ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms

Patrick Jattke* Max Wipfli* Flavien Solt

Michele Marazzi Matej Bölcskei Kaveh Razavi

COMSEC

ETH zürich

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

Abstract

AMD has gained a significant market share in recent years with the introduction of the Zen microarchitecture. While there are many recent Rowhammer attacks launched from Intel CPUs, they are completely absent on these newer AMD CPUs due to three non-trivial challenges: 1) reverse engineering the unknown DRAM addressing functions, 2) synchronizing with refresh commands for evading in-DRAM mitigations, and 3) achieving a sufficient row activation throughput. We address these challenges in the design of ZenHammer, the first Rowhammer attack on recent AMD CPUs. ZenHammer reverse engineers DRAM addressing functions despite their non-linear nature, uses specially crafted access patterns for proper synchronization, and carefully schedules flush and fence instructions within a pattern to increase the activation throughput while preserving the access order necessary to bypass in-DRAM mitigations. Our evaluation with ten DDR4 devices shows that ZenHammer finds bit flips on seven and six devices on AMD *Zen 2* and *Zen 3*, respectively, enabling Rowhammer exploitation on current AMD platforms. Fur-

DRAM Addressing. Previous work that reverse engineered the DRAM addressing functions on Intel and ARM platforms assumes that these functions are constructed by XOR-ing certain physical address bits with each other [52]. We find that on AMD platforms, this assumption leads to an incomplete recovery of address functions. Our experiments show that AMD's memory controllers require offsets for certain physical address ranges before applying the XOR functions. Adjusting for these offsets, better handling of noisy measurements, and considering higher physical address bits lead to the recovery of correct and complete DRAM addressing functions for these CPUs. Yet, this initial version of ZenHammer only triggers bit flips on five (*Zen 2*) and none (*Zen 3*) of our ten DDR4 DRAM modules, while an Intel-based fuzzer [17] triggers bit flips on eight of them. We track down the reason to inadequate synchronization with DRAM refresh commands and the low throughput of activations sent to DRAM.

Refresh Synchronization. Modern DRAM devices employ in-DRAM Target Row Refresh (TRR) mitigations that detect potential victims of a Rowhammer attack and internally

