

Tutorial on Memory-Centric Computing: Processing-Using-Memory

Geraldo F. Oliveira

<https://geraldofojunior.github.io>

MICRO 2024

02 November 2024

Agenda

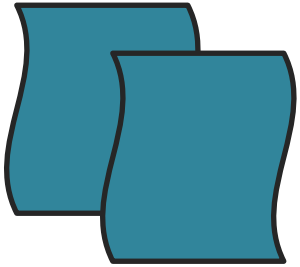
- Processing-Near-Memory Systems: Academia & Industry Developments
- **Invited Talk by Dr. Brian Schwedock:**
Architectures and Programming Models for General-Purpose Near-Data Computing
- **Processing-Using-Memory Systems for Bulk Bitwise Operations**
- Coffee Break
- Ataberk Olgun:
Infrastructure for Processing-Using-Memory Research
- **Invited Talk by Dr. Christina Giannoula:**
System Software and Libraries for Sparse Computational Kernels in PIM Architectures
- Nika Mansouri Ghiasi:
Storage-Centric Computing for Genomics and Metagenomics
- Research Challenges for PIM & Closing Remarks

Processing in Memory: Two Approaches

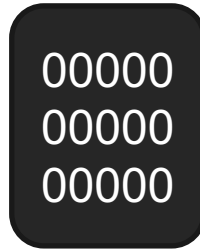
1. Processing near Memory
2. Processing **using** Memory

Starting Simple: Data Copy and Initialization

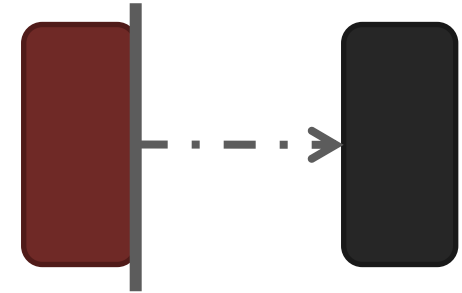
memcpy & memmove: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking



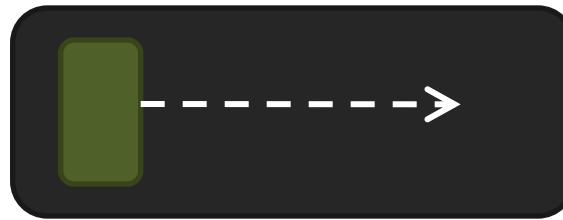
**Zero initialization
(e.g., security)**



Checkpointing



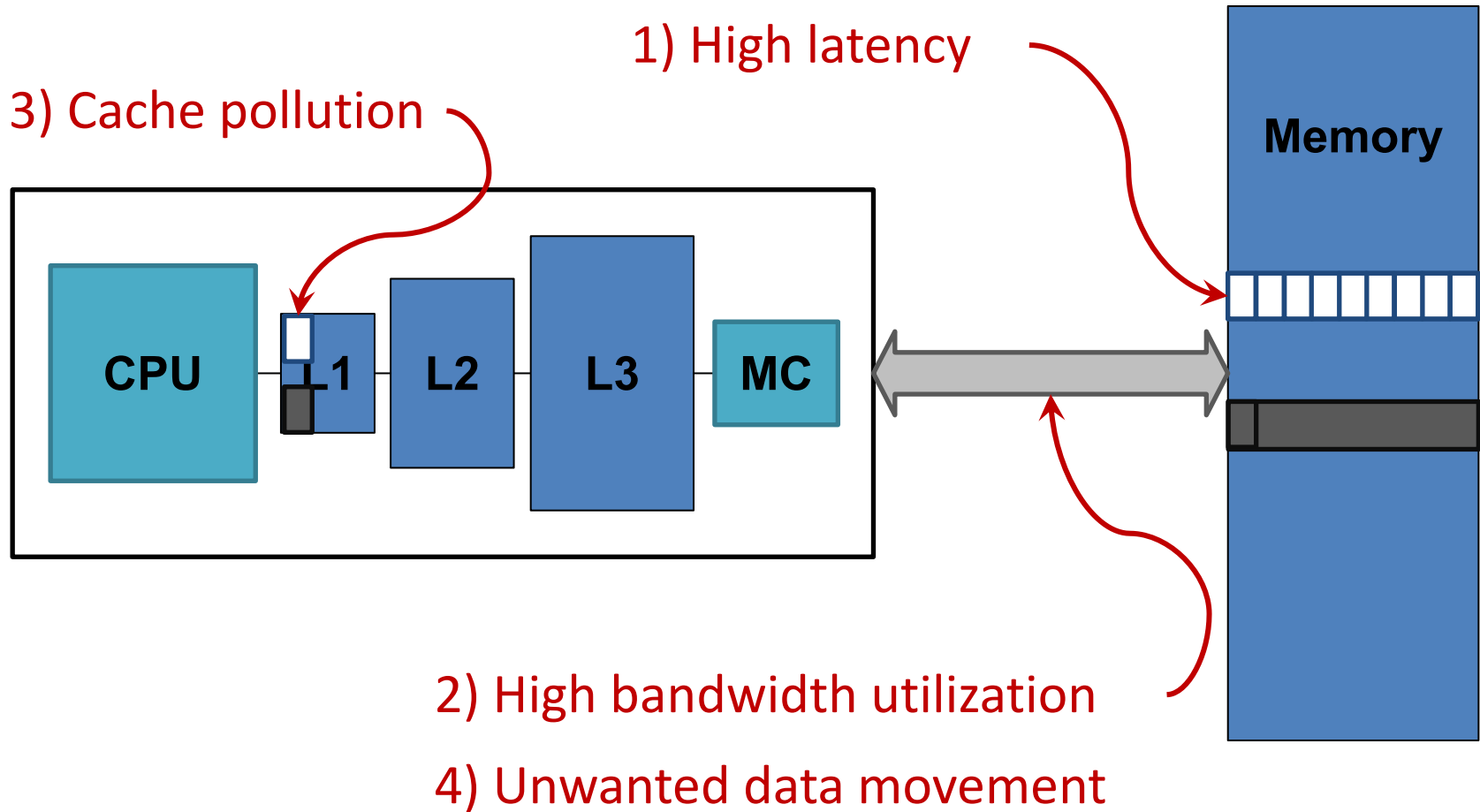
**VM Cloning
Deduplication**



Page Migration

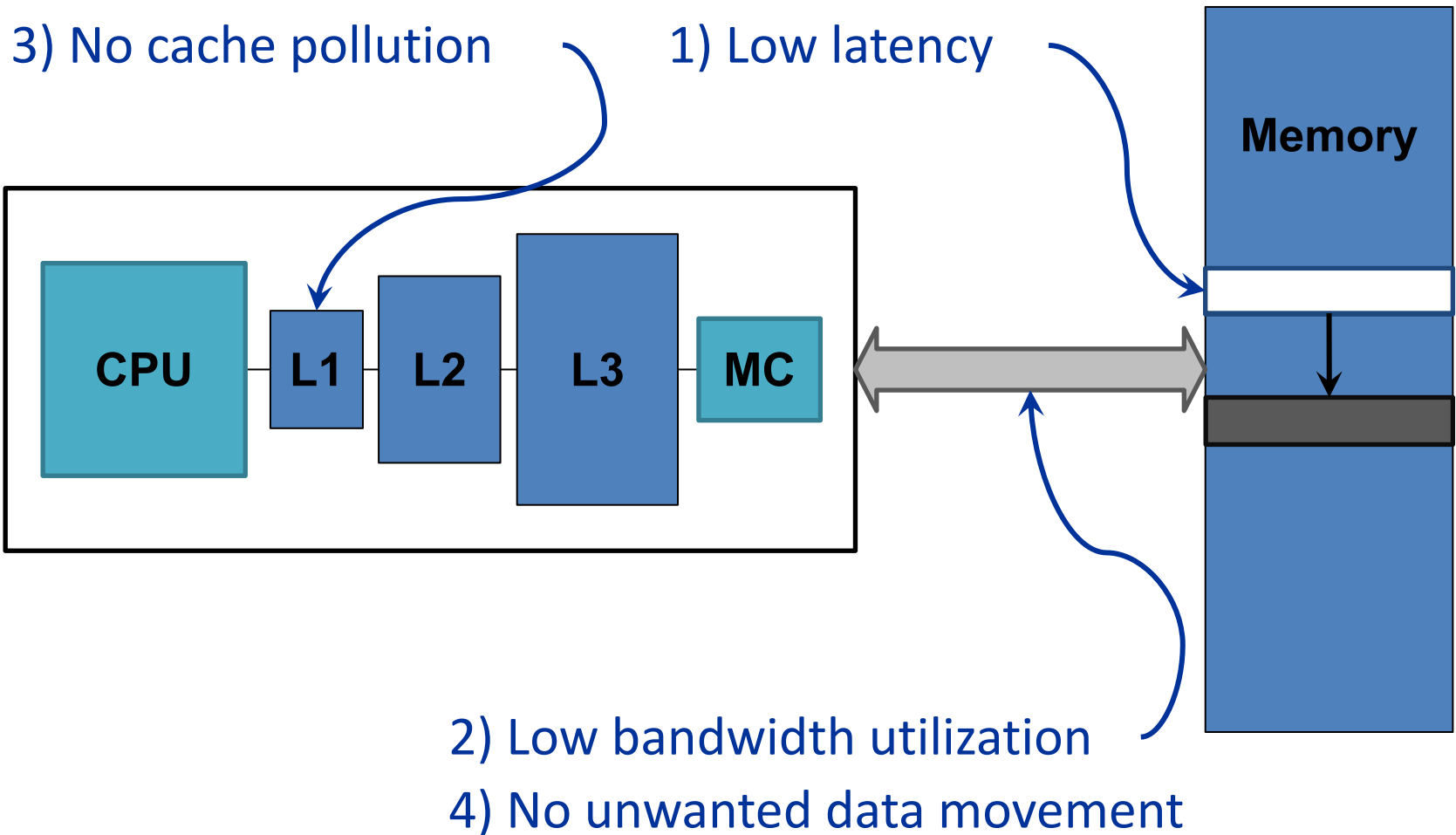
•••
Many more

Today's Systems: Bulk Data Copy



1046ns, 3.6uJ (for 4KB page copy via DMA)

Future Systems: In-Memory Copy

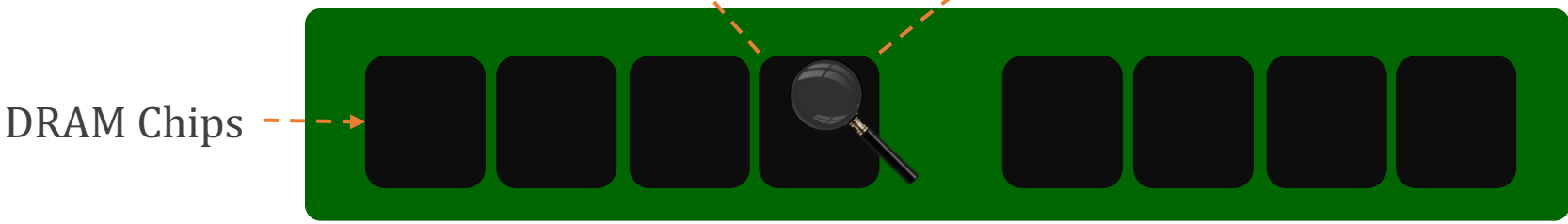
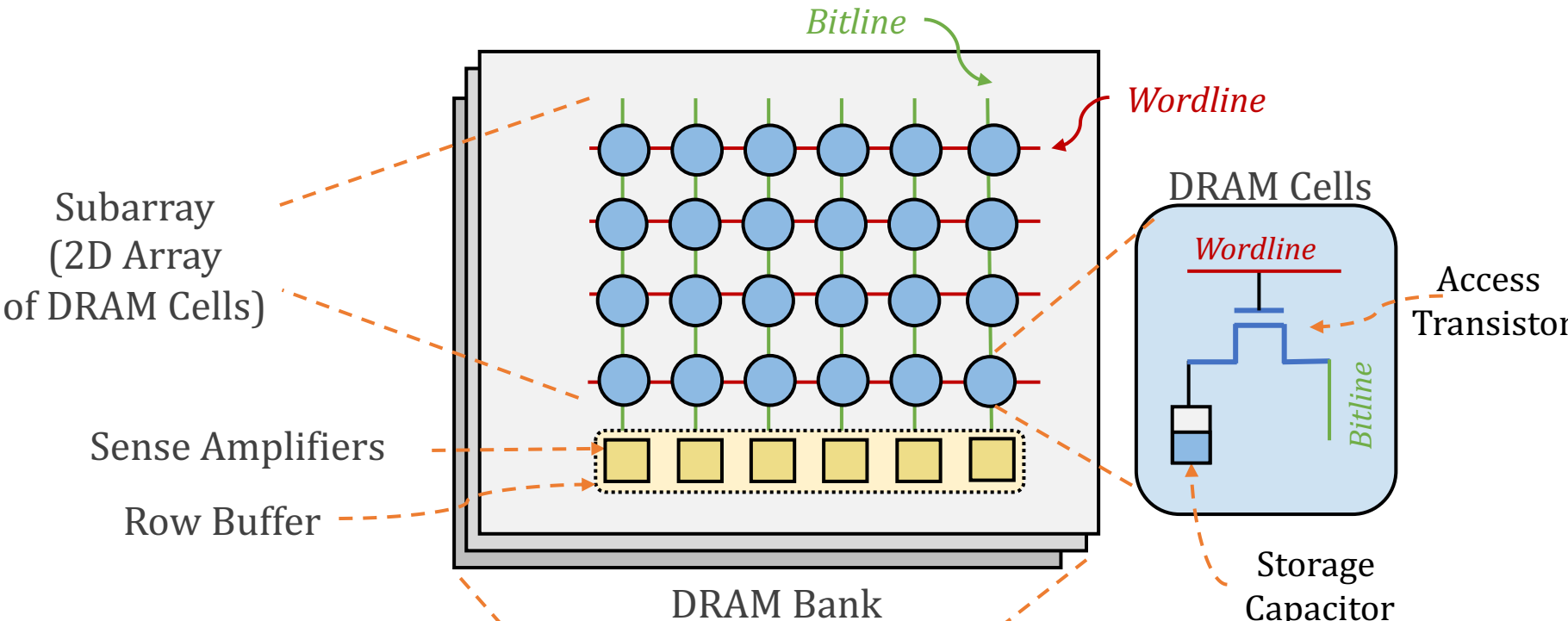


1046ns, 3.6uJ → 90ns, 0.04uJ

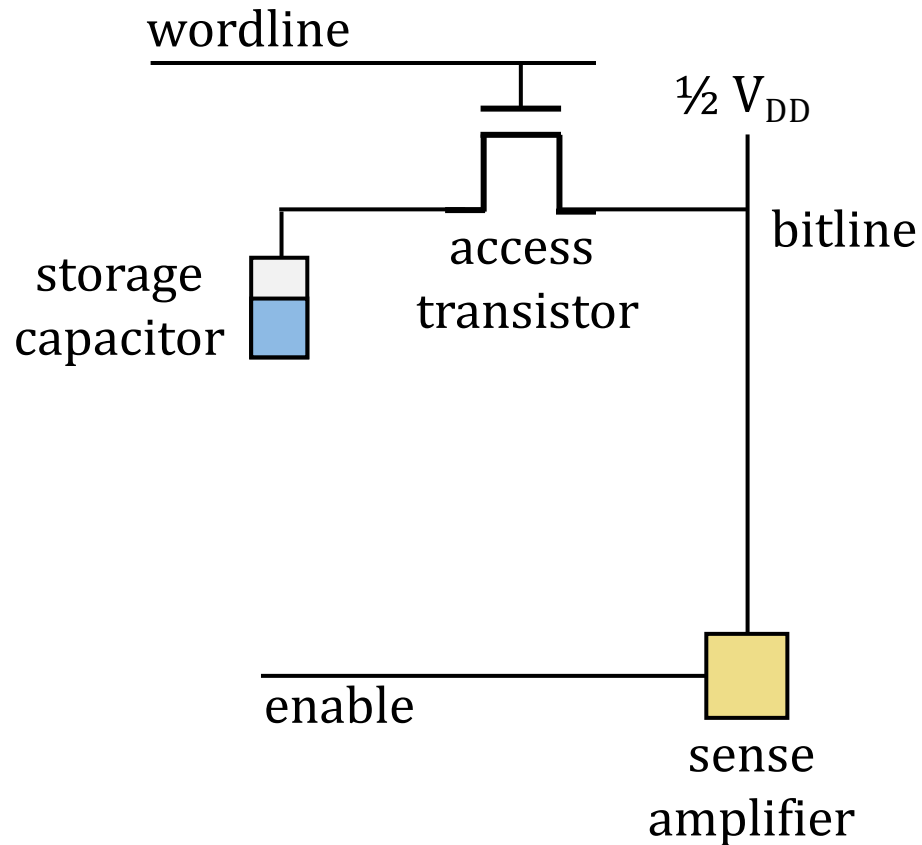
Brief Review:

Inside A DRAM Chip

Inside a DRAM Chip



DRAM Cell Operation

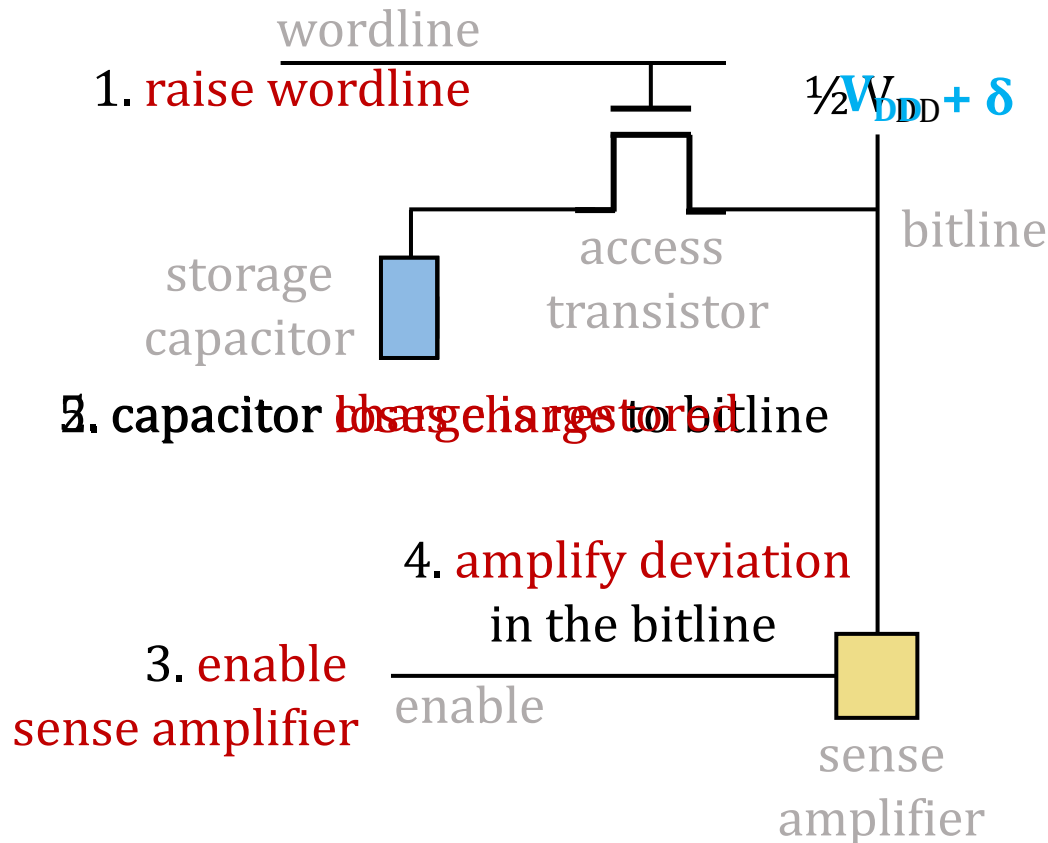


1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

DRAM Cell Operation (1/3)

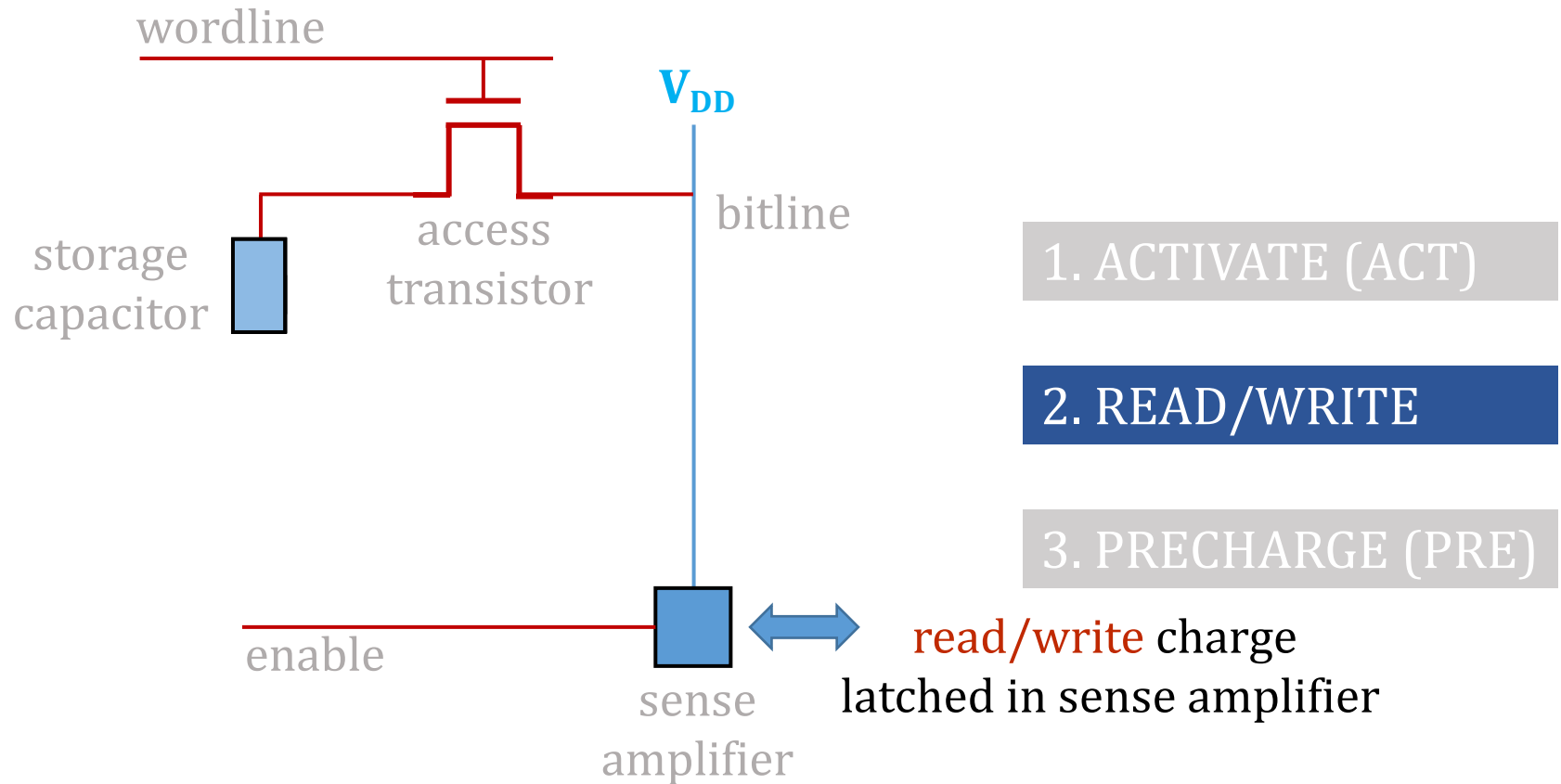


1. ACTIVATE (ACT)

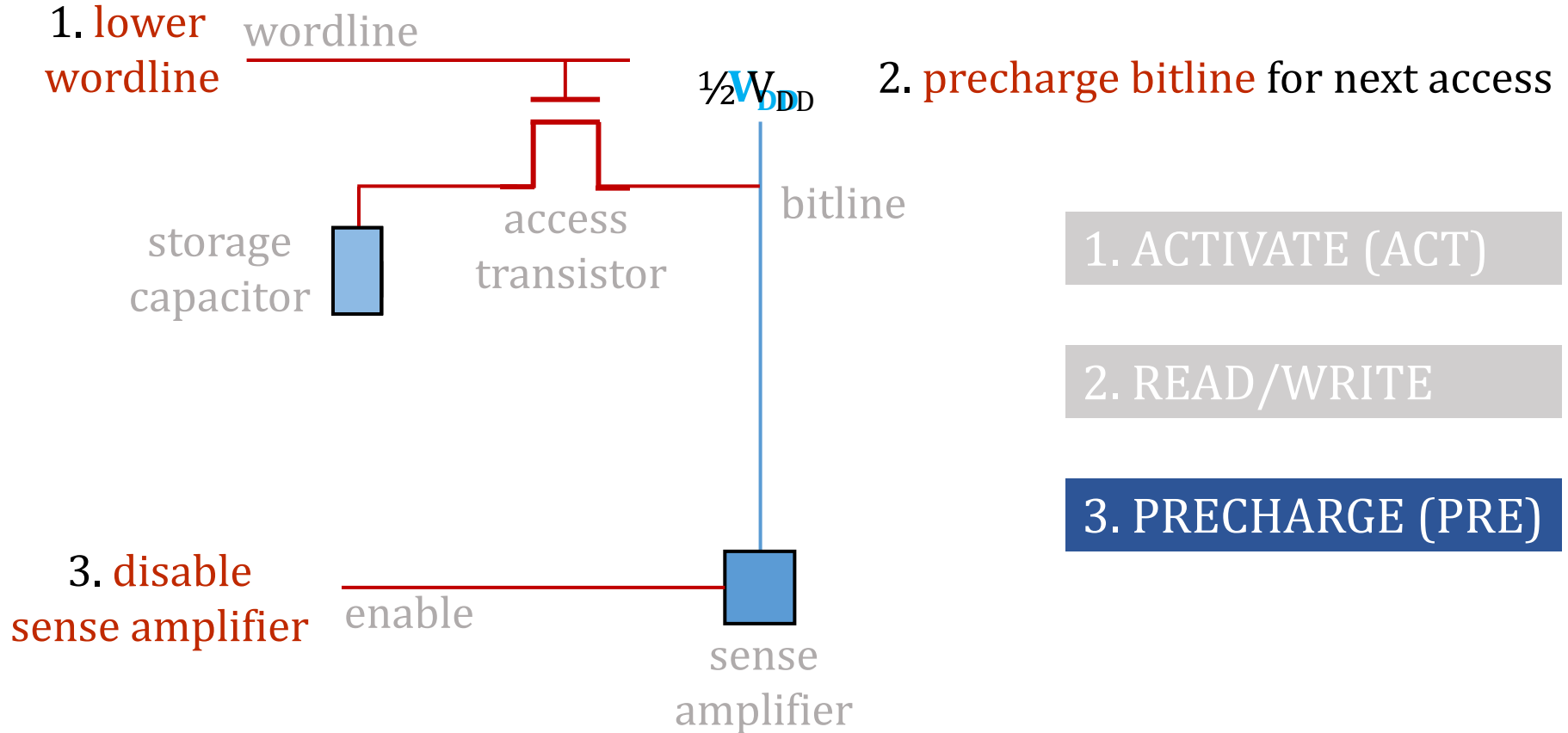
2. READ/WRITE

3. PRECHARGE (PRE)

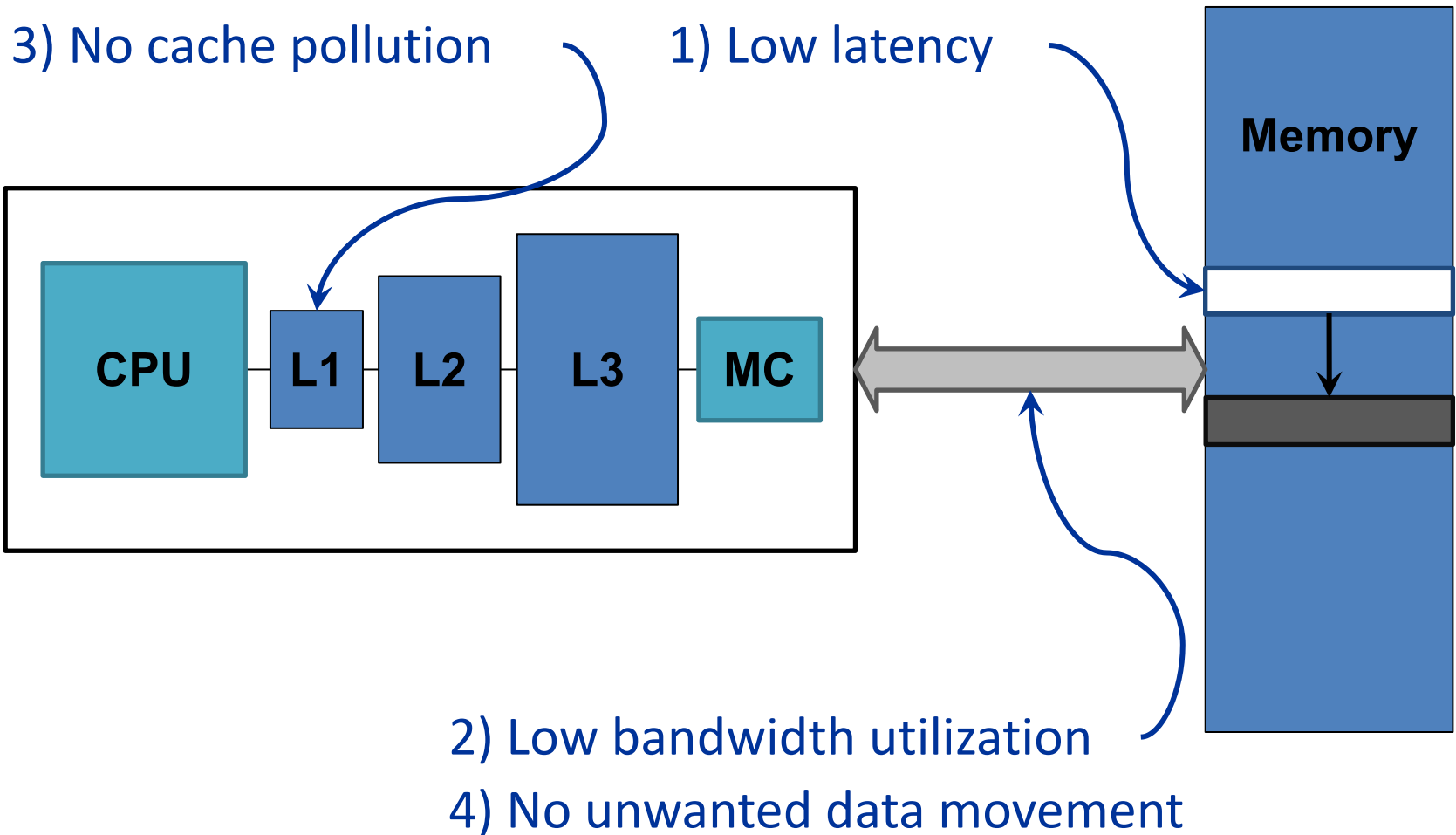
DRAM Cell Operation (2/3)



DRAM Cell Operation (3/3)



Future Systems: In-Memory Copy

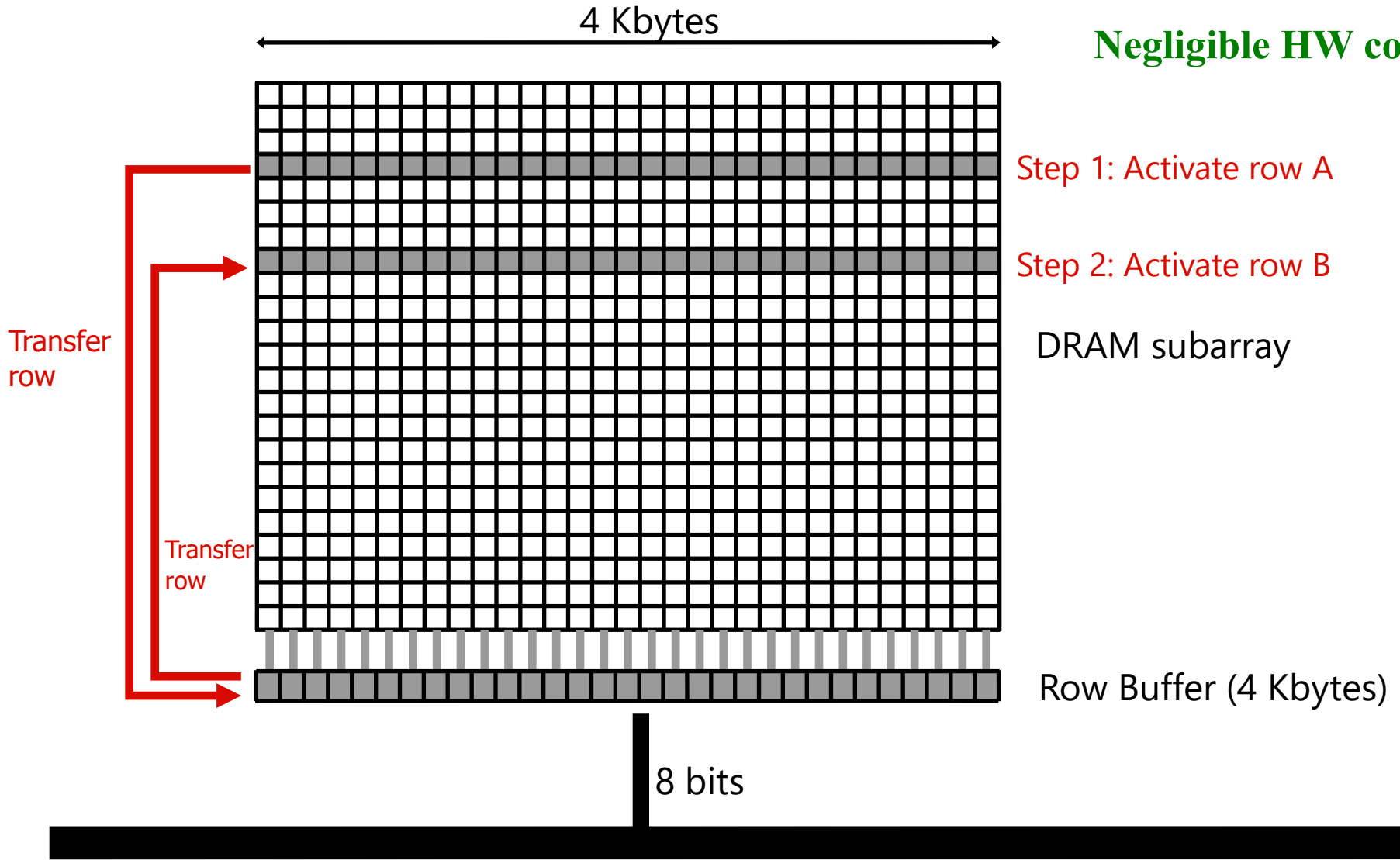


1046ns, 3.6uJ → 90ns, 0.04uJ

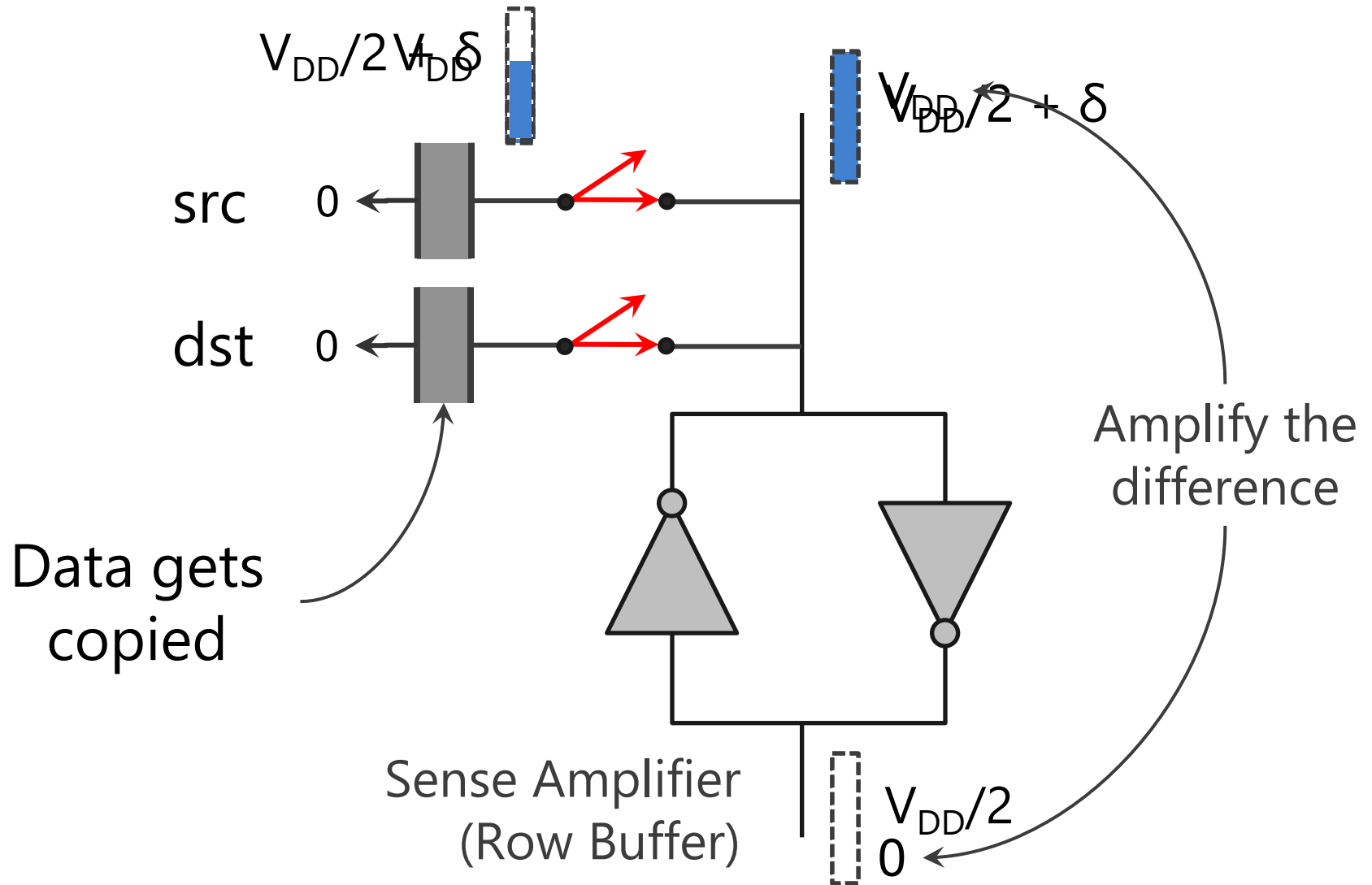
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

