7th Workshop on Memory-Centric Computing

Processing-Using-Memory in Real DRAM Chips

İsmail Emir Yüksel

ICS 2026

6 July 2026

SAFARI

ETH zürich

Brief Self Introduction

- **Ismail Emir Yuksel**
  - Researcher @ SAFARI Research Group since September 2022
    - PhD @ ETH Zurich (Feb 2024 – ongoing)
  - ismail.yuksel@safari.ethz.ch

- **Research interests:**
  - Computer architecture
  - Processing-in-Memory (specifically Processing-using-DRAM)
  - DRAM Safety, Security and Reliability
  - Hardware security, safety, reliability, performance, availability, fairness, energy efficiency
  - …

Problem

Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

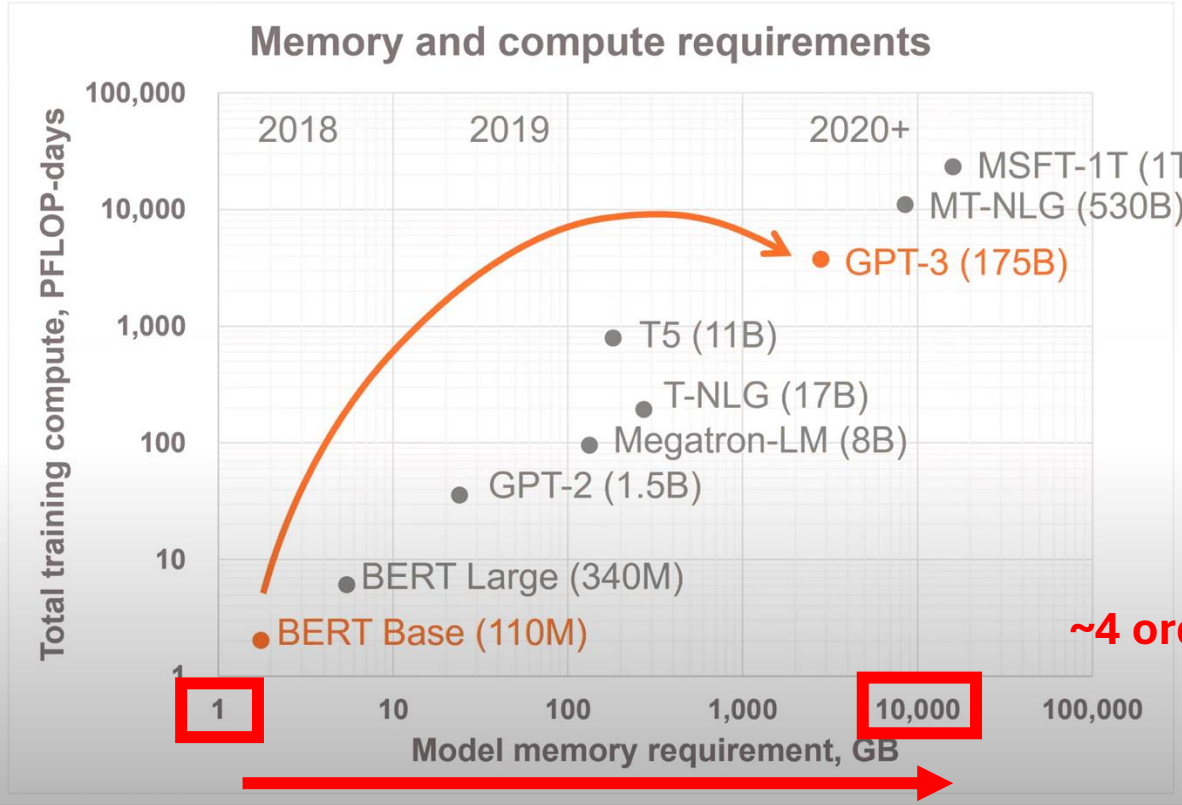
- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

- Data is increasing
  - We can generate more than we can process
  - We need to perform more sophisticated analyses on more data

Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks



1800x more compute

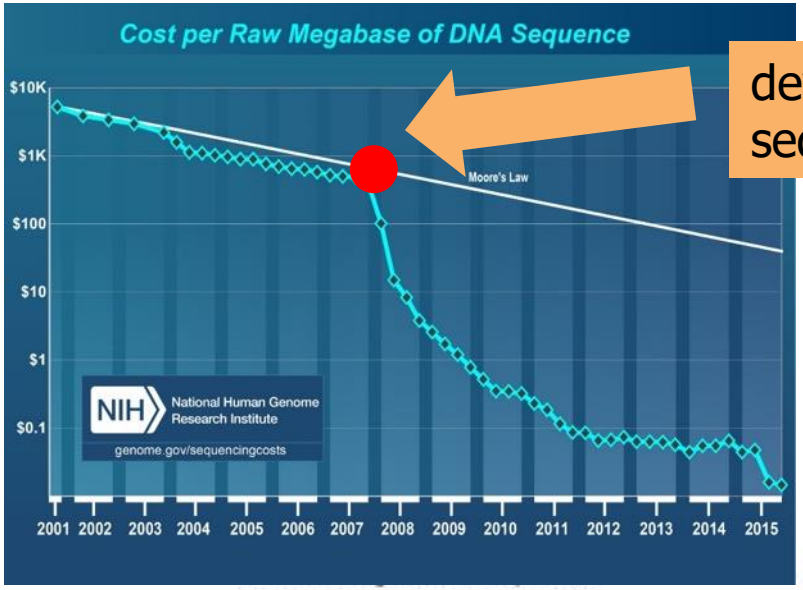
In just **2 years**

Tomorrow, **multi-trillion** parameter models

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

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

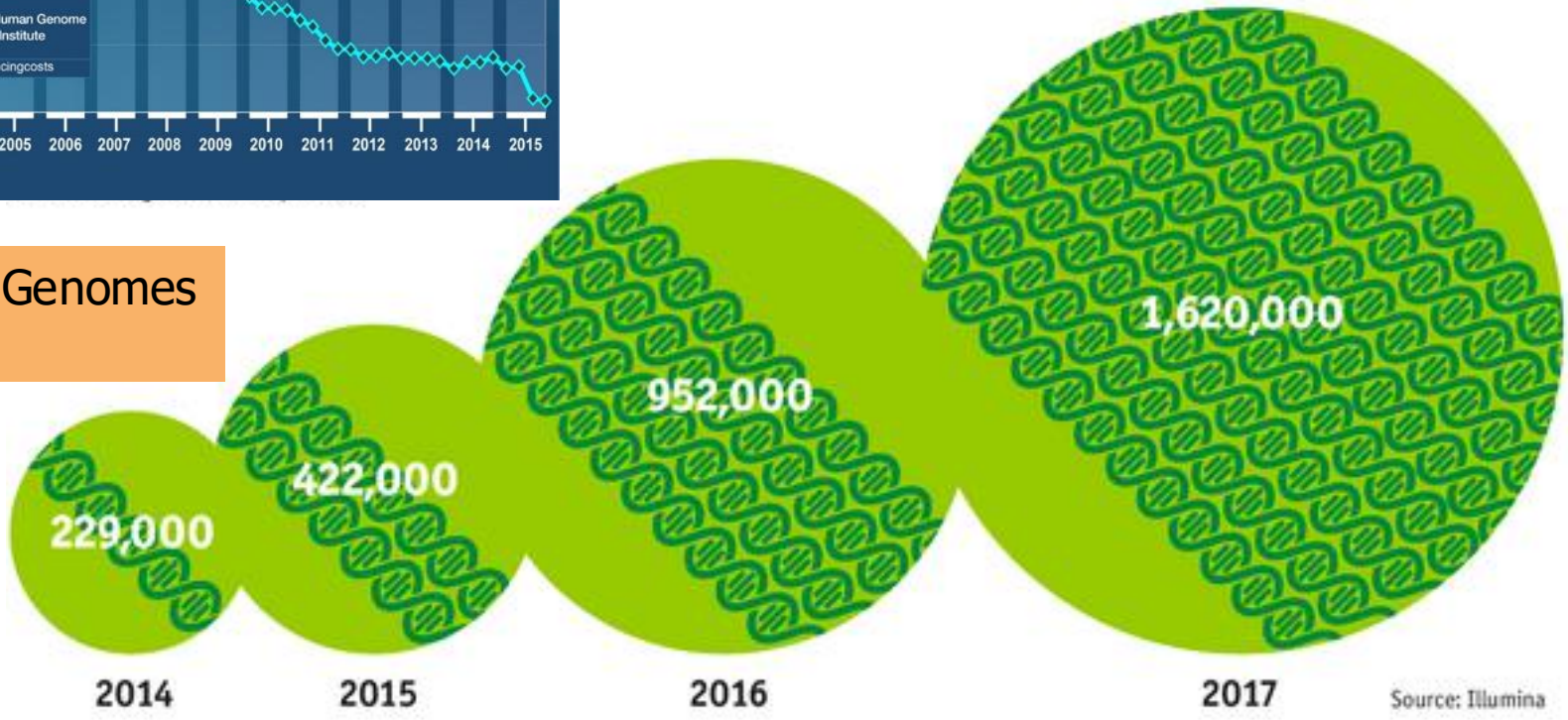
Huge Demand for Performance & Efficiency



development of new sequencing technologies

Oxford Nanopore MinION

Number of Genomes Sequenced



The Economist

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

The Problem

Data access is the major performance and energy bottleneck

Our current

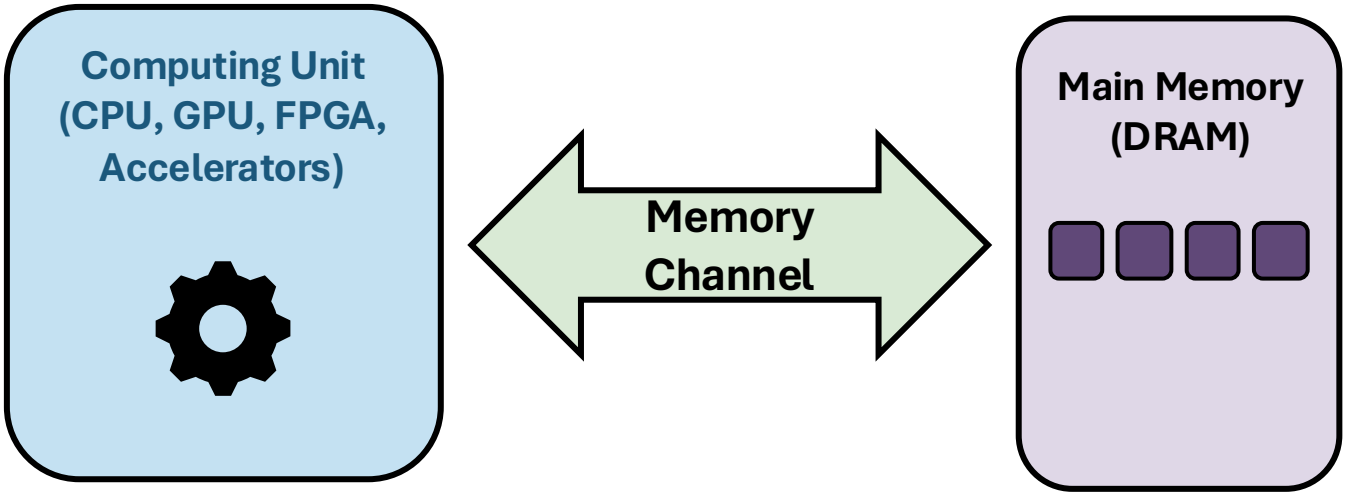
design principles

cause great energy waste

(and great performance loss)

Data Movement Bottleneck

- Today's computing systems are processor centric
- All data is processed in the processor → at great system cost

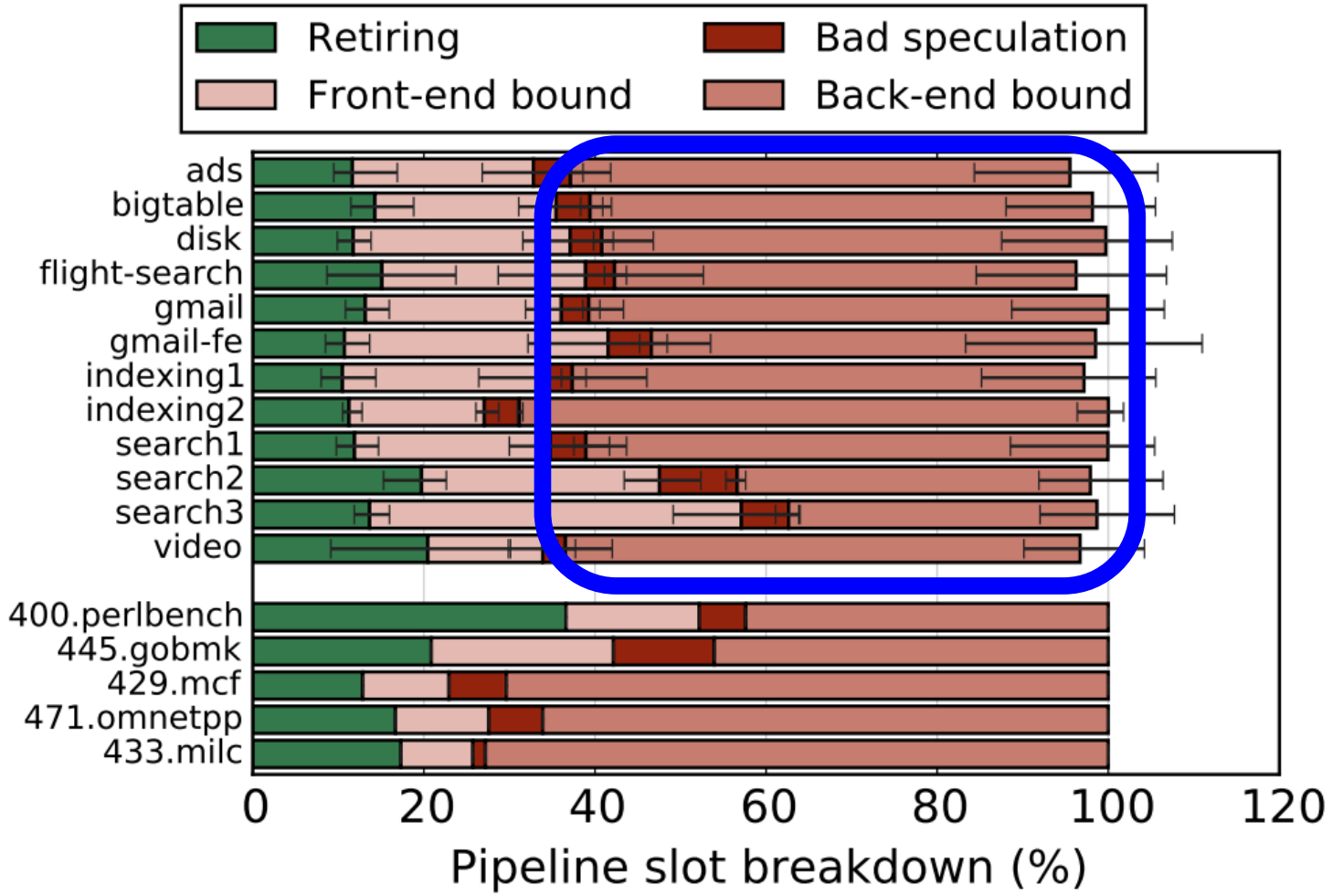


More than 60% of the total system energy is spent on data movement[1]

[1] A. Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS, 2018

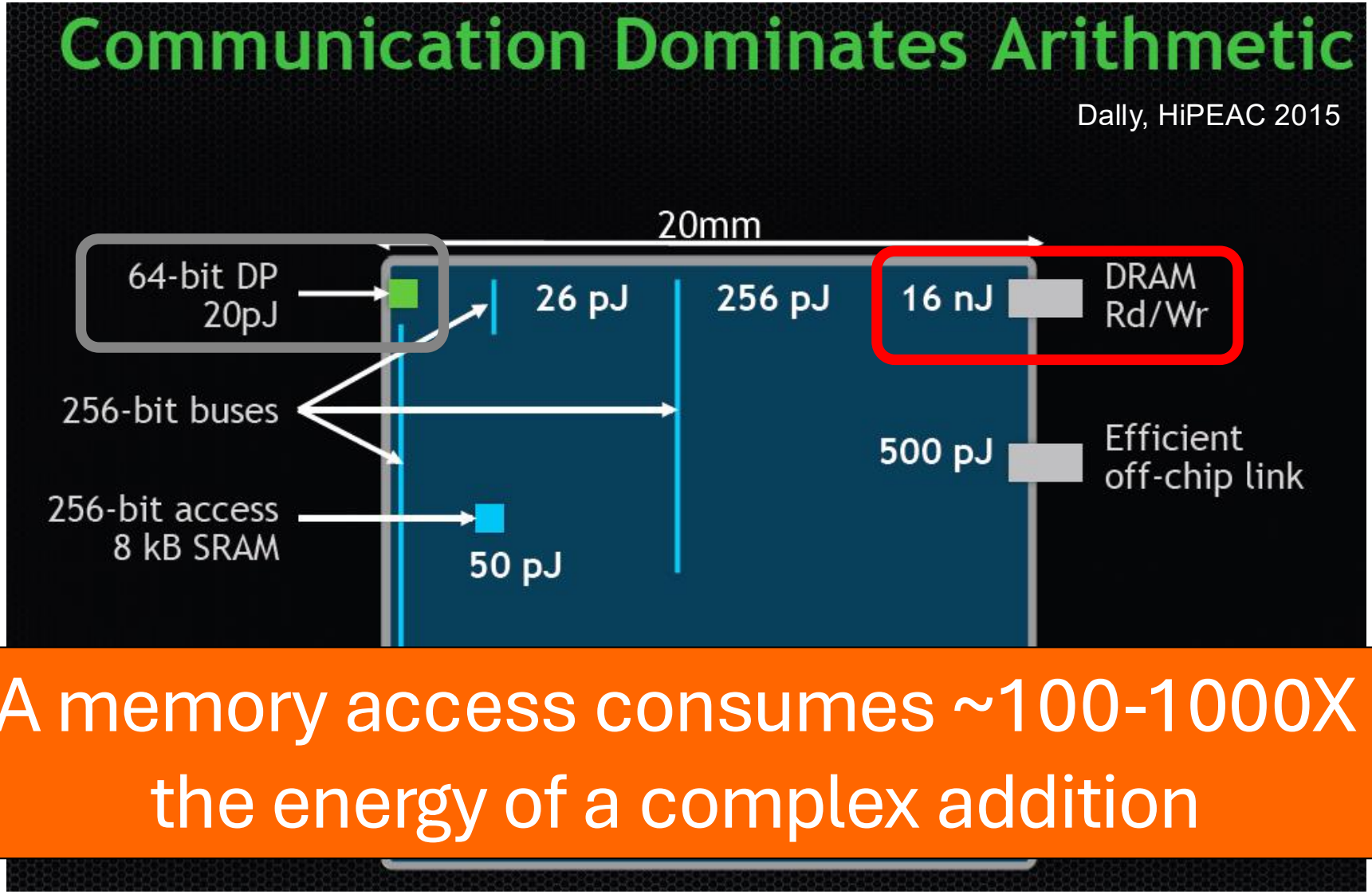
Processor-Centric System Performance

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



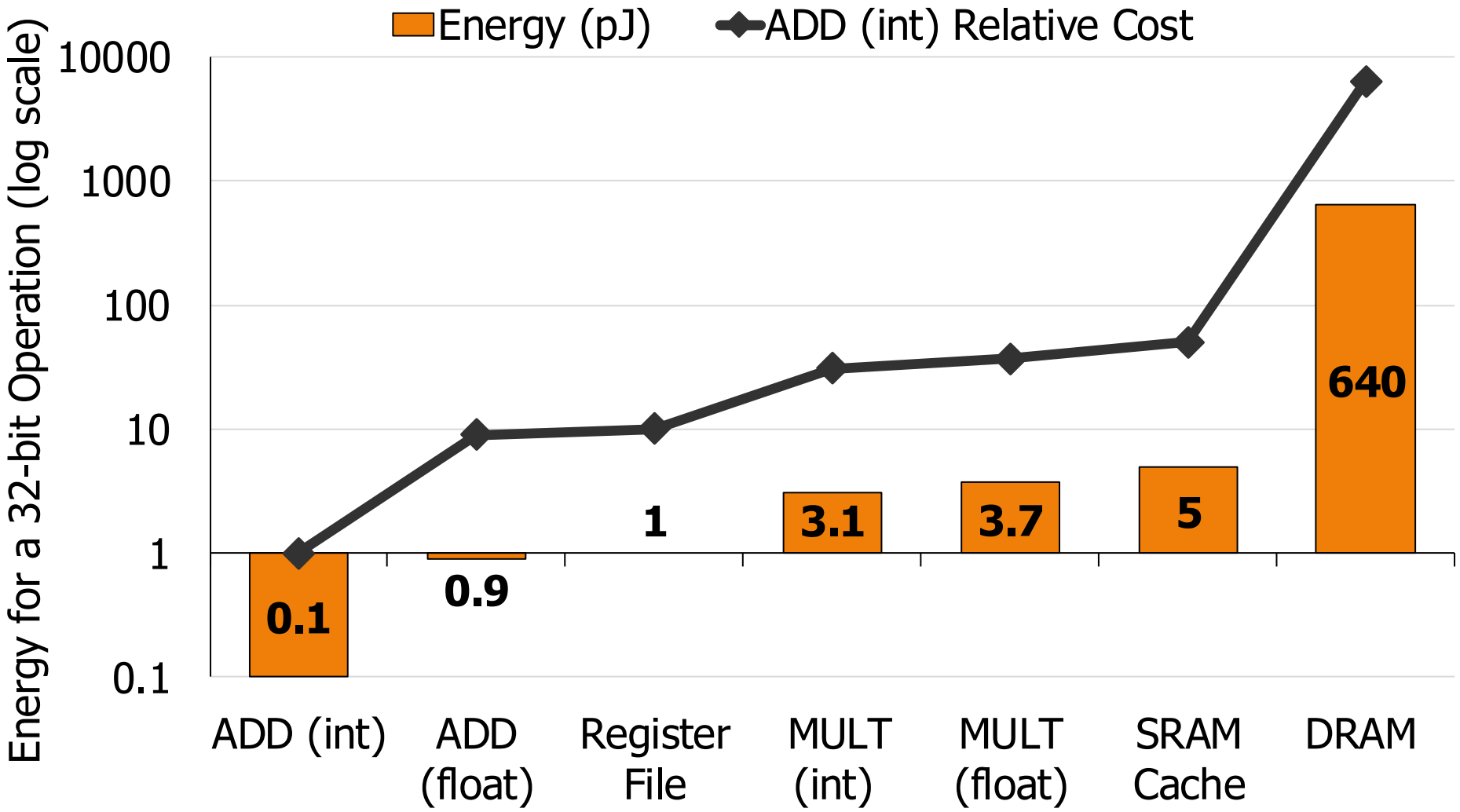
Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

Data Movement vs. Computation Energy



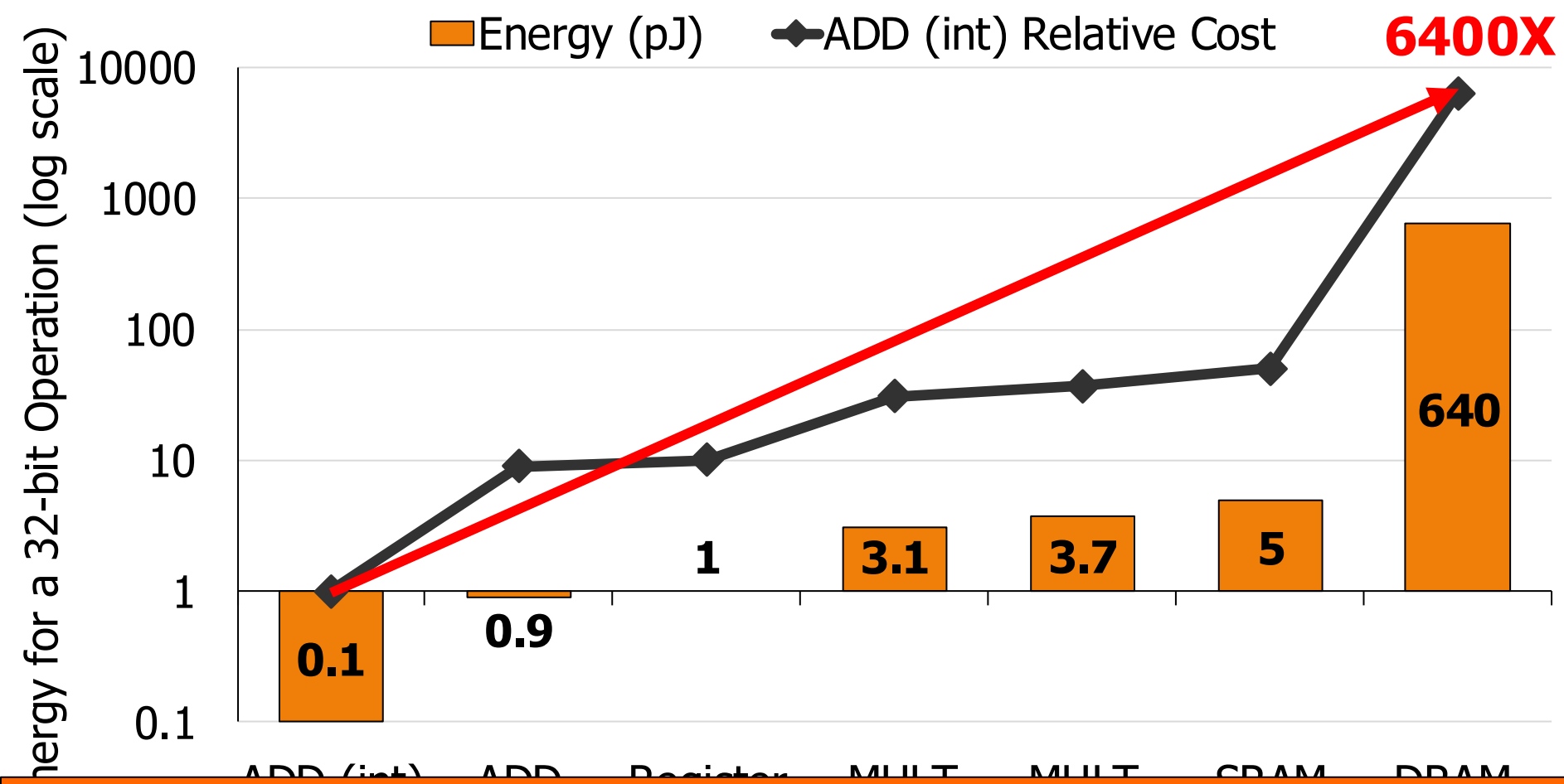
A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy



Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.

Data Movement vs. Computation Energy



A memory access consumes 6400X the energy of a simple integer addition

Energy Waste in Accelerators

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,
  "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"
  *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.

62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Energy Waste in Accelerators

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
  "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
  *Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (14 minutes)]

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

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

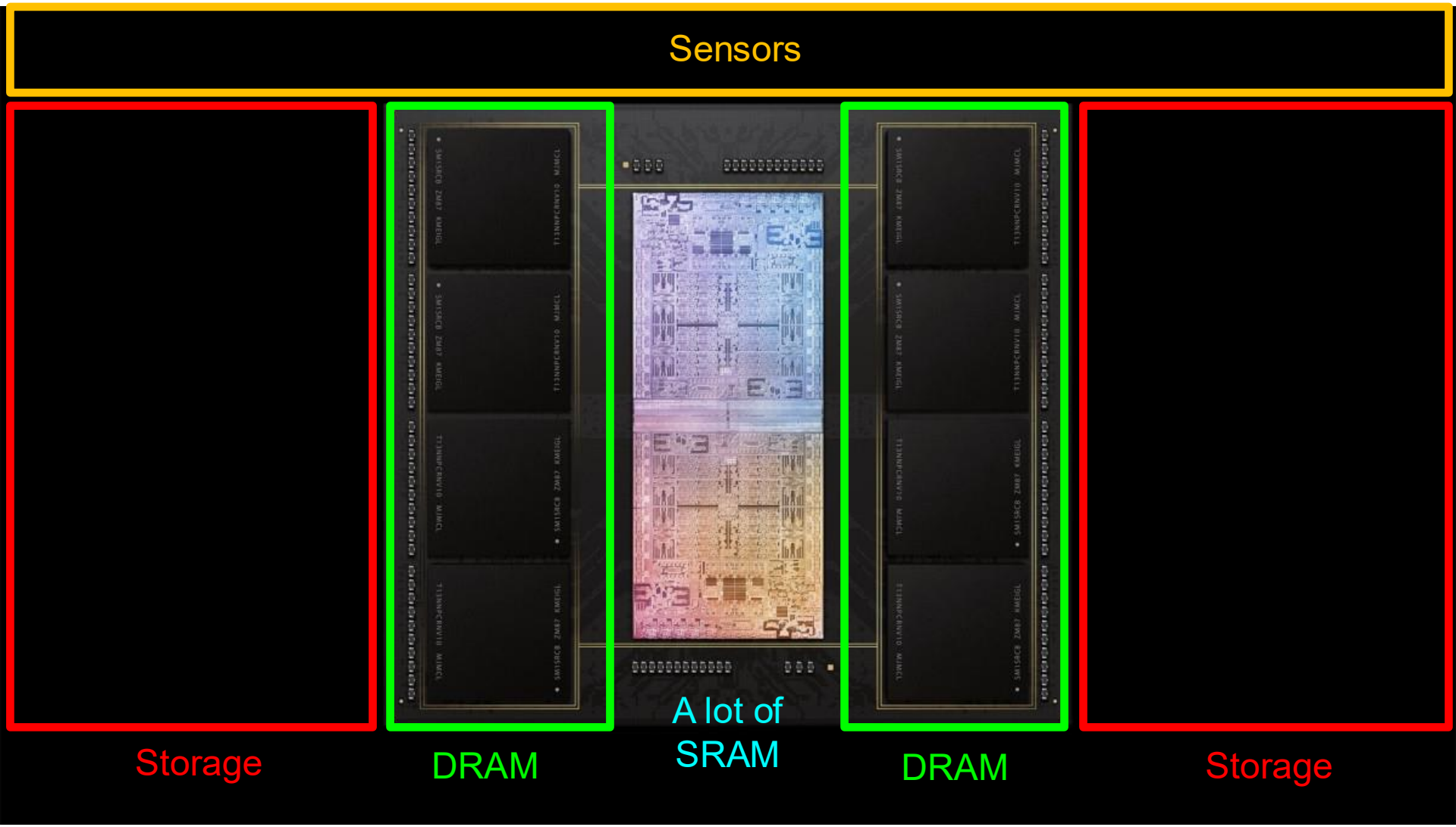
Fundamental Problem

Processing of data
is performed
far away from the data

We Need A Paradigm Shift To …

- Enable computation with minimal data movement
- Compute where it makes sense (where data resides)
- Make computing architectures more data-centric

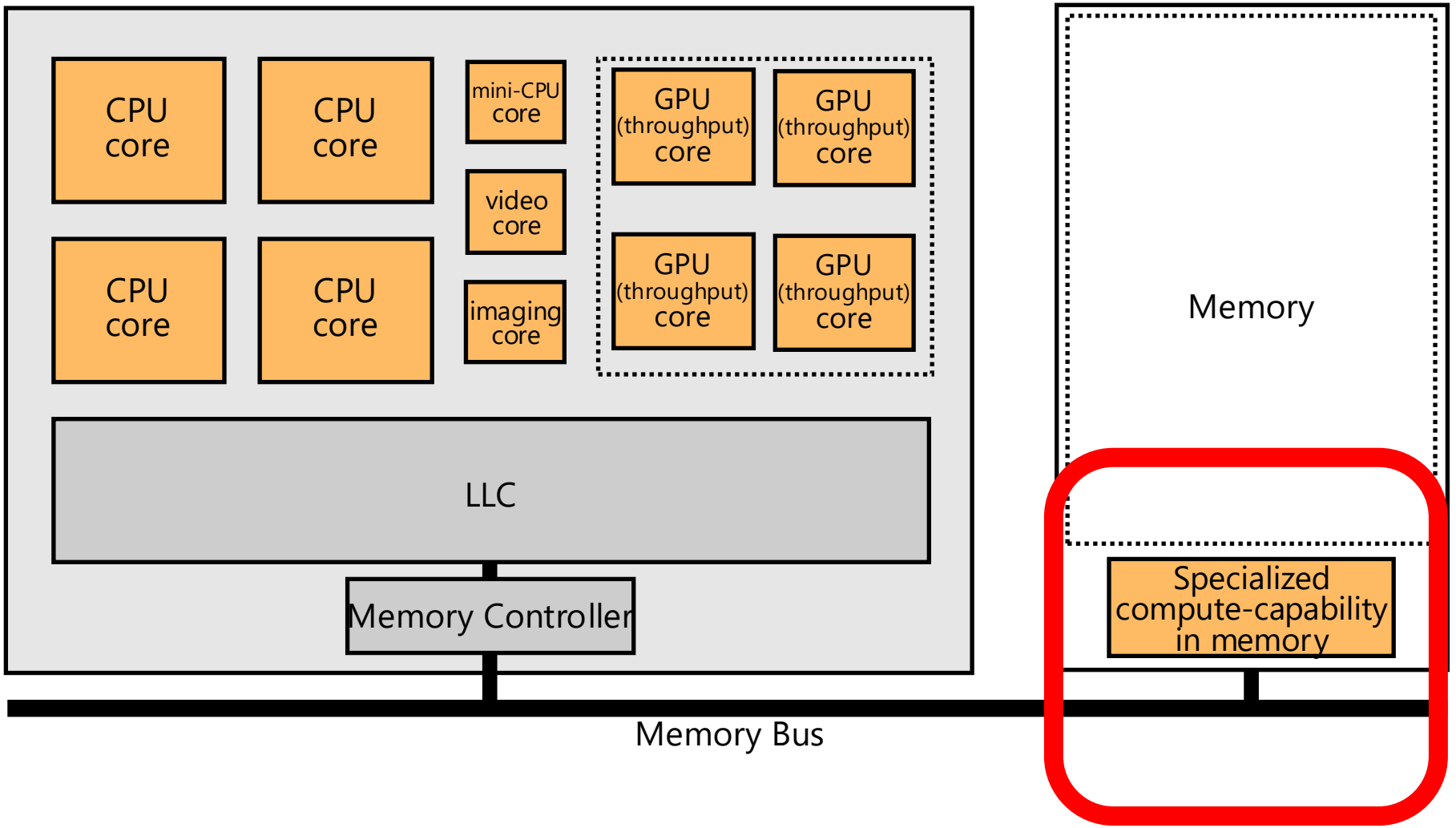
Process Data Where It Makes Sense



Apple M1 Ultra System (2022)

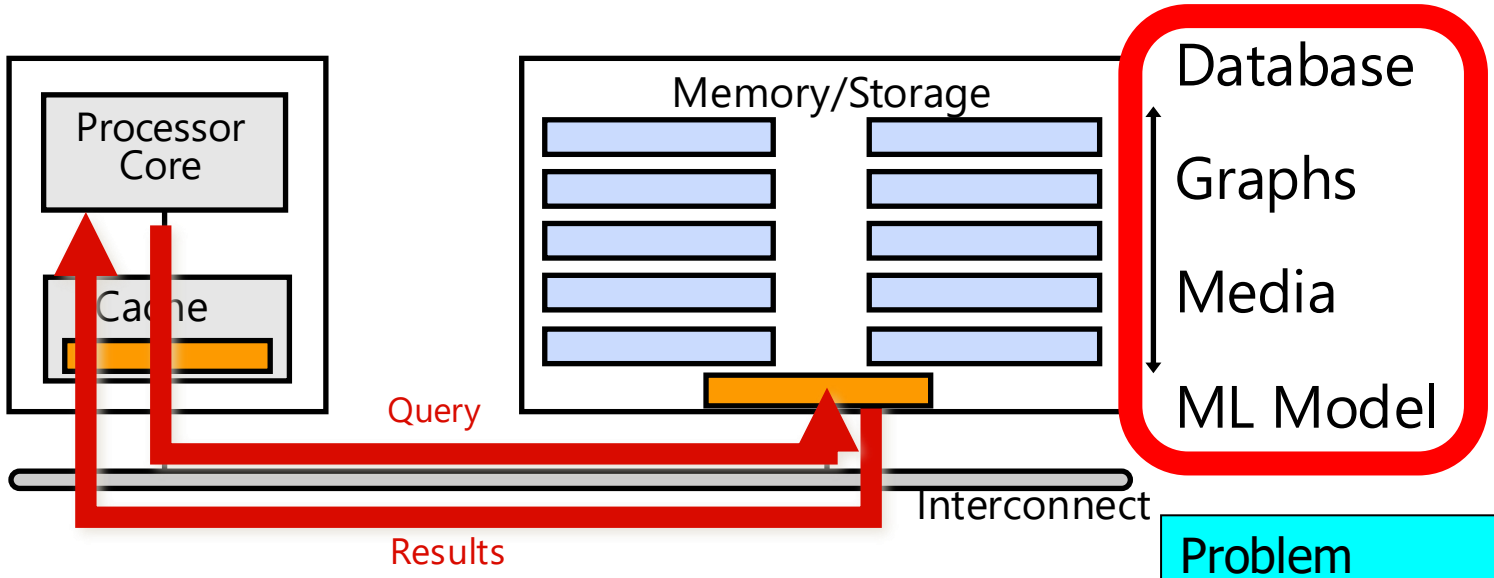
https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php

Memory as an Accelerator

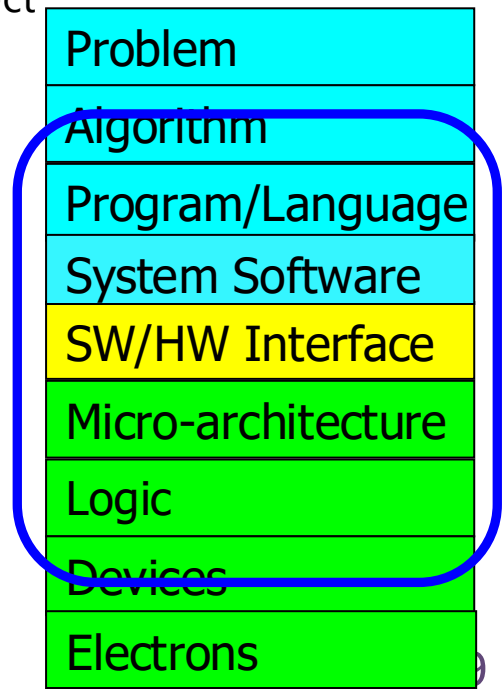


Memory similar to a "conventional" accelerator

Goal: Processing Inside Memory/Storage



- Many questions … How do we design the:
  - compute-capable memory & controllers?
  - processors & communication units?
  - software & hardware interfaces?
  - system software, compilers, languages?
  - algorithms & theoretical foundations?



Processing in/near Memory: An Old Idea

- Kautz, "Cellular Logic-in-Memory Arrays", IEEE TC 1969.

IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be "programmed" to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.