* Joint first authors

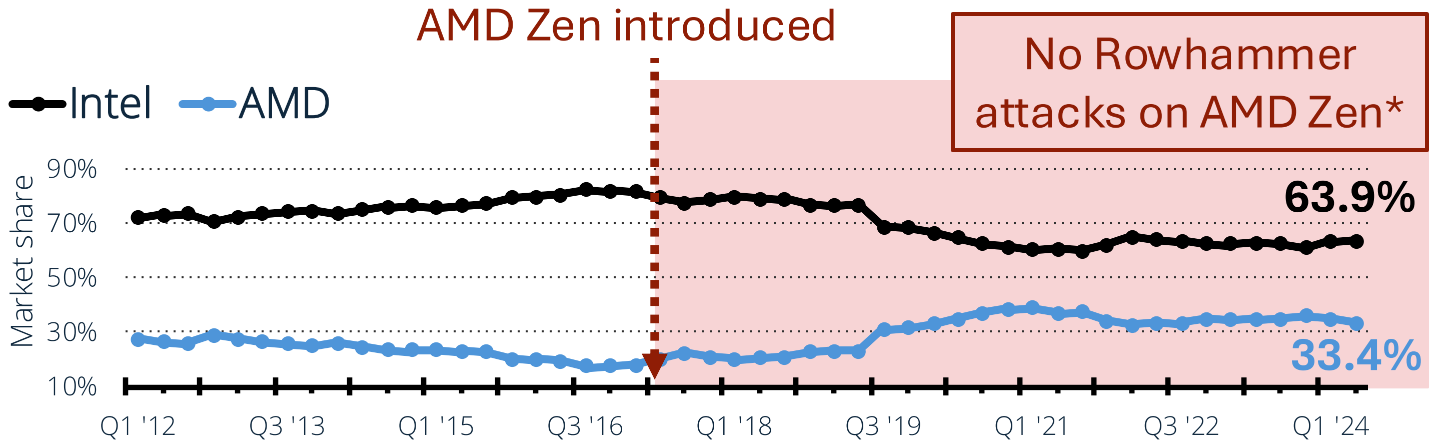
33RD USENIX Security Symposium

August 14–16, 2024
Philadelphia, PA, USA

Track 2 – Side Channel II: Rowhammer

Executive Summary

- Today, **every third** sold x86 CPU is from AMD



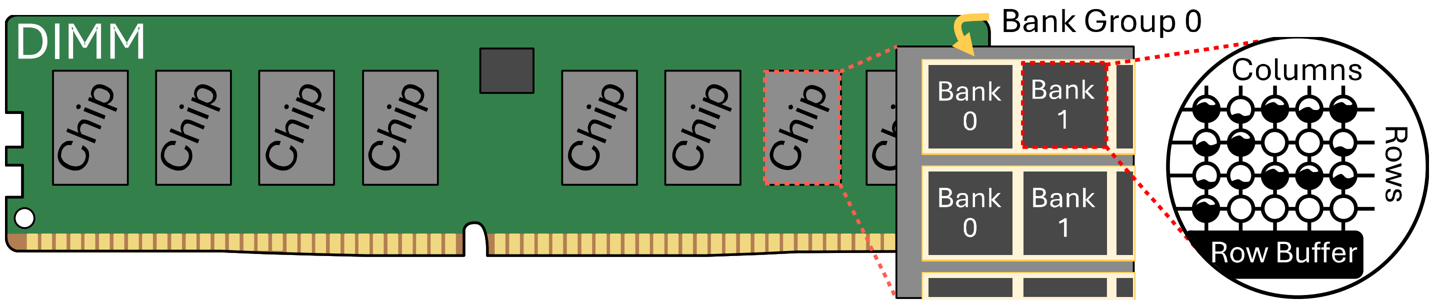
Are current AMD Zen-based platforms vulnerable to Rowhammer?

We built ZenHammer to answer this!

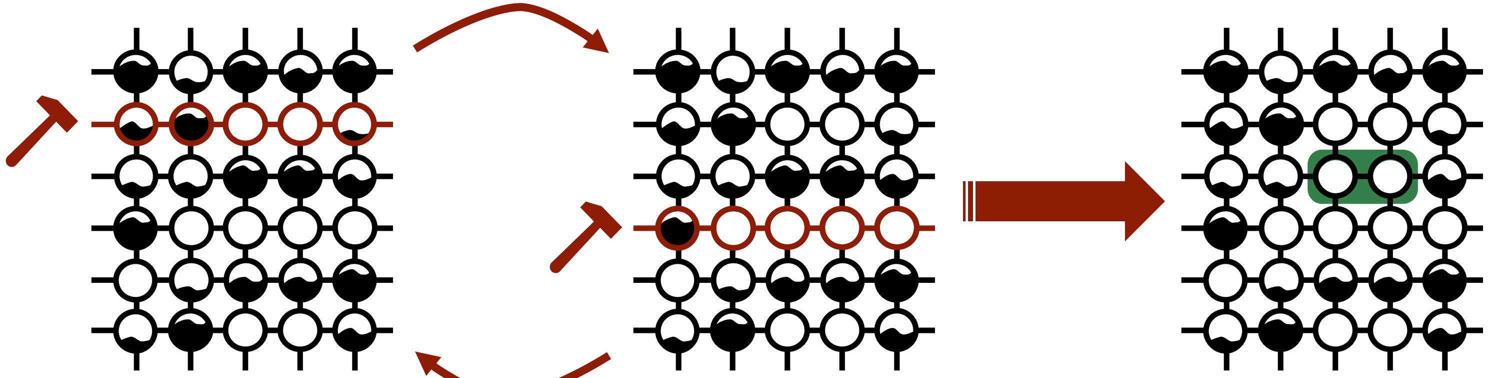
- We find bit flips on **7/10 DIMMs (Zen 2)** and **6/10 DIMMs (Zen 3).**
- Up to **46x more bit flips** on Zen 3 than on Coffee Lake.
- First bit flips on one **DDR5 DIMM** on Zen 4.

