

# Computer Architecture

## Lecture 7: Cutting Edge Research on DRAM Read Disturbance

A. Giray Yaglikci, Ataberk Olgun, Haocong Luo

Prof. Onur Mutlu

ETH Zürich

Fall 2023

19 October 2023

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- Understanding Read Disturbance in DRAM Chips
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# Outline

---

- **Infrastructure**
  - **DRAM Bender [Olgun+ IEEE TCAD'23]**
- Understanding Read Disturbance in DRAM Chips
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# DRAM Bender

---

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

Yahya Can Tugrul

Minesh Patel

Hasan Hassan

Lois Orosa

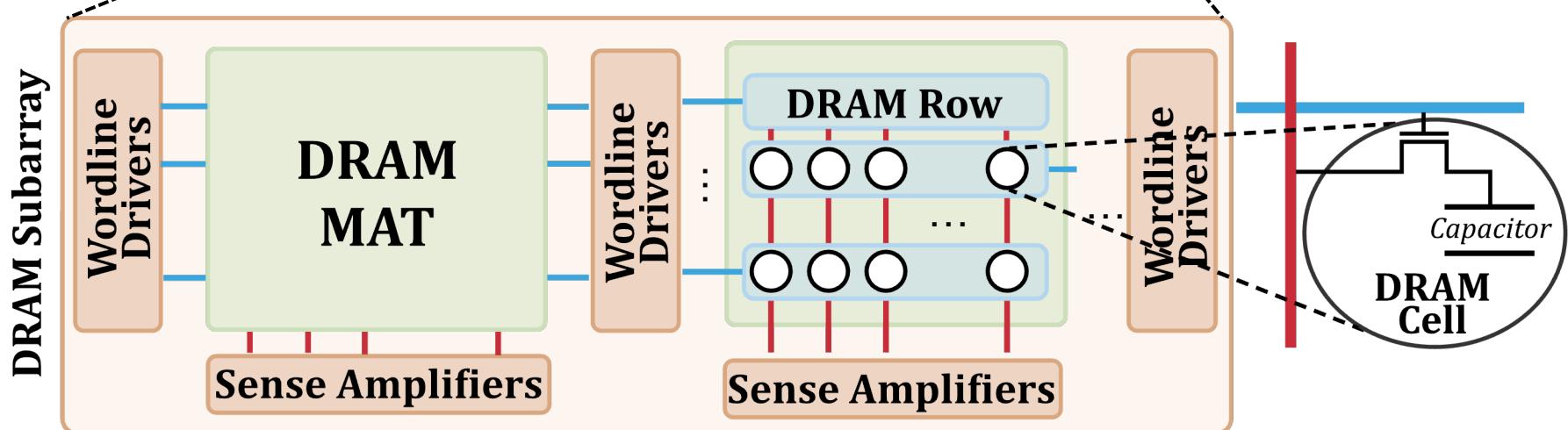
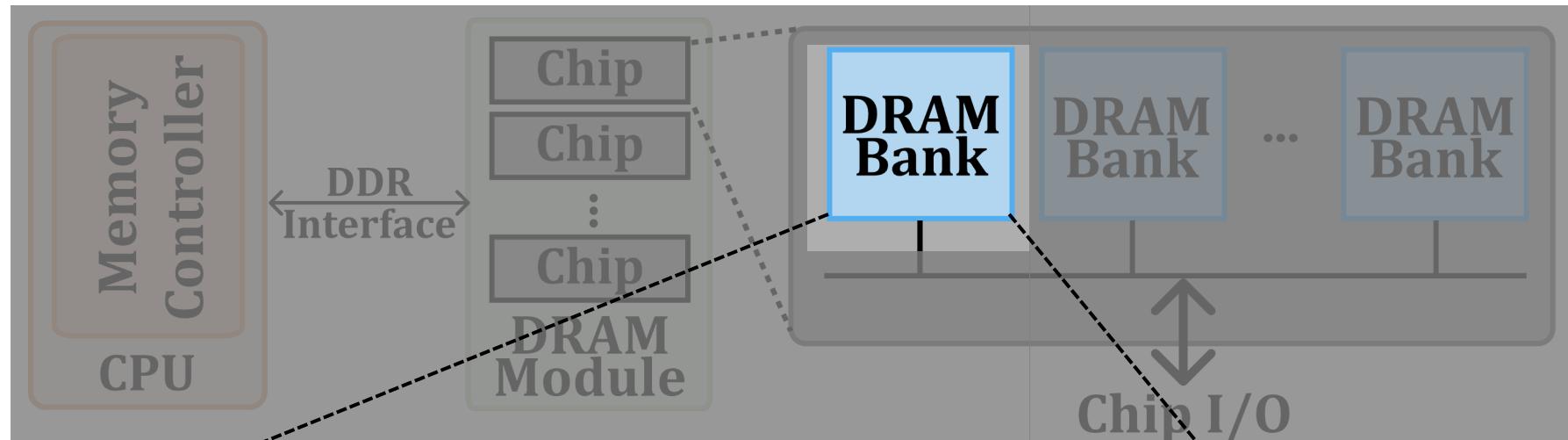
Oguz Ergin

A. Giray Yaglikci

Haocong Luo

Onur Mutlu

# DRAM Organization



# Factors Affecting DRAM Reliability and Latency



*DRAM timing  
violation*



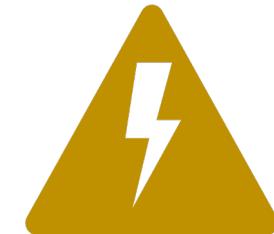
*Inter-cell  
interference*



*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**

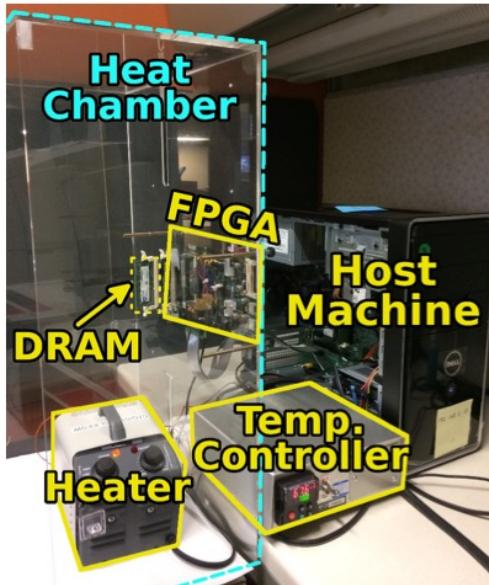
# DRAM Testing Infrastructure

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	✗	✓
<b>DRAM Bender (this work)</b>	<b>No Restrictions</b>	✓	✓

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

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

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

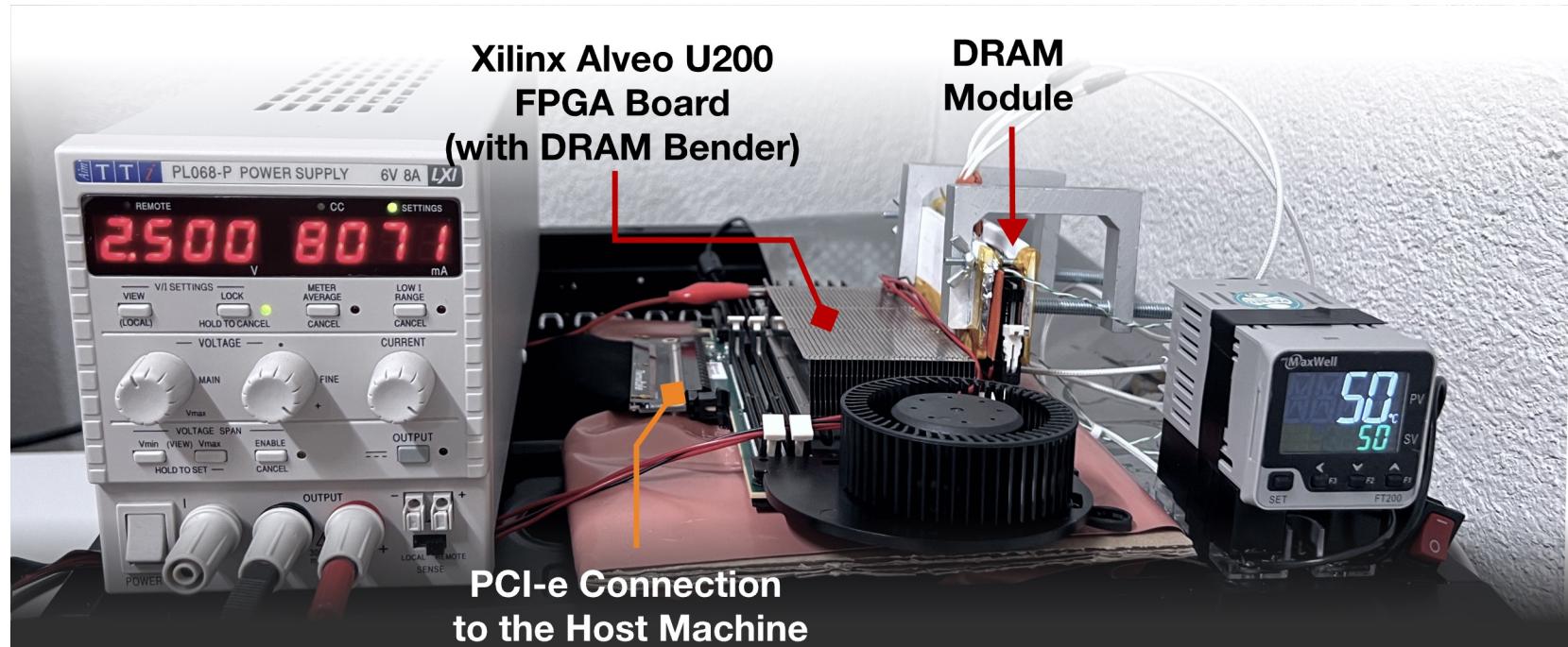
# DRAM Bender: Design Goals

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

# DRAM Bender: Overview

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

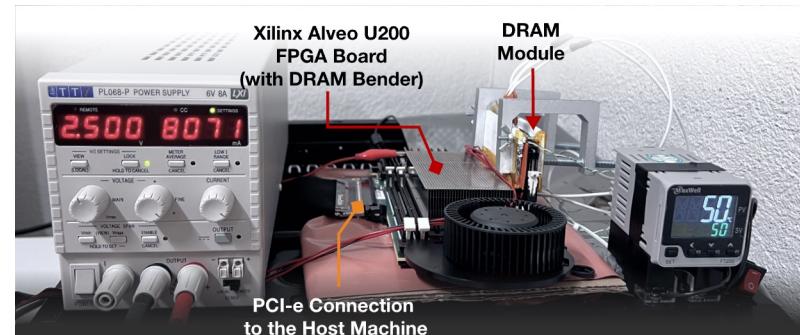
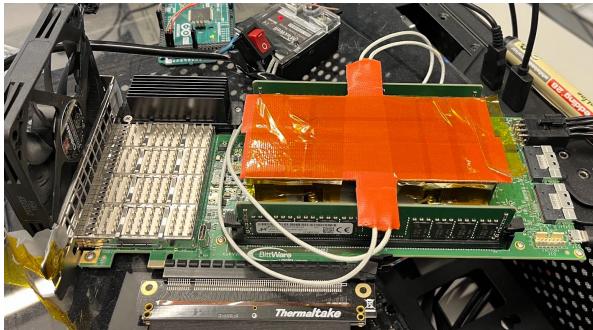
Easily programmable using the DRAM Bender C++ API



# DRAM Bender: Prototypes

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

Five out of the box FPGA-based prototypes



# DRAM Bender: Three Case Studies

1. RowHammer: Interleaving Pattern of Activations
  - Interleaving pattern **significantly affects** the **number** of RowHammer bitflips
2. RowHammer: Random Data Patterns
  - Use **512-bit random** data **patterns**
  - **Uncover more** bitflips than **8-bit** SoftMC random patterns
3. In-DRAM Bitwise Operations
  - Demonstrate in-DRAM **bitwise AND/OR** capability in **real DDR4 chips**

---

DRAM Bender is flexible:  
supports many different types of experiments

# DRAM Bender: Ease of Use

Easily programmable using the DRAM Bender C++ API

## 1. RowHammer: Interleaving Pattern of Activations

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

*one iteration of the RowHammer test*

Easy to devise new experiments to uncover new insights.

# More in the paper (II)

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

# More in the paper (II)

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

Ataberk Olgun<sup>§</sup>    Hasan Hassan<sup>§</sup>    A. Giray Yağlıkçı<sup>§</sup>    Yahya Can Tuğrul<sup>§†</sup>  
Lois Orosa<sup>§○</sup>    Haocong Luo<sup>§</sup>    Minesh Patel<sup>§</sup>    Oğuz Ergin<sup>†</sup>    Onur Mutlu<sup>§</sup>  
<sup>§</sup>*ETH Zürich*    <sup>†</sup>*TOBB ETÜ*    <sup>○</sup>*Galician Supercomputing Center*



<https://arxiv.org/abs/2211.05838>

# Research DRAM Bender Enabled

- 1) [ISCA'23] Luo+, "[RowPress: Amplifying Read Disturbance in Modern DRAM Chips](#)"
- 2) [DSN'23 Disrupt] Olgun+, "[An Experimental Analysis of RowHammer on HBM2 DRAM Chips](#)"
- 3) [arXiv Preprint, 2023] Orosa+, "[SpyHammer: Using RowHammer to Remotely Spy on Temperature](#)"
- 4) [MICRO'22] Yaglikci+, "[HIRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips](#)"
- 5) [DSN'22] Yaglikci+, "[Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices](#)"
- 6) [MICRO'21] Orosa+, "[A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses](#)"
- 7) [MICRO'21] Hassan+, "[Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications](#)"
- 8) [ISCA'21] Olgun+, "[QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips](#)"
- 9) [ISCA'21] Orosa+, "[CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations](#)"
- 10) [ISCA'20] Kim+, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)"
- 11) [S&P'20] Frigo+, "[TRRespass: Exploiting the Many Sides of Target Row Refresh](#)"
- 12) [HPCA'19] Kim+, "[D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput](#)"
- 13) [MICRO'19] Koppula+, "[EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM](#)"
- 14) [SIGMETRICS'18] Ghose+, "[What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study](#)"
- 15) [SIGMETRICS'17] Chang+, "[Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms](#)"
- 16) [MICRO'17] Khan+, "[Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content](#)"
- 17) [SIGMETRICS'16] Chang+, "[Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization](#)"

# More Research DRAM Bender Enabled

- 18) [DRAMSec'23] Lang+, "[BLASTER: Characterizing the Blast Radius of Rowhammer](#)"
- 19) [Applied Sciences'22] Bepary+, "[DRAM Retention Behavior with Accelerated Aging in Commercial Chips](#)"
- 20) [ETS'21] Farmani+, "[RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules](#)"
- 21) [HOST'20] Talukder+, "[Towards the Avoidance of Counterfeit Memory: Identifying the DRAM Origin](#)"
- 22) [MICRO'19] Gao+, "[ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs](#)"
- 23) [IEEE Access'19] Talukder+, "[PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures](#)"
- 24) [ICCE'18] Talukder+, "[Exploiting DRAM Latency Variations for Generating True Random Numbers](#)"

# Summary

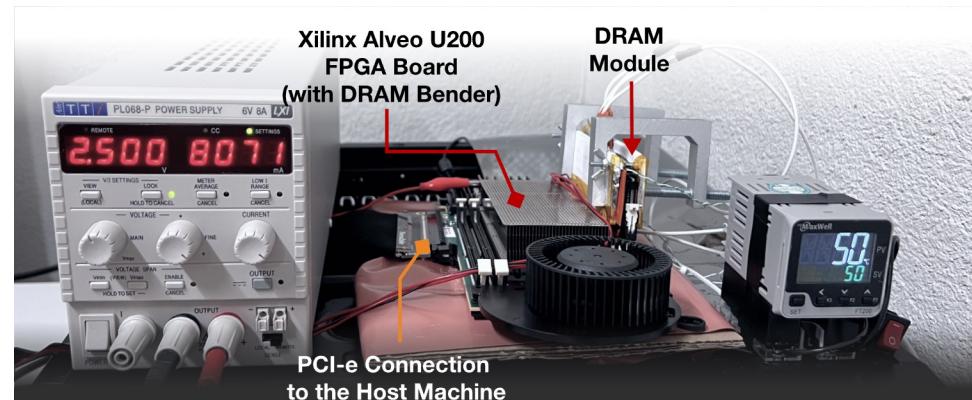
## DRAM Bender

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

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



[github.com/CMU-SAFARI/DRAMBender](https://github.com/CMU-SAFARI/DRAMBender)



[Yaglikci+, DSN'22]

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

# DRAM Bender

- 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<sup>§</sup>      Hasan Hassan<sup>§</sup>      A. Giray Yağlıkçı<sup>§</sup>      Yahya Can Tuğrul<sup>§†</sup>  
Lois Orosa<sup>§○</sup>      Haocong Luo<sup>§</sup>      Minesh Patel<sup>§</sup>      Oğuz Ergin<sup>†</sup>      Onur Mutlu<sup>§</sup>  
<sup>§</sup>*ETH Zürich*      <sup>†</sup>*TOBB ETÜ*      <sup>○</sup>*Galician Supercomputing Center*

# DRAM Bender

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

Ataberk Olgun

Yahya Can Tugrul

Minesh Patel

Hasan Hassan

Lois Orosa

Oguz Ergin

A. Giray Yaglikci

Haocong Luo

Onur Mutlu

# DRAM Bender

---

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)]

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- **Understanding Read Disturbance in DRAM Chips**
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# A Deeper Look into RowHammer

---

Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,

**"A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"**

*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](#) (21 minutes)]

[[Lightning Talk Video](#) (1.5 minutes)]

[[arXiv version](#)]

# *A Deeper Look into RowHammer's Sensitivities*

*Experimental Analysis of Real DRAM Chips  
and Implications on Future Attacks and Defenses*

**Lois Orosa    Abdullah Giray Yağlıkçı**

Haocong Luo   Ataberk Olgun   Jisung Park

Hasan Hassan   Minesh Patel   Jeremie S. Kim   Onur Mutlu

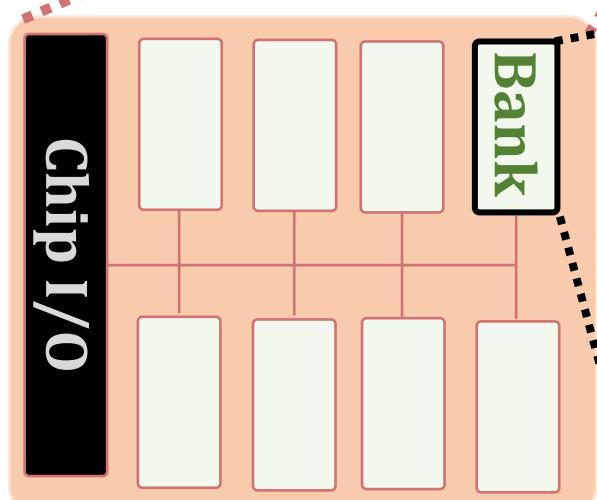
**SAFARI**

**ETH** zürich

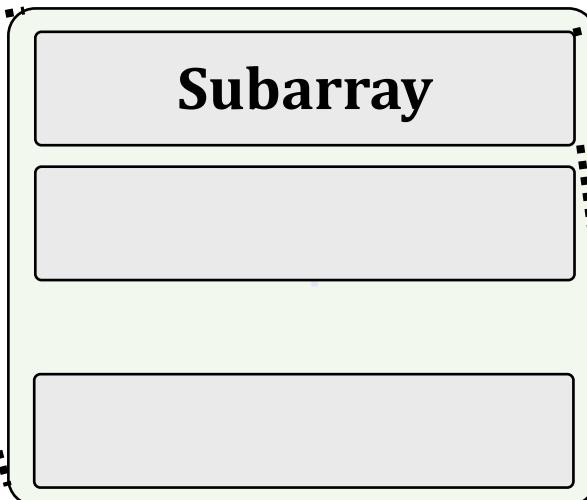


**TOBB ETÜ**  
University of Economics & Technology

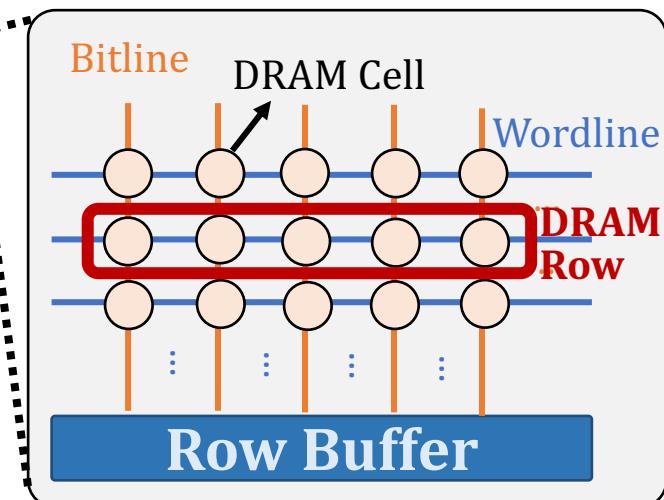
# DRAM Organization



DRAM Chip



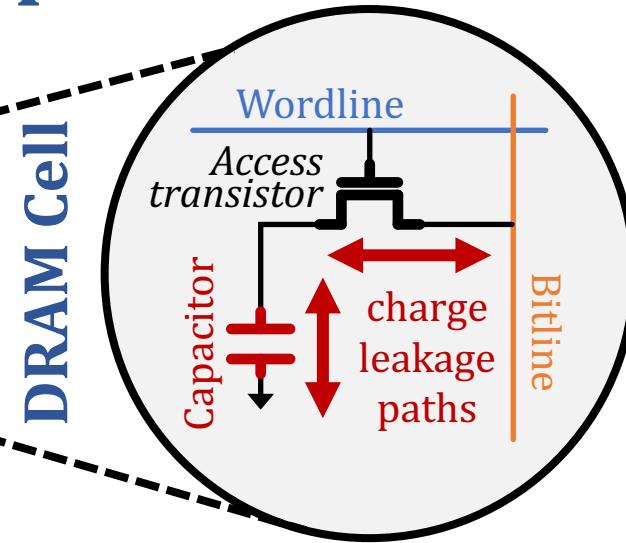
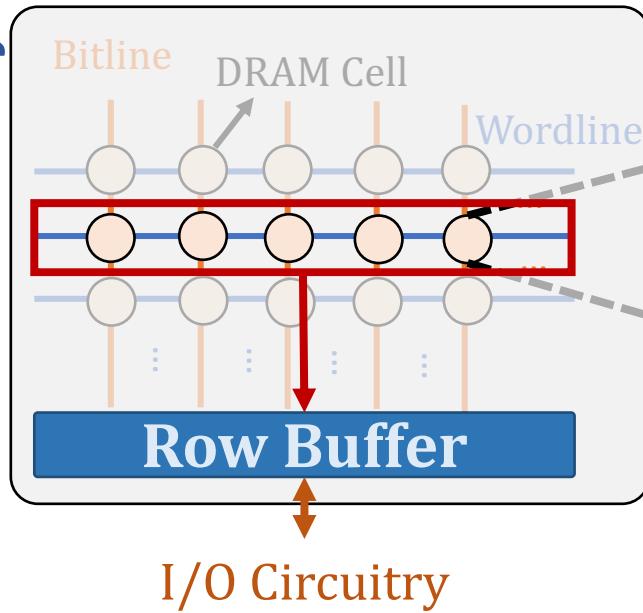
DRAM Bank



DRAM Subarray

# DRAM Organization and Operation

DRAM Subarray

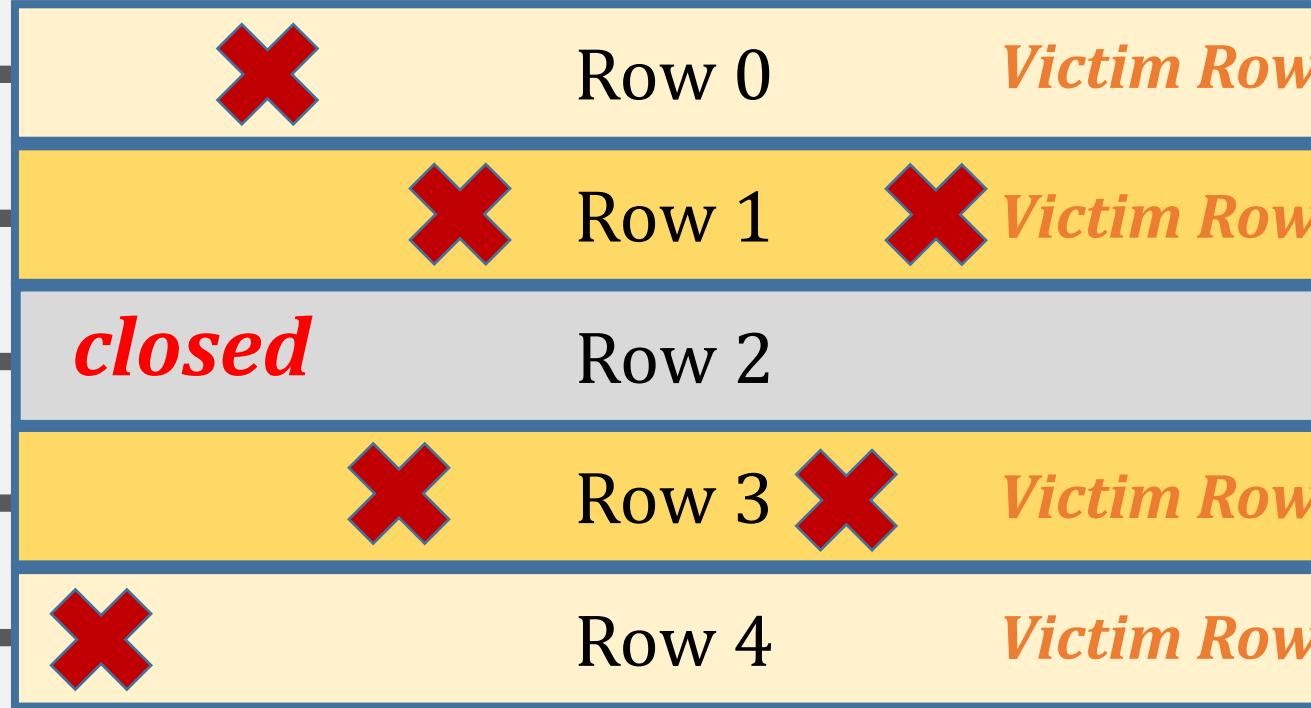


**Refresh:** Restores the capacitor voltage with a time period called **refresh window**

- 1. Row Activation:** Fetch the row's content into the row buffer
- 2. Column Access:** Read/Write a column in the row buffer
- 3. Precharge:** Disconnect the row from the row buffer

# The RowHammer Vulnerability

## DRAM Subarray



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

# Executive Summary

- **Motivation:**
  - Denser DRAM chips are **more vulnerable** to RowHammer
  - Understanding RowHammer enables designing **effective and efficient solutions**, but **no rigorous study** demonstrates how vulnerability varies under different conditions
- **Goal:** Provide insights into **three fundamental properties** of RowHammer that can be leveraged to design **more effective and efficient attacks and defenses**
  - 1) DRAM chip **temperature**
  - 2) The time that an **aggressor row stays active**
  - 3) Victim DRAM cell's **physical location**
- **Experimental study:** 272 DRAM chips from four major manufacturers
- **Key Results:** We provide **6 takeaways** from **16 novel observations**

A RowHammer bit flip is **more likely to occur**

  - 1) in a **bounded range of temperature**
  - 2) if the aggressor row is **active for longer time**
  - 3) in **certain physical regions** of the DRAM module under attack
- **Conclusion:** Our novel observations can inspire and aid future work
  - Craft **more effective attacks**
  - Design **more effective and efficient defenses**

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

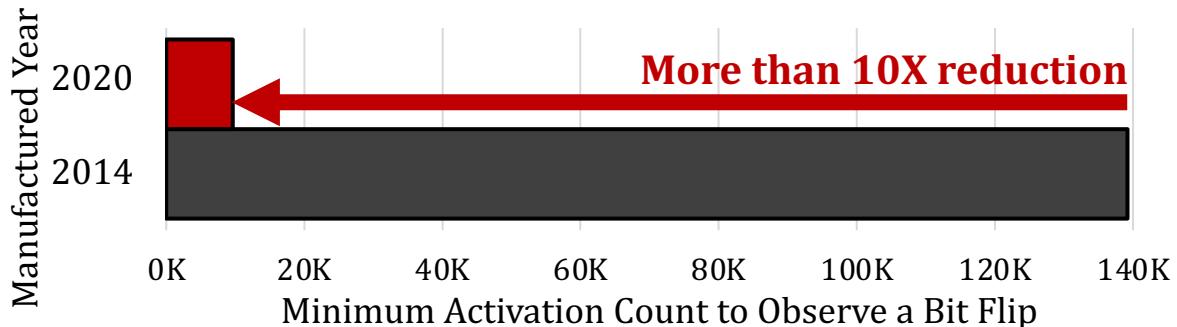
Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Motivation

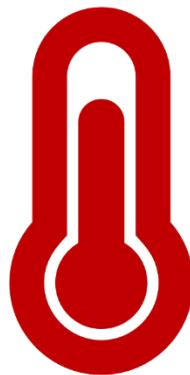


- Defenses are becoming **prohibitively expensive**
- A **deeper understanding** is needed
- **No rigorous experimental study** on fundamental properties of RowHammer to find **effective and efficient** solutions

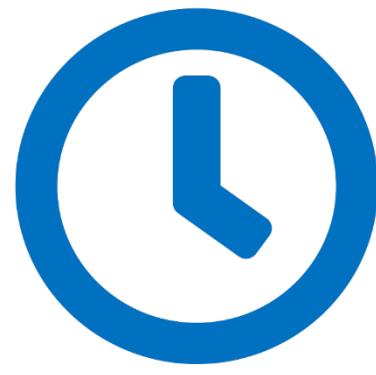
It is **critical** to gain insights into RowHammer and its **fundamental properties**

# Our Goal

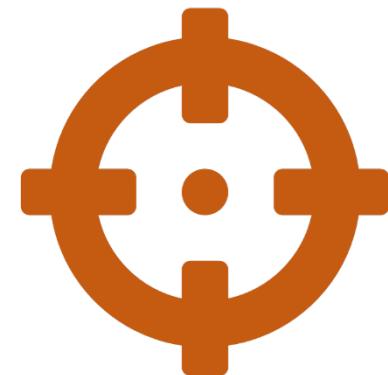
Provide insights into **three fundamental properties**



Temperature



Aggressor Row  
Active Time



Victim DRAM Cell's  
Physical Location

To find **effective and efficient** attacks and defenses

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

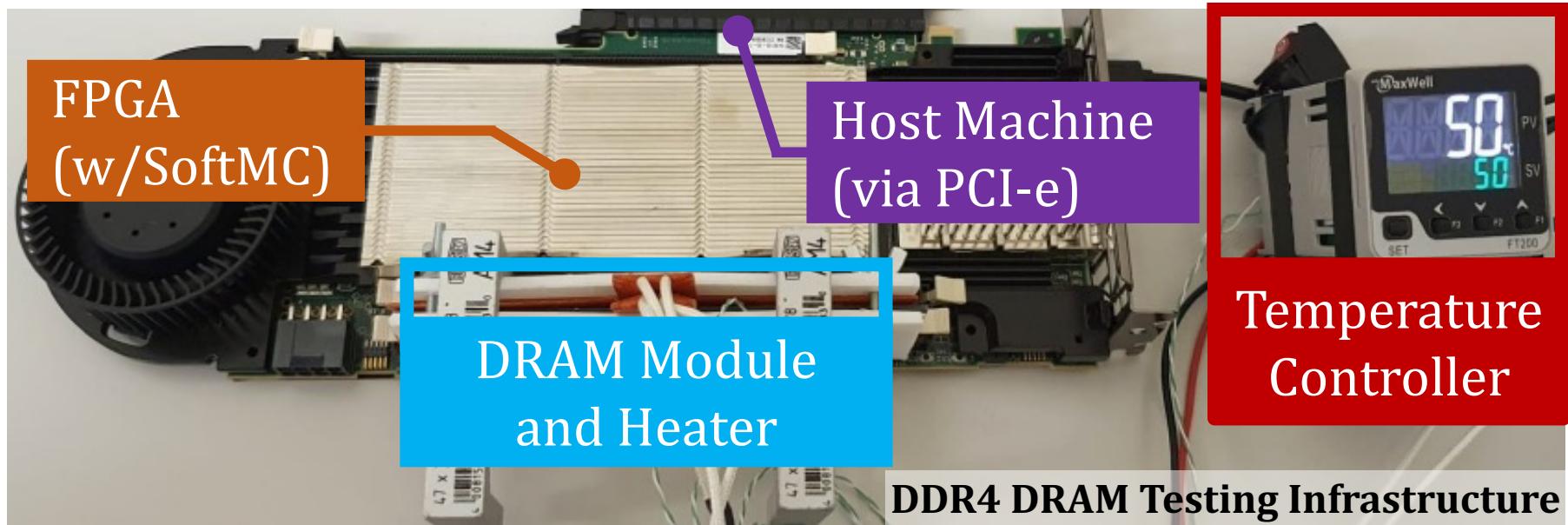
Conclusions

# DRAM Testing Infrastructures

Two separate testing infrastructures

1. DDR3: FPGA-based SoftMC (Xilinx ML605)

2. DDR4: FPGA-based SoftMC (Xilinx Virtex UltraScale+ XCU200)



Fine-grained control over **DRAM commands**,  
**timing parameters** and **temperature ( $\pm 0.1^\circ\text{C}$ )**

# DRAM Testing Methodology

To characterize our DRAM chips at **worst-case** conditions:

## 1. Prevent sources of interference during core test loop

- **No DRAM refresh**: to avoid refreshing victim row
- **No DRAM calibration events**: to minimize variation in test timing
- **No RowHammer mitigation mechanisms**: to observe circuit-level effects
- Test for **less than a refresh window (32ms)** to avoid retention failures

## 2. Worst-case access sequence

- We use **worst-case** access sequence based on prior works' observations
- For each row, **repeatedly access the two physically-adjacent rows as fast as possible**

# DRAM Chips Tested

Mfr.	DDR4 DIMMs	DDR3 SODIMMs	# Chips	Density	Die	Org.
A (Micron)	9	1	144 (8)	8Gb (4Gb)	B (P)	x4 (x8)
B (Samsung)	4	1	32 (8)	4Gb (4Gb)	F (Q)	x8 (x8)
C (SK Hynix)	5	1	40 (8)	4Gb (4Gb)	B (B)	x8 (x8)
D (Nanya)	4	-	32 (-)	8Gb (-)	C (-)	x8 (-)

Two DRAM standards

4 Major Manufacturers

272 DRAM Chips in total

# DRAM Chips Tested

## A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa\*  
ETH Zürich

A. Giray Yağlıkçı\*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich

- 272
- Four
- DD
- Diff

Table 4: Characteristics of the tested DDR4 and DDR3 DRAM modules.

Type	Chip Manufacturer	Chip Identifier	Module Vendor	Module Identifier	Freq. (MT/s)	Date Code	Density	Die Rev.	Org.	#Modules	#Chips
DDR4	A: Micron	MT40A2G4WE-083E:B	Micron	MTA18ASF2G72PZ-2G3B1QG [94]	2400	1911	8Gb	B	x4	6	96
	B: Samsung	K4A4G085WF-BCTD [132]	G.SKILL	F4-2400C17S-8GNT [35]	2400	1843				2	32
	C: SK Hynix	DWCW (Partial Marking) †	G.SKILL	F4-2400C17S-8GNT [35]	2400	1844				1	16
	D: Nanya	D1028AN9CPGRK ‡	Kingston	KVR24N17S8/ [75]	2400	2021 Jan *	4Gb	F	x8	4	32
DDR3	A: Micron	MT41K512M8DA-107:P [22]	Crucial	CT51264BF160BJ.M8FP	1600	1703	4Gb	P	x8	1	8
	B: Samsung	K4B4G0846Q	Samsung	M471B5173QH0-YK0 [131]	1600	1416	4Gb	Q	x8	1	8
	C: SK Hynix	H5TC4G83BFR-PBA	SK Hynix	HMT451S6BFR8A-PB [139]	1600	1535	4Gb	B	x8	1	8

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Key Takeaways from Temperature Analysis

## Key Takeaway 1

To ensure that a DRAM cell is **not vulnerable** to RowHammer, we **must characterize** the cell at **all operating temperatures**

## Key Takeaway 2

RowHammer vulnerability **tends to worsen**  
**as DRAM temperature increases**

However, **individual DRAM rows** can exhibit behavior  
**different from the dominant trend**

# Impact of Temperature on DRAM Cells



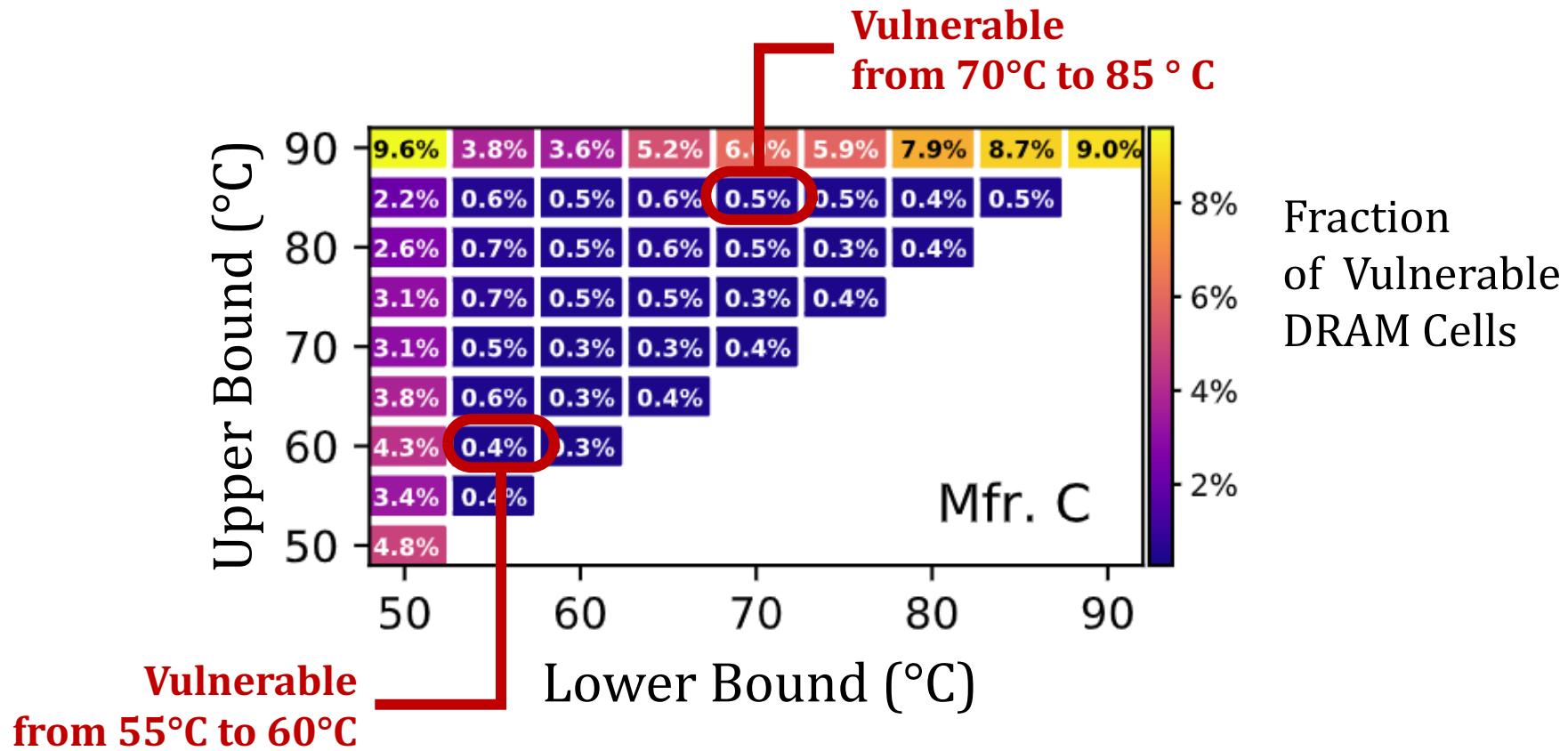
The fraction of vulnerable DRAM cells, experiencing bit flips **at all temperature levels** within their vulnerable temperature range

Mfr. A	Mfr. B	Mfr. C	Mfr. D
99.1%	98.9%	98.0%	99.2%

## OBSERVATION 1

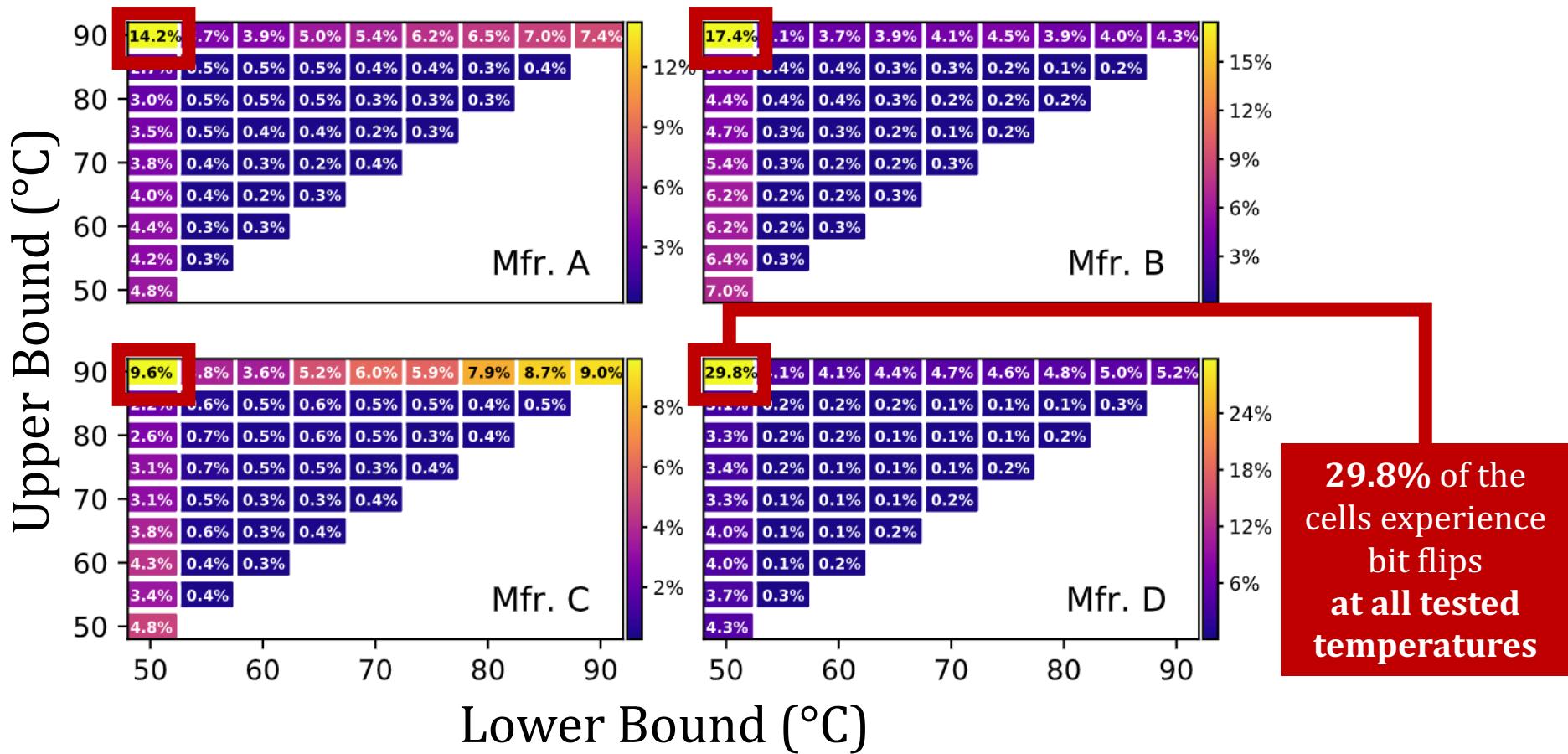
**Most DRAM cells** are vulnerable to RowHammer throughout **a continuous temperature range**

# Impact of Temperature on DRAM Cells



Different DRAM cells are vulnerable to RowHammer  
**within specific temperature ranges**

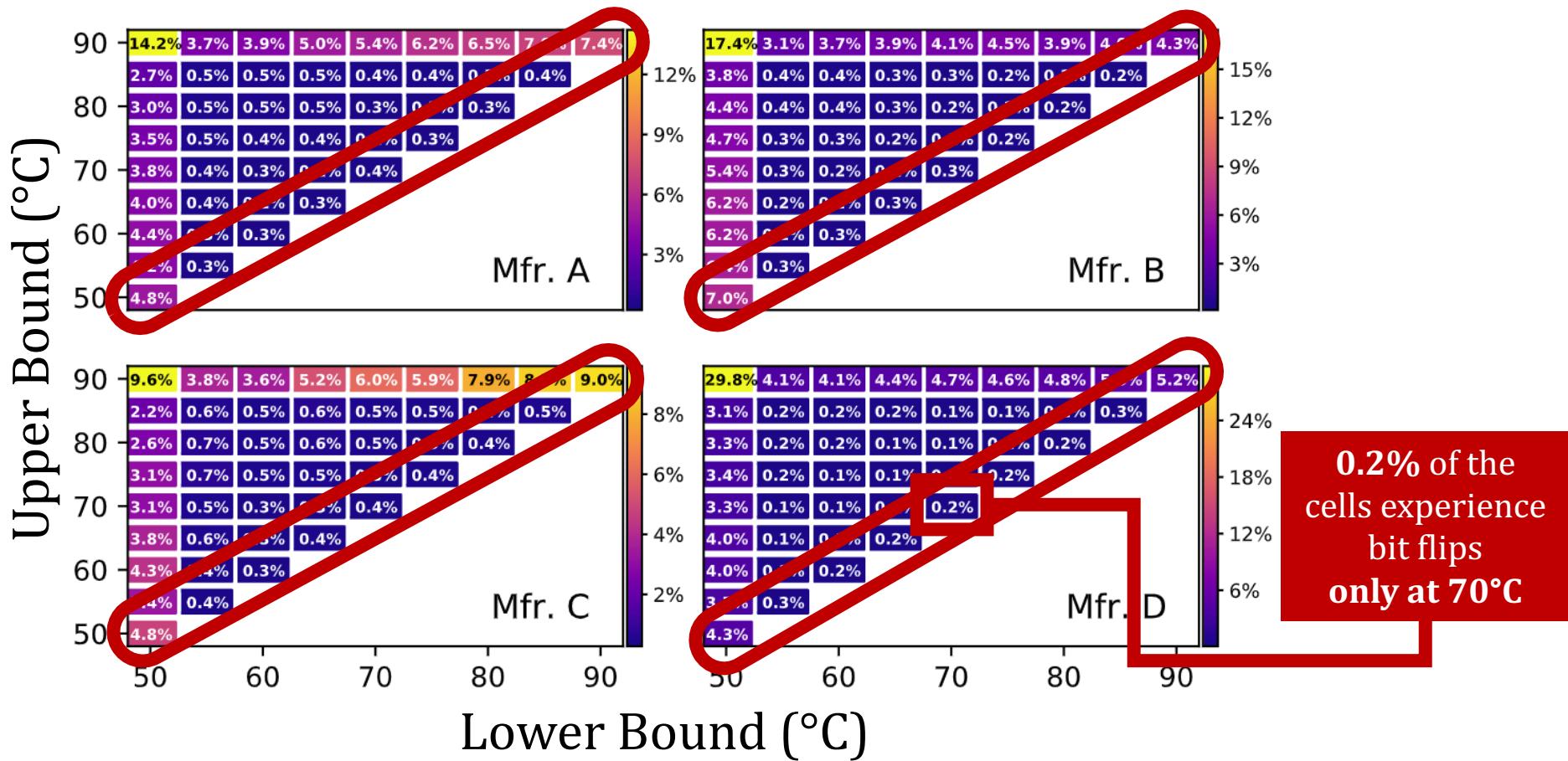
# Impact of Temperature on DRAM Cells



## OBSERVATION 2

A **significant fraction** of vulnerable DRAM cells exhibit bit flips at **all tested temperatures**

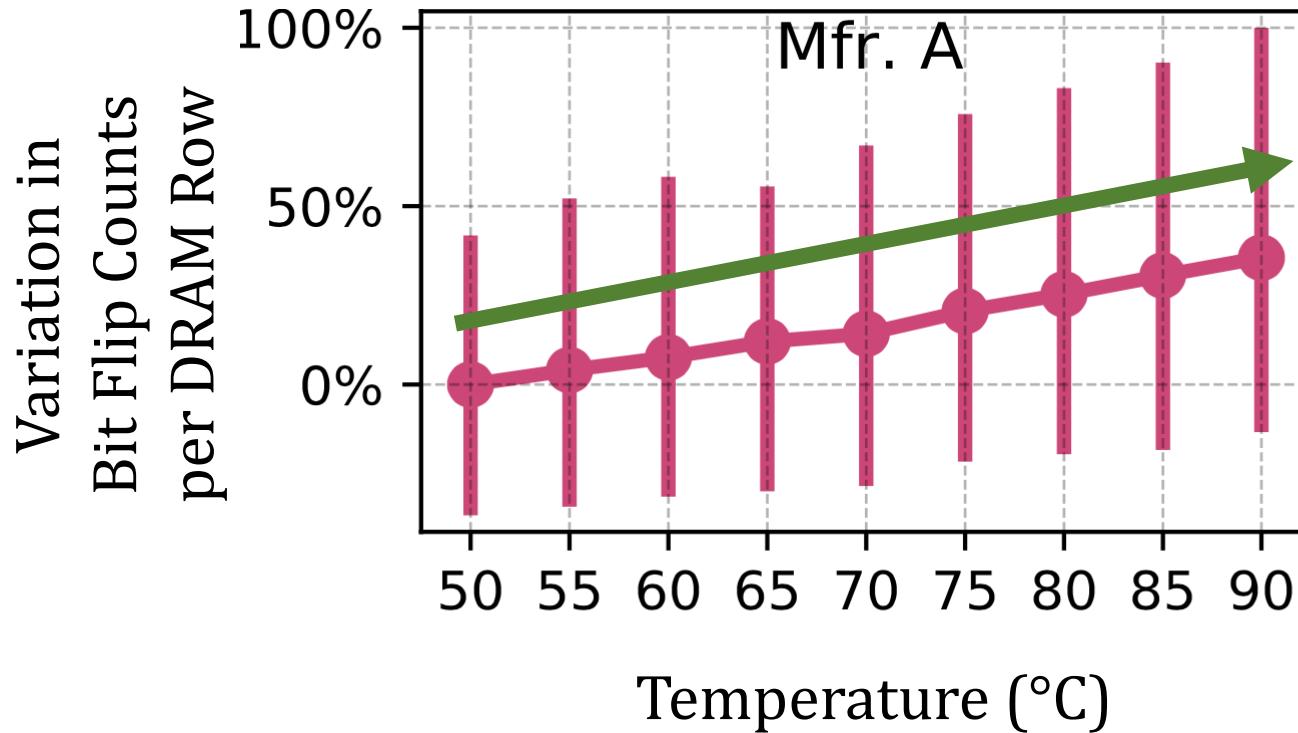
# Impact of Temperature on DRAM Cells



## OBSERVATION 3

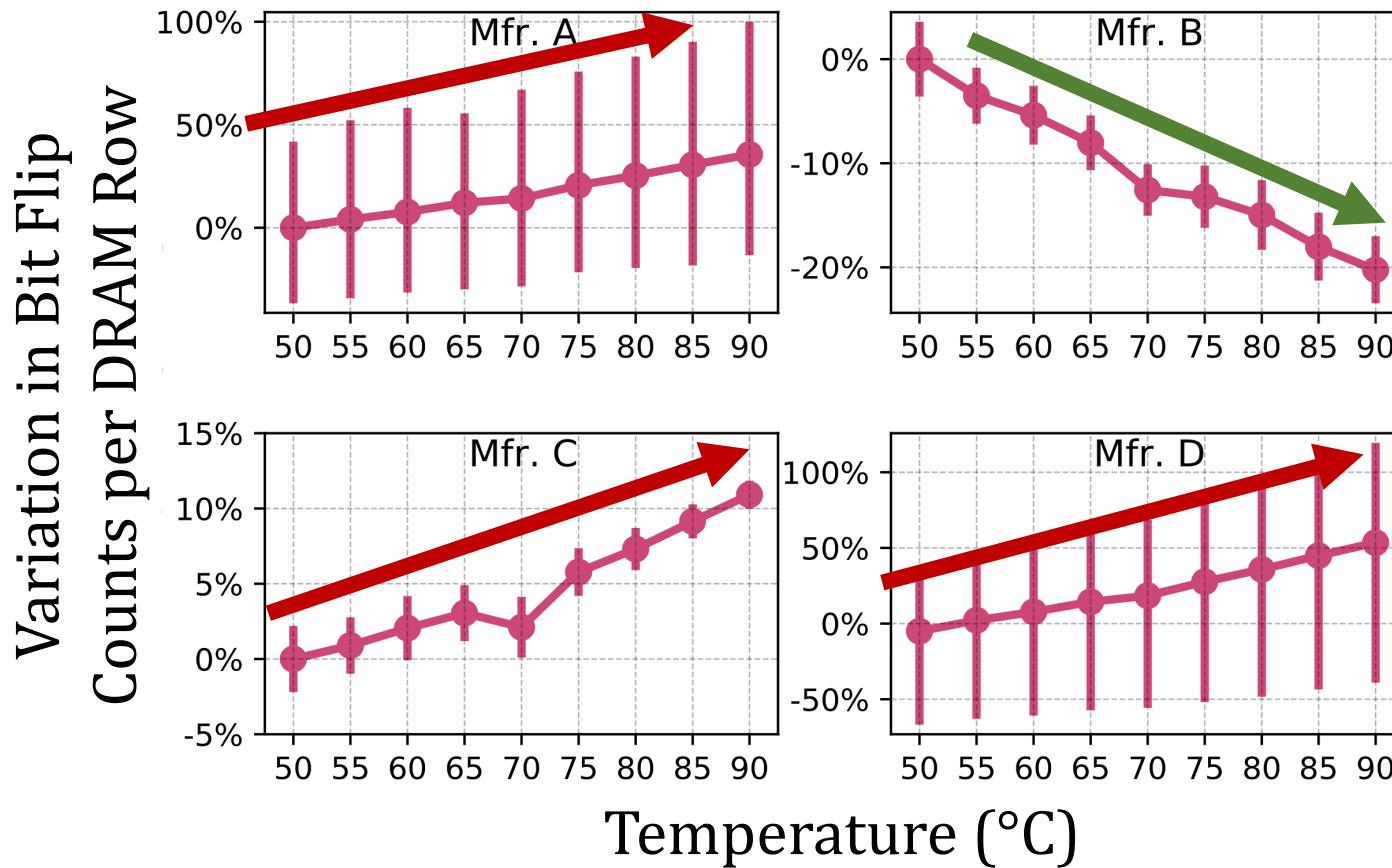
A small fraction of all vulnerable DRAM cells are vulnerable to RowHammer **only in a very narrow temperature range**

# Impact of Temperature on DRAM Rows



**More cells** experience bit flips as **temperature increases**

# Impact of Temperature on DRAM Rows



## OBSERVATION 4

A DRAM row's bit error rate can either **increase or decrease with temperature** depending on the DRAM manufacturer

# Also in the Paper

The **minimum activation count** at which a victim row experiences a bit flip ( $HC_{first}$ ) when **temperature changes**:

## OBSERVATION 5

DRAM rows can show **either higher or lower  $HC_{first}$**  when **temperature increases**

## OBSERVATION 6

$HC_{first}$  tends to generally **decrease** as **temperature change ( $\Delta T$ ) increases**

## OBSERVATION 7

The  $HC_{first}$  **change ( $\Delta HC_{first}$ ) tends to be larger** as **temperature change ( $\Delta T$ ) increases**

# Also in the Paper

The minimum activation count at which a victim row experiences a bit flip ( $HC_{first}$ ) when temperature changes:

## KEY OBSERVATION 5

### A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa\*  
ETH Zürich

A. Giray Yağlıkçı\*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich

## KEY OBSERVATION 7

The  $HC_{first}$  change ( $\Delta HC_{first}$ ) tends to be **larger** as temperature change ( $\Delta T$ ) increases

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Key Takeaways from Aggressor Row Active Time Analysis

## Key Takeaway 3

As an aggressor row stays **active longer**,  
victim DRAM cells become **more vulnerable** to RowHammer

## Key Takeaway 4

RowHammer vulnerability of victim cells **decreases**  
when the bank is **precharged for a longer time**

# Memory Access Patterns in Aggressor Row Active Time Analysis

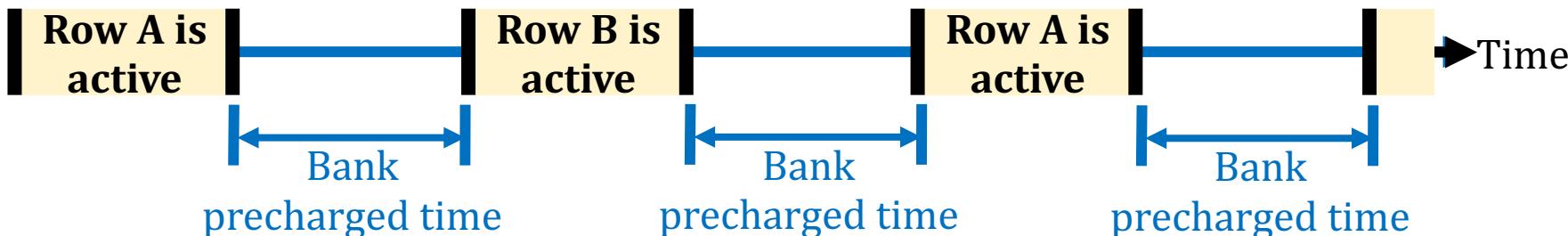
- Baseline access pattern:



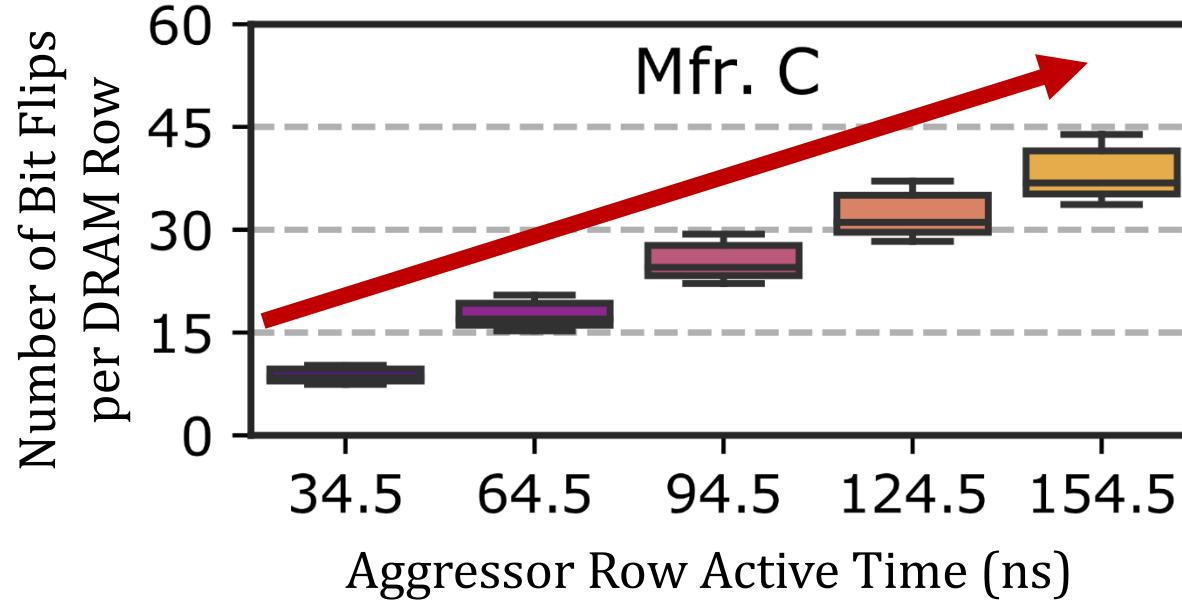
- Increasing **aggressor row active time**:



- Increasing **bank precharged time**:

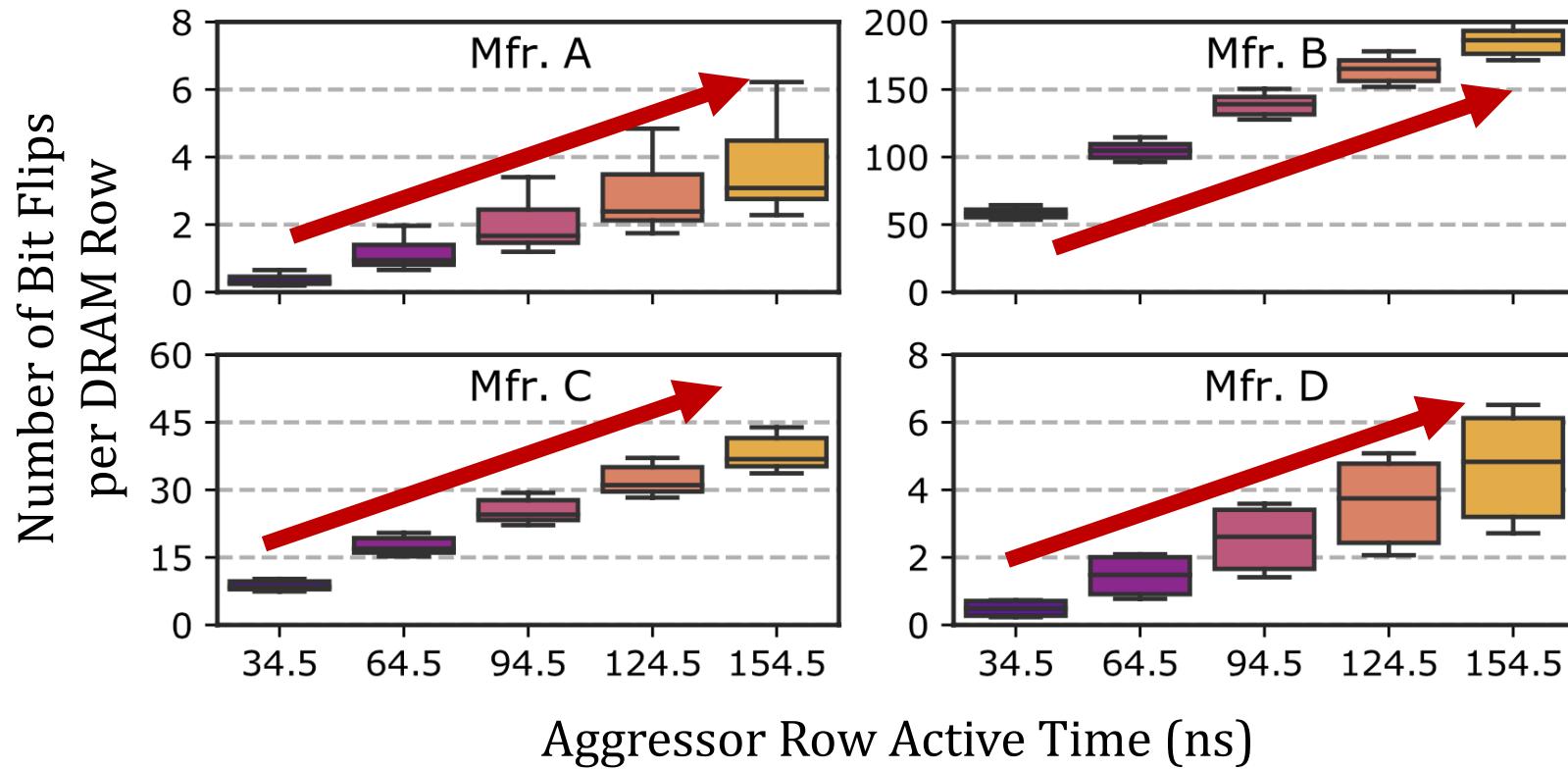


# Increasing Aggressor Row Active Time



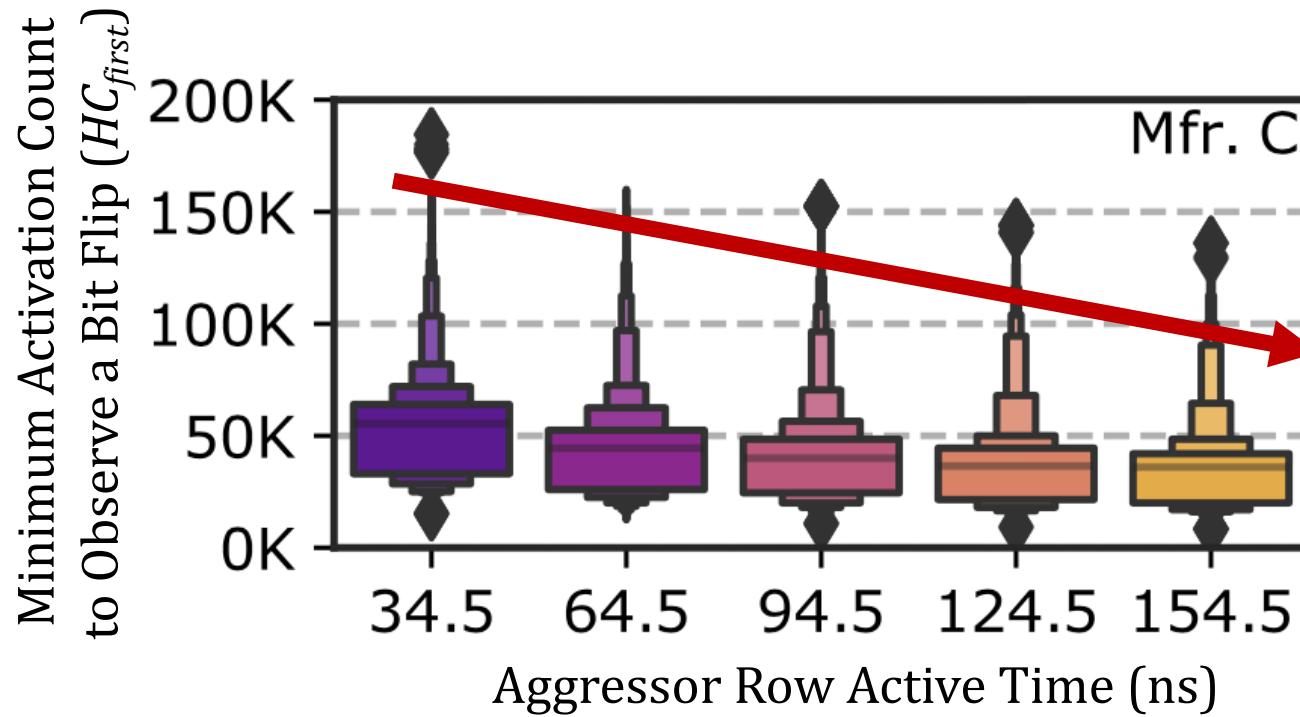
As the **aggressor row stays active longer**,  
**more DRAM cells** experience RowHammer bit flips

# Increasing Aggressor Row Active Time



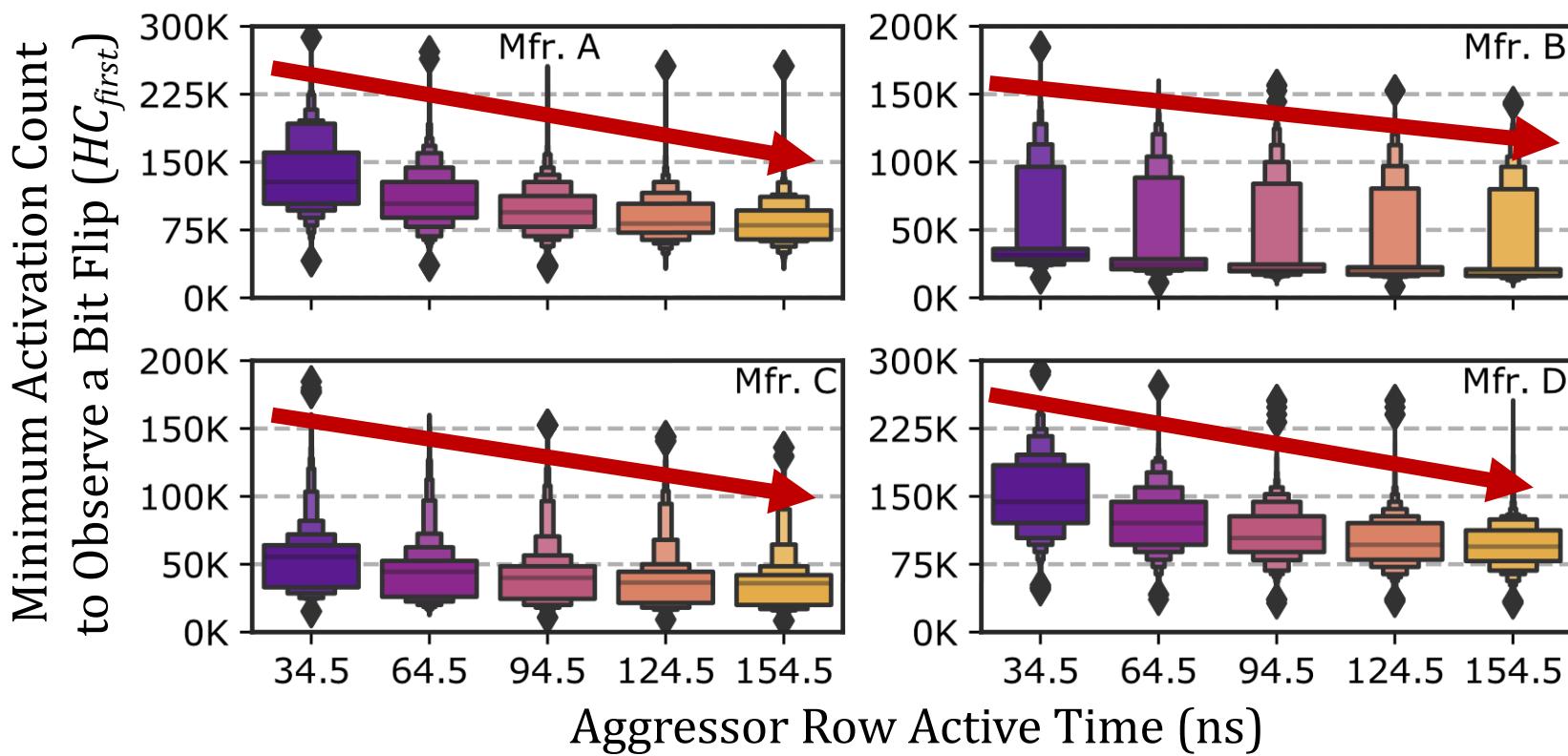
As the **aggressor row stays active longer**,  
**more DRAM cells** experience RowHammer bit flips

# Increasing Aggressor Row Active Time



**Fewer activations** are required to cause RowHammer bit flips when aggressor rows stay active **for longer time**

# Increasing Aggressor Row Active Time



## OBSERVATION 8

As the **aggressor row stays active longer**,  
**more DRAM cells** experience RowHammer bit flips and they  
experience RowHammer bit flips **at lower activation counts**

# Also in the Paper

The **variation** in aggressor row active time's effects across DRAM rows and the effect of increasing **bank precharged time**

## OBSERVATION 9

As the **aggressor row stays active longer**, the RowHammer vulnerability **consistently worsens** across tested DRAM rows

## OBSERVATION 10

As the **bank stays precharged longer**, **fewer DRAM cells** experience RowHammer bit flips and they experience RowHammer bit flips **at higher activation counts**

## OBSERVATION 11

As the **bank stays precharged longer**, the RowHammer vulnerability **consistently reduces** across tested DRAM rows

# Also in the Paper

The variation in these behaviors across DRAM rows and the effect of increasing bank precharged time

## KEY OBSERVATION 9

### A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa\*  
ETH Zürich

A. Giray Yağlıkçı\*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich

## KEY OBSERVATION 11

As the bank stays precharged longer, the RowHammer vulnerability **consistently reduces** across tested DRAM rows

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Key Takeaways from Spatial Variation Analysis

## Key Takeaway 5

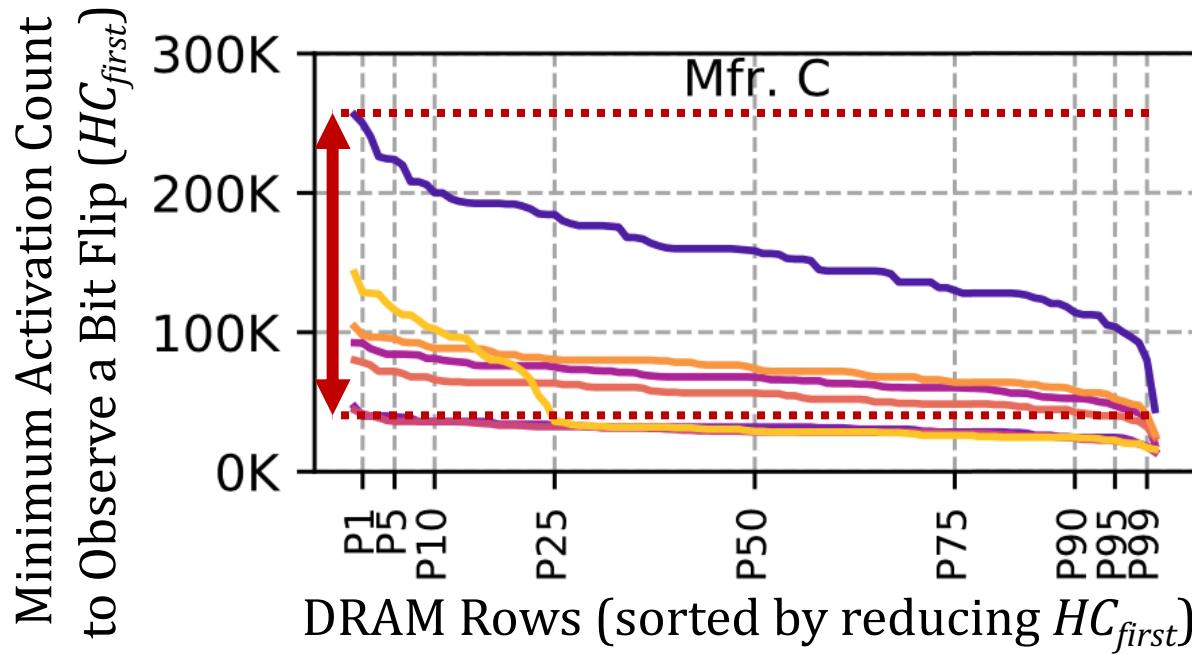
RowHammer vulnerability **significantly varies** across DRAM rows and columns due to **design-induced** and **manufacturing-process-induced** variation

## Key Takeaway 6

The distribution of **the minimum activation count to observe bit flips ( $HC_{first}$ )** exhibits **a diverse set of values in a subarray** but **similar values across subarrays** in the same DRAM module

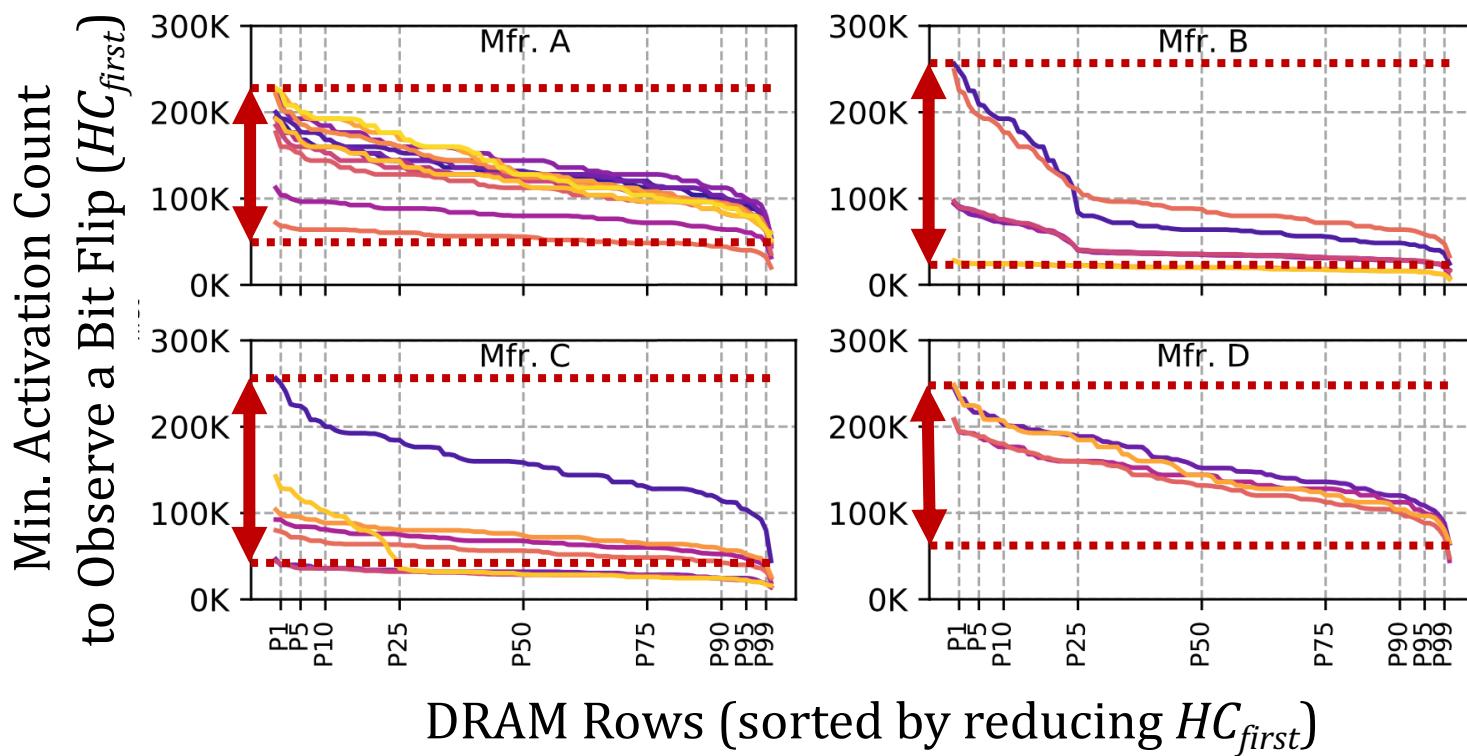
# Spatial Variation across Rows

The **minimum activation count** to observe bit flips ( $HC_{first}$ ) across **DRAM rows**:



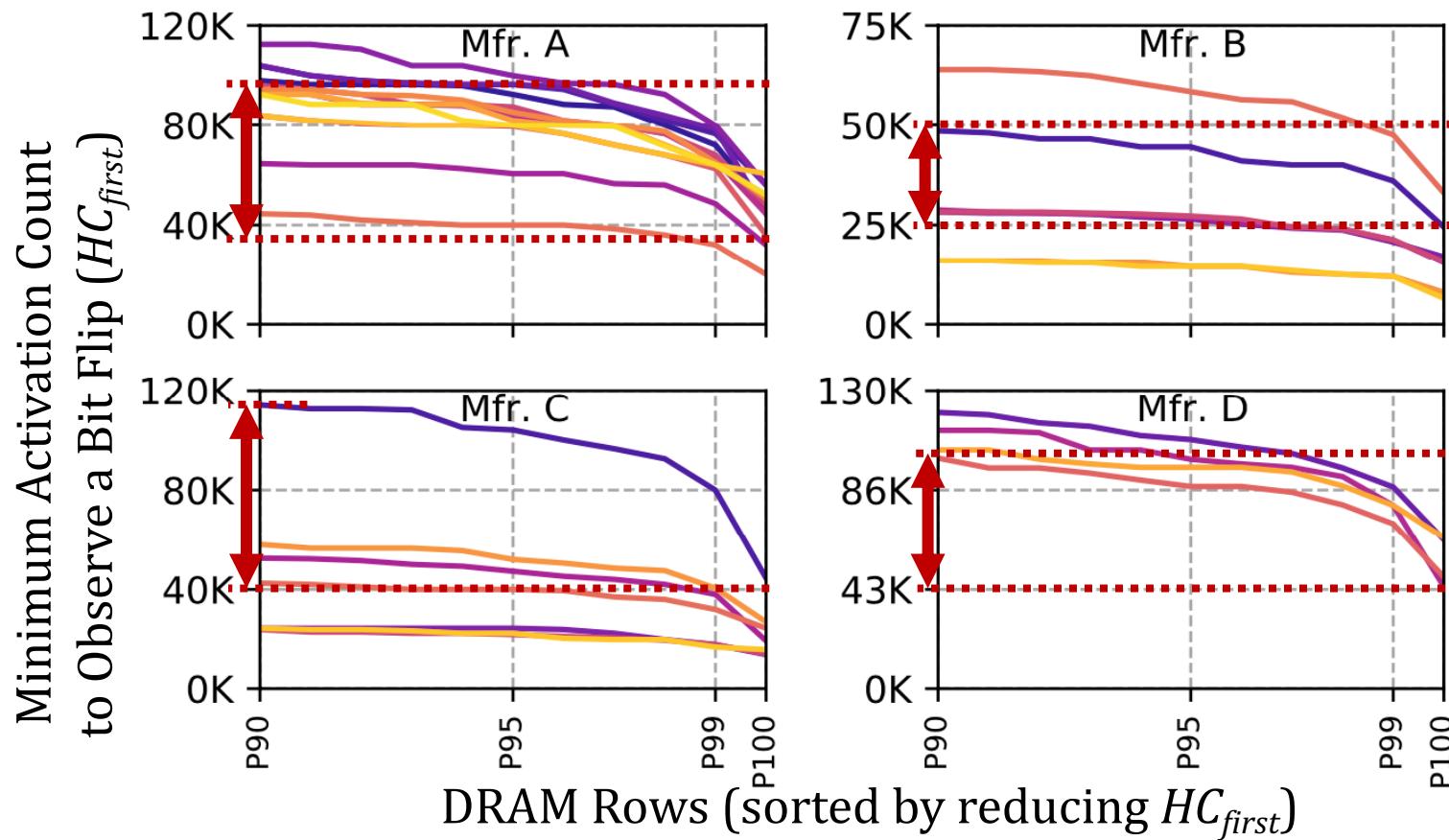
The RowHammer vulnerability  
**significantly varies** across DRAM rows

# Spatial Variation across Rows



The RowHammer vulnerability  
**significantly varies** across DRAM rows

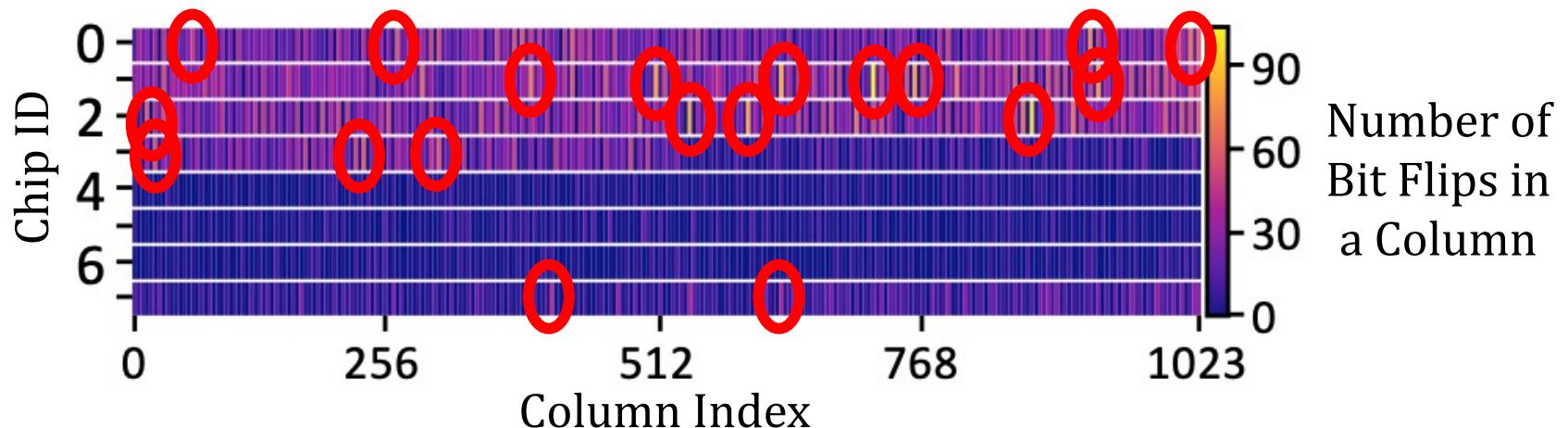
# Spatial Variation across Rows



## OBSERVATION 12

A small fraction of DRAM rows are significantly more vulnerable to RowHammer than the vast majority of the rows

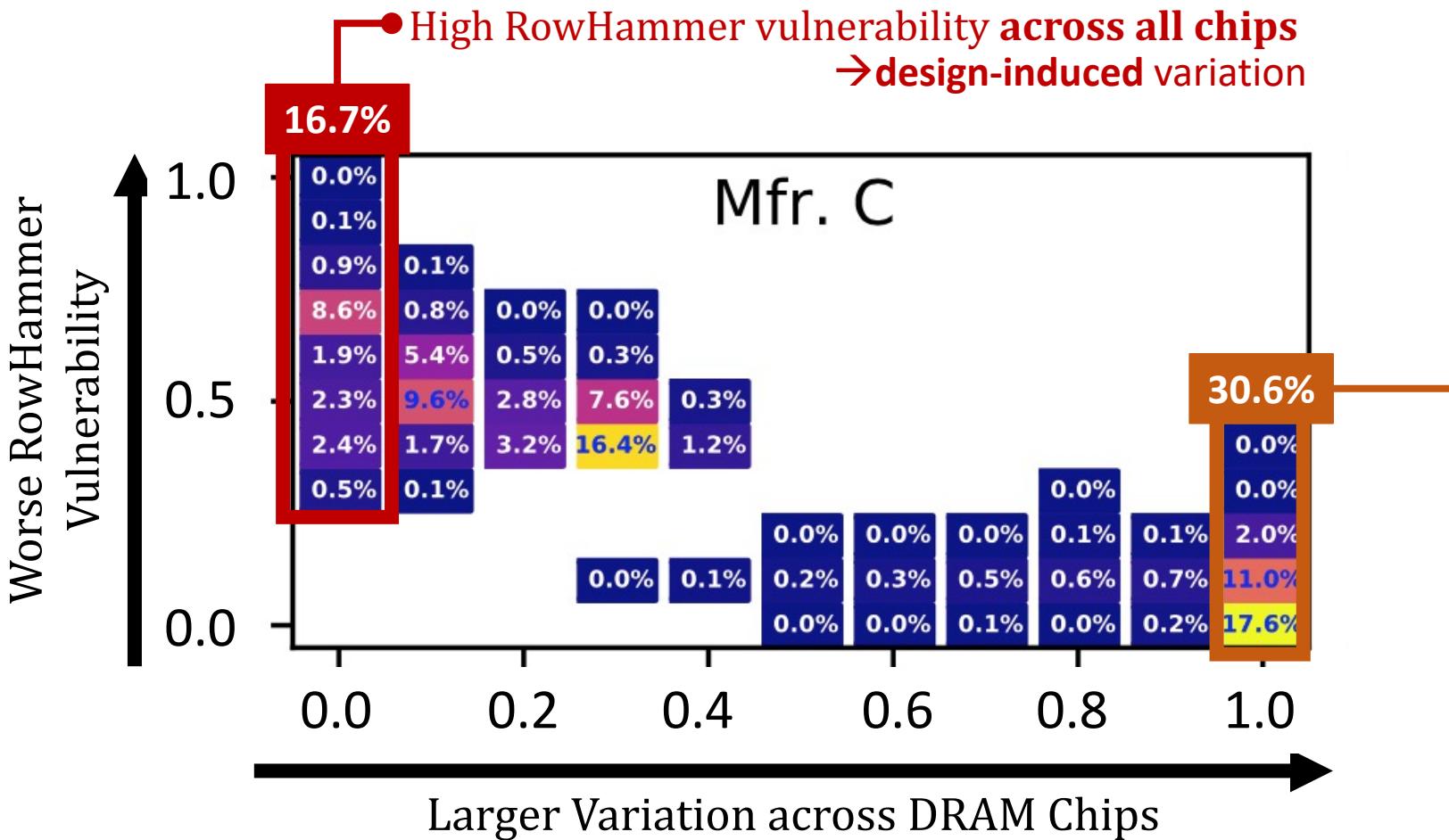
# Spatial Variation across Columns



## OBSERVATION 13

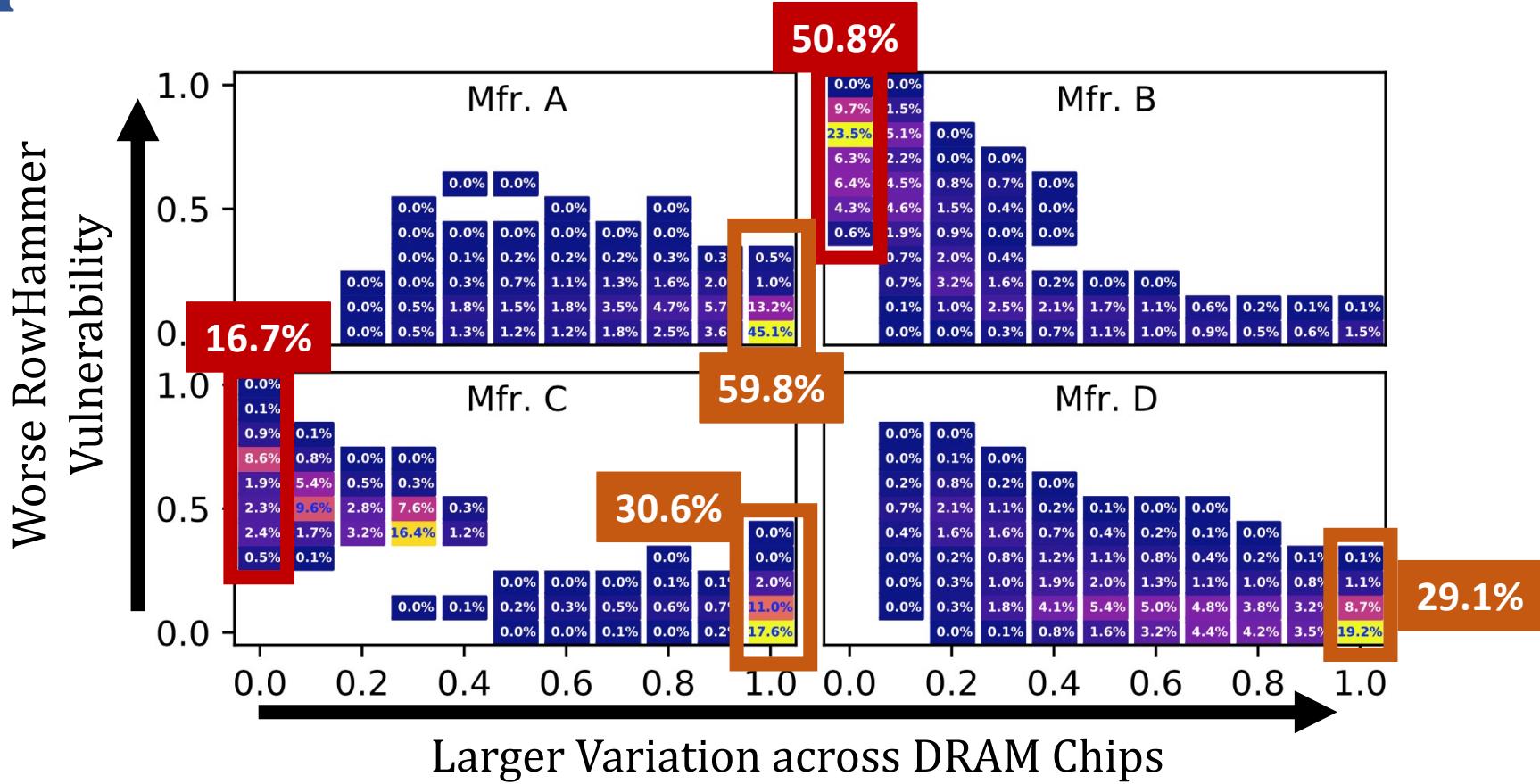
Certain columns are **significantly more vulnerable** to RowHammer than other columns

# Spatial Variation across Columns



High variation in vulnerability across chips  
→ manufacturing-process-induced variation

# Spatial Variation across Columns



## OBSERVATION 14

Both **manufacturing process** and **design** affect a DRAM column's RowHammer vulnerability

# Also in the Paper

The **minimum activation count** at which a victim row experiences a bit flip ( $HC_{first}$ ) across rows in a subarray and across subarrays in a DRAM module:

## OBSERVATION 15

The most vulnerable DRAM row in a subarray  
is **significantly more vulnerable**  
than the other rows **in the subarray**

## OBSERVATION 16

$HC_{first}$  distributions of subarrays **within a DRAM module**  
are **significantly more similar** to each other  
than those of subarrays **from different modules**

# Also in the Paper

The minimum activation count at which a victim row experiences a bit flip ( $HC_{first}$ ) across rows in a subarray and across subarrays in a module:

## A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa\*  
ETH Zürich

A. Giray Yağlıkçı\*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich

$HC_{first}$  distributions of subarrays within a DRAM module  
are significantly more similar to each other  
than those of subarrays from different modules

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Implications on Attacks and Defenses

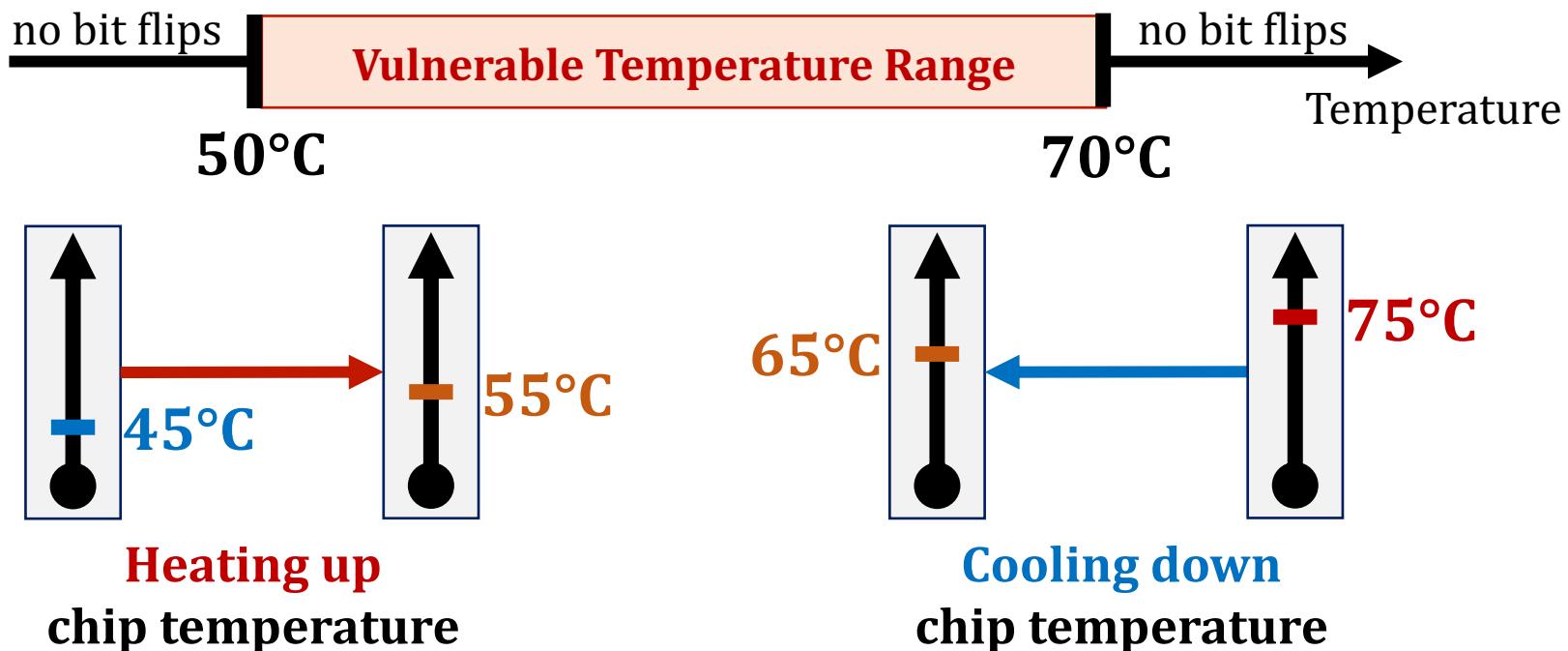
Our observations can be leveraged to craft  
**more effective RowHammer attacks**

Our observations can be leveraged to design  
**more effective and efficient RowHammer defenses**

# Attack Improvement 1: Making DRAM Cells More Vulnerable

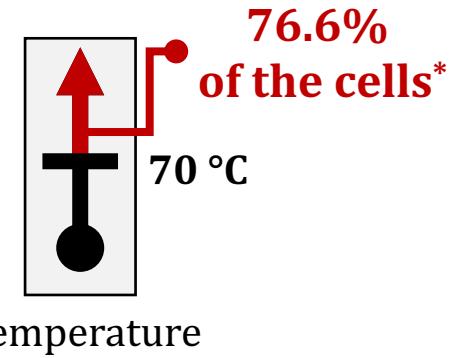
An attacker can **manipulate temperature** to make the cells that store sensitive data **more vulnerable**

DRAM cells are vulnerable in a **bounded temperature range**

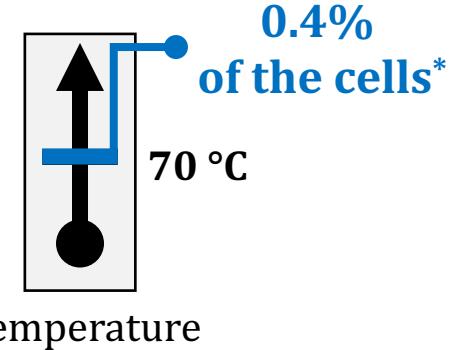


# Attack Improvement 2: Temperature-Dependent Trigger

1. Identify **abnormal increase** in temperature to attack a data center **during its peak hours**

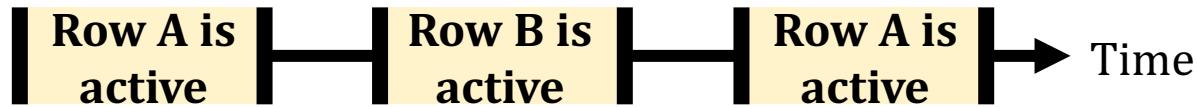


2. Precisely measure the temperature **to trigger an attack** exactly at the desired temperature



# Attack Improvement 3: Bypassing Defenses with Aggressor Row Active Time

Activating aggressor rows as frequently as possible:



Keeping the aggressor rows active for a longer time:

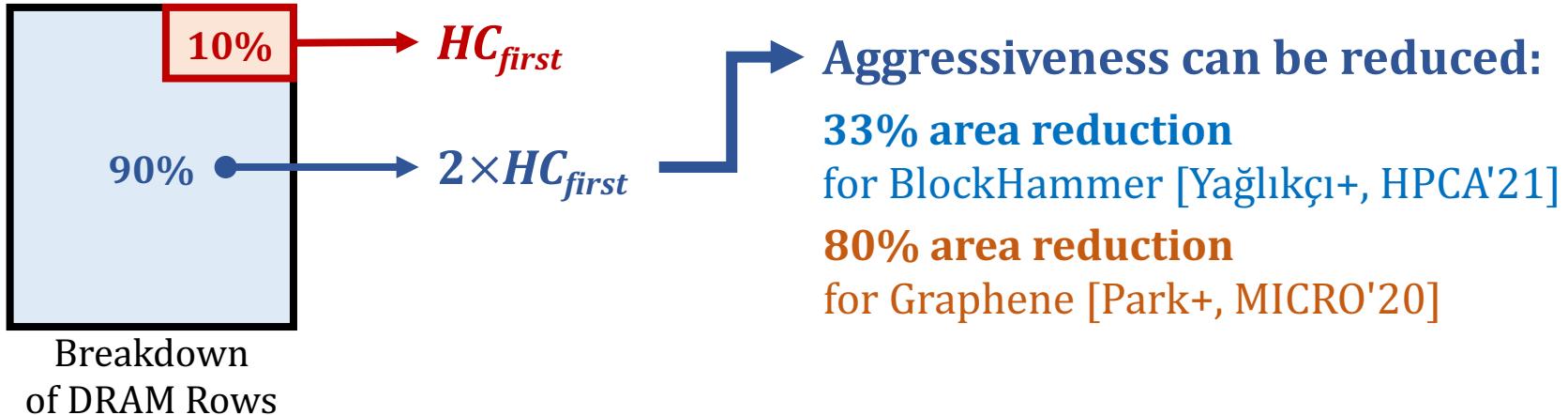


**Reduces** the minimum activation count to induce a bit flip **by 36%**

**Bypasses defenses** that do not account for this reduction

# Defense Improvements

- Example 1: Leveraging the variation across DRAM rows

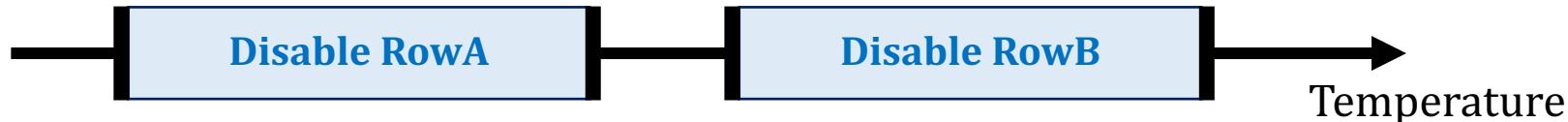


- Example 2: Leveraging the variation with temperature

- A DRAM cell experiences **bit flips** within a bounded temperature range



- A row can be **disabled** within the row's **vulnerable temperature range**



# More Defense Implications in the Paper

- Leveraging **the similarity across subarrays** in a DRAM module can **reduce the module's profiling time** for RowHammer errors
- Monitoring and limiting the **aggressor row active time** from the memory controller can **reduce the RowHammer vulnerability** and **make defenses more efficient**
- **ECC schemes** can target the **non-uniform bit error distribution** caused by **design-induced variation** across DRAM columns
- **Cooling** DRAM chips can **reduce overall bit error rate**

# More Defense Implications in the Paper

- Leveraging the similarity across subarrays in a DRAM module to speed up profiling the module for RowHammer errors

## A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa\*  
ETH Zürich

A. Giray Yağlıkçı\*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich

- Cooling DRAM chips can reduce overall bit error rate

# Outline

Motivation and Goal

Experimental Methodology

Temperature Analysis

Aggressor Row Active Time Analysis

Spatial Variation Analysis

Implications on Attacks and Defenses

Conclusions

# Conclusion

- **Motivation:**
  - Denser DRAM chips are **more vulnerable** to RowHammer
  - Understanding RowHammer enables designing **effective and efficient solutions**, but **no rigorous study** demonstrates how vulnerability varies under different conditions
- **Goal:** Provide insights into **three fundamental properties** of RowHammer that can be leveraged to design **more effective and efficient attacks and defenses**
  - 1) DRAM chip **temperature**
  - 2) The time that an **aggressor row stays active**
  - 3) Victim DRAM cell's **physical location**
- **Experimental study:** 272 DRAM chips from four major manufacturers
- **Key Results:** We provide **6 takeaways** from **16 novel observations**

A RowHammer bit flip is **more likely to occur**

  - 1) in a **bounded range of temperature**
  - 2) if the aggressor row is **active for longer time**
  - 3) in **certain physical regions** of the DRAM module under attack
- **Conclusion:** Our novel observations can inspire and aid future work
  - Craft **more effective attacks**
  - Design **more effective and efficient defenses**

# *A Deeper Look into RowHammer's Sensitivities*

*Experimental Analysis of Real DRAM Chips  
and Implications on Future Attacks and Defenses*

**Lois Orosa    Abdullah Giray Yağlıkçı**

Haocong Luo   Ataberk Olgun   Jisung Park

Hasan Hassan   Minesh Patel   Jeremie S. Kim   Onur Mutlu

**SAFARI**

**ETH** zürich



**TOBB ETÜ**  
University of Economics & Technology

# A Deeper Look into RowHammer

---

Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,

"A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses" in *MICRO*, 2021.

[Slides (pptx) (pdf)]

[Short Talk Slides (pptx) (pdf)]

[Lightning Talk Slides (pptx) (pdf)]

[Talk Video (21 minutes)]

[Lightning Talk Video (1.5 minutes)]

[arXiv version]

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- **Understanding Read Disturbance in DRAM Chips**
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - **RowPress [Luo+ ISCA'23]**
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# 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," in ISCA, 2023.**

[Extended arxiv version]

[Slides (pptx) (pdf)] [Talk Video (14 minutes, including Q&A)]

[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.*

*Distinguished 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*

**ISCA 2023 Distinguished Artifact Award**

**SAFARI**

**ETH** Zürich

# High-Level Summary

- We demonstrate and analyze **RowPress, a new read disturbance phenomenon** that causes bitflips in real DRAM chips
- We show that RowPress is **different from the RowHammer vulnerability**
- We demonstrate RowPress **using a user-level program** on a real Intel system with real DRAM chips
- We provide **effective solutions** to RowPress

# Outline

## DRAM Background

### What is RowPress?

### Real DRAM Chip Characterization

#### Characterization Methodology

#### Key Characteristics of RowPress

### Real-System Demonstration

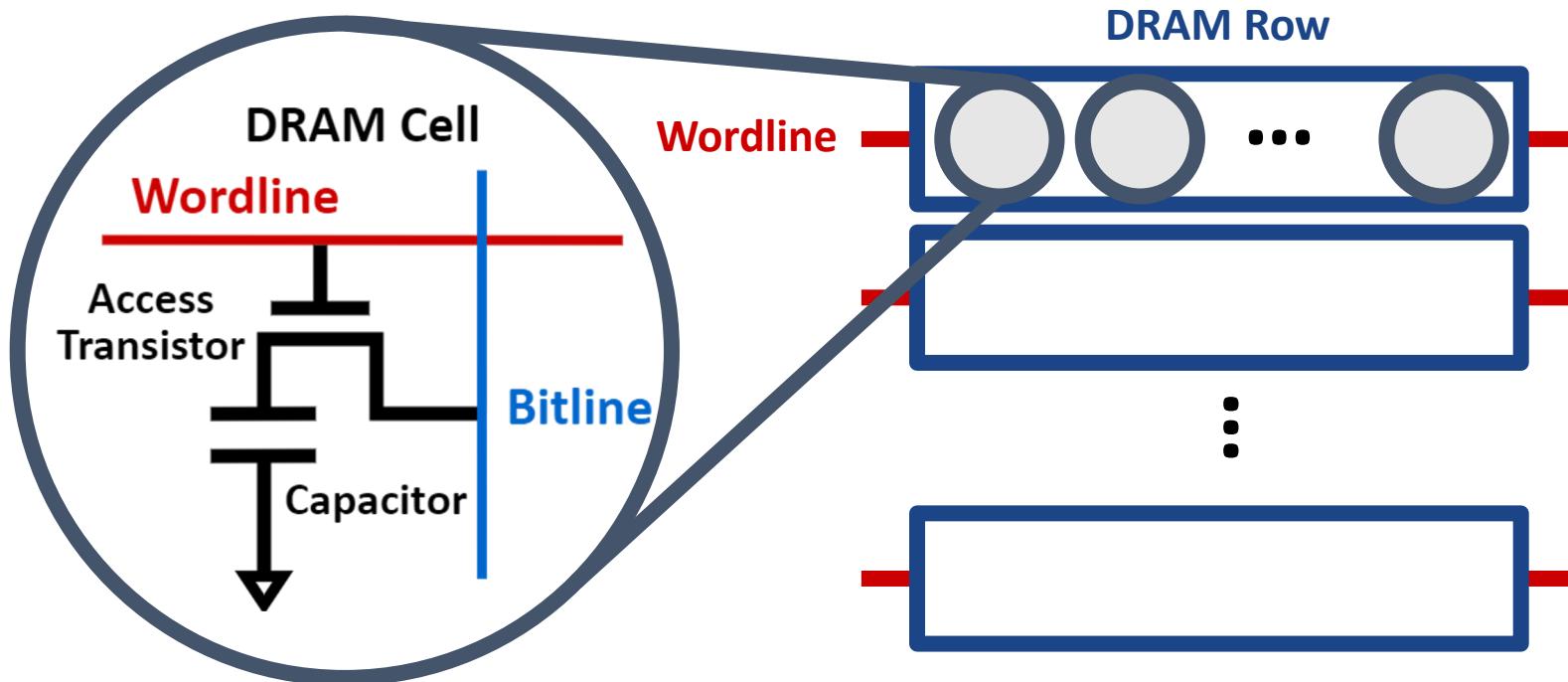
### Mitigating RowPress

### Conclusion

# DRAM Organization

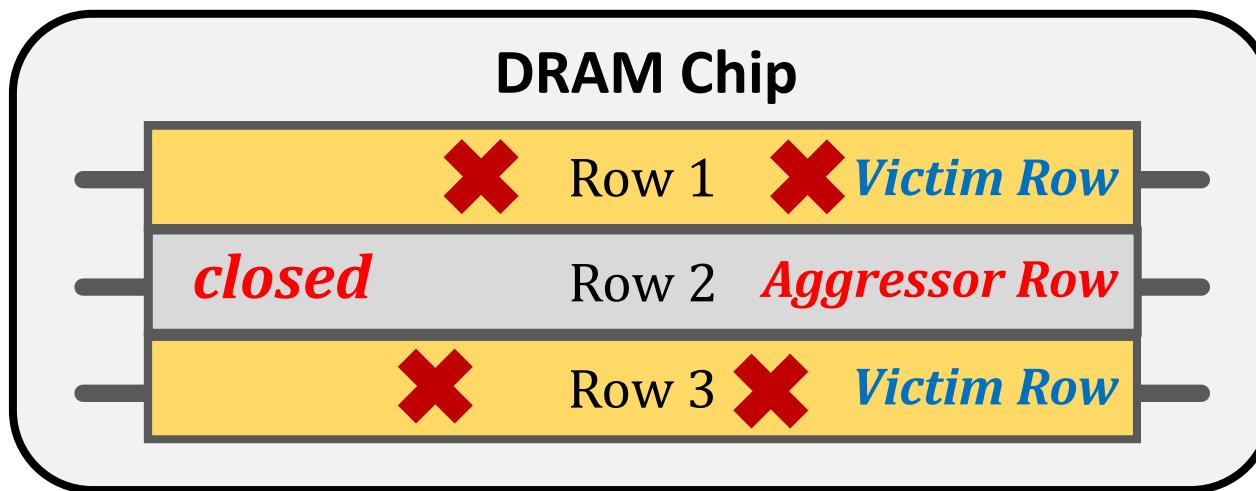
DRAM is the prevalent technology for main memory

- A **DRAM cell** stores 1 bit of information in a **leaky** capacitor
- DRAM cells are organized into **DRAM rows**



# Read Disturbance in DRAM

- 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

# Are There Other Read-Disturb Issues in DRAM?

- RowHammer is the only studied read-disturb phenomenon
- Mitigations work by detecting **high row activation count**

What if there is another read-disturb phenomenon  
that **does NOT rely on high row activation count**?