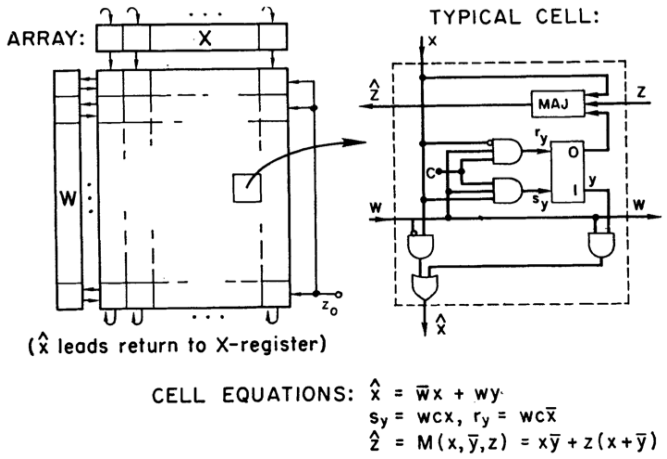


Fig. 1. Cellular sorting array I.

https://doi.org/10.1109/T-C.1969.222754

Processing in/near Memory: An Old Idea

- Stone, "A Logic-in-Memory Computer," IEEE TC 1970.

A Logic-in-Memory Computer

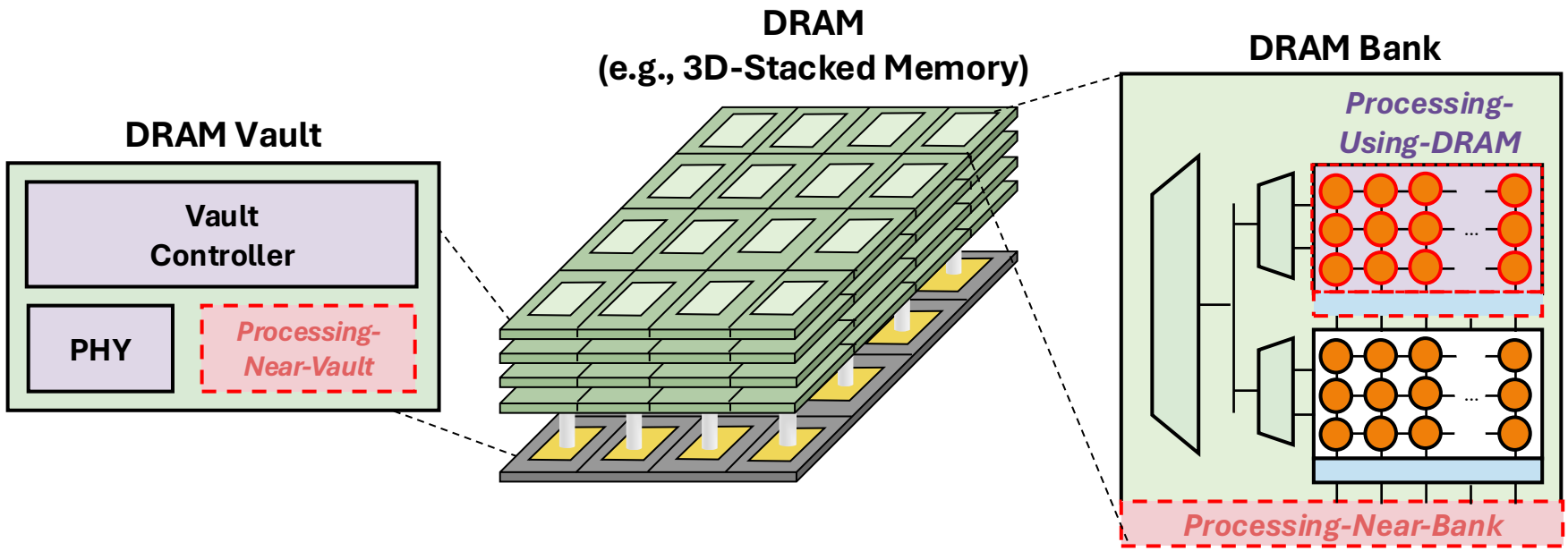
HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.

Processing-in-Memory: Two Types

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

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

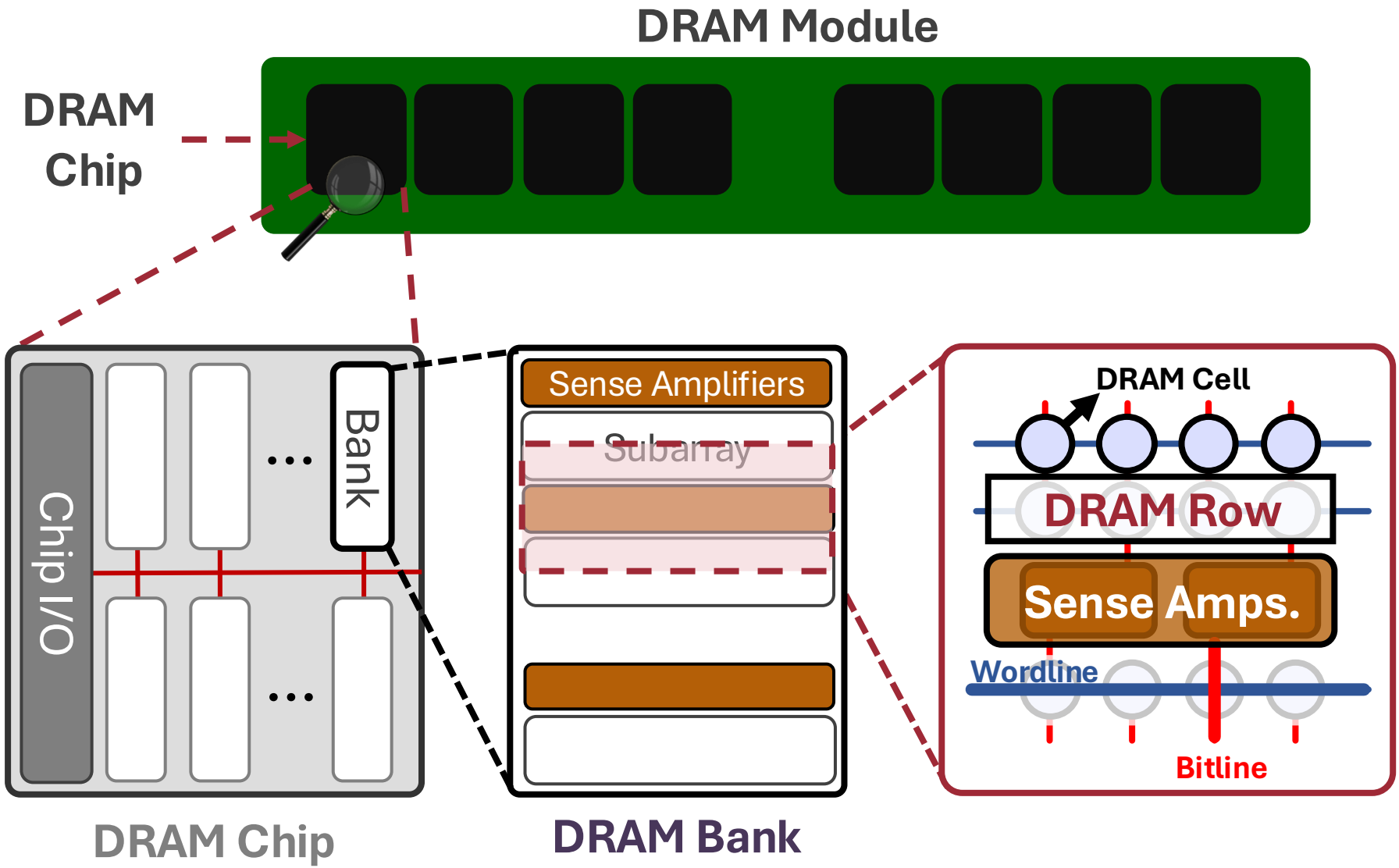


Focus: Processing-using-DRAM

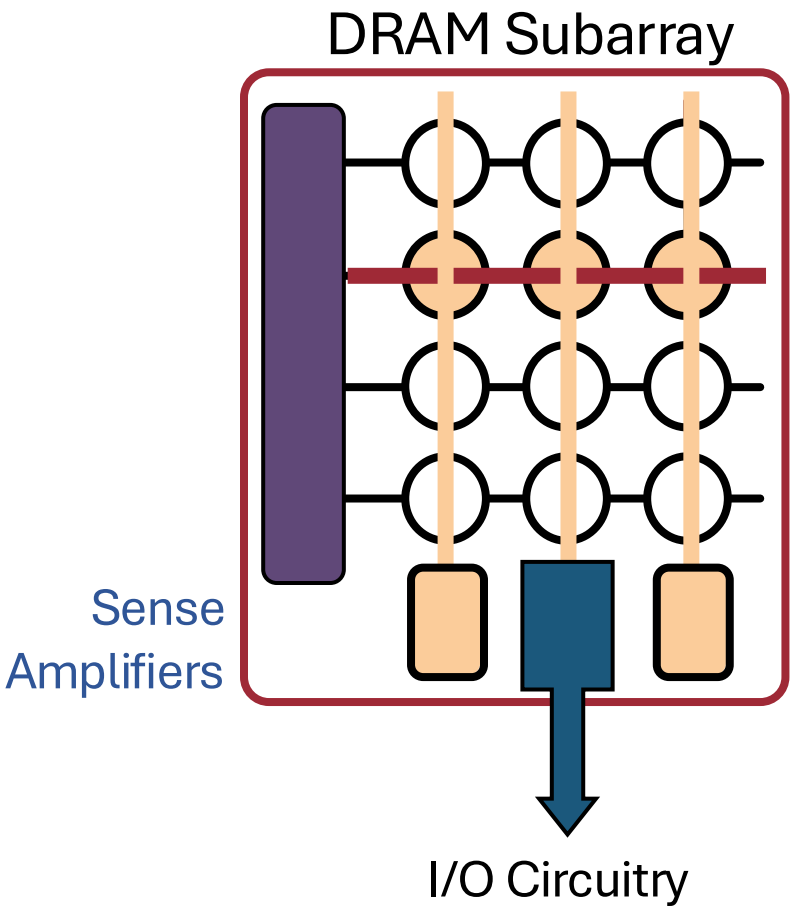
- We can natively support
  - Bulk bitwise COPY and INIT/ZERO
  - Bulk bitwise AND, OR, NOT, MAJ, NOR, NAND
  - True Random Number Generation; Physical Unclonable Functions
  - More complex computation using Lookup Tables
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating (multiple) rows performs computation
    - Even in commodity off-the-shelf DRAM chips!
- 30X-257X performance and energy improvements

- Seshadri+"RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.
- Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.
- Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
- Hajinazar+, "SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM," ASPLOS 2021.
- Oliveira+, "MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Processing," HPCA 2024.

DRAM Organization



Major DRAM Operations

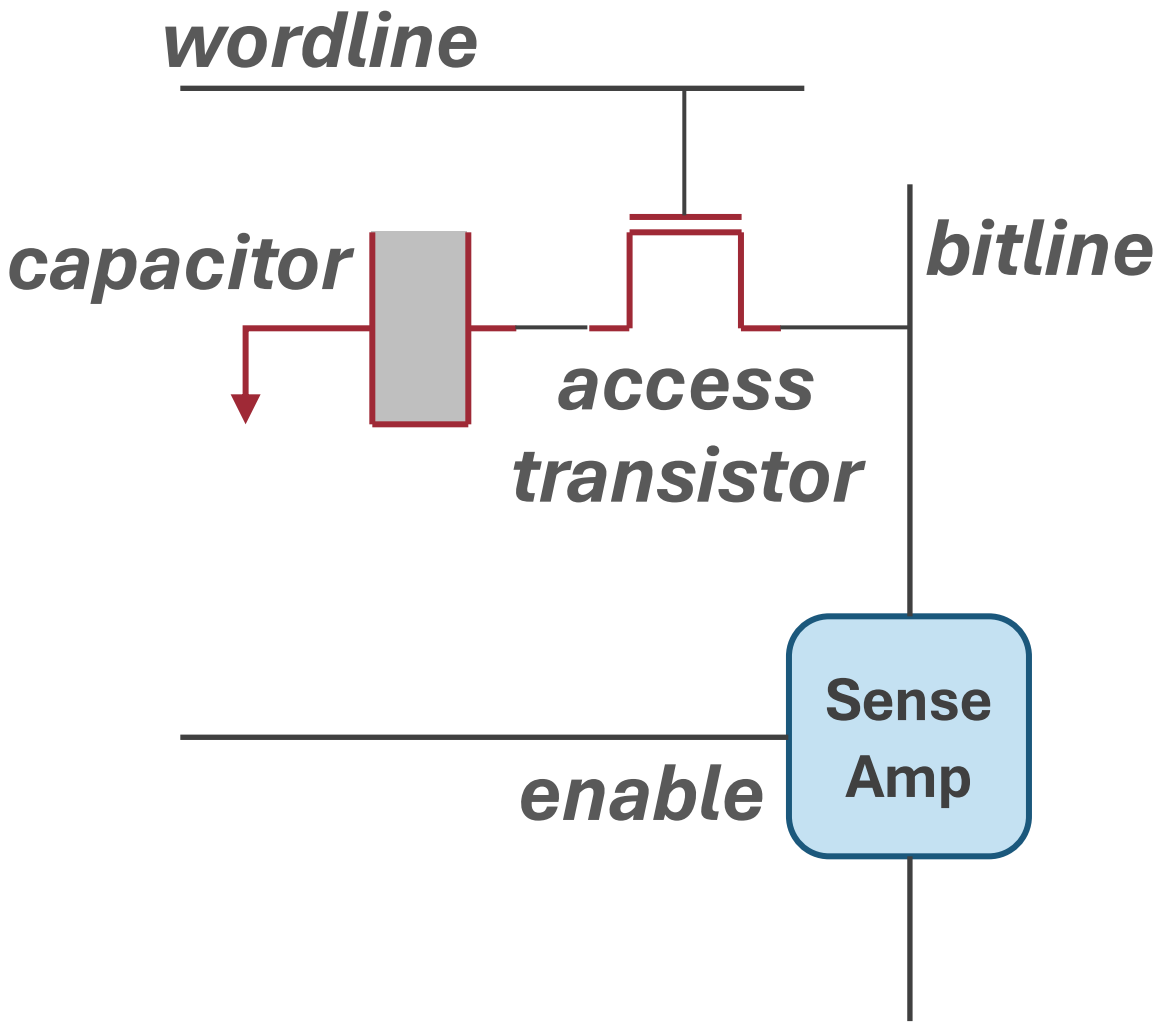


1 **ACTIVATE (ACT):**
Fetch the row's content into the **sense amplifiers**

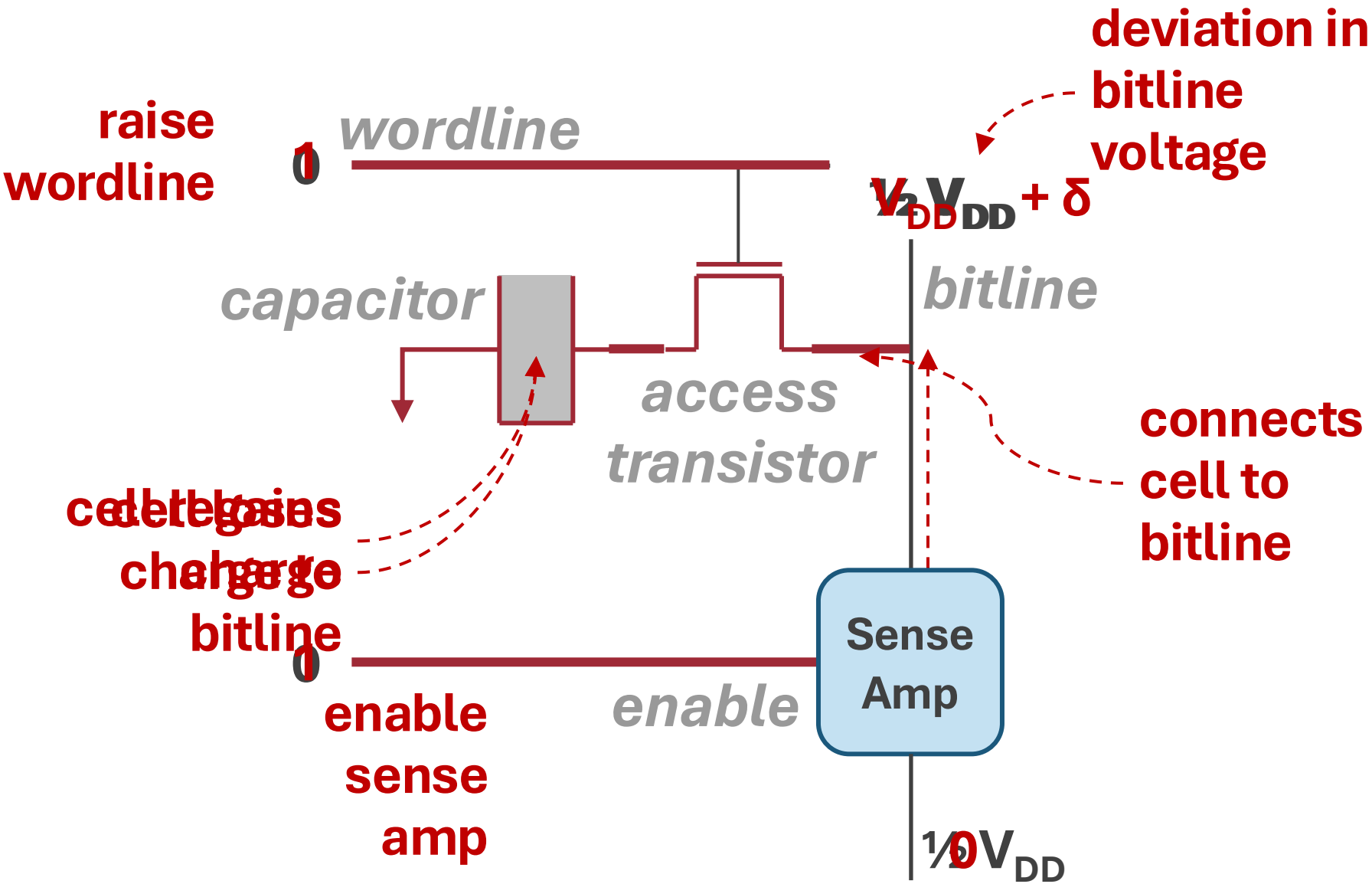
2 **Column Access (RD/WR):**
Read/Write the target column and drive to I/O

3 **PRECHARGE (PRE):**
Prepare the bank for a new ACTIVATE

Closer Look to a DRAM Cell

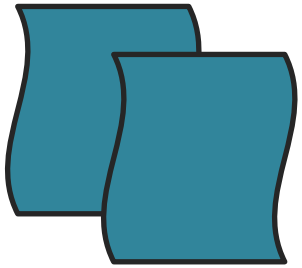


DRAM Cell Operation upon ACT

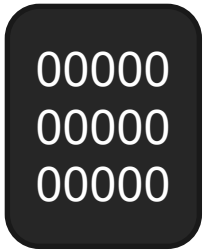


Starting Simple: Data Copy and Initialization

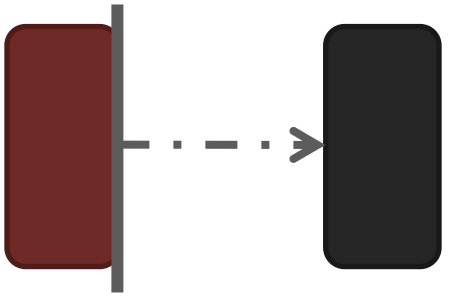
memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

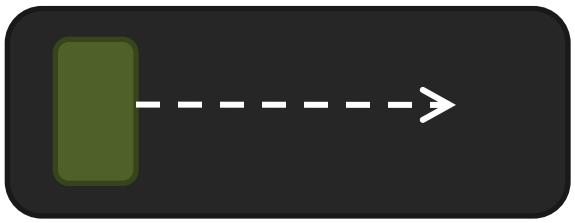


Zero initialization (e.g., security)



Checkpointing

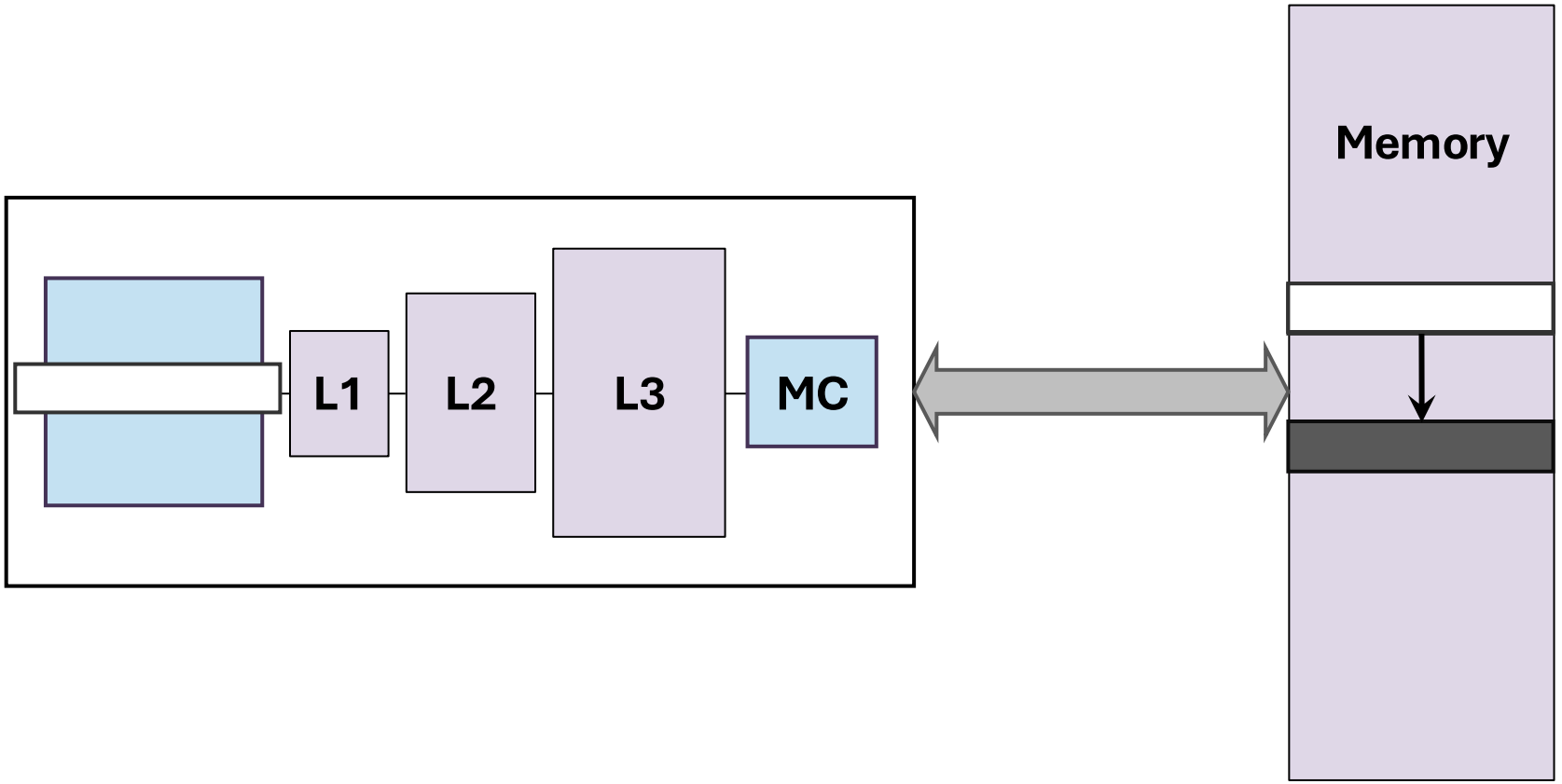
VM Cloning
Deduplication



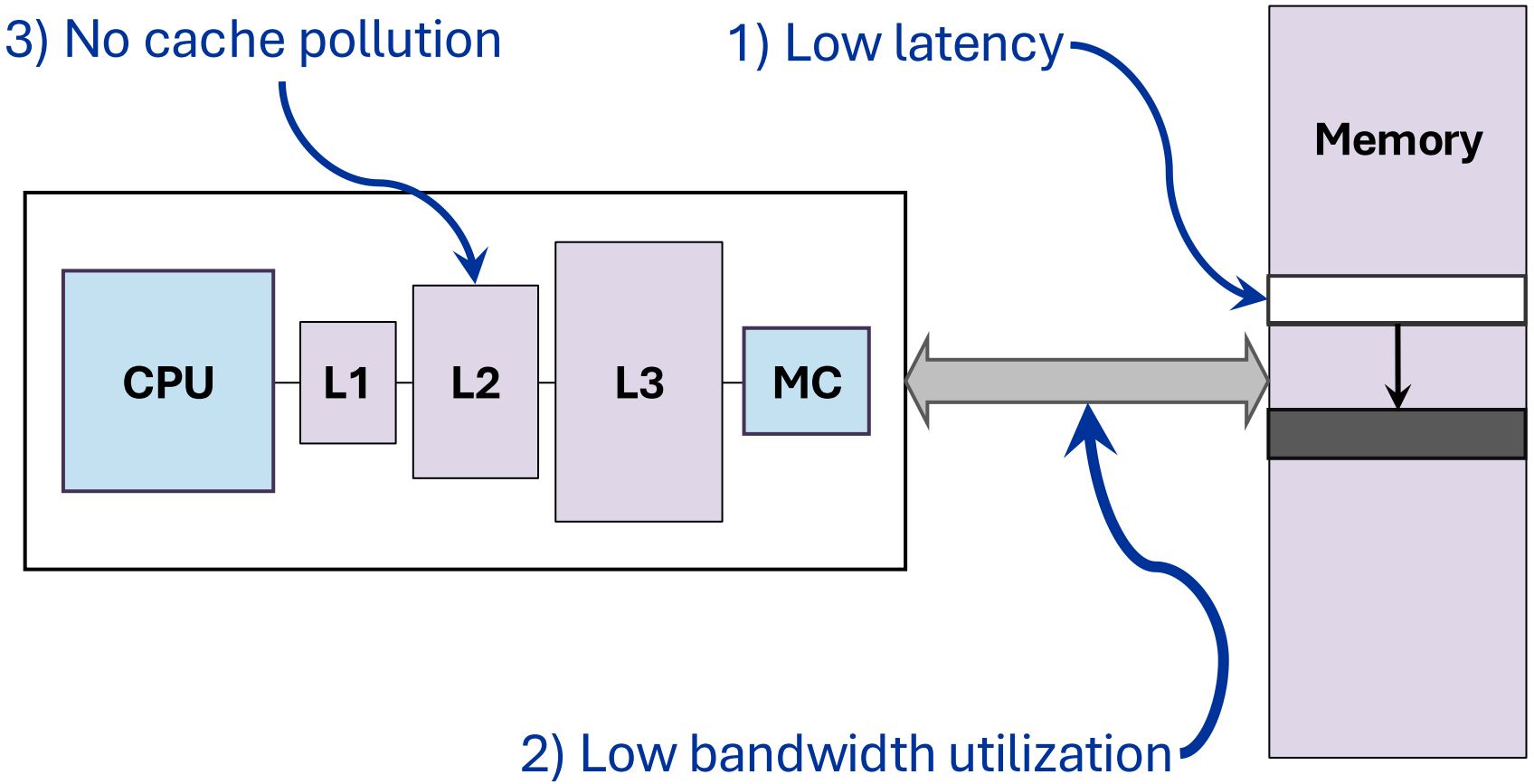
Page Migration

• • •
Many more

Current Systems: Data Copy



Future Systems: In-Memory Copy

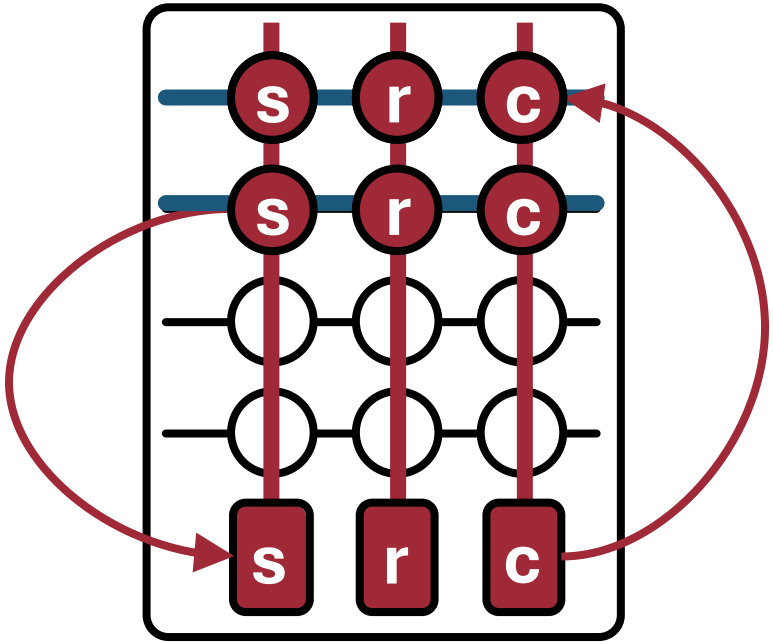


4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ

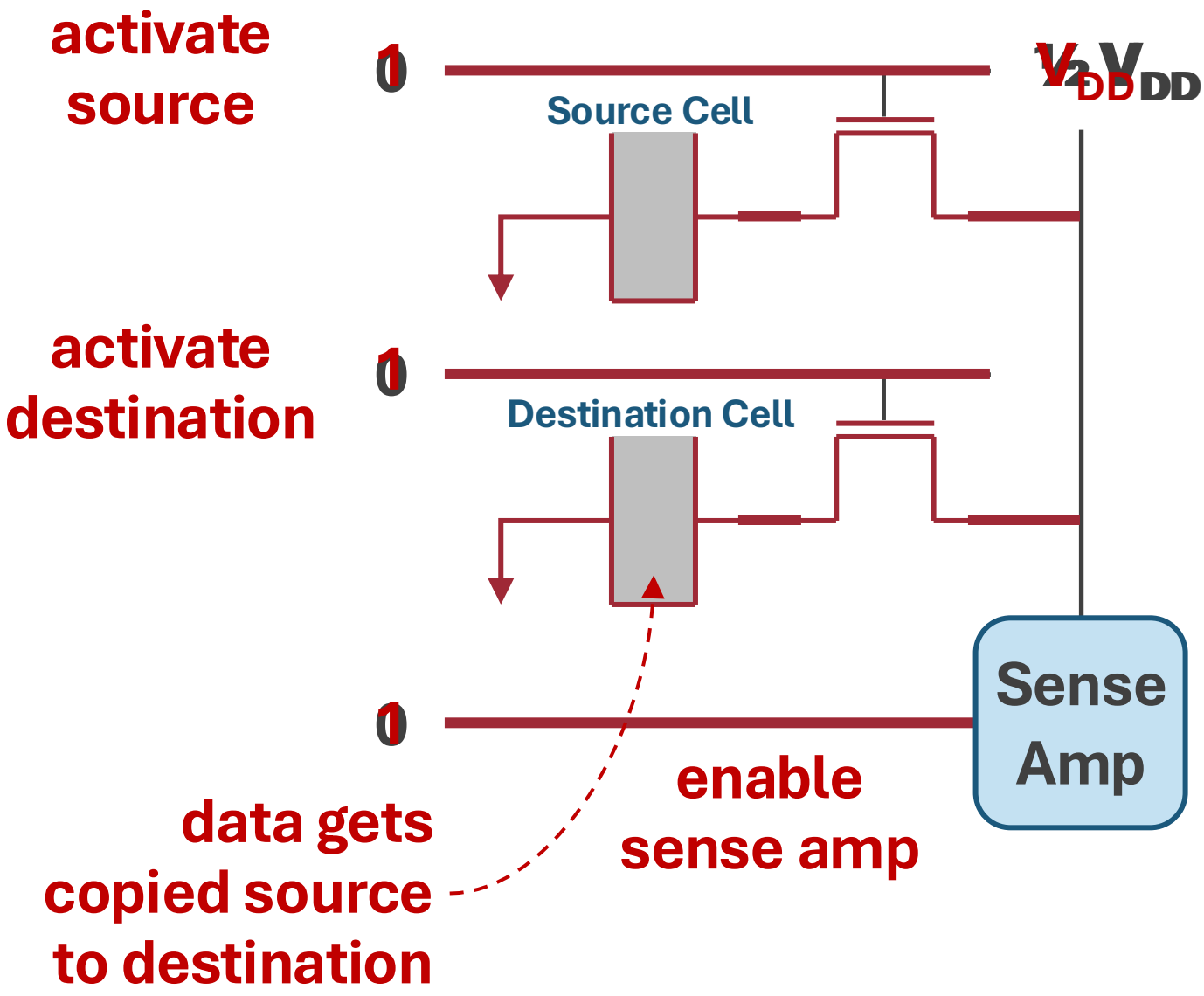
In-DRAM Row-Copy (RowClone)

Copying the source (src) row's content to the destination (dst) row

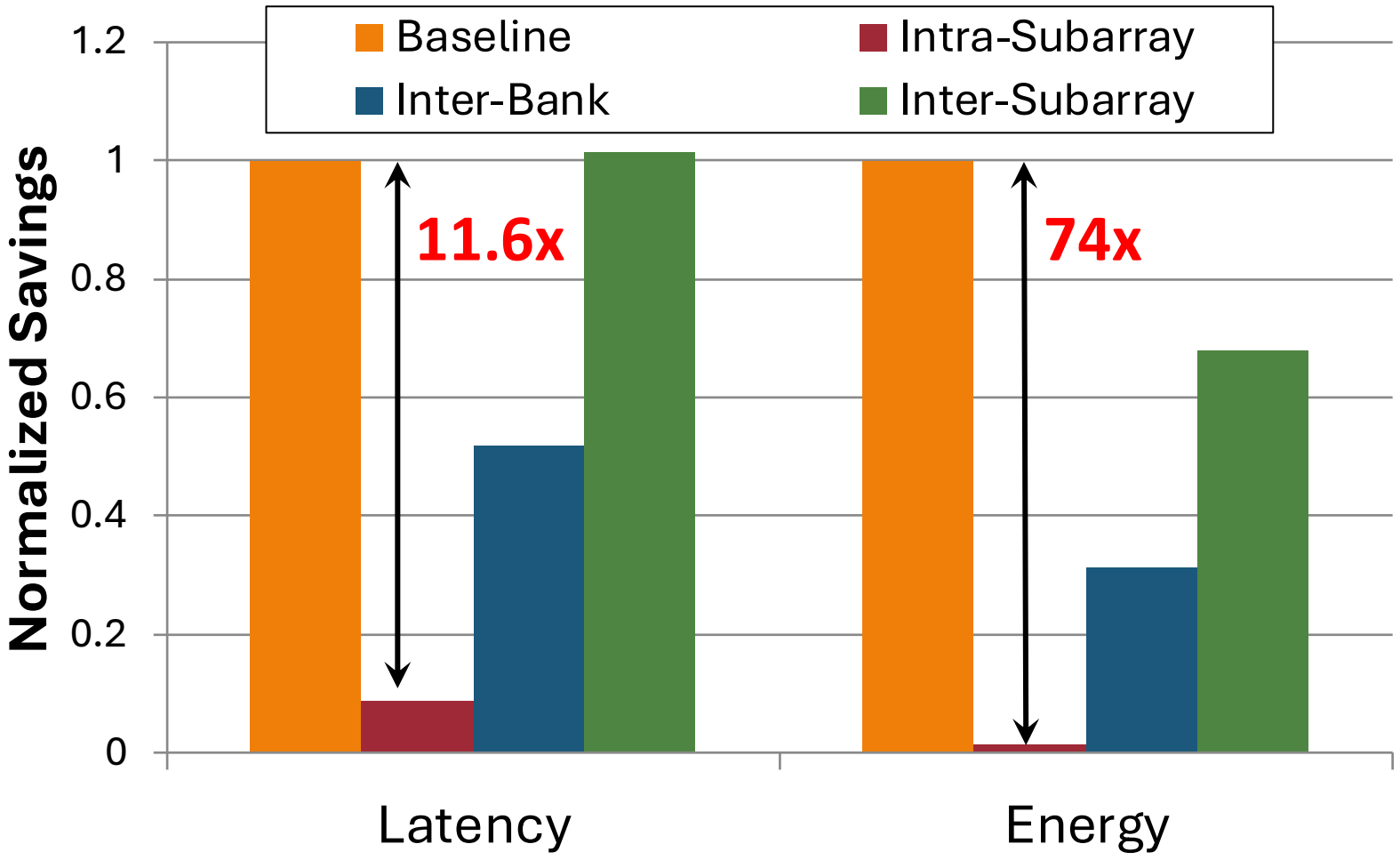


[Seshadri+ MICRO'13]

RowClone: In-DRAM Bulk Data Copy



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

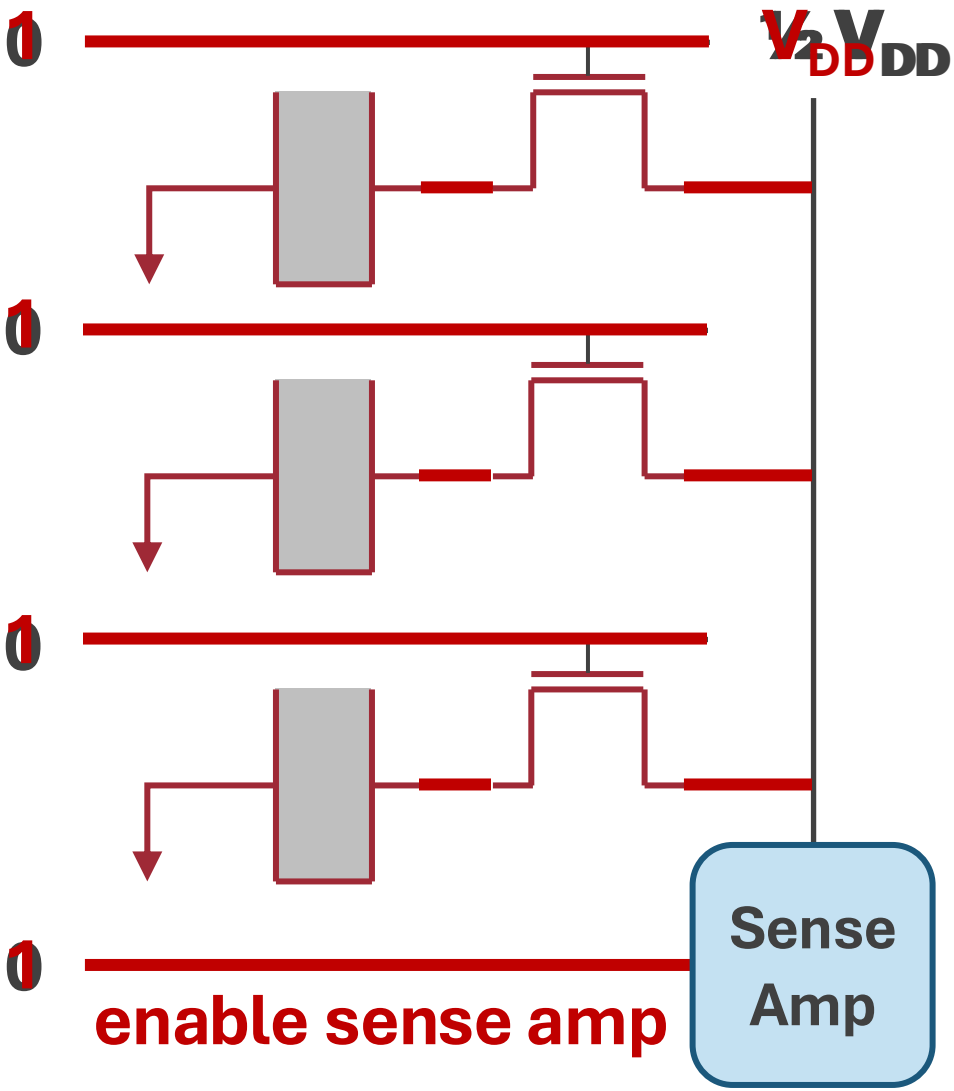
- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,
  "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"
  Proceedings of the 46th International 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

