

The 3rd
Ramulator & DRAM Bender Tutorial
DRAM Bender Overview

Ataberk Olgun

ICS 2026
6 July 2026

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*



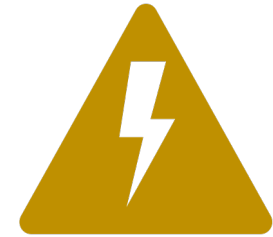
*Inter-cell
interference*



*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

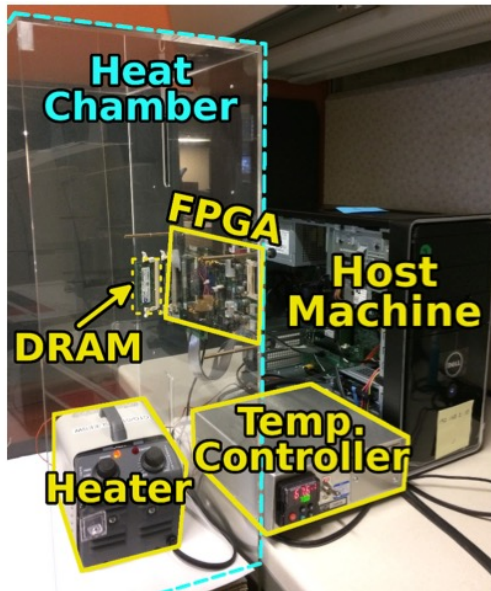
DRAM Testing Infrastructure

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++ and Python)
- **Minimal** programming effort and time
- **Accessible** to a wide range of users
 - *who may lack experience in hardware design*

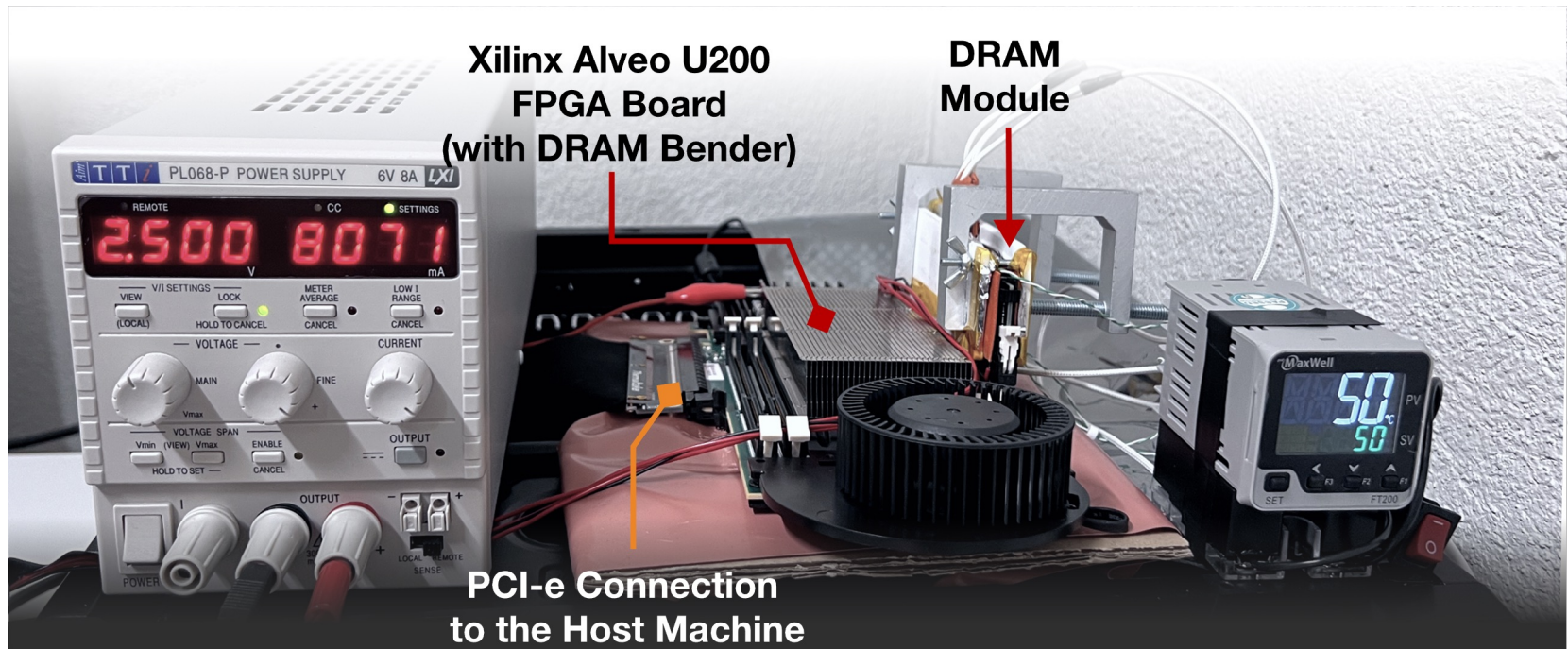
- Extensibility

- **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 C++ and Python APIs

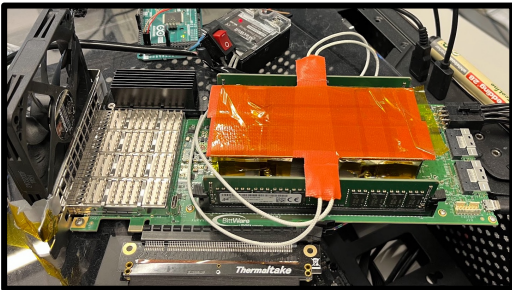


[Naglikci+, DSN'22]

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 Six Prototypes

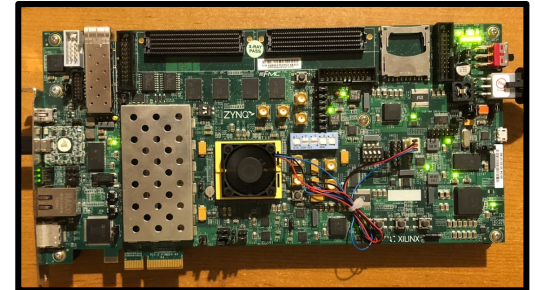
Six out of the box FPGA-based prototypes



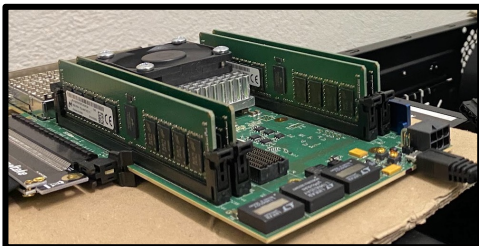
Bittware XUPVH
(DDR4 & HBM2)



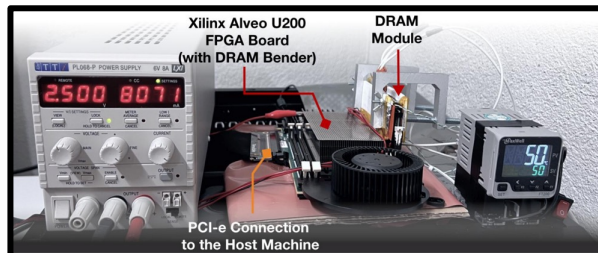
Bittware XUSP3S (DDR4)



AMD ZC706 (DDR3)



Bittware XUPP3R (DDR4)



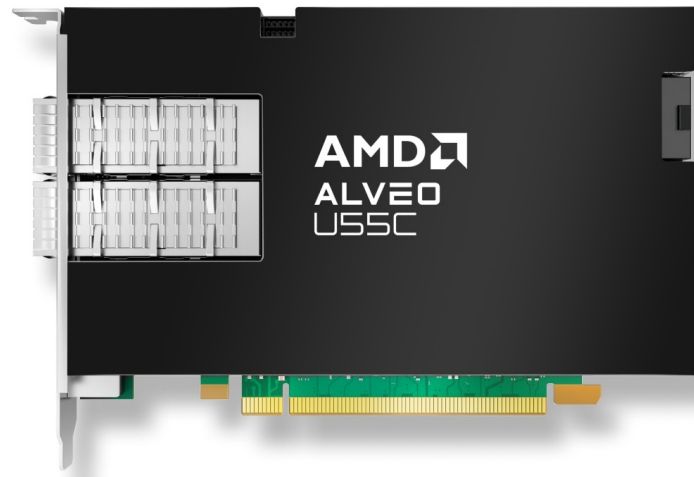
AMD Alveo U200 (DDR4)



AMD Alveo U50 (HBM2)

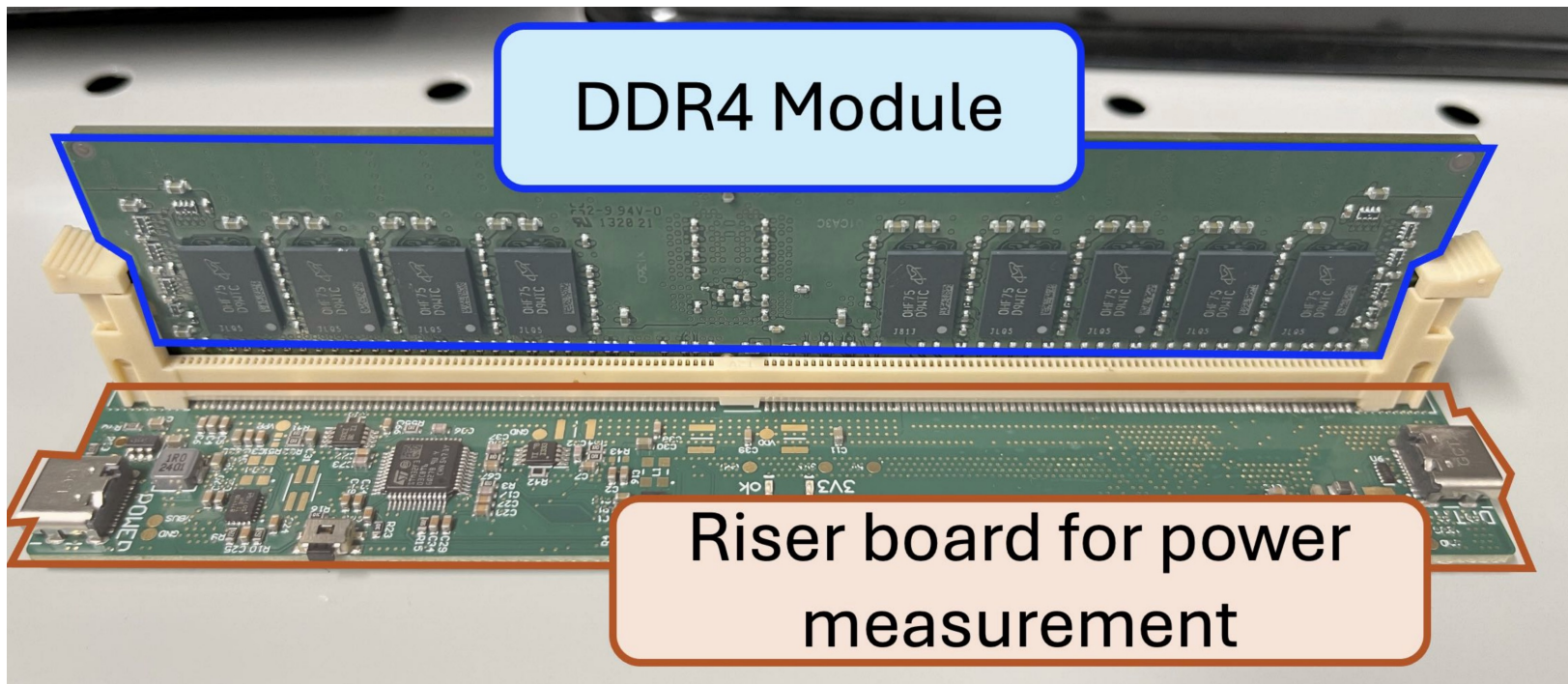
DRAM Bender: Coming Soon (I)

- Support for AMD Alveo U55C HBM2 board
- 16 GB 2x8-HI HBM2
- Enable HBM2 **power** measurement

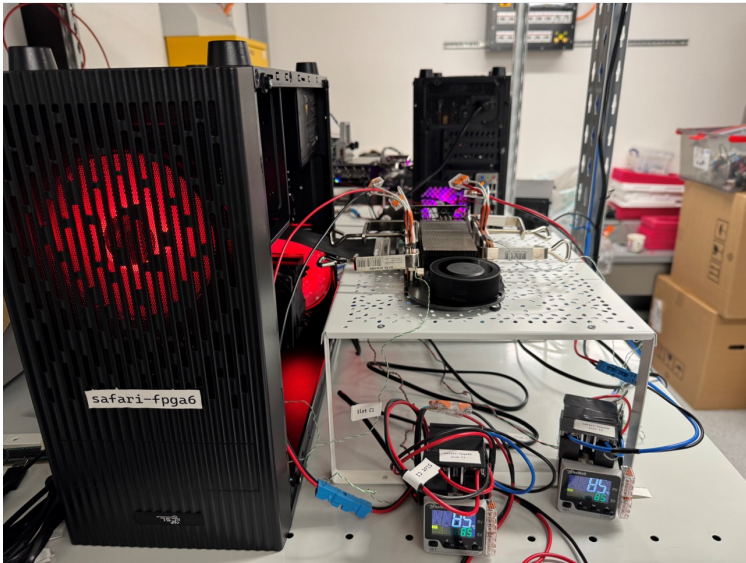


DRAM Bender: Coming Soon (II)

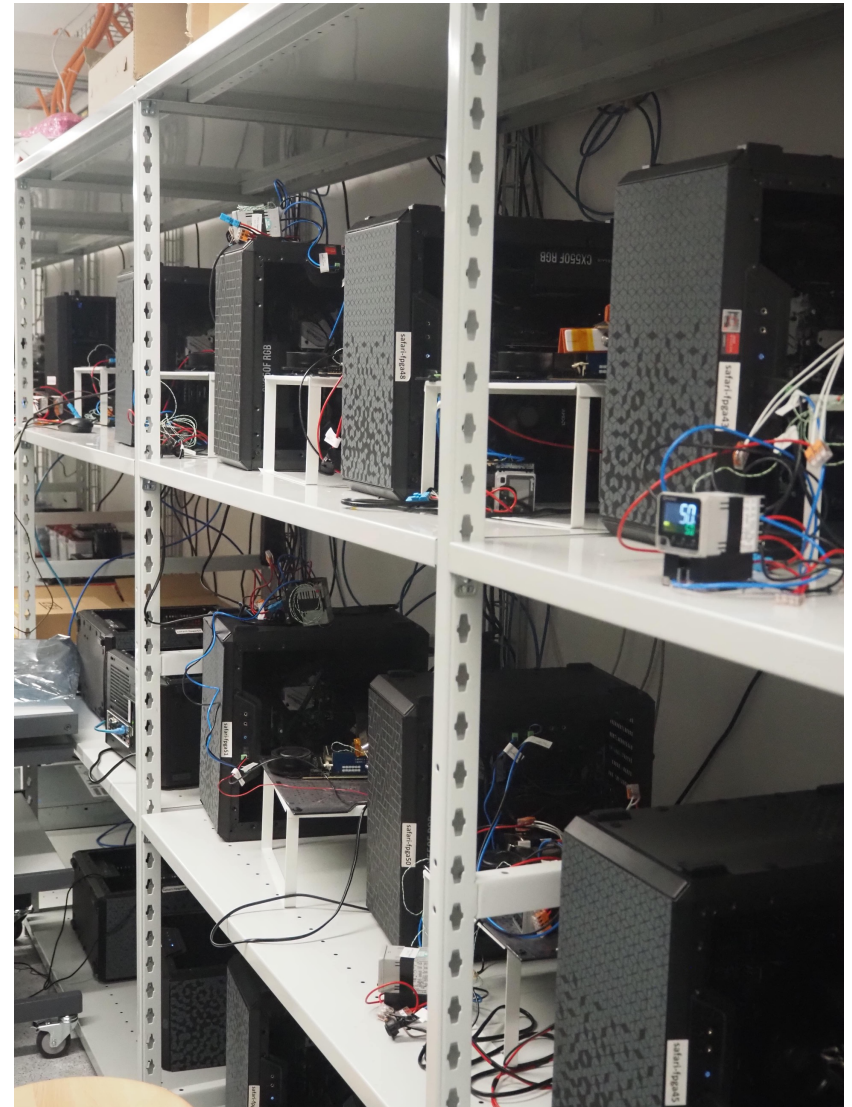
- Enable DDR4 **power** measurement
- Enable DDR4 **over/undervolting** studies



Laboratory for Understanding Memory



Laboratory for Understanding Memory



Laboratory for Understanding Memory

A Modern Large-Scale Memory Characterization Laboratory

Ataberk Olgun[§]

Haocong Luo[§]

İsmail Emir Yüksel[§]

F. Nisa Bostancı[§]

A. Giray Yağlıkcı[§]

Onur Mutlu[§]

[§]SAFARI Research Group

ETH Zürich

Real memory chip characterization yields insights into fundamental operational characteristics of modern memory, enabling new mechanisms that improve memory performance, robustness, security, and energy efficiency. We describe our large-scale DRAM characterization laboratory for understanding DRAM. A key building block of this laboratory is DRAM Bender [1, 2], a versatile and easy-to-use modern DRAM characterization infrastructure. We have updated DRAM Bender to i) introduce support for new types of characterization experiments, ii) expand on its DRAM interface standard support, and iii) make it easier to use at large scale. This paper introduces these updates for the first time. We hope our infrastructure enables the community to discover new problems and solve critical memory scaling issues, enabling the overcoming of the huge memory bottleneck that plagues modern computing systems.

and other major parameters. This infrastructure, which took us more than a year to build and which we later open sourced as SoftMC [13, 14] and DRAM Bender [1, 2], enabled us (and many others) to empirically study and understand many interesting and important characteristics of modern DRAM chips.

Many researchers and practitioners from industry and academia have been using DRAM Bender (and SoftMC) to deeply understand DRAM and solve major memory challenges [11, 15–68]. For example, DRAM Bender has enabled the discovery and study of major DRAM phenomena including RowHammer [28], RowPress [21], Variable Read Disturbance [18], ColumnDisturb [15], the DejaVu effect [57], Variable Retention Time [11, 67, 69, 70], data pattern dependence [11, 27, 71], data retention failures [11, 71], and computational capabilities of real commercial off-the-shelf

