BlockHammer

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

Abdullah Giray Yağlıkçı

Minesh Patel Jeremie S. Kim Roknoddin Azizi

Ataberk Olgun Lois Orosa Hasan Hassan Jisung Park

Konstantinos Kanellopoulos Taha Shahroodi

Saugata Ghose* Onur Mutlu

SAFARI

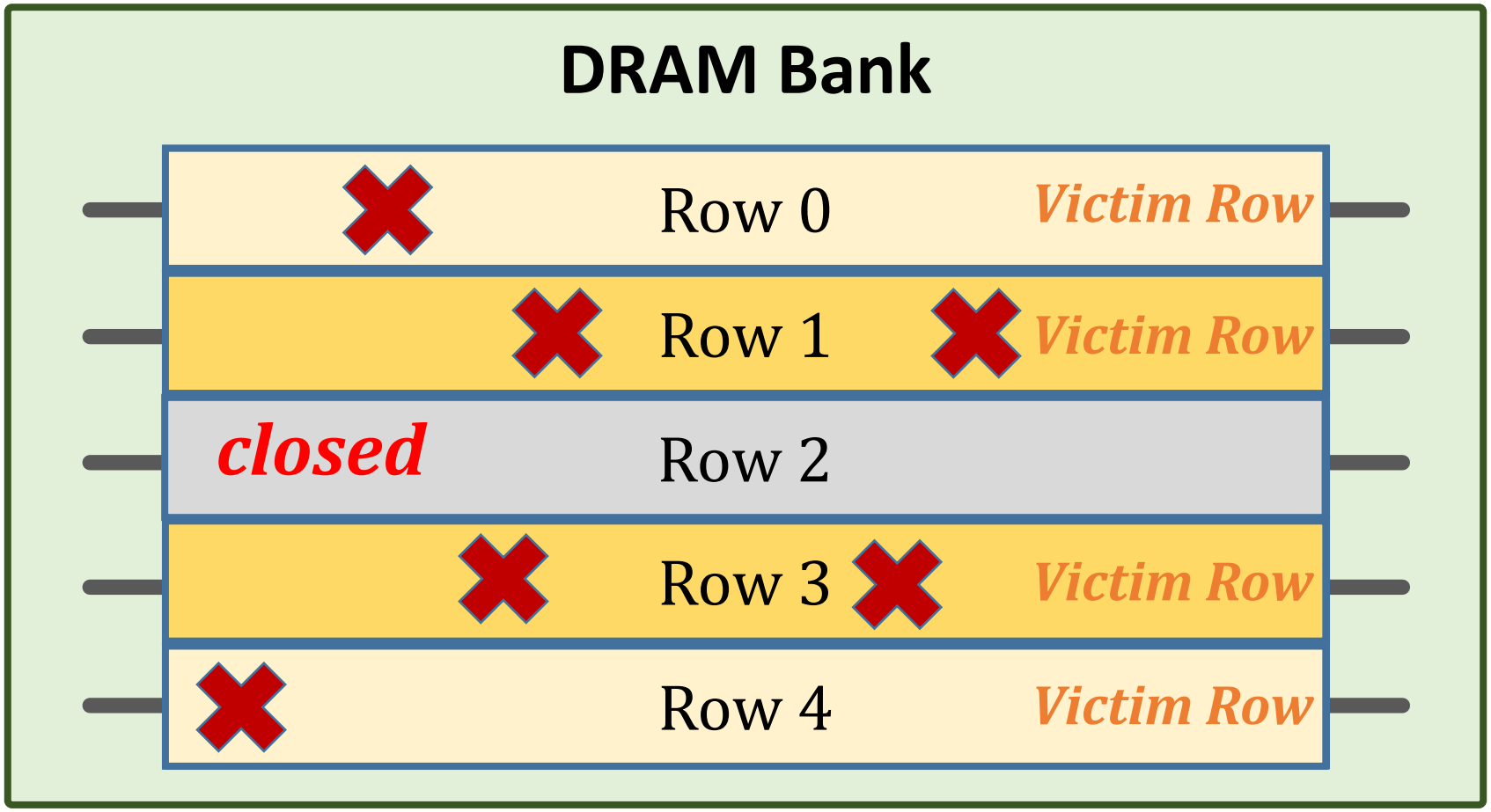
ETH zürich

* UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Executive Summary

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

The RowHammer Phenomenon

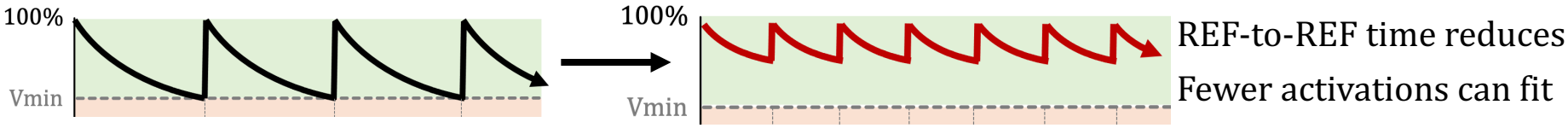


Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **RowHammer bit flips** in nearby cells

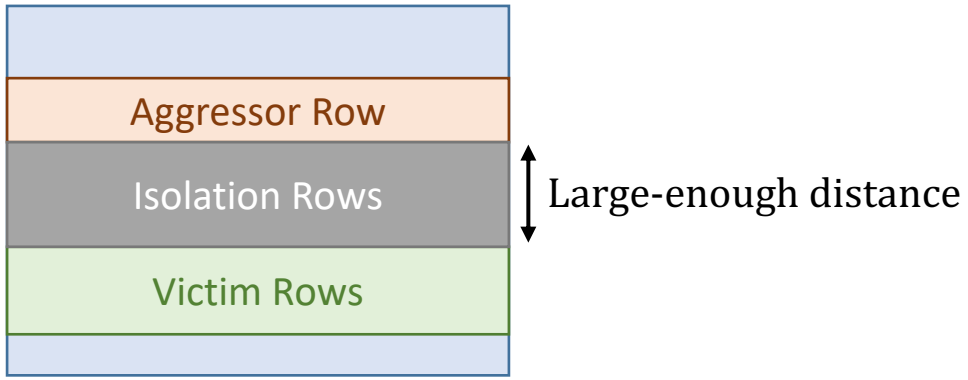
[Kim+ ISCA'20]

RowHammer Mitigation Approaches

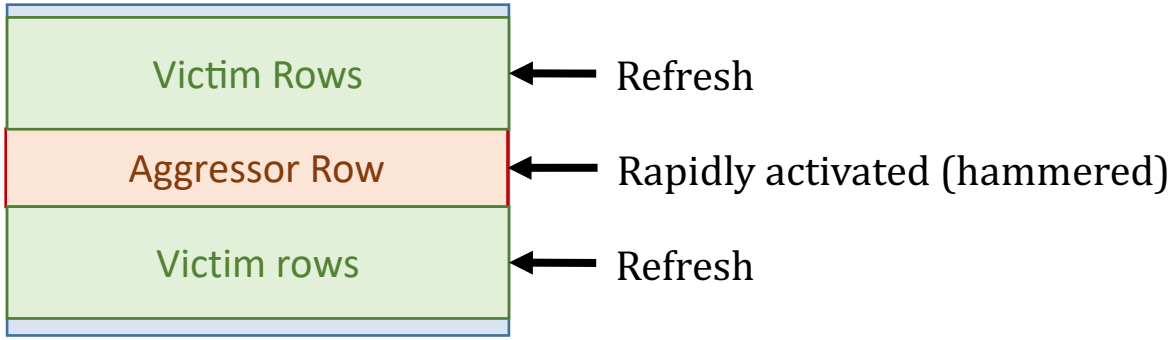
- Increased refresh rate



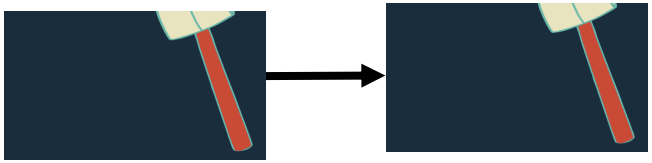
- Physical isolation



- Reactive refresh



- Proactive throttling



Fewer activations can be performed

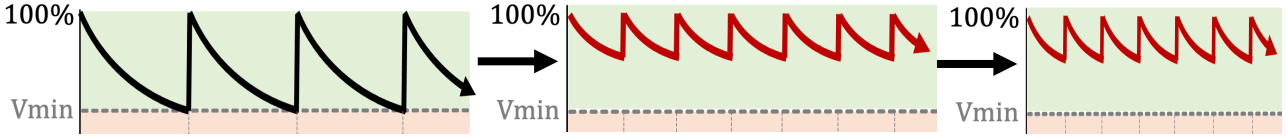
Two Key Challenges

1 **Scalability**
with worsening RowHammer vulnerability

2 **Compatibility**
with commodity DRAM chips

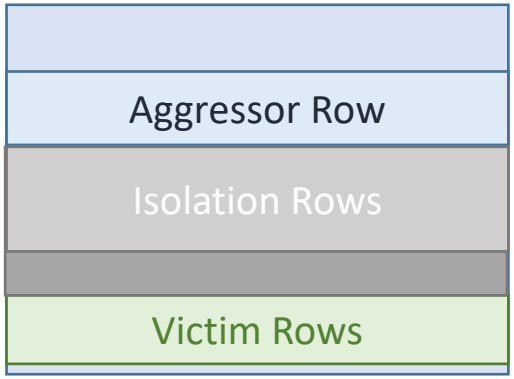
Mitigation Approaches
with Worsening RowHammer Vulnerability

- Increased refresh rate



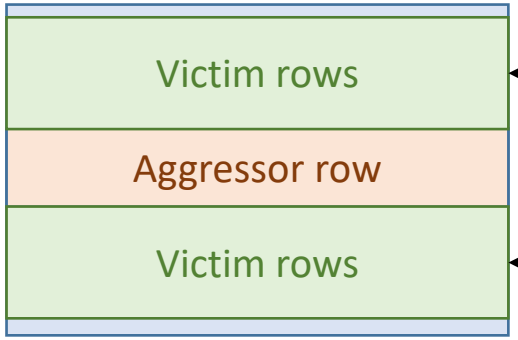
REF-to-REF time further reduces
Even fewer activations can fit

- Physical isolation



Larger distance
more isolation rows

- Reactive refresh



Refresh more frequently
Refresh more rows

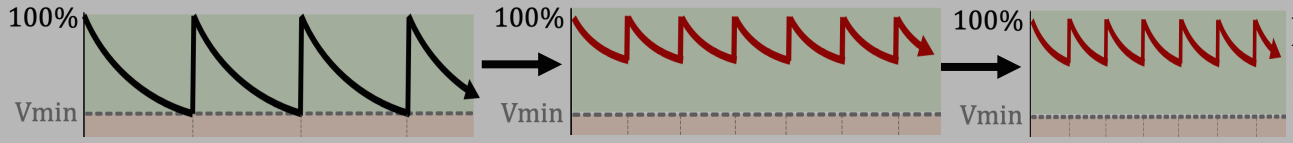
Refresh more frequently
Refresh more rows

- Proactive throttling

More aggressively throttles row activations

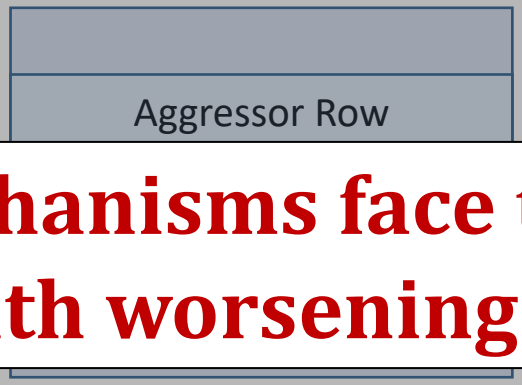
Mitigation Approaches
with Worsening RowHammer Vulnerability

