2nd Workshop on Memory-Centric Computing: Processing-Using-Memory - Part I

Dr. Geraldo F. Oliveira

https://geraldofojunior.github.io

ICS 2025

08 June 2025

SAFARI

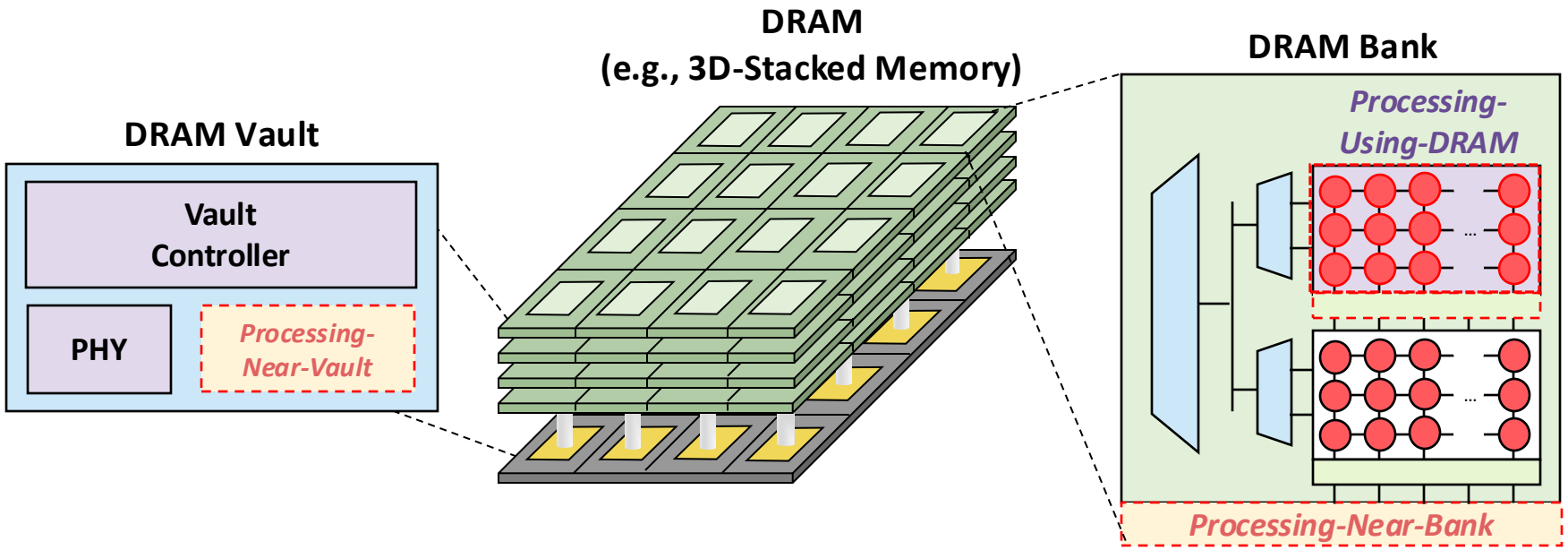
ETH zürich

Processing-in-Memory:
Overview

Two main approaches for Processing-in-Memory:

1. **Processing-Near-Memory**: PIM logic is added to the same die as memory or to the logic layer of 3D-stacked memory

2. **Processing-Using-Memory**: uses the operational principles of memory cells to perform computation

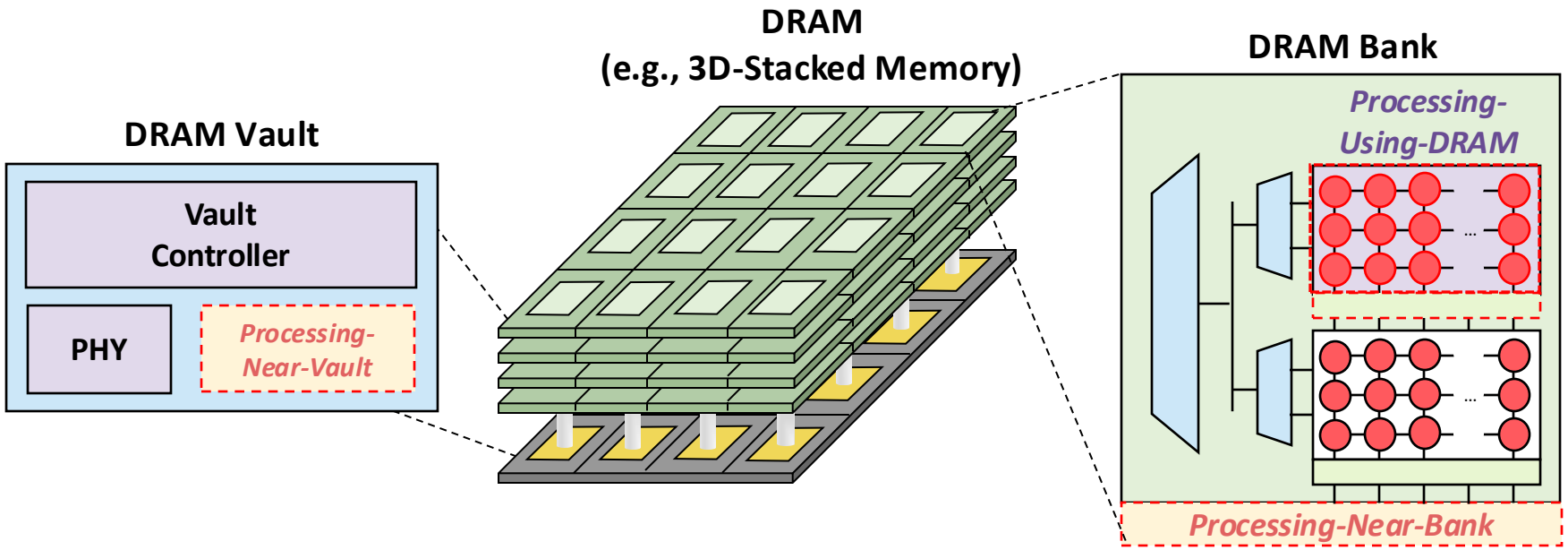


Processing-in-Memory:
Overview

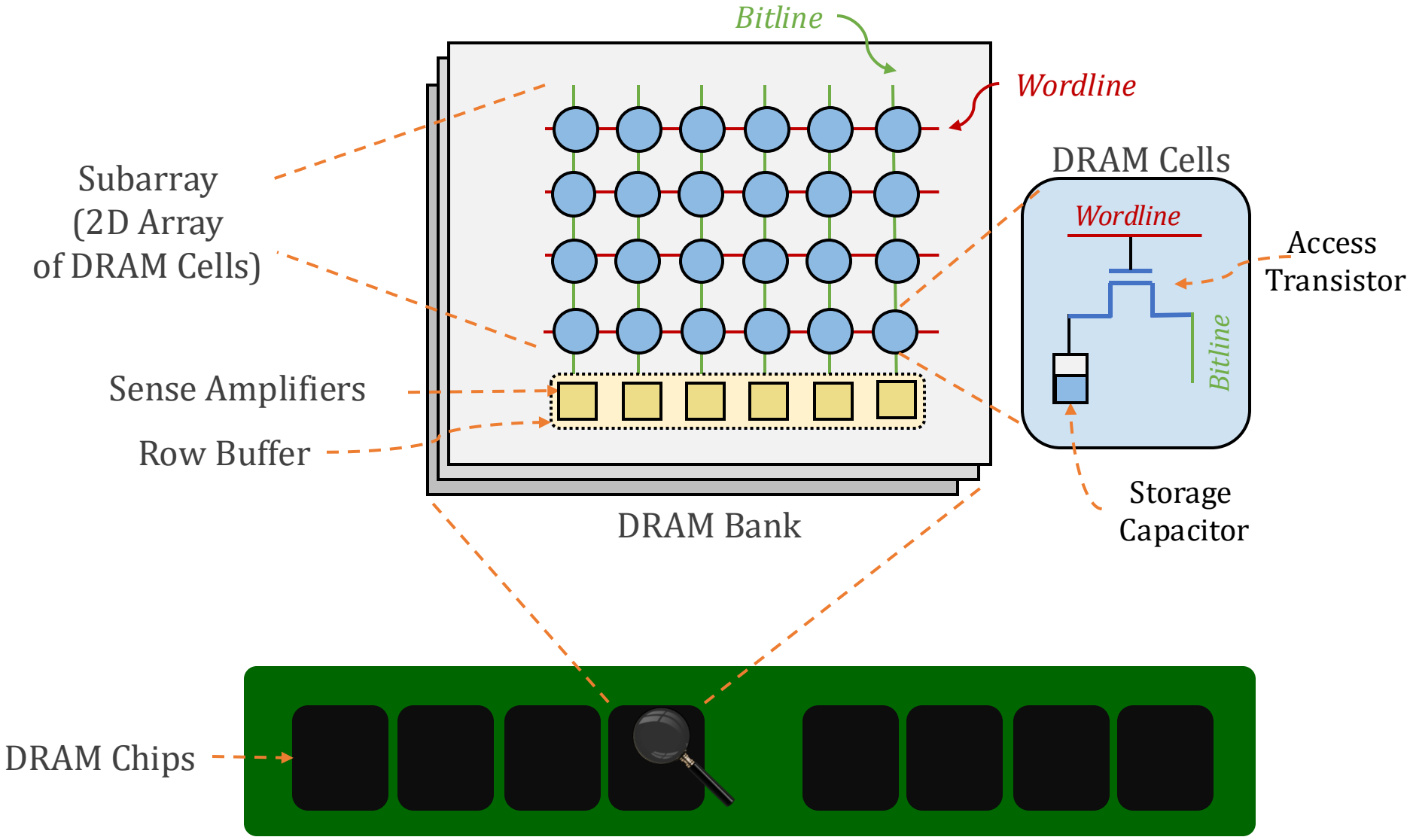
Two main approaches for Processing-in-Memory:

1. **Processing-Near-Memory**: PIM logic is added to the same die as memory or to the logic layer of 3D-stacked memory

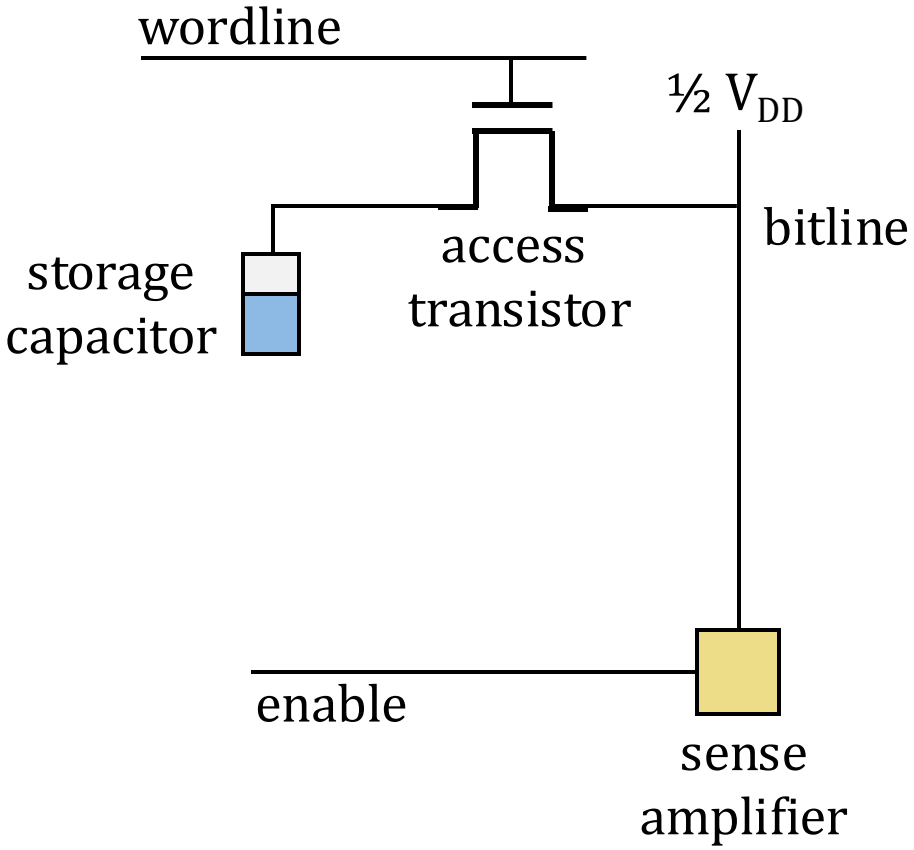
2. **Processing-Using-Memory**: uses the operational principles of memory cells to perform computation



Inside a DRAM Chip



DRAM Cell Operation

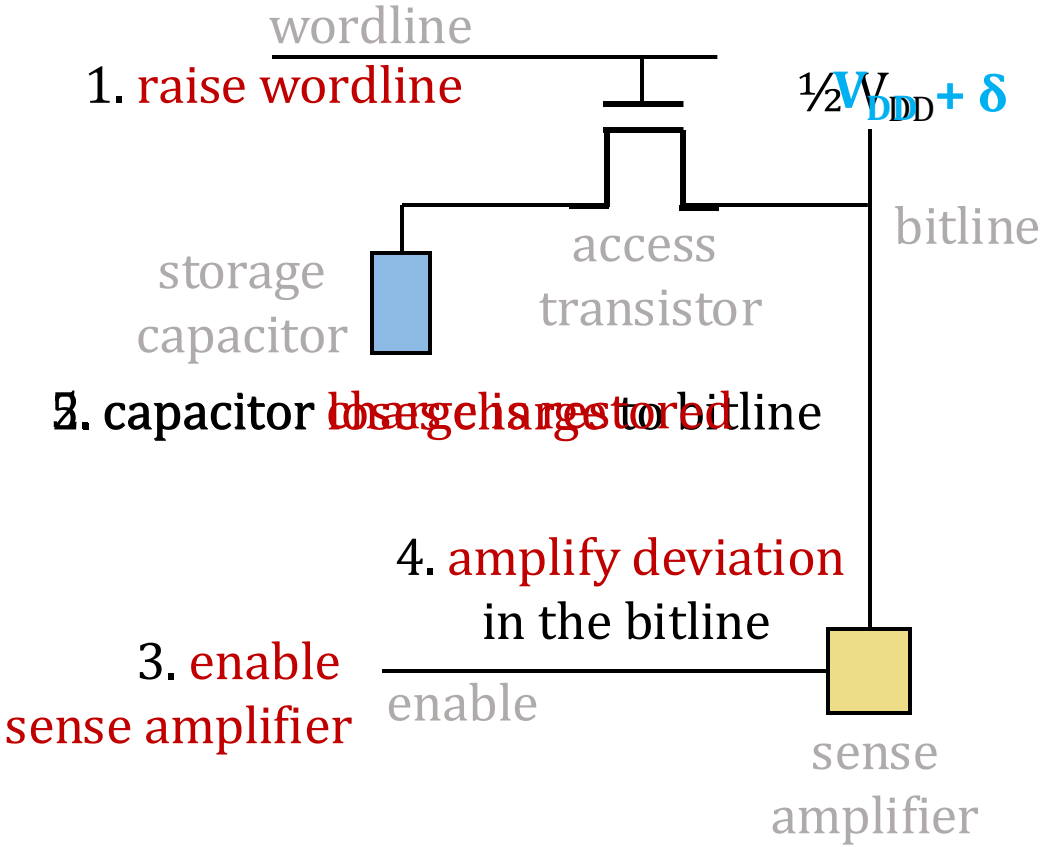


1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

DRAM Cell Operation (1/3)

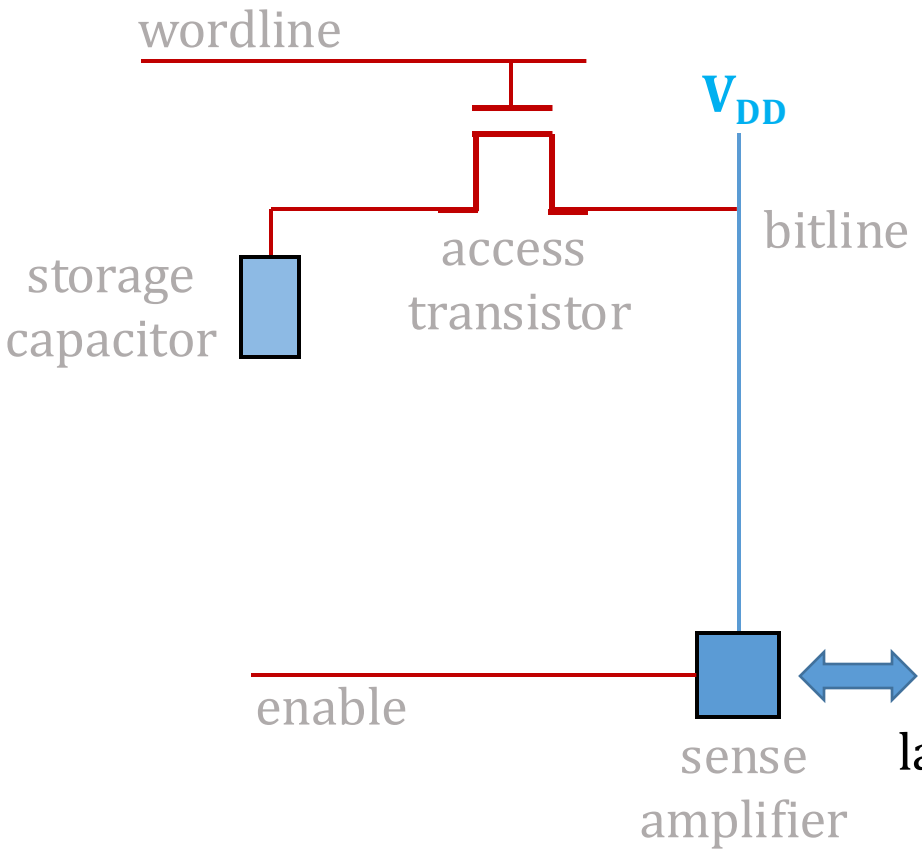


1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

DRAM Cell Operation (2/3)



