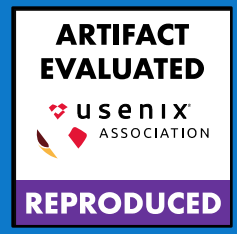


ABACuS

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

Ataberk Olgun | Yahya Can Tuğrul | F. Nisa Bostancı

İsmail Emir Yüksel | Haocong Luo | Steve Rhyner

A. Giray Yağlıkçı | Geraldo F. Oliveira | Onur Mutlu

SAFARI

ETH zürich

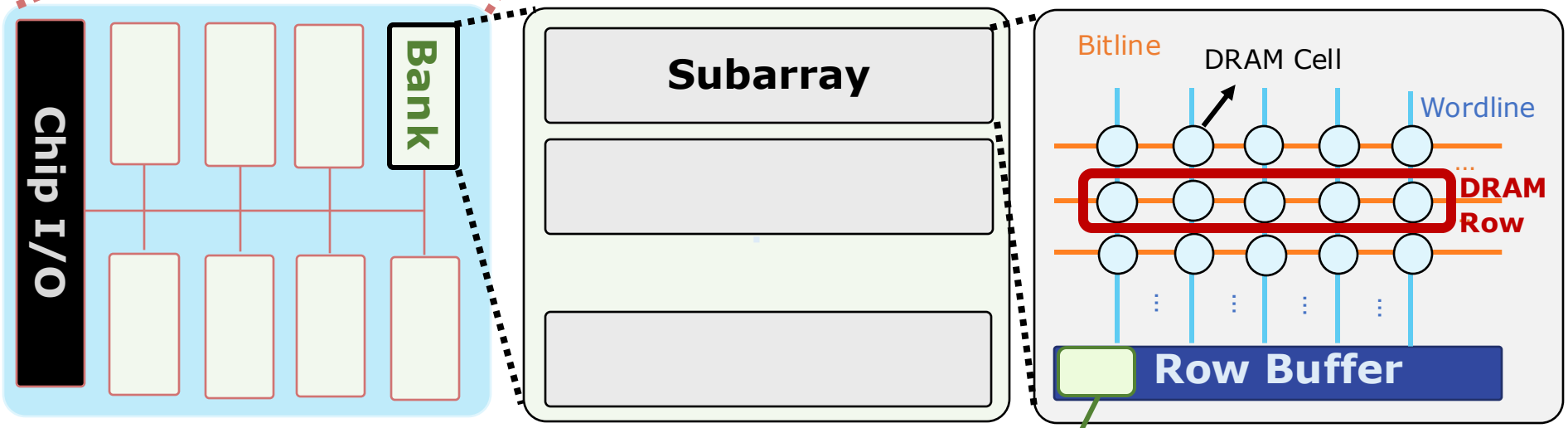
Outline

1. Background & Motivation
2. ABACuS: Key Idea and Mechanism
3. Evaluation
4. Conclusion

Outline

1. Background & Motivation
2. ABACuS: Key Idea and Mechanism
3. Evaluation
4. Conclusion

DRAM Organization

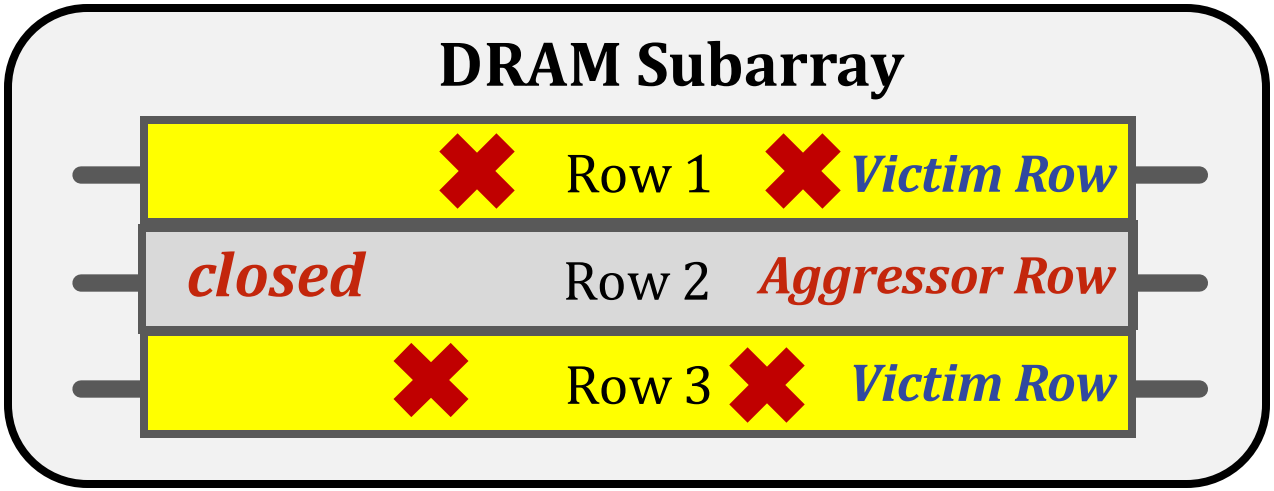


DRAM Chip **DRAM Bank** **DRAM Subarray**

Cache block (e.g., 512 bits)

DRAM Read Disturbance

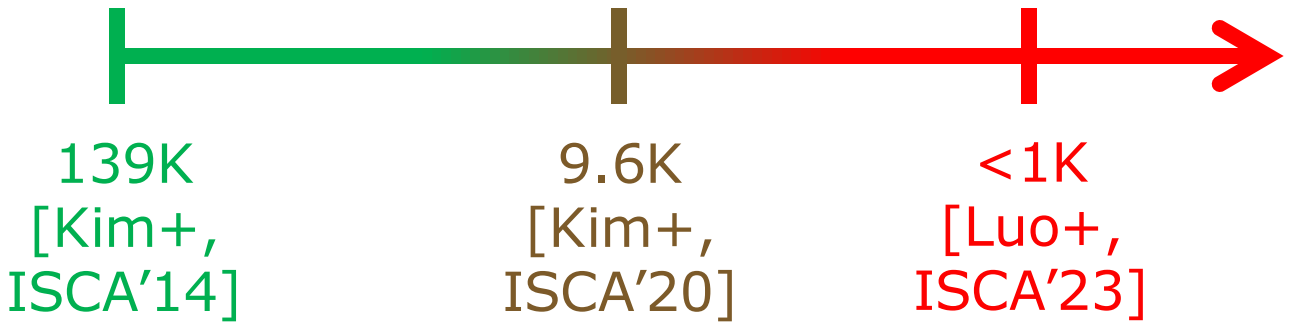
- Read disturbance in DRAM breaks memory isolation
- **Prominent example: RowHammer**



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

Read Disturbance Worsens

- Read disturbance bitflips occur at much smaller row activation counts
  - More than 100x decrease in less than a decade



Mitigation techniques against read disturbance attacks need to be effective and efficient for highly vulnerable systems

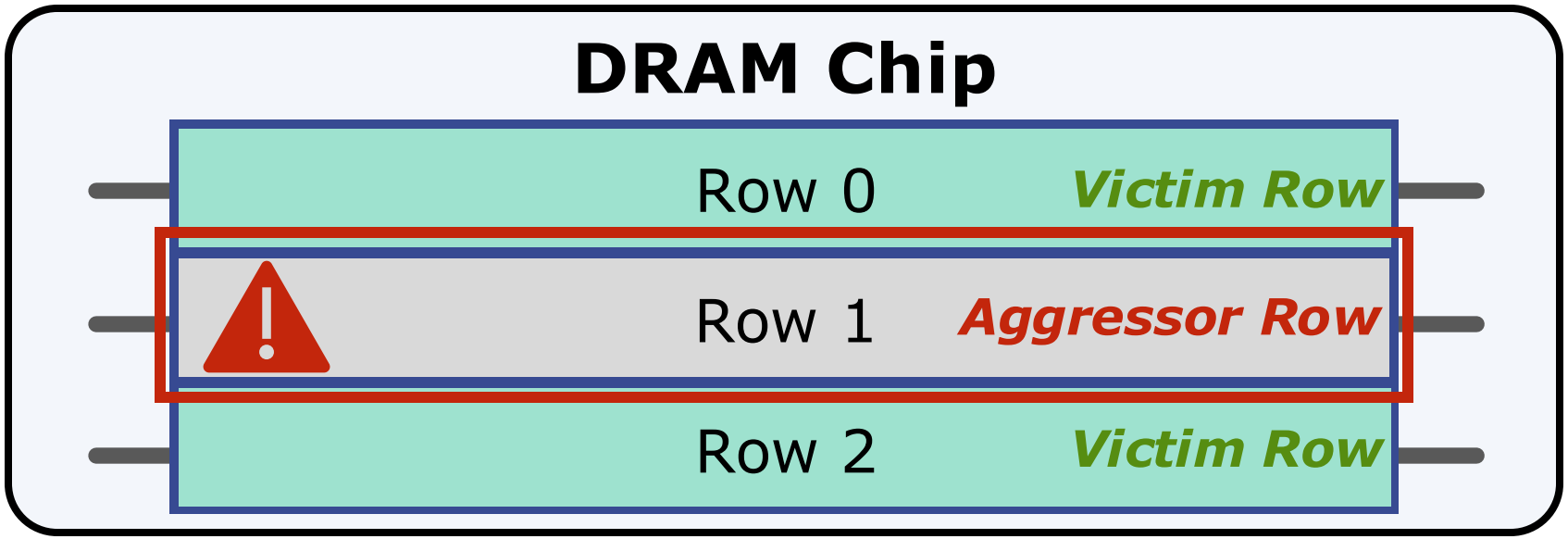
Read Disturbance Mitigation Approaches

There are many ways to mitigate RowHammer bitflips

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

Generally more resource-efficient and lower overhead than other approaches

Preventive Refresh



Refreshing potential victim rows
mitigates read disturbance bitflips

Requires aggressor row activation count estimation or tracking

Problem & Goal

Problem

No existing mitigation technique prevents RowHammer bitflips
at low area, performance and energy costs

Goal

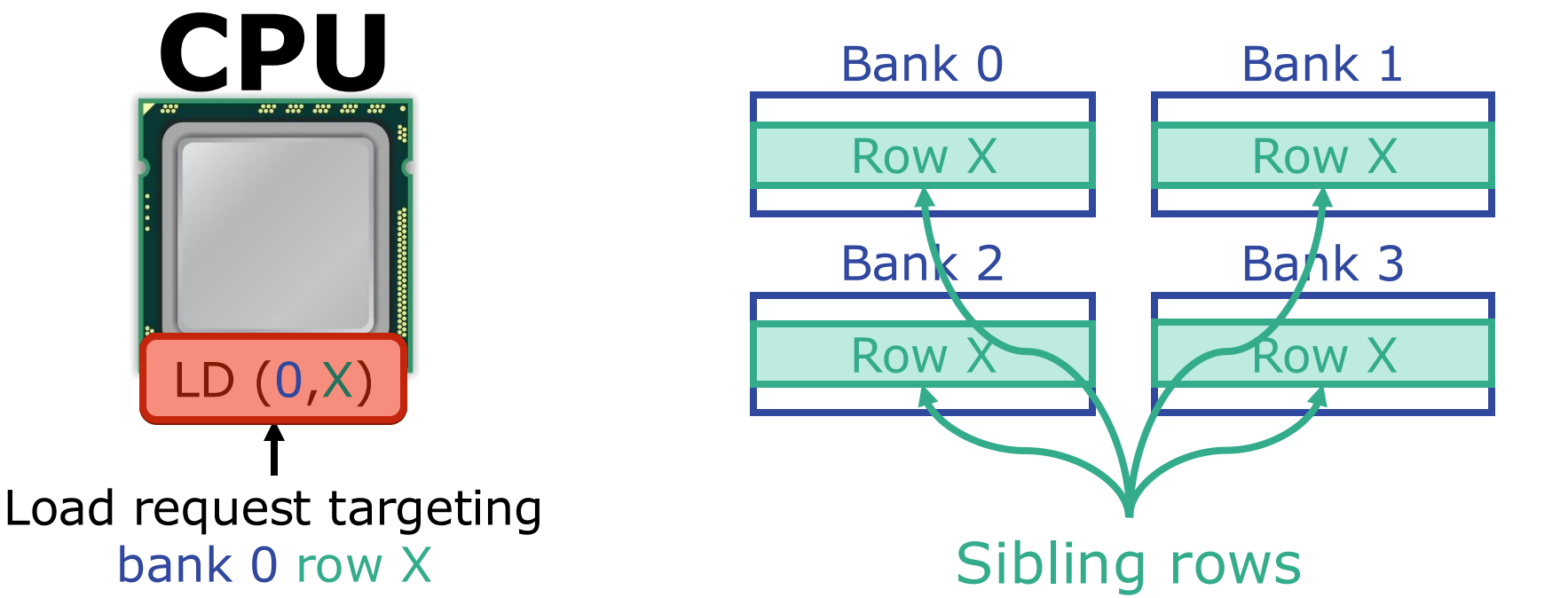
Prevent RowHammer bitflips
at low performance, energy, and area cost
especially at very low RowHammer thresholds
(e.g., 125 aggressor row activations induce a bitflip)

