

Enabling Efficient and Scalable DRAM Read Disturbance Mitigation via New Experimental Insights into Modern DRAM Chips

Abdullah Giray Yağlıkçı

Research Summary and Future Directions

July 9, 2024

SAFARI

ETH zürich

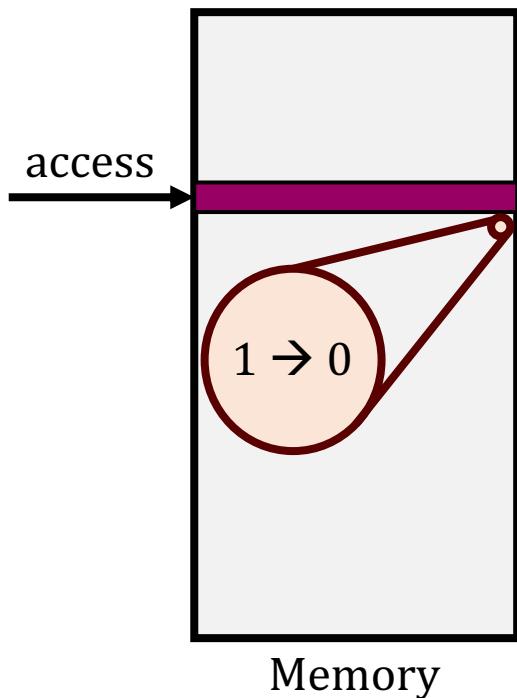
Brief Self Introduction



- Abdullah Giray Yaglikci
 - Researcher @ SAFARI Research Group since August 2016
 - ETH Zurich (Feb 2018 – ongoing)
 - Intel Labs (Aug 2017 – Feb 2018)
 - Carnegie Mellon University (Aug 2016 – Aug 2017)
 - Defended my PhD thesis, advised by Onur Mutlu, in April 2024
 - <https://agyaglikci.github.io/>
 - agirayyaglikci@gmail.com (Best way to reach me)
 - <https://safari.ethz.ch>
- **Research interests:**
 - Computer architecture, hardware security
 - Memory and storage systems
 - Hardware security, safety, reliability, performance, availability, fairness, energy efficiency
 - Hardware/software cooperation
 - ...

Memory Isolation

- A memory access should **not have unintended side effects** on data stored in **other addresses**
- A fundamental property for **robustness** (safety, security, and reliability)

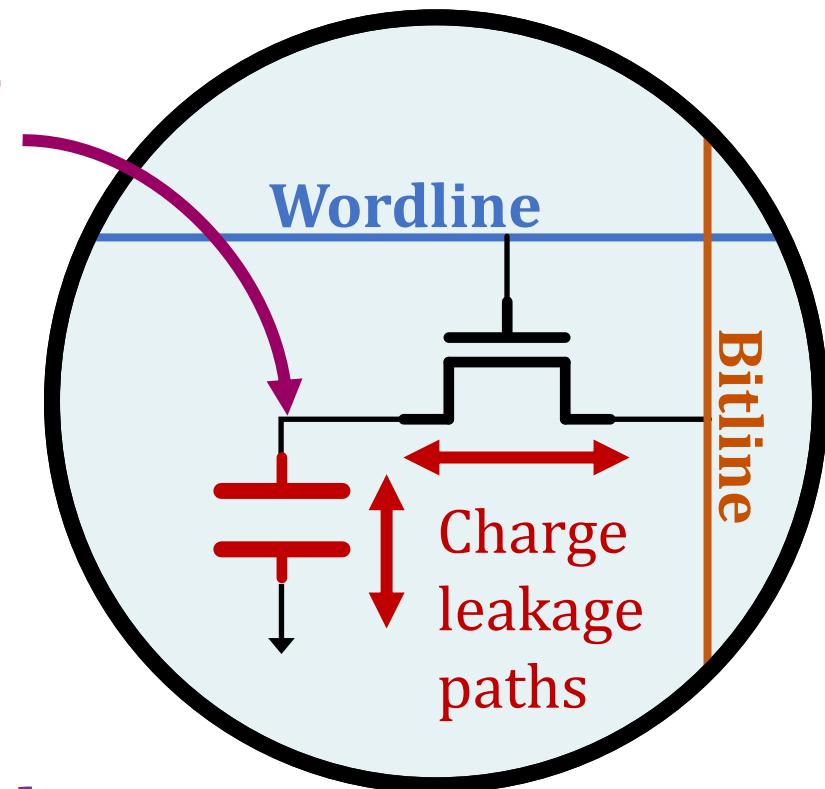


-  **Data loss or corruption**
-  Compromise **application correctness**
-  **Leak** private information
-  **Take over** a computer

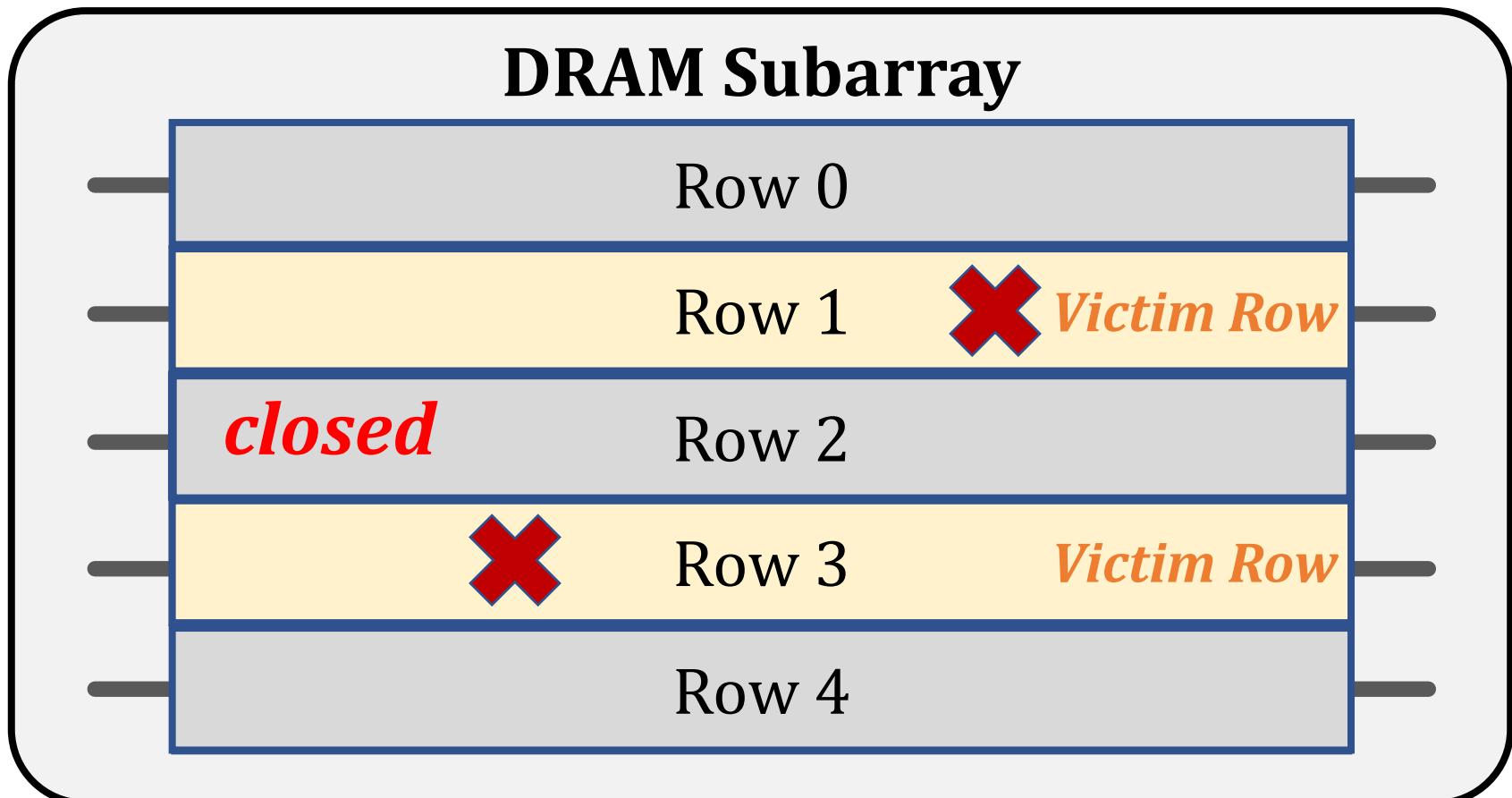
Memory isolation is **difficult in modern memory chips**

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 **RowHammer bitflips** in nearby cells and breaks **memory isolation**

One Can Take Over an Otherwise-Secure System

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

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

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

Project Zero

[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

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

GATED
COMMUNITIES

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

More Security Implications (II)

“Can gain control of a smart phone deterministically”

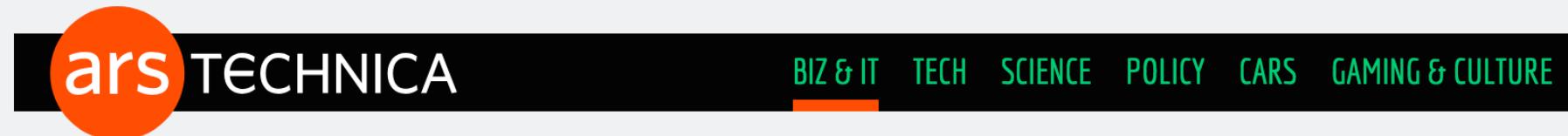


Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS'16

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)

- **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 (VI)

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[†]

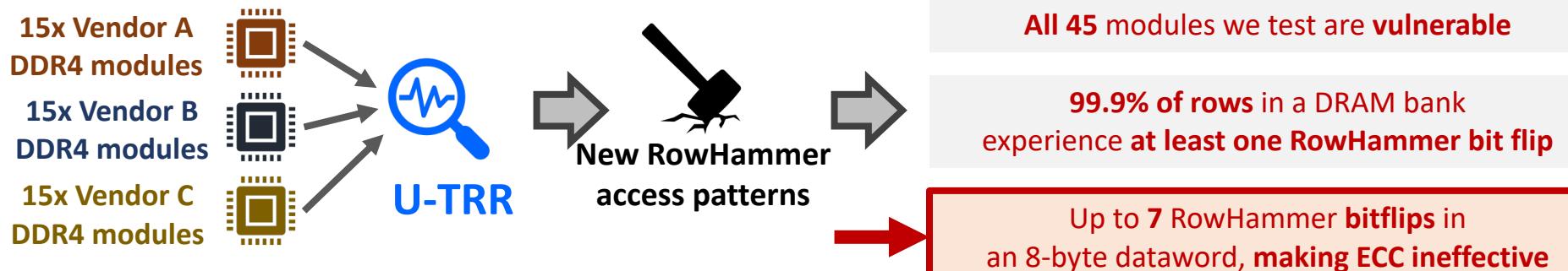
[‡]*TOBB University of Economics & Technology*

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

More Security Implications (VII)

- 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 (VIII)

JackHammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms

Zane Weissman¹, Thore Tiemann², Daniel Moghimi¹, Evan Custodio³, Thomas Eisenbarth² and Berk Sunar¹

¹ Worcester Polytechnic Institute, MA, USA

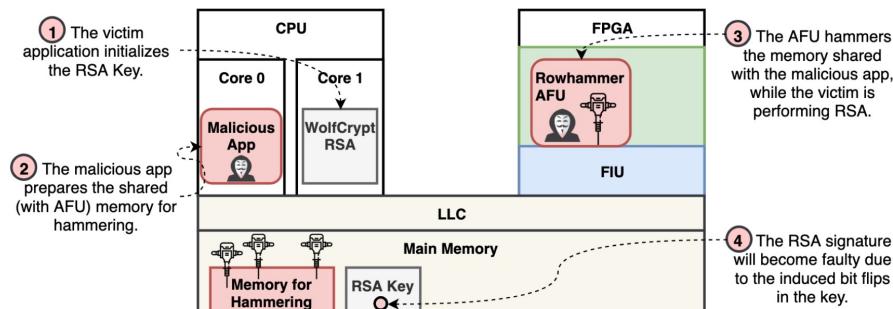
zweissman@wpi.edu, amoghimi@wpi.edu, sunar@wpi.edu

² University of Lübeck, Lübeck, Germany

thore.tiemann@student.uni-luebeck.de, thomas.eisenbarth@uni-luebeck.de

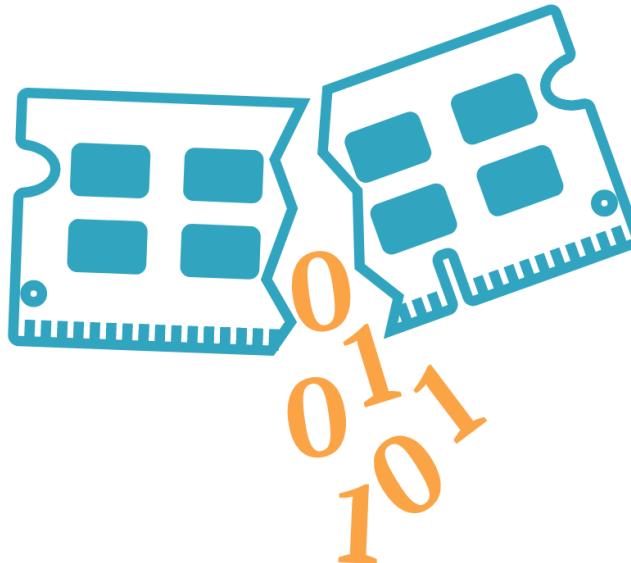
³ Intel Corporation, Hudson, MA, USA

evan.custodio@intel.com



An **FPGA-based** RowHammer attack recovering **private keys** twice as fast compared to **CPU-based** attacks

More Security Implications (IX)



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 (X)

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 (XI)

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

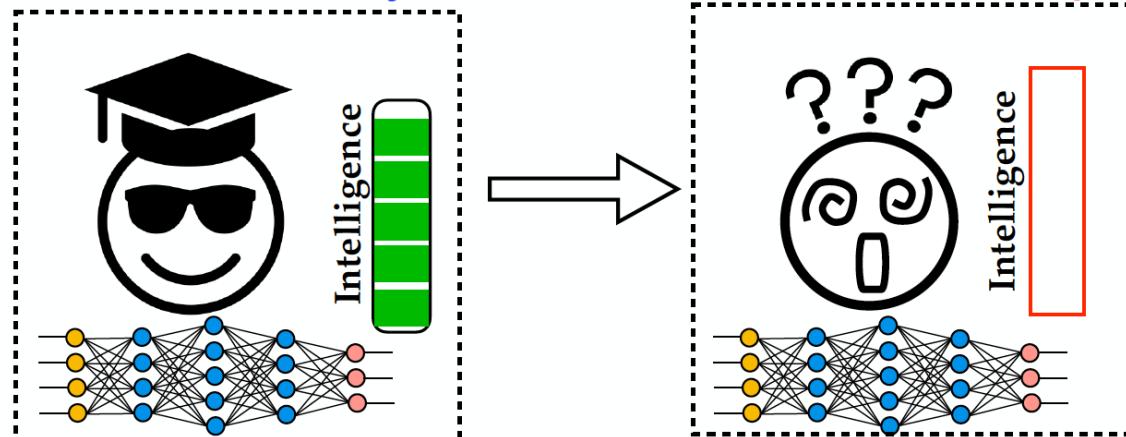
Deliang Fan

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 (XII)

HAMMERSCOPE: Observing DRAM Power Consumption Using Rowhammer

Yaakov Cohen*

Ben-Gurion University of the Negev
and Intel
Beer-Sheva, Israel
yaakoc@post.bgu.ac.il

Daniel Genkin

Georgia Institute of Technology
Atlanta, Georgia, USA
genkin@gatech.edu

Kevin Sam Tharayil*

Georgia Institute of Technology
Atlanta, Georgia, USA
kevinsam@gatech.edu

Arie Haenel*

Jerusalem College of Technology
and Intel
Jerusalem, Israel
arie.haenel@jct.ac.il

Angelos D. Keromytis

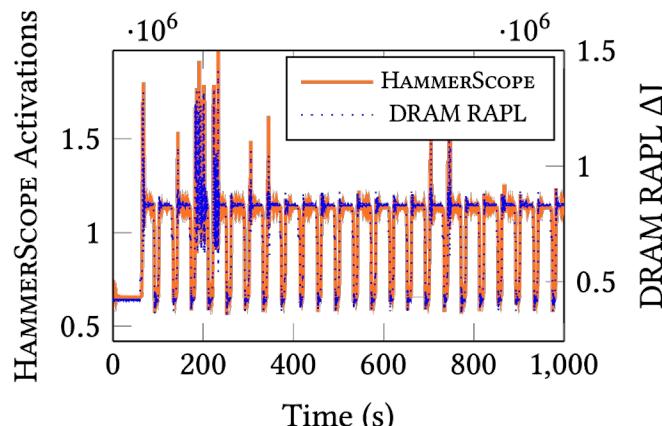
Georgia Institute of Technology
Atlanta, Georgia, USA
angelos@gatech.edu

Yossi Oren

Ben-Gurion University of the Negev
and Intel
Beer-Sheva, Israel
yos@bgu.ac.il

Yuval Yarom

University of Adelaide
Adelaide, Australia
yval@cs.adelaide.edu.au



HammerScope is a **software-based power analysis** method using **RowHammer** as a side channel

A RowHammer Survey:

Onur Mutlu, Ataberk Olgun, and A. Giray Yaglikci, "Fundamentally Understanding and Solving RowHammer" Invited Special Session Paper at the 28th Asia and South Pacific Design Automation Conference (ASP-DAC), Tokyo, Japan, January 2023.

[[arXiv version](#)]

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

[[Talk Video](#) (26 minutes)]

Fundamentally Understanding and Solving RowHammer

Onur Mutlu

onur.mutlu@safari.ethz.ch

ETH Zürich

Zürich, Switzerland

Ataberk Olgun

ataberk.olgun@safari.ethz.ch

ETH Zürich

Zürich, Switzerland

A. Giray Yağlıkçı

giray.yaglikci@safari.ethz.ch

ETH Zürich

Zürich, Switzerland

<https://arxiv.org/pdf/2211.07613.pdf>

Breaking Samsung's Best Practice

- 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

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

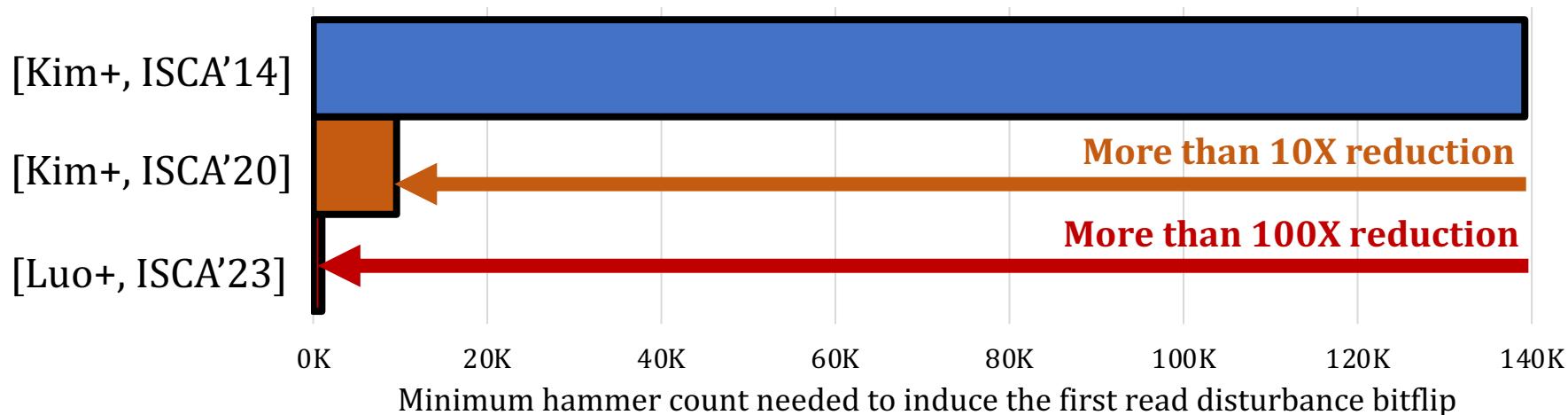
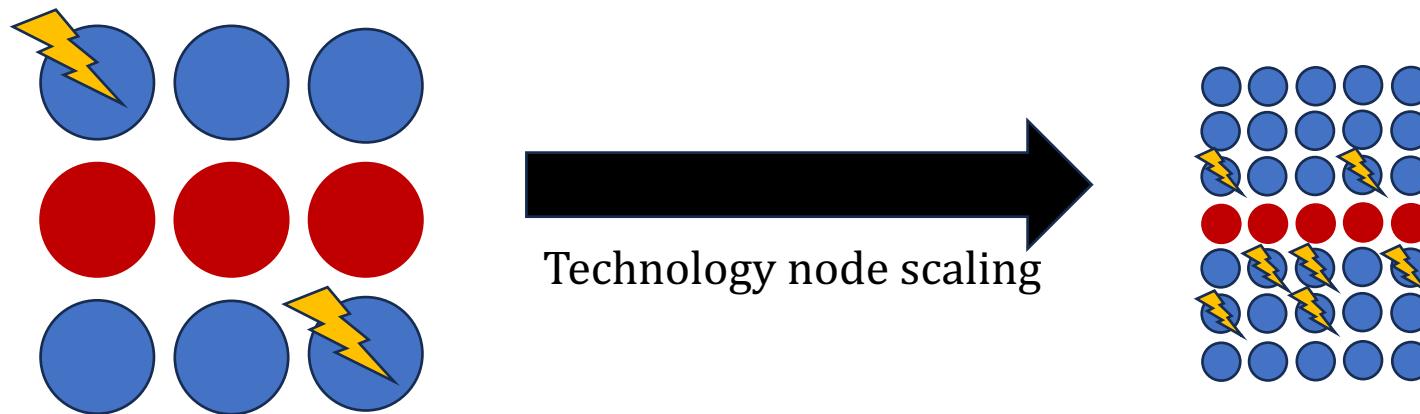
Patrick Jattke[†] Max Wipfli[†] Flavien Solt Michele Marazzi Matej Bölcskei Kaveh Razavi

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.

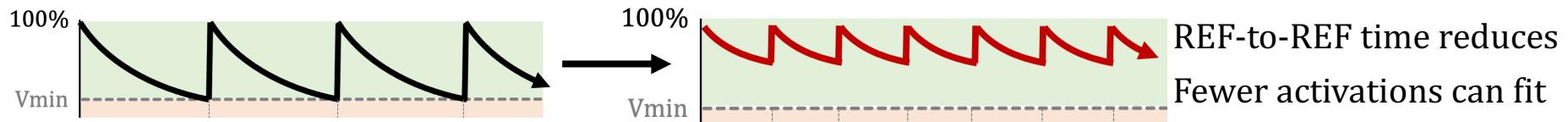
DRAM Read Disturbance: A Critical Challenge



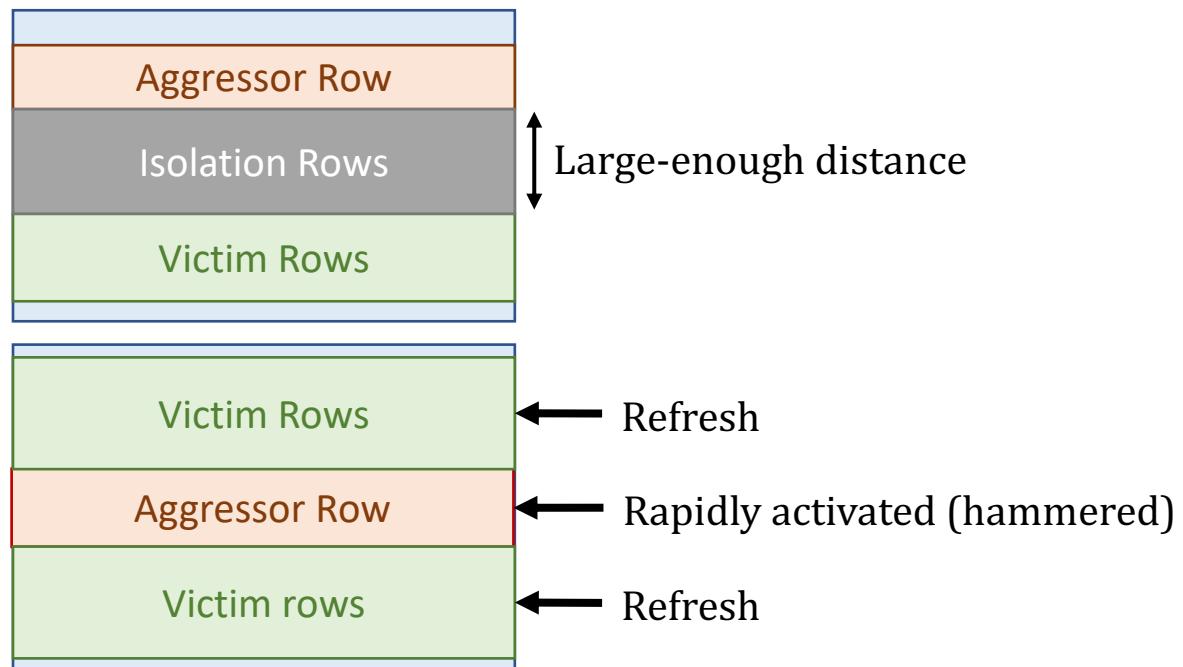
DRAM cells become **increasingly**
more vulnerable to read disturbance

How to Solve DRAM Read Disturbance?

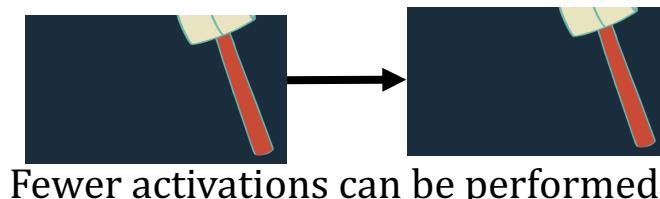
- Build better chips
- Increased refresh rate



- Physical isolation



- Reactive refresh



My PhD Research

Two Key Challenges

1

Scalability

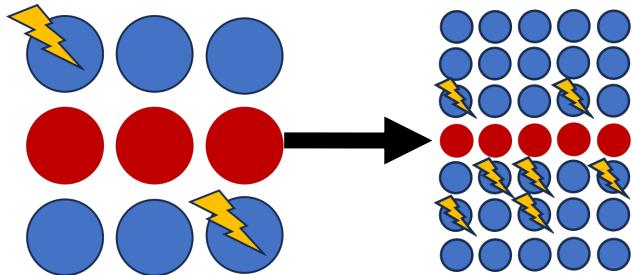
with worsening read disturbance vulnerability

2

Compatibility

with commodity DRAM chips

Challenge (1/2): Scalability



DRAM cells become
increasingly more vulnerable
to read disturbance at device level

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

[Kim+, ISCA'20]

[Frigo+, IEEE S&P'20]



[Hassan+, MICRO'21]

We need **more efficient and scalable** solutions
to DRAM read disturbance



A **more detailed understanding** of
DRAM read disturbance can help

Challenge (2/2): Compatibility

Application Level

Virtual Memory Address

System Level

Physical Memory Address

Memory Controller

DRAM Bus Addresses
(Channel, Rank, Bank, Row...)