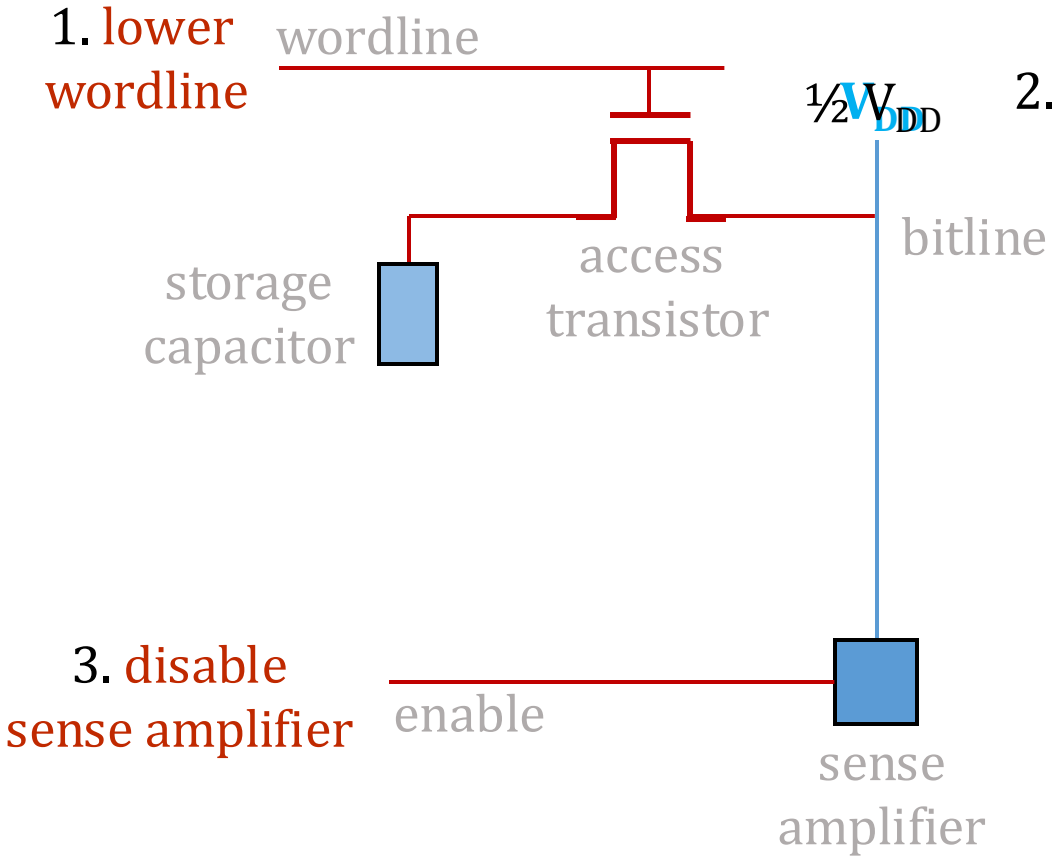
1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

read/write charge latched in sense amplifier

DRAM Cell Operation (3/3)



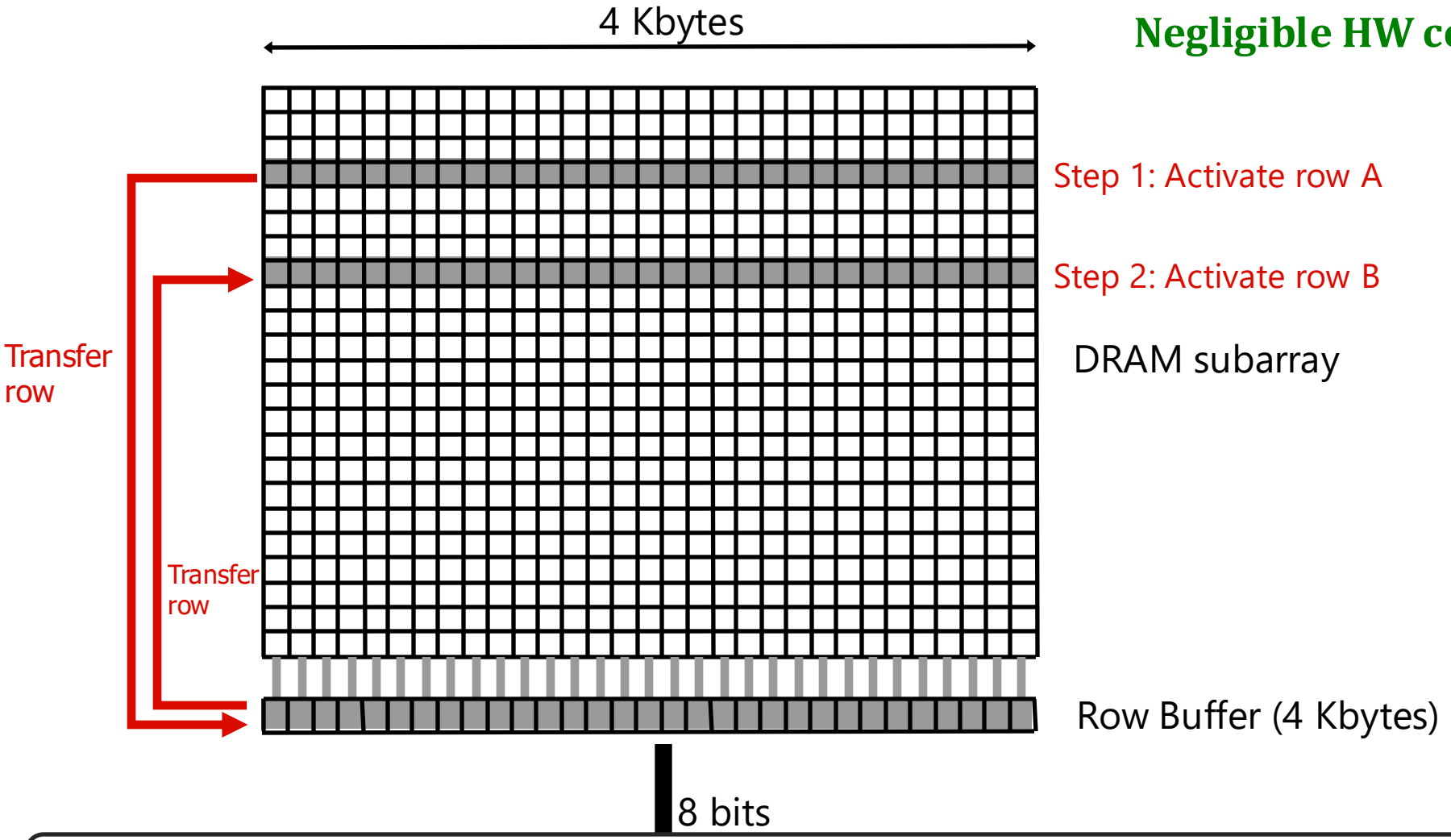
1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

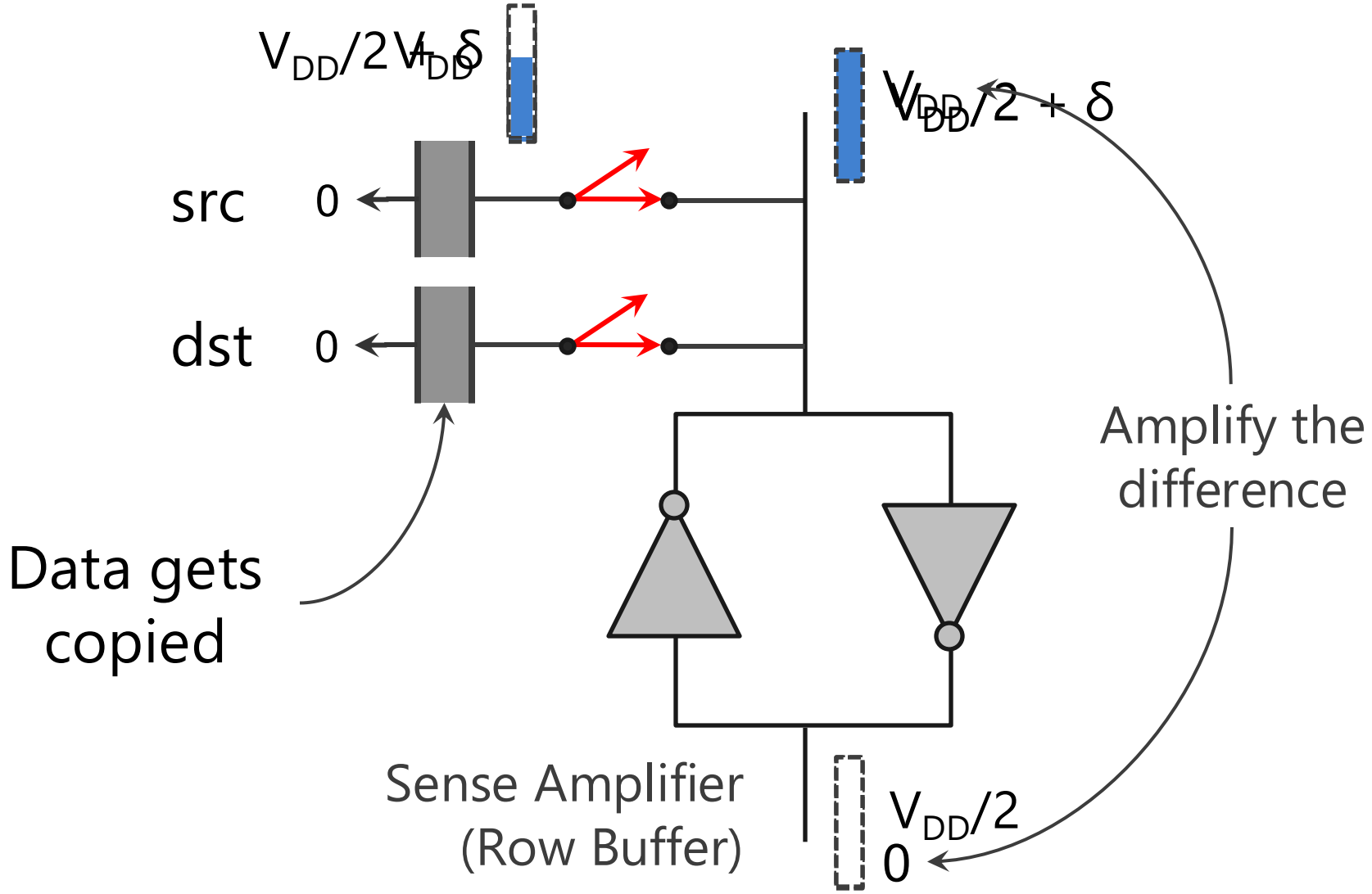
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates
Negligible HW cost

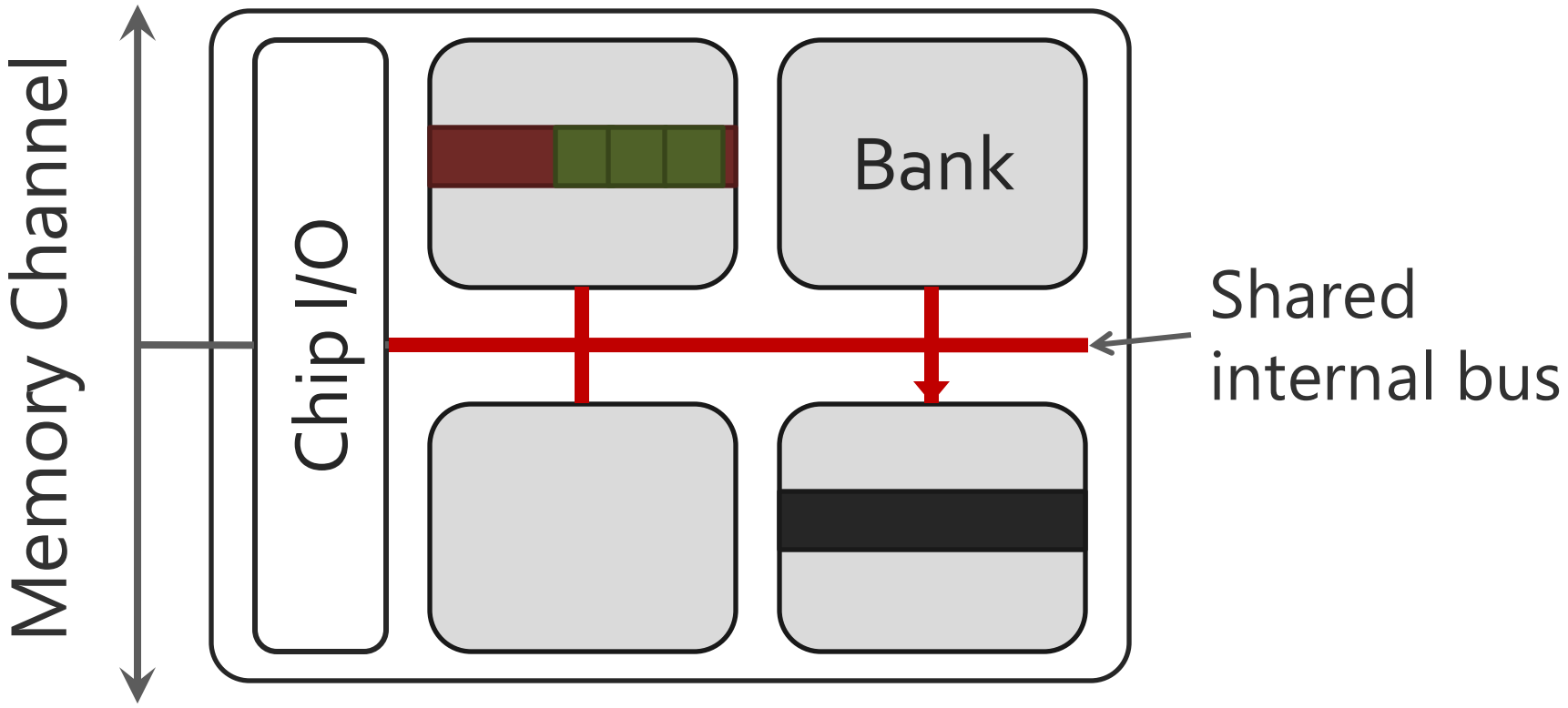


11.6X latency reduction, 74X energy reduction

RowClone: Intra-Subarray



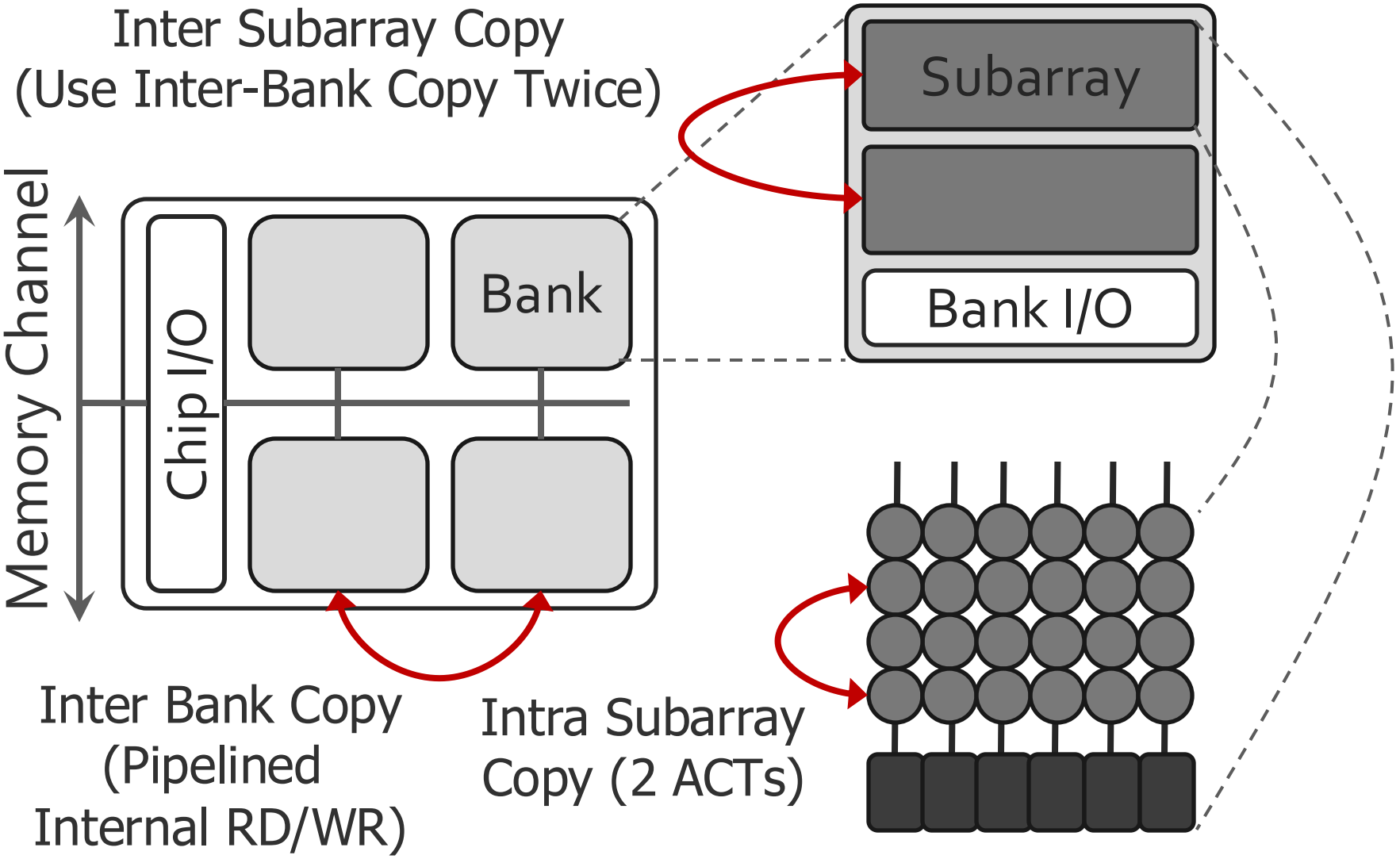
RowClone: Inter-Bank



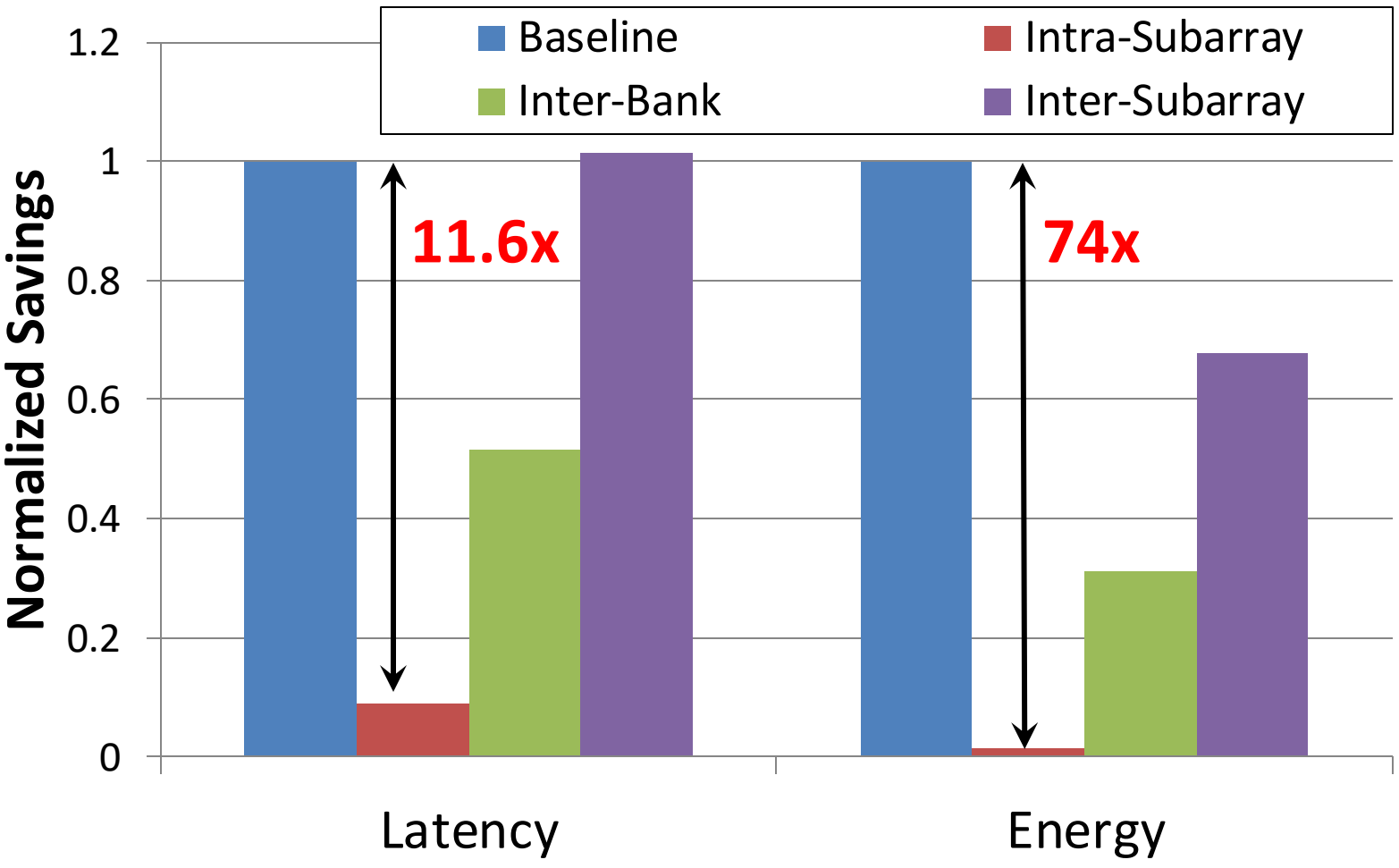
Overlap the latency of the read and the write
1.9X latency reduction, 3.2X energy reduction