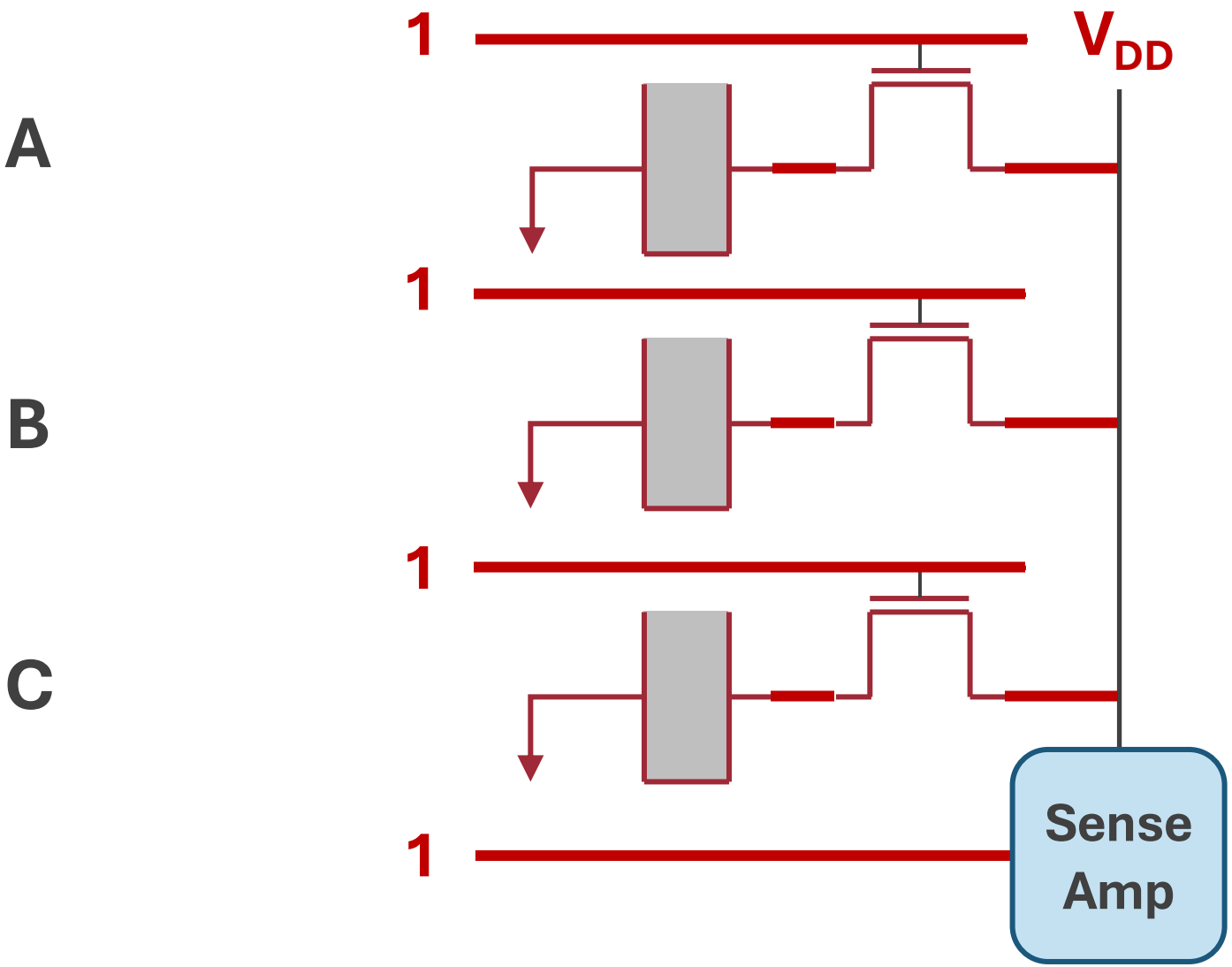
Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com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

Triple-Row Activation: Majority Function

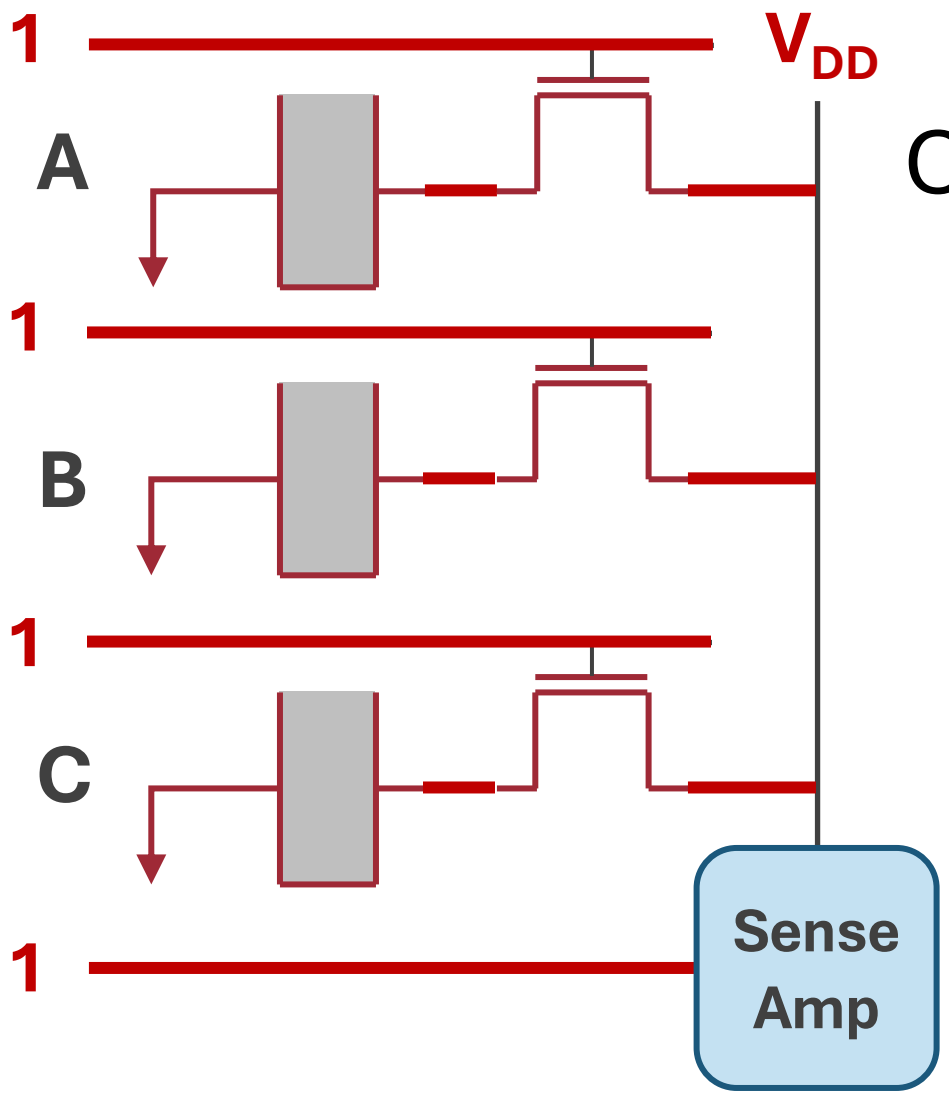
activate
all three rows
simultaneously



Bitwise AND/OR Using Triple-Row Activation



Bitwise AND/OR Using Triple-Row Activation



Output = MAJ(A, B, C)

= C (A **OR** B) +

~C (A **AND** B)

Control the value of C to perform bitwise OR or AND

In-DRAM Acceleration of Database Queries

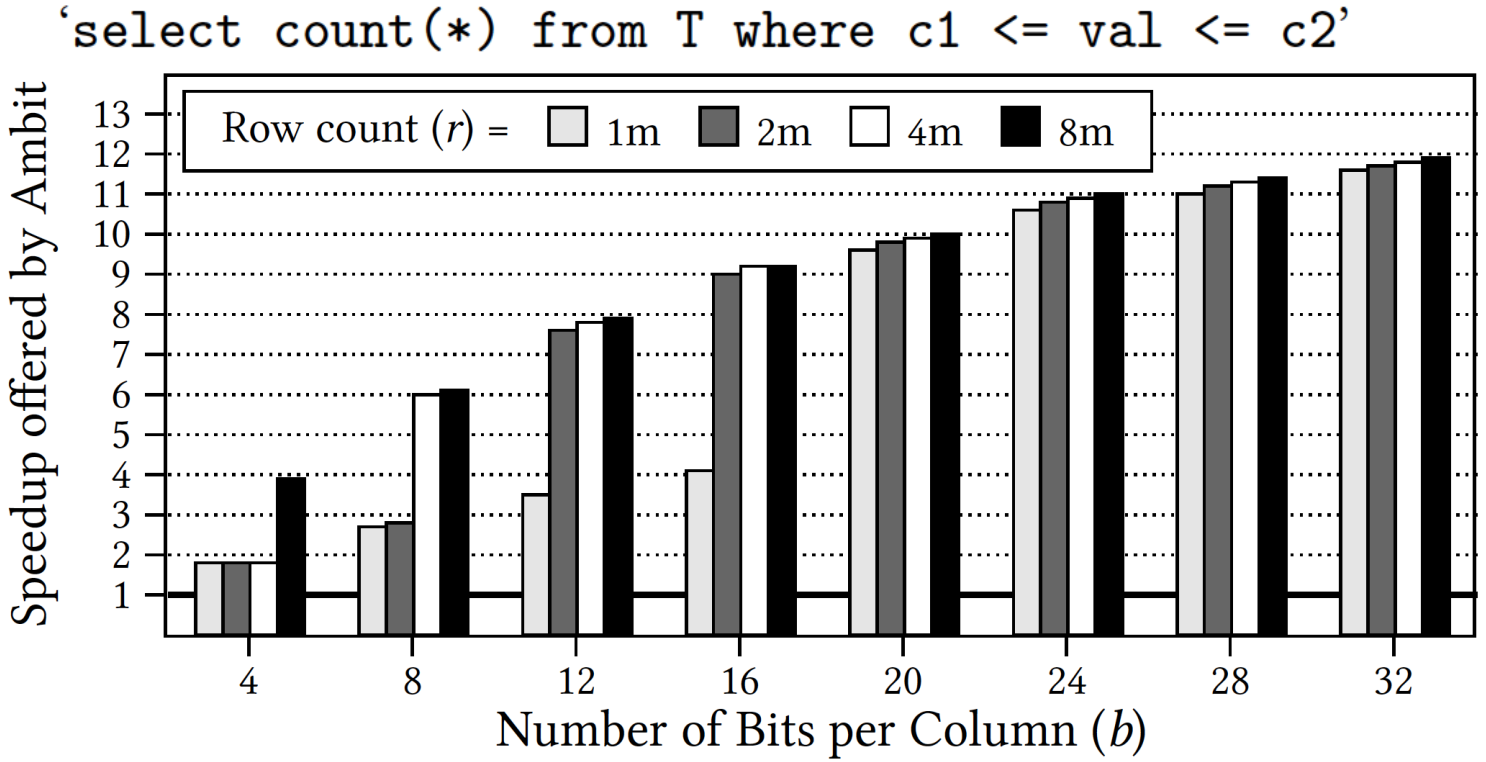


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

>4-12X Performance Improvement

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 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

SIMDRAM Framework

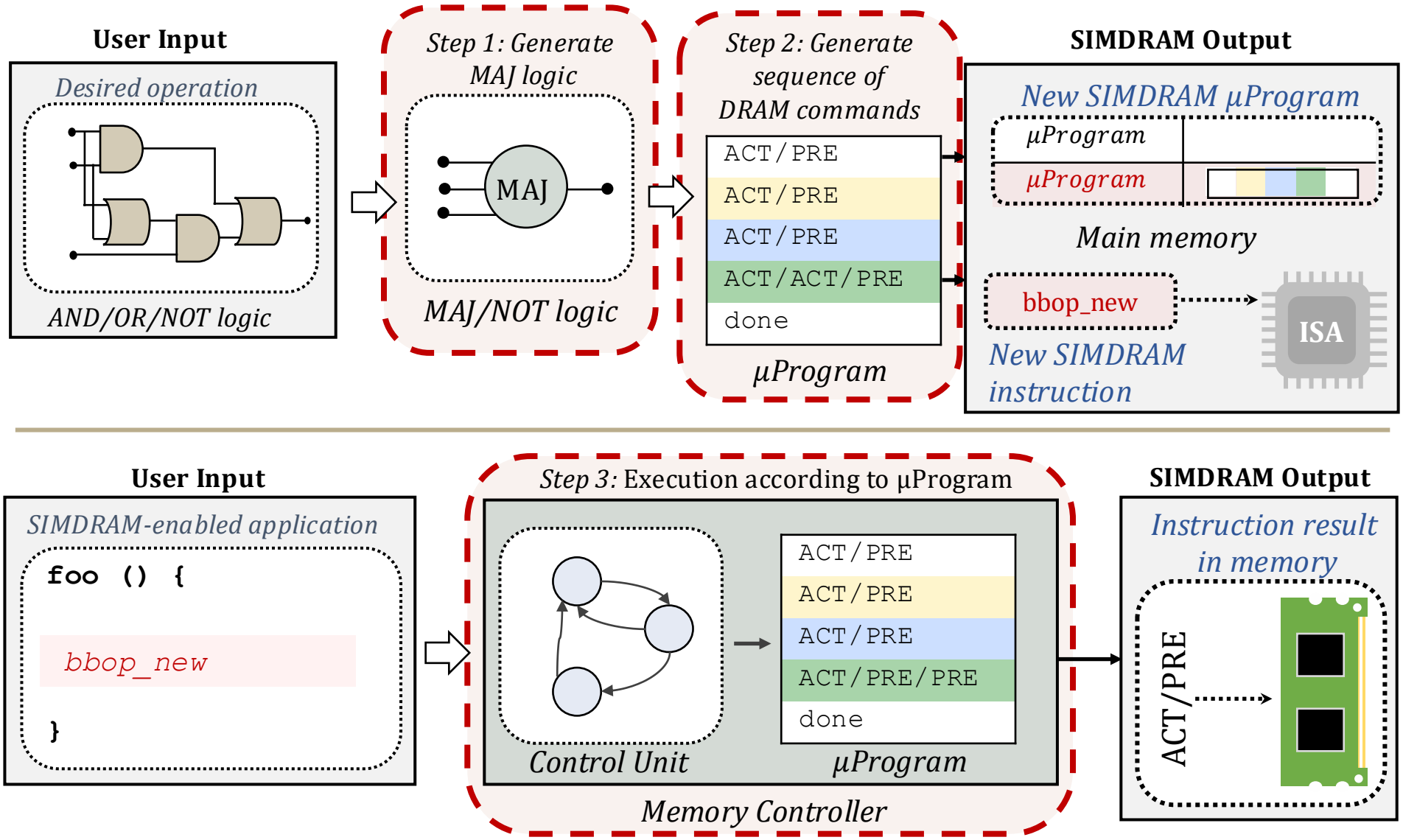
- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu,
  "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"
  *Proceedings of the 26th International Conference 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 Framework: Overview



SIMDRAM Key Results

Large improvements over **CPU** & **high-end GPU:**

Throughput: 88× and 5.8×

(16 complex operations)

Energy: 257× and 31×

(16 complex operations)

Application Performance: 21× and 2.1×

(seven common real-world applications)

More on SIMDRAM

- 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

MIMDRAM: More Flexible Processing using DRAM

- **Appears at HPCA 2024**

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

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

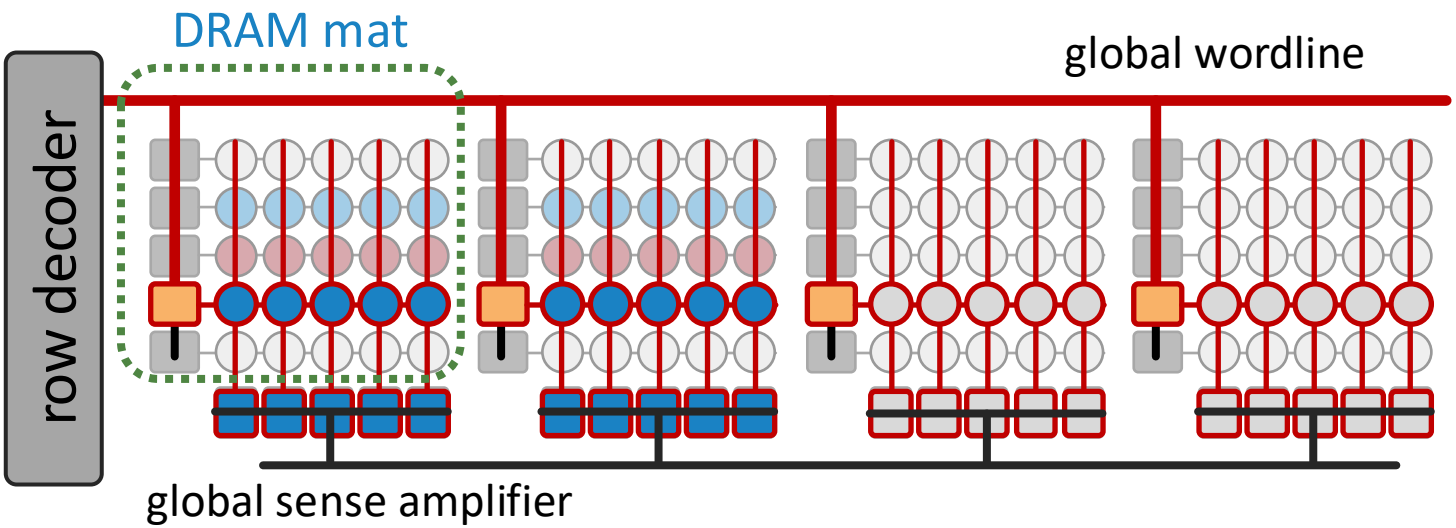
[†] *ETH Zürich* [‡] *Univ. of Illinois Urbana-Champaign*

*Our **goal** is to design a flexible PUD system that overcomes the limitations caused by the large and rigid granularity of PUD. To this end, we propose MIMDRAM, a hardware/software co-designed PUD system that introduces new mechanisms to allocate and control only the necessary resources for a given PUD operation. The key idea of MIMDRAM is to leverage fine-grained DRAM (i.e., the ability to independently access smaller segments of a large DRAM row) for PUD computation. MIMDRAM exploits this key idea to enable a multiple-instruction multiple-data (MIMD) execution model in each DRAM subarray (and SIMD execution within each DRAM row segment).*

https://arxiv.org/pdf/2402.19080.pdf

MIMDRAM: Key Idea (I)

Enable narrower-width operations than a DRAM row



Key Issue:

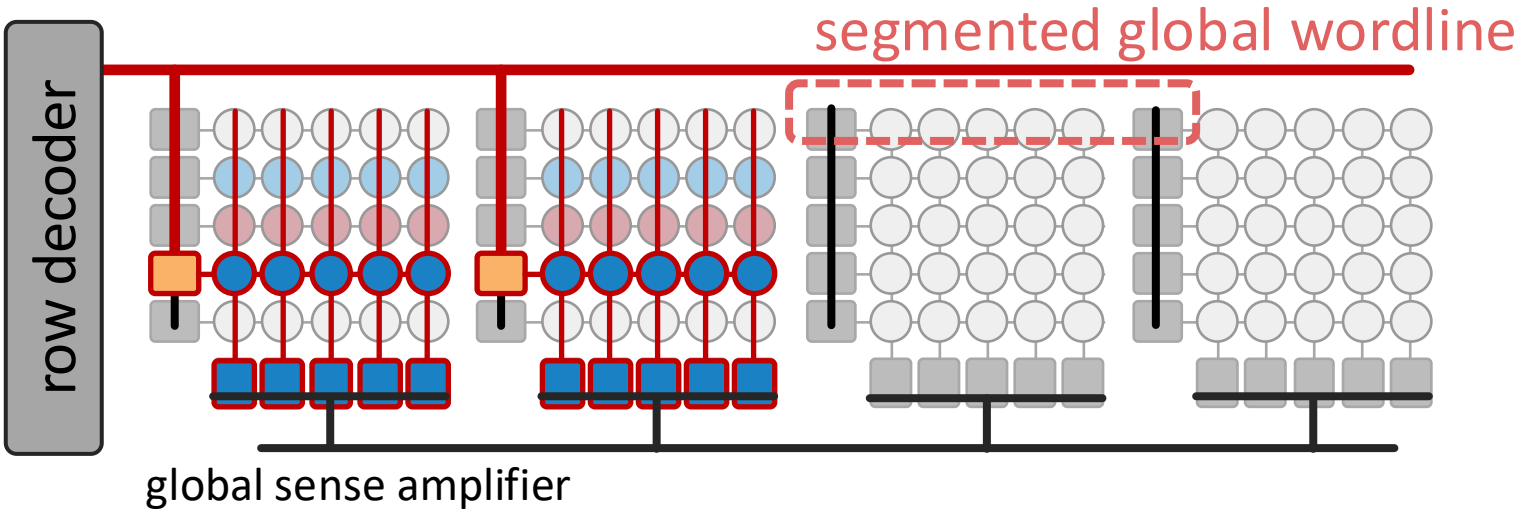
on a DRAM access, the global wordline propagates across all DRAM mats

Fine-Grained DRAM:

segment the global wordline to access individual DRAM mats

MIMDRAM: Key Idea (II)

Fine-Grained DRAM:
segment the global wordline to access individual DRAM mats

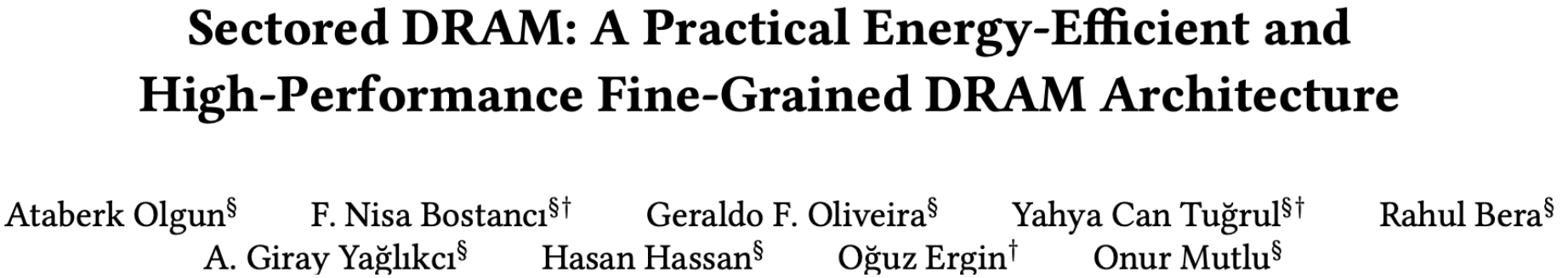


Fine-grained DRAM for energy-efficient DRAM access:

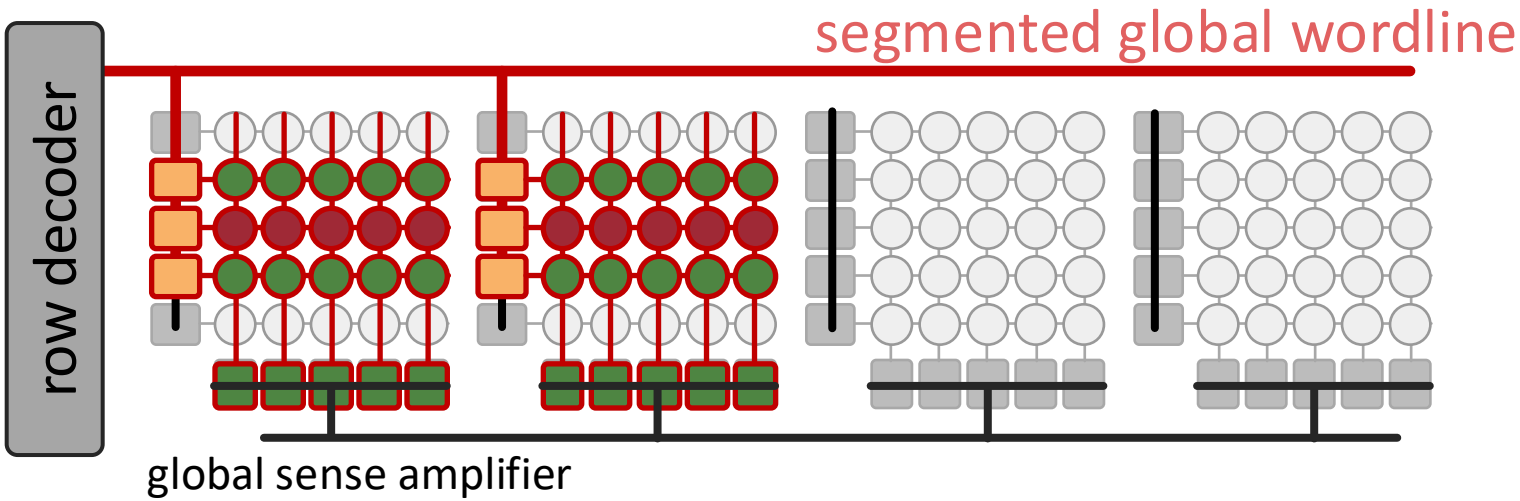
[Cooper-Balis+, 2010]: Fine-Grained Activation for Power Reduction in DRAM
[Udipi+, 2010]: Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores
[Zhang+, 2014]: Half-DRAM
[Ha+, 2016]: Improving Energy Efficiency of DRAM by Exploiting Half Page Row Access
[O'Connor+, 2017]: Fine-Grained DRAM
[Olgun+, 2024]: Sectored DRAM

Sectored DRAM

- Ataberk Olgun, F. Nisa Bostanci, Geraldo F. Oliveira, Yahya Can Tugrul, Rahul Bera, A. Giray Yaglikci, Hasan Hassan, Oguz Ergin, and Onur Mutlu, **"Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture"**
  ACM Transactions on Architecture and Code Optimization (***TACO***), [online] June 2024.
  [arXiv version]
  [ACM Digital Library version]



MIMDRAM: Key Idea (III)

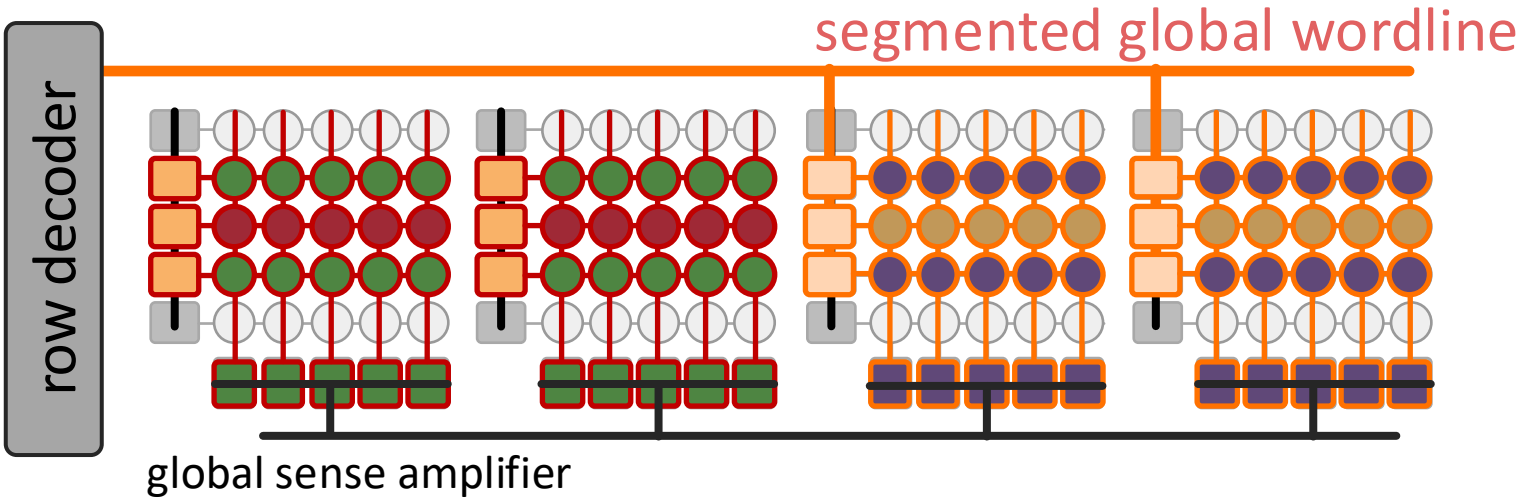


Use fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

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

MIMDRAM: Key Idea (III)

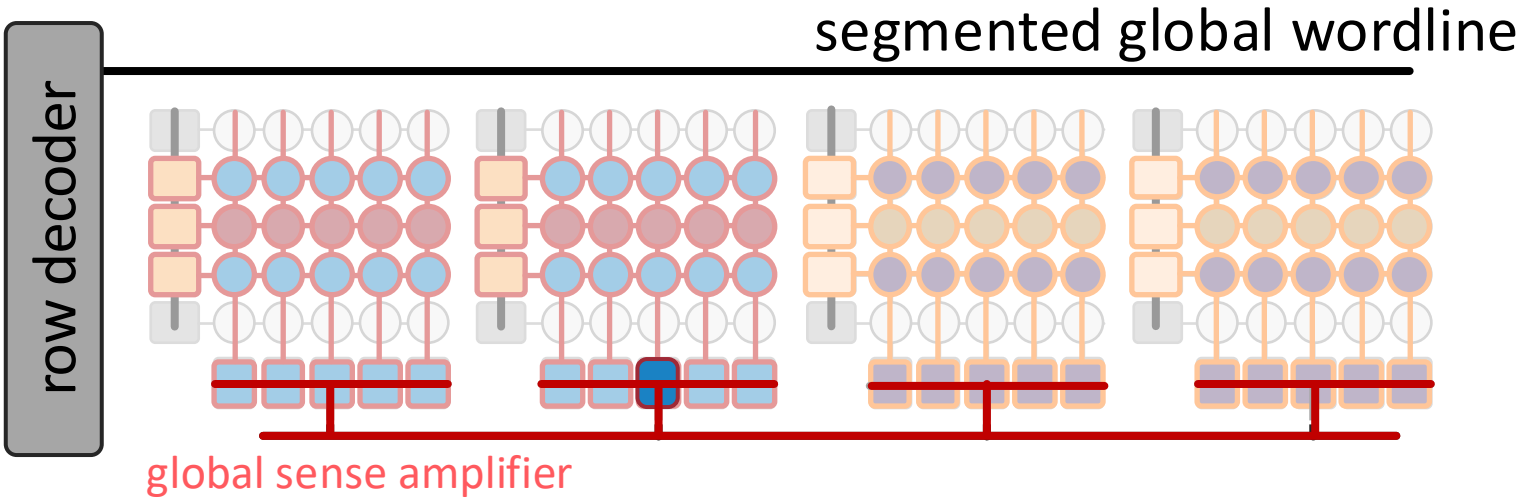


Use fine-grained DRAM for processing-using-DRAM:

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)



Use fine-grained DRAM for processing-using-DRAM:

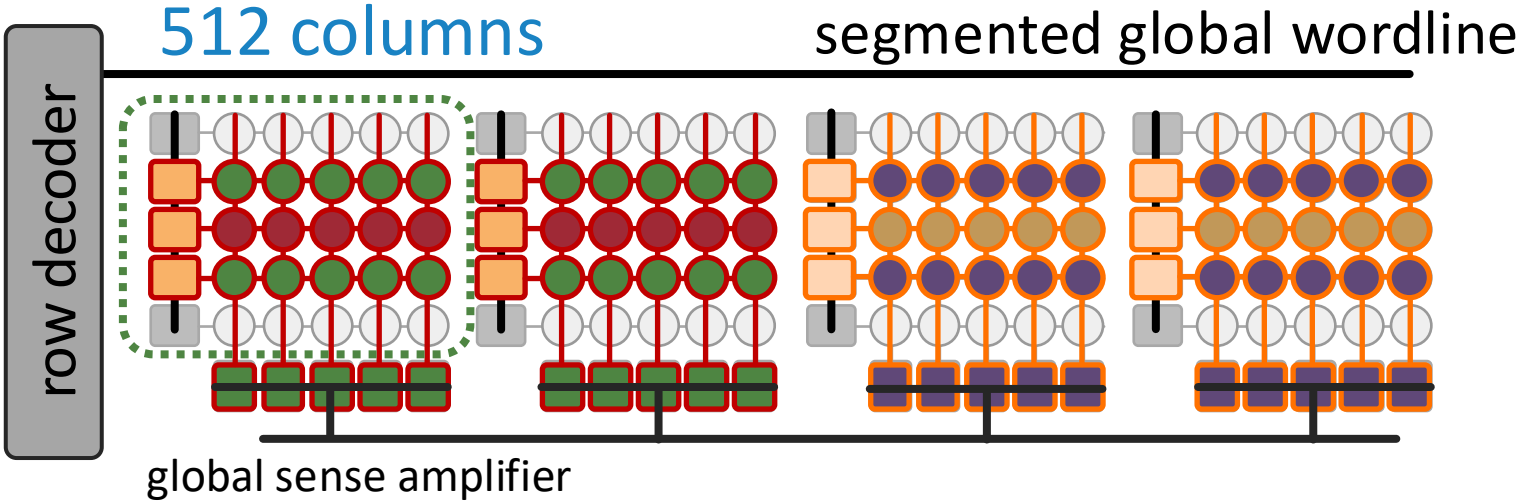
1 Improves SIMD utilization

- 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)



Use 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

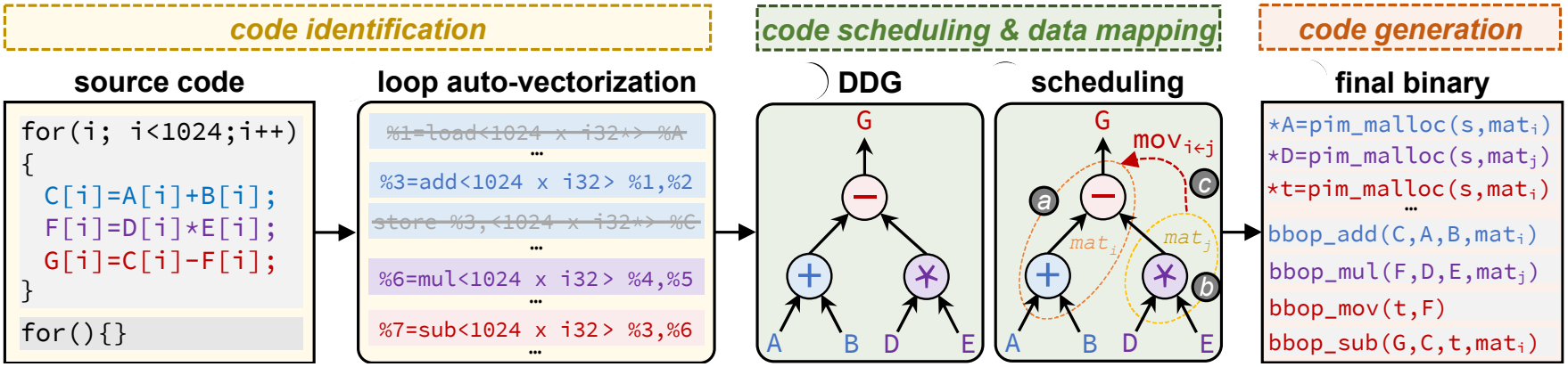
- SIMD parallelism in a DRAM mat is on par with vector ISAs' SIMD width

MIMDRAM: Compiler Support

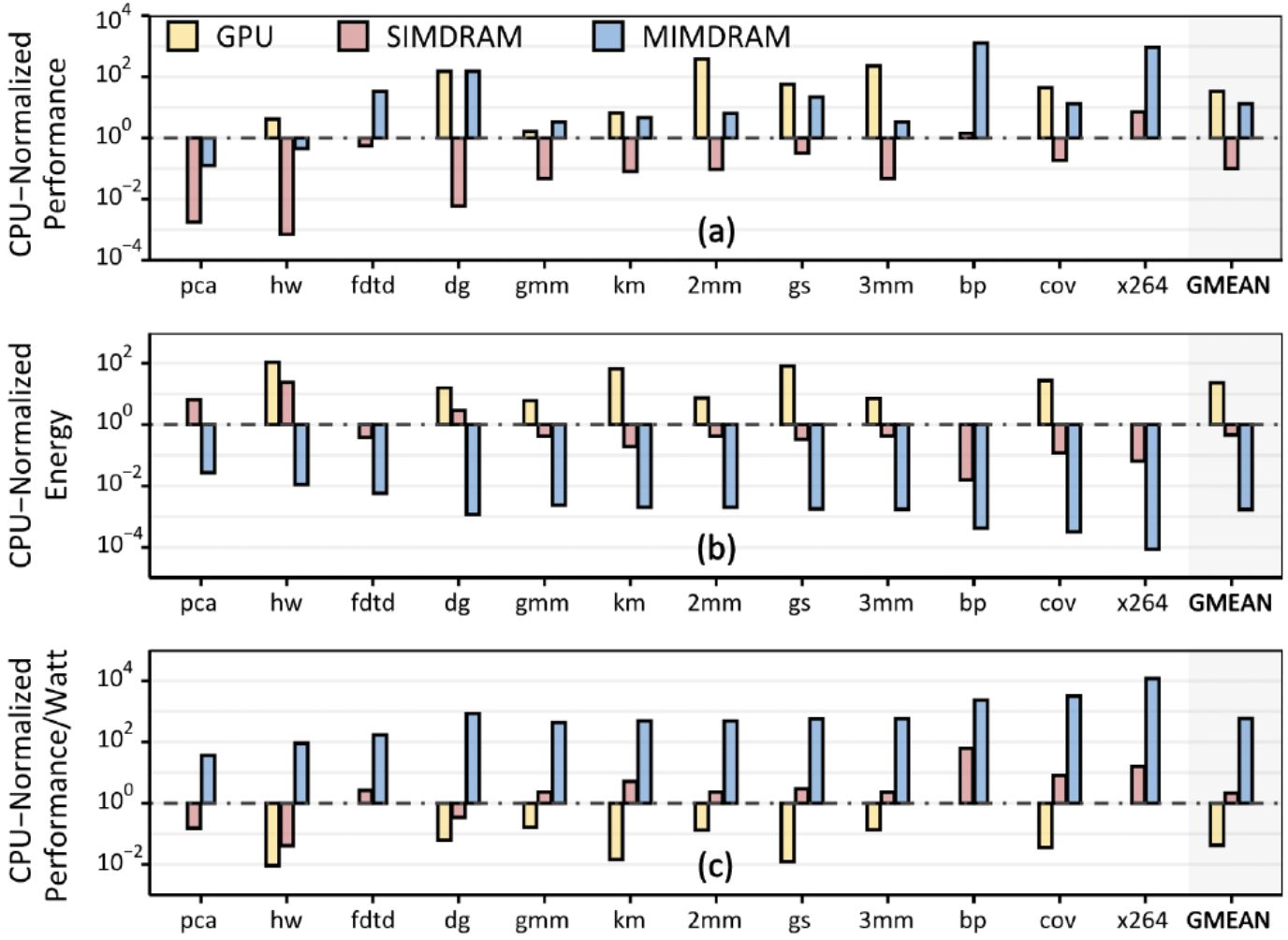
Goal

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

Three new LLVM-based passes targeting PUD execution



MIMDRAM Perf, Energy, Perf/Watt



582X and 13,612X the energy efficiency of CPU and GPU, respectively

Capabilities of Off-The-Shelf Memory

Existing DRAM Chips
Are Already Quite Capable

RowClone and MAJ3 in Real DRAM Chips

- **Key Idea:** Violate DRAM timing parameters to mimic RowClone and Ambit's MAJ3 operation