People for Understanding Memory



Konstantinos Sgouras



Ismail E. Yuksel



Umut Baser

Other People that Understand Memory

- Haocong Luo
- Jikun Wang
- Spyridon Galanopoulos
- Orhun Ecemis



DRAM Bender: Ease of Use

Easily programmable using the DRAM Bender C++ API

1. RowHammer: Repeatedly activating two DRAM rows

```
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

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

More in the Paper

- 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

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 (I)

- 1) [ICS'26] Tokuda+, "Clutch: High Performance Vector-Scalar Comparison using DRAM via Chunked Temporal Coding"
- 2) [ISCA'26] Luo+, "DejaVu: Why You Should Write to Your DRAM Rows Twice"
- 3) [ISCA'26] Tokuda+, "Altering Processing-using-DRAM Operation Results Without Changing Operands: A Study of Real DRAM Chips and Implications for Future Systems"
- 4) [DSN'26] Wang+, "ScaleDisturb: Exploiting Temporal Asymmetry to Amplify Read Disturbance in Modern DRAM Chips"
- 5) [DSN Disrupt'26] Baser+, "In-DRAM Signature Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Off-The-Shelf DRAM Chips"
- 6) [ICCD'25] Yuksel+, "[In-DRAM True Random Number Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Real DRAM Chips](#)"
- 7) [MICRO'25] Yuksel+, "[ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems](#)"
- 8) [ISCA'25] Yuksel+, "[PuDHammer: Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips](#)"
- 9) [DSN'25] Canpolat+, "[EasyDRAM: An FPGA-based Infrastructure for Fast and Accurate End-to-End Evaluation of Emerging DRAM Techniques](#)"
- 10) [VTS'25] Luo+, "[Revisiting DRAM Read Disturbance: Identifying Inconsistencies Between Experimental Characterization and Device-Level Studies](#)"
- 11) [HPCA'25] Olgun+, "[Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance](#)"
- 12) [HPCA'25] Tugrul+, "[Understanding RowHammer Under Reduced Refresh Latency: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions](#)"
- 13) [DSN'24] Olgun+, "[Read Disturbance in High Bandwidth Memory: A Detailed Experimental Study on HBM2 DRAM Chips](#)"
- 14) [DSN'24] Yuksel+, "[Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis](#)"

Research DRAM Bender Enabled (I)

- 15) [DSN'24 Disrupt] Luo+, "[An Experimental Characterization of Combined RowHammer and RowPress Read Disturbance in Modern DRAM Chips](#)"
- 16) [HPCA'24] Yaglikci+, "[Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions](#)"
- 17) [HPCA'24] Yuksel+, "[Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis](#)"
- 18) [ISCA'23] Luo+, "[RowPress: Amplifying Read Disturbance in Modern DRAM Chips](#)"
- 19) [DSN'23 Disrupt] Olgun+, "[An Experimental Analysis of RowHammer on HBM2 DRAM Chips](#)"
- 20) [arXiv Preprint, 2023] Orosa+, "[SpyHammer: Using RowHammer to Remotely Spy on Temperature](#)"
- 21) [MICRO'22] Yaglikci+, "[HIRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips](#)"
- 22) [DSN'22] Yaglikci+, "[Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices](#)"
- 23) [MICRO'21] Orosa+, "[A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses](#)"
- 24) [MICRO'21] Hassan+, "[Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications](#)"
- 25) [ISCA'21] Olgun+, "[QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips](#)"
- 26) [ISCA'21] Orosa+, "[CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations](#)"
- 27) [ISCA'20] Kim+, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)"
- 28) [S&P'20] Frigo+, "[TRRespass: Exploiting the Many Sides of Target Row Refresh](#)"
- 29) [HPCA'19] Kim+, "[D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput](#)"

Research DRAM Bender Enabled (I)

- 30) [MICRO'19] Koppula+, "[EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM](#)"
- 31) [SIGMETRICS'18] Ghose+, "[What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study](#)"
- 32) [SIGMETRICS'17] Chang+, "[Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms](#)"
- 33) [MICRO'17] Khan+, "[Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content](#)"
- 34) [SIGMETRICS'16] Chang+, "[Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization](#)"
- 35) [ISCA'14] Kim+, "[Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors](#)"
- 36) [SIGMETRICS'14] Khan+, "[The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study](#)"
- 37) [ISCA'13] Liu+, "[An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms](#)"

Even More Research DRAM Bender Enabled

- 38) [IMW'26] Jiang+, "DRAM Operations under Cryogenic Temperatures: From Device Physics to DIMM Behavior"
- 39) [JSA'26] Garzon+, "[CADM: Content addressable commodity off-the-shelf DRAM-based genome classifier](#)"
- 40) [IEEE CAL'25] Kubo+, "[PUDTune: Multi-Level Charging for High-Precision Calibration in Processing-Using-DRAM](#)"
- 41) [arXiv'25] Zhou+, "[EIM-TRNG: Obfuscating Deep Neural Network Weights with Encoding-in-Memory True Random Number Generator via RowHammer](#)"
- 42) [ASPLOS'25] Baek+, "[Marionette: A RowHammer Attack via Row Coupling](#)"
- 43) [DATE'25] Zhou+, "[Compromising the Intelligence of Modern DNNs: On the Effectiveness of Targeted RowPress](#)"
- 44) [ISCA'24] Nam+, "[DRAMScope: Uncovering DRAM Microarchitecture and Characteristics by Issuing Memory Commands](#)"
- 45) [DATE'24] Zhou+, "[DRAM-Locker: A General-Purpose DRAM Protection Mechanism Against Adversarial DNN Weight Attacks](#)"
- 46) [DRAMSec'23] Lang+, "[BLASTER: Characterizing the Blast Radius of Rowhammer](#)"
- 47) [IEEE CAL'23] Nam+, "[X-ray: Discovering DRAM Internal Structure and Error Characteristics by Issuing Memory Commands](#)"
- 48) [MICRO'22] Gao+, "[FracDRAM: Fractional Values in Off-the-Shelf DRAM](#)"
- 49) [Applied Sciences'22] Bepary+, "[DRAM Retention Behavior with Accelerated Aging in Commercial Chips](#)"
- 50) [ETS'21] Farmani+, "[RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules](#)"
- 51) [HOST'20] Talukder+, "[Towards the Avoidance of Counterfeit Memory: Identifying the DRAM Origin](#)"
- 52) [MICRO'19] Gao+, "[ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs](#)"
- 53) [IEEE Access'19] Talukder+, "[PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures](#)"
- 54) [ICCE'18] Talukder+, "[Exploiting DRAM Latency Variations for Generating True Random Numbers](#)"

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

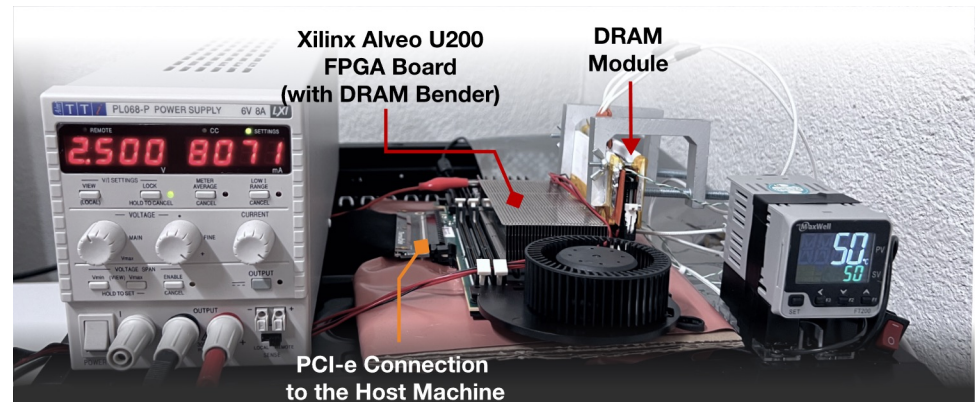
Ataberk Olgun[§] Hasan Hassan[§] A. Giray Yağlıkçı[§] Yahya Can Tuğrul^{§†}
Lois Orosa^{§⊙} Haocong Luo[§] Minesh Patel[§] Oğuz Ergin[†] Onur Mutlu[§]
 [§]*ETH Zürich* [†]*TOBB ETÜ* [⊙]*Galician Supercomputing Center*

Summary

DRAM Bender

The first **publicly-available** DDR4/3 & HBM2 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

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

The 3rd
Ramulator & DRAM Bender Tutorial
DRAM Bender Overview

Ataberk Olgun

ICS 2026
6 July 2026

The 3rd

Ramulator & DRAM Bender Tutorial

DRAM Bender Research Highlights

Ataberk Olgun

ICS 2026

6 July 2026

DRAM Bender is a Versatile Platform (I)

- DRAM Bender is widely used for:

- ① Understanding DRAM robustness problems
- ② Uncovering new DRAM functionality
- ③ Learning DRAM internals

Understanding DRAM Robustness Problems

- [ISCA'26] Luo+, "DejaVu: Why You Should Write to Your DRAM Rows Twice"
- [ISCA'26] Tokuda+, "Altering Processing-using-DRAM Operation Results Without Changing Operands: A Study of Real DRAM Chips and Implications for Future Systems"
- [DSN'26] Wang+, "ScaleDisturb: Exploiting Temporal Asymmetry to Amplify Read Disturbance in Modern DRAM Chips"
- [IMW'26] Jiang+, "DRAM Operations under Cryogenic Temperatures: From Device Physics to DIMM Behavior"
- [MICRO'25] Yuksel+, "[ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems](#)"
- [ISCA'25] Yuksel+, "[PuDHammer: Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips](#)"
- [VTS'25] Luo+, "[Revisiting DRAM Read Disturbance: Identifying Inconsistencies Between Experimental Characterization and Device-Level Studies](#)"
- [HPCA'25] Olgun+, "[Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance](#)"
- [HPCA'25] Tugrul+, "[Understanding RowHammer Under Reduced Refresh Latency: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions](#)"
- [DSN'24] Olgun+, "[Read Disturbance in High Bandwidth Memory: A Detailed Experimental Study on HBM2 DRAM Chips](#)"
- [DSN'24 Disrupt] Luo+, "[An Experimental Characterization of Combined RowHammer and RowPress Read Disturbance in Modern DRAM Chips](#)"
- [HPCA'24] Yaglikci+, "[Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions](#)"
- [ISCA'23] Luo+, "[RowPress: Amplifying Read Disturbance in Modern DRAM Chips](#)"
- [DSN'22] Yaglikci+, "[Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices](#)"
- [MICRO'21] Orosa+, "[A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses](#)"
- [ISCA'20] Kim+, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)"
- [MICRO'19] Koppula+, "[EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM](#)"
- [MICRO'17] Khan+, "[Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content](#)"
- [ISCA'14] Kim+, "[Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors](#)"
- [SIGMETRICS'14] Khan+, "[The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study](#)"
- [ISCA'13] Liu+, "[An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms](#)"
- [ASPLOS'25] Baek+, "[Marionette: A RowHammer Attack via Row Coupling](#)"
- [DATE'25] Zhou+, "[Compromising the Intelligence of Modern DNNs: On the Effectiveness of Targeted RowPress](#)"
- [DRAMSec'23] Lang+, "[BLASTER: Characterizing the Blast Radius of Rowhammer](#)"
- [Applied Sciences'22] Bepary+, "[DRAM Retention Behavior with Accelerated Aging in Commercial Chips](#)"
- [ETS'21] Farmani+, "[RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules](#)"

Uncovering New DRAM Functionality

- [ICS'26] Tokuda+, “Clutch: High Performance Vector-Scalar Comparison using DRAM via Chunked Temporal Coding”
- [DSN Disrupt'26] Baser+, “In-DRAM Signature Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Off-The-Shelf DRAM Chips”
- [ICCD'25] Yuksel+, [“In-DRAM True Random Number Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Real DRAM Chips”](#)
- [DSN'24] Yuksel+, [“Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis”](#)
- [HPCA'24] Yuksel+, [“Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis”](#)
- [MICRO'22] Yaglikci+, [“HIRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips”](#)
- [ISCA'21] Orosa+, [“CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations”](#)
- [ISCA'21] Olgun+, [“QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips”](#)
- [HPCA'19] Kim+, [“D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput”](#)
- [JSA'26] Garzon+, [“CADM: Content addressable commodity off-the-shelf DRAM-based genome classifier”](#)
- [IEEE CAL'25] Kubo+, [“PUdTune: Multi-Level Charging for High-Precision Calibration in Processing-Using-DRAM”](#)
- [arXiv'25] Zhou+, [“EIM-TRNG: Obfuscating Deep Neural Network Weights with Encoding-in-Memory True Random Number Generator via RowHammer”](#)
- [DATE'24] Zhou+, [“DRAM-Locker: A General-Purpose DRAM Protection Mechanism Against Adversarial DNN Weight Attacks”](#)
- [MICRO'22] Gao+, [“FracDRAM: Fractional Values in Off-the-Shelf DRAM”](#)
- [HOST'20] Talukder+, [“Towards the Avoidance of Counterfeit Memory: Identifying the DRAM Origin”](#)
- [MICRO'19] Gao+, [“ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs”](#)
- [IEEE Access'19] Talukder+, [“PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures”](#)
- [ICCE'18] Talukder+, [“Exploiting DRAM Latency Variations for Generating True Random Numbers”](#)

Learning proprietary DRAM details

- **[ISCA'24]** Nam+, "[DRAMScope: Uncovering DRAM Microarchitecture and Characteristics by Issuing Memory Commands](#)"
- **[IEEE CAL'23]** Nam+ "[X-ray: Discovering DRAM Internal Structure and Error Characteristics by Issuing Memory Commands](#)"
- **[MICRO'21]** Hassan+, "[Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications](#)"
- **[S&P'20]** Frigo+, "[TRRespass: Exploiting the Many Sides of Target Row Refresh](#)"

DRAM Bender is a Versatile Platform (II)

- Other noteworthy use cases:



Understanding DRAM voltage and latency sensitivity

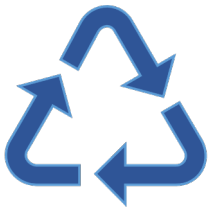
[SIGMETRICS'17] Chang+, "[Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms](#)"

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



DRAM power characterization

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



Reused in new FPGA-based evaluation infrastructures

[DSN'25] Canpolat+, "[EasyDRAM: An FPGA-based Infrastructure for Fast and Accurate End-to-End Evaluation of Emerging DRAM Techniques](#)"

DRAM Bender

Enables Understanding

DRAM Robustness Problems

Variable Read Disturbance

- Ataberk Olgun, Nisa Bostanci, Ismail Emir Yuksel, Abdullah Giray Yaglikci, Geraldo Francisco de Oliveira, Haocong Luo, Oguzhan Canpolat, Minesh Patel, and Onur Mutlu,

"Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance"

Proceedings of the 31st International Symposium on High-Performance Computer Architecture (HPCA), Las Vegas, NV, USA, March 2025.

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

Variable Read Disturbance:

An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

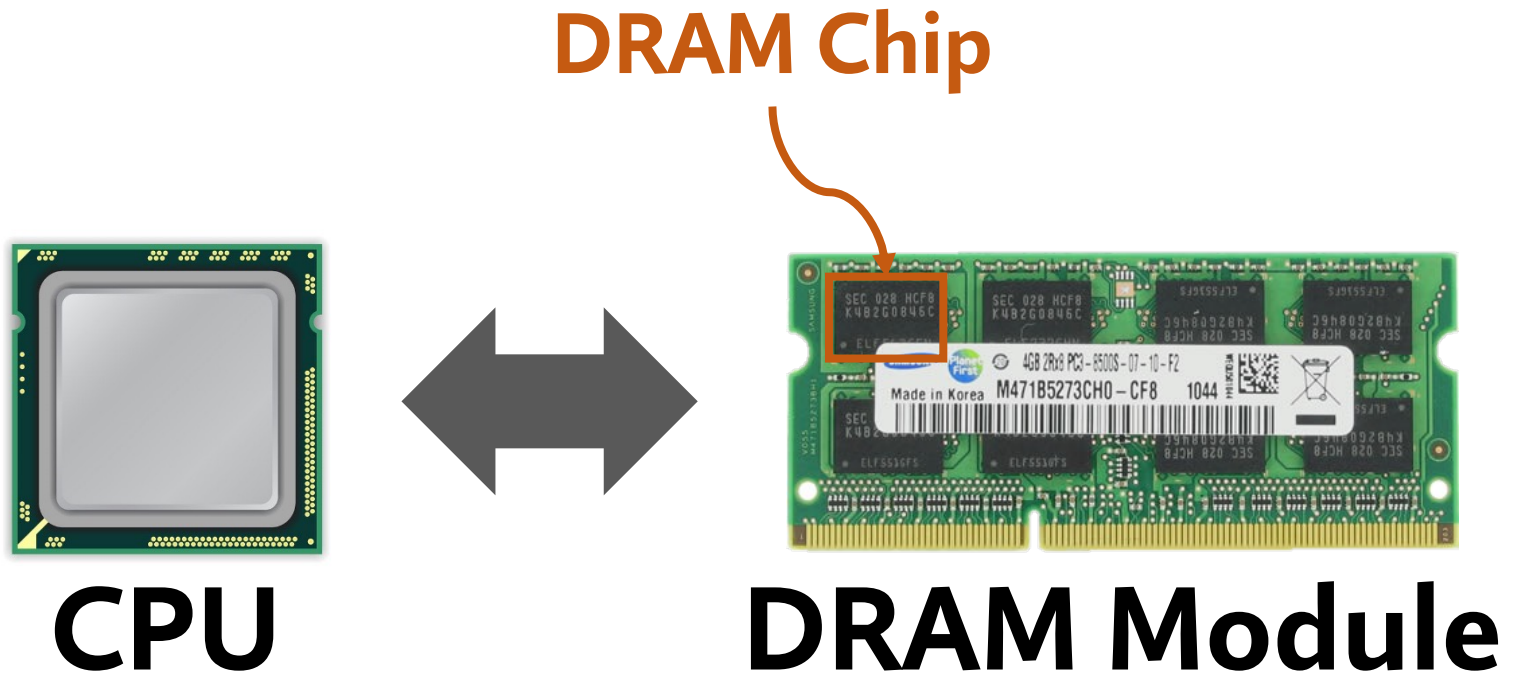
Ataberk Olgun† F. Nisa Bostanci† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†

ETH Zurich† *Rutgers University*‡

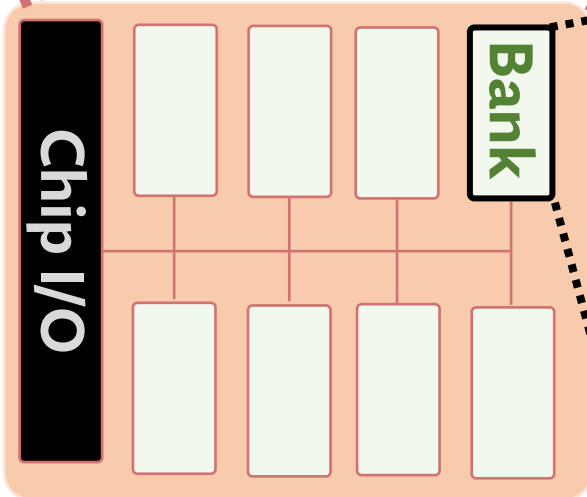
Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive char-

row) many times (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