[https://www.reddit.com/r/CrappyDesign/comments/arw0q8/now\\_this\\_this\\_is\\_poor\\_fencing/](https://www.reddit.com/r/CrappyDesign/comments/arw0q8/now_this_this_is_poor_fencing/)

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

Conclusion

# What is 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!

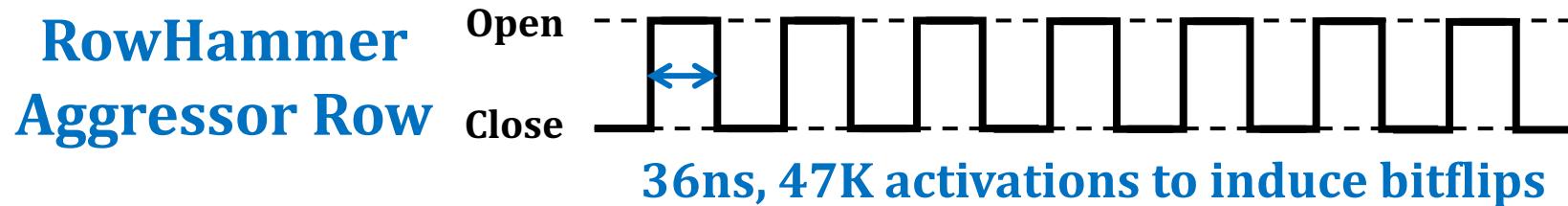


Now, let's see how this is **different from RowHammer**

# RowPress vs. RowHammer

Instead of using a high activation count,

- ☛ increase the time that the aggressor row stays open



We observe bitflips even with **ONLY ONE activation** in extreme cases where the row stays open for 30ms

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

Conclusion

# Major Takeaways from Real DRAM Chips

RowPress significantly **amplifies** DRAM's vulnerability to **read disturbance**

RowPress has a **different** underlying failure **mechanism** from RowHammer

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

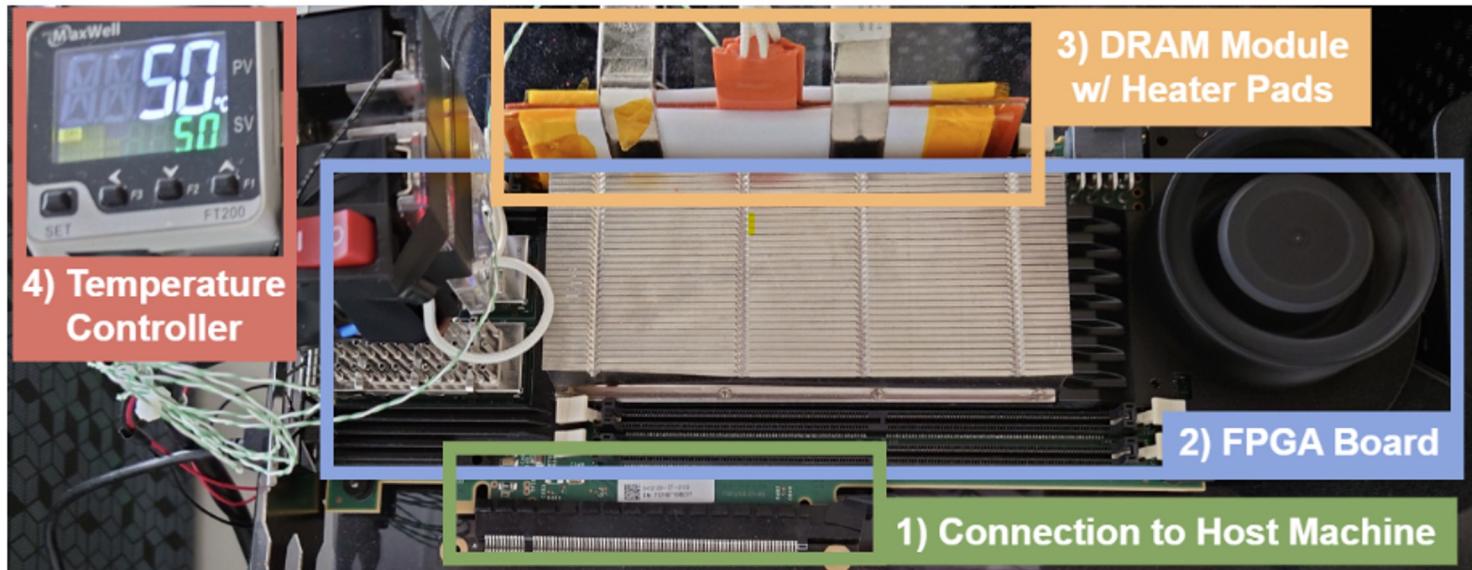
Mitigating RowPress

Conclusion

# Characterization Methodology (I)

## FPGA-based DDR4 testing infrastructure

- Developed from **SoftMC** [Hassan+, HPCA'17] and **DRAM Bender** [Olgun+, TCAD'23]
- Fine-grained control over DRAM commands, timings, and temperature



# Characterization Methodology (II)

## DRAM chips tested

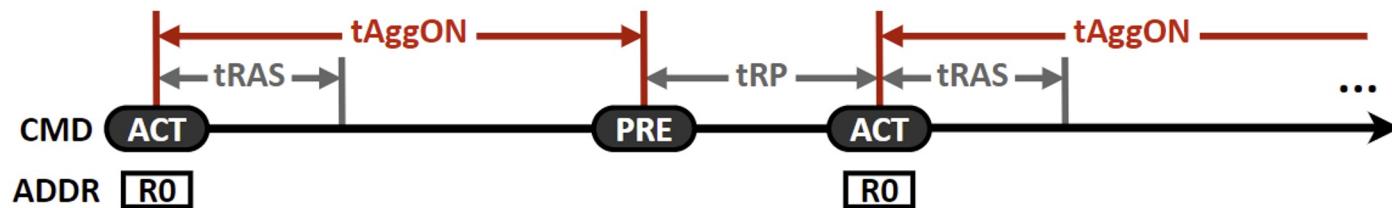
- 164 DDR4 chips from all 3 major DRAM manufacturers
- Covers different die densities and revisions

Mfr.	#DIMMs	#Chips	Density	Die Rev.	Org.	Date
Mfr. S (Samsung)	2	8	8Gb	B	x8	20-53
	1	8	8Gb	C	x8	N/A
	3	8	8Gb	D	x8	21-10
	2	8	4Gb	F	x8	N/A
Mfr. H (SK Hynix)	1	8	4Gb	A	x8	19-46
	1	8	4Gb	X	x8	N/A
	2	8	16Gb	A	x8	20-51
	2	8	16Gb	C	x8	21-36
Mfr. M (Micron)	1	16	8Gb	B	x4	N/A
	2	4	16Gb	B	x16	21-26
	1	16	16Gb	E	x4	20-14
	2	4	16Gb	E	x16	20-46
	1	4	16Gb	F	x16	21-50

# Characterization Methodology (III)

**Metric:** The minimum number of aggressor row activations in total to cause at least one bitflip (ACmin)

**Access Pattern:** Single-sided (i.e., only one aggressor row). Sweep aggressor row on time (**tAggON**) from 36ns to 30ms



**Data Pattern:** Checkerboard (0xAA in aggressor and 0x55 in victim)

**Temperature:** 50°C

**Algorithm:** Bisection-based ACmin search

- Each search iteration is capped at 60ms (<64ms refresh window)
- Repeat 5 times and report the minimum ACmin value observed
- Sample 3072 DRAM rows per chip

[More sensitivity studies in the paper]

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

Conclusion

# Major Takeaways from Real DRAM Chips

RowPress significantly **amplifies** DRAM's vulnerability to **read disturbance**

RowPress has a **different** underlying failure **mechanism** from RowHammer

# Key Characteristics of RowPress

## Amplifying read disturbance in DRAM

- Reduces the minimum number of row activations needed to induce a bitflip ( $AC_{min}$ ) by **1-2 orders of magnitude**
- In extreme cases, activating a row **only once** induces bitflips
- Gets worse as **temperature increases**

## Different from RowHammer

- Affects a **different set of cells** compared to RowHammer and retention failures
- **Behaves differently** as access pattern or temperature changes compared to RowHammer

# Key Characteristics of RowPress

## Amplifying read disturbance in DRAM

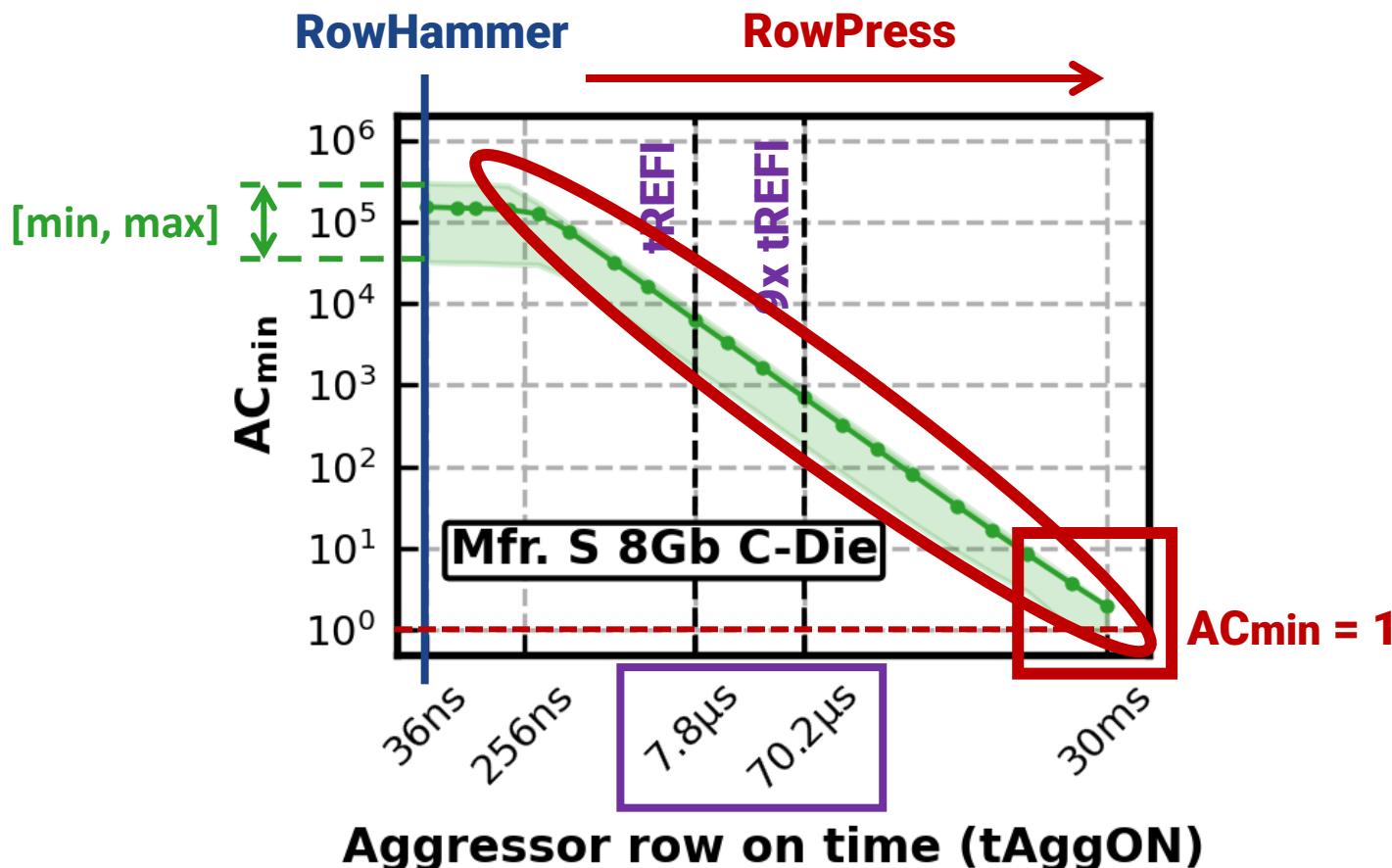
- Reduces the minimum number of row activations needed to induce a bitflip ( $AC_{min}$ ) by **1-2 orders of magnitude**
- In extreme cases, activating a row **only once** induces bitflips
- Gets worse as **temperature increases**

## Different from RowHammer

- Affects a **different set of cells** compared to RowHammer and retention failures
- **Behaves differently** as access pattern or temperature changes compared to RowHammer

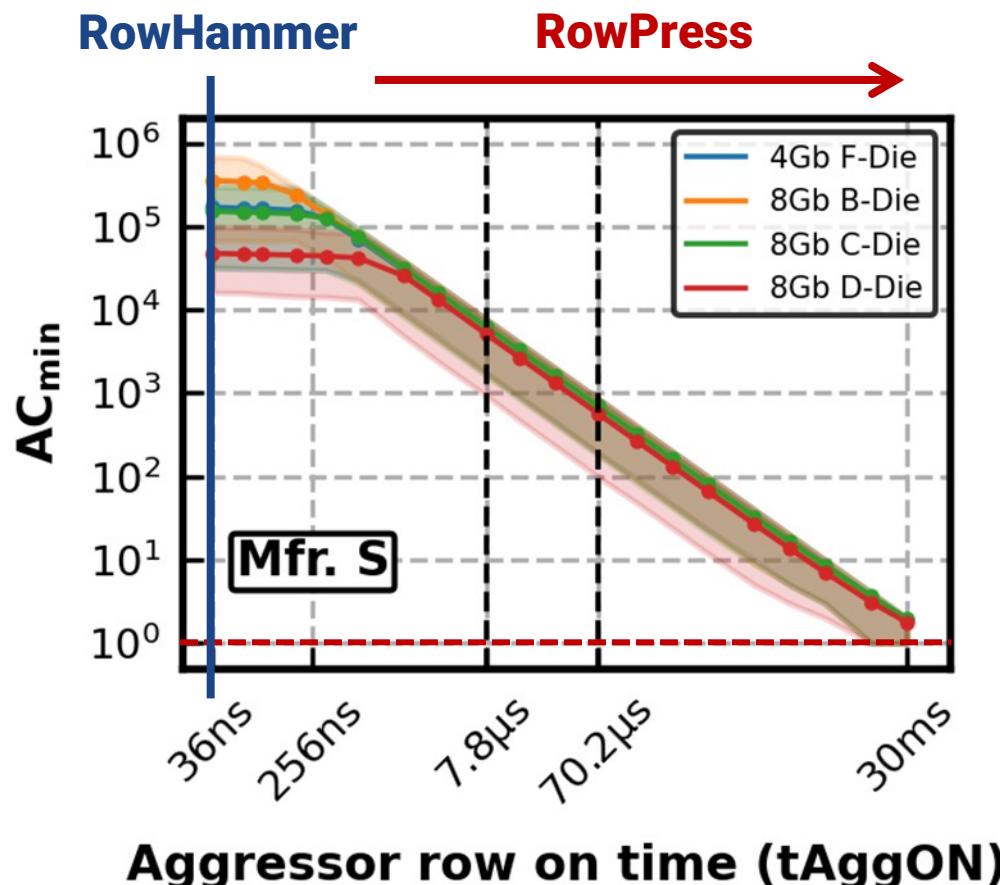
# Amplifying Read Disturbance (I)

How minimum activation count to induce a bitflip ( $AC_{min}$ ) changes as aggressor row on time ( $t_{AggON}$ ) increases



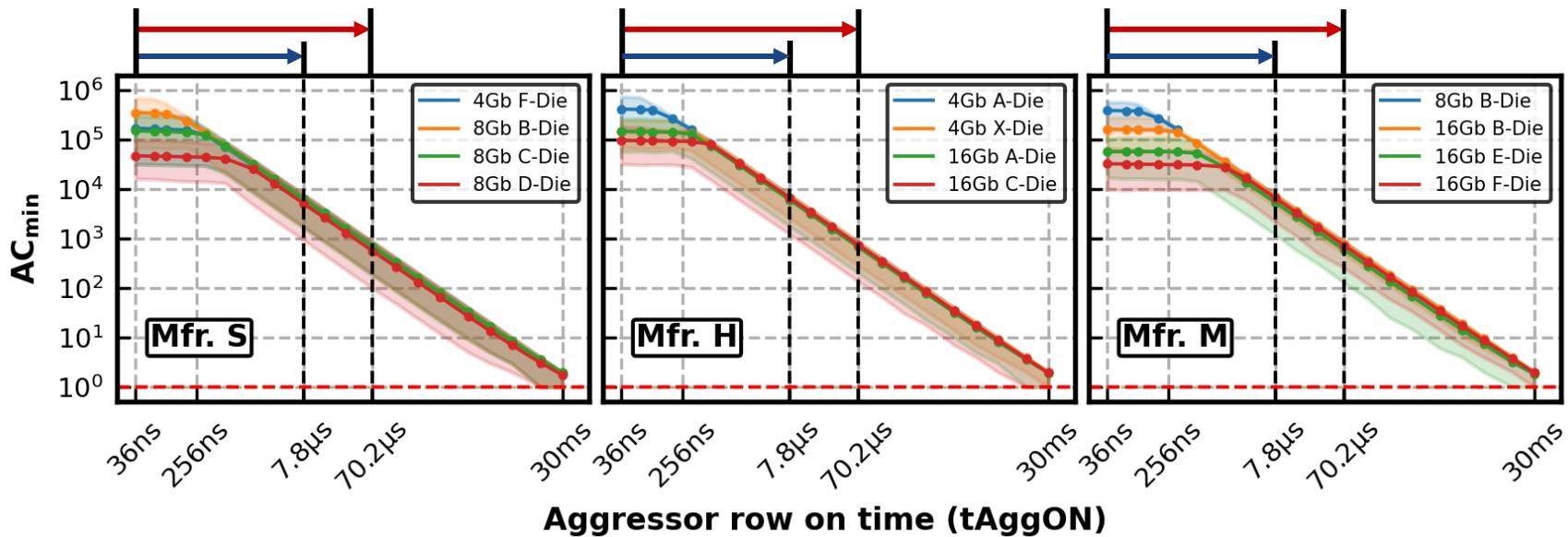
# Amplifying Read Disturbance (II)

How minimum activation count to induce a bitflip ( $AC_{min}$ ) changes as aggressor row on time ( $t_{AggON}$ ) increases



# Amplifying Read Disturbance (III)

How minimum activation count to induce a bitflip ( $AC_{min}$ ) changes as aggressor row on time ( $t_{AggON}$ ) increases



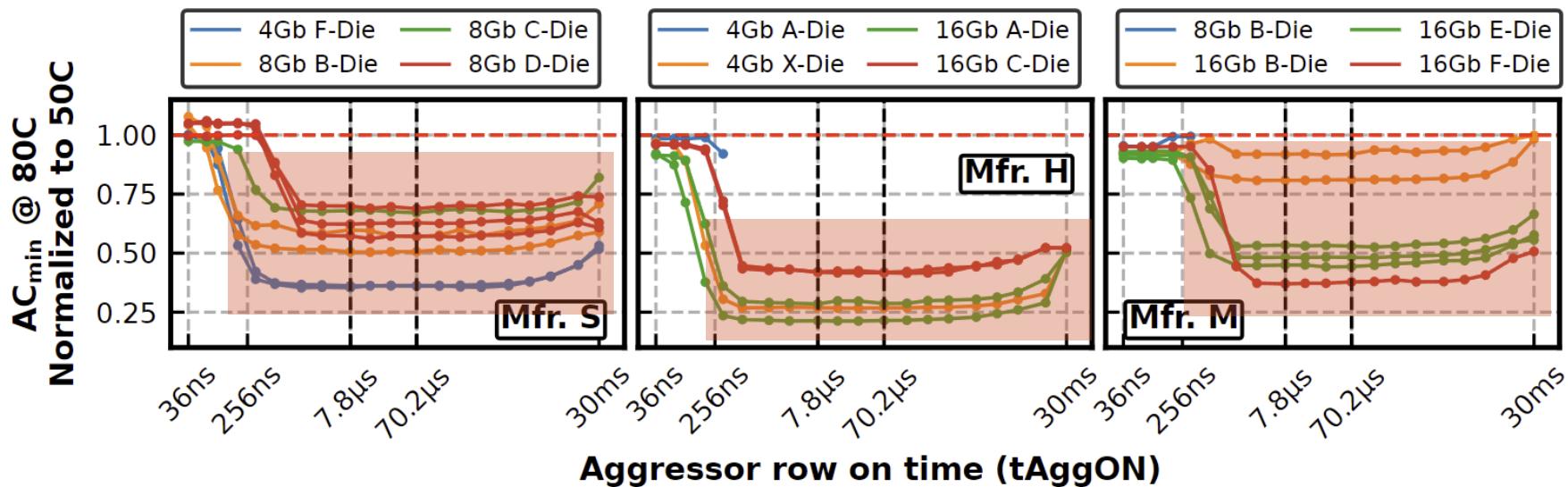
$AC_{min}$  reduces by **21X** on average when  $t_{AggON}$  increases from 36ns to 7.8 $\mu$ s  
**191X** **70.2 $\mu$ s**

RowPress significantly reduces  $AC_{min}$  as  $t_{AggON}$  increases

# Amplifying Read Disturbance (IV)

## AC<sub>min</sub> @ 80°C normalized to AC<sub>min</sub> @ 50°C

- Data point below 1 means fewer activations to cause bitflips @ 80°C compared to 50°C



When tAggON is 7.8 μs, RowPress requires about 50% fewer activations to induce bitflips at 80°C compared to 50°C

RowPress gets worse as temperature increases

# Key Characteristics of RowPress

## Amplifying read disturbance in DRAM

- Reduces the minimum number of row activations needed to induce a bitflip ( $AC_{min}$ ) by **1-2 orders of magnitude**
- In extreme cases, activating a row **only once** induces bitflips
- Gets worse as **temperature increases**

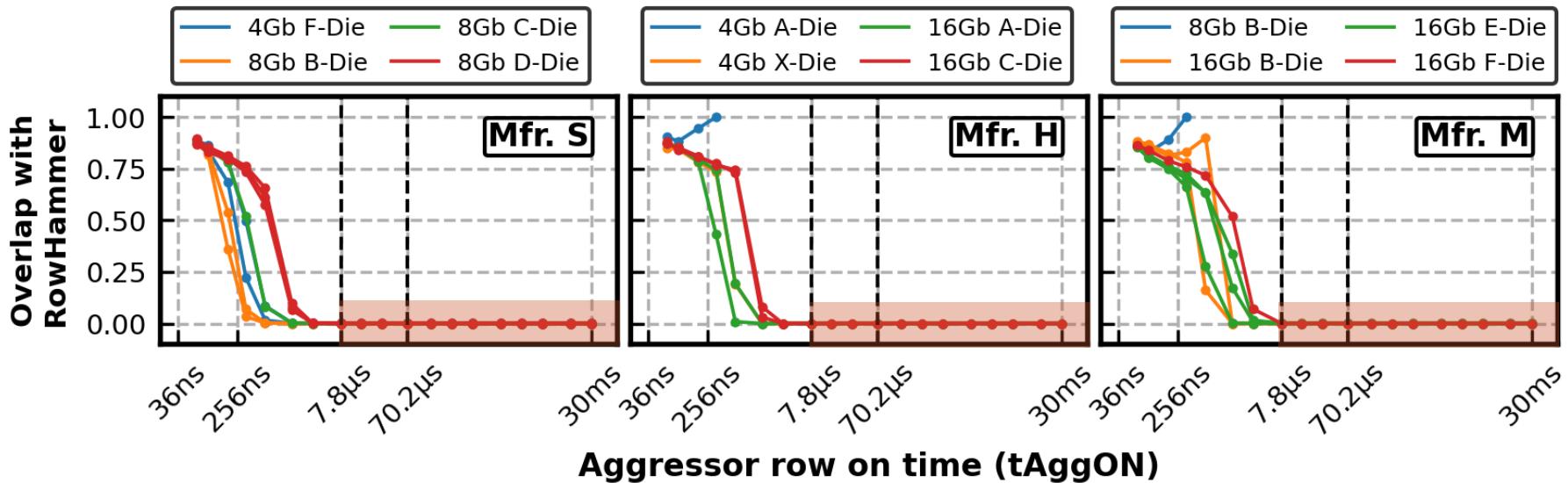
## Different from RowHammer

- Affects a **different set of cells** compared to RowHammer and retention failures
- **Behaves differently** as access pattern or temperature changes compared to RowHammer

# Difference Between RowPress and RowHammer (I)

## Cells vulnerable to RowPress vs. RowHammer

- Cells vulnerable to RowPress (RowHammer) are those that flip @ ACmin
- Overlap = 
$$\frac{\text{Number of Cells Vulnerable to Both RowPress and RowHammer}}{\text{Number of Cells Vulnerable to RowPress}}$$

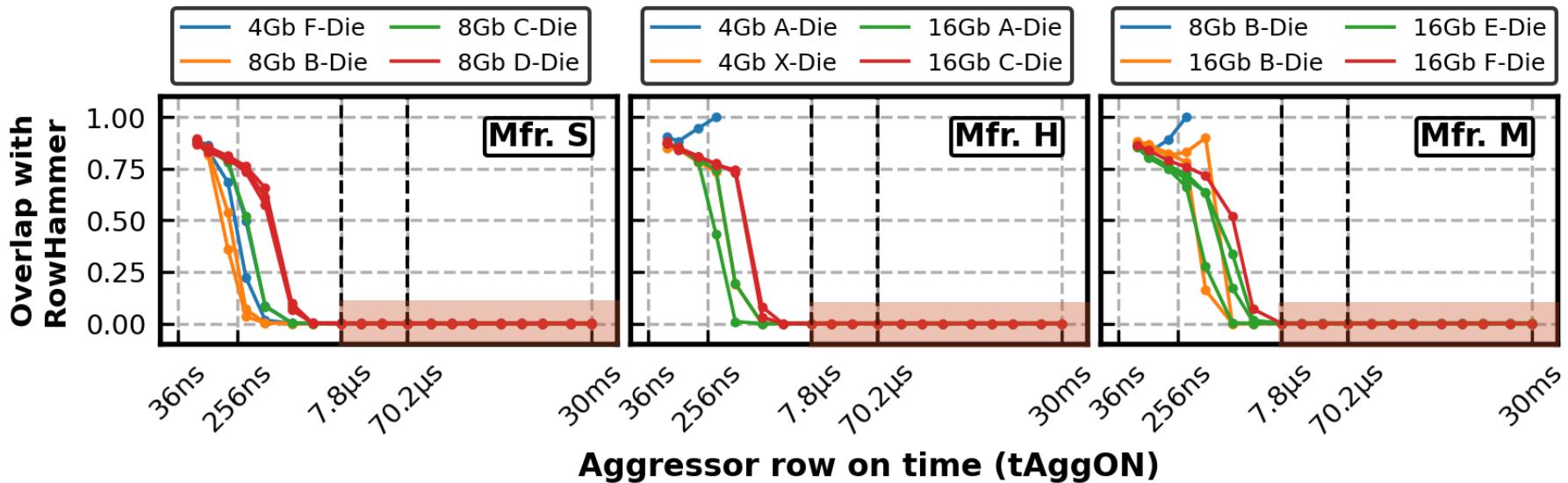


On average, only **0.013%** of DRAM cells vulnerable to RowPress  
are also vulnerable to RowHammer, when **tAggON  $\geq$  7.8us**

# Difference Between RowPress and RowHammer (II)

## Cells vulnerable to RowPress vs. RowHammer

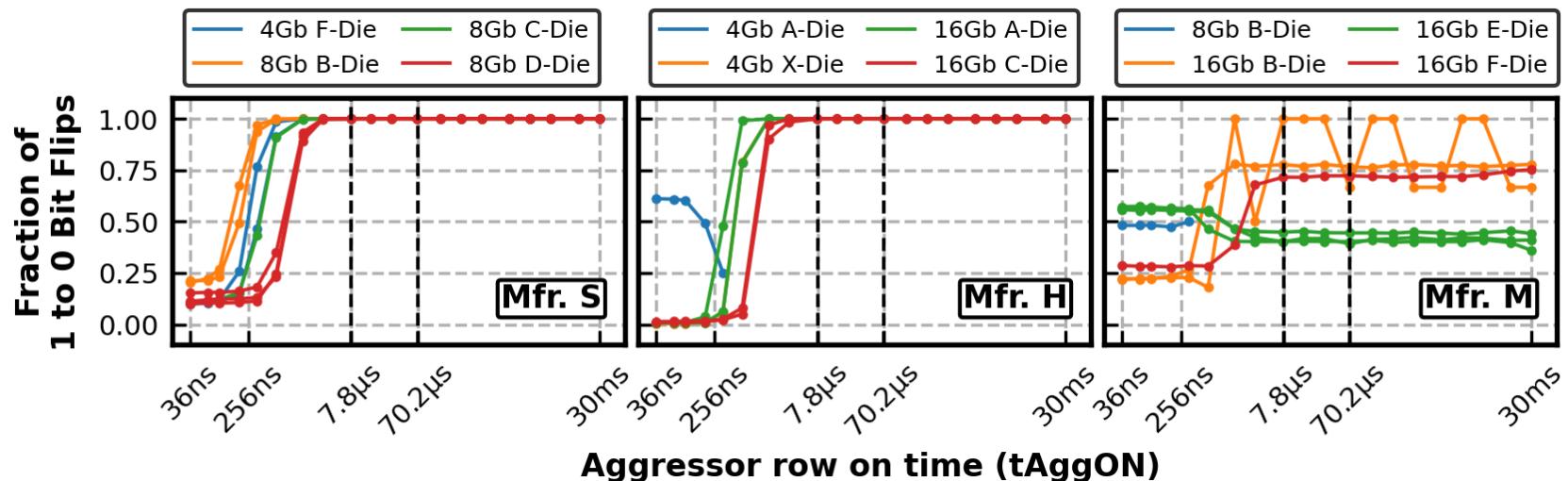
- Cells vulnerable to RowPress (RowHammer) are those that flip @ ACmin
- Overlap = 
$$\frac{\text{Number of Cells Vulnerable to Both RowPress and RowHammer}}{\text{Number of Cells Vulnerable to RowPress}}$$



**Most cells vulnerable to RowPress  
are NOT vulnerable to RowHammer**

# Difference Between RowPress and RowHammer (III)

## Directionality of RowHammer and RowPress bitflips



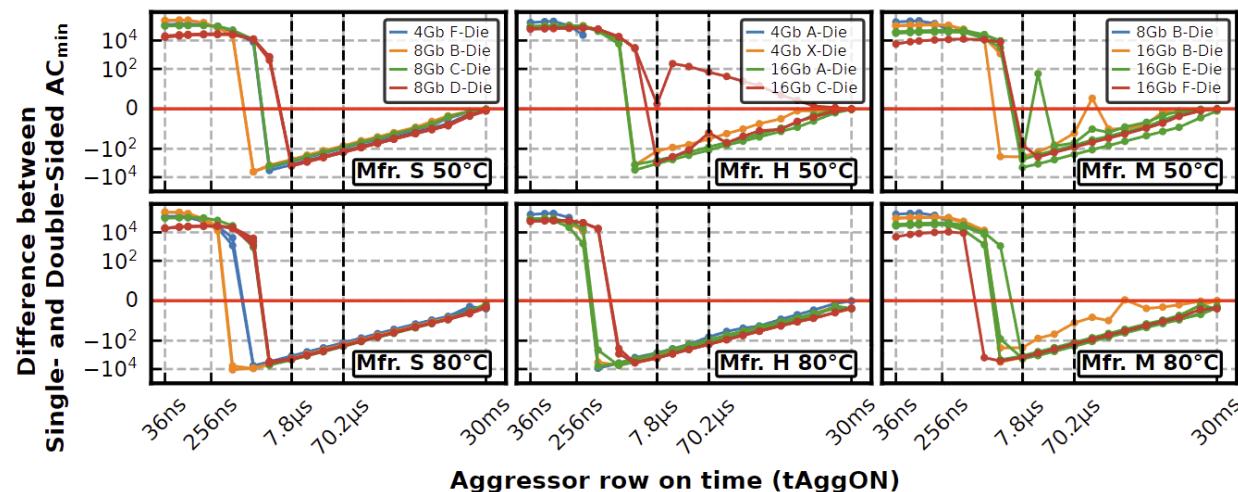
The majority of RowHammer bitflips are 0 to 1  
The majority of RowPress bitflips are 1 to 0

RowPress and RowHammer bitflips have opposite directions

# Difference Between RowPress and RowHammer (IV)

## Effectiveness of single-sided vs. double-sided RowPress

- Data point below 0 means fewer activations to cause bitflips with single-sided RowPress compared to double-sided RowPress



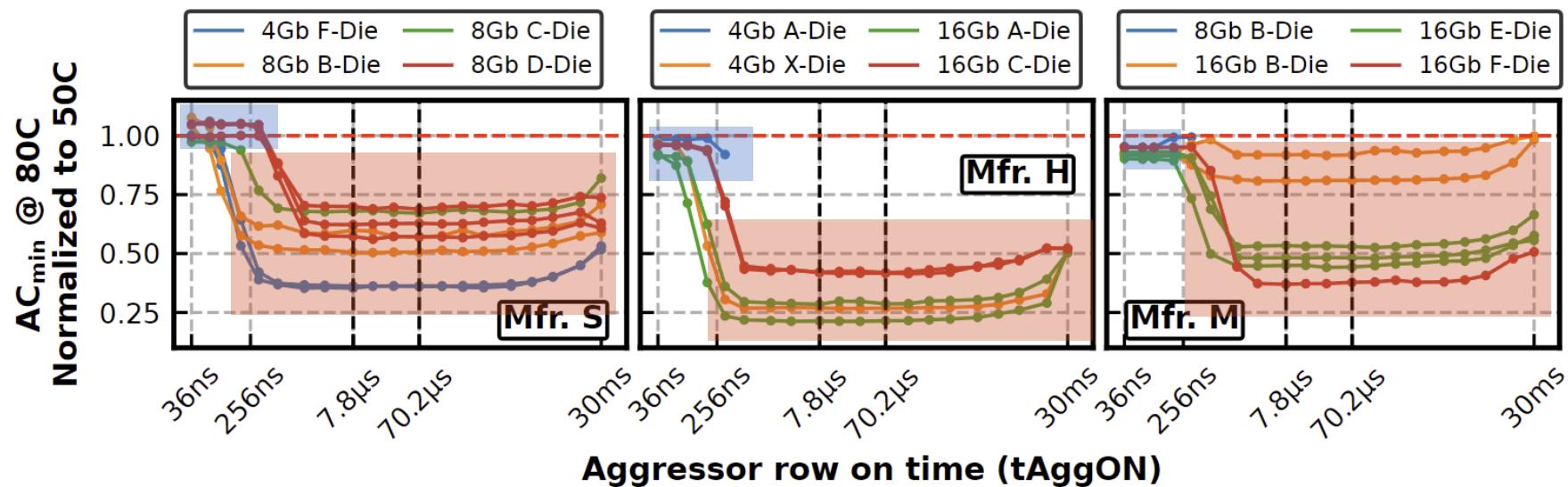
As tAggON increases beyond a certain level, **single-sided RowPress becomes more effective** compared to double-sided

Different from RowHammer where **double-sided is more effective**

# Difference Between RowPress and RowHammer (V)

## Sensitivity to temperature

- Data point below 1 means fewer activations to cause bitflips @ 80°C compared to 50°C



RowPress gets worse as temperature increases,  
which is very different from RowHammer

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

Conclusion

# Real-System Demonstration (I)



Intel Core i5-10400  
(Comet Lake)



Samsung DDR4 Module  
M378A2K43CB1-CTD  
(Date Code: 20-10)  
w/ TRR RowHammer Mitigation

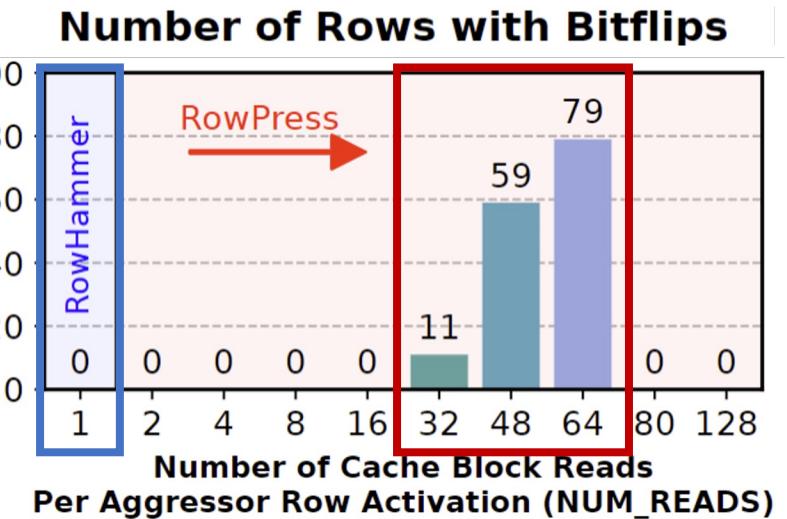
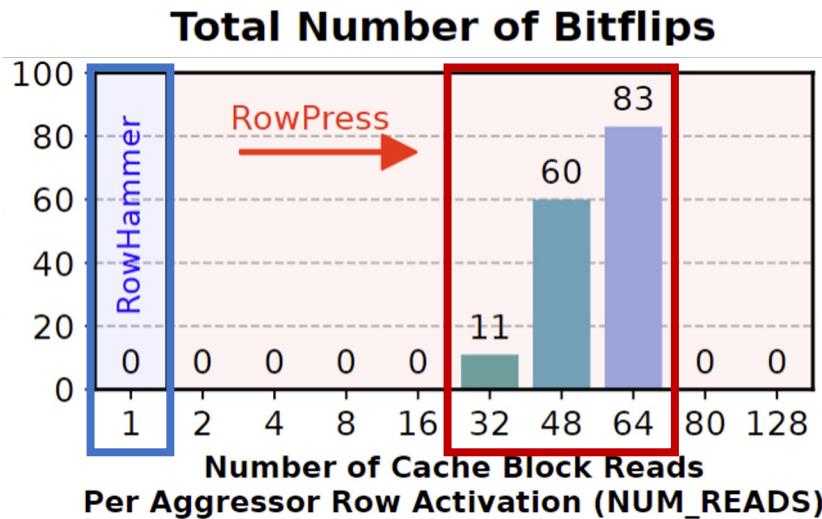
**Key Idea:** A proof-of-concept RowPress program keeps a DRAM row open for a longer period by **keeping on accessing different cache blocks in the row**

```
// 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

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

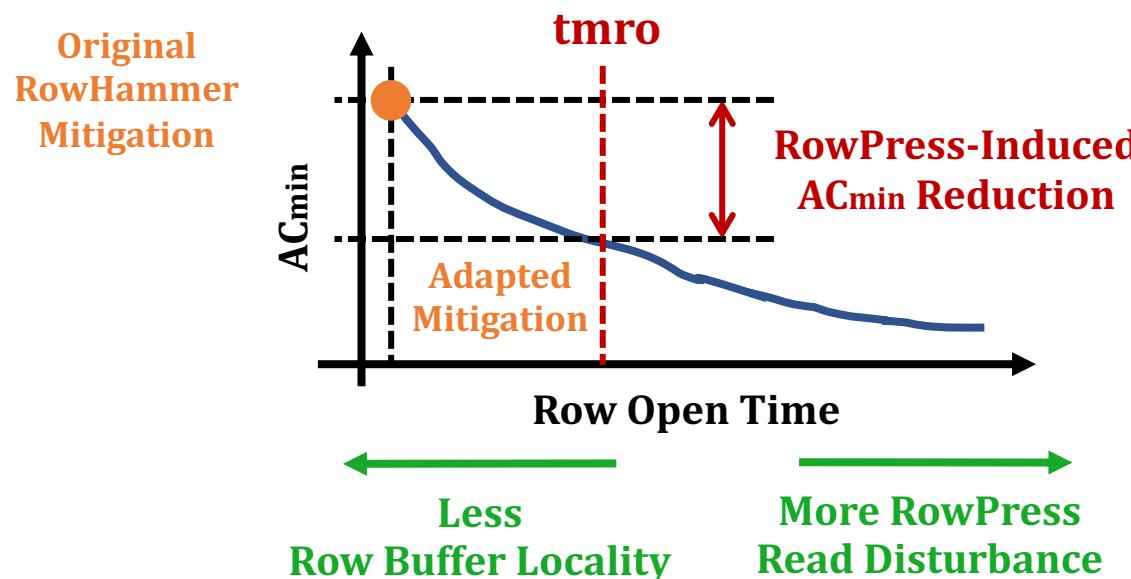
Conclusion

# Mitigating RowPress (I)

We propose a methodology to adapt existing RowHammer mitigations to **also mitigate RowPress**

## Key Idea:

1. Limit the maximum row open time (**tmro**)
2. Configure the RowHammer mitigation to account for the **RowPress-induced reduction in ACmin**



# Mitigating RowPress (II)

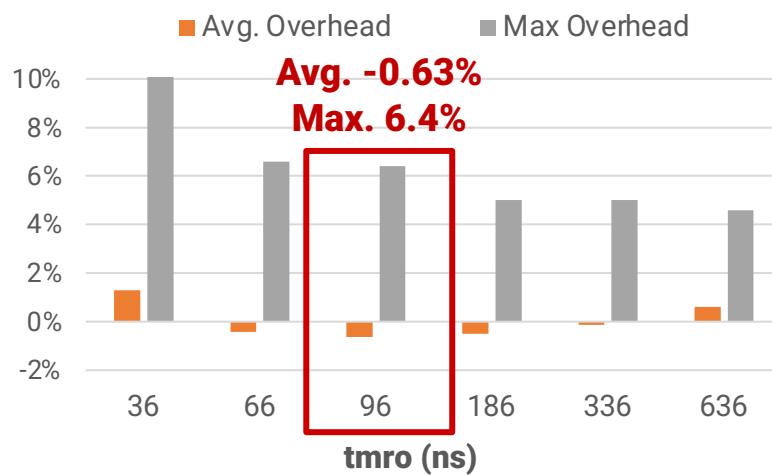
## Evaluation methodology

- **Adapted RowHammer Mitigations:** Graphene (**Graphene-RP**) and PARA (**PARA-RP**)
- Cycle-accurate DRAM simulator: Ramulator [Kim+, CAL'15]
  - 4 GHz Out-of-Order Core, dual-rank DDR4 DRAM
  - FR-FCFS scheduling
  - Open-row policy (with limited maximum row open time)
- 58 four-core multiprogrammed workloads from SPEC CPU2017, TPC-H, and YCSB
- **Metric: Additional performance overhead** of Graphene-RP (PARA-RP) over Graphene (PARA)
  - Measured by weighted speedup

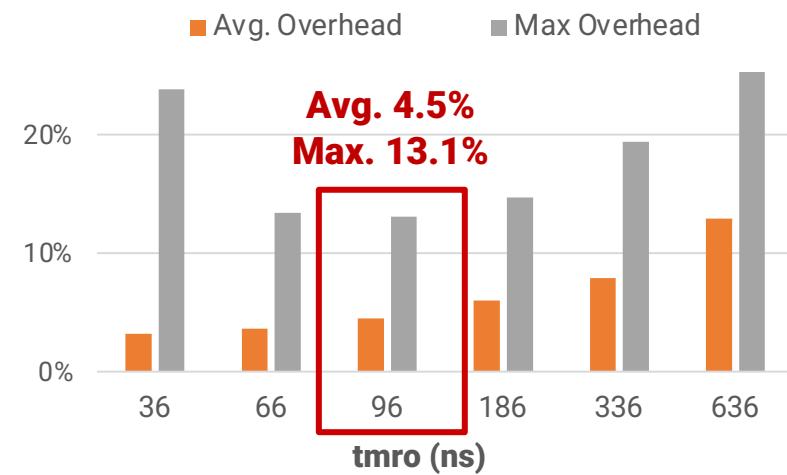
# Mitigating RowPress (III)

## Key evaluation results

**Additional Performance Overhead of Graphene-RP**



**Additional Performance Overhead of PARA-RP**



Our solutions **mitigate RowPress**  
at low additional performance overhead

# Outline

DRAM Background

What is RowPress?

Real DRAM Chip Characterization

Characterization Methodology

Key Characteristics of RowPress

Real-System Demonstration

Mitigating RowPress

Conclusion

# Conclusion

We demonstrate and analyze **RowPress, a widespread read disturbance phenomenon** that causes bitflips in real DRAM chips

We **characterize RowPress** on 164 DDR4 chips from all 3 major DRAM manufacturers

- RowPress greatly **amplifies read disturbance**: minimum activation count **reduces by 1-2 orders of magnitude**
- RowPress has a **different mechanism** from RowHammer & retention failures

We **demonstrate RowPress** using a user-level program

- Induces bitflips when RowHammer cannot

We provide **effective solutions** to RowPress

- Low additional performance overhead

# More Results & Source Code

Many more results & analyses in the paper

- 6 major takeaways
- 19 major empirical observations
- 3 more potential mitigations



Fully open source and artifact evaluated

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



ISCA 2023 Distinguished Artifact Award



# 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*

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

**SAFARI**

**ETH** Zürich

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," in ISCA, 2023.**

[Extended arxiv version]

[Slides (pptx) (pdf)] [Talk Video (14 minutes, including Q&A)]

[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.*

*Distinguished artifact award at ISCA 2023.*

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- **Understanding Read Disturbance in DRAM Chips**
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - **HBM RowHammer [Olgun+ DSN Disrupt'23]**
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# Read Disturbance in HBM Chips

---

Ataberk Olgun, Majd Osserian, A. Giray Yağlıkçı, Yahya Can Tugrul, Haocong Luo, Steve Rhyner, Behzad Salami, Juan Gomez-Luna, and Onur Mutlu,

**"An Experimental Analysis of RowHammer in HBM2 DRAM Chips"**

*Proceedings of the 53nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Disrupt Track (**DSN Disrupt**), Porto, Portugal, June 2023.*

[arXiv version]

[Slides (pptx) (pdf)]

[Talk Video (24 minutes, including Q&A)]

# An Experimental Analysis of RowHammer in HBM2 DRAM Chips

Ataberk Olgun Majd Osseiran

A. Giray Yağlıkçı Yahya Can Tuğrul Haocong Luo Steve Rhyner  
Behzad Salami Juan Gomez Luna Onur Mutlu

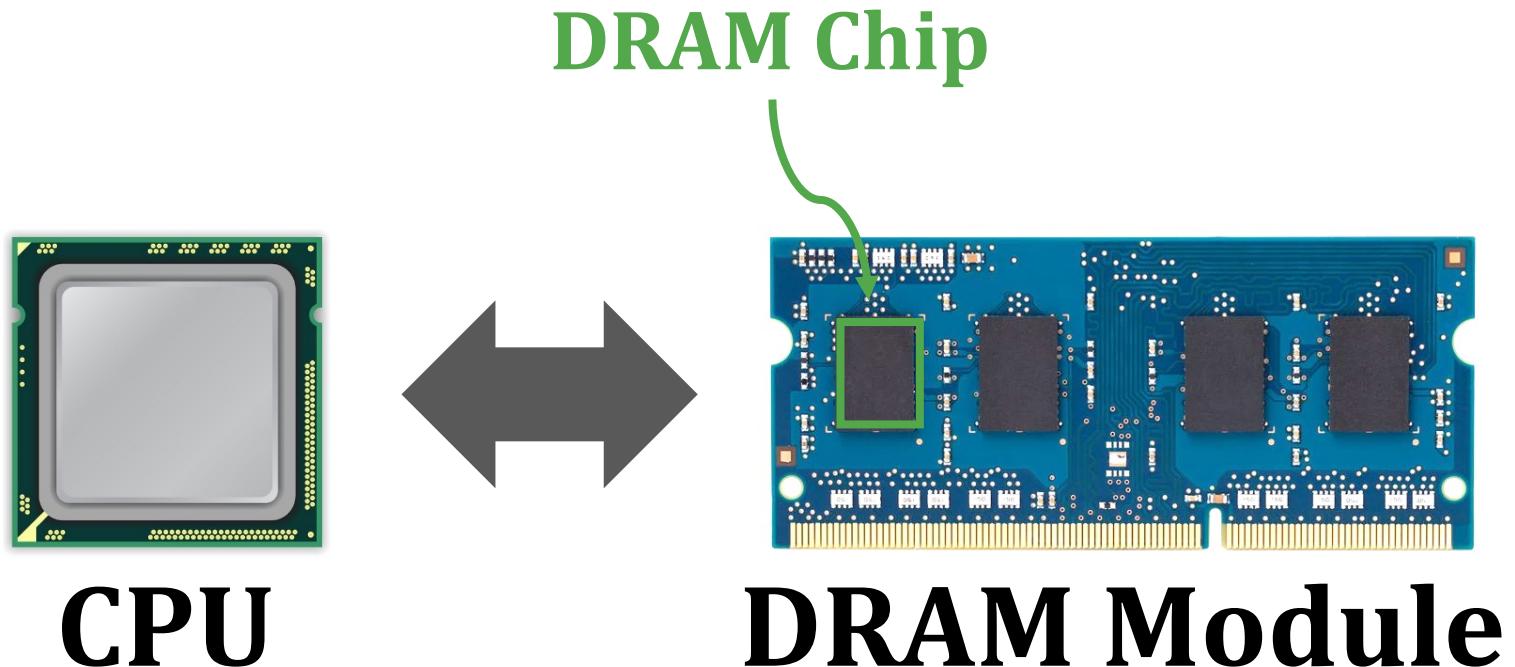
**SAFARI**

**ETH** zürich

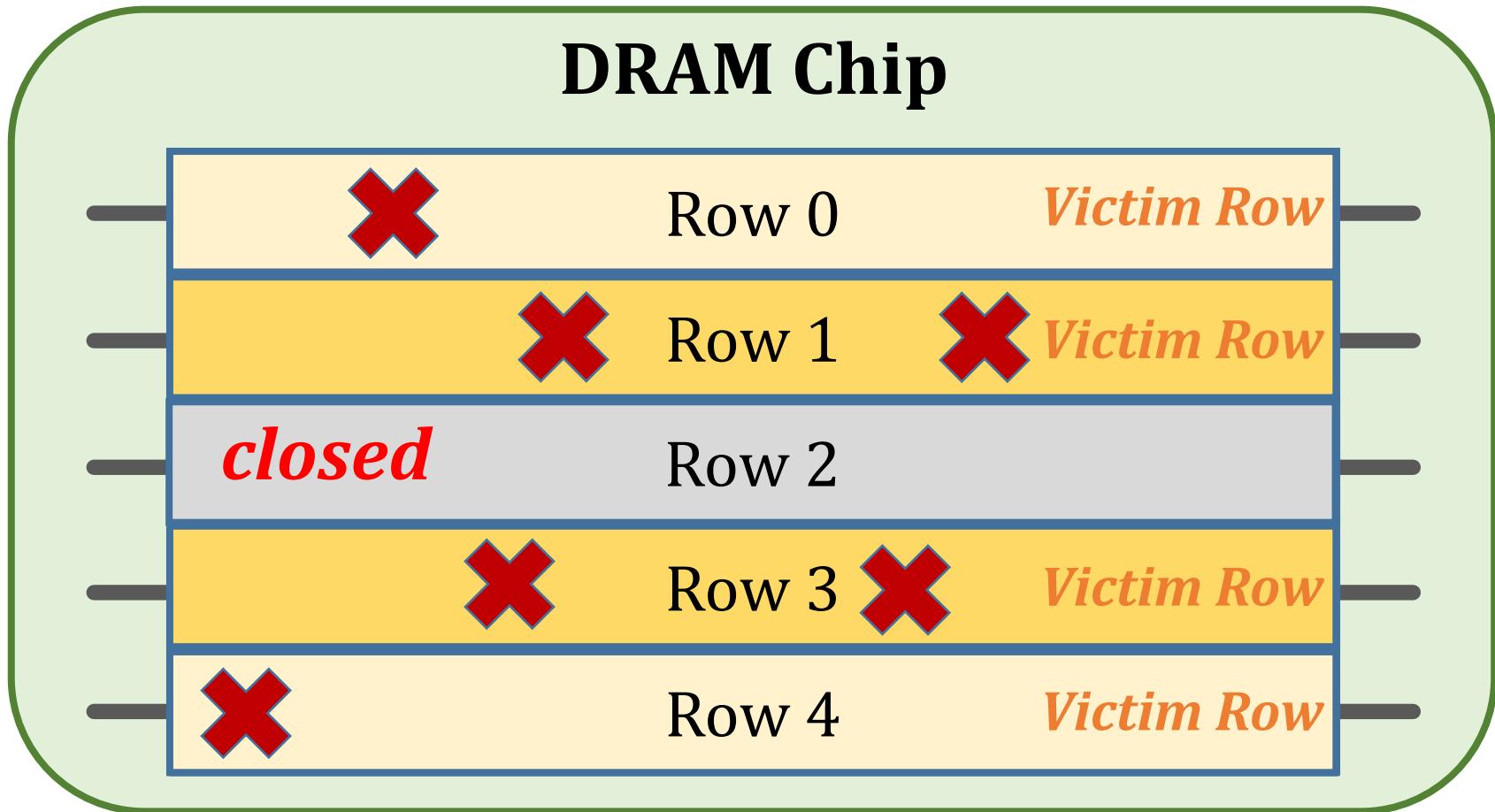


AMERICAN  
UNIVERSITY  
OF BEIRUT

# The RowHammer Vulnerability (I)

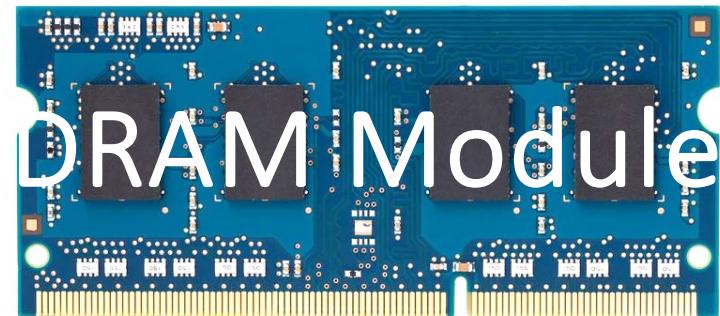
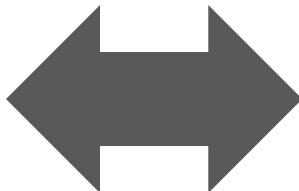
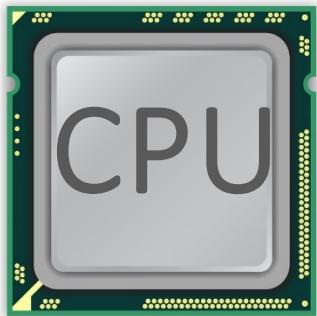


# The RowHammer Vulnerability (II)

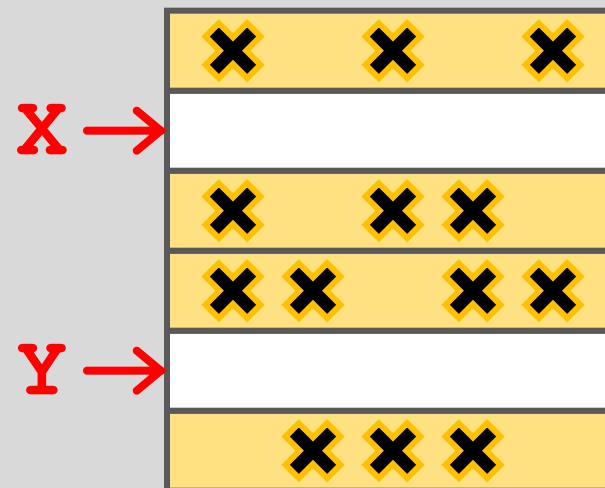


Repeatedly **opening** (activating) and **closing** (precharging)  
a DRAM row causes **RowHammer bit flips** in nearby rows

# A Simple Program Can Induce Bitflips