Outline

1. Background & Motivation
2. ABACuS: Key Idea and Mechanism
3. Evaluation
4. Conclusion

Key Observation

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

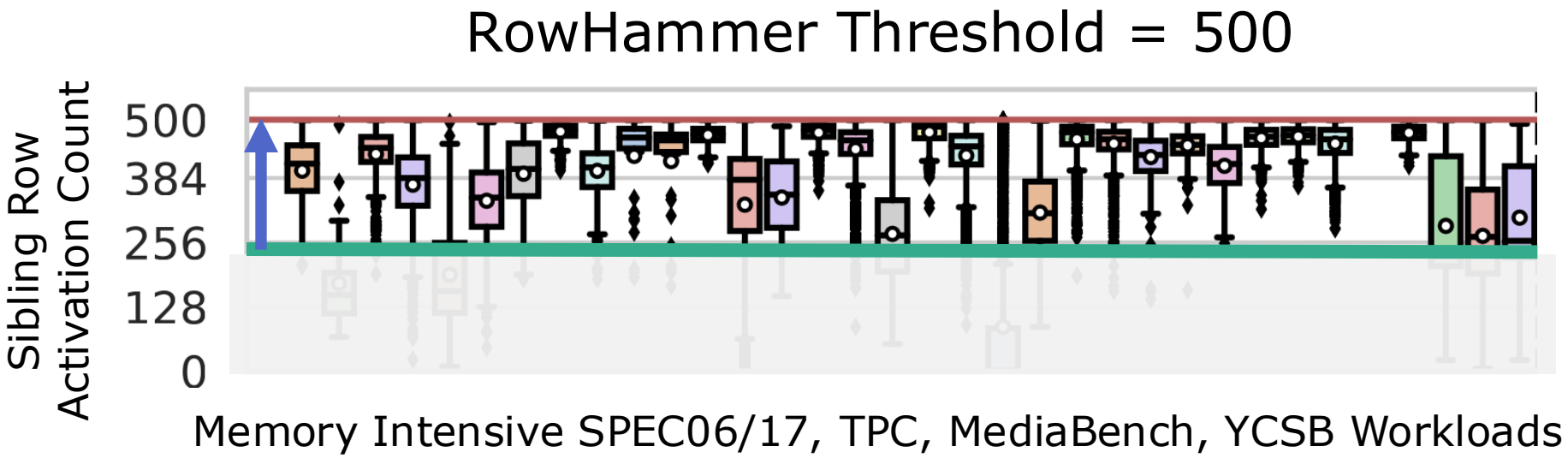


Explanation for the Key Observation

1. Spatial locality in memory accesses
(e.g., [Smith+, ACM CSUR 1982])
   - A program tends to access neighboring cache blocks at around the same time
   - e.g., a streaming access to an array

2. Modern physical → DRAM address mappings
(e.g., [Pessl+, USENIX Security 2016 and Kaseridis+, MICRO 2011])
   - Place neighboring cache blocks into different banks, but into the same row
   - Leverage DRAM bank-level parallelism for higher-throughput DRAM access

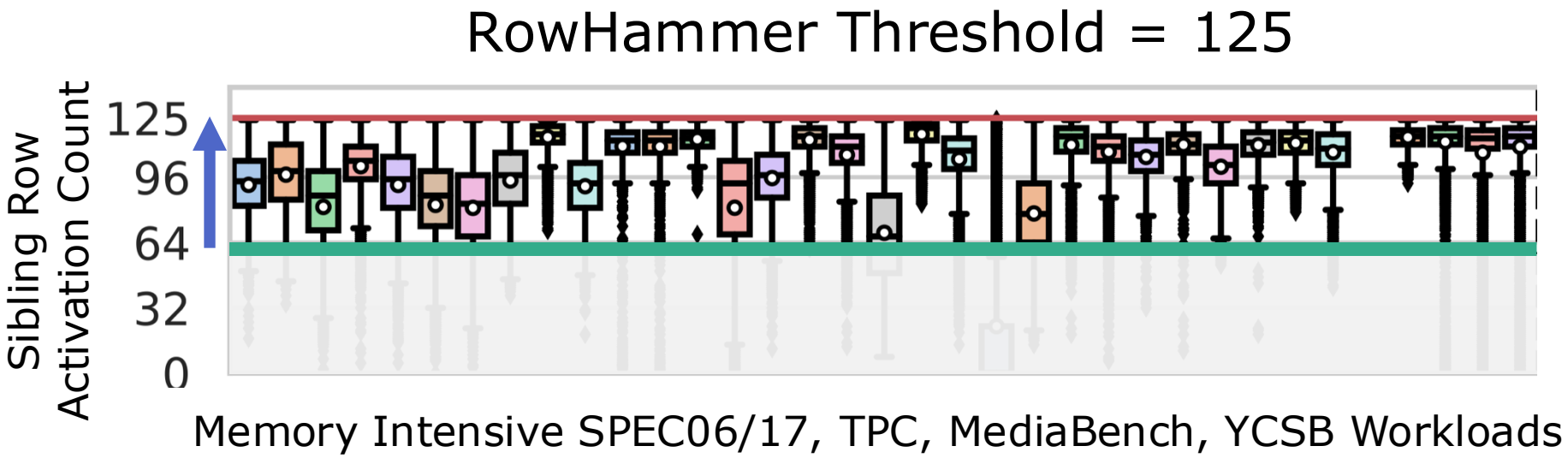
Sibling Row Activation Count for RowHammer Threshold = 500



If a row is activated 500 times
its siblings are likely activated more than 250 times

The sibling row with the highest activation count
yields a good estimate for the activation count of all siblings

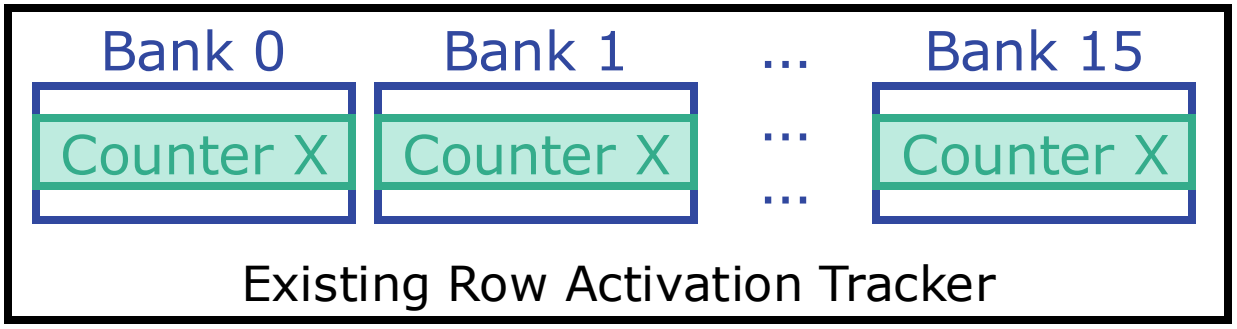
Sibling Row Activation Count for RowHammer Threshold = 125



The sibling row with the highest activation count yields an even better estimate for the activation count of all sibling rows

Existing Per-Bank Activation Counters Induce High Storage Overhead

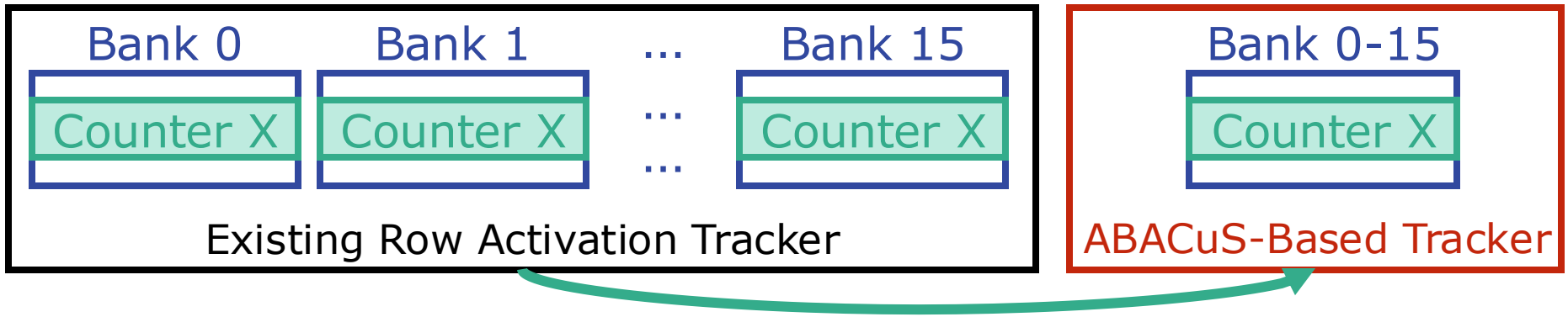
- There are many (e.g., 16) banks in a DRAM chip
  - Newer DRAM standards (DDR5) have more (32) banks
  - # of activation counters linearly increases with # of banks



Need twice as many counters for DDR5

ABACuS: Key Idea

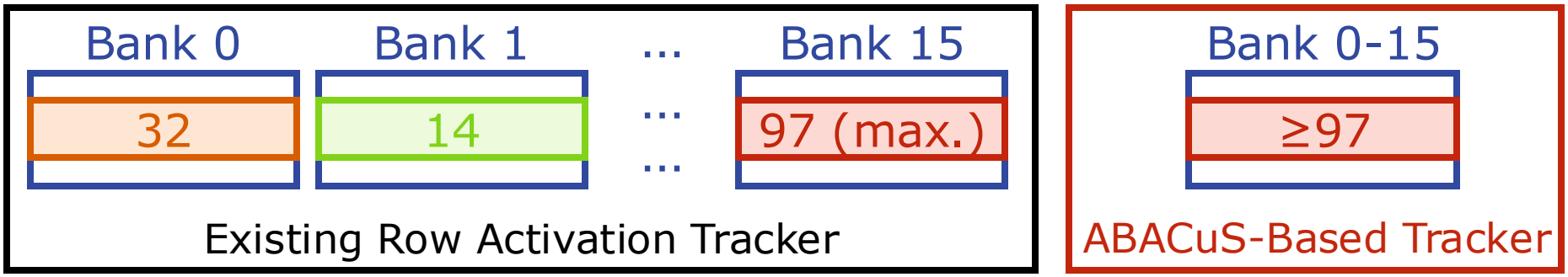
- There are many (e.g., 16) banks in a DRAM chip
  - Newer DRAM standards (DDR5) have more (32) banks
  - # of activation counters linearly increases with # of banks
- Sibling rows have similar activation counts
- Have one counter for all siblings
  - Reduce the number of counters by a factor of the number of banks



16x reduction in number of counters

Maximum Activation Count

- Track the maximum (worst) activation count across all sibling rows using one counter



ABACuS counter value vs. maximum activation count

If ABACuS counter is smaller

- Cannot preventively refresh on time
- Cannot mitigate bitflips
- Not secure

If ABACuS counter is larger

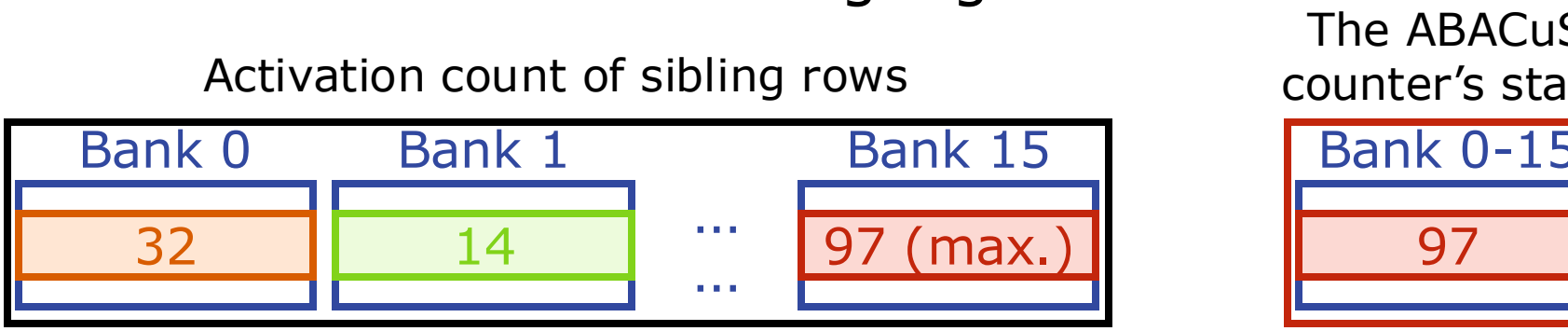
- Unnecessary preventive refreshes
- Higher perf. and energy overheads
- Lower performance

Our design goal

ABACuS counter value == maximum activation count

ABACuS Counting Algorithm

- Intuition behind the counting algorithm



- ABACuS "remembers" the sibling row whose activation increased the counter value to 97
  - The row in bank 15 in this example

Increment ABACuS counter if
1) bank 15 is activated again OR 2) any other bank is activated twice

- Need one bit per bank to store additional state

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

ABACuS: Implementation

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

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

RowHammer threshold (N_{RH}), has reduced by more than an order of magnitude in less than a decade [14].[1] As many prior works demonstrate on real systems [1,2,4,15,20–83], RowHammer bitflips can lead to security exploits that 1) take over a system, 2) leak security-critical or private data, and 3) manipulate safety-critical applications' behavior in undesirable ways. As a result, a large body of work [1,15,19,38,44,55,84–88,88–135] proposes mitigation mechanisms to prevent RowHammer bitflips.