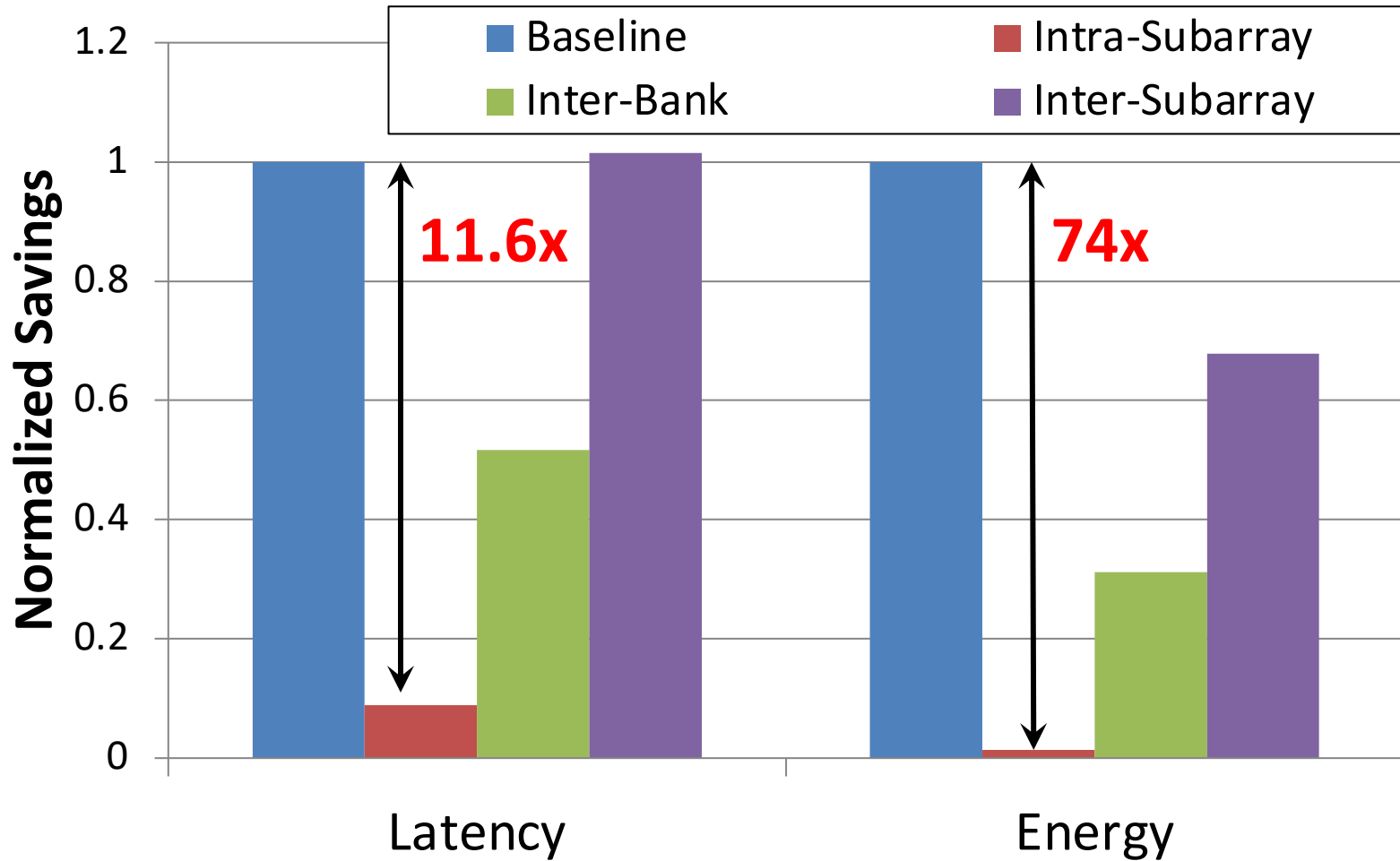
Negligible HW cost



RowClone: Intra-Subarray



RowClone: Latency and Energy Savings



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

More on RowClone

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

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

Vivek Seshadri Yoongu Kim Chris Fallin* Donghyuk Lee
vseshadr@cs.cmu.edu yoongukim@cmu.edu cfallin@c1f.net donghyuk1@cmu.edu

Rachata Ausavarungnirun Gennady Pekhimenko Yixin Luo
rachata@cmu.edu gpekhime@cs.cmu.edu yixinluo@andrew.cmu.edu

Onur Mutlu Phillip B. Gibbons† Michael A. Kozuch† Todd C. Mowry
onur@cmu.edu phillip.b.gibbons@intel.com michael.a.kozuch@intel.com tcm@cs.cmu.edu

RowClone Extensions and Follow-Up Work

- Can we do **faster inter-subarray copy**?
 - Yes, **see LISA [Chang et al., HPCA 2016]**
- Can we enable **data movement at smaller granularities within a bank**?
 - Yes, **see FIGARO [Wang et al., MICRO 2020]**
- Can we do **better inter-bank copy**?
 - Yes, **see Network-on-Memory [CAL 2020]**
- Can similar ideas and DRAM properties be used to perform **computation on data**?
 - Yes, **see Ambit [Seshadri et al., CAL 2015, MICRO 2017]**

LISA: Increasing Connectivity in DRAM

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,

"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"

Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA), Barcelona, Spain, March 2016.

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

[[Source Code](#)]

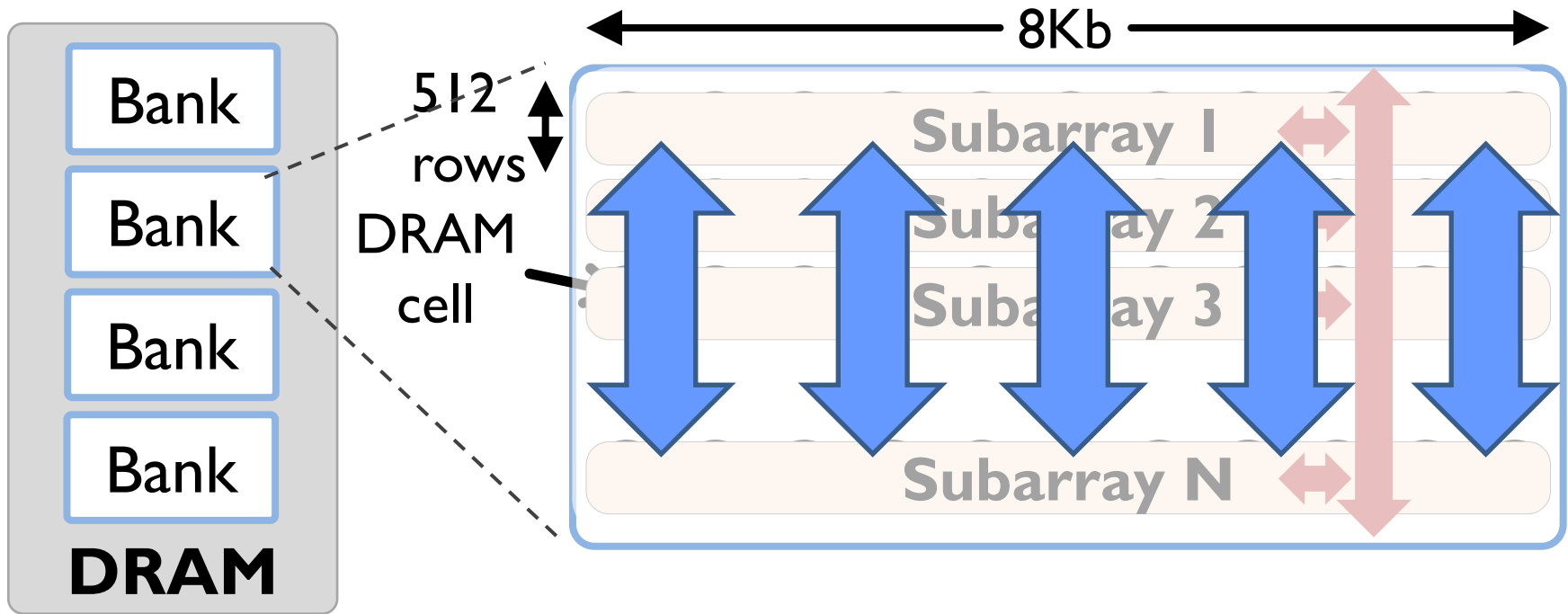
Low-Cost Inter-Linked Subarrays (LISA):

Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang[†], Prashant J. Nair^{*}, Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi^{*}, and Onur Mutlu[†]

[†]*Carnegie Mellon University* ^{*}*Georgia Institute of Technology*

Moving Data Inside DRAM?

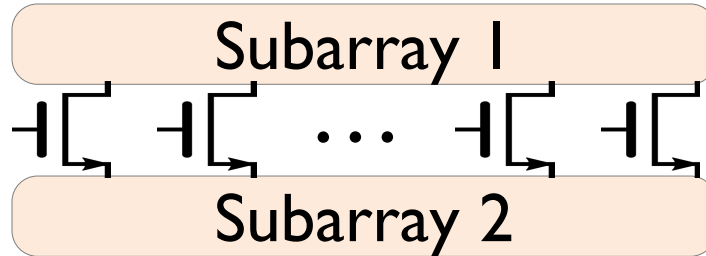


Goal: Provide a new substrate to enable wide connectivity between subarrays

Key Idea and Applications

- **Low-cost Inter-linked subarrays (LISA)**

- Fast bulk data movement between subarrays
- **Wide datapath via isolation transistors:** 0.8% DRAM chip area



- LISA is a **versatile substrate** → new applications
 - Fast bulk data copy:** Copy latency 1.363ms→0.148ms (**9.2x**)
→ 66% speedup, -55% DRAM energy
 - In-DRAM caching:** Hot data access latency 48.7ns→21.5ns (**2.2x**)
→ 5% speedup
 - Fast precharge:** Precharge latency 13.1ns→5.0ns (**2.6x**)
→ 8% speedup

More on LISA

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,

"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"

Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA), Barcelona, Spain, March 2016.

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

[[Source Code](#)]

Low-Cost Inter-Linked Subarrays (LISA):

Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang[†], Prashant J. Nair^{*}, Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi^{*}, and Onur Mutlu[†]

[†]*Carnegie Mellon University* ^{*}*Georgia Institute of Technology*

FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu,
"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"
Proceedings of the 53rd International Symposium on Microarchitecture (MICRO), Virtual, October 2020.

FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang^{*} Lois Orosa[†] Xiangjun Peng^{⊙*} Yang Guo^{*} Saugata Ghose^{◇‡} Minesh Patel[†]
Jeremie S. Kim[†] Juan Gómez Luna[†] Mohammad Sadrosadati[§] Nika Mansouri Ghiasi[†] Onur Mutlu^{†‡}

^{*}National University of Defense Technology [†]ETH Zürich [⊙]Chinese University of Hong Kong

[◇]University of Illinois at Urbana–Champaign [‡]Carnegie Mellon University [§]Institute of Research in Fundamental Sciences

Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,
["NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"](#)
[IEEE Computer Architecture Letters](#) (**CAL**), to appear in 2020.

NOM: NETWORK-ON-MEMORY FOR INTER-BANK DATA TRANSFER IN HIGHLY-BANKED MEMORIES

Seyyed Hossein SeyyedAghaei Rezaei¹
Mohammad Sadrosadati³

Mehdi Modarressi^{1,3}
Onur Mutlu⁴

Rachata Ausavarungnirun²
Masoud Daneshtalab⁵

¹University of Tehran

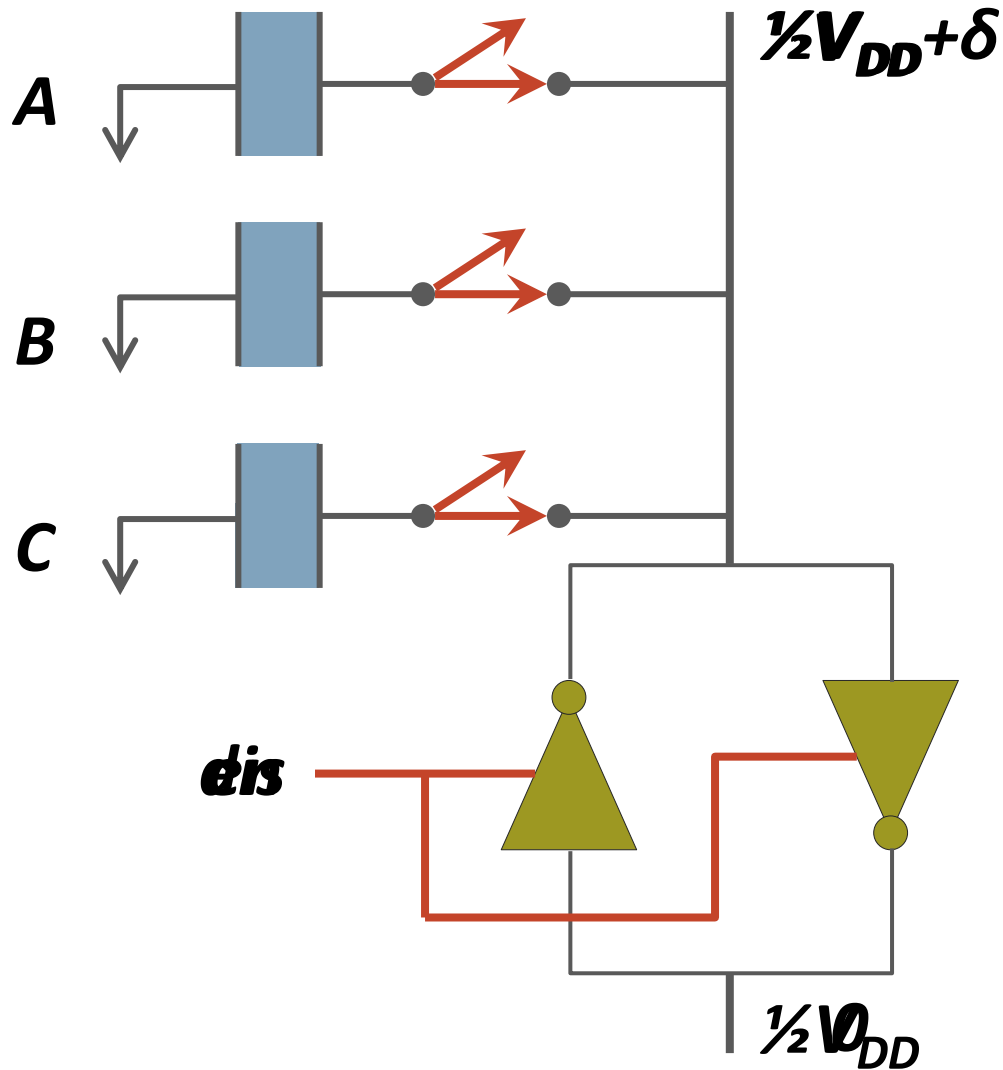
²King Mongkut's University of Technology North Bangkok

³Institute for Research in Fundamental Sciences

⁴ETH Zürich

⁵Mälardalens University

In-DRAM AND/OR: Triple Row Activation



Final State
 $AB + BC + AC$

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

In-DRAM NOT: Dual Contact Cell

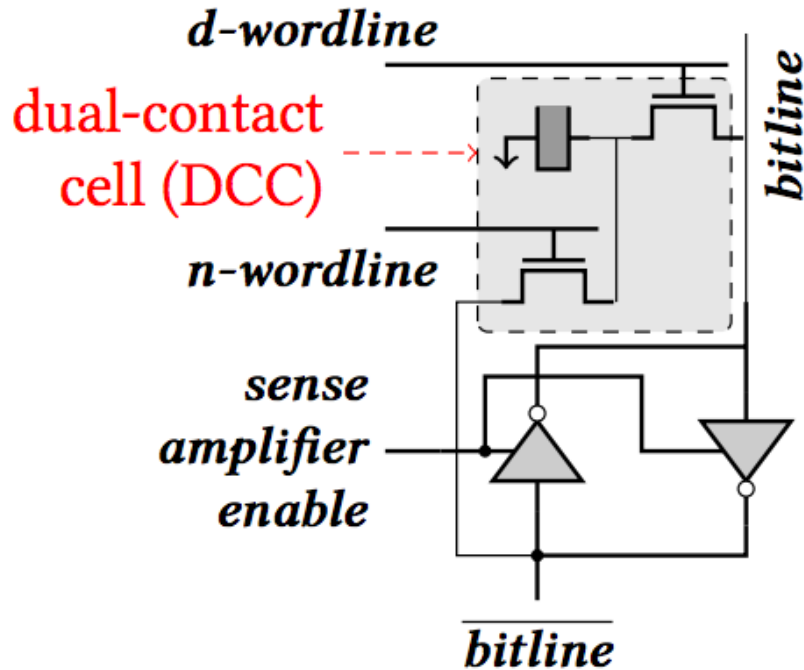


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

Performance: In-DRAM Bitwise Operations

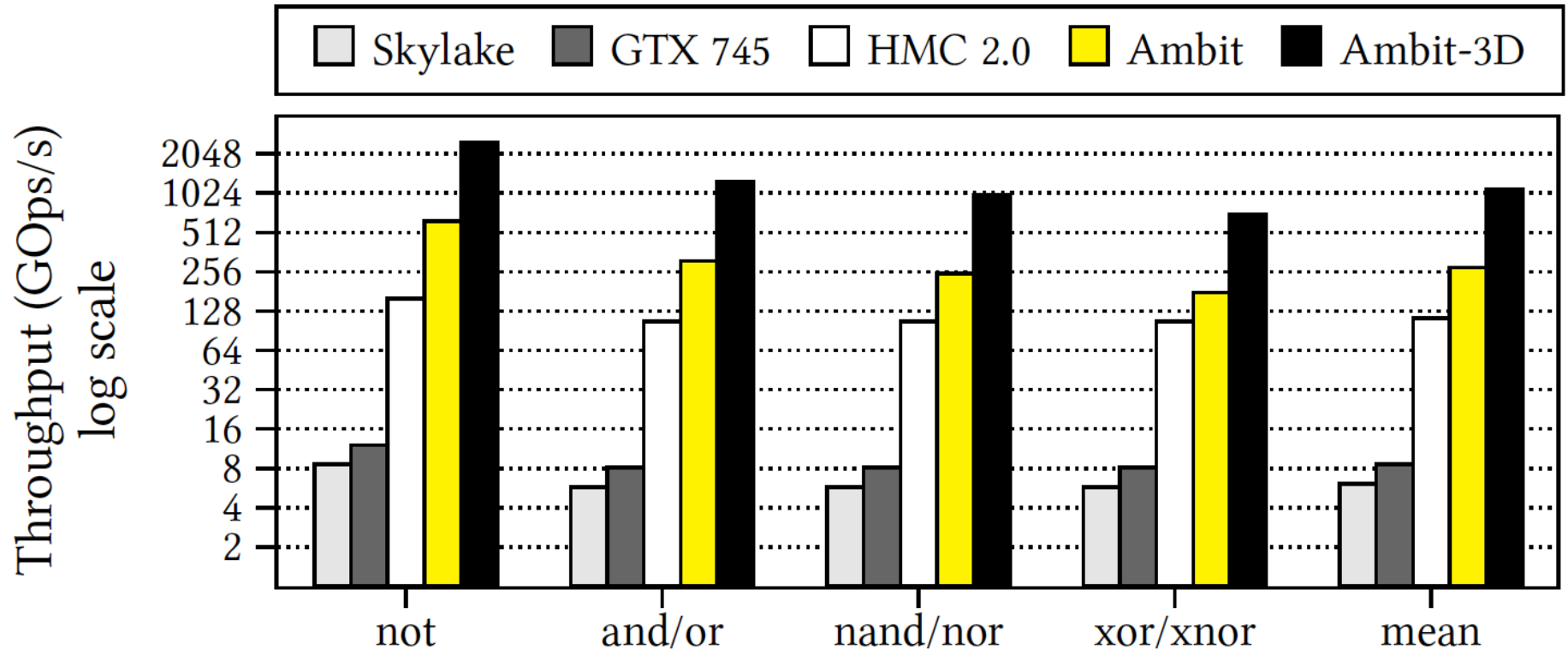


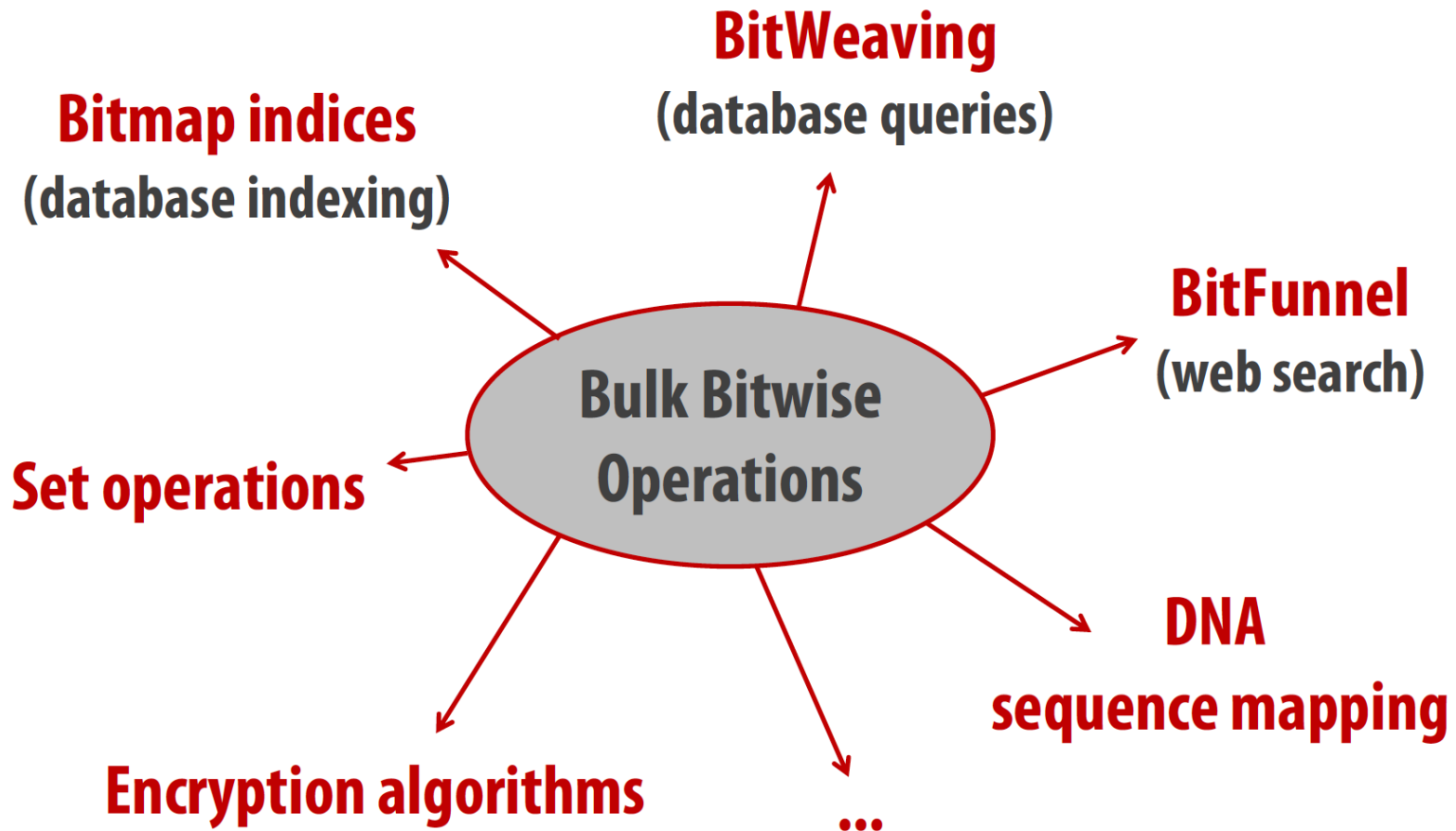
Figure 9: Throughput of bitwise operations on various systems.

Energy of In-DRAM Bitwise Operations

	Design	not	and/or	nand/nor	xor/xnor
DRAM &	DDR3	93.7	137.9	137.9	137.9
Channel Energy	Ambit	1.6	3.2	4.0	5.5
(nJ/KB)	(↓)	59.5X	43.9X	35.1X	25.1X

Table 3: Energy of bitwise operations. (↓) indicates energy reduction of Ambit over the traditional DDR3-based design.

Bulk Bitwise Operations in Workloads



In-DRAM Acceleration of Database Queries

`'select count(*) from T where c1 <= val <= c2'`

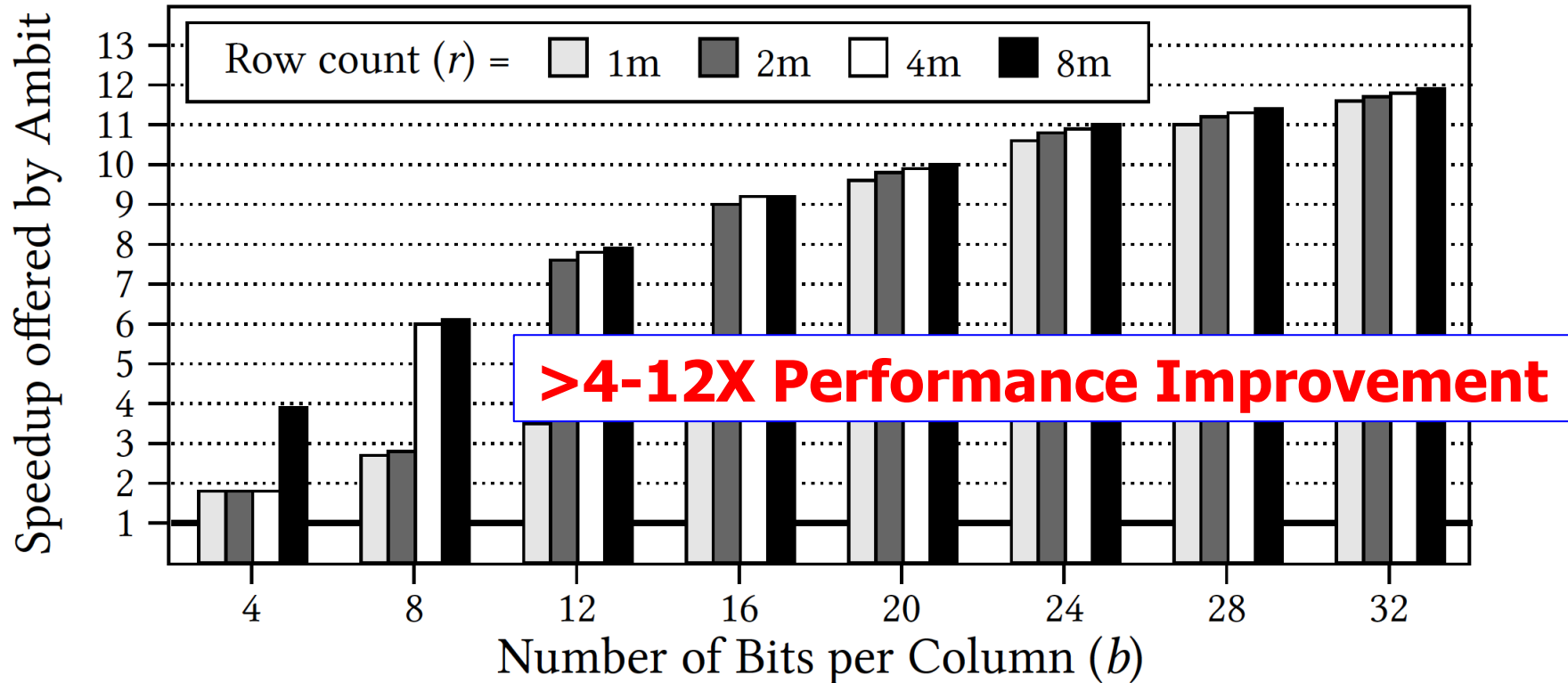


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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

More on Ambit

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

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

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

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

In-DRAM Bulk Bitwise Execution

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

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

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

SIMDRAM Framework

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

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

*Nastaran Hajinazar^{1,2}

Nika Mansouri Ghiasi¹

*Geraldo F. Oliveira¹

Minesh Patel¹

Juan Gómez-Luna¹

Sven Gregorio¹

Mohammed Alser¹

Onur Mutlu¹

João Dinis Ferreira¹

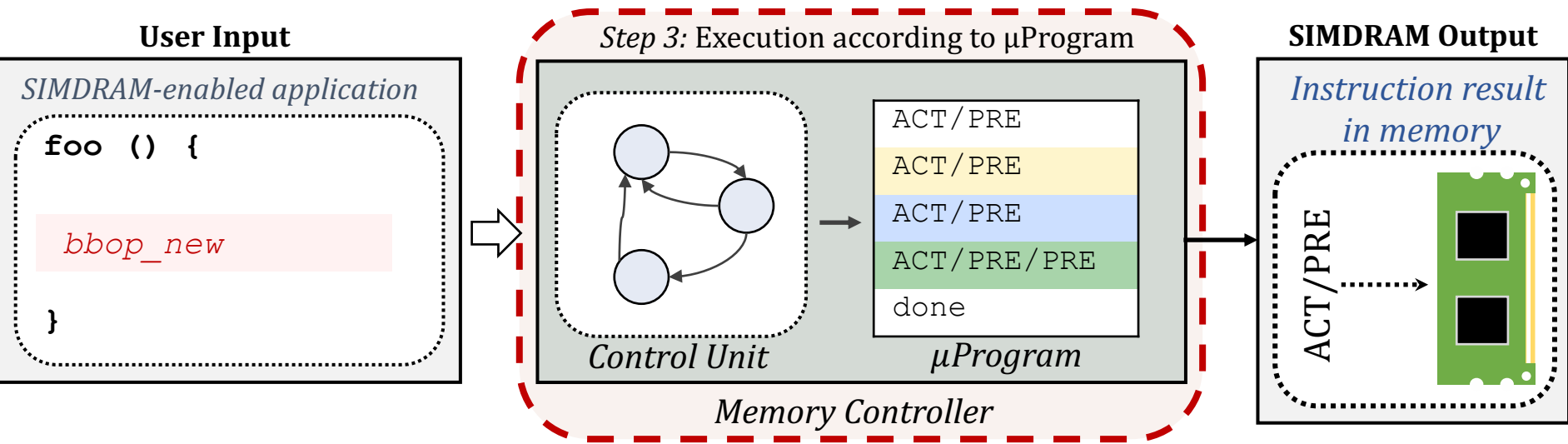
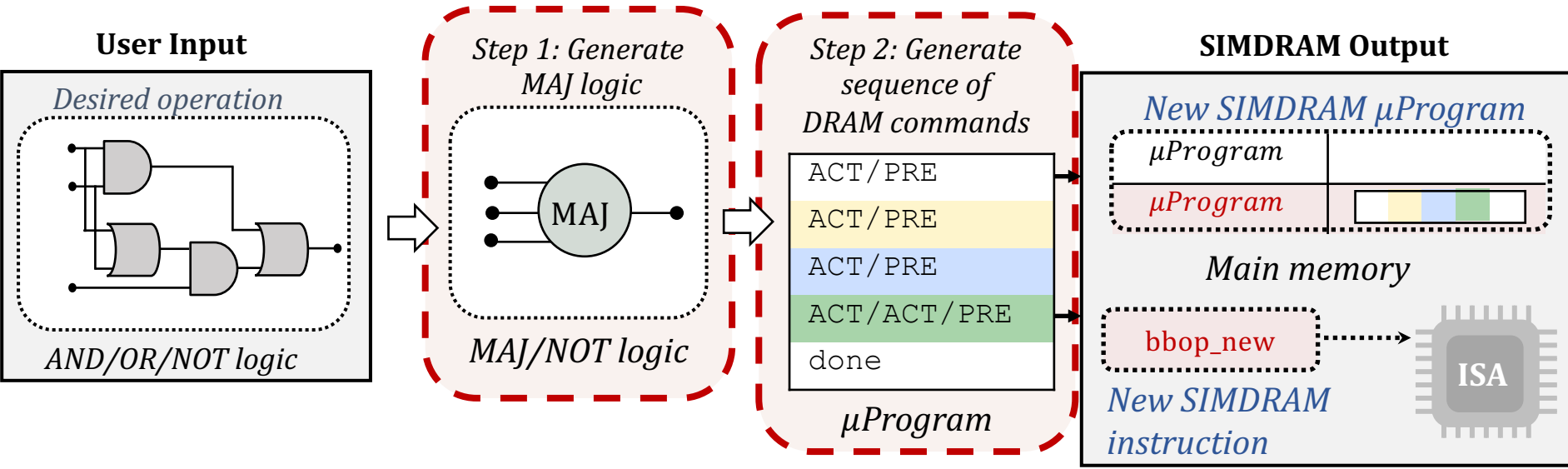
Saugata Ghose³

¹ETH Zürich

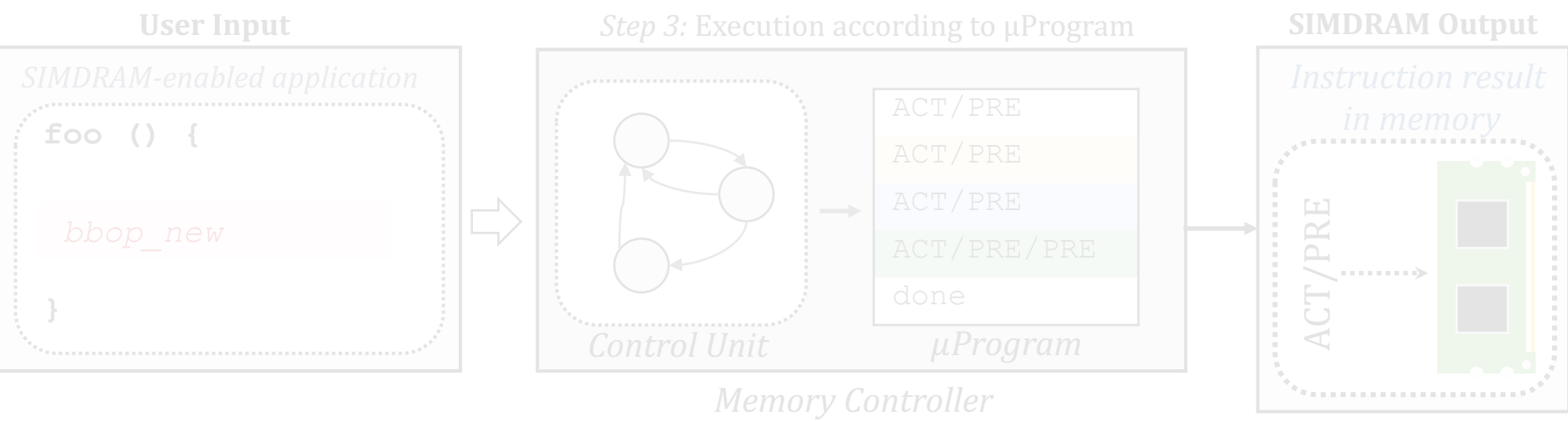
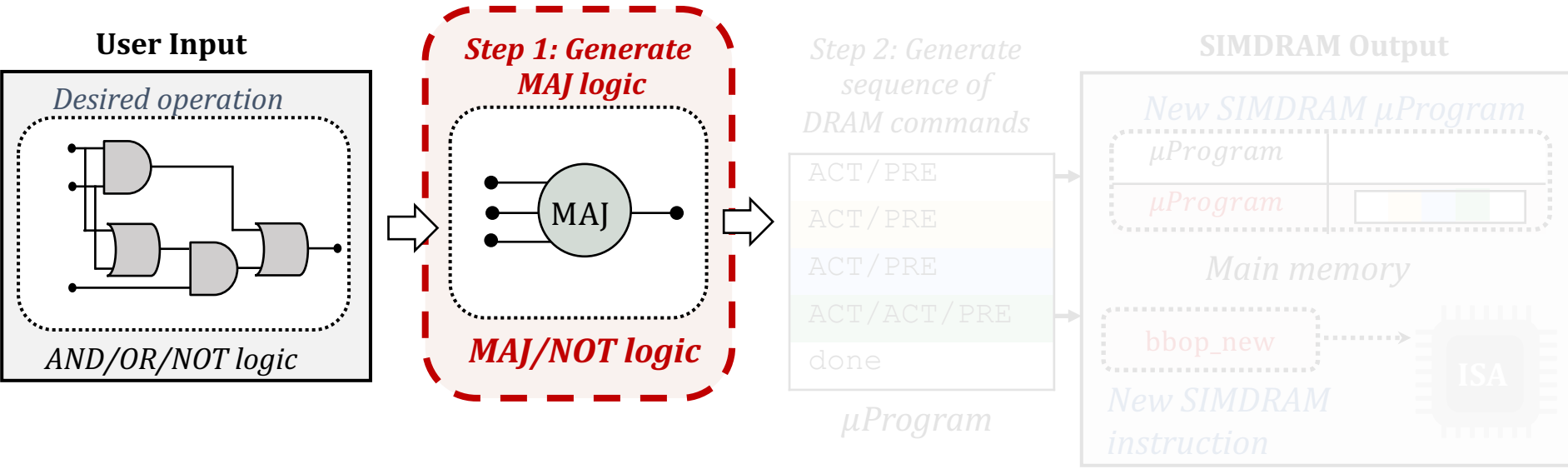
²Simon Fraser University

