1st Workshop on Memory-Centric Computing: Storage-Centric Computing

Mohammad Sadrosadati

m.sadr89@gmail.com

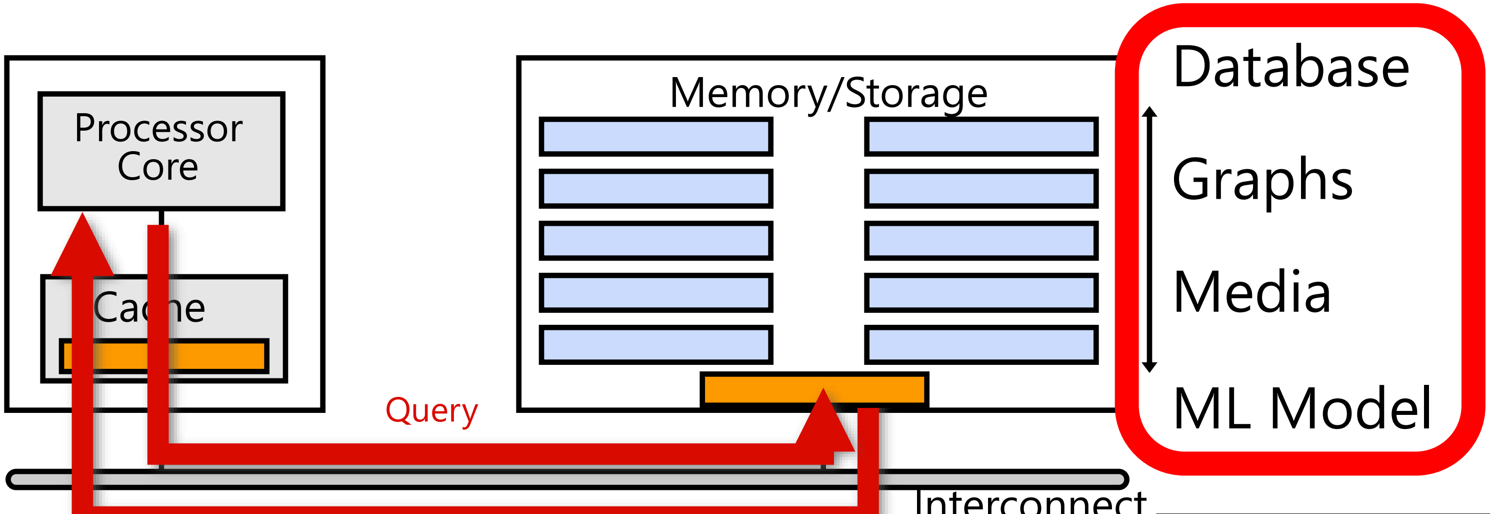
ASPLOS 2025

30 March 2025

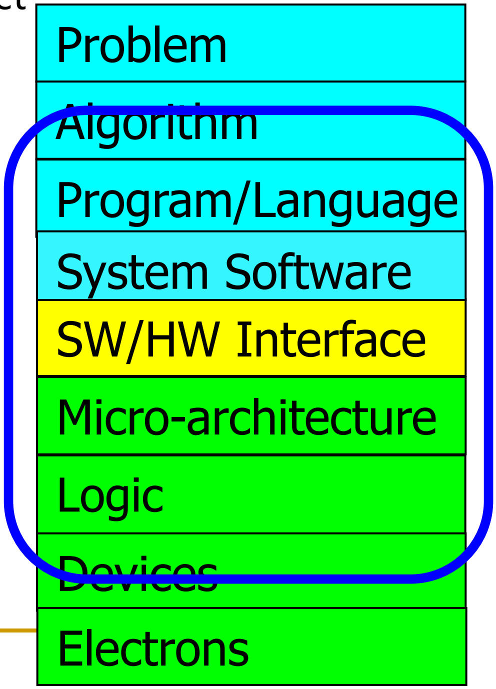
SAFARI

ETH zürich

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

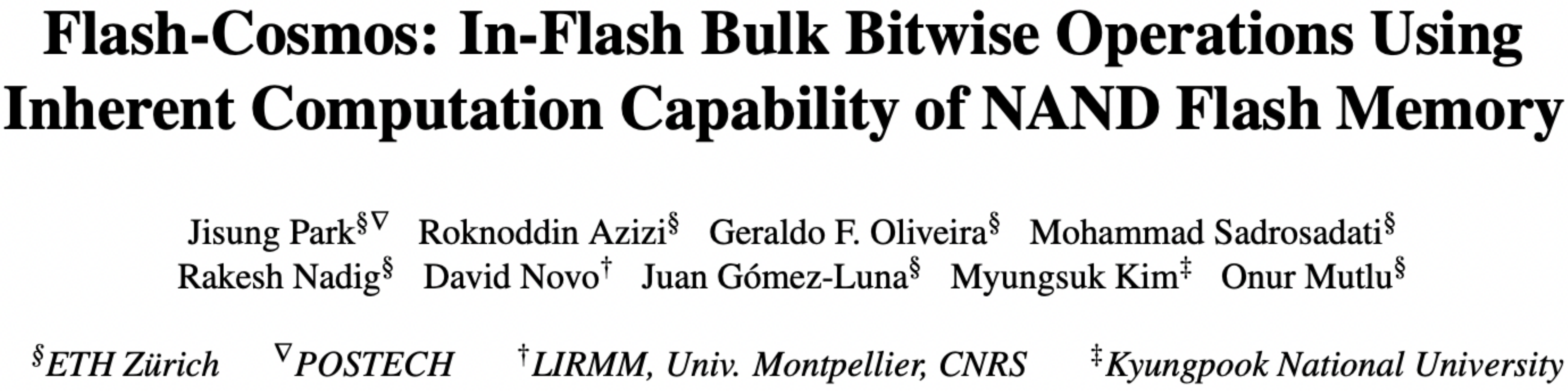
1. Processing **near** Memory
2. Processing **using** Memory

Storage-Centric Computing: Two Types

1. Processing near Storage
2. Processing using Storage

Flash-Cosmos: 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

Talk Outline

Motivation

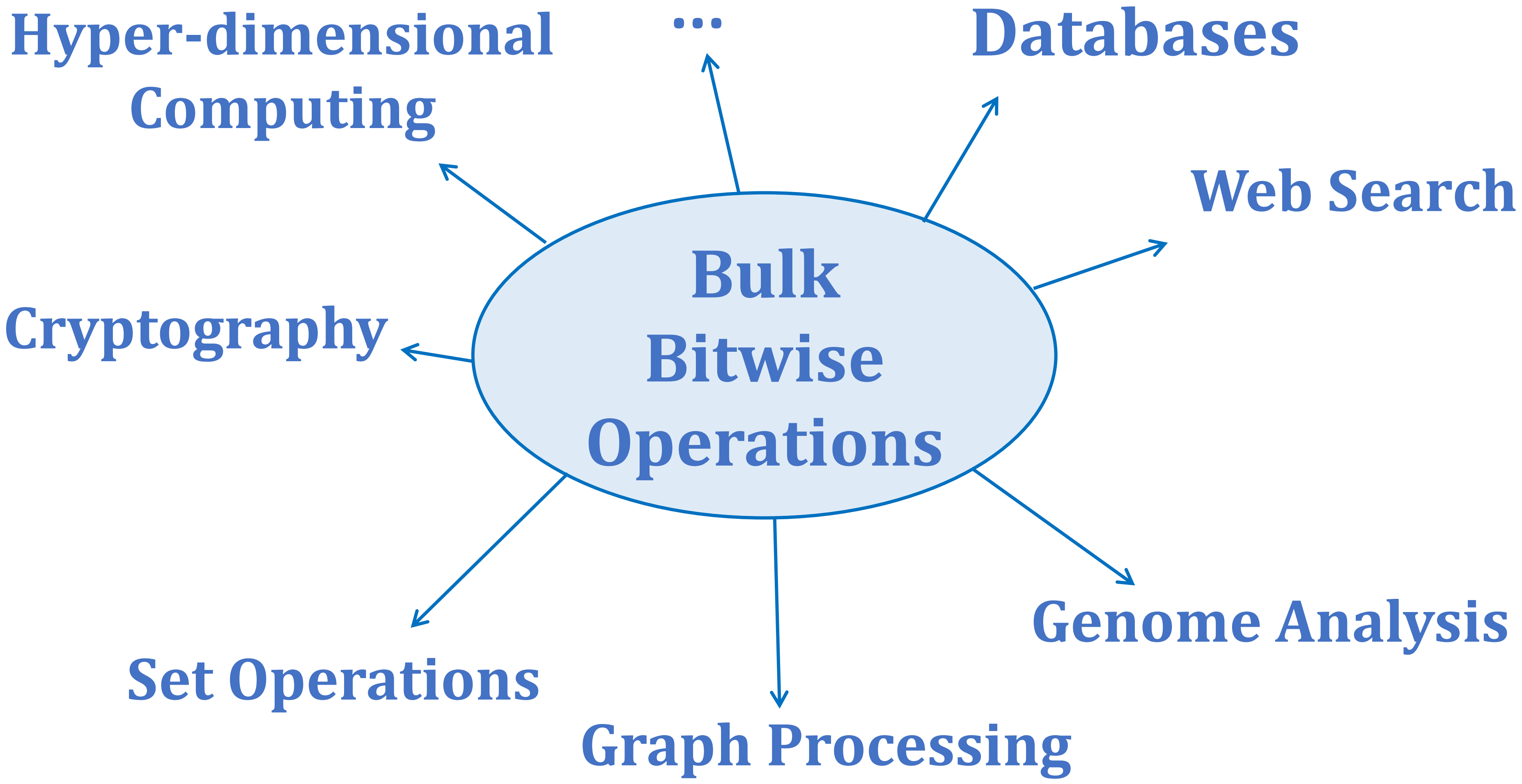
Background

Flash-Cosmos

Evaluation

Summary

Bulk Bitwise Operations



Bulk Bitwise Operations

Hyper-dimensional Computing

…

Databases

Web Search

Cryptography

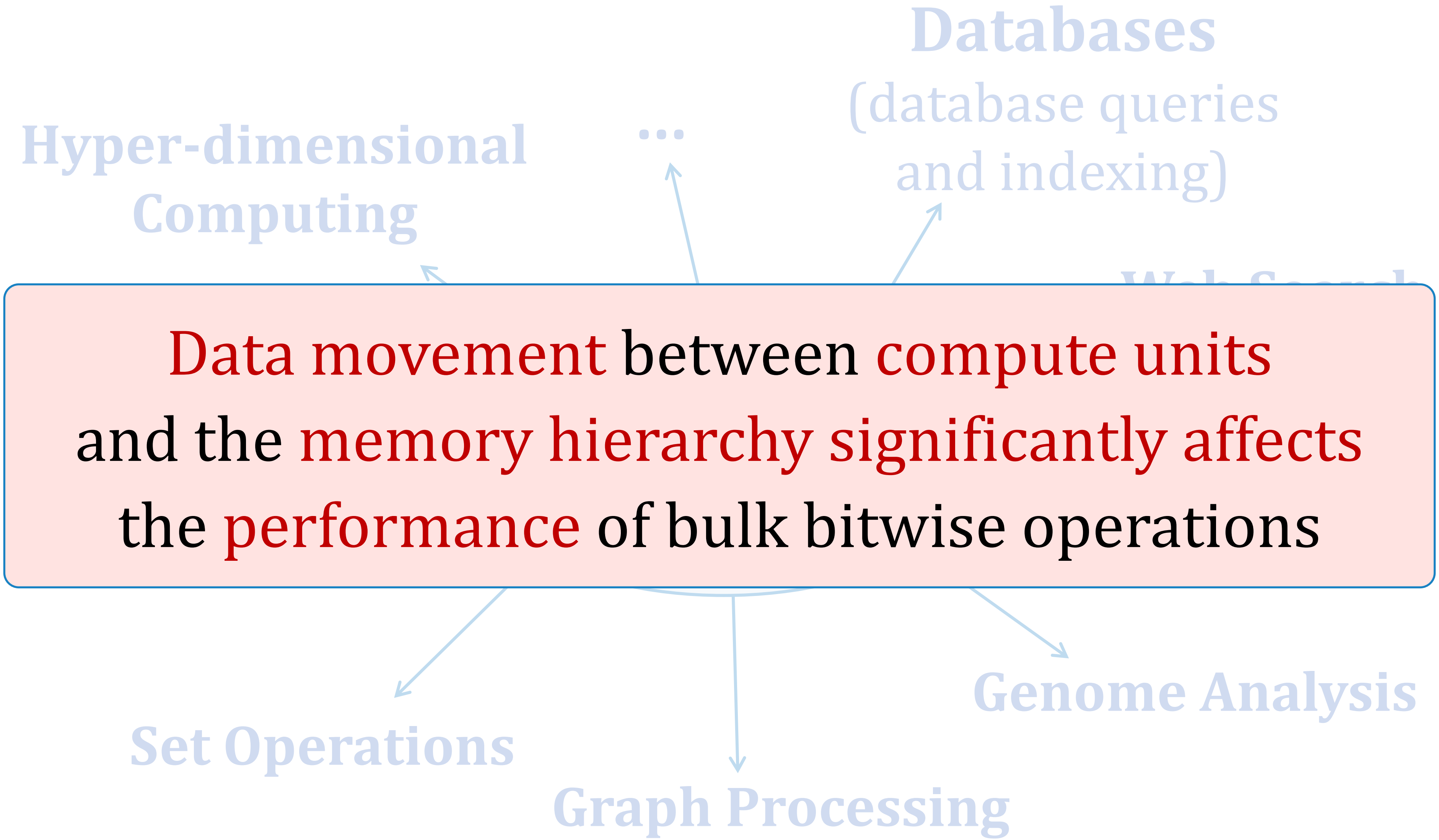
Genome Analysis

Set Operations

Graph Processing

SAFARI

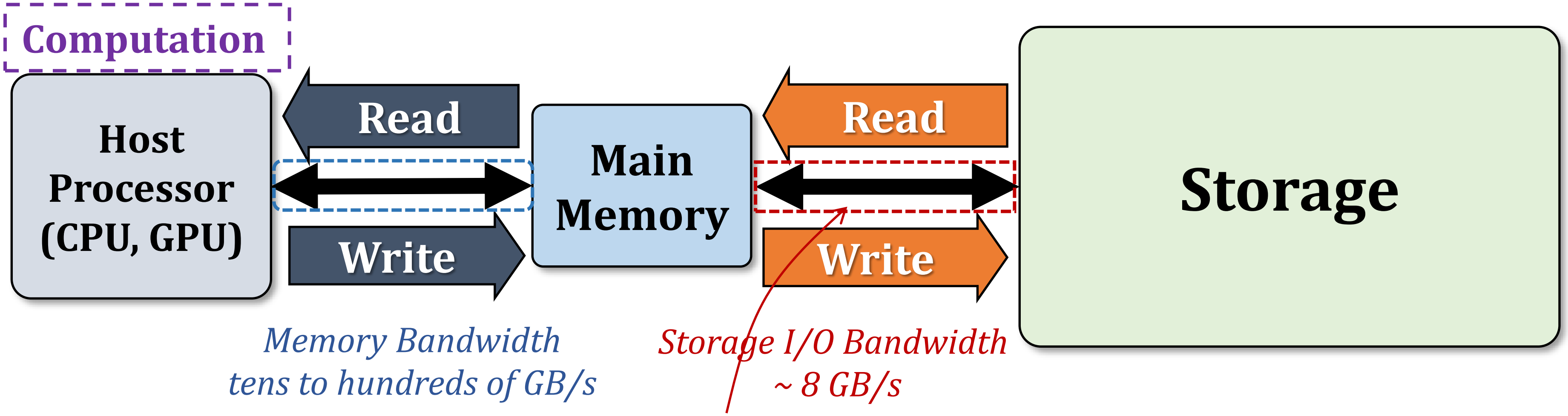
Bulk Bitwise Operations



Data movement between compute units and the memory hierarchy significantly affects the performance of bulk bitwise operations

Data-Movement Bottleneck

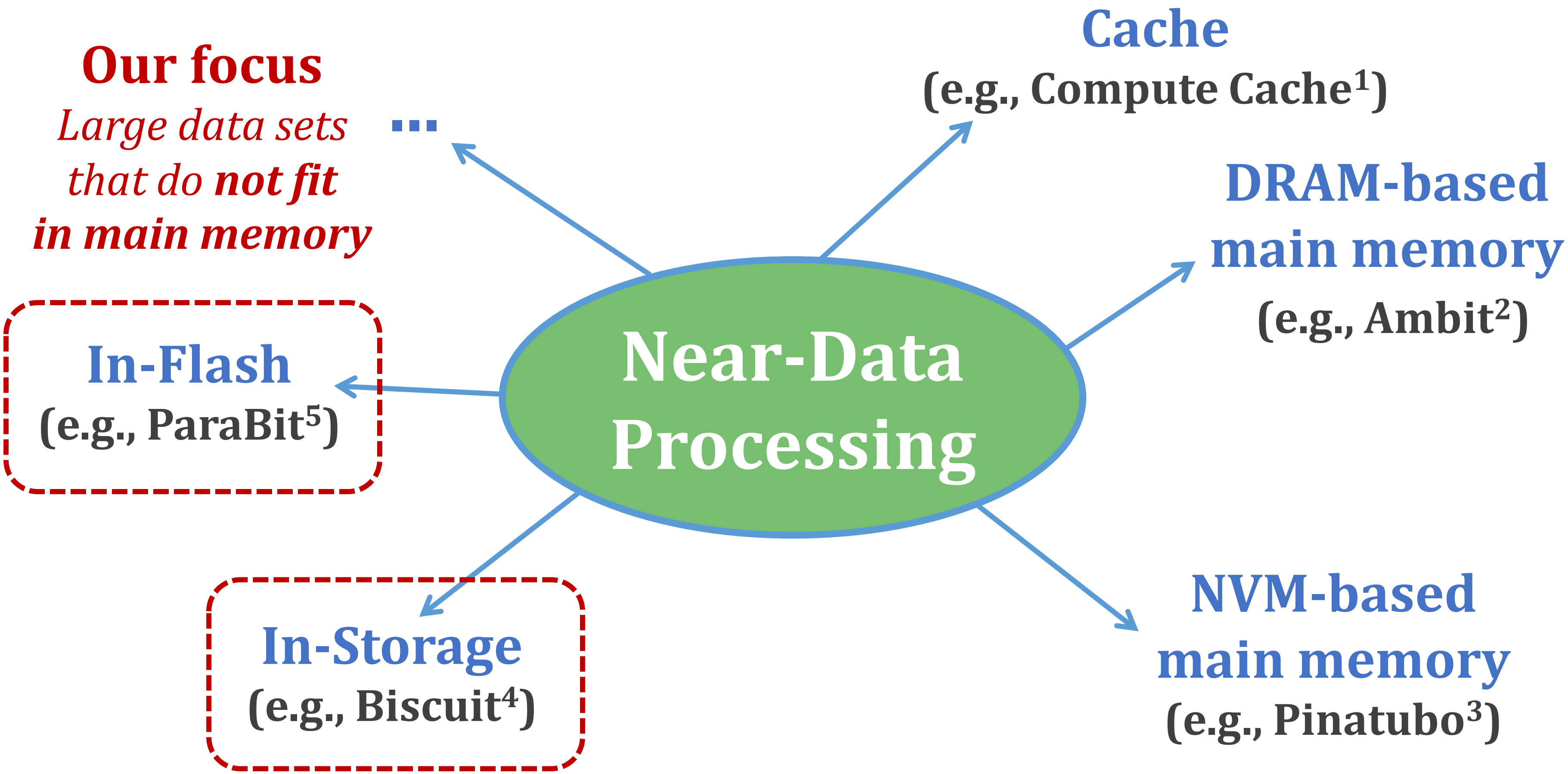
- Conventional systems perform outside-storage processing (OSP) after moving the data to host CPU through the memory hierarchy



Data Movement Bottleneck

The external I/O bandwidth of storage is the main bottleneck for data movement in OSP

NDP for Bulk Bitwise Operations



[1] Aga+, “Compute Caches," HPCA, 2017

[2] Seshadri+, “Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology,” MICRO, 2017

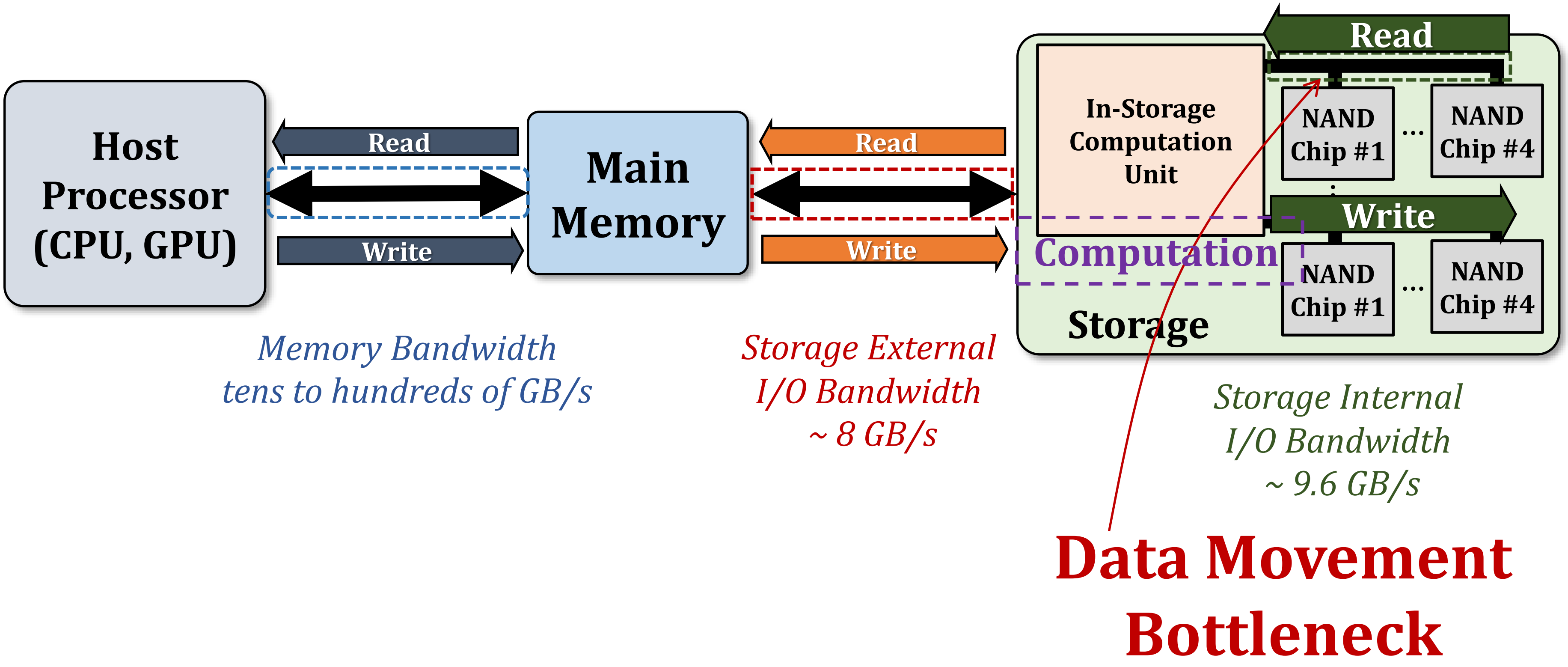
[3] Li+, “Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-Volatile Memories,” DAC, 2016

[4] Gu+, “Biscuit: A Framework for Near-Data Processing of Big Data Workloads,” ISCA, 2016

[5] Gao+, “ParaBit: Processing Parallel Bitwise Operations in NAND Flash Memory Based SSDs,” MICRO, 2021

In-Storage Processing (ISP)

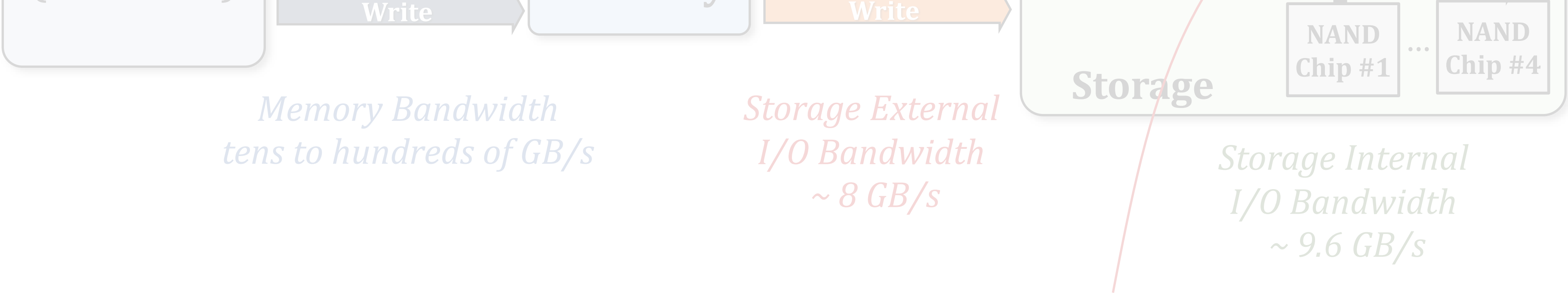
- ISP performs computation using an in-storage computation unit
- ISP reduces external data movement by transferring only the computation results to the host



In-Storage Processing (ISP)

- ISP performs computation using the in-storage computation unit
- ISP reduces external data movement by transferring only the computation results to the host