Key Problem. Many prior works (e.g., [1,98,102,106,107,110,112,116,117,125,134,135]) propose using a set of counters to track the activation counts of potential aggressor rows (counter-based mechanisms). Using counters to determine

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

Outline

1. Background & Motivation

2. ABACuS: Key Idea and Mechanism

3. Evaluation

4. Conclusion

Evaluation Methodology

- **Performance and energy consumption evaluation:** Cycle-level simulations using Ramulator [Kim+, CAL 2015] and DRAMPower [Chandrasekar+, DATE 2013]

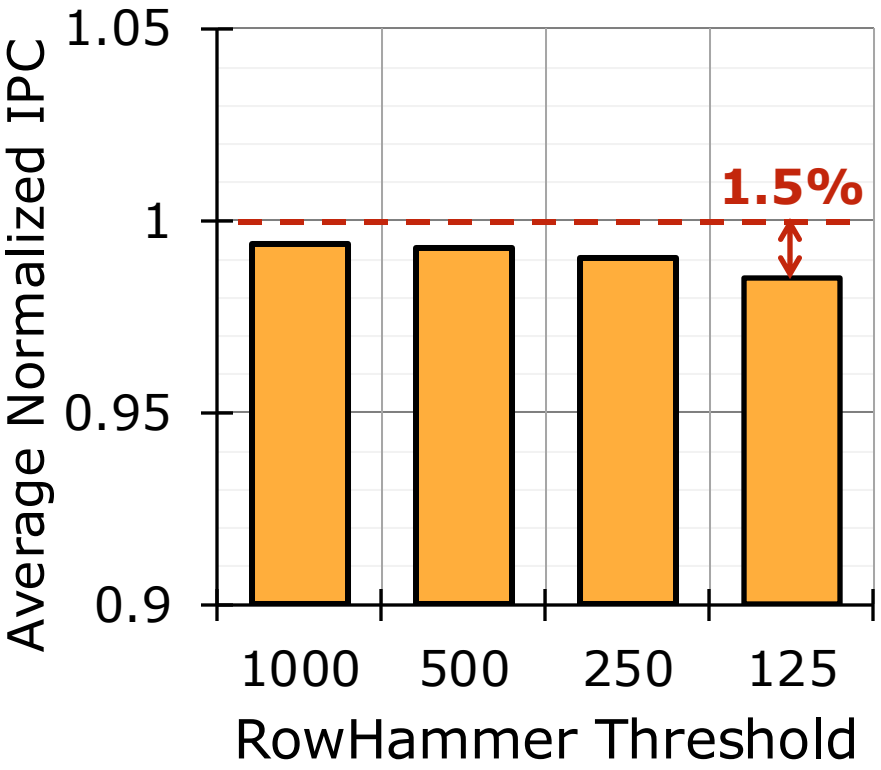
- **System Configuration:**

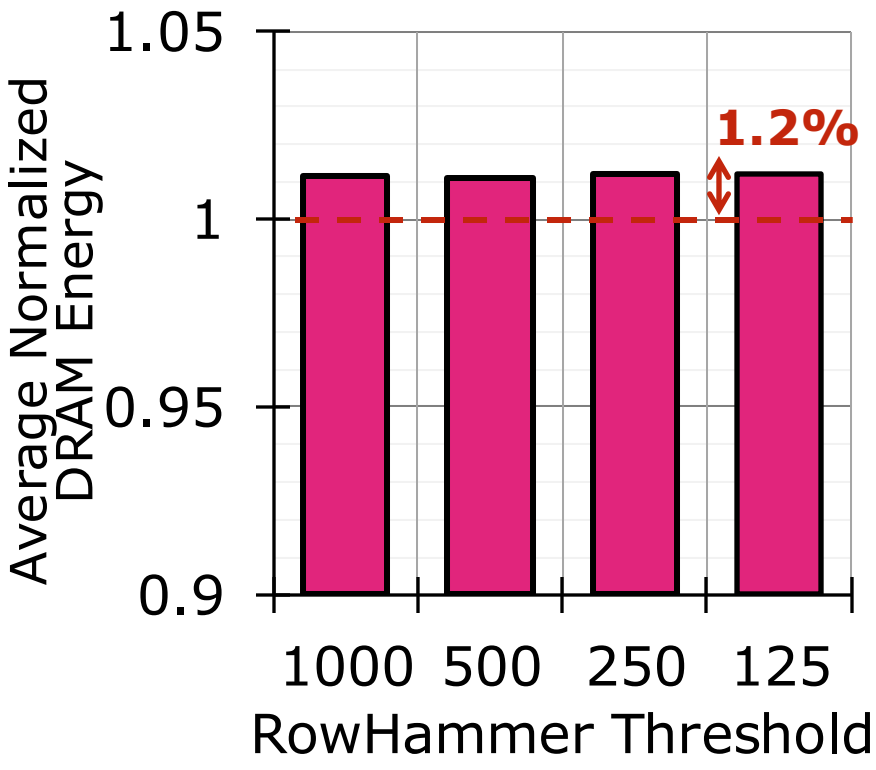
Processor	1 or 8 cores, 3.6GHz clock frequency, 4-wide issue, 128-entry instruction window
DRAM	DDR4, 1 channel, 2 rank/channel, 4 bank groups, 4 banks/bank group, 128K rows/bank, 3200 MT/s
Memory Ctrl.	64-entry read and write requests queues, Scheduling policy: FR-FCFS with a column cap of 16 Last-Level Cache 2 MiB (single-core), 16 MiB (8-core)

- **Comparison Points:** 4 state-of-the-art RowHammer mitigations
  - Graphene (best performing), Hydra (area-optimized best performing), Low Processor Chip Area Cost: REGA, PARA

- **Workloads:** 62 1- & 8-core (multiprogrammed) workloads
  - SPEC CPU2006, SPEC CPU2017, TPC, MediaBench, YCSB

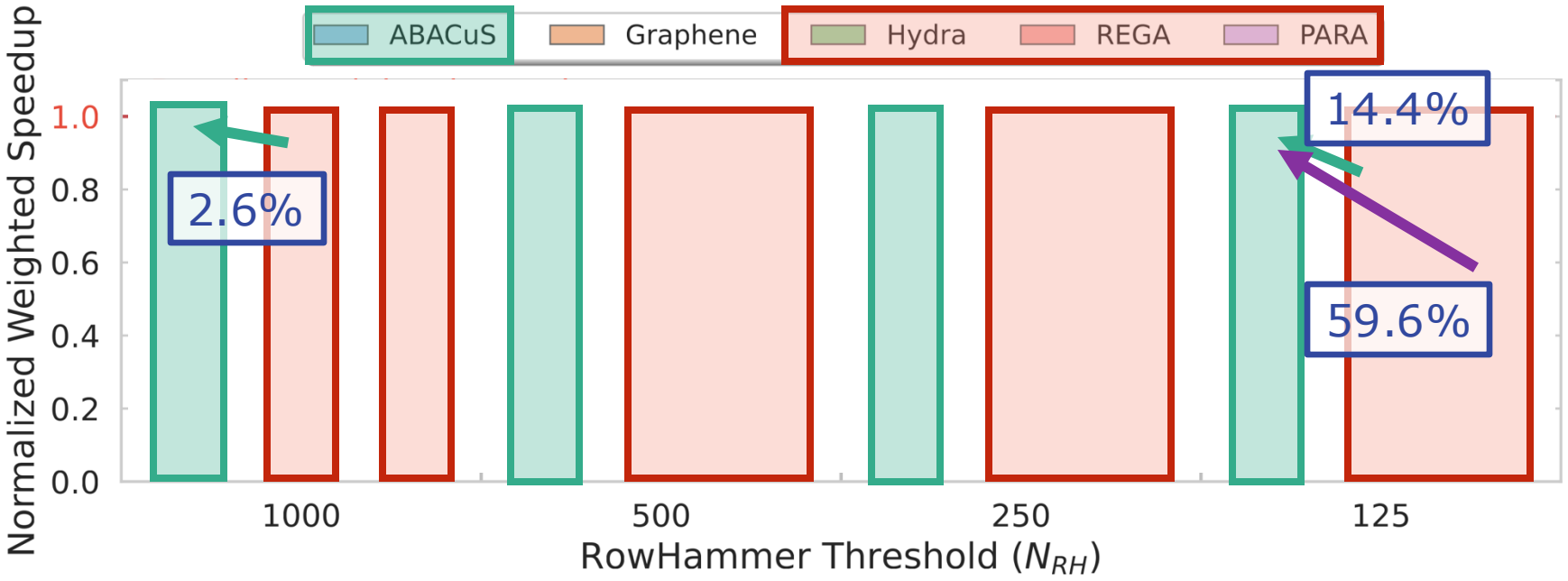
Single-Core Performance and Energy





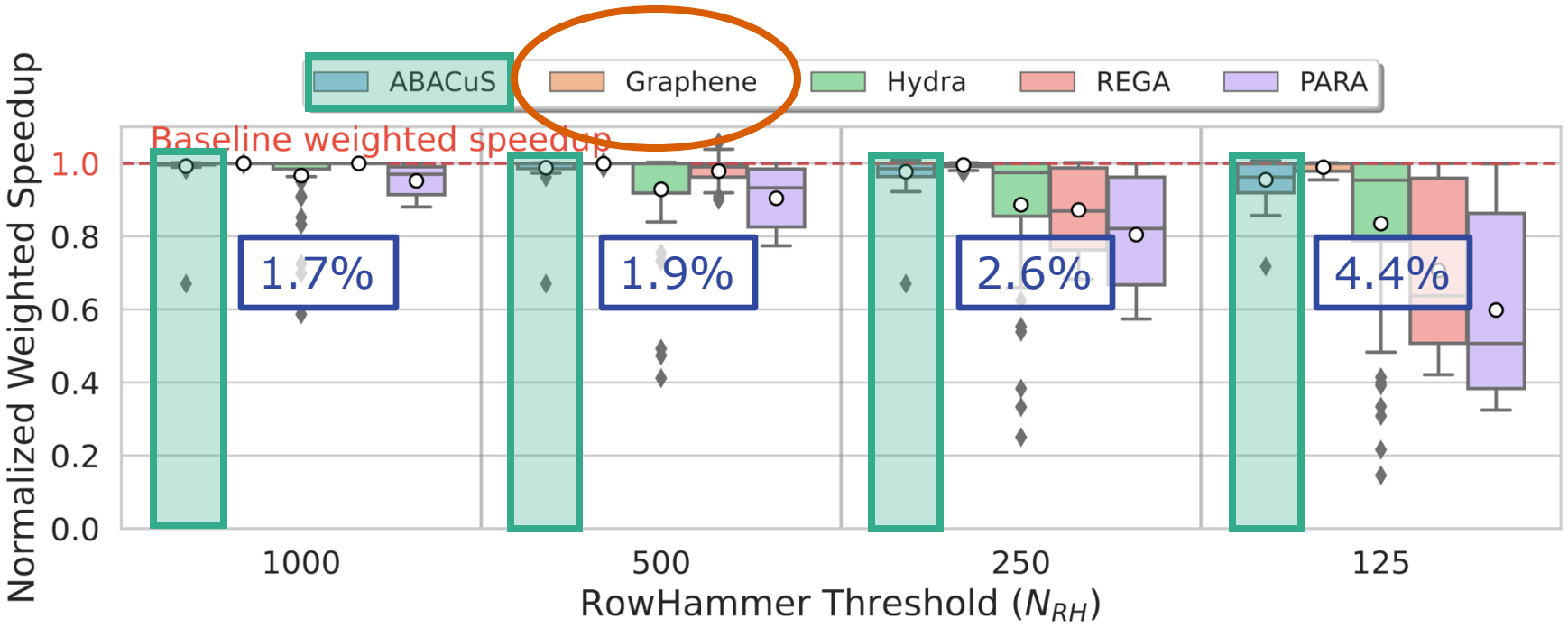
ABACuS prevents bitflips with very small
average performance and DRAM energy overheads
compared to a baseline system with *no* RowHammer mitigation