Generalized RowClone

0.01% area cost



RowClone: Latency and Energy Savings



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

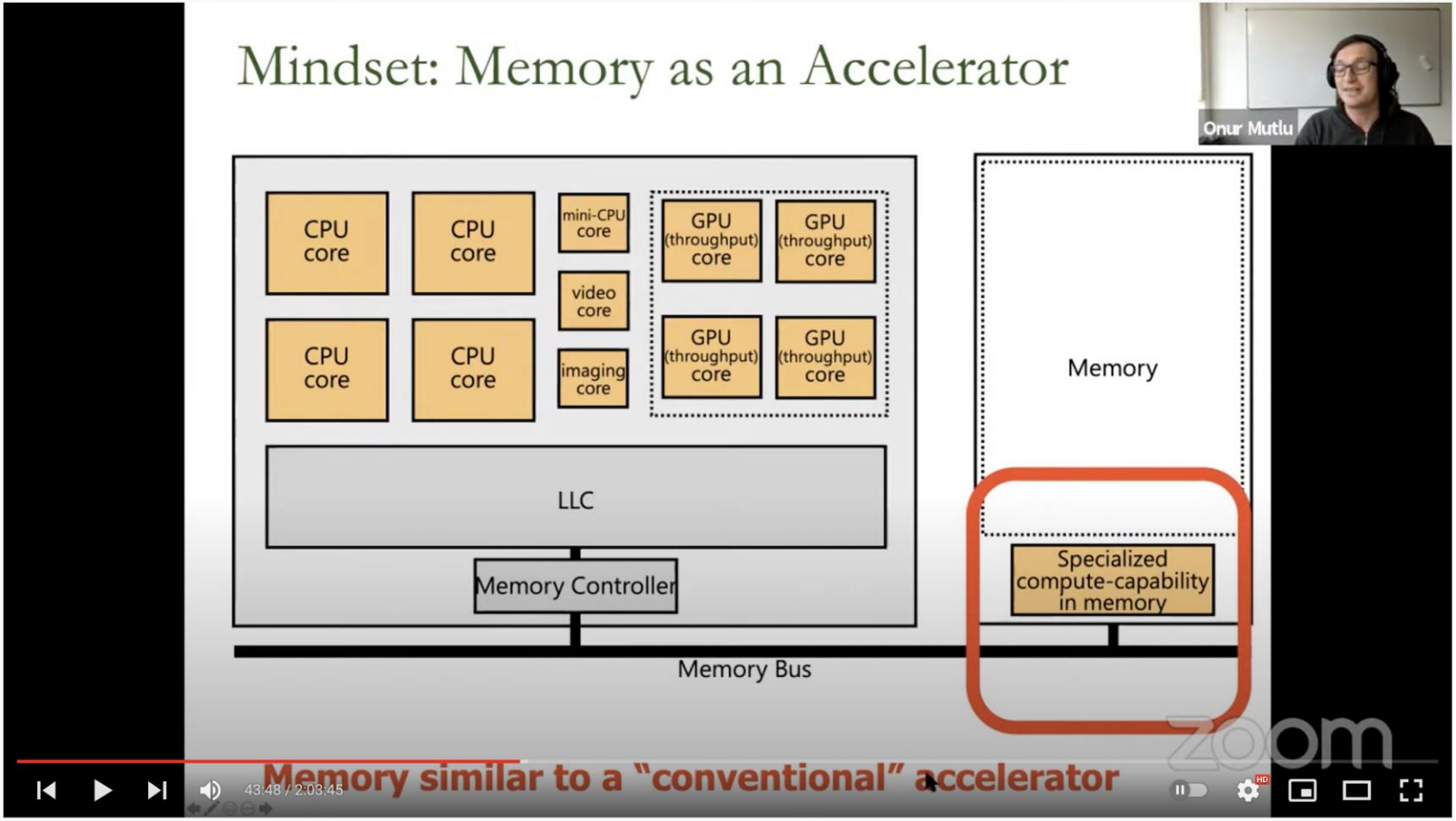
More on RowClone

- 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 vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University †Intel Pittsburgh

Lecture on RowClone & Processing using DRAM



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

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]
  - Yes, see SIMDRAM [Hajinazar & Oliveira, ASPLOS 2021]

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***), 2020.

NoM: Network-on-Memory for Inter-bank Data Transfer in Highly-banked Memories

Seyyed Hossein SeyyedAghaei Rezaei[1] Mehdi Modarressi[1,3] Rachata Ausavarungnirun[2]
Mohammad Sadrosadati[3] Onur Mutlu[4] Masoud Daneshtalab[5]

[1]University of Tehran [2]King Mongkut's University of Technology North Bangkok [3]Institute for Research in Fundamental Sciences
[4]ETH Zürich [5]Mälardalens University

LISA: Increasing Connectivity in DRAM

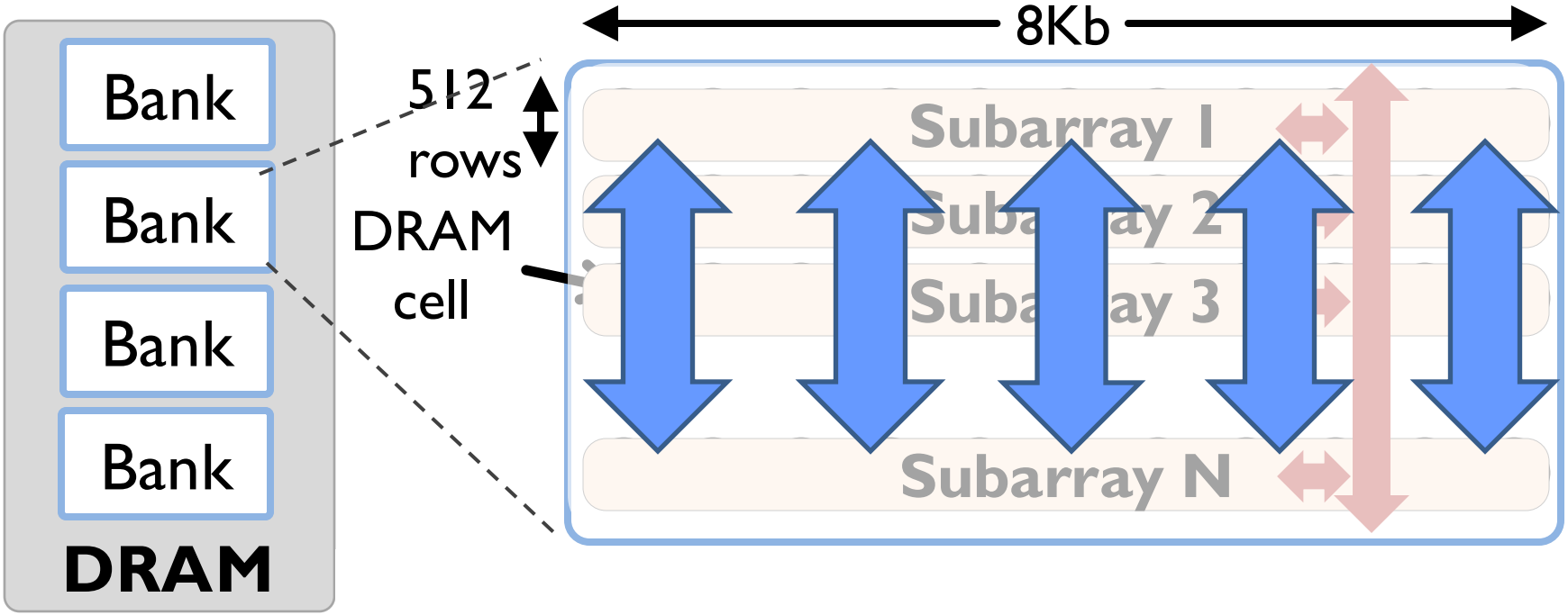
- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
  "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
  *Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**)*, 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*

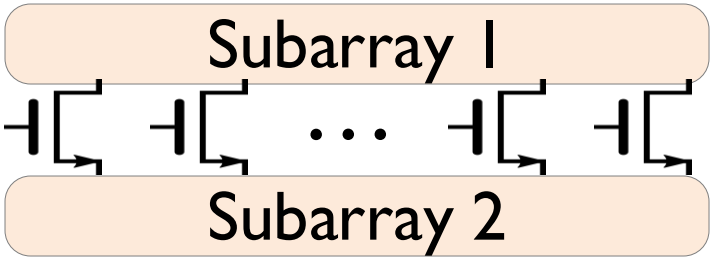
Moving Data Inside DRAM?



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]

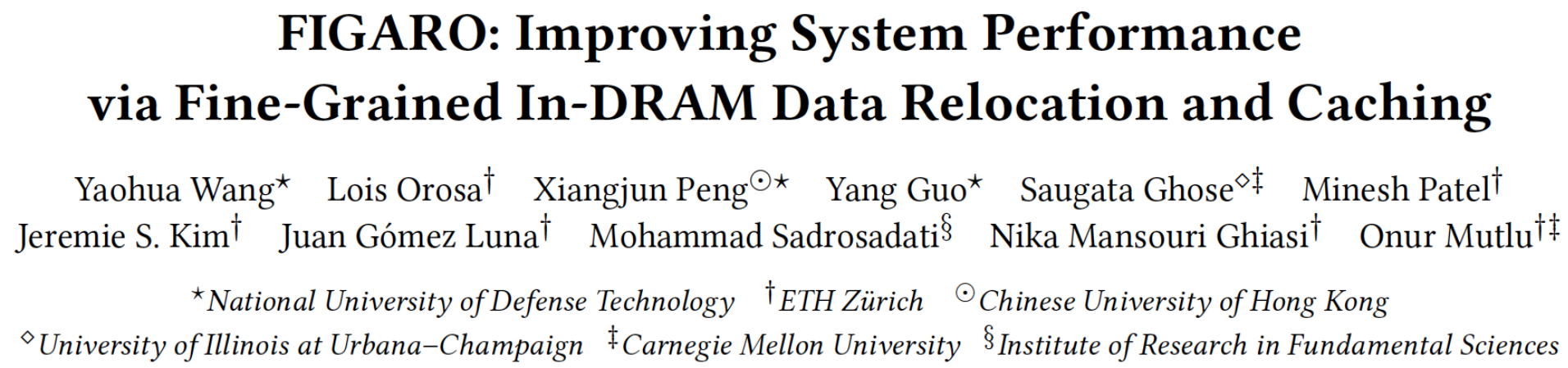
Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

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[†‡]

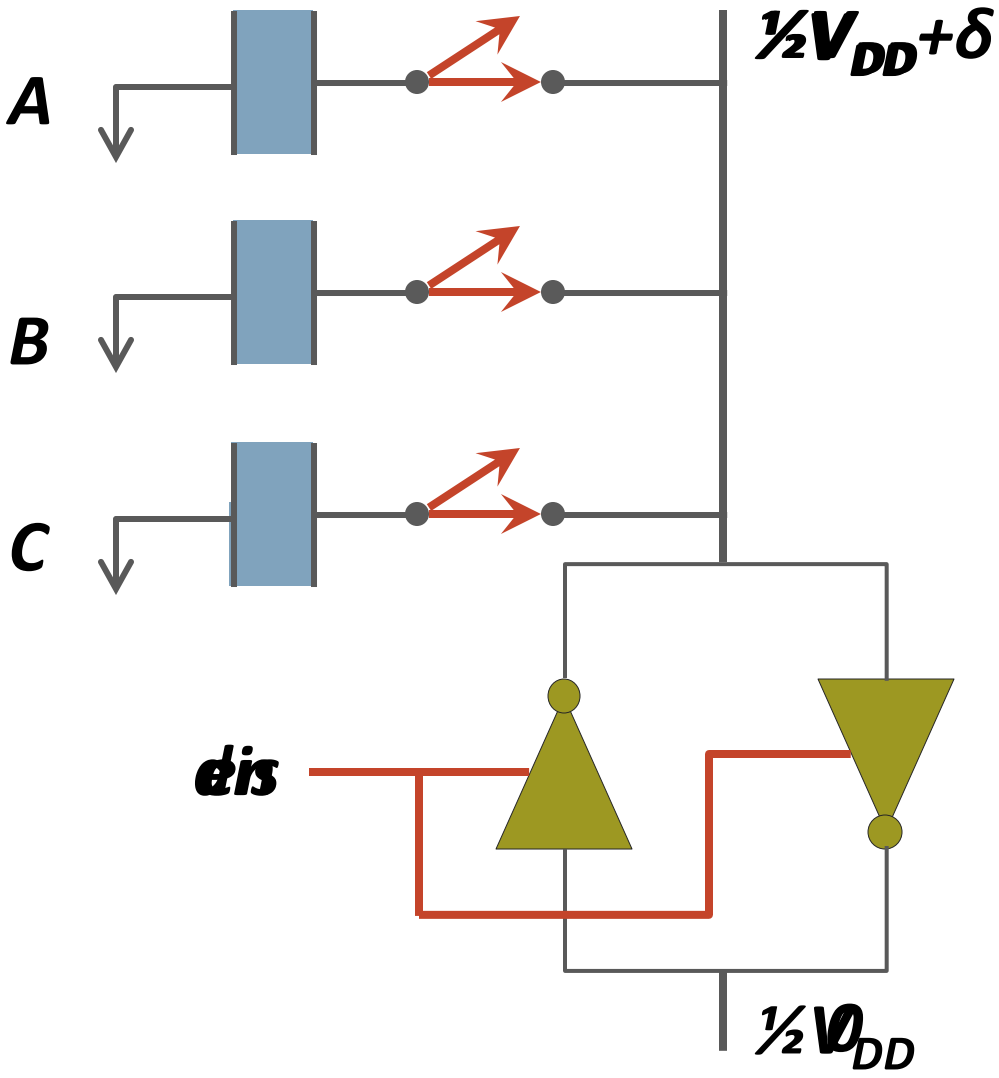
[⋆]*National University of Defense Technology* [†]*ETH Zürich* [⊙]*Chinese University of Hong Kong*
[◇]*University of Illinois at Urbana–Champaign* [‡]*Carnegie Mellon University* [§]*Institute of Research in Fundamental Sciences*

In-Memory Bulk Bitwise Operations

- We can support in-DRAM COPY, ZERO, AND, OR, NOT, MAJ
- At low cost
- Using inherent analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.

- New memory technologies enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, …
  - Can operate on data with minimal movement

In-DRAM AND/OR: Triple Row Activation



Final State
AB + BC + AC

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

Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
  "Fast Bulk Bitwise AND and OR in DRAM"
  IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University †Intel Pittsburgh