Visible within the processor

In-DRAM

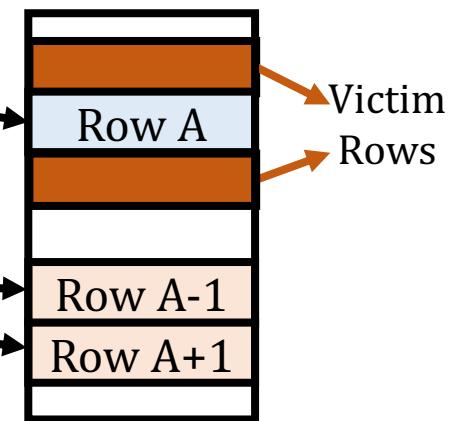
Proprietary

Processor-visible
Row Addresses

Physical
Row Addresses

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**



Physical Row Layout

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

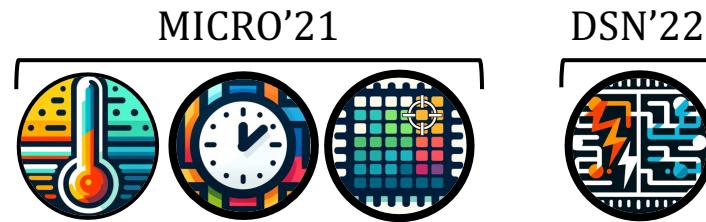
Thesis Statement

We can mitigate DRAM read disturbance efficiently and scalably by

1 building a detailed understanding of DRAM read disturbance

2 leveraging insights into modern DRAM chips and memory controllers

3 devising novel solutions that do not require proprietary knowledge of DRAM chip internals



Core Contributions

1

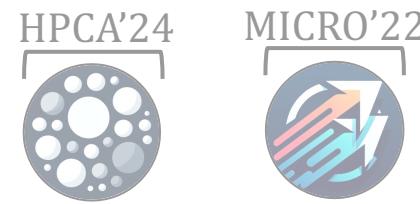
building a
detailed understanding
of DRAM read disturbance

2

leveraging **insights into**
modern DRAM chips
and memory controllers

3

devising **novel solutions**
that do not require
proprietary knowledge
of DRAM chip internals



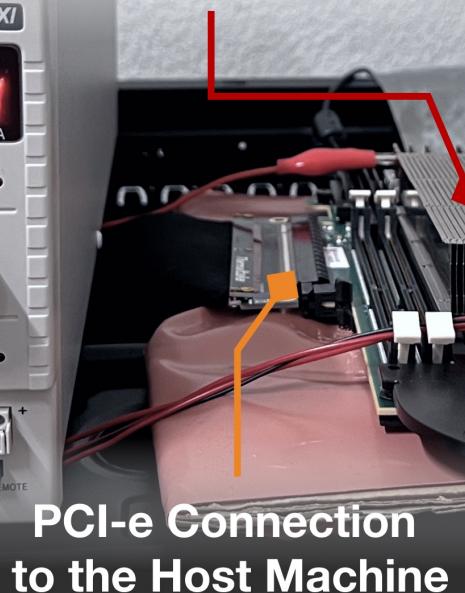
DRAM Testing Infrastructure

DRAM Bender on a Xilinx Virtex UltraScale+ XCU200

TTi PL068-P
Power Supply



Xilinx Alveo U200
FPGA Board
(with DRAM Bender)



DRAM
Module

Heater
Pads

FT200
Temperature
Controller



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

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**

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

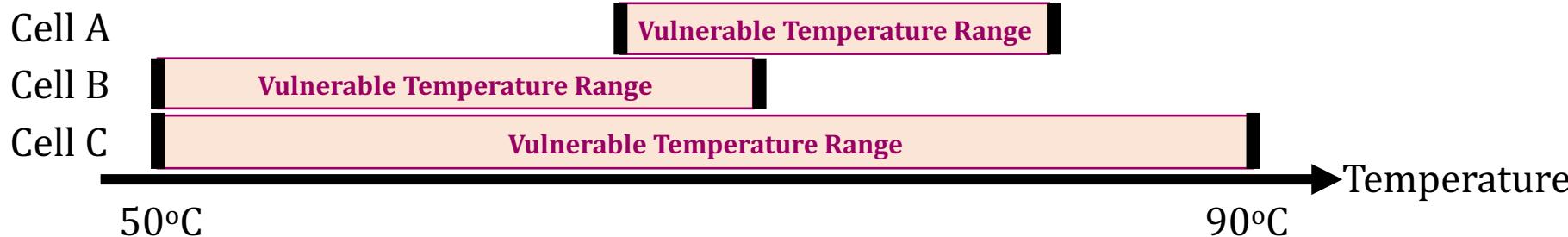


Key Finding 1: Temperature Range

- DRAM read disturbance is more effective within a **bounded vulnerable temperature range**



- Vulnerable temperature range **varies across DRAM cells**



- Most cells are vulnerable in a **continuous temperature range**

Micron	Samsung	SK Hynix	Nanya
99.1%	98.9%	98.0%	99.2%



Key Finding 1: Temperature Range

- DRAM read disturbance is more effective within a

Implication on testing

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

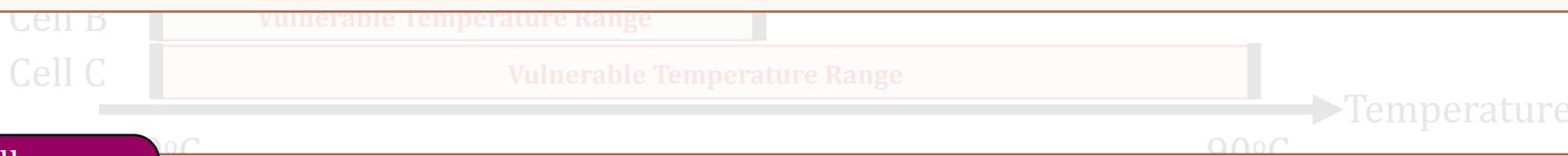
50°C 90°C

Lower
Bound

Upper
Bound

Implication on defenses

A DRAM cell **can be retired**
based on its **vulnerable temperature range**



Follow-up

This finding paved the way to **SpyHammer**,
a new attack that spies on temperature [Orosa+, 2022]

99.1%

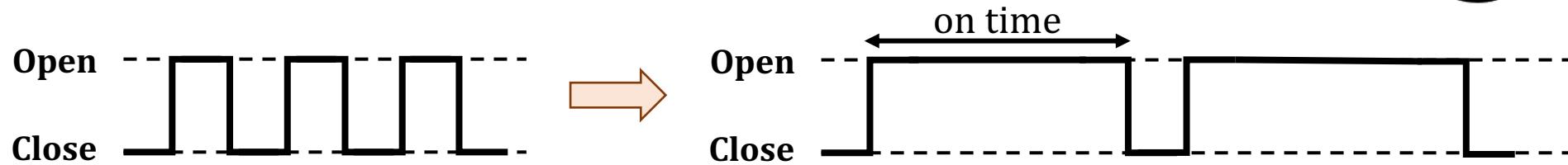
98.9%

98.0%

99.2%



Key Finding 2: Aggressor Row On Time



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

State-of-the-Art
Our Finding
w/ 15 column accesses

36% reduction

Minimum hammer count to induce the first bitflip

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]



Key Finding 3: Variation across DRAM Rows

Weakest
10% of rows

90% of 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)

Core Contributions

1

building a
detailed understanding
of DRAM read disturbance

2

leveraging **insights into**
modern DRAM chips
and memory controllers

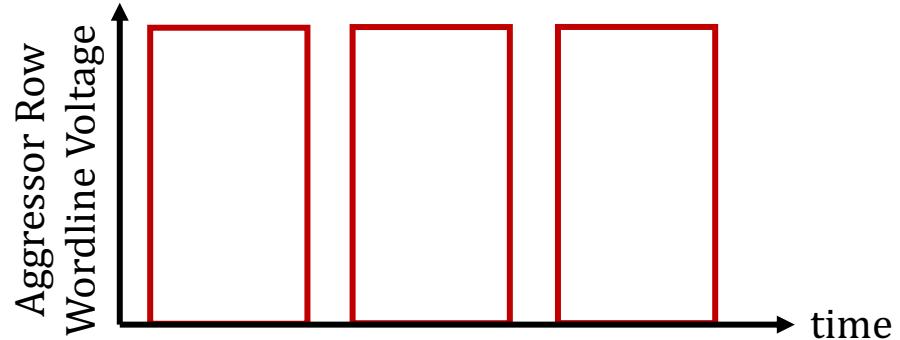
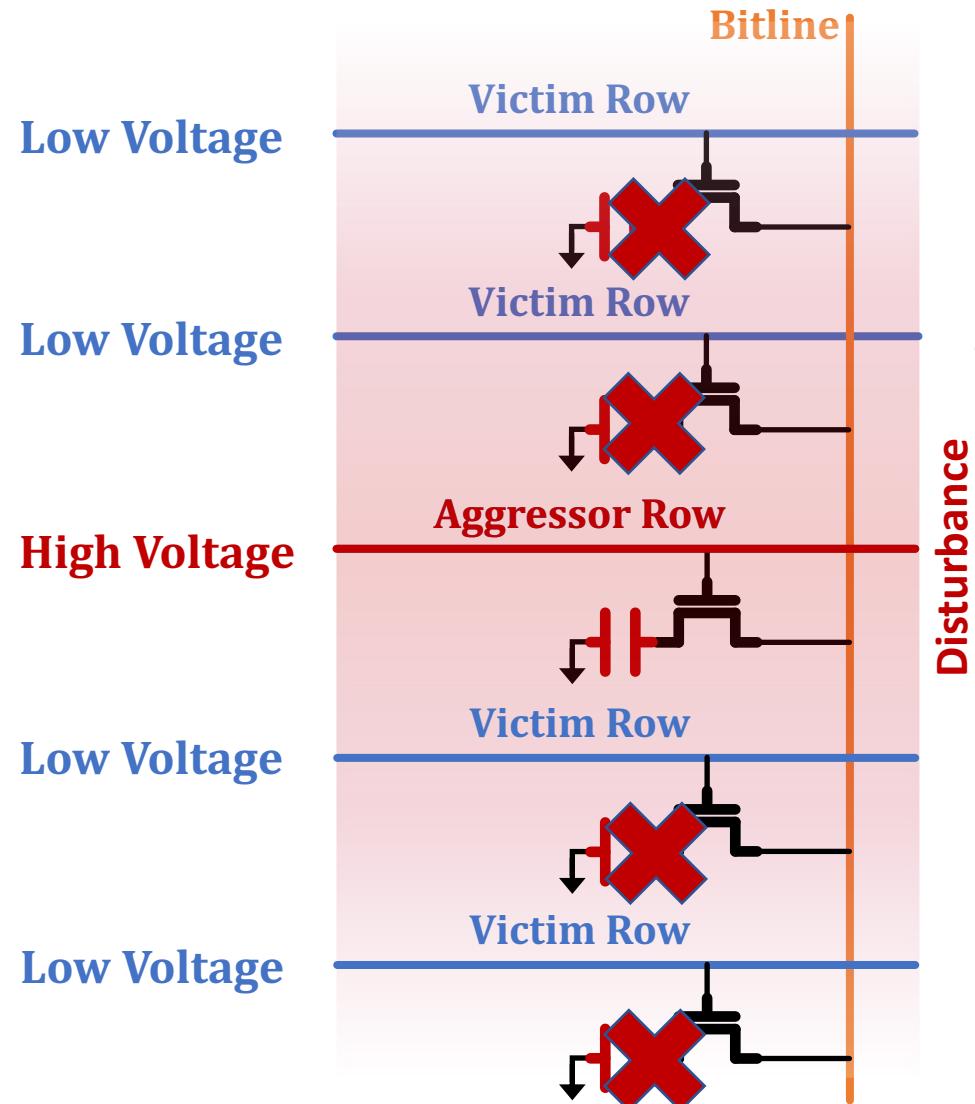
3

devising **novel solutions**
that do not require
proprietary knowledge
of DRAM chip internals





A Closer Look into RowHammer

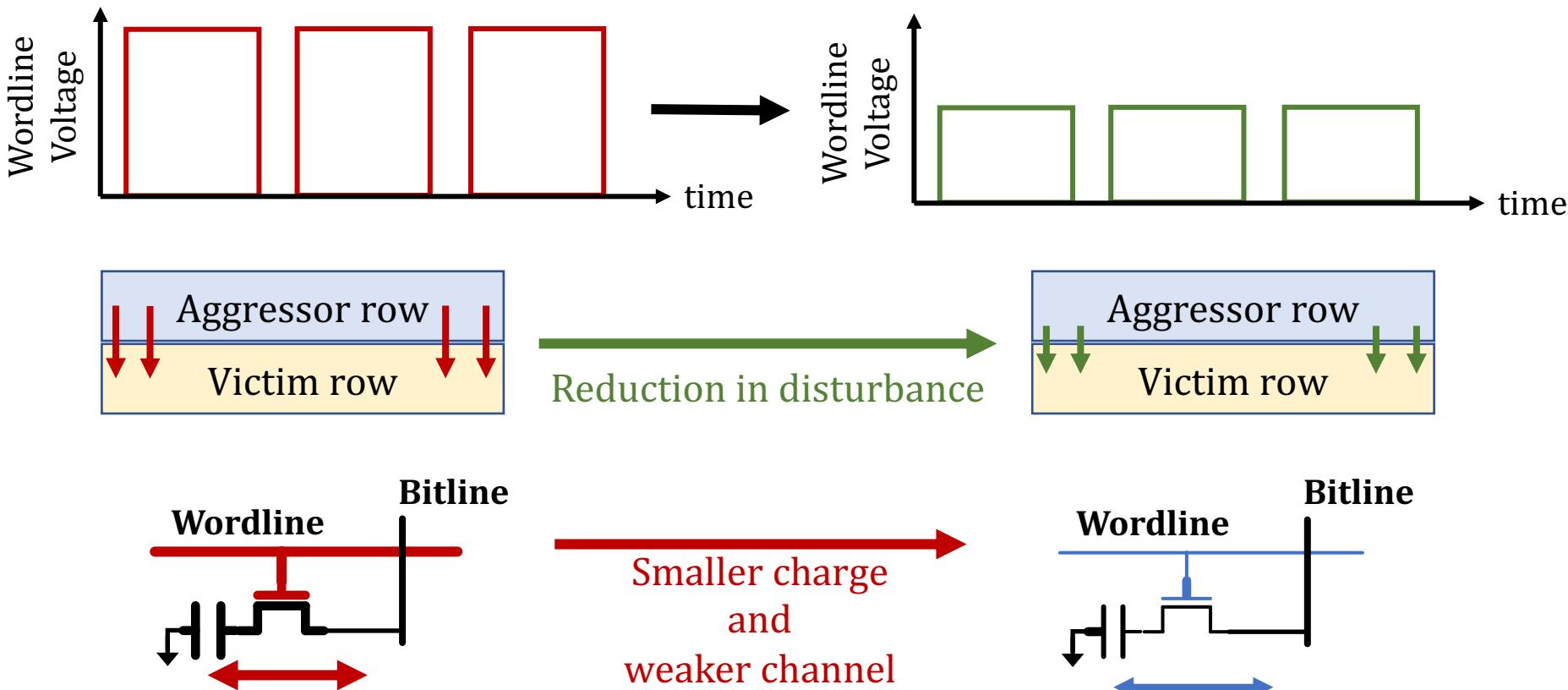


Repeatedly toggling
wordline voltage
leads to
RowHammer bitflips



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

Core Contributions

1

building a
detailed understanding
of DRAM read disturbance

2

leveraging **insights into**
modern DRAM chips
and **memory controllers**

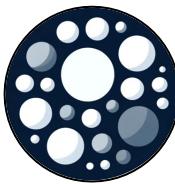
3

devising **novel solutions**
that do not require
proprietary knowledge
of DRAM chip internals



Svärd: Leveraging
heterogeneity



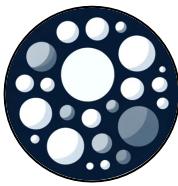


Tested DRAM Chips

**144 DDR4 DRAM chips from
SK Hynix, Micron, and Samsung**

All rows in four different banks

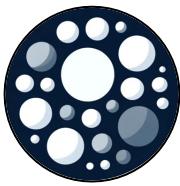
Mfr.	DIMM ID	# of Chips	Density Die Rev.	Chip Org.	Date (ww-yy)
Mfr. H (SK Hynix)	H0	8	16Gb – A	x8	51-20
	H1, H2, H3	3 × 8	16Gb – C	x8	48-20
	H4	8	8Gb – D	x8	48-20
Mfr. M (Micron)	M0	4	16Gb – E	x16	46-20
	M1, M3	2 × 16	8Gb – B	x4	N/A
	M2	16	16Gb – E	x4	14-20
	M4	4	16Gb – B	x16	26-21
Mfr. S (Samsung)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

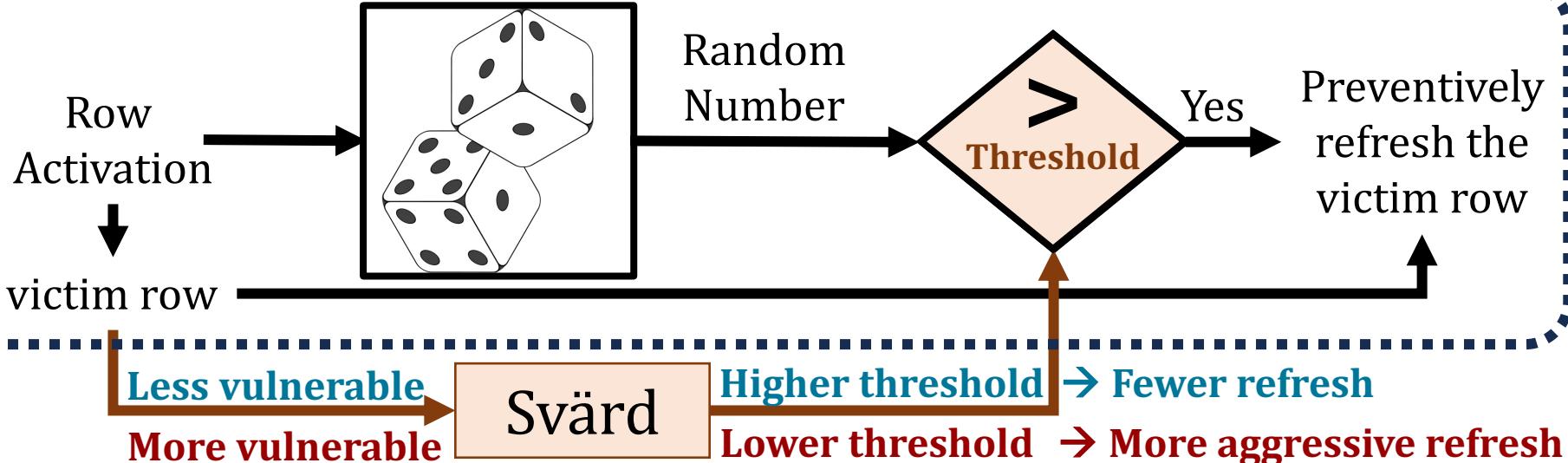
- **Significant and irregular variation** in read disturbance vulnerability **across DRAM rows**
- Read disturbance solutions:
 - Configured **for the worst row**
 - **Overprotect** many rows
 - Incur **large performance overheads**
- **Key Idea: Dynamically tune the aggressiveness of existing solutions to the victim row's read disturbance vulnerability**
- **Svärd: Spatial Variation-Aware Read Disturbance Defenses**
Fewer preventive actions (e.g., refresh) for stronger rows

*Svärd is the Swedish word for sword, a weapon with a long blade for cutting or thrusting [[Merriam-Webster](#)]



Integrating Svärd with Existing Solutions

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]

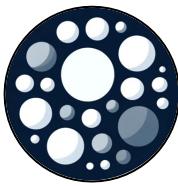
Performance Evaluation

- 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

BlockHammer

[Yaglikci+, HPCA'21]

4.9x

Hydra

[Qureshi+, ISCA'22]

1.1x

PARA

[Kim+, ISCA'14]

2.0x

RRS

[Saileshwar+, ASPLOS'22]

4.8x

- Svärd's hardware complexity:

No
Additional
Latency

In-DRAM
Implementation
0.006%

In-Processor
Implementation
0.027%

Core Contributions

1

building a
detailed understanding
of DRAM read disturbance

2

leveraging **insights into**
modern DRAM chips
and **memory controllers**

3

devising **novel solutions**
that do not require
proprietary knowledge
of DRAM chip internals



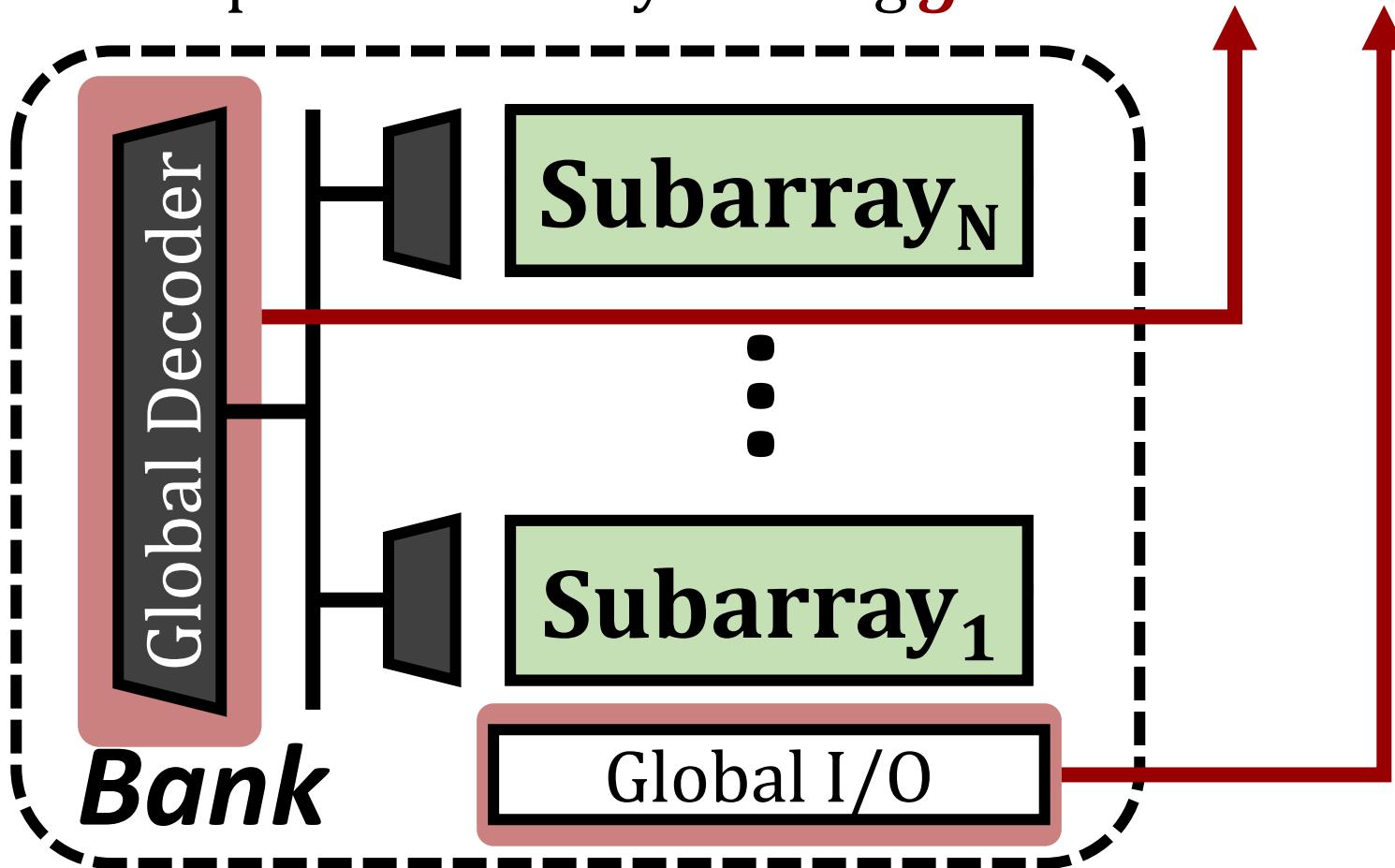
HiRA: Parallelizing
Preventive Actions



Subarray-Level Parallelism

Each subarray is mostly independent...

- except occasionally sharing *global structures*

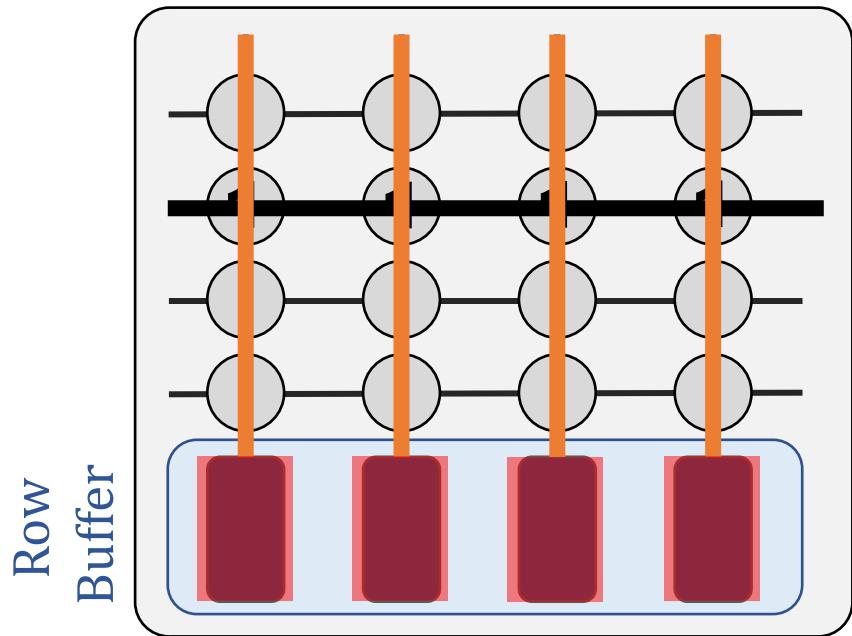


Yoongu Kim, et al., ["A Case for Exploiting Subarray-Level Parallelism \(SALP\) in DRAM,"](#) in *ISCA*, 2012

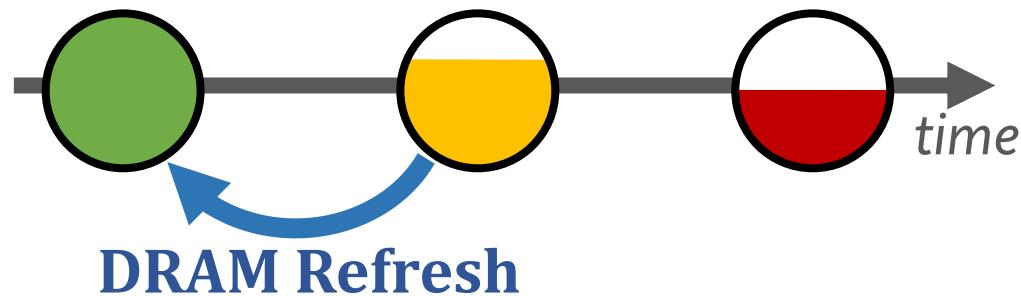


DRAM Refresh

DRAM Subarray



Fully charged



DRAM cells **leak charge** over time

DRAM Refresh **is the key maintenance operation** to **avoid bit flips** due to charge leakage

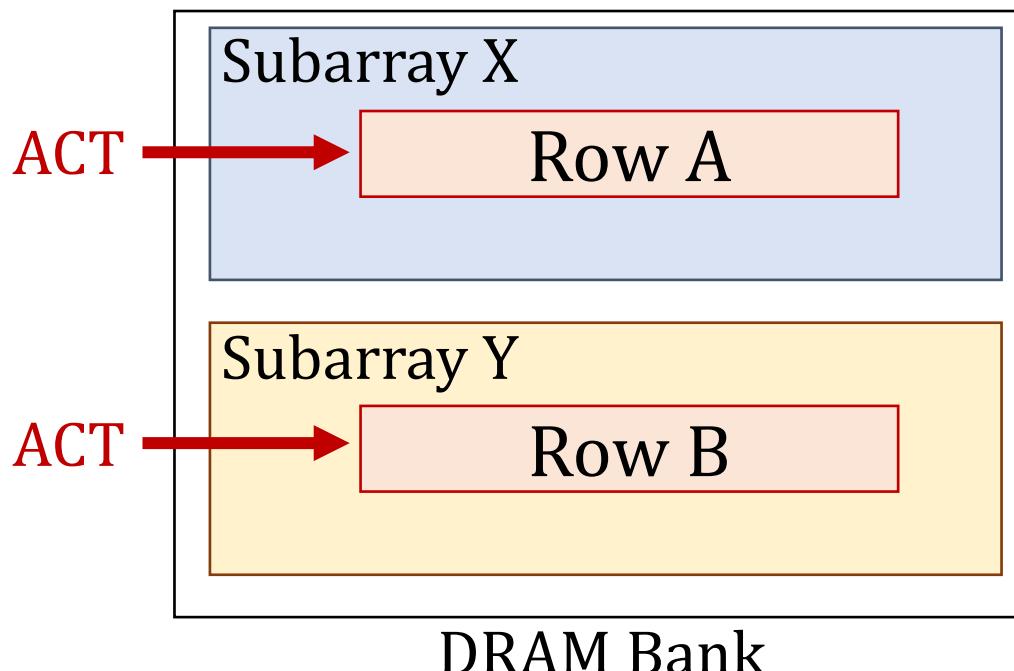
Activates (ACT) a row and precharges (PRE) the bank

Problem: DRAM Refresh **blocks** accesses to the **whole bank / rank**



HiRA: Hidden Row Activation – Key Insight

Concurrently activating two rows
in **different subarrays** of the **same bank**
can **refresh one row** while **opening the other row**

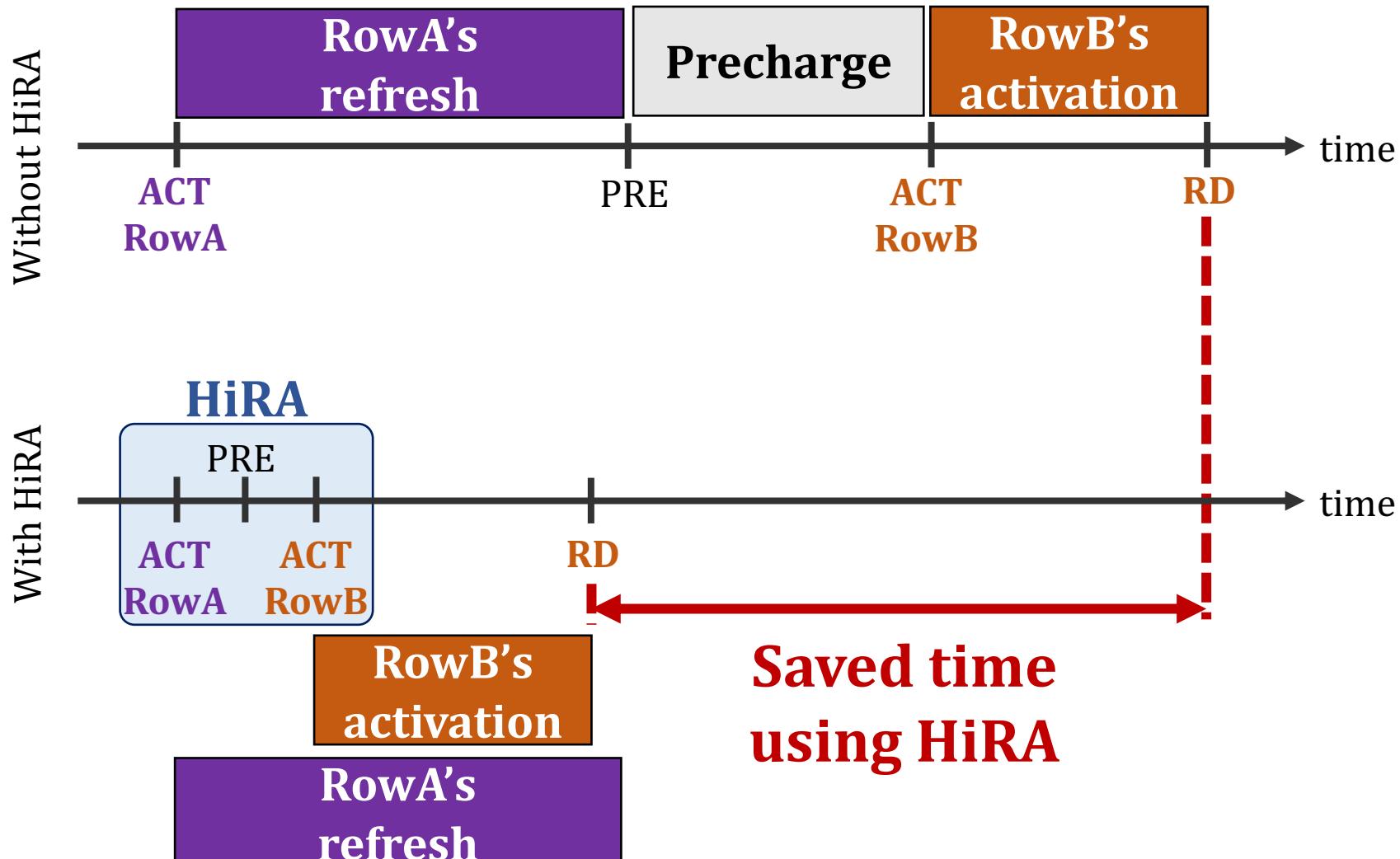


Refreshes RowA
concurrently with
opening RowB



HiRA: Hidden Row Activation

Refresh RowA concurrently with Activating RowB





HiRA in Off-the-Shelf 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.

- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**
- **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-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 hammer count** needed to induce the first RowHammer bitflip



Key Results of HiRA

- **Performance Evaluation**
 - **3.7x speedup** by reducing time spent on **RowHammer-preventive refreshes**
 - **12.6% speedup** by reducing time spent on **periodic refreshes**
- **Hardware Complexity Analysis**
 - **Chip area cost of 0.0023%** of a processor die
 - **No additional latency overhead**

Core Contributions

1

building a
detailed understanding
of DRAM read disturbance

2

leveraging **insights into**
modern DRAM chips
and memory controllers

3

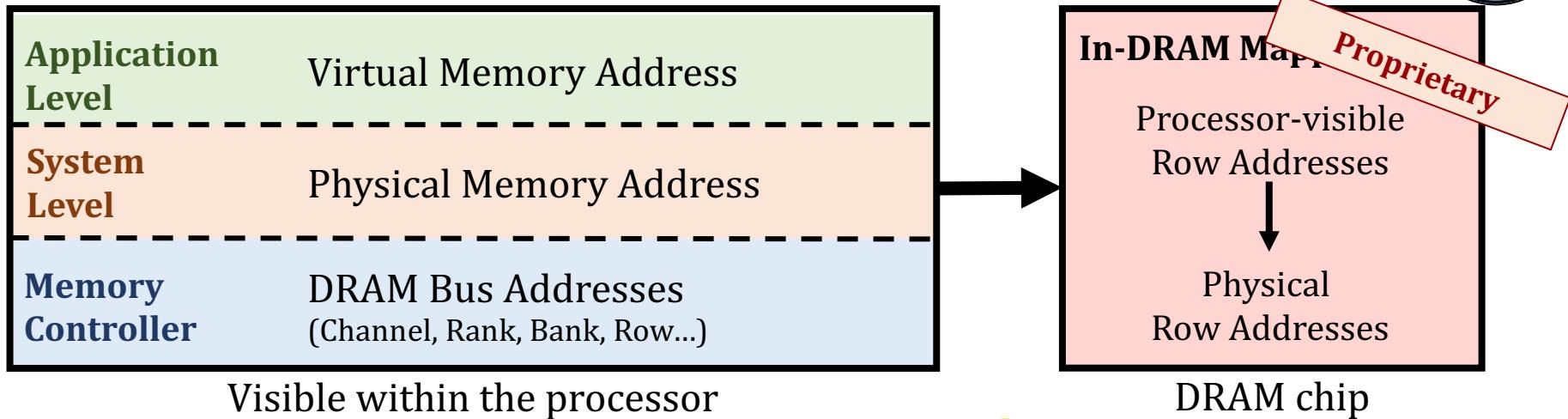
devising **novel solutions**
that do not require
proprietary knowledge
of DRAM chip internals



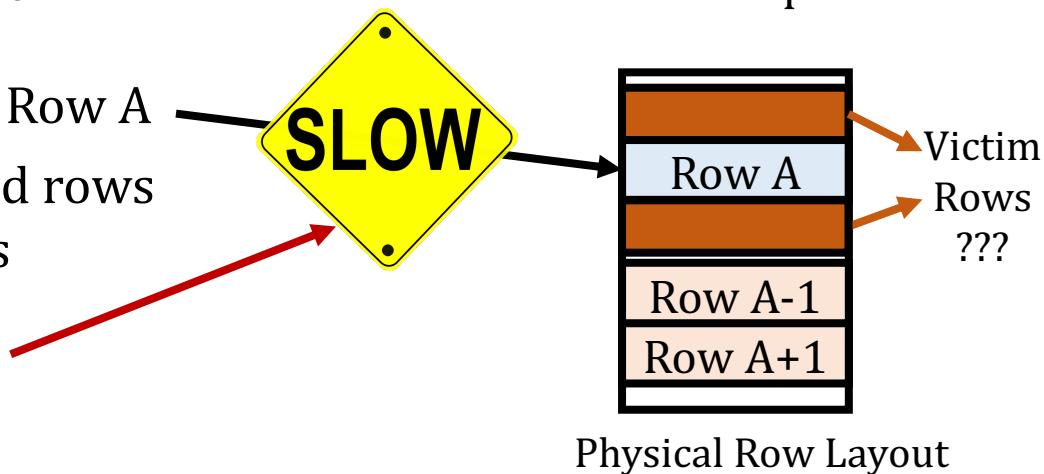
BlockHammer: Selectively
throttling unsafe accesses



BlockHammer: Key Idea



- 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 (1/2)

- **Qualitative** comparison against **14 mechanisms**

- Comprehensive protection
- Compatibility with commodity DRAM chips
- Scalability with worsening RowHammer vulnerability
- Deterministic protection

BlockHammer is the **only mechanism** that satisfies
all **four properties**

- **Quantitative** comparison against **6 state-of-the-art mechanisms**

- PARA [Kim+, ISCA'14]
- ProHIT [Son+, DAC'17]
- MRLoc [You+, DAC'19]
- CBT [Seyedzadeh+, ISCA'18]
- TWiCe [Lee+, ISCA'19]
- Graphene [Park+, MICRO'20]

BlockHammer is **low cost** and **competitive**



BlockHammer: Evaluation (2/2)

- 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 (2/2)

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

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**

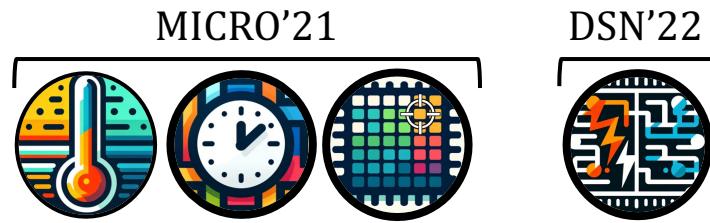
Conclusion

We can mitigate DRAM read disturbance efficiently and scalably by

1 building a
detailed understanding
of DRAM read disturbance

2 leveraging insights into
modern DRAM chips
and memory controllers

3 devising novel solutions
that do not require
proprietary knowledge
of DRAM chip internals



More Details and Discussion on YouTube

SAFARI Live Seminars in Computer Architecture

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



Temperature



Memory access patterns



Victim cell's physical location



Voltage



SPEAKER
Abdullah Giray Yağlıkçı
SAFARI Research Group, ETH Zurich

JAN 17, 2024 5:00PM CET

SAFARI Live Seminars in Computer Architecture

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



Leveraging Heterogeneity



Throttling Unsafe Accesses



Parallelizing Preventive Actions



SPEAKER
Abdullah Giray Yağlıkçı
SAFARI Research Group, ETH Zurich

JAN 22, 2024 5:00PM CET



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



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

Industry Solutions to RowHammer

Per-Row Activation Counters in DRAM

- Tanj Bennett, Stefan Saroiu, Alec Wolman, and Lucian Cojocar,
“Panopticon: A Complete In-DRAM Rowhammer Mitigation”
DRAMSec Workshop (co-located with ISCA), 2021.

Panopticon: A Complete In-DRAM Rowhammer Mitigation

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

Accurate RowHammer detection using **an activation counter per DRAM row**
stored within the **DRAM array** leveraging **high density → low cost**

RowHammer in 2023: SK Hynix

ISSCC 2023 / SESSION 28 / HIGH-DENSITY MEMORIES /

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

Woongrae Kim, Chulmoon Jung, Seongnyuh Yoo, Duckhwa Hong,
Jeongjin Hwang, Jungmin Yoon, Ohyong Jung, Joonwoo Choi, Sanga Hyun,
Mankeun Kang, Sangho Lee, Dohong Kim, Sanghyun Ku, Donhyun Choi,
Nogeun Joo, Sangwoo Yoon, Junseok Noh, Byeongyong Go, Cheolhoe Kim,
Sunil Hwang, Mihyun Hwang, Seol-Min Yi, Hyungmin Kim, Sanghyuk Heo,
Yeonsu Jang, Kyoungchul Jang, Shinho Chu, Yoonna Oh, Kwidong Kim,
Junghyun Kim, Soohwan Kim, Jeongtae Hwang, Sangil Park, Junphyo Lee,
Inchul Jeong, Joohwan Cho, Jonghwan Kim

SK hynix Semiconductor, Icheon, Korea



RowHammer in 2023: Samsung

DSAC: Low-Cost Rowhammer Mitigation Using In-DRAM Stochastic and Approximate Counting Algorithm

Seungki Hong Dongha Kim Jaehyung Lee Reum Oh
Changsik Yoo Sangjoon Hwang Jooyoung Lee

DRAM Design Team, Memory Division, Samsung Electronics

[**https://arxiv.org/pdf/2302.03591v1.pdf**](https://arxiv.org/pdf/2302.03591v1.pdf)

DDR5 Update in 2024

- DRAM implements per-row activation counters
similar to Panopticon
- DRAM asserts a back-off signal when refresh is needed
similar to SMD, Mithril+, and Panopticon
- Memory controller issues refresh upon back-off signal

**JEDEC
STANDARD**

DDR5 SDRAM

JESD79-5C_v1.30

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

April 2024

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION



JEDEC, "[**JESD79-5C_v1.30: DDR5 SDRAM Specification**](#)," April, 2024.

Understanding PRAC

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

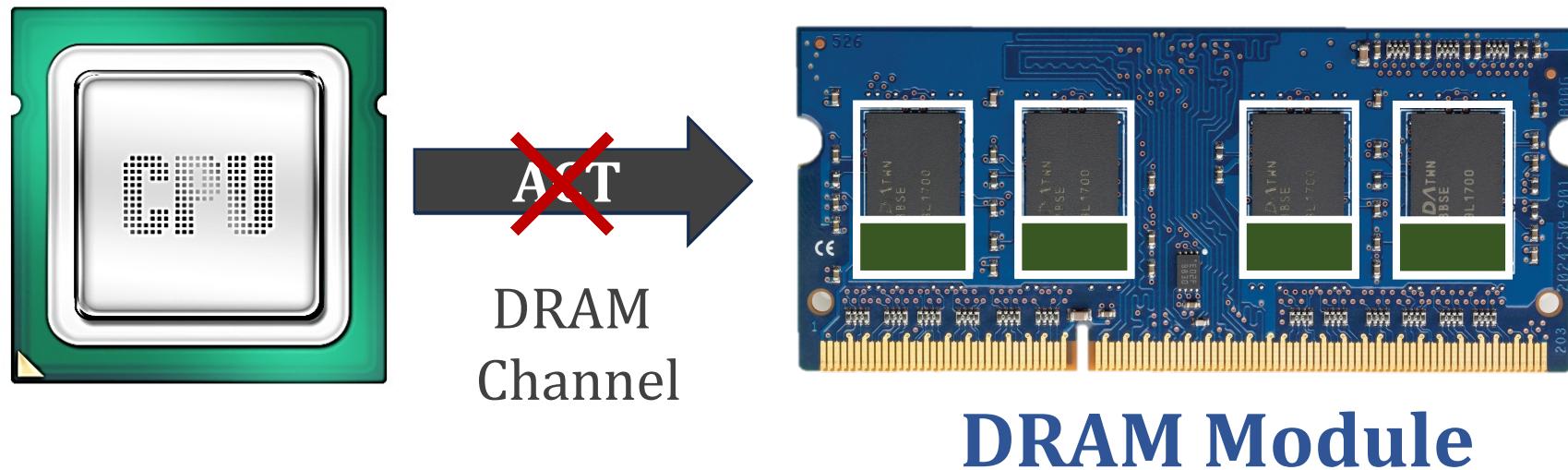
[†]TOBB University of Economics and Technology

presented in DRAMSec 2024

- PRAC is the latest update to DDR5 in April 2024
- We show that
 - PRAC is secure and has low performance, energy, and area overheads for benign workloads
 - PRAC can be exploited by memory performance attacks (up to **79% reduction** in DRAM chip's throughput)

Industry Solutions to Read Disturbance: When To Refresh? (I)

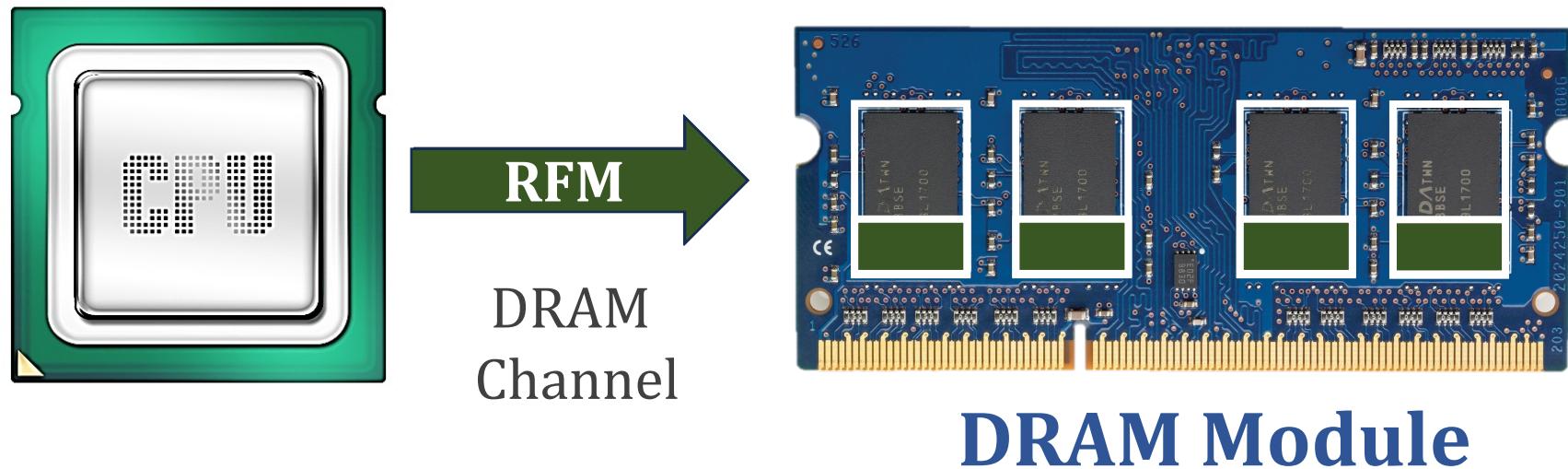
Preventive refresh is a **blocking** operation



Memory controller **cannot activate** a row
while **a preventive refresh blocks the bank**

Industry Solutions to Read Disturbance: When To Refresh? (II)

Earlier JEDEC DDR5 specifications introduce
Refresh Management (RFM) commands



Memory controller sends an **RFM command**
to allow time for preventive refreshes

Industry Solutions to Read Disturbance

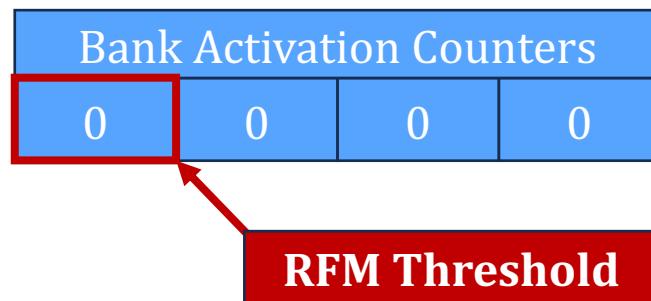
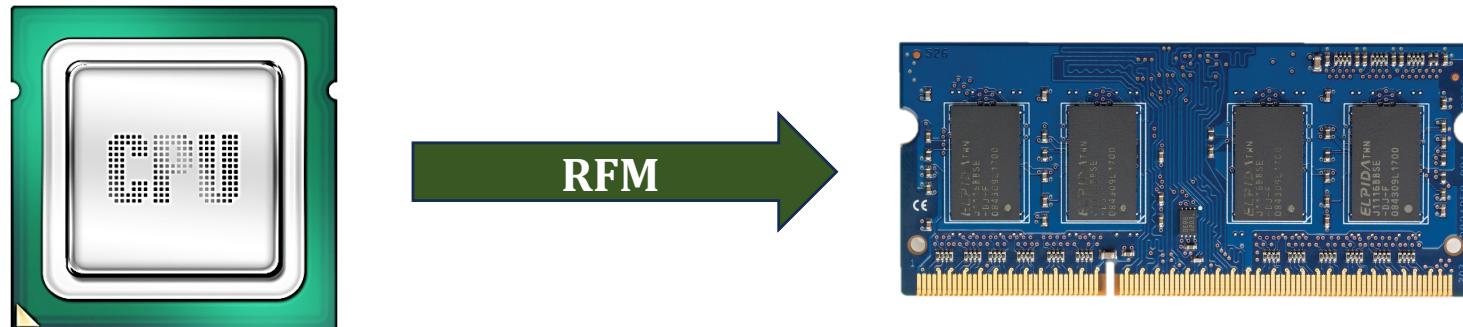
Periodic Refresh Management (PRFM)

Memory controller **periodically** issues RFM commands

Per Row Activation Counting and Back-Off (PRAC)

DRAM chip **tracks** row activations and **requests** RFMs by sending **back-off** signals

Industry Solutions to Read Disturbance: Periodic Refresh Management (PRFM)



PRFM tracks activations with **low accuracy**,
causing **high number** of preventive refreshes,
leading to **large** performance and energy overheads



Industry Solutions to Read Disturbance

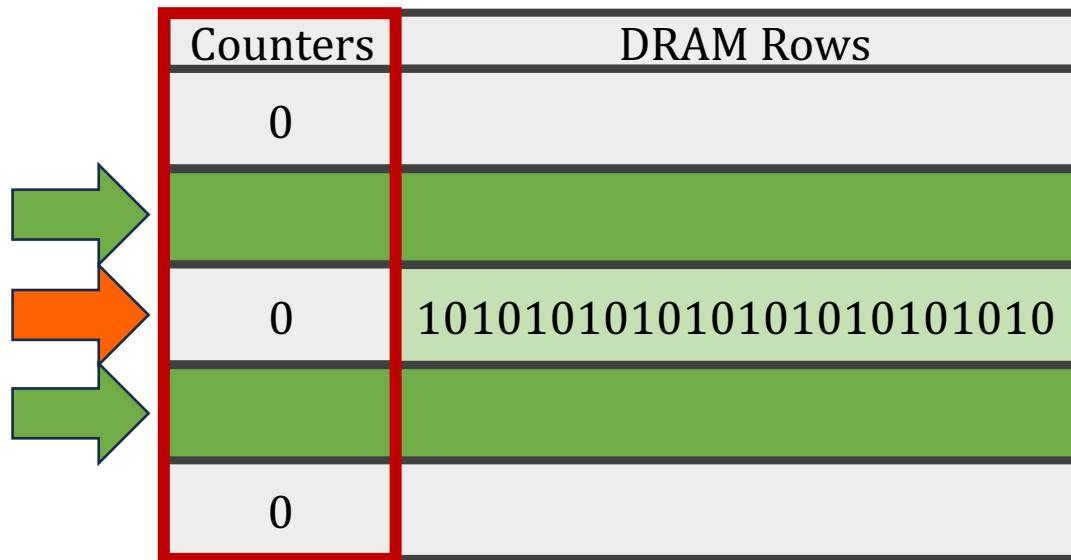
Periodic Refresh Management (PRFM)

Memory controller **periodically** issues RFM commands

Per Row Activation Counting and Back-Off (PRAC)

DRAM chip **tracks** row activations and **requests** RFMs by sending **back-off**

Industry Solutions to Read Disturbance: Per Row Activation Counting



PRAC allows accurate tracking of aggressor row activations

Industry Solutions to Read Disturbance: Per Row Activation Counting DRAM Timings

Counters	DRAM Rows
0	101010101010101010101010
0	101010101010101010101010
0	101010101010101010101010
0	101010101010101010101010
0	101010101010101010101010

Row counter updates are **not** completely parallelized with DRAM access

PRAC increases row-close time (t_{RP}) by **~140%**



Industry Solutions to Read Disturbance: Per Row Activation Counting DRAM Timings

Timing parameter changes for DDR5-3200AN speed bin
[JEDEC JESD79-5C, April 2024]

t_{RP} : +21ns (+140%)

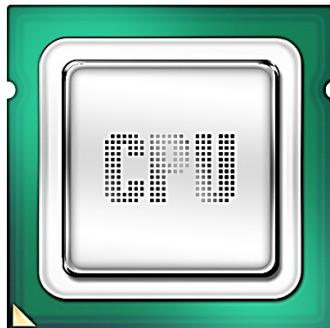
t_{RAS} : -16ns (-50%)

t_{RTP} : -2.5ns (-33%)

t_{WR} : -20ns (-66%)

t_{RC} : +5ns (+10%)

Industry Solutions to Read Disturbance: Per Row Activation Counting (PRAC)



RFM



Row Counters

0

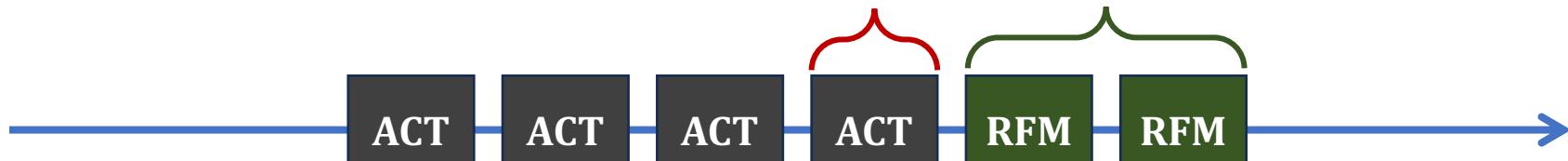
0

0

Back-Off Threshold

normal traffic recovery
(180 ns) (N RFMs)

PRAC-N



DRAM Commands

Performance Comparison: Industry Solutions

1 PRFM

Memory controller **periodically** issues RFM

2 PRAC-N

Memory controller issues **N** RFMs each with **back-off**

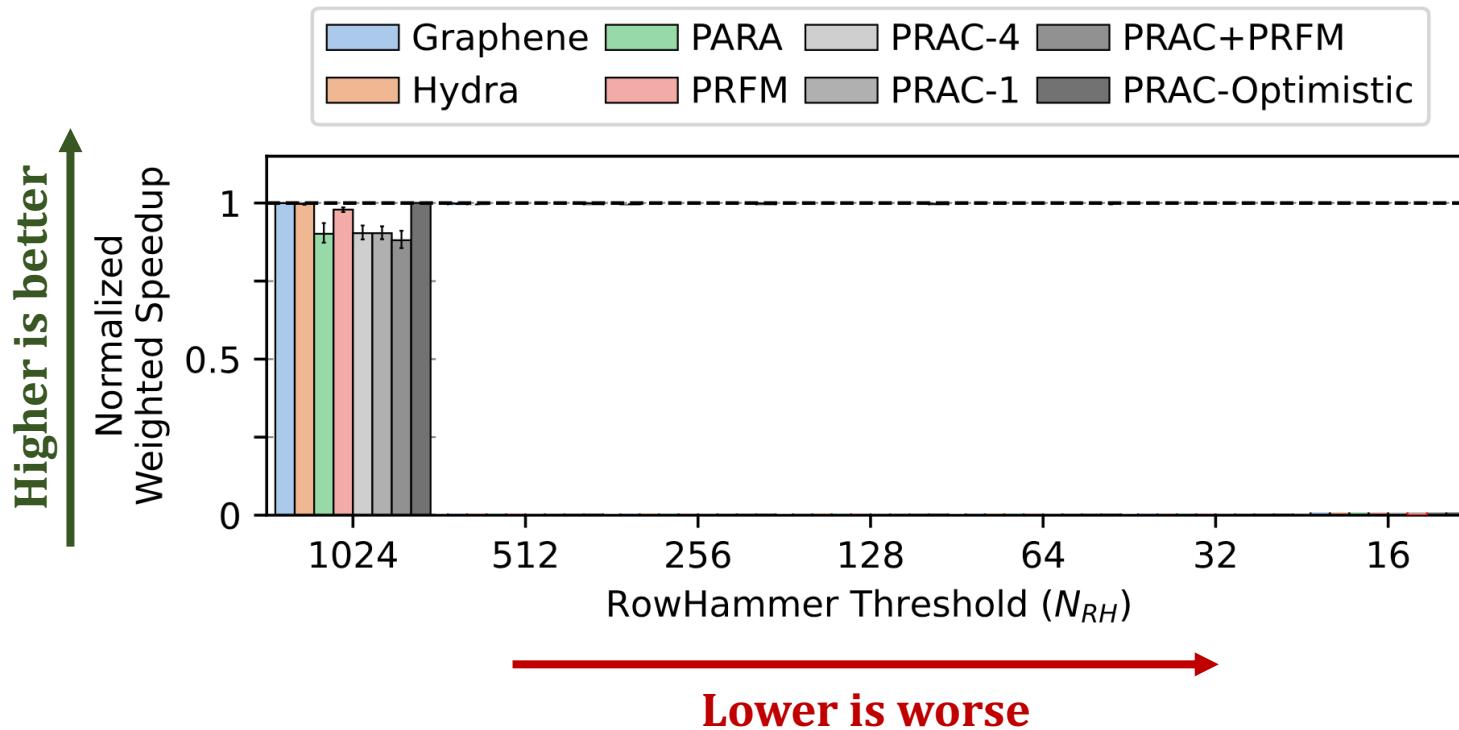
3 PRAC+PRFM

Memory controller issues RFM **periodically** and with **back-offs**

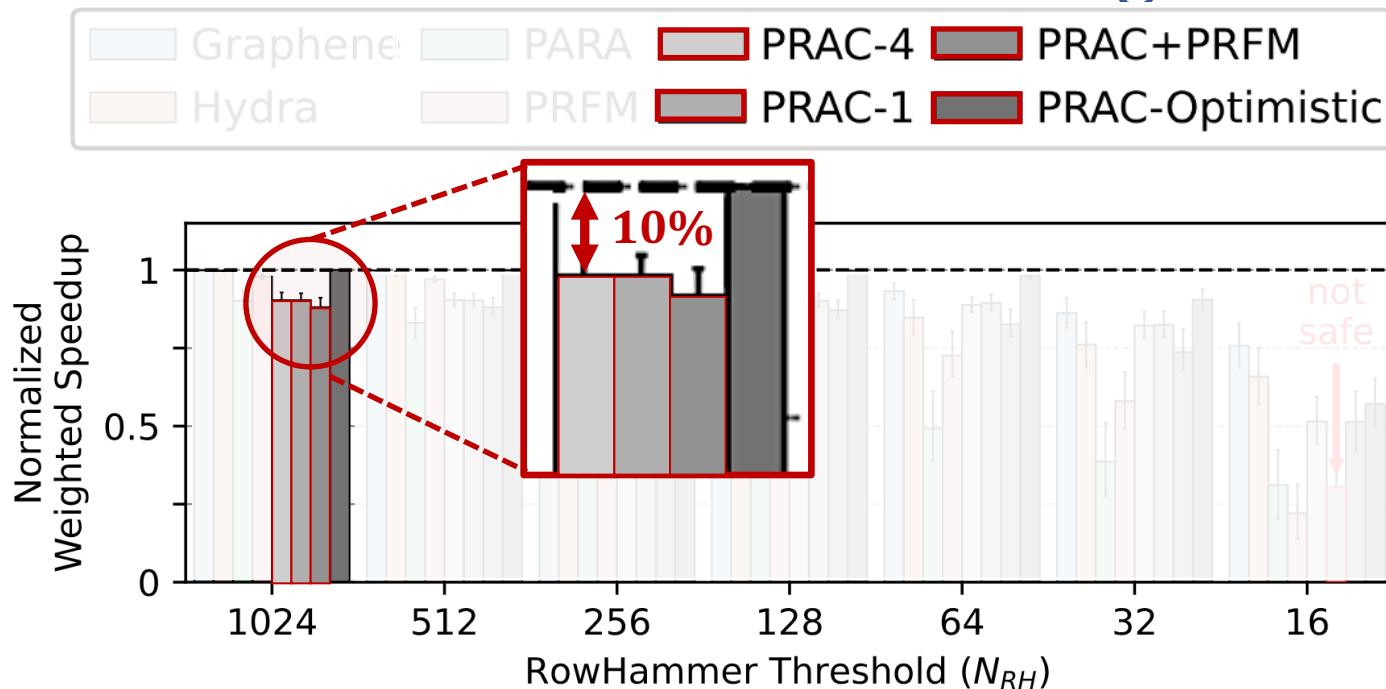
4 PRAC-Optimistic

PRAC-4 with **no** change in DRAM timing parameters

Experimental Results: Performance Overhead and Its Scaling

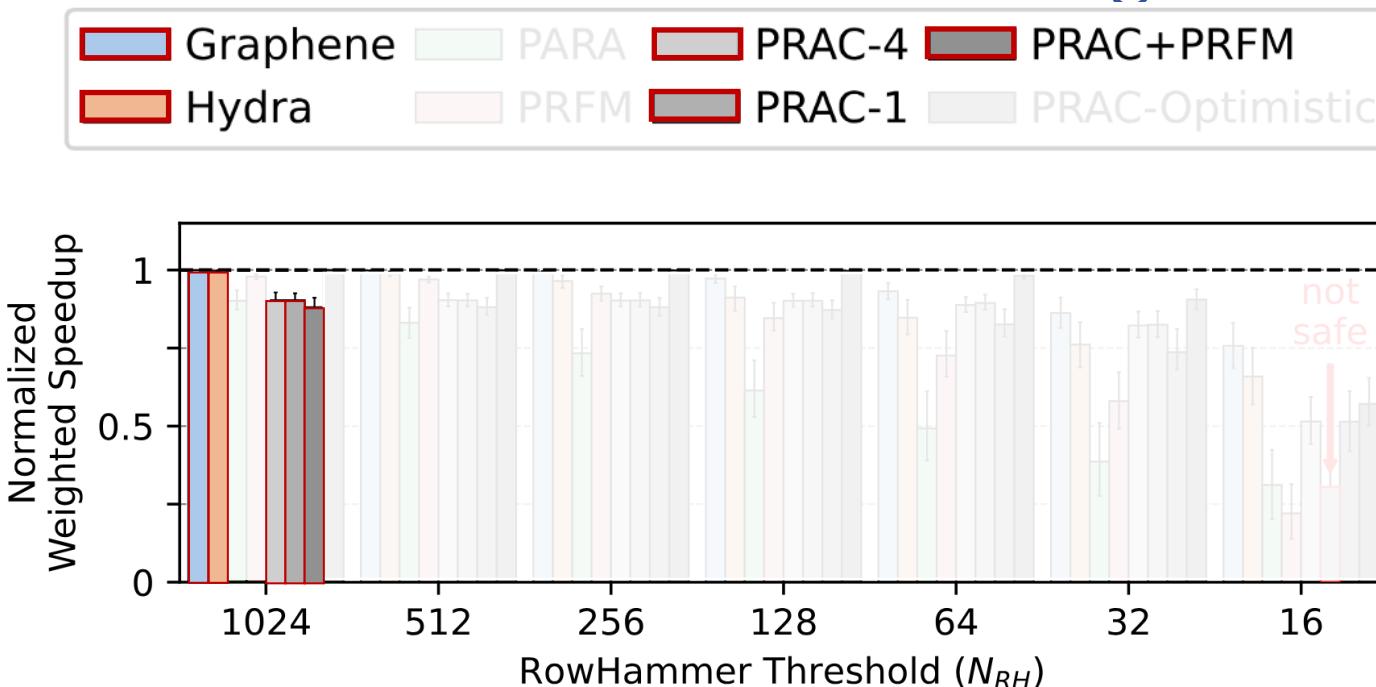


Experimental Results: Performance Overhead and Its Scaling



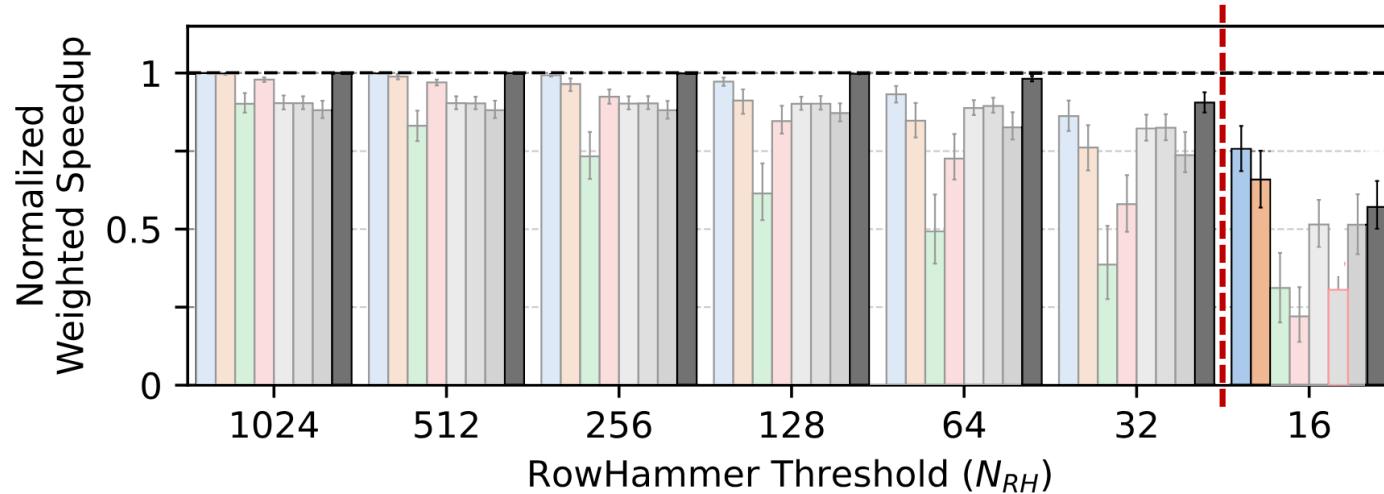
PRAC has **non-negligible** performance overhead (**10%**)
due to **increased** access latency