8-Core Performance Comparison



ABACuS outperforms Hydra and PARA at all RowHammer thresholds

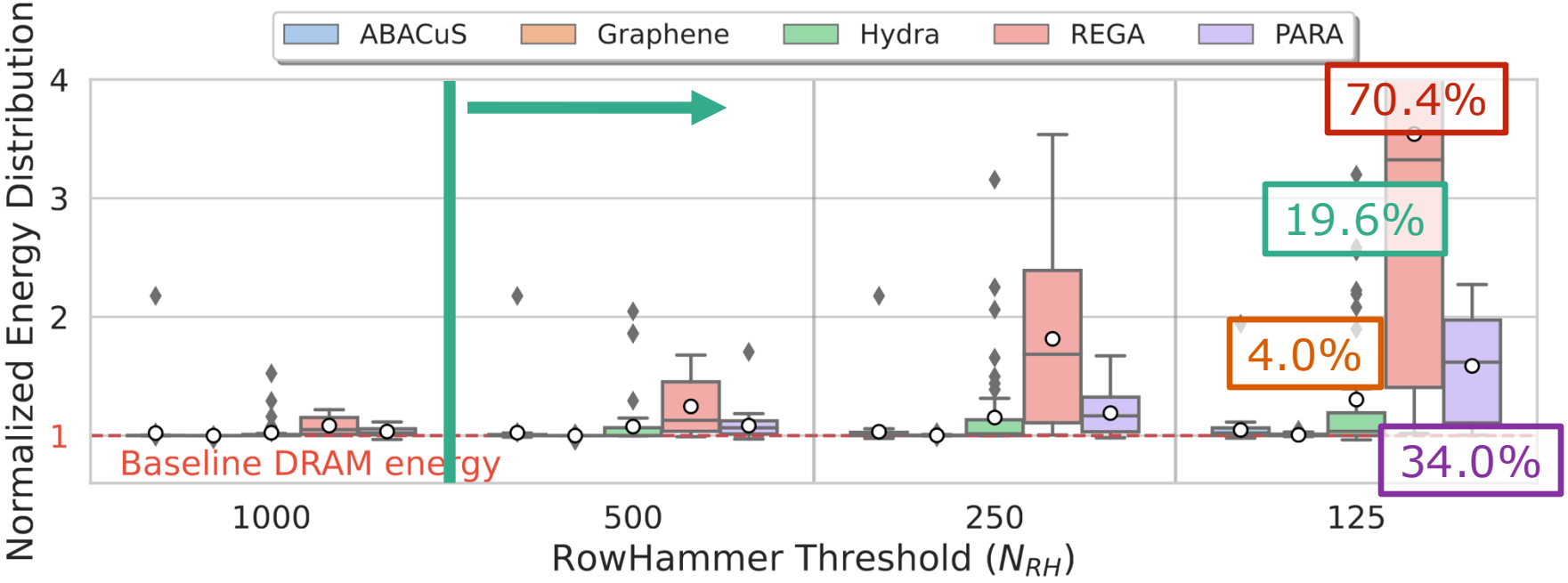
8-Core Performance Comparison



ABACuS outperforms Hydra and PARA at all RowHammer thresholds

ABACuS incurs a small performance overhead over Graphene

8-Core DRAM Energy Comparison

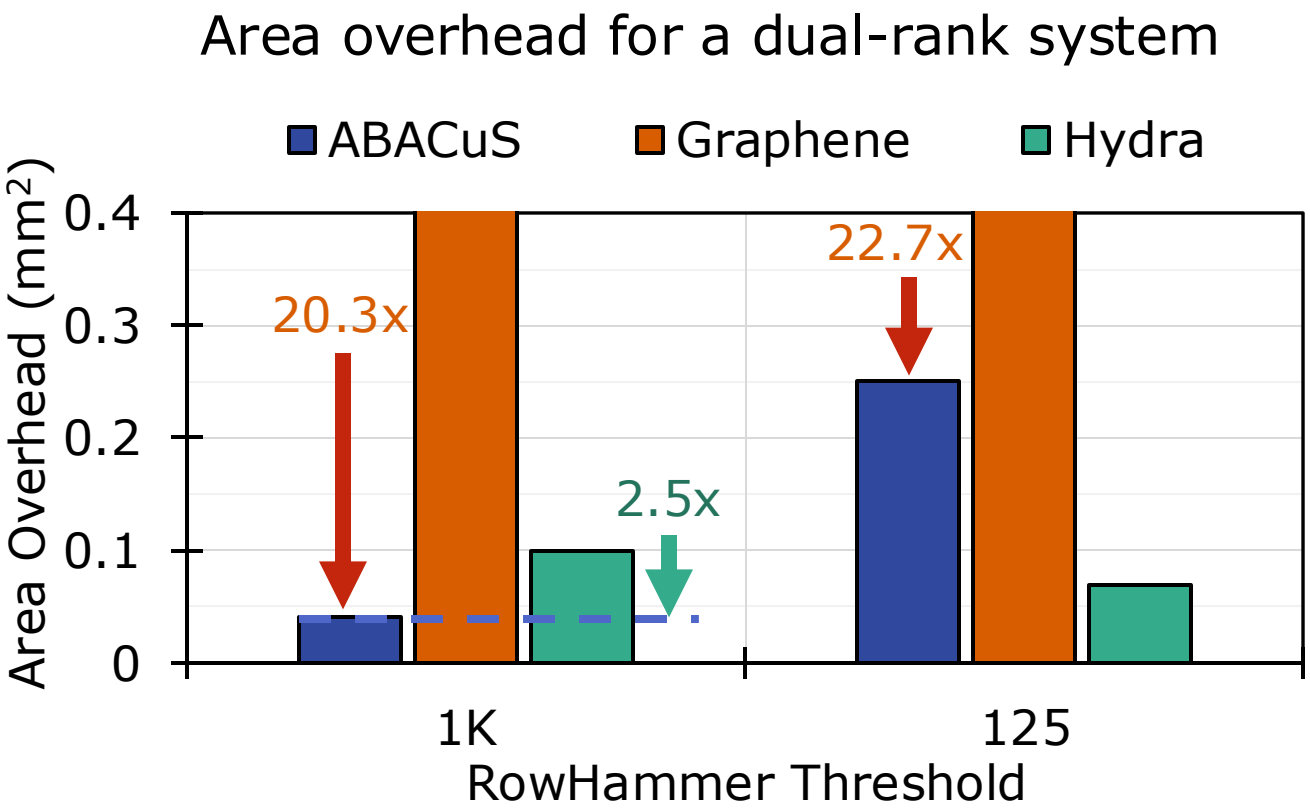


ABACuS consumes less energy than Hydra, REGA, and PARA for RowHammer thresholds smaller than 1000

ABACuS incurs a small DRAM energy over Graphene

Area Overhead

Area overhead analysis using
CACTI [Balasubramonian+, ACM TACO 2017]



*REGA modifies DRAM chips

*PARA needs a secure random number generator

More in the Paper

- Security analysis of ABACuS:
  - Inductive proof for maximum activation count tracking
- Single-core performance and energy comparison
- Verilog-level circuit area, latency, energy, and power
  - E.g., ABACuS takes 1.2 ns to update one counter
- Performance under adversarial workloads
  - Alternative ABACuS design
- Performance & energy sensitivity analysis
  - Number of ABACuS counters
  - Number of banks
  - DRAM address mapping functions…
- Discussion on accounting for RowPress

The Paper

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

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

RowHammer threshold (N_{RH}), has reduced by more than an order of magnitude in less than a decade [14].[1] As many prior works demonstrate on real systems [1,2,4,15,20–83], RowHammer bitflips can lead to security exploits that 1) take over a system, 2) leak security-critical or private data, and 3) manipulate safety-critical applications' behavior in undesirable ways. As a result, a large body of work [1,15,19,38,44,55,84–88,88–135] proposes mitigation mechanisms to prevent RowHammer bitflips.

Key Problem. Many prior works (e.g., [1,98,102,106,107,110,112,116,117,125,134,135]) propose using a set of counters to track the activation counts of potential aggressor rows (counter-based mechanisms). Using counters to determine

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

Outline

1. Background & Motivation

2. ABACuS: Key Idea and Mechanism

3. Evaluation

4. Conclusion

ABACuS Summary

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

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

Key Results: At very low RowHammer thresholds (e.g., 125), ABACuS:

- Induces small system performance and DRAM energy overhead
- Outperforms the state-of-the-art mitigations Hydra, REGA, and PARA except the highly area costly Graphene
- Induces 22.7X smaller chip area than Graphene

Extended Version on arXiv

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

Computer Science > Cryptography and Security

[Submitted on 15 Oct 2023]

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

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

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

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

ABACuS is Open Source and Artifact Evaluated

CMU-SAFARI / ABACuS

Type / to search

Code Issues Pull requests Actions Projects Security Insights Settings

ABACuS Public

Edit Pins | Unwatch 4 | Fork 0 | Starred 3

main | 1 branch | 0 tags | Go to file | Add file | Code

olgunataberk add verilog sources and update readme ef1c89c yesterday 8 commits

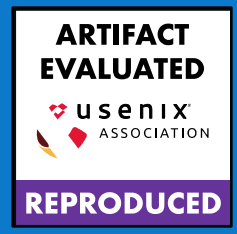
abacus_cacti	add abacus cacti sources	yesterday
abacus_verilog	add verilog sources and update readme	yesterday
configs/ABACUS	Initial commit	3 days ago
ext	Initial commit	3 days ago
scripts	Initial commit	3 days ago
src	Initial commit	3 days ago
.gitignore	Initial commit	3 days ago
CMakeLists.txt	Initial commit	3 days ago
Doxyfile	Initial commit	3 days ago

About

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

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

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



ABACuS

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

Ataberk Olgun | Yahya Can Tuğrul | F. Nisa Bostancı

İsmail Emir Yüksel | Haocong Luo | Steve Rhyner

A. Giray Yağlıkçı | Geraldo F. Oliveira | Onur Mutlu

SAFARI

ETH zürich

Backup Slides

ABACuS Summary

Problem: As DRAM becomes more vulnerable to read disturbance, existing RowHammer mitigation techniques either prevent bitflips

- at high area overheads or
- with prohibitively large performance and energy overheads

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

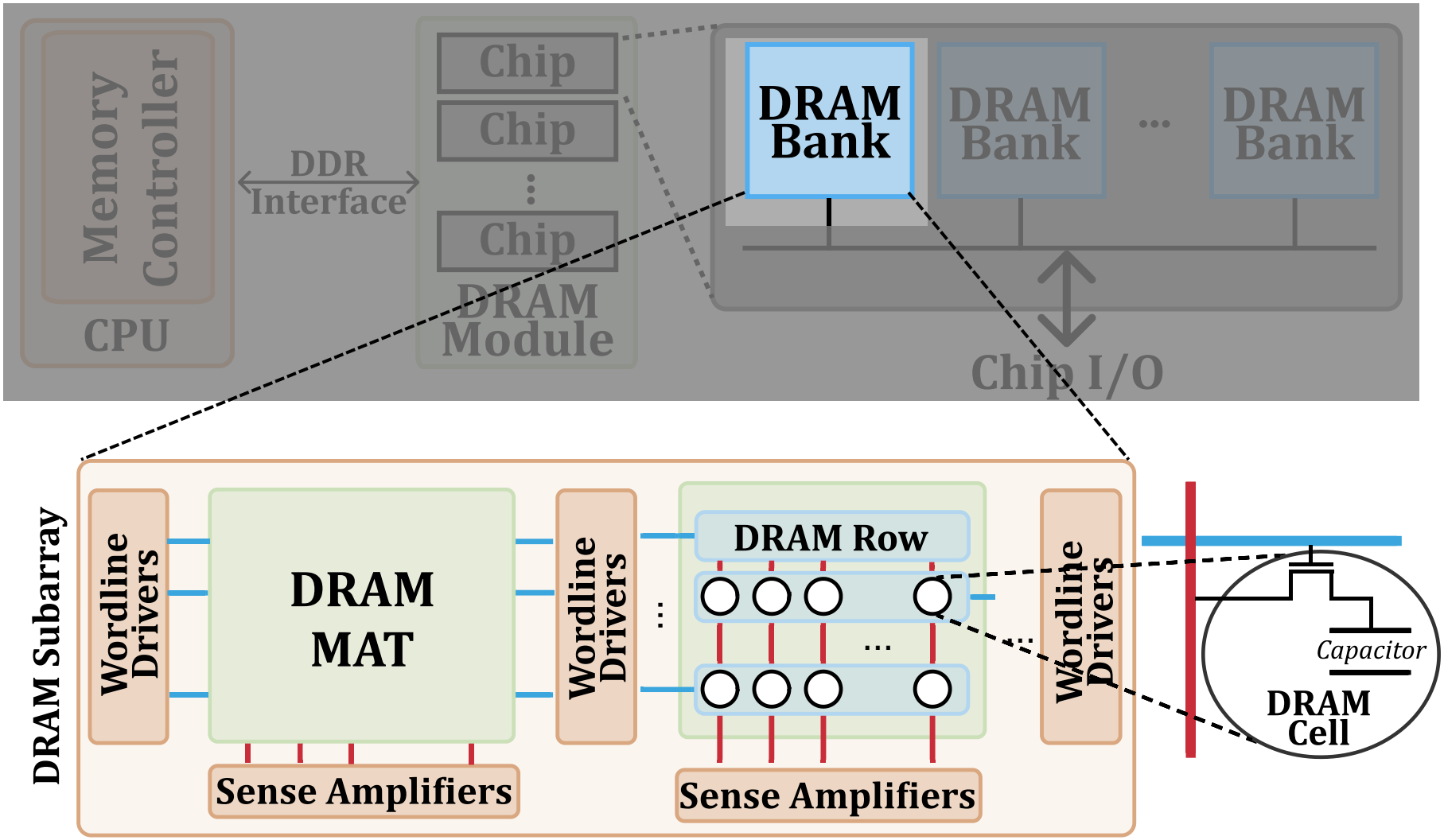
Key Observation: Many workloads access the same row address in different DRAM banks at around the same time