³University of Illinois at Urbana–Champaign

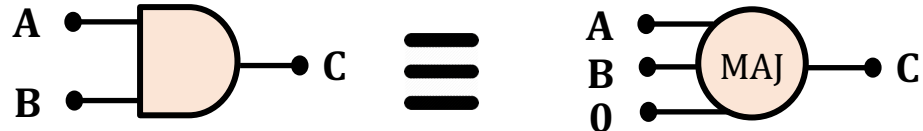
SIMDRAM Framework: Overview



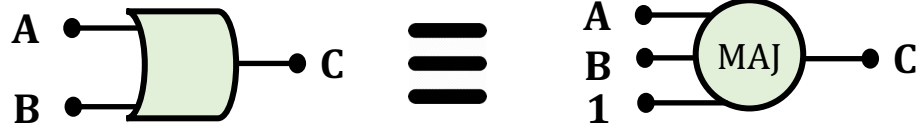
SIMDRAM Framework: Step 1



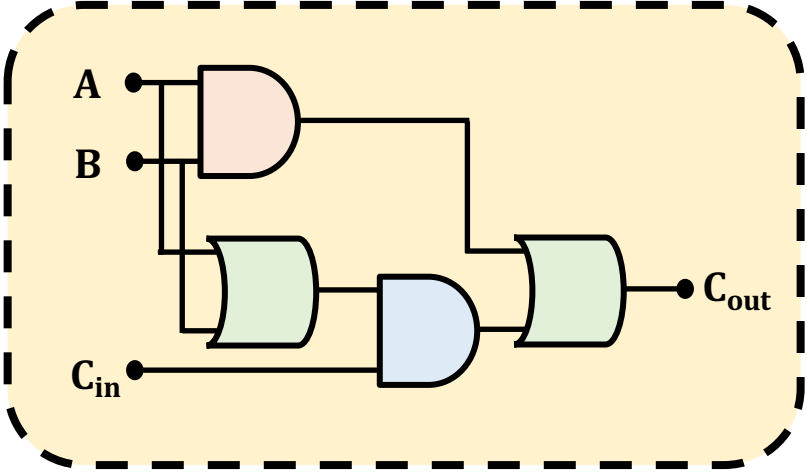
Step 1: Naïve MAJ/NOT Implementation



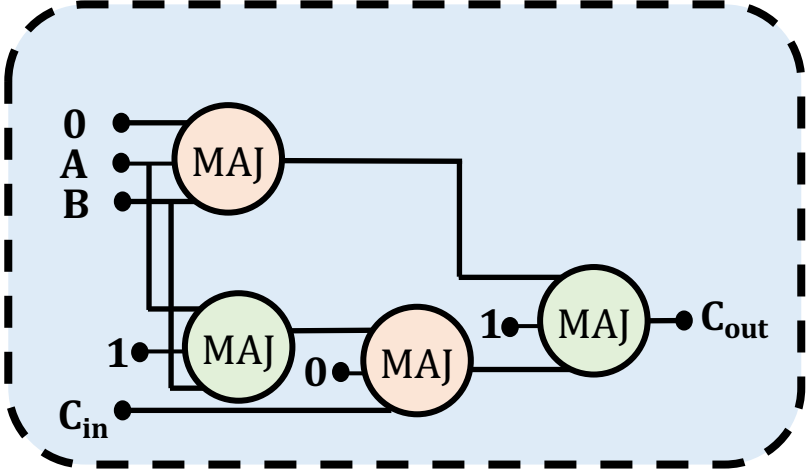
output is "1" only when $A = B = "1"$



output is "0" only when $A = B = "0"$

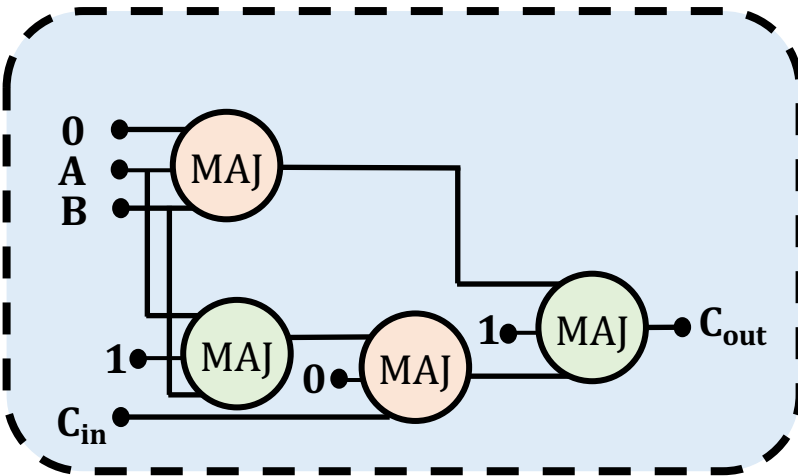


Part 1

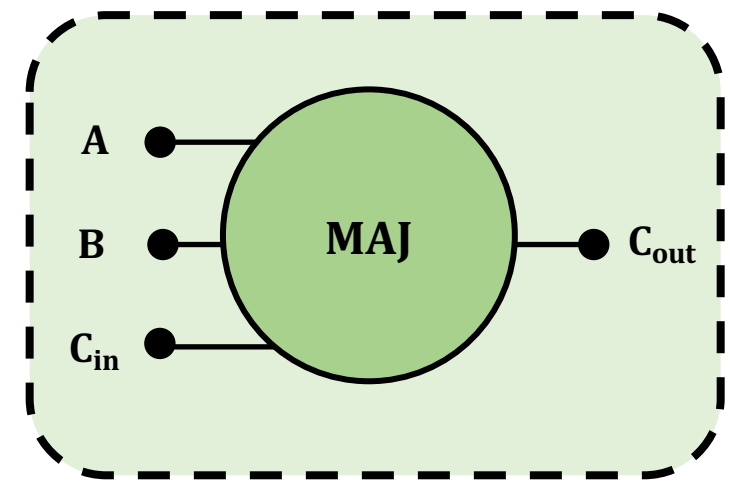
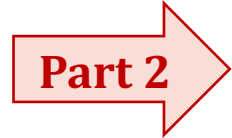


Naïvely converting AND/OR/NOT-implementation to MAJ/NOT-implementation leads to an **unoptimized circuit**

Step 1: Efficient MAJ/NOT Implementation



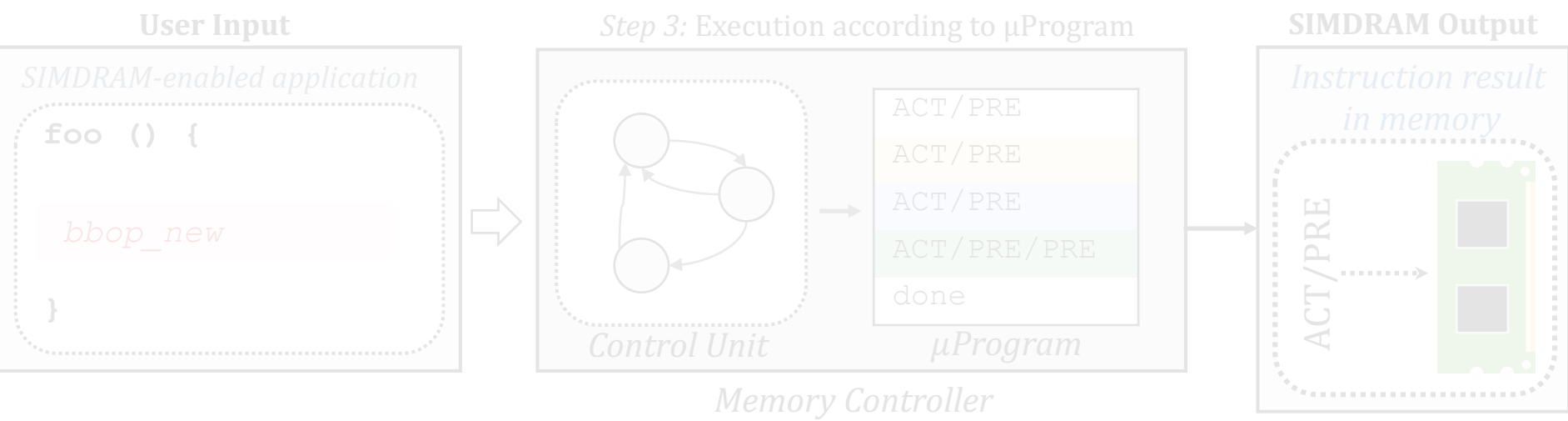
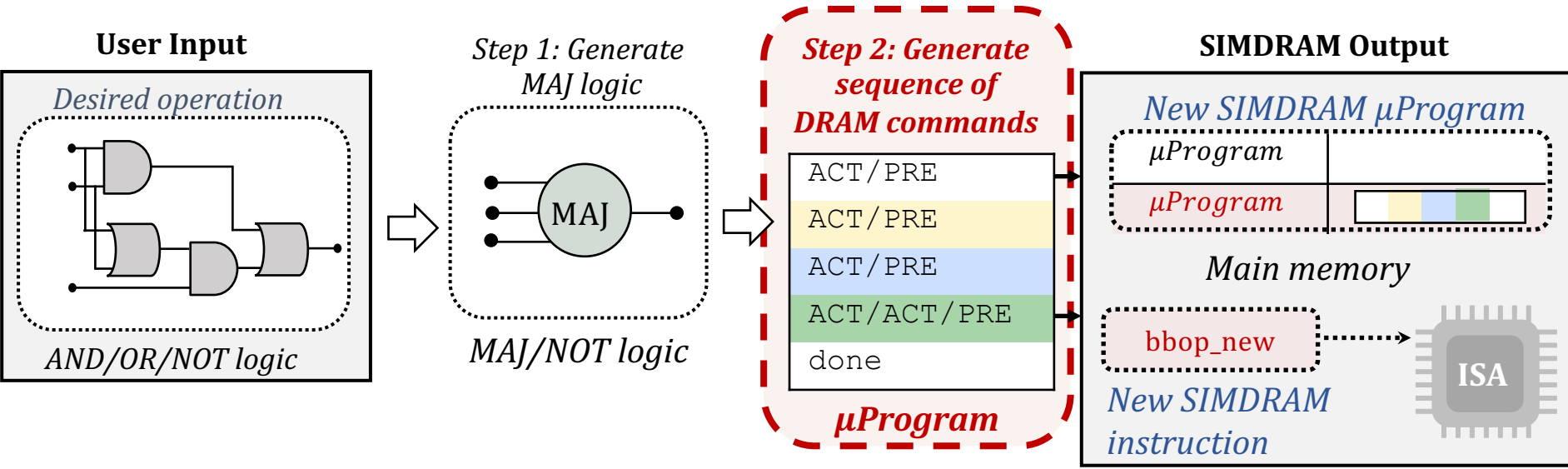
Greedy optimization algorithm⁴



Step 1 generates an **optimized MAJ/NOT-implementation** of the desired operation

⁴ L. Amarù et al, "Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization", DAC, 2014.

SIMDRAM Framework: Step 2



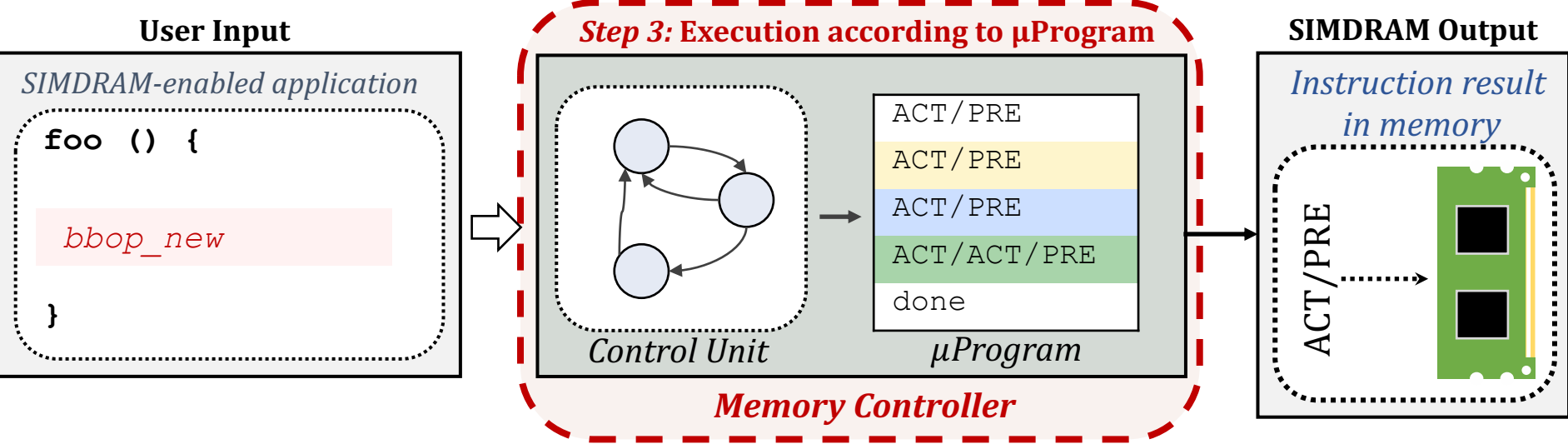
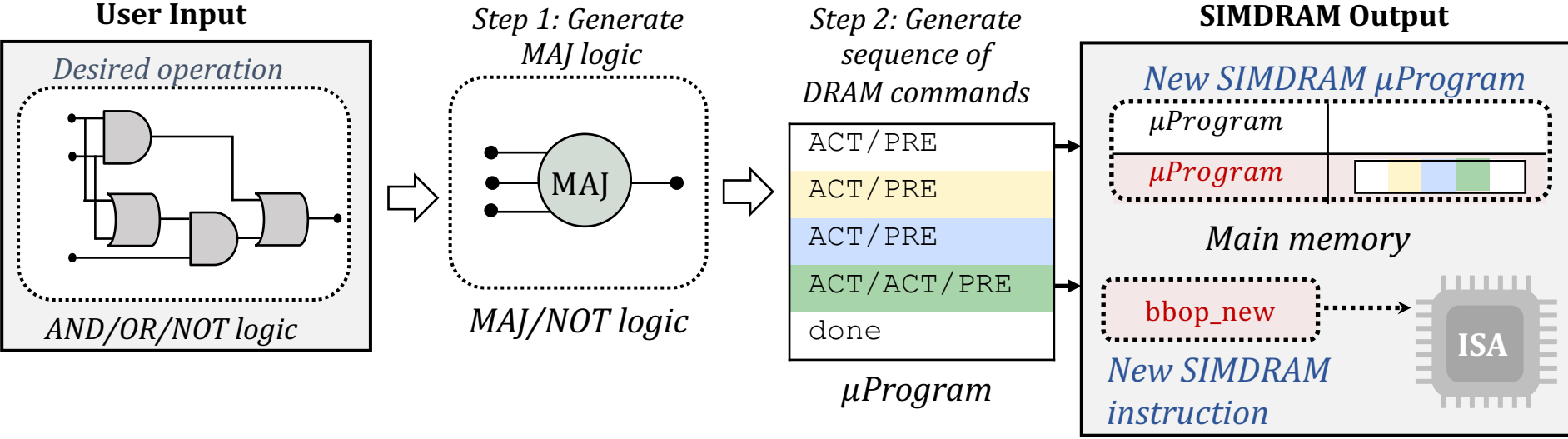
Step 2: μ Program Generation

- **μ Program:** A series of **microarchitectural operations** (e.g., ACT/PRE) that SIMD RAM uses to execute **SIMDRAM operation in DRAM**
- **Goal of Step 2:** To generate the **μ Program** that **executes** the desired SIMD RAM operation **in DRAM**

Task 1: Allocate DRAM rows to the operands

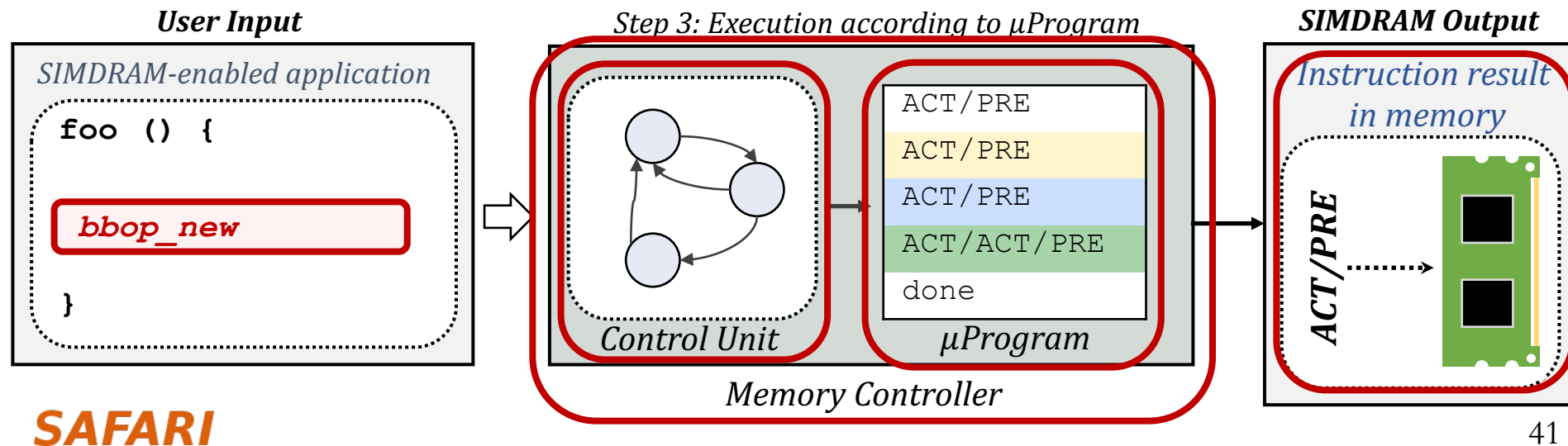
Task 2: Generate μ Program

SIMDRAM Framework: Step 3



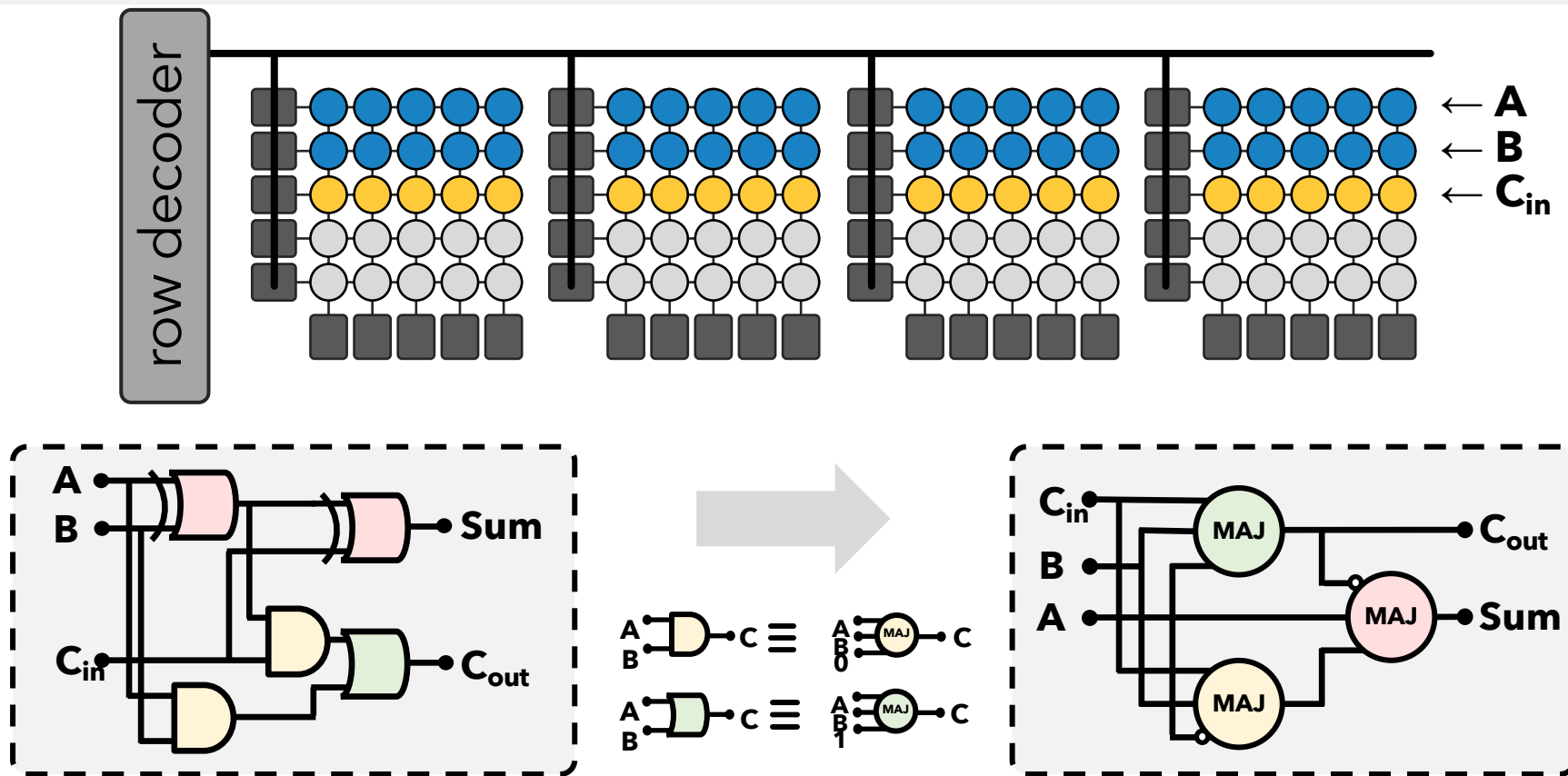
Step 3: μ Program Execution

- **SIMDRAM control unit:** handles the execution of the μ Program at runtime
- Upon receiving a **bbop instruction**, the control unit:
 1. Loads the μ Program corresponding to SIMDRAM operation
 2. Issues the sequence of DRAM commands (ACT/PRE) stored in the μ Program to SIMDRAM subarrays to perform the in-DRAM operation



Example: Bulk Bitwise Arithmetic Operations

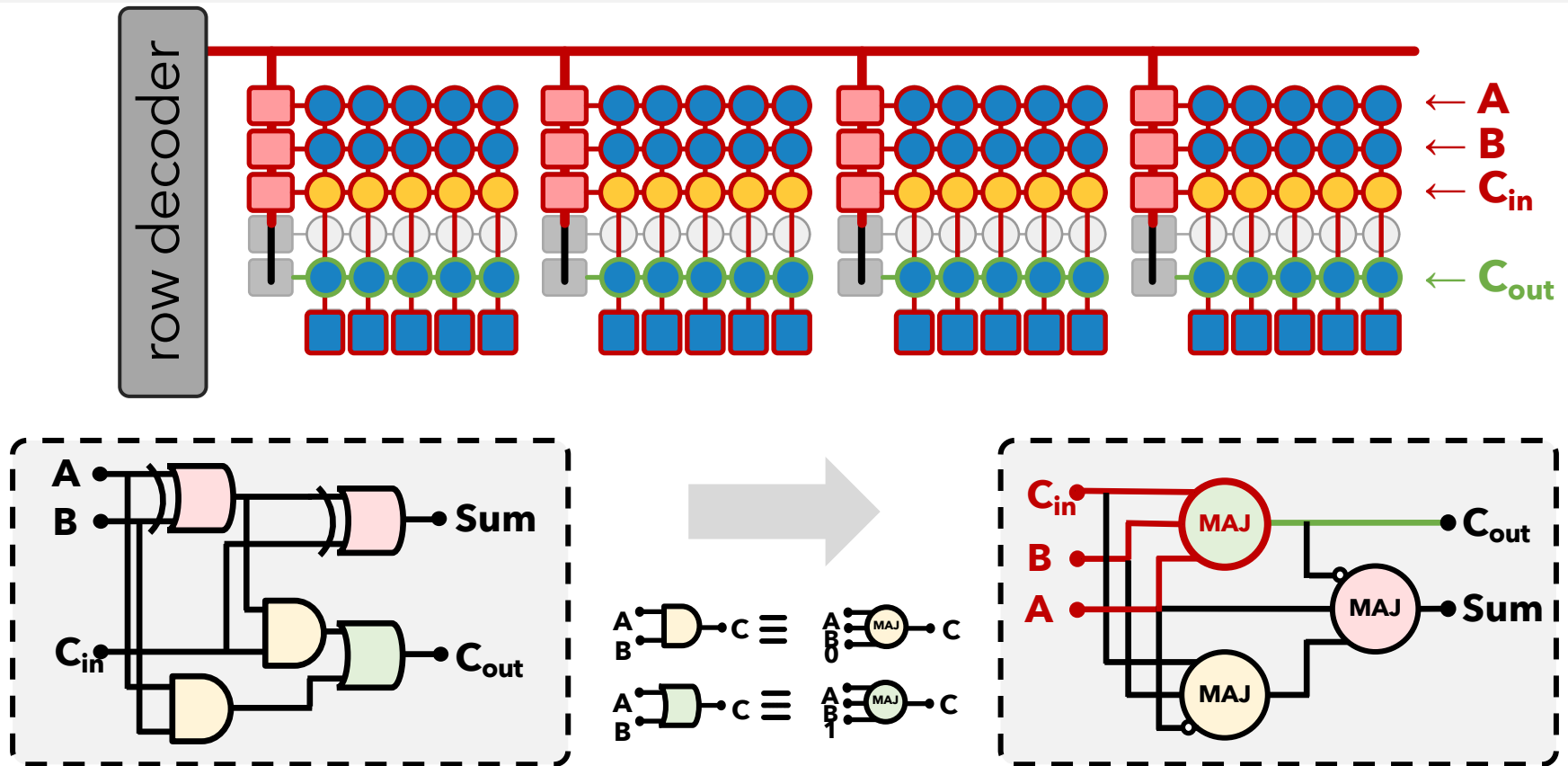
Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations



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

Example: Bulk Bitwise Arithmetic Operations

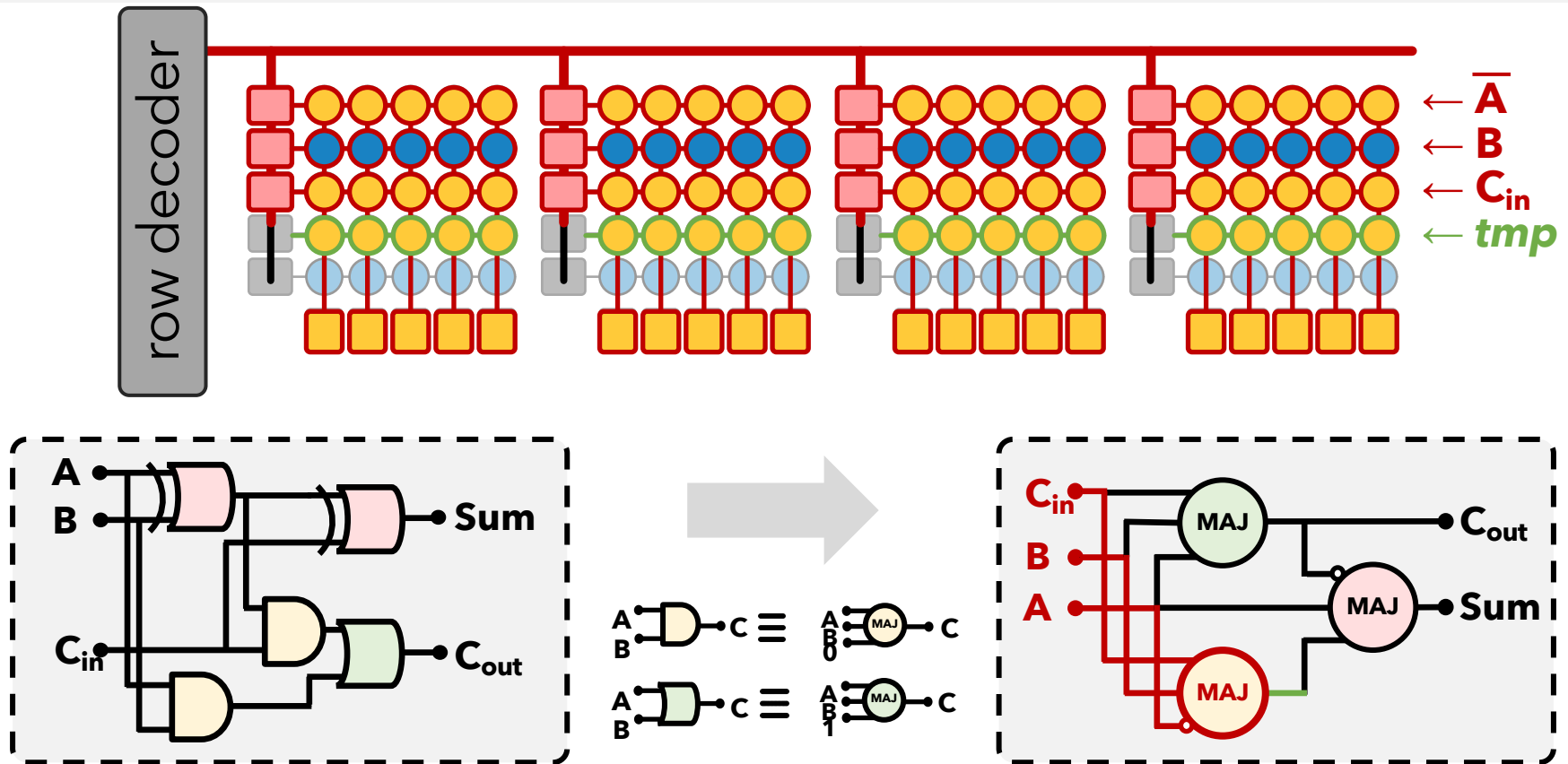
Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations



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

Example: Bulk Bitwise Arithmetic Operations

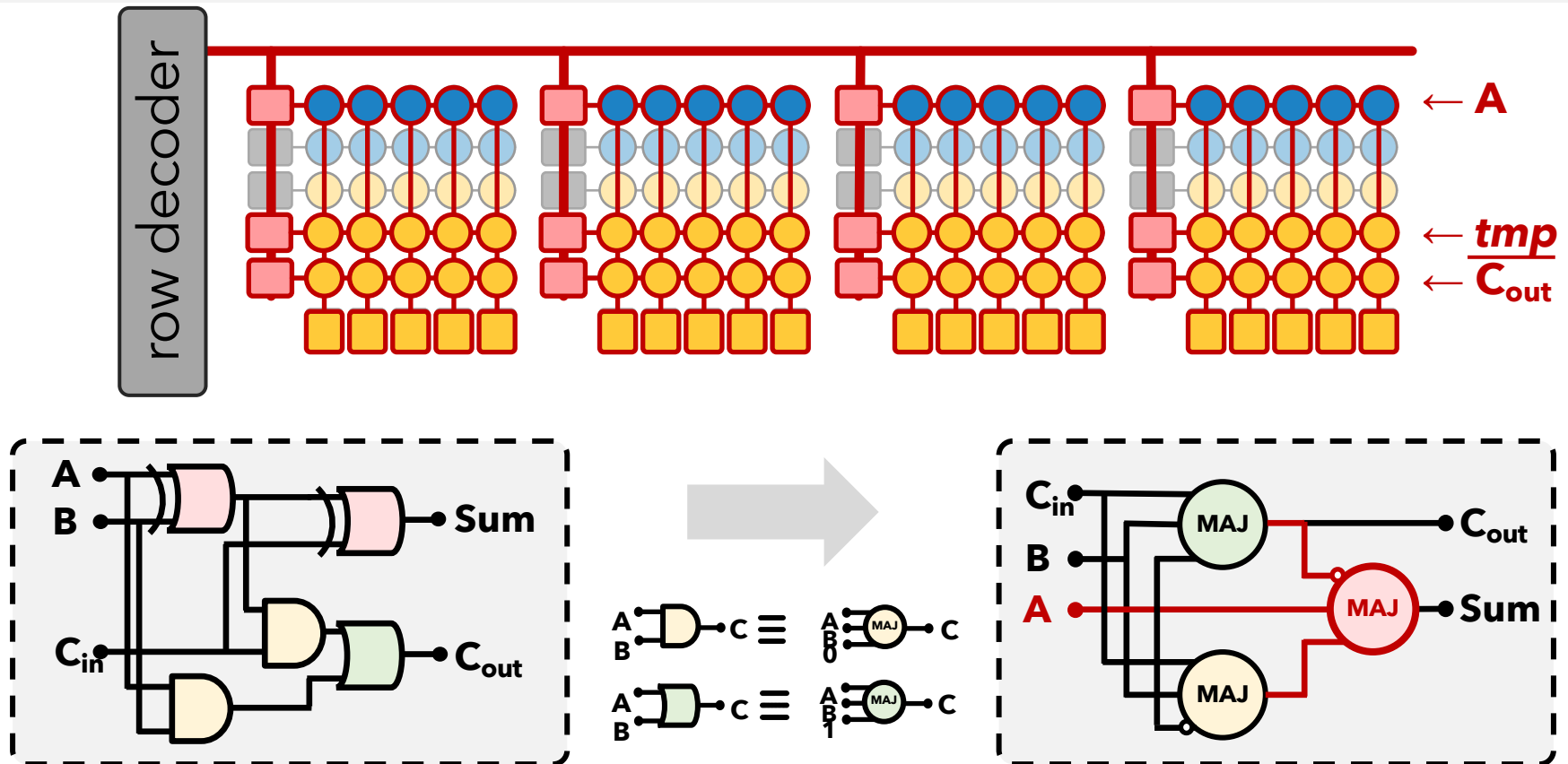
Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations



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

Example: Bulk Bitwise Arithmetic Operations

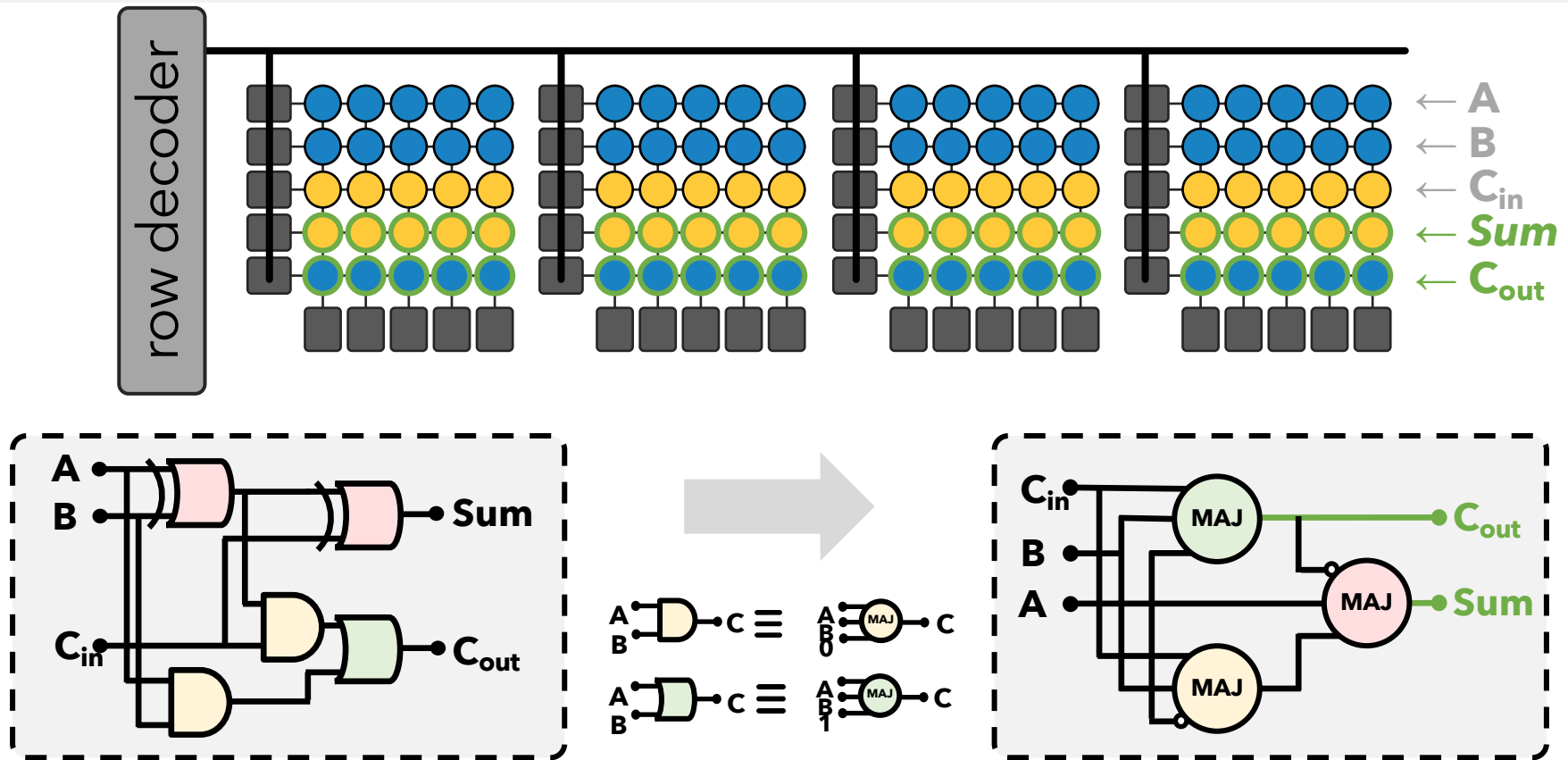
Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations



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

Example: Bulk Bitwise Arithmetic Operations

Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations

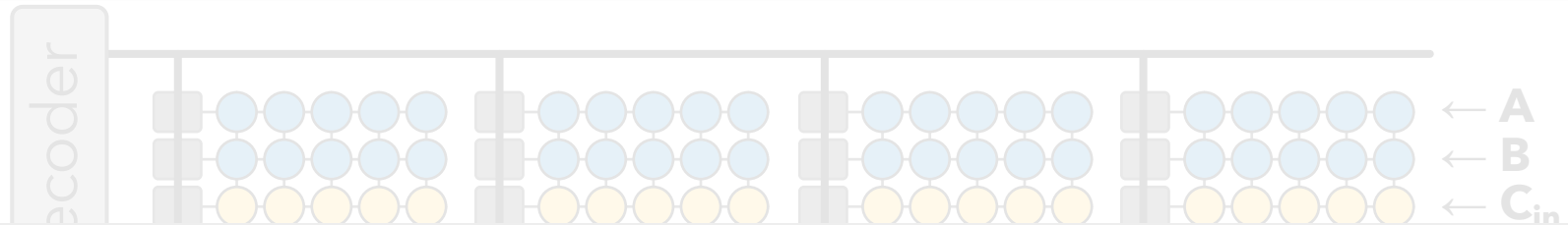


Oliveira, Geraldo F., et al. "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM," in ASPLOS, 2021

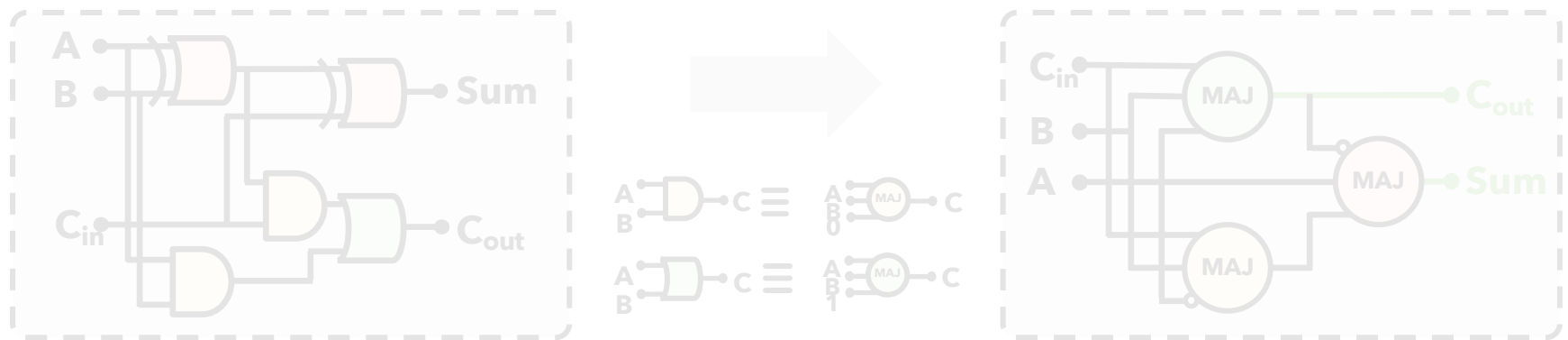
Example:

Bulk Bitwise Arithmetic Operations

Bulk bitwise arithmetic can be performed by orchestrating in-DRAM row copy and majority operations



Processing-Using-DRAM architectures (e.g., SIMDDRAM) are **very-wide** (e.g., 65,536 wide) bit-serial **SIMD** engines



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

More in the Paper

https://people.inf.ethz.ch/omutlu/pub/SIMDRAM_asplos21.pdf

SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM

*Nastaran Hajinazar^{1,2} *Geraldo F. Oliveira¹ Sven Gregorio¹ João Dinis Ferreira¹
Nika Mansouri Ghiasi¹ Minesh Patel¹ Mohammed Alser¹ Saugata Ghose³
Juan Gómez-Luna¹ Onur Mutlu¹

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana-Champaign

coherence, and interrupts

Handling limited subarray size

Security implications

Limitations of our framework

SIMDRAM Key Results

Evaluated on:

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

SIMDRAM provides:

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

More on SIMD RAM

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

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

*Nastaran Hajinazar^{1,2}

Nika Mansouri Ghiasi¹

*Geraldo F. Oliveira¹

Minesh Patel¹

Juan Gómez-Luna¹

Sven Gregorio¹

Mohammed Alser¹

Onur Mutlu¹

João Dinis Ferreira¹

Saugata Ghose³

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

SIMDRAM: Follow-Ups

- **Limitations of current substrate?**
 - Computing granularity
 - Data layout conversion
 - High-latency bit-serial operations
 - Assembly-like programming model
 - Application scope
 - ...

