Infrastructure for Processing-Using-Memory Research

Ataberk Olgun

PPoPP 2025

1st March 2025

SAFARI

ETH zürich

Brief Self Introduction

- Ataberk Olgun
  - 4th year PhD Student @ SAFARI Research Group
    - ETH Zurich, Switzerland (Jan 2022 – ongoing)
    - BSc + MSc, TOBB ETU, Turkey
  - https://ataberkolgun.com
  - olgunataberk@gmail.com
  - https://safari.ethz.ch

- Research interests:
  - Computer architecture
  - Memory systems:
    - Security, safety, reliability, availability, performance, energy efficiency
  - Processing in memory

Papers We Will Discuss

- 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]

- Ataberk Olgun, Hasan Hassan, A Giray Yağlıkçı, Yahya Can Tuğrul, Lois Orosa,
  Haocong Luo, Minesh Patel, Oğuz Ergin, and Onur Mutlu,
  "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips"
  IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (***TCAD***), 2023.
  [Extended arXiv version]
  [DRAM Bender Source Code]
  [DRAM Bender Tutorial Video (43 minutes)]

- Ismail Emir Yuksel, Yahya Can Tugrul, Ataberk Olgun, F. Nisa Bostanci, A. Giray Yaglikci, Geraldo F. Oliveira, Haocong Luo,
  Juan Gomez-Luna, Mohammad Sadrosadati, and Onur Mutlu,
  "Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis"
  Proceedings of the 30th International Symposium on High-Performance Computer Architecture (***HPCA***), April 2024.
  [Slides (pptx) (pdf)]
  [arXiv version]
  [FCDRAM Source Code]

- Ismail Emir Yuksel, Yahya Can Tugrul, F. Nisa Bostanci, Geraldo F. Oliveira, A. Giray Yaglikci, Ataberk Olgun, Melina Soysal,
  Haocong Luo, Juan Gomez-Luna, Mohammad Sadrosadati, and Onur Mutlu,
  "Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis"
  Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (***DSN***), Brisbane, Australia, June 2024.
  [Slides (pptx) (pdf)]
  [arXiv version]
  [SiMRA-DRAM Source Code (Officially Artifact Evaluated with All Badges)]
  Officially artifact evaluated as both code and dataset available, reviewed and reproducible.

Recall: Processing using DRAM

- We can support
  - Bulk bitwise AND, OR, NOT, MAJ
  - Bulk bitwise COPY and INIT/ZERO
  - True Random Number Generation; Physical Unclonable Functions
  - Lookup Table based more complex computation
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating (multiple) rows performs computation
    - Even in commodity off-the-shelf DRAM chips!

PiDRAM

An FPGA-based Framework for End-to-end Evaluation of Processing-in-DRAM Techniques

Ataberk Olgun

Juan Gomez Luna Konstantinos Kanellopoulos Behzad Salami

Hasan Hassan Oğuz Ergin Onur Mutlu

PiDRAM Summary

Motivation: Commodity DRAM based PiM techniques improve the performance and energy efficiency of computing systems at no additional DRAM hardware cost

Problem: Challenges of integrating these PiM techniques into real systems are not solved
General-purpose computing systems, special-purpose testing platforms, and system simulators *cannot* be used to efficiently study system integration challenges

Goal: Design and implement a flexible framework that can be used to:

- solve system integration challenges
- analyze trade-offs of end-to-end implementations of commodity DRAM-based-PiM techniques

Key idea: PiDRAM, an FPGA-based framework that enables:

- system integration studies
- end-to-end evaluations of PIM techniques using real unmodified DRAM chips

Evaluation: End-to-end integration of two PiM techniques on PiDRAM's FPGA prototype

Case Study #1 – RowClone: In-DRAM bulk data copy operations

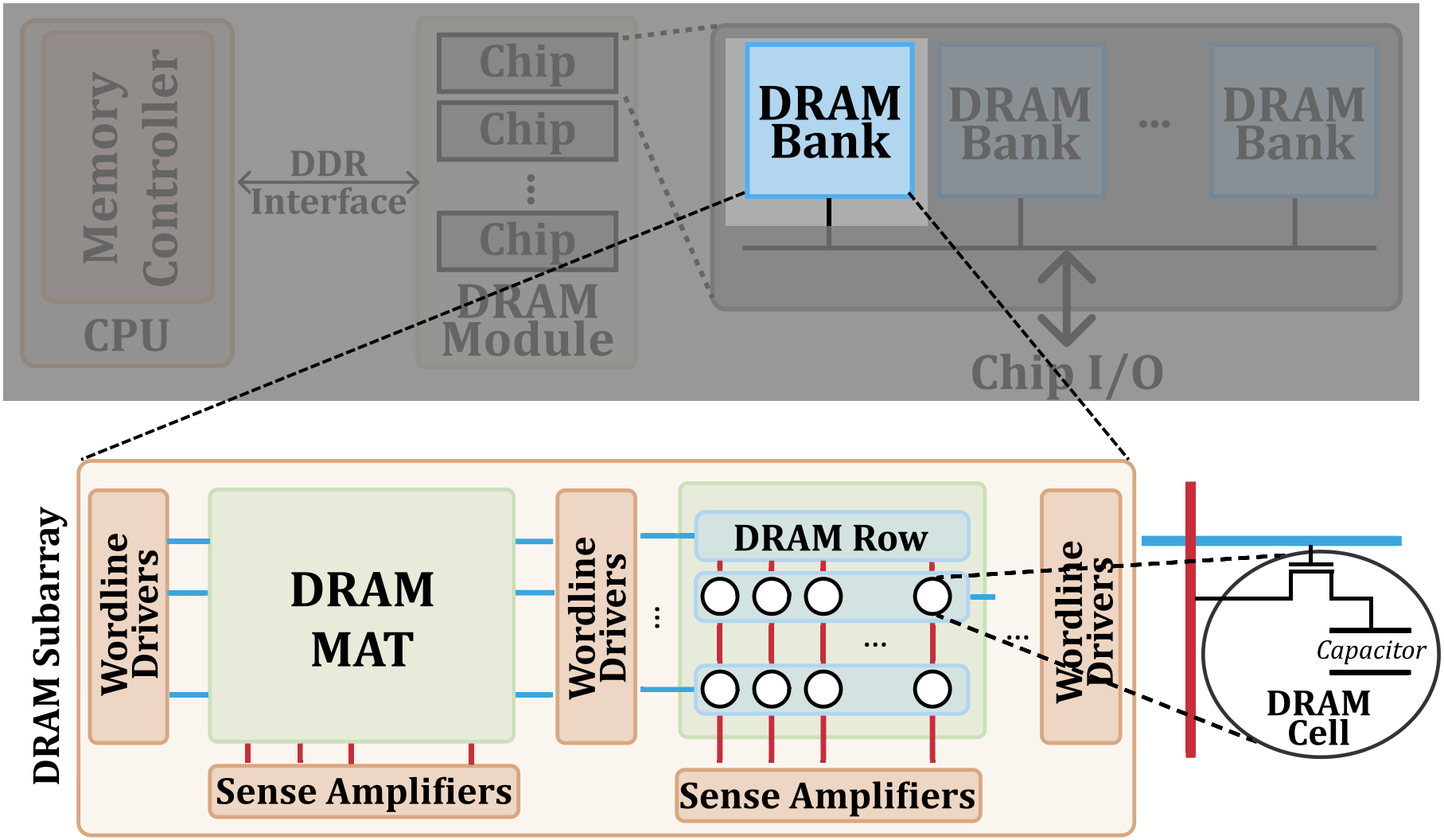
- 119x speedup for copy operations compared to CPU-copy with system support
- 198 lines of Verilog and 565 lines of C++ code over PiDRAM's flexible codebase

Case Study #2 – D-RaNGe: DRAM-based random number generation technique

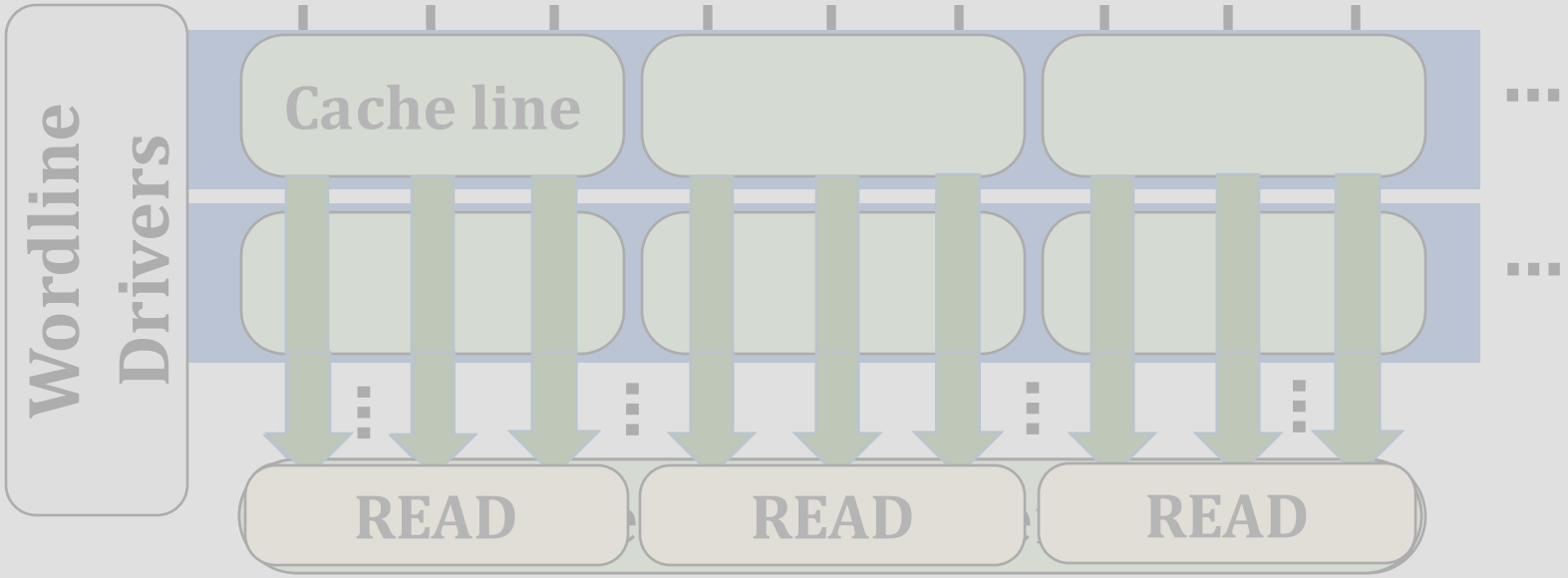
- 8.30 Mb/s true random number generator (TRNG) throughput, 220 ns TRNG latency
- 190 lines of Verilog and 78 lines of C++ code over PiDRAM's flexible codebase

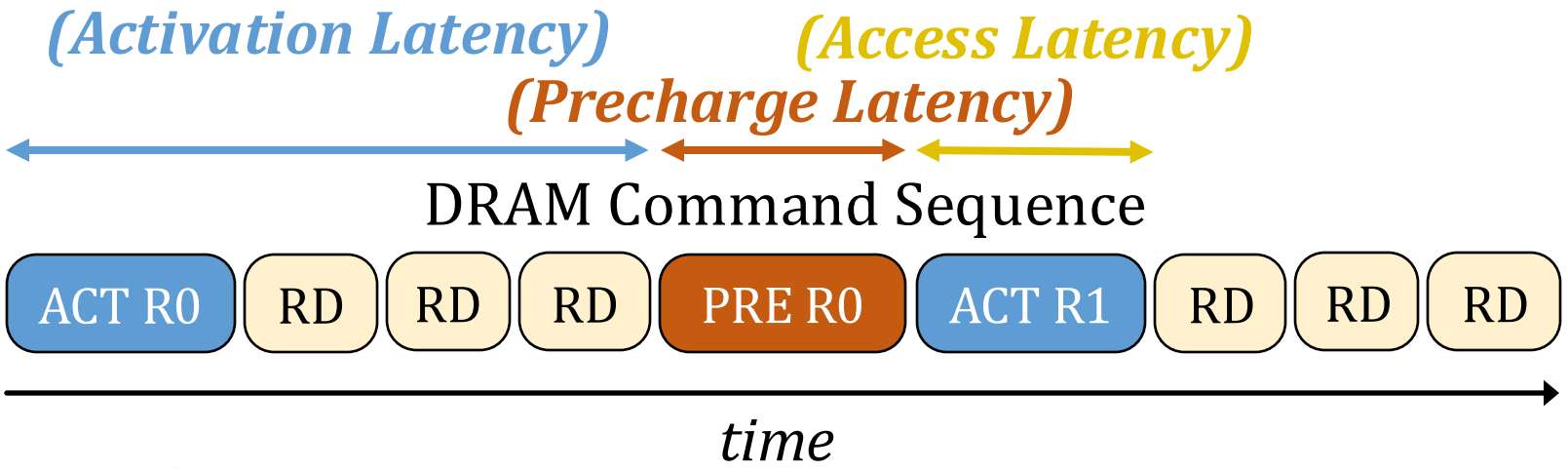
Outline

DRAM Organization



DRAM Operation





Processing-in-Memory Techniques

Use operational principles of memory
to perform bulk data movement and computation

Commodity DRAM chips can already perform:

[Gao+, MICRO'19]-[Gao+, MICRO'22]

1) Row-copy: In-DRAM bulk data copy (or initialization) at DRAM row granularity

(e.g., [Kim+, HPCA'19]-[Olgun+, ISCA'21])

2) True random number generation

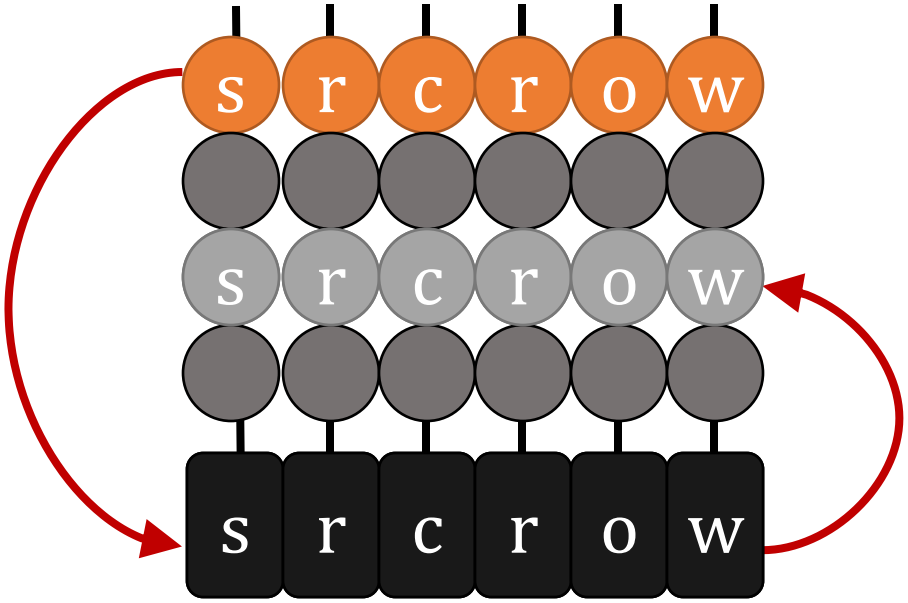
(e.g., [Kim+, HPCA'18])

3) Physical uncloneable functions

[Gao+, MICRO'19]-[Gao+, MICRO'22]

4) Majority operation

Row-Copy: Key Idea (RowClone)



Sense Amplifiers

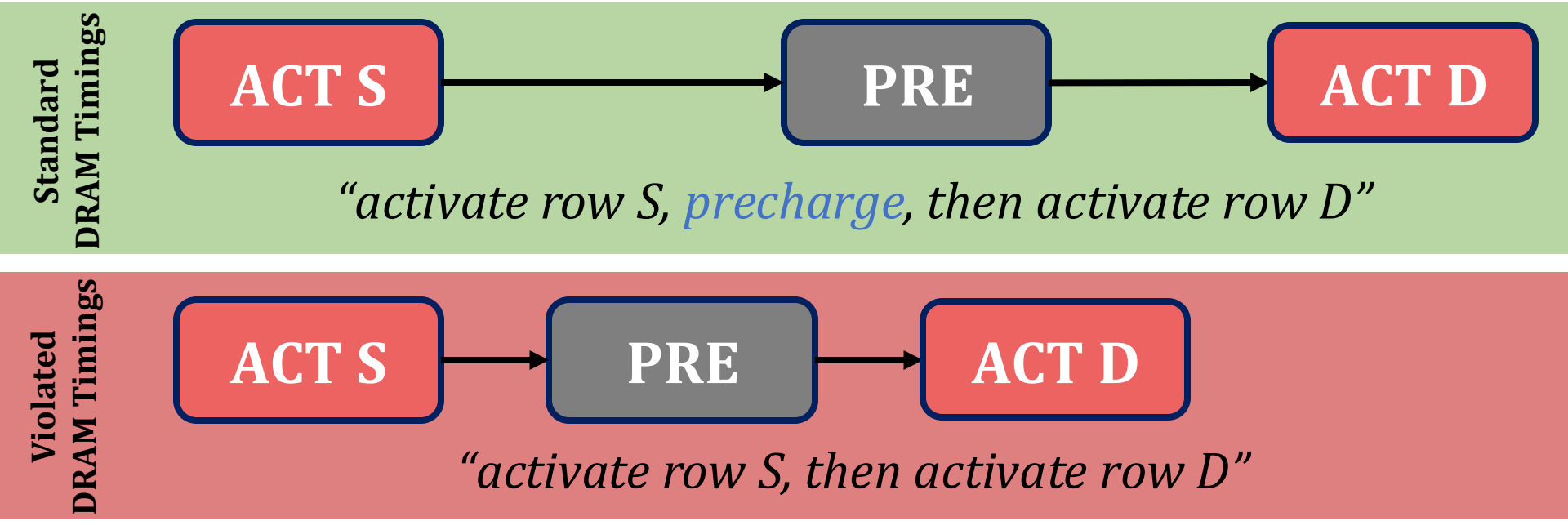
✓ **1. Source row to sense amplifiers**

? **2. Sense amplifiers to destination row**

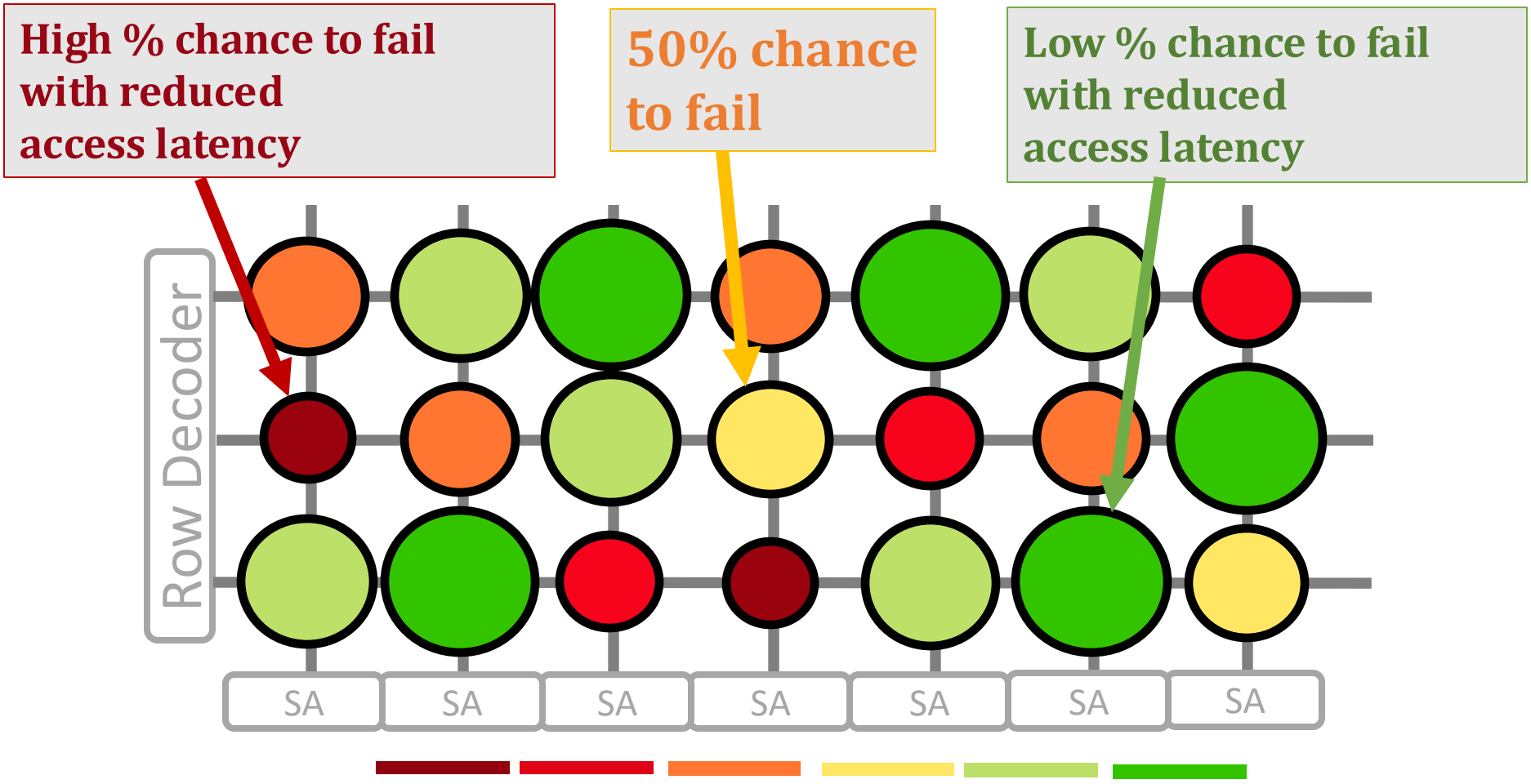
RowClone in Real DRAM Chips

Key Idea: Use carefully created DRAM command sequences

- ACT → PRE → ACT command sequence with greatly reduced DRAM timing parameters
- ComputeDRAM **[Gao+, MICRO'19]** demonstrates in-DRAM copy operations in real DDR3 chips

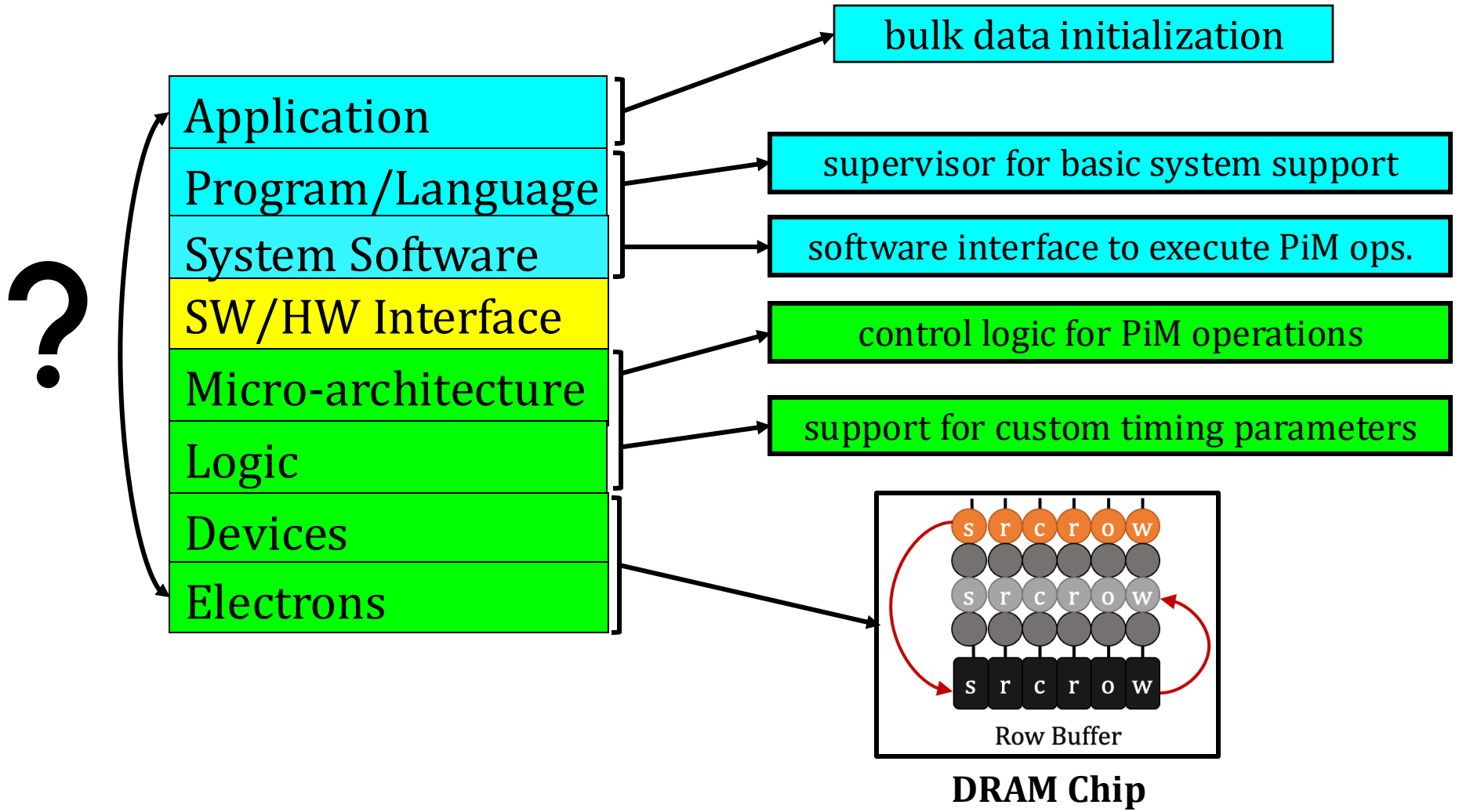


In-DRAM TRNG: Key Idea (D-RaNGe)