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao
feig@princeton.edu
Department of Electrical Engineering
Princeton University

Georgios Tziantzioulis
georgios.tziantzioulis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlaff
wentzlaf@princeton.edu
Department of Electrical Engineering
Princeton University

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

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

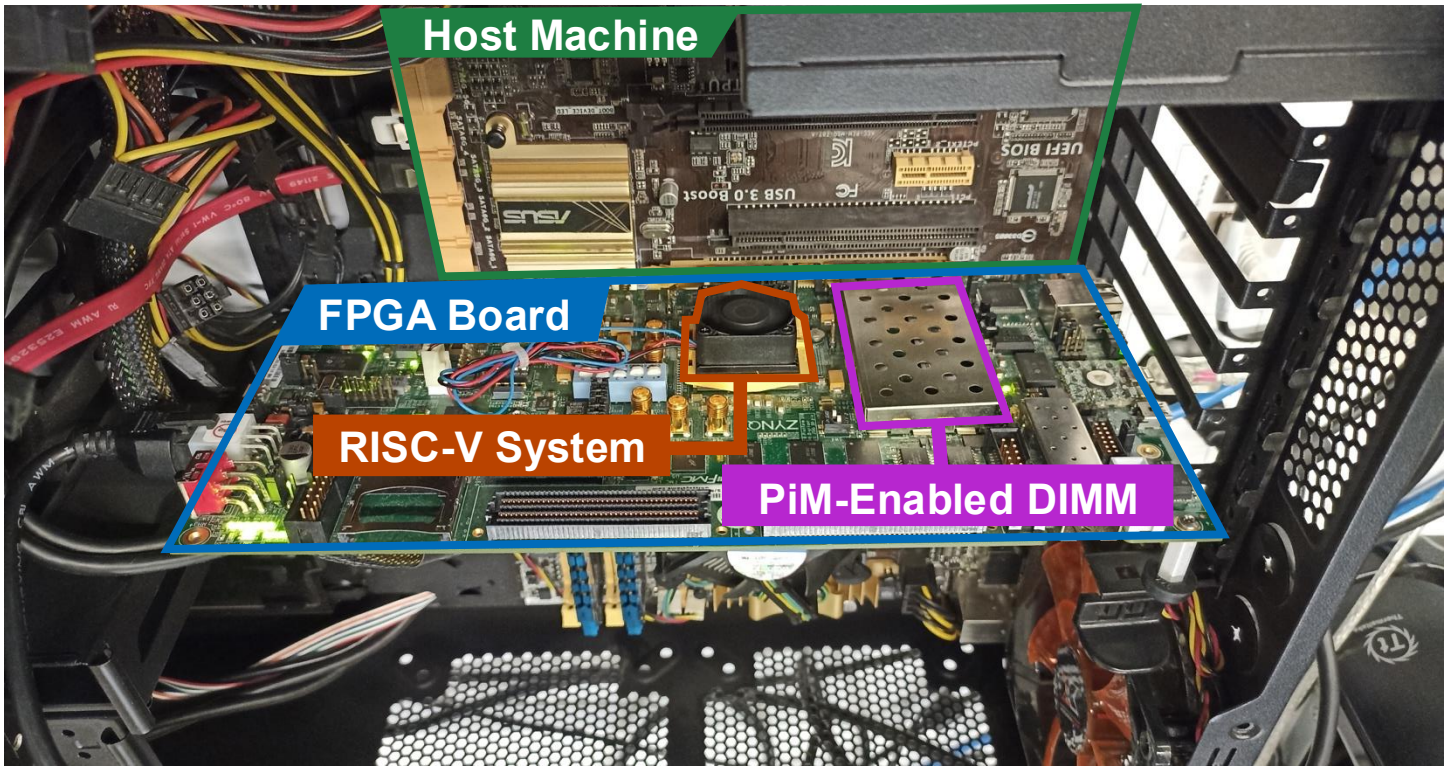
[§]ETH Zürich [†]TOBB ETÜ [*]BSC

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing Using Memory Prototype



https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5Xl3g&t=4192s

Real Processing Using Memory Prototype

README.md

Building a PiDRAM Prototype

To build PiDRAM's prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR's fpga-zynq repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM's memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPGAConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPGAConfig -j<number of cores>`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (system_top.bit) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx PHY IP we use in PiDRAM's memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:

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

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5Xl3g&t=4192s

More on PiDRAM

- Ataberk Olgun, Juan Gomez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"
  ACM Transactions on Architecture and Code Optimization (***TACO***), March 2023.
  [arXiv version]
  Presented at the 18th HiPEAC Conference, Toulouse, France, January 2023.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (40 minutes)]
  [PiDRAM Source Code]

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

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

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

Focus on Recent Real DRAM Chip-based PuD Works

HPCA 2024

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

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

DSN 2024

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

ISCA 2025

PuDHammer: Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips

İsmail Emir Yüksel Akash Sood Ataberk Olgun Oğuzhan Canpolat Haocong Luo
F. Nisa Bostancı Mohammad Sadrosadati A. Giray Yağlıkçı Onur Mutlu

ETH Zürich

Functionally-Complete Boolean Logic in Real DRAM Chips

Experimental Characterization and Analysis

Ismail Emir Yüksel

Yahya C. Tugrul Ataberk Olgun F. Nisa Bostancı

A. Giray Yaglıkçı Geraldo F. Oliveira Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI

ETH zürich

Executive Summary

Motivation: Processing-using-DRAM can alleviate the performance and energy bottlenecks caused by **data movement**

- **Prior works** show that **existing DRAM chips** can perform **three-input majority** and **two-input AND and OR** operations

Problem: Proof-of-concept demonstrations on **commercial off-the-shelf (COTS)** DRAM chips do not provide

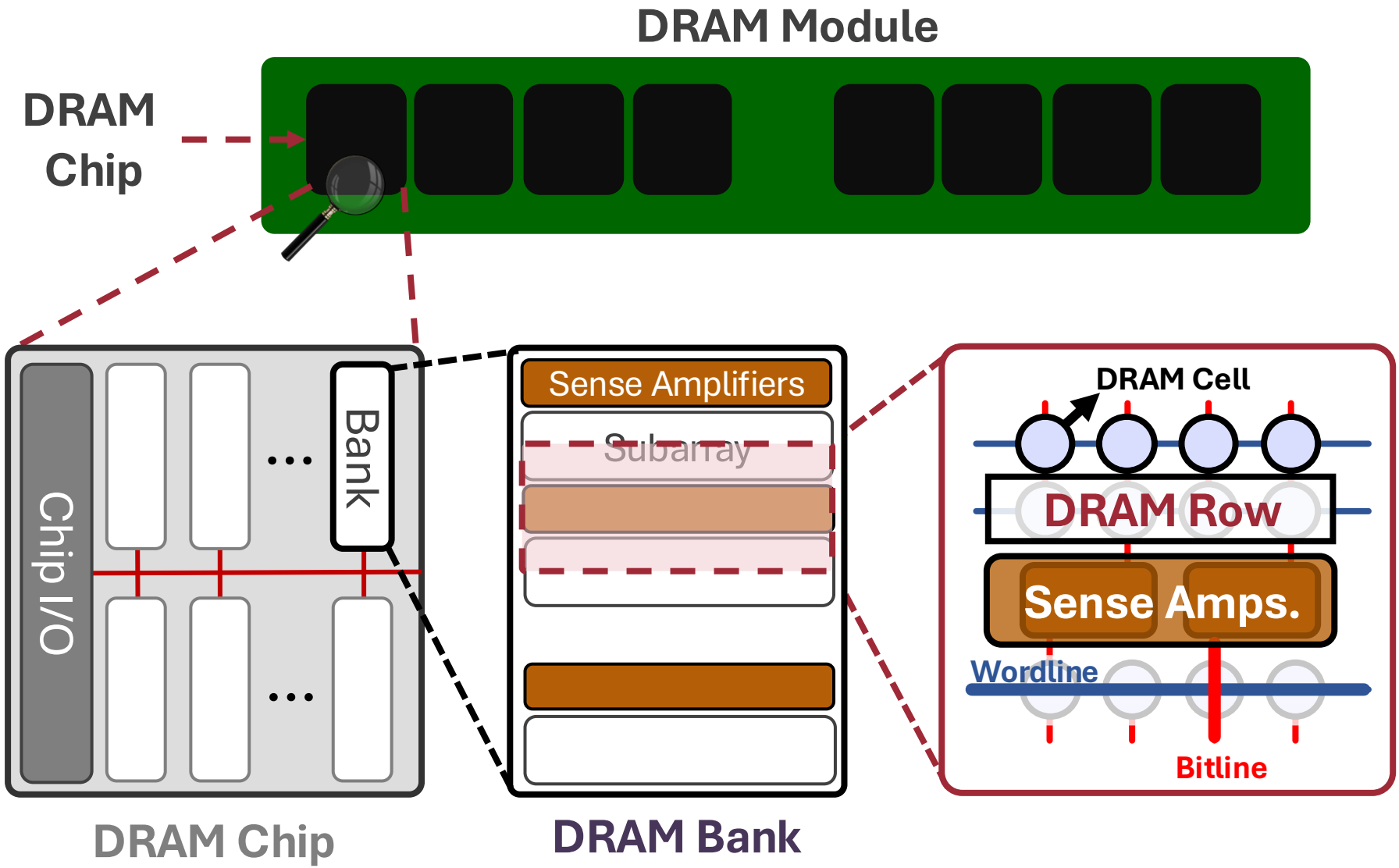
- **functionally-complete** operations (e.g., NAND or NOR)
- **NOT operation**
- **AND** and **OR** operations with **more than two inputs**

Experimental Study: 256 DDR4 chips from **two major manufacturers**

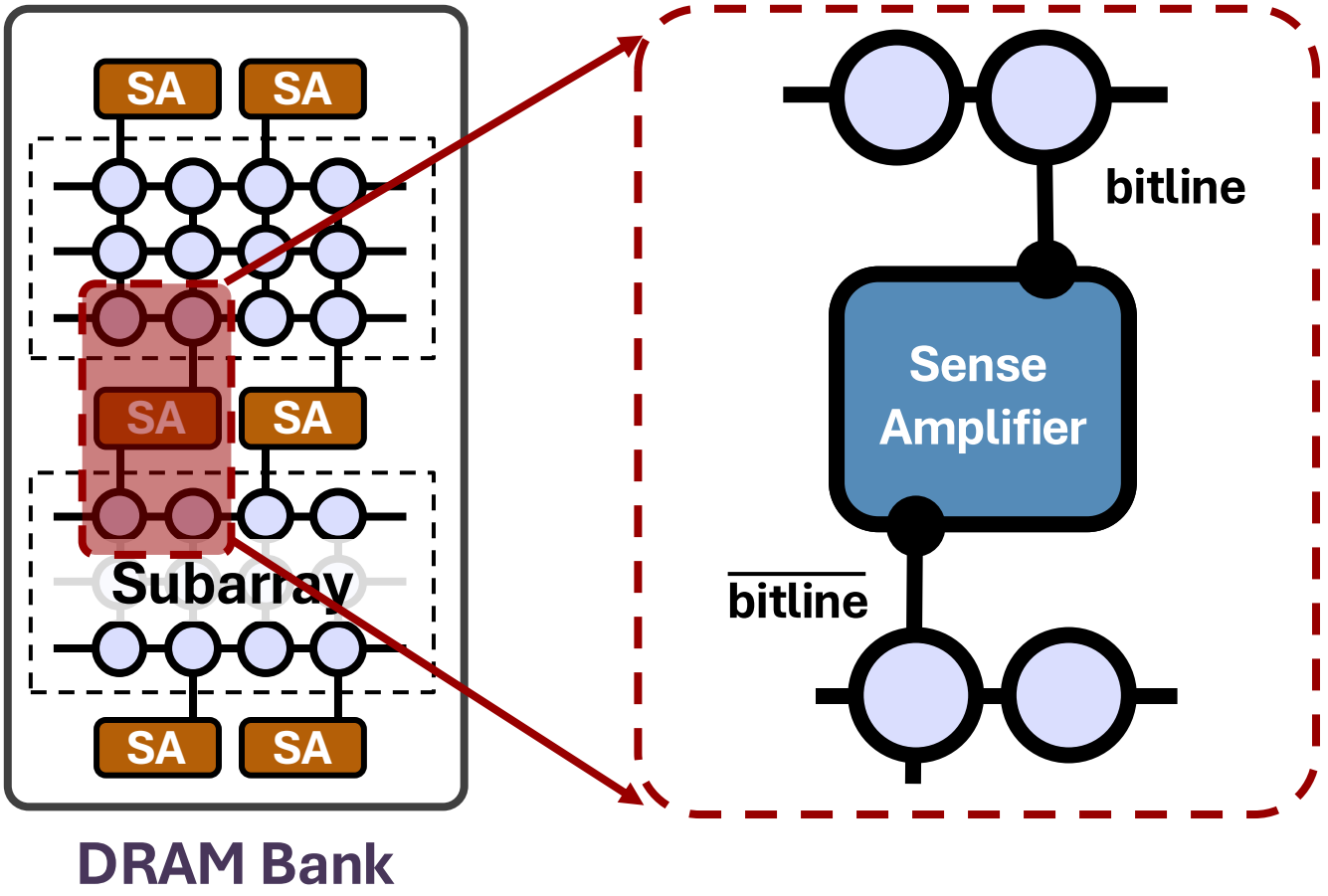
Key Results:

- COTS DRAM chips can perform **NOT** and **{2, 4, 8, 16}-input AND, NAND, OR, and NOR** operations with **very high reliability (>94% success rate)**
- **Data pattern** and **temperature** only slightly affect the reliability of these operations **(<1.98% decrease in success rate)**

DRAM Organization

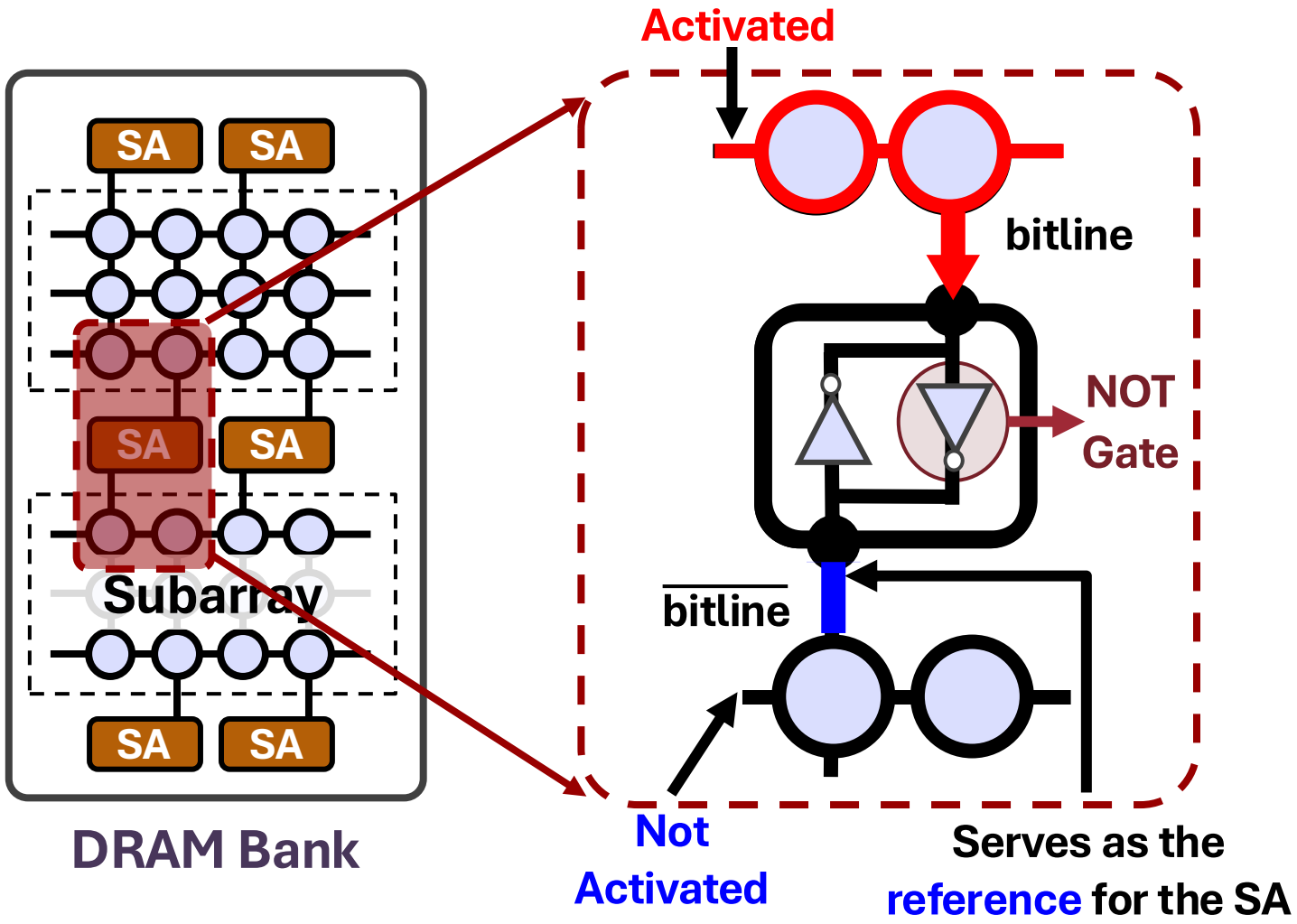


DRAM Open Bitline Architecture

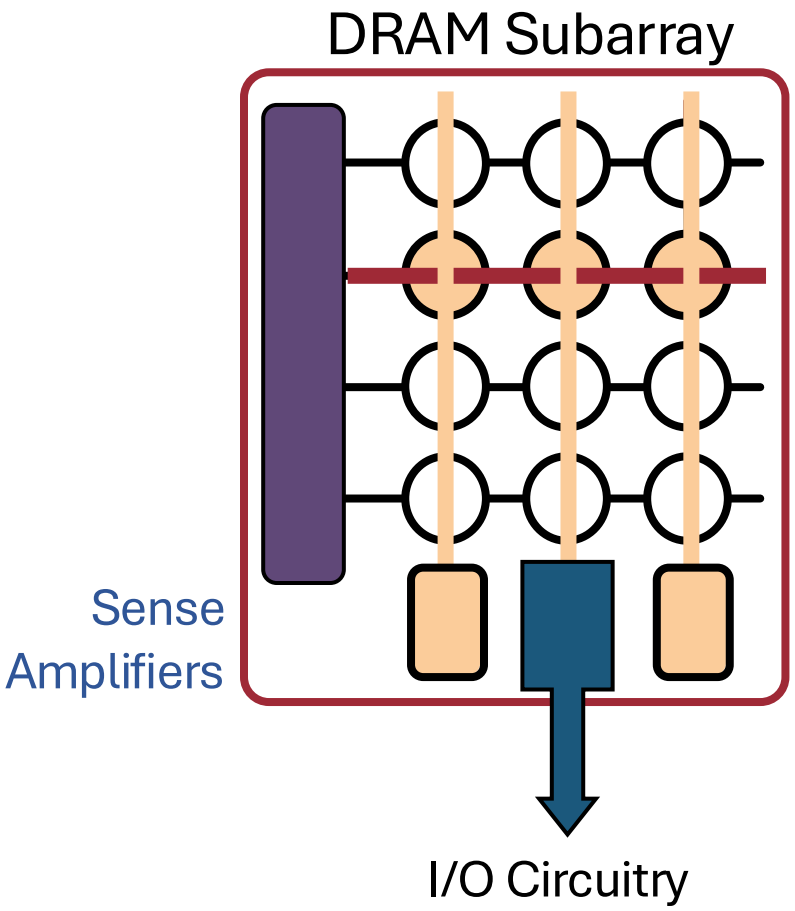


DRAM Bank

DRAM Open Bitline Architecture



DRAM Operation



1 **ACTIVATE (ACT):**
Fetch the row's content into the **sense amplifiers**

2 **Column Access (RD/WR):**
Read/Write the target column and drive to I/O

3 **PRECHARGE (PRE):**
Prepare the bank for a new ACTIVATE

The Capability of COTS DRAM Chips

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

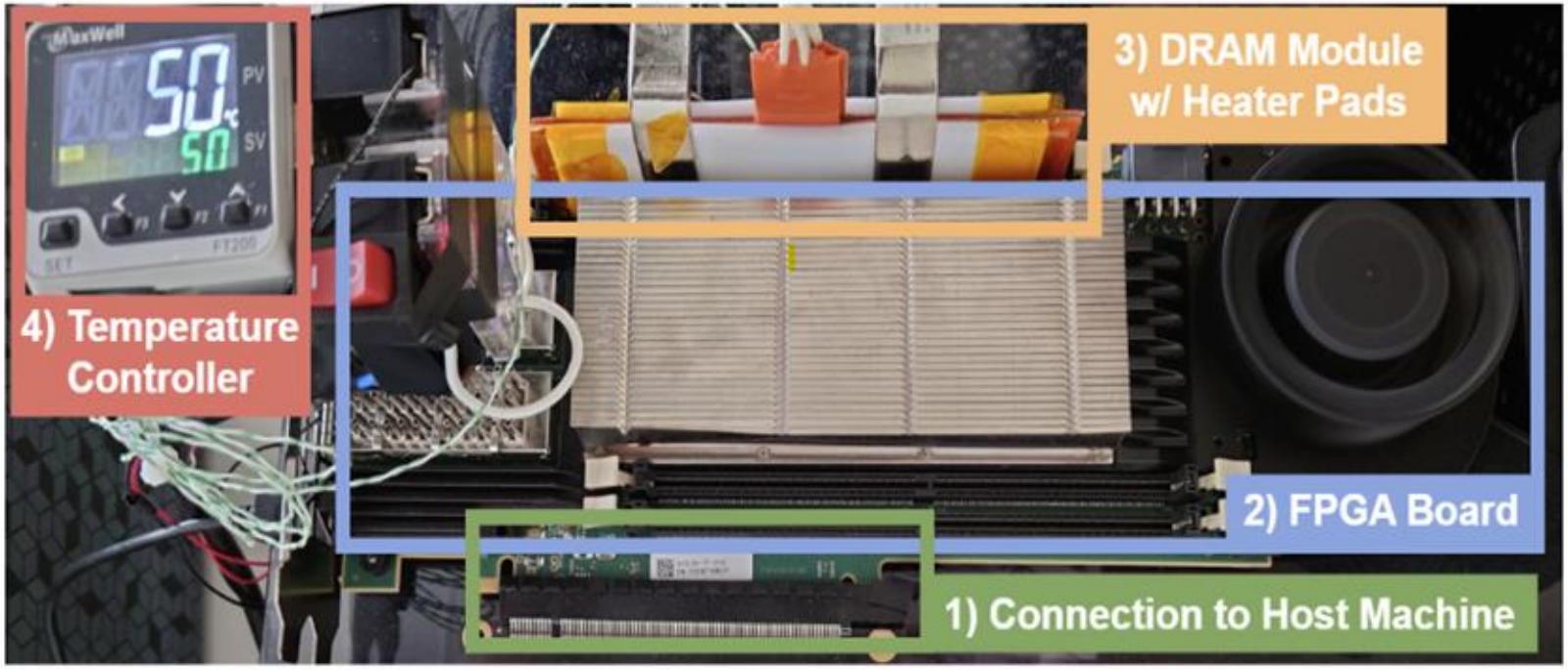
1 Can **simultaneously activate** up to **48 rows** in **two neighboring subarrays**

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

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

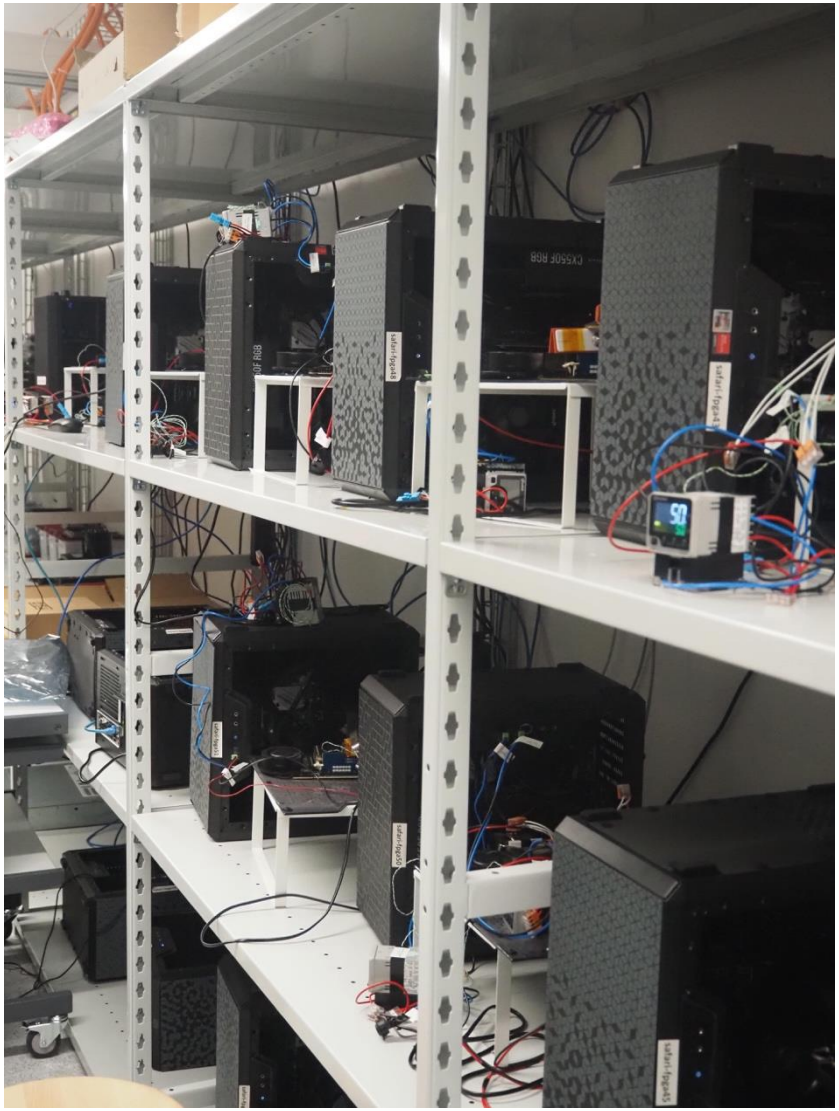
DRAM Testing Infrastructure

- Developed from DRAM Bender [Olgun+, TCAD'23]*
- Fine-grained control over DRAM commands, timings, and temperature

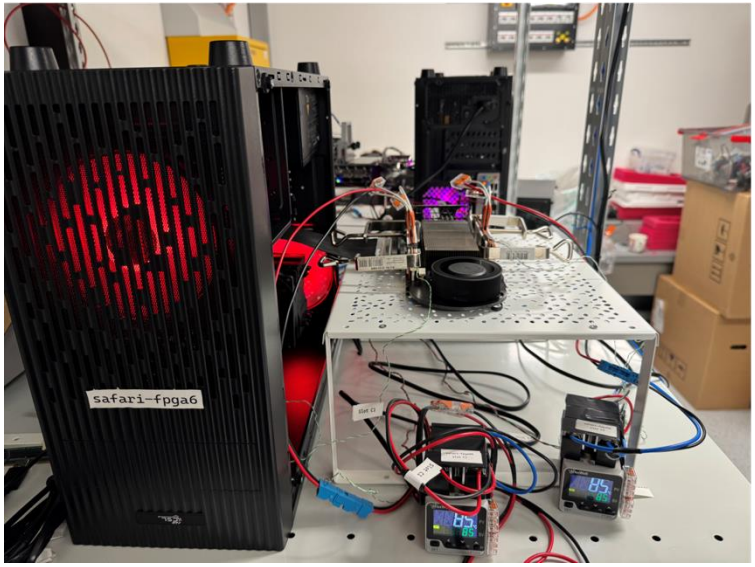


*Olgun et al., "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," TCAD, 2023.

Laboratory for Understanding Memory



Laboratory for Understanding Memory



DRAM Chips Tested

- 256 DDR4 chips from two major DRAM manufacturers
- Covers different die revisions and chip densities

Chip Mfr.	#Modules (#Chips)	Die Rev.	Mfr. Date[a]	Chip Density	Chip Org.	Speed Rate
SK Hynix	9 (72)	M	N/A	4Gb	x8	2666MT/s
	5 (40)	A	N/A	4Gb	x8	2133MT/s
	1 (16)	A	N/A	8Gb	x8	2666MT/s
	1 (32)	A	18-14	4Gb	x4	2400MT/s
	1 (32)	A	16-49	8Gb	x4	2400MT/s
	1 (32)	M	16-22	8Gb	x4	2666MT/s
Samsung	1 (8)	F	21-02	4Gb	x8	2666MT/s
	2 (16)	D	21-10	8Gb	x8	2133MT/s
	1 (8)	A	22-12	8Gb	x8	3200MT/s

The Capability of COTS DRAM Chips

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

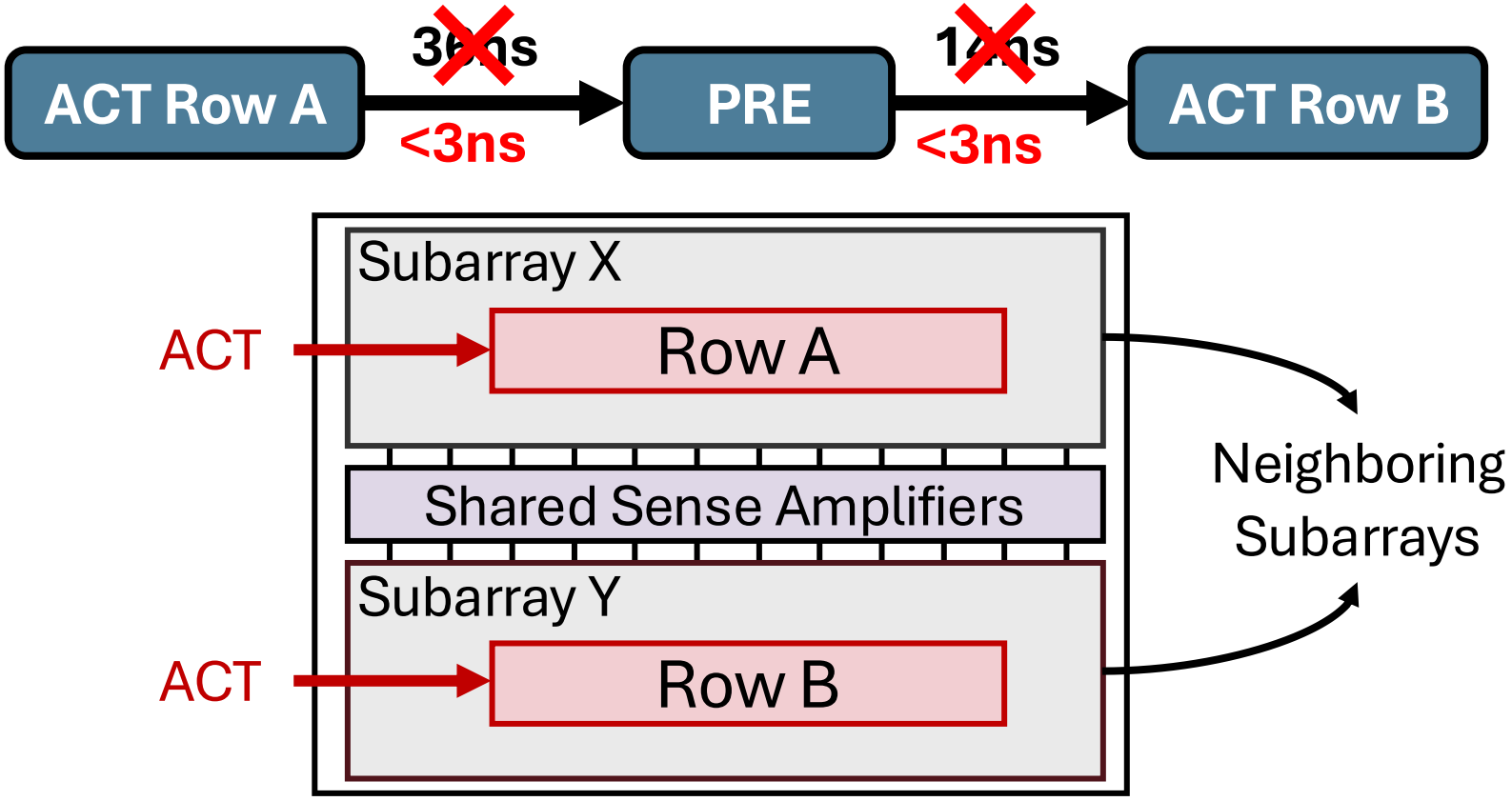
1 Can **simultaneously activate** up to **48 rows** in **two neighboring subarrays**

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

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

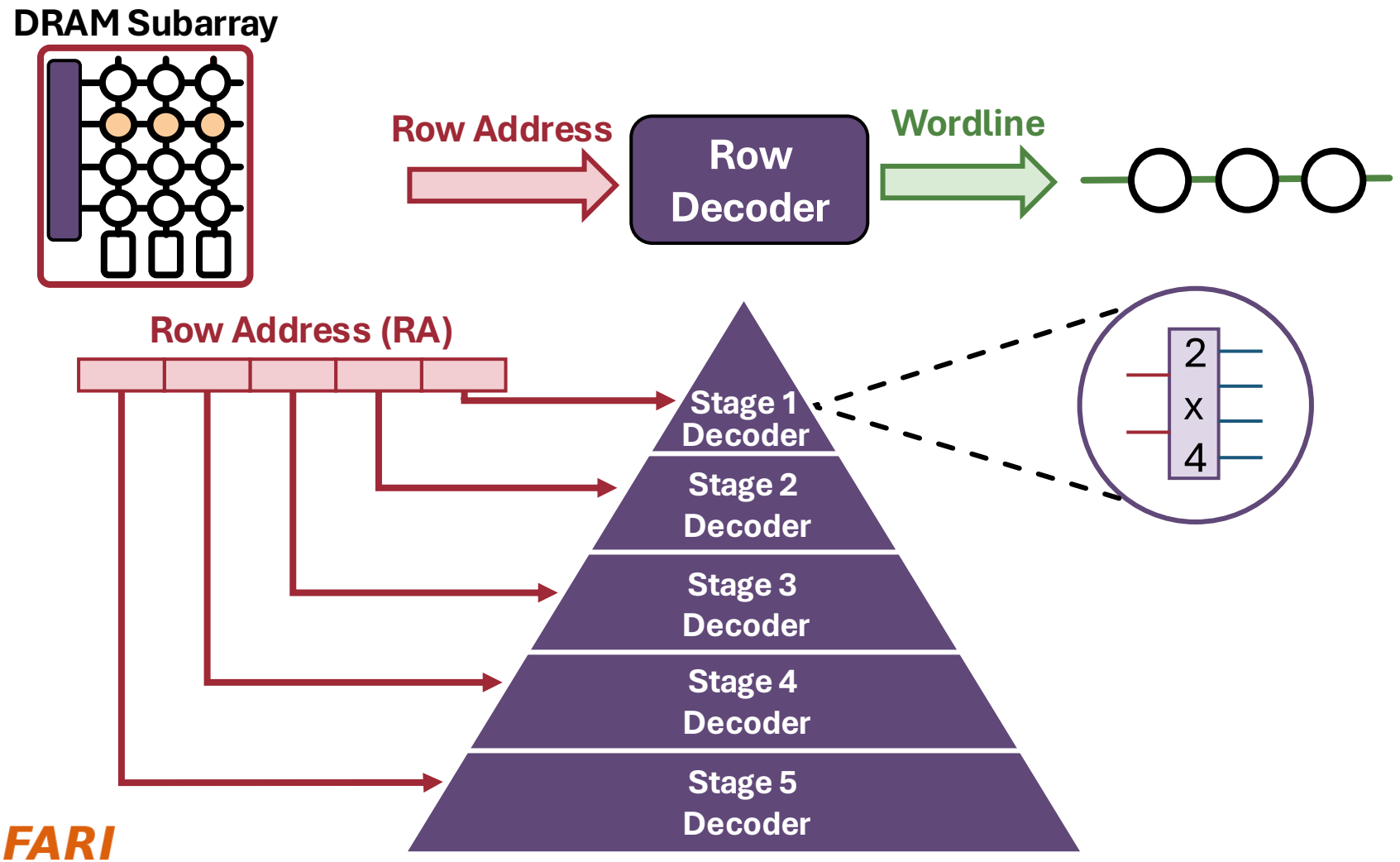
Key Observation

Activating two rows in **quick succession**
can **simultaneously** activate
multiple rows in neighboring subarrays



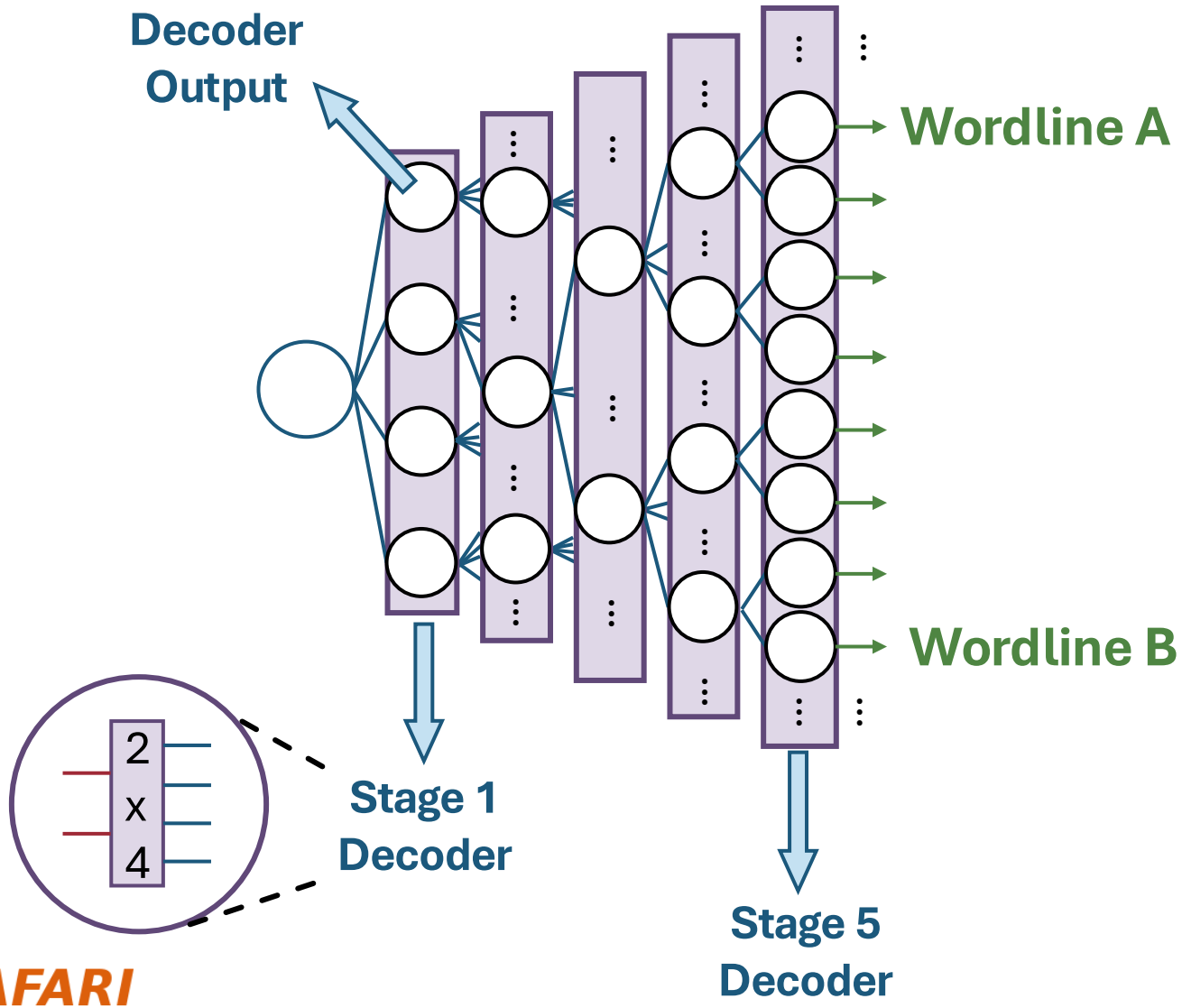
Hypothesis: Row Decoder Circuitry

Simultaneous many-row activation is possible
due to the **hierarchical DRAM row decoder design**