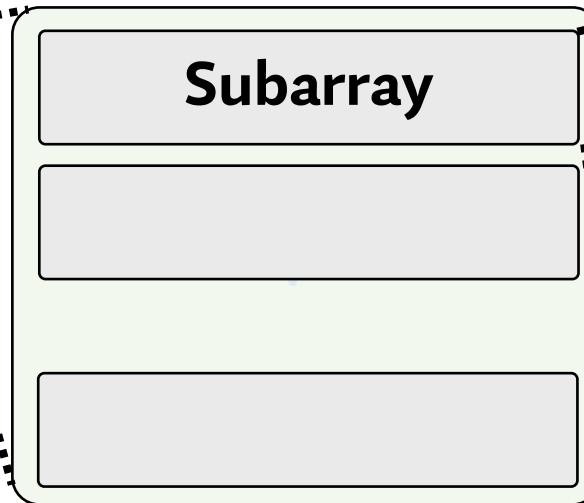
A Typical DRAM-Based Computing System



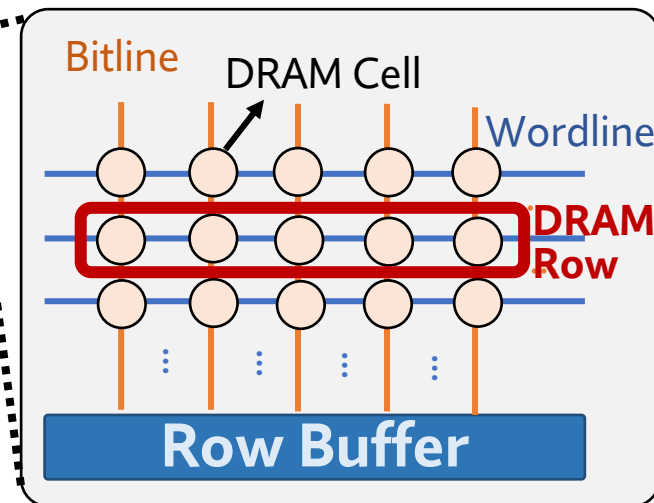
DRAM Organization



DRAM Chip



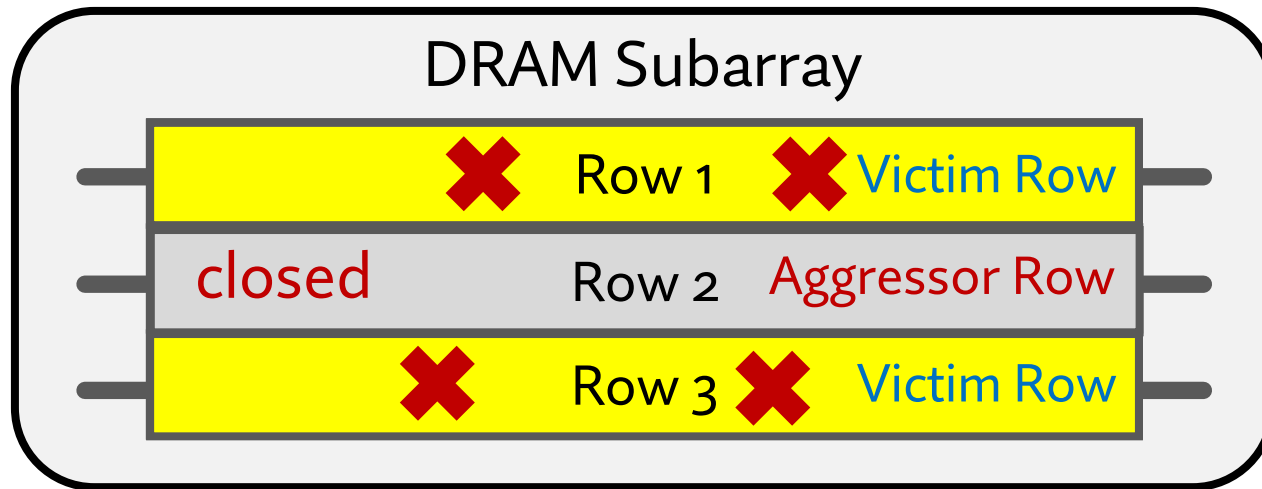
DRAM Bank



DRAM Subarray

Read Disturbance in DRAM (I)

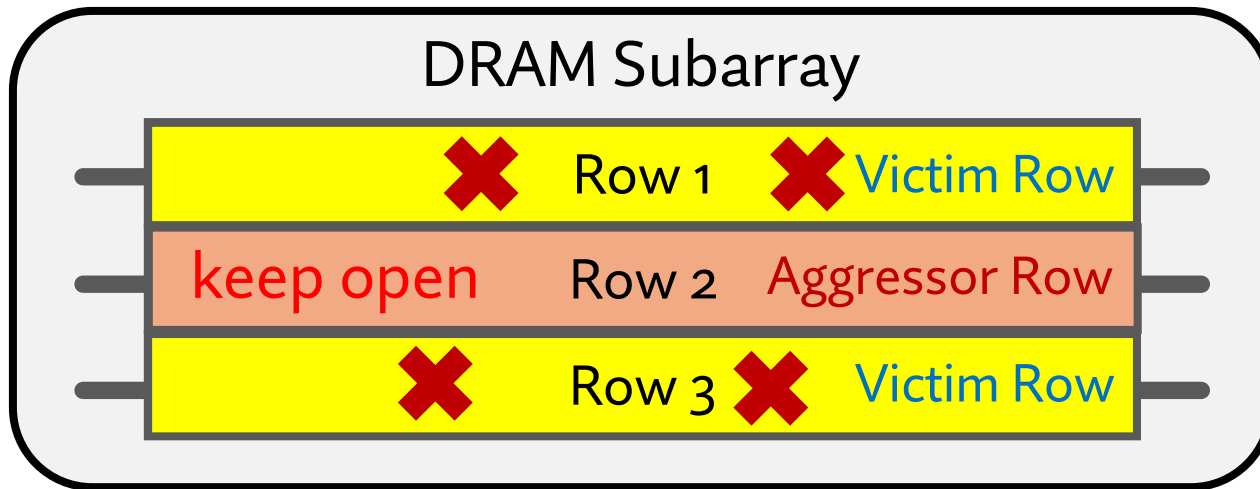
- Read disturbance in DRAM **breaks memory isolation**
- Prominent example: **RowHammer**



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

Read Disturbance in DRAM (II)

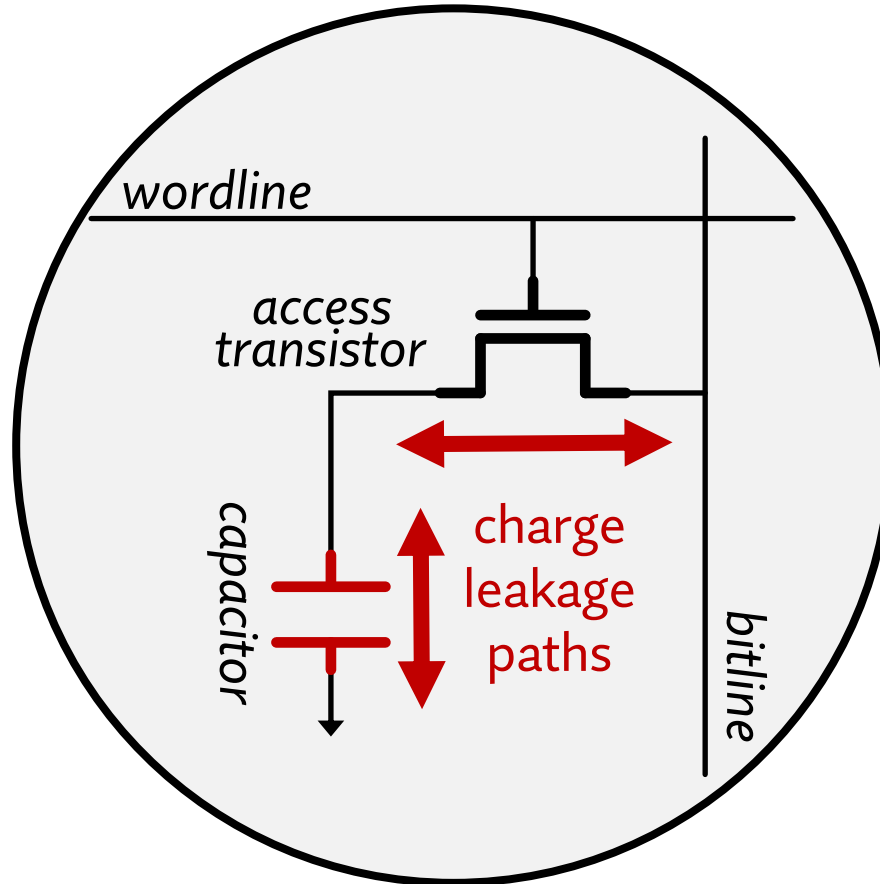
- Read disturbance in DRAM **breaks memory isolation**
- A new read disturbance phenomenon: **RowPress**



Keeping a DRAM row **open for a long time** causes bitflips in adjacent rows

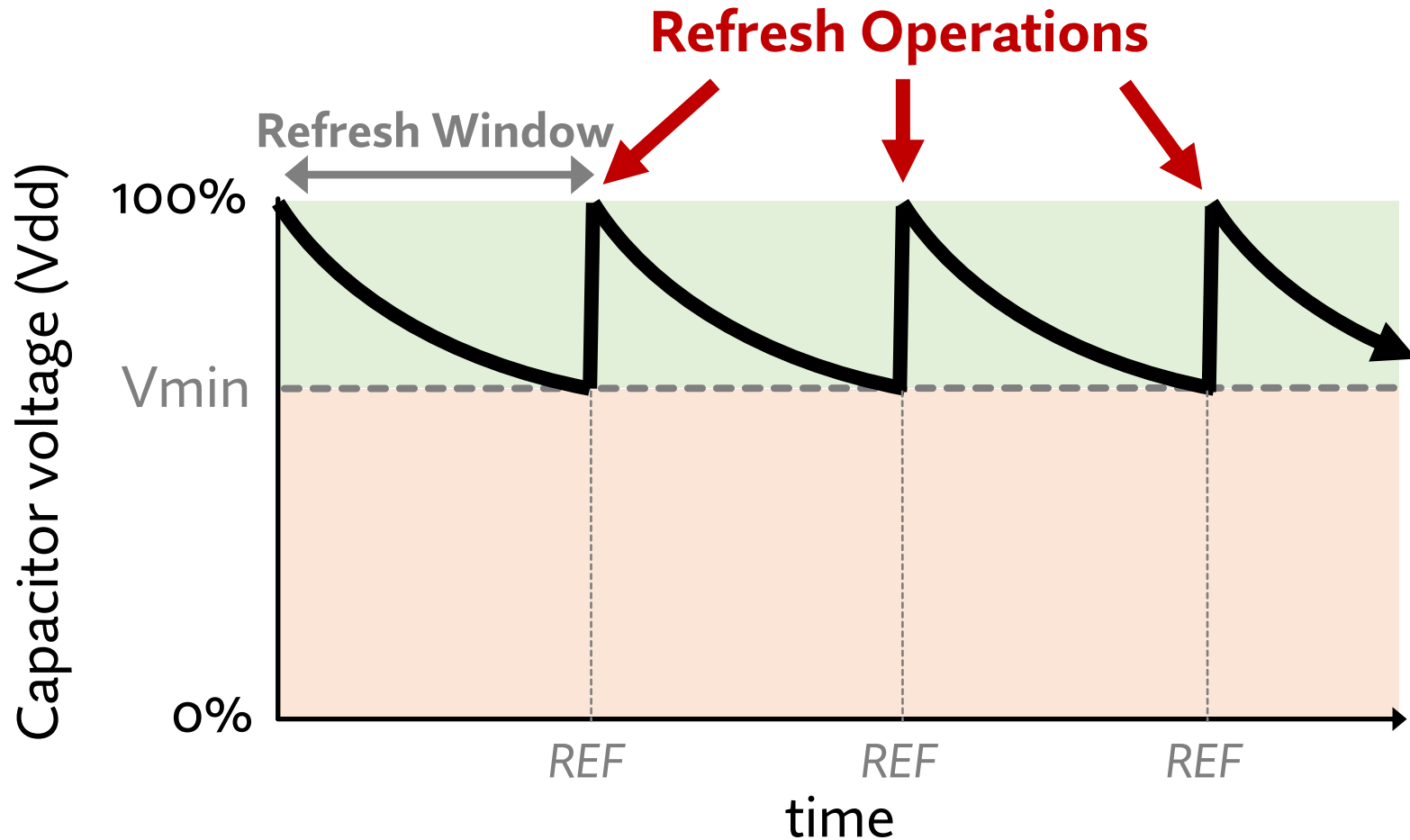
DRAM Cell Leakage

Each cell encodes information in **leaky** capacitors



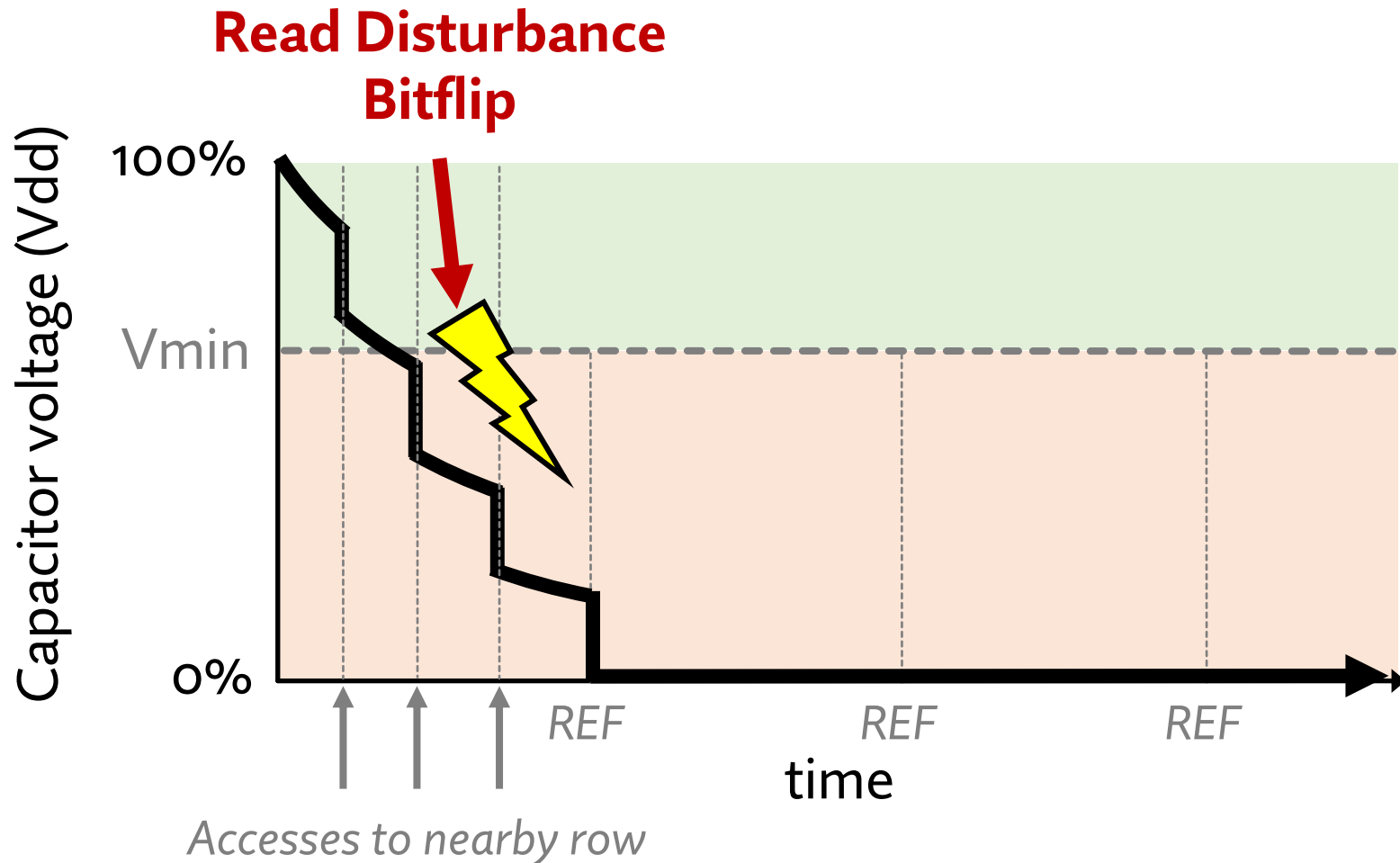
Stored data is **corrupted** if too much charge leaks (i.e., the capacitor voltage degrades too much)

DRAM Refresh



Periodic **refresh operations** preserve stored data

Read Disturbance Bitflips



Read Disturbance Solutions

There are many solutions to mitigate read disturbance bitflips

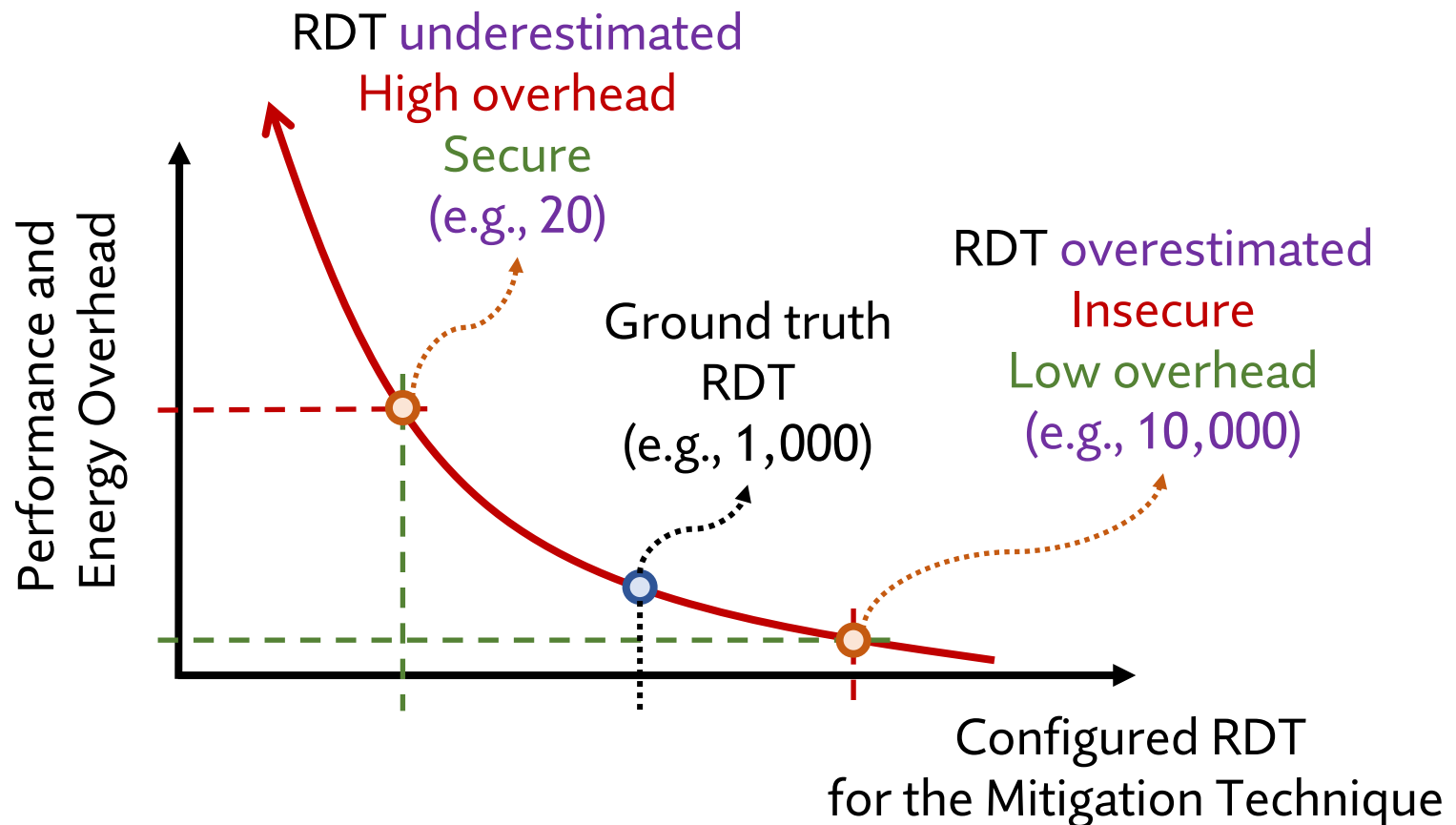
- More robust DRAM chips and/or error-correcting codes
- Increased refresh rate
- Physical isolation
- Row remapping
- Preventive refresh
- Proactive throttling

Each solution offers a different system design point in reliability, performance, energy, and area tradeoff space

The Read Disturbance Threshold (RDT)

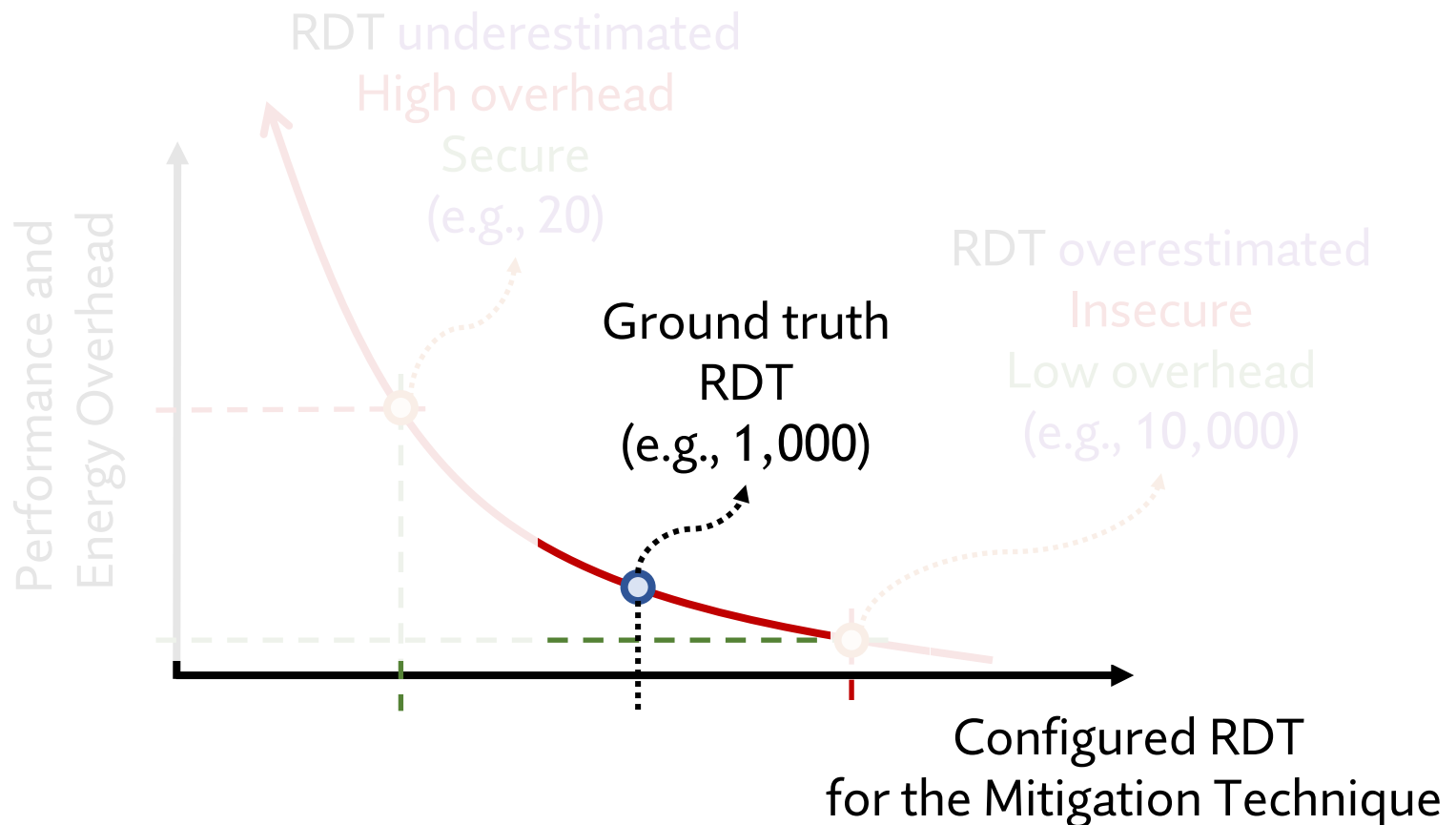
- Many **secure** read disturbance solutions take a **preventive action** before a bitflip manifests
 - E.g., **preventively refresh** a victim row
- Must **accurately quantify** the **amount of disturbance** that a row can **withstand** before experiencing a bitflip
 - Typically identified by **testing** for read disturbance failures
- **Read Disturbance Threshold (RDT):**
The **number** of aggressor row **activations** needed to induce the first bitflip

Accurate Identification of Read Disturbance Threshold is Critical for System Security and Performance



To **securely** prevent bitflips at **low overhead** RDT must be **accurately identified** and carefully configured

Motivation



Prior works assume that the **ground truth** Read Disturbance Threshold (RDT) **can be identified**

Our Goal

Answer two research questions:

- 1 Does RDT **change** over time?
- 2 How **reliably** and **efficiently** can RDT be **measured**?

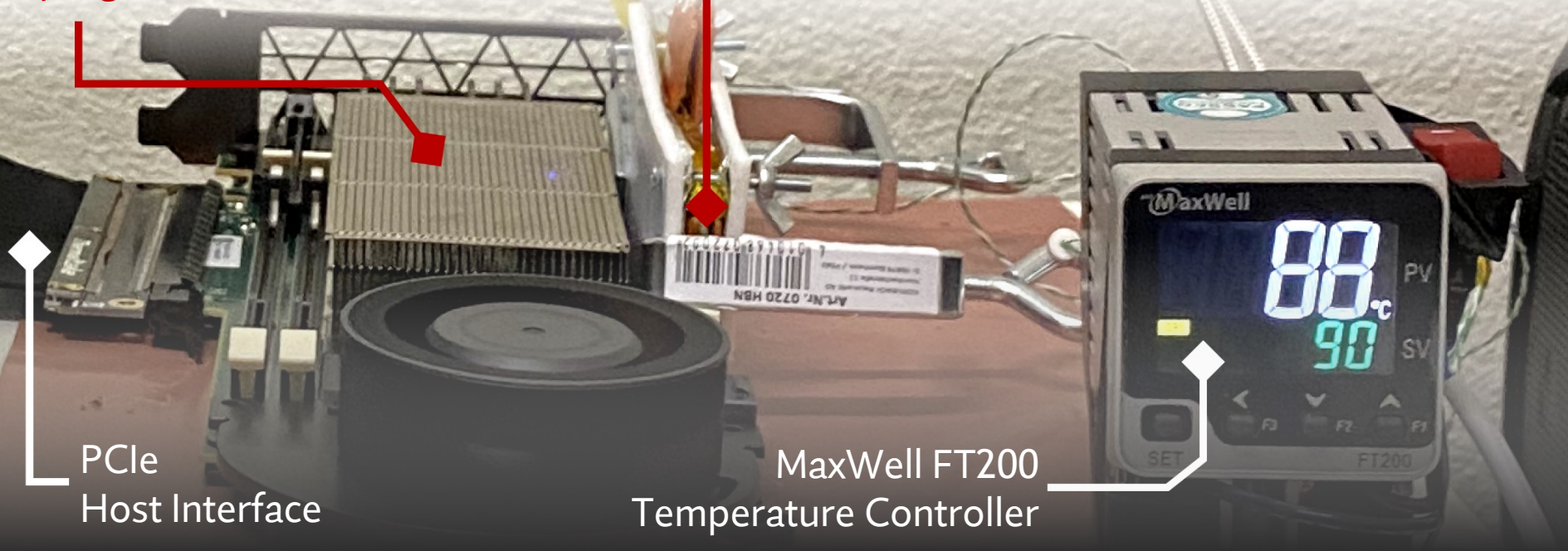
Analyze implications for
read disturbance solutions

DDR4 DRAM Testing Infrastructure

DRAM Bender on a Xilinx Virtex UltraScale+ XCU200

Xilinx Alveo U200 FPGA Board
(programmed with DRAM Bender*)

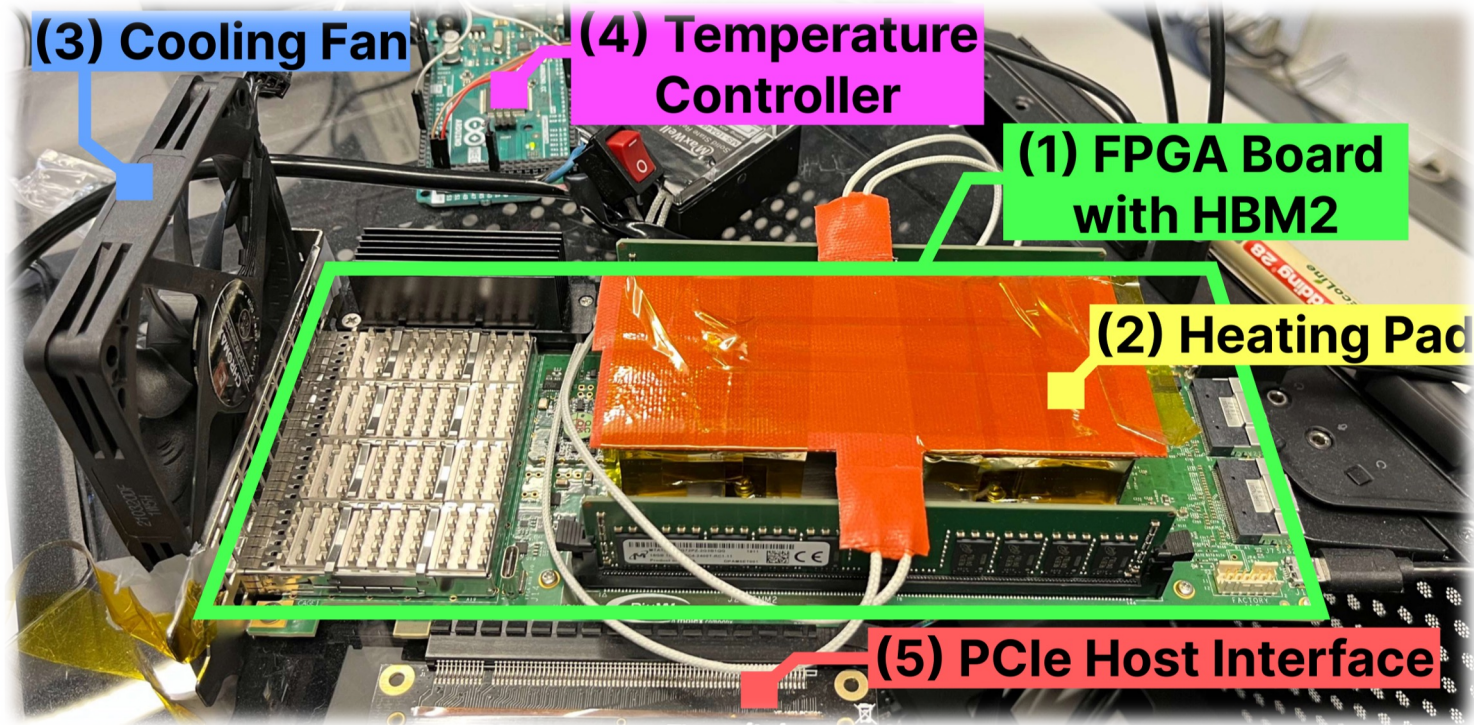
DRAM Module with Heaters



Fine-grained control over DRAM commands,
timing parameters ($\pm 1.5\text{ns}$), and temperature ($\pm 0.5^\circ\text{C}$)

HBM2 DRAM Testing Infrastructure

DRAM Bender on a Bittware XUPV VH



Fine-grained control over DRAM commands, timing parameters ($\pm 1.67\text{ns}$), and temperature ($\pm 0.5^\circ\text{C}$)

Tested DRAM Chips

160 DDR4 and 4 HBM2 Chips from SK Hynix, Micron, Samsung

Mfr.	DDR4 Module	# of Chips	Density Die Rev.	Chip Org.	Date (ww-yy)
Mfr. H (SK Hynix)	H0	8	8Gb – J	x8	N/A
	H1	8	16Gb – C	x8	36-21
	H2	8	8Gb – A	x8	43-18
	H3, H4	8	8Gb – D	x8	38-19
	H5, H6	8	8Gb – D	x8	24-20
Mfr. M (Micron)	M0	4	16Gb – E	x16	46-20
	M1	8	16Gb – F	x8	37-22
	M2	8	16Gb – F	x8	37-22
	M3, M4	8	8Gb – R	x8	12-24
	M5	8	8Gb – R	x8	10-24
	M6	8	16Gb – F	x8	12-24
Mfr. S (Samsung)	S0	8	8Gb – C	x8	N/A
	S1	8	8Gb – B	x8	53-20
	S2	8	8Gb – D	x8	10-21
	S3	8	16Gb – A	x8	20-23
	S4	4	4Gb – C	x16	19-19
	S5, S6	8	16Gb – B	x16	15-23
Mfr. S (Samsung)	HBM2 Chip Chip0 – Chip3	4	N/A	N/A	N/A

Foundational Results: Key Takeaway

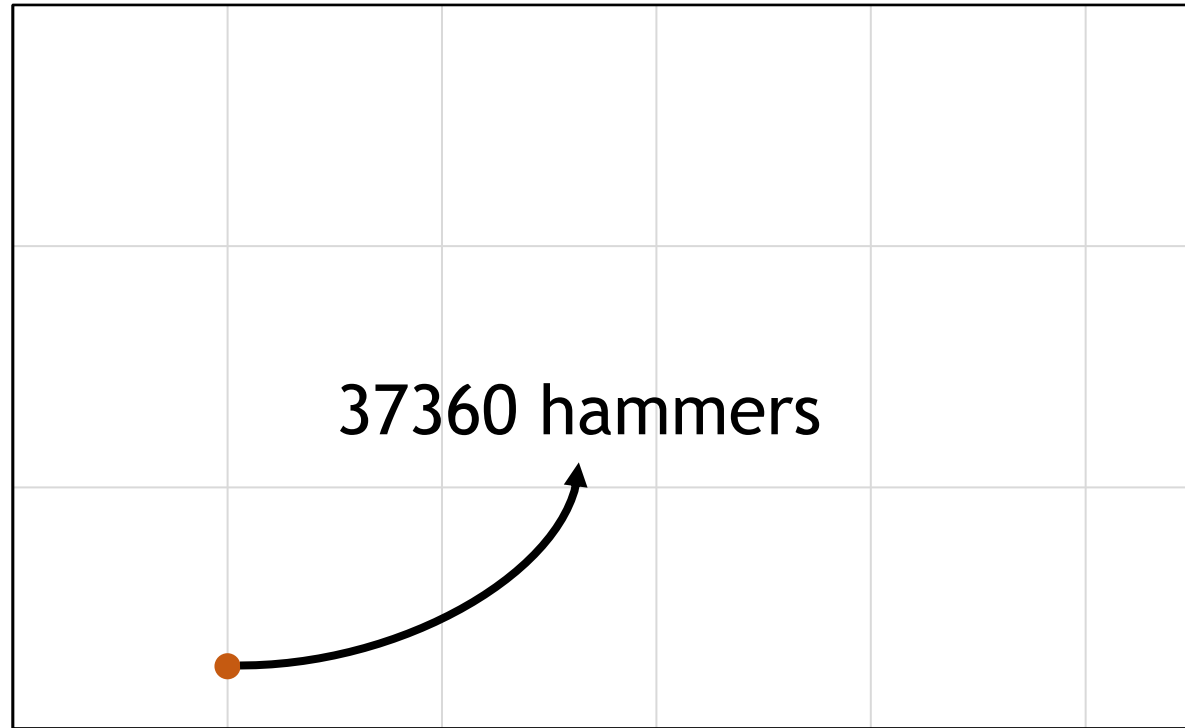
Key Takeaway

The Read Disturbance Threshold (RDT) of a row
changes randomly and unpredictably over time

Accurately identifying RDT is challenging

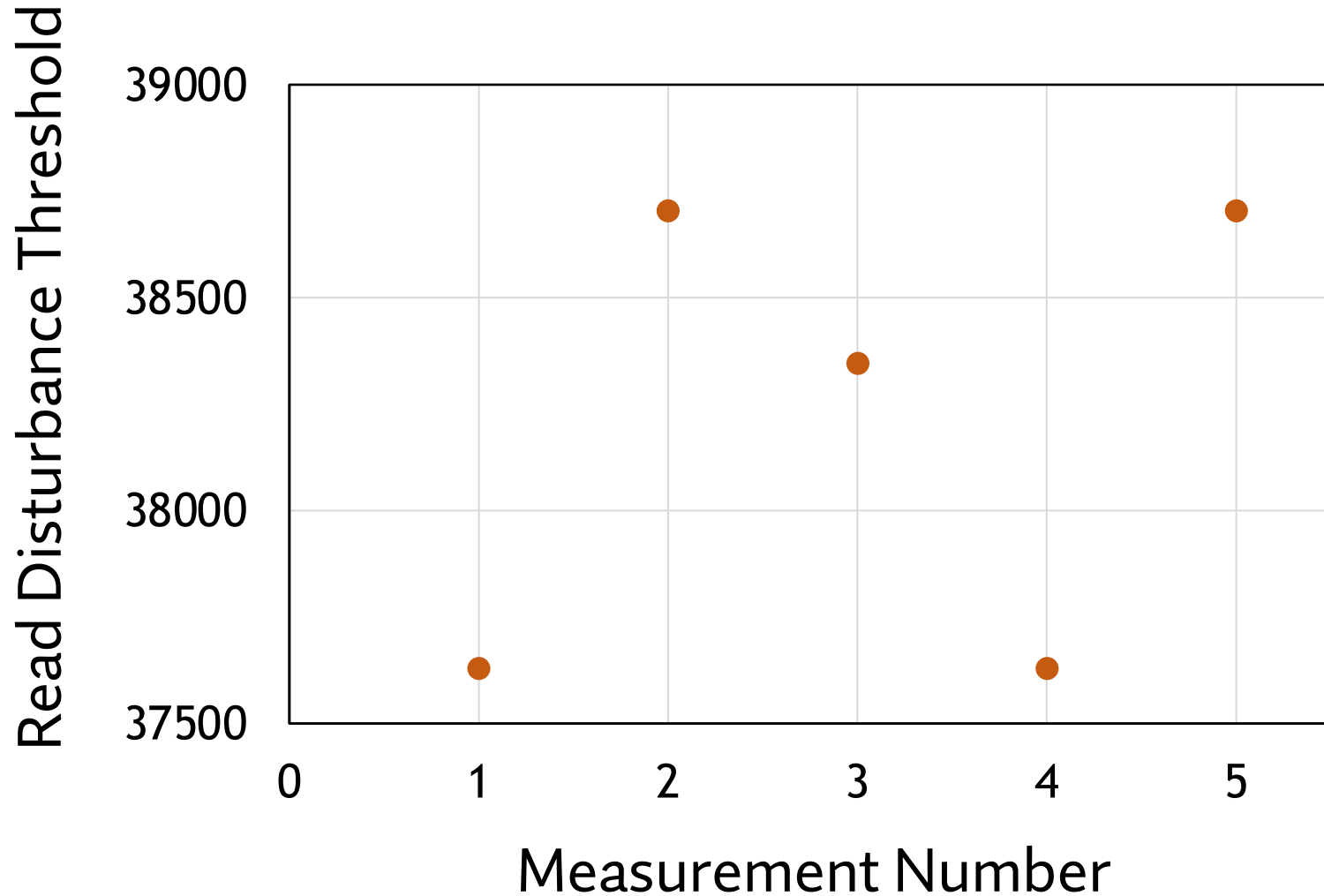
Read Disturbance Threshold Changes Over Time

Read Disturbance Threshold

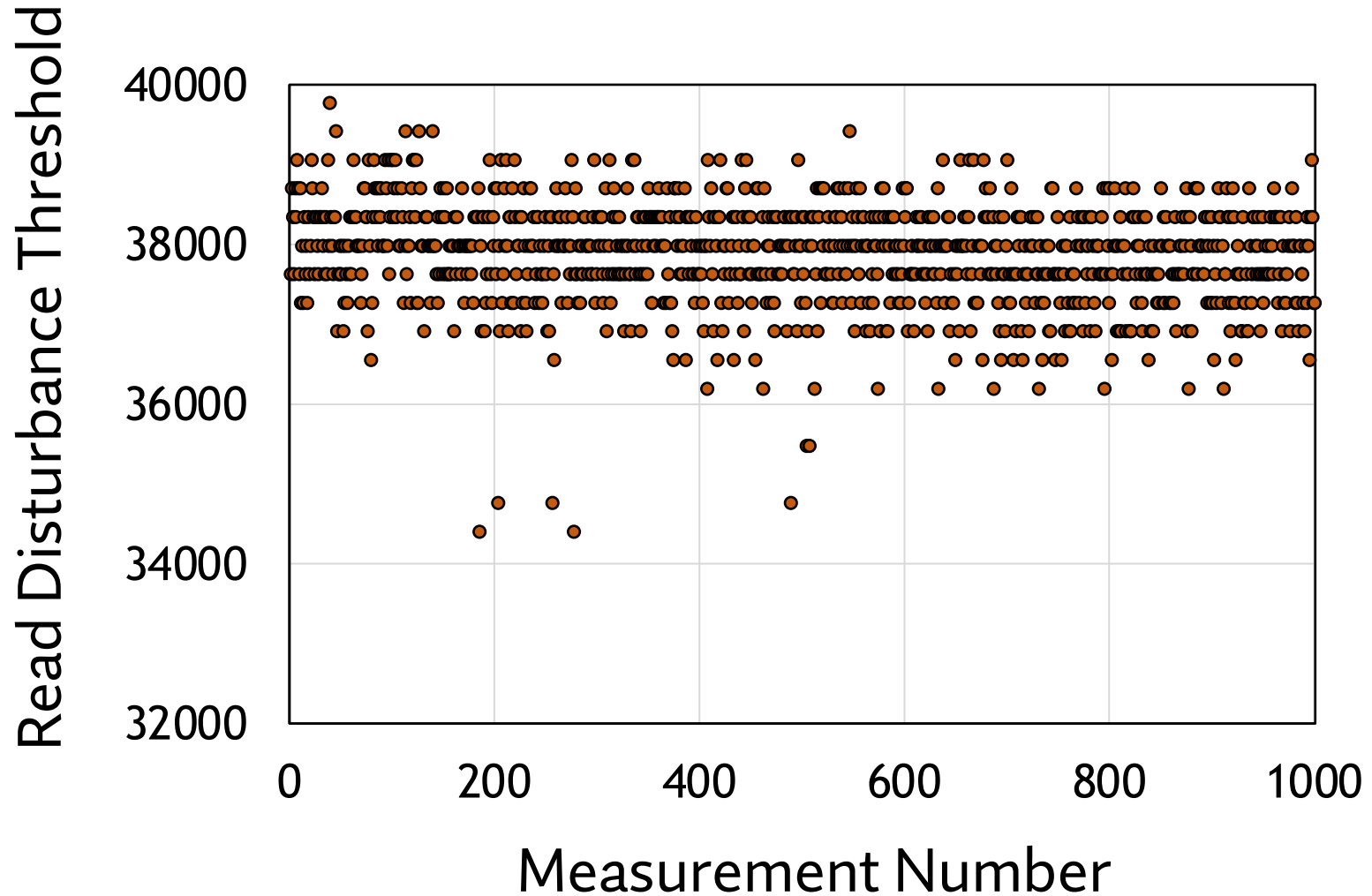


Measurement Number

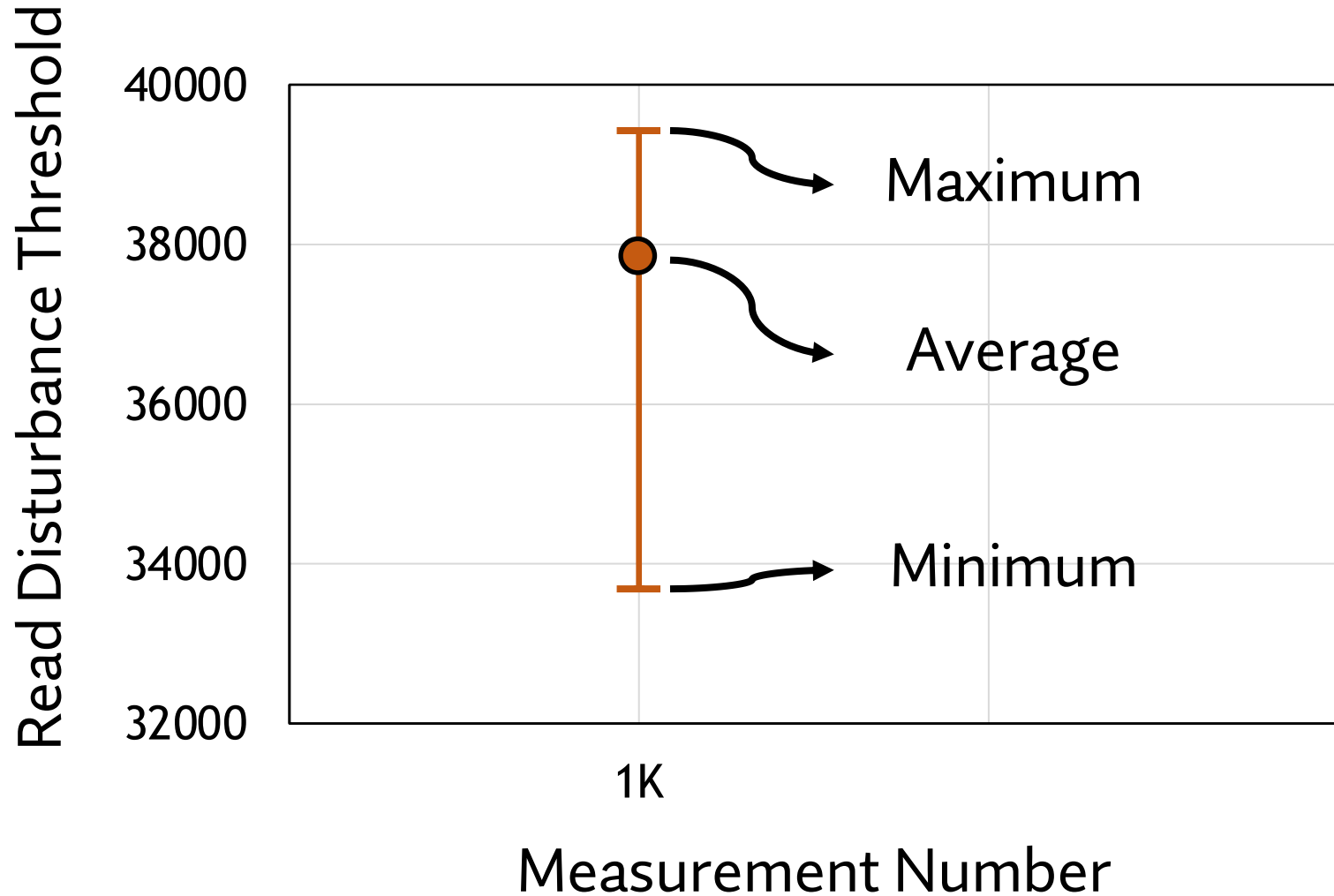
Read Disturbance Threshold Changes Over Time



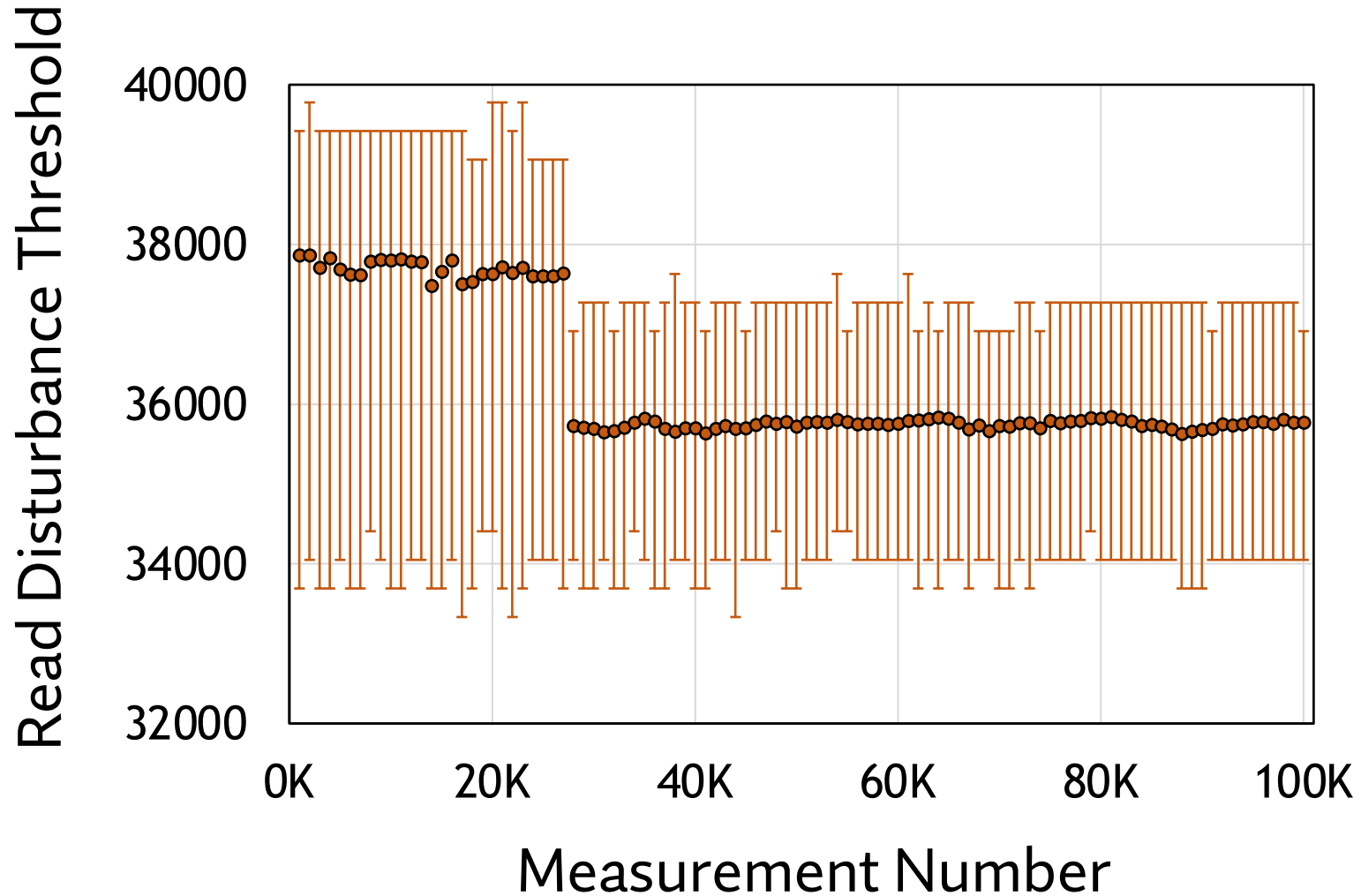
Read Disturbance Threshold Changes Over Time



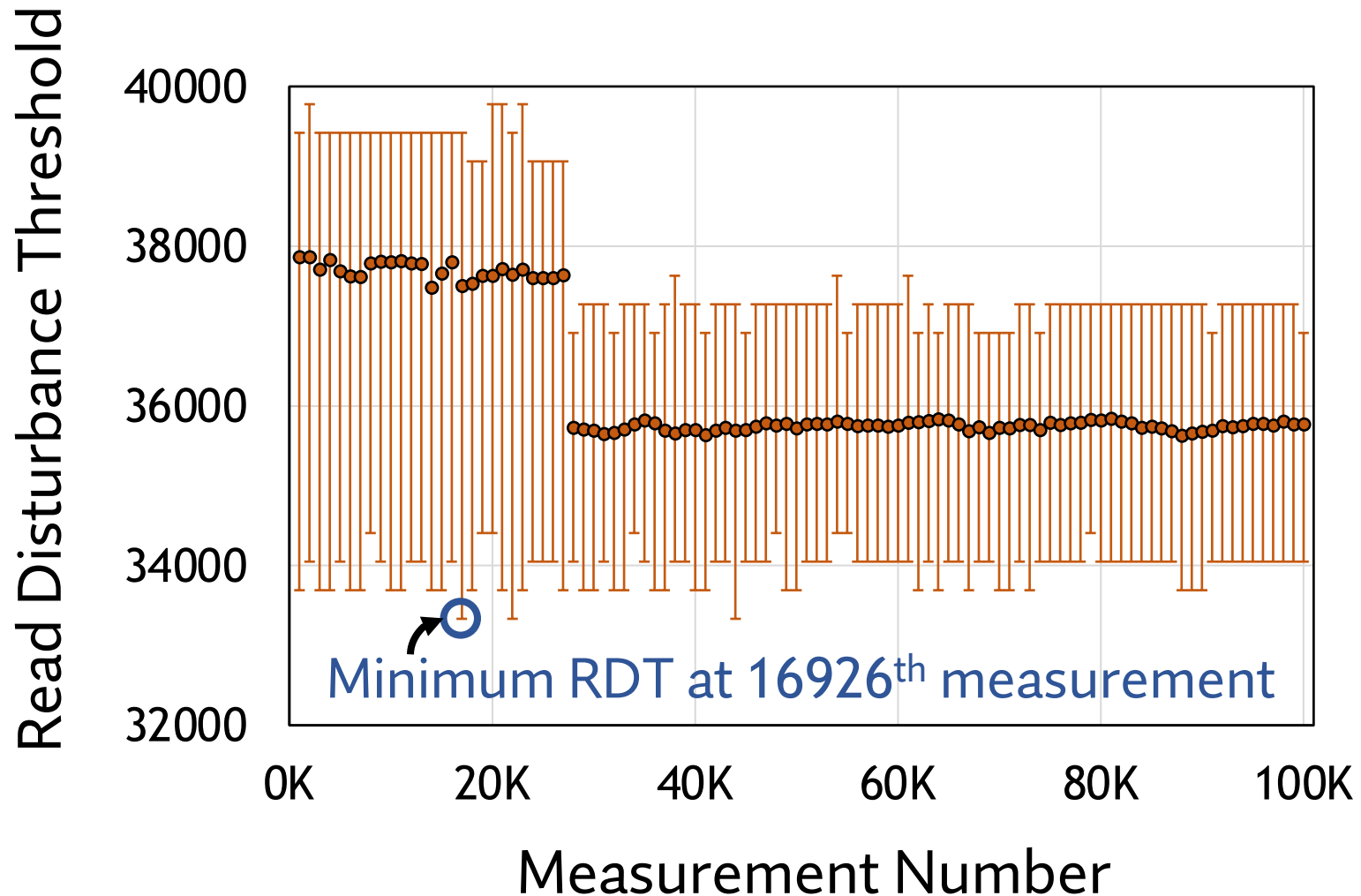
Read Disturbance Threshold Changes Over Time



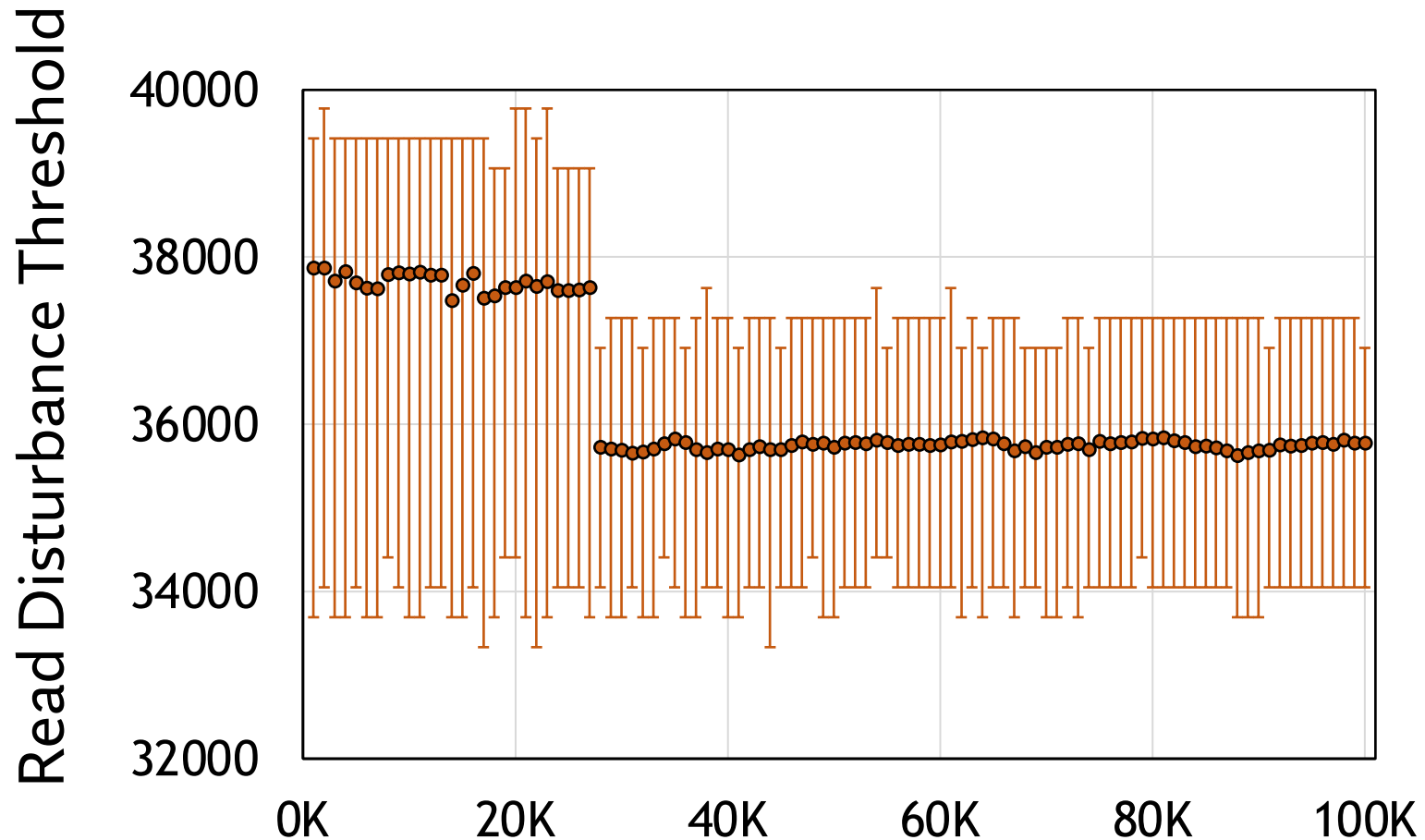
Read Disturbance Threshold Changes Over Time



Read Disturbance Threshold Changes Over Time

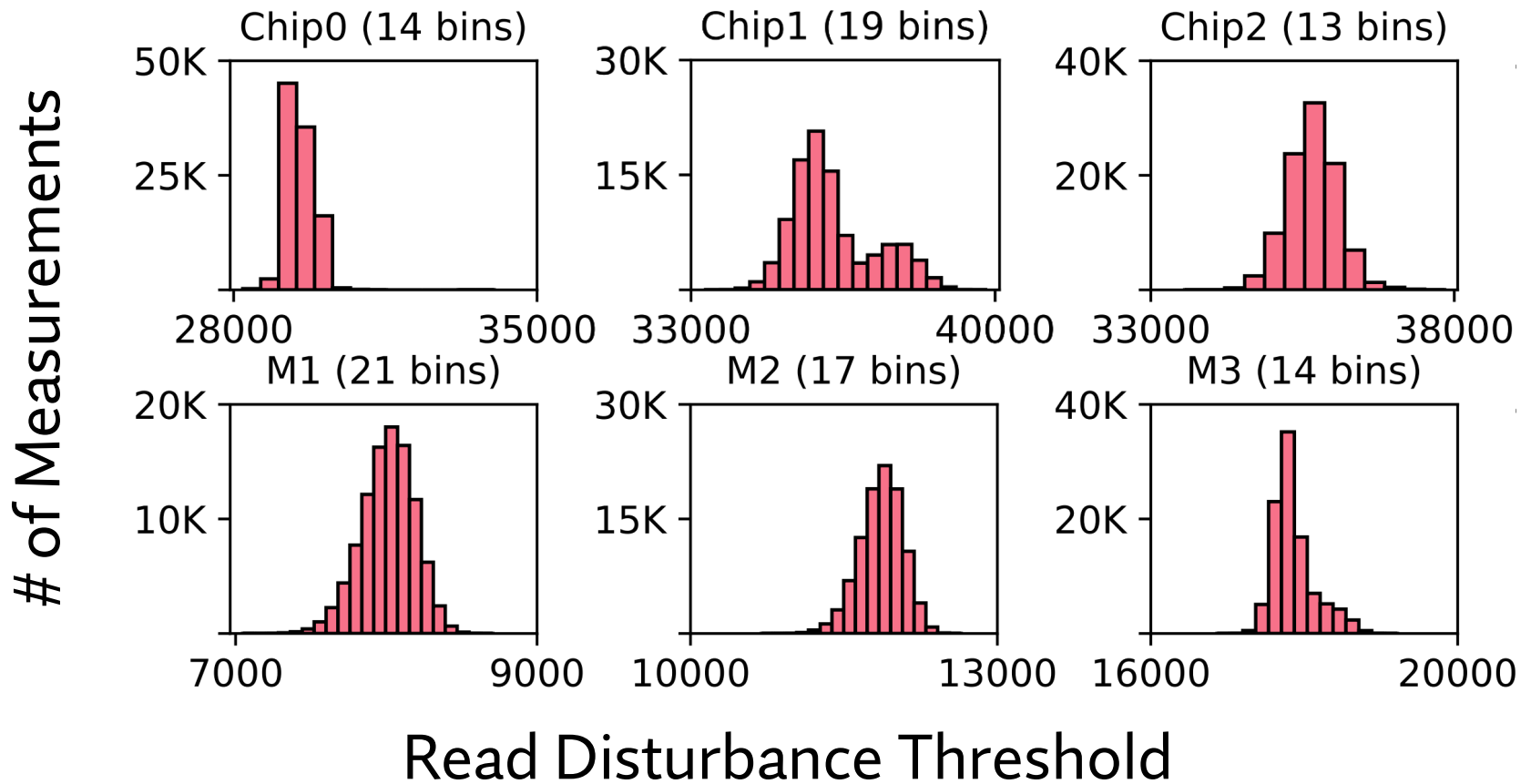


Read Disturbance Threshold Changes Over Time



Read Disturbance Threshold of a DRAM row
varies over time: **Variable Read Disturbance (VRD)**

Variable Read Disturbance Across DRAM Chips



RDT **consistently varies** over time
across all tested DRAM **chips**

Variable Read Disturbance Across DRAM Chips

<https://arxiv.org/pdf/2502.13075>

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†
ETH Zurich† Rutgers University‡

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is $3.5\times$ smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) many times (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

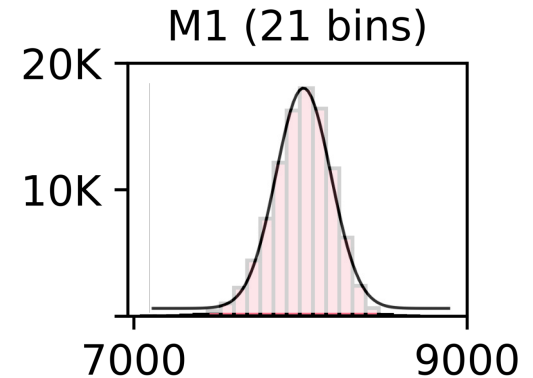
To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count* (the number of aggressor row activations) needed to induce the first read disturbance bitflip in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

VRD is (Likely) Unpredictable

- The outcome of the **next read disturbance threshold (RDT) measurement** cannot be **predicted** given past measurements

1

RDT histograms **well resemble*** **random probability distributions** e.g., normal distribution



2

Analyze and find **no repeating patterns** in the series of **consecutively measured RDT values** using the **autocorrelation function**

VRD is (Likely) Unpredictable

<https://arxiv.org/pdf/2502.13075>

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun[†] F. Nisa Bostancı[†] İsmail Emir Yüksel[†] Oğuzhan Canpolat[†] Haocong Luo[†]
Geraldo F. Oliveira[†] A. Giray Yağlıkçı[†] Minesh Patel[‡] Onur Mutlu[‡]
ETH Zurich[†] Rutgers University[‡]

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is $3.5\times$ smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) many times (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by preventively refreshing (i.e., opening and closing) a victim row before a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to accurately identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the hammer count (the number of aggressor row activations) needed to induce the first read disturbance bitflip in a victim row. We call this metric the read disturbance threshold (RDT) of the victim row.

In-Depth Analysis: Parameter Space

- Four **data patterns**

Row Addresses	<i>Rowstripe0</i>	<i>Rowstripe1</i>	<i>Checkered0</i>	<i>Checkered1</i>
Victim (V)	0x00	0xFF	0x55	0xAA
Aggressors ($V \pm 1$)	0xFF	0x00	0xAA	0x55
$V \pm [2:8]$	0x00	0xFF	0x55	0xAA

- Three **temperature** levels: 50°C, 65°C, 80°C
- Three **aggressor row on time** values (RowPress):
 - Minimum $t_{RAS} = \sim 35ns$
 - Interval between two periodic refresh commands $t_{REFI} = 7.8\mu s$ (DDR4)
 - Maximum interval between two refresh $9 \times t_{REFI} = 70.2\mu s$ (DDR4)
- Test **3750 rows** and **measure RDT 1000 times** per row
 - Aside: what would happen if we measure **>1M** times?

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows **exhibit VRD**

Takeaway 2

Relatively **few** (<500) RDT **measurements** are **unlikely to yield the minimum RDT** of a row

Takeaway 3

Data patterns, temperature, and aggressor row on time **affect VRD**

Implications Summary

- **Security guarantees** provided by **mitigation techniques** rely on **accurately identified minimum read disturbance threshold (RDT)**
- **Accurate identification** of minimum RDT (for **each row**) is **extremely challenging (even with 1000s measurements)** because **RDT unpredictably changes over time**
- We analyze the use of a **guardband** and **ECC**
 - **May** prevent VRD-induced bitflips
 - Large guardbands induce **performance overhead**
- Call for future work on **online RDT profiling** and **runtime configurable** read disturbance **mitigations**

Combining ECC and Guardbands (II)

RDT guardbands $\geq 20\%$ yield 1 unique bitflip in a row

Given our limited measurement dataset (10K measurements)

RDT guardbands $\geq 20\%$ combined with ECC
may prevent VRD-induced read disturbance bitflips

More detailed analysis (following a large-scale study)
needed to make a definitive conclusion

Extended Version on arXiv

<https://arxiv.org/pdf/2502.13075>

 > cs > arXiv:2502.13075

Search... All fields  Search

Help | Advanced Search

Computer Science > Hardware Architecture

[Submitted on 18 Feb 2025]

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun, F. Nisa Bostanci, Ismail Emir Yuksel, Oguzhan Canpolat, Haocong Luo, Geraldo F. Oliveira, A. Giray Yaglikci, Minesh Patel, Onur Mutlu

Modern DRAM chips are subject to read disturbance errors. State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead. We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield two key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5X smaller than the maximum RDT observed for that row. Second, the probability of accurately identifying a row's RDT with a relatively small number of measurements reduces with increasing chip density or smaller technology node size. Our empirical results have implications for the security guarantees of read disturbance

Access Paper:

- [View PDF](#)
- [TeX Source](#)
- [Other Formats](#)

 [view license](#)

Current browse context:
cs.AR
[< prev](#) | [next >](#)
[new](#) | [recent](#) | [2025-02](#)
Change to browse by:
[cs](#)
[cs.CR](#)

References & Citations

- [NASA ADS](#)
- [Google Scholar](#)
- [Semantic Scholar](#)

Export BibTeX Citation

Bookmark

Variable Read Disturbance

- Ataberk Olgun, Nisa Bostanci, Ismail Emir Yuksel, Abdullah Giray Yaglikci, Geraldo Francisco de Oliveira, Haocong Luo, Oguzhan Canpolat, Minesh Patel, and Onur Mutlu,

"Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance"

Proceedings of the 31st International Symposium on High-Performance Computer Architecture (HPCA), Las Vegas, NV, USA, March 2025.

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

Variable Read Disturbance:

An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†

ETH Zurich† *Rutgers University*‡

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive char-

acteristics of the aggressor row (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

DiscoRD (Preprint)

How can we **reliably** identify RDT given VRD?

DiscoRD: An Experimental Methodology for Quickly Discovering the Reliable Read Disturbance Threshold of Real DRAM Chips

Ataberk Olgun[†] F. Nisa Bostancı[†] İsmail Emir Yüksel[†] Haocong Luo[†]
Minesh Patel[‡] A. Giray Yağlıkçı[†] Onur Mutlu[†]
ETH Zurich[†] *Rutgers University*[‡]

Modern DRAM chips suffer from read disturbance. Repeatedly activating an aggressor DRAM row (RowHammer) or keeping an aggressor row open for a very long time (RowPress) induces read disturbance bitflips in a physically nearby victim DRAM row. State-of-the-art read disturbance mitigations rely on the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first read disturbance bitflip) to securely and performance- and energy-efficiently prevent read disturbance bitflips. However, accurately and exhaustively characterizing the RDT of every DRAM row in a chip is time intensive.

1. Introduction

Read disturbance [1–6] (e.g., RowHammer and RowPress) in modern DRAM chips is a widespread phenomenon and is reliably used for breaking memory isolation [1–73], a fundamental building block for building robust systems. Repeatedly opening/activating and closing a DRAM row (i.e., aggressor row) *many times* (e.g., tens of thousands of times) induces *RowHammer bitflips* in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces *RowPress bitflips*, *without requiring many repeated aggressor*



- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,
"RowPress: Amplifying Read Disturbance in Modern DRAM Chips"
Proceedings of the 50th International Symposium on Computer Architecture (ISCA),
Orlando, FL, USA, June 2023.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Video](#) (3 minutes)]
[[RowPress Source Code and Datasets \(Officially Artifact Evaluated with All Badges\)](#)]
Officially artifact evaluated as available, reusable and reproducible.
Best artifact award at ISCA 2023.
One of the 12 computer architecture papers of 2023
selected as Top Picks by IEEE Micro.

RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

Haocong Luo Ataberk Olgun A. Giray Yağlıkçı Yahya Can Tuğrul Steve Rhyner
Meryem Banu Cavlak Joël Lindegger Mohammad Sadrosadati Onur Mutlu
ETH Zürich

RowPress: Key Observation

Keeping a DRAM row **open for a long time** causes bitflips in adjacent rows

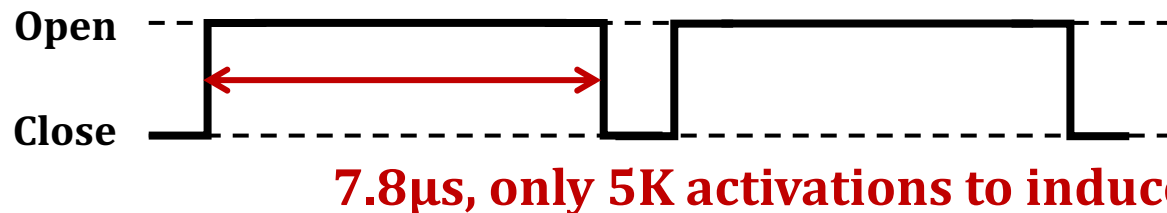
These bitflips do **NOT** require many row activations

Only one activation is enough in some cases!

RowHammer
Aggressor Row



RowPress
Aggressor Row



Real-System Demonstration (I)



Intel Core i5-10400
(Comet Lake)



Samsung DDR4 Module
M378A2K43CB1-CTD
(Date Code: 20-10)

w/ TRR RowHammer Mitigation

Key Idea: A proof-of-concept RowPress program keeps a DRAM row open for a longer period by **keeping on accessing different cache blocks in the row**

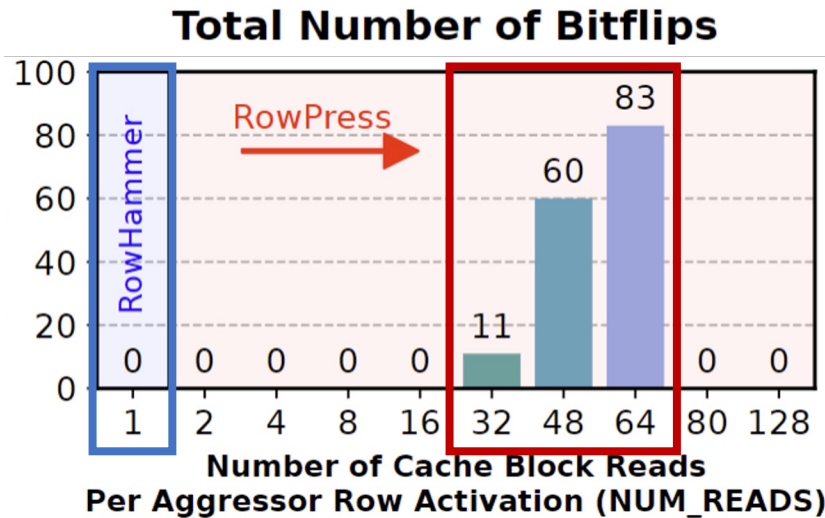
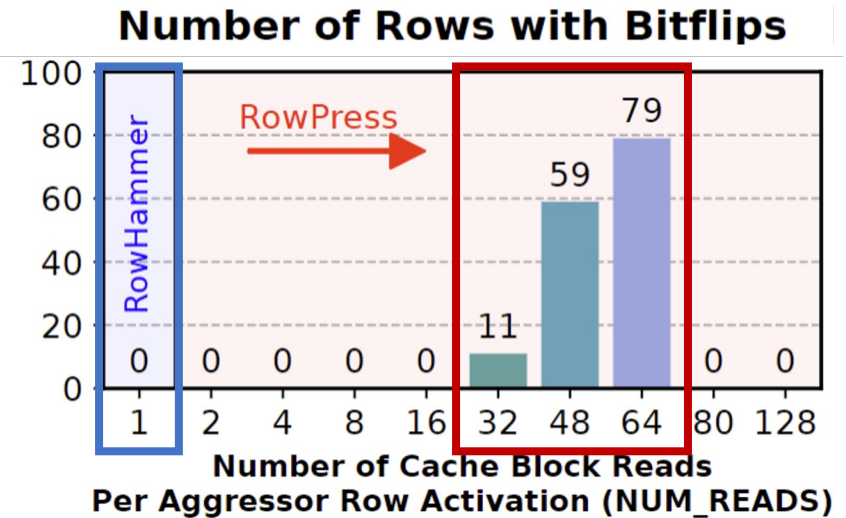
```
// Sync with Refresh and Loop Below
```