Commodity DRAM chips can *already* perform D-RaNGe

System Support for PiM



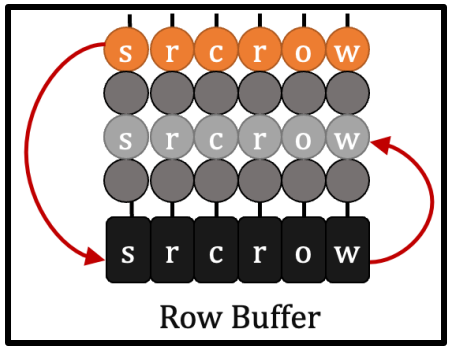
PiDRAM

Bridge the "system gap" with customizable HW/SW components

in doing so, allow users to

rapidly implement PiM techniques, solve system integration challenges, analyze end-to-end implementations

- bulk data initialization
- supervisor for basic system support
- software interface to execute PiM ops.
- control logic for PiM operations
- support for custom timing parameters

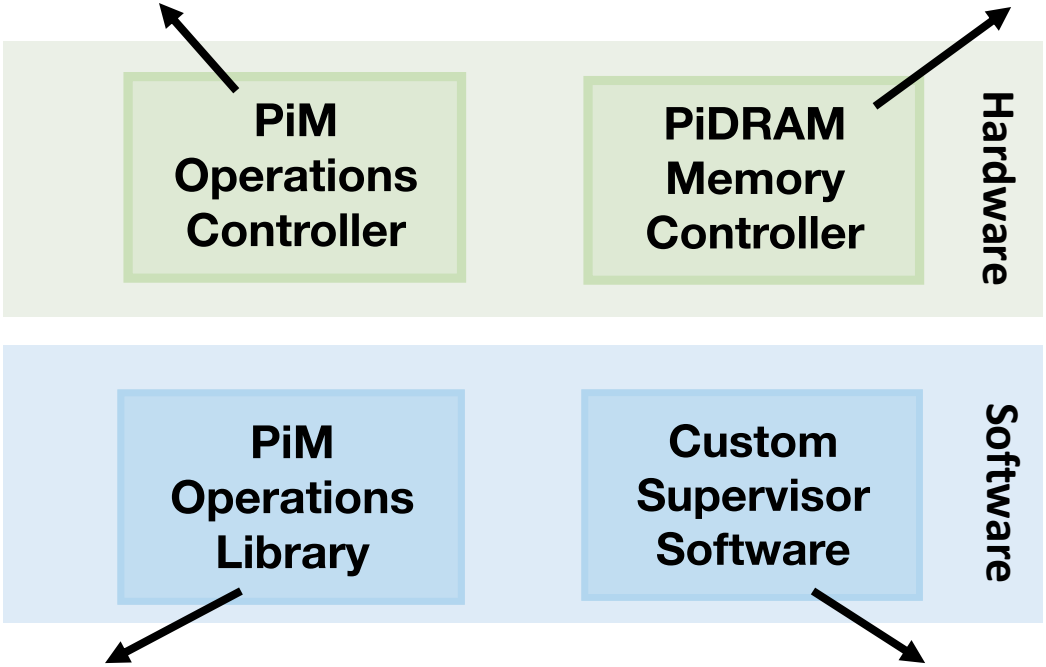


DRAM Chip

PiDRAM: Key Components

Control PiM operation

Memory controller for custom timing parameters

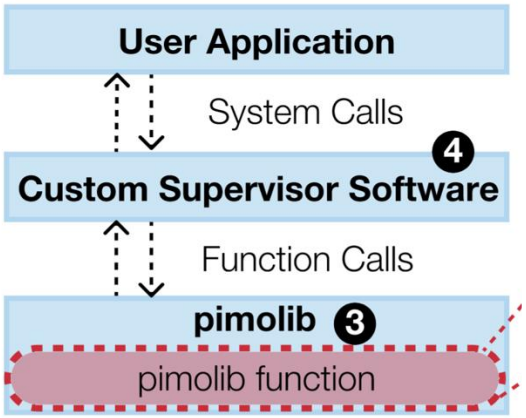


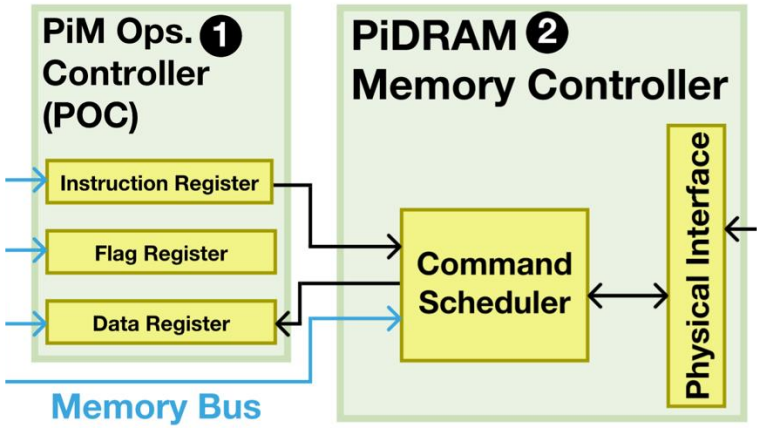
Interface for Applications

Supervisor software for basic system support

PiDRAM: System Design

Key components attached to a real computing system

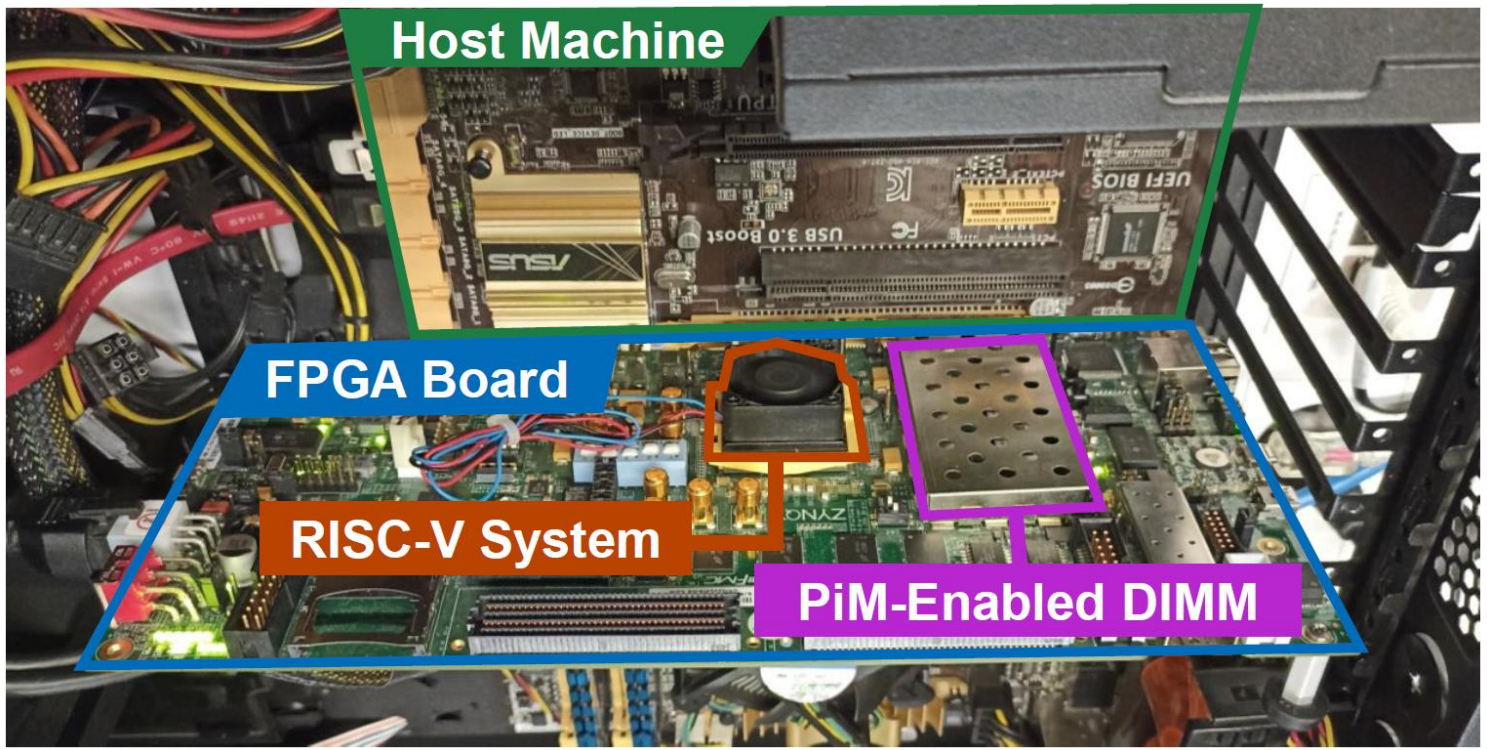




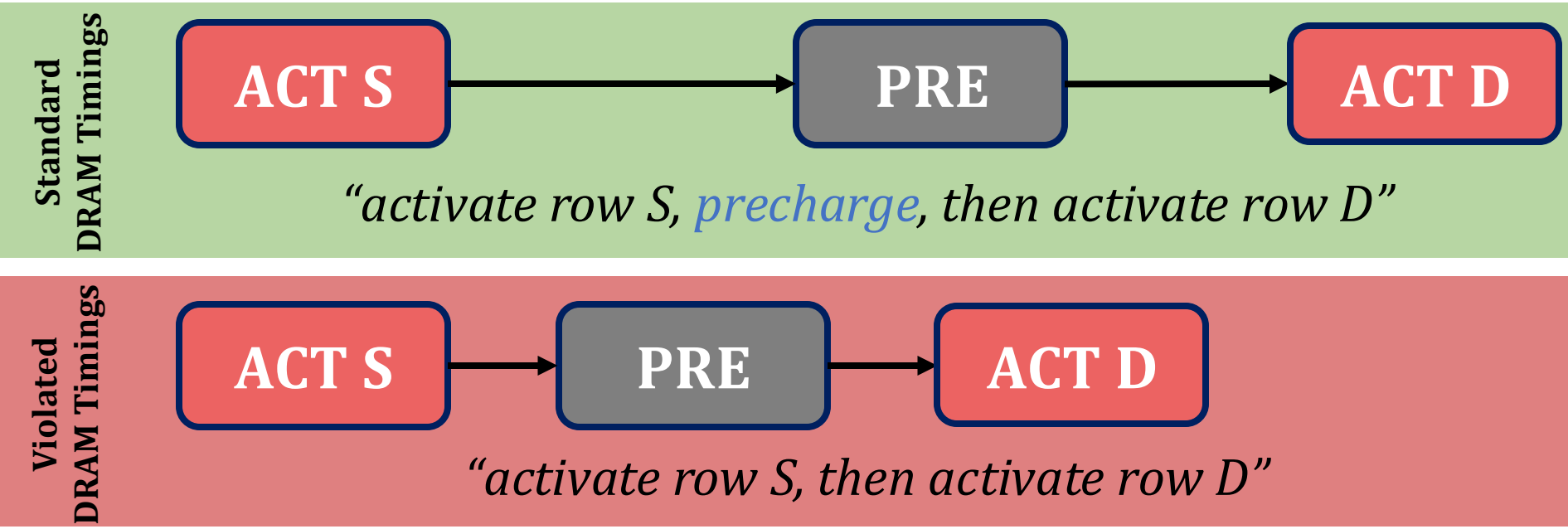
PiDRAM's FPGA Prototype

Full system prototype on Xilinx ZC706 FPGA board

- **RISC-V System:** In-order, pipelined RISC-V Rocket CPU core, L1D/I$, TLB
- **PiM-Enabled DIMM (Commodity):** Micron MT8JTF12864, 1 GiB, 8 banks



RowClone Implementation



1. Extend the PiDRAM memory controller to support the DRAM command sequence
2. Expose the operation to pimolib by implementing the `copy()` PiDRAM instruction

Only 198 lines of Verilog code

RowClone System Integration

Identify two challenges in end-to-end RowClone

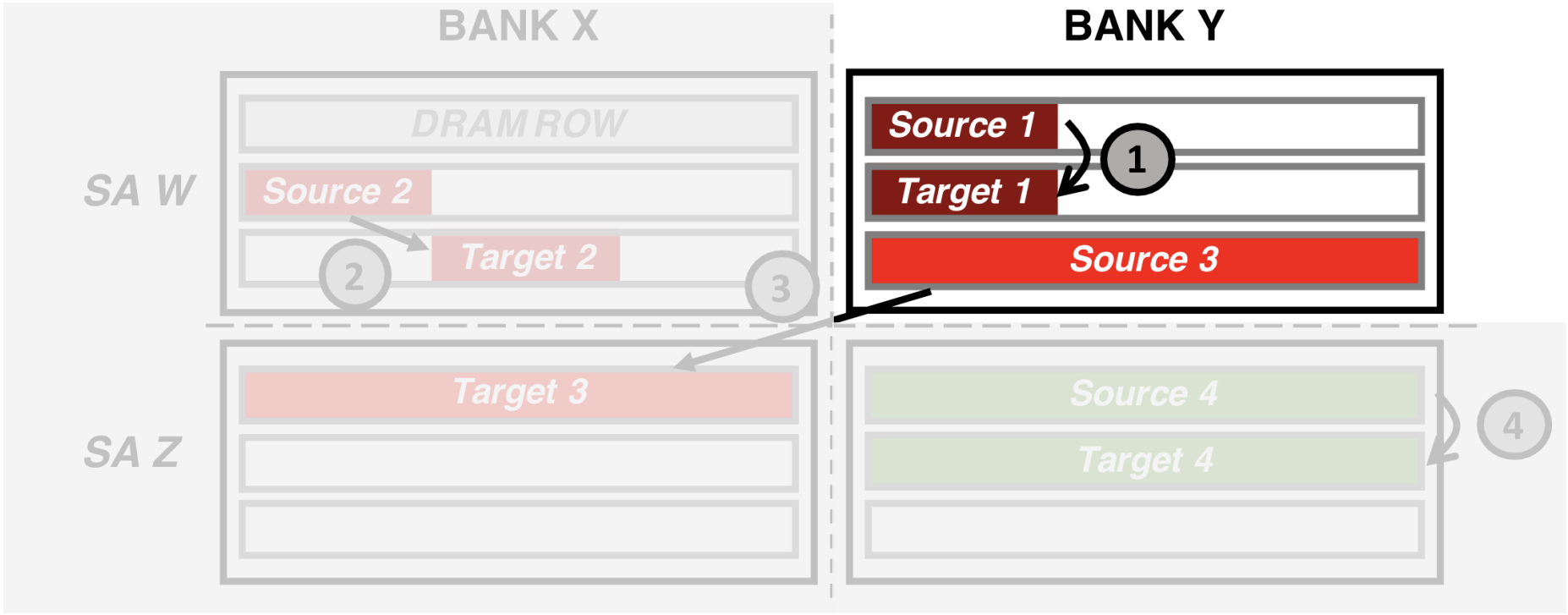
1. Memory allocation (intra-subarray operation)

2. Memory coherency (computation in DRAM)

Implement CLFLUSH instruction in the RISC-V CPU
Evict a cache block from the CPU caches to the DRAM module

RowClone Memory Allocation (I)

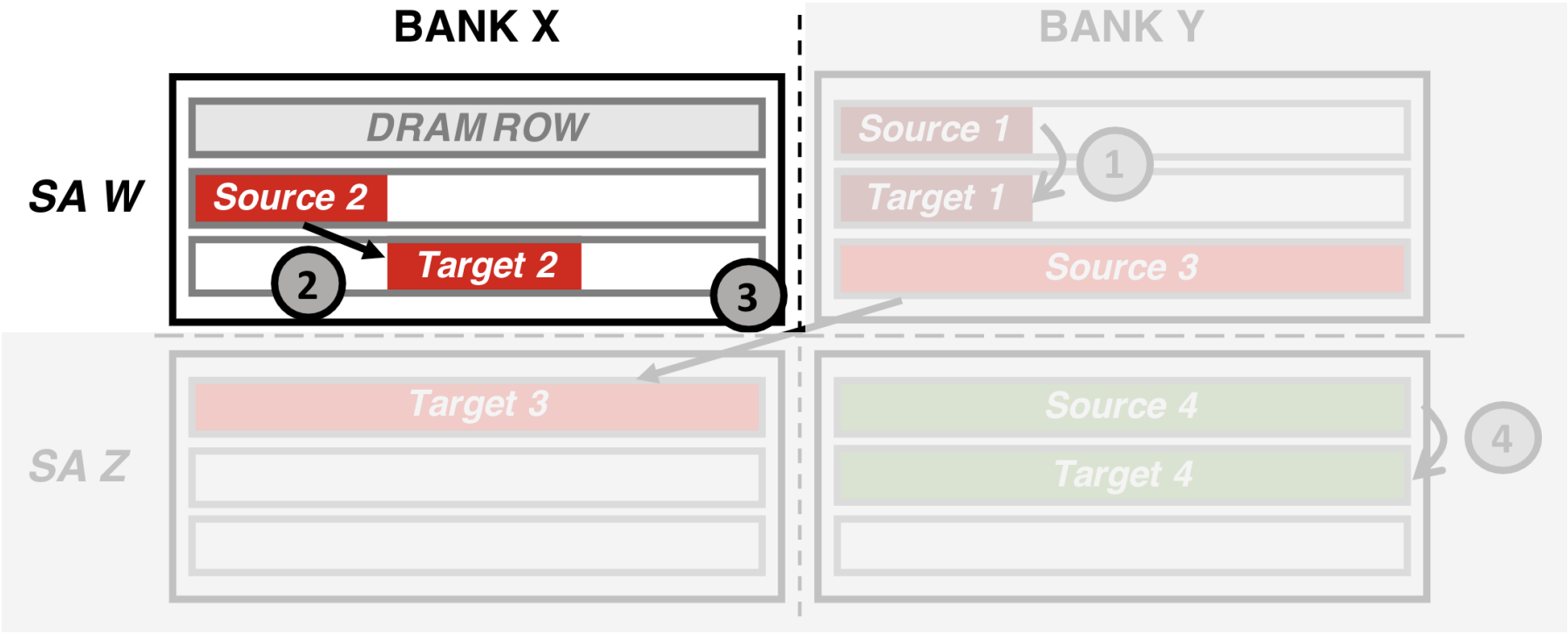
Memory allocation requirements



1 **Granularity:** Operands must occupy DRAM rows fully

RowClone Memory Allocation (I)

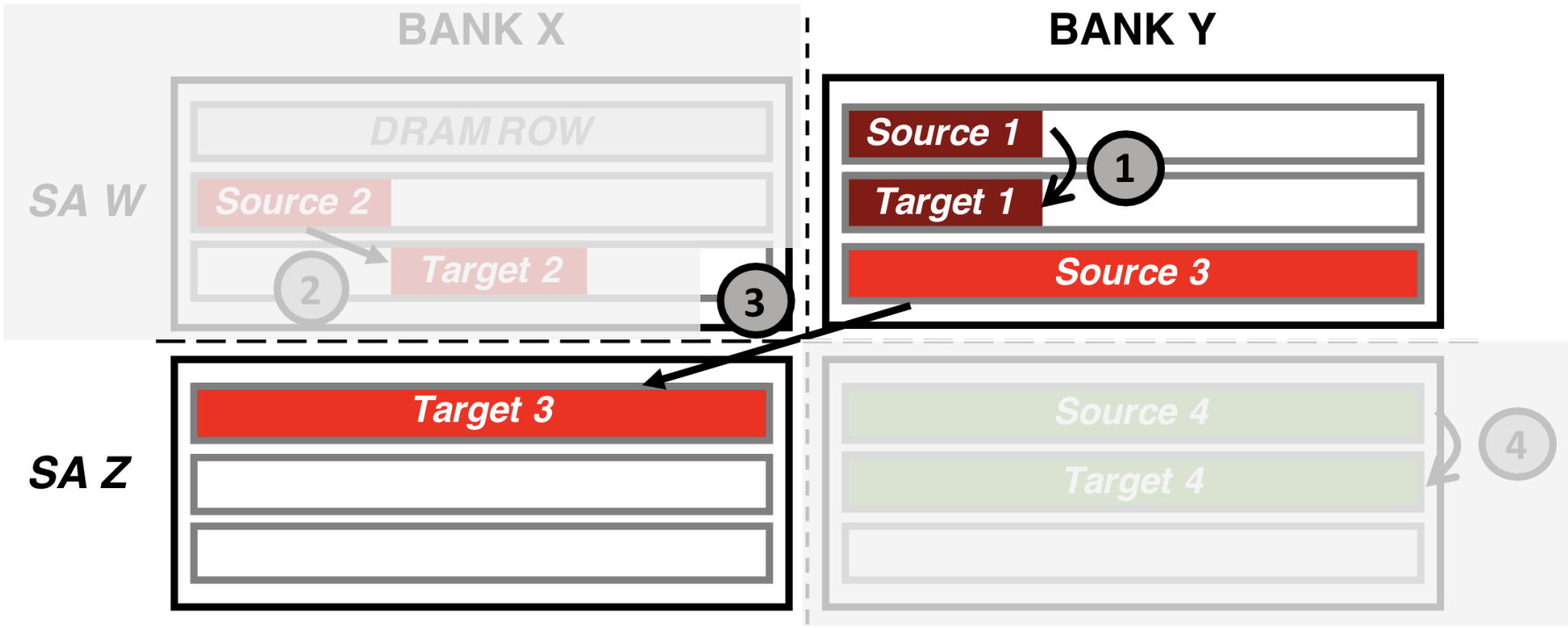
Memory allocation requirements



2 **Alignment:** Operands must be placed at the same offset

RowClone Memory Allocation (I)

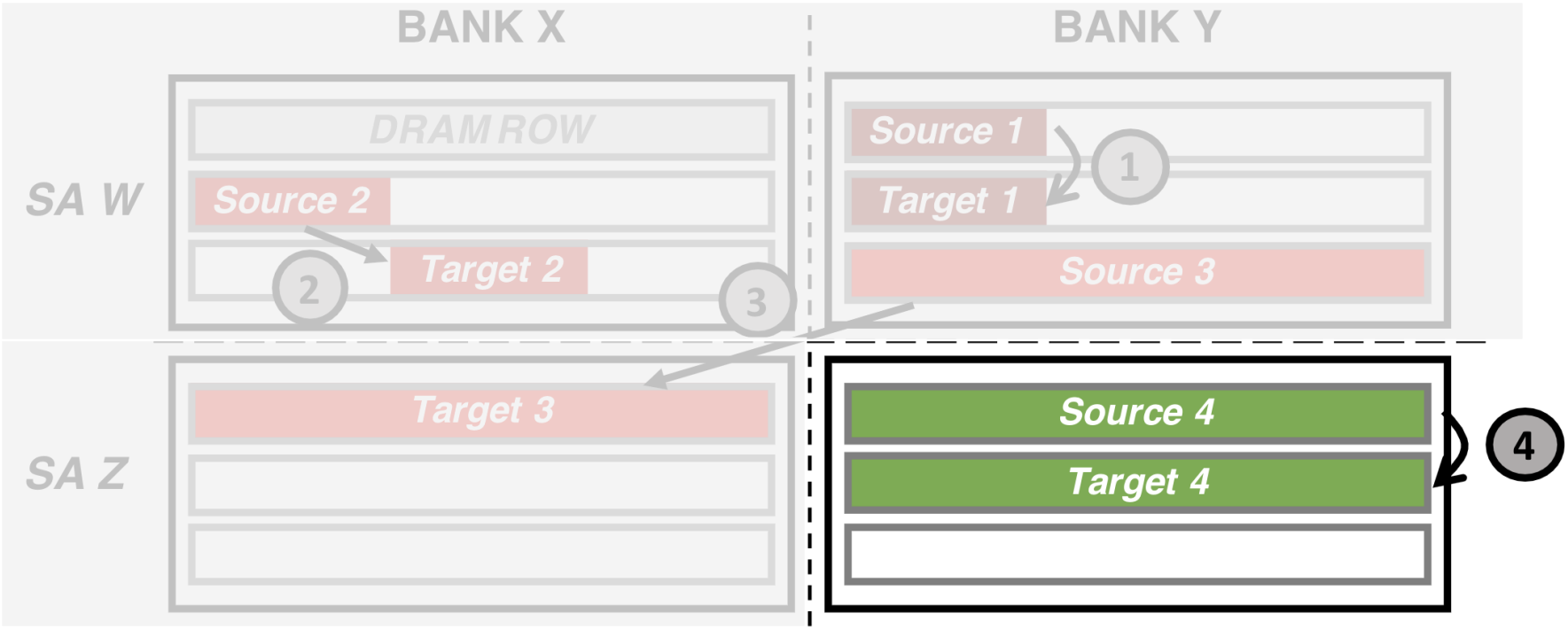
Memory allocation requirements



3 **Mapping:** Operands must be placed in the same subarray

RowClone Memory Allocation (I)

Memory allocation requirements



4 **Satisfies all three requirements**

RowClone Memory Allocation (II)

Implement a new memory allocation function
to overcome the memory allocation challenges

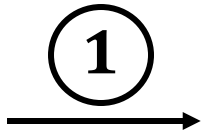
Goal: Allocate virtual memory pages that are
mapped to the same DRAM subarray and aligned with each other

```
virtual_address = alloc_align(int size, int id)
```

size: # of bytes allocated

id: allocations with the same id go to the same subarray

```
alloc_align(
4 KiB,
"Subarray 0")
```



Subarray Mapping Table

Page Table

1. Get physical address pointing to a DRAM row in subarray 0
2. Update the page table to map virtual address to subarray 0

RowClone Memory Allocation (II)