Experimental Results: Performance Overhead and Its Scaling



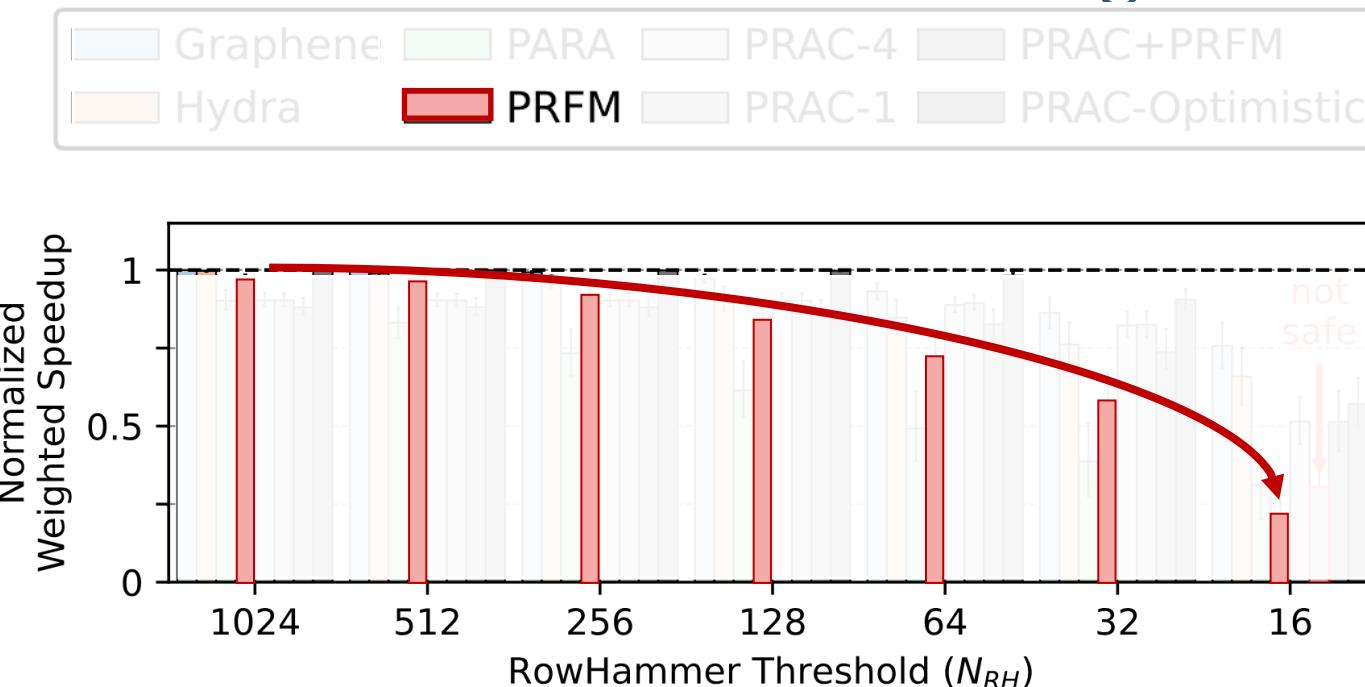
Graphene and **Hydra** outperform **PRAC**
at relatively high N_{RH} values

Experimental Results: Performance Overhead and Its Scaling



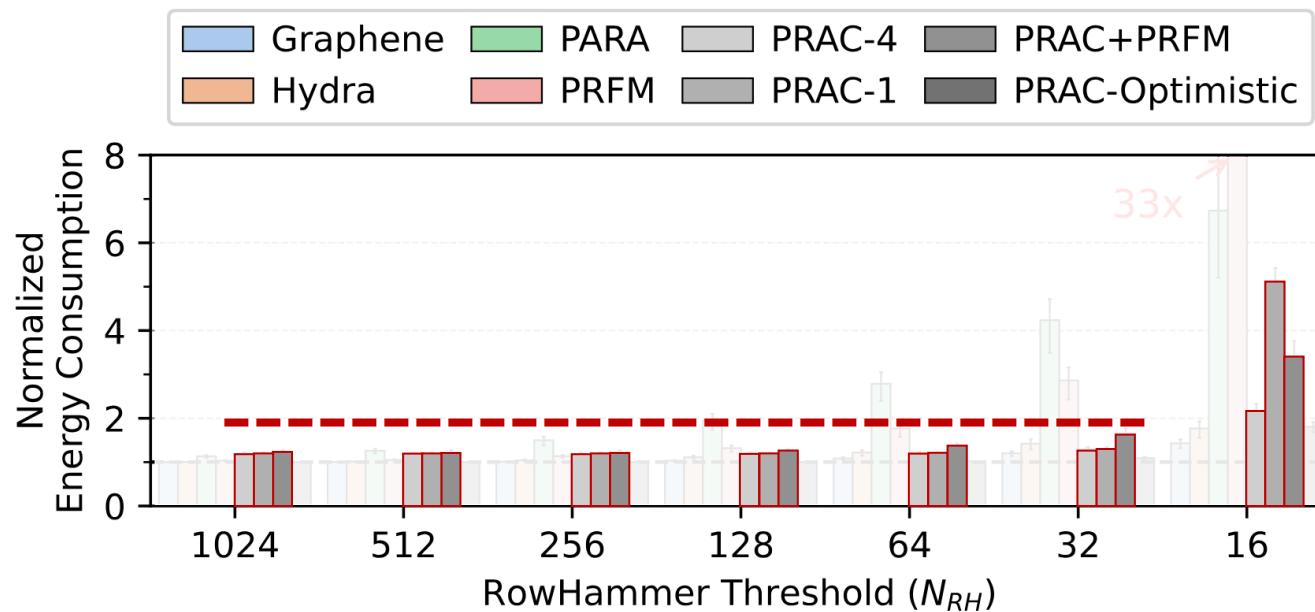
**PRAC-Optimistic outperforms all evaluated mitigation mechanisms
(above N_{RH} of 32)**

Experimental Results: Performance Overhead and Its Scaling



**PRFM's system performance overheads
significantly increase (by 37x) as N_{RH} decreases**

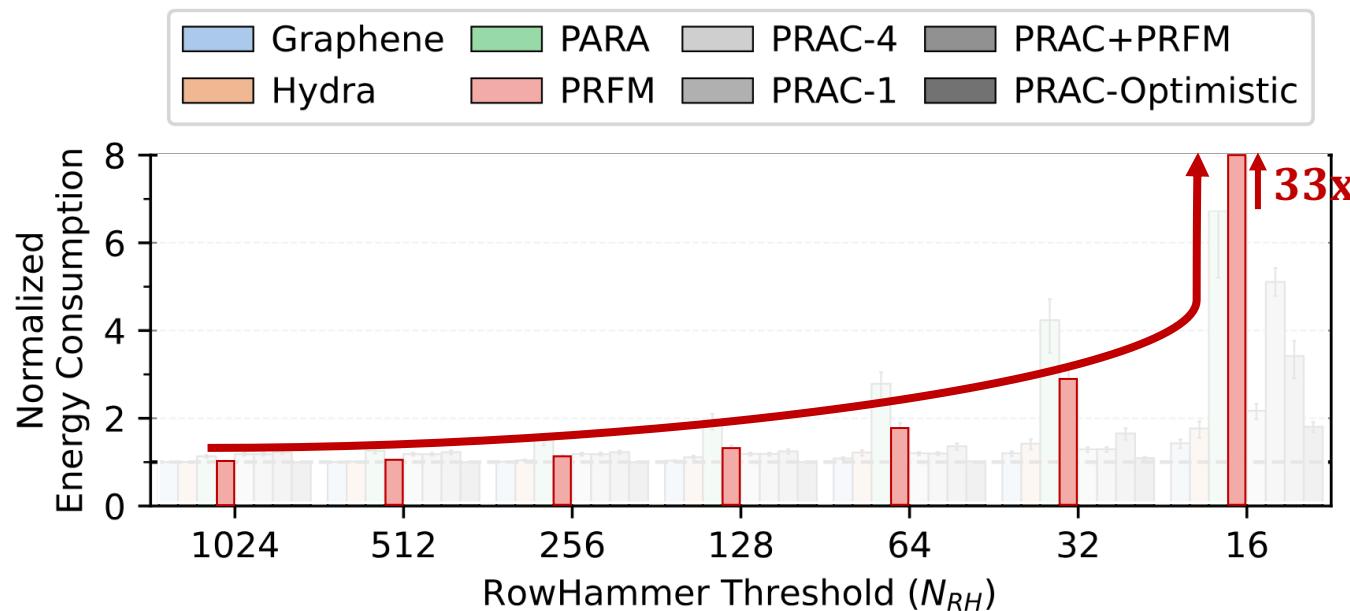
Experimental Results: DRAM Energy Overhead and Its Scaling



Above N_{RH} of 32, **PRAC** overhead only
slightly increases due to **timely** preventive refreshes

Below N_{RH} of 32, **PRAC** overhead **significantly** increases
due to conservative thresholds against a **wave attack**

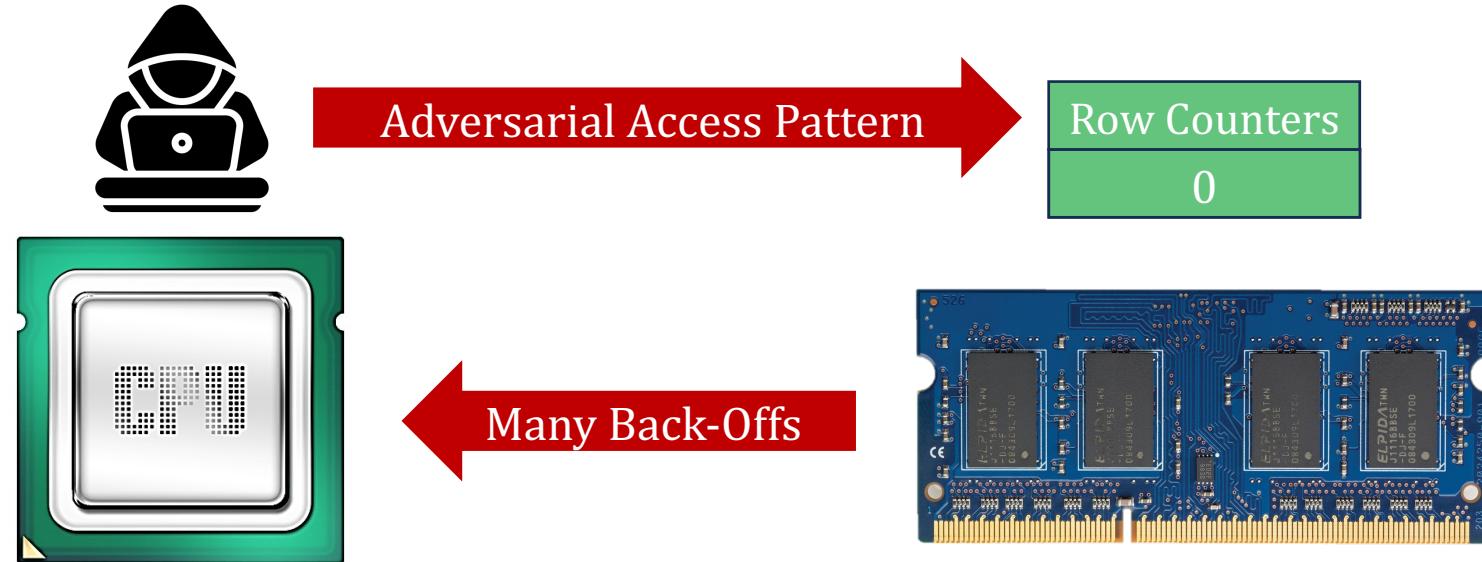
Experimental Results: DRAM Energy Overhead and Its Scaling



PRFM's DRAM energy overhead
significantly increase (to 33x) as N_{RH} decreases

Memory Performance Attacks

Access pattern to trigger **most** back-offs
with **fewest** activations possible by targeting a single row



Mathematically hogs up to **79% of DRAM throughput**
of future DRAM chips

Degrades system performance by up to **65%** (53% on average)

More in the Paper

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

Oğuzhan Canpolat^{§†}

[§]ETH Zürich

A. Giray Yağlıkçı[§]

Oğuz Ergin[†]

[†]TOBB University of Economics and Technology

Geraldo F. Oliveira[§]

Onur Mutlu[§]

Ataberk Olgun[§]

We present the first rigorous security, performance, energy, and cost analyses of the state-of-the-art on-DRAM-die read disturbance mitigation method, widely known as Per Row Activation Counting (PRAC), with respect to its description in the updated (as of April 2024) JEDEC DDR5 specification. Unlike prior state-of-the-art that advises the memory controller to periodically issue a DRAM command called refresh management (RFM), which provides the DRAM chip with time to perform its countermeasures, PRAC introduces a new back-off signal. PRAC's back-off signal propagates from the DRAM chip to the

data integrity of other physically close but unaccessed DRAM rows. RowHammer [1] is a prime example of DRAM read disturbance, where a DRAM row (i.e., victim row) can experience bitflips when at least one nearby DRAM row (i.e., aggressor row) is repeatedly activated (i.e., hammered) [1, 3–69] more times than a threshold, called the *minimum hammer count to induce the first bitflip* (N_{RH}). RowPress [70] is another prime example of DRAM read disturbance that amplifies the effect of RowHammer and consequently reduces N_{RH} .

Open Sourced



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

Screenshot of the GitHub repository page for `ramulator2`.

The repository has 13 forks and 175 stars.

The main branch has 54 commits from `RichardLuo79`, including merges and fixes for various files like `perf_comparison`, `resources/gem5_wrappers`, and `src`.

The repository description states:

Ramulator 2.0 is a modern, modular, extensible, and fast cycle-accurate DRAM simulator. It provides support for agile implementation and evaluation of new memory system designs (e.g., new DRAM standards, emerging RowHammer mitigation techniques). Described in our paper https://people.inf.ethz.ch/omutlu/pub/Ramulator2_arxiv23.pdf

Tags: simulation, memory, dram

Upcoming Papers

Upcoming Paper I: BreakHammer

Leveraging Adversarial Detection to Enable Scalable and Low Overhead RowHammer Mitigations

Oğuzhan Canpolat^{§†}

A. Giray Yağlıkçı[§]

Ataberk Olgun[§]

İsmail Emir Yüksel[§]

Yahya Can Tuğrul^{§†}

Konstantinos Kanellopoulos[§]

Oğuz Ergin[†]

Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB University of Economics and Technology

^{*}SAFARI Research Group

available on arXiv and under submission

- **Key Idea:** Throttling memory accesses of threads that trigger mitigation mechanisms repeatedly
- **BreakHammer:**
 - Detects the threads that repeatedly trigger the mitigation mechanisms
 - Limits their on-the-fly memory request counts and MSHRs
 - Near-zero area overhead and no additional memory access latency
- **Evaluation:**
 - Improves **system performance** by **48.7%** on average (**105.5%** max)
 - Reduces the **maximum slowdown** by **14.6%** on average

Upcoming Paper II: Self-Managing DRAM

Self-Managing DRAM: A Low-Cost Framework for Enabling Autonomous and Efficient in-DRAM Operations

Hasan Hassan[†]

Ataberk Olgun[†]

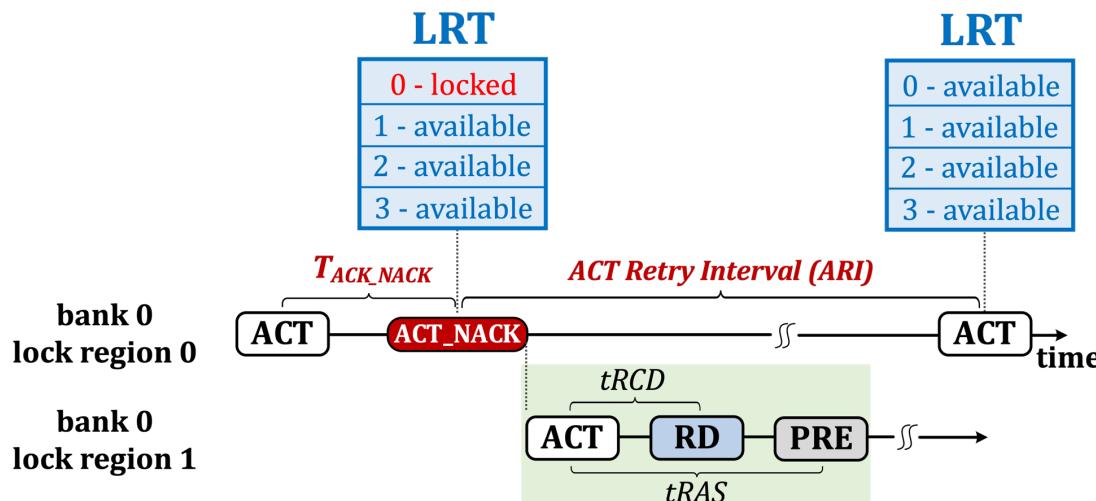
A. Giray Yağlıkçı
ETH Zürich

Haocong Luo

Onur Mutlu

available on arXiv and under submission

Enables new **in-DRAM** maintenance mechanisms without modifications other than the **ACT-NACK** signal from the DRAM chip to the memory controller



Three example DRAM maintenance operations:

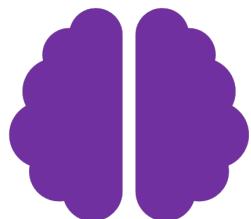
- ❖ Periodic refresh
- ❖ RowHammer-preventive refresh
- ❖ Memory scrubbing

Ongoing and Future Research on DRAM Read Disturbance



Deeper Understanding of Physics and Vulnerabilities

- Aging
- Online Profiling



Flexible and Intelligent Memory Chips, Interfaces, and Controllers

- In-field patchability
- DRAM-initiated pause
- Subarray-level parallelism
- Identify threads that cause the problem



Cross-Layer Communication

- Hardware-level detection
- System-level mitigation

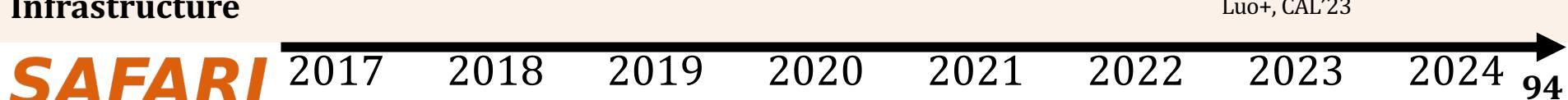
Ongoing works

Broader Research Interests

Broader Research Interests

Thesis Publications

Understanding Read Disturbance	Kim+ ISCA'20	Orosa+ MICRO'21	Yaglikci+ DSN'22	Olgun+ Disrupt'23 Luo+ ISCA'23	Olgun+, DSN'24 Yaglikci+ HPCA'24
Mitigating Read Disturbance		Yaglikci+ HPCA'21	Yaglikci+ MICRO'22		Olgun+, USENIX Sec'24 Bostanci+, HPCA'24
Read Disturbance Survey				Mutlu+ ASP-DAC'23	
Covert Channels and Security Primitives		Haj-Yahya+, ISCA'21 Olgun+, ISCA'21	Bostanci+ HPCA'22		Bostanci+ arXiv'24
Reducing DRAM Energy	Chang+ SIGMETRICS'17 Ghose+ SIGMETRICS'18	Koppula+ MICRO'19			
Reducing DRAM Latency		Hassan+ ISCA'19	Luo+ ISCA'20		
System Energy Saving Methods			Haj-Yahya+ ISCA'20	Haj-Yahya+ HPCA'22	
Processing Using Memory					Yuksel+, HPCA'24 Yuksel+, DSN'24 Oliveira+, HPCA'24
Experimental Infrastructure				Olgun+, TACO'23 Luo+, CAL'23	



Future Research

The Problem

Computing
is Bottlenecked by Data

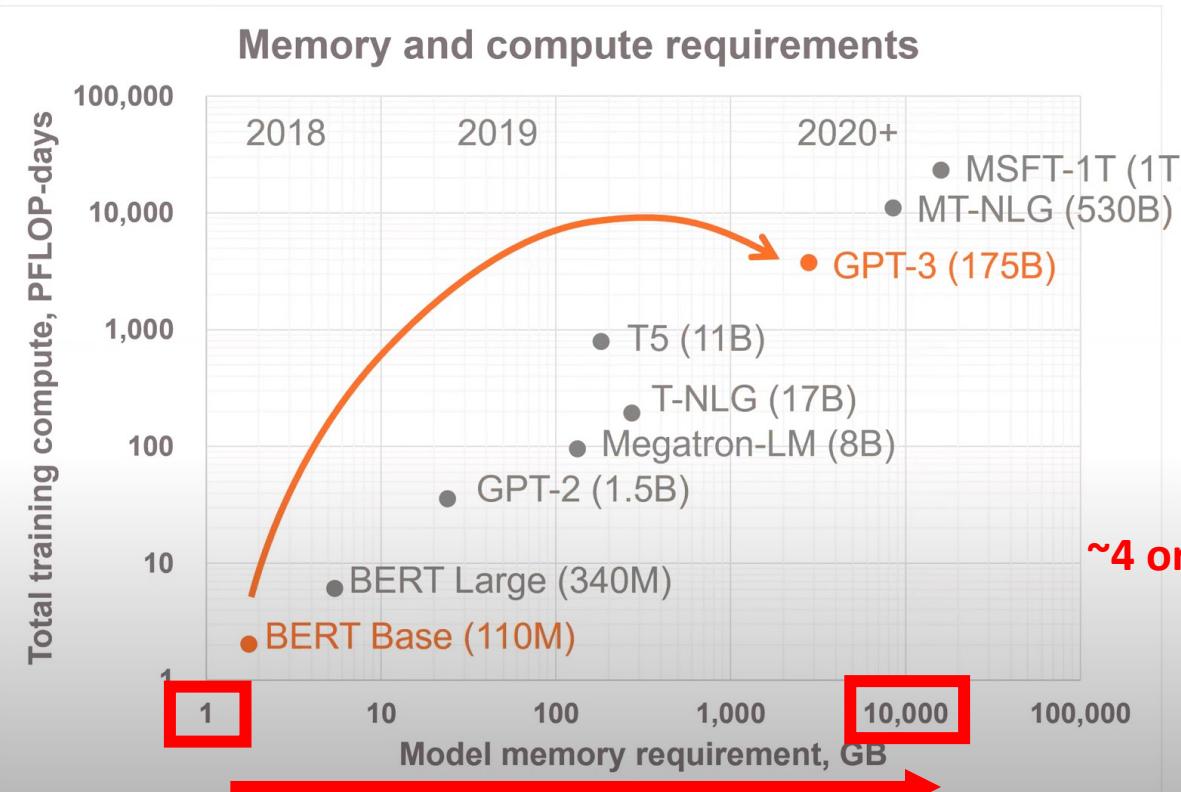
Data is Key for AI, ML, Genomics, ...

- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
 - We can generate more than we can process
 - We need to perform more sophisticated analyses on more data

Huge Demand for Performance & Efficiency



Exponential Growth of Neural Networks

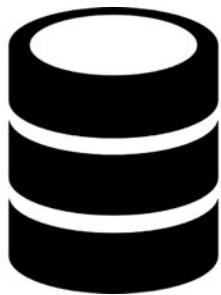


**1800x more compute
In just 2 years**

**Tomorrow, multi-trillion
parameter models**

**~4 orders of magnitude increase in
memory requirement in
just two years!**

Data is Key for Future Workloads



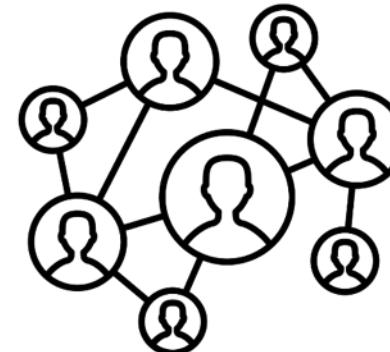
In-memory Databases

[Mao+, EuroSys'12;
Clapp+ (Intel), IISWC'15]



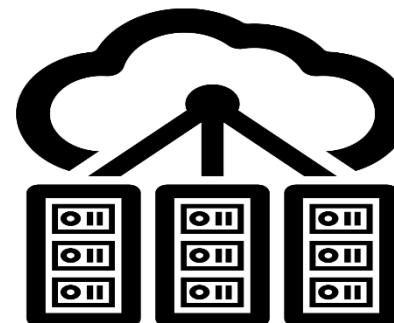
In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Graph/Tree Processing

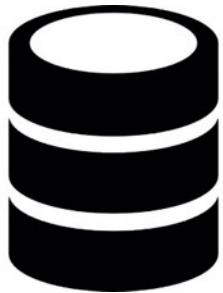
[Xu+, IISWC'12; Umuroglu+, FPL'15]



Datacenter Workloads

[Kanев+ (Google), ISCA'15]

Data Overwhelms Modern Machines



In-memory Databases



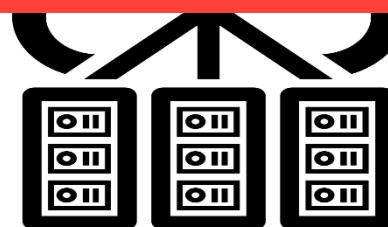
Graph/Tree Processing

Data → performance & energy bottleneck



In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Datacenter Workloads

[Kanев+ (Google), ISCA'15]

Data is Key for Future Workloads



Chrome

Google's web browser



TensorFlow Mobile

Google's machine learning
framework



Video Playback

Google's **video codec**



Video Capture

Google's **video codec**

Data Overwhelms Modern Machines



Chrome



TensorFlow Mobile

Data → performance & energy bottleneck



Video Playback

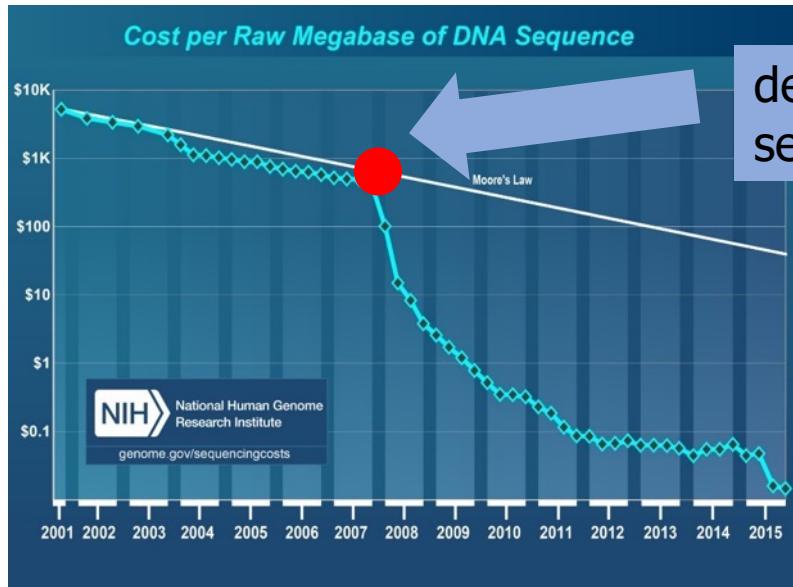
Google's **video codec**



Video Capture

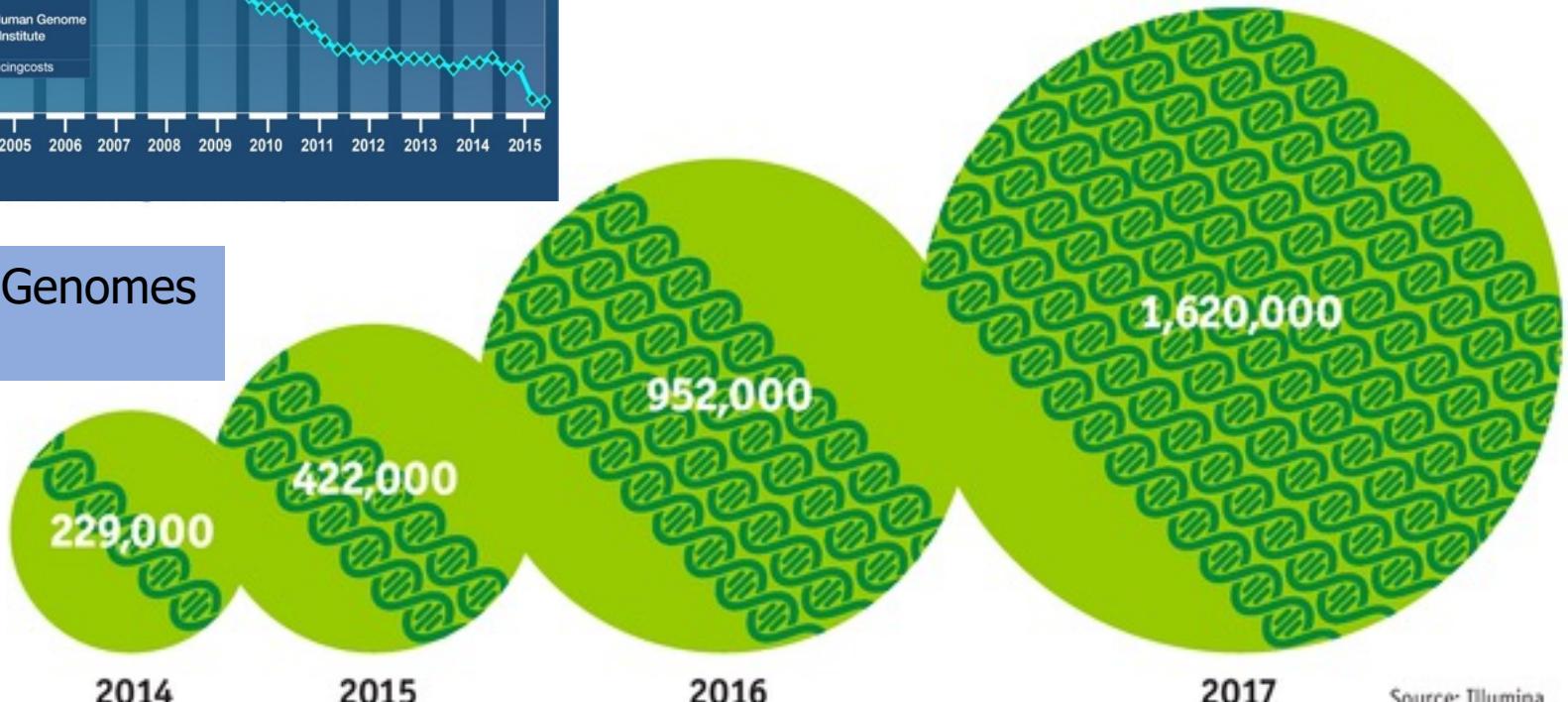
Google's **video codec**

Data is Key for Future Workloads



development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

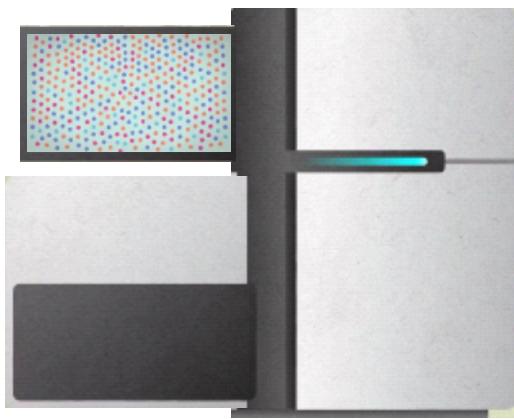


The Economist

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

Source: Illumina

Data is Key for Future Workloads

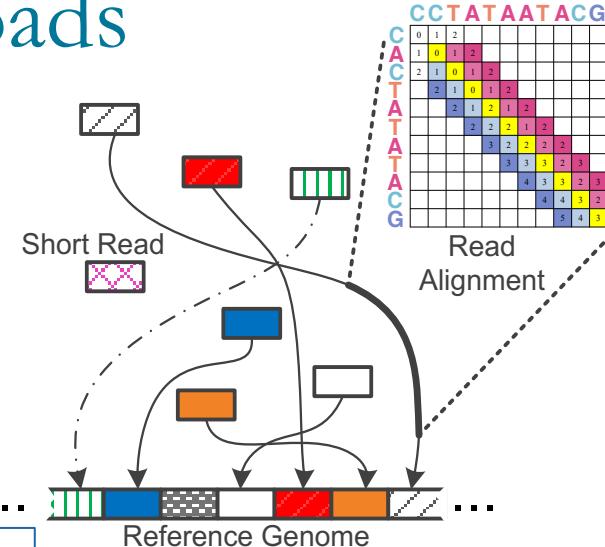


Billions of Short Reads

```
ATATATACGTACTAGTACGT  
TTTAGTACGTACGT  
ATACGTACTAGTACGT  
CG CCCCTACGT  
ACGTACTAGTACGT  
TTAGTACGTACGT  
TACGTACTAAAGTACGT  
TACGTACTAGTACGT  
TTTAAAAACGTA  
CGTACTAGTACGT  
GGGAGTACGTACGT
```