- Increased refresh rate



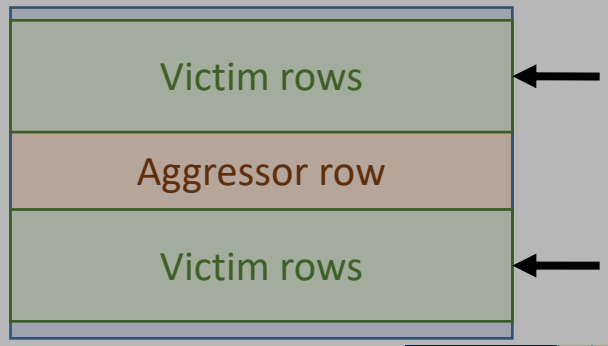
REF-to-REF time further reduces
Even fewer activations can fit

- Physical isolation



Mitigation mechanisms face the challenge of scalability with worsening RowHammer

- Reactive refresh



Refresh more frequently
Refresh more rows

Refresh more frequently
Refresh more rows

- Proactive throttling

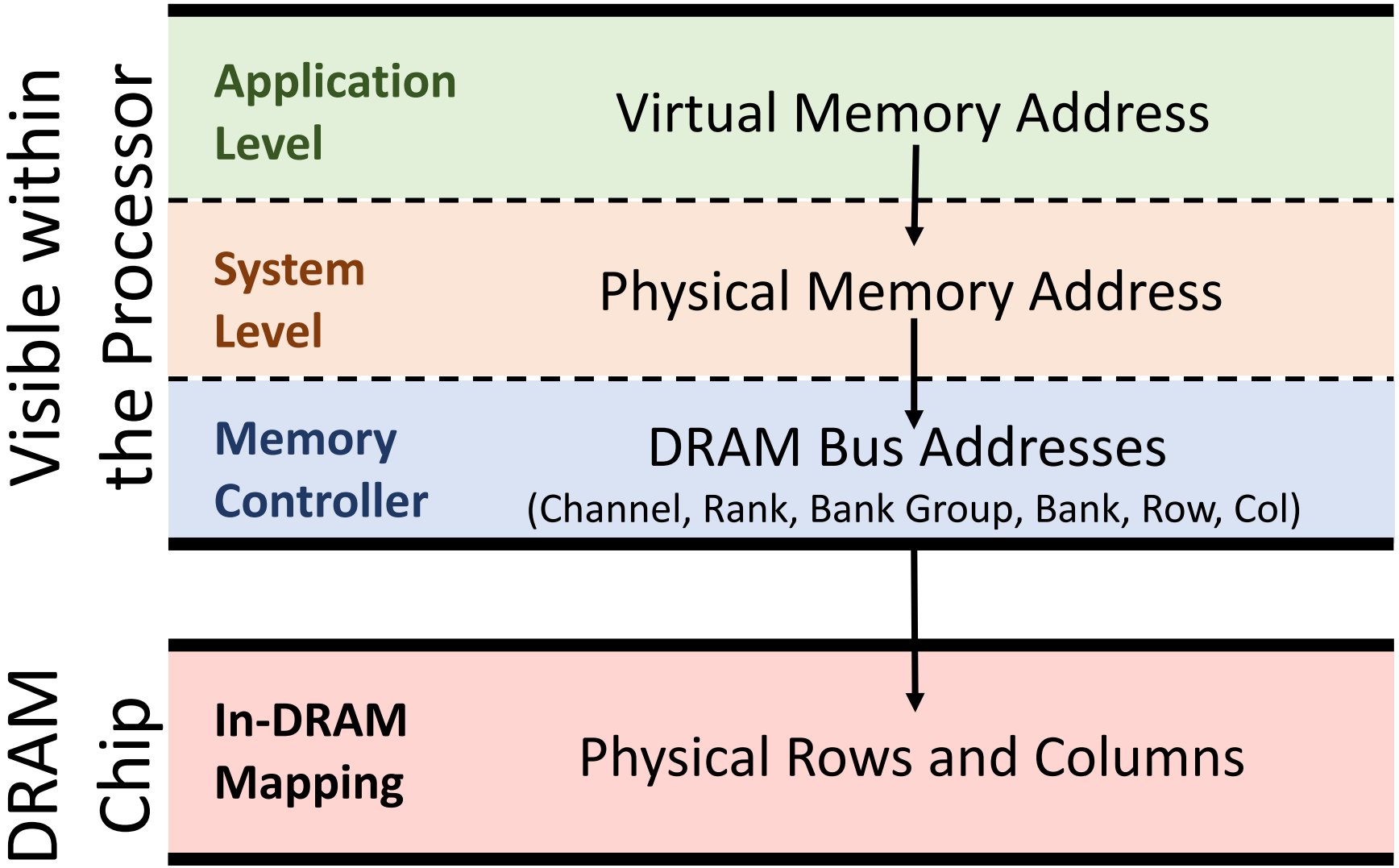
More aggressively throttles row activations

Two Key Challenges

1 **Scalability**
with worsening RowHammer vulnerability

2 **Compatibility**
with commodity DRAM chips

Compatibility
with Commodity DRAM Chips



Compatibility
with Commodity DRAM Chips

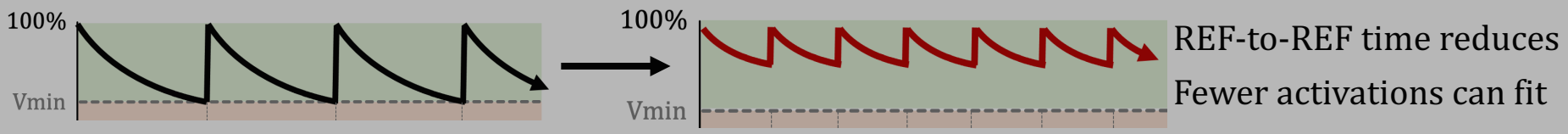
Vendors apply in-DRAM mapping for two reasons:

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

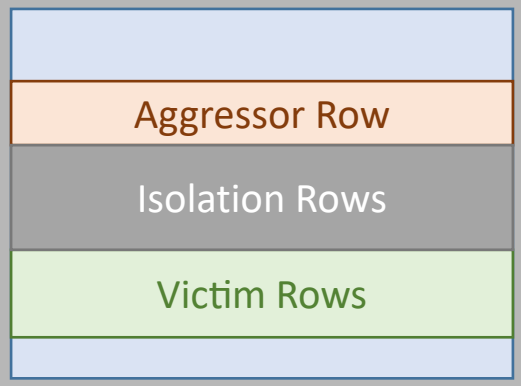
In-DRAM mapping is proprietary information

RowHammer Mitigation Approaches

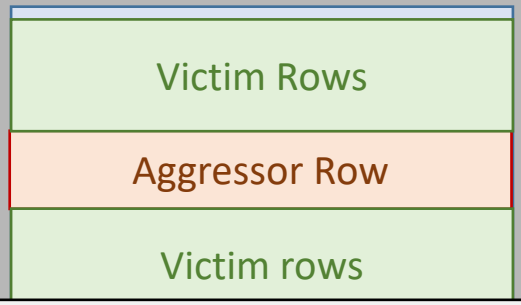
- Increased refresh rate



- Physical isolation



- Reactive refresh



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

Our Goal

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

BlockHammer
Key Idea

Selectively throttle memory accesses
that may cause RowHammer bit-flips

BlockHammer
Overview of Approach

RowBlocker

Tracks row activation rates using area-efficient Bloom filters

Blacklists rows that are activated at a high rate

Throttles activations targeting a blacklisted row

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

AttackThrottler

Identifies threads that perform a RowHammer attack

Reduces memory bandwidth usage of identified threads

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

Evaluation
Performance and DRAM Energy

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

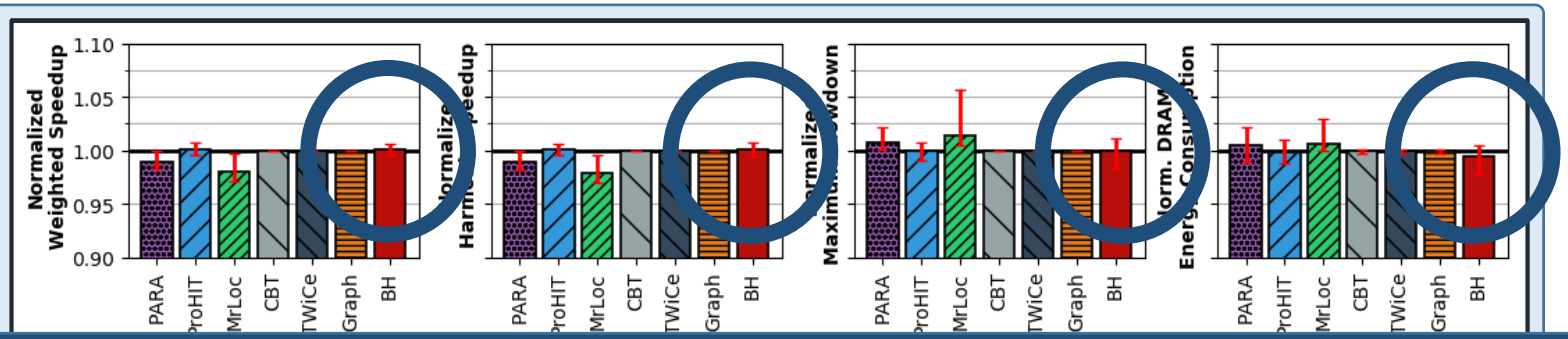
Processor	3.2 GHz, {1,8} core, 4-wide issue, 128-entry instr. window
LLC	64-byte cacheline, 8-way set-associative, {2,16} MB
Memory scheduler	FR-FCFS
Address mapping	Minimalistic Open Pages
DRAM	DDR4 1 channel, 1 rank, 4 bank group, 4 banks per bank group
RowHammer Threshold	32K

- Single-Core Benign Workloads:
  - 22 SPEC CPU 2006
  - 4 YCSB Disk I/O
  - 2 Network Accelerator Traces
  - 2 Bulk Data Copy with Non-Temporal Hint (movnti)
- Randomly Chosen Multiprogrammed Workloads:
  - 125 workloads containing **8 benign applications**
  - 125 workloads containing **7 benign applications** and **1 RowHammer attack thread**

Evaluation
Performance and DRAM Energy

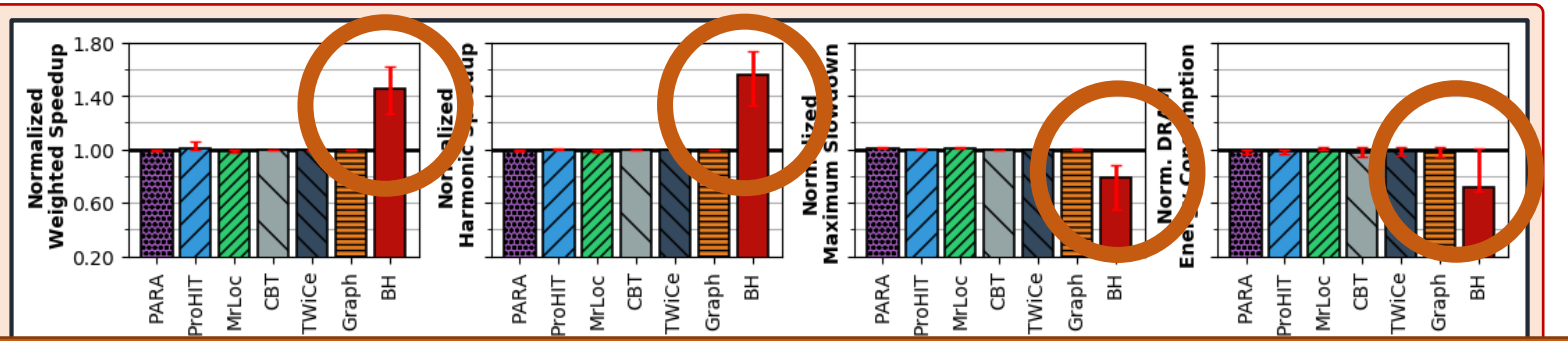
- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