Implement a new memory allocation function

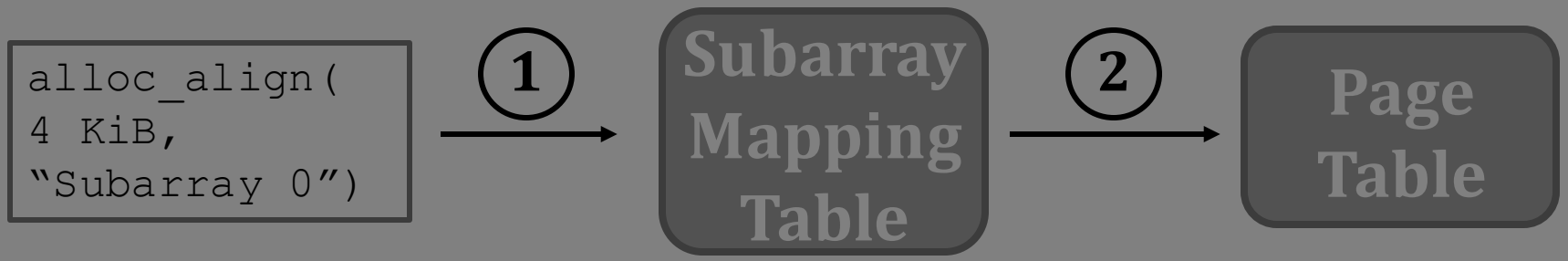
https://arxiv.org/abs/2111.00082

Goal: Allocate virtual memory pages that are mapped to the same DRAM subarray and aligned with each other

```
virtual_address = alloc_align(int size, int id)
```

size: # of bytes allocated

id: allocations with the same id go to the same subarray



1. Get physical address pointing to a DRAM row in subarray 0
2. Update the page table to map virtual address to subarray 0

Evaluation: Methodology

Table 2: PiDRAM system configuration

CPU: 50 MHz; in-order Rocket core [16]; **TLB** 4 entries DTLB; LRU policy
L1 Data Cache: 16 KiB, 4-way; 64 B line; random replacement policy
DRAM Memory: 1 GiB DDR3; 800MT/s; single rank; 8 KiB row size

in-DRAM copy/initialization granularity

Microbenchmarks

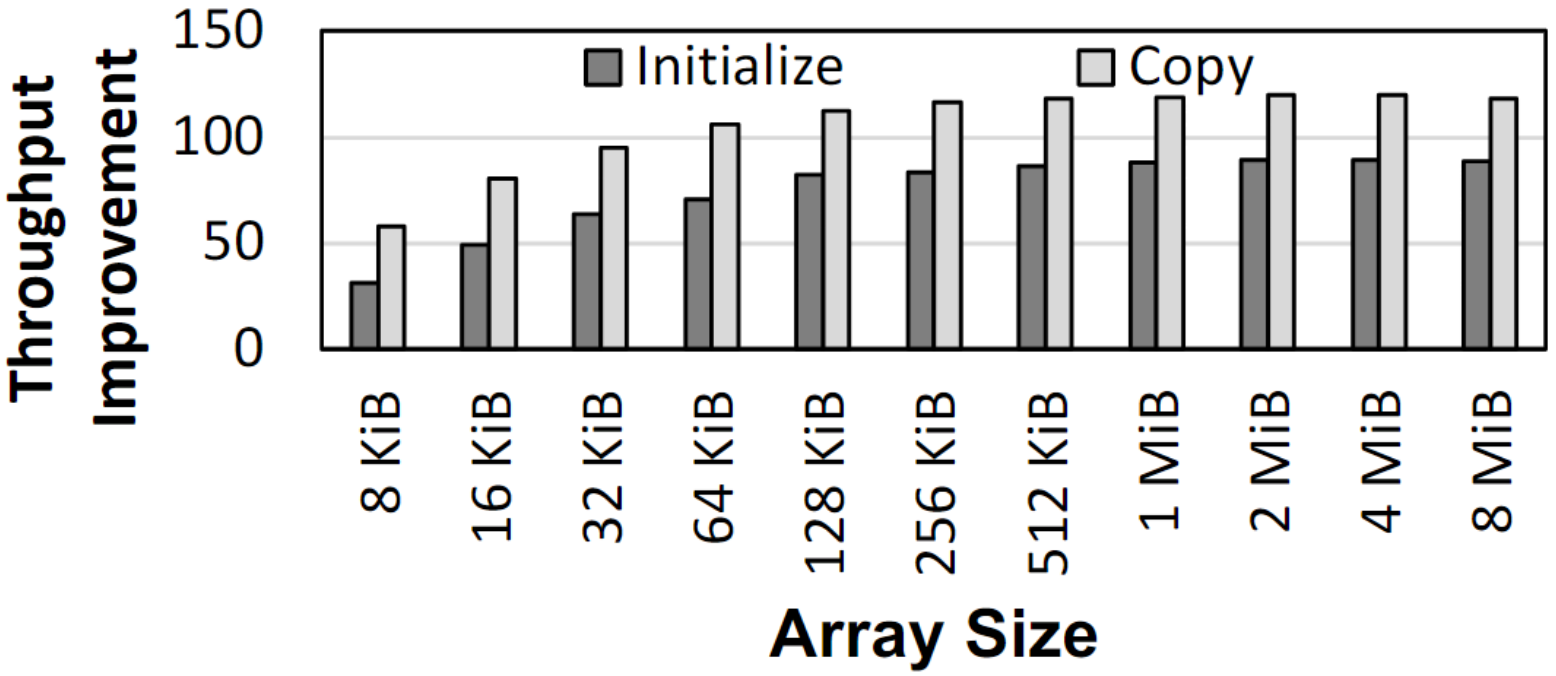
- CPU-Copy (using LOAD/STORE instructions)
- RowClone-Copy (using in-DRAM copy operations) with and without CLFLUSH

Copy/Initialization Heavy Workloads

- forkbench (copy)
- compile (initialization)

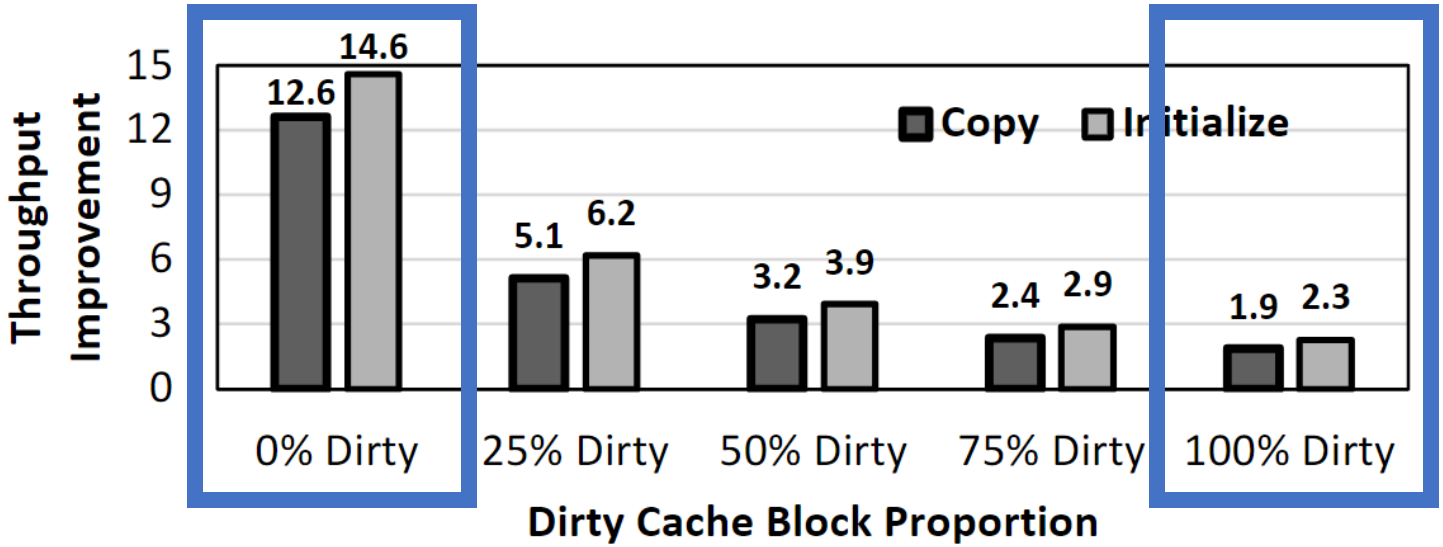
SPEC2006 libquantum: replace "calloc()" with in-DRAM initialization

Microbenchmark Copy/Initialization Throughput Improvement



In-DRAM Copy and Initialization improve throughput by 119x and 89x, respectively

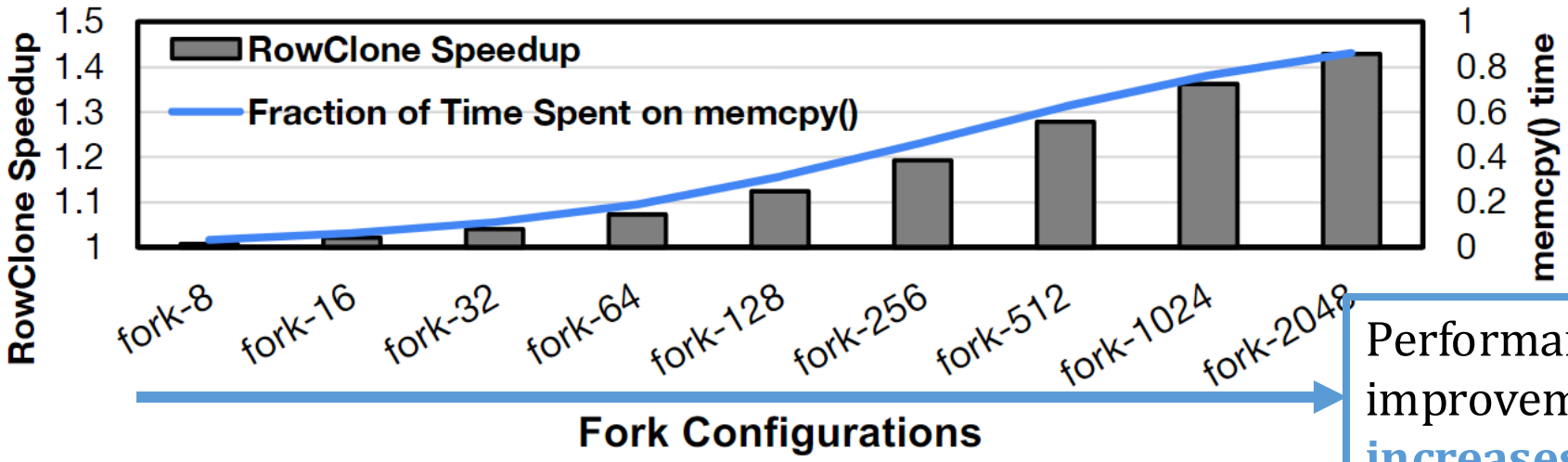
CLFLUSH Overhead



CLFLUSH dramatically reduces the potential throughput improvement

Other Workloads

forkbench (copy-heavy workload)



compile (initialization-heavy workload)

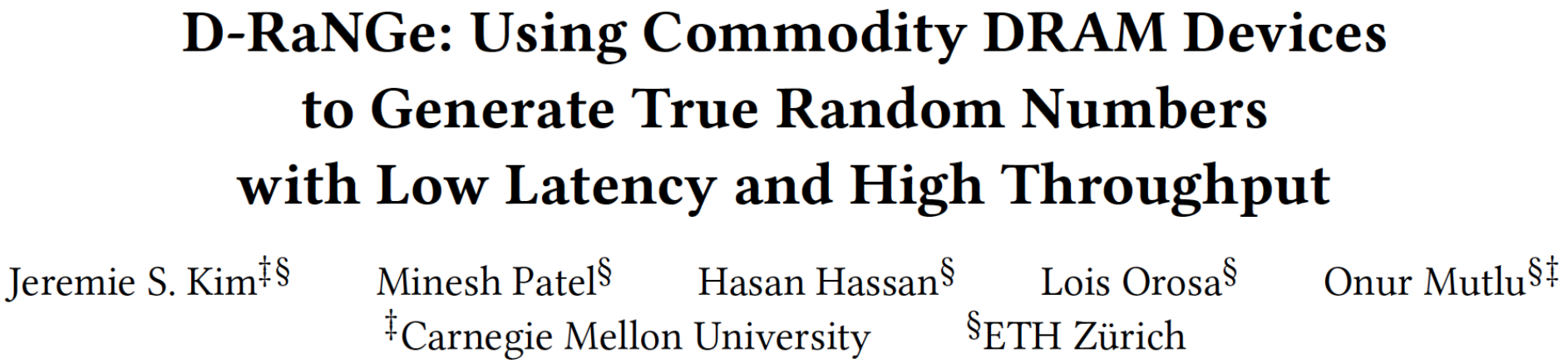
- 9% execution time reduction by in-DRAM initialization
  - 17% of compile's execution time is spent on initialization

SPEC2006 libquantum

- 1.3% end-to-end execution time reduction
  - 2.3% of libquantum's time is spent on initialization

In-DRAM True Random Number Generation

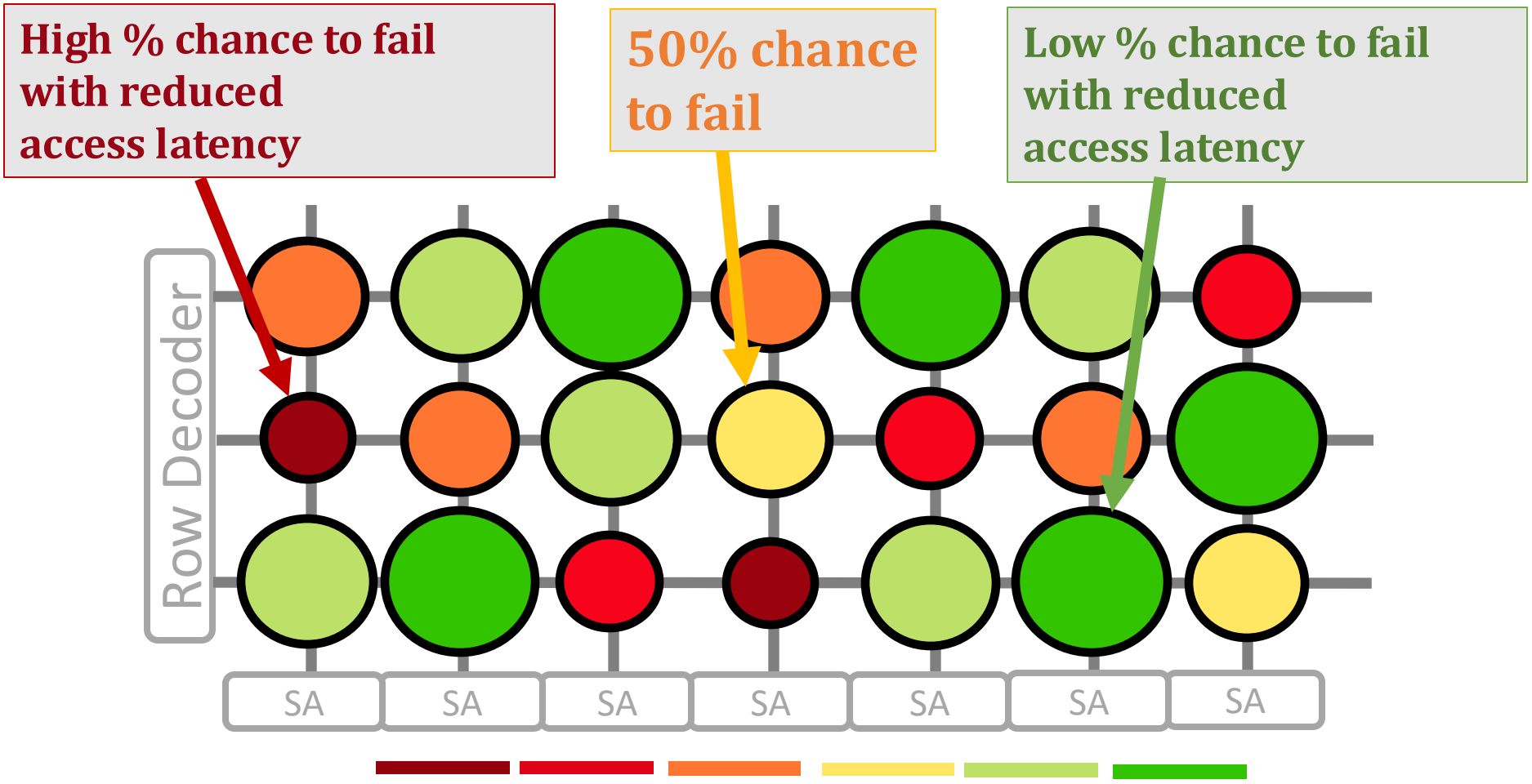
- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu,
  "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"
  *Proceedings of the 25th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Washington, DC, USA, February 2019.
  [Slides (pptx) (pdf)]
  [Full Talk Video (21 minutes)]
  [Full Talk Lecture Video (27 minutes)]
  Top Picks Honorable Mention by IEEE Micro.



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

Jeremie S. Kim[‡§] Minesh Patel[§] Hasan Hassan[§] Lois Orosa[§] Onur Mutlu[§‡]
[‡]Carnegie Mellon University [§]ETH Zürich

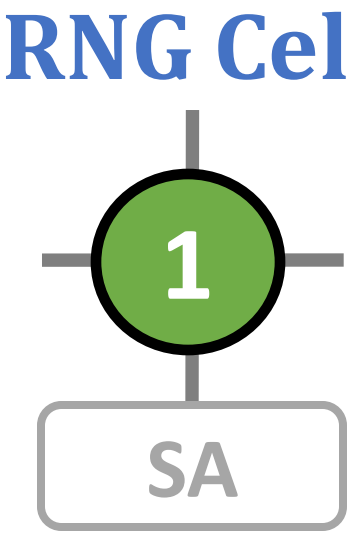
Recall: D-RaNGe Key Idea



Commodity DRAM chips can *already* perform D-RaNGe

D-RaNGe Implementation

Identify four DRAM cells that fail randomly in a cache block



D-RaNGe Implementation

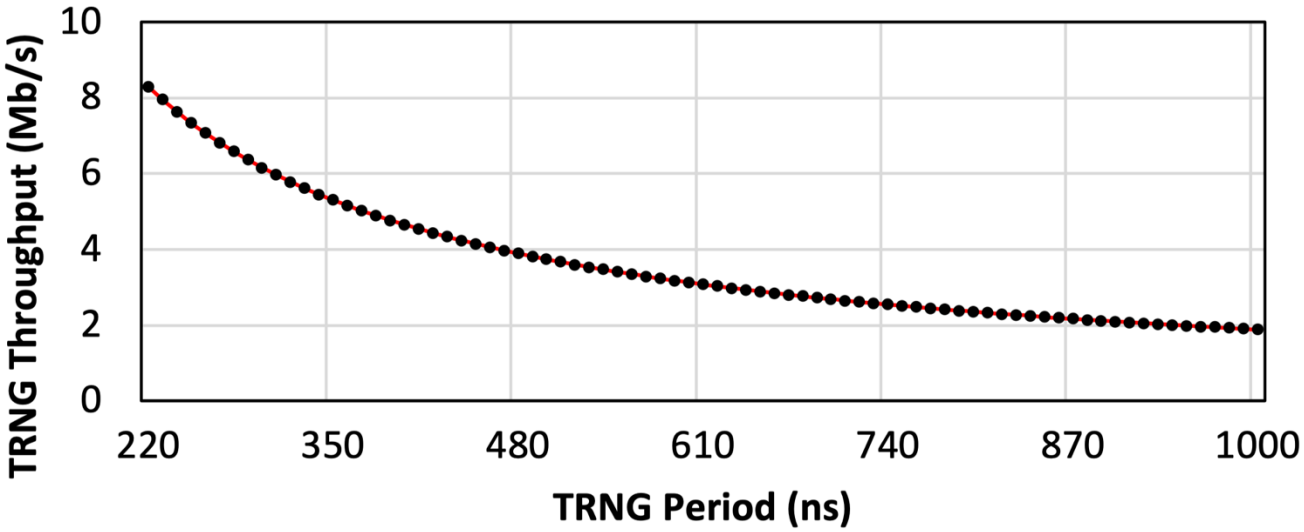
Periodically generate true random numbers by accessing the identified cache block

- Reduce access latency
- 1 KiB random number buffer in POC
- Programmers read random numbers from the *data register* using the `rand_dram()` function call

190 lines of Verilog code
74 lines of C++ code

Evaluation

Methodology: Microbenchmark that reads true random numbers



PiDRAM's D-RaNGe generates true random numbers at 8.30 Mb/s throughput

PiDRAM Summary

Motivation: Commodity DRAM based PiM techniques improve the performance and energy efficiency of computing systems at no additional DRAM hardware cost

Problem: Challenges of integrating these PiM techniques into real systems are not solved
General-purpose computing systems, special-purpose testing platforms, and system simulators *cannot* be used to efficiently study system integration challenges

Goal: Design and implement a flexible framework that can be used to:

- solve system integration challenges
- analyze trade-offs of end-to-end implementations of commodity DRAM-based-PiM techniques

Key idea: PiDRAM, an FPGA-based framework that enables:

- system integration studies
- end-to-end evaluations of PIM techniques using real unmodified DRAM chips

Evaluation: End-to-end integration of two PiM techniques on PiDRAM's FPGA prototype

Case Study #1 – RowClone: In-DRAM bulk data copy operations

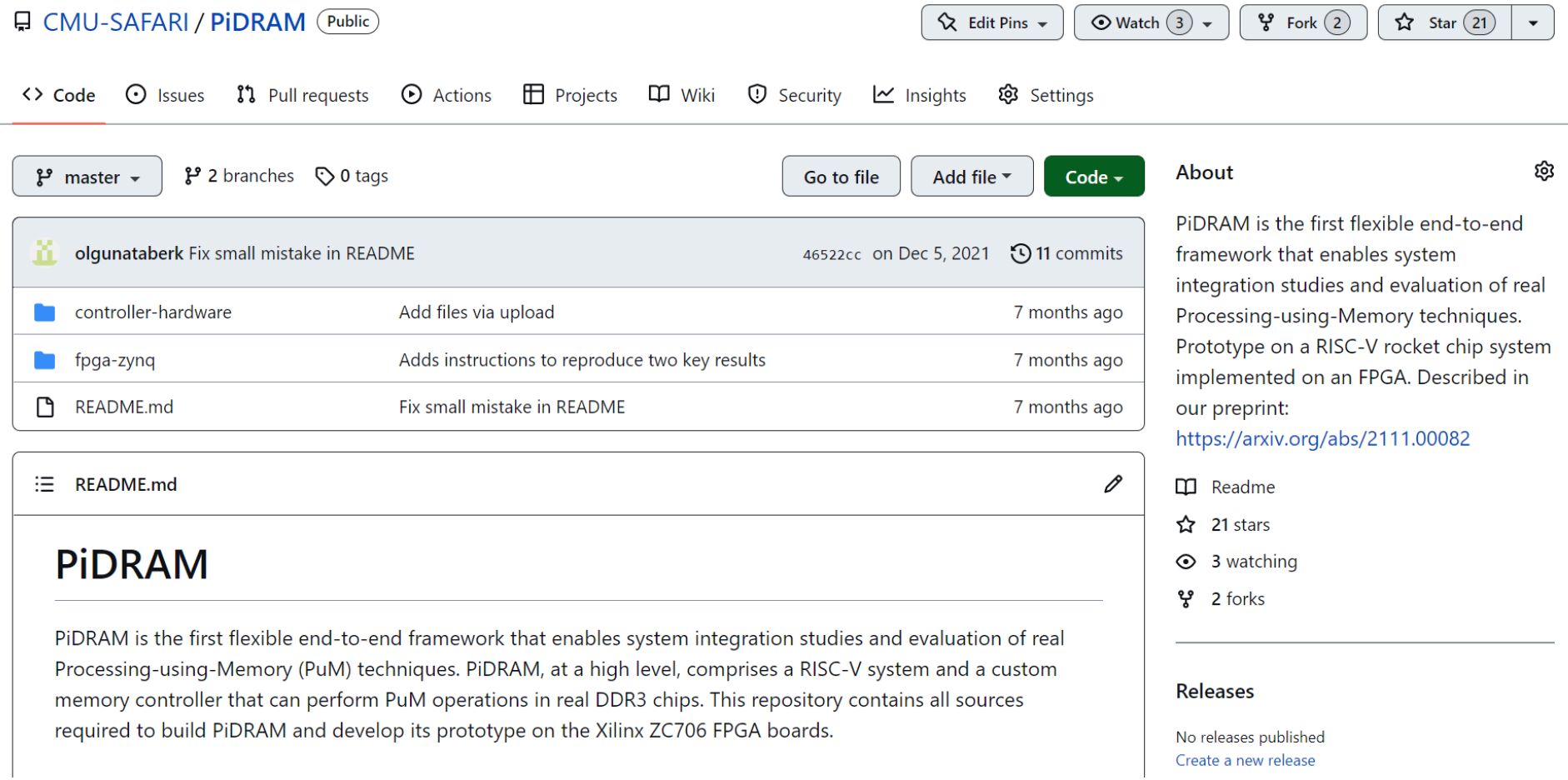
- 119x speedup for copy operations compared to CPU-copy with system support
- 198 lines of Verilog and 565 lines of C++ code over PiDRAM's flexible codebase

Case Study #2 – D-RaNGe: DRAM-based random number generation technique

- 8.30 Mb/s true random number generator (TRNG) throughput, 220 ns TRNG latency
- 190 lines of Verilog and 74 lines of C++ code over PiDRAM's flexible codebase

PiDRAM is Open Source

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



Extended Version on ArXiv

https://arxiv.org/abs/2111.00082

Computer Science > Hardware Architecture

[Submitted on 29 Oct 2021 (v1), last revised 19 Dec 2021 (this version, v3)]

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

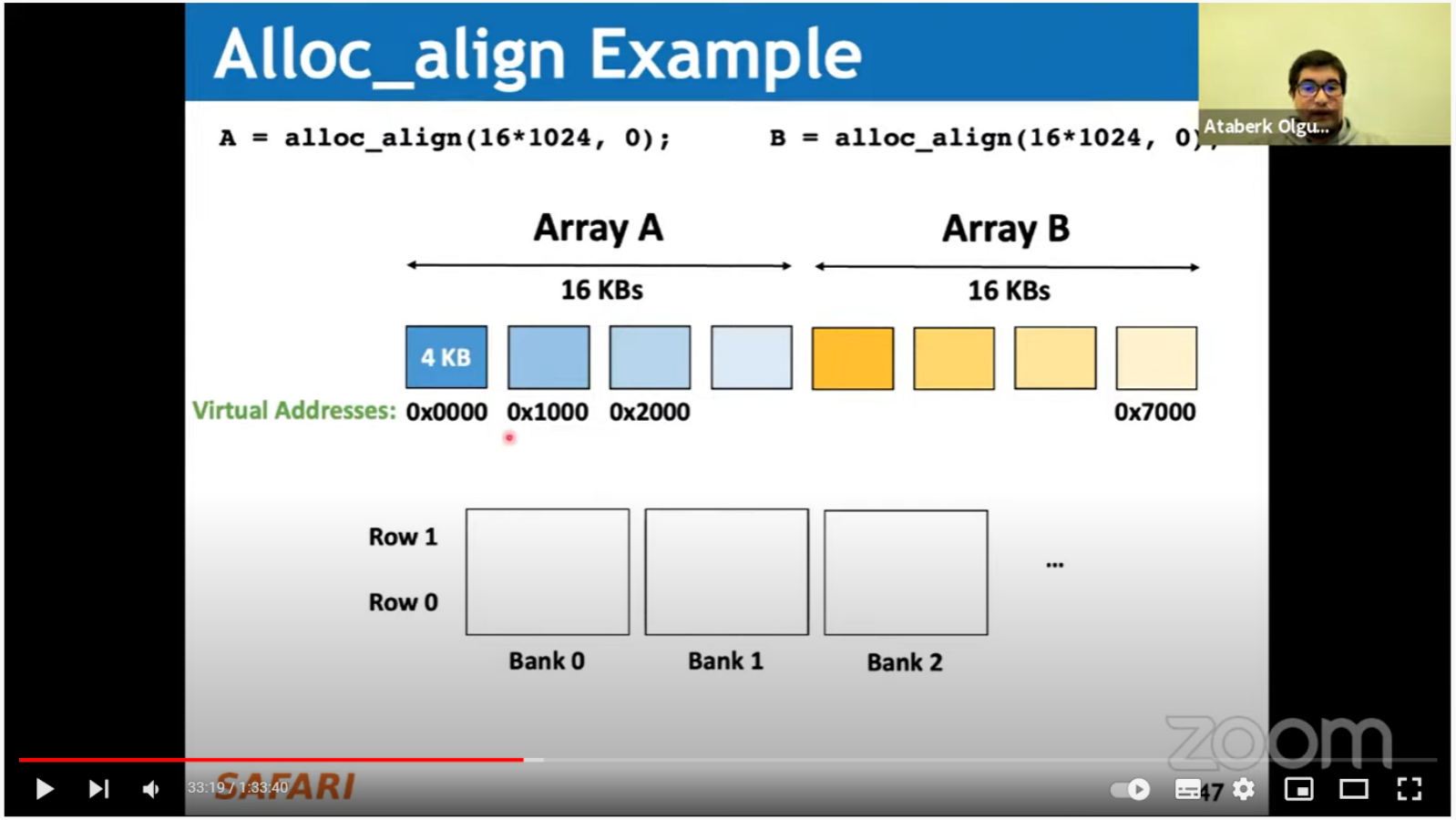
Processing-using-memory (PuM) techniques leverage the analog operation of memory cells to perform computation. Several recent works have demonstrated PuM techniques in off-the-shelf DRAM devices. Since DRAM is the dominant memory technology as main memory in current computing systems, these PuM techniques represent an opportunity for alleviating the data movement bottleneck at very low cost. However, system integration of PuM techniques imposes non-trivial challenges that are yet to be solved. Design space exploration of potential solutions to the PuM integration challenges requires appropriate tools to develop necessary hardware and software components. Unfortunately, current specialized DRAM-testing platforms, or system simulators do not provide the flexibility and/or the holistic system view that is necessary to deal with PuM integration challenges.