1 Sequencing

Genome Analysis



2 Read Mapping

Data → performance & energy bottleneck

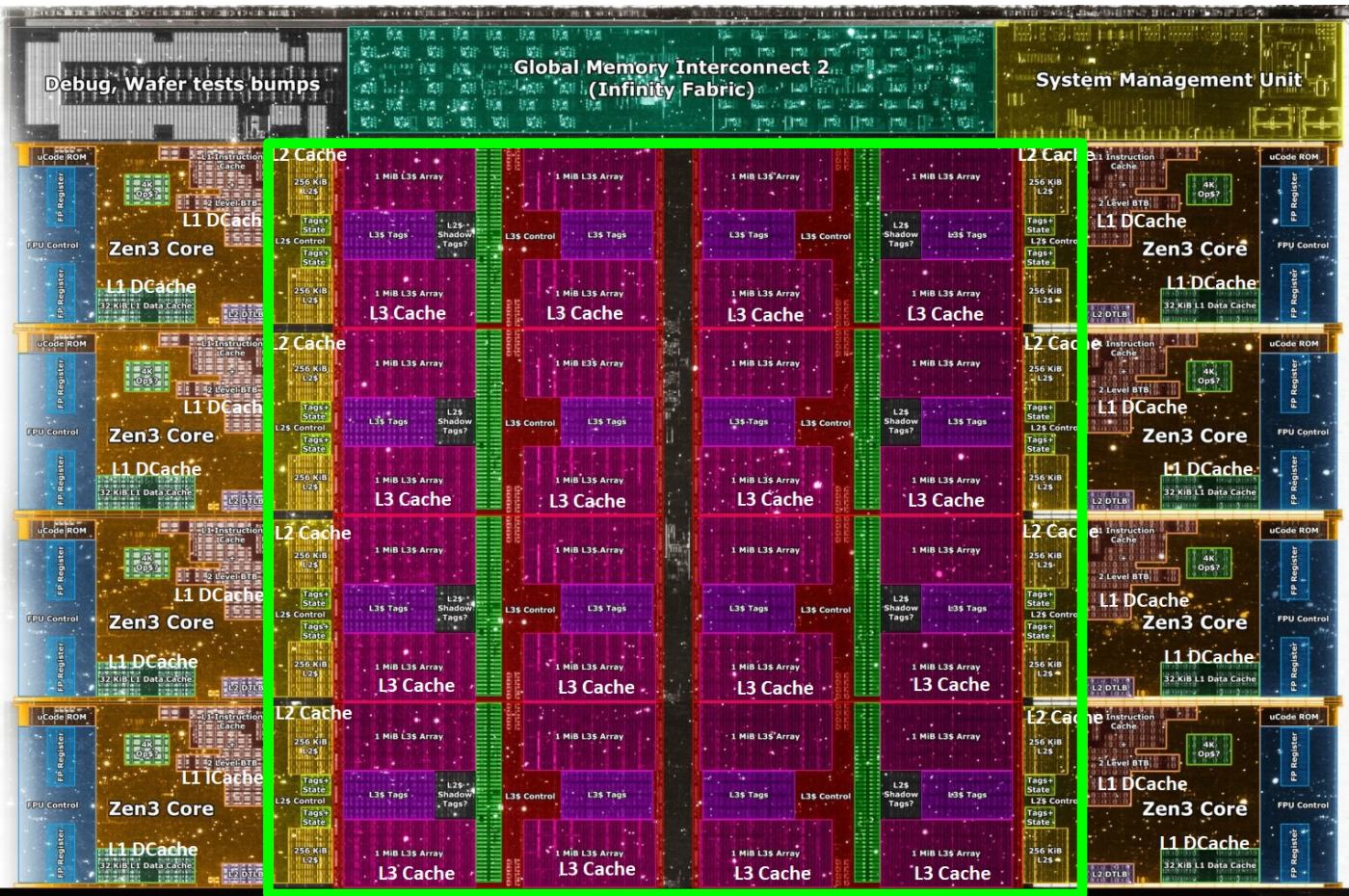
```
read4: CGCTTCCAT  
read5: CCATGACGC  
read6: TTCCATGAC
```



3 Variant Calling

Scientific Discovery 4

A Solution: Deeper and Larger Memory Hierarchies



Core Count:
8 cores/16 threads

L1 Caches:
32 KB per core

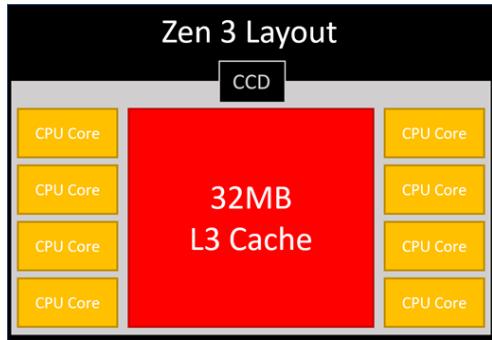
L2 Caches:
512 KB per core

L3 Cache:
32 MB shared

AMD Ryzen 5000, 2020

<https://wccftech.com/amd-ryzen-5000-zen-3-vermeer-undressed-high-res-die-shots-close-ups-pictured-detailed/>

AMD's 3D Last Level Cache (2021)

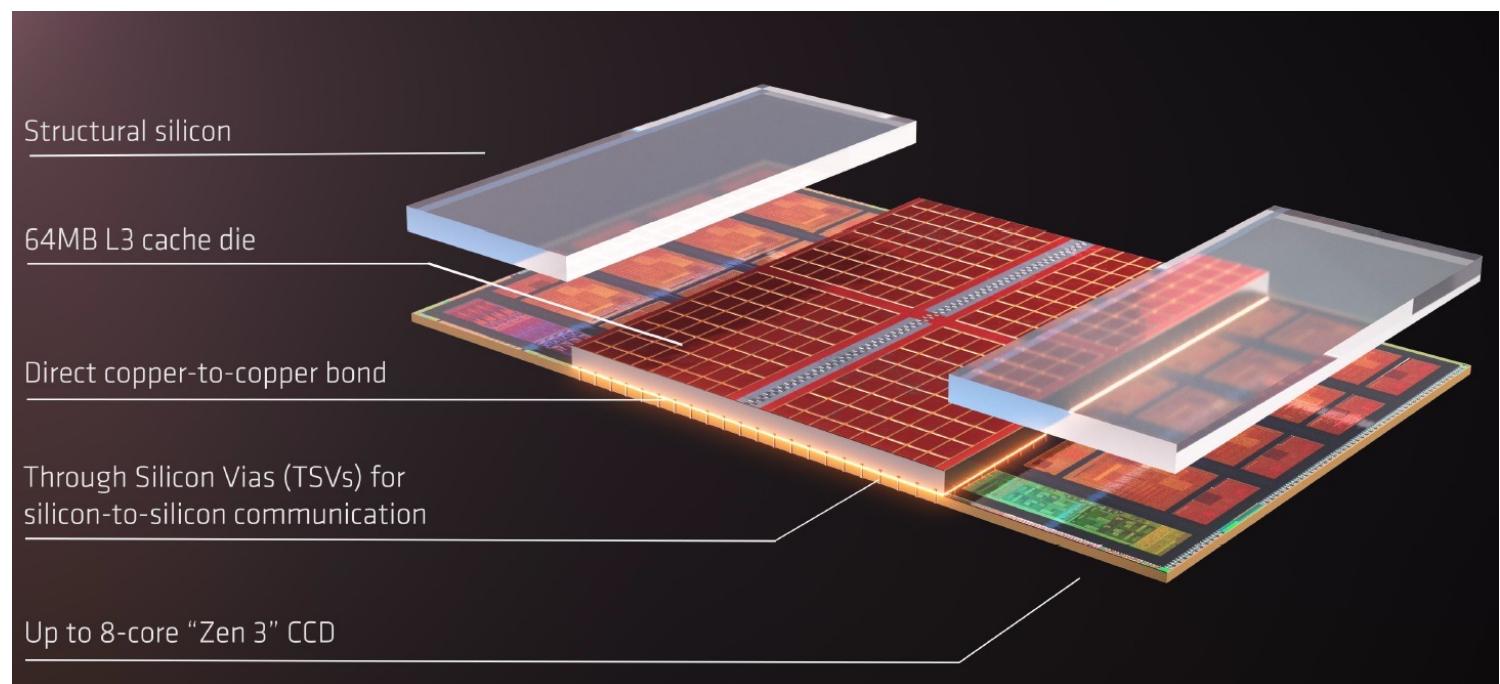


<https://community.microcenter.com/discussion/5134/comparing-zen-3-to-zen-2>

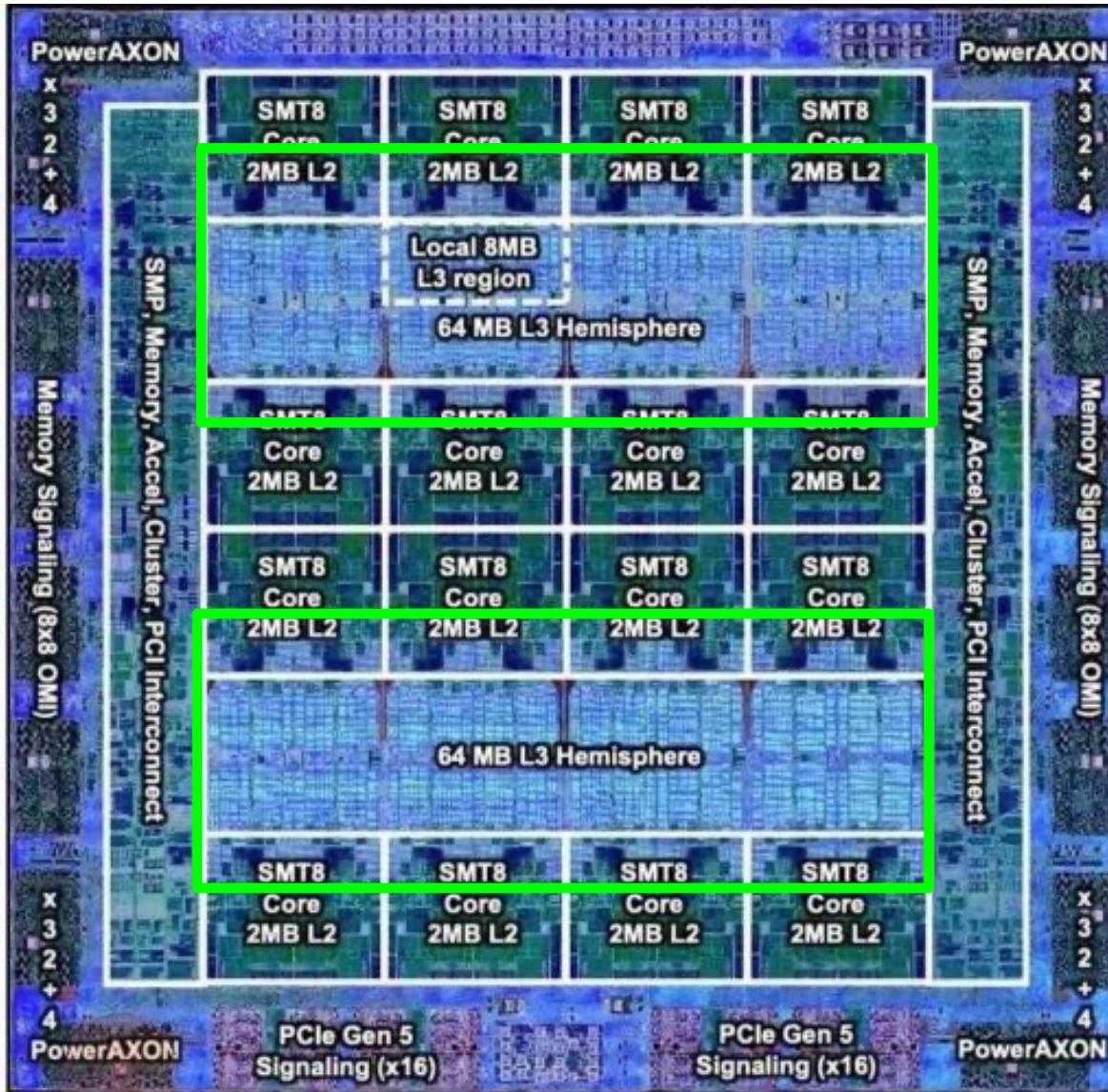
AMD increases the L3 size of their 8-core Zen 3 processors from 32 MB to 96 MB

Additional 64 MB L3 cache die stacked on top of the processor die

- Connected using Through Silicon Vias (TSVs)
- Total of 96 MB L3 cache



Deeper and Larger Memory Hierarchies



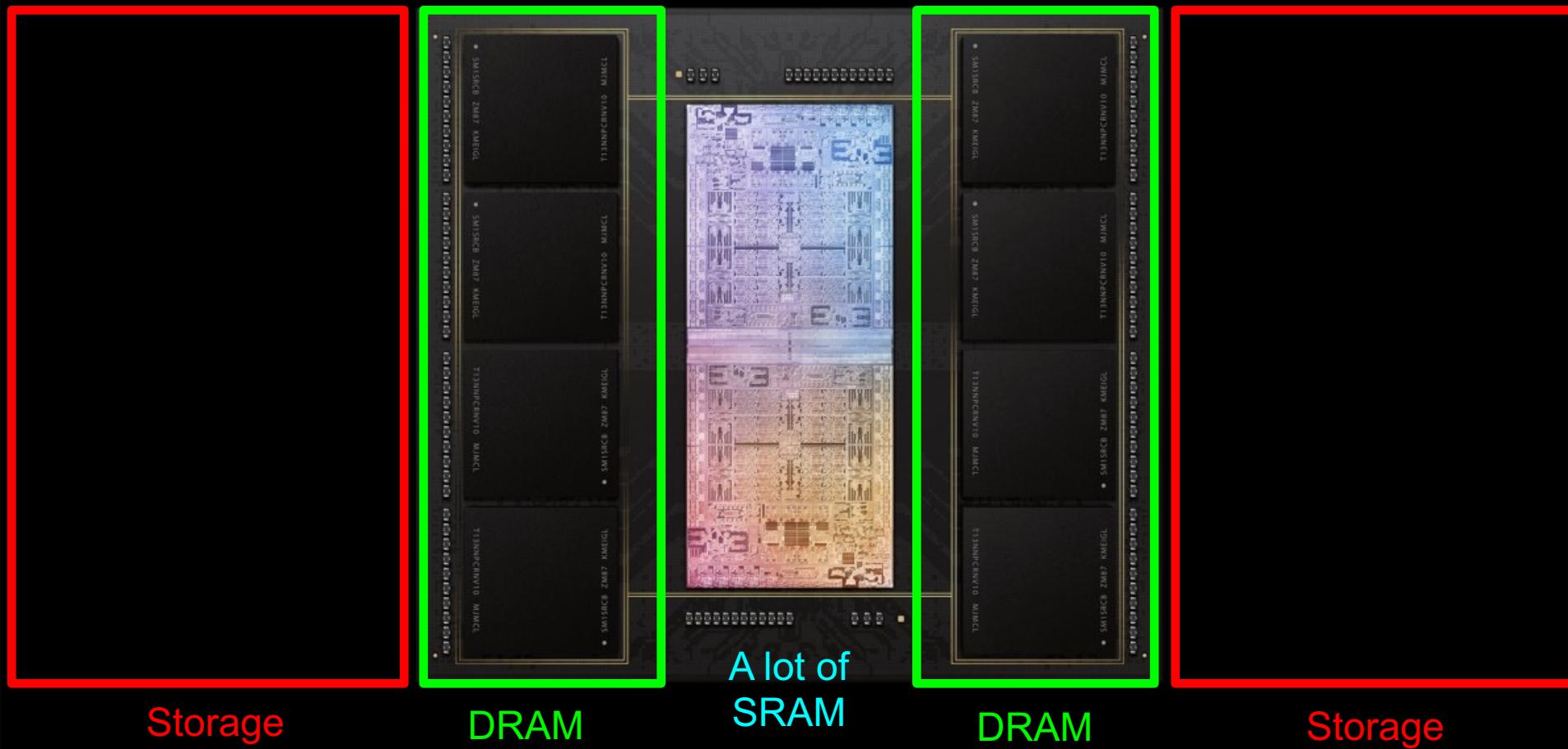
IBM POWER10,
2020

Cores:
15-16 cores,
8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared

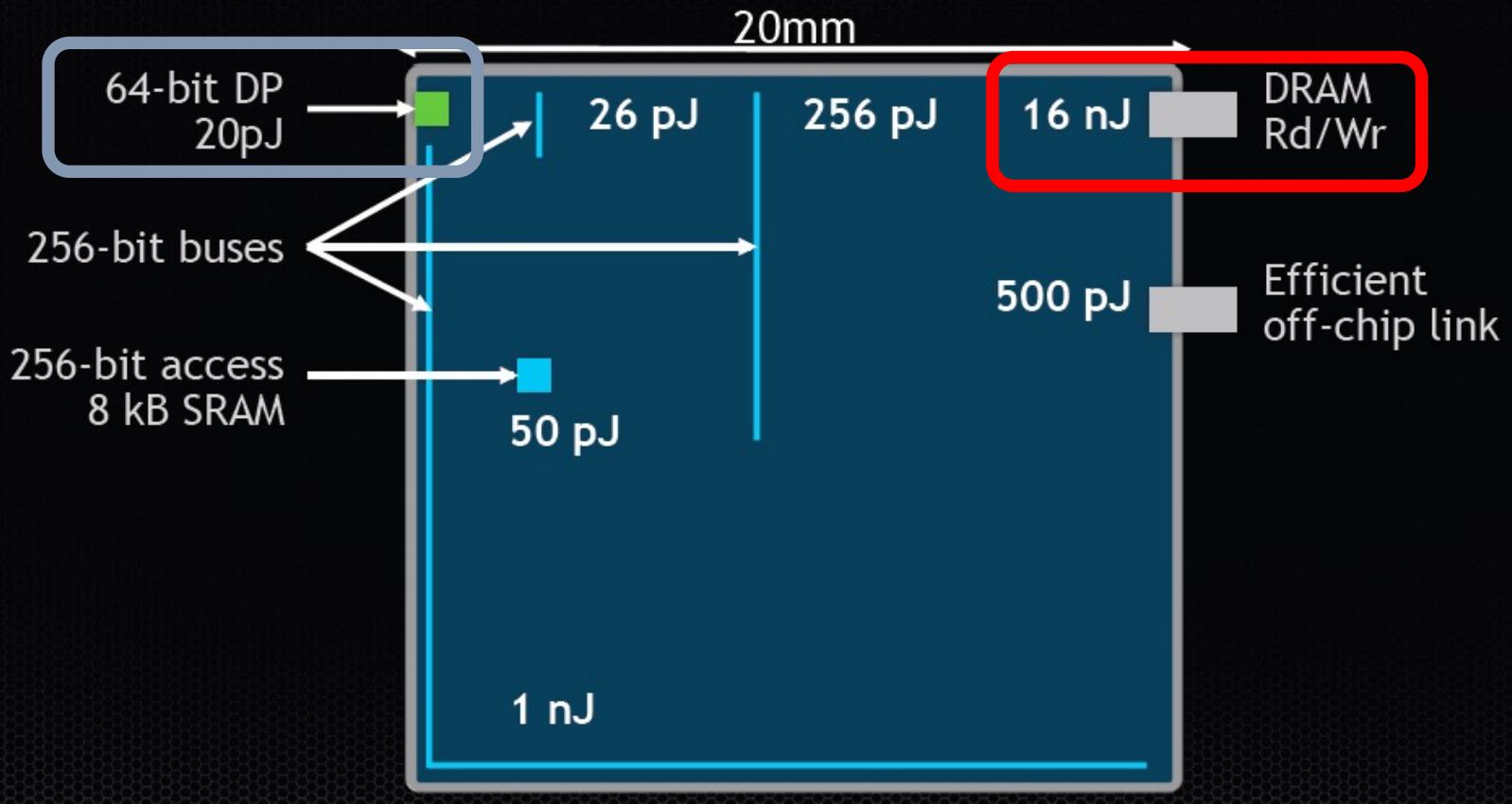
Deeper and Larger Memory Hierarchies



The Energy Perspective

Communication Dominates Arithmetic

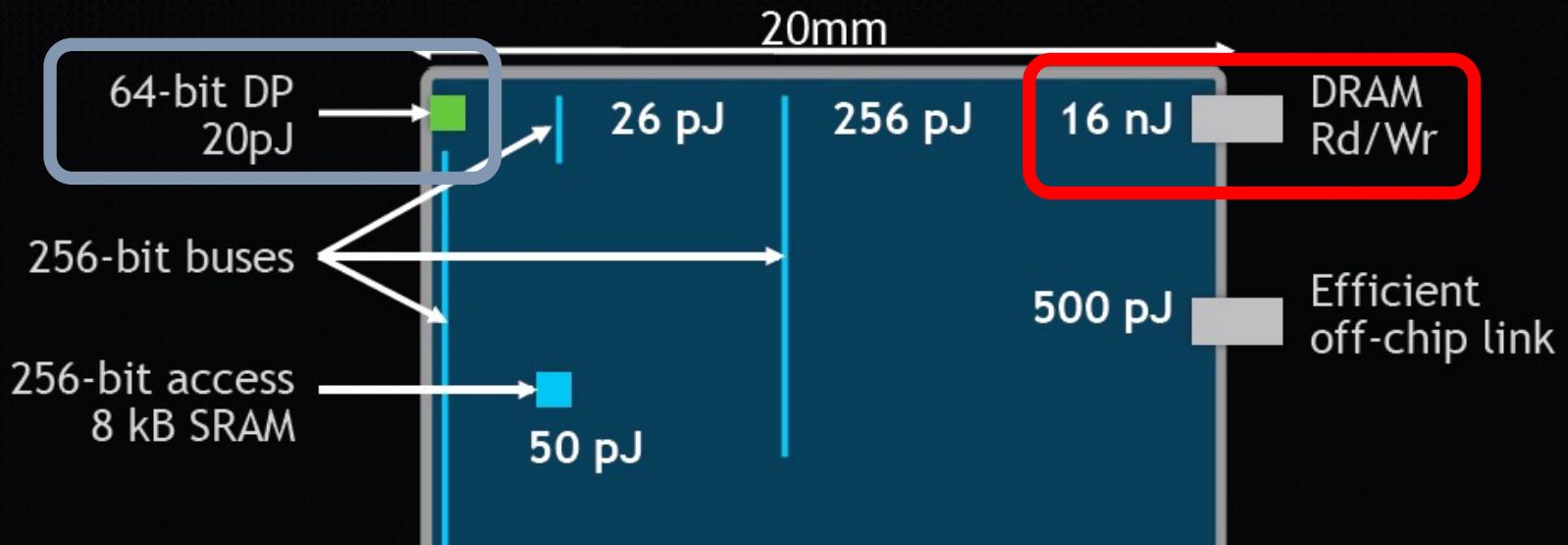
Dally, HiPEAC 2015



Data Movement vs Computation Energy

Communication Dominates Arithmetic

Dally, HiPEAC 2015

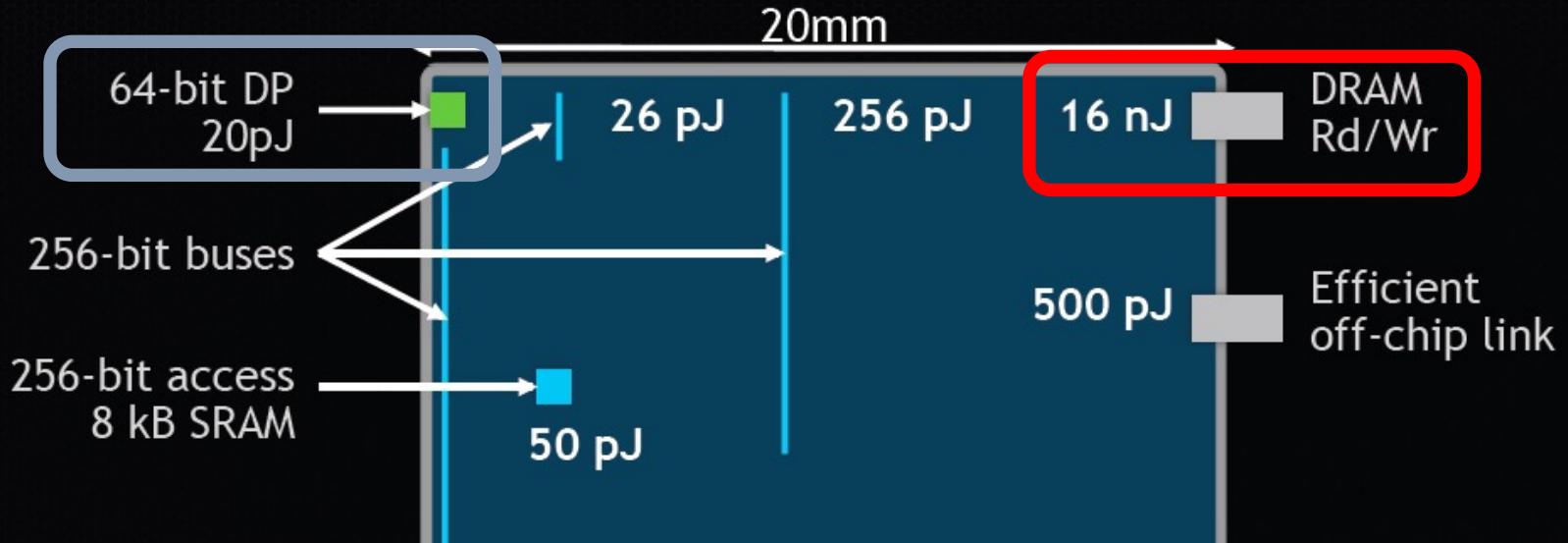


A memory access consumes \sim 100-1000X
the energy of a complex addition

We Do Not Want to Move Data!