- We are working on **even better processing-using-memory substrates**
 - One step at a time!

Limitations of PUD Systems: Overview

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

1 SIMD Underutilization

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

2 Limited Computation Support

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

3 Challenging Programming Model

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

Problem & Goal

Problem

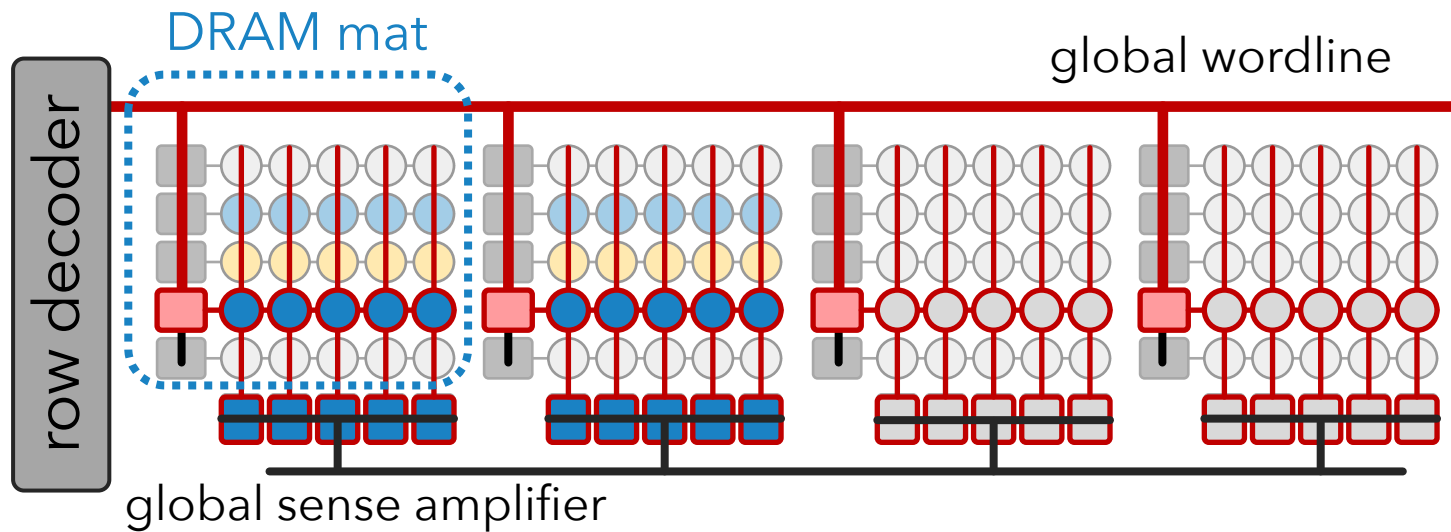
Processing-Using-DRAM's large and rigid granularity limits its applicability and efficiency for different applications

Goal

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

MIMDRAM: Key Idea (I)

**DRAM's hierarchical organization can enable
fine-grained access**



Key Issue:

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



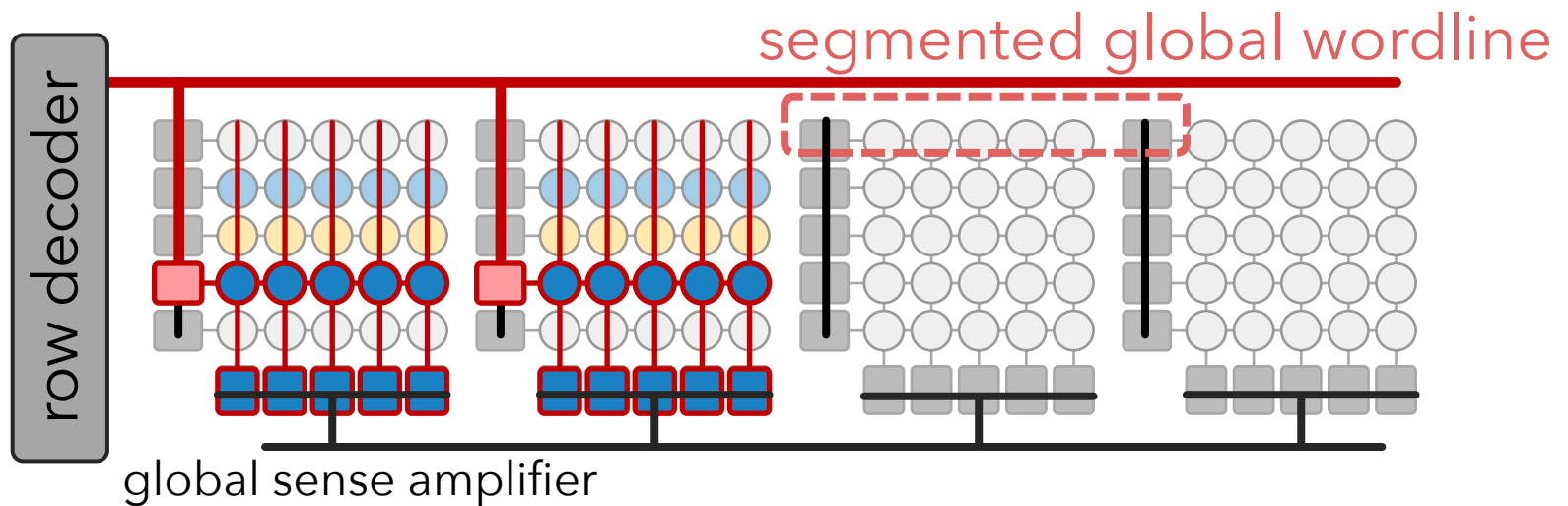
Fine-Grained DRAM:

segments the global wordline to access individual DRAM mats

MIMDRAM: Key Idea (II)

Fine-Grained DRAM:

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



Fine-grained DRAM for energy-efficient DRAM access:

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

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

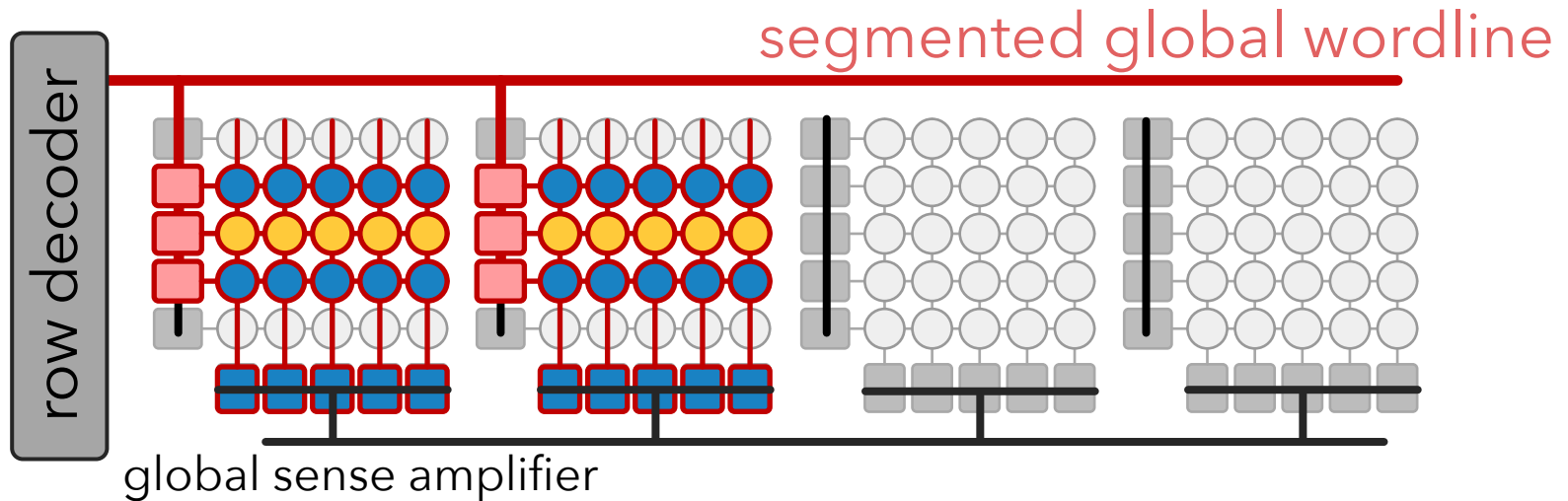
[Zhang+, 2014]: Half-DRAM

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

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

[Olgun+, 2024]: Sectored DRAM

MIMDRAM: Key Idea (III)

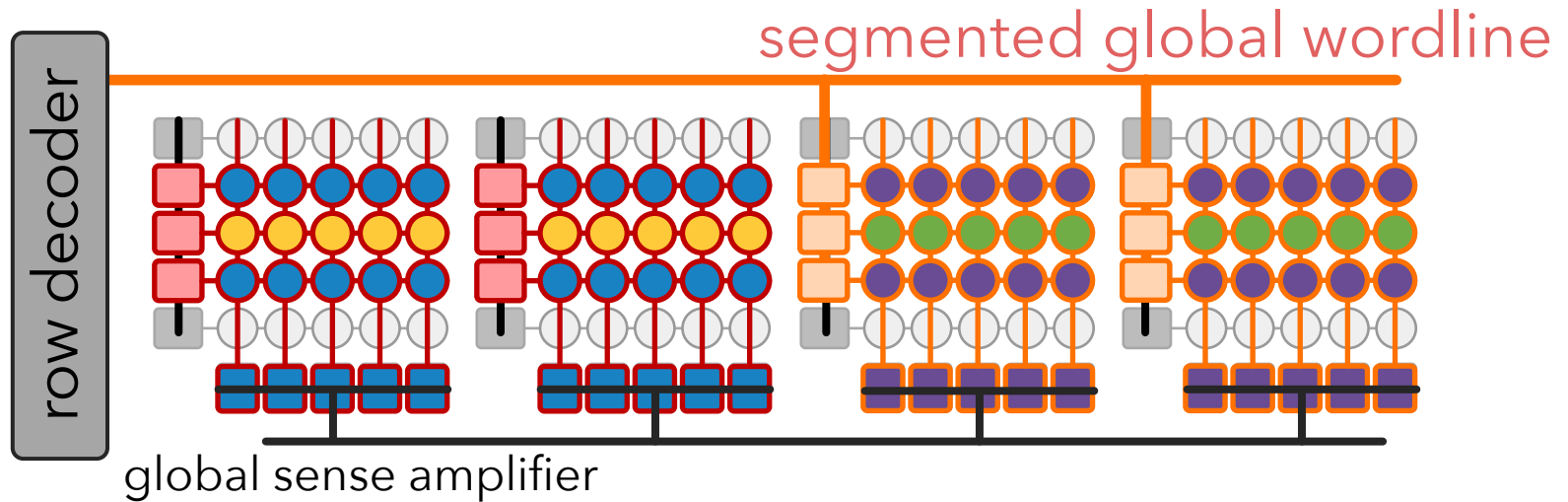


Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

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

MIMDRAM: Key Idea (III)

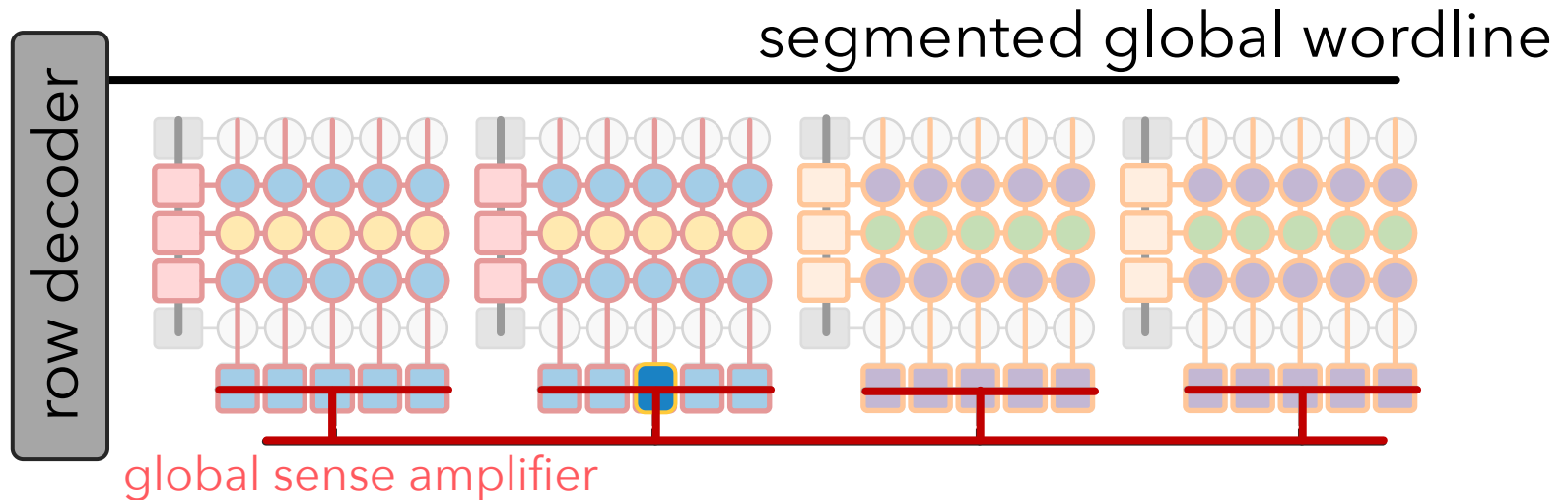


Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

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

MIMDRAM: Key Idea (III)



Fine-grained DRAM for processing-using-DRAM:

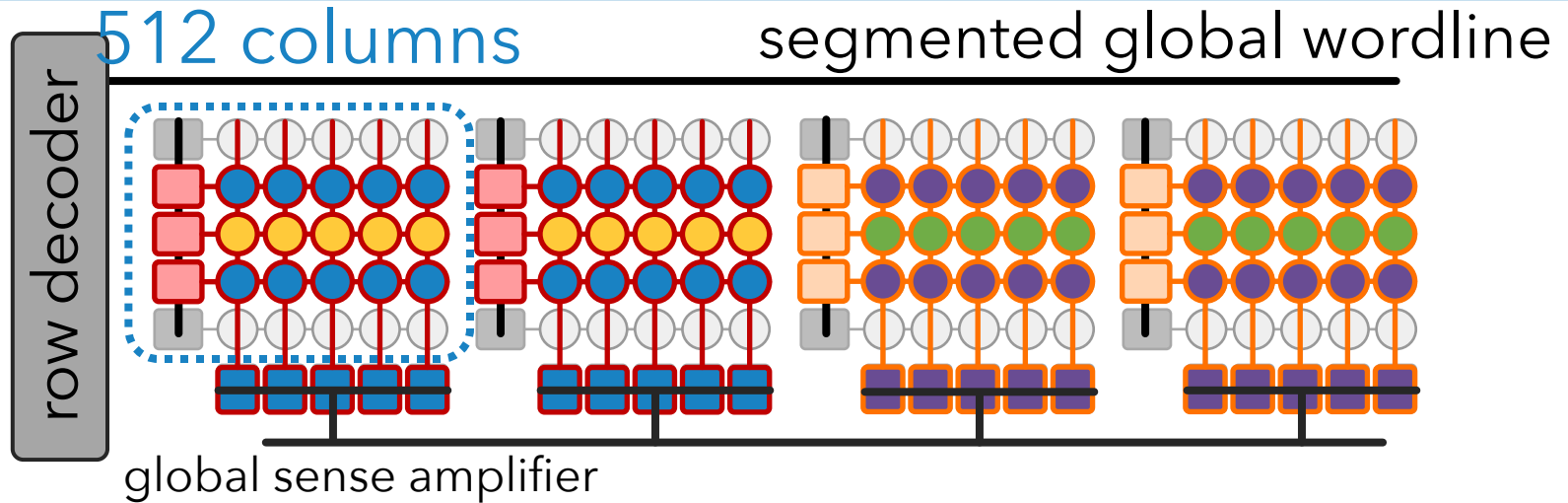
1 Improves SIMD utilization

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

2 Enables low-cost interconnects for vector reduction

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

MIMDRAM: Key Idea (III)



Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

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

2 Enables low-cost interconnects for vector reduction

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

3 Eases programmability

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

MIMDRAM: Overview

MIMDRAM is a hardware/software co-designed PUD system that enables **fine-grained PUD computation** at **low cost and programming effort**



Main components of MIMDRAM:

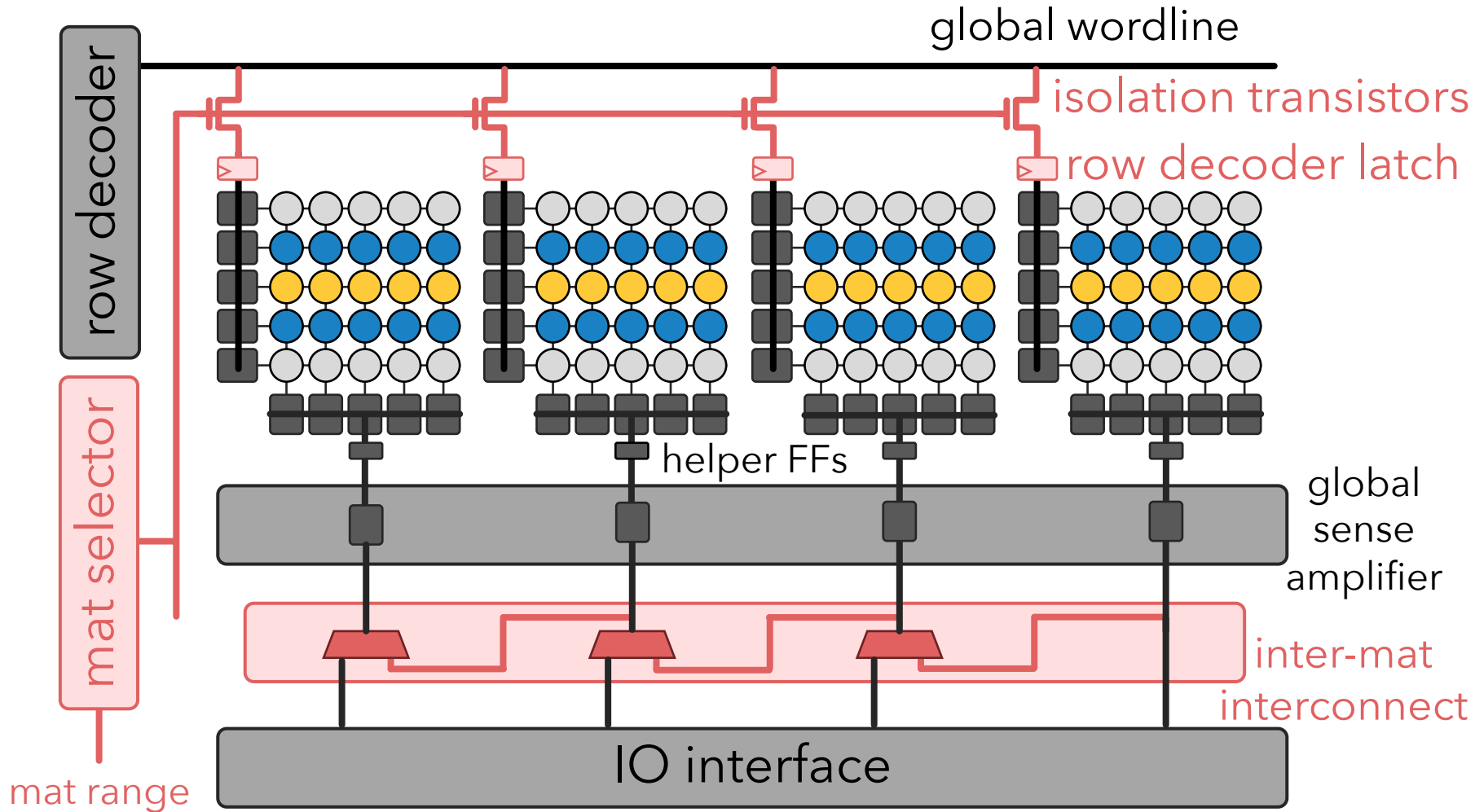
1 Hardware

- DRAM array modification to enable fine-grained PUD computation
- inter- and intra-mat interconnects to enable PUD vector reduction
- control unit design to orchestrate PUD execution

2 Software

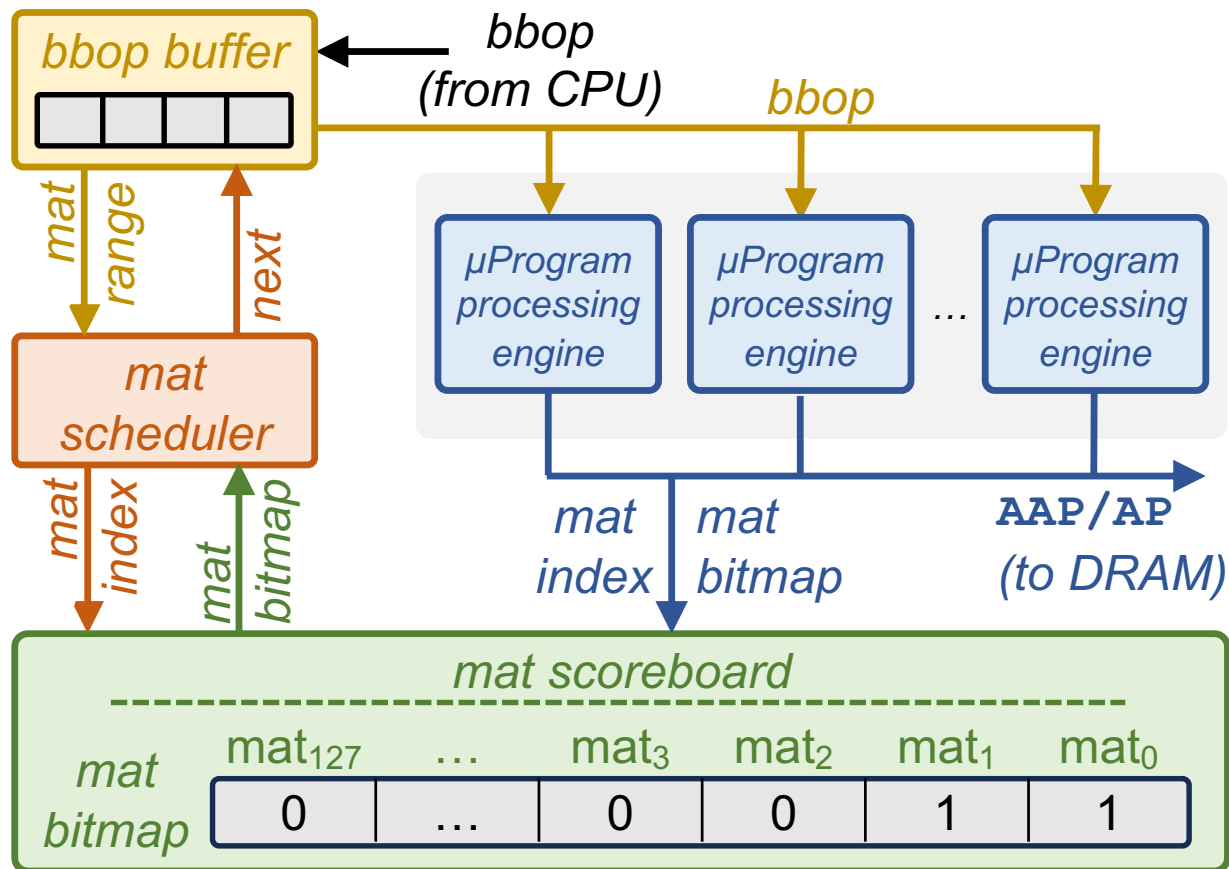
- compiler support to transparently generate PUD instructions
- system support to map and execute PUD instructions

MIMDRAM: Modifications to DRAM Chip



MIMDRAM: Control Unit Design

The control unit **schedules** and **orchestrates** the execution of multiple **PUD** operations **transparently**



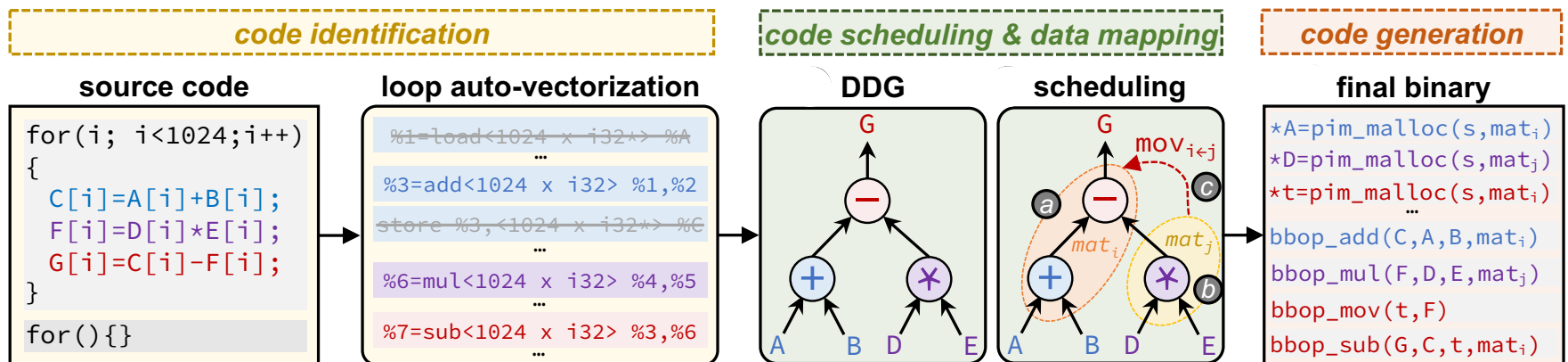
MIMDRAM: Compiler Support

Goal

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

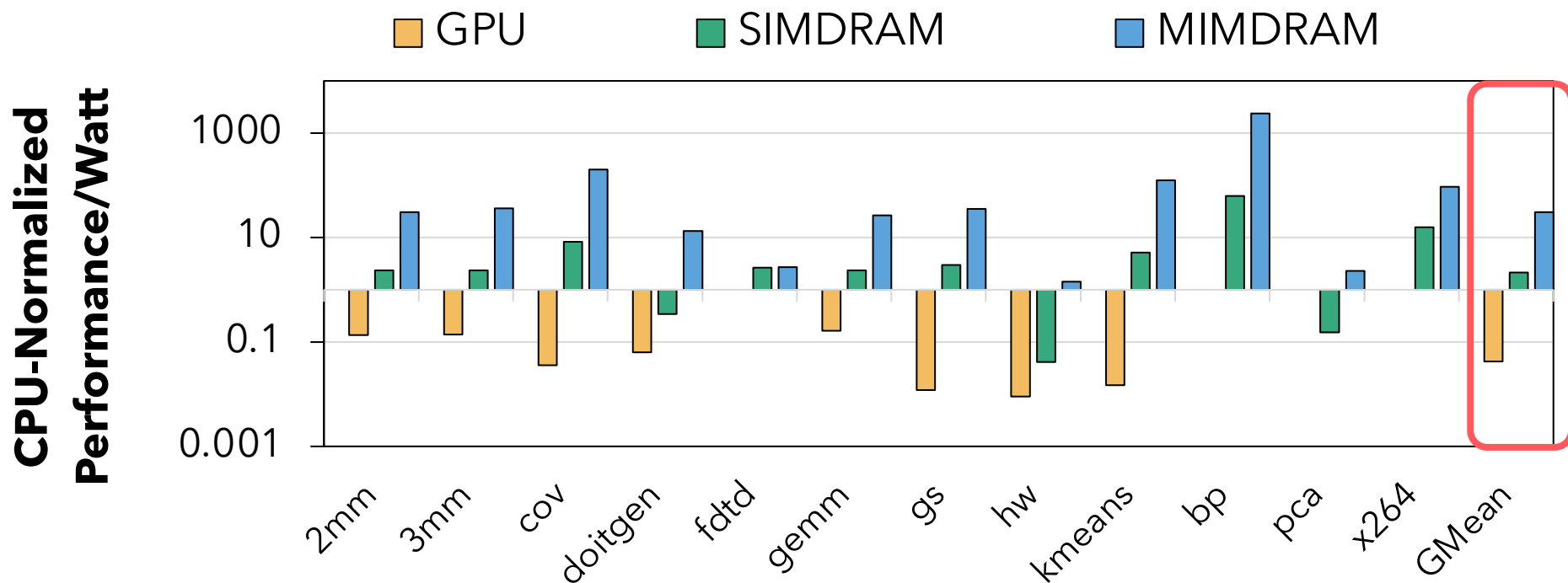


Three new LLVM-based passes targeting PUD execution



Evaluation:

Single Application Analysis - Energy Efficiency



Takeaway

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

More on MIMDRAM

- Geraldo F. Oliveira, Ataberk Olgun, Abdullah Giray Yağlıkçı, F. Nisa Bostancı, Juan Gómez-Luna, Saugata Ghose, and Onur Mutlu
" MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Computing"
Proceedings of the 30th International Symposium on High-Performance Computer Architecture (HPCA), Edinburgh, Scotland, March 2024.

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

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

[†] *ETH Zürich*

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

In-DRAM Lookup-Table Based Execution

João Dinis Ferreira, Gabriel Falcao, Juan Gómez-Luna, Mohammed Alser, Lois Orosa, Mohammad Sadrosadati, Jeremie S. Kim, Geraldo F. Oliveira, Taha Shahroodi, Anant Nori, and Onur Mutlu, "**pLUTO: Enabling Massively Parallel Computation in DRAM via Lookup Tables**" *Proceedings of the 55th International Symposium on Microarchitecture (MICRO)*, Chicago, IL, USA, October 2022.