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

Key Results: At very low RowHammer thresholds, ABACuS:

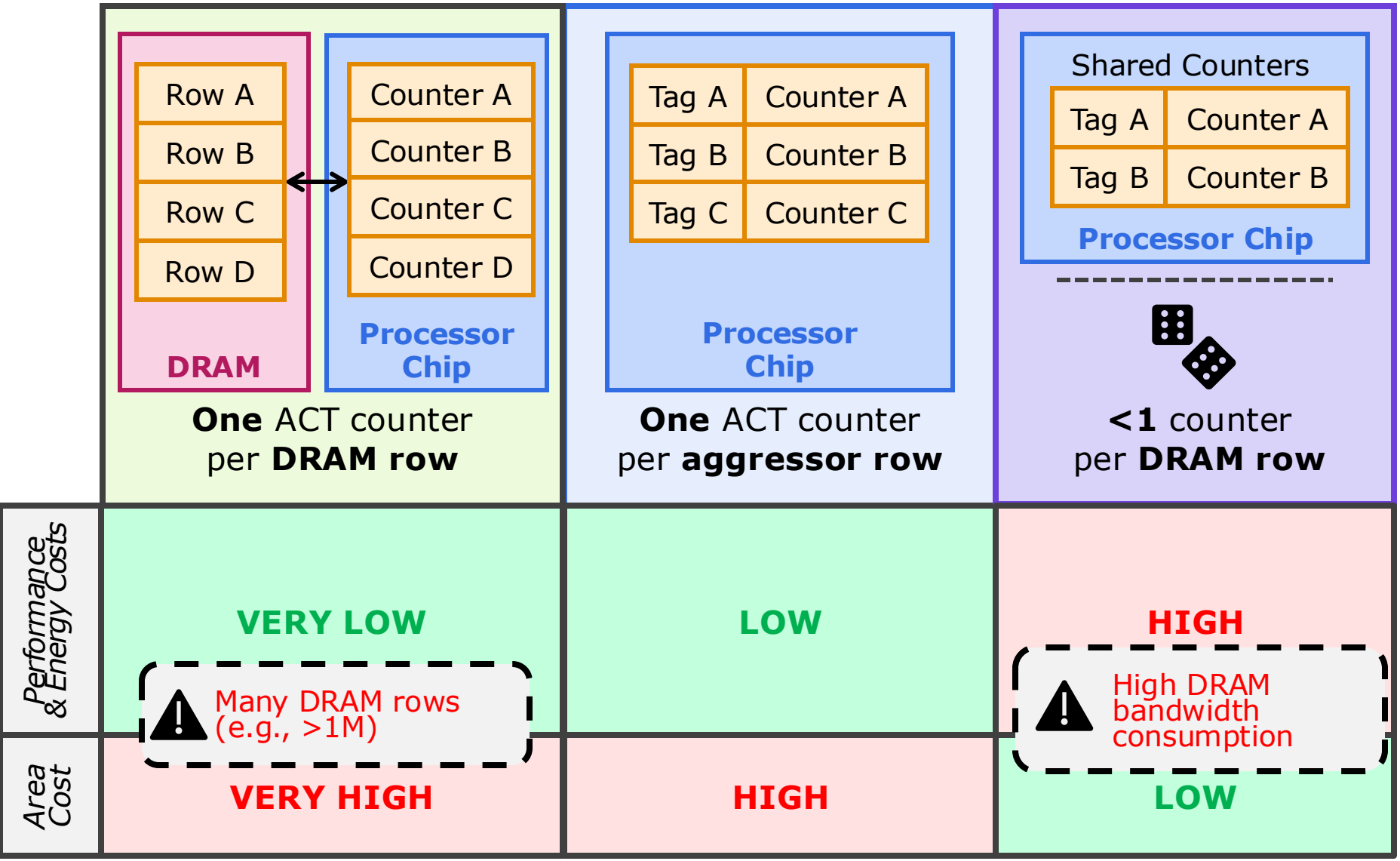
- Induces small system performance and DRAM energy overhead
- Outperforms the state-of-the-art mitigation (Hydra)
- Takes up 22.7X smaller chip area than state-of-the-art (Graphene)

DRAM Organization

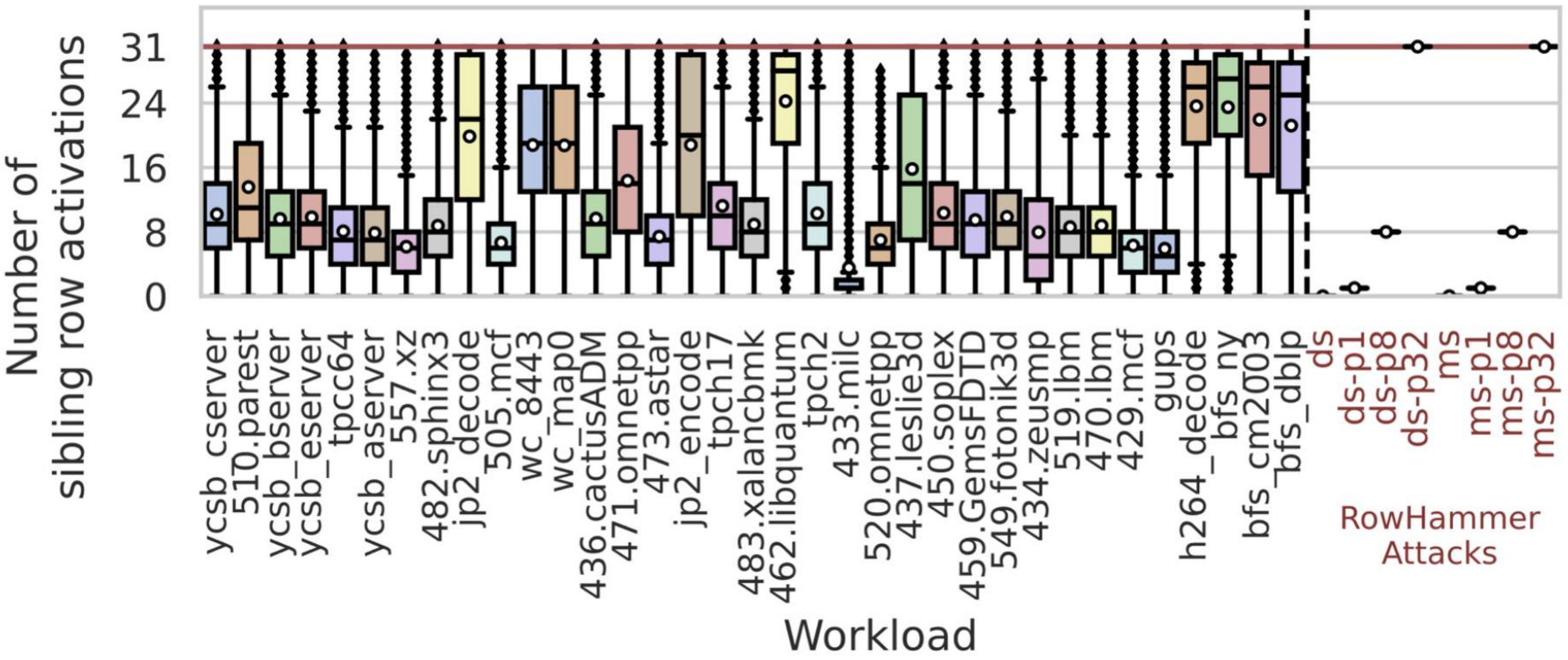


[Olgun+, ISCA'21]

Preventive-Refresh-Based Mitigations

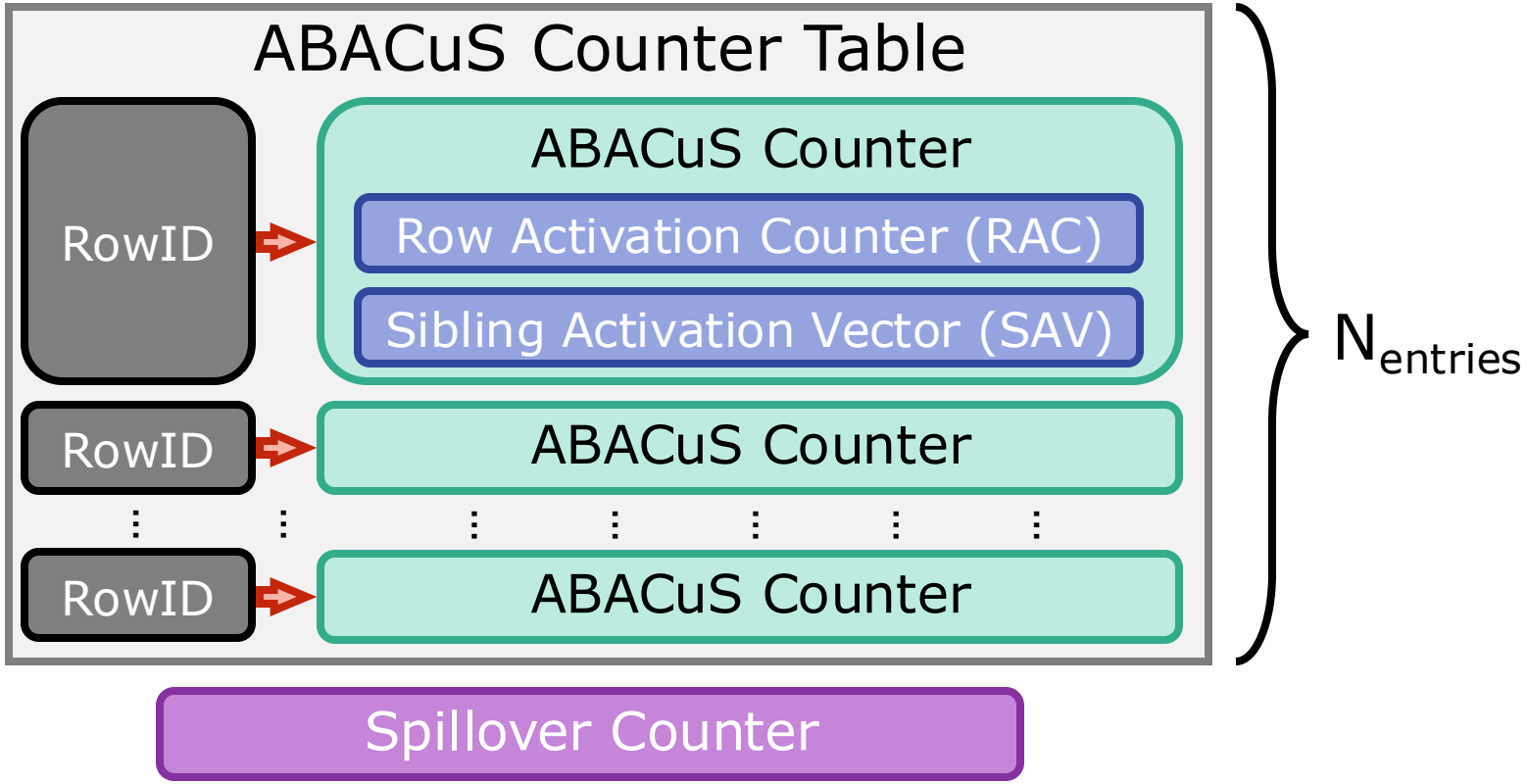


Number of sibling rows activated before one sibling row is activated again

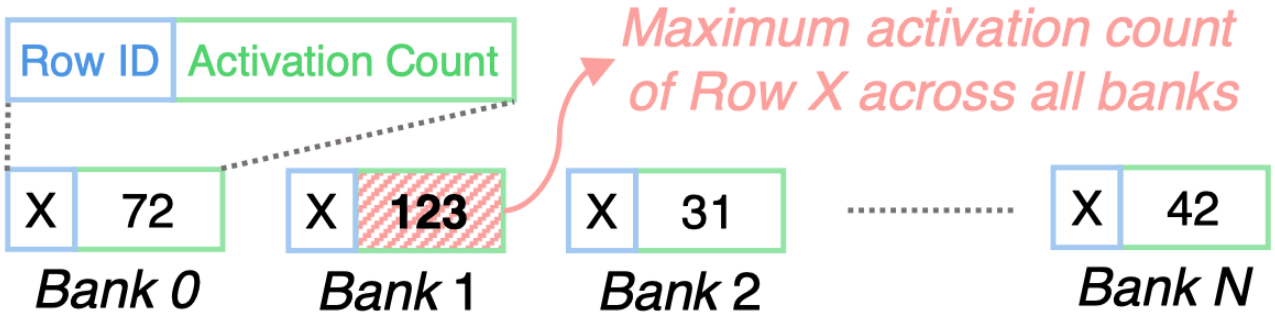


ABACuS: Key Components

- Adopt a frequent item counting algorithm
  - Area-efficient, fewer counters to track more DRAM rows
  - ABACuS is compatible with other counter-based mitigations