Storage internal I/O bandwidth is the main bottleneck for data movement in ISP



Memory Bandwidth
tens to hundreds of GB/s

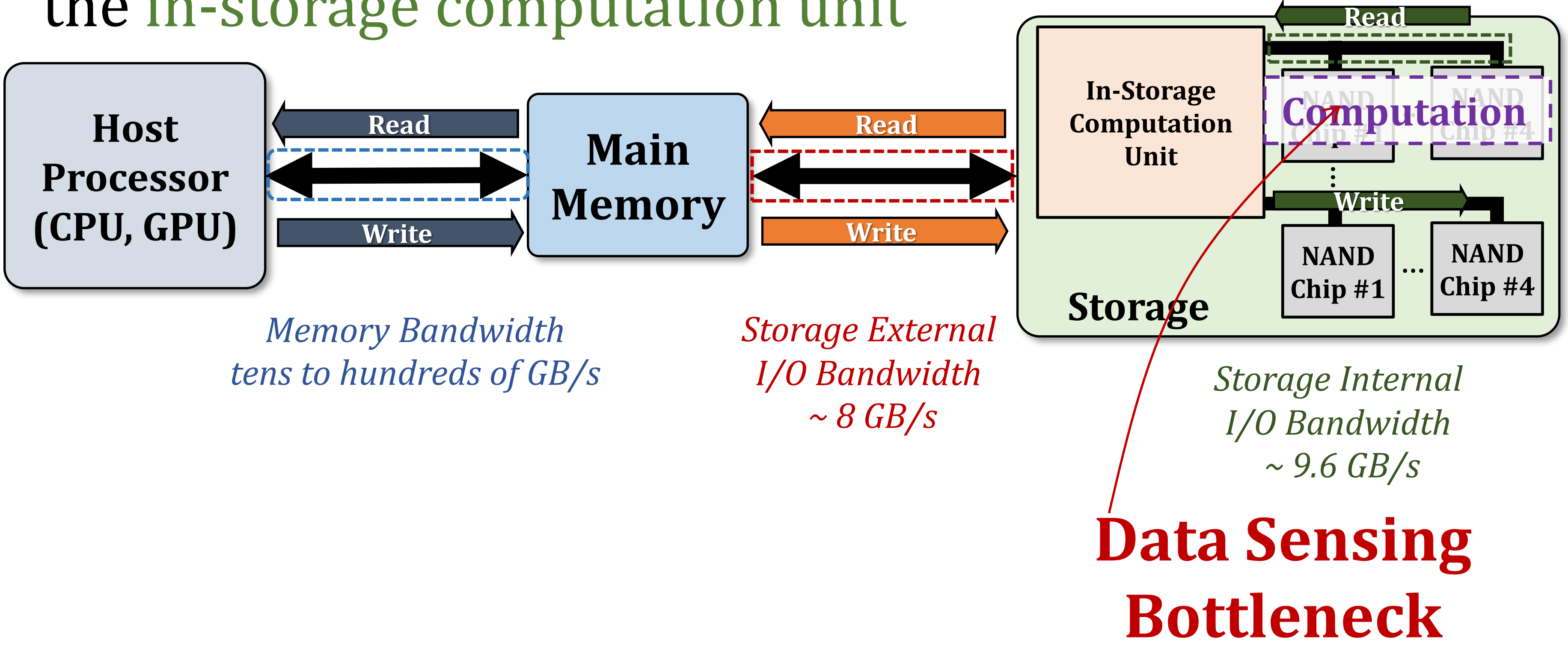
Storage External
I/O Bandwidth
~ 8 GB/s

Storage Internal
I/O Bandwidth
~ 9.6 GB/s

Data Movement Bottleneck

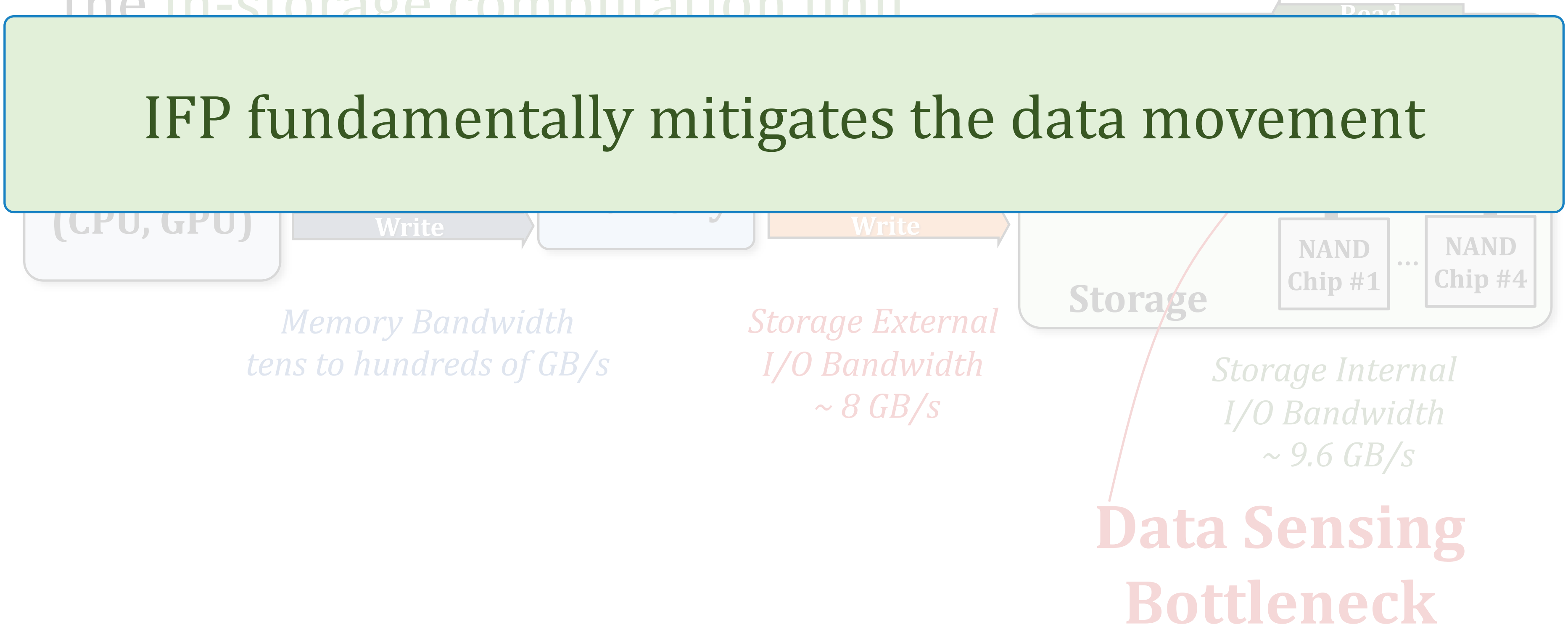
In-Flash Processing (IFP)

- IFP performs computation within the flash chips as the data operands are being read serially
- IFP reduces the internal data movement bottleneck in storage by transferring only the computation results to the in-storage computation unit


Host Processor (CPU, GPU)
Read
Write
Main Memory
Read
Write
In-Storage Computation Unit
Read
NAND Chip #1
NAND Chip #4
Computation
Write
NAND Chip #1
…
NAND Chip #4
Storage
Memory Bandwidth tens to hundreds of GB/s
Storage External I/O Bandwidth ~ 8 GB/s
Storage Internal I/O Bandwidth ~ 9.6 GB/s
Data Sensing Bottleneck

In-Flash Processing (IFP)

- IFP performs computation within the flash chips as the data operands are being read serially
- IFP reduces the internal data movement bottleneck in storage by transferring only the computation results to the in-storage computation unit



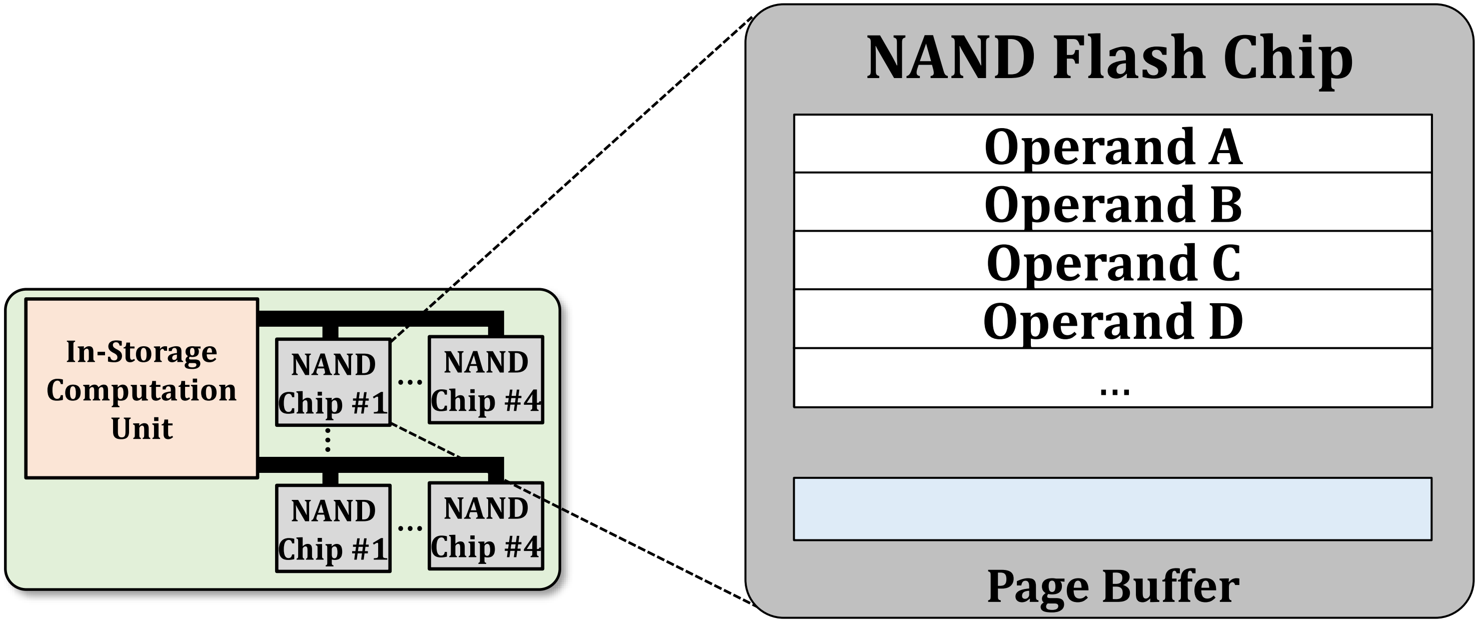
Read
(CPU, GPU)
Write
Write
NAND Chip #1
…
NAND Chip #4
Storage
Memory Bandwidth tens to hundreds of GB/s
Storage External I/O Bandwidth ~ 8 GB/s
Storage Internal I/O Bandwidth ~ 9.6 GB/s
Data Sensing Bottleneck

IFP fundamentally mitigates the data movement

SAFARI

Data Sensing Bottleneck in IFP

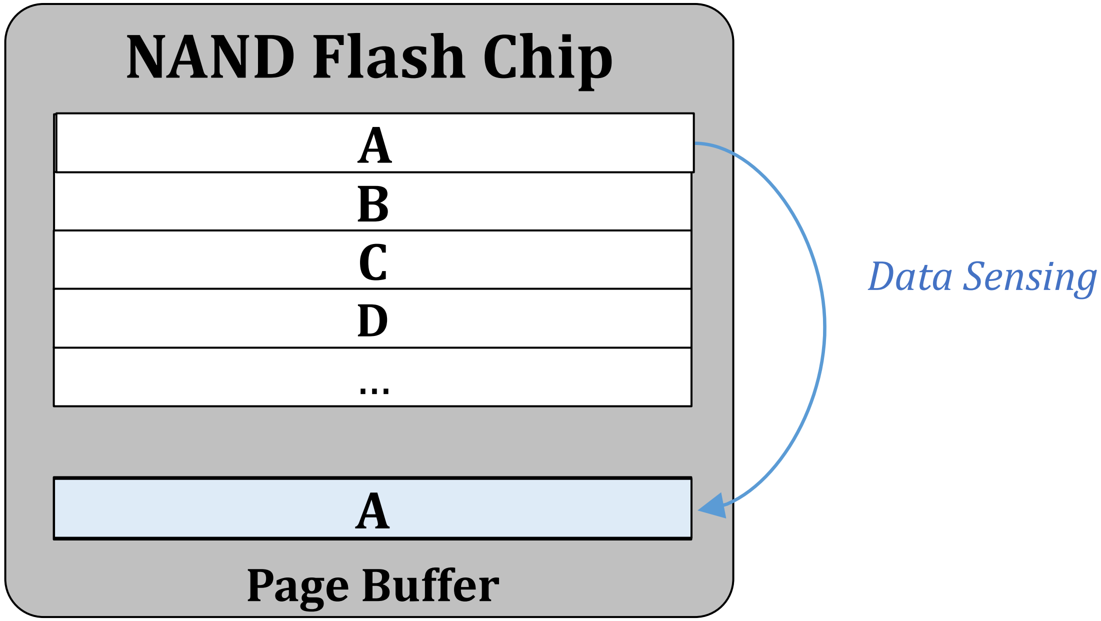
- State-of-the-art IFP technique [1] performs bulk bitwise operations by controlling the latching circuit of the page buffer



[1] Gao+, "ParaBit: Processing Parallel Bitwise Operations in NAND Flash Memory Based SSDs," MICRO, 2021

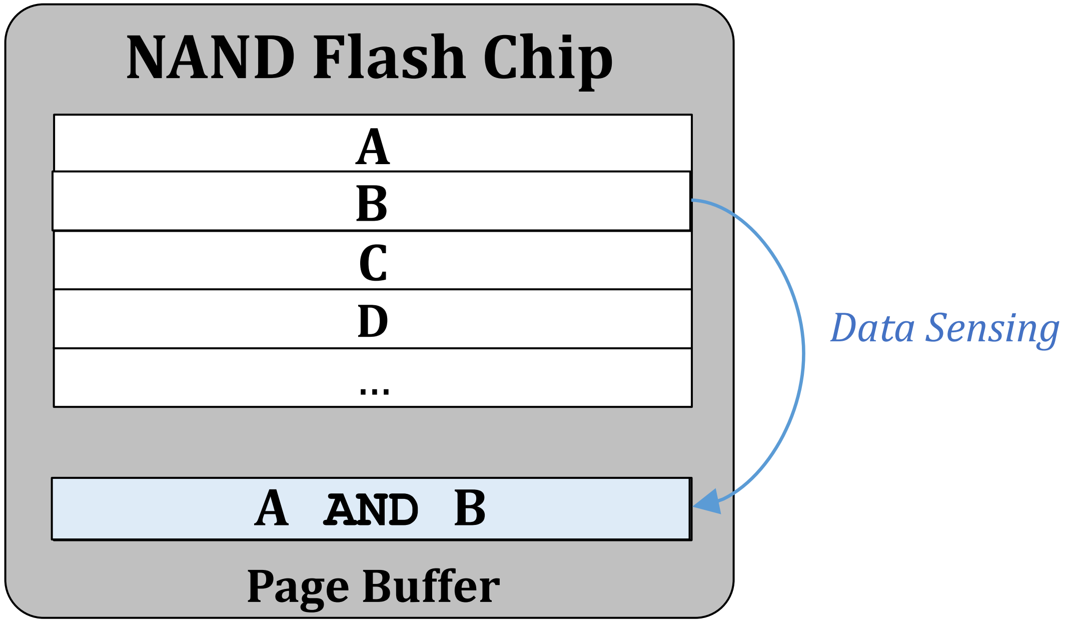
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique [1] performs bulk bitwise operations by controlling the latching circuit of the page buffer



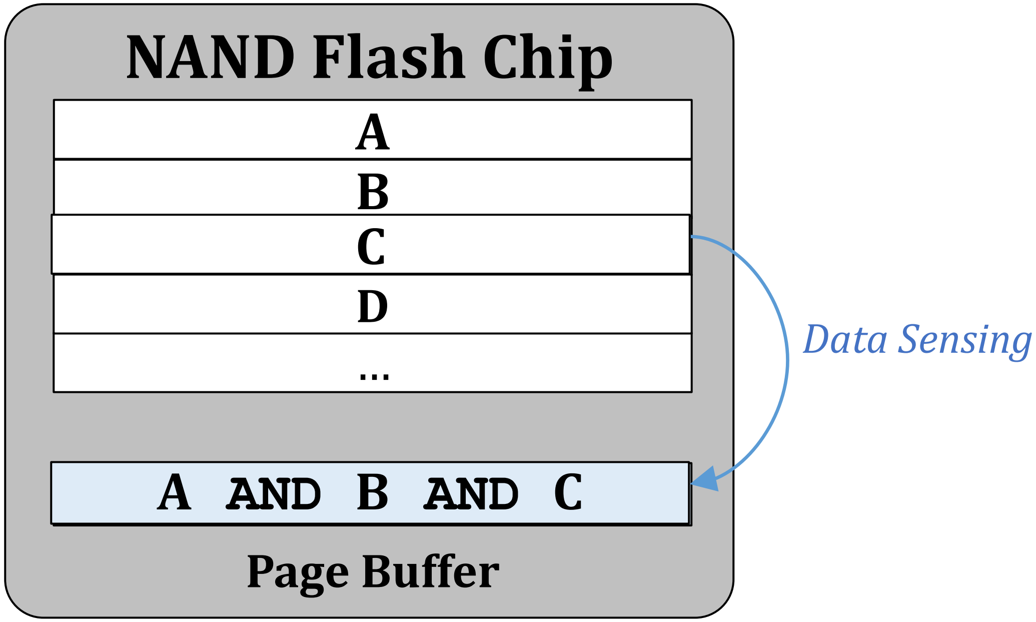
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique [1] performs bulk bitwise operations by controlling the latching circuit of the page buffer



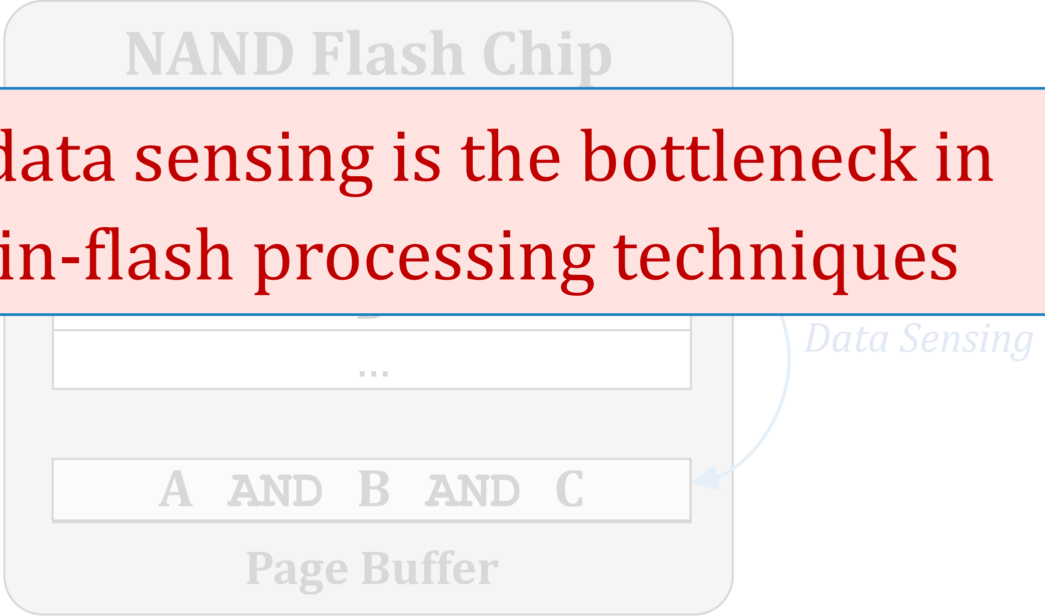
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique [1] performs bulk bitwise operations by controlling the latching circuit of the page buffer



Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique [1] performs bulk bitwise operations by controlling the latching circuit of the page buffer

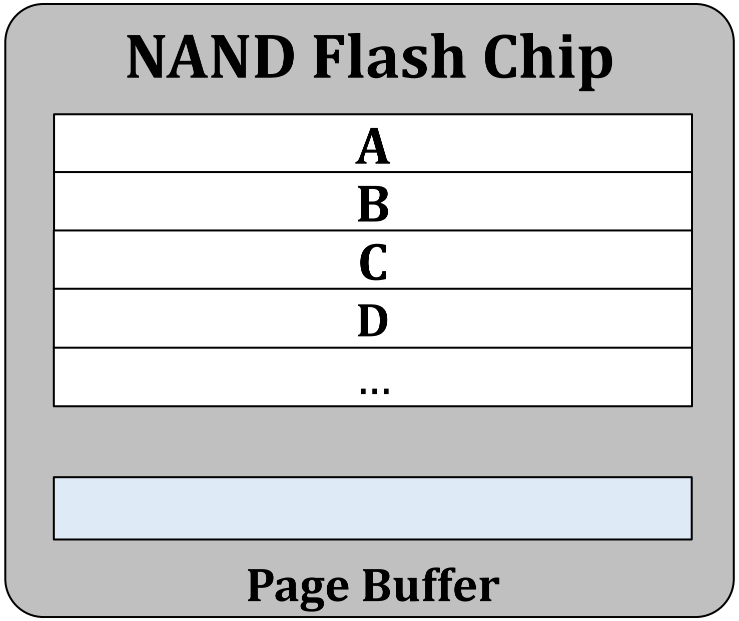


Serial data sensing is the bottleneck in prior in-flash processing techniques

SAFARI

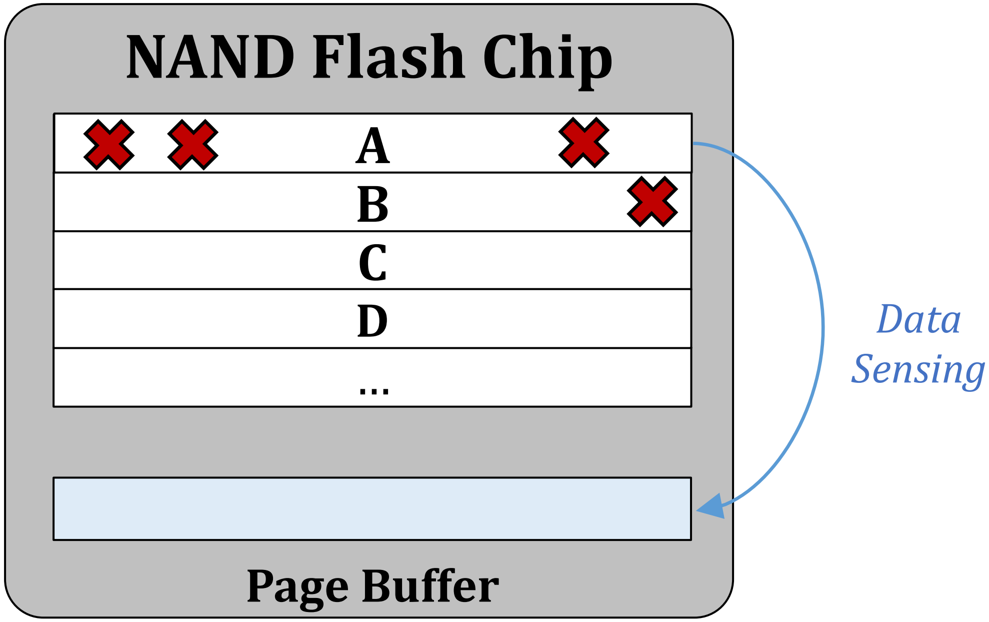
Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing



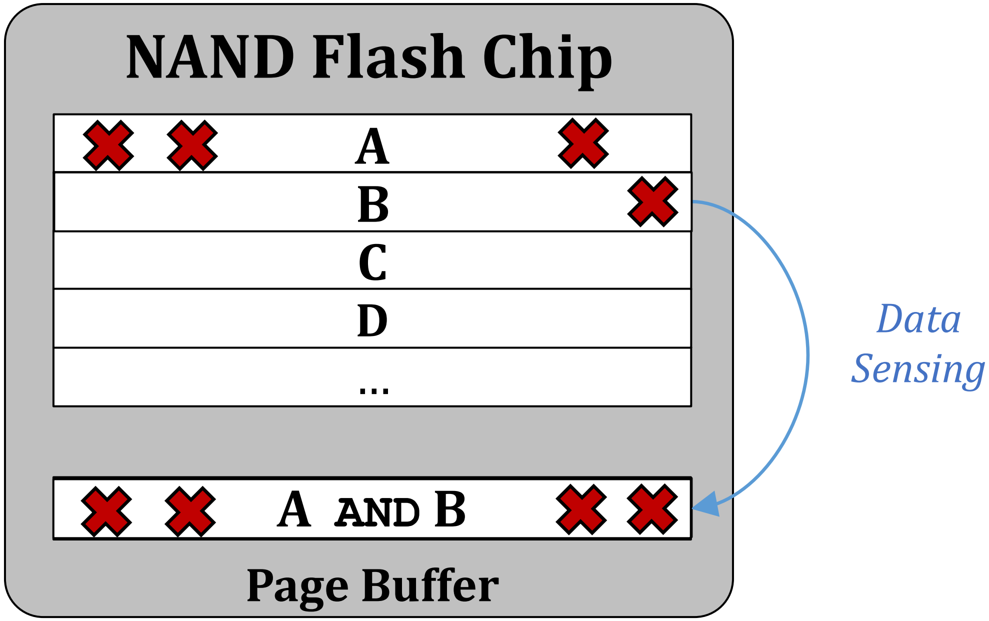
Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing



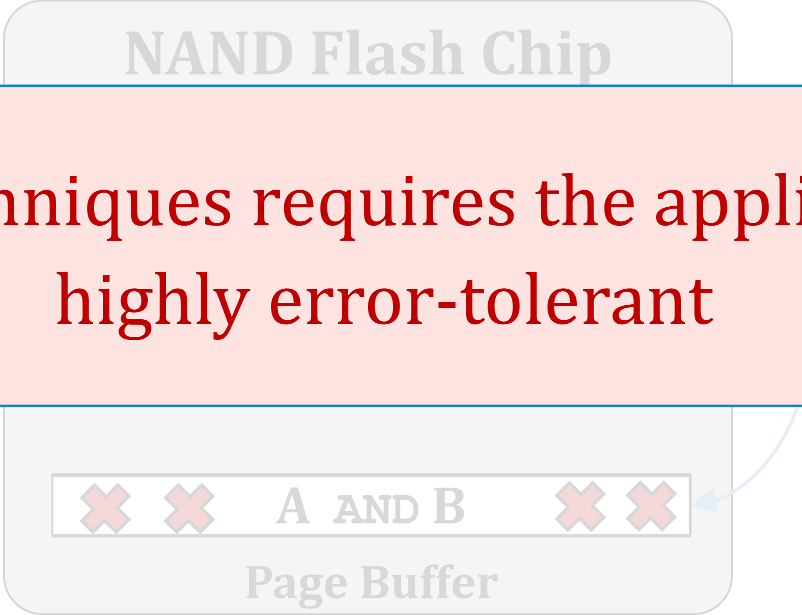
Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing



Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing



Prior IFP techniques requires the application to be highly error-tolerant

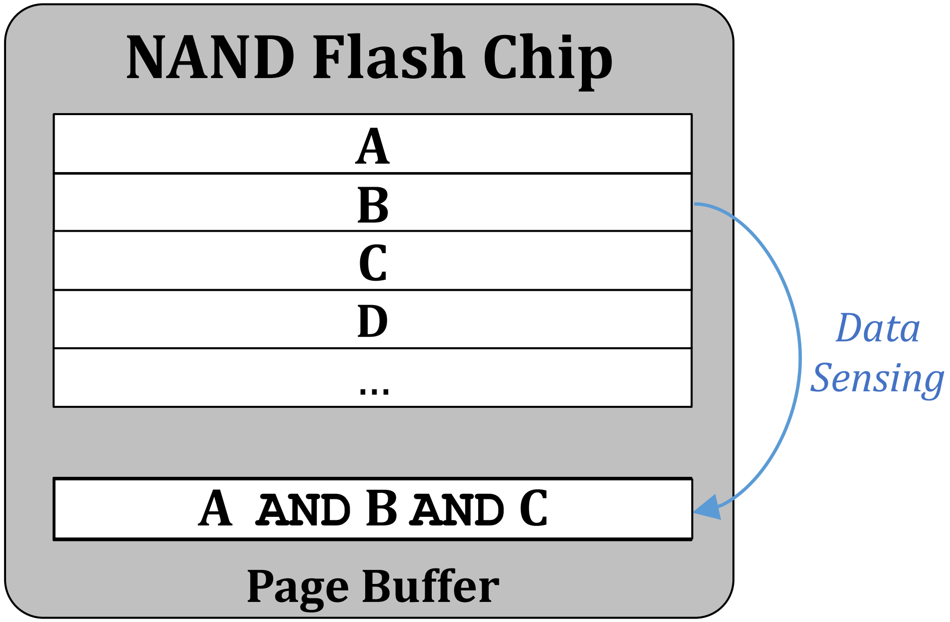
Our Goal

Address the bottleneck of state-of-the-art IFP techniques (serial sensing of operands)

Make IFP reliable (provide accurate computation results)

Our Proposal

- Flash-Cosmos enables
  - Computation on multiple operands using a single sensing operation
  - Provide high reliability during in-flash computation