Background

DIMM Organization



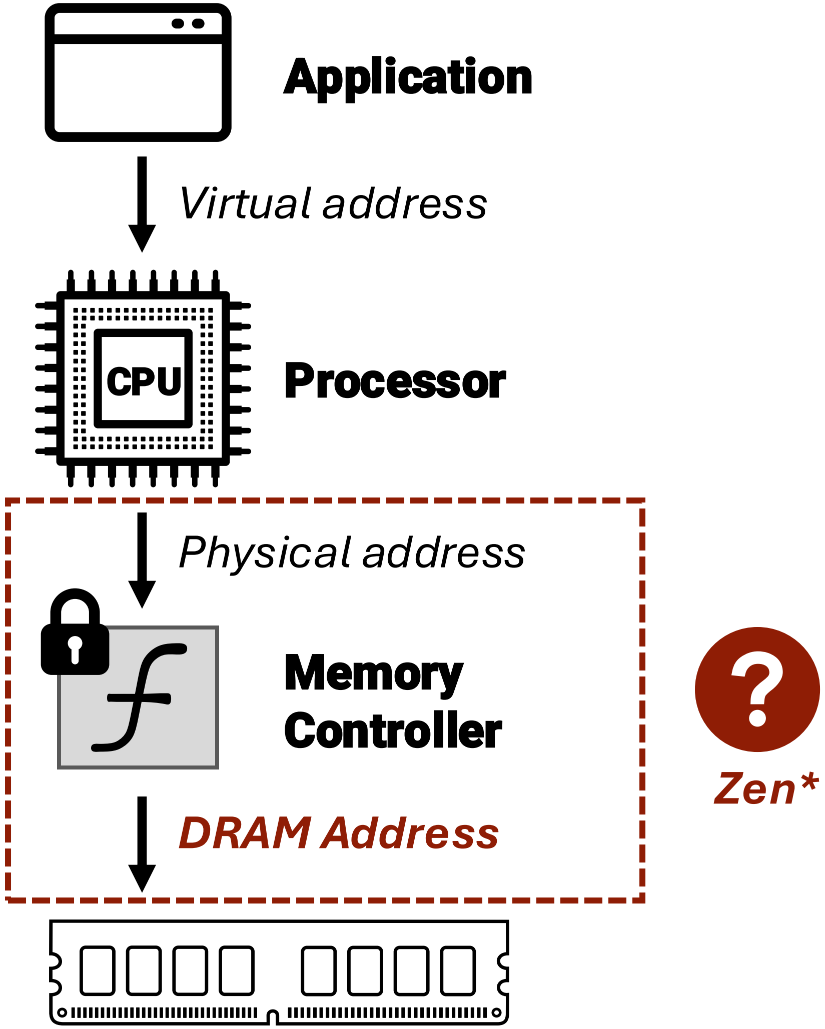
Rowhammer



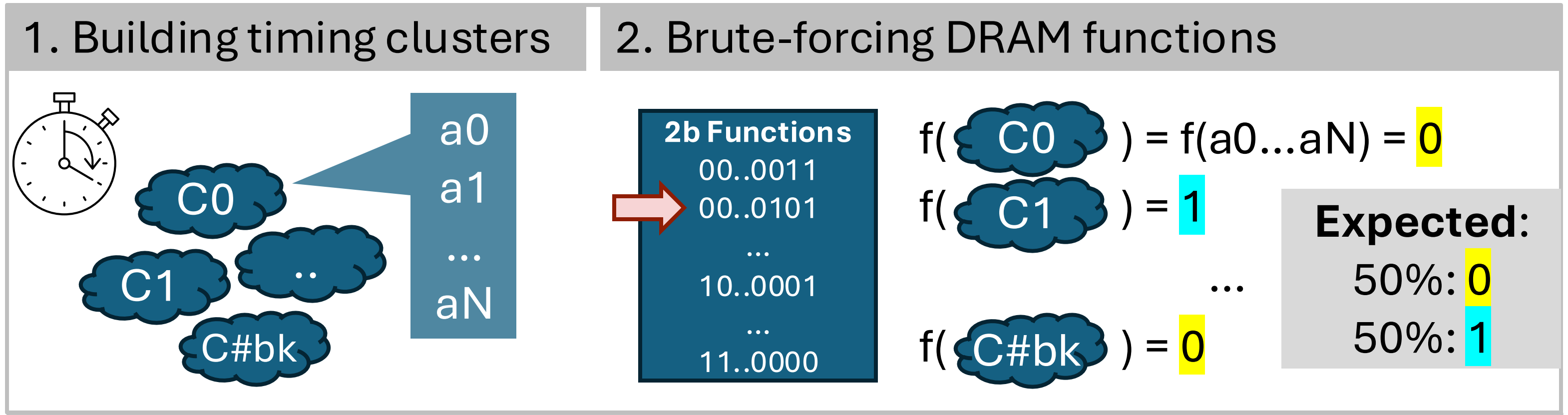
Double-sided pattern

Two bits flipped!