We design and develop PiDRAM, the first flexible end-to-end framework that enables system integration studies and evaluation of real PuM techniques. PiDRAM provides software and hardware components to rapidly integrate PuM techniques across the whole system software and hardware stack (e.g., necessary modifications in the operating system, memory controller). We implement PiDRAM on an FPGA-based platform along with an open-source RISC-V system. Using PiDRAM, we implement and evaluate two state-of-the-art PuM techniques: in-DRAM (i) copy and initialization, (ii) true random number generation. Our results show that the in-memory copy and initialization techniques can improve the performance of bulk copy operations by 12.6x and bulk initialization operations by 14.6x on a real system. Implementing the true random number generator requires only 190 lines of Verilog and 74 lines of C code using PiDRAM's software and hardware components.

Comments: 15 pages, 12 figures
Subjects: **Hardware Architecture (cs.AR)**
Cite as: arXiv:2111.00082 **[cs.AR]**
(or arXiv:2111.00082v3 **[cs.AR]** for this version)
https://doi.org/10.48550/arXiv.2111.00082

Long Talk + Tutorial on Youtube

https://youtu.be/s_z_S6FYpC8



Processing in Memory Course: Meeting 6: End-to-end Framework for Processing-using-Memory - Fall'21

615 views • Streamed live on 9 Nov 2021 • Project & Seminar, ETH Zürich, Fall 202 Show more

25 Dislike Share Download Clip Save ...

Onur Mutlu Lectures
25.7K subscribers

SUBSCRIBED

PiDRAM

An FPGA-based Framework for End-to-end Evaluation of Processing-in-DRAM Techniques

Ataberk Olgun

Juan Gomez Luna Konstantinos Kanellopoulos Behzad Salami

Hasan Hassan Oğuz Ergin Onur Mutlu

SAFARI

ETH zürich

kasırga

TOBB ETÜ
University of Economics & Technology

DRAM Bender

An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun
Hasan Hassan
A. Giray Yaglikci
Yahya Can Tugrul
Lois Orosa
Haocong Luo
Minesh Patel
Oguz Ergin
Onur Mutlu

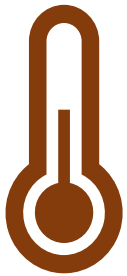
SAFARI
ETH zürich

Factors Affecting DRAM Reliability and Latency

DRAM timing violation

Inter-cell interference

Manufacturing process



Temperature

Voltage

Factors affecting DRAM reliability and latency cannot be properly **modeled** in simulation or analytically

We need to perform experimental studies of real DRAM chips

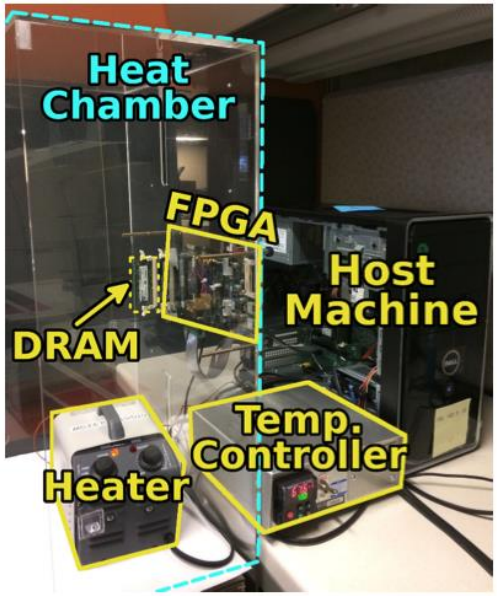
DRAM Testing Infrastructure

Allow experimental studies of real DRAM chips

Open-source FPGA-based testing infrastructure

- Publicly-available: Start using today
- Relatively low cost: An FPGA board + DRAM modules

SoftMC



Litex Tester

Limitations of Existing Infrastructure

Testing Infrastructure	Interface (IF) Restrictions	Ease of Use	Extensibility
SoftMC [134]	Data IF	✗	✗
LiteX RowHammer Tester (LRT) [17]	Command & Data IF	✗	✓
DRAM Bender (this work)	**No Restrictions**	✓	✓

Impose restrictions on the DDR4 interface.
Restrictions limit various characterization experiments.

Difficult to set up (based on discontinued HW/SW)
and use (require developing HW)

Monolithic hardware design
makes extensions (new standards, prototypes) relatively difficult

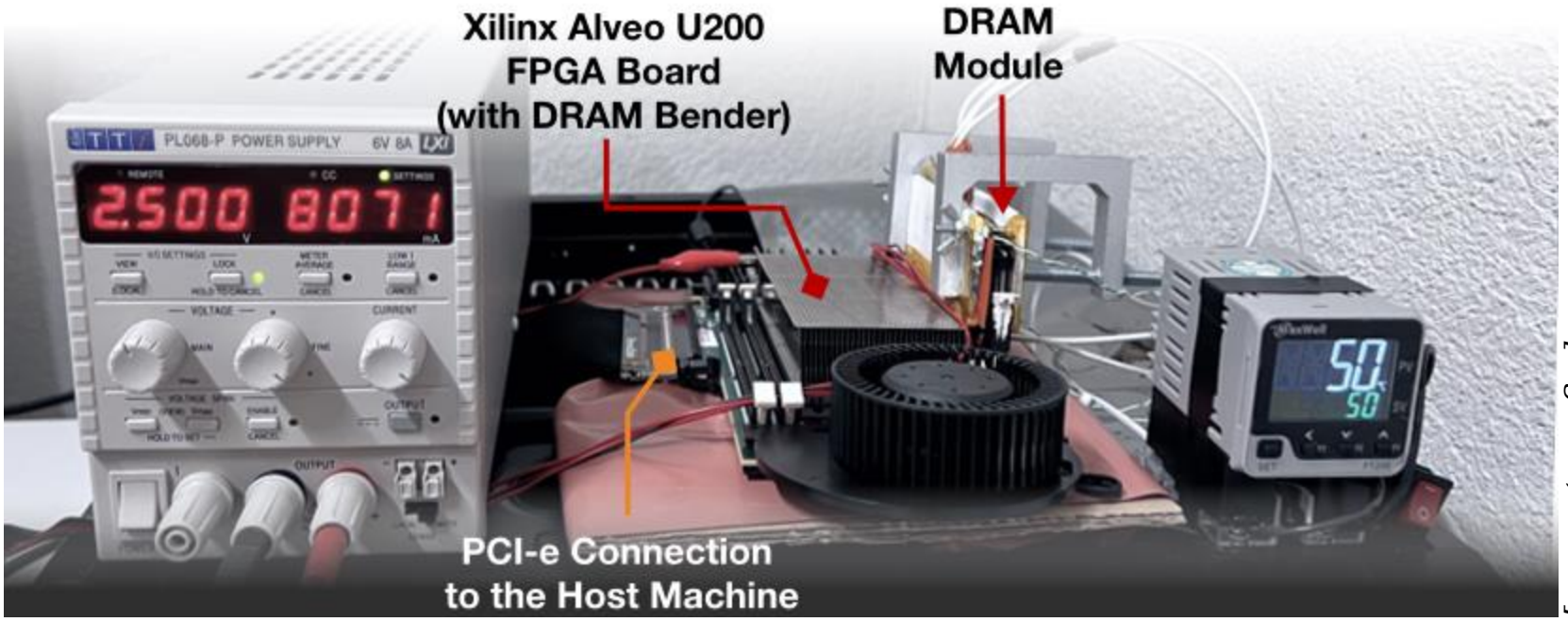
DRAM Bender: Design Goals

- Flexibility
  - Ability to test any DRAM operation
  - Ability to test any combination of DRAM operations and custom timing parameters
- Ease of use
  - Simple programming interface (C++)
  - Minimal programming effort and time
  - Accessible to a wide range of users
    - *who may lack experience in hardware design*
- Extensibility
  - Modular design
  - Well-defined interfaces between hardware modules

DRAM Bender: Overview

Publicly-available FPGA-based
DDR4/3 (and HBM2) characterization infrastructure

Easily programmable using the DRAM Bender C++ API

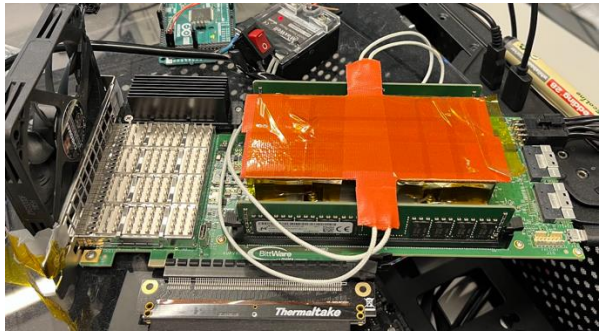


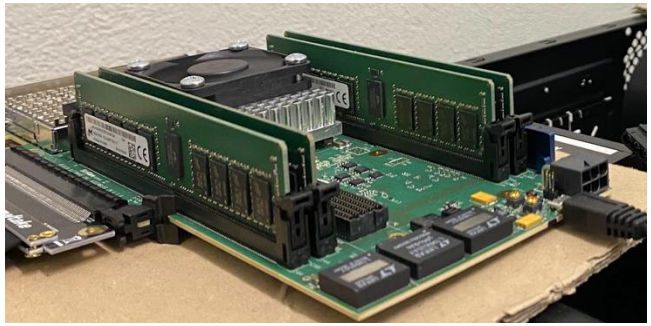
[Yaglikci+, DSN'22]

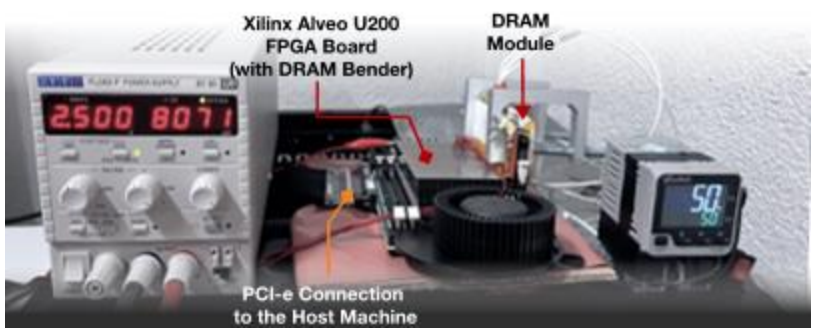
DRAM Bender: Prototypes

Testing Infrastructure	Protocol Support	FPGA Support
SoftMC [134]	DDR3	One Prototype
LiteX RowHammer Tester (LRT) [17]	DDR3/4, LPDDR4	Two Prototypes
DRAM Bender (this work)	**DDR3/DDR4**	**Five Prototypes**

Five out of the box FPGA-based prototypes







DRAM Bender: Three Case Studies

1. RowHammer: Interleaving Pattern of Activations
   - Interleaving pattern significantly affects the number of RowHammer bitflips

2. RowHammer: Random Data Patterns
   - Use 512-bit random data patterns
   - Uncover more bitflips than 8-bit SoftMC random patterns

3. In-DRAM Bitwise Operations
   - Demonstrate in-DRAM bitwise AND/OR capability in real DDR4 chips

DRAM Bender is flexible:
supports many different types of experiments

DRAM Bender: Ease of Use

Easily programmable using the DRAM Bender C++ API

1. RowHammer: Interleaving Pattern of Activations

```
1   p.appendLI(hammerCount, 0);
2   p.appendLabel("HAMMER1");
3   p.appendACT(bank, false, A1, false, tRAS);
4   p.appendPRE(bank, false, false, tRP);
5   p.appendADDI(hammerCount, hammerCount, 1);
6   p.appendBL(hammerCount, T, "HAMMER1");
7   p.appendLI(hammerCount, 0);
8   p.appendLabel("HAMMER2");
9   p.appendACT(bank, false, A2, false, tRAS);
10  p.appendPRE(bank, false, false, tRP);
11  p.appendADDI(hammerCount, hammerCount, 1);
12  p.appendBL(hammerCount, T, "HAMMER2");
```

one iteration of the RowHammer test

Easy to devise new experiments to uncover new insights.

More in the paper (II)

- DRAM Bender design details
  - DRAM Bender instruction set architecture
  - Hardware & software modules
  - Prototype design
  - Temperature controller setup
- DRAM Bender application programming interface
- Detailed results for three case studies
- Future work & improvements

More in the paper (II)

DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun[§] Hasan Hassan[§] A. Giray Yağlıkçı[§] Yahya Can Tuğrul[§†]
Lois Orosa[§⊙] Haocong Luo[§] Minesh Patel[§] Oğuz Ergin[†] Onur Mutlu[§]
[§]*ETH Zürich* [†]*TOBB ETÜ* [⊙]*Galician Supercomputing Center*

https://arxiv.org/abs/2211.05838

Research DRAM Bender Enabled

1) **[HPCA'25]** Olgun+, "Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance"
2) **[HPCA'25]** Tugrul+, "Understanding RowHammer Under Reduced Refresh Latency: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions"
3) **[DSN'24]** Olgun+, "Read Disturbance in High Bandwidth Memory: A Detailed Experimental Study on HBM2 DRAM Chips"
4) **[DSN'24]** Yuksel+, "Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis"
5) **[DSN'24 Disrupt]** Luo+, "An Experimental Characterization of Combined RowHammer and RowPress Read Disturbance in Modern DRAM Chips"
6) **[HPCA'24]** Yaglikci+, "Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions"
7) **[HPCA'24]** Yuksel+, "Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis"
8) **[ISCA'23]** Luo+, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips"
9) **[DSN'23 Disrupt]** Olgun+, "An Experimental Analysis of RowHammer on HBM2 DRAM Chips"
10) **[arXiv Preprint, 2023]** Orosa+, "SpyHammer: Using RowHammer to Remotely Spy on Temperature"
11) **[MICRO'22]** Yaglikci+, "HIRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips"
12) **[DSN'22]** Yaglikci+, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices"
13) **[MICRO'21]** Orosa+, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"
14) **[MICRO'21]** Hassan+, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"
15) **[ISCA'21]** Olgun+, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"

More Research DRAM Bender Enabled

14) **[ISCA'21]** Orosa+, "CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations"

15) **[ISCA'20]** Kim+, "Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"

16) **[S&P'20]** Frigo+, "TRRespass: Exploiting the Many Sides of Target Row Refresh"

17) **[HPCA'19]** Kim+, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"

18) **[MICRO'19]** Koppula+, "EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM"

19) **[SIGMETRICS'18]** Ghose+, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study"

20) **[SIGMETRICS'17]** Chang+, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms"

21) **[MICRO'17]** Khan+, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"

22) **[SIGMETRICS'16]** Chang+, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"

Even More Research DRAM Bender Enabled

23) **[ISCA'24]** Nam+, "DRAMScope: Uncovering DRAM Microarchitecture and Characteristics by Issuing Memory Commands"

24) **[DATE'24]** Zhou+, "DRAM-Locker: A General-Purpose DRAM Protection Mechanism Against Adversarial DNN Weight Attacks"

25) **[DRAMSec'23]** Lang+, "BLASTER: Characterizing the Blast Radius of Rowhammer"

26) **[IEEE CAL'23]** Nam+ "X-ray: Discovering DRAM Internal Structure and Error Characteristics by Issuing Memory Commands"

27) **[MICRO'22]** Gao+, **"**FracDRAM: Fractional Values in Off-the-Shelf DRAM"

28) **[Applied Sciences'22]** Bepary+, "DRAM Retention Behavior with Accelerated Aging in Commercial Chips"

29) **[ETS'21]** Farmani+, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules"

30) **[HOST'20]** Talukder+, "Towards the Avoidance of Counterfeit Memory: Identifying the DRAM Origin"

31) **[MICRO'19]** Gao+, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs"

32) **[IEEE Access'19]** Talukder+, "PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures"

33) **[ICCE'18]** Talukder+, "Exploiting DRAM Latency Variations for Generating True Random Numbers"

DRAM Chips Are Already (Quite) Capable!

- **Appears at HPCA 2024** https://arxiv.org/pdf/2402.18736.pdf

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

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 characterization of* new *bulk bitwise operations in 256 off-the-shelf modern DDR4 DRAM chips. We evaluate the reliability of these operations using a metric called success rate: the fraction of correctly performed bitwise operations. Among our 19 new observations, we highlight four major results. First, we can perform the NOT operation on COTS DRAM chips with 98.37% success rate on average. Second, we can perform up to 16-input NAND, NOR, AND, and OR operations on COTS DRAM chips with high reliability (e.g., 16-input NAND, NOR, AND, and OR with average success rate of 94.94%, 95.87%, 94.94%, and 95.85%, respectively). Third, data pattern only slightly*

DRAM Chips Are Already (Quite) Capable!

- **Appears at 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*

https://arxiv.org/pdf/2405.06081

The Capability of COTS DRAM Chips

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

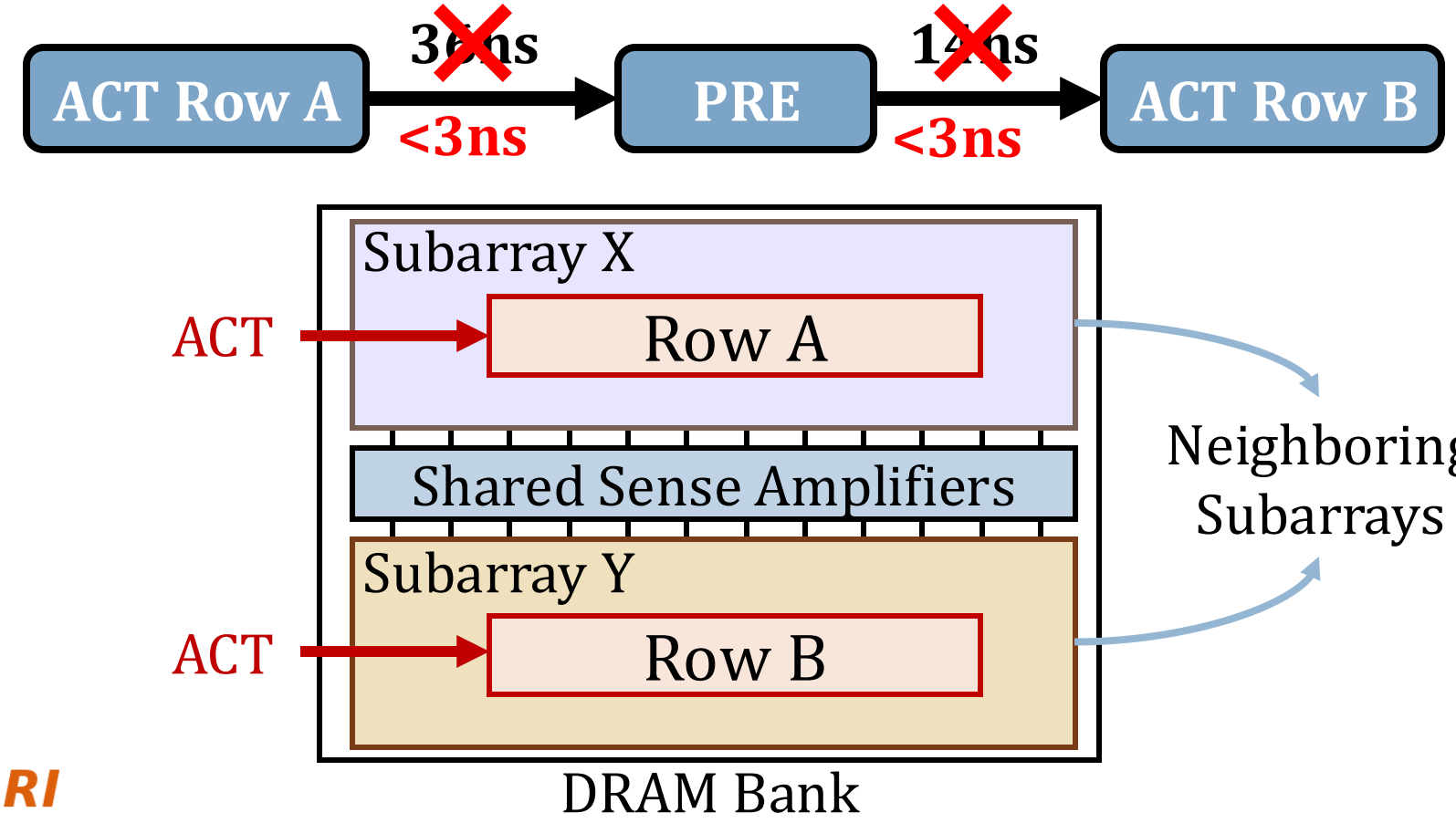
1 Can **simultaneously activate** up to **48 rows** in **two neighboring subarrays**