```
for (k = 0; k < NUM_AGGR_ACTS; k++)  
    for (j = 0; j < NUM_READS; j++) *AGGRESSOR1[j];  
    for (j = 0; j < NUM_READS; j++) *AGGRESSOR2[j];  
    for (j = 0; j < NUM_READS; j++)  
        clflushopt(AGGRESSOR1[j]);  
        clflushopt(AGGRESSOR2[j]);  
    mfence();  
    activate_dummy_rows();
```

**Number of Cache Blocks Accessed
Per Aggressor Row ACT
(NUM_READS=1 is Rowhammer)**

Real-System Demonstration (II)

On 1500 victim rows



Leveraging RowPress, our user-level program induces bitflips when RowHammer cannot

RowPress Results & Source Code

RowPress: Amplifying Read Disturbance in Modern DRAM Chips

Haocong Luo Ataberk Olgun A. Giray Yağlıkçı Yahya Can Tuğrul Steve Rhyner
Meryem Banu Cavlak Joël Lindegger Mohammad Sadrosadati Onur Mutlu

ETH Zürich



Fully open source and artifact evaluated

➤ <https://github.com/CMU-SAFARI/RowPress>



ColumnDisturb

- Ismail Emir Yüksel, Ataberk Olgun, Nisa Bostancı, Haocong Luo, A. Giray Yağlıkçı, and Onur Mutlu,
"ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems"
Proceedings of the 58th International Symposium on Microarchitecture (MICRO), Seoul, South Korea, October 2025.
[[Slides \(pptx\)](#)] [[pdf](#)]

ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems

Ismail Emir Yüksel¹ Ataberk Olgun¹ F. Nisa Bostancı¹
Haocong Luo¹ A. Giray Yağlıkçı^{1,2} Onur Mutlu¹
¹ETH Zürich ²CISPA

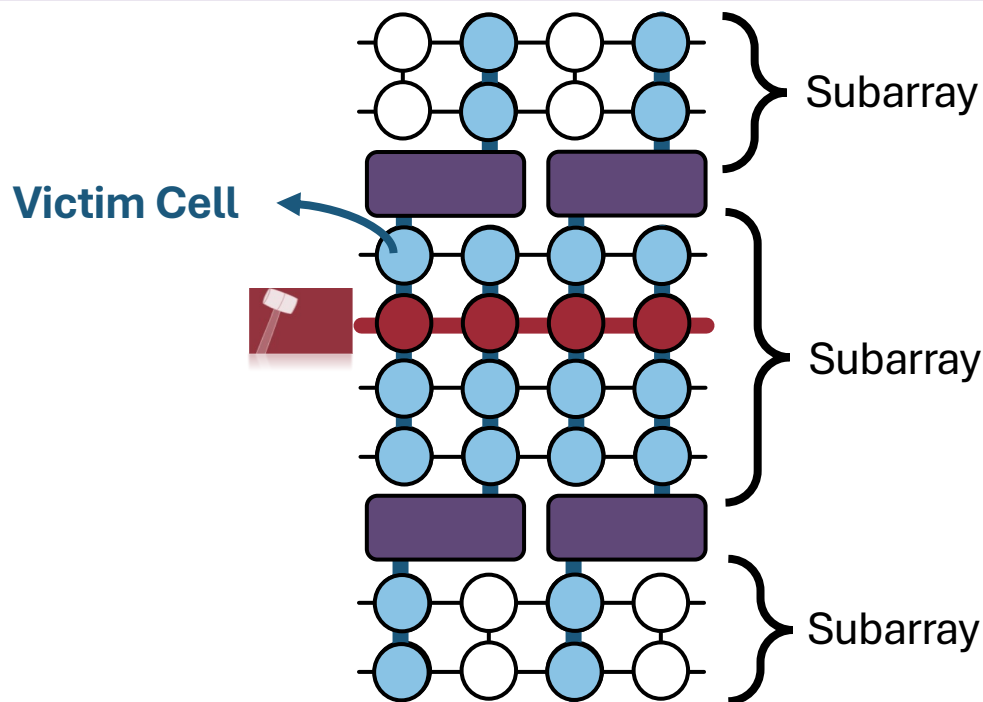
We experimentally demonstrate a new widespread read disturbance phenomenon, ColumnDisturb, in real commodity DRAM chips. By repeatedly opening or keeping a DRAM row (aggressor row) open, we show that it is possible to disturb DRAM cells through a DRAM column (i.e., bitline) and in-

DRAM cells by reducing (i.e., scaling) the technology node size of DRAM, which in turn reduces DRAM cell size, cell-to-cell spacing, and cell-to-bitline spacing [2]. Unfortunately, as a result of this aggressive technology node scaling, DRAM suffers from worsening read disturbance problems [3–6].

What if there is a **much worse**
read disturbance phenomenon
that affects
thousands of DRAM rows
through the **bitlines**?

ColumnDisturb

By repeatedly opening or keeping a DRAM row open, it is possible to **disturb** many DRAM cells on the same DRAM column



With **ColumnDisturb**, the activation of a single row **concurrently** affects DRAM cells across **as many as three DRAM subarrays** (e.g., up to 3072 DRAM rows)

ColumnDisturb

- Ismail Emir Yüksel, Ataberk Olgun, Nisa Bostancı, Haocong Luo, A. Giray Yağlıkçı, and Onur Mutlu,
"ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems"
Proceedings of the 58th International Symposium on Microarchitecture (MICRO), Seoul, South Korea, October 2025.
[[Slides \(pptx\)](#)] [[pdf](#)]

ColumnDisturb: Understanding Column-based Read Disturbance in Real DRAM Chips and Implications for Future Systems

Ismail Emir Yüksel¹ Ataberk Olgun¹ F. Nisa Bostancı¹
Haocong Luo¹ A. Giray Yağlıkçı^{1,2} Onur Mutlu¹
¹ETH Zürich ²CISPA

We experimentally demonstrate a new widespread read disturbance phenomenon, ColumnDisturb, in real commodity DRAM chips. By repeatedly opening or keeping a DRAM row (aggressor row) open, we show that it is possible to disturb DRAM cells through a DRAM column (i.e., bitline) and in-

DRAM cells by reducing (i.e., scaling) the technology node size of DRAM, which in turn reduces DRAM cell size, cell-to-cell spacing, and cell-to-bitline spacing [2]. Unfortunately, as a result of this aggressive technology node scaling, DRAM suffers from worsening read disturbance problems [3–6].

DRAM Bender

Enables Uncovering

New DRAM Functionality

QUAC-TRNG

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

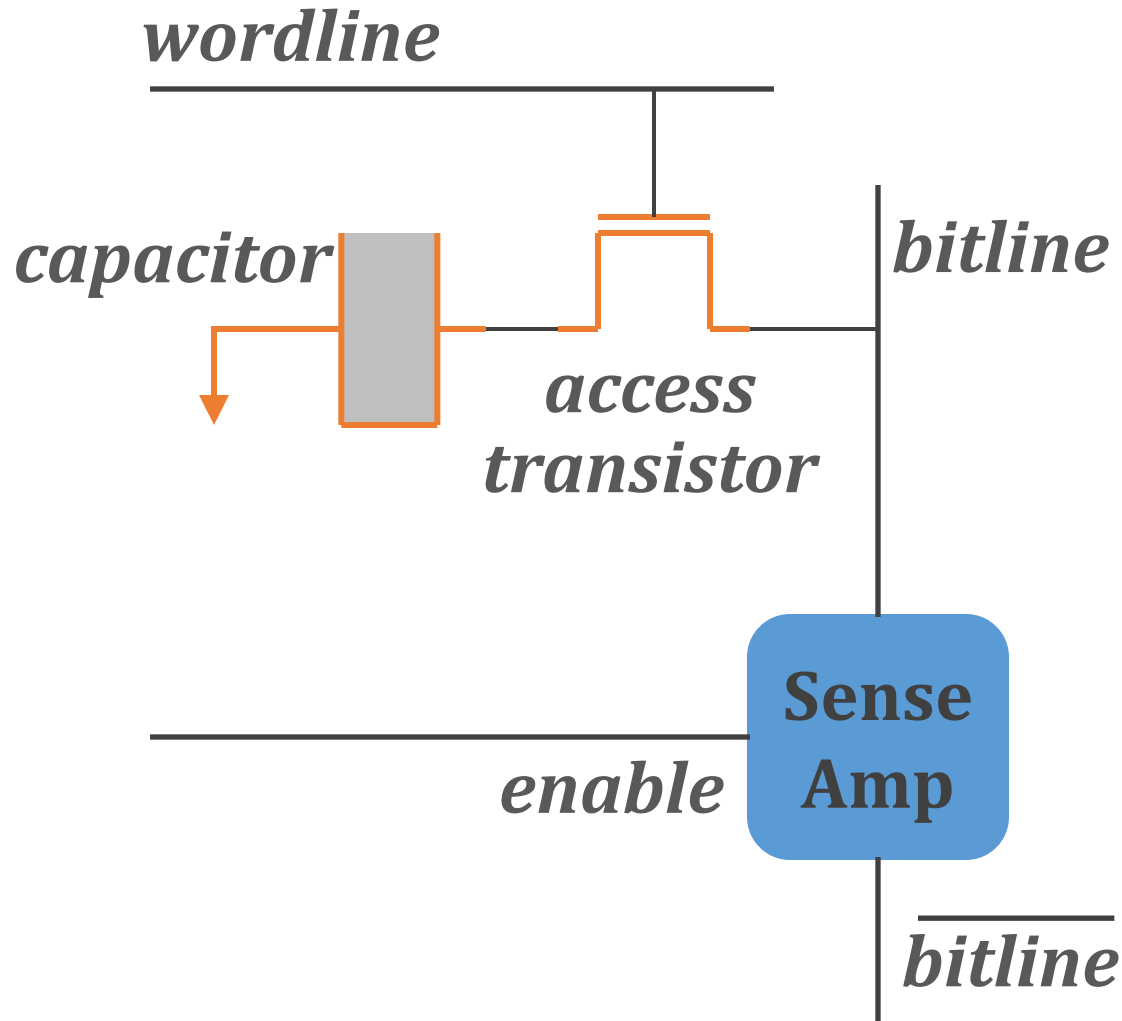
QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

Ataberk Olgun^{§†} Minesh Patel[§] A. Giray Yağlıkçı[§] Haocong Luo[§]
Jeremie S. Kim[§] F. Nisa Bostancı^{§†} Nandita Vijaykumar^{§⊙} Oğuz Ergin[†] Onur Mutlu[§]
[§]ETH Zürich [†]TOBB University of Economics and Technology [⊙]University of Toronto

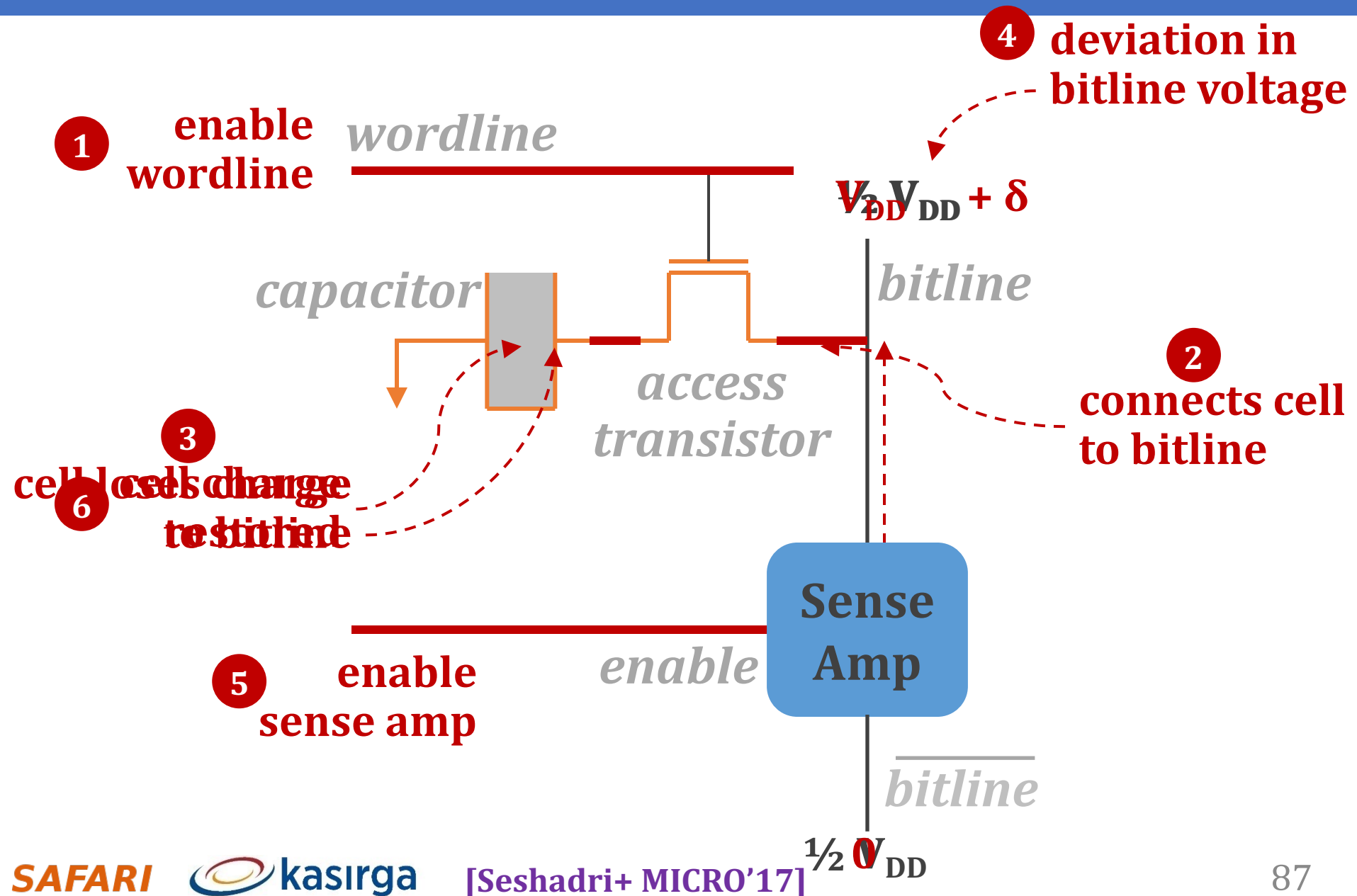
True random number generators (TRNG) sample random physical processes to create large amounts of random numbers for various use cases, including security-critical cryptographic primitives, scientific simulations, machine learning applications, and

many works have attempted to provide true random number generators purely using commodity hardware components that can be found in most systems today (e.g., DRAM [15, 81, 88, 126, 150] and SRAM [67, 68, 158]).

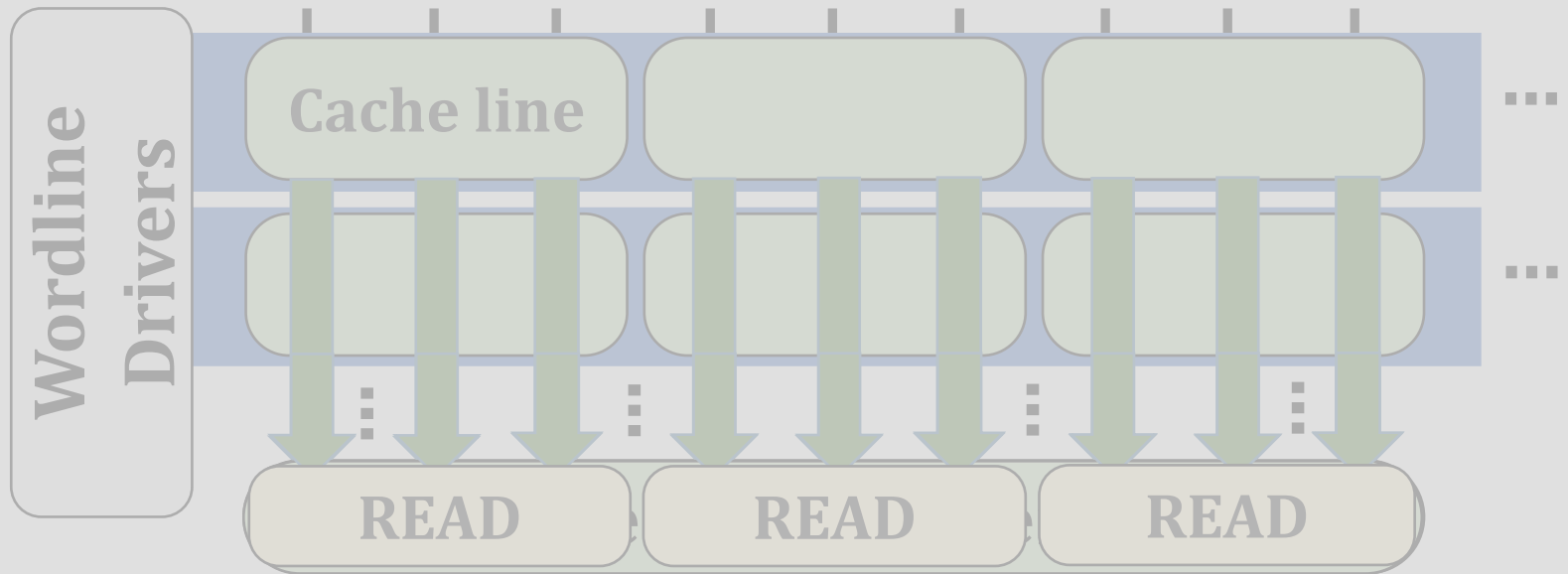
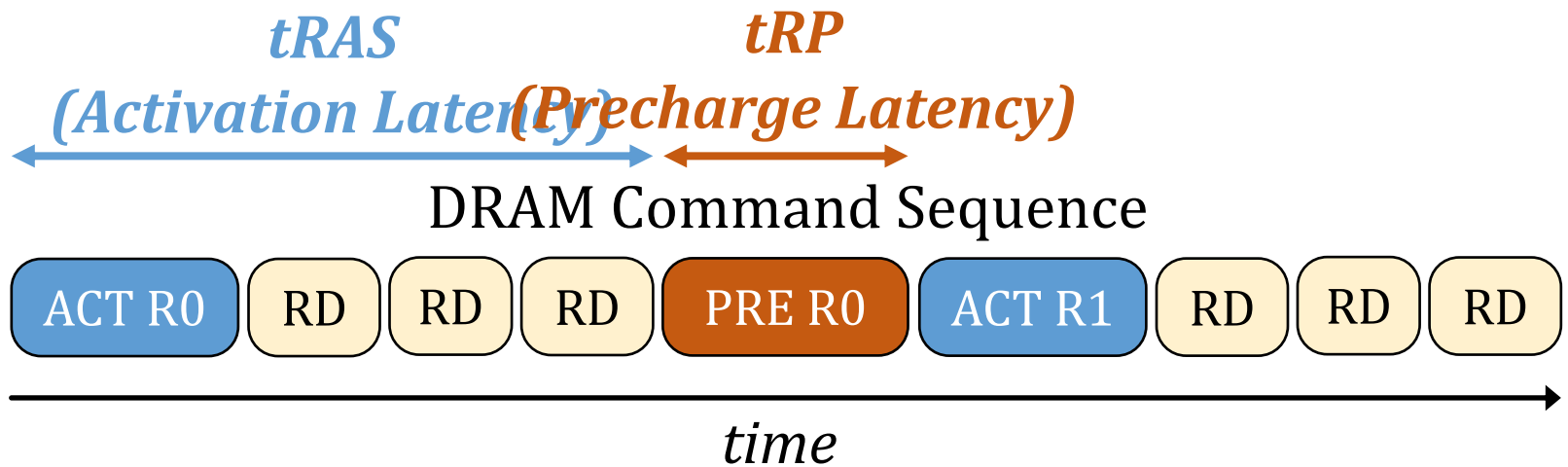
Accessing a DRAM Cell



Accessing a DRAM Cell



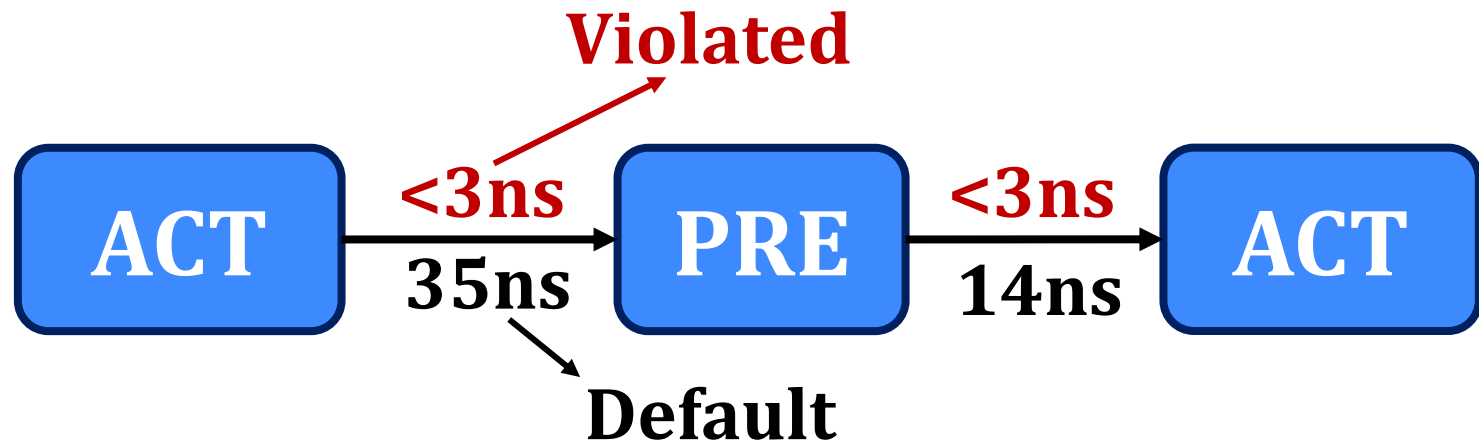
DRAM Operation



Quadruple Activation (QUAC)

New Observation

Carefully-engineered DRAM commands can activate four rows in real DRAM chips



Activate four rows with two ACT commands

Why Does QUAC Work?

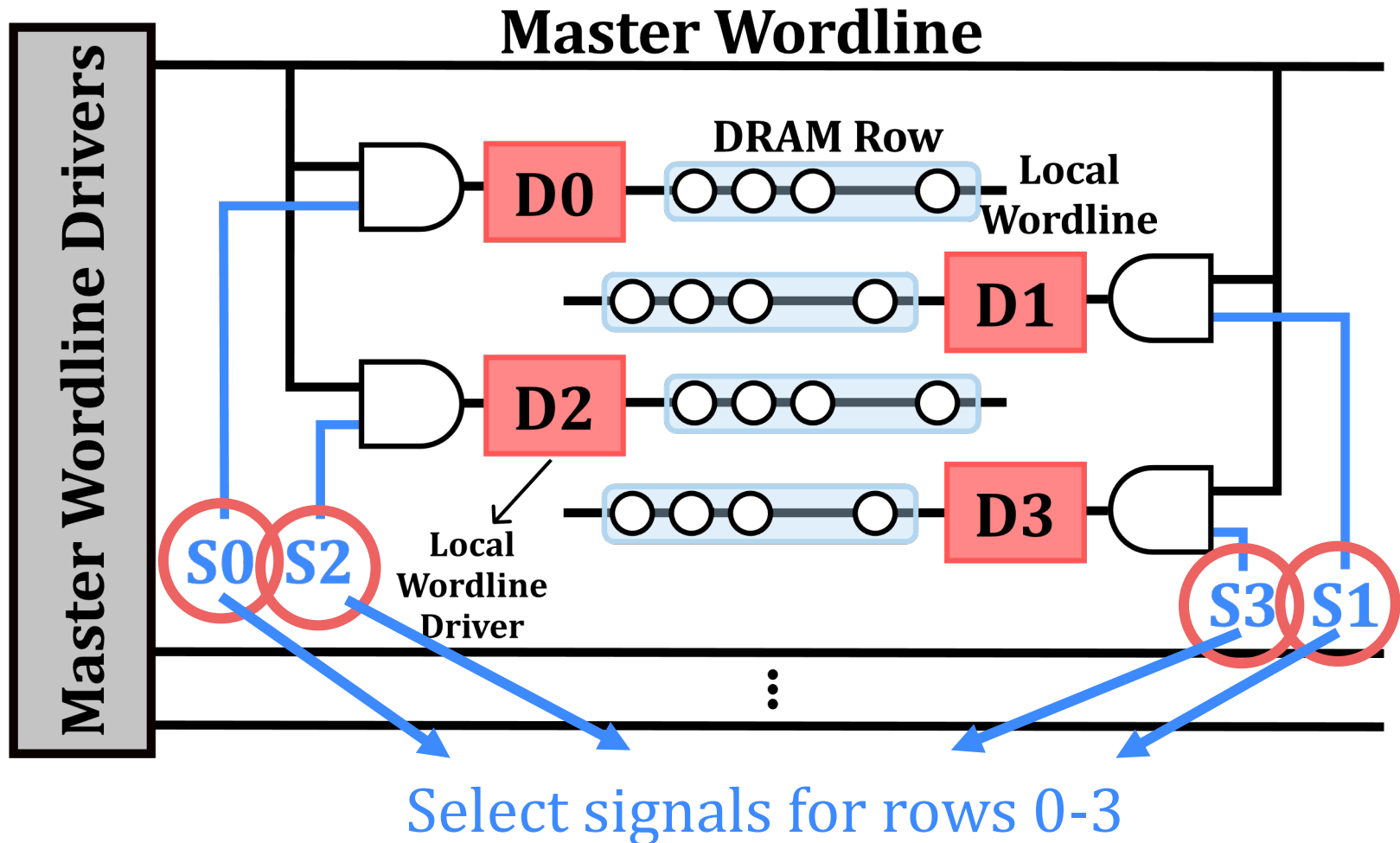
Hypothetical circuit to explain QUAC

Hierarchical Wordlines

- High *density* and *performance* requirements
- Hierarchical organization of DRAM wordlines *enable high-density* and *low-latency* DRAM operation

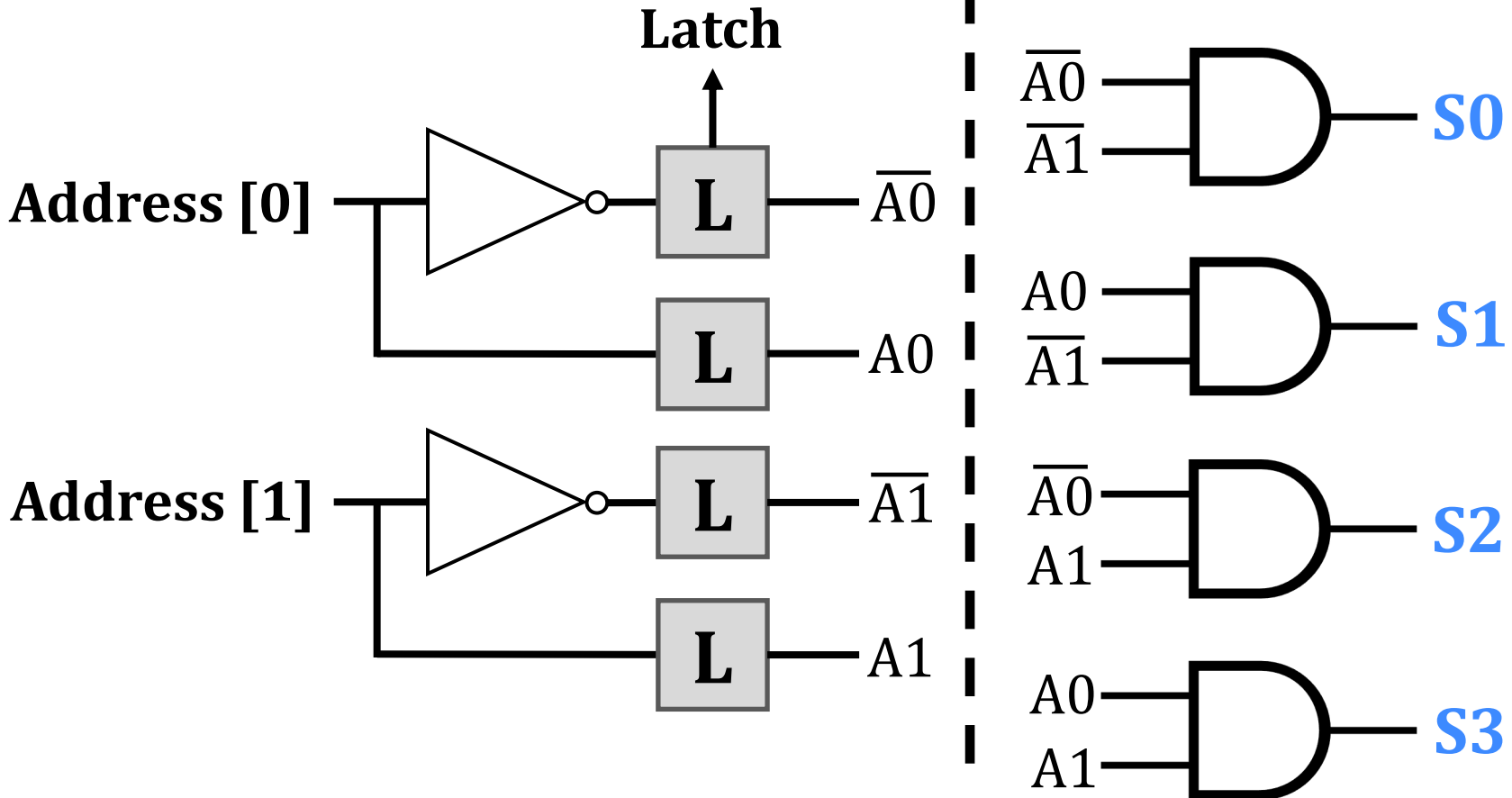
Hierarchical Wordlines

A **master** wordline drives multiple **local** wordlines

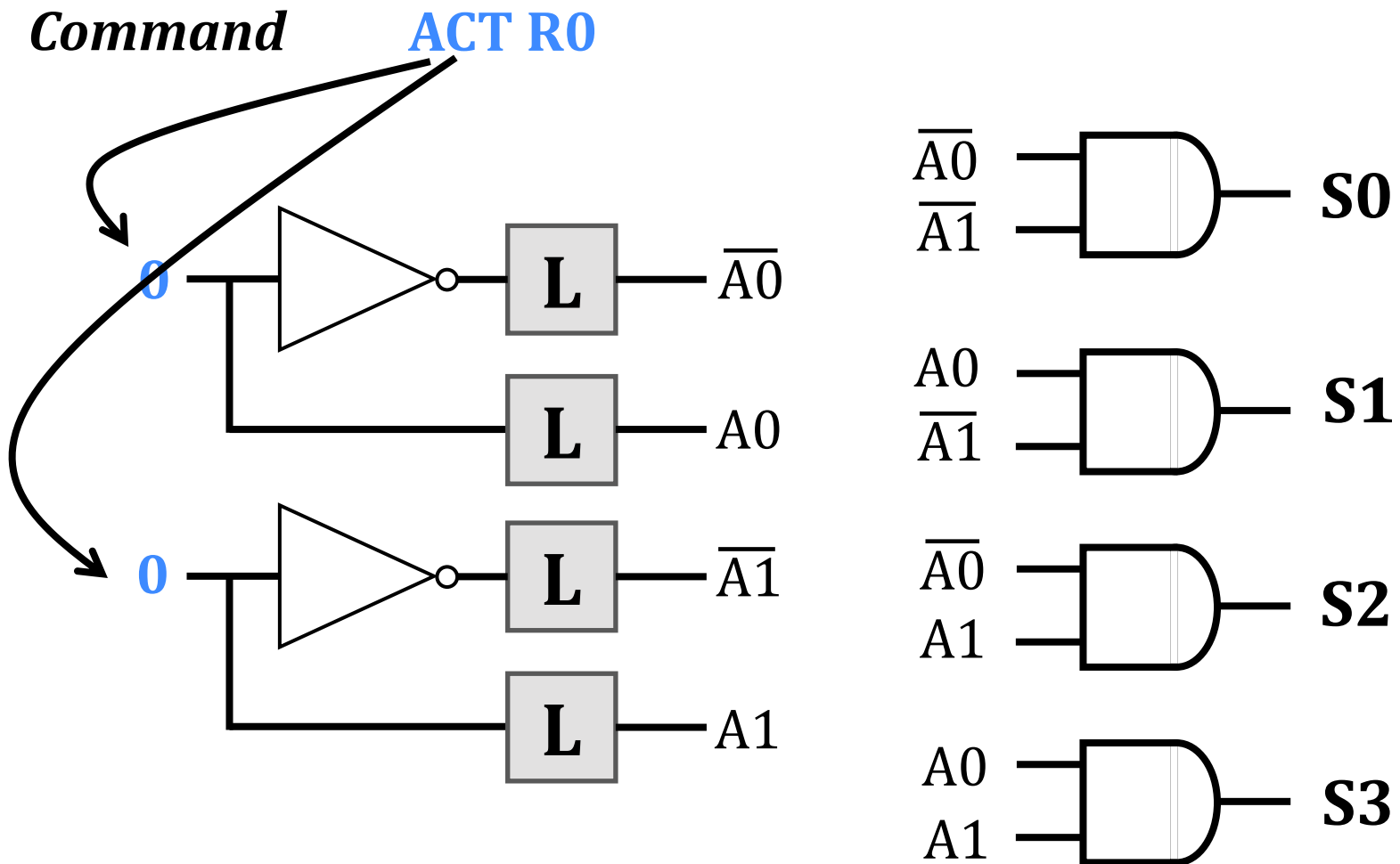


Hypothetical Row Decoder

Predecode the least significant two bits | *Drive control signals*



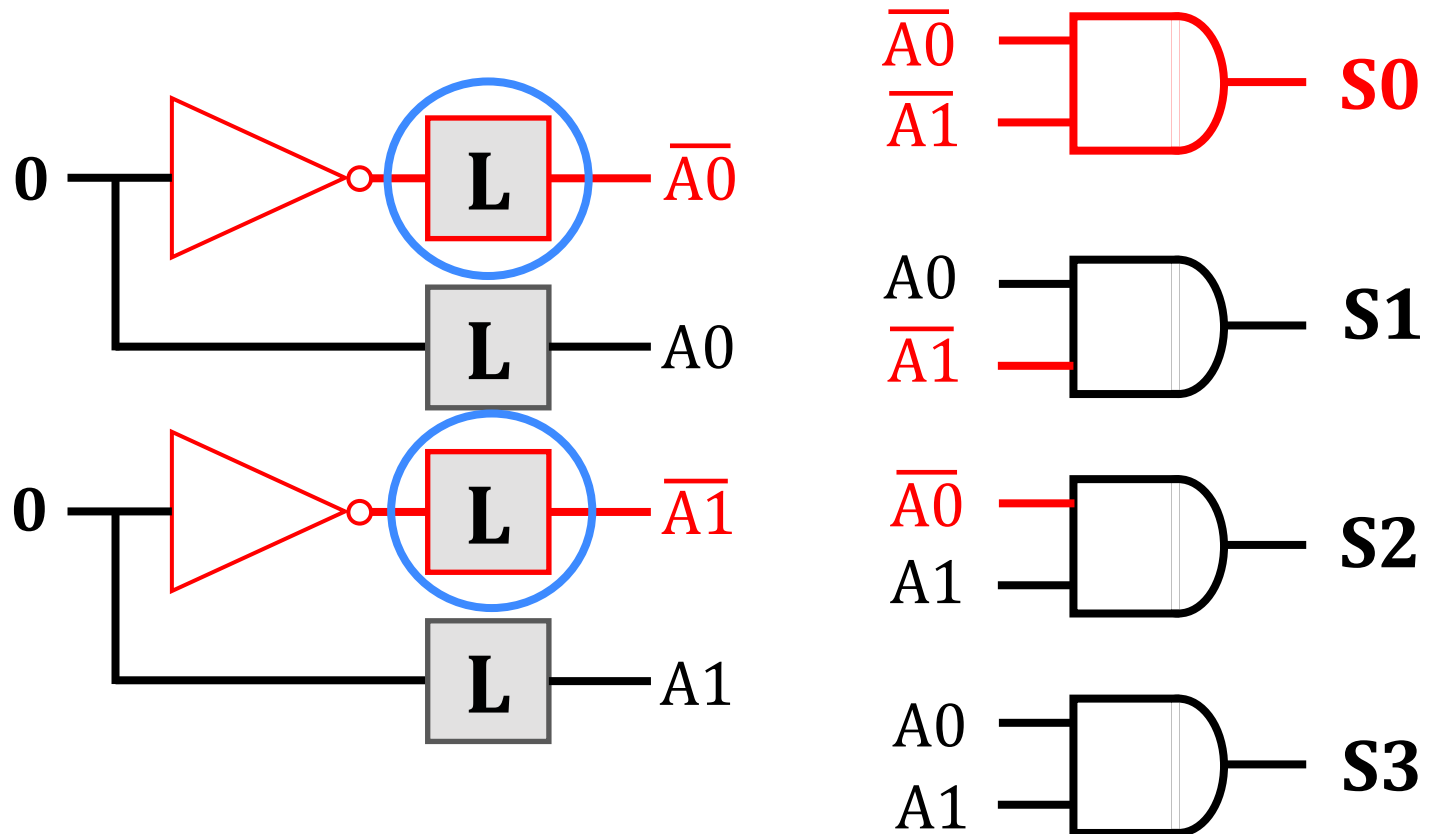
Hypothetical Row Decoder



First ACT command drives a single wordline

Hypothetical Row Decoder

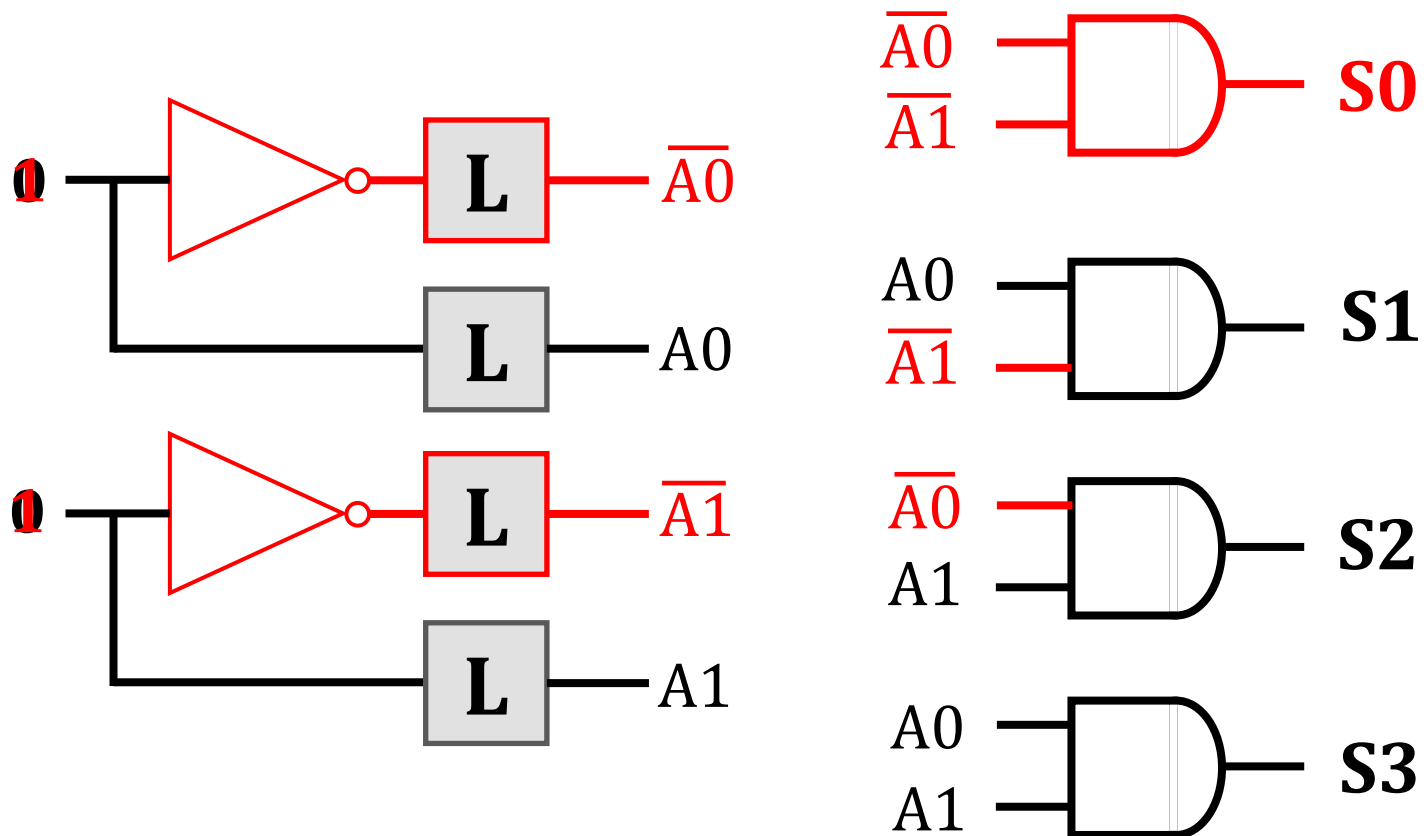
Command ACT R0 $\xrightarrow[\text{Timing}]{\text{Violate}}$ PRE



PRE command **cannot** disable latches

Hypothetical Row Decoder

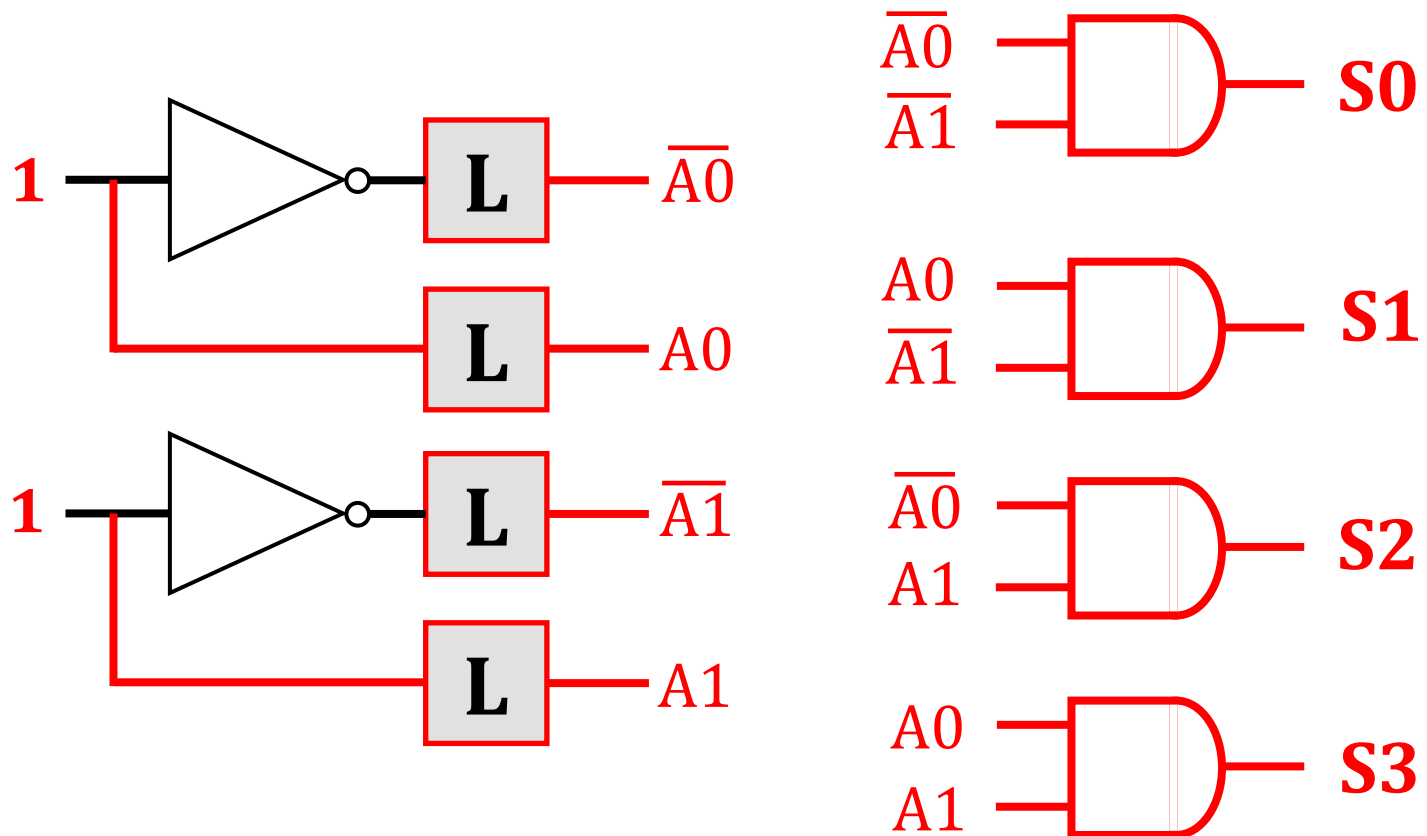
Command ACT R0 $\xrightarrow[\text{Violate Timing}]{}$ PRE $\xrightarrow[\text{Violate Timing}]{}$ ACT R3



Second ACT drives the **remaining three** wordlines

Hypothetical Row Decoder

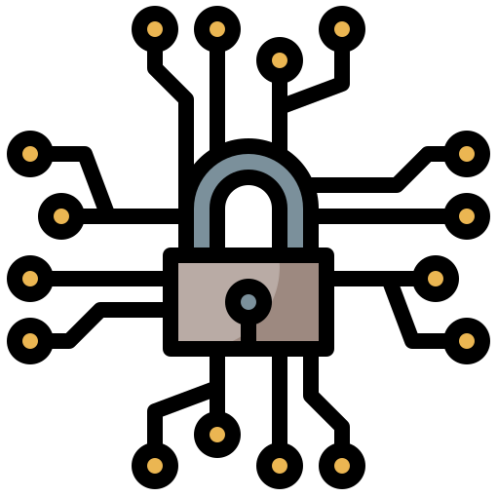
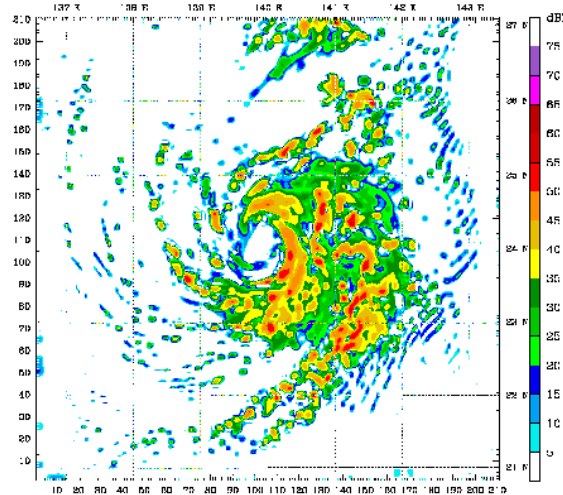
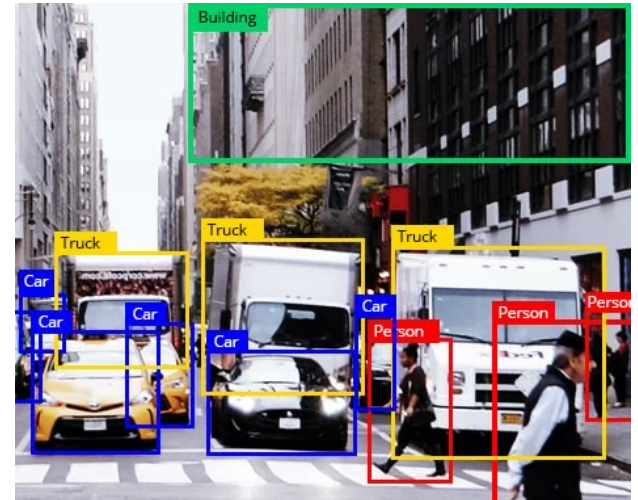
Command ACT R0 $\xrightarrow[\text{Timing}]{\text{Violate}}$ PRE $\xrightarrow[\text{Timing}]{\text{Violate}}$ ACT R3



All four wordlines are enabled
Quadruple Activation

Use Cases of True Random Numbers

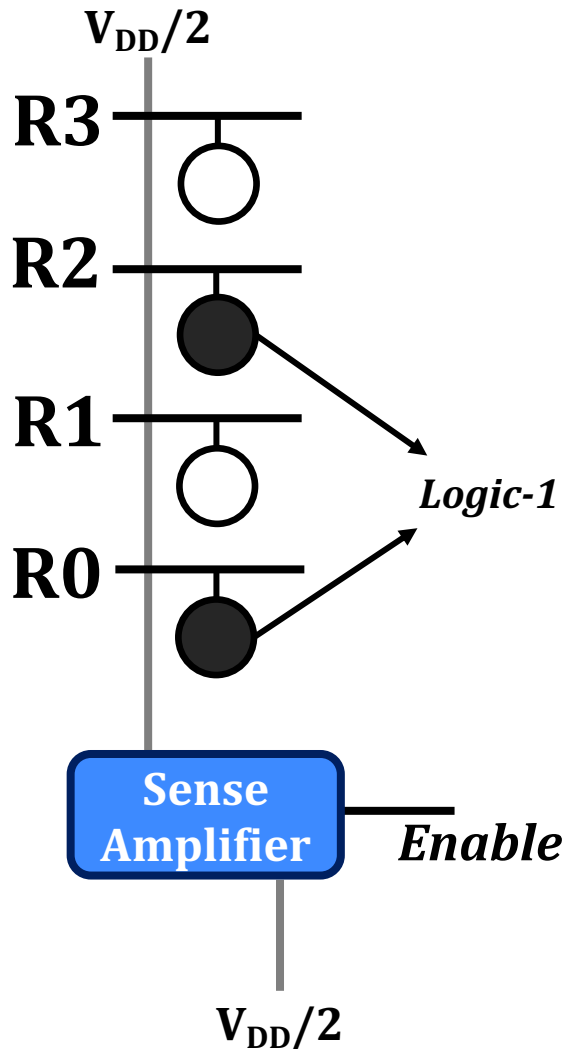
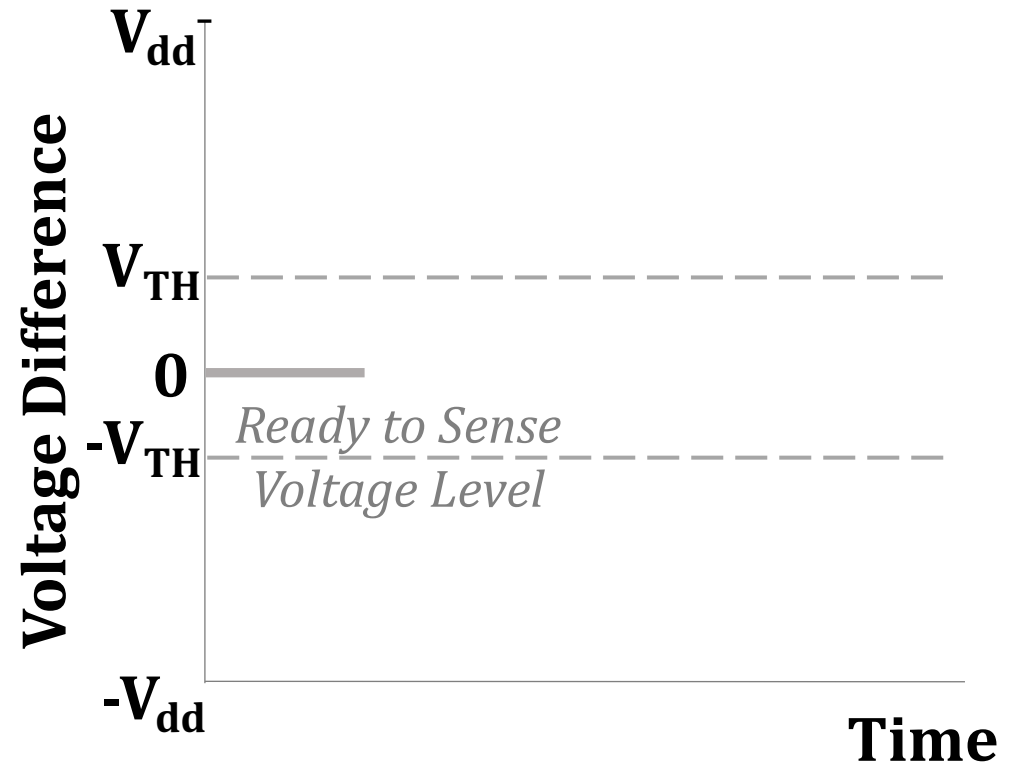
High-quality true random numbers
are **critical** to **many applications**



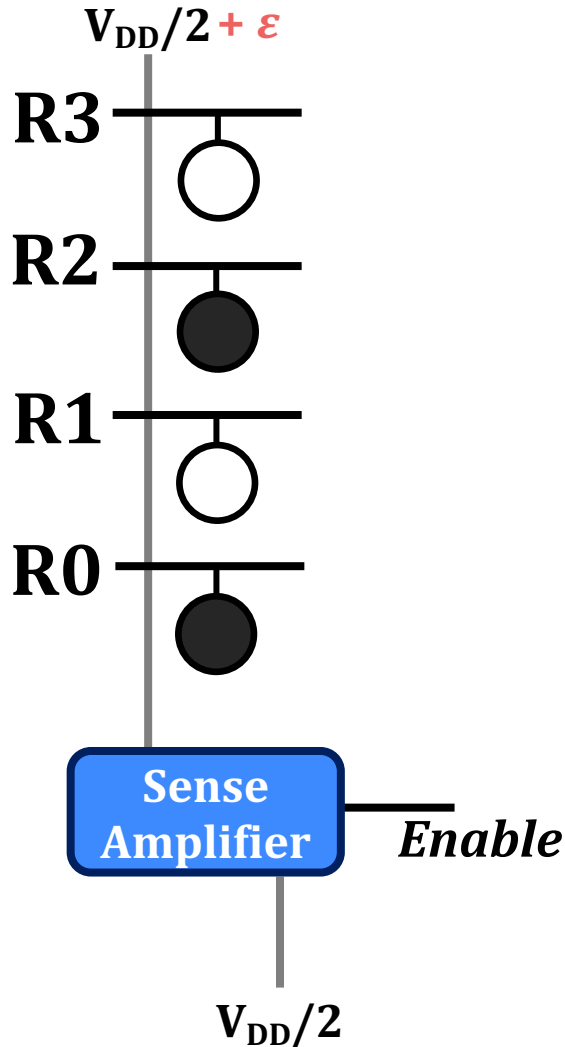
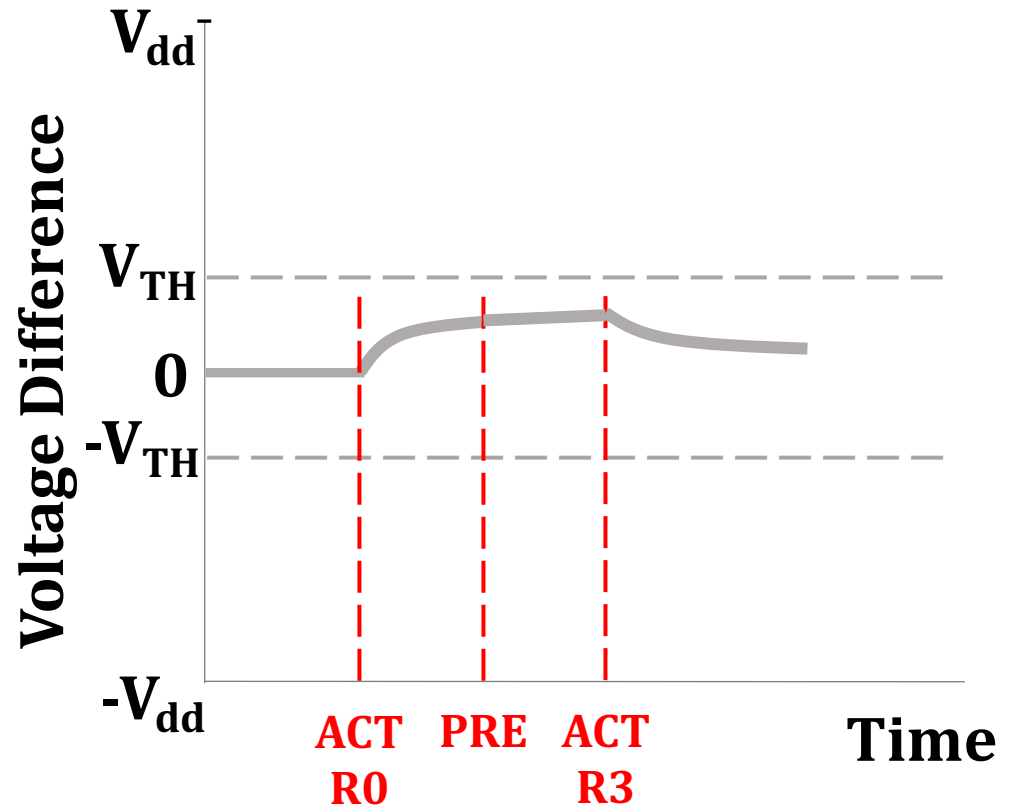
True random numbers can **only** be obtained
by sampling random physical processes

Unfortunately, **not all computing systems** are equipped
with **TRNG hardware** (e.g., dedicated circuitry)

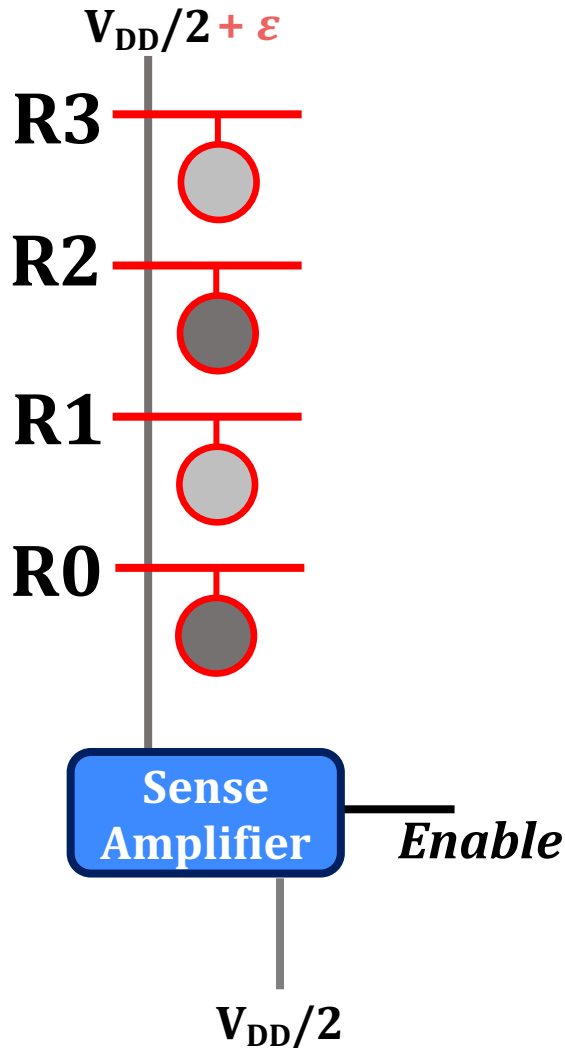
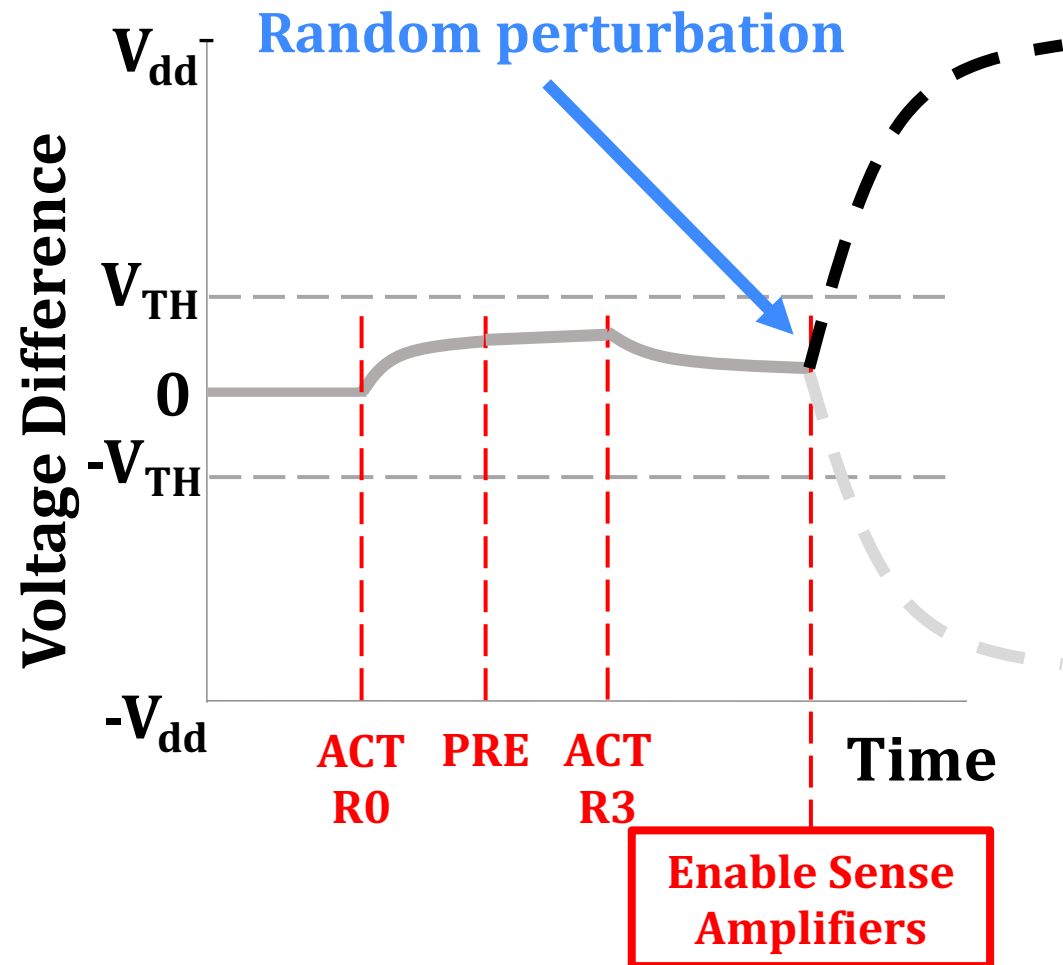
Generating Random Values via QUAC



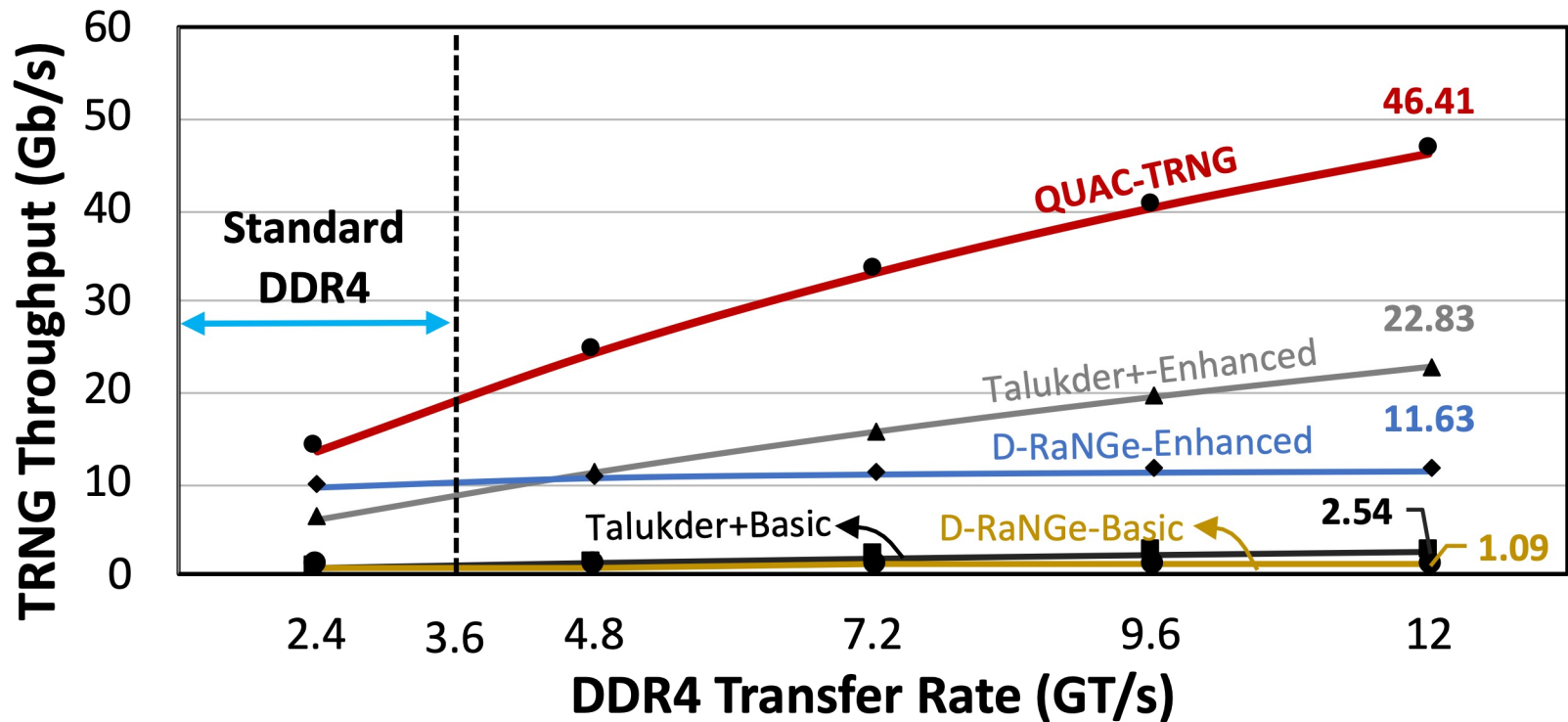
Generating Random Values via QUAC



Generating Random Values via QUAC



QUAC-TRNG vs State-Of-The-Art



Outperforms best prior DRAM-based TRNG

(i) “base” by **15.08x** at **2.4 GT/s**

(ii) “enhanced” by **2.03x** at **12 GT/s**

QUAC-TRNG

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

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

Ataberk Olgun^{§†} Minesh Patel[§] A. Giray Yağlıkçı[§] Haocong Luo[§]
Jeremie S. Kim[§] F. Nisa Bostancı^{§†} Nandita Vijaykumar^{§⊙} Oğuz Ergin[†] Onur Mutlu[§]
[§]ETH Zürich [†]TOBB University of Economics and Technology [⊙]University of Toronto

True random number generators (TRNG) sample random physical processes to create large amounts of random numbers for various use cases, including security-critical cryptographic primitives, scientific simulations, machine learning applications, and

many works have attempted to provide true random number generators purely using commodity hardware components that can be found in most systems today (e.g., DRAM [15, 81, 88, 126, 150] and SRAM [67, 68, 158]).

- Ismail Emir Yüksel, Ataberk Olgun, Nisa Bostancı, Oğuzhan Canpolat, Geraldo Francisco de Oliveira Junior, Mohammad Sadrosadati, Abdullah Giray Yağlıkçı, and Onur Mutlu,

"In-DRAM True Random Number Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Real DRAM Chips"

Proceedings of the 43rd IEEE International Conference on Computer Design (ICCD), Dallas, TX, USA, November 2025.

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

**In-DRAM True Random Number Generation
Using Simultaneous Multiple-Row Activation:
An Experimental Study of Real DRAM Chips**

Ismail Emir Yüksel[§] Ataberk Olgun[§] F. Nisa Bostancı[§] Oğuzhan Canpolat[§]
Geraldo F. Oliveira[§] Mohammad Sadrosadati[§] A. Giray Yağlıkçı^{§Γ} Onur Mutlu[§]

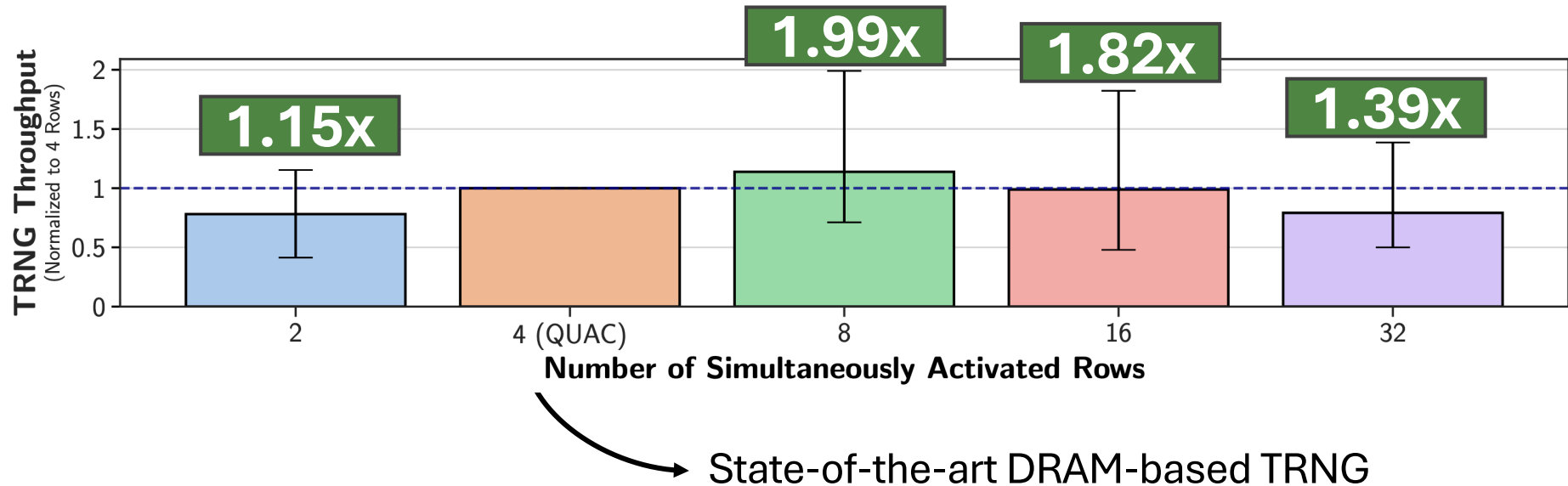
[§]ETH Zürich

^ΓCISPA

In this work, we experimentally demonstrate that it is possible to generate true random numbers at high throughput and low latency in commercial off-the-shelf (COTS) DRAM chips by leveraging simultaneous multiple-row activation (SiMRA) via an extensive characterization of 96 DDR4 DRAM chips. We rigorously analyze SiMRA's true random generation potential in

computing systems, ranging from microcontrollers to supercomputers, and systems that employ Processing-in-Memory (PIM) [29–46]. Several prior works [47–53] demonstrate that by leveraging the analog operational properties of DRAM, it is possible to generate true random numbers in commercial off-the-shelf (COTS) DRAM chips.

SiMRA-TRNG: Throughput



**SiMRA-based TRNG designs outperform
the state-of-the-art DRAM-based TRNG**

- Ismail Emir Yüksel, Ataberk Olgun, Nisa Bostancı, Oğuzhan Canpolat, Geraldo Francisco de Oliveira Junior, Mohammad Sadrosadati, Abdullah Giray Yağlıkçı, and Onur Mutlu,

"In-DRAM True Random Number Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Real DRAM Chips"

Proceedings of the 43rd IEEE International Conference on Computer Design (ICCD), Dallas, TX, USA, November 2025.

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

**In-DRAM True Random Number Generation
Using Simultaneous Multiple-Row Activation:
An Experimental Study of Real DRAM Chips**

Ismail Emir Yüksel[§] Ataberk Olgun[§] F. Nisa Bostancı[§] Oğuzhan Canpolat[§]
Geraldo F. Oliveira[§] Mohammad Sadrosadati[§] A. Giray Yağlıkçı^{§Γ} Onur Mutlu[§]

[§]ETH Zürich

^ΓCISPA

In this work, we experimentally demonstrate that it is possible to generate true random numbers at high throughput and low latency in commercial off-the-shelf (COTS) DRAM chips by leveraging simultaneous multiple-row activation (SiMRA) via an extensive characterization of 96 DDR4 DRAM chips. We rigorously analyze SiMRA's true random generation potential in

computing systems, ranging from microcontrollers to supercomputers, and systems that employ Processing-in-Memory (PIM) [29–46]. Several prior works [47–53] demonstrate that by leveraging the analog operational properties of DRAM, it is possible to generate true random numbers in commercial off-the-shelf (COTS) DRAM chips.

In-DRAM Signature Generation Using Simultaneous Multiple-Row Activation: An Experimental Study of Off-The-Shelf DRAM Chips

Umut Başer^{§†} İsmail Emir Yüksel[§] F. Nisa Bostancı[§] Konstantinos Sgouras[§] Ataberk Olgun[§]
Emre Hakan Demirli^{§†} Zhiheng Yue[§] Harsh Songara[§] Oğuz Ergin^{‡†} Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB ETÜ

[‡]University of Sharjah

We experimentally demonstrate that it is possible to generate unique, repeatable, and device-specific signatures suitable for use as Physical Unclonable Function (PUF) responses in commercial off-the-shelf (COTS) DRAM chips by leveraging simultaneous multiple-row activation (SiMRA). Based on a rigorous experimental characterization of 112 modern DDR4 DRAM chips (from 10 modules), we introduce SiMRA-PUF, the first DRAM-based PUF that uses SiMRA-generated signatures as PUF responses. We analyze SiMRA-PUF in terms of reliability, uniqueness, and evaluation latency for varying numbers of simultaneously activated DRAM rows (i.e., 2, 4, 8, 16, and 32), DRAM chip density & die revision, and evaluate how temperature affects the similarity of SiMRA-generated responses. Among our 8 key experimental observations, we highlight two major results. First, SiMRA-PUF provides average intra-Jaccard indices of 89.02%, 89.81%, 93.03%, 94.06%, and 94.86%, and average inter-Jaccard indices of 3.98%, 2.37%, 3.44%, 2.92%, and 3.24% for 2-, 4-, 8-, 16-, and 32-row activations, respectively, showing that SiMRA-generated signatures are both repeatable within a device and unique across devices. Second, SiMRA-PUF provides a 5.75% lower evaluation latency than the state-of-the-art COTS DRAM-based PUF [39].

we first experimentally characterize PUF responses generated using SiMRA in 112 COTS DDR4 DRAM chips across two major parameters: 1) the number of simultaneously activated rows, and 2) DRAM chip density & revision. We analyze how temperature affects the similarity of SiMRA-generated responses. Second, we evaluate the quality of the generated PUF responses using standard PUF metrics (i.e., inter- and intra-Jaccard indices [25, 26, 32, 33, 40–42]) and evaluation latency for varying numbers of simultaneously activated rows.

Based on our extensive real DRAM chip experiments, we make eight new empirical observations. We summarize our analysis with two key findings. First, we show that SiMRA-PUF provides average intra-Jaccard indices of 89.02%, 89.81%, 93.03%, 94.06%, and 94.86%, and average inter-Jaccard indices of 3.98%, 2.37%, 3.44%, 2.92%, and 3.24% for 2-, 4-, 8-, 16-, and 32-row activations, respectively. Second, the 2-row activation-based SiMRA-PUF leads to 5.75% lower evaluation latency than the state-of-the-art COTS DRAM-based PUF [39]. These results demonstrate that SiMRA-PUF can be used as a fast, low-overhead PUF on real COTS DRAM chips.