DRAM addressing



C1 Recovering DRAM address mappings



1. Building timing clusters

2. Brute-forcing DRAM functions

f(C0) = f(a0...aN) = 0

f(C1) = 1

...

f(C#bk) = 0

Expected:

50%: 0

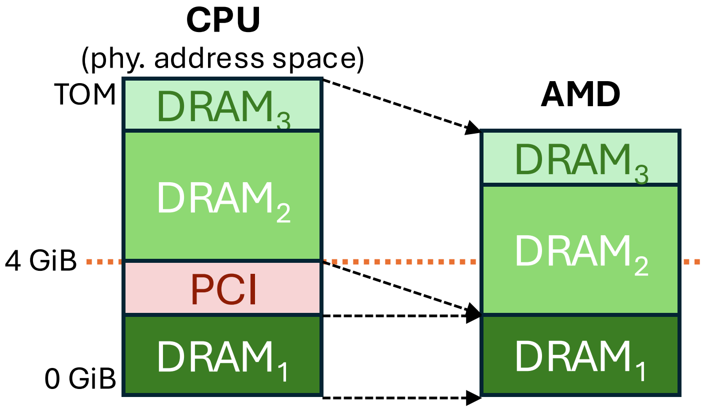
50%: 1

DRAMA [1] could not recover mappings.

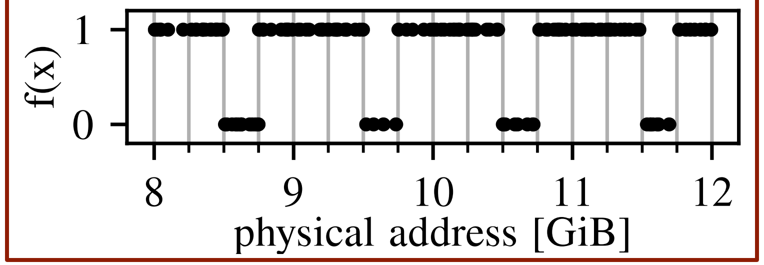
⇨ Functions only worked on **limited** memory regions.

O1. DRAM functions are non-linear and require an address offset.

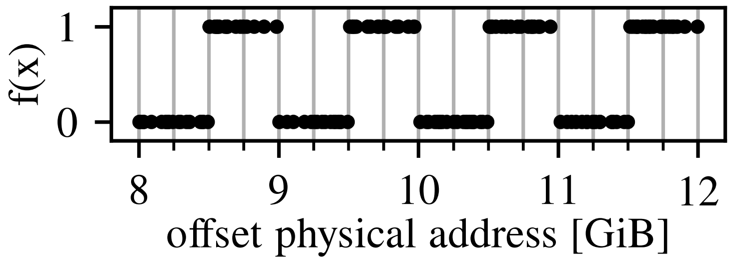
O2. Memory blocks >1 GiB need to be accessed for mapping recovery.



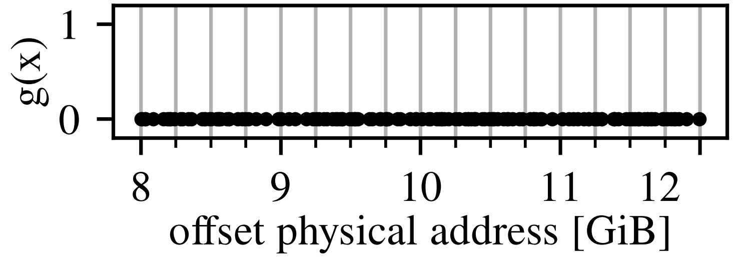
Function output of *f(x)* = *0x64440100* for same-cluster addresses.



Function output of *f(x)* after applying an **offset** of **768 MiB**.



Recovery of **correct function** *g(x)* = *0x44440100* with offset 768 MiB.



[1] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," in USENIX Security '16.

See our paper for more DRAM configurations!