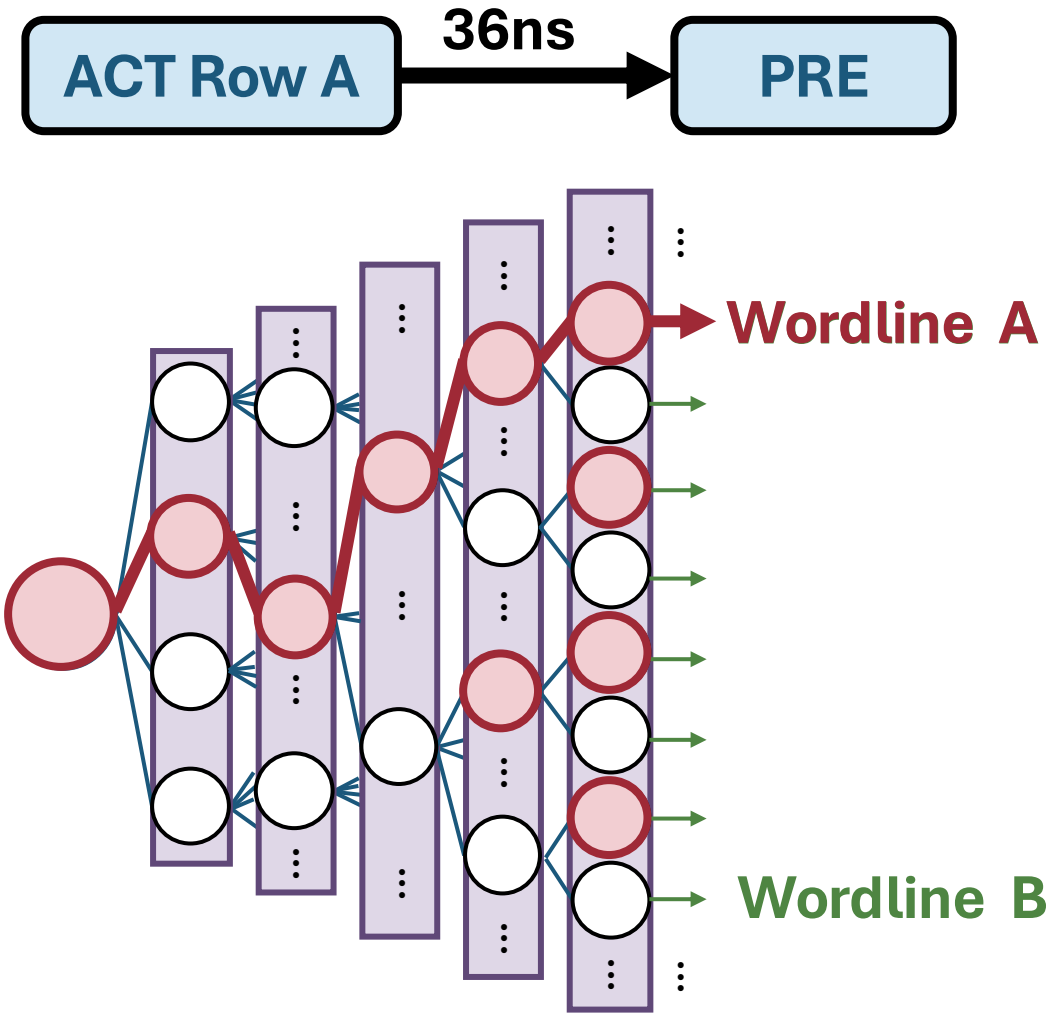


Row Decoder: A Tree Example

- We can visualize the hierarchical row decoder circuitry as a tree



Activating a Single Row



1. Decode Row A address

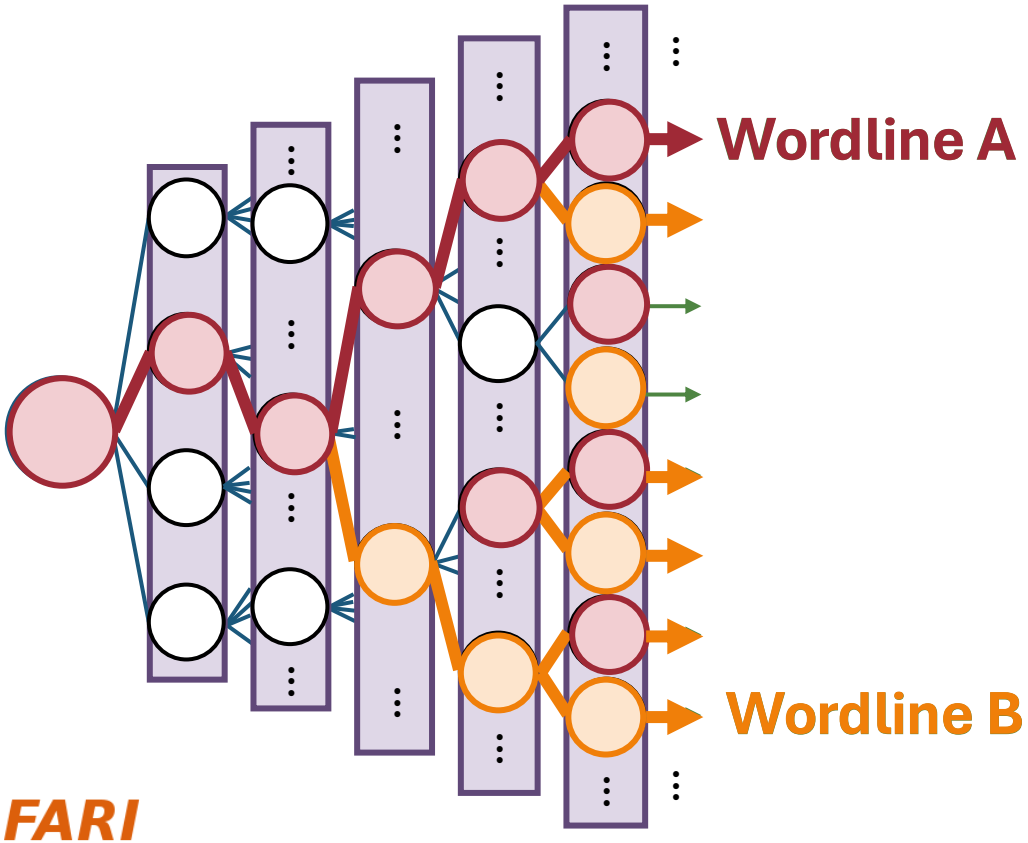
2. Assert corresponding signals

3. Follow asserted nodes path from root to leaf

4. Assert wordline

Activating Many Rows: A Walkthrough

Back-to-back ACT commands with violated timings **asserts many more signals** in the row decoder



1. Keep the previously asserted signals active

2. Decode Row B & assert corresponding signals

3. Follow newly asserted nodes' path

4. Simultaneously activate many rows

Hypothesis: Row Decoder Circuitry

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

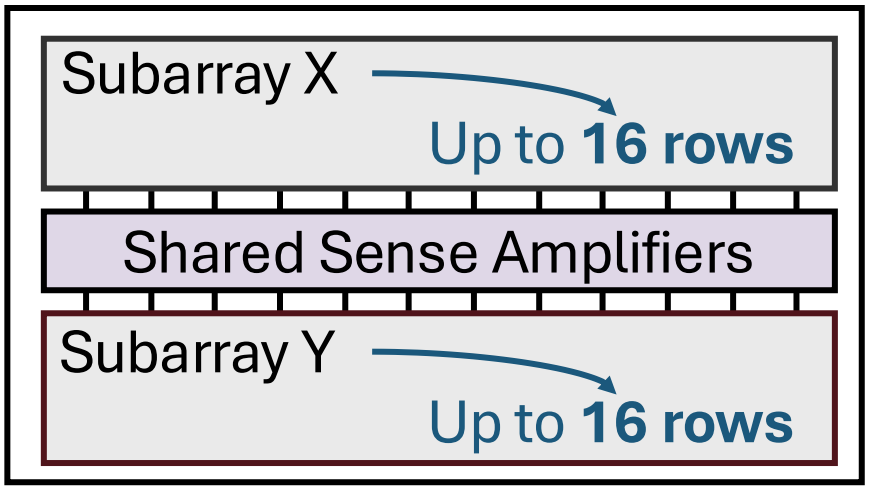
(More discussions & hypotheses in the paper)

https://arxiv.org/pdf/2405.06081

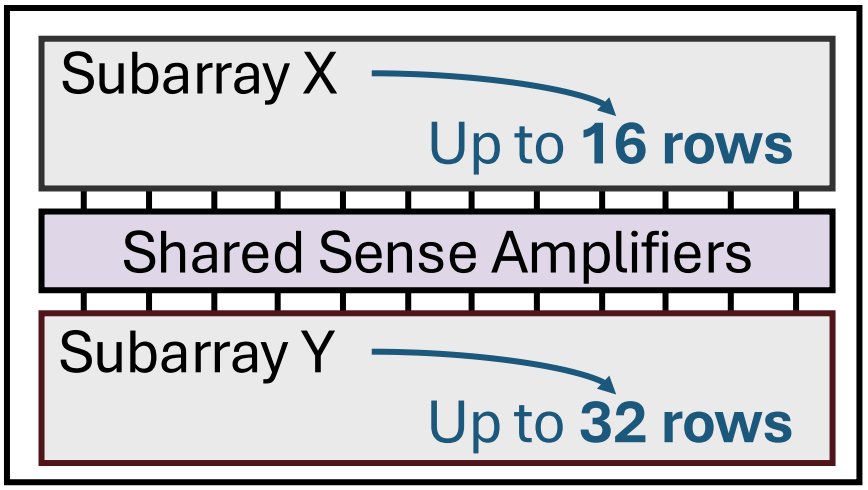
Key Results

COTS DRAM chips have **two distinct** sets of activation patterns in **neighboring subarrays** when two rows are activated with **violated timings**

Exactly the same number of rows in each subarray are activated	**Twice as many** rows in one subarray **compared to its neighbor subarray** are activated



A total of **32 rows**



A total of **48 rows**

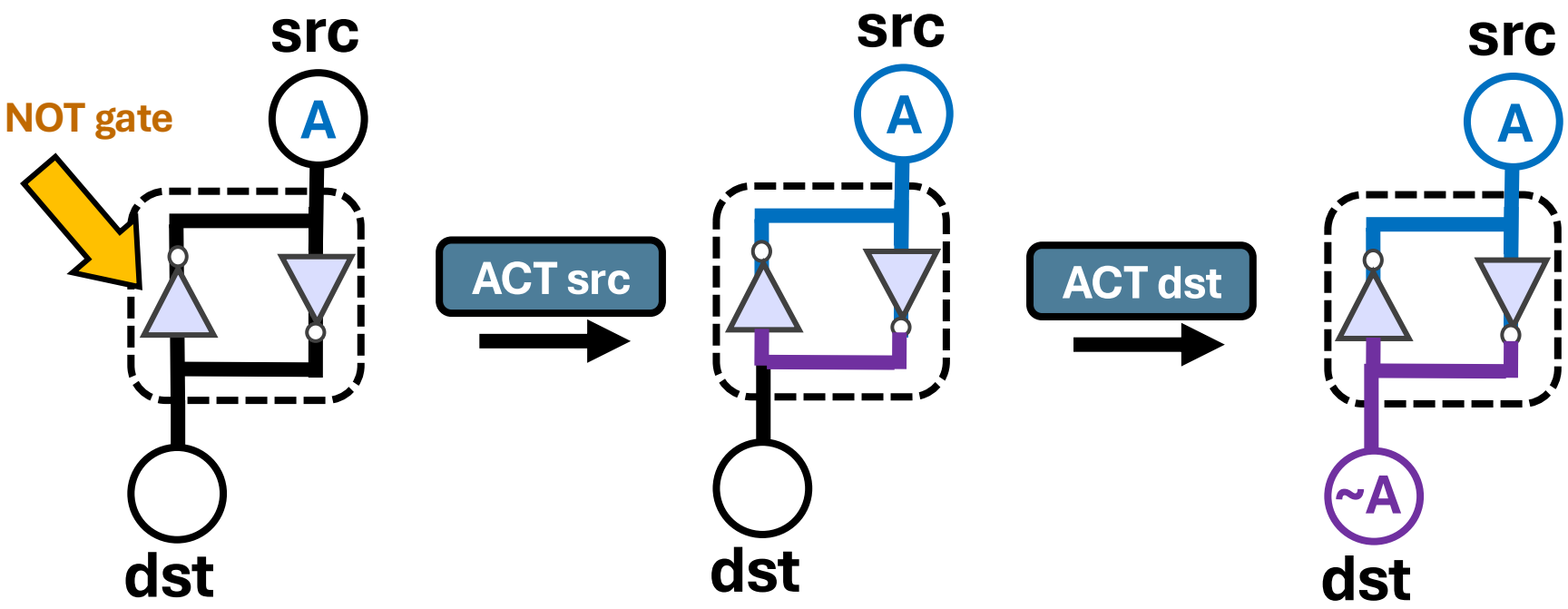
The Capability of COTS DRAM Chips

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

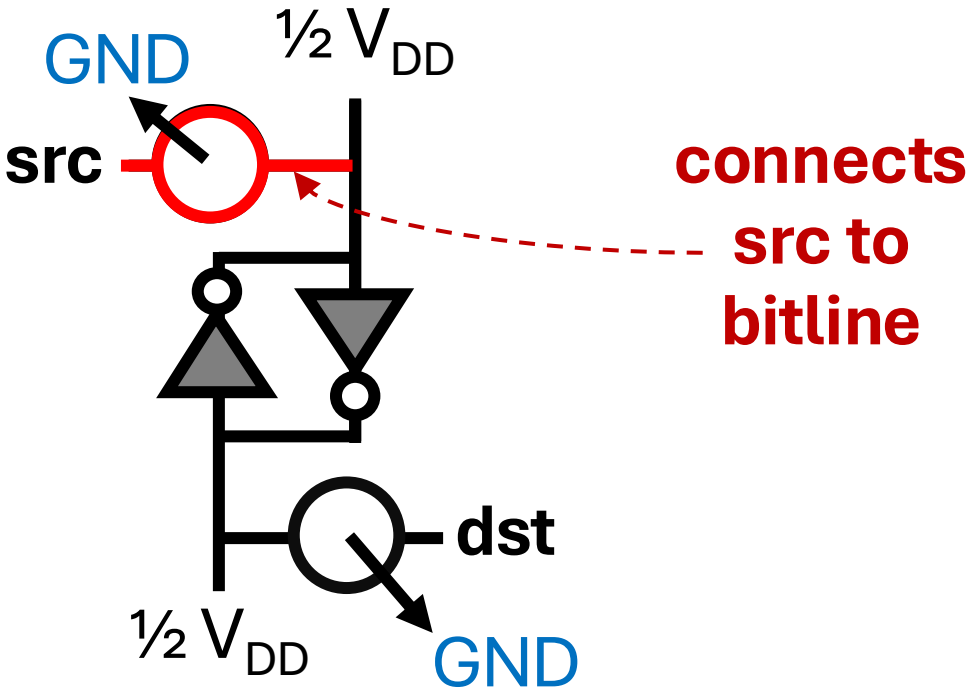
Key Idea

Connect rows in neighboring subarrays
through **a NOT gate** by simultaneously activating rows

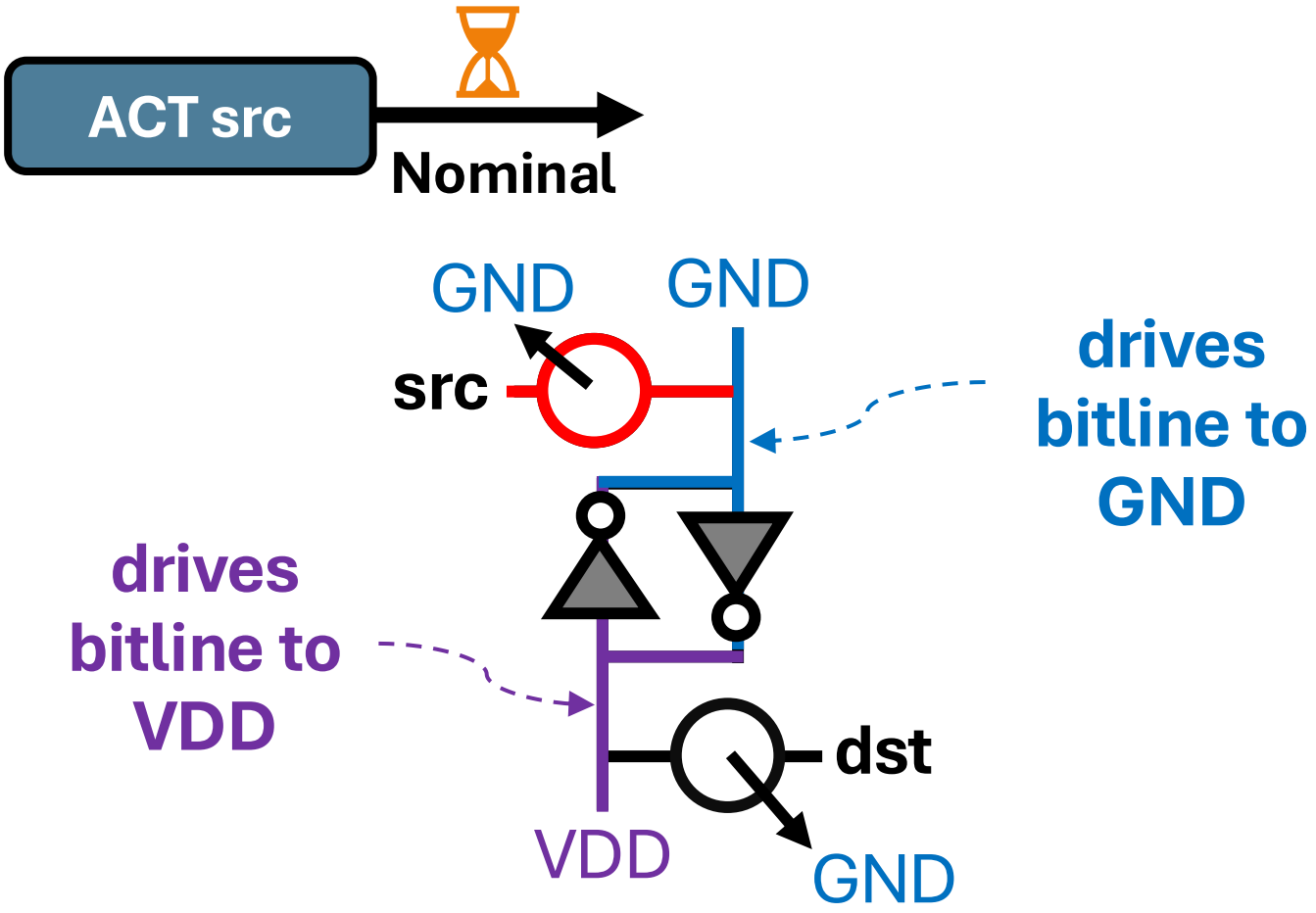


NOT Operation: A Walkthrough

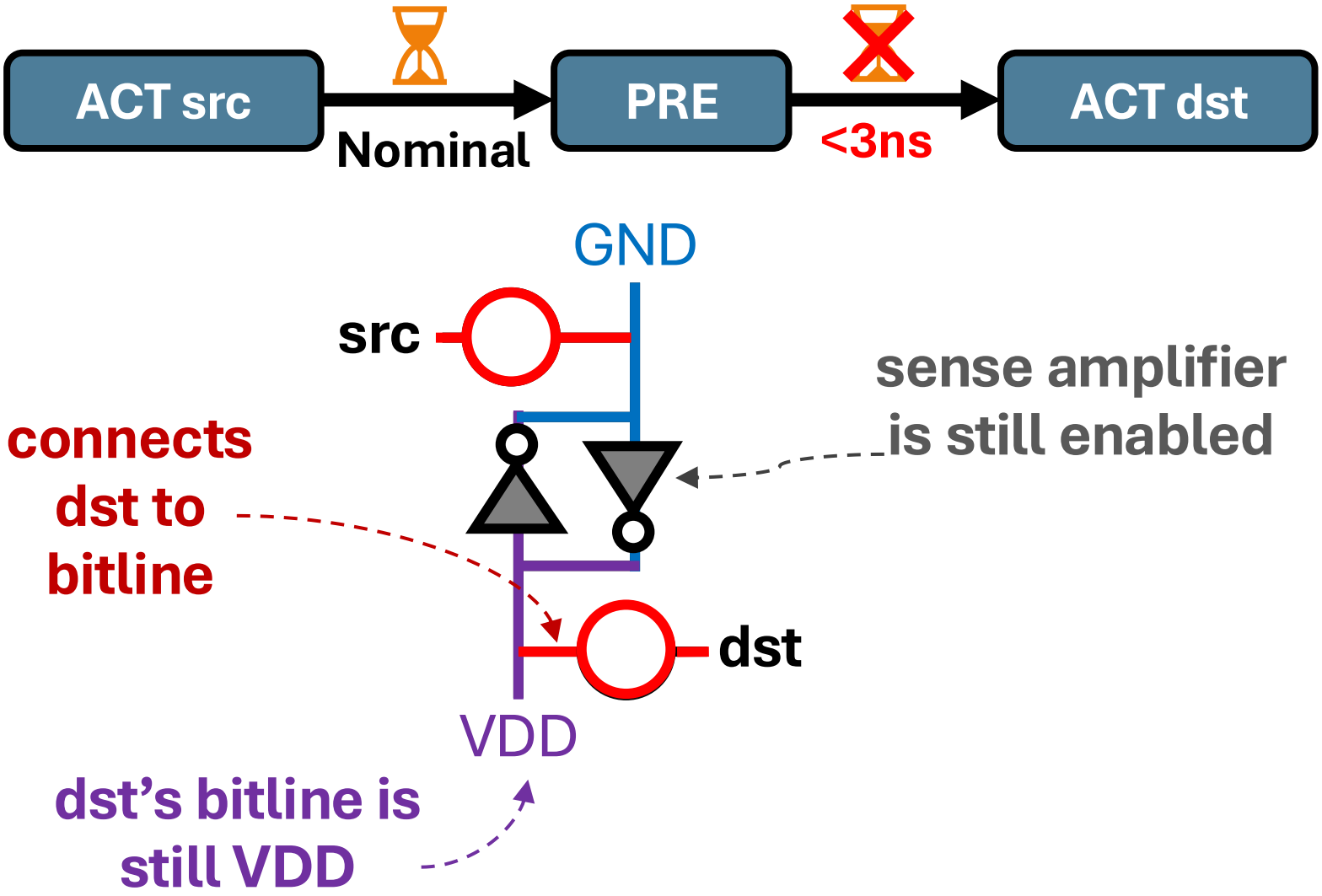
ACT src



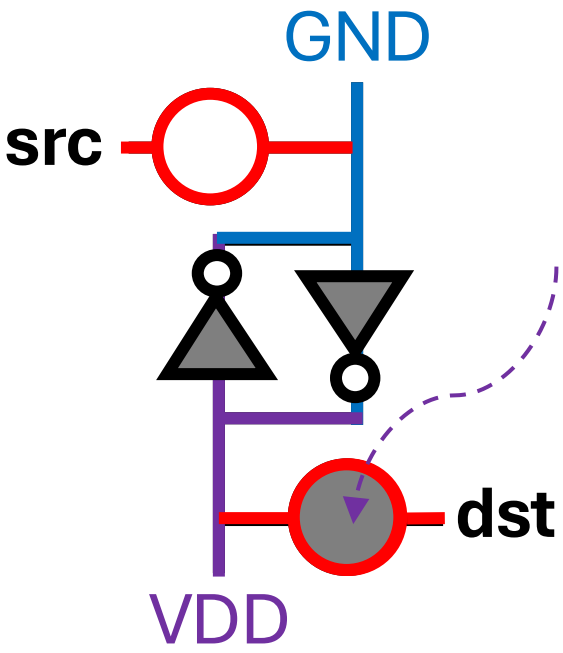
NOT Operation: A Walkthrough



NOT Operation: A Walkthrough



NOT Operation: A Walkthrough

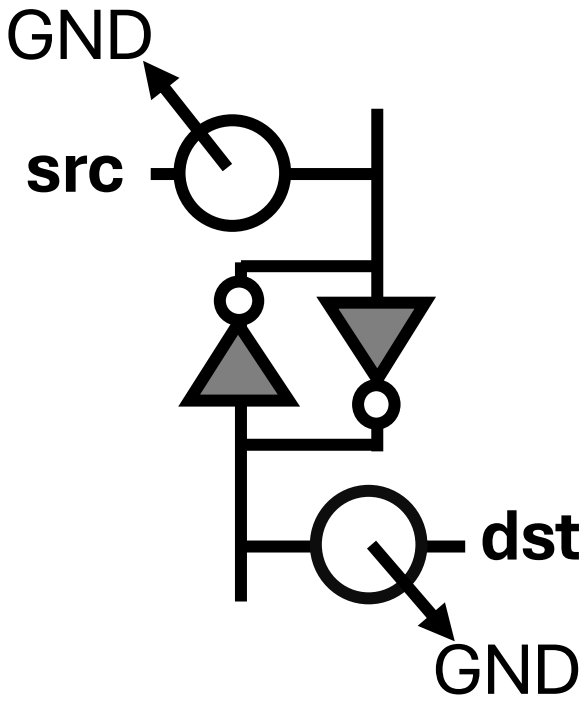


negated value of src (VDD) is written to dst

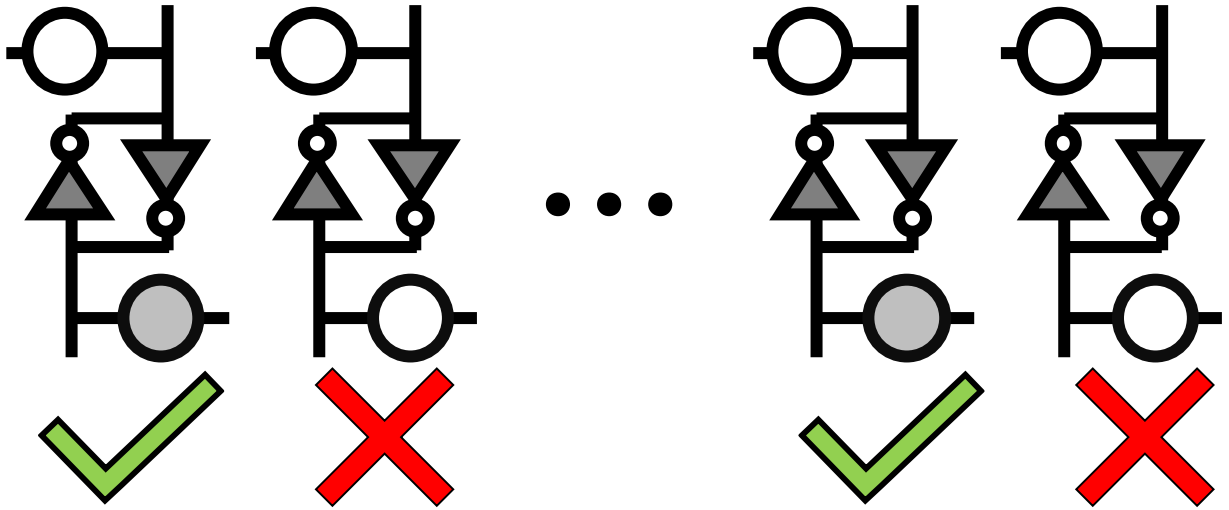
Reliability Metric

Success Rate (for a DRAM cell)

Percentage of trials where the **correct output** of a tested operation is stored in the cell



Total of 10000 trials



Success rate for this cell: 50%

Key Takeaways from In-DRAM NOT Operation

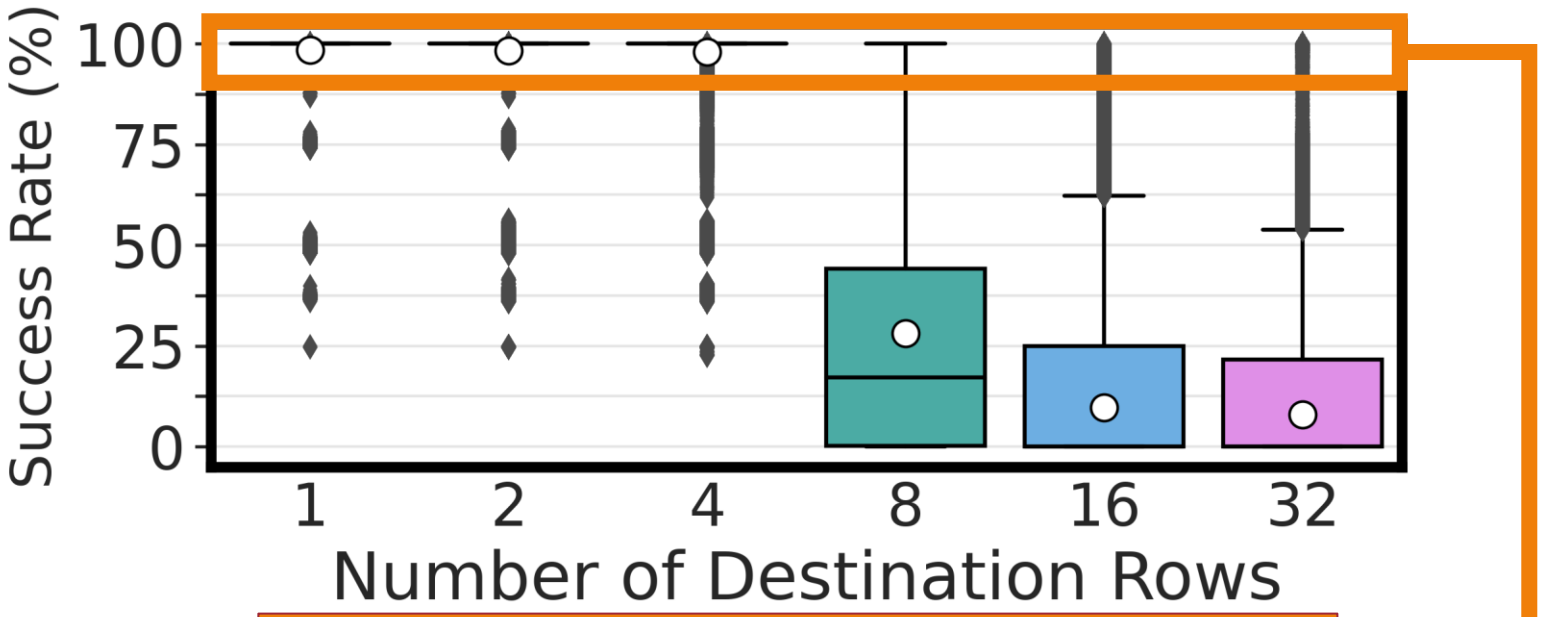
Key Takeaway 1

COTS DRAM chips can perform NOT operations with up to 32 destination rows

Key Takeaway 2

Temperature has a small effect on the reliability of NOT operations

Performing NOT in COTS DRAM Chips



There is **at least one DRAM cell** that can perform **the NOT operation with a 100% success rate**

COTS DRAM chips can perform NOT operations with up to 32 destination rows

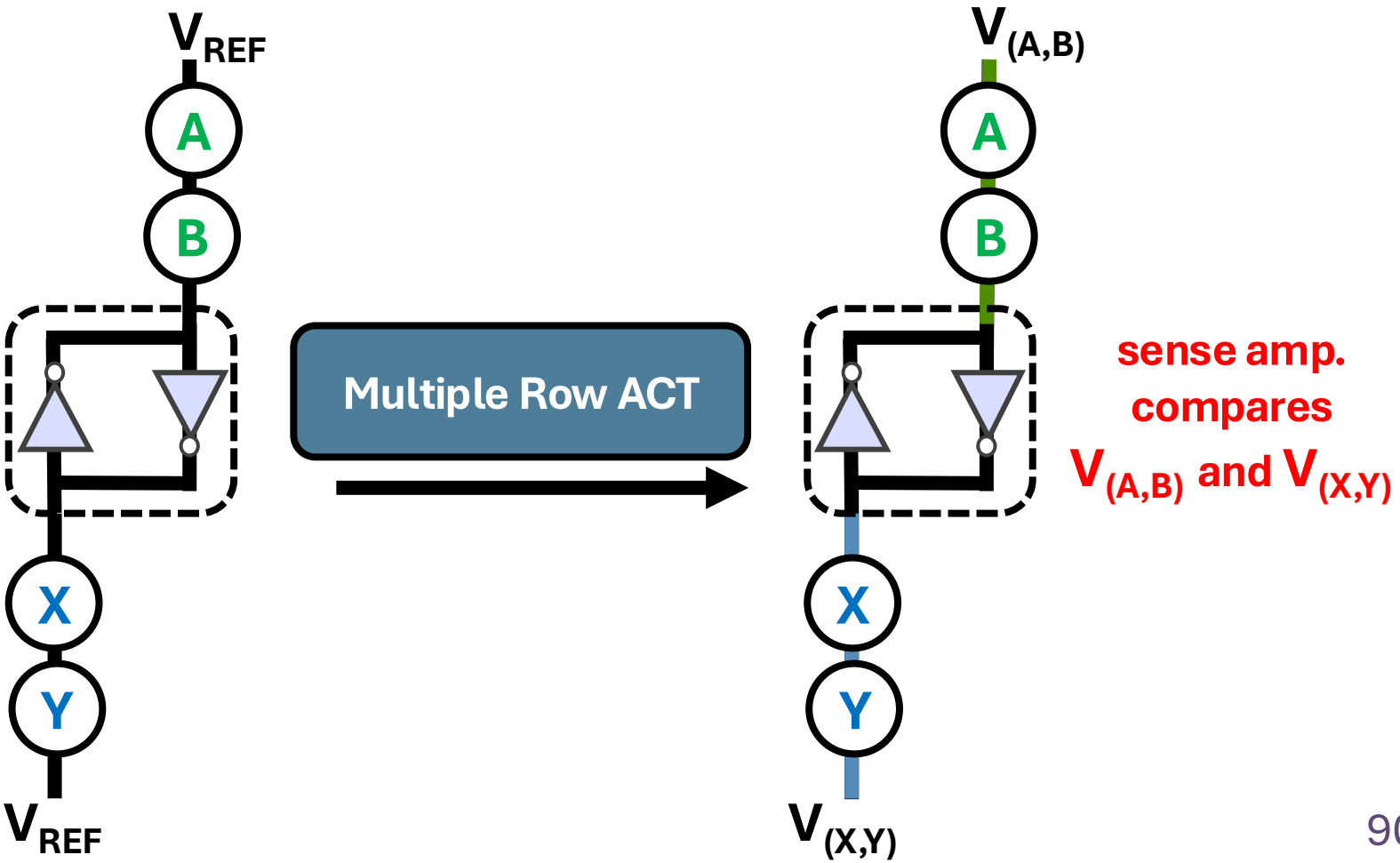
The Capability of COTS DRAM Chips

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

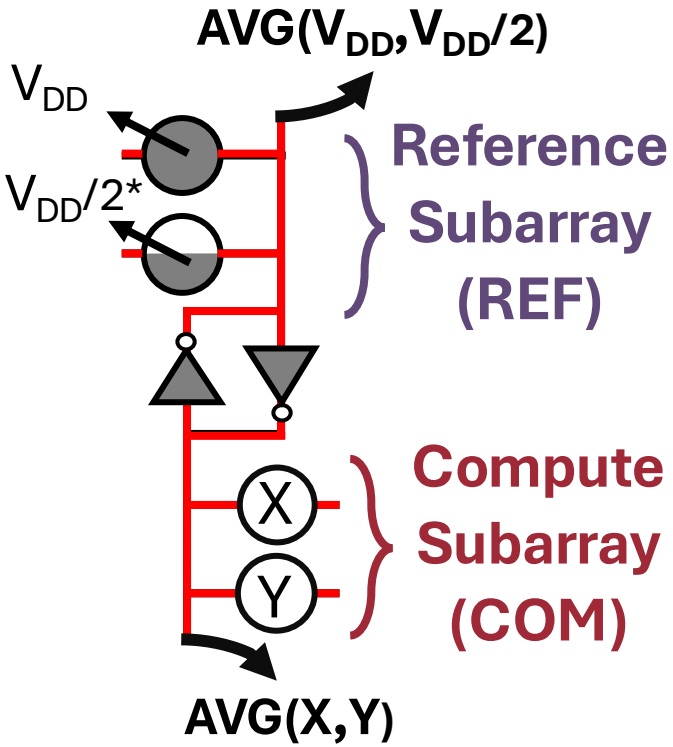
1. Can **simultaneously activate** up to **48 rows** in **two neighboring subarrays**
2. Can perform **NOT** operation with up to **32 output operands**
3. Can perform up to **16-input AND, NAND, OR, and NOR** operations

Key Idea

Manipulate the bitline voltage to express
a wide variety of functions using
multiple-row activation in neighboring subarrays

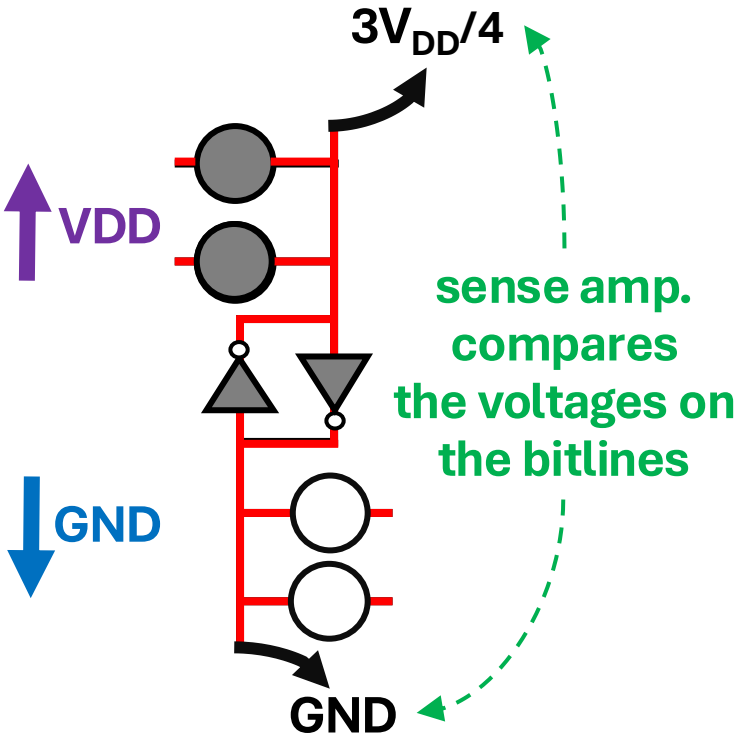


Two-Input AND and NAND Operations



*Gao et al., "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in MICRO, 2022.

