

Secure and Sustainable System Scaling

Abdullah Giray Yağlıkçı

UC Berkeley
August 5, 2025

Brief Self Introduction

- Researcher
 - SAFARI Research Group since 2016
 - ETH Zurich (Sep 2024 - ongoing)
Postdoc Researcher and Lecturer
 - ETH Zurich (Feb 2018 – Aug 2024)
PhD Student and Teaching and Research Assist.
 - Intel Labs (Aug 2017 – Feb 2018)
Research Assist.
 - Carnegie Mellon University (Aug 2016 – Aug 2017)
Research Assist.
 - Earlier:
 - Univ. of Notre Dame, IN, USA (Aug 2014-Aug 2016)
 - TOBB Univ., Ankara, Turkey (Aug 2011-Aug 2014)
- Research interests:
 - Computer arch., memory subsystem
 - Hardware security, dependability, and sustainability
 - Hardware/software cooperation
 - Testing and verification
 - ...



Abdullah Giray Yaglikci

Postdoc and Lecturer

ETH Zurich

<https://agyaglikci.github.io>

agirayyaglikci@gmail.com

PhD: ETH Zurich, 2024

Incoming Faculty at CISPA

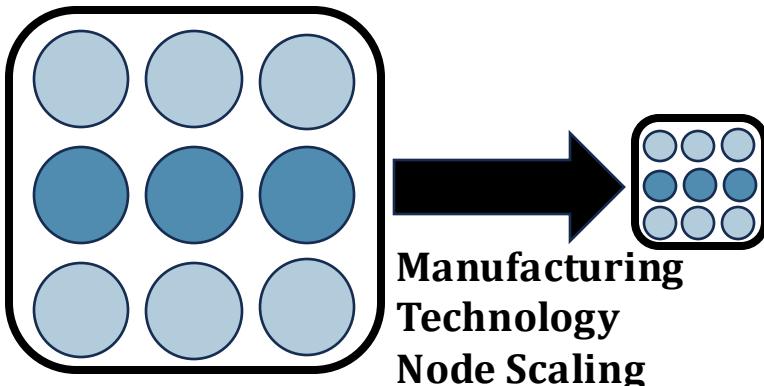
Helmholtz Institute

Center for Information

Security, Privacy, and

Accountability

System Scaling in Two Aspects

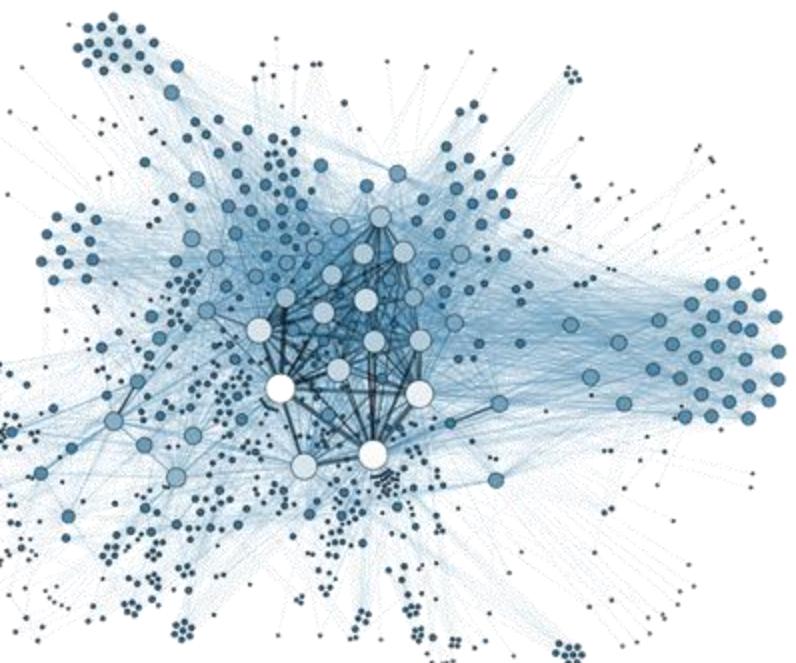


Scaling down manufacturing technology

- Higher performance
- Lower energy consumption
- Circuit-level failure mechanisms
- Mitigations incur
 - high hardware complexity
 - high energy consumption
 - reduced availability

Pros

Cons



Scaling out systems in a distributed manner

- Lower cost of ownership
- Dynamic scaling on demand
- Security and privacy concerns can prevent achieving the full potential
 - Encryption worsens latency and energy
 - Isolation leads to subpar utilization

Pros

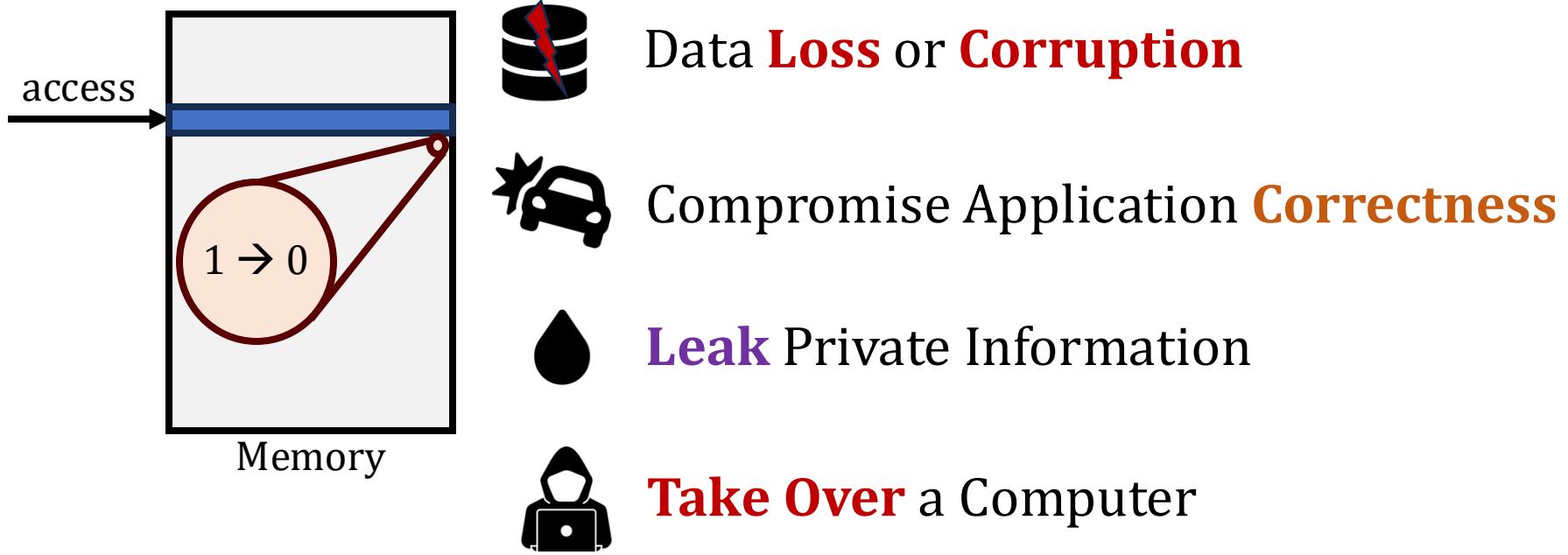
Cons

Systems should scale
robustly and sustainably
as chips get denser
and many users share systems

Towards Robust and Sustainable Scaling

- Pre-PhD Research
 - Understanding and reducing **memory access latency**
 - Understanding and reducing **memory energy consumption**
- PhD Research
 - Understanding **read disturbance** failures in main memory
 - Mitigating **read disturbance** failures in main memory
- Post-PhD Research
 - Improving the **availability of data and resources** while mitigating read disturbance failures in main memory
- Future Research
 - Hardware-software **co-design**
 - **Data-centric** robustness
 - Better understanding of **hardware failures**
for technology node scaling AND scale-out systems

Memory Isolation



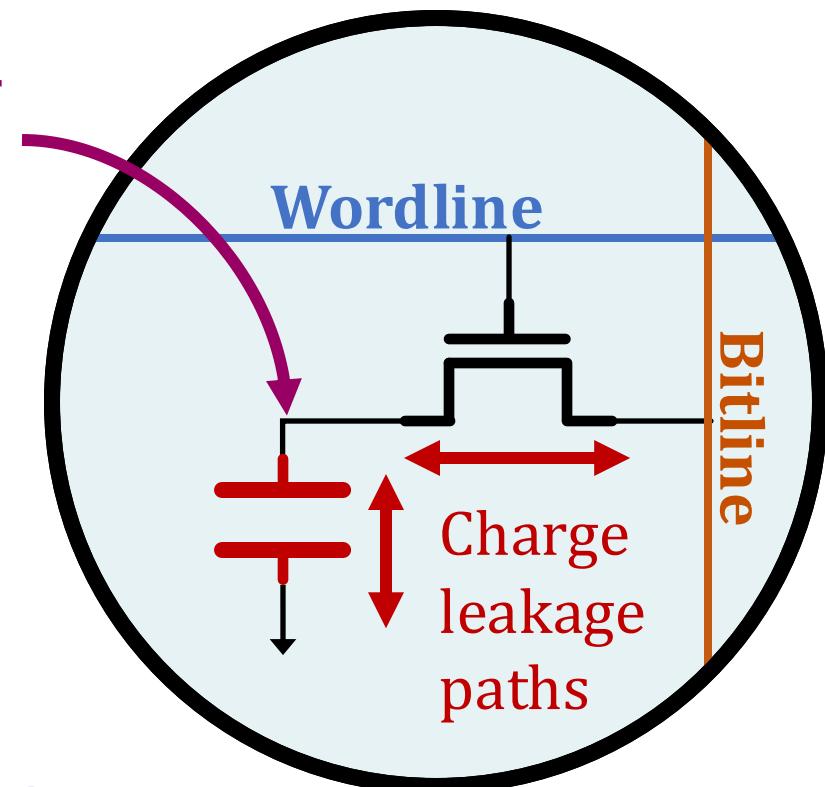
Data integrity and confidentiality issues may emerge

Rule of thumb: An access to one memory address should not have unintended side effects on data stored in other addresses

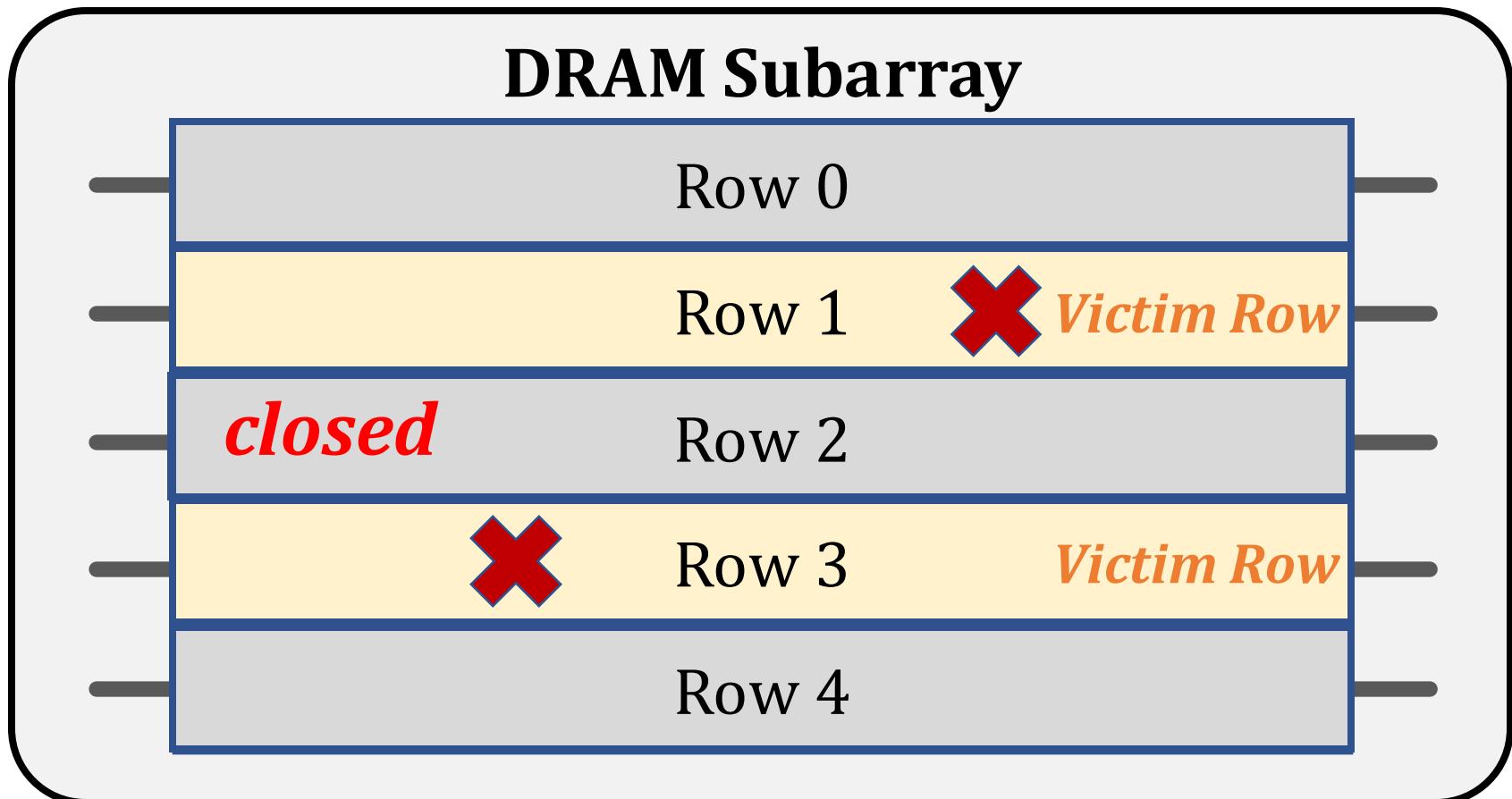
Memory isolation is difficult in modern memory chips

Read Disturbance in Modern Memory Chips

- Prevalent memory technology:
Dynamic Random Access Memory (DRAM)
- DRAM stores **data** in the form of **electrical charge** on a capacitor
- DRAM **leaks charge** over time and needs **periodic refresh**
- **DRAM Read Disturbance:**
Accessing a DRAM cell disturbs other **physically nearby cells** and exacerbates their **charge leakage**



RowHammer: An Example of DRAM Read Disturbance



Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **read disturbance bitflips** in nearby cells and breaks **memory isolation**

One Can Take Over an Otherwise-Secure System in 2015

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

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

News and updates from the Project Zero team at Google

Induce bit flips in page table entries (PTEs).

Gain write access to its own page table,
and hence gain read-write access to all of physical memory.

Exploiting the DRAM rowhammer bug to gain kernel privileges

Fast Forward: Breaking Samsung's Best Practice in 2024

- Salman Qazi and Daniel Moghimi, “[SoothSayer: Bypassing DSAC Mitigation by Predicting Counter Replacement](#),” DRAMSec, 2024

SoothSayer: Bypassing DSAC Mitigation by Predicting Counter Replacement

Salman Qazi
Google

Daniel Moghimi
Google

Abstract—In-DRAM Stochastic and Approximate Counting (DSAC) is a recently published algorithm that aims to mitigate Rowhammer at low cost. Existing in-DRAM counter-based schemes keep track of row activations and issue Targeted Row Refresh (TRR) upon detecting a concerning pattern. However, due to insufficiency of the tracking ability they are vulnerable to attacks utilizing decoy rows. DSAC claims to improve upon existing TRR mitigation by filtering out decoy-row accesses, so they cannot saturate the limited number of counters available for detecting Rowhammer, promising a reliable mitigation without the area cost of deterministic and provable schemes such as per-row activation counting (PRAC).

In this paper, we analyze DSAC and discover some gaps that make it vulnerable to Rowhammer and Rowpress attacks.

The main focus of this work is a novel attack named SoothSayer that targets the counter replacement policy in DSAC by cloning the random number generator. We describe and simulate

literature such as Graphene [18] (implemented in the memory controller) and ProTRR [17] (implemented in DRAM) that utilize frequent item counting schemes. These can account for all Rowhammer activity if the threshold is sufficiently large and enough counters are provided. As the Rowhammer threshold decreases, the number of counters required for a correct implementation increases. According to the authors of DSAC, who are affiliated with a memory vendor, the number of counters used in these implementations are unacceptably large for a memory vendor to implement within their designs. To avoid this cost, deployed counter-based mitigations employ fewer counters than necessary and are often probabilistic. Due to this limitation, recent Rowhammer techniques [2], [8], [13] have managed to bypass TRR with decoy DRAM

More Security Implications (I)

"We can gain unrestricted access to systems of website visitors."

www.iaik.tugraz.at ■

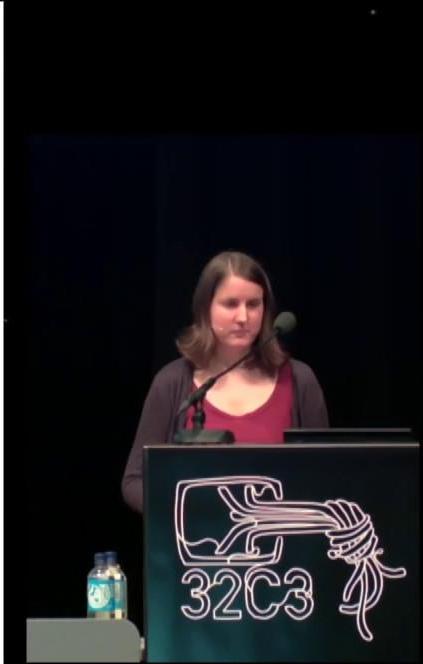
Not there yet, but ...



ROOT privileges for web apps!

29

Daniel Gruss (@lavados), Clémentine Maurice (@BloodyTangerine),
December 28, 2015 — 32c3, Hamburg, Germany



Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA'16)

More Security Implications (II)

“Can gain control of a smart phone deterministically”

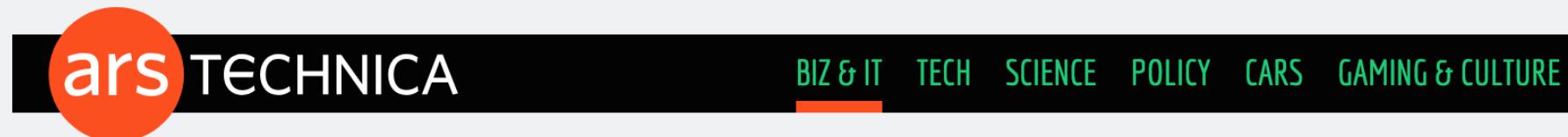


Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS'16

Source: <https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/>

More Security Implications (III)

- Using an integrated GPU in a mobile system to remotely escalate privilege via the WebGL interface. [IEEE S&P 2018](#)



"GRAND PWNING UNIT" —

Drive-by Rowhammer attack uses GPU to compromise an Android phone

JavaScript based GLitch pwns browsers by flipping bits inside memory chips.

Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU

Pietro Frigo
Vrije Universiteit
Amsterdam
p.frigo@vu.nl

Cristiano Giuffrida
Vrije Universiteit
Amsterdam
giuffrida@cs.vu.nl

Herbert Bos
Vrije Universiteit
Amsterdam
herbertb@cs.vu.nl

Kaveh Razavi
Vrije Universiteit
Amsterdam
kaveh@cs.vu.nl

More Security Implications (IV)

- Rowhammer over RDMA (I)

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

THROWHAMMER —

Packets over a LAN are all it takes to trigger serious Rowhammer bit flips

The bar for exploiting potentially serious DDR weakness keeps getting lower.

DAN GOODIN - 5/10/2018, 5:26 PM

Throwhammer: Rowhammer Attacks over the Network and Defenses

Andrei Tatar
VU Amsterdam

Radhesh Krishnan
VU Amsterdam

Herbert Bos
VU Amsterdam

Elias Athanasopoulos
University of Cyprus

Kaveh Razavi
VU Amsterdam

Cristiano Giuffrida
VU Amsterdam

More Security Implications (V)

- Rowhammer over RDMA (II)



Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests

Nethammer: Inducing Rowhammer Faults through Network Requests

Moritz Lipp
Graz University of Technology

Daniel Gruss
Graz University of Technology

Misiker Tadesse Aga
University of Michigan

Clémentine Maurice
Univ Rennes, CNRS, IRISA

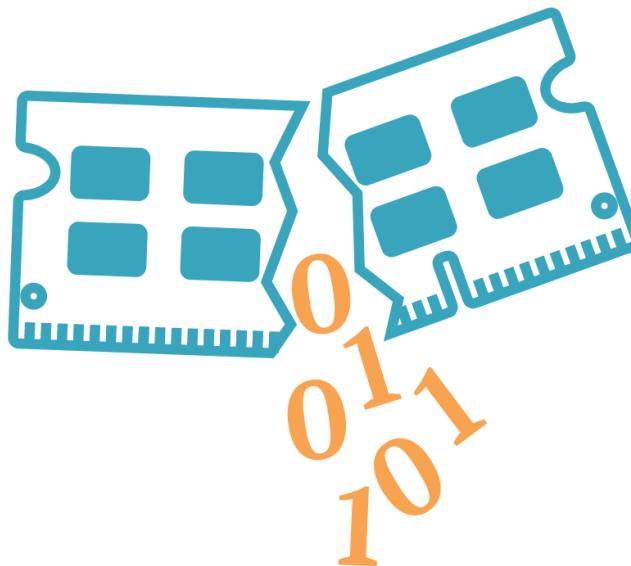
Lukas Lamster
Graz University of Technology

Michael Schwarz
Graz University of Technology

Lukas Raab
Graz University of Technology



More Security Implications (VII)



RAMBleed

RAMBleed: Reading Bits in Memory Without Accessing Them

Andrew Kwong

University of Michigan

ankwong@umich.edu

Daniel Genkin

University of Michigan

genkin@umich.edu

Daniel Gruss

Graz University of Technology

daniel.gruss@iaik.tugraz.at

Yuval Yarom

University of Adelaide and Data61

yval@cs.adelaide.edu.au

More Security Implications (VIII)

Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks

Sanghyun Hong, Pietro Frigo[†], Yiğitcan Kaya, Cristiano Giuffrida[†], Tudor Dumitraş

University of Maryland, College Park

†Vrije Universiteit Amsterdam



A Single Bit-flip Can Cause Terminal Brain Damage to DNNs

One specific bit-flip in a DNN's representation leads to accuracy drop over 90%

Our research found that a specific bit-flip in a DNN's bitwise representation can cause the accuracy loss up to 90%, and the DNN has 40-50% parameters, on average, that can lead to the accuracy drop over 10% when individually subjected to such single bitwise corruptions...

[Read More](#)

More Security Implications (IX)

DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao

University of Central Florida

fan.yao@ucf.edu

Adnan Siraj Rakin

Arizona State University

asrakin@asu.edu

Deliang Fan

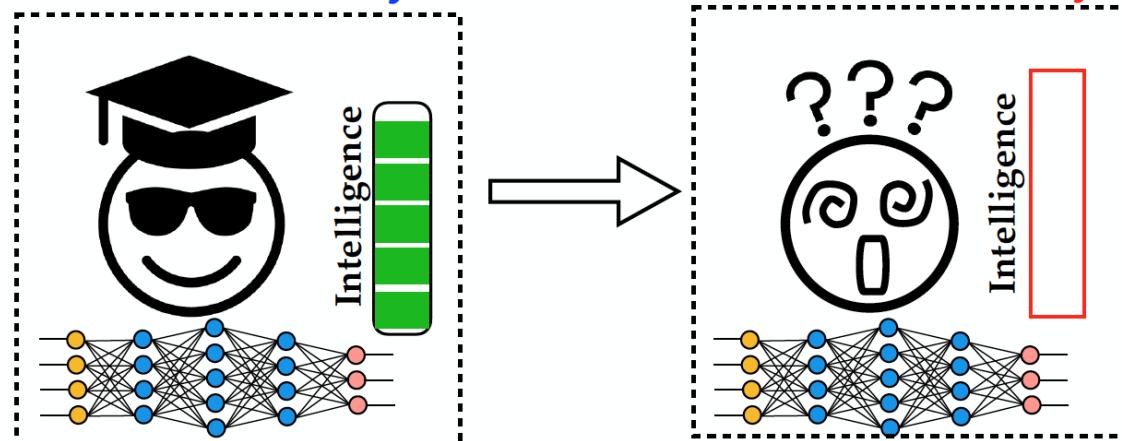
Arizona State University

dfan@asu.edu

Degrade the **inference accuracy** to the level of **Random Guess**

Example: ResNet-20 for CIFAR-10, 10 output classes

Before attack, **Accuracy: 90.2%** After attack, **Accuracy: ~10% (1/10)**



More Security Implications (XI)

- [Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks](#). Cojocar, L. .; Razavi, K.; Giuffrida, C.; and Bos, H. In *S&P*, May 2019 *Best Practical Paper Award, Pwnie Award Nomination for Most Innovative Research* [Paper] [Slides]

Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks

Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, Herbert Bos
Vrije Universiteit Amsterdam

*Thus, many believed that Rowhammer on ECC memory, even if plausible in theory, is simply impractical. This paper shows this to be false: while harder, **Rowhammer attacks are still a realistic threat even to modern ECC-equipped systems.***

More Security Implications (XII)

Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,
["Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications."](#) *MICRO*, 2021. [[Slides \(pptx\)](#) [\(pdf\)](#)] [[Short Talk Slides \(pptx\)](#) [\(pdf\)](#)] [[Lightning Talk Slides \(pptx\)](#) [\(pdf\)](#)] [[Full Talk](#) (25 mins)] [[Lightning Talk](#) (1.5 mins)]

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

Hasan Hassan[†]

[†]*ETH Zürich*

Yahya Can Tuğrul^{†‡}

Kaveh Razavi[†]

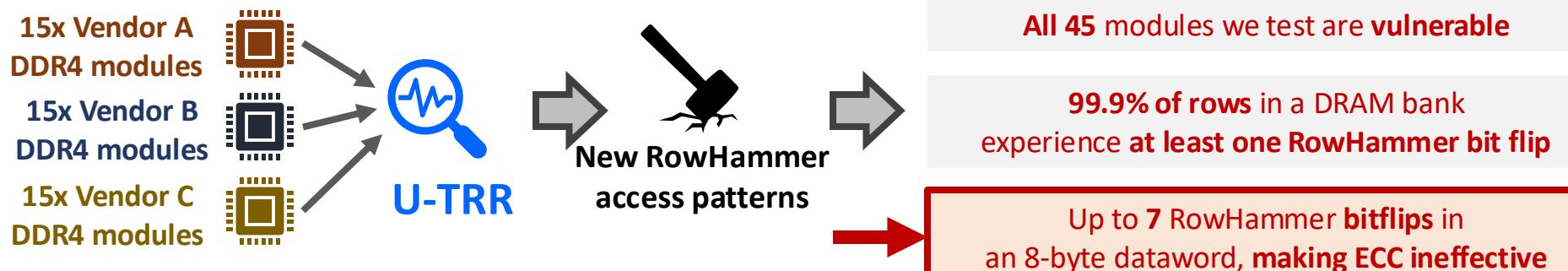
[‡]*TOBB University of Economics & Technology*

Jeremie S. Kim[†]

Onur Mutlu[†]

Victor van der Veen^σ

^σ*Qualcomm Technologies Inc.*



TRR does not provide security against RowHammer

U-TRR can facilitate the development of new RowHammer attacks and more secure RowHammer protection mechanisms

RowHammer in DDR5

Patrick Jattke; Max Wipfli; Flavien Solt; Michele Marazzi; Matej Bölcskei; and Kaveh Razavi, [**“ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms,”**](#) in *USENIX Security*, 2024.
[\[Paper\]](#) [\[URL\]](#)

ZENHAMMER: Rowhammer Attacks on AMD Zen-based Platforms

Patrick Jattke[†] Max Wipfli[†] Flavien Solt Michele Marazzi Matej Bölcskei Kaveh Razavi

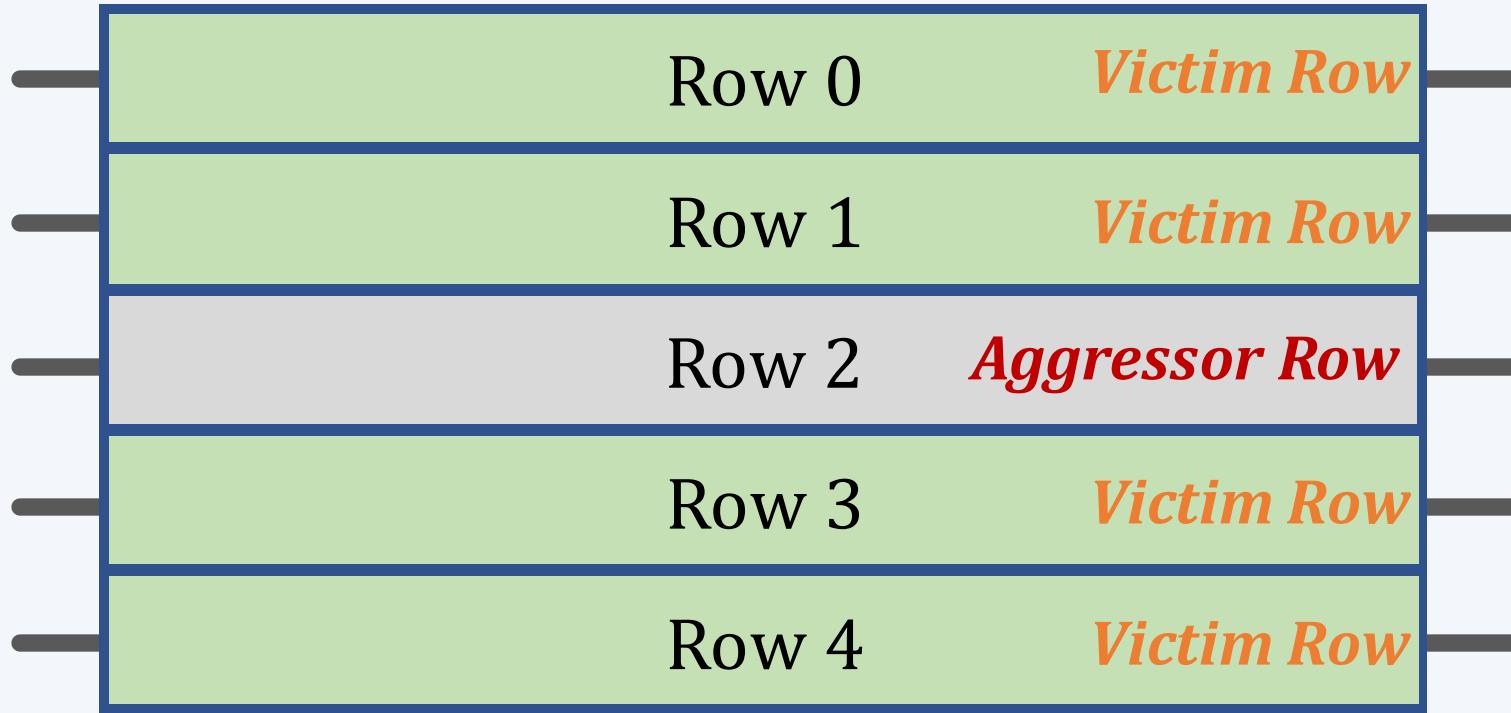
ETH Zurich

[†]Equal contribution first authors

We found bit flips on only 1 of 10 tested devices (S1), suggesting that the changes in DDR5 such as improved Rowhammer mitigations, on-die error correction code (ECC), and a higher refresh rate (32 ms) make it harder to trigger bit flips.

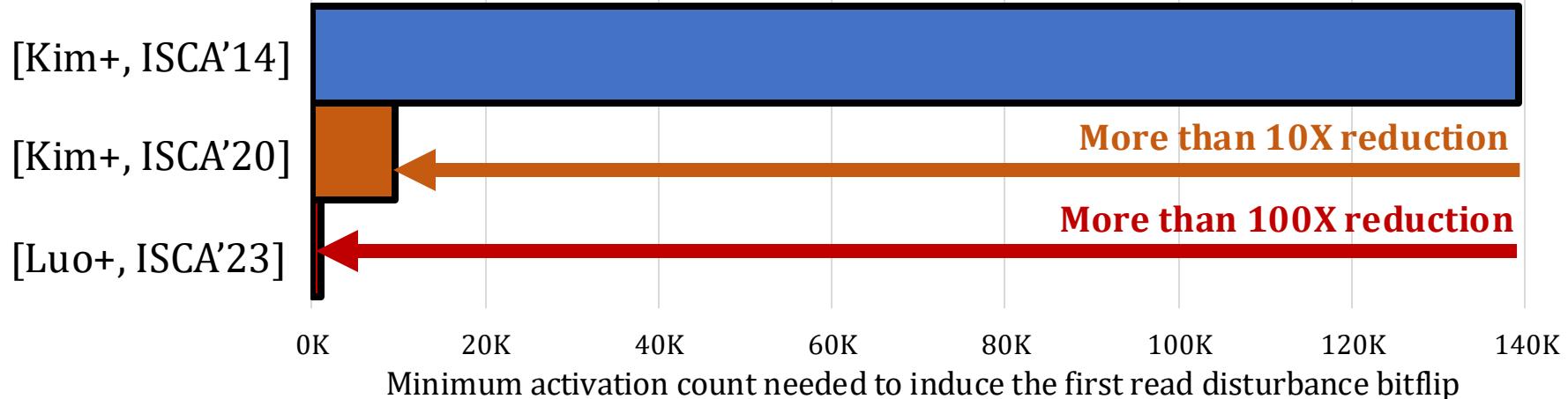
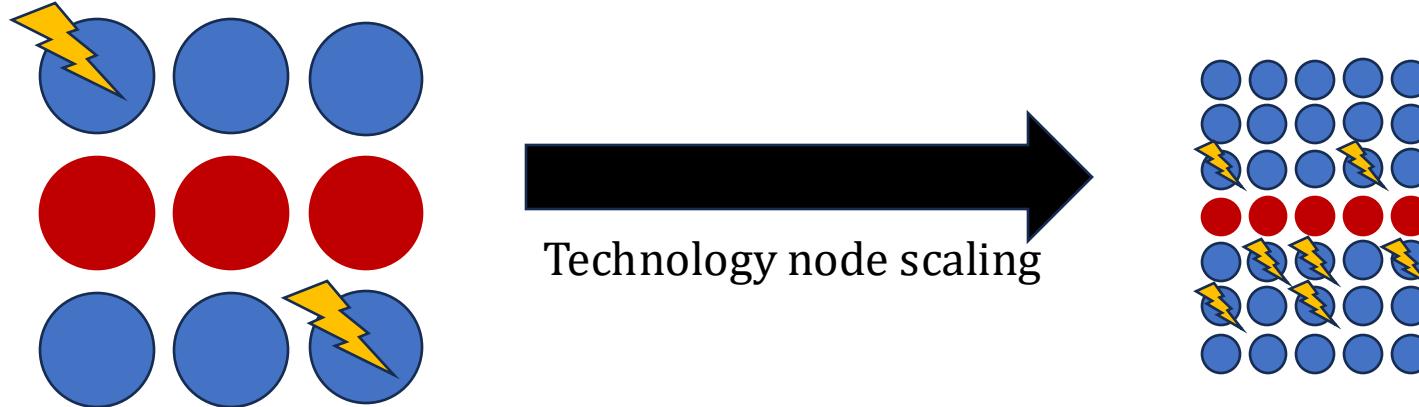
Mitigating RowHammer: Preventive Refresh

DRAM Subarray



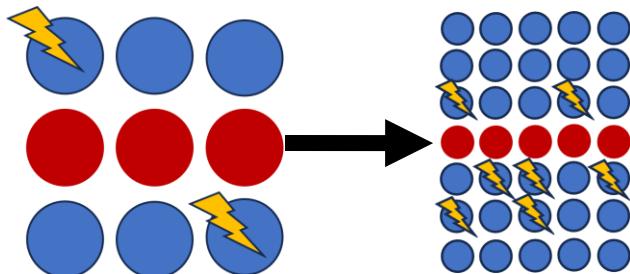
Refreshing potential victim rows
mitigates read disturbance bitflips

DRAM Read Disturbance: A Critical Challenge



DRAM cells become **increasingly**
more vulnerable to read disturbance

Challenge: Scalability with Worsening Vulnerability



DRAM cells become
increasingly more vulnerable
to read disturbance at device level
[Kim+, ISCA'14] [Kim+, ISCA'20] [Luo+, ISCA'23]



Existing solutions become **prohibitively expensive**
or **ineffective** going forward

[Kim+, ISCA'20] [Frigo+, IEEE S&P'20] [Hassan+, MICRO'21]



More efficient, scalable, and comprehensive
solutions to DRAM read disturbance

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

1 build a
detailed understanding
of DRAM read disturbance



2 devise novel solutions
leveraging insights into
DRAM chip internals
and **memory controllers**



Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance



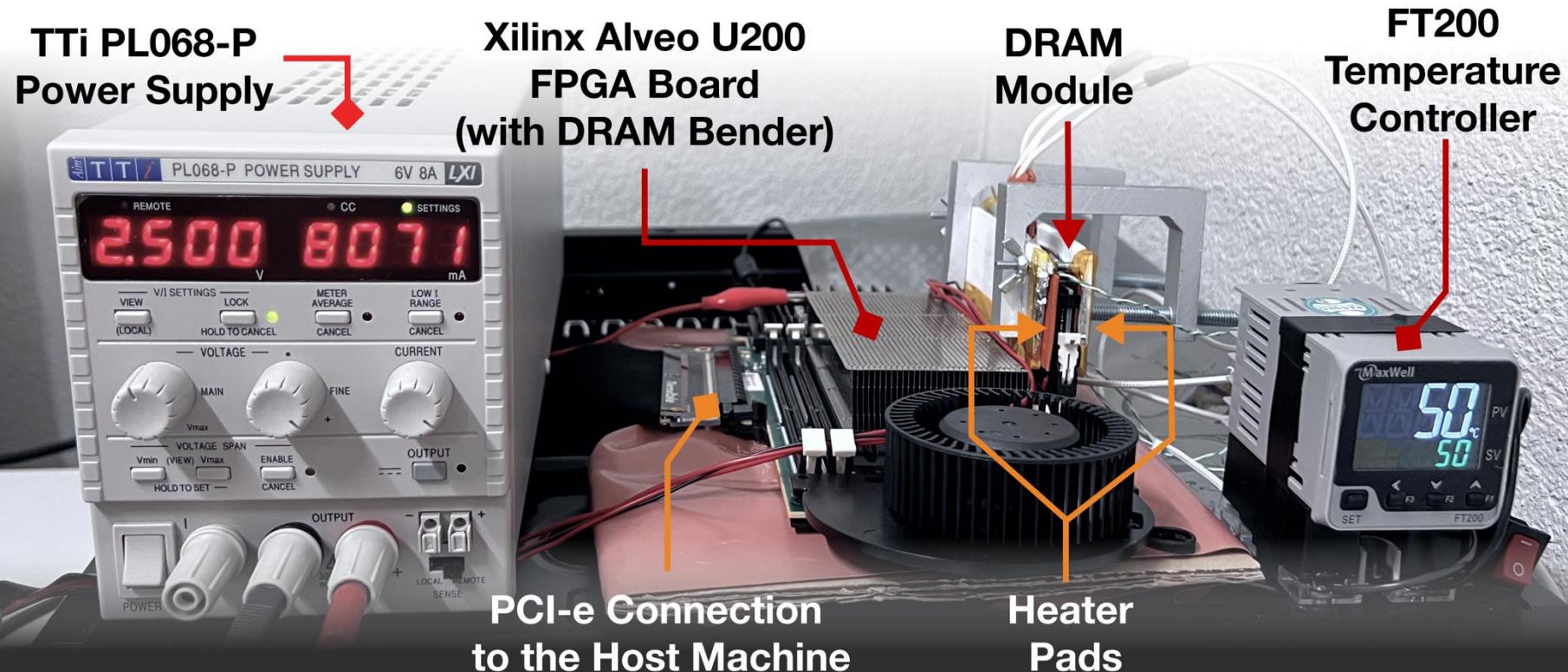
Representative Experimental Findings

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers



DRAM Testing Infrastructure

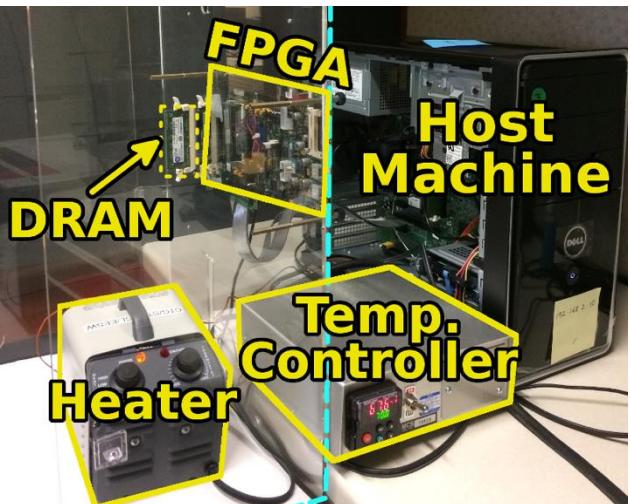
DRAM Bender on a Xilinx Virtex UltraScale+ XCU200



Fine-grained control over **DRAM commands**,
timing parameters ($\pm 1.5\text{ns}$), **temperature ($\pm 0.5^\circ\text{C}$)**,
and **voltage ($\pm 1\text{mV}$)**

*Olgun et al., “[DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips](#),” in TCAD, 2023. [GitHub: <https://github.com/CMU-SAFARI/DRAM-Bender>]

DRAM Bender for Various Memory Standards



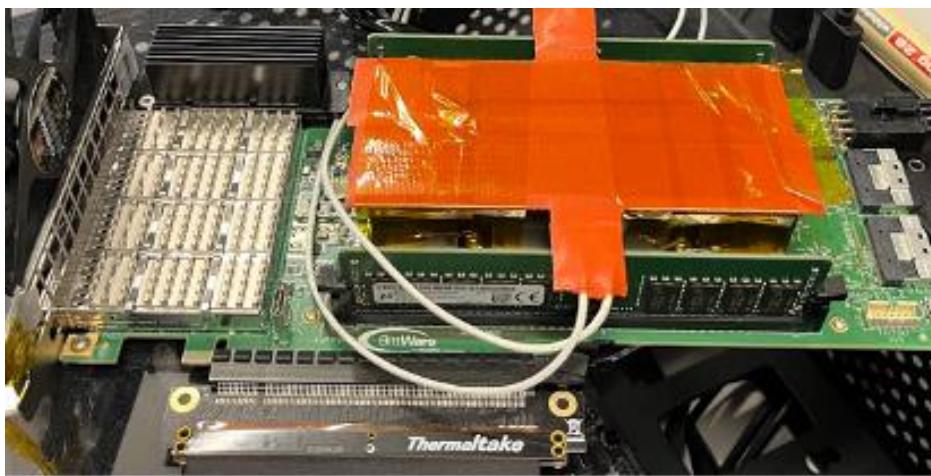
DDR3 DRAM SODIMMs
Xilinx ML605



DDR4 DRAM R/UDIMMs
Xilinx Alveo U200



DDR4 DRAM (SODIMM)
Bittware XUSP3S



HBM2 DRAM Chips
Xilinx Alveo U50

Tested DRAM Chips

- **272 DRAM Chips (24 DDR3 and 248 DDR4 DRAM Chips)**
- 4 major manufacturers:
Micron, Samsung, SK Hynix, and Nanya

2 DRAM standards

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 (-)

4 major manufacturers 272 DRAM chips

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**

Impact of Temperature on DRAM Cells



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

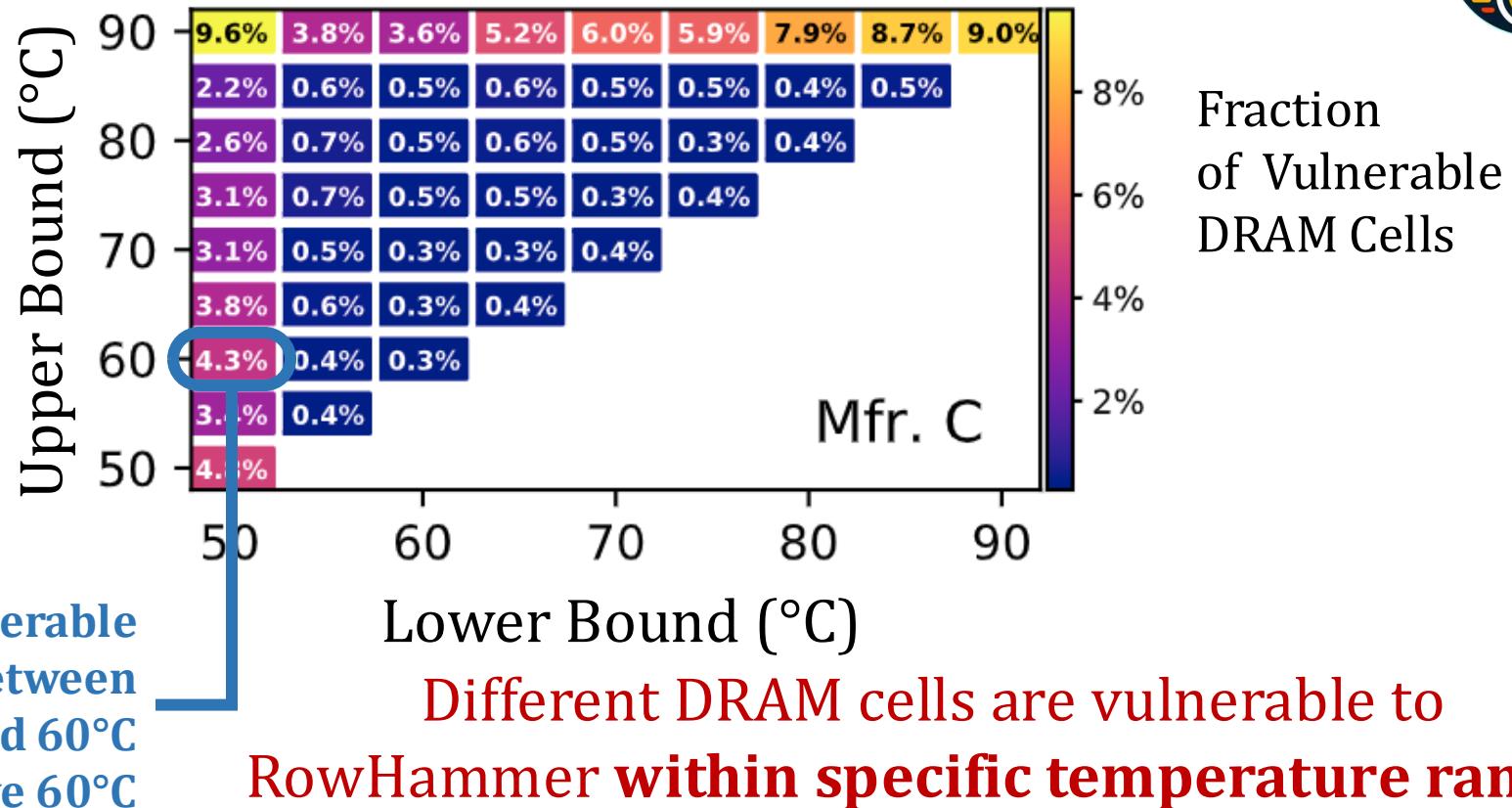
Mfr. A	Mfr. B	Mfr. C	Mfr. D
99.1%	98.9%	98.0%	99.2%

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

Temperature's Impact at DRAM Cell Granularity

Hammer count: 150000 activations to each of the two aggressor rows

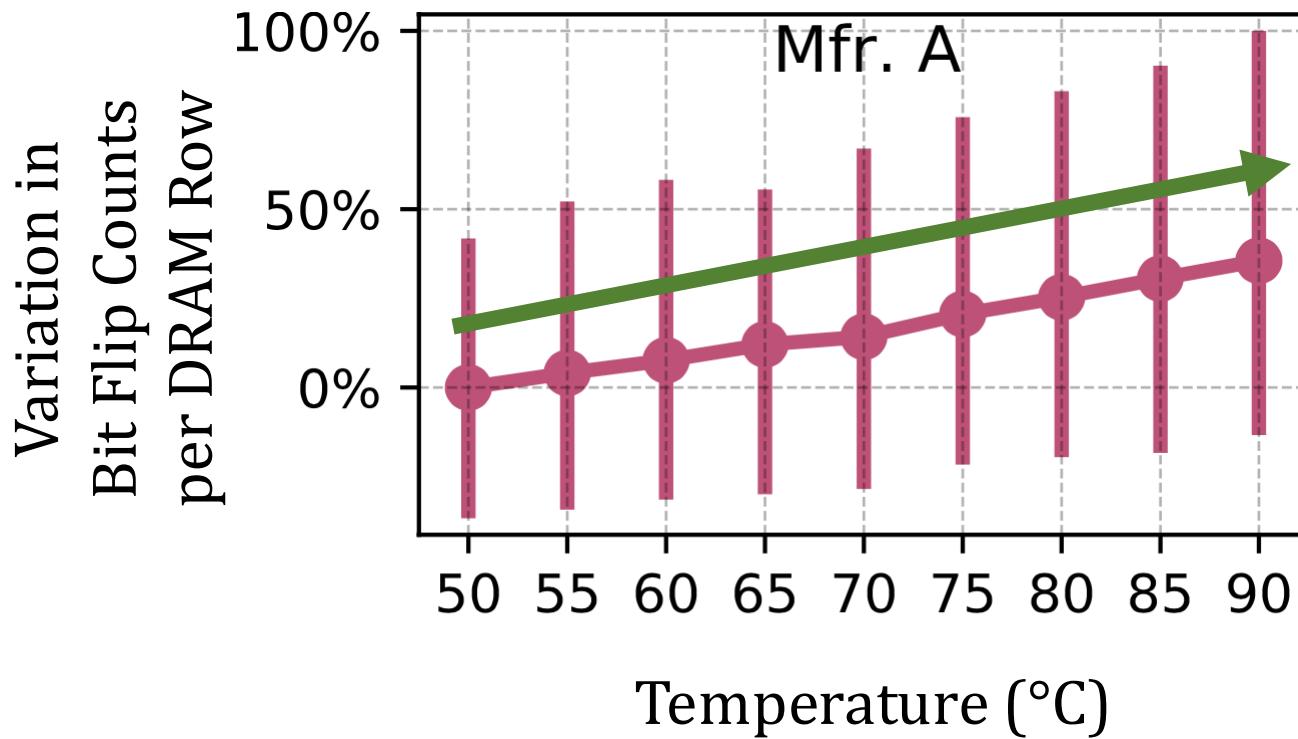
A population density distribution



Implication on testing

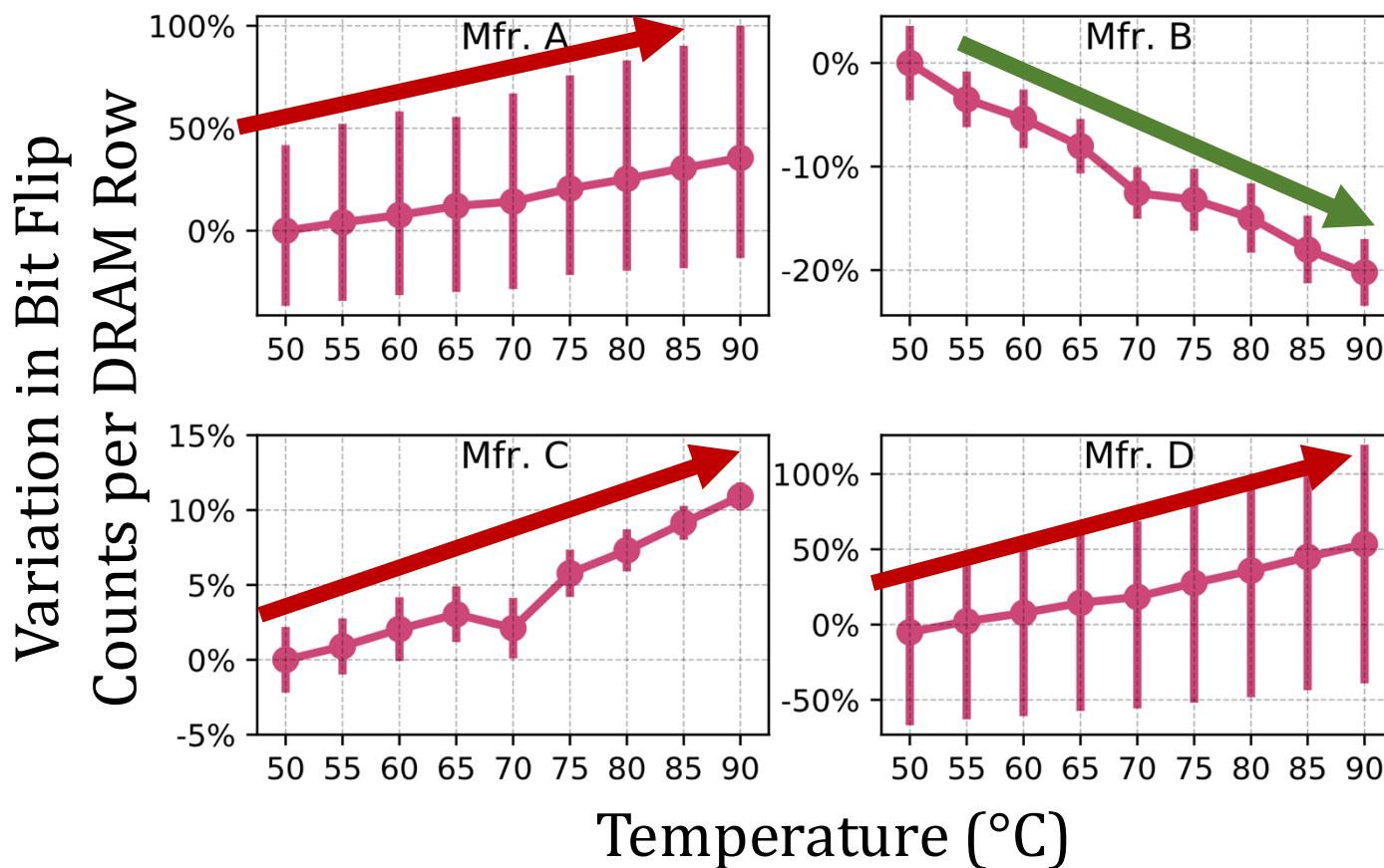
A DRAM cell should be tested at **each possible** operating temperature

Temperature's Impact at DRAM Row Granularity



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

Temperature's Impact at DRAM Row Granularity

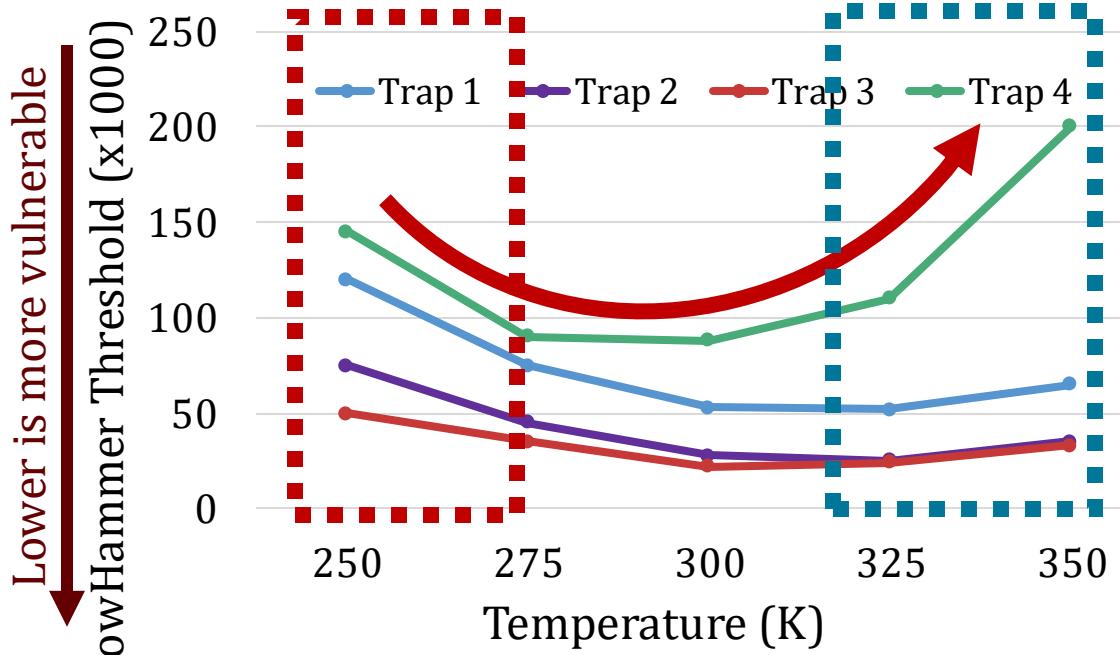


A DRAM row's bit error rate can
either **increase** or **decrease with temperature**
depending on the DRAM manufacturer



Temperature's Impact on RowHammer

- Ours are *empirical* observations
- Similar to the behavior of *Trap Assisted Charge Leakage*
- Increasing temperature
 - helps electrons move easier → higher leakage
→ smaller RowHammer threshold
 - makes traps lose charge → lower trap-assisted leakage
→ larger RowHammer threshold
- Device characteristics determine the dominant effect
- Varies across
 - Cells
 - Chips
 - Manufacturers



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

Temperature Study's Impact: Spying on a Machine's Temperature via RowHammer

- Observe RowHammer BER to measure **change in temperature**
- Find canary cells that exhibit erroneous behavior at **particular temperature points**

IEEE Access

Multidisciplinary | Rapid Review | Open Access Journal

Received 8 February 2024, accepted 2 May 2024, date of publication 4 June 2024, date of current version 14 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409389



SpyHammer: Understanding and Exploiting RowHammer Under Fine-Grained Temperature Variations

LOIS OROSA^{ID 1,2}, (Member, IEEE), ULRICH RÜHRMAIR^{3,4},
A. GİRAY YAĞLIKÇI^{ID 1}, (Graduate Student Member, IEEE), HAOCONG LUO^{ID 1},
ATABERK OLGUN^{ID 1}, (Graduate Student Member, IEEE), PATRICK JATTKE¹,
MINESH PATEL^{ID 1}, (Member, IEEE), JEREMIE S. KIM^{ID 1},
KAVEH RAZAVI¹, (Member, IEEE), AND ONUR MUTLU^{ID 1}, (Fellow, IEEE)

¹Information Technology and Electrical Engineering Department, ETH Zürich, 8092 Zürich, Switzerland

²Galicia Supercomputing Center (CESGA), 15705 Santiago de Compostela, Spain

³Experimental Quantum Physics Department, LMU München, 80539 Munich, Germany

⁴Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT 06269, USA

- Reverse-engineer the DRAM module's model
- Characterize an identical model in lab
- Build a regression model of BER vs temperature
- Identify canary cells
- Monitor canary cells to identify temperature

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance



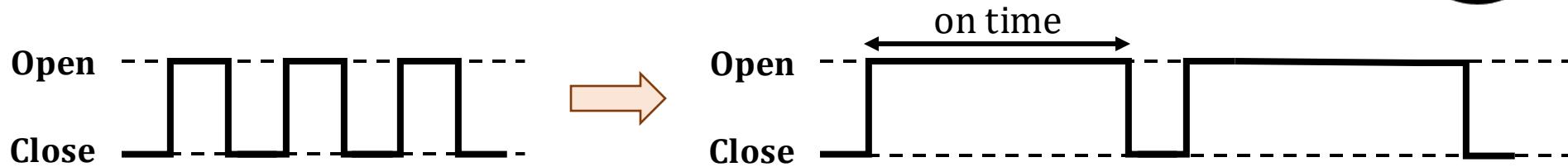
Representative Experimental Findings

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers





Impact of Memory Access Pattern



A smaller hammer count is sufficient to induce bitflips
with increased **aggressor row on time**



Implication on defenses

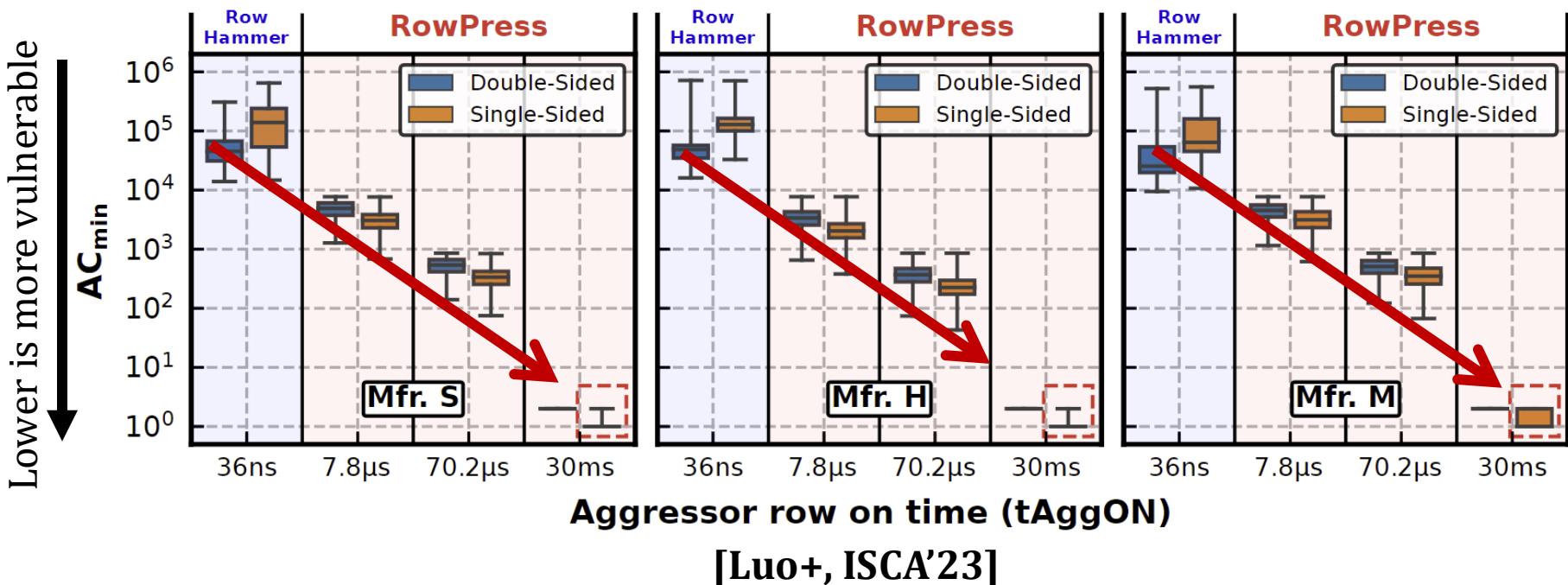
Existing mitigations are ineffective without this insight

Follow-up

This finding paved the way to
the discovery of RowPress [Luo+, ISCA'23]

Our Work Led to the Discovery of RowPress

- 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

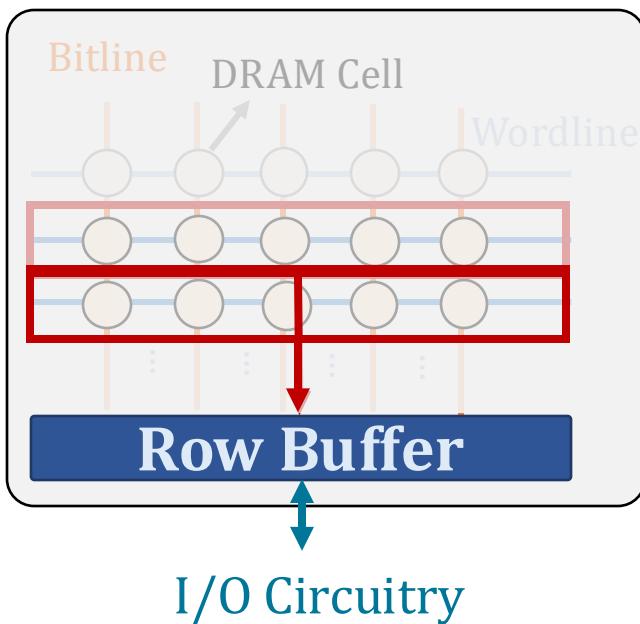


RowPress on HBM2-20nm for row on time of 3.9 us
Read Disturbance Threshold: **335 activations [Olgun+, DSN'24]**

Benign becomes attacker: Even SPEC workloads reach to this activation count

More Background: Row Buffer Locality

DRAM Subarray



1. **Row Hit:** Row buffer has the desired content
 - Service RD/WR from the row buffer (~10ns)
2. **Row Miss:** Row buffer has no content
 - Open the row (~13ns)
 - Service RD/WR from the row buffer (~10ns)
3. **Row Conflict:** Row buffer has different content
 - Close the active row (~13ns)
 - Open the desired row (~13ns)
 - Serve RD/WR from the row buffer (~10ns)

Row buffer hit is significantly faster than miss and conflict

Key Insight: Memory controller keeps a row open
if the row is likely to be accessed again

Key Idea: Keep accessing the aggressor row to incentivize
the memory controller to keep the row open

RowPress Demonstration on a Real-System



Intel Core i5-10400
(Comet Lake)

```
// 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();
```

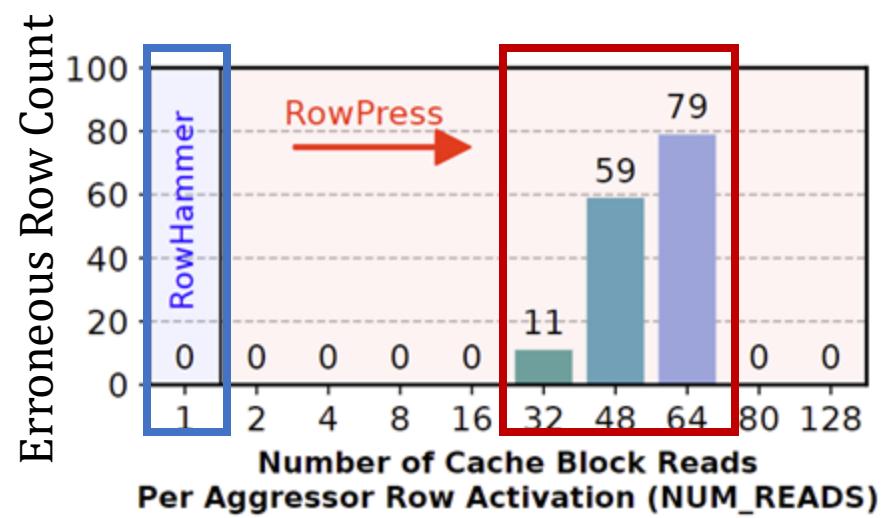
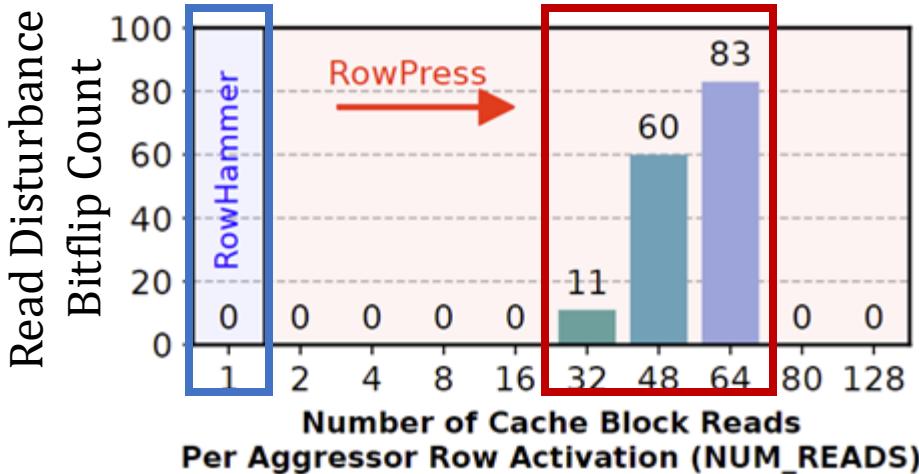


Samsung DDR4 Module
M378A2K43CB1-CTD

(Date Code: 20-10)

w/ TRR RowHammer Mitigation

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



💀 SON OF ROWHAMMER

There's a new way to flip bits in DRAM, and it works against the latest defenses

New technique produces lots of bitflips and could one day help form an attack.

DAN GOODIN – OCT 19, 2023 5:30 AM | 65

“ Samsung is aware of industry concerns around the latest DDR4 DRAM RowPress attack scenario and has been implementing measures in tandem with industry partners to counter such schemes.”