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

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

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

[[arXiv version](#)]

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

Officially artifact evaluated as available, reusable and reproducible.



pLUTO: Enabling Massively Parallel Computation in DRAM via Lookup Tables

João Dinis Ferreira[§]

Gabriel Falcao[†]

Juan Gómez-Luna[§]

Mohammed Alser[§]

Lois Orosa^{§∇}

Mohammad Sadrosadati[§]

Jeremie S. Kim[§]

Geraldo F. Oliveira[§]

Taha Shahroodi[‡]

Anant Nori^{*}

Onur Mutlu[§]

[§]ETH Zürich

[†]IT, University of Coimbra

[∇]Galicia Supercomputing Center

[‡]TU Delft

^{*}Intel

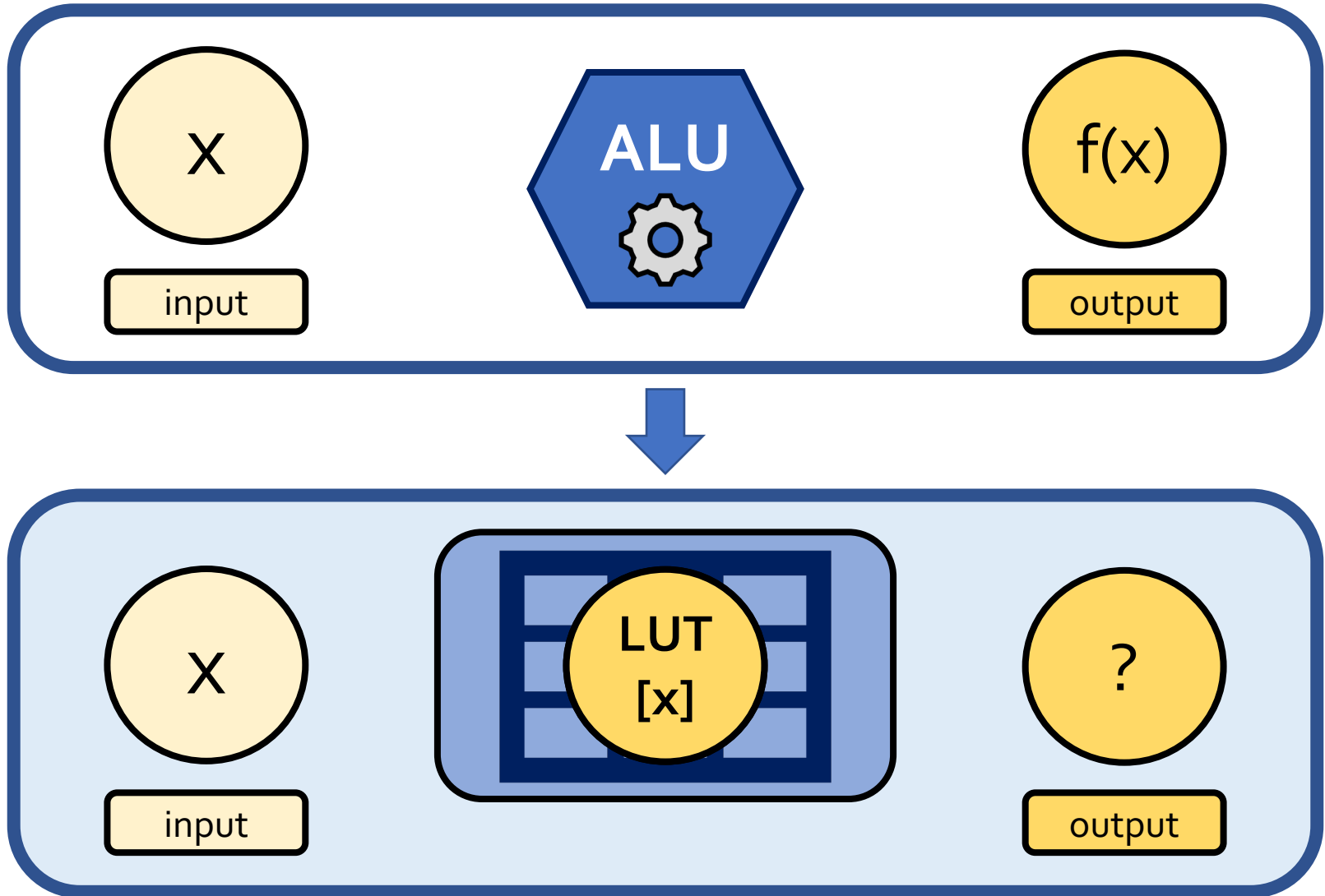
The Goal of pLUTo

Extend Processing-using-DRAM to support the execution of *arbitrarily complex operations*

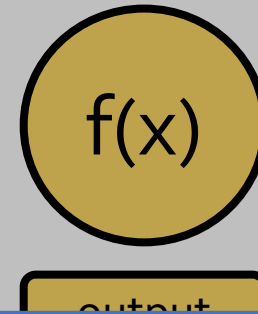
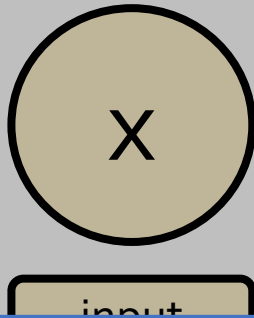
pLUTo: Key Idea



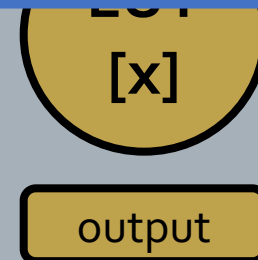
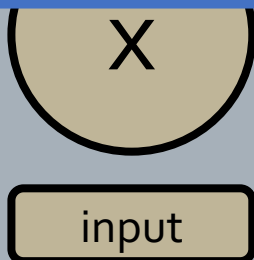
pLUTo: Key Idea



pLUTo: Key Idea



Replace **computation** with **memory accesses**
→ *pLUTo LUT Query* operation



In-DRAM pLUTo LUT Query: Setup

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

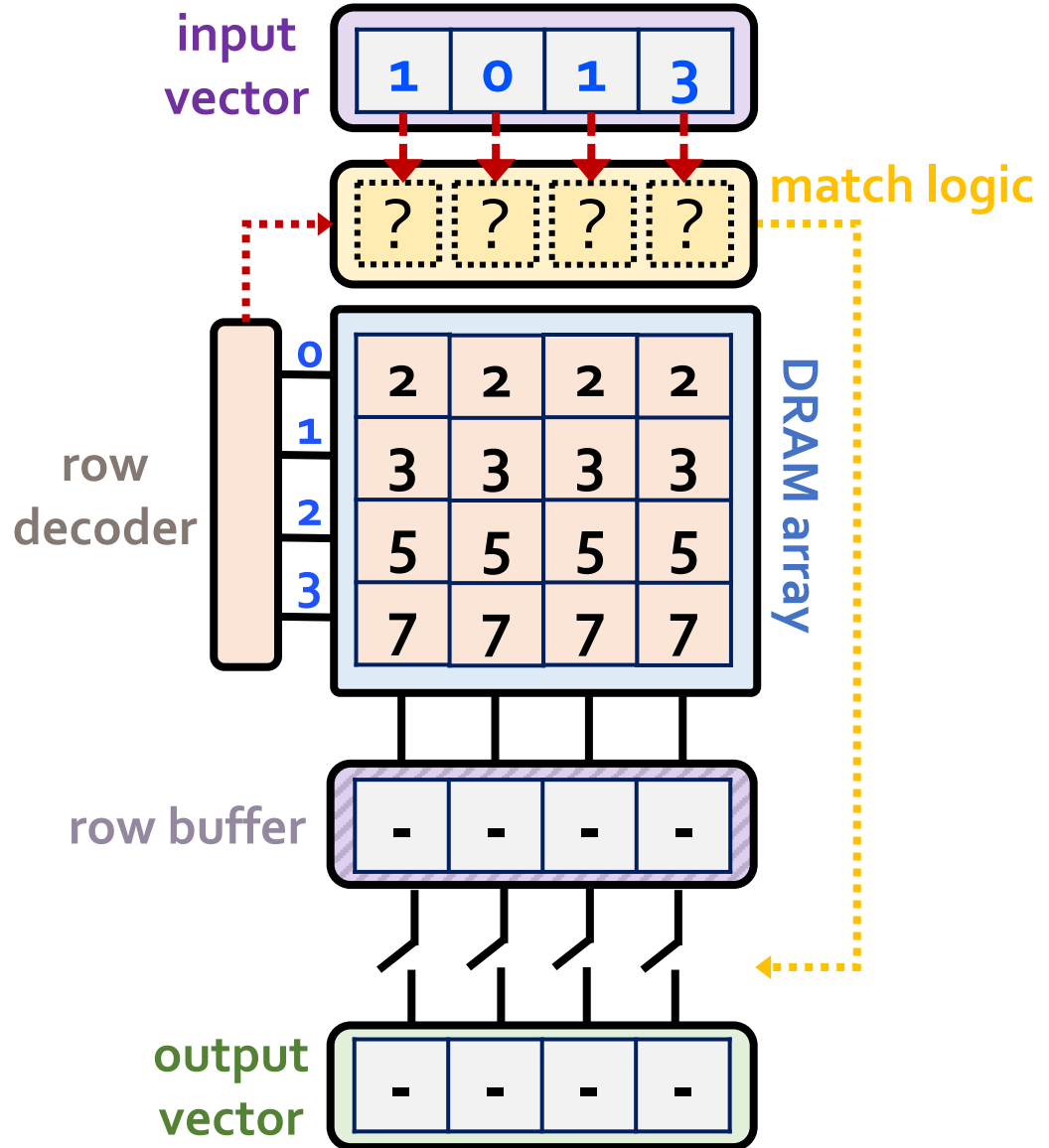
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

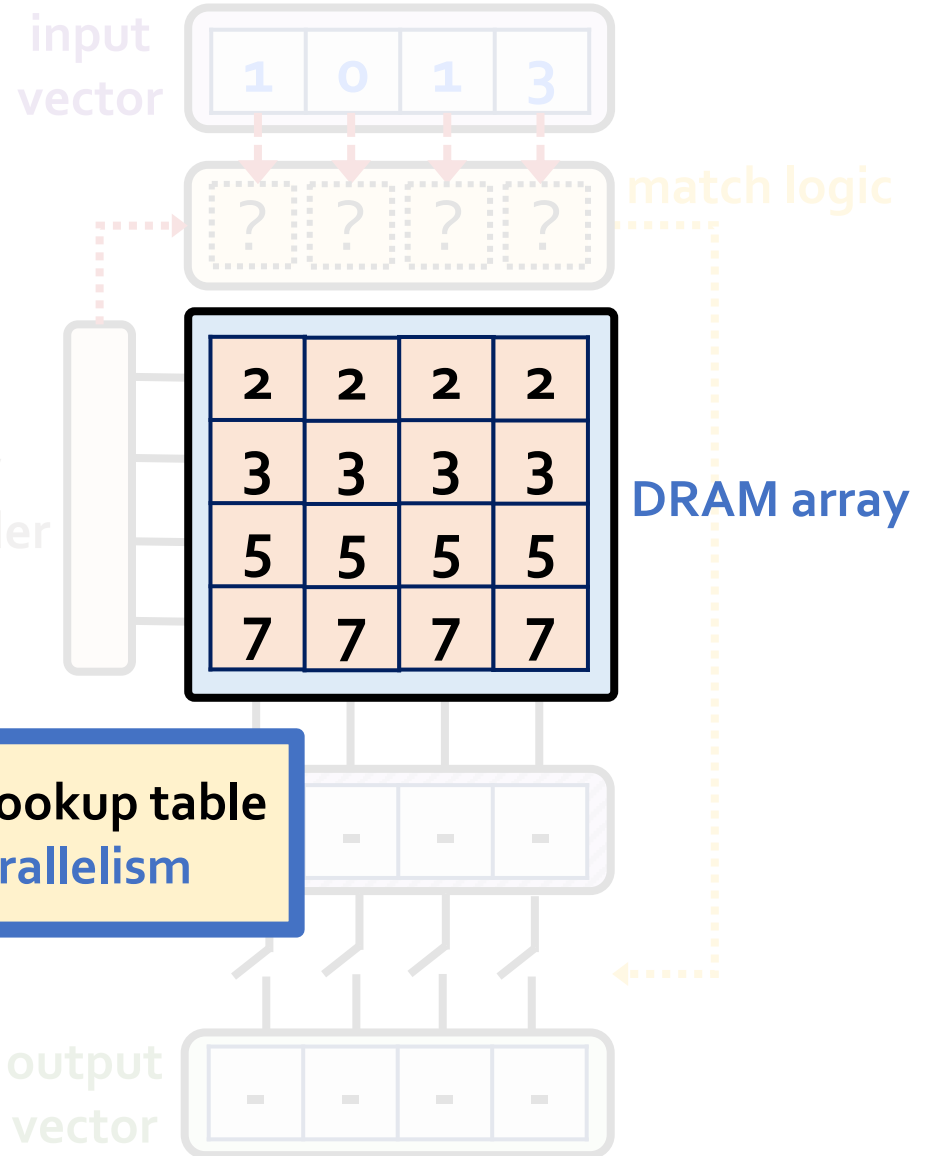
output vector



In-DRAM pLUTo LUT Query: Setup

LUT Query:
Return {2nd, 1st, 2nd, 4th}
prime numbers

LUT index	i	f(i)
0	1 st	2
1	2 nd	3
2	3 rd	5
3	4 th	7



Multiple copies of the lookup table
→ exploit DRAM parallelism

input vector

output vector

In-DRAM pLUTo LUT Query: Setup

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

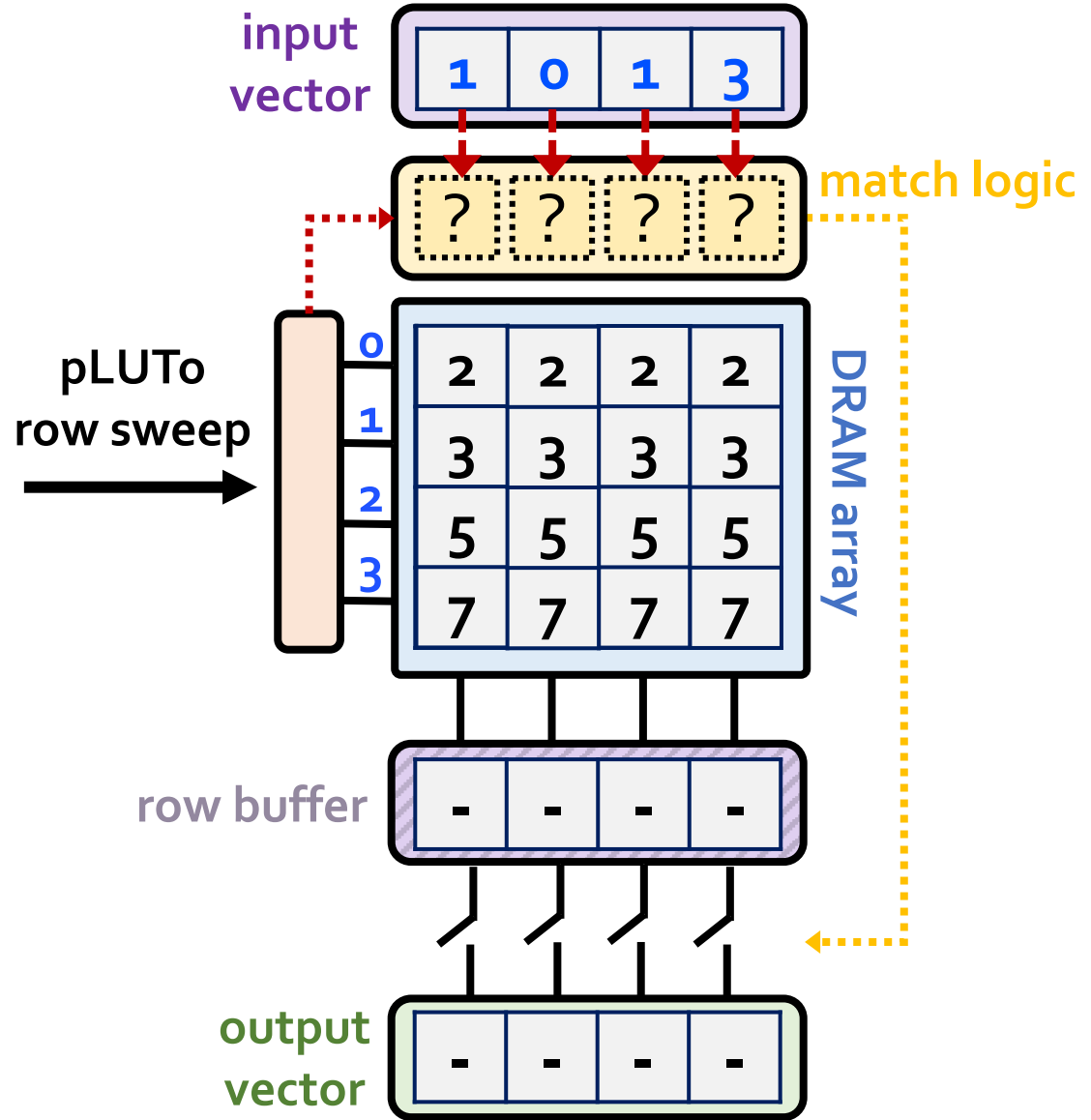
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

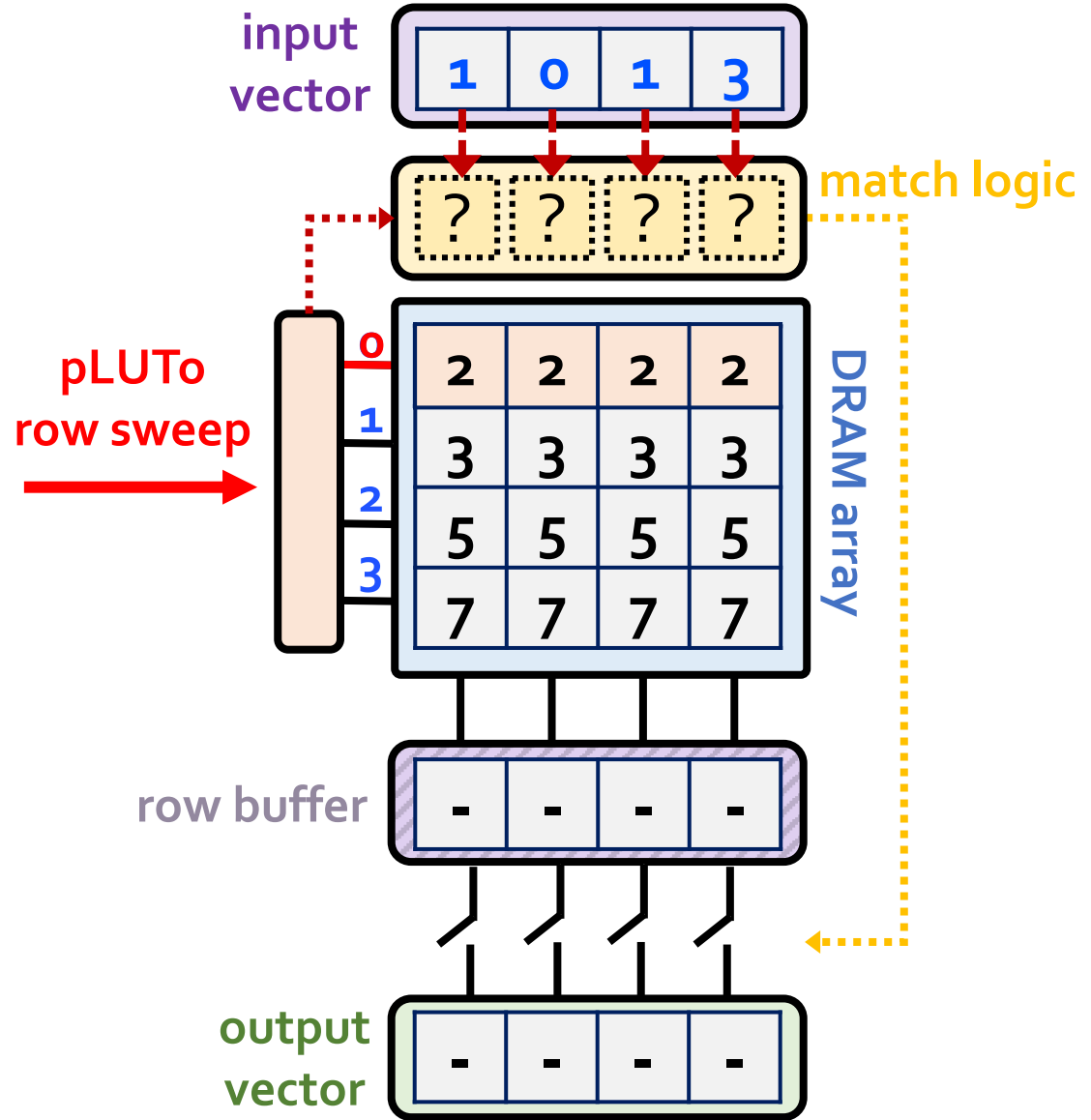
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

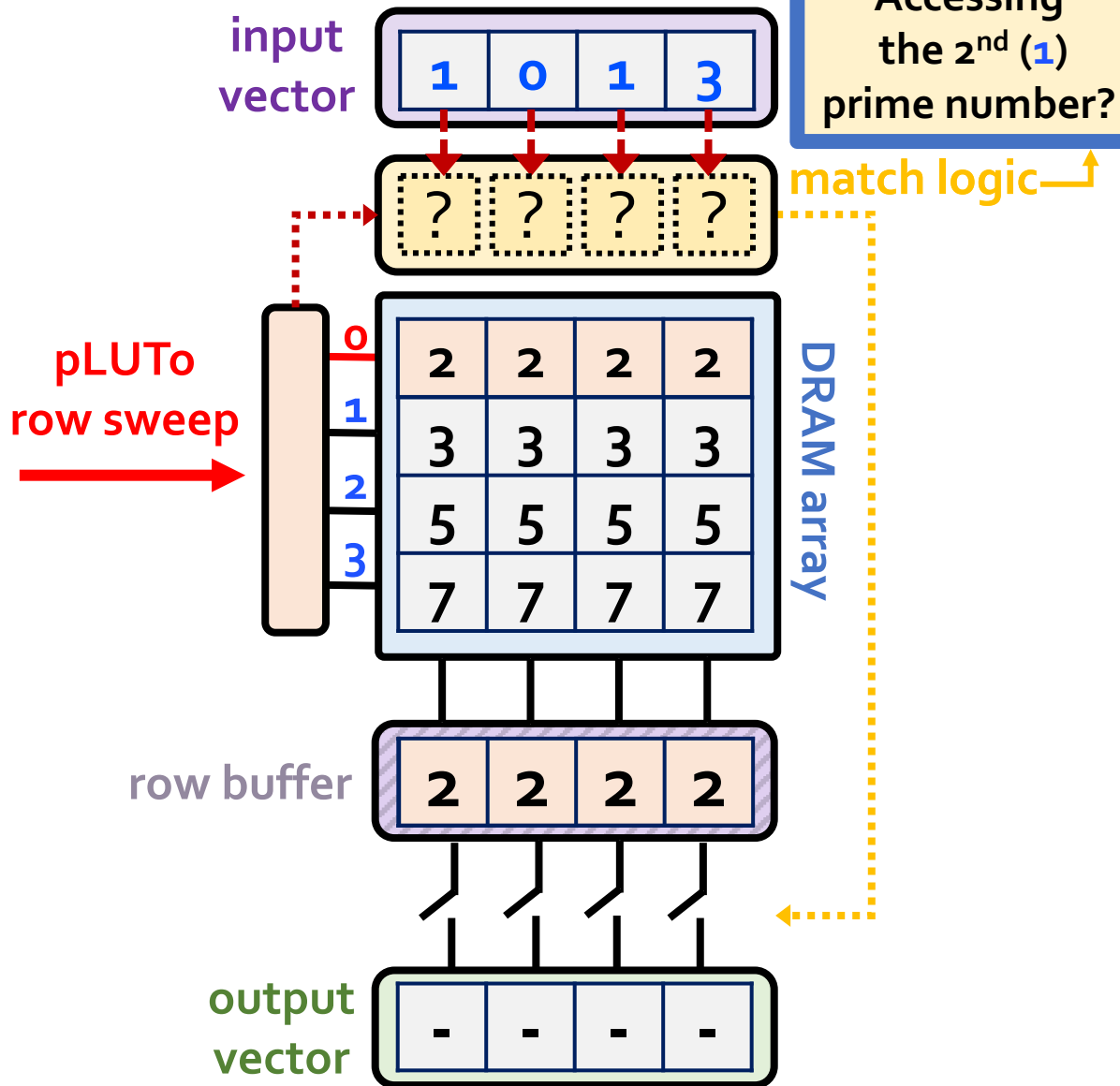
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

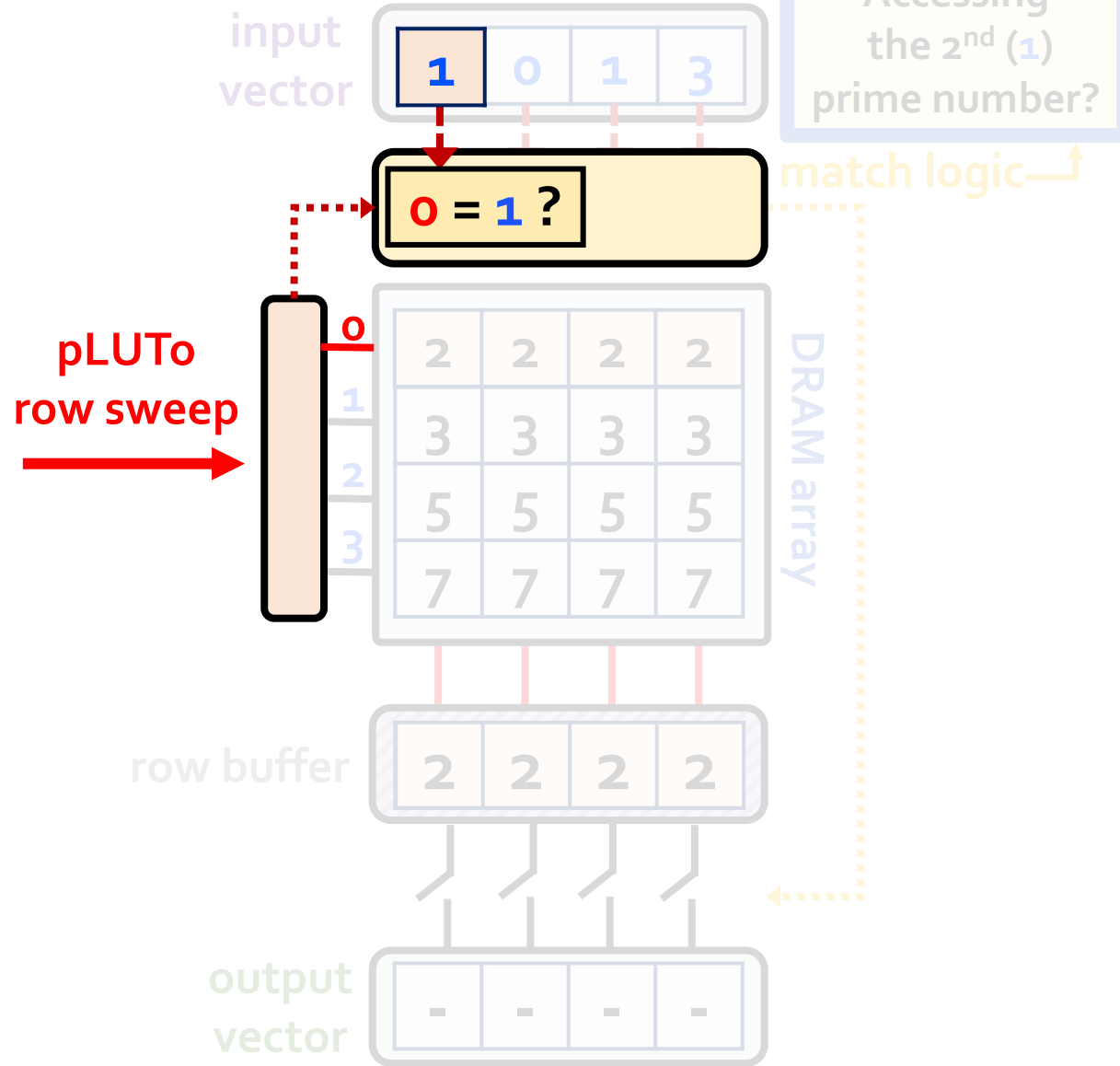
lookup table

1 0 1 3

input vector

3 2 3 7

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

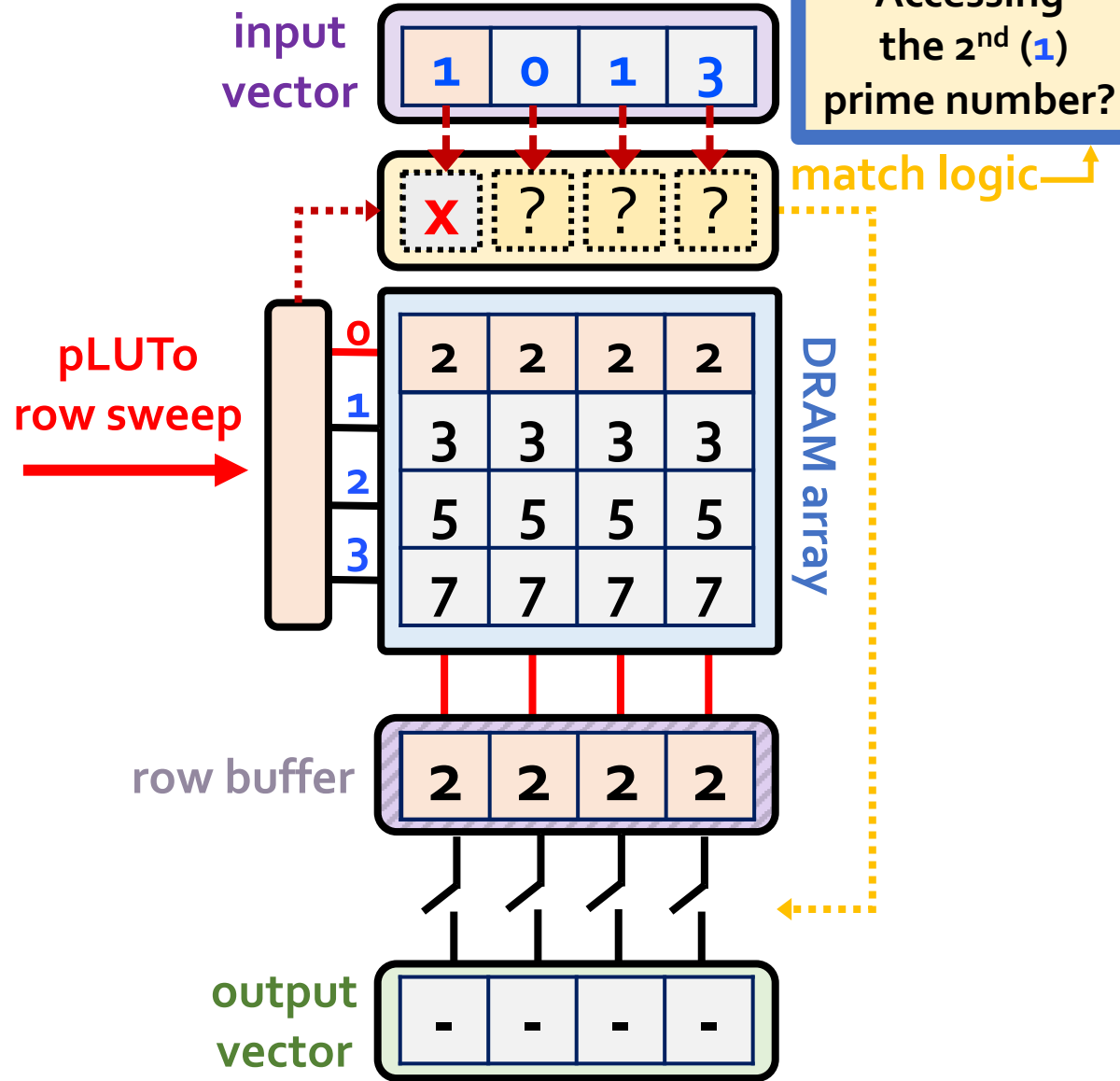
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

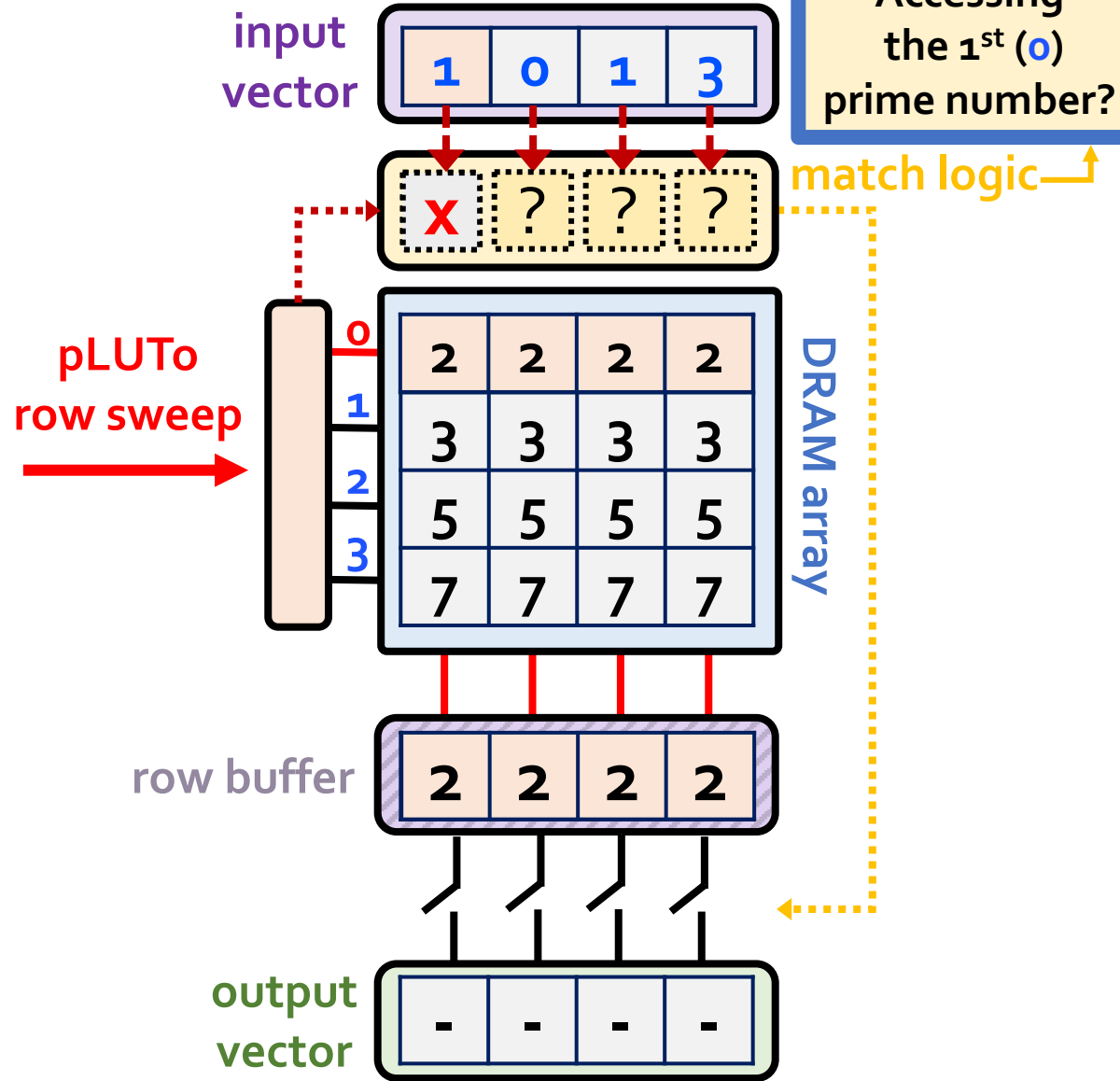
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

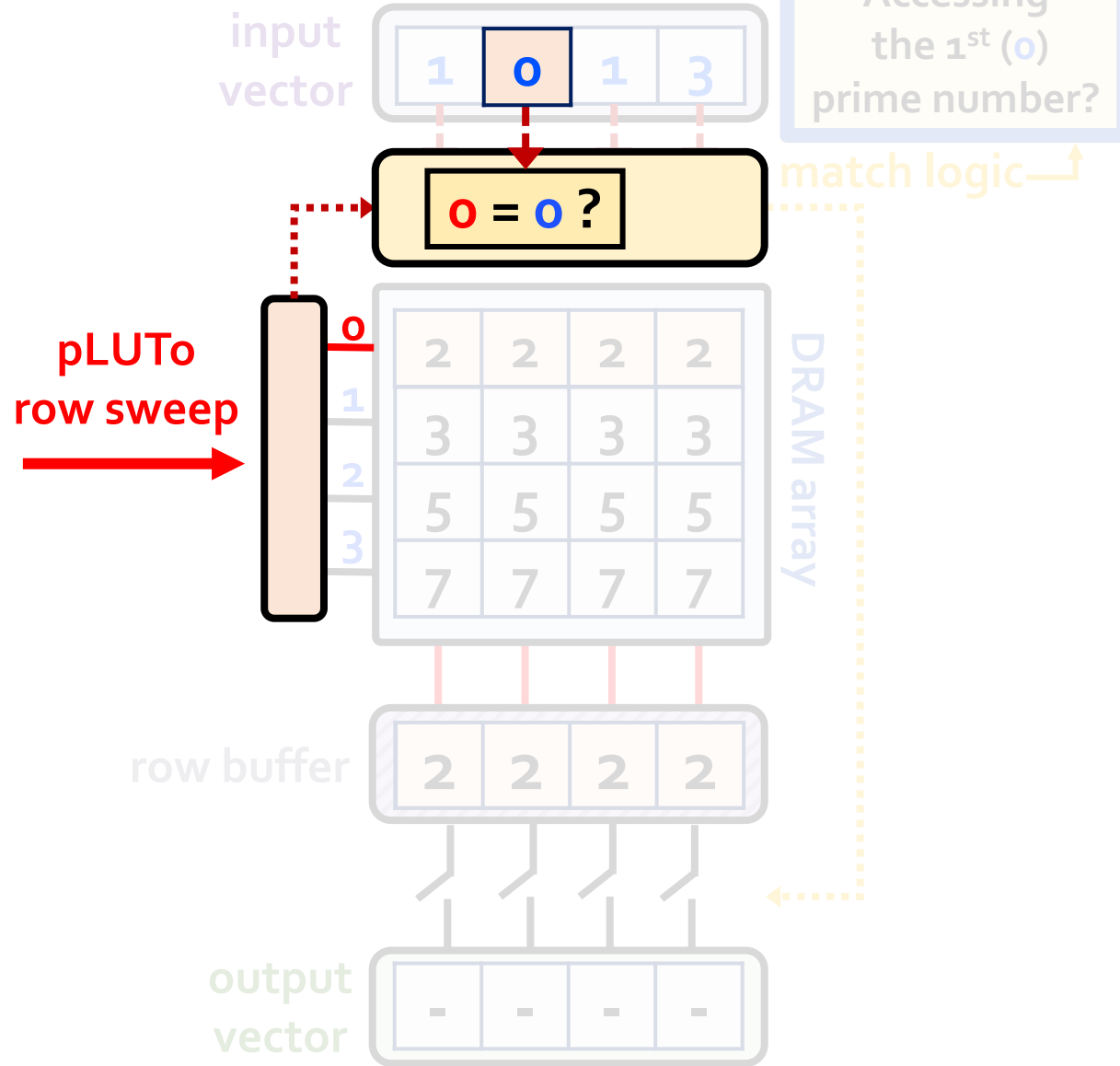
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