No RowHammer Attack



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

RowHammer Attack Present

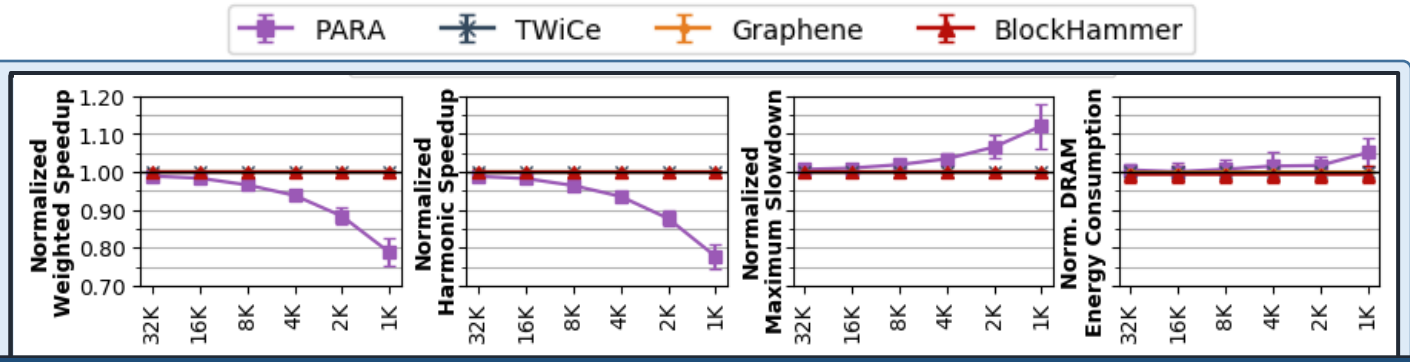


BlockHammer **significantly increases** benign application performance (by 45% on average) and **reduces** DRAM energy consumption (by 29% on average)

Evaluation
Scaling with RowHammer Vulnerability

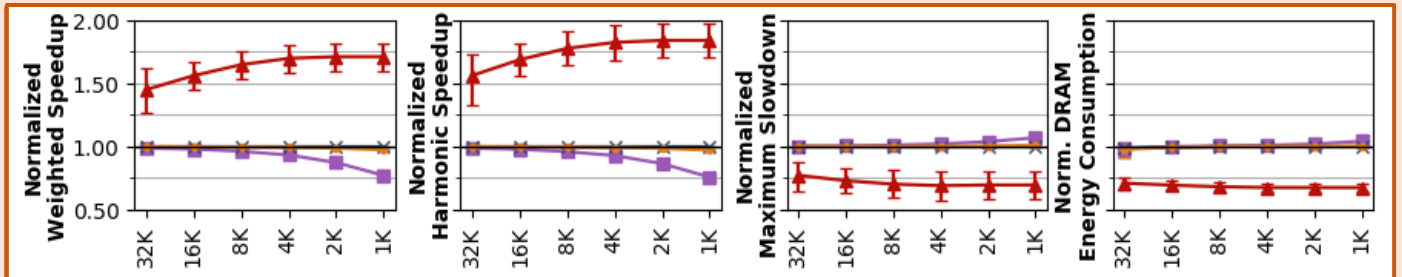
- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

No RowHammer Attack



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

RowHammer Attack Present



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

Evaluation

BlockHammer's Hardware Complexity

	Mitigation Mechanism	SRAM KB	CAM KB	Area mm²	%CPU	Access Energy pJ	Static Power mW
N_{RH}=32K	BlockHammer	51.48	1.73	0.14	0.06	20.30	22.27
	PARA [73]	-	-	<0.01	-	-	-
	ProHIT [137]	-	0.22	<0.01	<0.01	3.67	0.20
	MRLoc [161]	-	0.47	<0.01	<0.01	4.44	0.21
	CBT [132]	16.00	8.50	0.20	0.08	9.13	35.55
	TWiCe [84]	23.10	14.02	0.15	0.06	7.99	21.28
	Graphene [113]	-	5.22	0.04	0.02	40.67	3.11
N_{RH}=1K	BlockHammer	441.33	55.58	1.57	0.64	99.64	220.99
	PARA [73]	-	-	<0.01	-	-	-
	ProHIT [137]	x	x	x	x	x	x
	MRLoc [161]	x	x	x	x	x	x
	CBT [132]	512.00	272.00	3.95	1.60	127.93	535.50
	TWiCe [84]	738.32	448.27	5.17	2.10	124.79	631.98
	Graphene [113]	-	166.03	1.14	0.46	917.55	93.96

10x 5x 10x

20x 35x 23x

23x

15x 30x 30x

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

More in the Paper

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

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

BlockHammer

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

Abdullah Giray Yağlıkçı

Minesh Patel Jeremie S. Kim Roknoddin Azizi

Ataberk Olgun Lois Orosa Hasan Hassan Jisung Park

Konstantinos Kanellopoulos Taha Shahroodi

Saugata Ghose* Onur Mutlu

SAFARI

ETH zürich

* UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

BlockHammer

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

Backup Slides

Evaluation
Performance and DRAM Energy

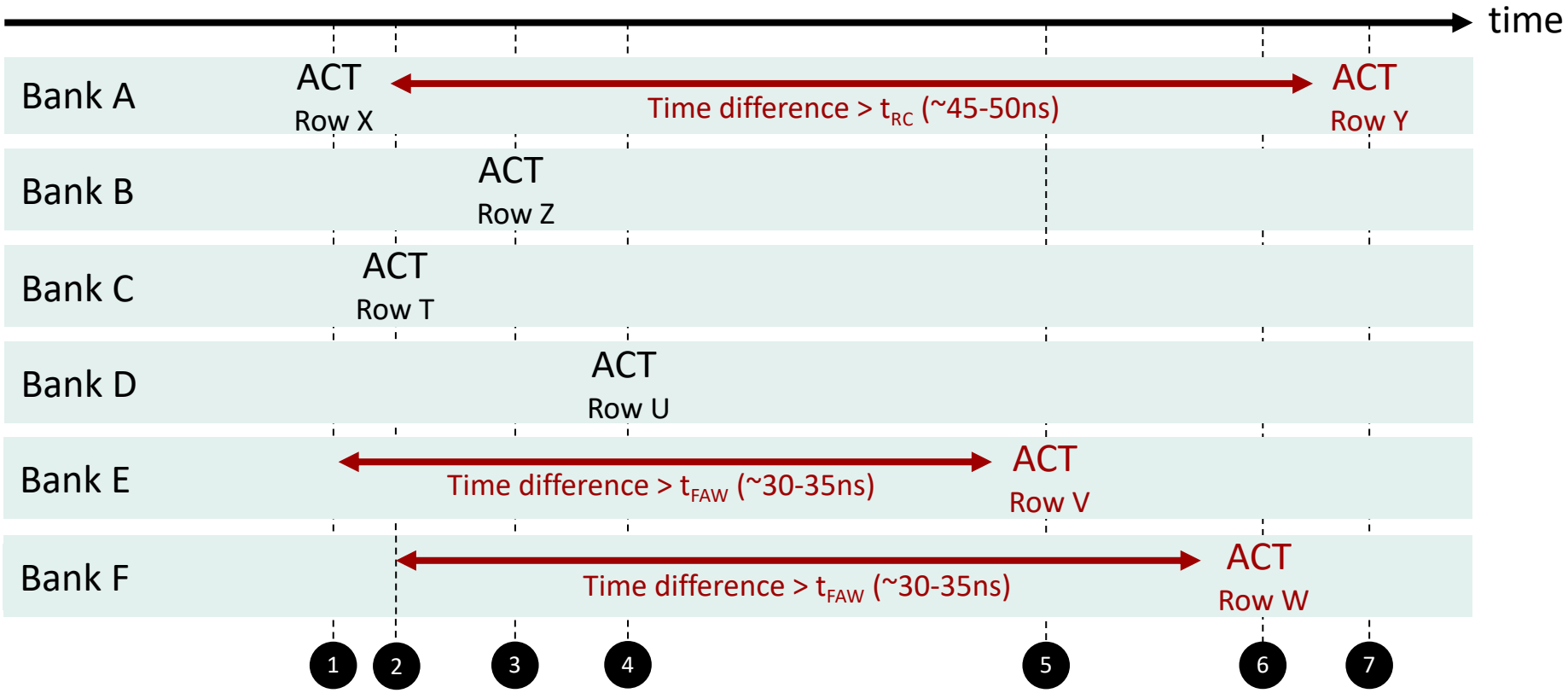
- Cycle-level simulations using **Ramulator** and **DRAMPower**
- System Configuration:

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

- Single-Core Benign Workloads:
  - 22 SPEC CPU 2006
  - 4 YCSB Disk I/O
  - 2 Network Accelerator Traces
  - 2 Bulk Data Copy with Non-Temporal Hint (movnti)
- Randomly Chosen Multiprogrammed Workloads:
  - 125 workloads containing **8 benign applications**
  - 125 workloads containing **7 benign applications** and **1 RowHammer attack thread**

Timing Constraints for DRAM Row Activations

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



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

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

Scalability
with Worsening RowHammer Vulnerability

- DRAM chips are more vulnerable to RowHammer today
- RowHammer bit-flips occur at much lower activation counts (more than an order of magnitude decrease):
  - 139.2K [Y. Kim+, ISCA 2014]
  - 9.6K [J. S. Kim+, ISCA 2020]

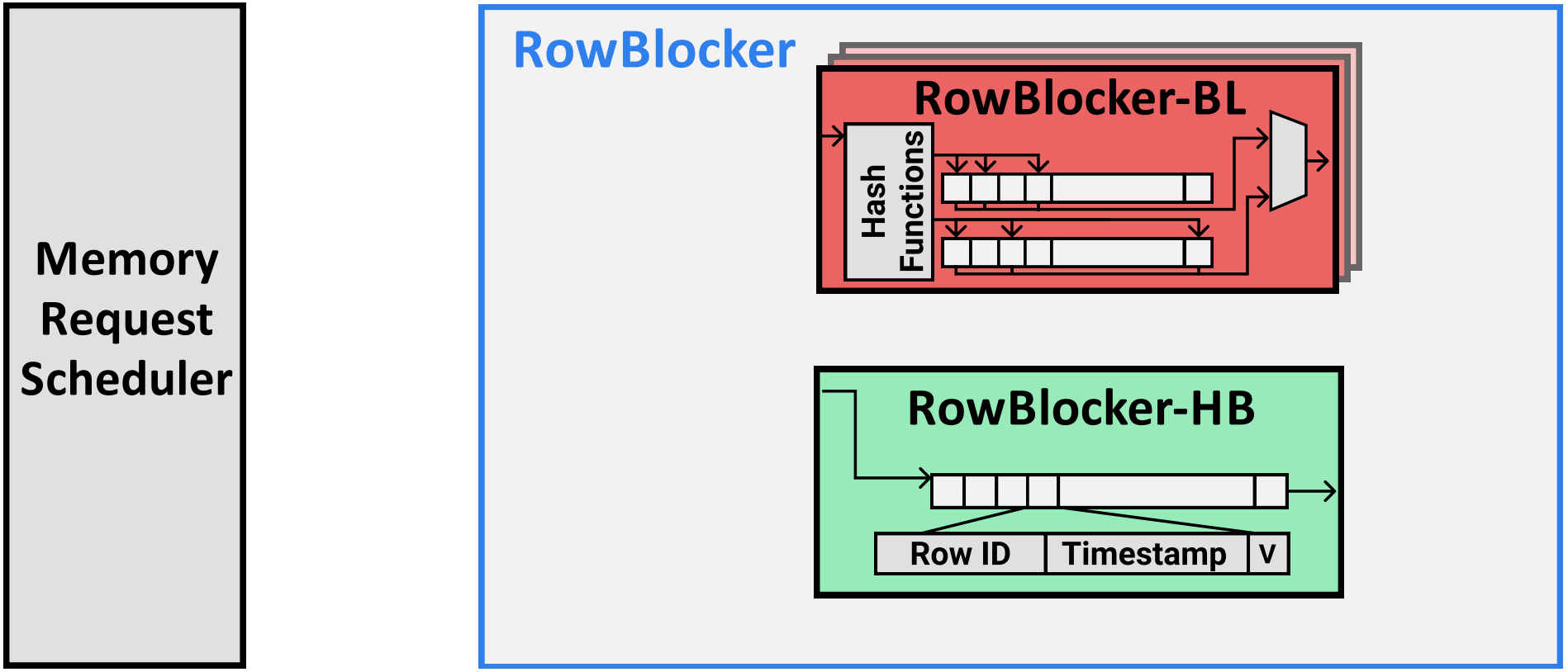
- RowHammer blast radius has increased by 33%:
  - 9 rows [Y. Kim+, ISCA 2014]
  - 12 rows [J. S. Kim+, ISCA 2020]