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

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

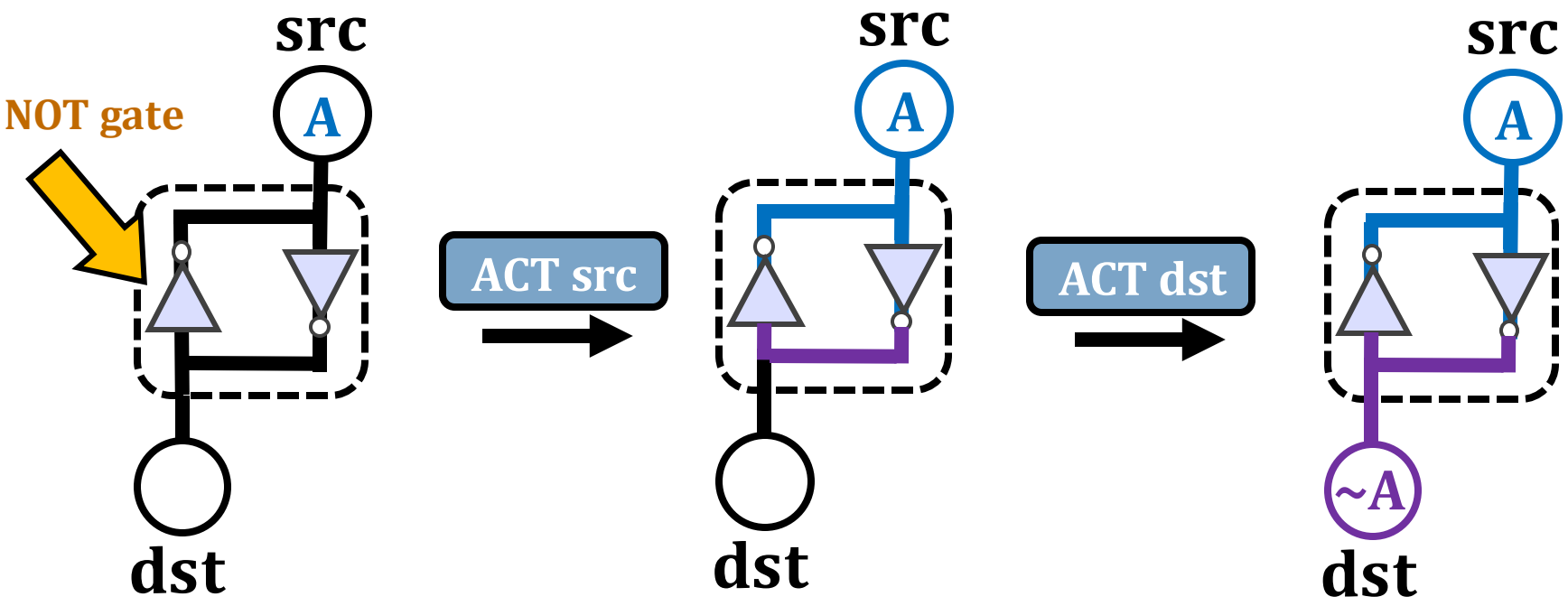
Finding: SiMRA Across Subarrays

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



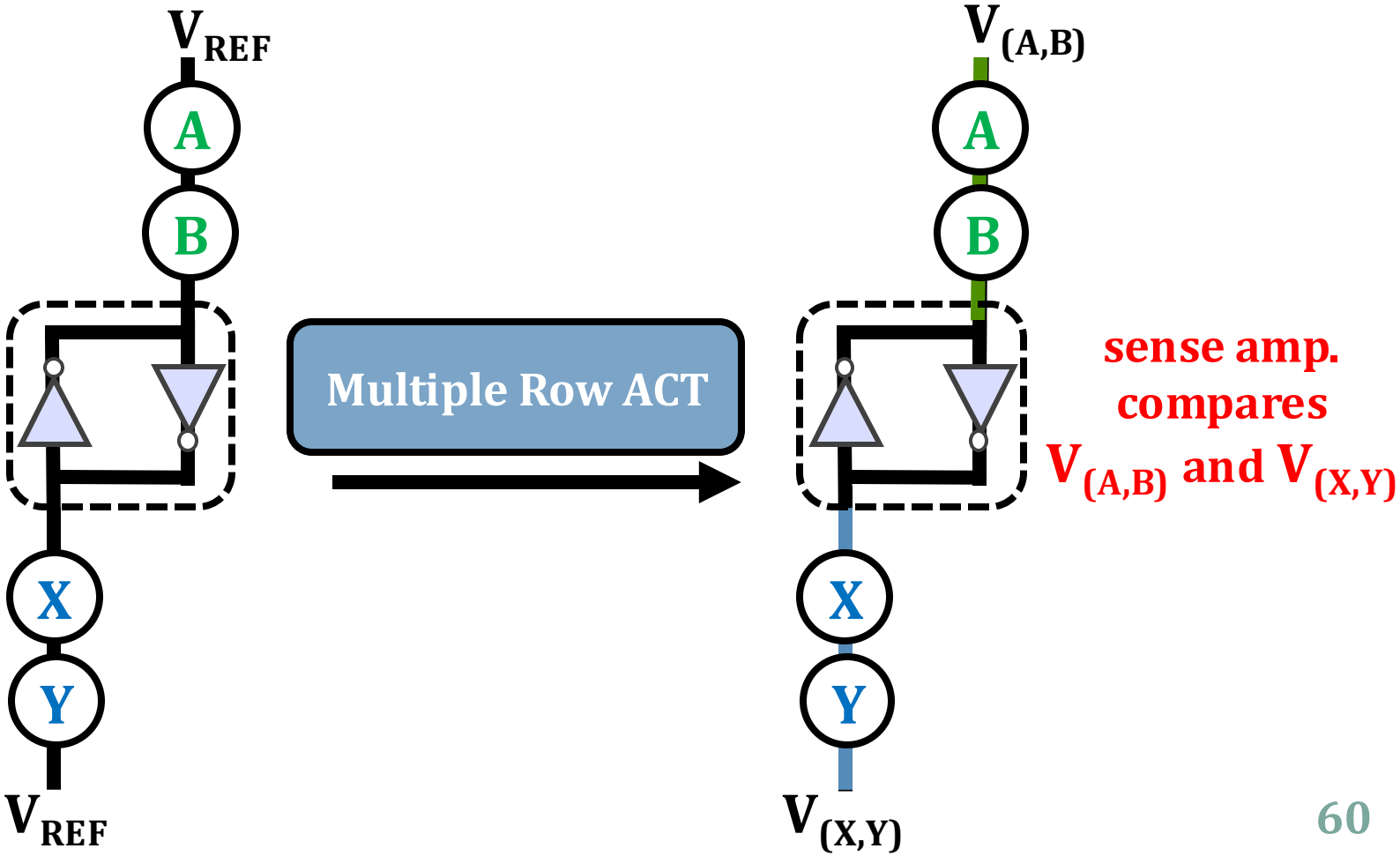
Key Idea: NOT Operation

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

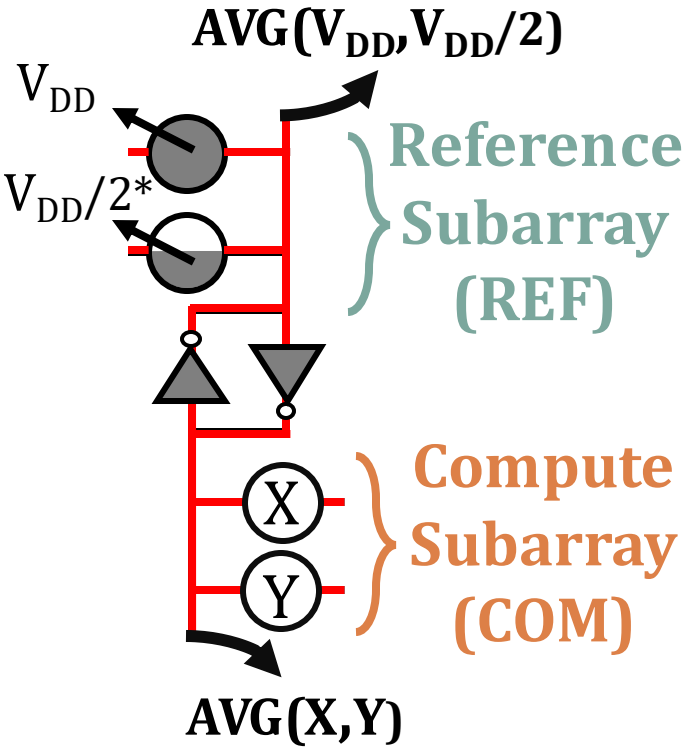


Key Idea: NAND, NOR, AND, OR

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

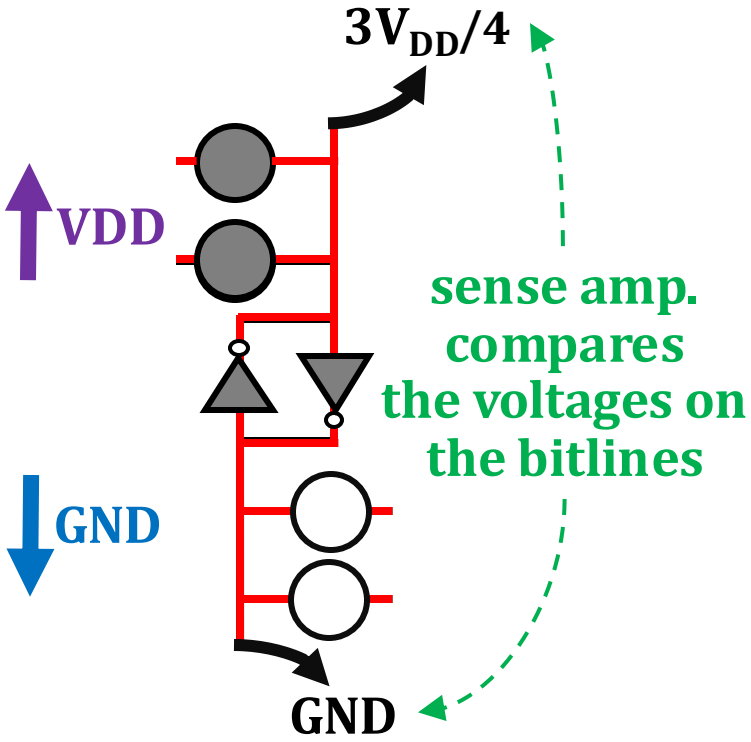


Two-Input AND and NAND Operations



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

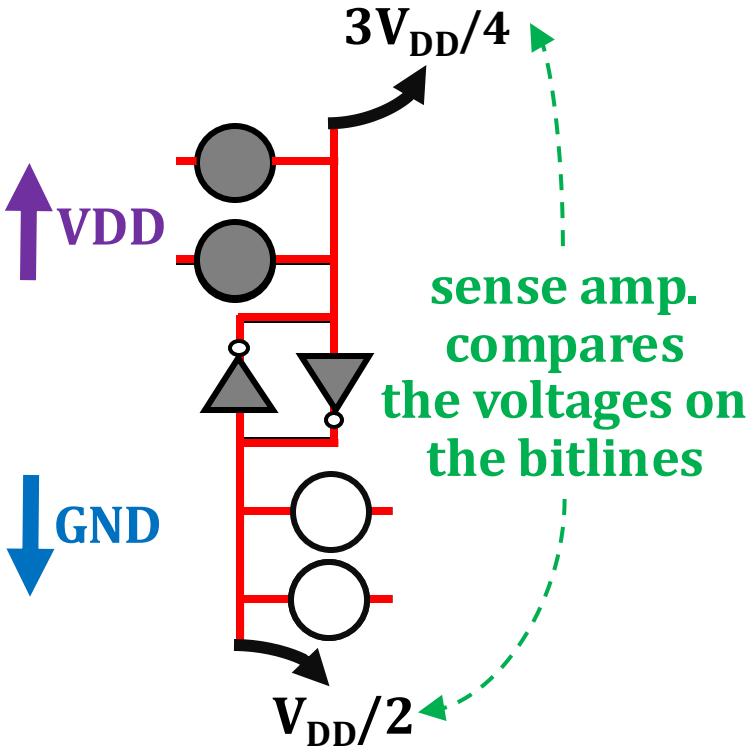
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1

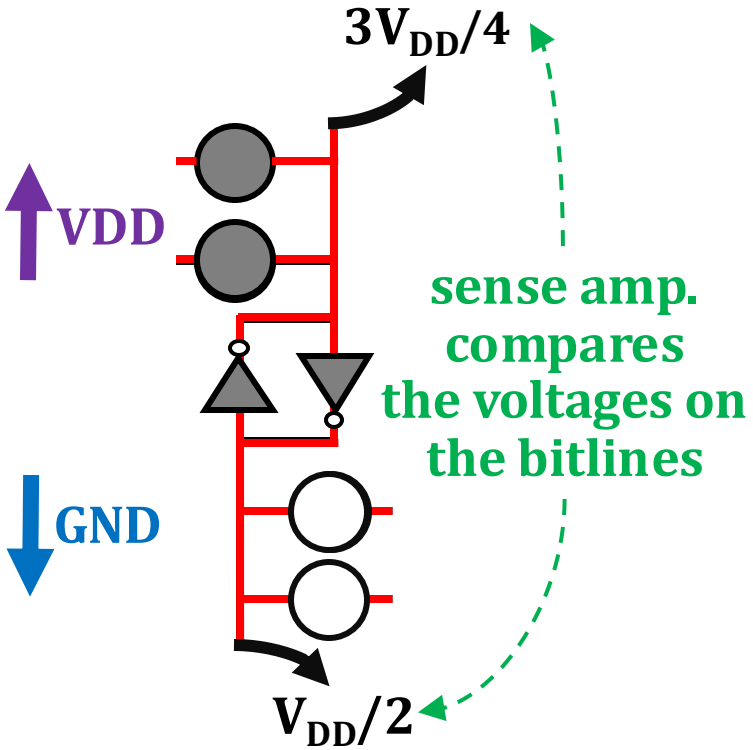
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

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

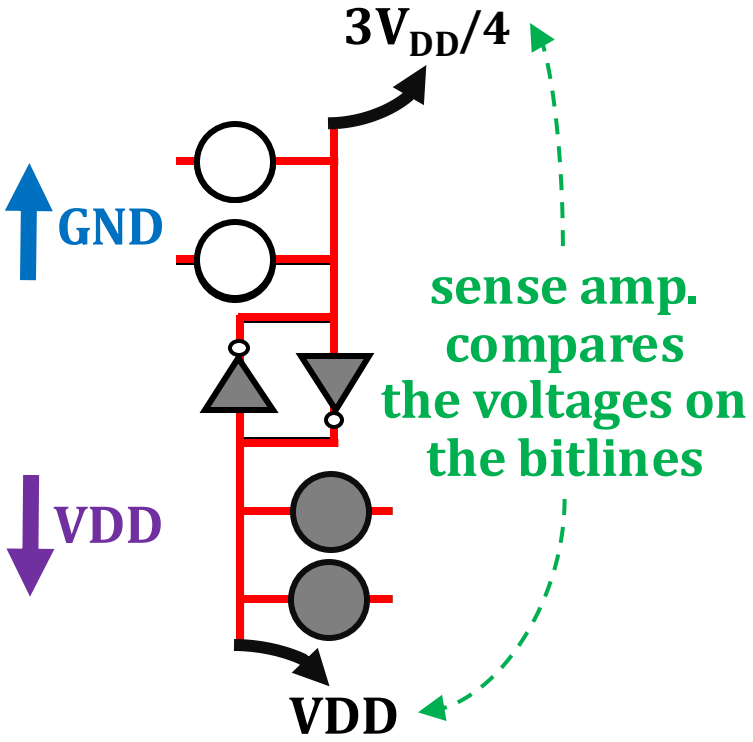
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

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

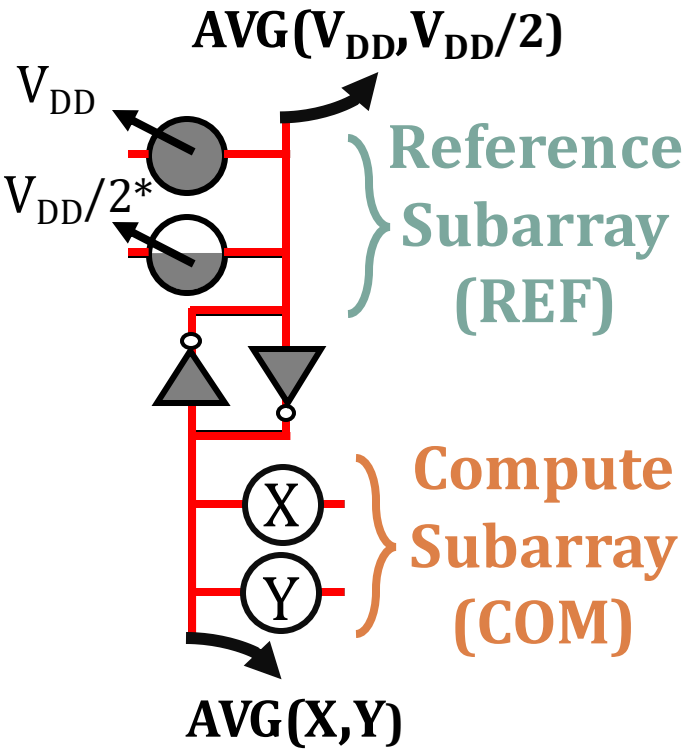
Two-Input AND and NAND Operations



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.

Many-Input AND, NAND, OR, and NOR Operations

We can express AND, NAND, OR, and NOR operations by carefully manipulating the **reference voltage**

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

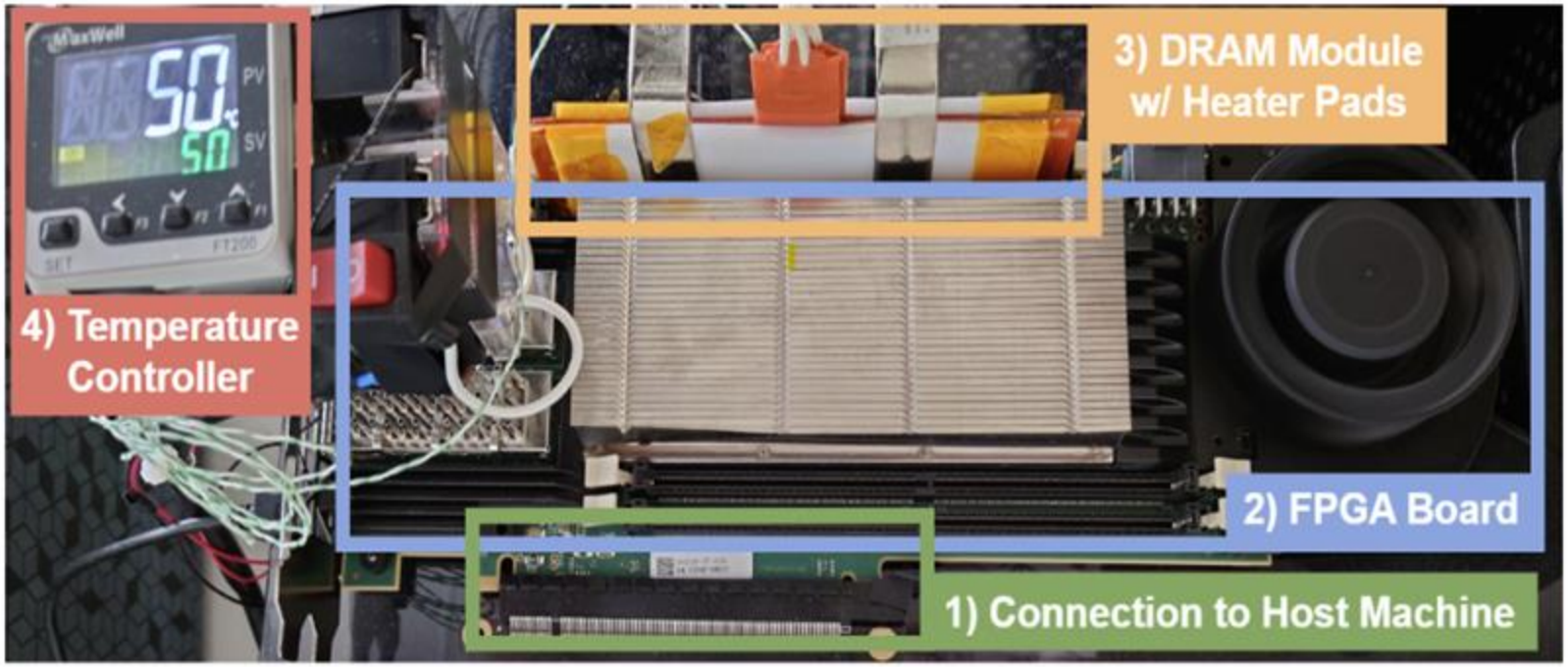
ETH Zürich

(More details in the paper)

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

DRAM Testing Infrastructure

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



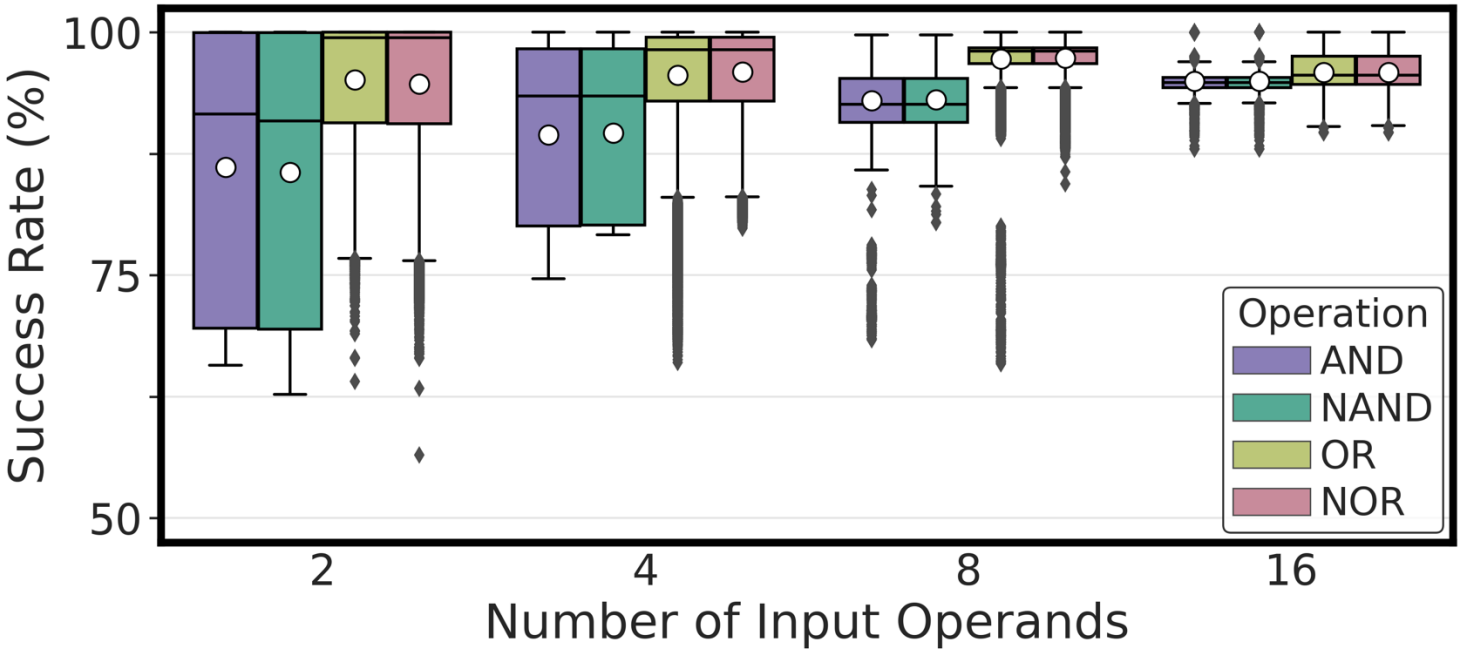
*Olgun et al., "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," TCAD, 2023.

DRAM Chips Tested

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

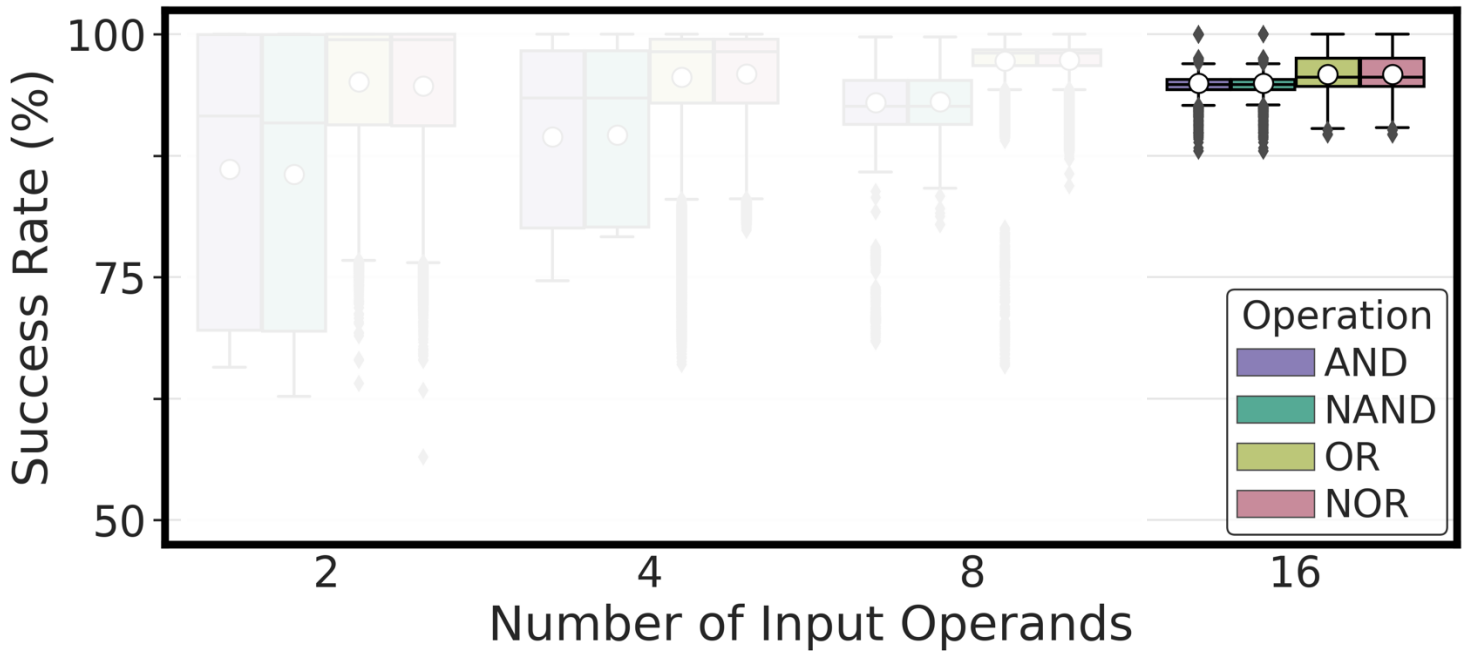
Chip Mfr.	#Modules (#Chips)	Die Rev.	Mfr. Date[a]	Chip Density	Chip Org.	Speed Rate
SK Hynix	9 (72)	M	N/A	4Gb	x8	2666MT/s
	5 (40)	A	N/A	4Gb	x8	2133MT/s
	1 (16)	A	N/A	8Gb	x8	2666MT/s
	1 (32)	A	18-14	4Gb	x4	2400MT/s
	1 (32)	A	16-49	8Gb	x4	2400MT/s
	1 (32)	M	16-22	8Gb	x4	2666MT/s
Samsung	1 (8)	F	21-02	4Gb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