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance

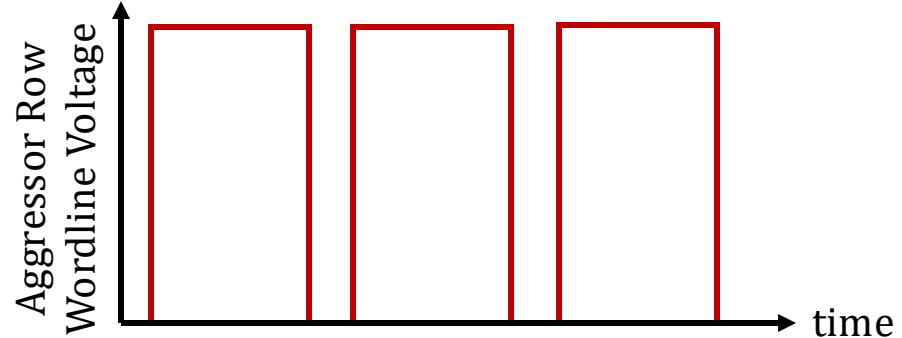
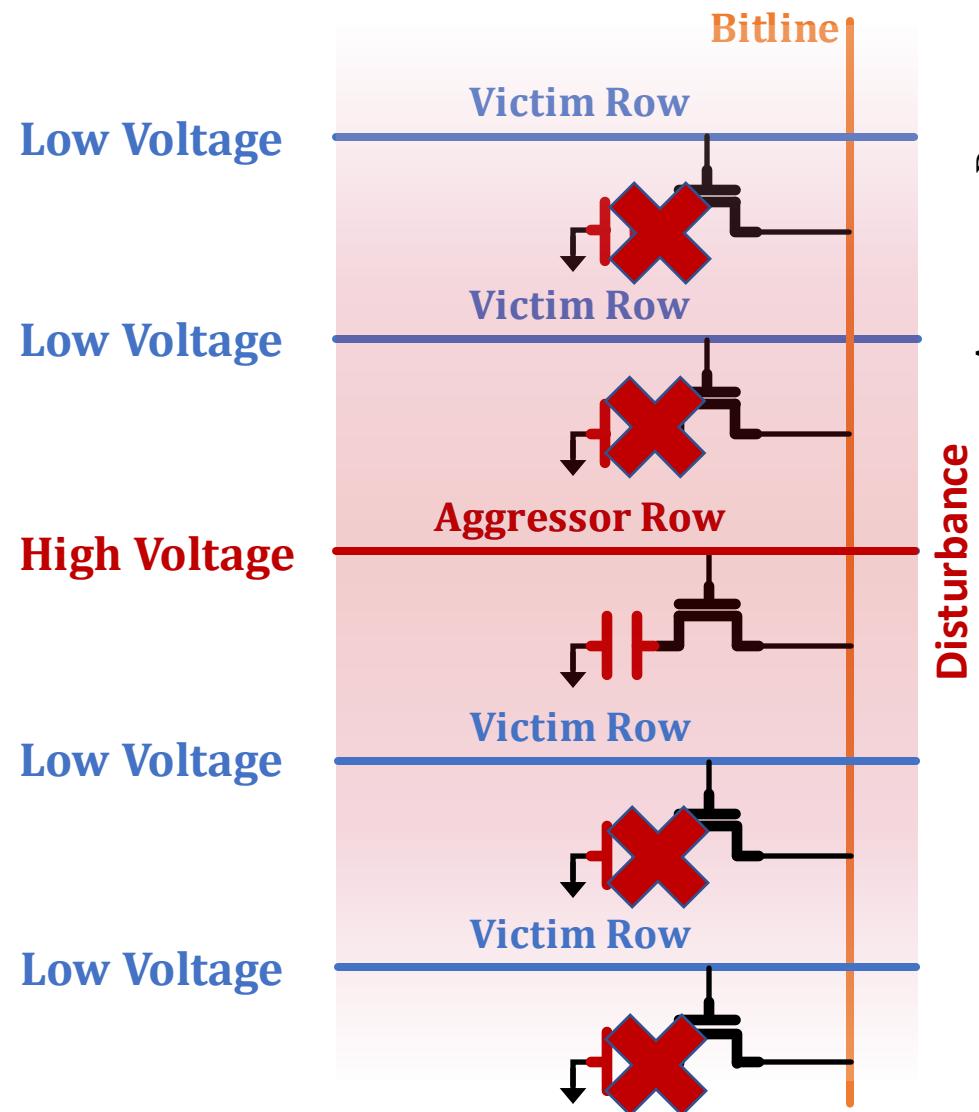


Representative Experimental Findings

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers



A Closer Look into RowHammer

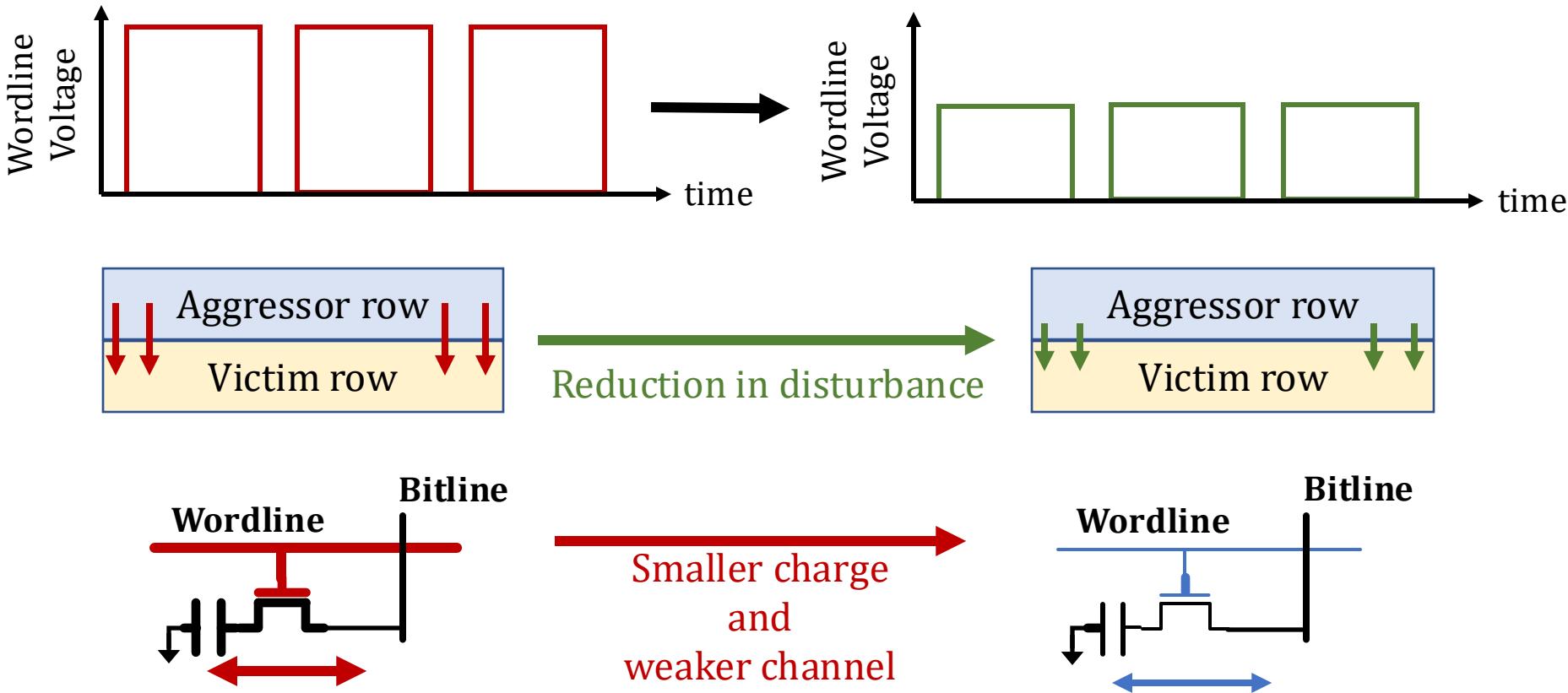


Repeatedly toggling
wordline voltage
leads to
RowHammer bitflips



Effect of Reducing Wordline Voltage

Reducing wordline voltage
can **reduce RowHammer vulnerability**
at the cost of **weaker cells**



Key Results of Voltage Scaling Study



- Reducing wordline voltage can reduce RowHammer vulnerability
 - **15.2% (66.9% max)** fewer bitflips occur
 - **7.4% (85.8% max)** more activations needed to induce a bitflip
- Row activation latency **increases** with reduced **wordline voltage**
 - **208 / 272** DRAM chips have **no bitflips** at **nominal latency**
 - Changing timing constraint from 13.5ns to 24ns **avoids bitflips**
- **More DRAM cells** tend to experience **data retention bitflips** when **wordline voltage is reduced**
 - **216 / 272** DRAM chips have **no bitflips** at **nominal refresh rate**
 - **SECDED ECC** at **nominal refresh rate** avoids bitflips
 - **16% increase in refresh rate** avoids bitflips

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance



Representative Experimental Findings

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers





Another Key Finding: Variation across Rows



Say, bitflips occur at N hammers in these rows

We need at least $2N$ hammers to induce bitflips in these rows

RowHammer vulnerability
significantly varies across DRAM rows

Implication on defenses

Configuring a solution for **the worst-case overprotects** many rows and makes it **expensive**

Follow-up

This finding paved the way to
the design of Svärd [Yaglikci+, HPCA'24] (will be covered)

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance



Representative Novel Solutions

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers





Spatial Variation-Aware RowHammer Defenses

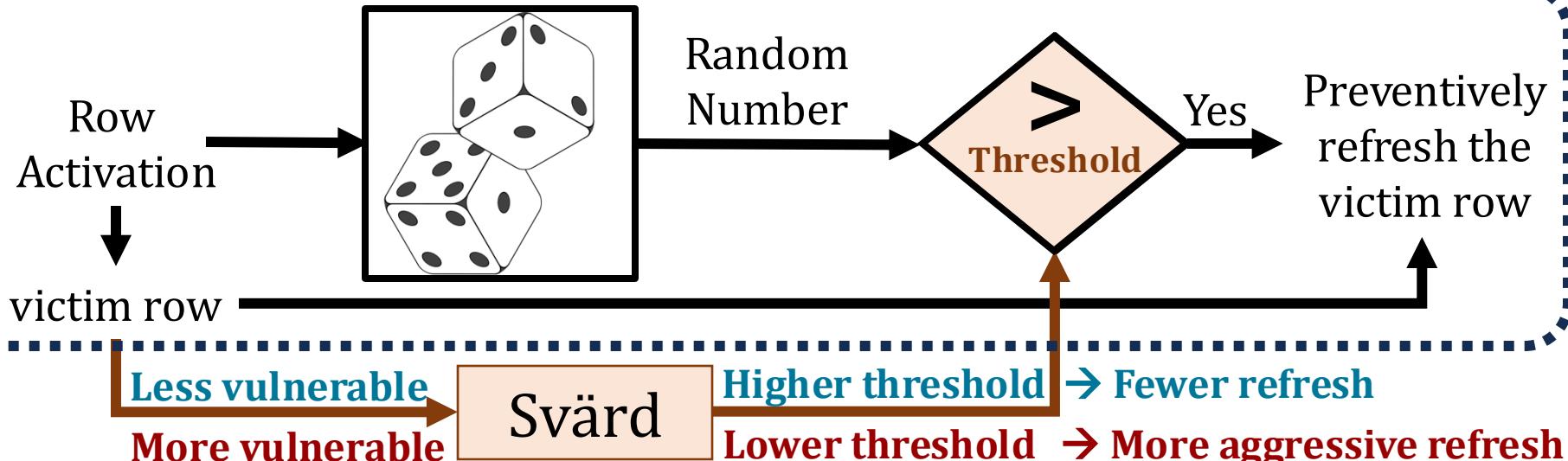
Key insight: Significant variation across DRAM rows

Key Idea: Dynamically tune the aggressiveness of existing solutions to the victim row's read disturbance vulnerability

Svärd: Spatial Variation-Aware Read Disturbance Defenses

Fewer preventive actions (e.g., refresh) for stronger rows

PARA: Probabilistic Row Activation [Kim+, ISCA'14]



Svärd works with many other read disturbance solutions, including:

AQUA

[Saxena+, MICRO'22]

BlockHammer

[Yaglikci+, HPCA'21]

Hydra

[Qureshi+, ISCA'22]

RRS

[Saileshwar+, ASPLOS'22]

Performance Evaluation

- Cycle-level simulations using **Ramulator 2.0** [Luo+, CAL 2023]

- **System Configuration:**

Processor	3.2 GHz, 8 core, 4-wide issue, 128-entry instr. window
Last-Level Cache	64-byte cache line, 8-way set-associative, 8 MB
Memory Scheduler	FR-FCFS
Address Mapping	Minimalistic Open Pages
Main Memory	DDR4, 4 bank group, 4 banks per bank group (16 banks per rank)

- **Workloads:** 120 different **8-core** multiprogrammed workloads from **SPEC CPU2006, SPEC CPU2017, TPC, MediaBench, and YCSB** benchmark suites
- Integrated with **AQUA, BlockHammer, PARA, Hydra, and RRS**
- **HC_{first}:** {4K, 2K, 1K, 512, 256, 128, 64} hammers
The **minimum hammer count** needed to induce **the first bitflip**



Key Results

- Svärd **reduces the performance overhead** of existing solutions and **significantly improves system throughput**

AQUA
[Saxena+, MICRO'22]

1.6x

PARA
[Kim+, ISCA'14]

2.0x

BlockHammer
[Yaglikci+, HPCA'21]

4.9x

RRS
[Saileshwar+, ASPLOS'22]

4.8x

Hydra
[Qureshi+, ISCA'22]

1.1x

- Svärd's hardware complexity:

No
Additional
Latency

In-DRAM
Implementation
0.006%

In-Processor
Implementation
0.027%

Svärd improves system's **availability at low cost**
without compromising data integrity

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

- 1 build a detailed understanding of DRAM read disturbance

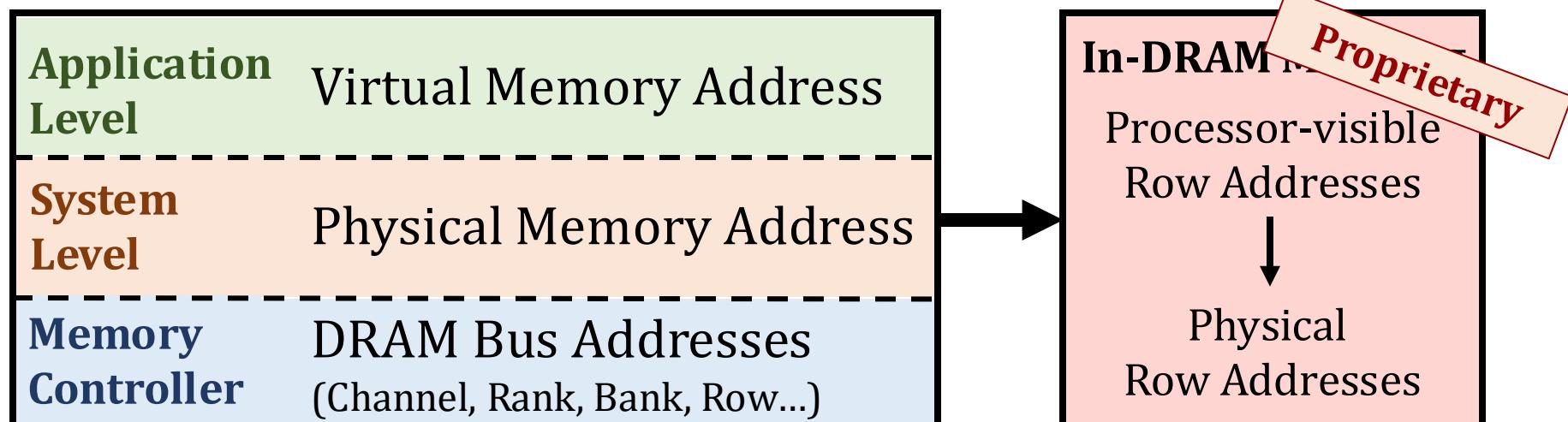


Representative Novel Solutions

- 2 devise novel solutions leveraging insights into DRAM chip internals and memory controllers



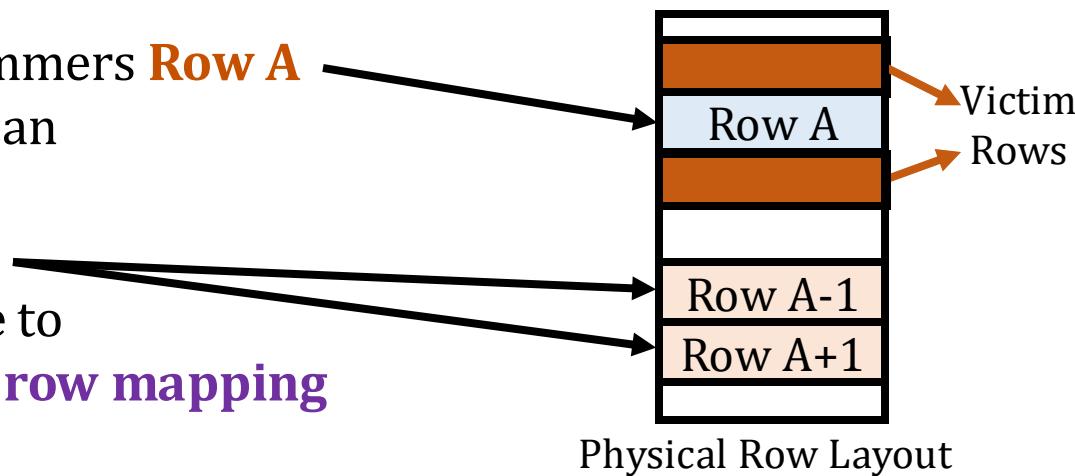
Challenge (2/2): Compatibility



Visible within the processor

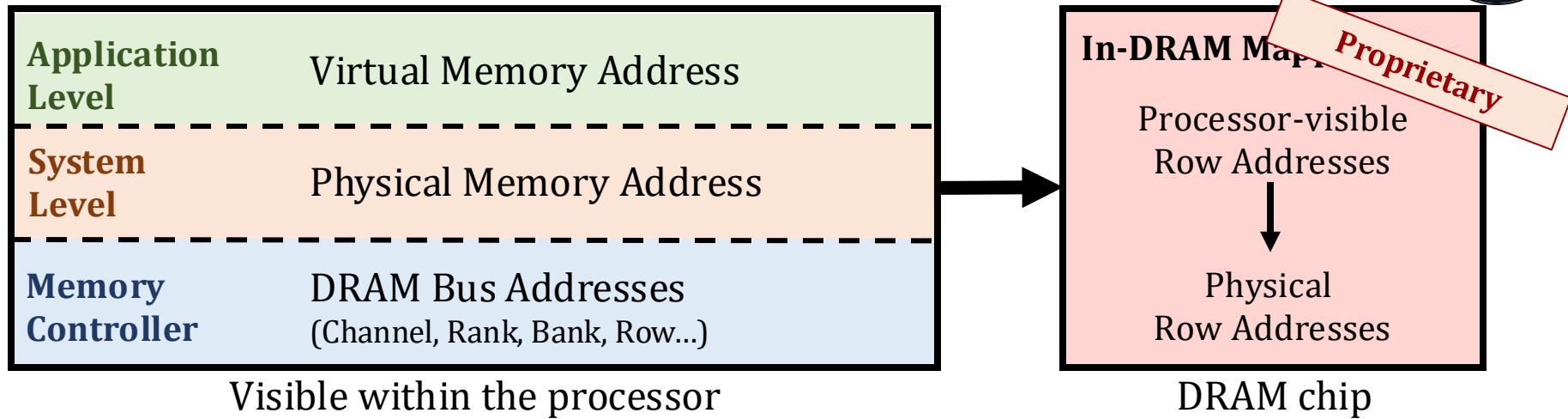
DRAM chip

- A **RowHammer attack** hammers **Row A**
- Processor-based defenses can **detect the attack**
- Refresh rows **A+1** and **A-1**
- Bitflips **still may occur** due to **unknown DRAM-internal row mapping**

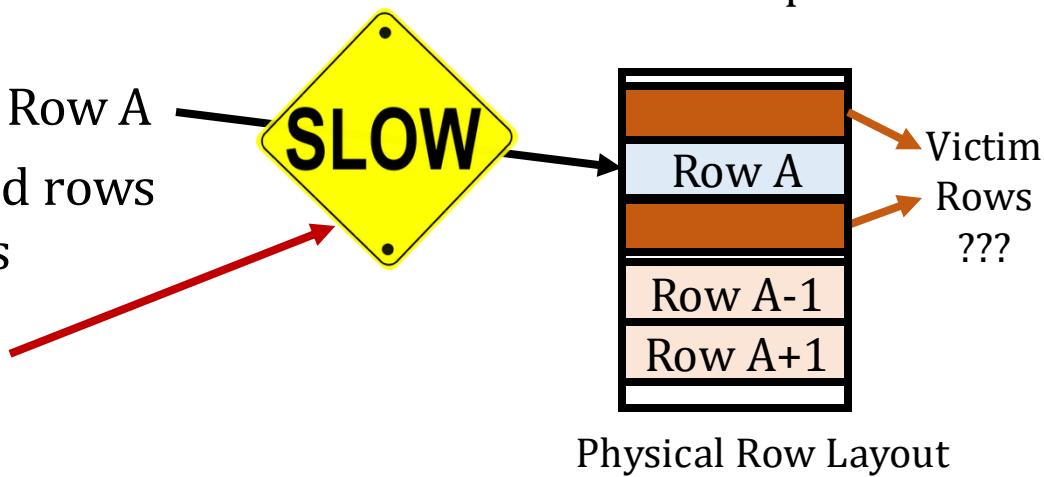


Many existing solutions rely on this **proprietary information**

BlockHammer



- A RowHammer attack hammers Row A
- BlockHammer detects hammered rows using area-efficient Bloom filters
- **Selectively throttles** accesses targeting hammered rows



BlockHammer **prevents** read disturbance bitflips **without the knowledge of or modifications to DRAM chip internals**



BlockHammer: Evaluation

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

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

- 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**



BlockHammer: Evaluation

- Cycle-level simulations using **Ramulator** and **DRAMPower**

System Configuration

No RowHammer attack:

Negligible (<0.6%) performance and energy overheads

Address mapping

DRAM

RowHammer Threshold

Mimimistic Open Pages

DDR4 1 channel, 1 rank, 4 bank group, 4 banks per bank group

32K

- Single-Core Benign Workloads:

RowHammer attack present:

Significant improvement on system performance (71%)
and energy consumption (32%)

- Randomly Chosen Multiprogrammed Workloads:

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

Publications in PhD Dissertation

- **Dissertation:** "[Enabling Efficient and Scalable DRAM Read Disturbance Mitigation via New Experimental Insights into Modern DRAM Chips](#)," ETH Zürich, 2024.
- A. Giray Yağlıkçı, Yahya Can Tuğrul, Geraldo F. Oliveira, İsmail Emir Yüksel, Ataberk Olgun, Haocong Luo, Onur Mutlu "[Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions](#)," in [HPCA, 2024](#).
- A. Giray Yağlıkçı, Ataberk Olgun, Minesh Patel, Haocong Luo, Hasan Hassan, Lois Orosa, Oguz Ergin, and Onur Mutlu, "[HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips](#)," in [MICRO, 2022](#).
- A. Giray Yağlıkçı, Haocong Luo, Geraldo F. de Oliveira, Ataberk Olgun, Minesh Patel, Jisung Park, Hasan Hassan, Jeremie S. Kim, Lois Orosa, and Onur Mutlu, "[Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices](#)," in [DSN, 2022](#).
- A. Giray Yağlıkçı, Lois Orosa, 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](#).
- A. Giray Yağlıkçı, 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](#)," in [HPCA, 2021](#).

More Details and Discussion on YouTube

SAFARI Live Seminars in Computer Architecture

A Deeper Look into RowHammer's Characteristics in Real Modern DRAM Chips



SPEAKER
Abdullah Giray Yağlıkçı
SAFARI Research Group, ETH Zurich

JAN 17, 2024 5:00PM CET



https://www.youtube.com/live/CRtm1es4n3o?si=8N5zB6e_RUc5Ejl8

SAFARI Live Seminars in Computer Architecture

Efficiently and Scalably Mitigating RowHammer in Modern and Future DRAM-Based Memory Systems



SPEAKER
Abdullah Giray Yağlıkçı
SAFARI Research Group, ETH Zurich

JAN 22, 2024 5:00PM CET



<https://www.youtube.com/live/YQwRYWpCsk0?si=jXPueMHb5wgs69-q>

RowHammer in ISCA 2025

Fifth Workshop on DRAM Security (DRAMSec)
June 21, 2025, co-located with ISCA 2025

The screenshot shows the homepage of the Fifth Workshop on DRAM Security (DRAMSec). It features a header with a photo of a man, the workshop name, and the date. Below the header, there are sections for 'Workshop chairs' (Onur Mutlu, Nisa Bostancı, Kuljit Bains), 'Submissions and Web Chairs' (Nisa Bostancı, A. Giray Yağlıkçı), and a 'Panel' discussion. A red YouTube play button is overlaid on the page. To the right, there's a 'Share' icon and a binary sequence '0 0 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1'. At the bottom, there's a link to 'Watch on YouTube'.

<https://dramsec.ethz.ch>

In participation of experts from



NVIDIA

Microsoft



SAMSUNG

Session 5A: RowHammer

Location: Okuma Auditorium (Main)

Session Chair: Gururaj Saileshwar

08:30 AM – 08:50 AM

MoPAC: Efficiently Mitigating Rowhammer with Probabilistic Activation Counting

Suhas Vittal, Salman Qazi, Poulami Das, Moin Qureshi

08:50 AM – 09:10 AM

When Mitigations Backfire: Timing Channel Attacks and Defense for PRAC-Based Rowhammer

Mitigations

Jeonghyun Woo, Joyce Qu, Gururaj Saileshwar, Prashant Nair

09:10 AM – 09:30 AM

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

Ismail Emir Yuksel, Akash Sood, Ataberk Olgun, O?uzhan Canpolat, Haocong Luo, Nisa Bostancı, Mohammad Sadrosadati, Giray Yaglikci, Onur Mutlu

09:30 AM – 09:50 AM

DREAM: Enabling Low-Overhead Rowhammer Mitigation via Directed Refresh Management

Hritvik Taneja, Moin Qureshi

Towards Robust and Sustainable Scaling

- Pre-PhD Research
 - Understanding and reducing **memory access latency**
 - Understanding and reducing **memory energy consumption**
- PhD Research
 - Understanding **read disturbance** failures in main memory
 - Mitigating **read disturbance** failures in main memory
- Post-PhD Research
 - Improving the **availability of data and resources** while mitigating read disturbance failures in main memory
- Future Research
 - Hardware-software **co-design**
 - **Data-centric** robustness
 - Better understanding of **hardware failures**
technology node scaling AND scale-out systems

Meanwhile: RowHammer in Industry Research

- JEDEC* formed a RowHammer task force
- Manufacturers' attitude changed: Denial → Solution
- An early secure solution emerged from UPMEM
- Then Microsoft, SK Hynix, Samsung, NVIDIA
- Google hacked Samsung's solution

2023

ISSCC 2023 / SESSION 28 / HIGH-DENSITY MEMORIES

28.8 A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Se

Woongrae Kim, Chulmoon J, Jeongjin Hwang, Jungmin Y, Mankeun Kang, Sangho Lee, Noguen Joo, Sangwoo Yoo, Sunil Hwang, Mi hyun Hwan, Yeonsu Jang, Kyoongchul J, Junghyun Kim, Soohwan Ki

Panopticon: A Complete In-DRAM Rowhammer Mitigation

2021

J. Bennett[§], Stefan Saroiu, Alec Wolman, and Lucian Cojocar
Microsoft, [§]Avant-Gray LLC

Security Analysis of the Silver Bullet Technique for RowHammer Prevention

21 January 2021

A. Giray Yağlıkçı, SAFARI Research Group, ETH Zürich

Jeremie S. Kim, SAFARI Research Group, ETH Zürich

Fabrice Devaux, UPMEM

Mutlu, SAFARI Research Group, ETH Zürich

DSAC: Low-Cost Rowhammer Mitigation Using

In-DRAM Trackers

2023

2024

Aamer Jaleel

Gururaj Saileshwar*

Stephen W. Keckler
NVIDIA
skeckler@nvidia.com

Moinuddin Qureshi
Georgia Tech
moin@gatech.edu

SoothSayer: Bypassing DSAC Mitigation by Predicting Counter Replacement

2024

Salman Qazi
Google

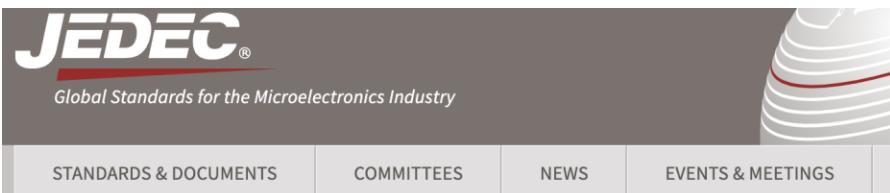
Daniel Moghimi
Google

FARI
research Group

up
mem

*JEDEC: Joint Electron Device Engineering Council

Meanwhile: RowHammer in Industry Research



The JEDEC logo is at the top left. Below it is the tagline "Global Standards for the Microelectronics Industry". To the right is a stylized graphic of a stack of memory chips. At the bottom is a horizontal navigation bar with four items: "STANDARDS & DOCUMENTS", "COMMITTEES", "NEWS", and "EVENTS & MEETINGS".

NEAR-TERM DRAM LEVEL ROWHAMMER MITIGATION

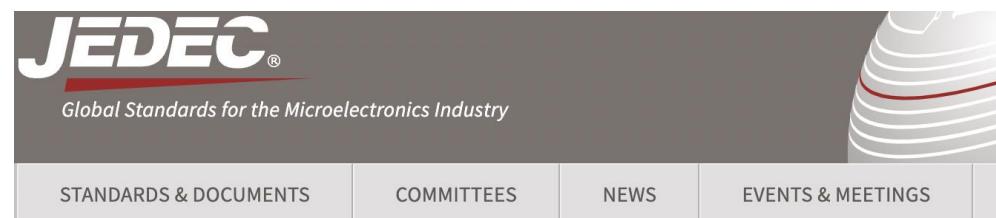
JEP300-1

Published: Mar 2021

RAM process node transistor scaling for power and DRAM capacity has made DRAM cells more sensitive to disturbances or transient faults. This sensitivity becomes much worse if external stresses are applied in a meticulously manipulated sequence, such as Rowhammer. Rowhammer related papers have been written outside of JEDEC, but some assumptions used in those papers didn't explain the problem very clearly or correctly, so the perception for this matter is not precisely understood within the industry. This publication defines the problem and recommends following mitigations to address such concerns across the DRAM industry or academia. Item 1866.01.

Committee(s): [JC-42](#)

<https://www.jedec.org/standards-documents/docs/jep300-1>



The JEDEC logo is at the top left. Below it is the tagline "Global Standards for the Microelectronics Industry". To the right is a stylized graphic of a stack of memory chips. At the bottom is a horizontal navigation bar with four items: "STANDARDS & DOCUMENTS", "COMMITTEES", "NEWS", and "EVENTS & MEETINGS".

SYSTEM LEVEL ROWHAMMER MITIGATION

JEP301-1

Published: Mar 2021

A DRAM rowhammer security exploit is a serious threat to cloud service providers, data centers, laptops, smart phones, self-driving cars and IoT devices. Hardware research and development will take time. DRAM components, DRAM DIMMs, System-on-chip (SoC), chipsets and system products have their own design cycle time and overall life time. This publication recommends best practices to mitigate the security risks from rowhammer attacks. Item 1866.02.

Committee(s): [JC-42](#)

<https://www.jedec.org/standards-documents/docs/jep301-1>

Meanwhile: RowHammer in Industry Research

The screenshot shows the JEDEC website with the following details:

- JEDEC® Global Standards for the Microelectronics Industry**
- STANDARDS & DOCUMENTS**, **COMMITTEES**, **NEWS**, **EVENTS & MEETINGS**, **JOIN**
- DDR5 SDRAM** (highlighted)
- JESD79-5C**
- Release Number: Version 1.30**
- Version 1.30**
- Description:** This standard defines the DDR5 SDRAM specification, including features, functionalities, AC and DC characteristics, packages, and ball/signal assignments. The purpose of this Standard is to define the minimum set of requirements for JEDEC compliant 8 Gb through 32 Gb for x4, x8, and x16 DDR5 SDRAM devices. This standard was created based on the DDR4 standards (JESD79-4) and some aspects of the DDR, DDR2, DDR3, and LPDDR4 standards (JESD79, JESD79-2, JESD79-3, and JESD209-4).
- Committee(s):** [JC-42](#), [JC-42.3](#)

A red oval and arrow highlight the release date **Apr 2024**.

The first promising update
a decade after the discovery of RowHammer
PRAC: Per Row Activation Counting in DRAM

Understanding Industry's Solution

Understanding the Security Benefits and Overheads of Emerging Industry Solutions to DRAM Read Disturbance

Oğuzhan Canpolat^{§†}

A. Giray Yağlıkçı[§]

Geraldo F. Oliveira[§]

Ataberk Olgun[§]

Oğuz Ergin[†]

Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB University of Economics and Technology

*presented in DRAMSec 2024
(co-located with ISCA)*

<https://github.com/CMU-SAFARI/ramulator2>

JEDEC
STANDARD

DDR5 SDRAM

JESD79-5C_v1.30

(Revision of JESD79-5B_v1.20, September 2022)

April 2024

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION



- PRAC is secure against RowHammer
- PRAC increases the latency of closing a DRAM row, causing **~10% performance degradation** for benign workloads
- PRAC can be exploited by memory performance attacks (up to **94% reduction** in DRAM chip's availability)

PRAC: Industry's Best Solution to RowHammer

- Per Row Activation Counting (PRAC) [JEDEC, 2024]

- Tracking RowHammer: DRAM chip

- Maintains an activation counter for each DRAM row
- **Updates the counter of a row while the row is closed**

The devil is
in this detail

- Mitigating RowHammer: DRAM chip

- Internally performs **preventive refreshes**
- Needs time to perform preventive refreshes
- Sends a **back-off signal** to the memory controller to ask for time
- **Memory controller provides DRAM chip with time** for RowHammer-preventive refreshes



Per Row
Activation Counters

Chronus: Making Industry's Best Solution Better

Problem:

PRAC incurs **10% slowdown** (common case)

Key Insight:

Closing a row has the **added latency** of updating the row's counter because the counter is stored within the row

Counters	DRAM Rows
0	10101010101010101010101010
0	10101010101010101010101010
:	:

PRAC **increases** the latency of closing a DRAM row by **140%**

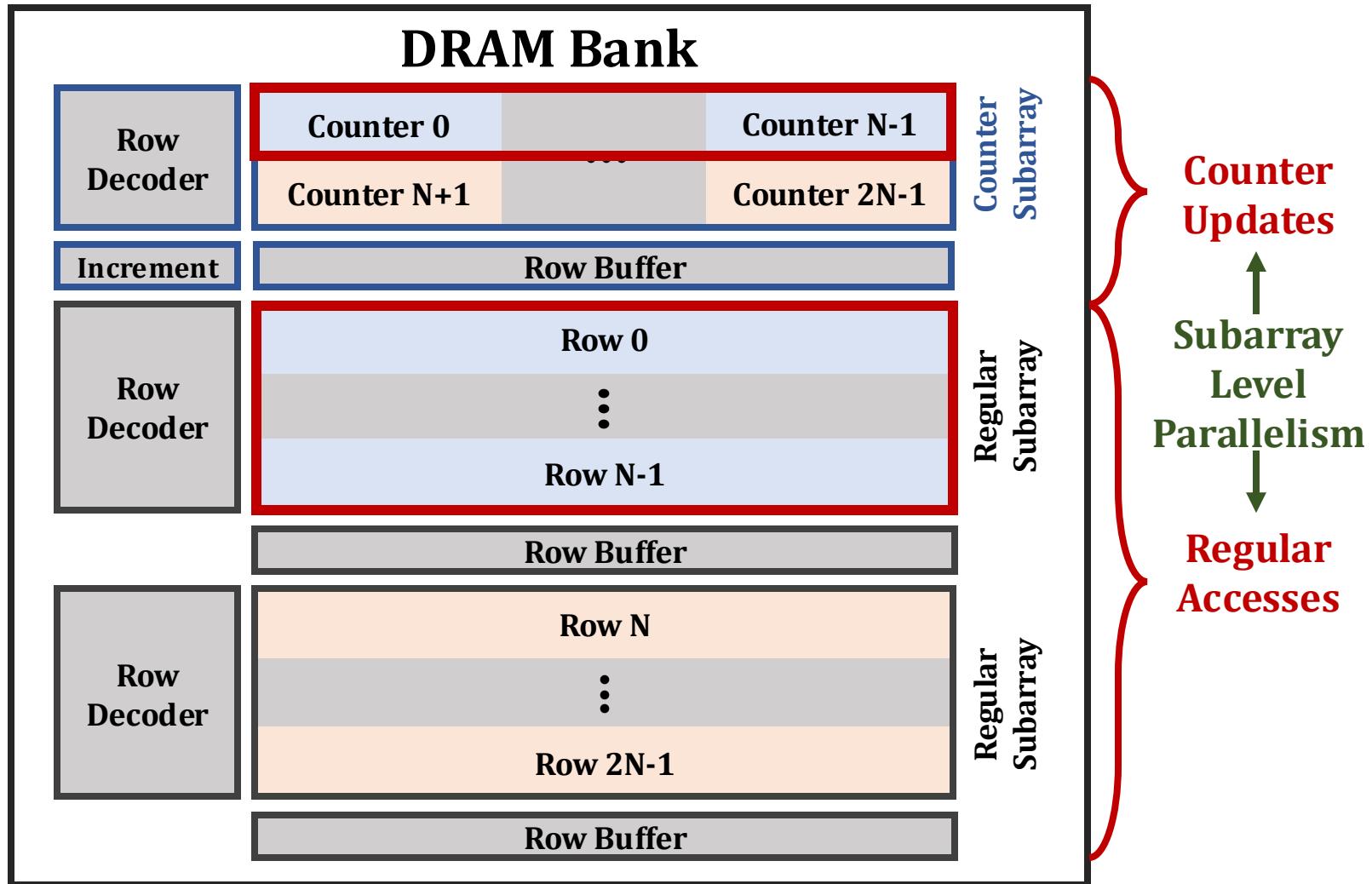
Key Idea:

Update row activation counters in parallel to column accesses by leveraging subarray-level parallelism

Key Result: Chronus significantly reduces PRAC's performance overhead to 0.04%

Chronus's Key Idea: Concurrent Counter Update

Chronus concurrently updates counters while serving accesses



We leverage **inherently-available subarray-level parallelism**
to alleviate PRAC's performance overhead

Making Industry's Best Solution Better



Chronus: Understanding and Securing the Cutting-Edge Industry Solutions to DRAM Read Disturbance

Oğuzhan Canpolat^{1,2} A. Giray Yağlıkçı¹ Geraldo F. Oliveira¹ Ataberk Olgun¹

Nisa Bostancı¹ İsmail Emir Yüksel¹ Haocong Luo¹

Oğuz Ergin^{2,3} Onur Mutlu¹

¹ETH Zürich ²TOBB University of Economics and Technology ³University of Sharjah

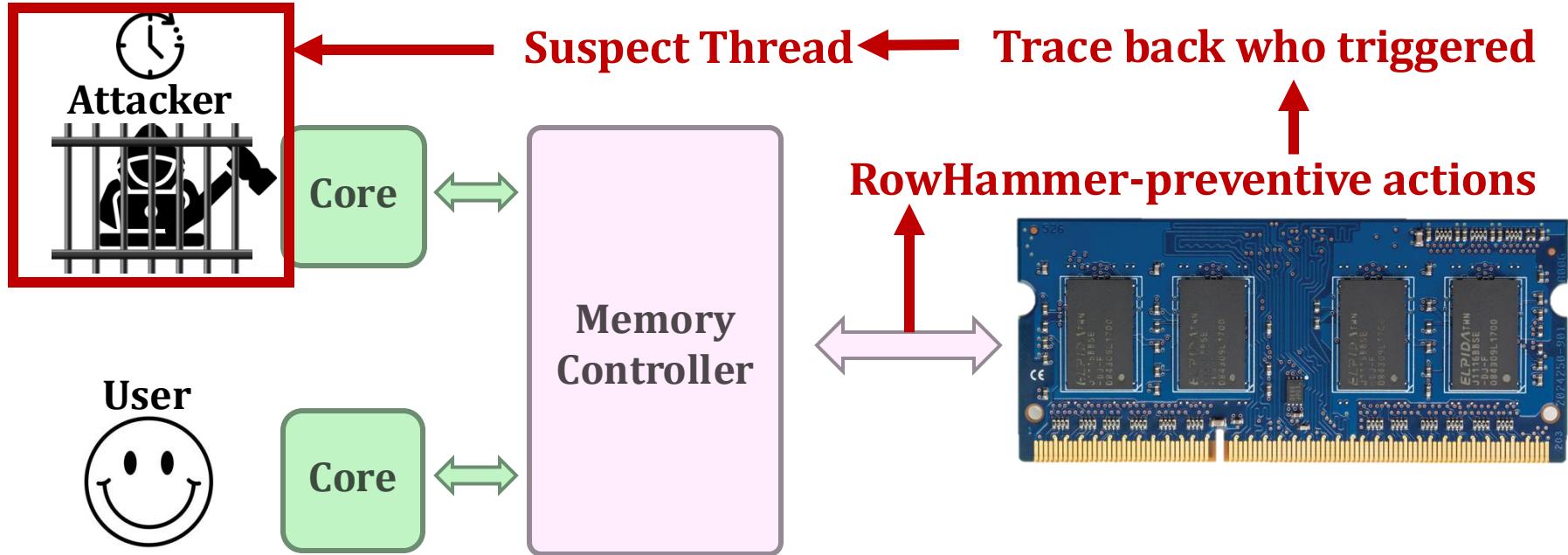
published in HPCA 2025

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

The screenshot shows the GitHub repository page for 'Chronus' under the 'CMU-SAFARI' organization. The repository is public. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there are buttons for Edit Pins, Watch (3), Fork (0), and Star (1). The main content area shows the 'Code' tab selected, with a master branch and 1 branch. There are three commits listed: one by 'kirbyydoge' adding camera ready plotting scripts, another by 'ae_results' adding camera ready plotting scripts, and one by 'mixes' adding plotting scripts and updating a CPU trace zenodo link. The right sidebar contains an 'About' section with a note: 'No description, website, or topics provided.' It also lists Readme, Activity, and Custom properties.

Author	Commit Message	Date
kirbyydoge	Add and update camera ready plotting scripts	ffdb2fb · last month
ae_results	Add and update camera ready plotting scripts	last month
mixes	Add plotting scripts and update CPU trace zenodo link	2 months ago

BreakHammer: Throttling Suspect Threads in Hardware



Detect and **slow down** the memory accesses of threads
that trigger **many** RowHammer-preventive actions

Key Results:

- **Under attack:**
 - Significantly **improves** system performance (by 90% on average)
 - Significantly **reduces** energy consumption (by 55% on average)
- **No attack:**
 - Slightly (<1%) **improves** performance and energy consumption



BreakHammer: Enhancing RowHammer Mitigations by Carefully Throttling Suspect Threads

Oğuzhan Canpolat^{\$†}

Yahya Can Tuğrul^{\$†}

^{\$}ETH Zürich

A. Giray Yağlıkçı[§]

Konstantinos Kanellopoulos[†]

[†]TOBB University of Economics and Technology

Ataberk Olgun[§]

Oğuz Ergin^{‡\$†}

[‡]University of Sharjah

Ismail Emir Yuksel[§]

Onur Mutlu[§]

RowHammer is a major read disturbance mechanism in DRAM where repeatedly accessing (hammering) a row of DRAM cells (DRAM row) induces bitflips in other physically nearby DRAM rows. RowHammer solutions perform preventive actions (e.g.,

can experience bitflips when a nearby DRAM row (i.e., aggressor row) is repeatedly opened (i.e., hammered) [2–70].

Many prior works demonstrate attacks on a wide range of systems where they exploit read disturbance to escalate

published in MICRO 2024

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

The screenshot shows the GitHub repository page for 'BreakHammer'. 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 a search bar with placeholder text 'Type / to search'. To the right of the search bar are icons for fork, star, and profile. The main header of the page says 'BreakHammer' and 'Public'. On the left, there are buttons for master branch, 2 branches, and Tags. A 'Code' dropdown menu is open. On the right, there's an 'About' section with the message 'No description, website, or topics provided.' Below the 'About' section are links for Readme, Activity, and Custom notifications.

Code Issues Pull requests Actions Projects Security Insights Settings

SAFARI BreakHammer Public

Edit Pins Watch 3 Fork 0 Star 4

master 2 Branches Tags

Go to file + <> Code About

kirbyydoge Update README.md 2ea4b97 · last month 32 Commits

ae_results Update existing csvs and plots with full artifact evaluat... 2 months ago

No description, website, or topics provided.

Readme Activity

Towards Robust and Sustainable Scaling

- Pre-PhD Research
 - Understanding and reducing **memory access latency**
 - Understanding and reducing **memory energy consumption**
- PhD Research
 - Understanding **read disturbance** failures in main memory
 - Mitigating **read disturbance** failures in main memory
- Post-PhD Research
 - Improving the **availability of data and resources** while mitigating read disturbance failures in main memory
- Future Research
 - **Hardware-software co-design**
 - **Data-centric** robustness
 - Better understanding of **hardware failures**
technology node scaling AND scale-out systems

Hardware-Software Co-Design for Sustainable Robustness

Insights into data structures

Software semantics

User / Process / Thread / Address Space

Software



Hardware

Accurate measurements
Prompt responses

Many opportunities for ensuring security at low cost

Hardware-Assisted Malware Detection

An Example HW-SW Co-Design Direction

Hardware-level sensors and counters can **quickly** and **accurately** collect data on application behavior and react in nanoseconds

- RowHammer attacks (e.g., BlockHammer [Yaglikci+, HPCA'21])
- Performance attacks (e.g., BreakHammer [Canpolat+, MICRO'24])
- Timing side channel attacks (e.g., Cyclone [Harris, MICRO'19])
- ...

Software-level malware detection has a **better big picture**

- Program(s) that spawn those malicious threads
- User(s) that run those programs
- More constructive reactions
(e.g., flag functions and inform programmers)

Key Insight: System-level attacks often try the attack many times

Key Idea: Provide software-level malware detection with hardware-level observations and statistics

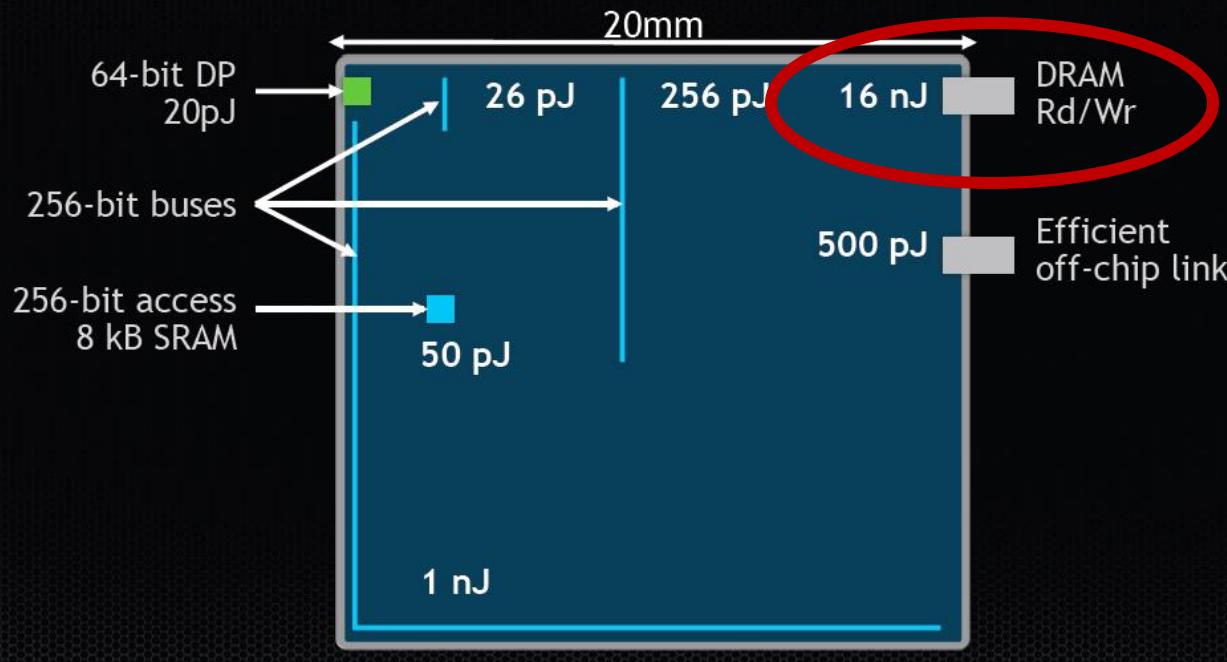
Towards Robust and Sustainable Scaling

- Pre-PhD Research
 - Understanding and reducing **memory access latency**
 - Understanding and reducing **memory energy consumption**
- PhD Research
 - Understanding **read disturbance** failures in main memory
 - Mitigating **read disturbance** failures in main memory
- Post-PhD Research
 - Improving the **availability of data and resources** while mitigating read disturbance failures in main memory
- Future Research
 - Hardware-software **co-design**
 - **Data-centric** robustness
 - Better understanding of **hardware failures**
technology node scaling AND scale-out systems

Data-Centric Security and Privacy (1/5)

Motivation for Sustainable Scaling

Communication Dominates Arithmetic



William Dally,
HiPEAC 2015

A memory access (off-chip data movement)
consumes $\sim 100x\text{-}1000x$ the energy
of an arithmetic operation

Data-Centric Security and Privacy (2/5)

High-Level Motivation for Data-Centric S&P

Data-centric computation
for better **performance** (-per-Joule and -per-area)

Processing in Memory and Storage
are two promising computation paradigms

More to do for Security in Memory and Storage

Data-Centric Security and Privacy

Problem Definition

- Common security practices are processor-centric
 - e.g., Intel SGX and ARM TrustZone
 - Moving encrypted data across chips consumes time and energy, increases bus usage, and reduces processor's and memory's availability
 - Supporting accelerators near memory and storage becomes complicated
- Processing in memory and storage can perform
 - Encryption
 - Isolation
 - Access control
 - Error correction
 - Timing obfuscation
 - ...

Processing in memory and storage have a lot to offer

Data-Centric Security and Privacy

- Specialized circuitry near data
 - Hardware complexity
 - Added latency

IceClave: A Trusted Execution Environment for In-Storage Computing

Luyi Kang^{*}
University of Maryland, College Park

Xiaohao Wang[‡]
UIUC

Myeong Joon Kang
SK Hynix

MICRO'21

Yuqi Xue^{*}
UIUC

Weiwei Jia^{*}
UIUC

Jongryool Kim
SK Hynix

Changhwan Youn
SK Hynix

Hyung Jin Lim
SK Hynix

Bruce Jacob
University of Maryland, College Park

Jian Huang
UIUC

HPCA'25

Chronus: Understanding and Securing the Cutting-Edge Industry Solutions to DRAM Read Disturbance

Oğuzhan Canpolat^{1,2} A. Giray Yağlıkçı¹ Geraldo F. Oliveira¹ Ataberk Olgun¹

Nisa Bostancı¹ İsmail Emir Yüksel¹ Haocong Luo¹

Oğuz Ergin^{2,3} Onur Mutlu¹

¹ETH Zürich ²TOBB University of Economics and Technology ³University of Sharjah



• Computation using homomorphic encryption

- High capacity overhead → Against density scaling
- Increased execution time

CIPHERMATCH: Accelerating Homomorphic Encryption-Based String Matching via Memory-Efficient Data Packing and In-Flash Processing

Mayank Kabra[†] Rakesh Nadig[†] Harshita Gupta[†] Rahul Bera[†] Manos Frouzakis[†]
Vamanan Arulchelvan[†] Yu Liang[†] Haiyu Mao[‡] Mohammad Sadrosadati[†] Onur Mutlu[†]

ETH Zurich[†] King's College London[‡]

ASPLOS'25

State-of-the-art is far from a definitive solution

Security of Data-Centric Computation

Timing side- and covert-channel attacks
are easier near/in memory

- Less noise
- No (or simpler) caches
- Finer-grained control



Larger channel bandwidth
No low-cost and effective solution yet

DSN 2025



Code Available



Code Reviewed



Code Reproducible

Revisiting Main Memory-Based Covert and Side Channel Attacks in the Context of Processing-in-Memory

F. Nisa Bostancı^{†*} Konstantinos Kanellopoulos^{†*} Ataberk Olgun[†]
A. Giray Yağlıkçı[†] İsmail Emir Yüksel[†] Nika Mansouri Ghiasi[†]
Zülal Bingöl^{†‡} Mohammad Sadrosadati[†] Onur Mutlu[†]

[†]ETH Zürich [‡]Bilkent University

Data-Centric Security and Privacy

Workloads of Special Interest

Critical health applications

- Agent-based simulations
- Genomics

Common characteristics

- Excessively large amount of data
- Computation in **distributed and shared** environments
- Data integrity is important for correctness
- Confidentiality and privacy are first-order concerns
- Performance challenges are barely solved so far

Bad news: We are far from ideal

Good news: Lots of research to conduct

Towards Robust and Sustainable Scaling

- Pre-PhD Research
 - Understanding and reducing **memory access latency**
 - Understanding and reducing **memory energy consumption**
- PhD Research
 - Understanding **read disturbance** failures in main memory
 - Mitigating **read disturbance** failures in main memory
- Post-PhD Research
 - Improving the **availability of data and resources** while mitigating read disturbance failures in main memory
- Future Research
 - Hardware-software **co-design**
 - **Data-centric** robustness
 - Better understanding of **hardware failures**
technology node scaling AND scale-out systems

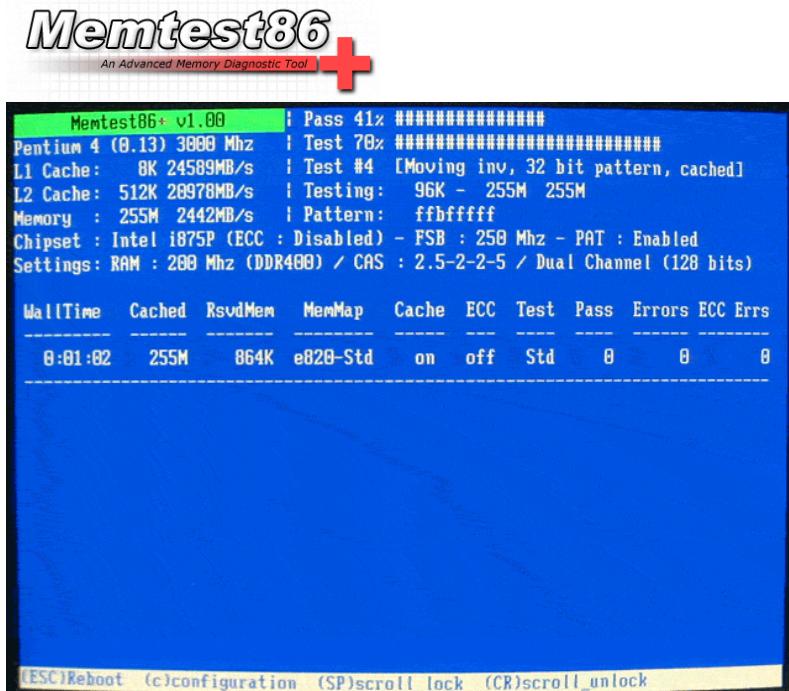
Better Understanding of Failure Mechanisms

A generic question: what is the worst-case for a given system?

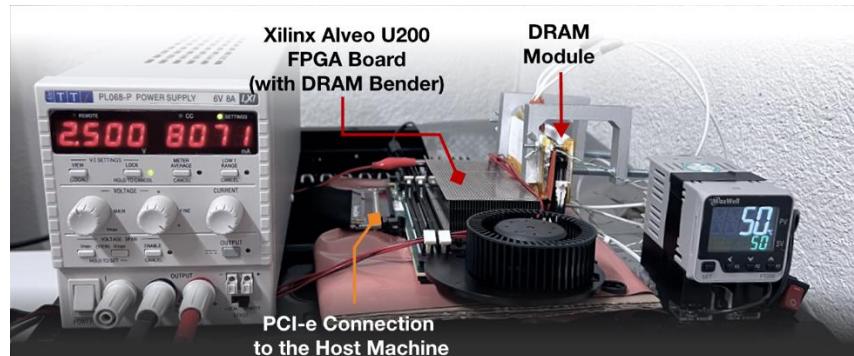
- Failure mechanisms depend on **many conditions** and can be observed somewhat **probabilistically**
- Post-manufacturing testing cannot cover **all conditions** practically
- Better **methodologies** and **infrastructures** are needed
- Testing is **costly**
 - the chip cannot be used while testing
 - you need to buy chips to test
- We need to revolutionize testing and data collection with
 - **Online testing methodologies:** test a chip in runtime when it is idle
 - Automated **reverse-engineering** tools
 - **Crowdsourcing** data collection for scaling up sample sets

Some Good Examples of Testing Infrastructures

Examples of good *offline* testing tools



DRAM Bender



Revizor

We need plug-and-play *online (runtime)* testing tools

Systems should scale
robustly and sustainably
as chips get denser
and many users share systems

Secure and Sustainable System Scaling

Abdullah Giray Yağlıkçı

UC Berkeley
August 5, 2025

Secure and Sustainable System Scaling

Backup Slides

Abdullah Giray Yağlıkçı

UC Berkeley
August 5, 2025

Brief Self Introduction

- Researcher
 - SAFARI Research Group
 - ETH Zurich (Feb 2018 – ongoing)
 - Intel Labs (Aug 2017 – Feb 2018)
 - Carnegie Mellon University (Aug 2016 – Aug 2017)
 - Earlier:
 - Univ. of Notre Dame, IN, USA (Aug 2014-Aug 2016)
 - TOBB Univ., Ankara, Turkey (Aug 2011-Aug 2014)
- Research interests:
 - Computer architecture
 - Memory and storage systems
 - Hardware security, safety, reliability, performance, availability, fairness, energy efficiency
 - Hardware/software cooperation
 - Memory-centric security
 - Testing and verification
 - ...



Abdullah Giray Yaglikci

Postdoc. Researcher
and Lecturer

ETH Zurich

<https://agyaglikci.github.io>

agirayyaglikci@gmail.com

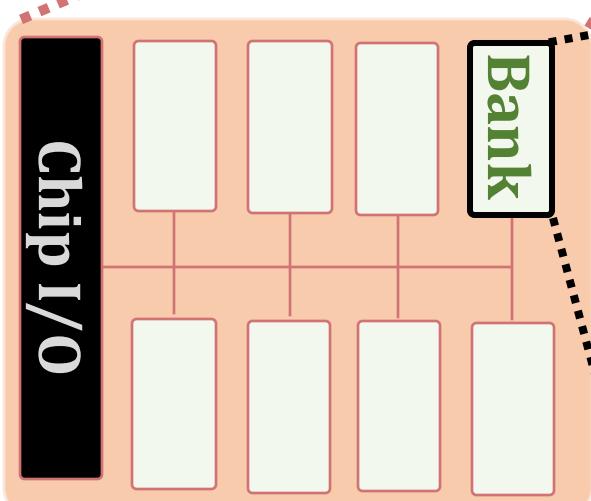
PhD: ETH Zurich, 2024

Memory Organization

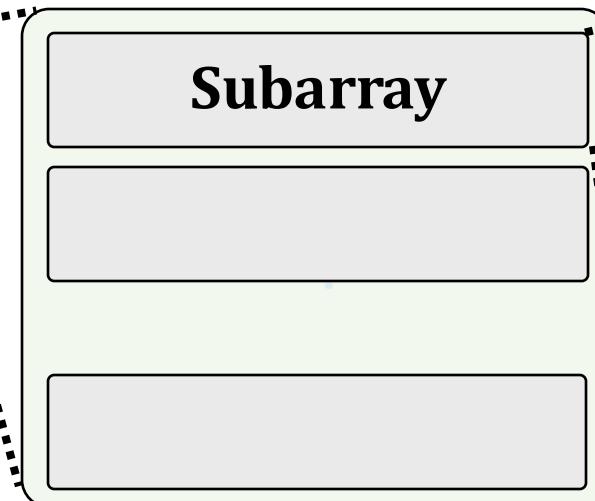
Prevalent memory technology: Dynamic Random Access Memory (DRAM)



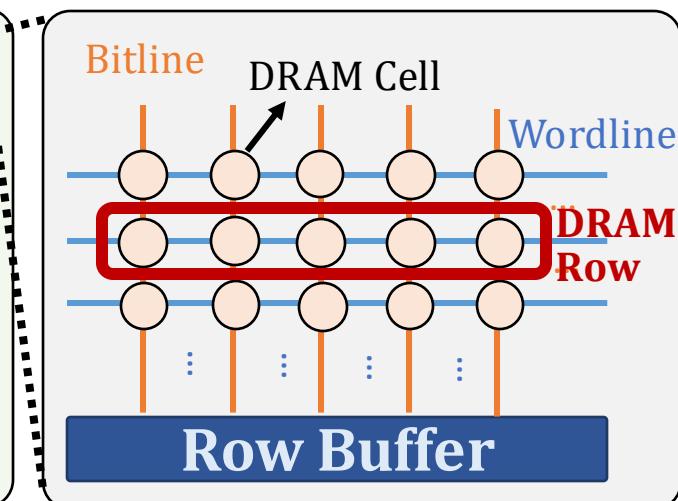
DRAM
Module



DRAM Chip



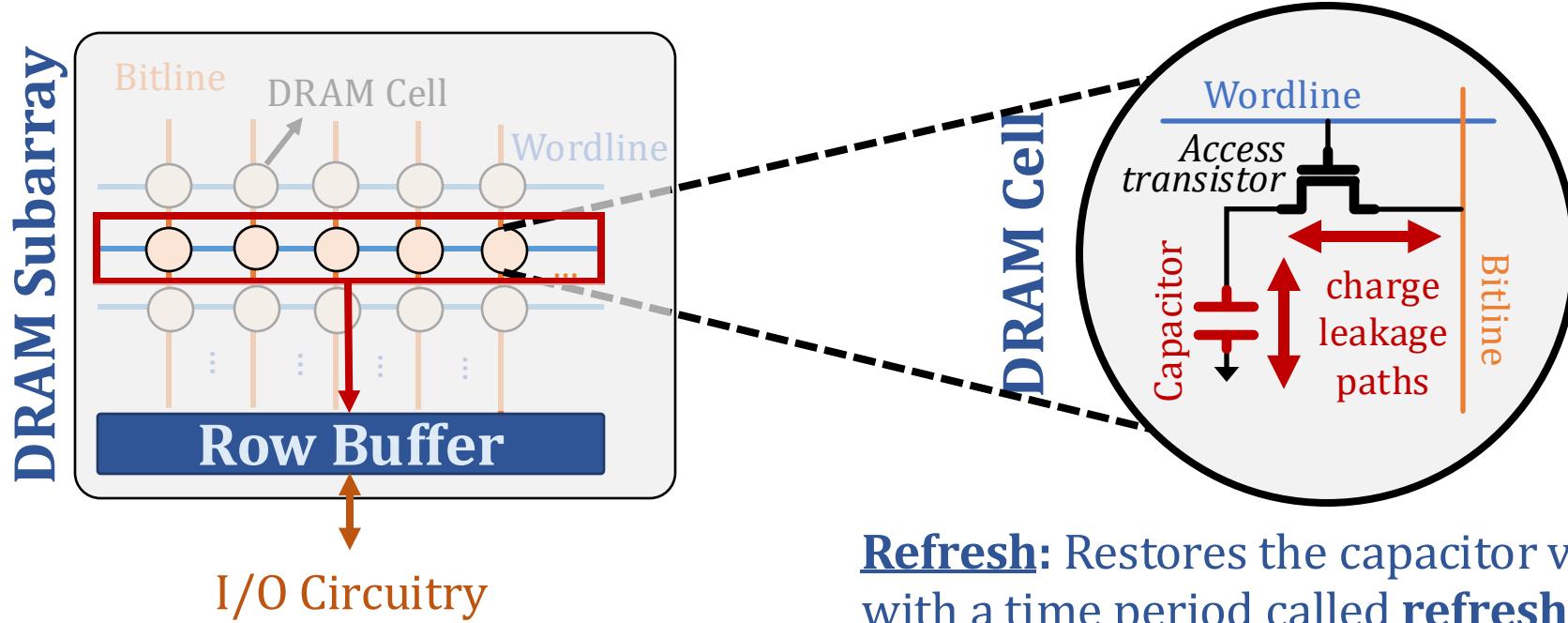
DRAM Bank



DRAM Subarray

DRAM Operation

Prevalent memory technology: Dynamic Random Access Memory (DRAM)



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

RowHammer under Reduced Refresh Latency

- Experimental data out of **388 DDR4 DRAM** chips from 3 major vendors
- RowHammer-preventive **refresh latency can be significantly reduced** without jeopardizing the **DRAM chip's data integrity**

PaCRAM: Partial Charge Restoration for Aggressive Mitigation

- **Reduces the latency** of RowHammer-preventive refreshes
- Performs preventive refreshes **slightly more frequently**
- **Improves system performance** and **energy efficiency**
- **Near-zero hardware complexity**



Understanding RowHammer Under Reduced Refresh Latency: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Yahya Can Tuğrul^{§†} A. Giray Yağlıkçı[§] İsmail Emir Yüksel[§] Ataberk Olgun[§]
Oğuzhan Canpolat^{§†} Nisa Bostancı[§] Mohammad Sadrosadati[§] Oğuz Ergin^{‡†} Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB University of Economics and Technology

[‡]University of Sharjah

published in HPCA 2025

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

Major Contributions

To efficiently and scalably mitigate DRAM read disturbance, we

1 build a **detailed understanding** of DRAM read disturbance



2 leverage insights into modern DRAM chips and memory controllers



3 devise a **novel solution** that do not require proprietary knowledge of DRAM chip internals



Other Works (Not Covered due to Time Limitation)

DSN 2022

Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices

A. Giray Yağlıkçı¹ Haocong Luo¹ Geraldo F. de Oliviera¹ Ataberk Olgun¹ Minesh Patel¹
Jisung Park¹ Hasan Hassan¹ Jeremie S. Kim¹ Lois Orosa^{1,2} Onur Mutlu¹

¹ETH Zürich

²Galicia Supercomputing Center (CESGA)

- First work to investigate **wordline voltage's effect** on RowHammer vulnerability
- Reducing wordline voltage **reduces RowHammer vulnerability**
- Worsens **data retention time** and **row activation latency**

HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı¹ Ataberk Olgun¹ Minesh Patel¹ Haocong Luo¹ Hasan Hassan¹
Lois Orosa^{1,3} Oğuz Ergin² Onur Mutlu¹

¹ETH Zürich

²TOBB University of Economics and Technology

³Galicia Supercomputing Center (CESGA)

- Leverages **inherently-available** but **not exposed** subarray-level parallelism in DRAM
- Reduces **time spent for refresh by half** and improves system performance

MICRO 2022

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

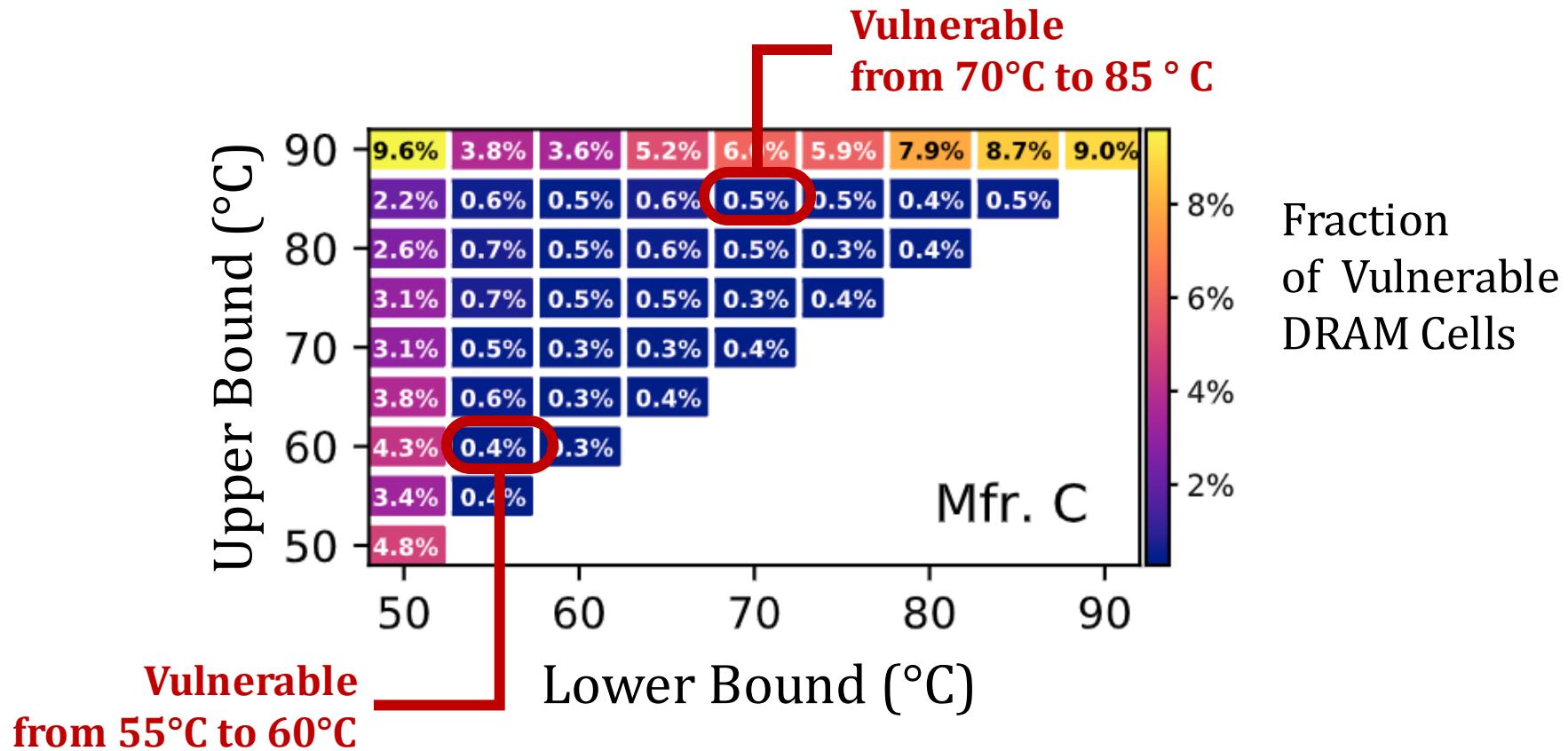
SAFARI

ETH zürich



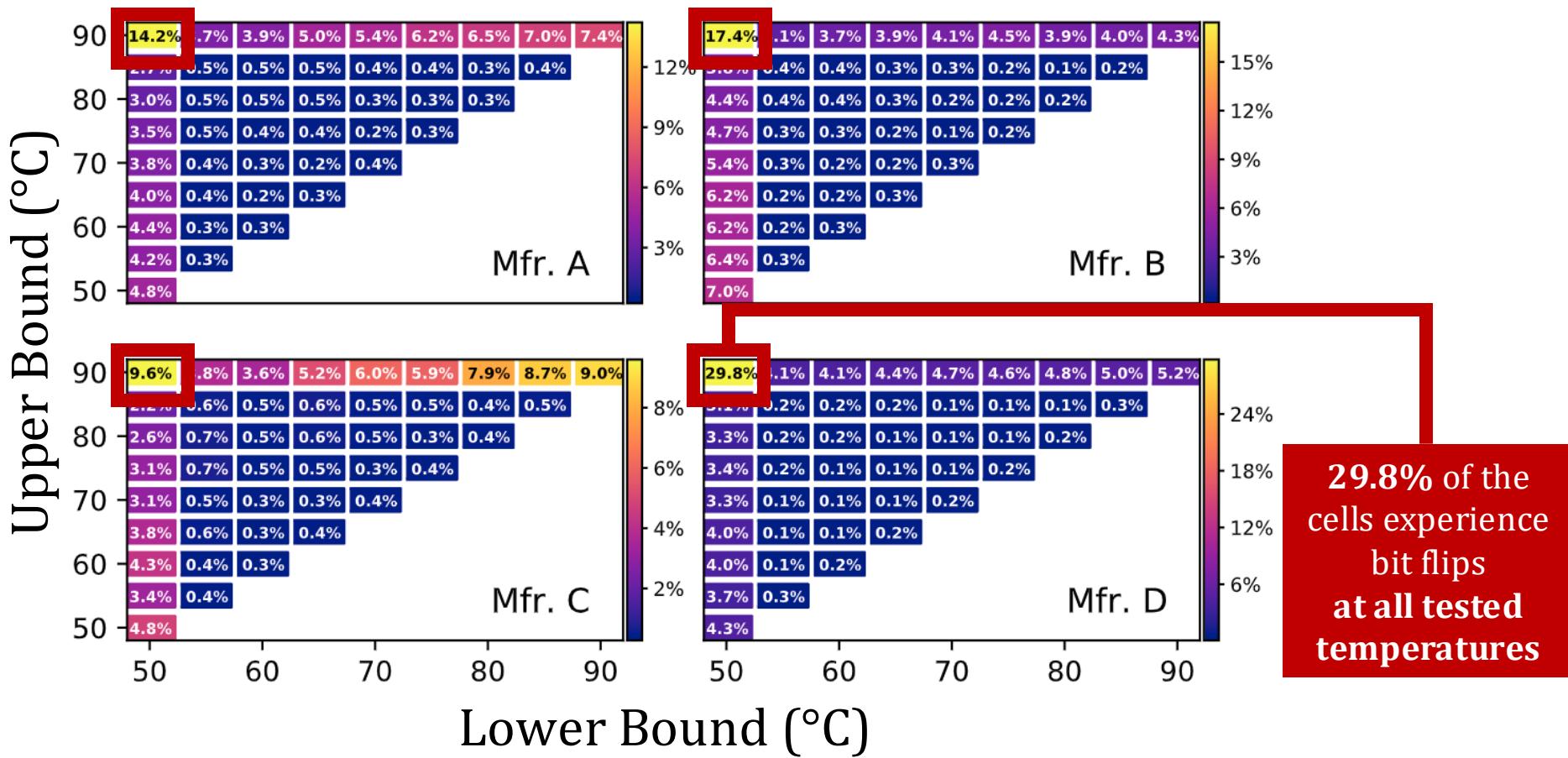
TOBB ETÜ
University of Economics & Technology

Impact of Temperature on DRAM Cells



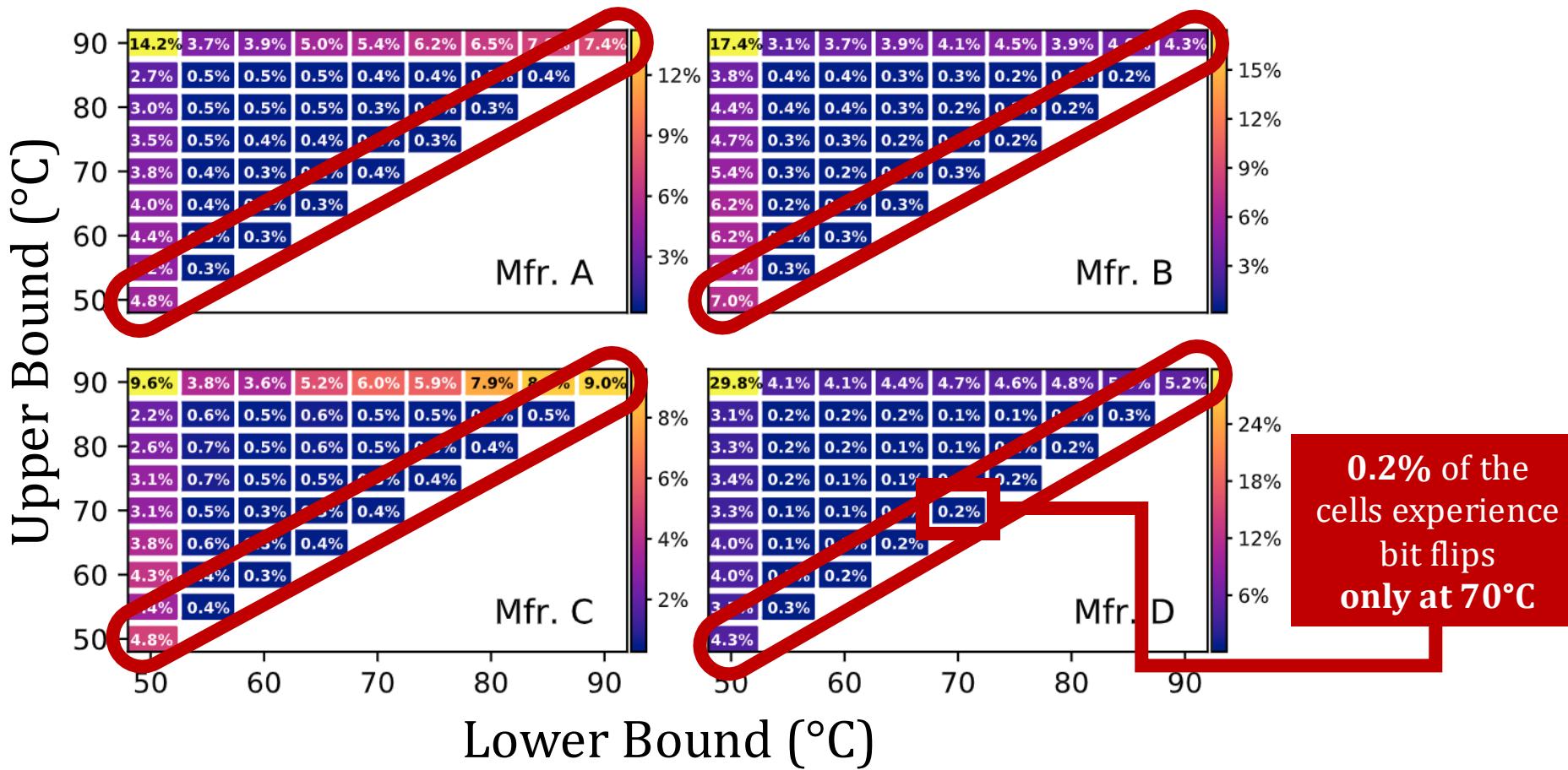
Different DRAM cells are vulnerable to RowHammer
within specific temperature ranges

Impact of Temperature on DRAM Cells



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

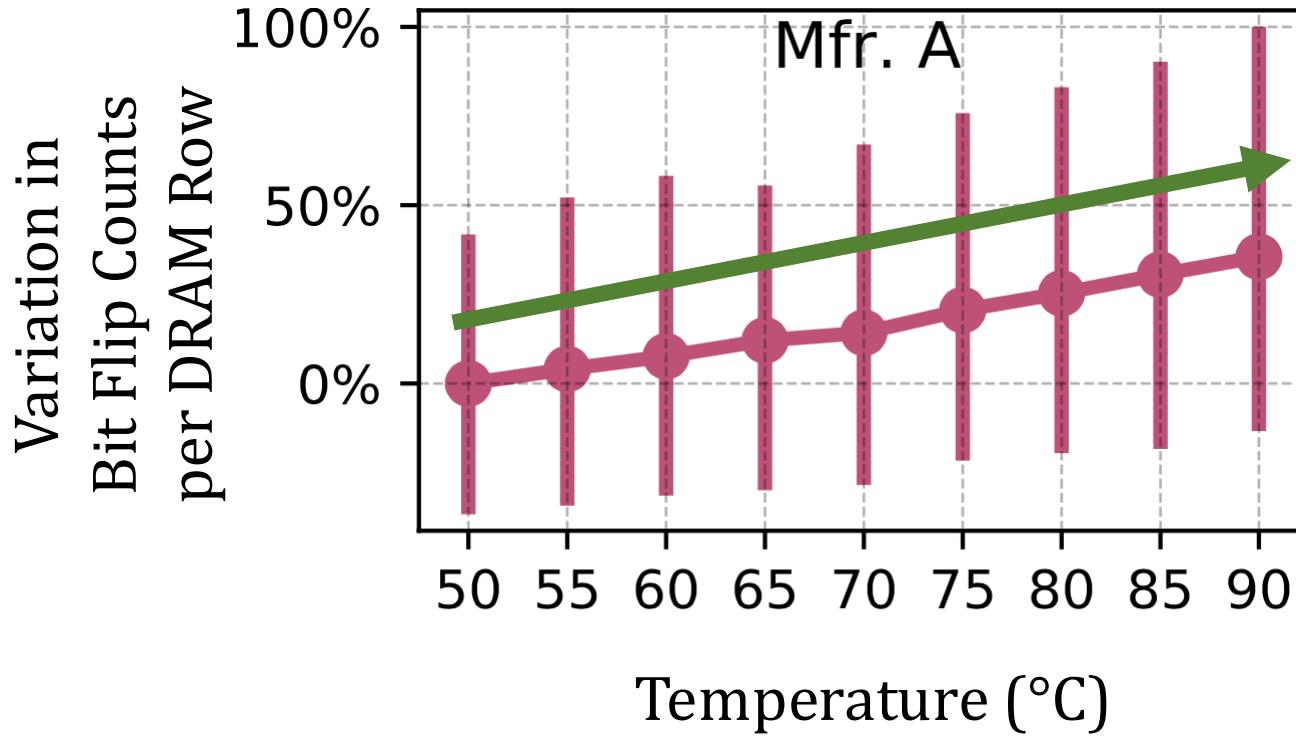
Impact of Temperature on DRAM Cells



OBSERVATION 3

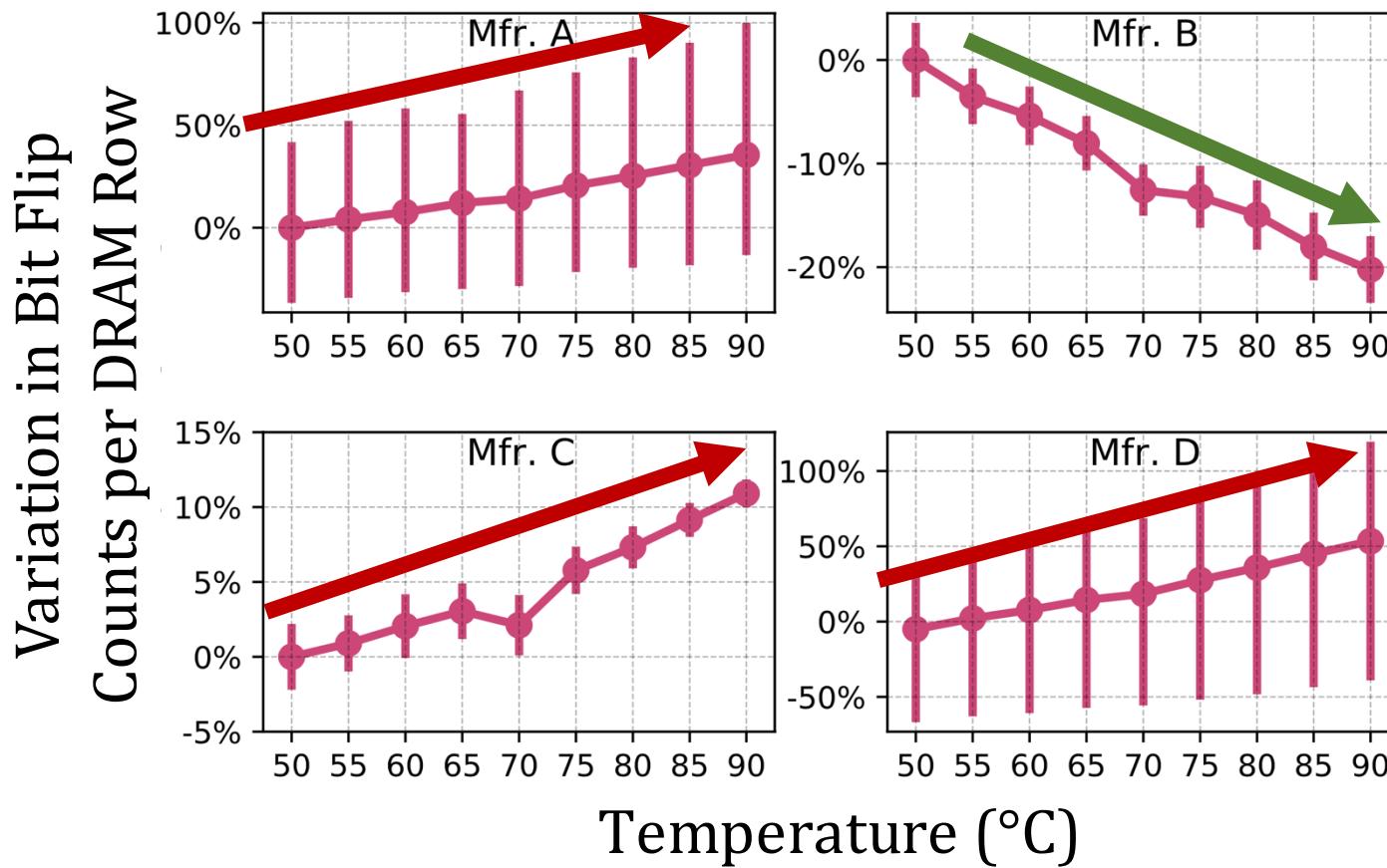
A **small fraction** of all vulnerable DRAM cells are vulnerable to RowHammer **only in a very narrow temperature range**

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

Memory Access Patterns in Aggressor Row Active Time Analysis

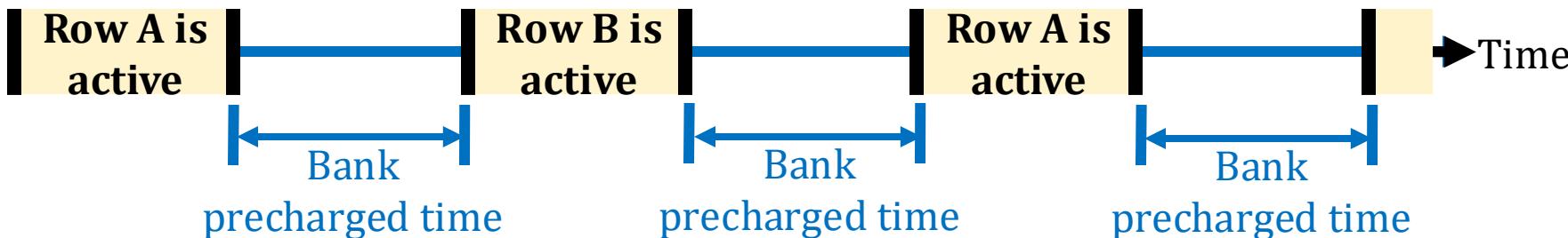
- Baseline access pattern:



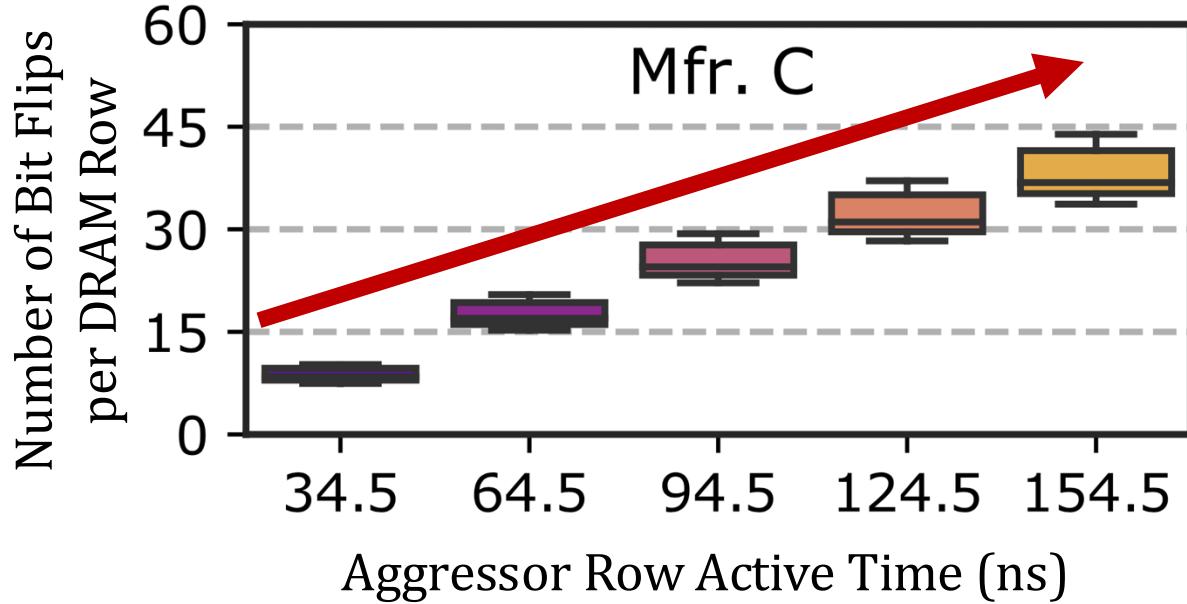
- Increasing **aggressor row active time**:



- Increasing **bank precharged time**:

