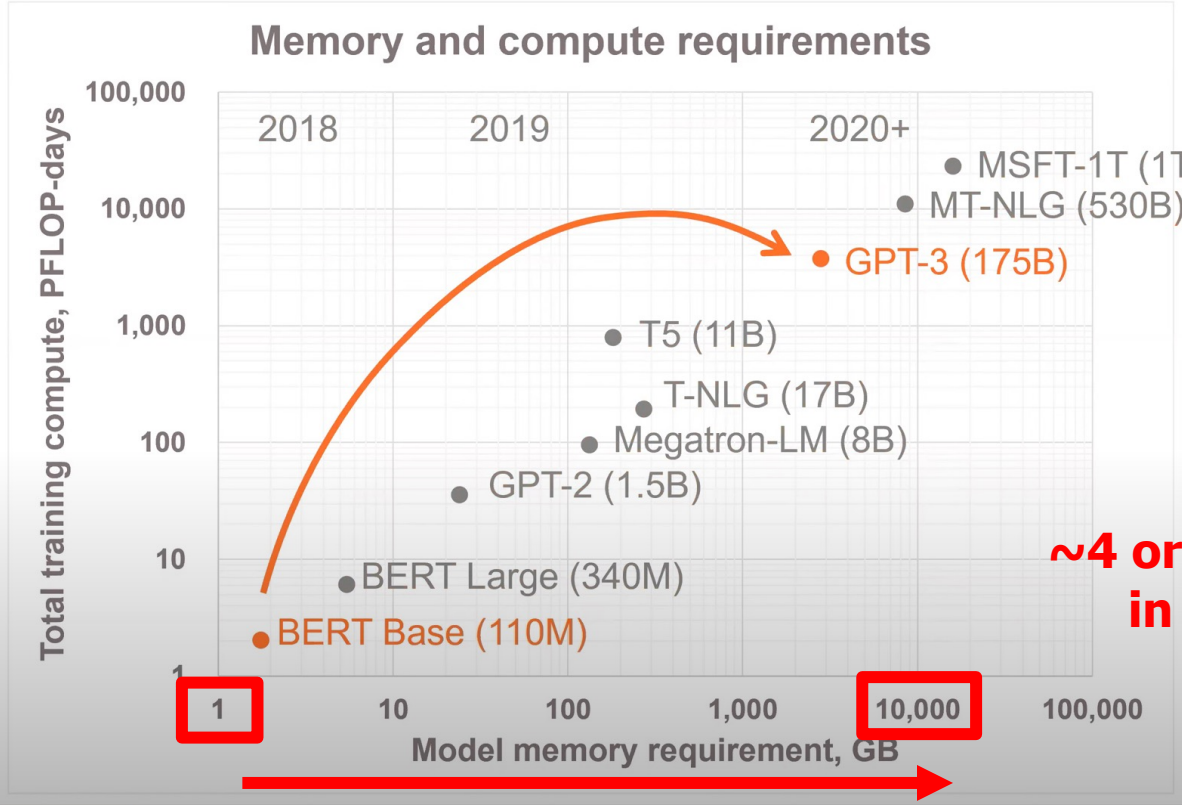
is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
  - We can generate more than we can process
  - We need to perform more sophisticated analyses on more data

Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks



1800x more compute

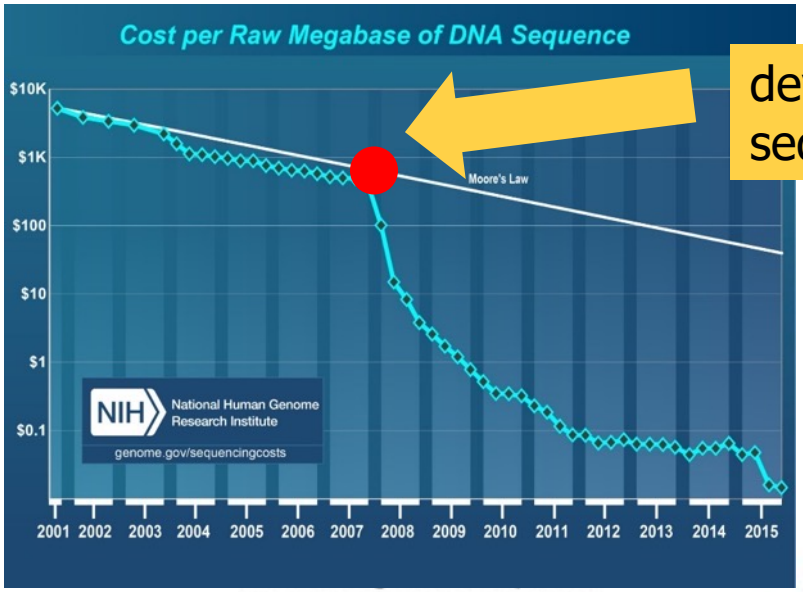
In just **2 years**

Tomorrow, **multi-trillion** parameter models

~4 orders of magnitude increase in memory requirement in just a few years!

Source: https://youtu.be/Bh13Idwcb0Q?t=283

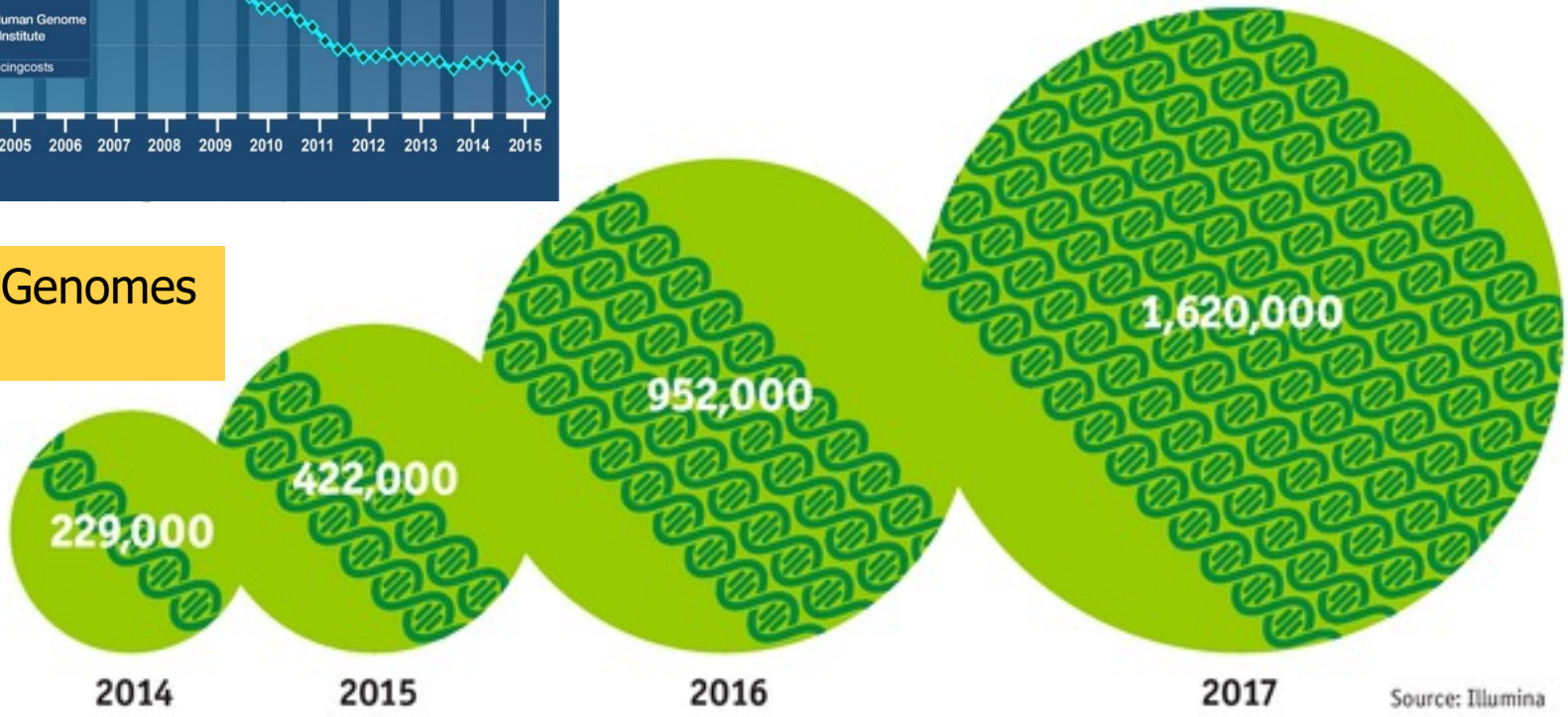
Huge Demand for Performance & Efficiency



development of new sequencing technologies

Oxford Nanopore MinION

Number of Genomes Sequenced



The Economist

http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped

Challenge and Opportunity for Future

High Performance,

Energy Efficient,

Sustainable

(All at the Same Time)

The Problem

Data access is the major performance and energy bottleneck

Our current

design principles

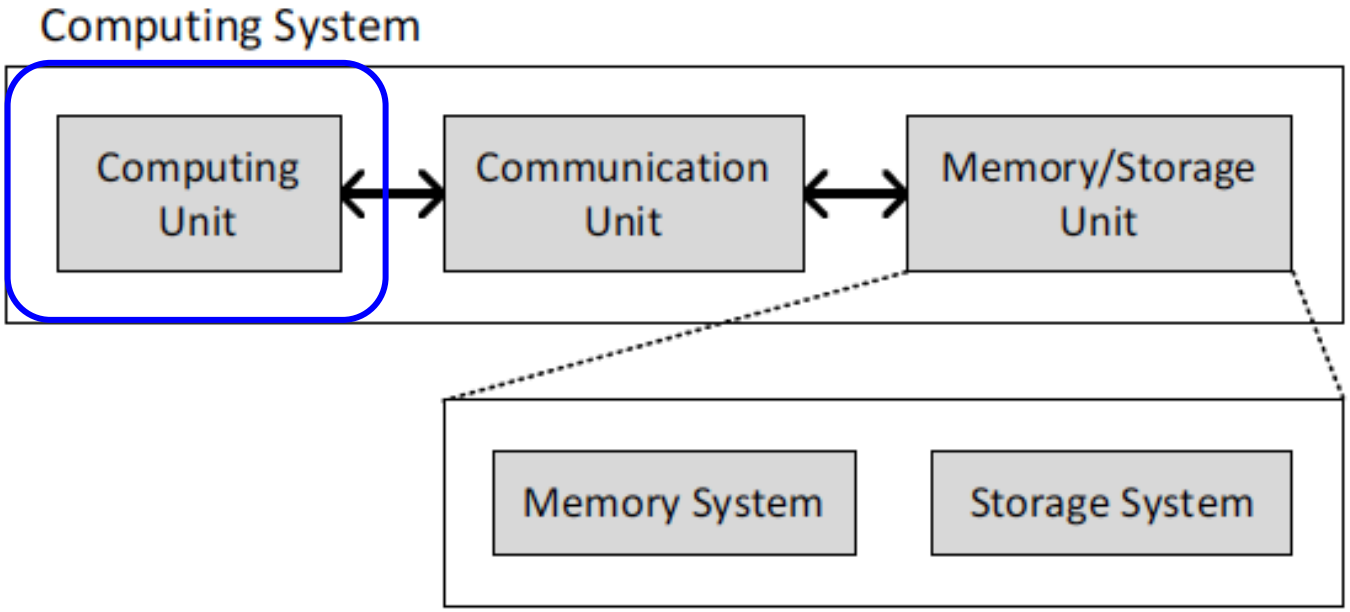
cause great energy waste

(and great performance loss)

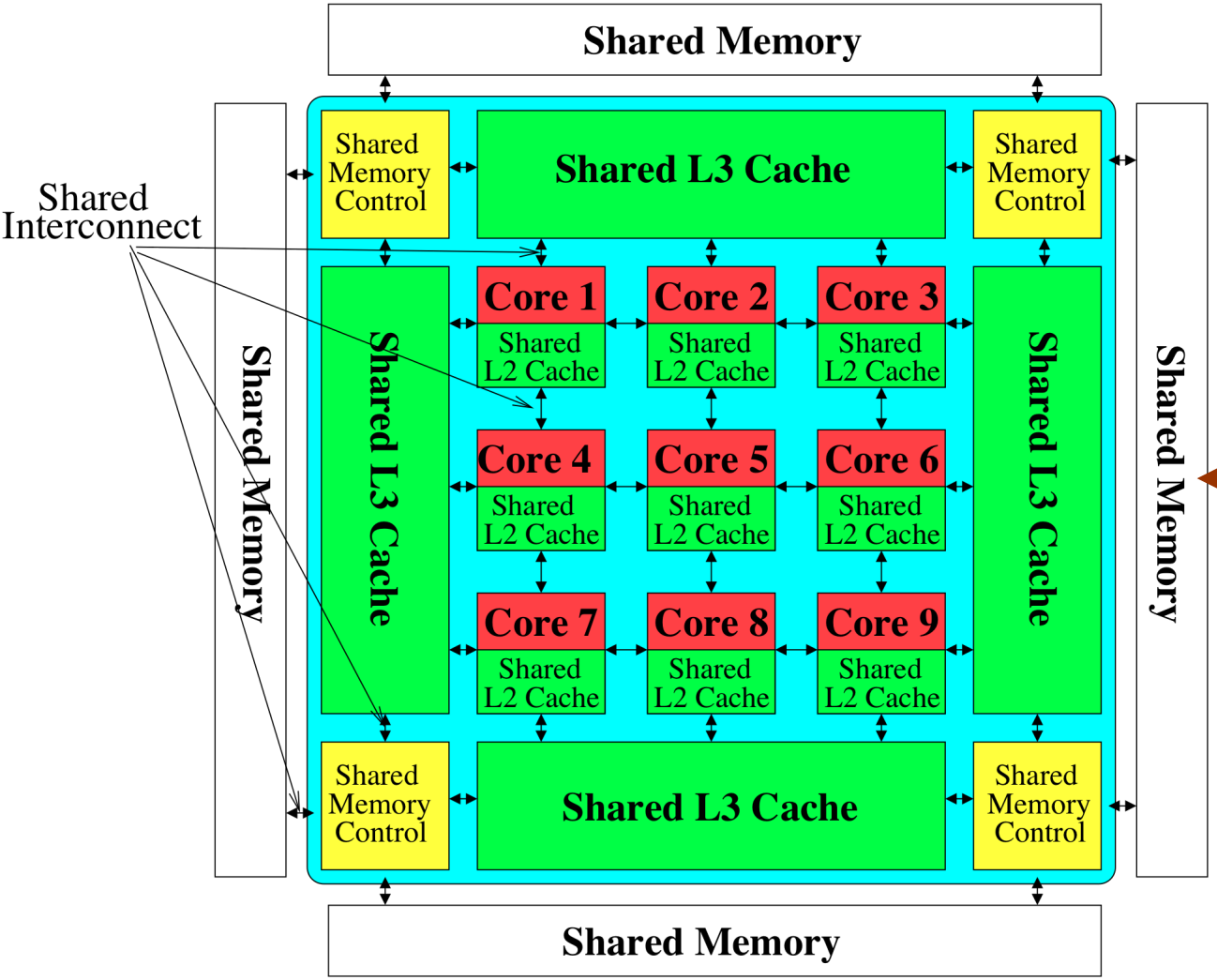
Today's Computing Systems

- Processor centric

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



Perils of Processor-Centric Design

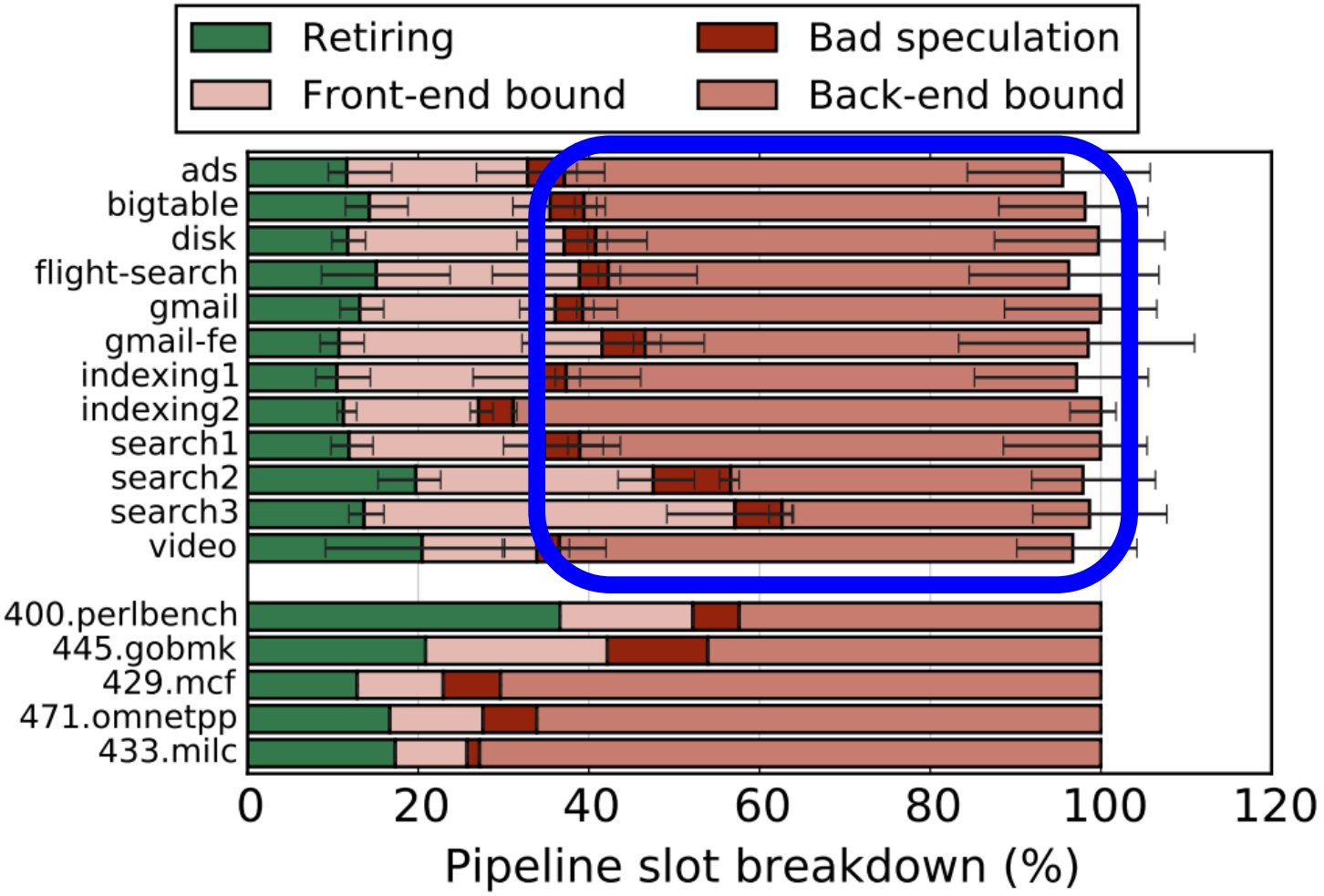


Most of the system is dedicated to storing and moving data

Yet, system is still bottlenecked by memory & storage

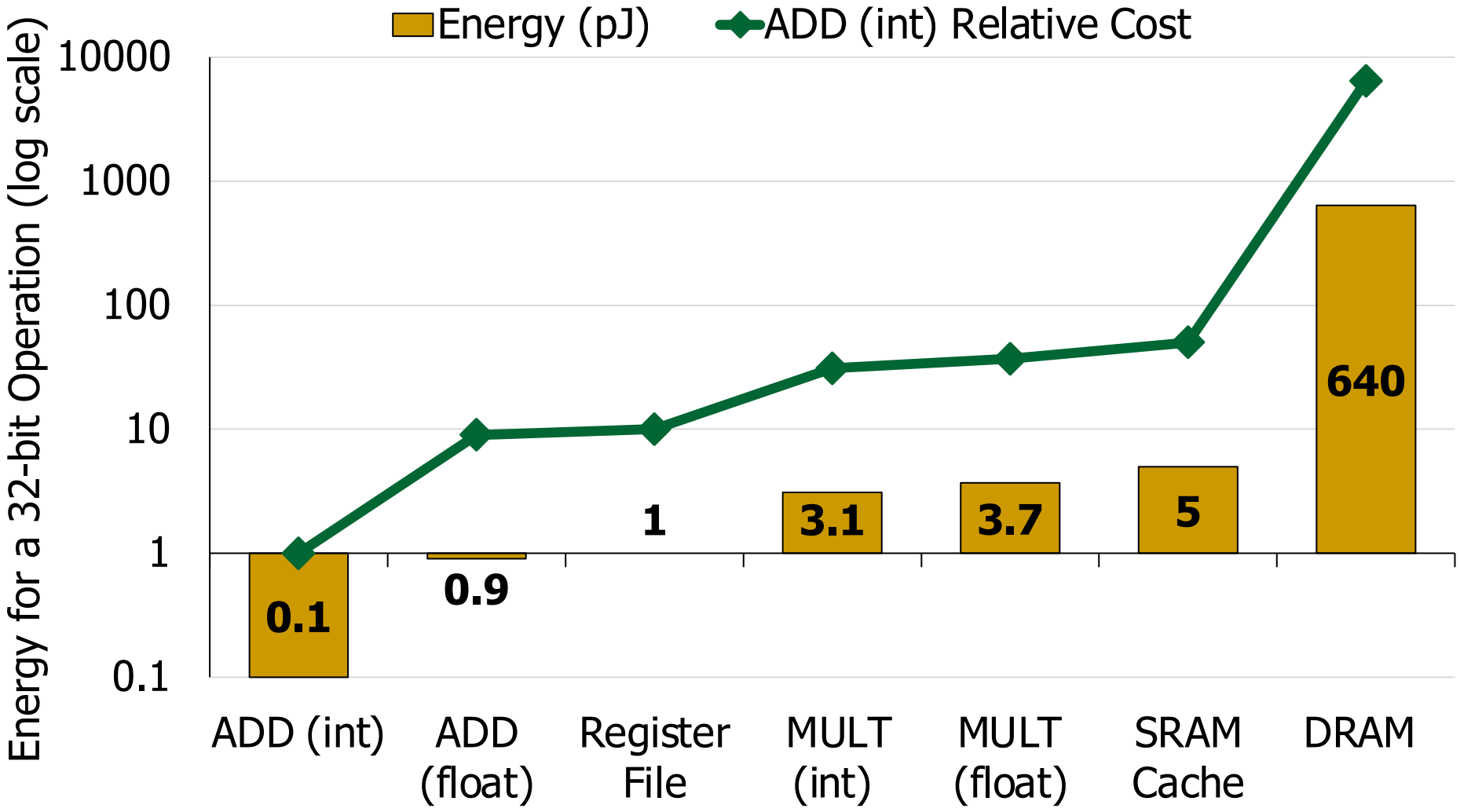
Processor-Centric System Performance

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



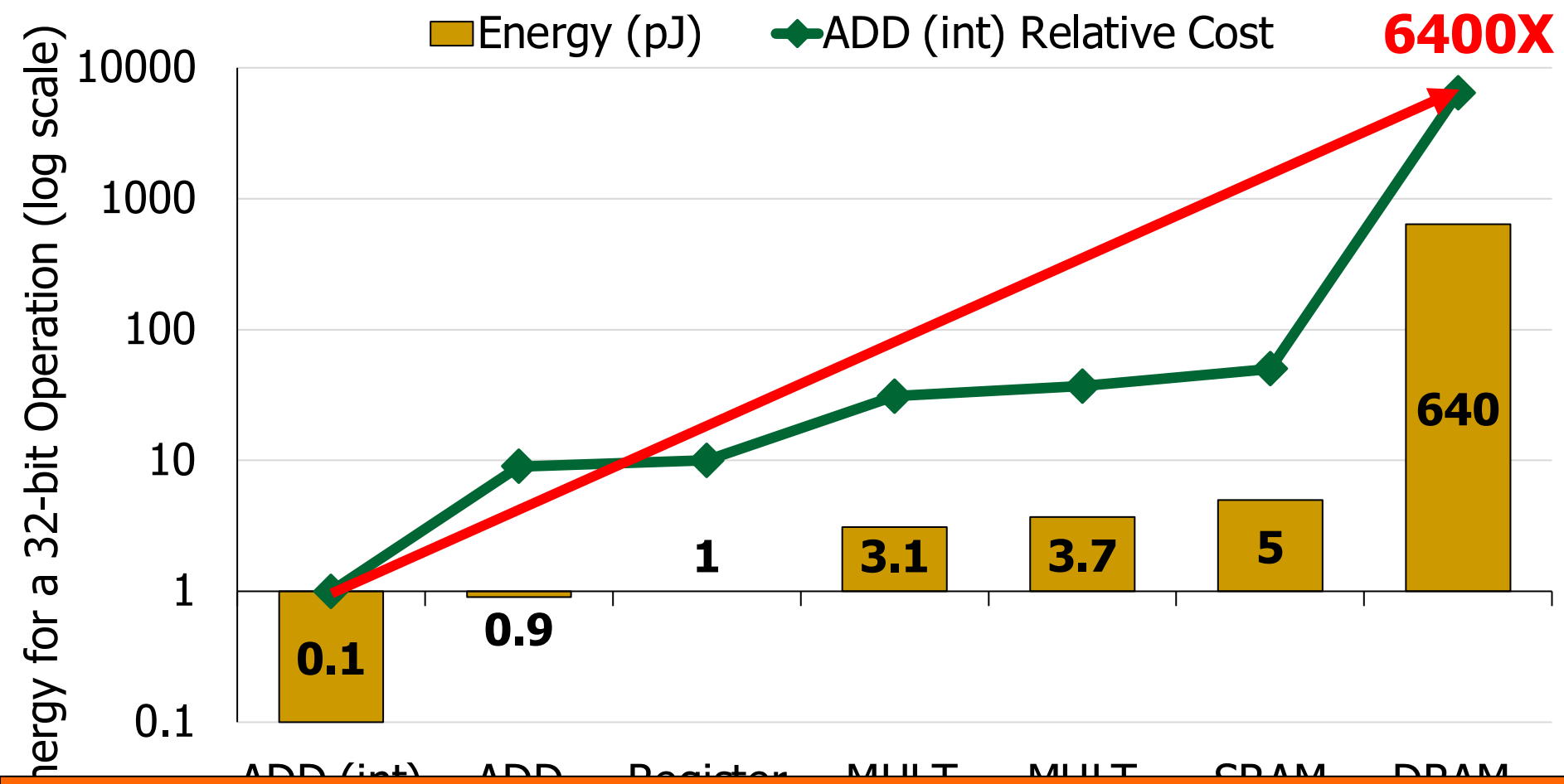
Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

Data Movement vs. Computation Energy



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]

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)]

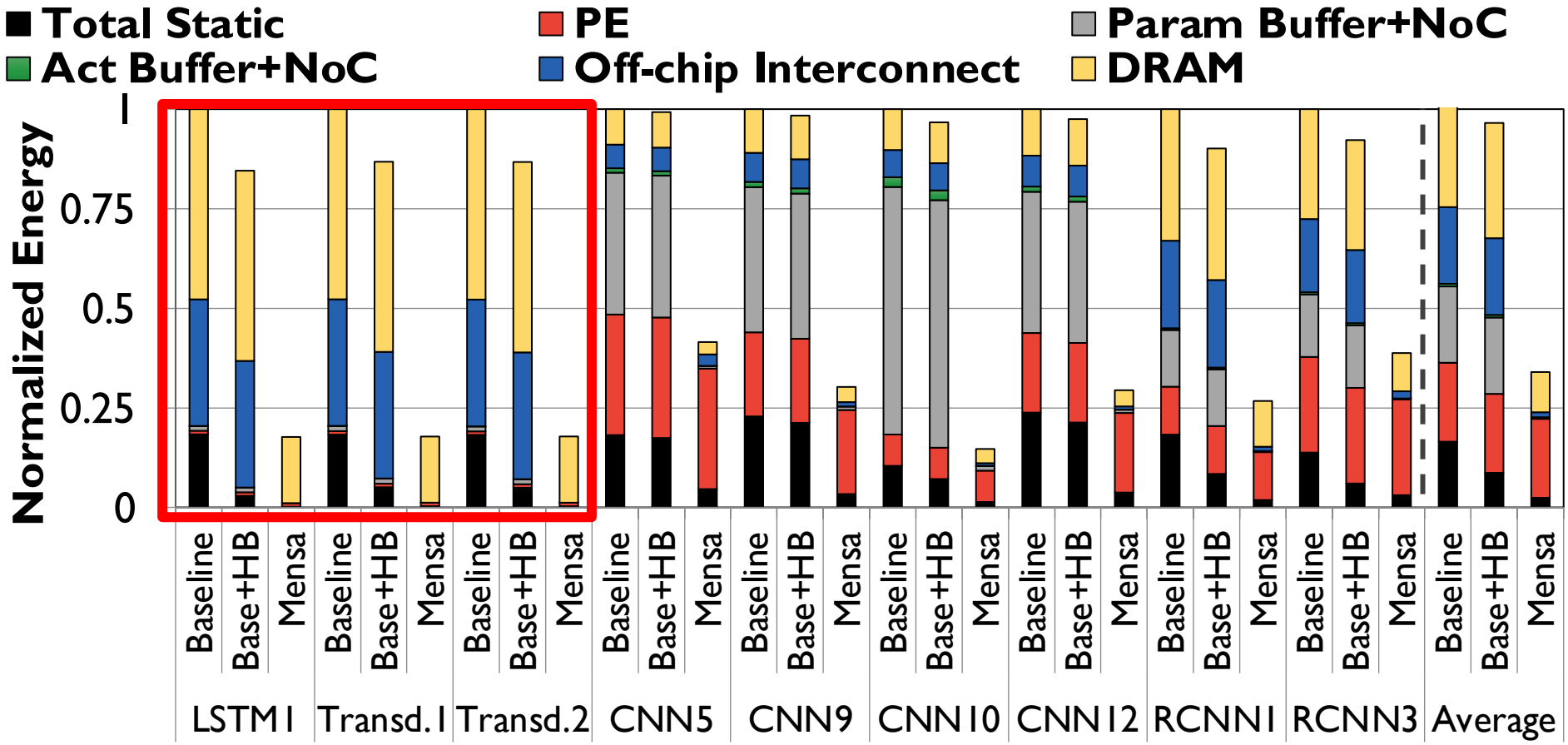
> 90% of the total system energy is spent on memory in large ML models