In-DRAM NOT: Dual Contact Cell

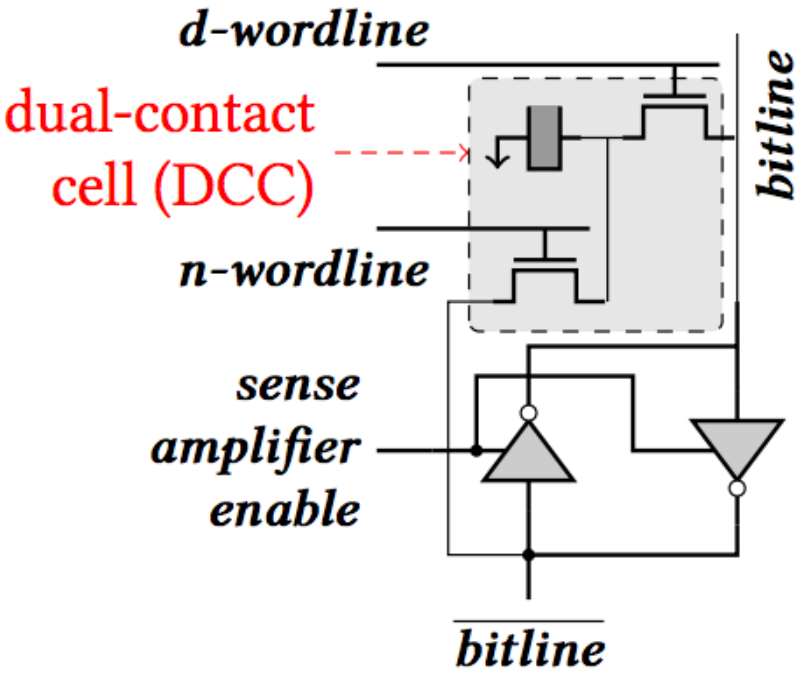


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

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

In-DRAM NOT Operation

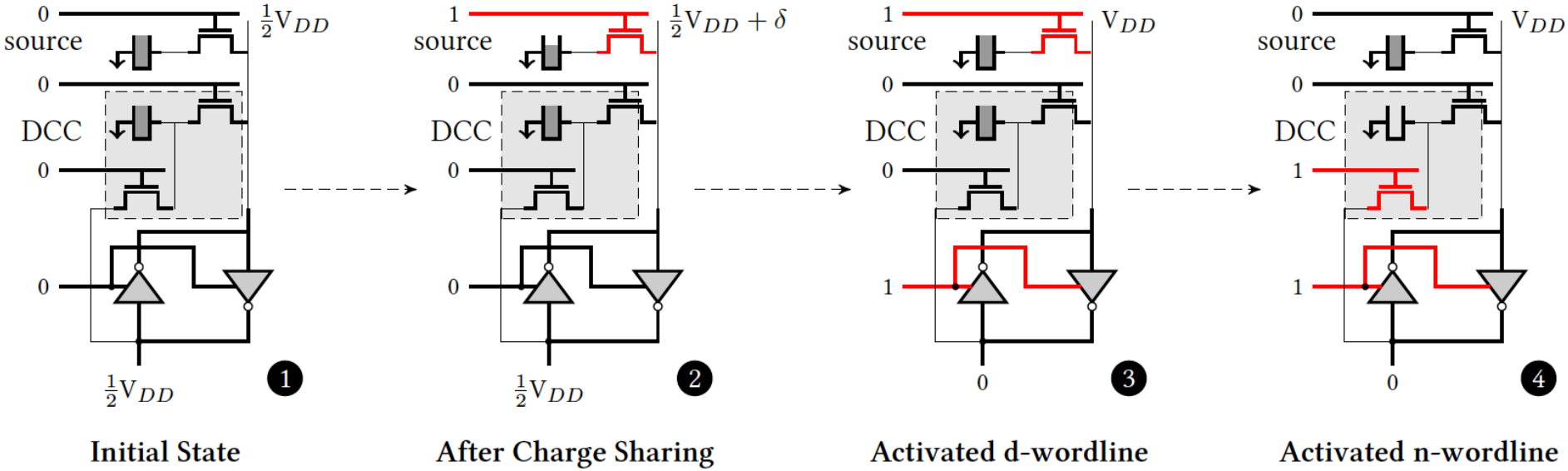
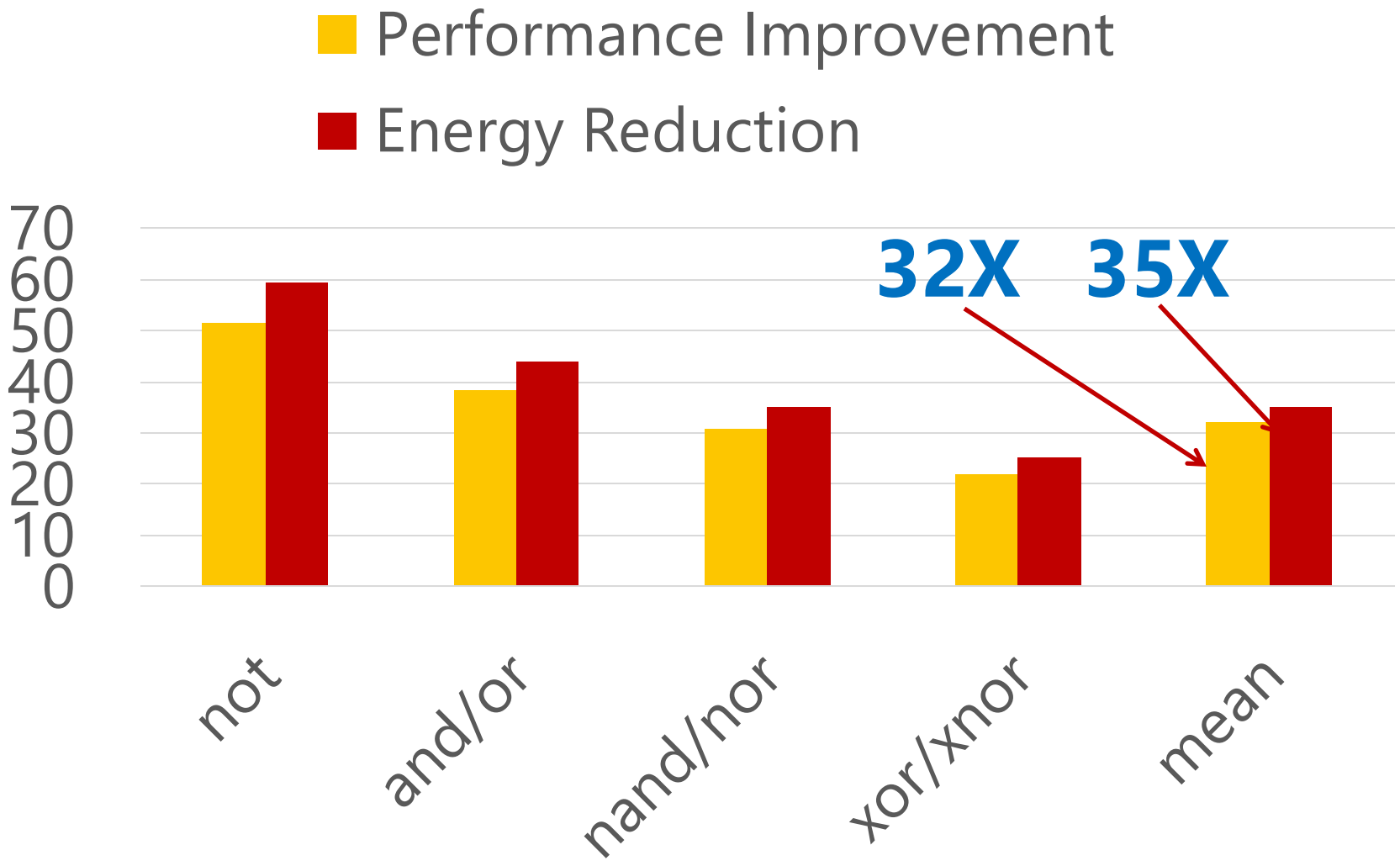


Figure 5: Bitwise NOT using a dual contact capacitor

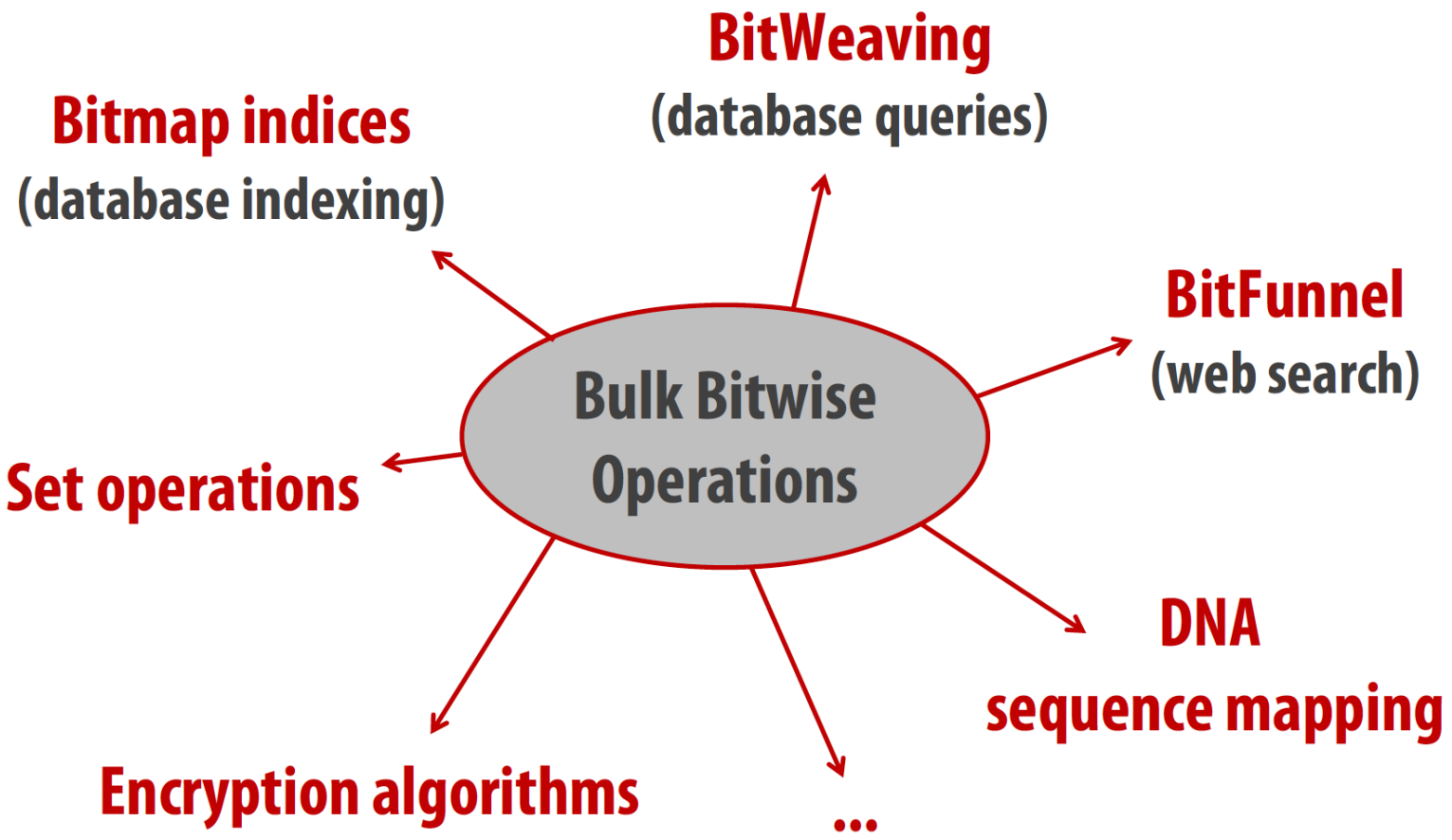
Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Ambit vs. DDR3: Performance and Energy



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

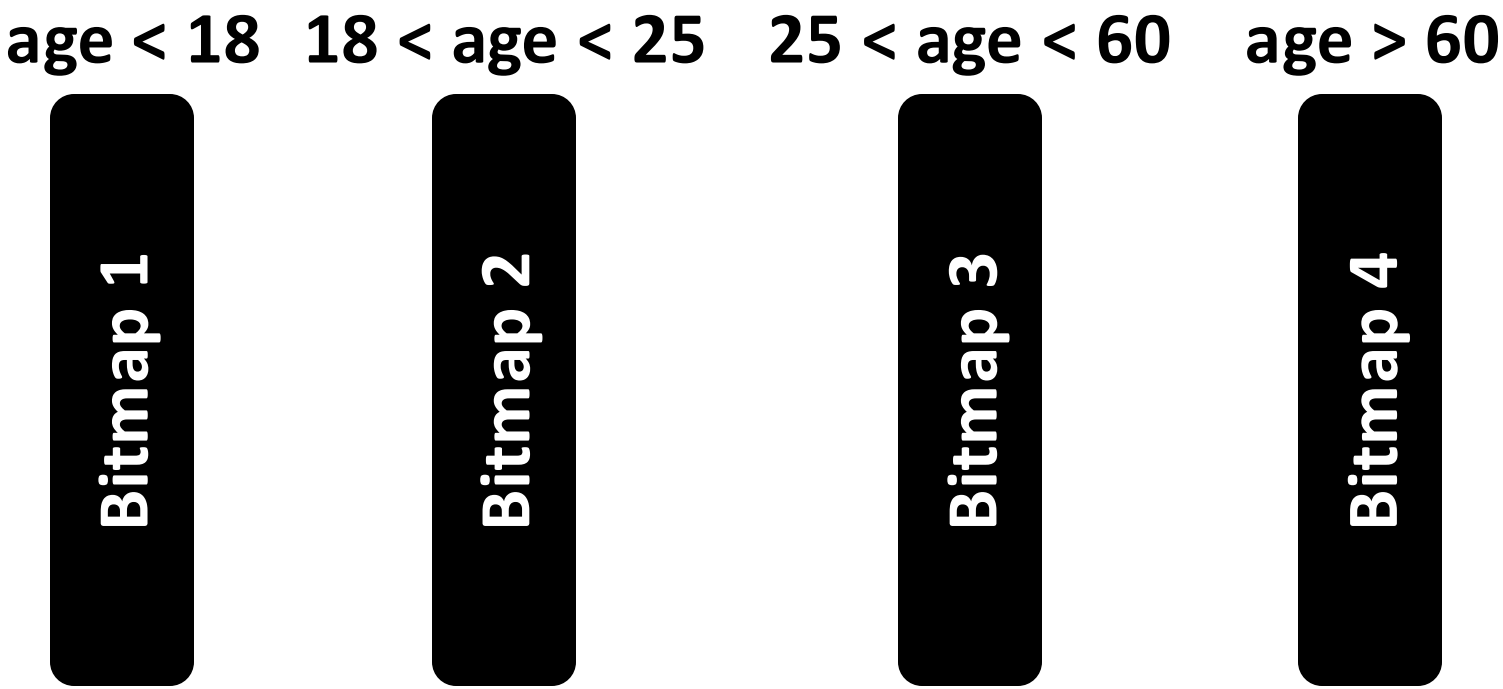
Bulk Bitwise Operations in Workloads



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

Example Data Structure: Bitmap Index

- Alternative to B-tree and its variants
- Efficient for performing *range queries* and *joins*
- **Many bitwise operations to perform a query**



Performance: Bitmap Index on Ambit

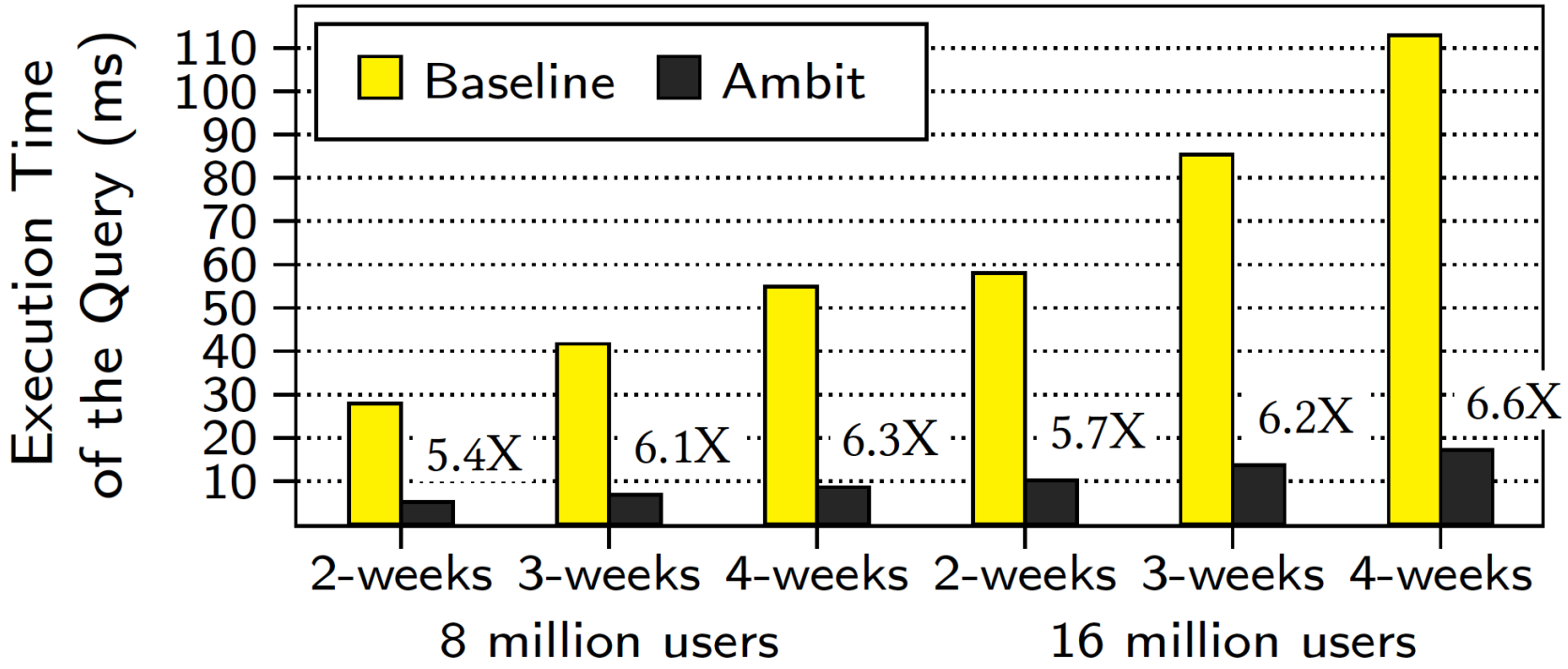


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

>5.4-6.6X Performance Improvement

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

Performance: BitWeaving on Ambit

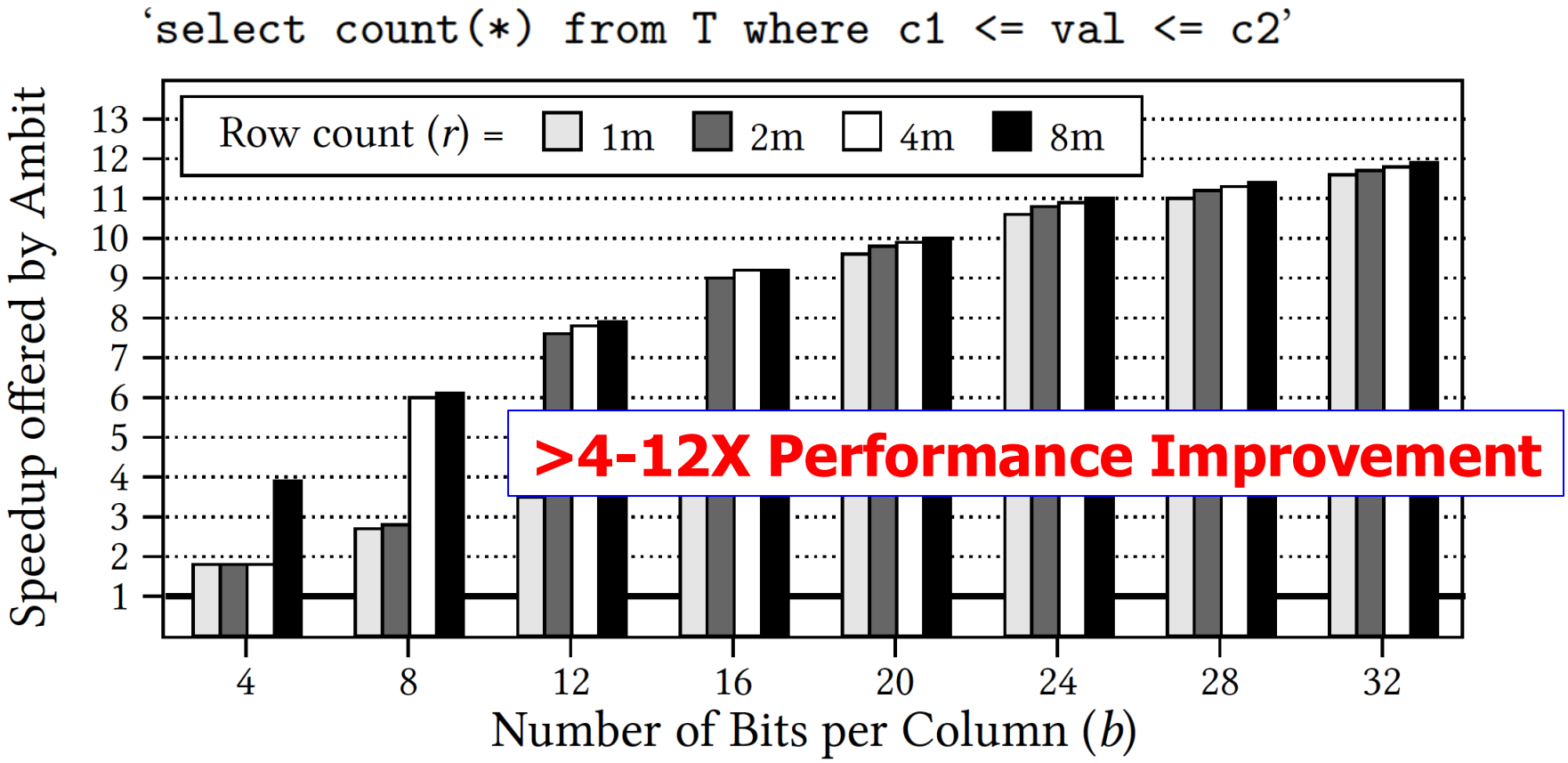


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
  "Fast Bulk Bitwise AND and OR in DRAM"
  IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University †Intel Pittsburgh

More on In-DRAM Bitwise Operations

- Vivek Seshadri et al., "**Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology**," MICRO 2017.