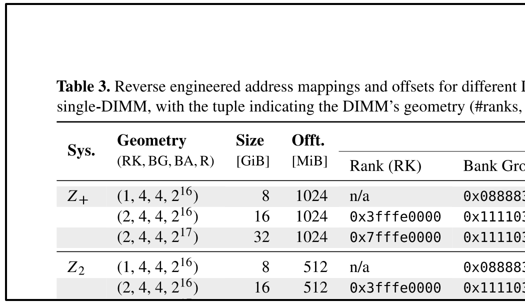


Table 3. Reverse engineered address mappings and offsets for different I
single-DIMM, with the tuple indicating the DIMM's geometry (#ranks,

Sys.	**Geometry** (RK, BG, BA, R)	**Size** [GiB]	**Offt.** [MiB]	Rank (RK)	Bank Gro
Z_+	$(1, 4, 4, 2^{16})$	8	1024	n/a	0x088883
	$(2, 4, 4, 2^{16})$	16	1024	0x3fffe0000	0x111103
	$(2, 4, 4, 2^{17})$	32	1024	0x7fffe0000	0x111103
Z_2	$(1, 4, 4, 2^{16})$	8	512	n/a	0x088883
	$(2, 4, 4, 2^{16})$	16	512	0x3fffe0000	0x111103

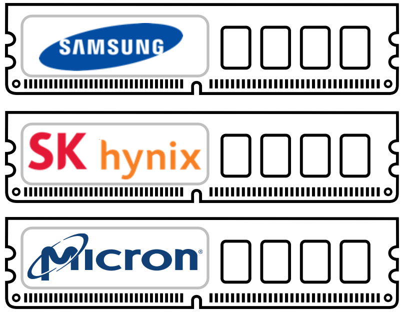
C1 Recovering DRAM address mappings

Visualization of **<1 RK, 4 BG, 4 BK, 2^{16} rows>** functions.

Physical Address Bit			32				28				24				20				16				12				8				4				0
AMD Zen 3																																			
BG 0x0022220100						🟥				🟥				🟥				🟥									🟥								
BG 0x0044440200					🟥				🟥				🟥				🟥									🟥									
BA 0x0088880400				🟥				🟥				🟥				🟥									🟥										
BA 0x0111100800			🟥				🟥				🟥				🟥									🟥											
ROW 0x01fffe0000			🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																	
AMD Zen+/Zen 2																																			
BG 0x0088883fc0				🟥				🟥				🟥				🟥						🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥						
BG 0x0111104000			🟥				🟥				🟥				🟥						🟥														
BA 0x0022228000						🟥				🟥				🟥				🟥		🟥															
BA 0x0044450000					🟥				🟥				🟥				🟥		🟥																
ROW 0x01fffe0000			🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																	

Testing for Rowhammer

- We created the Blacksmith [2] fork ***ZenHammer*** with our found DRAM address mappings.
- DIMMs from major manufacturers:



6x

2x

2x

- **6h fuzzing** runs on each DIMM.

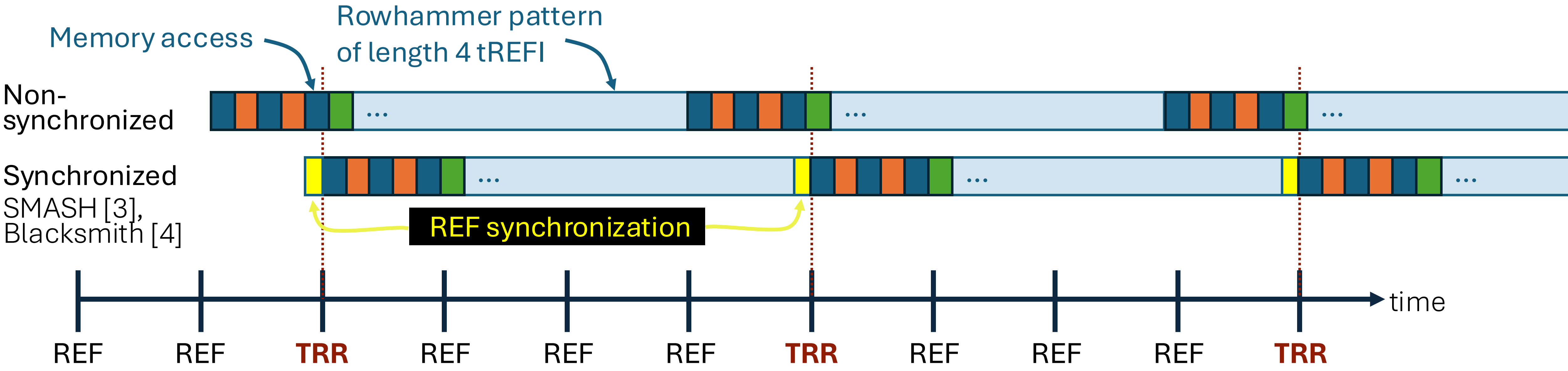
⇨ Porting the DRAM address functions is **insufficient** to do Rowhammer on AMD Zen-based systems.