Talk Outline

Motivation

Background

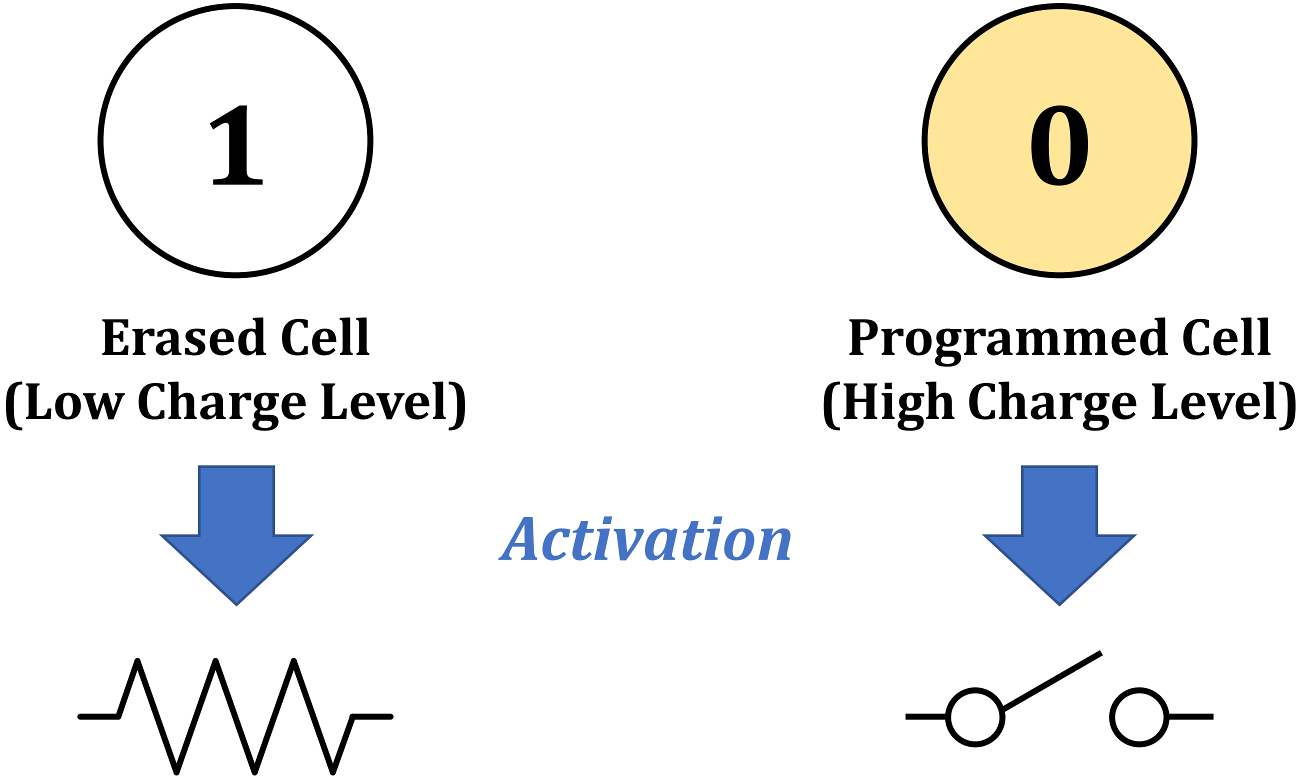
Flash-Cosmos

Evaluation

Summary

NAND Flash Basics: A Flash Cell

- **A flash cell stores data by adjusting the amount of charge in the cell**



1

Erased Cell
(Low Charge Level)

0

Programmed Cell
(High Charge Level)

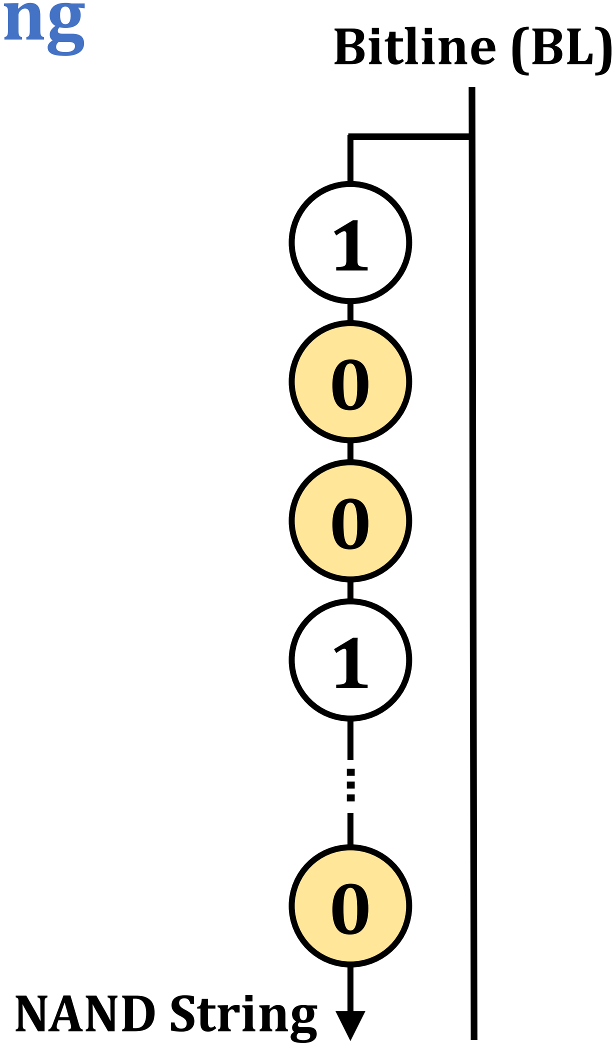
Activation

Operates as a resistor

Operates as an open switch

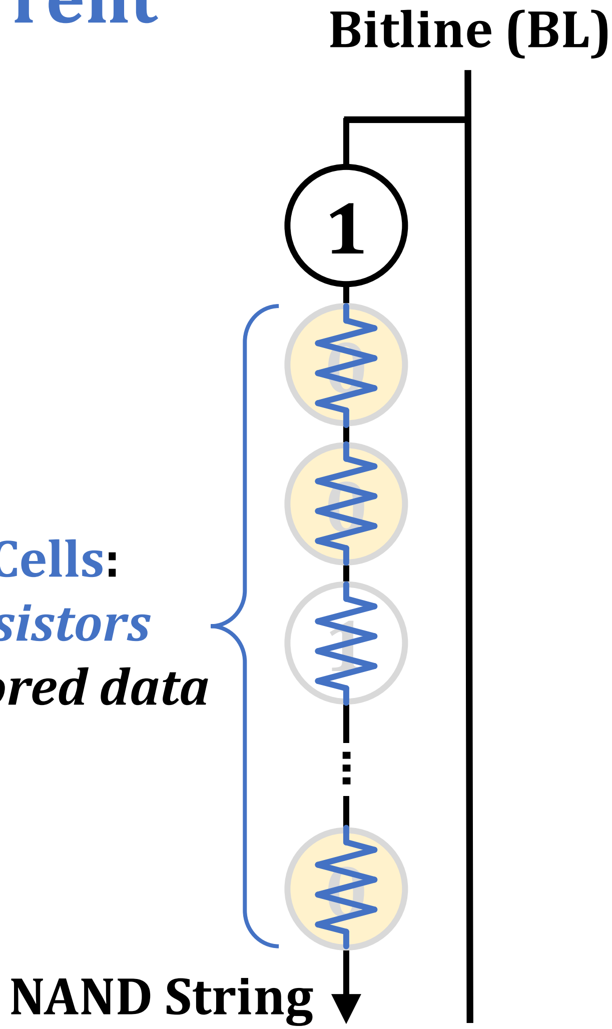
NAND Flash Basics: A NAND String

- A set of flash cells are **serially connected** to form a **NAND String**



NAND Flash Basics: Read Mechanism

- **NAND flash memory reads data by checking the bitline current**



Bitline (BL)

1

Non-Target Cells:
Operate as resistors regardless of stored data

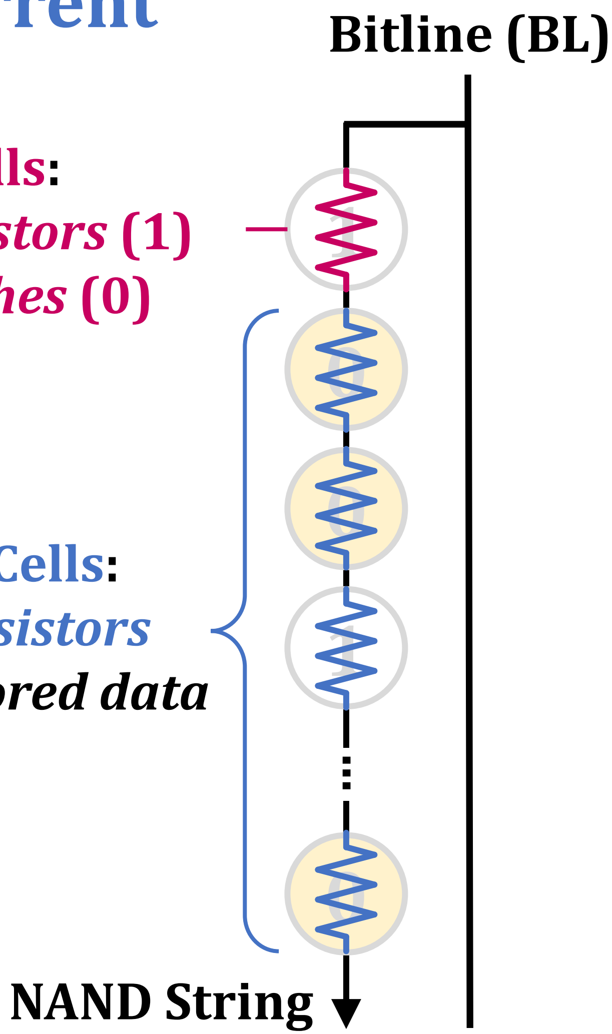
NAND String

NAND Flash Basics: Read Mechanism

- **NAND flash memory reads data by checking the bitline current**

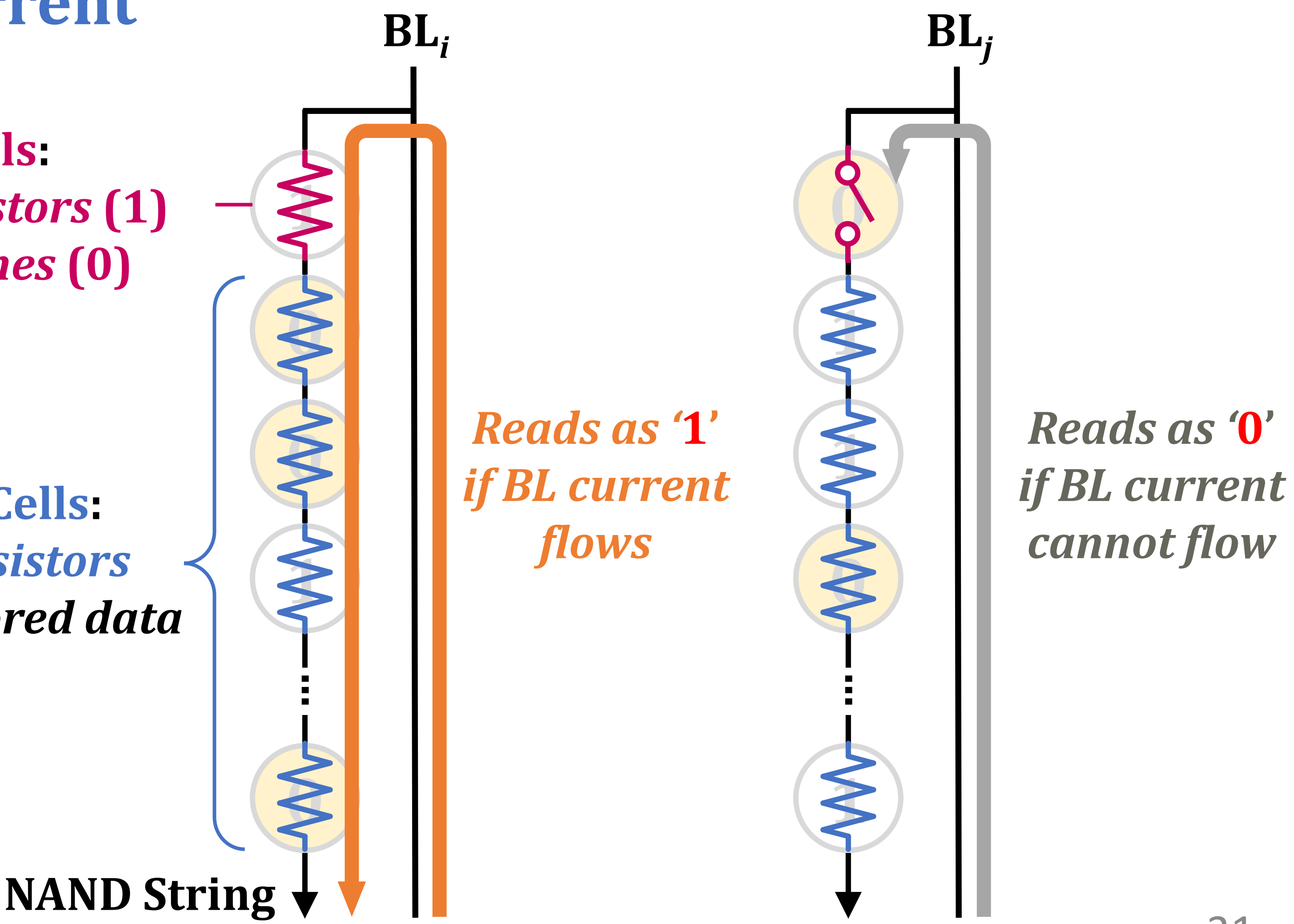
Target Cells:
Operate as resistors (1)
or open switches (0)

Non-Target Cells:
Operate as resistors
regardless of stored data



NAND Flash Basics: Read Mechanism

- NAND flash memory reads data by **checking the bitline current**



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

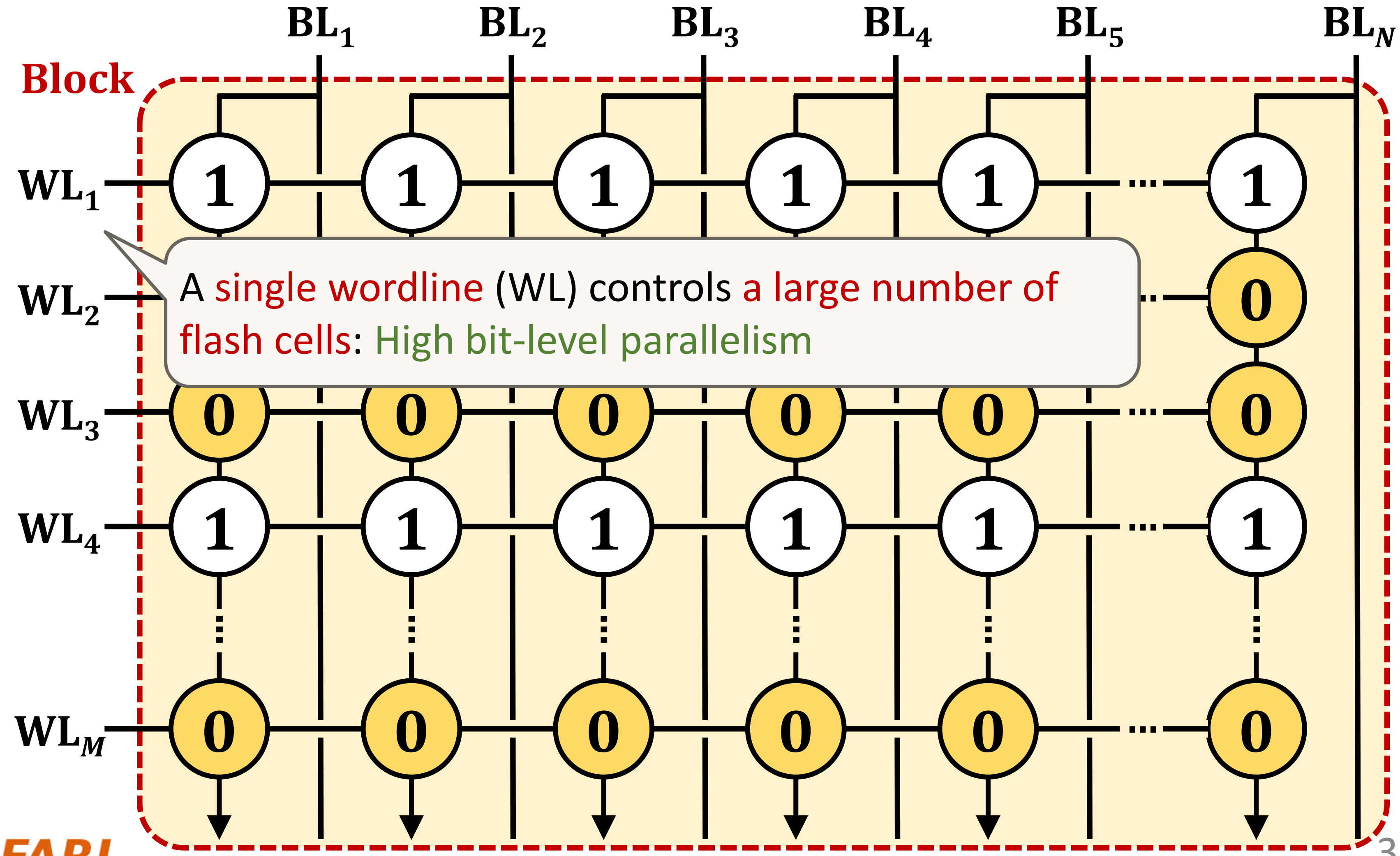
Non-Target Cells:
Operate as resistors
regardless of stored data

Reads as '1'
if BL current
flows

Reads as '0'
if BL current
cannot flow

NAND Flash Basics: A NAND Flash Block

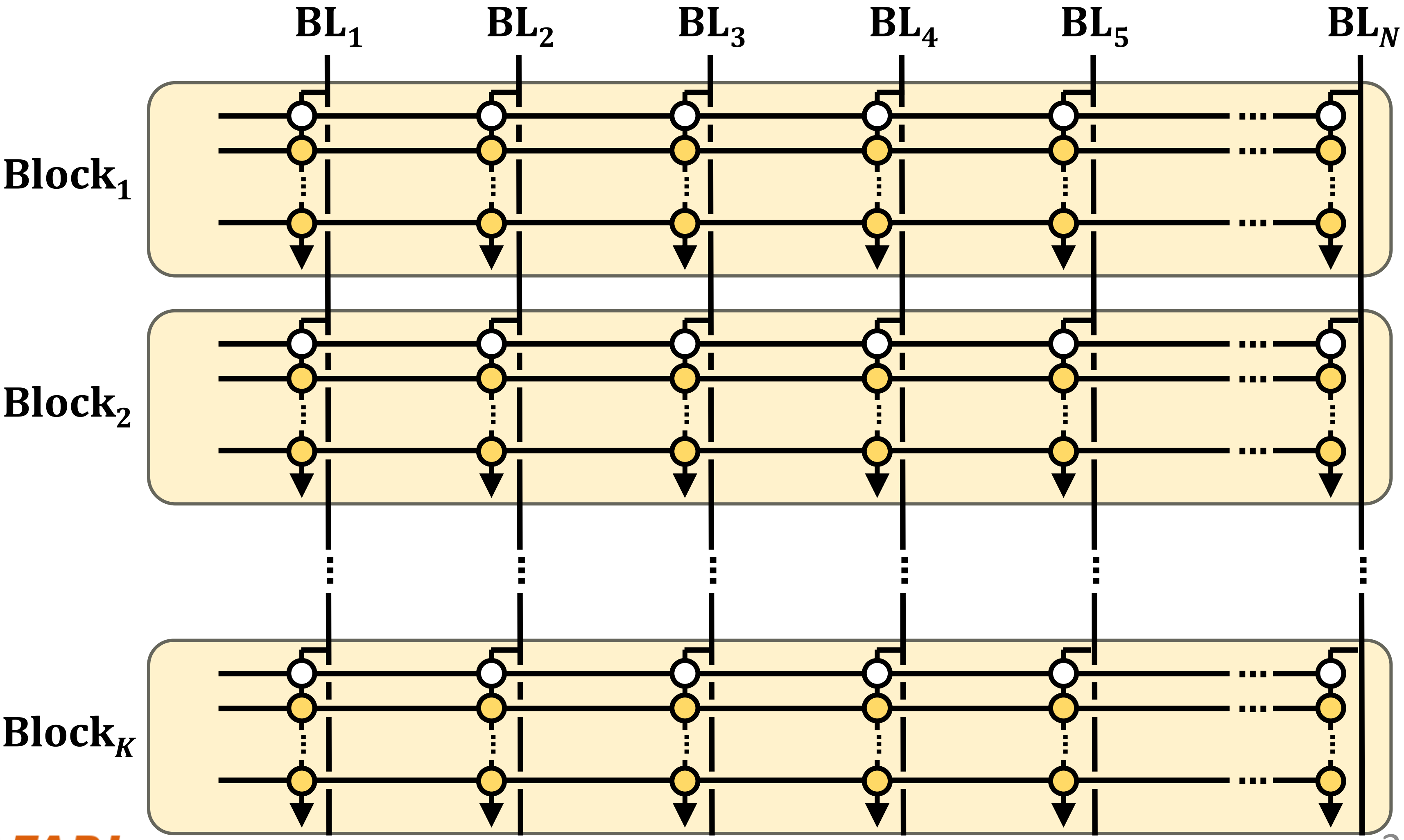
- NAND strings connected to different bitlines comprise a NAND block



A single wordline (WL) controls a large number of flash cells: High bit-level parallelism

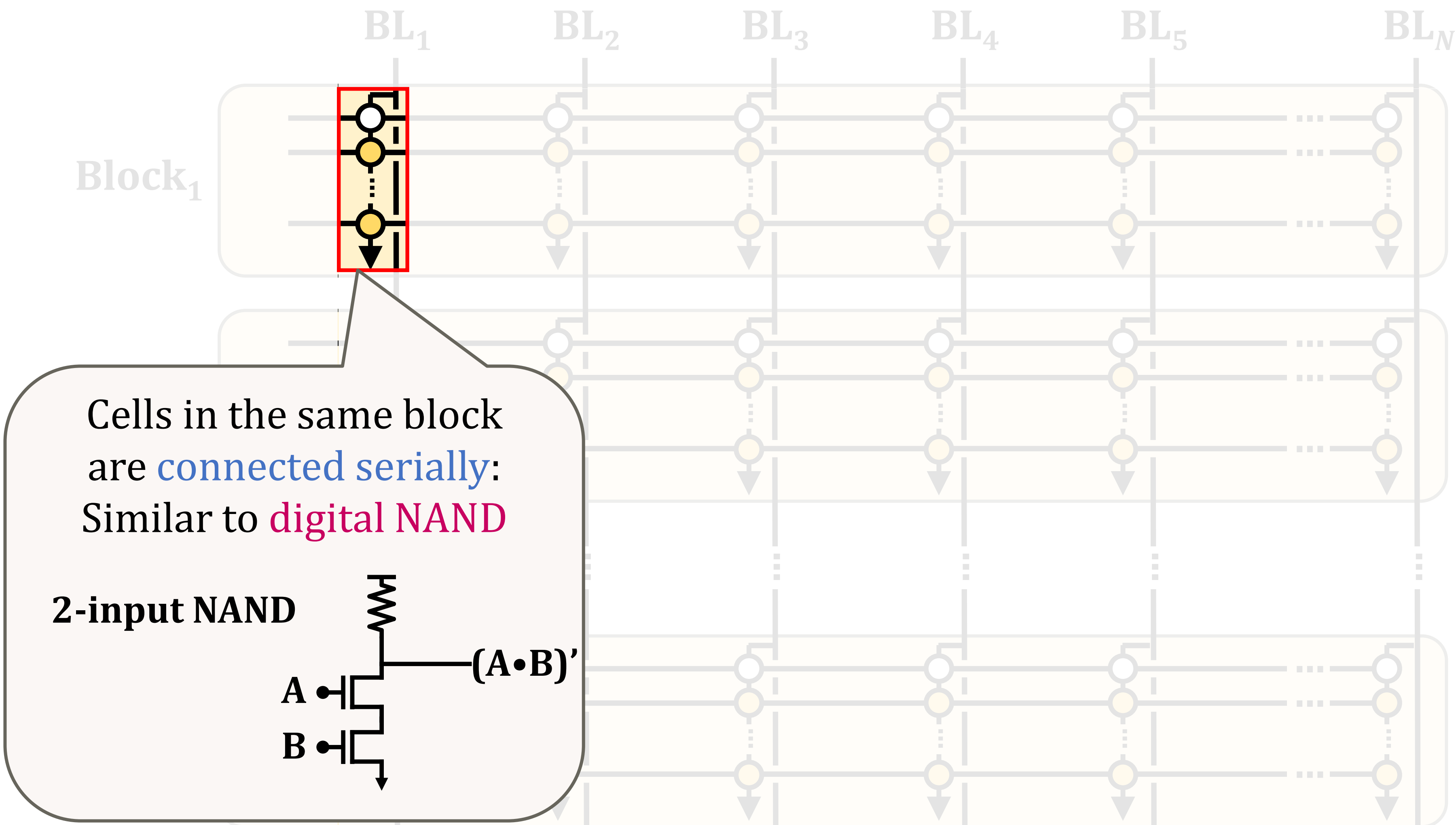
NAND Flash Basics: Block Organization

- **A large number of blocks share the same bitlines**



Similarity to Digital Logic Gates

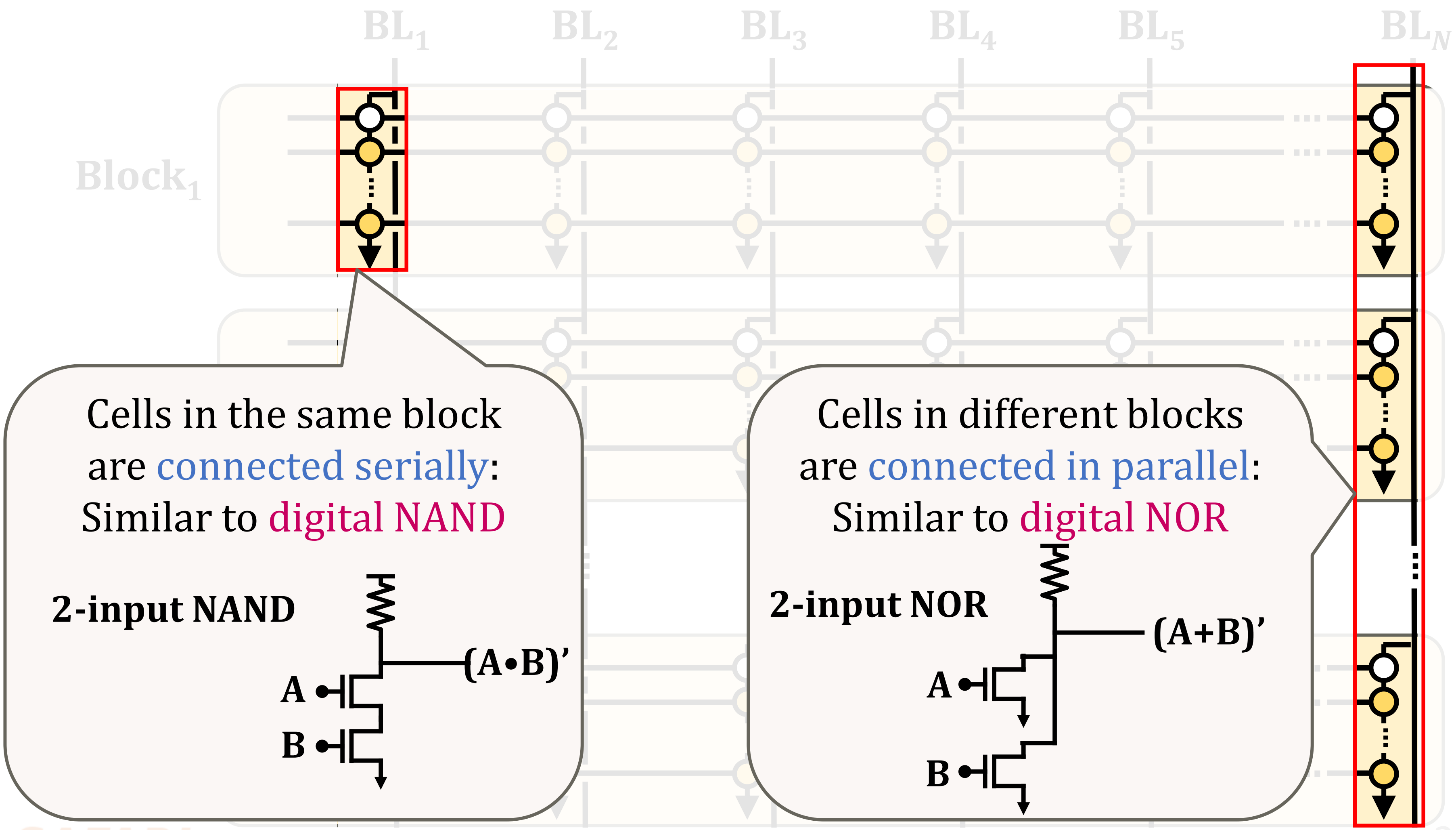
- A large number of blocks share the same bitlines



Cells in the same block are connected serially: Similar to digital NAND

Similarity to Digital Logic Gates

- A large number of blocks share the same bitlines.