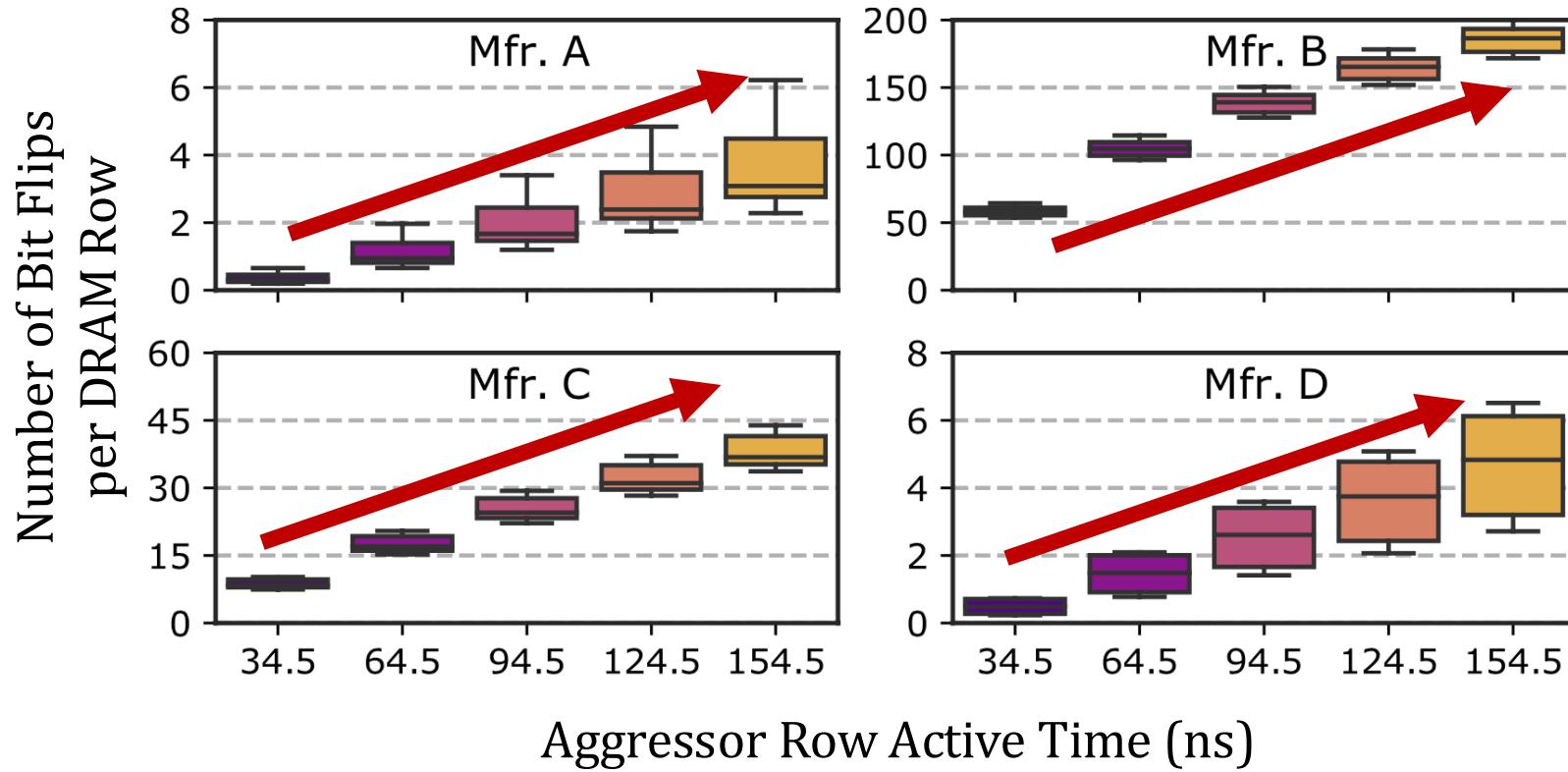


Increasing Aggressor Row Active Time



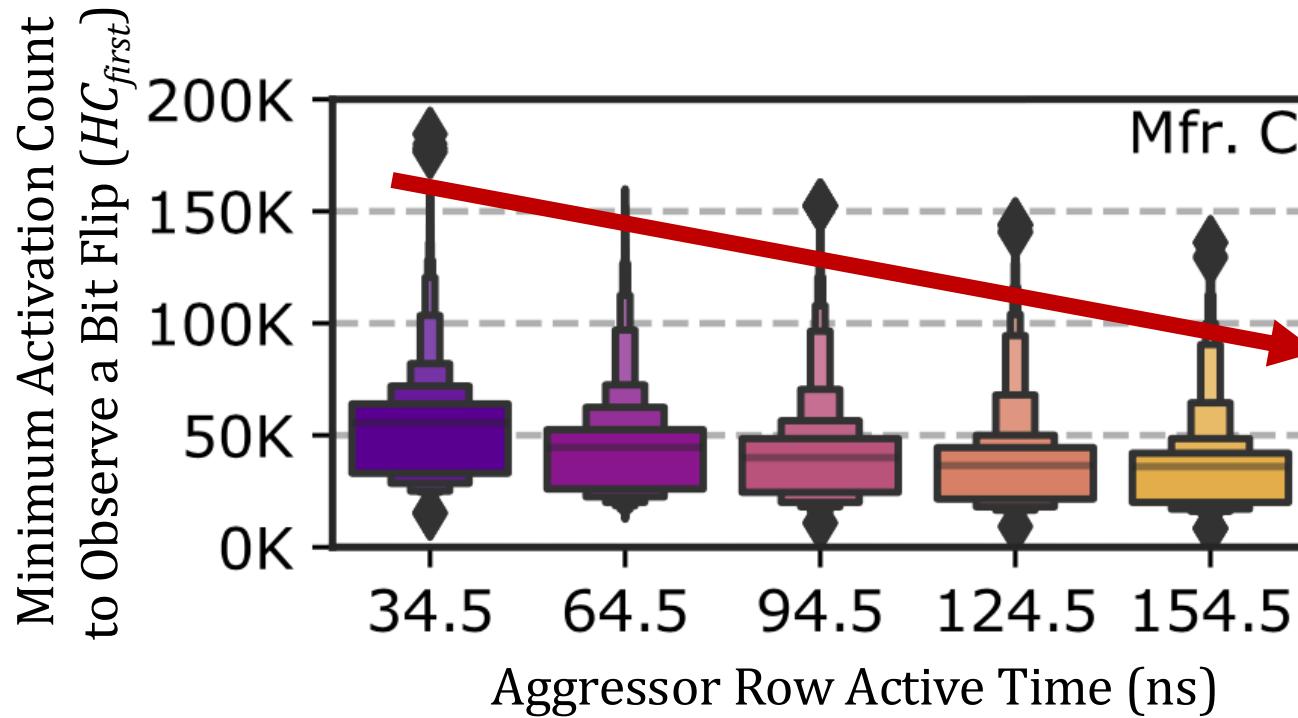
As the **aggressor row stays active longer**,
more DRAM cells experience RowHammer bit flips

Increasing Aggressor Row Active Time



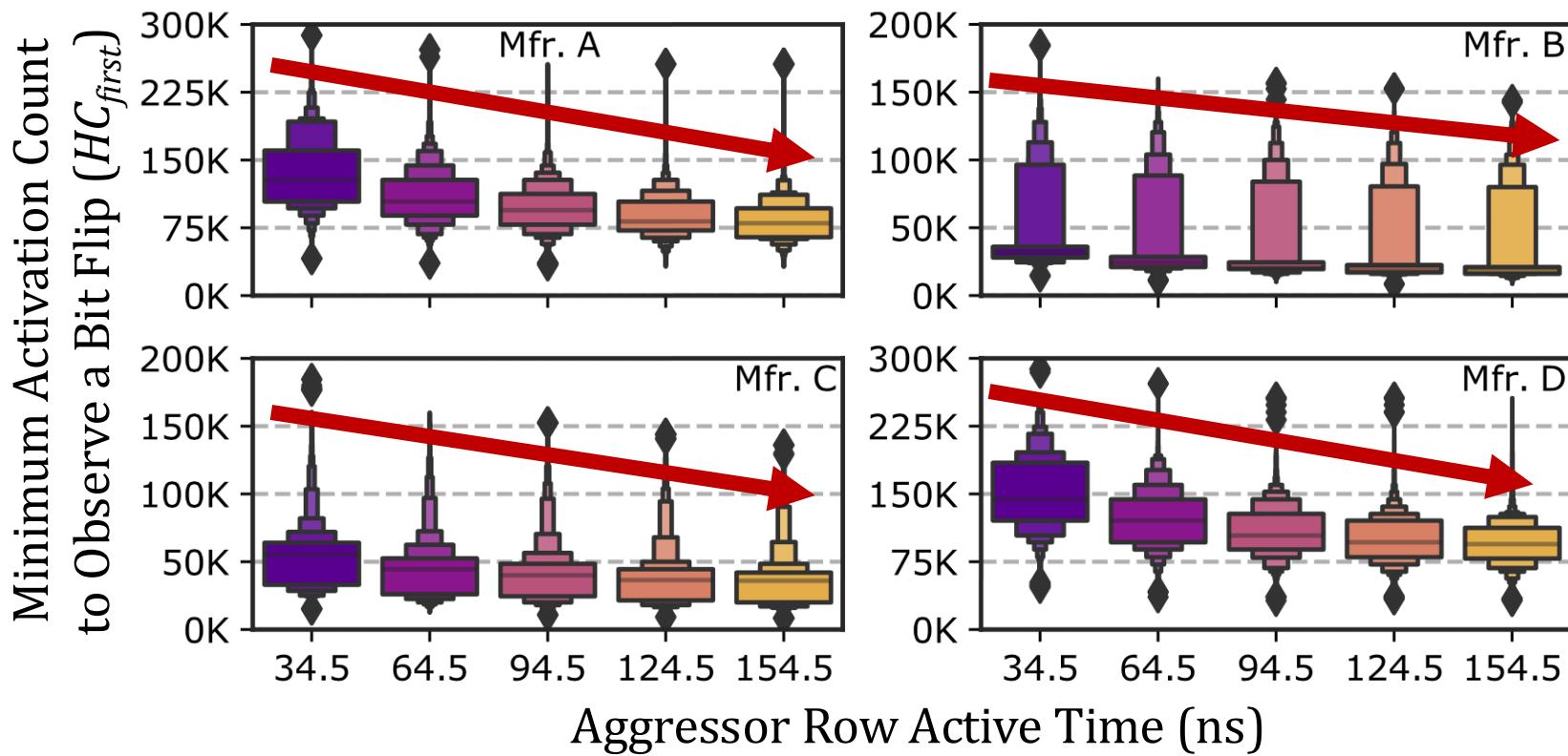
As the **aggressor row stays active longer**,
more DRAM cells experience RowHammer bit flips

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



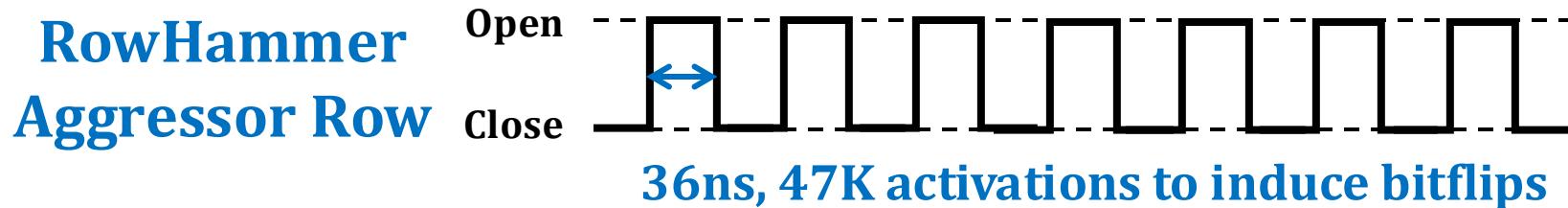
OBSERVATION 8

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

RowPress vs. RowHammer

Instead of using a high activation count,

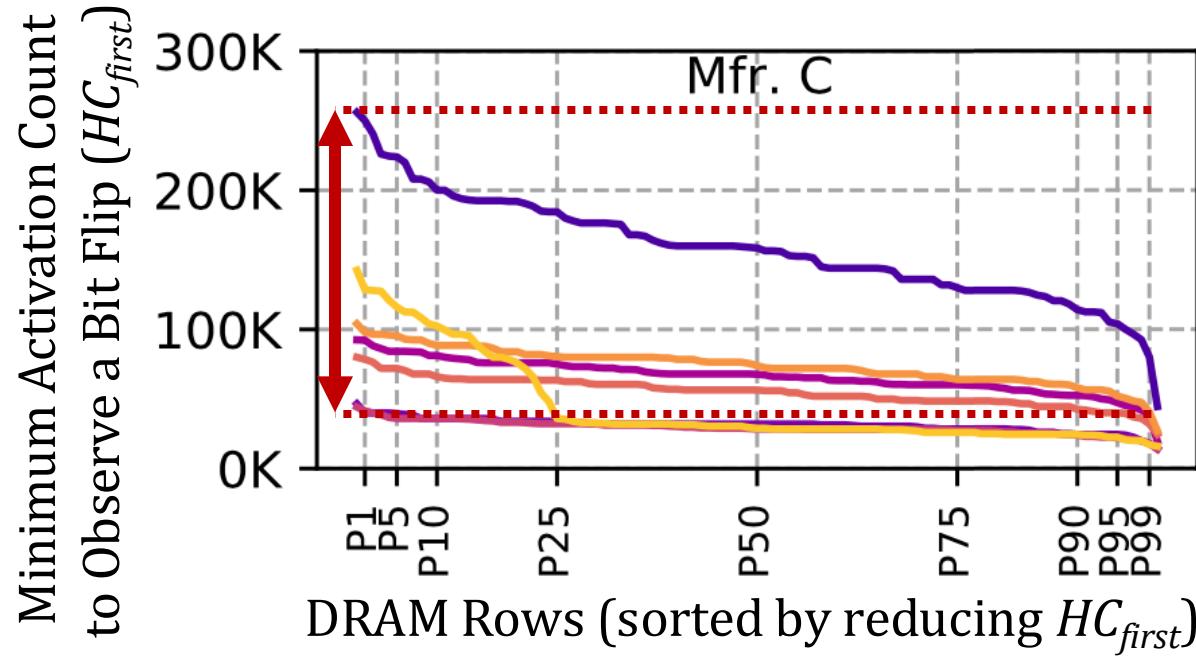
- ☛ increase the time that the aggressor row stays open



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

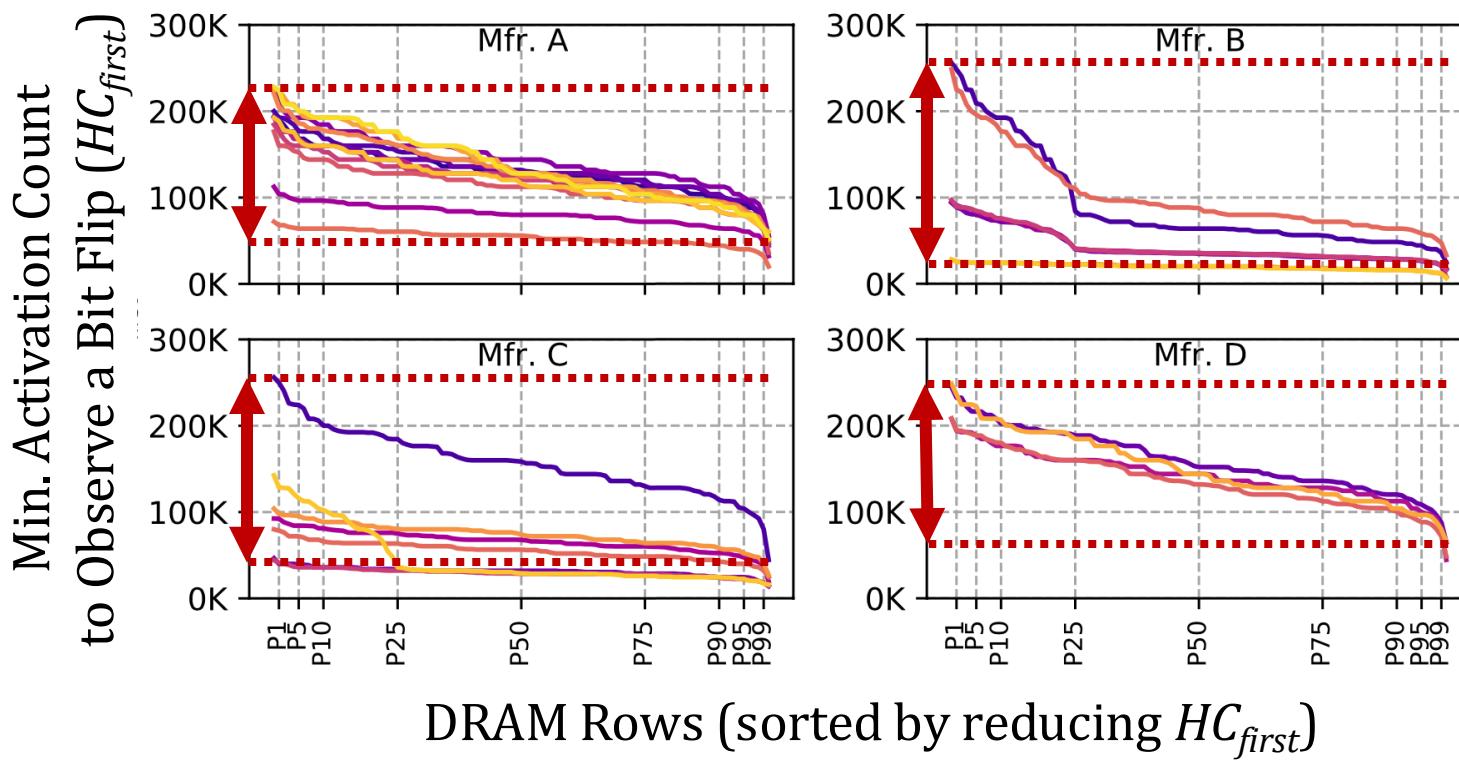
Spatial Variation across Rows

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



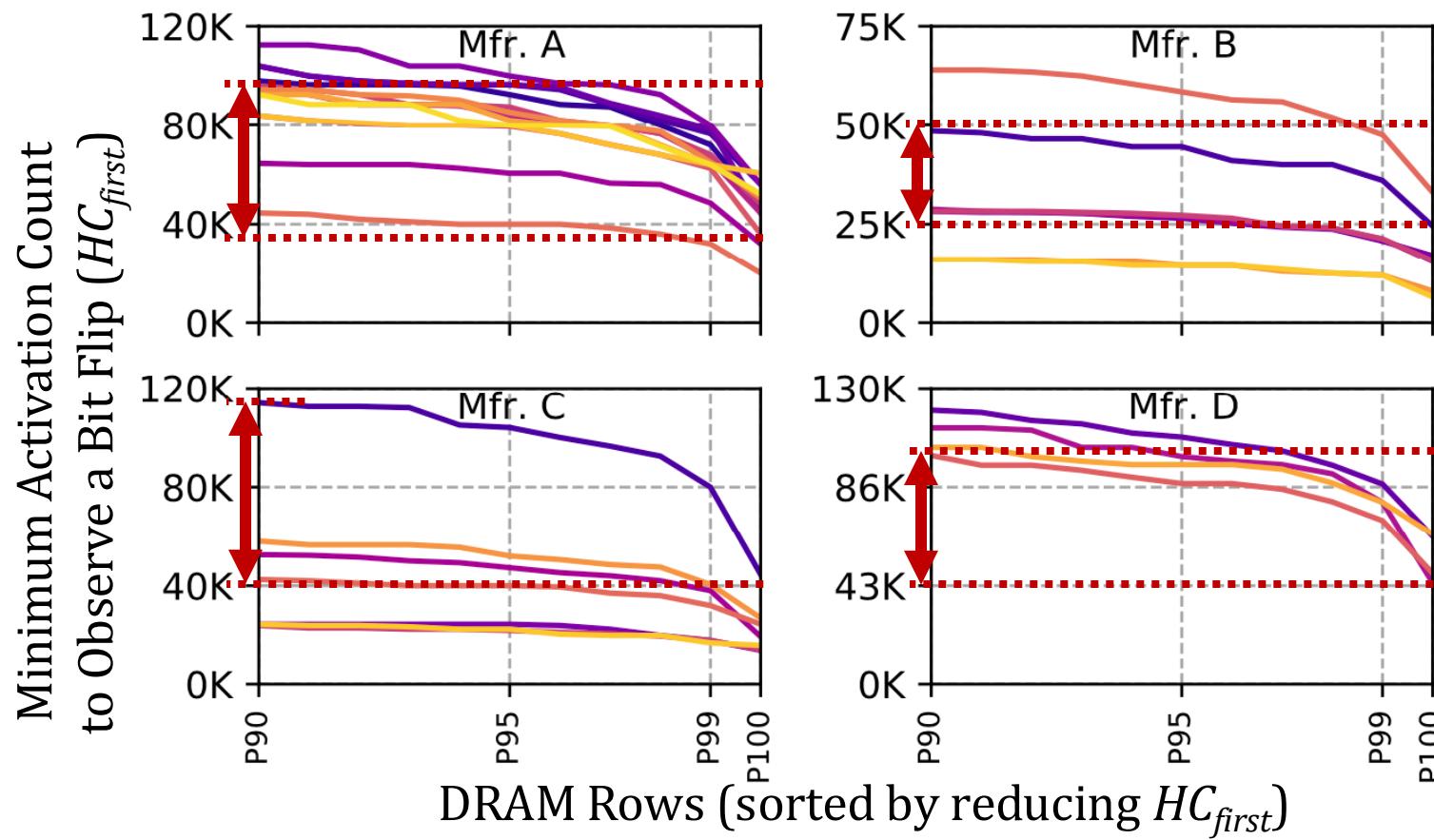
The RowHammer vulnerability
significantly varies across DRAM rows

Spatial Variation across Rows



The RowHammer vulnerability
significantly varies across DRAM rows

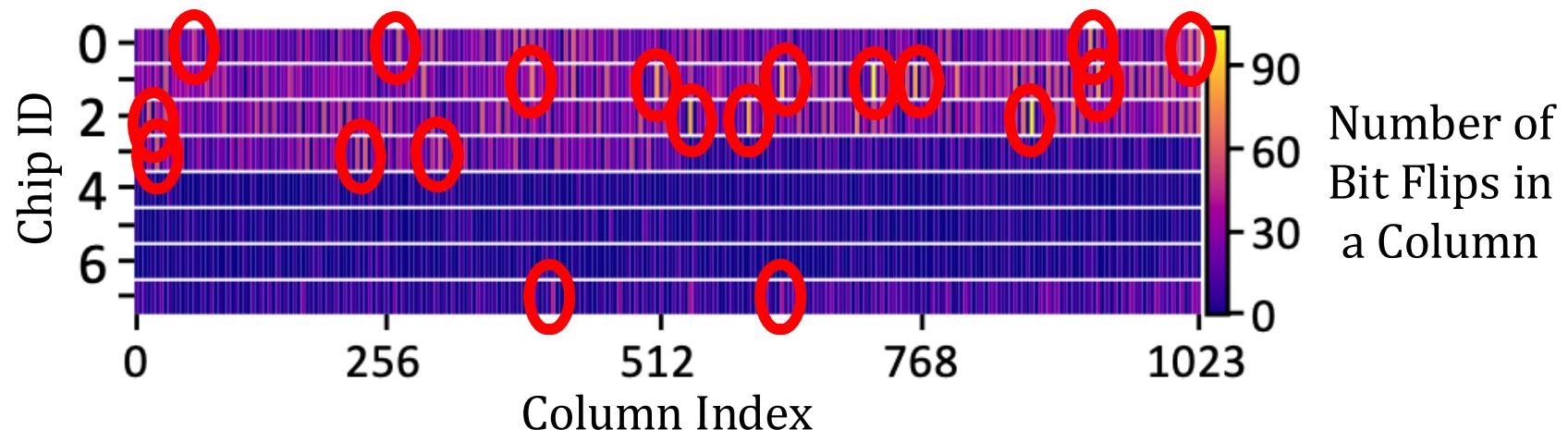
Spatial Variation across Rows



OBSERVATION 12

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

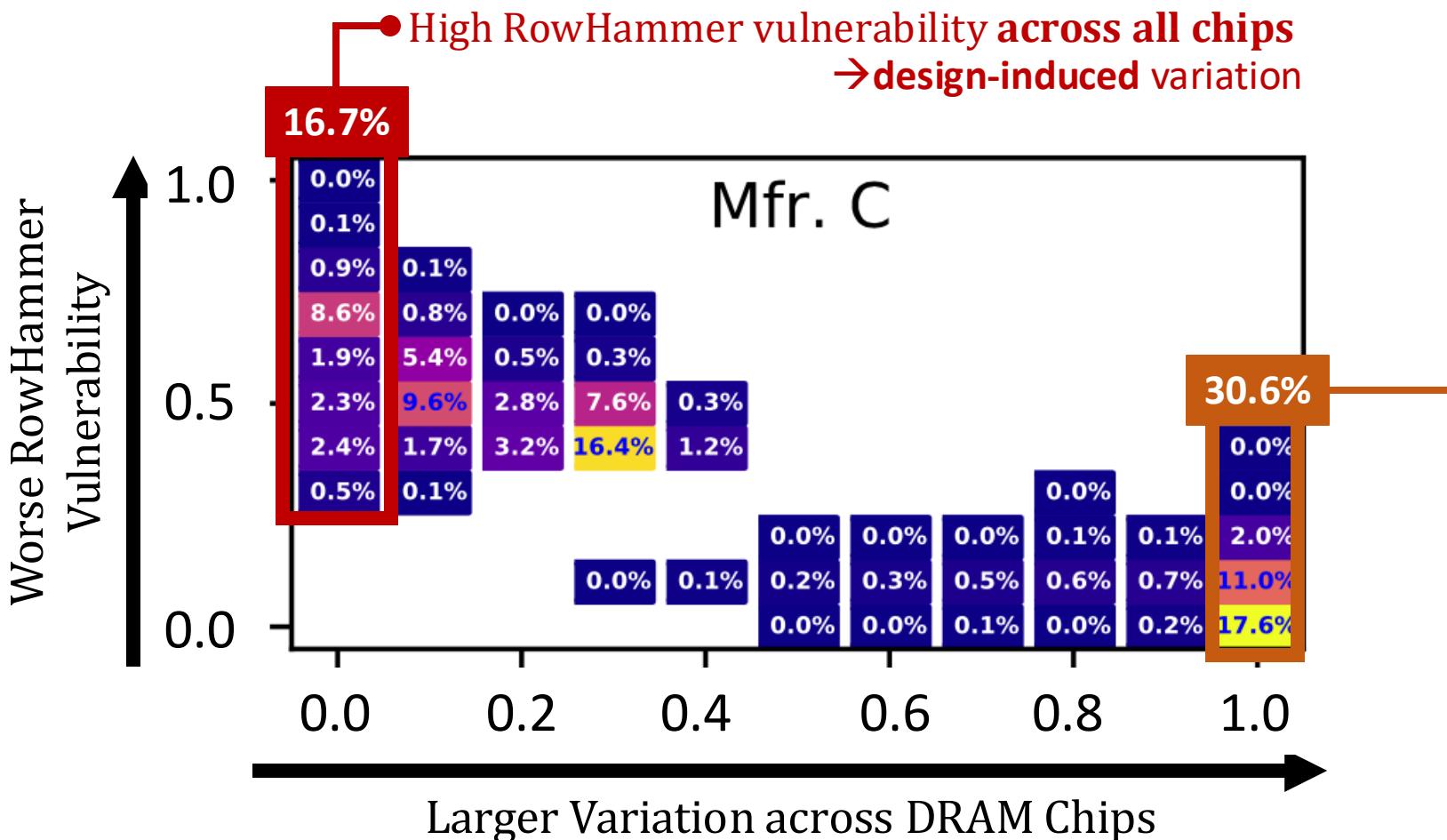
Spatial Variation across Columns



OBSERVATION 13

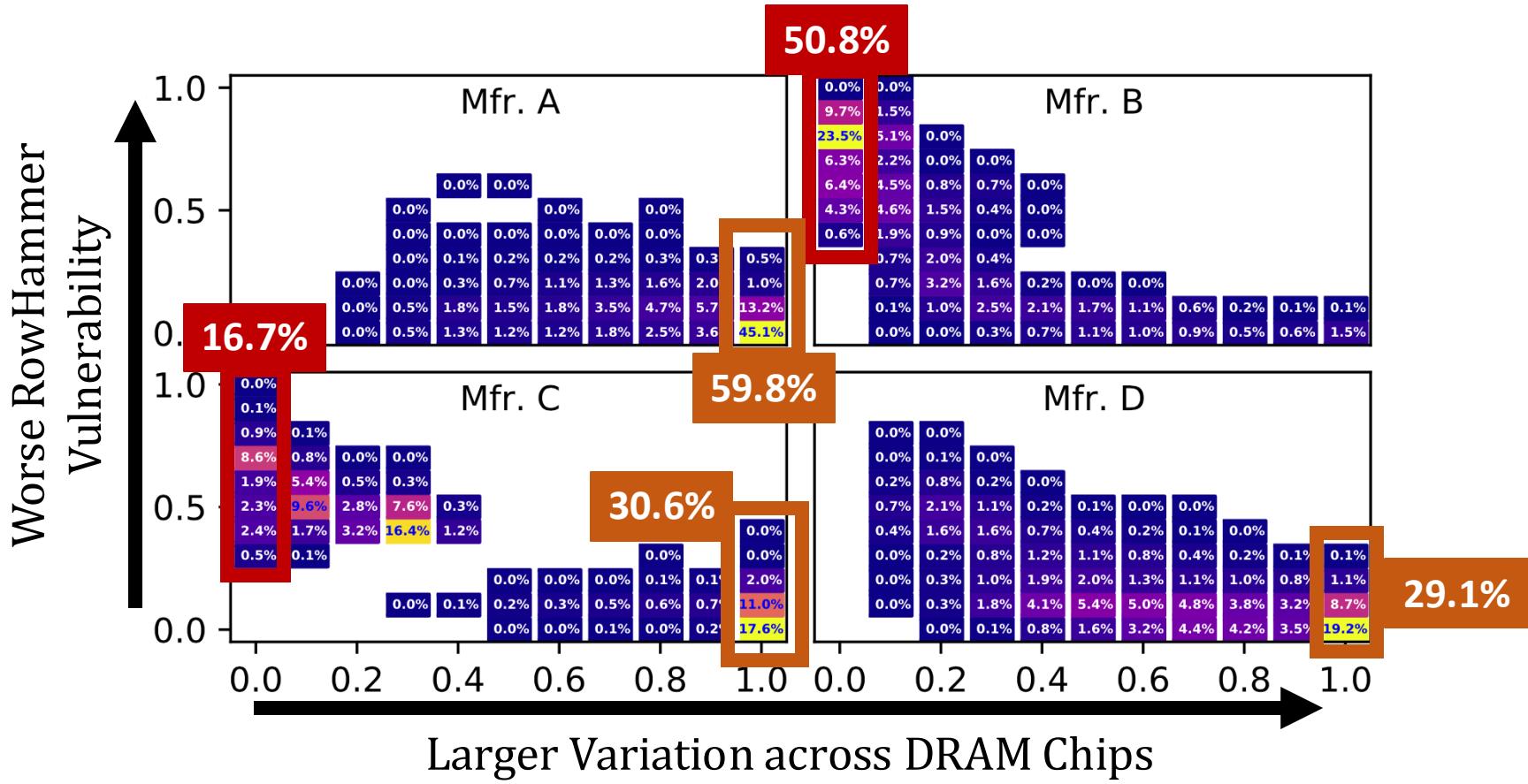
Certain columns are **significantly more vulnerable** to RowHammer than other columns

Spatial Variation across Columns



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

Spatial Variation across Columns



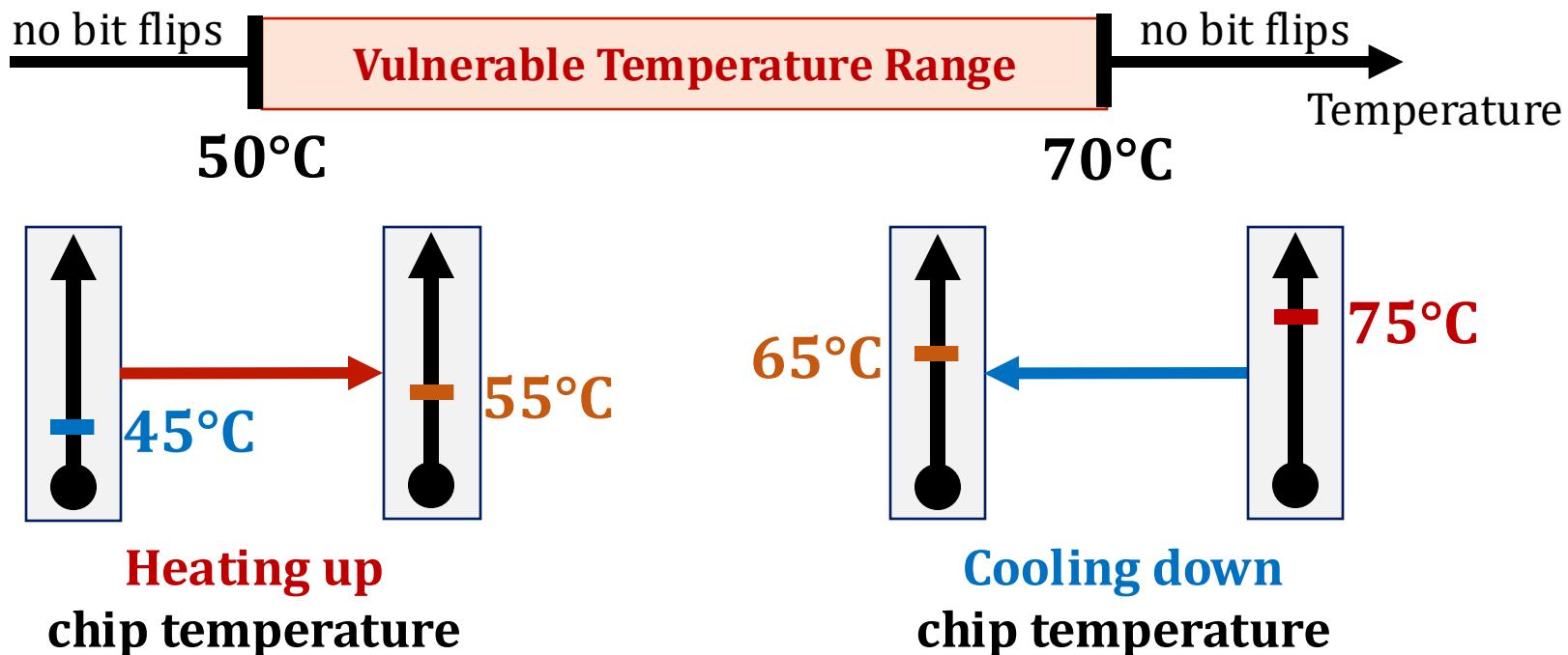
OBSERVATION 14

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

Attack Improvement 1: Making DRAM Cells More Vulnerable

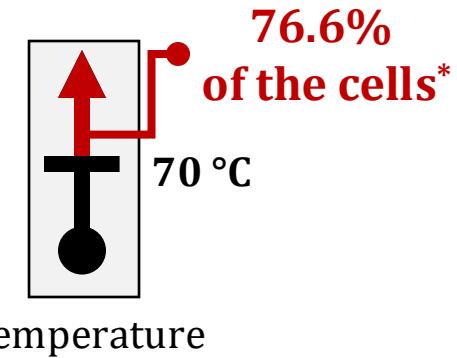
An attacker can **manipulate temperature** to make the cells that store sensitive data **more vulnerable**

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

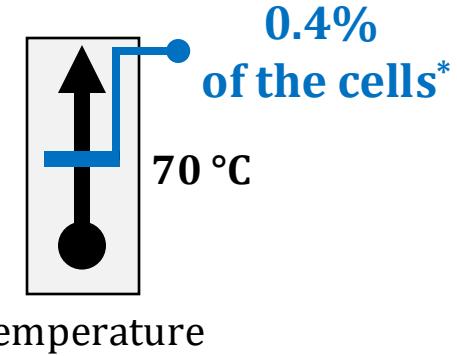


Attack Improvement 2: Temperature-Dependent Trigger

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



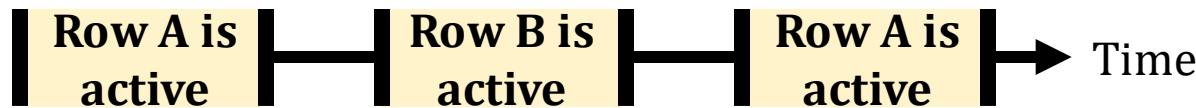
2. Precisely measure the temperature **to trigger an attack** exactly at the desired temperature



*Example fraction values from Mfr. C

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:

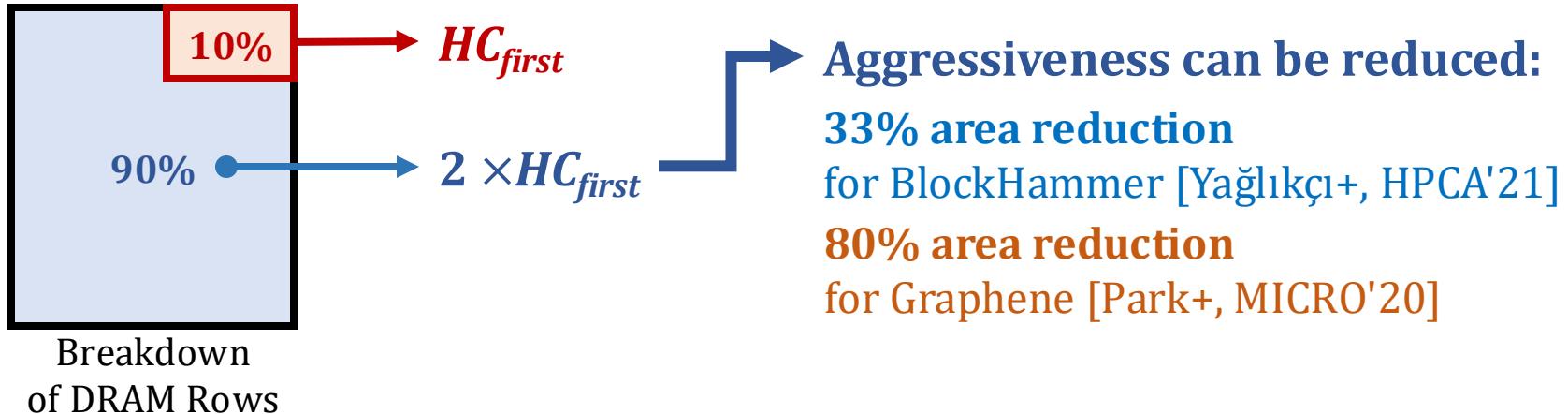


Reduces the minimum activation count to induce a bit flip **by 36%**

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**



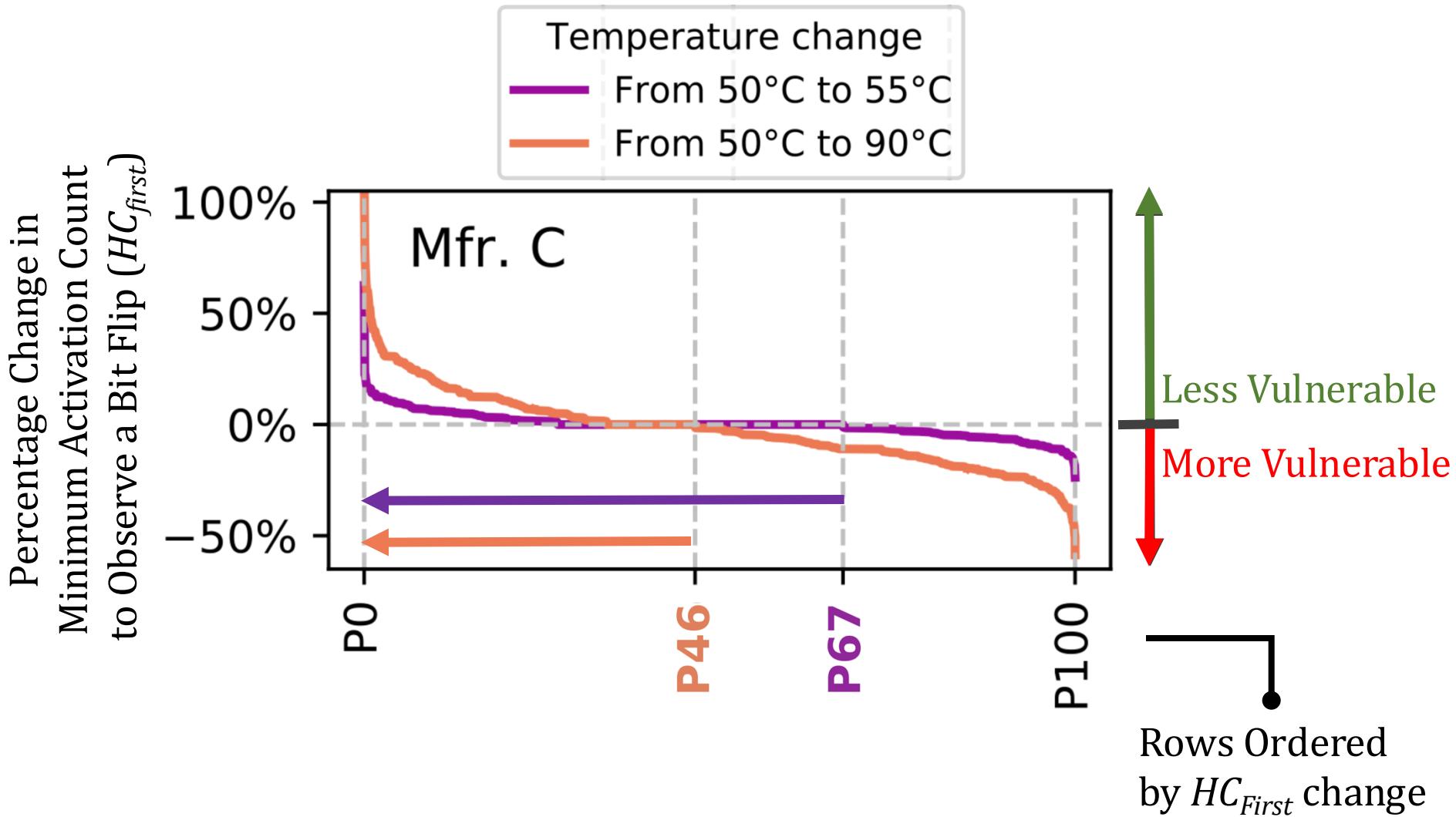
- A row can be **disabled** within the row's **vulnerable temperature range**



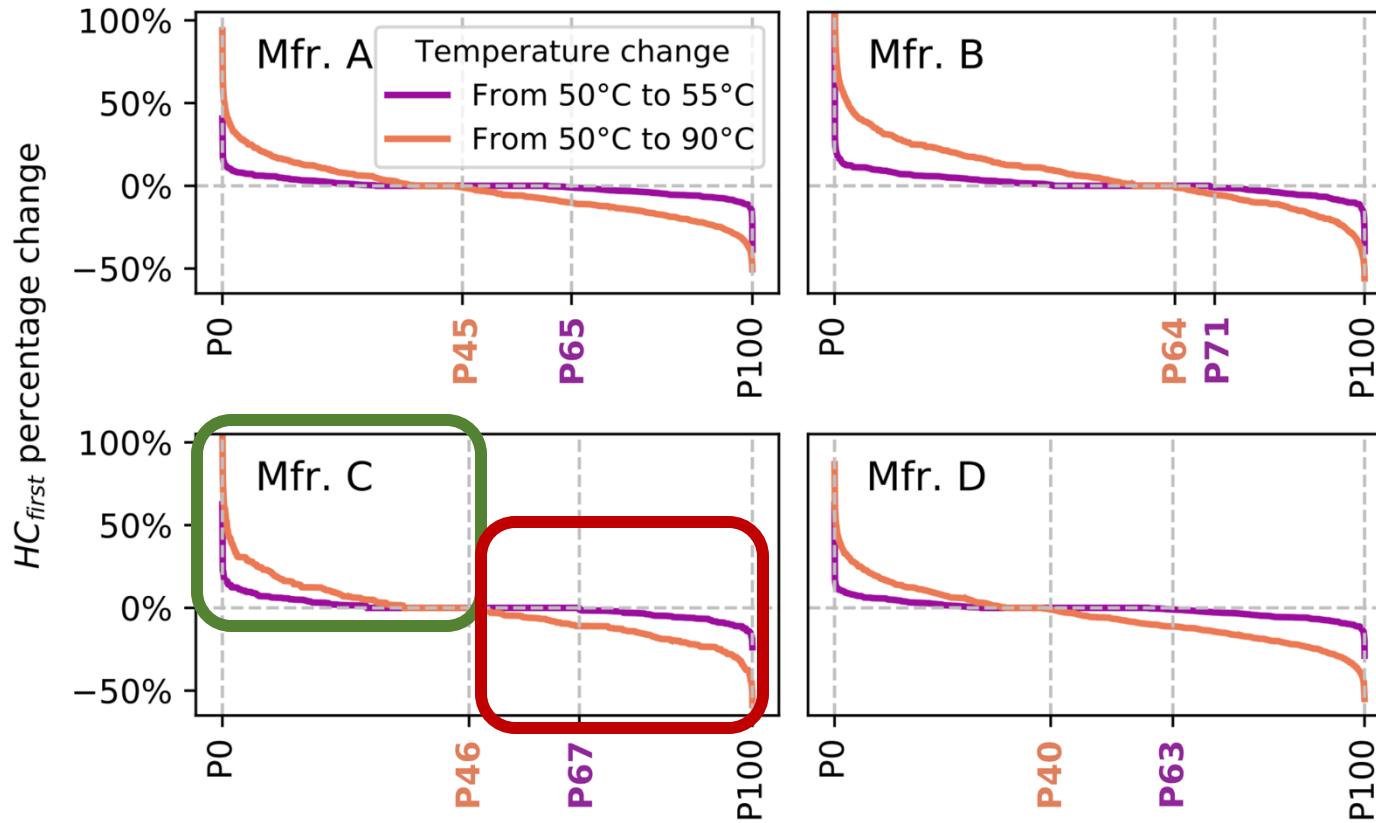
More Defense Implications in the Paper

- 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**

Distribution of the Change in HC_{first}



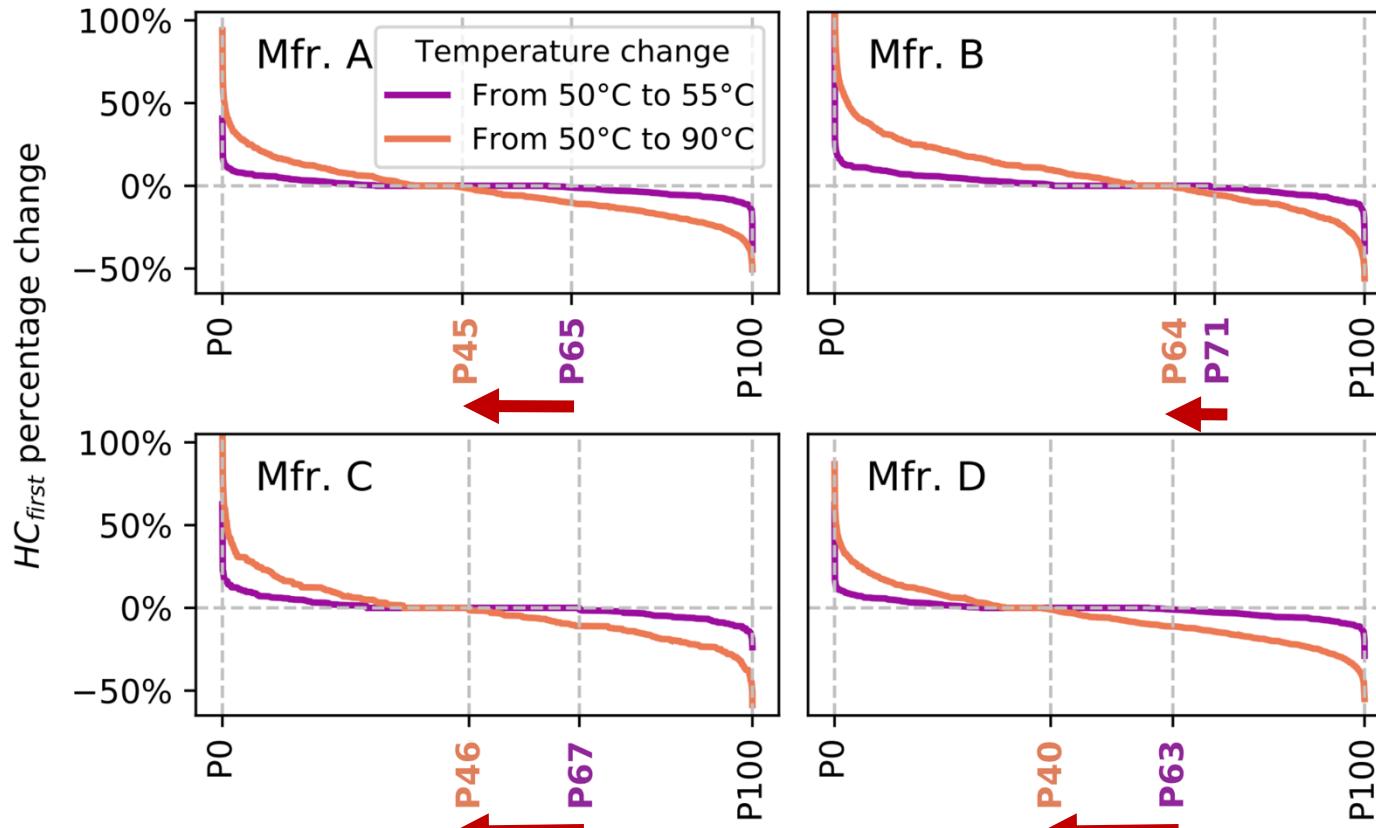
Distribution of the Change in HC_{first}



OBSERVATION 5

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

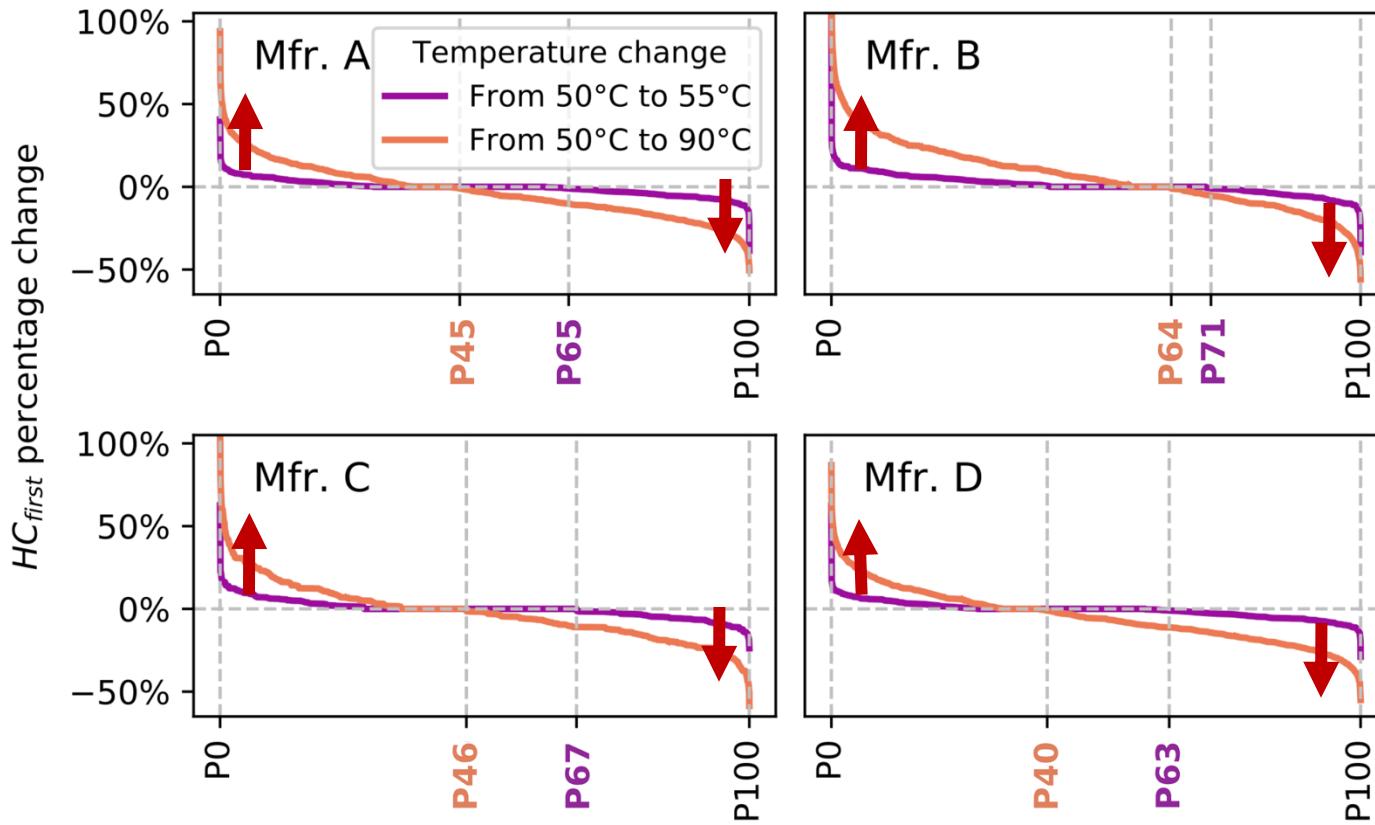
Distribution of the Change in HC_{first}



OBSERVATION 6

HC_{first} tends to generally **decrease** as **temperature change (ΔT)** increases

Distribution of the Change in HC_{first}



OBSERVATION 7

The HC_{first} change (ΔHC_{first}) tends to be **larger** as **temperature change (ΔT) 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]

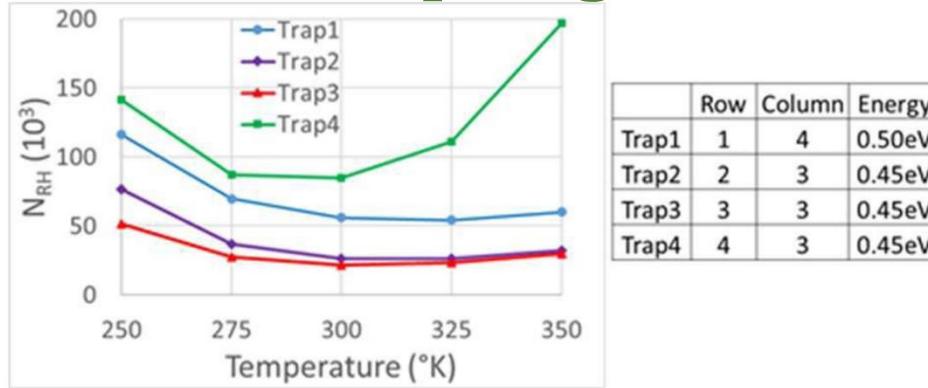


Fig. 6. Hammering threshold N_{RH} vs. temperature from 250 to 350°K for different traps. Location in row and column refers to matrix in **Fig. 2b**.

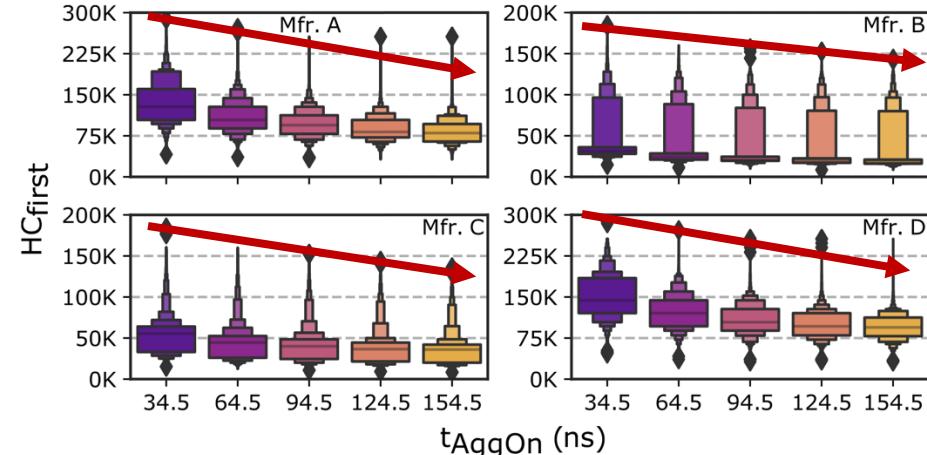
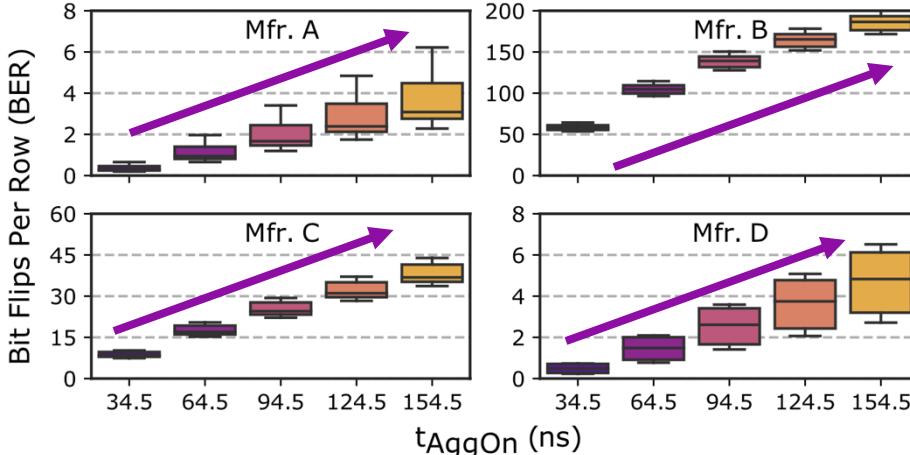
HC_{first} decreases as temperature increases, until a temperature inflection point where **HC_{first} starts to increase as temperature increases**

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

Increasing Aggressor Row Active Time (t_{AggOn})



Bit Error Rate



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_{first} change across rows when the **aggressor row stays active longer**

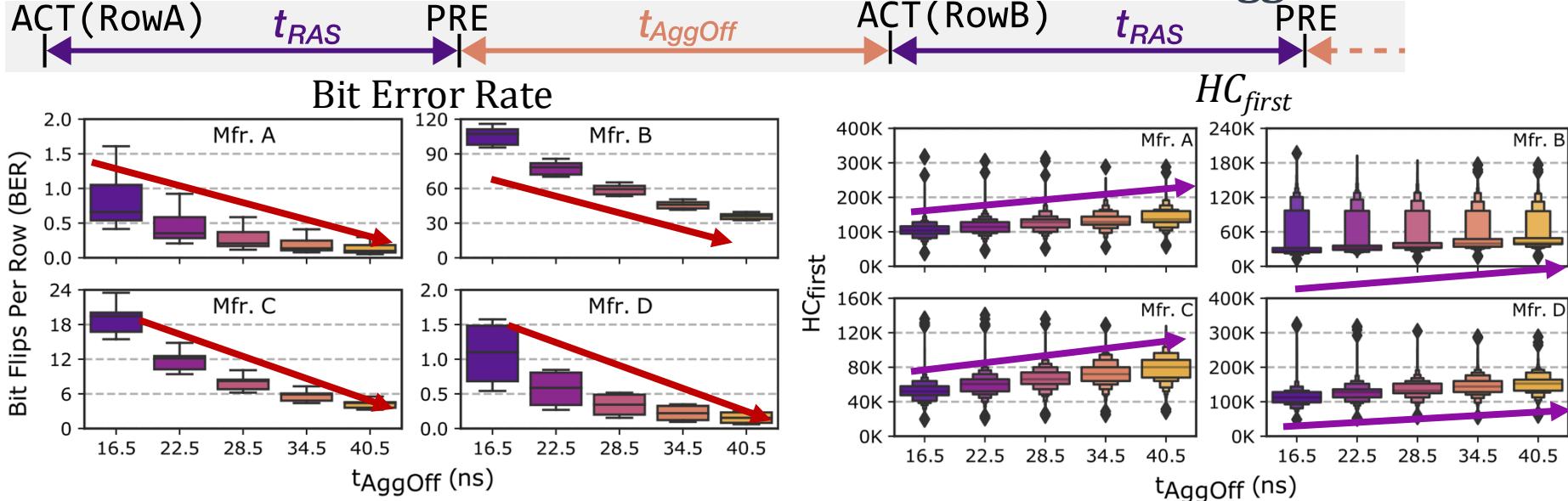
OBSERVATION 9

RowHammer vulnerability **consistently worsens**
as t_{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_{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_{first}** change
across rows when the **bank stays precharged longer**

OBSERVATION 11

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

*Coefficient of Variation (CV) = $\text{Standard Deviation}/\text{Average}$

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

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})

HC_{first} worst-to-best ratio in this range: P100/P90

120K

80K

0K

P90

P95

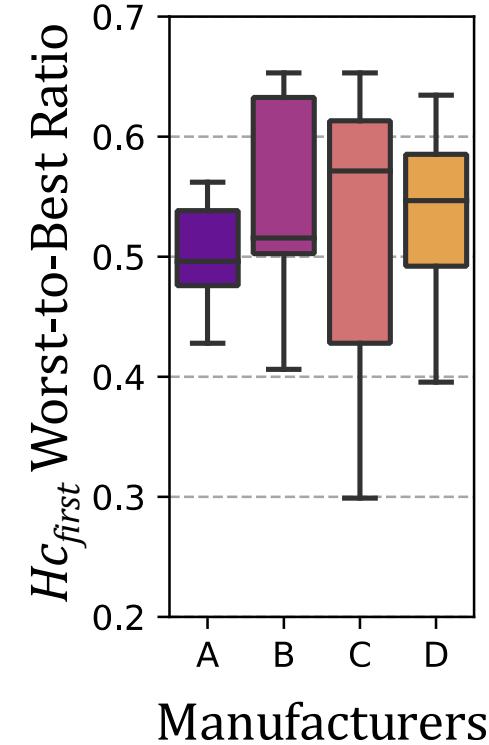
P99

P100

DRAM Rows (sorted by reducing HC_{first})

Mfr. C

HC_{first} worst-to-best ratio in this range: P100/P90

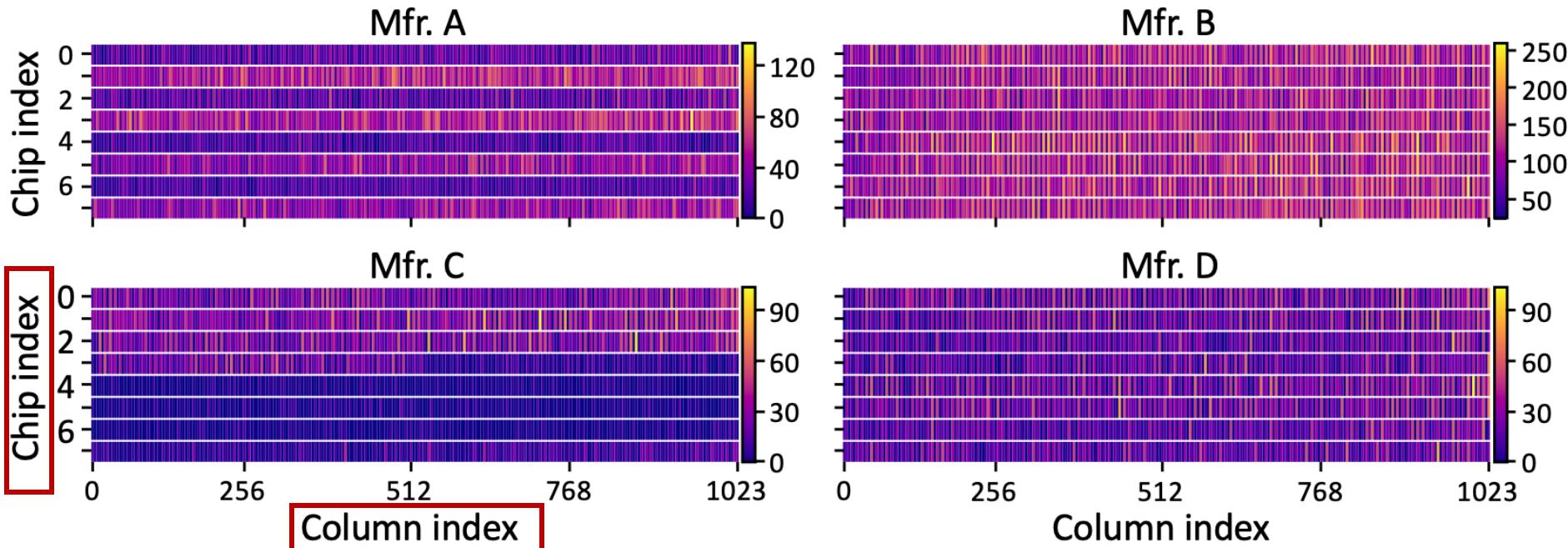


OBSERVATION 12

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

Spatial Variation across Columns

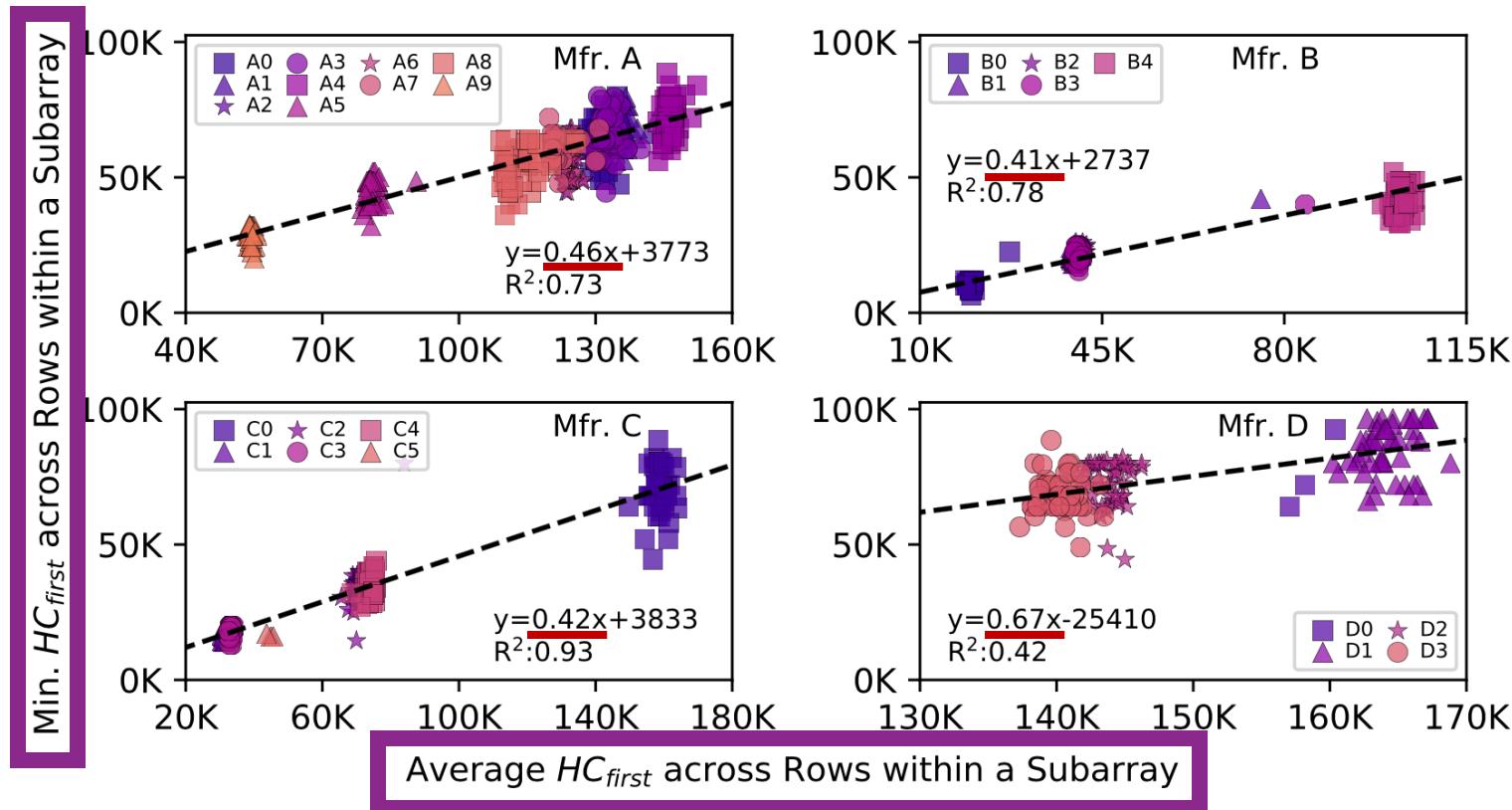
We analyze BER variation across DRAM columns



OBSERVATION 13

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

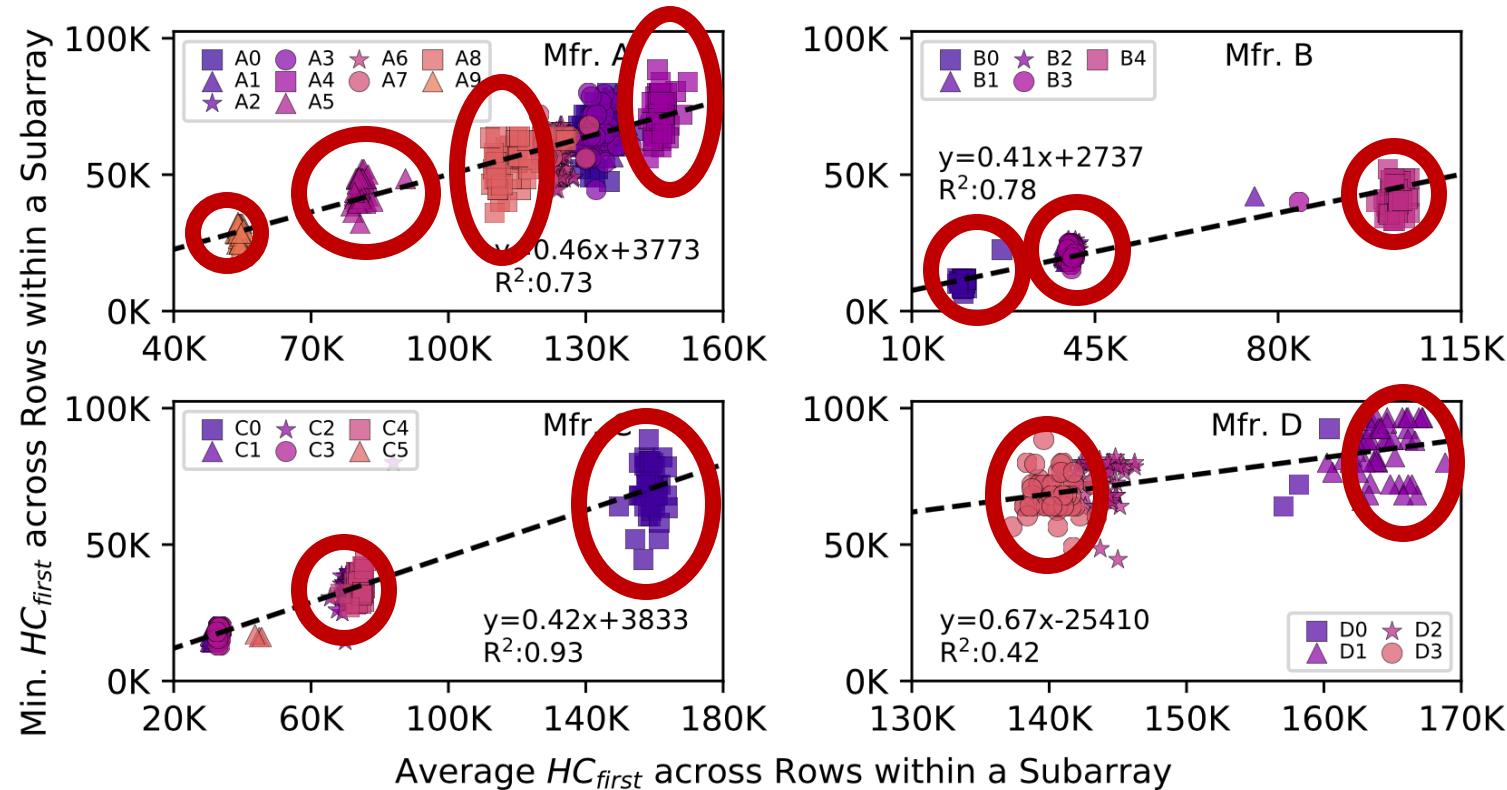
Spatial Variation across Subarrays



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



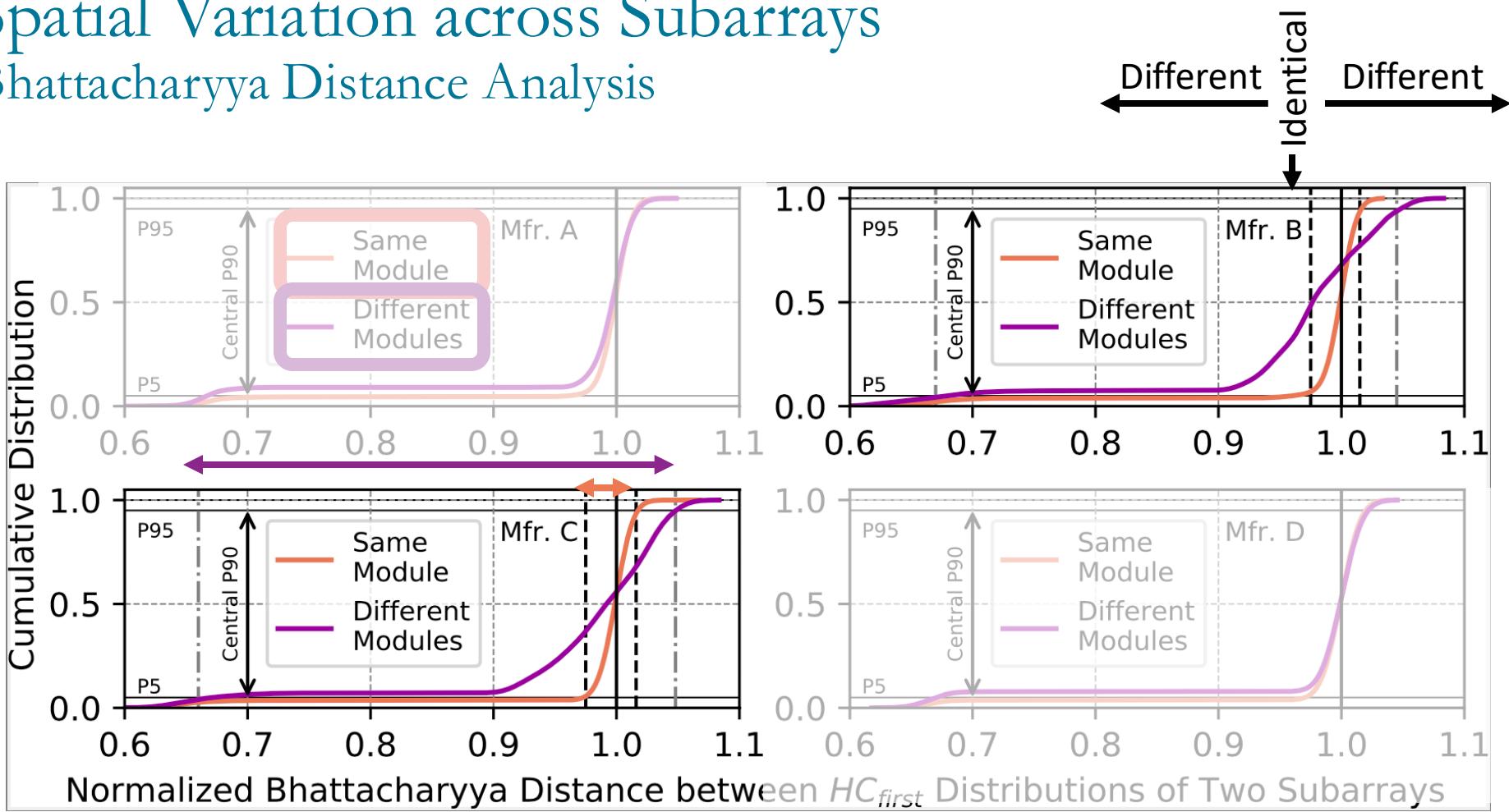
OBSERVATION 16

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:

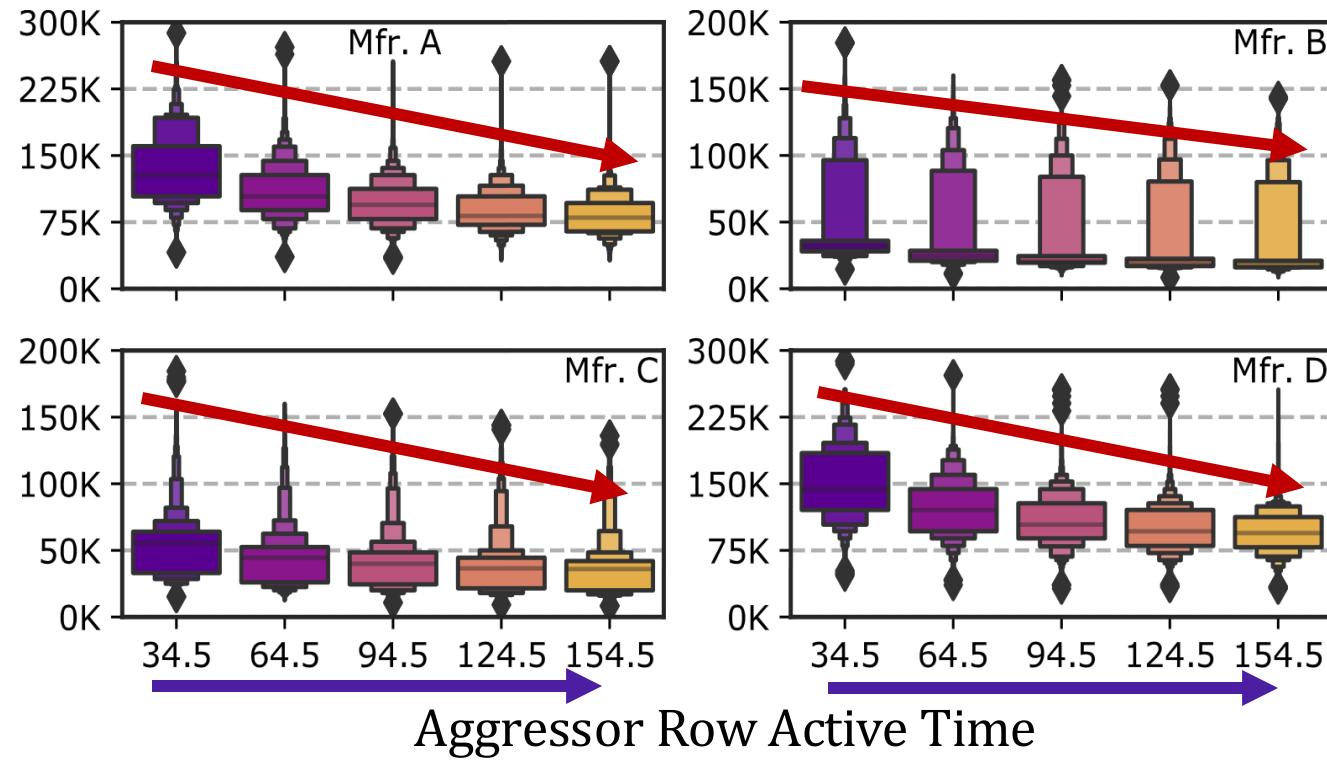
Manufacturing process variation causes **differences in cell size and bitline/wordline impedance values**, which introduces variation in cell reliability characteristics within and across DRAM chips