	Zen 2		**Zen 3**		**Coffee Lake**	
DIMM	#Patt.	#Bit Flips	#Patt.	#Bit Flips	#Patt.	#Bit Flips
S_0	14	19	0	0	122	3’502
S_1	4	4	0	0	102	1’374
S_2	14	28	0	0	782	22’339
S_3	0	0	0	0	3	3
S_4	4	5	0	0	47	654
S_5	6	7	0	0	155	4’131
H_0	0	0	0	0	24	35
M_0	0	0	0	0	16	23
	5/10 devices		**0**/10 devices		**8**/10 devices	

[2] P. Jattke, V. van der Veen, P. Frigo, S. Gunter, and K. Razavi, “BLACKSMITH: Scalable Rowhammering in the Frequency Domain,” in *IEEE S&P ’22*.

In-DRAM TRR: REF synchronization

- Rowhammer mitigations (**TRR**) act at the **same time as** periodic **REFs.**



Requirement for doing Rowhammer:

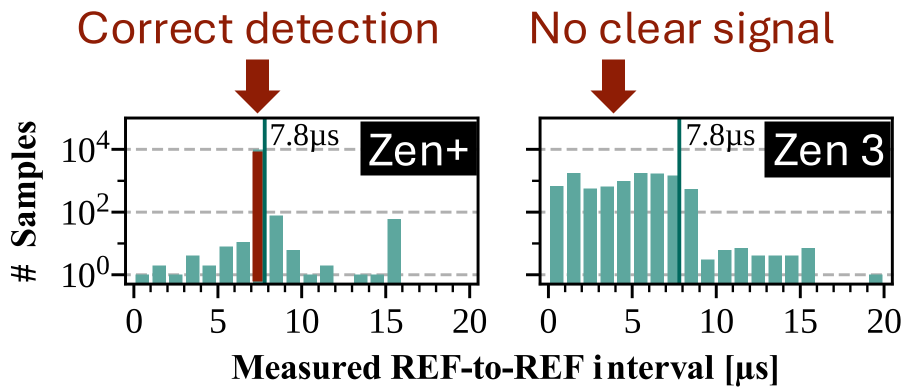
⇨ proper **synchronization with REFs** (C2)

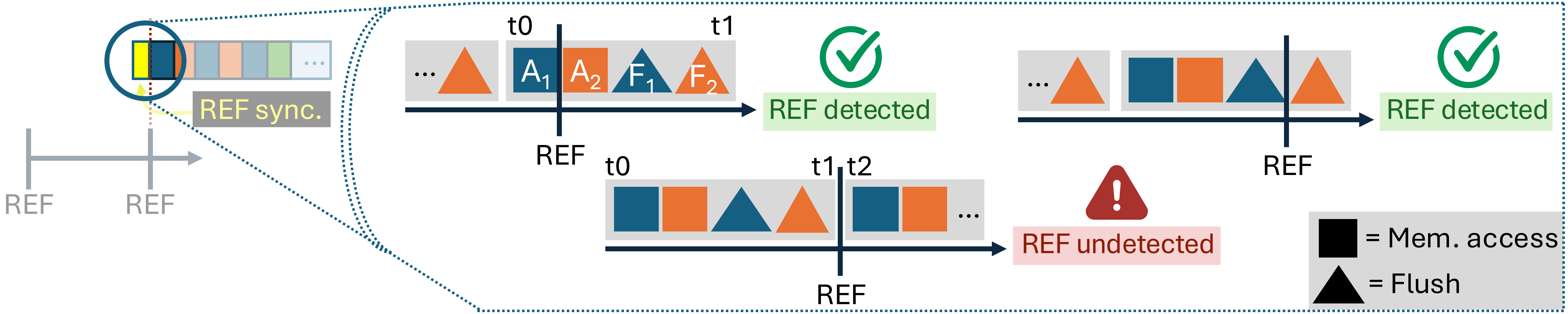
[3] F. de Ridder, P. Frigo, E. Vannacci, H. Bos, C. Giuffrida, and K. Razavi, “SMASH: Synchronized Many-sided Rowhammer Attacks From JavaScript,” in *USENIX Security ’21*