```
loop:  
    mov (%X), %eax  
    mov (%Y), %ebx  
    clflush (%X)  
    clflush (%Y)  
    mfence  
    jmp loop
```



# One Can Take Over a System

## Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

*Abstract.* **Memory isolation** is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

[Flipping Bits in Memory Without Accessing Them:  
An Experimental Study of DRAM Disturbance Errors](#)  
(Kim et al., ISCA 2014)

## Project Zero

News and updates from the Project Zero team at Google

Monday, March 9, 2015

[Exploiting the DRAM rowhammer bug to  
gain kernel privileges](#) (Seaborn, 2015)

Exploiting the DRAM rowhammer bug to gain kernel privileges

# Most DRAM Modules Are Vulnerable (2020)

DRAM type-node	Number of Chips (Modules) Tested			
	Mfr. A	Mfr. B	Mfr. C	Total
DDR3-old	56 (10)	88 (11)	28 (7)	<b>172 (28)</b>
DDR3-new	80 (10)	52 (9)	104 (13)	<b>236 (32)</b>
DDR4-old	112 (16)	24 (3)	128 (18)	<b>264 (37)</b>
DDR4-new	264 (43)	16 (2)	108 (28)	<b>388 (73)</b>
LPDDR4-1x	12 (3)	180 (45)	N/A	<b>192 (48)</b>
LPDDR4-1y	184 (46)	N/A	144 (36)	<b>328 (82)</b>

All tested DRAM types are **susceptible** to RowHammer bitflips

## What about High Bandwidth Memory (HBM)?

# Executive Summary

**Motivation:** HBM chips have new architectural characteristics (e.g., 3D-stacked dies) that might affect the RowHammer vulnerability in various ways

Understanding RowHammer enables designing effective and efficient solutions

**Problem:** No prior study demonstrates the RowHammer vulnerability in HBM

**Goal:** Experimentally analyze how vulnerable HBM DRAM chips are to RowHammer

**Experimental Study:** Detailed experimental characterization of RowHammer in a modern HBM2 DRAM chip. Our study provides two main findings:

## 1. Spatial variation of RowHammer vulnerability

- Different channels in a 3D-stacked HBM chip exhibit different RowHammer vulnerability
- DRAM rows near the end of a DRAM bank are more RowHammer resilient

## 2. On-DRAM-die RowHammer mitigations

- A modern HBM chip implements undisclosed on-DRAM-die RowHammer mitigation
- The mitigation refreshes a victim row after every 17 periodic refresh operations (e.g., similar to DDR4 chips)

# Outline

1. HBM DRAM Organization & Operation
2. DRAM Cell Leakage & RowHammer
3. HBM DRAM Testing Methodology
4. RowHammer Spatial Variation Analysis
5. On-die RowHammer Mitigation Analysis
6. Conclusion

# Outline

---

1. HBM DRAM Organization & Operation

2. DRAM Cell Leakage & RowHammer

3. HBM DRAM Testing Methodology

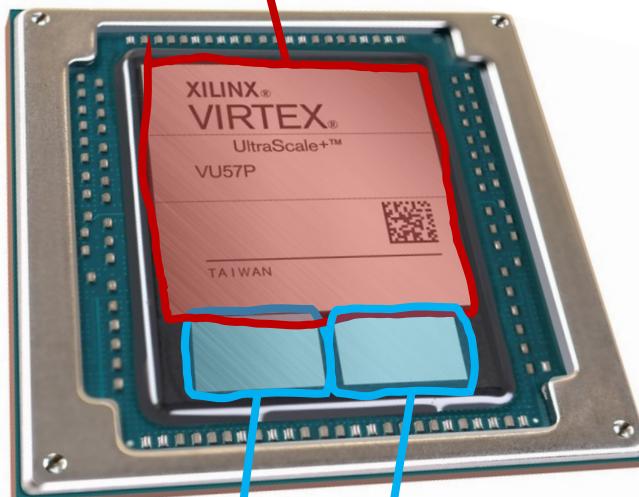
4. RowHammer Spatial Variation Analysis

5. On-die RowHammer Mitigation Analysis

6. Conclusion

# System with High Bandwidth Memory

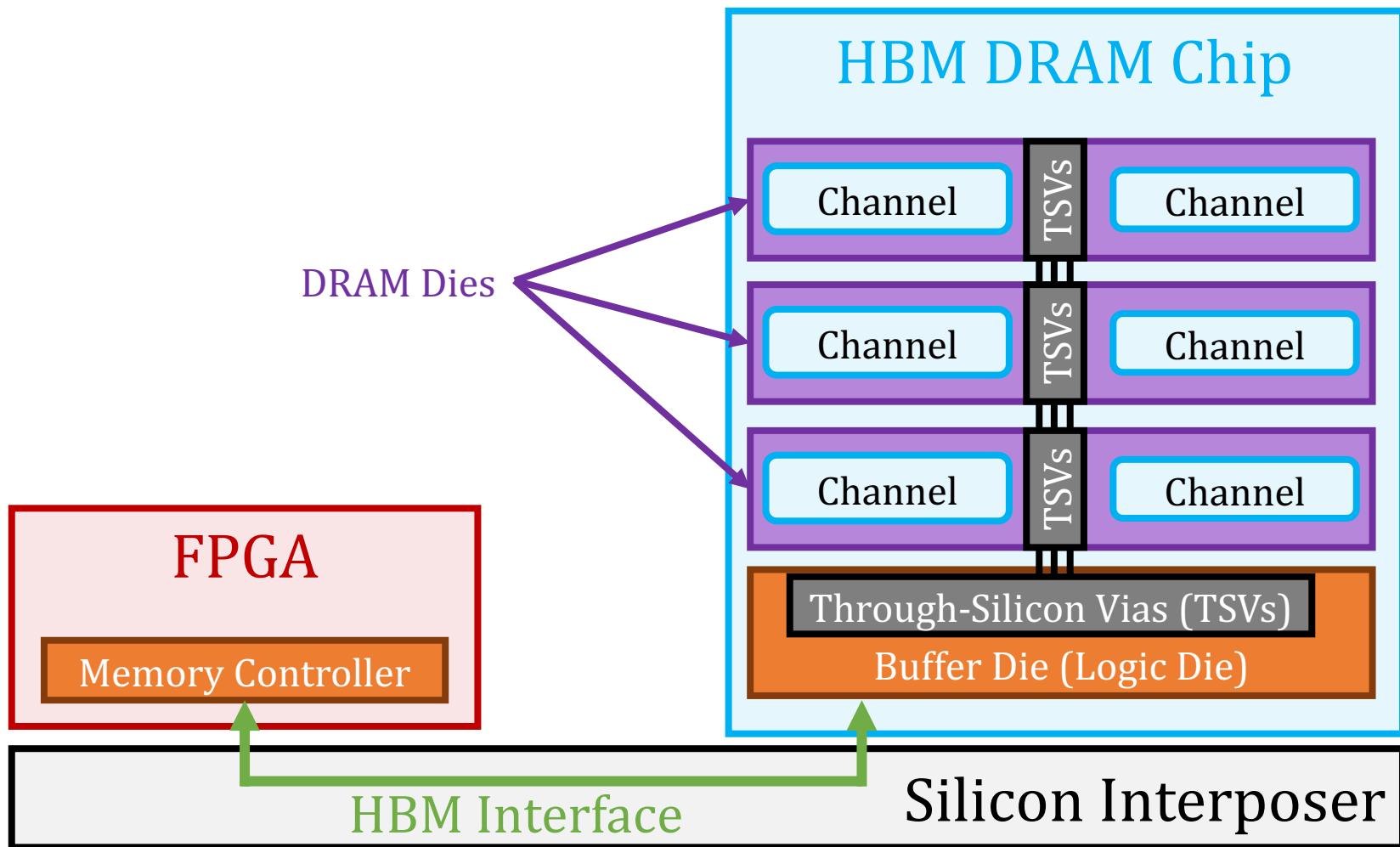
Compute Chip (e.g., FPGA)



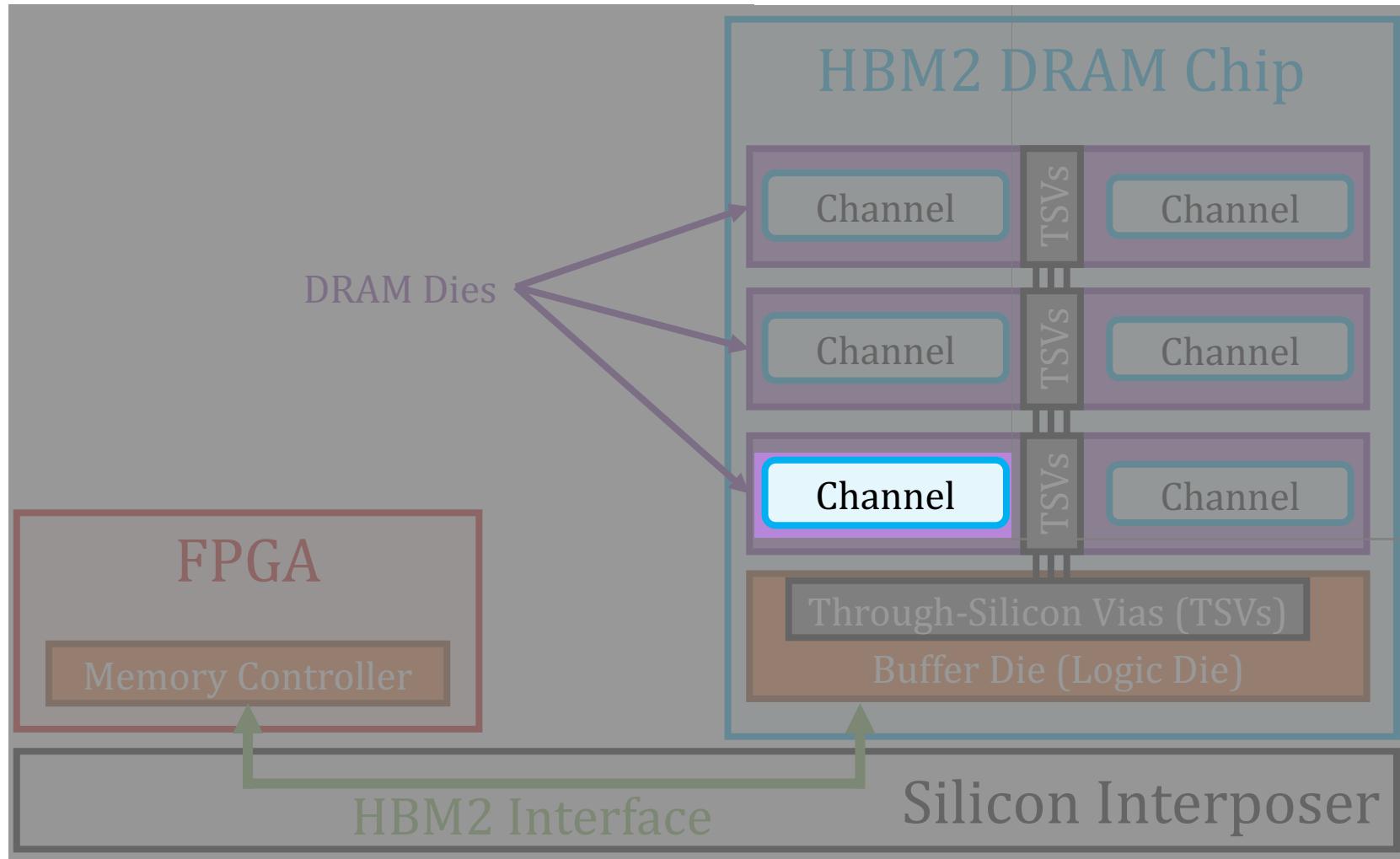
Memory Chip  
(e.g., HBM DRAM)

Inside one package

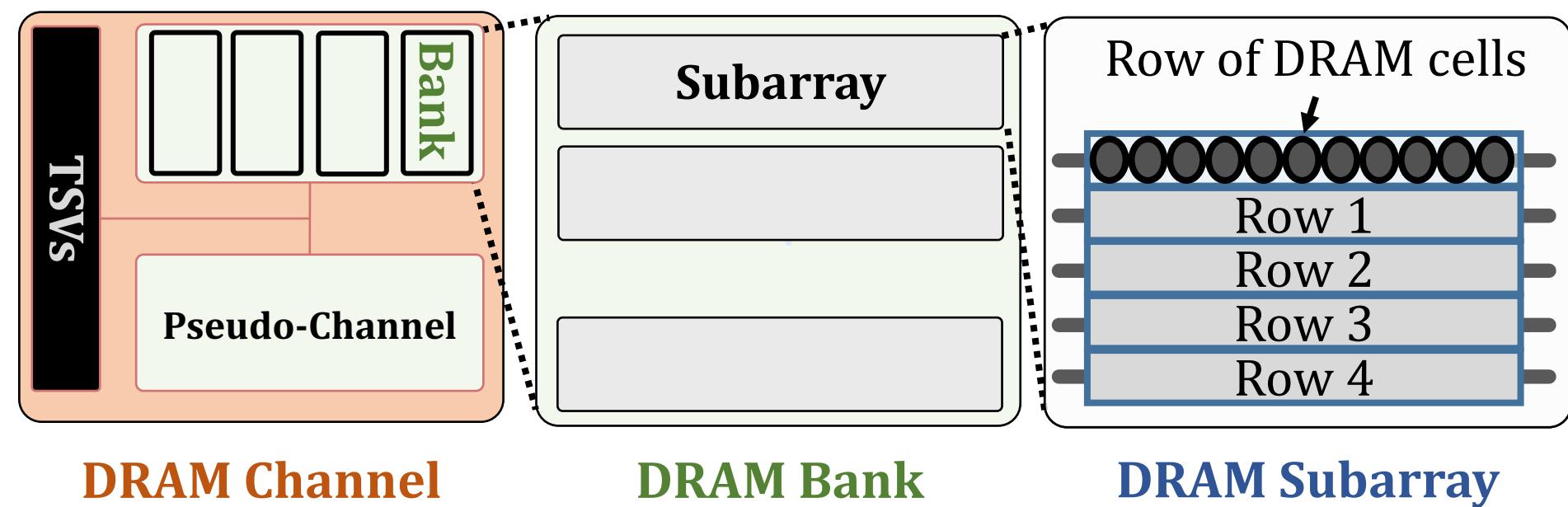
# HBM DRAM Organization (I)



# HBM DRAM Organization (I)



# HBM DRAM Organization (II)



DRAM Channel

DRAM Bank

DRAM Subarray

# Outline

1. HBM DRAM Organization & Operation

2. DRAM Cell Leakage & RowHammer

3. HBM DRAM Testing Methodology

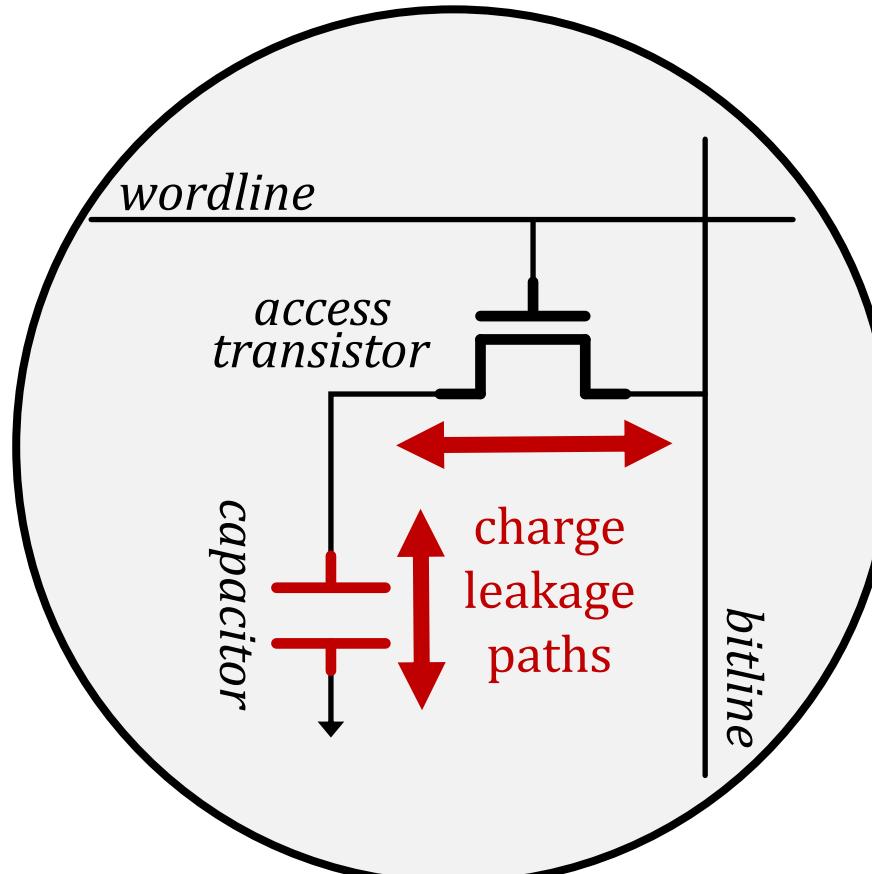
4. RowHammer Spatial Variation Analysis

5. On-die RowHammer Mitigation Analysis

6. Conclusion

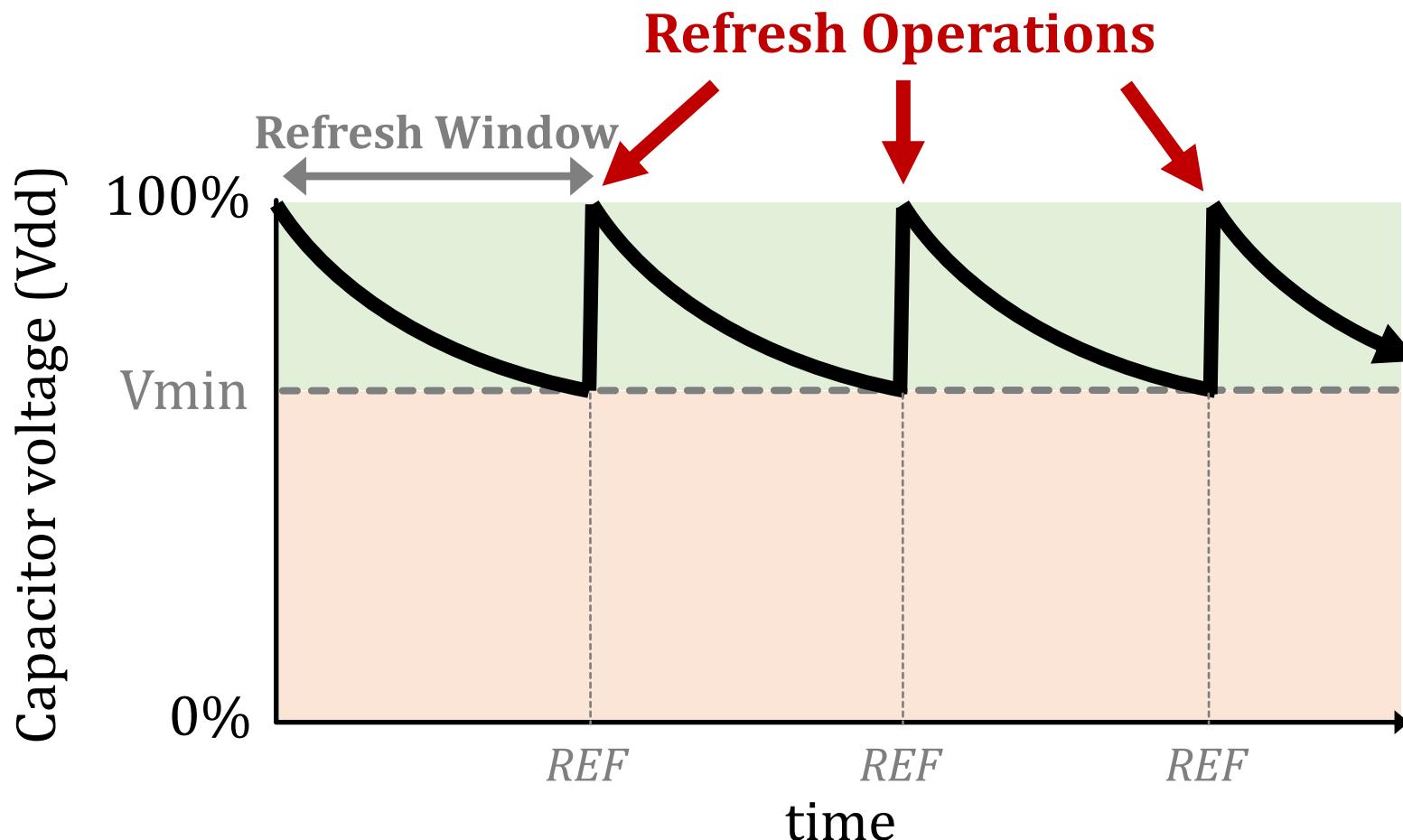
# DRAM Cell Leakage

Each cell encodes information in **leaky** capacitors



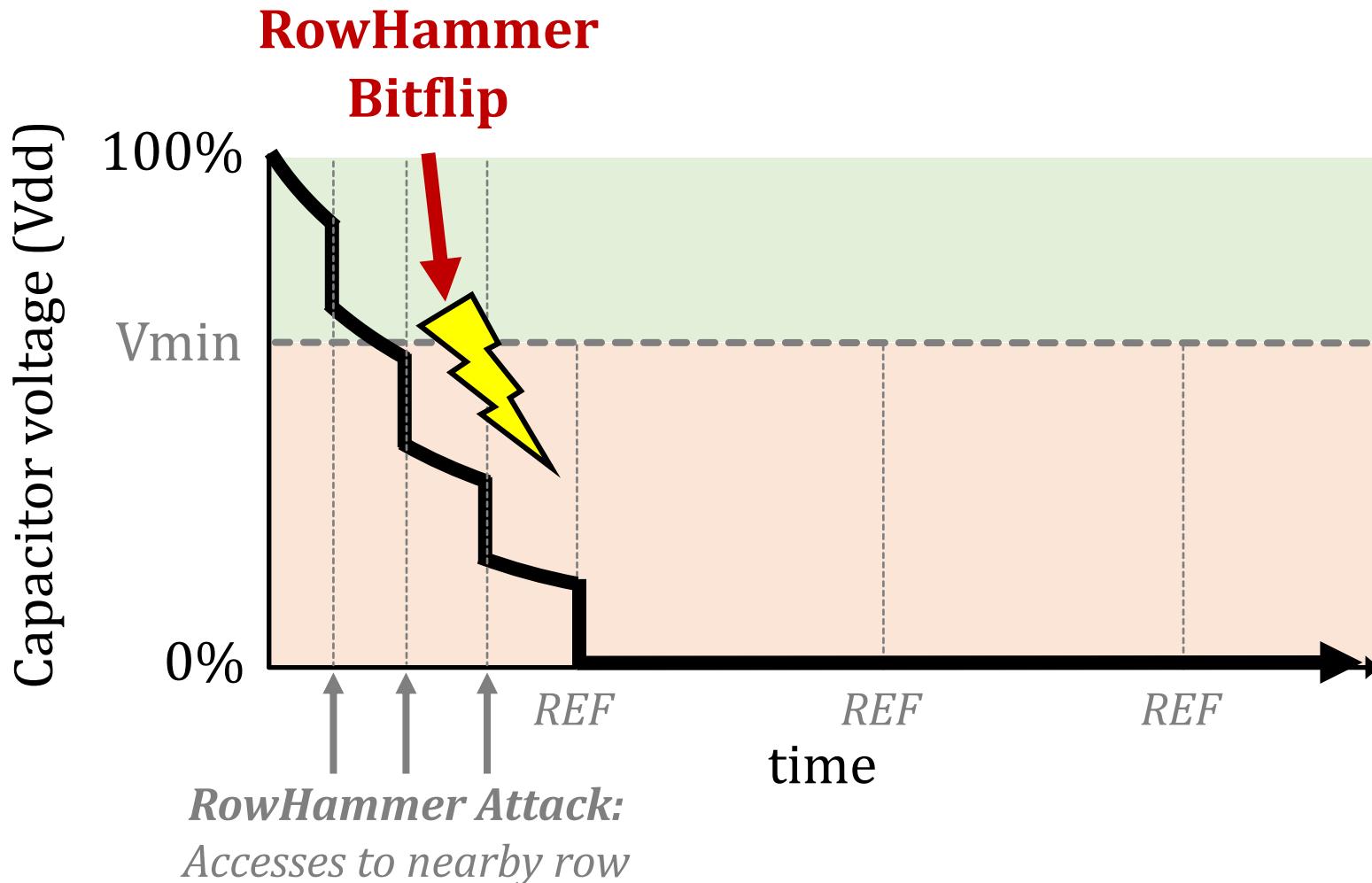
Stored data is **corrupted** if too much charge leaks  
(i.e., the capacitor voltage degrades too much)

# DRAM Refresh



Periodic **refresh operations** preserve stored data

# RowHammer Bitflips



# Problem & Goal

## Problem

No prior study demonstrates  
the RowHammer vulnerability in high bandwidth memory

## Our Goal

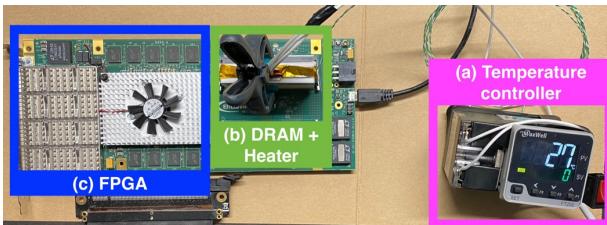
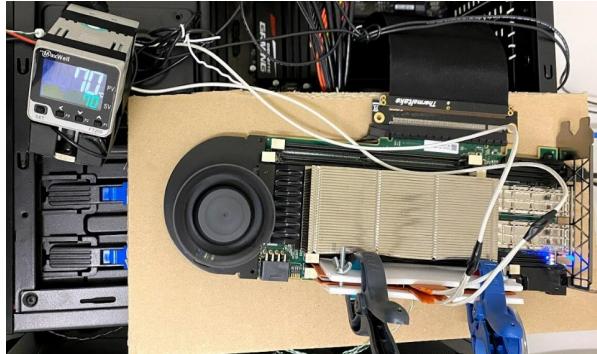
Experimentally analyze how vulnerable  
real high bandwidth memory chips are to RowHammer

# Outline

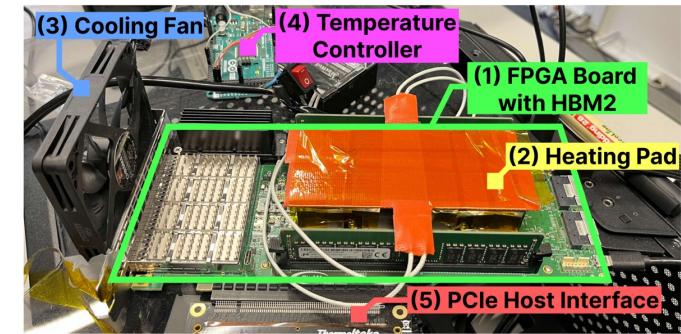
1. HBM DRAM Organization & Operation
2. DRAM Cell Leakage & RowHammer
3. HBM DRAM Testing Methodology
4. RowHammer Spatial Variation Analysis
5. On-die RowHammer Mitigation Analysis
6. Conclusion

# DRAM Testing Infrastructure

## DRAM Bender DDR3/4 Testing Infrastructure



Adapt to work  
with HBM2 chips



<https://github.com/CMU-SAFARI/DRAM-Bender>



CMU-SAFARI / DRAM-Bender

<> Code

Issues ①

Pull requests ①



DRAM-Bender

Public

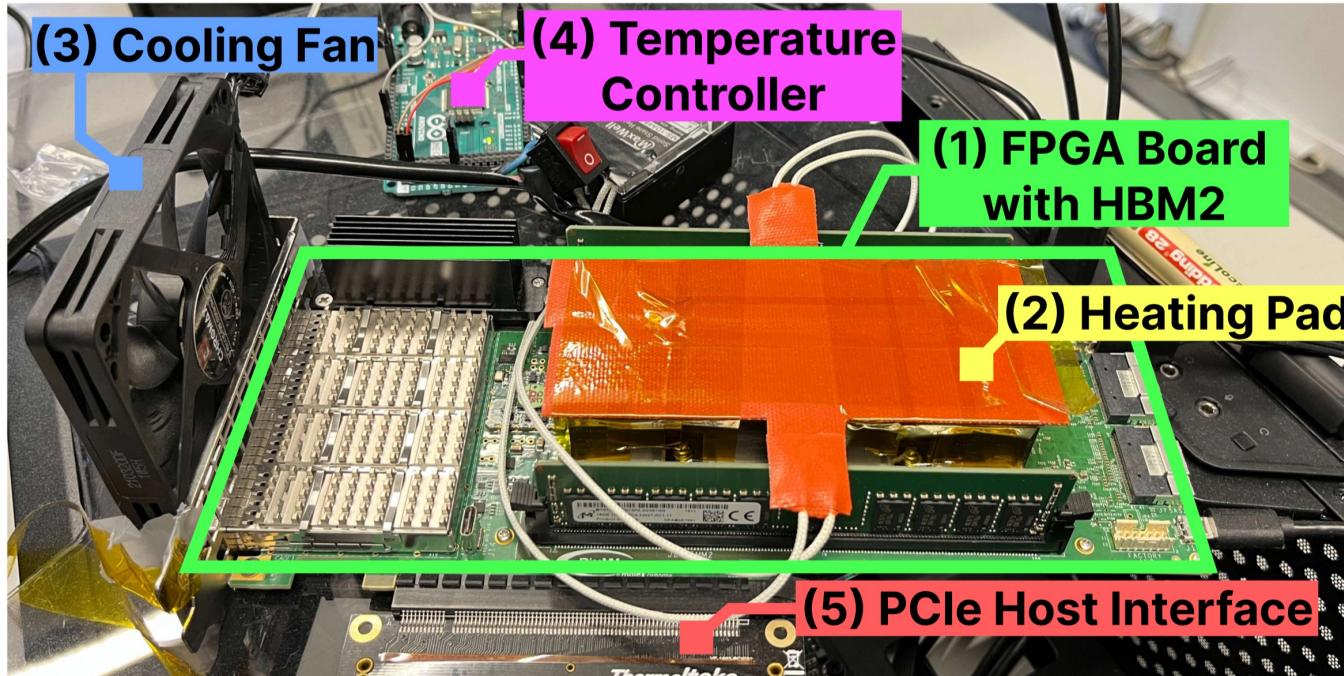
About



DRAM Bender is the first open source DRAM testing infrastructure that can be used to easily and comprehensively test state-of-the-art DDR4 modules of different form factors. Five prototypes are available on different FPGA boards.

# DRAM Testing Infrastructure

## FPGA-based HBM2 Testing Setup (Bittware XUPVVF)



Fine-grained control over **DRAM commands, timing parameters ( $\pm 1.66\text{ns}$ )**

# RowHammer Testing Methodology (I)

To characterize our DRAM chips at **worst-case** conditions:

## 1. Prevent sources of interference during core test loop

- **No DRAM refresh**: to avoid refreshing victim row
- **No RowHammer mitigation mechanisms**: to observe circuit-level effects
- Test for **less than a refresh window (32ms)** to avoid retention failures
- **Repeat tests** for five times

## 2. Worst-case RowHammer access sequence

- We use **worst-case** RowHammer access sequence based on prior works' observations
- Double-sided RowHammer: **repeatedly access the two physically-adjacent rows as fast as possible**



# RowHammer Testing Methodology (II)

- Tested HBM2 chip's organization:

- 8 channels
- 2 pseudo-channels
- 16 banks
- 16384 rows (1 KiB each)



Xilinx FPGA  
with HBM2 DRAM chips

- Test all channels, pseudo-channels, banks
- Test first, middle, and last 3K rows in a bank
  - 9K out of 16K (more than half)
- Keep HBM2 chip temperature at 85°C

# Metrics

## 1. Bit error rate (BER):

The fraction of DRAM cells in a row  
that experience a bitflip after 512K activations

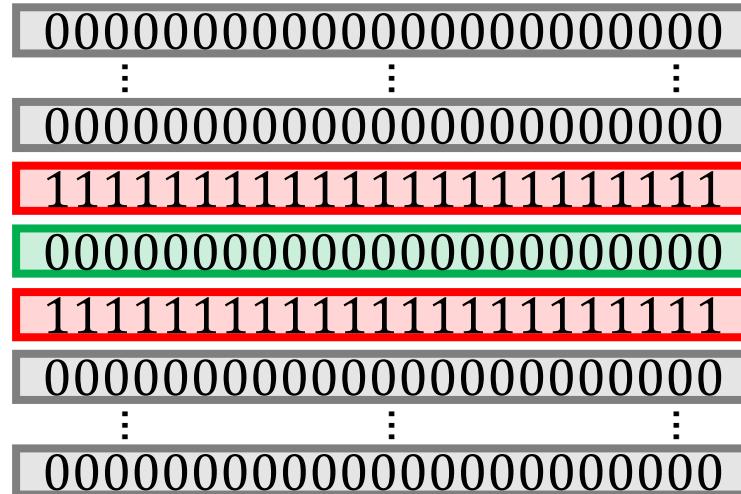
**Higher is worse**

## 2. Hammer Count for the First Bitflip ( $HC_{first}$ ):

Aggressor row activation count  
to cause the first bitflip in the victim row

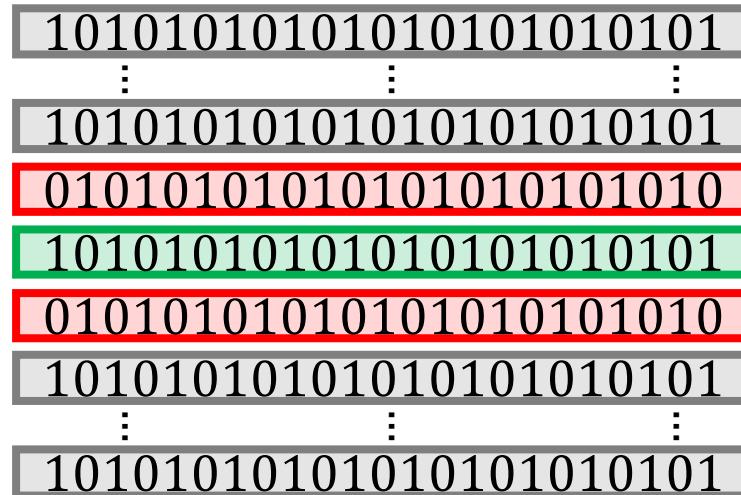
**Lower is worse**

# Tested 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

# Tested Data Patterns



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

# Tested Data Patterns



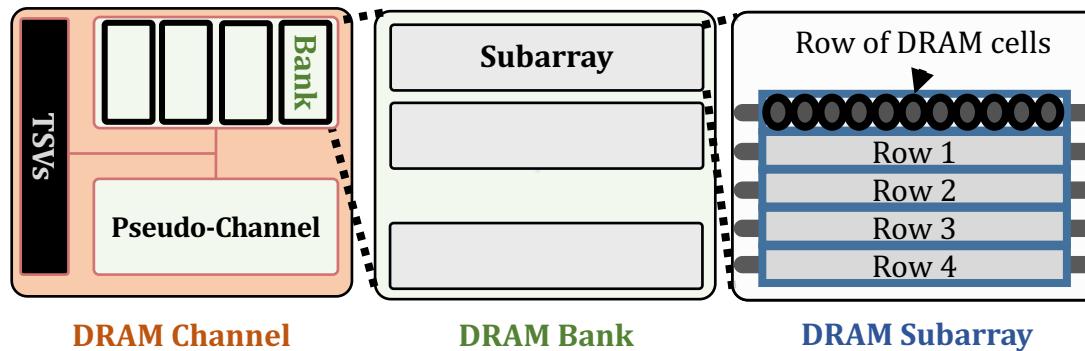
Row Addresses	<i>Rowstripe0</i>	<i>Rowstripe1</i>	<i>Checkered0</i>	<i>Checkered1</i>
Victim (V)	0x00	0xFF	0x55	0xAA
Aggressors ( $V \pm 1$ )	0xFF	0x00	0xAA	0x55
$V \pm [2:8]$	0x00	0xFF	0x55	0xAA

Worst-case data pattern (**WCDP**) of a row: Causes smallest  $HC_{first}$  for a row

# Two Main Analyses

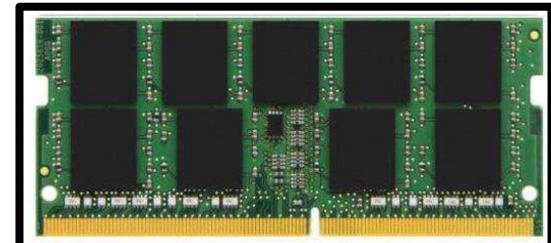
## 1. Spatial variation of RowHammer vulnerability

How does the RowHammer vulnerability change across **channels, pseudo-channels, banks, rows** in HBM?



## 2. On-DRAM-die RowHammer mitigations

Do real HBM chips implement  
undisclosed RowHammer mitigations  
resembling those that exist in DDR4?



# Outline

1. HBM DRAM Organization & Operation

2. DRAM Cell Leakage & RowHammer

3. HBM DRAM Testing Methodology

4. RowHammer Spatial Variation Analysis

5. On-die RowHammer Mitigation Analysis

6. Conclusion

# Key Takeaways from Spatial Variation Analysis

## Takeaway 1

Different 3D-stacked HBM2 channels exhibit different RowHammer vulnerability

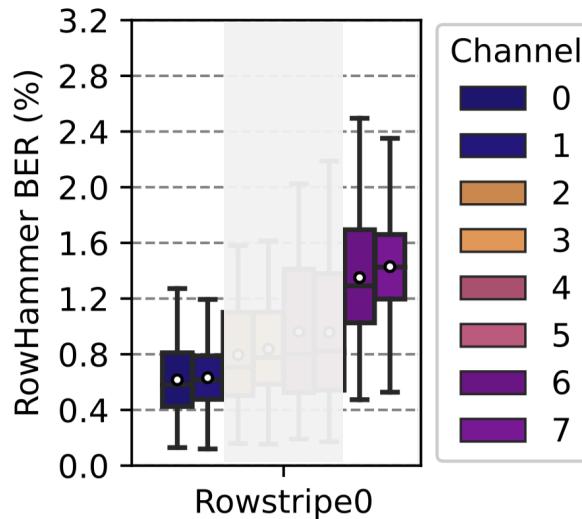
## Takeaway 2

DRAM rows near the end of a DRAM bank experience smaller bit error rate (BER) than others

## Takeaway 3

Activation count needed to induce the first RowHammer bitflip ( $HC_{first}$ ) changes with the data pattern and the physical location of the DRAM row

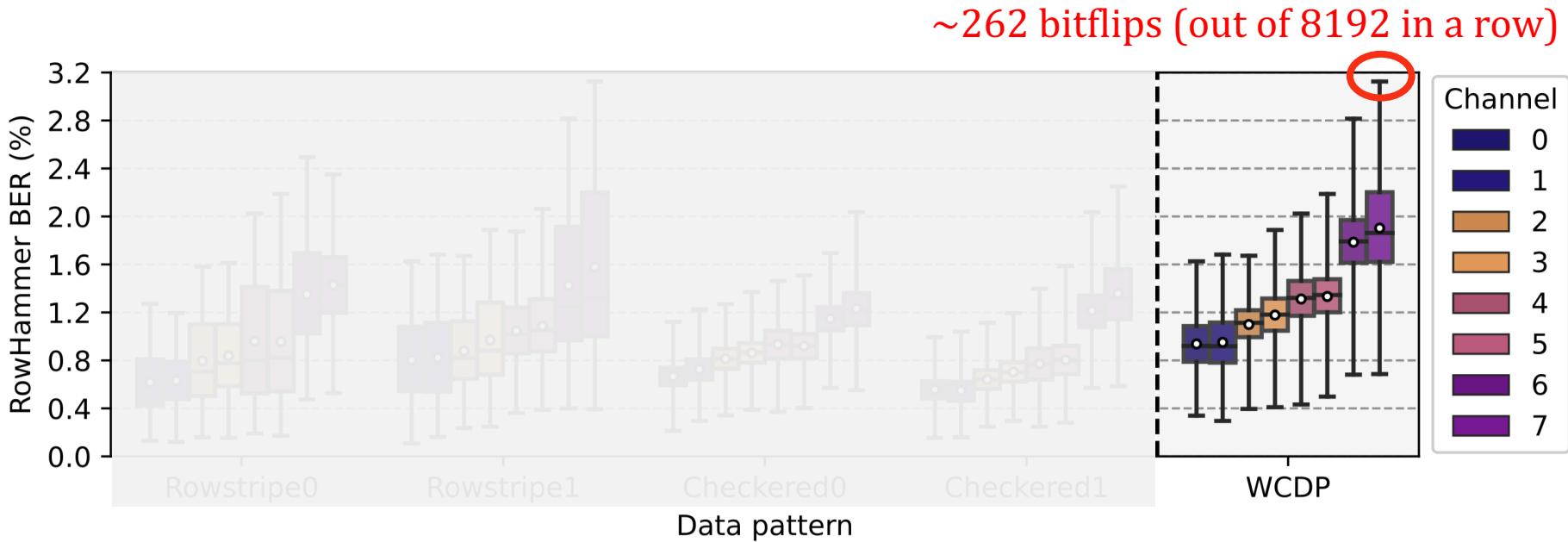
# Spatial Distribution of BER (I)



There are **bitflips** in **every** tested DRAM row  
across **all** tested HBM2 **channels**

BER **varies across channels**:  
groups of two channels have different BERs

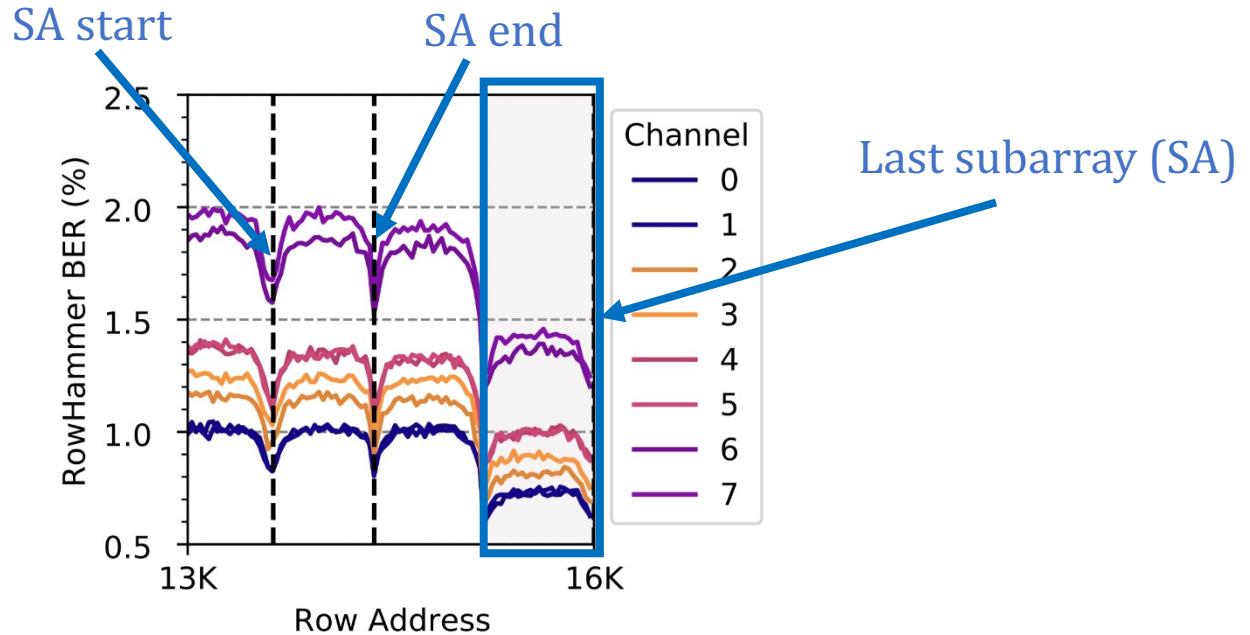
# Spatial Distribution of BER (I)



The data pattern affects the BER distribution

Up to ~262 bitflips in a row of 8K bits  
with 512K aggressor row activations

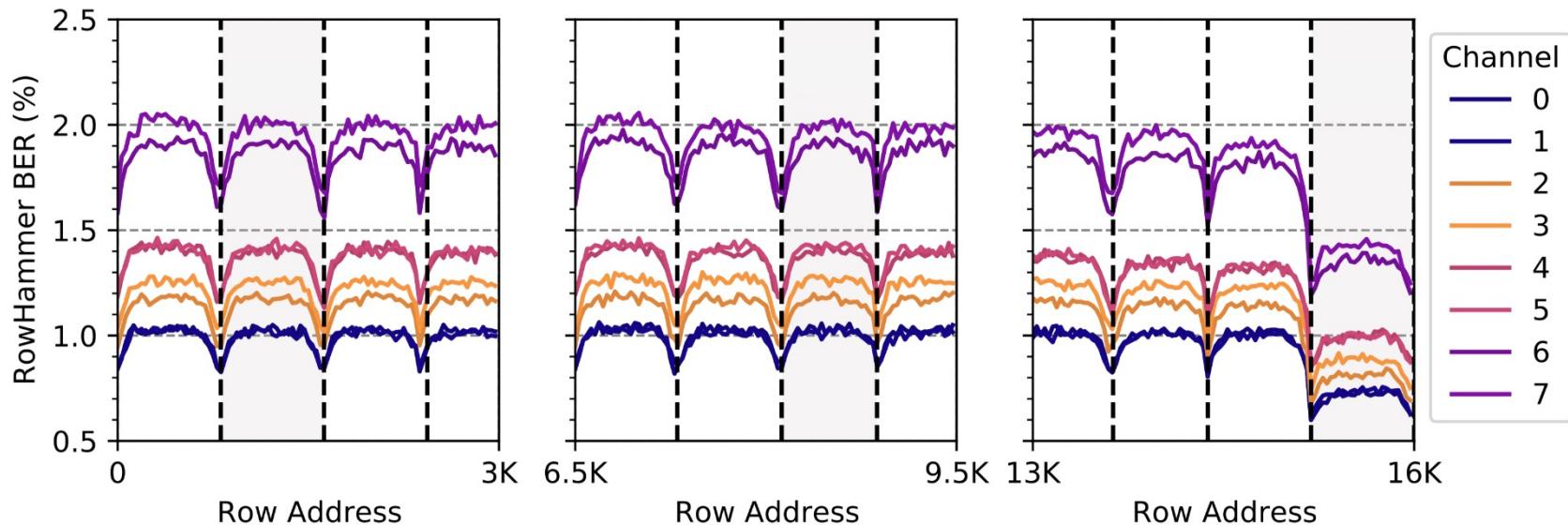
# Spatial Distribution of BER (II)



BER is substantially smaller in the **last subarray** (i.e., last 832 rows)

BER periodically increases and decreases across rows:  
BER is **higher** in the **middle of a subarray**

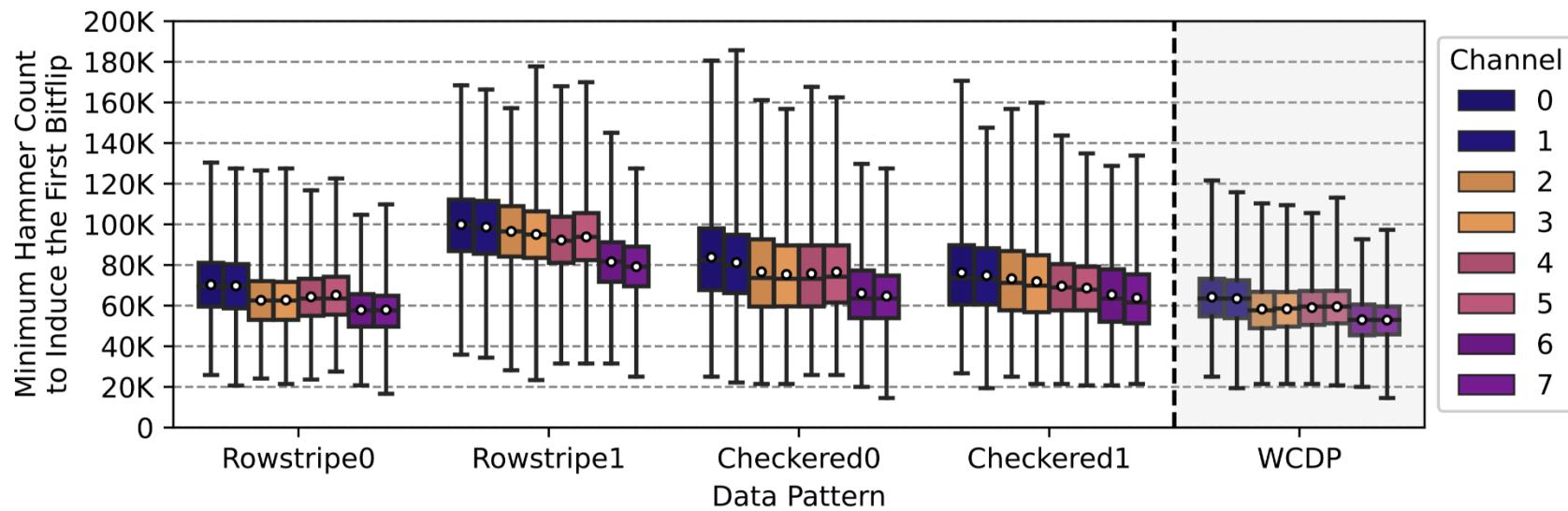
# Spatial Distribution of BER (II)



BER is substantially smaller in the **last subarray** (i.e., last 832 rows)

BER periodically increases and decreases across rows:  
BER is **higher** in the **middle of a subarray**

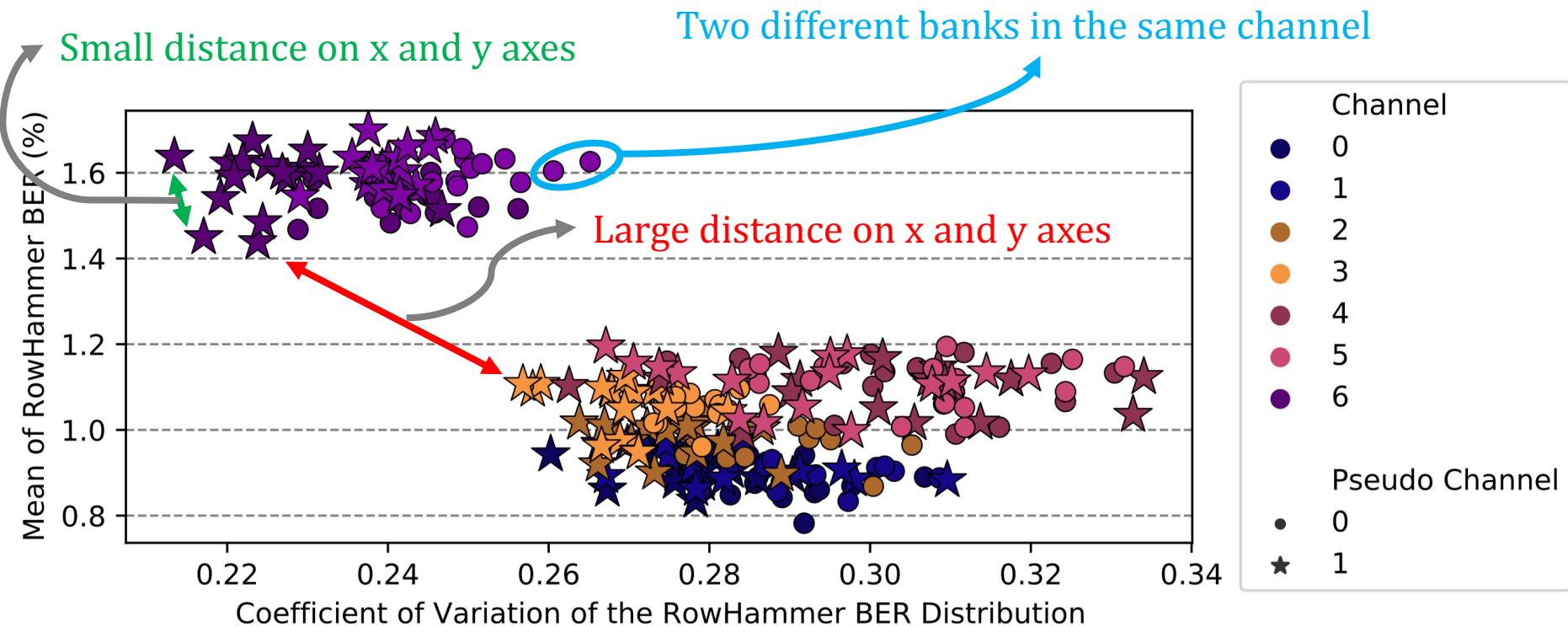
# Spatial Distribution of HC<sub>first</sub>



HC<sub>first</sub> is as low as 14531 across all tested rows/channels:  
*Only ~1.3 ms* to induce a RowHammer bitflip

HC<sub>first</sub> distribution heavily depends on the data pattern

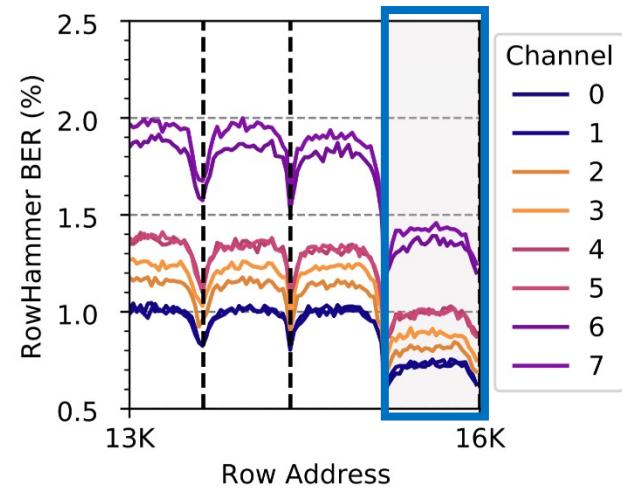
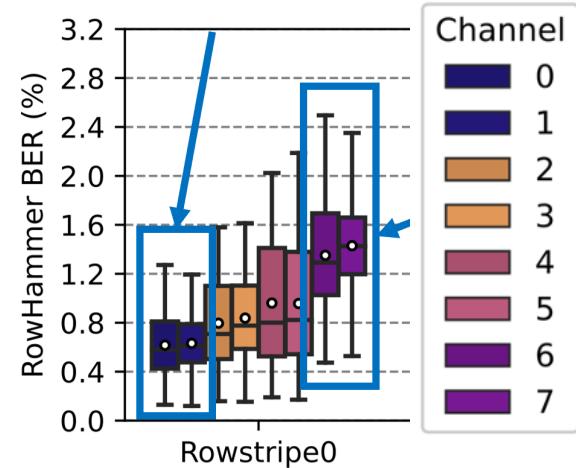
# Variation in Bit Error Rate



Banks in the same channel have similar variation in BER

# Hypotheses from Characterization

1. Similar BER & HC<sub>first</sub> within **groups of two channels** suggests these channels share DRAM dies
2. RowHammer BER changes with the row's proximity to **sense amplifiers and bank I/O**



# Implications on Attacks and Mitigations

**Key Observation:** RowHammer BER and HC<sub>first</sub> vary across channels

Two implications for RowHammer attacks and mitigations

A RowHammer attack can use the most-RH-vulnerable HBM2 channel to prepare for and perform the attack faster

A RowHammer mitigation can allocate fewer resources for RowHammer-resilient channels and more efficiently prevent RowHammer bitflips

# Outline

1. HBM DRAM Organization & Operation

2. DRAM Cell Leakage & RowHammer

3. HBM DRAM Testing Methodology

4. RowHammer Spatial Variation Analysis

5. On-die RowHammer Mitigation Analysis

6. Conclusion

# Key Takeaways from on-die Mitigation Analysis

## Takeaway 1

A modern HBM2 chip implements an undisclosed on-DRAM-die RowHammer mitigation

## Takeaway 2

This mitigation resembles the one in DDR4 chips from one major manufacturer as shown in prior work

# On-Die RowHammer Mitigation Analysis (I)

HBM2 standard defines a “Target Row Refresh (TRR)-mode”

- Memory controller and DRAM **cooperate** to prevent RH bitflips

Real DDR4 chips implement **on-die mitigation mechanisms**

- Memory-controller-**transparent**, **hidden** behind periodic REF

*Does a similar **hidden** mitigation mechanism exist in HBM2?*

# On-Die RowHammer Mitigation Analysis (II)

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<sup>†</sup>

<sup>†</sup>ETH Zürich

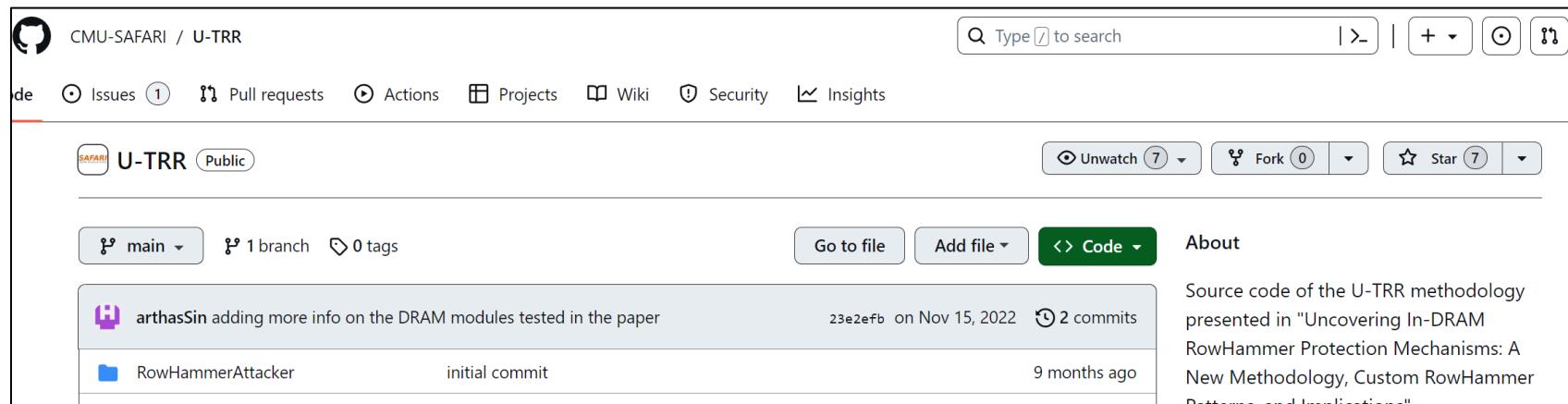
Yahya Can Tuğrul<sup>†‡</sup>  
Kaveh Razavi<sup>†</sup>

<sup>‡</sup>TOBB University of Economics & Technology

Jeremie S. Kim<sup>†</sup>  
Onur Mutlu<sup>†</sup>

Victor van der Veen<sup>σ</sup>  
<sup>σ</sup>Qualcomm Technologies Inc.

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



The screenshot shows a GitHub repository page for "CMU-SAFARI / U-TRR". The repository is public and contains one branch and zero tags. The last commit was made by "arthasSin" on Nov 15, 2022, with two commits. The commit message is "adding more info on the DRAM modules tested in the paper". The repository has 7 forks and 7 stars. The "Code" button is highlighted in green. The "About" section on the right provides a brief description of the methodology presented in the paper.

CMU-SAFARI / U-TRR

Issues (1) Pull requests Actions Projects Wiki Security Insights

U-TRR Public

Unwatch (7) Fork (0) Star (7)

main 1 branch 0 tags Go to file Add file Code

arthasSin adding more info on the DRAM modules tested in the paper 23e2efb on Nov 15, 2022 2 commits

RowHammerAttacker initial commit 9 months ago

About

Source code of the U-TRR methodology presented in "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"

# 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

Sample as aggressor row

On-DRAM-die  
Mitigation

Aggressor Row R + 1

Victim Row R

Refresh victim row

Refresh R

Mitigation refreshes R

Read R

time = 0

time =  $T/2$

time = T

Timeline

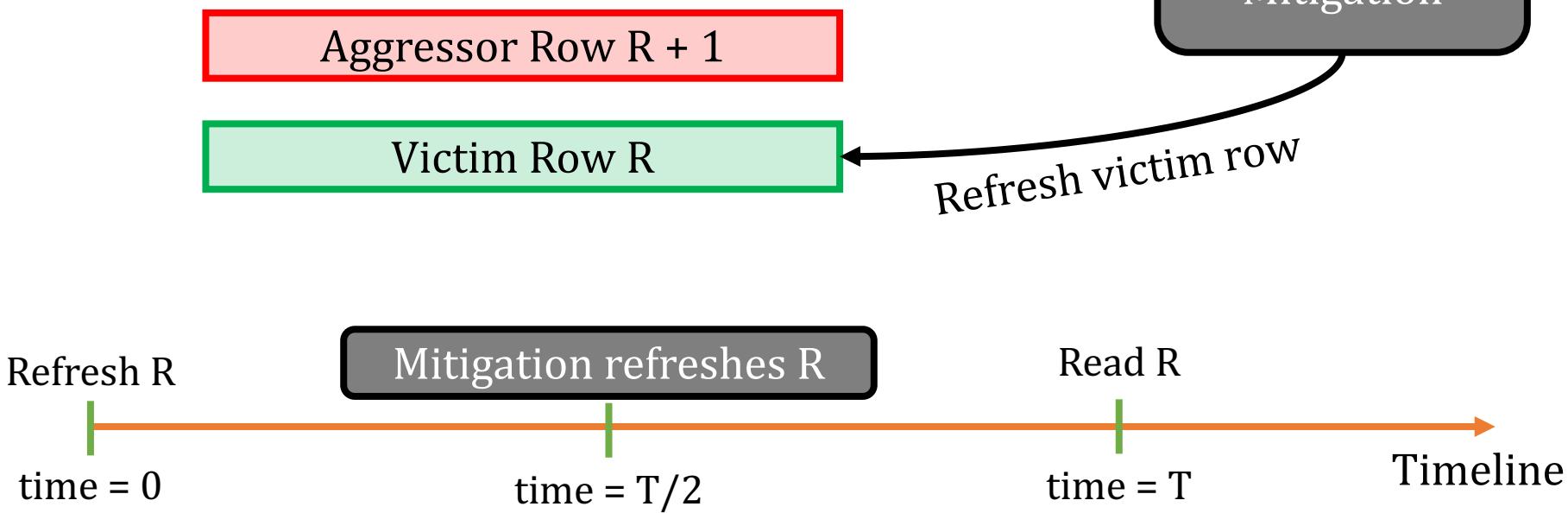
# Experimental Methodology

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

On-DRAM-die  
Mitigation



# Experimental Methodology

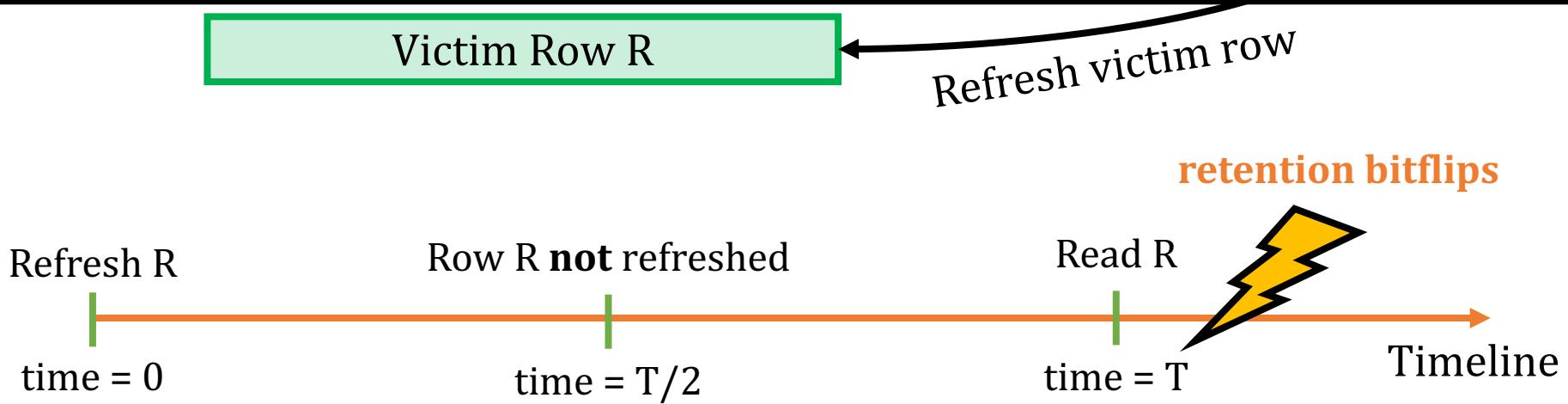
1. Identify a row ( $R$ ) with  $T$  retention time

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  $T/2$