Cells in the same block are connected serially: Similar to digital NAND

Cells in different blocks are connected in parallel: Similar to digital NOR

Talk Outline

Motivation

Background

Flash-Cosmos

Evaluation

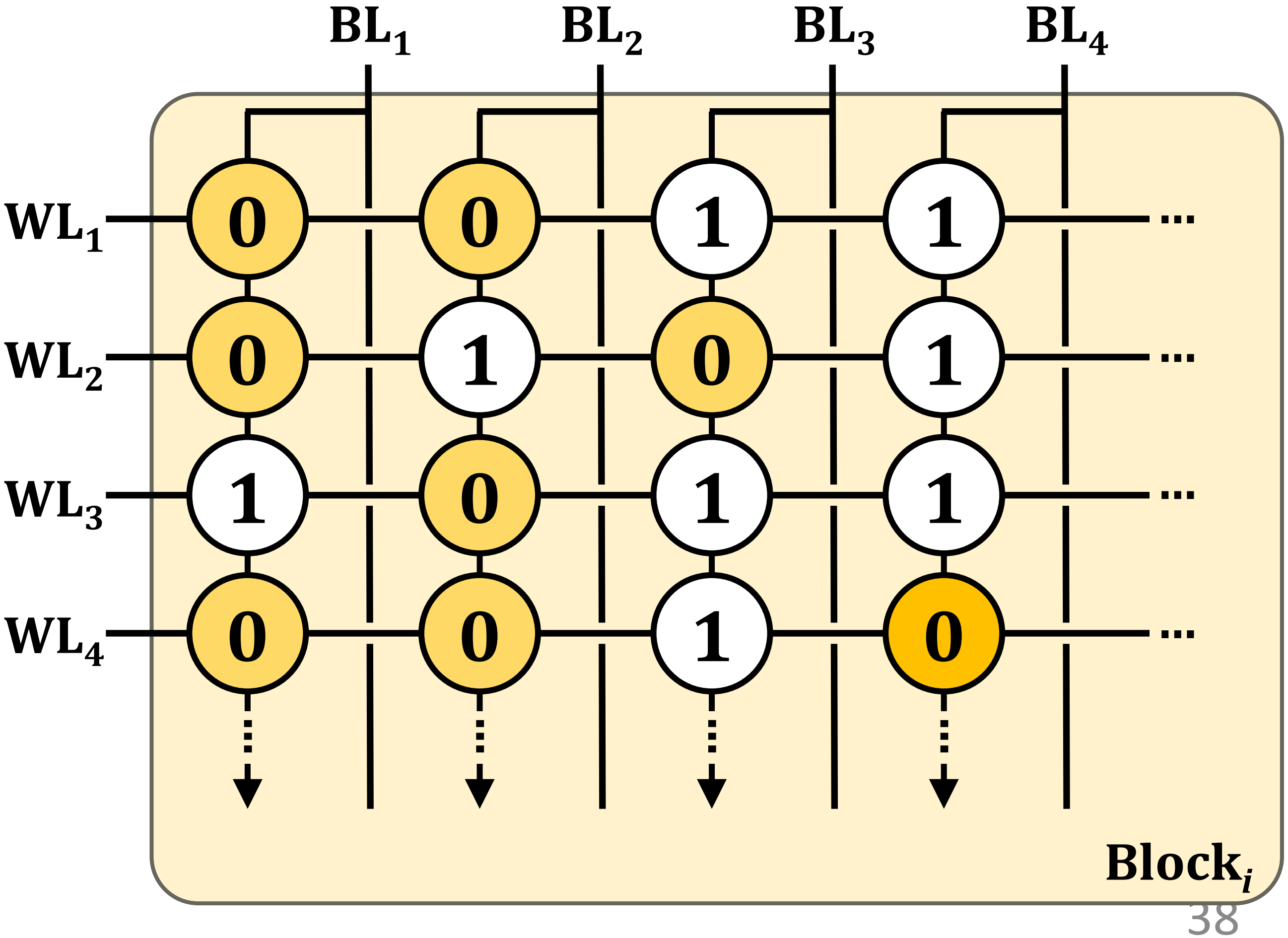
Summary

Flash-Cosmos: Overview

Enables in-flash bulk bitwise operations on multiple operands with a *single* sensing operation using
Multi-Wordline Sensing (MWS)

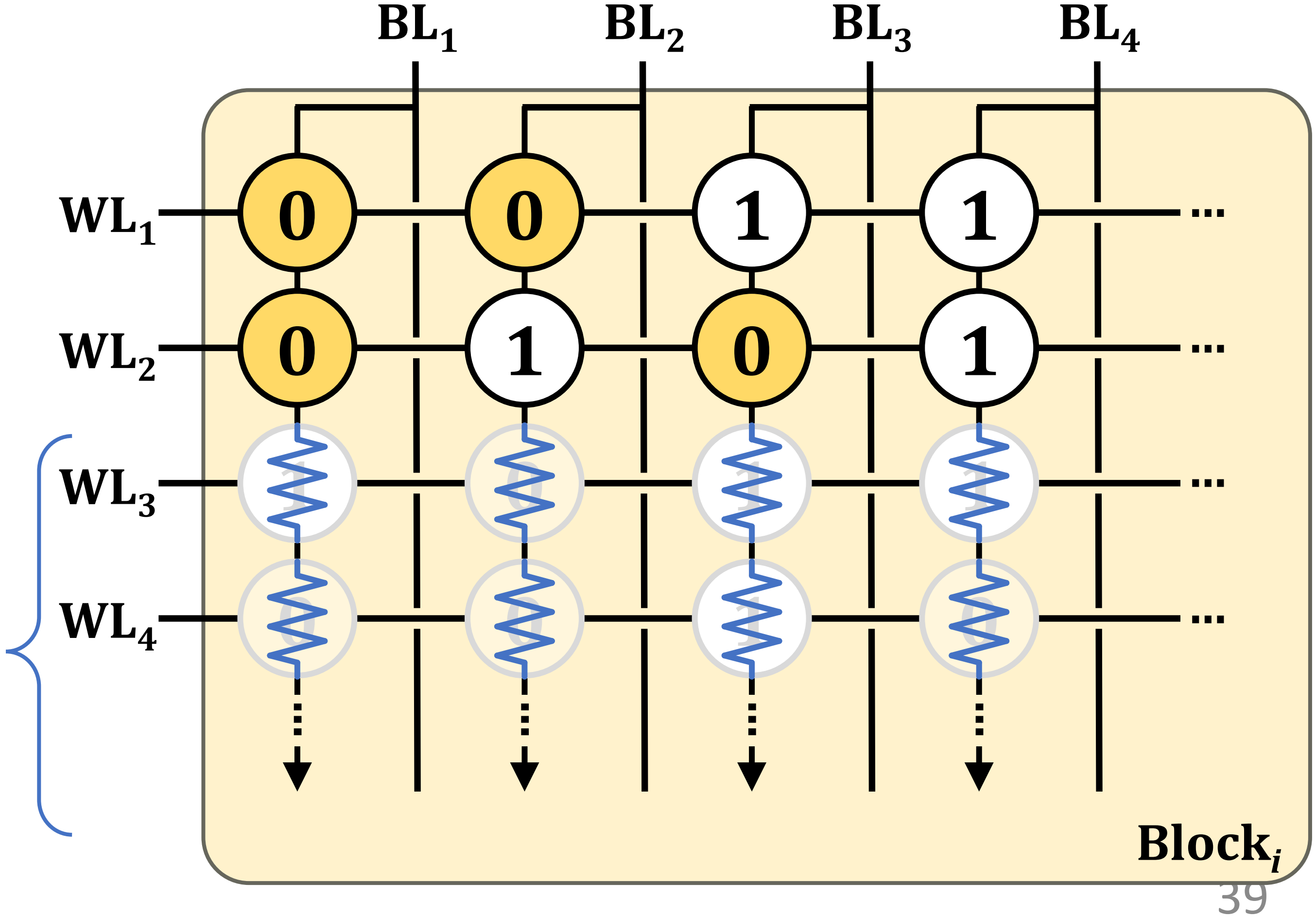
Multi-Wordline Sensing (MWS): Bitwise AND

- **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

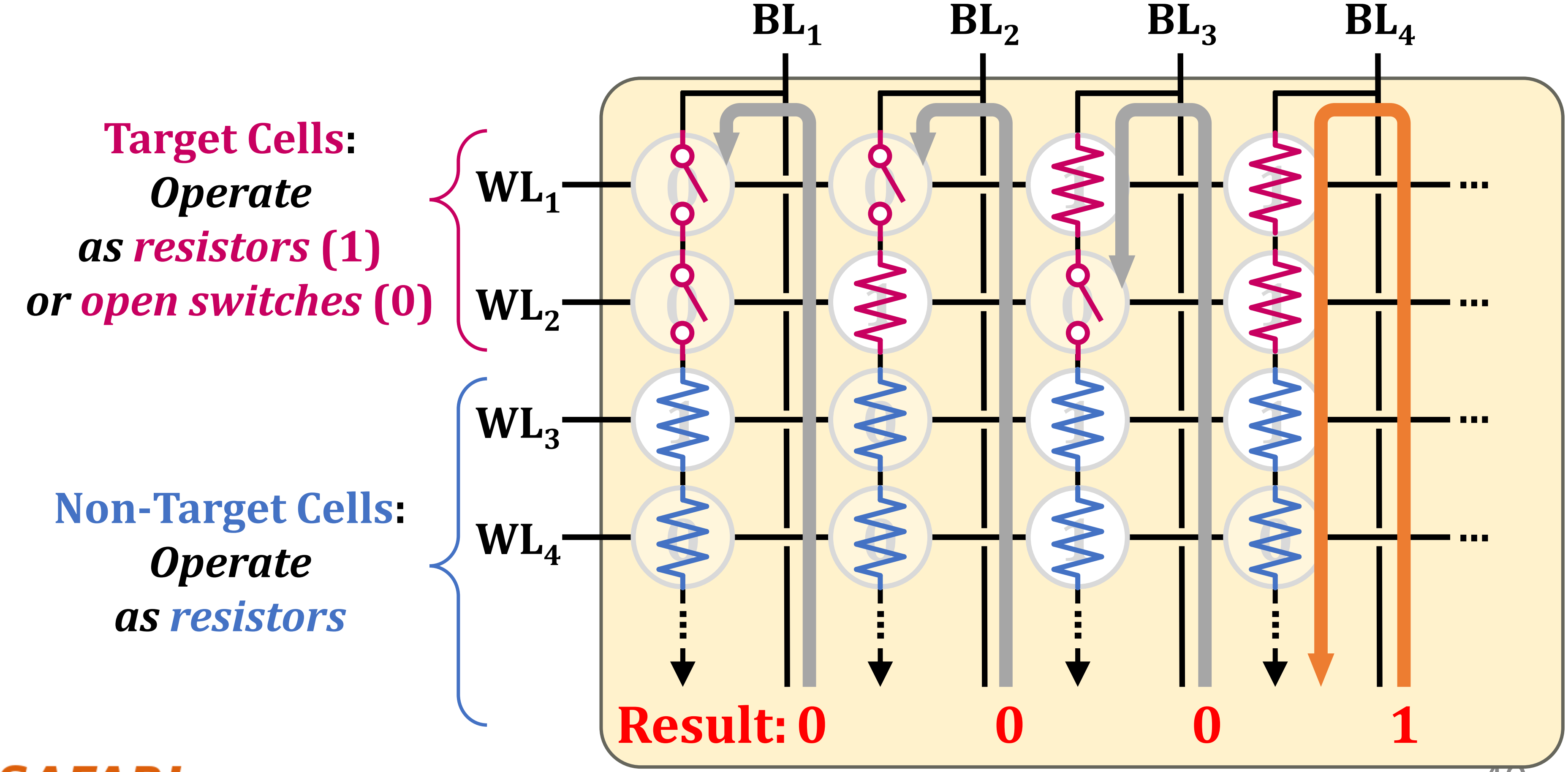
- **Intra-Block MWS**: **Simultaneously activates multiple WLs in the same block**
  - **Bitwise AND of the stored data in the WLs**



Non-Target Cells: ***Operate as resistors***

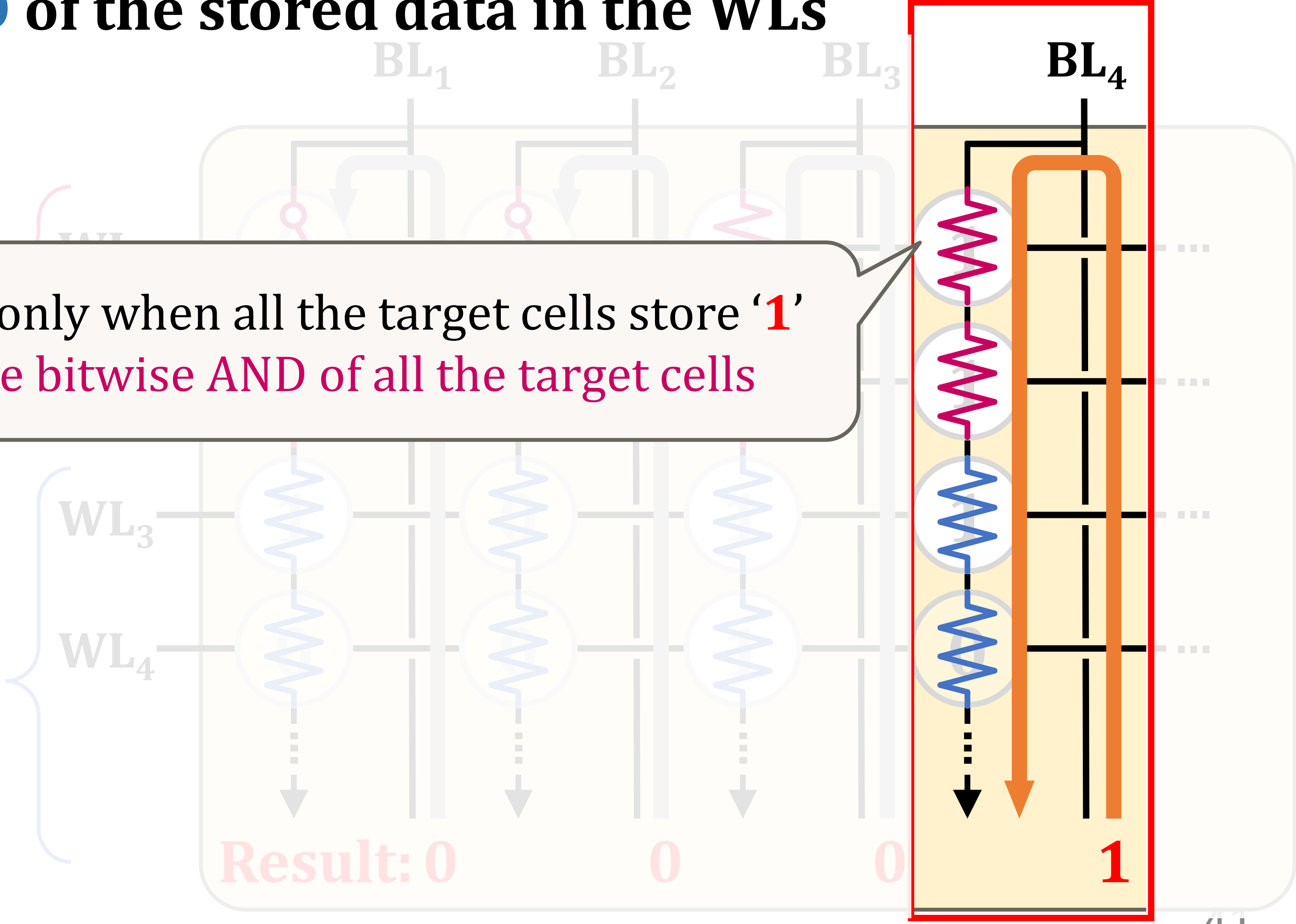
Multi-Wordline Sensing (MWS): Bitwise AND

- **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

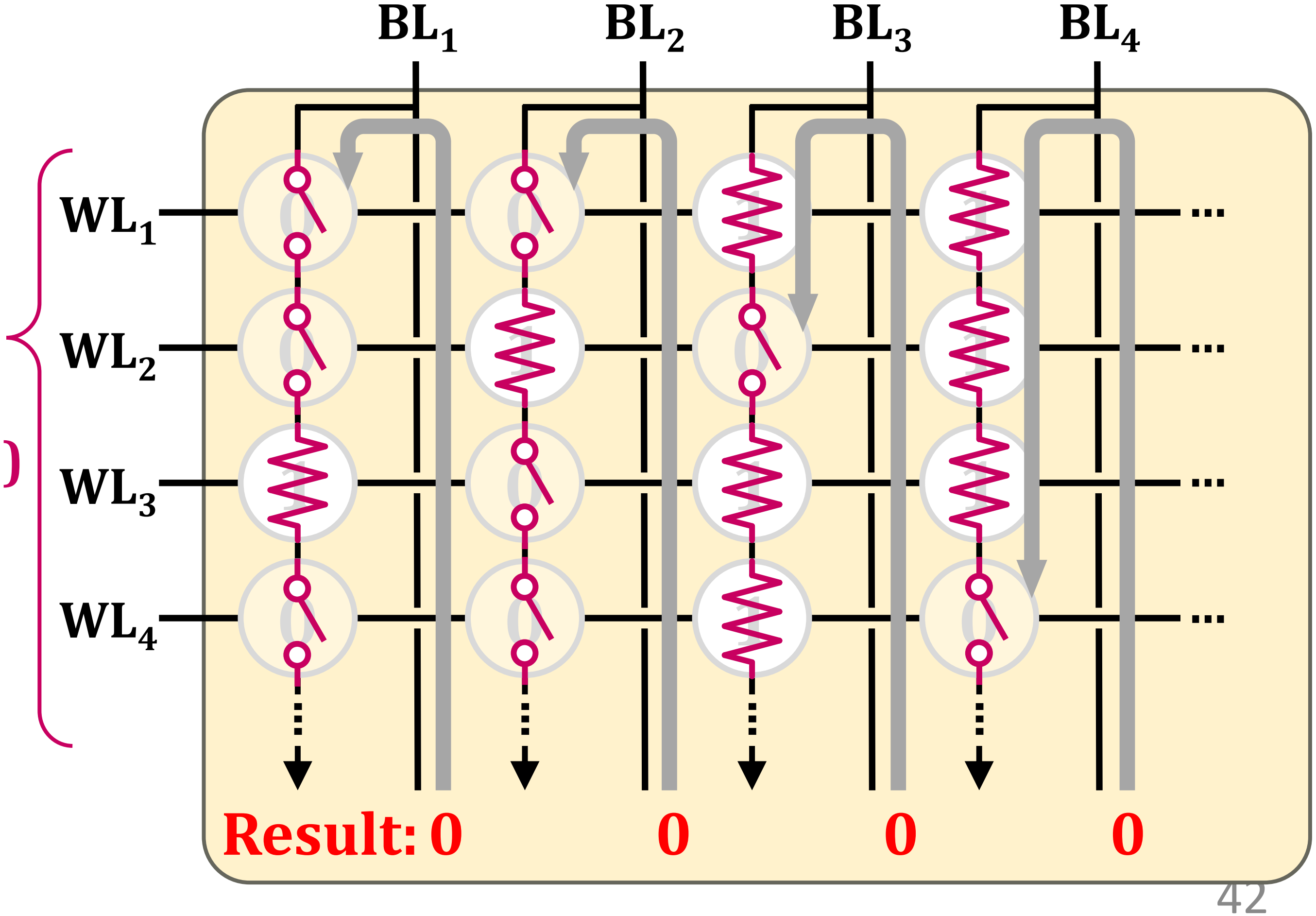
- **Intra-Block MWS: Simultaneously activates multiple WLs in the same block**
  - **Bitwise AND of the stored data in the WLs**



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**



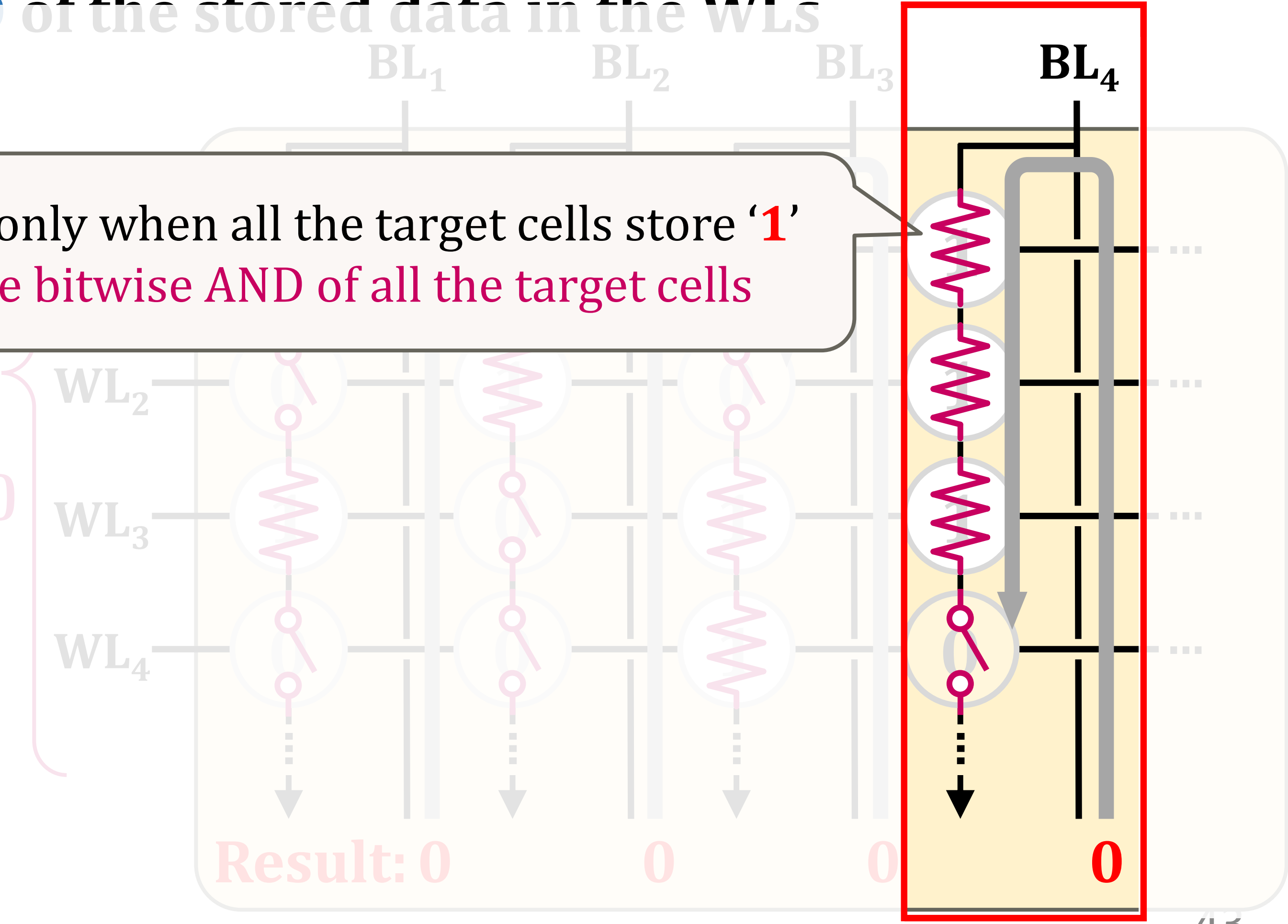
Target Cell:
Operate
as a resistance (1)
or an open switch (0)

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

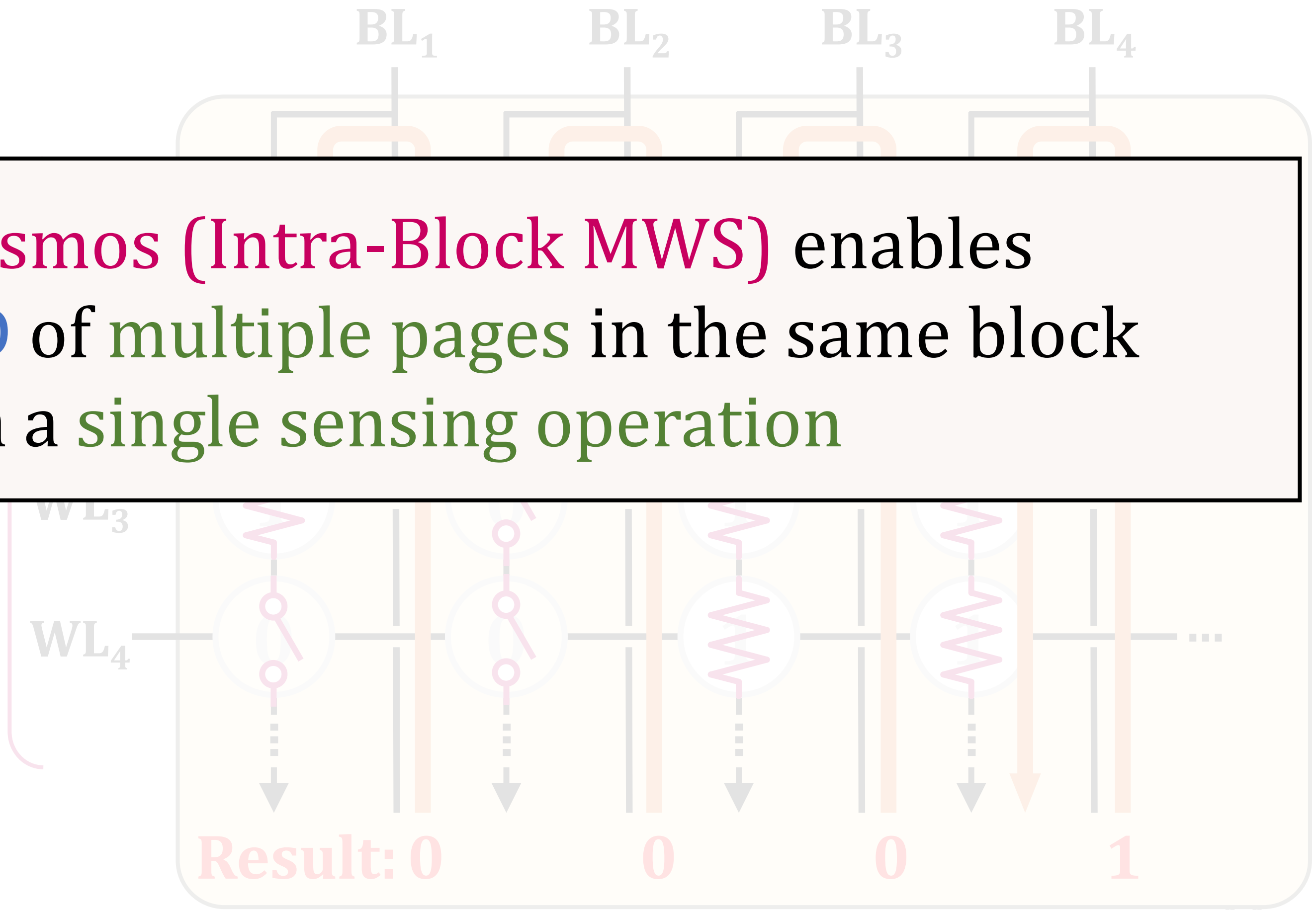
A bitline reads as ‘1’ only when all the target cells store ‘1’
→ Equivalent to the bitwise AND of all the target cells