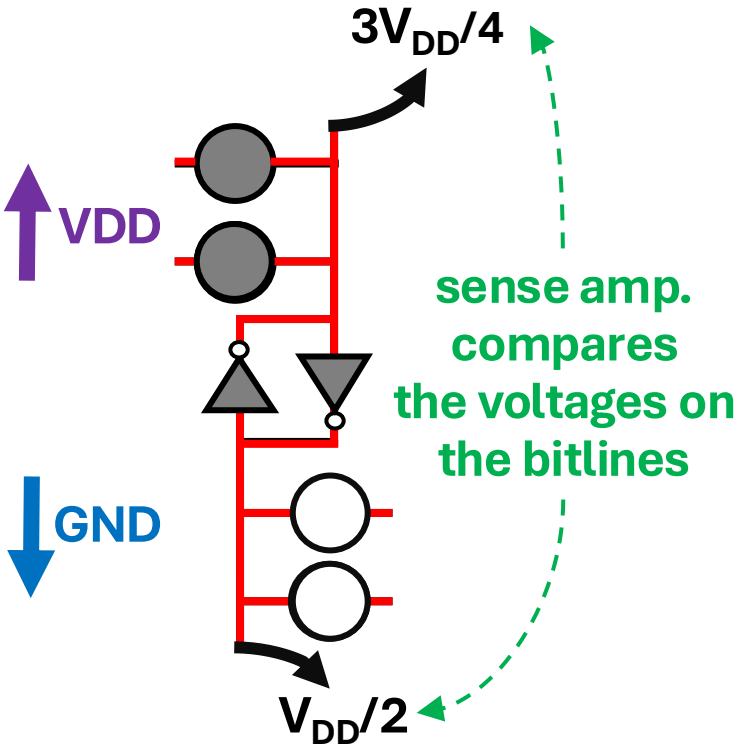
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1

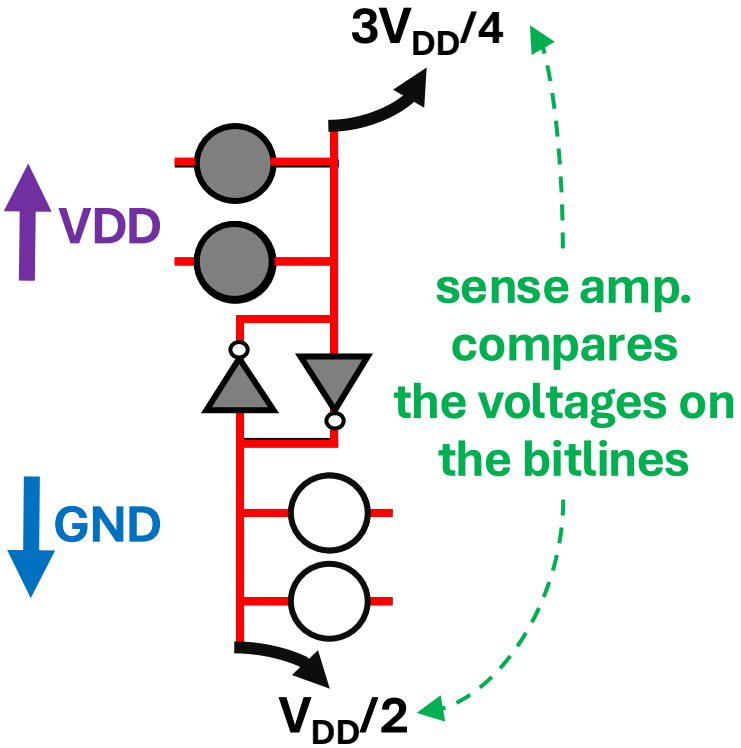
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1

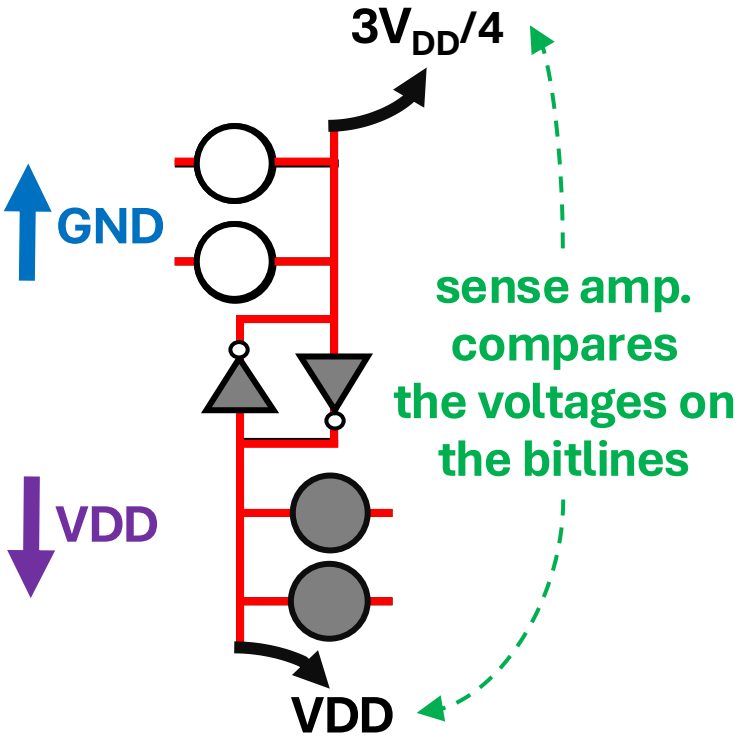
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1

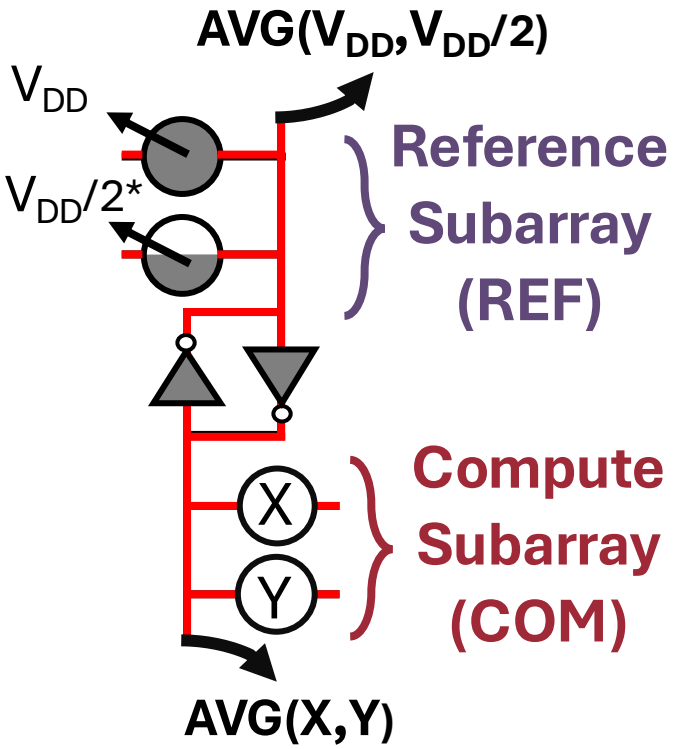
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
		AND	NAND

*Gao et al., "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in MICRO, 2022.

Key Takeaways from In-DRAM Operations

Key Takeaway 1

COTS DRAM chips can perform
{2, 4, 8, 16}-input AND, NAND, OR, and NOR operations

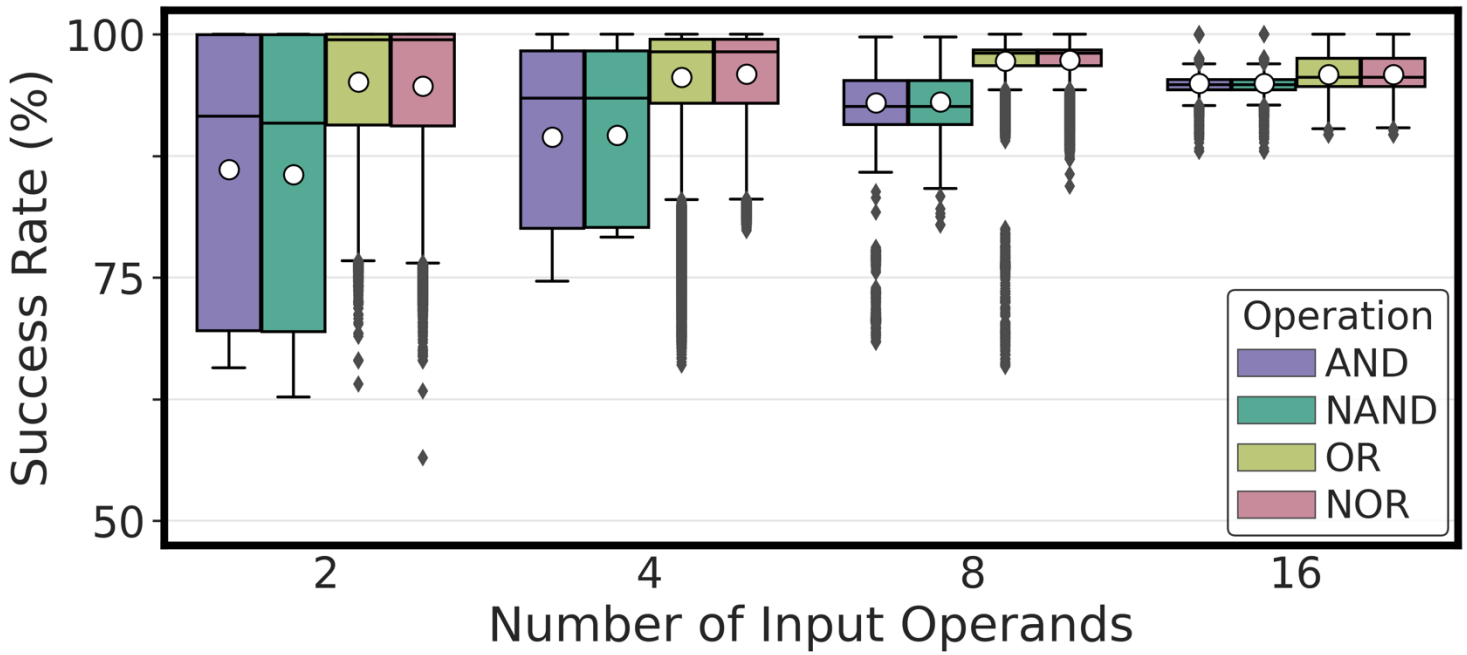
Key Takeaway 2

COTS DRAM chips can perform
AND, NAND, OR, and NOR operations
with very high reliability

Key Takeaway 3

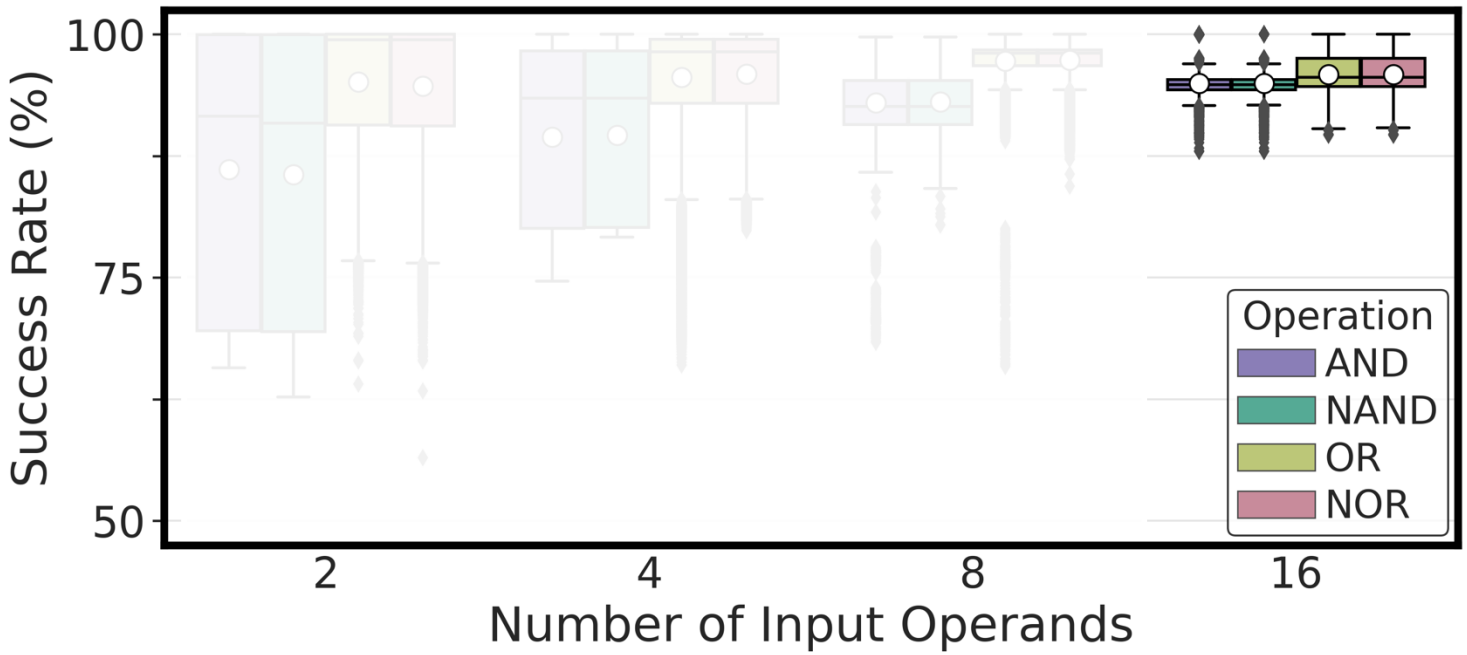
Data pattern slightly affects
the reliability of AND, NAND, OR, and NOR operations

Performing AND, NAND, OR, and NOR



COTS DRAM chips can perform {2, 4, 8, 16}-input AND, NAND, OR, and NOR operations

Performing AND, NAND, OR, and NOR



COTS DRAM chips can perform 16-input AND, NAND, OR, and NOR operations with very high success rate (>94%)

Available on arXiv

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

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

Processing-using-DRAM (PuD) is an emerging paradigm that leverages the analog operational properties of DRAM circuitry to enable massively parallel in-DRAM computation. PuD has the potential to significantly reduce or eliminate costly data movement between processing elements and main memory. A common approach for PuD architectures is to make use of bulk bitwise computation (e.g., AND, OR, NOT). Prior works experimentally demonstrate three-input MAJ (i.e., MAJ3) and two-input AND and OR operations in commercial off-the-shelf (COTS) DRAM chips. Yet, demonstrations on COTS DRAM chips do not *provide a functionally complete set of operations (e.g., NAND or AND and NOT).*

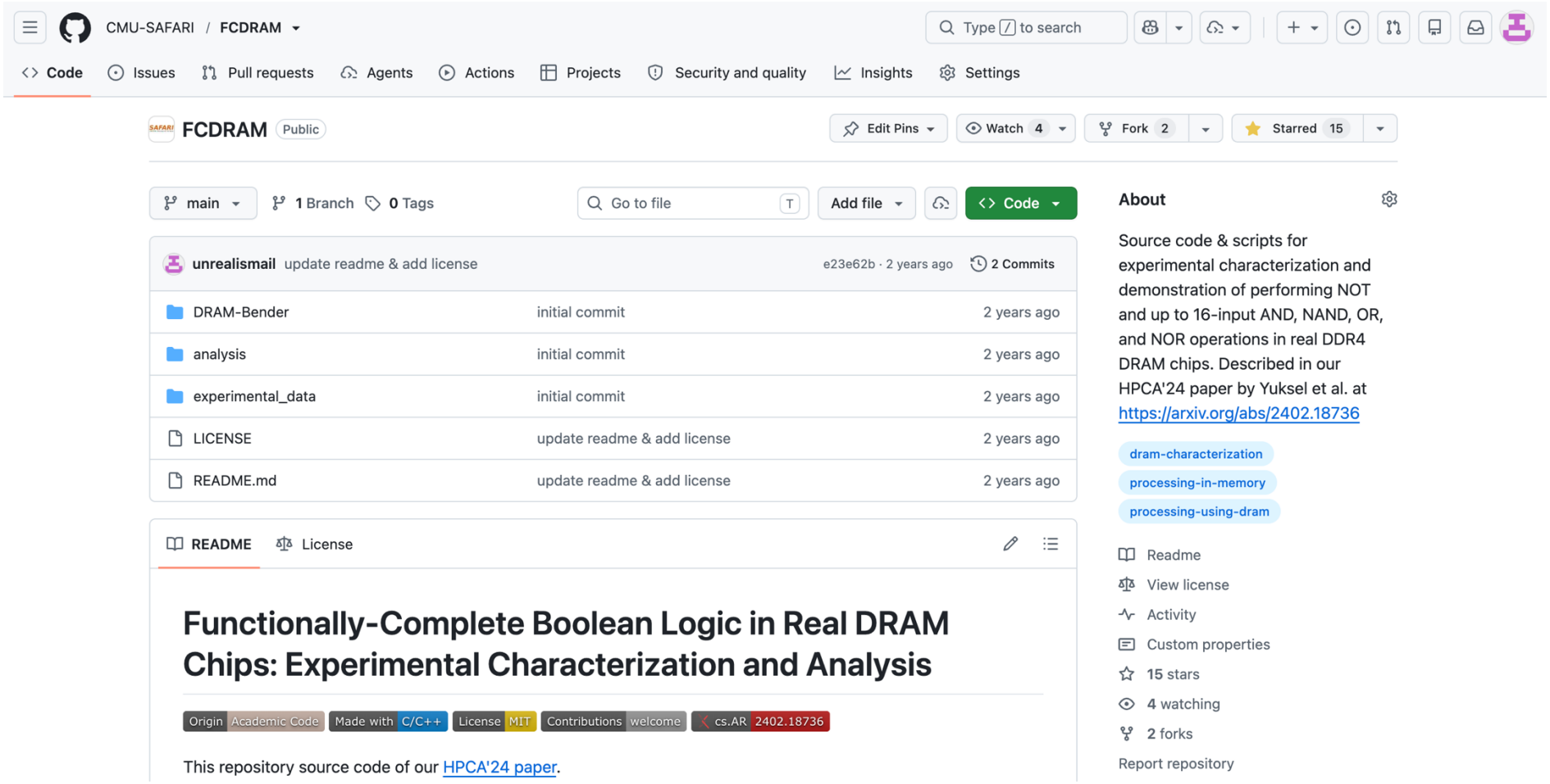
We experimentally demonstrate that COTS DRAM chips are capable of performing 1) functionally-complete *Boolean operations: NOT, NAND, and NOR and 2) many-input (i.e., more than two-input) AND and OR operations. We present an extensive*

systems and applications [12, 13]. Processing-using-DRAM (PuD) [29–32] is a promising paradigm that can alleviate the data movement bottleneck. PuD uses the analog operational properties of the DRAM circuitry to enable massively parallel in-DRAM computation. Many prior works [29–53] demonstrate that PuD can greatly reduce or eliminate data movement.

A widely used approach for PuD is to perform bulk bitwise operations, i.e., bitwise operations on large bit vectors. To perform bulk bitwise operations using DRAM, prior works propose modifications to the DRAM circuitry [29–31, 33, 35, 36, 43, 44, 46, 48–58]. Recent works [38, 41, 42, 45] experimentally demonstrate the feasibility of executing data copy & initialization [42, 45], i.e., the RowClone operation [49], and a subset of bitwise operations, i.e., three-input bitwise majority (MAJ3) and two-input AND and OR operations in unmodified commercial off-the-shelf (COTS) DRAM chips by operating beyond

https://arxiv.org/pdf/2402.18736.pdf

FCDRAM is Open Source



Functionally-Complete Boolean Logic in Real DRAM Chips

Experimental Characterization and Analysis

Ismail Emir Yüksel

Yahya C. Tugrul Ataberk Olgun F. Nisa Bostancı

A. Giray Yaglıkçı Geraldo F. Oliveira Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI

ETH zürich

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips

Experimental Characterization and Analysis

İsmail Emir Yüksel

Yahya C. Tuğrul F. Nisa Bostancı Geraldo F. Oliveira

A. Giray Yağlıkçı Ataberk Olgun Melina Soysal Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI ETH zürich

Leveraging Simultaneous Many-Row Activation

1 Perform **MAJX (where X>3)** operations

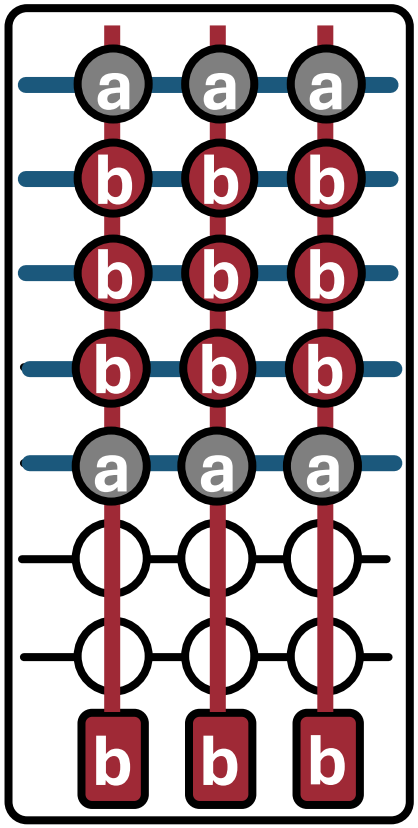
2 **Increase the robustness** of MAJX operations

3 Copy **one row's content** to **multiple rows**

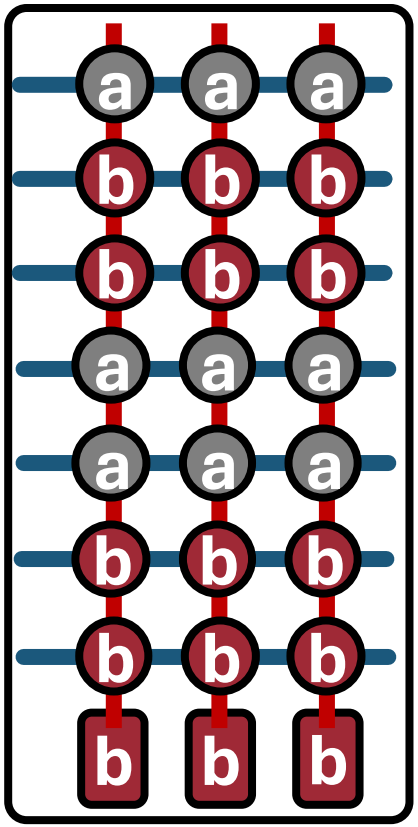
In-DRAM Majority-of-X (MAJX)

Simultaneously activate many rows to perform **MAJX (where X>3)** operations

MAJ5(a, b, b, b, a) = b

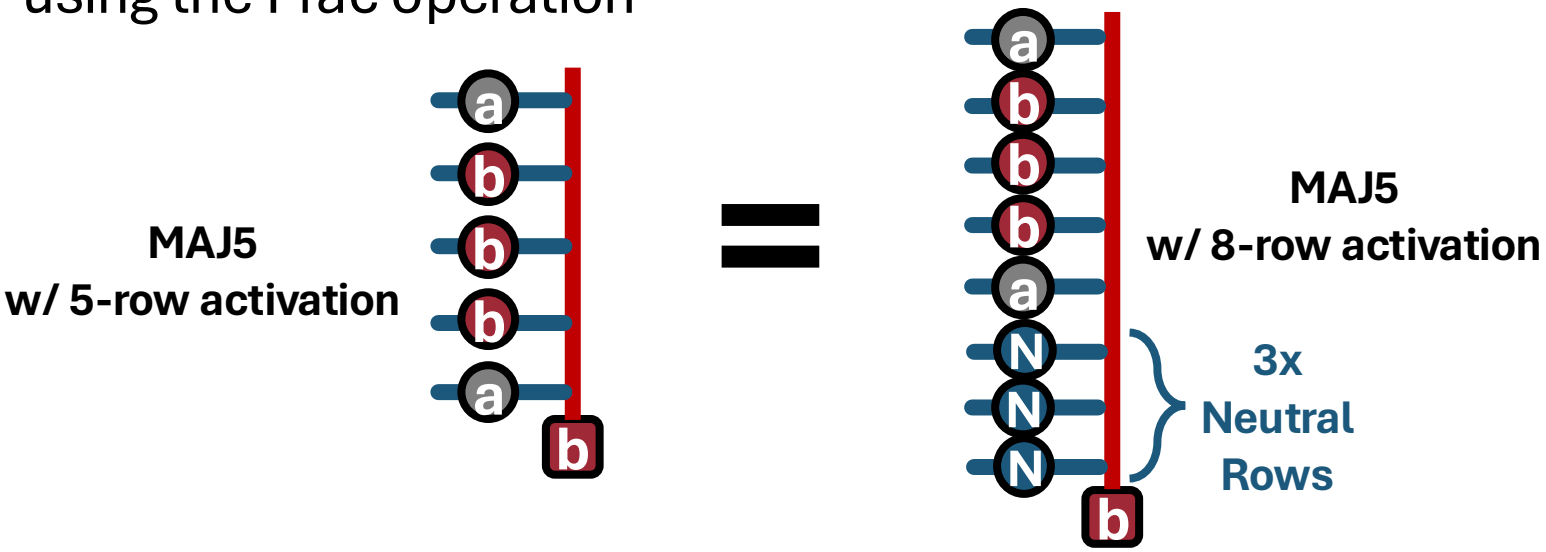


MAJ7(a, b, b, a, a, b, b) = b



MAJX in Real DRAM Chips

- For MAJX, we need to activate X rows simultaneously
- We can only simultaneously activate 2, 4, 8, 16, and 32 rows
- **Question**
  - How do we perform **MAJX** while simultaneously activating **more than X rows**?
- **Answer**
  - Making **some rows neutral** during the MAJX operation using the Frac operation*



*[Gao+MICRO'22]

Leveraging Simultaneous Many-Row Activation

1 Perform MAJX (where X>3) operations

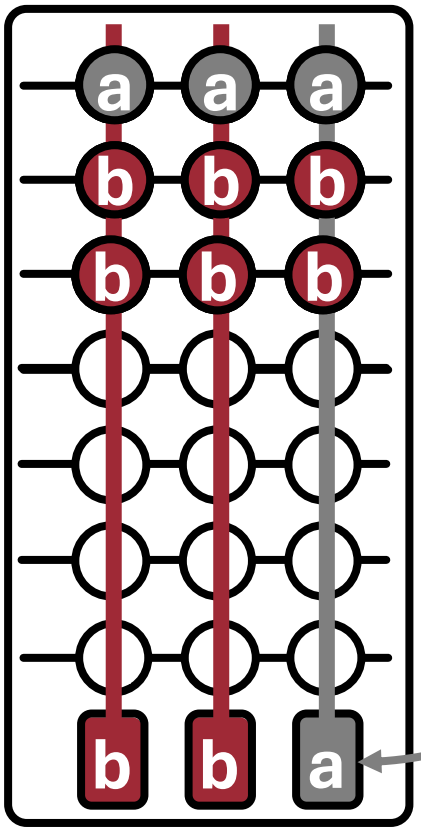
2 **Increase the robustness** of MAJX operations

3 Copy one row's content to multiple rows

Improving the Robustness (Input Replication)

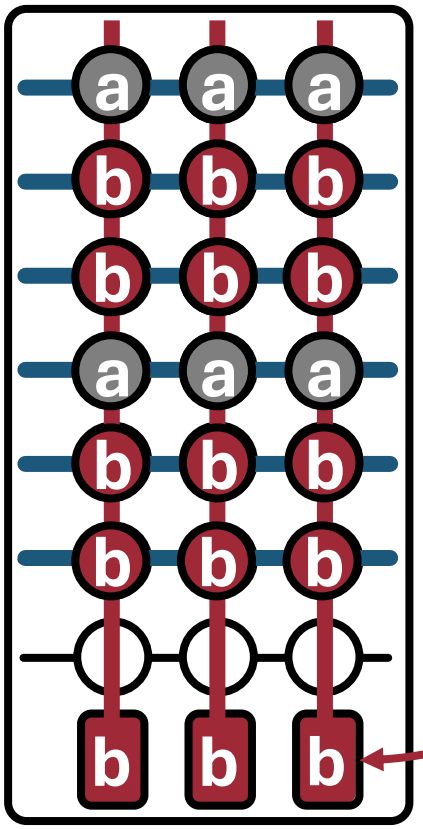
Storing **multiple copies** of MAJX input operands can **increase the robustness** of MAJX operations

MAJ3(a, b, b) = b



Incorrect result

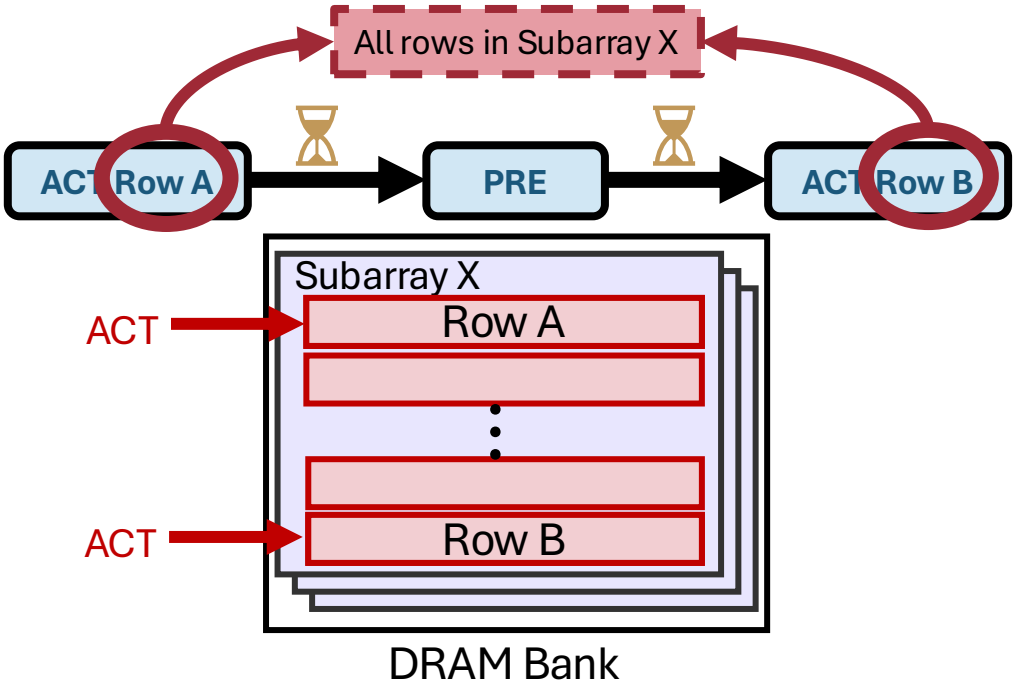
MAJ6(a, b, b, a, b, b) = b

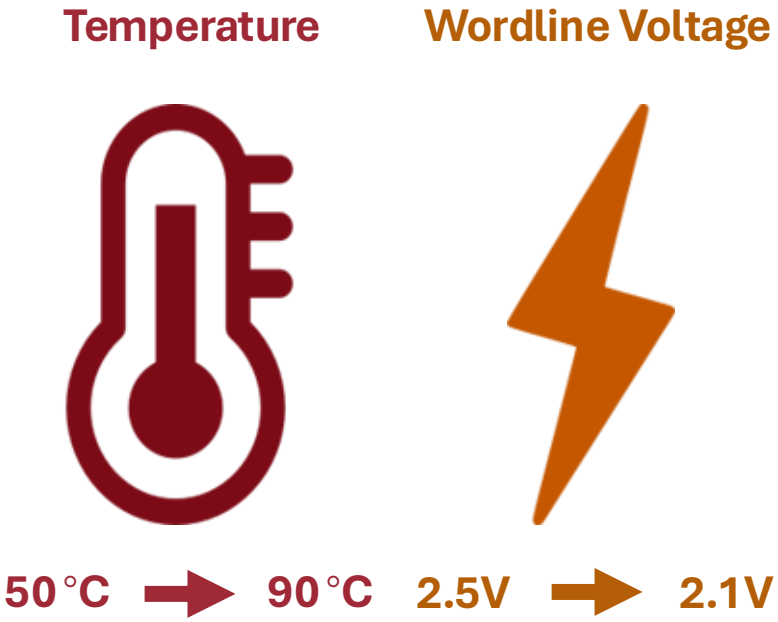


Correct result

Characterization Methodology

- Carefully sweep
  - Row addresses: Row A and Row B
  - Timing parameters: Between ACT → PRE and PRE → ACT
  - Temperature (°C): 50, 60, 70, 80, and 90
  - Wordline Voltage (V): 2.5, 2.4, 2.3, 2.2, and 2.1





Key Takeaways from MAJX Operation

Key Takeaway 1

COTS DRAM chips are capable of performing MAJ5, MAJ7, and MAJ9 operations

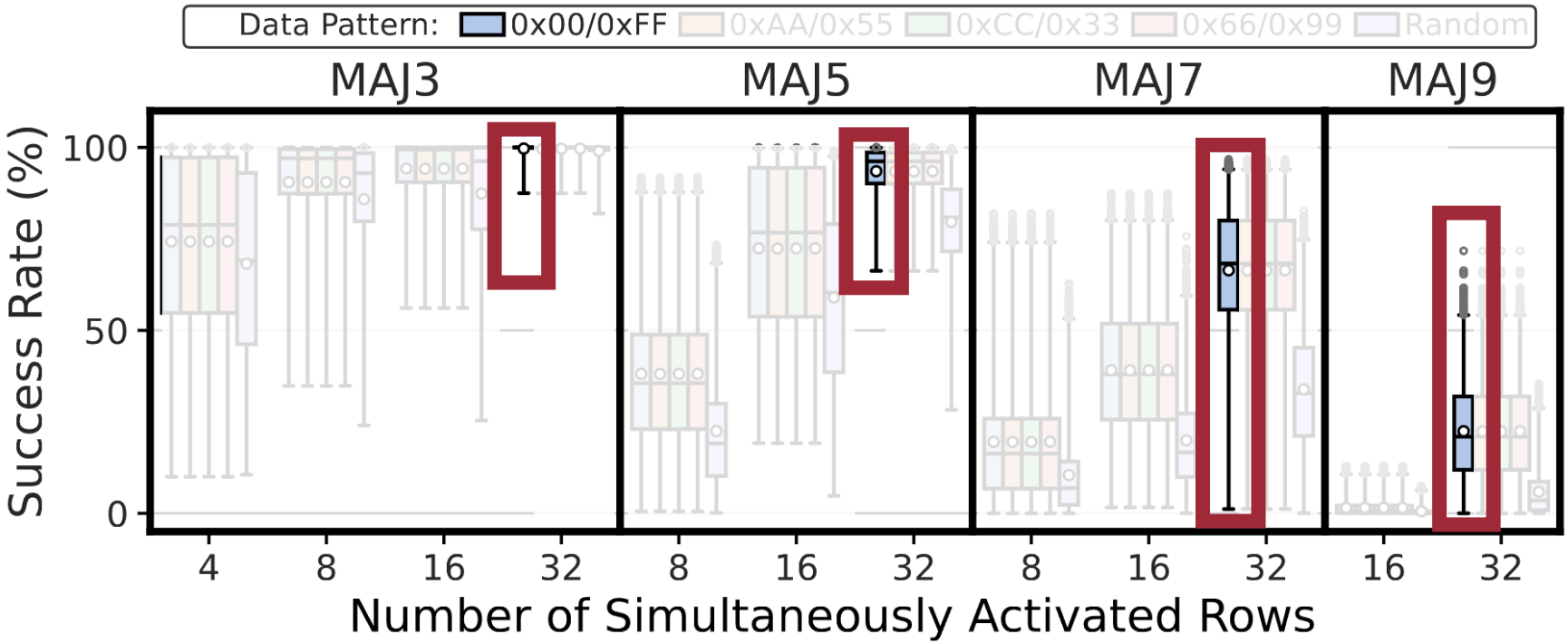
Key Takeaway 2

Storing multiple copies of MAJX's input operands significantly increases the MAJX's success rate

Key Takeaway 3

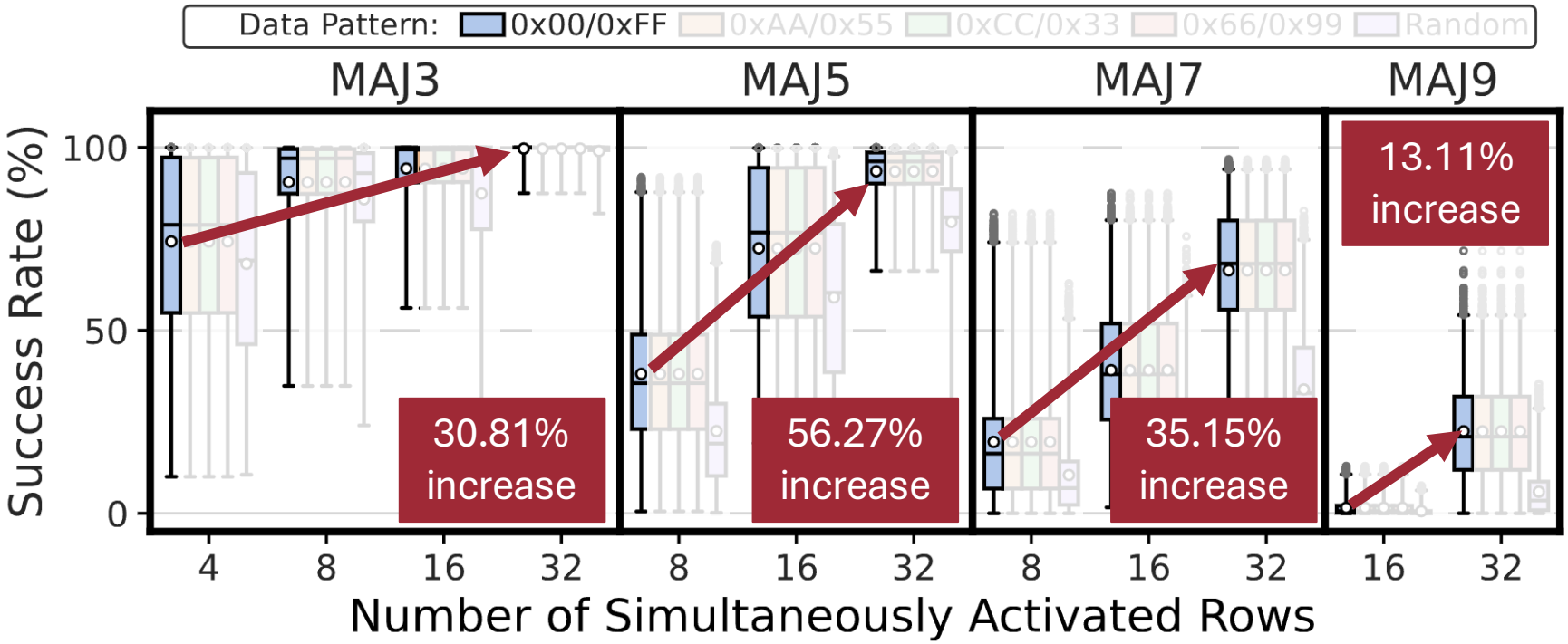
Voltage and temperature slightly affect the success rate, whereas data pattern affects significantly