# HBM2 DRAM Chips Implement Undisclosed TRR

The HBM2 chip **implements** an **undisclosed** on-die RowHammer mitigation mechanism

This mechanism **performs** a victim row refresh operation every **17** periodic refresh (REF) operations

This mitigation **resembles** the one in DDR4 chips from one major manufacturer

# Outline

1. HBM DRAM Organization & Operation

2. DRAM Cell Leakage & RowHammer

3. HBM DRAM Testing Methodology

4. RowHammer Spatial Variation Analysis

5. On-die RowHammer Mitigation Analysis

6. Conclusion

# Conclusion

We provide the first detailed experimental characterization of RowHammer in a modern HBM2 DRAM chip

Different channels in 3D-stacked HBM chips exhibit different RowHammer vulnerability

DRAM rows near the end of a DRAM bank are more RowHammer resilient

Two implications for RowHammer attacks and mitigations:

1. Faster and more effective attacks
2. More efficient mitigations

A modern HBM chip implements undisclosed on-DRAM-die RowHammer mitigation (e.g., similar to DDR4 chips)

**Future Directions:** To present more insights into how RowHammer behaves in HBM

1. Test more HBM DRAM chips, data patterns, at different temperature and voltage levels
2. Investigate read-disturb-based interference across different 3D-stacked HBM DRAM channels
3. Study the effects of the new read-disturb phenomenon, RowPress [Luo+, ISCA'23]

# Available on ArXiv

<https://arxiv.org/abs/2305.17918>

arXiv > cs > arXiv:2305.17918

Search... All fields  Help | Advanced Search

Computer Science > Cryptography and Security

[Submitted on 29 May 2023]

## An Experimental Analysis of RowHammer in HBM2 DRAM Chips

Ataberk Olgun, Majd Osseiran, Abdullah Giray Yağılık, Yahya Can Tuğrul, Haocong Luo, Steve Rhyner, Behzad Salami, Juan Gomez Luna, Onur Mutlu

RowHammer (RH) is a significant and worsening security, safety, and reliability issue of modern DRAM chips that can be exploited to break memory isolation. Therefore, it is important to understand real DRAM chips' RH characteristics. Unfortunately, no prior work extensively studies the RH vulnerability of modern 3D-stacked high-bandwidth memory (HBM) chips, which are commonly used in modern GPUs.

In this work, we experimentally characterize the RH vulnerability of a real HBM2 DRAM chip. We show that 1) different 3D-stacked channels of HBM2 memory exhibit significantly different levels of RH vulnerability (up to 79% difference in bit error rate), 2) the DRAM rows at the end of a DRAM bank (rows with the highest addresses) exhibit significantly fewer RH bitflips than other rows, and 3) a modern HBM2 DRAM chip implements undisclosed RH defenses that are triggered by periodic refresh operations. We describe the implications of our observations on future RH attacks and defenses and discuss future work for understanding RH in 3D-stacked memories.

Comments: To appear at DSN Disrupt 2023

Subjects: Cryptography and Security (cs.CR); Hardware Architecture (cs.AR)

Cite as: arXiv:2305.17918 [cs.CR]  
(or arXiv:2305.17918v1 [cs.CR] for this version)  
<https://doi.org/10.48550/arXiv.2305.17918>

**Download:**

- PDF
- Other formats

Current browse context:  
cs.CR  
< prev | next >  
new | recent | 2305

Change to browse by:  
cs  
cs.AR

References & Citations

- NASA ADS
- Google Scholar
- Semantic Scholar

Export BibTeX Citation

Bookmark

# Extended version

## Understanding Read Disturbance in High Bandwidth Memory: An Experimental Analysis of Real HBM2 DRAM Chips

- Tests 5 more HBM2 chips
- Tests more DRAM components (e.g., banks and rows) per chip
- Analyzes hammer counts to induce more than one bitflip ( $HC_{\text{second, third, \dots, tenth}}$ )
- Analyzes the RowPress vulnerability of HBM2 chips
- Further reverse engineers the on-DRAM-die RH defense mechanism

# Methodology

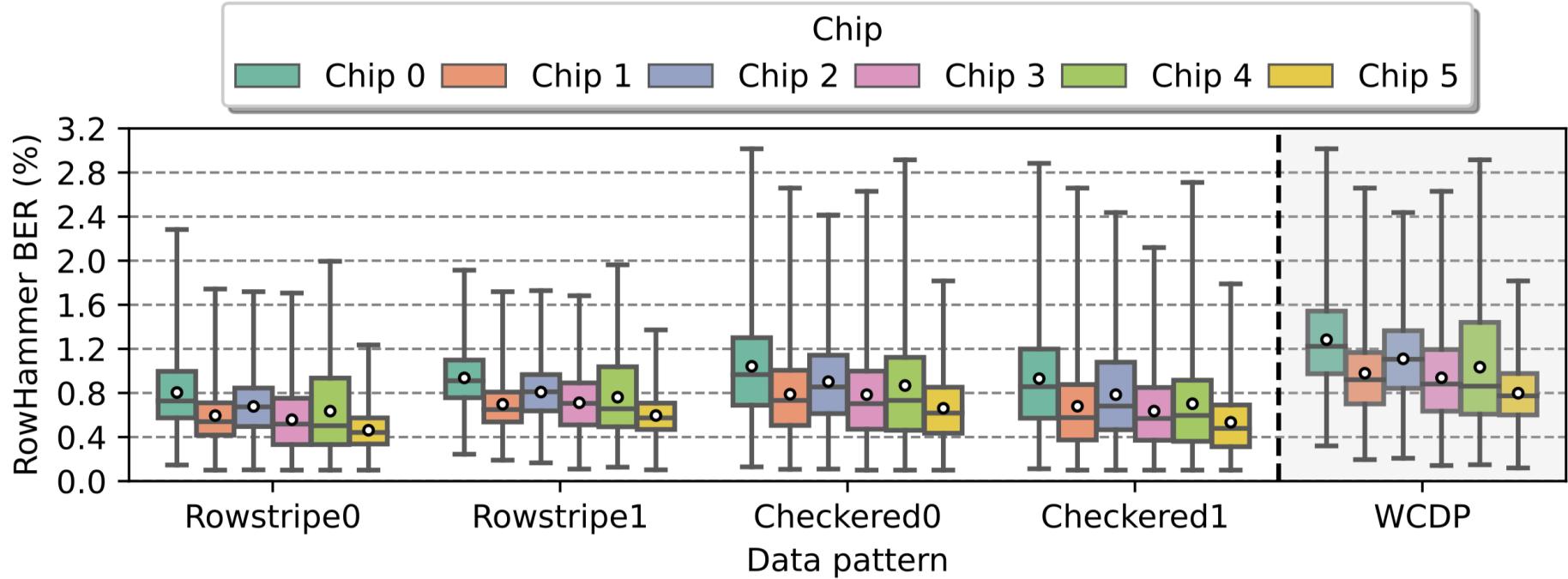
TABLE II  
TESTED DRAM COMPONENTS FOR EACH EXPERIMENT TYPE

Experiment Type	Rows (Per Bank)	Banks	Pseudo Channels	Channels
RowHammer <i>BER</i>	16384	1	1	8
RowHammer <i>HC<sub>first</sub></i>	3072	3	2	8
RowPress <i>BER</i>	384	1	1	3
RowPress <i>HC<sub>first</sub></i>	384	1	1	3

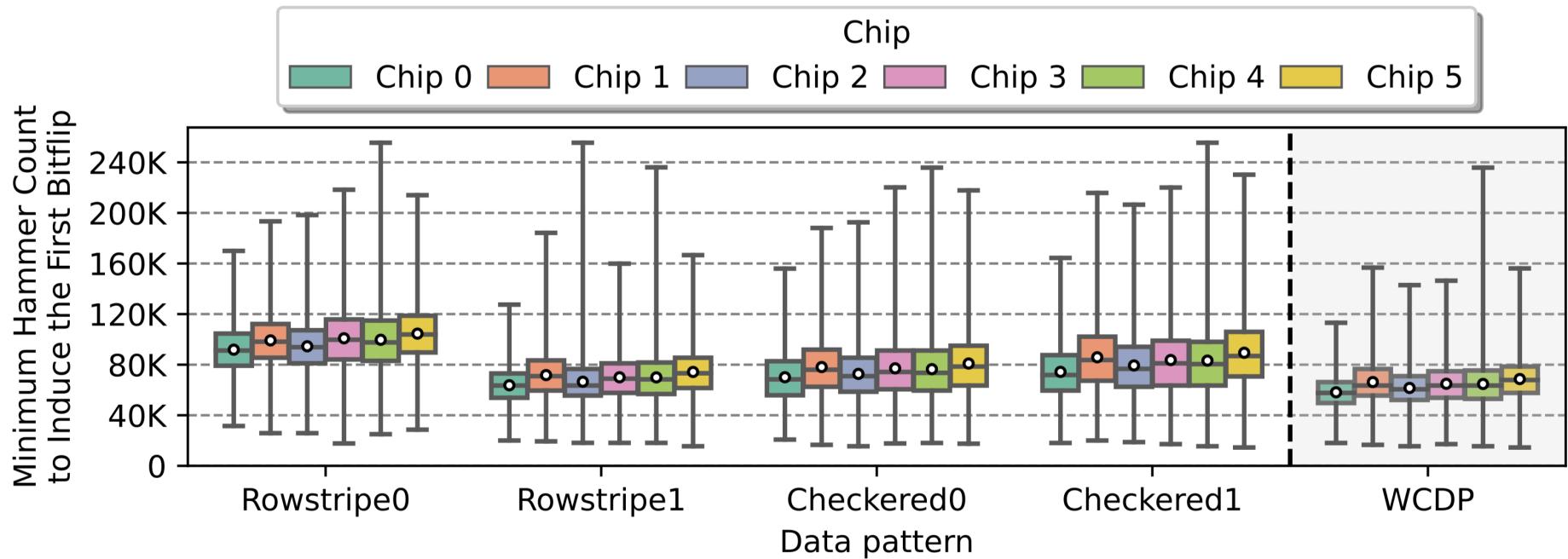
TABLE III  
LABELS FOR THE HBM2 CHIPS IN EACH TESTED FPGA BOARD

FPGA Board	Chip Label
Bittware XUPVVH	Chip 0
AMD Xilinx Alveo U50	Chip 1, 2, 3, 4, 5

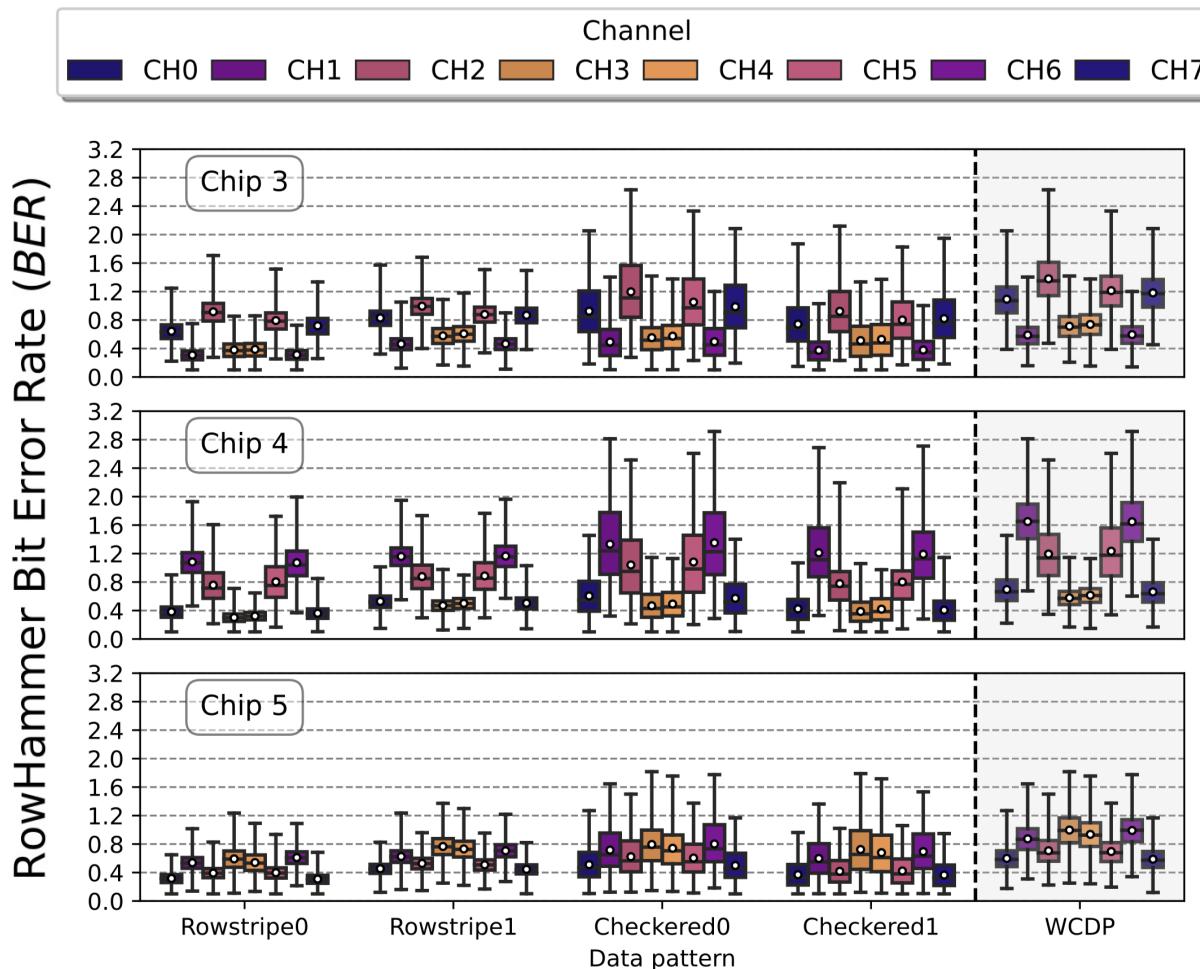
# RowHammer BER Across Chips



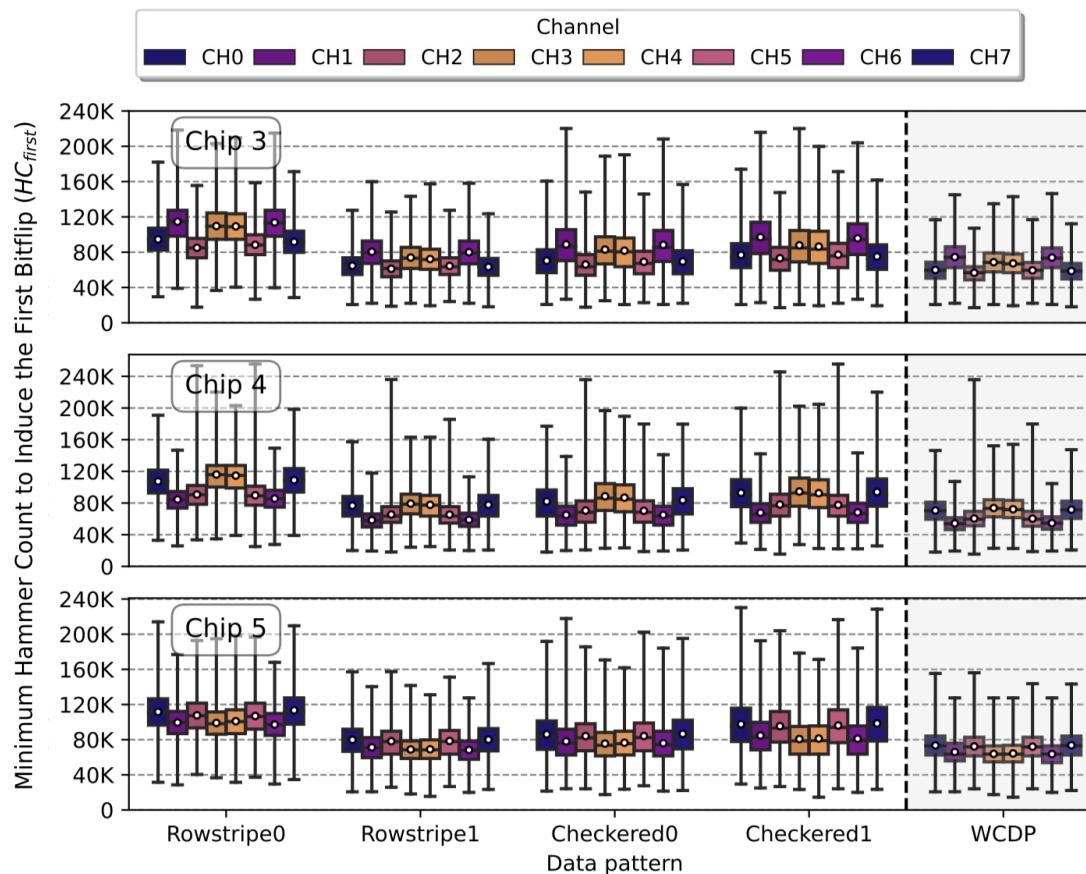
# RowHammer HC<sub>first</sub> Across Chips



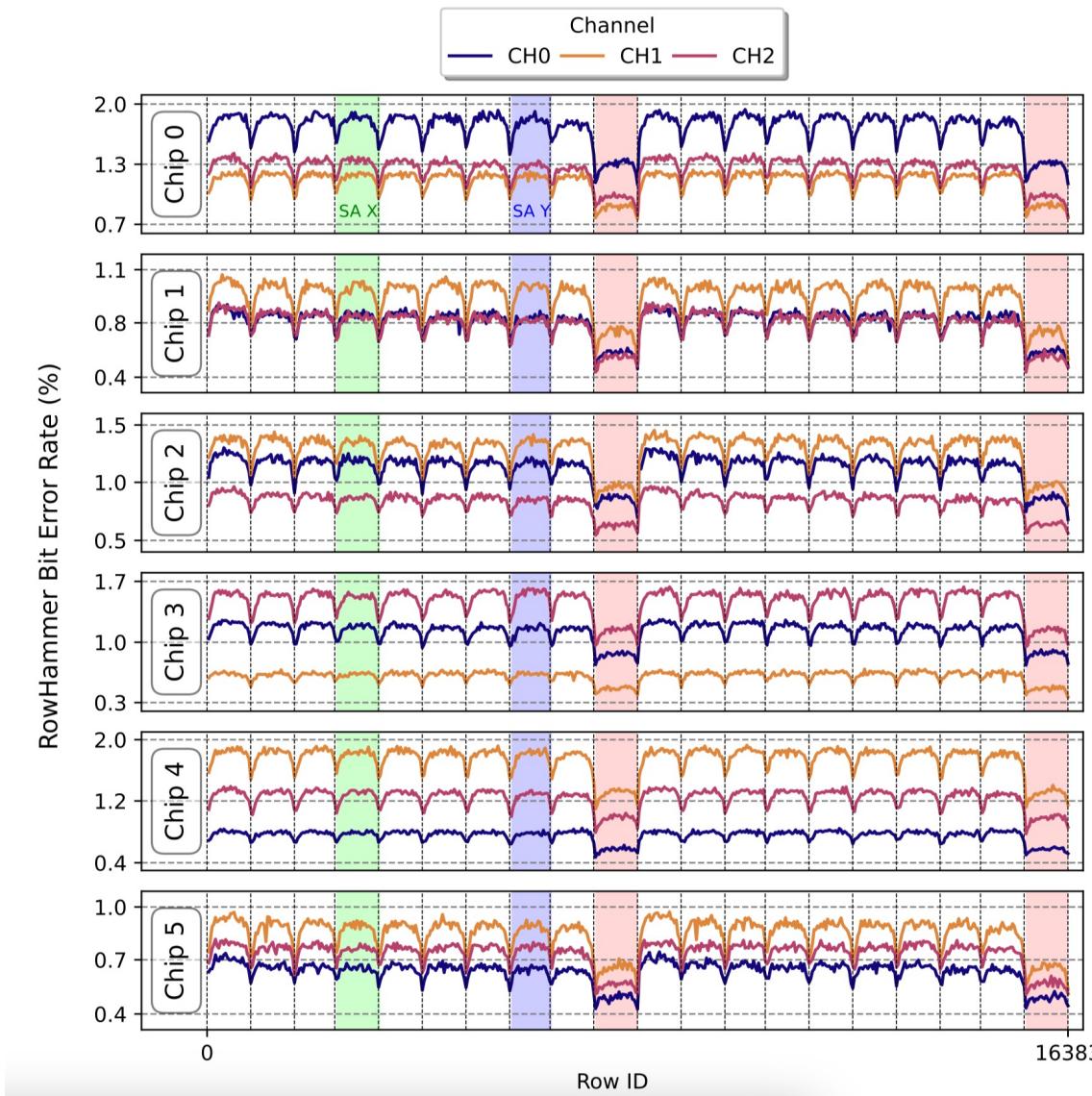
# RowHammer BER Across Channels



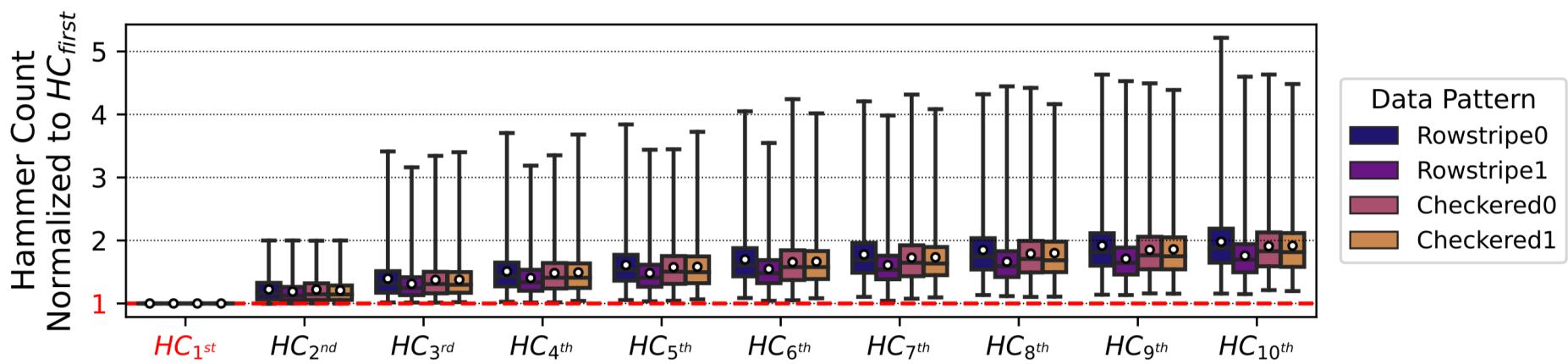
# RowHammer HC<sub>first</sub> Across Channels



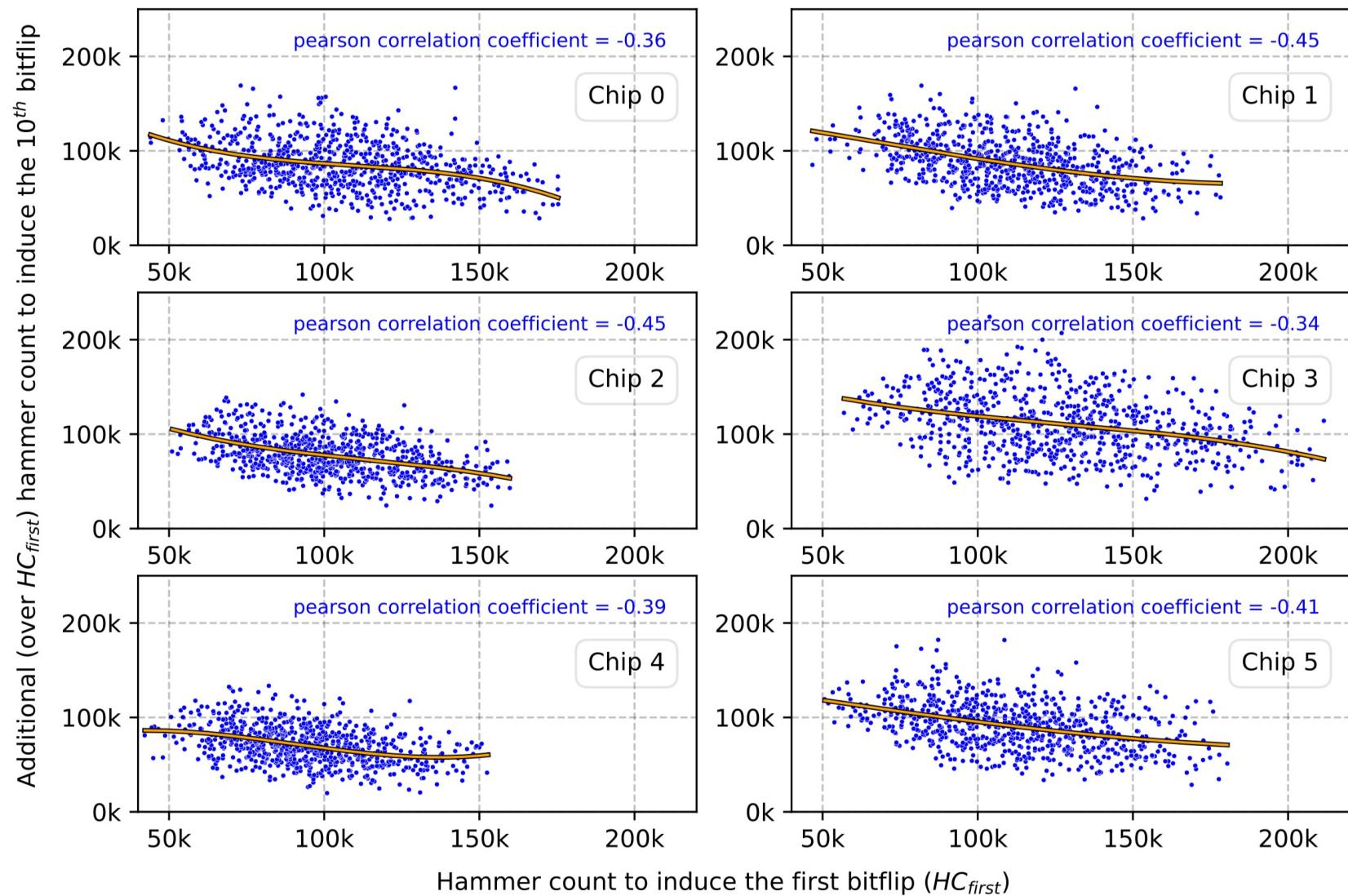
# RowHammer BER Across Rows



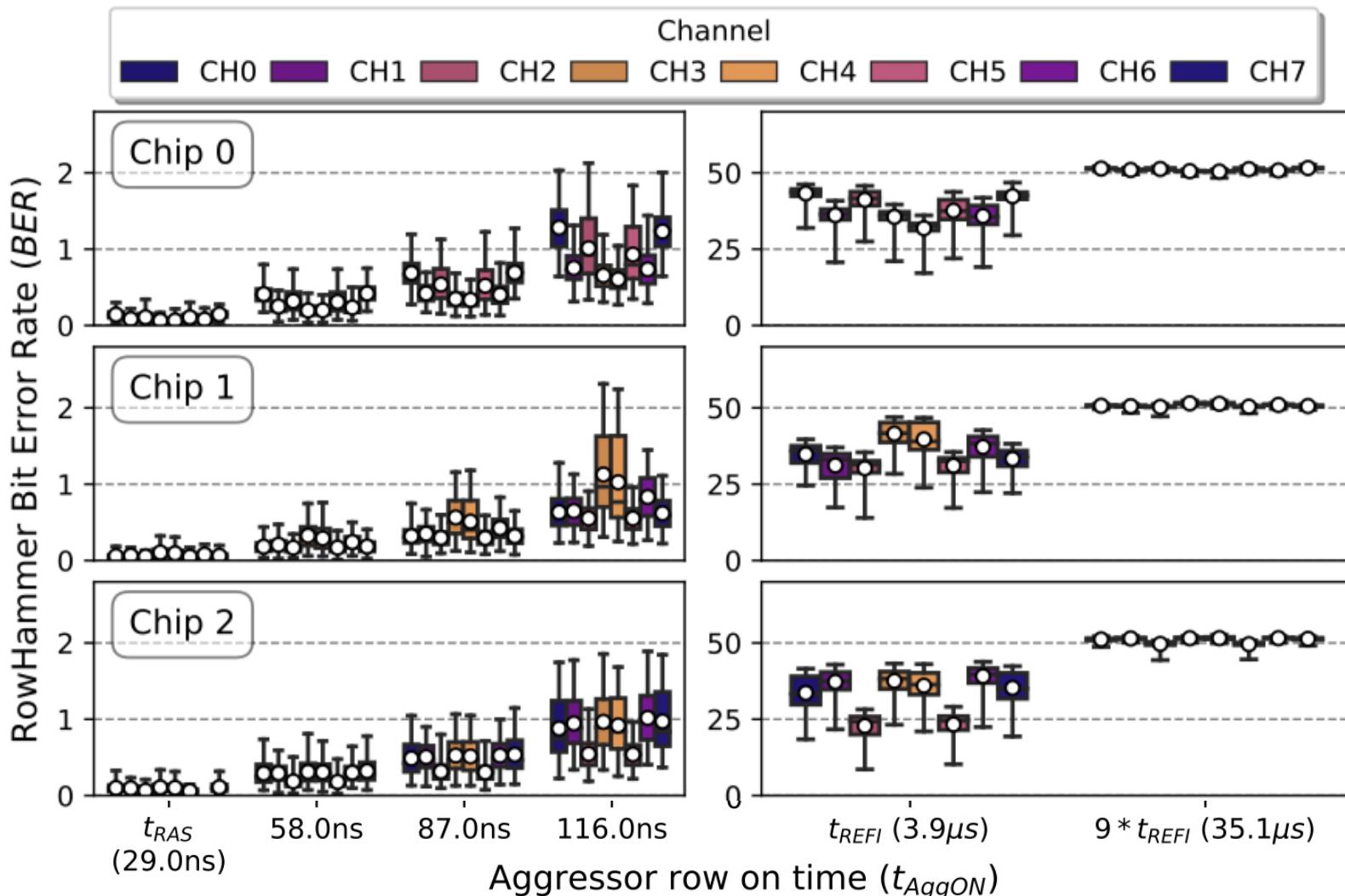
# RowHammer's Sensitivity to Hammer Count (I)



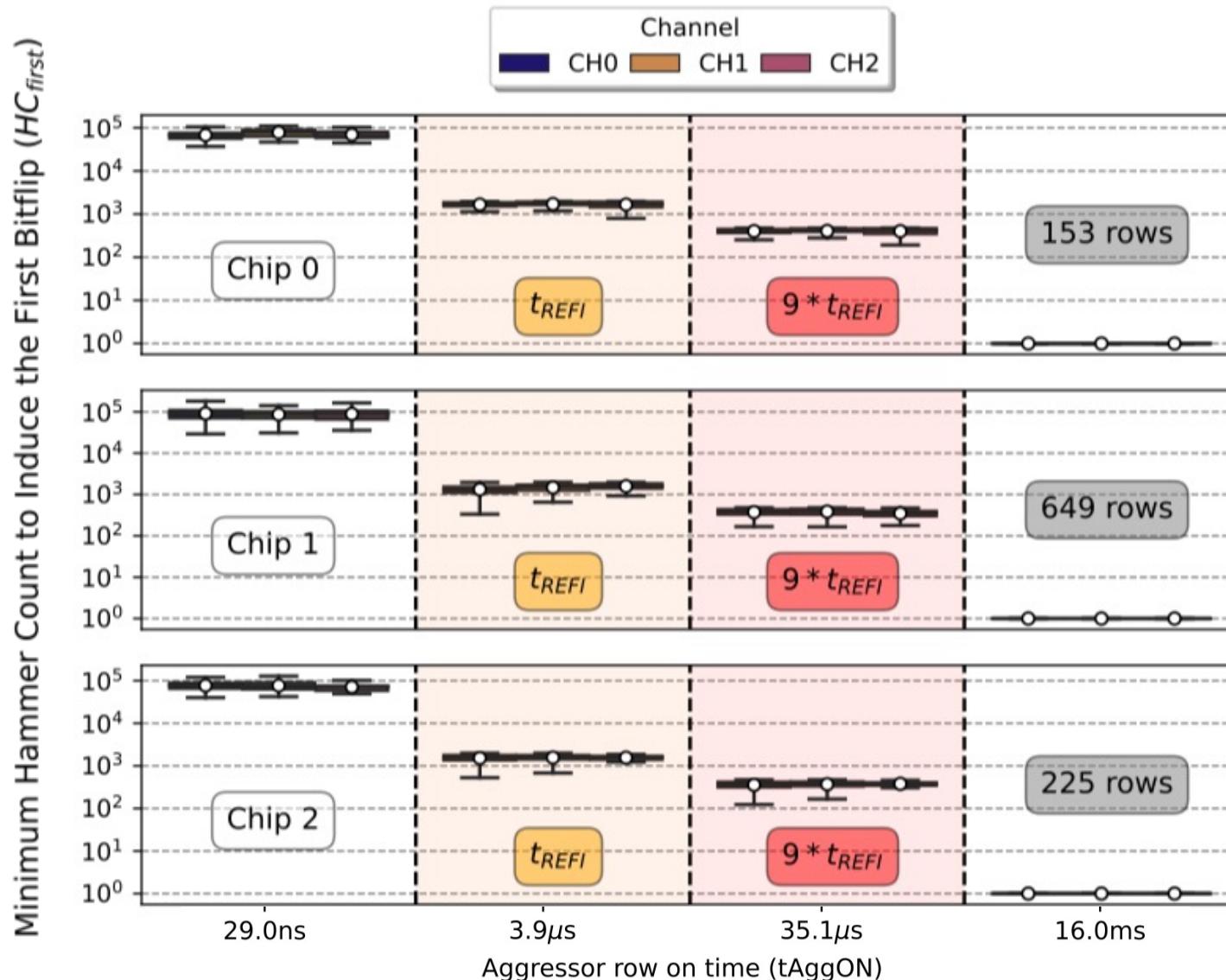
# RowHammer's Sensitivity to Hammer Count (II)



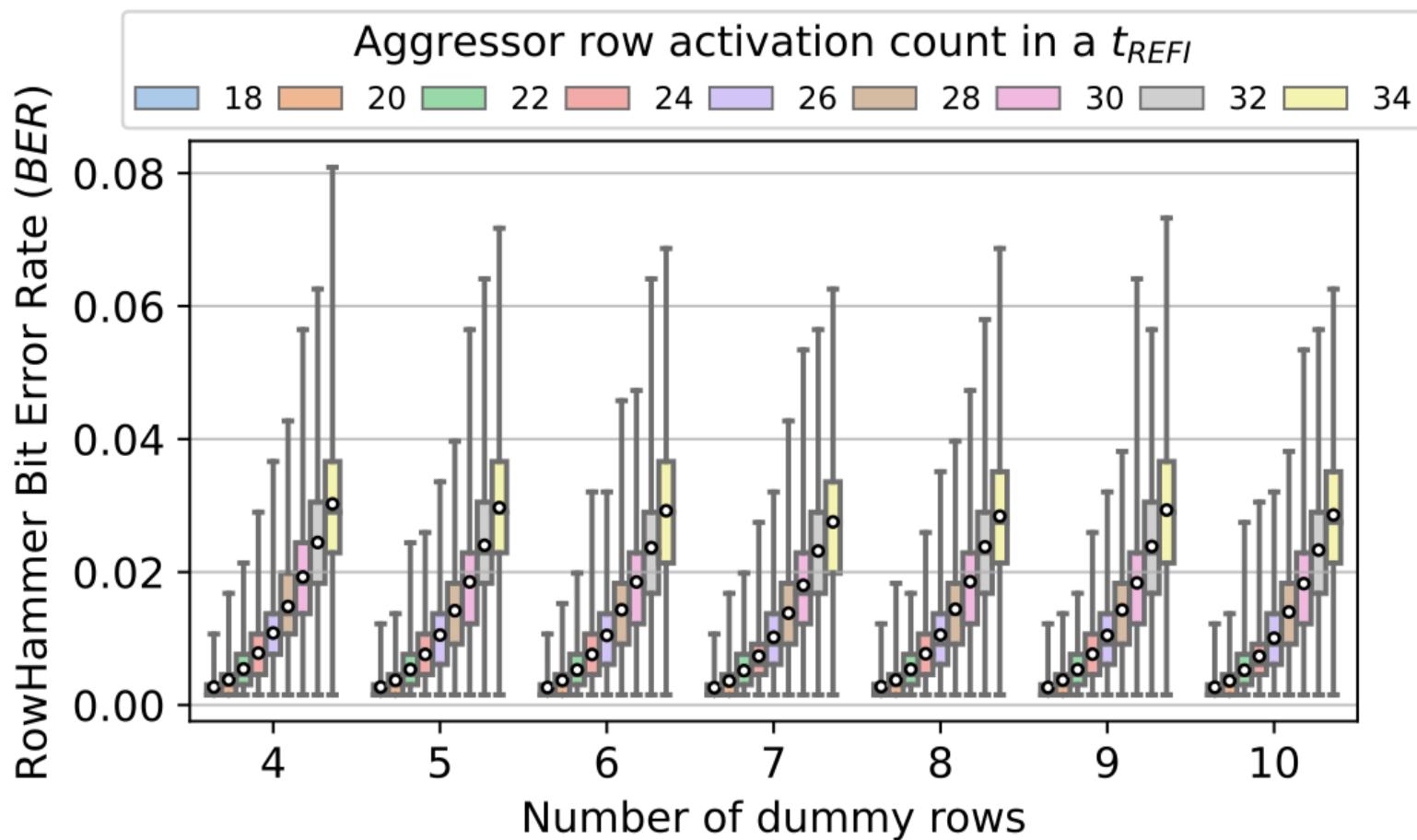
# RowPress BER Across Channels



# RowPress HC<sub>first</sub> Across Channels



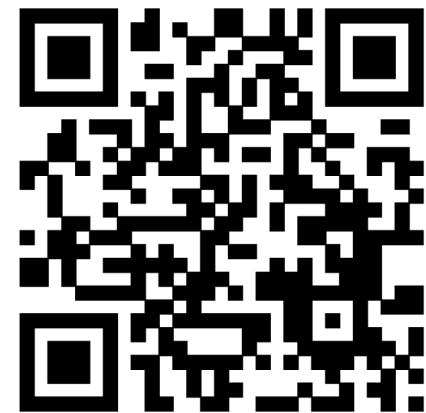
# Attack Patterns to Bypass Undisclosed TRR



# An Experimental Analysis of RowHammer in HBM2 DRAM Chips

**Link/QR code to full paper**

<https://arxiv.org/pdf/2305.17918.pdf>



Ataberk Olgun Majd Osseiran

A. Giray Yağlıkçı Yahya Can Tuğrul Haocong Luo Steve Rhyner

Behzad Salami Juan Gomez Luna Onur Mutlu

**ETH** zürich

**SAFARI**



AMERICAN  
UNIVERSITY  
OF BEIRUT

# Read Disturbance in HBM Chips

---

Ataberk Olgun, Majd Osserian, A. Giray Yağlıkçı, Yahya Can Tugrul, Haocong Luo, Steve Rhyner, Behzad Salami, Juan Gomez-Luna, and Onur Mutlu,

**"An Experimental Analysis of RowHammer in HBM2 DRAM Chips"**

*Proceedings of the 53nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Disrupt Track (**DSN Disrupt**), Porto, Portugal, June 2023.*

[arXiv version]

[Slides (pptx) (pdf)]

[Talk Video (24 minutes, including Q&A)]

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- Understanding Read Disturbance in DRAM Chips
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - **BlockHammer [Yaglikci+ HPCA'21]**
  - ABACuS [Olgun+ USENIX Security'24]

# BlockHammer

---

A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu, **"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," *HPCA*, 2021.**

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

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

[[Intel Hardware Security Academic Awards Short Talk Slides \(pptx\)](#) ([pdf](#))]

[[Talk Video](#) (22 minutes)]

[[Short Talk Video](#) (7 minutes)]

[[Intel Hardware Security Academic Awards Short Talk Video](#) (2 minutes)]

[[BlockHammer Source Code](#)]

*Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations).*

# *BlockHammer*

## *Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows*

**Abdullah Giray Yağlıkçı**

Minesh Patel Jeremie S. Kim Roknoddin Azizi

Ataberk Olgun Lois Orosa Hasan Hassan Jisung Park

Konstantinos Kanellopoulos Taha Shahroodi

Saugata Ghose\* Onur Mutlu

**SAFARI**

**ETH** zürich



# Executive Summary

- **Motivation:** RowHammer is a worsening DRAM reliability and security problem
- **Problem:** Mitigation mechanisms have limited support for current/future chips
  - **Scalability** with worsening RowHammer vulnerability
  - **Compatibility** with commodity DRAM chips
- **Goal:** Efficiently and scalably prevent RowHammer bit-flips without knowledge of or modifications to DRAM internals
- **Key Idea:** Selectively throttle memory accesses that may cause RowHammer bit-flips
- **Mechanism:** BlockHammer
  - **Tracks** activation rates of all rows by using area-efficient Bloom filters
  - **Throttles** row activations that could cause RowHammer bit flips
  - **Identifies and throttles** threads that perform RowHammer attacks
- **Scalability with Worsening RowHammer Vulnerability:**
  - **Competitive** with state-of-the-art mechanisms **when there is no attack**
  - **Superior** performance and DRAM energy **when a RowHammer attack is present**
- **Compatibility with Commodity DRAM Chips:**
  - **No proprietary information** of DRAM internals
  - **No modifications** to DRAM circuitry

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

AttackThrottler

Evaluation

Conclusion

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

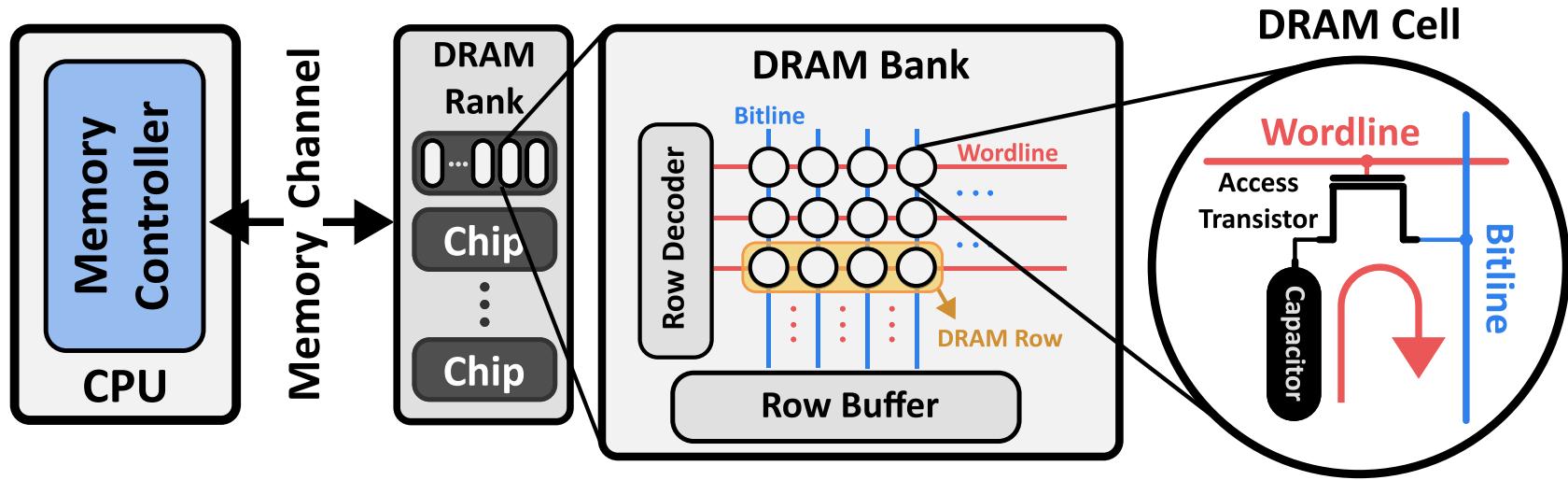
RowBlocker

AttackThrottler

Evaluation

Conclusion

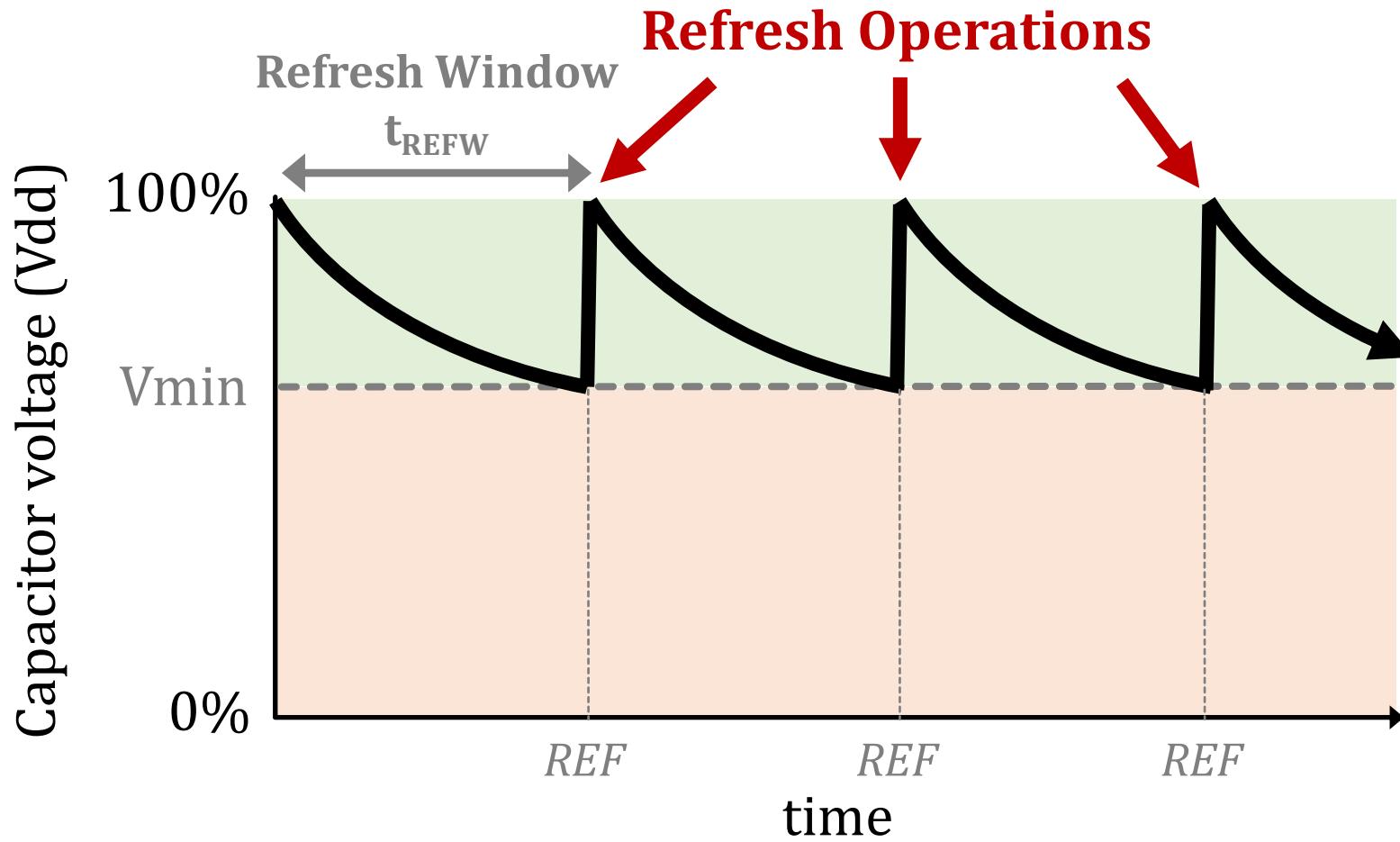
# Organizing and Accessing DRAM Cells



A DRAM cell consists of a **capacitor** and an **access transistor**

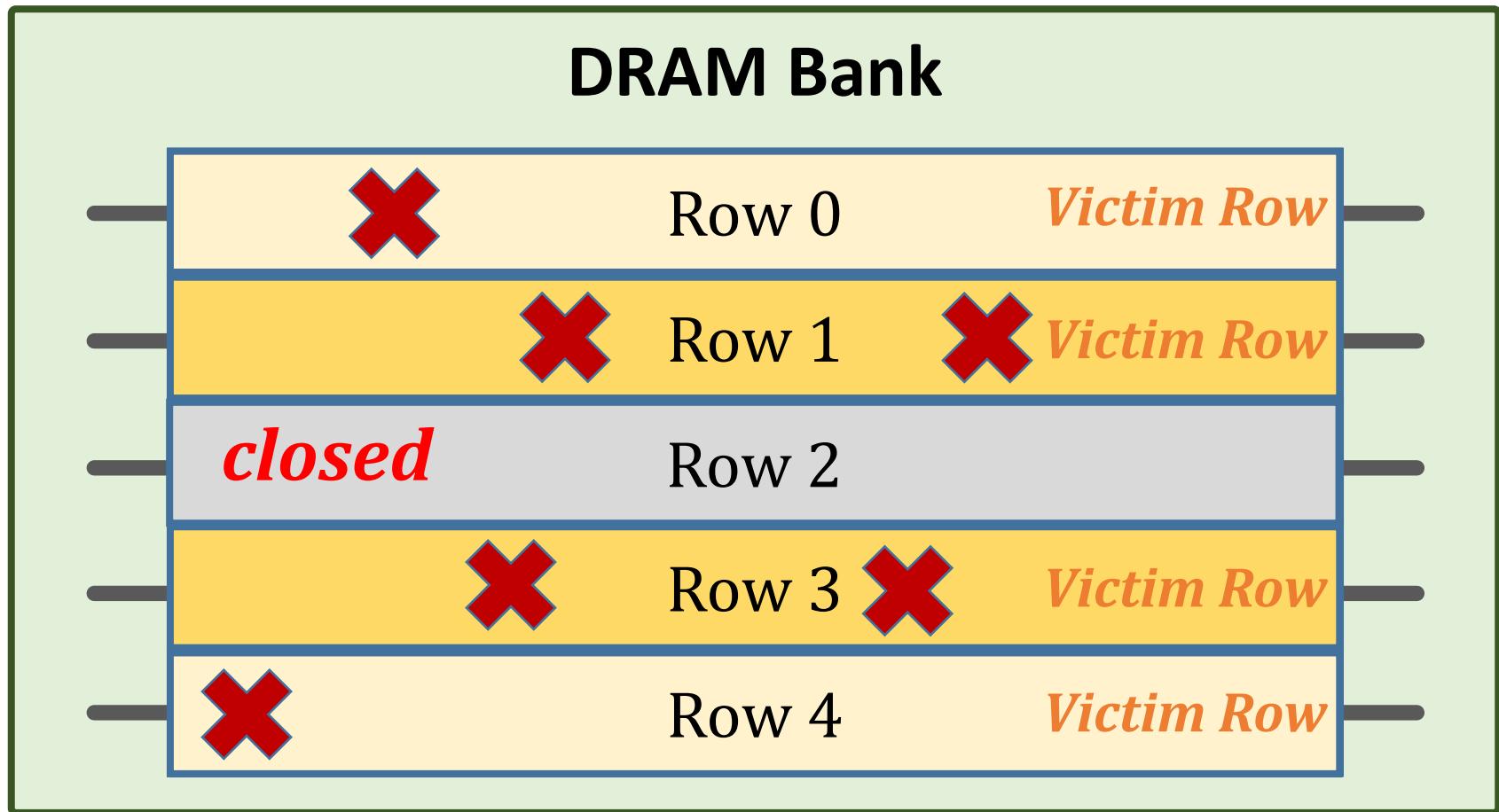
A row needs to be **activated** to access its content

# DRAM Refresh



Periodic **refresh operations** preserve stored data

# The RowHammer Phenomenon



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

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

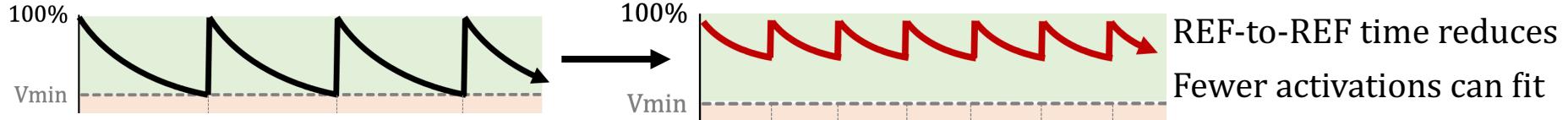
AttackThrottler

Evaluation

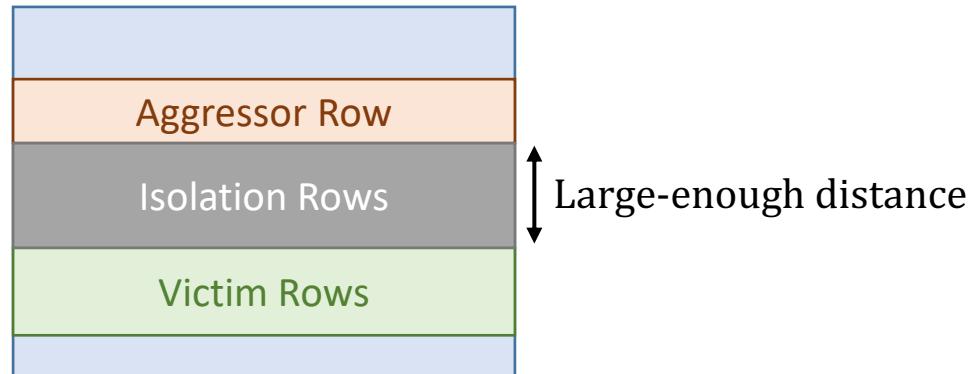
Conclusion

# RowHammer Mitigation Approaches

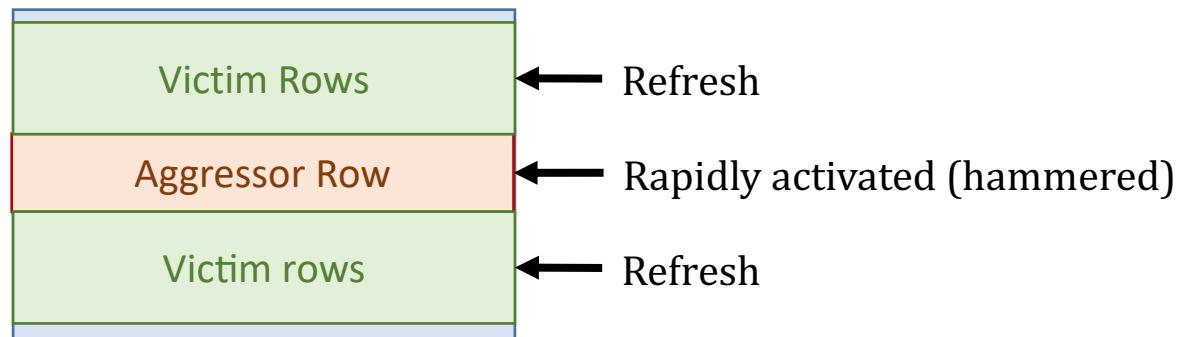
- Increased refresh rate



- Physical isolation

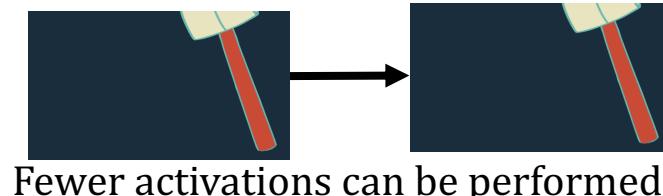


- Reactive refresh



- Proactive throttling

**SAFARI**



# Two Key Challenges

1

## Scalability

with worsening RowHammer vulnerability

2

## Compatibility

with commodity DRAM chips

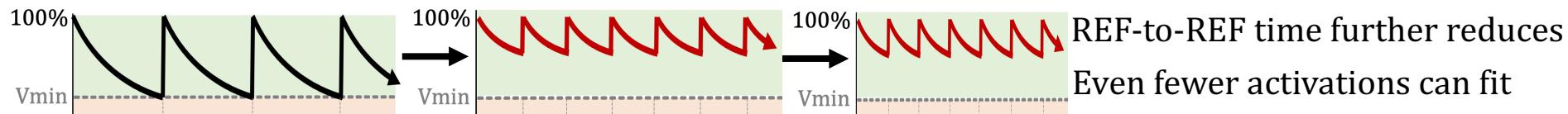
# Scalability with Worsening RowHammer Vulnerability

- DRAM chips are more vulnerable to RowHammer today
- RowHammer bit-flips occur at much lower activation counts  
**(more than an order of magnitude decrease):**
  - 139.2K [Y. Kim+, ISCA 2014]
  - 9.6K [J. S. Kim+, ISCA 2020]
- RowHammer blast radius has increased by **33%**:
  - 9 rows [Y. Kim+, ISCA 2014]
  - 12 rows [J. S. Kim+, ISCA 2020]
- In-DRAM mitigation mechanisms are ineffective [Frigo+, S&P 2020]

**RowHammer is a more serious problem than ever**

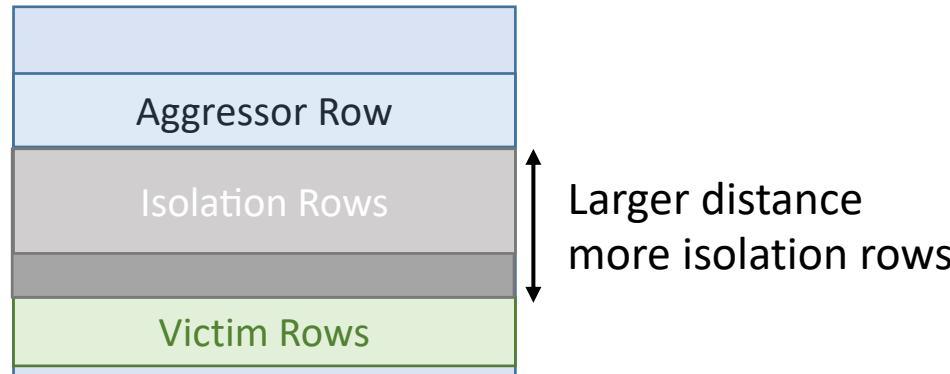
# Mitigation Approaches with Worsening RowHammer Vulnerability

- Increased refresh rate



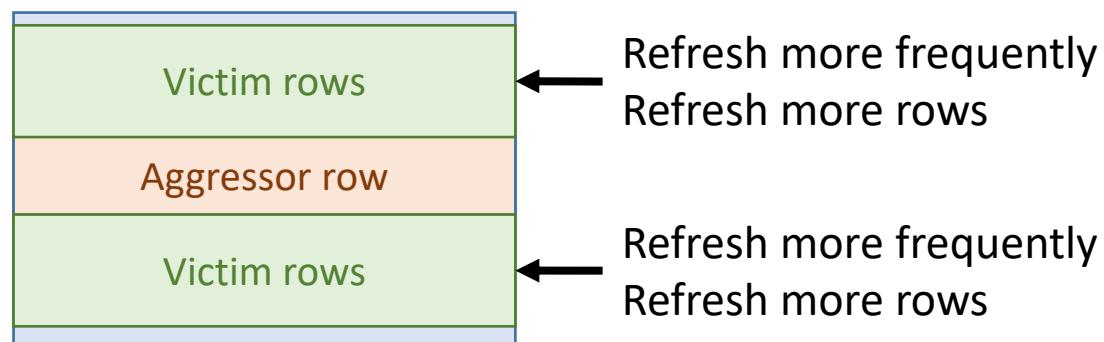
REF-to-REF time further reduces  
Even fewer activations can fit

- Physical isolation



Larger distance  
more isolation rows

- Reactive refresh



Refresh more frequently  
Refresh more rows

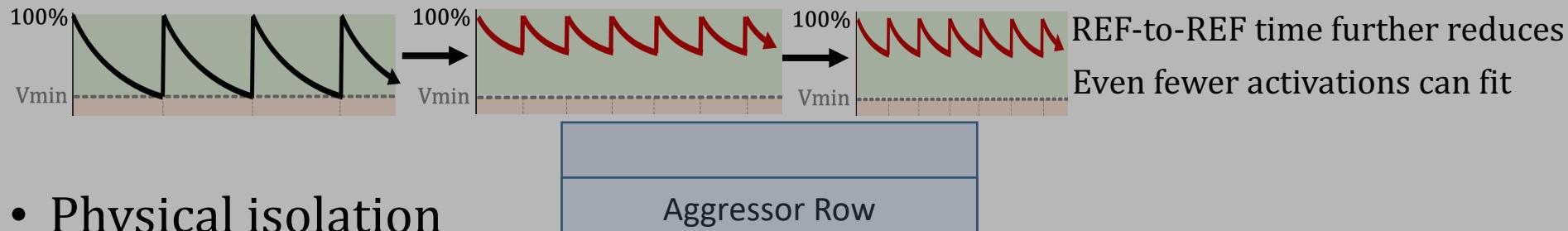
Refresh more frequently  
Refresh more rows

- Proactive throttling



# Mitigation Approaches with Worsening RowHammer Vulnerability

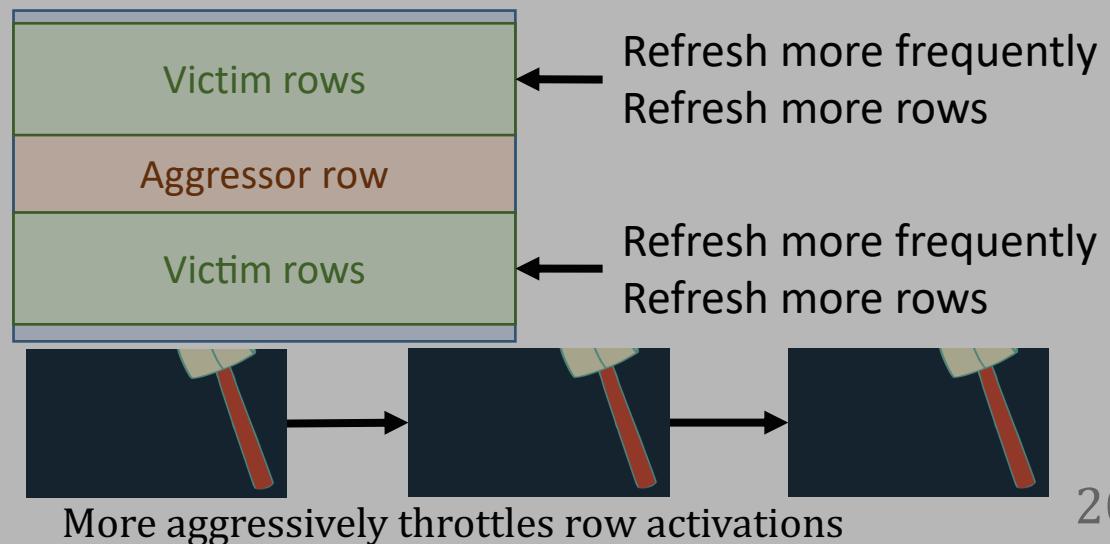
- Increased refresh rate



- Physical isolation

**Mitigation mechanisms face the challenge of scalability with worsening RowHammer**

- Reactive refresh



- Proactive throttling

# Two Key Challenges

1

## Scalability

with worsening RowHammer vulnerability

2

## Compatibility

with commodity DRAM chips

# Compatibility with Commodity DRAM Chips

Visible within  
the Processor

Application  
Level

Virtual Memory Address

System  
Level

Physical Memory Address

Memory  
Controller

DRAM Bus Addresses

(Channel, Rank, Bank Group, Bank, Row, Col)

In-DRAM  
Mapping

Physical Rows and Columns

# Compatibility with Commodity DRAM Chips

Vendors apply in-DRAM mapping for two reasons:

- **Design Optimizations:** By simplifying DRAM circuitry to provide better density, performance, and power
- **Yield Improvement:** By mapping faulty rows and columns to redundant ones
- In-DRAM mapping scheme includes insights into **chip design** and **manufacturing quality**

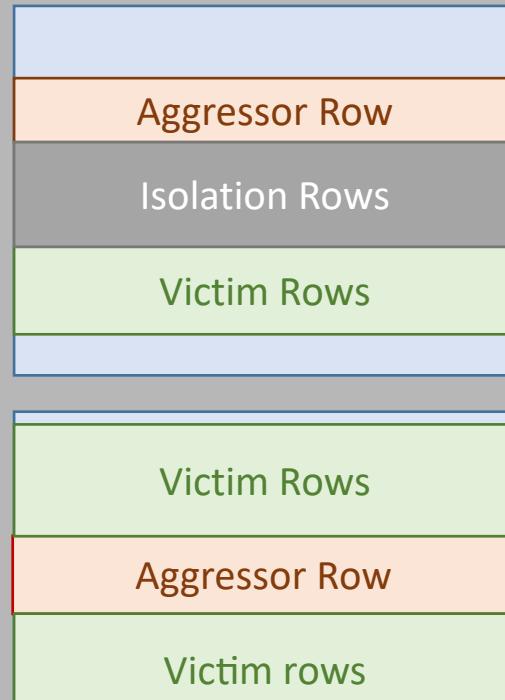
**In-DRAM mapping is proprietary information**

# RowHammer Mitigation Approaches

- Increased refresh rate



- Physical isolation



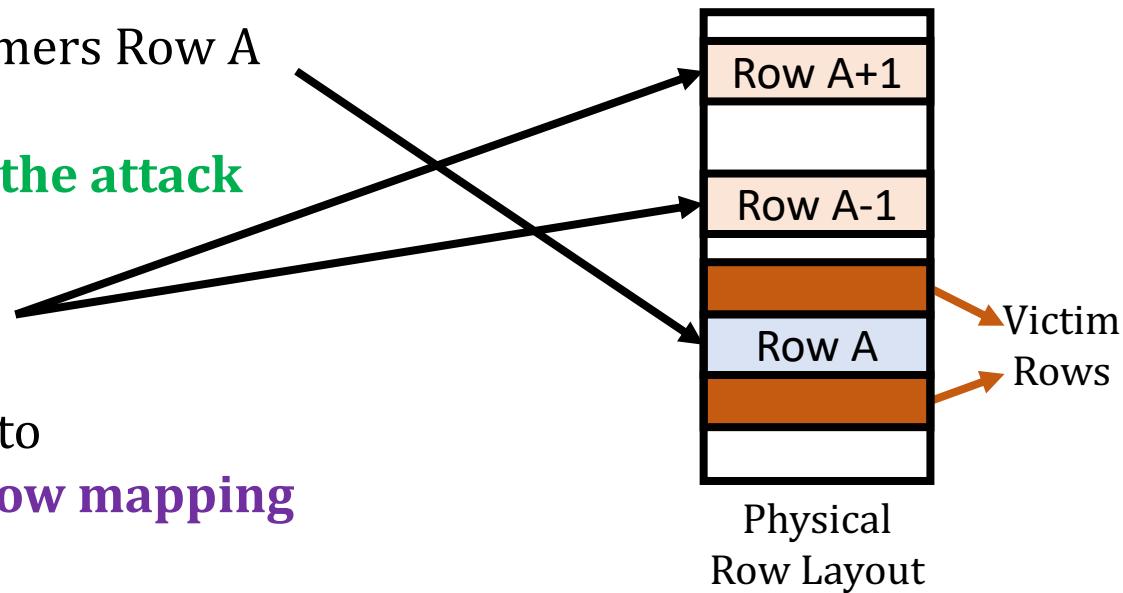
- Reactive refresh

Identifying **victim** and **isolation** rows requires  
**proprietary** knowledge of ***in-DRAM mapping***

# Major Problem with Past RowHammer Mitigations



- A **RowHammer attack** hammers Row A
- Existing mechanisms **detect the attack**
- Refresh rows **A+1** and **A-1**
- Bit flips **still may occur** due to **unknown DRAM-internal row mapping**



Existing **RowHammer mitigation mechanisms**  
need to know **proprietary DRAM-internal row address mapping**

# Our Goal

To prevent RowHammer **efficiently and scalably**  
*without* knowledge of or modifications to DRAM internals

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

AttackThrottler

Evaluation

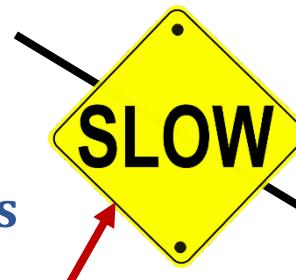
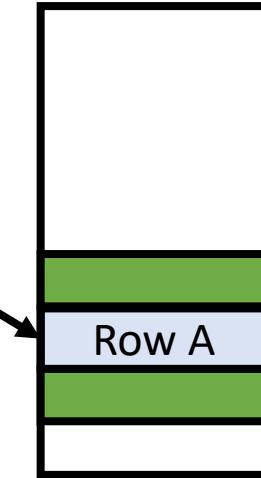
Conclusion

# BlockHammer

## Key Idea

Selectively throttle memory accesses  
that may cause RowHammer bit-flips

# BlockHammer: Practical Throttling-based Mechanism



- A RowHammer attack hammers Row A