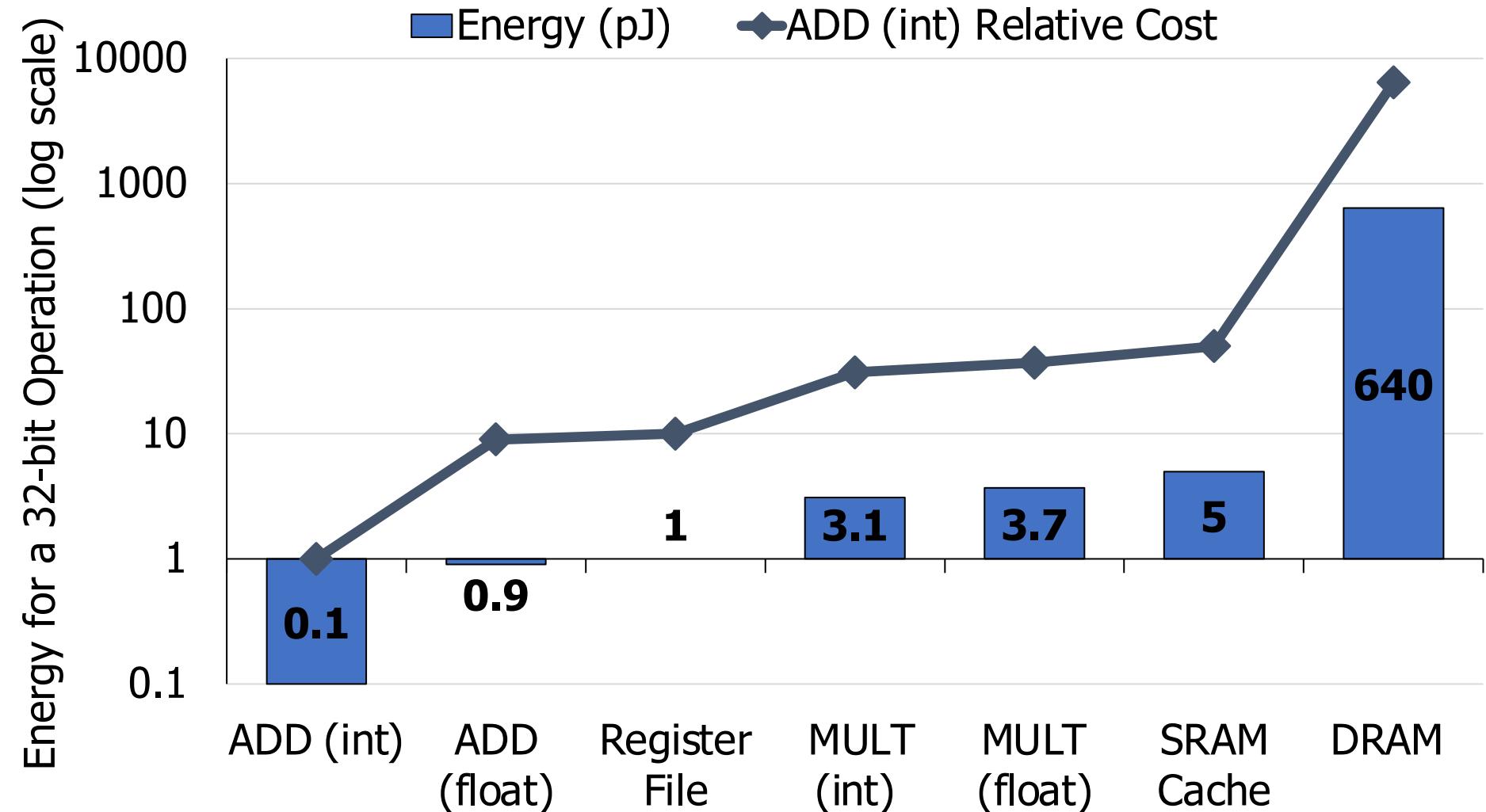
Communication Dominates Arithmetic

Dally, HiPEAC 2015



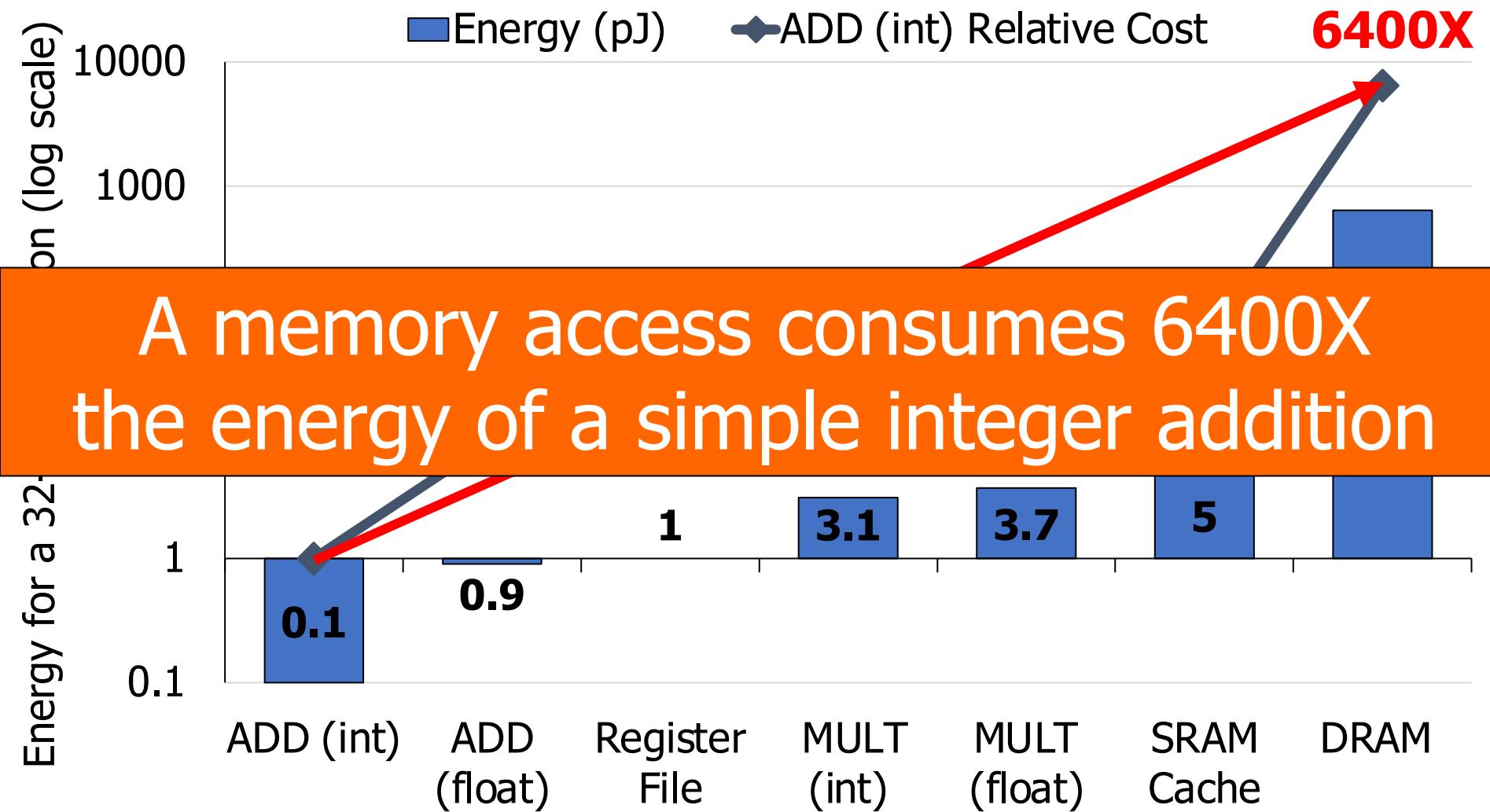
A memory access consumes \sim 100-1000X
the energy of a complex addition

Data Movement vs. Computation Energy



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

Data Movement vs. Computation Energy



Data Movement Overwhelms Modern Machines

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

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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹

Saugata Ghose¹

Youngsok Kim²

Rachata Ausavarungnirun¹

Eric Shiu³

Rahul Thakur³

Daehyun Kim^{4,3}

Aki Kuusela³

Allan Knies³

Parthasarathy Ranganathan³

Onur Mutlu^{5,1}

Data Movement Overwhelms Accelerators

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,

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

Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Virtual, September 2021.

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

[[Talk Video](#) (14 minutes)]

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

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

Amirali Boroumand^{†◊}

Geraldo F. Oliveira*

Saugata Ghose[‡]

Xiaoyu Ma[§]

Berkin Akin[§]

Eric Shiu[§]

Ravi Narayanaswami[§]

Onur Mutlu^{†*}

[†]*Carnegie Mellon Univ.*

[◊]*Stanford Univ.*

[‡]*Univ. of Illinois Urbana-Champaign*

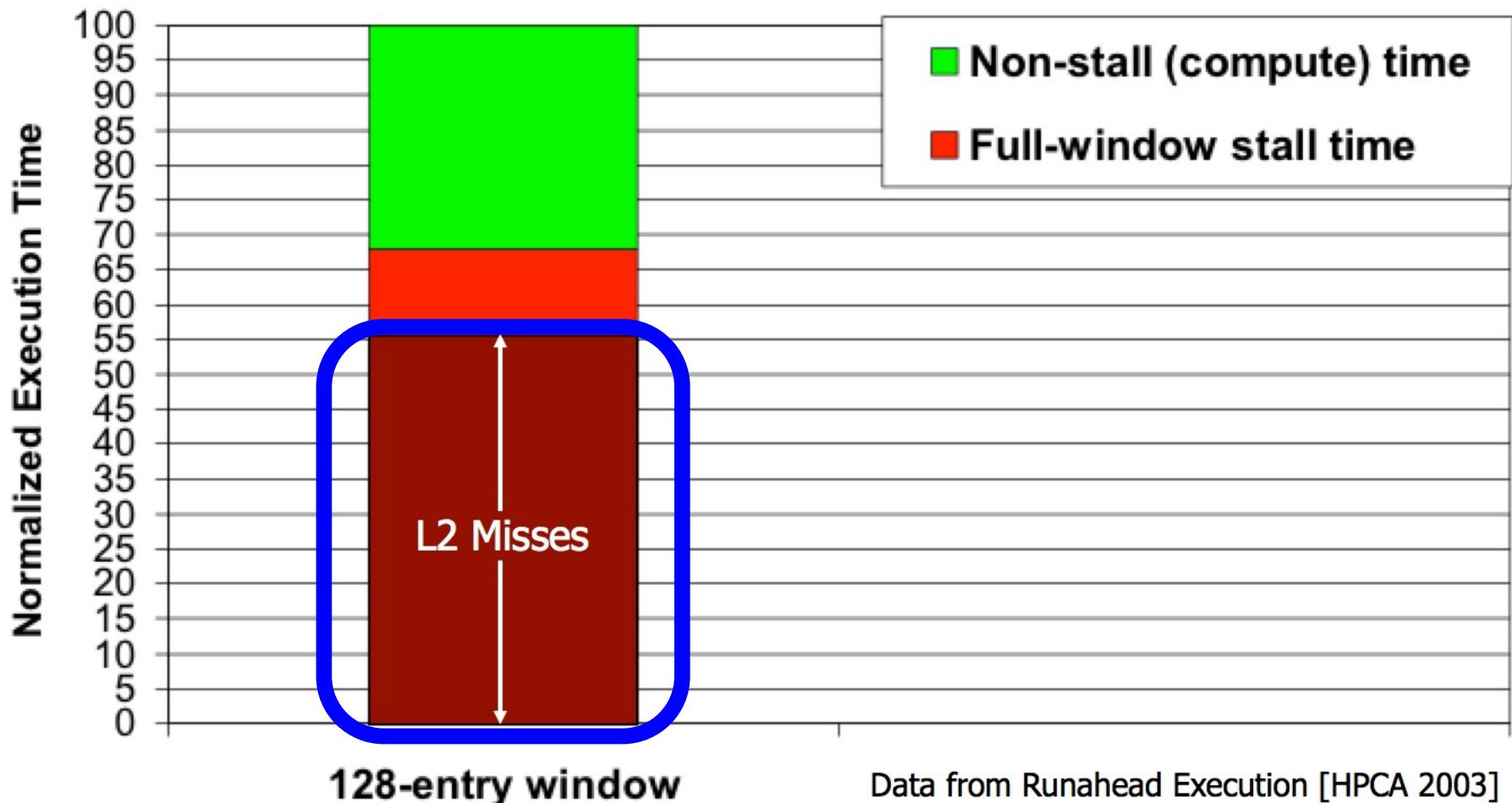
[§]*Google*

^{*}*ETH Zürich*

Yet ...

I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.

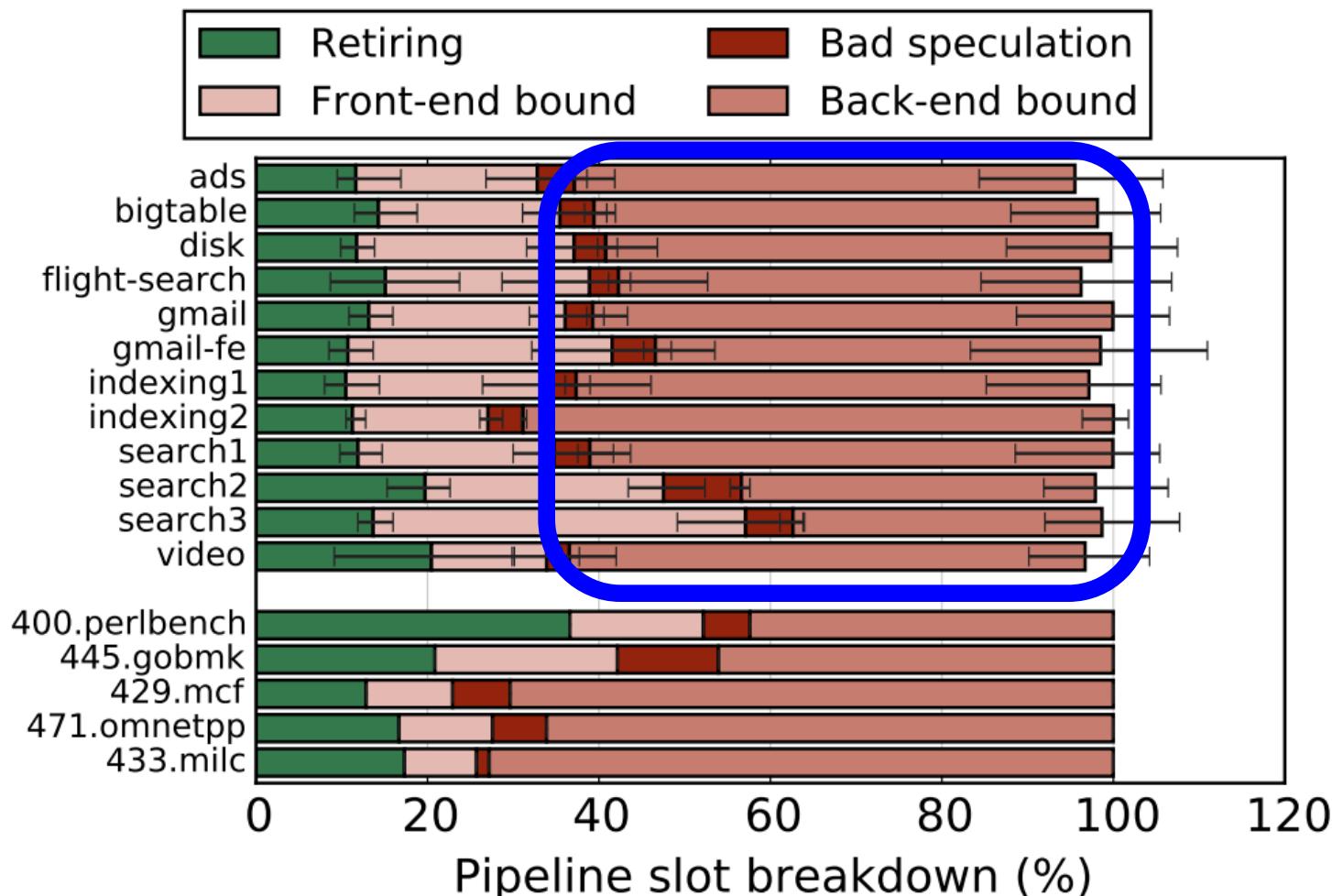
- “**It’s the Memory, Stupid!**” (Richard Sites, MPR, 1996)



Mutlu+, “Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors,” HPCA 2003.

Performance Perspective (Today)

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



Performance Perspective (Today)

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

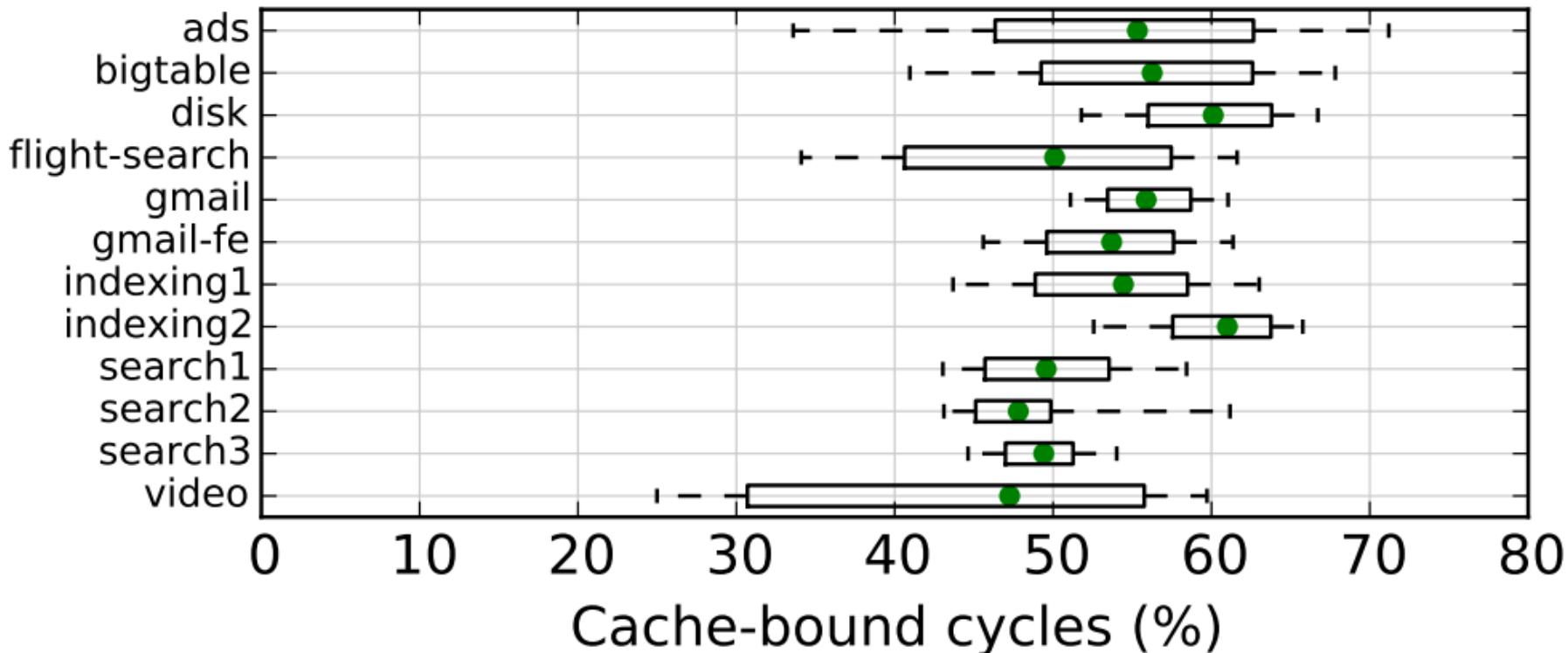


Figure 11: Half of cycles are spent stalled on caches.

Problem

Data access is the major performance and energy bottleneck

Our current
design principles cause
great energy waste
and
performance loss

Problem

Processing of data
is performed
far away from the data

Promising Solutions: Processing Near Memory (PnM)

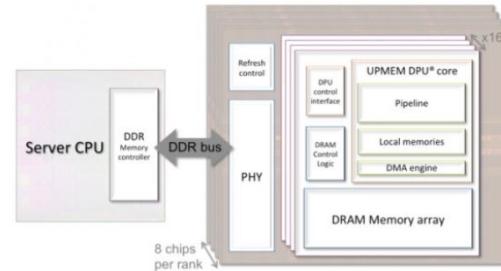
- UPMEM, founded in January 2015, announces the first real-world PIM architecture in 2016
- UPMEM's PIM-enabled DIMMs start getting commercialized in 2019
- In early 2021, Samsung announces FIMDRAM at ISSCC conference
- Samsung's LP-DDR5 and DIMM-based PIM announced a few months later
- In early 2022, SK Hynix announces AiM and Alibaba announces HB-PNM at ISSCC conference



Startup plans to embed processors in DRAM

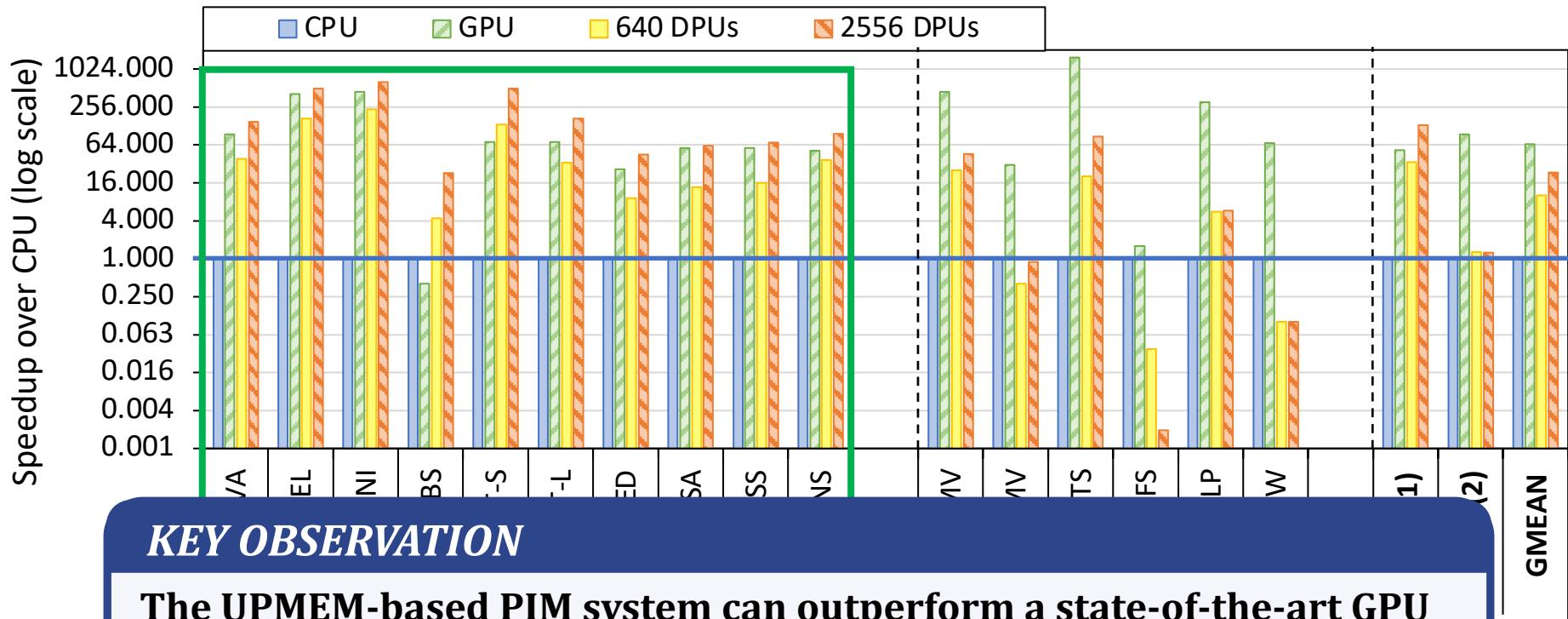
October 13, 2016 // By Peter Clarke

[Email](#) [print](#) [Share](#) [Share](#) [reddit](#)



Fabless chip company Upmem SAS (Grenoble, France), founded in January 2015, is developing a microprocessor for use in data-intensive applications in the datacenter that will sit embedded in DRAM to be close to the data.

Promising Solutions: Processing Near Memory (PnM)



KEY OBSERVATION

The UPMEM-based PIM system can outperform a state-of-the-art GPU on workloads with three key characteristics:

1. Streaming memory accesses
2. No or little inter-DPU synchronization
3. No or little use of integer multiplication, integer division, or floating point operations

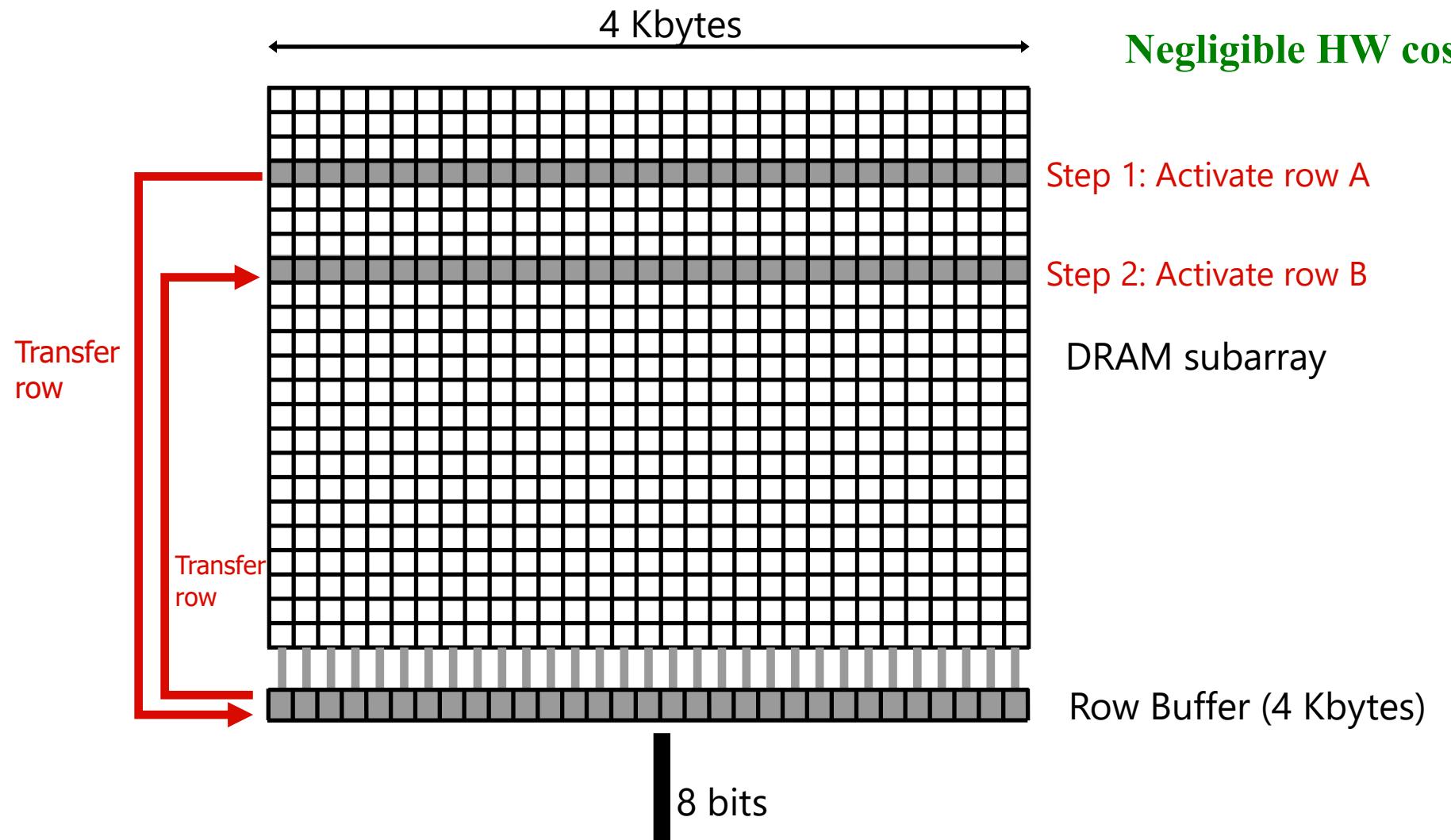
These three key characteristics make a workload potentially suitable to the UPMEM PIM architecture.