Performing AND, NAND, OR, and NOR



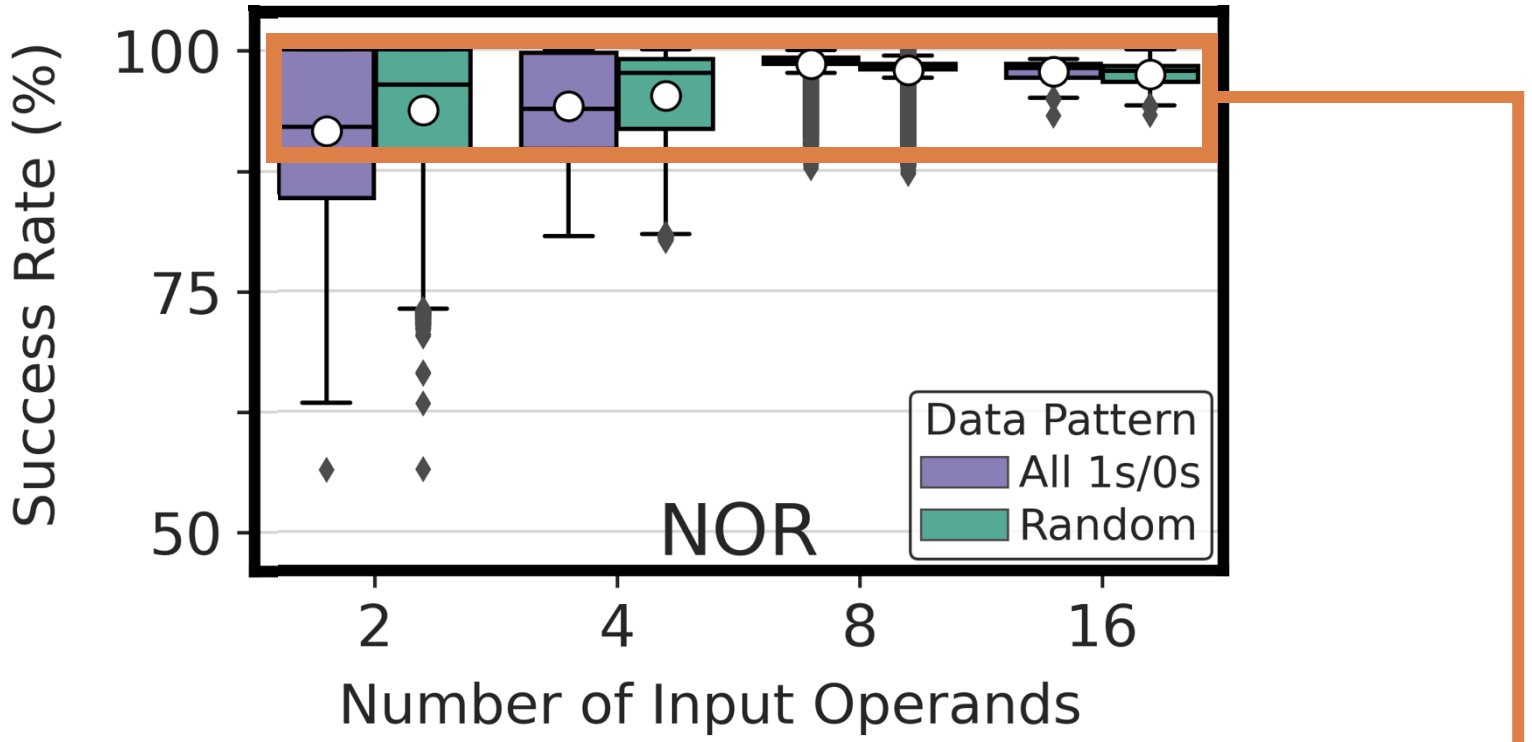
COTS DRAM chips can perform {2, 4, 8, 16}-input AND, NAND, OR, and NOR operations

Performing AND, NAND, OR, and NOR



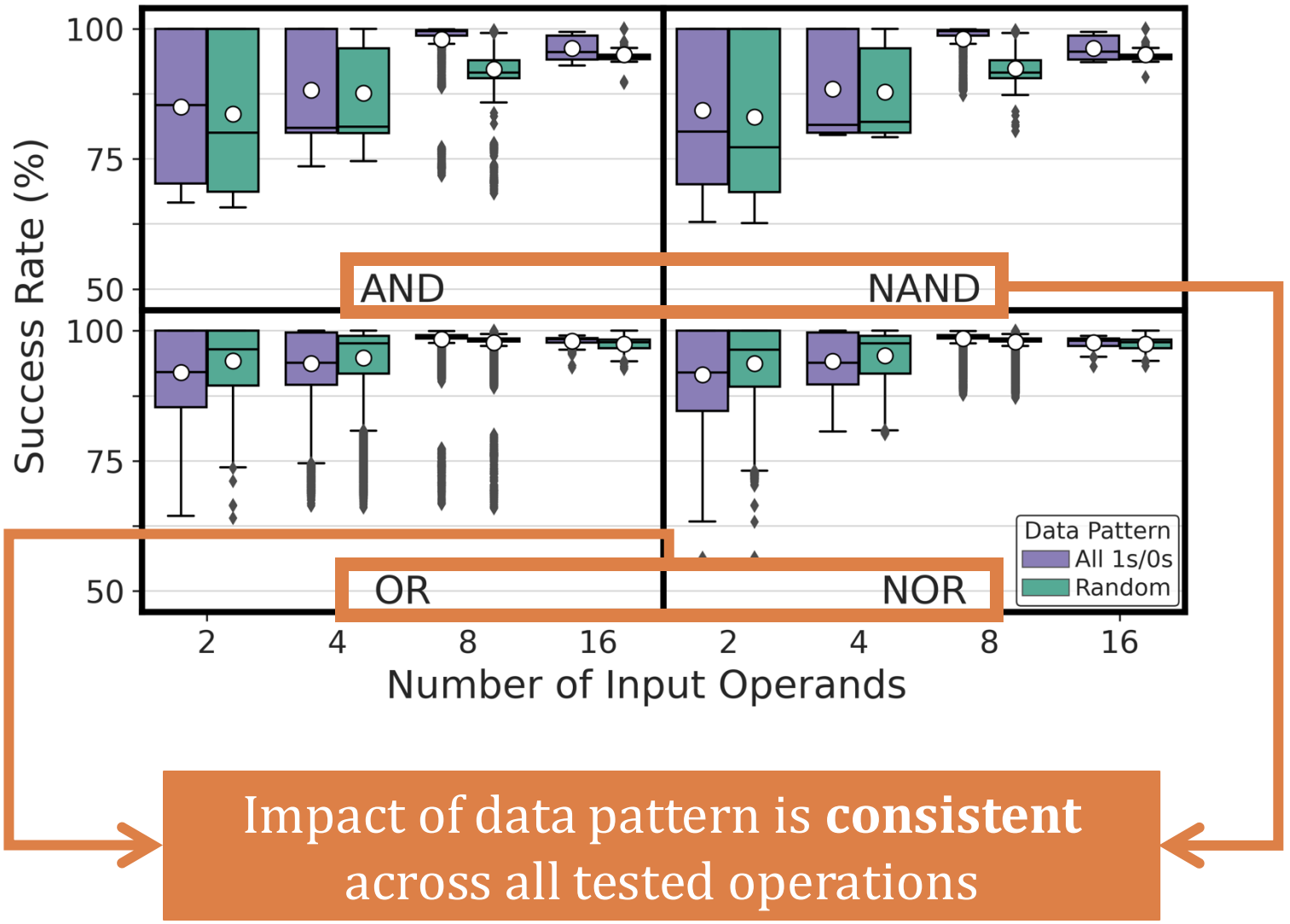
COTS DRAM chips can perform 16-input AND, NAND, OR, and NOR operations with very high success rate (>94%)

Impact of Data Pattern



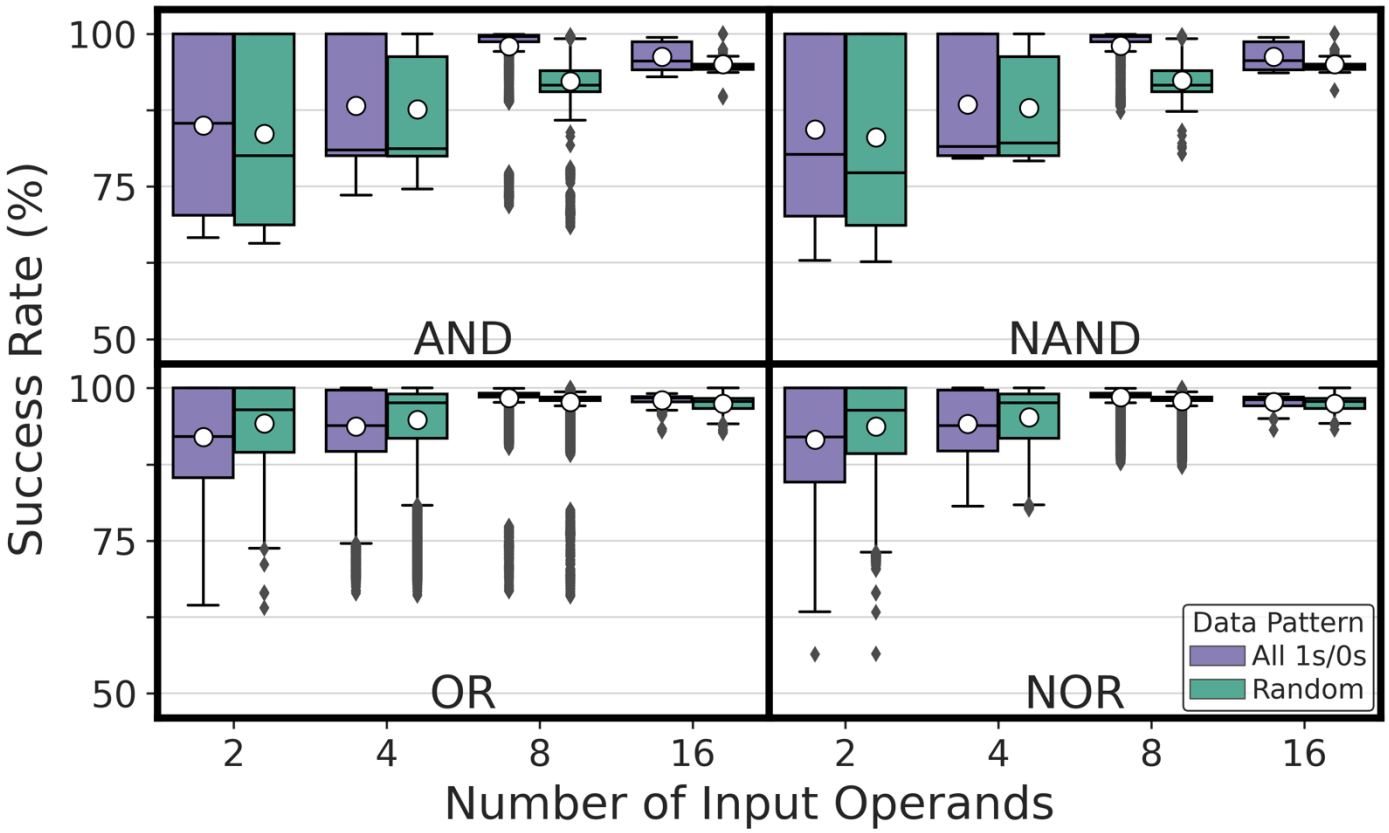
1.98% variation in **average success rate** across **all number of input operands**

Impact of Data Pattern



Impact of data pattern is **consistent** across all tested operations

Impact of Data Pattern



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

More in the Paper

- Detailed hypotheses & key ideas to perform
  - NOT operation
  - Many-input AND, NAND, OR, and NOR operations

- How the reliability of bitwise operations are affected by
  - The location of activated rows
  - Temperature (for AND, NAND, OR, and NOR)
  - DRAM speed rate
  - Chip density and die revision

- Discussion on the limitations of COTS DRAM chips

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

Summary

- We experimentally demonstrate that **commercial off-the-shelf (COTS)** DRAM chips can perform:
  - **Functionally-complete** Boolean operations: NOT, NAND, and NOR
  - **Up to 16-input** AND, NAND, OR, and NOR operations
- We characterize **the success rate** of these operations on **256 COTS DDR4 chips** from **two major manufacturers**
- We highlight **two key results:**
  - We can perform **NOT** and **{2, 4, 8, 16}-input AND, NAND, OR, and NOR** operations on COTS DRAM chips with **very high success rates (>94%)**
  - **Data pattern** and **temperature** only slightly affect the reliability of these operations

We believe these empirical results demonstrate the promising potential of using DRAM as a computation substrate

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

Executive Summary

Motivation:

- **Processing-Using-DRAM (PUD)** alleviates **data movement bottlenecks**
- Commercial off-the-shelf (COTS) DRAM chips can perform **three-input majority (MAJ3)** and **in-DRAM copy** operations

Goal: To experimentally analyze and understand

- The **computational capability** of COTS DRAM chips beyond that of prior works
- The **robustness** of such capability under various **operating conditions**

Experimental Study: 120 DDR4 chips from **two major manufacturers**

- COTS DRAM chips can perform **MAJ5, MAJ7, and MAJ9** operations and **copy** one DRAM row to **up to 31 different rows** at once
- Storing **multiple redundant copies** of MAJ's input operands (i.e., input replication) drastically increases **robustness** (>30% higher success rate)
- **Operating conditions** (temperature, voltage, and data pattern) **affect** the robustness of in-DRAM operations (by up to 11.52% success rate)

Leveraging Simultaneous Many-Row Activation

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

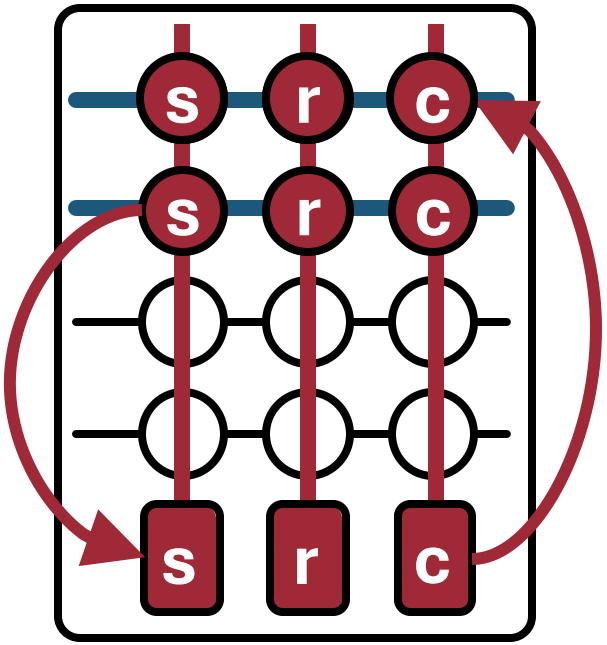
2 **Increase the robustness** of MAJX operations

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

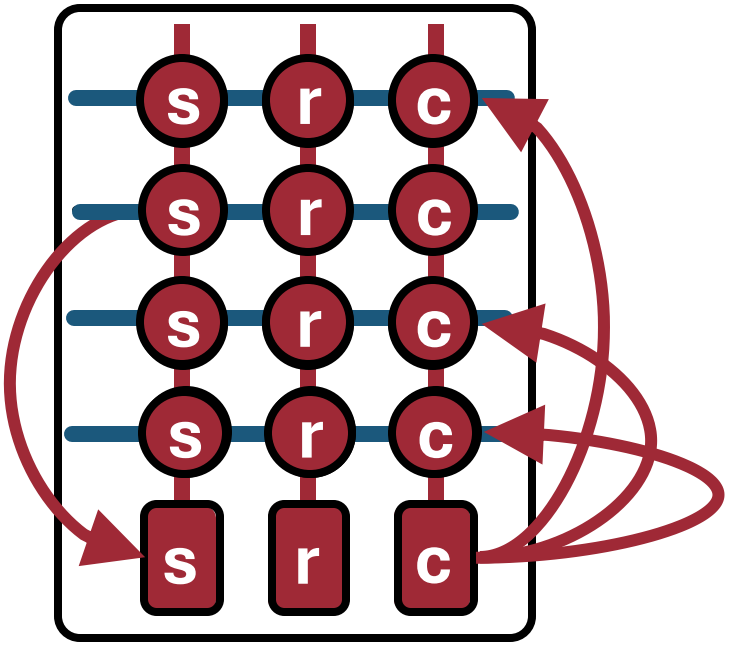
In-DRAM Multiple Row Copy (Multi-RowCopy)

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

RowClone



Multi-RowCopy



[Seshadri+ MICRO'13]

Key Takeaways from Multi-RowCopy

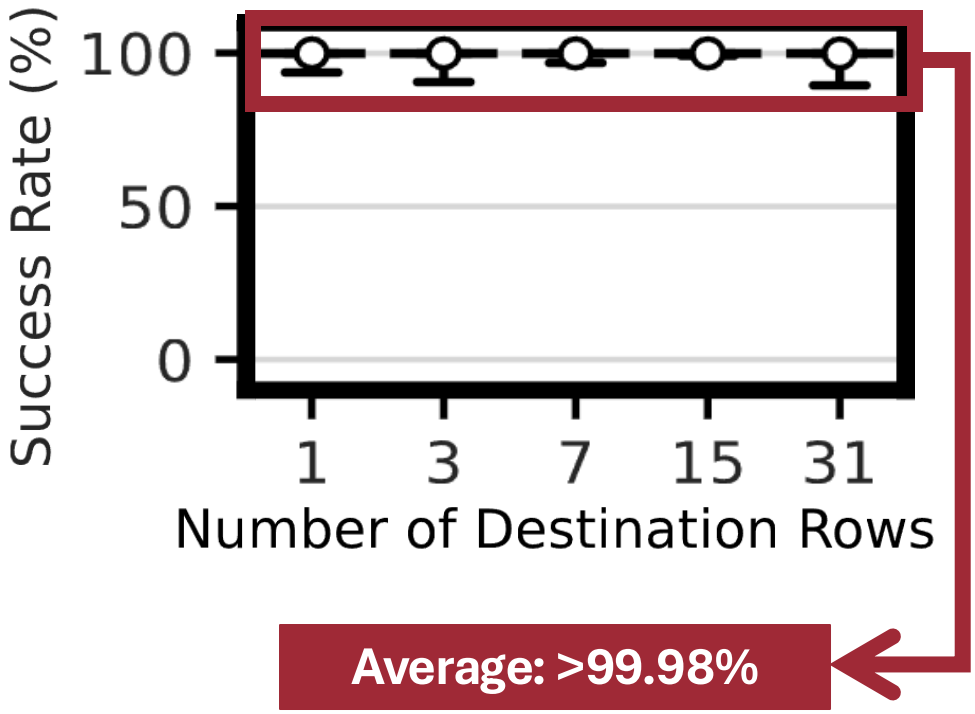
Key Takeaway 1

COTS DRAM chips are capable of copying one row's data to 1, 3, 7, 15, and 31 other rows at very high success rates

Key Takeaway 2

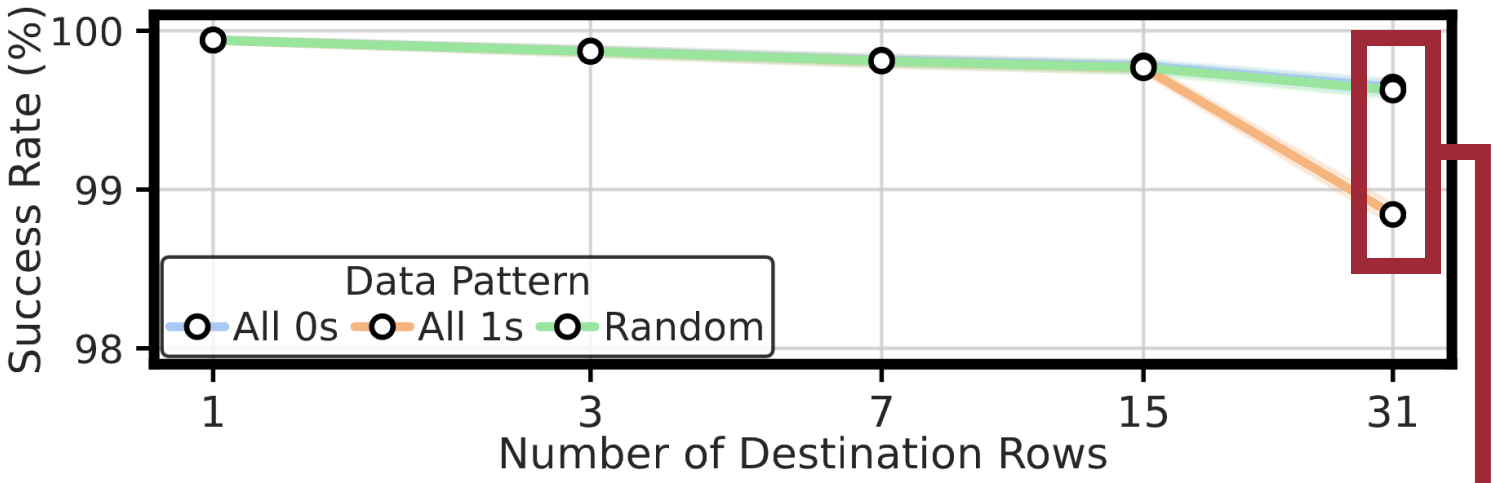
Multi-RowCopy in COTS DRAM chips is highly resilient to changes in data pattern, temperature, and wordline voltage

Robustness of Multi-RowCopy



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

Impact of Data Pattern



At most 0.79% decrease in average success rate

Data pattern has a small effect on the success rate of the Multi-RowCopy operation

Also in the Paper: Impact of Temperature & Voltage

Temperature

50°C → 90°C

Increasing temperature up to 90°C
has a very small effect on
the success rate of the Multi-RowCopy operation

Wordline Voltage

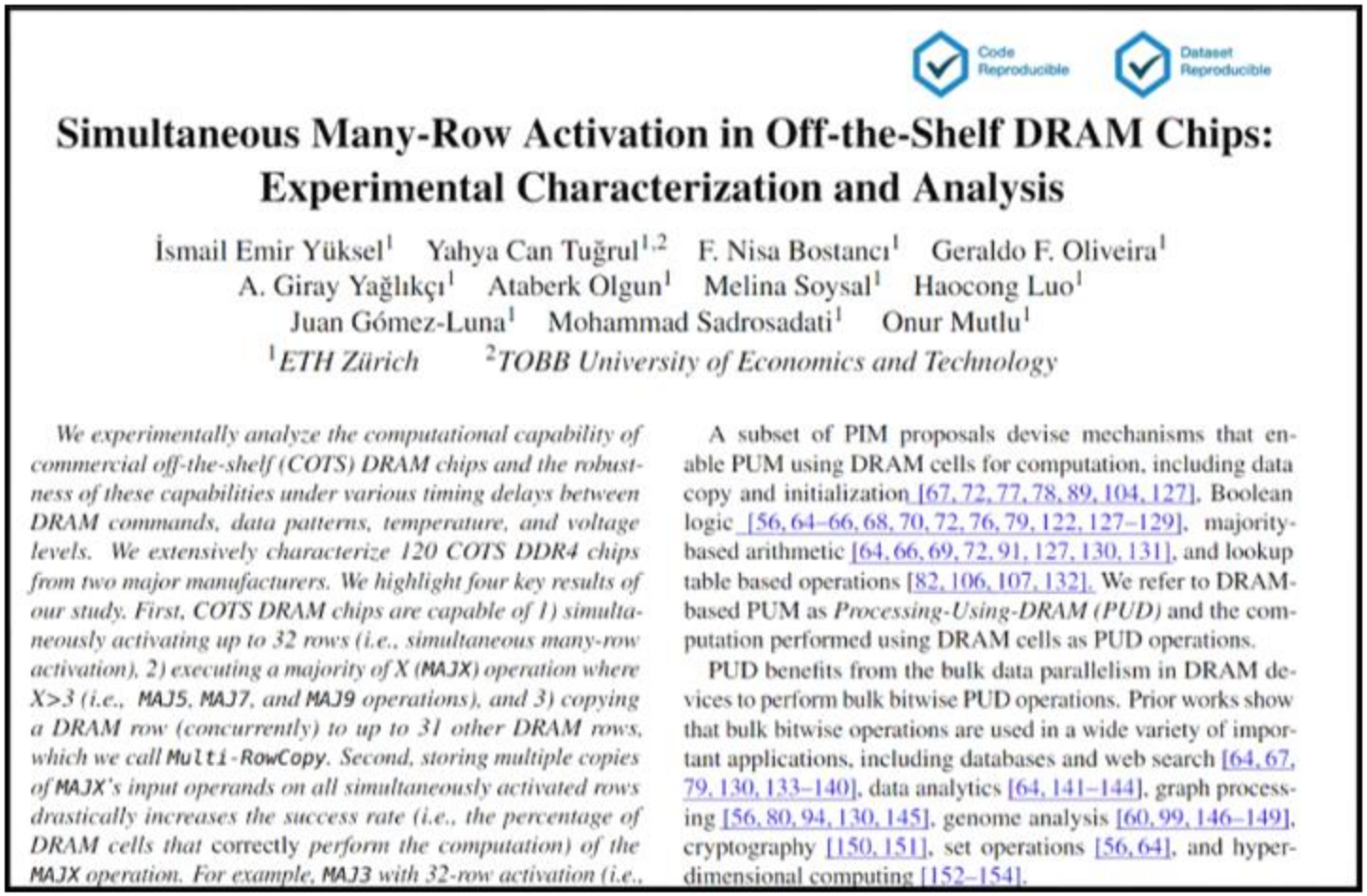
2.5V → 2.1V

Reducing the wordline voltage
only slightly affects
the success rate of the Multi-RowCopy operation

More in the Paper

- Detailed hypotheses and key ideas on
  - Hypothetical row decoder circuitry
  - Input Replication
- More characterization results
  - Power consumption of simultaneous many-row activation
  - Effect of timing delays between ACT-PRE and PRE-ACT commands
  - Effect of temperature and wordline voltage
- Circuit-level (SPICE) experiments for input replication
- Potential performance benefits of enabling new in-DRAM operations
  - Majority-based computation
  - Content destruction-based cold-boot attack prevention
- Discussions on the limitations of tested COTS DRAM chips

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.