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*

Example Energy Breakdowns



In LSTMs and Transducers used by Google,
>90% energy spent on off-chip interconnect and DRAM

https://arxiv.org/pdf/2109.14320

Fundamental Problem

Processing of data

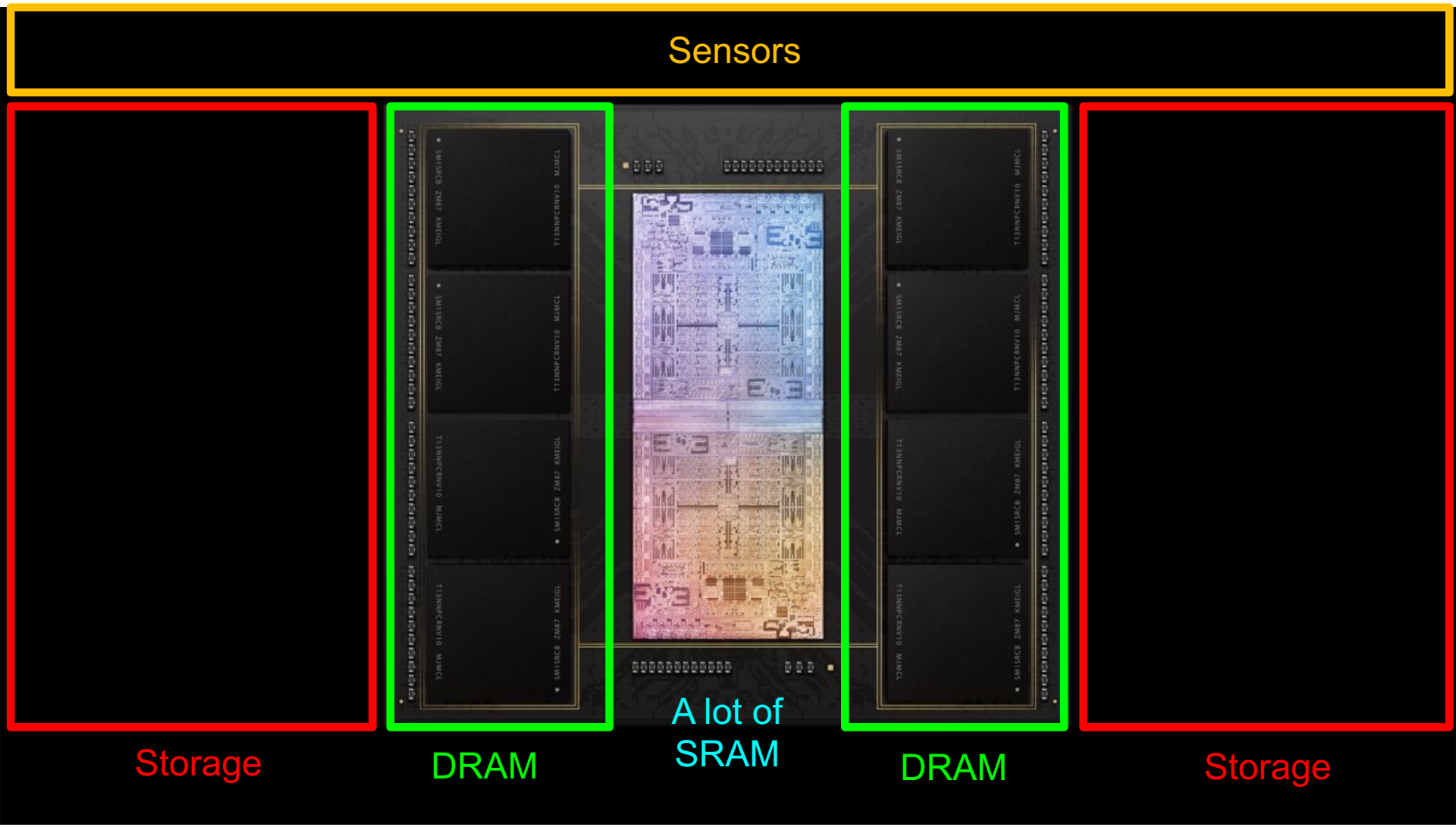
is performed

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

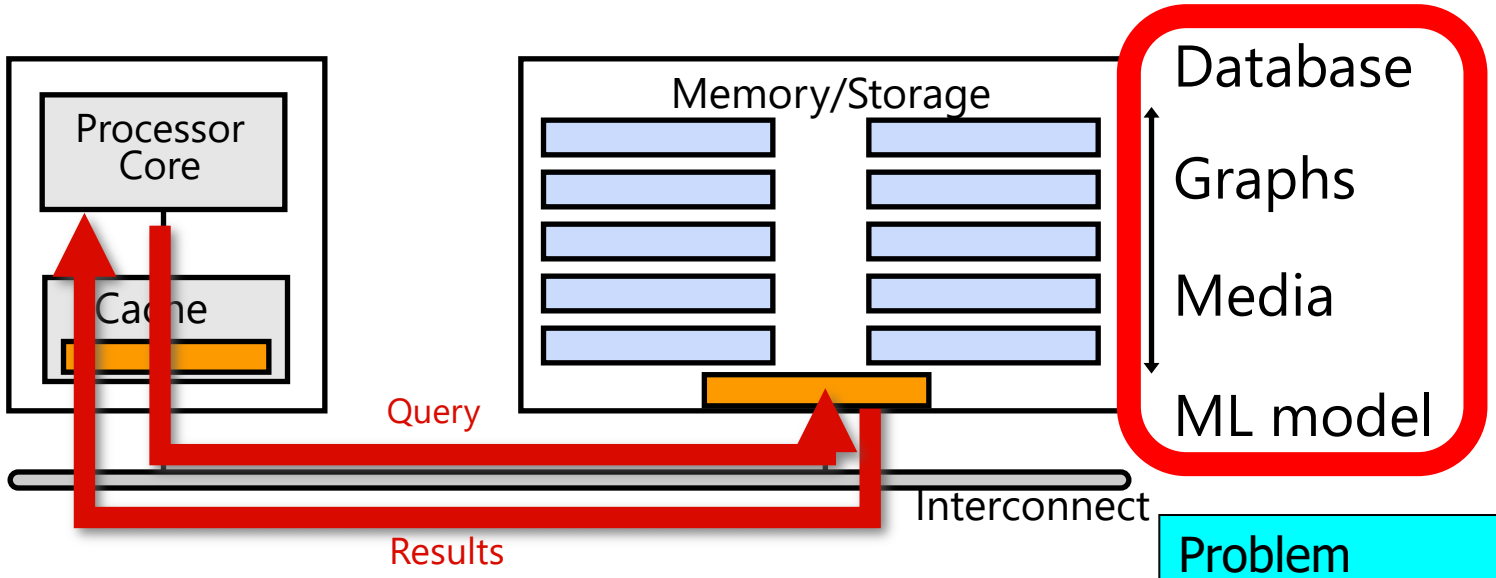
Process Data Where It Makes Sense



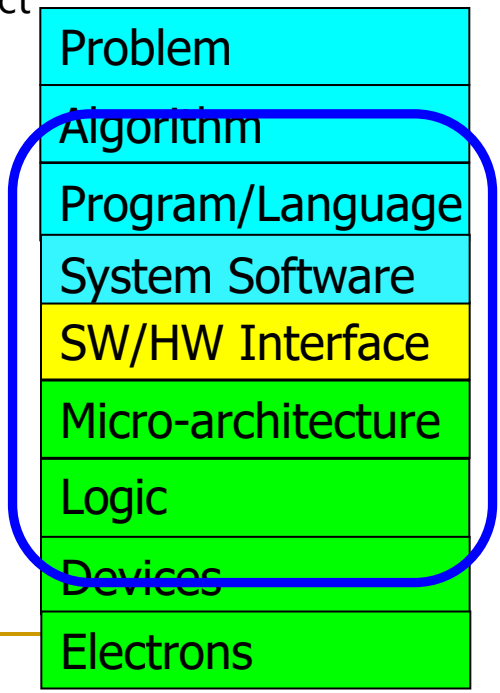
Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php

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?



An Overview Paper

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.

https://arxiv.org/pdf/2012.03112.pdf

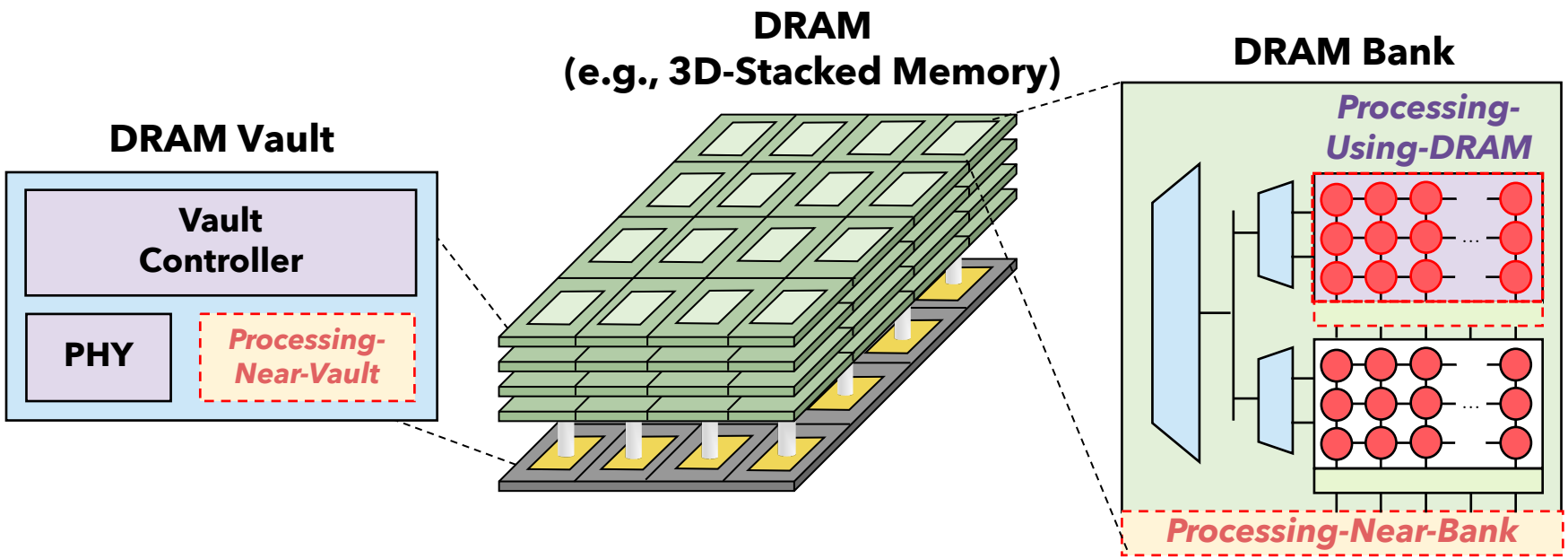
Processing in Storage: Two Types

1. Processing **near** Storage Devices
2. Processing **using** Storage Devices

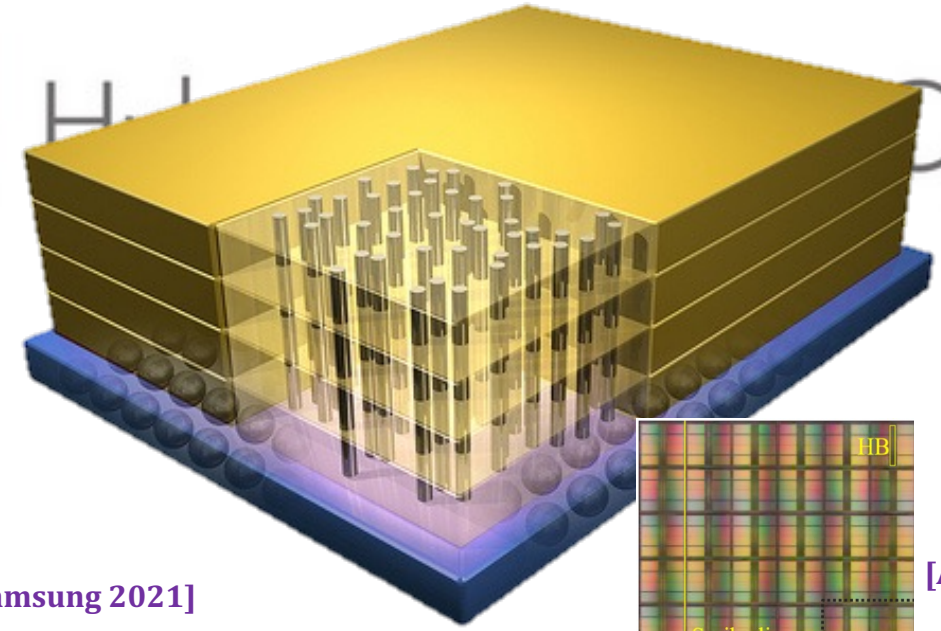
Processing-in-Memory: Two Types

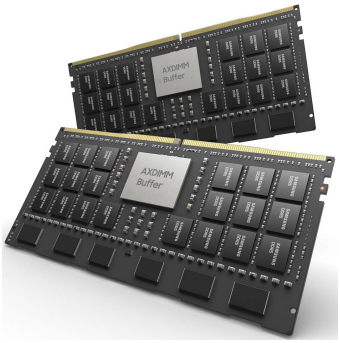
Two main approaches for Processing-in-Memory:

1. **Processing-Near-Memory**: Computation logic is added to the same die as memory or to the logic layer of 3D-stacked memory
2. **Processing-Using-Memory**: uses the operational principles of memory cells & circuitry to perform computation



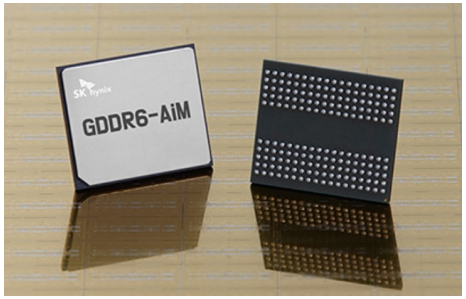
Processing-in-Memory Landscape Today



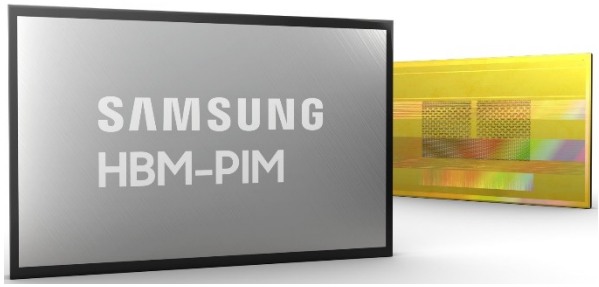


[Samsung 2021]

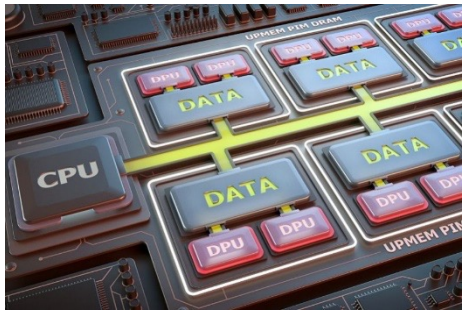
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]



[UPMEM 2019]

And, many other experimental chips and startups

Processing-in-Memory Landscape Today

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

Computational CXL-Memory Solution for Accelerating Memory-Intensive Applications

Joonseop Sim, Soohong Ahn, Taeyoung Ahn, Seungyong Lee, Myunghyun Rhee, Jooyoung Kim, Kwangsik Shin, Donguk Moon, Euiseok Kim, and Kyoung Park

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× better performance/power efficiency than the existing CPU system.

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

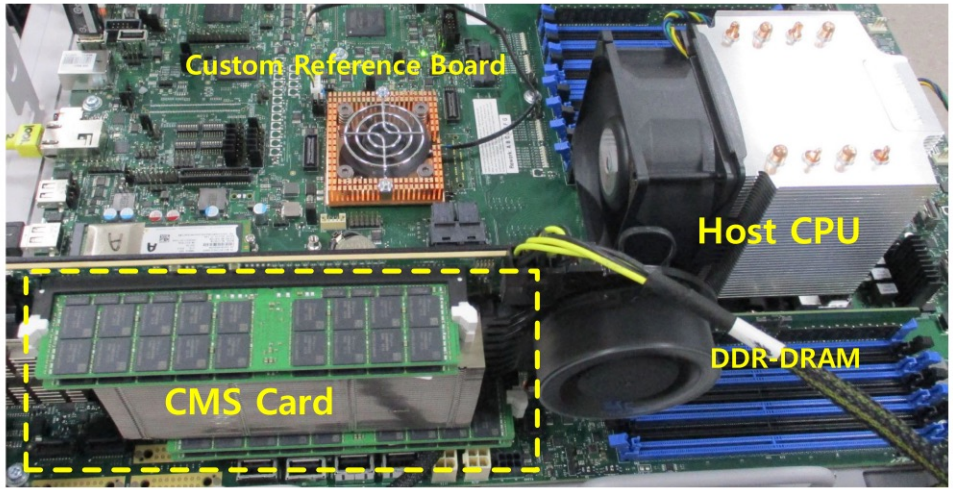


Fig. 6. FPGA prototype of proposed CMS card.

Processing-in-Memory Landscape Today

Samsung Processing in Memory Technology at Hot Chips 2023

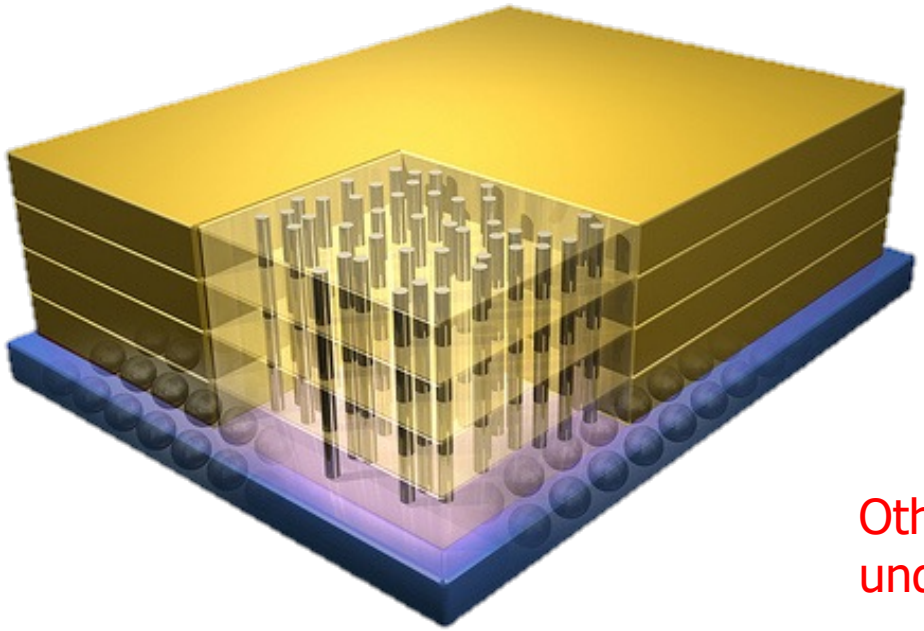
By **Patrick Kennedy** - August 28, 2023

Samsung PIM PNM For Transformer Based AI HC35_Page_24

https://www.servethehome.com/samsung-processing-in-memory-technology-at-hot-chips-2023/

Opportunity: 3D-Stacked Logic+Memory

Hybrid Memory Cube
CONSORTIUM



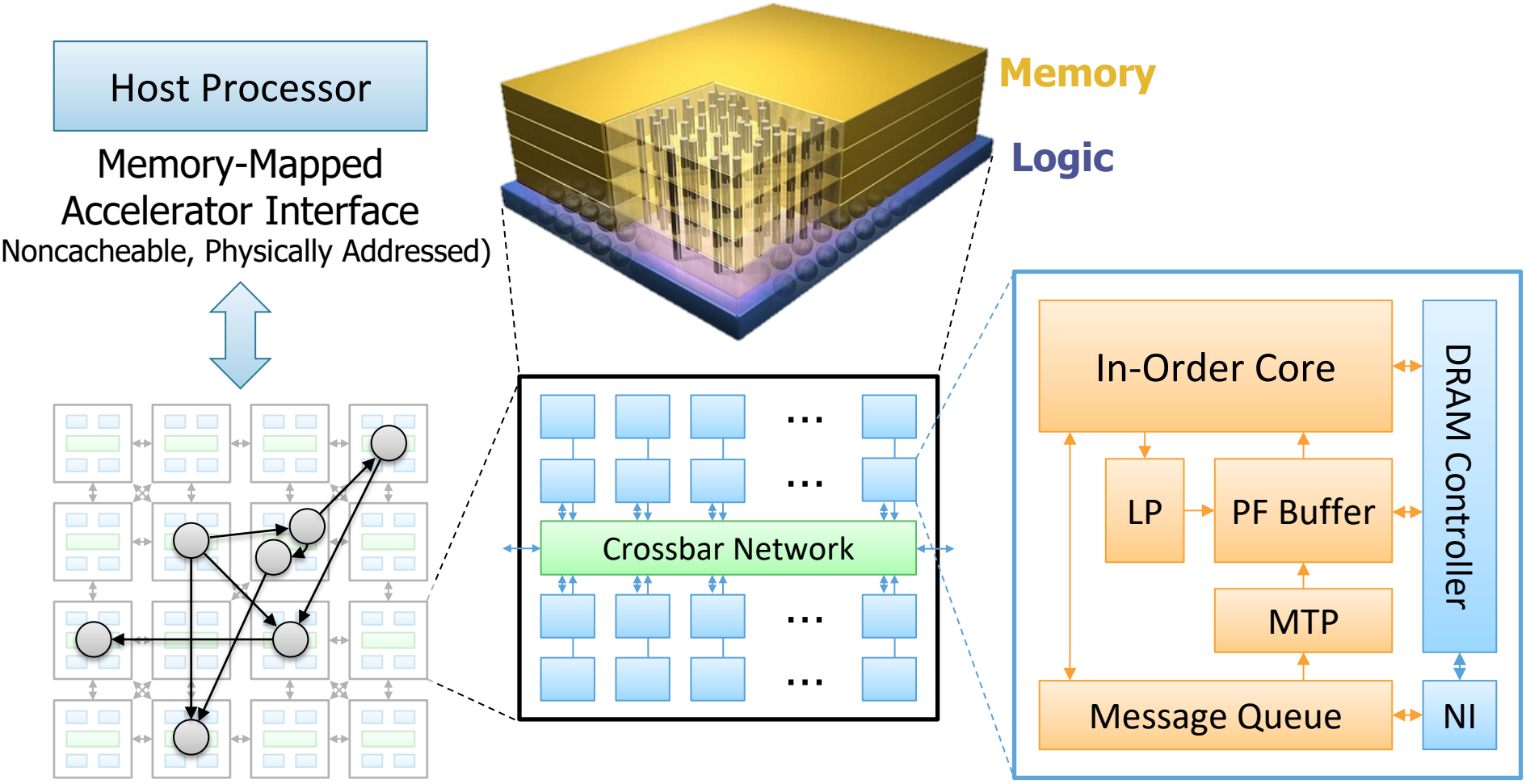
Memory

Logic

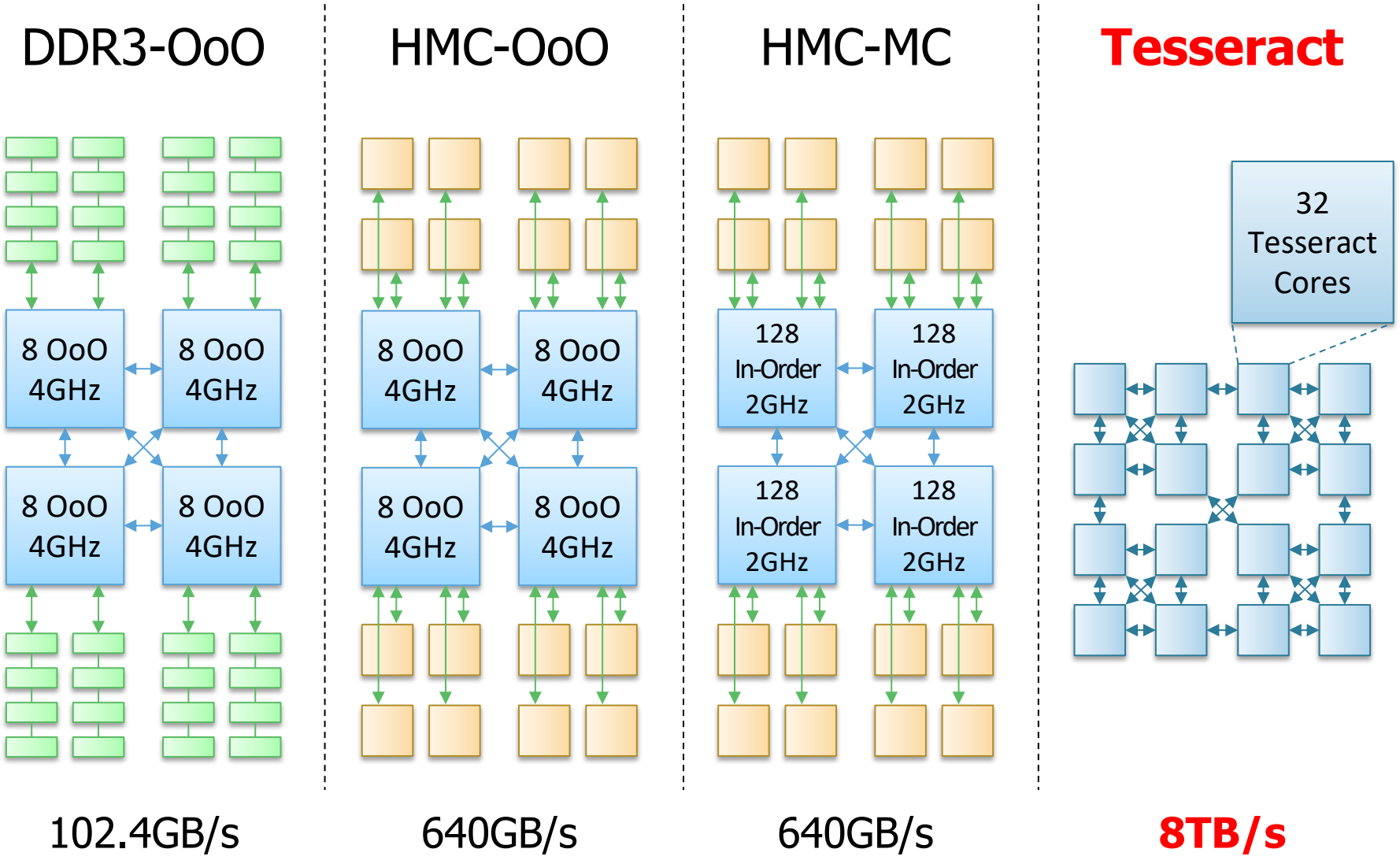
Other "True 3D" technologies under development