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

Vivek Seshadri[1,5] Donghyuk Lee[2,5] Thomas Mullins[3,5] Hasan Hassan[4] Amirali Boroumand[5]
Jeremie Kim[4,5] Michael A. Kozuch[3] Onur Mutlu[4,5] Phillip B. Gibbons[5] Todd C. Mowry[5]

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

More on 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

Outline

- **Processing-Using-Memory (PUM) Solutions**
  - Review of DRAM Background, RowClone, and Ambit
  - **SIMDRAM & MIMDRAM: Generalizing PUM Computing**
  - pLUTo: Extending Operation Support with LUTs
  - PUM in Off-the-Shelf DRAM Chips
  - PUM Beyond Boolean/Arithmetic and DRAM
- Processing-Near-Memory (PNM) Solutions
  - Overview
  - Accelerating Neural Networks & Databases
  - Real PNM Systems
- Barriers for Adoption (and Current Solutions)
- Conclusion

SIMDRAM Framework

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

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

*Nastaran Hajinazar[1,2] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

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

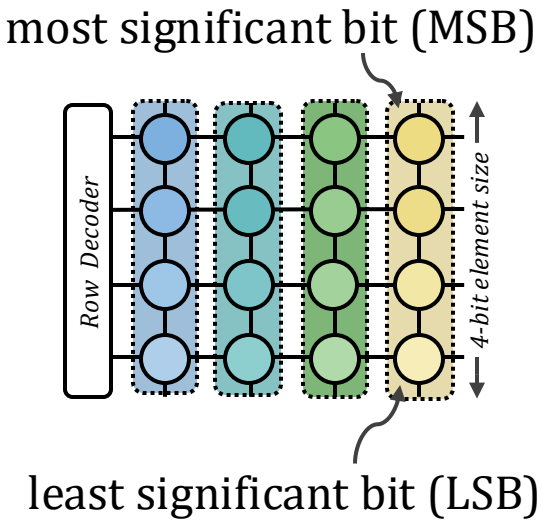
SIMDRAM Key Idea

- **SIMDRAM:** An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:
  - **Efficiently** computing **complex** operations in DRAM
  - Providing the ability to implement **arbitrary** operations as required
  - Using an **in-DRAM massively-parallel SIMD substrate** that requires **minimal** changes to DRAM architecture

SIMDRAM: PuM Substrate

- SIMDRAM framework is built around a DRAM substrate that enables two techniques:

(1) Vertical data layout

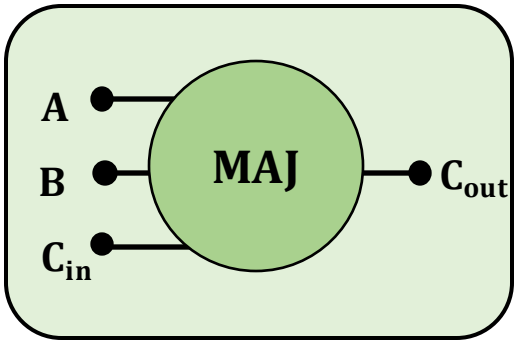


Pros compared to the conventional horizontal layout:

- Implicit shift operation
- Massive parallelism

(2) Majority-based computation

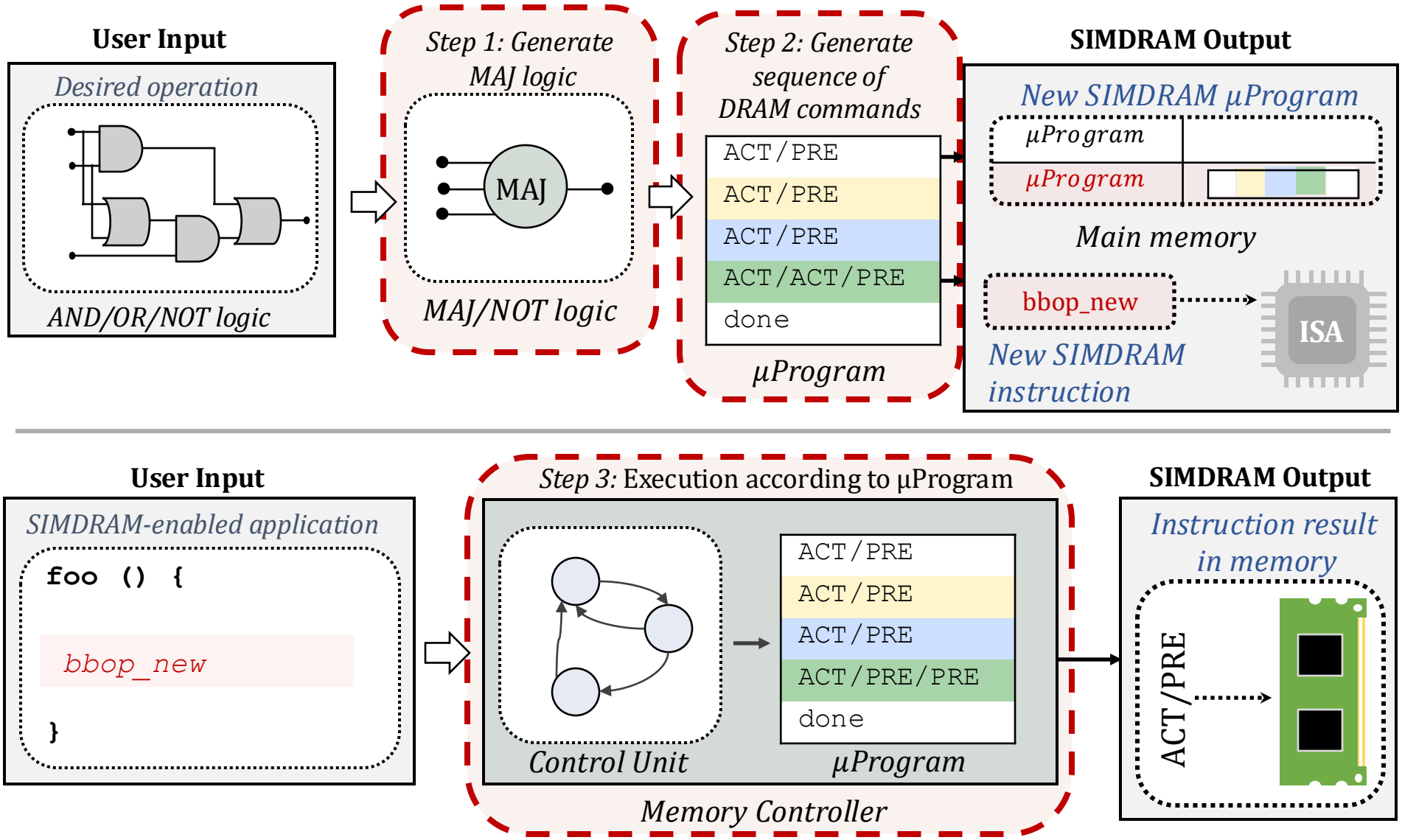
$C_{out} = AB + AC_{in} + BC_{in}$



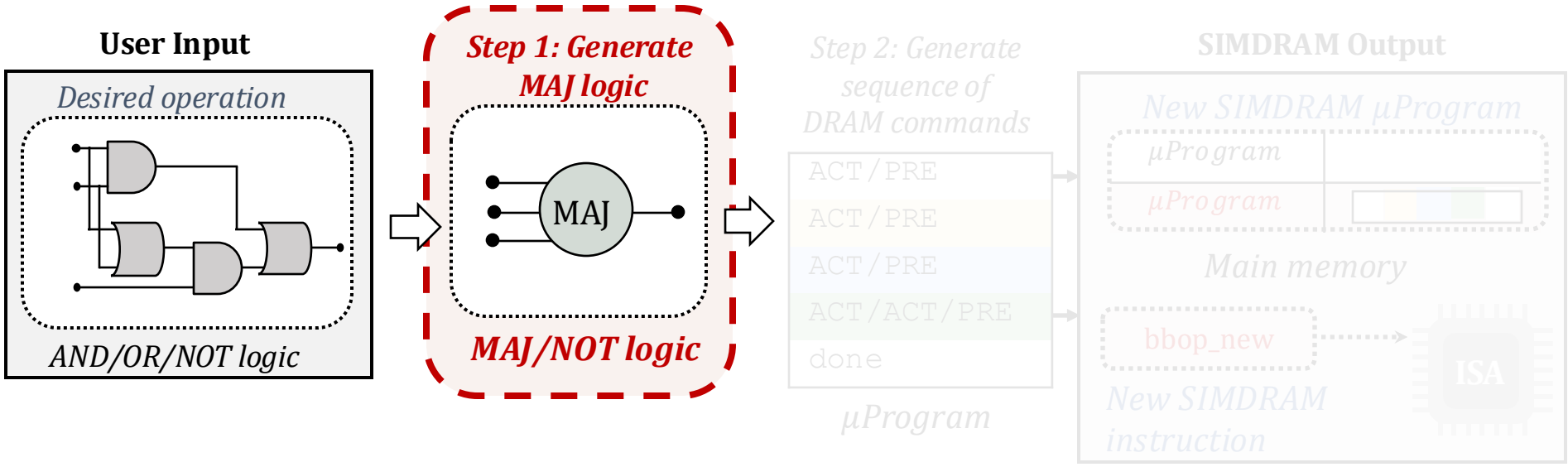
Pros compared to AND/OR/NOT-based computation:

- Higher performance
- Higher throughput
- Lower energy consumption

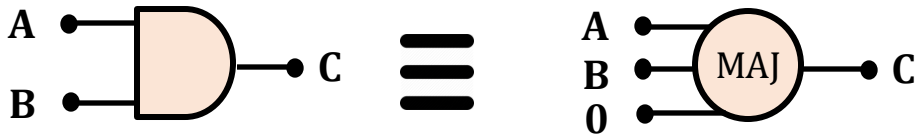
SIMDRAM Framework: Overview



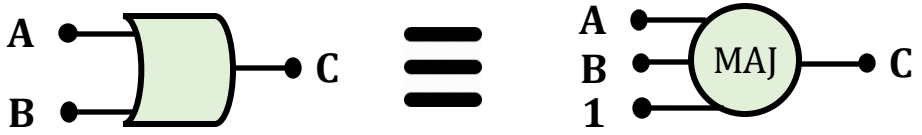
SIMDRAM Framework: Step 1



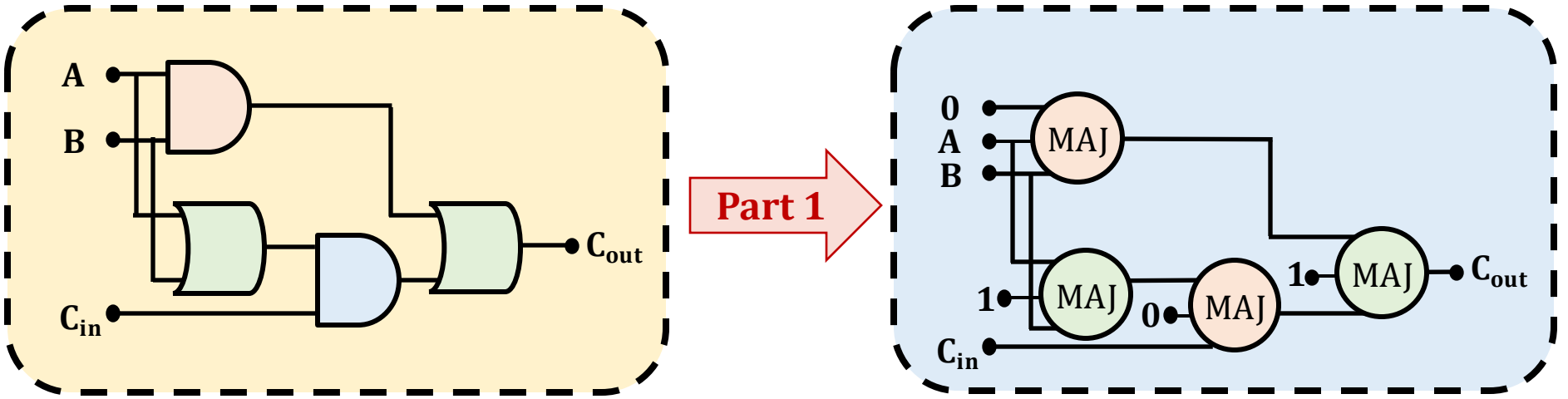
Step 1: Naïve MAJ/NOT Implementation



output is “1” only when A = B = “1”

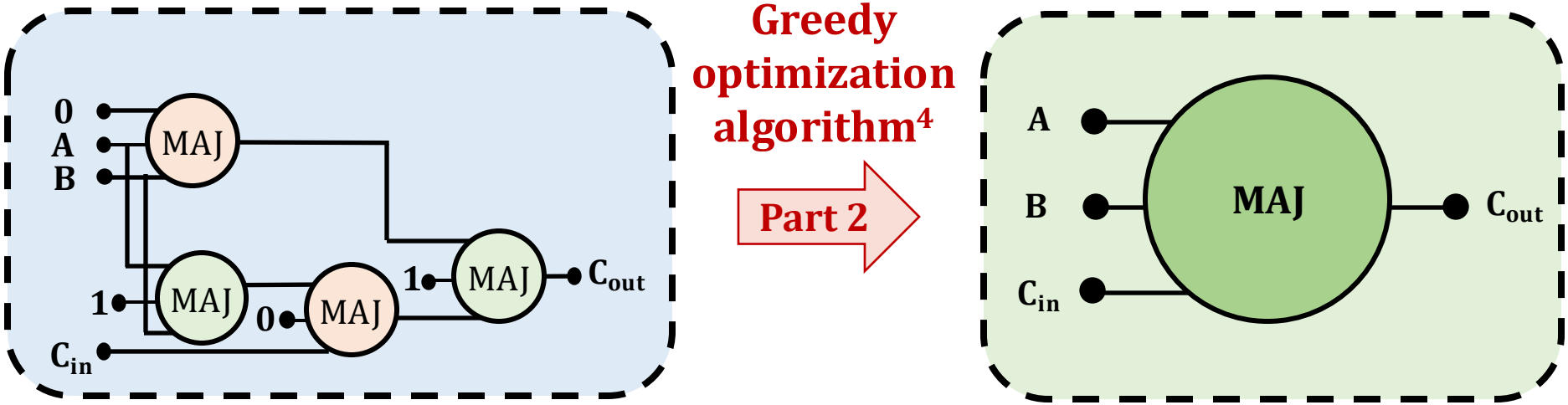


output is “0” only when A = B = “0”



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

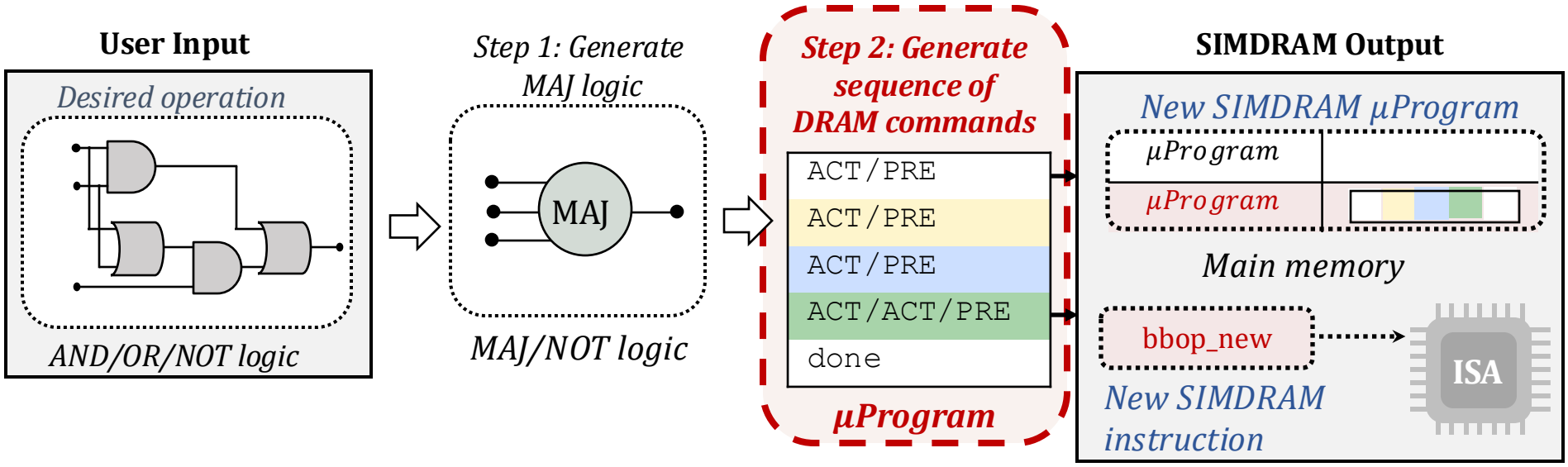
Step 1: Efficient MAJ/NOT Implementation



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: µProgram Generation

- **µProgram:** A series of microarchitectural operations (e.g., ACT/PRE) that SIMDRAM uses to execute SIMDRAM operation in DRAM

- **Goal of Step 2**: To generate the µProgram that executes the desired SIMDRAM operation in DRAM

Task 1: Allocate DRAM rows to the operands

Task 2: Generate µProgram

Step 2: µProgram Generation

- **µProgram:** A series of microarchitectural operations (e.g., ACT/PRE) that SIMDRAM uses to execute SIMDRAM operation in DRAM

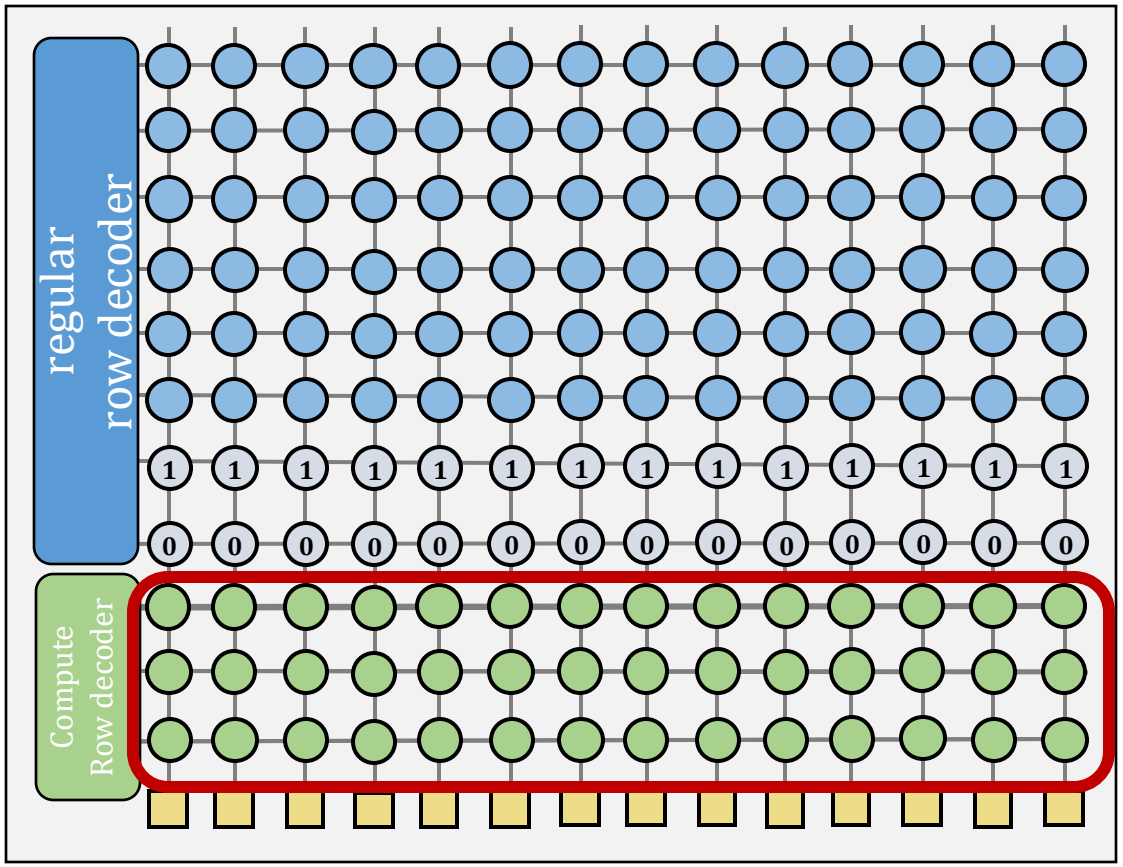
- **Goal of Step 2**: To generate the µProgram that executes the desired SIMDRAM operation in DRAM