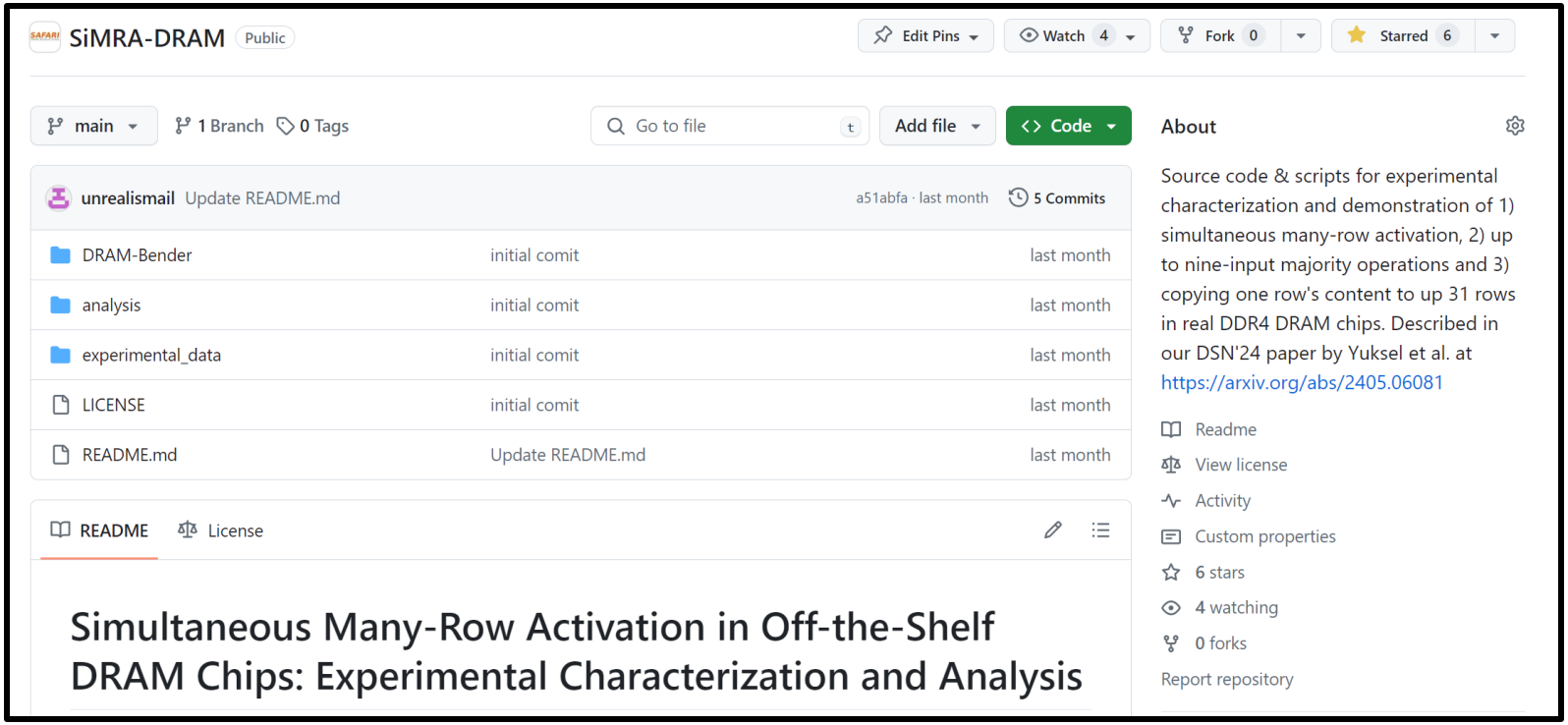
PUD benefits from the bulk data parallelism in DRAM devices to perform bulk bitwise PUD operations. Prior works show that bulk bitwise operations are used in a wide variety of important applications, including databases and web search [64, 67, 79, 130, 133–140], data analytics [64, 141–144], graph processing [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

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

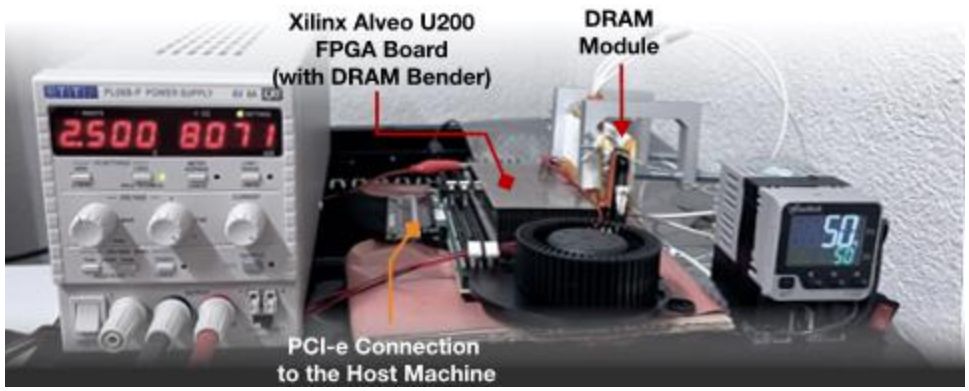
SAFARI ETH zürich

DRAM Bender Summary

DRAM Bender

The first **publicly-available** DDR4 characterization infrastructure

- Flexible and Easy to Use
- **Source code** available:



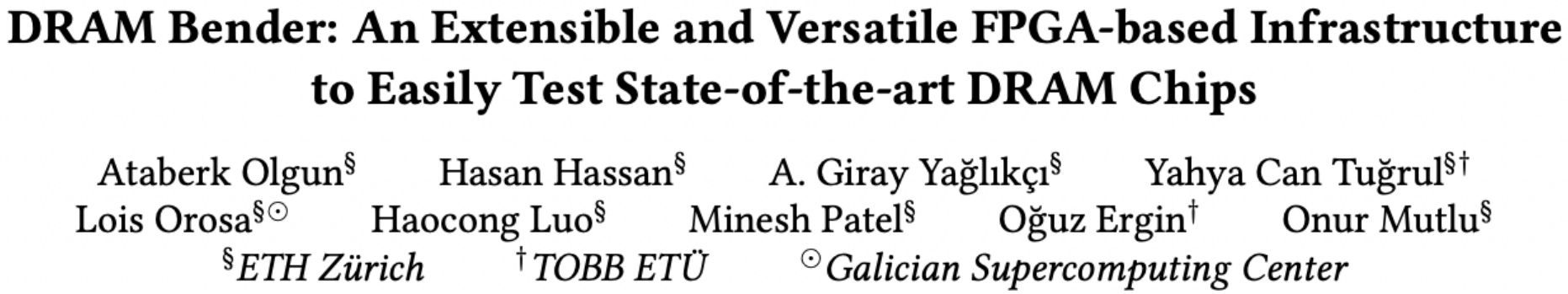
[Yaglikci+, DSN'22]

github.com/CMU-SAFARI/DRAMBender

DRAM Bender enables many studies, ideas, and methodologies in the design of future memory systems

DRAM Bender Paper, Slides, Videos, Code

- Ataberk Olgun, Hasan Hassan, A Giray Yağlıkçı, Yahya Can Tuğrul, Lois Orosa, Haocong Luo, Minesh Patel, Oğuz Ergin, and Onur Mutlu,
  "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips"
  IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (***TCAD***), 2023.
  [Extended arXiv version]
  [DRAM Bender Source Code]
  [DRAM Bender Tutorial Video (43 minutes)]



DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun[§] Hasan Hassan[§] A. Giray Yağlıkçı[§] Yahya Can Tuğrul[§†]
Lois Orosa[§⊙] Haocong Luo[§] Minesh Patel[§] Oğuz Ergin[†] Onur Mutlu[§]
[§]*ETH Zürich* [†]*TOBB ETÜ* [⊙]*Galician Supercomputing Center*

DRAM Bender

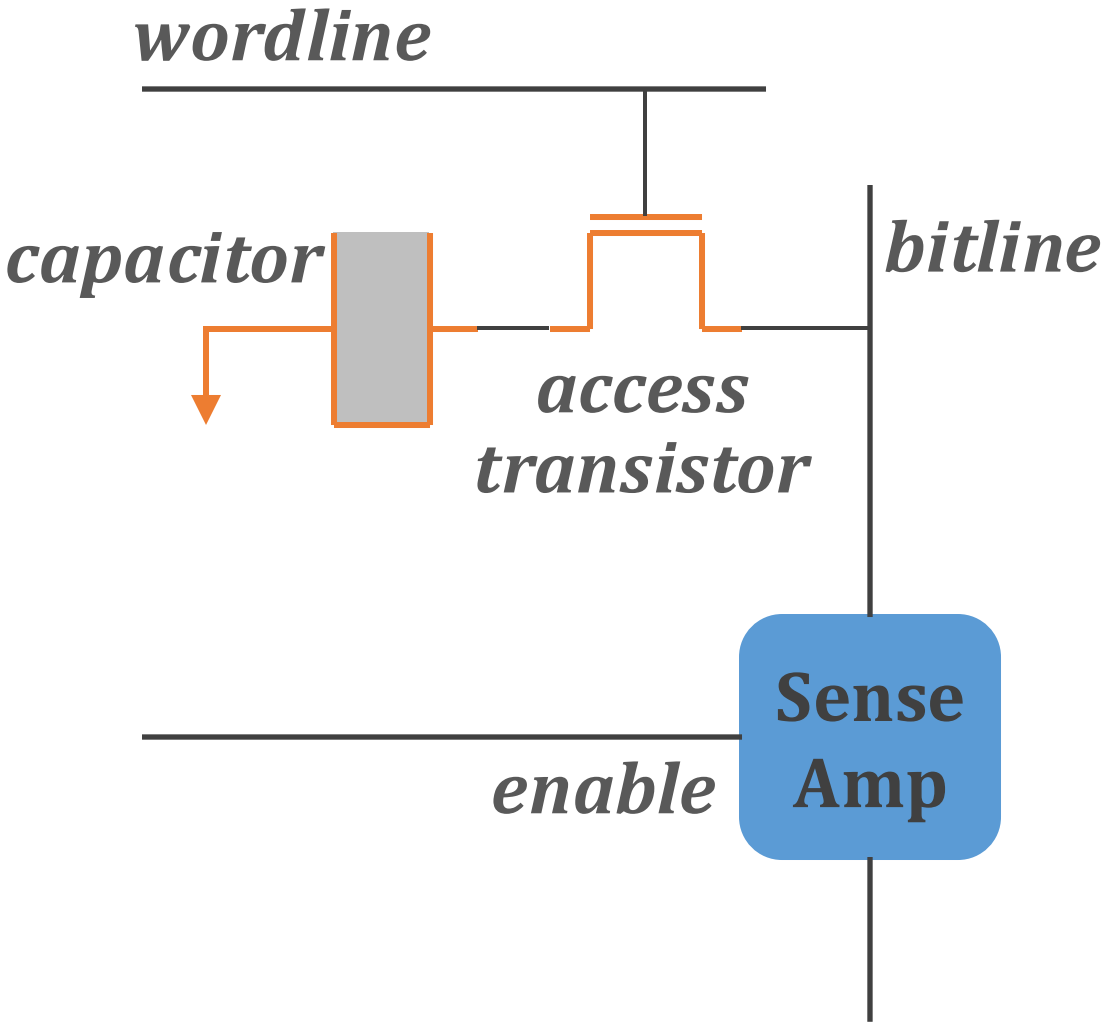
An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun
Hasan Hassan
A. Giray Yaglikci
Yahya Can Tugrul
Lois Orosa
Haocong Luo
Minesh Patel
Oguz Ergin
Onur Mutlu

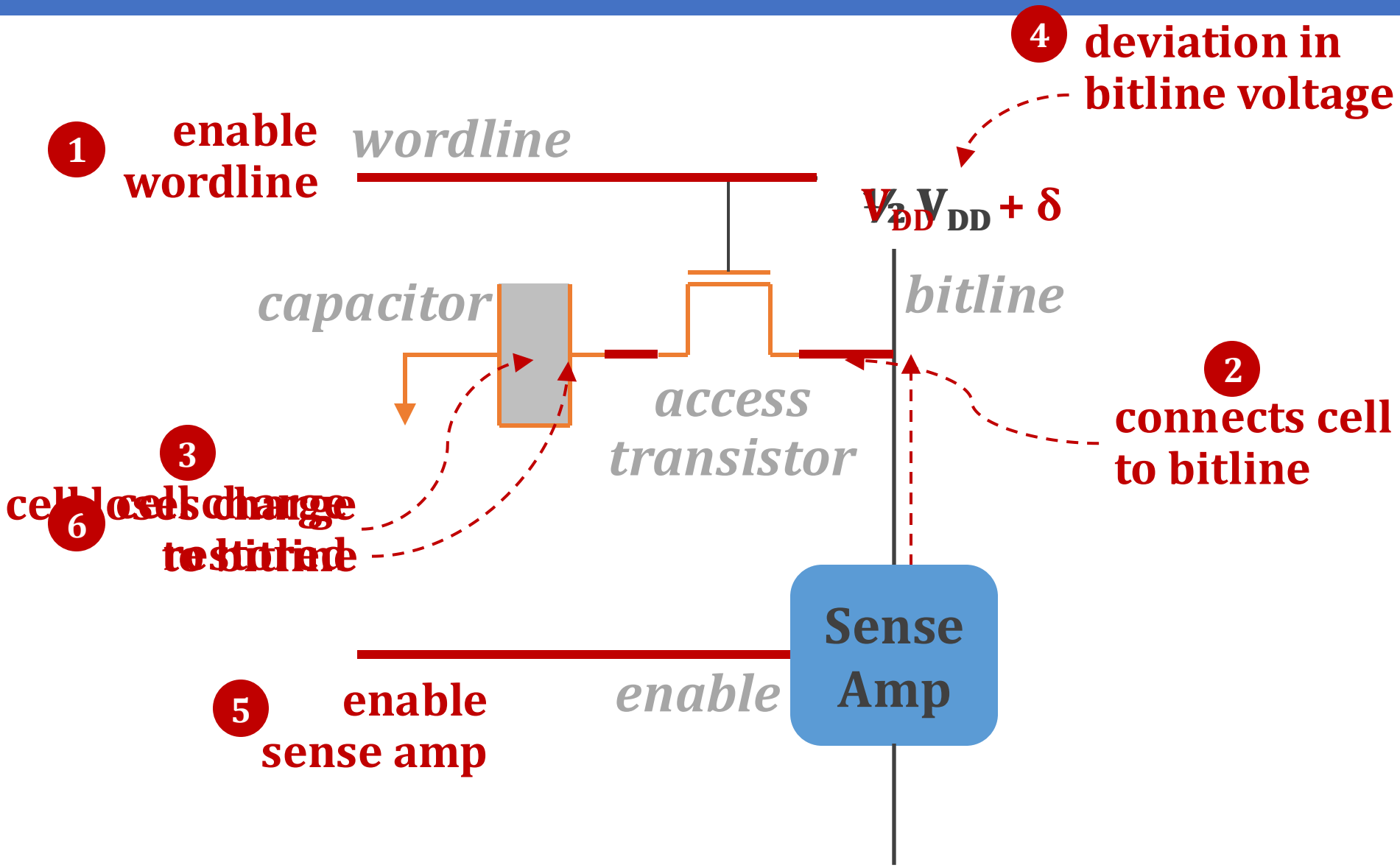
SAFARI
ETH zürich

BACKUP SLIDES

Accessing a DRAM Cell



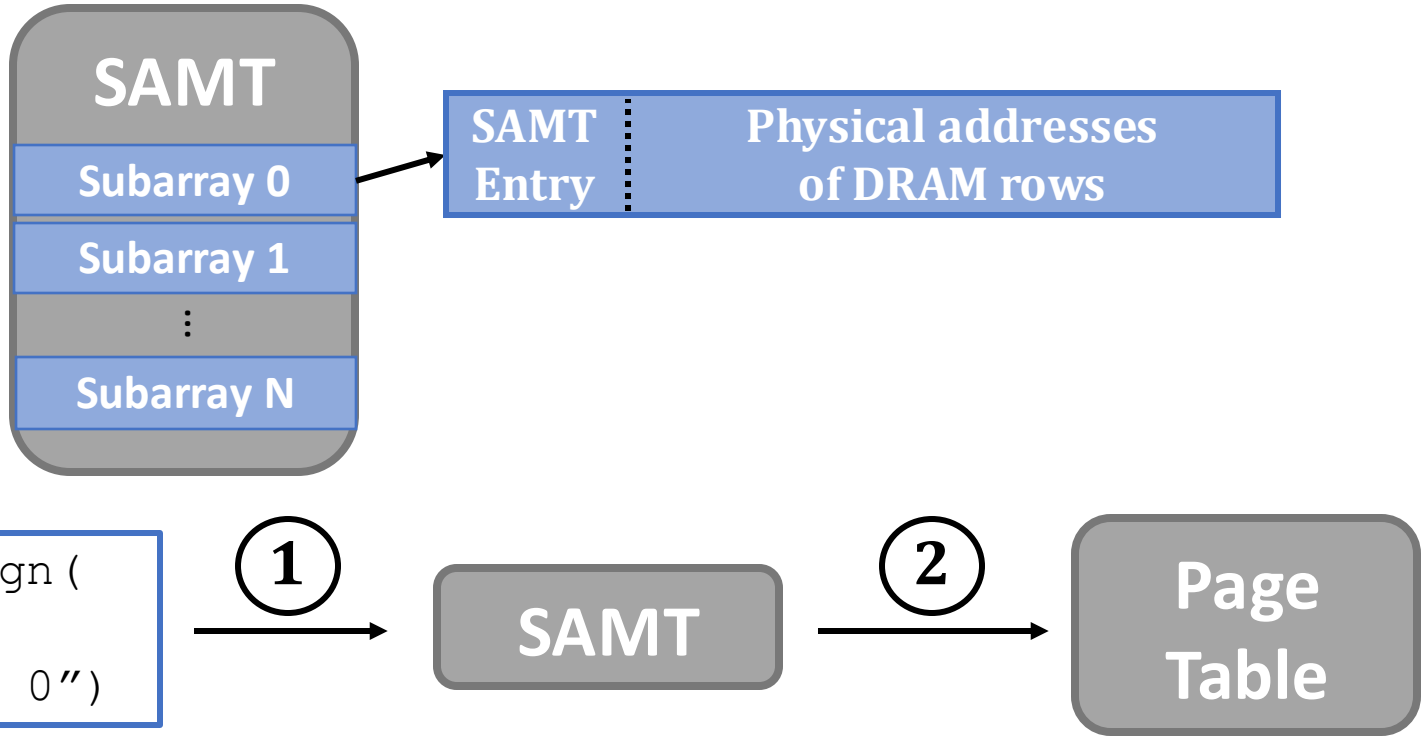
Accessing a DRAM Cell



[Seshadri+ MICRO'17]

alloc_align() function

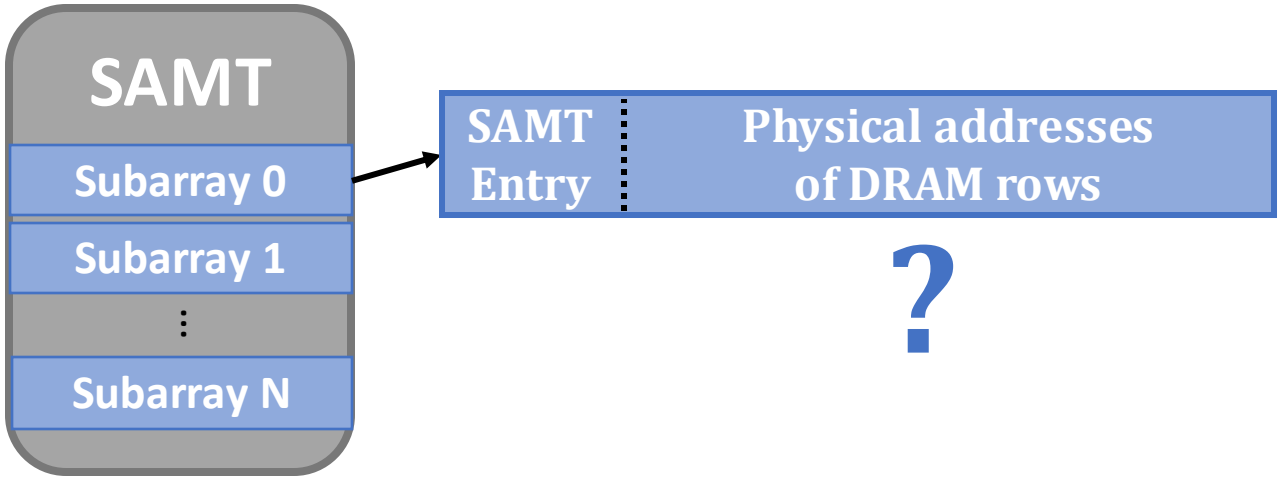
SubArray Mapping Table (SAMT) enables `alloc_align()`



1. Retrieve a physical address pointing to a DRAM row in subarray 0

2. Update the page table to map programmer-allocated address to subarray 0

Initializing SAMT



Perform in-DRAM copy using every DRAM row address as source and destination rows

If the in-DRAM copy operation succeeds source and destination rows are in the same subarray

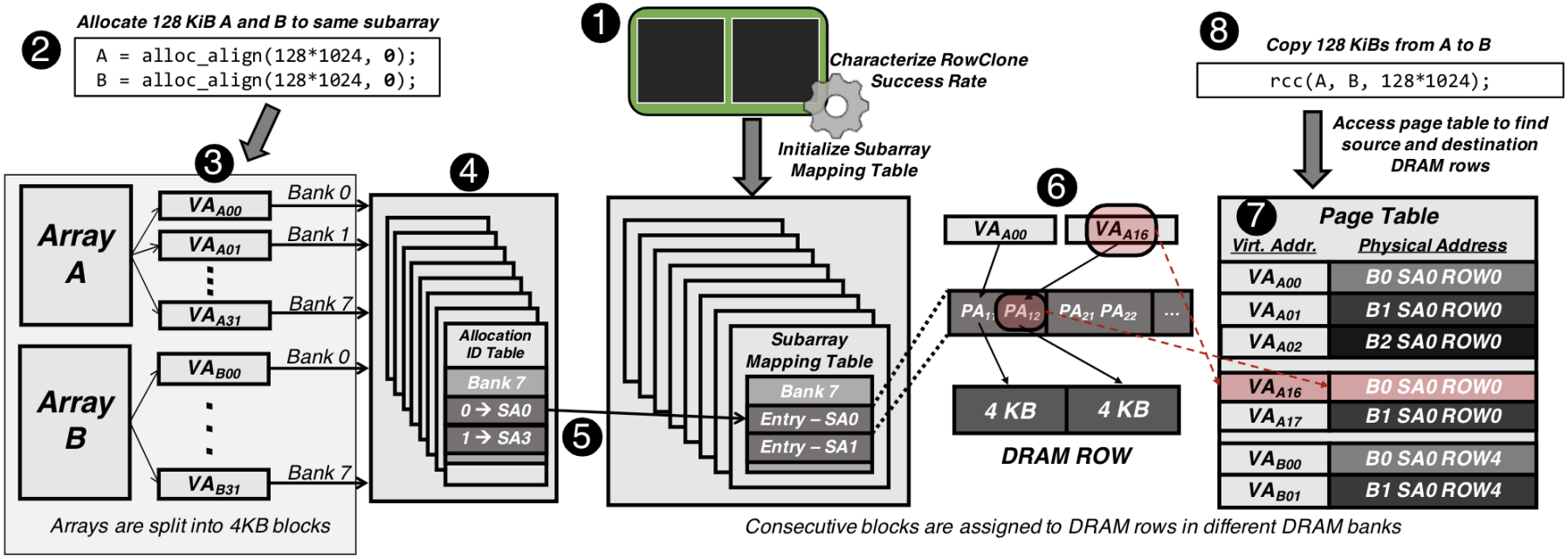

1
Characterize RowClone Success Rate
Initialize Subarray Mapping Table
2
Allocate 128 KiB A and B to same subarray
A = alloc_align(128*1024, 0);
B = alloc_align(128*1024, 0);
3
Array A
Array B
VA_A00
VA_A01
VA_A31
VA_B00
VA_B31
Bank 0
Bank 1
Bank 7
Bank 0
Bank 7
Arrays are split into 4KB blocks
4
Allocation ID Table
Bank 7
0 → SA0
1 → SA3
5
Subarray Mapping Table
Bank 7
Entry – SA0
Entry – SA1
6
VA_A00
VA_A16
PA_11
PA_12
PA_21
PA_22
...
4 KB
4 KB
DRAM ROW
7
Page Table
Virt. Addr.
Physical Address
VA_A00
B0 SA0 ROW0
VA_A01
B1 SA0 ROW0
VA_A02
B2 SA0 ROW0
VA_A16
B0 SA0 ROW0
VA_A17
B1 SA0 ROW0
VA_B00
B0 SA0 ROW4
VA_B01
B1 SA0 ROW4
8
Copy 128 KiBs from A to B
rcc(A, B, 128*1024);
Access page table to find source and destination DRAM rows
Consecutive blocks are assigned to DRAM rows in different DRAM banks

Table 1: PuM techniques that can be studied using PiDRAM. PuM techniques that we implement in this work are highlighted in bold

PuM Technique	Description	Integration Challenges
RowClone [91]	Bulk data-copy and initialization within DRAM	(i) *memory allocation and alignment mechanisms* that map source & destination operands of a copy operation into same DRAM subarray; (ii) *memory coherence*, i.e., source & destination operands must be up-to-date in DRAM.
D-RaNGe [62]	True random number generation using DRAM	(i) periodic generation of true random numbers; (ii) *memory scheduling policies* that minimize the interference caused by random number requests.
Ambit [89]	Bitwise operations in DRAM	(i) *memory allocation and alignment mechanisms* that map operands of a bitwise operation into same DRAM subarray; (ii) *memory coherence*, i.e., operands of the bitwise operations must be up-to-date in DRAM.
SIMDRAM [43]	Arithmetic operations in DRAM	(i) *memory allocation and alignment mechanisms* that map operands of an arithmetic operation into same DRAM subarray; (ii) *memory coherence*, i.e., operands of the arithmetic operations must be up-to-date in DRAM; (iii) *bit transposition*, i.e., operand bits must be laid out vertically in a single DRAM bitline.
DL-PUF [61]	Physical unclonable functions in DRAM	*memory scheduling policies* that minimize the interference caused by generating PUF responses.
QUAC-TRNG [82]	True random number generation using DRAM	(i) periodic generation of true random numbers; (ii) *memory scheduling policies* that minimize the interference caused by random number requests; (iii) efficient integration of the SHA-256 cryptographic hash function.