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



Multi-Wordline Sensing (MWS): Bitwise AND

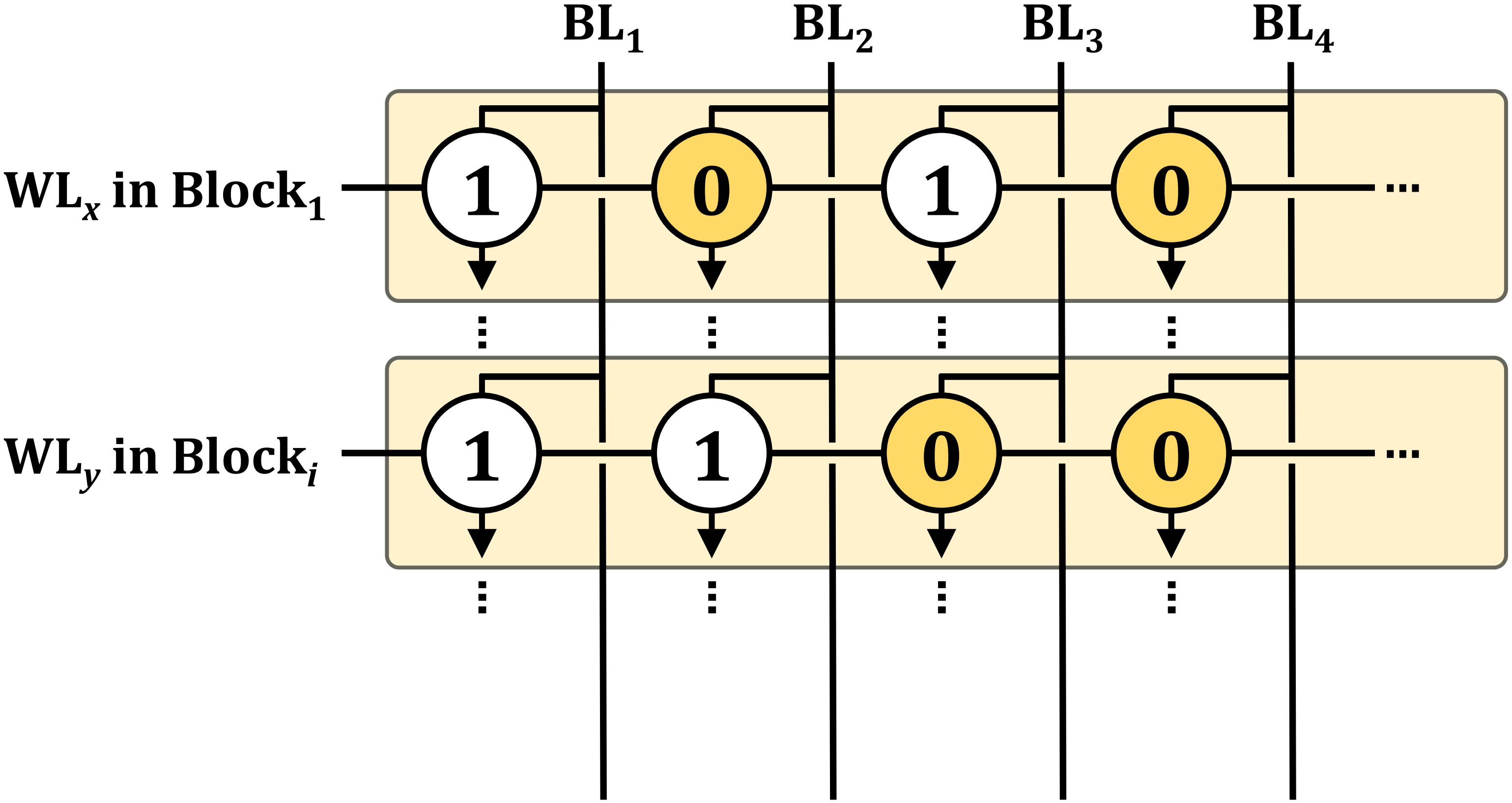
- Intra-Block MWS: Simultaneously activates multiple WLs in the same block
  - Bitwise AND of the stored data in the WLs



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

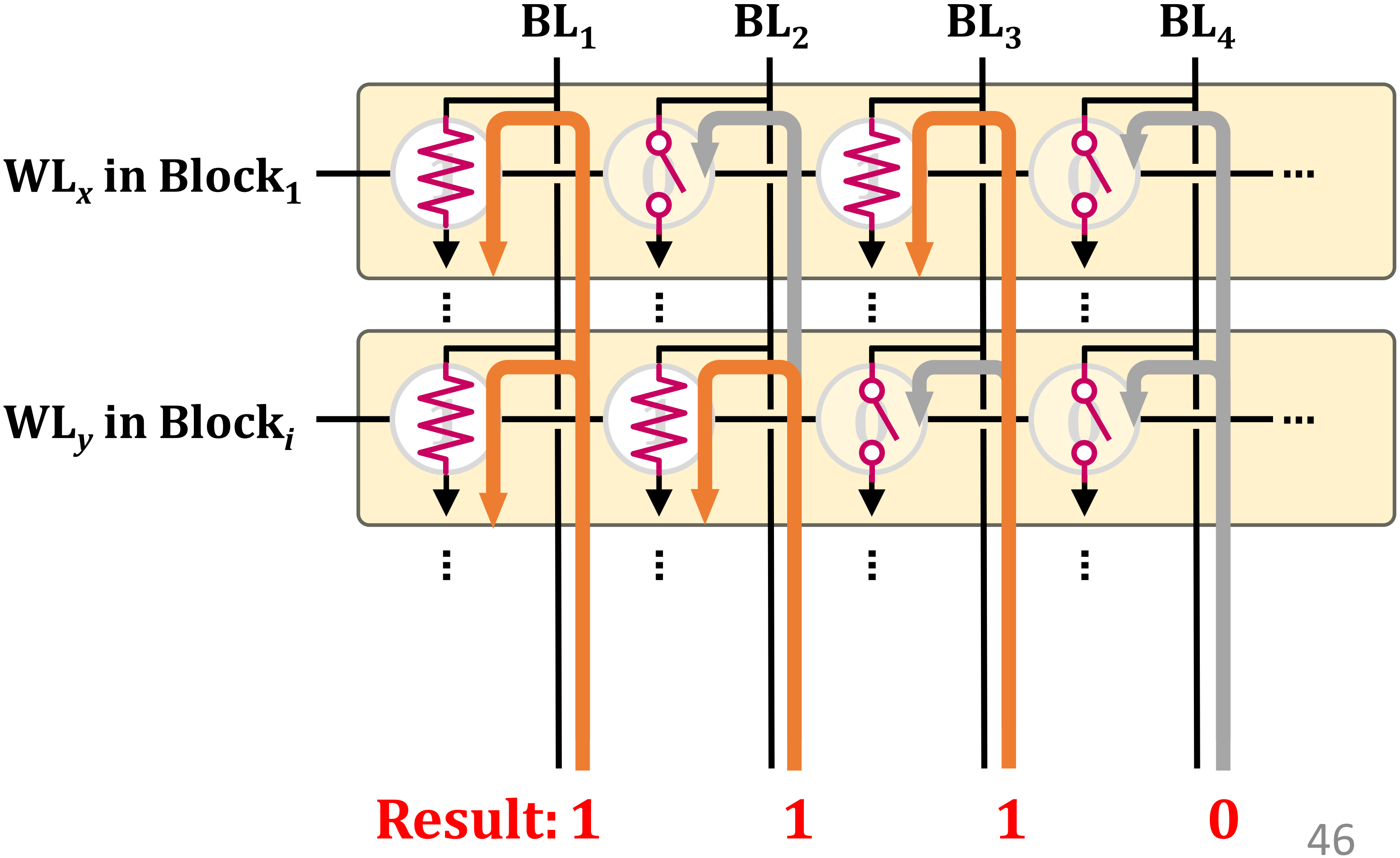
Multi-Wordline Sensing (MWS): Bitwise OR

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



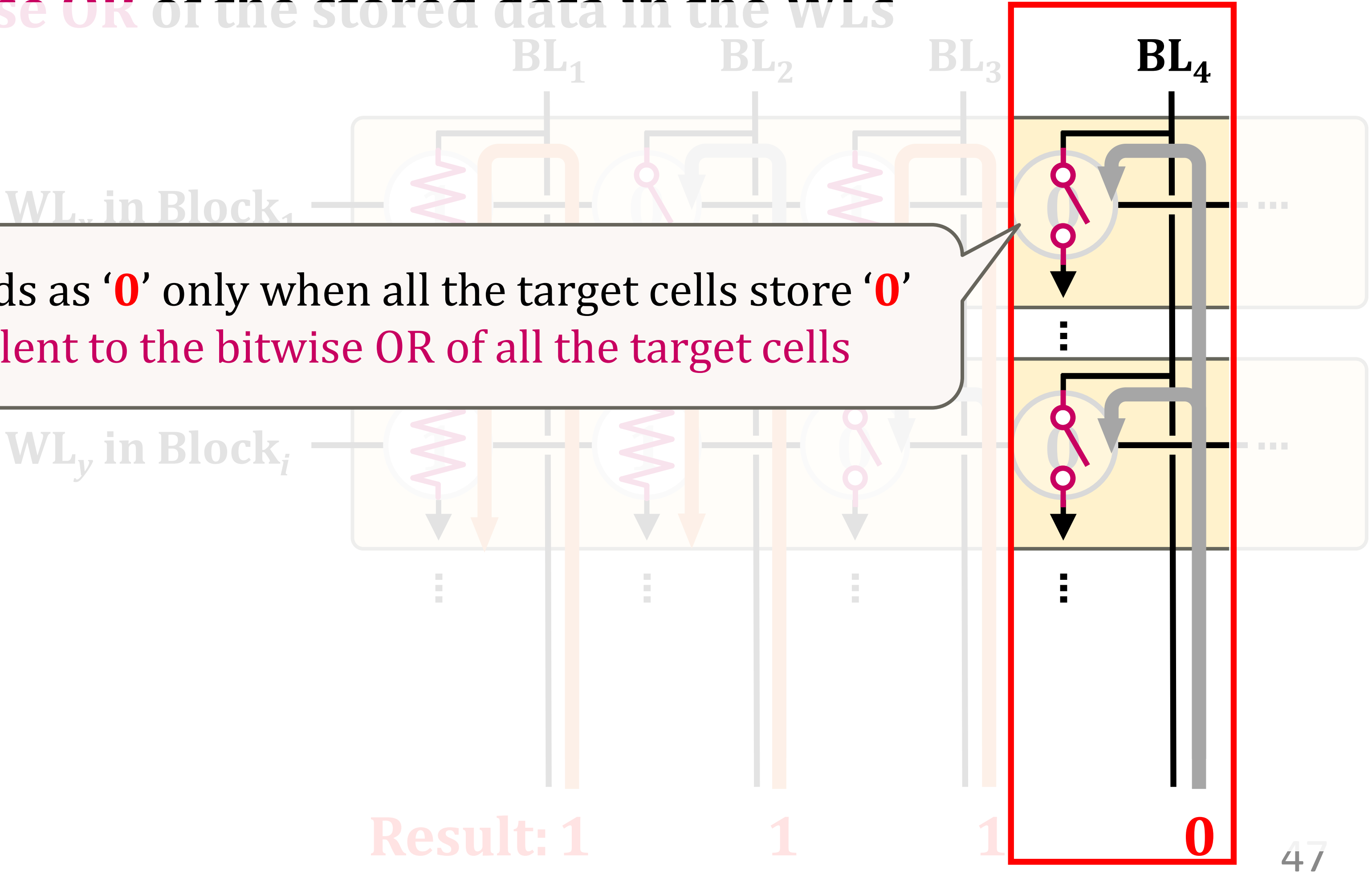
Multi-Wordline Sensing (MWS): Bitwise OR

- **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

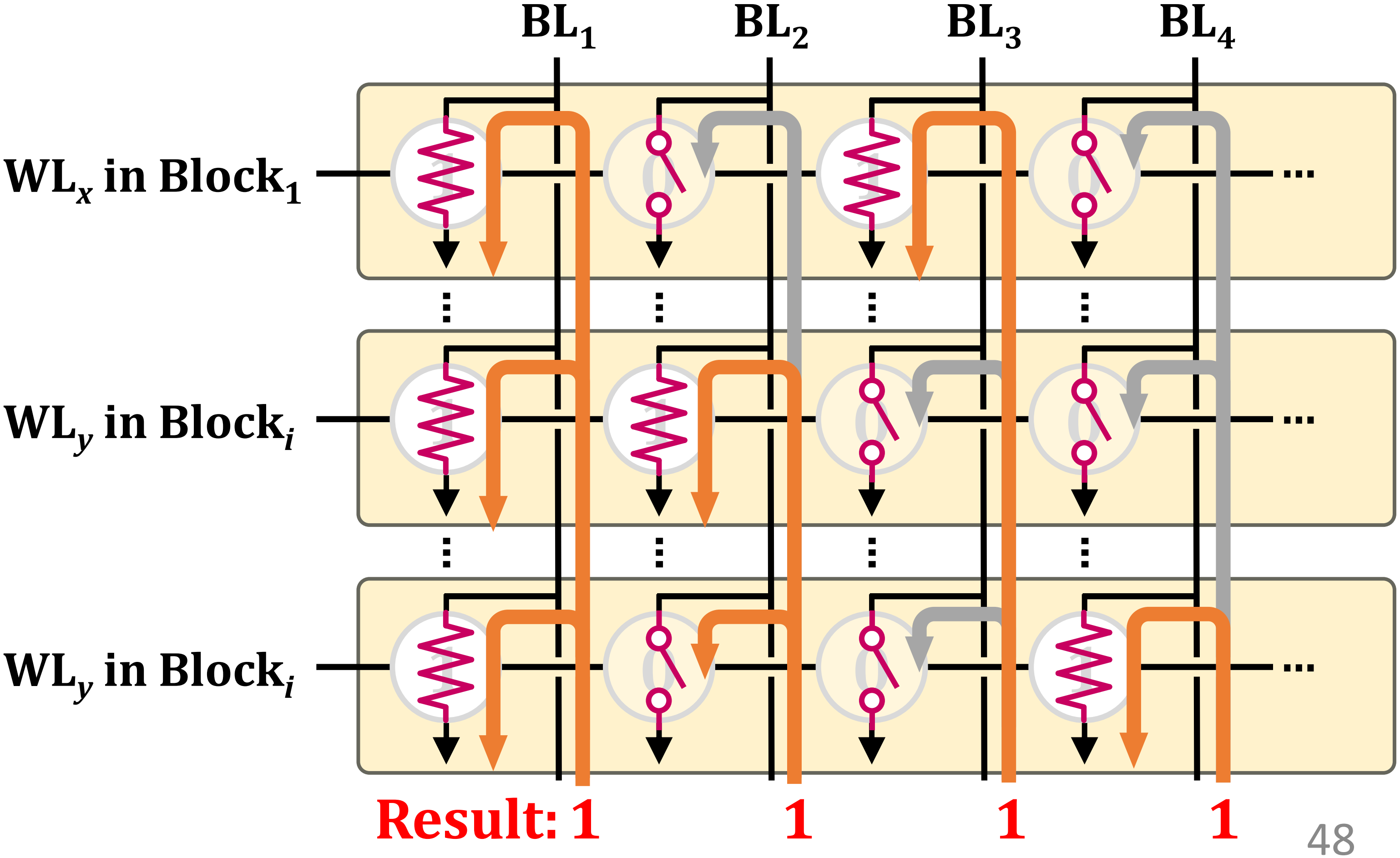
- **Inter-Block MWS**: **Simultaneously activates multiple WLs in different blocks**
  - **Bitwise OR of the stored data in the WLs**



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

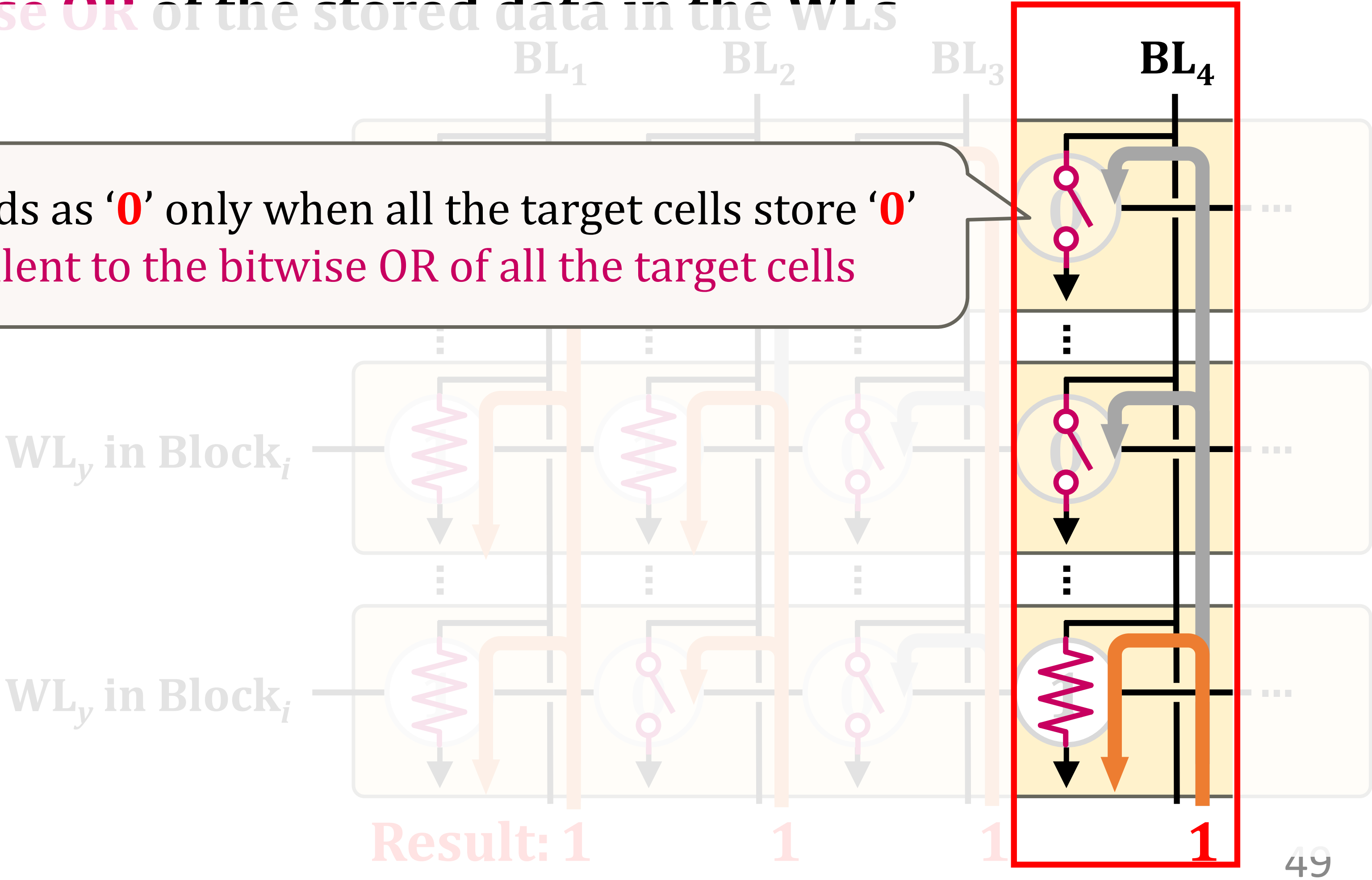
Multi-Wordline Sensing (MWS): Bitwise OR

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



Multi-Wordline Sensing (MWS): Bitwise OR

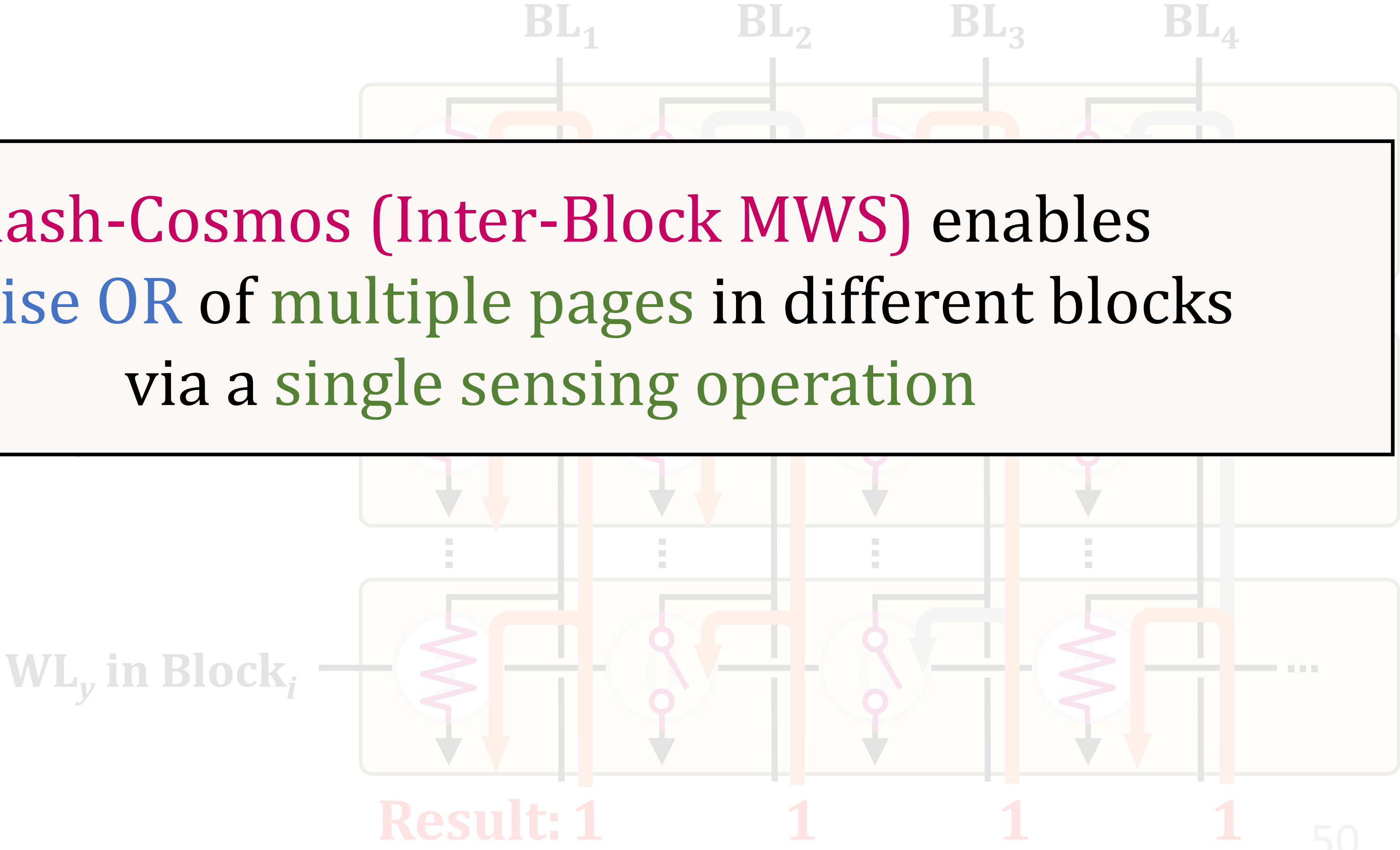
- **Inter-Block MWS**: **Simultaneously activates multiple WLs in different blocks**
  - **Bitwise OR** of the stored data in the WLs



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

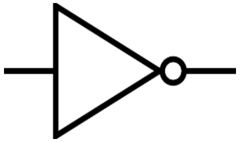
Multi-Wordline Sensing (MWS): Bitwise OR

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



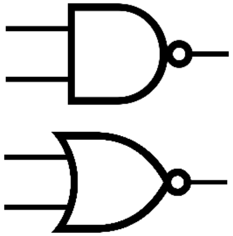
Flash-Cosmos (Inter-Block MWS) enables bitwise OR of multiple pages in different blocks via a single sensing operation

Supporting Other Bitwise Operations



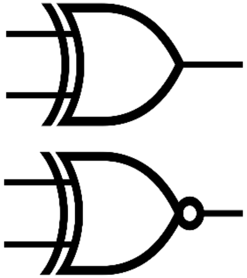
Bitwise NOT

Exploit **Inverse Read**[1] which is supported in modern NAND flash memory



Bitwise NAND/ NOR

Exploit **MWS + Inverse Read**



Bitwise XOR/XNOR

Use **XOR between sensing and cache latches** [2] which is also supported in NAND flash memory

[1] Lee+, “High-Performance 1-Gb-NAND Flash Memory with 0.12-μm Technology,” JSSC, 2002

[2] Kim+, “A 512-Gb 3-b/Cell 64-Stacked WL 3-D-NAND Flash Memory,” JSSC, 2018

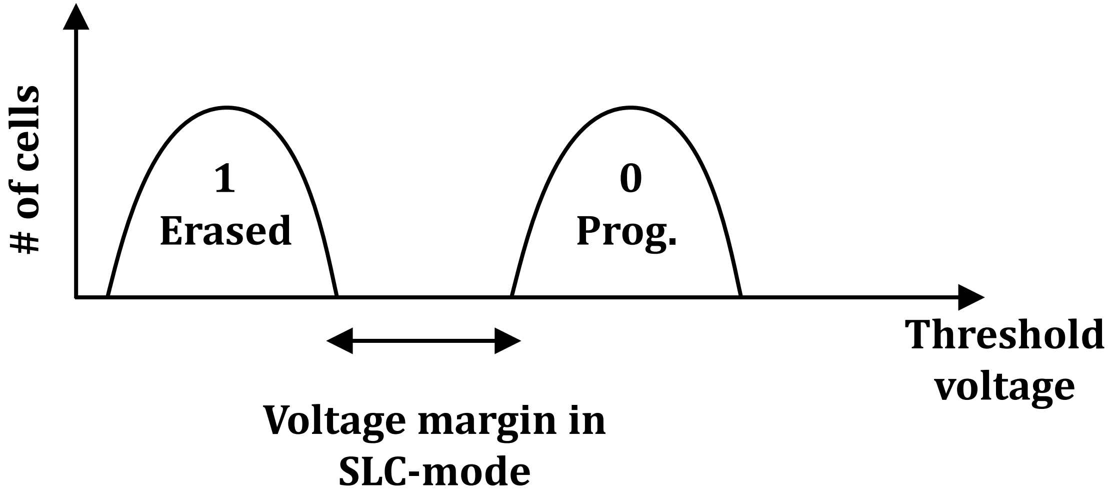
Flash-Cosmos: Overview

Enables in-flash bulk bitwise operations on multiple operands with a *single* sensing operation using Multi-Wordline Sensing (MWS)

Increases the reliability of in-flash bulk bitwise operations by using Enhanced SLC-mode Programming (ESP)