Promising Solutions: Processing Using Memory (PuM)

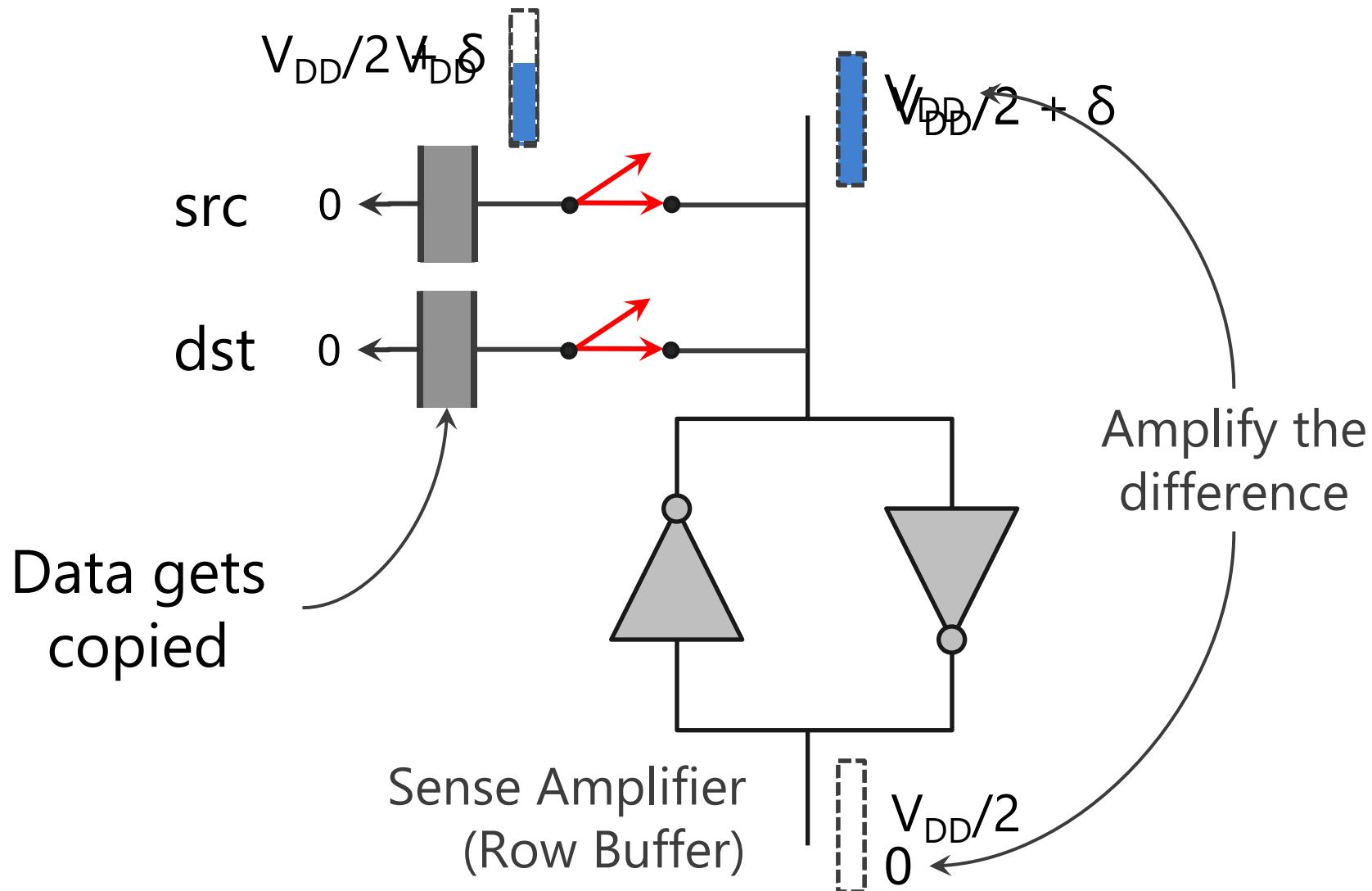
Implementing Data Movement

Idea: Two consecutive ACTivates

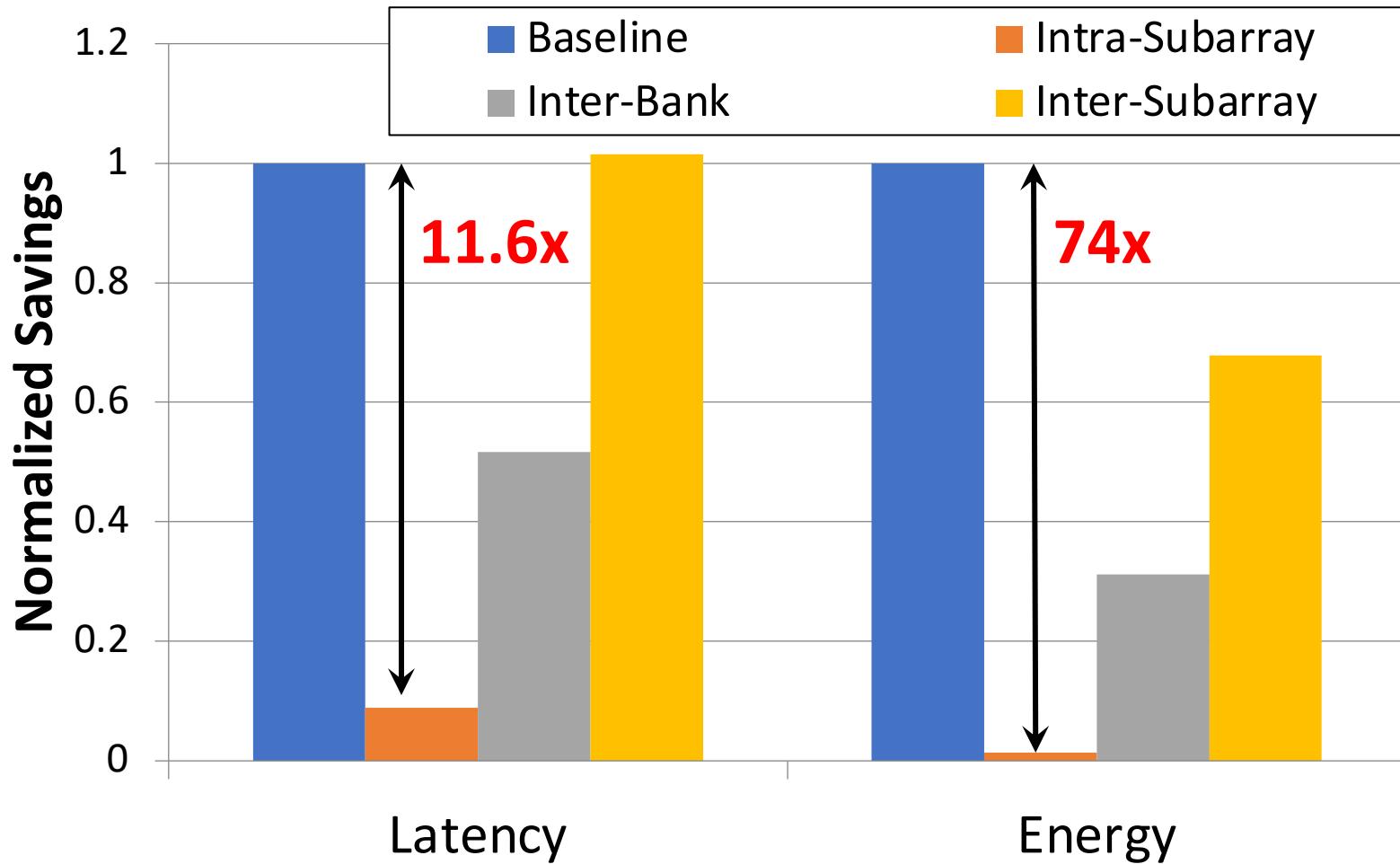
Negligible HW cost



Promising Solutions: Processing Using Memory (PuM) Implementing Data Movement

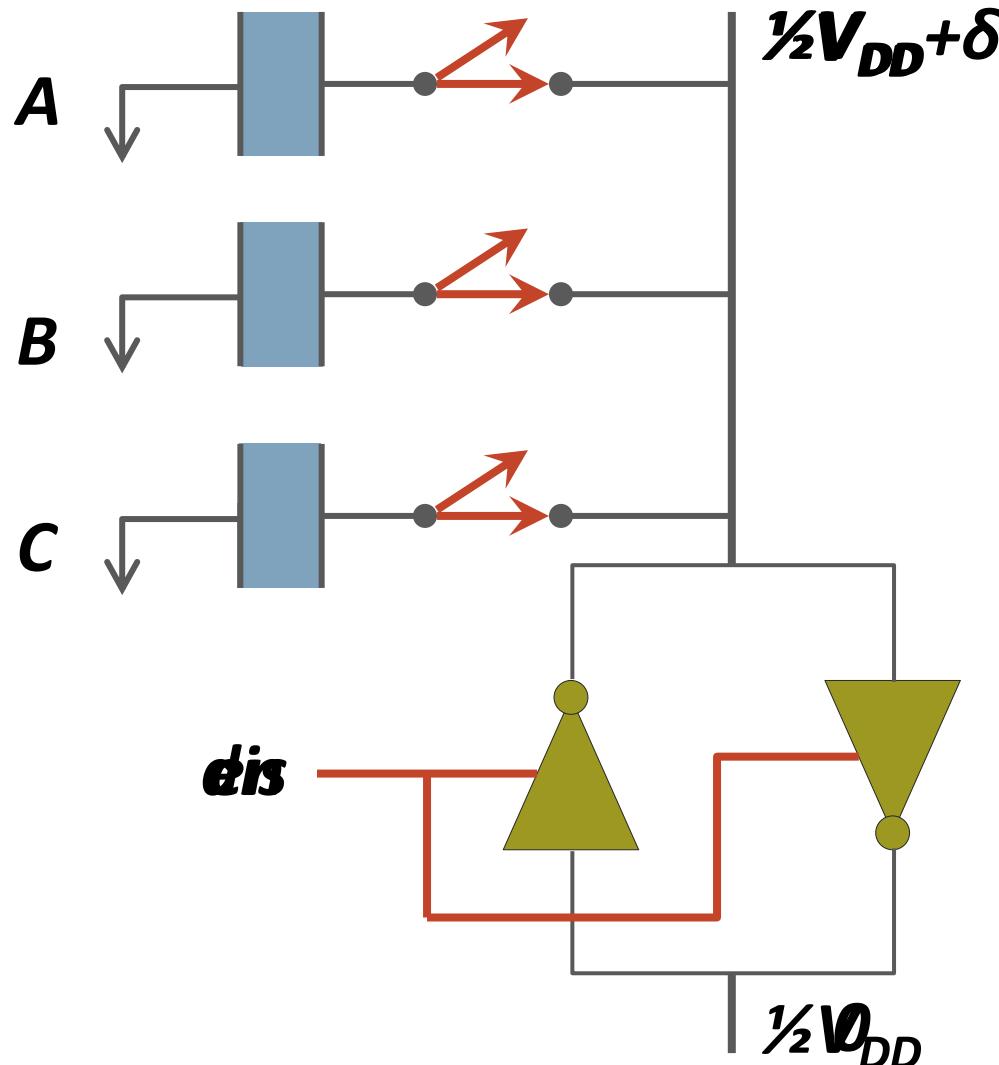


Promising Solutions: Processing Using Memory (PuM) Implementing Data Movement



Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

Promising Solutions: Processing Using Memory (PuM) Implementing Logic Operations



Final State
 $AB + BC + AC$

$C(A + B) +$
 $\sim C(AB)$

Promising Solutions: Processing Using Memory (PuM) Implementing Logic Operations

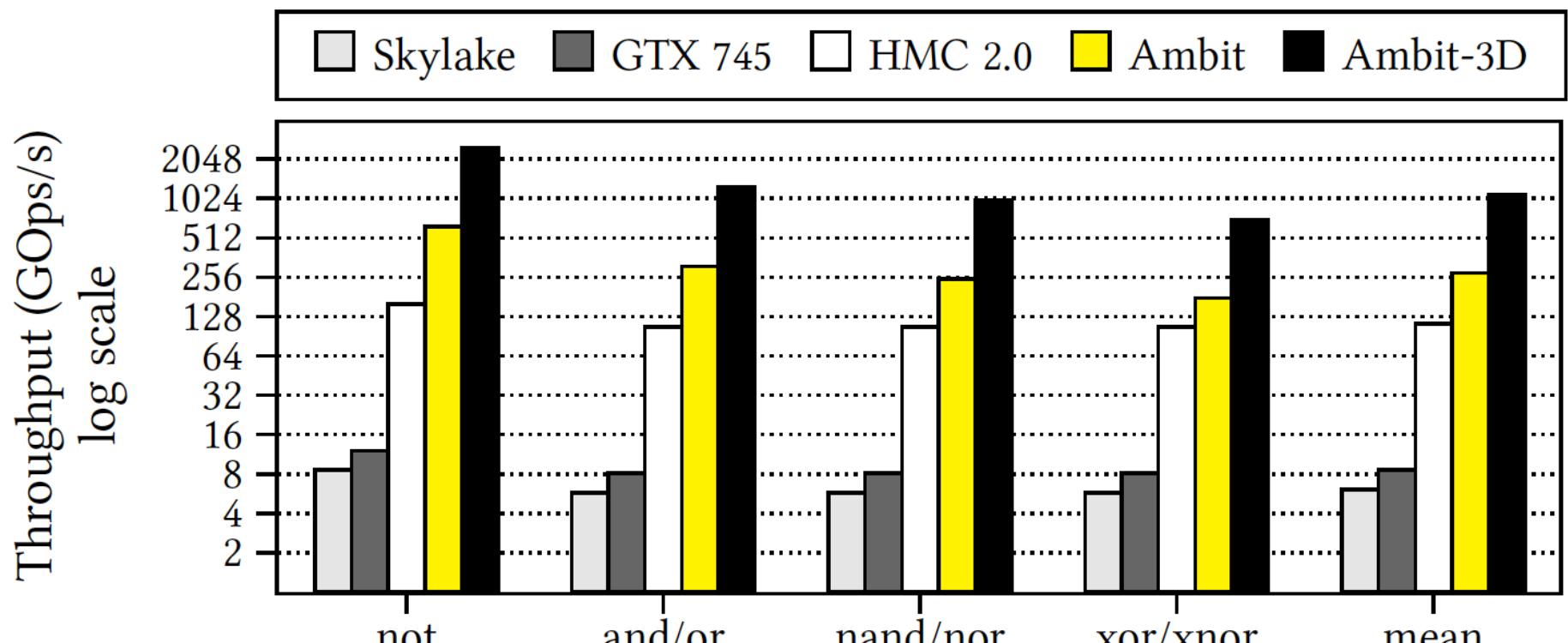


Figure 9: Throughput of bitwise operations on various systems.

Promising Solutions: Processing Using Memory (PuM) Implementing Logic Operations

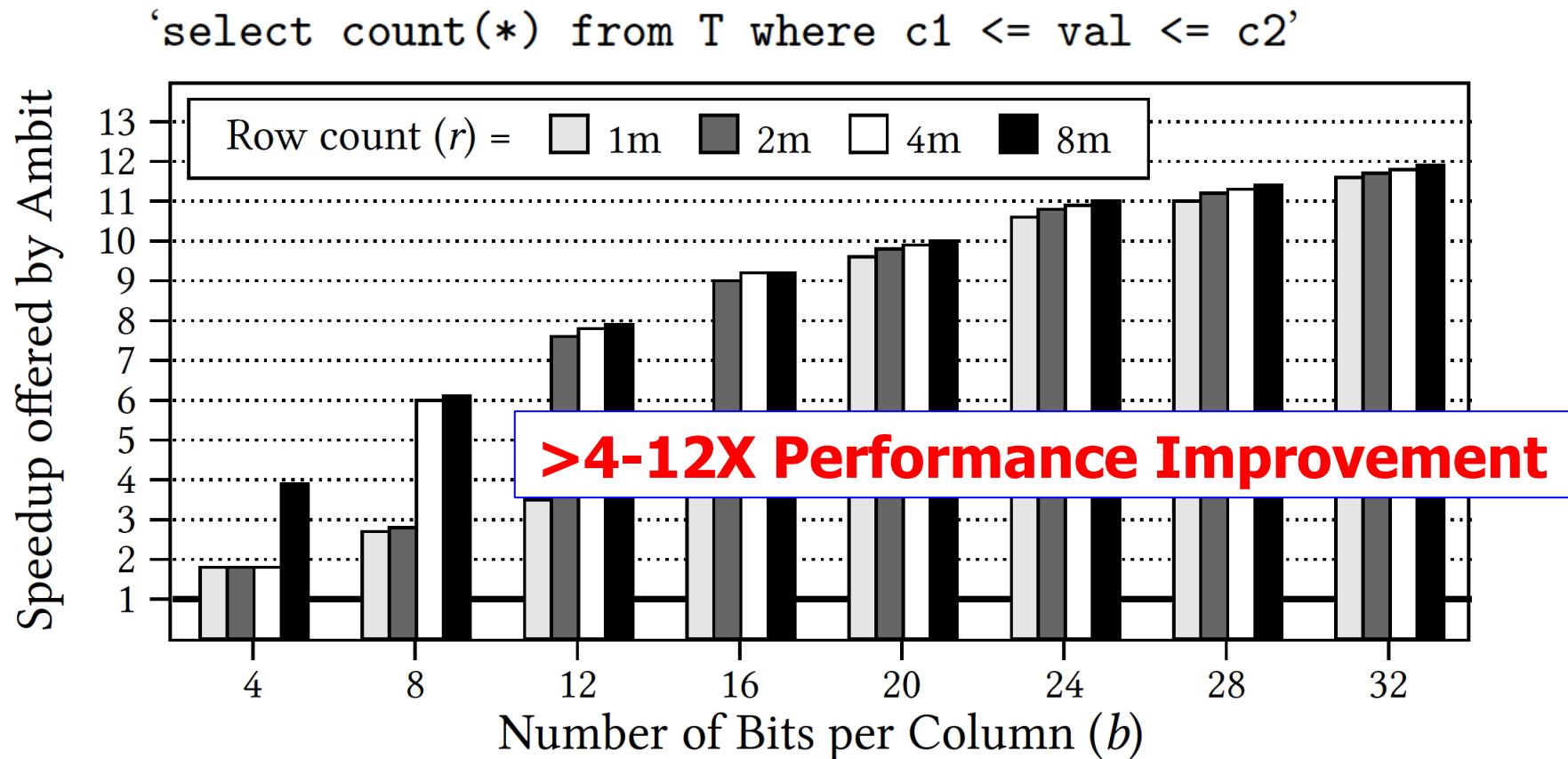


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Motivation

Processing near/using memory
is a promising computation paradigm

Distributing execution across
CPU, GPU, PnM, and PuM
should NOT hurt security and privacy

Research Scope (Going Forward)

- **Scope:** Confidentiality, integrity, and availability in large-scale & vastly-shared processing in memory and storage
- **Confidentiality:**
 - Physical isolation
 - Architecture support for encryption in memory/storage
- **Integrity:**
 - In-field patchable and dynamically tunable maintenance mechanisms
 - Integrity check and efficient roll-back mechanisms
- **Availability:**
 - Intelligent scheduling and fairness mechanisms
 - Adaptive maintenance supported with online profiling

Enabling Efficient and Scalable DRAM Read Disturbance Mitigation via New Experimental Insights into Modern DRAM Chips

Abdullah Giray Yağlıkçı

Research Summary and Future Directions

July 9, 2024

SAFARI

ETH zürich

Enabling Efficient and Scalable DRAM Read Disturbance Mitigation via New Experimental Insights into Modern DRAM Chips

Backup Slides

Abdullah Giray Yağlıkçı

Research Summary and Future Directions

July 9, 2024

How Large is 1000 Activations?

- Bitflips occur at ~1000 activations
- Mitigation mechanisms trigger **preventive actions** (e.g., preventive refresh) at ~500 activations
- Is 500 a **distinctive activation count**?
- Benign workloads activate **hundreds of rows** more than **500 times** in a refresh window

Memory intensive workloads

from SPEC'06, SPEC'17, TPC, YCSB, and MediaBench

Workload	MPKI	ACT-64+	ACT-128+	ACT-512+
429.mcf	68.27	2564	2564	2564
470.lbm	28.09	7089	6596	664
462.libquantum	25.95	1	0	0
549.fotonik3d	25.28	10065	88	0
459.GemsFDTD	24.93	10572	218	0
519.lbm	24.37	5824	5455	2482
434.zeusmp	22.24	11085	4825	292
510.parest	17.79	803	185	94
433.milc	17.22	321	92	0
437.leslie3d	15.82	4678	631	7
483.xalancbmk	13.67	4354	776	113
482.sphinx3	12.59	1385	762	304
505.mcf	11.35	1582	1384	732
471.omnetpp	10.72	1015	419	122
tpch2	9.09	875	307	88
520.omnetpp	9.00	1185	84	32
tpch17	7.43	1196	158	26
473.astar	5.18	5957	22	0
436.cactusADM	4.94	6151	2354	1134
jp2_encode	4.18	0	0	0

Benign workloads might not be so benign

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]

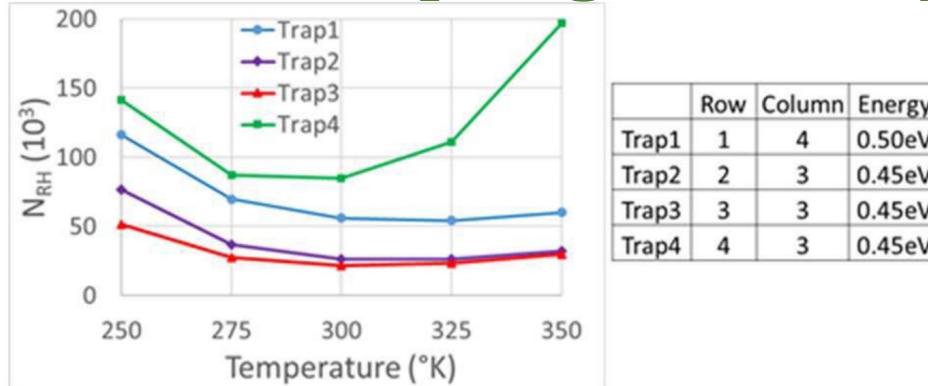


Fig. 6. Hammering threshold N_{RH} vs. temperature from 250 to 350°K for different traps. Location in row and column refers to matrix in **Fig. 2b**.

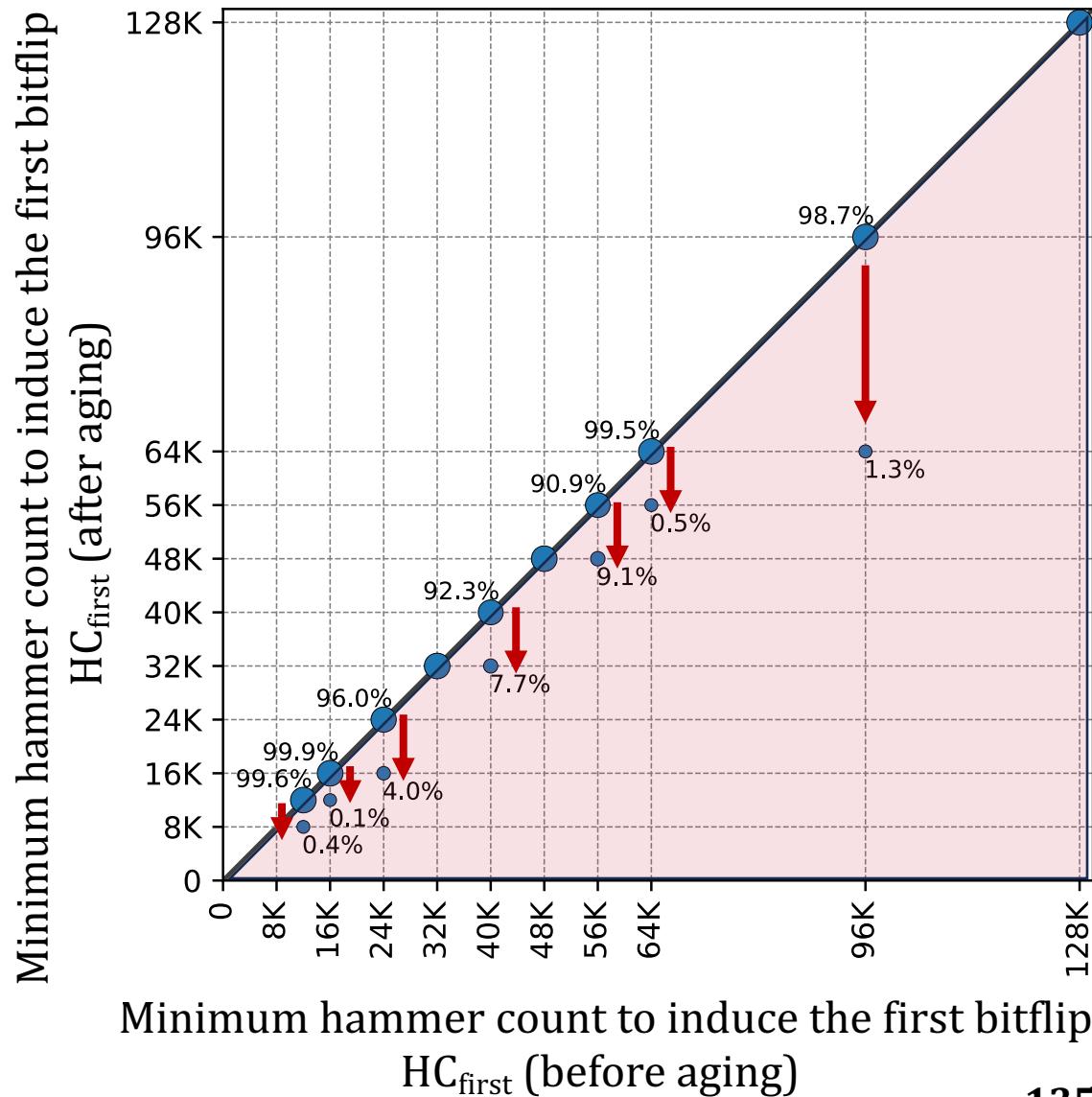
HC_{first} decreases as temperature increases, until a temperature inflection point where **HC_{first} starts to increase as temperature increases**

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

Deeper Understanding of Physics and Vulnerabilities

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

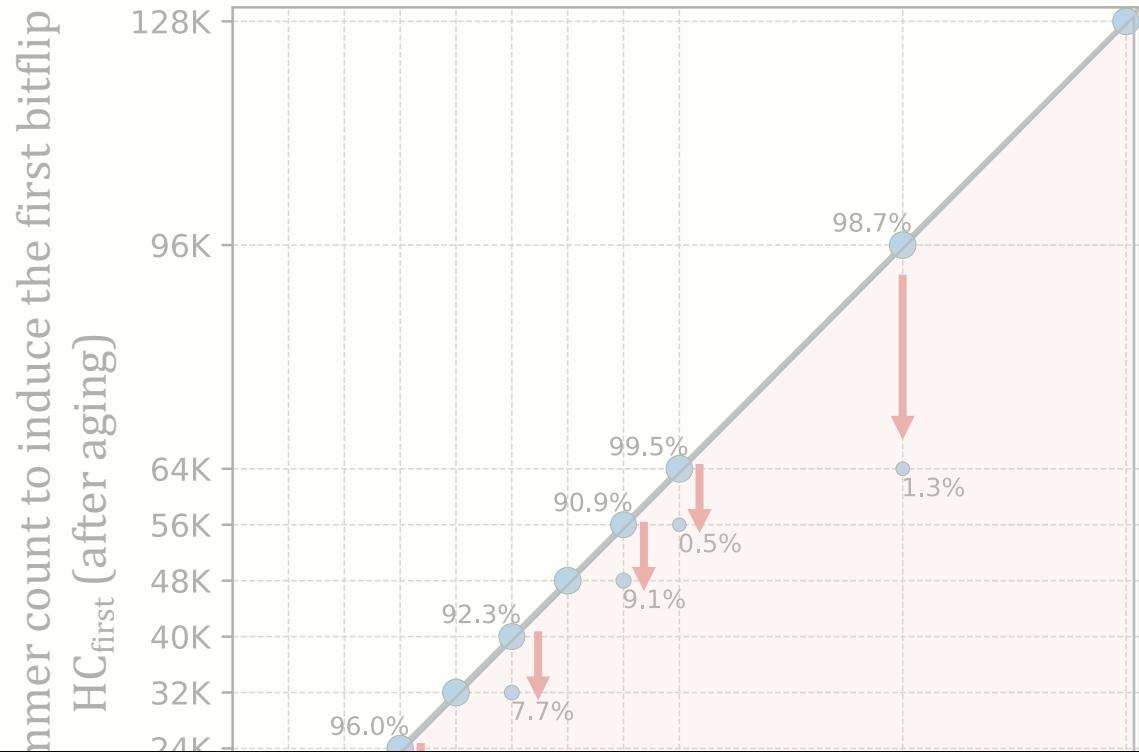
Aging can lead to read disturbance bitflips at **smaller** hammer counts



Deeper Understanding of Physics and Vulnerabilities

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

Aging can lead to read disturbance bitflips at smaller hammer counts



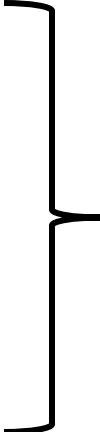
Future work:
rigorous aging characterization
and **online profiling of read disturbance vulnerability**

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

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
 - RowHammer
 - RowPress
 - Data retention time errors
 - Variable retention time
 - ...

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
- What is **the worst-case**?
 - Temperature
 - Data pattern
 - Memory access pattern
 - Spatial variation
 - Voltage

What is **the worst-case** considering all **these sensitivities**?

What is **the minimum hammer count** to induce a read disturbance bitflip?

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
- What is **the worst-case**?

How reliable are our DRAM chips?

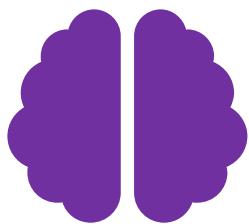
How reliable will our DRAM chips be tomorrow?

We **do not** know! This is an **open research problem**

Future Research for Better Memory Systems



Deeper Understanding of
Physics and Vulnerabilities



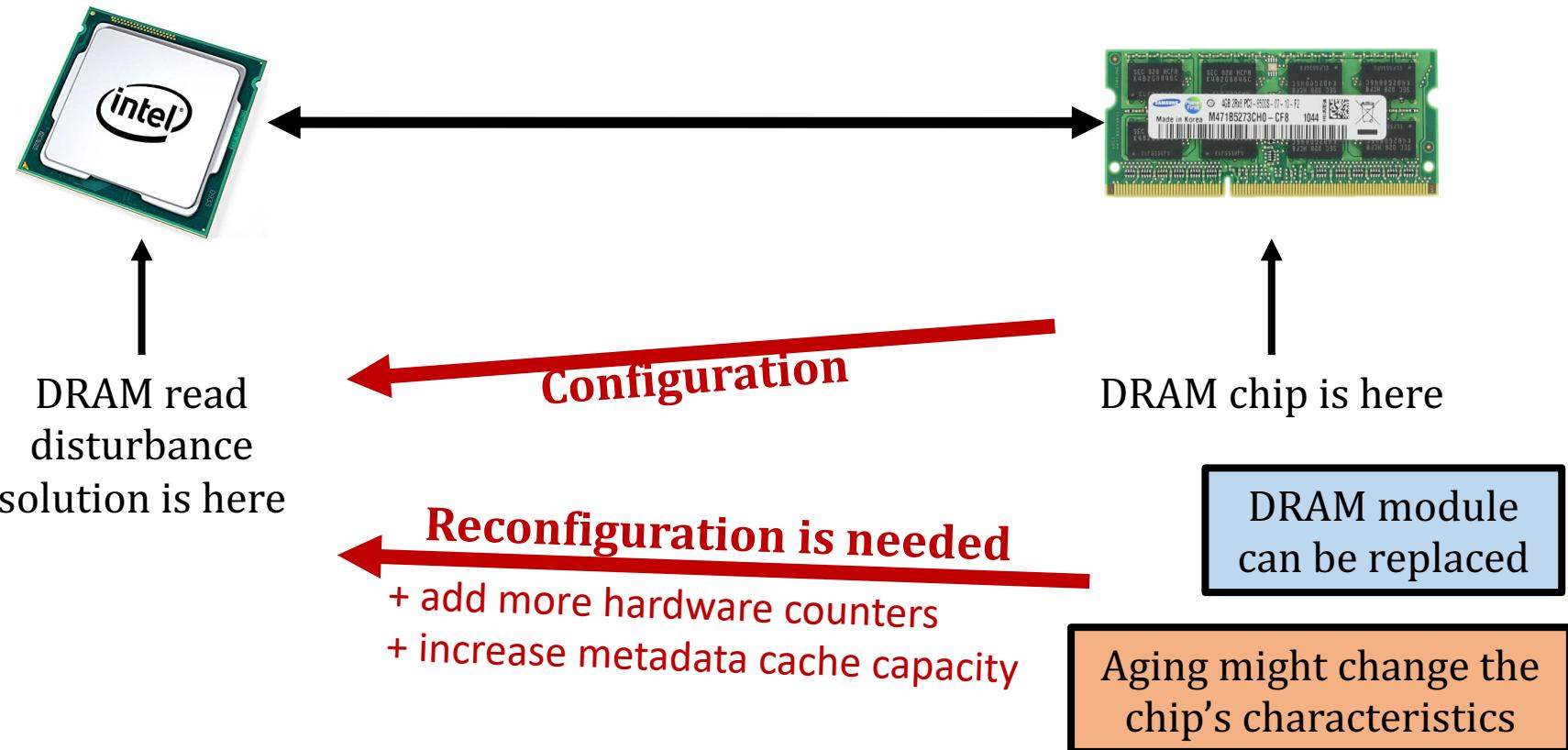
Flexible and Intelligent Memory
Chips, Interfaces, Controllers



Cross-Layer
Communication

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary

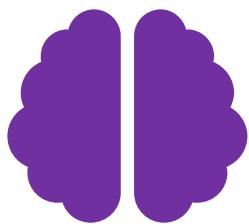


Deployed solutions should be patchable in field

Future Research for Better Memory Systems



Deeper Understanding of
Physics and Vulnerabilities

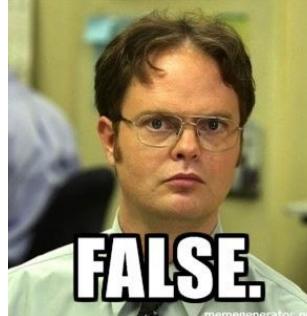


Flexible and Intelligent Memory
Chips, Interfaces, Controllers



Cross-Layer
Communication

Cross-Layer Communication

	Detection	Mitigation
Software	<ul style="list-style-type: none">• Memory allocations• Memory access patterns• Control flow patterns• Time / power measurements	 + +
uArch	<ul style="list-style-type: none">• Memory request scheduling• Speculative execution• Prefetching, branch prediction• Power management	+ +
Device	<ul style="list-style-type: none">• Bitflips occur• Memory isolation is broken	~

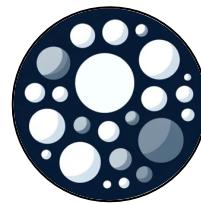
Cross-Layer Communication

	Detection	Mitigation
Software	<ul style="list-style-type: none">• Memory allocations• Memory access patterns• Control flow patterns• Time / power measurements	 + +
Memory / Power / CPU		
Cross-layer communication is crucial going forward		
Device	<ul style="list-style-type: none">• Bitflips occur• Memory isolation is broken	+ + ~

Awards and Honorable Mentions



BlockHammer: One of four finalists
Intel Hardware Security Academic Awards in 2022



Svärd: First place
ACM SRC (PACT) in 2023 (grand final is ongoing)



Thesis: One of five finalists (ongoing)
HOST Ph.D. Dissertation Competition in 2024

IChannels

Exploiting Current Management Mechanisms
to Create Covert Channels in Modern Processors

Jawad Haj-Yahya

*Jeremie S. Kim A. Giray Yağlıkçı Ivan Puddu Lois Orosa
Juan Gómez Luna Mohammed Alser Onur Mutlu*

SAFARI

ETH zürich

SAFARI

Executive Summary

Problem: Current management mechanisms throttle instruction execution and adjust voltage/frequency to accommodate power-hungry instructions (PHIs). These mechanisms may compromise a system's confidentiality guarantees

Goal:

1. Understand the throttling side-effects of current management mechanisms
2. Build high-capacity covert channels between otherwise isolated execution contexts
3. Practically and effectively mitigate each covert channel

Characterization: Variable execution times and frequency changes due to running PHIs
We observe five different levels of throttling in real Intel systems

IChannels: New covert channels that exploit side-effects of current management mechanisms

- On the same hardware thread
- Across co-located Simultaneous Multi-Threading (SMT) threads
- Across different physical cores

Evaluation: On three generations of Intel processors, IChannels provides a channel capacity

- 2× that of PHIs' variable latency-based covert channels
- 24× that of power management-based covert channels

ABACuS: All-Bank Activation Counters

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

To appear in Proceedings of the 33rd USENIX Security Symposium (USENIX Security), Philadelphia, PA, USA, August 2024.