- BlockHammer detects a RowHammer attack using **area-efficient Bloom filters**
- BlockHammer **selectively throttles accesses** from within **the memory controller**
- Bit flips **do not** occur
- BlockHammer can *optionally inform the system software* about the attack

**BlockHammer is compatible with commodity DRAM chips**  
**No need for proprietary info or modifications to DRAM chips**

# BlockHammer

## Overview of Approach

### RowBlocker

Tracks row activation rates using area-efficient Bloom filters

Blacklists rows that are activated at a high rate

Throttles activations targeting a blacklisted row

**No row can be activated at a high enough rate to induce bit-flips**

### AttackThrottler

Identifies threads that perform a RowHammer attack

Reduces memory bandwidth usage of identified threads

Greatly reduces the **performance degradation** and **energy wastage** a RowHammer attack inflicts on a system

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

AttackThrottler

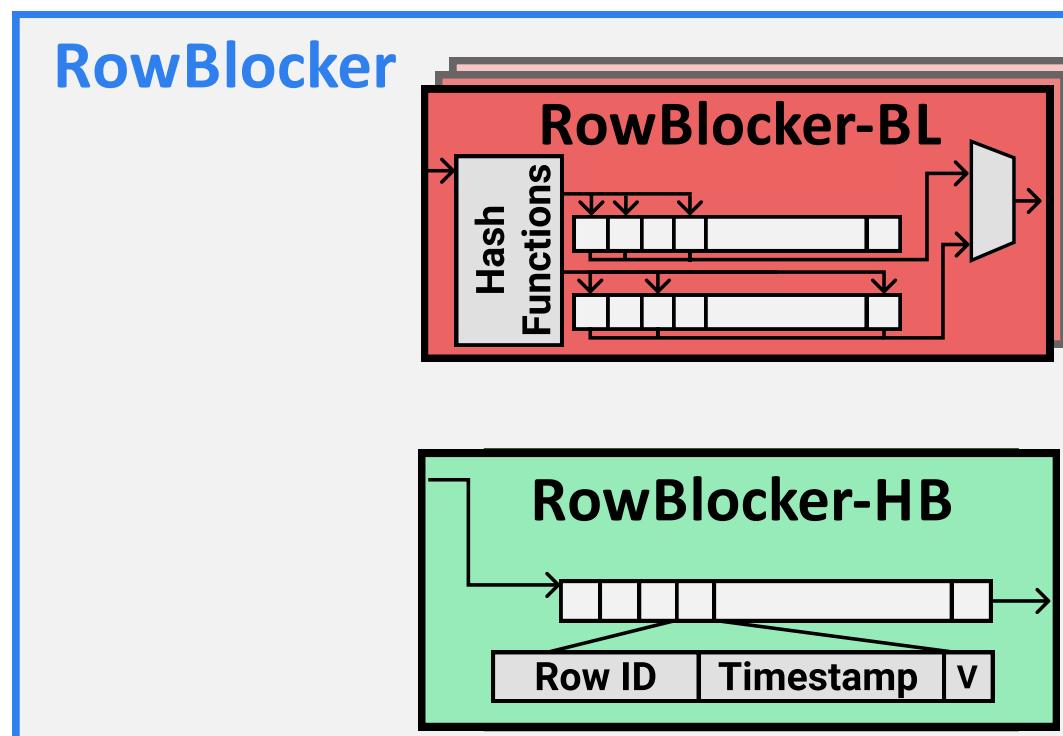
Evaluation

Conclusion

# RowBlocker

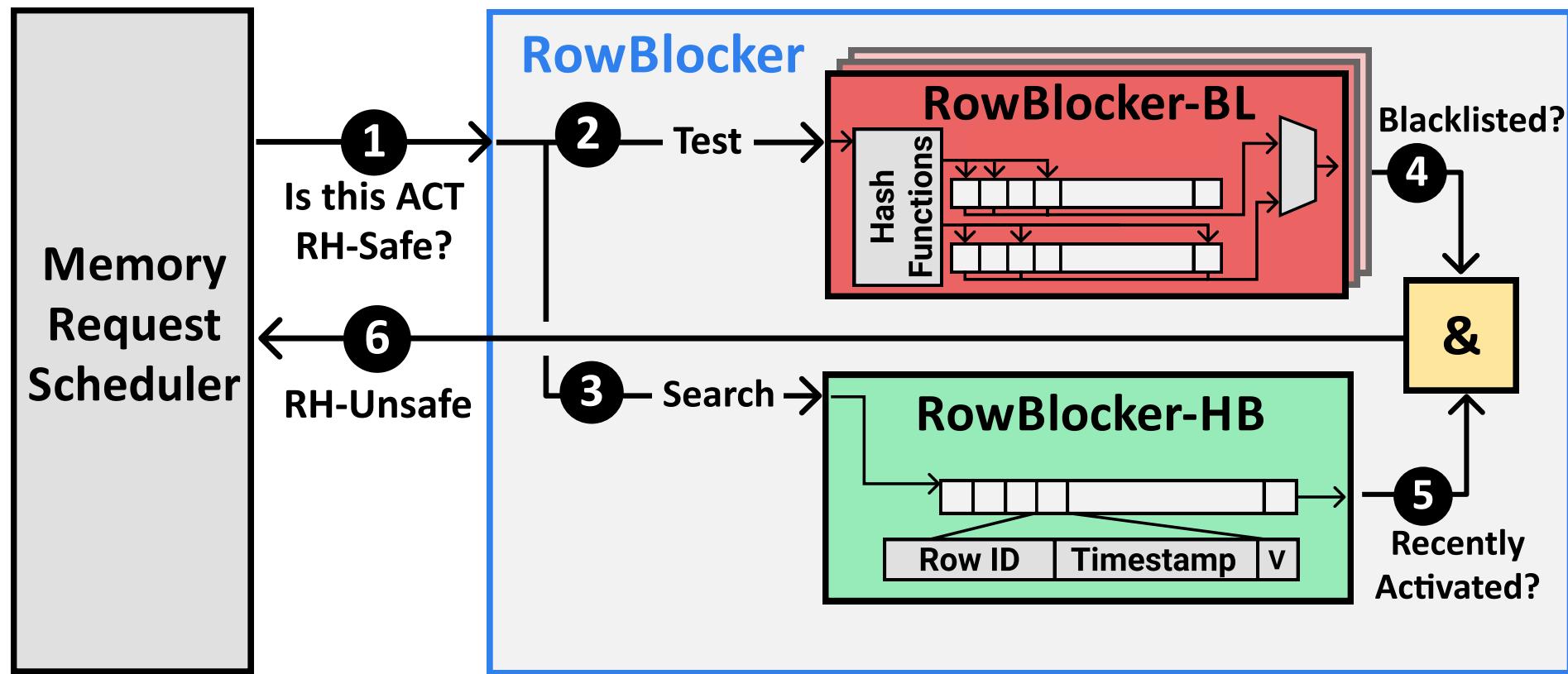
- Modifies the memory request scheduler **to throttle** row activations
- **Blacklists** rows with a high activation rate and **delays** subsequent activations targeting blacklisted rows

Memory  
Request  
Scheduler



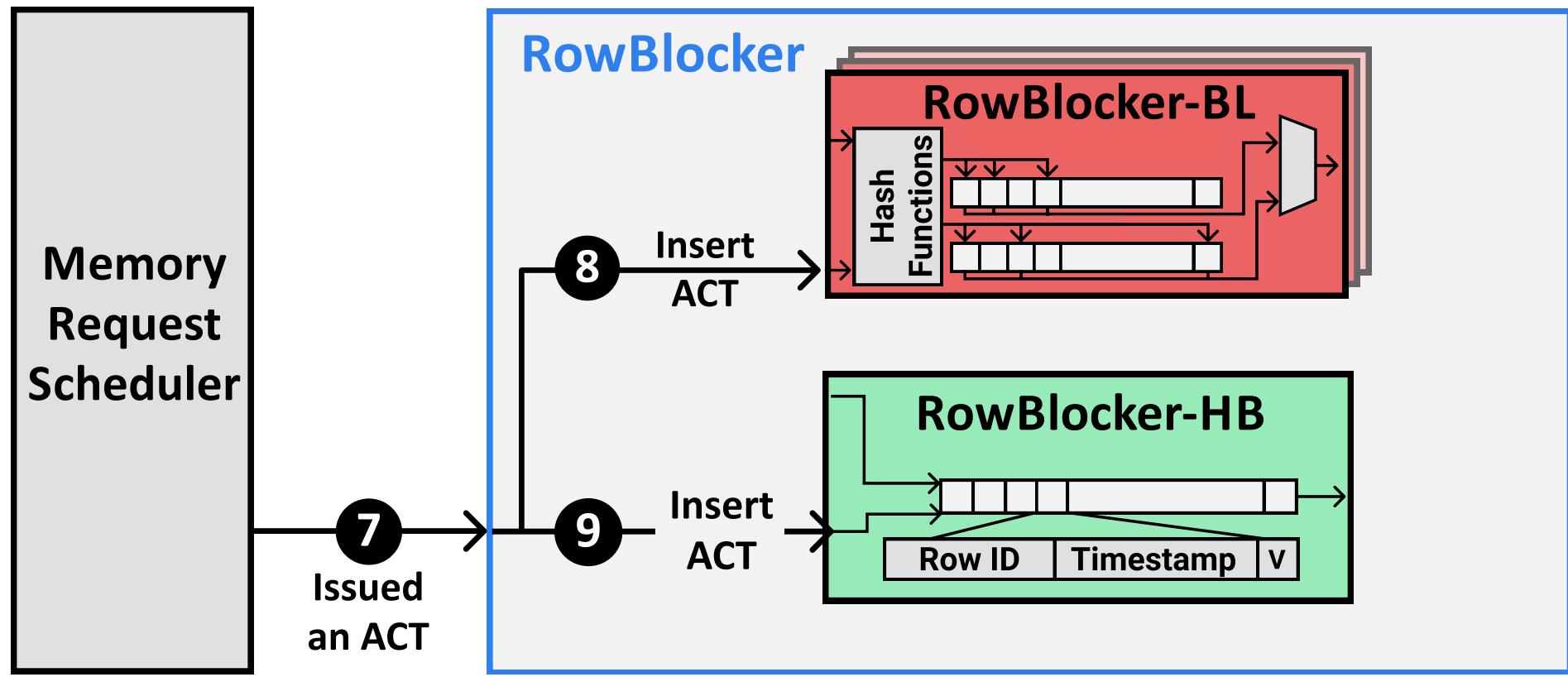
# RowBlocker

- Blocks a row activation if the row is **both** blacklisted and recently activated



# RowBlocker

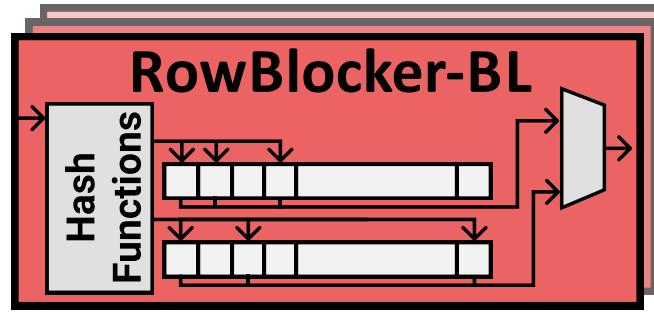
- When a row activation is performed, both **RowBlocker-BL** and **RowBlocker-HB** are updated with the row activation information



# RowBlocker-BL

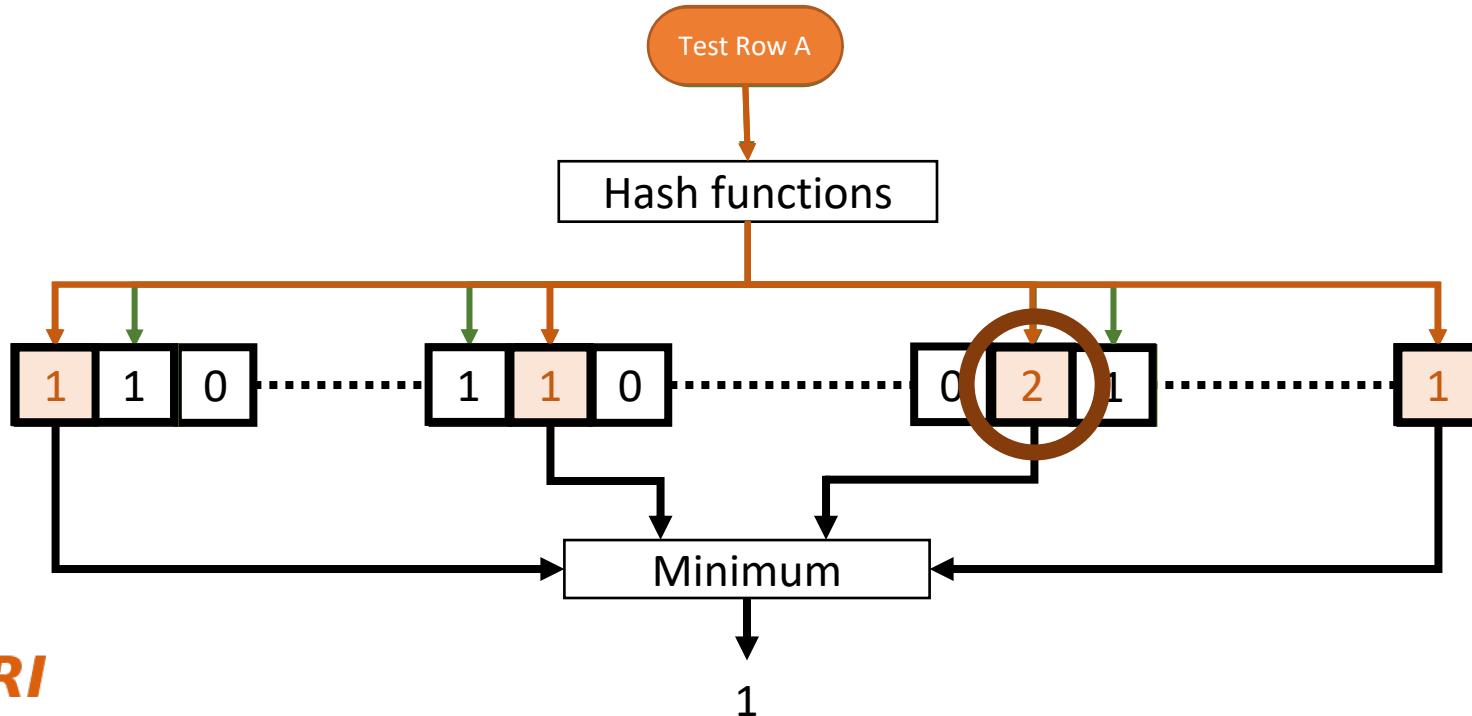
## Blacklisting Logic

- **Blacklists** a row when the row's activation count in a time window exceeds a threshold
- Employs **two counting Bloom filters** for area-efficient activation rate tracking



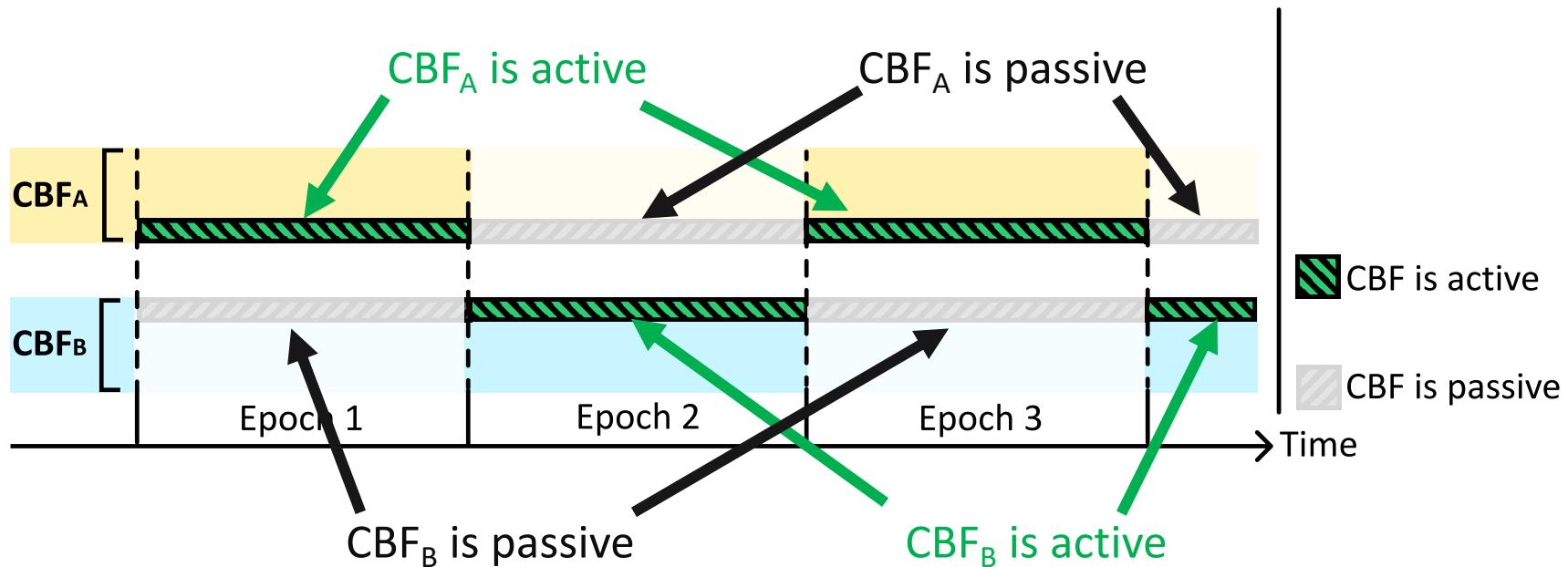
# Counting Bloom Filters

- Blacklisting logic counts activations using counting Bloom filters
- A row's activation count
  - can be observed more than it is (**false positive**)
  - cannot be observed less than it is (**no false negative**)
- To avoid saturating counters, we use a time-interleaving approach



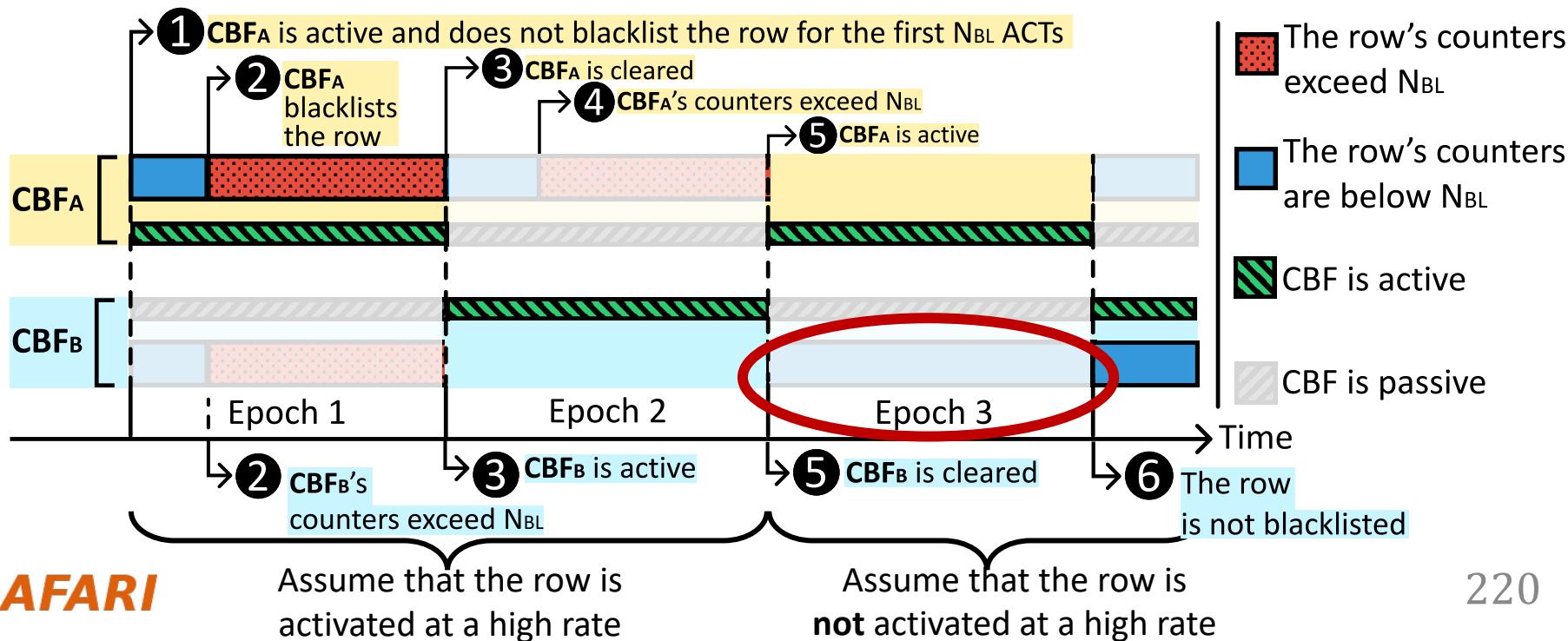
# RowBlocker-BL Blacklisting Logic

- Blacklisting logic employs **two counting Bloom filters**
- A new row activation is **inserted in both filters**
- Only one filter (**active filter**) responds to test queries
- The active filter changes at every epoch



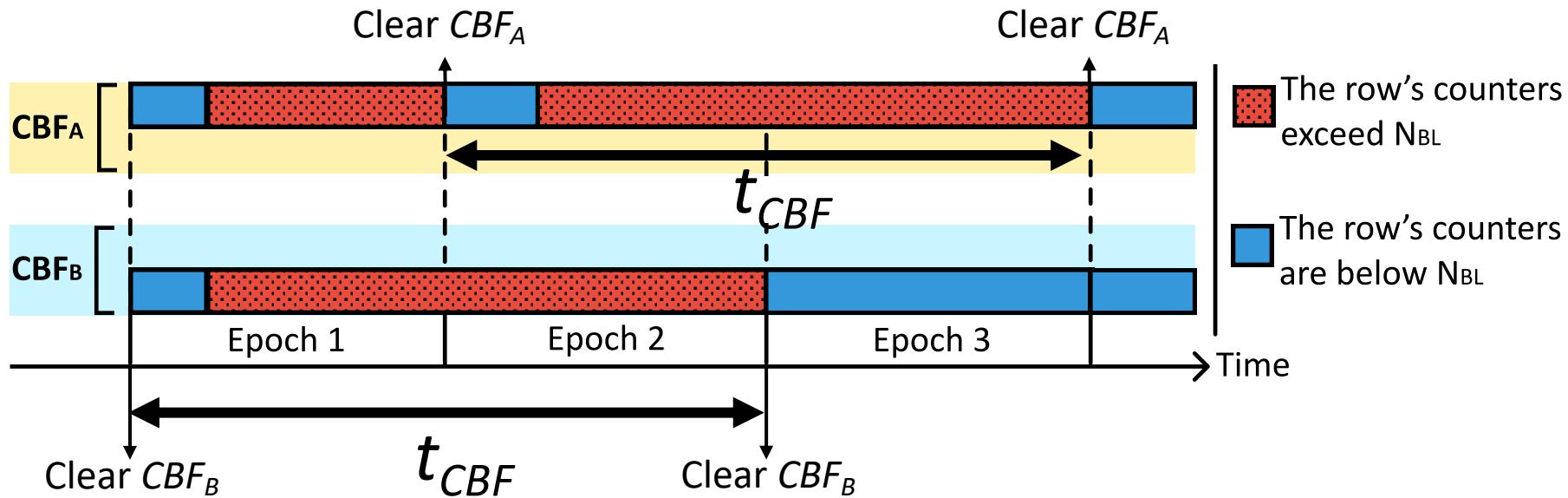
# RowBlocker-BL Blacklisting Logic

- Blacklisting logic employs **two counting Bloom filters**
- A new row activation is **inserted in both filters**
- Only one filter (**active filter**) responds to test queries
- The active filter changes at every epoch
- Blacklists a row if its activation count reaches the **blacklisting threshold ( $N_{BL}$ )**



# Limiting the Row Activation Rate

- The activation rate is **RowHammer-safe** if it is smaller than or equal to **RowHammer threshold ( $N_{RH}$ )** activations in a **refresh window ( $t_{REFW}$ )**
- RowBlocker limits the **activation count ( $N_{CBF}$ )** in a **CBF's lifetime ( $t_{CBF}$ )**  
*Activation Rate in a  $t_{CBF} \leq N_{RH}$  activations in a refresh window ( $t_{REFW}$ )*

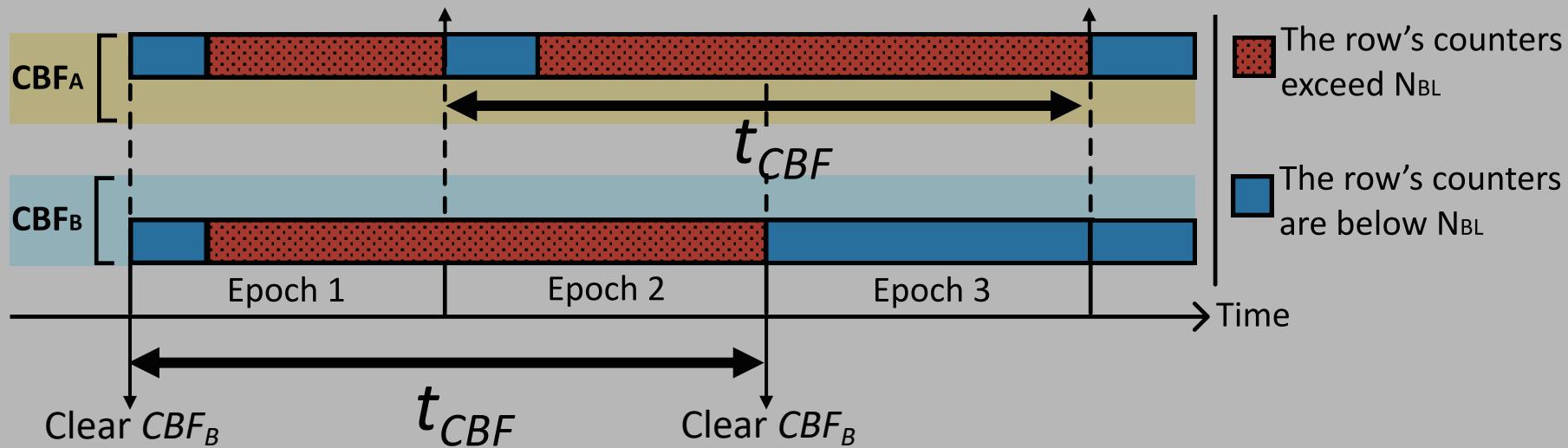


# Limiting the Row Activation Rate

- The activation rate is **RowHammer-safe** if it is smaller than or equal to **RowHammer threshold ( $N_{RH}$ )** activations in a **refresh window ( $t_{REFW}$ )**
- RowBlocker limits the **activation count ( $N_{CBF}$ )** in a **CBF's lifetime ( $t_{CBF}$ )**  
*Activation Rate in a  $t_{CBF} \leq N_{RH}$  activations in a refresh window ( $t_{REFW}$ )*

## RowHammer Safety Constraint

$$N_{CBF}/t_{CBF} \leq N_{RH}/t_{REFW}$$

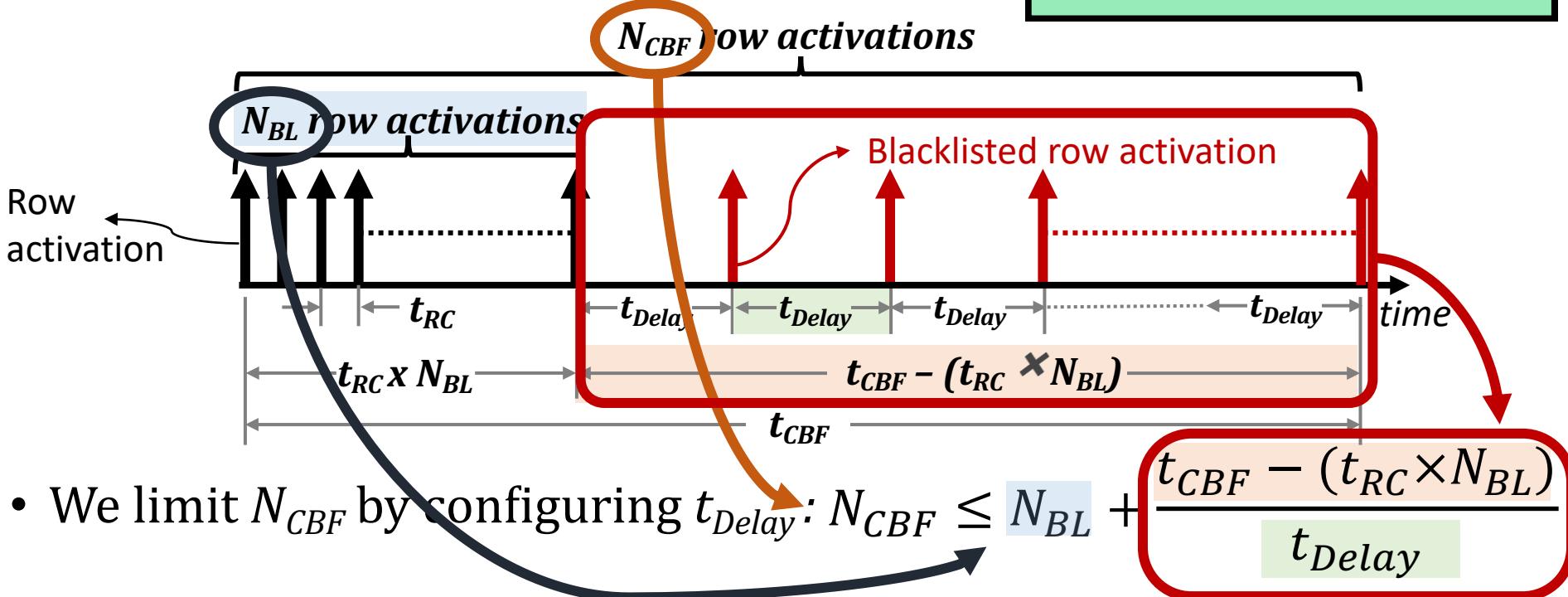
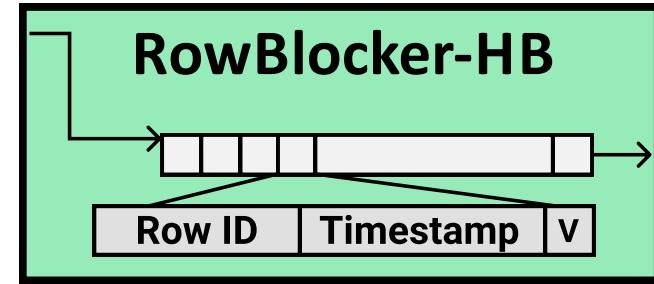


# RowBlocker-HB

## Limiting the Row Activation Rate

- Ensures that all rows experience a RowHammer-safe activation rate

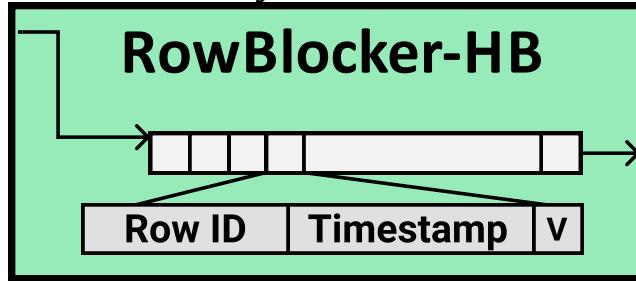
$$N_{CBF}/t_{CBF} \leq N_{RH}/t_{REFW}$$



# RowBlocker-HB

## Delaying Row Activations

- RowBlocker-HB ensures no subsequent blacklisted row activation is performed sooner than  $t_{Delay}$



- RowBlocker-HB implements a history buffer for row activations that can fit in a  $t_{Delay}$  time window
- A blacklisted row activation is blocked as long as a valid activation record of the row exists in the history buffer

No row can be activated at a high enough rate to induce bit-flips

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

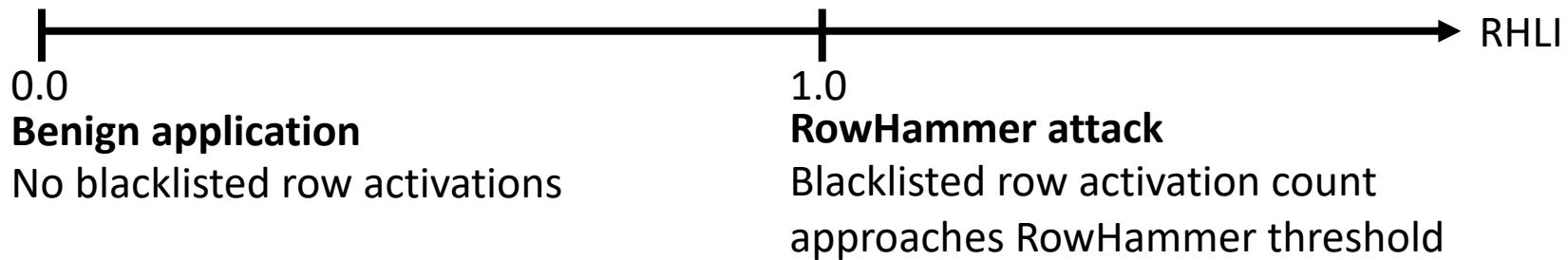
AttackThrottler

Evaluation

Conclusion

# AttackThrottler

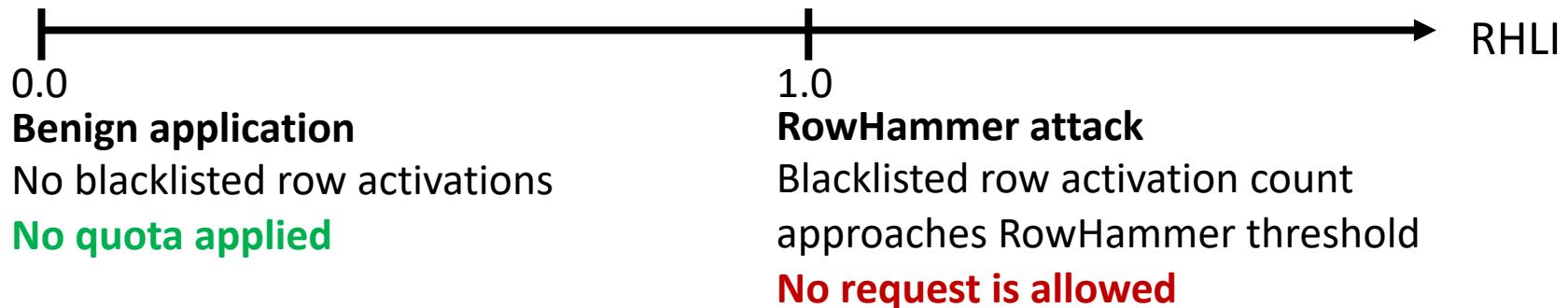
- Tackles a RowHammer attack's **performance degradation** and **energy wastage** on a system
- A RowHammer attack intrinsically keeps activating **blacklisted rows**
- **RowHammer Likelihood Index (RHLI):** Number of activations that target blacklisted rows (normalized to maximum possible activation count)



**RHLI is larger** when the thread's access pattern  
is more **similar to a RowHammer attack**

# AttackThrottler

- Applies a smaller quota to a thread's in-flight request count as RHLI increases



- Reduces a RowHammer attack's **memory bandwidth consumption**, enabling a larger memory bandwidth for **concurrent benign applications**

**Greatly reduces the performance degradation and energy wastage a RowHammer attack inflicts on a system**

- RHLI can also be used as a **RowHammer attack indicator** by the system software

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

AttackThrottler

Evaluation

Conclusion

# Evaluation

## BlockHammer's Hardware Complexity

- We analyze **six state-of-the-art mechanisms** and **BlockHammer**
- We calculate **area**, **access energy**, and **static power** consumption\*

Mitigation Mechanism	SRAM KB	CAM KB	Area mm <sup>2</sup>	%CPU	Access Energy pJ	Static Power mW
BlockHammer	51.48	1.73	0.14	0.06	20.30	22.27
PARA [73]	-	-	<0.01	-	-	-
ProHIT [137]	-	0.22	<0.01	<0.01	3.67	0.14
MRLoc [161]	-	0.47	<0.01	<0.01	4.44	0.21
CBT [132]	16.00	8.50	0.20	0.08	9.13	35.55
TWiCe [84]	23.10	14.02	0.15	0.06	7.99	21.28
Graphene [113]	-	5.22	0.04	0.02	40.67	3.11

BlockHammer is **low cost** and **competitive**  
with state-of-the-art mechanisms

\*Assuming a high-end 28-core Intel Xeon processor system with 4-channel single-rank DDR4 DIMMs  
with a RowHammer threshold (NRH) of 32K

# Evaluation

## BlockHammer's Hardware Complexity

Mitigation Mechanism	SRAM KB	CAM KB	Area mm <sup>2</sup>	%CPU	Access Energy pJ	Static Power mW
$N_{RH}=32K$	51.48	1.73	0.14	0.06	20.30	22.27
PARA [73]	-	-	<0.01	-	-	-
ProHIT [137]	-	0.22	<0.01	<0.01	3.67	0.12
MRLoc [161]	-	0.47	<0.01	<0.01	4.4	0.12
CBT [132]	16.00	8.50	0.20	0.08	9.13	35.55
TWiCe [84]	23.10	14.02	0.15	0.06	7.99	21.28
Graphene [113]	-	5.22	0.04	0.02	40.67	3.11
$N_{RH}=1K$	441.33	55.58	1.57	0.64	99.64	220.99
PARA [73]	-	-	<0.01	-	-	-
ProHIT [137]	x	x	x	x	x	x
MRLoc [161]	x	x	x	x	x	x
CBT [132]	512.00	272.00	3.95	20x	127.93	535.50
TWiCe [84]	738.32	448.27	5.17	35x	124.79	631.98
Graphene [113]	-	166.03	1.14	23x	917.55	93.96

The diagram illustrates the performance scaling of various mechanisms. For  $N_{RH}=32K$ , BlockHammer shows significantly lower resource usage and energy consumption compared to other mechanisms. For  $N_{RH}=1K$ , while other mechanisms show higher resource usage, BlockHammer maintains its efficiency, demonstrating superior scaling.

BlockHammer's hardware complexity **scales more efficiently** than state-of-the-art mechanisms

# Evaluation

## Performance and DRAM Energy

- Cycle-level simulations using **Ramulator** and **DRAMPower**
- System Configuration:

<b>Processor</b>	3.2 GHz, {1,8} core, 4-wide issue, 128-entry instr. window
<b>LLC</b>	64-byte cacheline, 8-way set-associative, {2,16} MB
<b>Memory scheduler</b>	FR-FCFS
<b>Address mapping</b>	Minimalistic Open Pages
<b>DRAM</b>	DDR4 1 channel, 1 rank, 4 bank group, 4 banks per bank group
<b>RowHammer Threshold</b>	32K

- Single-Core Benign Workloads:

- 22 SPEC CPU 2006
- 4 YCSB Disk I/O
- 2 Network Accelerator Traces
- 2 Bulk Data Copy with Non-Temporal Hint (movnti)

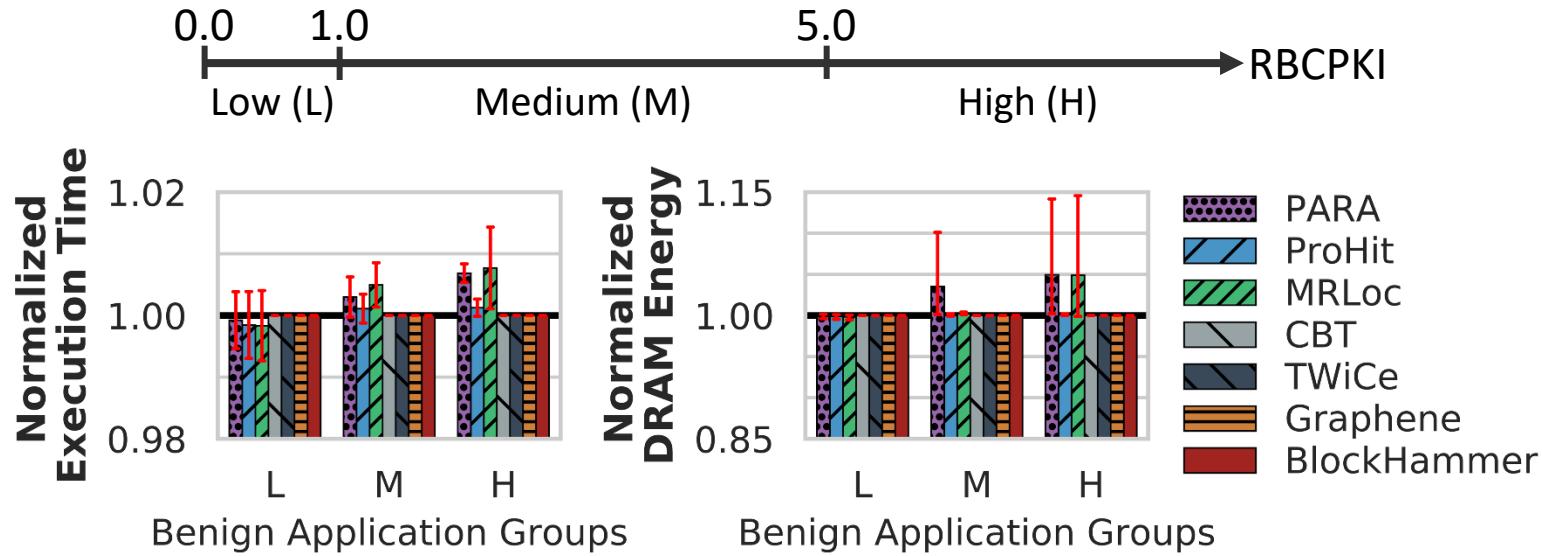
- Randomly Chosen Multiprogrammed Workloads:

- 125 workloads containing **8 benign applications**
- 125 workloads containing **7 benign applications** and **1 RowHammer attack thread**

# Evaluation

## Performance and DRAM Energy

- We classify single-core workloads into three categories based on row buffer conflicts per thousand instructions



- No application's row activation count exceeds BlockHammer's blacklisting threshold ( $N_{BL}$ )

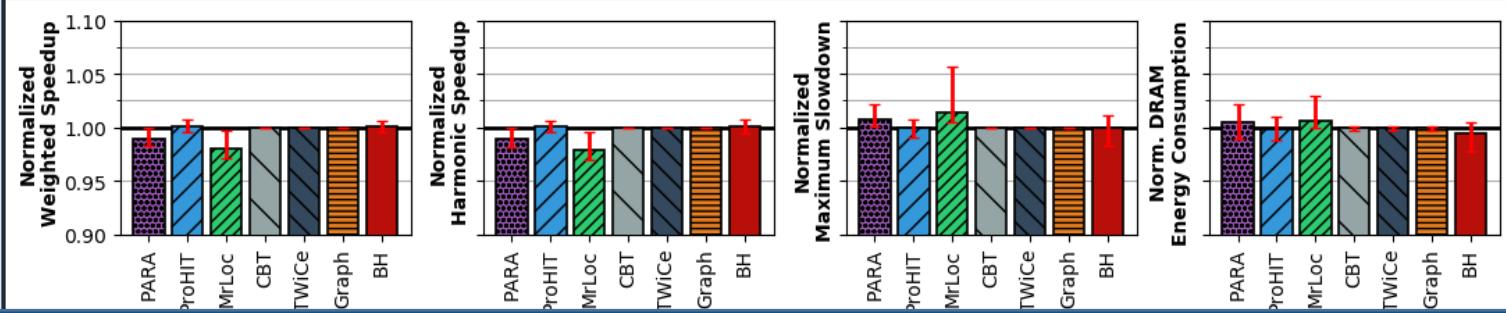
BlockHammer does not incur **performance or DRAM energy** overheads for single-core benign applications

# Evaluation

## Performance and DRAM Energy

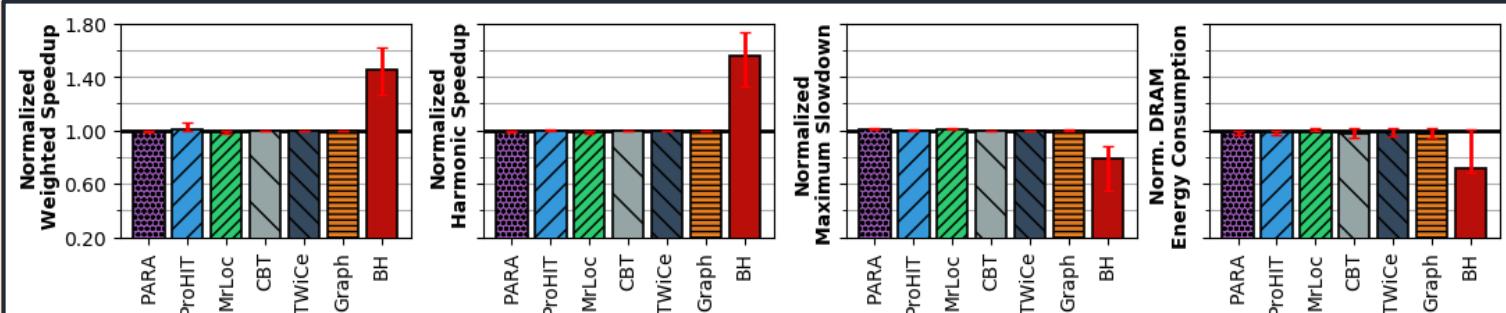
- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

No RowHammer Attack



BlockHammer introduces **very low** performance (<0.5%) and DRAM energy (<0.4%) overheads

RowHammer Attack Present



BlockHammer **significantly increases** benign application performance (by 45% on average) and **reduces** DRAM energy consumption (by 29% on average)

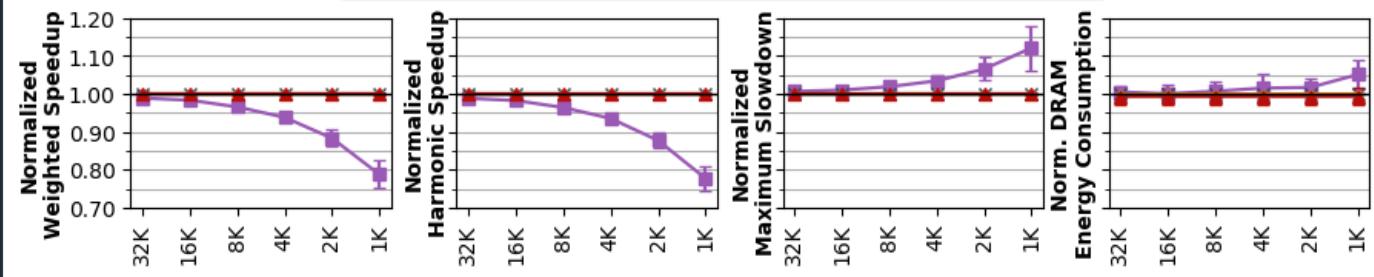
# Evaluation

## Scaling with RowHammer Vulnerability

- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

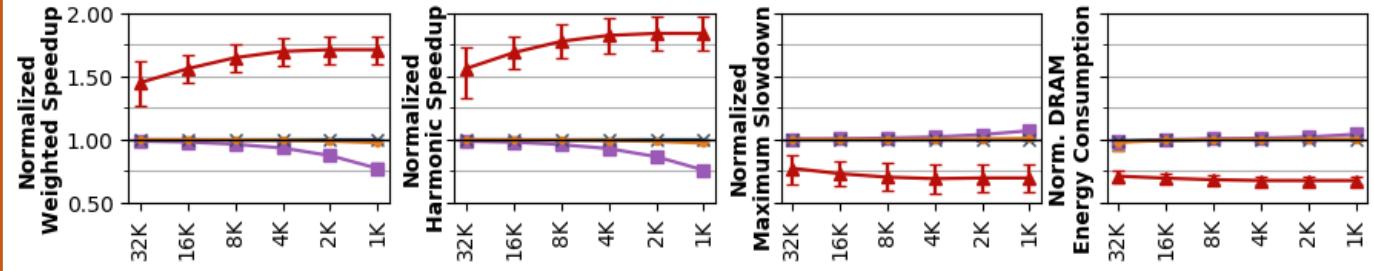


No RowHammer Attack



BlockHammer's performance and energy overheads remain **negligible (<0.6%)**

RowHammer Attack Present



BlockHammer scalably provides **much higher performance** (71% on average) and **lower energy consumption** (32% on average) than state-of-the-art mechanisms

# More in the Paper

- Security Proof
  - Mathematically represent **all possible** access patterns
  - We show that **no row can be activated high-enough times** to induce bit-flips when BlockHammer is configured correctly
- Addressing **Many-Sided Attacks**
- Evaluation of **14 mechanisms** representing **four mitigation approaches**
  - Comprehensive Protection
  - Compatibility with Commodity DRAM Chips
  - Scalability with RowHammer Vulnerability
  - Deterministic Protection

Approach	Mechanism	Comprehensive Protection	Compatible w/ Commodity DRAM Chips	Scaling with RowHammer Vulnerability	Deterministic Protection
	Increased Refresh Rate [2, 73]	✓	✓	✗	✓
Physical Isolation	CATT [14] GuardION [148] ZebRAM [78]	✗ ✗ ✗	✗ ✗ ✗	✗ ✗ ✗	✗ ✗ ✓
Reactive Refresh	ANVIL [5] PARA [73] PROHIT [137] MRLoc [161] CBT [132] TWiCe [84] Graphene [113]	✗ ✓ ✓ ✓ ✓ ✓ ✓	✗ ✗ ✗ ✗ ✗ ✗ ✗	✗ ✗ ✗ ✗ ✗ ✗ ✓	✗ ✗ ✗ ✗ ✓ ✓ ✓
Proactive Throttling	Naive Thrott. [102] Thrott. Supp. [40] <b>BlockHammer</b>	✓ ✓ ✓	✓ ✗ ✓	✗ ✗ ✓	✓ ✓ ✓

# Outline

DRAM and RowHammer Background

Motivation and Goal

BlockHammer

RowBlocker

AttackThrottler

Evaluation

Conclusion

# Conclusion

- **Motivation:** RowHammer is a worsening DRAM reliability and security problem
- **Problem:** Mitigation mechanisms have limited support for current/future chips
  - **Scalability** with worsening RowHammer vulnerability
  - **Compatibility** with commodity DRAM chips
- **Goal:** Efficiently and scalably prevent RowHammer bit-flips  
without knowledge of or modifications to DRAM internals
- **Key Idea:** Selectively throttle memory accesses that may cause RowHammer bit-flips
- **Mechanism:** BlockHammer
  - **Tracks** activation rates of all rows by using area-efficient Bloom filters
  - **Throttles** row activations that could cause RowHammer bit flips
  - **Identifies and throttles** threads that perform RowHammer attacks
- **Scalability with Worsening RowHammer Vulnerability:**
  - **Competitive** with state-of-the-art mechanisms **when there is no attack**
  - **Superior** performance and DRAM energy **when a RowHammer attack is present**
- **Compatibility with Commodity DRAM Chips:**
  - **No proprietary information** of DRAM internals
  - **No modifications** to DRAM circuitry

# *BlockHammer*

## *Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows*

**Abdullah Giray Yağlıkçı**

Minesh Patel Jeremie S. Kim Roknoddin Azizi

Ataberk Olgun Lois Orosa Hasan Hassan Jisung Park

Konstantinos Kanellopoulos Taha Shahroodi

Saugata Ghose\* Onur Mutlu

**SAFARI**

**ETH** zürich



# BlockHammer

---

A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu, **"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," *HPCA*, 2021.**

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

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