Robustness of MAJX Operations



COTS DRAM chips are capable of performing MAJ5, MAJ7, and MAJ9 operations

Impact of Input Replication



Similar trend in success rate increase in all tested MAJX operations

Final average success rate MAJ3 (MAJ5): 99.68% (93.49%)

Storing multiple copies of MAJ's input operands increases the success rate of MAJ3, MAJ5, MAJ7, and MAJ9 operations

Leveraging Simultaneous Many-Row Activation

1 Perform **MAJX (where X>3)** operations

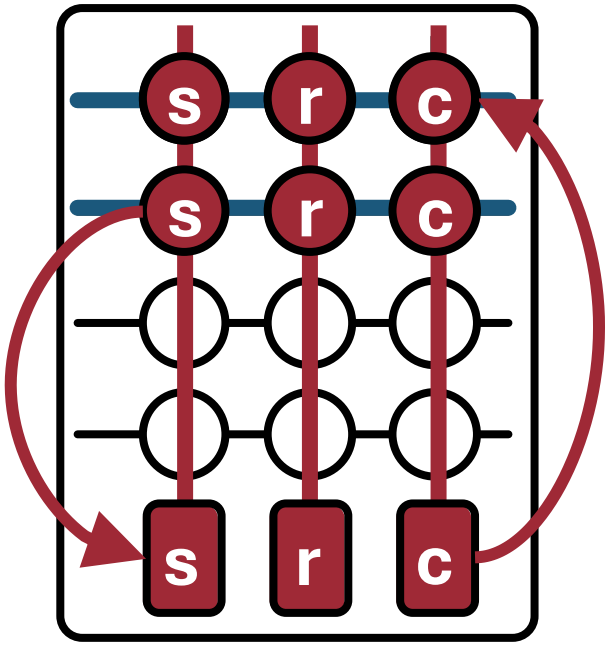
2 **Increase the robustness** of MAJX operations

3 Copy **one row's content** to **multiple rows**

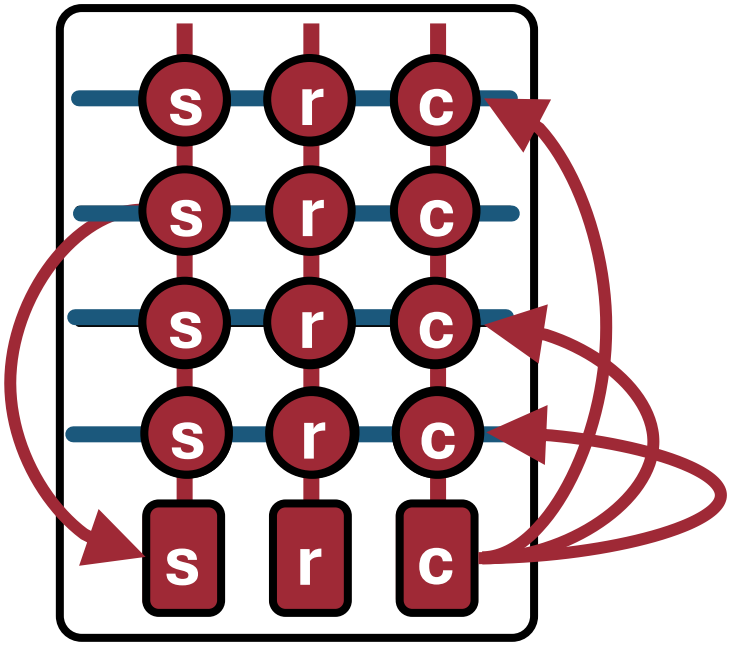
In-DRAM Multiple Row Copy (Multi-RowCopy)

Simultaneously activate many rows to
copy **one row's content** to **multiple destination rows**

RowClone

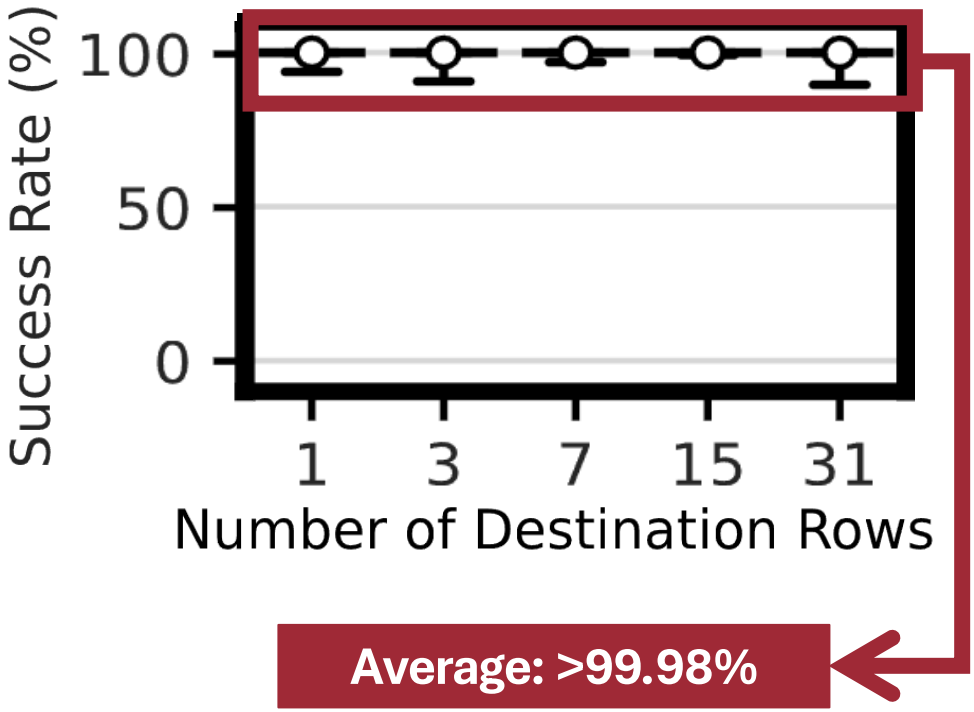


Multi-RowCopy



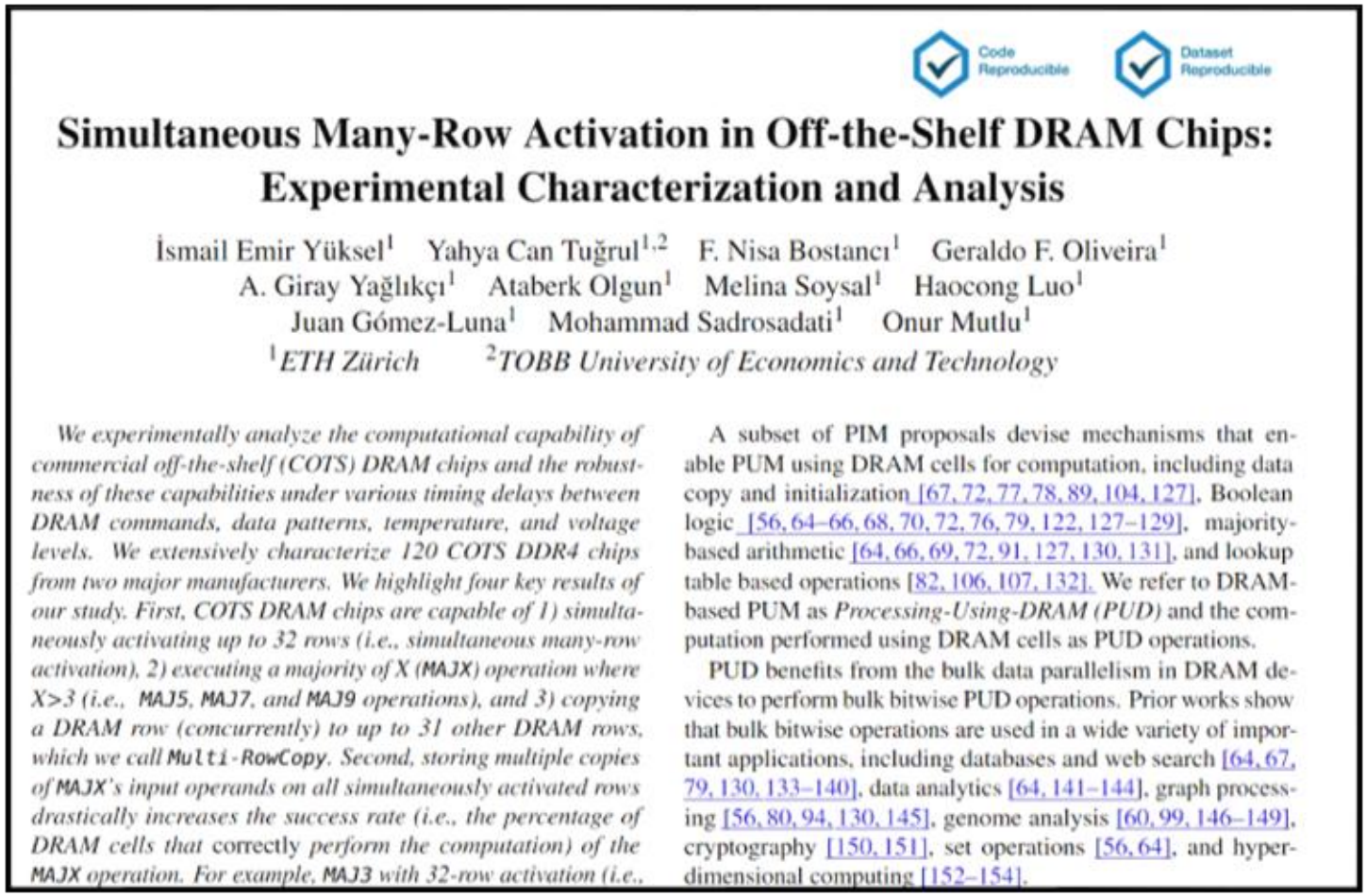
[Seshadri+ MICRO'13]

Robustness of Multi-RowCopy



COTS DRAM chips can copy one row's content to up to 31 rows with a very high success rate

Available on arXiv



Code Reproducible

Dataset Reproducible

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

We experimentally analyze the computational capability of commercial off-the-shelf (COTS) DRAM chips and the robustness of these capabilities under various timing delays between DRAM commands, data patterns, temperature, and voltage levels. We extensively characterize 120 COTS DDR4 chips from two major manufacturers. We highlight four key results of our study. First, COTS DRAM chips are capable of 1) simultaneously activating up to 32 rows (i.e., simultaneous many-row activation), 2) executing a majority of X (`MAJX`) operation where X>3 (i.e., `MAJ5`, `MAJ7`, and `MAJ9` operations), and 3) copying a DRAM row (concurrently) to up to 31 other DRAM rows, which we call `Multi-RowCopy`. Second, storing multiple copies of `MAJX`'s input operands on all simultaneously activated rows drastically increases the success rate (i.e., the percentage of DRAM cells that correctly *perform the computation) of the `MAJX` operation. For example, `MAJ3` with 32-row activation (i.e.,*

A subset of PIM proposals devise mechanisms that enable PUM using DRAM cells for computation, including data copy and initialization [67, 72, 77, 78, 89, 104, 127], Boolean logic [56, 64–66, 68, 70, 72, 76, 79, 122, 127–129], majority-based arithmetic [64, 66, 69, 72, 91, 127, 130, 131], and lookup table based operations [82, 106, 107, 132]. We refer to DRAM-based PUM as *Processing-Using-DRAM (PUD)* and the computation performed using DRAM cells as PUD operations.

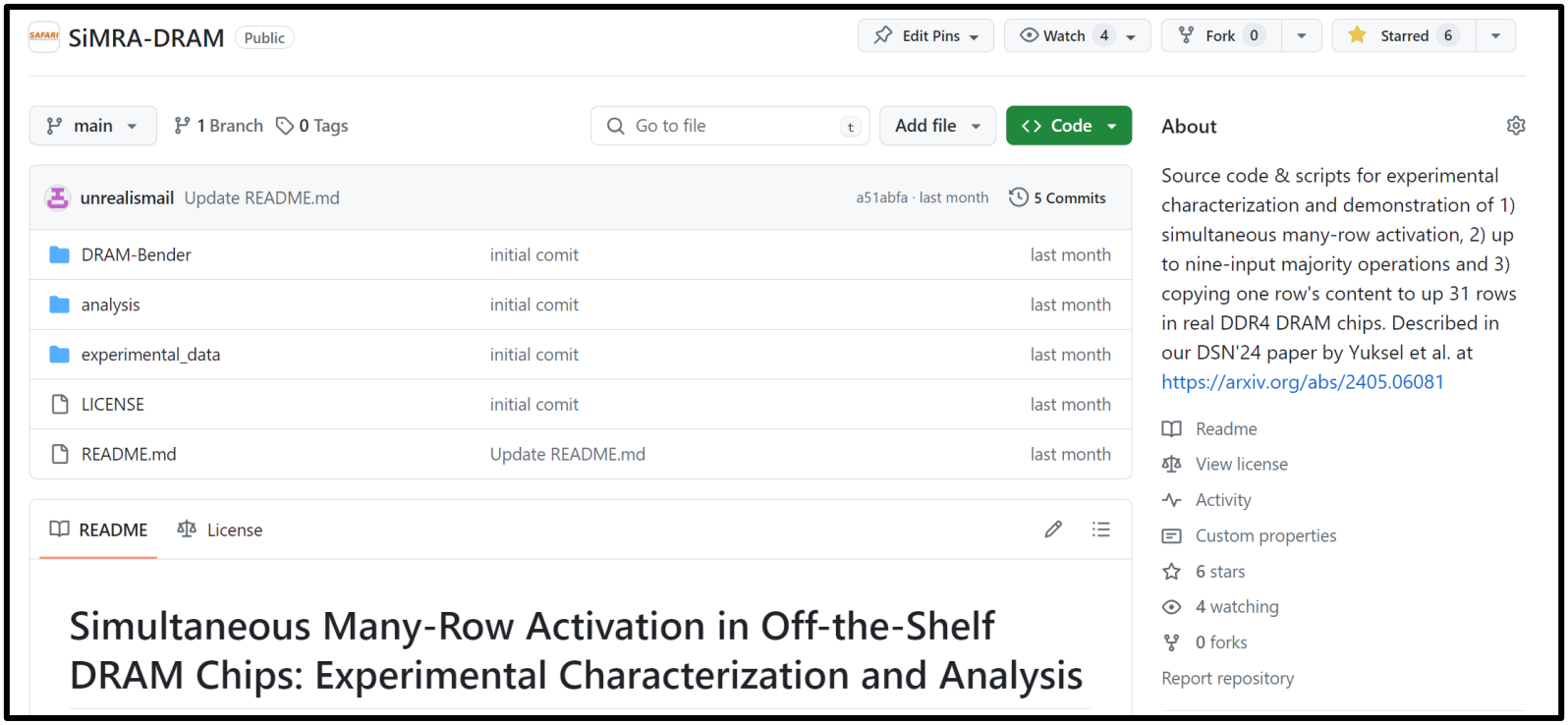
PUD benefits from the bulk data parallelism in DRAM devices to perform bulk bitwise PUD operations. Prior works show that bulk bitwise operations are used in a wide variety of important applications, including databases and web search [64, 67, 79, 130, 133–140], data analytics [64, 141–144], graph processing [56, 80, 94, 130, 145], genome analysis [60, 99, 146–149], cryptography [150, 151], set operations [56, 64], and hyperdimensional computing [152–154].

https://arxiv.org/pdf/2405.06081

Our Work is Open Source and Artifact Evaluated

Code Reproducible

Dataset Reproducible



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

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips

Experimental Characterization and Analysis

Paper

GitHub

İsmail Emir Yüksel

Yahya C. Tuğrul F. Nisa Bostancı Geraldo F. Oliveira

A. Giray Yağlıkçı Ataberk Olgun Melina Soysal Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI ETH zürich

PuDHammer

Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips

İsmail Emir Yüksel

Akash Sood Ataberk Olgun Oğuzhan Canpolat Haocong Luo

Nisa Bostancı Mohammad Sadrosadati Giray Yağlıkçı Onur Mutlu

SAFARI

ETH zürich

Executive Summary

Motivation: Processing-using-DRAM (PuD) alleviates data movement bottlenecks

- DRAM can perform many PuD operations by activating multiple DRAM rows in quick succession or simultaneously (i.e., multiple-row activation)
- Modern DRAM is subject to read disturbance (e.g., RowHammer)
  - Repeatedly activating even a single row induces bitflips in unaccessed rows

Problem: No prior work studies the read disturbance effects of multiple-row activation

Goal: Understand and analyze read disturbance effects of multiple-row activation

Experimental Study: 316 COTS DDR4 chips from four major manufacturers to study read disturbance effects of multiple-row activation, which we call PuDHammer

- PuDHammer significantly exacerbates DRAM read disturbance, causing up to 158.58x reduction in min. hammer count to induce first bitflip
- PuDHammer bypasses in-DRAM RowHammer mitigation in DDR4 and induces 11340x more bitflips than RowHammer

Mitigation: We adapt Per Row Activation Counting (PRAC) for PuDHammer

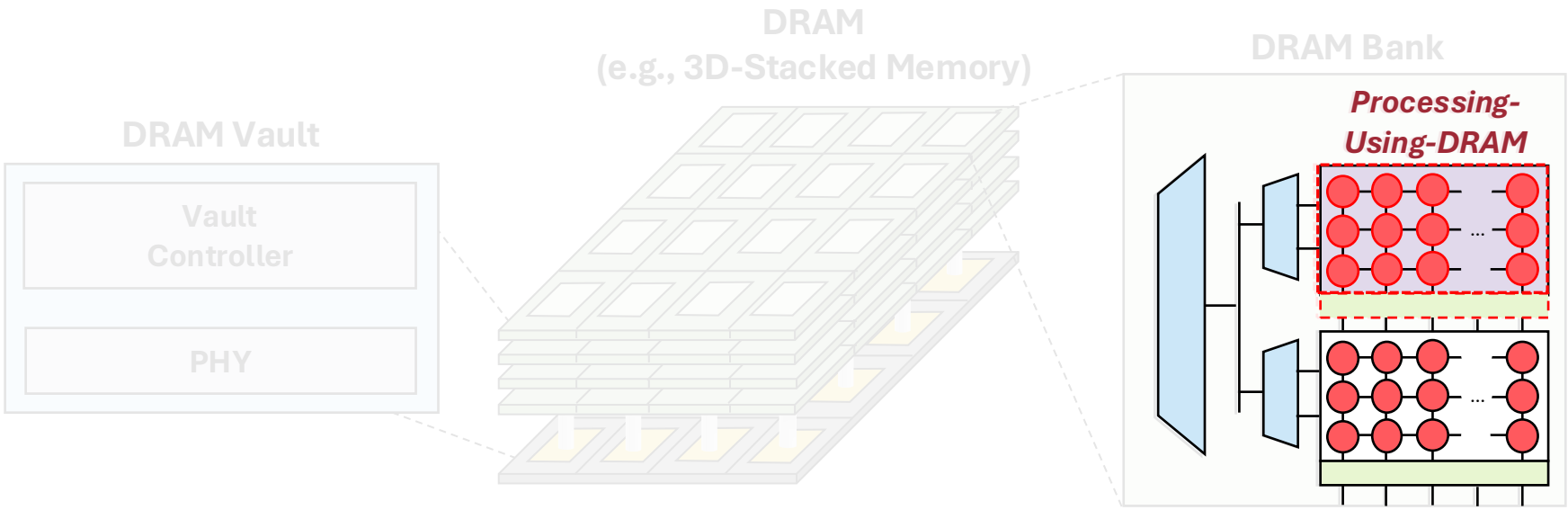
- Adapted PRAC solution incurs an average system performance overhead of 48.26%

Processing-using-DRAM (PuD)

Multiple-row activation is a key technique to realize many PuD operations

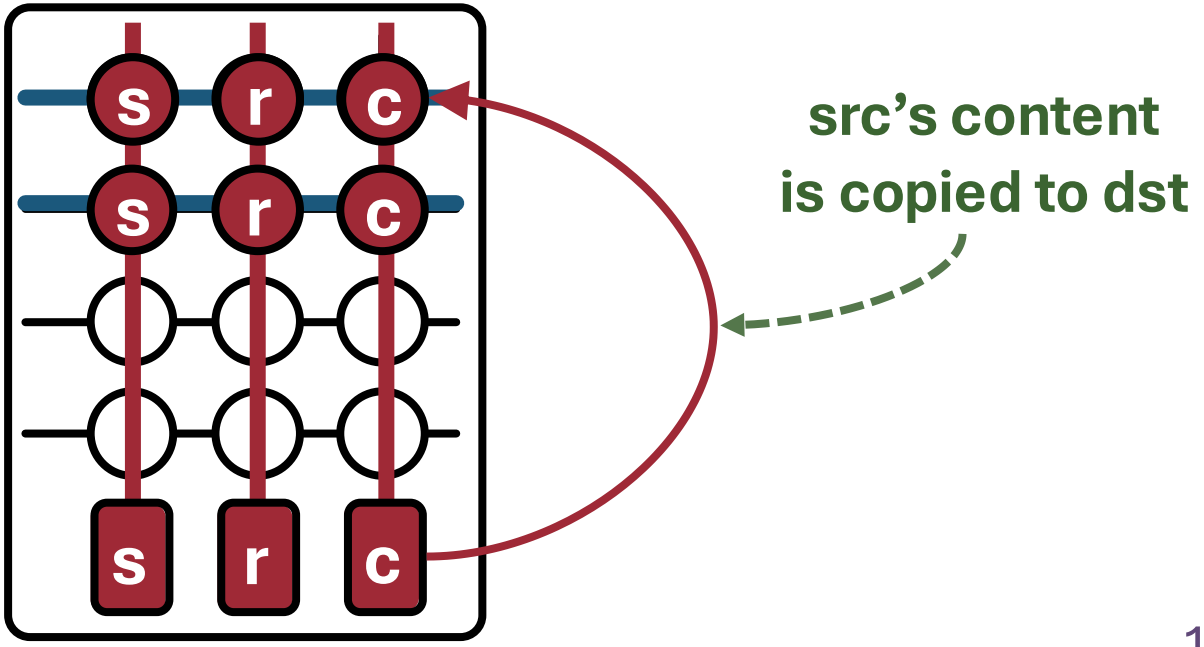
1. **Consecutive Multiple-Row Activation (CoMRA)**

2. **Simultaneous Multiple-Row Activation (SiMRA)**



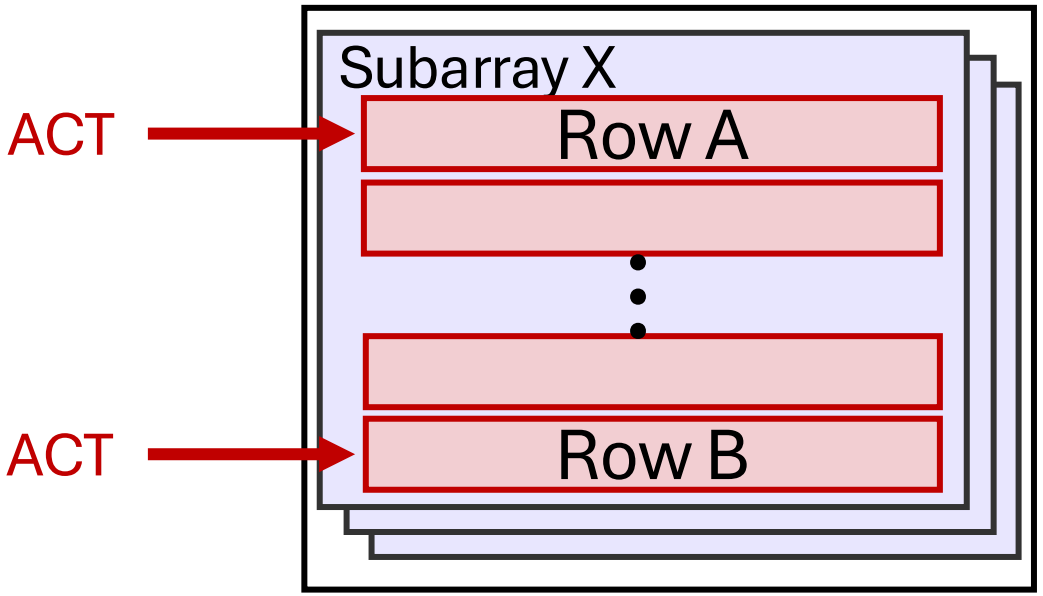
Consecutive Multiple-Row Activation

Violating timing of PRE command **greatly**
in **COTS** DRAM chips
activates two rows **consecutively**



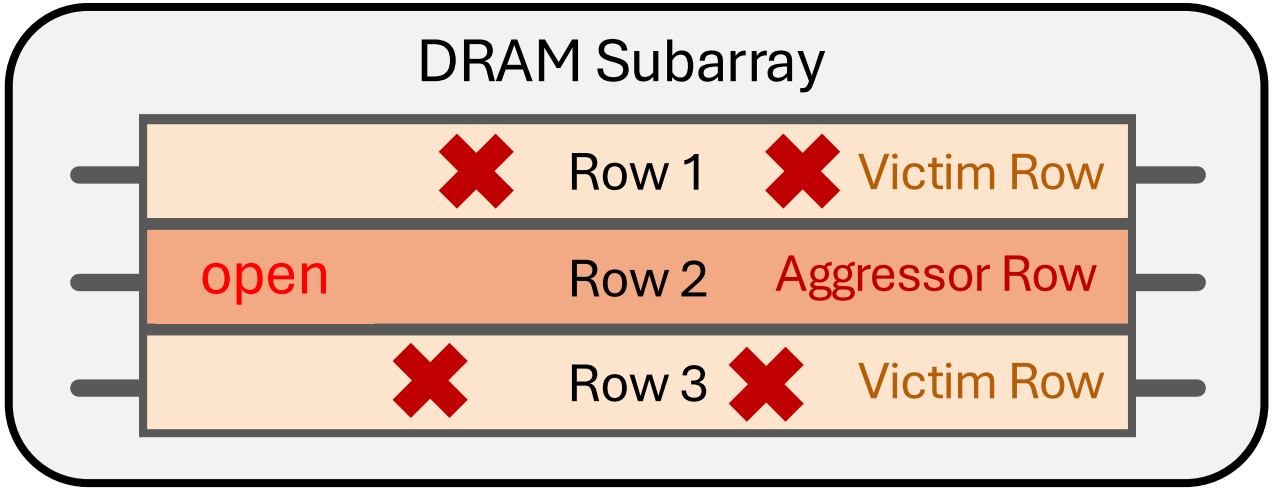
Simultaneous Multiple-Row Activation

Activating two rows in **quick succession** in **COTS** DRAM chips **simultaneously** activates **multiple rows** in a subarray



Read Disturbance in DRAM

- Unfortunately, even activating **a single DRAM row** **disturbs** the data integrity of other **unaccessed DRAM rows**
- Prominent example: **RowHammer**



Repeatedly **opening (activating)** and **closing** a DRAM row **many times** causes **RowHammer bitflips** in adjacent rows

[Kim+, ISCA'14]

Problem & Goal

Problem

*No prior work **investigates** the **read disturbance effects of multiple-row activation***

Goal

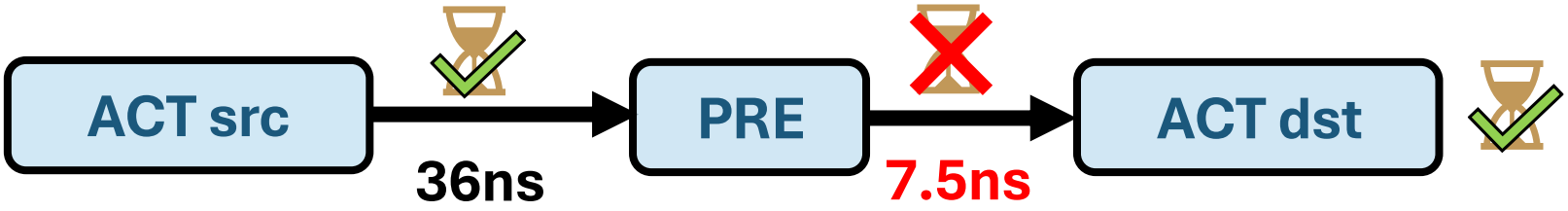
*Understand how **multiple-row activation affects read disturbance vulnerability** in DRAM*

DRAM Chips Tested

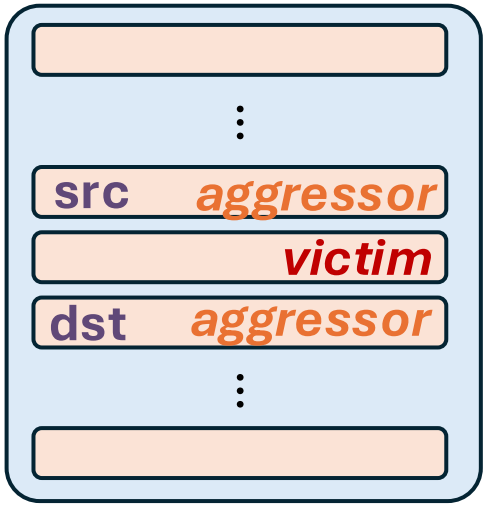
- 316 DDR4 chips from four major DRAM manufacturers
- Covers different die revisions and chip densities

Chip Mfr.	#Modules	#Chips	Die Rev.	Density	Org.
SK Hynix	1	8	A	4Gb	x8
	8	64	A	8Gb	x8
	2	16	C	16Gb	x8
	6	48	D	8Gb	x8
Micron	1	8	B	4Gb	x8
	4	32	E	16Gb	x16
	4	32	F	16Gb	x8
	2	16	R	8Gb	x8
Samsung	1	8	A	16Gb	x8
	5	40	B	16Gb	x8
	1	4	C	4Gb	x16
	1	8	C	16Gb	x8
	1	8	E	4Gb	x8
Nanya	3	24	C	8Gb	x8

Hammering with CoMRA

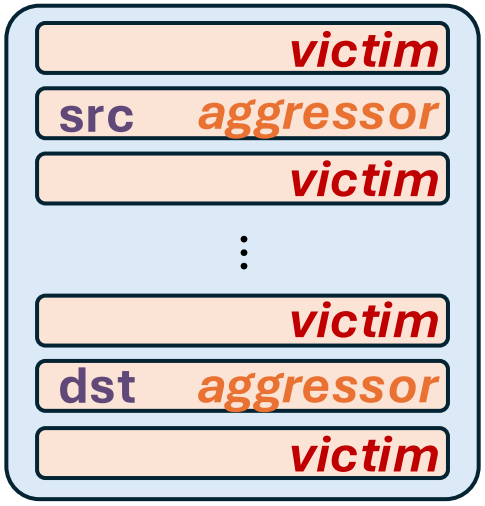


Double-Sided



DRAM Subarray

Single-Sided

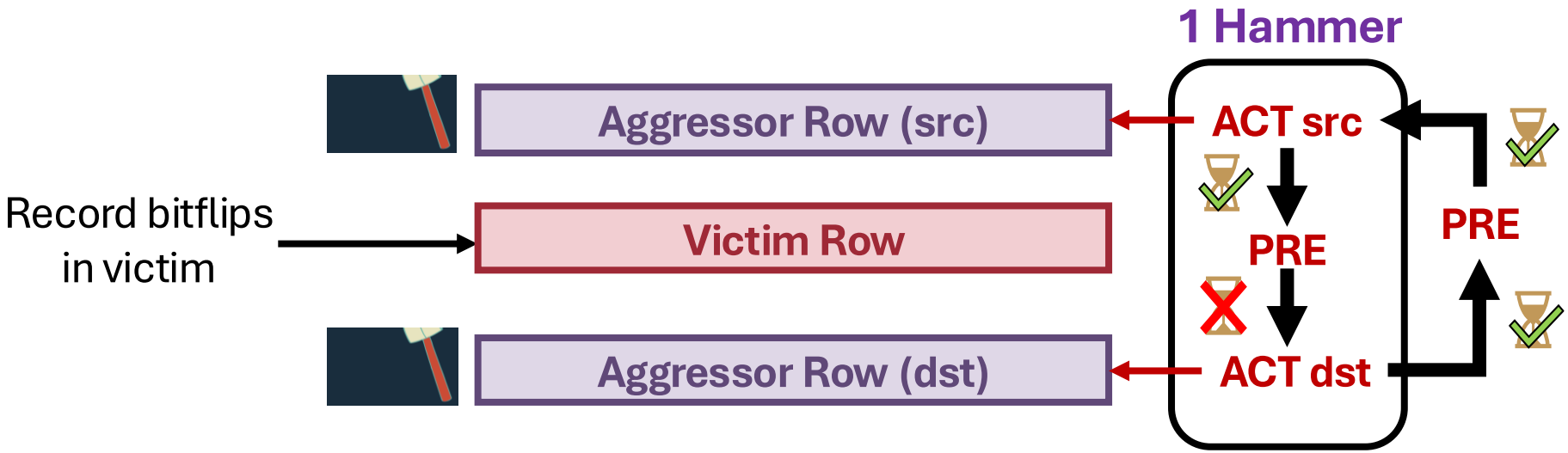


DRAM Subarray

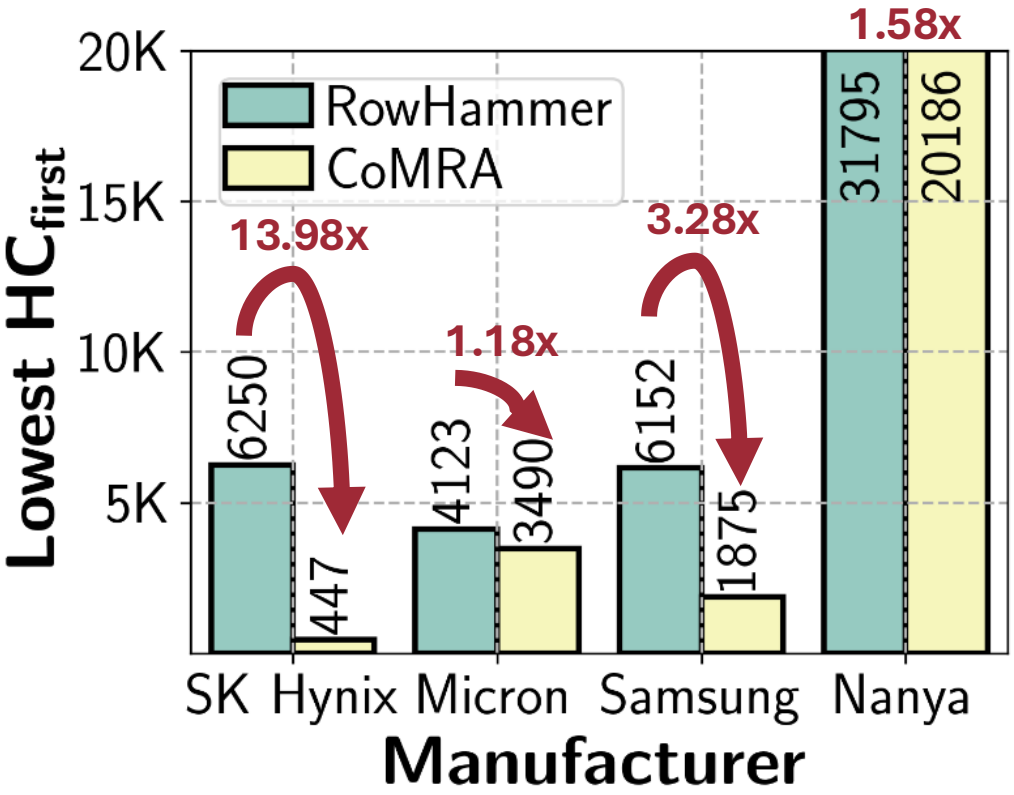
Read Disturbance Vulnerability Metric: HC_{first}

The minimum hammer count
required to induce the **first bitflip**

Lower is worse



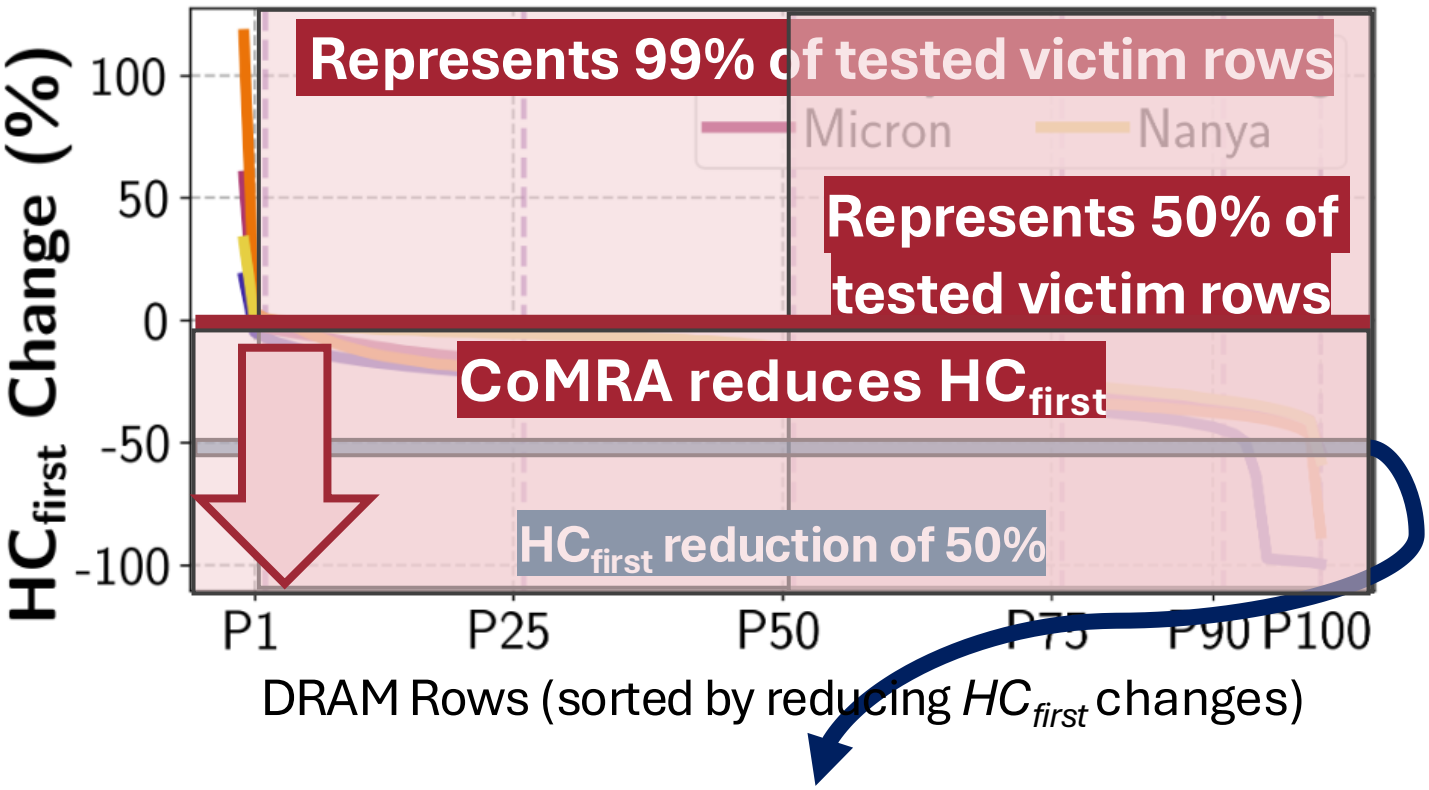
Double-Sided CoMRA vs. RowHammer (I)