FC-DRAM

- 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), Edinburgh, UK, March 2024.
[\[Slides \(pptx\) \(pdf\)\]](#)
[\[arXiv version\]](#)
[\[FCDRAM Source Code\]](#)

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 Yağlıkcı
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

ingly significant bottleneck across a wide variety of computing 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.

The Capability of COTS DRAM Chips

We demonstrate that COTS DRAM chips:

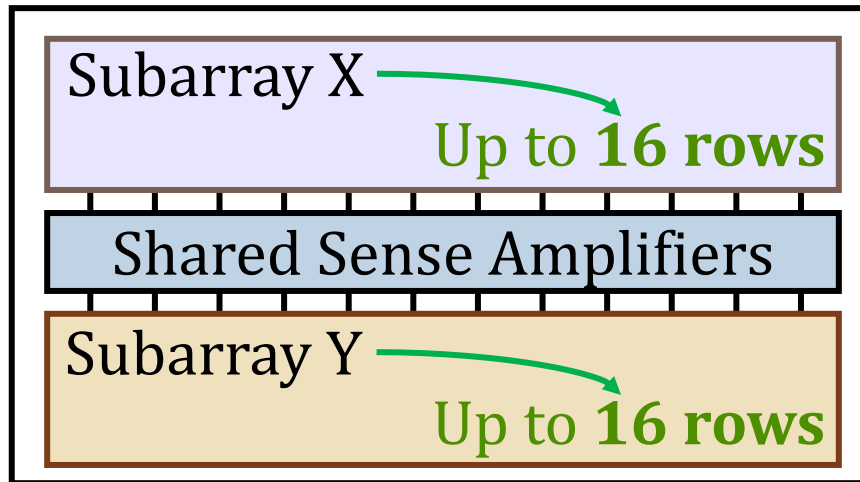
- 1 Can simultaneously activate up to 48 rows in two neighboring subarrays
- 2 Can perform **NOT operation** with up to **32 output operands**
- 3 Can perform up to **16-input AND, NAND, OR, and NOR** operations

48 Simultaneously Activated Rows

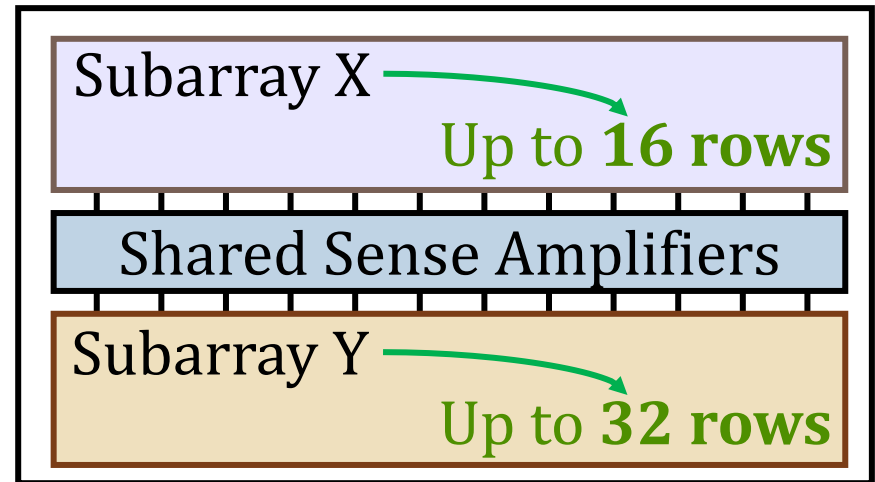
COTS DRAM chips have **two distinct** sets of activation patterns in **neighboring subarrays** when two rows are activated with **violated timings**

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



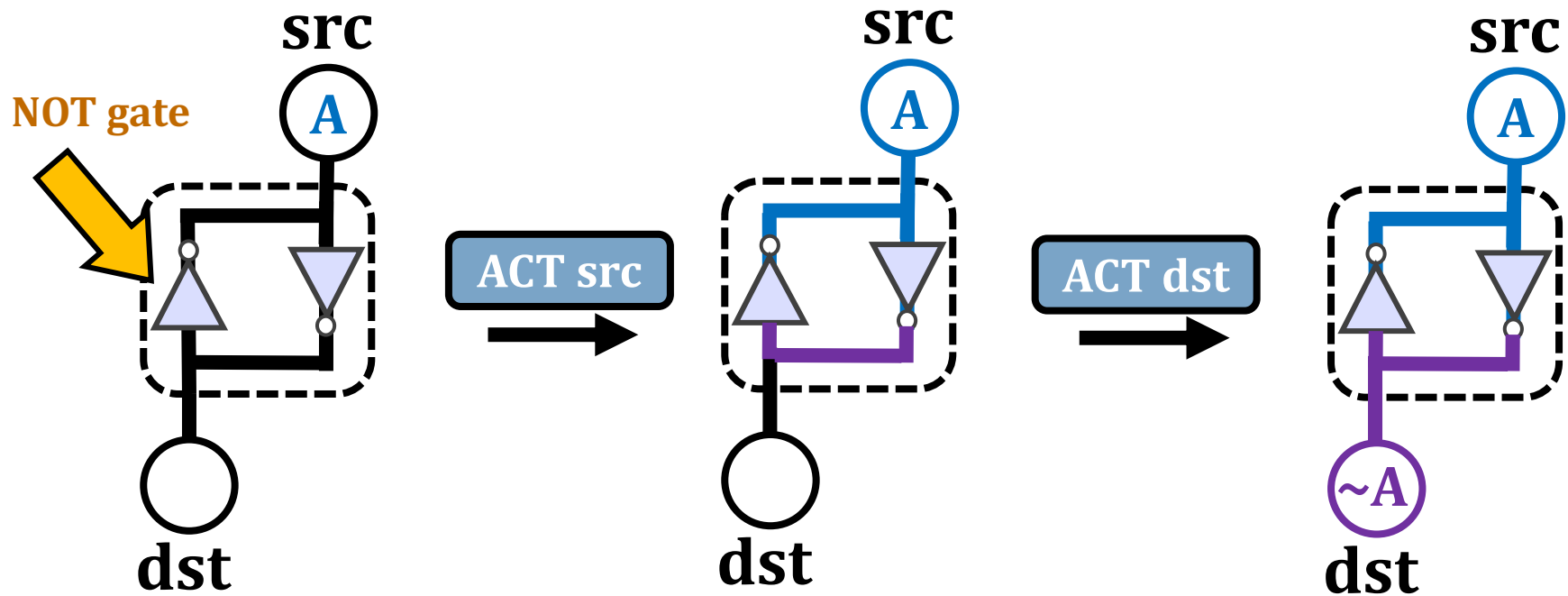
A total of **32 rows**



A total of **48 rows**

Bitwise NOT: Key Idea

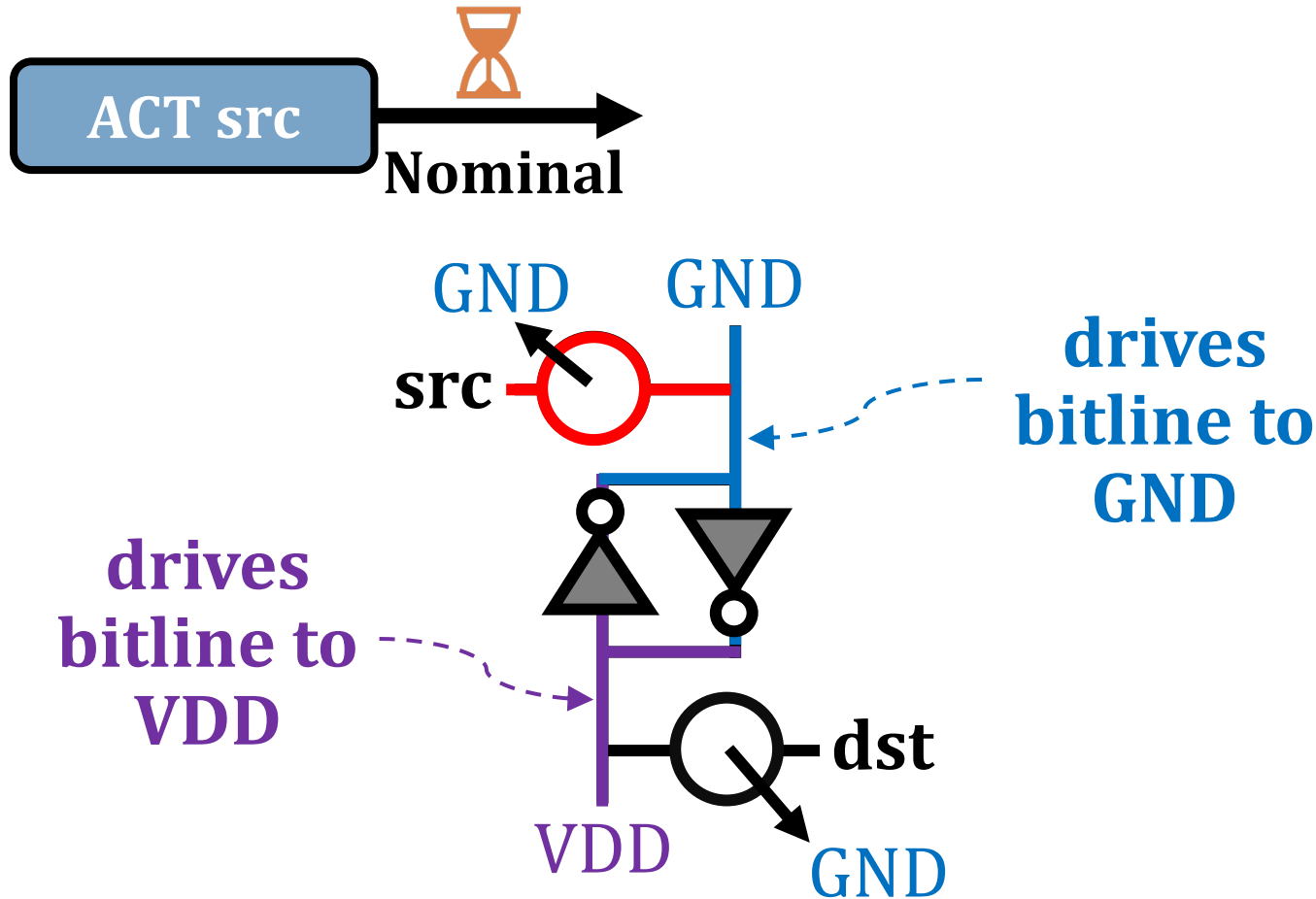
Connect rows in neighboring subarrays through a **NOT gate** by simultaneously activating rows



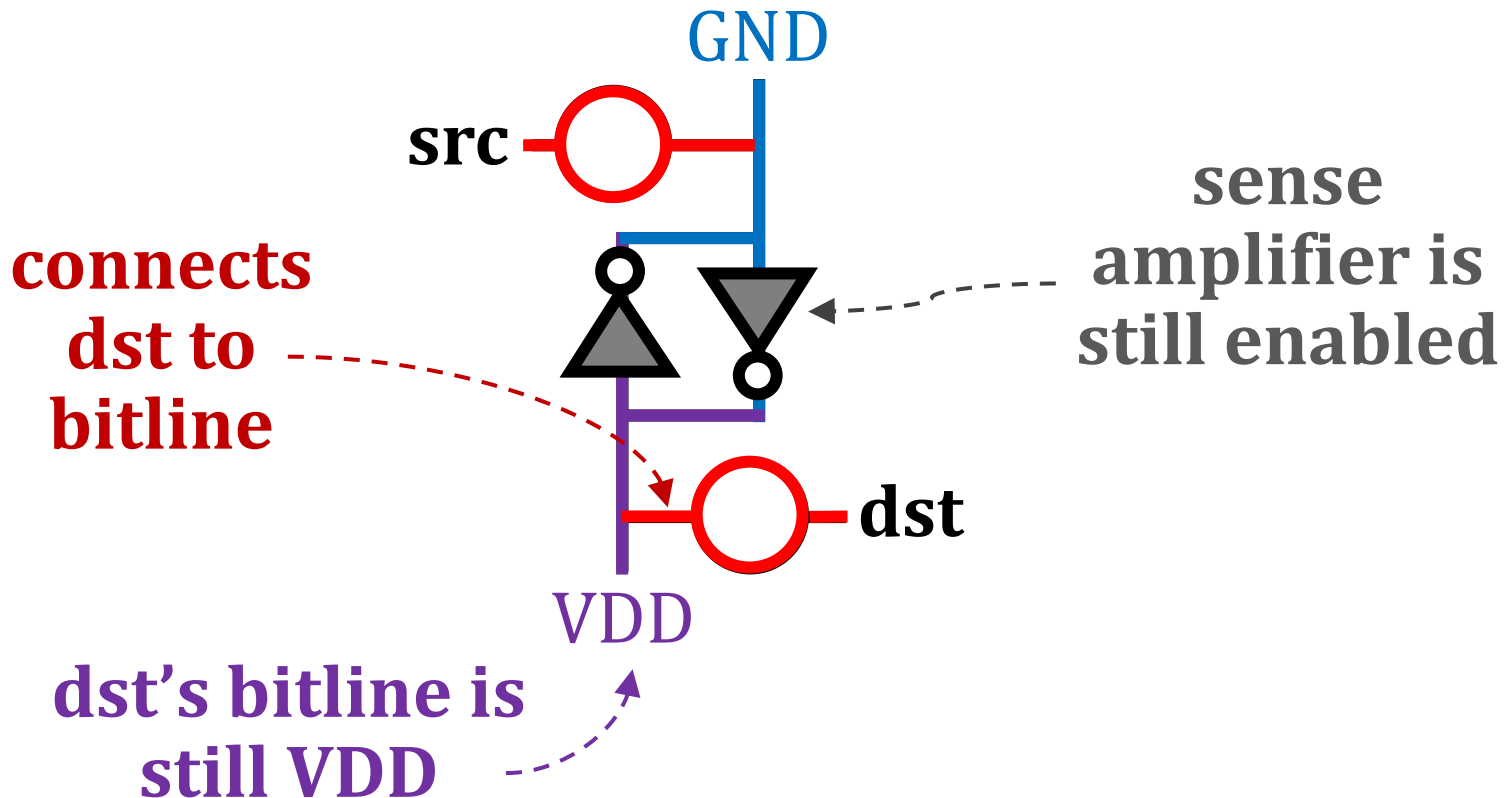
ACT src



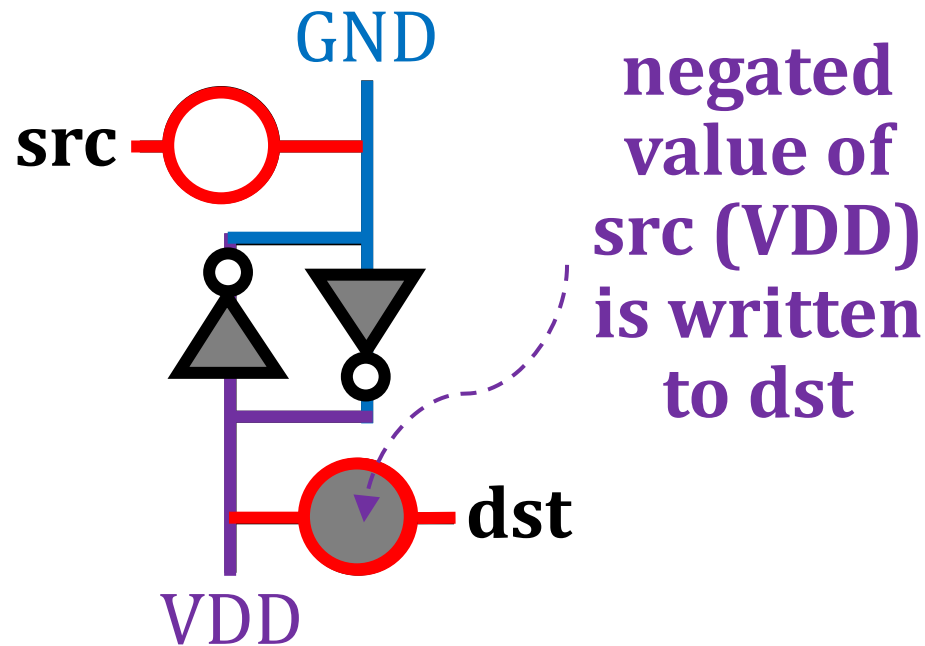
NOT Operation: A Walkthrough



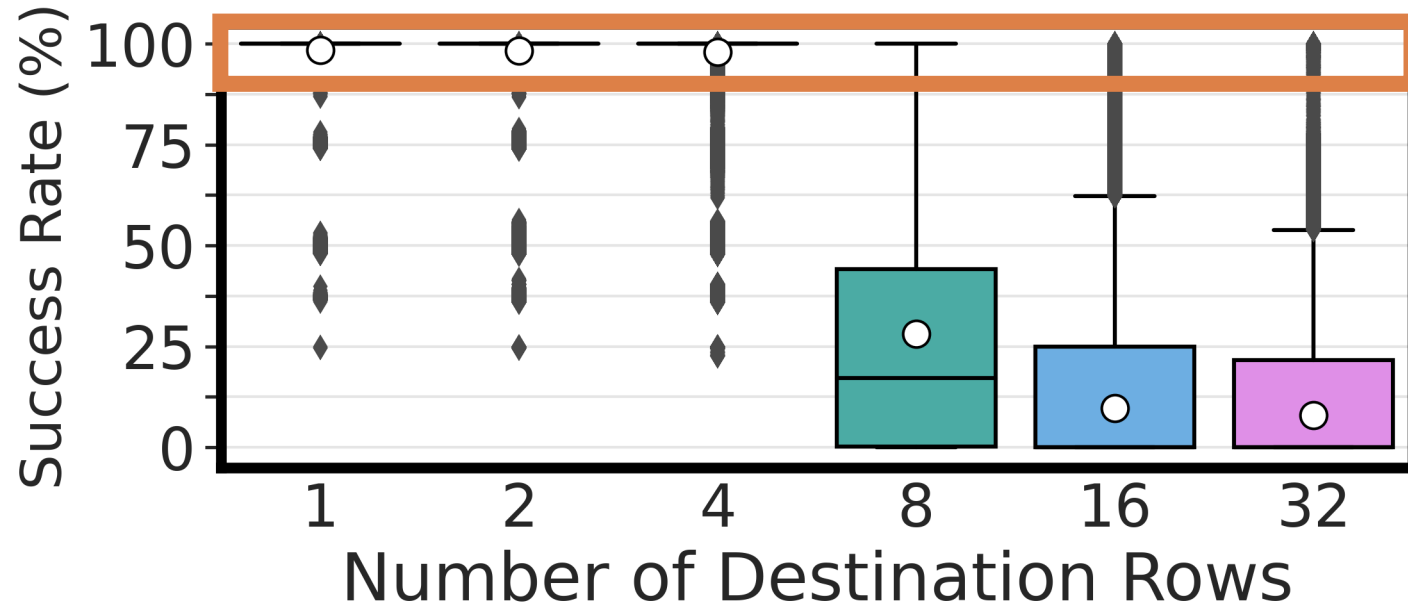
NOT Operation: A Walkthrough



NOT Operation: A Walkthrough



Performing NOT in COTS DRAM Chips

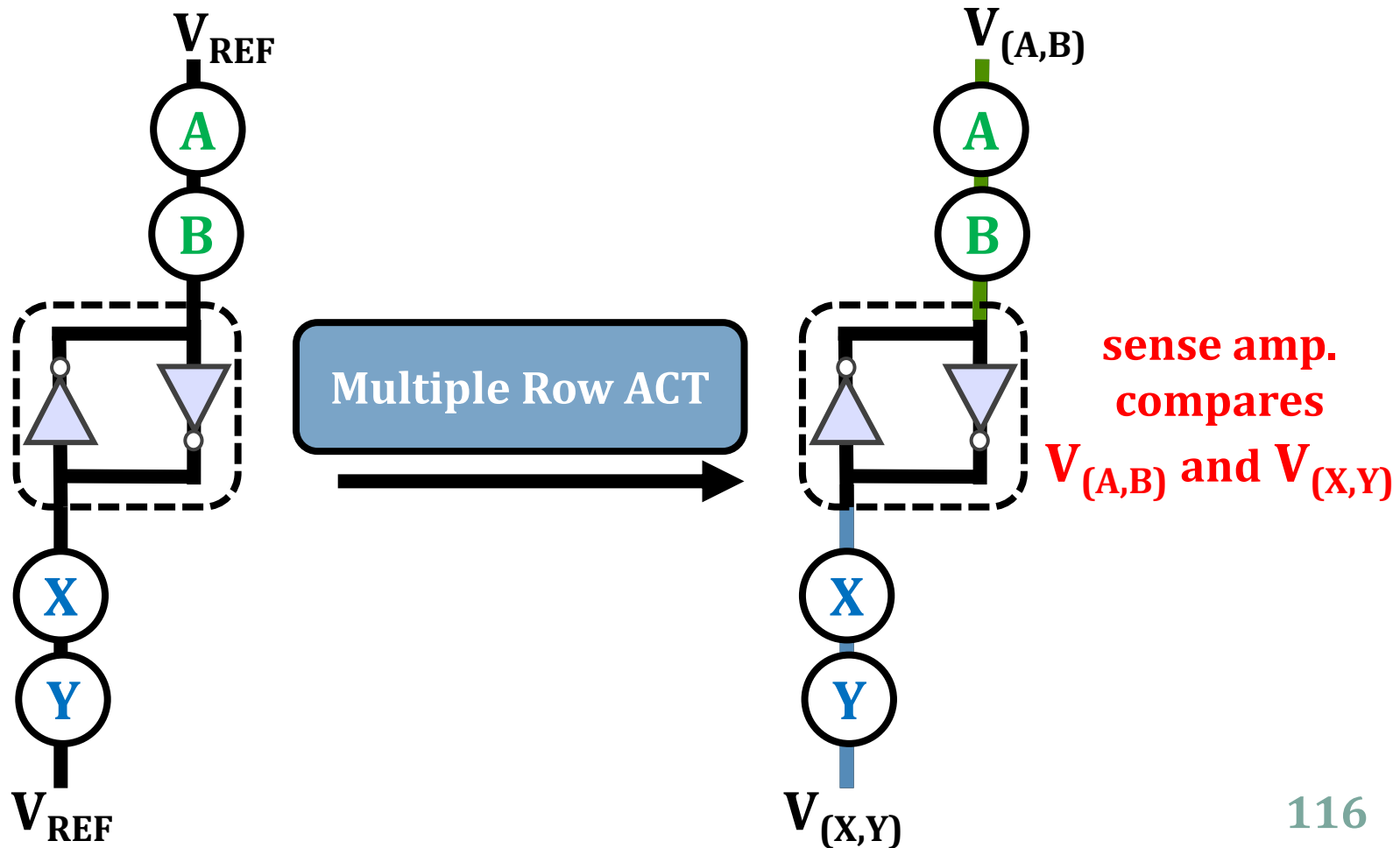


There is at least one DRAM cell that can perform the NOT operation with a 100% success rate

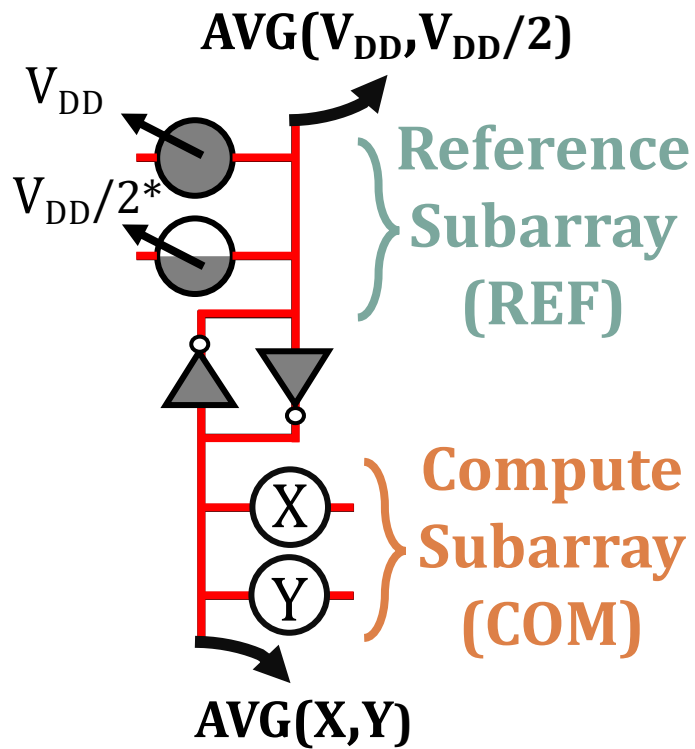
COTS DRAM chips can perform NOT operations with up to 32 destination rows

16-Input NAND/NOR/AND/OR

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



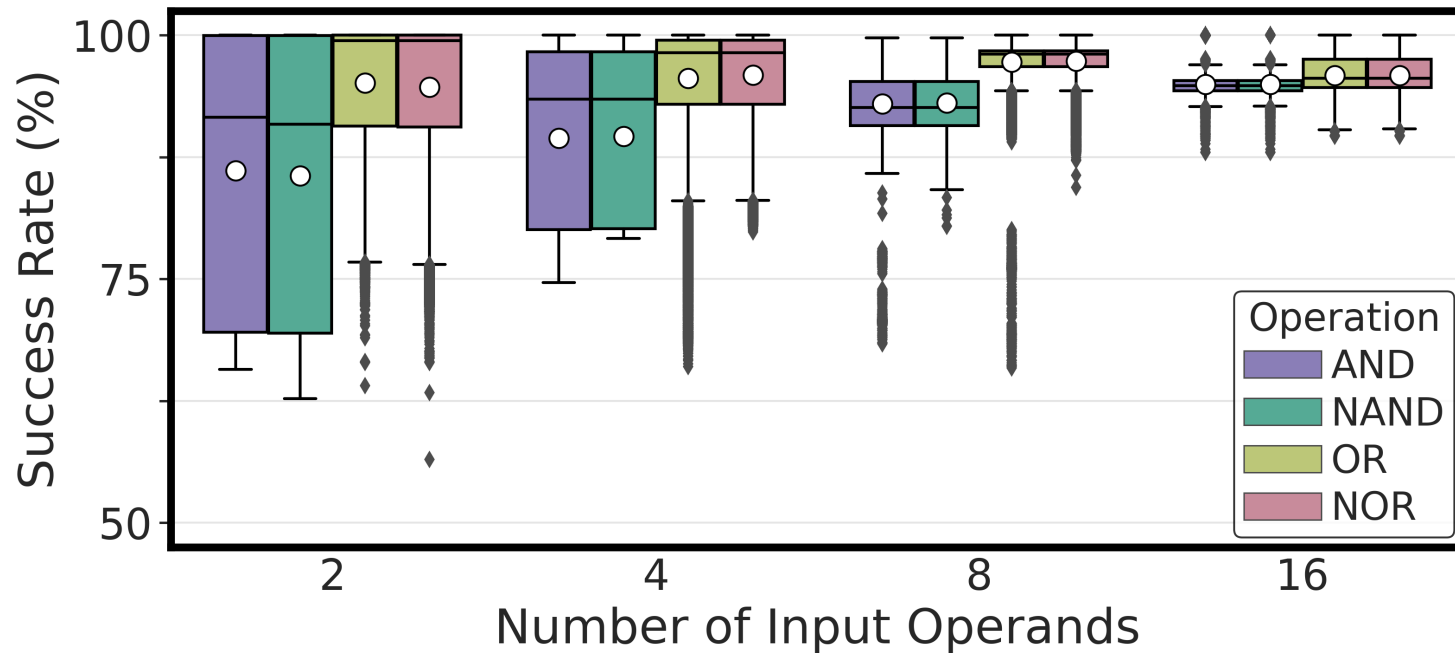
Two-Input AND and NAND Operations



$V_{DD}=1$ & $\text{GND} = 0$

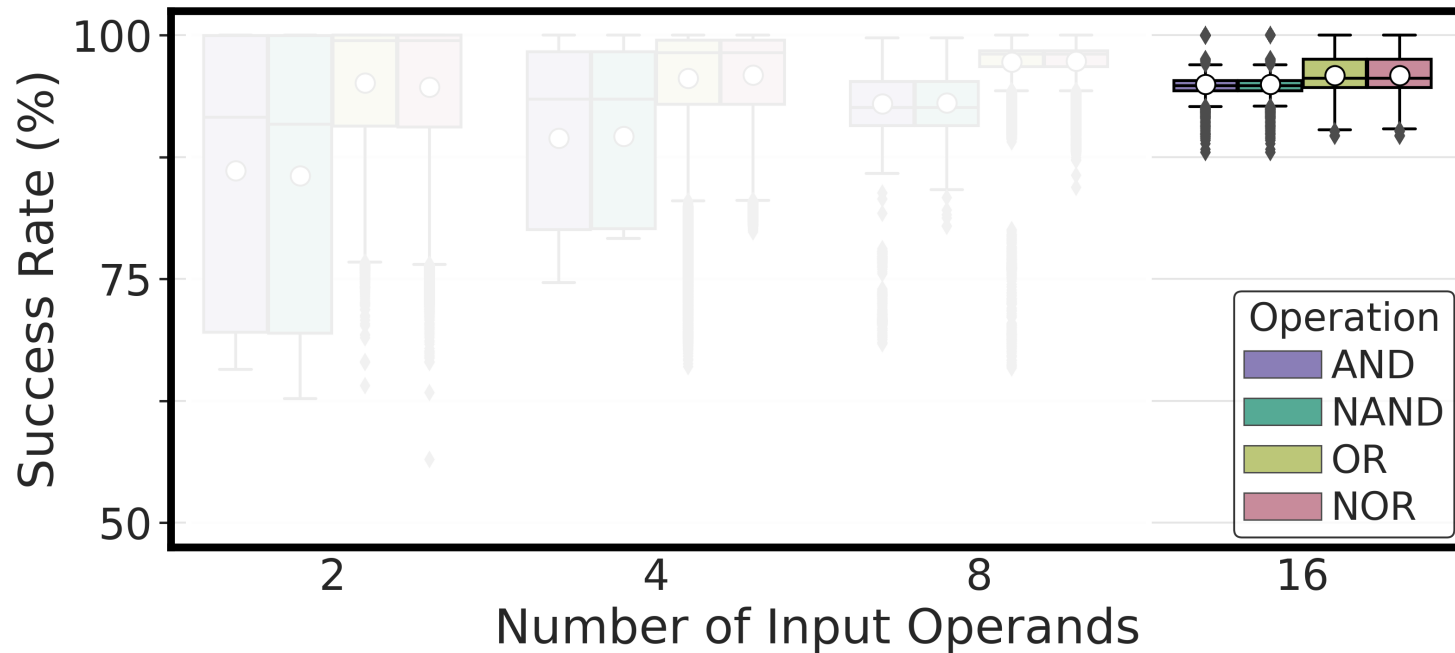
X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
		AND	NAND

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%)**

FC-DRAM

- 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 \(HPCA\)](#), Edinburgh, UK, March 2024.
[\[Slides \(pptx\) \(pdf\)\]](#)
[\[arXiv version\]](#)
[\[FCDRAM Source Code\]](#)

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 Yağlıkcı
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

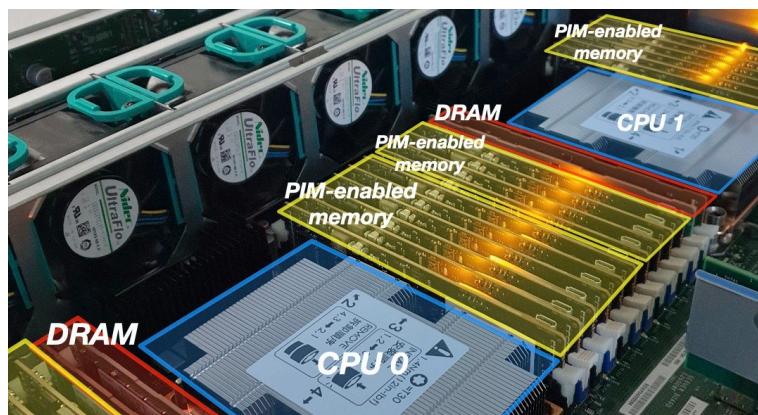
ingly significant bottleneck across a wide variety of computing systems and applications [12, 13]. Processing-using-DRAM (PuD) [29–32] is a promising paradigm that can alleviate the data movement bottleneck. PuD uses the analog operational properties of the DRAM circuitry to enable massively parallel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

ISCA 2026



June 27– July 1, 2026

6th Workshop on Memory-Centric Computing Systems



*More information &
call for papers:*



MCCSys

*A full-day workshop @ ISCA 2026
28th June 2026 (Sunday)*

Organizers: Ismail Emir Yuksel, F. Nisa Bostanci,
Ataberk Olgun, Prof. Onur Mutlu

- Overview of PIM & PIM Taxonomy
- PIM in Memory & Storage
- Infrastructures for PIM Research
- Real-World PnM Systems
- PuM for Bulk Bitwise Operations
- Programming Techniques & Tools
- Research Challenges & Opportunities

SAFARI

<https://events.safari.ethz.ch/isca26-mccsys/>

DRAM Bender
Enables Learning
DRAM Internals

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,
"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[\[Slides \(pptx\) \(pdf\)\]](#)
[\[Short Talk Slides \(pptx\) \(pdf\)\]](#)
[\[Lightning Talk Slides \(pptx\) \(pdf\)\]](#) [\[Talk Video \(25 minutes\)\]](#)
[\[Lightning Talk Video \(100 seconds\)\]](#) [\[arXiv version\]](#)

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan[†]

[†]ETH Zürich

Yahya Can Tuğrul^{†‡}

Kaveh Razavi[†]

[‡]TOBB University of Economics & Technology

Jeremie S. Kim[†]

Onur Mutlu[†]

^σQualcomm Technologies Inc.

Victor van der Veen^σ

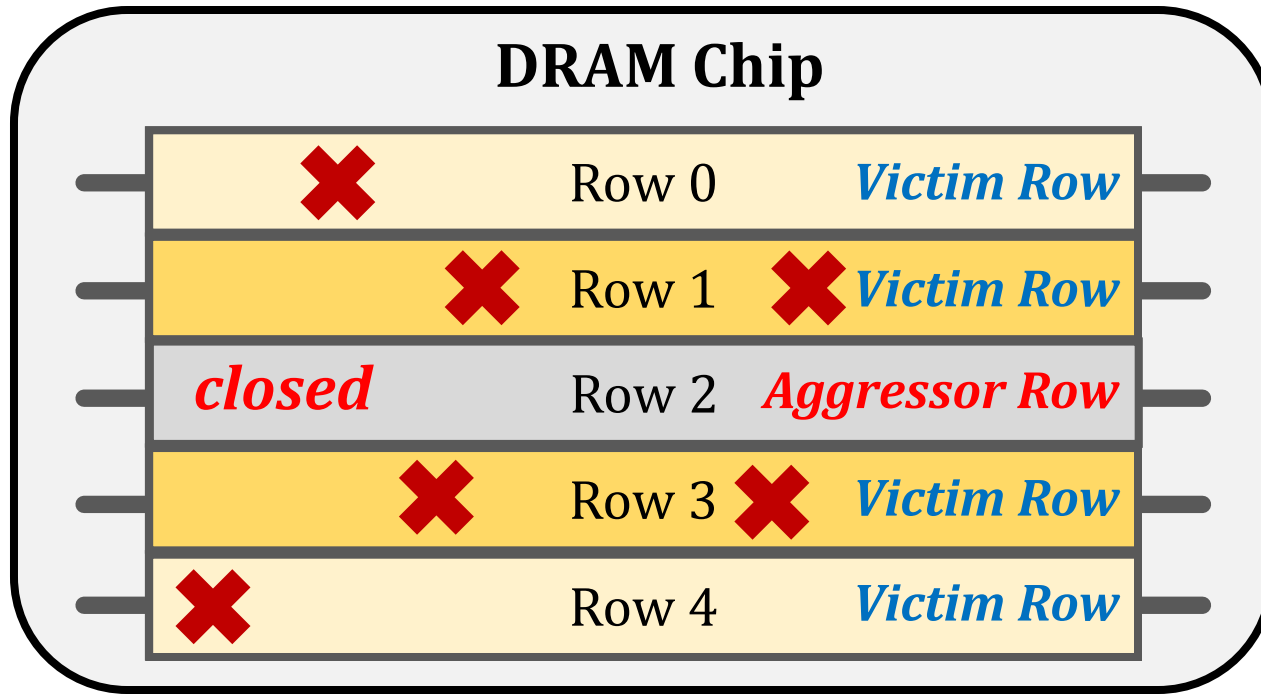
Abstract

The RowHammer vulnerability in DRAM is a critical threat to system security. To protect against RowHammer, vendors commit to security-through-obscurity: modern DRAM chips rely on un-

1 Introduction

DRAM [21] has long been the dominant memory technology used in almost all computing systems due to its low latency and low cost per bit. DRAM vendors still push DRAM technology scaling

The RowHammer Vulnerability

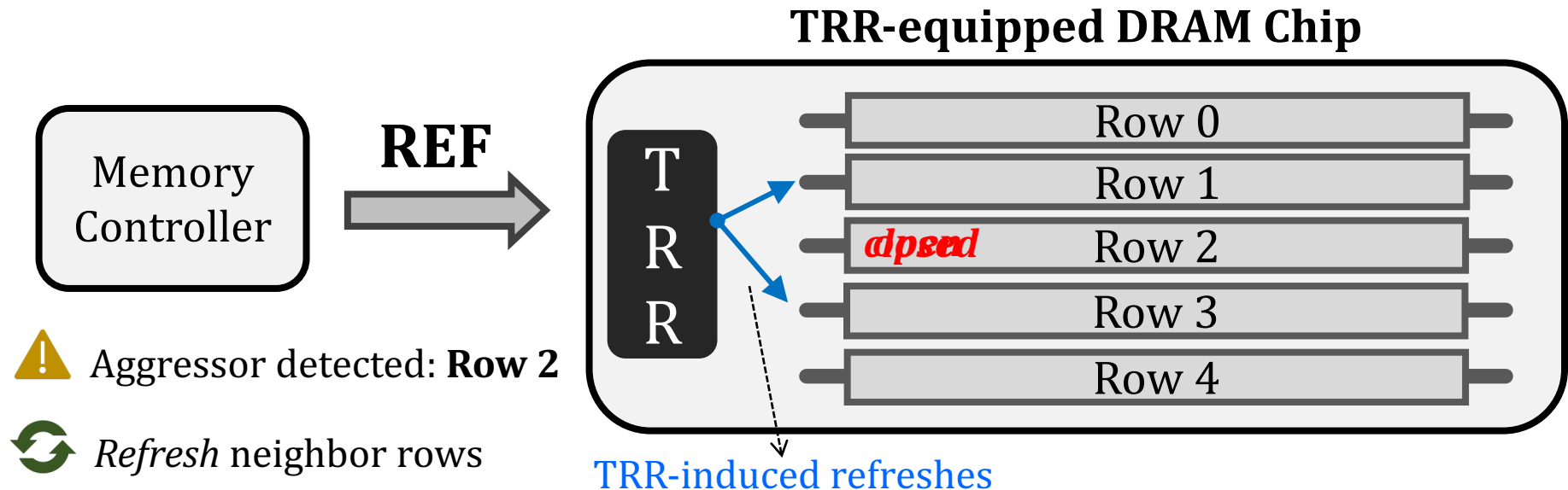


Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **RowHammer bit flips** in nearby cells

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



Goal

Study in-DRAM TRR mechanisms to

- 1 **understand** how they operate
- 2 **assess** their security
- 3 **secure** DRAM completely against RowHammer

U-TRR: Key Idea

Hassan et al., "[Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications](#)," in MICRO, 2021.

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan[†]

[†]ETH Zürich

Yahya Can Tuğrul^{†‡}

Kaveh Razavi[†]

[‡]TOBB University of Economics & Technology

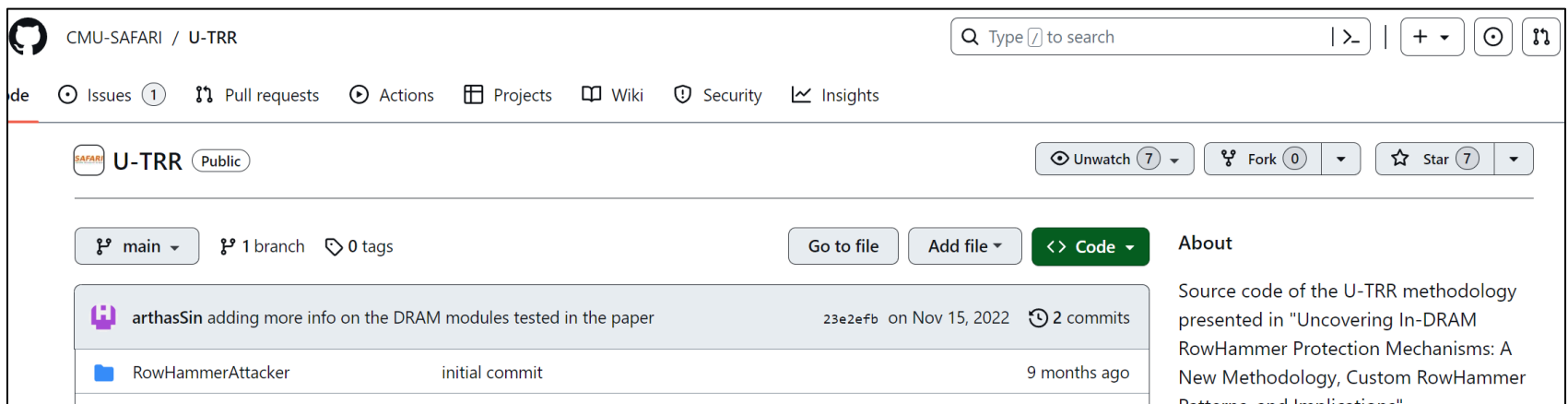
Jeremie S. Kim[†]

Onur Mutlu[†]

^σQualcomm Technologies Inc.

Victor van der Veen^σ

Key idea: Use **data retention failures** as a side channel to **detect when a row is refreshed** by on-die mitigation



Experimental Methodology

1. Identify a row (R) with T retention time
2. Wait for $T/2$
3. Hammer $R+1$ once
4. Issue a periodic REF command (trigger mitigation)
5. Wait for $T/2$, read out row R and check for bitflips

Aggressor Row $R + 1$

Victim Row R

On-DRAM-die
Mitigation

Refresh victim row



Experimental Methodology

1. Identify a row (R) with T retention time

Row R experiences no bitflips
only if on-DRAM-die mitigation exists

4. Issue a periodic REF command (trigger mitigation)

5. Wait for $T/2$, read out row R and check for bitflips

Aggressor Row $R + 1$

Victim Row R

On-DRAM-die
Mitigation

Refresh victim row



Experimental Methodology

1. Identify a row (R) with T retention time

Row R experiences no bitflips
only if on-DRAM-die mitigation exists

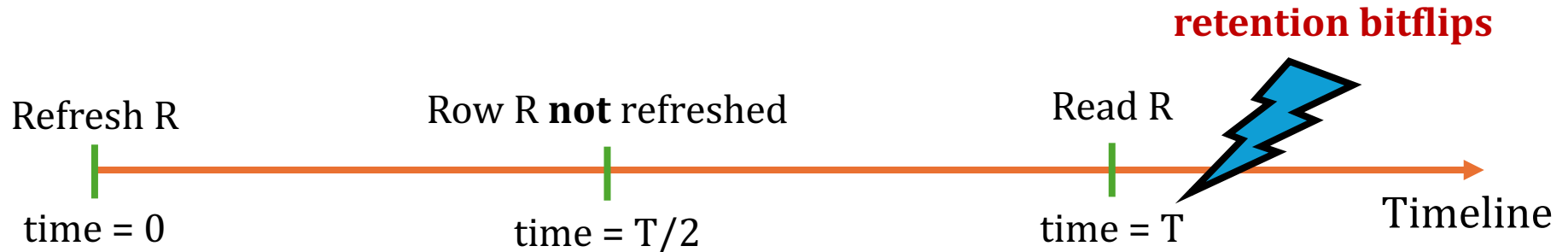
4. Issue a periodic REF command (trigger mitigation)

5. Wait for $T/2$, read out row R and check for bitflips

Row R experiences retention bitflips
if not refreshed at $T/2$

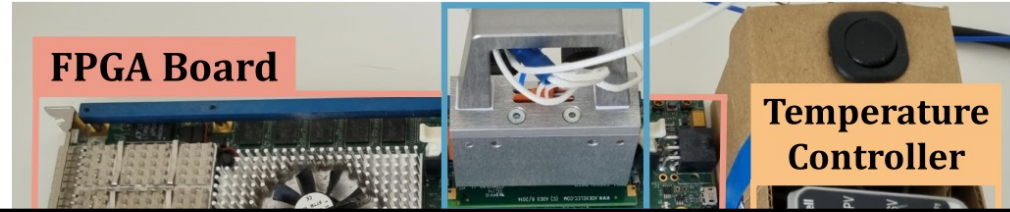
Victim Row R

Refresh victim row



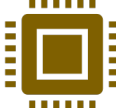
DRAM Testing Infrastructure

We implement **U-TRR** using
FPGA-based *SoftMC* [Hassan+, HPCA'17]
modified to support DDR4 DRAM



Module	Date (yy-ww)	Chip Density (Gbit)	Organization			$HC_{first}^{\dagger}$	Our Key TRR Observations and Results							
			Ranks	Banks	Pins		Version	Aggressor Detection	Aggressor Capacity	Per-Bank TRR	TRR-to-REF Ratio	Neighbors Refreshed	% Vulnerable DRAM Rows [†]	Max. Bit Flips per Row per Hammer [†]
A0	19-50	8	1	16	8	16K	$ATRR_1$	Counter-based	16	✓	1/9	4	73.3%	1.16
A1-5	19-36	8	1	8	16	13K-15K	$ATRR_1$	Counter-based	16	✓	1/9	4	99.2% - 99.4%	2.32 - 4.73
A6-7	19-45	8	1	8	16	13K-15K	$ATRR_1$	Counter-based	16	✓	1/9	4	99.3% - 99.4%	2.12 - 3.86
A8-9	20-07	8	1	16	8	12K-14K	$ATRR_1$	Counter-based	16	✓	1/9	4	74.6% - 75.0%	1.96 - 2.96
A10-12	19-51	8	1	16	8	12K-13K	$ATRR_1$	Counter-based	16	✓	1/9	4	74.6% - 75.0%	1.48 - 2.86
A13-14	20-31	8	1	8	16	11K-14K	$ATRR_2$	Counter-based	16	✓	1/9	2	94.3% - 98.6%	1.53 - 2.78
B0	18-22	4	1	16	8	44K	$BTRR_1$	Sampling-based	1	✗	1/4	2	99.9%	2.13
B1-4	20-17	4	1	16	8	159K-192K	$BTRR_1$	Sampling-based	1	✗	1/4	2	23.3% - 51.2%	0.06 - 0.11
B5-6	16-48	4	1	16	8	44K-50K	$BTRR_1$	Sampling-based	1	✗	1/4	2	99.9%	1.85 - 2.03
B7	19-06	8	2	16	8	20K	$BTRR_1$	Sampling-based	1	✗	1/4	2	99.9%	31.14
B8	18-03	4	1	16	8	43K	$BTRR_1$	Sampling-based	1	✗	1/4	2	99.9%	2.57
B9-12	19-48	8	1	16	8	42K-65K	$BTRR_2$	Sampling-based	1	✗	1/9	2	36.3% - 38.9%	16.83 - 24.26
B13-14	20-08	4	1	16	8	11K-14K	$BTRR_3$	Sampling-based	1	✓	1/2	4	99.9%	16.20 - 18.12
C0-3	16-48	4	1	16	x8	137K-194K	$CTRR_1$	Mix	Unknown	✓	1/17	2	1.0% - 23.2%	0.05 - 0.15
C4-6	17-12	8	1	16	x8	130K-150K	$CTRR_1$	Mix	Unknown	✓	1/17	2	7.8% - 12.0%	0.06 - 0.08
C7-8	20-31	8	1	8	x16	40K-44K	$CTRR_1$	Mix	Unknown	✓	1/17	2	39.8% - 41.8%	9.66 - 14.56
C9-11	20-31	8	1	8	x16	42K-53K	$CTRR_2$	Mix	Unknown	✓	1/9	2	99.7%	9.30 - 32.04
C12-14	20-46	16	1	8	x16	6K-7K	$CTRR_3$	Mix	Unknown	✓	1/8	2	99.9%	4.91 - 12.64

i Table 1 in our paper provides more information about the analyzed modules

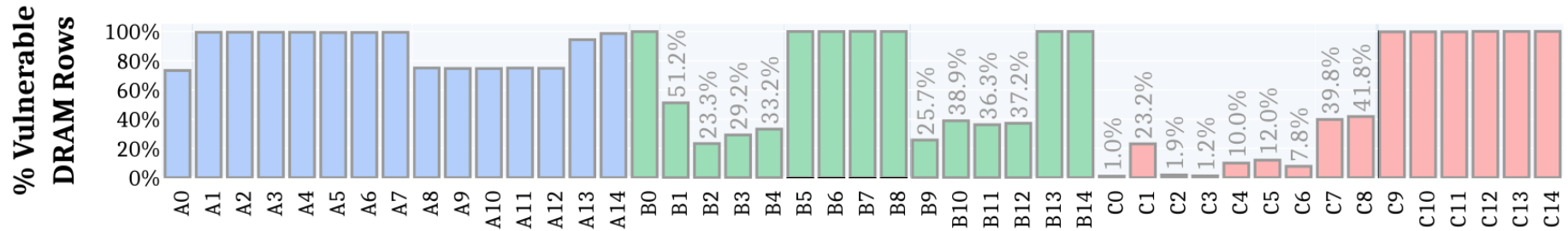
 **15x Vendor C
DDR4 modules**

Bypassing TRR with New RowHammer Access Patterns

We craft **new RowHammer access patterns** that **circumvent TRR** of three major DRAM vendors

On the **45** DDR4 modules we test, the **new access patterns** cause **a large number of RowHammer bit flips**

Effect on Individual Rows



All 45 modules we tested are vulnerable to our new RowHammer access patterns

Our RowHammer access patterns cause bit flips in more than 99.9% of the rows

Why are some modules less vulnerable?

- 1) Fundamentally less vulnerable to RowHammer
- 2) Different TRR mechanisms
- 3) Unique row organization

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,
"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[\[Slides \(pptx\) \(pdf\)\]](#)
[\[Short Talk Slides \(pptx\) \(pdf\)\]](#)
[\[Lightning Talk Slides \(pptx\) \(pdf\)\]](#) [\[Talk Video \(25 minutes\)\]](#)
[\[Lightning Talk Video \(100 seconds\)\]](#) [\[arXiv version\]](#)

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan[†]

[†]ETH Zürich

Yahya Can Tuğrul^{†‡}

Kaveh Razavi[†]

[‡]TOBB University of Economics & Technology

Jeremie S. Kim[†]

Onur Mutlu[†]

^σQualcomm Technologies Inc.

Victor van der Veen^σ

Abstract

The RowHammer vulnerability in DRAM is a critical threat to system security. To protect against RowHammer, vendors commit to security-through-obscurity: modern DRAM chips rely on un-

1 Introduction

DRAM [21] has long been the dominant memory technology used in almost all computing systems due to its low latency and low cost per bit. DRAM vendors still push DRAM technology scaling

DRAMScope: Uncovering DRAM Microarchitecture and Characteristics by Issuing Memory Commands

Hwayong Nam[†], Seungmin Baek[†], Minbok Wi[†], Michael Jaemin Kim[†], Jaehyun Park[†], Chihun Song[‡],
Nam Sung Kim[‡], Jung Ho Ahn[†]

[†]Seoul National University, [‡]University of Illinois at Urbana-Champaign

[†]{nhy4916, qortmdalss, homakaka, michael604, wogus20002, gajh}@snu.ac.kr, [‡]{chihuns2, nskim}@illinois.edu

Abstract—The demand for precise information on DRAM microarchitectures and error characteristics has surged, driven by the need to explore processing in memory, enhance reliability, and mitigate security vulnerability. Nonetheless, DRAM manufacturers have disclosed only a limited amount of information, making it difficult to find specific information on their DRAM microarchitectures. This paper addresses this gap by presenting more rigorous findings on the microarchitectures of commodity DRAM chips and their impacts on the characteristics of activate-induced bitflips (AIBs), such as RowHammer and RowPress. The previous studies have also attempted to understand the DRAM microarchitectures and associated behaviors, but we have found some of their results to be misled by inaccurate address mapping and internal data swizzling, or lack of a deeper understanding of the modern DRAM cell structure. For accurate and efficient reverse-engineering, we use three tools: AIBs, retention time test, and RowCopy, which can be cross-validated. With these

To fill this critical gap, a large body of prior work has exploited creative reverse-engineering methodologies. They have relied on scarcely disclosed knowledge or assumptions [4], [17], [18], [21], [43], [44] to uncover error characteristics [4], [25], [29], [36], [39], [50], undefined DRAM operations [62], [82], microarchitectural components transparent to memory controllers, such as AIB protection solutions [9], [13] or on-die ECC [54], [55], to list a few. Nonetheless, we have found a number of previous efforts to discover the DRAM microarchitecture are limited in scope, outdated, or even misleading due to an insufficient understanding of the modern DRAM 6F² cell structure (see Figure 2), complex mapping of CPU physical addresses to DRAM addresses, and swizzling of CPU data within DRAM.¹

Research Highlights Conclusion

- DRAM Bender enables many **studies, ideas,** and **methodologies** in the design of future memory systems
 - Understanding DRAM robustness problems
 - Uncovering new DRAM functionality
 - Learning DRAM internals
 - ...
- We attribute DRAM Bender's widespread success in **enabling research** to its **flexibility** and **ease of use**
- **Research** drives **new DRAM Bender features** & improvements
 - DDR3 → DDR4 → HBM2
 - Support for 6 FPGA boards out of the box

The 3rd
Ramulator & DRAM Bender Tutorial
DRAM Bender Research Highlights

Ataberk Olgun

ICS 2026
6 July 2026

The 3rd

Ramulator & DRAM Bender Tutorial

DRAM Bender Live Demo
& Hardware/Software Overview

Ataberk Olgun

ICS 2026

6 July 2026

RowHammer Using DRAM Bender

- Connect to a DRAM Bender host machine via SSH



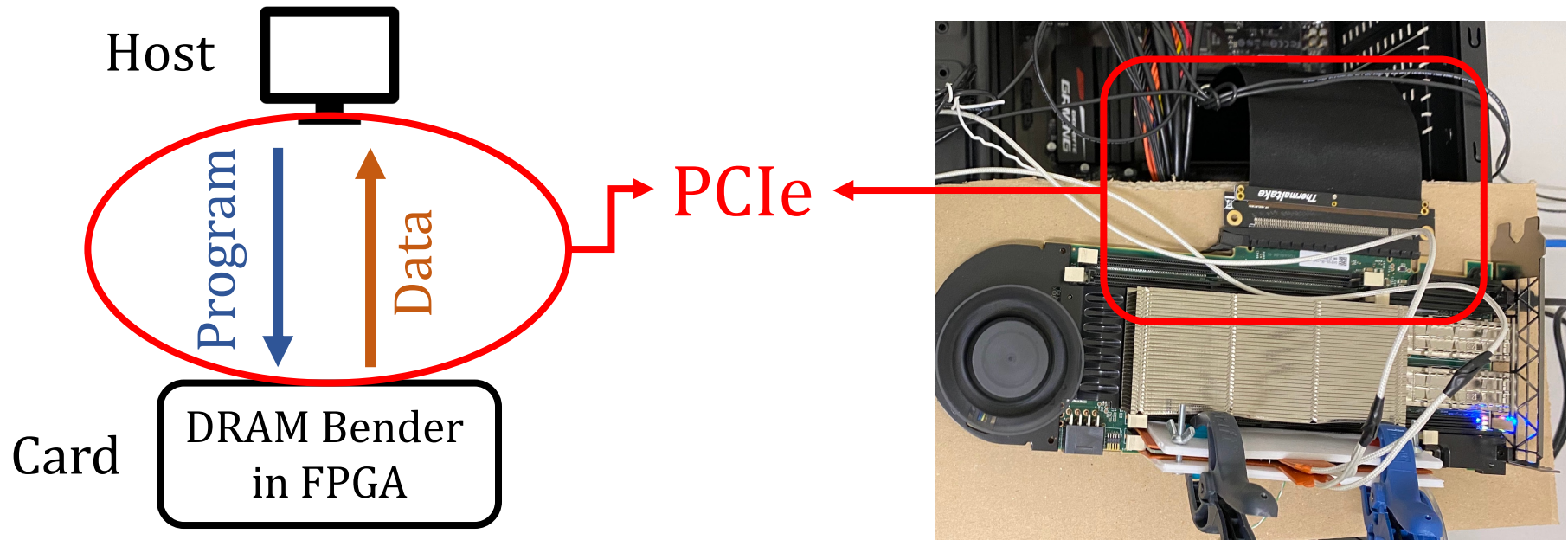
- Execute a program to **hammer a DRAM row**
 - Vary program parameters
- Understand how the program gets executed

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- 1 Initialize DRAM Bender platform
- 2 Create DRAM Bender test program
- 3 Execute DRAM Bender test program
- 4 Read & analyze experimental data

Aside: Transferring Programs & Data



- Host to card (H2C) transfer: Write to a file
- Card to host (C2H) transfer: Read from a file