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**

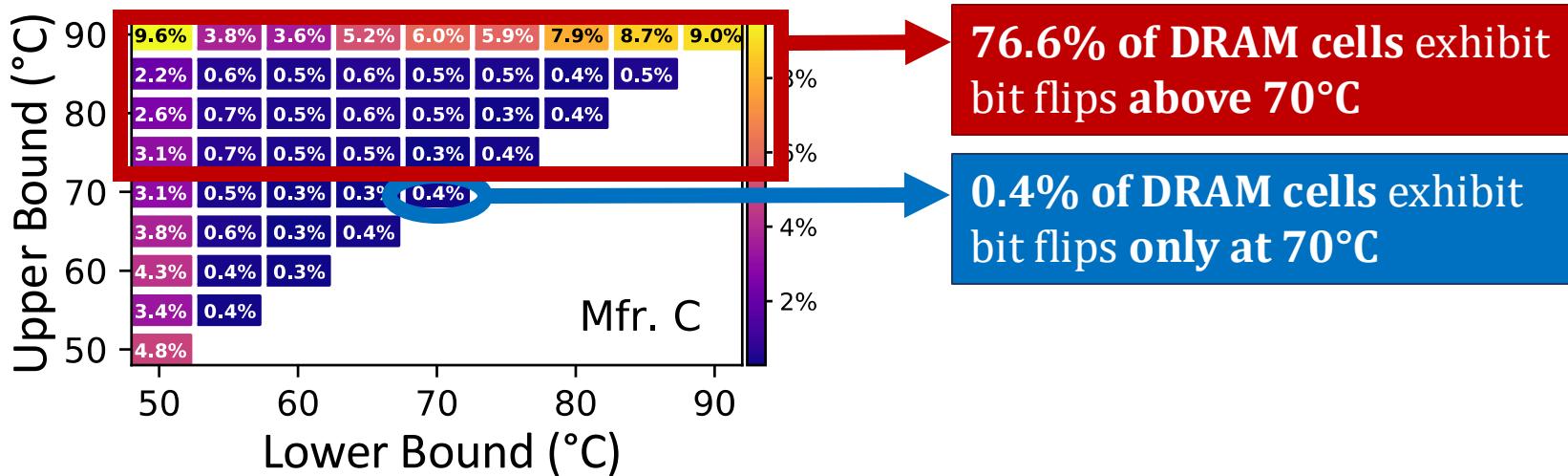
Example Attack Improvements



- 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

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**

Understanding RowHammer Under Reduced Wordline Voltage

An Experimental Study Using Real DRAM Devices

Abdullah Giray Yağlıkçı

Haocong Luo Geraldo F. de Oliviera Ataberk Olgun

Minesh Patel Jisung Park Hasan Hassan Jeremie S. Kim

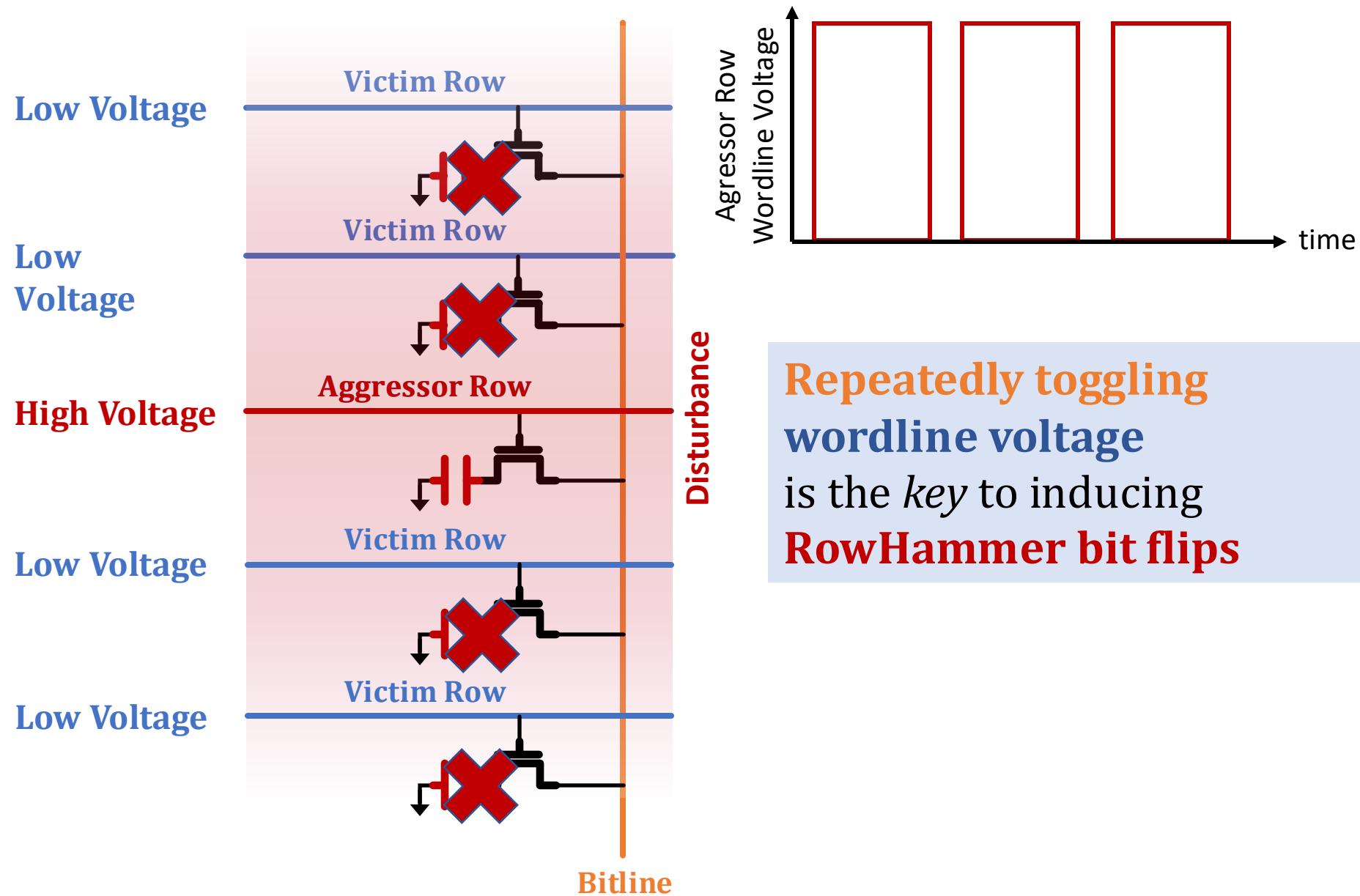
Lois Orosa Onur Mutlu

ETH zürich

SAFARI

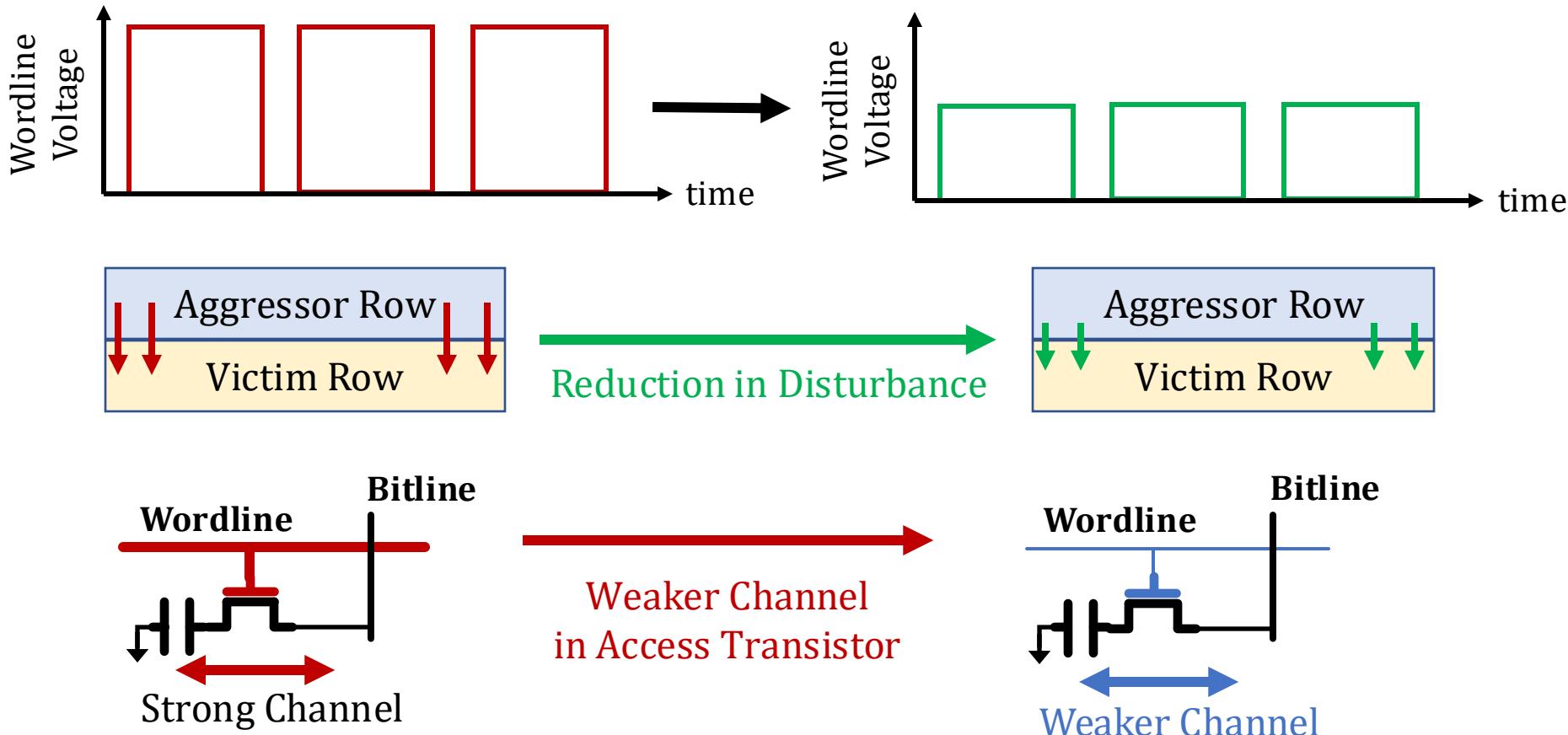
 CESGA
Centro de Supercomputación de Galicia

A Closer Look into RowHammer



Our Hypothesis

Reducing wordline voltage
can **reduce RowHammer vulnerability**
without significantly affecting reliable DRAM operation



DRAM Chips Tested

Mfr.	# DIMMs	# Chips	Density	Die	Org.	Date
A (Micron)	1	8	4Gb	-	x8	48-16
	4	64	8Gb	B	x4	11-19
	3	24	4Gb	F	x8	07-21
	2	16	4Gb	-	x8	
B (Samsung)	2	16	8Gb	B	x8	52-20
	1	8	8Gb	C	x8	19-19
	3	24	8Gb	D	x8	10-21
	1	8	4Gb	E	x8	08-17
	1	8	4Gb	F	x8	02-21
	2	16	8Gb		x8	
C (SK Hynix)	2	16	16Gb	A	x8	51-20
	3	24	4Gb	B	x8	02-21
	2	16	4Gb	C	x8	
	3	24	8Gb	D	x8	48-20

3 Major Manufacturers

272 DDR4 DRAM Chips

More Details in the Paper

272 DDR4 DRAM Chips

Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices

A. Giray Yağlıkçı¹ Haocong Luo¹ Geraldo F. de Oliviera¹ Ataberk Olgun¹ Minesh Patel¹
Jisung Park¹ Hasan Hassan¹ Jeremie S. Kim¹ Lois Orosa^{1,2} Onur Mutlu¹

¹ETH Zürich

²Galicia Supercomputing Center (CESGA)

Alg. 1: Test for HC_{first} and BER for a Given V_{PP}

```
// RAvictim: victim row address
// WCDP: worst-case data pattern
// HC: number of activations per aggressor row
Function measure_BER(RAvictim, WCDP, HC):
    initialize_row(RAvictim, WCDP)
    initialize_aggressor_rows(RAvictim, bitwise_inverse(WCDP))
    hammer_doublesided(RAvictim, HC)
    BERrow = compare_data(RAvictim, WCDP)
    return BERrow

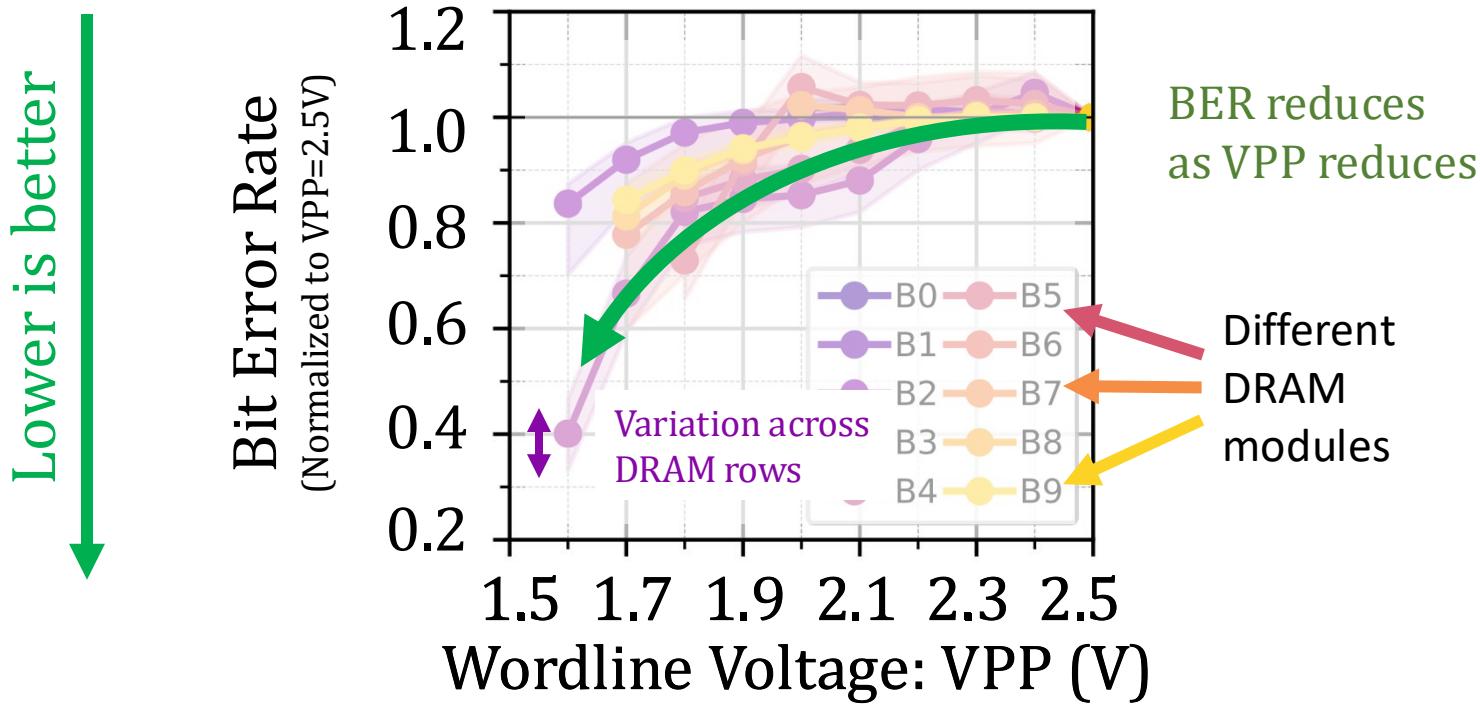
// Vpp: wordline voltage for the experiment
// WCDP_list: the list of WCDPs (one WCDP per row)
// row_list: the list of tested rows
Function test_loop(Vpp, WCDP_list):
    set_vpp(Vpp)
    foreach RAvictim in row_list do
        HC = 300K // initial hammer count to test
        HCstep = 100K // how much to increment/decrement HC
        while HCstep > 100 do
            BERrowmax = 0
            for i ← 0 to num_iterations do
                BERrow = measure_BER(RAvictim, WCDP, HC)
                record_BER(Vpp, RAvictim, WCDP, HC, BERrow, i)
                BERrowmax = max(BERrowmax, BERrow)
            end
            if BERrowmax == 0 then
                | HC += HCstep // Increase HC if no bit flips occur
            end
            else
                | HC -= HCstep // Reduce HC if a bit flip occurs
            end
            HCstep = HCstep/2
        end
    record_HCfirst(Vpp, RAvictim, WCDP, HC)
```

Table 3: Tested DRAM modules and their characteristics when $V_{PP}=2.5$ V (nominal) and $V_{PP}=V_{PPmin}$. V_{PPmin} is specified for each module.

Mfr. A (Micron)	DIMM Model	Die Density	Frequency (MHz)	Chip Org.	Die Revision	Mfr. Date	$V_{PP} = 2.5$ V	$V_{PP} = V_{PPmin}$	$V_{PP} = V_{PPmin}$	Recommended $V_{PP} (V_{PPrec})$	$V_{PP} = V_{PPrec}$
A0	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	39.8K	1.24e-03	1.4	42.2K	1.00e-03
A1	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	42.2K	9.90e-04	1.4	46.4K	7.83e-04
A2	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	41.0K	1.24e-03	1.7	39.8K	1.35e-03
A3	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	16.7K	3.33e-02	1.4	16.5K	3.52e-02
A4	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	14.4K	3.18e-02	1.5	14.4K	3.33e-02
A5	CT4G4DFS8213.C8FB'D1	4Gb	2400	x8	-	48-16	140.7K	1.39e-06	2.4	145.4K	3.39e-06
A6	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	16.5K	3.50e-02	1.5	16.5K	3.66e-02
A7	CMV4GX4M1A213C15 [150]	4Gb	2133	x8	-	-	16.5K	3.42e-02	1.8	16.5K	3.52e-02
A8	MTA18ASF2G72PZ-2G3B1QG [148]	8Gb	2400	x4	B	11-19	35.2K	2.38e-03	1.4	39.8K	2.07e-03
A9	CMV4GX4M1A213C15 [150]	4Gb	2133	x8	-	-	14.3K	3.33e-02	1.5	14.3K	3.48e-02
B0	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.9K	1.18e-01	2.0	7.6K	1.22e-01
B1	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.3K	1.26e-01	2.0	7.6K	1.28e-01
B2	F4-2400C17S-8GNT [152]	4Gb	2400	x8	F	02-21	11.2K	2.52e-02	1.6	12.0K	2.22e-02
B3	M393A1K43BB1-CTD6Y [153]	8Gb	2666	x8	B	52-20	16.6K	2.73e-03	1.6	21.1K	1.09e-03
B4	M393A1K43BB1-CTD6Y [153]	8Gb	2666	x8	B	52-20	21.0K	2.95e-03	1.8	19.9K	2.52e-03
B5	M471A5143EB0-CPB [154]	4Gb	2133	x8	E	08-17	21.0K	7.78e-03	1.8	21.0K	6.02e-03
B6	CMK16GX4M2B3200C16 [155]	8Gb	3200	x8	-	-	10.3K	1.14e-02	1.7	10.5K	9.82e-03
B7	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.3K	1.32e-01	2.0	7.6K	1.33e-01
B8	CMK16GX4M2B3200C16 [155]	8Gb	3200	x8	-	-	11.6K	2.88e-02	1.7	10.5K	2.37e-02
B9	M471A524ACB0-CRC [156]	8Gb	2133	x8	C	19-19	11.8K	2.68e-02	1.7	8.8K	2.39e-02
C0	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	19.3K	7.29e-03	1.7	23.4K	6.61e-03
C1	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	19.3K	6.31e-03	1.7	20.6K	5.90e-03
C2	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	9.6K	2.82e-02	1.5	9.2K	2.34e-02
C3	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	9.3K	2.57e-02	1.5	8.9K	2.21e-02
C4	HMAA4GU6AJR8N-XN [158]	16Gb	3200	x8	A	51-20	11.6K	3.22e-02	1.5	11.7K	2.88e-02
C5	HMAA4GU6AJR8N-XN [158]	16Gb	3200	x8	A	51-20	9.4K	3.28e-02	1.5	12.7K	2.85e-02
C6	CMV4GX4M1A213C15 [150]	4Gb	2133	x8	C	-	14.2K	3.08e-02	1.6	15.5K	2.25e-02
C7	CMV4GX4M1A213C15 [150]	4Gb	2133	x8	C	-	11.7K	3.24e-02	1.6	13.6K	2.60e-02
C8	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	11.4K	2.69e-02	1.6	9.5K	2.57e-02
C9	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	12.6K	2.18e-02	1.7	15.2K	1.63e-02

Full paper on arXiv: <https://arxiv.org/abs/2206.09999>

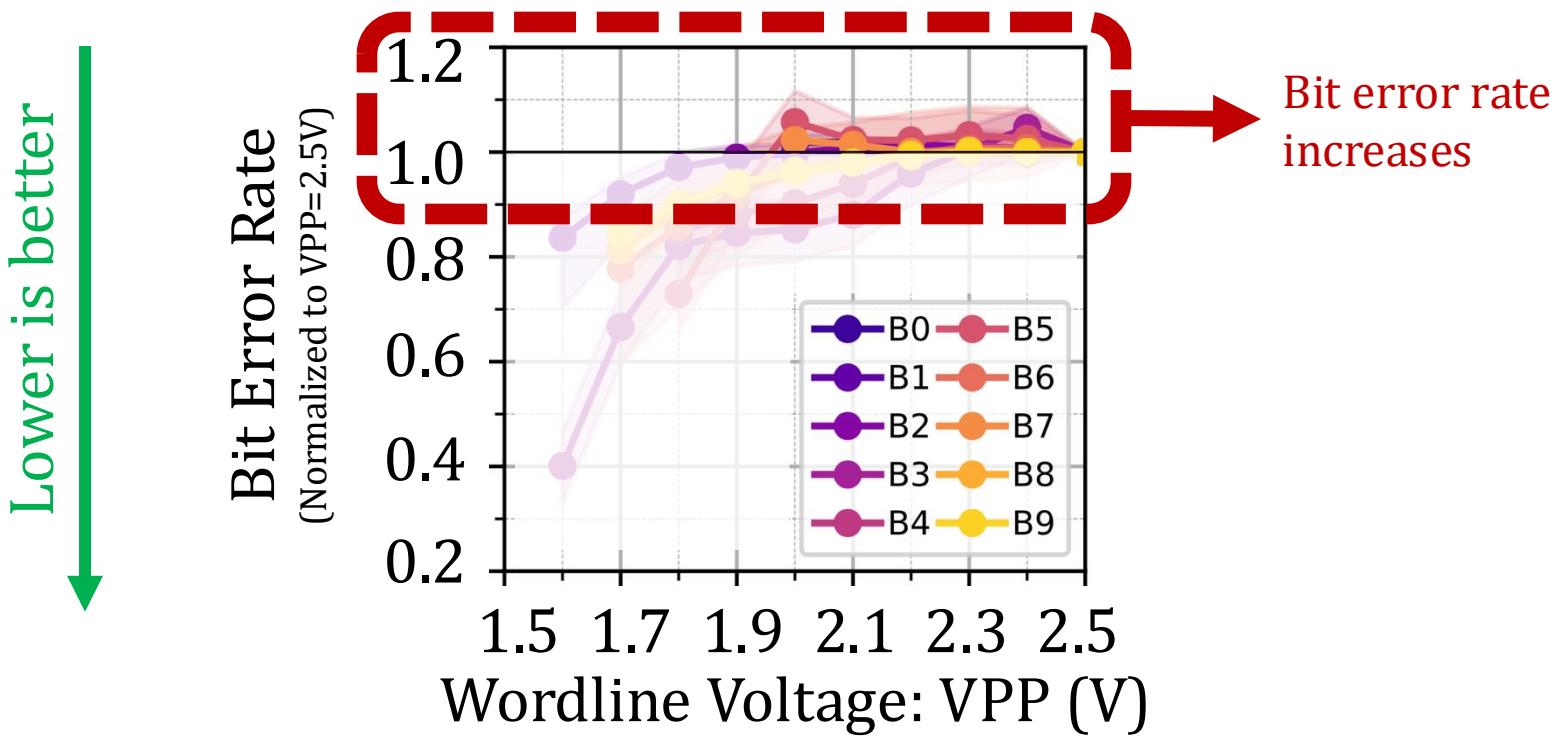
Wordline Voltage's Effect on RowHammer



OBSERVATION 1

Fewer DRAM cells experience RowHammer bit flips under reduced wordline voltage

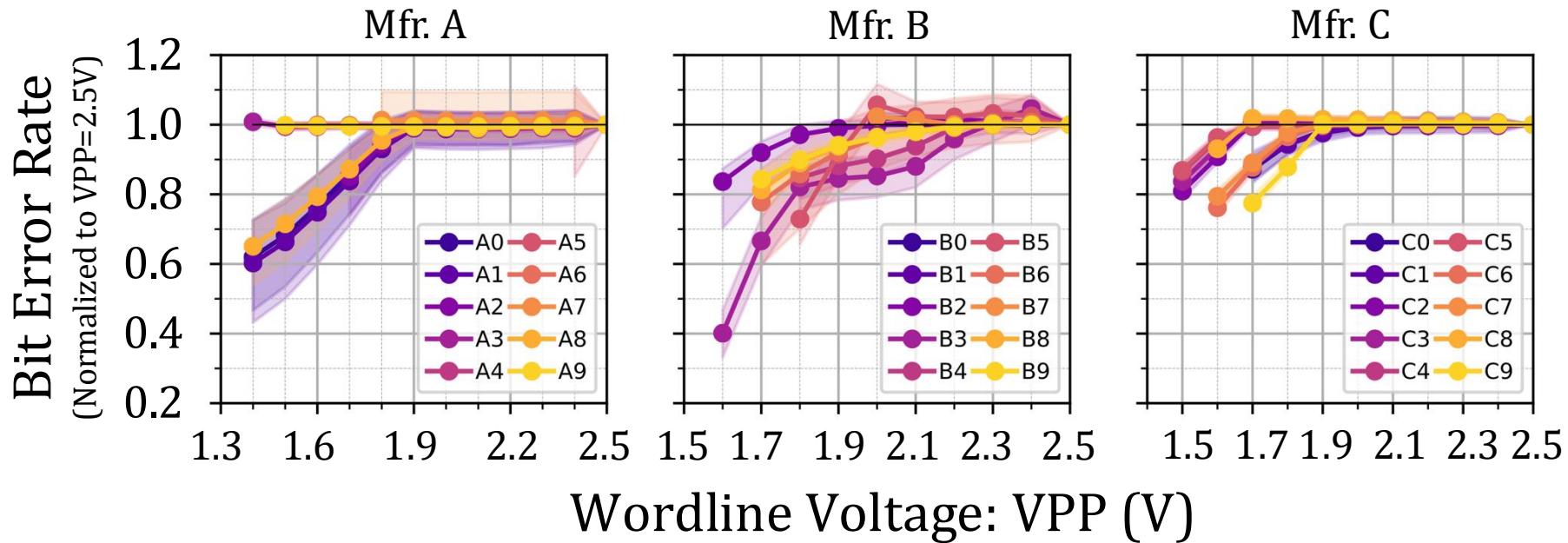
Wordline Voltage's Effect on RowHammer



OBSERVATION 2

Reducing wordline voltage can cause more DRAM cells to experience bit flips in a small fraction of rows (15.4%)

Wordline Voltage's Effect on RowHammer



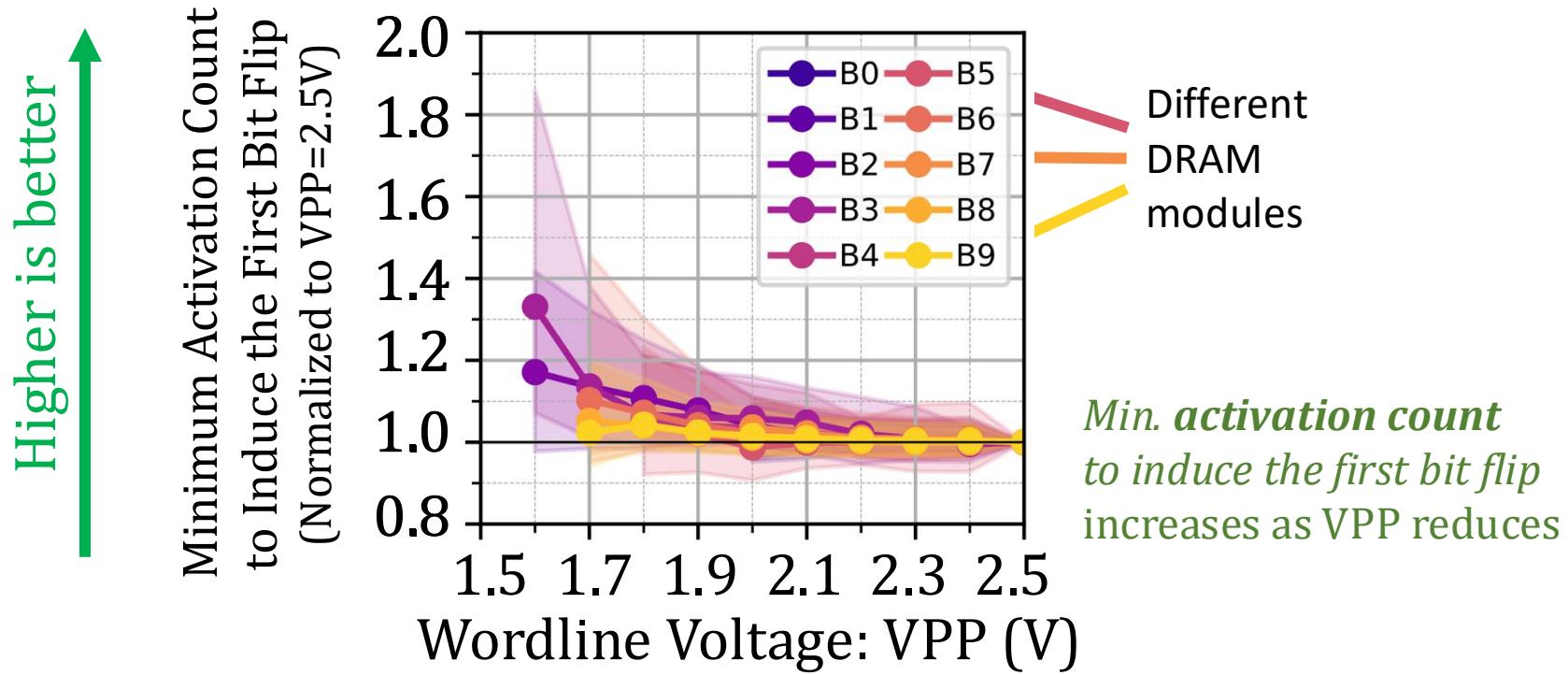
OBSERVATION 1

Fewer DRAM cells experience RowHammer bit flips
under **reduced wordline voltage**

OBSERVATION 2

Reducing wordline voltage can cause **more DRAM cells**
to experience bit flips in **a small fraction of rows (15.4%)**

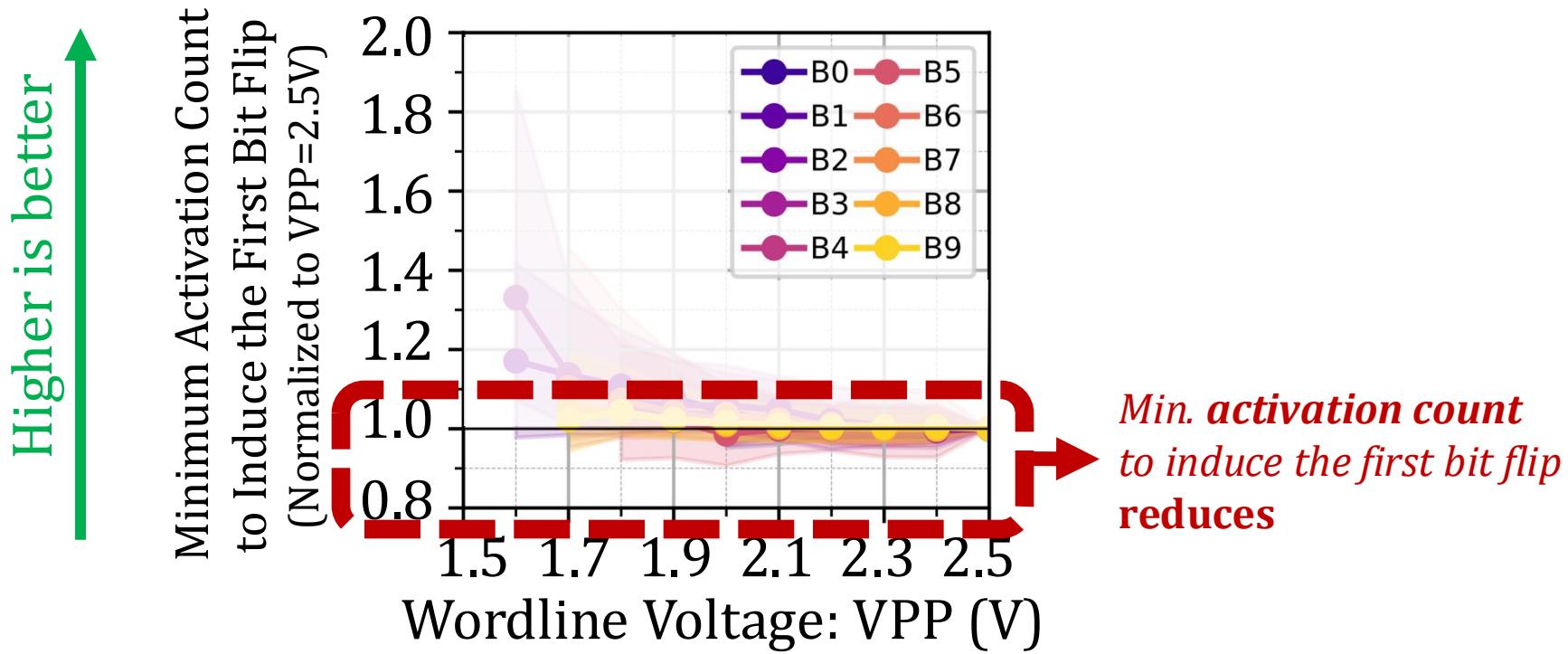
Wordline Voltage's Effect on RowHammer



OBSERVATION 4

The first bit flip occurs at **higher activation counts** as **wordline voltage reduces**

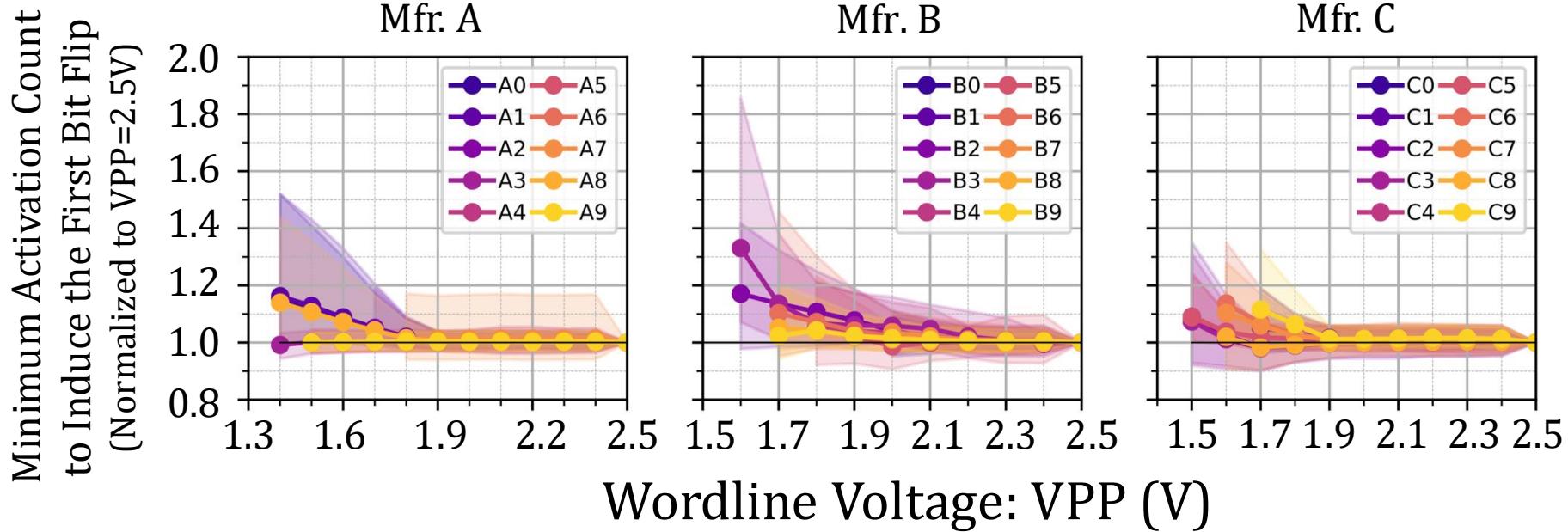
Wordline Voltage's Effect on RowHammer



OBSERVATION 5

For a **small fraction of rows (14.2%)**, the first bit flip occurs at a **smaller activation count** as **wordline voltage reduces**

Wordline Voltage's Effect on RowHammer



OBSERVATION 4

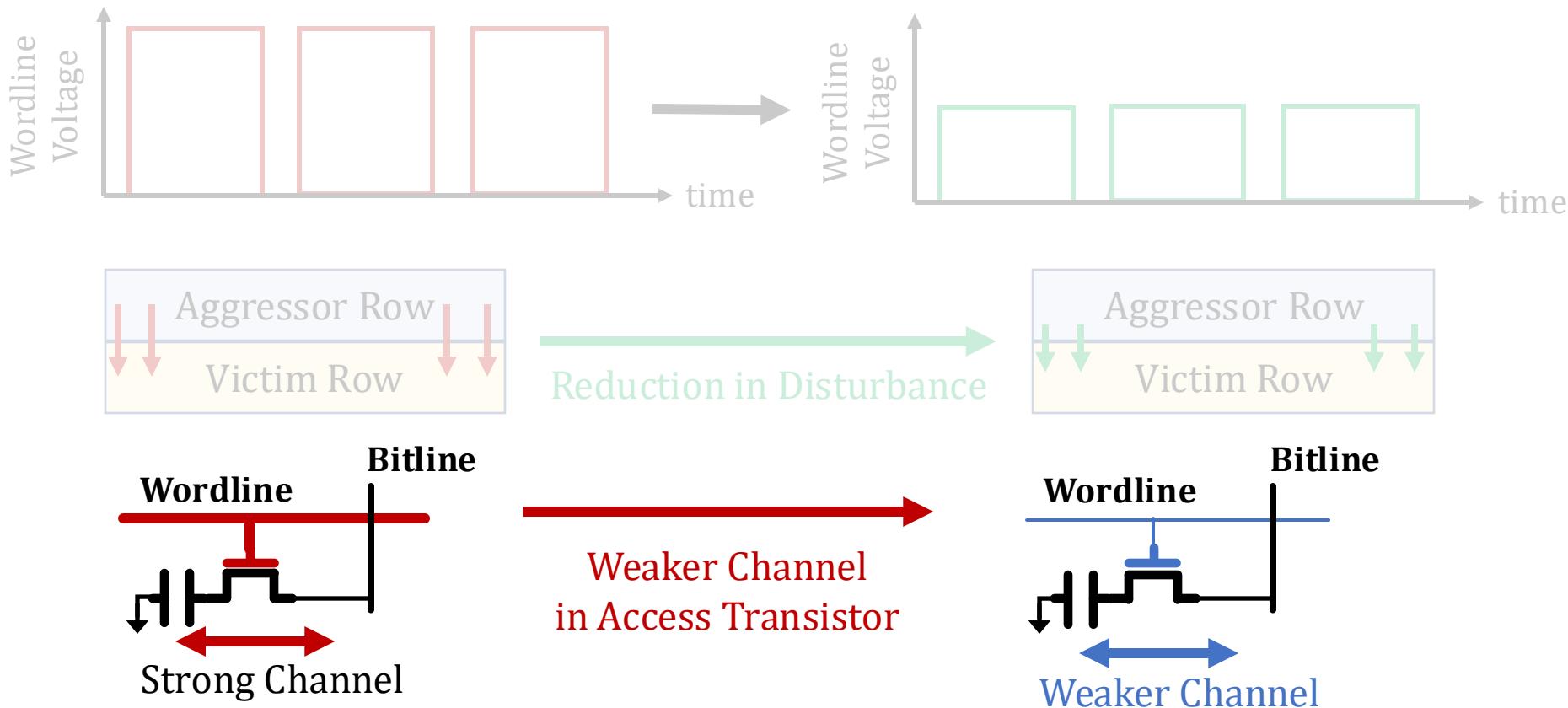
The first bit flip occurs at **higher activation counts** as **wordline voltage reduces**

OBSERVATION 5

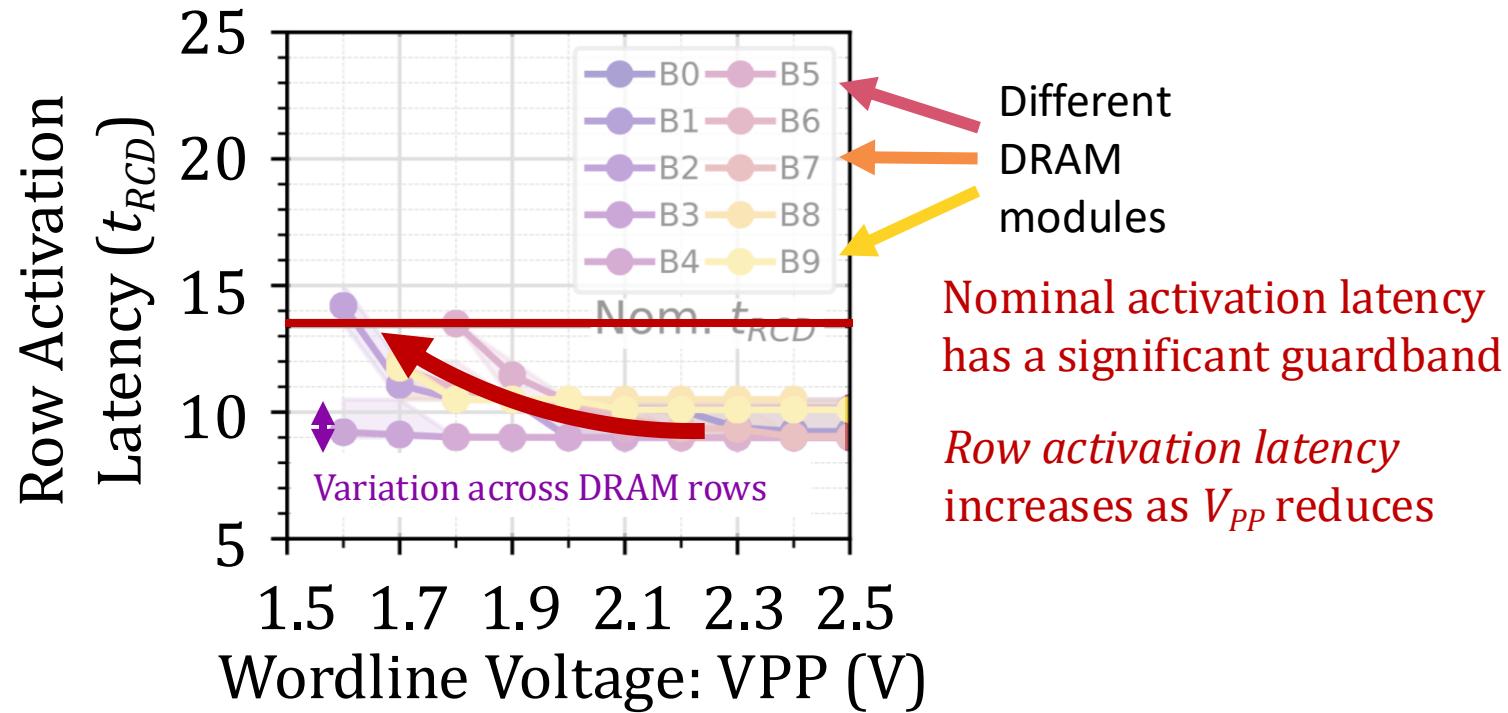
For a **small fraction of rows (14.2%)**, the first bit flip occurs at a **smaller activation count** as **wordline voltage reduces**

Our Hypothesis

Reducing wordline voltage
can **reduce RowHammer vulnerability**
without significantly affecting reliable DRAM operation



Wordline Voltage's Effect on Row Activation Latency

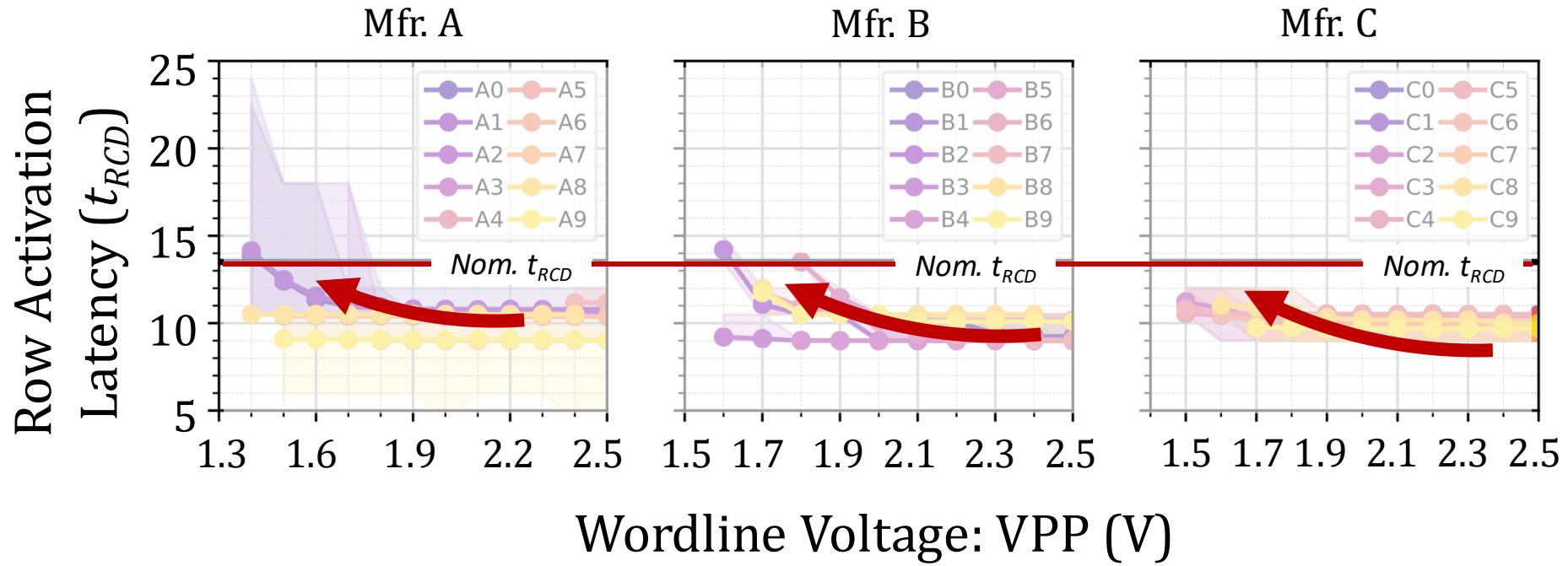


OBSERVATION 7

Row activation latency **increases** with reduced **wordline voltage**

208 out of 272 DRAM chips complete row activation
before the nominal activation latency

Wordline Voltage's Effect on Row Activation Latency

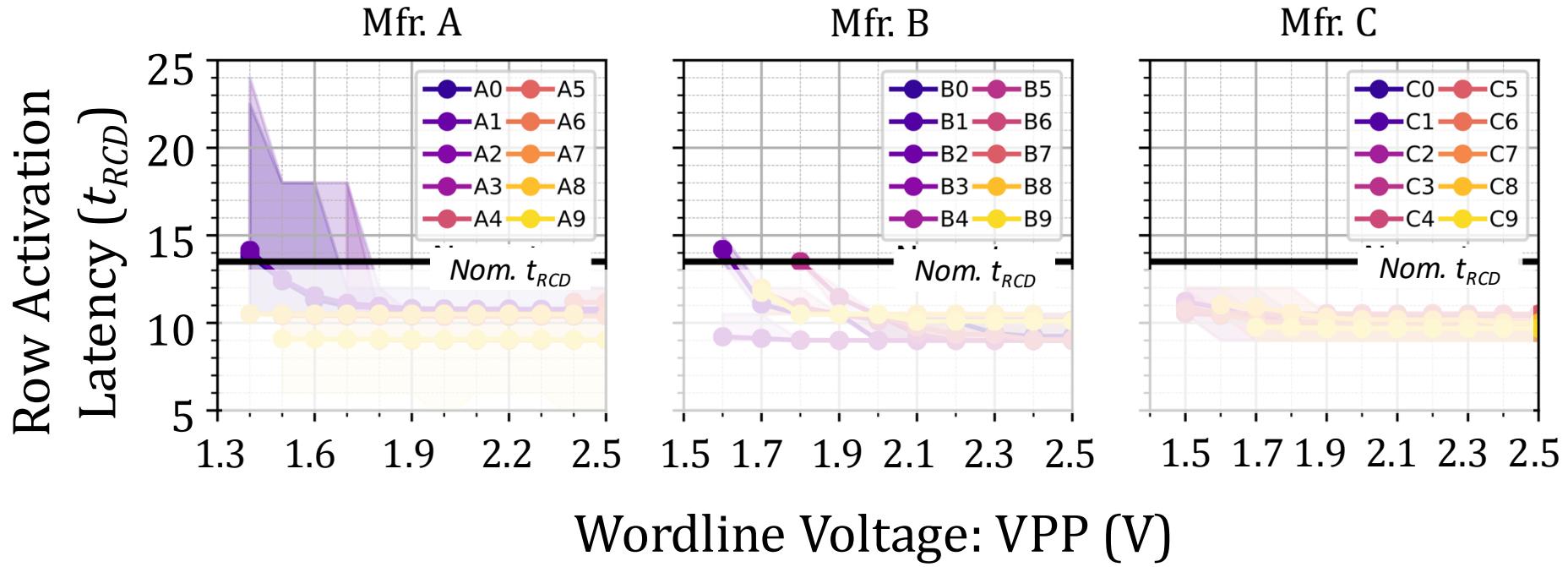


OBSERVATION 7

Row activation latency **increases** with reduced **wordline voltage**

208 out of 272 DRAM chips complete row activation
before the nominal activation latency

Wordline Voltage's Effect on Row Activation Latency



48 DRAM chips from Mfr A. **reliably work** with a row activation latency of **24 ns**

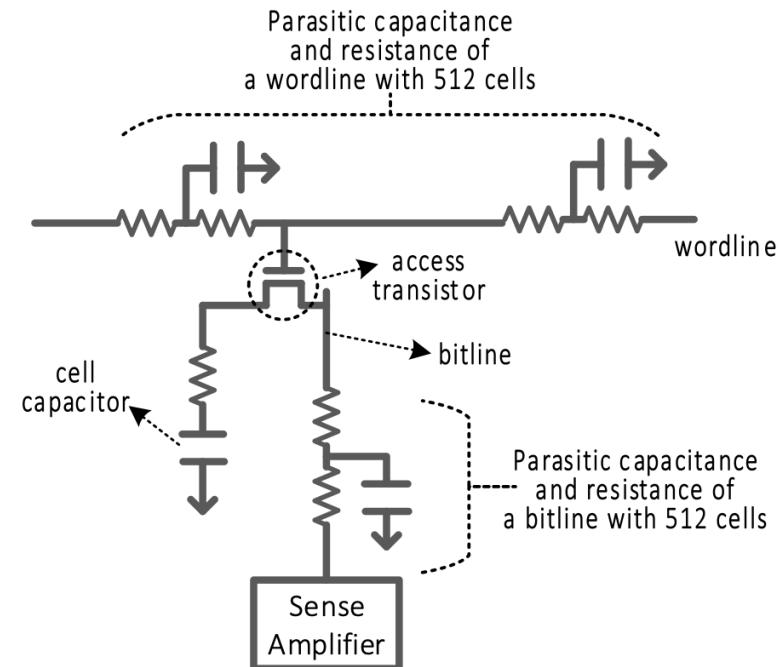
16 DRAM chips from Mfr. B **reliably work** with a row activation latency of **15 ns**

All DRAM chips from Mfr. C **reliably work** using the nominal latency of **13.5ns**

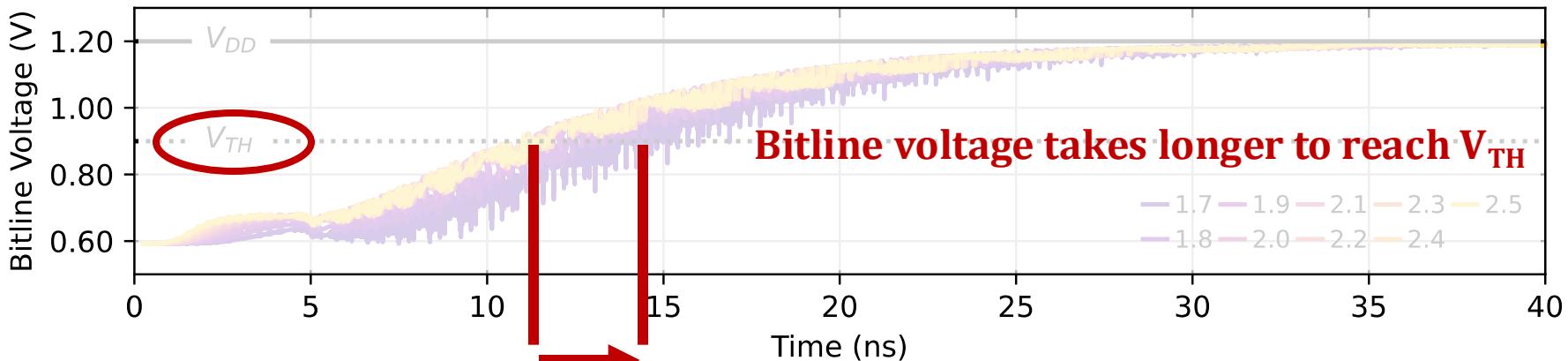
SPICE Simulation Methodology

- Insights into **wordline voltage's affect on DRAM operation**
- 22 nm transistor model
- **Monte-Carlo analysis** with 5% variation and **10K iterations**

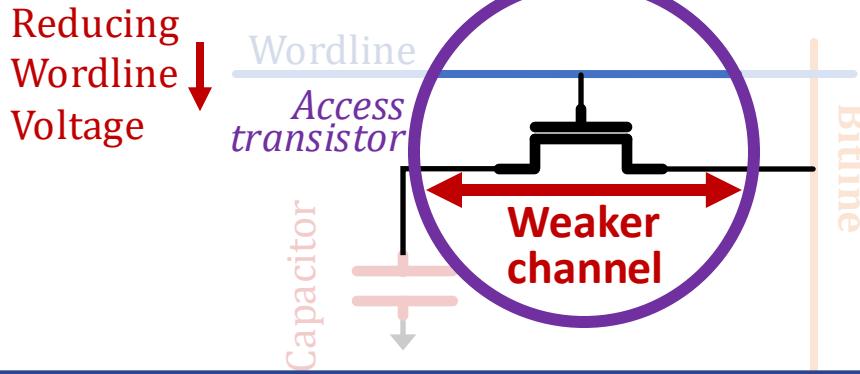
Component	Parameters
DRAM Cell	C: 16.8 fF, R: 698 Ω
Bitline	C: 100.5 fF, R: 6980 Ω
Cell Access NMOS	W: 55 nm, L: 85 nm
Sense Amp. NMOS	W: 1.3 μm , L: 0.1 μm
Sense Amp. PMOS	W: 0.9 μm , L: 0.1 μm



A Closer Look into Row Activation Latency



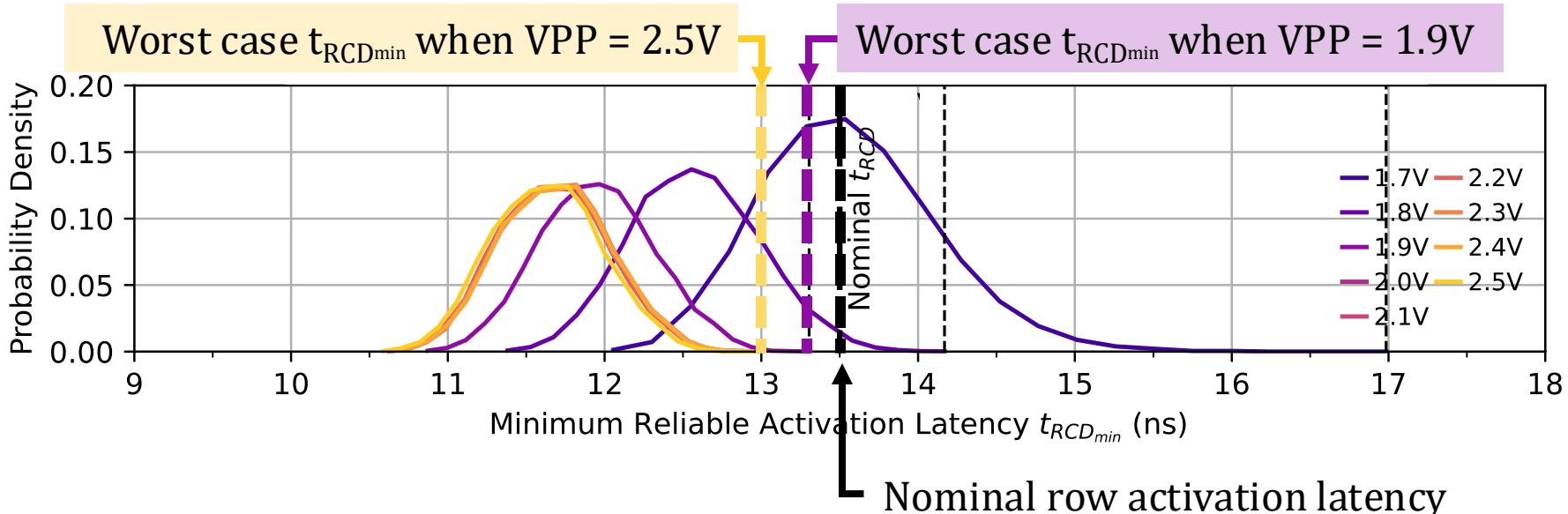
- Row activation completes when the **bitline voltage** reaches a threshold (V_{TH})
- Reduced **wordline voltage** leads to a **weaker channel** in the access transistor



OBSERVATION 8

Row activation latency **increases** with **reduced** wordline voltage

Variation in Row Activation Latency



When wordline voltage is reduced from 2.5V to 1.9V:

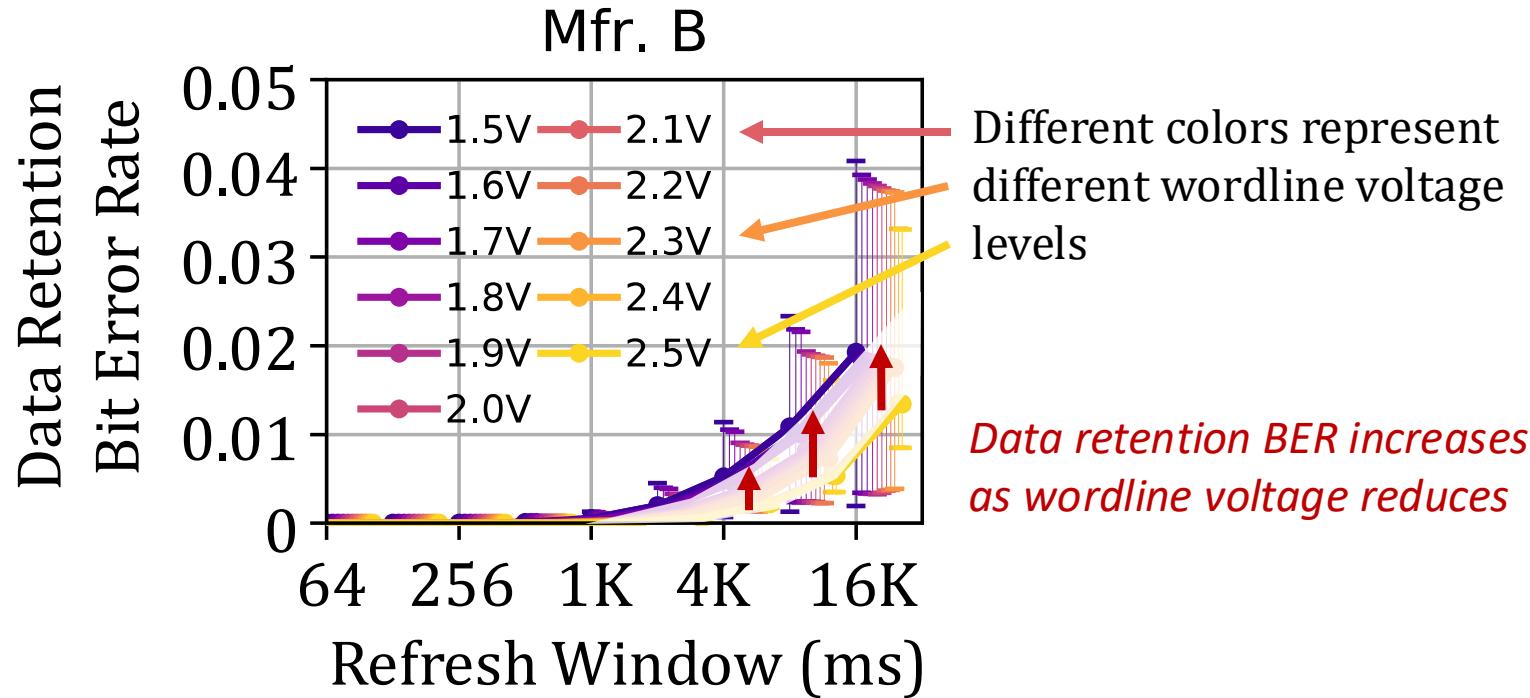
- The worst-case row activation latency is still lower than nominal value
- The guardband reduces from **4.4%** to **1.5%** as the worst-case latency increases from **12.9ns** to **13.3ns**

OBSERVATION 9

SPICE simulation results **agree with** our observations based on experiments **on real chips**

The SPICE simulation results are *not identical* with real chip observations because the SPICE model *cannot* simulate a real DRAM chip's **exact behavior** without **proprietary** design and manufacturing information

Wordline Voltage's Effect on DRAM Refresh

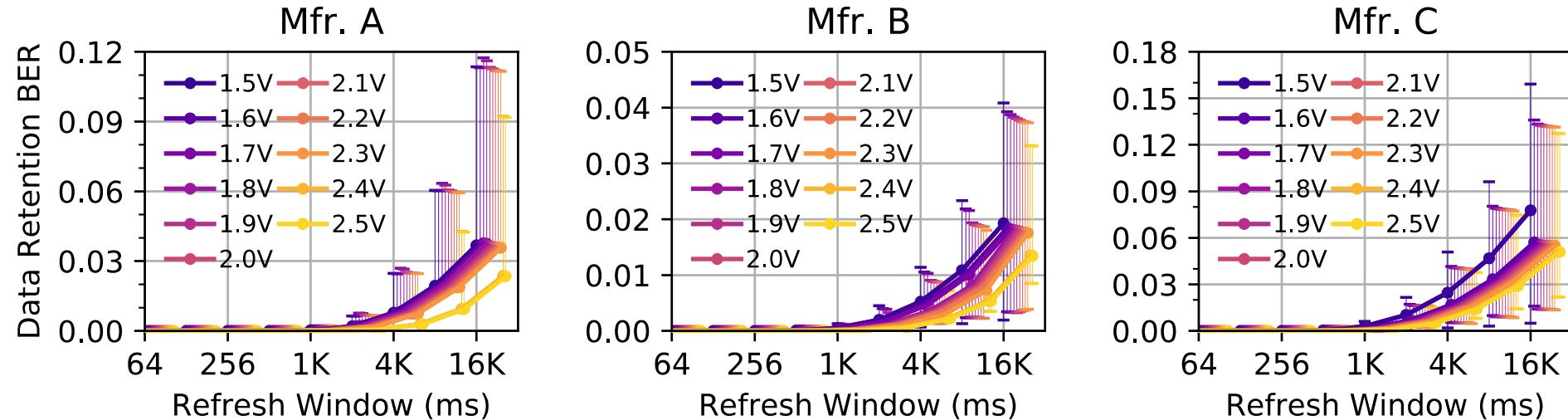


More DRAM cells tend to experience **data retention bit flips** when **wordline voltage is reduced**

OBSERVATION 13

216 out of 272 DRAM chips **reliably operate** using **nominal refresh rate** due to the **built-in safety margins** (guardbands)

Wordline Voltage's Effect on DRAM Refresh



OBSERVATION 12

More DRAM cells tend to experience **data retention bit flips** when **wordline voltage is reduced**

OBSERVATION 13

216 out of 272 DRAM chips **reliably operate** using **nominal refresh rate** due to the **built-in safety margins** (guardbands)

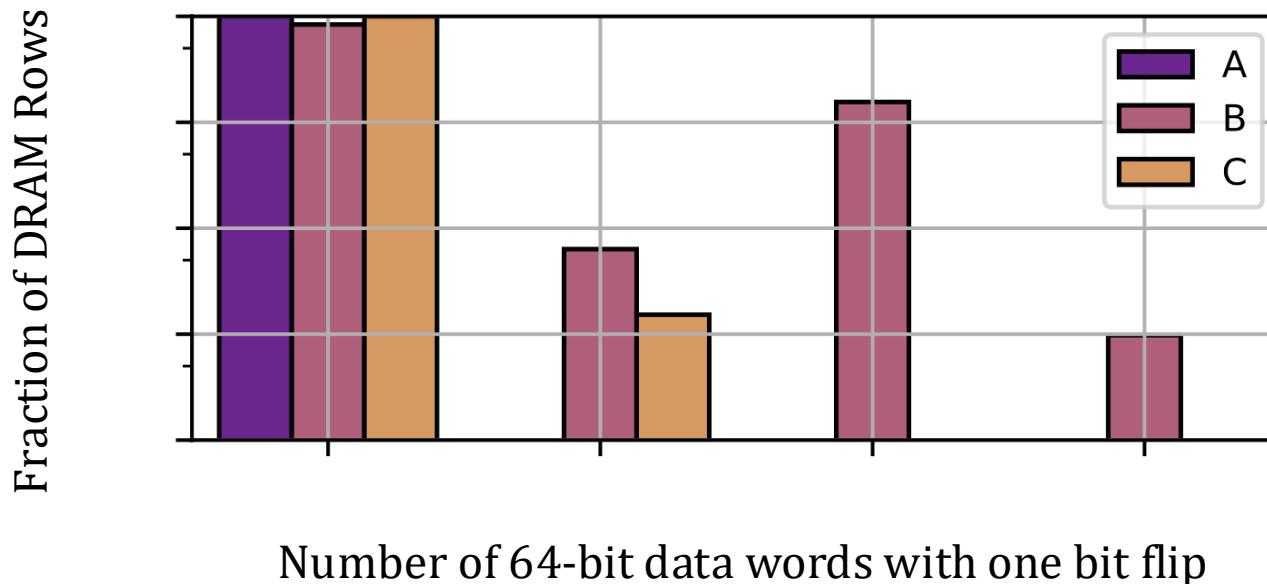
Spatial Distribution of Data Retention Bit Flips

- There are **no 64-bit words** with **more than one bit flip**

OBSERVATION 14

Data retention errors **can be avoided** using **single error correcting codes** at the smallest refresh window that yields *non-zero* bit error rate

- A **small fraction** of DRAM rows contain **erroneous words**



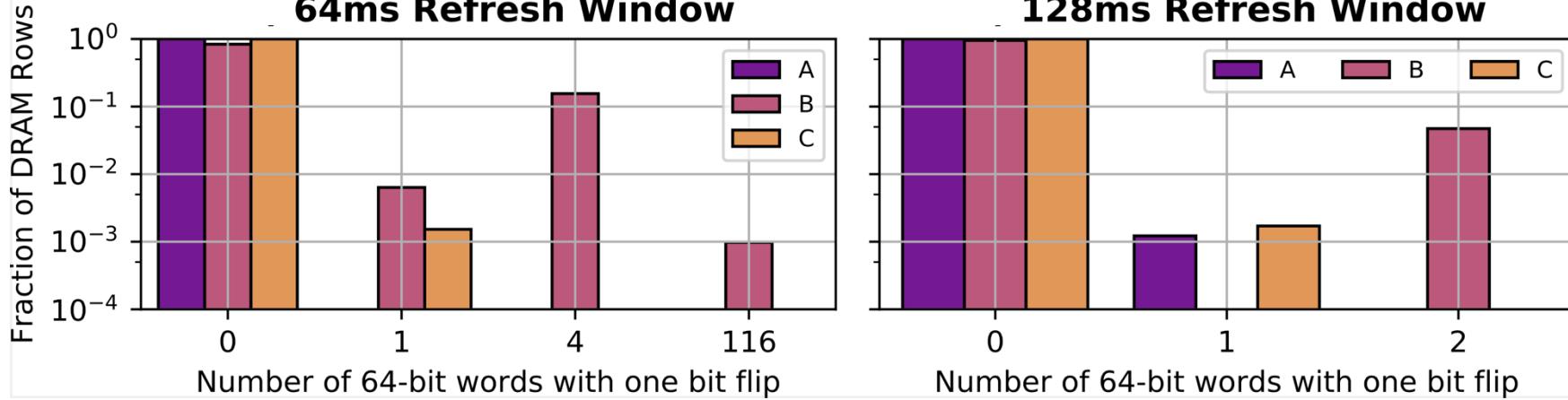
Spatial Distribution of Data Retention Bit Flips

- There are **no 64-bit words** with **more than one bit flip**

OBSERVATION 14

Data retention errors **can be avoided** using **single error correcting codes** at the smallest refresh window that yields *non-zero* bit error rate

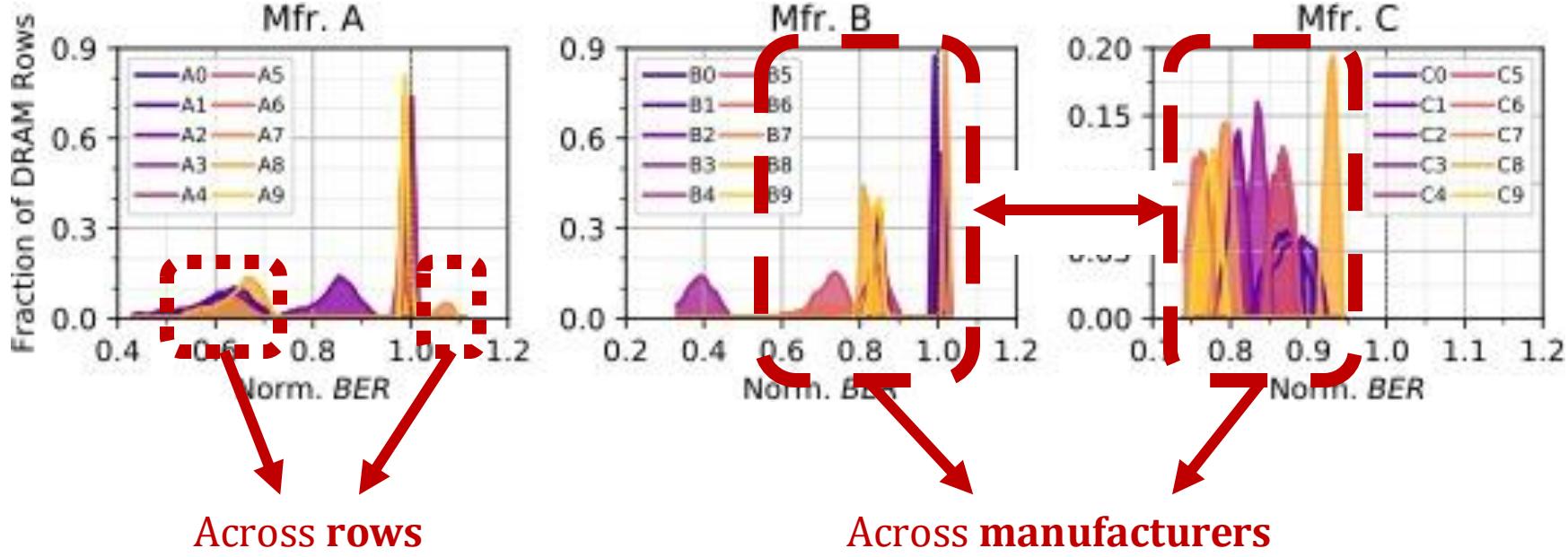
- A **small fraction** of DRAM rows contain **erroneous words**



OBSERVATION 15

Only a **small fraction** (**16.4%/5.0%**) of DRAM rows have **erroneous words** at the smallest refresh rate (64ms/128ms) that yields *non-zero* bit error rate

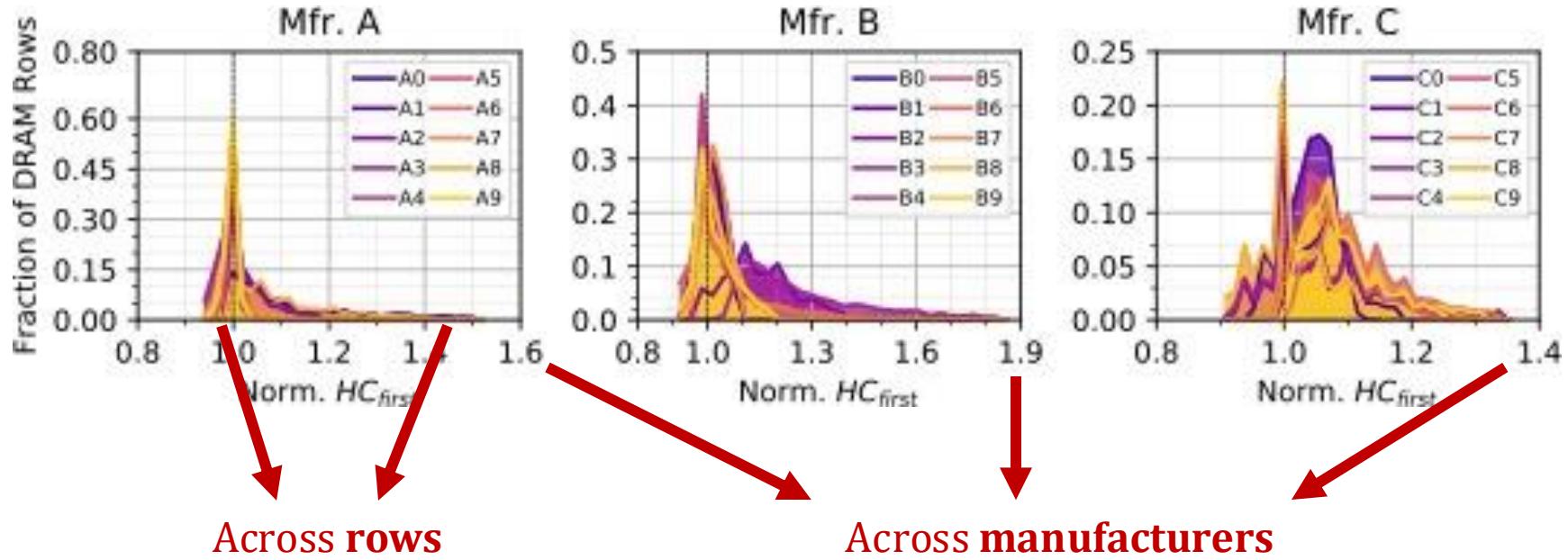
Distribution of Bit Flips across DRAM Rows



OBSERVATION 3

BER reduction with reduced wordline voltage varies across different **DRAM rows** and **manufacturers**

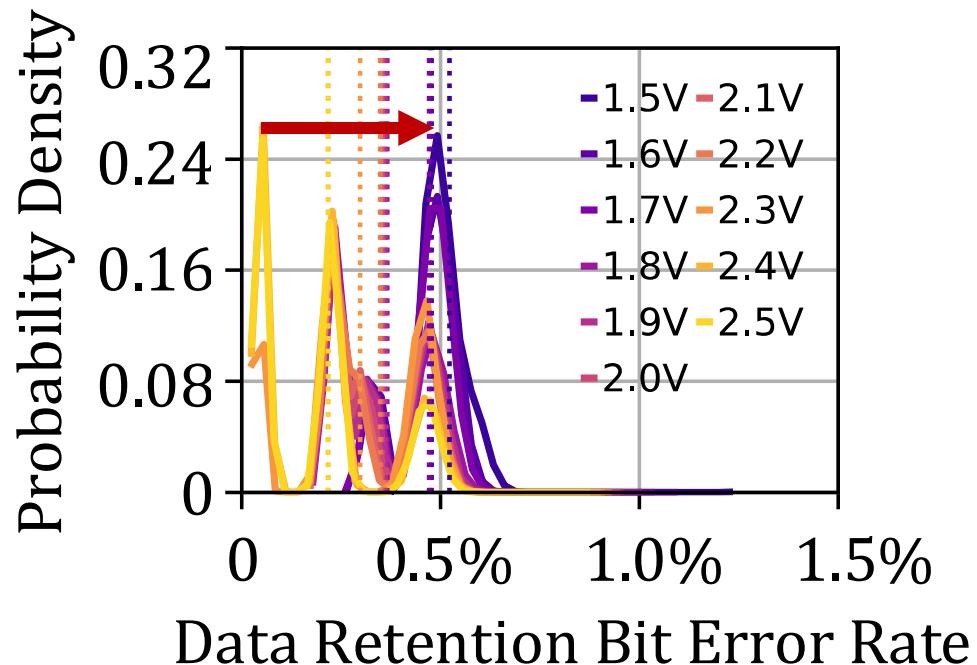
Distribution of HC_{first} across DRAM Rows



OBSERVATION 3

HC_{first} reduction with reduced wordline voltage varies across different **DRAM rows** and **manufacturers**

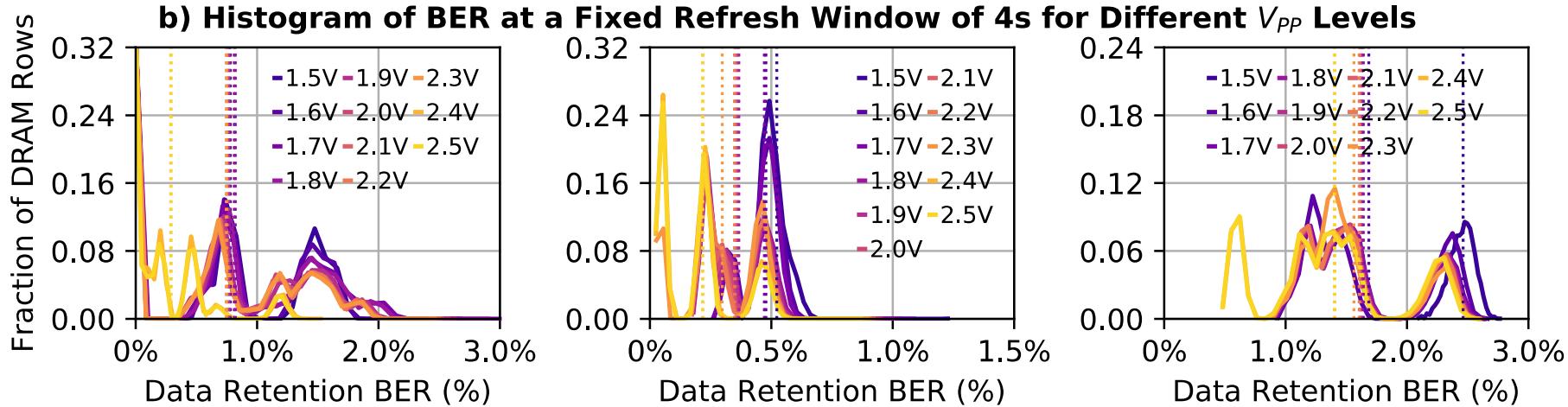
Wordline Voltage's Effect on DRAM Refresh



OBSERVATION 12

More DRAM cells tend to experience **data retention bit flips** when **wordline voltage is reduced**

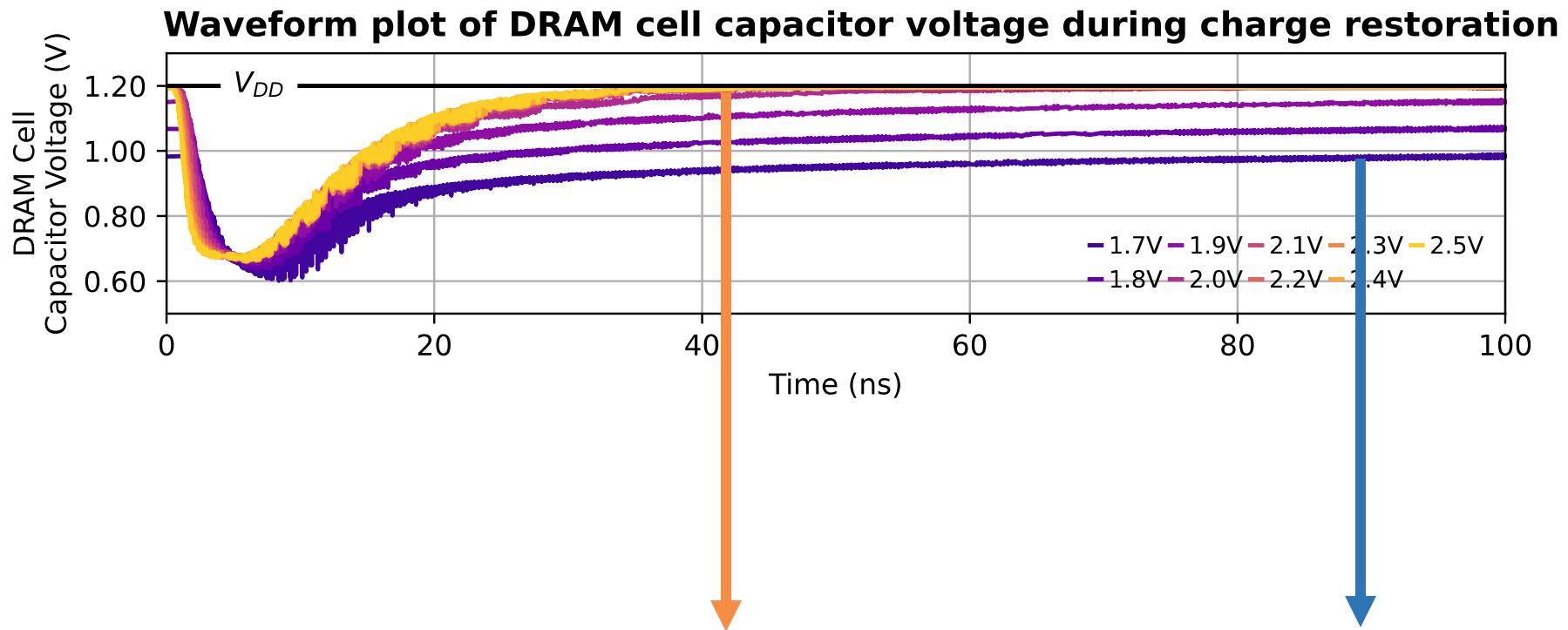
Wordline Voltage's Effect on DRAM Refresh



OBSERVATION 12

More DRAM cells tend to experience **data retention bit flips** when **wordline voltage is reduced**

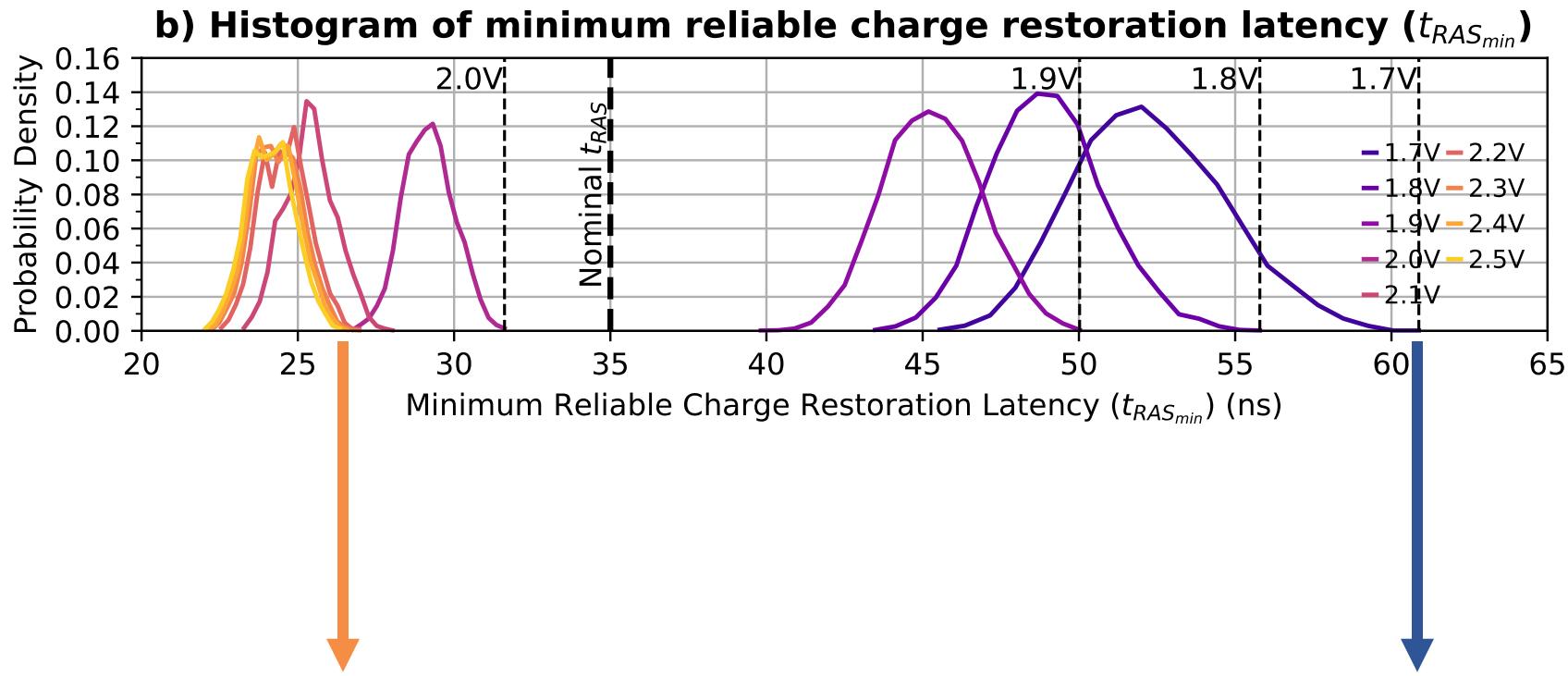
Charge Restoration Process



OBSERVATION 10

A DRAM cell's capacitor voltage **can saturate** at a **lower voltage** level when wordline voltage is reduced

Also in the Paper



OBSERVATION 11

A DRAM cell's **charge restoration latency** ($t_{RAS_{min}}$)
can increase with **reduced wordline voltage**

DRAM Chips Tested

Table 3: Tested DRAM modules and their characteristics when $V_{PP}=2.5$ V (nominal) and $V_{PP}=V_{PPmin}$. V_{PPmin} is specified for each module.