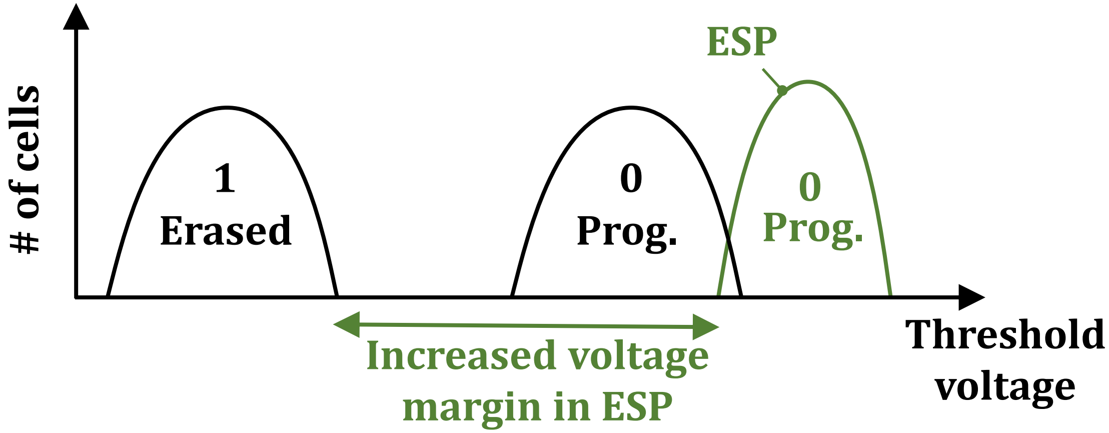
Enhanced SLC-Mode Programming (ESP)

- SLC-mode programming provides a large voltage margin between the erased and programmed states
- Based on our real device characterization, we observe that SLC-mode programming is still highly error-prone without the use of ECC and data-randomization



Enhanced SLC-Mode Programming (ESP)

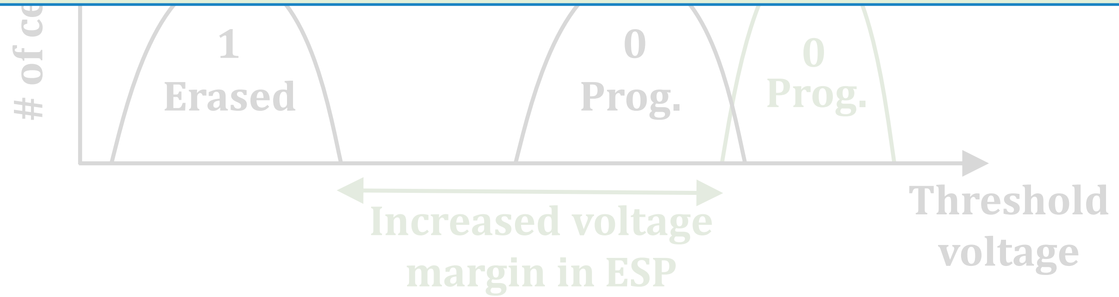
- ESP further increases the voltage margin between the erased and programmed states
- A wider voltage margin between the two states improves reliability by making the cells less vulnerable to errors



Enhanced SLC-Mode Programming (ESP)

- ESP increases the voltage margin between the erased and programmed states
- A wider voltage margin between the two states improves reliability during data sensing by making the cells less vulnerable to errors

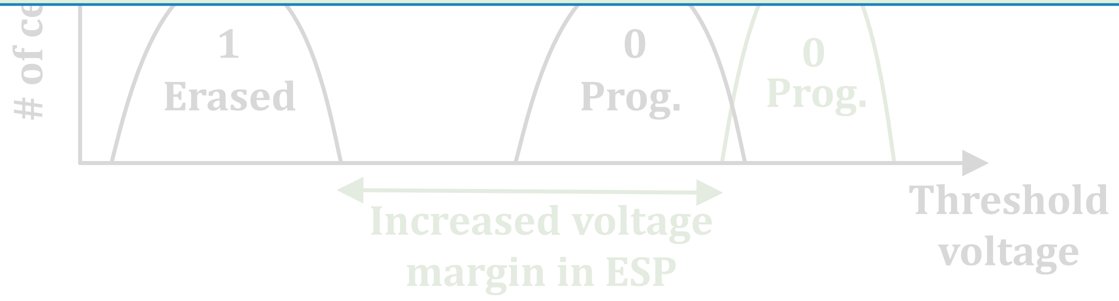
ESP improves the reliability of in-flash computation without the use of ECC or data-randomization techniques



Enhanced SLC-Mode Programming (ESP)

- ESP increases the voltage margin between the erased and programmed states
- A wider voltage margin between the two states improves reliability during data sensing by making the cells less vulnerable to errors

ESP can improve the reliability of prior in-flash processing techniques as well



Talk Outline

Motivation

Background

Flash-Cosmos

Evaluation

Summary

Evaluation Methodology

- We evaluate Flash-Cosmos using

160 real state-of-the-art 3D NAND flash chips

Real Device Characterization

- We validate the feasibility, performance, and reliability of Flash-Cosmos

- 160 48-layer 3D TLC NAND flash chips
  - 3,686,400 tested wordlines

- Under worst-case operating conditions
  - 1-year retention time at 10K P/E cycles
  - Worst-case data patterns

Results: Real-Device Characterization

Both intra- and inter-block MWS operations
require no changes to the cell array
of commodity NAND flash chips

Both MWS operations can activate multiple WLs
(intra: up to 48, inter: up to 4) at the same time
with small increase in sensing latency (< 10%)

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

Evaluation Methodology

- We evaluate Flash-Cosmos using

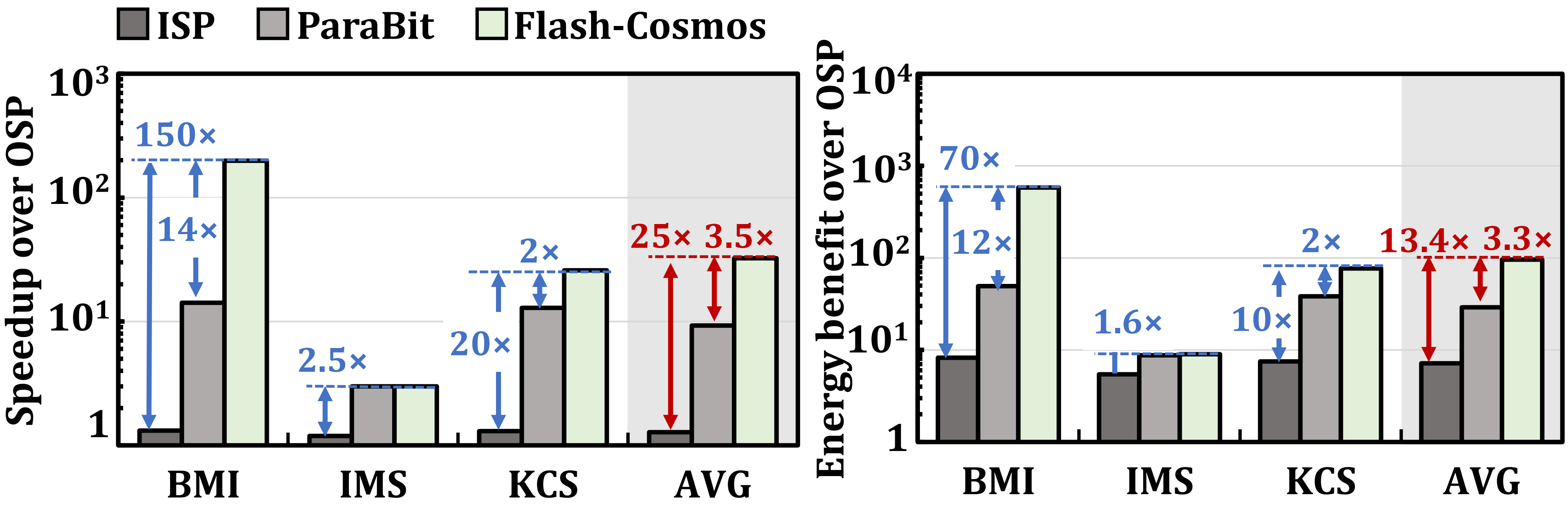
160 real state-of-the-art 3D NAND flash chips

Three real-world applications that perform bulk bitwise operations

Evaluation with real-world workloads

- **Simulation**
  - **MQSim [Tavakkol+, FAST'18]** to model the performance of Flash-Cosmos and the baselines
- **Workloads**
  - Three real-world applications that heavily rely on bulk bitwise operations
  - **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]:** the state-of-the-art in-flash processing (IFP) mechanism

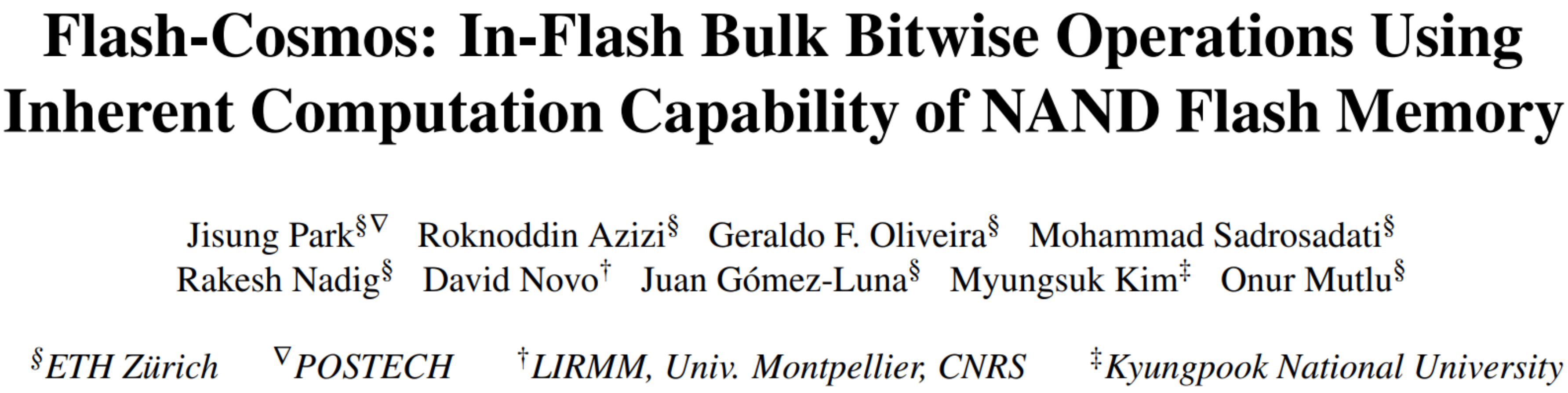
Results: Performance & Energy



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

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

More in the Paper



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

Talk Outline

Motivation

Background

Flash-Cosmos

Evaluation of Flash-Cosmos and Key Results

Summary

Flash-Cosmos: Summary

First work to enable multi-operand bulk bitwise operations with a single sensing operation and high reliability

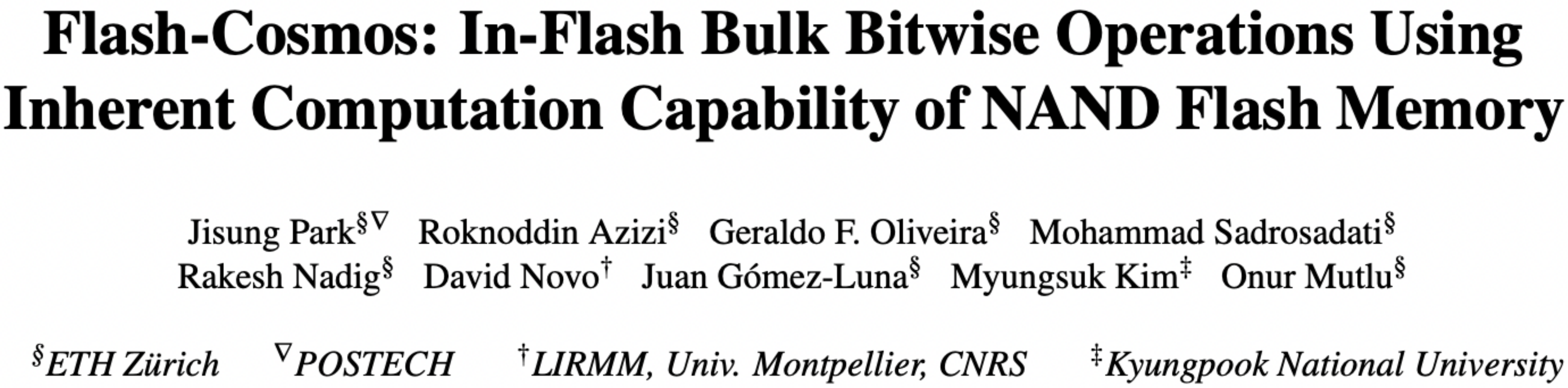
Improves performance by 3.5x/25x/32x on average over ParaBit/ISP/OSP across the workloads

Improves energy efficiency by 3.3x/13.4x/95x on average over ParaBit/ISP/OSP across the workloads

$ Low-cost & requires no changes to flash cell arrays

More on Flash-Cosmos

- 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

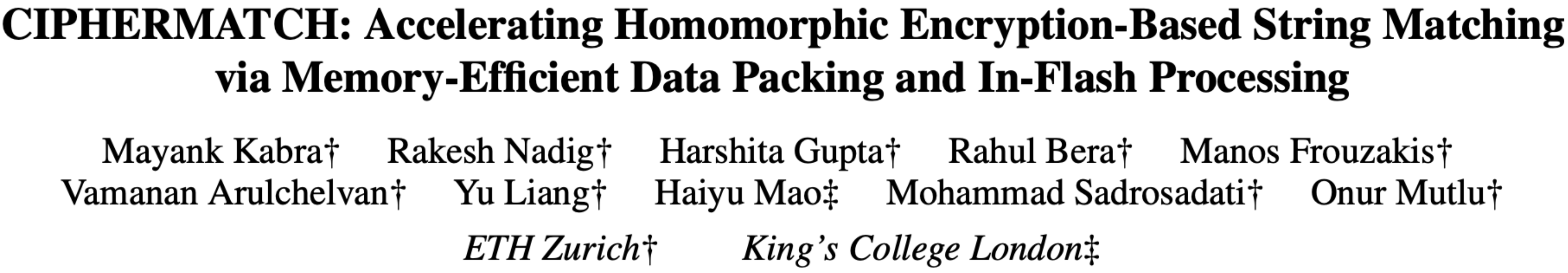
Jisung Park[§∇] Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

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

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

CIPHERMATCH: Accelerating Secure String Matching

- Mayank Kabra, Rakesh Nadig, Harshita Gupta, Rahul Bera, Manos Frouzakis, Vamanan Arulchelvan, Yu Liang, Haiyu Mao, Mohammad Sadrosadati and Onur Mutlu, **"CIPHERMATCH: Accelerating Homomorphic Encryption-Based String Matching via Memory-Efficient Data Packing and In-Flash Processing"** *Proceedings of the 30th International Conference on Architectural Support for Programming Languages and Operating System (**ASPLOS**)*, Rotterdam, Netherlands April 2025.
  [arXiv version]



CIPHERMATCH: Accelerating Homomorphic Encryption-Based String Matching via Memory-Efficient Data Packing and In-Flash Processing

Mayank Kabra† Rakesh Nadig† Harshita Gupta† Rahul Bera† Manos Frouzakis†
Vamanan Arulchelvan† Yu Liang† Haiyu Mao‡ Mohammad Sadrosadati† Onur Mutlu†

ETH Zurich† *King's College London*‡

https://arxiv.org/pdf/2503.08968.pdf

Upcoming Presentation at ASPLOS 2025

CIPHERMATCH: Accelerating Homomorphic Encryption-Based String Matching via Memory-Efficient Data Packing and In-Flash Processing

Mayank Kabra† Rakesh Nadig† Harshita Gupta† Rahul Bera† Manos Frouzakis†
Vamanan Arulchelvan† Yu Liang† Haiyu Mao‡ Mohammad Sadrosadati† Onur Mutlu†
ETH Zurich† *King's College London‡*

To be presented at ASPLOS 2025

Presenter - Mayank Kabra

Visit us in *Session 1D: Homomorphic Encryption*

Location: Van Oldenbarneveld

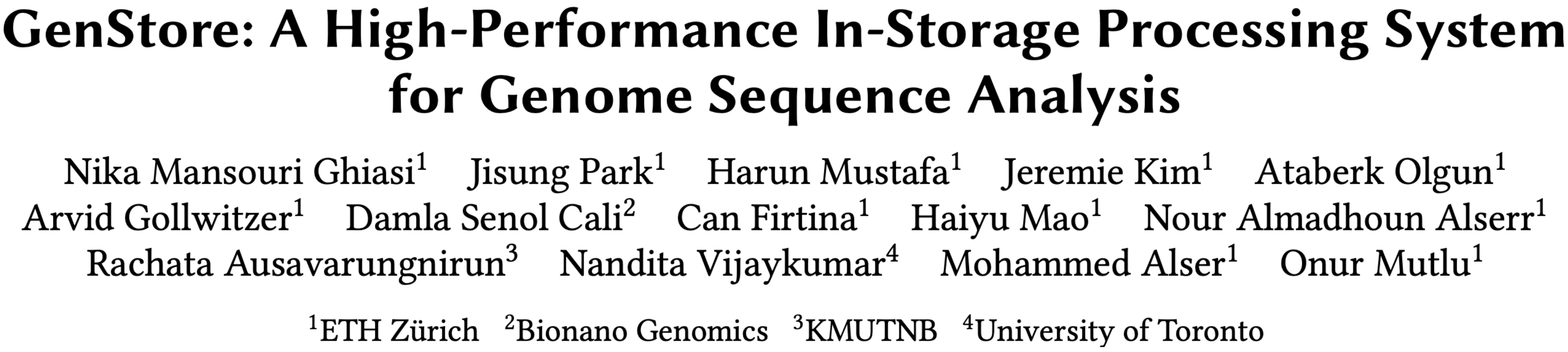
https://arxiv.org/pdf/2503.08968.pdf

Storage-Centric Computing: Two Types

1. Processing near Storage
2. Processing using Storage

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

https://arxiv.org/pdf/2202.10400

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

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

SAFARI

ETH zürich

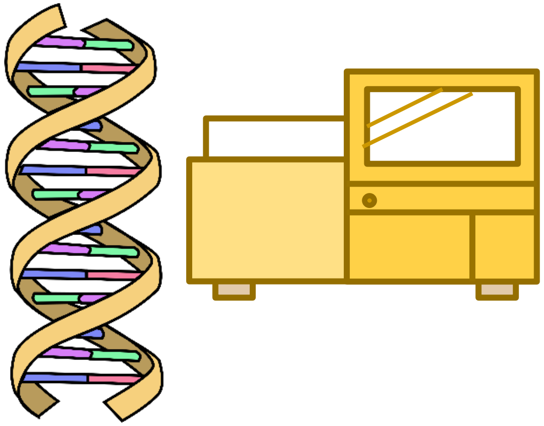
bionano GENOMICS

UNIVERSITY OF TORONTO

Genome Sequence Analysis

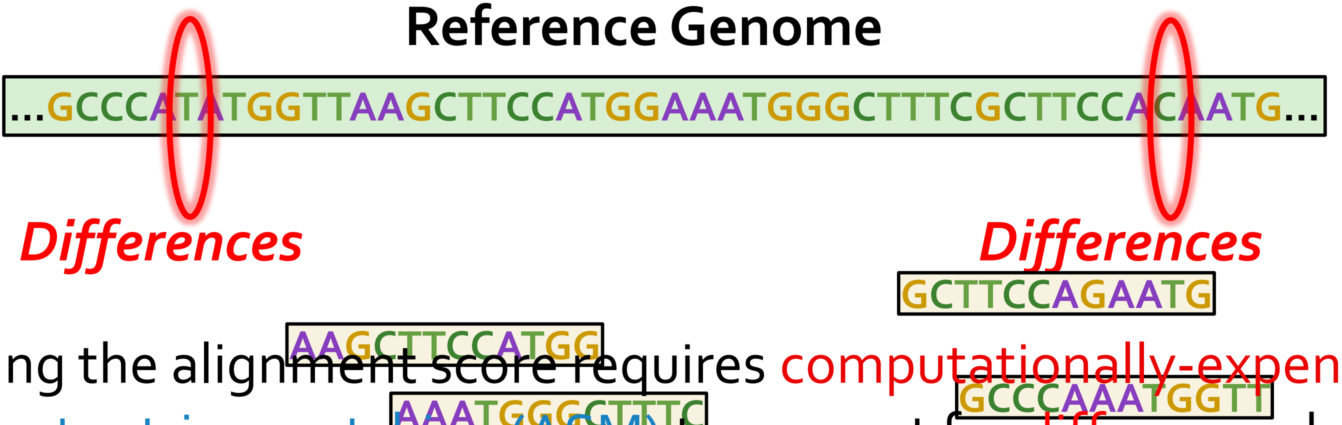
- Genome sequence analysis is critical for many applications
  - ❑ Personalized medicine
  - ❑ Outbreak tracing
  - ❑ Evolutionary studies

- Genome sequencing machines extract smaller fragments of the original DNA sequence, known as reads



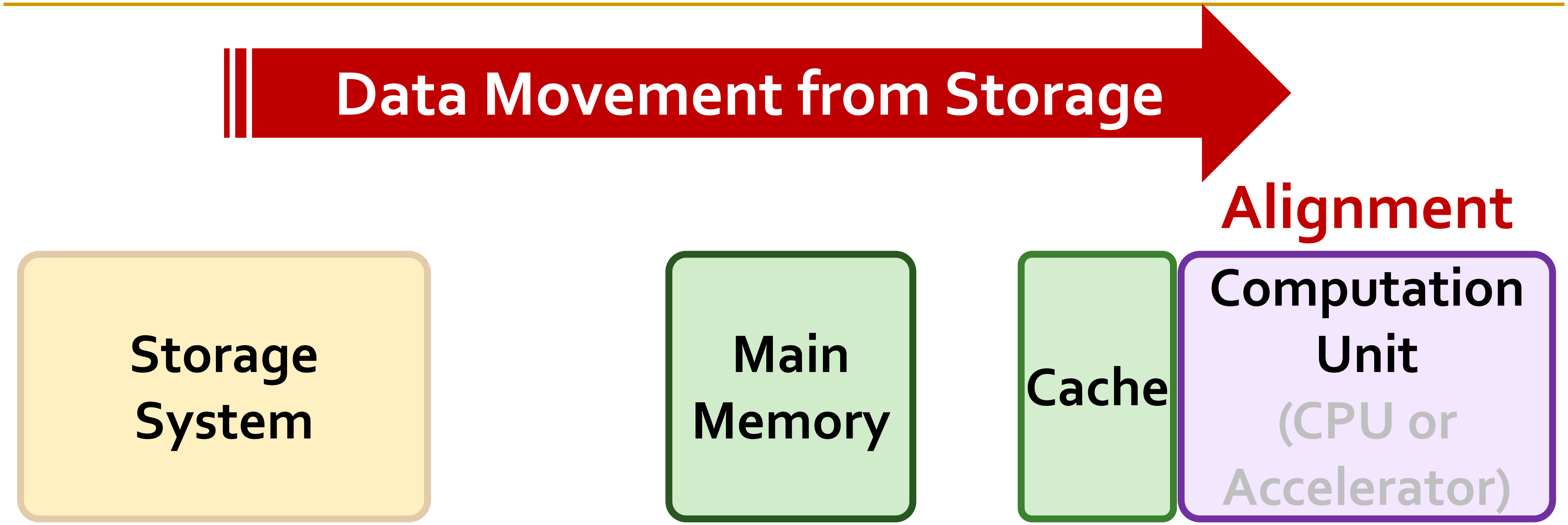
Genome Sequence Analysis

- **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

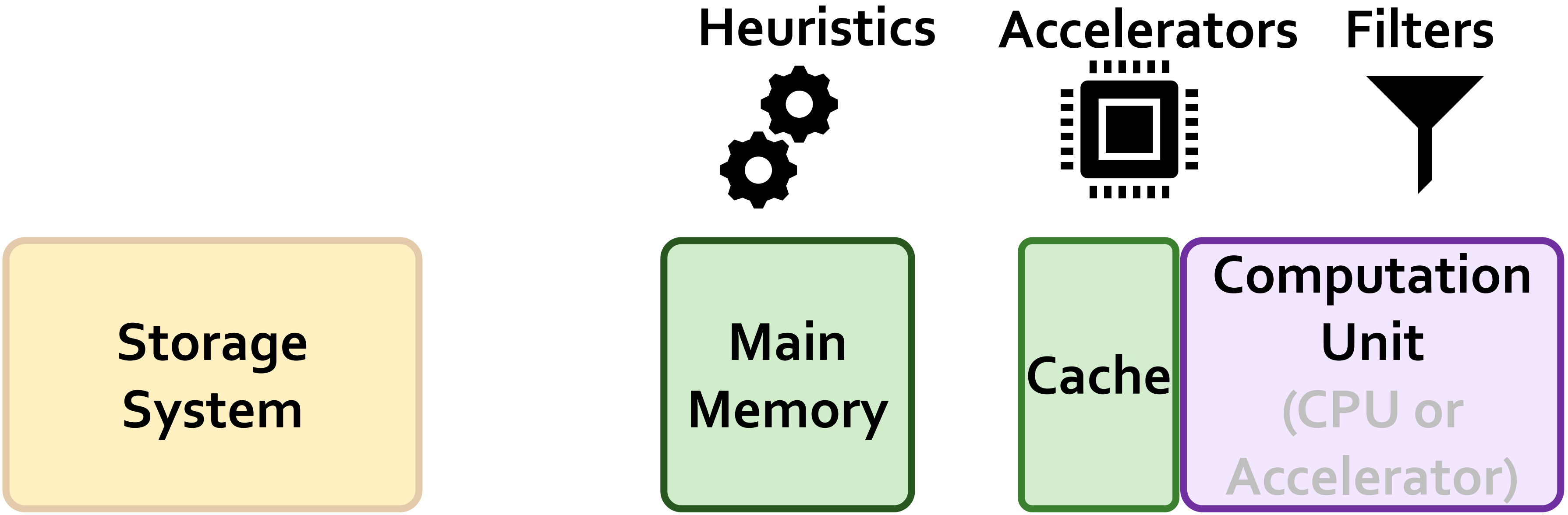
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Accelerating Genome Sequence Analysis

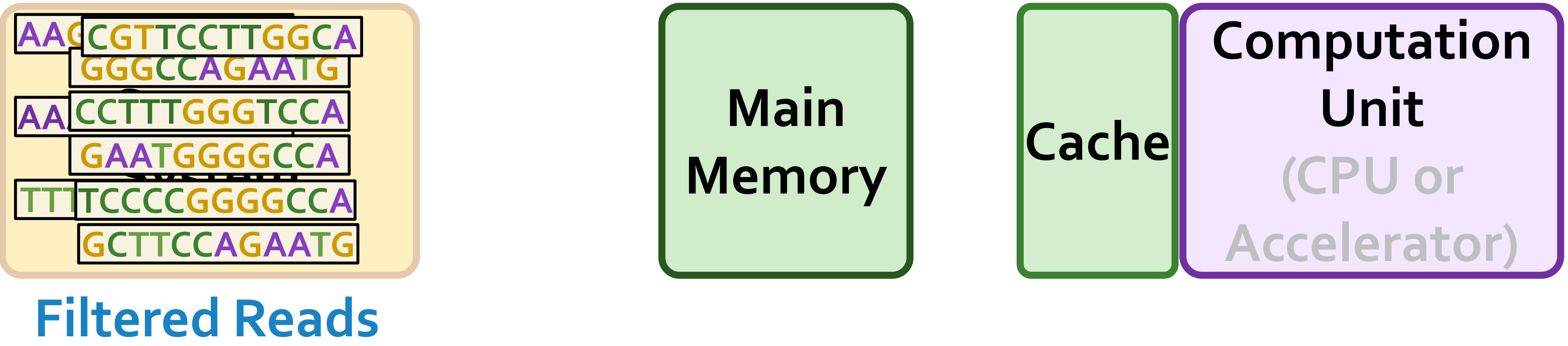


✓ Computation overhead

✗ Data movement overhead

Key Idea

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



Exactly-matching reads

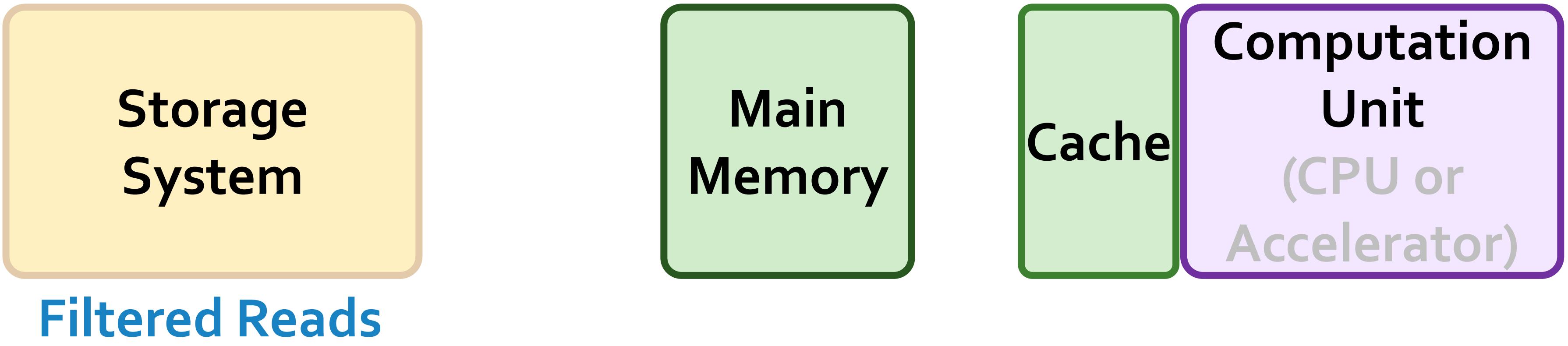
Do not need expensive approximate string matching during alignment