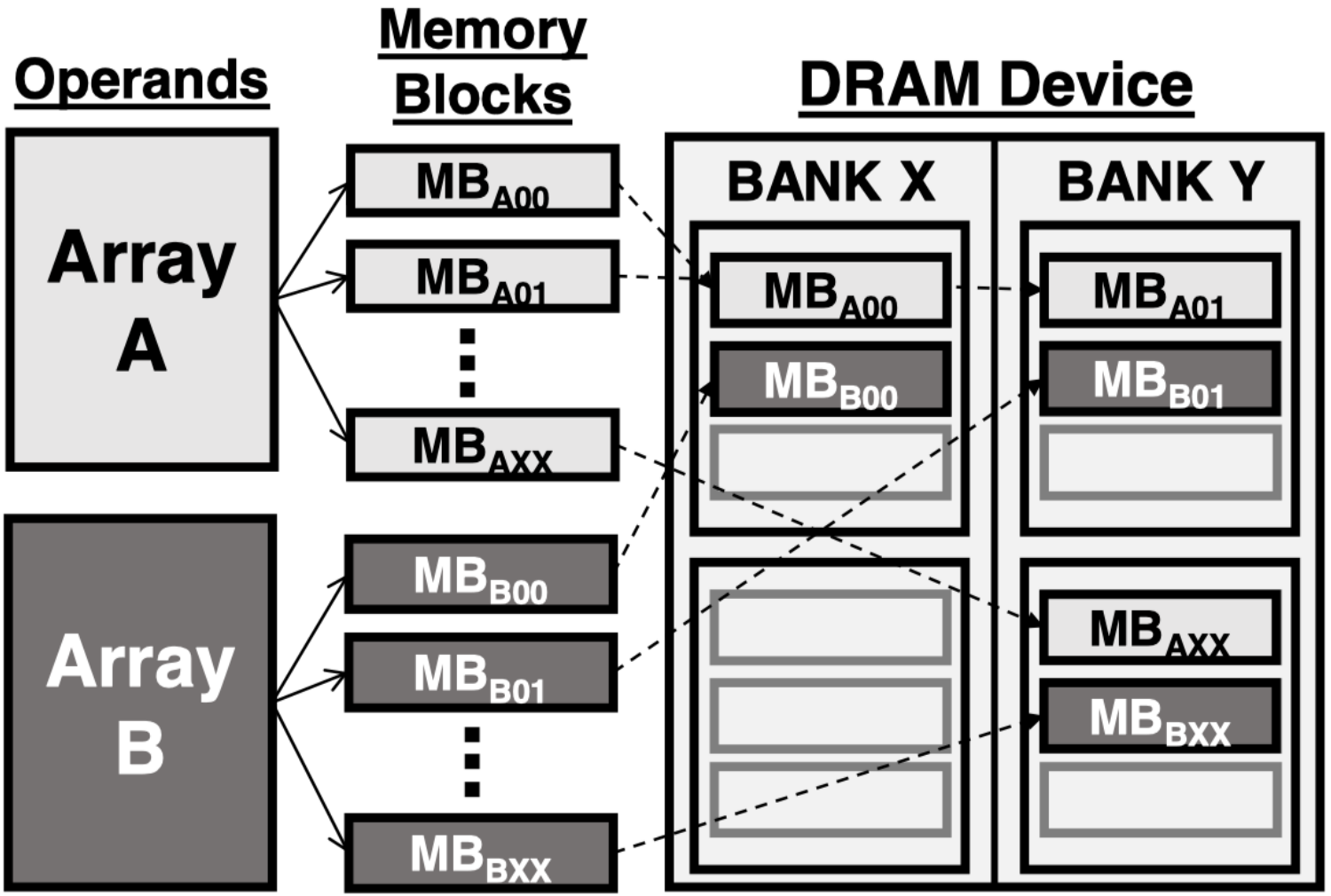


Figure 6: Overview of our memory allocation mechanism

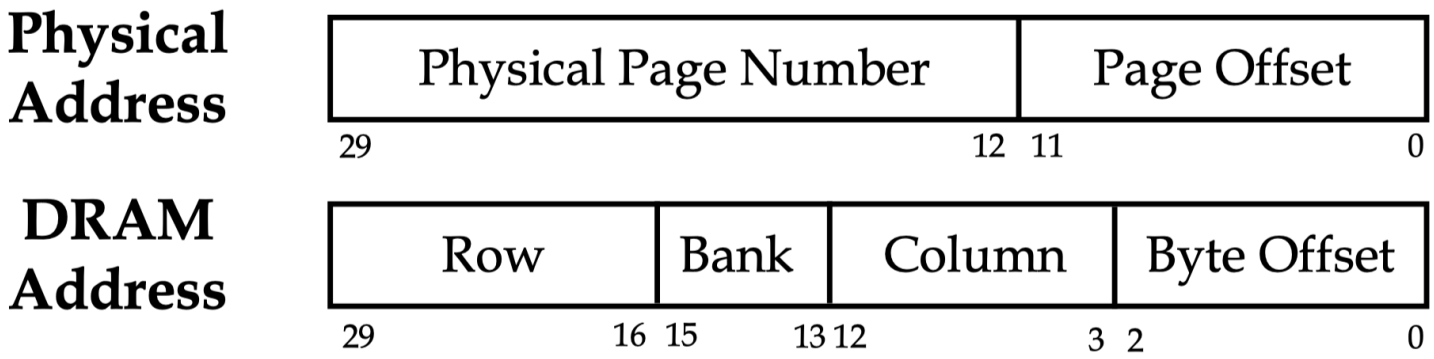


Figure 8: Physical address to DRAM address mapping in PiDRAM. Byte offset is used to address the byte in the DRAM burst.

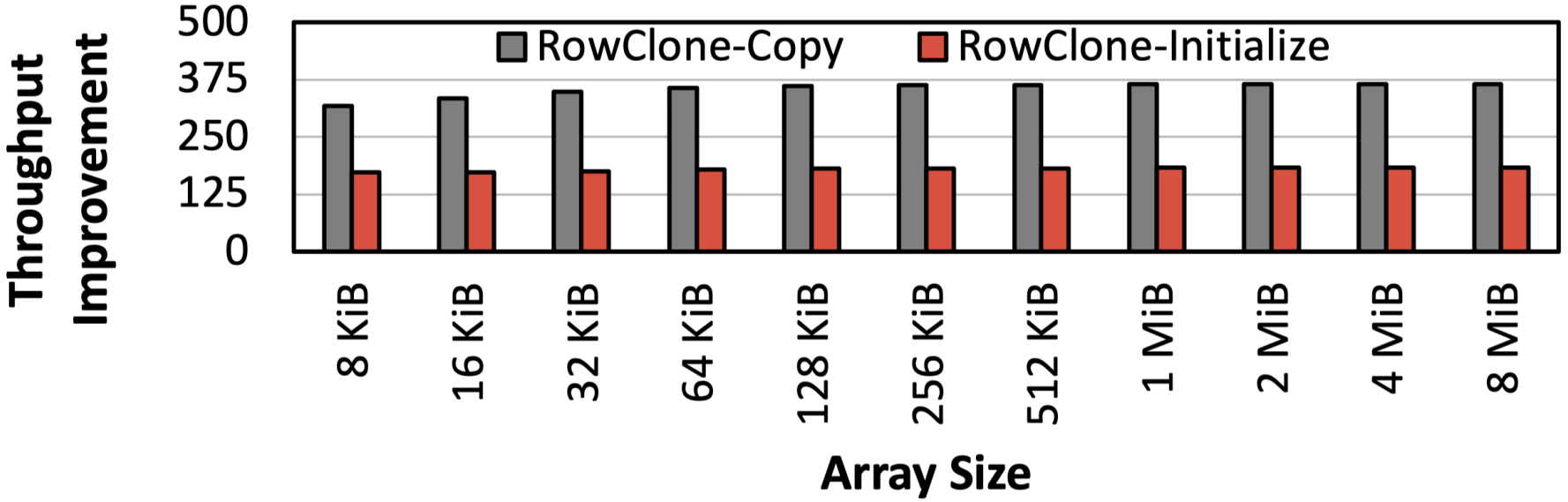


Figure 9: RowClone-Copy and RowClone-Initialize over traditional CPU-copy and -initialization for the Bare-Metal configuration

Table 4: Comparison of PiDRAM with related state-of-the-art prototyping and evaluation platforms

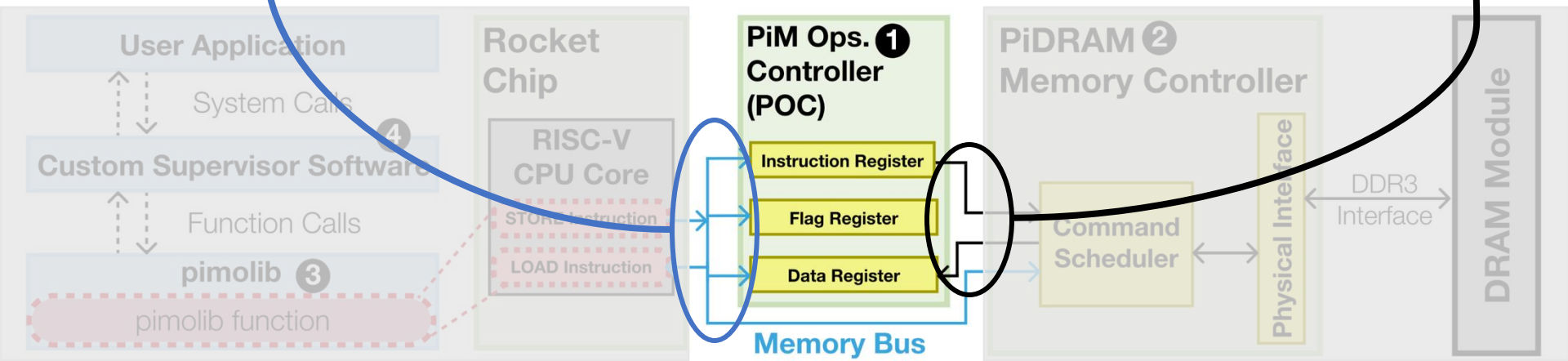
Platforms	Interface with real DRAM chips	Flexible MC for PuM	System software support	Open-source
Silent-PIM [78]	✗	✗	✓	✗
SoftMC [60]	✓(DDR3)	✗	✗	✓
ComputeDRAM [44]	✓(DDR3)	✗	✗	✗
MEG [174]	✓(HBM)	✗	✓	✓
PiMulator [119]	✗	✓	✗	✓
Commercial platforms (e.g., ZYNQ [166])	✓(DDR3/4)	✗	✓	✗
Simulators [18, 35, 90, 132, 140, 169, 170, 175]	✗	✓	✓(potentially)	✓
PiDRAM (this work)	✓(DDR3)	✓	✓	✓

PiM Operations Controller (POC)

Decode & execute PiDRAM instructions (e.g., in-DRAM copy)

Receive instructions over memory-mapped interface

Simple interface to the PiDRAM memory controller
(i) send request, (ii) wait until completion, (iii) read results



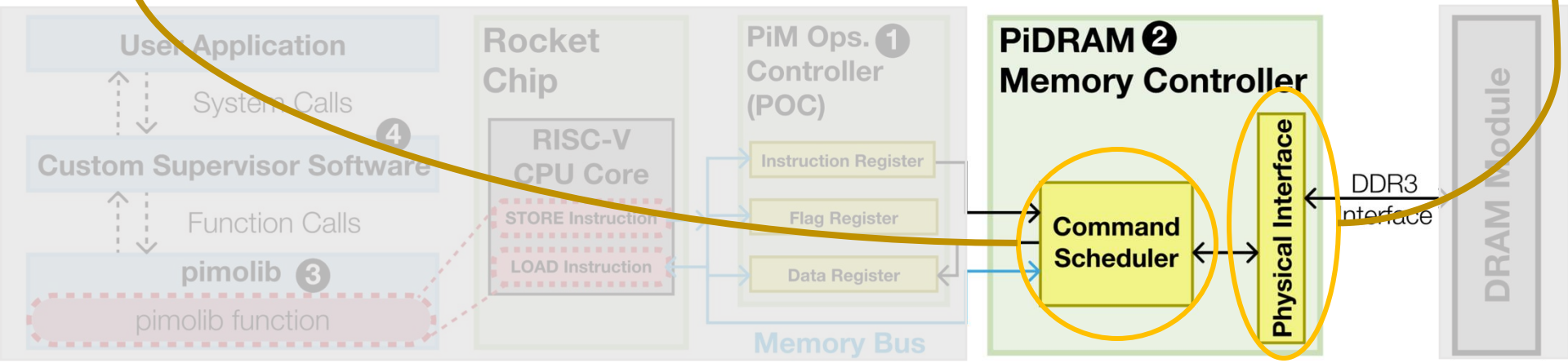
PiDRAM Memory Controller

Perform PiM operations by violating DRAM timing parameters

Support conventional memory operations (e.g., LOAD/STORE)
One state machine per operation (e.g., LOAD/STORE, in-DRAM copy)

Easily replicate a state machine to implement a new operation

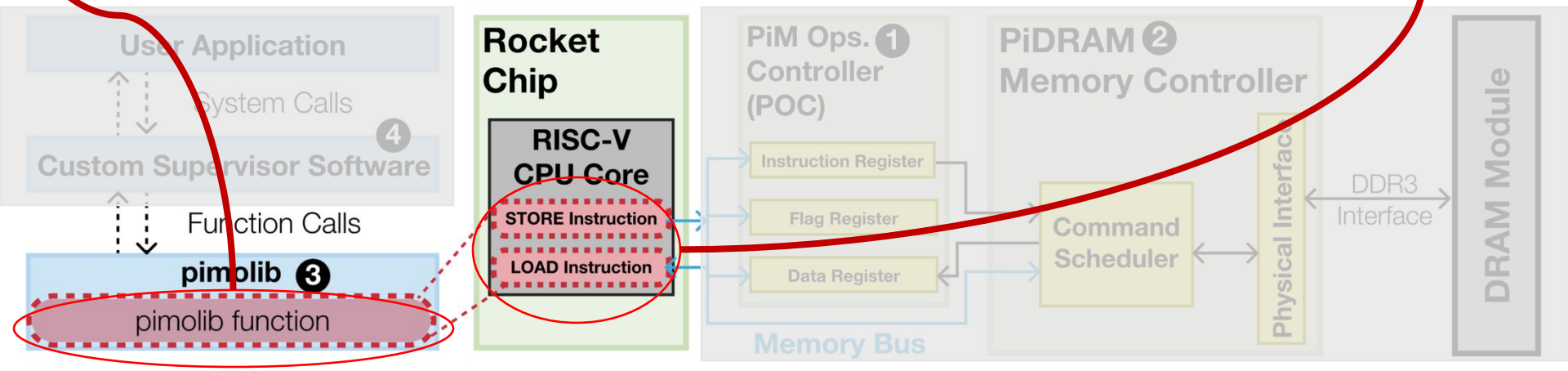
Controls the physical DDR3 interface
Receives commands from command scheduler & operates DDR3 pins



PiM Operations Library (pimolib)

Contains customizable functions that interface with the POC
Software interface for performing PiM operations

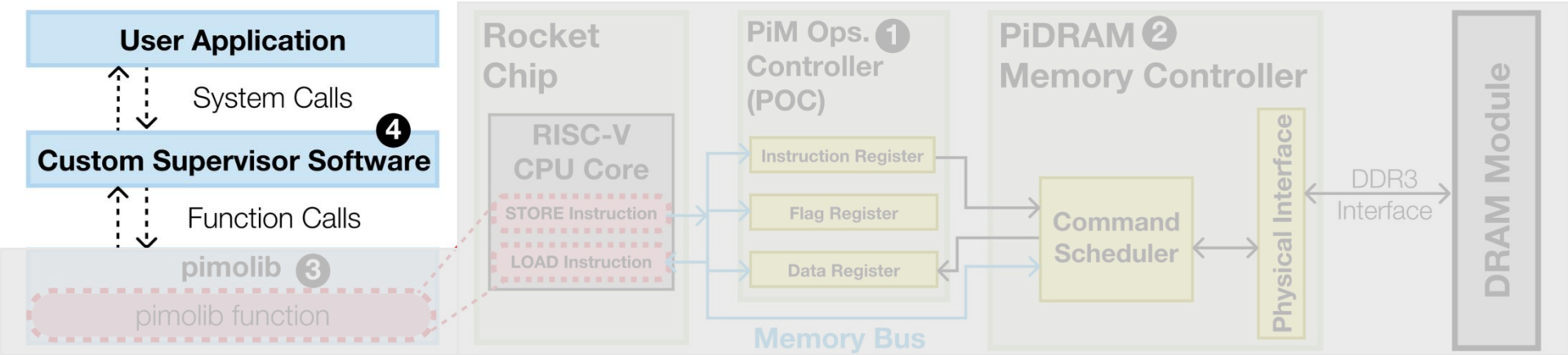
Executes LOAD & STORE requests to communicate with the POC



Custom Supervisor Software

Exposes PiM operations to the user application via system calls

Contains the necessary OS primitives to develop end-to-end PiM techniques (e.g., memory management and allocation for RowClone)



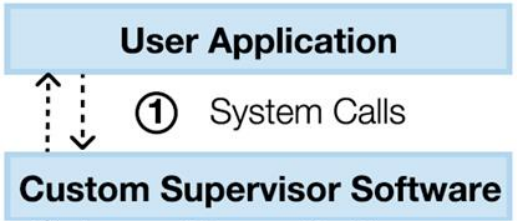
PiM Operation Execution Flow

`Copy()` function called by the user to perform a RowClone-Copy operation in DRAM

1. Application makes a system call: `Copy(A, B, N bytes)`
2. Custom Supervisor Software calls the `Copy()` pimolib function

`Copy (S, D)`

S: source DRAM row
D: destination DRAM row



PiM Operation Execution Flow

3. `Copy(S, D)` executes two store instructions in the CPU

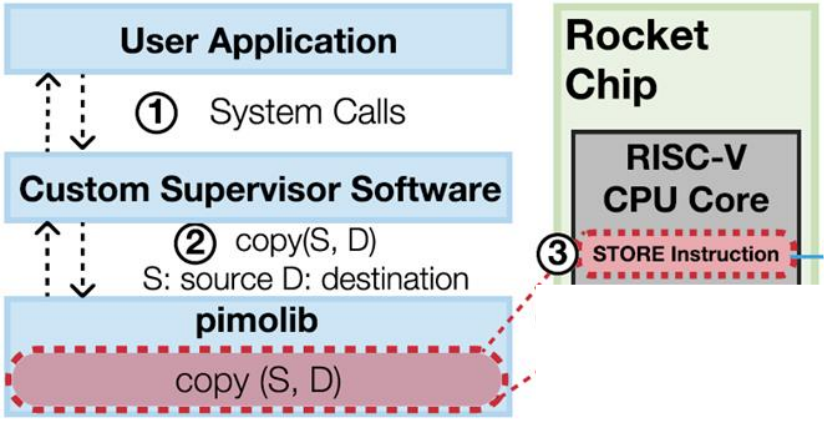
4. The first store updates the *instruction* register with `Copy(S, D)`

5. The second store sets the "Start" flag in the *flag* register

Start (S)

1

Start the execution of PiM operation

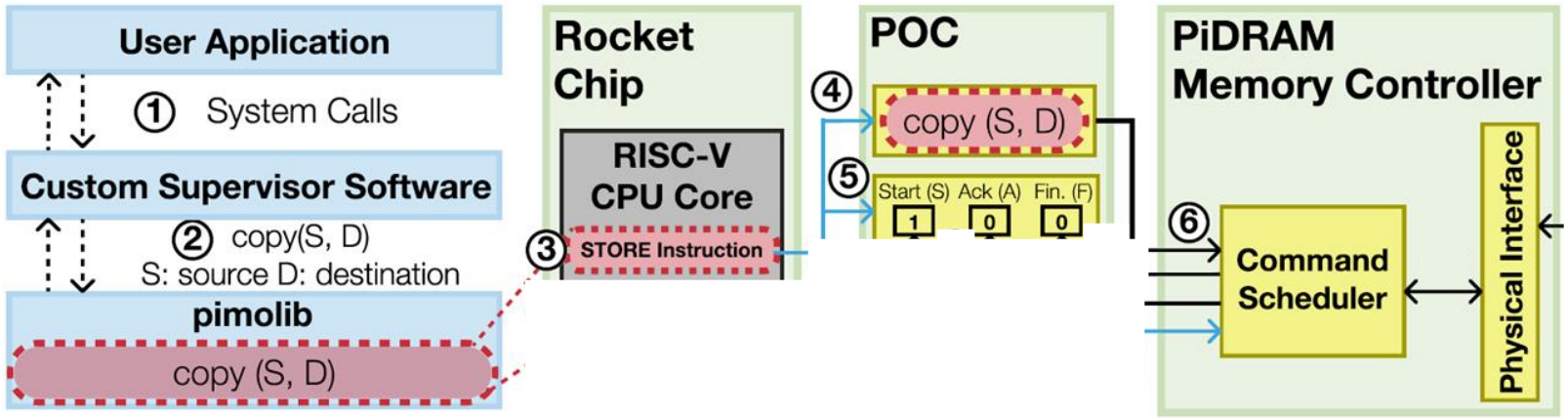


PiM Operation Execution Flow

6. **POC instructs the memory controller to perform RowClone**

7. **POC resets the "Start" flag, and sets the "Ack" flag**

8. **PiDRAM memory controller issues commands with violated timing parameters to the DDR3 module**

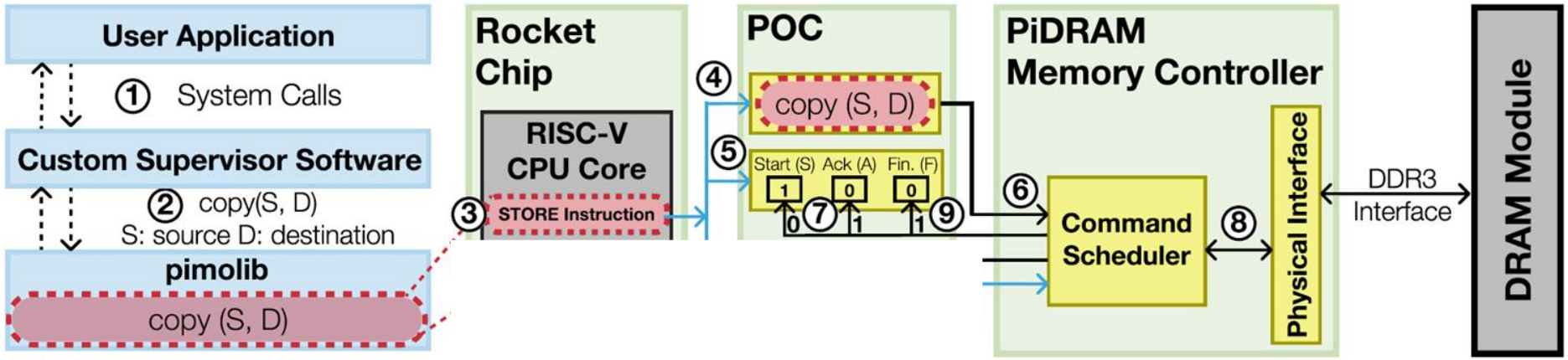


PiM Operation Execution Flow

9 The memory controller sets the "Fin." flag

10 `Copy(S, D)` periodically checks either "Ack" or "Fin." flags using LOAD instructions

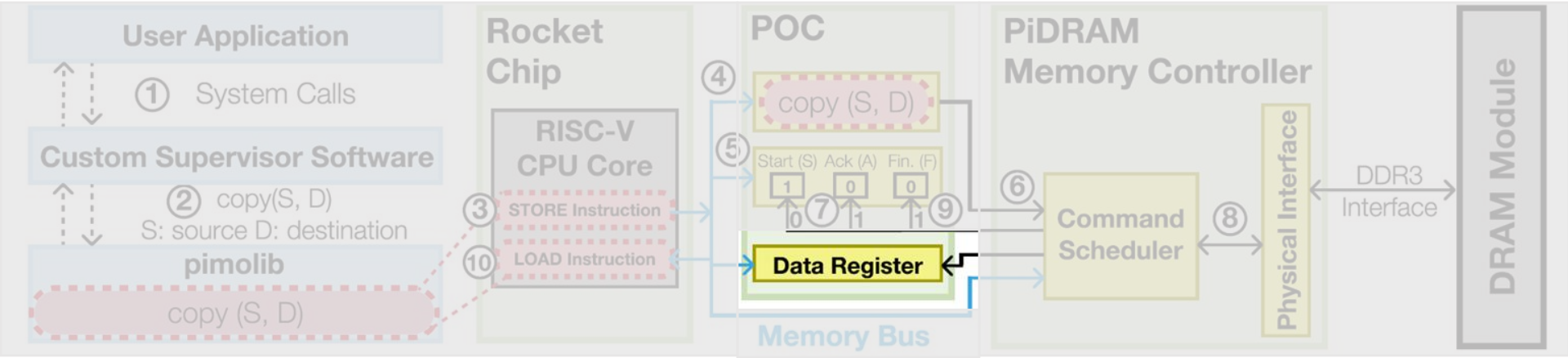
`Copy(S, D)` returns when the periodically checked flag is set



PiM Operation Execution Flow

Data Register is not used in RowClone operations because the result is stored *in memory*

It is used to read true random numbers generated by D-RaNGe



PiDRAM Components Summary

Four key components orchestrate PiM operation execution