- In-DRAM mitigation mechanisms are ineffective [Frigo+, S&P 2020]

RowHammer is a more serious problem than ever

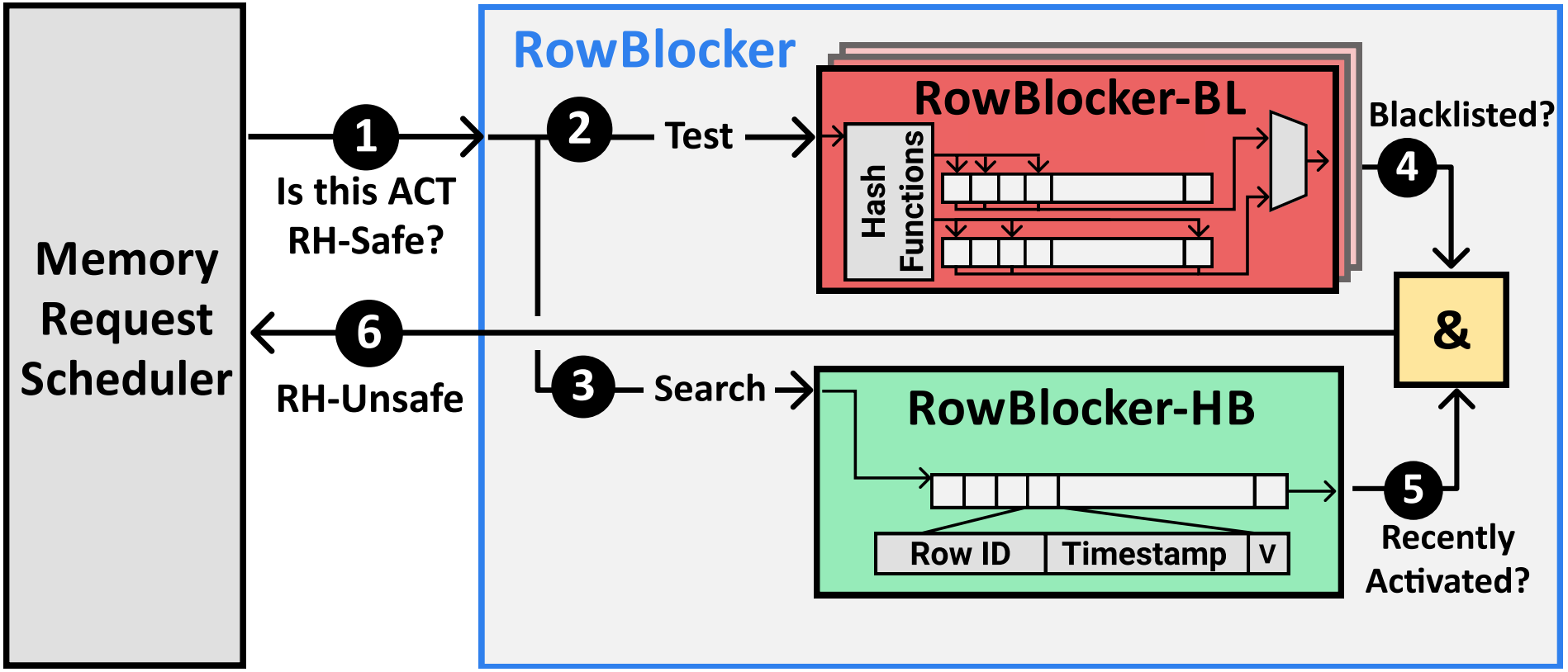
RowBlocker

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



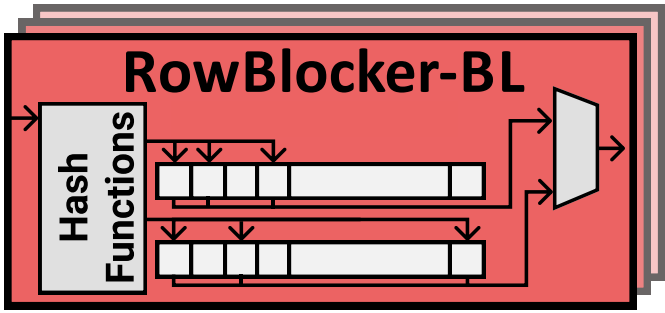
RowBlocker

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



RowBlocker-BL
Blacklisting Logic

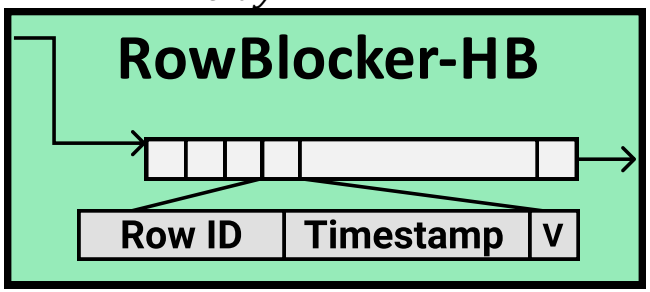
- Blacklists a row when the row's activation count in a time window exceeds a threshold



- Employs two counting Bloom filters for area-efficient activation rate tracking

RowBlocker-HB
Delaying Row Activations

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



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

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

BlockHammer
Overview of Approach

RowBlocker

Tracks row activation rates using area-efficient Bloom filters

Blacklists rows that are activated at a high rate

Throttles activations targeting a blacklisted row

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

AttackThrottler

Identifies threads that perform a RowHammer attack

Reduces memory bandwidth usage of identified threads

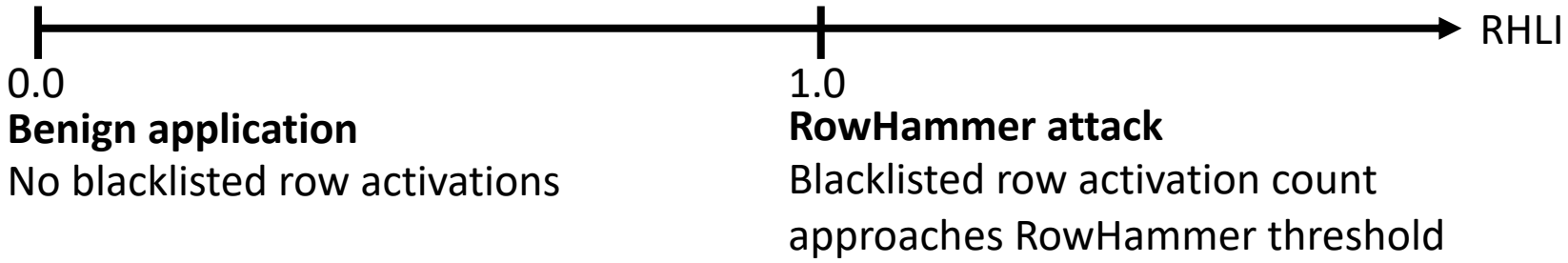
Greatly reduces the **performance degradation**
and **energy wastage** a RowHammer attack inflicts on a system

AttackThrottler

- Tackles a RowHammer attack's **performance degradation** and **energy wastage** on a system

- A RowHammer attack intrinsically keeps activating blacklisted rows

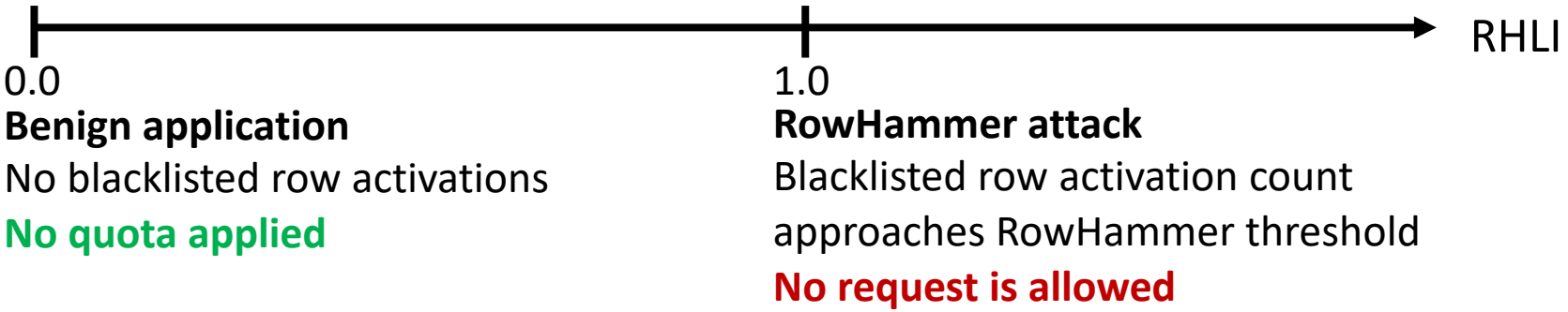
- **RowHammer Likelihood Index (RHLI):** Number of activations that target blacklisted rows (normalized to maximum possible activation count)



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

AttackThrottler

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



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

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

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

BlockHammer Hardware Complexity

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

RowHammer Characteristics

- **RowHammer Threshold (N_{RH}):**
  The minimum row activation count in a refresh window to induce a RowHammer bit-flip.

- **Blast Radius (r_{Blast}):**
  The maximum physical distance from the aggressor row at which RowHammer bit-flips can be observed.

- **Blast Impact Factor (c_i):**
  Set of coefficients that scale a RowHammer attacks impact on victim rows based on their physical distance to the aggressor row.