Non-matching reads

Do not have potential matching locations and can skip alignment

Challenges

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

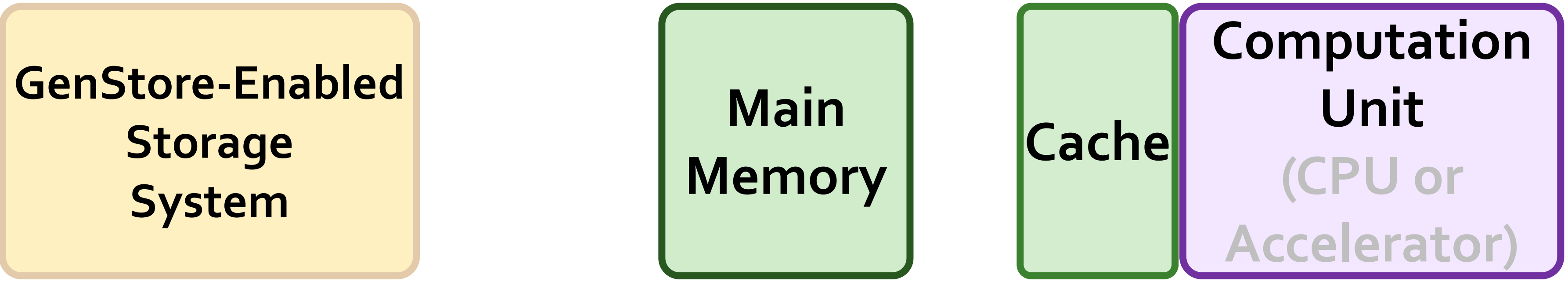


Read mapping workloads can exhibit different behavior

There are limited hardware resources in the storage system

GenStore

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



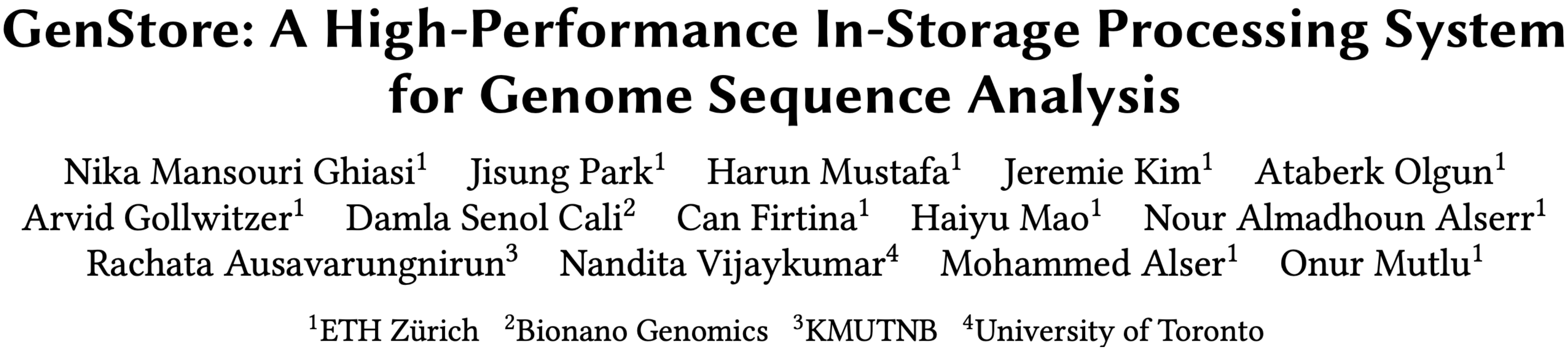
✓ Computation overhead

✓ Data movement overhead

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

More on GenStore

- 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]

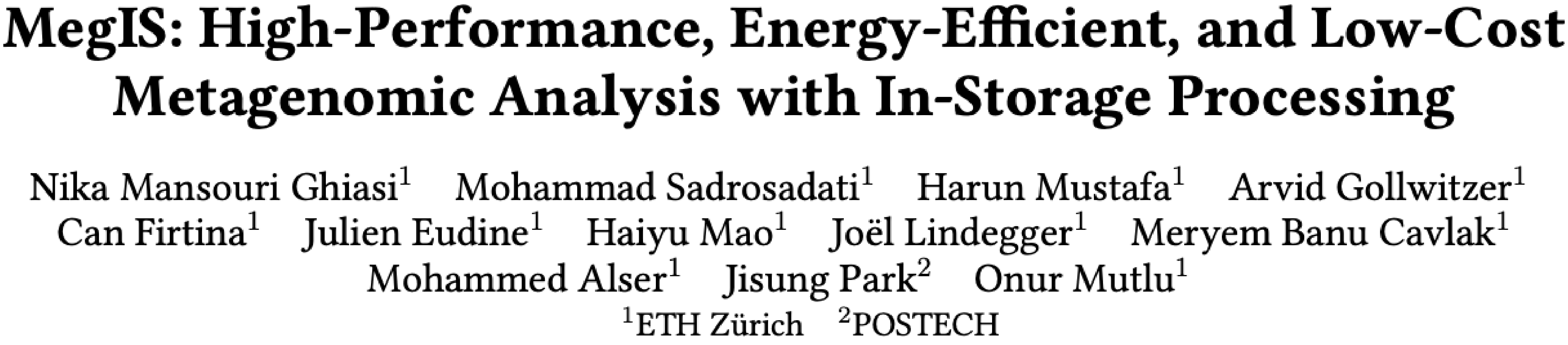
[1]ETH Zürich [2]Bionano Genomics [3]KMUTNB [4]University of Toronto

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

https://arxiv.org/pdf/2202.10400

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

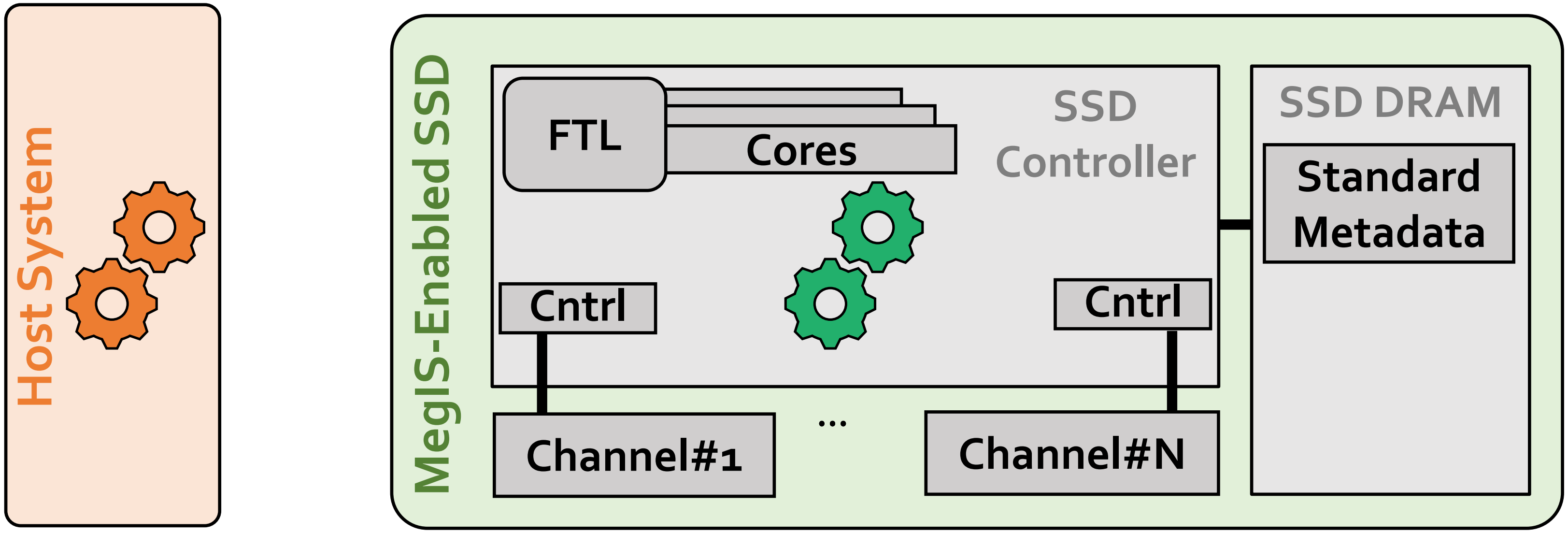
Nika Mansouri Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

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

https://arxiv.org/pdf/2406.19113

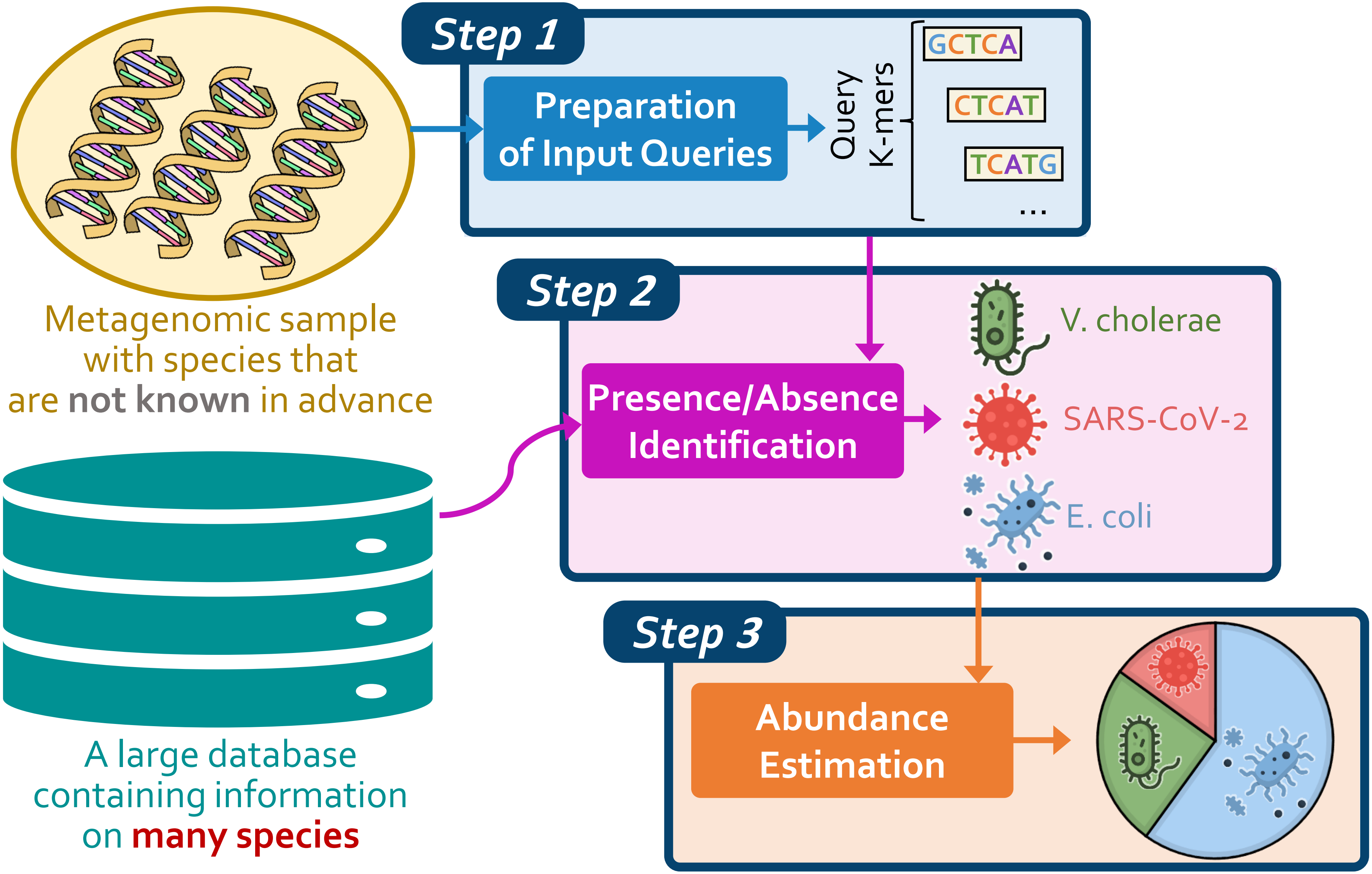
MegIS: Metagenomics In-Storage

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



SAFARI

MegIS's Steps



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

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

Step 1

Preparation of Input Queries

Query K-mers

GCTCA

CTCAT

TCATG

...

Step 2

Presence/Absence Identification

V. cholerae

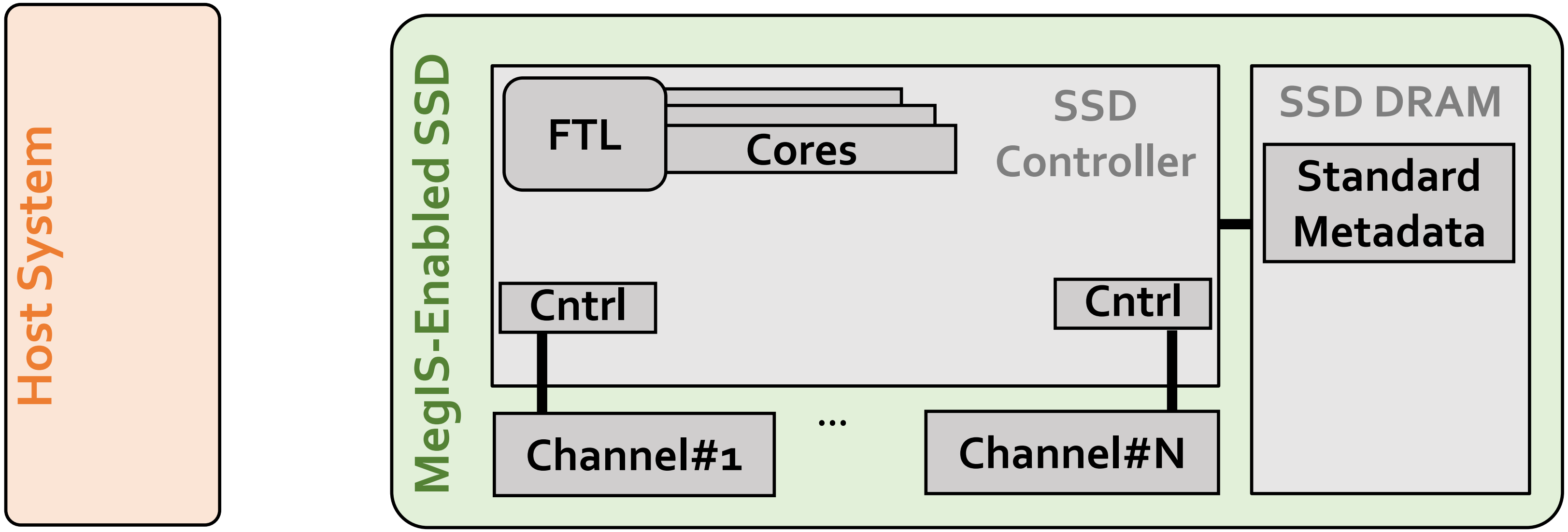
SARS-CoV-2

E. coli

Step 3

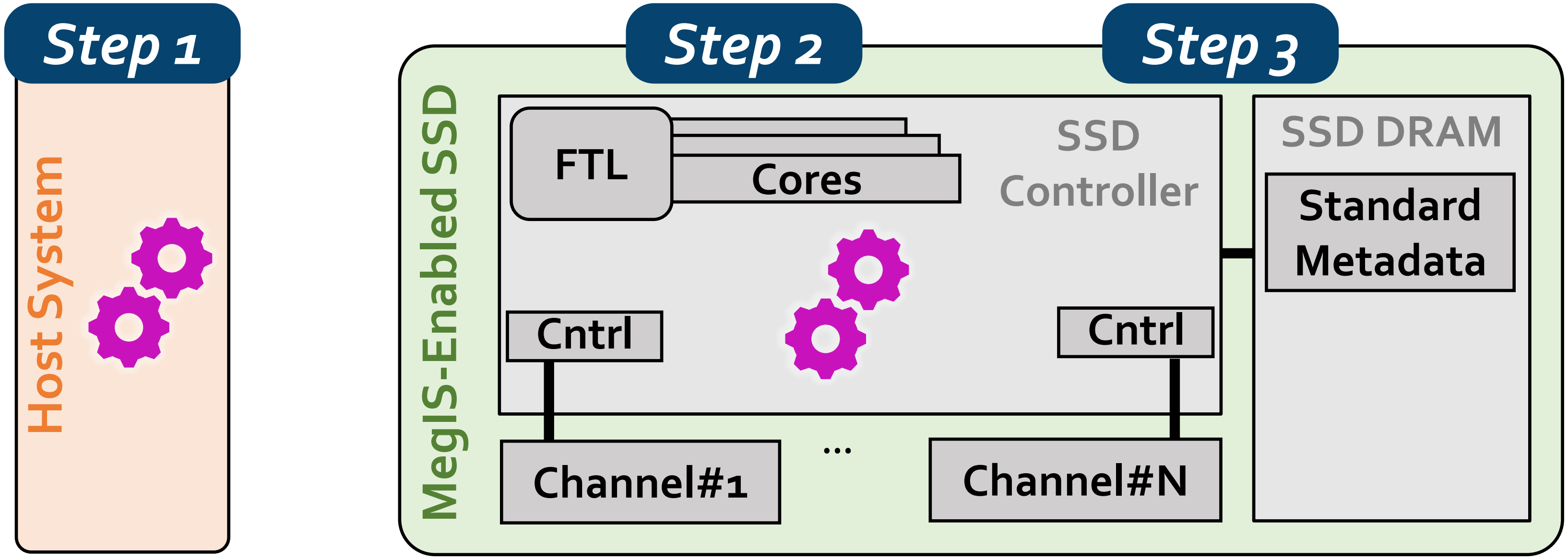
Abundance Estimation

MegIS Hardware-Software Co-Design



MegIS Hardware-Software Co-Design

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



SAFARI

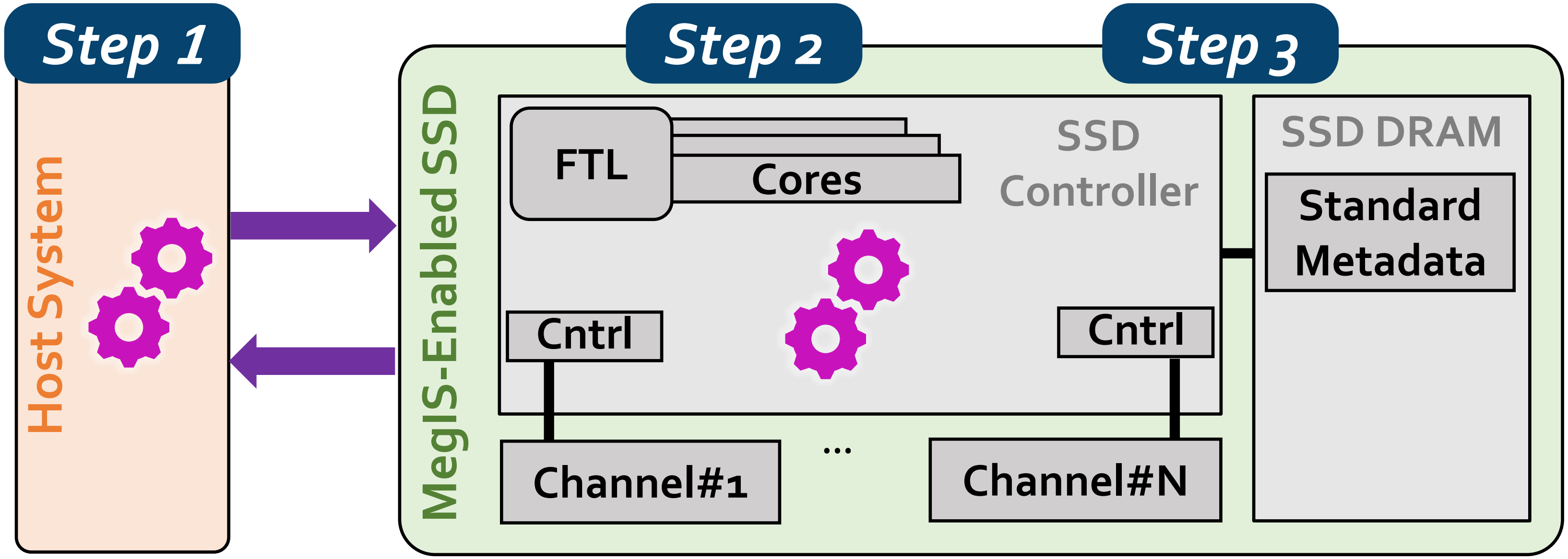
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- *Each step executes in its most suitable system*