```
aolgun@safari-fpga45:~/DRAM-Bender$ ls -l /dev/xdma0_*2*_0
crw-rw---- 1 root smc_user 237, 36 Mar 21 15:38 /dev/xdma0_c2h_0
crw-rw---- 1 root smc_user 237, 32 Mar 21 15:38 /dev/xdma0_h2c_0
```

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

DRAM Bender Programming Model

- **Assembly-like programming** using C++/Python API
- Two “types” of instructions
 - **DRAM command batch** (ACT, READ, WRITE...)
 - **Regular** (arithmetic, control-flow, load/store)
- 13 general purpose 32-bit registers
- 3 special-purpose DRAM address stride registers

Regular DRAM Bender Instructions

Table 2: DRAM Bender ISA description

Type	Instruction	Description
MEM	LD	Load one word from the data scratchpad
	ST	Store one word to the data scratchpad
ARITH.	AND/OR/XOR	Bitwise AND/OR/XOR two registers
	ADD/SUB	Add/Sub two registers
	MV	Move a register's value to another register
	SRC	Cyclic shift a register to right
	LI	Load a register with an immediate value
CNTRL.	BL/BEQ	Branch if less than/equal
	JUMP	Jump to an address
	SLEEP	Stall for a number of cycles
MISC.	LDWD	Load the wide data register
	LDPC	Read a performance counter
	SRE/SRX	Self-refresh entry/exit
	END	Stop executing the program

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

RowHammer Test Program

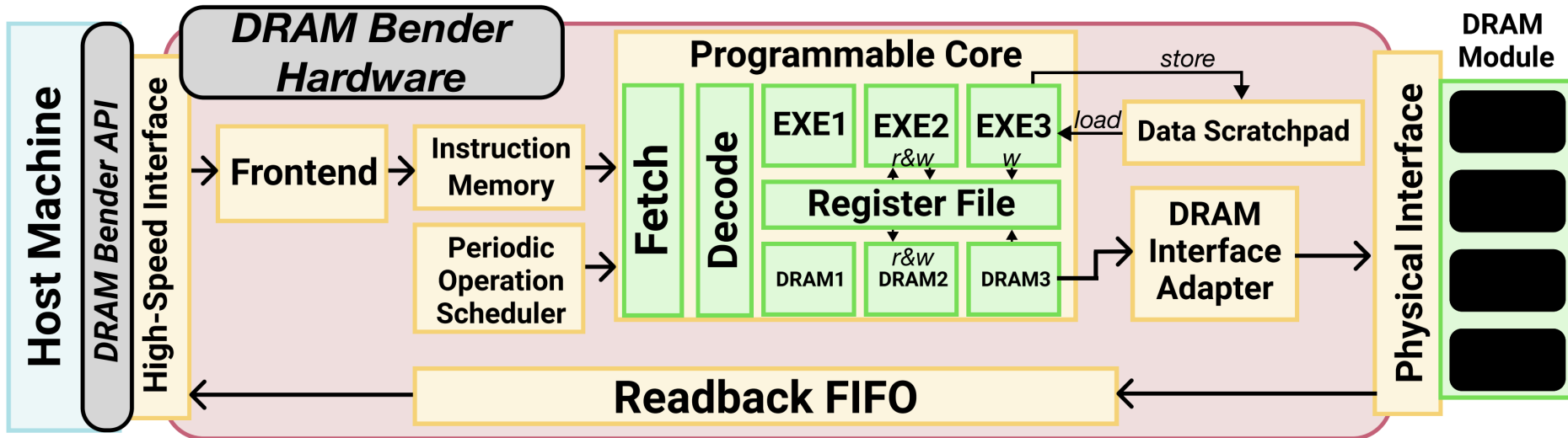
- Create programs by appending instructions to a list
 1. Initialize 8 rows surrounding the victim row
 - Including the two aggressor rows
 2. Initialize the victim row
 3. Hammer the two aggressor rows
 4. Read the victim row

Typical DRAM Bender Test Workflow

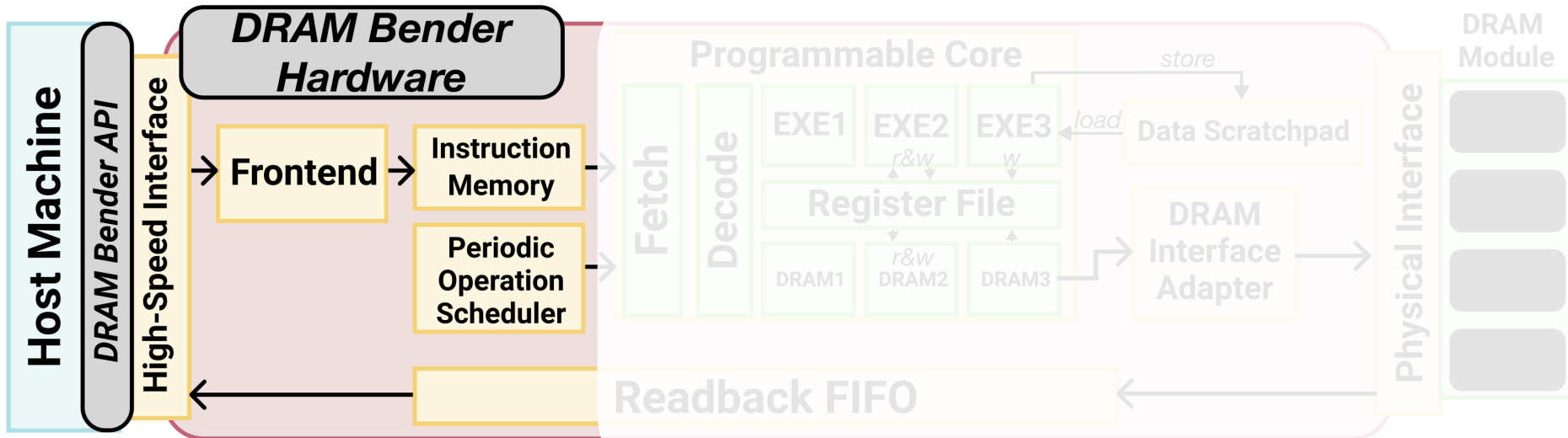
In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

DRAM Bender Hardware Overview

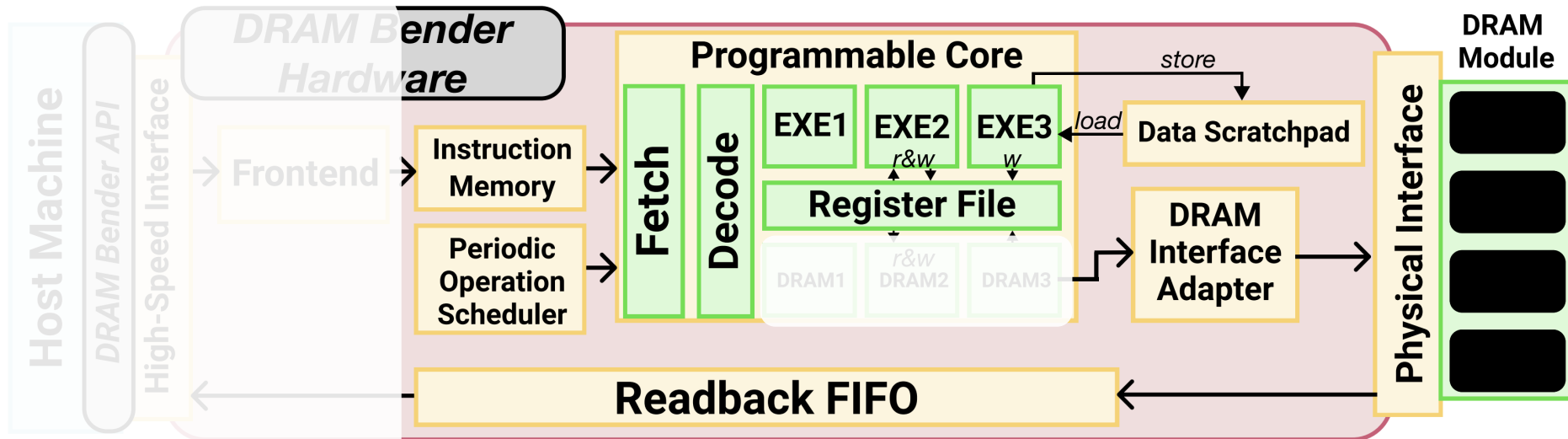


Life of a DRAM Bender Program



- 1 Host machine calls *platform.execute(prog)*
- 2 Frontend stores instructions in instruction memory

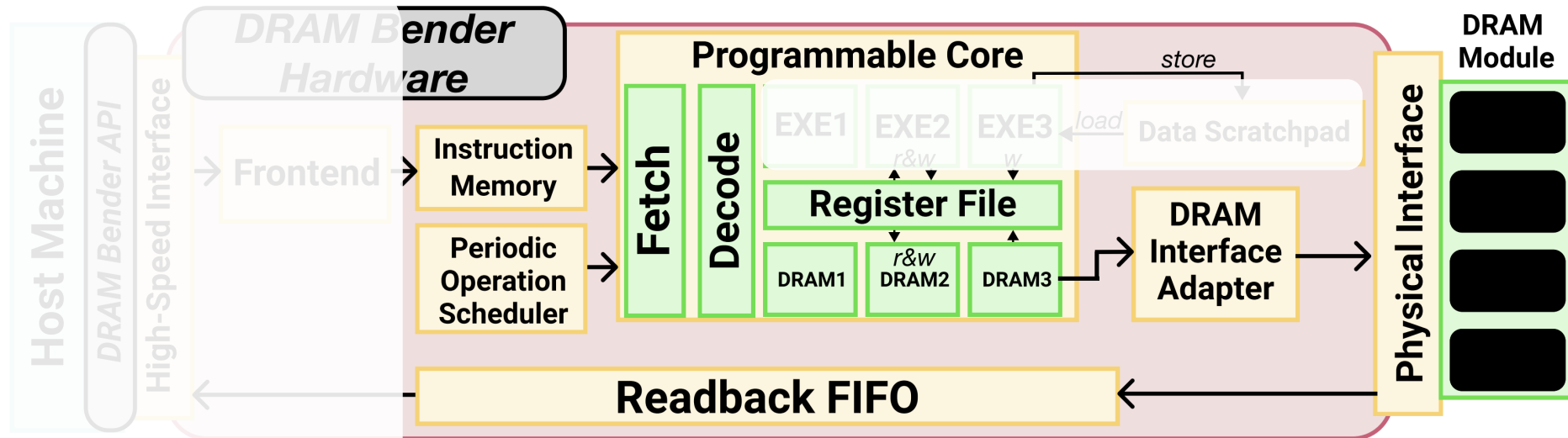
Life of a DRAM Bender Program



3

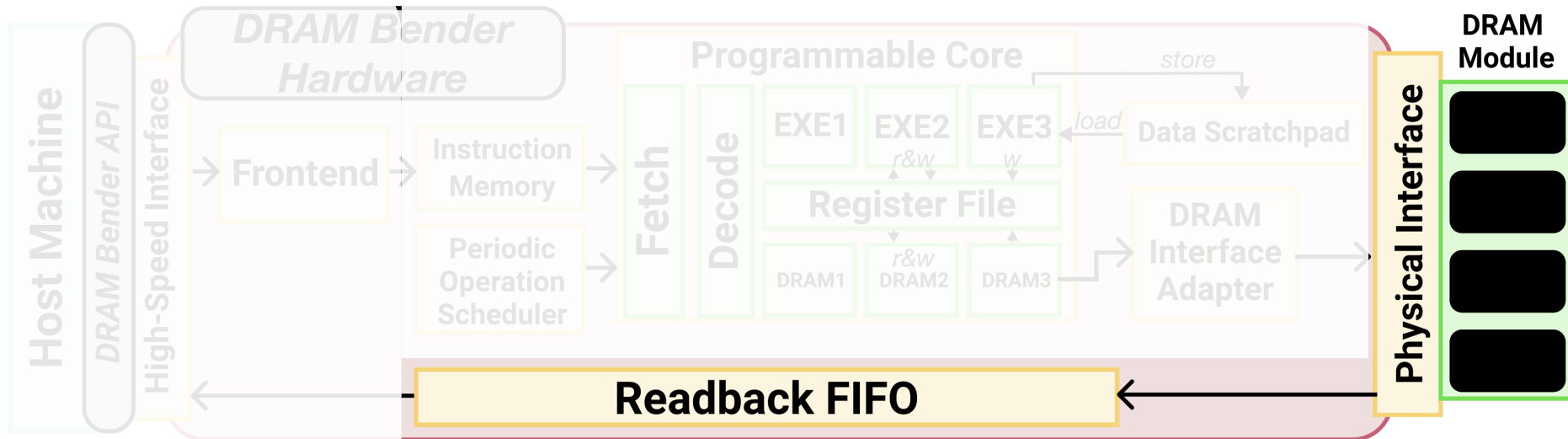
Regular instructions take the *exe* pipeline path

Life of a DRAM Bender Program



- 3 Regular instructions take the *exe* pipeline path
- 4 DRAM command batches take the *dram* pipeline path

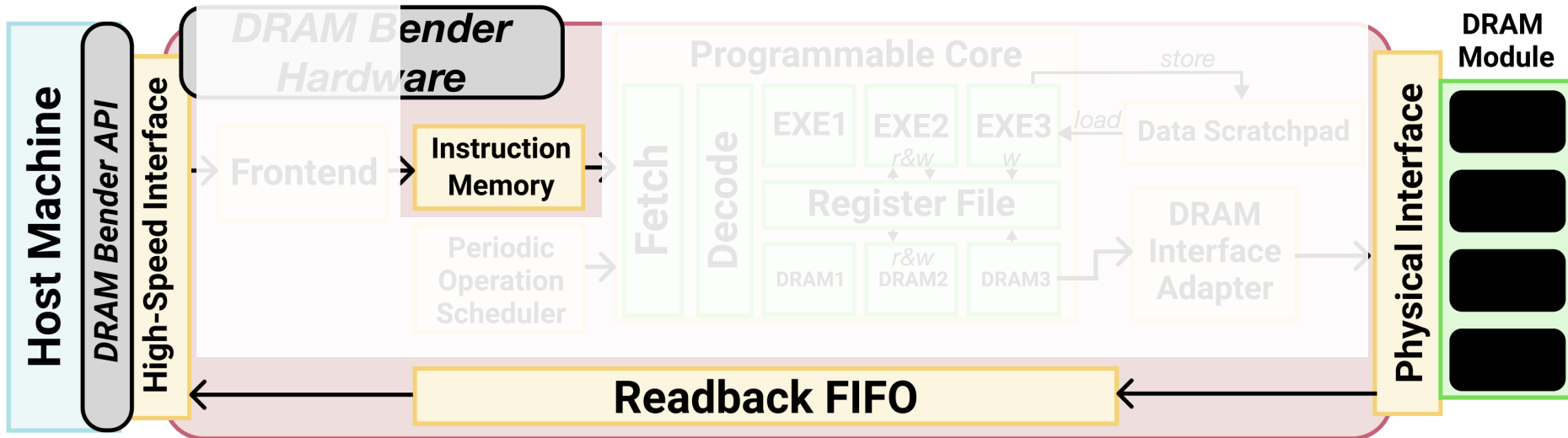
Life of a DRAM Bender Program



5

READ commands in the program fill up the Readback FIFO

Life of a DRAM Bender Program



5

READ commands in the program fill up the Readback FIFO

6

The *END* instruction flushes the Readback FIFO

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

Aside: DRAM Bender Instruction Encoding

https://docs.google.com/spreadsheets/d/18mPiKa1HBo000mzAbWRvo5OnIg_uL6A9tLEIwNeVfOe8

DRAM Bender Instruction Encodings																																							
Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																						
DDR	WRITE	1	0	0	0	iCOL	iBANK	AP	BL4	rs_car [3:0]				rs_bar [3:0]																									
	READ	1	0	0	1	iCOL	iBANK	AP	BL4	rs_car [3:0]				rs_bar [3:0]																									
	PRE	1	0	1	0	PALL	iBANK	X	X	X	X	X	X	rs_bar [3:0]																									
	ACT	1	0	1	1	iROW	iBANK	X	X	rs_rar [3:0]				rs_bar [3:0]																									
	ZQ	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X																					
	REF	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X																					
	DUMMYRD	1	1	1	0	iCOL	iBANK	AP	BL4	rs_car [3:0]				rs_bar [3:0]																									
MEMORY	NOP	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X																						
		63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30				
	INFO	0	0	1	0	0	X	X	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	END	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
MEMORY	LD	0	0	0	1	0	X	X	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	ST	0	0	0	1	0	X	X	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
ARITHMETIC	AND	0	0	0	0	1	X	X	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	OR	0	0	0	0	1	X	X	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	XOR	0	0	0	0	1	X	X	0	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	ADD[I]	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
		0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	SUB[I]	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
		0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	MV	0	0	0	0	0	X	X	0	0	0	0	0	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	SRC	0	0	0	0	0	X	X	0	0	0	0	0	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	LI	0	0	0	0	0	X	X	0	0	0	0	0	0	0	1	1	0	X	X	X	X	X	X	X	immediate [31:16]													
	LDWD	0	0	0	0	0	X	X	0	0	0	0	0	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
LDPC	0	0	0	0	0	X	X	0	0	0	0	0	0	1	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
SELF REFRESH	SRE	0	0	0	0	0	X	X	1	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	SRX	0	0	0	0	0	X	X	1	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
CONTROL FLOW	BL	0	1	0	0	X	X	X	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	BEQ	0	1	0	0	X	X	X	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	JUMP	0	1	0	0	X	X	X	0	0	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	SLEEP	0	1	0	0	X	X	X	0	0	0	0	0	0	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			

Typical DRAM Bender Test Workflow

In a **C++** or a **Python** executable:

- ① Initialize DRAM Bender platform
- ② Create DRAM Bender test program
- ③ Execute DRAM Bender test program
- ④ Read & analyze experimental data

The 3rd

Ramulator & DRAM Bender Tutorial

DRAM Bender Live Demo
& Hardware/Software Overview

Ataberk Olgun

ICS 2026

6 July 2026

SAFARI

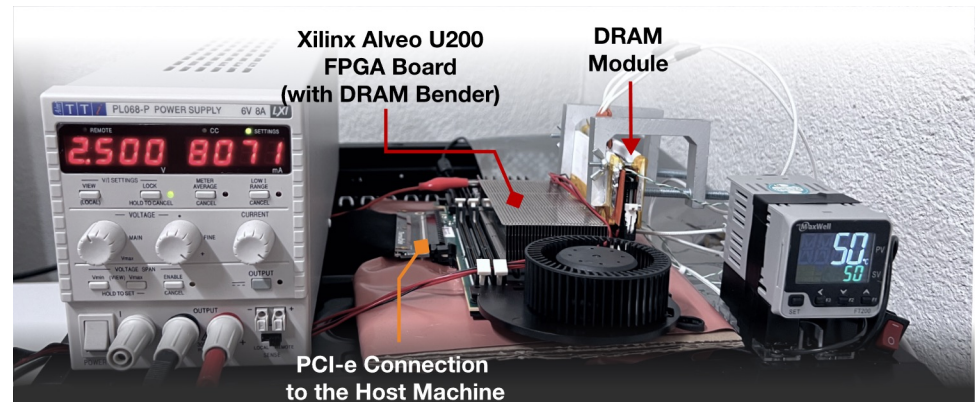
ETH zürich

Summary

DRAM Bender

The first **publicly-available** DDR4/3 & HBM2 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

The 3rd

Ramulator & DRAM Bender Tutorial

Realistic DRAM Timing for HLS: Closing the Memory
Fidelity Gap with Ramulator-Backed Co-Simulation

Dr. Antonino Tumeo

ICS 2026

6 July 2026

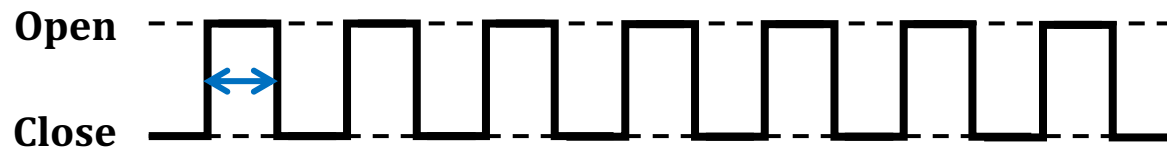
A Highlight: RowPress

Keeping a DRAM row **open for a long time** causes bitflips in adjacent rows

These bitflips do **NOT** require many row activations

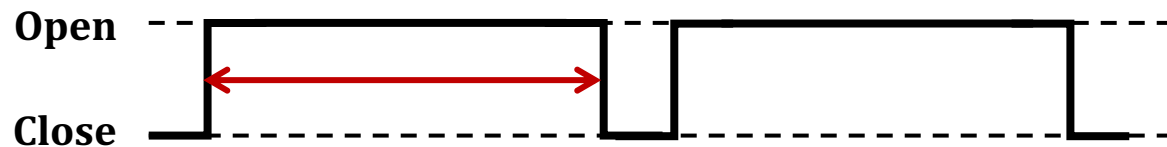
Only one activation is enough in some cases!

**RowHammer
Aggressor Row**



36ns, 47K activations to induce bitflips

**RowPress
Aggressor Row**



7.8 μ s, only 5K activations to induce bitflips

RowPress Results & Source Code

RowPress: Amplifying Read Disturbance in Modern DRAM Chips

Haocong Luo Ataberk Olgun A. Giray Yağlıkçı Yahya Can Tuğrul Steve Rhyner
Meryem Banu Cavlak Joël Lindegger Mohammad Sadrosadati Onur Mutlu

ETH Zürich



Fully open source and artifact evaluated

➤ <https://github.com/CMU-SAFARI/RowPress>





- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,
"RowPress: Amplifying Read Disturbance in Modern DRAM Chips"