Task 1: Allocate DRAM rows to the operands

Task 2: Generate µProgram

Task 1: Allocating DRAM Rows to Operands

- Allocation algorithm considers two constraints specific to processing-using-DRAM



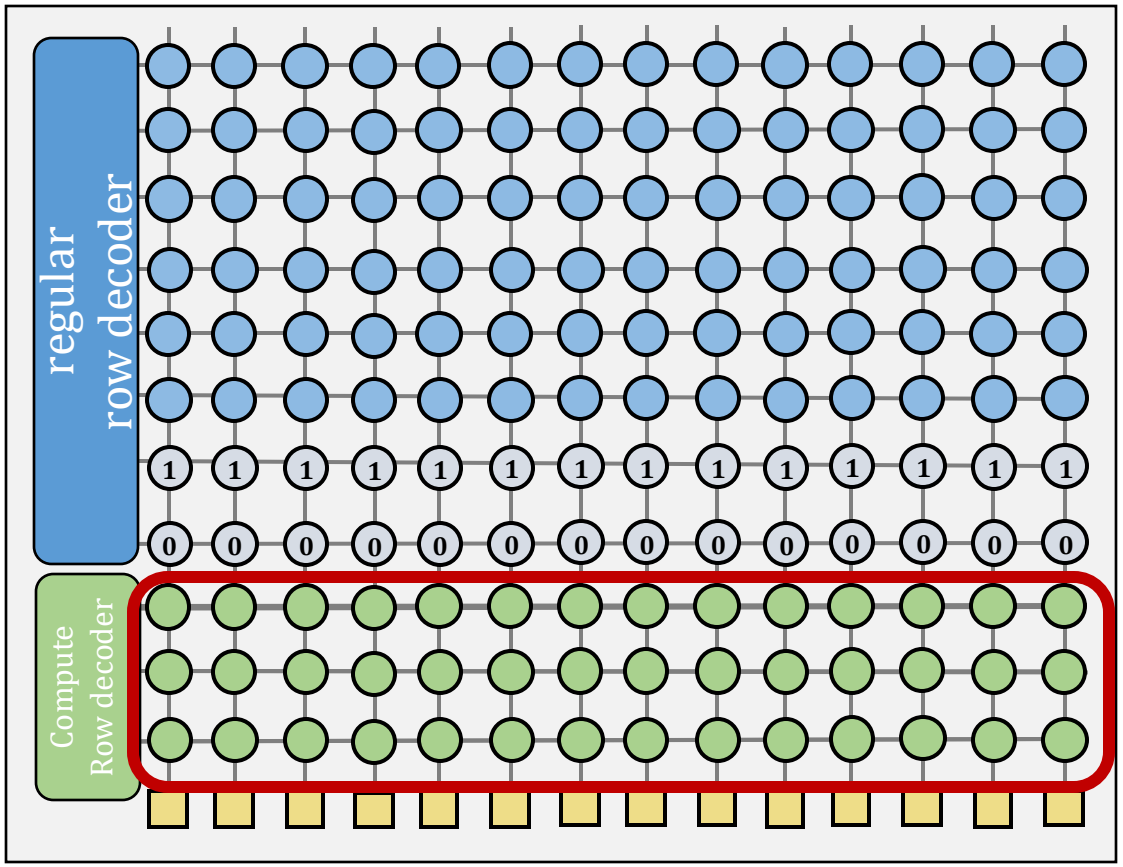
subarray organization

Constraint 1:
Limited number of rows reserved for computation

Compute rows

Task 1: Allocating DRAM Rows to Operands

- Allocation algorithm considers two constraints specific to processing-using-DRAM



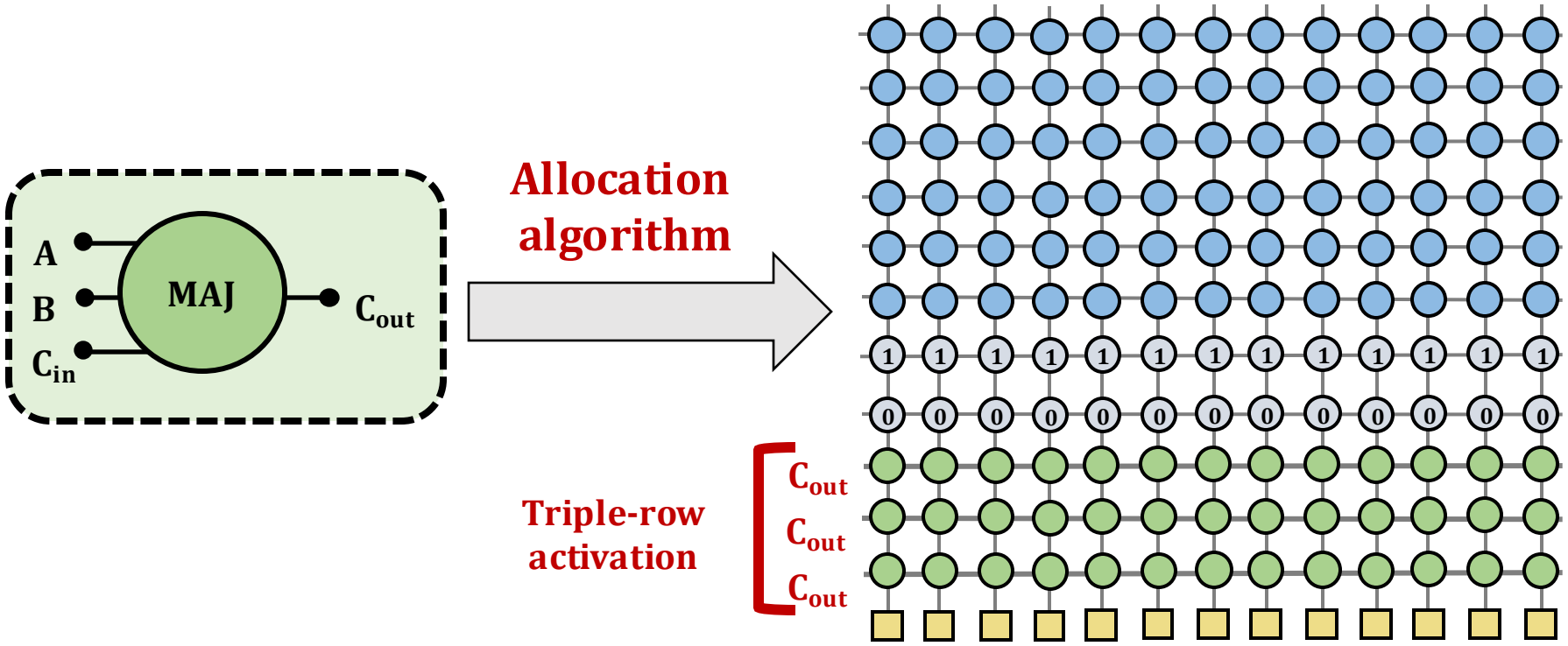
subarray organization

Constraint 2:
Destructive behavior of triple-row activation

Overwritten with MAJ output

Task 1: Allocating DRAM Rows to Operands

- Allocation algorithm:
  - Assigns as many inputs as the number of free compute rows
  - All three input rows contain the MAJ output and can be reused



Step 2: µProgram Generation

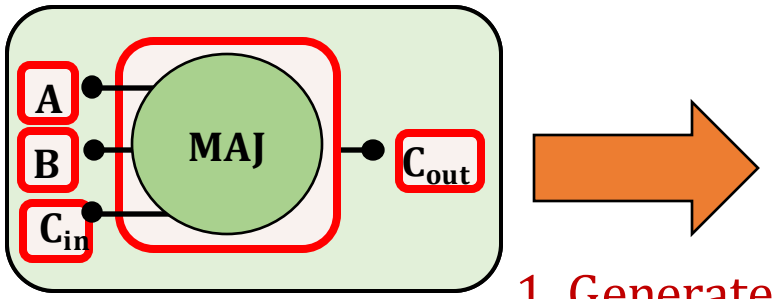
- **µProgram:** A series of microarchitectural operations (e.g., ACT/PRE) that SIMDRAM uses to execute SIMDRAM operation in DRAM

- **Goal of Step 2**: To generate the µProgram that executes the desired SIMDRAM operation in DRAM

Task 1: Allocate DRAM rows to the operands

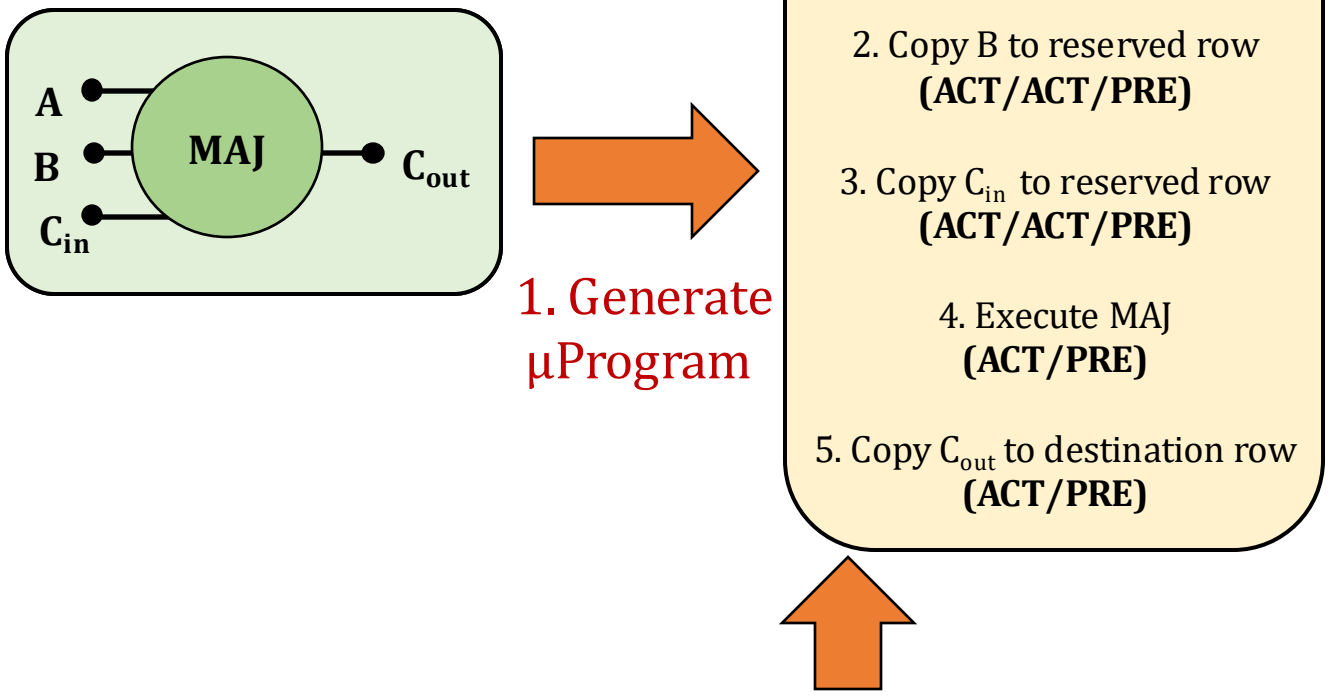
Task 2: Generate µProgram

Task 2: Generate an Initial μProgram

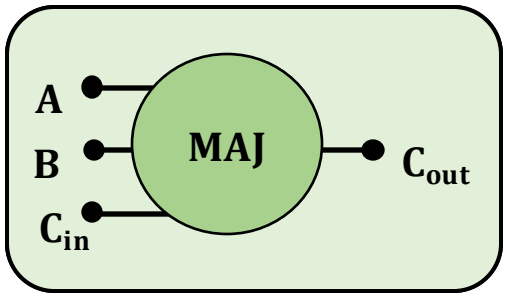


1. Generate μProgram

Task 2: Optimize the µProgram



Task 2: Optimize the µProgram



1. Generate µProgram

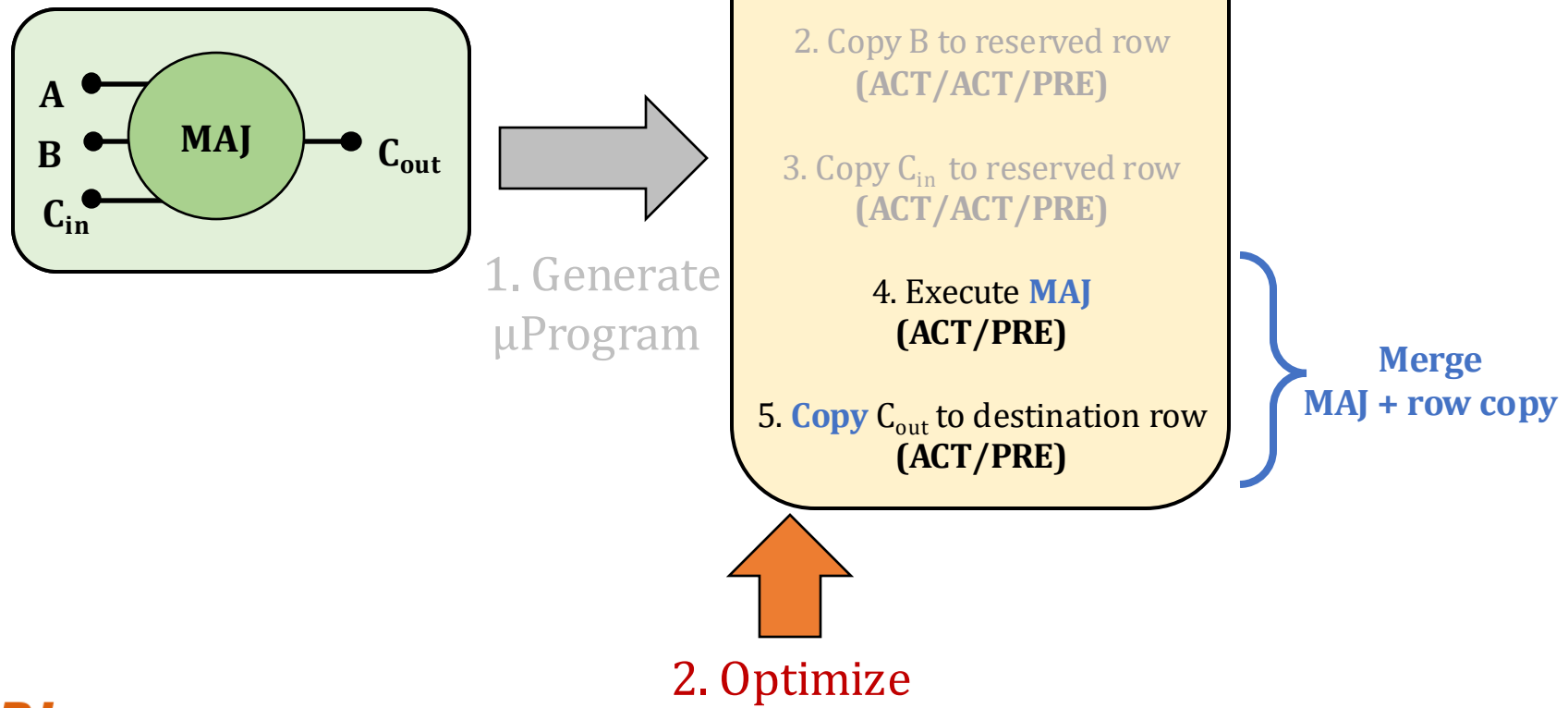
Initial µProgram

1. **Copy** A to reserved row **(ACT/ACT/PRE)**
2. **Copy** B to reserved row **(ACT/ACT/PRE)**
3. **Copy** C_{in} to reserved row **(ACT/ACT/PRE)**
4. Execute MAJ **(ACT/PRE)**
5. Copy C_{out} to destination row **(ACT/PRE)**