Tesseract System for Graph Processing

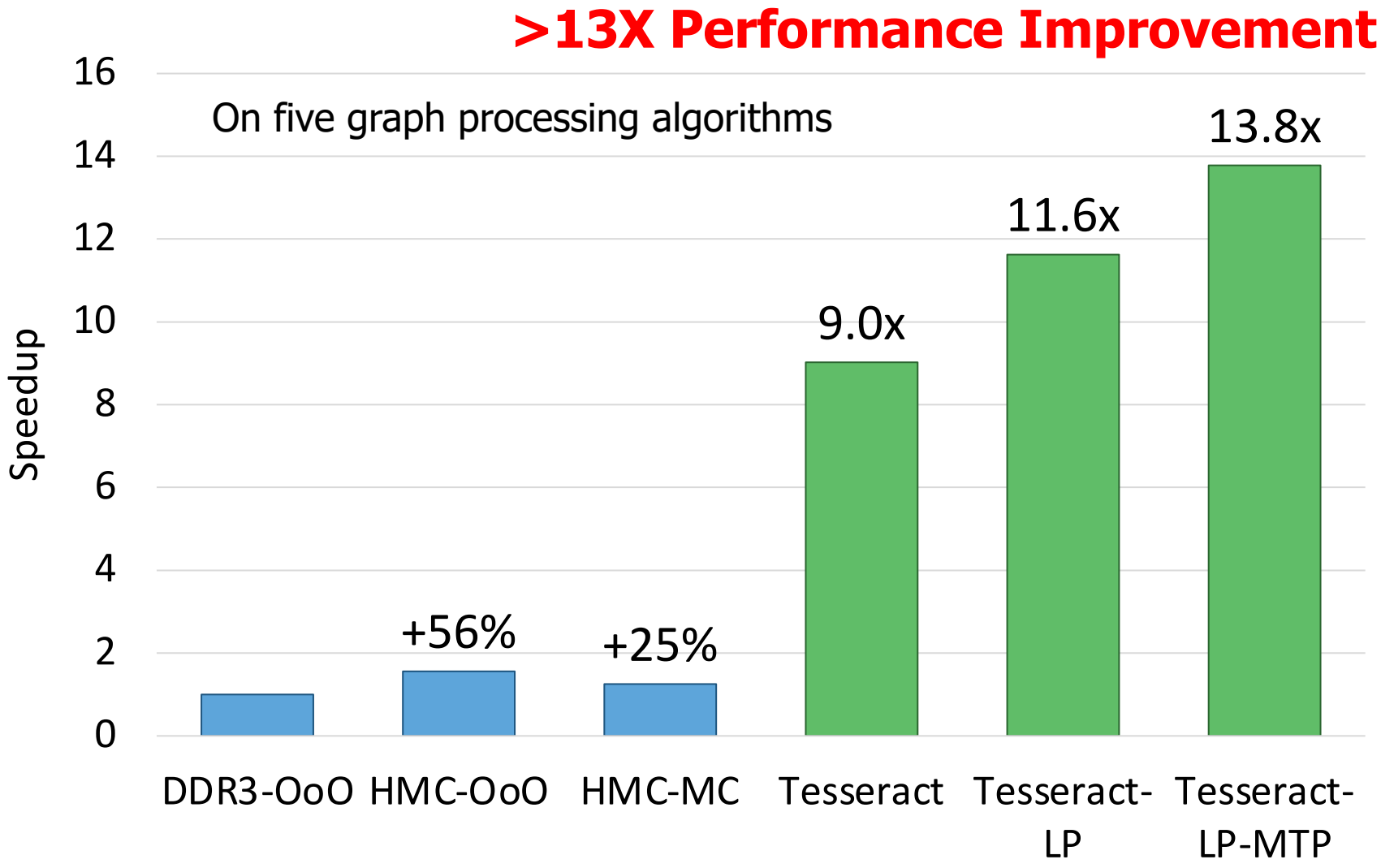
Interconnected set of 3D-stacked memory+logic chips with simple cores



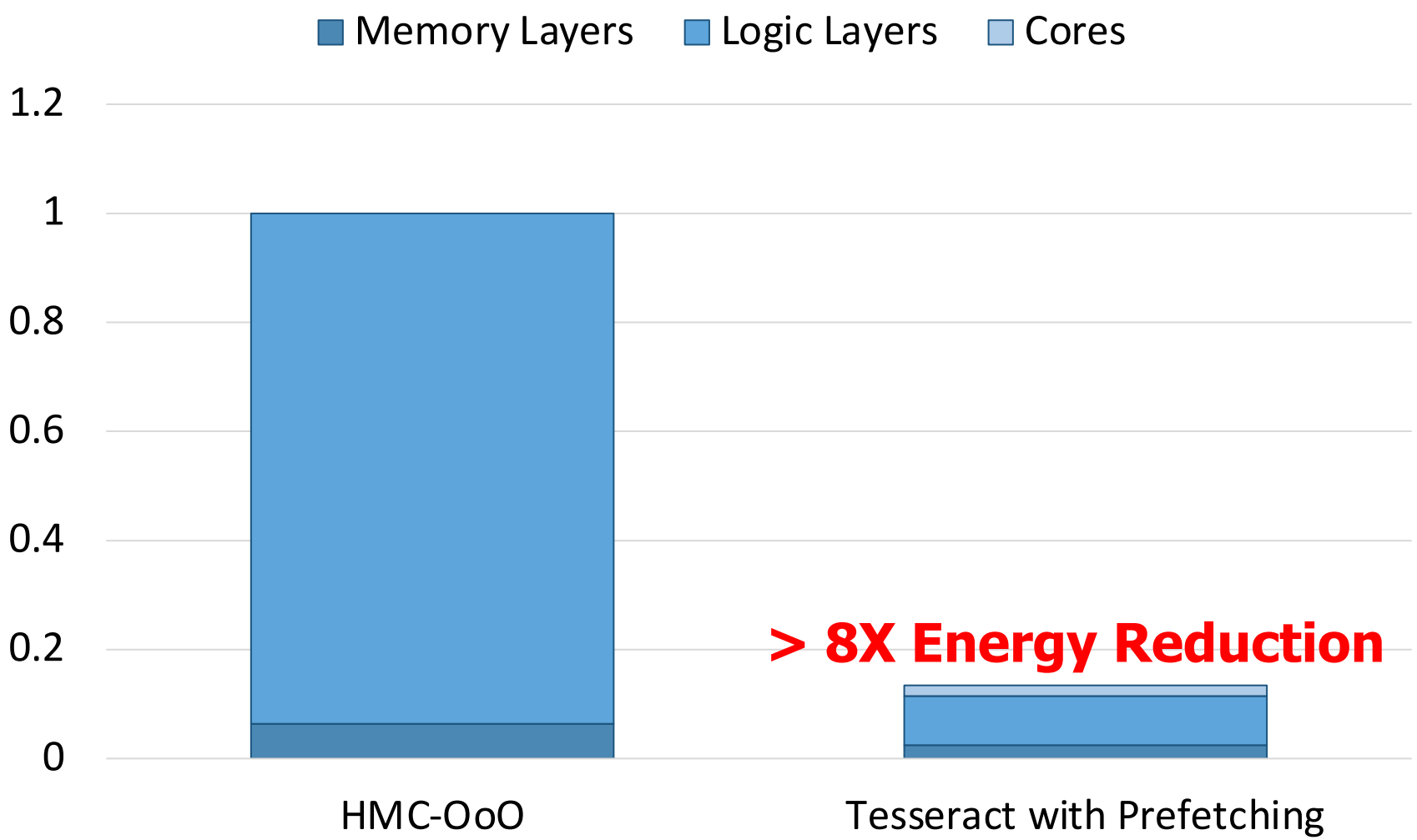
Evaluated Systems



Tesseract Graph Processing Performance



Tesseract Graph Processing System Energy



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

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 cores communicate with each other 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. Ensuing 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 become more constrained due to the continuously-increasing 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. Jeddeloh 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 DOJO, 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] J. Ahn *et al.*, "AIM: Energy-Efficient Aggregation Inside the Memory Hierarchy," *ACM TACO*, 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/2012.03112.
[16] M. Zhang *et al.*, "GraphP: 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] Y. Zhuo *et al.*, "GraphQ: Scalable PIM-Based Graph Processing," in *MICRO*, 2019.
[19] G. Dai *et al.*, "GraphH: A Processing-in-Memory Architecture for Large-Scale Graph Processing," *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.*, "NEMESYS: Near-Memory Graph Copy Enhanced System-Software," in *MEMSYS*, 2019.
[22] L. Belayneh and V. Bertacco, "GraphVine: Exploiting Multicast for Scalable Graph Analytics," in *DATE*, 2020.
[23] N. Challapalle *et al.*, "GaaS-X: Graph Analytics Accelerator Supporting Sparse Data Representation using Crossbar Architectures," in *ISCA*, 2020.
[24] M. Zhou *et al.*, "Ultra Efficient Acceleration for De Novo Genome Assembly via Near-Memory Computing," in *PACT*, 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 *DATE*, 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 TFLOPS 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," *IEEE Micro*, 2021.
[36] I. Oliveira, "Improving In-Memory Database Operations with Acceleration DIMM (AXDIMM)," in *DaMoN*, 2022.
[37] S. Lee *et al.*, "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," in *ISSCC*, 2022.
[38] D. Niu *et al.*, "184QPS/W 64Mb/mm^2 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 TRETS*, 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] M. Besta *et al.*, "SISA: Set-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.

https://arxiv.org/pdf/2306.16093

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

[1]ETH Zurich, Switzerland [2]IIT Tirupati, India [3]King Mongkut's University of Technology North Bangkok, Thailand [4]Technical University of Ostrava, Czech Republic [5]AGH-UST, Poland

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

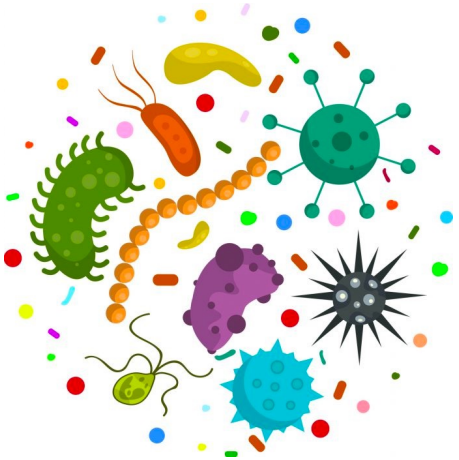
Nika Mansouri Ghiasi[1] Jisung Park[1] Harun Mustafa[1] Jeremie Kim[1] Ataberk Olgun[1]
Arvid Gollwitzer[1] Damla Senol Cali[2] Can Firtina[1] Haiyu Mao[1] Nour Almadhoun Alserr[1]
Rachata Ausavarungnirun[3] Nandita Vijaykumar[4] Mohammed Alser[1] Onur Mutlu[1]

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

https://arxiv.org/abs/2202.10400

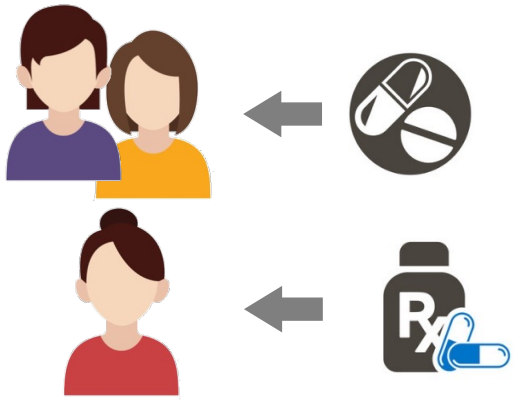
We Need Faster & Scalable Genome Analysis

Understanding **genetic variations, species, evolution, …**



Predicting the presence and relative abundance of **microbes** in a sample

Rapid surveillance of **disease outbreaks**



Developing **personalized medicine**

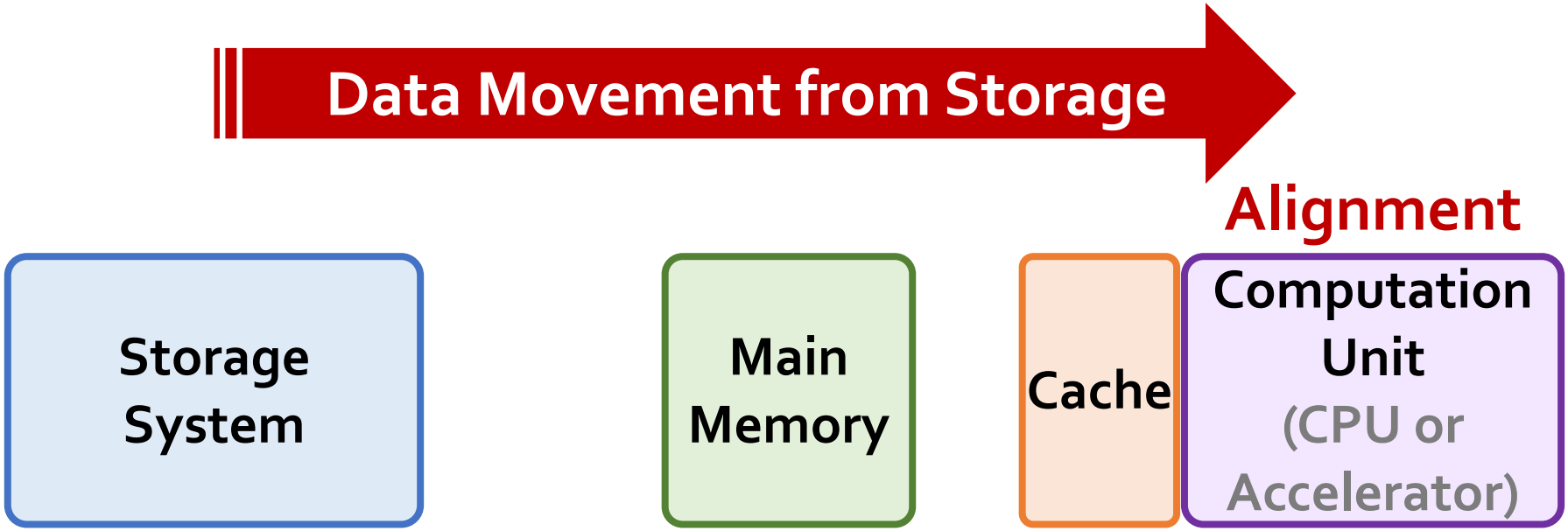
And, many, many other applications …

Genome Sequence Analysis

- **Read mapping:** first key step in genome sequence analysis
  - Aligns reads to potential matching locations in the reference genome
  - For each matching location, the alignment step finds the degree of similarity (alignment score)

- Calculating the alignment score requires computationally-expensive approximate string matching (ASM) to account for differences between reads and the reference genome due to:
  - Sequencing errors
  - Genetic variation

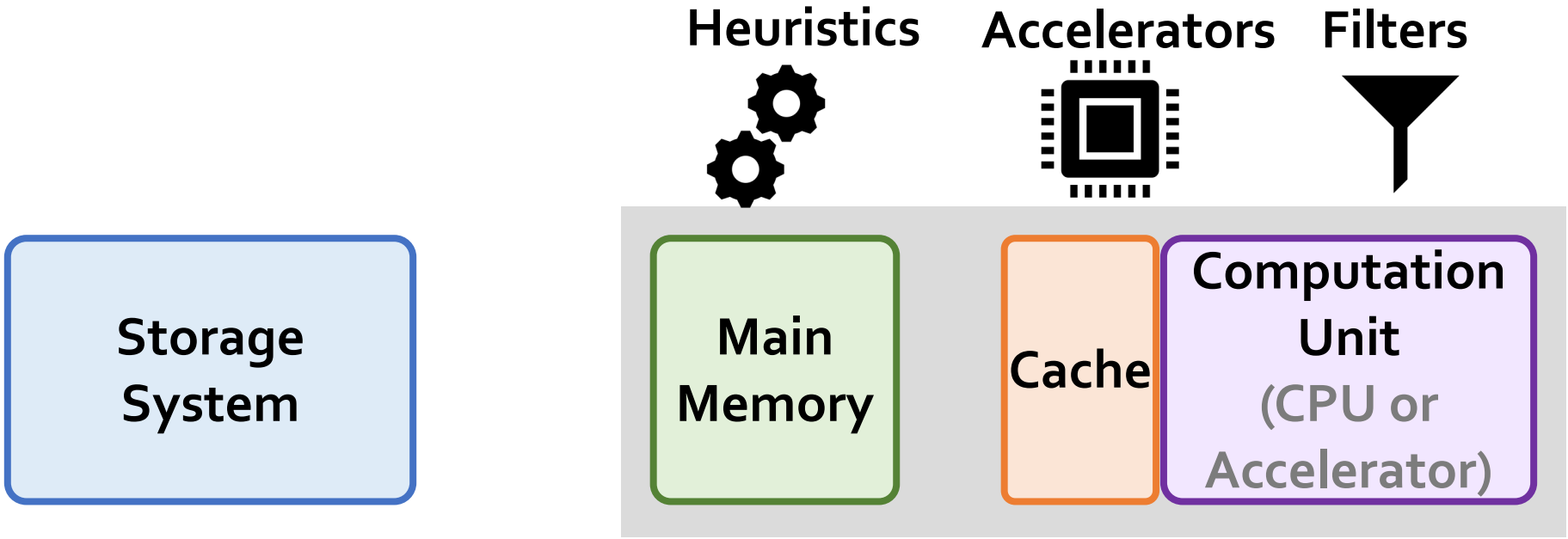
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Compute-Centric Accelerators

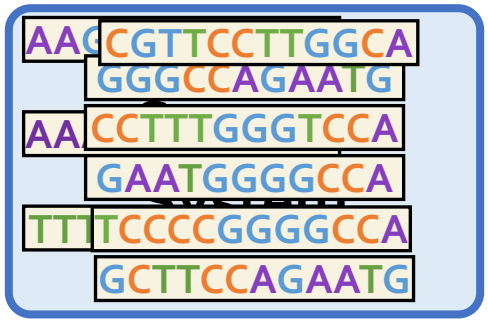


✓ Computation overhead

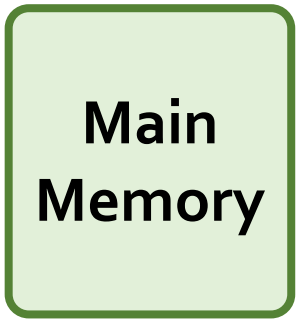
✗ Data movement overhead

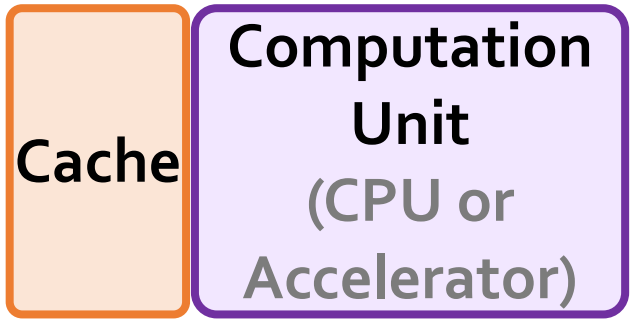
Key Idea: In-Storage Filtering

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



Filtered Reads





Exactly-matching reads

Do not need expensive approximate string matching during alignment

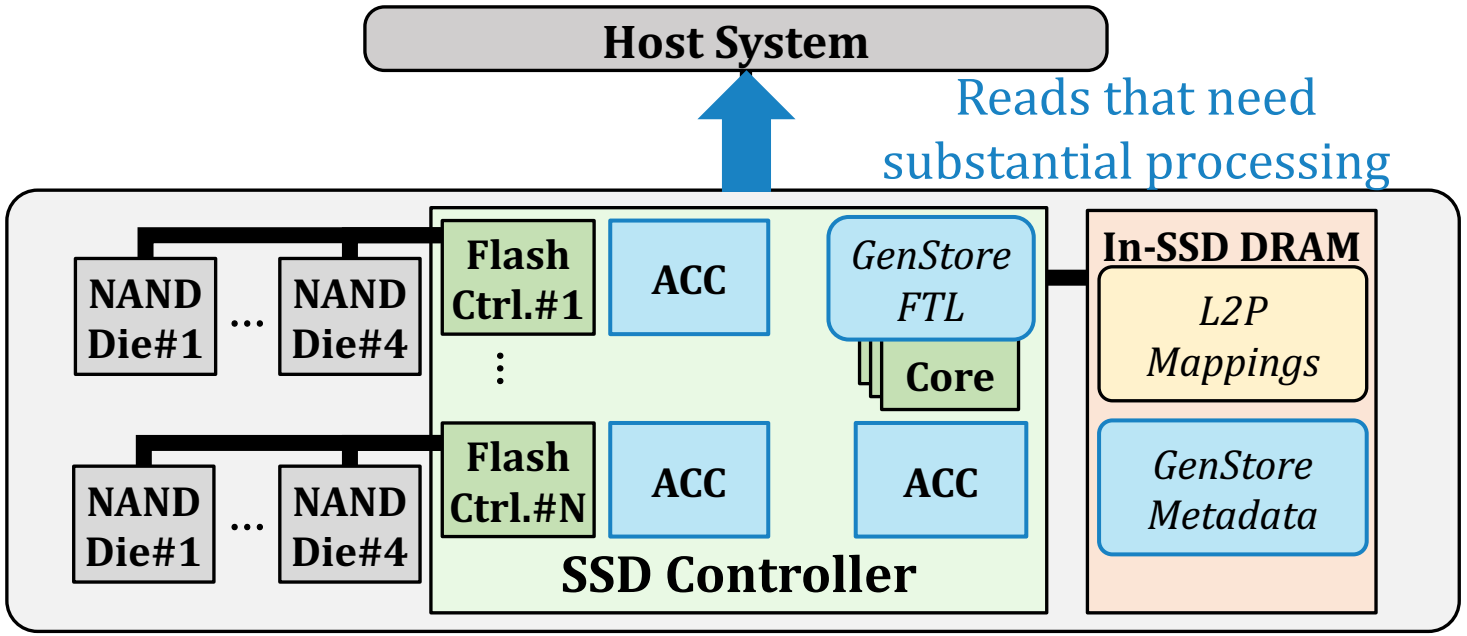
Non-matching reads

Do not have potential matching locations and can skip alignment

GenStore

- **Key idea:** Filter reads that do not require alignment inside the storage system

- **Challenges**
  - Different behavior across read mapping workloads
  - Limited hardware resources in the SSD



Filtering Opportunities

- Sequencing machines produce one of two kinds of reads
  - **Short reads:** highly accurate and short
  - **Long reads:** less accurate and long

Reads that do not require the expensive alignment step:

Exactly-matching reads

Do not need expensive approximate string matching during alignment

- Low sequencing error rates (short reads) combined with
- Low genetic variation

Non-matching reads

Do not have potential matching locations, so they skip alignment

- High sequencing error rates (long reads) or
- High genetic variation (short or long reads)

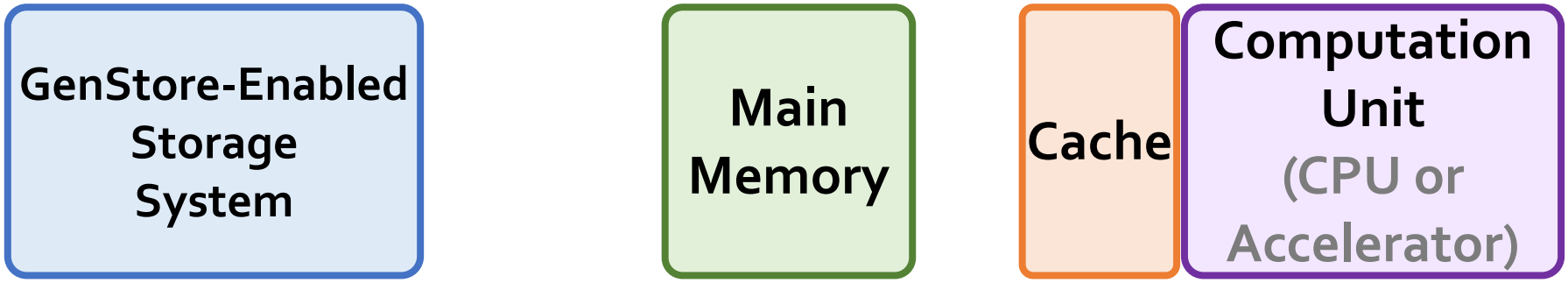
GenStore

GenStore-EM for Exactly-Matching Reads

GenStore-NM for Non-Matching Reads

GenStore

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

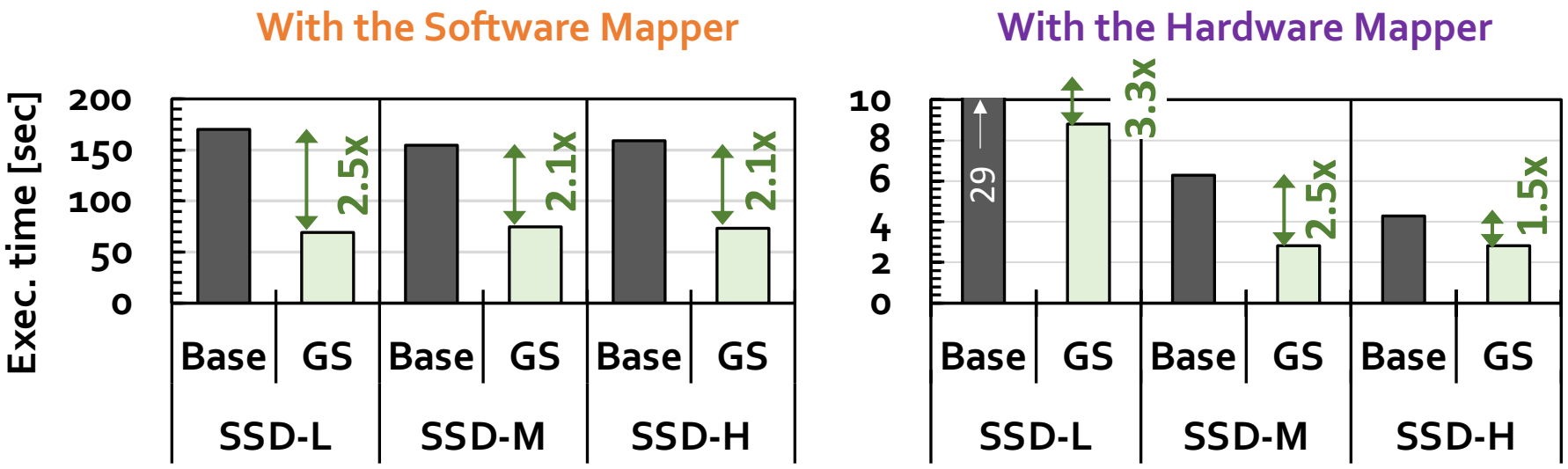


Computation overhead

Data movement overhead

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

Performance – GenStore-EM

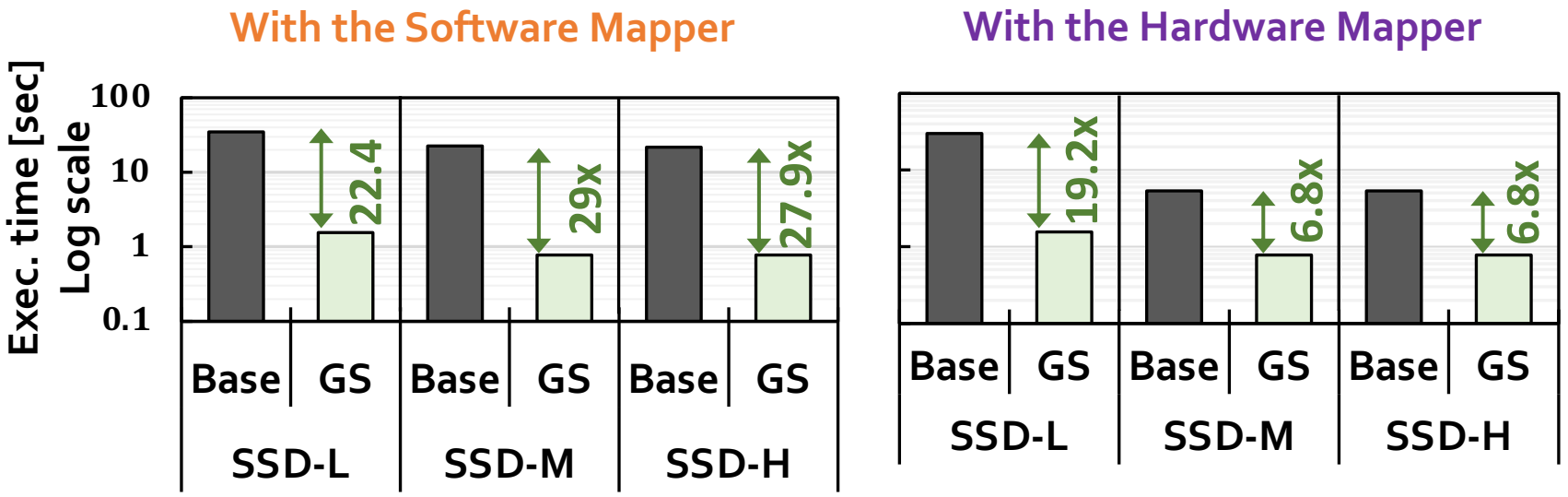


2.1× - 2.5× speedup compared to the software Base

1.5× – 3.3× speedup compared to the hardware Base

On average 3.92× energy reduction

Performance – GenStore-NM



22.4× – 27.9× speedup compared to the software Base

6.8× – 19.2× speedup compared to the hardware Base

On average 27.2× energy reduction

Area and Power Consumption

- Based on **Synthesis** of **GenStore** accelerators using the Synopsys Design Compiler @ 65nm technology node

Logic unit	# of instances	Area [mm²]	Power [mW]
Comparator	1 per SSD	0.0007	0.14
K -mer Window	2 per channel	0.0018	0.27
Hash Accelerator	2 per SSD	0.008	1.8
Location Buffer	1 per channel	0.00725	0.37375
Chaining Buffer	1 per channel	0.008	0.95
Chaining PE	1 per channel	0.004	0.98
Control	1 per SSD	0.0002	0.11
Total for an 8-channel SSD	-	0.2	26.6

Only 0.006% of a 14nm Intel Processor, less than 9.5% of the three ARM processors in a SATA SSD controller

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 Ghiasi[1] Jisung Park[1] Harun Mustafa[1] Jeremie Kim[1] Ataberk Olgun[1]
Arvid Gollwitzer[1] Damla Senol Cali[2] Can Firtina[1] Haiyu Mao[1] Nour Almadhoun Alserr[1]
Rachata Ausavarungnirun[3] Nandita Vijaykumar[4] Mohammed Alser[1] Onur Mutlu[1]

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

https://arxiv.org/abs/2202.10400

Tight Integration of Genome Analysis Tasks

- 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 Mao[1] Mohammed Alser[1] Mohammad Sadrosadati[1] Can Firtina[1] Akanksha Baranwal[1]
Damla Senol Cali[2] Aditya Manglik[1] Nour Almadhoun Alserr[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bionano Genomics*

Accelerating Sequence-to-Graph Mapping

- Damla Senol Cali, Konstantinos Kanellopoulos, Joel Lindegger, Zulal Bingol, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie Kim, Nika MansouriGhiasi, 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 Cali[1] Konstantinos Kanellopoulos[2] Joël Lindegger[2] Zülal Bingöl[3]
Gurpreet S. Kalsi[4] Ziyi Zuo[5] Can Firtina[2] Meryem Banu Cavlak[2] Jeremie Kim[2]
Nika Mansouri Ghiasi[2] Gagandeep Singh[2] Juan Gómez-Luna[2] Nour Almadhoun Alserr[2]
Mohammed Alser[2] Sreenivas Subramoney[4] Can Alkan[3] Saugata Ghose[6] Onur Mutlu[2]

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

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 Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

https://arxiv.org/pdf/2406.19113

MegIS

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

Nika Mansouri Ghiasi

Mohammad Sadrosadati Harun Mustafa Arvid Gollwitzer Can Firtina

Julien Eudine Haiyu Mao Joël Lindegger Meryem Banu Cavlak

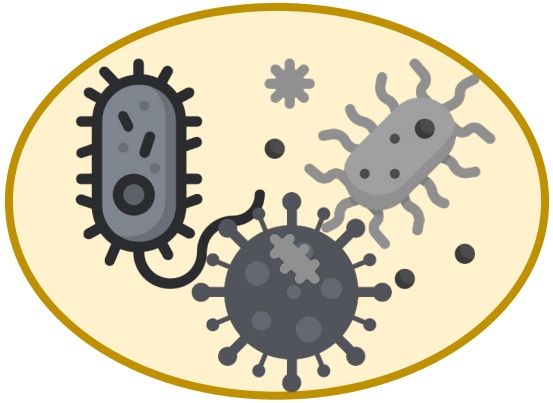
Mohammed Alser Jisung Park Onur Mutlu

SAFARI

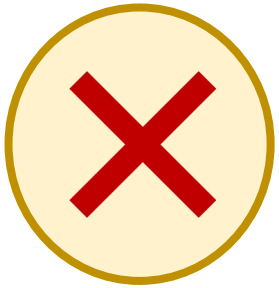
ETH zürich POSTECH

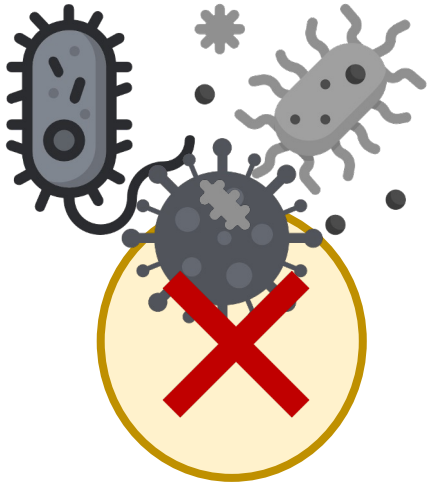
What is Metagenomics?