DRAM Chip Mfr.	DIMM Name	DIMM Model	Die Density	Frequency (MT/s)	Chip Org.	Die Revision	Mfr. Date	Minimum HC _{first}	V _{PP} = 2.5V		V _{PP} = V _{PPmin}		Recommended V _{PP} (V _{PPRec})	V _{PP} = V _{PPRec}	
Mfr. A (Micron)	A0	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	39.8K	1.24e-03	1.4	42.2K	1.00e-03	1.4	42.2K	1.00e-03
Mfr. A (Micron)	A1	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	42.2K	9.90e-04	1.4	46.4K	7.83e-04	1.4	46.4K	7.83e-04
Mfr. A (Micron)	A2	MTA18ASF2G72PZ-2G3B1QK [148]	8Gb	2400	x4	B	11-19	41.0K	1.24e-03	1.7	39.8K	1.35e-03	2.1	42.1K	1.55e-3
Mfr. A (Micron)	A3	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	16.7K	3.33e-02	1.4	16.5K	3.52e-02	1.7	17.0K	3.48e-02
Mfr. A (Micron)	A4	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	14.4K	3.18e-02	1.5	14.4K	3.33e-02	2.5	14.4K	3.18e-02
Mfr. A (Micron)	A5	CT4G4SFS8213.C8FBD1	4Gb	2400	x8	-	48-16	140.7K	1.39e-06	2.4	145.4K	3.39e-06	2.4	145.4K	3.39e-06
Mfr. A (Micron)	A6	CT4G4DFS8266.C8FF [149]	4Gb	2666	x8	F	07-21	16.5K	3.50e-02	1.5	16.5K	3.66e-02	2.5	16.5K	3.50e-02
Mfr. A (Micron)	A7	CMV4GX4M1A2133C15 [150]	4Gb	2133	x8	-	-	16.5K	3.42e-02	1.8	16.5K	3.52e-02	2.5	16.5K	3.42e-02
Mfr. A (Micron)	A8	MTA18ASF2G72PZ-2G3B1QG [148]	8Gb	2400	x4	B	11-19	35.2K	2.38e-03	1.4	39.8K	2.07e-03	1.4	39.8K	2.07e-03
Mfr. A (Micron)	A9	CMV4GX4M1A2133C15 [150]	4Gb	2133	x8	-	-	14.3K	3.33e-02	1.5	14.3K	3.48e-02	1.6	14.6K	3.47e-02
Mfr. B (Samsung)	B0	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.9K	1.18e-01	2.0	7.6K	1.22e-01	2.5	7.9K	1.18e-01
Mfr. B (Samsung)	B1	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.3K	1.26e-01	2.0	7.6K	1.28e-01	2.0	7.6K	1.28e-01
Mfr. B (Samsung)	B2	F4-2400C17S-8GNT [152]	4Gb	2400	x8	F	02-21	11.2K	2.52e-02	1.6	12.0K	2.22e-02	1.6	12.0K	2.22e-02
Mfr. B (Samsung)	B3	M393A1K43BB1-CTD6Y [153]	8Gb	2666	x8	B	52-20	16.6K	2.73e-03	1.6	21.1K	1.09e-03	1.6	21.1K	1.09e-03
Mfr. B (Samsung)	B4	M393A1K43BB1-CTD6Y [153]	8Gb	2666	x8	B	52-20	21.0K	2.95e-03	1.8	19.9K	2.52e-03	2.0	21.1K	2.68e-03
Mfr. B (Samsung)	B5	M471A5143EB0-CPB [154]	4Gb	2133	x8	E	08-17	21.0K	7.78e-03	1.8	21.0K	6.02e-03	2.0	21.1K	8.67e-03
Mfr. B (Samsung)	B6	CMK16GX4M2B3200C16 [155]	8Gb	3200	x8	-	-	10.3K	1.14e-02	1.7	10.5K	9.82e-03	1.7	10.5K	9.82e-03
Mfr. B (Samsung)	B7	M378A1K43DB2-CTD [151]	8Gb	2666	x8	D	10-21	7.3K	1.32e-01	2.0	7.6K	1.33e-01	2.0	7.6K	1.33e-01
Mfr. B (Samsung)	B8	CMK16GX4M2B3200C16 [155]	8Gb	3200	x8	-	-	11.6K	2.88e-02	1.7	10.5K	2.37e-02	1.8	11.7K	2.58e-02
Mfr. B (Samsung)	B9	M471A5244CB0-CRC [156]	8Gb	2133	x8	C	19-19	11.8K	2.68e-02	1.7	8.8K	2.39e-02	1.8	12.3K	2.54e-02
Mfr. C (SK Hynix)	C0	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	19.3K	7.29e-03	1.7	23.4K	6.61e-03	1.7	23.4K	6.61e-03
Mfr. C (SK Hynix)	C1	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	19.3K	6.31e-03	1.7	20.6K	5.90e-03	1.7	20.6K	5.90e-03
Mfr. C (SK Hynix)	C2	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	9.6K	2.82e-02	1.5	9.2K	2.34e-02	2.3	10.0K	2.89e-02
Mfr. C (SK Hynix)	C3	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	9.3K	2.57e-02	1.5	8.9K	2.21e-02	2.3	9.7K	2.66e-02
Mfr. C (SK Hynix)	C4	HMAA4GU6AJR8N-XN [158]	16Gb	3200	x8	A	51-20	11.6K	3.22e-02	1.5	11.7K	2.88e-02	1.5	11.7K	2.88e-02
Mfr. C (SK Hynix)	C5	HMAA4GU6AJR8N-XN [158]	16Gb	3200	x8	A	51-20	9.4K	3.28e-02	1.5	12.7K	2.85e-02	1.5	12.7K	2.85e-02
Mfr. C (SK Hynix)	C6	CMV4GX4M1A2133C15 [150]	4Gb	2133	x8	C	-	14.2K	3.08e-02	1.6	15.5K	2.25e-02	1.6	15.5K	2.25e-02
Mfr. C (SK Hynix)	C7	CMV4GX4M1A2133C15 [150]	4Gb	2133	x8	C	-	11.7K	3.24e-02	1.6	13.6K	2.60e-02	1.6	13.6K	2.60e-02
Mfr. C (SK Hynix)	C8	KSM32RD8/16HDR [157]	8Gb	3200	x8	D	48-20	11.4K	2.69e-02	1.6	9.5K	2.57e-02	2.5	11.4K	2.69e-02
Mfr. C (SK Hynix)	C9	F4-2400C17S-8GNT [152]	4Gb	2400	x8	B	02-21	12.6K	2.18e-02	1.7	15.2K	1.63e-02	1.7	15.2K	1.63e-02

RowHammer Test

Alg. 1: Test for HC_{first} and BER for a Given V_{PP}

```
// RAvictim: victim row address
// WCDP: worst-case data pattern
// HC: number of activations per aggressor row
Function measure_BER(RAvictim, WCDP, HC):
    initialize_row(RAvictim, WCDP)
    initialize_aggressor_rows(RAvictim, bitwise_inverse(WCDP))
    hammer_doublesided(RAvictim, HC)
    BERrow = compare_data(RAvictim, WCDP)
    return BERrow

// Vpp: wordline voltage for the experiment
// WCDP_list: the list of WCDPs (one WCDP per row)
// row_list: the list of tested rows
Function test_loop(Vpp, WCDP_list):
    set_vpp(Vpp)
    foreach RAvictim in row_list do
        HC = 300K // initial hammer count to test
        HCstep = 150K // how much to increment/decrement HC
        while HCstep > 100 do
            BERrowmax = 0
            for i ← 0 to num_iterations do
                BERrow = measure_BER(RAvictim, WCDP, HC)
                record_BER(Vpp, RAvictim, WCDP, HC, BERrow, i)
                BERrowmax = max(BERrowmax, BERrow)
            end
            if BERrowmax == 0 then
                | HC+ = HCstep // Increase HC if no bit flips occur
            end
            else
                | HC- = HCstep // Reduce HC if a bit flip occurs
            end
            HCstep = HCstep/2
        end
        record_HCfirst(Vpp, RAvictim, WCDP, HC)
    end
```

Row Activation and Refresh Rate Tests

Alg. 2: Test for Row Activation Latency for a Given V_{PP}

```
//  $V_{pp}$ : wordline voltage for the experiment
// WCDP_list: the list of WCDPs (one WCDP per row)
// row_list: the list of tested rows
Function test_loop( $V_{pp}$ , WCDP_list, row_list):
    set_vpp( $V_{pp}$ )
    foreach RA in row_list do
         $t_{RCD} = 13.5\text{ ns}$ 
        found_faulty, found_reliable = False, False
        while not found_faulty or not found_reliable do
            is_faulty = False
            for i  $\leftarrow 0$  to num_iterations do
                foreach column C in row RA do
                    initialize_row(RA, WCDP_list[RA])
                    activate_row(RA,  $t_{RCD}$ ) //activate the row using  $t_{RCD}$ 
                    read_data = read_col(C)
                    close_row(RA)
                    BER_col = compare(WCDP_list[RA], read_data)
                    if  $BER_{col} > 0$  then is_faulty=True
                end
            end
            if is_faulty then { $t_{RCD} += 1.5\text{ ns}$ ; found_faulty = True;}
            else { $t_{RCDmin} = t_{RCD}$ ;  $t_{RCD} -= 1.5\text{ ns}$ ; found_reliable = True;}
        end
        record_tRCDmin(RA,  $t_{RCDmin}$ )
    end
```

Alg. 3: Test for Data Retention Times for a Given V_{PP}

```
//  $V_{pp}$ : wordline voltage for the experiment
// WCDP_list: the list of WCDPs (one WCDP per row)
// row_list: the list of tested rows
Function test_loop( $V_{pp}$ , WCDP_list, row_list):
    set_vpp( $V_{pp}$ )
     $t_{REFW} = 16\text{ ms}$ 
    while  $t_{REFW} \leq 16\text{ s}$  do
        for i  $\leftarrow 0$  to num_iterations do
            foreach RA in row_list do
                initialize_row(RA, WCDP_list[RA])
                wait( $t_{REFW}$ )
                read_data = read_row(RA)
                BER_row = compare_data(WCDP_list[RA], read_data)
                record_retention_errors(RA,  $t_{REFW}$ , BER_row)
            end
        end
         $t_{REFW} = t_{REFW} \times 2$ 
    end
```

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Abdullah Giray Yağlıkçı

Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

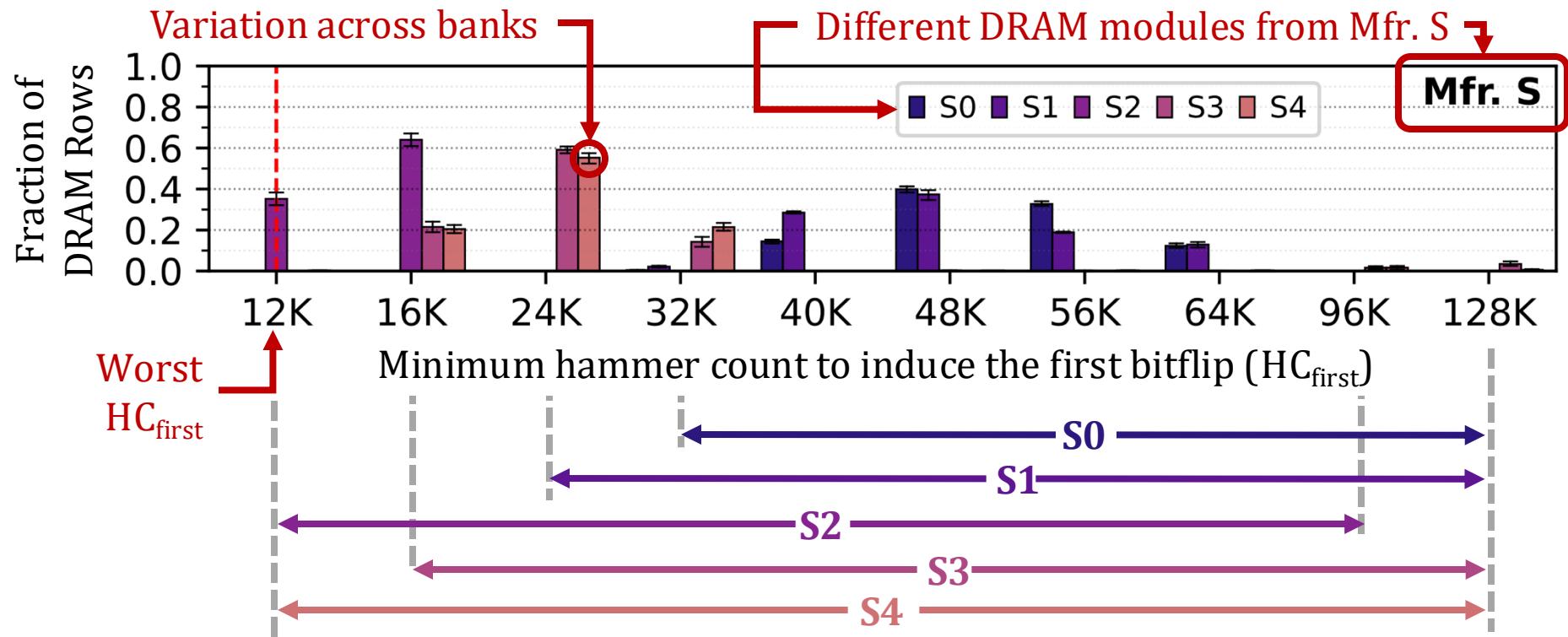
ETH zürich

Tested DRAM Chips

144 DRAM chips from SK Hynix, Micron, and Samsung

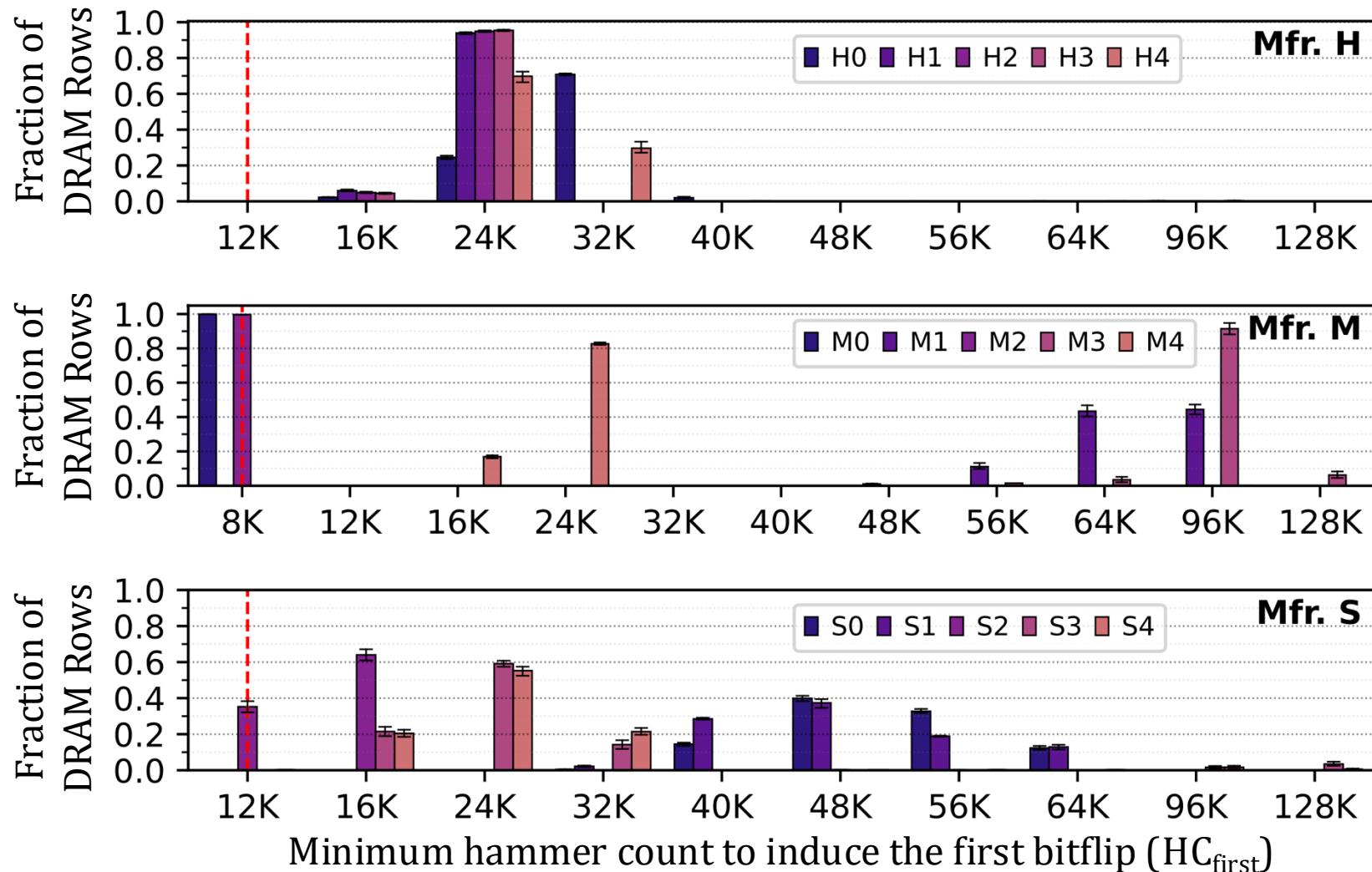
Mfr.	DIMM ID	# of Chips	Density Die Rev.	Chip Org.	Date (ww-yy)
Mfr. H (SK Hynix)	H0	8	16Gb – A	x8	51-20
	H1, H2, H3	3 × 8	16Gb – C	x8	48-20
	H4	8	8Gb – D	x8	48-20
Mfr. M (Micron)	M0	4	16Gb – E	x16	46-20
	M1, M3	2 × 16	8Gb – B	x4	N/A
	M2	16	16Gb – E	x4	14-20
	M4	4	16Gb – B	x16	26-21
Mfr. S (Samsung)	S0, S1	2 × 8	8Gb – B	x8	52-20
	S2	8	8Gb – B	x8	10-21
	S3	8	4Gb – F	x8	N/A
	S4	16	8Gb – C	x4	35-21

Spatial Variation in the Minimum Hammer Count to Induce the First Bitflip across DRAM Rows

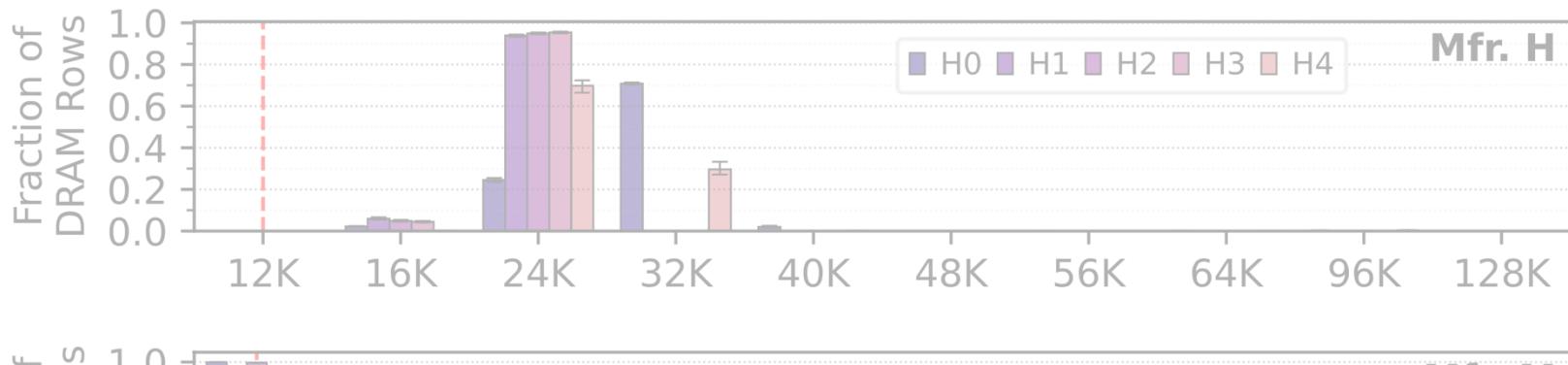


The minimum hammer count to induce the first bitflip
significantly varies across rows in a DRAM bank

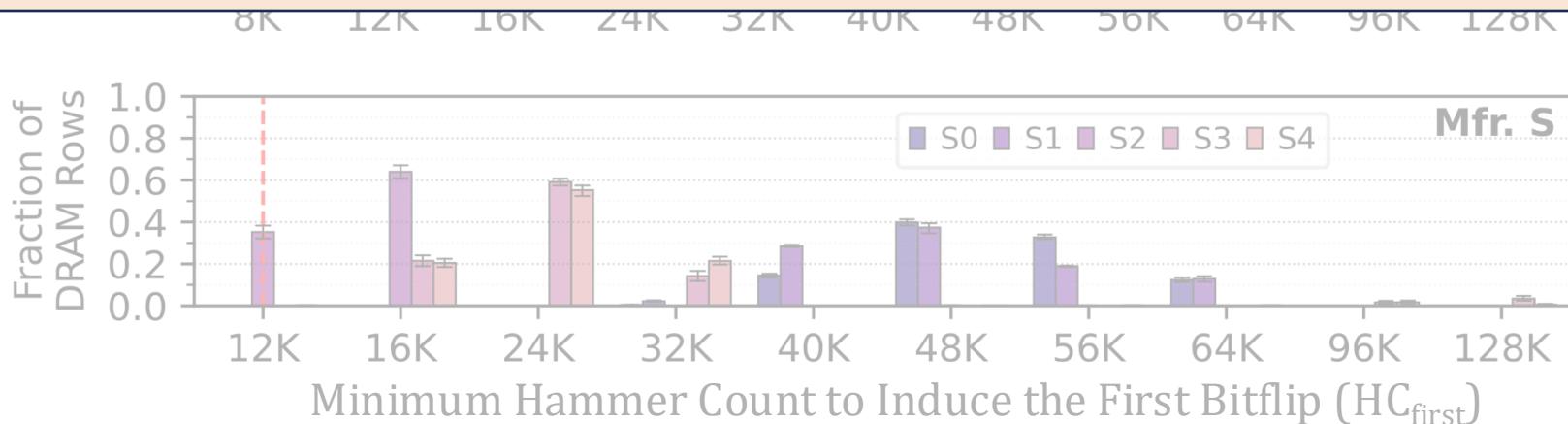
Spatial Variation in the Minimum Hammer Count to Induce the First Bitflip across DRAM Rows



Spatial Variation in the Minimum Hammer Count to Induce the First Bitflip across DRAM Rows



The minimum hammer count to induce the first bitflip **significantly varies across rows** in a DRAM bank



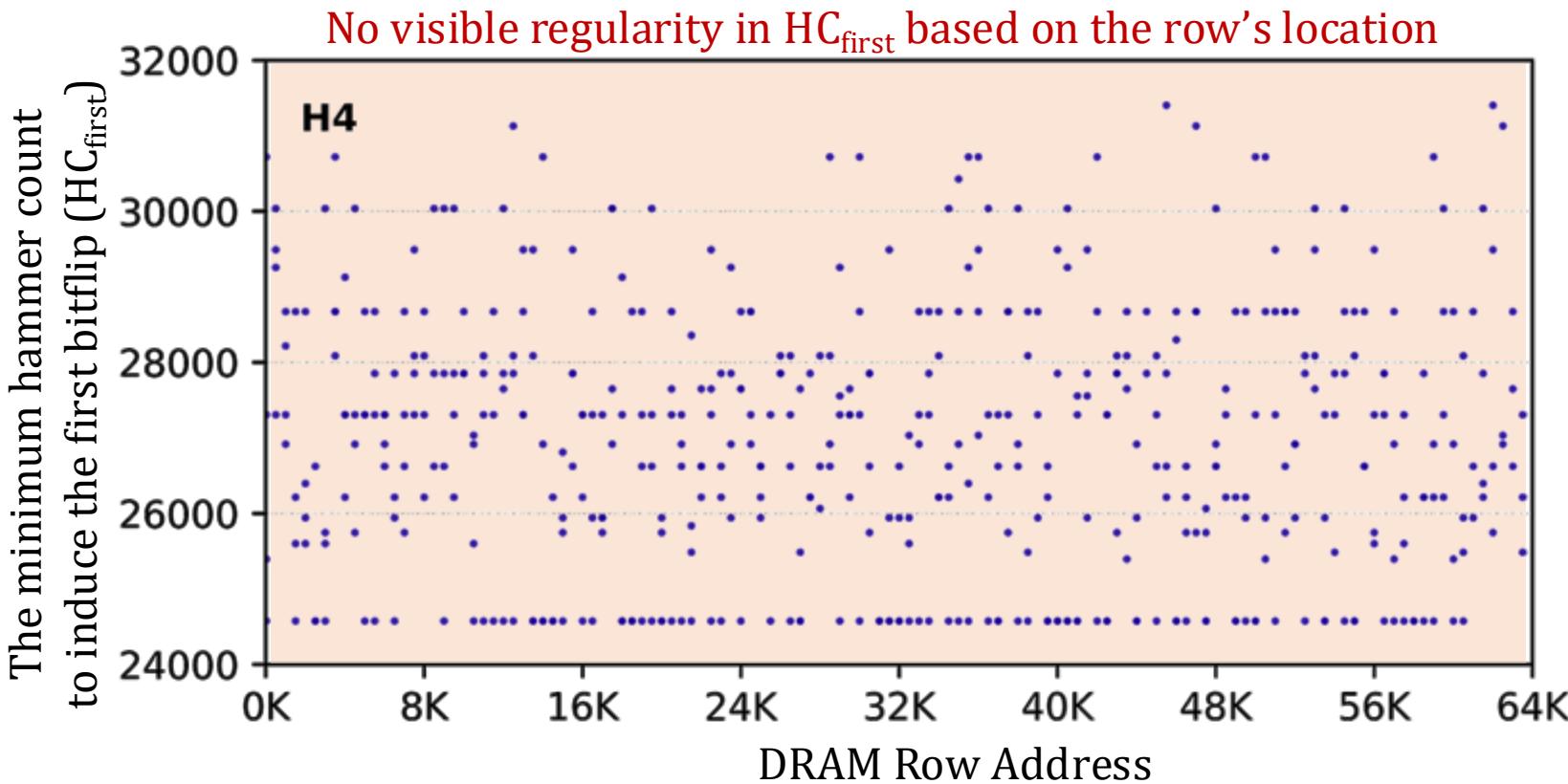
More Detailed Information in the Extended Version

Table 5: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier Chip Identifier	Freq (MT/s)	Mfr. Date ww-yy	Chip Den.	Die Rev.	Chip Org.	Num. of Rows per Bank	HC_{first}	Min.	Avg.	Max.
H0	SK Hynix	HMAA4GU6AJR8N-XN [287] H5ANAG8NAJR-XN [288]	3200	51-20	16Gb	A	×8	128K	16K	46.2K	96K	
H1		HMAA4GU7CJR8N-XN [289] H5ANAG8NCJR-XN [231]	3200	51-20	16Gb	C	×8	128K	12K	54.0K	128K	
H2		HMAA4GU7CJR8N-XN [289] H5ANAG8NCJR-XN [231]	3200	36-21	16Gb	C	×8	128K	12K	55.4K	128K	
H3		HMAA4GU7CJR8N-XN [289] H5ANAG8NCJR-XN [231]	3200	36-21	16Gb	C	×8	128K	12K	57.8K	128K	
H4		KSM32RD8/16HDR [290] H5AN8G8NDJR-XNC [232]	3200	48-20	8Gb	D	×8	64K	16K	38.1K	96K	
M0	Micron	MTA4ATF1G64HZ-3G2E1 [233] MT40A1G16KD-062E [234]	3200	46-20	16Gb	E	×16	128K	8K	24.5K	40K	
M1		MTA18ASF2G72PZ-2G3B1QK [235] MT40A2G4WE-083E:B [291]	2400	N/A	8Gb	B	×4	128K	40K	64.5K	96K	
M2		MTA36ASF8G72PZ-2G9E1TI [236] MT40A4G4JC-062E:E [292]	2933	14-20	16Gb	E	×4	128K	8K	28.6K	48K	
M3		MTA18ASF2G72PZ-2G3B1QK [235] MT40A2G4WE-083E:B [291]	2400	36-21	8Gb	B	×4	128K	56K	90.0K	128K	
M4		MTA4ATF1G64HZ-3G2B2 [237] MT40A1G16RC-062E:B [293]	3200	26-21	16Gb	B	×16	128K	12K	42.2K	96K	
S0	Samsung	M393A1K43BB1-CTD [294] K4A8G085WB-BCTD [230]	2666	52-20	8Gb	B	×8	64K	32K	57.0K	128K	
S1		M393A1K43BB1-CTD [294] K4A8G085WB-BCTD [230]	2666	52-20	8Gb	B	×8	64K	24K	59.8K	128K	
S2		M393A1K43BB1-CTD [294] K4A8G085WB-BCTD [230]	2666	10-21	8Gb	B	×8	64K	12K	42.7K	96K	
S3		F4-2400C17S-8GNT [295] K4A4G085WF-BCTD [296]	2400	04-21	4Gb	F	×8	32K	16K	59.2K	128K	
S4		M393A2K40CB2-CTD [229] K4A8G045WC-BCTD [297]	2666	35-21	8Gb	C	×4	128K	12K	55.4K	128K	

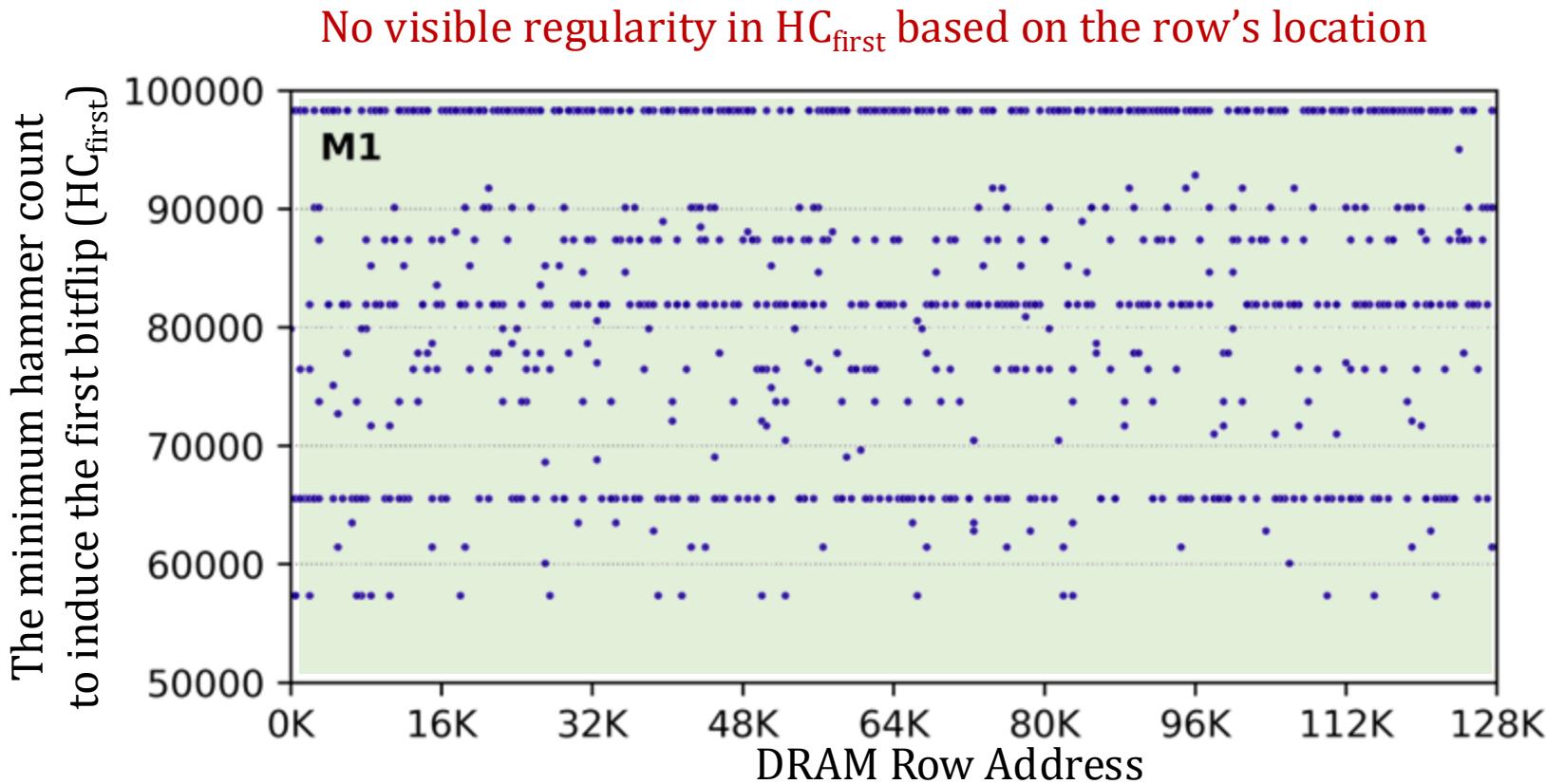
<https://arxiv.org/pdf/2402.18652.pdf>

Regularity in Spatial Variation of Read Disturbance across DRAM Rows



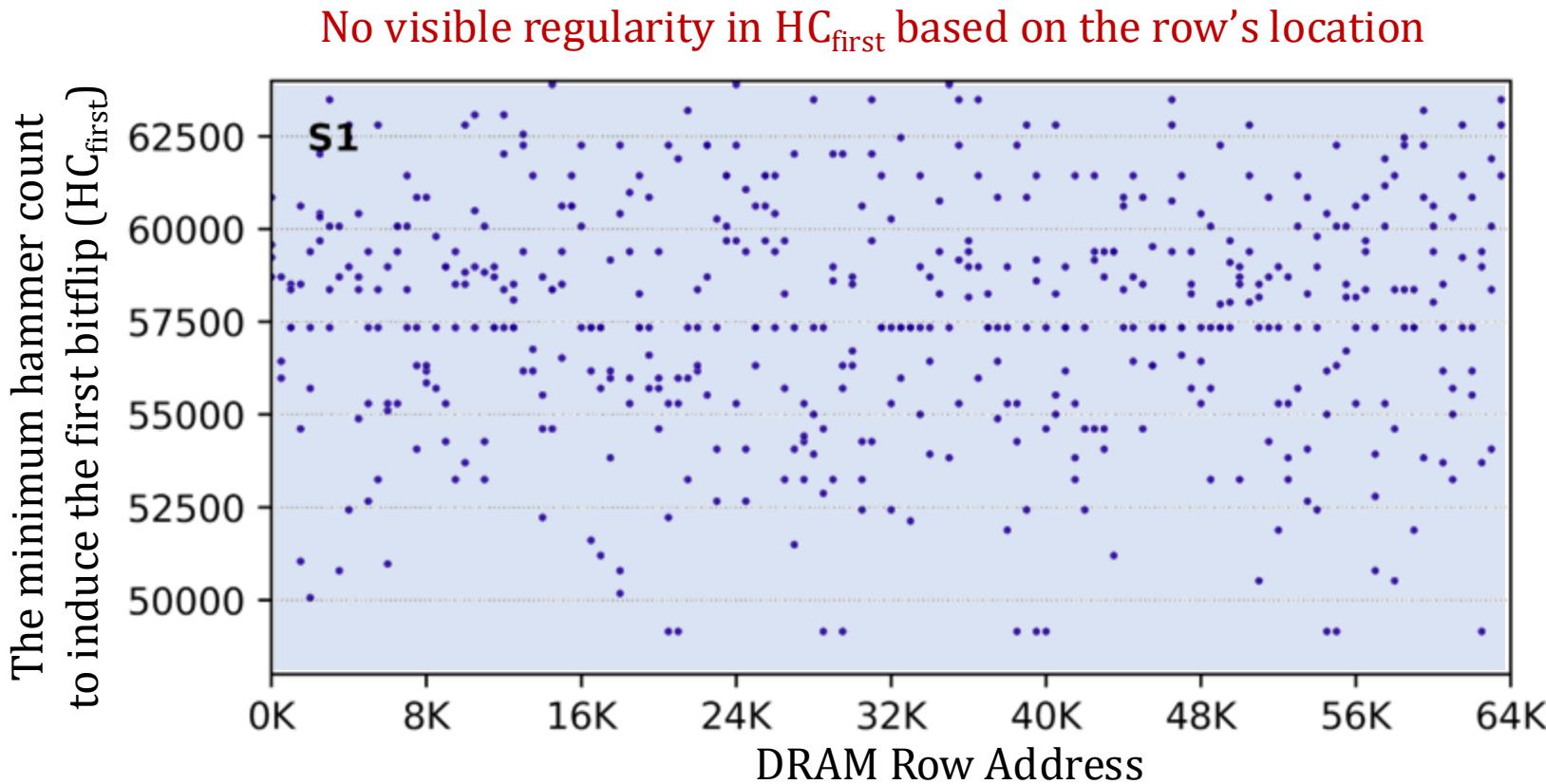
The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

Regularity in Spatial Variation of Read Disturbance across DRAM Rows



The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

Regularity in Spatial Variation of Read Disturbance across DRAM Rows



The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

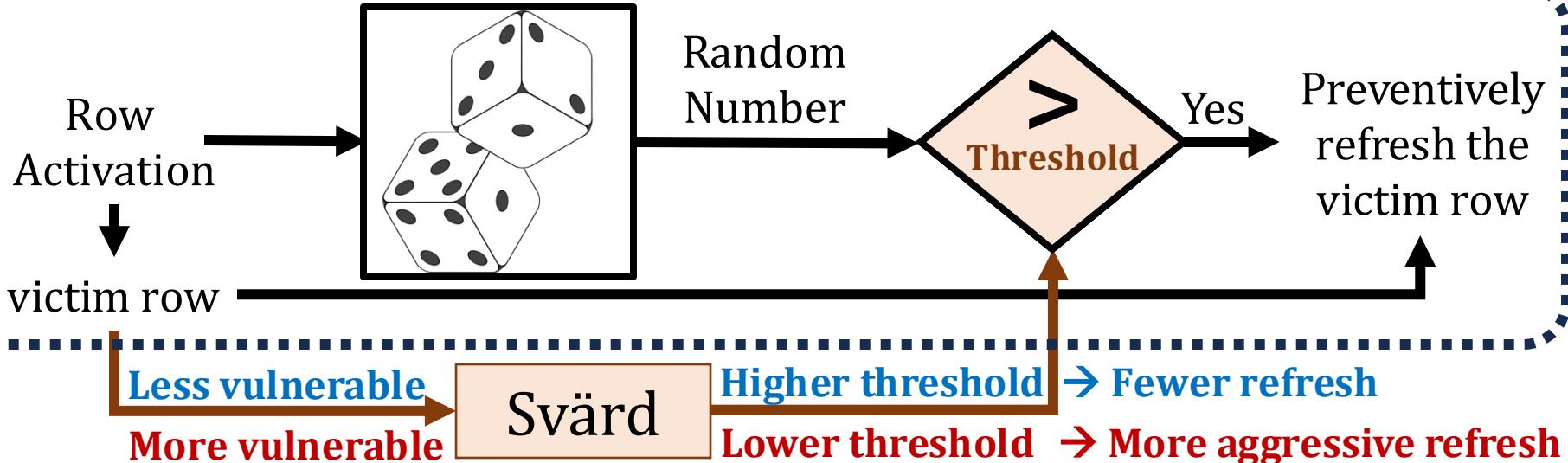
Svärd: Spatial Variation-Aware Read Disturbance Defenses

- **Key Experimental Takeaway:** Read disturbance vulnerability **varies significantly** and **irregularly** across DRAM rows
- **Key Idea:** Leverage the variation in read disturbance vulnerability across DRAM rows
- **Svärd:** Spatial Variation-Aware DRAM Read Disturbance Defenses **Dynamically tunes the aggressiveness** of existing solutions to **the victim row's read disturbance vulnerability**
- Svärd performs **fewer preventive actions (e.g., refresh)** for rows that are **less vulnerable to read disturbance**

Svärd significantly **reduces**
the performance overhead of existing solutions



PARA: Probabilistic Row Activation [Kim+, ISCA'14]



Svärd dynamically tunes PARA's threshold to the victim row's vulnerability

Svärd works with many read disturbance solutions, including:

BlockHammer
[Yaglikci+, HPCA'21]

Hydra
[Qureshi+, ISCA'22]

RRS
[Saileshwar+, ASPLOS'22]

AQUA
[Saxena+, MICRO'22]

- Classifies DRAM rows into **several vulnerability bins**
Maintains **a few (e.g., four) bits** per DRAM row
- Implemented **where the read disturbance solution is**
- Memory controller-based implementation:
Metadata can be maintained in
 - SRAM table in the memory controller
 - Data integrity bits in the DRAM chip
 - ...
- In-DRAM implementation:
Metadata can be maintained in
 - DRAM rows
 - Separate DRAM array
 - ...

Performance Evaluation

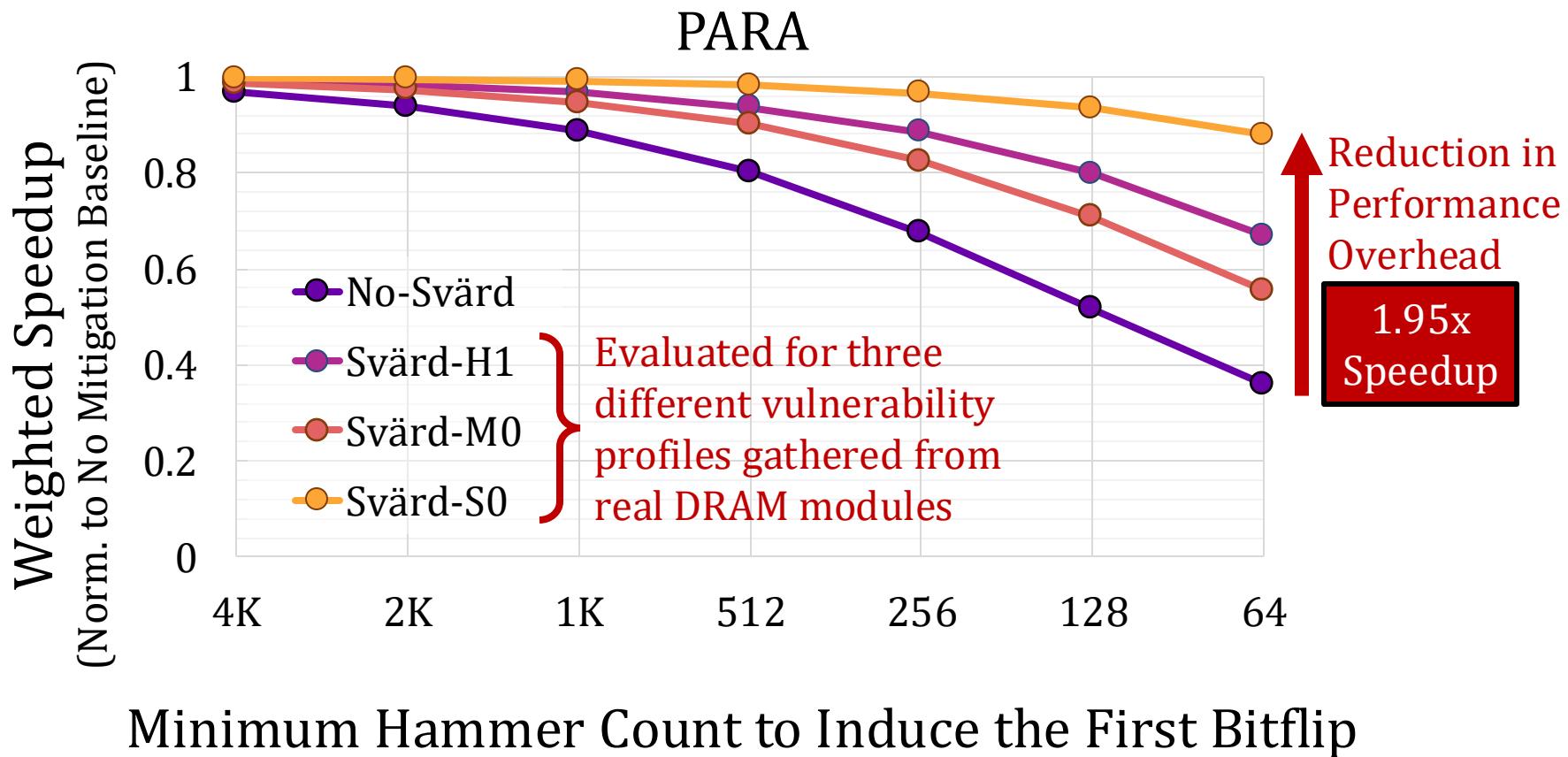
- Cycle-level simulations using **Ramulator 2.0** [Luo+, CAL 2023]
<https://github.com/CMU-SAFARI/ramulator2>

- **System Configuration:**

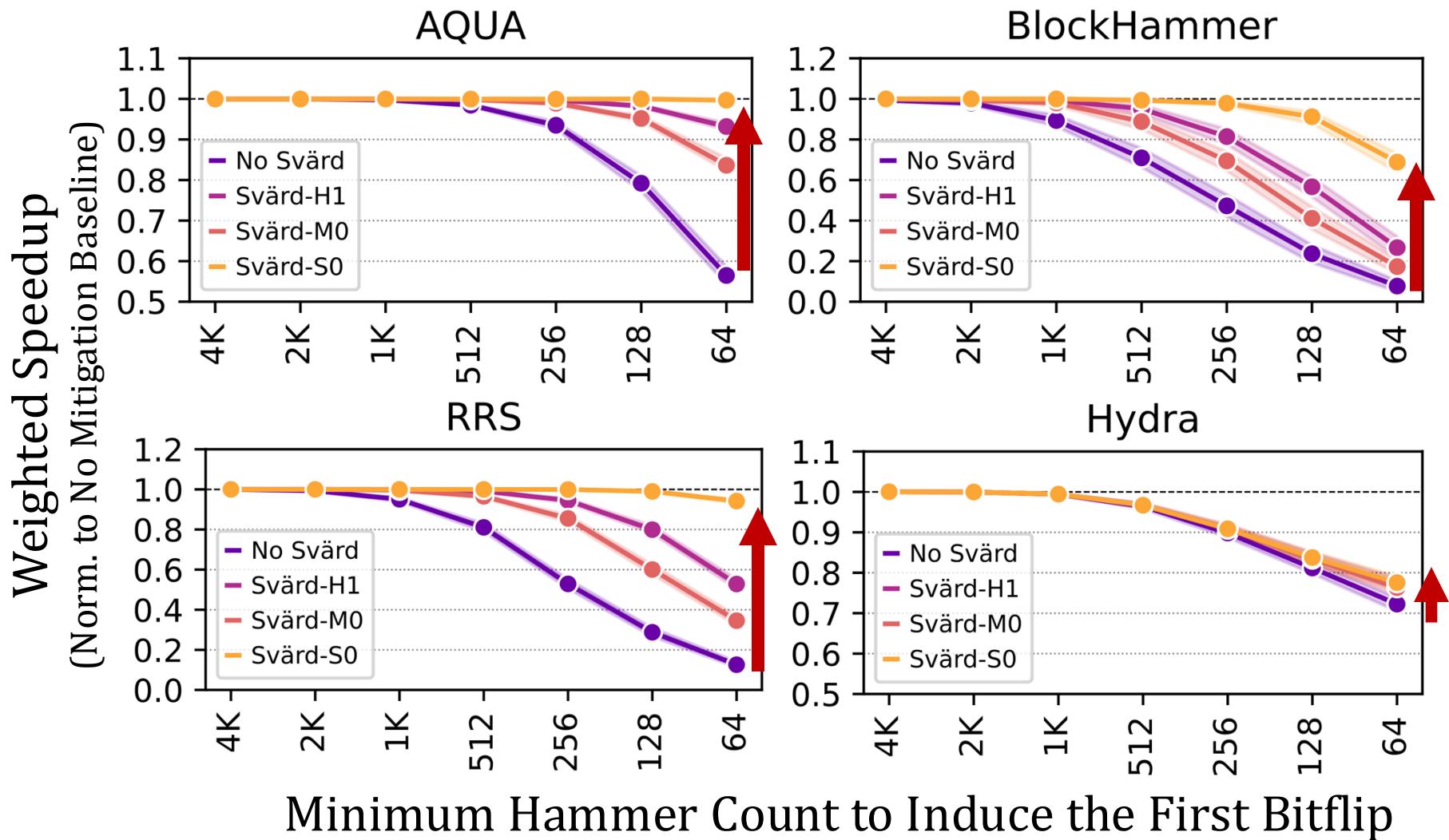
Processor	3.2 GHz, 8 core, 4-wide issue, 128-entry instr. window
Last-Level Cache	64-byte cache line, 8-way set-associative, 8 MB
Memory Scheduler	FR-FCFS
Address Mapping	Minimalistic Open Pages
Main Memory	DDR4, 4 bank group, 4 banks per bank group (16 banks per rank)

- **Workloads:** 120 different 8-core multiprogrammed workloads from **SPEC CPU2006**, **SPEC CPU2017**, **TPC**, **MediaBench**, and **YCSB** benchmark suites
- Integrated with **AQUA**, **BlockHammer**, **PARA**, **Hydra**, and **RRS**
- **HC_{first}:** {4K, 2K, 1K, 512, 256, 128, 64} hammers
The **minimum hammer count** needed to induce **the first bitflip**

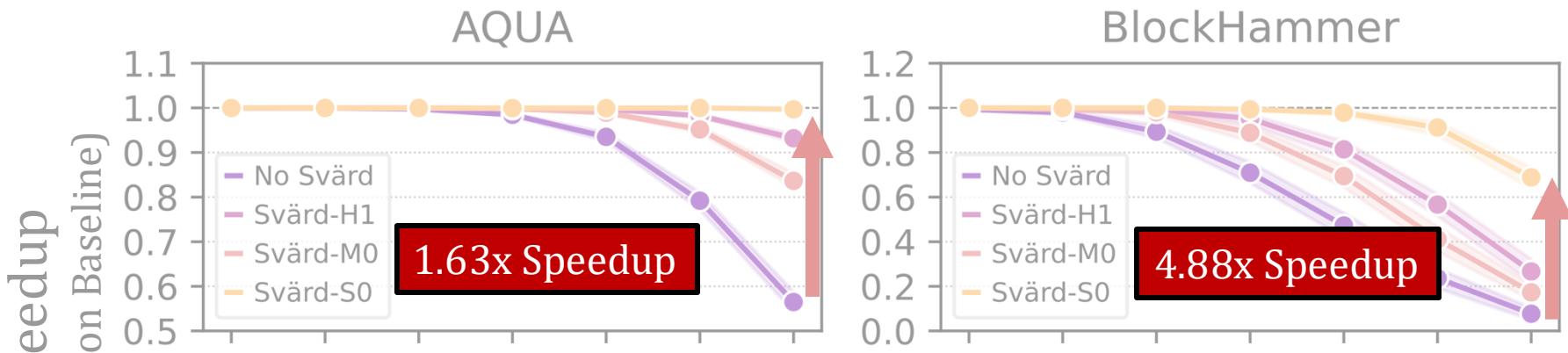
Implications on Future Solutions



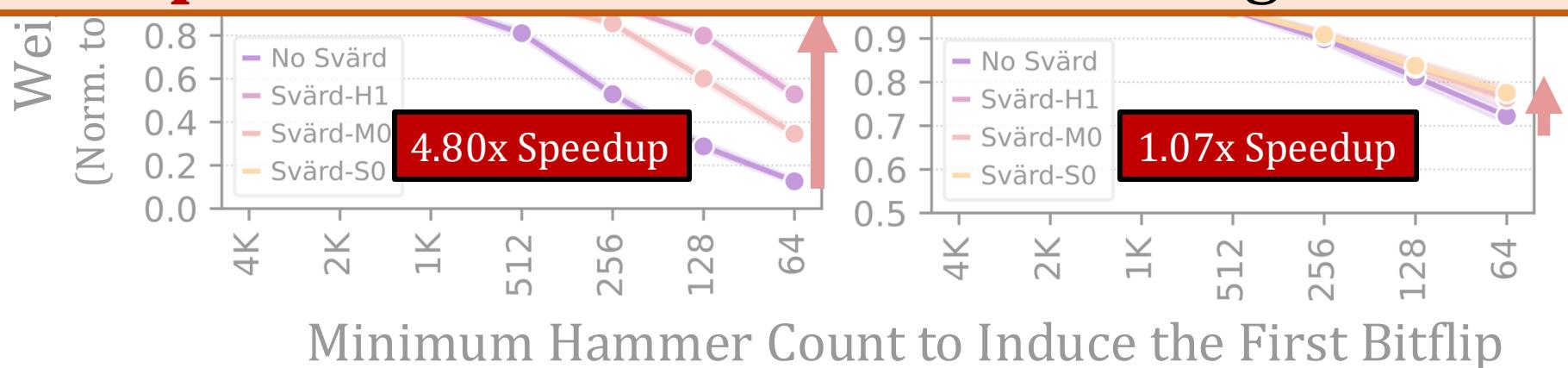
Implications on Future Solutions



Implications on Future Solutions



Svärd significantly **reduces**
the performance overhead of existing solutions



More in the Paper

- Spatial variation in read disturbance **bit error rate** across rows
- Spatial variation of **RowPress [Luo+ ISCA23]** across DRAM rows
- **Algorithms and details** of our experiments
- Reverse-engineering of **subarray boundaries**
- Svärd's **memory controller-based** and **on-DRAM-die** implementations
- Hardware complexity analysis
 - Chip area cost
 - MC-based: 0.027% of a processor die per DRAM bank
 - DRAM-based: 0.006% increase the DRAM array (4 bits per row per chip)
 - No additional latency overhead
- A preliminary analysis of **the effect of aging** on DRAM read disturbance vulnerability

Conclusion

- **The first rigorous experimental study** on the spatial variation of DRAM read disturbance across DRAM rows
 - **144 DDR4 DRAM chips** from **three major manufacturers**
 - Characterize **all rows in a bank** and four banks in a DRAM chip

Read disturbance vulnerability varies **significantly** and **irregularly** across DRAM rows

- **Svärd: Spatial Variation-Aware Read Disturbance Defenses**
 - **Dynamically tunes** a solution's aggressiveness (e.g., perform more/less refresh) to the **victim row's vulnerability** to DRAM read disturbance
 - Implemented either in **the memory controller** or in **the DRAM chip**

Svärd significantly **reduces**
the performance overhead of existing solutions

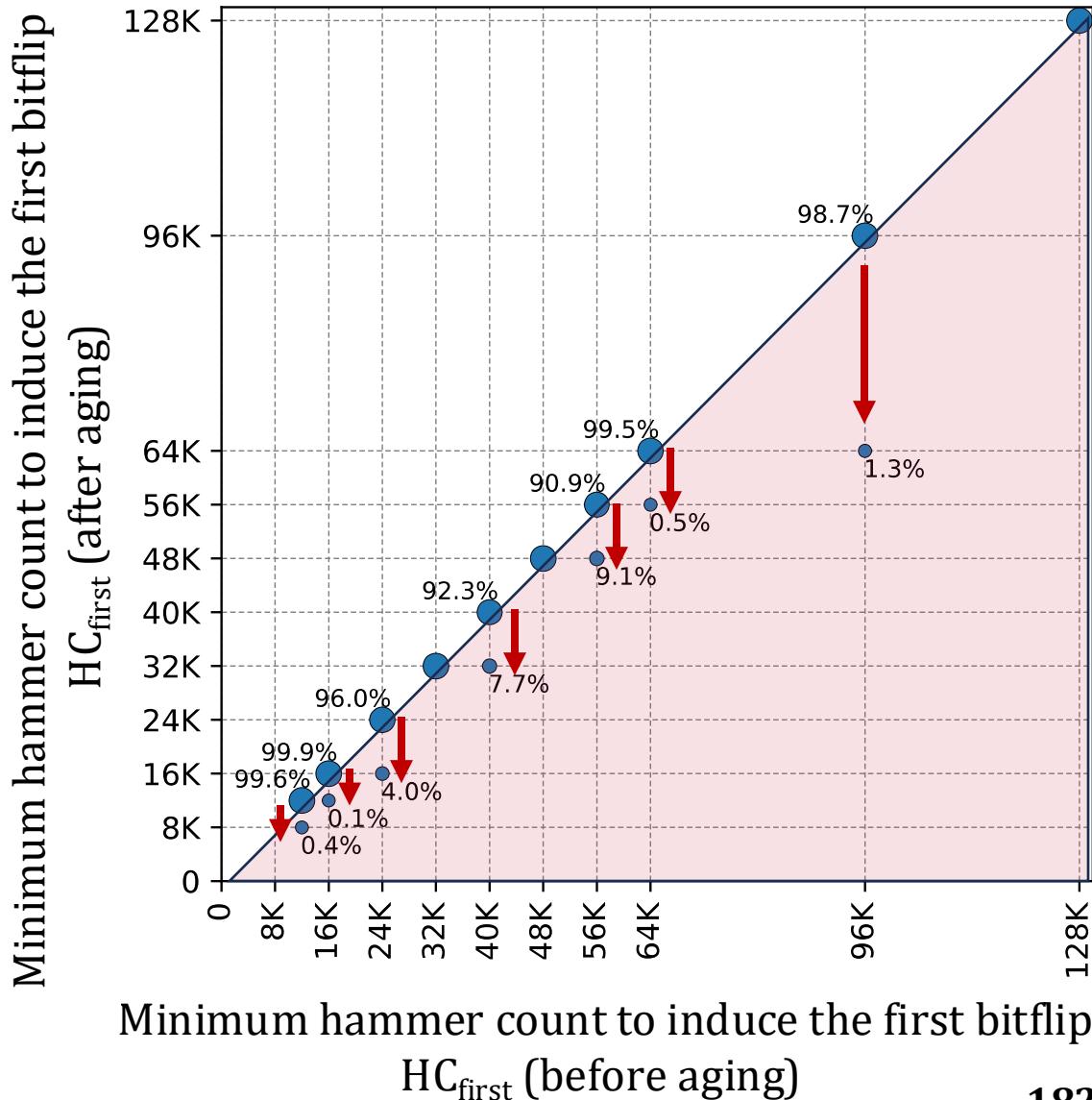
Svärd may present itself to any worthy read disturbance solution



More in the Paper (2/2): Aging Study

Preliminary data on aging via 68-day of continuous hammering

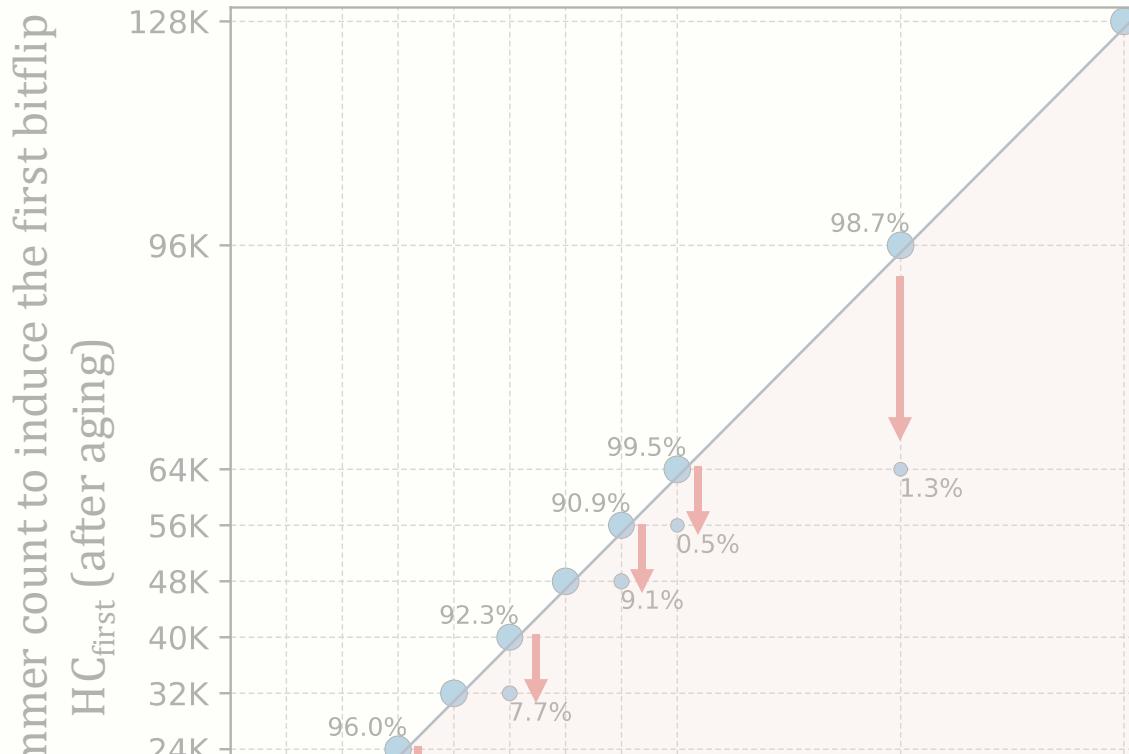
Aging can lead to read disturbance bitflips at smaller hammer counts



More in the Paper (2/2): Aging Study

Preliminary data on aging via 68-day of continuous hammering

Aging can lead to read disturbance bitflips at smaller hammer counts



Future work:
rigorous aging characterization
and online profiling of read disturbance vulnerability

Minimum hammer count to induce the first bitflip
 HC_{first} (before aging)

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Hydra's Performance



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

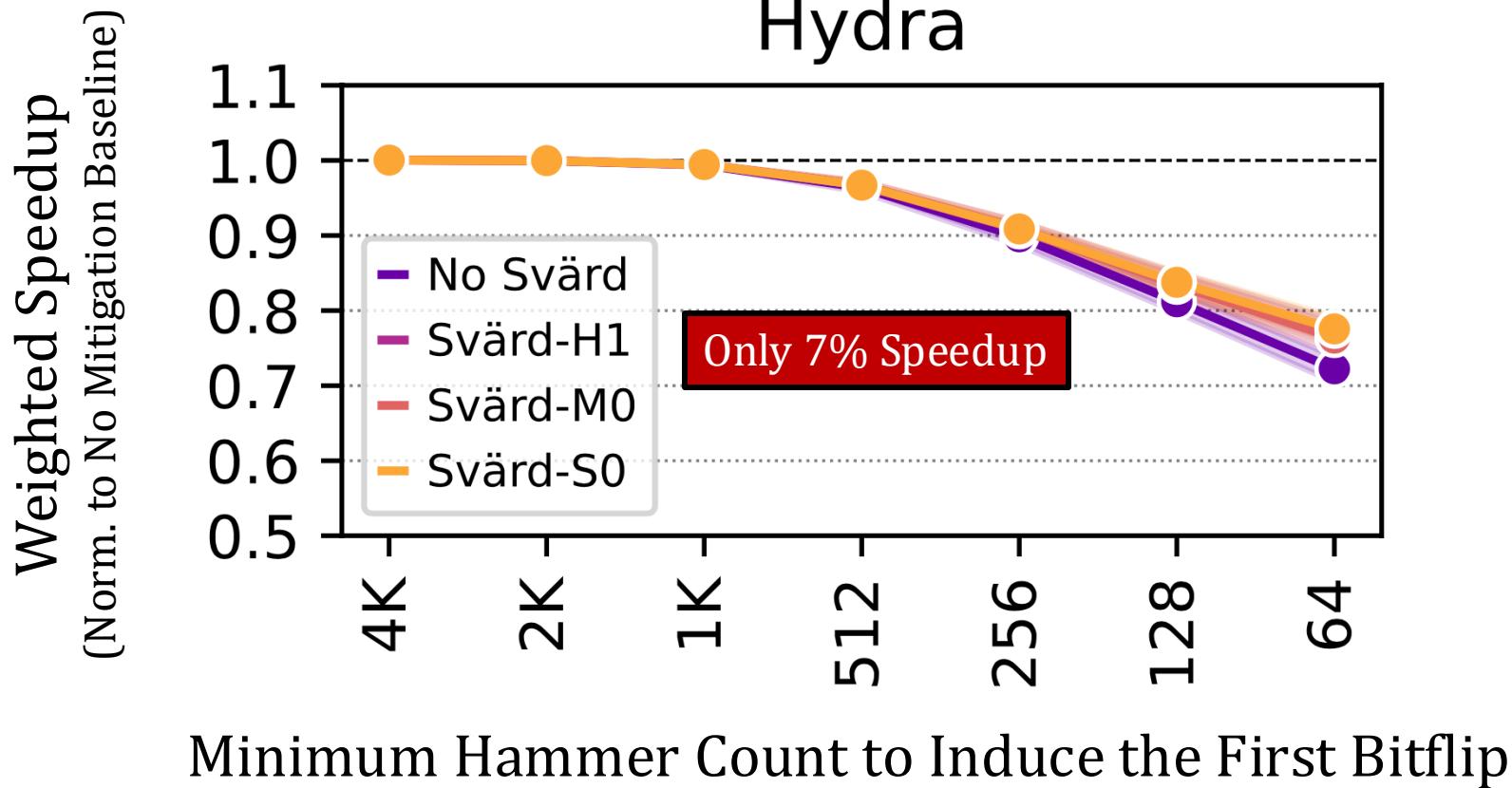
Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

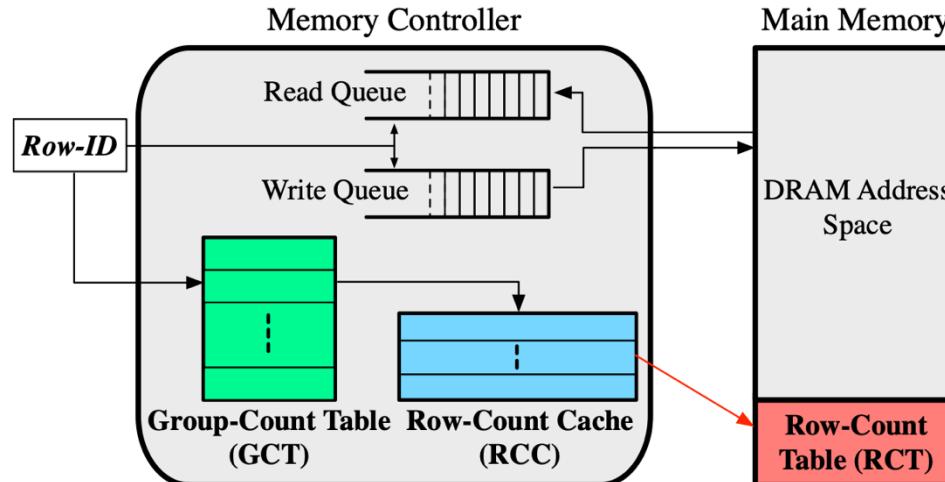
ETH zürich

An Outlier Solution: Hydra



Hydra Mitigation Mechanism

- Hydra maintains **a row activation counter for each DRAM row**
- **Stores** these activation counters **in the DRAM array**
- **Caches** the counters of hot rows **in the memory controller**
- At low HC_{first} configurations, **many rows are hot**
- **Fetching / evicting counters** dominate the performance overhead
- Svärd needs **further customizations** for Hydra



Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

HC_{first} across Rows



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

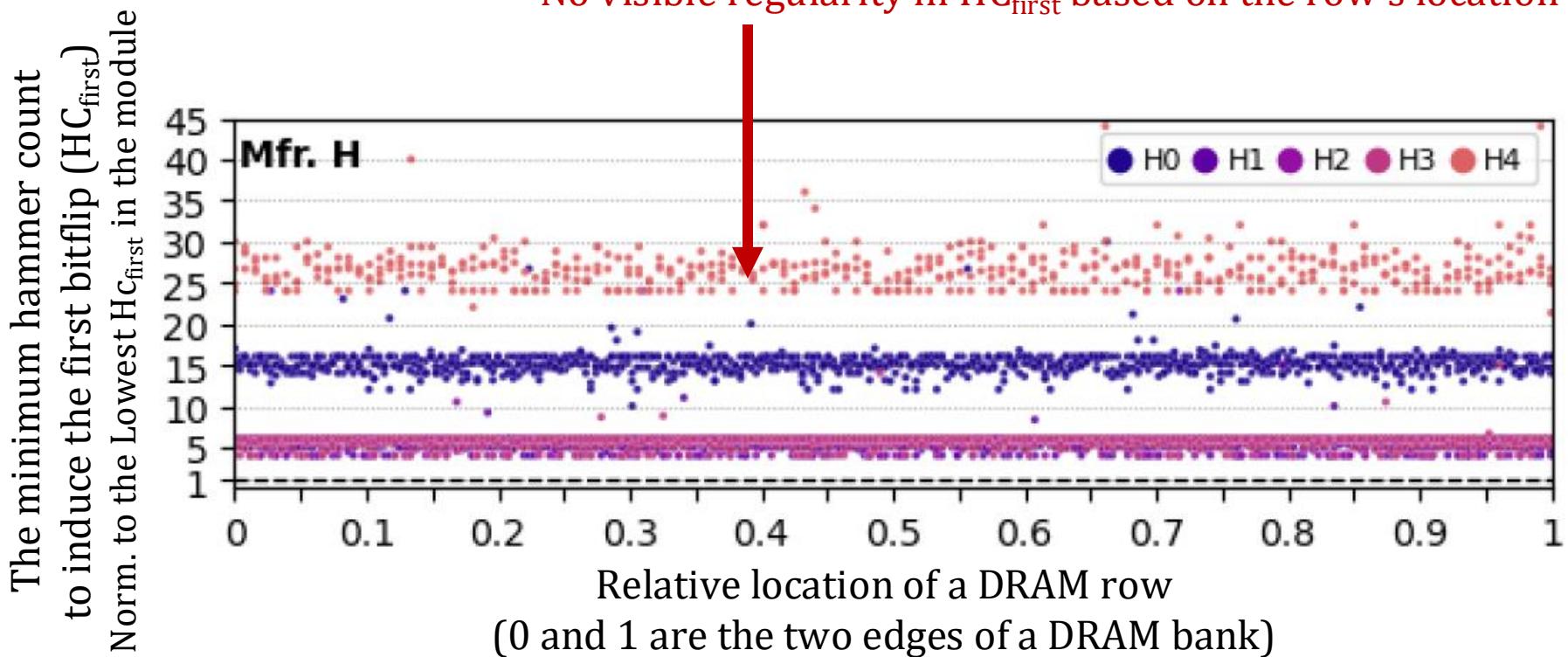
Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