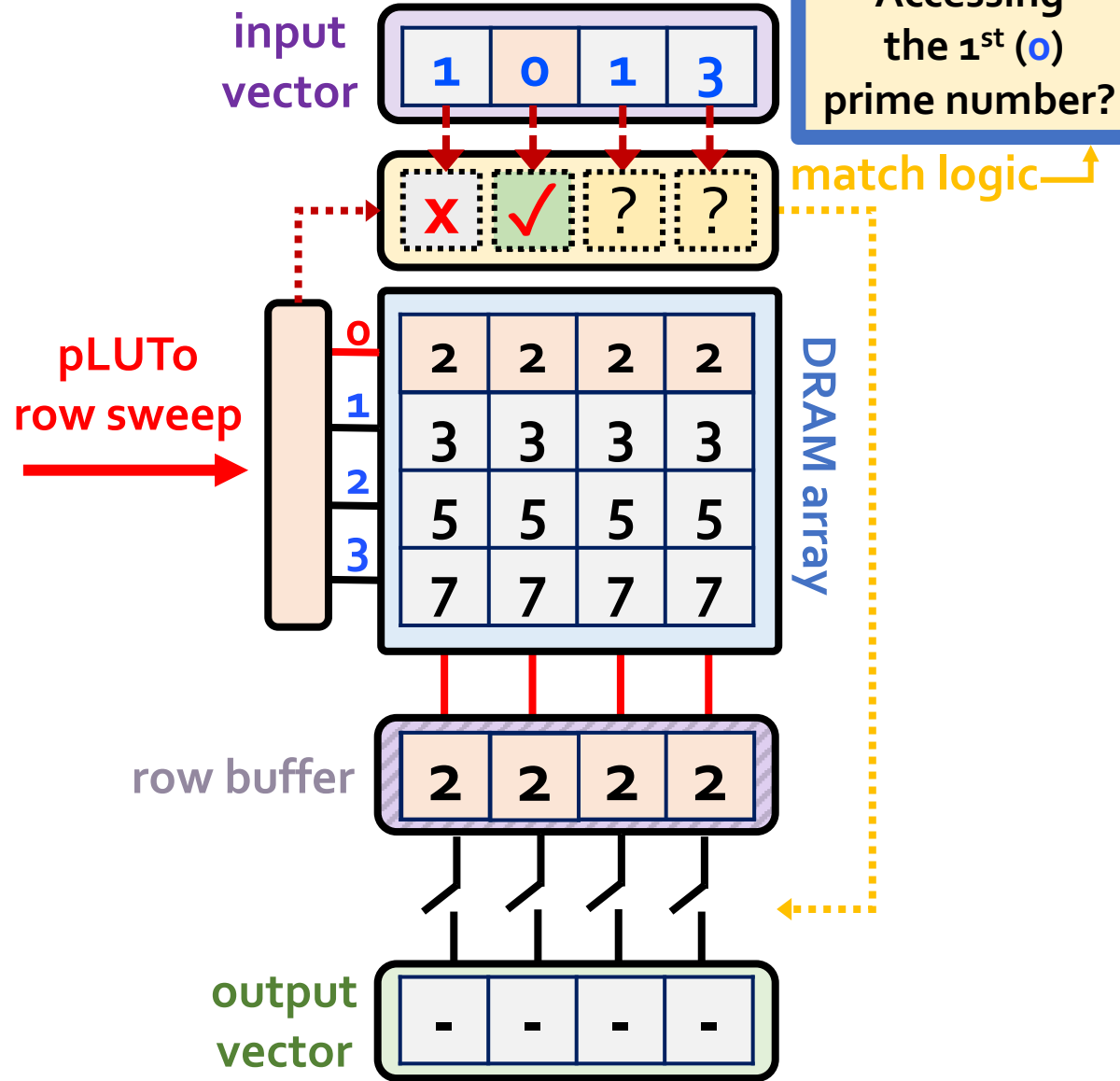
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

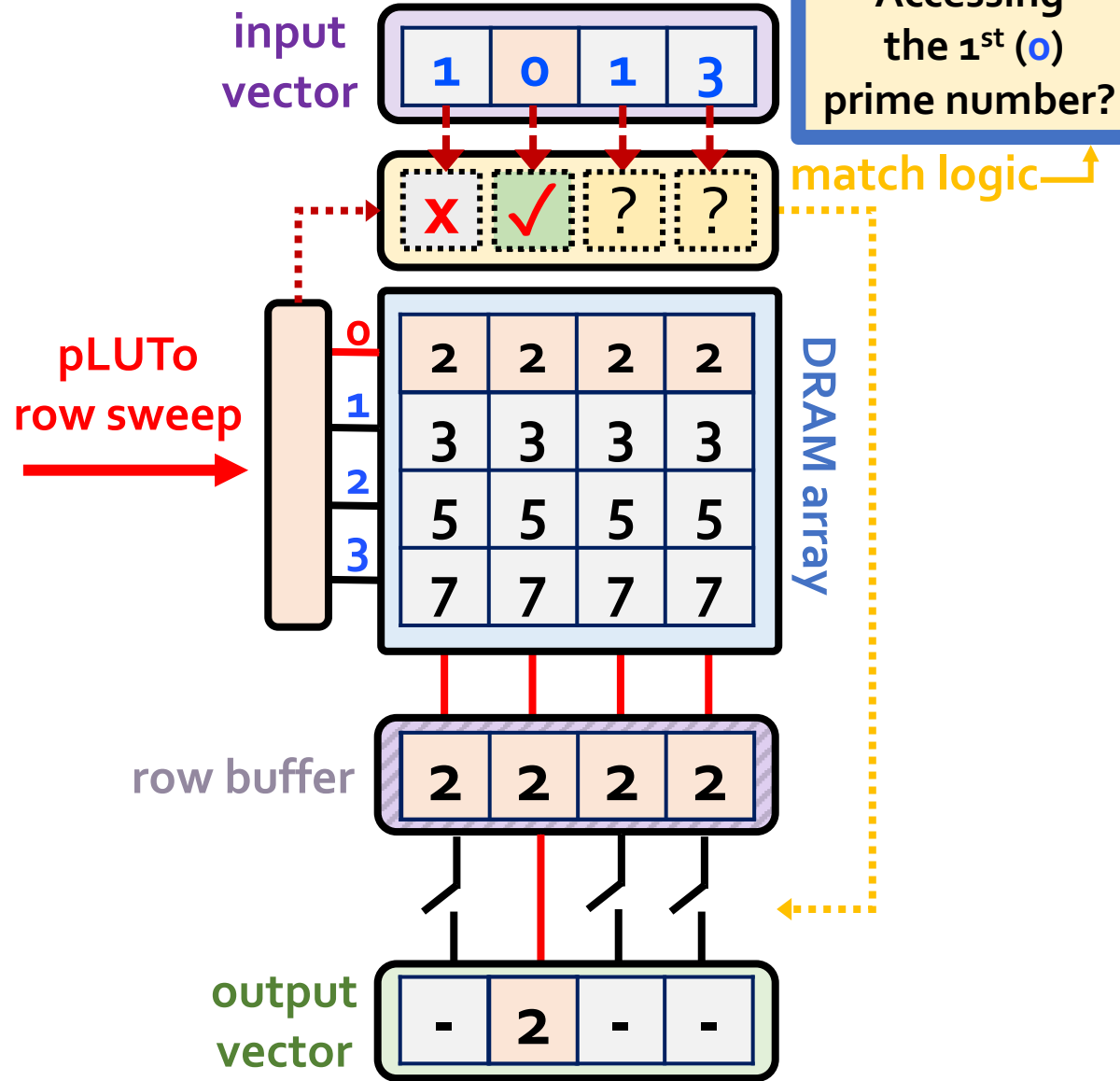
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

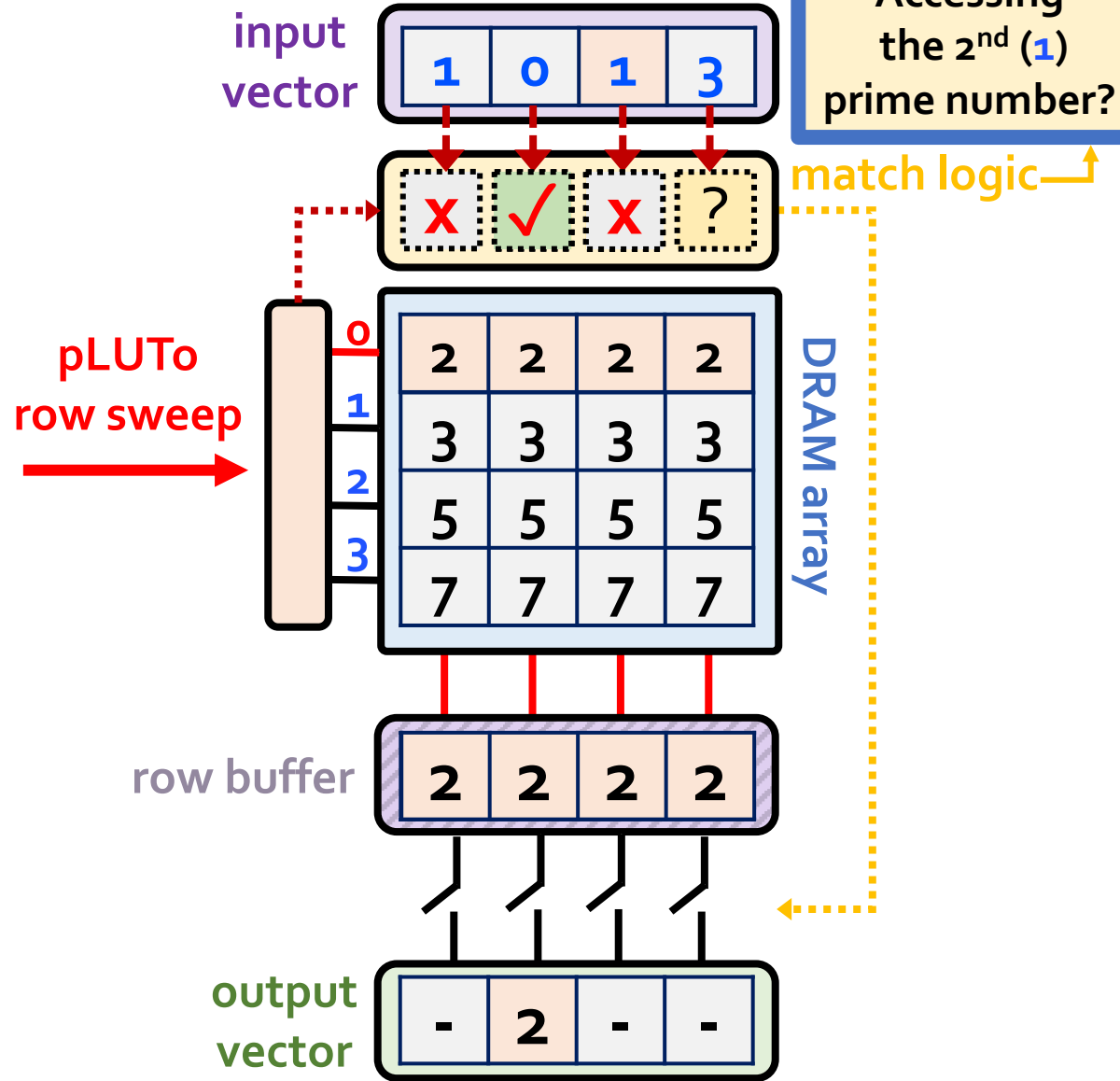
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 1

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

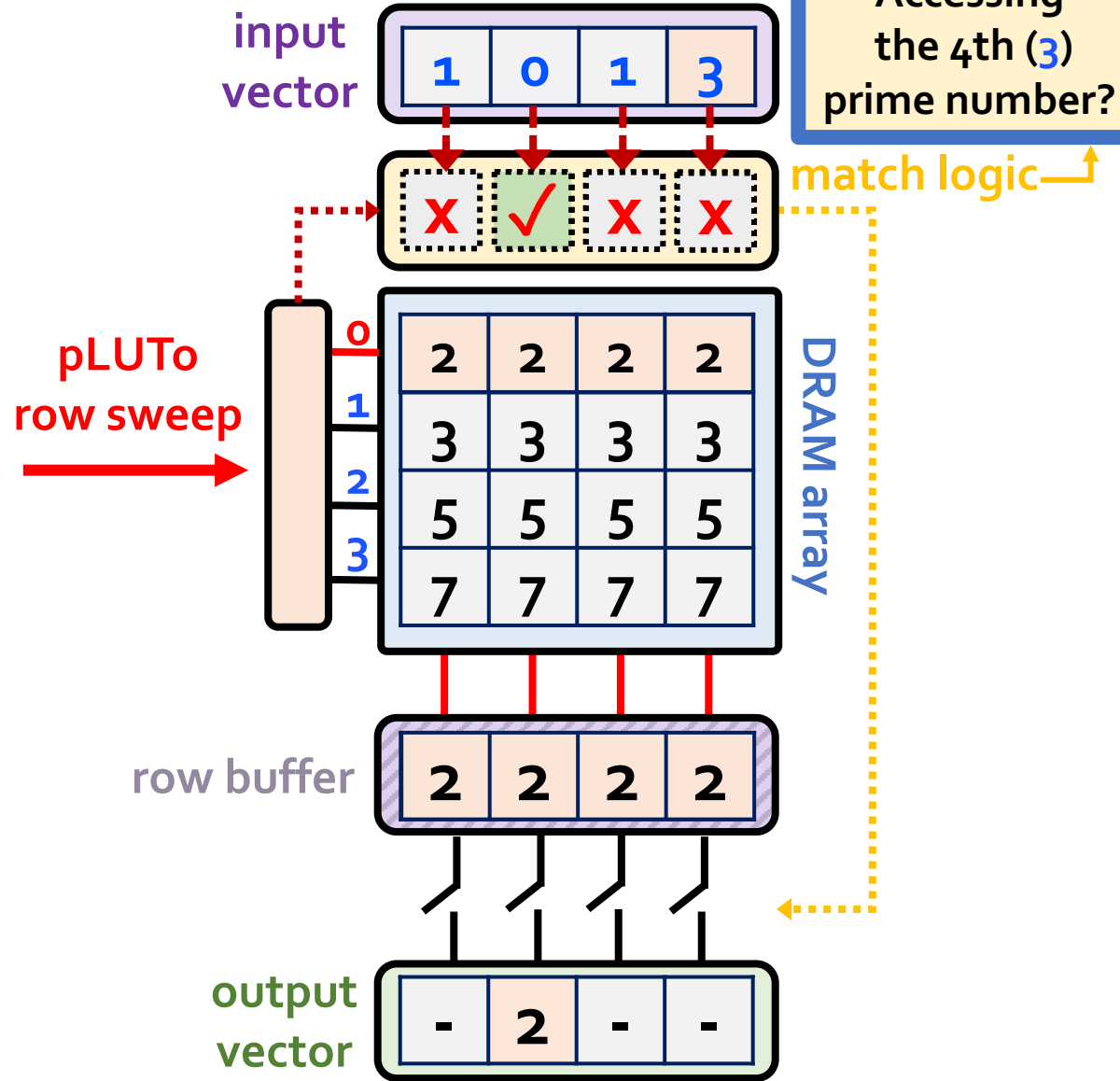
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 2

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

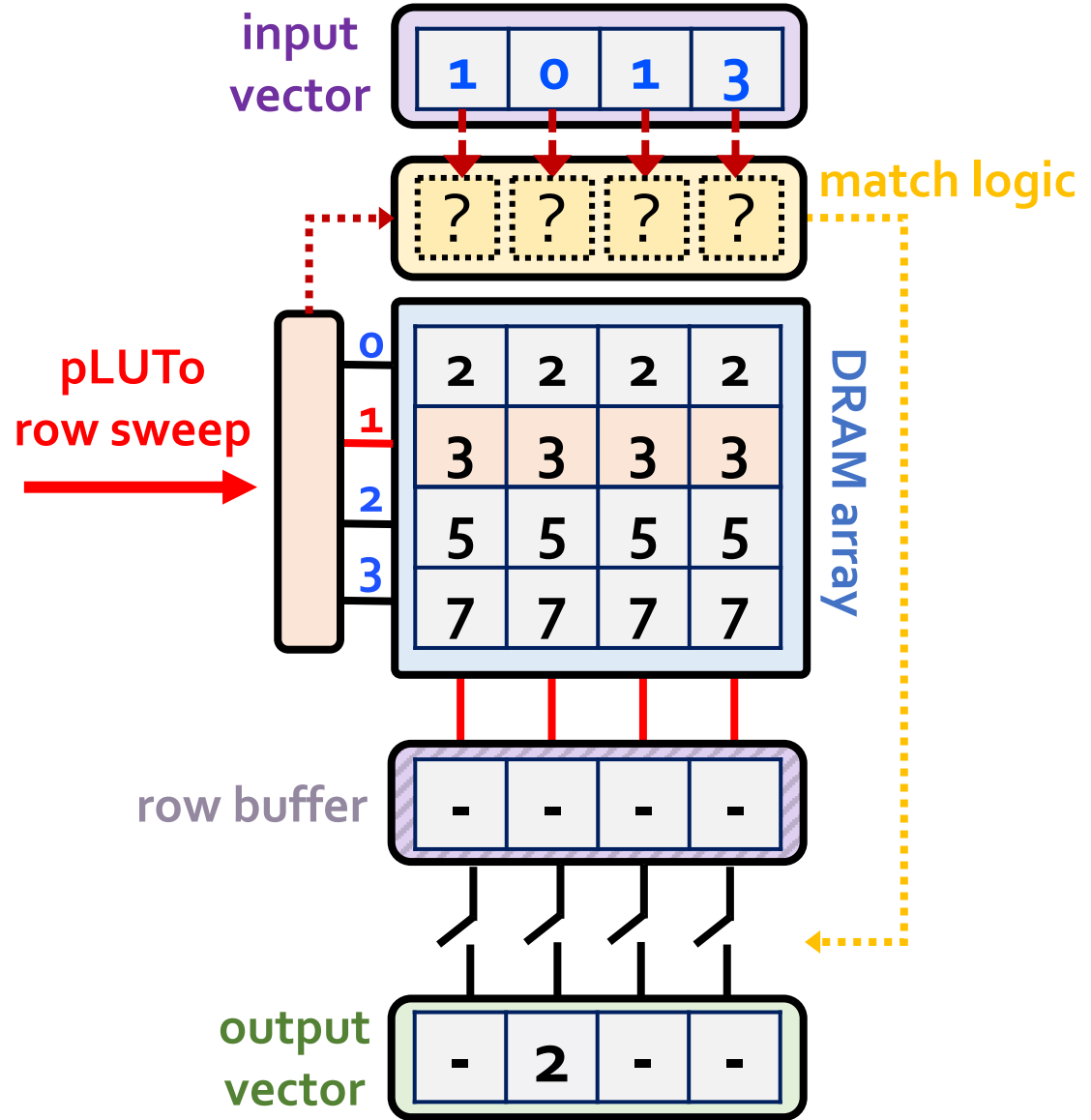
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 2

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

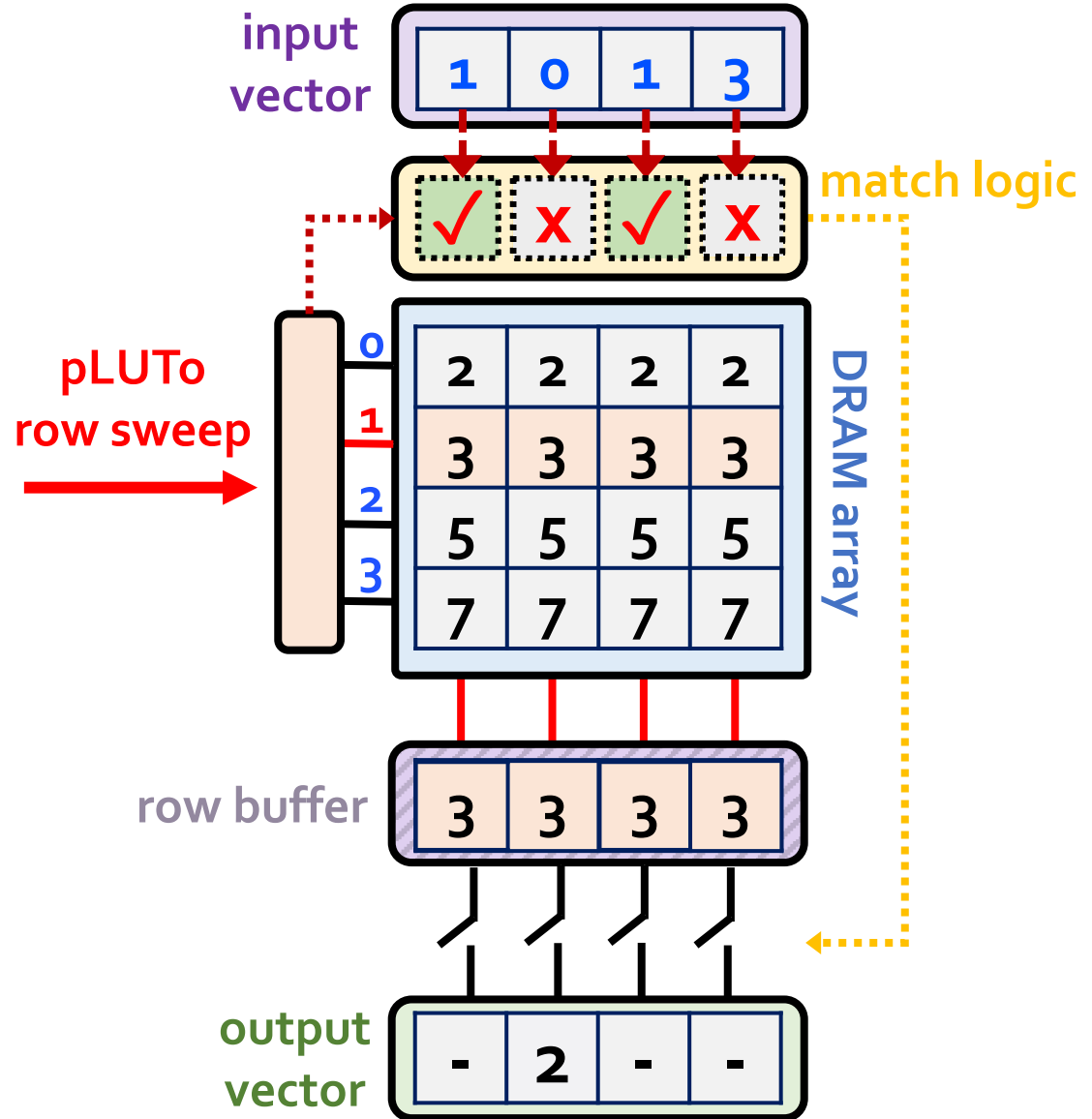
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 2

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

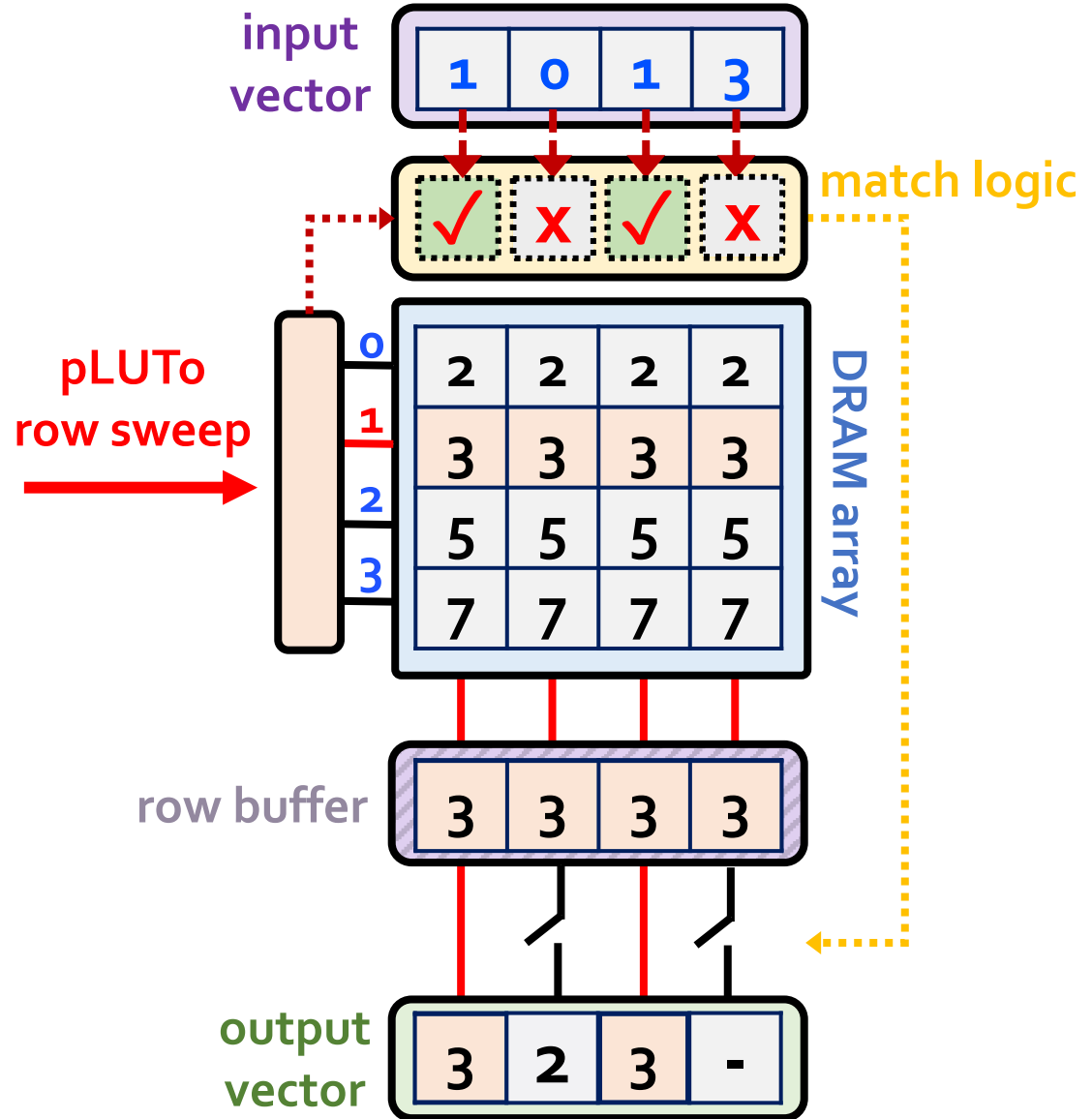
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 3

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

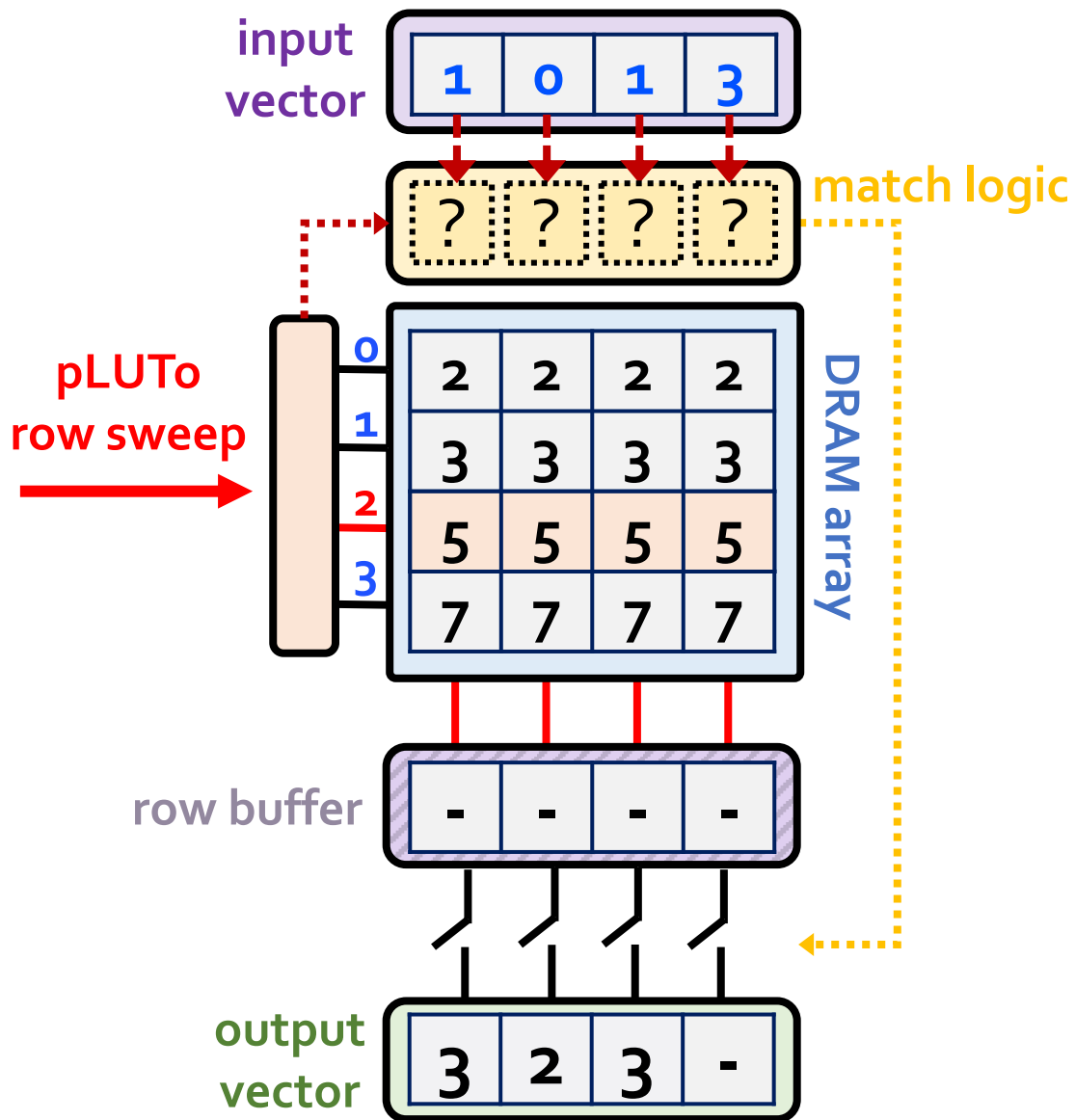
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 3

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

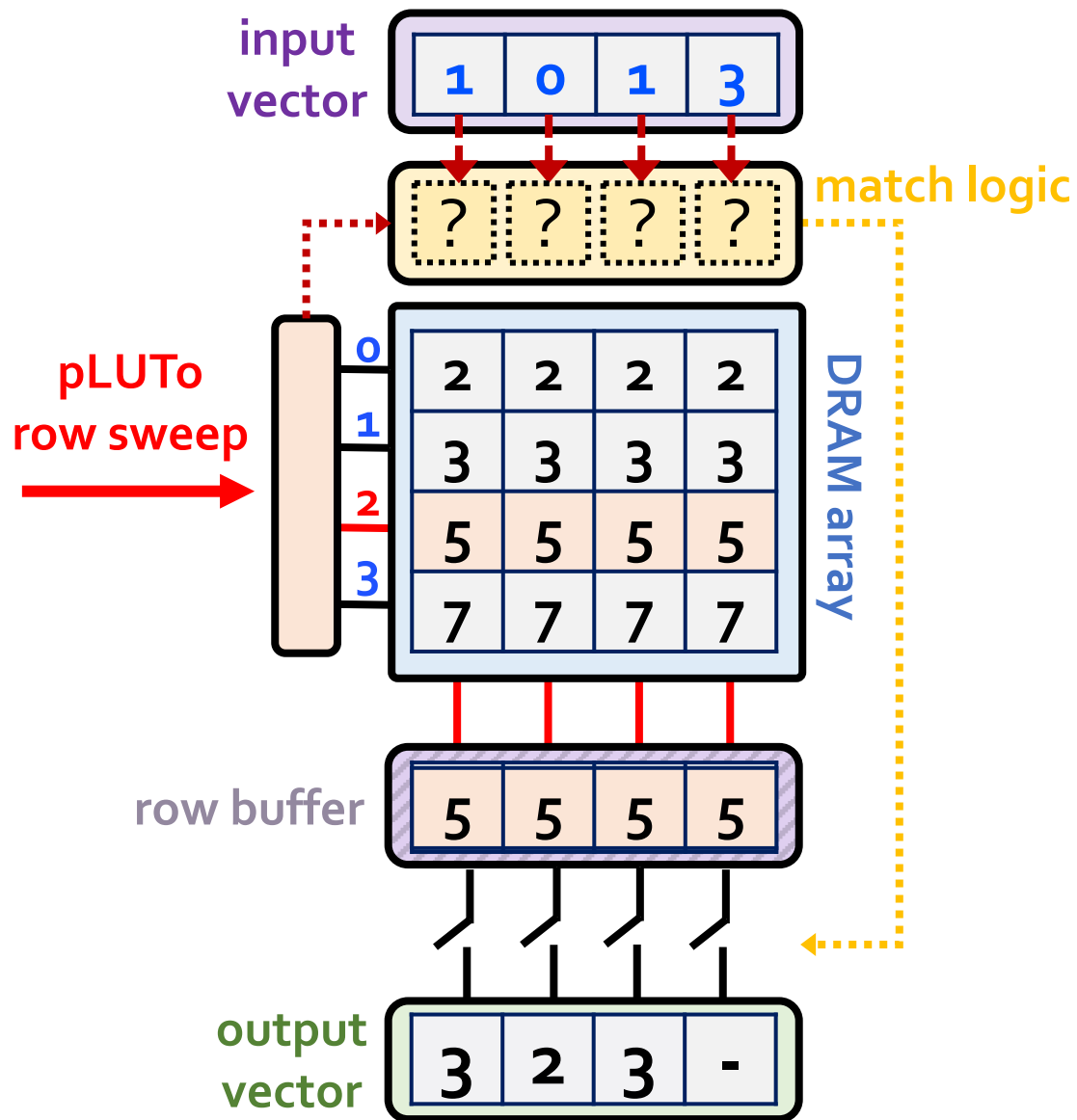
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 3

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

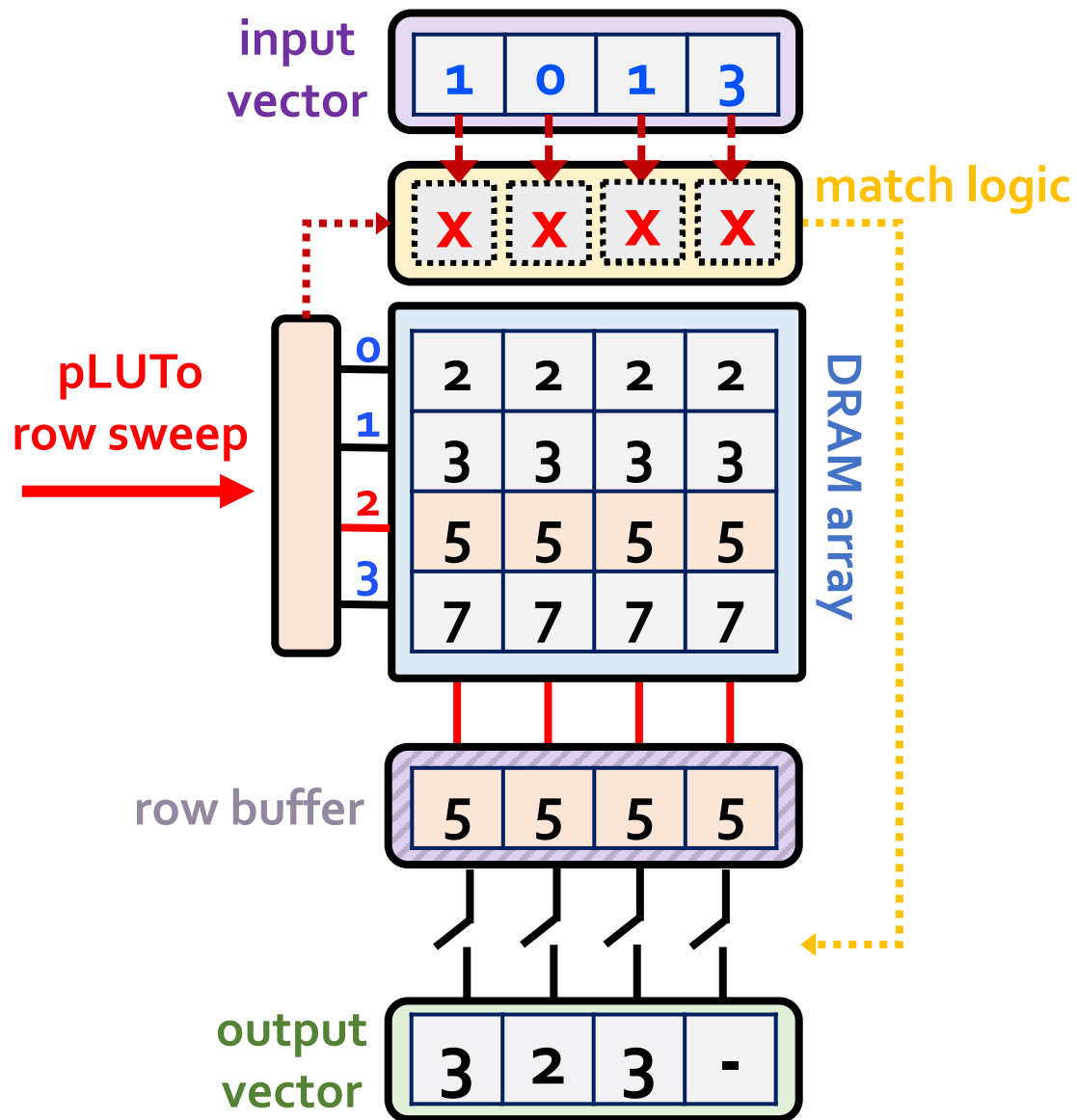
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 4