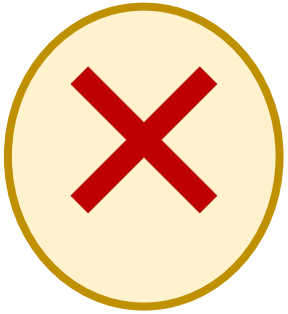
- ***Metagenomics:*** Study of genome sequences of **diverse organisms** within a **shared environment** (e.g., blood, ocean, soil)



- **Overcomes the limitations of traditional genomics**
  - Bypasses the need for analyzing individual species in isolation

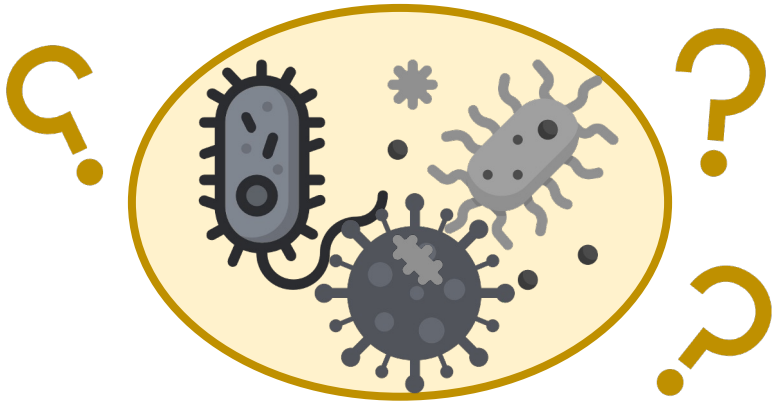






What is Metagenomics?

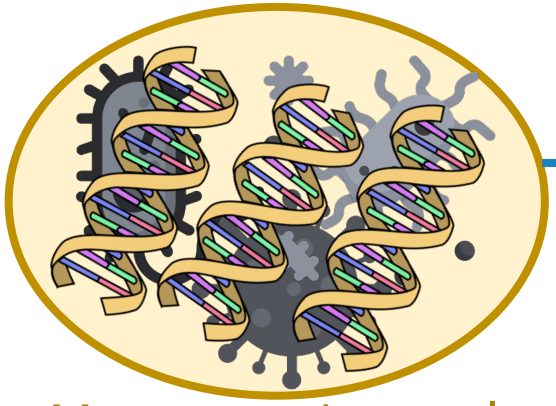
- ***Metagenomics:*** Study of genome sequences of **diverse organisms** within a **shared environment** (e.g., blood, ocean, soil)



Has led to groundbreaking advances

- Precision medicine
- Understanding microbial diversity of an environment
- Discovering early warnings of communicable diseases

Metagenomic Analysis



Metagenomic sample with species that are **not known** in advance

A large database containing information on **many species**

Preparation of Input Queries

Query K-mers: GCTCA, CTCAT, TCATG, ...

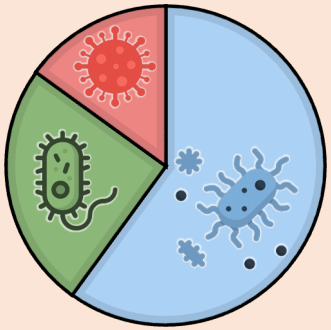
Presence/Absence Identification

V. cholerae

SARS-CoV-2

E. coli

Abundance Estimation

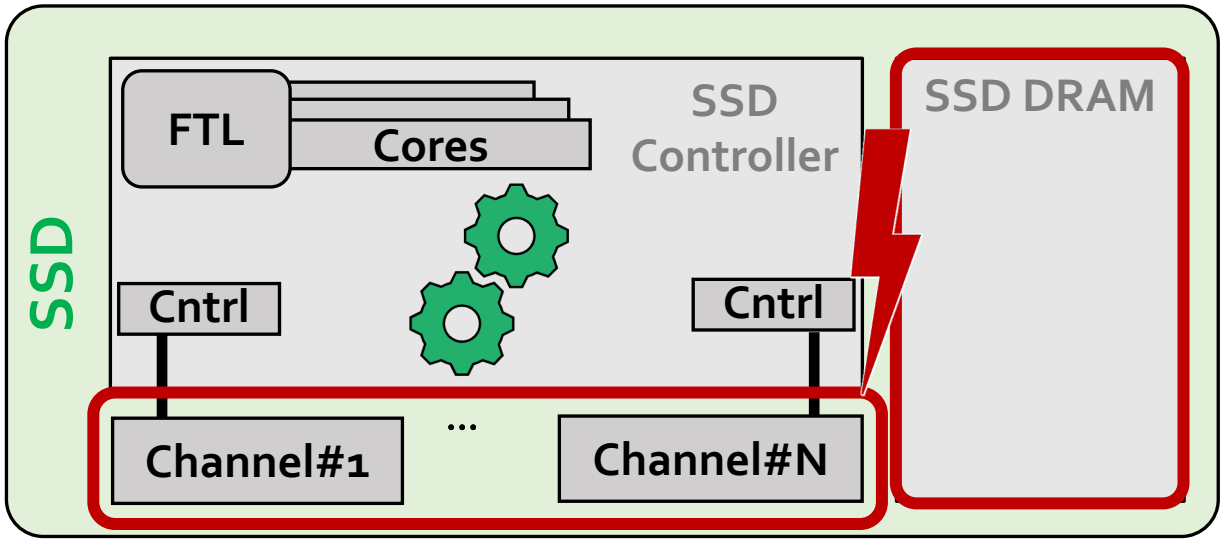


(e.g., > 100 TBs in emerging databases)

Challenges of In-Storage Processing

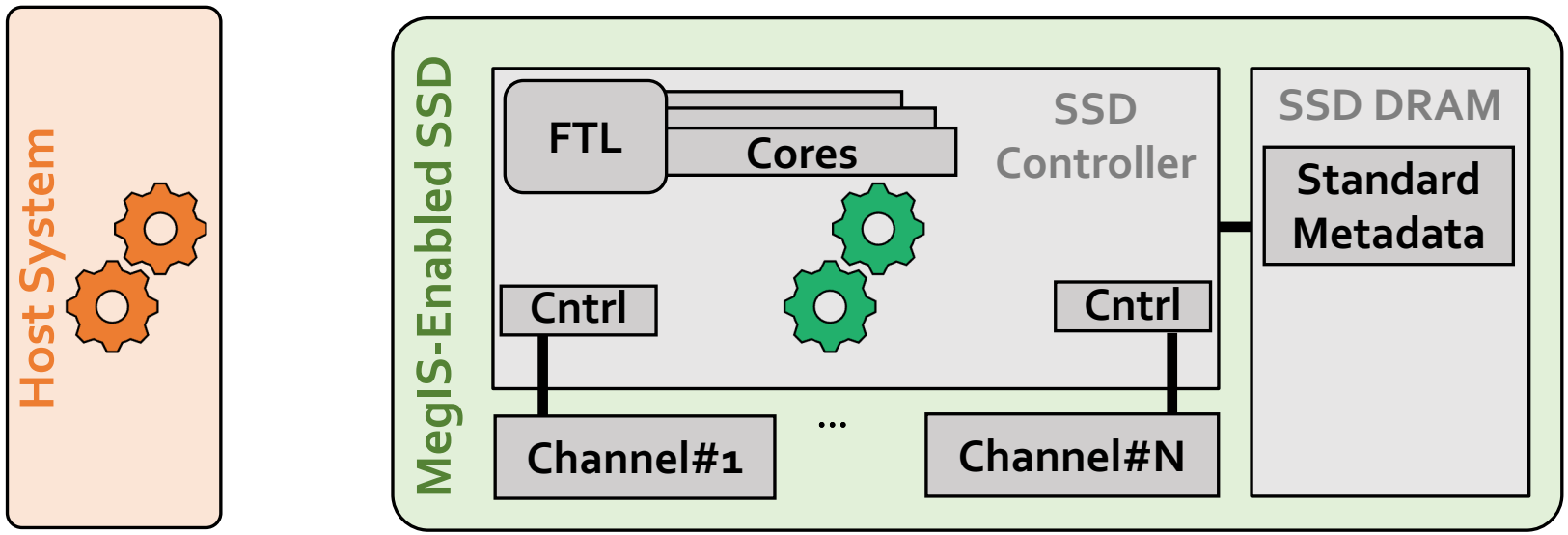
No metagenomic analysis tool can run in-storage due to SSD limits

- Long **latency of NAND flash** chips
- Limited **DRAM capacity** inside the SSD
- Limited **DRAM bandwidth** inside the SSD

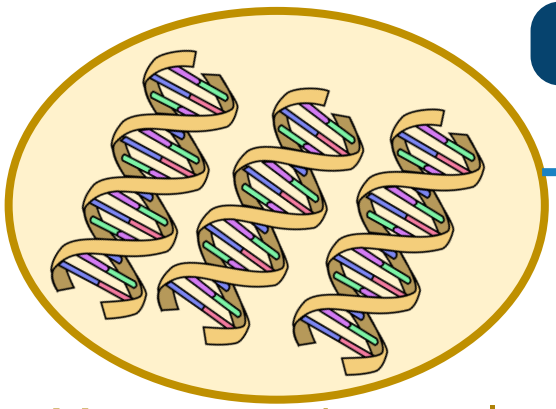


MegIS: Metagenomics In-Storage

- First in-storage system for *end-to-end* metagenomic analysis
- **Idea:** Cooperative in-storage processing for metagenomic analysis
  - Hardware/software co-design between

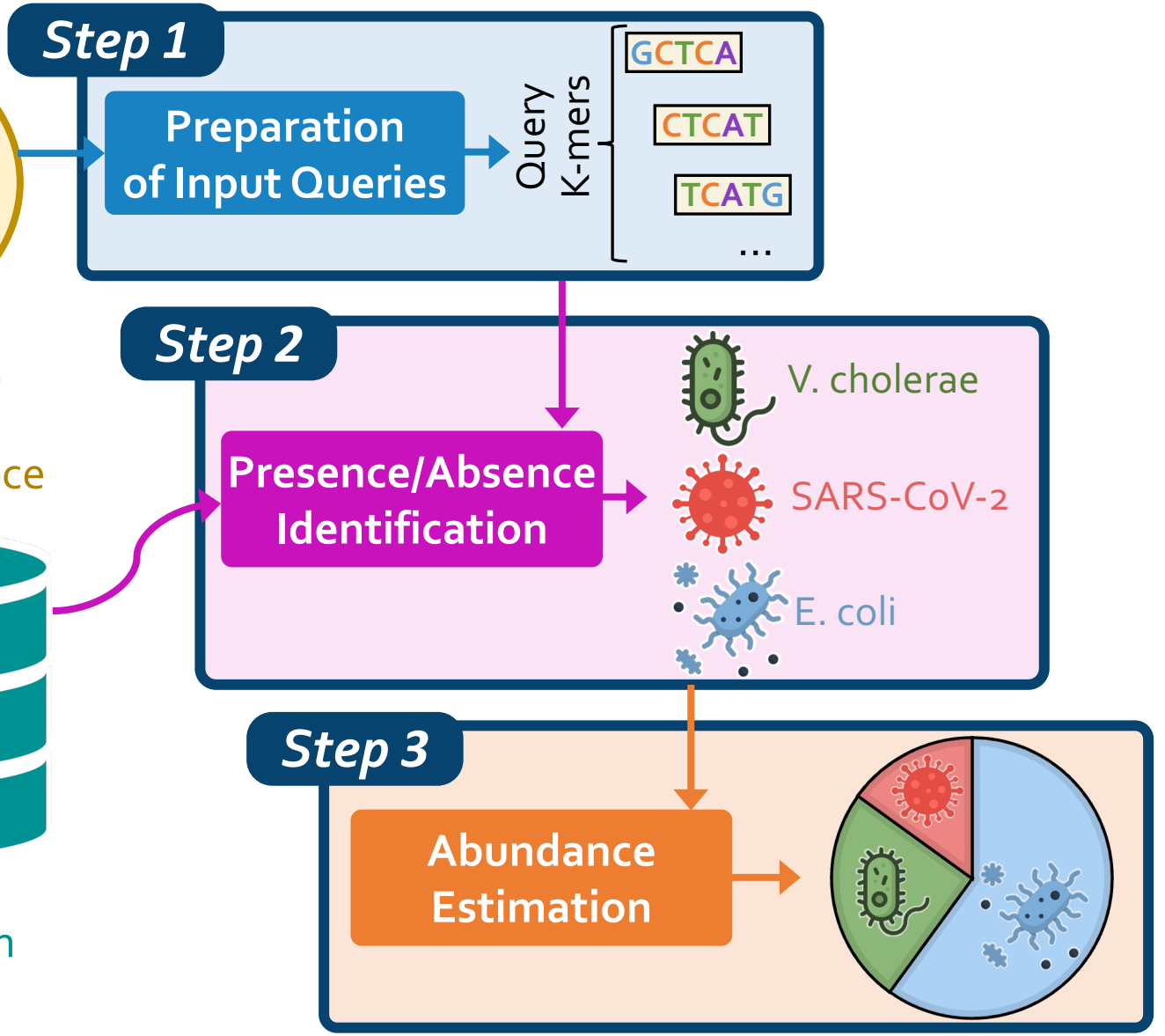


MegIS's Steps

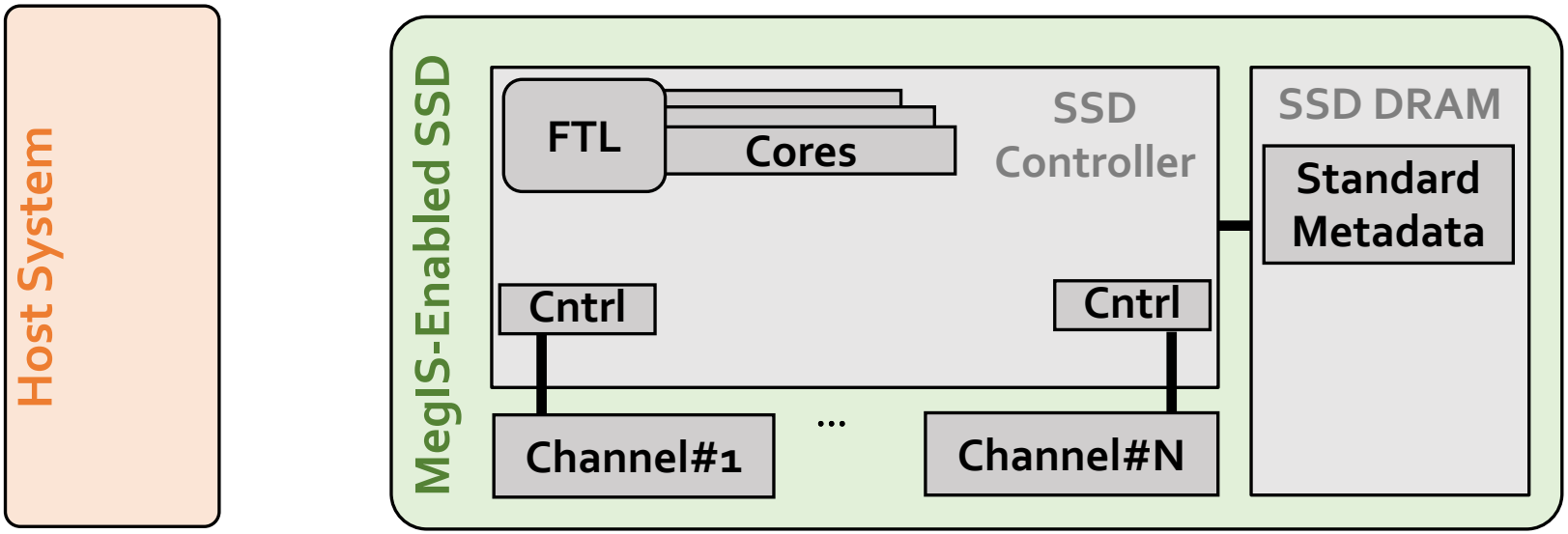


Metagenomic sample with species that are **not known** in advance

A large database containing information on **many species**



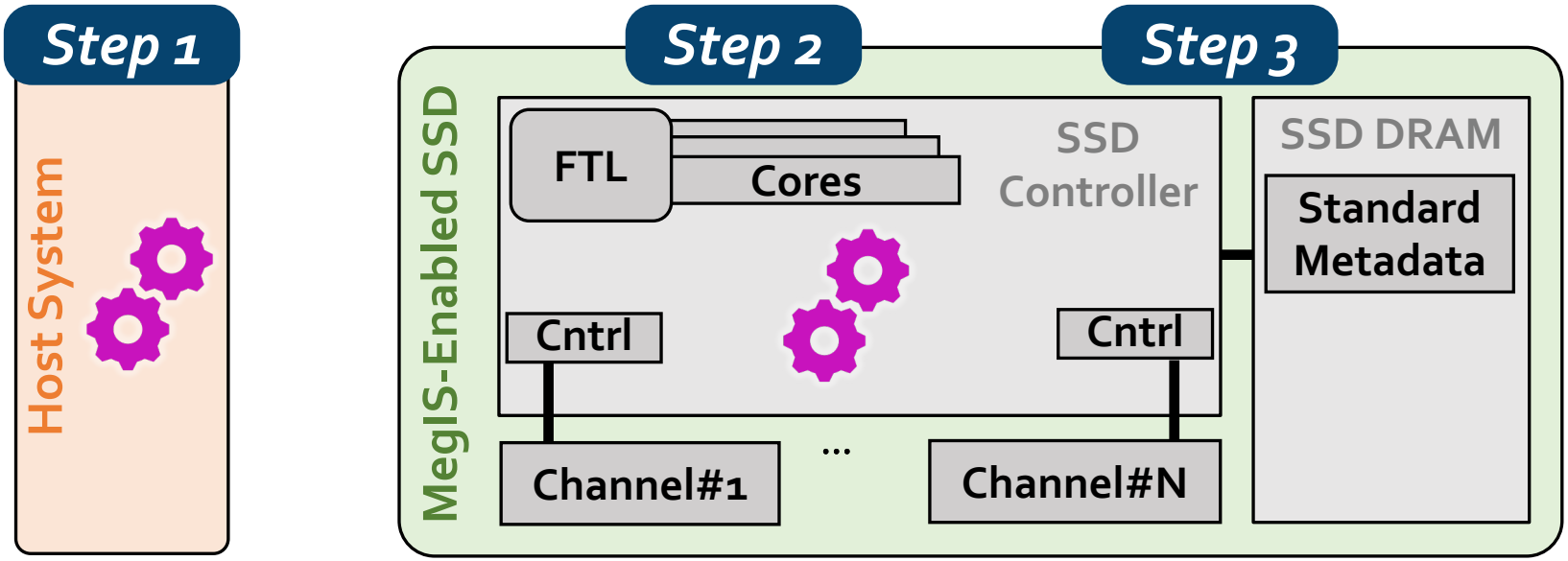
MegIS Hardware-Software Co-Design



MegIS Hardware-Software Co-Design

Task partitioning and mapping

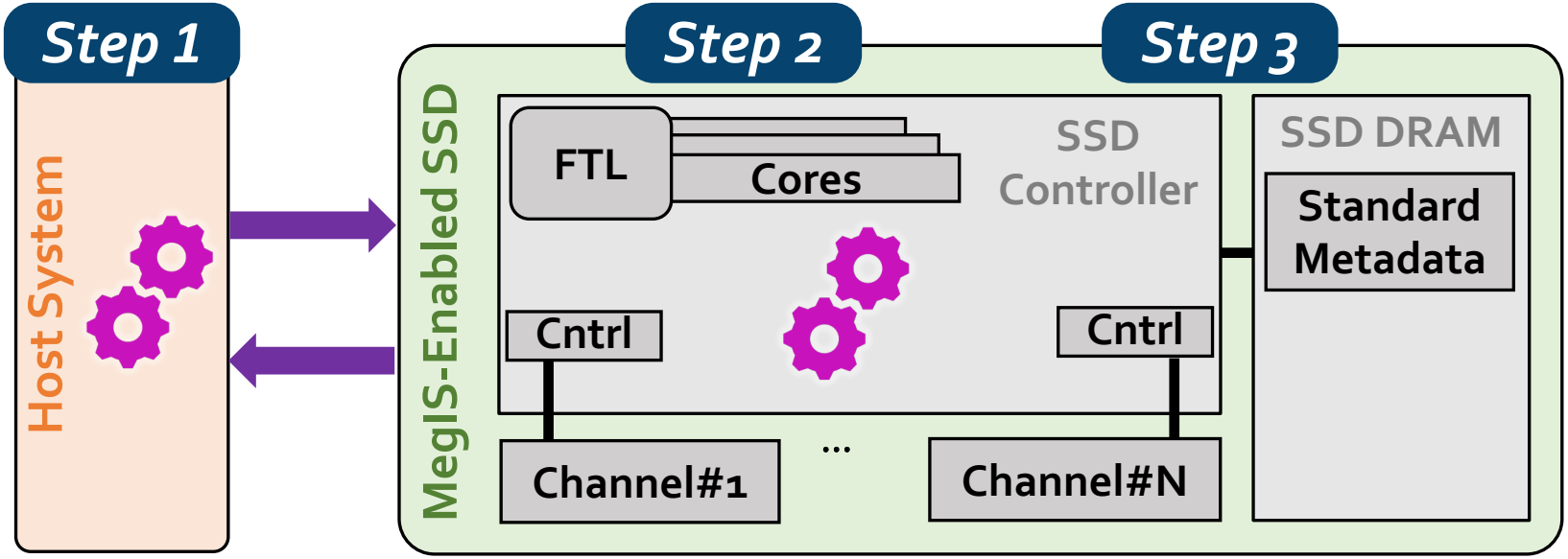
- *Each step executes in its most suitable system*



MegIS Hardware-Software Co-Design

Task partitioning and mapping
- *Each step executes in its most suitable system*

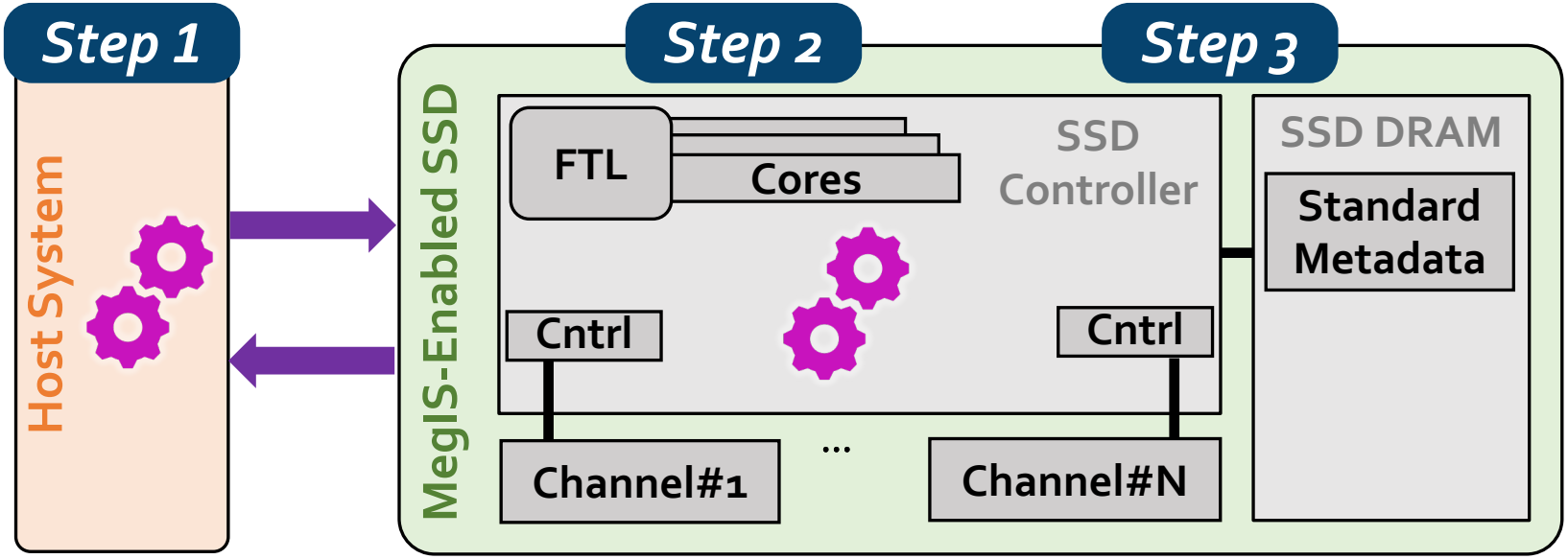
Data/computation flow coordination
- *Reduce communication overhead*
- *Reduce #writes to flash chips*



MegIS Hardware-Software Co-Design

Task partitioning and mapping
- *Each step executes in its most suitable system*

Data/computation flow coordination
- *Reduce communication overhead*
- *Reduce #writes to flash chips*

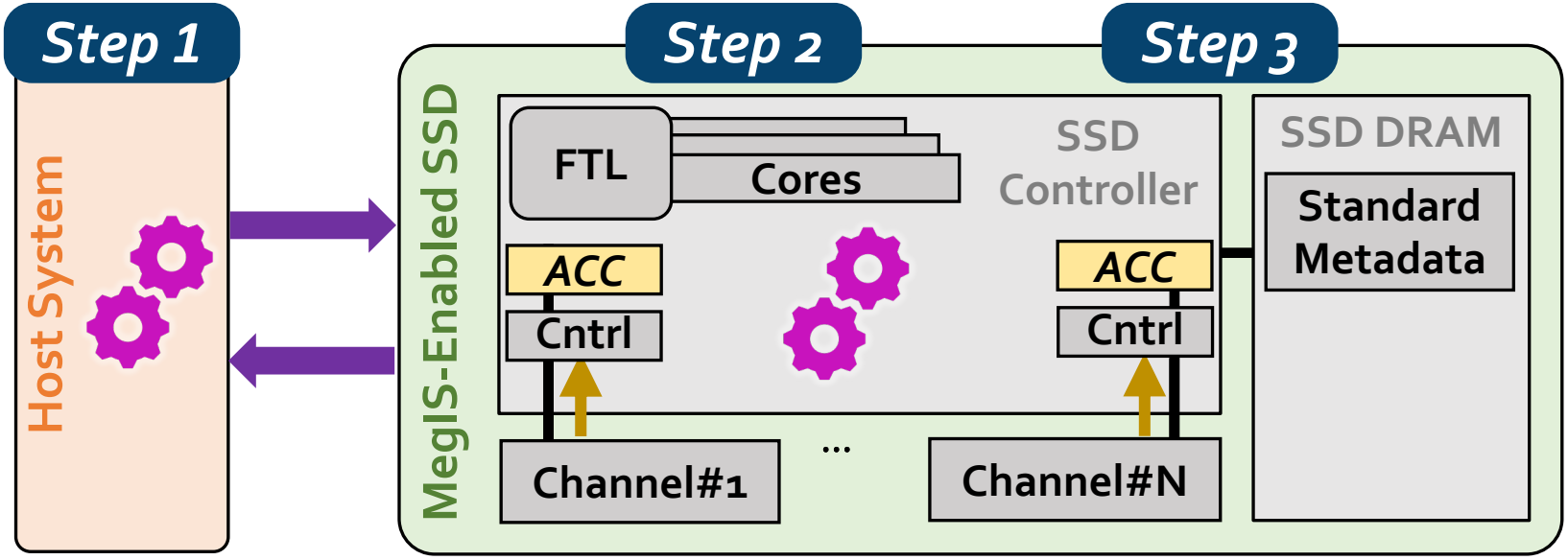


Storage-aware algorithms
- *Enable efficient access patterns to the SSD*

MegIS Hardware-Software Co-Design

Task partitioning and mapping
- *Each step executes in its most suitable system*

Data/computation flow coordination
- *Reduce communication overhead*
- *Reduce #writes to flash chips*