Many-Sided Attacks

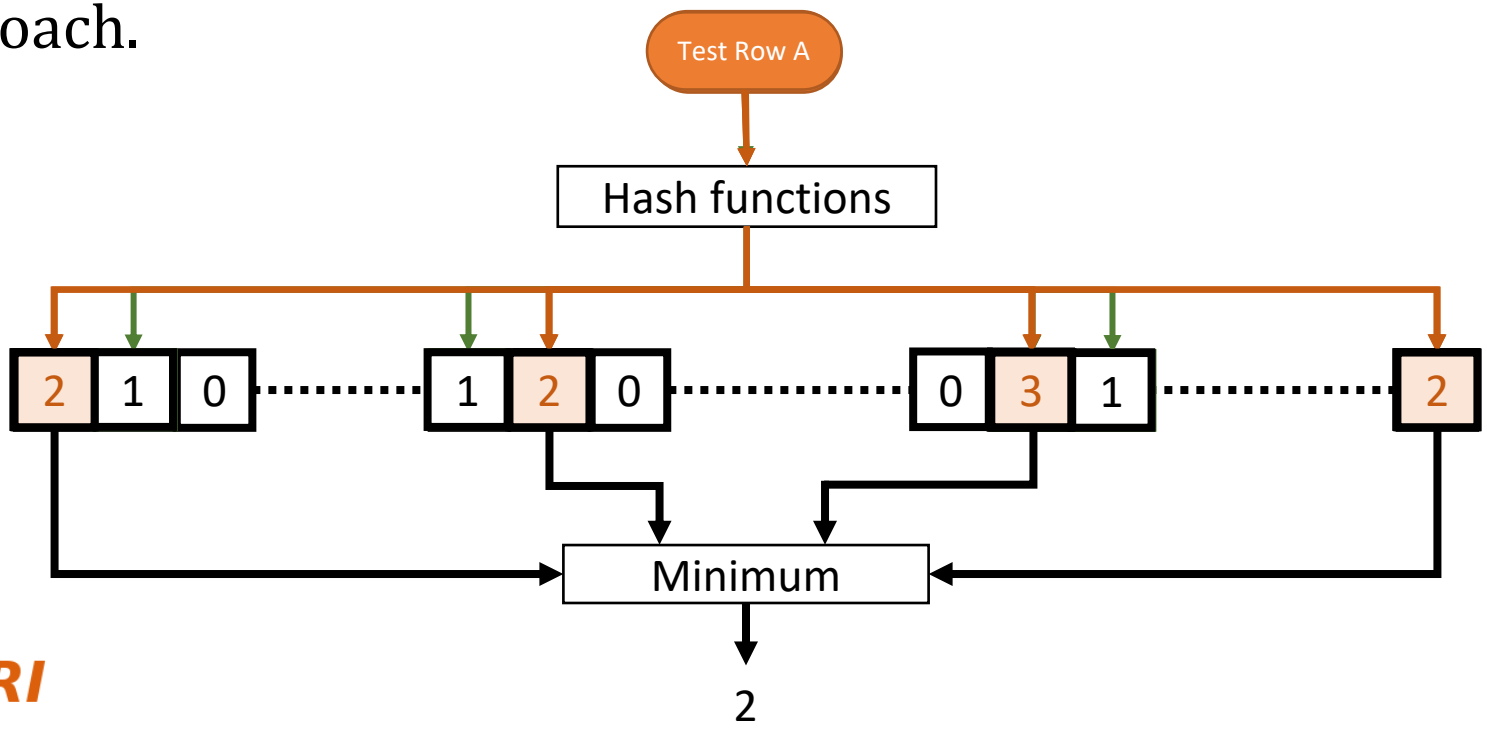
- N_{RH} : RowHammer threshold for single-sided attack.
- N_{RH}^* : Maximum activation count that BlockHammer allows in a refresh window.
- r_{Blast} : Blast radius
- c_i : Blast impact factor
- We configure N_{RH}^* such that hammering all rows N_{RH}^* times does not cause bit-flips.

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

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

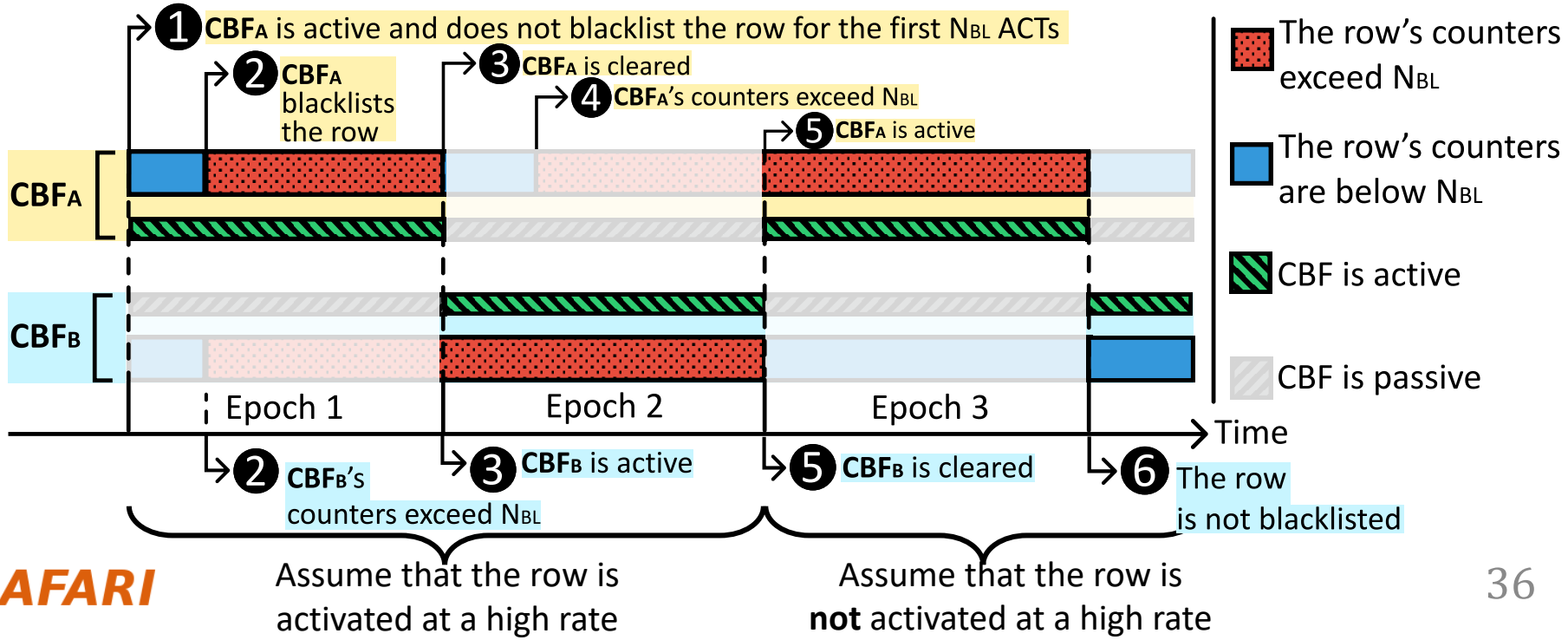
Counting Bloom Filters

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



RowBlocker-BL Blacklisting Logic

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



Evaluation
BlockHammer's Hardware Complexity

When configured for a RowHammer threshold of 32K:

- 0.06% chip area overhead[1]
- 20pJ access energy and 22mW static power consumption

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

For a RowHammer threshold of 1K:
BlockHammer's area, access energy, static power are 1.25X – 2.5X smaller than two major prior mechanisms.

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

[1]Assuming a high-end 28-core Intel Xeon processor system with 4-channel single-rank DDR4 DIMMs

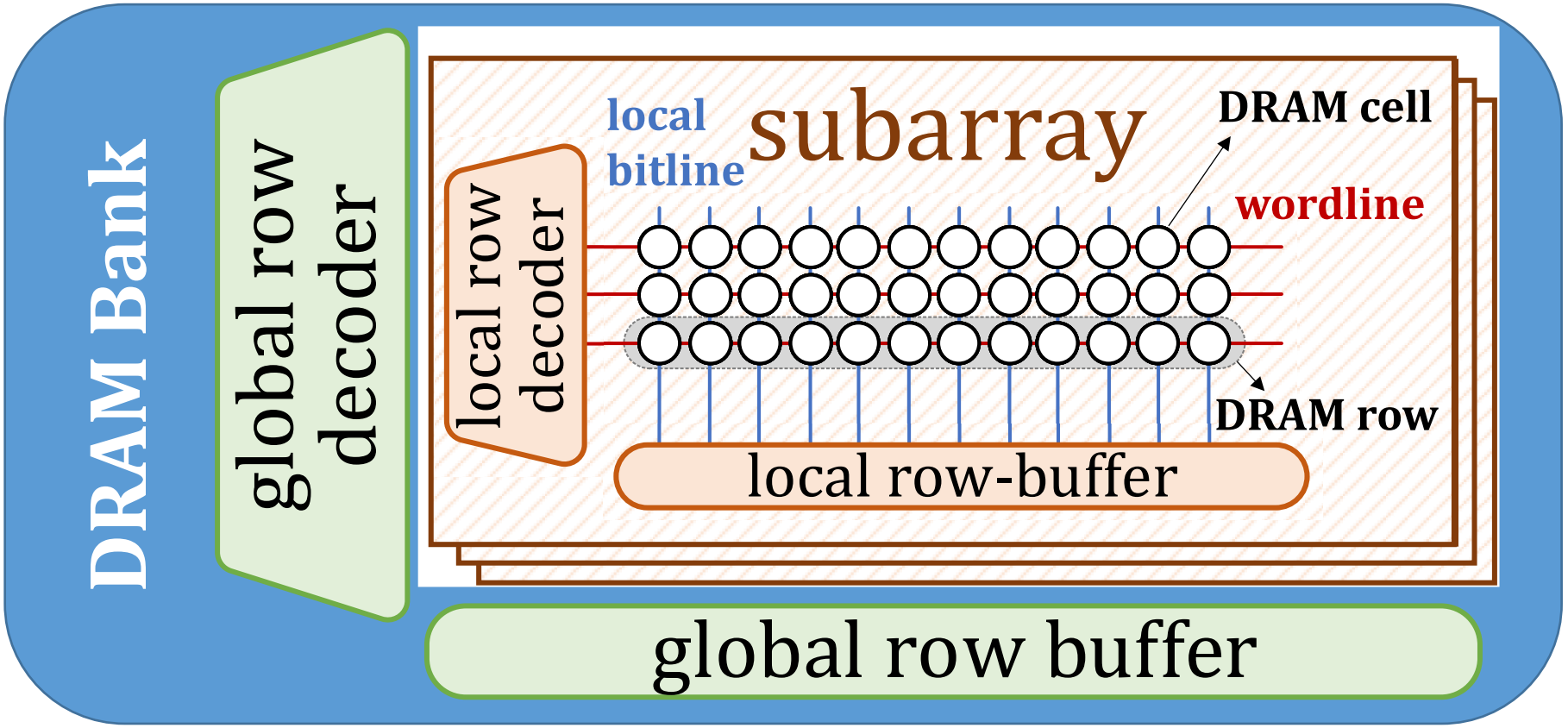
More in the Paper

- Evaluation of 14 Mechanisms representing Four Mitigation Approaches
  - Comprehensive Protection
  - Compatibility with Commodity DRAM Chips
  - Scalability with RowHammer Vulnerability
  - Deterministic Protection

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

DRAM Organization

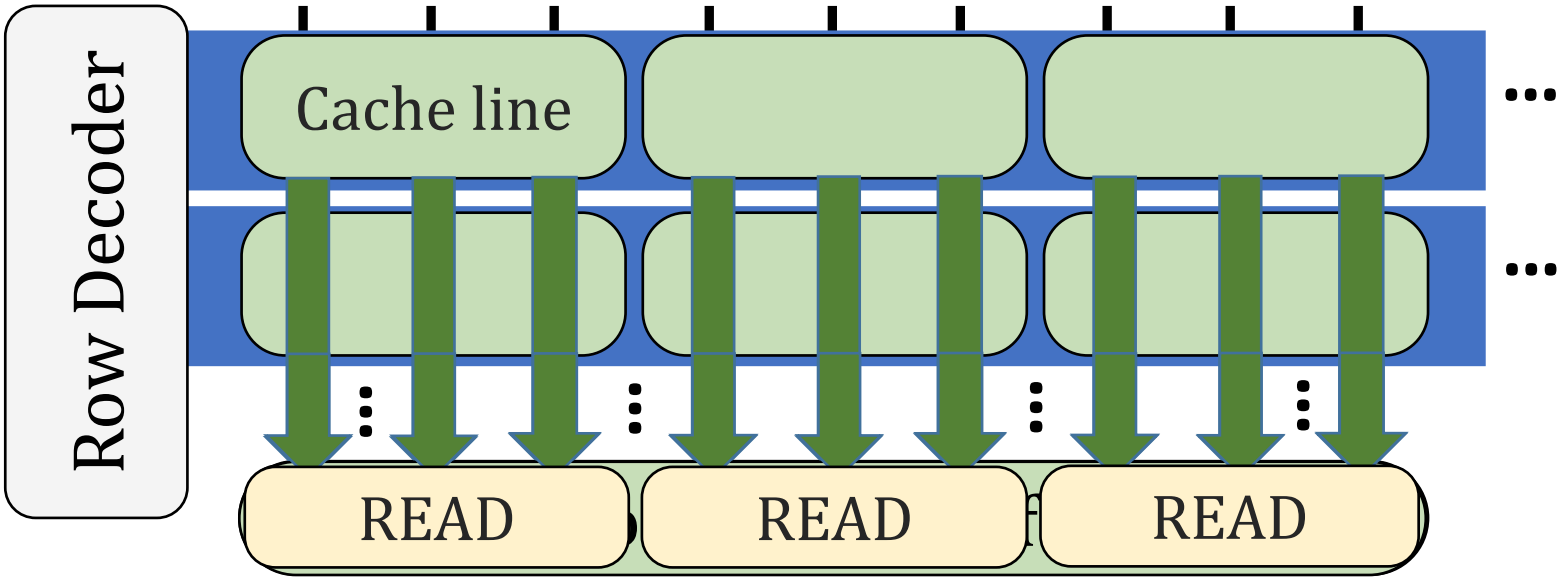
A DRAM bank is hierarchically organized into **subarrays**