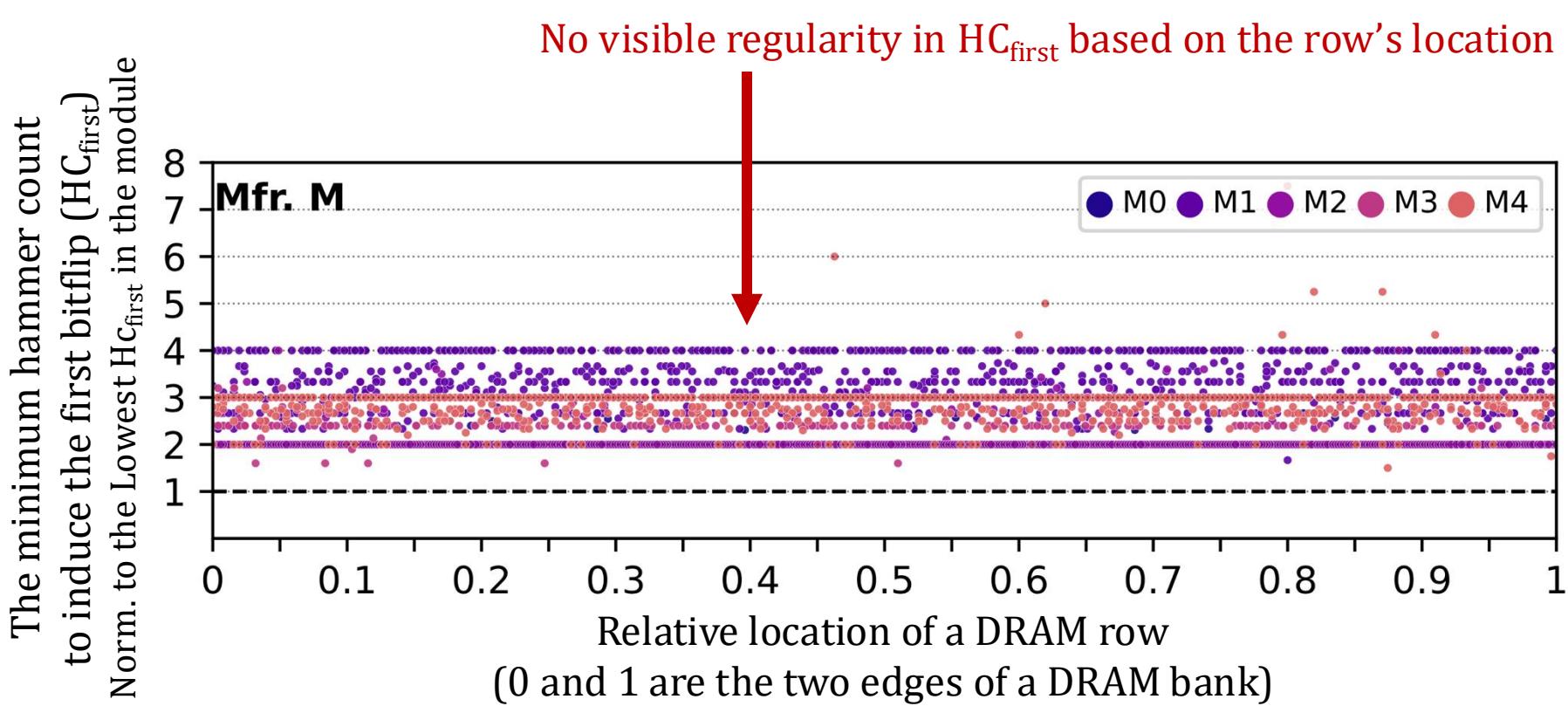
ETH zürich

The Minimum Hammer Count to Induce the First Bitflip across DRAM Rows



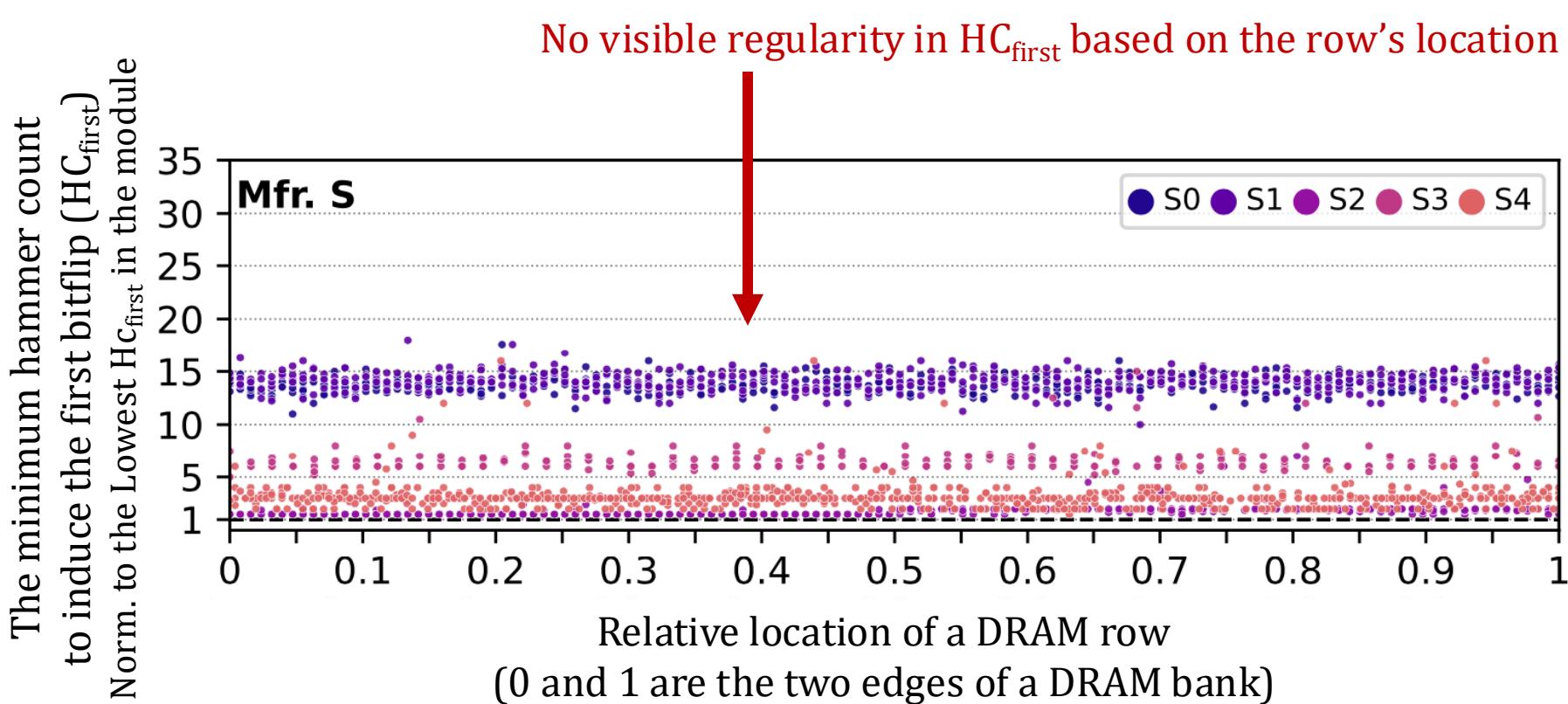
The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

The Minimum Hammer Count to Induce the First Bitflip across DRAM Rows



The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

The Minimum Hammer Count to Induce the First Bitflip across DRAM Rows



The minimum hammer count to induce the first bitflip **irregularly varies** with respect to row's location in DRAM bank

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

BER across Rows



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

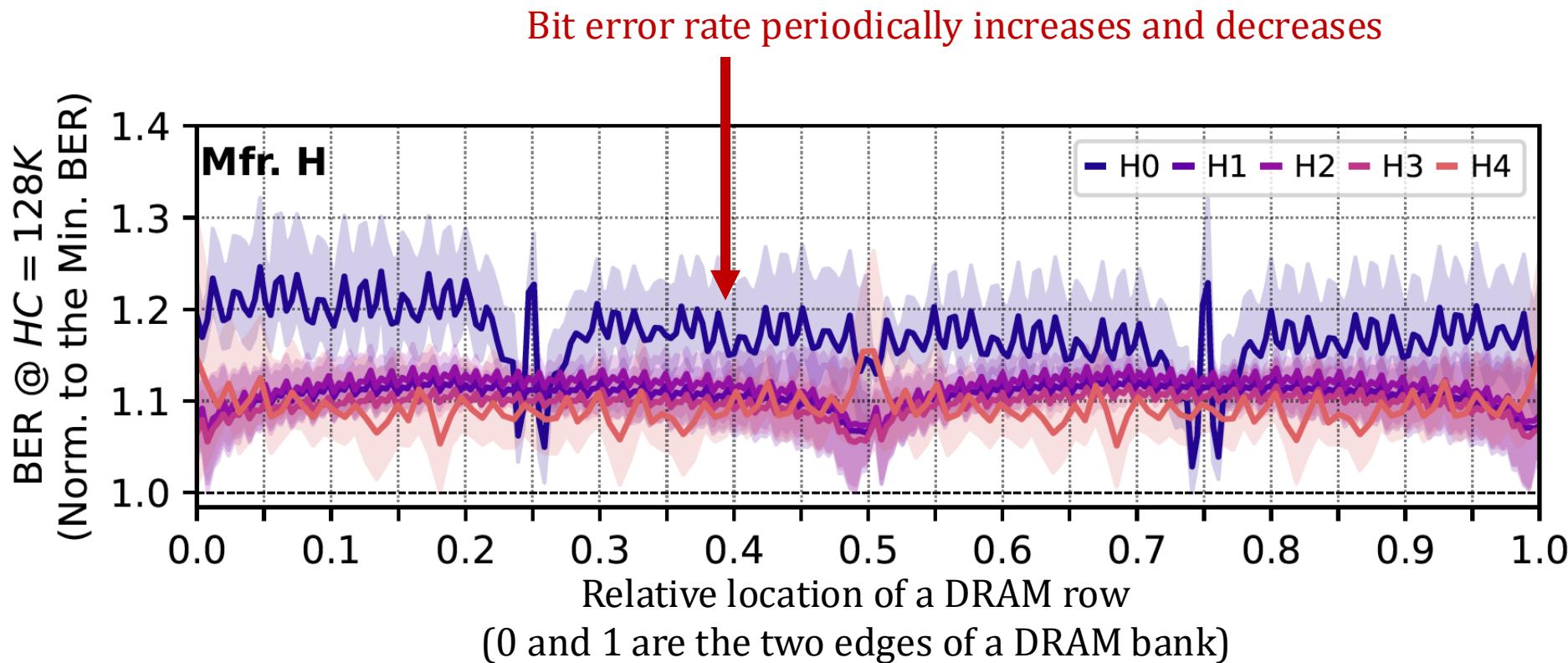
Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

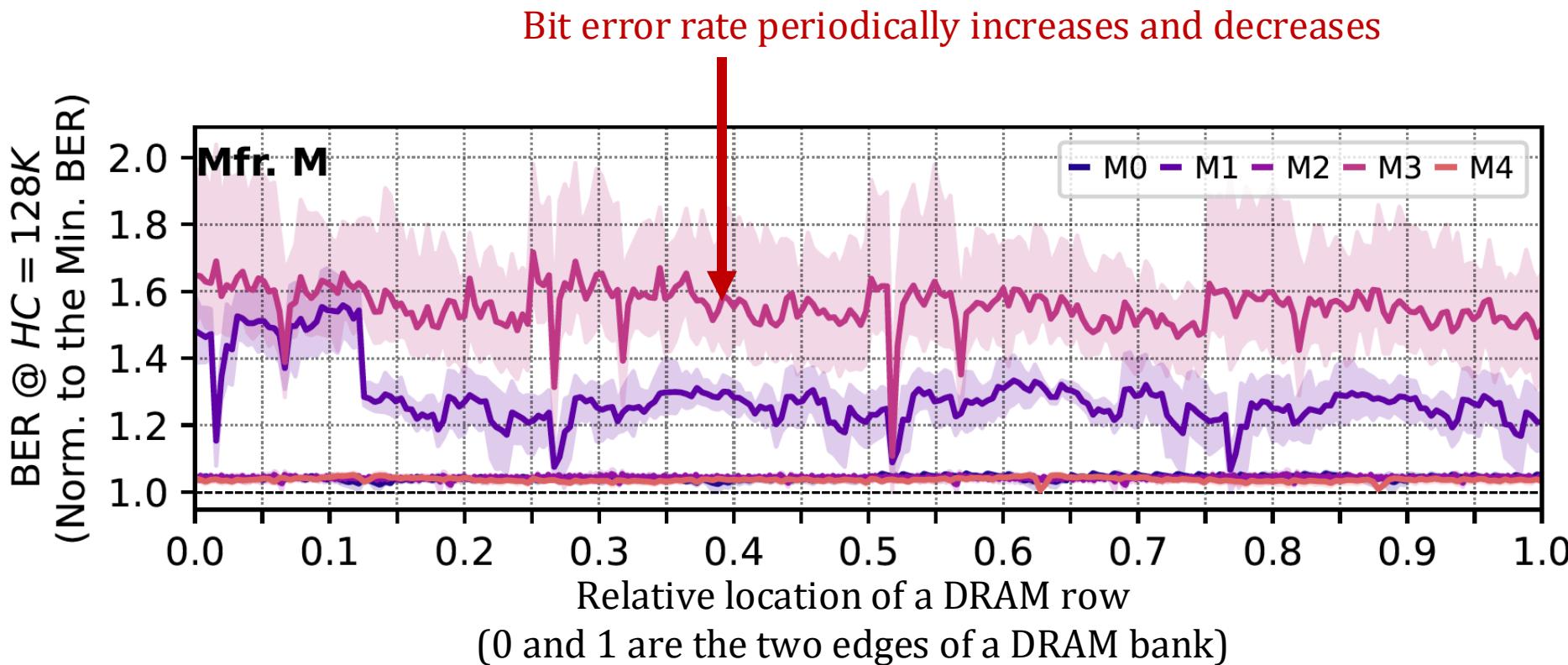
ETH zürich

The Read Disturbance Bit Error Rate across DRAM Rows



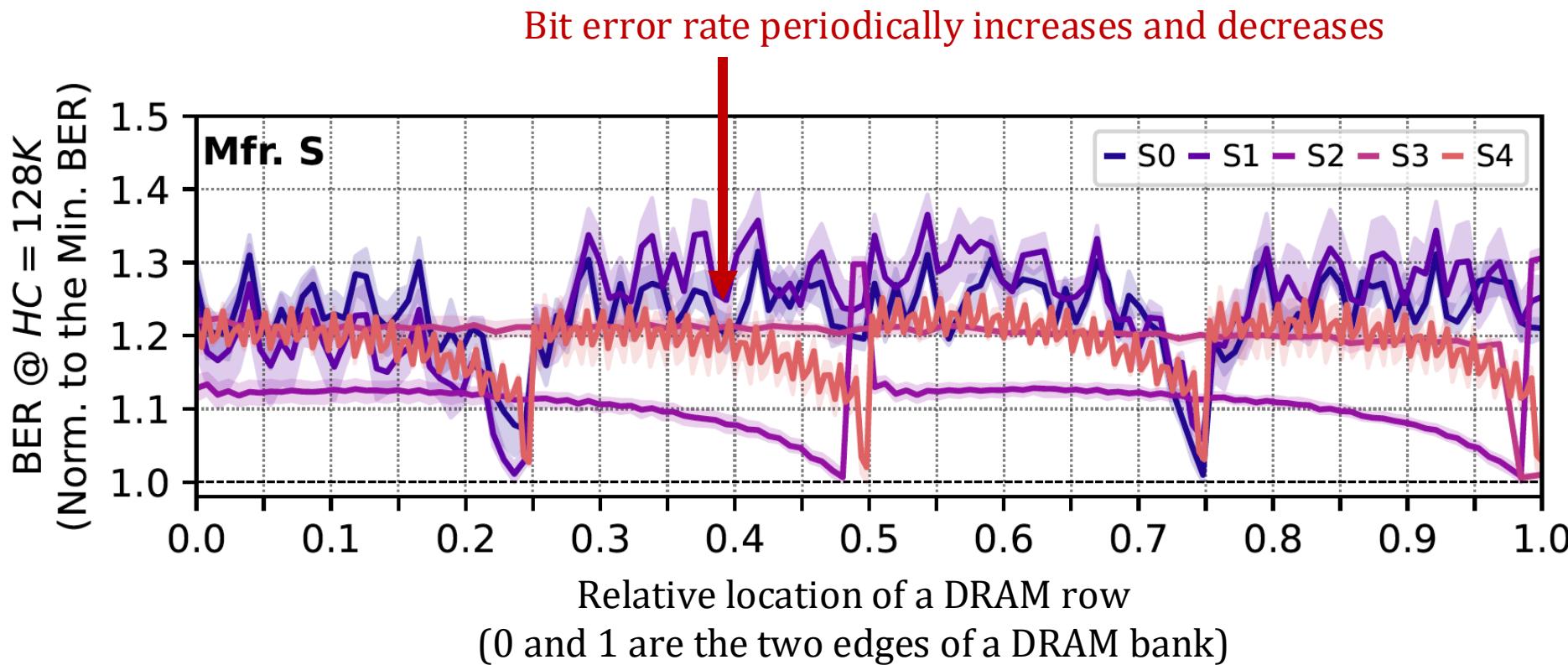
The variation in read disturbance bit error rate exhibits a stronger regularity compared to the variation in varies with respect to row's location in DRAM bank

The Read Disturbance Bit Error Rate across DRAM Rows



The variation in read disturbance bit error rate exhibits a stronger regularity compared to the variation in varies with respect to row's location in DRAM bank

The Read Disturbance Bit Error Rate across DRAM Rows



The variation in read disturbance bit error rate
exhibits a stronger regularity compared to the variation in
varies with respect to row's location in DRAM bank

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Background



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

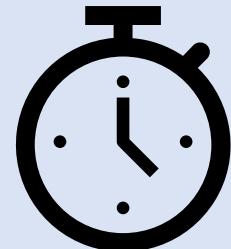
SAFARI

ETH zürich

Two Main Types of DRAM Refresh

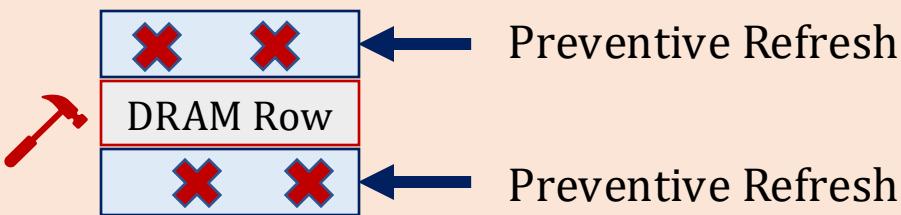
1

Periodic Refresh: Periodically **restores** the charge
DRAM cells leak **over time**



2

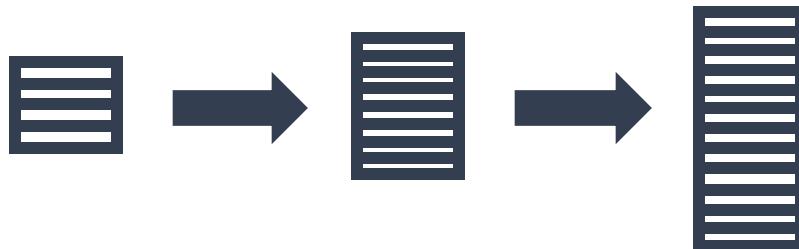
RowHammer: Repeatedly accessing a DRAM row can cause
bit flips in other **physically nearby rows**



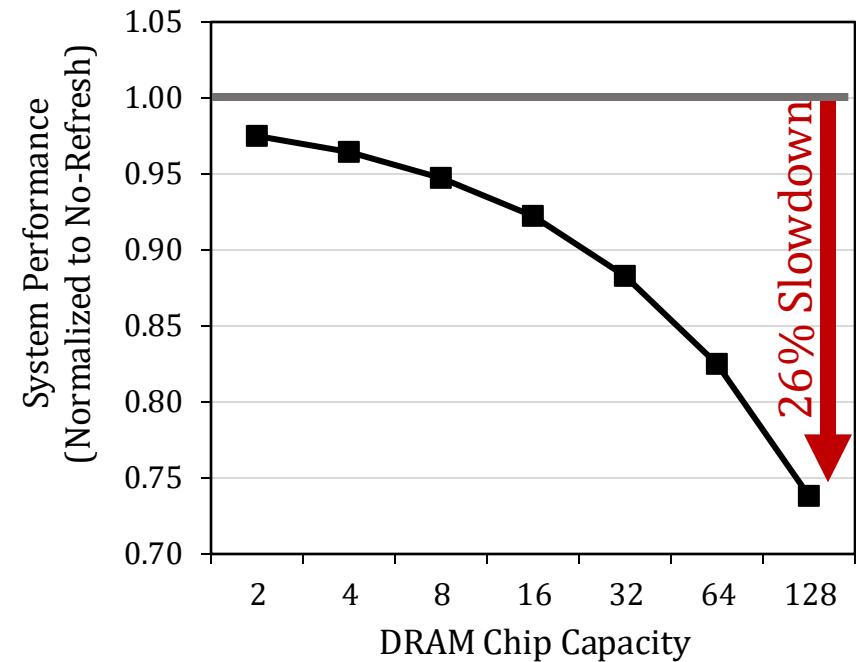
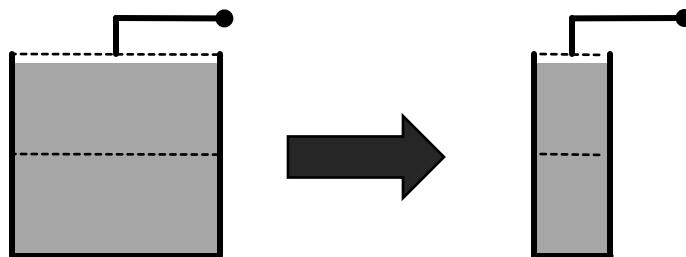
Preventive Refresh: Mitigates RowHammer
by **refreshing physically nearby rows**
of a repeatedly accessed row

Periodic Refresh with Increasing DRAM Chip Density

A **larger capacity** chip has **more rows** to be refreshed



A **smaller** cell stores **less charge**



More periodic refresh operations incur
larger performance overhead as DRAM **chip density increases**

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

RowPress



Full Paper
[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

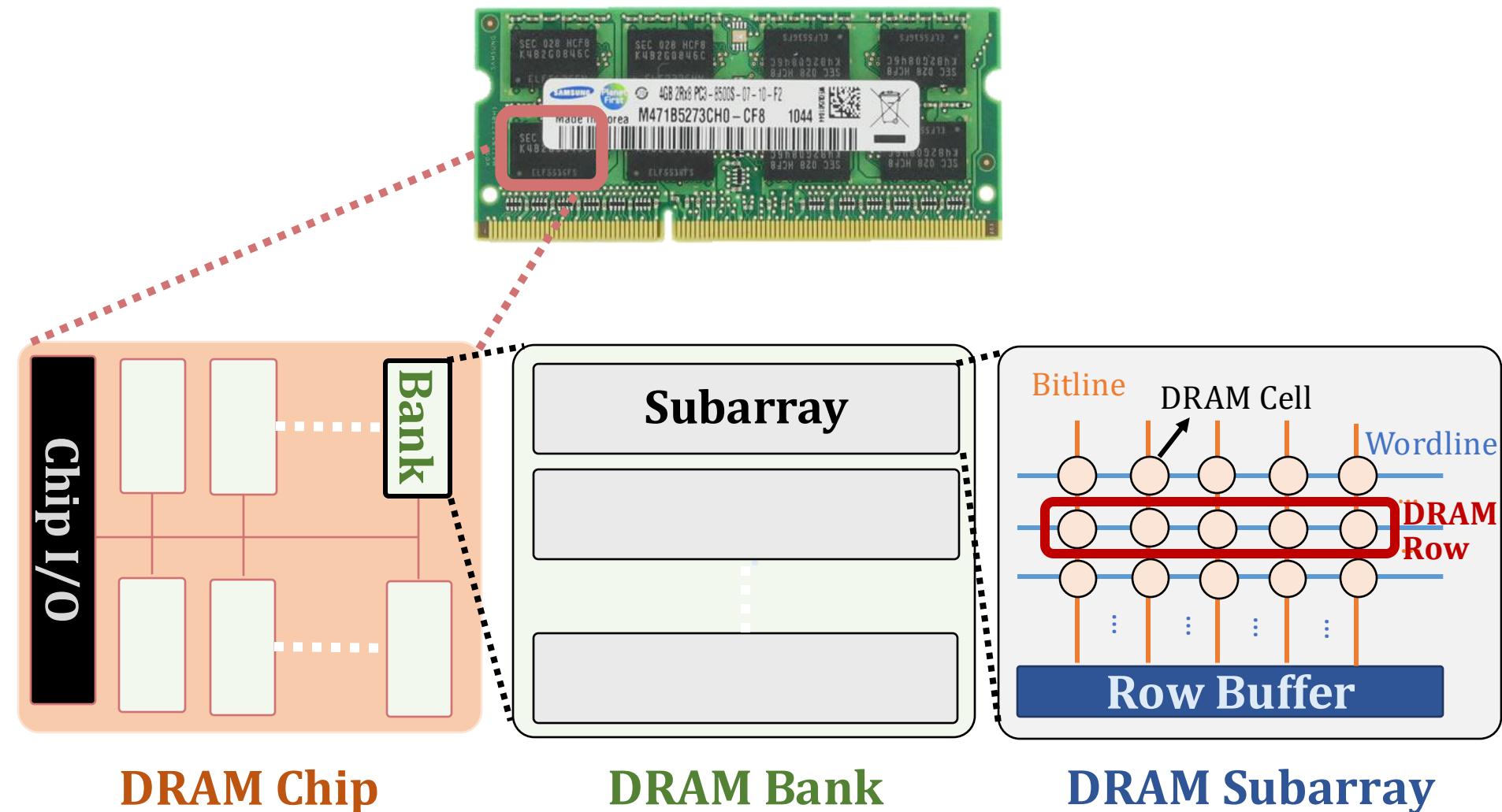
Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

ETH zürich

DRAM Organization

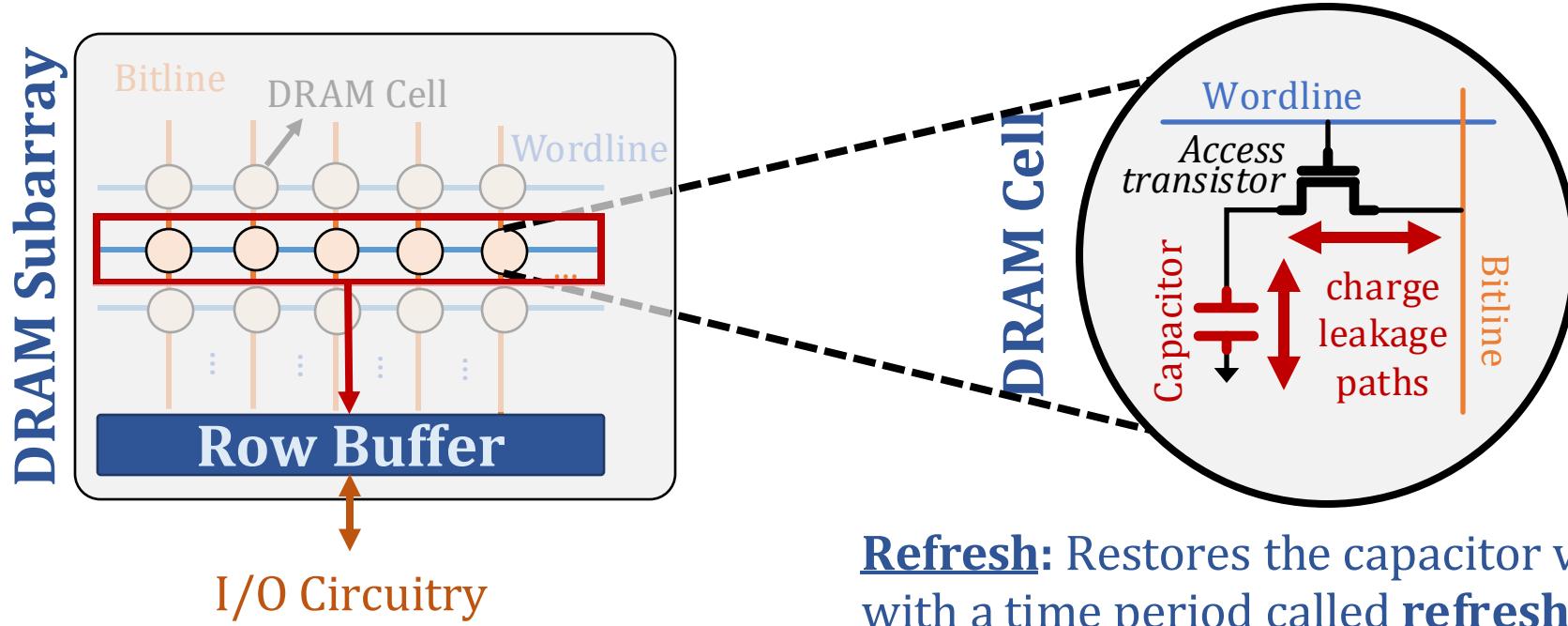


DRAM Chip

DRAM Bank

DRAM Subarray

DRAM Operation

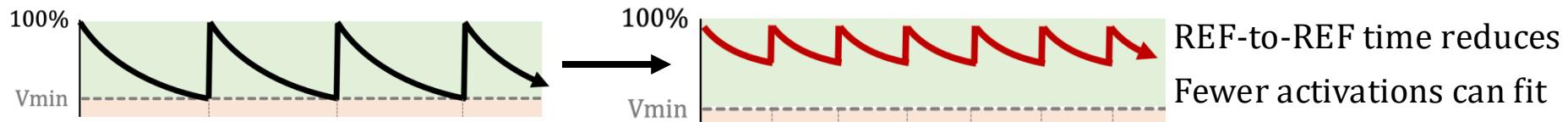


Refresh: Restores the capacitor voltage with a time period called **refresh window**

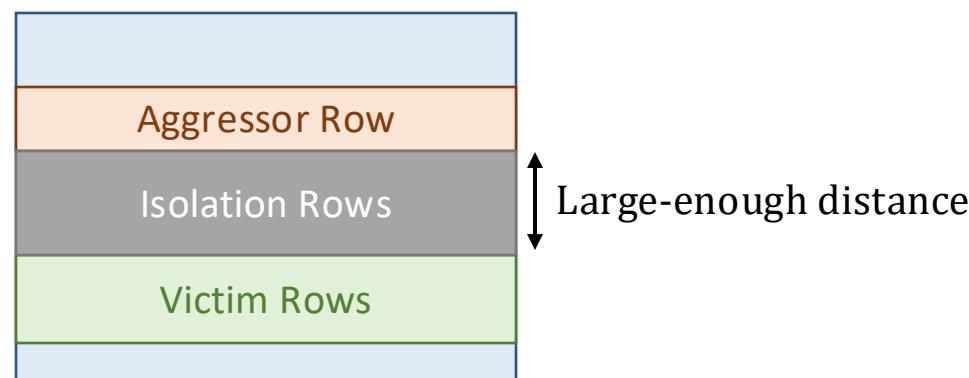
- 1. Row Activation:** Fetch the row's content into the row buffer
- 2. Column Access:** Read/Write a column in the row buffer
- 3. Precharge:** Disconnect the row from the row buffer

RowHammer Mitigation Approaches

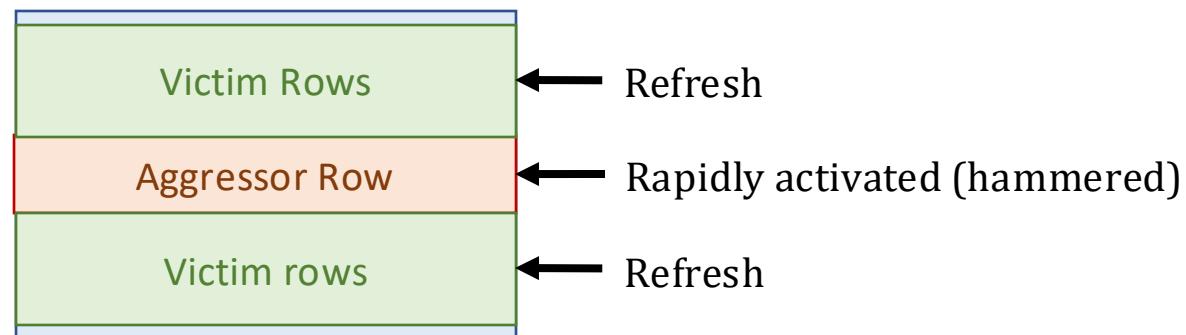
- Increased refresh rate



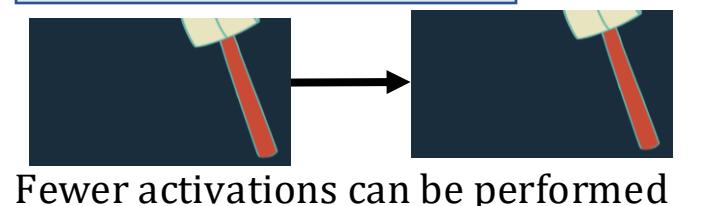
- Physical isolation



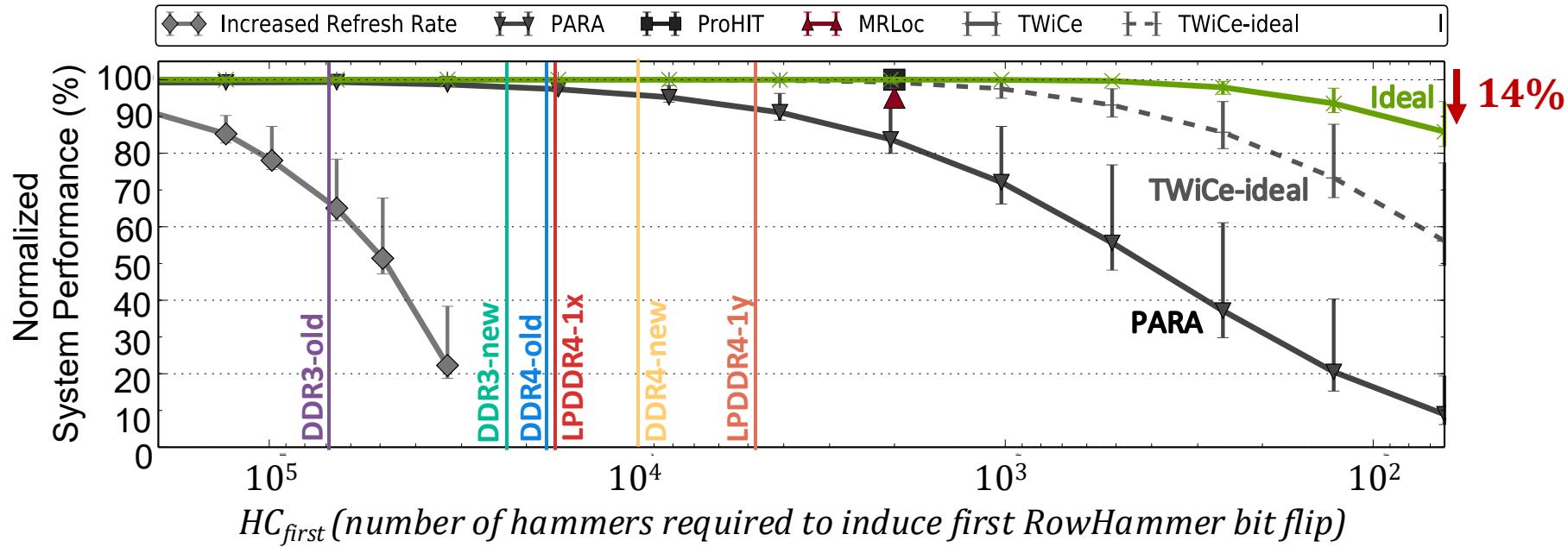
- Reactive refresh



- Proactive throttling



RowHammer Mitigation across Generations



J. S. Kim, M. Patel, A. G. Yaglikci, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)," in *ISCA*, 2020.

RowPress [ISCA 2023]

- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,
"RowPress: Amplifying Read Disturbance in Modern DRAM Chips"

Proceedings of the 50th International Symposium on Computer Architecture (ISCA), Orlando, FL, USA, June 2023.

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

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

[[Lightning Talk Video](#) (3 minutes)]

[[RowPress Source Code and Datasets](#) ([Officially Artifact Evaluated with All Badges](#))]

Officially artifact evaluated as available, reusable and reliable.
Best artifact award at ISCA 2023.

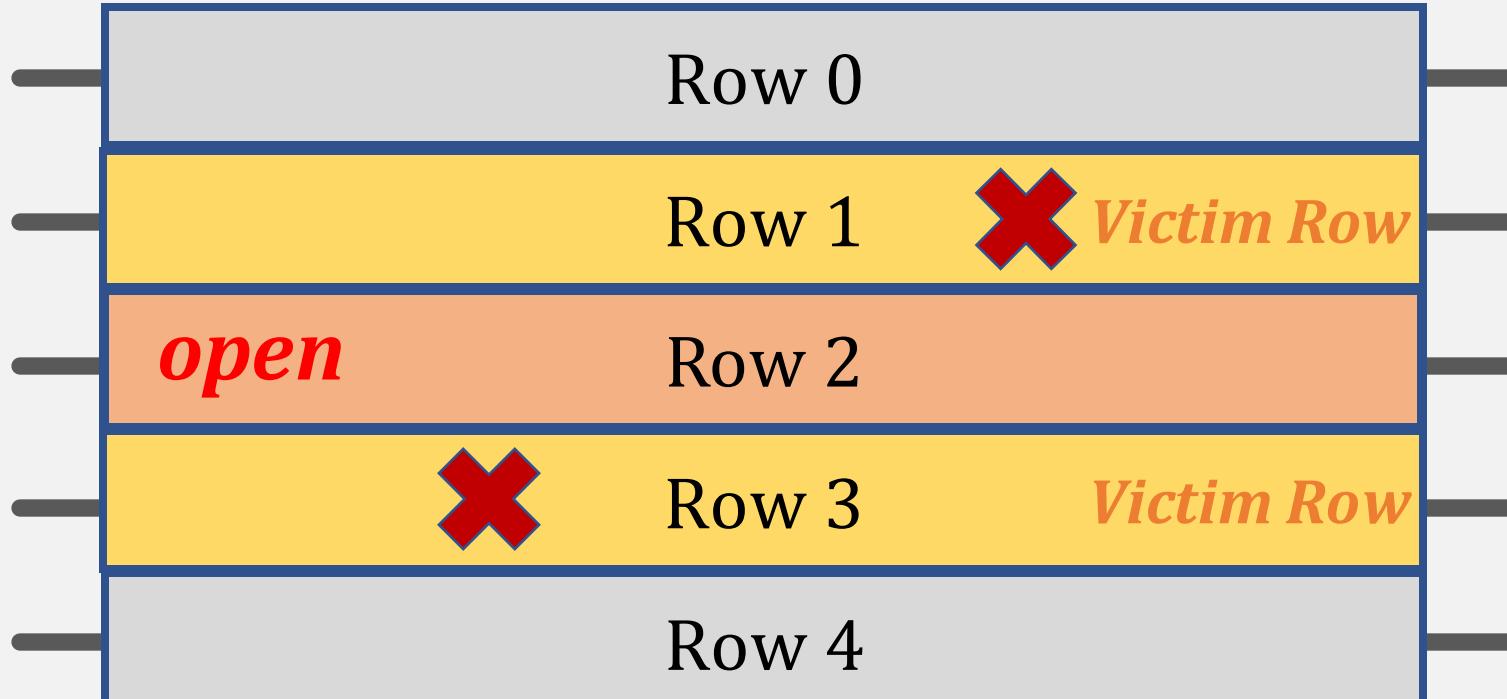


RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

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

ETH Zürich

DRAM Subarray

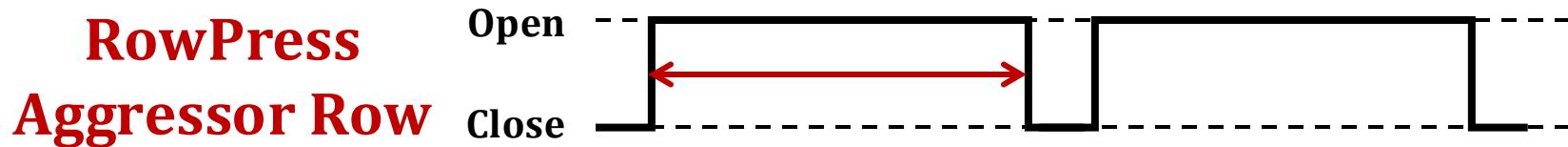
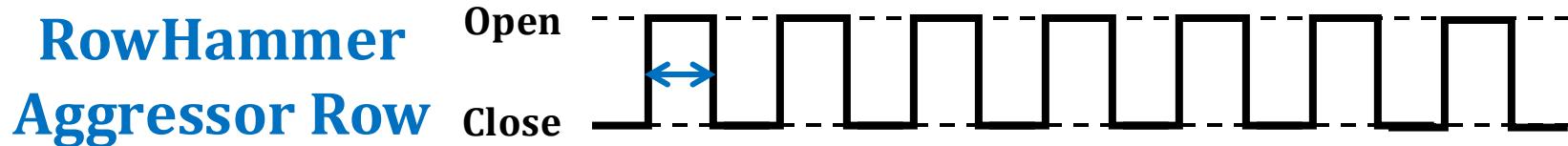


Keeping a DRAM row **open** (activated)
causes **RowPress bitflips** in nearby cells

Two Prime Examples of DRAM Read Disturbance: RowHammer and RowPress

Instead of using a high activation count,

- ☛ increase the time that the aggressor row stays open



RowPress [ISCA 2023]

- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,

"RowPress: Amplifying Read Disturbance in Modern DRAM Chips"

Proceedings of the 50th International Symposium on Computer Architecture (ISCA), Orlando, FL, USA, June 2023.

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

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

[[Lightning Talk Video](#) (3 minutes)]

[[RowPress Source Code and Datasets](#) ([Officially Artifact Evaluated with All Badges](#))]

Officially artifact evaluated as available, reusable and reliable
Best artifact award at ISCA 2023.

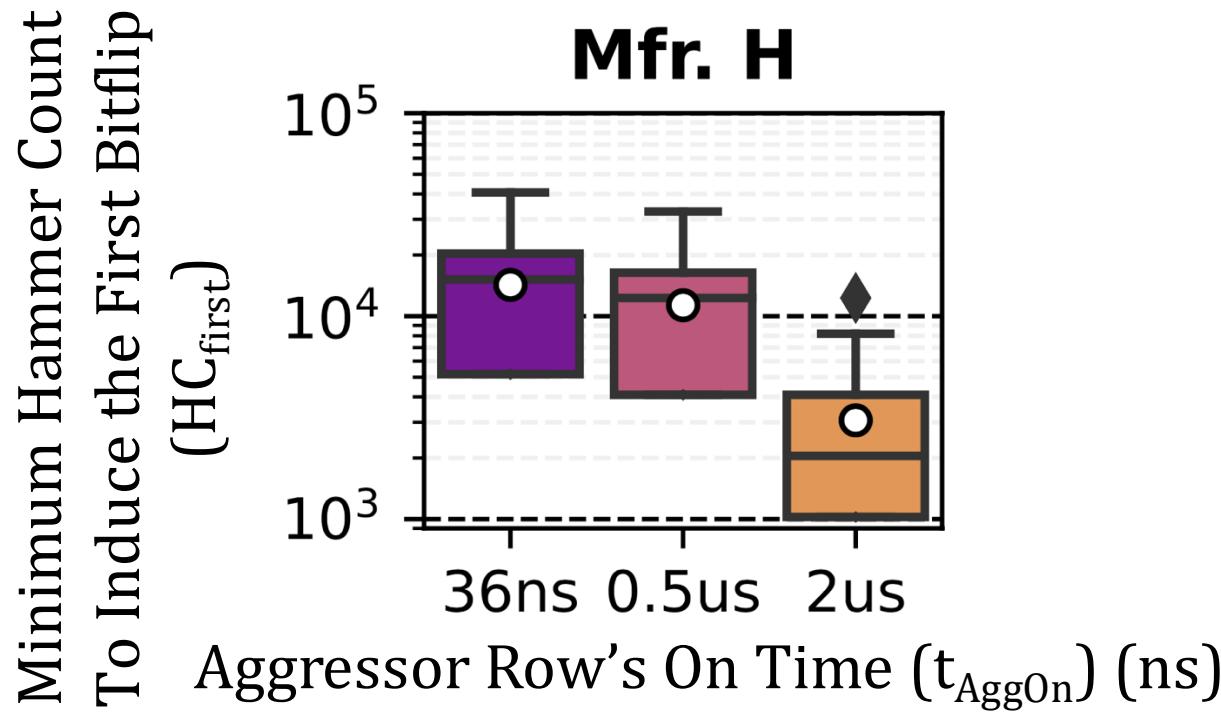


RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

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

ETH Zürich

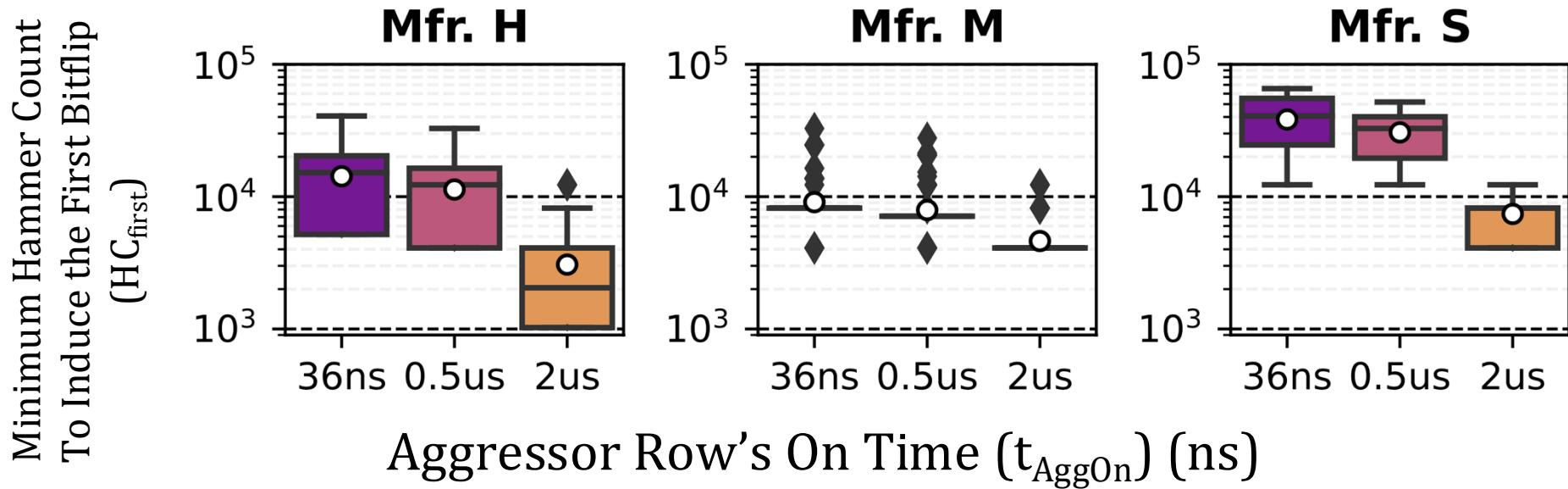
Effect of RowPress on the Spatial Variation of Read Disturbance Vulnerability across Rows



RowPress reduces the mean of the distribution with increased t_{AggOn}

There is still significant variation in the HC_{first} across rows

Effect of RowPress on the HCfirst Distribution



Bitflips occur at **lower hammer counts** with RowPress and these hammer counts still **significantly vary** across DRAM rows

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Predictability



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

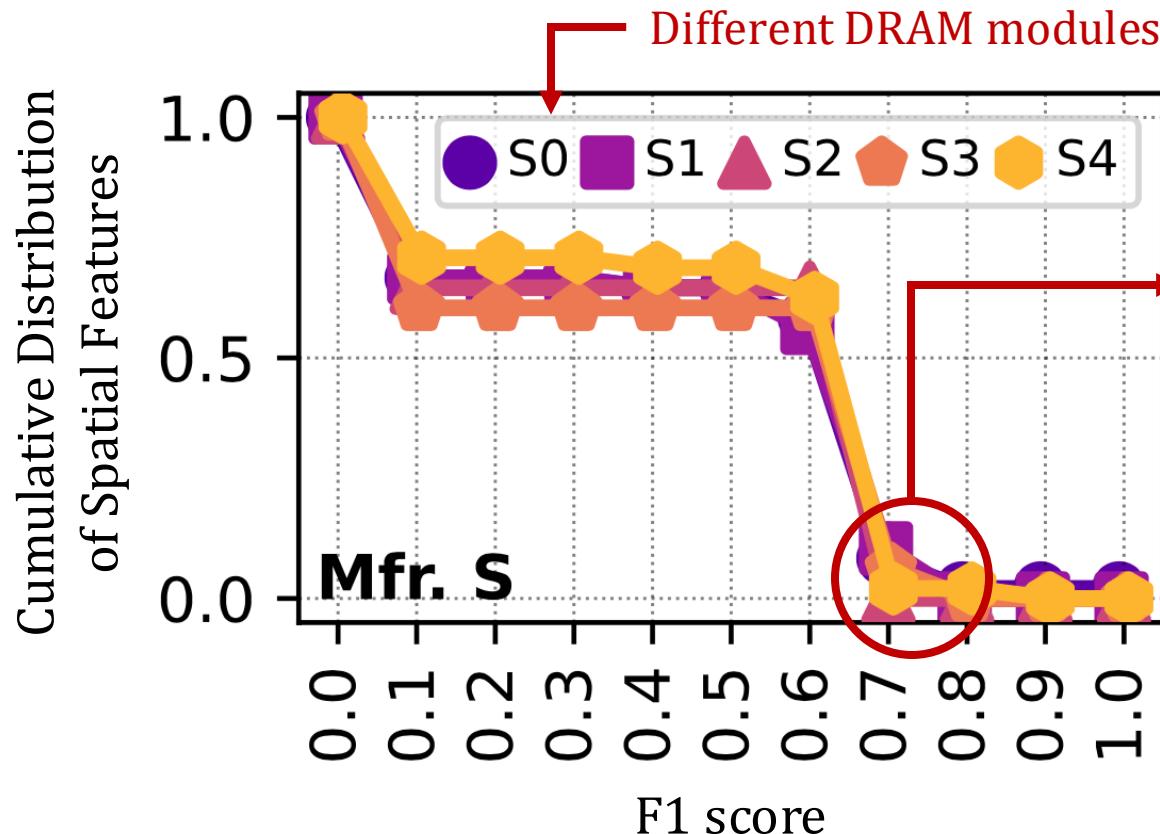
Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

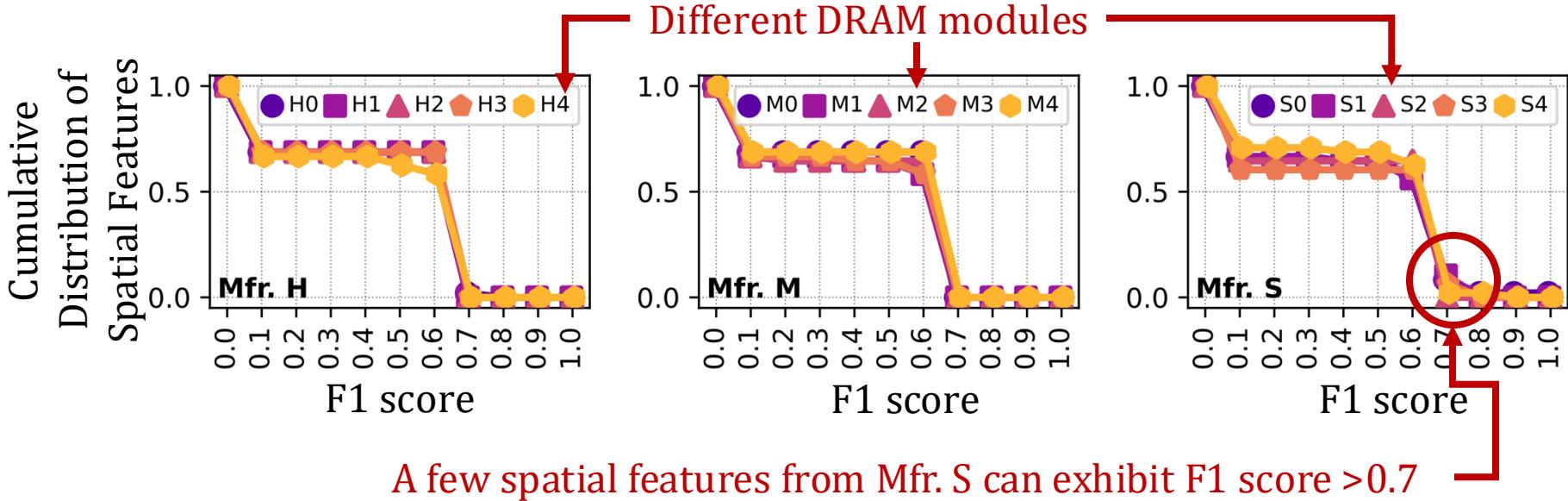
ETH zürich

Cumulative Distribution of Spatial Features based on F1 Score



Weak prediction observed between a row's
spatial features & read disturbance **vulnerability**

Cumulative Distribution of Spatial Features based on F1 Score



Weak prediction observed between a row's
spatial features & read disturbance **vulnerability**

Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Implementation Details



Full Paper

[arXiv \[cs.CR\] 2402.18652](https://arxiv.org/abs/2402.18652)

Abdullah Giray Yağlıkçı

Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel

Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

ETH zürich

Svärd: Spatial Variation-Aware Read Disturbance Defenses

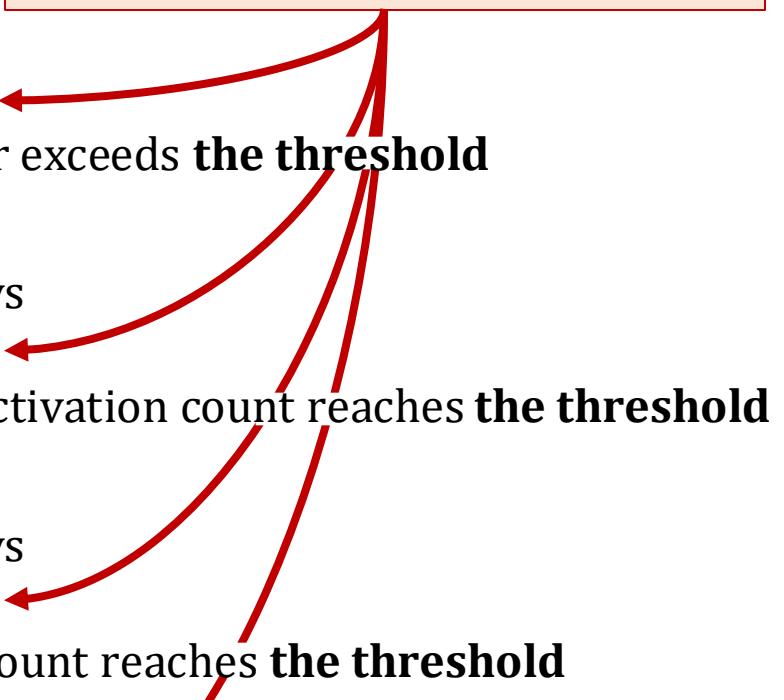
Integration Examples with Existing Defenses

- Integrate Svärd with five solutions

1) PARA [Kim+, ISCA'14]

- Generates a **random number**
- Compares the random number with a **threshold**
- **Refreshes the victim row** if the random number exceeds **the threshold**

Svärd tunes **the threshold** based on the **victim row's vulnerability**



2) BlockHammer [Yaglikci+, HPCA'21]

- Counts **the number of activations** of DRAM rows
- Compares the activation count **with a threshold**
- **Throttles accesses** to the aggressor row if the activation count reaches **the threshold**

3) Hydra [Qureshi+, ISCA'22]

- Counts **the number of activations** of DRAM rows
- Compares the activation count **with a threshold**
- **Refreshes the victim row** when the activation count reaches **the threshold**

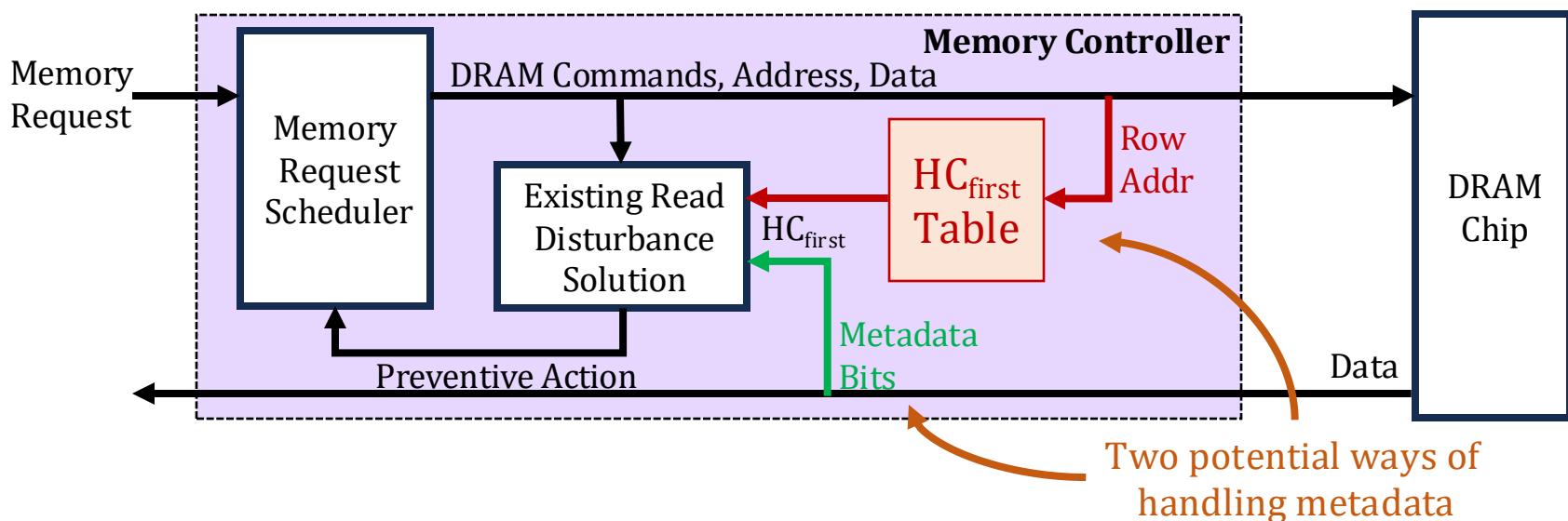
4) AQUA [Saxena+, MICRO'22] and 5) RRS [Saileshwar+, ASPLOS'22]

- Counts **the number of activations** of DRAM rows
- Compares the activation count **with a threshold**
- **Relocates the aggressor row** when the activation count reaches **the threshold**

Svärd: Spatial Variation-Aware Read Disturbance Defenses

Example Implementation 1

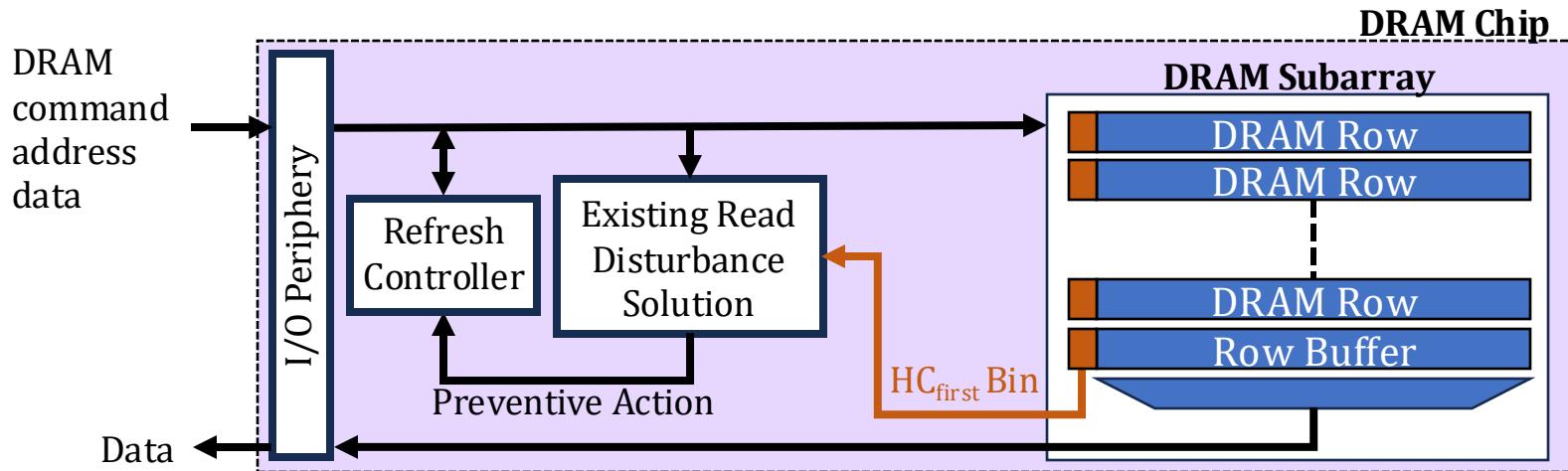
- Classifies DRAM rows into **several vulnerability-level bins**
 - Maintains **a few (e.g., four) bits** per DRAM row
- Implemented in either the memory controller or the DRAM chip
- Two example memory controller-based implementations:



Svärd: Spatial Variation-Aware Read Disturbance Defenses

Example Implementation 2

- Classifies DRAM rows into **several vulnerability-level bins**
 - Maintains **a few (e.g., four) bits** per DRAM row
- Implemented in either the memory controller or the DRAM chip
- An example DRAM chip-based implementation:
 - **Additional few (e.g., four) bits** per DRAM row (e.g., 8Kb)



Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions

Backup Slides

Abdullah Giray Yağlıkçı

Yahya Can Tuğrul Geraldo F. Oliveira İsmail Emir Yüksel
Ataberk Olgun Haocong Luo Onur Mutlu

SAFARI

ETH zürich

More in the Paper

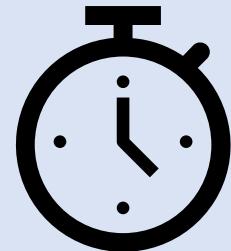
- Security Proof
 - Mathematically represent **all possible** access patterns
 - We show that **no row can be activated high-enough times** to induce bit-flips when BlockHammer is configured correctly
- Addressing **Many-Sided Attacks**
- Evaluation of **14 mechanisms** representing **four mitigation 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] BlockHammer	✓ ✓ ✓	✓ ✗ ✓	✗ ✗ ✓	✓ ✓ ✓

Two Main Types of DRAM Refresh

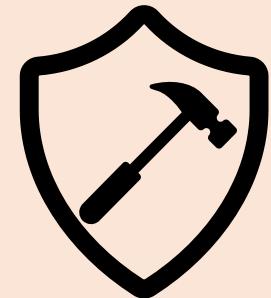
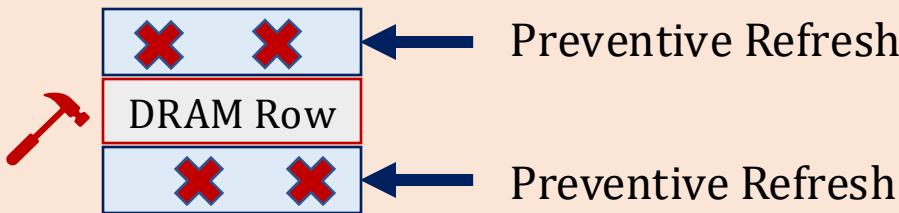
1

Periodic Refresh: Periodically **restores** the charge
DRAM cells leak **over time**



2

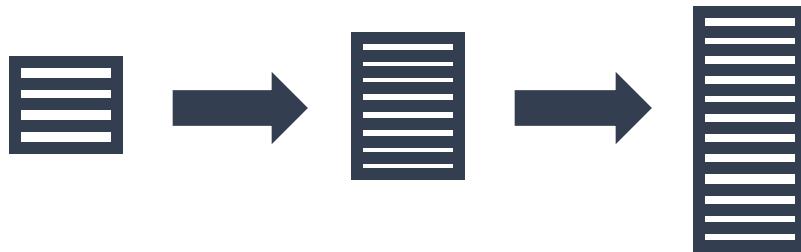
RowHammer: Repeatedly **accessing** a DRAM row can cause
bit flips in other **physically nearby rows**



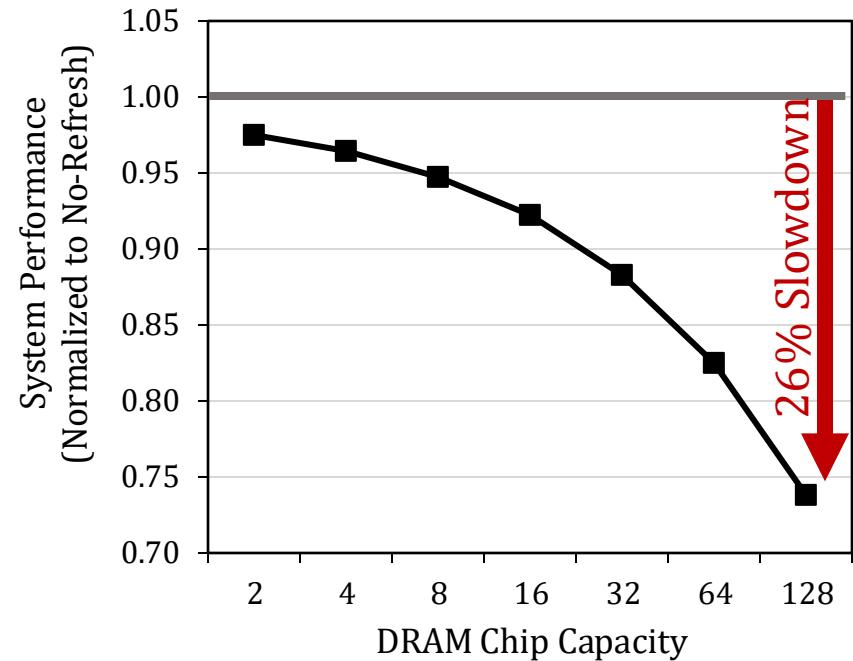
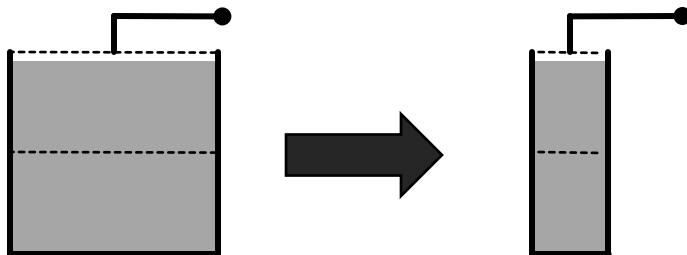
Preventive Refresh: Mitigates RowHammer
by **refreshing physically nearby rows**
of a repeatedly accessed row

Periodic Refresh with Increasing DRAM Chip Density

A **larger capacity** chip has **more rows** to be refreshed



A **smaller** cell stores **less charge**

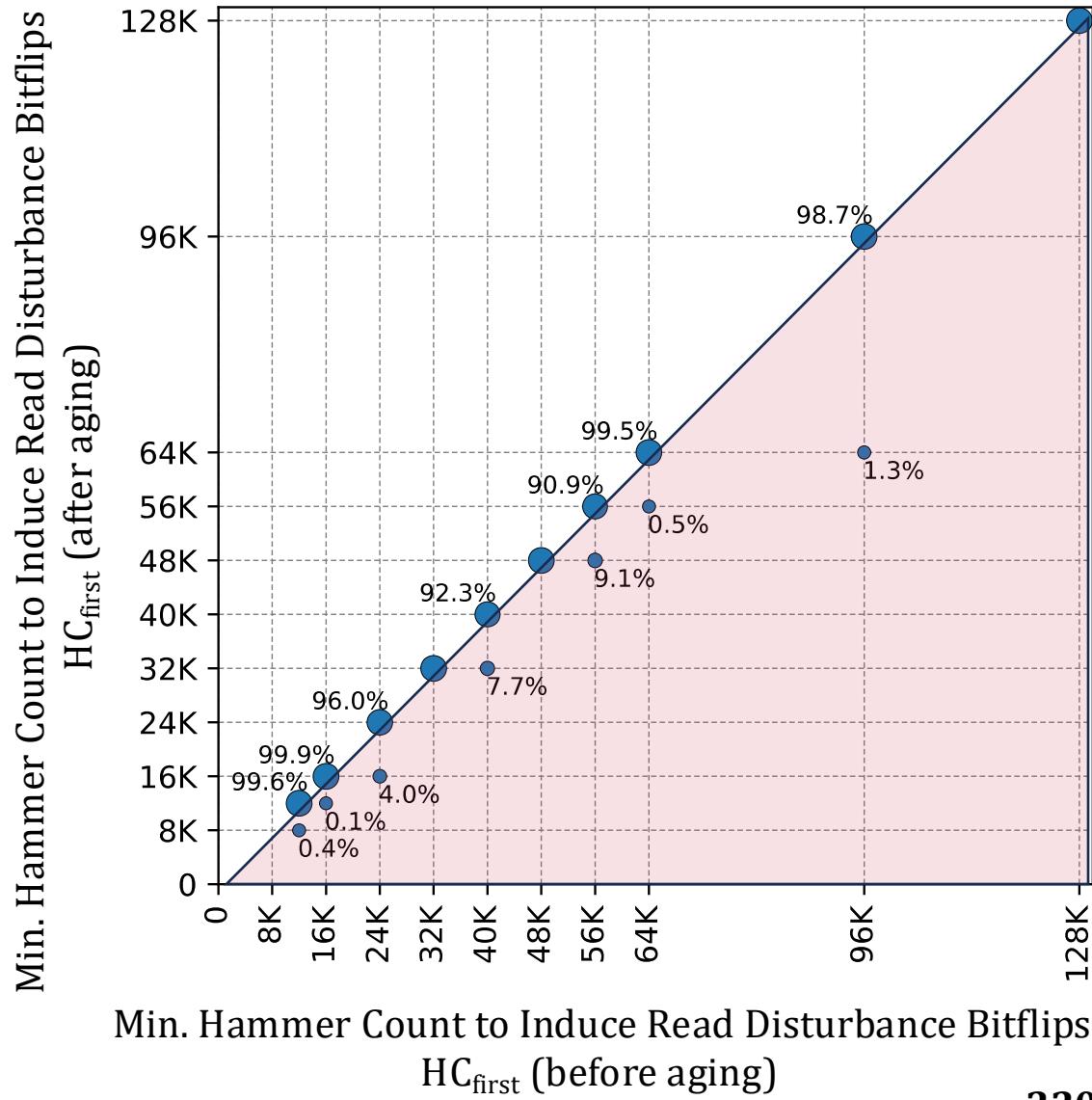


More periodic refresh operations incur
larger performance overhead as DRAM **chip density increases**

Future Research

- The effect of **aging**
- Preliminary data on aging via 68-day of hammering

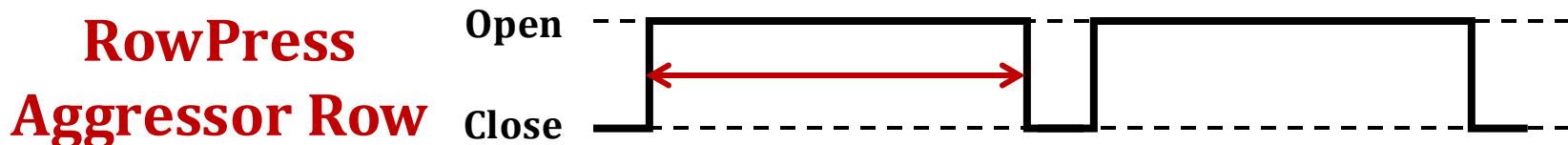
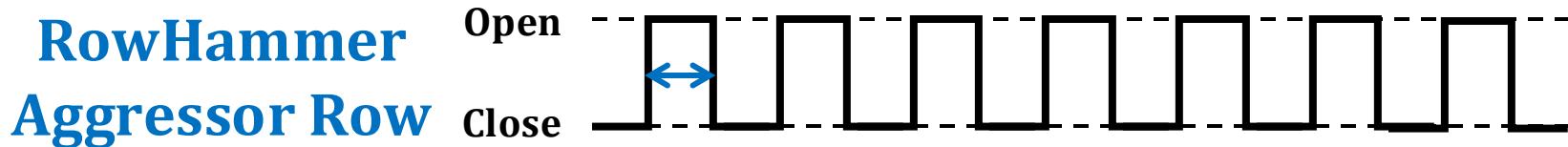
Aging can lead to read disturbance bitflips at **fewer hammers**



Two Prime Examples of DRAM Read Disturbance: RowHammer and RowPress

Instead of using a high activation count,

- ☛ increase the time that the aggressor row stays open



RowPress [ISCA 2023]

- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,
"RowPress: Amplifying Read Disturbance in Modern DRAM Chips"

Proceedings of the 50th International Symposium on Computer Architecture (ISCA), Orlando, FL, USA, June 2023.