Proceedings of the 50th International Symposium on Computer Architecture (ISCA),
Orlando, FL, USA, June 2023.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Video](#) (3 minutes)]
[[RowPress Source Code and Datasets \(Officially Artifact Evaluated with All Badges\)](#)]
Officially artifact evaluated as available, reusable and reproducible.
Best artifact award at ISCA 2023.

RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

Haocong Luo Ataberk Olgun A. Giray Yağlıkçı Yahya Can Tuğrul Steve Rhyner
Meryem Banu Cavlak Joël Lindegger Mohammad Sadrosadati Onur Mutlu
ETH Zürich

Variable Read Disturbance (VRD) Summary

Research Question

- How **accurately** and **efficiently** can we measure the read disturbance threshold (RDT) of **each** DRAM row?

Experimental Characterization

- Record **>100M RDT measurements** across **3750 rows** and **many test parameters** (e.g., temperature, data pattern) in 160 DDR4 and 4 HBM2 chips

Key Observations

- RDT changes **significantly** and **unpredictably** over time: VRD
- **Maximum** observed RDT is **3.5X higher** than **minimum** (for a row)
- **Smallest** RDT (for a row) may appear after **94,467** measurements

Implications for System Security and Robustness

- RDT **cannot** be **accurately identified quickly**
- *Given our **limited dataset**, **guardbands** (>10%) and **ECC** (SECDDED or Chipkill) **may prevent** VRD-induced bitflips at significant **performance cost***
 - **More data and analyses needed to make definitive conclusion**
- Call for **future work** on **understanding** and **efficiently mitigating** VRD

Talk Outline

- I. Motivation
- II. Experimental Characterization Methodology
- III. Foundational Results
- IV. In-Depth Analysis of VRD
- V. Implications for System Security and Robustness
- VI. Conclusion

Talk Outline

I. Motivation

II. Experimental Characterization Methodology

III. Foundational Results

IV. In-Depth Analysis of VRD

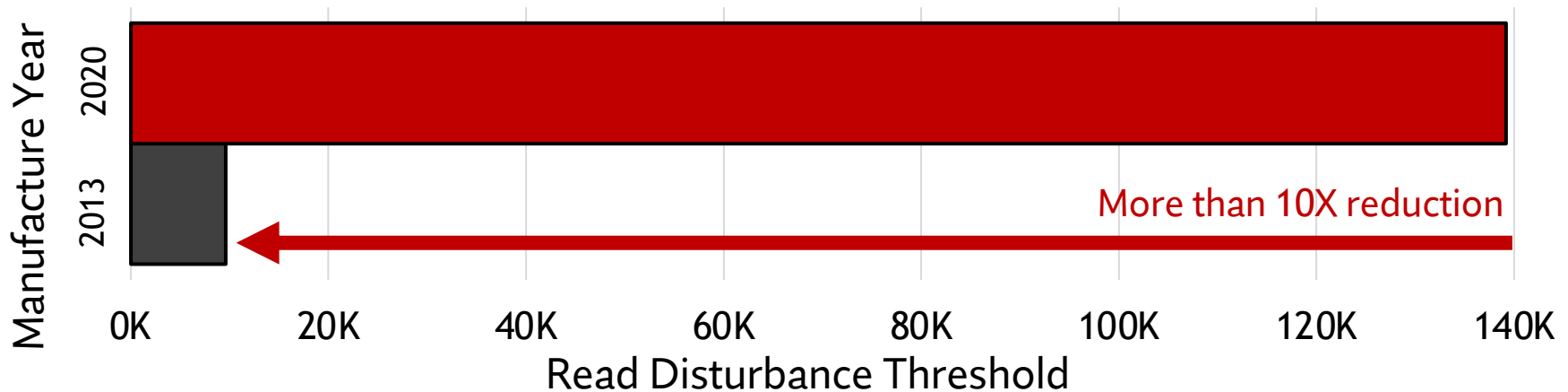
V. Implications for System Security and Robustness

VI. Conclusion

Motivation



DRAM chips are increasingly more vulnerable to read disturbance with technology scaling



Motivation

DRAM read disturbance worsens
as DRAM chip density increases

Existing solutions become more aggressive

Aggressive preventive actions make
existing solutions prohibitively expensive

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows **exhibit VRD**

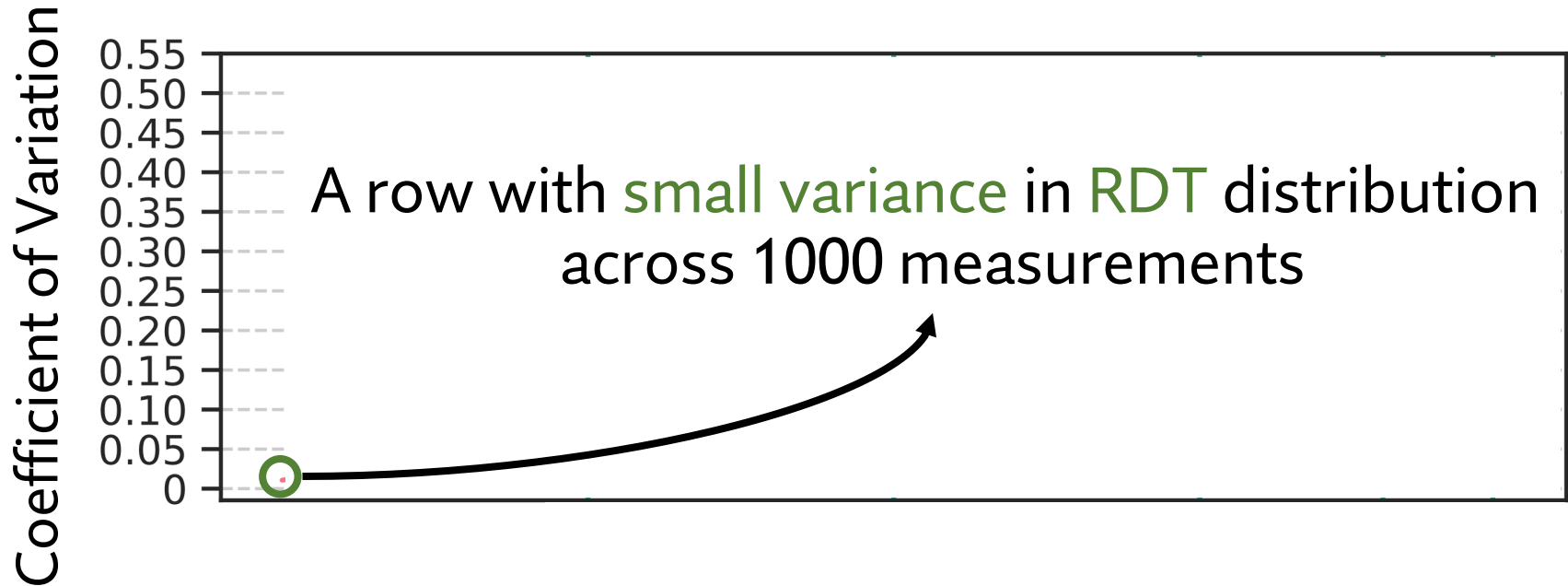
Takeaway 2

Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

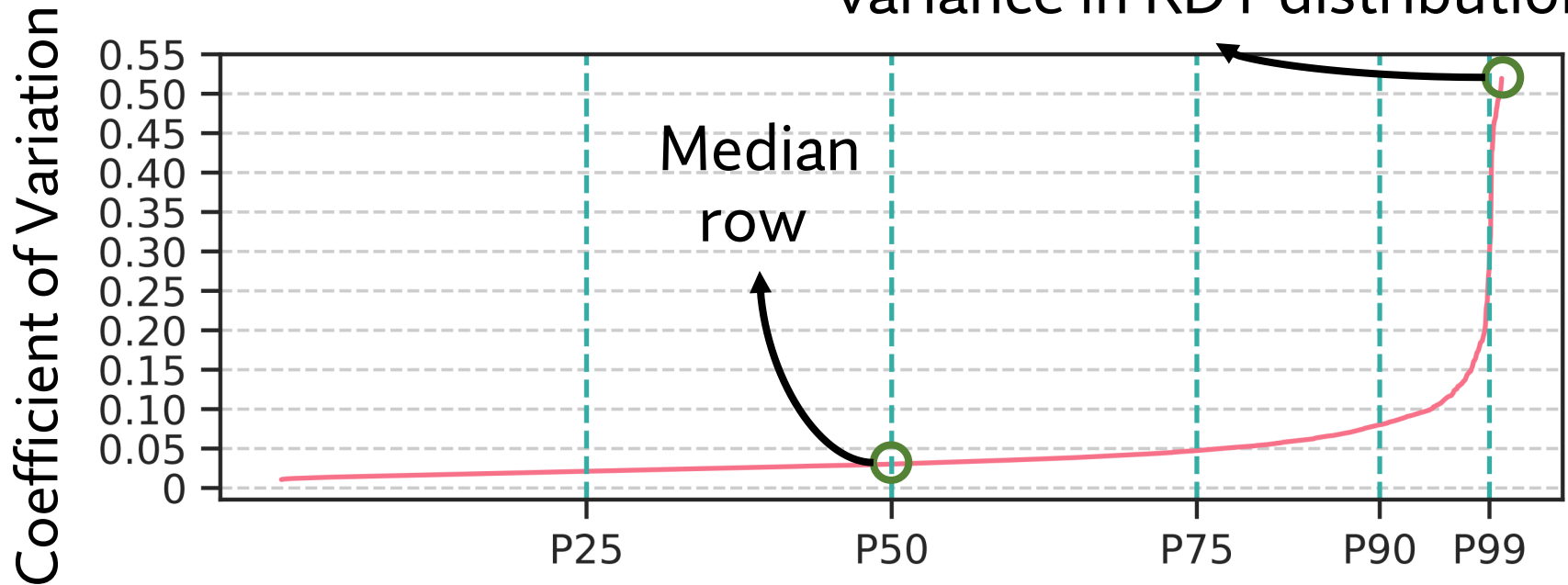
Data patterns, temperature, and aggressor row on time affect VRD

VRD Across DRAM Rows



VRD Across DRAM Rows

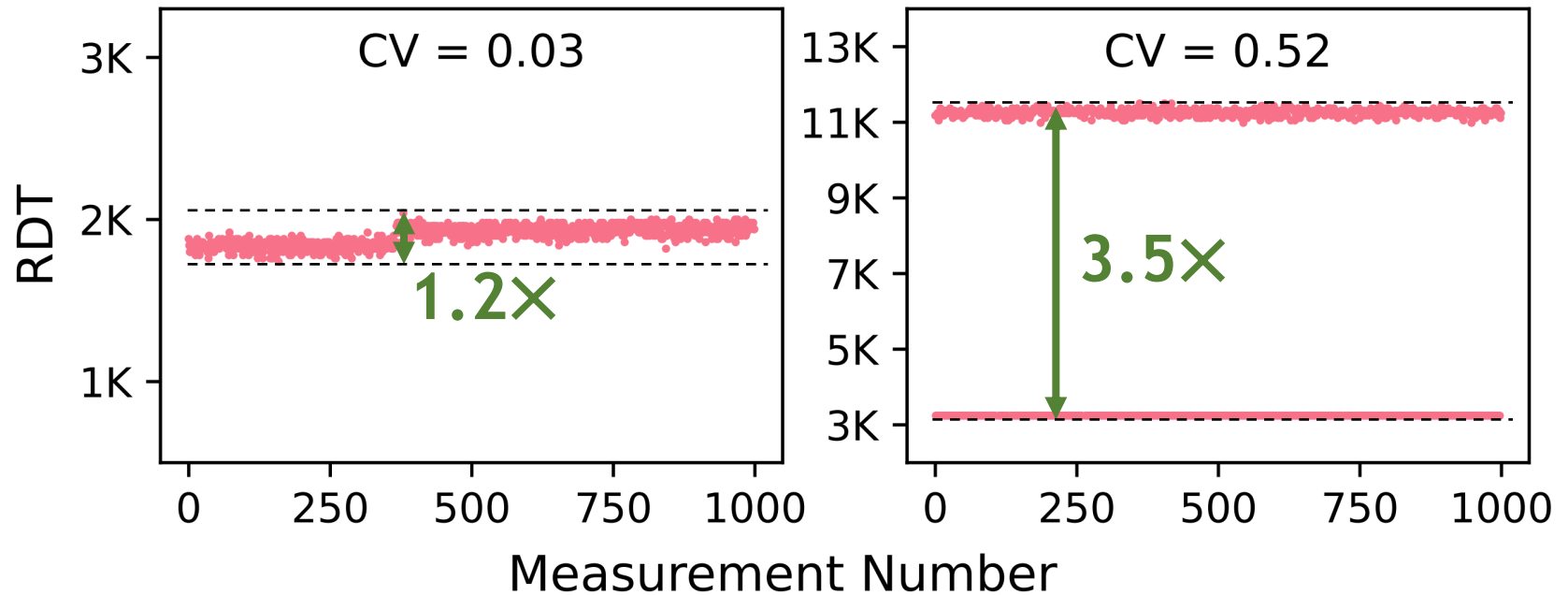
Row with the greatest variance in RDT distribution



DRAM Rows Sorted by Increasing Coefficient of Variation of RDT Across 1000 RDT Measurements

All tested rows **exhibit** VRD

VRD in Two Example Rows



Variation in read disturbance threshold
can reach **3.5x**

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows exhibit VRD

Takeaway 2

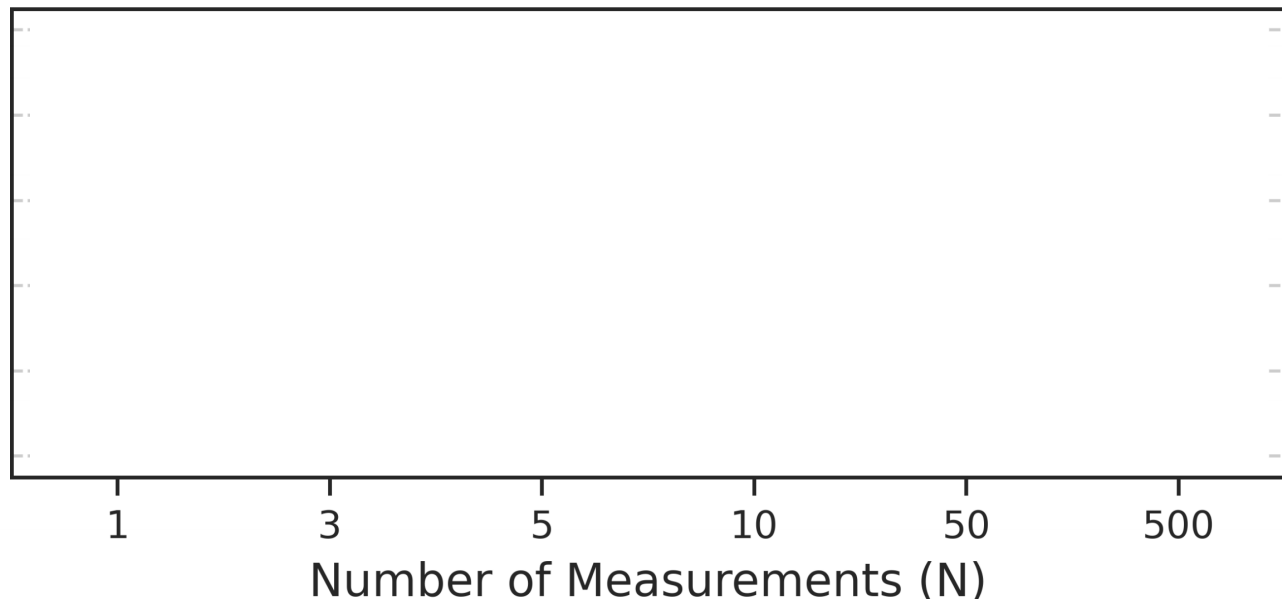
Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

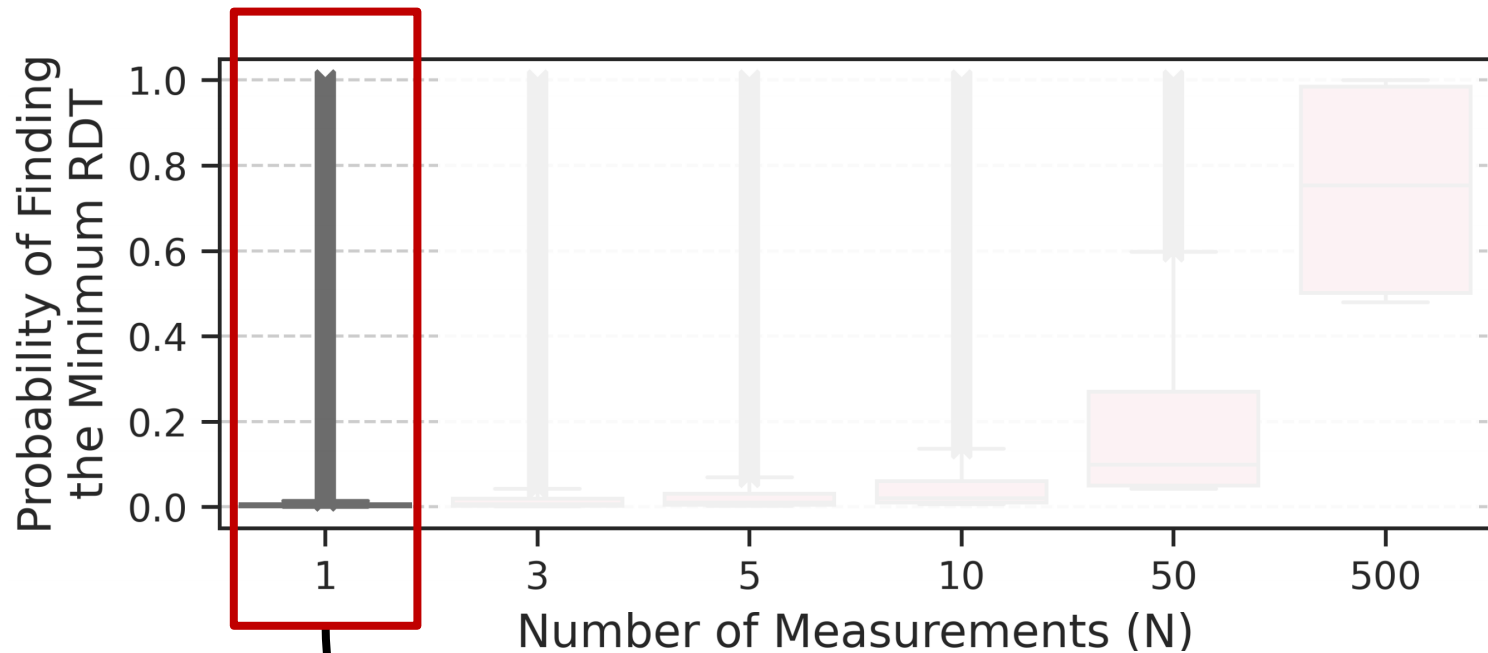
Data patterns, temperature, and aggressor row on time affect VRD

Probability of Identifying the Minimum RDT

- How likely is it that $N < 1000$ measurements yield the minimum RDT value across 1000 measurements?
- $N = 1, 3, 5, 10, 50$, and 500
- Monte Carlo simulations for 10K iterations



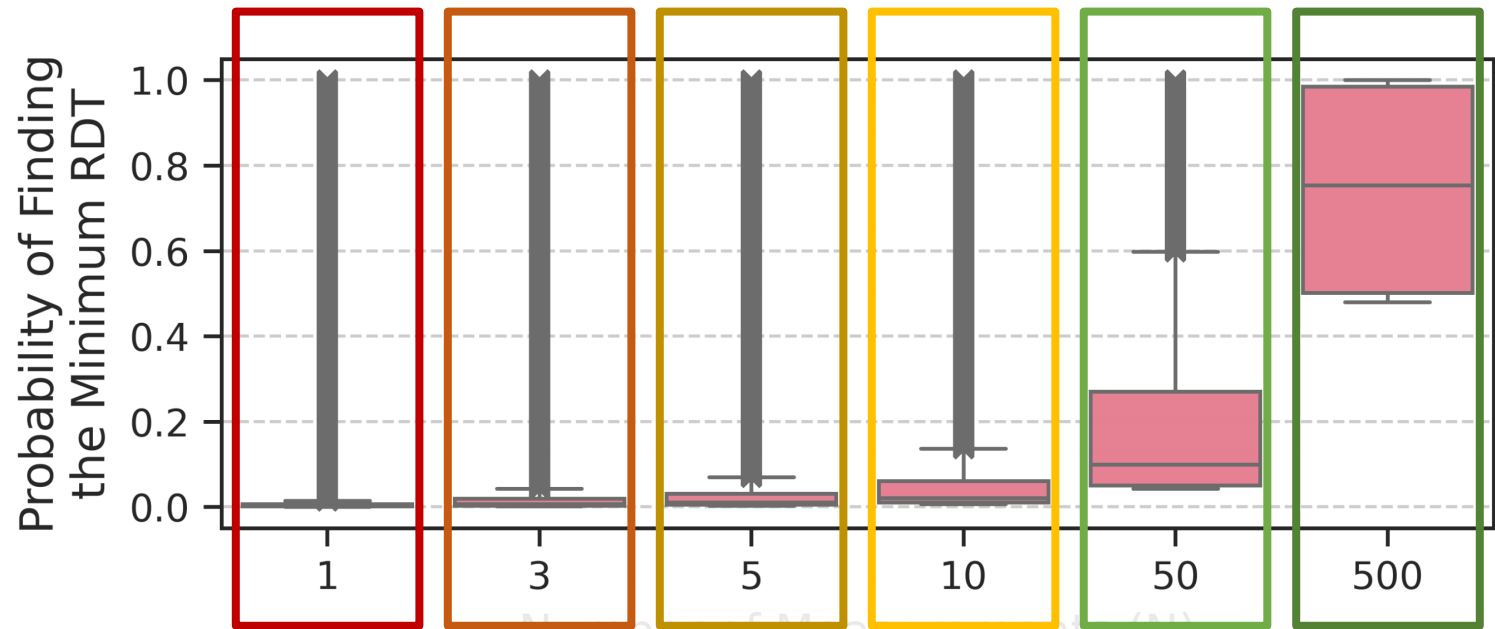
Probability of Identifying the Minimum RDT



only 0.2% for the median row

Very **unlikely** to find the minimum RDT of a DRAM row with **N = 1** measurement

Probability of Identifying the Minimum RDT



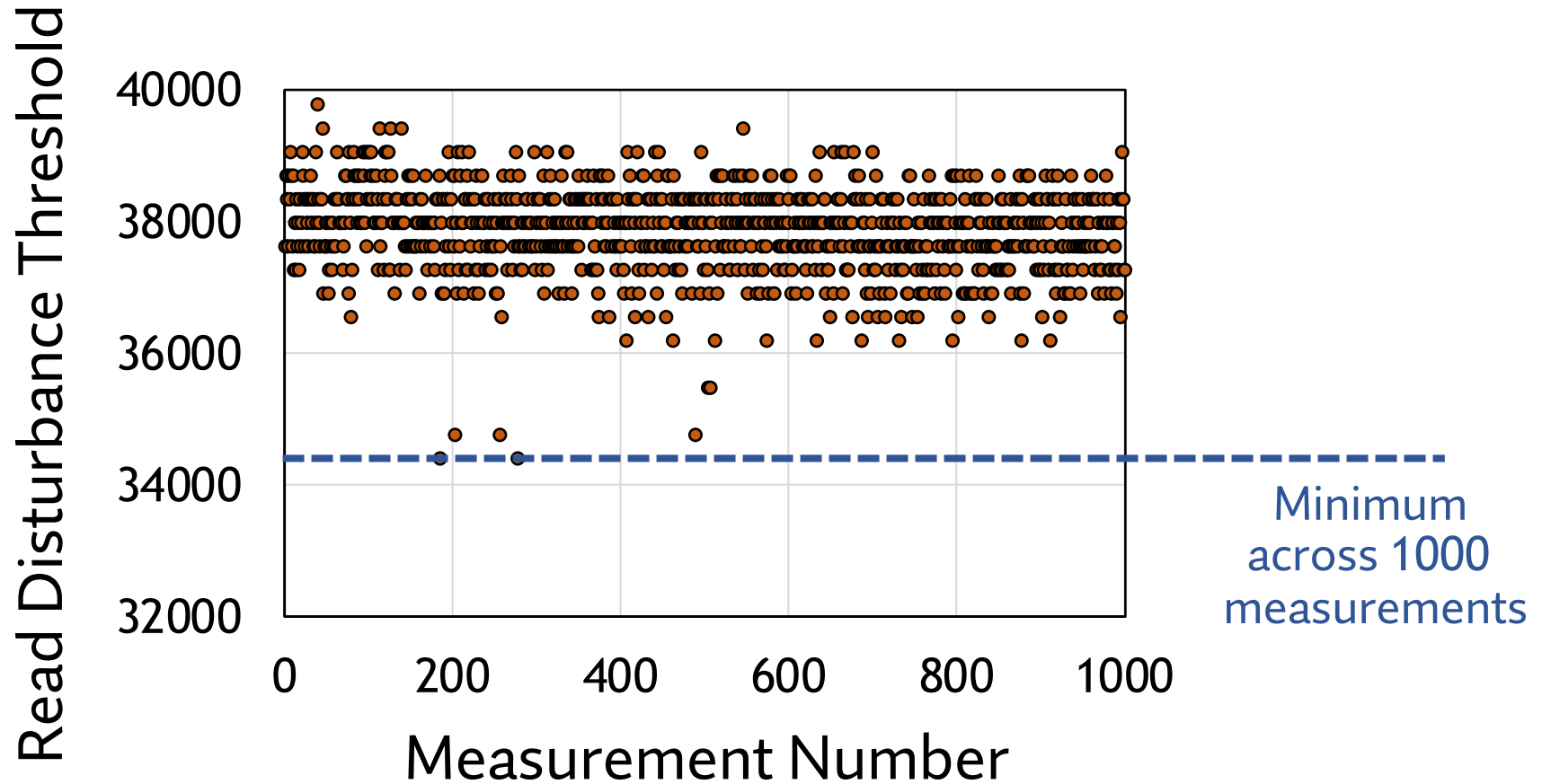
0.2% 0.7% 1.1% 2.1% 10.0% 75.3%

Probability values for the median row

Probability of finding the minimum read disturbance threshold increases with N (i.e., with more and more testing)

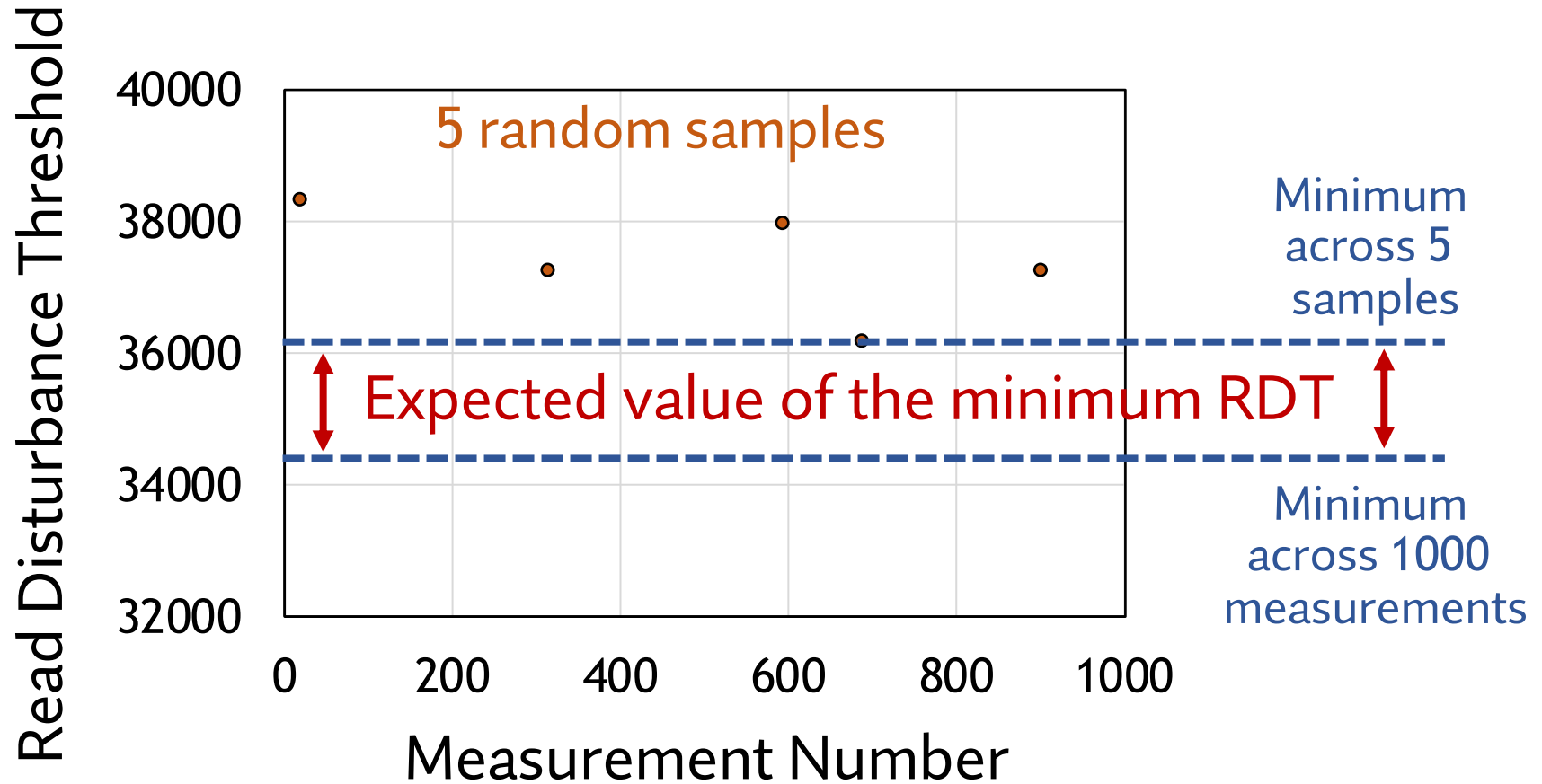
Expected Value of the Minimum RDT

- With only $N < 1000$ RDT measurements
how far are we from the minimum RDT across 1000 measurements

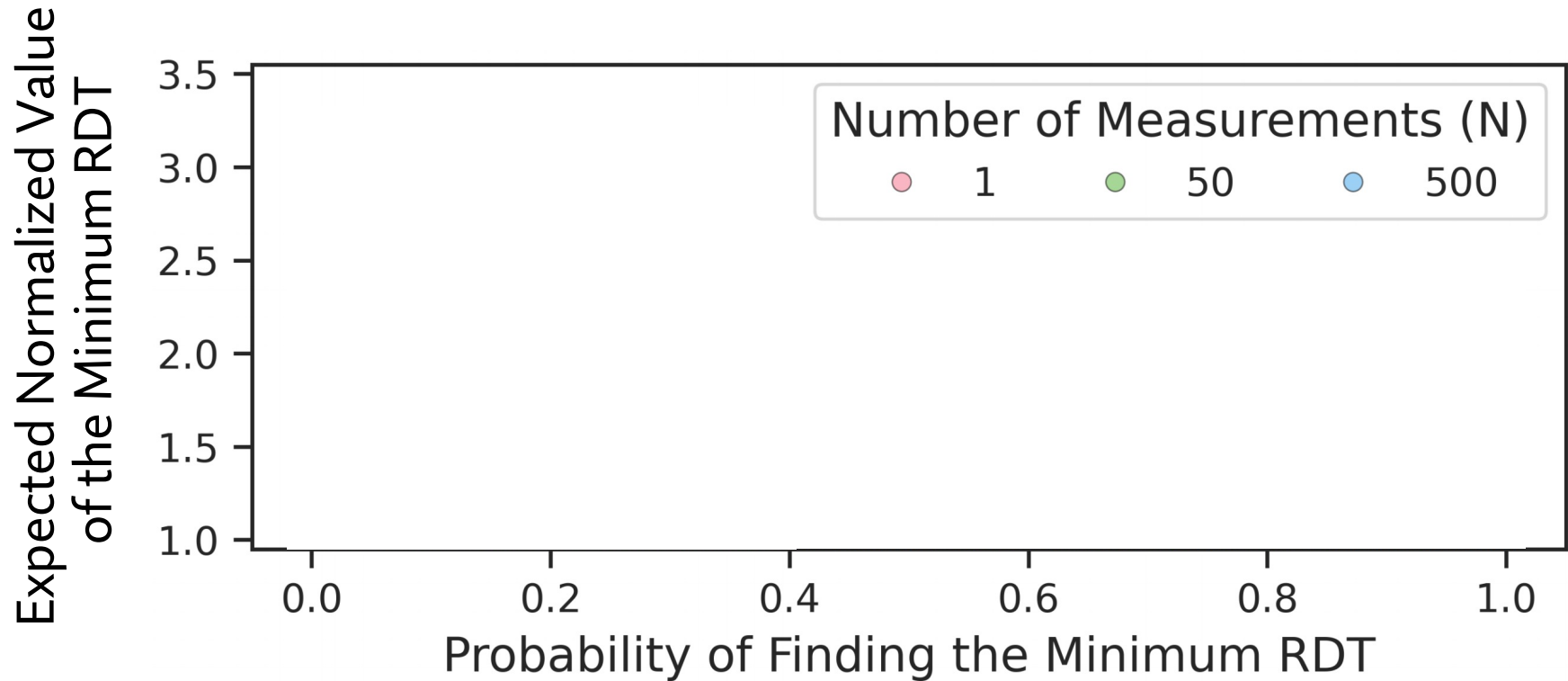


Expected Value of the Minimum RDT

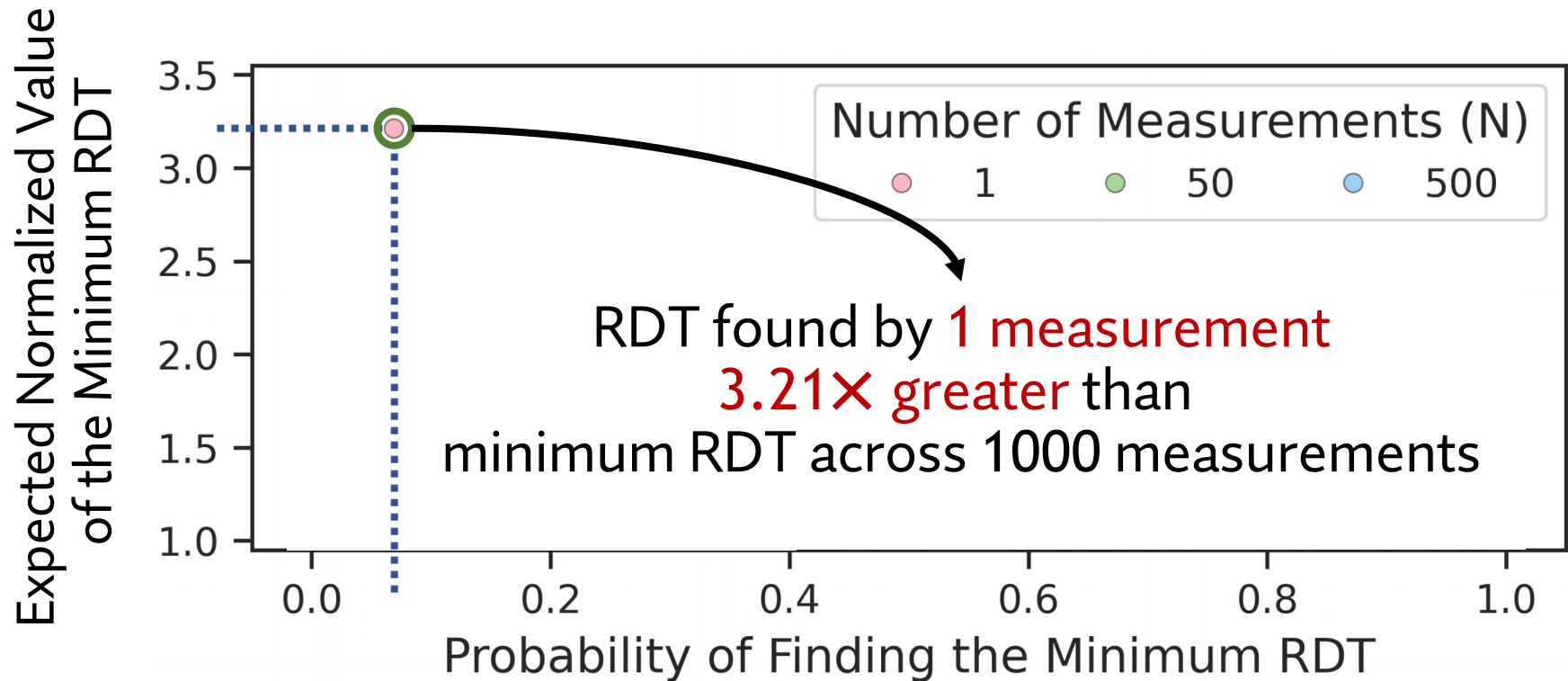
- With only $N < 1000$ RDT measurements
how far are we from the minimum RDT across 1000 measurements



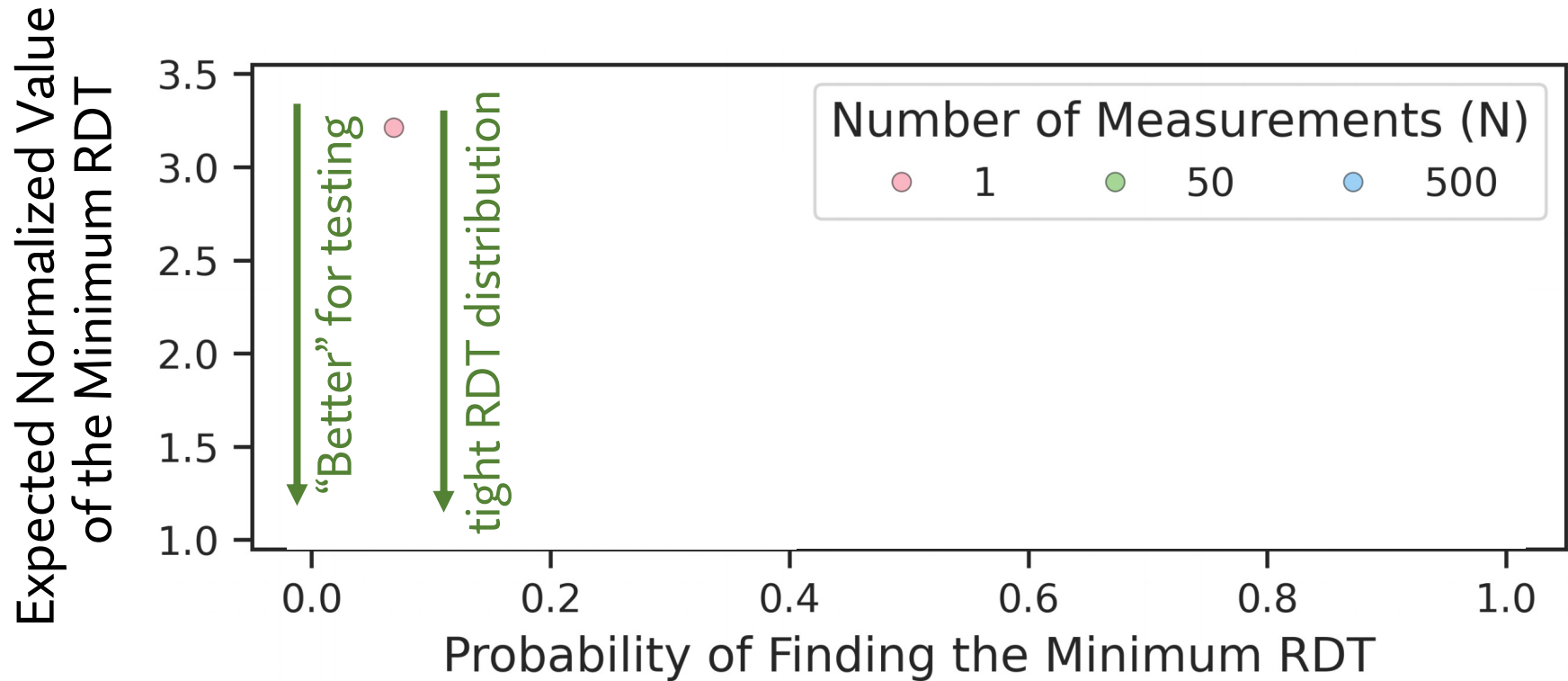
Expected Value of the Minimum RDT



Expected Value of the Minimum RDT



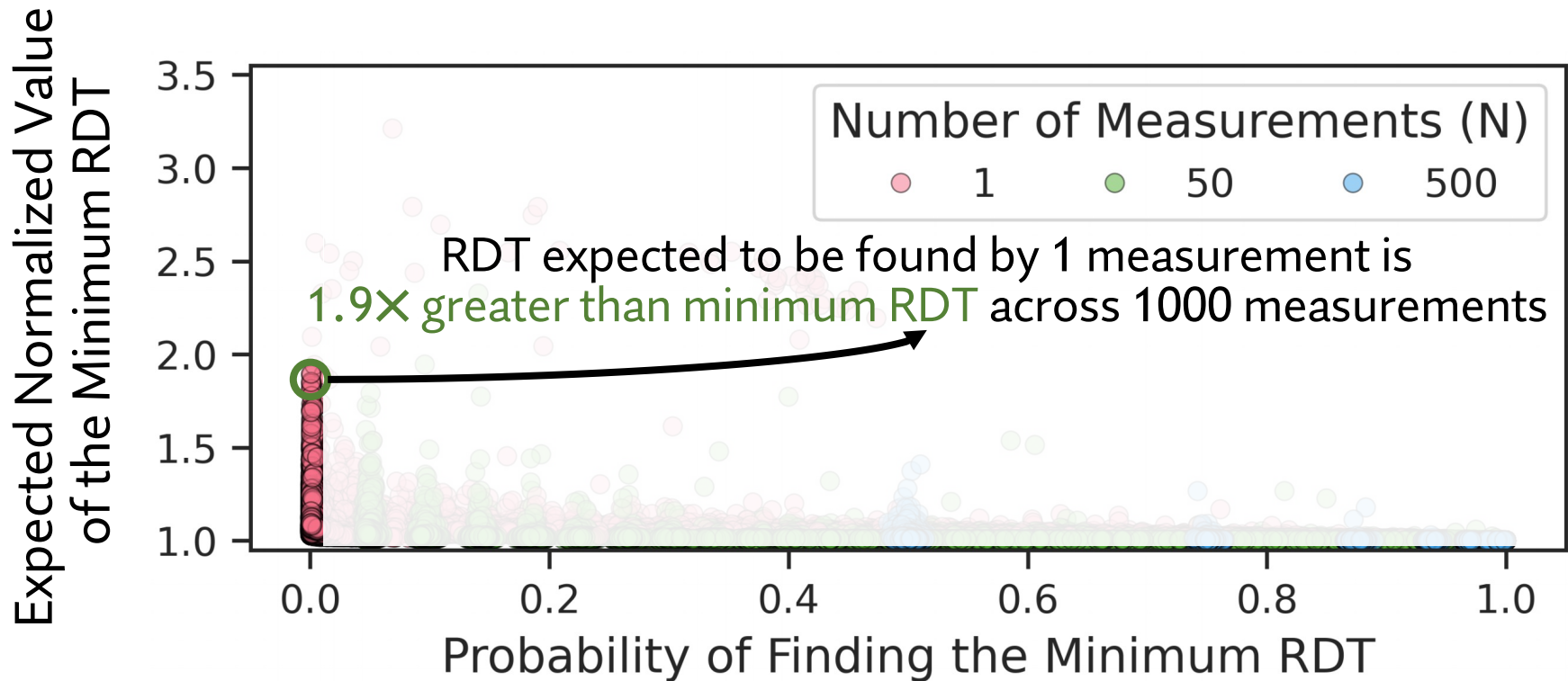
Expected Value of the Minimum RDT



Plot interpretation

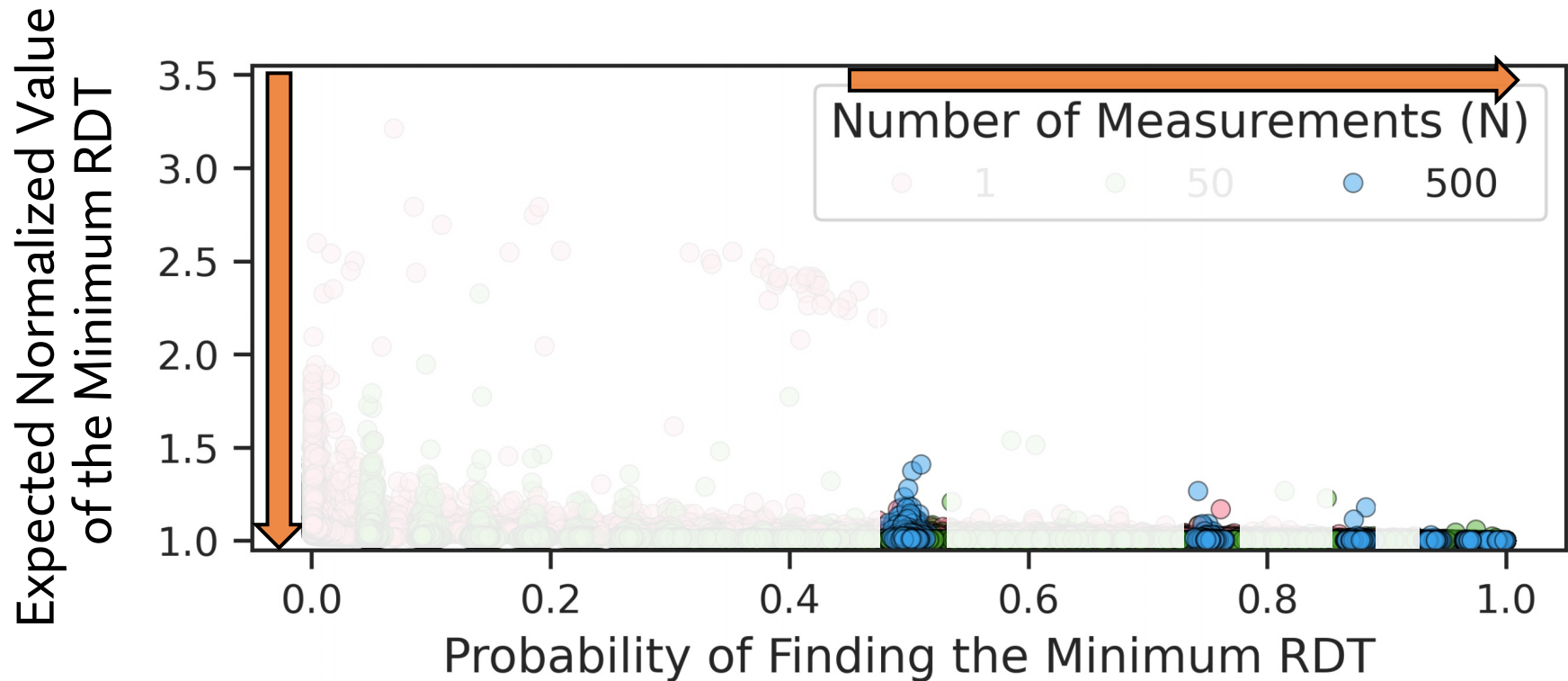
“Better” for testing == as **tight** an RDT distribution as possible

Expected Value of the Minimum RDT



The minimum RDT is **significantly smaller** than the one expected to be found with **N = 1** measurement

Expected Value of the Minimum RDT



With **increasing N** (number of measurements)
we expect to identify an RDT value
closer to the minimum across 1000 measurements

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows **exhibit VRD**

Takeaway 2

Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

Data patterns, temperature, and aggressor row on time **affect VRD**

Talk Outline

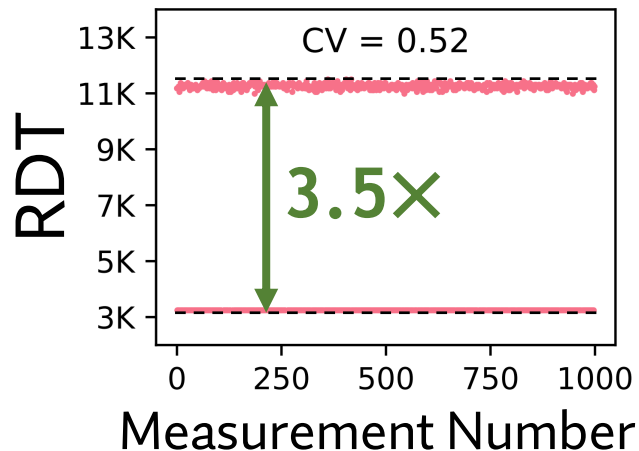
- I. Motivation
- II. Experimental Characterization Methodology
- III. Foundational Results
- IV. In-Depth Analysis of VRD
- V. Implications for System Security and Robustness**
- VI. Conclusion

Important Caveat

- VRD solution analysis based on **1K** or **10K** read disturbance threshold **measurements** per row
- **More measurements could yield worse results**
 - Read disturbance threshold distribution **tail could expand**
- What results would **millions** or **billions** of RDT **measurements** yield?

Challenges of Accurately Identifying RDT

Variation in read disturbance threshold across 1000 measurements can reach 3.5X and may not be bounded

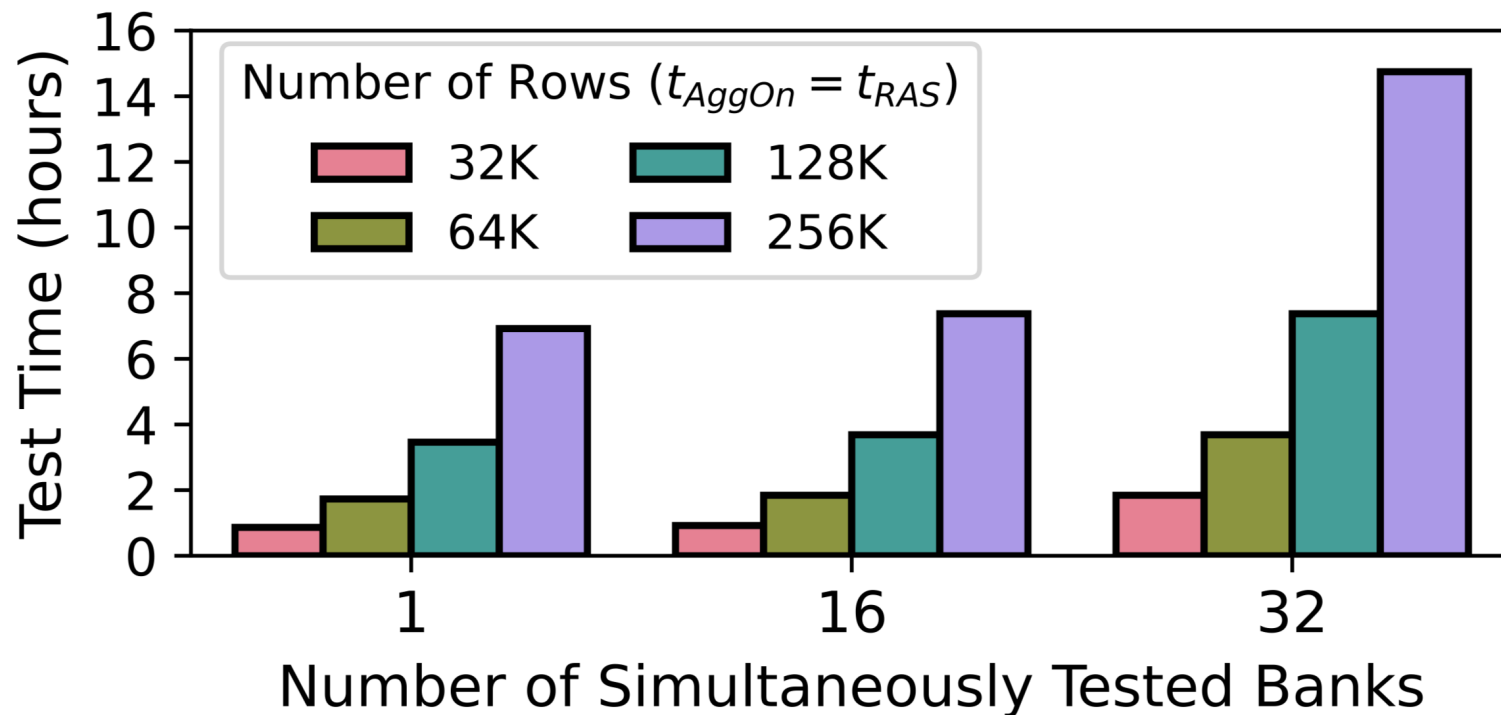


VRD is affected by data pattern, temperature, aggressor row on time

Comprehensive RDT profiling is time-intensive

Measuring RDT of *each row only once* with 8000 hammers using *four data patterns*, at *three temperature levels* takes 39 minutes in a bank of 256K rows

RDT Profiling is Time-Intensive



Comprehensive RDT testing can take tens of hours
(**only** 1000 measurements, one data pattern,
one temperature level, one aggressor row on time)

RDT Profiling is Time-Intensive

<https://arxiv.org/pdf/2502.13075>

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun[†] F. Nisa Bostancı[†] İsmail Emir Yüksel[†] Oğuzhan Canpolat[†] Haocong Luo[†]
Geraldo F. Oliveira[†] A. Giray Yağlıkçı[†] Minesh Patel[‡] Onur Mutlu[‡]

ETH Zurich[†] Rutgers University[‡]

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is $3.5\times$ smaller than the maximum RDT observed for that row. Second, the probability of accu-