Data/computation flow coordination

- *Reduce communication overhead*
- *Reduce #writes to flash chips*



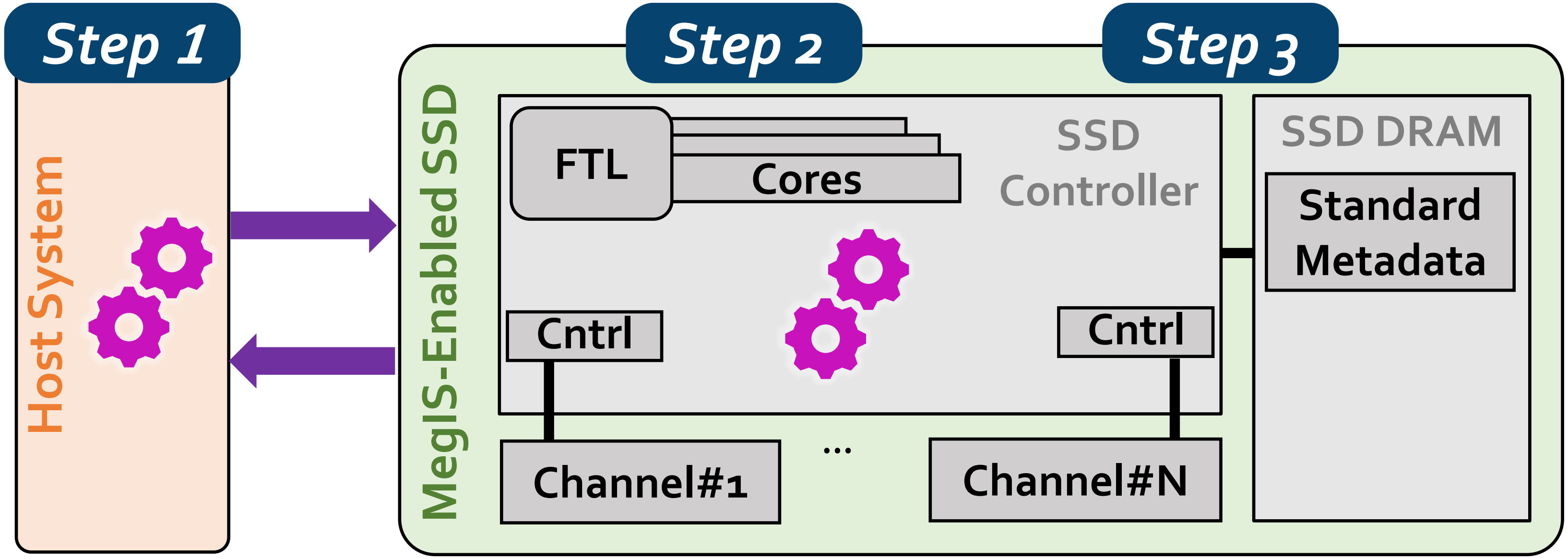
MegIS Hardware-Software Co-Design

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*

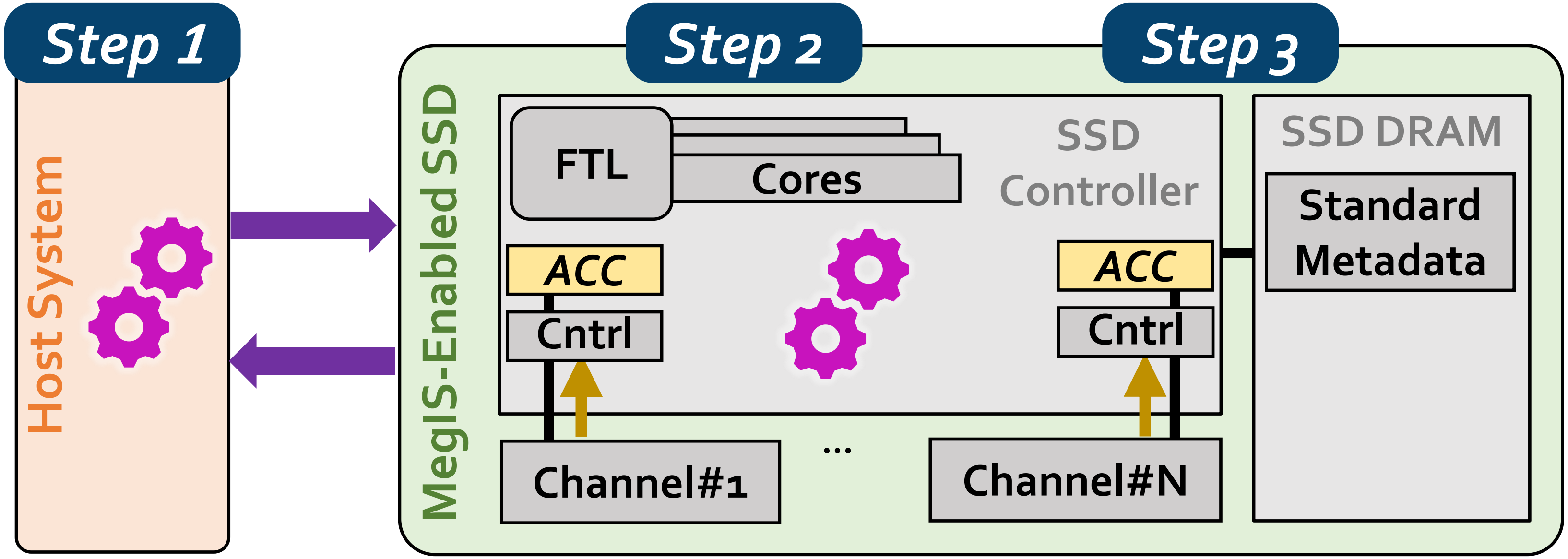
MegIS Hardware-Software Co-Design

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*

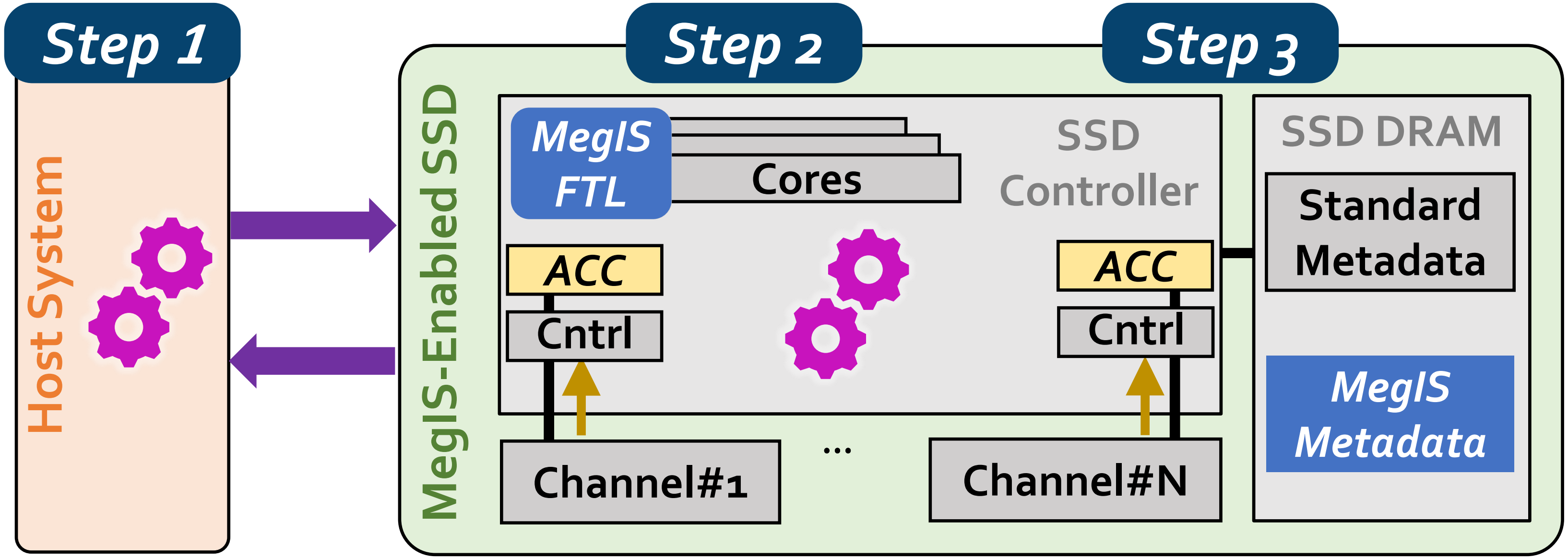
MegIS Hardware-Software Co-Design

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

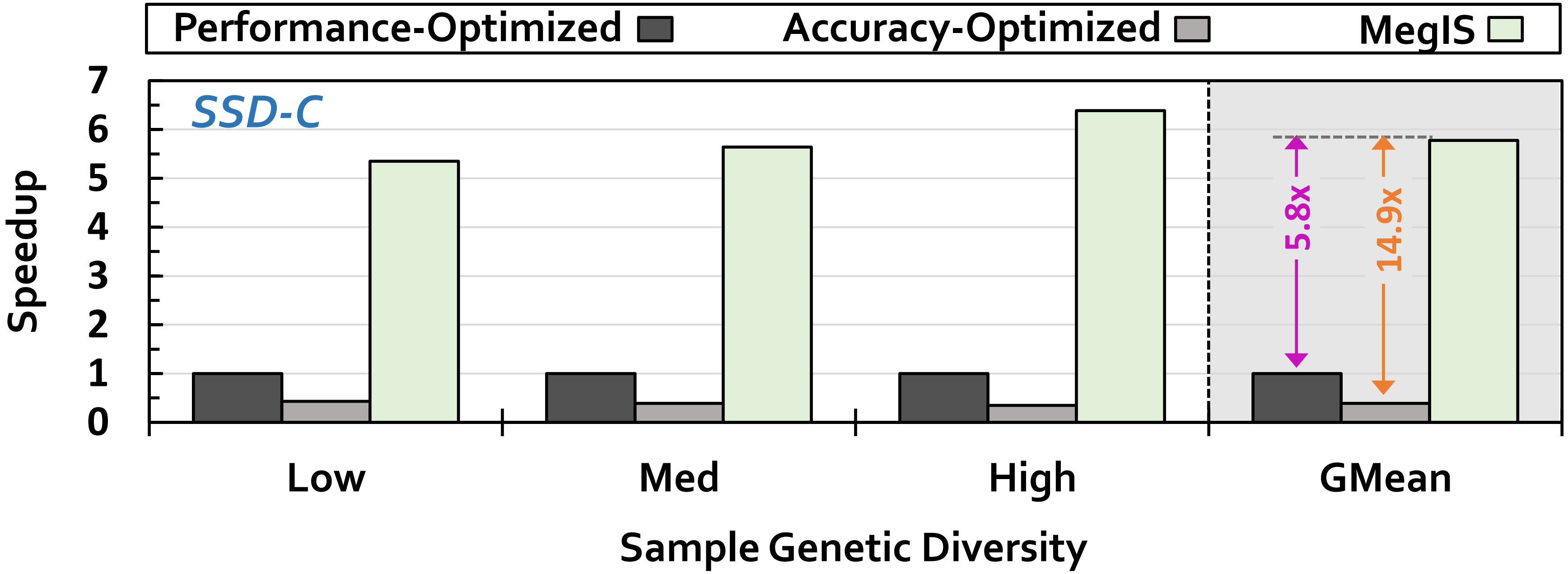
Baseline Comparison Points

- **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

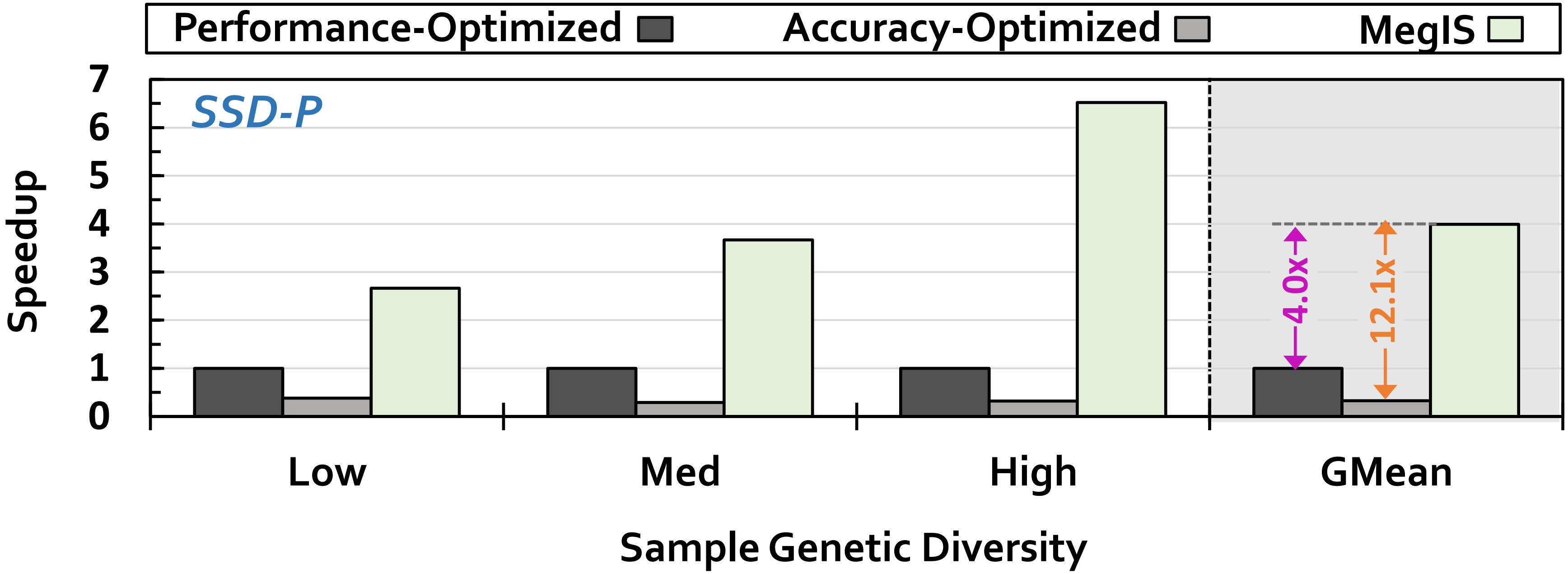
- **SSD-C:** with SATA3 interface (0.5 GB/s sequential read bandwidth)
- **SSD-P:** with PCIe Gen4 interface (7 GB/s sequential read bandwidth)

Evaluation: Speedup over the Software Baselines



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

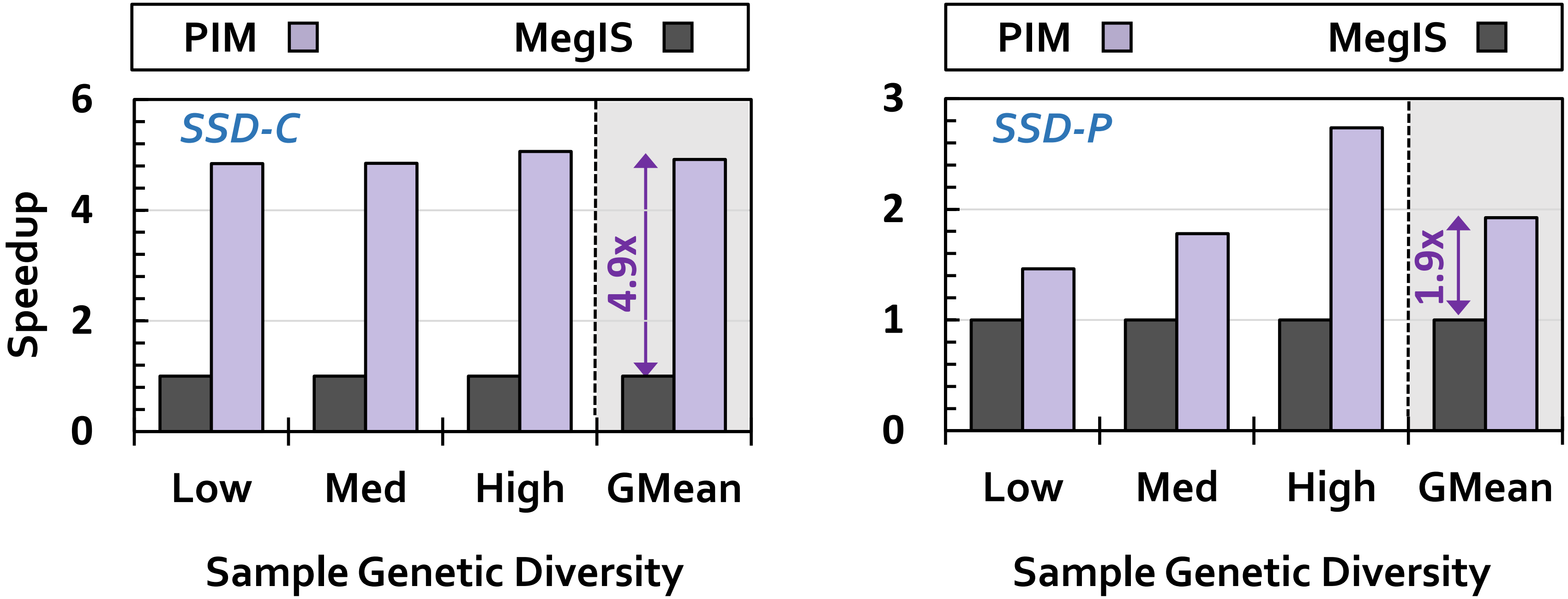
Evaluation: Speedup over the Software Baselines



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

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

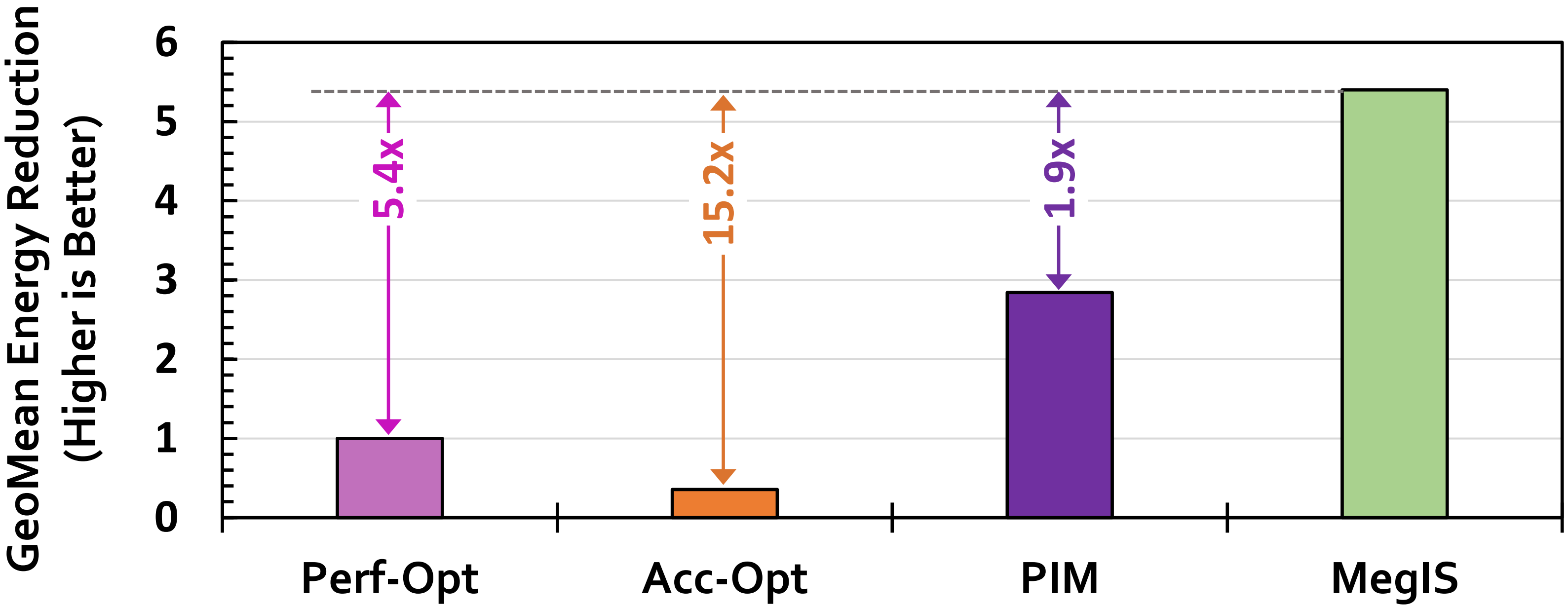
Evaluation: Speedup over the PIM Baseline



MegIS provides significant speedup over the PIM baseline

Evaluation: 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

SAFARI

Evaluation: 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 scores
  - 3 – 24% lower L1 norm error

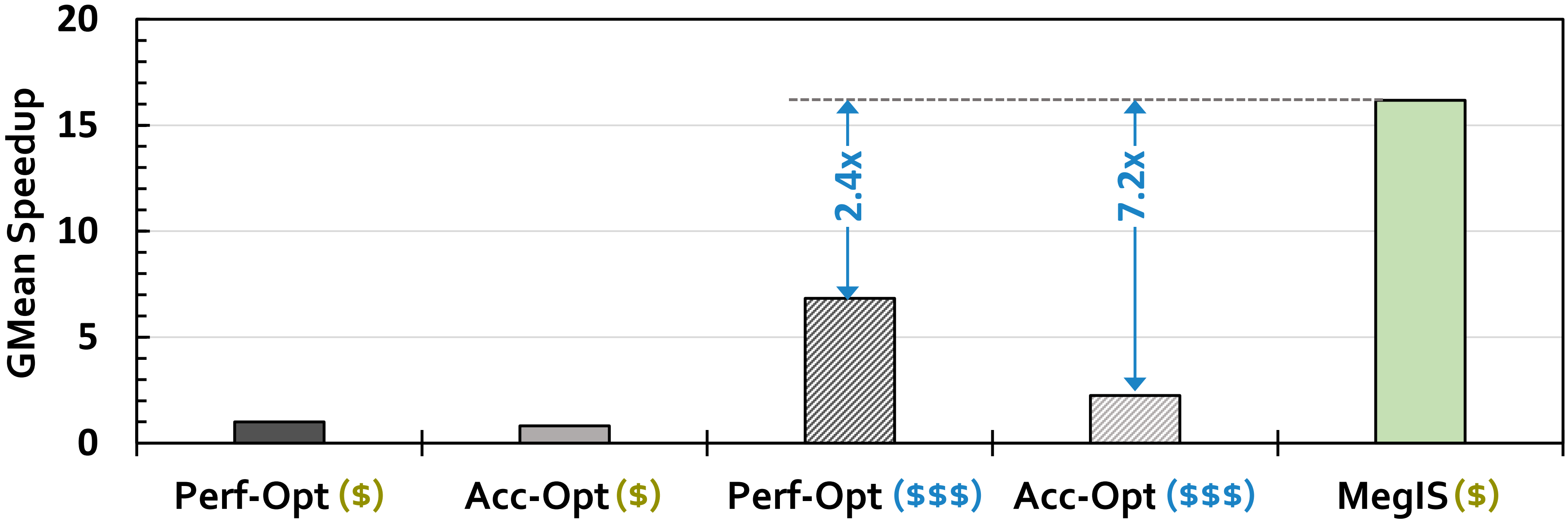
Area and Power

Total for an 8-channel SSD:

- **Area:** 0.04 mm^2 *(Only **1.7%** of the area of three ARM Cortex R4 cores in an SSD controller)*
- **Power:** 7.658 mW

Evaluation: System Cost-Efficiency

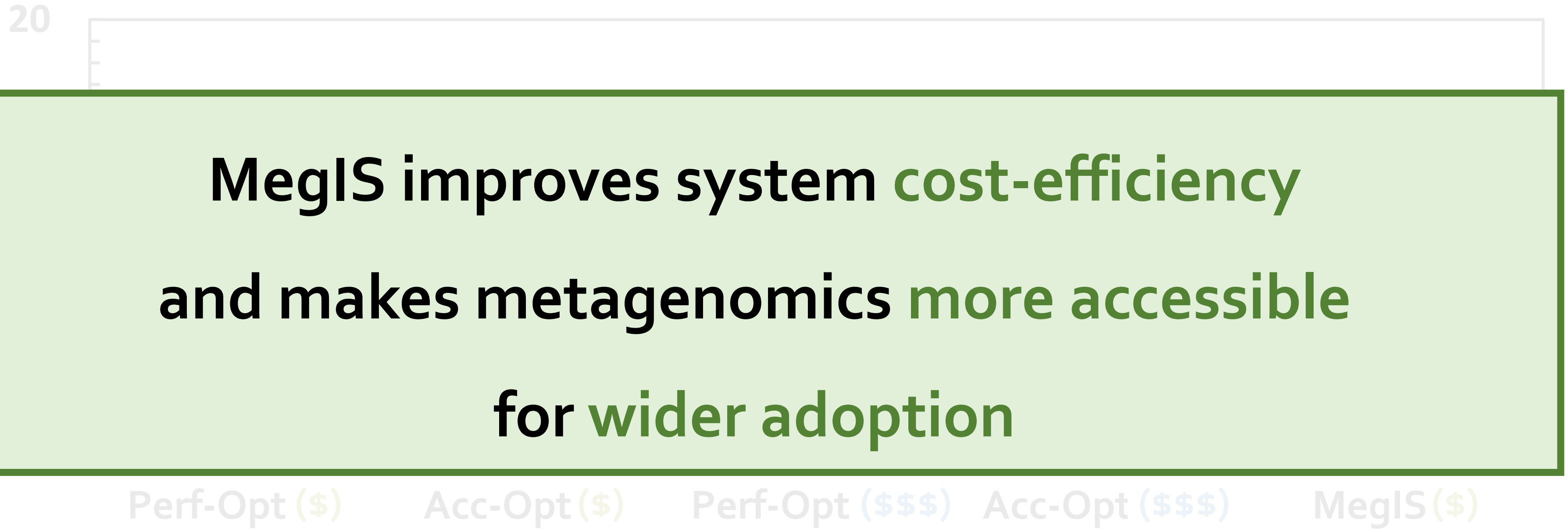
- **Cost-optimized system ($):** With SSD-C and 64-GB DRAM
- **Performance-optimized system ($$$):** With SSD-P and 1-TB DRAM



MegIS outperforms the baselines

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

- 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 metagenomics more accessible for wider adoption

MegIS outperforms the baselines *even* when running on a much less costly system

More in the Paper

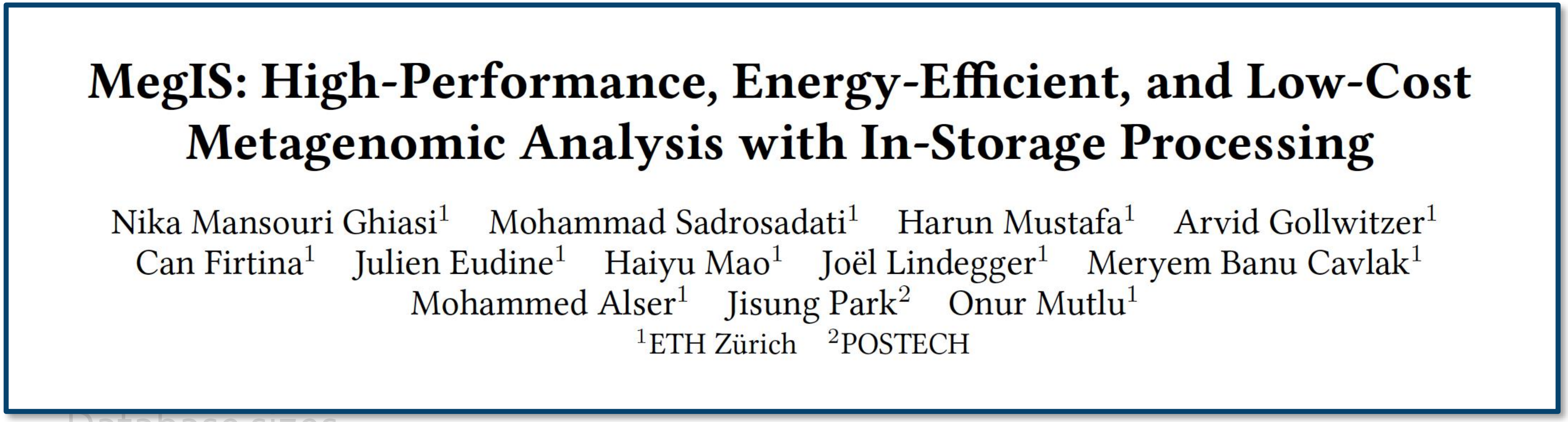
- MegIS's performance when running in-storage processing operations on the **cores existing in the SSD controller**

- MegIS's performance when using the same accelerators **outside SSD**

- **Sensitivity analysis with varying**
  - Database sizes
  - Memory capacities
  - #SSDs
  - #Channels
  - #Samples

- MegIS's performance for **abundance estimation**

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

SAFARI

MegIS: Summary

Metagenomic analysis suffers from
significant storage I/O data movement overhead

MegIS

The *first* **in-storage processing** system for *end-to-end* metagenomic analysis
Leverages and orchestrates **processing inside** and **outside** the storage system

Improves performance

2.7×–37.2× over performance-optimized software
6.9×–100.2× over accuracy-optimized software
1.5×–5.1× over hardware-accelerated PIM baseline

High accuracy

Same as accuracy-optimized
4.8× higher F1 scores
over **performance-optimized/PIM**

Reduces energy consumption

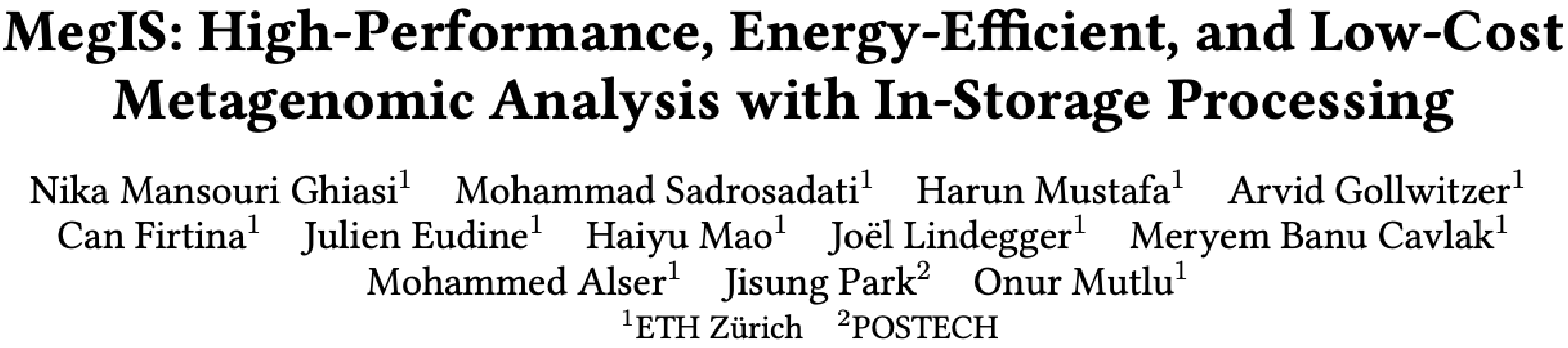
5.4× over **performance-optimized** software
15.2× over **accuracy-optimized** software
1.9× over **hardware-accelerated PIM** baseline

Low area overhead

1.7% of the three cores
in an SSD controller

More on MegIS

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

https://arxiv.org/pdf/2406.19113

Storage-Centric Computing: Two Types

1. Processing near Storage
2. Processing using Storage

Summary and Future Outlook

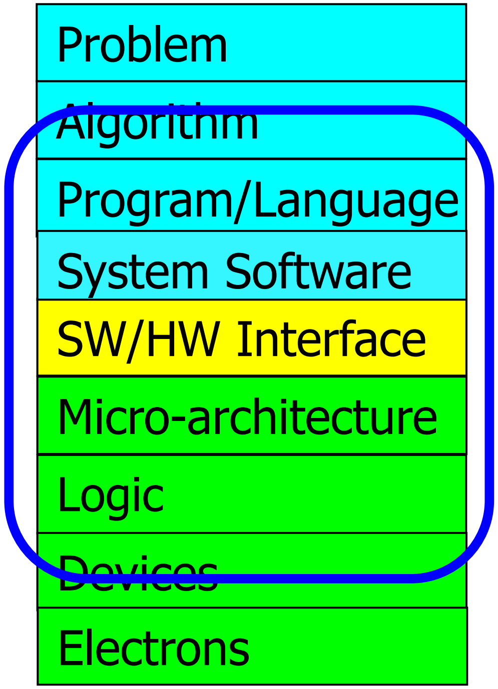
Our Vision on Storage-Centric Computing

- Entire storage system as a **specialized-enough accelerator**
  - Special-purpose accelerators
  - General-purpose computation
  - Multiple different memory technologies
    - Processing-using-Flash/DRAM
    - Processing-near-Flash/DRAM

- Storage system becomes a **first-class citizen** where computation takes place when it makes
  - greatly improving **performance, energy efficiency, system cost, sustainability**, …

Storage-Centric Computing: Some Challenges

- Reliability of computation
- Limited endurance
- Higher latencies of flash memories
- Small internal DRAMs
- Limited power and area budgets
- Programming framework
- Security guarantees
- …



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

We can get there step by step

1st Workshop on Memory-Centric Computing: Storage-Centric Computing

Mohammad Sadrosadati

m.sadr89@gmail.com

ASPLOS 2025

30 March 2025