ABACuS: Operation



- The RAC always stores the maximum activation count
  - Store small additional information in SAV

"Which siblings were activated since RAC was last incremented?"
One bit per bank

ABACuS Counter

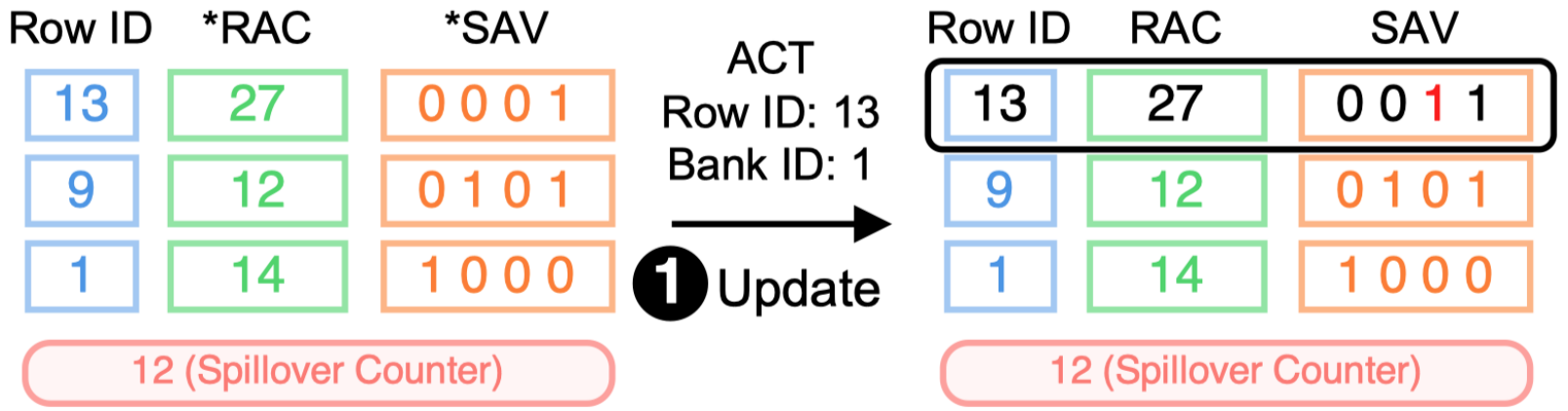
Row Activation Counter (RAC)

Sibling Activation Vector (SAV)

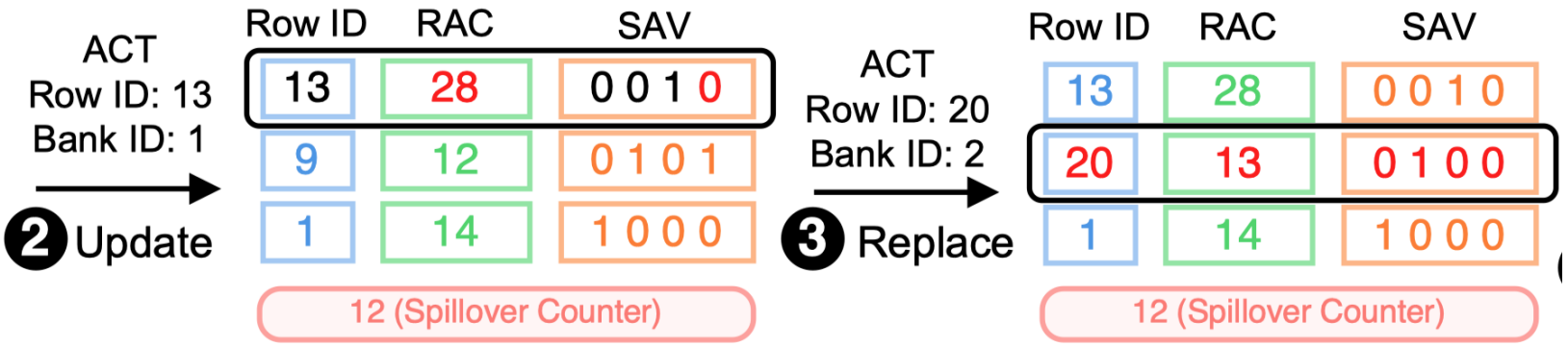
Increment only when a sibling is activated "again" (i.e., activate targets a set SAV bit)

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

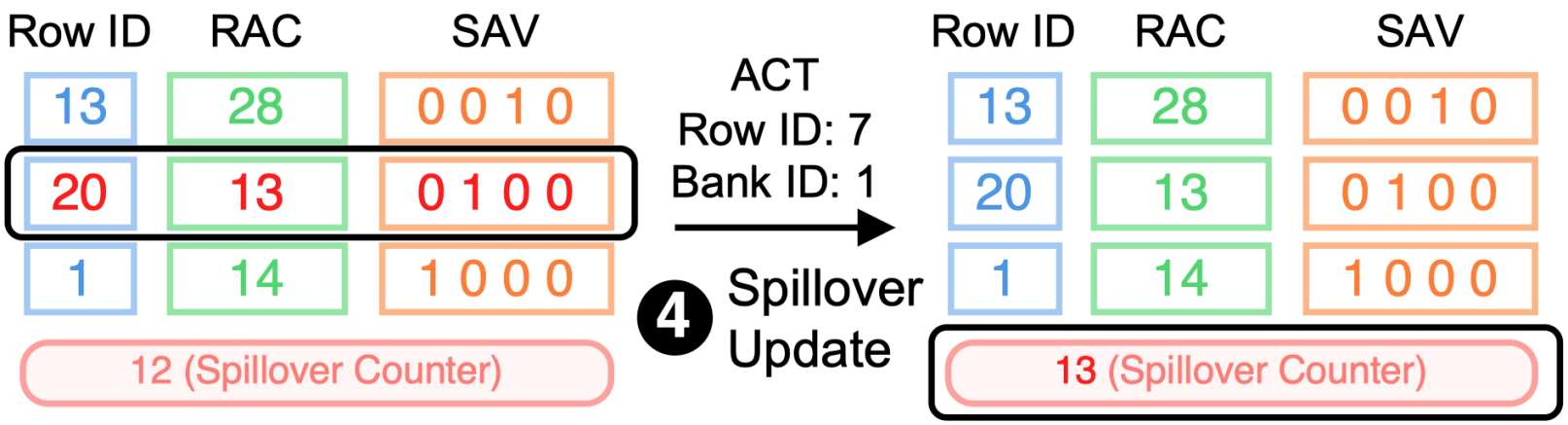
ABACuS: Operation (I)



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



ABACuS: Operation (II)



Area, Energy, and Power

Mitigation Mechanism	N_{RH} = 1000						
	SRAM KB	*CAM* KB	*Area* mm^2	*Area* % CPU	*Area* % DRAM	Access Energy (pJ)	Static Power (mW)
ABACuS	**10.63**	**8.30**	**0.04**	**0.02**	-	**25.98**	**12.22**
Row ID Table	-	5.64	0.01	<0.01	-	12.85	6.61
Row Activation Counter Table	-	2.66	0.02	<0.01	-	11.13	4.66
Sibling Activation Vector	10.63	-	0.01	<0.01	-	2.00	0.95
PARA [1]	-	-	-	<0.01	-	-	-
Graphene [102]	-	**286.51**	**0.81**	**0.35**	-	**873.38**	**187.98**
Hydra [106]	**61.56**	-	**0.10**	**0.04**	-	**43.07**	**24.17**
REGA [177]	-	-	-	-	**2.06**	-	-

Mitigation Mechanism	N_{RH} = 125						
	SRAM KB	*CAM* KB	*Area* mm^2	*Area* % CPU	*Area* % DRAM	Access Energy (pJ)	Static Power (mW)
ABACuS	**85.00**	**66.41**	**0.25**	**0.11**	-	**36.87**	**50.54**
Row ID Table	-	45.16	0.12	0.05	-	20.64	27.56
Row Activation Counter Table	-	21.25	0.06	0.03	-	11.66	15.53
Sibling Activation Vector	85.00	-	0.07	0.03	-	4.57	7.44
PARA [1]	-	-	-	<0.01	-	-	-
Graphene [102]	-	**2037.09**	**5.68**	**2.43**	-	**1042.49**	**1385.52**
Hydra [106]	**56.5**	-	**0.07**	**0.03**	-	**40.26**	**23.21**
REGA [177]	-	-	-	-	**2.06**	-	-

DRAM Address Mapping Function

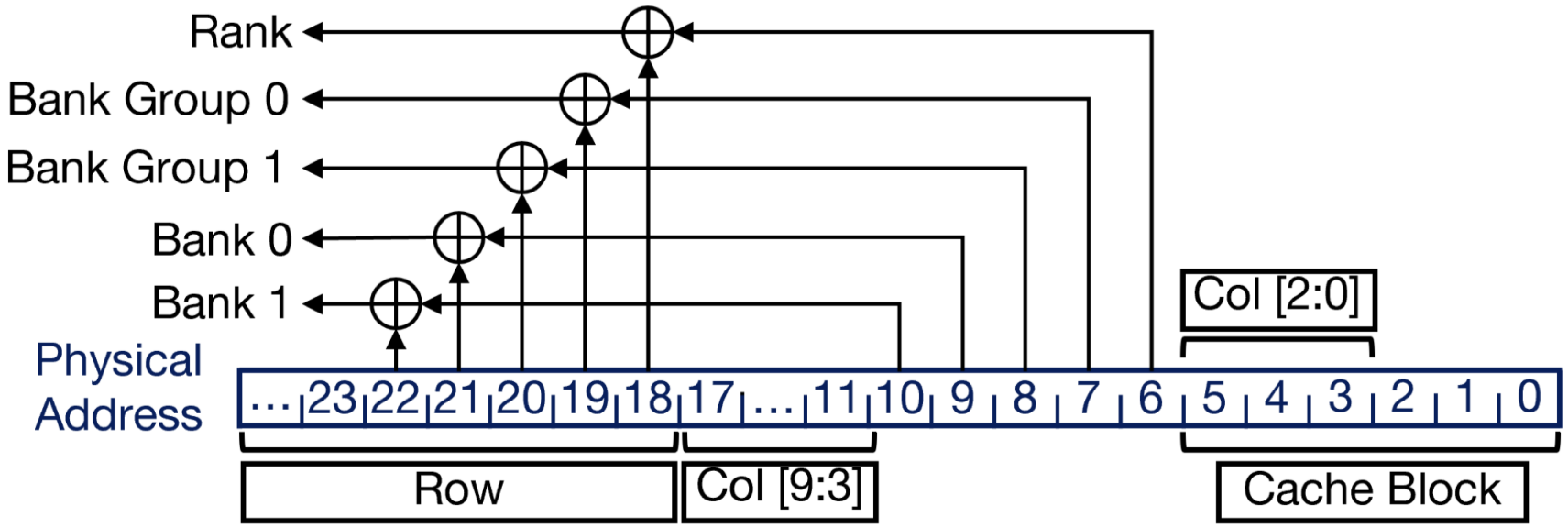
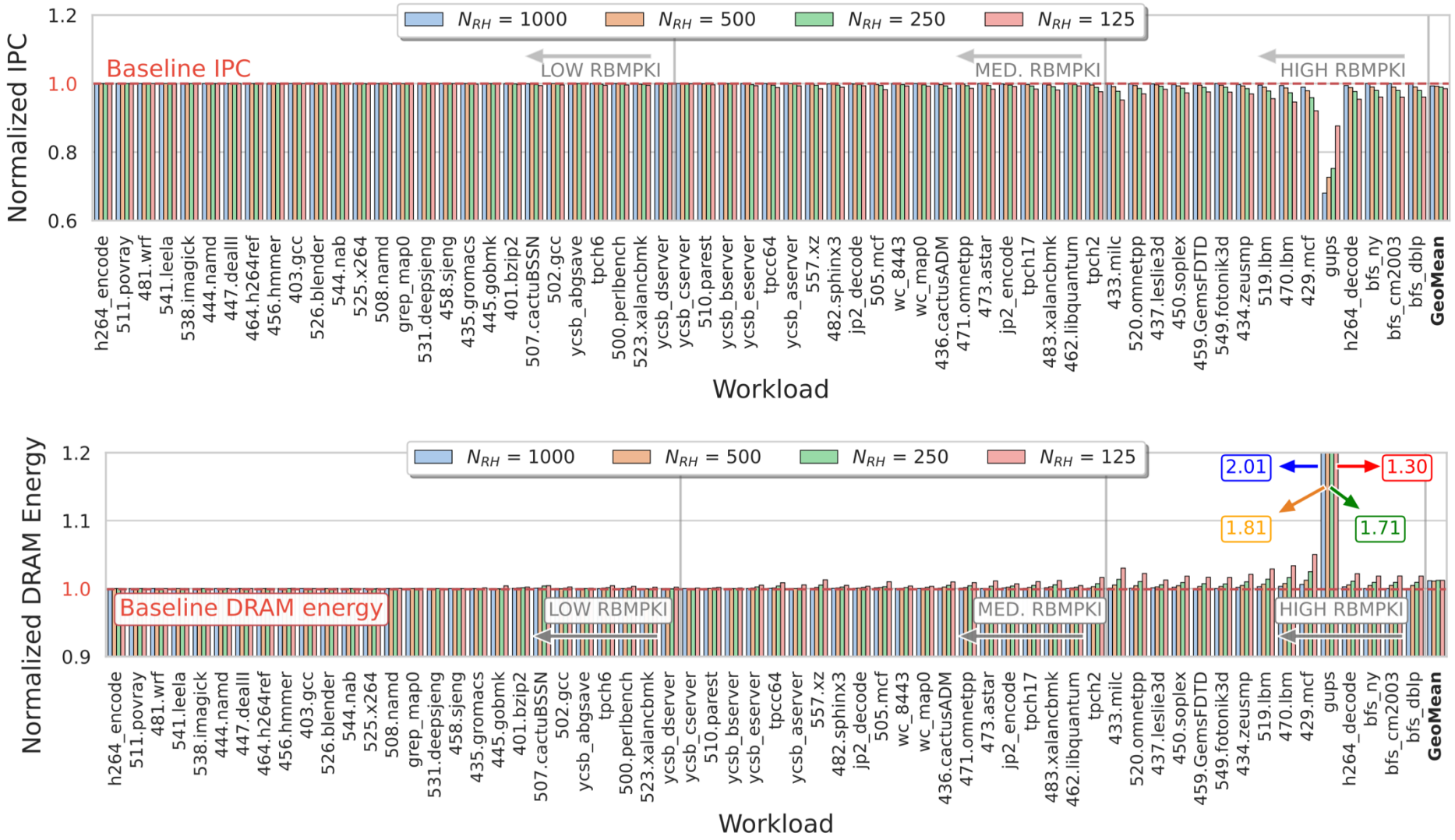