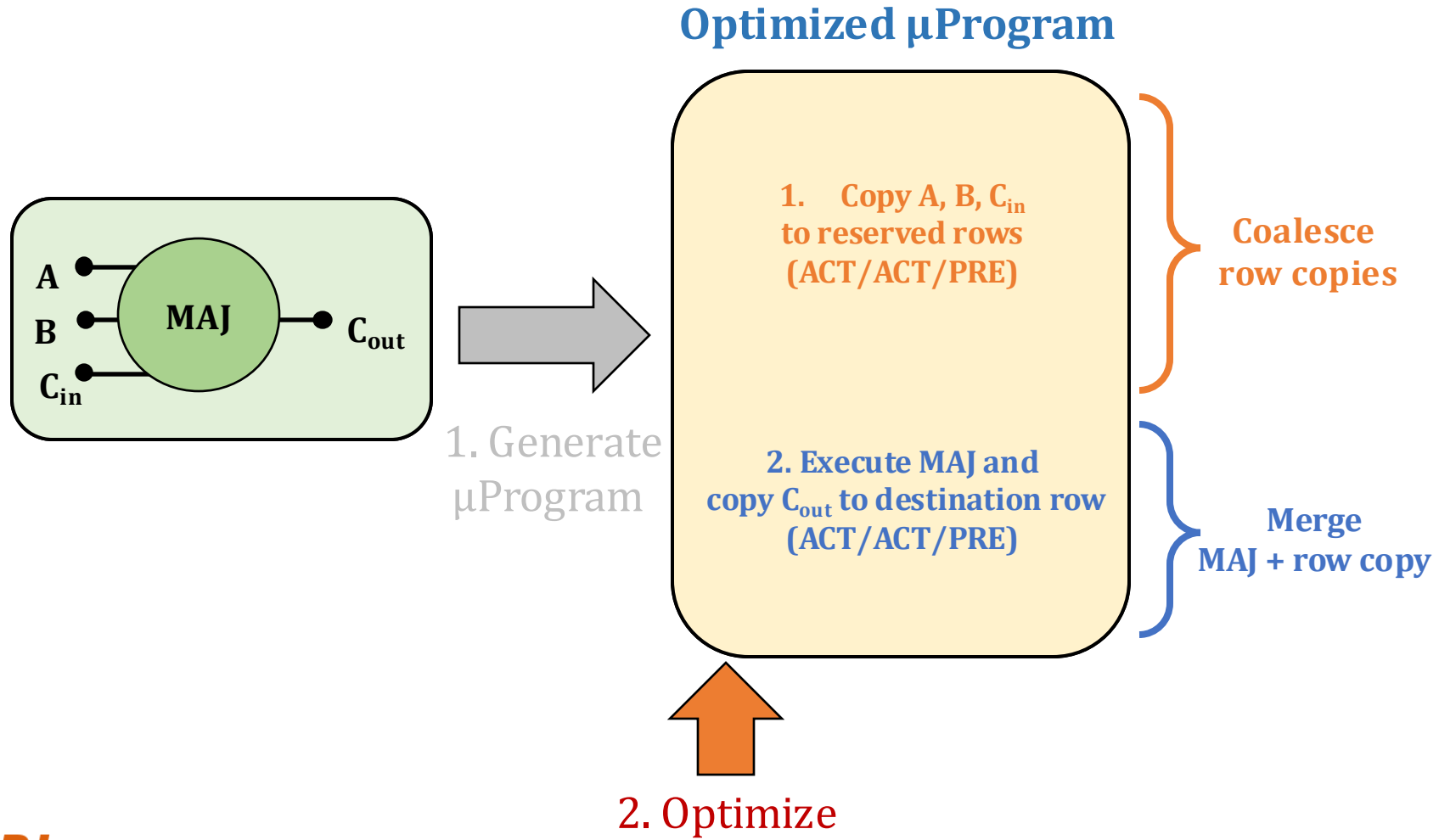
Coalesce row copies

2. Optimize

Task 2: Optimize the µProgram

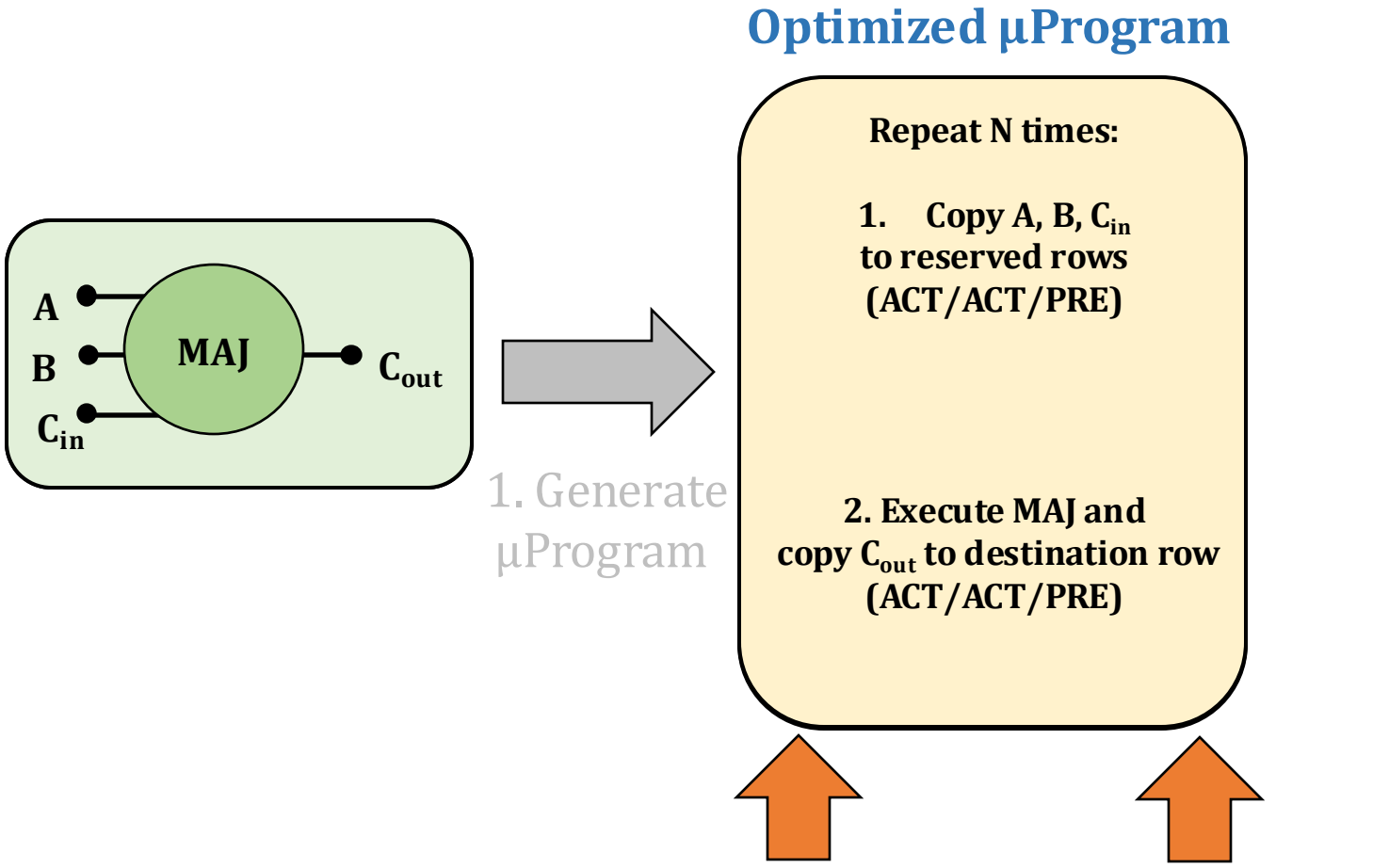


Task 2: Optimize the µProgram



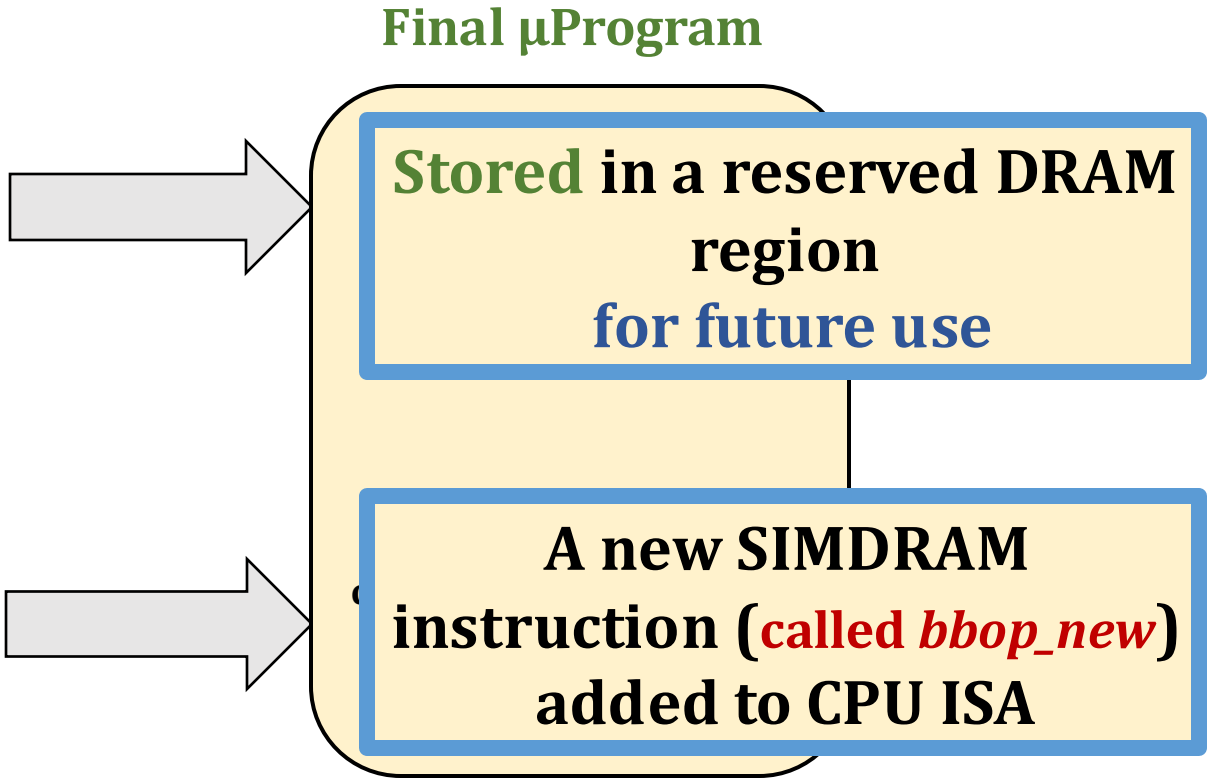
Task 2: Generate N-bit Computation

- **Final μProgram is optimized and computes the desired operation for operands of N-bit size in a bit-serial fashion**

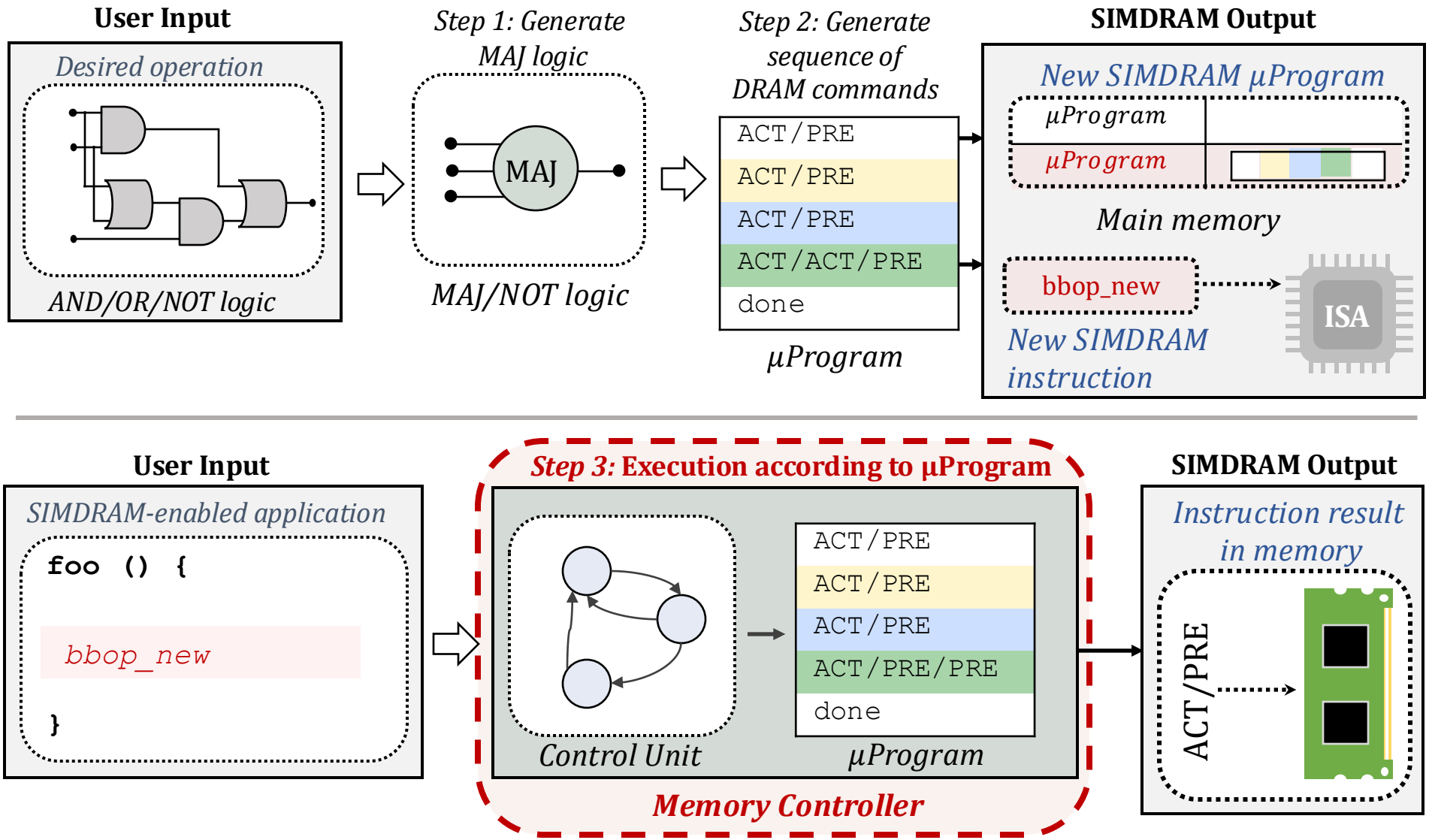


Task 2: Generate µProgram

- **Final µProgram is optimized and computes the desired operation for operands of N-bit size in a bit-serial fashion**

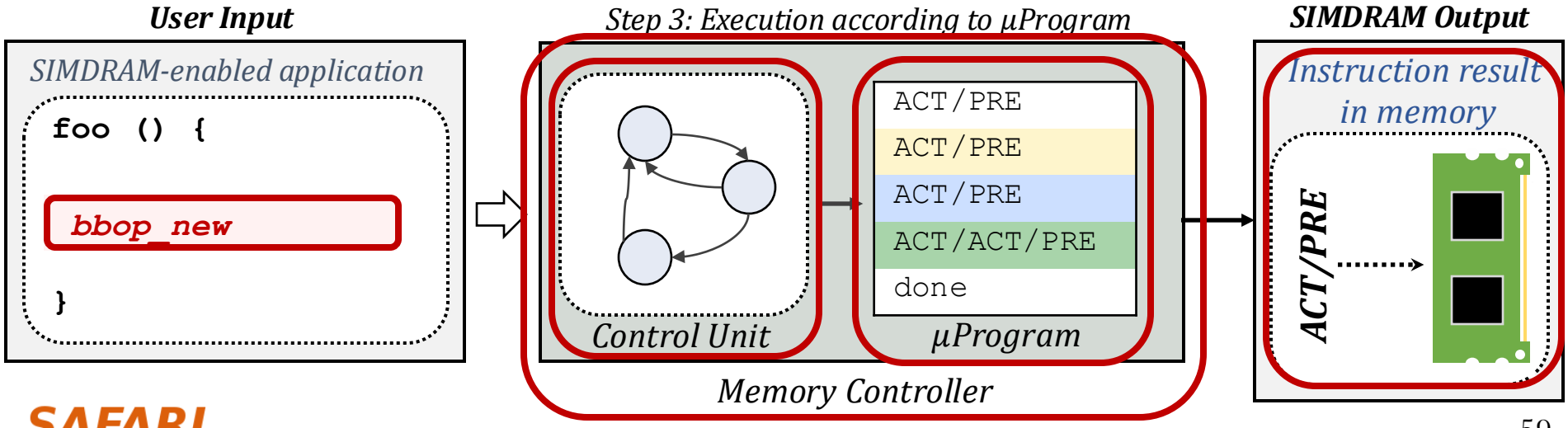


SIMDRAM Framework: Step 3



Step 3: µProgram Execution

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



System Integration

Efficiently transposing data

Programming interface

Handling page faults, address translation, coherence, and interrupts

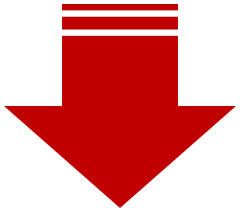
Handling limited subarray size

Security implications

Limitations of our framework

Transposing Data

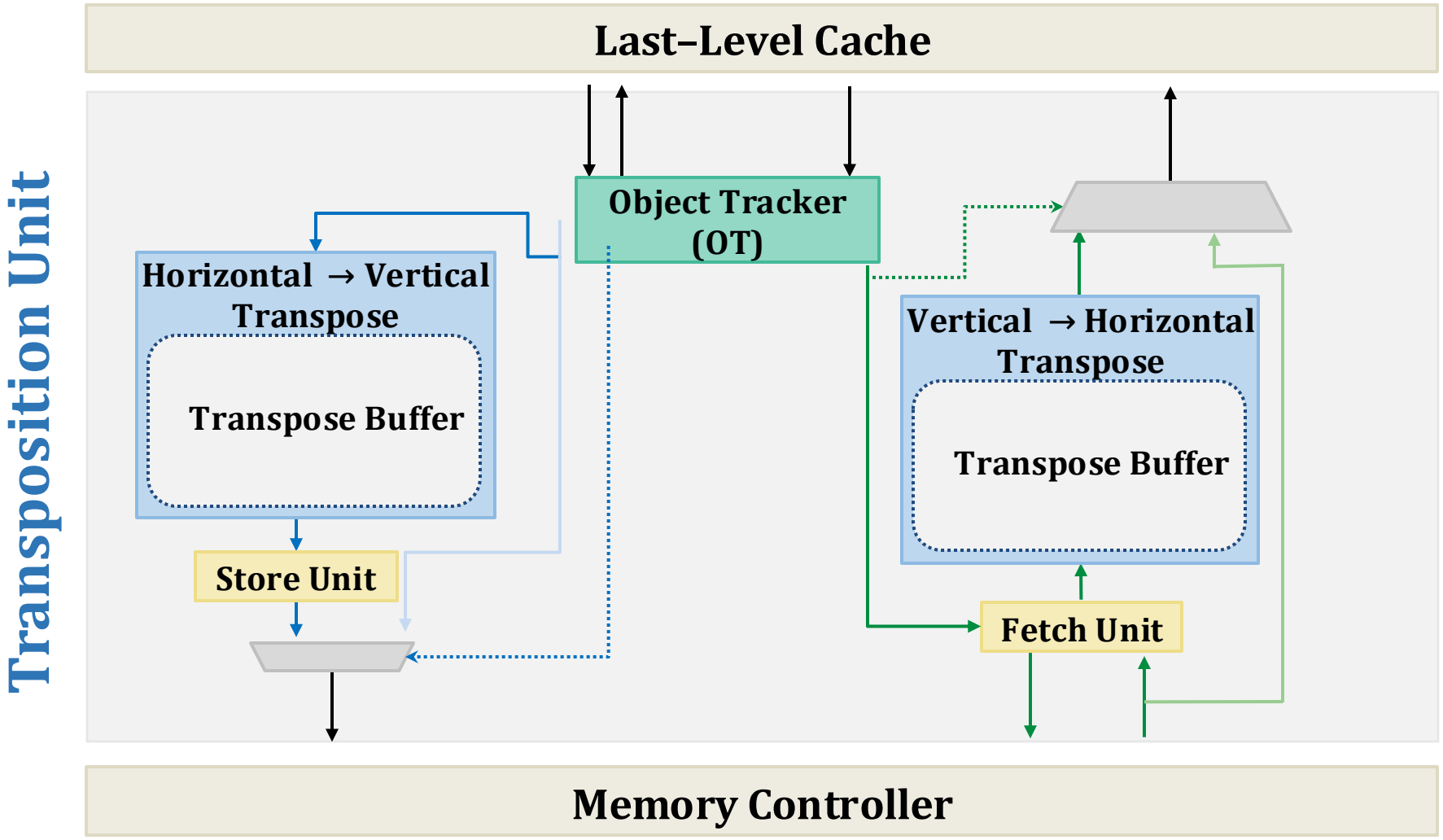
- SIMDRAM operates on vertically-laid-out data
- Other system components expect data to be laid out horizontally