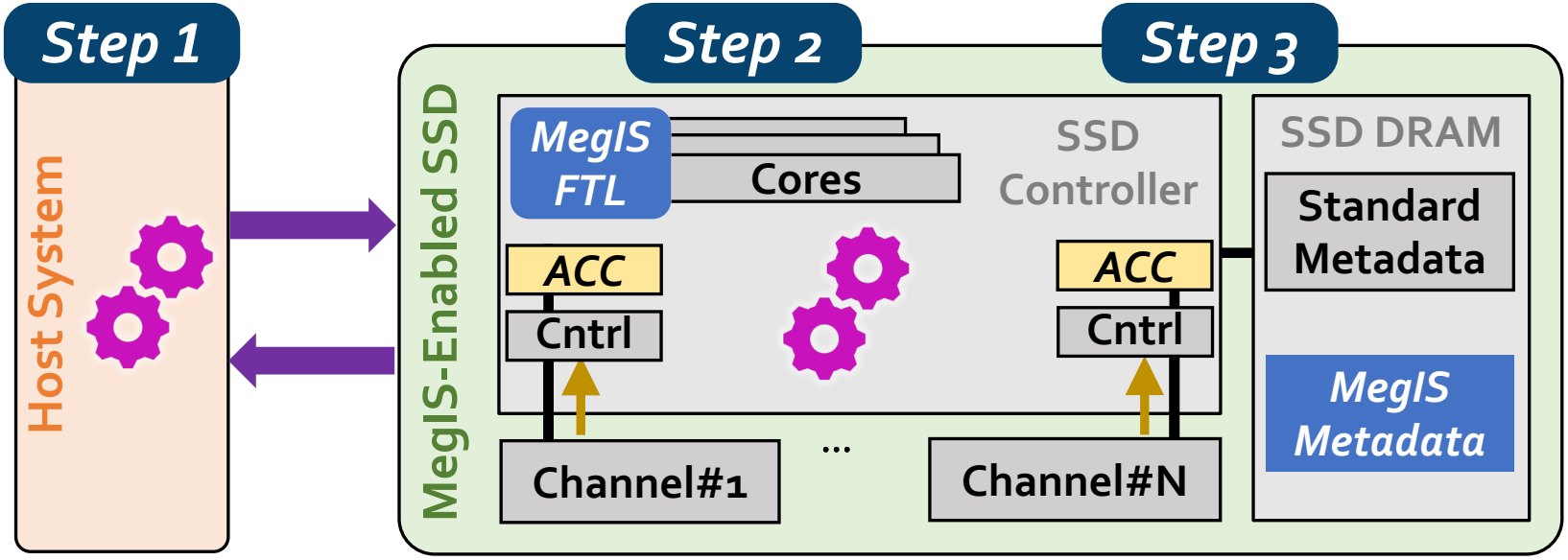
Storage-aware algorithms
- *Enable efficient access patterns to the SSD*

Lightweight in-storage accelerators
- *Minimize SRAM/DRAM buffer spaces needed inside the SSD*

MegIS Hardware-Software Co-Design

Task partitioning and mapping
- *Each step executes in its most suitable system*

Data/computation flow coordination
- *Reduce communication overhead*
- *Reduce #writes to flash chips*



Storage-aware algorithms
- *Enable efficient access patterns to the SSD*

Lightweight in-storage accelerators
- *Minimize SRAM/DRAM buffer spaces needed inside the SSD*

Data mapping scheme and Flash Translation Layer (FTL)
- *Specialize to the characteristics of metagenomic analysis*
- *Leverage the SSD's full internal bandwidth*

Evaluation Methodology Overview

Performance, Energy, and Power Analysis

Hardware Components

- Synthesized Verilog model for the in-storage accelerators
- MQSim *[Tavakkol+, FAST'18]* for SSD's internal operations
- Ramulator *[Kim+, CAL'15]* for SSD's internal DRAM

Software Components

Measure on a real system:

- AMD® EPYC® CPU with 128 physical cores
- 1-TB DRAM

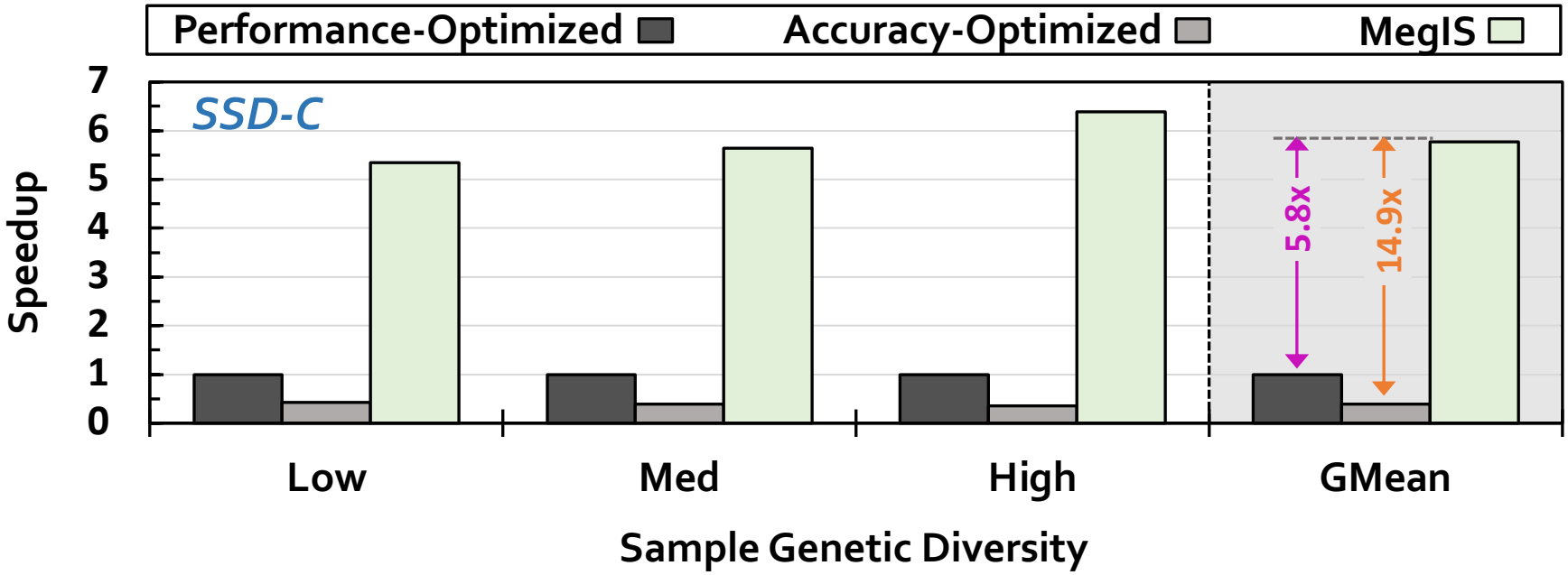
Baseline Comparison Points

- **Performance-optimized software**, Kraken2 *[Genome Biology'19]*
- **Accuracy-optimized software**, Metalign *[Genome Biology'20]*
- **PIM hardware-accelerated tool** (using processing-in-memory), Sieve *[ISCA'21]*

SSD Configurations

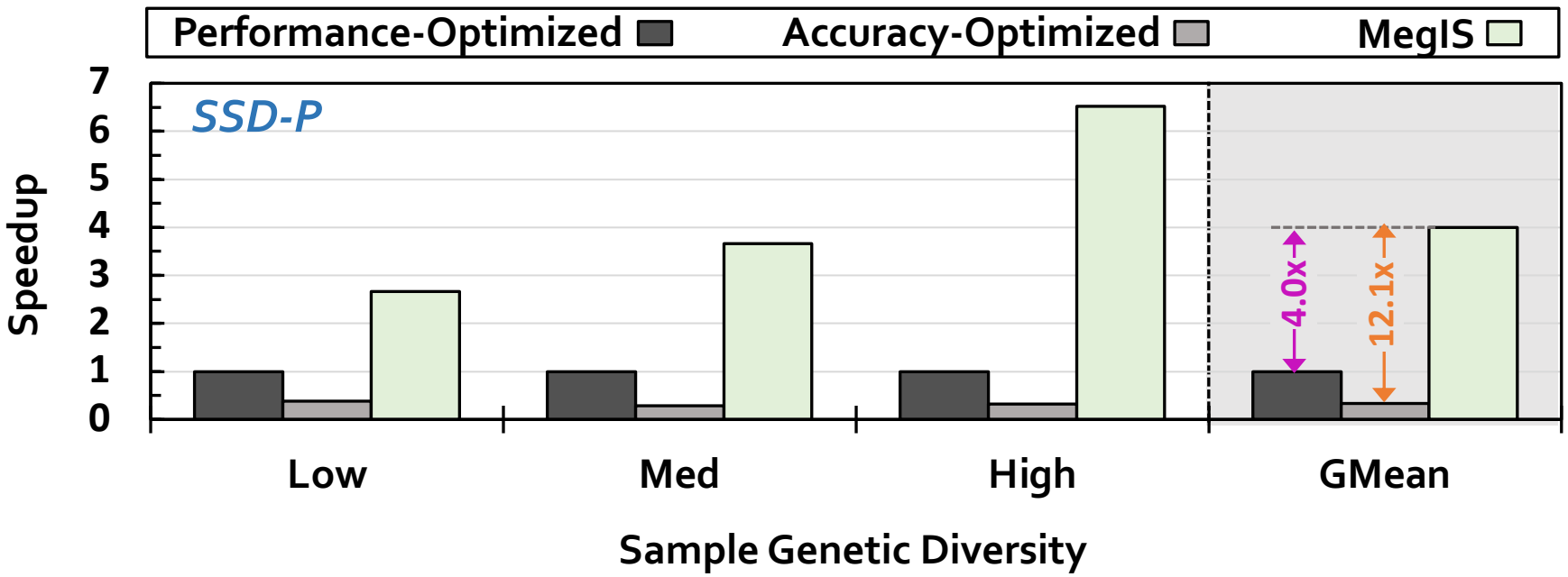
- **SSD-C:** with SATA3 interface (0.5 GB/s sequential read bandwidth)
- **SSD-P:** with PCIe Gen4 interface (7 GB/s sequential read bandwidth)

Speedup over Software (with Cost-Optimized SSD)



MegIS provides significant speedup over both Performance-Optimized and Accuracy-Optimized baselines

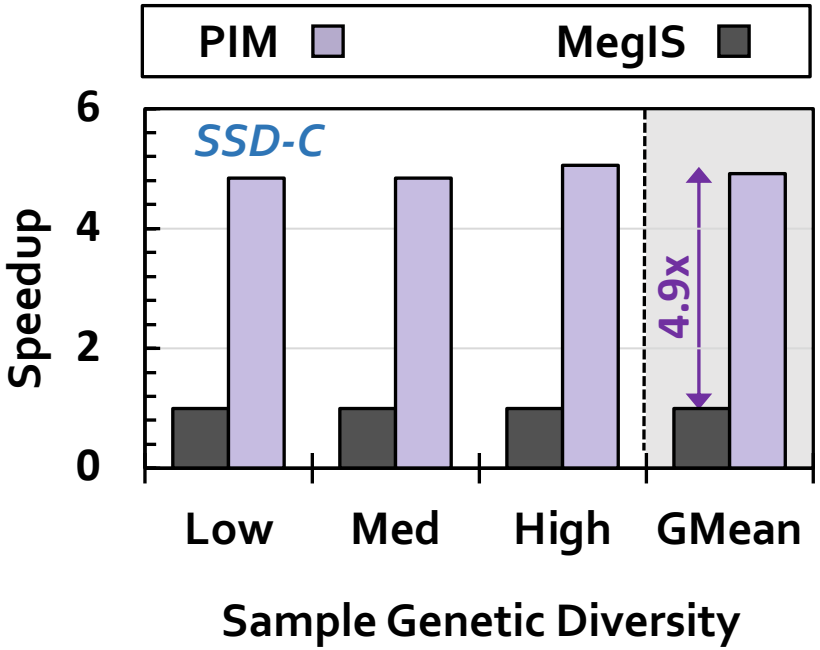
Speedup over Software (with Performance-Optimized SSD)

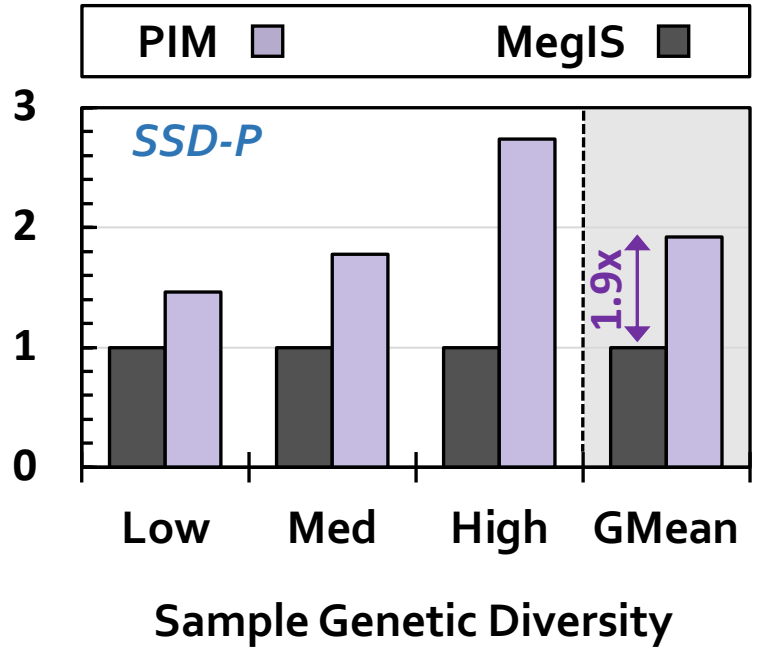


MegIS provides significant speedup over both Performance-Optimized and Accuracy-Optimized baselines

MegIS improves performance on both cost-optimized and performance-optimized SSDs

Speedup over the PIM Hardware Baseline

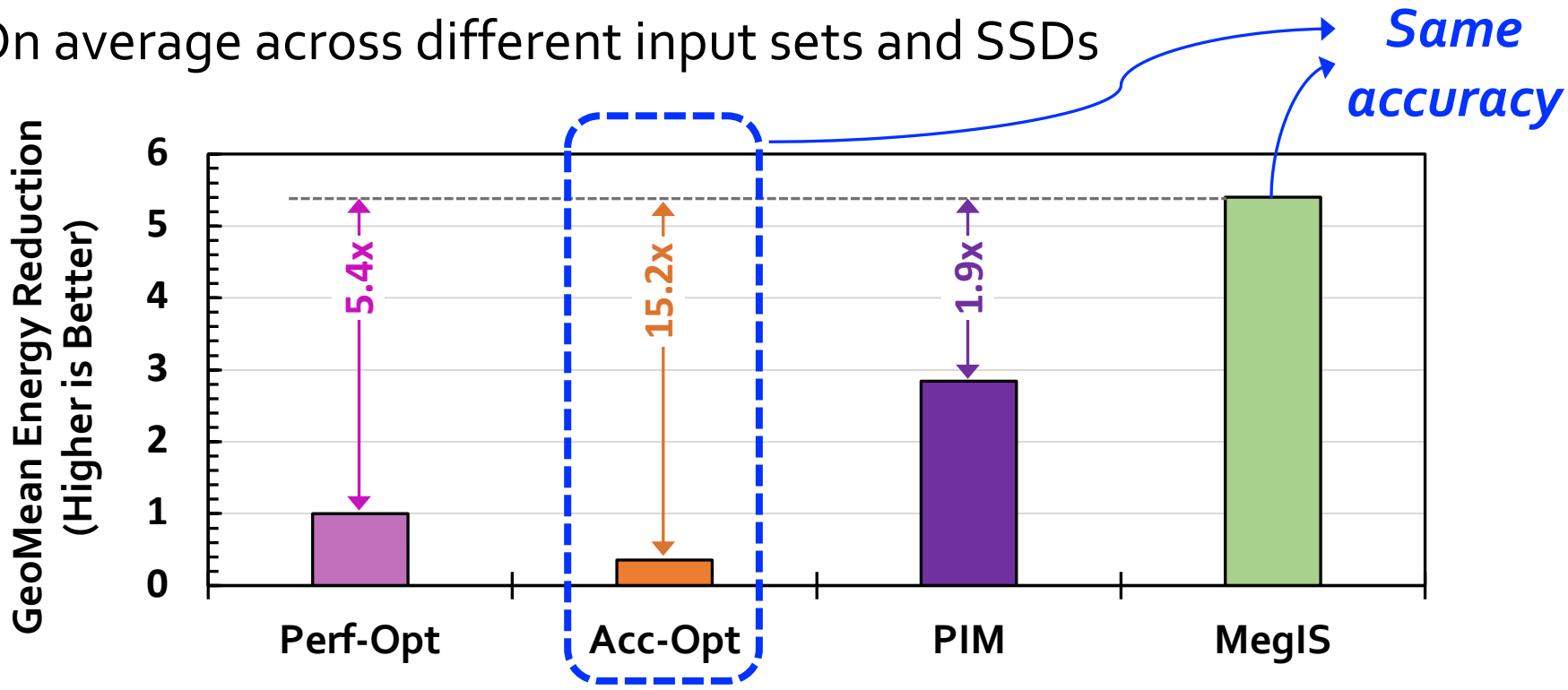




MegIS provides significant speedup over the PIM baseline

Reduction in Energy Consumption

- On average across different input sets and SSDs



MegIS provides significant energy reduction over the Performance-Optimized, Accuracy-Optimized, and PIM baselines

Accuracy, Area, and Power

Accuracy

- **Same accuracy** as the **accuracy-optimized** baseline
- **Significantly higher accuracy** than the **performance-optimized** and **PIM** baselines
  - 4.6 – 5.2× higher F1 score
  - 3 – 24% lower L1 norm error

Area and Power

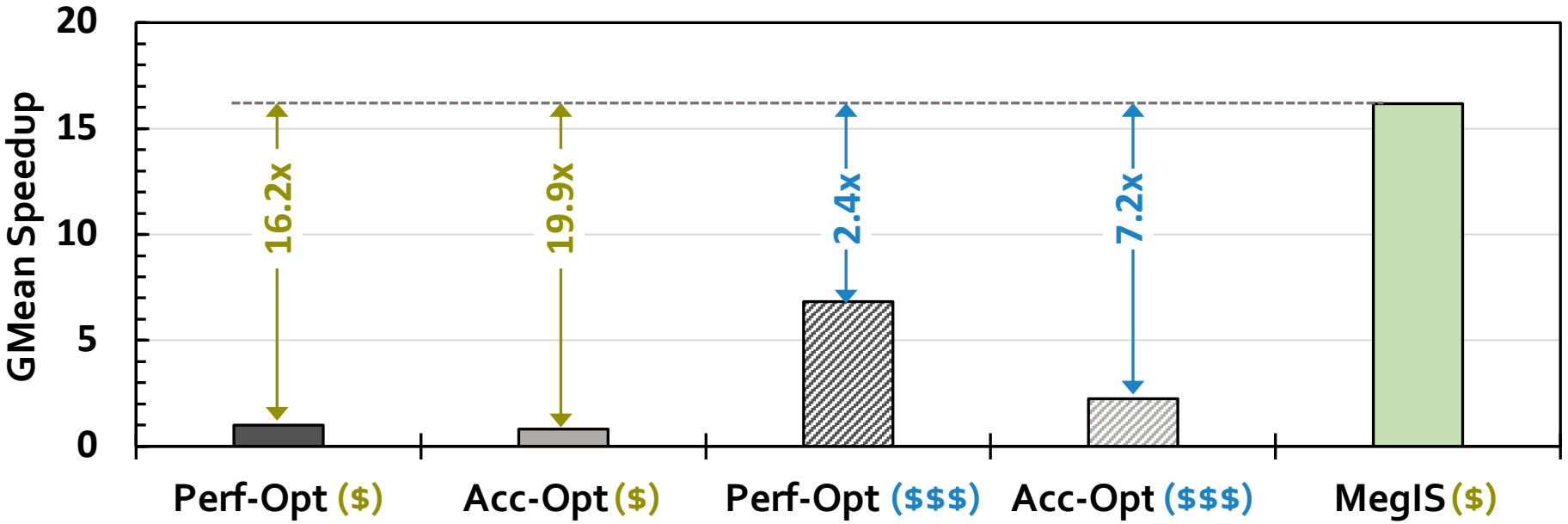
Total for an 8-channel SSD:

- **Area:** 0.04 mm^2
- **Power:** 7.658 mW

*(Only **1.7%** of the area and **4.6%** of the power consumption of three ARM Cortex R4 cores in an SSD controller)*

System Cost-Efficiency

- **Cost-optimized system ($):** With SSD-C and 64-GB DRAM
- **Performance-optimized system ($$$):** With SSD-P and 1-TB DRAM

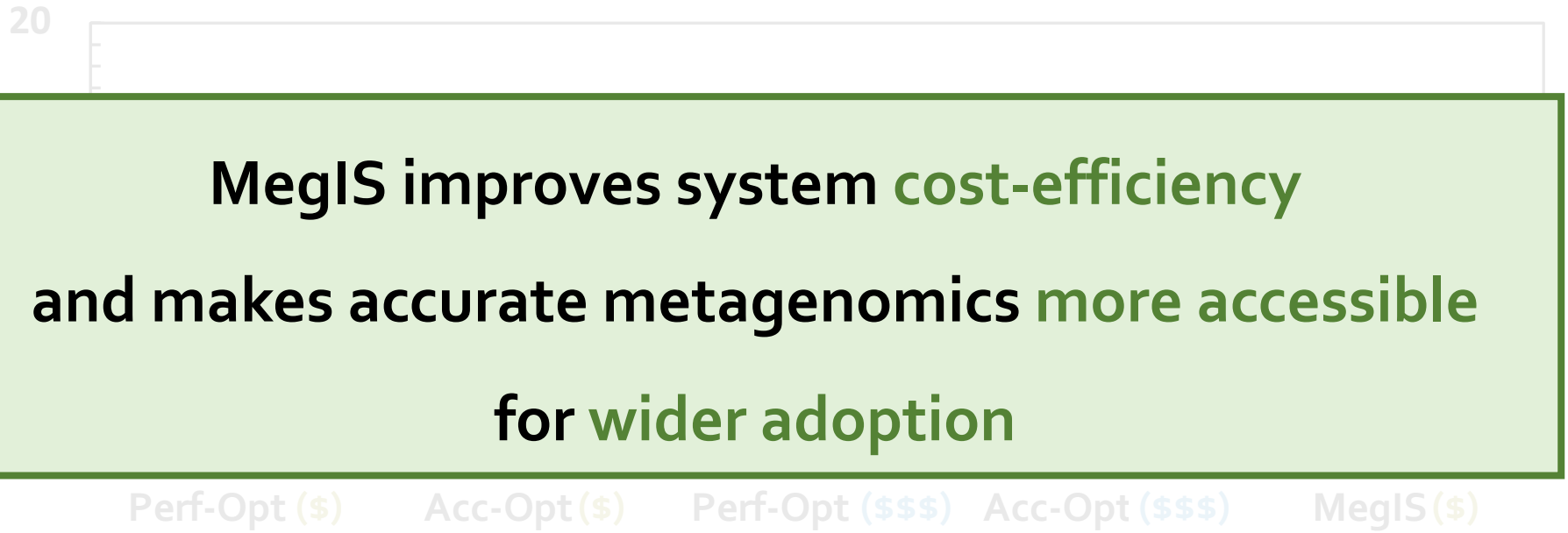


MegIS outperforms the baselines

***even* when running on a much less costly system**

System Cost-Efficiency

- **Cost-optimized system ($):** With SSD-C and 64-GB DRAM
- **Performance-optimized system ($$$):** With SSD-P and 1-TB DRAM



MegIS improves system cost-efficiency

and makes accurate metagenomics more accessible

for wider adoption

MegIS outperforms the baselines

even when running on a much less costly system

More in the Paper

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

Nika Mansouri Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

- Database sizes
- Memory capacities
- #SSDs
- #Channels
- #Samples

• MegIS's performance for abundance estimation

https://arxiv.org/abs/2406.19113

More to Come…

Processing in Storage: Two Types

1. Processing **near** Storage Devices
2. Processing **using** Storage Devices

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

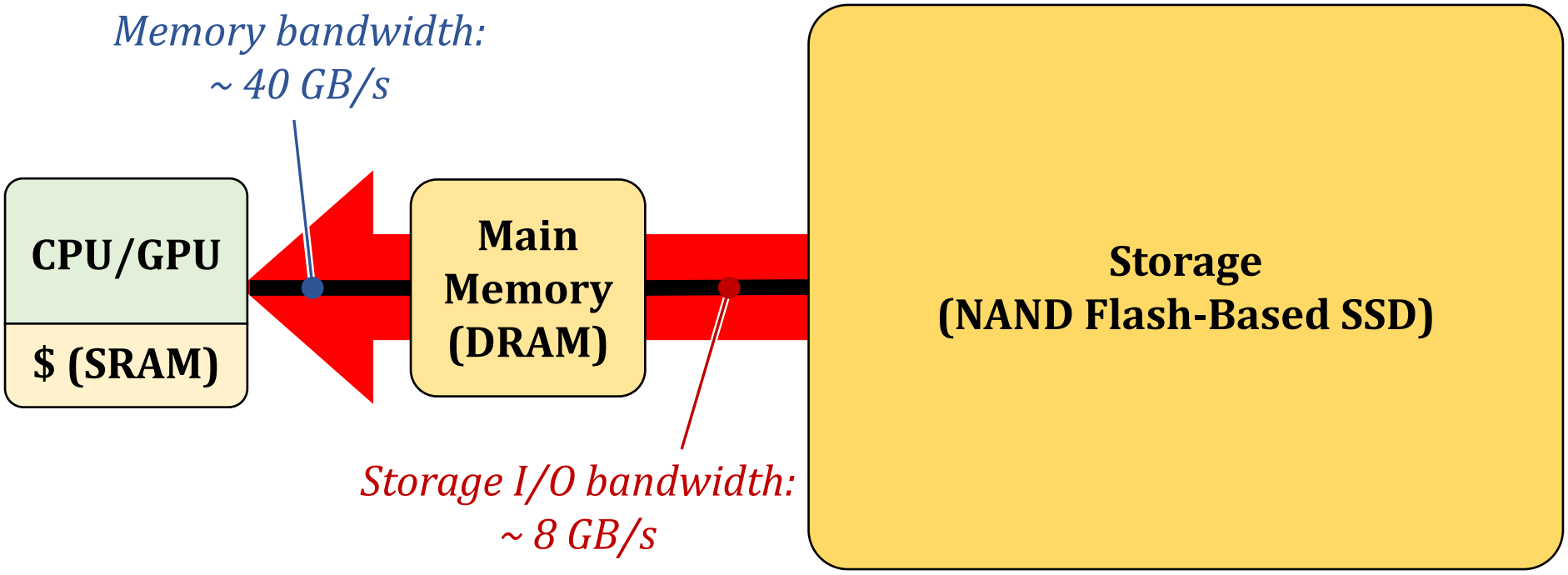
Jisung Park[§∇] Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

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

https://arxiv.org/pdf/2209.05566.pdf

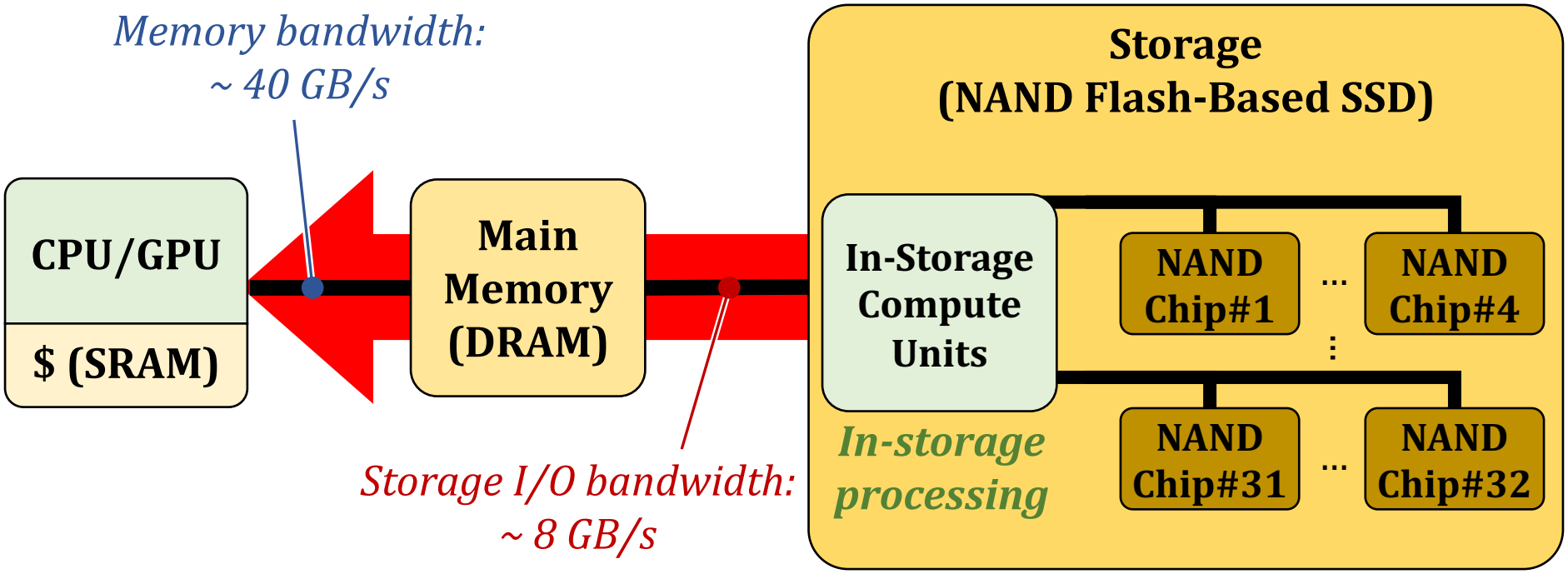
In-Storage Processing (ISP)