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

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

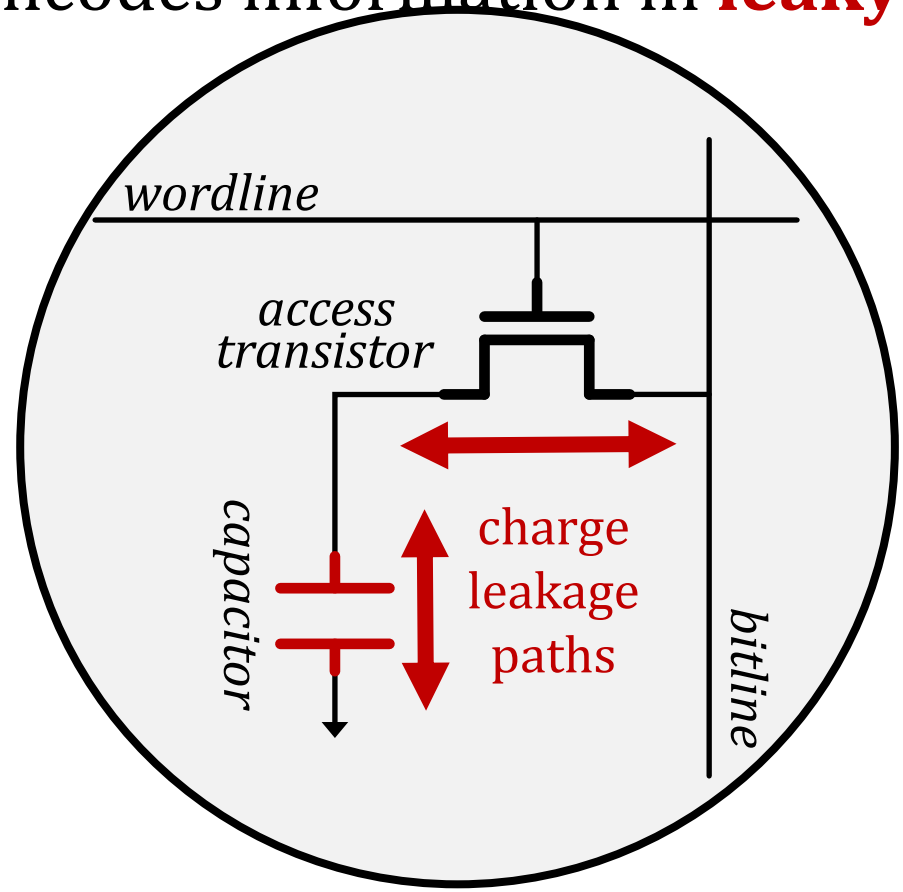
DRAM Operation



DRAM Command Sequence

DRAM Cell

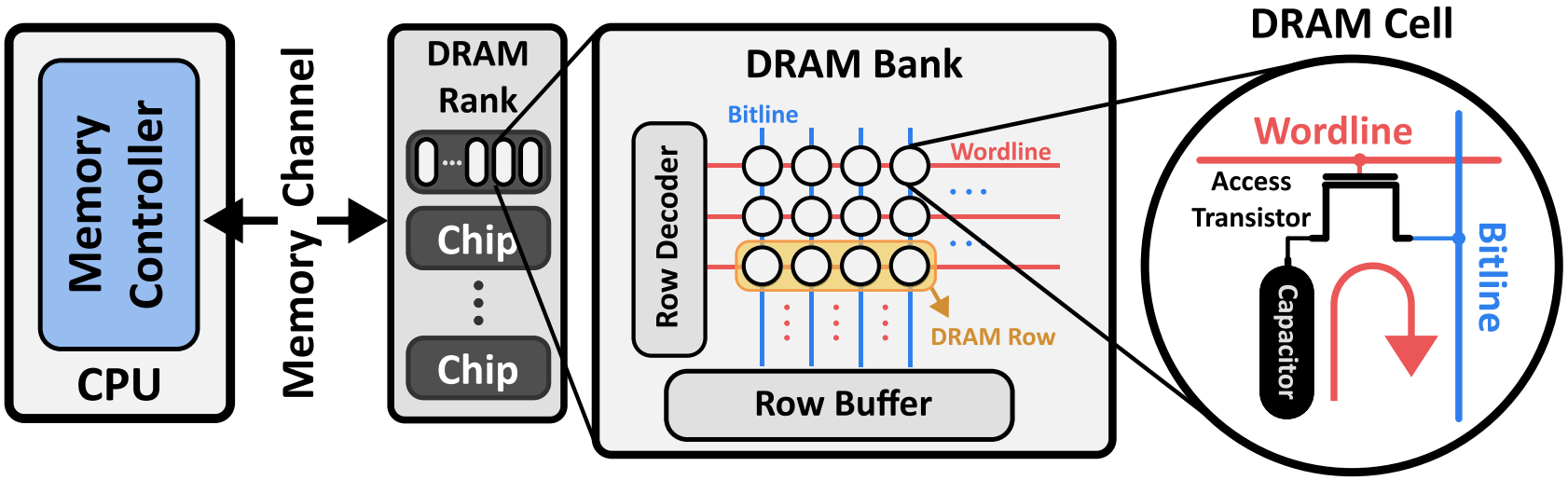
Each cell encodes information in **leaky** capacitors



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

[Patel+ ISCA'17, Kim+ ISCA'20]

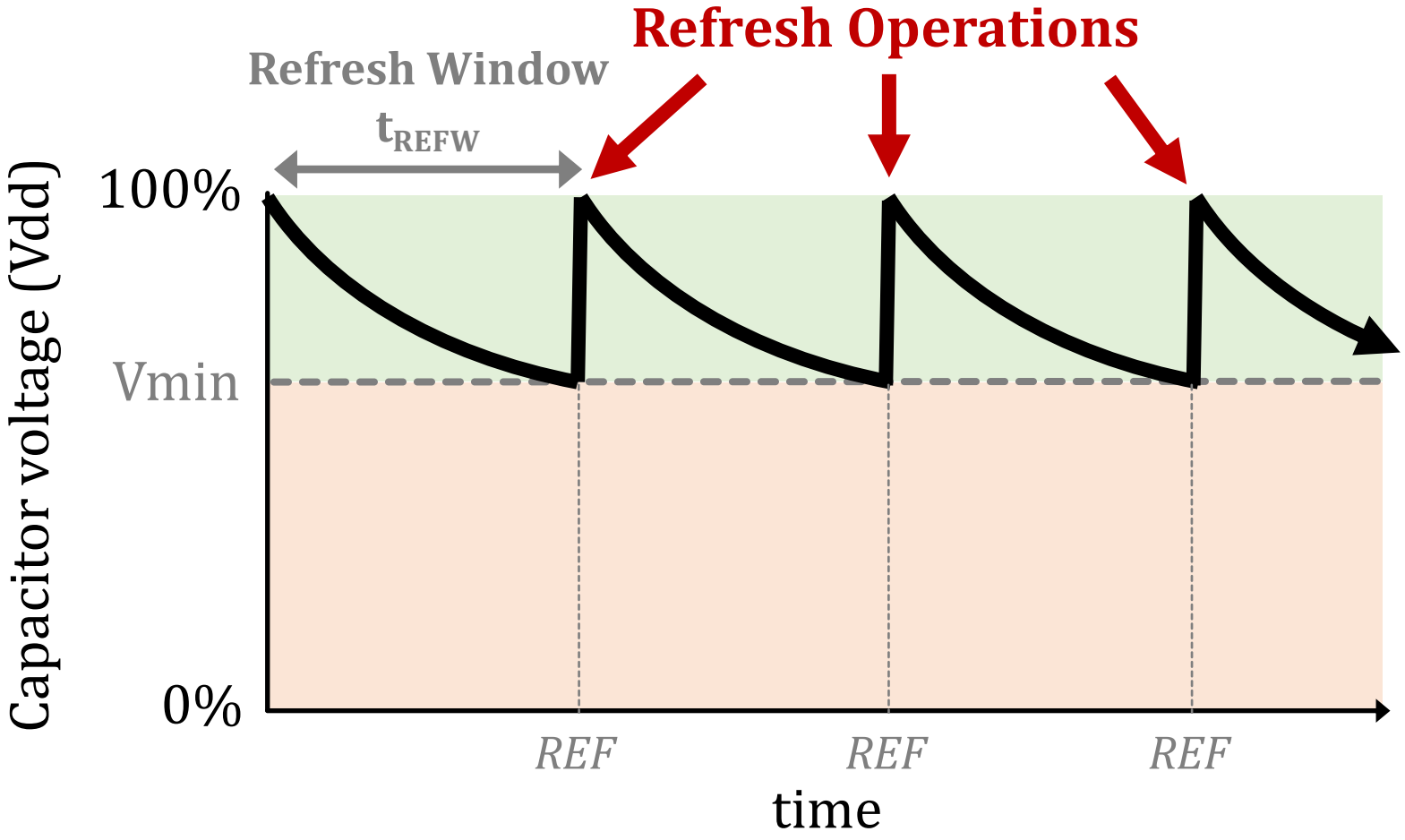
Organizing and Accessing DRAM Cells



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

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

DRAM Refresh

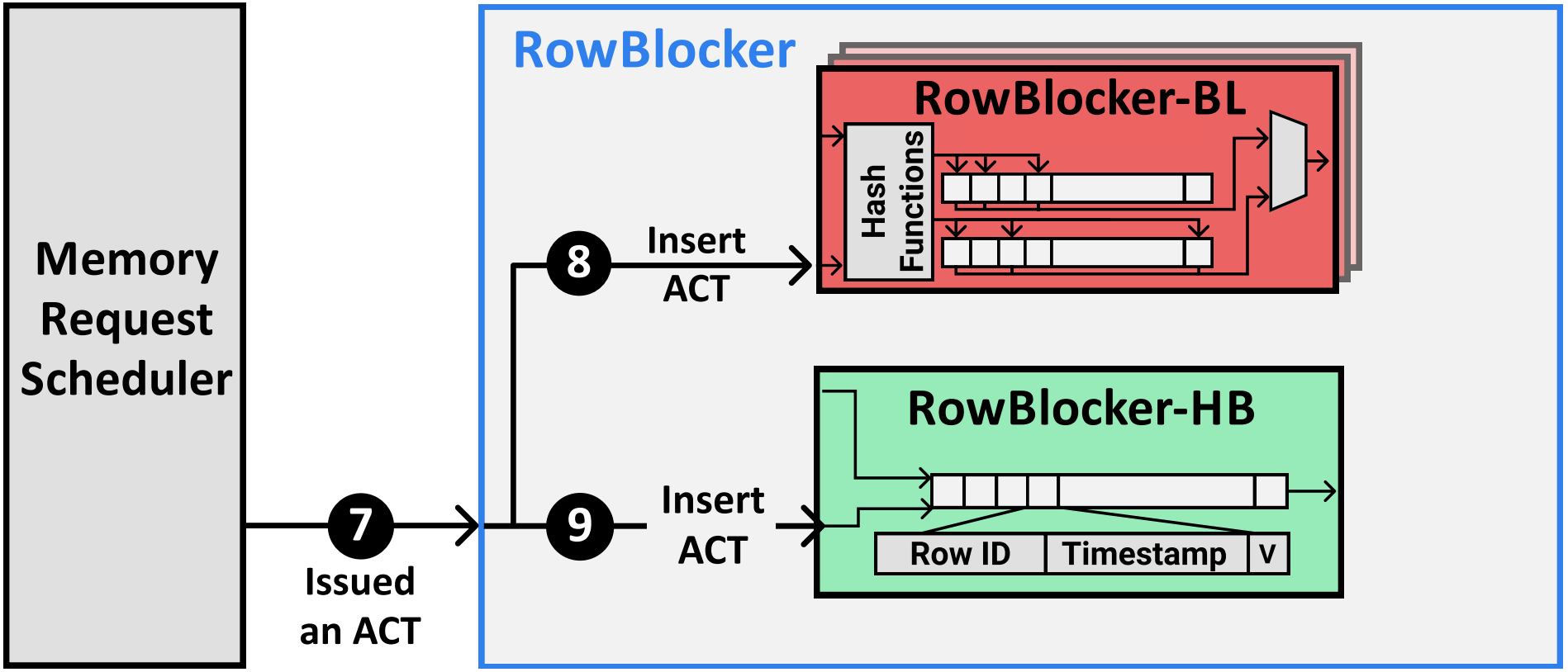


Periodic **refresh operations** preserve stored data

[Patel+ ISCA'17, Kim+ ISCA'20]