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

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

[[Lightning Talk Video](#) (3 minutes)]

[[RowPress Source Code and Datasets](#) ([Officially Artifact Evaluated with All Badges](#))]

*Officially artifact evaluated as available, reusable and reliable.
Best artifact award at ISCA 2023.*

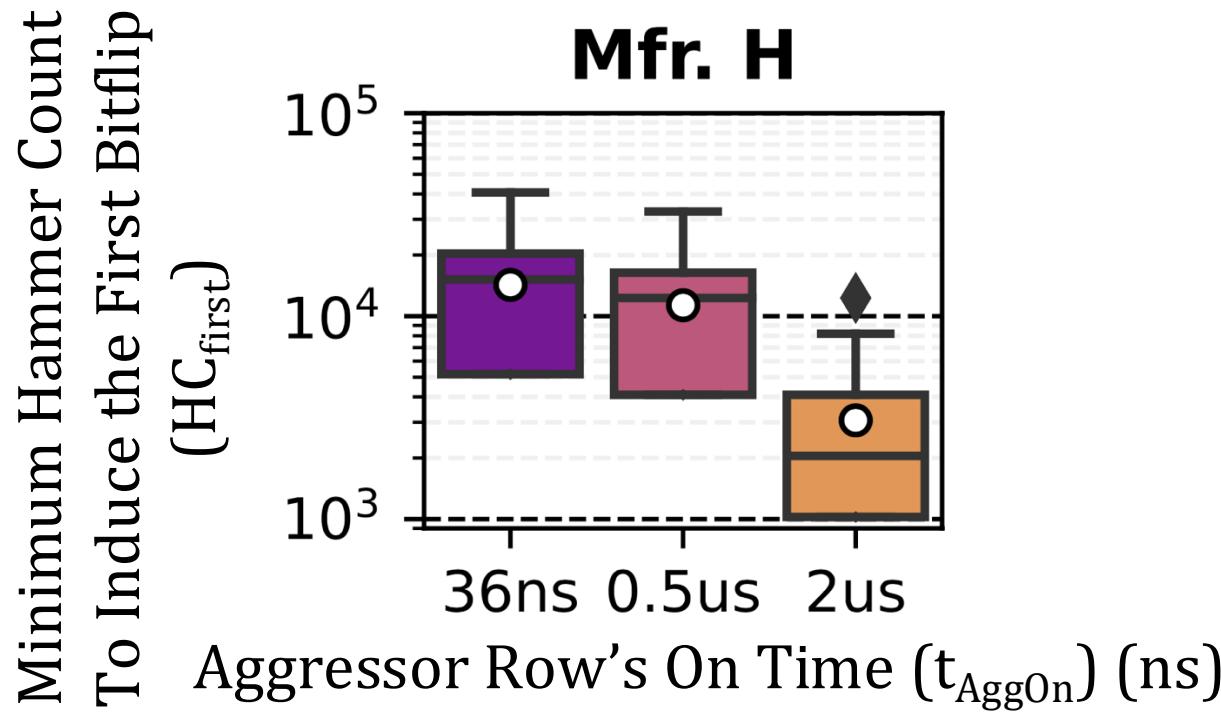


RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

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

ETH Zürich

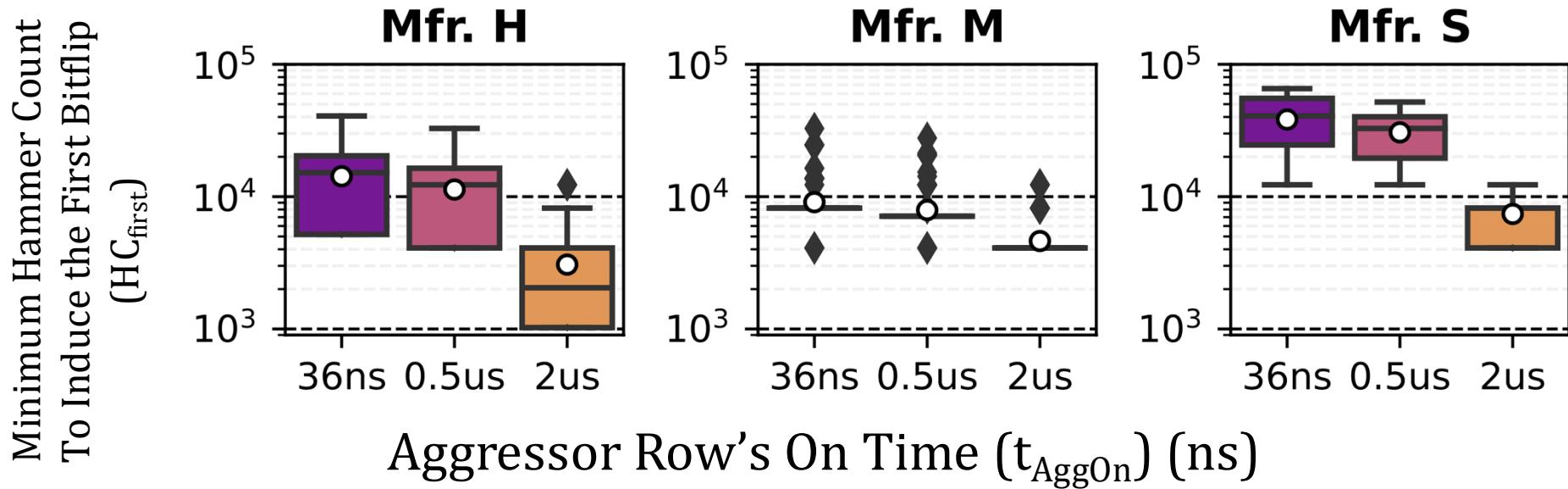
Effect of RowPress on the Spatial Variation of Read Disturbance Vulnerability across Rows



RowPress reduces the mean of the distribution with increased t_{AggOn}

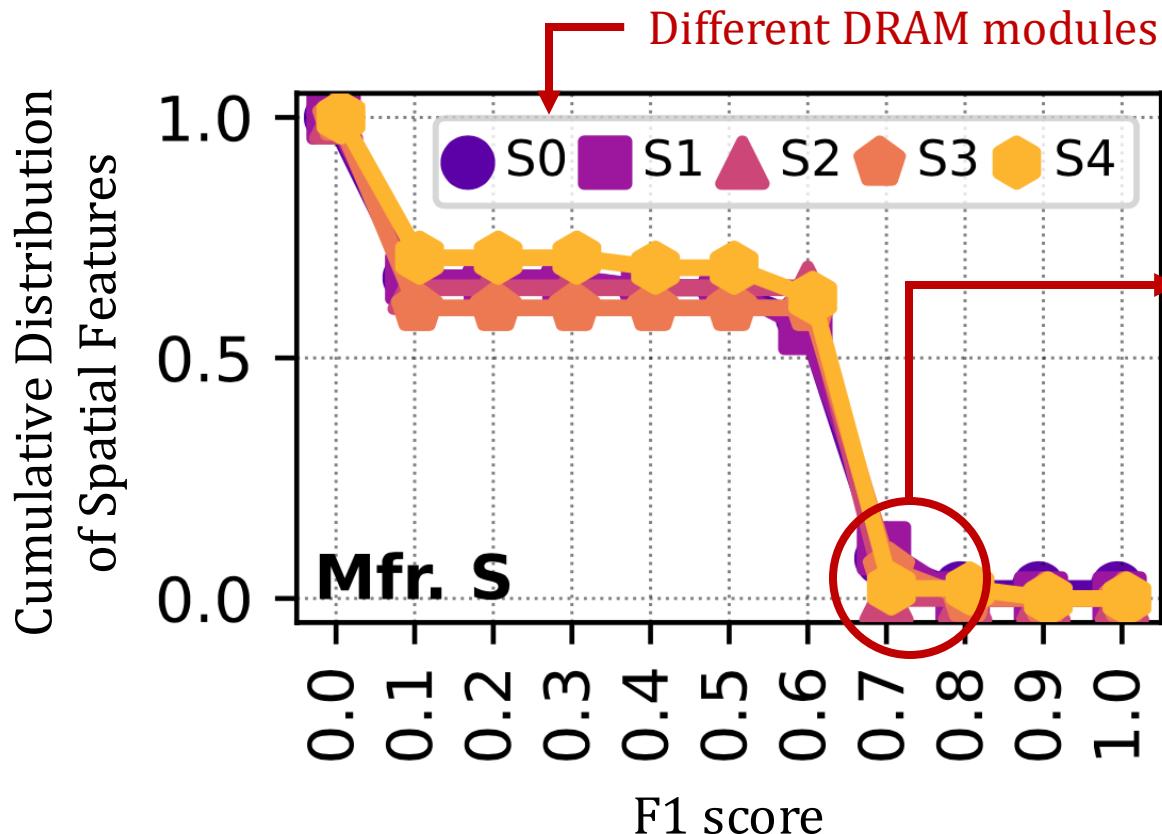
There is still significant variation in the HC_{first} across rows

Effect of RowPress on the HCfirst Distribution



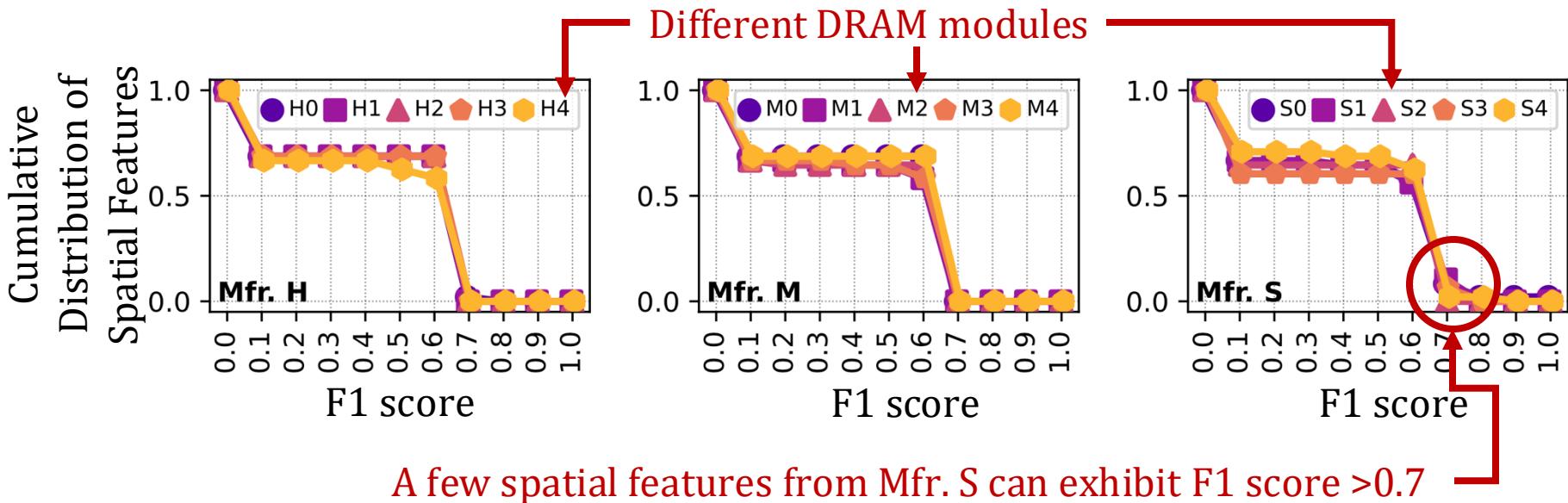
Bitflips occur at **lower hammer counts** with RowPress and these hammer counts still **significantly vary** across DRAM rows

Cumulative Distribution of Spatial Features based on F1 Score



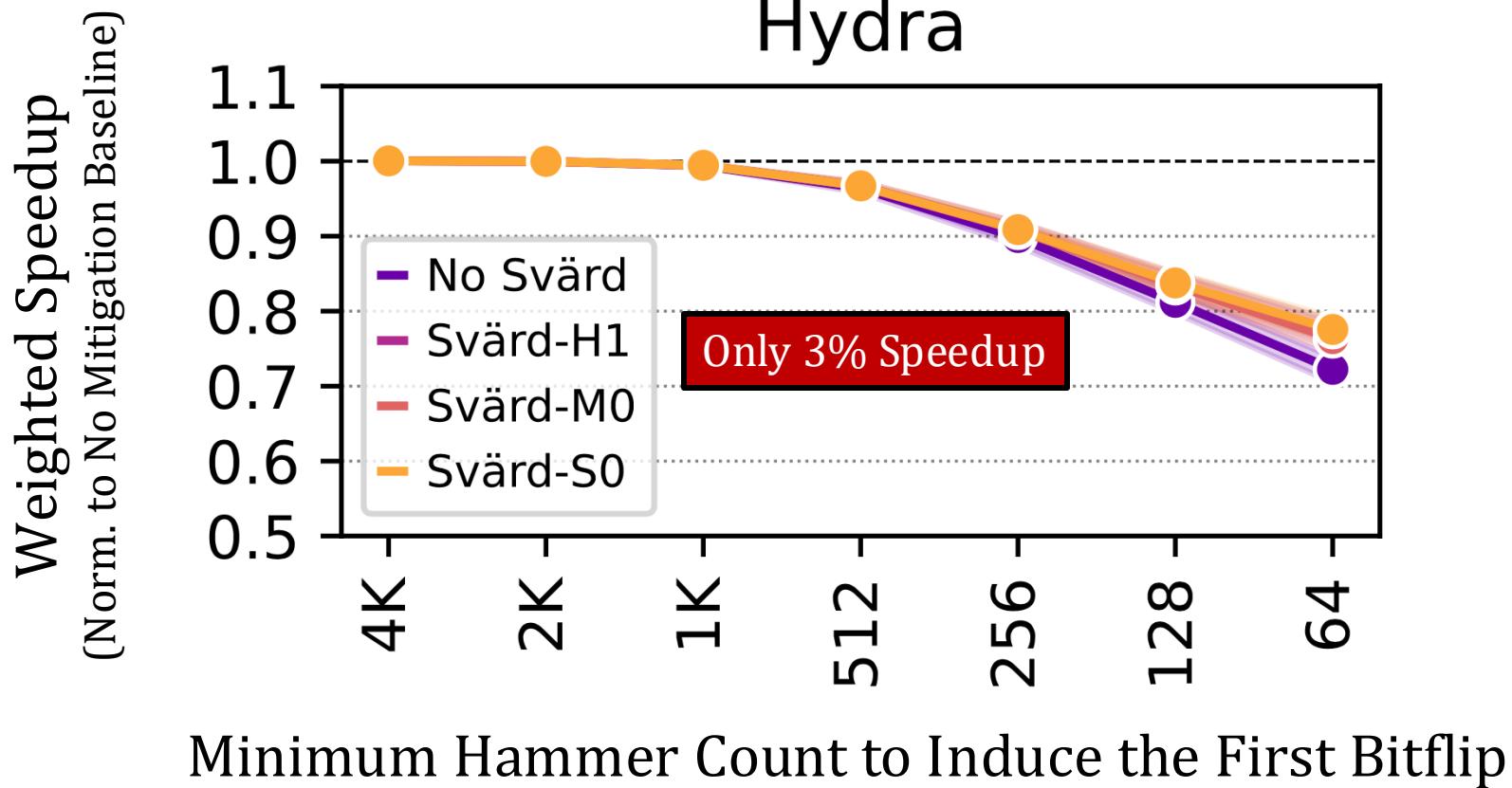
No good prediction observed between a row's **spatial features** & read disturbance **vulnerability**

Cumulative Distribution of Spatial Features based on F1 Score



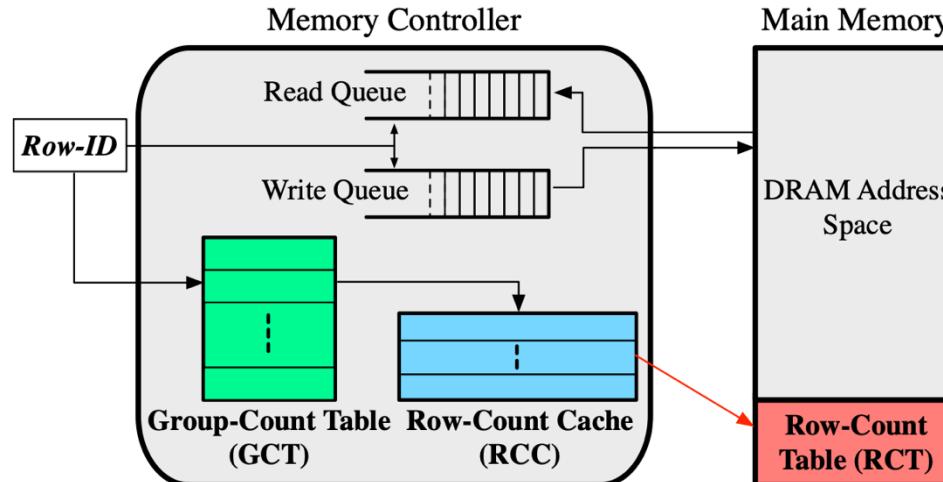
No good prediction observed between a row's
spatial features & read disturbance **vulnerability**

An Outlier Solution: Hydra



Hydra Mitigation Mechanism

- Hydra maintains **a row activation counter for each DRAM row**
- **Stores** these activation counters in the DRAM array
- **Caches** the counters of hot rows in the memory controller
- At low HC_{first} configurations, **many rows are hot**
- **Fetching / evicting counters** dominate the performance overhead
- Svärd needs **further customizations** for Hydra



Conclusion

Experimental study: 136 DDR4 DRAM chips from 3 major vendors

- A **large variation** in the necessary activation count to induce the first bitflip
- No strong correlation between a row's **spatial features** and **its vulnerability** to read disturbance

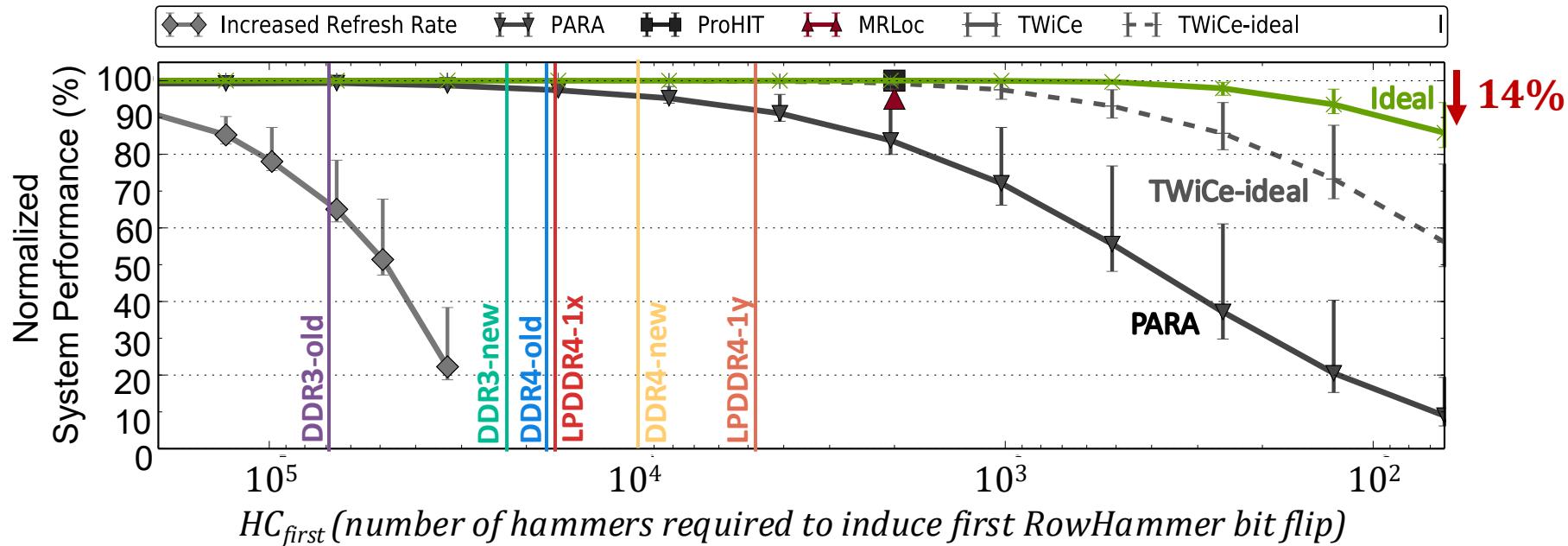
SVÄRD: Dynamically adapts the aggressiveness of mitigations

- Implemented in either **the DRAM chip** or **the memory controller**
- Reduces the performance overheads by **2.4x, 2.7%, 1.6x, and 3.0x** for **BlockHammer, Hydra, PARA, and RRS**

Future Work:

- A **deeper understanding** is needed to account for **irregularities in row and column addresses** across chips
- **Finding correlations** is essential to reduce the hardware cost
- Reducing also **the hardware cost of defenses**

RowHammer Mitigation across Generations



J. S. Kim, M. Patel, A. G. Yaglikci, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)," in *ISCA*, 2020.

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



Compatibility with Commodity DRAM Chips

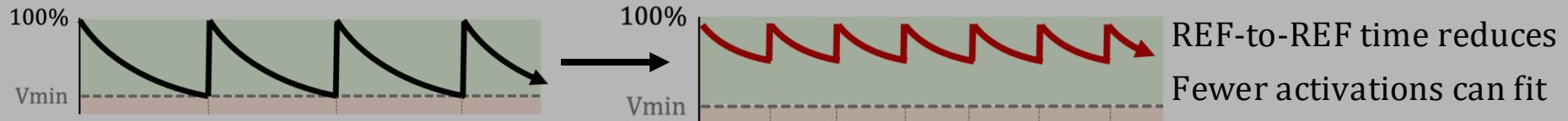
Vendors apply in-DRAM mapping for two reasons:

- **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**

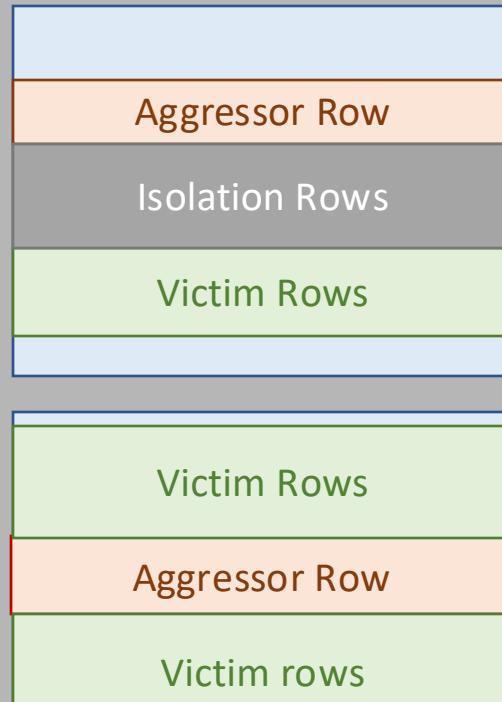
In-DRAM mapping is proprietary information

RowHammer Mitigation Approaches

- Increased refresh rate



- Physical isolation



- Reactive refresh

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

Our Goal

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

BlockHammer

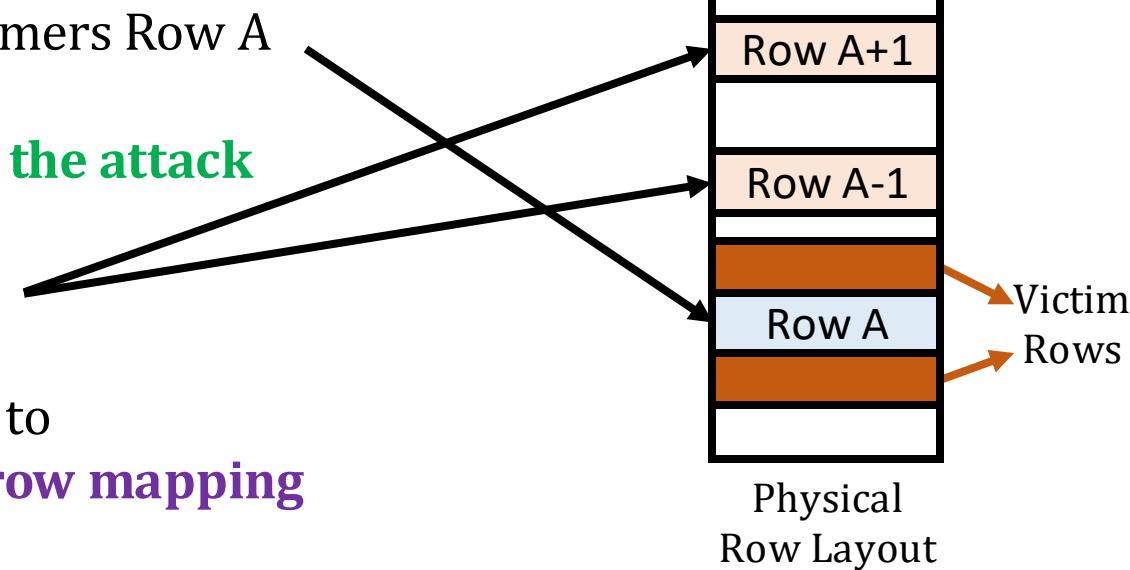
Key Idea

Selectively throttle memory accesses
that may cause RowHammer bit-flips

A Major Issue of Past Read Disturbance 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**

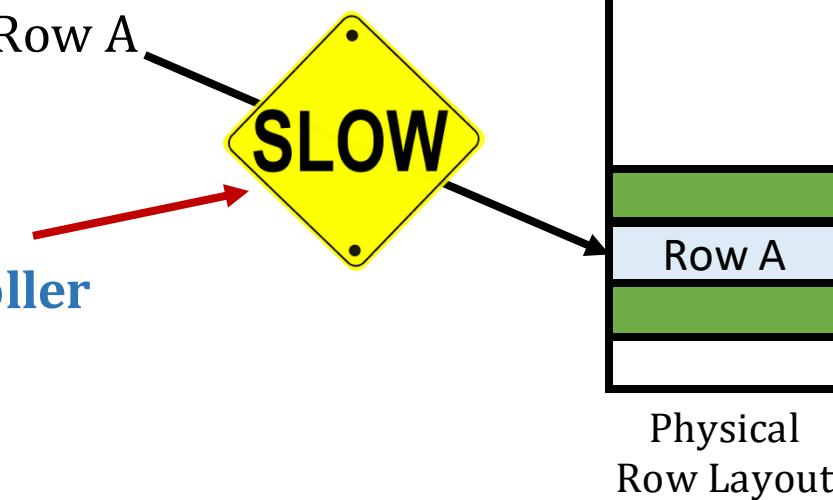


Existing **read disturbance mitigation mechanisms**
need to know **proprietary DRAM-internal row address mapping**

BlockHammer: Throttling Unsafe Accesses



- A RowHammer attack hammers Row A
- BlockHammer detects and **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

Intel Hardware Security Academic Award

Congratulations to A. Giray Yaglikci & Team!

Finalists – 2022 Intel Hardware Security Academic Award for
"BlockHammer: Preventing RowHammer at Low Cost by
Blacklisting Rapidly-Accessed DRAM Rows"



SAFARI
SAFARI Research Group



Abdullah Giray Yağlıkçı, 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](#)," in *HPCA*, 2021.

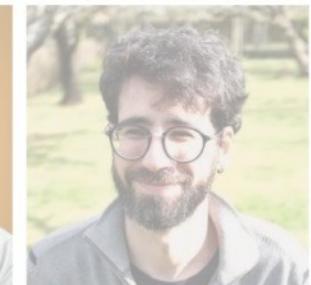
Intel Hardware Security Academic Award

Congratulations to A. Giray Yaglikci & Team!

Finalists – 2022 Intel Hardware Security Academic Award for

"BlockHammer:

Blacklisting



Abdullah Giray Yağlıkçı, 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](#)," in *HPCA*, 2021.

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

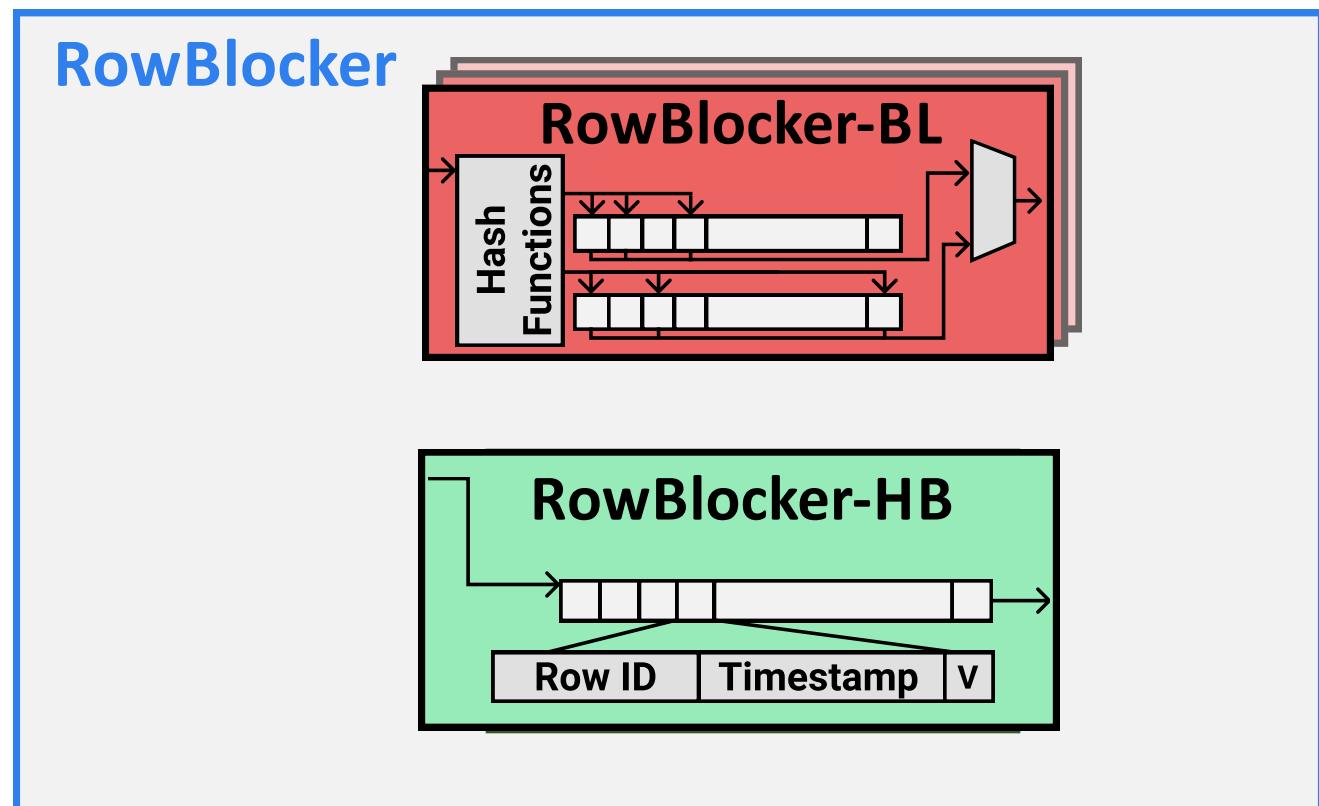
Reduces memory bandwidth usage of identified threads

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

RowBlocker

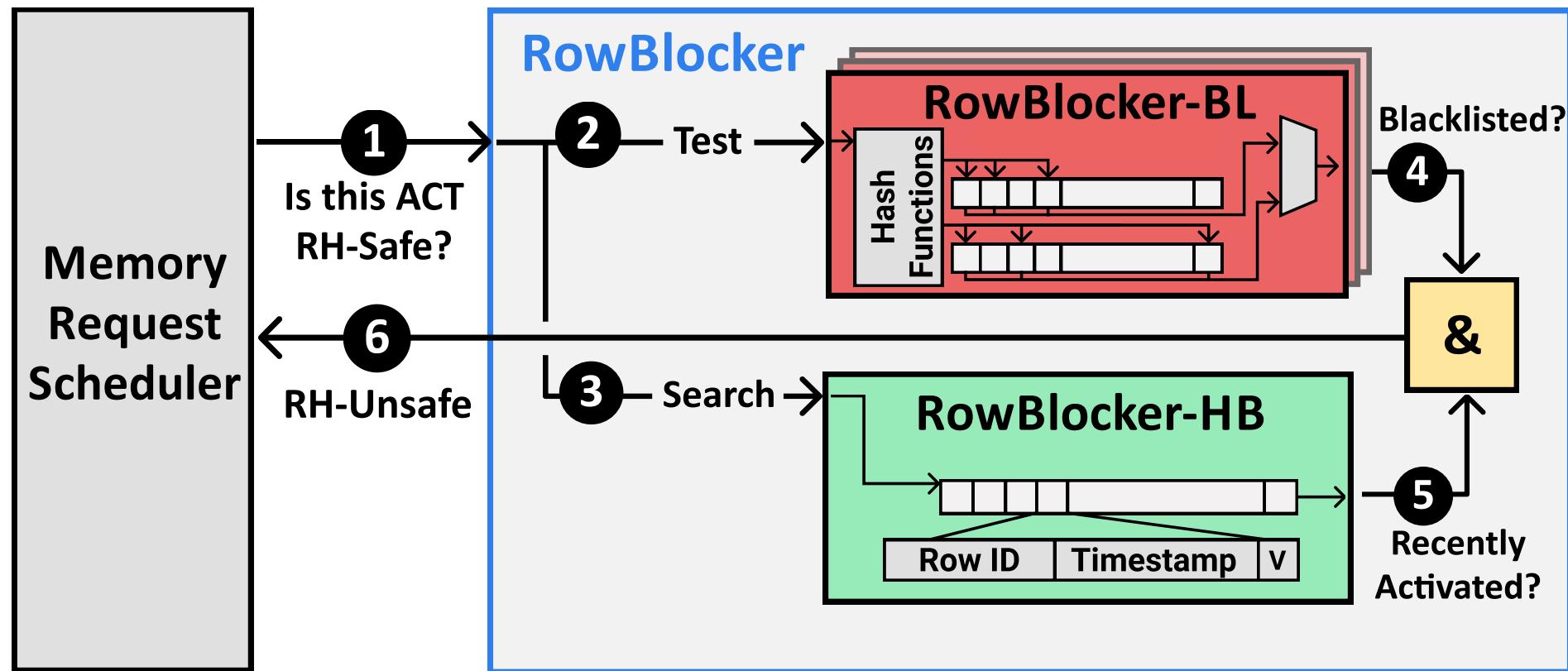
- Modifies the memory request scheduler **to throttle** row activations
- **Blacklists** rows with a high activation rate and **delays** subsequent activations targeting blacklisted rows

Memory
Request
Scheduler



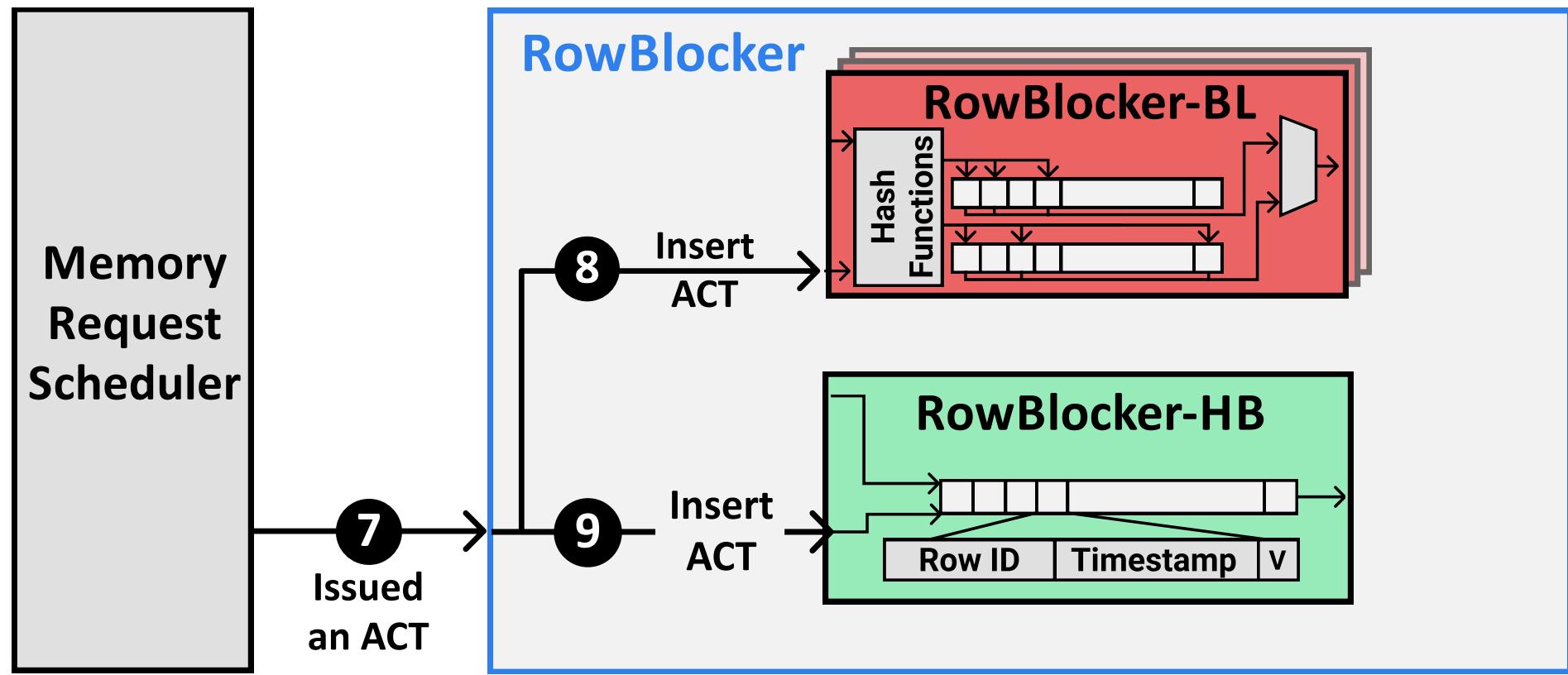
RowBlocker

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



RowBlocker

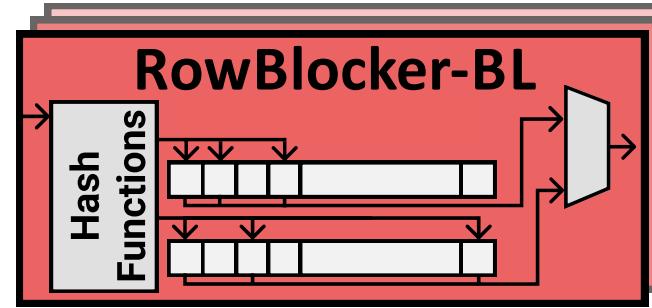
- When a row activation is performed, both **RowBlocker-BL** and **RowBlocker-HB** are updated with the row activation information



RowBlocker-BL

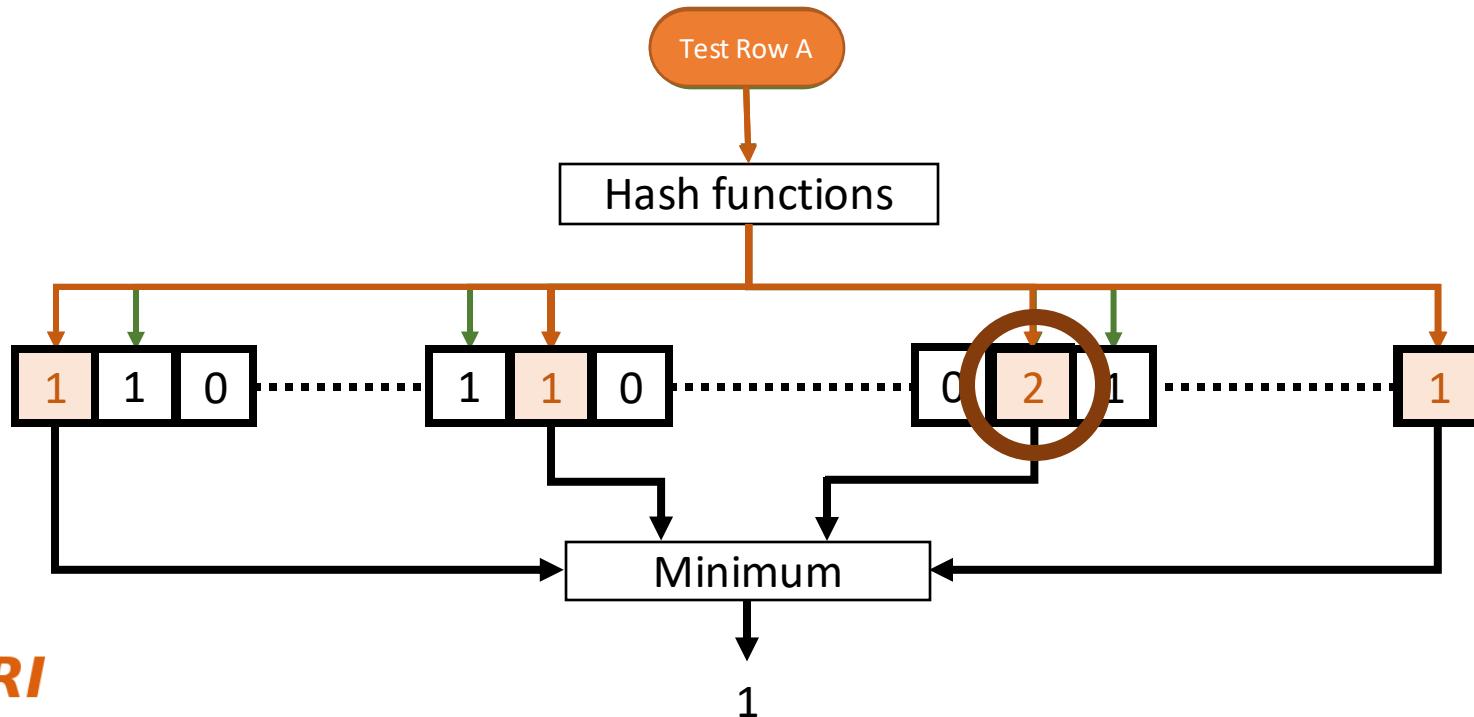
Blacklisting Logic

- **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



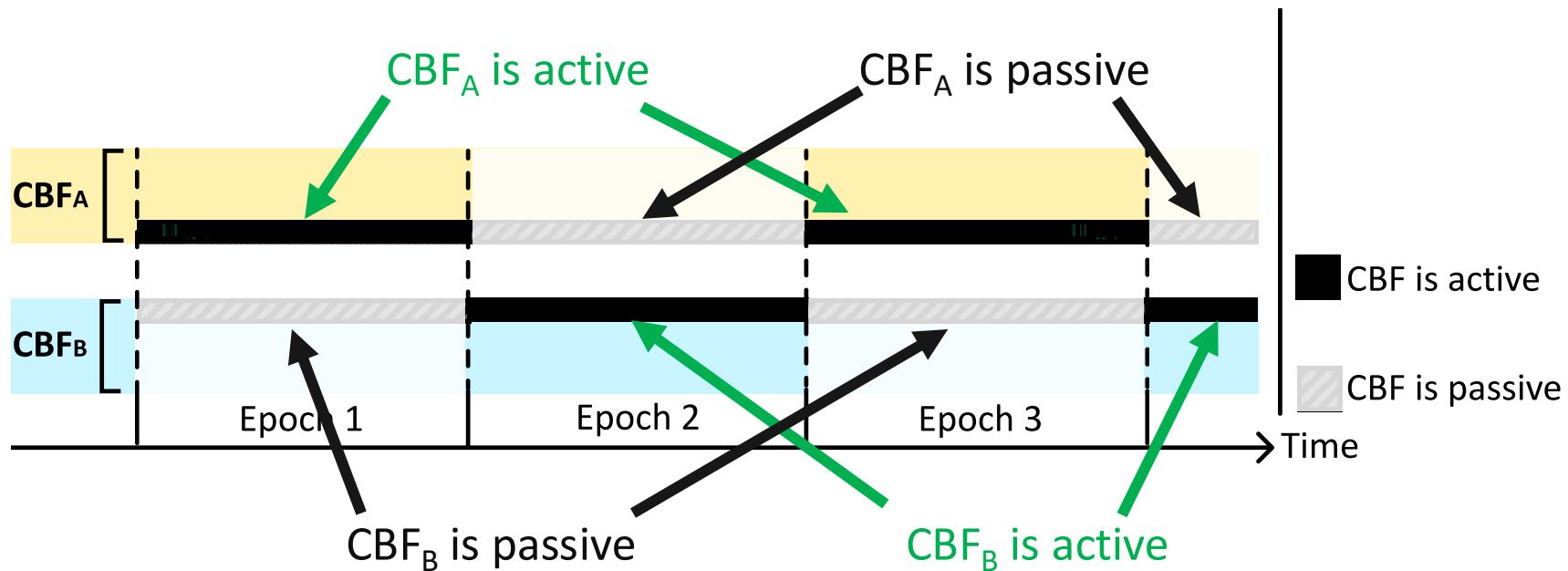
Counting Bloom Filters

- 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

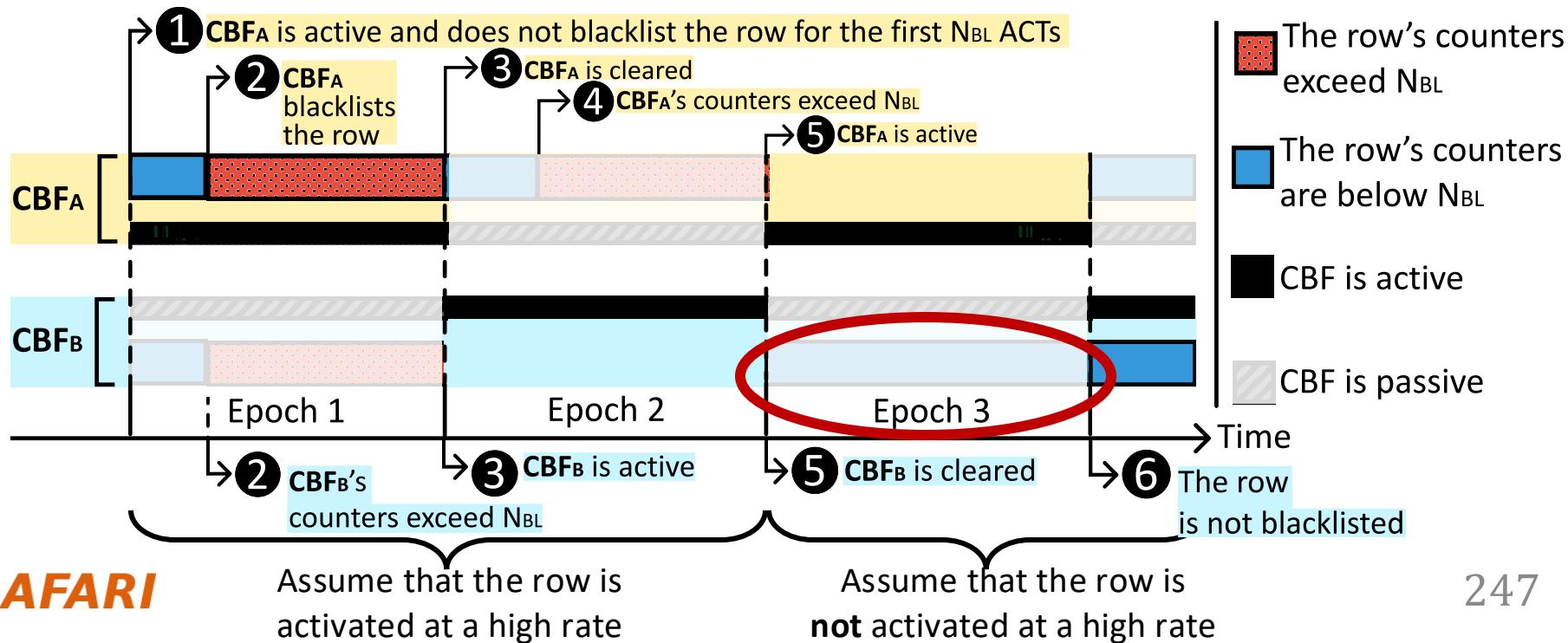
- Blacklisting logic employs **two counting Bloom filters**
- A new row activation is **inserted in both filters**
- Only one filter (**active filter**) responds to test queries
- The active filter changes at every epoch



RowBlocker-BL

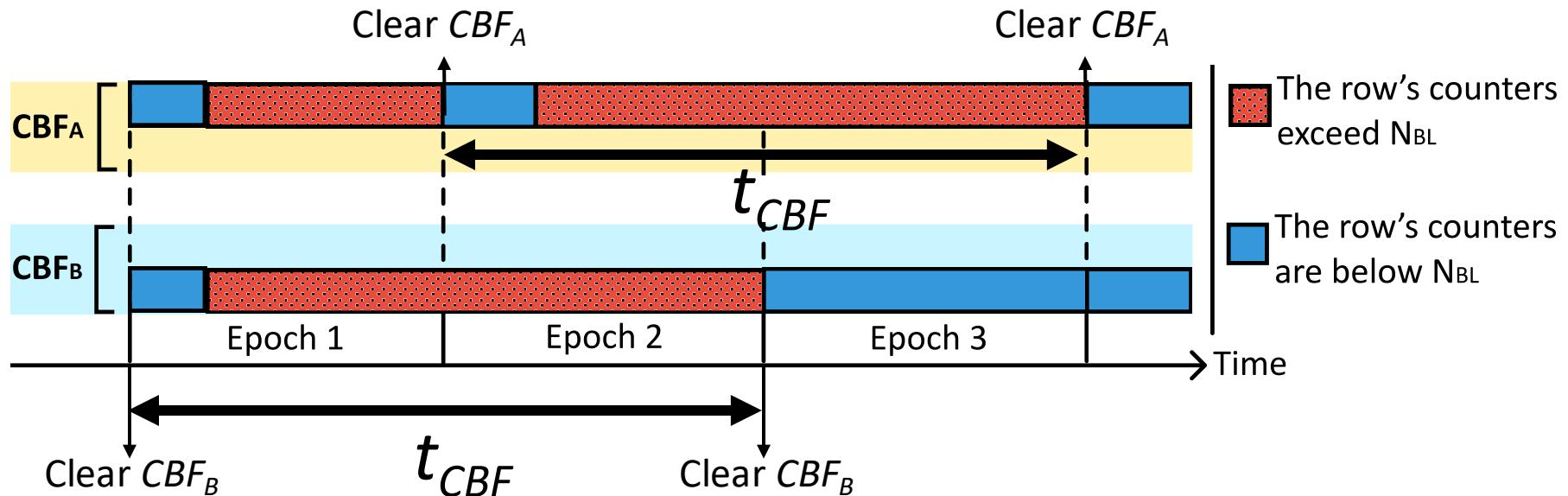
Blacklisting Logic

- 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})

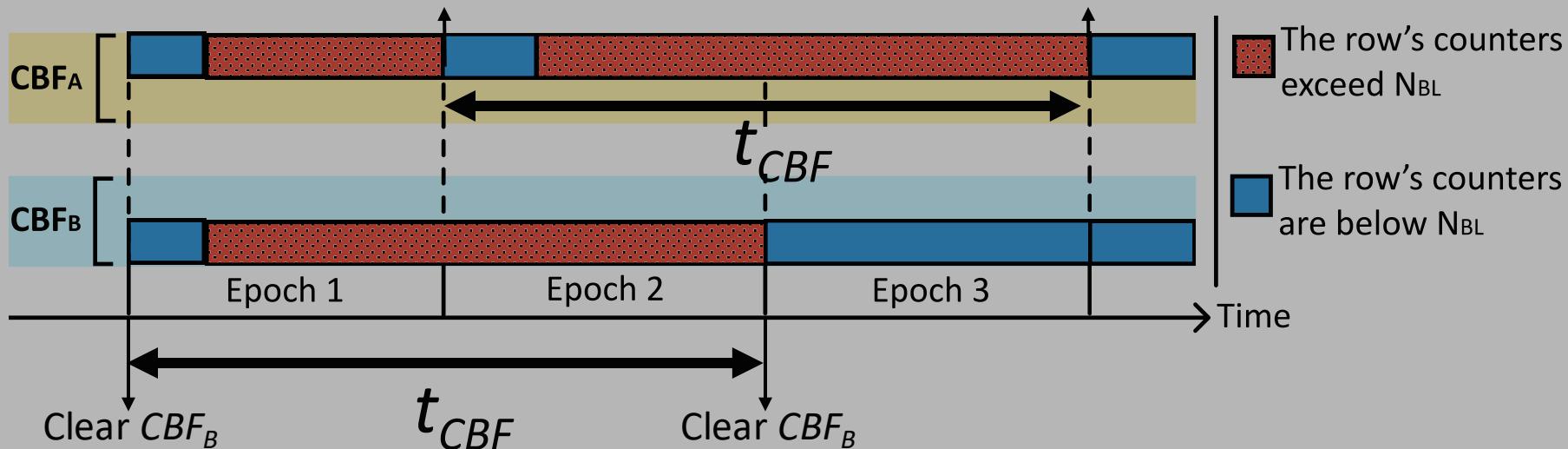


Limiting the Row Activation Rate

- 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

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

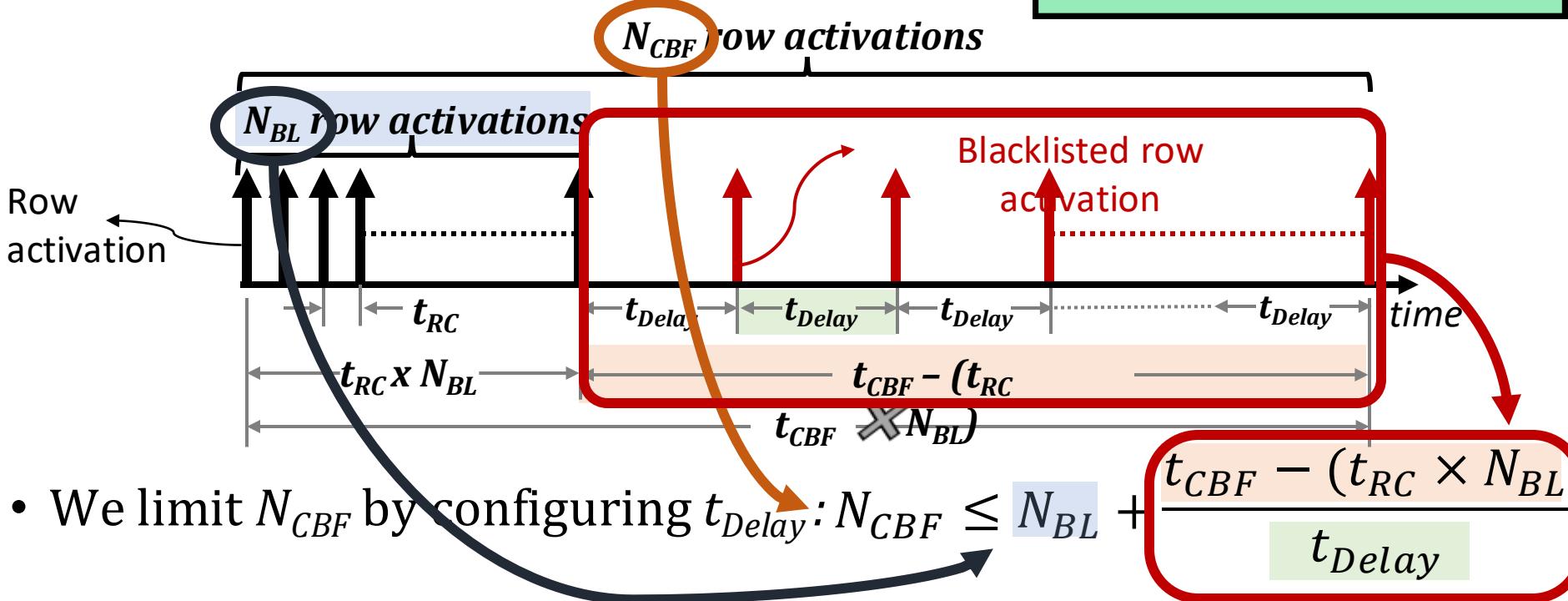


RowBlocker-HB

Limiting the Row Activation Rate

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

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



- We limit N_{CBF} by configuring t_{Delay} : $N_{CBF} \leq N_{BL} + \frac{t_{CBF} - (t_{RC} \times N_{BL})}{t_{Delay}}$

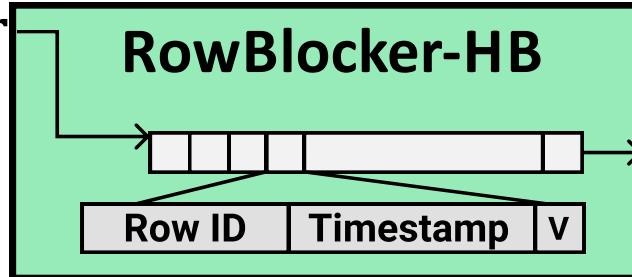
Example Adversarial of BlockHammer

- Blacklisting threshold: 512
- RowHammer threshold: 1024
- 512 activations can be performed in 2.56us
- We need to spread the next 512 activations across 64ms-2.56us
- tDelay: 125us instead of 50ns
- An adversarial can blacklist 25K rows in a refresh window and practically block the whole bank

RowBlocker-HB

Delaying Row Activations

- RowBlocker-HB ensures no subsequent blacklisted row activation is performed sooner

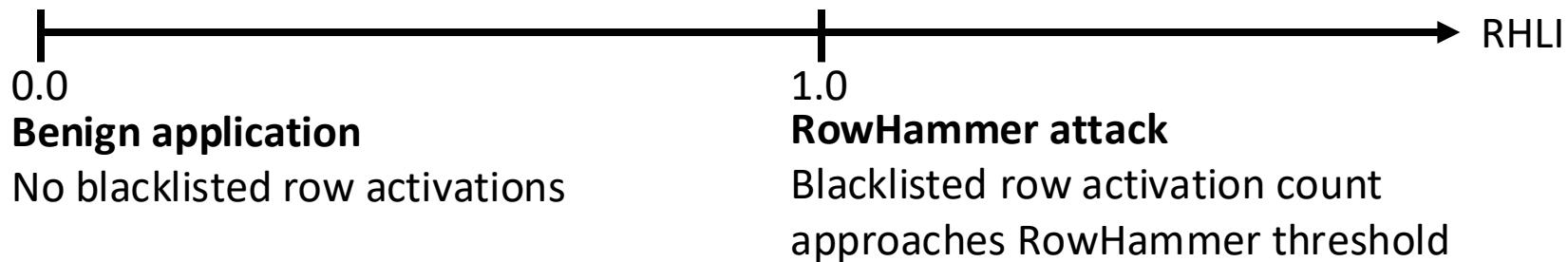


- RowBlocker-HB implements a history buffer for row activations that can fit in a t_{Delay} time window
- A blacklisted row activation is blocked as long as a valid activation record of the row exists in the history buffer

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

AttackThrottler

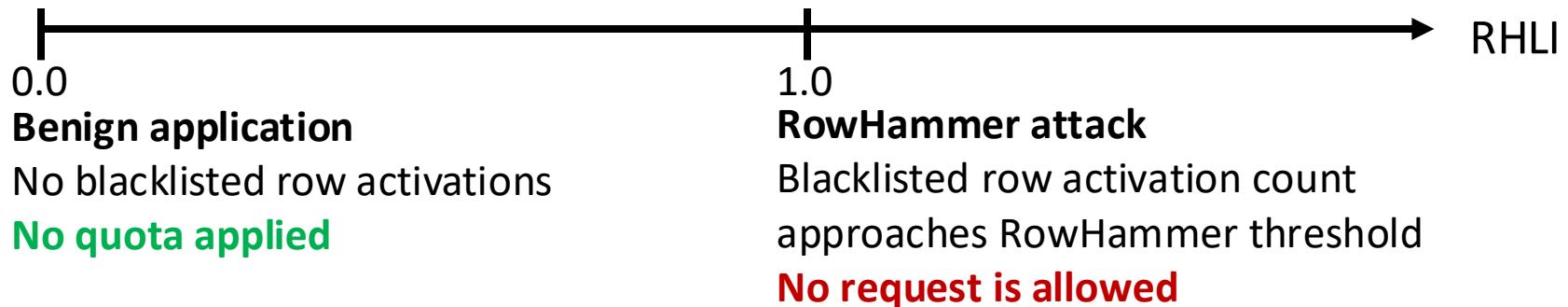
- Tackles a RowHammer attack's **performance degradation** and **energy wastage** on a system
- A RowHammer attack intrinsically keeps activating **blacklisted rows**
- **RowHammer Likelihood Index (RHLI):** Number of activations that target blacklisted rows (normalized to maximum possible activation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

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 ²	%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 ²	%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	3.6	0.1
MRLoc [161]	-	0.47	<0.01	<0.01	4.4	0.2
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	1.60	535.50
TWiCe [84]	738.32	448.27	5.17	35x	2.10	631.98
Graphene [113]	-	166.03	1.14	23x	0.46	93.96

The diagram illustrates efficiency ratios between BlockHammer and other mechanisms. Red boxes highlight ratios for Access Energy and Static Power. Green boxes highlight ratios for SRAM, CAM, Area, and %CPU.

- Access Energy:**
 - BlockHammer vs PARA [73]: 10x
 - BlockHammer vs ProHIT [137]: 5x
 - BlockHammer vs MRLoc [161]: 23x
 - BlockHammer vs CBT [132]: 20x
 - BlockHammer vs TWiCe [84]: 35x
 - BlockHammer vs Graphene [113]: 30x
- Static Power:**
 - BlockHammer vs PARA [73]: 10x
 - BlockHammer vs ProHIT [137]: 5x
 - BlockHammer vs MRLoc [161]: 23x
 - BlockHammer vs CBT [132]: 15x
 - BlockHammer vs TWiCe [84]: 30x
 - BlockHammer vs Graphene [113]: 30x
- SRAM, CAM, Area, %CPU:**
 - BlockHammer vs PARA [73]: -
 - BlockHammer vs ProHIT [137]: 10x
 - BlockHammer vs MRLoc [161]: 5x
 - BlockHammer vs CBT [132]: 20x
 - BlockHammer vs TWiCe [84]: 35x
 - BlockHammer vs Graphene [113]: 23x

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:

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

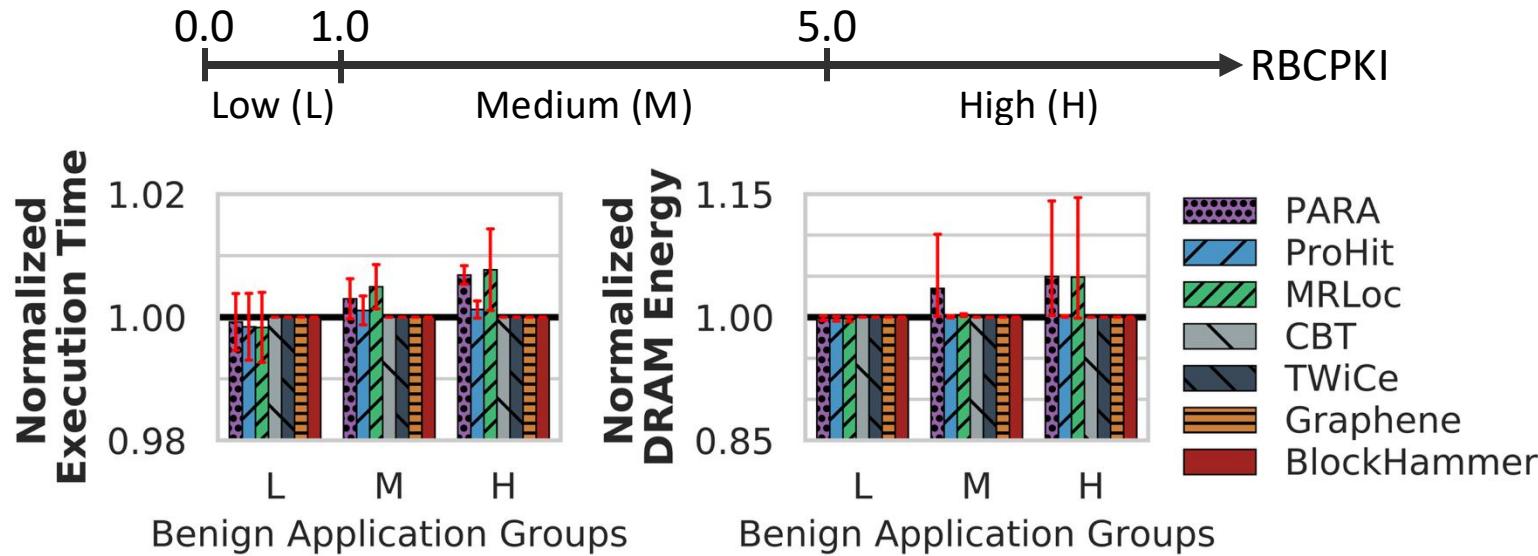
- Randomly Chosen Multiprogrammed Workloads:

- 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})

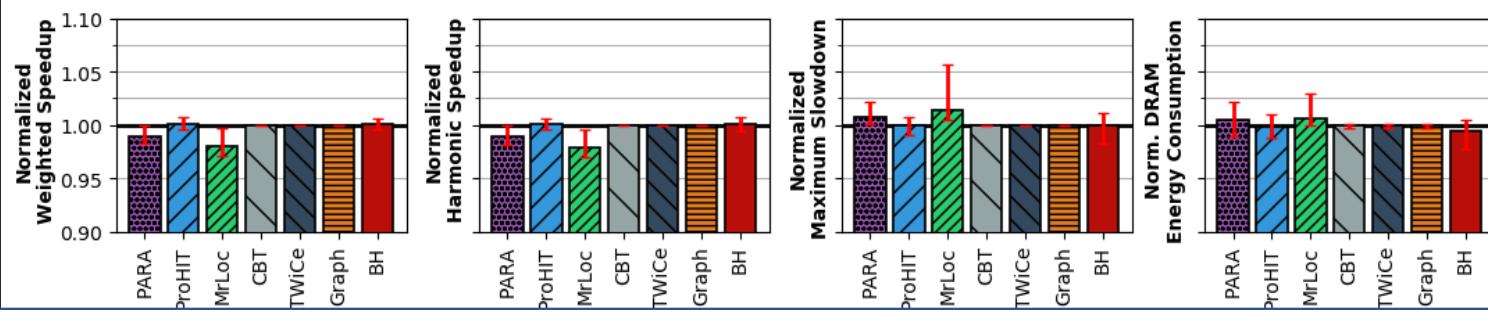
BlockHammer does not incur **performance** or **DRAM energy** overheads for single-core benign applications

Evaluation

Performance and DRAM Energy

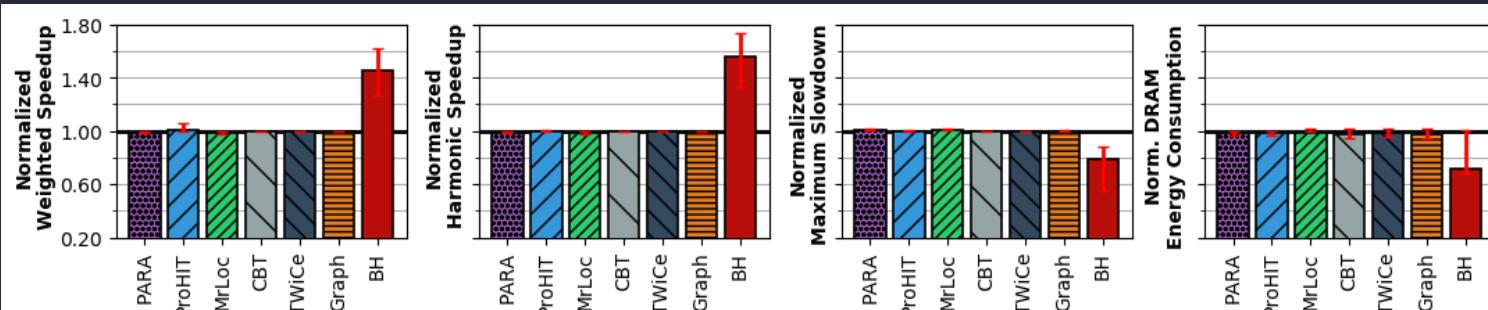
- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

No RowHammer Attack



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

RowHammer Attack Present

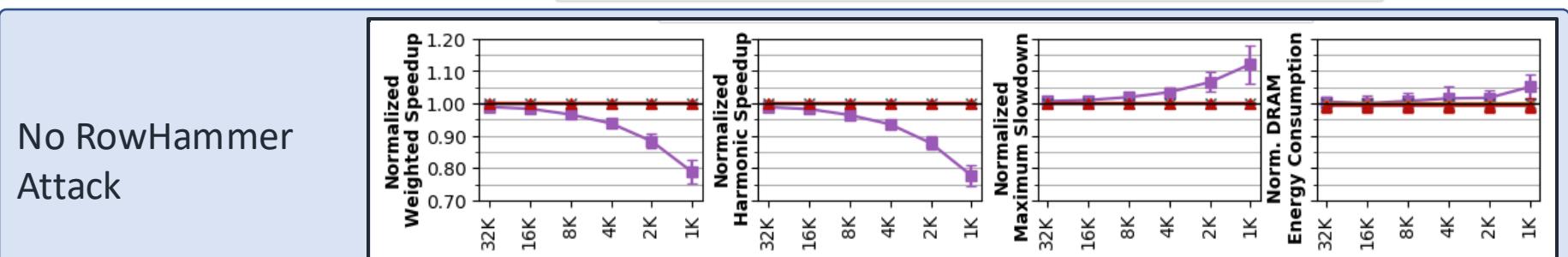


BlockHammer **significantly increases** benign application performance (by 45% on average) and **reduces** DRAM energy consumption (by 29% on average)

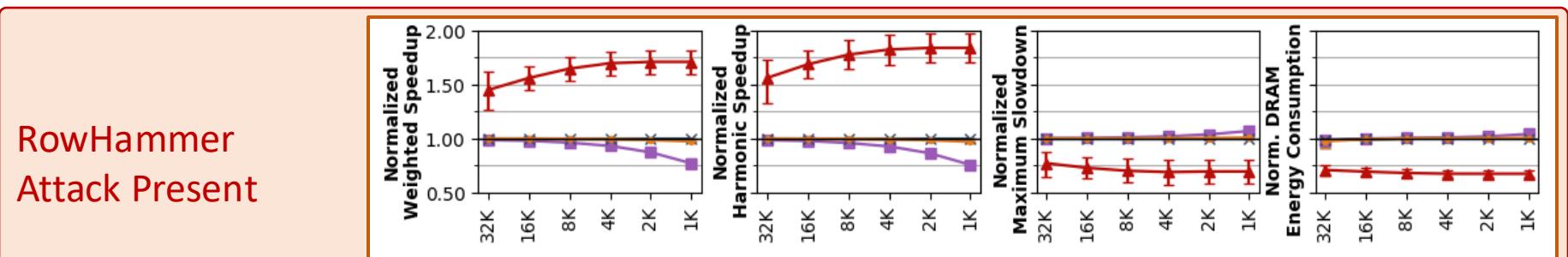
Evaluation

Scaling with RowHammer Vulnerability

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



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



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] BlockHammer	✓ ✓ ✓	✓ ✗ ✓	✗ ✗ ✓	✓ ✓ ✓

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.

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

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

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

HiRA: Hidden Row Activation

for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yağlıkçı

Ataberk Olgun Minesh Patel Haocong Luo Hasan Hassan

Lois Orosa Oğuz Ergin Onur Mutlu

SAFARI

ETH zürich



CESGA

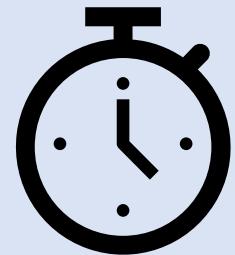


TOBB ETÜ
University of Economics & Technology

Two Main Types of DRAM Refresh

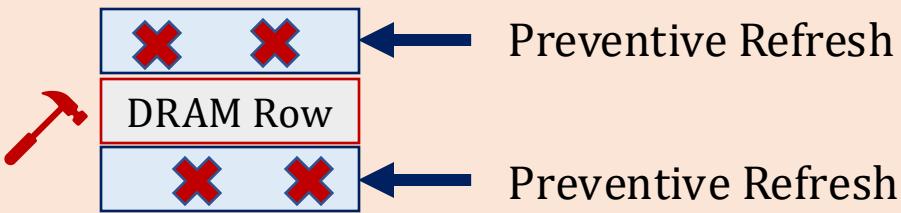
1

Periodic Refresh: Periodically **restores** the charge
DRAM cells leak **over time**



2

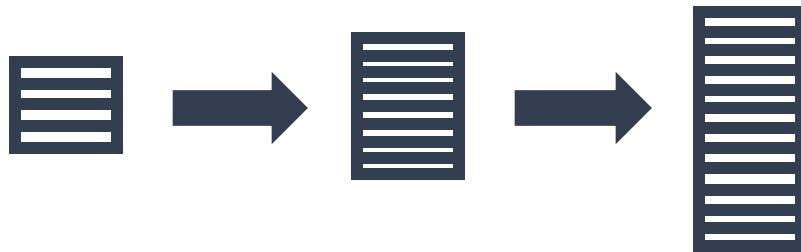
RowHammer: Repeatedly accessing a DRAM row can cause
bit flips in other **physically nearby rows**



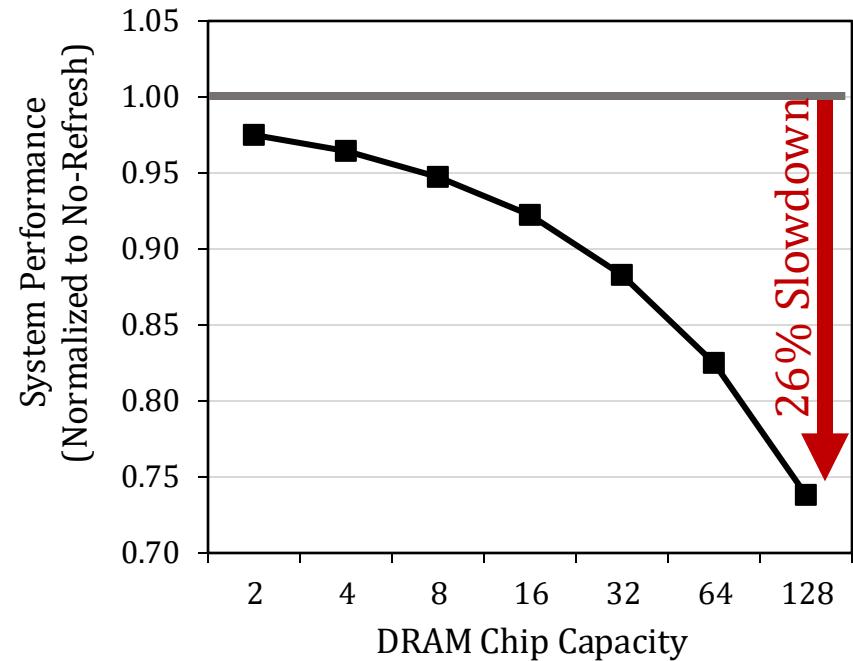
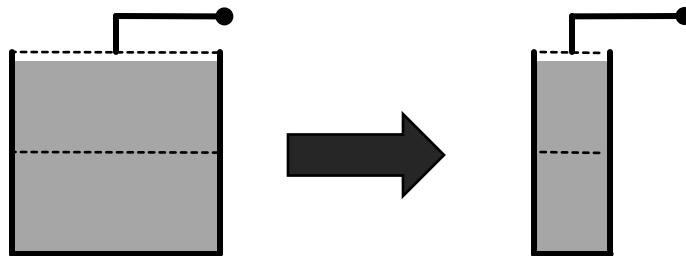
Preventive Refresh: Mitigates RowHammer
by **refreshing physically nearby rows**
of a repeatedly accessed row

Periodic Refresh with Increasing DRAM Chip Density

A **larger capacity** chip has **more rows to be refreshed**



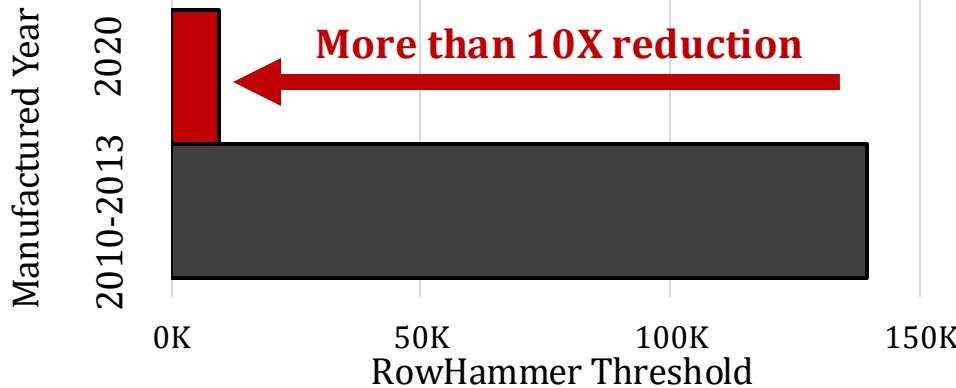
A **smaller** cell stores **less charge**



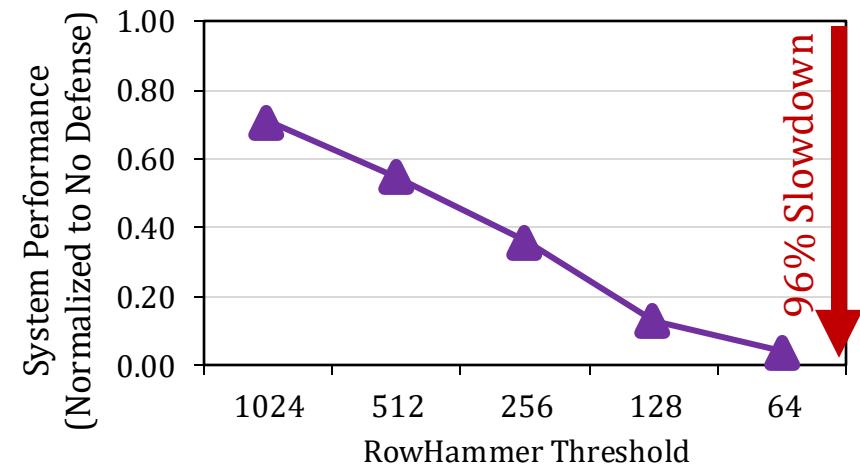
More periodic refresh operations incur
larger performance overhead as DRAM **chip density increases**

RowHammer and Preventive Refresh with Increasing DRAM Chip Density

RowHammer vulnerability worsens
as DRAM chip density increases



The Minimum Activation Count needed
to induce the first RowHammer bit flip

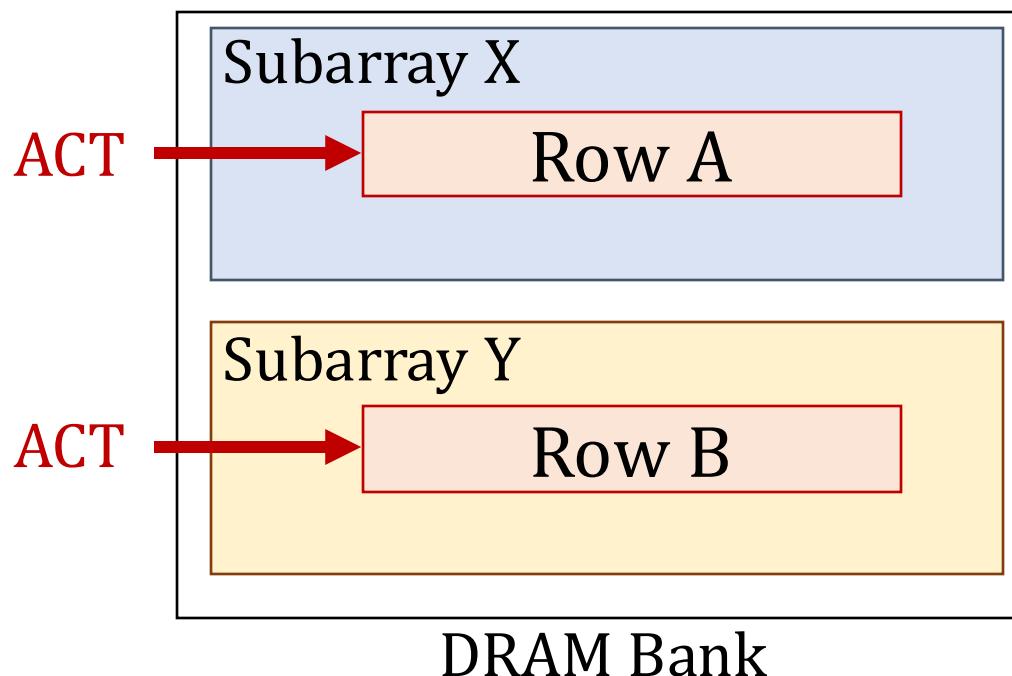


The Minimum Activation Count needed
to induce the first RowHammer bit flip

Preventive refresh operations need to be performed
more aggressively as DRAM chip density increases

HiRA: Hidden Row Activation – Key Insight

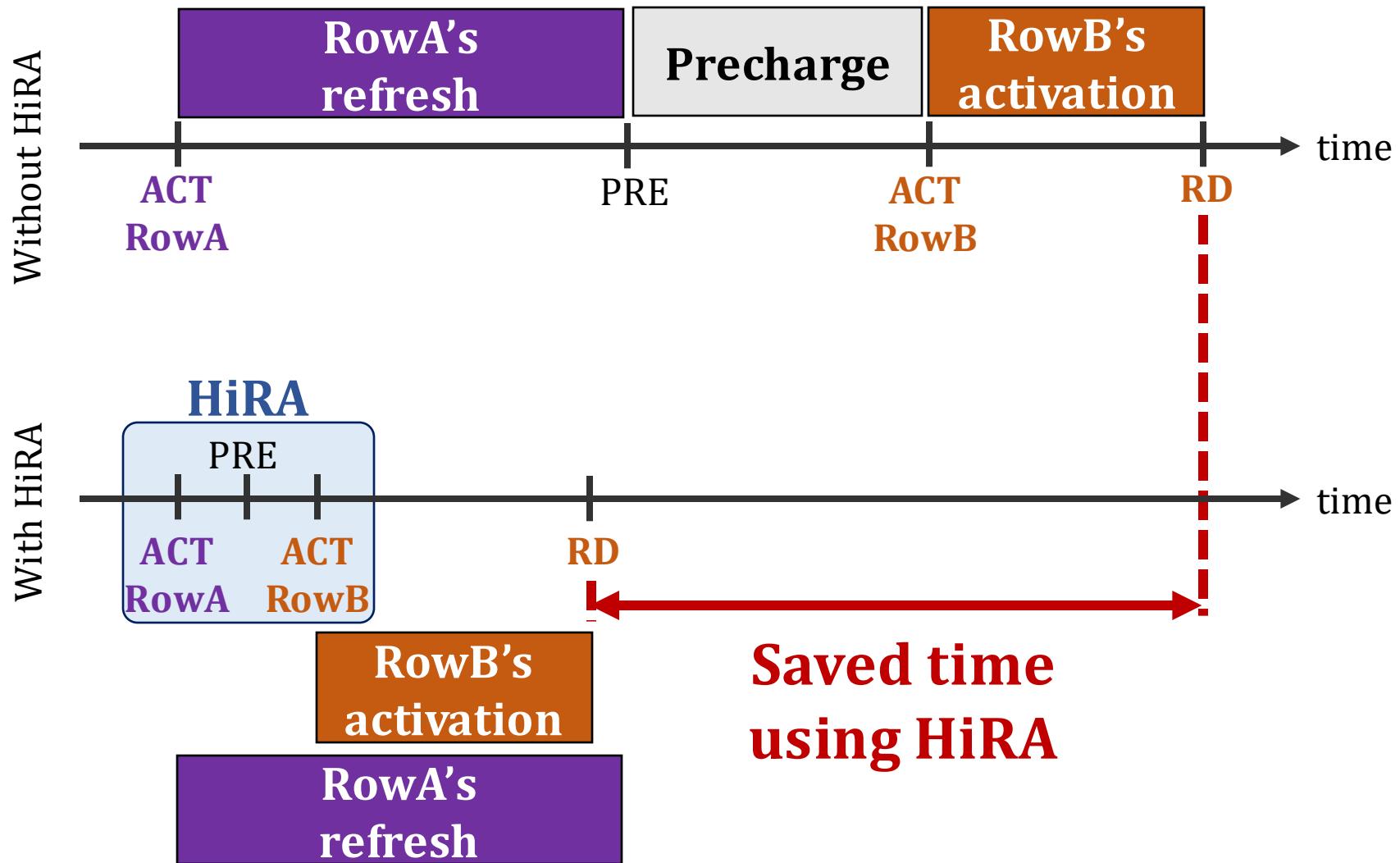
Activating two rows in **quick succession** that are in **different subarrays** in the **same bank** can **refresh one row** concurrently with **activating the other row**



Refreshes RowA
concurrently with
Activating RowB

HiRA: Hidden Row Activation

Refresh RowA concurrently with Activating RowB



HiRA in Off-the-Shelf DRAM Chips: Key Result 1

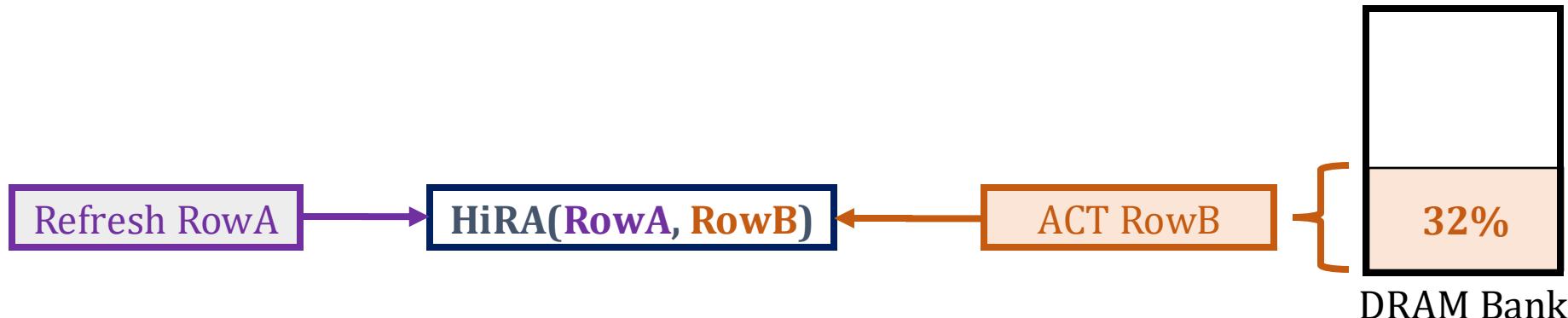
- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

Table 4: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage Min.	HiRA Coverage Avg.	HiRA Coverage Max.	Norm. N_{RH} Min.	Norm. N_{RH} Avg.	Norm. N_{RH} Max.
A0	G.SKILL	DWCW (Partial Marking)* F4-2400C17S-8GNT [39]	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1								24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC KSM32RD8/16HDR [87]	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1								25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN HMAA4GU6AJR8N-XN [109]	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1								29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

- HiRA performs a given row's **refresh concurrently with activating** any of the **32% of the rows** in the same bank



HiRA in Off-the-Shelf DRAM Chips: Key Result 2

- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

Table 4: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage			Norm. N_{RH}		
								Min.	Avg.	Max.	Min.	Avg.	Max.
A0	G.SKILL	DWCW (Partial Marking)* F4-2400C17S-8GNT [39]	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1								24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC KSM32RD8/16HDR [87]	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1								25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN HMAA4GU6AJR8N-XN [109]	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1								29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

- **51.4% reduction** in the time spent for refresh operations

HiRA **effectively reduces the time spent**
for **refresh** operations in **off-the-shelf** DRAM chips

HiRA in Off-the-Shelf DRAM Chips: Key

Results

- HiR
- 51.4

• HiR
acti

HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı¹ Ataberk Olgun¹ Minesh Patel¹ Haocong Luo¹ Hasan Hassan¹

Lois Orosa^{1,3} Oğuz Ergin² Onur Mutlu¹

¹ETH Zürich ²TOBB University of Economics and Technology ³Galicia Supercomputing Center (CESGA)

DRAM is the building block of modern main memory systems. DRAM cells must be periodically refreshed to prevent data loss. Refresh operations degrade system performance by interfering with memory accesses. As DRAM chip density increases with technology node scaling, refresh operations also increase because: 1) the number of DRAM rows in a chip increases; and 2) DRAM cells need additional refresh operations to mitigate bit failures caused by RowHammer, a failure mechanism that becomes worse with technology node scaling. Thus, it is critical to enable refresh operations at low performance overhead. To this end, we propose a new operation, Hidden Row Activation (HiRA), and the HiRA Memory Controller (HiRA-MC) to perform HiRA operations.