- Uses in-storage compute units (embedded cores or FPGA) to send only the computation results



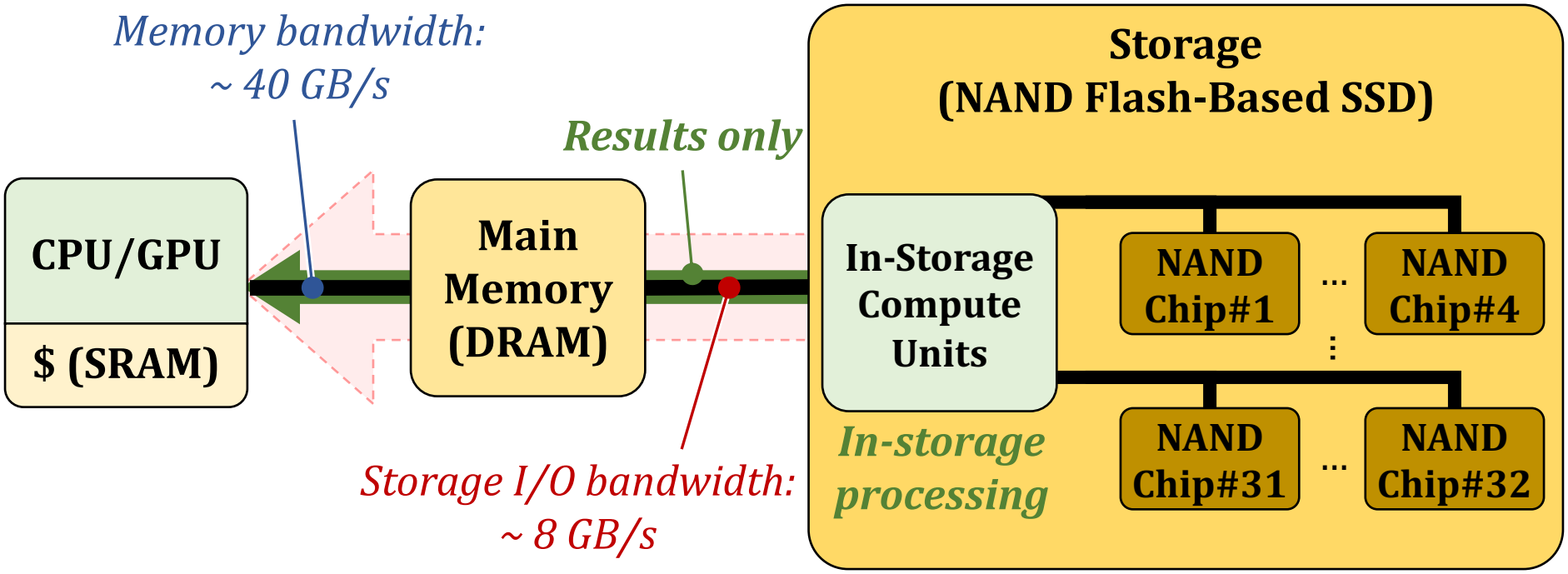
In-Storage Processing (ISP)

- Uses in-storage compute units (embedded cores or FPGA) to send only the computation results



In-Storage Processing (ISP)

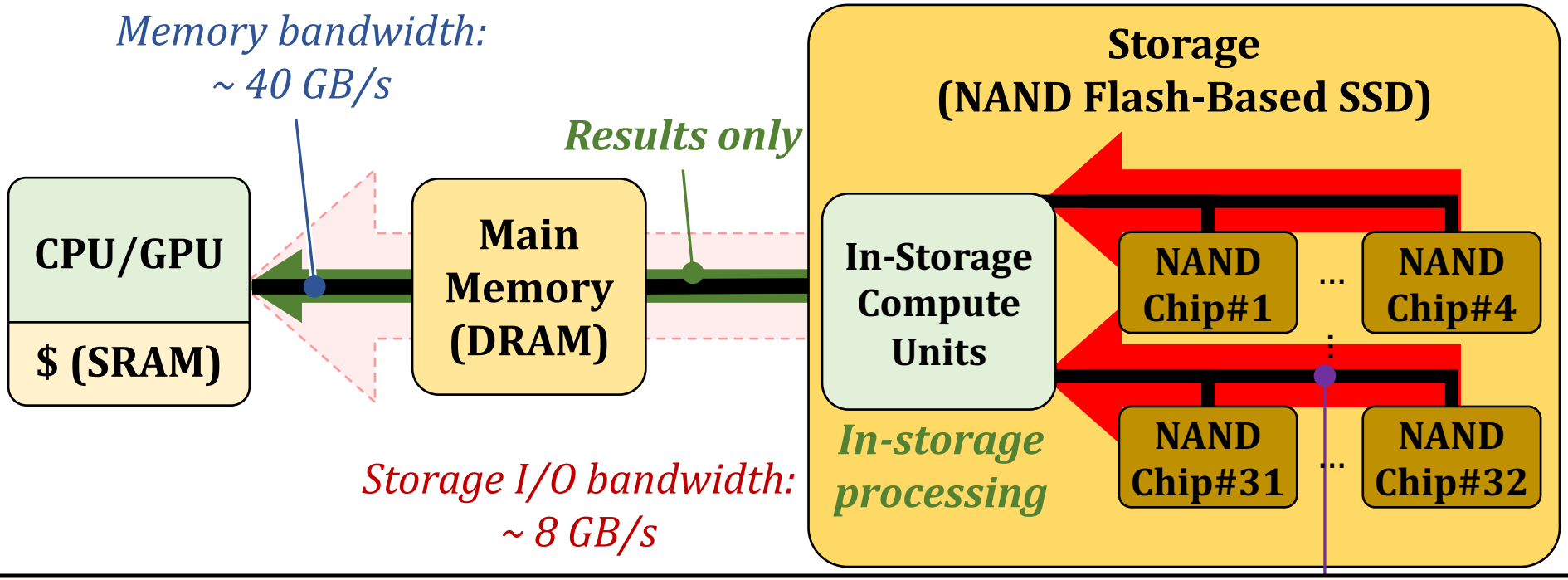
- Uses in-storage compute units (embedded cores or FPGA) to send only the computation results



ISP can mitigate data movement overhead by reducing SSD-external data movement

In-Storage Processing (ISP)

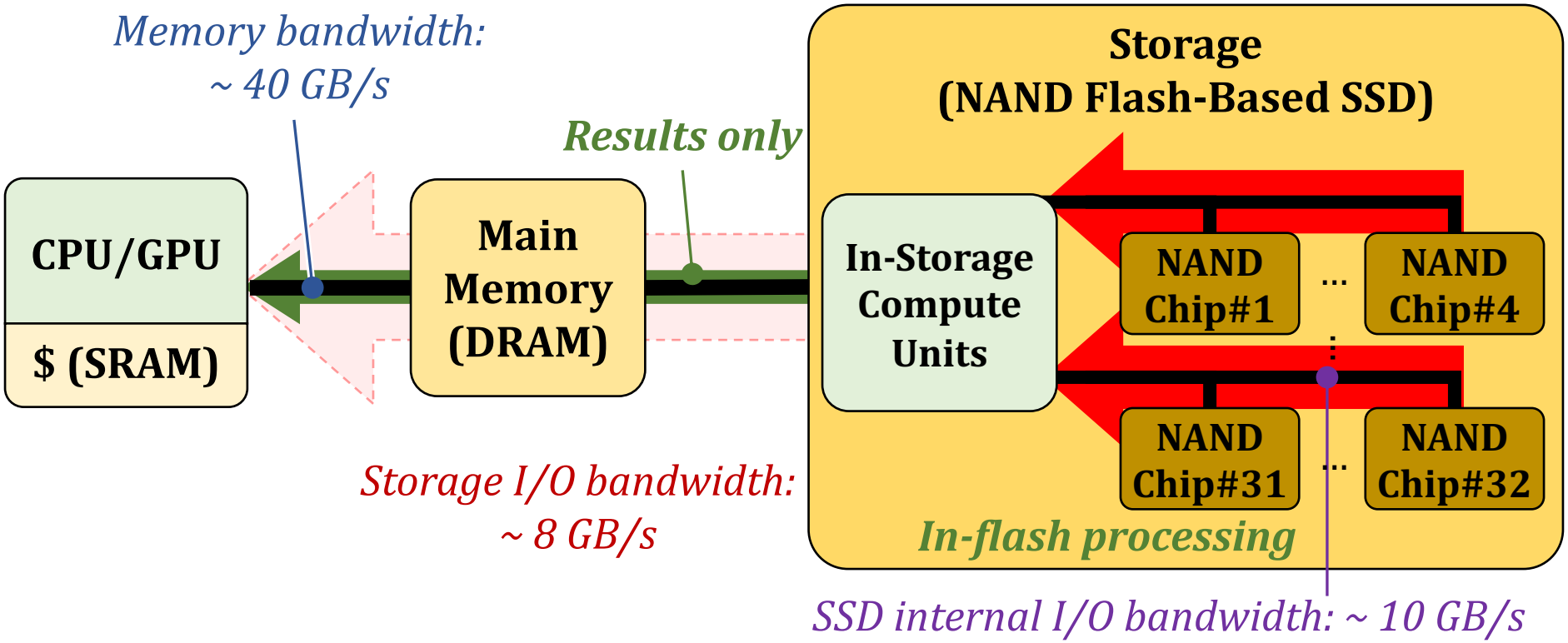
- Uses in-storage compute units (embedded cores or FPGA) to send only the computation results



SSD-internal bandwidth
becomes the new bottleneck in ISP

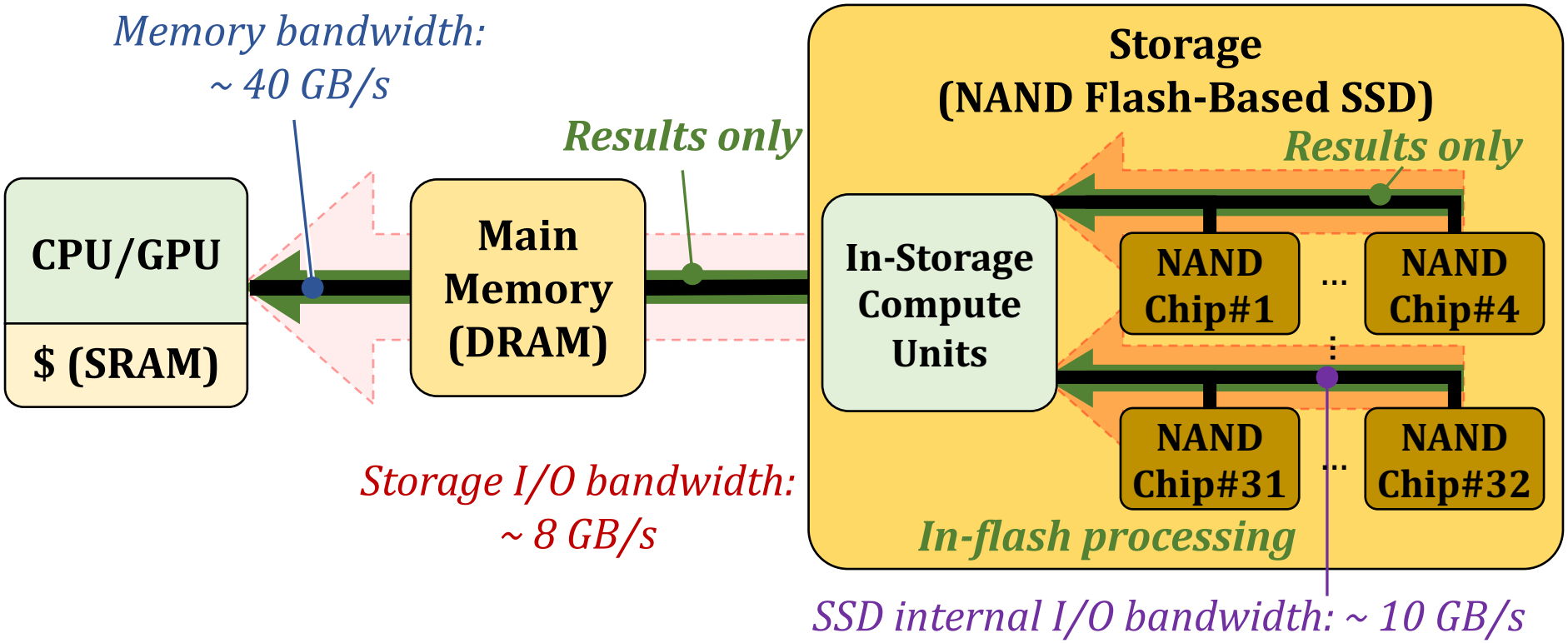
In-Flash Processing (IFP)

- Performs computation inside NAND flash chips



In-Flash Processing (IFP)

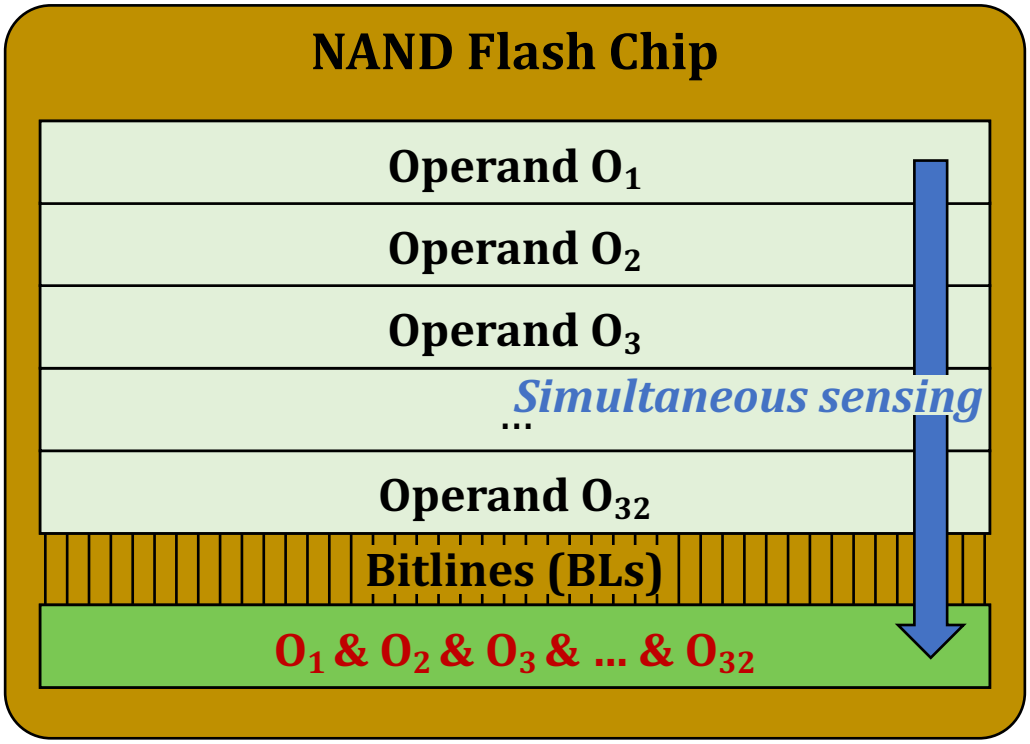
- Performs computation inside NAND flash chips



IFP fundamentally mitigates data movement

Our Proposal: Flash-Cosmos

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



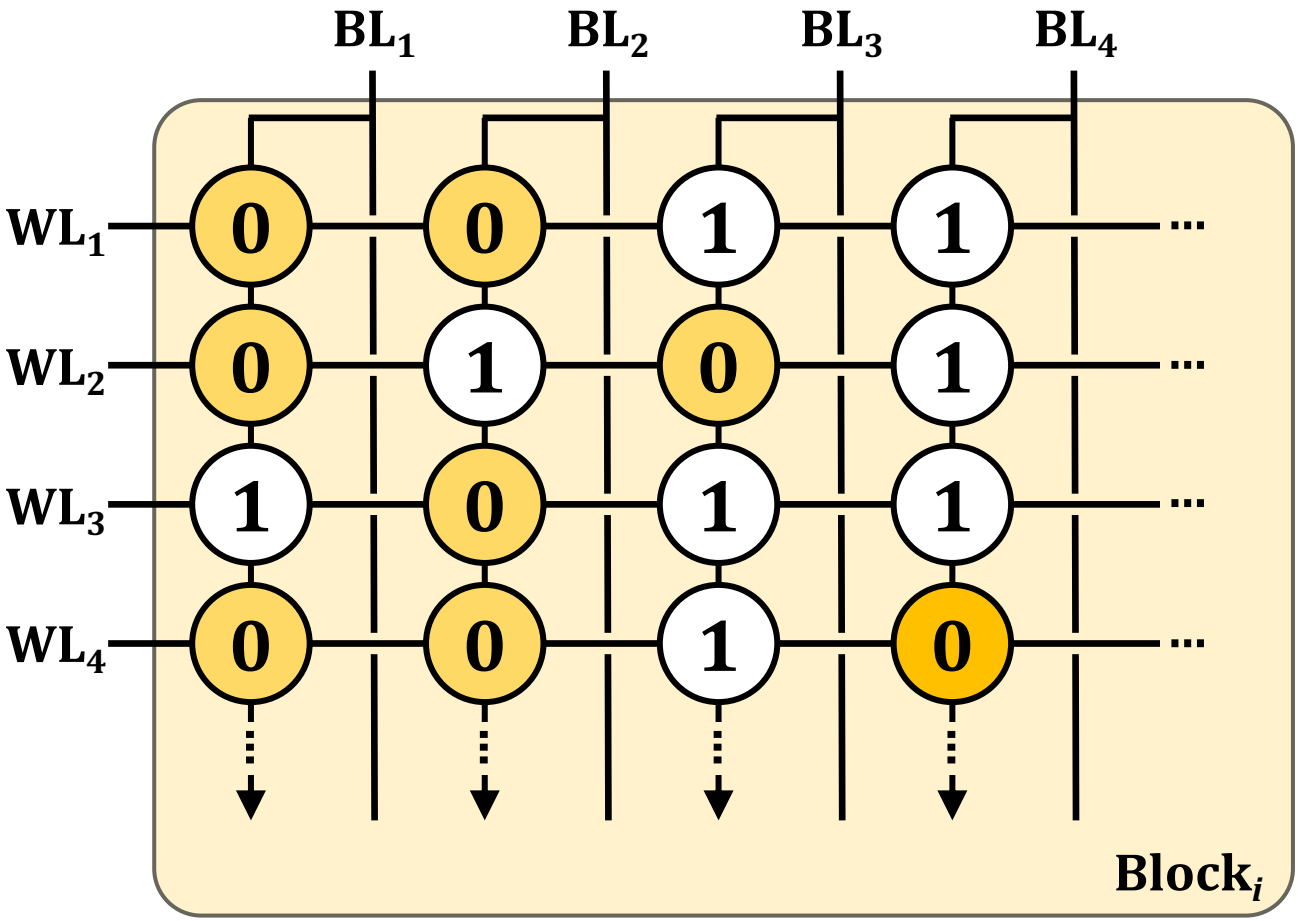
Key Ideas of Flash-Cosmos

Multi-Wordline Sensing (MWS)
to enable in-flash bulk bitwise operations via a single sensing operation

Enhanced SLC-Mode Programming (ESP)
to eliminate raw bit errors in stored data (and thus in computation results)

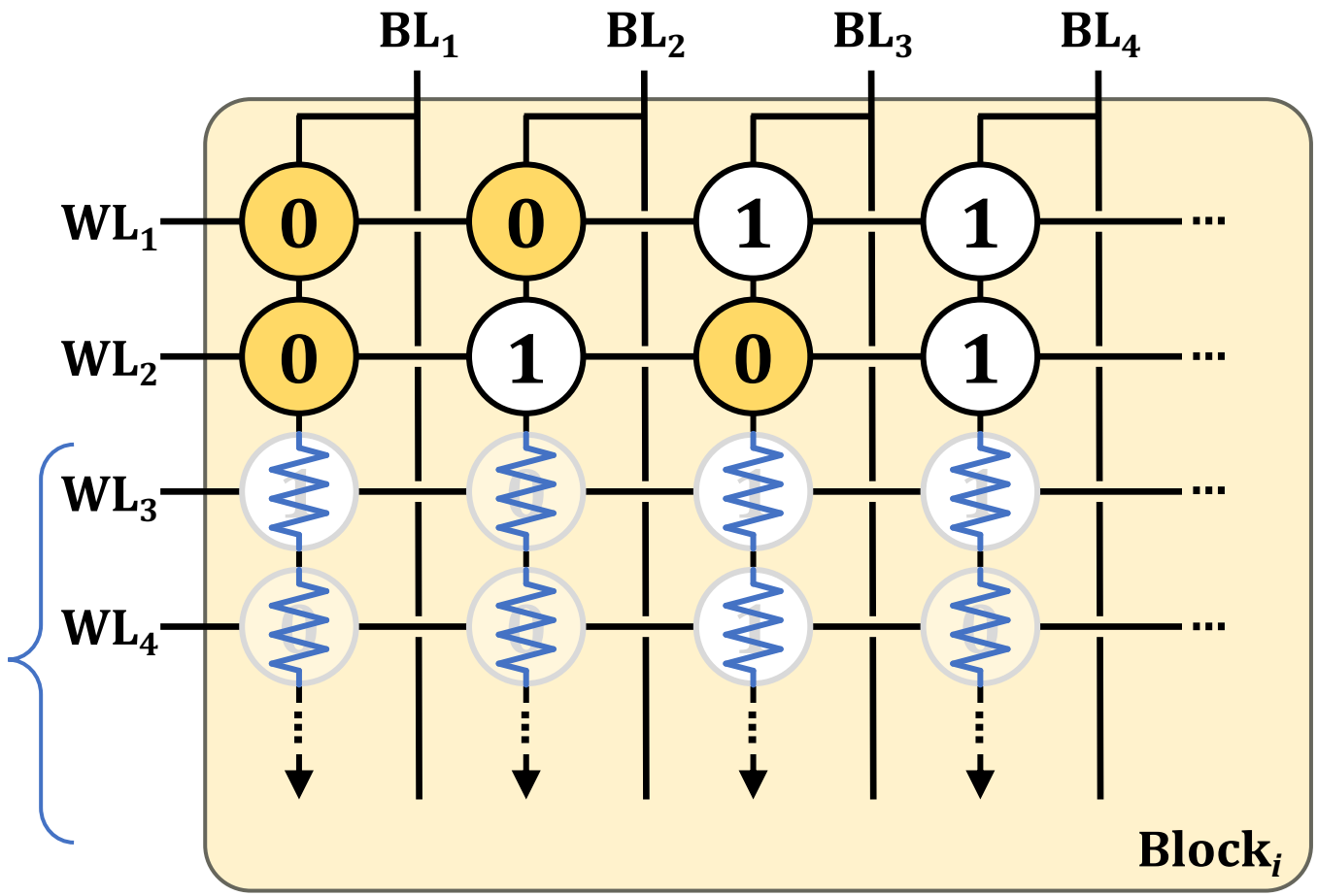
Multi-Wordline Sensing (MWS): Bitwise AND

- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs



Multi-Wordline Sensing (MWS): Bitwise AND

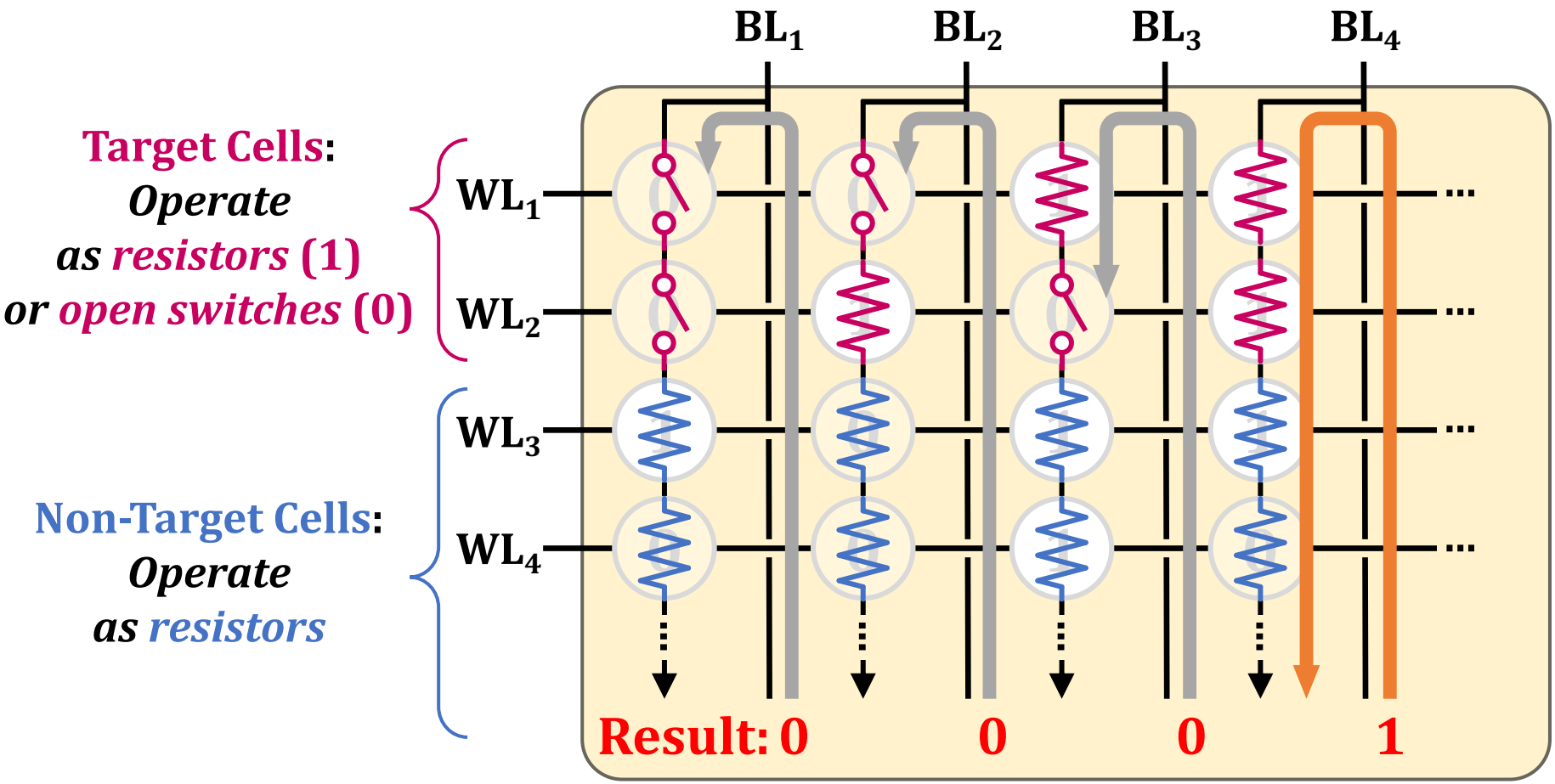
- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs