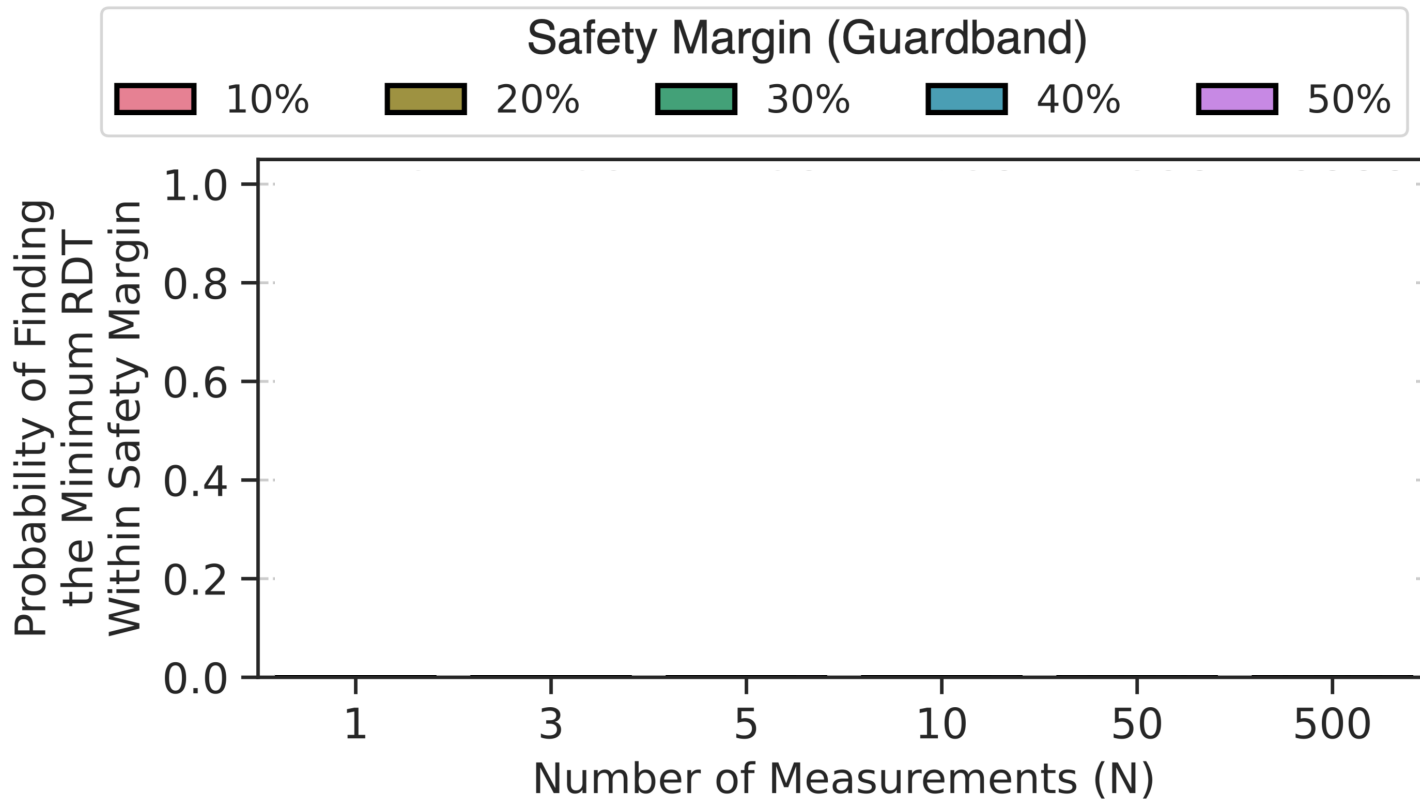
row) many times (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count* (the number of aggressor row activations) needed to induce the first read disturbance bitflip in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

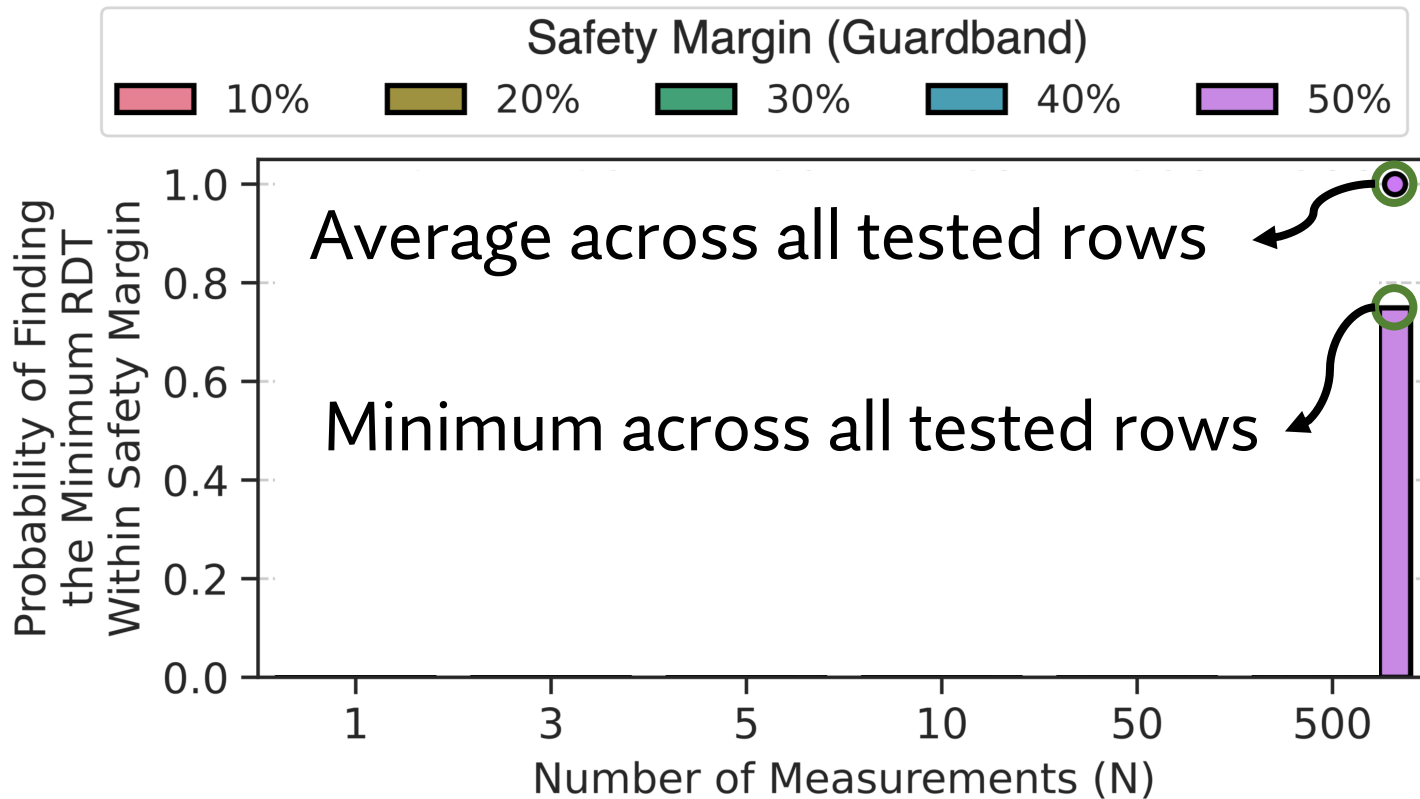
Making Do With Few RDT Measurements

- A system designer might measure RDT **a few times** and apply a **safety margin (guardband)** to the minimum **observed** value

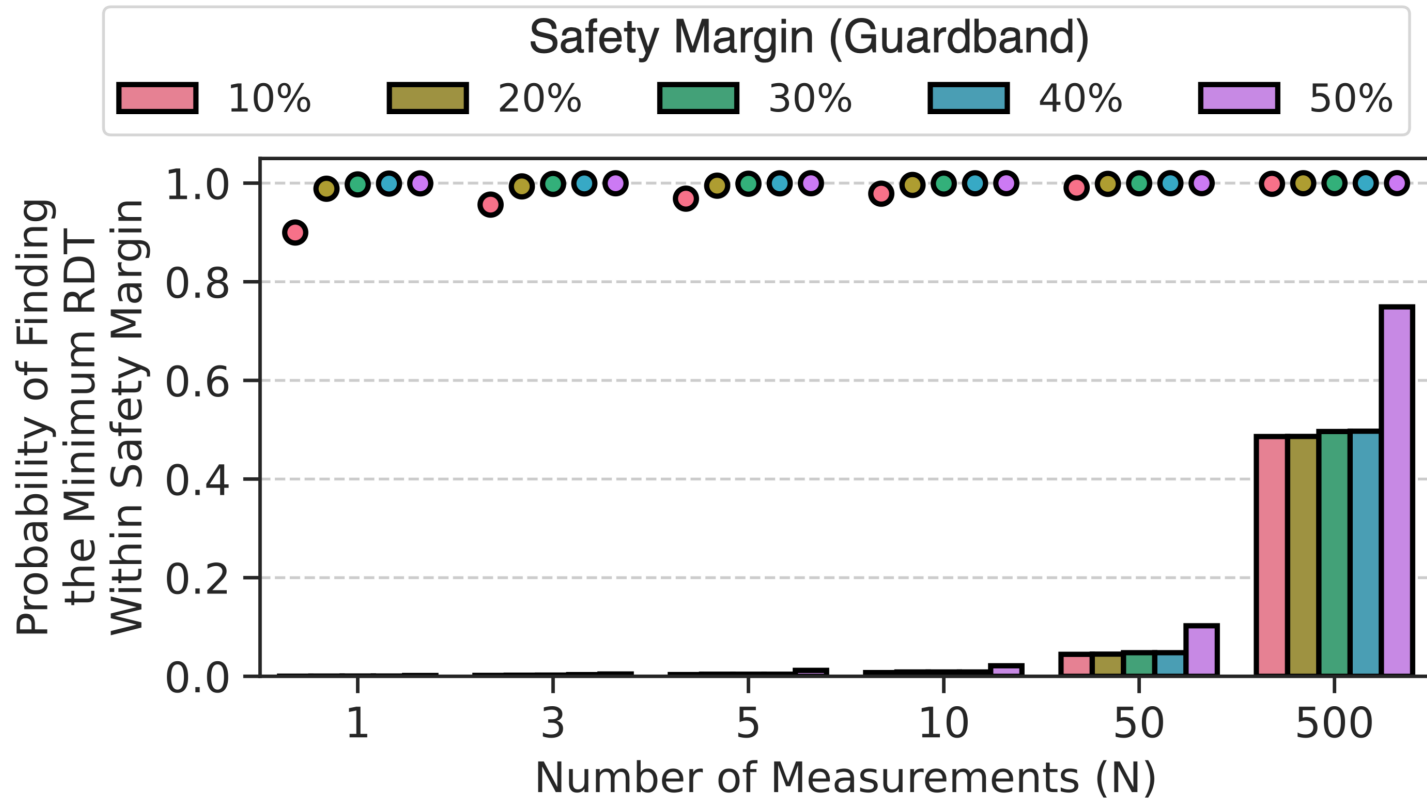


Making Do With Few RDT Measurements

- A system designer might measure RDT **a few times** and apply a **safety margin (guardband)** to the minimum **observed** value



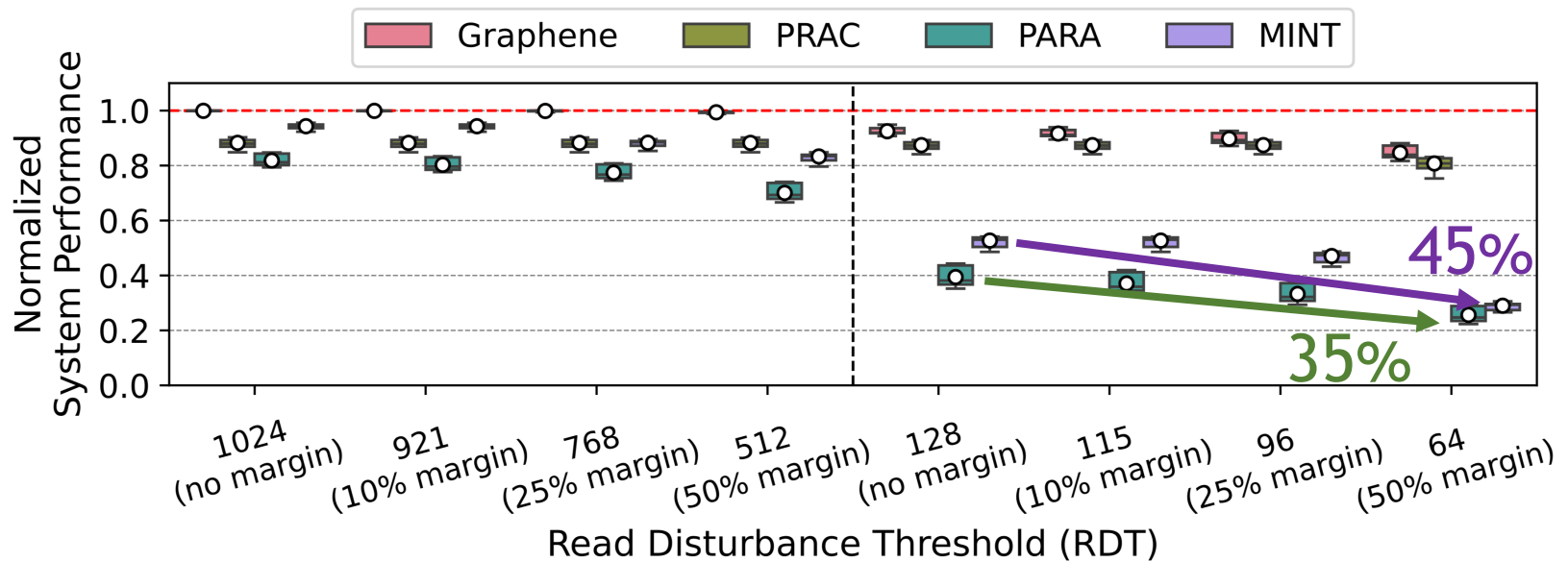
Making Do With Few RDT Measurements



A large guardband **does not guarantee** that the **minimum** RDT is **always identified**

Using guardbands alone is **likely not effective**

RDT Guardband Increases Performance Overheads



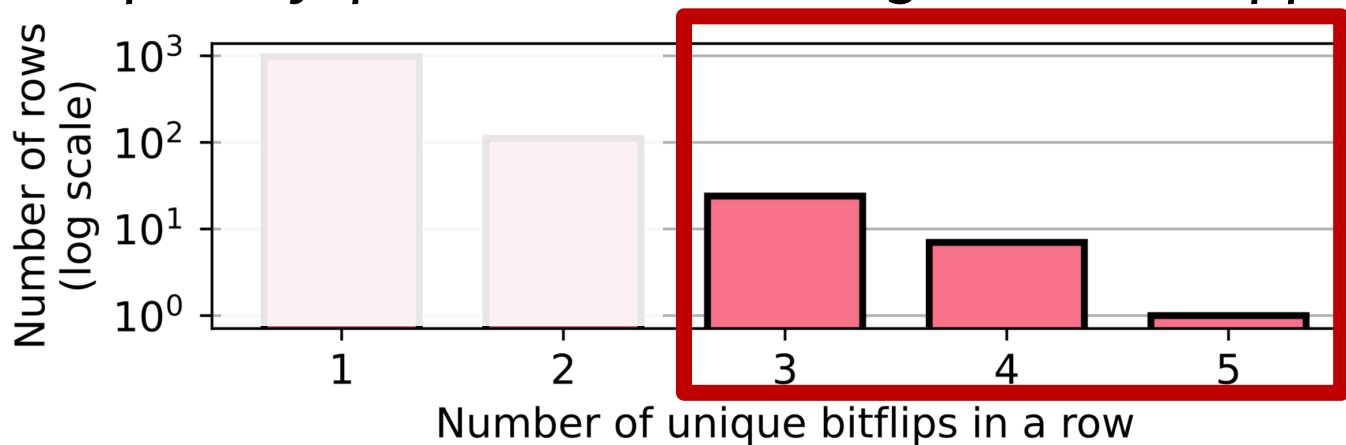
50% RDT **safety margin** can induce
45% additional overhead (over no margin)

Relying **solely** on guardbands **not** recommended

Combining ECC and Guardbands (I)

- Single-error correcting double-error detecting (SECEDED) or Chipkill ECC combined with guardbands could mitigate VRD-induced bitflips

Unique bitflips when 10% RDT guardband applied



10% guardband combined w/ ECC is likely unsafe

More in the Paper

- Hypothetical explanation for VRD
- Effect of True- and Anti-Cell Layout
 - Presence of true- and anti-cells in the victim row does not significantly affect the RDT distribution
- Read disturbance mitigation evaluation methodology
- Probability of errors at the worst observed bitflip rate for 10% RDT guardband
 - SEC, SECDED, and Chipkill-like (SSC)
- Read disturbance testing time and energy consumption
- Detailed information on tested modules and chips

More in the Paper

<https://arxiv.org/pdf/2502.13075>

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun[†] F. Nisa Bostancı[†] İsmail Emir Yüksel[†] Oğuzhan Canpolat[†] Haocong Luo[†]
Geraldo F. Oliveira[†] A. Giray Yağlıkçı[†] Minesh Patel[‡] Onur Mutlu[‡]
ETH Zurich[†] Rutgers University[‡]

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is $3.5\times$ smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) many times (e.g., tens of thousands of times) induces RowHammer bitflips in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces RowPress bitflips, without requiring many repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count* (the number of aggressor row activations) needed to induce the first read disturbance bitflip in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

Talk Outline

- I. Motivation
- II. Experimental Characterization Methodology
- III. Foundational Results
- IV. In-Depth Analysis of VRD
- V. Implications for System Security and Robustness
- VI. Conclusion**

Variable Read Disturbance

Backup Slides

Is this Experiment Noise?

- Short answer: **no**
- Hypothetical explanation for VRD:
Randomly changing charge trap state in the active region
- Long answer:
We cannot identify any independent variables within our control that allow reliably predicting the minimum RDT despite extensive testing
- Device-level studies should confirm our hypotheses

Hypothetical Explanation for VRD

- No device-level study shows temporal variations in read disturbance vulnerability
- Electron migration and injection into victim cell is a major read disturbance failure mechanism
- This mechanism is assisted by charge traps in the shared active region of the victim and aggr. cell
- Temporal variation attributed to randomly changing occupied/unoccupied states of charge traps

RDT Distribution Across Chips

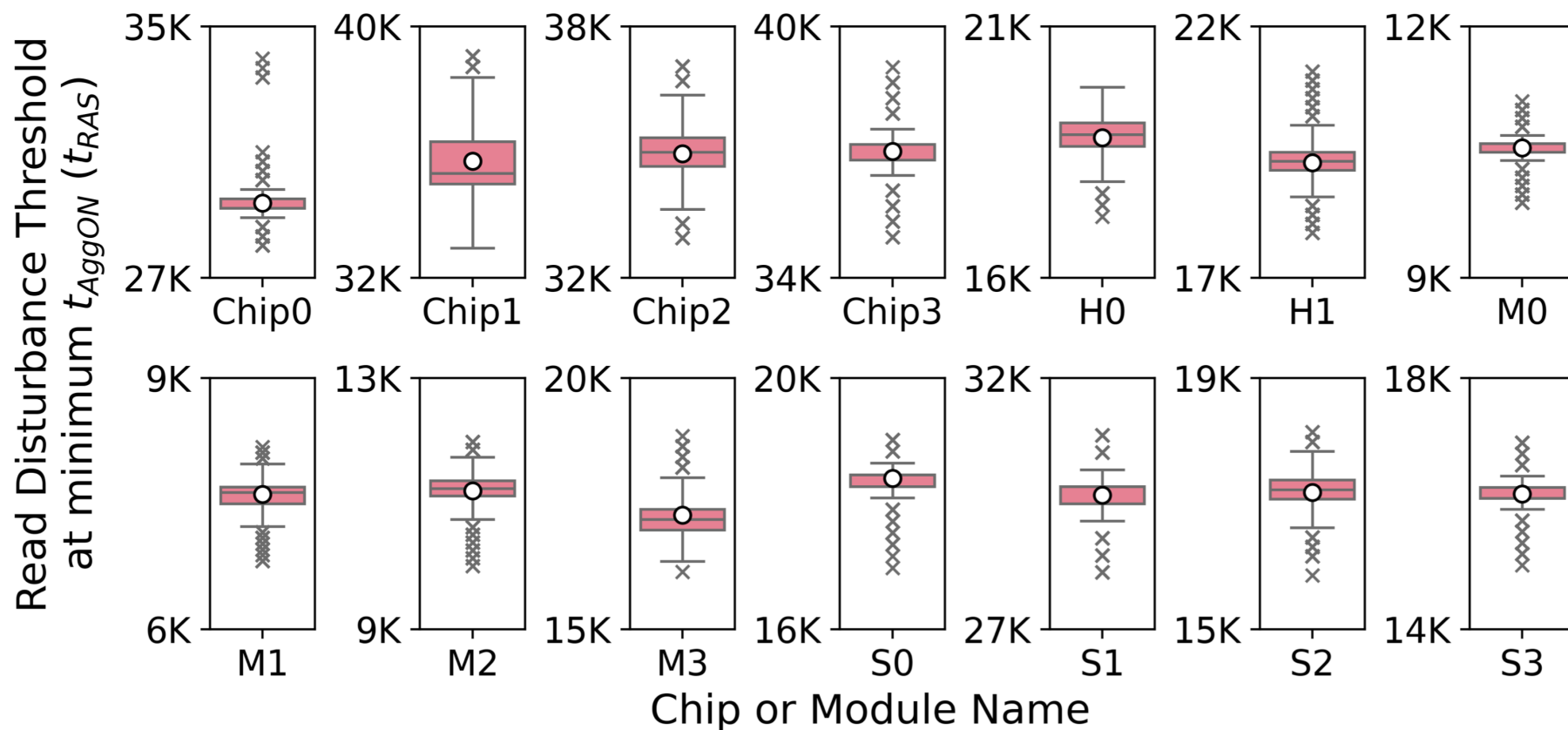


Figure 3: RDT distribution of a single victim row in each tested module and chip

of Consecutive Measurements That Yield the Same RDT Value

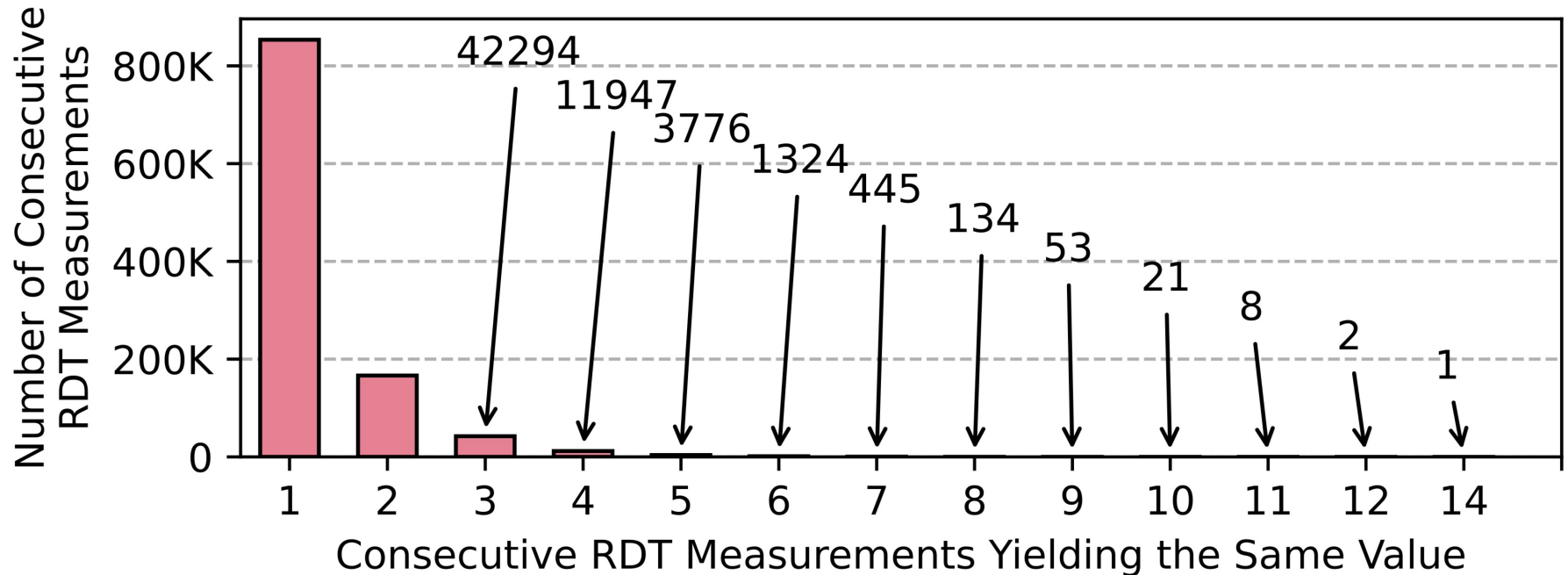
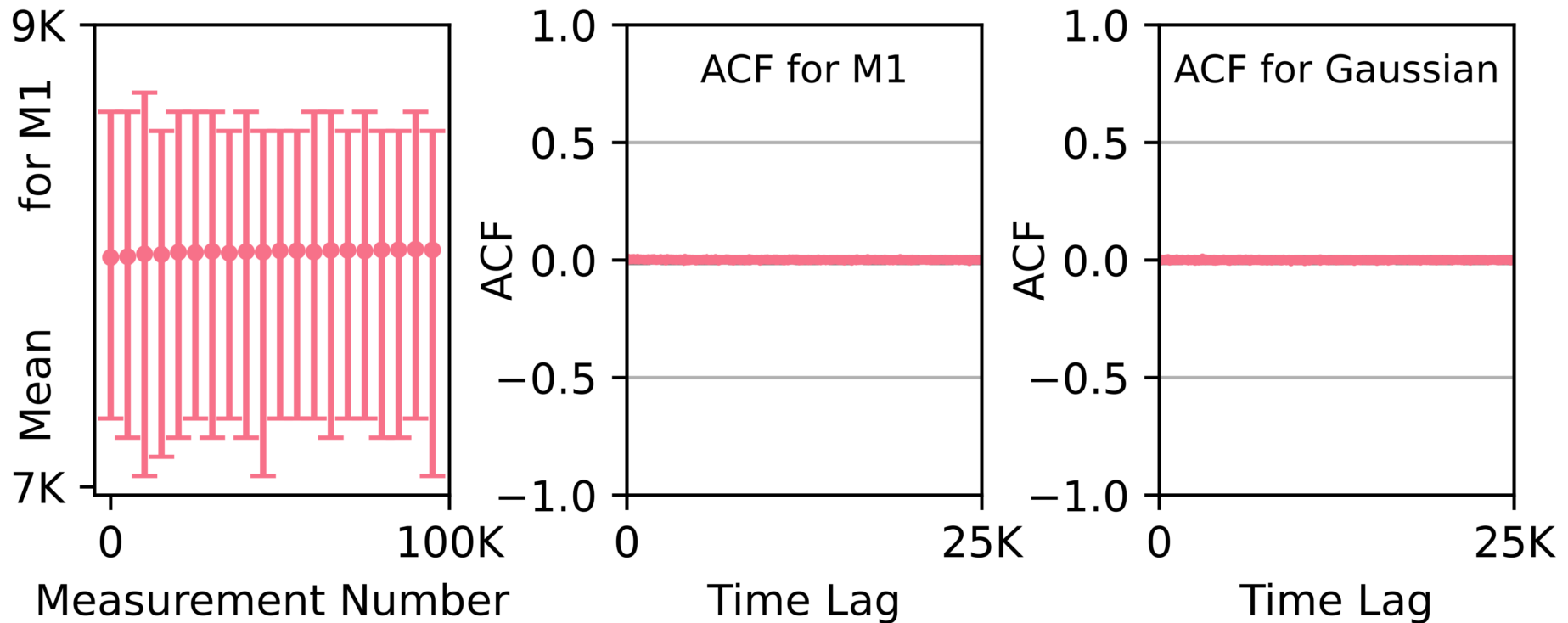
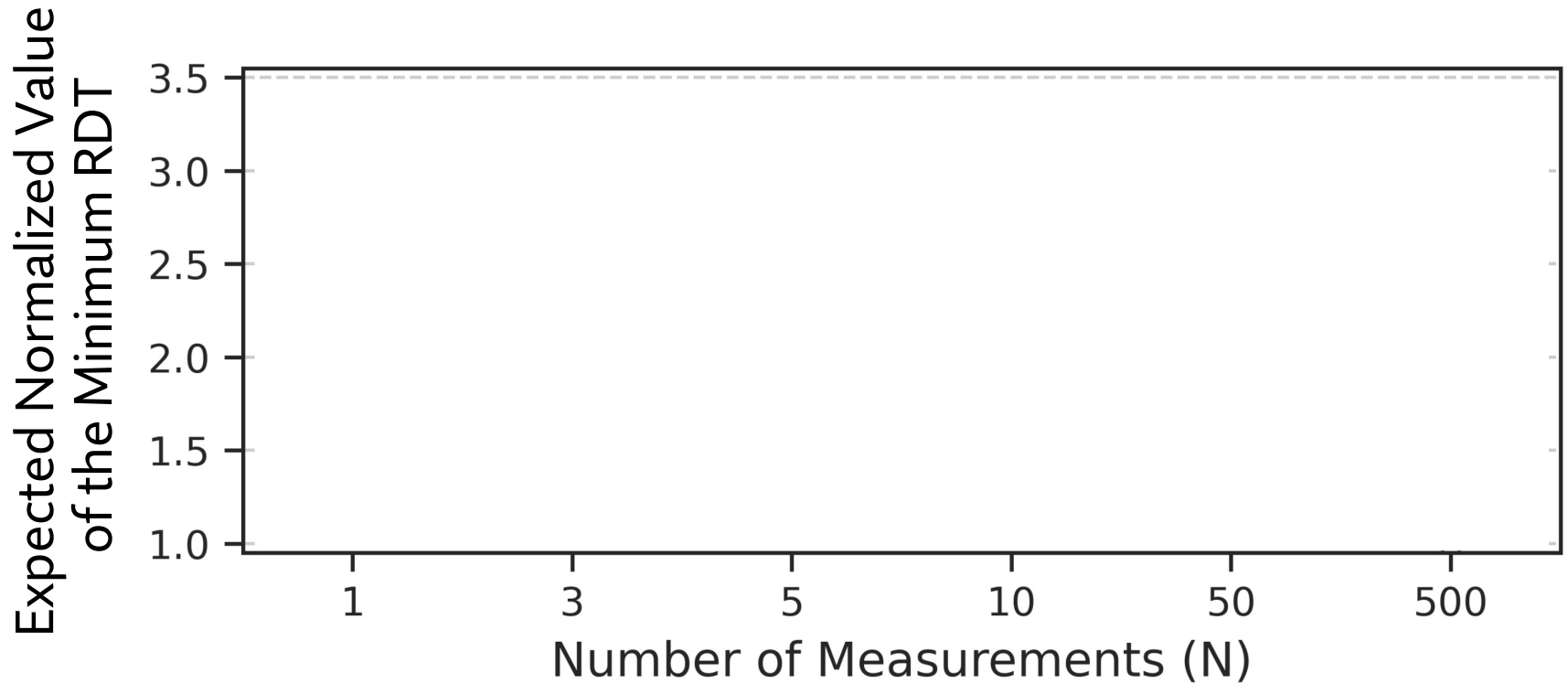


Figure 5: Histogram of the number of measurements across which a row's RDT exhibits the same value

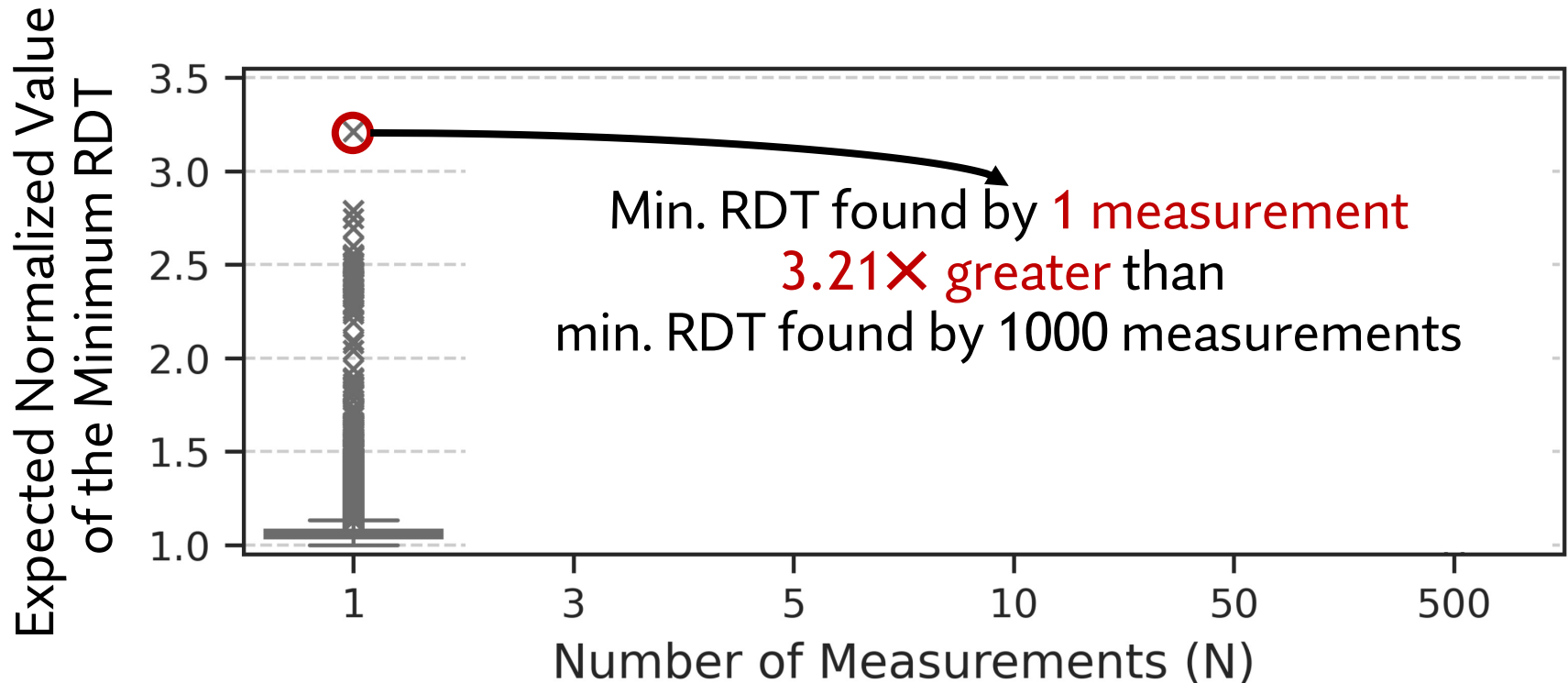
Autocorrelation Function Tests



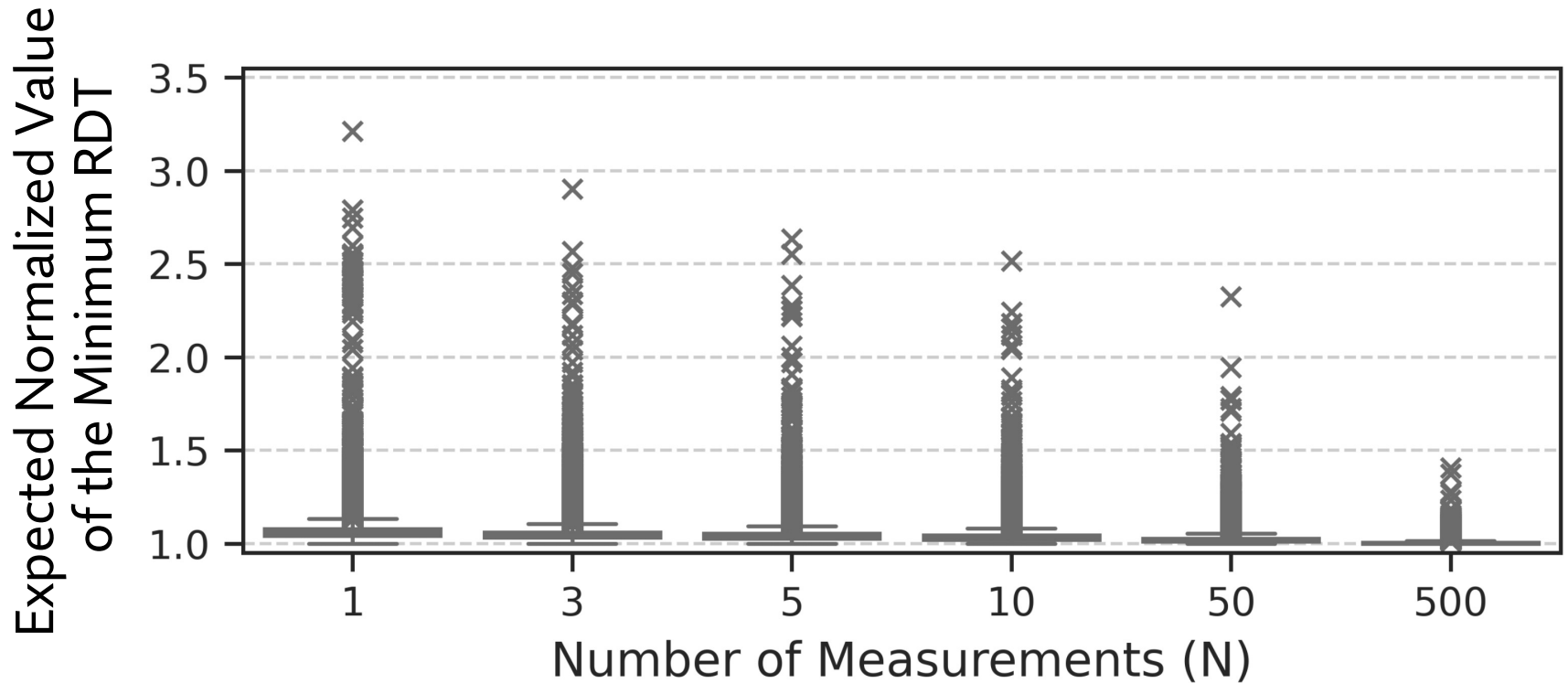
Expected Value of the Minimum RDT



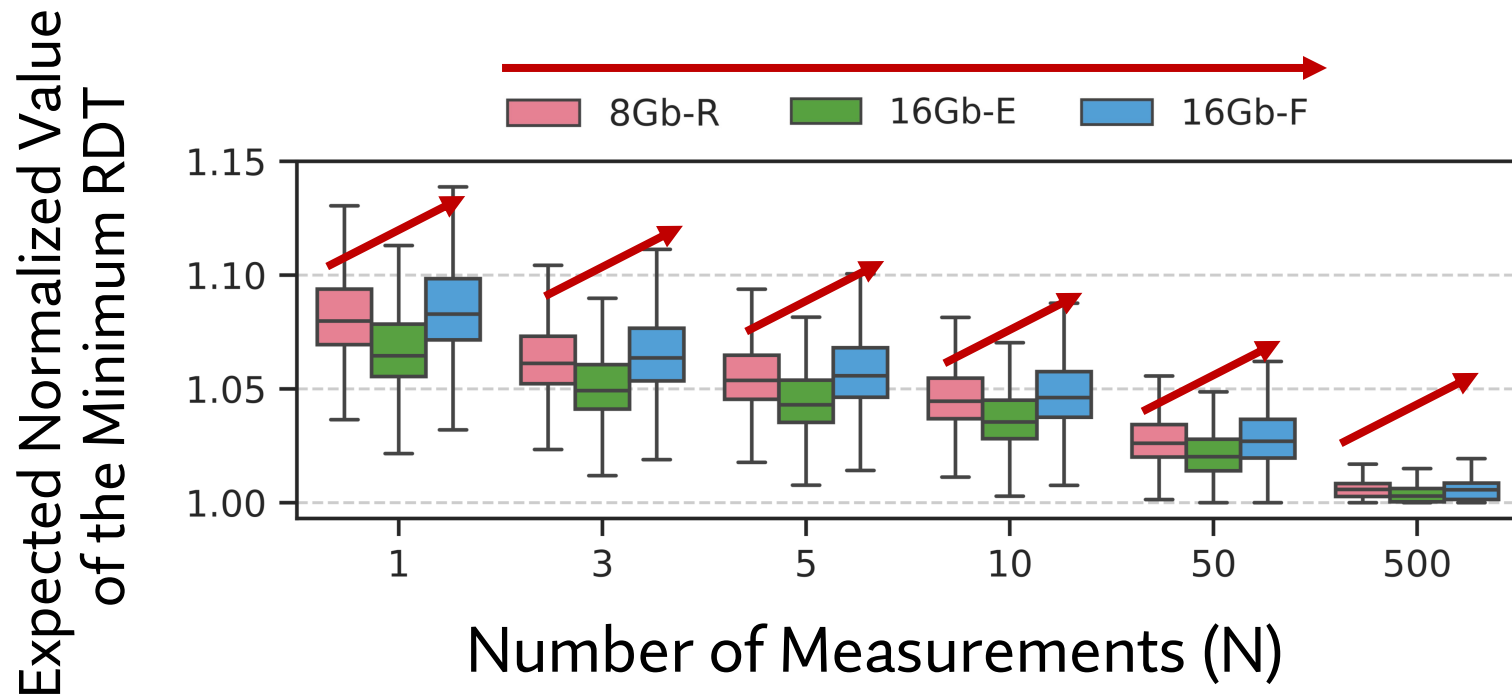
Expected Value of the Minimum RDT



Expected Value of the Minimum RDT

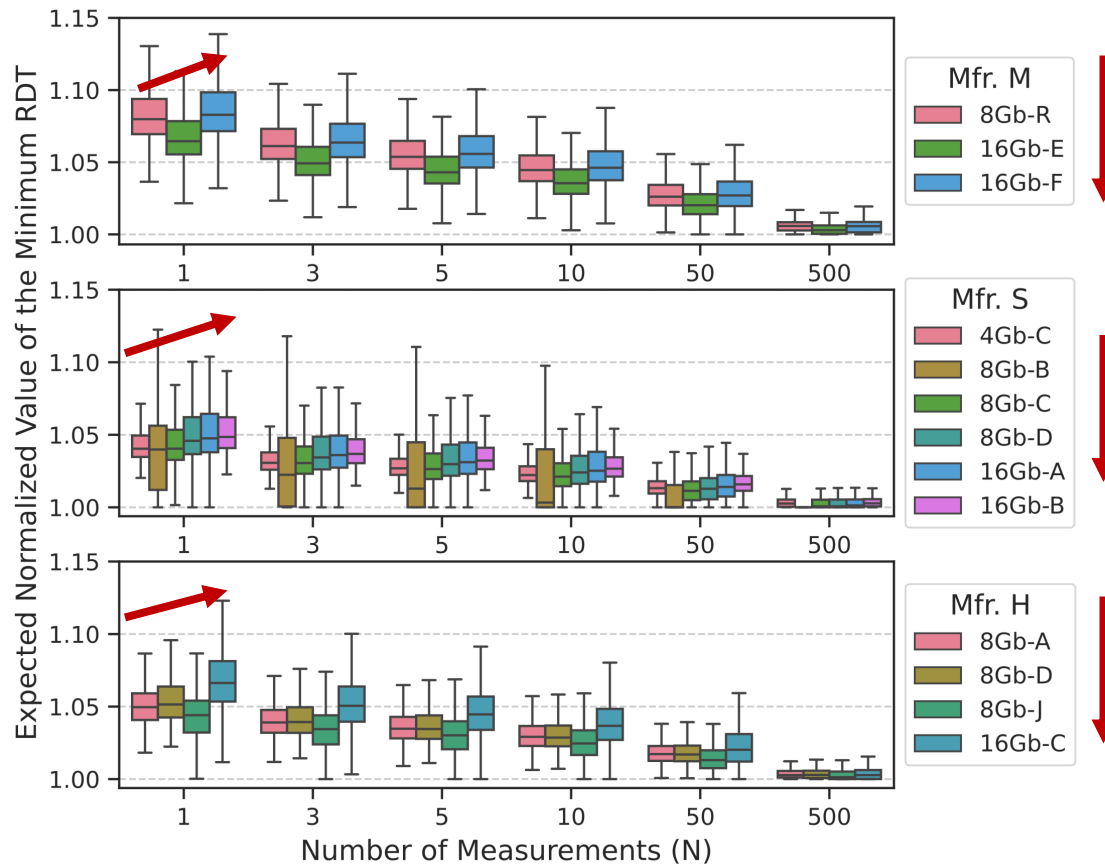


Effect of Die Density and Die Revision (I)



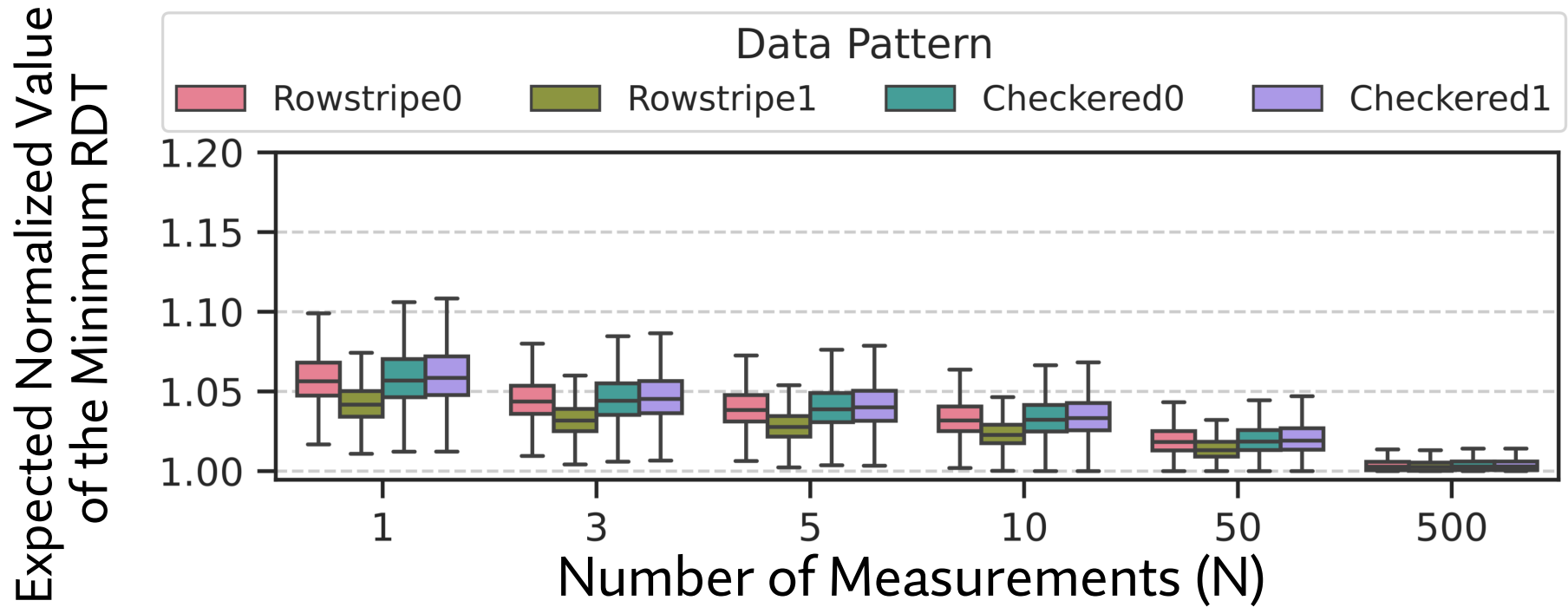
RDT distribution **worsens** with increasing die **density** and with **advanced DRAM technology**

Effect of Die Density and Die Revision (II)

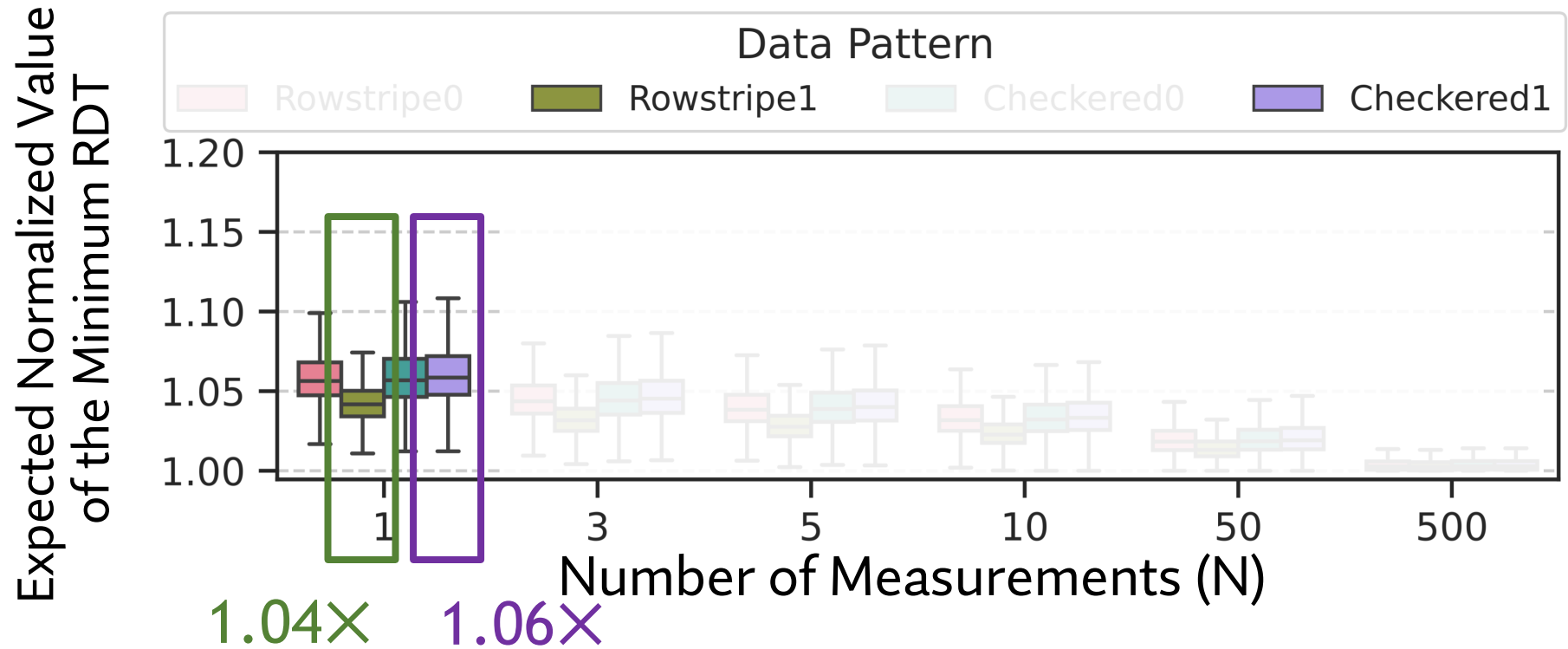


The effect of die density and die revision is **consistent** across all tested modules

Effect of Data Pattern (I)



Effect of Data Pattern (I)

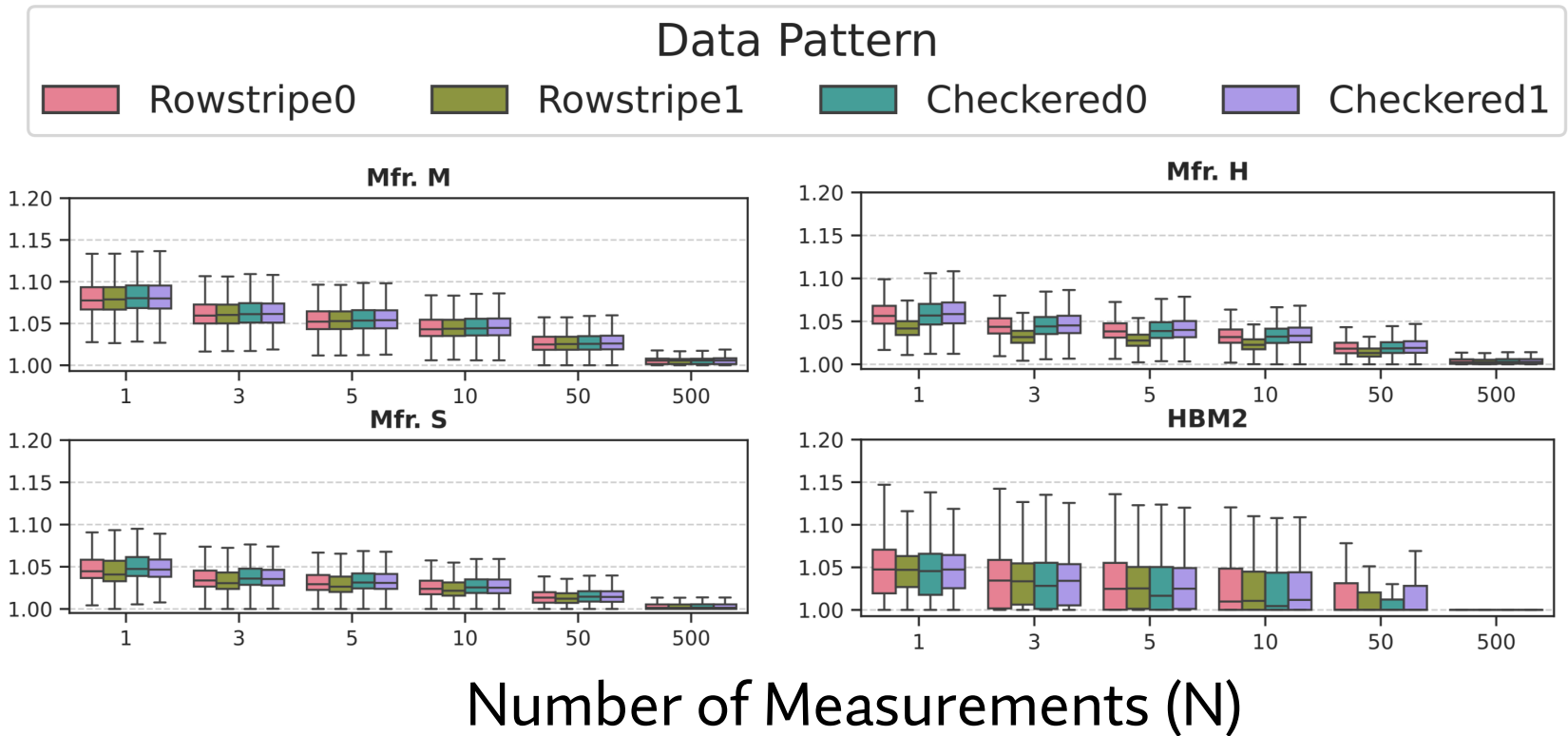


expected normalized value for the median row

RDT distribution **changes with data pattern**

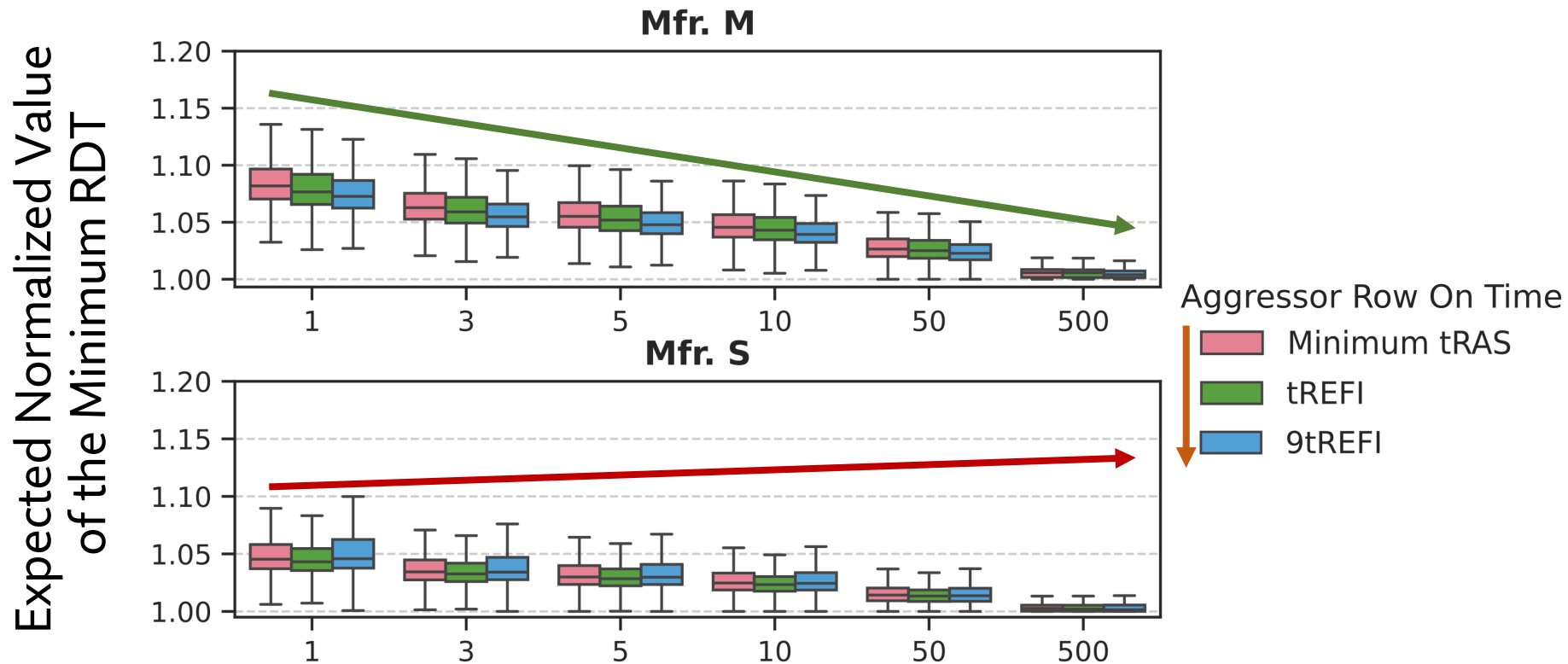
Effect of Data Pattern (II)

Expected Normalized Value
of the Minimum RDT



No single data pattern causes the **worst** RDT distribution across all tested DRAM chips

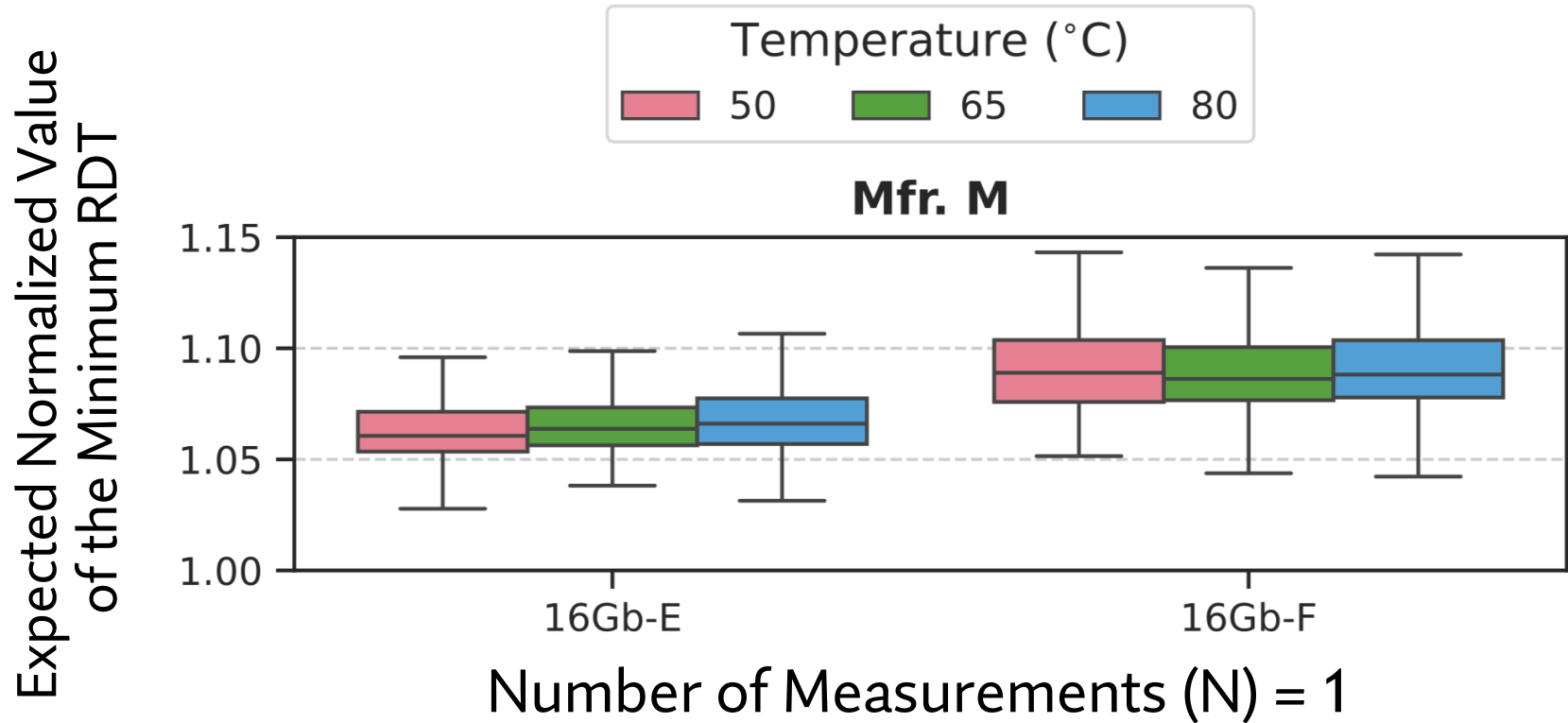
Effect of Aggressor Row On Time



RDT distribution **changes** with aggressor row on time

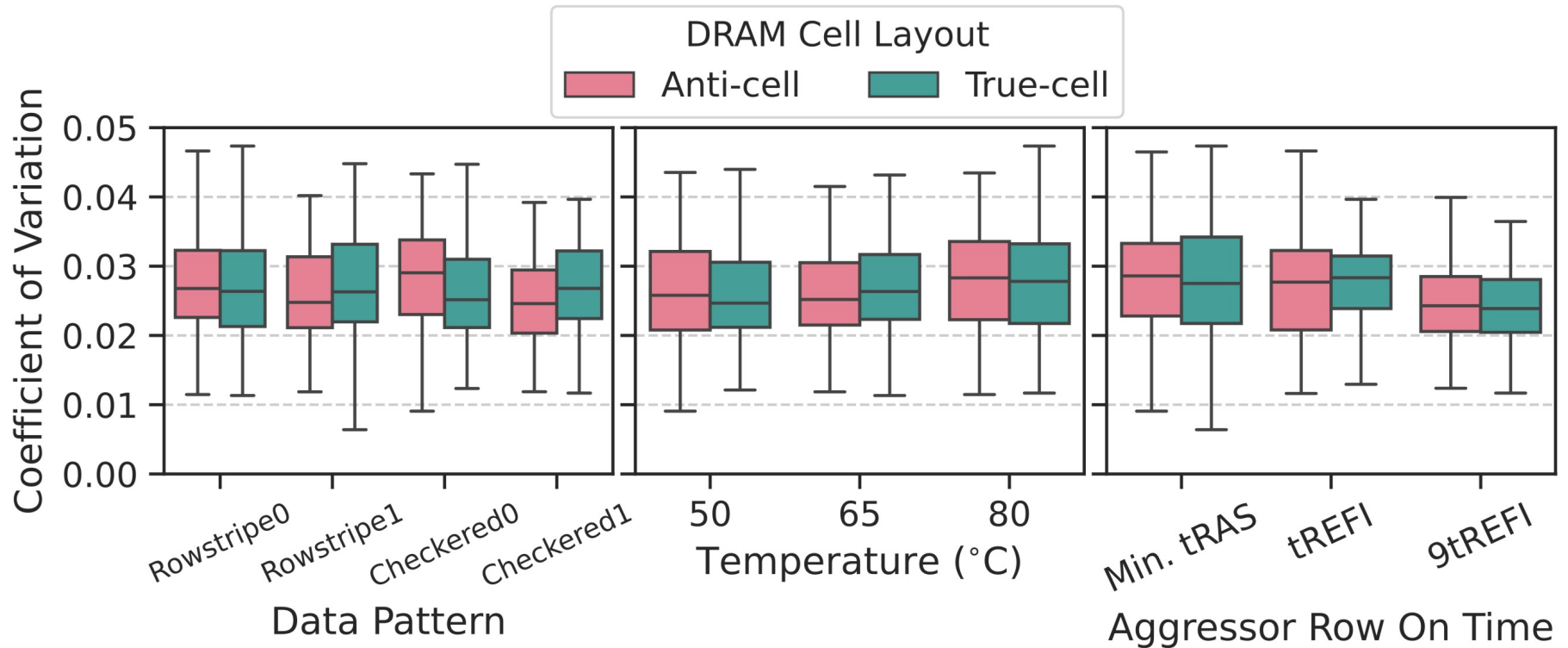
RDT distribution can become **better** or **worse**
with **increasing row on time**

Effect of Temperature



RDT distribution tends to change with temperature

Effect of True- and Anti-Cell Layout



The presence of true- and anti-cells in the victim row does not significantly affect the RDT distribution

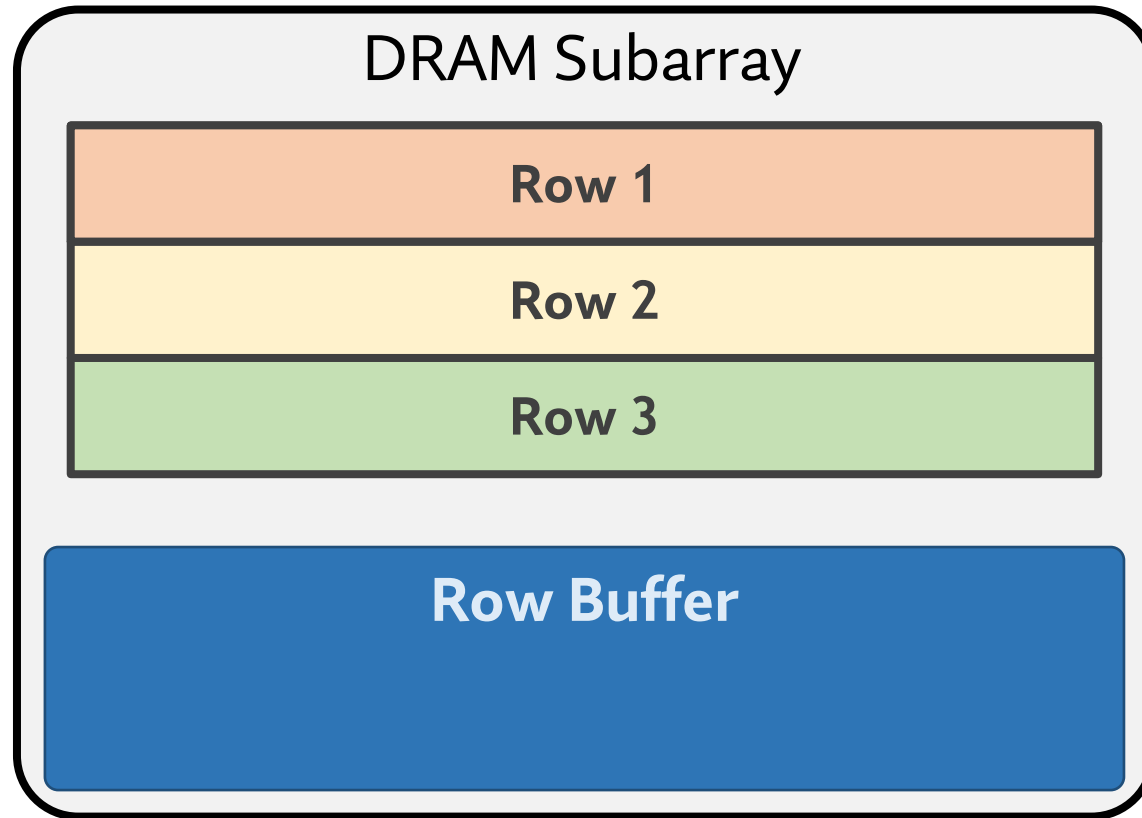
Error Probability Analysis

Table 3: Probability of uncorrectable, undetectable, and detectable uncorrectable errors at the worst error rate we observed empirically so far ($7.6e - 5$) using an RDT safety margin of 10% for SEC, SECDED, and Chipkill-like SSC codes. N/A indicates the result category does not exist for the shown ECC type.

Type of error	SEC	SECDED	Chipkill-like (SSC)
Uncorrectable	1.48e-05	1.48e-05	5.66e-05
Undetectable	1.48e-05	2.64e-08	5.66e-05
Detectable uncorrectable	N/A	1.48e-05	N/A

DRAM Operation: Activate and Precharge

Access data in Row 1

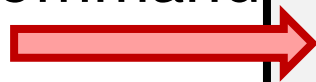
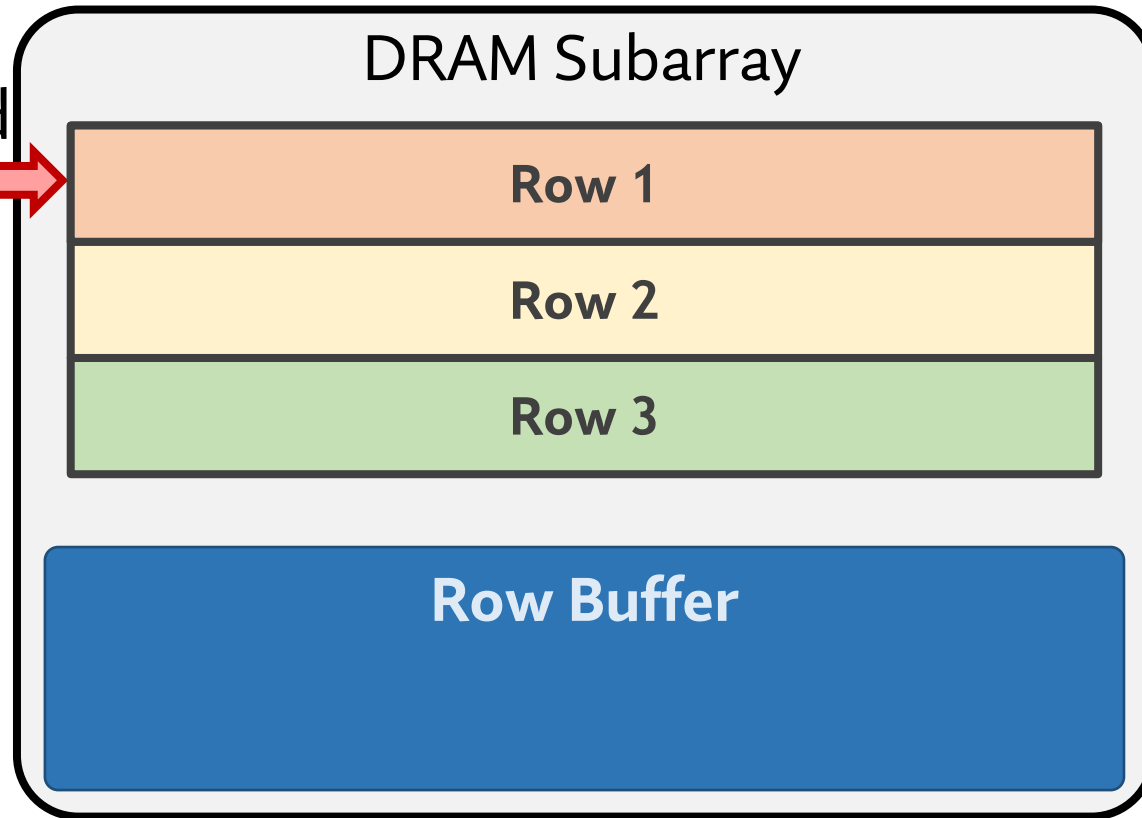


Row 1 is closed

DRAM Operation: Activate and Precharge

Access data in Row 1

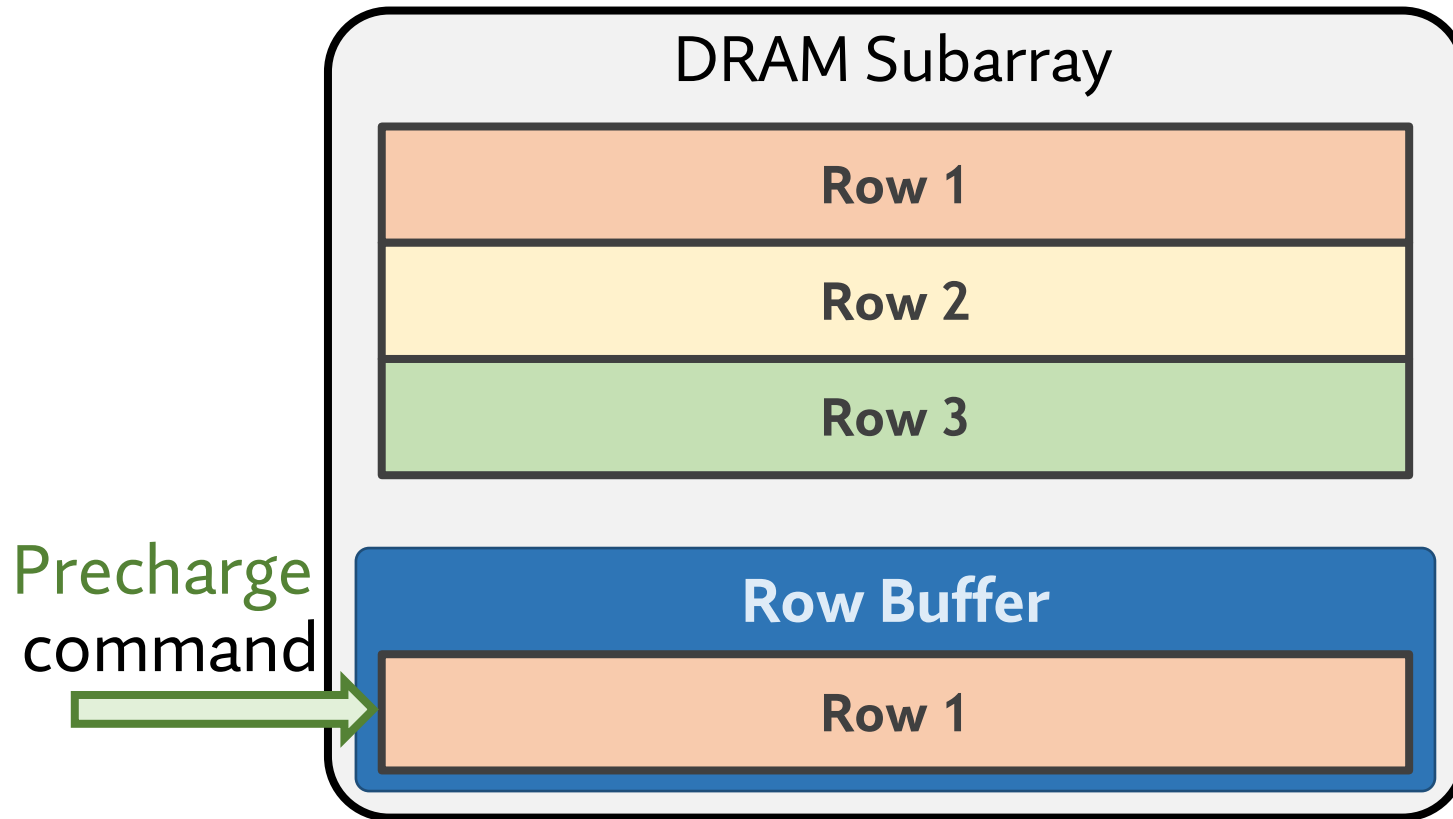
Activate
command



Row 1 is closed

DRAM Operation: Activate and Precharge

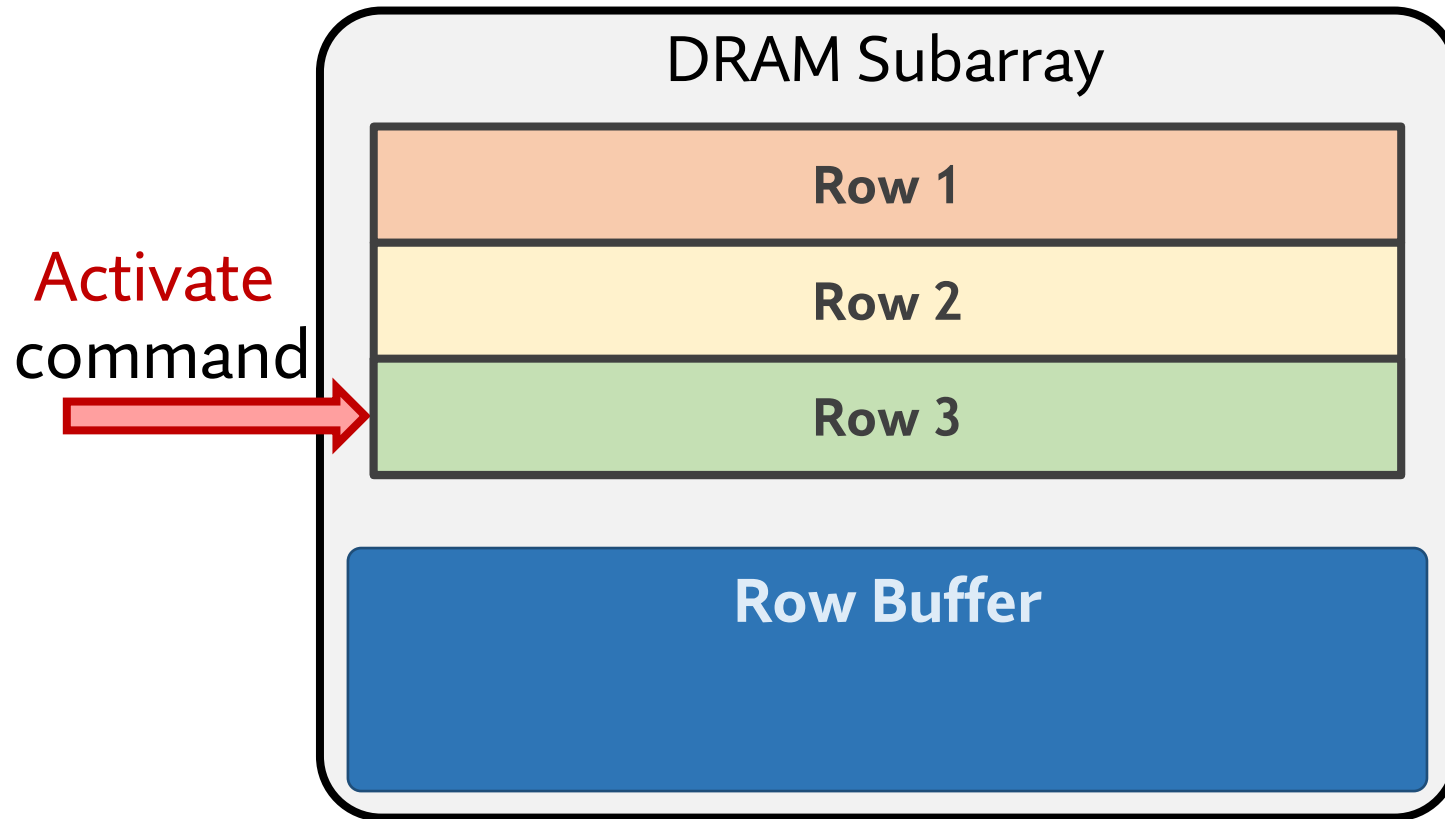
Access data in Row 3



Row 3 is closed

DRAM Operation: Activate and Precharge

Access data in Row 3



Row 3 is closed