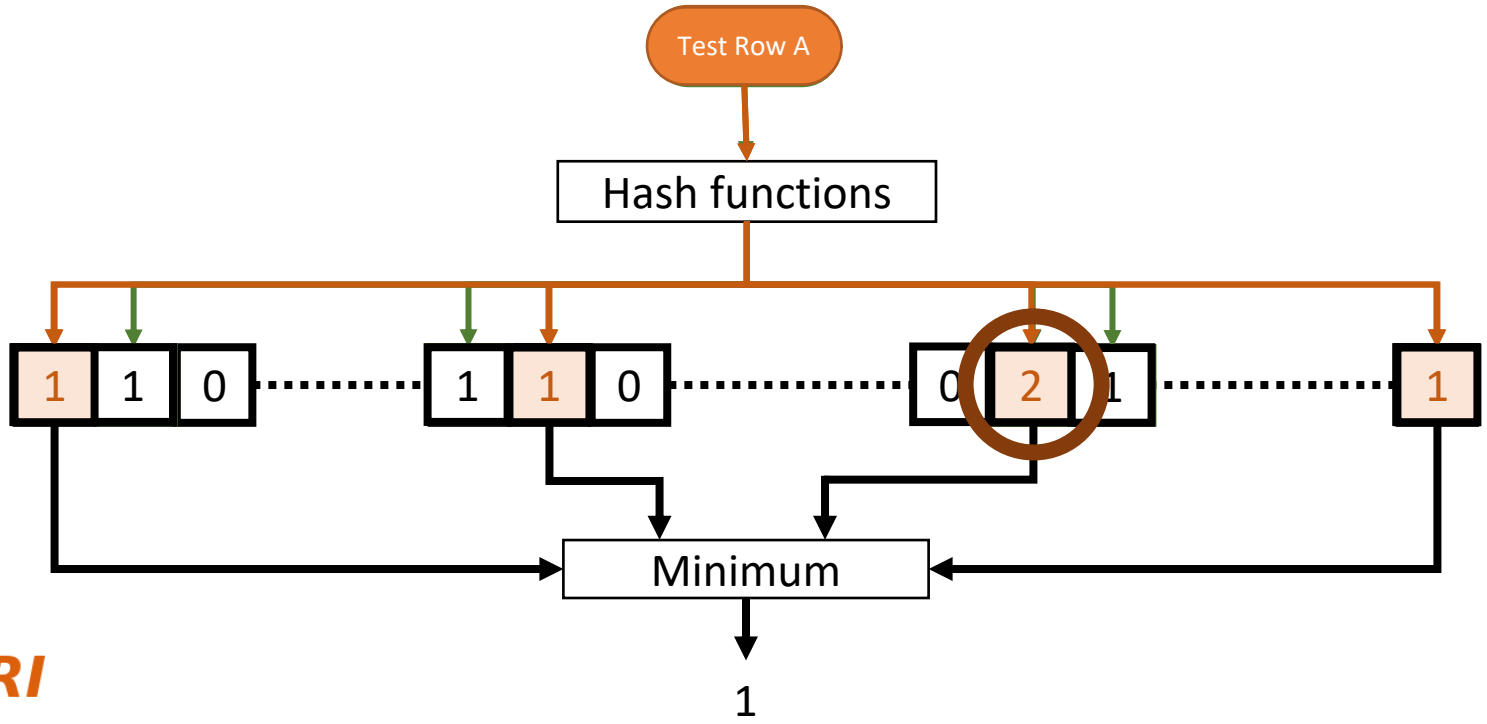
RowBlocker

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



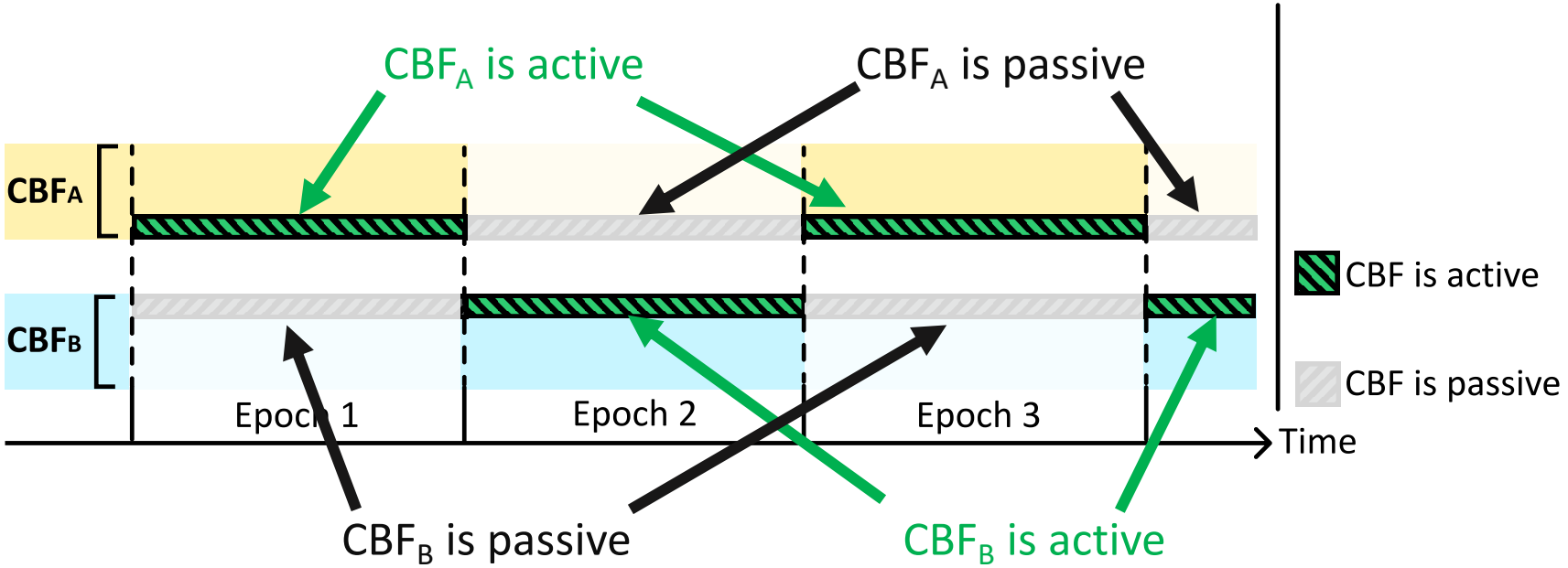
Counting Bloom Filters

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



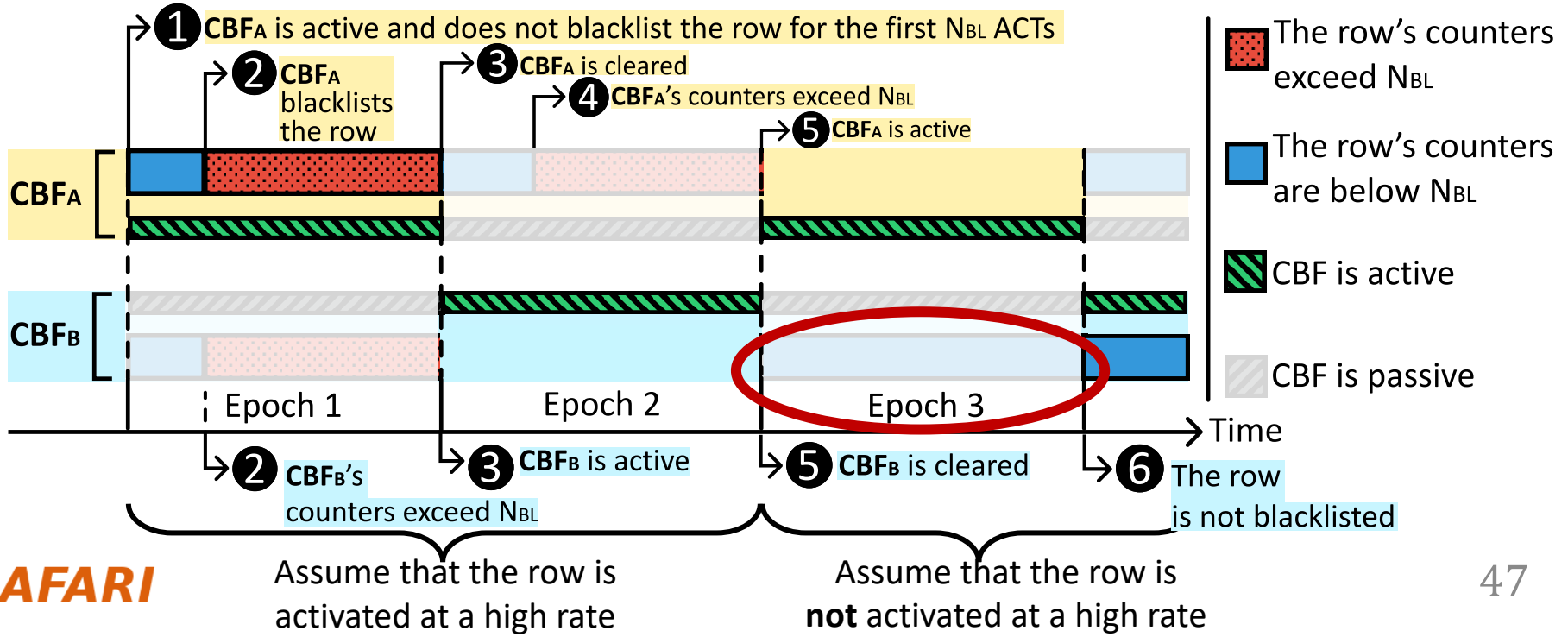
RowBlocker-BL
Blacklisting Logic

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



RowBlocker-BL
Blacklisting Logic

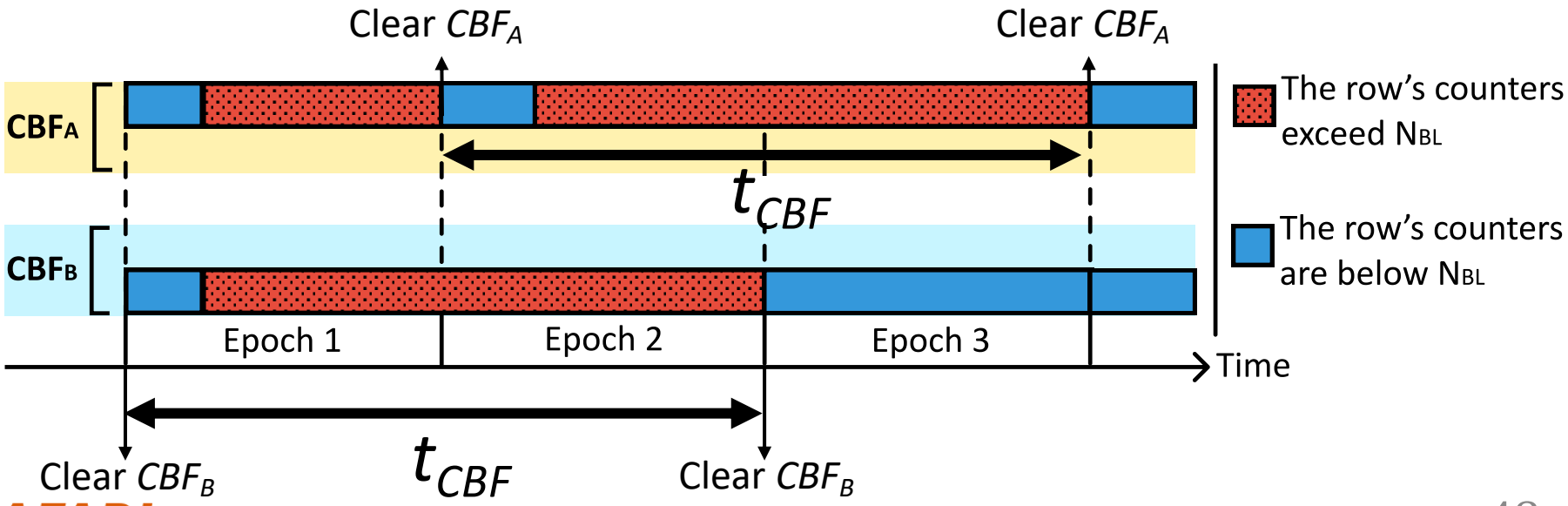
- Blacklisting logic employs two counting Bloom filters
- A new row activation is inserted in both filters
- Only one filter (active filter) responds to test queries
- The active filter changes at every epoch
- Blacklists a row if its activation count reaches the blacklisting threshold (N_{BL})