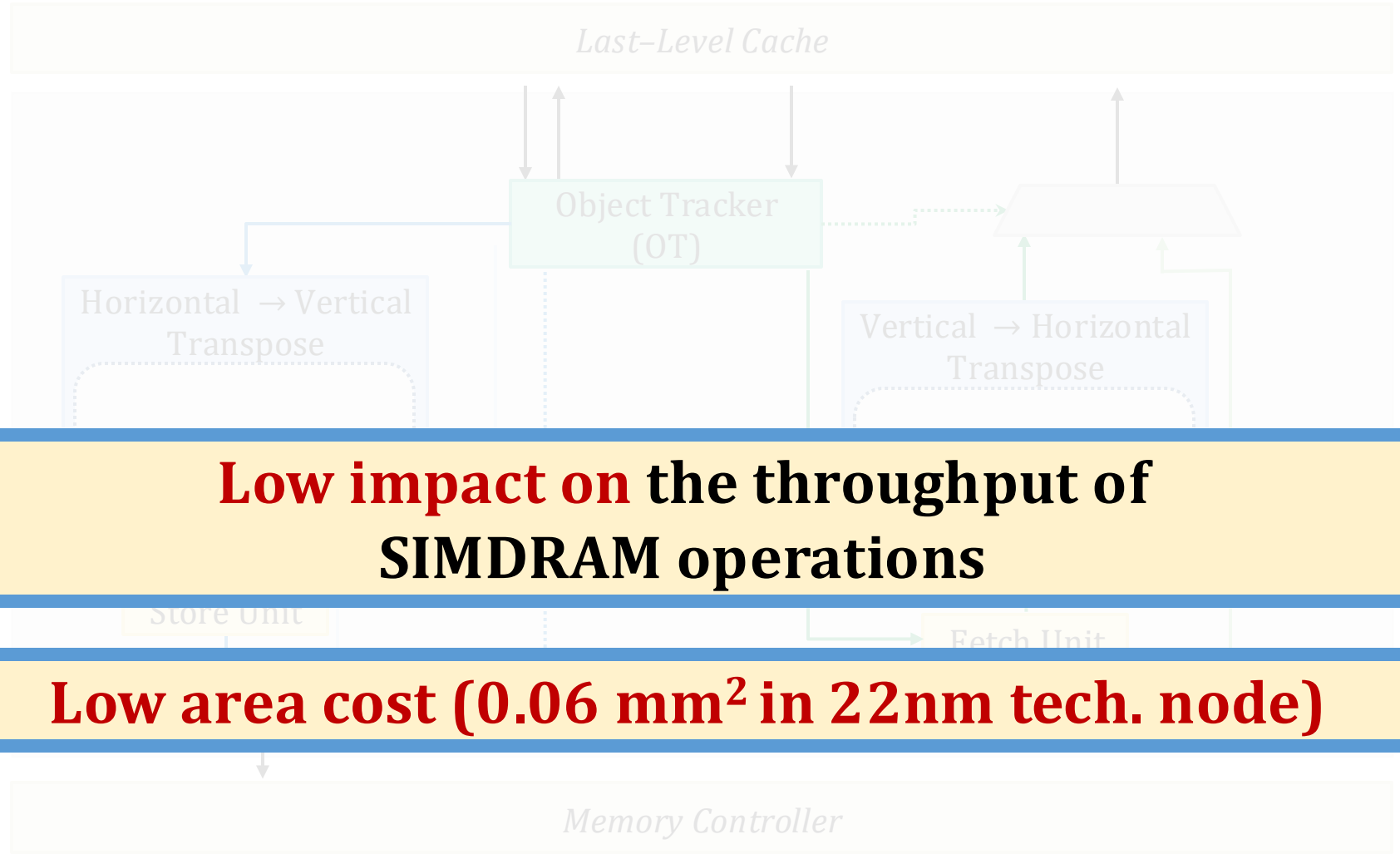
Challenging to share data between SIMDRAM and CPU

Transposition Unit

Transforms the data layout from horizontal to vertical, and vice versa



Efficiently Transposing Data

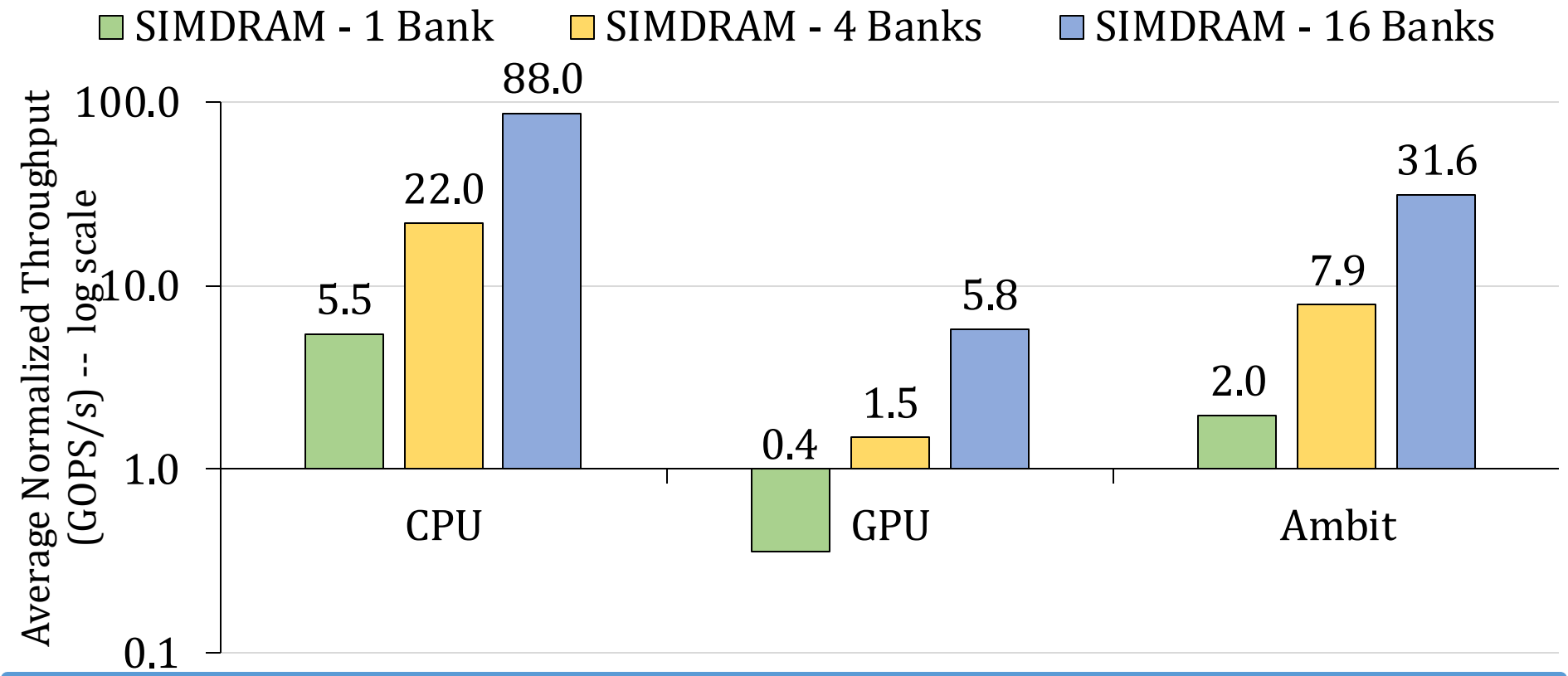


Low impact on the throughput of SIMDRAM operations

Low area cost (0.06 mm^2 in 22nm tech. node)

Throughput Analysis

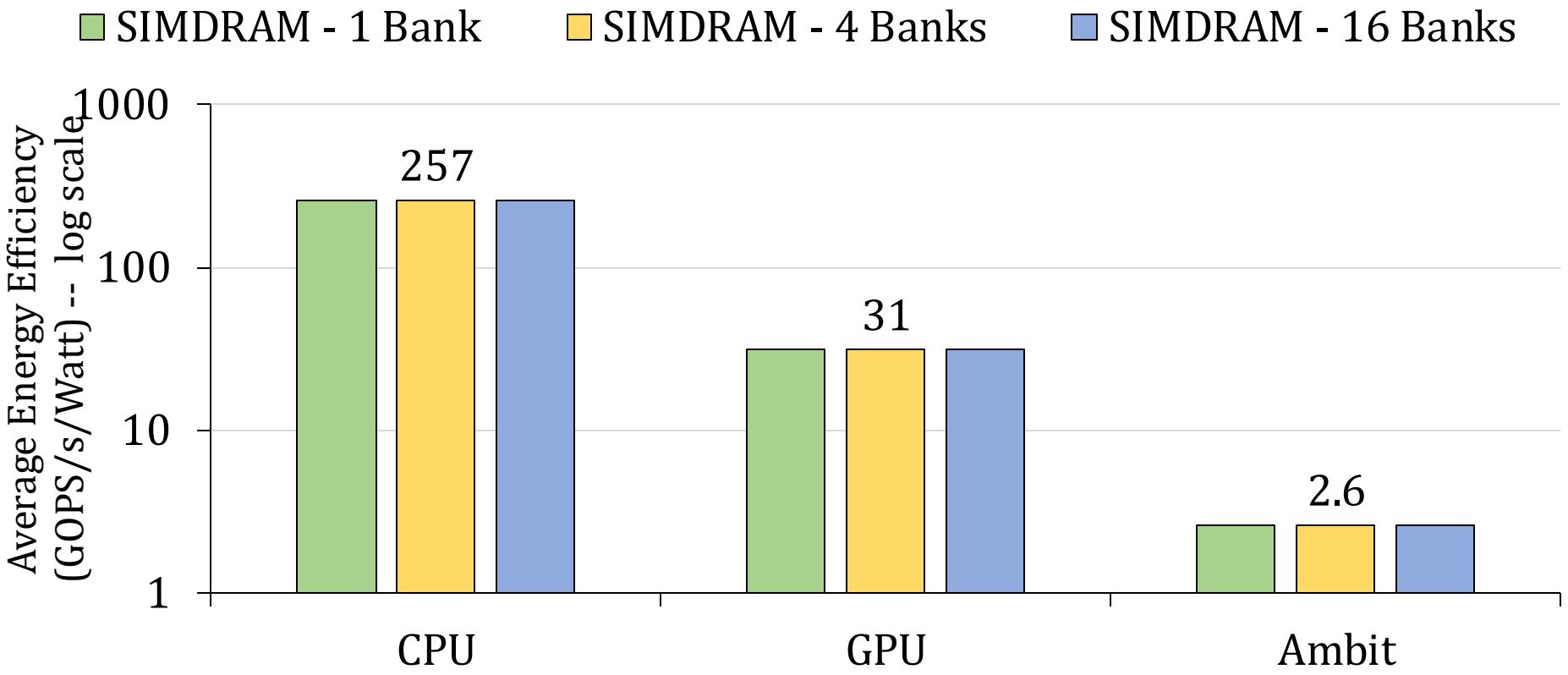
Average normalized throughput across all 16 SIMDRAM operations



SIMDRAM significantly outperforms all state-of-the-art baselines for a wide range of operations

Energy Analysis

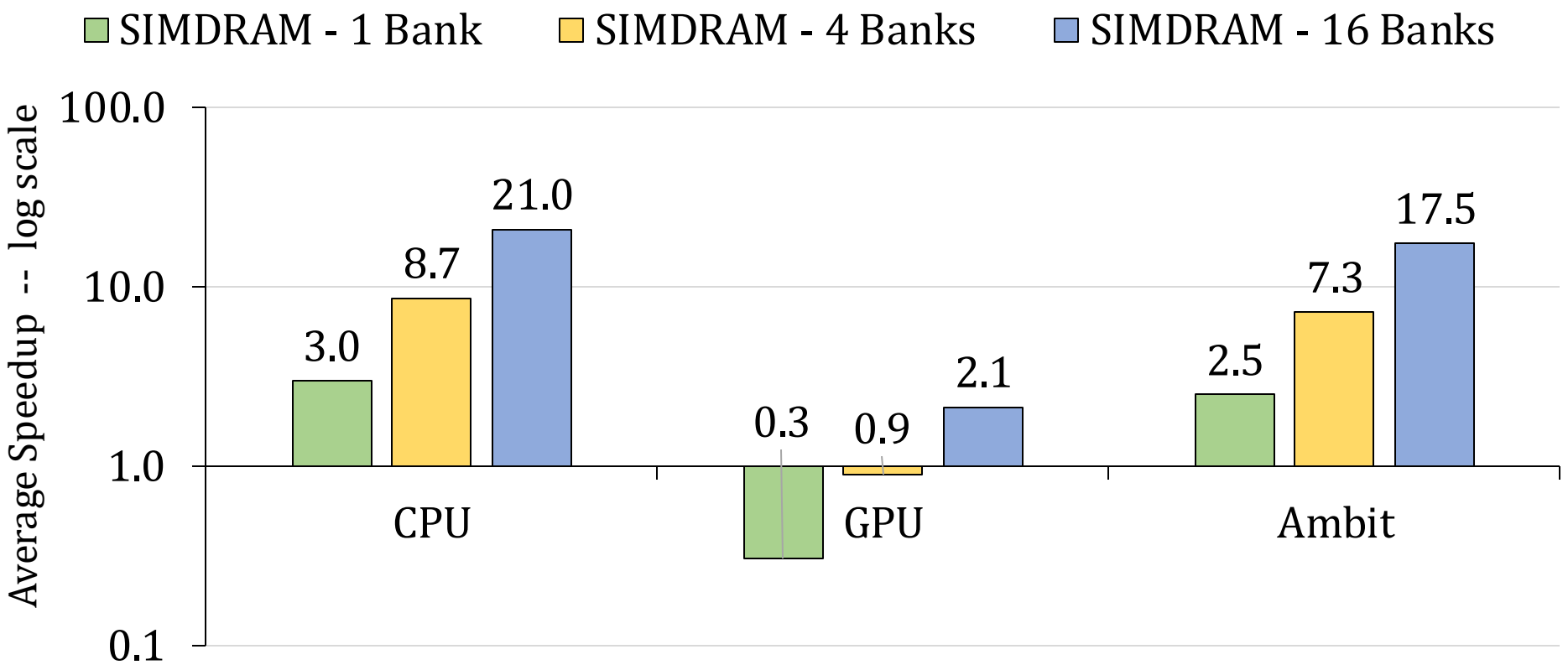
Average normalized energy efficiency across all 16 SIMDRAM operations



SIMDRAM is more energy-efficient than all state-of-the-art baselines for a wide range of operations

Real-World Applications

Average speedup across 7 real-world applications



SIMDRAM effectively and efficiently accelerates many commonly-used real-world applications

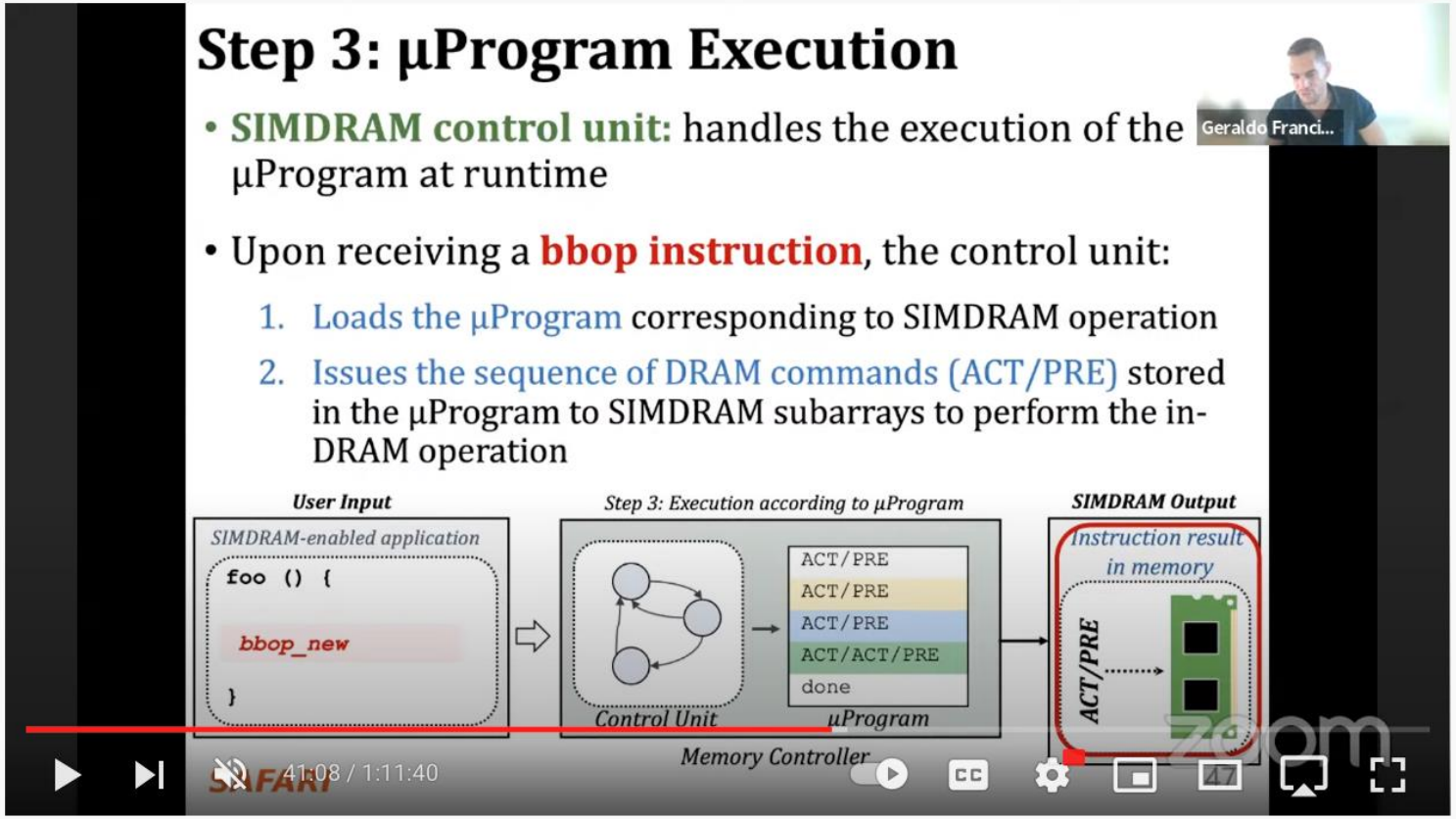
SIMDRAM Conclusion

- **SIMDRAM:**
  - Enables efficient computation of a flexible set and wide range of operations in a PuM massively parallel SIMD substrate
  - Provides the hardware, programming, and ISA support, to:
    - Address key system integration challenges
    - Allow programmers to define and employ new operations without hardware changes

SIMDRAM is a promising PuM framework

- Can **ease the adoption** of processing-using-DRAM architectures
- Improves the **performance** and **efficiency** of processing-using-memory architectures

Lecture on SIMDRAM



https://youtu.be/pmZoAAhvkRQ

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!

2nd Workshop on Memory-Centric Computing: Processing-Using-Memory - Part I

Dr. Geraldo F. Oliveira

https://geraldofojunior.github.io

ICS 2025

08 June 2025

SAFARI

ETH zürich