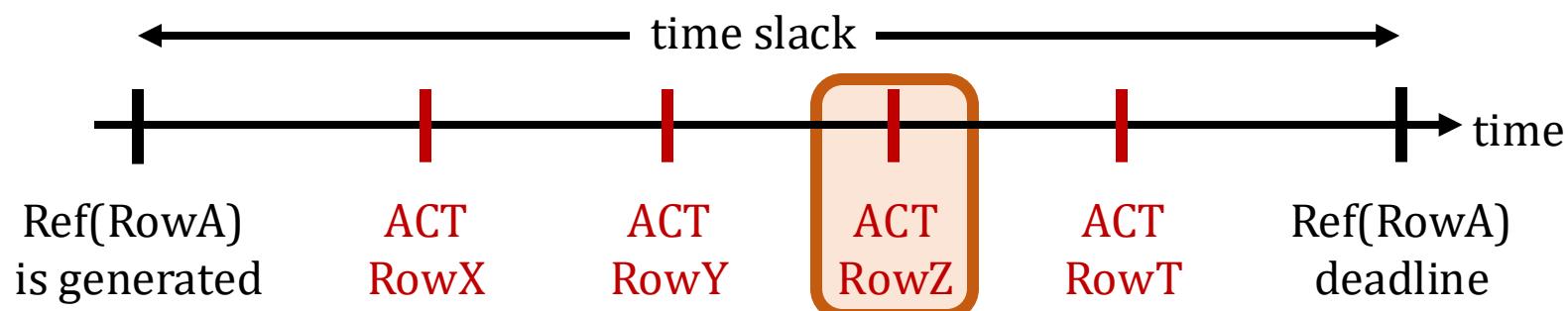
As DRAM density increases with technology node scaling, the performance overhead of refresh also increases due to three major reasons. First, as the DRAM chip density increases, more DRAM rows need to be periodically refreshed in a DRAM chip [55, 57–61]. Second, as DRAM technology node scales down, DRAM cells become smaller and thus can store less amount of charge, requiring them to be refreshed more frequently [10, 20, 67, 102, 103, 118, 122–124]. Third, with increasing DRAM density, DRAM cells are placed closer to each other, exacerbating charge leakage via a disturbance error mechanism called RowHammer [79, 84, 119, 120, 133, 134, 167, 180, 183], and thus requiring additional refresh operations (called *preventive* refreshes) to avoid data corruption due to RowHam-

<https://arxiv.org/pdf/2209.10198.pdf>

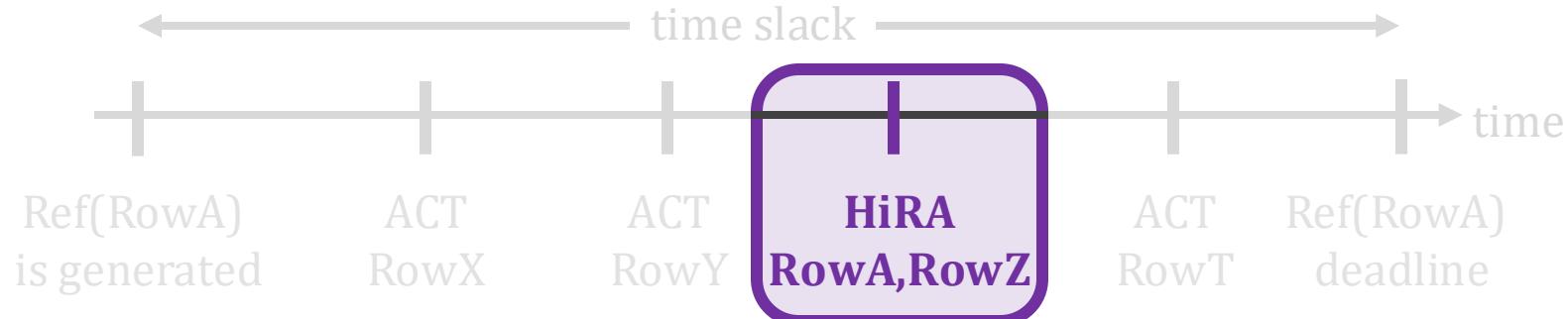


HiRA-MC: HiRA Memory Controller

- **Goal:** Leverage HiRA's parallelism as much as possible
- **Key Insight:** A time slack is needed to find a **row activation** and a **refresh** to perform HiRA



RowA and RowZ are in two electrically disconnected subarrays



HiRA-MC: HiRA Memory Controller

1

Generates each **periodic refresh** and **RowHammer-preventive refresh with a deadline**

2

Buffers each **refresh request** and **performs** the refresh request **until** the **deadline**

3

Finds if it can **refresh a DRAM row** concurrently with a **DRAM access** or **another refresh**

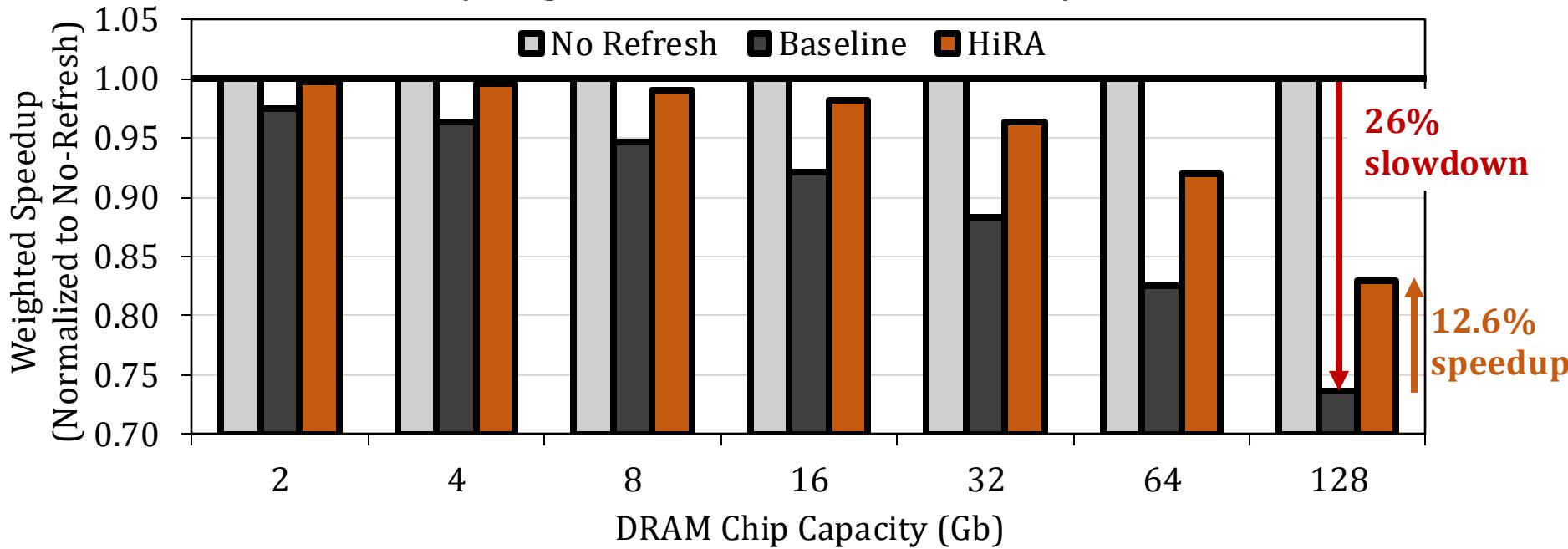
Performance Evaluation

- Cycle-level simulations using **Ramulator** [Kim+, CAL 2015]
- **System Configuration:**

Processor	3.2 GHz, 8 core, 4-wide issue, 128-entry instr. window
Last-Level Cache	64-byte cache line, 8-way set-associative, 8 MB
Memory Scheduler	FR-FCFS
Address Mapping	Minimalistic Open Pages
Main Memory	DDR4, 4 bank group, 4 banks per bank group (16 banks per rank)
Timing Parameters	$t_1=t_2=3\text{ns}$, $t_{RC}=46.25\text{ns}$, $t_{FAW}=16\text{ns}$
- **Workloads:** 125 different **8-core** multiprogrammed workloads from the SPEC2006 benchmark suite
- **DRAM Chip Capacity:** {2, 4, 8, 16, 32, 64, 128} Gb
- **RowHammer Threshold:** {1024, 512, 256, 128, 64} activations
The minimum number of row activations needed to induce the first RowHammer bit flip

HiRA for Periodic Refreshes

- **No-Refresh:** No periodic refresh is performed (Ideal case)
- **Baseline:** Auto-Refresh (using conventional REF commands)

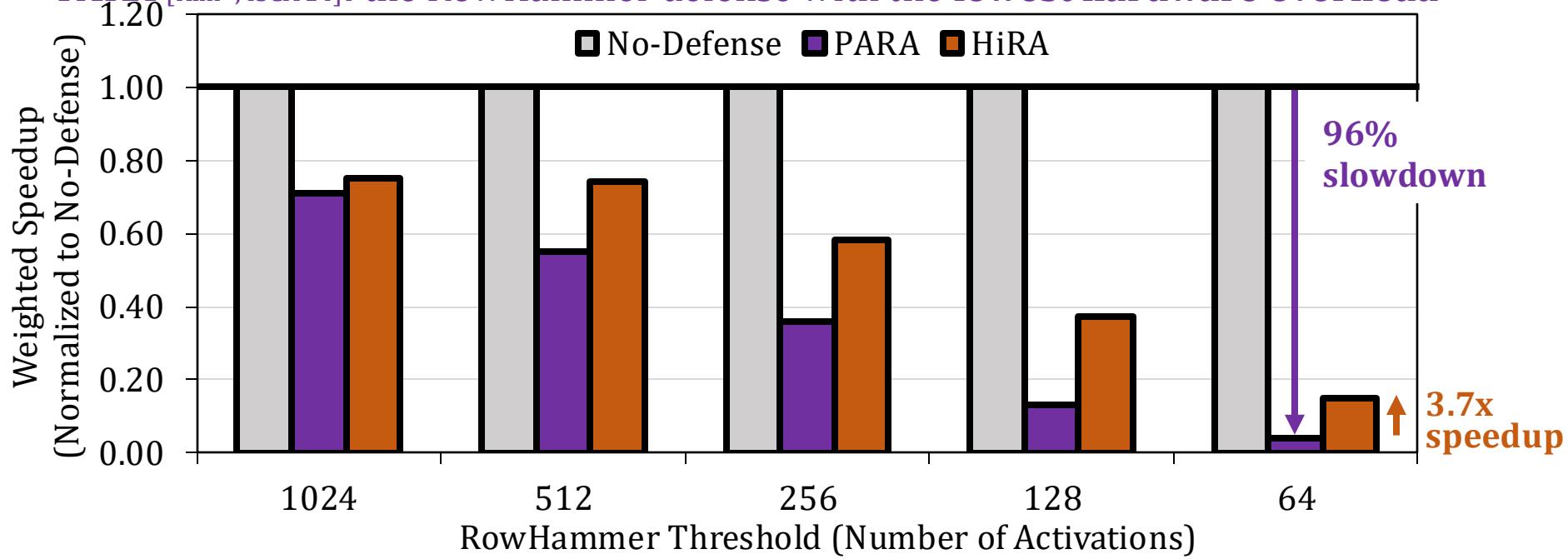


Periodic refreshes cause *significant (26%) performance overhead*

HiRA improves system performance **by 12.6%** over the baseline

HiRA for Preventive Refreshes

- **No Defense:** No RowHammer mitigation employed (i.e., no preventive refresh)
- **PARA** [Kim+, ISCA'14]: the RowHammer defense with the **lowest hardware overhead**



PARA ***significantly reduces (by 96%)*** system performance

HiRA **improves** system performance **by 3.7x** over PARA

More in the Full Paper

- **Real DRAM Chip Experiments**
 - Verification of **HiRA's functionality**
 - **Variation in HiRA's characteristics across banks**
- Sensitivity to
 - length of **time slack** for refreshes
 - **number of channels**
 - **number of ranks**
- **Hardware Complexity Analysis**
 - Chip area cost of **0.0023%** of a processor die per DRAM rank
 - **No additional latency** overhead
- Experimental Methodology
 - **Detailed algorithms** for each set of **real chip** experiments
 - Extensive **security analysis** for RowHammer-preventive refreshes
- **Detailed Algorithm of Finding Concurrent Refreshes**

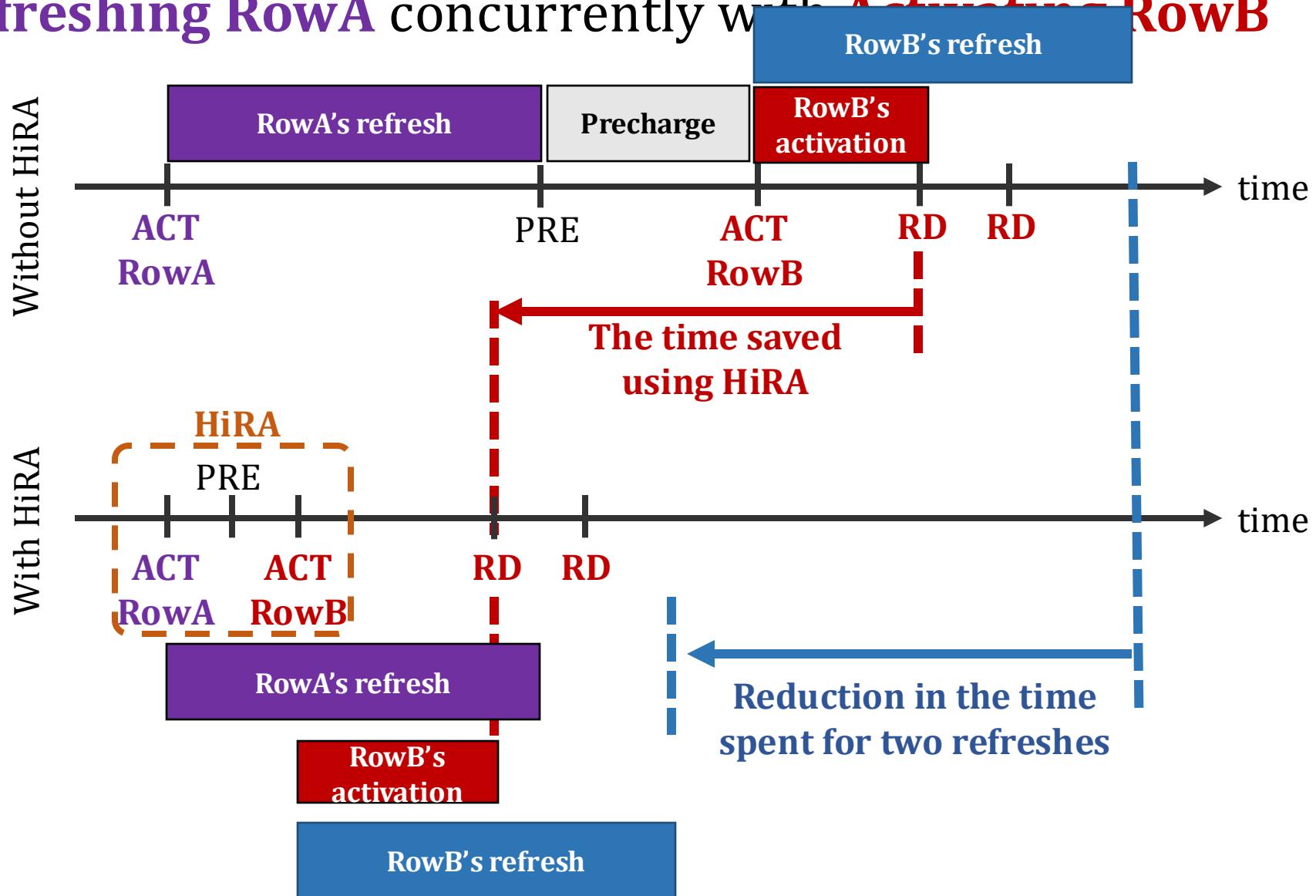
HiRA: Hidden Row Activation

- HiRA **concurrently activates** two rows in a DRAM bank
 - **Challenge 1:** Only one row can be activated in a DRAM bank at a given time
 - **Solution 1 :** HiRA **violates timing constraints** for concurrent row activations
- HiRA issues **two row activation (ACT)** commands **in quick succession**
 - **Challenge 2:** DRAM chips **ignore the second** activation before precharge
 - **Solution 2 :** HiRA issues a **precharge (PRE)** command **between two ACTs**
- HiRA activates **two DRAM rows** in the **same bank**
 - **Challenge 3:** The two rows can **override** each other's data **via shared bitlines**
 - **Solution 3 :** HiRA uses rows from two **electrically disconnected subarrays**

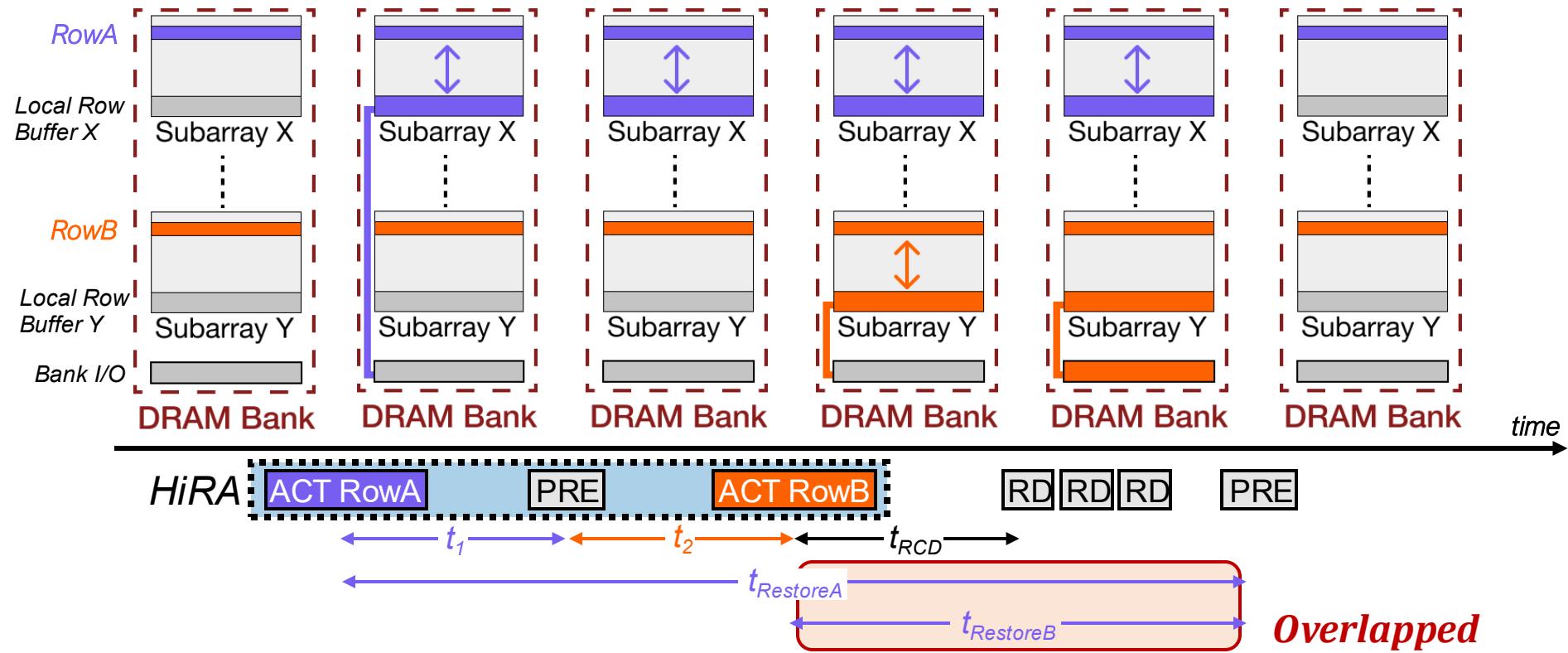
HiRA **violates DRAM timing constraints**
by issuing a sequence of **ACT-PRE-ACT** commands
that target two rows in two **electrically disconnected subarrays**

HiRA: Hidden Row Activation

Refreshing RowA concurrently with Activating RowB



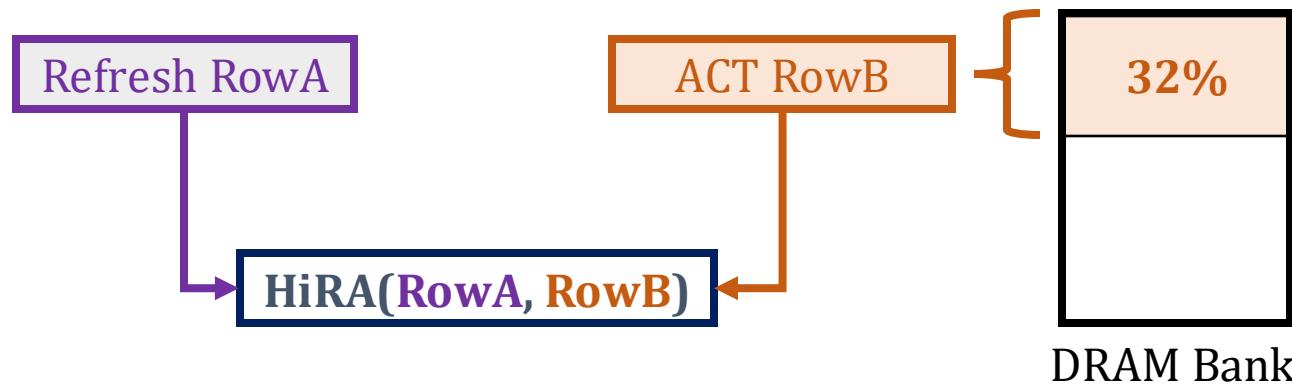
HiRA Operation



HiRA **refreshes RowA concurrently with activating RowB** by issuing **ACT-PRE-ACT** commands in **quick succession**

HiRA in Off-the-Shelf DRAM Chips: Key Results

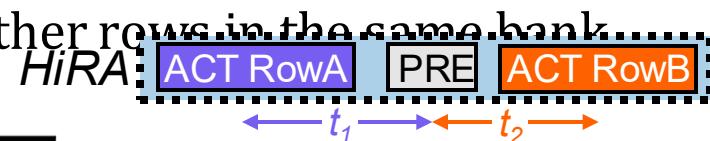
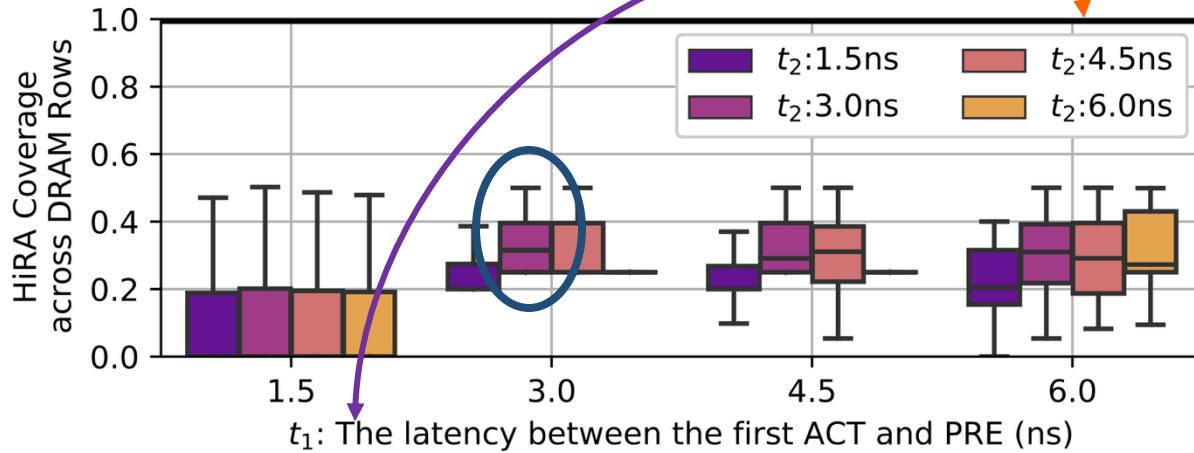
- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**
- **51.4% reduction** in the time spent for refresh operations
- HiRA performs a given row's **refresh concurrently with activating** any of the **32% of the rows** in the same bank



HiRA **effectively reduces the time spent** for **refresh** operations in **off-the-shelf** DRAM chips

HiRA Support in Off-the-Shelf DRAM Chips

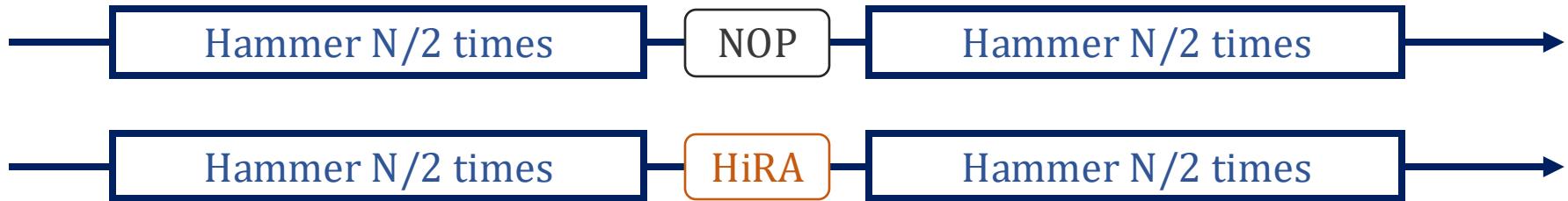
- 56 off-the-shelf DDR4 DRAM chips support HiRA (from SK Hynix)
- HiRA Coverage of a given DRAM row:
 - Refresh a given DRAM row while activating other rows in the same bank.
 - We sweep two timing parameters: t_1 and t_2 .



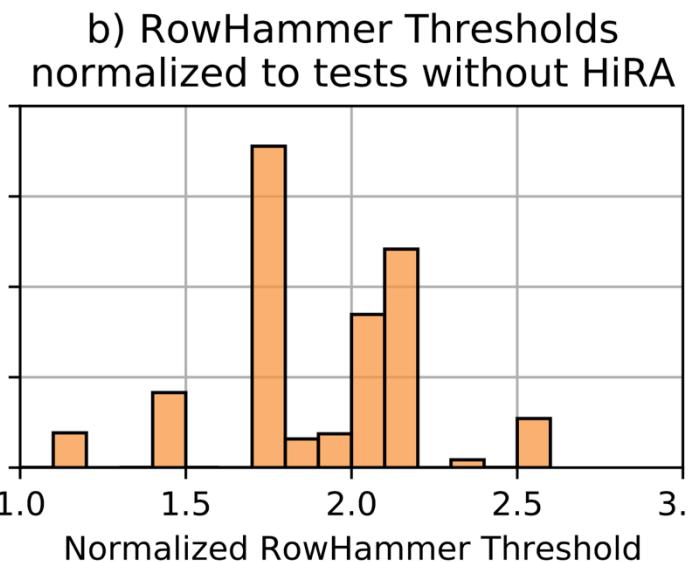
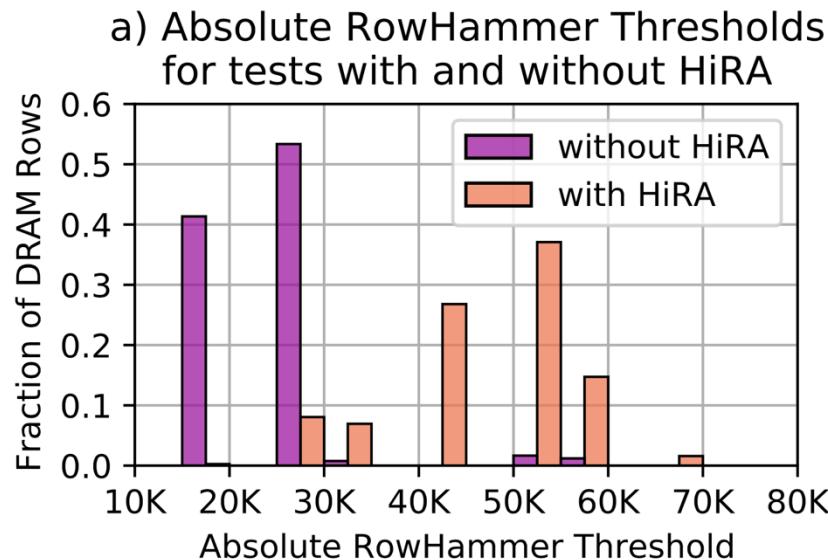
t_1 and t_2 can be
as small as 3ns

HiRA can refresh a DRAM row concurrently with 32% of any of the other DRAM rows in the same bank

HiRA's Second Row Activation

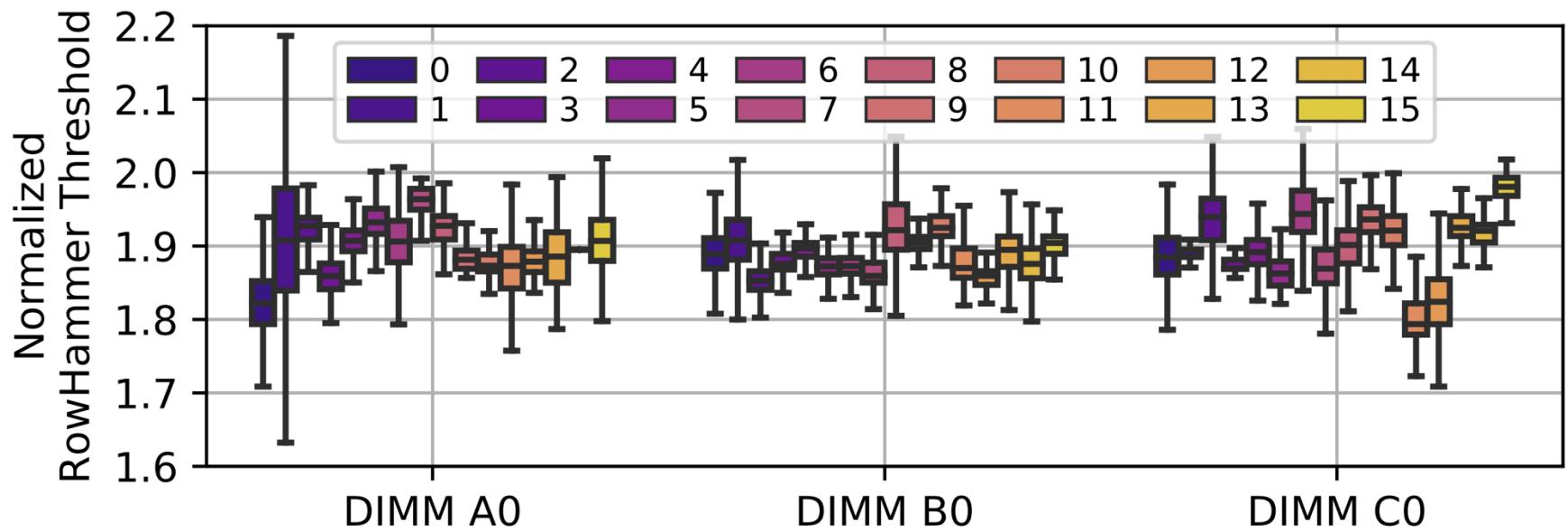


- Does performing HiRA in between refresh the victim row?
 - If HiRA's second row activation is performed, more activations are needed to induce RowHammer bit flips
 - If HiRA's second row activation is ignored, RowHammer threshold should not change



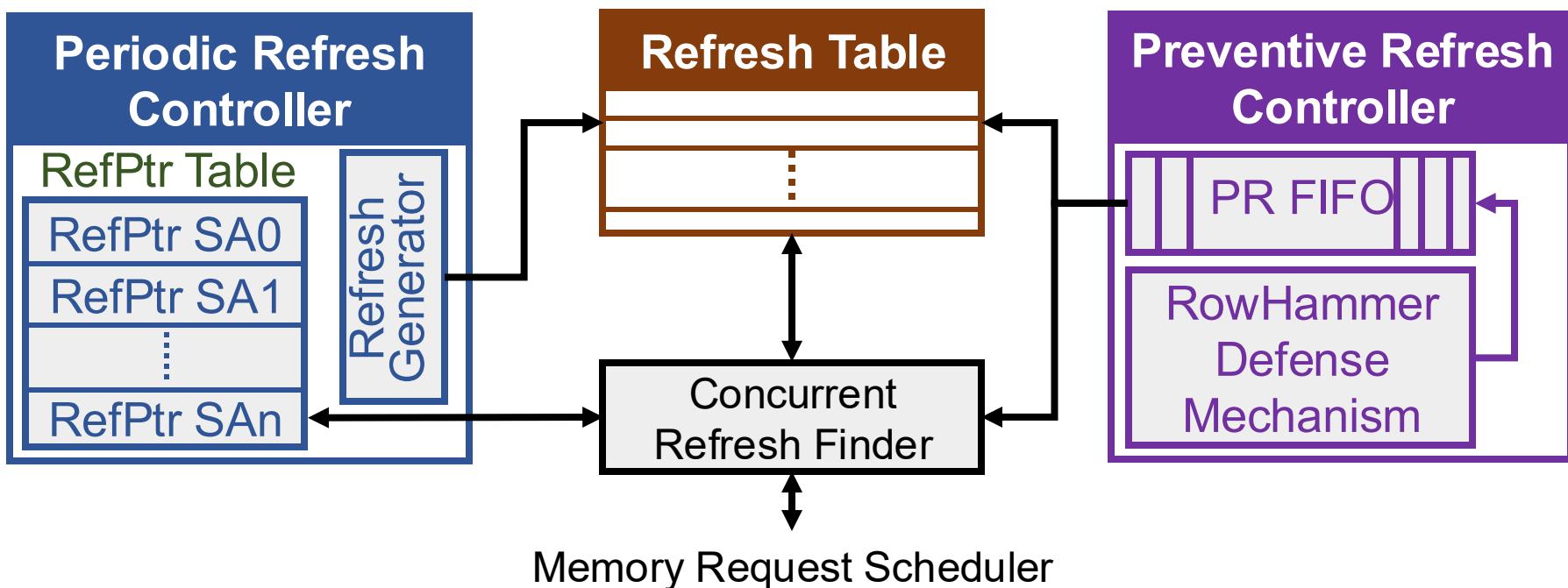
Variation across DRAM Banks

- Coverage: Identical across banks
- The effect of second row activation

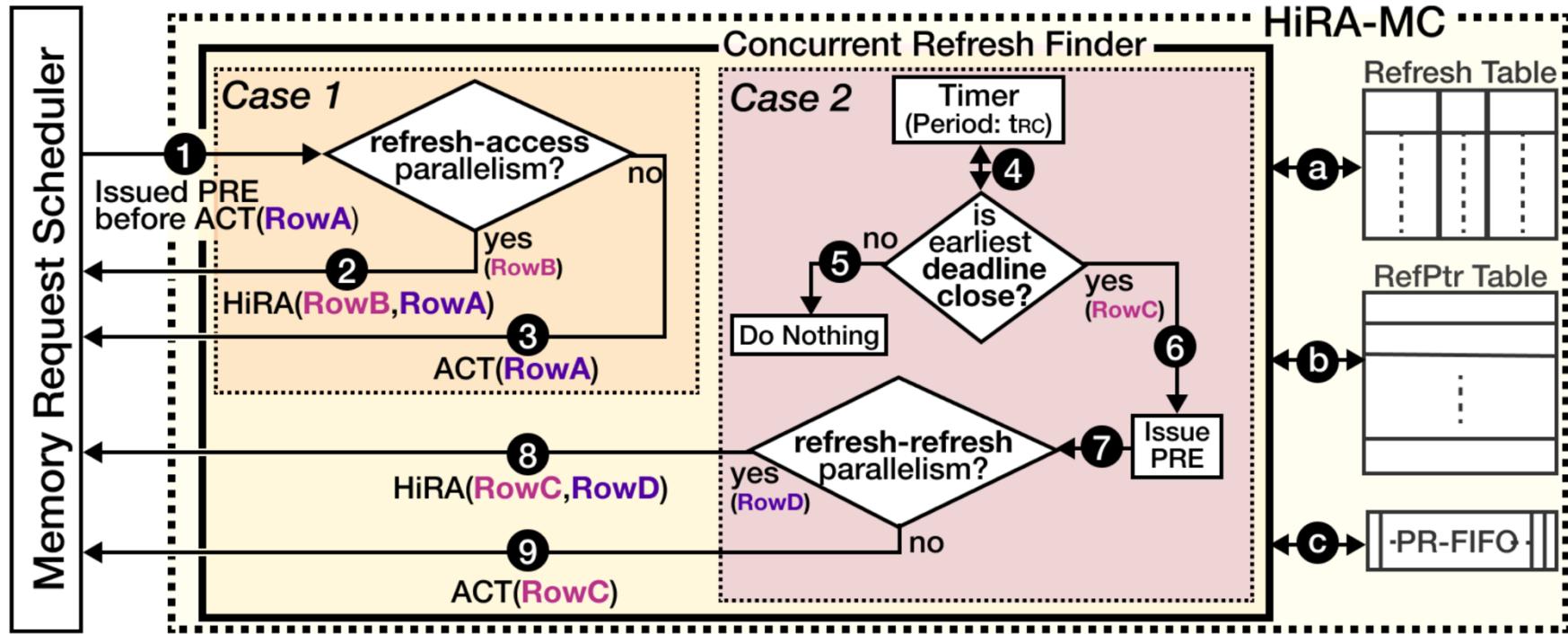


HiRA-MC: HiRA Memory Controller

- **Goal:** Leverage HiRA's parallelism as much as possible
- **Periodic** and **preventive** refresh controllers generate each refresh request **with a deadline**
- **Refresh Table** buffers a refresh request until its **deadline**
- **Concurrent Refresh Finder** finds if HiRA can refresh a row
 - *Concurrently with a memory request*
 - *Concurrently with another refresh request*



The Concurrent Refresh Finder



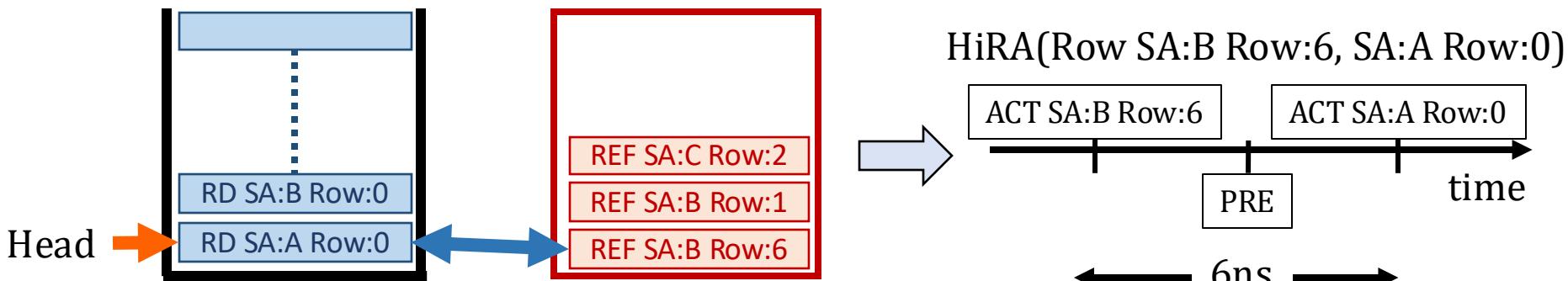
Case 1: Executes when a precharge is issued (completes before the precharge completes)

Case 2: Periodically executes after every t_{RC} (completes before t_{RC})

HiRA-MC Example

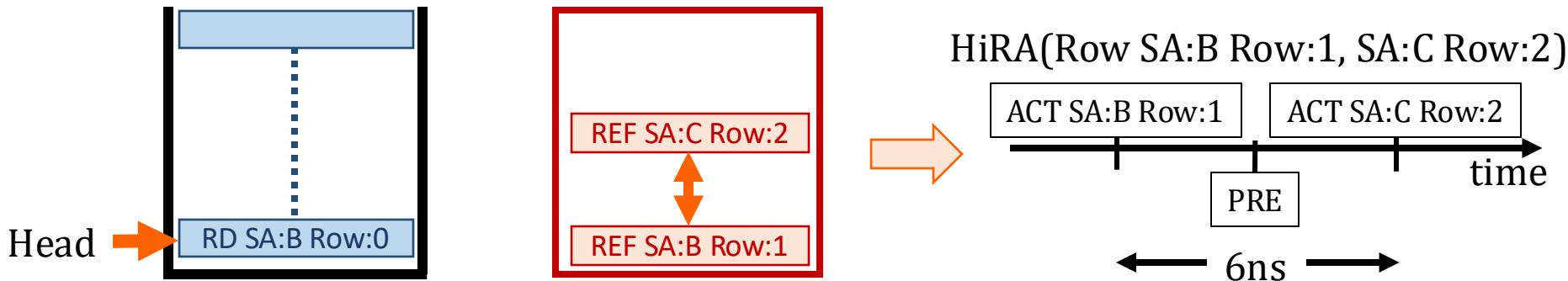
- Case 1: Refresh – Access Parallelism

Memory Request Queue Refresh Table



- Case 2: Refresh – Refresh Parallelism

Memory Request Queue Refresh Table



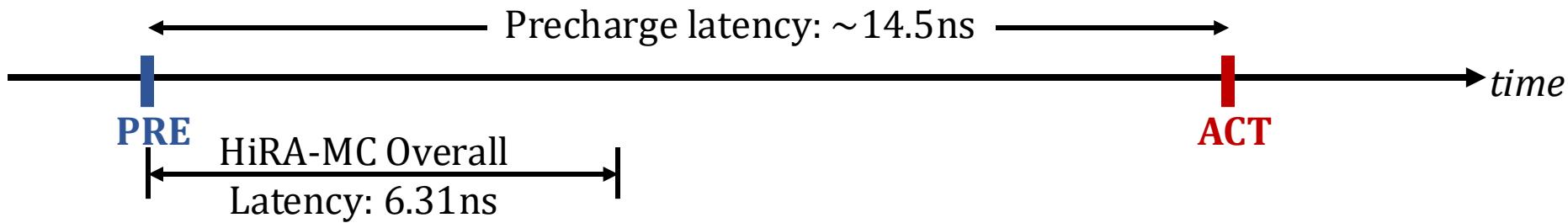
HiRA-MC provides **refresh-access** and **refresh-refresh** parallelism

HiRA-MC Hardware Complexity

- We use CACTI with 22nm technology node

HiRA-MC Component	Area (mm ²)	Area (% of Chip Area)	Access Latency
Refresh Table	0.00031	<0.0001%	0.07ns
RefPtr Table	0.00683	0.0017%	0.12ns
PR-FIFO	0.00029	<0.0001%	0.07ns
Subarray Pairs Table	0.00180	0.0005%	0.09ns
Overall	0.00923	0.0023%	6.31ns

HiRA-MC consumes only **0.0023%** of CPU chip area per DRAM rank



HiRA-MC **does not increase** memory access latency

Estimating Periodic Refresh Overhead

$$t_{RFC} = 110 \times C_{chip}^{0.6}$$

Latency of a REF command

DRAM Chip Capacity

2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture

Nonblocking Memory Refresh

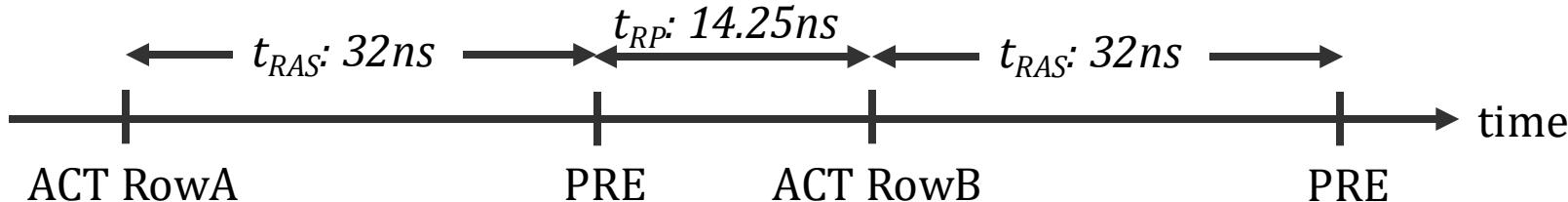
Kate Nguyen, Kehan Lyu, Xianze Meng
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
katevy@vt.edu, kehan@vt.edu, xianze@vt.edu

Vilas Sridharan
RAS Architecture
Advanced Micro Devices, Inc
Boxborough, Massachusetts
vilas.sridharan@amd.com

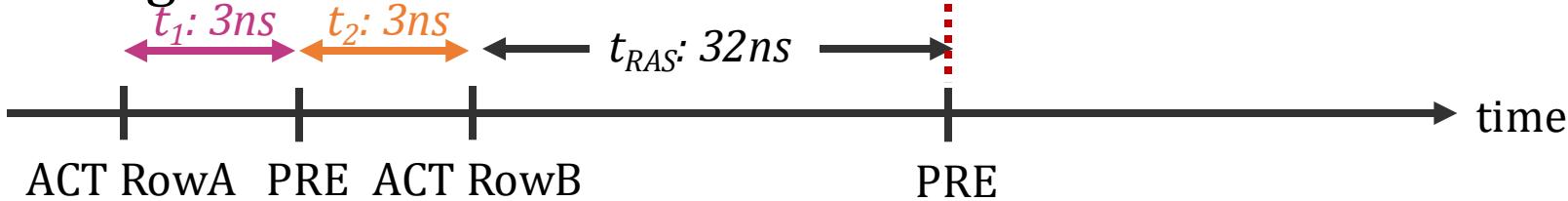
Xun Jian
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
xunj@vt.edu

Reducing Overall Latency of Two Refreshes

- Refreshing two rows using nominal timing parameters:



- Using HiRA:



Overall latency of refreshing two rows reduces **by 51.4%**
from 78.25ns down to 38ns

Tested DRAM Chips

Table 4: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage			Norm. N_{RH}		
								Min.	Avg.	Max.	Min.	Avg.	Max.
A0	G.SKILL	DWCW (Partial Marking)* F4-2400C17S-8GNT [39]	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1								24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC KSM32RD8/16HDR [87]	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1								25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN HMAA4GU6AJR8N-XN [109]	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1								29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

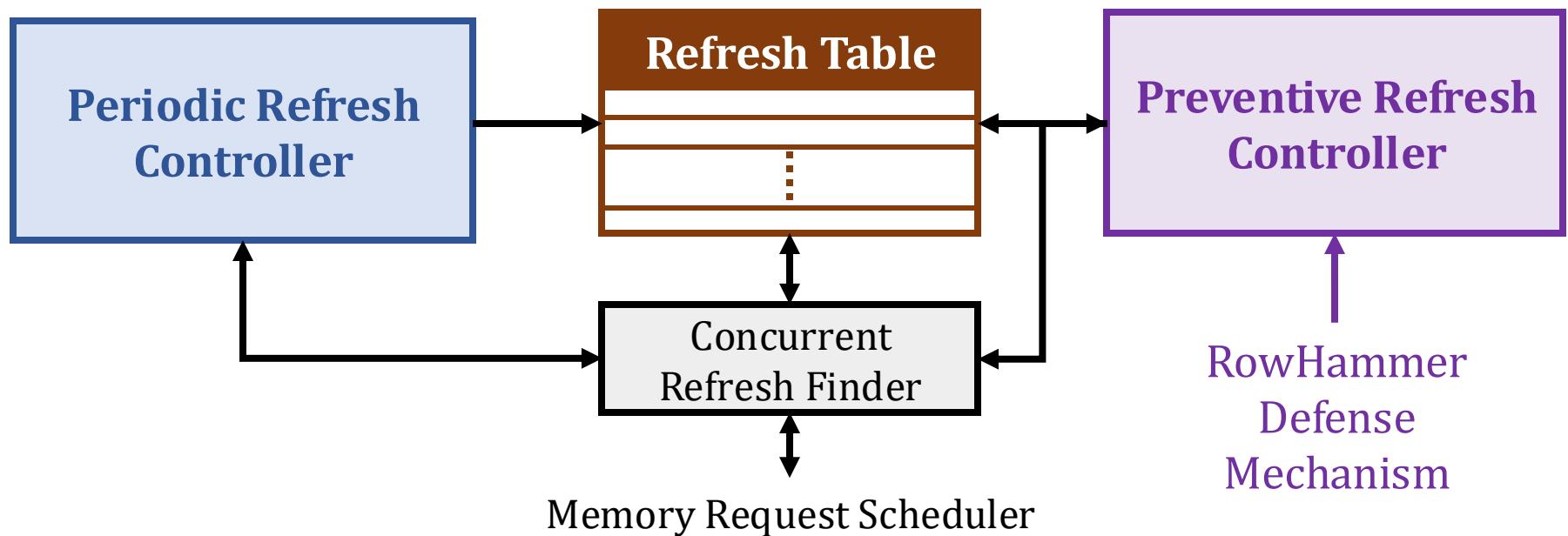
* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

<https://arxiv.org/pdf/2209.10198.pdf>

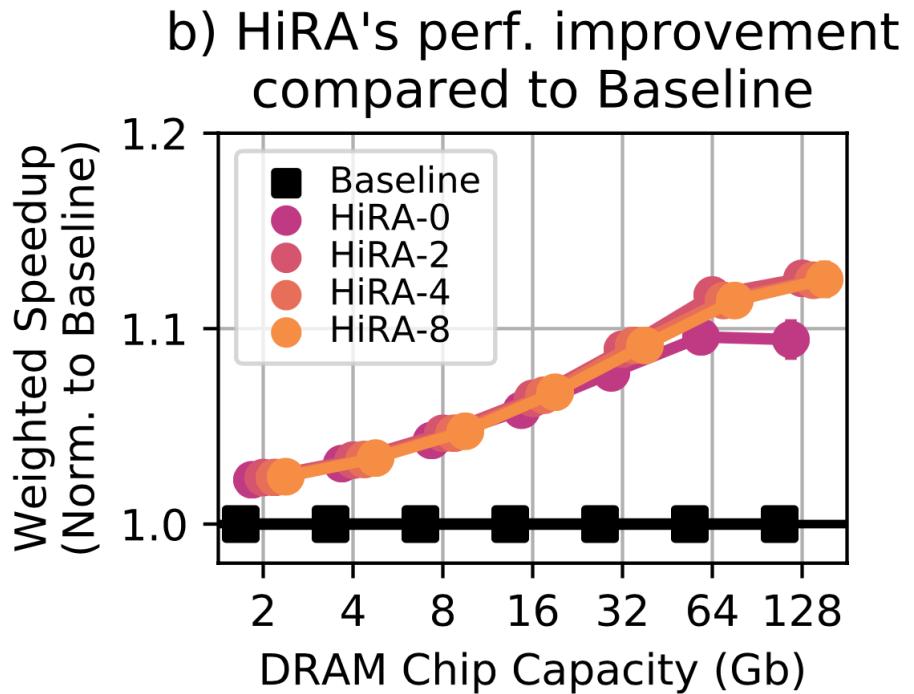
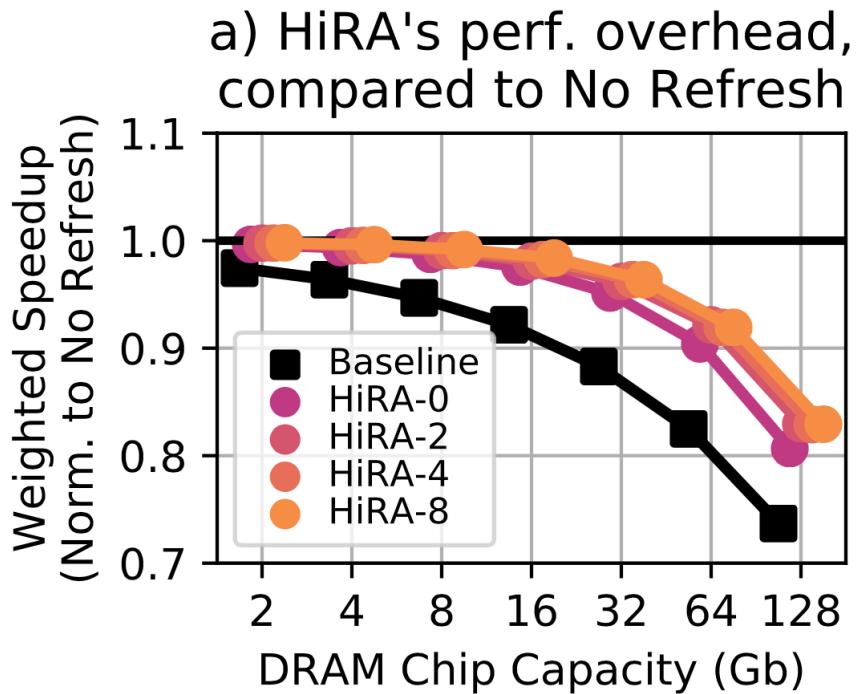


HiRA-MC: HiRA Memory Controller

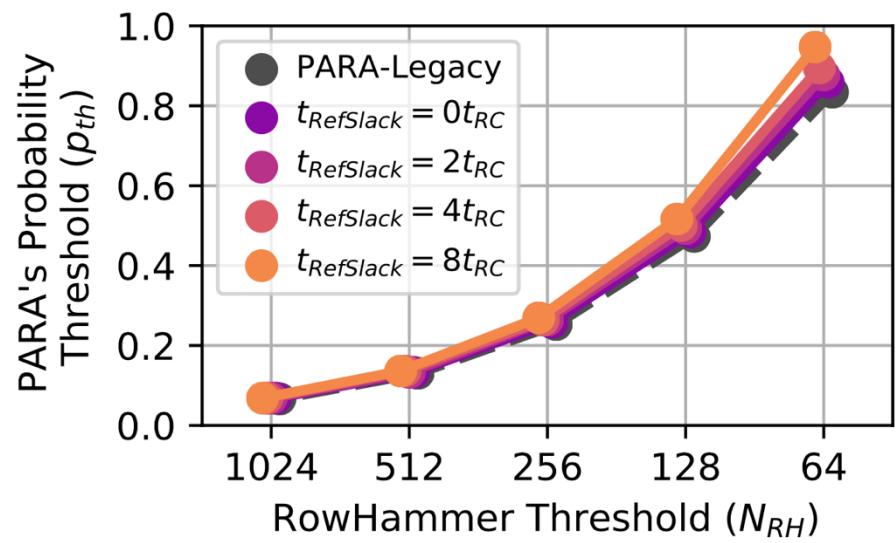
- Periodic and preventive refresh controllers generate each refresh request **with a deadline**
- Refresh Table buffers a refresh request **until its deadline**
- Concurrent Refresh Finder finds if HiRA can refresh a row
 - *Concurrently with a DRAM access*
 - *Concurrently with another refresh request*



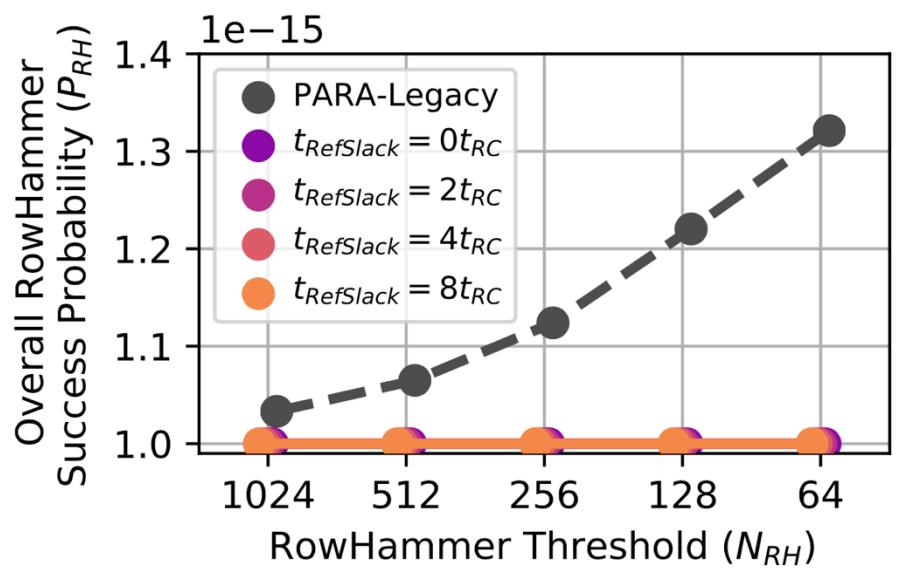
HiRA for Periodic Refreshes



RowHammer Thresholds



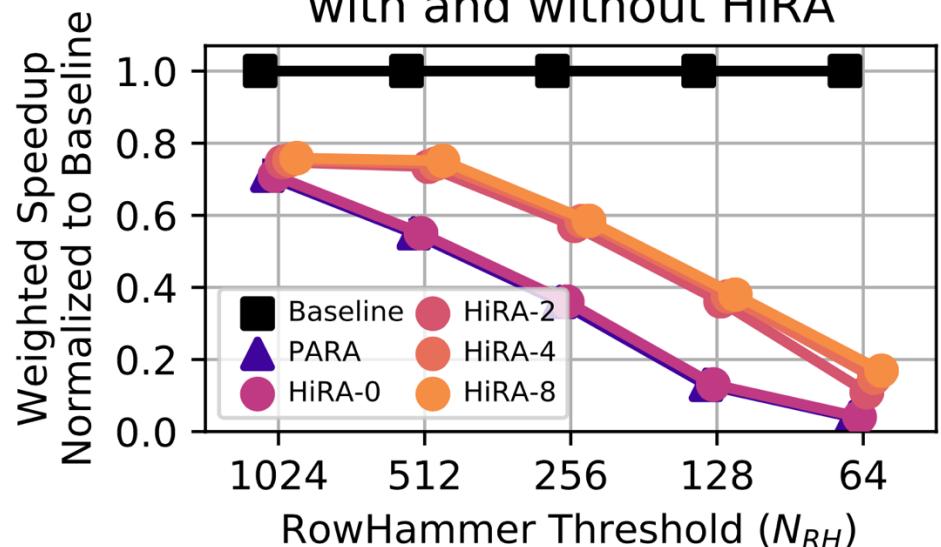
a) PARA's probability threshold (p_{th})
for different values of N_{RH} and $t_{RefSlack}$



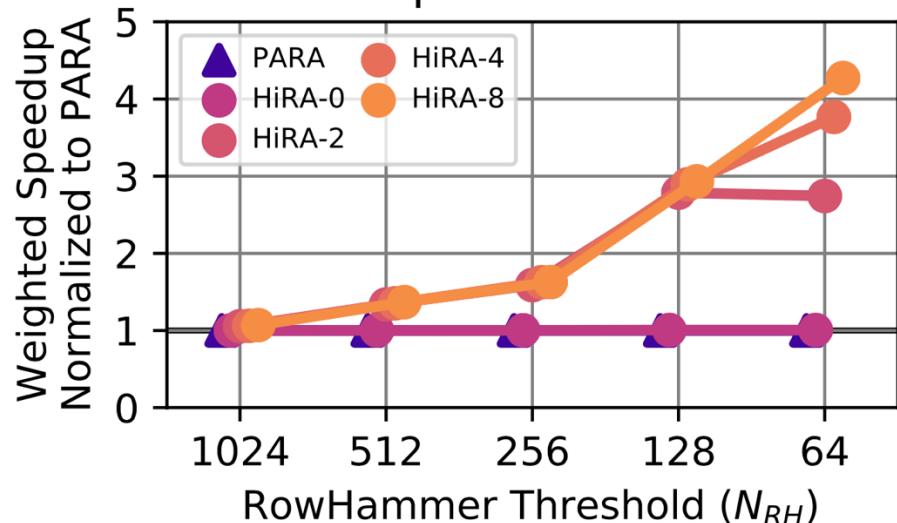
b) Overall RowHammer success probability
for different values of N_{RH} and $t_{RefSlack}$

HiRA for Preventive Refreshes

a) PARA's perf. overhead
with and without HiRA

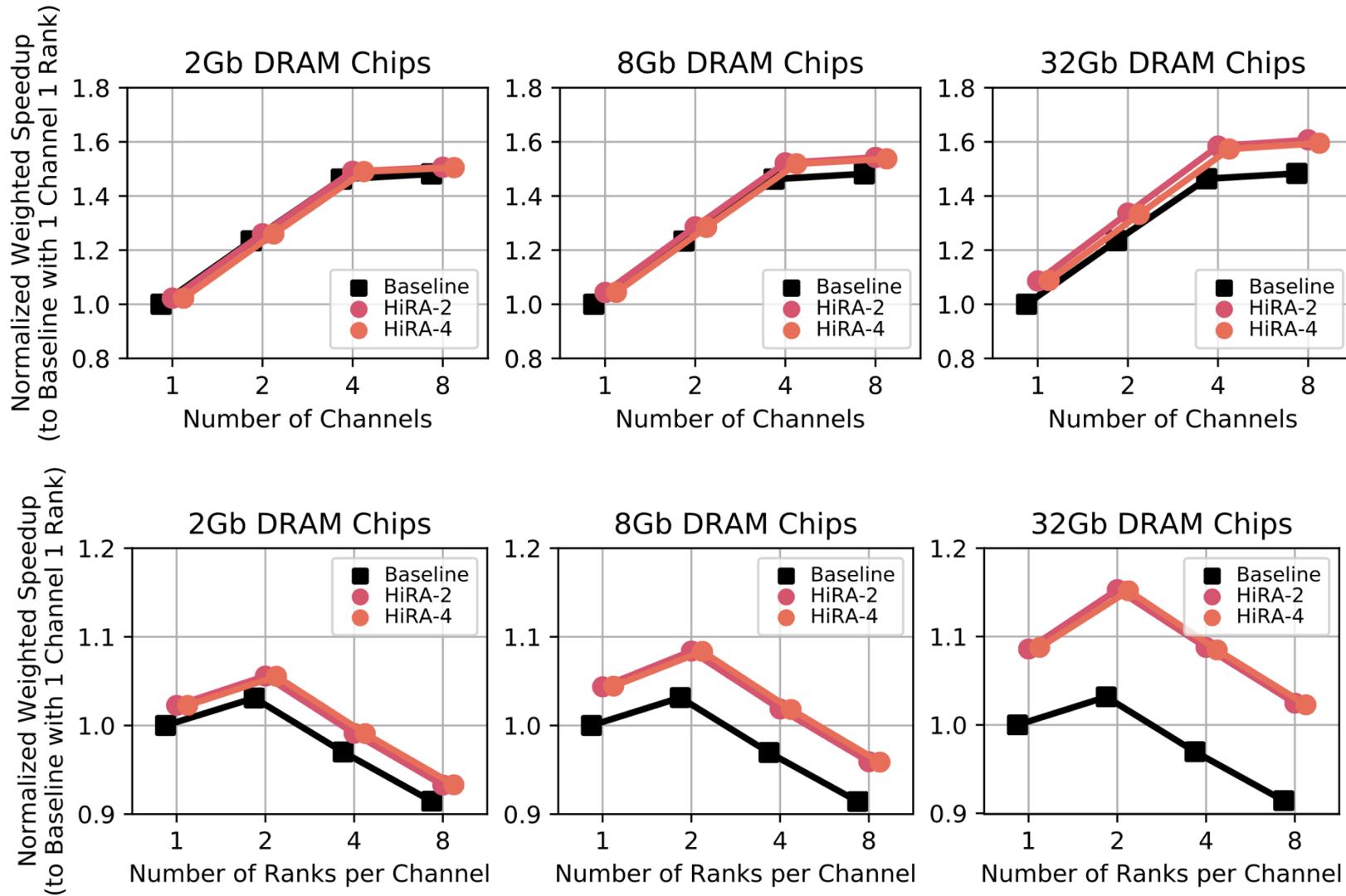


b) HiRA's perf. improvement
compared to PARA



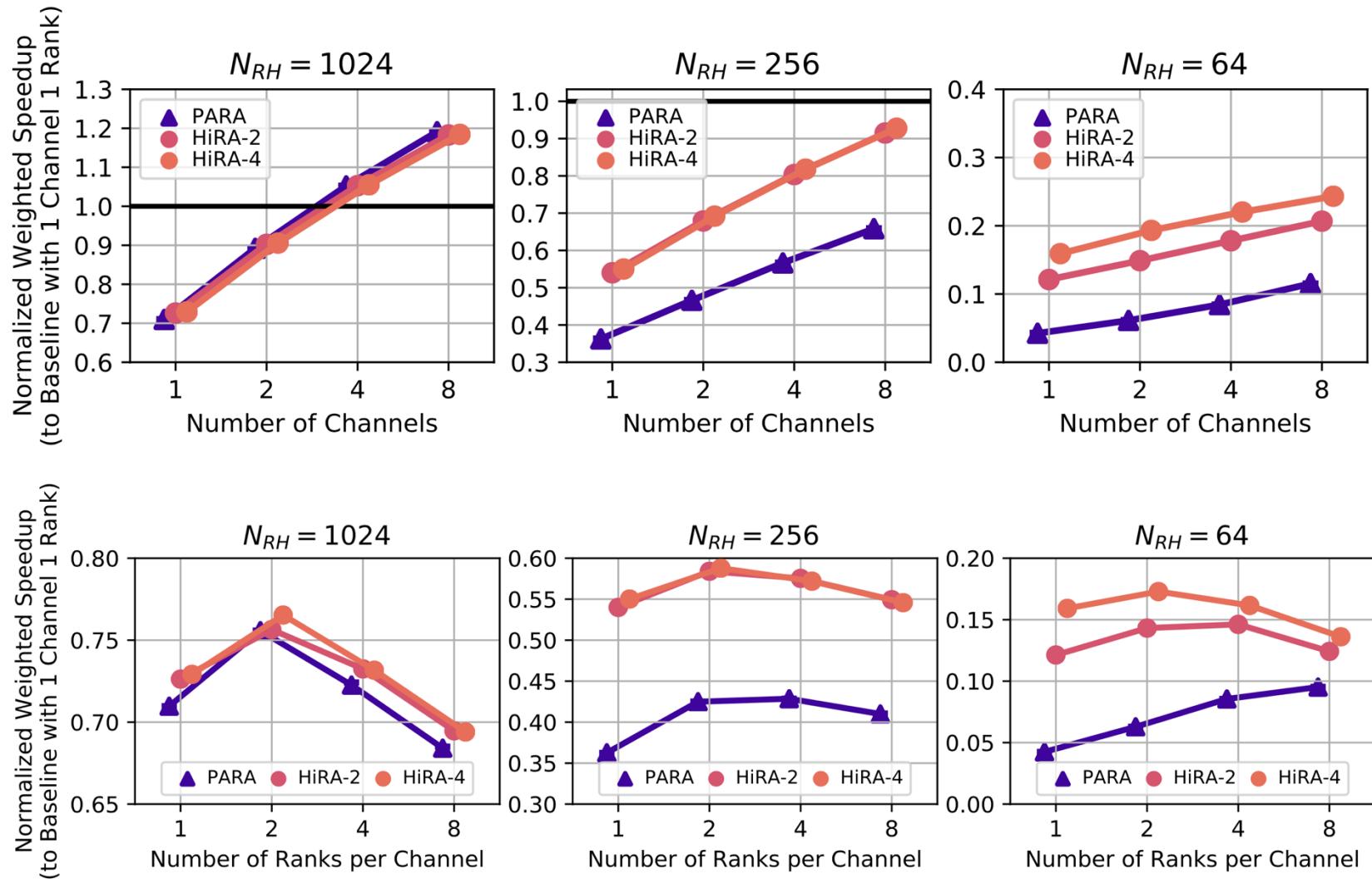
HiRA for Periodic Refresh

Sensitivity to Number of Channels and Ranks



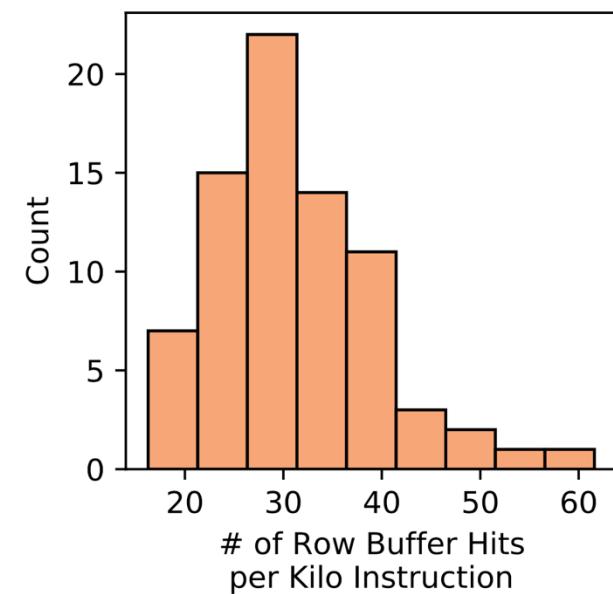
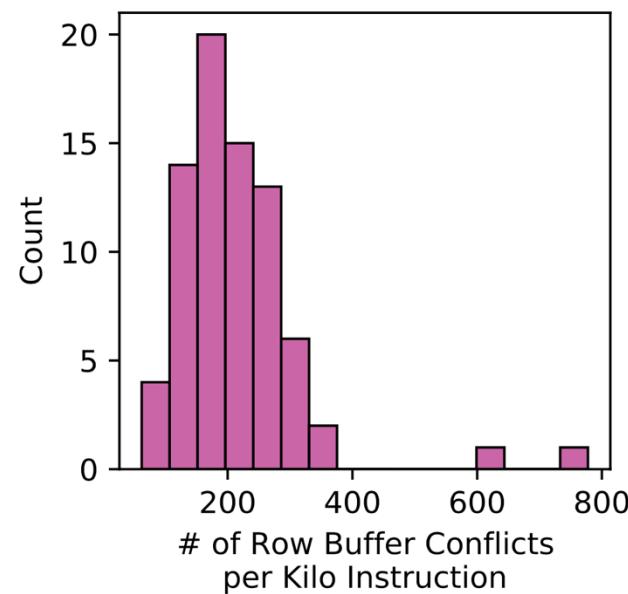
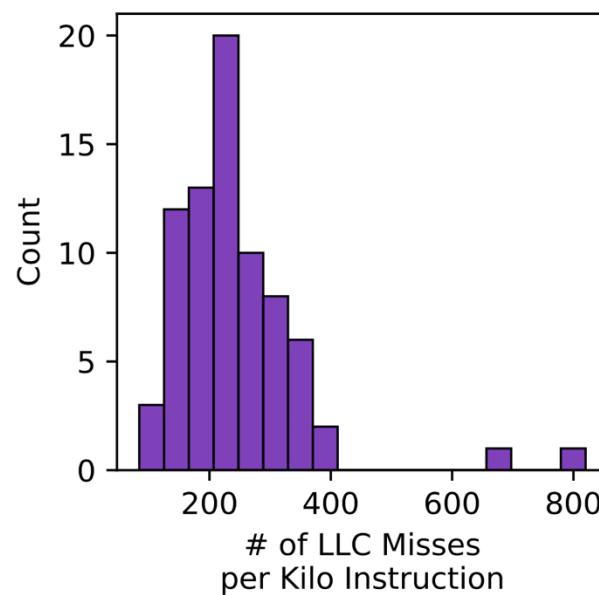
HiRA for Preventive Refresh

Sensitivity to Number of Channels and Ranks

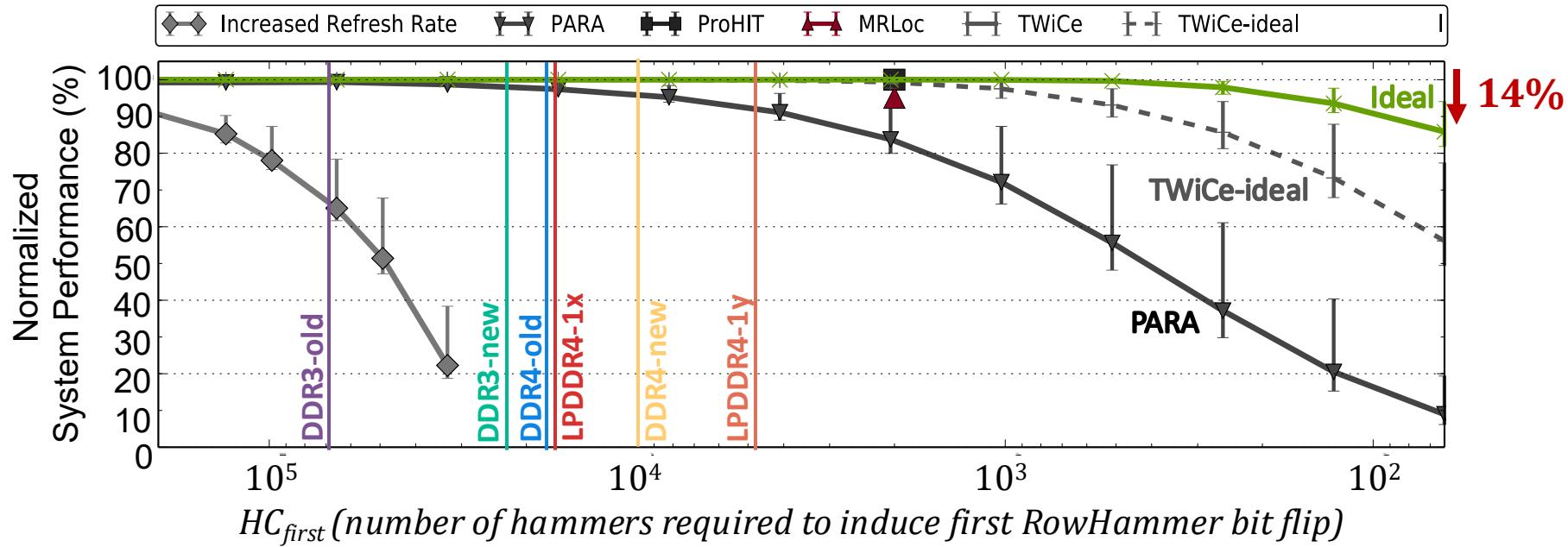


Workload Memory Access Characteristics

- 125 different 8-core multiprogrammed workloads
- Three histograms showing MPKI, RBCPKI, and RBHPKI respectively



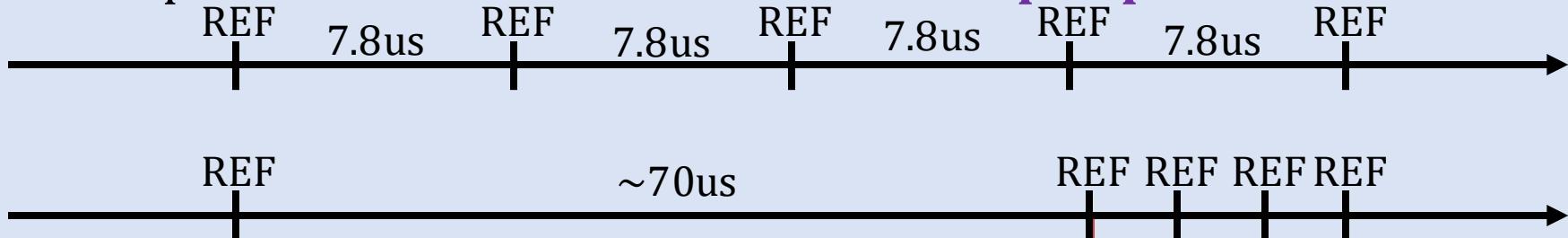
RowHammer Mitigation across Generations



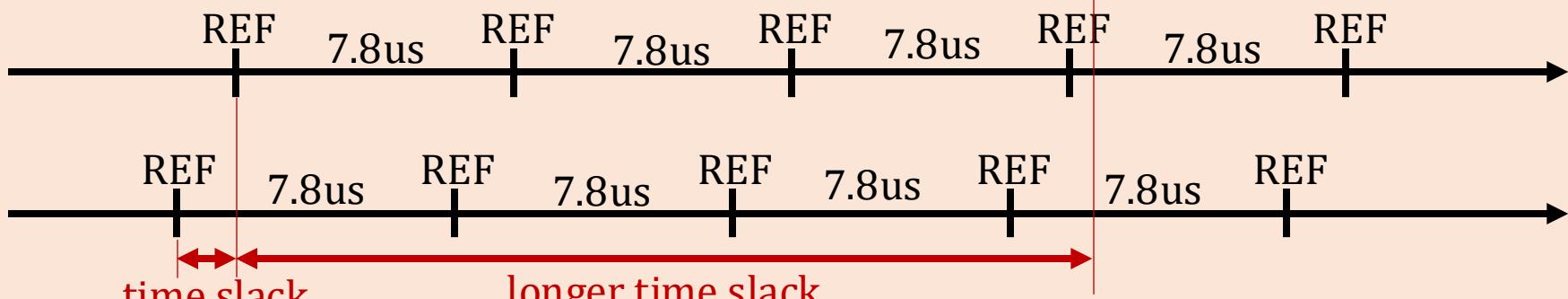
J. S. Kim, M. Patel, A. G. Yaglikci, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "[Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques](#)," in *ISCA*, 2020.

Refresh Delay

- DDRx protocols allow a REF command to be **postponed** for $\sim 70\text{us}$



- HiRA-MC's current design **does not** leverage this flexibility



- A **longer time slack** allows
 - the baseline to **better utilize DRAM idle time** to perform refresh operations
 - HiRA to find **more opportunities** to perform a refresh operation **concurrently with a DRAM access**
- **Future sensitivity study:** the effect of long refresh delays

Energy

- HiRA *does not change* the **number of refresh operations** at **a given time window**
 - Overall energy consumed for refresh operations is the same
- HiRA **improves system performance**
 - **Reduces** the background **energy consumption**
- Evaluation requires an **accurate power model** based on **real system measurements**, similar to VAMPIRE [Ghose+ SIGMETRICS'17], but for HiRA operations

HiRA: Hidden Row Activation

for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yağlıkçı

Ataberk Olgun Minesh Patel Haocong Luo Hasan Hassan

Lois Orosa Oğuz Ergin Onur Mutlu

SAFARI

ETH zürich



CESGA



TOBB ETÜ
University of Economics & Technology

Identifying Suspect Threads: High Level Algorithm

BreakHammer detects threads that trigger **too many** RowHammer-preventive actions