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

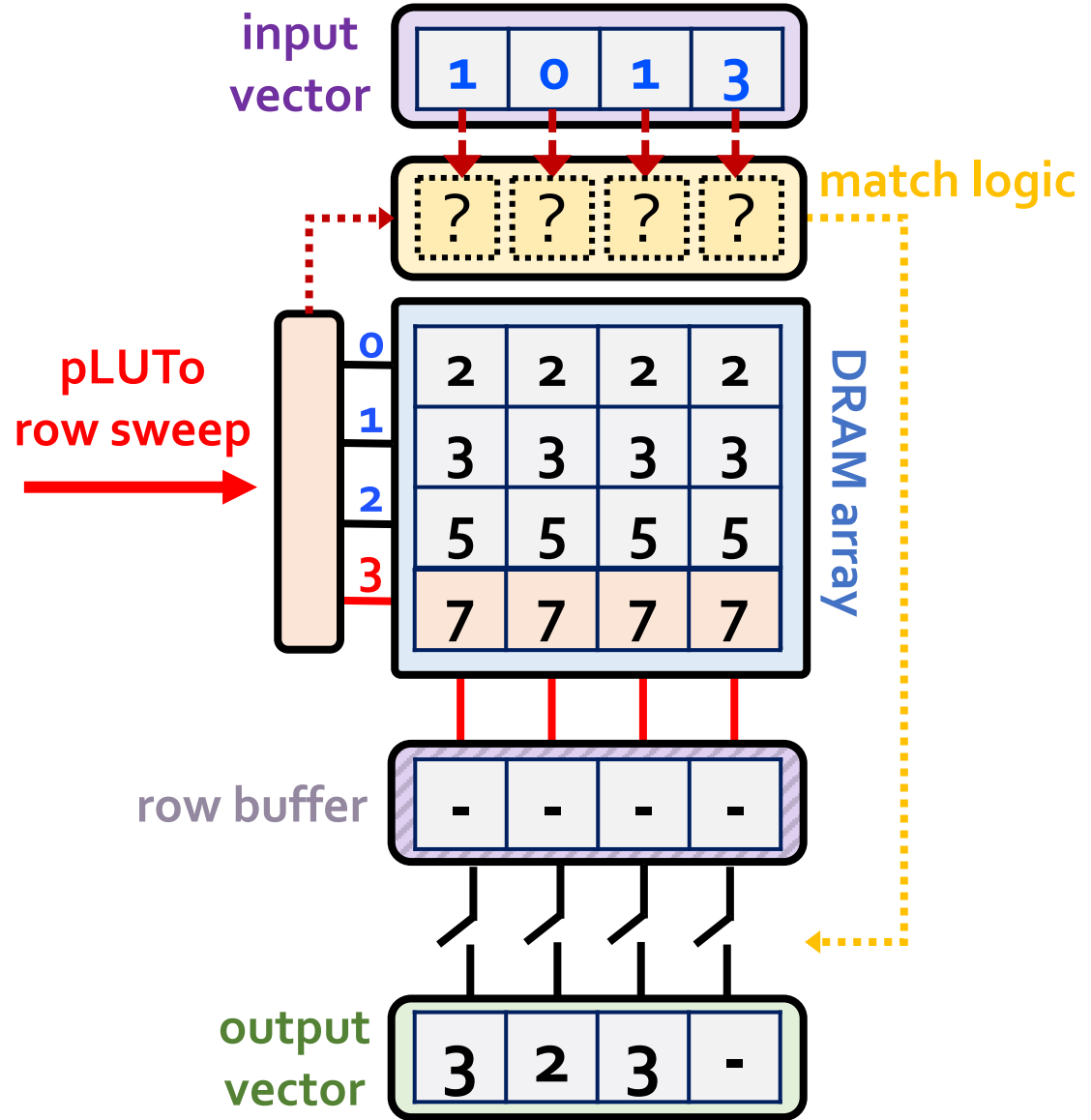
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 4

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

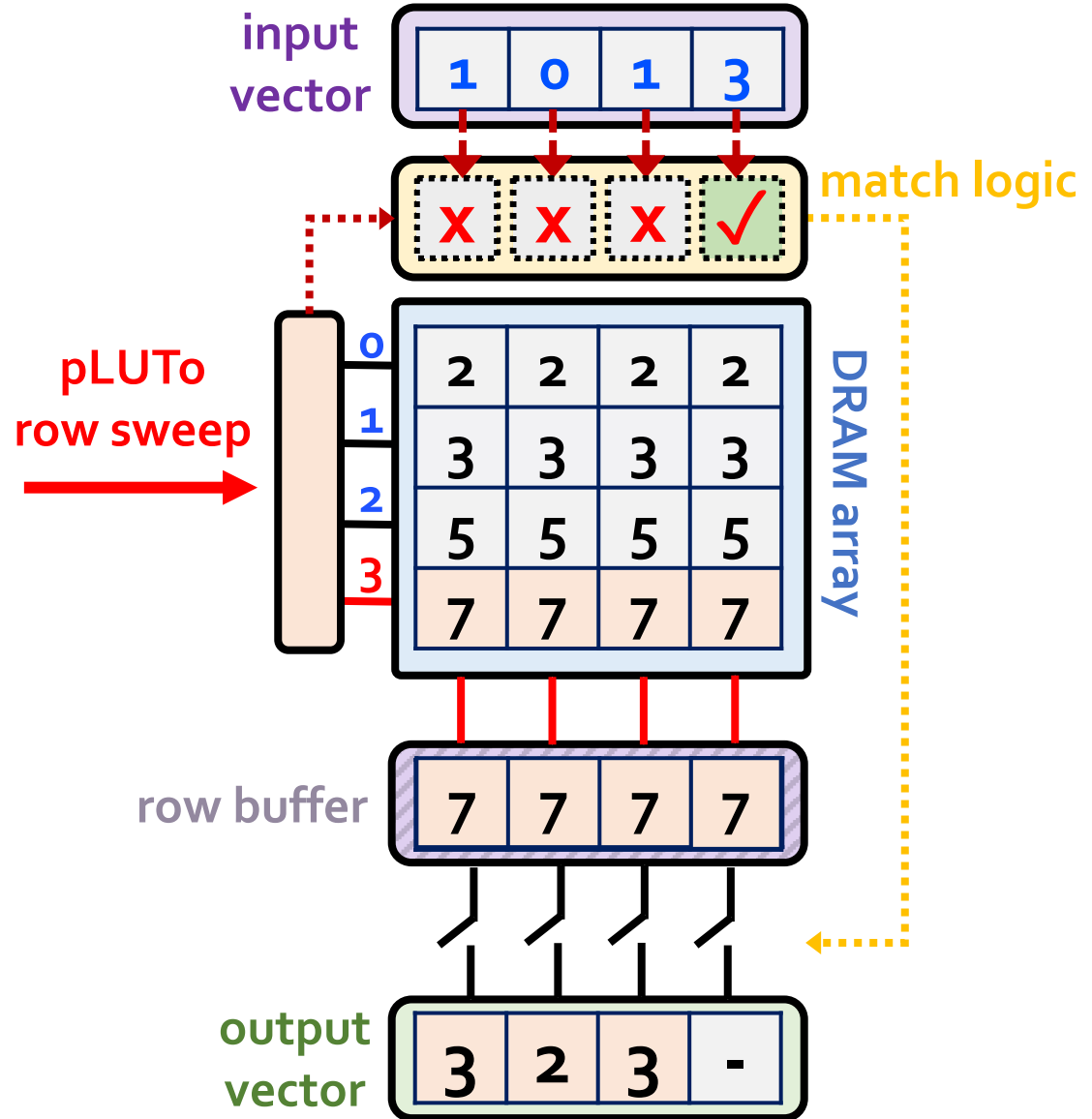
lookup table

1	0	1	3
---	---	---	---

input vector

3	2	3	7
---	---	---	---

output vector



In-DRAM pLUTo LUT Query: Step 4

LUT Query:
Return $\{2^{\text{nd}}, 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}\}$
prime numbers

		Prime numbers	
LUT index	i	f(i)	
0	1 st	2	
1	2 nd	3	
2	3 rd	5	
3	4 th	7	

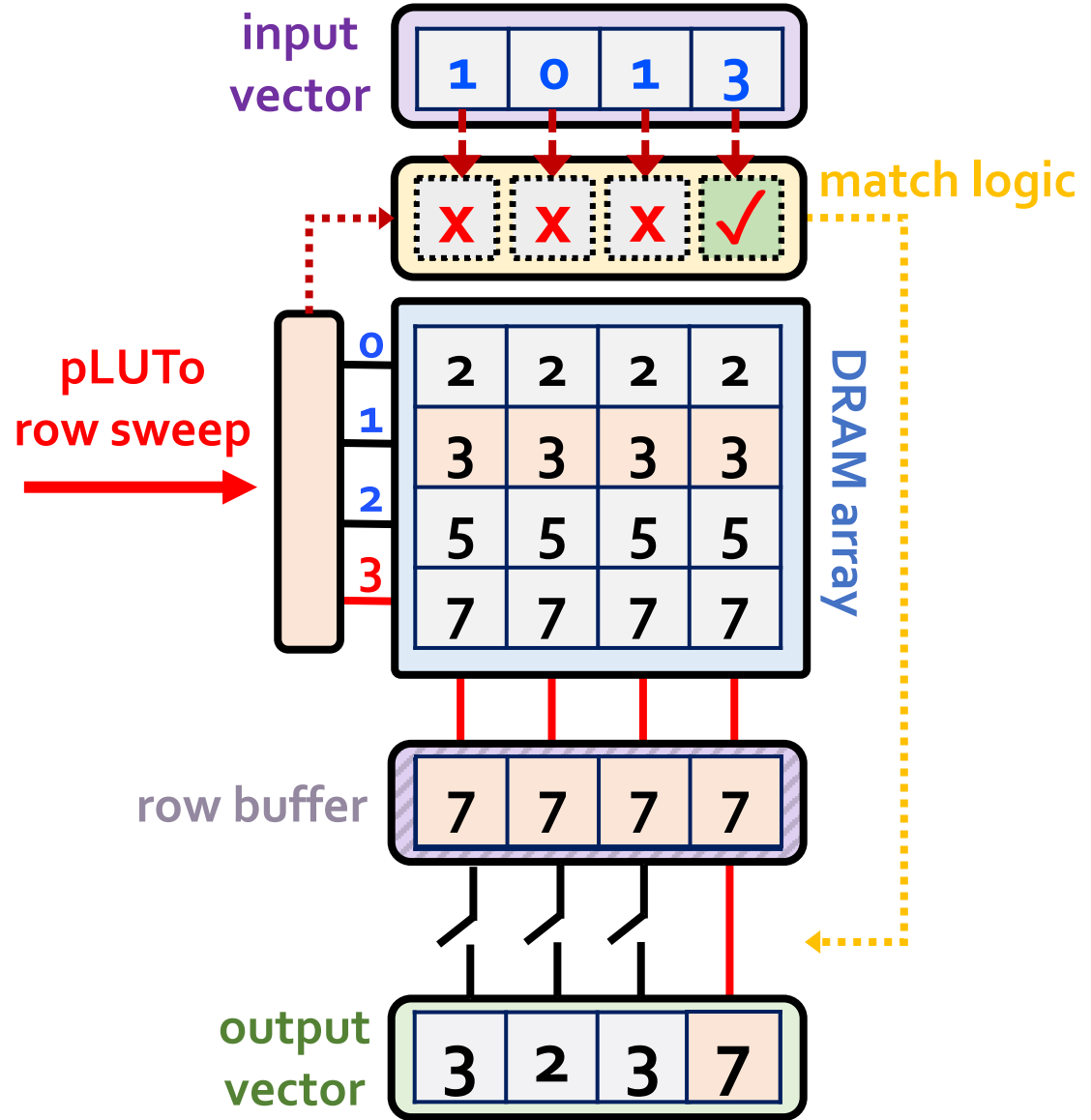
lookup table

1	0	1	3
---	---	---	---

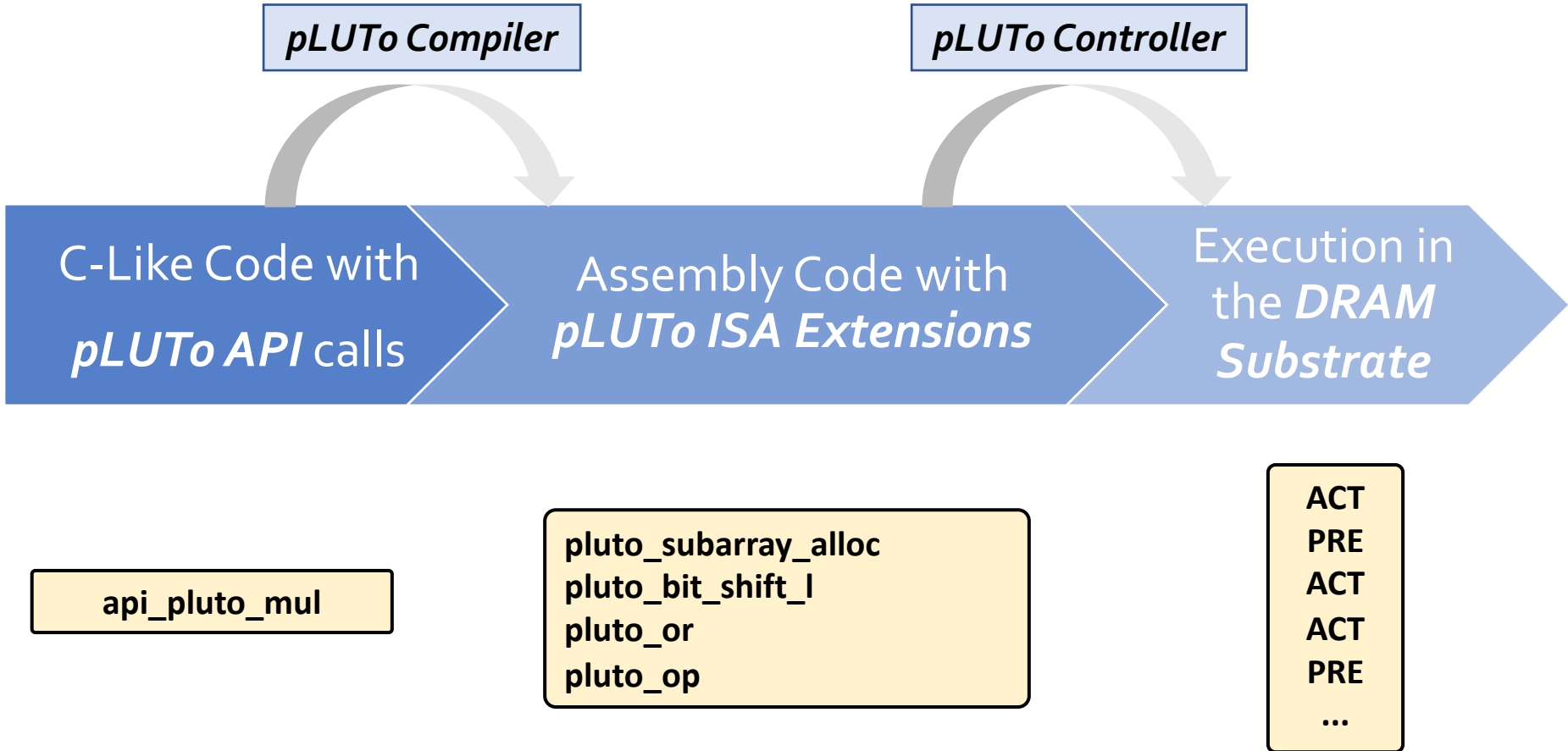
input vector

3	2	3	7
---	---	---	---

output vector

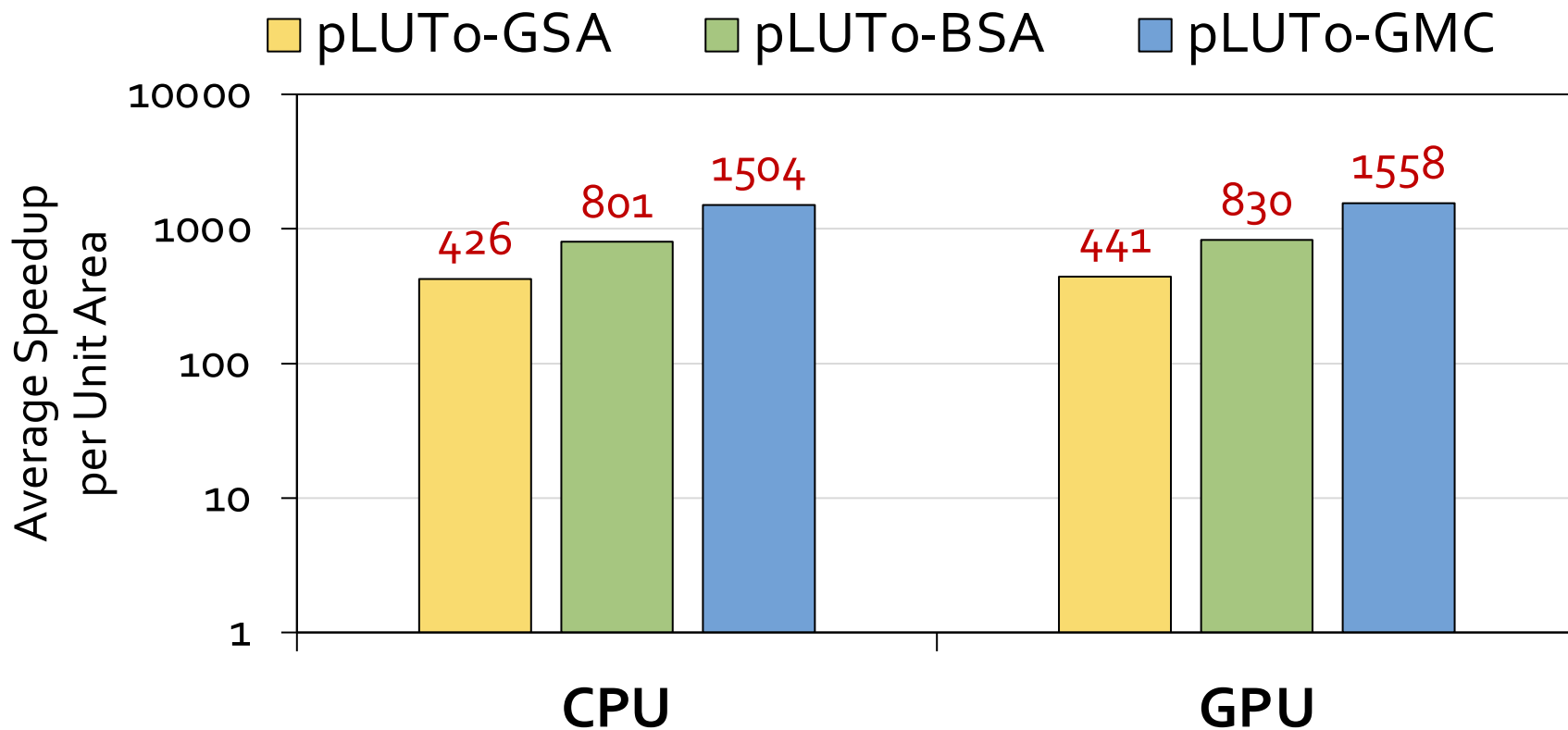


System Integration



Performance (normalized to area)

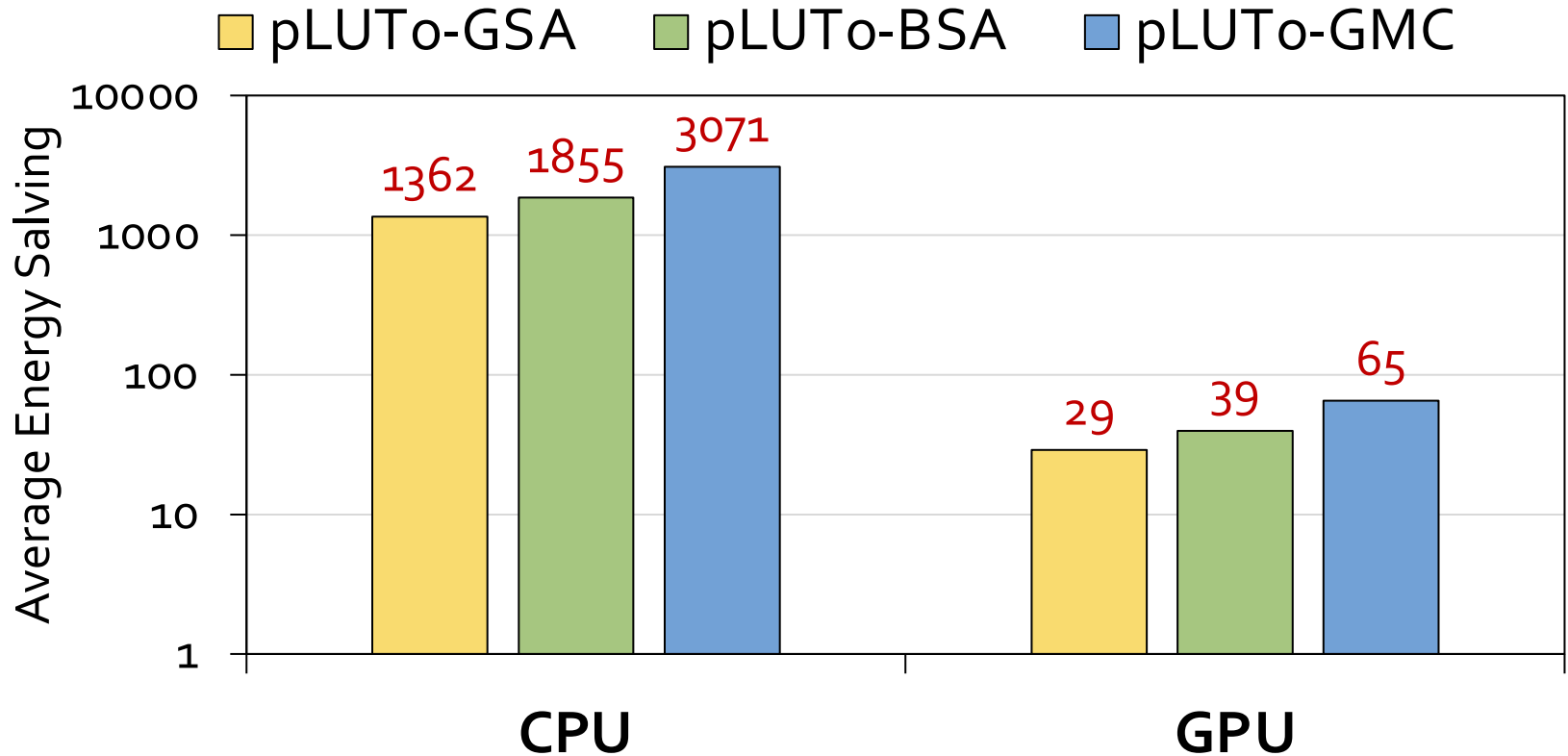
Average speedup normalized to area across 7 real-world workloads



pLUTo provides *substantially higher* performance per unit area than *both* the CPU and the GPU

Energy Consumption

Average energy consumption across 7 real-world workloads



pLUTo *significantly reduces energy consumption* compared to processor-centric architectures for various workloads

More Results in the Paper

- Comparison with FPGA
- Area Overhead Analysis
- Circuit-Level Reliability & Correctness
- Subarray-Level Parallelism
- LUT Loading Overhead
- Range of Supported Operations



pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables

João Dinis Ferreira[§]

Gabriel Falcao[†]

Juan Gómez-Luna[§]

Mohammed Alser[§]

Lois Orosa^{§∇}

Mohammad Sadrosadati[§]

Jeremie S. Kim[§]

Geraldo F. Oliveira[§]

Taha Shahroodi[‡]

Anant Nori^{*}

Onur Mutlu[§]

[§]ETH Zürich

[†]IT, University of Coimbra

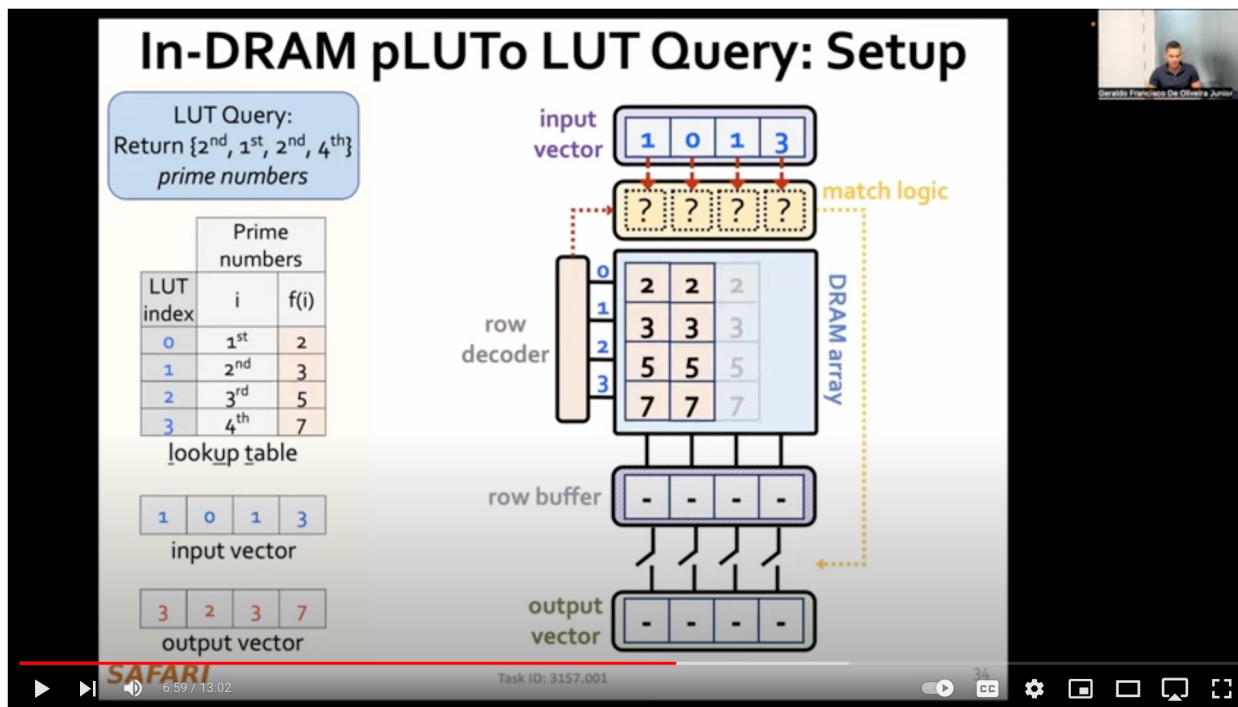
[∇]Galicia Supercomputing Center

[‡]TU Delft

^{*}Intel

SRC TECHCON Presentation

- Geraldo F. Oliveira
 - pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables
 - <https://arxiv.org/pdf/2104.07699.pdf>



pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables, SRC TECHCON 2023

Onur Mutlu Lectures
35.5K subscribers

Subscribed

17 | Share | Clip | Save

321 views 9 days ago
pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables
Speaker: Geraldo F. Oliveira ...more

Bulk Bitwise Operations in Real DRAM Chips

- Ismail Emir Yüksel, Yahya Can Tugrul Ataberk Olgun, F. Nisa Bostancı, A. Giray Yaglıkçı, Geraldo F. Oliveira, Haocong Luo, Juan Gómez-Luna, Mohammad Sadrosadati, Onur Mutlu, "**Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis**," *Proceedings of the 30th International Symposium on High-Performance Computer Architecture (HPCA)*, Edinburgh, Scotland, March 2024.

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

Ismail 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

The Capability of COTS DRAM Chips

We demonstrate that COTS DRAM chips:

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

In-DRAM True Random Number Generation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, "[D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput](#)"

Proceedings of the [25th International Symposium on High-Performance Computer Architecture \(HPCA\)](#), Washington, DC, USA, February 2019.

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

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

[[Full Talk Lecture Video](#) (27 minutes)]

Top Picks Honorable Mention by IEEE Micro.

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

Jeremie S. Kim^{‡§}

Minesh Patel[§]

Hasan Hassan[§]

Lois Orosa[§]

Onur Mutlu^{§‡}

[‡]Carnegie Mellon University

[§]ETH Zürich

In-DRAM True Random Number Generation

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

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

Ataberk Olgun^{§†}

Minesh Patel[§]

A. Giray Yağlıkçı[§]

Haocong Luo[§]

Jeremie S. Kim[§]

F. Nisa Bostanci^{§†}

Nandita Vijaykumar^{§⊙}

Oğuz Ergin[†]

Onur Mutlu[§]

[§]*ETH Zürich*

[†]*TOBB University of Economics and Technology*

[⊙]*University of Toronto*

In-DRAM True Random Number Generation

- F. Nisa Bostanci, Ataberk Olgun, Lois Orosa, A. Giray Yaglikci, Jeremie S. Kim, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
"DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators"
Proceedings of the 28th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, April 2022.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Short Talk Slides \(pptx\)](#)] [[pdf](#)]

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

F. Nisa Bostanci^{†§} Ataberk Olgun^{†§} Lois Orosa[§] A. Giray Yağlıkçı[§]
Jeremie S. Kim[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

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

Agenda

- Processing-Near-Memory Systems: Academia & Industry Developments
- **Invited Talk by Dr. Brian Schwedock:**
Architectures and Programming Models for General-Purpose Near-Data Computing
- Processing-Using-Memory Systems for Bulk Bitwise Operations
- Coffee Break
- Ataberk Olgun:
Infrastructure for Processing-Using-Memory Research
- **Invited Talk by Dr. Christina Giannoula:**
System Software and Libraries for Sparse Computational Kernels in PIM Architectures
- Nika Mansouri Ghiasi:
Storage-Centric Computing for Genomics and Metagenomics
- Research Challenges for PIM & Closing Remarks

In-DRAM Physical Unclonable Functions

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

The DRAM Latency PUF:

Quickly Evaluating Physical Unclonable Functions

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

Jeremie S. Kim^{†§}

Minesh Patel[§]

Hasan Hassan[§]

Onur Mutlu^{§†}

[†]Carnegie Mellon University

[§]ETH Zürich

In-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]

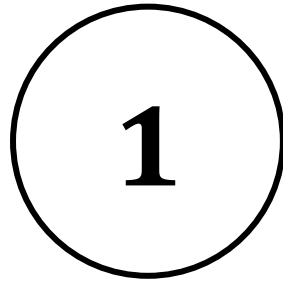
Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

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

NAND Flash Basics: A Flash Cell

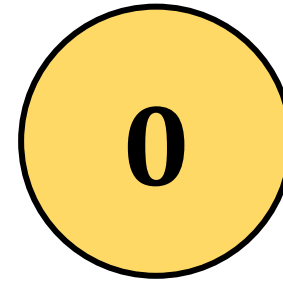
- A flash cell stores data by adjusting the **amount of charge** in the cell