[[Intel Hardware Security Academic Awards Short Talk Slides \(pptx\)](#) ([pdf](#))]

[[Talk Video](#) (22 minutes)]

[[Short Talk Video](#) (7 minutes)]

[[Intel Hardware Security Academic Awards Short Talk Video](#) (2 minutes)]

[[BlockHammer Source Code](#)]

***Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations).***

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- Understanding Read Disturbance in DRAM Chips
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- **Solving RowHammer**
  - BlockHammer [Yaglikci+ HPCA'21]
  - **ABACuS [Olgun+ USENIX Security'24]**

# ABACuS

---

Ataberk Olgun, Yahya Can Tugrul, Nisa Bostanci, Ismail Emir Yuksel, Haocong Luo, Steve Rhynier, Abdullah Giray Yaglikci, Geraldo F. Oliveira, and Onur Mutlu,  
“ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation,”  
in USENIX Security 2024

# ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation

*USENIX Security 2024*

Ataberk Olgun

Yahya Can Tugrul

Nisa Bostancı

İsmail Emir Yüksel

Haocong Luo

Steve Rhyner

Abdullah Giray Yaglikci

Geraldo F. Oliveira    Onur Mutlu

**SAFARI**

**ETH** zürich

# Executive Summary

**Problem:** RowHammer bitflips appear more “easily” as DRAM technology nodes **shrink**. Fewer row activations (**hammers**) induce the first bitflip (**smaller RowHammer threshold**). Existing defenses become more **costly** (performance, energy, and area) with **increasing** i) DRAM density and ii) RowHammer vulnerability.

**Goal:** Prevent RowHammer bitflips at **low** performance, energy, and area **cost** especially at **very low** RowHammer thresholds (e.g., 125 activations induce a bitflip)

**Key Observation:** Many workloads access **the same row ID** in different DRAM banks at around the same time

**Key Idea:** Use **one hardware counter** to track the **activation count** of many rows with the same ID across all DRAM banks

- Implement RowHammer mitigation with **very small hardware storage overhead**

**Key Results:** Ramulator simulations using 62 single- (1C) and eight-core (8C) workloads

- Only **1.45%** (1C) and **4.48%** (8C) perf. overhead at RowHammer threshold = 125
- **1.27%** (1C) and **4.76%** (8C) energy overhead at the same threshold

Circuit area and latency estimation using CACTI and Synopsys DC

- Occupy only **0.11% (0.25 mm<sup>2</sup>)** of an Intel CPU’s area (RowHammer threshold = 125)
- Circuit implementation is **off the critical path**

# Mitigation Approaches with Worsening RowHammer Vulnerability

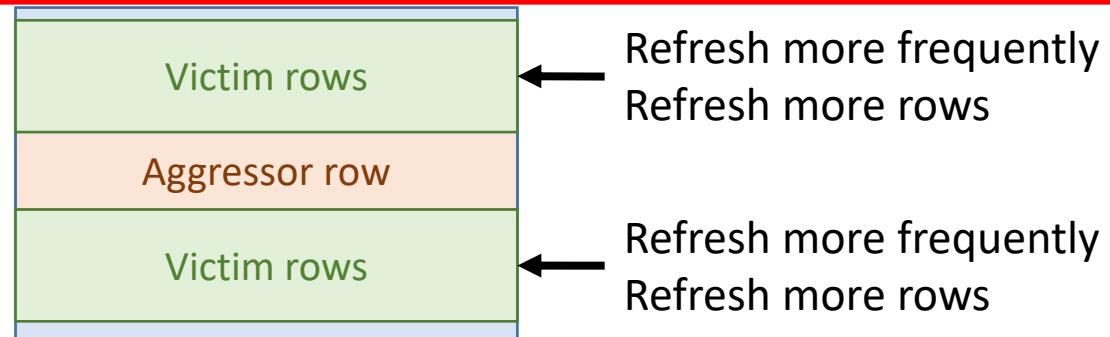
- Increased refresh rate



- Physical isolation



- Preventive refresh



- Proactive throttling



# Identifying Aggressor Rows

Row address R receives an ACT command

Answer the following question: “Is R an aggressor row?”

Identify probabilistically  
“YES” for 1% of ACTs

Needs one  
random number generator

Little chip area cost

A lot of performance cost

Identify deterministically  
“YES” only if the next activation  
will induce a bitflip

Needs many counters  
(counter-based mitigations)

A lot of chip area cost

Little performance cost

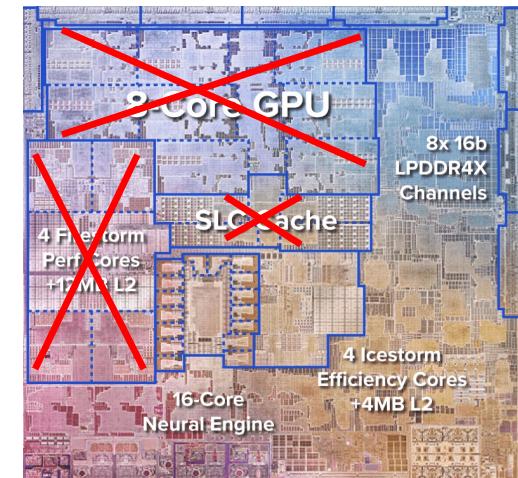
Deterministically prevent bitflips

# Counter-based Mitigations

How many counters are too many counters?

- One counter per DRAM row

- 8 MiB storage per DRAM rank
- 32 MiB storage per DRAM module



- Fewest possible (total) number of counters

- Misra-Gries summaries (Graphene [MICRO'19])
- Require **too much storage at low RowHammer thresholds**
  - 5.68 mm<sup>2</sup> at a RowHammer threshold of 125

# ABA CuS' Motivation & Key Idea

There are **many** (e.g., 16) banks in a DRAM chip

- # of activation counters **linearly increases** w/ # of banks

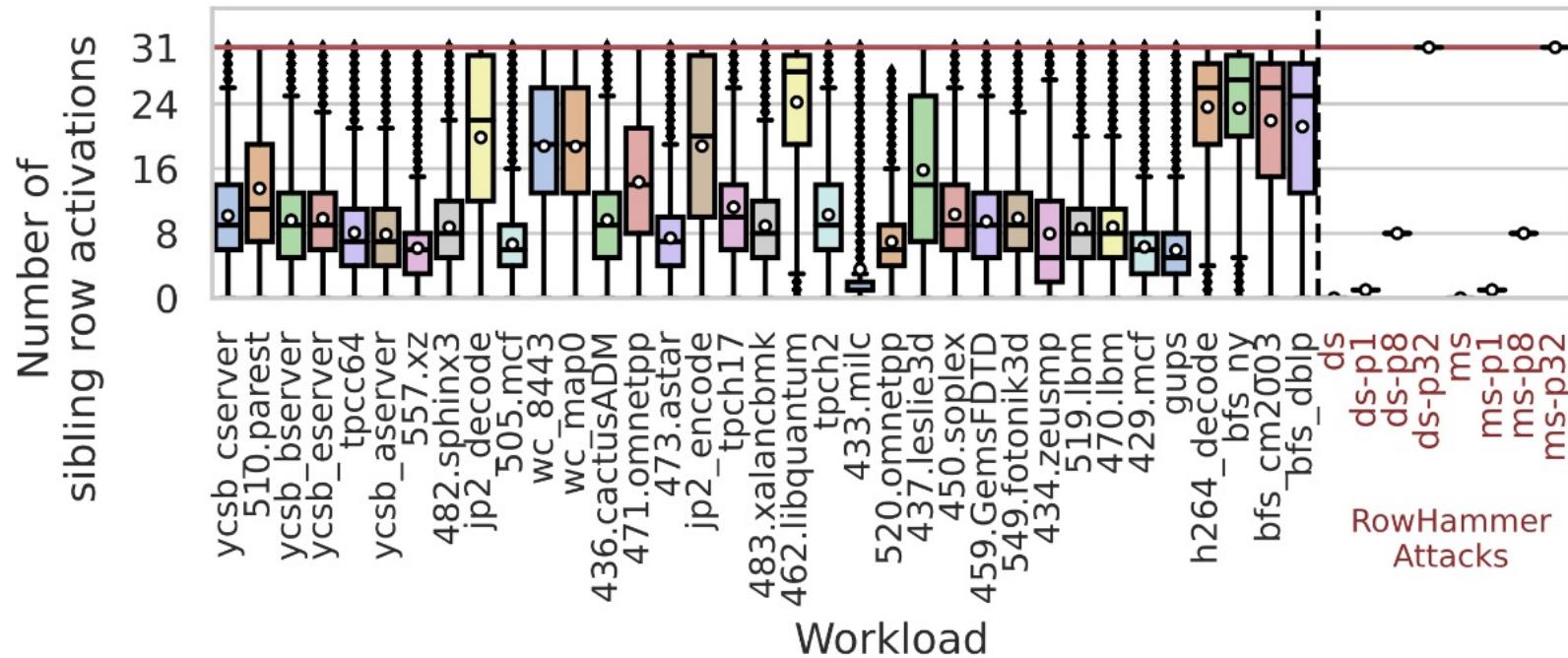
Many workloads access the **same** row ID  
in **different banks** at around the **same time**

**Sibling rows:** Rows with the same ID across all banks

**Key Idea:** Sibling rows can **share** one hardware counter

- **Reduce** the number of counters by  
**a factor of the number of banks** in the chip

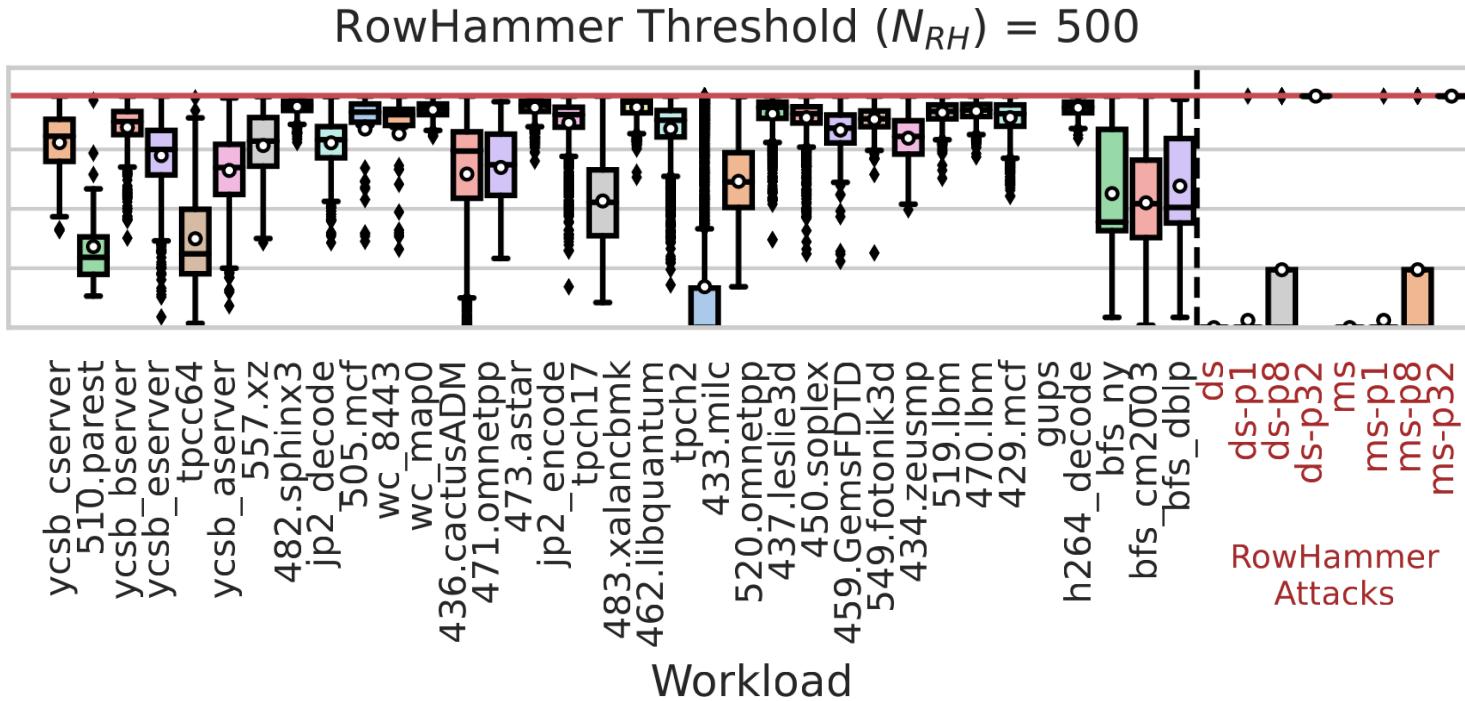
# Motivational Analysis (I)



Workloads access the same row address  
in different DRAM banks at around the same time

# Motivational Analysis (II)

Sibling row average activation count

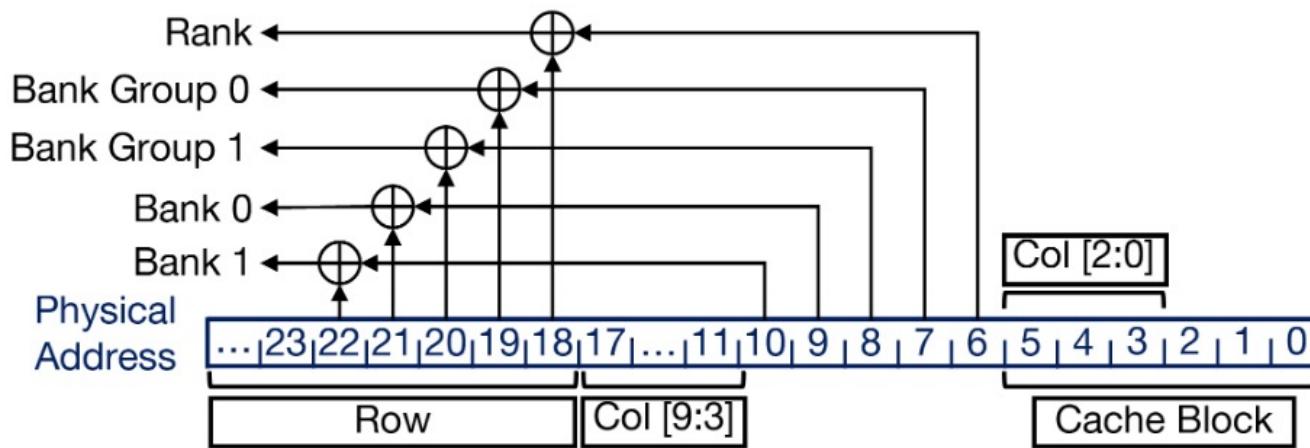


**Sibling rows' activation counts increase  
(arguably) in unison until they reach nRH**

# Why Do the Workloads Do What They Do?

1

- Address mappings distribute **consecutive** cache blocks to **different** banks (but to the **same row ID**)

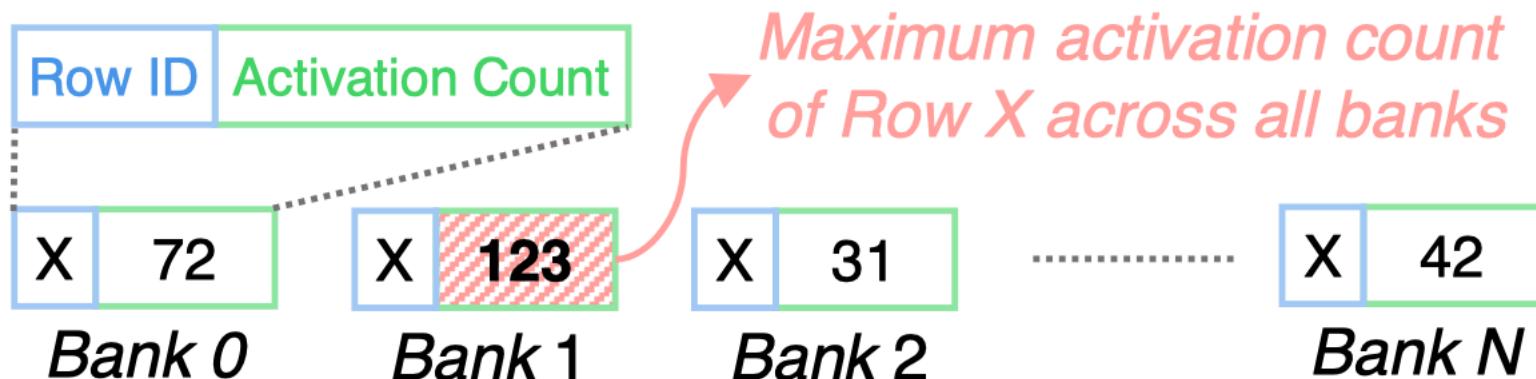


2

- Workloads tend to access cache blocks in close proximity at around the same time

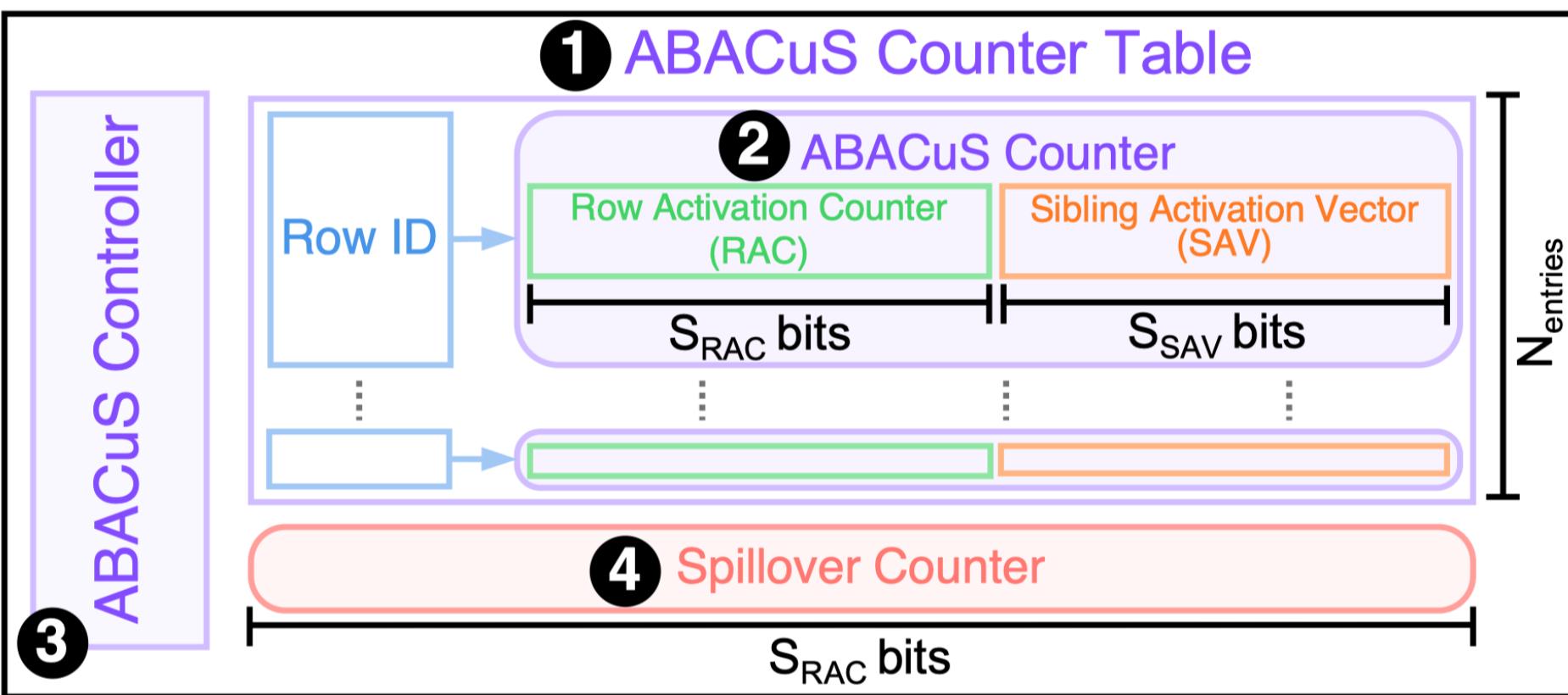
# ABA CuS: High-level Overview

**Key Mechanism:** Track the **maximum** (worst) activation count of sibling rows using **one counter**

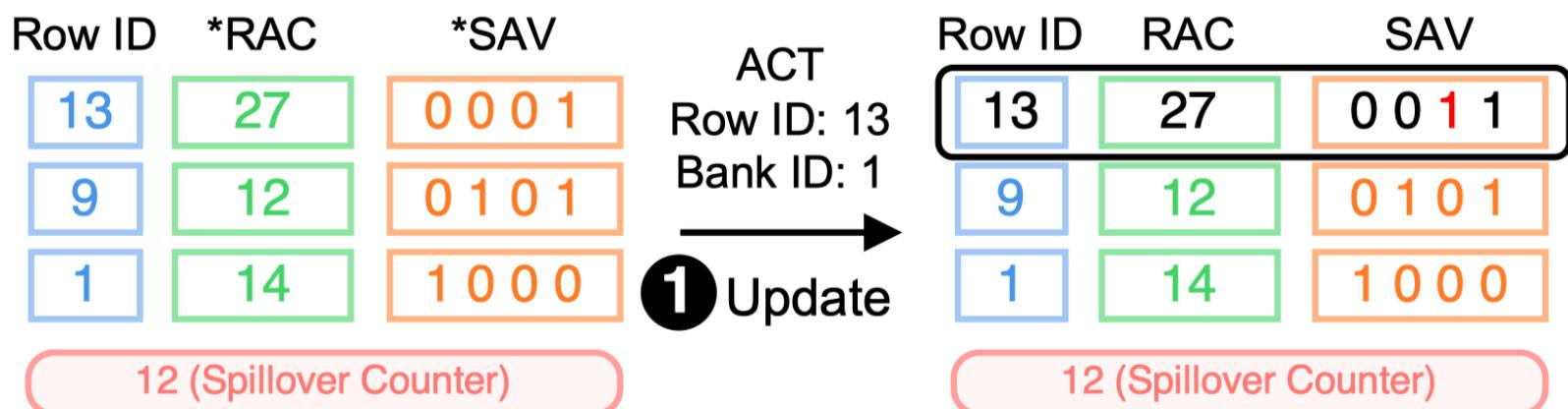


# Key Components

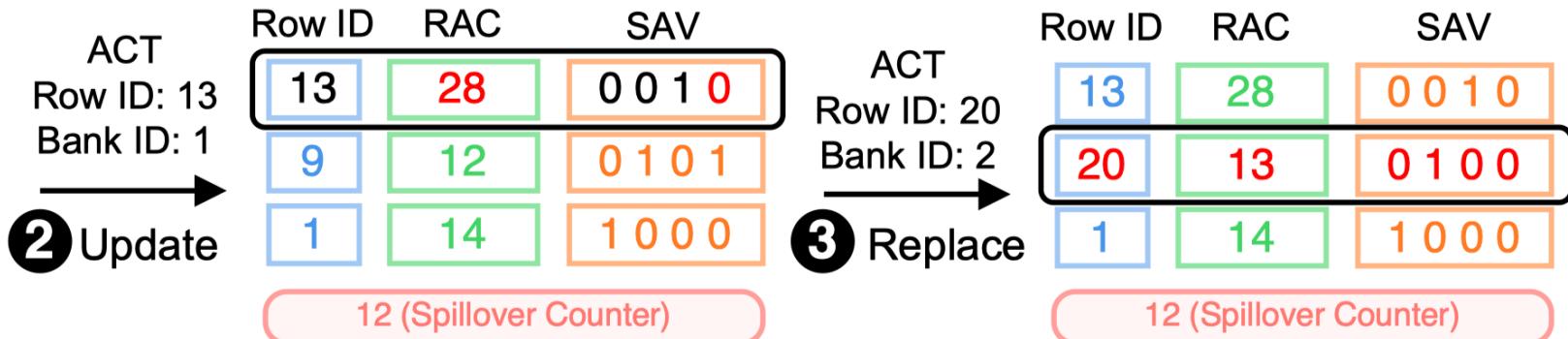
## Memory Controller



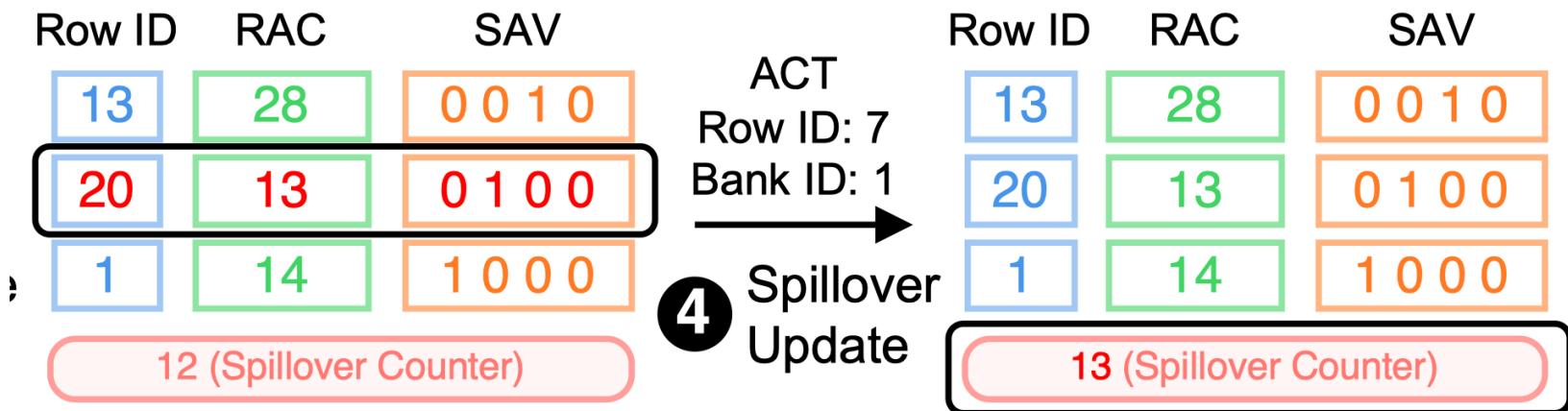
# Operation (I)



\*RAC: Row Activation Counter, SAV: Sibling Activation Vector



# Operation (II)



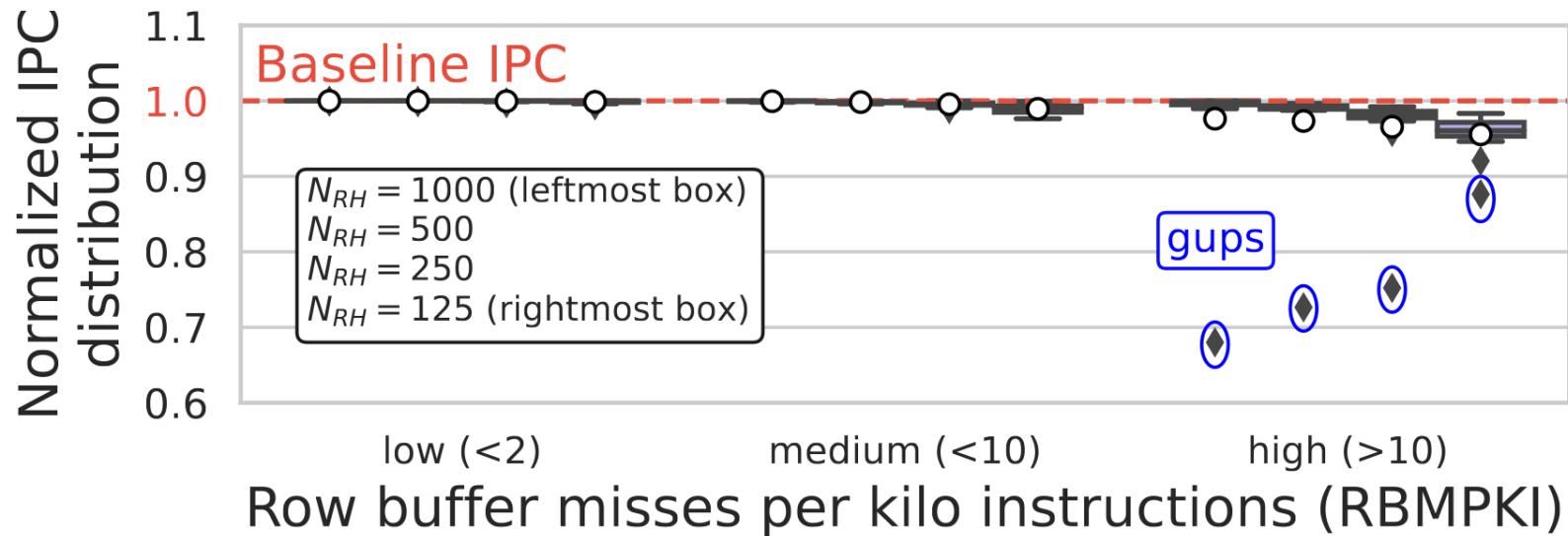
# Evaluation

## System Configuration

<b>Processor</b>	1 or 8 cores, 3.6GHz clock frequency, 4-wide issue, 128-entry instruction window
<b>DRAM</b>	DDR4, 1 channel, 2 rank/channel, 4 bank groups, 4 banks/bank group, 128K rows/bank, 3200 MT/s
<b>Memory Ctrl.</b>	64-entry read and write requests queues, Scheduling policy: FR-FCFS [181, 182] with a column cap of 16 [183], Address mapping: MOP [166, 168] 45 ns $tRC$ , 7.9 $\mu$ s $tREFI$ , 64 ms $tREFW$ 64 ms ABACuS reset period
<b>Last-Level Cache</b>	2 MiB per core

- **Workloads:** 62 1- & 8-core workloads
- Four different very low nRH values: 1000, 500, 250, 125
- Four state-of-the-art mitigation mechanisms

# Performance Normalized to Baseline

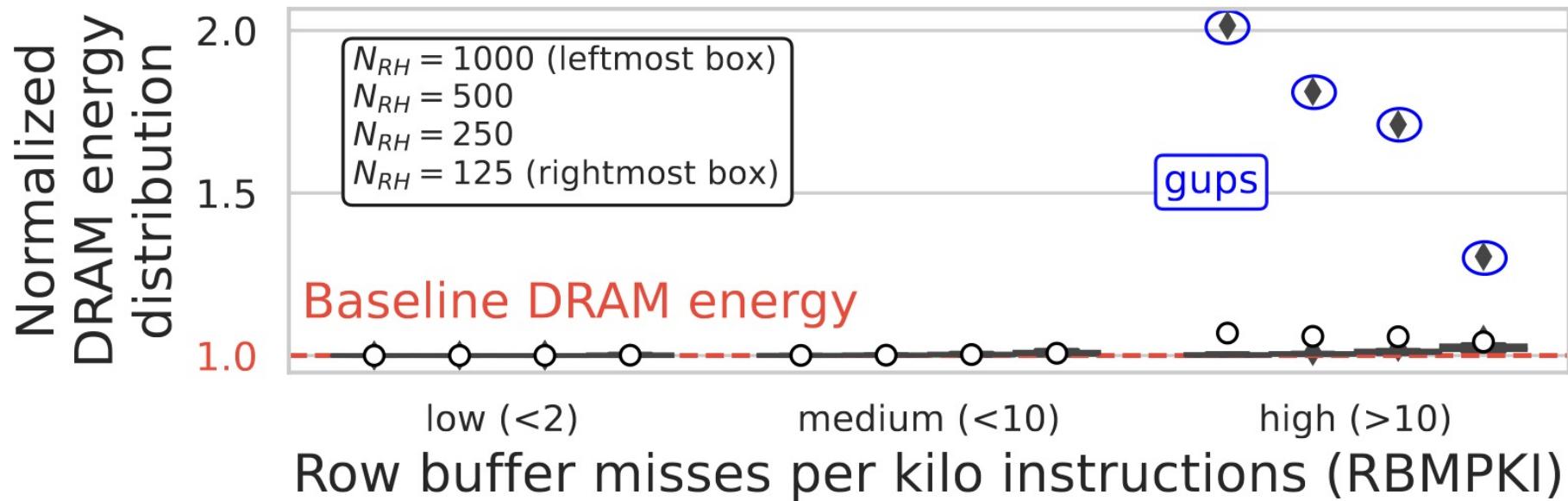


Very low 0.58% average performance overhead at nRH = 1000

Very low 1.45% average performance overhead at nRH = 125

Workload ‘gups’ triggers refresh cycles frequently

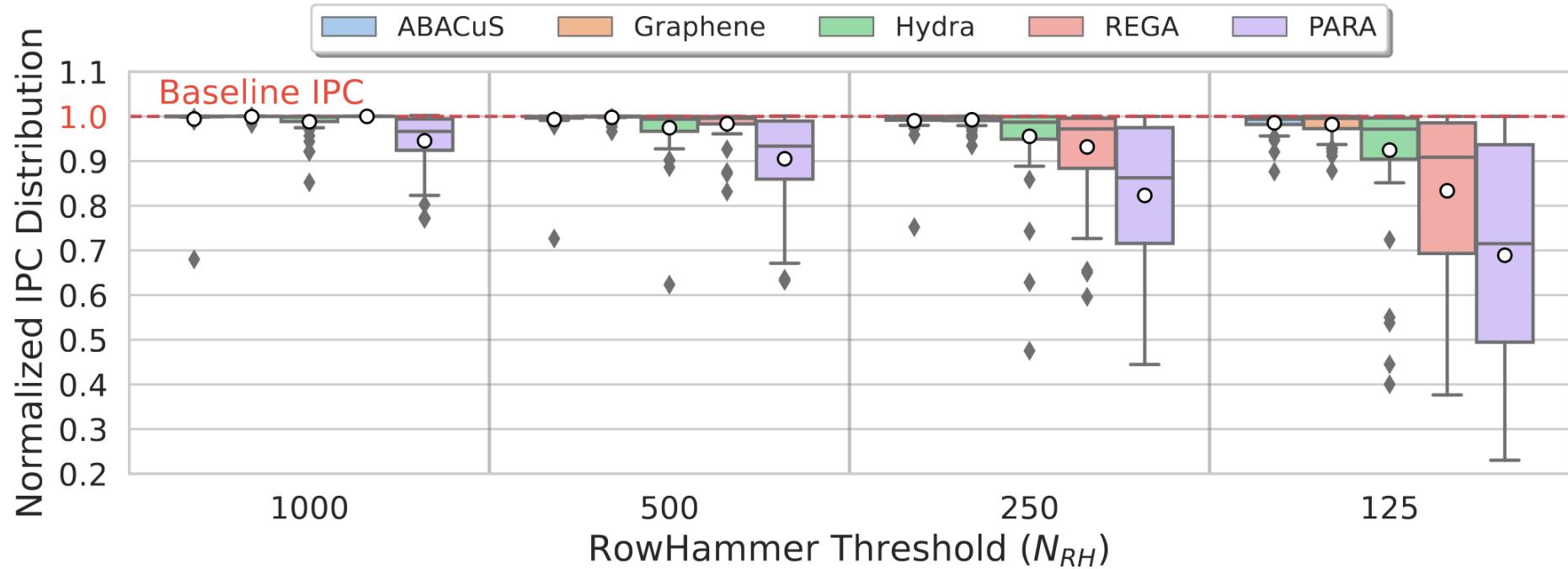
# DRAM Energy Normalized to Baseline



Very low 1.65% average energy overhead at nRH = 1000

Very low 1.27% average energy overhead at nRH = 125

# Performance Comparison



Lower overhead than all evaluated state-of-the-art mechanisms (except Graphene)

# More in the Paper

- More motivational analysis
- Multi-core performance & energy results
- Performance under adversarial workloads
  - Alternative ABACuS design
- Performance & energy sensitivity to:
  - Blast radius
  - Number of ABACuS counters
  - Number of banks
- Circuit area, latency, energy, and power
- Security proof

# Extended Version on arXiv

<https://arxiv.org/pdf/2310.09977.pdf>

arXiv > cs > arXiv:2310.09977

Search... All fields Help | Advanced Search

Computer Science > Cryptography and Security

[Submitted on 15 Oct 2023]

## ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation

Ataberk Olgun, Yahya Can Tugrul, Nisa Bostancı, Ismail Emir Yuksel, Haocong Luo, Steve Rhyner, Abdullah Giray Yaglikci, Geraldo F. Oliveira, Onur Mutlu

We introduce ABACuS, a new low-cost hardware-counter-based RowHammer mitigation technique that performance-, energy-, and area-efficiently scales with worsening RowHammer vulnerability. We observe that both benign workloads and RowHammer attacks tend to access DRAM rows with the same row address in multiple DRAM banks at around the same time. Based on this observation, ABACuS's key idea is to use a single shared row activation counter to track activations to the rows with the same row address in all DRAM banks. Unlike state-of-the-art RowHammer mitigation mechanisms that implement a separate row activation counter for each DRAM bank, ABACuS implements fewer counters (e.g., only one) to track an equal number of aggressor rows.

Our evaluations show that ABACuS securely prevents RowHammer bitflips at low performance/energy overhead and low area cost. We compare ABACuS to four state-of-the-art mitigation mechanisms. At a near-future RowHammer threshold of 1000, ABACuS incurs only 0.58% (0.77%) performance and 1.66% (2.12%) DRAM energy overheads, averaged across 62 single-core (8-core) workloads, requiring only 9.47 KiB of storage per DRAM rank. At the RowHammer threshold of 1000, the best prior low-area-cost mitigation mechanism incurs 1.80% higher average performance overhead than ABACuS, while ABACuS requires 2.50X smaller chip area to implement. At a future RowHammer threshold of 125, ABACuS performs very similarly to (within 0.38% of the performance of) the best prior performance- and energy-efficient RowHammer mitigation mechanism while requiring 22.72X smaller chip area. ABACuS is freely and openly available at [this https URL](https://arxiv.org/pdf/2310.09977.pdf).

Access Paper:

- Download PDF
- PostScript
- Other Formats

(cc) BY

Current browse context: cs.CR  
< prev | next >  
new | recent | 2310

Change to browse by:  
cs  
cs.AR

References & Citations  

- NASA ADS
- Google Scholar
- Semantic Scholar

Export BibTeX Citation

Bookmark

# ABAQUS is Open Source

<https://github.com/CMU-SAFARI/ABACuS>

The screenshot shows the GitHub repository page for ABACuS. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. Below the navigation is the repository header for ABACuS, which is public. The main content area displays a list of commits from olgunataberk, showing additions of Verilog sources and updates to the README. Each commit includes a link to the file, a 'Go to file' button, and a 'Add file' button. On the right side, there's an 'About' section with a detailed description of a new RowHammer mitigation mechanism, a link to the USENIX Security'24 paper, and various repository statistics like stars, forks, and watching.

Code Issues Pull requests Actions Projects Security Insights Settings

ABACuS Public

Edit Pins Unwatch 4 Fork 0 Starred 3

main 1 branch 0 tags Go to file Add file Code

olgunataberk add verilog sources and update readme ef1c89c yesterday 8 commits

abacus\_cacti add abacus cacti sources yesterday

abacus\_verilog add verilog sources and update readme yesterday

configs/ABACUS Initial commit 3 days ago

ext Initial commit 3 days ago

scripts Initial commit 3 days ago

src Initial commit 3 days ago

.gitignore Initial commit 3 days ago

CMakeLists.txt Initial commit 3 days ago

Doxmyfile Initial commit 3 days ago

About

New RowHammer mitigation mechanism that is area-, performance-, and energy-efficient especially at very low (e.g., 125) RowHammer thresholds, as described in the USENIX Security'24 paper <https://arxiv.org/pdf/2310.09977.pdf>

Readme MIT license Activity 3 stars 4 watching 0 forks Report repository

# ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation

*USENIX Security 2024*

Ataberk Olgun

Yahya Can Tugrul

Nisa Bostancı

İsmail Emir Yüksel

Haocong Luo

Steve Rhyner

Abdullah Giray Yaglikci

Geraldo F. Oliveira    Onur Mutlu

**SAFARI**

**ETH** zürich

# ABACuS

---

Ataberk Olgun, Yahya Can Tugrul, Nisa Bostanci, Ismail Emir Yuksel, Haocong Luo, Steve Rhynier, Abdullah Giray Yaglikci, Geraldo F. Oliveira, and Onur Mutlu,  
“ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation,”  
in USENIX Security 2024

# Outline

---

- Infrastructure
  - DRAM Bender [Olgun+ IEEE TCAD'23]
- Understanding Read Disturbance in DRAM Chips
  - A Deeper Look into RowHammer [Orosa+ MICRO'21]
  - RowPress [Luo+ ISCA'23]
  - HBM RowHammer [Olgun+ DSN Disrupt'23]
- Solving RowHammer
  - BlockHammer [Yaglikci+ HPCA'21]
  - ABACuS [Olgun+ USENIX Security'24]

# Computer Architecture

## Lecture 7: Cutting Edge Research on DRAM Read Disturbance

A. Giray Yaglikci, Ataberk Olgun, Haocong Luo

Prof. Onur Mutlu

ETH Zürich

Fall 2023

19 October 2023

# *A Deeper Look into RowHammer's Sensitivities*

*Experimental Analysis of Real DRAM Chips  
and Implications on Future Attacks and Defenses*

## ***BACKUP SLIDES***

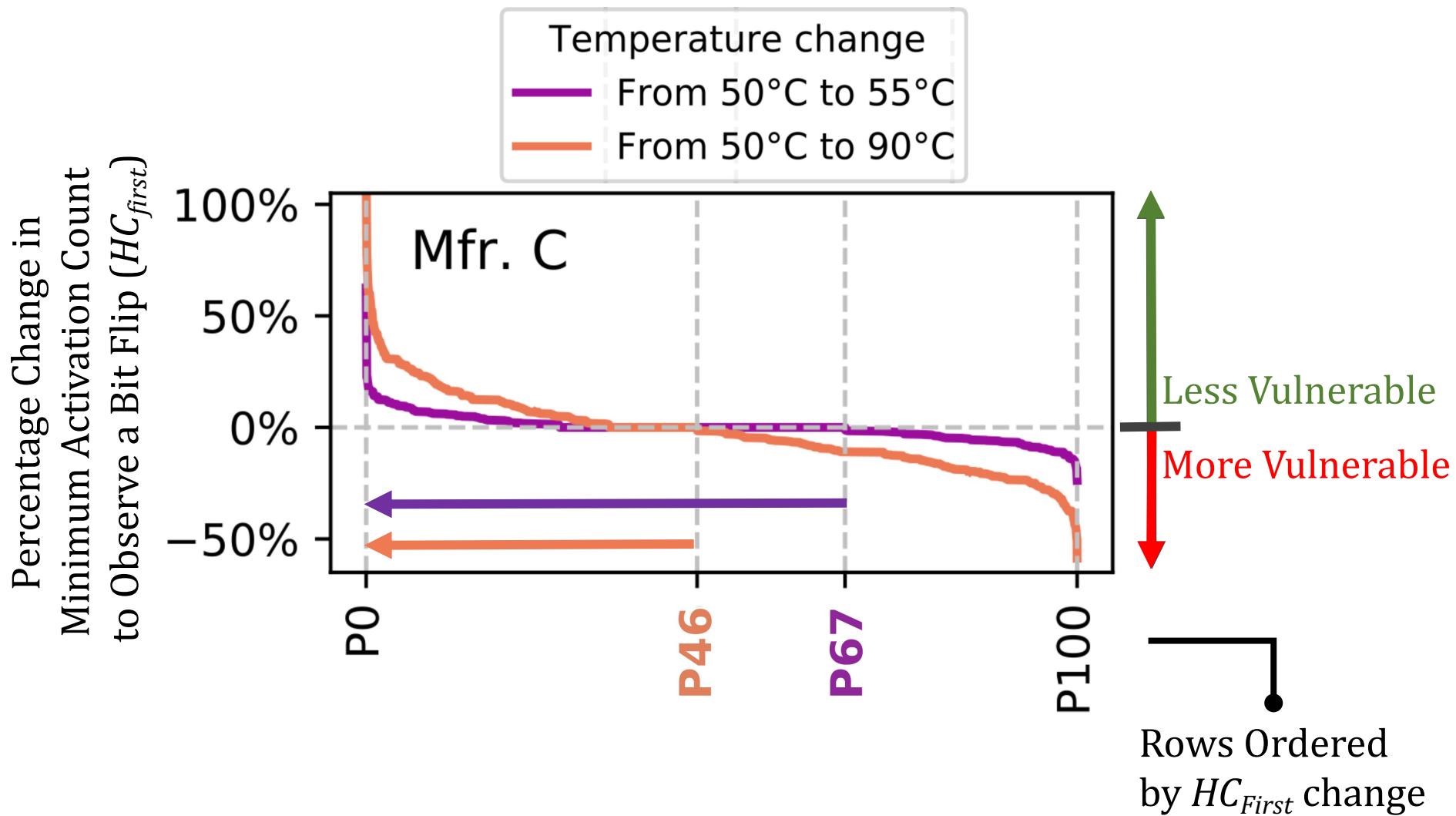
**Lois Orosa    Abdullah Giray Yağlıkçı**

Haocong Luo   Ataberk Olgun   Jisung Park

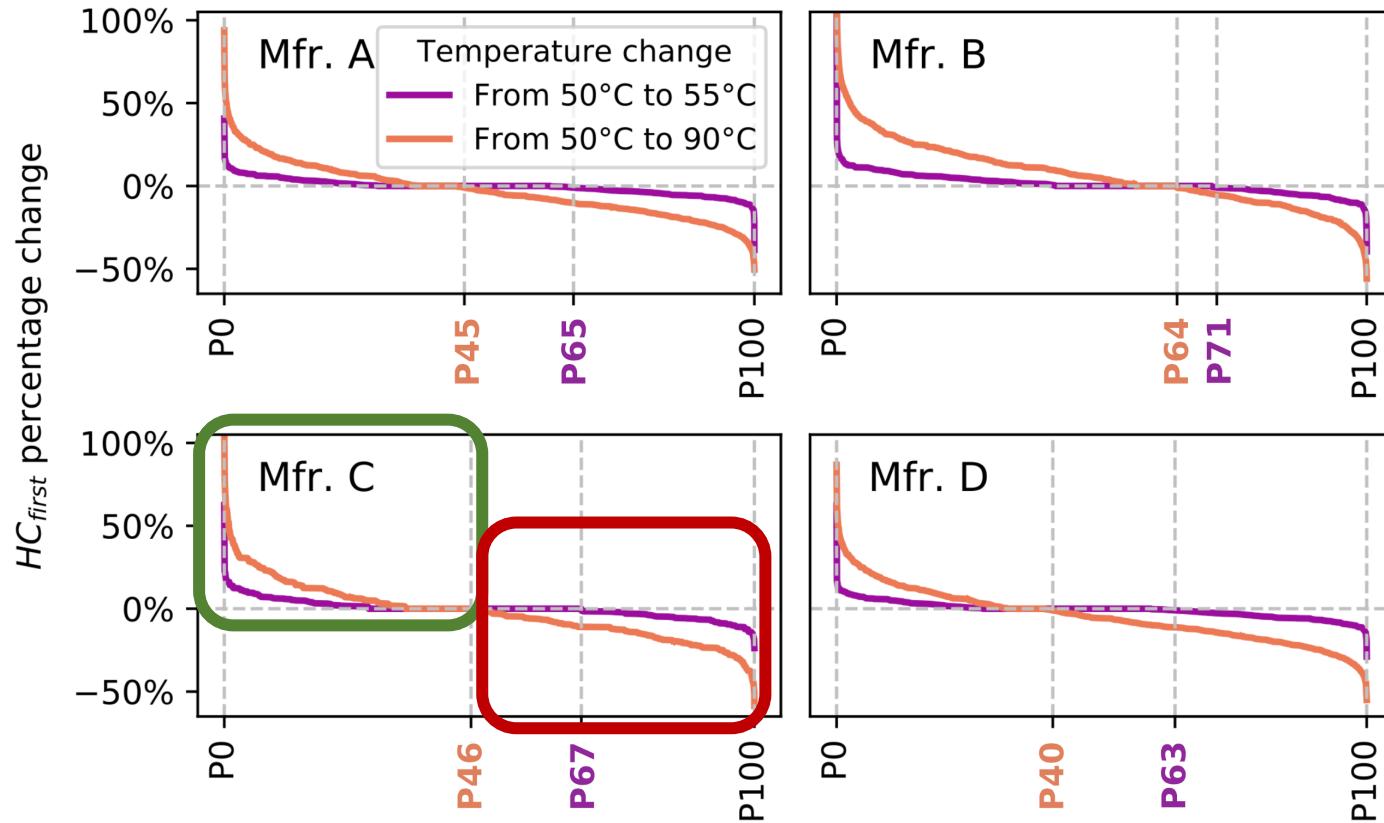
Hasan Hassan   Minesh Patel   Jeremie S. Kim   Onur Mutlu

***SAFARI***

# Distribution of the Change in $HC_{first}$



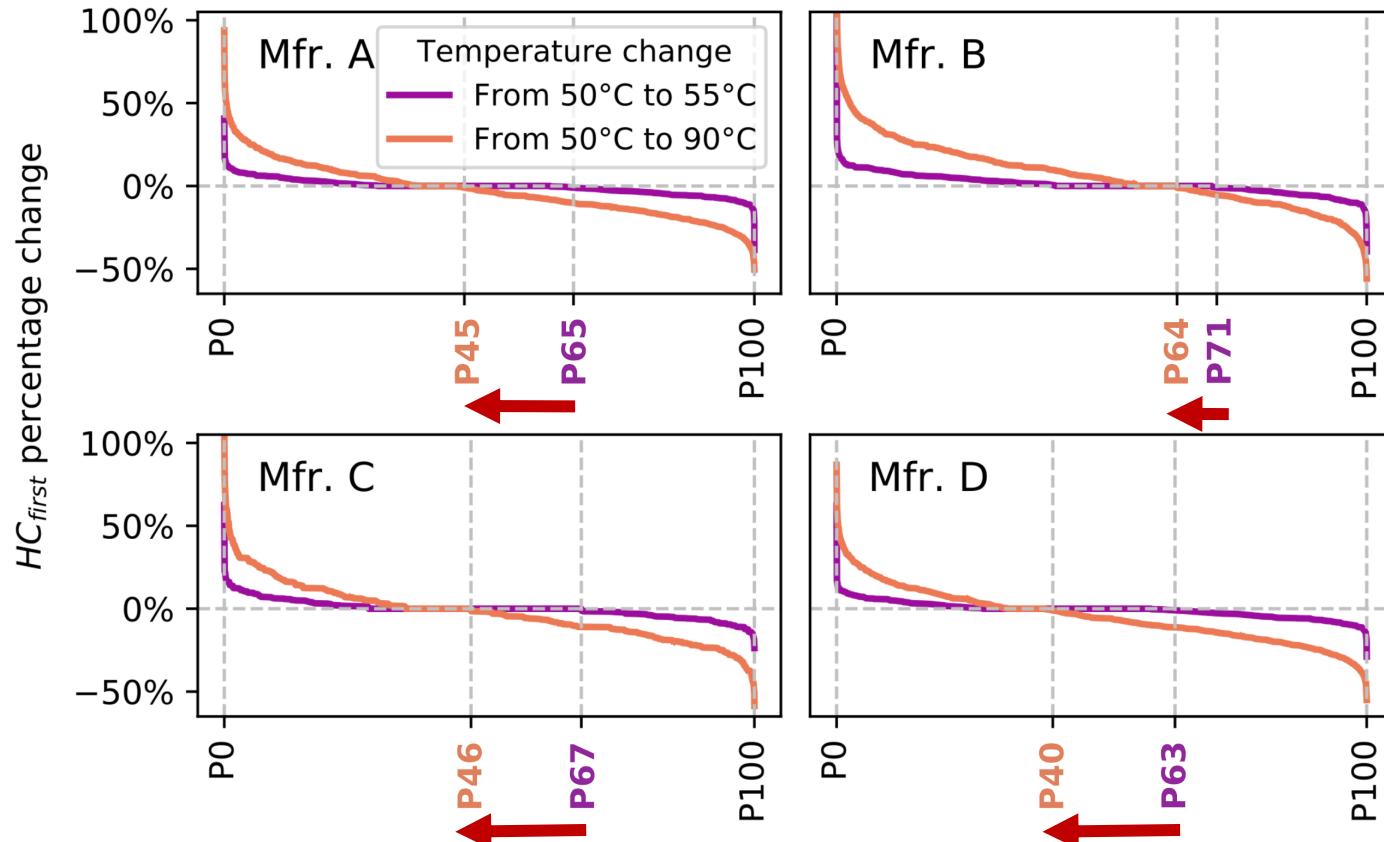
# Distribution of the Change in $HC_{first}$



## OBSERVATION 5

DRAM rows can show either **higher** or **lower**  $HC_{first}$  when **temperature increases**

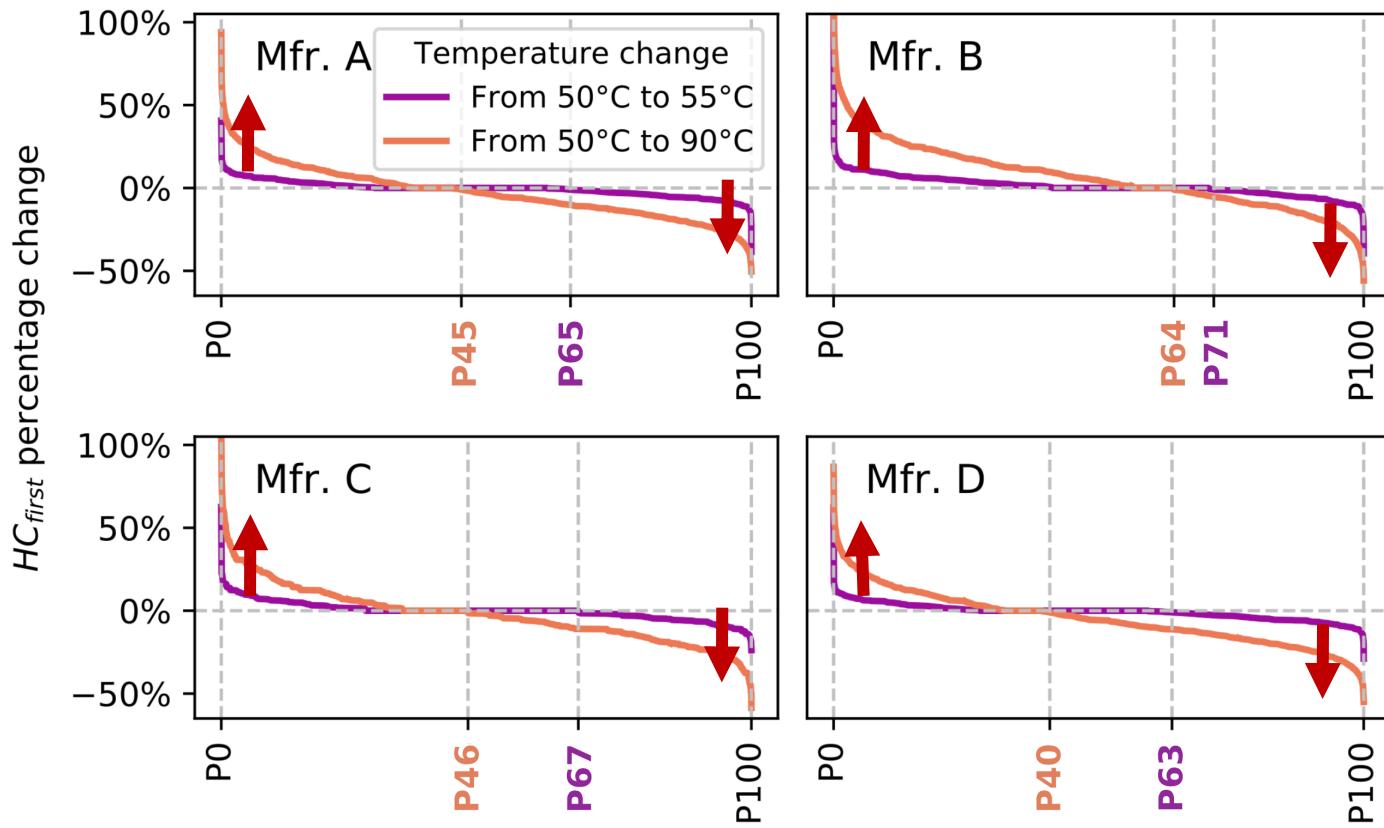
# Distribution of the Change in $HC_{first}$



## OBSERVATION 6

$HC_{first}$  tends to generally **decrease** as temperature change ( $\Delta T$ ) increases