Figure 7: Simulated address mapping

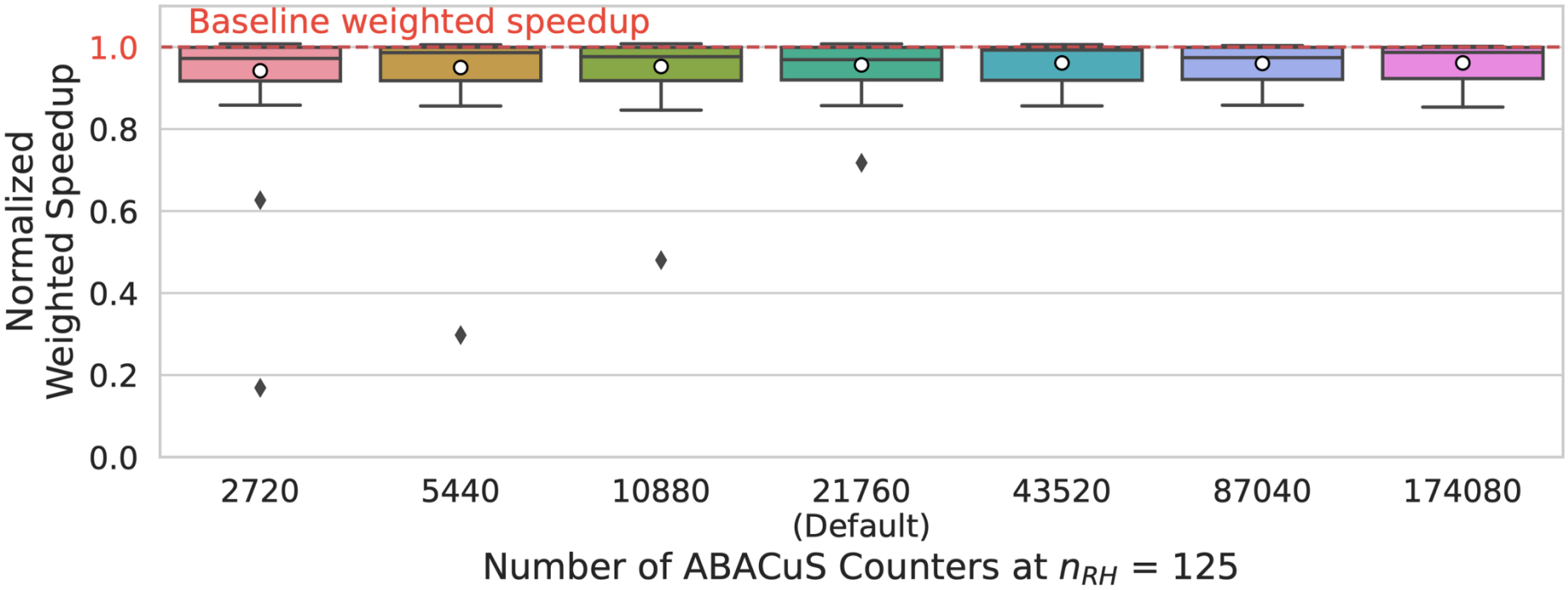
Key Configuration Parameters of RowHammer Mitigations

Mechanism	Configuration Parameter	Value			
All mechanisms	RowHammer Threshold	1000	500	250	125
Graphene	Number of table entries	2720	5440	10880	21760
	Threshold for aggressor tracking	500	250	125	63
	Reset window	64 ms			
Hydra	Row group size	128 rows			
	Row count table entry size	2B	1B		
	Row count cache size	4K entires per DRAM rank			
	Group count table threshold	400	200	100	50
	Tracking threshold	500	250	125	63
	Periodic reset	64 ms			
REGA	Row cycle time (t_{RC})	45.0 ns	62.5 ns	97.5 ns	167.5 ns
PARA	Probability threshold	0.034	0.067	0.129	0.241

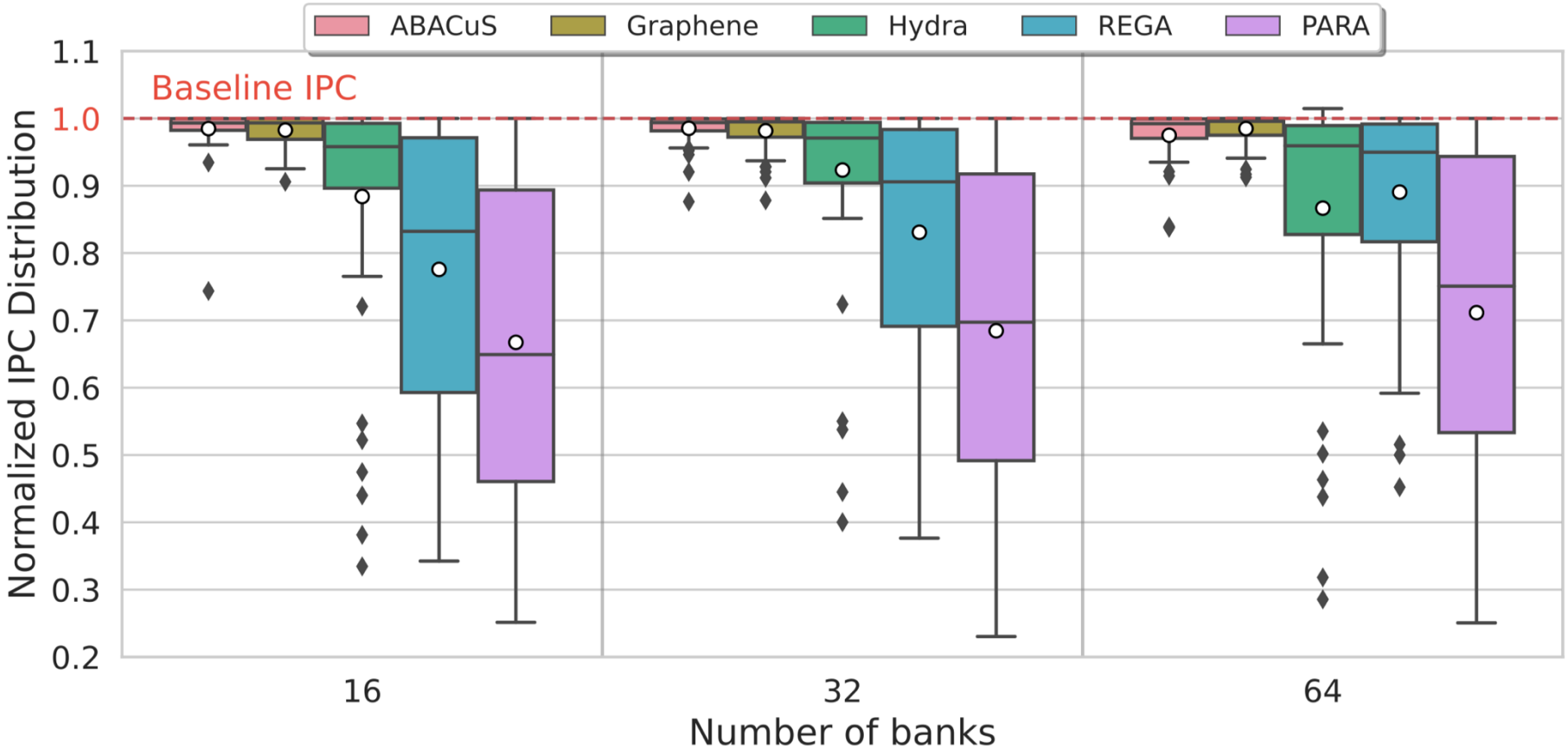
Single Core Performance and DRAM Energy



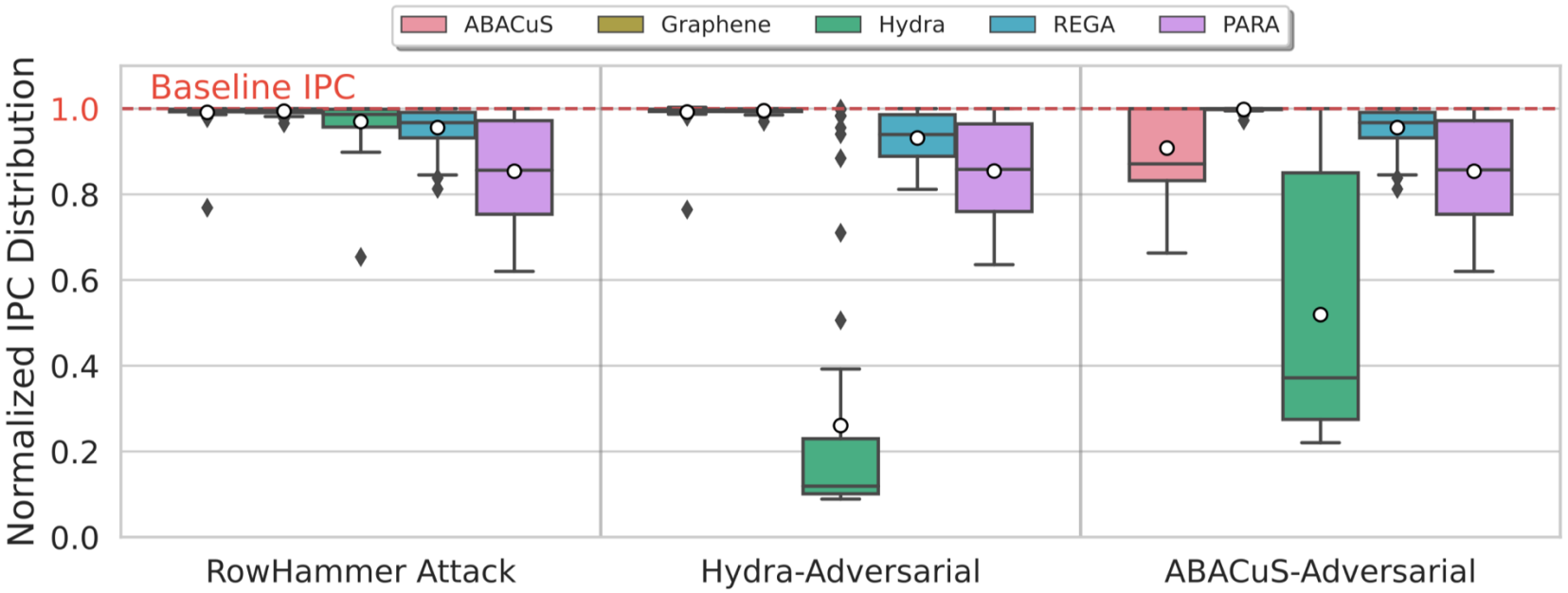
Sensitivity to Number of ABACuS Counters