Limiting the Row Activation Rate

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

$$Activation\ Rate\ in\ a\ t_{CBF} \leq\ N_{RH}\ activations\ in\ a\ refresh\ window\ (t_{REFW})$$



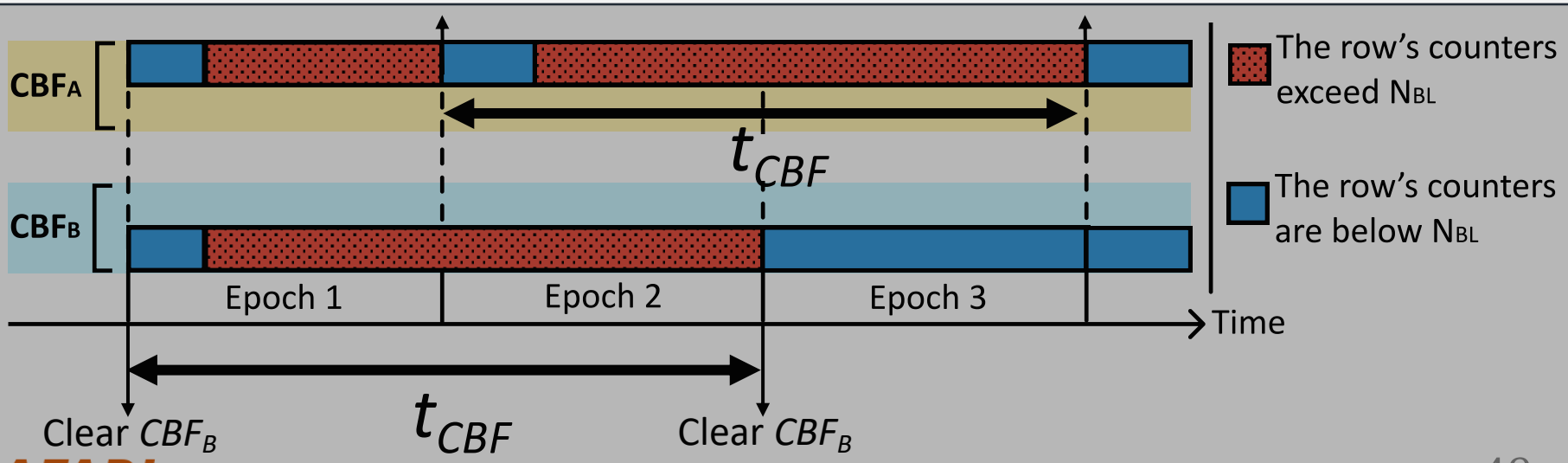
Limiting the Row Activation Rate

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

$$Activation\ Rate\ in\ a\ t_{CBF} \leq N_{RH}\ activations\ in\ a\ refresh\ window\ (t_{REFW})$$

RowHammer Safety Constraint

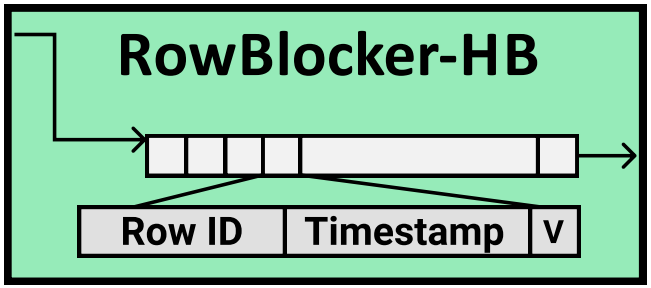
$$N_{CBF}/t_{CBF} \leq N_{RH}/t_{REFW}$$

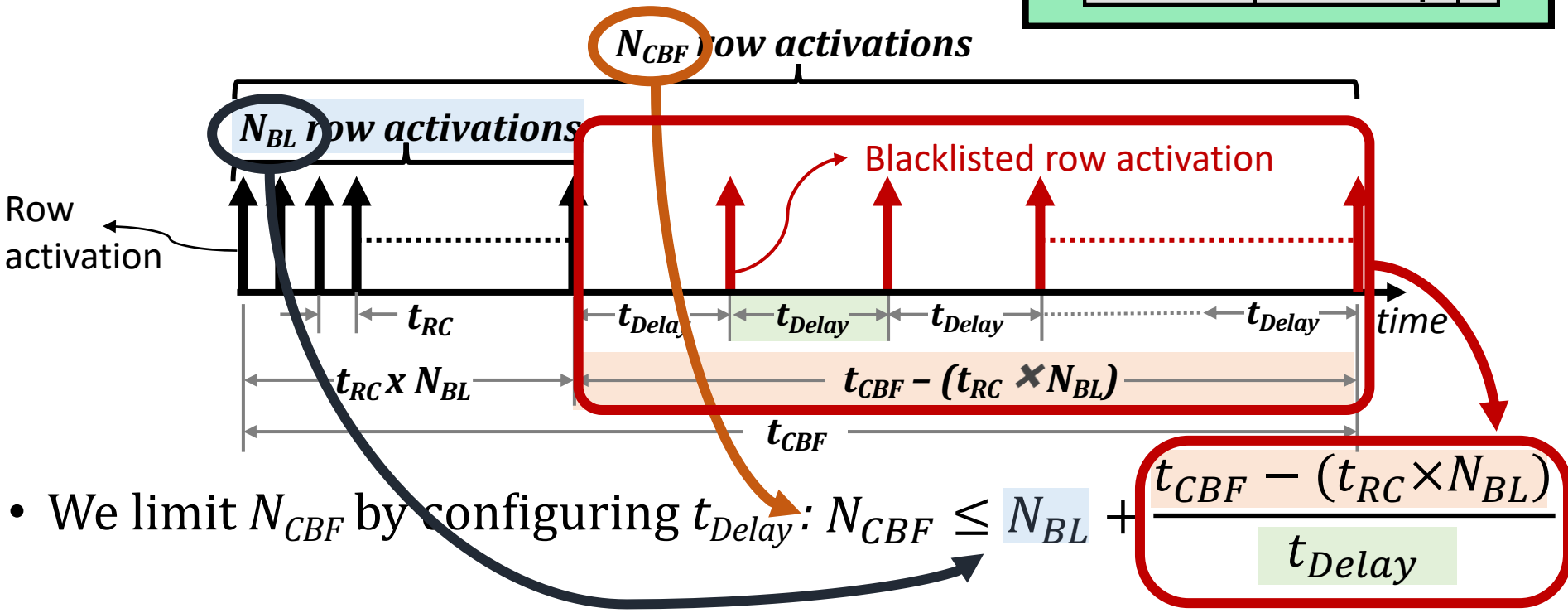


RowBlocker-HB
Limiting the Row Activation Rate

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

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





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

Evaluation

BlockHammer's Hardware Complexity

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

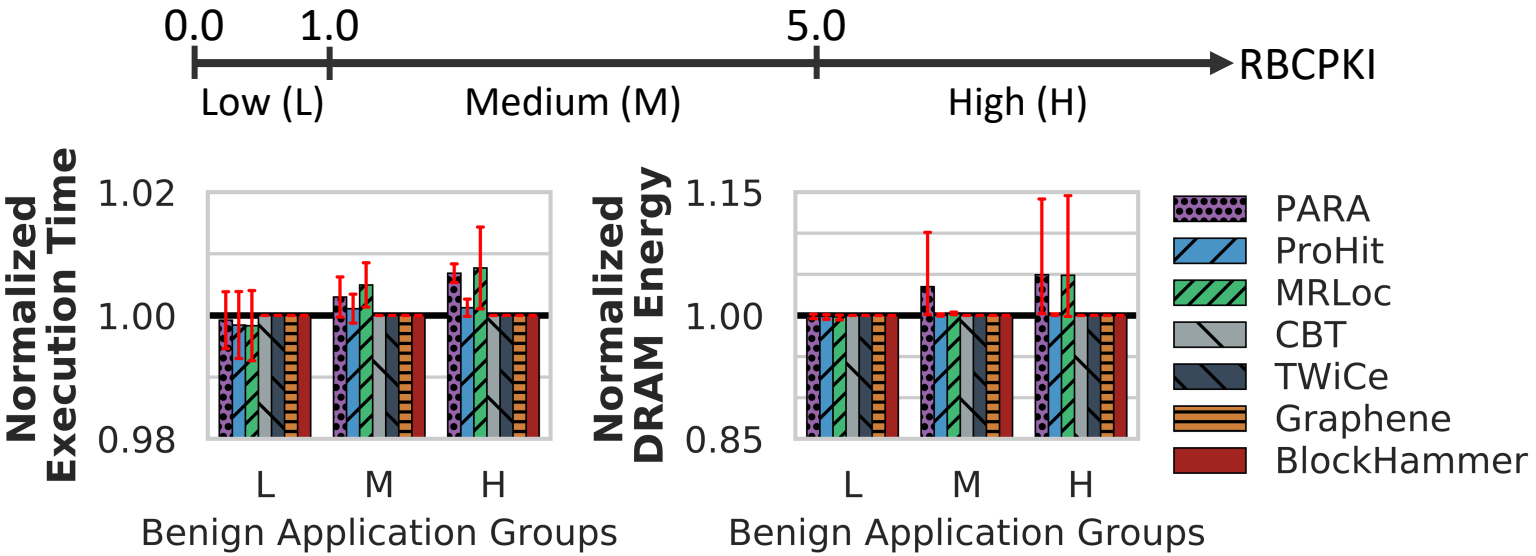
	Mitigation Mechanism	SRAM KB	CAM KB	Area mm^2	%CPU	Access Energy pJ	Static Power mW
N_{RH}=32K	BlockHammer	51.48	1.73	0.14	0.06	20.30	22.27
	PARA [73]	-	-	<0.01	-	-	-
	ProHIT [137]	-	0.22	<0.01	<0.01	3.67	0.14
	MRLoc [161]	-	0.47	<0.01	<0.01	4.44	0.21
	CBT [132]	16.00	8.50	0.20	0.08	9.13	35.55
	TWiCe [84]	23.10	14.02	0.15	0.06	7.99	21.28
	Graphene [113]	-	5.22	0.04	0.02	40.67	3.11

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

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

Evaluation
Performance and DRAM Energy

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



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

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

Conclusion

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

Security Analysis

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

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

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

Table 3: Necessary constraints of a successful attack.

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