Non-Target Cells:
Operate as resistors

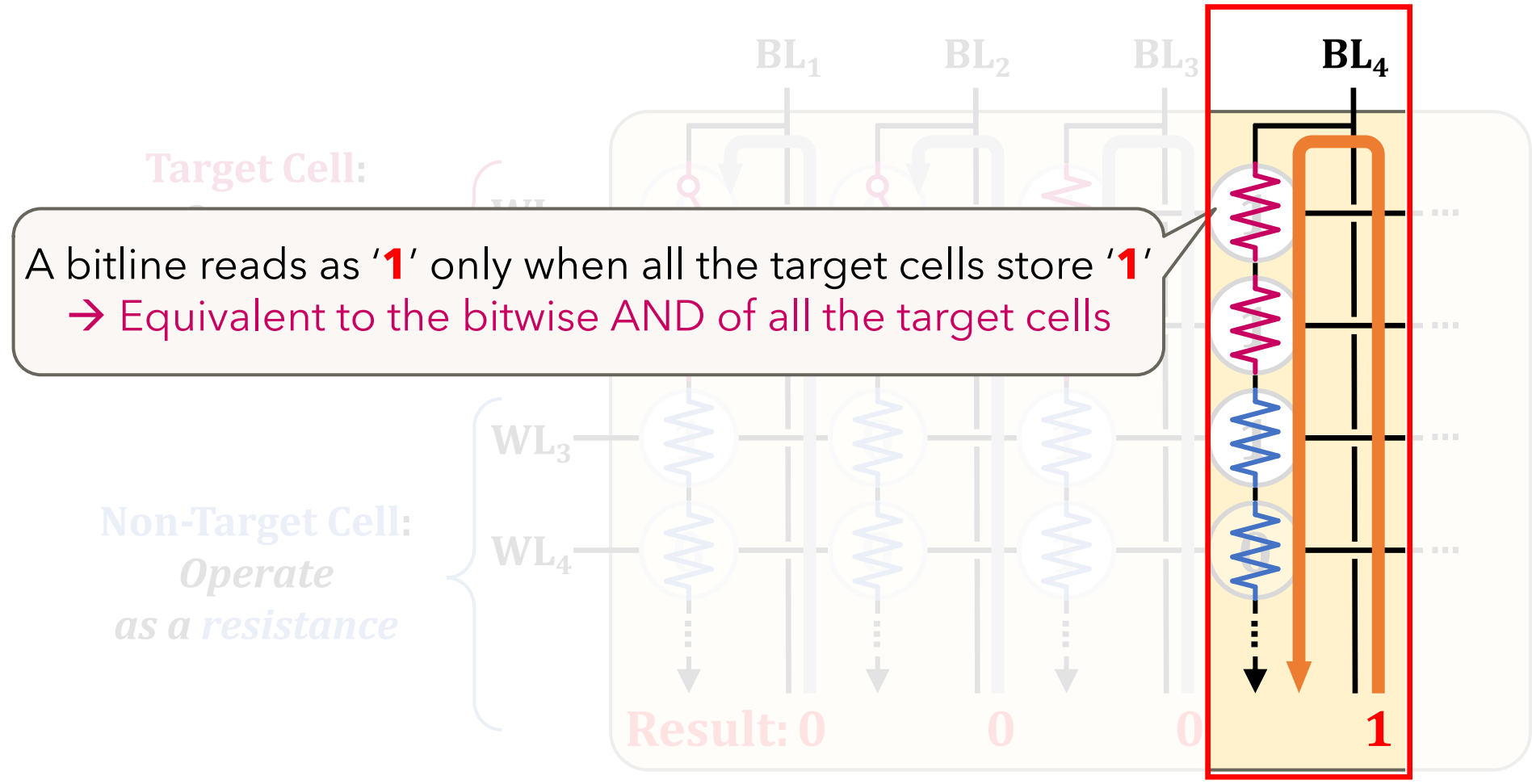
Multi-Wordline Sensing (MWS): Bitwise AND

- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs



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

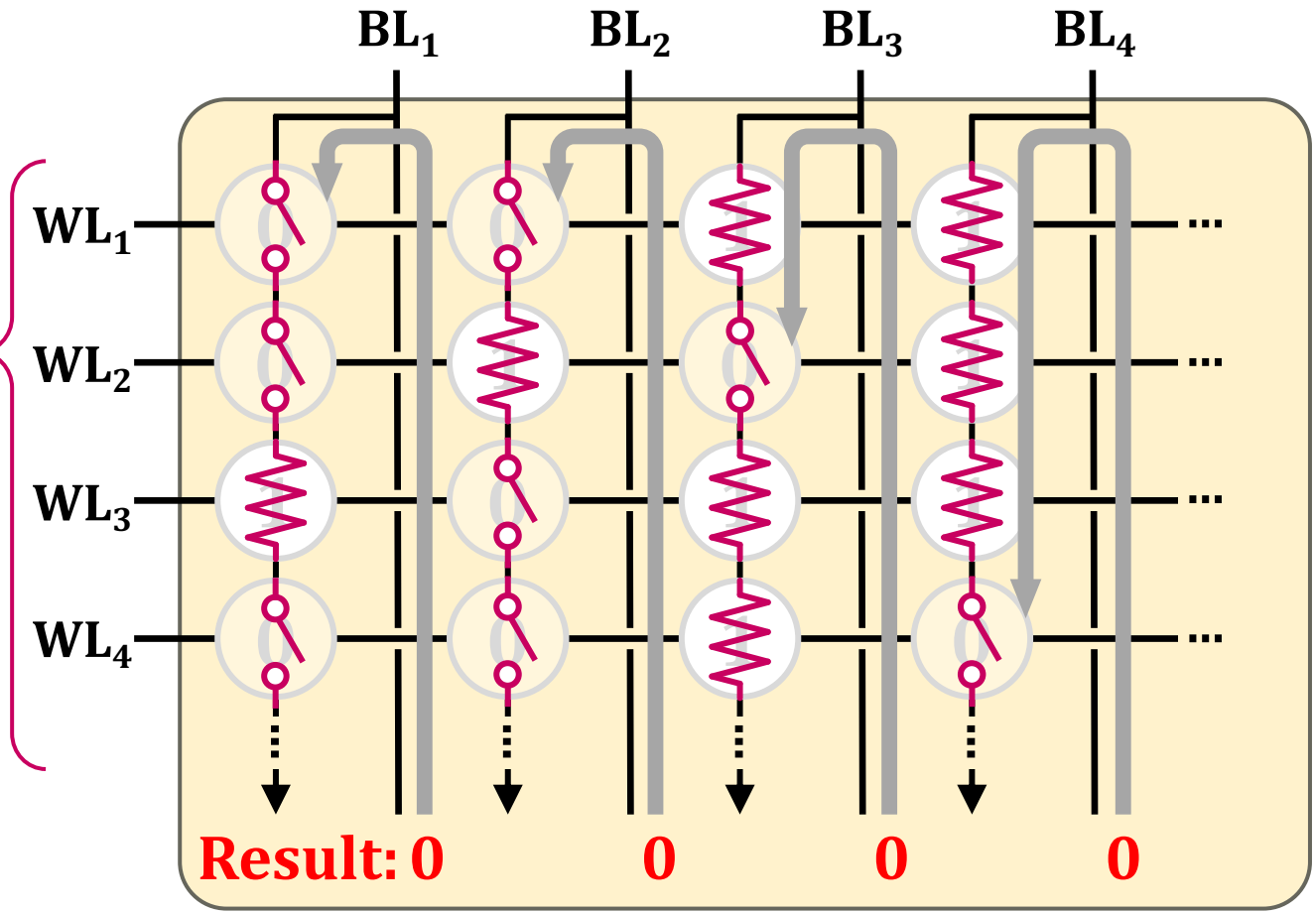


A bitline reads as '**1**' only when all the target cells store '**1**'
→ Equivalent to the bitwise AND of all the target cells

Multi-Wordline Sensing (MWS): Bitwise AND

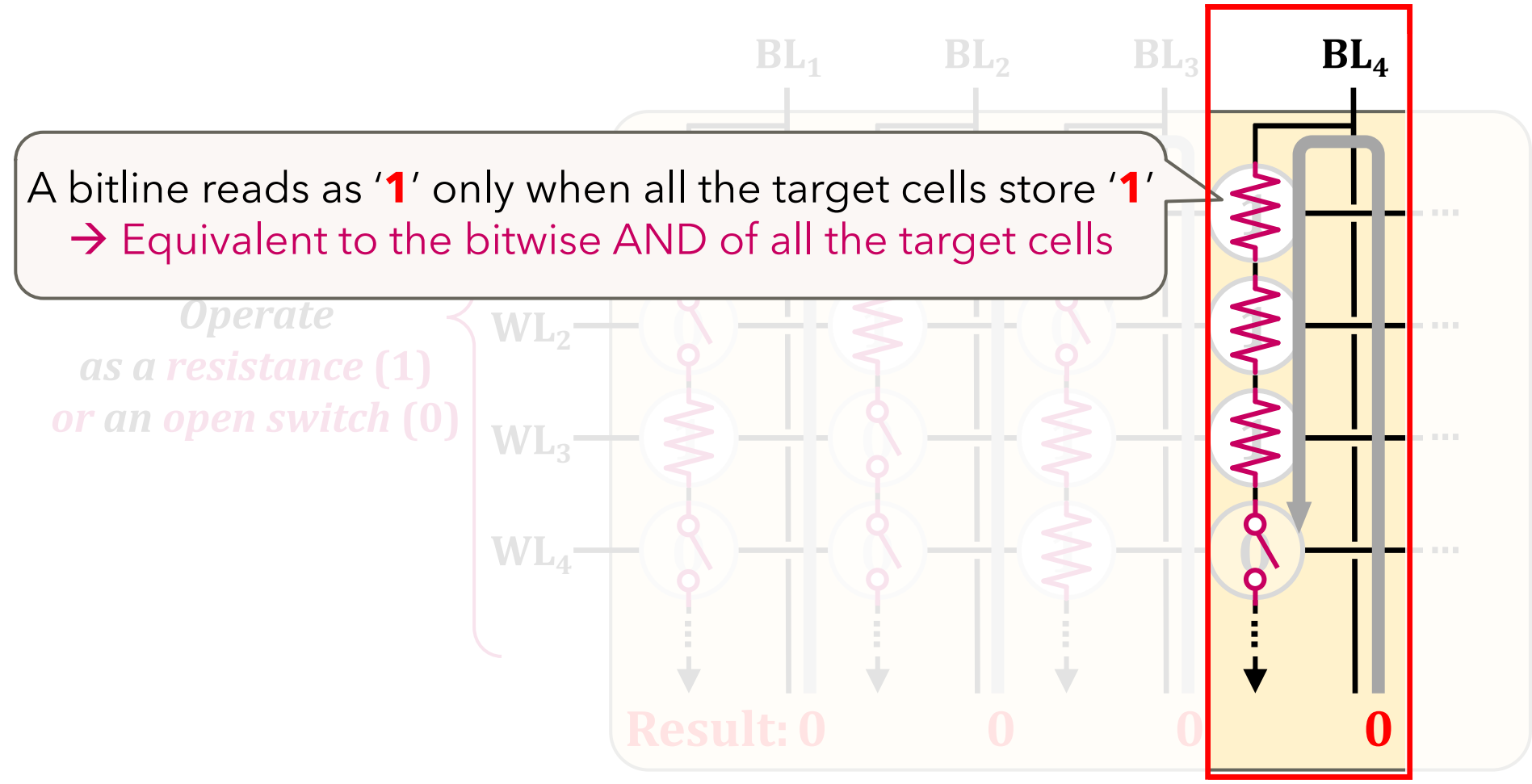
- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs

Target Cells:
Operate
***as resistors* (1)**
***or open switches* (0)**



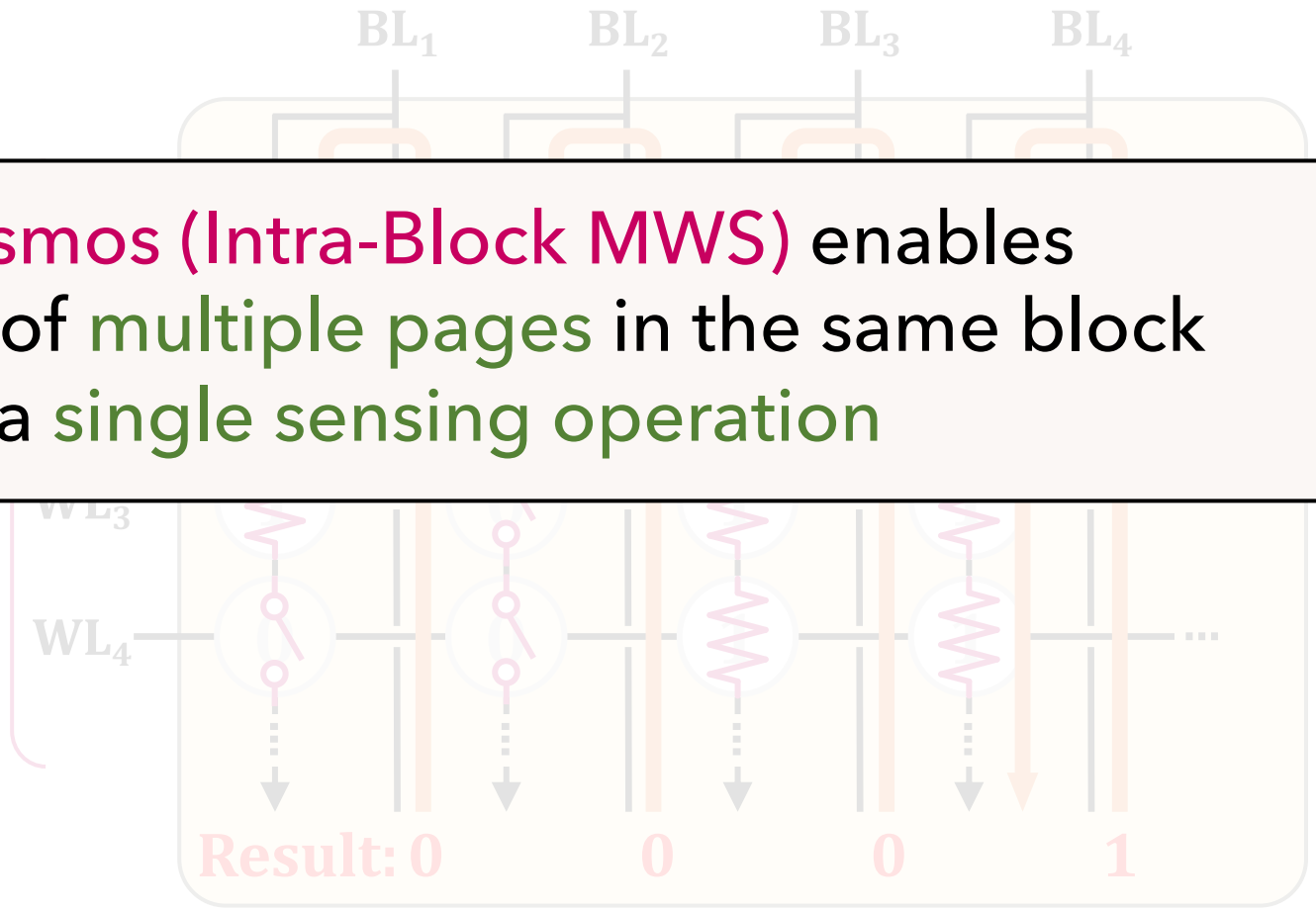
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs



Multi-Wordline Sensing (MWS): Bitwise AND

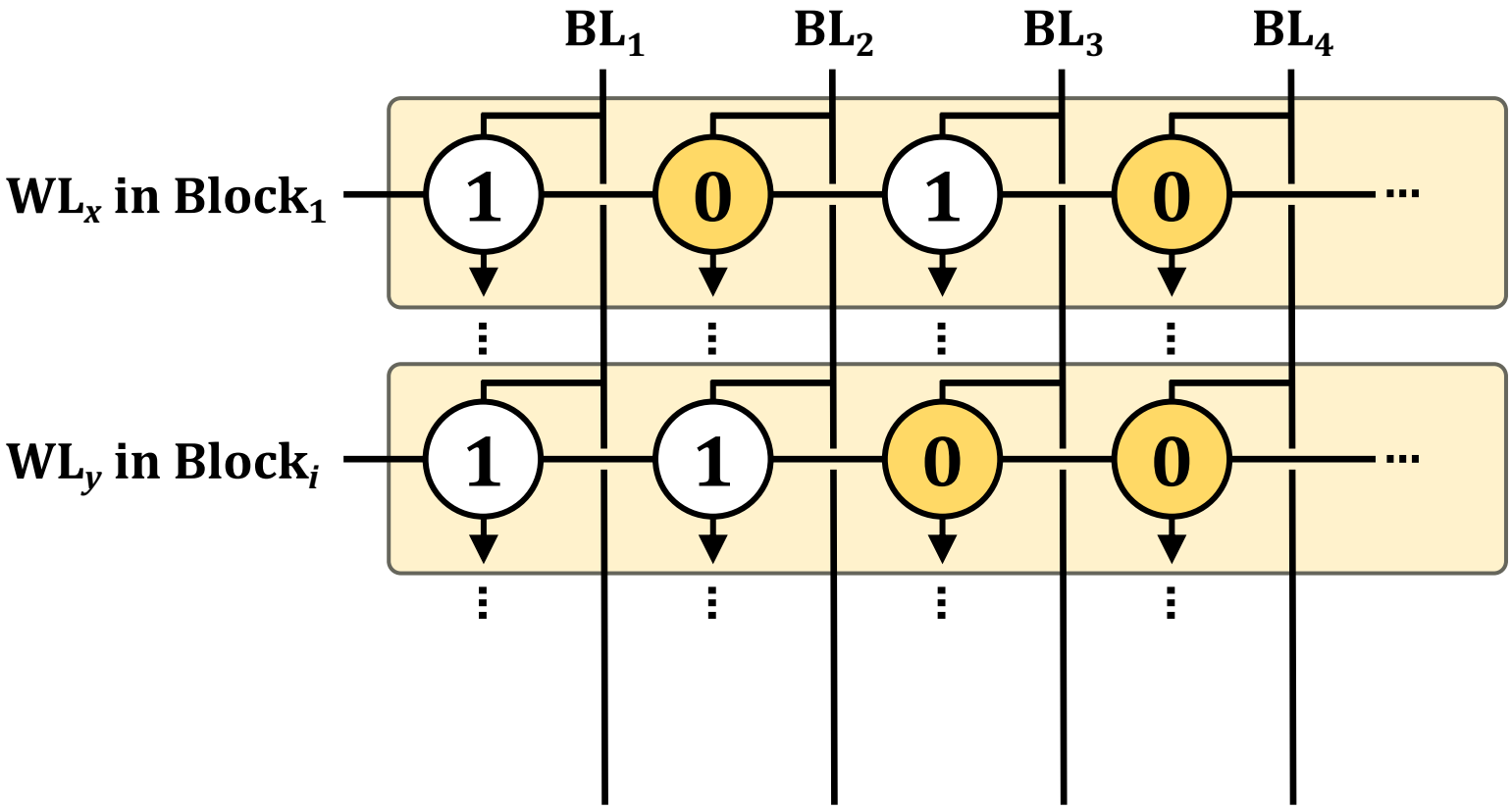
- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs



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

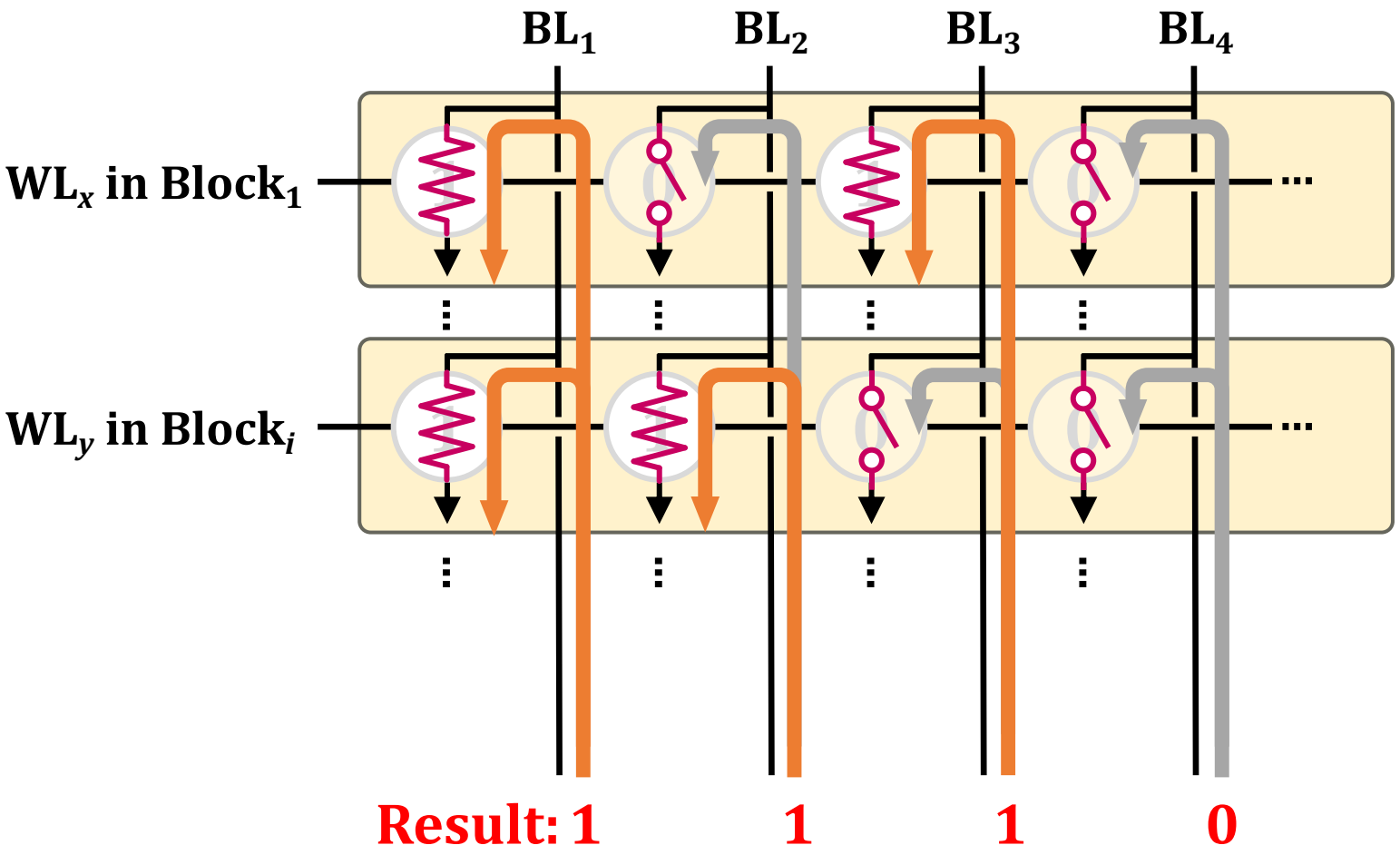
Multi-Wordline Sensing (MWS): Bitwise OR

- Inter-Block MWS:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs



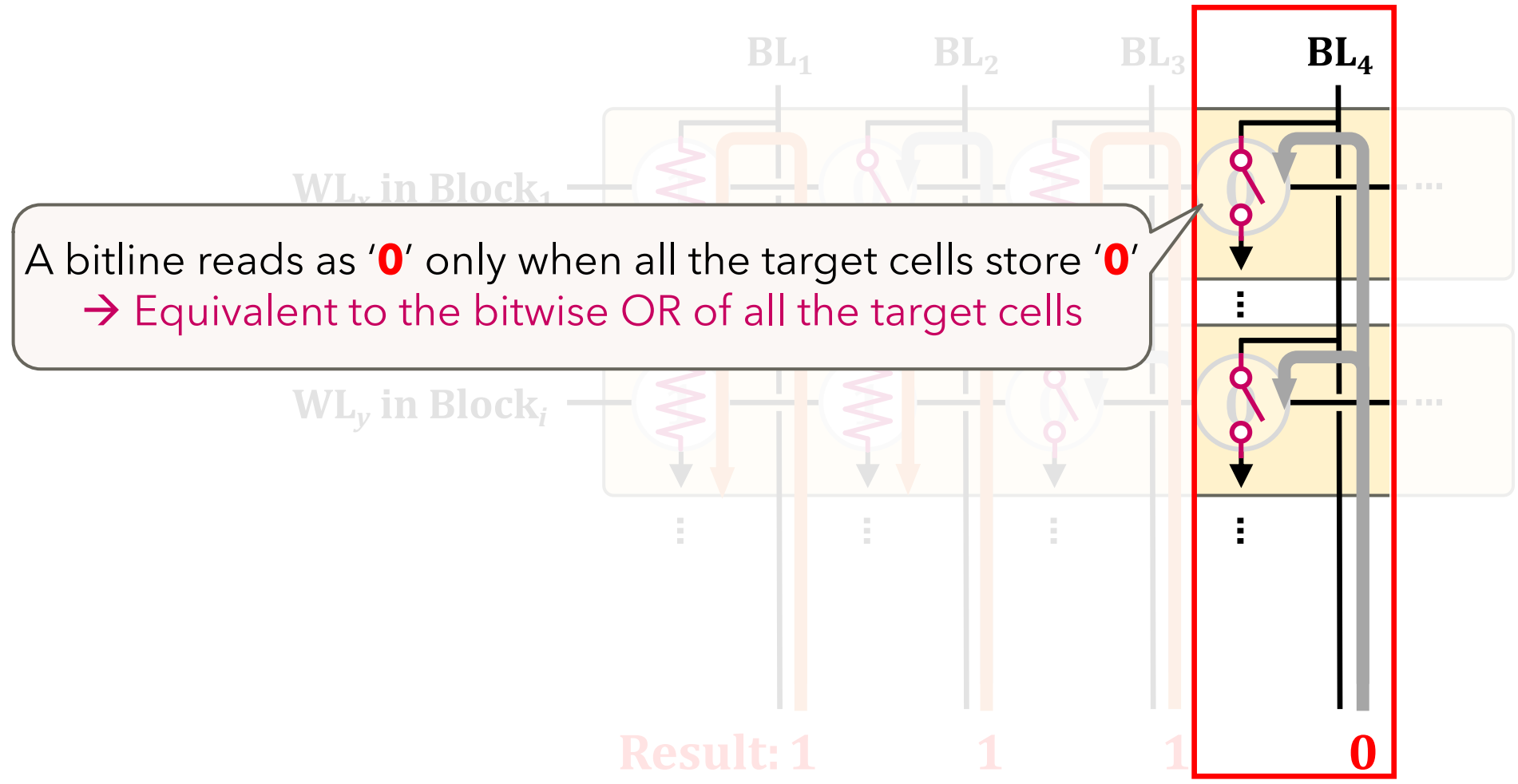
Multi-Wordline Sensing (MWS): Bitwise OR

- Inter-Block MWS:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs



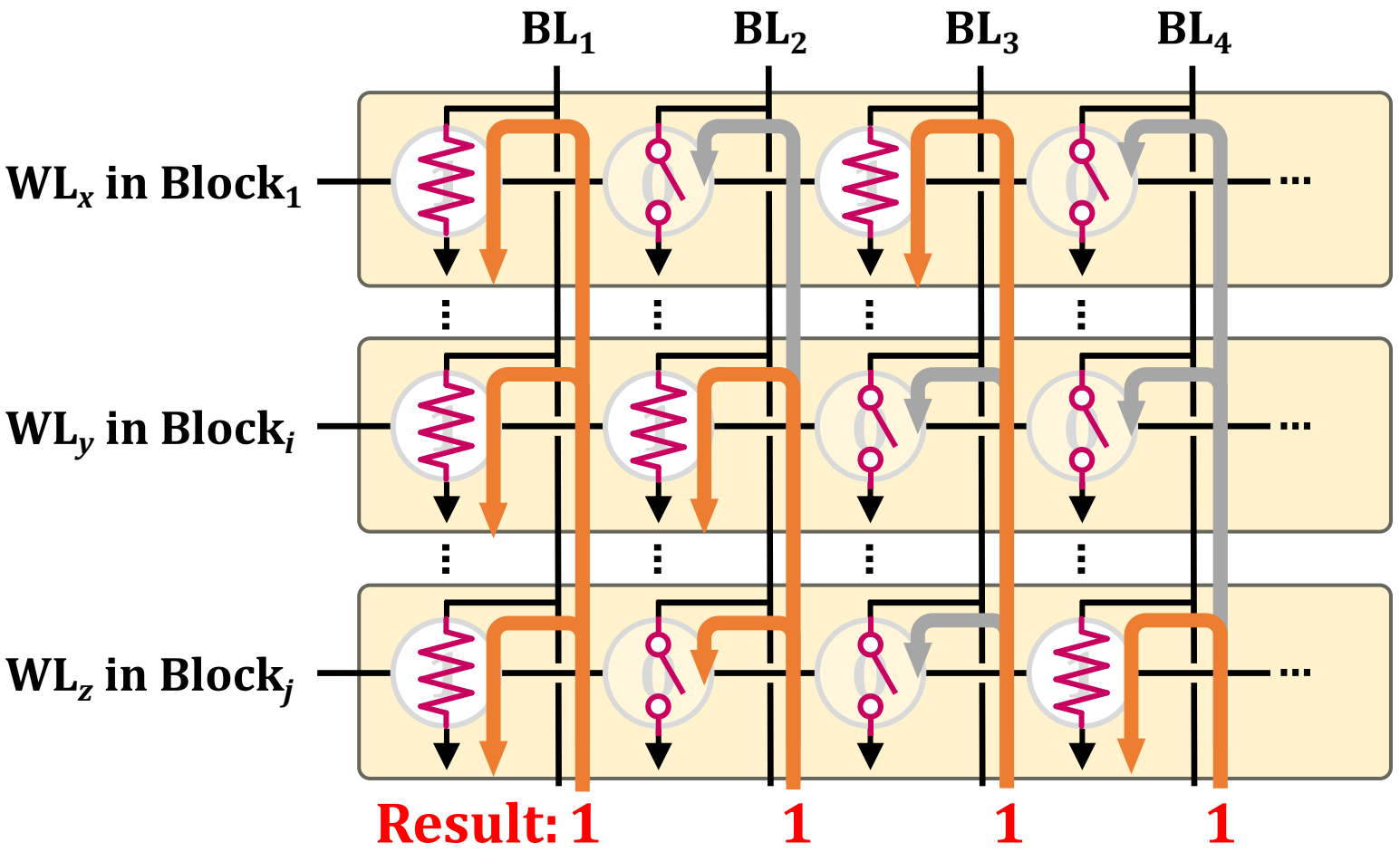
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS**:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs



Multi-Wordline Sensing (MWS): Bitwise OR

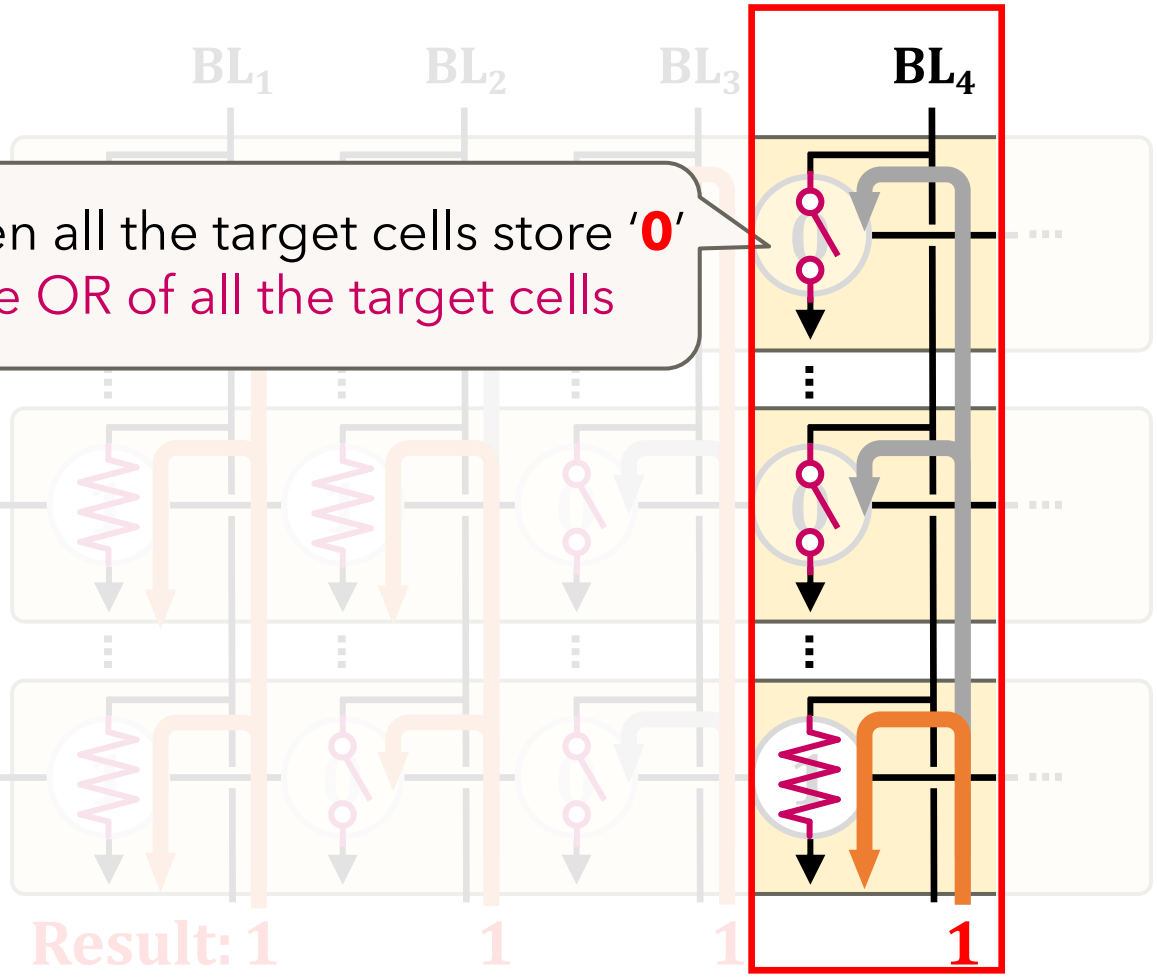
- Inter-Block MWS:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs



Multi-Wordline Sensing (MWS): Bitwise OR

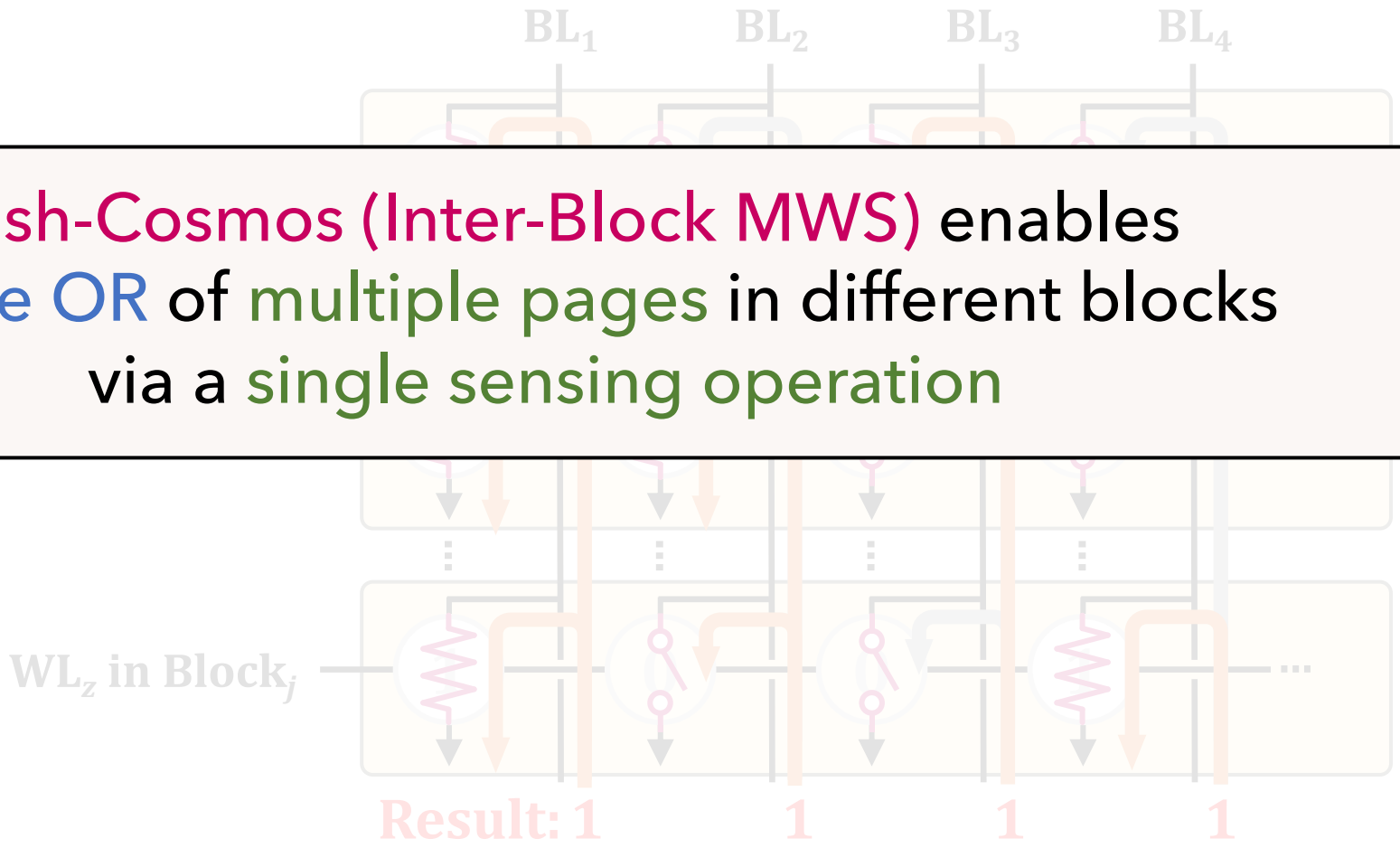
- **Inter-Block MWS**:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs

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

Key Ideas

Multi-Wordline Sensing (MWS)
to enable in-flash bulk bitwise operations
via a single sensing operation

Enhanced SLC-Mode Programming (ESP)
to eliminate raw bit errors in stored data
(and thus in computation results)

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 computaion

Maximizes the reliability margin between the different states of flash cells

Enhanced SLC-Mode Programming (ESP)

- To eliminate raw bit errors in stored data (and computation results)

Flash-Cosmos (ESP) enables
reliable in-flash computation
by trading storage density & programming latency

Storage & latency overheads affect
only data used in in-flash computation

Evaluation Methodology

- **Real-device characterization**
  - To validate the feasibility and reliability of Flash-Cosmos
  - Using 160 48-WL-layer 3D Triple-Level Cell NAND flash chips
    - 3,686,400 tested wordlines
  - Under worst-case operating conditions
    - Under a 1-year retention time at 10K P/E cycles
    - Worst-case data patterns

- **System-level evaluation**
  - Using the state-of-the-art SSD simulator (MQSim [Tavakkol+, FAST'18])
  - Three real-world applications
    - Bitmap Indices (BMI): Bitwise AND of up to ~1,000 operands
    - Image Segmentation (IMS): Bitwise AND of 3 operands
    - K-clique Star Listing (KCS): Bitwise OR of up to 32 operands
  - Baselines
    - Outside-Storage Processing (OSP): A multi-core CPU (Intel i7-11700K)
    - In-Storage Processing (ISP): An in-storage hardware accelerator
    - ParaBit [Gao+, MICRO'21]: State-of-the-art in-flash processing mechanism

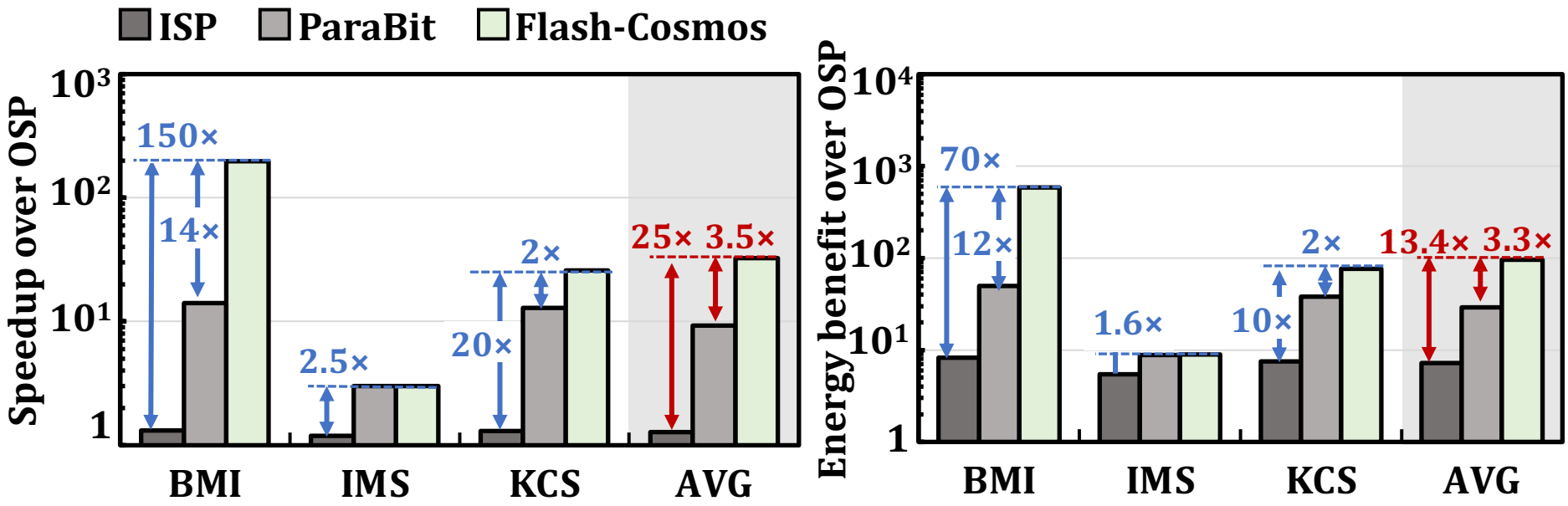
Results: Real-Device Characterization

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

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*

https://arxiv.org/pdf/2209.05566.pdf

More to Come…

Concluding Remarks

Challenge and Opportunity for Future

Fundamentally
Energy-Efficient
(Data-Centric)
Computing Architectures

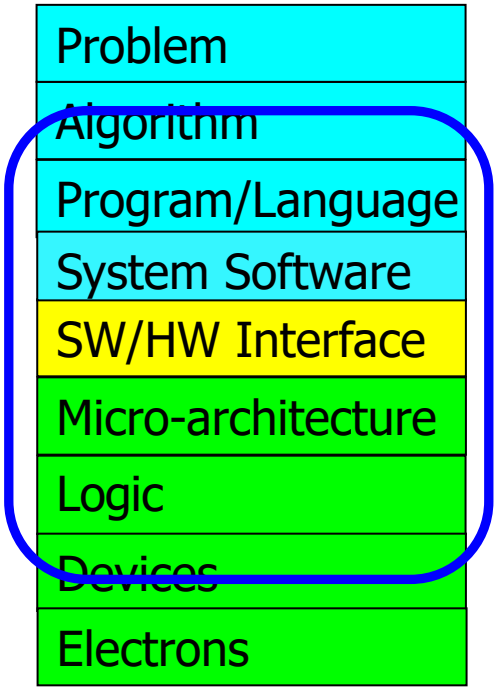
Challenge and Opportunity for Future

Fundamentally
High-Performance
(Data-Centric)
Computing Architectures

Computing Architectures with Minimal Data Movement

We Need to Revisit the Entire Stack

- With a **storage-centric mindset**



We can get there step by step

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

An Overview Paper

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.

https://arxiv.org/pdf/2012.03112.pdf

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/

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 80 | 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++ ☆ 583 ⑂ 209

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 ☆ 137 ⑂ 50

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++ ☆ 277 ⑂ 149

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 ☆ 217 ⑂ 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 ☆ 127 ⑂ 28

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++ ☆ 117 ⑂ 36

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

Acknowledgments

SAFARI

SAFARI Research Group

safari.ethz.ch

Think BIG, Aim HIGH!

https://safari.ethz.ch

SAFARI Newsletter June 2023 Edition

- https://safari.ethz.ch/safari-newsletter-june-2023/

SAFARI
SAFARI Research Group

Think Big, Aim High

ETH zürich

View in your browser
June 2023

SAFARI Newsletter July 2024 Edition

- **https://safari.ethz.ch/safari-newsletter-july-2024/**

PIM Tutorial November 2024 Edition

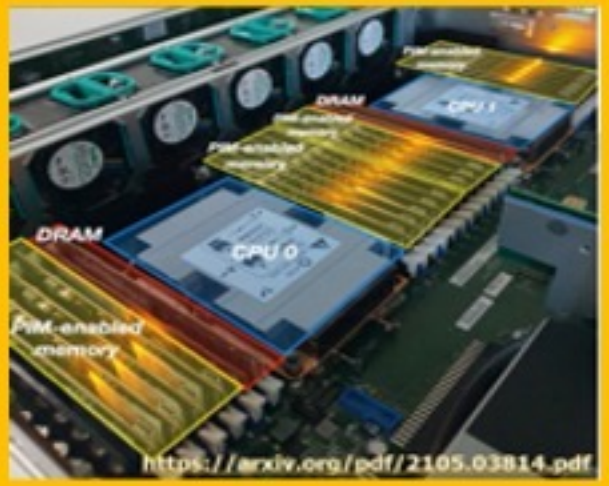
MICRO 2024 - Tutorial on Memory-Centric Computing Systems

Saturday, November 2nd, Austin, Texas, USA

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

Program: https://events.safari.ethz.ch/micro24-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/micro24-memorycentric-tutorial/

Storage-Centric Computing

Enabling Fundamentally-Efficient Computers

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

1 December 2024

CCF China Storage Keynote Talk

SAFARI

ETH zürich

Backup Slides

Processing in Memory: Two Types

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

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
  - More complex computation using Lookup Tables
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating (multiple) rows performs computation
    - Even in commodity off-the-shelf DRAM chips!

- 30X-257X performance and energy improvements

Seshadri+"RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," 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.

In-DRAM Acceleration of Database Queries

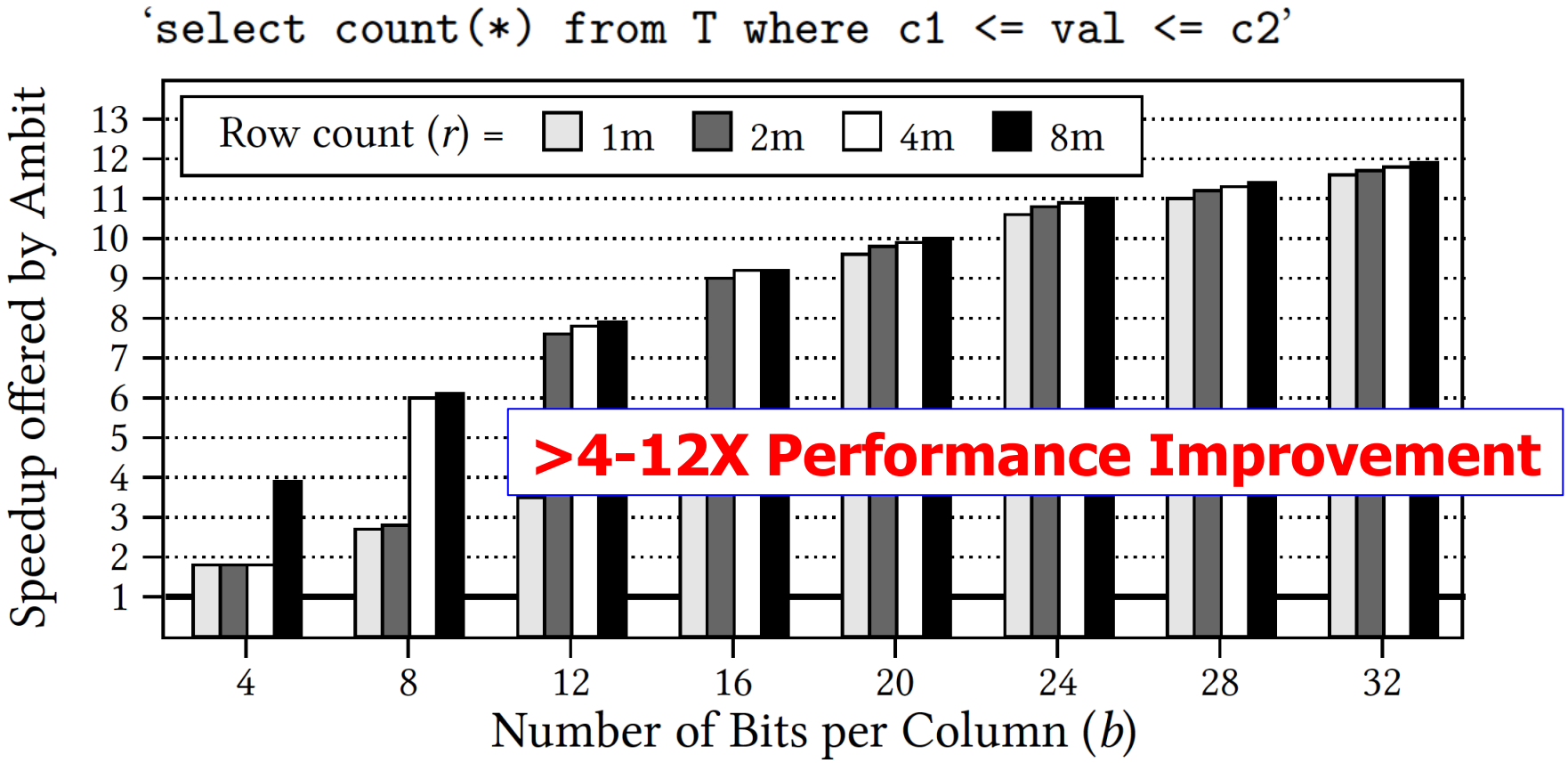


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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

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 Processing-in-DRAM

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

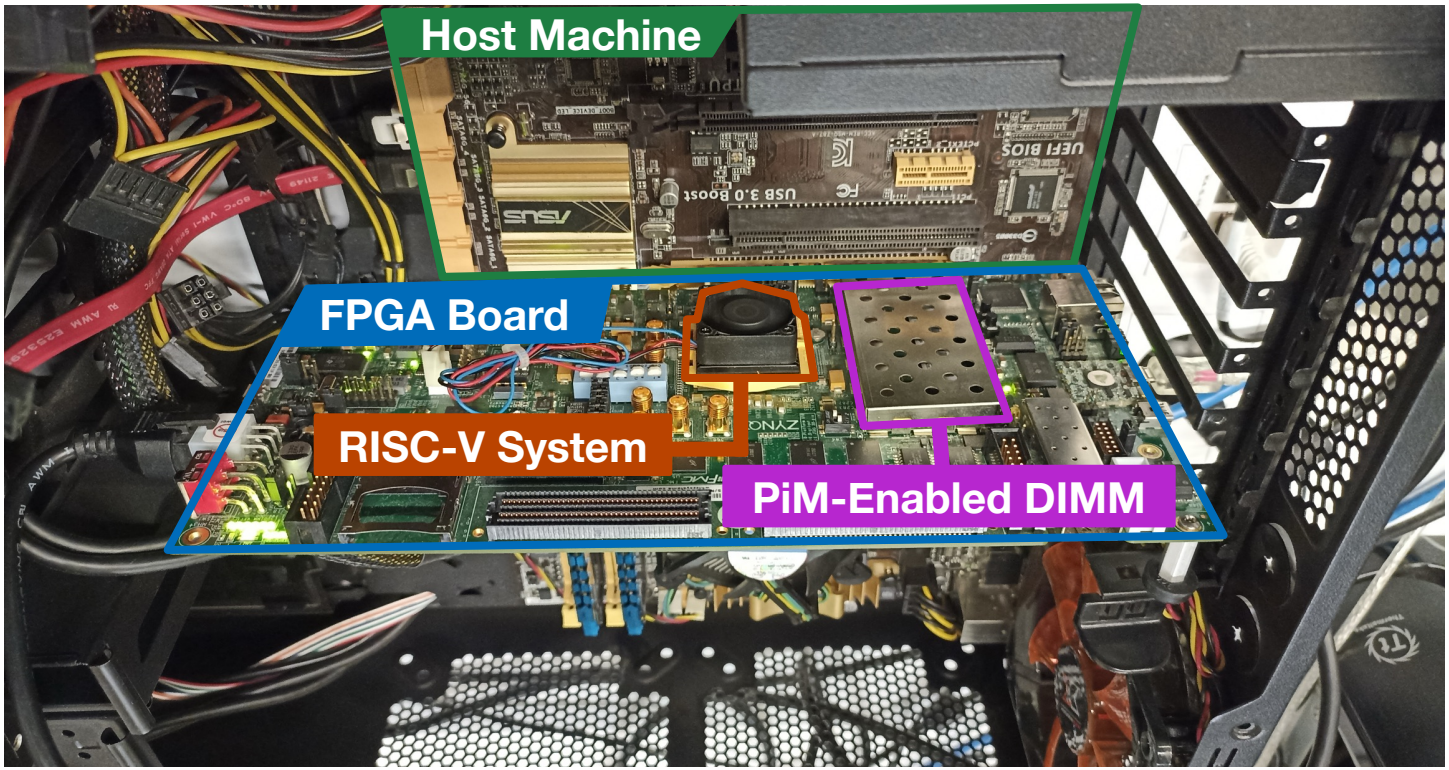
[§]ETH Zürich [†]TOBB ETÜ [*]BSC

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:

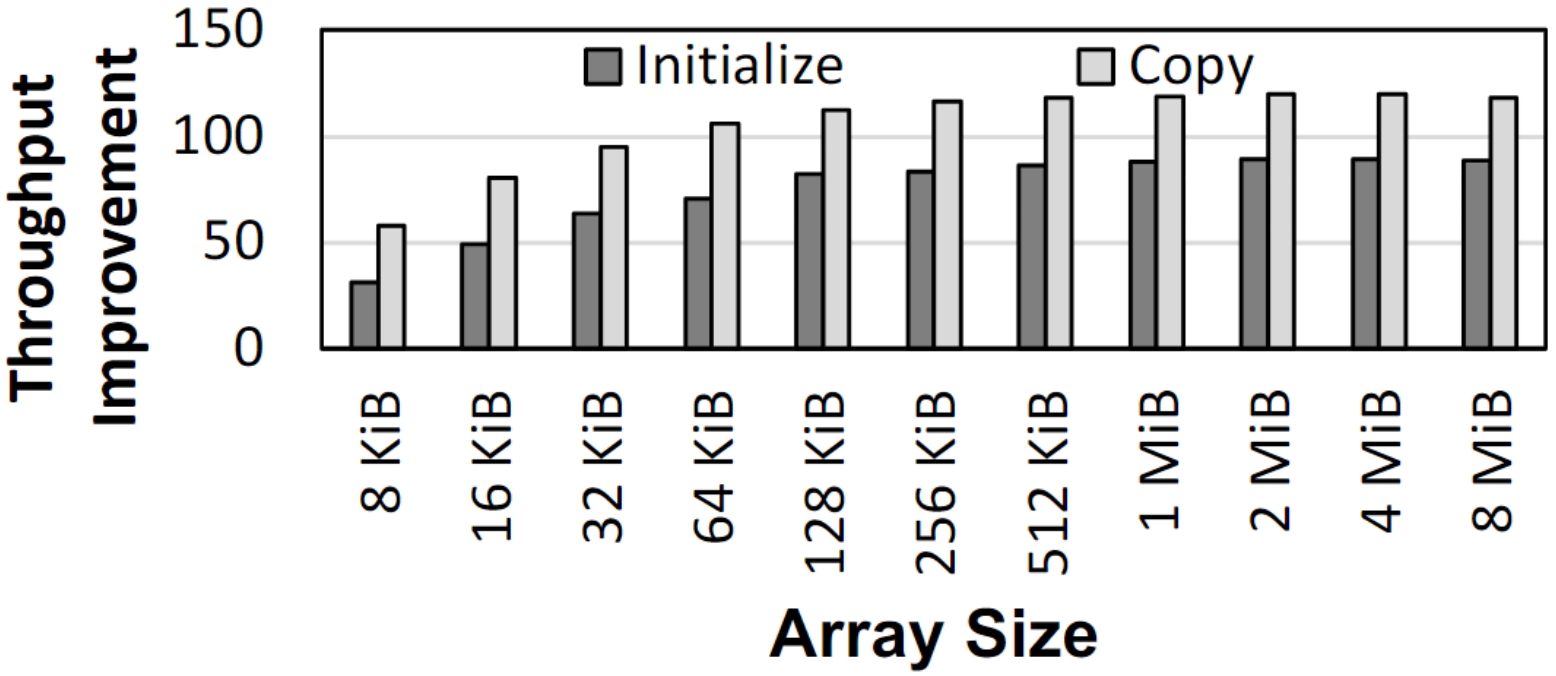
1- Open IP Catalog
2- Find "Memory Interface Generator (MIG 7 Series)" IP and double click

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*

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*

The Capability of COTS DRAM Chips

We **demonstrate** that **COTS DRAM chips:**

1 Can **copy one row into up to 31 other rows with >99.98% success rate**

2 Can perform **NOT operation** with up to **32 output operands**

3 Can perform up to **16-input AND, NAND, OR, and NOR** operations

Eliminating the Adoption Barriers

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, …

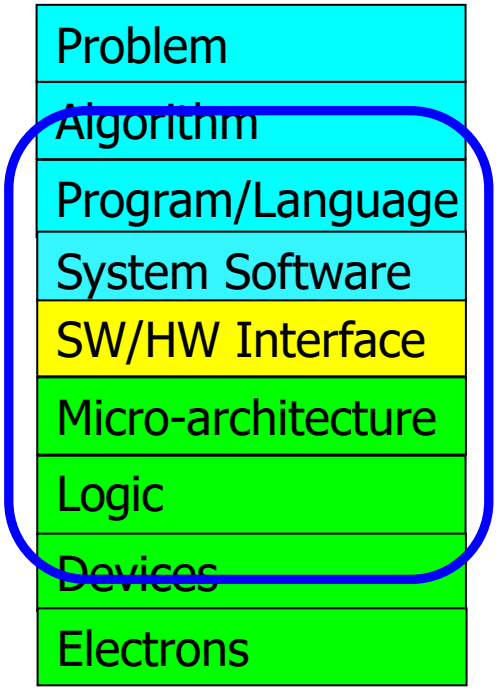
4. **Runtime** and **compilation** systems for adaptive scheduling, data mapping, access/sharing control, …

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**



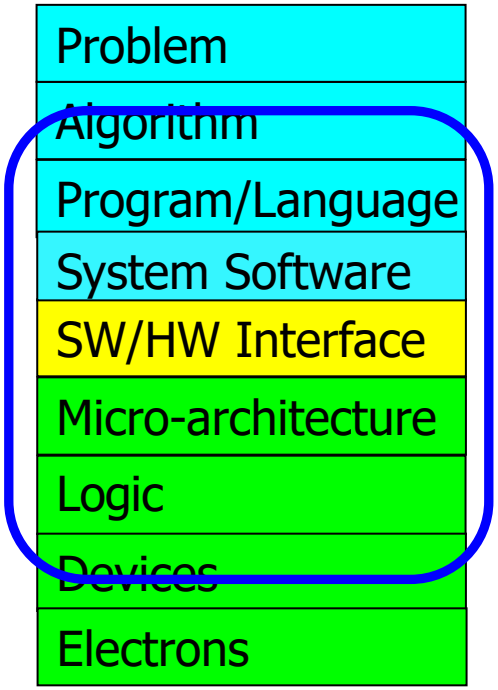
We can get there step by step

Concluding Remarks

- We must design systems to be **balanced**, **high-performance**, **energy-efficient** (all at the same time) → intelligent systems
  - **Data-centric, data-driven, data-aware**
- Enable computation capability inside and close to 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

We Need to Revisit the Entire Stack

- With a **data-centric mindset**



We can get there step by step

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
- ACCESS

Thank you!