Double-sided CoMRA decreases lowest HC_{first} by 13.98x compared to double-sided RowHammer

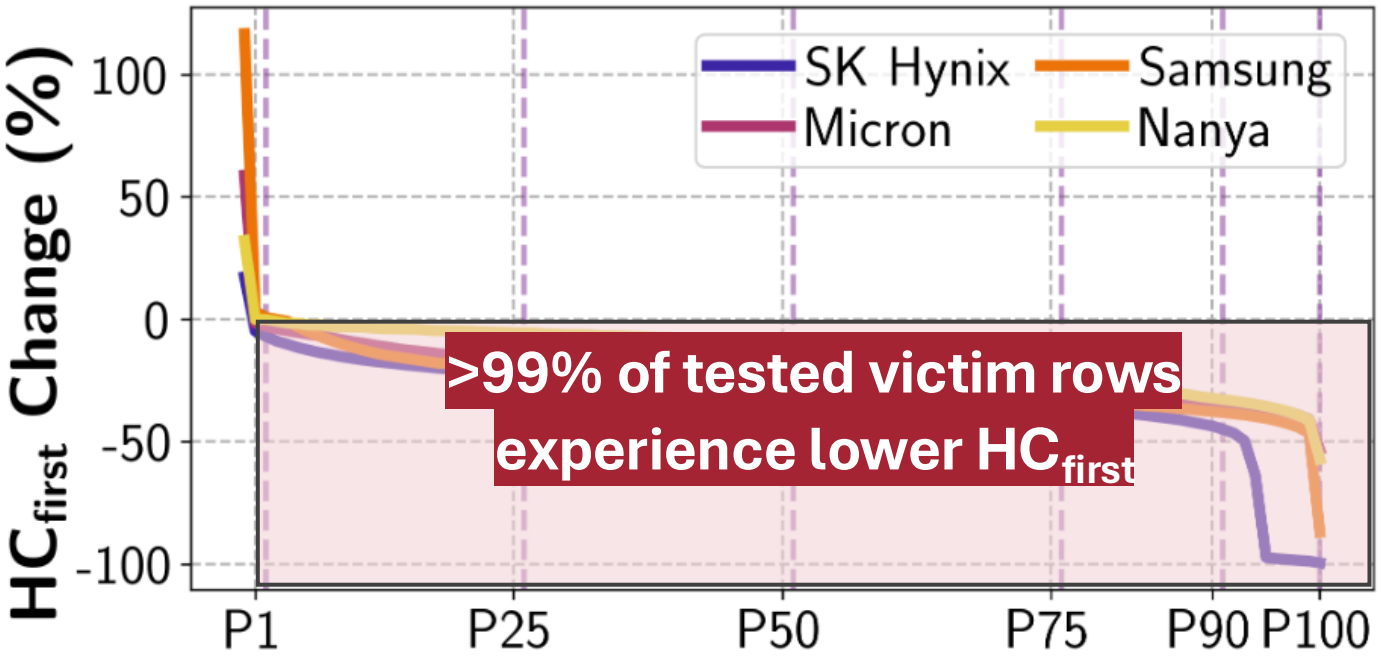
Lowest HC_{first} reduction trend consistent across all four manufacturer

Double-Sided CoMRA vs. RowHammer (II)



RowHammer HC_{first} = 100
CoMRA HC_{first} = 50

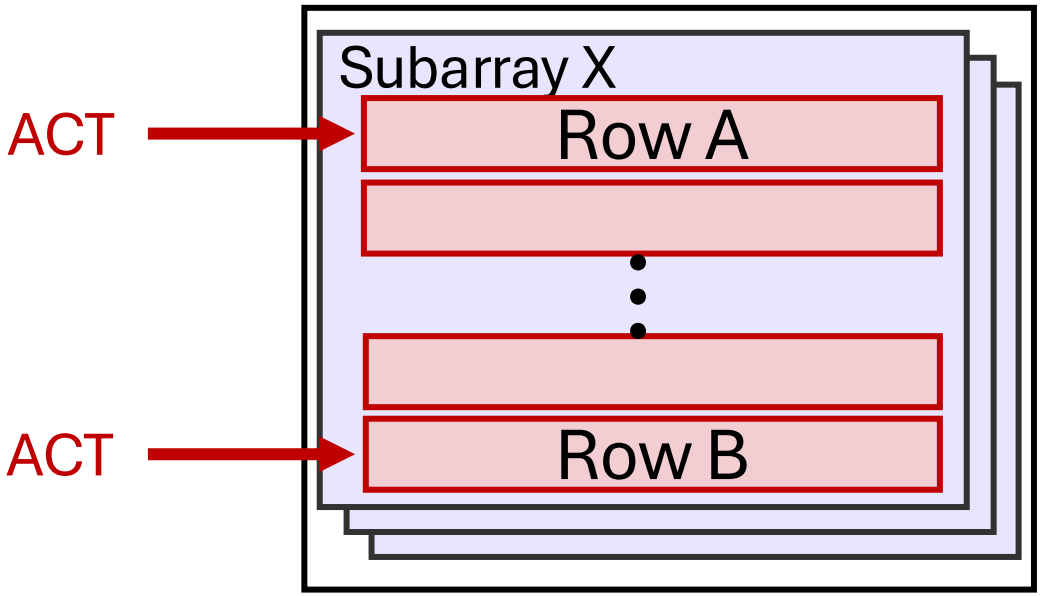
Double-Sided CoMRA vs. RowHammer (II)



Double-sided CoMRA decreases HC_{first} for a large fraction of rows compared to double-sided RowHammer

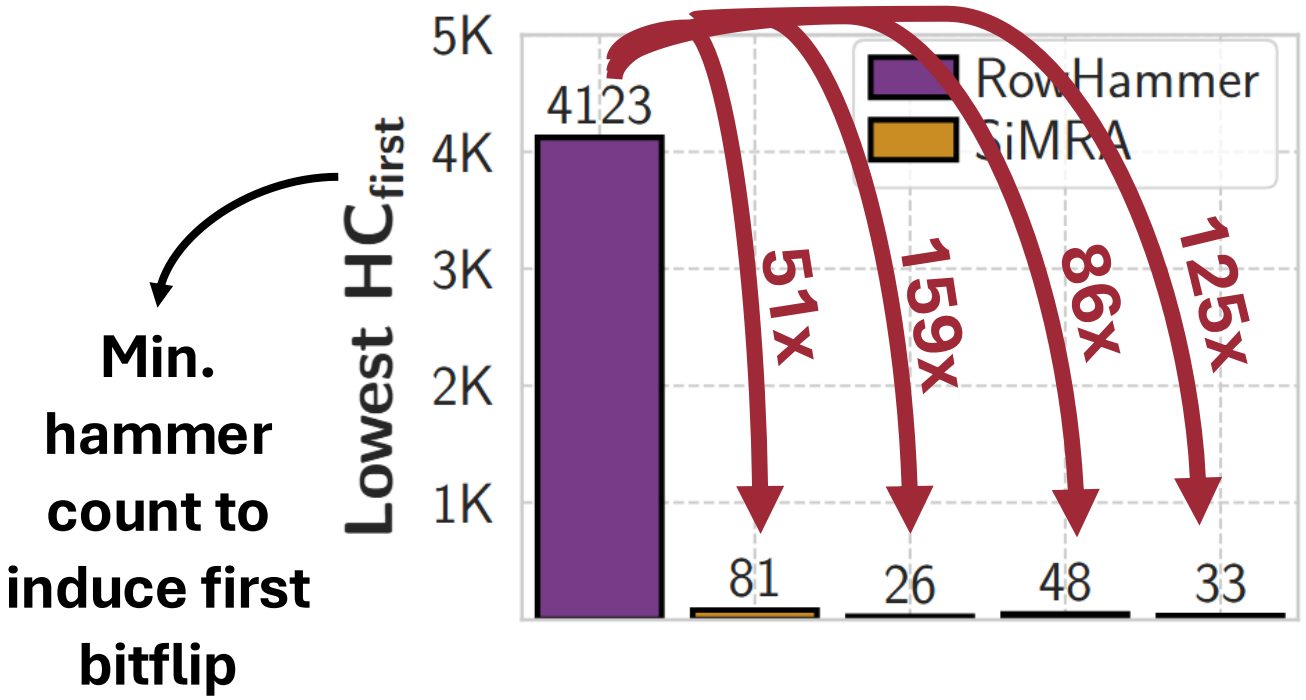
Recall: SiMRA

Activating two rows in **quick succession** in **COTS** DRAM chips **simultaneously** activates **multiple rows** in a subarray



DRAM Bank

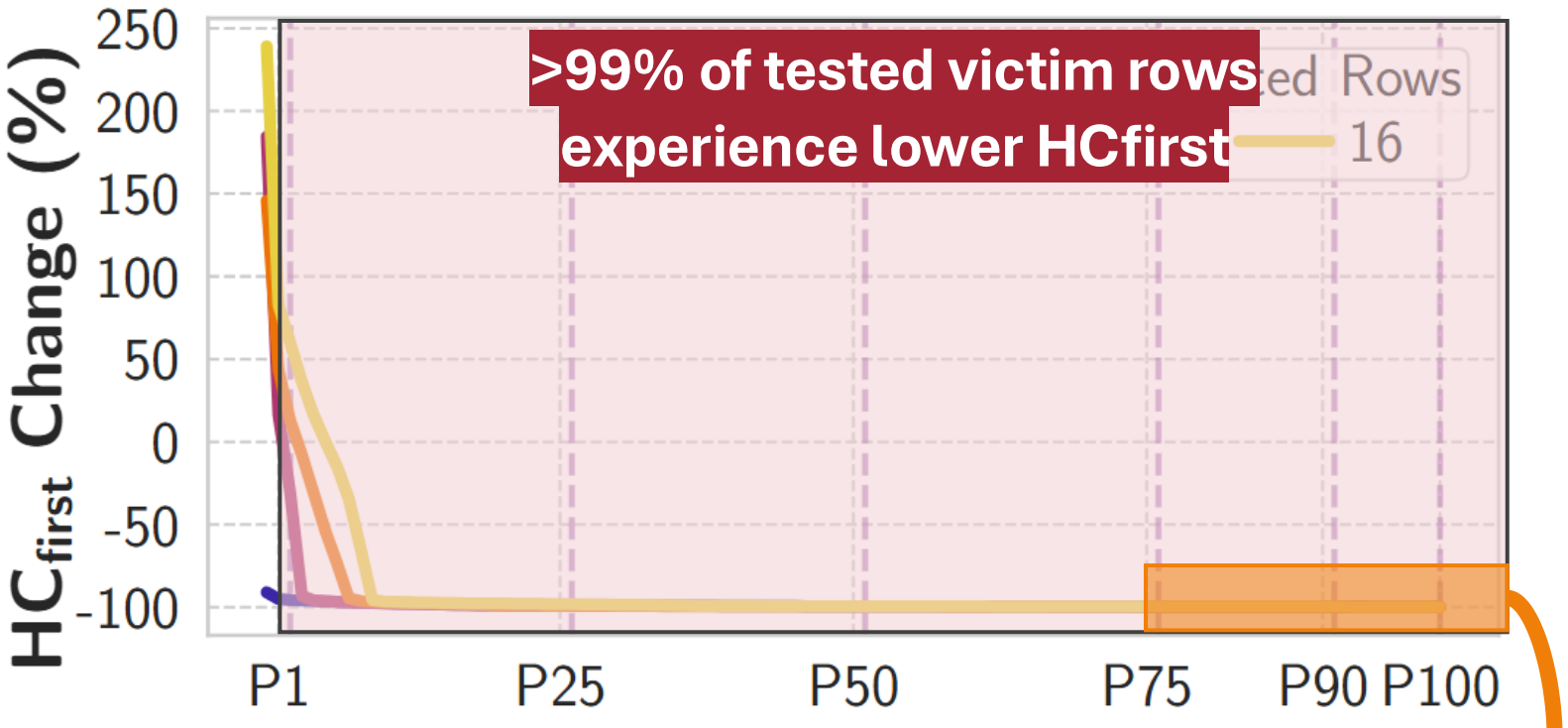
Double-Sided SiMRA vs. RowHammer (I)



Double-sided SiMRA decreases lowest HC_{first} by 159x compared to double-sided RowHammer

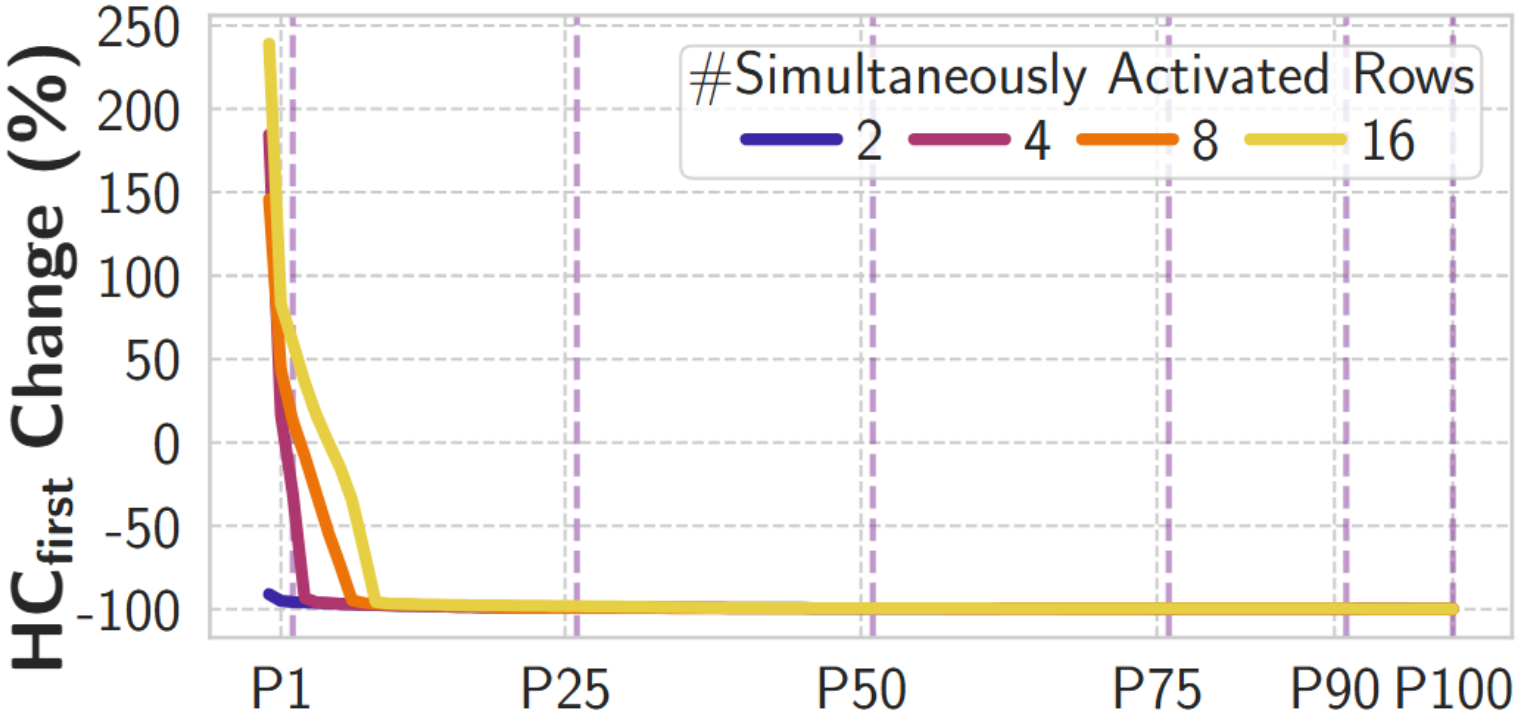
Lowest HC_{first} reduction trend consistent across all tested number of simultaneously activated rows

Double-Sided SiMRA vs. RowHammer (II)



at least 25.19%
of tested victim rows
experience >99% HCfirst reduction

Double-Sided SiMRA vs. RowHammer (II)



Double-sided SiMRA significantly decreases HC_{first} for a majority of rows compared to double-sided RowHammer

Mitigating PuDHammer

1 Analyze the effectiveness of existing in-DRAM RowHammer mitigation against PuDHammer

2 Adapt RowHammer mitigations to account for PuDHammer

Mitigating PuDHammer

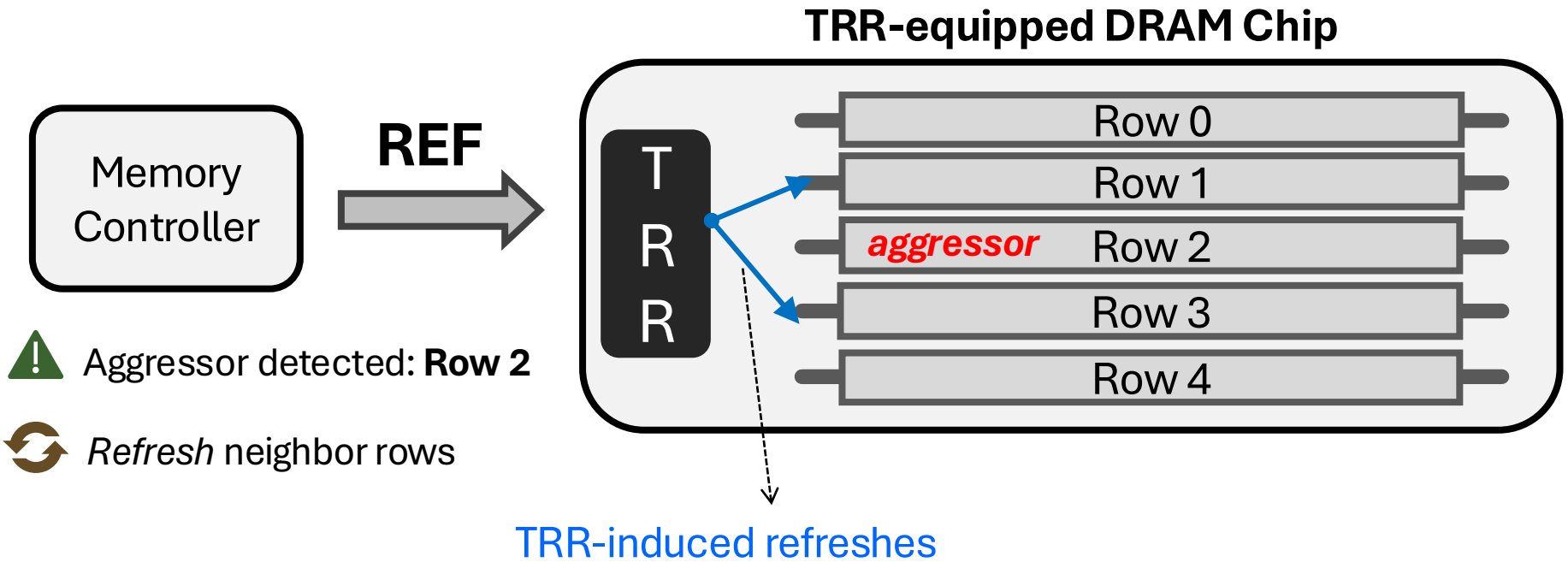
1 **Analyze the effectiveness of existing in-DRAM RowHammer mitigation against PuDHammer**

2 **Adapt RowHammer mitigations to account for PuDHammer**

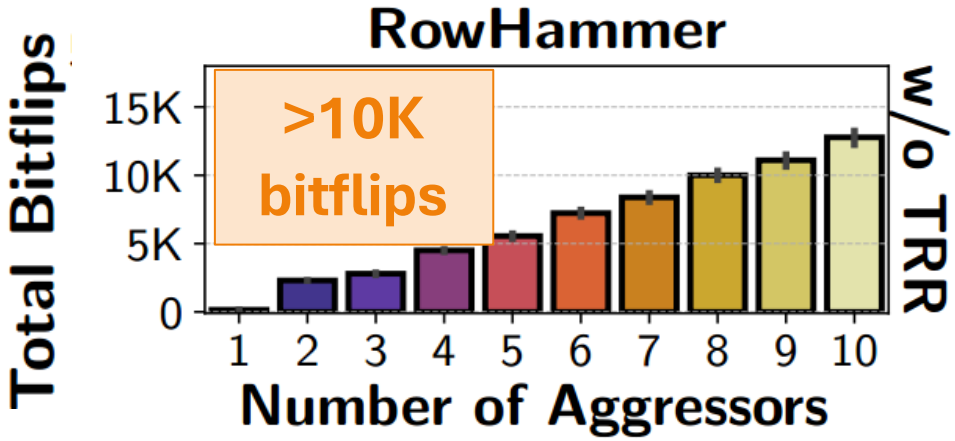
Target Row Refresh (TRR)

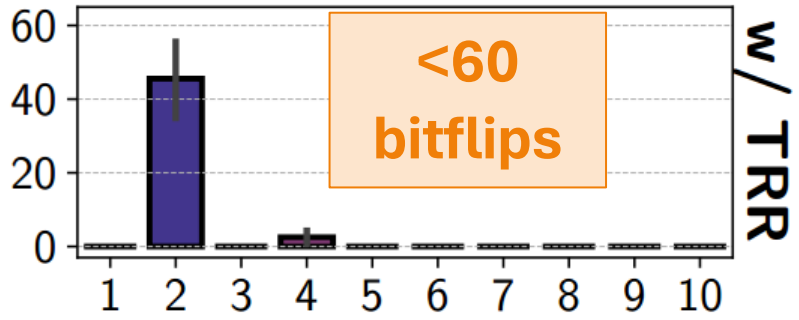
DRAM vendors equip their DRAM chips with a *proprietary* mitigation mechanisms known as **Target Row Refresh (TRR)**

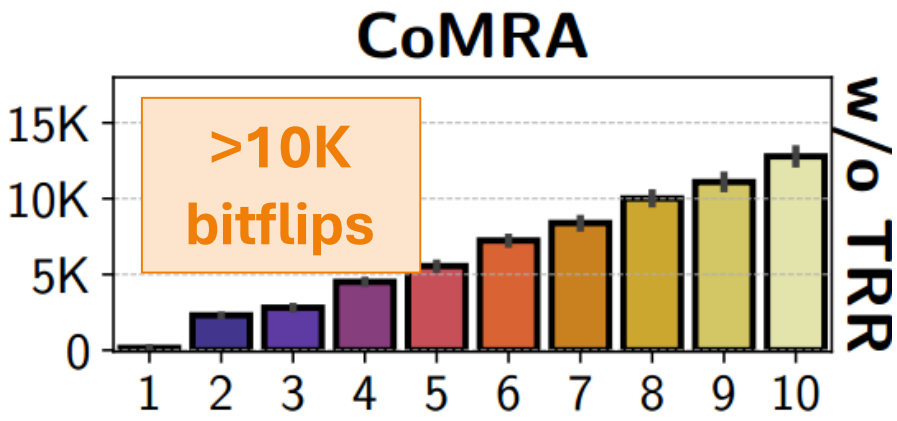
Key Idea: TRR refreshes nearby rows upon detecting an aggressor row

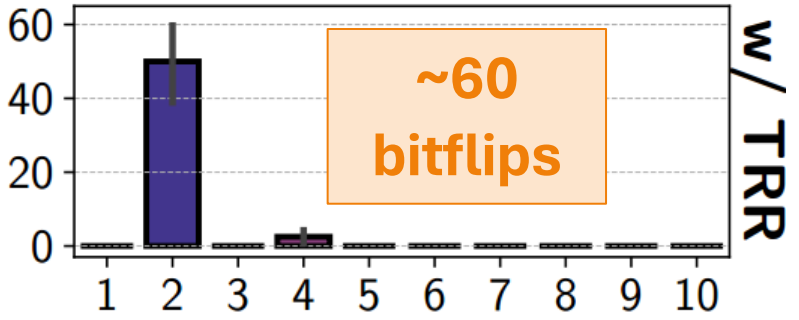


Real DRAM Chip Results



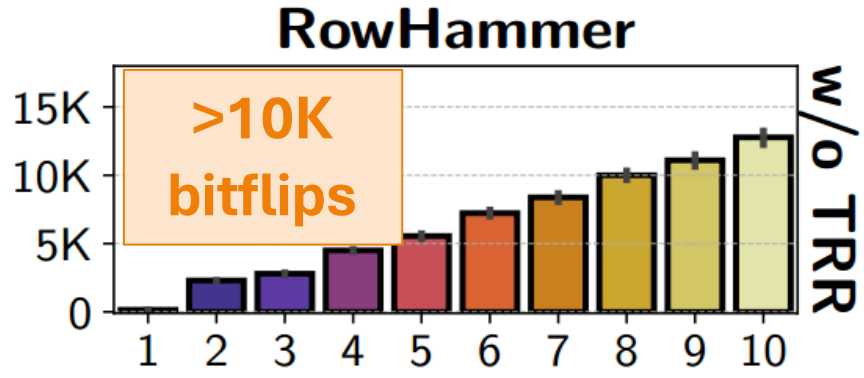


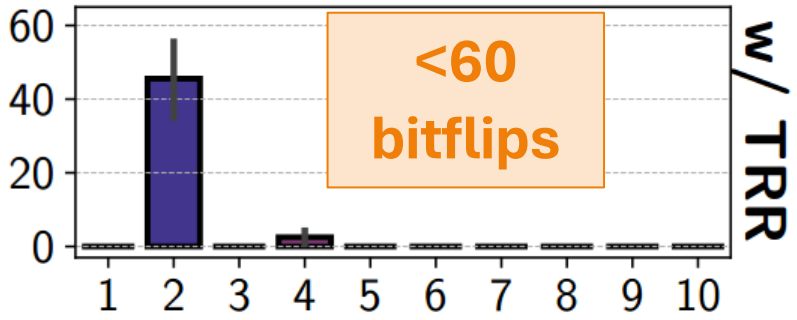


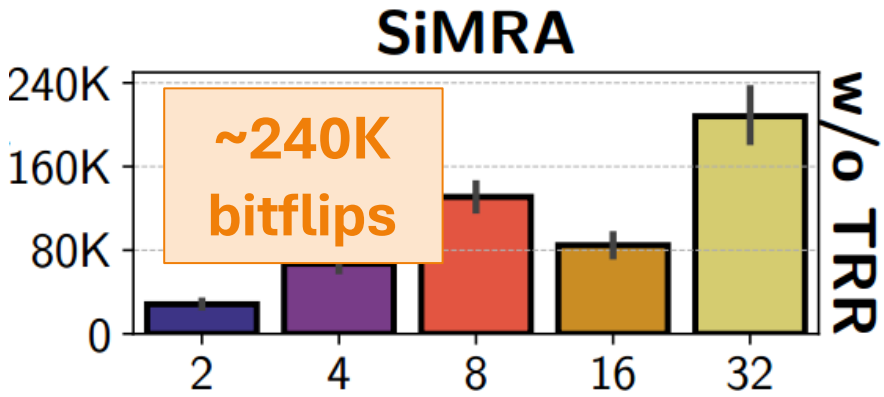


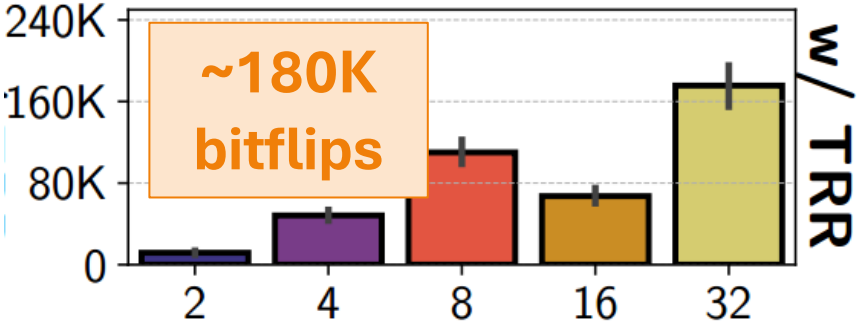
CoMRA induces 1.10x more bitflips than RowHammer on average in the presence of TRR

Real DRAM Chip Results









SiMRA induces 11340x more bitflips than RowHammer on average in the presence of TRR

Mitigating PuDHammer

1 Analyze the effectiveness of existing in-DRAM RowHammer mitigation against PuDHammer

2 Adapt RowHammer mitigations to account for PuDHammer

Adapting PRAC for PuDHammer

A Naïve Approach (PRAC-Naïve)

Reducing RowHammer threshold to SiMRA's lowest observed HCfirst

Weighted Counting (PRAC-WC)

Increase counters based on the vulnerability caused by the operation (RH, CoMRA, and SiMRA)

Methodology

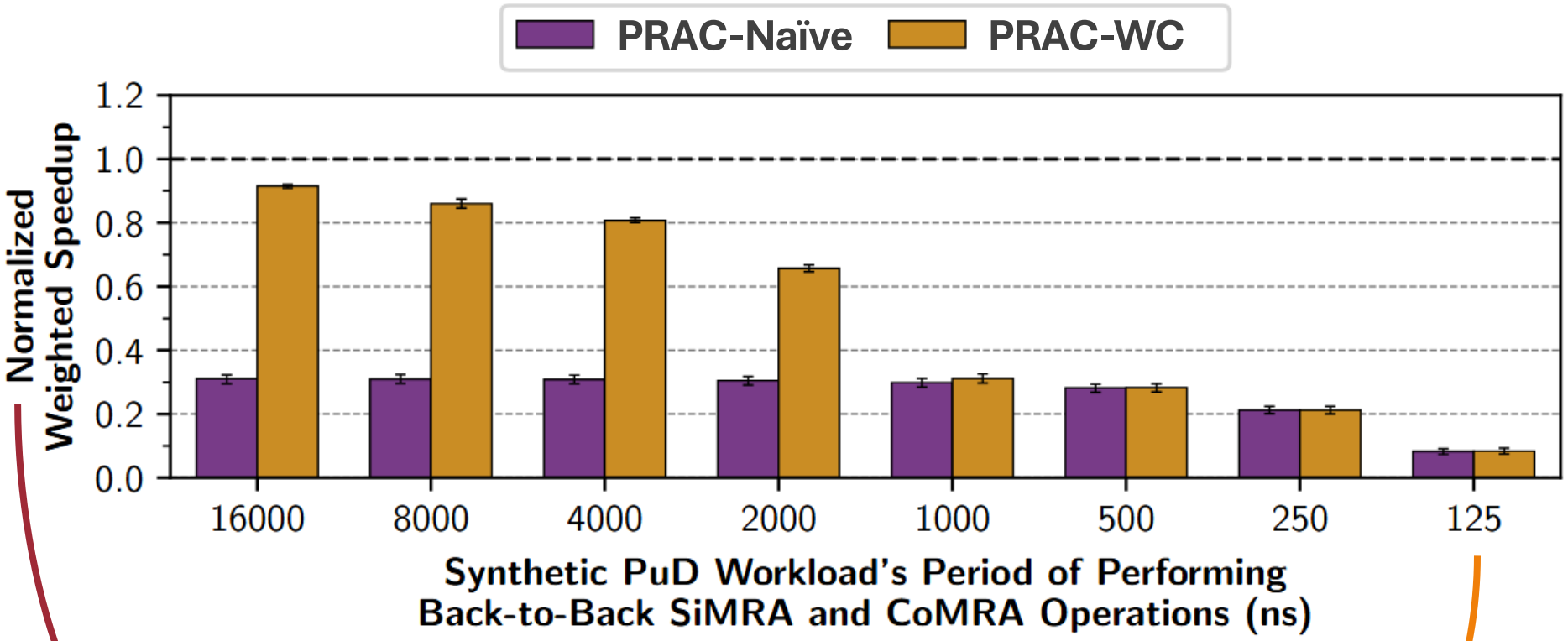
- **Environment:** Cycle-level DRAM simulator Ramulator2 [Kim+ CAL'15, Luo+ CAL'23]

- **System Configuration:**

Processor	4-core, out-of-order, 4.2GHz clock frequency
DRAM	DDR5, 1 channel, 2 rank/channel, 8 bank groups, 4 banks/bank group, 128K rows/bank
Memory Ctrl.	64-entry read and write requests queues, Scheduling policy: FR-FCFS with a column cap of 4
PRAC	4 RFMs per Back-off
PIM Unit	1-core, issues CoMRA & SiMRA operations with a fixed invertal

- **Workloads:**
  - 60 five-core multiprogrammed workload mixes.
  - Each mix is composed of
    - four workloads from five major benchmark suites and
    - one synthetic workload that periodically performs back-to-back one SiMRA with 32-row activation and one CoMRA operation

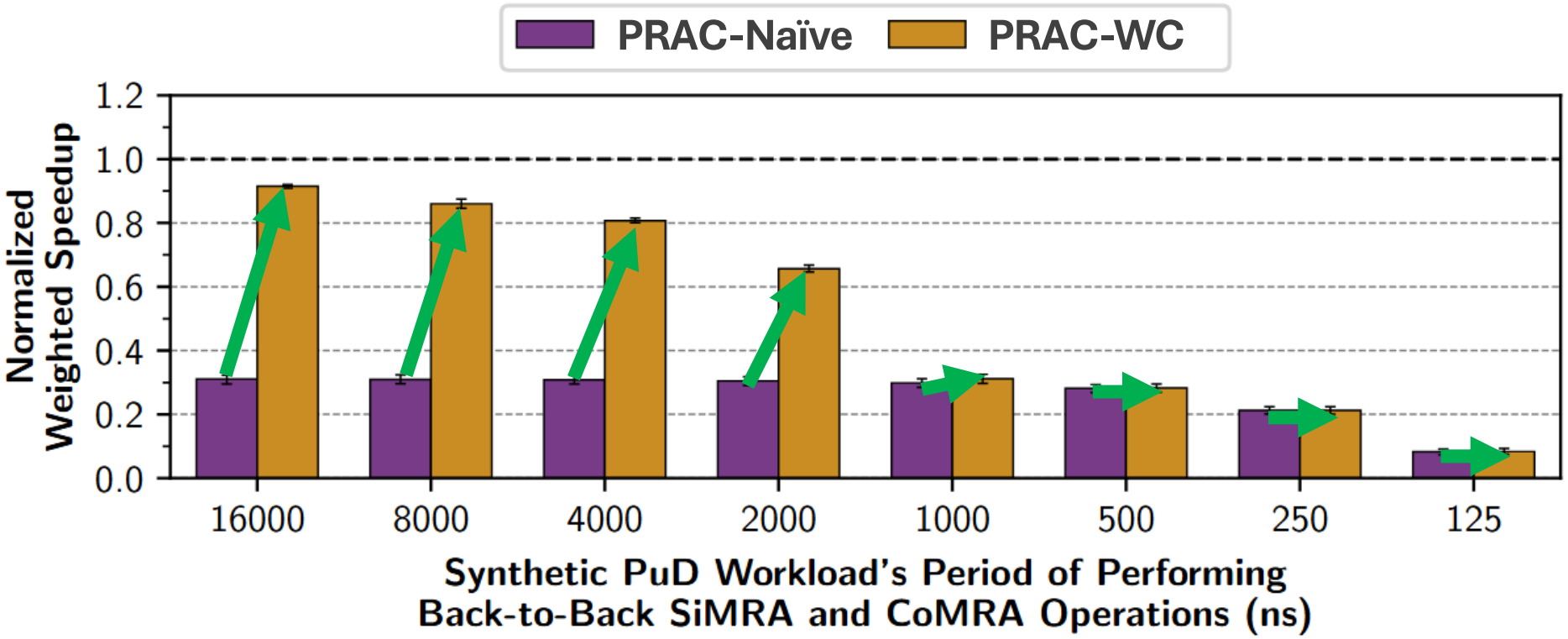
Weighted Speedup Results



Normalized to no RH mitigation

DRAM performs PuD operations in every 125ns

Weighted Speedup Results



PRAC-WC greatly outperforms PRAC-Naïve until 2us

PRAC-WC incurs an average (maximum) performance overhead of 48.26% (98.83%)

More in the Paper

- **More characterization results**
  - Temperature, spatial variation, timing parameters, access pattern, data pattern, aggressor row on time analyses for CoMRA and SiMRA
  - Combined Read Disturbance Effect of RowHammer with PuD
    - Combined pattern is more effective than RowHammer alone (e.g., 1.66x reduction in HCfirst on average)

- **Details and analyses on mitigating PuDHammer**
  - Discussion on how modifying DRAM chips could help mitigating PuDHammer
  - More details on PRAC adaptation
    - Challenges on adapting for PuDHammer
    - An area-efficient PRAC solution
    - ..

Available on arXiv

PuDHammer: Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips

İsmail Emir Yüksel Akash Sood Ataberk Olgun Oğuzhan Canpolat Haocong Luo
F. Nisa Bostancı Mohammad Sadrosadati A. Giray Yağlıkçı Onur Mutlu

ETH Zürich

Processing-using-DRAM (PuD) is a promising paradigm for alleviating the data movement bottleneck using a DRAM array's massive internal parallelism and bandwidth to execute very wide data-parallel operations. Performing a PuD operation involves activating multiple DRAM rows in quick succession or simultaneously, i.e., multiple-row activation. Multiple-row activation is fundamentally different from conventional memory access patterns that activate one DRAM row at a time. However, repeatedly activating even one *DRAM row (e.g., RowHammer) can induce bitflips in unaccessed DRAM rows because modern DRAM is subject to read disturbance, a worsening safety, security, and reliability issue. Unfortunately, no prior work investigates the effects of multiple-row activation, as commonly used by PuD operations, on DRAM read disturbance.*

In this paper, we present the first characterization study of read disturbance effects of multiple-row activation-based PuD (which we call PuDHammer) using 316 real DDR4 DRAM chips from four major DRAM manufacturers. Our detailed characterization results covering various operational conditions and parameters (i.e., temperature, data patterns, access patterns, timing parameters,

CPU and GPU) [1, 2]. Unfortunately, this data movement is a major bottleneck that consumes a large fraction of execution time and energy in many modern applications [1–28]. Processing-using-DRAM (PuD) [29–34] is a promising paradigm that can alleviate the data movement bottleneck. PuD uses the analog operational properties of the DRAM array circuitry to enable massively parallel in-DRAM computation (i.e., PuD operations), which can be used to accelerate important applications including databases and web search [29, 30, 32, 35–43], data analytics [29, 44–48], graph processing [32, 48–51], genome analysis [52–57], cryptography [58, 59], hyper-dimensional computing [60–62], and generative AI [63–72].

A wide variety of PuD operations (e.g., in-DRAM data copy and bulk bitwise operations) rely on a key PuD technique called *multiple-row activation*, which accesses (activates) multiple DRAM rows in quick succession or simultaneously [29–32, 40, 73–84]. Multiple-row activation is fundamentally different from conventional DRAM operations that access only a *single* DRAM row at a time.

https://arxiv.org/pdf/2506.12947

PuDHammer

Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips

İsmail Emir Yüksel

Akash Sood Ataberk Olgun Oğuzhan Canpolat Haocong Luo

Nisa Bostancı Mohammad Sadrosadati Giray Yağlıkçı Onur Mutlu

SAFARI

ETH zürich

7th Workshop on Memory-Centric Computing

Processing-Using-Memory in Real DRAM Chips

İsmail Emir Yüksel

ICS 2026

6 July 2026

SAFARI

ETH zürich