# Distribution of the Change in $HC_{first}$



## OBSERVATION 7

The  $HC_{first}$  change ( $\Delta HC_{first}$ ) tends to be **larger** as **temperature change** ( $\Delta T$ ) increases

# Circuit-Level Justification

## Temperature Analysis

We hypothesize that our observations are caused by the **non-monotonic behavior of charge trapping** characteristics of DRAM cells

### 3D TCAD model [Yang+, EDL'19]

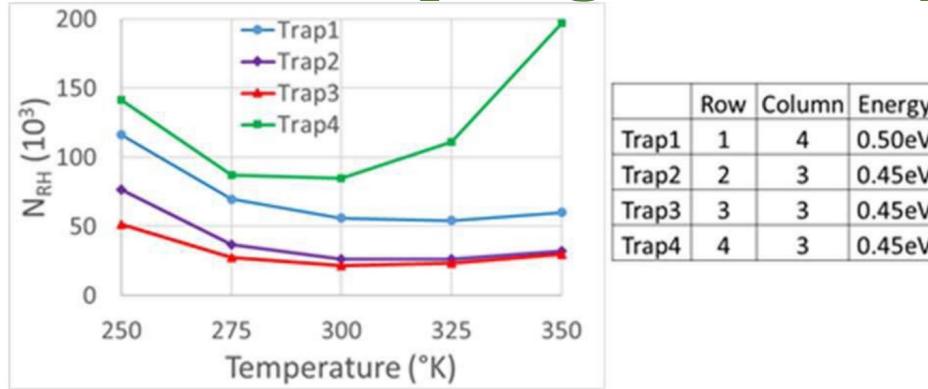
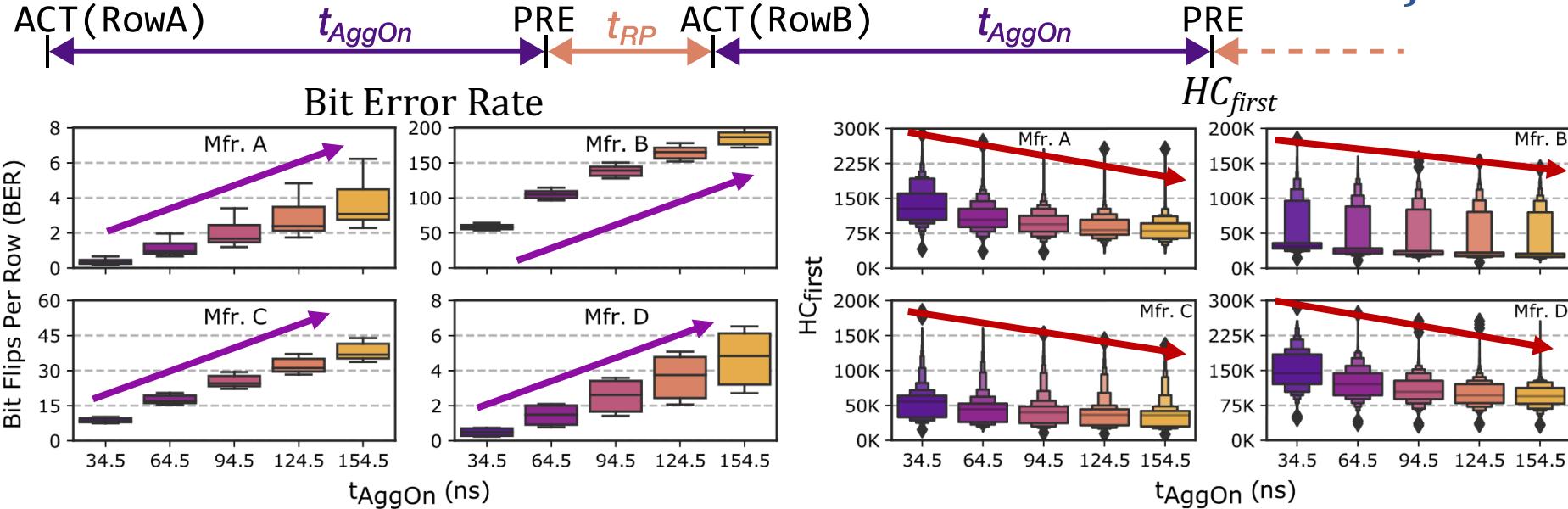


Fig. 6. Hammering threshold  $N_{RH}$  vs. temperature from 250 to 350°K for different traps. Location in row and column refers to matrix in Fig. 2b.

**$HC_{first}$  decreases as temperature increases**, until a temperature inflection point where  **$HC_{first}$  starts to increase as temperature increases**

A **cell is more vulnerable** to RowHammer at **temperatures close to its temperature inflection point**

# Increasing Aggressor Row Active Time ( $t_{\text{AggOn}}$ )



## OBSERVATION 8

As the **aggressor row stays active longer**,  
**more DRAM cells** experience RowHammer bit flips and  
they experience RowHammer bit flips **at lower hammer counts**

We analyze how the *coefficient of variation*\* values for **BER** and **HC<sub>first</sub>** change  
across rows when the **aggressor row stays active longer**

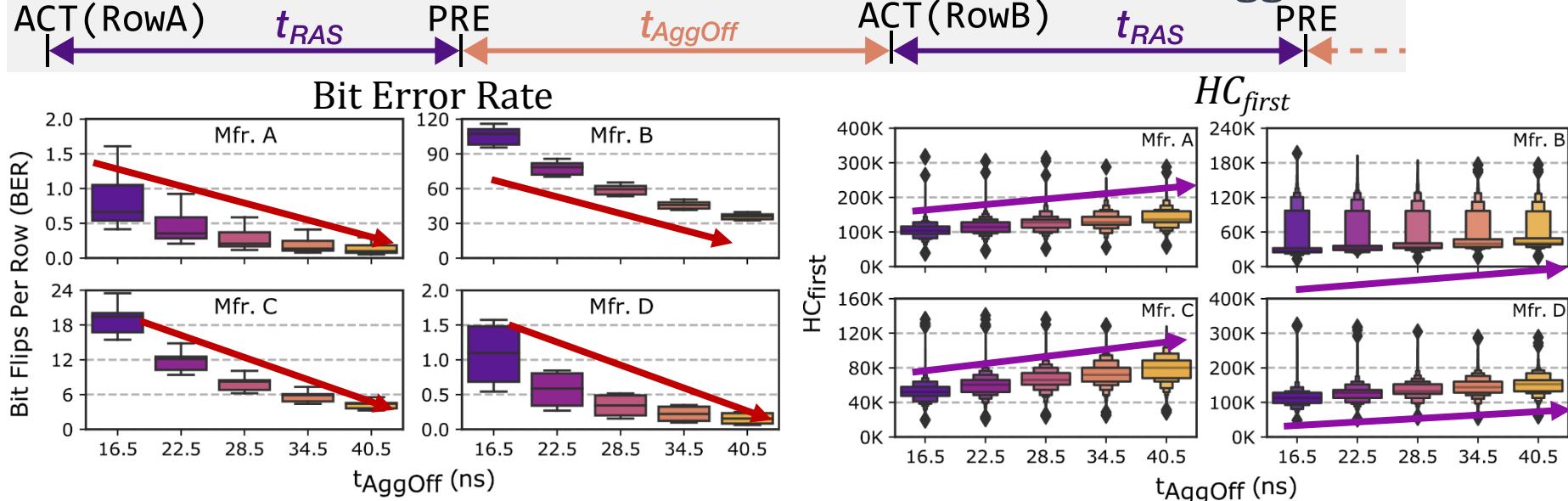
## OBSERVATION 9

RowHammer vulnerability **consistently worsens**  
**as  $t_{\text{AggOn}}$  increases** across all tested DRAM rows\*\*

\*Coefficient of Variation (CV) = Standard Deviation/Average

\*\* Please refer to the full paper for coefficient of variation-based (CV) analysis

# Increasing Bank Precharged Time ( $t_{\text{AggOff}}$ )



## OBSERVATION 10

As the **bank stays precharged longer, fewer DRAM cells**  
experience RowHammer bit flips and they experience RowHammer  
bit flips **at higher hammer counts**

We repeat the *coefficient of variation*\* analysis for **BER** and **HC<sub>first</sub>** change  
across rows when the **bank stays precharged longer**

## OBSERVATION 11

RowHammer vulnerability **consistently reduces**  
**as  $t_{\text{AggOff}}$  increases** across all tested DRAM rows\*\*

# Circuit-Level Justification

## Aggressor Row Active Time Analysis

Two possible circuit level justifications for RowHammer bit flips:

1. Electron injection in the victim cell [Walker+, TED'21][Yang+, TDMR'16]
2. Wordline-to-wordline cross-talk noise between aggressor and victim rows that occurs when the aggressor row is being activated [Ryu+, IEDM'17][Walker+, TED'21]

We hypothesize that **increasing the aggressor row's active time ( $t_{AggOn}$ ) has a larger impact on exacerbating electron injection to the victim cell**, compared to the reduction in cross-talk noise due to lower activation frequency. Thus, RowHammer vulnerability worsens when  $t_{AggOn}$  increases

Increasing a bank's precharged time ( $t_{AggOff}$ ) decreases RowHammer vulnerability because **longer  $t_{AggOff}$  reduces the effect of cross-talk noise without affecting electron injection** (since  $t_{AggOn}$  is unchanged).

# Spatial Variation across Rows

Minimum Activation Count  
to Observe a Bit Flip ( $HC_{first}$ )

120K  
80K  
40K  
0K

DRAM Rows (sorted by reducing  $HC_{first}$ )

$HC_{first}$

worst-to-best ratio in this range: P100/P90

Mfr. C

P90

P95

P99

P100

$HC_{first}$  Worst-to-Best Ratio

0.2 0.3 0.4 0.5 0.6 0.7

A B C D

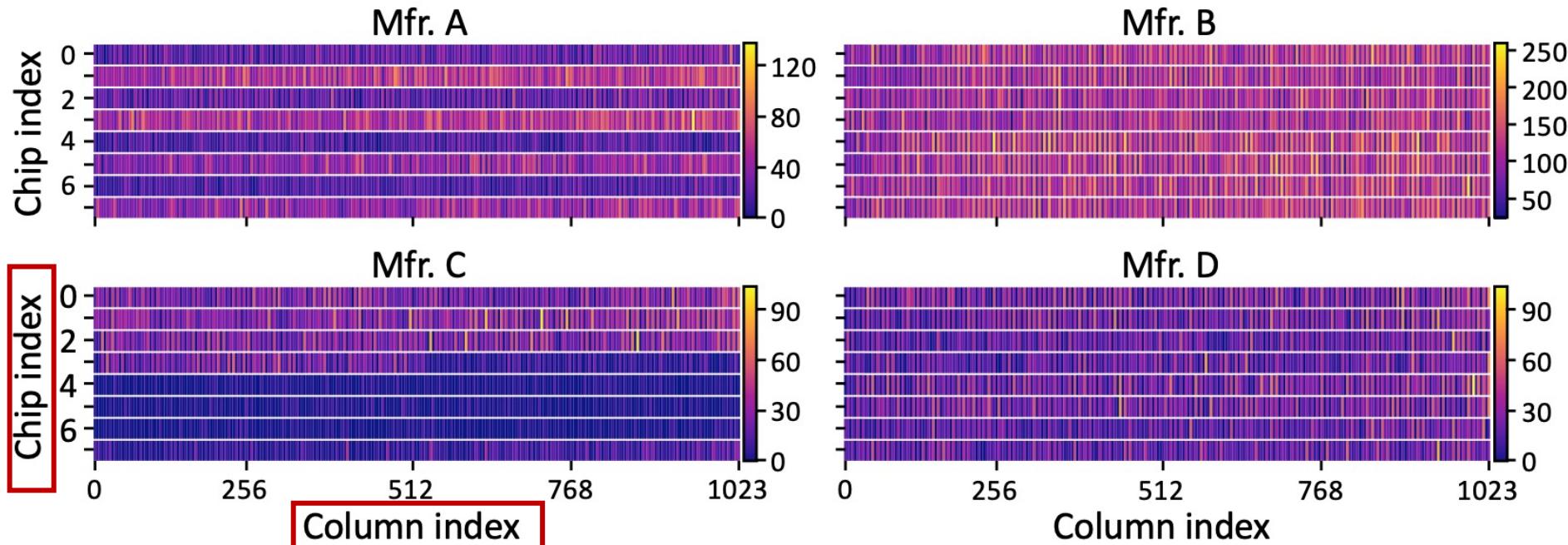
Manufacturers

## OBSERVATION 12

A small fraction of DRAM rows  
are significantly more vulnerable to RowHammer  
than the vast majority of the rows

# Spatial Variation across Columns

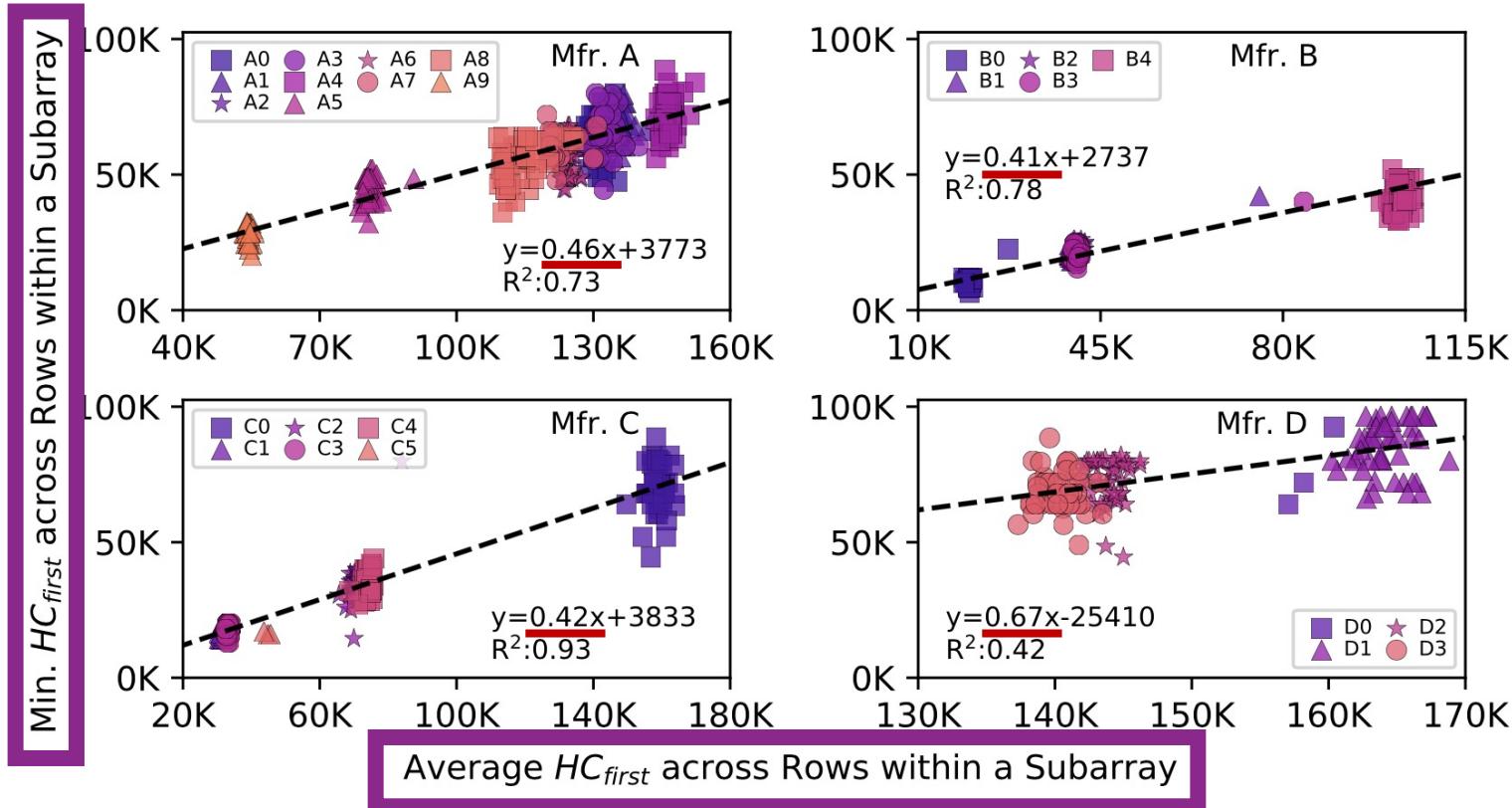
We analyze BER variation across DRAM columns



## OBSERVATION 13

Certain columns are **significantly more vulnerable** to RowHammer than other columns

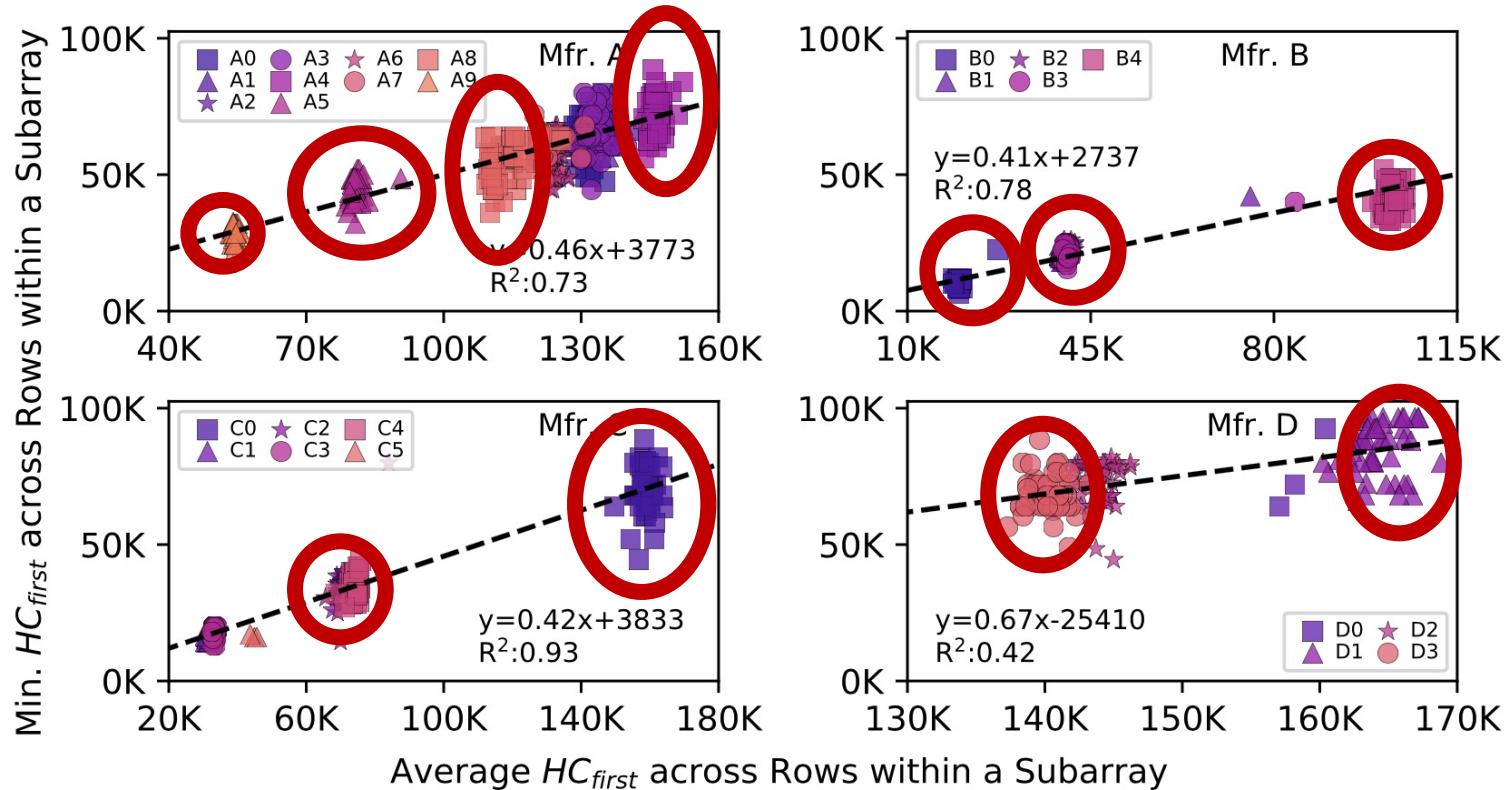
# Spatial Variation across Subarrays



## OBSERVATION 15

The most vulnerable DRAM row in a subarray  
is significantly more vulnerable  
than the other rows in the subarray

# Spatial Variation across Subarrays



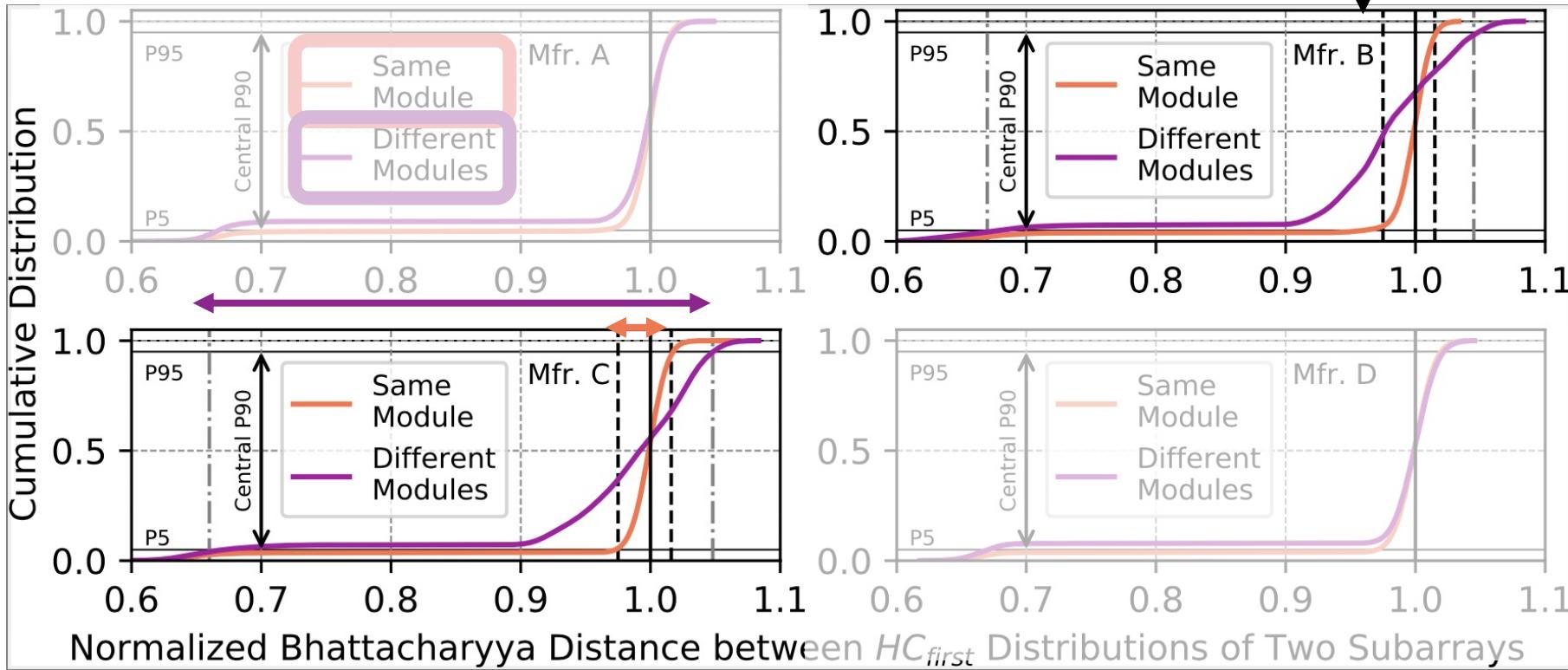
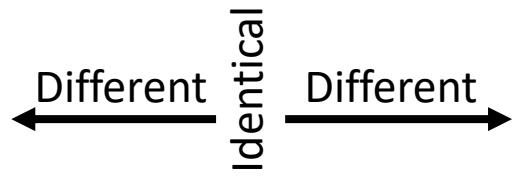
## OBSERVATION 16

$HC_{first}$  distributions of subarrays **within a DRAM module** are **significantly more similar** to each other than those of subarrays **from different modules**

\* We analyze the similarity between Hcfirst distributions of different subarrays based on Bhattacharyya distance in the paper

# Spatial Variation across Subarrays

## Bhattacharyya Distance Analysis



$HC_{first}$  distributions of subarrays **within a DRAM module** exhibit **significantly more similarity** to each other than  $HC_{first}$  distributions of subarrays **from different modules**

# Circuit-Level Justification

## Spatial Variation Analysis

Variation across rows, columns, and chips:

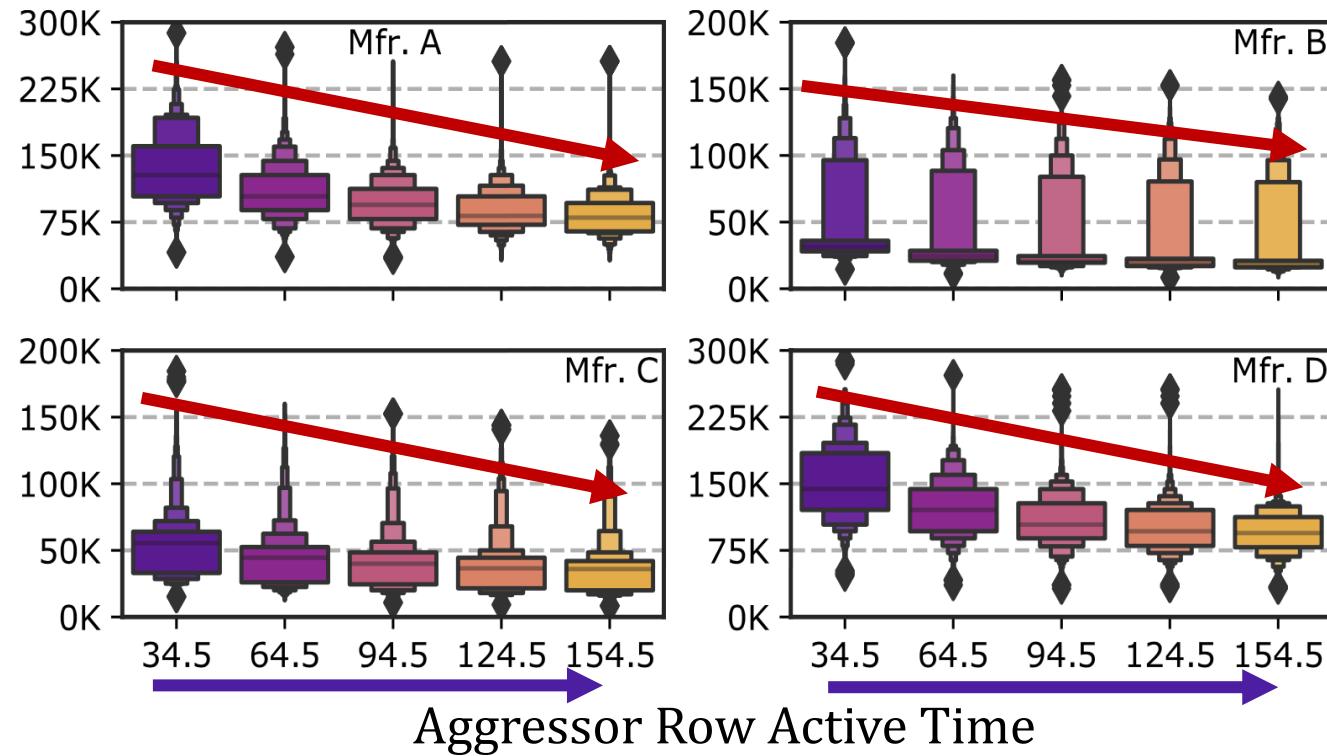
**Manufacturing process variation** causes **differences in cell size and bitline/wordline impedance values**, which introduces variation in cell reliability characteristics within and across DRAM chips

**Design-induced variation** causes cell access **latency characteristics to vary deterministically based on a cell's physical location** in the memory chip (e.g., its proximity to I/O circuitry)

Similarity across subarrays:

Cell's **access latency** is dominated by its **physical distance from the peripheral structures** (e.g., local sense amplifiers and wordline drivers) **within the subarray**, causing **corresponding cells in different subarrays to exhibit similar access latency characteristics**

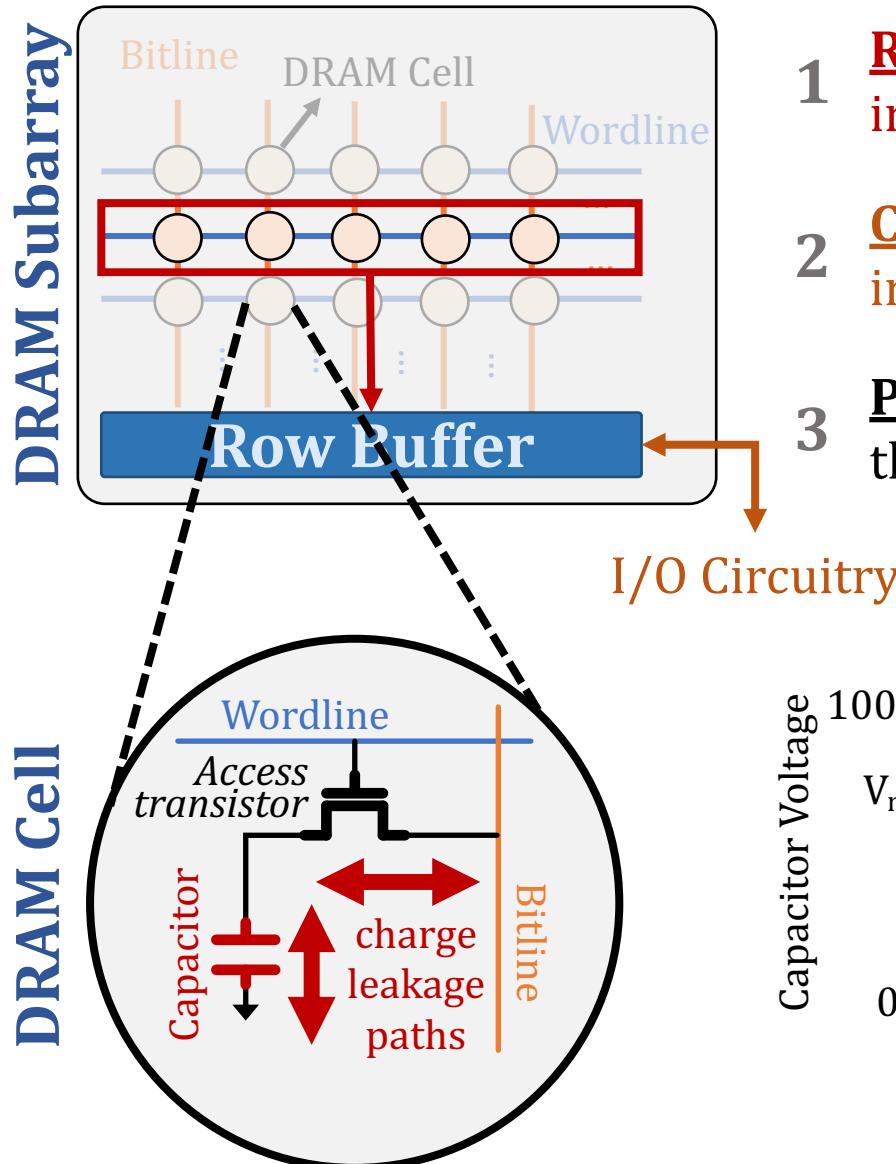
# Example Attack Improvements



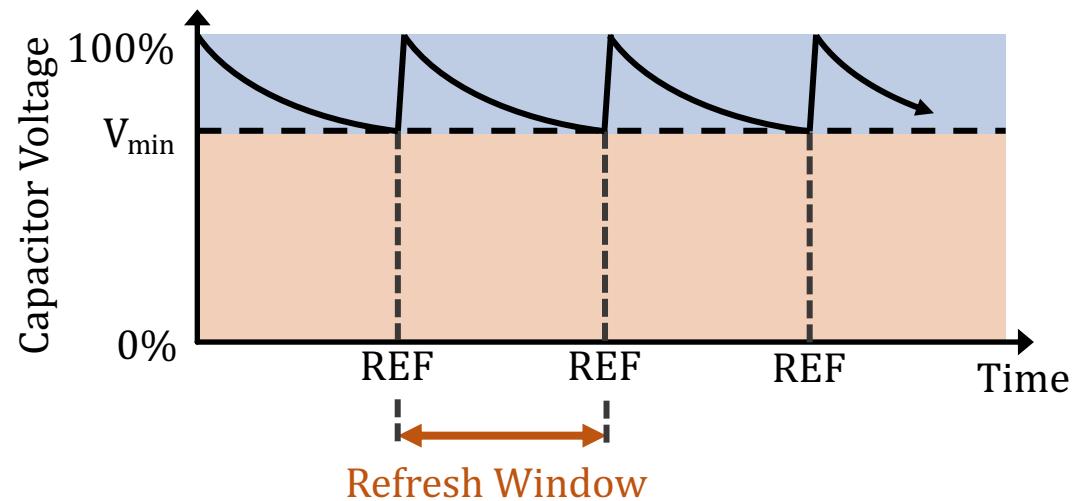
- The attacker can **reduce  $HC_{first}$  (by 36%)** by **performing (10-15) additional READ commands targeting the aggressor row to bypass RowHammer defenses** that do not account for this reduction

These observations can be leveraged  
to craft more effective RowHammer attacks

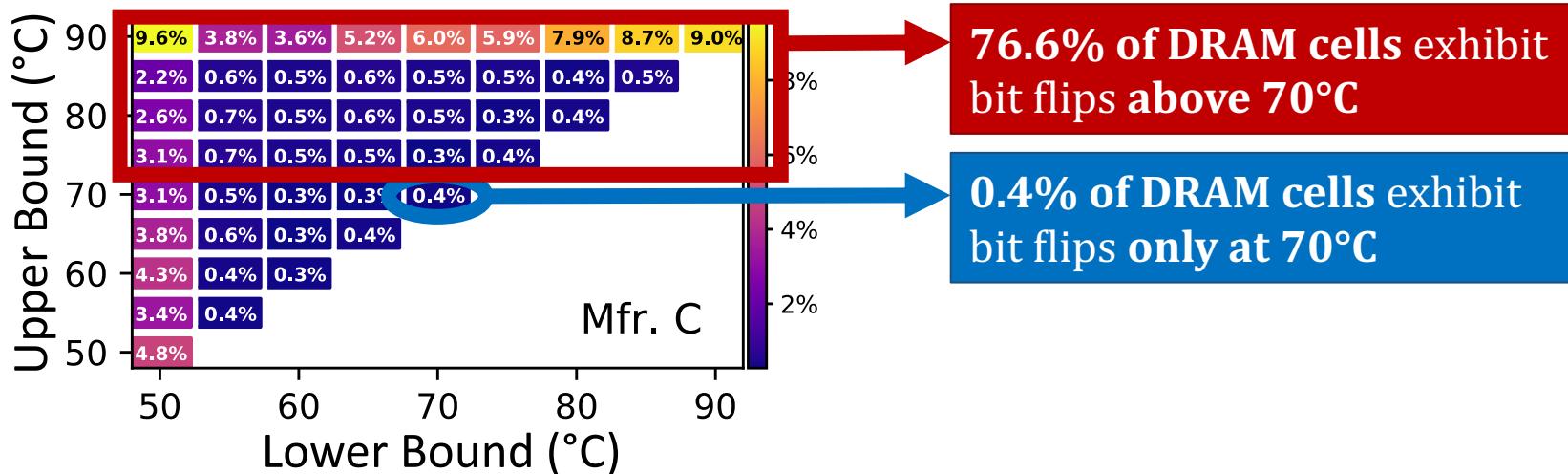
# DRAM Operation



- 1 **Row Activation:** Fetching the row's content into the **row buffer**
- 2 **Column Access:** Read/Write a column in the row buffer
- 3 **Precharge:** Disconnect the row from the row buffer



# Example Attack Improvements



- **Example 1: Temperature-dependent trigger**

An attacker can measure DRAM chip's **current temperature**

- To identify when the DRAM chip is **at a certain temperature**
  - To precisely measure temperature for **covert channels**
- To identify **abnormal operating conditions** (e.g., warmer than usual)
  - To attack a data center **during its peak hours**
  - To **spy on** an end-user's behavioral patterns

- **Example 2: Manipulating temperature to make chips more vulnerable**

An attacker can **heat up or cool down** a DRAM chip to a temperature level  
**where the victim cells are vulnerable**

# *A Deeper Look into RowHammer's Sensitivities*

*Experimental Analysis of Real DRAM Chips  
and Implications on Future Attacks and Defenses*

## ***BACKUP SLIDES***

**Lois Orosa    Abdullah Giray Yağlıkçı**

Haocong Luo   Ataberk Olgun   Jisung Park

Hasan Hassan   Minesh Patel   Jeremie S. Kim   Onur Mutlu

***SAFARI***



# 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*

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

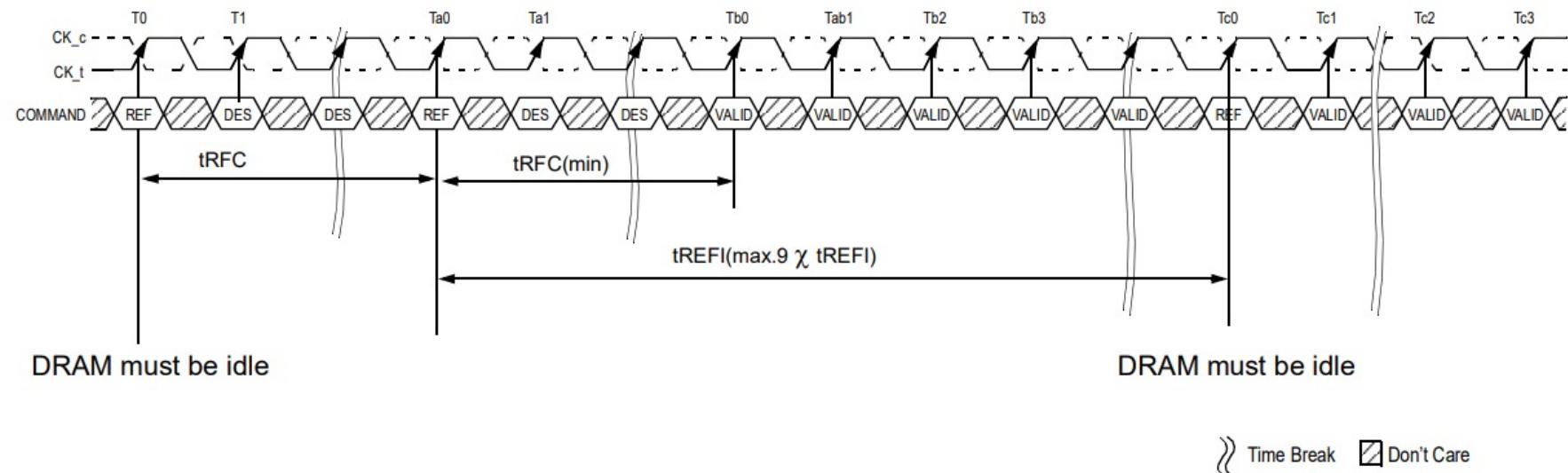
**SAFARI**

**ETH** Zürich

# Backup Slide

## Potential tAggON upper bounds

- **tREFI:** Interval between two REF commands



NOTE 1 Only DES commands allowed after Refresh command registered until tRFC(min) expires.

NOTE 2 Time interval between two Refresh commands may be extended to a maximum of 9 X tREFI.

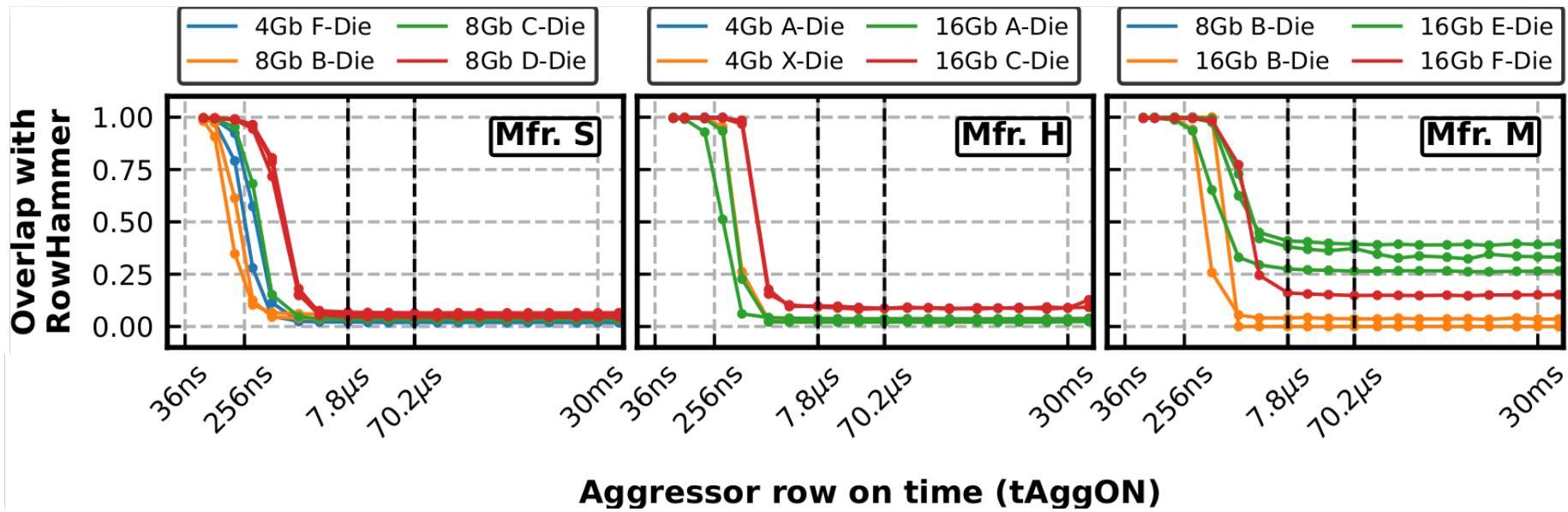
**Figure 157 — Refresh Command Timing (Example of 1x Refresh mode)**

JESD79-4C

# Backup Slide

## Cells vulnerable to RowPress vs RowHammer

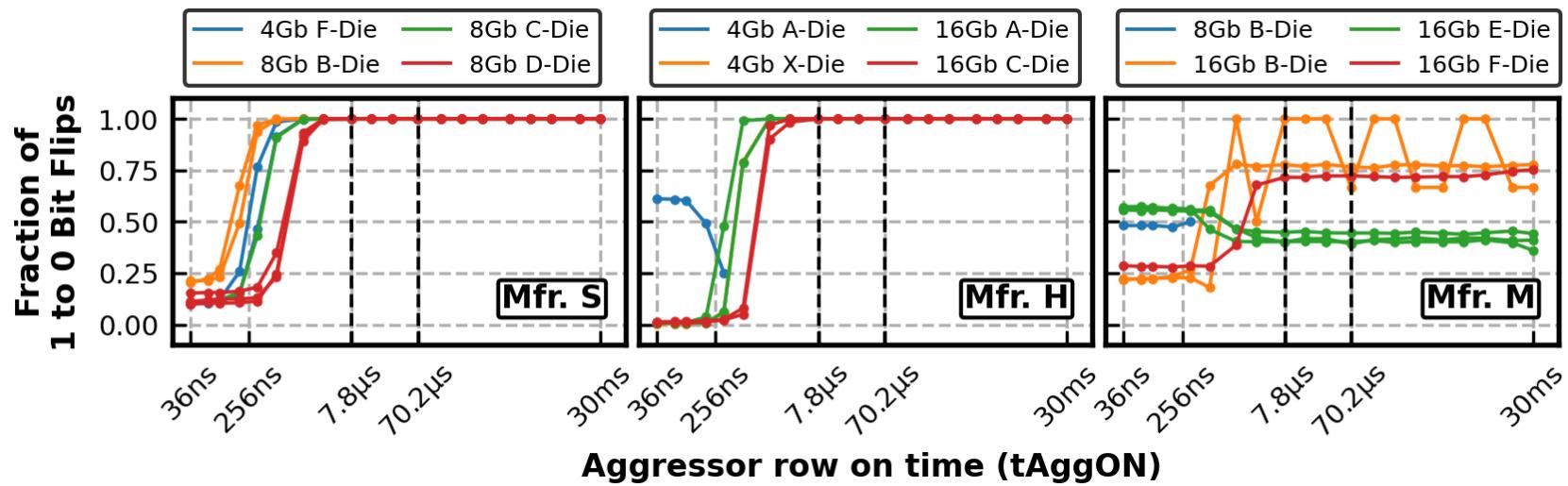
- Cells vulnerable to RowPress (RowHammer) are those that flip @ ACmax
- Overlap = 
$$\frac{\text{Number of Cells Vulnerable to Both RowPress and RowHammer}}{\text{Number of Cells Vulnerable to RowPress}}$$



# Backup Slide

## Directionality of RowHammer and RowPress

I



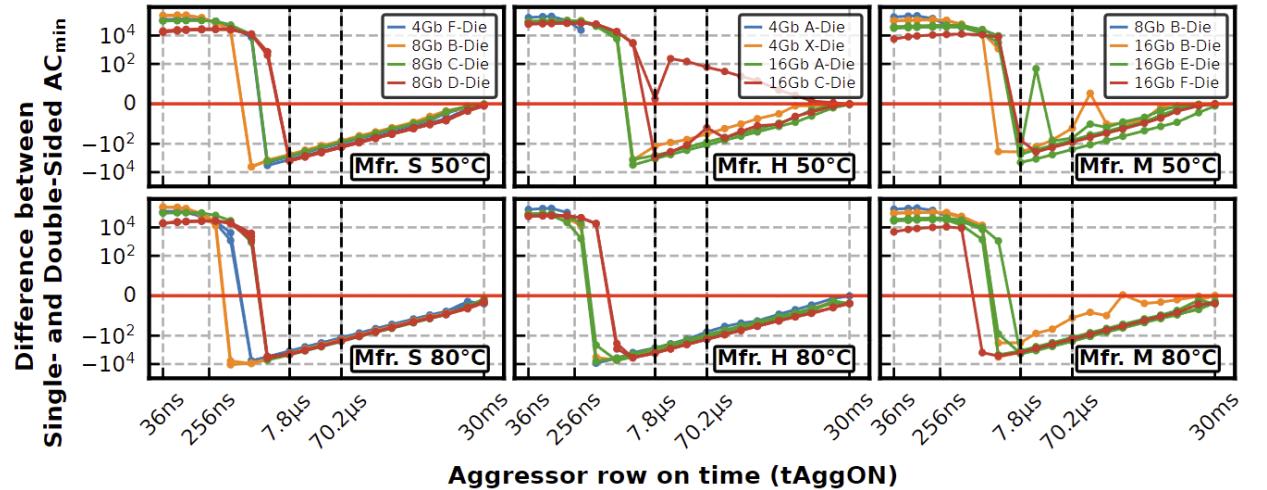
The majority of RowHammer bitflips are 1 to 0  
The majority of RowPress bitflips are 0 to 1

RowPress and RowHammer bitflips have opposite directions

# Backup Slide

## Effectiveness of single-sided vs double-sided RowPress

- Data points for single-sided



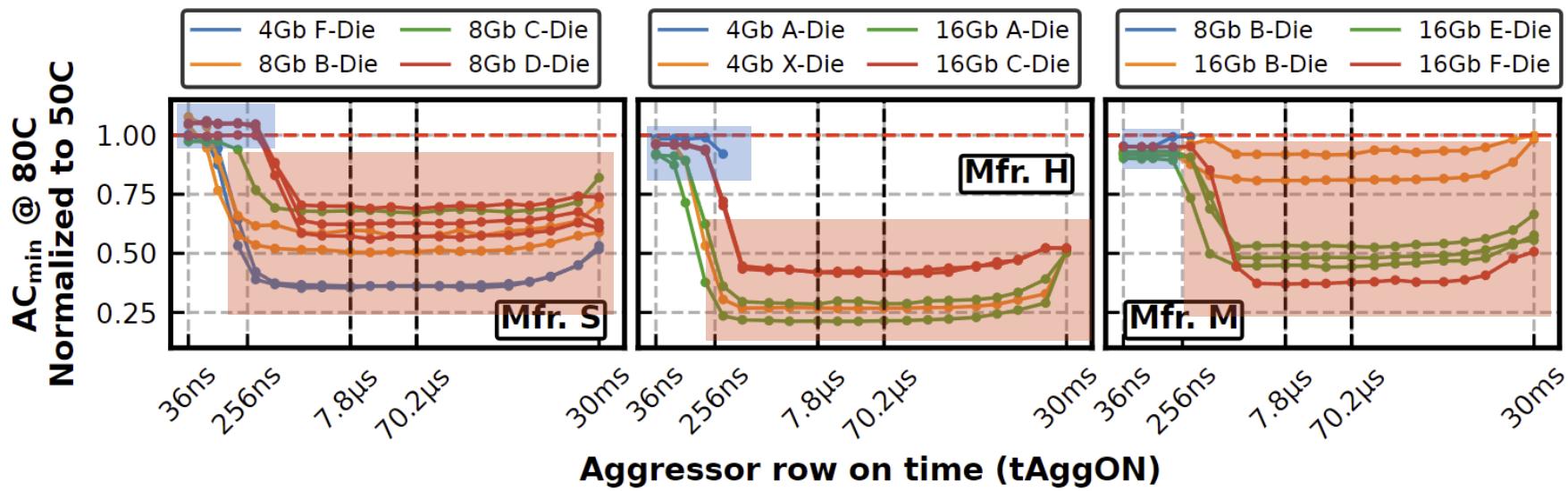
As tAggON increases beyond a certain level, **single-sided RowPress becomes more effective** compared to double-sided

Different from RowHammer where **double-sided is more effective**

# Backup Slide

## Sensitivity to temperature

- Data point below 1 means fewer activations to cause bitflips @ 80°C compared to 50°C



RowPress gets worse as temperature increases,  
which is very different from RowHammer

# Backup Slide

**RowPress significantly reduces ACmin as tAggON increases**

**Most Cells Vulnerable to RowPress  
are NOT vulnerable to RowHammer**

**RowPress and RowHammer bitflips have opposite directions**

As tAggON increases beyond a certain level, **single-sided RowPress becomes more effective** compared to double-sided



# 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*

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

**SAFARI**

**ETH** Zürich

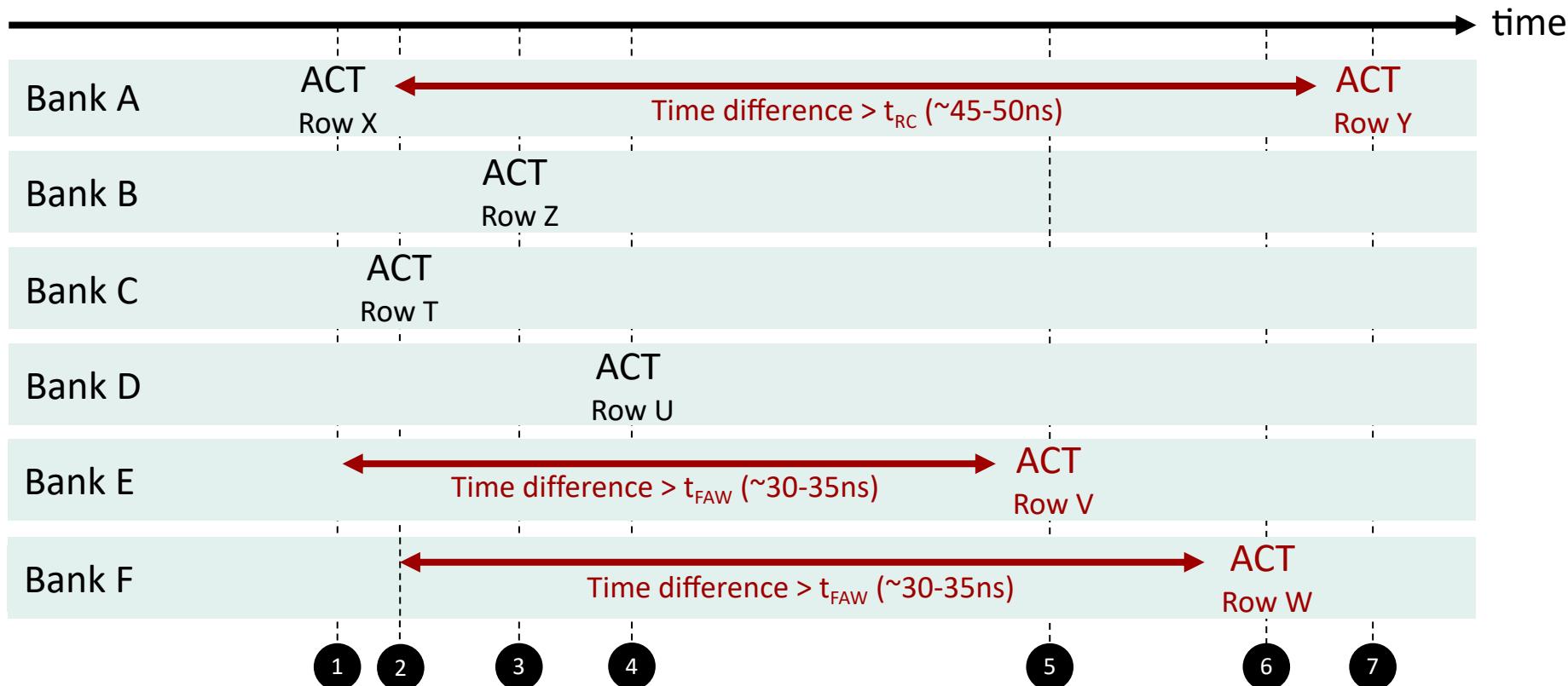
# *BlockHammer*

*Preventing RowHammer at Low Cost  
by Blacklisting Rapidly-Accessed DRAM Rows*

**Backup Slides**

# Timing Constraints for DRAM Row Activations

- Timing row activations is critical to meet **reliability** and **power** constraints.
- Two timing constraints **limit row activation rates**.



$t_{RC}$  : Minimum delay between two consecutive activations in a bank.

$t_{FAW}$ : Rolling time window in which at most four rows can be activated in a rank.

# BlockHammer Hardware Complexity

- RowBlocker
  - RowBlocker-BL: Implemented per-bank
    - 1K counters in a CBF
    - 4 H3 hash functions
  - RowBlocker-HB: Implemented per-rank
    - 887 entries
- AttackThrottler
  - Two counters per <Bank, Thread> pair.

# RowHammer Characteristics

- **RowHammer Threshold ( $N_{RH}$ ):**  
The minimum row activation count in a refresh window to induce a RowHammer bit-flip.
- **Blast Radius ( $r_{Blast}$ ):**  
The maximum physical distance from the aggressor row at which RowHammer bit-flips can be observed.
- **Blast Impact Factor ( $c_i$ ):**  
Set of coefficients that scale a RowHammer attacks impact on victim rows based on their physical distance to the aggressor row.

# Many-Sided Attacks

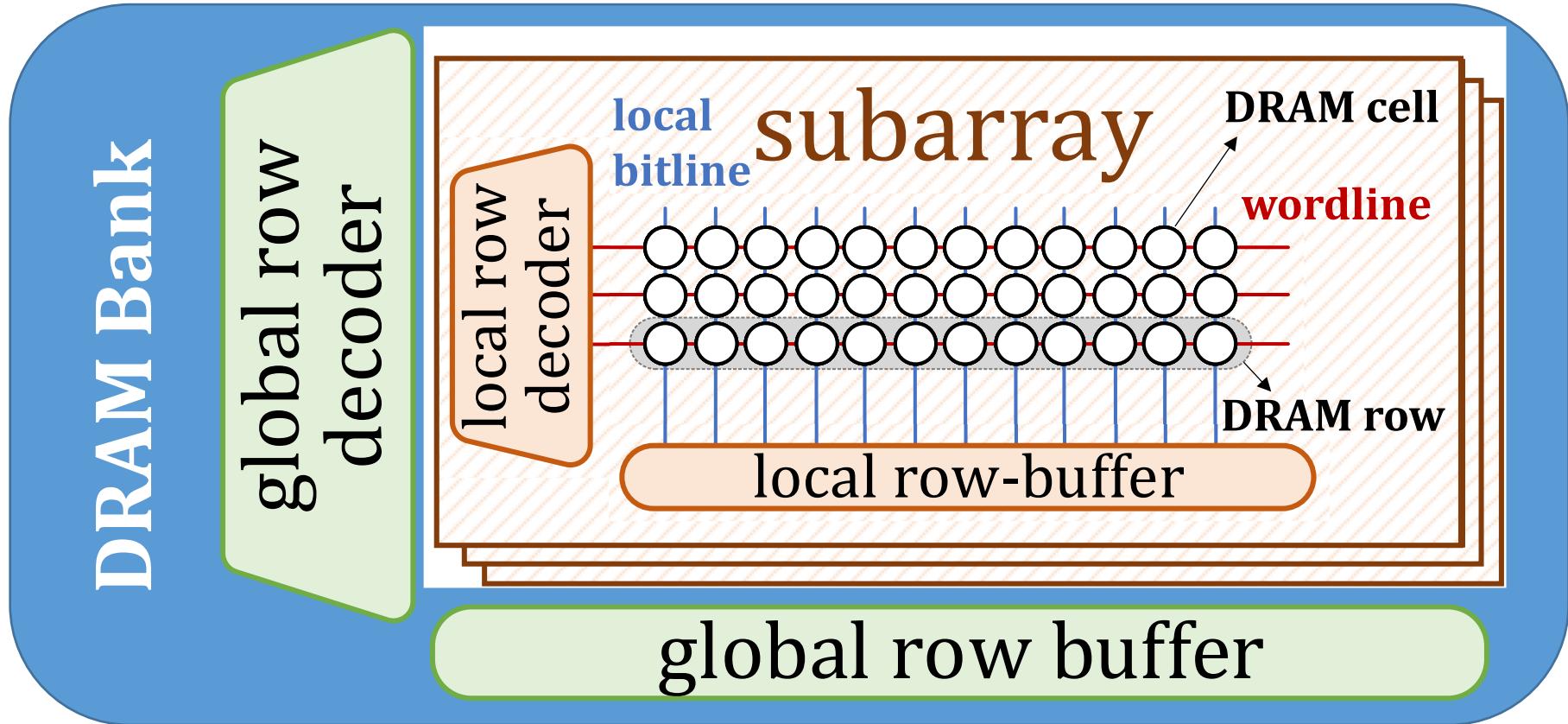
- $N_{RH}$  : RowHammer threshold for single-sided attack.
- $N_{RH}^*$  : Maximum activation count that BlockHammer allows in a refresh window.
- $r_{Blast}$  : Blast radius
- $c_i$  : Blast impact factor
- We configure  $N_{RH}^*$  such that hammering all rows  $N_{RH}^*$  times does not cause bit-flips.

$$2(c_1 + c_2 + c_3 + \dots + c_{r_{Blast}})N_{RH}^* = N_{RH}$$

$$2N_{RH}^* \sum_{i=1}^{r_{Blast}} c_i \leq N_{RH}$$

# DRAM Organization

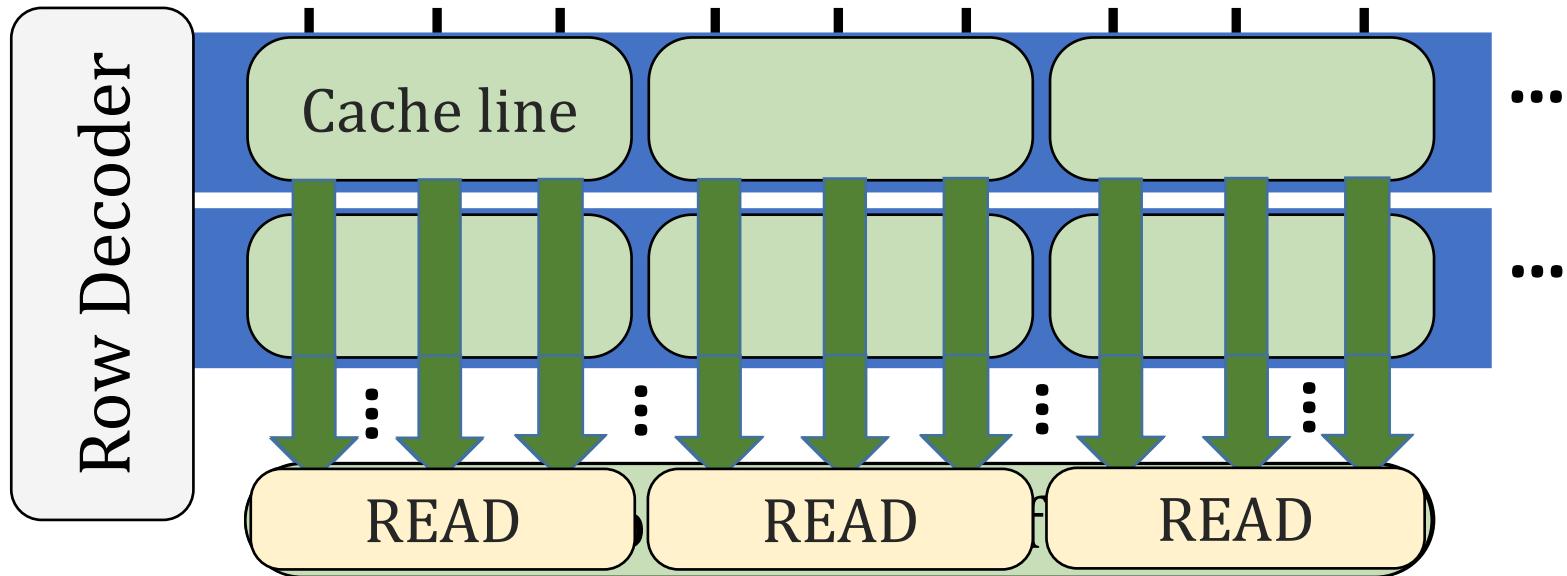
A DRAM bank is hierarchically organized into **subarrays**



Columns of cells in subarrays share a **local bitline**

Rows of cells in a subarray share a **wordline**

# DRAM Operation

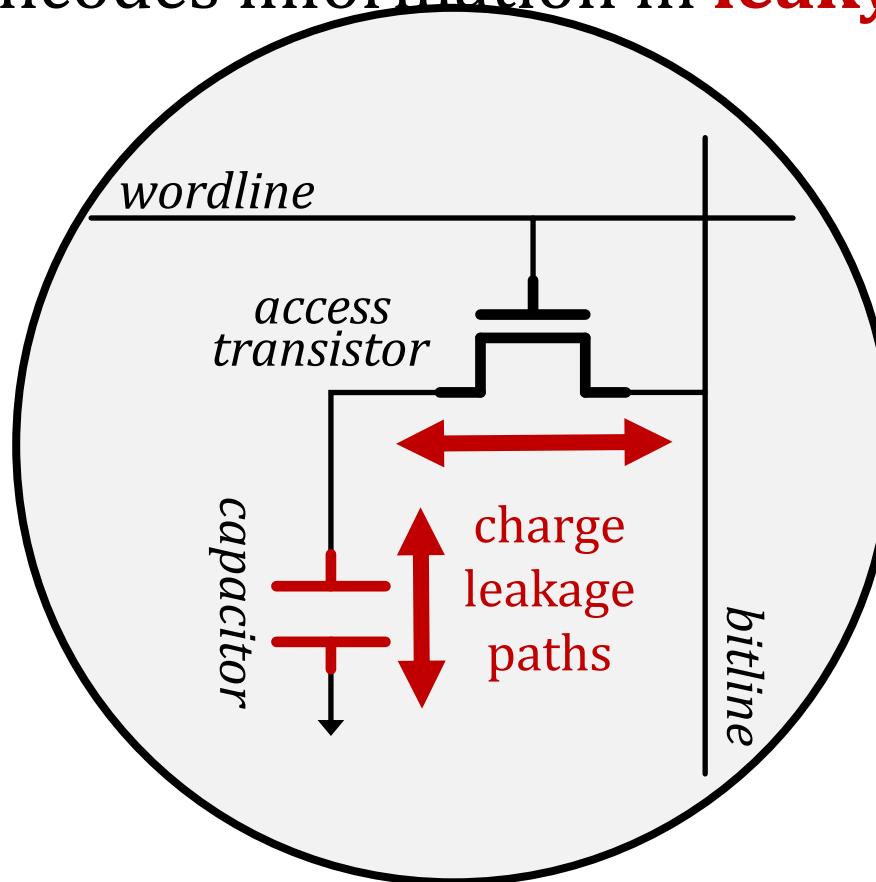


DRAM Command Sequence



# DRAM Cell

Each cell encodes information in **leaky** capacitors



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

# Security Analysis

Epoch Type	$N_{ep-1}$	$N_{ep}$	$N_{epmax}$
$T_0$		$N_{ep} < N_{BL}^*$	$N_{BL}^* - 1$
$T_1$	$< N_{BL}$	$N_{BL}^* \leq N_{ep} < N_{BL}$	$N_{BL} - 1$
$T_2$		$N_{ep} \geq N_{BL}$	$t_{ep}/t_{Delay} - (1 - t_{RC}/t_{Delay})N_{BL}^*$
$T_3$	$\geq N_{BL}$	$N_{ep} < N_{BL}$	$N_{BL} - 1$
$T_4$		$N_{ep} \geq N_{BL}$	$t_{ep}/t_{Delay}$

**Table 2: Five possible epoch types that span all possible memory access patterns, defined by the number of row activations the aggressor row can receive in the previous epoch ( $N_{ep-1}$ ) and in the current epoch ( $N_{ep}$ ).  $N_{epmax}$  shows the maximum value of  $N_{ep}$ .**

- 
- |     |   |  |
|-----|---|--|
| (1) | $N_{RH} \leq \sum(n_i \times N_{epmax}),$ | $t_{REFW} \geq t_{ep} \times \sum n_i$ |
| (2) | $n_{0,1,2} \leq n_0 + n_1 + n_3;$         | $n_{3,4} \leq n_2 + n_4;$              |
| (3) | $\forall n_i \geq 0$                      |  |
- 

**Table 3: Necessary constraints of a successful attack.**

No permutation of epochs can satisfy  
the necessary constraints of a successful attack