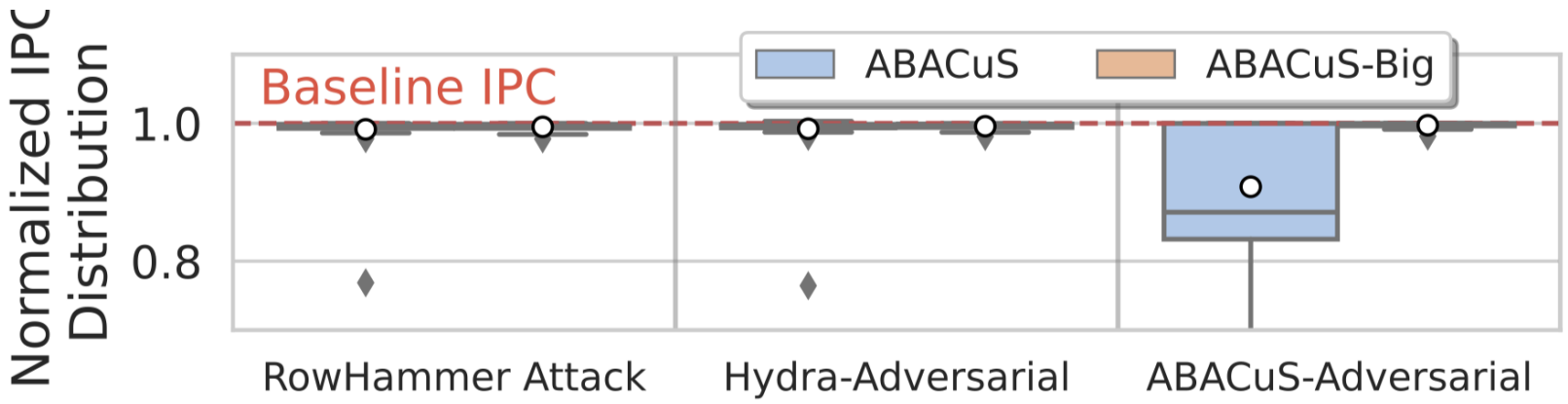


Sensitivity to Number of Banks

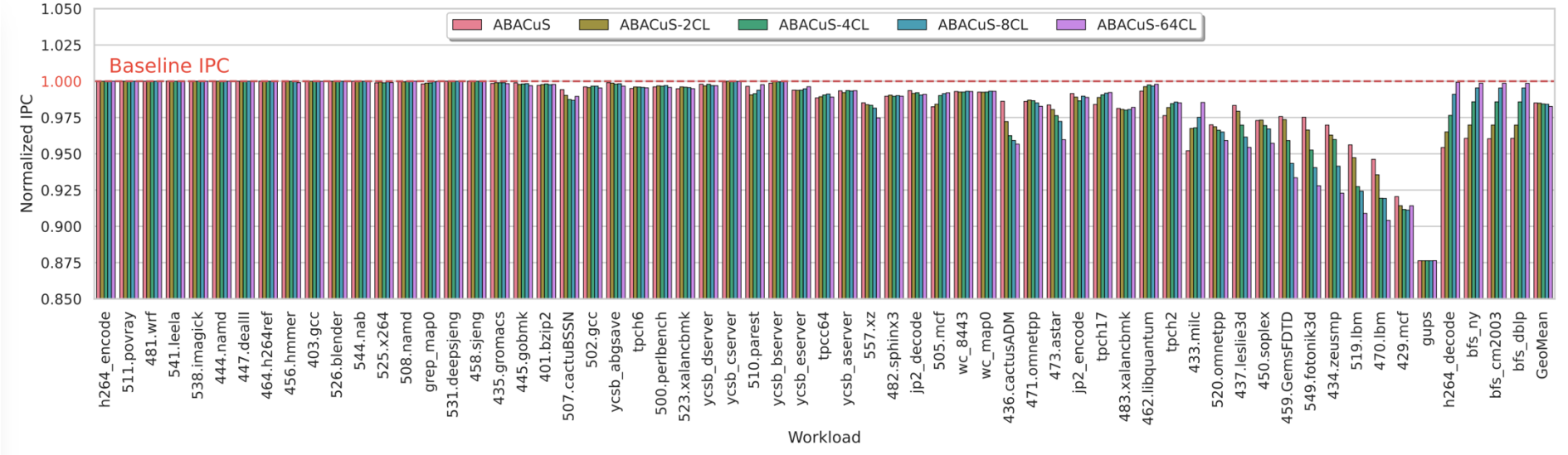


Performance Under Adversarial Workloads

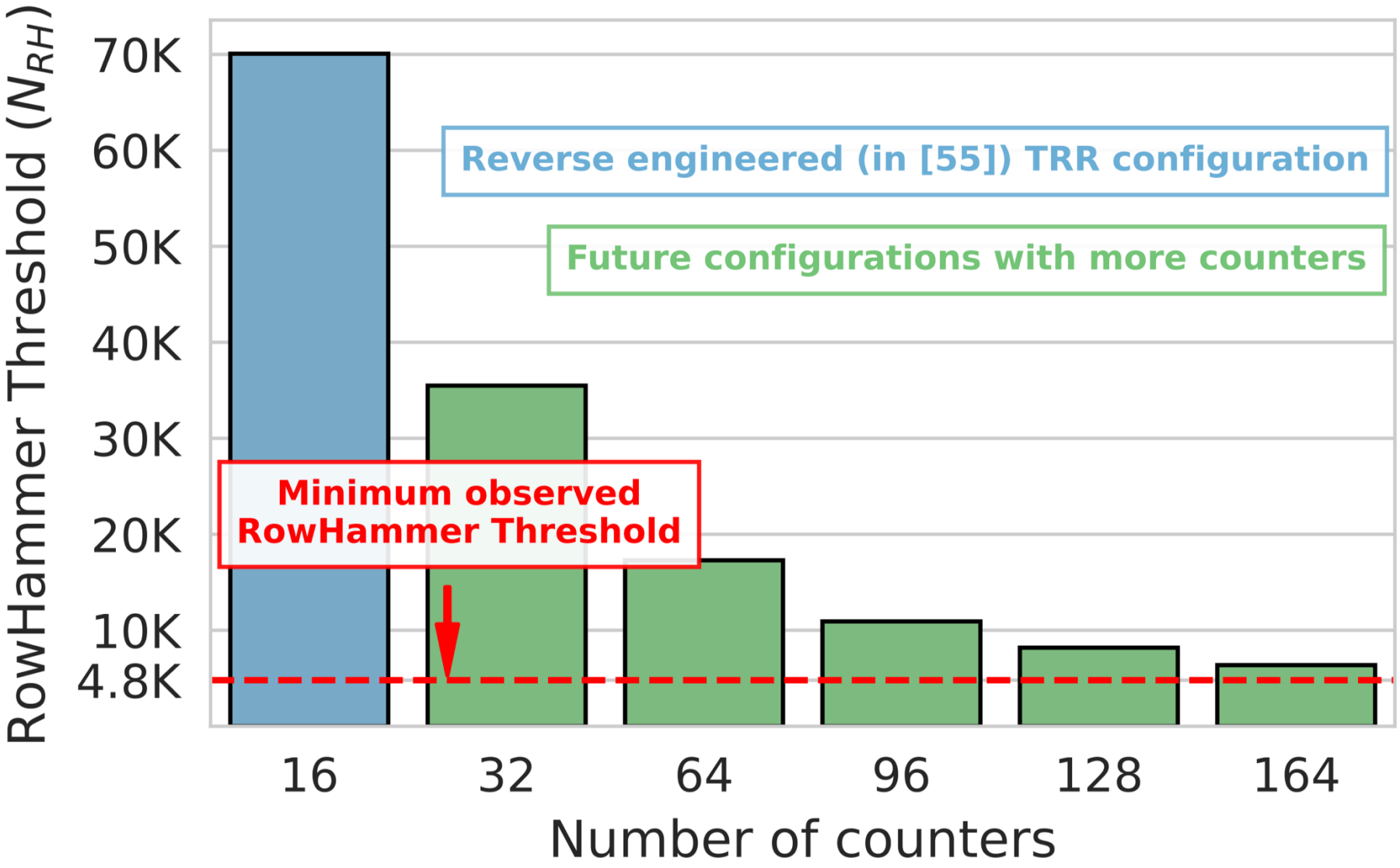




Sensitivity to Address Mapping



Limitations of Target Row Refresh

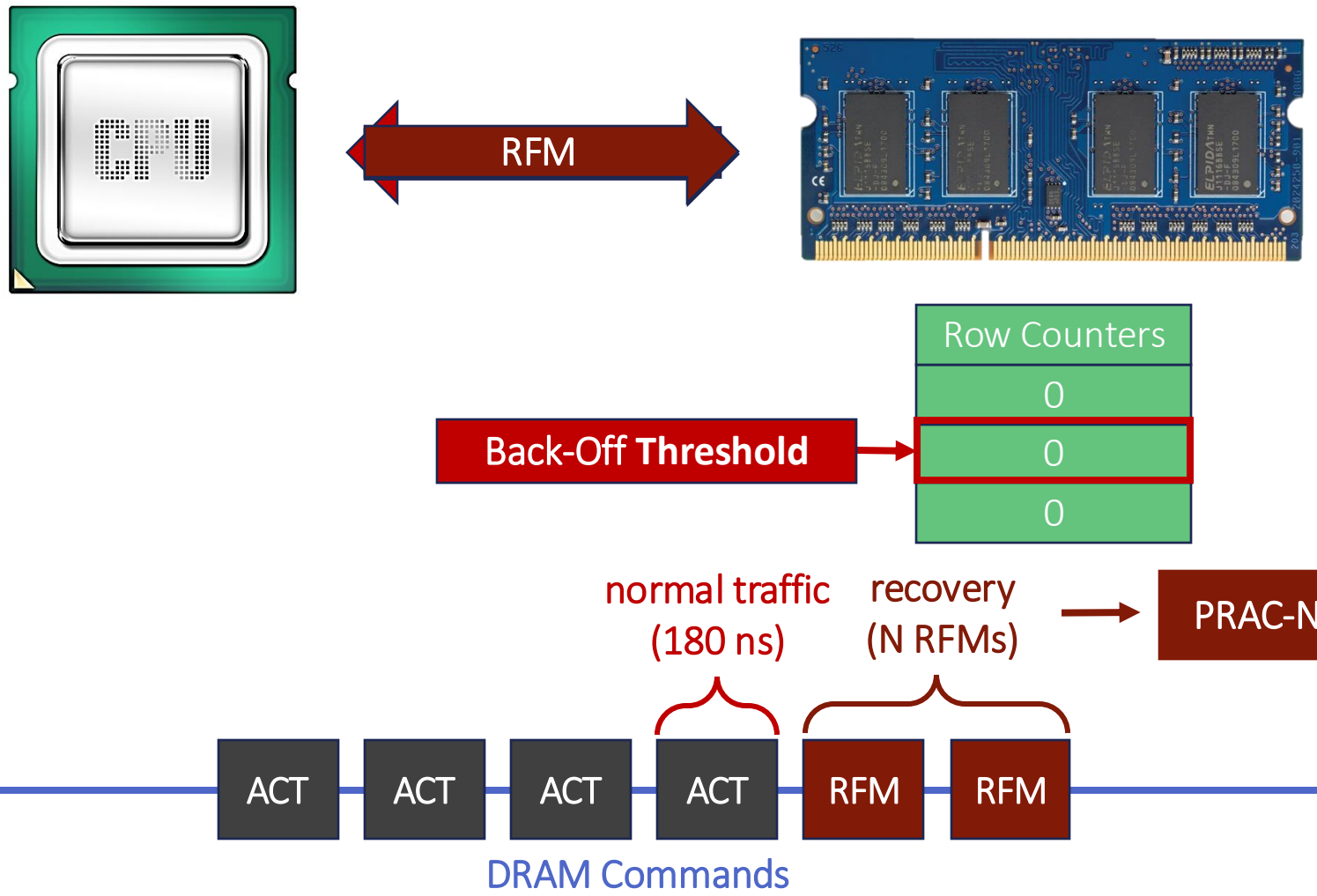


Applicability to Other Mitigations

- Many workloads access the same row address in different banks at around the same time

- This observation can be leveraged by many other RowHammer mitigations
  - Hydra, Graphene (what we showcase), Per Row Activation Counting (PRAC), ProTRR, …

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



[Canpolat+, DRAMSec 2024]

PRAC is NOT the Silver Bullet

Goal: Rigorously analyze and characterize the **security** and **performance** implications of the DDR5 standard **PRAC** mechanism

Mathematical analysis & extensive simulations show that PRAC:

- provides security as long as no bitflip occurs below **10 activations**
- has **non-negligible** performance (10%) and energy (18%) **overheads**
- **poorly scales** for future DRAM chips, leading to **significant overheads** on performance (49%) and energy (136%)
- allows memory performance attacks to hog significant amount of **DRAM throughput** (up to 79% throughput loss)

Future work: More research is needed to improve PRAC by

- reducing the overheads due to **increased DRAM timing parameters**
- solving the **exacerbated performance impact** as N_{RH} decreases
- stopping preventive refreshes from being **exploited** by memory performance attacks

[Canpolat+, DRAMSec 2024]