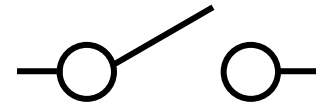
**Erased Cell
(Low Charge Level)**



*Operates as a **resistor***



**Programmed Cell
(High Charge Level)**

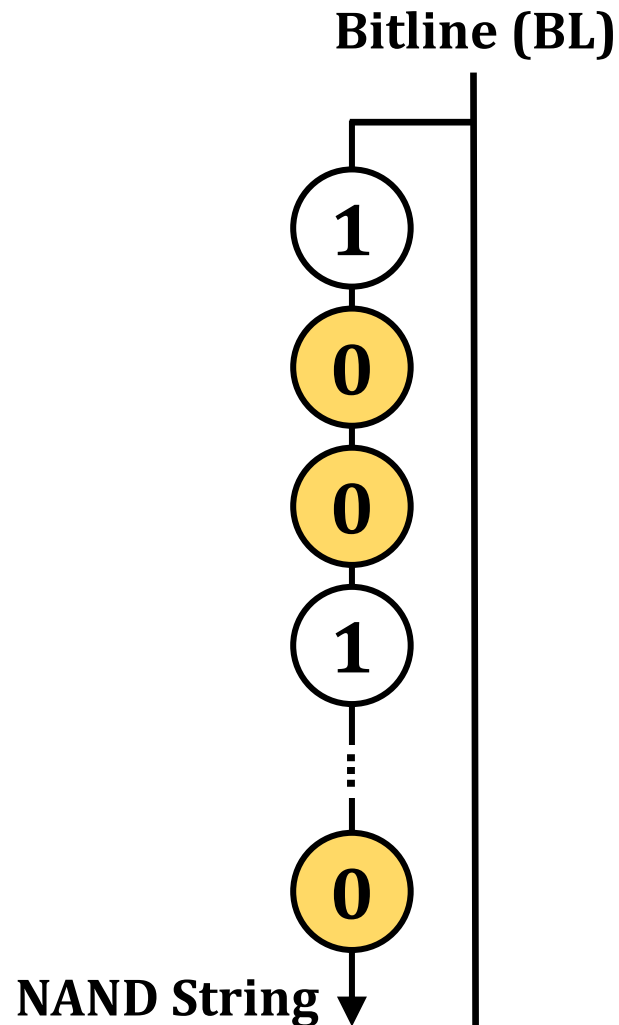


*Operates as an **open switch***

Activation

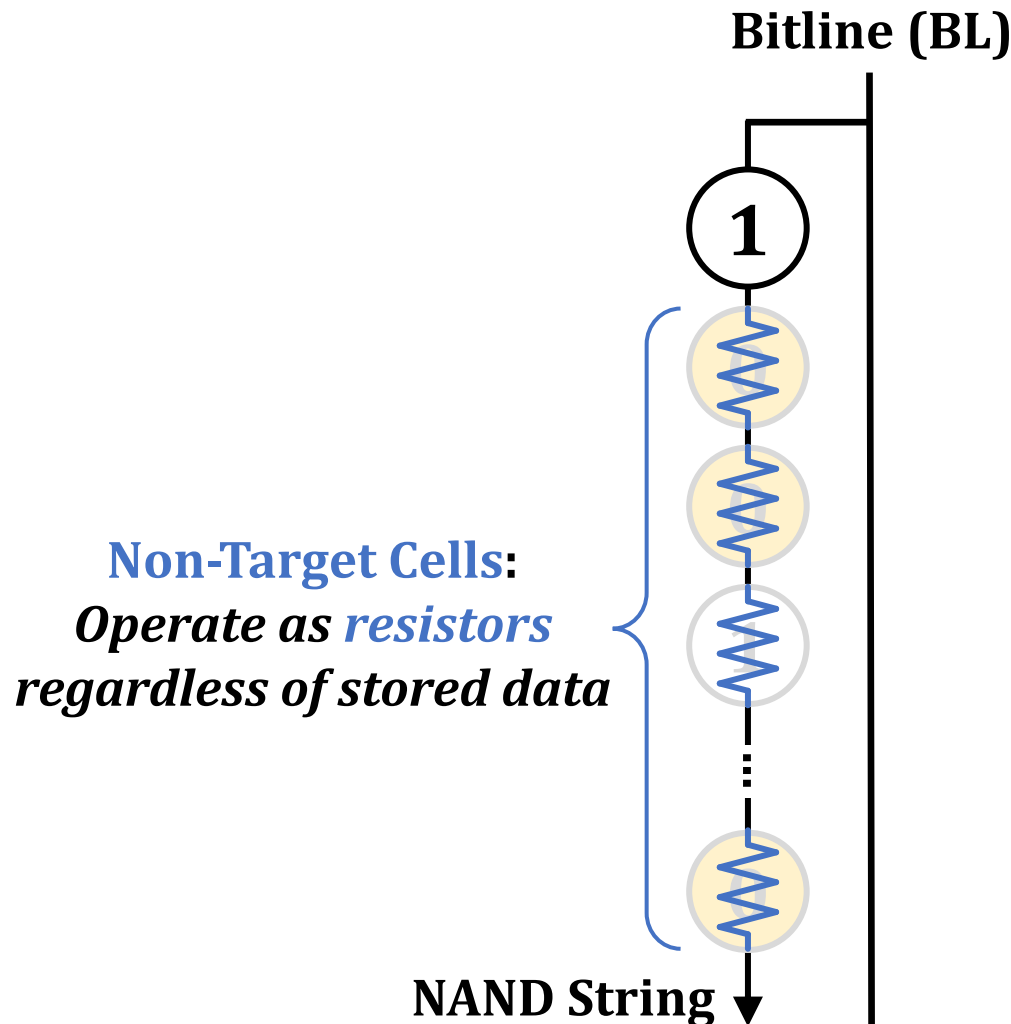
NAND Flash Basics: A NAND String

- A set of flash cells are **serially connected**, forming a **NAND string**



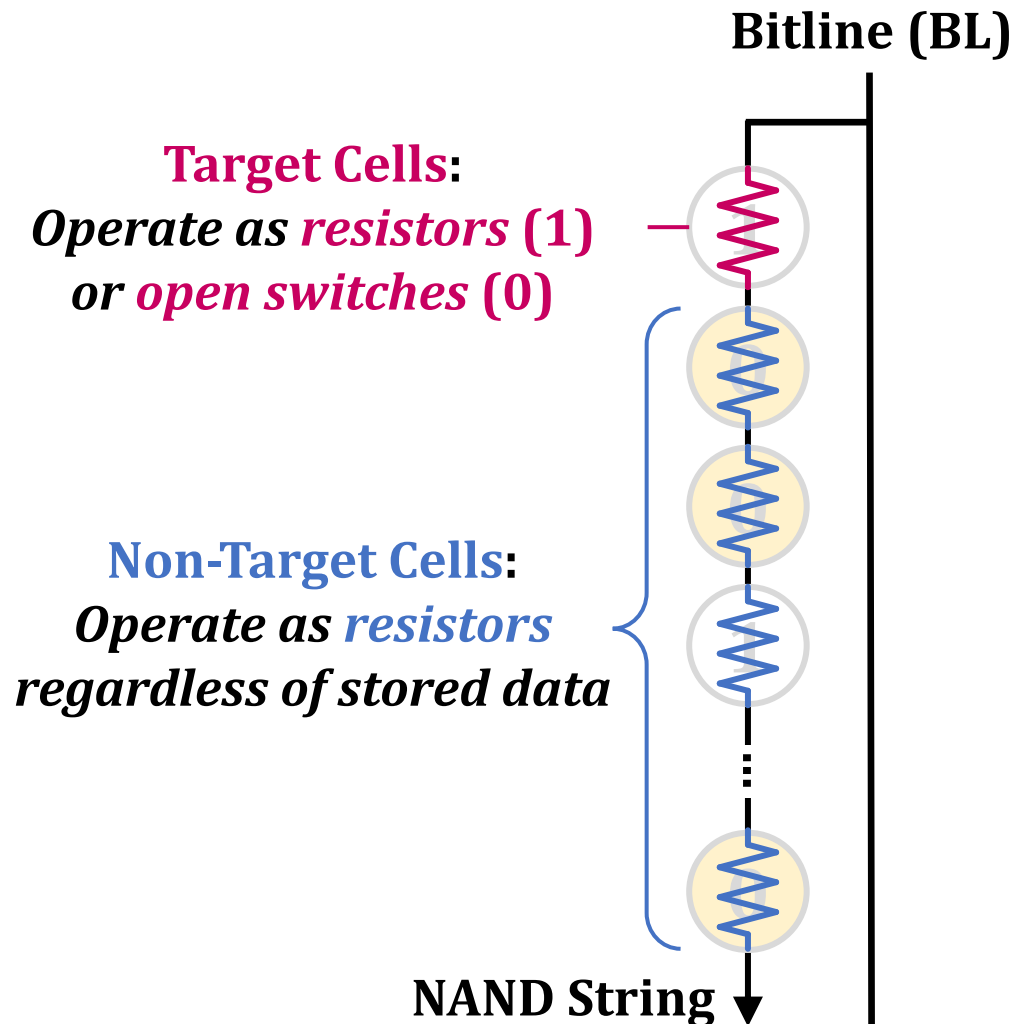
NAND Flash Basics: Read Mechanism

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



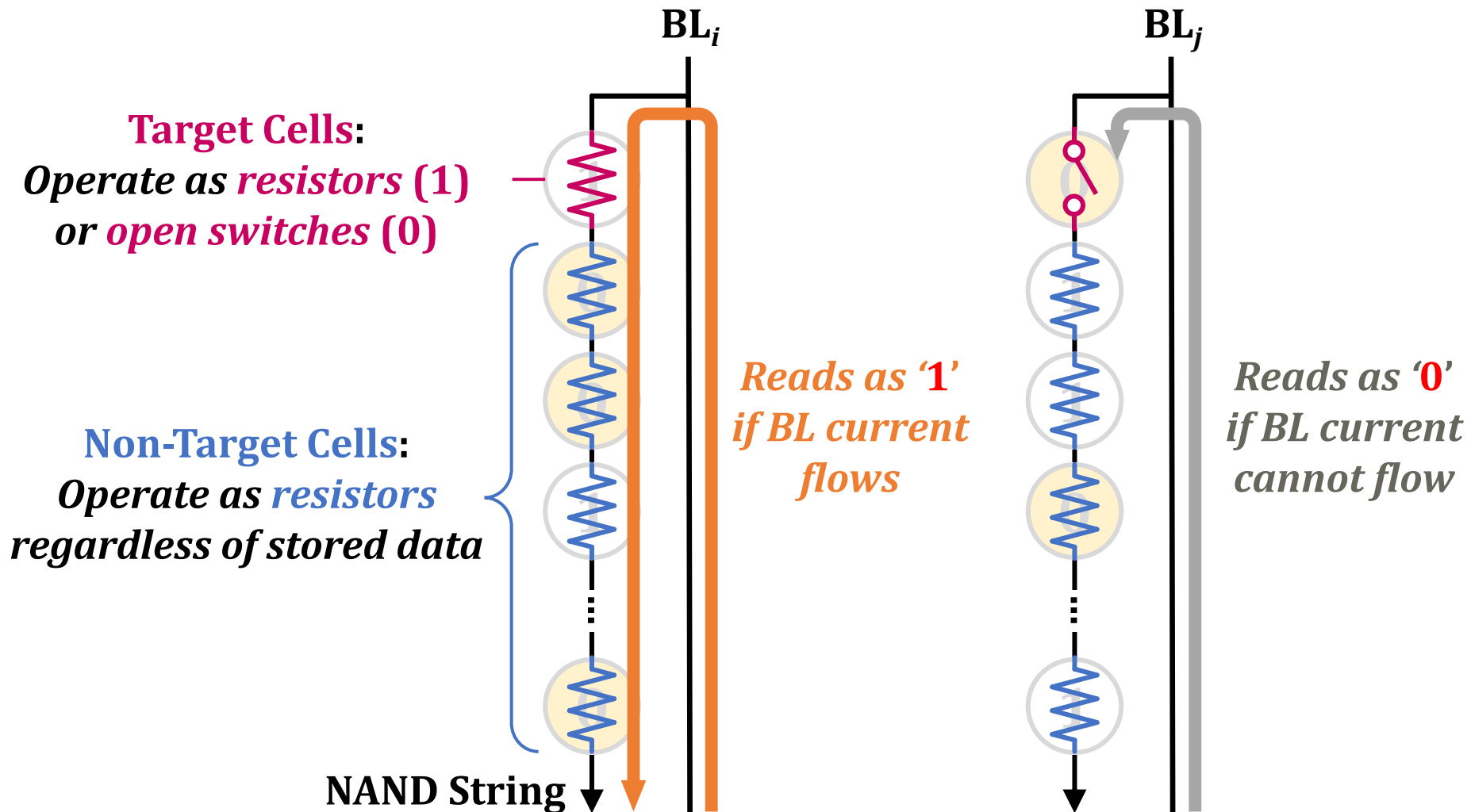
NAND Flash Basics: Read Mechanism

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



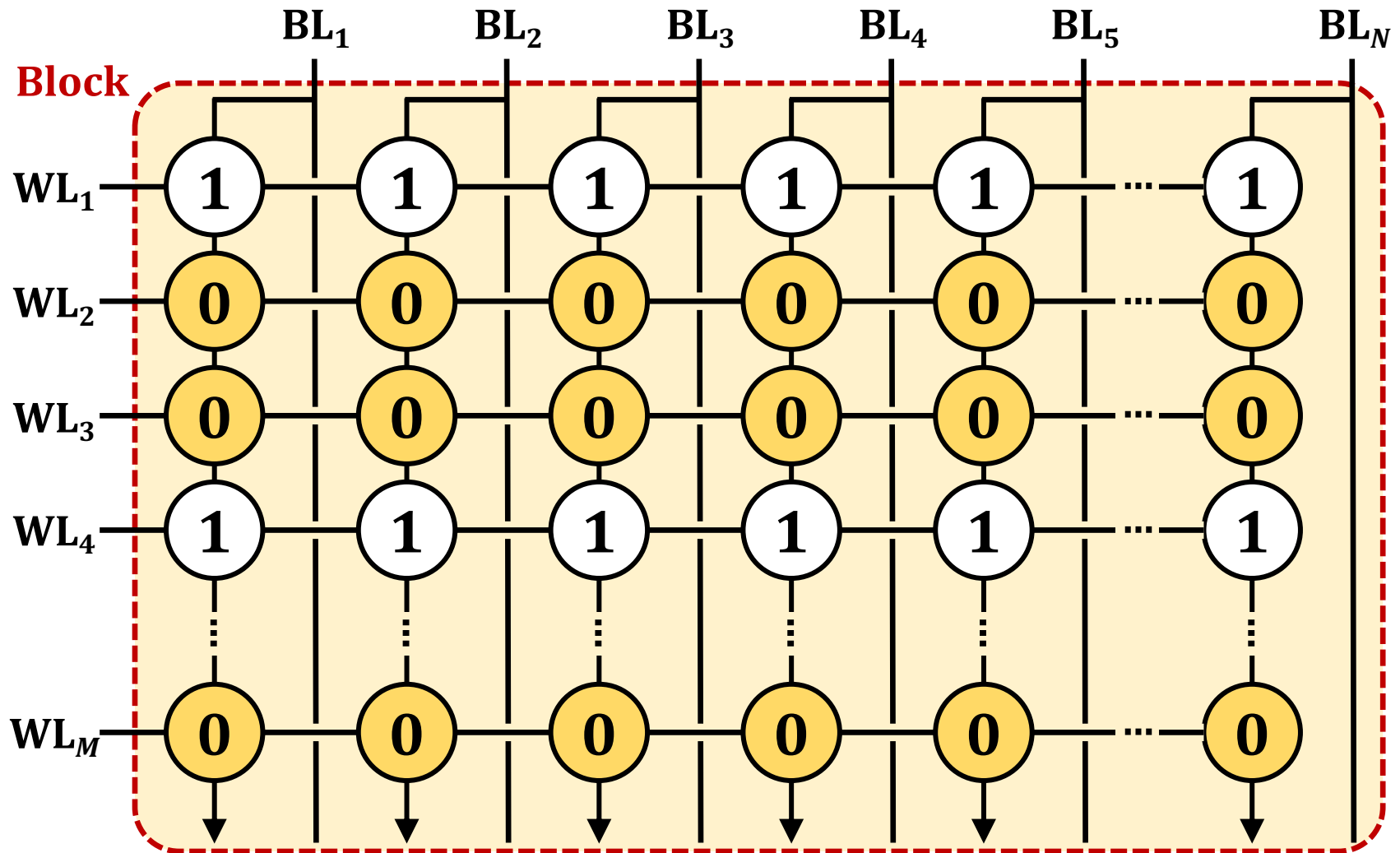
NAND Flash Basics: Read Mechanism

- NAND flash memory reads data by checking the bitline current



NAND Flash Basics: A NAND Flash Block

- NAND strings connected to different bitlines comprise a **block**





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



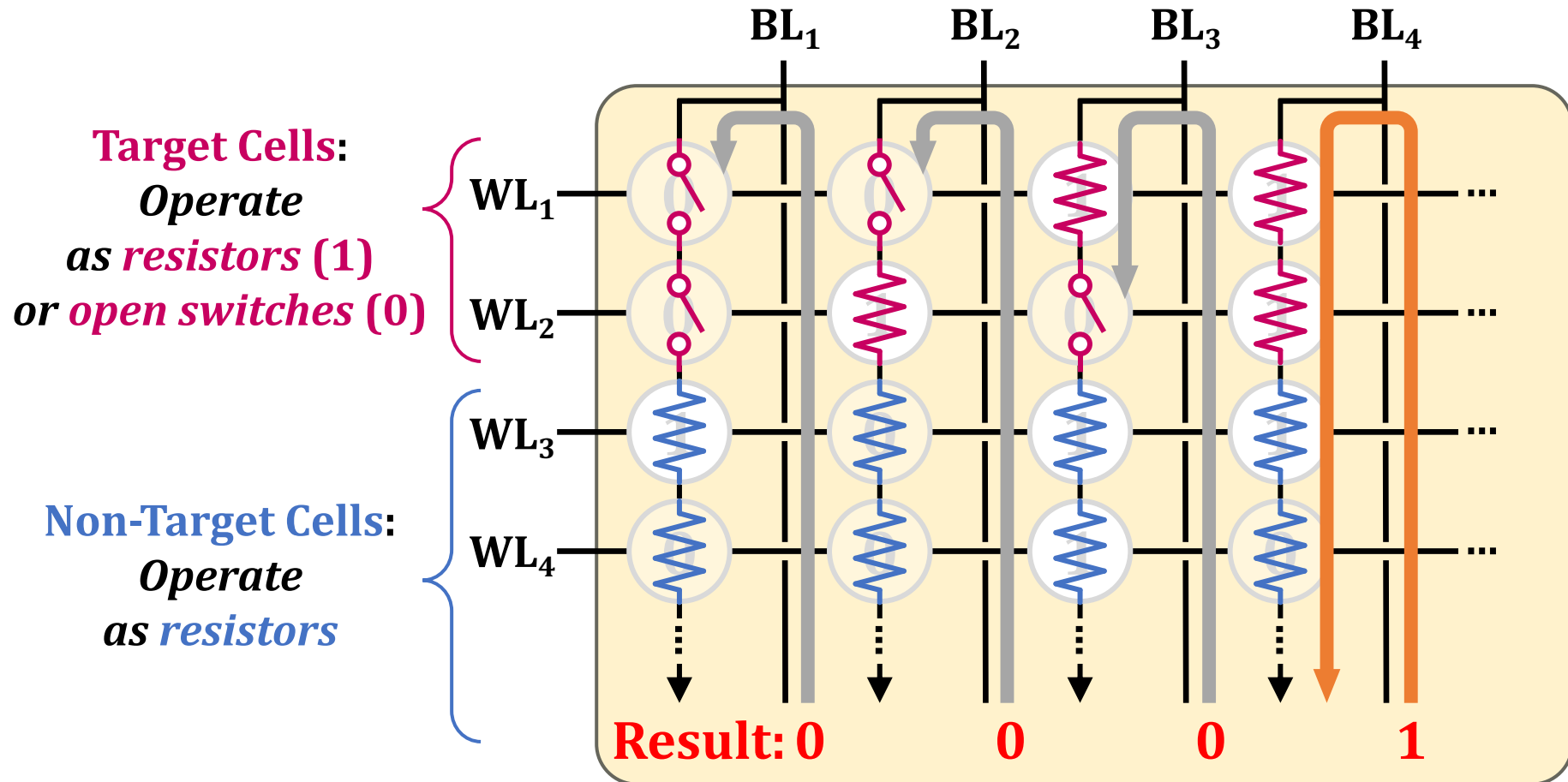
Enhanced SLC-Mode Programming (ESP)
to eliminate raw bit errors in stored data
(and thus in computation results)

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:

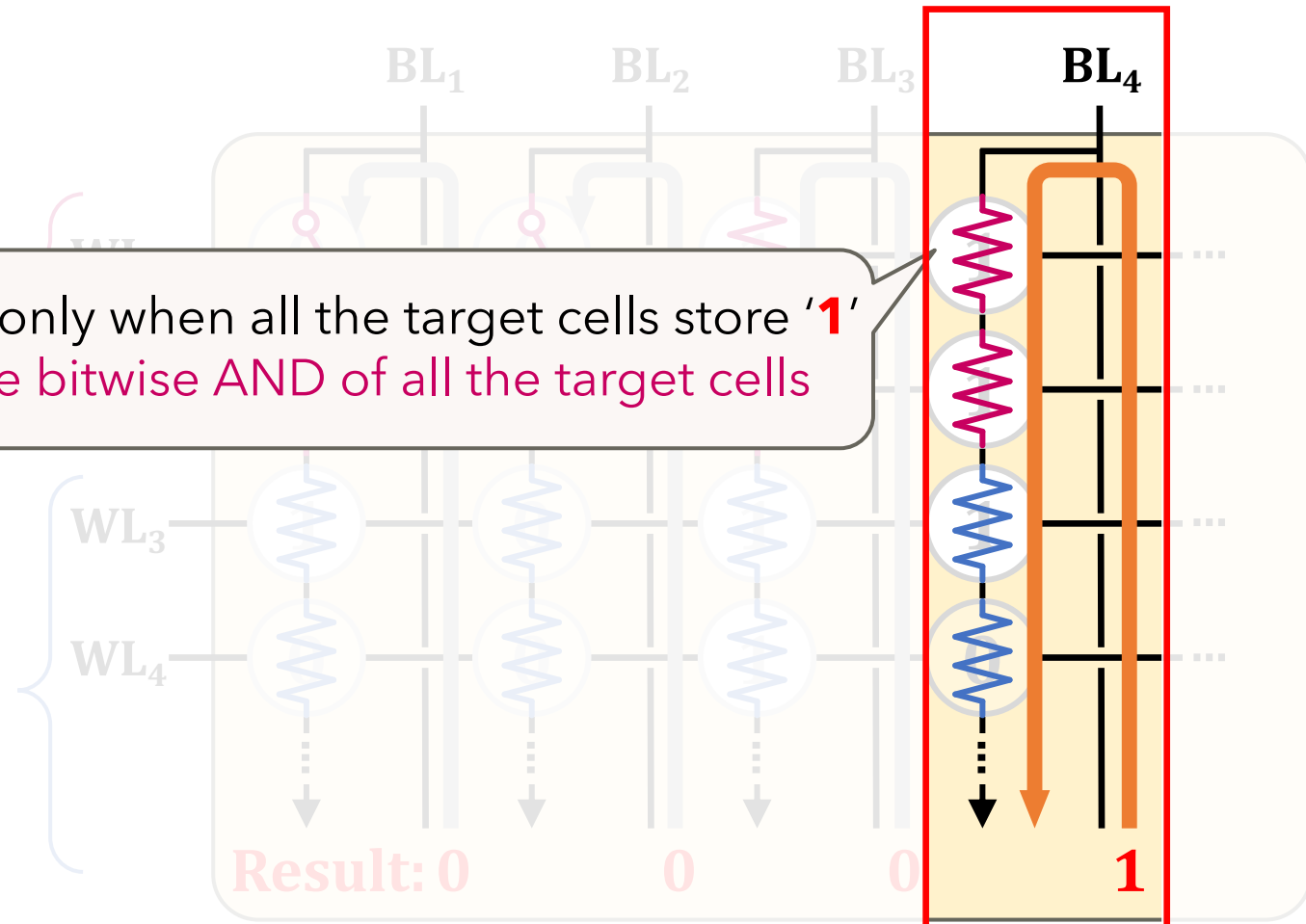
Simultaneously activates multiple WLs in the same block

→ Bitwise AND of the stored data in the WLs

Target Cell:

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

Non-Target Cell:
*Operate
as a resistance*

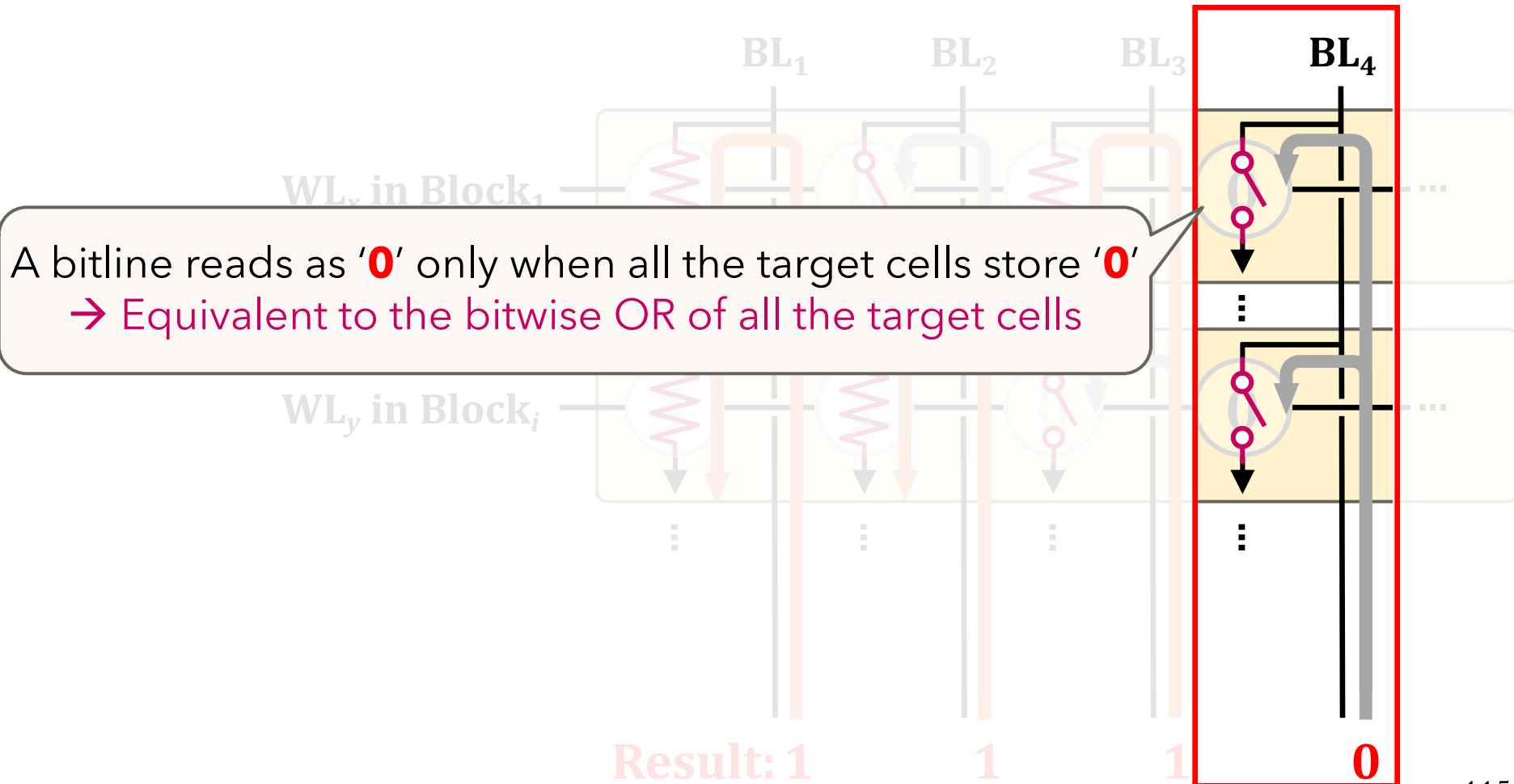


Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS:**

Simultaneously activates multiple WLs in different blocks

→ Bitwise OR of the stored data in the WLs

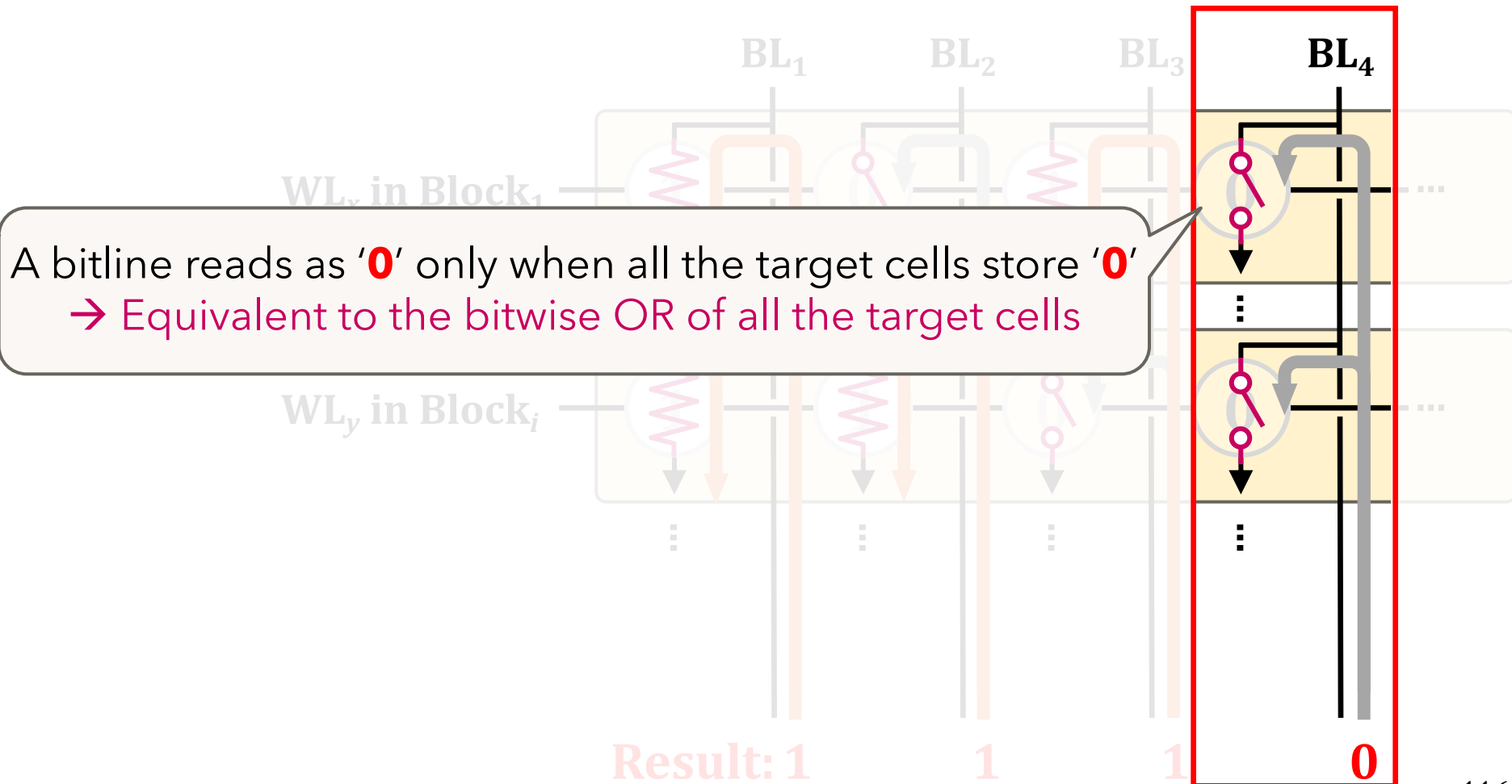


Multi-Wordline Sensing (MWS): Bitwise OR

▪ Inter-Block MWS:

Simultaneously activates multiple WLs in different blocks

→ Bitwise OR of the stored data in the WLs



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>

Summary: Flash-Cosmos



The first work that enables
in-flash multi-operand bulk bitwise operations
with a single sensing operation and high reliability



Improves performance
by 32x/25x/3.5x over OSP/ISP/ParaBit



Improves energy efficiency
by 95x/13.4x/3.3x over OSP/ISP/ParaBit



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

Pinatubo: RowClone and Bitwise Ops in PCM

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

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

University of California, Santa Barbara¹, Hewlett Packard Labs²

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

Pinatubo: RowClone and Bitwise Ops in PCM

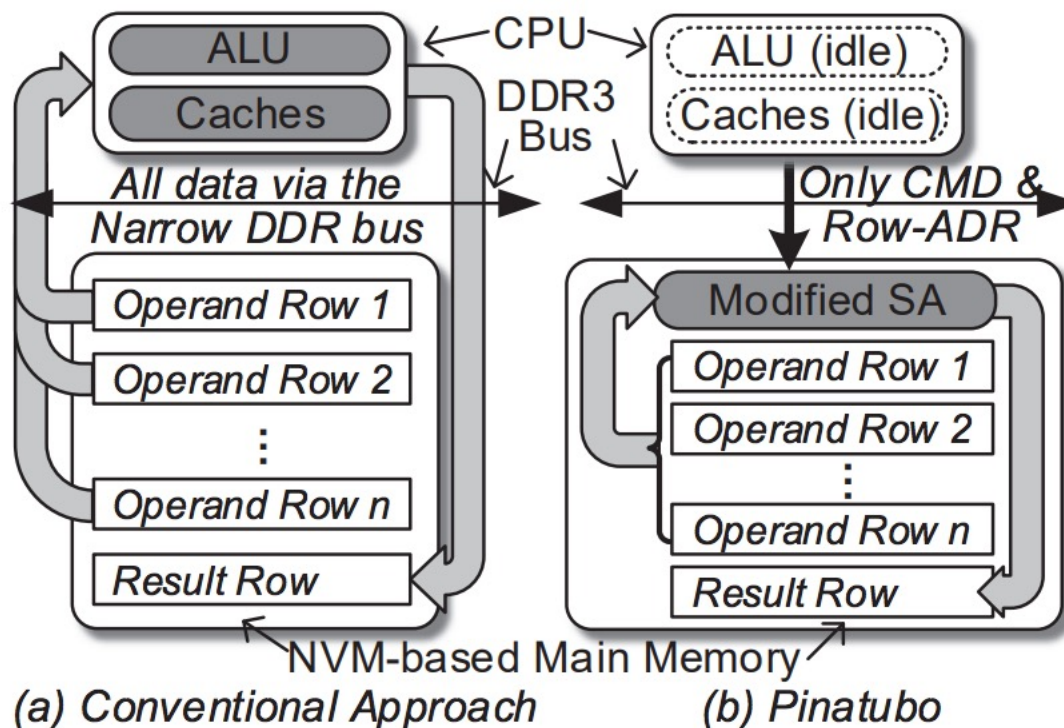


Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n -row bitwise operations inside NVM in one step.

Aside: In-Memory Crossbar Computation

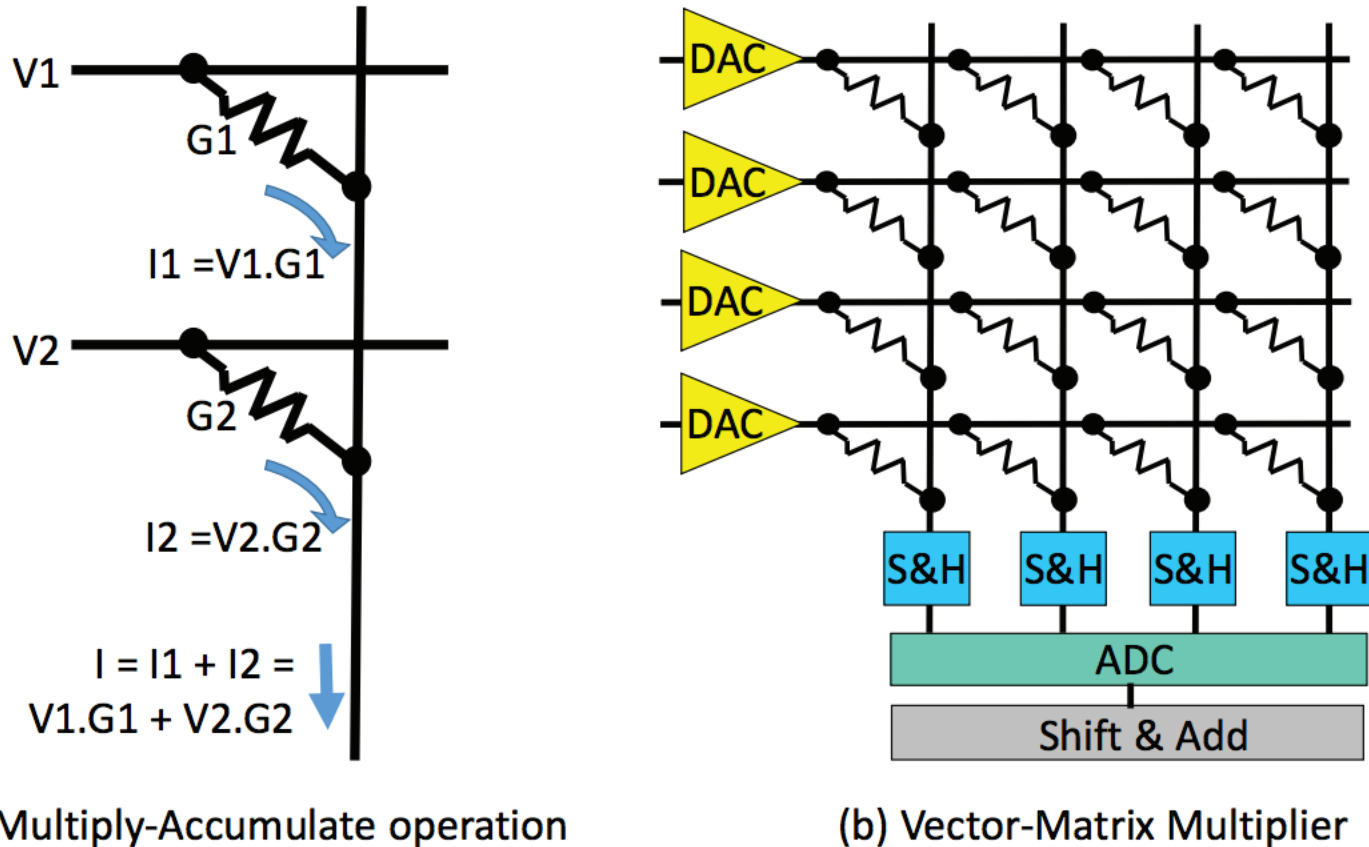
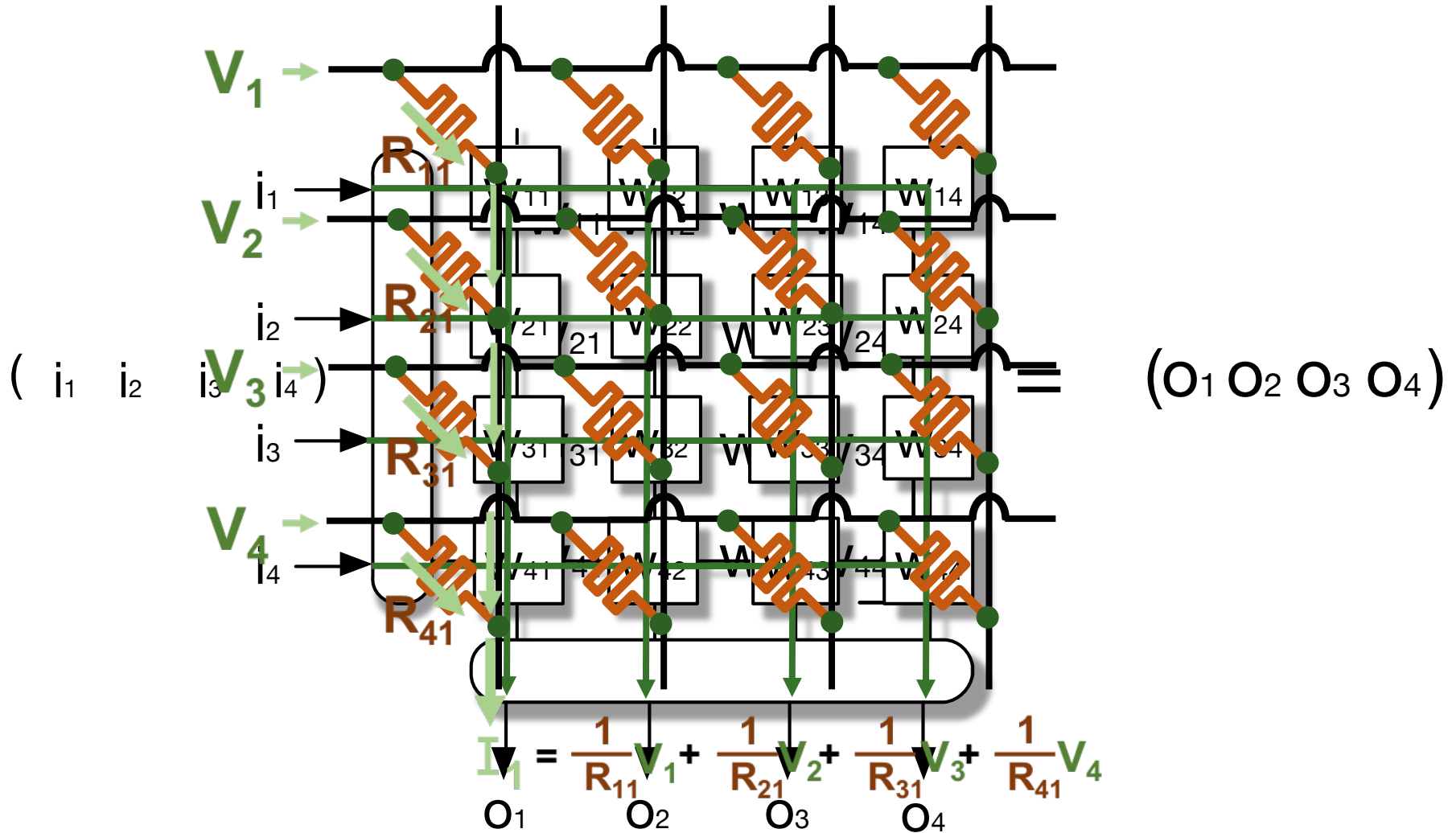


Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

Aside: In-Memory Crossbar Computation



Tutorial on Memory-Centric Computing: Processing-Using-Memory

Geraldo F. Oliveira

<https://geraldofojunior.github.io>

MICRO 2024

02 November 2024

Agenda

- Processing-Near-Memory Systems: Academia & Industry Developments
- **Invited Talk by Dr. Brian Schwedock:**
Architectures and Programming Models for General-Purpose Near-Data Computing
- Processing-Using-Memory Systems for Bulk Bitwise Operations
- **Coffee Break**
- Ataberk Olgun:
Infrastructure for Processing-Using-Memory Research
- **Invited Talk by Dr. Christina Giannoula:**
System Software and Libraries for Sparse Computational Kernels in PIM Architectures
- Nika Mansouri Ghiasi:
Storage-Centric Computing for Genomics and Metagenomics
- Research Challenges for PIM & Closing Remarks