[[arXiv version](#)]

[[ABACuS Source Code](#)]

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

Ataberk Olgun Yahya Can Tugrul Nisa Bostanci Ismail Emir Yuksel

Haocong Luo Steve Rhyner Abdullah Giray Yaglikci Geraldo F. Oliveira Onur Mutlu

ETH Zurich

ABACuS: All-Bank Activation Counters

Goal: Prevent RowHammer bitflips at **low performance, energy, and area cost**

Key Observation: Workloads tend to access **the same row** in all DRAM banks at around the **same time**

Key Idea: Use **one hardware counter** to keep track of activation counts of the **same row** across all banks

- Make high-performance, area-hungry counter-based mechanisms **practical**

Key Results:

Faster than the **lowest-area-cost** counter-based defense mechanism

Smaller than the **lowest-performance-overhead** counter-based defense mechanism

0.59% avg. performance overhead (single-core) at a **RowHammer threshold (1K)**

- Only 9.79 KiB **on-chip** storage per DRAM rank (0.02% of a Xeon processor)

1.52% avg. performance overhead (single-core) at an **ultra-low threshold (125)**

- 75.70 KiB **on-chip** storage per DRAM rank (0.11% of the Xeon processor)

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

CoMeT: Count-Min-Sketch-based Row Tracking

- Nisa Bostancı, Ismail Emir Yuksel, Ataberk Olgun, Konstantinos Kanellopoulos, Yahya Can Tugrul, A. Giray Yaglikci, Mohammad Sadrosadati, Onur Mutlu

"CoMeT: Count-Min-Sketch-based Row Tracking to Mitigate RowHammer at Low Cost,"

in Proceedings the 30th International Symposium on High-Performance Computer Architecture (HPCA), Edinburgh, March 2024.

[\[arXiv version\]](#)

[\[CoMeT Source Code\]](#)



CoMeT: Count-Min-Sketch-based Row Tracking to Mitigate RowHammer at Low Cost

F. Nisa Bostancı
Yahya Can Tuğrul

İsmail Emir Yüksel
A. Giray Yağlıkçı

Ataberk Olgun
Mohammad Sadrosadati

Konstantinos Kanellopoulos
Onur Mutlu

ETH Zürich

Executive Summary

Goal: Prevent RowHammer bitflips **with low area, performance, and energy overheads** in highly RowHammer-vulnerable DRAM-based systems

Key Idea: Use **low-cost** and **scalable hash-based counters** to accurately track DRAM rows

CoMeT:

- tracks most DRAM rows with **scalable hash-based counters** by employing **the Count-Min-Sketch technique** to achieve a low area cost
- tracks only **a small set of DRAM rows that are activated many times** with **highly accurate per-DRAM-row activation counters** to reduce performance penalties

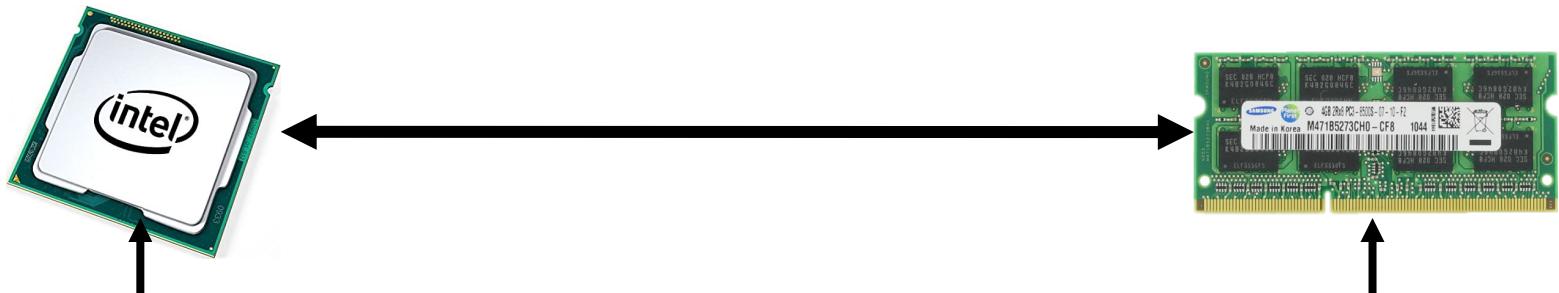
Evaluation: CoMeT achieves a good trade-off between area, performance and energy costs

- incurs significantly less area overhead (**74.2×**) compared to the state-of-the-art technique
- outperforms the state-of-the-art technique (by up to **39.1%**)

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

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary
- Interfaces should be **more flexible**



Memory controller
decides what should
be done when

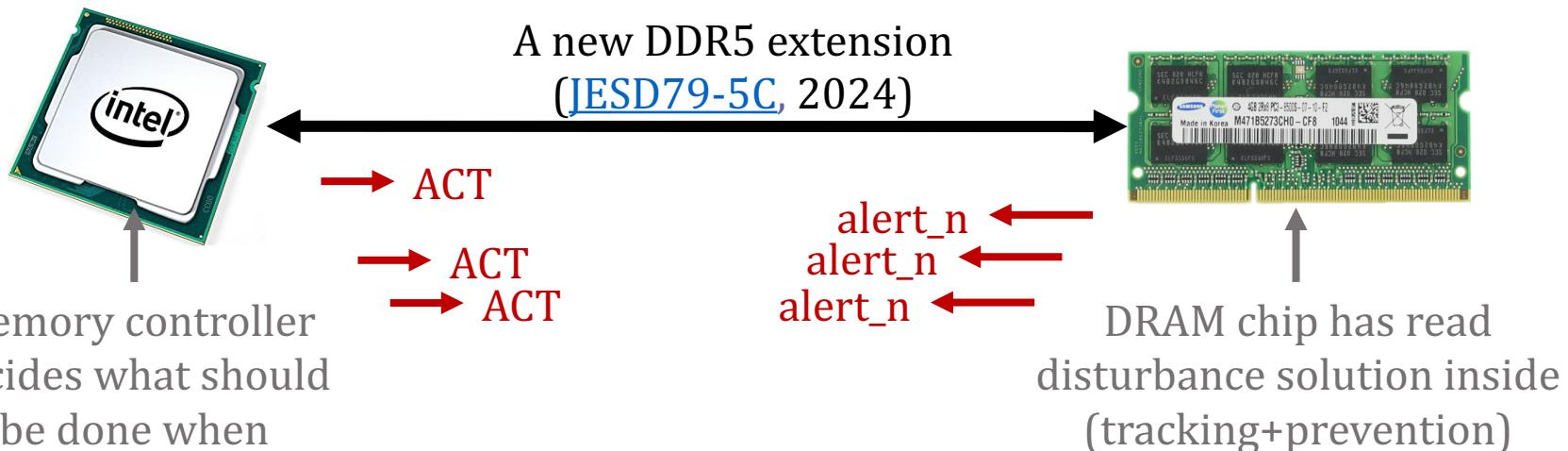
DRAM chip has read
disturbance solution inside
(tracking+prevention)

- The memory controller should provide the DRAM chip with **necessary time window** to perform **preventive actions (e.g., refreshing rows)**
- The memory controller **does not have** the tracking information
- Communicating is **not straightforward** due to strict communication protocol

A more flexible interface is necessary

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary
- Interfaces should be **more flexible**



- **Two Key Issues:**
- A naïve implementation can lead to **very high command bus occupancy**
 - Very high bandwidth consumption
 - Poor parallelism
- DRAM **self-manages preventive refreshes**. Why not **other maintenance routines?**

Flexible and Intelligent Chips, Interfaces, Controllers

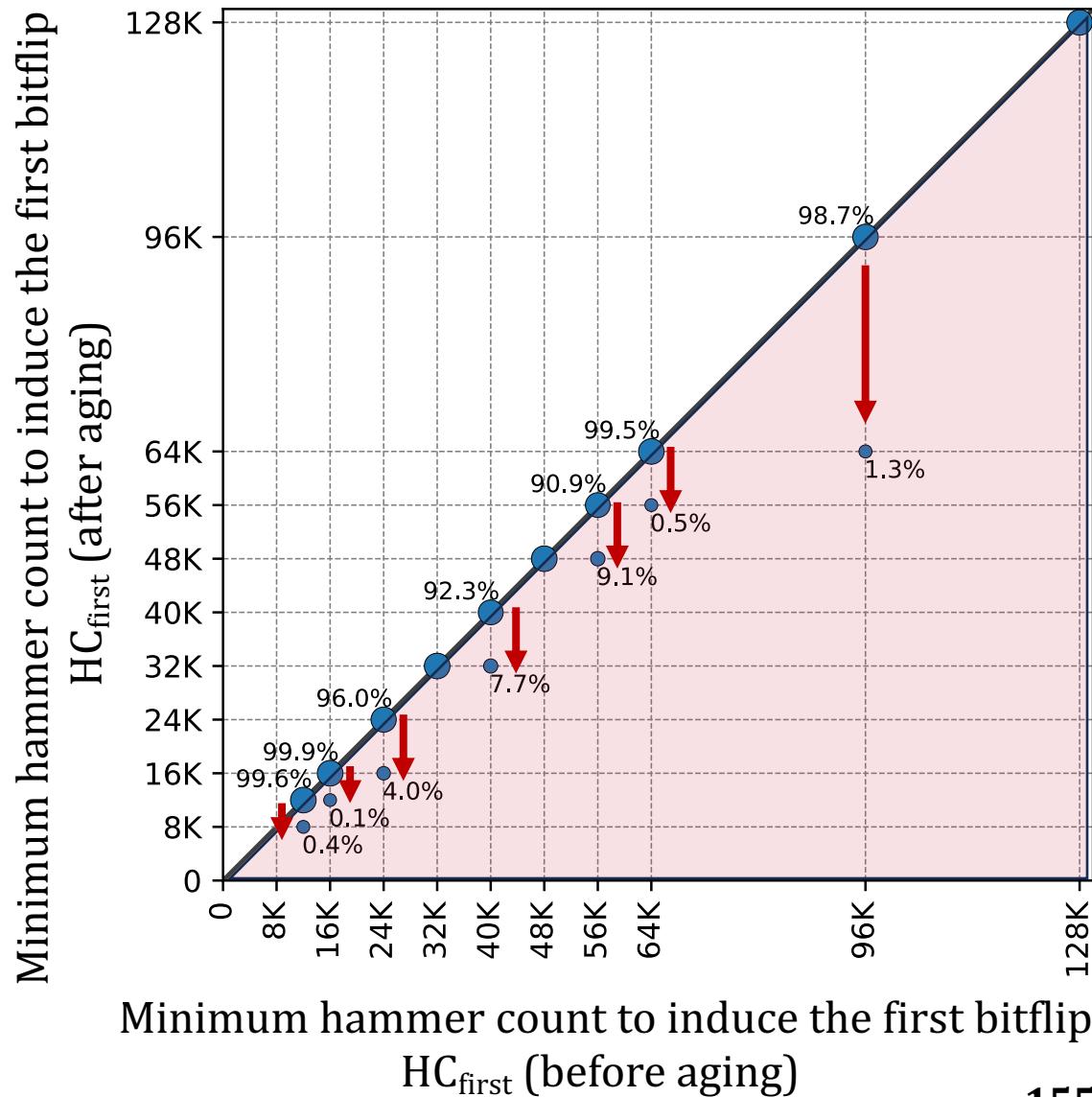
- **In-field patching** is necessary
- Interfaces should be **more flexible**
- Memory controllers should be **more intelligent** in detecting **malicious activity**
- DRAM chips become **more and more vulnerable** to RowHammer **and RowPress**
- **Key Insight:** Activation counts of **benign applications** get close to **unsafe levels**
- **Problem:** DRAM read disturbance solutions are getting **prohibitively expensive**
- **Research Question:** How to identify malicious threads/processes/users?
- More **intelligent detection** mechanisms are needed
- The **memory controller** observes all memory accesses

More intelligent memory controllers can help

Deeper Understanding of Physics and Vulnerabilities

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

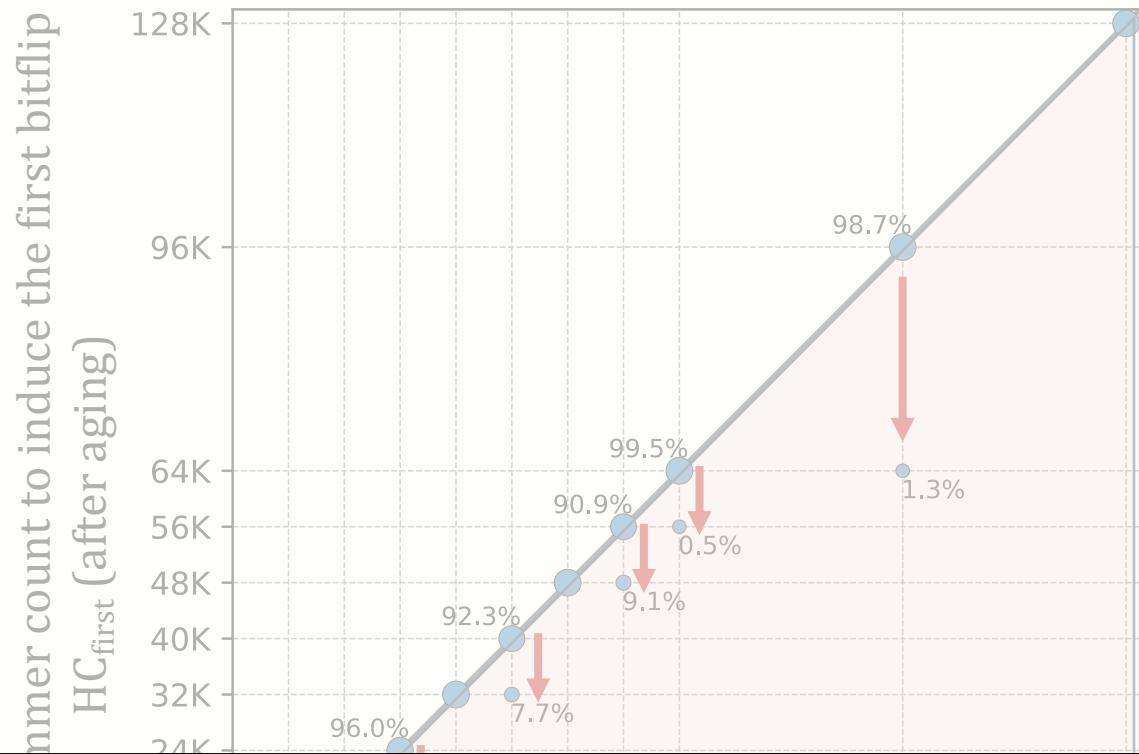
Aging can lead to read disturbance bitflips at **smaller** hammer counts



Deeper Understanding of Physics and Vulnerabilities

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

Aging can lead to read disturbance bitflips at smaller hammer counts



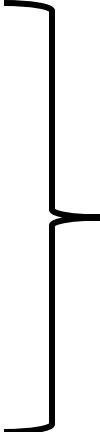
Future work:
rigorous aging characterization
and **online profiling of read disturbance vulnerability**

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

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
 - RowHammer
 - RowPress
 - Data retention time errors
 - Variable retention time
 - ...

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
- What is **the worst-case**?
 - Temperature
 - Data pattern
 - Memory access pattern
 - Spatial variation
 - Voltage

What is **the worst-case** considering all **these sensitivities**?

What is **the minimum hammer count** to induce a read disturbance bitflip?

Deeper Understanding of Physics and Vulnerabilities

- The effect of **aging**
- **Interactions** across different error mechanisms
- What is **the worst-case**?

How reliable are our DRAM chips?

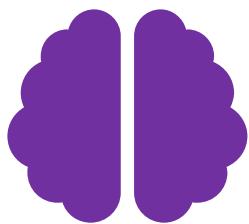
How reliable will our DRAM chips be tomorrow?

We **do not** know! This is an **open research problem**

Future Research for Better Memory Systems



Deeper Understanding of
Physics and Vulnerabilities



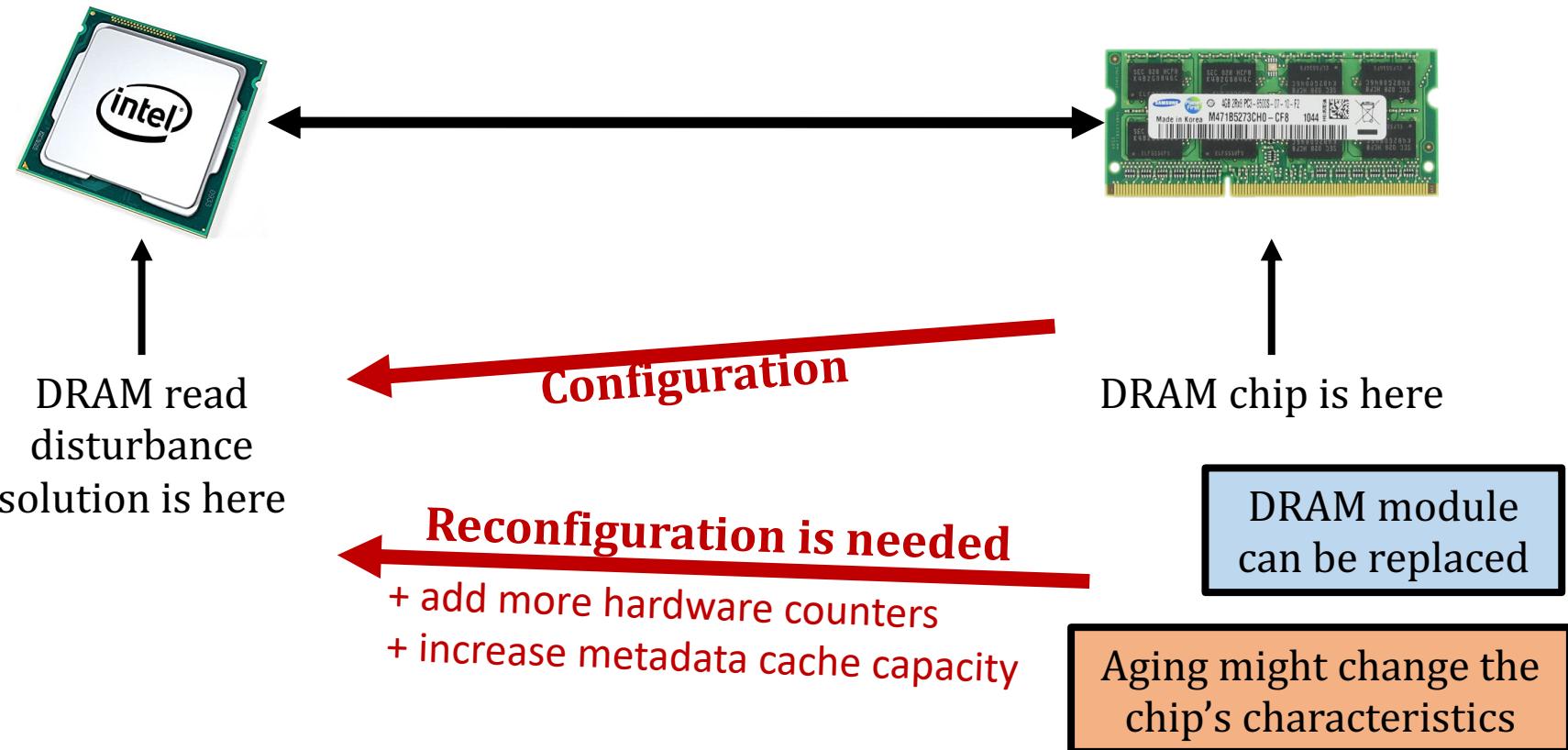
Flexible and Intelligent Memory
Chips, Interfaces, Controllers



Cross-Layer
Communication

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary



Deployed solutions should be patchable in field

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary
- Interfaces should be **more flexible**



Memory controller
decides what should
be done when

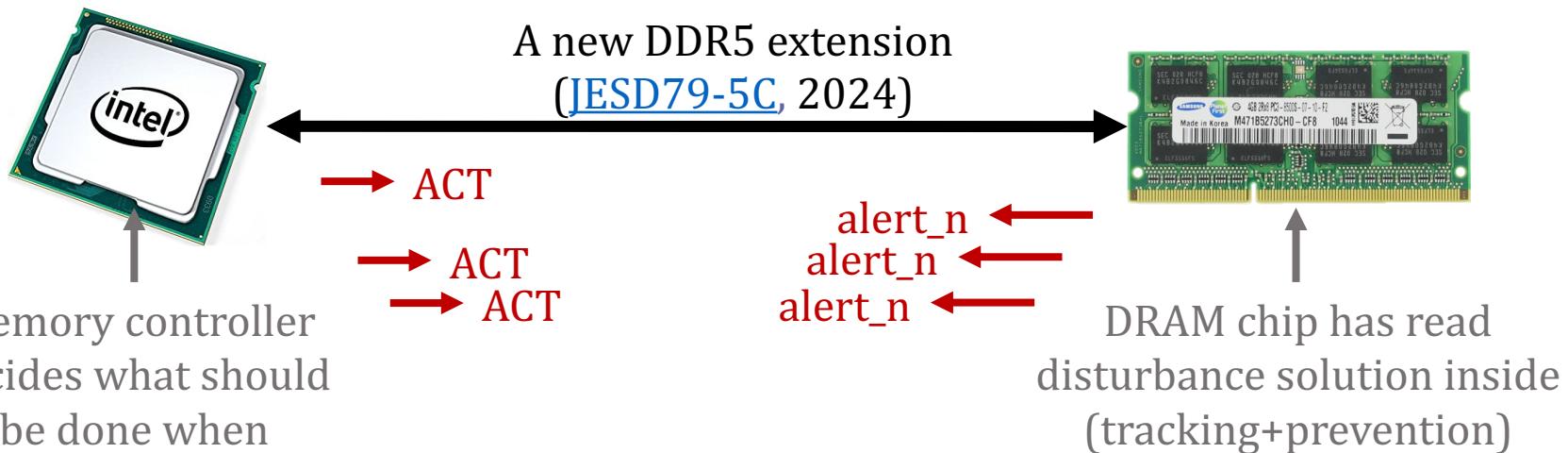
DRAM chip has read
disturbance solution inside
(tracking+prevention)

- The memory controller should provide the DRAM chip with **necessary time window** to perform **preventive actions (e.g., refreshing rows)**
- The memory controller **does not have** the tracking information
- Communicating is **not straightforward** due to strict communication protocol

A more flexible interface is necessary

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary
- Interfaces should be **more flexible**



- **Two Key Issues:**
- A naïve implementation can lead to **very high command bus occupancy**
 - Very high bandwidth consumption
 - Poor parallelism
- DRAM self-manages preventive refreshes. Why not **other maintenance routines?**

Self-Managing DRAM

The three major DRAM maintenance operations:

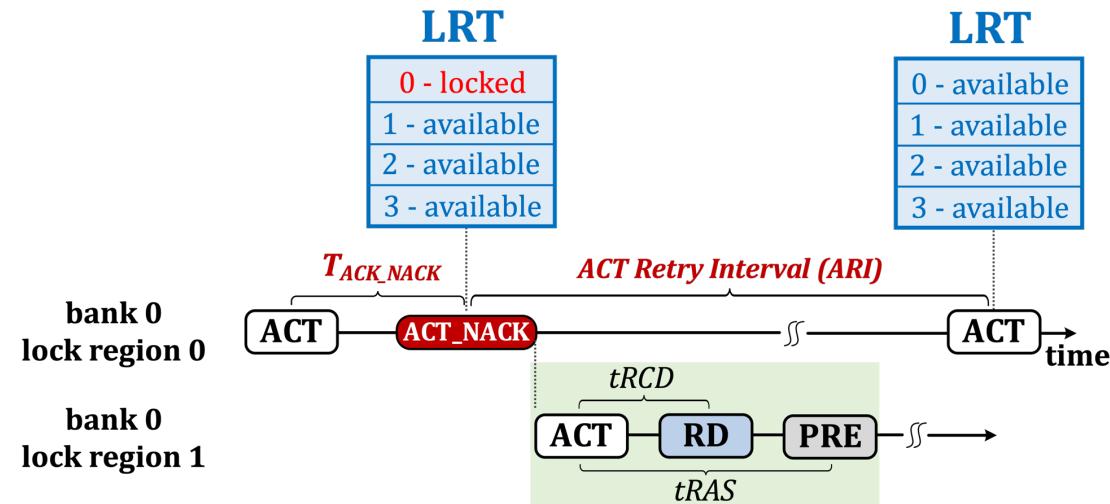
- ❖ Refresh
- ❖ RowHammer Protection
- ❖ Memory Scrubbing

Implementing new **maintenance mechanisms** often requires **difficult-to-realize changes**

Self-Managing DRAM (SMD)

Enables implementing

- new **in-DRAM** maintenance mechanisms
- with **no further changes** in the *DRAM interface* and the *memory controller*



SMD-based *maintenance operations* provide significant **performance** and **energy** benefits across many system configurations and workloads

Flexible and Intelligent Chips, Interfaces, Controllers

- In-field patching is necessary
- Interfaces should be more flexible
- Memory controllers should be more intelligent in detecting malicious activity
- DRAM chips become more and more vulnerable to RowHammer and RowPress
- Key Insight: Activation counts of benign applications get close to unsafe levels
- Problem: DRAM read disturbance solutions are getting prohibitively expensive
- Research Question: How to identify malicious threads/processes/users?
- More intelligent detection mechanisms are needed
- The memory controller observes all memory accesses

More intelligent memory controllers can help

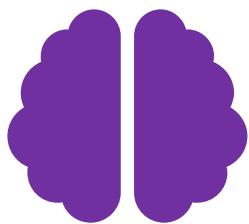
BreakHammer

- **Key Observation:** Mitigating DRAM read disturbance causes delays in memory accesses
- **Our Exploit:** Denial of memory service is possible via triggering mitigation mechanisms
- **Key Idea:** Throttling memory accesses of threads that trigger mitigation mechanisms repeatedly
- **BreakHammer:**
 - Detects the threads that repeatedly trigger the mitigation mechanisms
 - Limits their on-the-fly memory request counts and MSHRs
 - Near-zero area overhead and no additional memory access latency
- **Evaluation:**
 - Improves **system performance** by **48.7%** on average (**105.5%** max)
 - Reduces the **maximum slowdown** by **14.6%** on average

Future Research for Better Memory Systems



Deeper Understanding of
Physics and Vulnerabilities

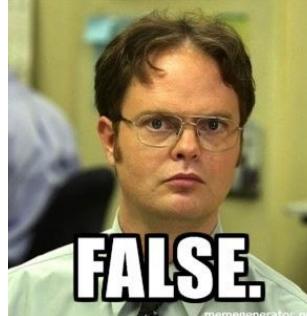


Flexible and Intelligent Memory
Chips, Interfaces, Controllers



Cross-Layer
Communication

Cross-Layer Communication

	Detection	Mitigation
Software	<ul style="list-style-type: none">• Memory allocations• Memory access patterns• Control flow patterns• Time / power measurements	 + +
uArch	<ul style="list-style-type: none">• Memory request scheduling• Speculative execution• Prefetching, branch prediction• Power management	+ +
Device	<ul style="list-style-type: none">• Bitflips occur• Memory isolation is broken	~

How Large is 1000 Activations?

- Bitflips occur at **~1000 activations**
- Mitigation mechanisms trigger **preventive actions** (e.g., preventive refresh) at **~500 activations**
- Is 500 a **distinctive activation count**?
- Benign workloads activate **hundreds of rows** more than **500 times** in a refresh window

Memory intensive workloads

from SPEC'06, SPEC'17, TPC, YCSB, and MediaBench

Workload	MPKI	ACT-64+	ACT-128+	ACT-512+
429.mcf	68.27	2564	2564	2564
470.lbm	28.09	7089	6596	664
462.libquantum	25.95	1	0	0
549.fotonik3d	25.28	10065	88	0
459.GemsFDTD	24.93	10572	218	0
519.lbm	24.37	5824	5455	2482
434.zeusmp	22.24	11085	4825	292
510.parest	17.79	803	185	94
433.milc	17.22	321	92	0
437.leslie3d	15.82	4678	631	7
483.xalancbmk	13.67	4354	776	113
482.sphinx3	12.59	1385	762	304
505.mcf	11.35	1582	1384	732
471.omnetpp	10.72	1015	419	122
tpch2	9.09	875	307	88
520.omnetpp	9.00	1185	84	32
tpch17	7.43	1196	158	26
473.astar	5.18	5957	22	0
436.cactusADM	4.94	6151	2354	1134
jp2_encode	4.18	0	0	0

Benign workloads **might not be so benign**

Cross-Layer Communication

	Detection	Mitigation
Software	<ul style="list-style-type: none">• Memory allocations• Memory access patterns• Control flow patterns• Time / power measurements	 + +
Memory / Power / CPU		
Cross-layer communication is crucial going forward		
Device	<ul style="list-style-type: none">• Bitflips occur• Memory isolation is broken	+ + ~