EFCL Summer School -- Track 4
Lecture 8b: Storage-Centric Computing

Mohammad Sadrosadati
Prof. Onur Mutlu

ETH Zürich
Summer 2024
6 June 2024
Brief Self Introduction

Mohammad Sadrosadati

- Senior Researcher and Lecturer @ SAFARI Research Group, ETHZ
- PhD from Sharif University of Technology, 2014-2019
- mohammad.sadrosadati@safari.ethz.ch

Research Area

- Computer Architecture
- Memory/Storage Systems
- Near-Data Processing
- Heterogeneous System Architecture
- Bioinformatics
- Interconnection Network
The Problem

Computing is Bottlenecked by Data
How to Handle Data Well

- **Ensure data does not overwhelm** the components
  - via intelligent algorithms, architectures & system designs: algorithm-architecture-devices

- **Take advantage of** vast amounts of **data** and metadata
  - to improve architectural & system-level decisions

- **Understand and exploit** properties of (different) **data**
  - to improve algorithms & architectures in various metrics
Corollaries: Computing Systems Today …

- Are processor-centric vs. data-centric

- Make designer-dictated decisions vs. data-driven

- Make component-based myopic decisions vs. data-aware
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Computing Systems"
  Invited Paper in Proceedings of the Design, Automation, and Test in
  Europe Conference (DATE), Virtual, February 2021.
  [Slides (pptx) (pdf)]
  [IEDM Tutorial Slides (pptx) (pdf)]
  [Short DATE Talk Video (11 minutes)]
  [Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com
Data-Centric Architectures
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
Challenge and Opportunity for Future

High Performance, Energy Efficient, Sustainable
(All at the Same Time)
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
Goal: Processing Inside Memory/Storage

- Many questions ... How do we design the:
  - compute-capable storage/memory systems?
  - processors & communication units?
  - software & hardware interfaces?
  - system software, compilers, languages?
  - algorithms & theoretical foundations?
We Need to Think Differently from the Past Approaches
Processing inside Storage: Two Approaches

1. In-Flash Processing
2. Processing near Flash Memory
In-Flash Bulk Bitwise Execution

- Jisung Park, Roknoddin Azizi, Geraldo F. Oliveira, Mohammad Sadrosadati, Rakesh Nadig, David Novo, Juan Gómez-Luna, Myungsuk Kim, and Onur Mutlu,

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

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

[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (44 minutes)]
[arXiv version]

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

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

§ETH Zürich  ∀POSTECH  †LIRMM, Univ. Montpellier, CNRS  ‡Kyungpook National University
Flash-Cosmos
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, and Onur Mutlu

MICRO 2022
Executive Summary

- **Background:** Bulk bitwise operations are widely used in many important data-intensive applications, e.g., databases, graph processing, cryptography etc.

- **Problem:**
  - Performance and energy efficiency of bulk bitwise operations are bottlenecked by
    1) data movement between storage and the compute unit in traditional systems
    2) data sensing (serial reading of operands) in prior in-flash processing (IFP) techniques
  - Prior IFP techniques provide low reliability during computation

- **Goal:** Improve performance, energy efficiency and reliability of bulk bitwise operations in in-flash processing

- **Key Idea:** Flash-Cosmos (Flash-Computation with One-Shot Multi-Operand Sensing) is an in-flash processing technique that improves performance, energy efficiency and reliability of bulk bitwise operations using two key techniques:
  - Multi-Wordline Sensing (MWS): Enables multi-operand bulk bitwise operations with a single sensing (read) operation
  - Enhanced SLC-mode Programming (ESP): Increases the voltage margin between the erased and programmed states to provide higher reliability during in-flash computation

- **Key Results:** Flash-Cosmos is evaluated using 160 real 3D NAND flash chips and three real-world workloads
  - Flash-Cosmos improves the performance and energy efficiency by 3.5x and 3.3x over state-of-the-art IFP technique while providing high reliability during computation
Bulk Bitwise Operations

- Hyper-dimensional Computing
- Cryptography
- Set Operations
- Graph Processing
- Databases
- Web Search
- Genome Analysis
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.

The external I/O bandwidth of storage is the main bottleneck for data movement in OSP.
NDP for Bulk Bitwise Operations

Our focus
Large data sets that do not fit in main memory

In-Flash (e.g., ParaBit\(^5\))

In-Storage (e.g., Biscuit\(^4\))

Near-Data Processing

Cache
(e.g., Compute Cache\(^1\))

DRAM-based main memory
(e.g., Ambit\(^2\))

NVM-based main memory
(e.g., Pinatubo\(^3\))

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

![Diagram showing ISP](image)

**Host Processor (CPU, GPU)**

**Main Memory**

**In-Storage Computation Unit**

**Storage**

**Data Movement Bottleneck**

- 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

**SAFARI**
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
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

- 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.

IFP fundamentally mitigates the data movement.
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

![NAND Flash Chip Diagram]

**NAND Flash Chip**

| A | B | C | D | ...
|---|---|---|---|

**Page Buffer**

**Data Sensing**
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique \cite{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.
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
NAND Flash Basics: A Flash Cell

• A flash cell stores data by adjusting the amount of charge in the cell.

Erased Cell (Low Charge Level)

Programmed Cell (High Charge Level)

 Activation

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erased Cell</td>
<td>Programmed Cell</td>
</tr>
<tr>
<td>Operates as a resistor</td>
<td>Operates as an open switch</td>
</tr>
</tbody>
</table>
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**

![Diagram showing NAND Flash Basics]

- **Bitline (BL)**
- **Non-Target Cells:**
  - Operate as **resistors** regardless of stored data

**Diagram Details:**
- Blue circles represent non-target cells.
- Dashed line indicates electrical connection.

**Legend:**
- Blue circles with spiral indicate non-target cells.
- Straight line indicates a path through the NAND string.
- **SAFARI** logo at the bottom.
NAND Flash Basics: Read Mechanism

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

  Bitline (BL)

  **Target Cells:**
  \( \text{Operate as resisters (1) or open switches (0)} \)

  **Non-Target Cells:**
  \( \text{Operate as resisters regardless of stored data} \)
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*

NAND String

BL$_i$

BL$_j$

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

\[ WL_1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad \ldots \quad 1 \]

\[ WL_2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \ldots \quad 0 \]

\[ WL_3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \ldots \quad 0 \]

\[ WL_4 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad \ldots \quad 1 \]

\[ WL_M \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \ldots \quad 0 \]

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

2-input NAND

\[(A \cdot B)'\]
A large number of blocks share the same bitlines.

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

2-input NAND\[\overline{A \cdot B}\]

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

2-input NOR\[\overline{A + B}\]
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

```
0 0 1 1
0 1 0 1
1 0 1 1
0 0 1 0
```

```
WL_1
WL_2
WL_3
WL_4
```
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

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

- **Result**: 0 0 0 1
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)

Result: 0 0 0 0
Multi-Wordline Sensing (MWS): Bitwise AND

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

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)

Result: 0 0 0 0

0
Multi-Wordline Sensing (MWS): Bitwise AND

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

\[
\begin{array}{cccc}
\text{BL}_1 & \text{BL}_2 & \text{BL}_3 & \text{BL}_4 \\
0 & 0 & 1 & 1 \\
\end{array}
\]

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

Result: A bitline reads as ‘1’ only when all the target cells store ‘1’

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

\[
\begin{array}{cccc}
\text{WL}_x \text{ in Block}_1 & \text{BL}_1 & \text{BL}_2 & \text{BL}_3 & \text{BL}_4 \\
1 & 0 & 1 & 0 \\
\vdots & \vdots & \vdots & \vdots \\
\text{WL}_y \text{ in Block}_i & \text{BL}_1 & \text{BL}_2 & \text{BL}_3 & \text{BL}_4 \\
1 & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]
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

![Diagram of Multi-Wordline Sensing](image)

Result: **1** 1 1 0
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

![Diagram showing multi-wordline sensing](image)
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

Exploit **Inverse Read**\(^1\) which is supported in modern NAND flash memory

Exploit **MWS + Inverse Read**

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.

![Diagram showing increased voltage margin in ESP]
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 $\sim 1,000$ 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

<table>
<thead>
<tr>
<th></th>
<th>ISP</th>
<th>ParaBit</th>
<th>Flash-Cosmos</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI</td>
<td>150×</td>
<td>14×</td>
<td>10×</td>
</tr>
<tr>
<td>IMS</td>
<td>2.5×</td>
<td>2×</td>
<td>10×</td>
</tr>
<tr>
<td>KCS</td>
<td>20×</td>
<td>25×</td>
<td>3.5×</td>
</tr>
<tr>
<td>AVG</td>
<td>10×</td>
<td>3×</td>
<td>1.6×</td>
</tr>
</tbody>
</table>

**Speedup over OSP**

<table>
<thead>
<tr>
<th></th>
<th>Energy benefit over OSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI</td>
<td>70×</td>
</tr>
<tr>
<td>IMS</td>
<td>12×</td>
</tr>
<tr>
<td>KCS</td>
<td>10×</td>
</tr>
<tr>
<td>AVG</td>
<td>2×</td>
</tr>
</tbody>
</table>

**Energy benefit over OSP**

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$^{\dagger\dagger}$  Roknoddin Azizi$^\S$  Geraldo F. Oliveira$^\S$  Mohammad Sadrosadati$^\S$
Rakesh Nadig$^\S$  David Novo$^{\dagger}$  Juan Gómez-Luna$^\S$  Myungsuk Kim$^\ddagger$  Onur Mutlu$^\S$

$^\S$ETH Zürich  $^{\dagger}$POSTECH  $^{\dagger\dagger}$LIRMM, Univ. Montpellier, CNRS  $^\ddagger$Kyungpook National University

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
Processing inside Storage: Two Approaches

1. In-Flash Processing
2. Processing near Flash Memory
EFCL Summer School -- Track 4

Lecture 8b: Storage-Centric Computing

Mohammad Sadrosadati
Prof. Onur Mutlu

ETH Zürich
Summer 2024
6 June 2024