[4] P. Jattke, V. van der Veen, P. Frigo, S. Gunter, and K. Razavi, “BLACKSMITH: Scalable Rowhammering in the Frequency Domain,” in *IEEE S&P ’22*.

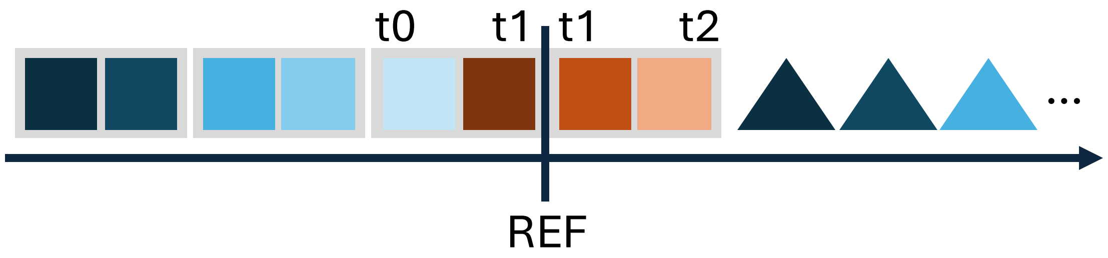
C2 Adapting timing-based REF synchronization

- We measured the time between REFs.
  ⇨ Synchronization does not work on **Zen 3.**

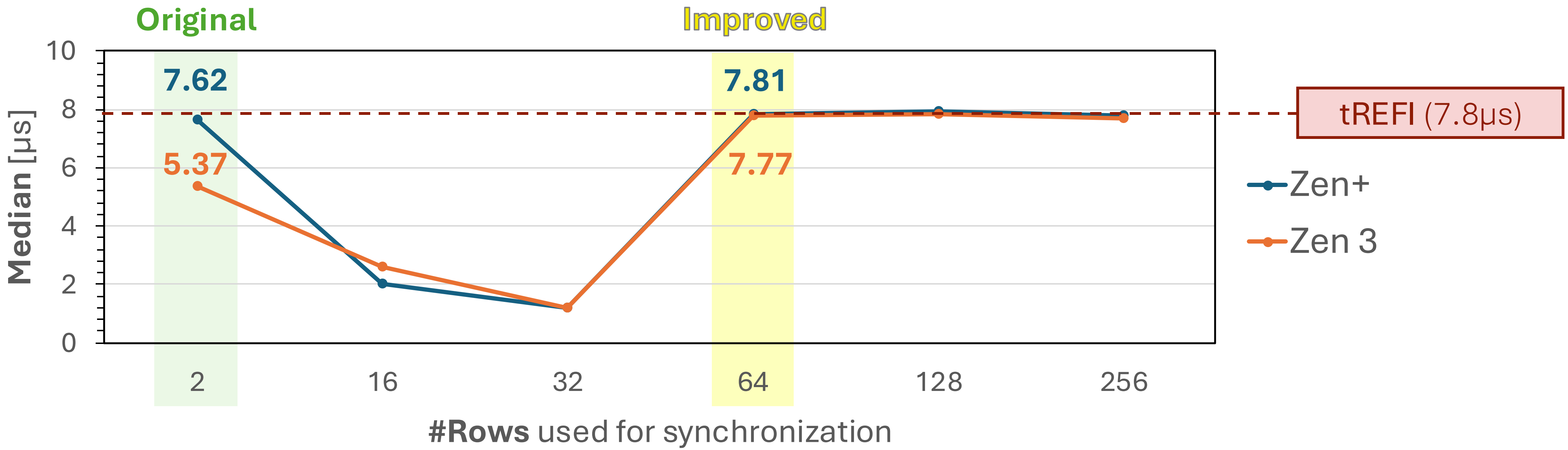




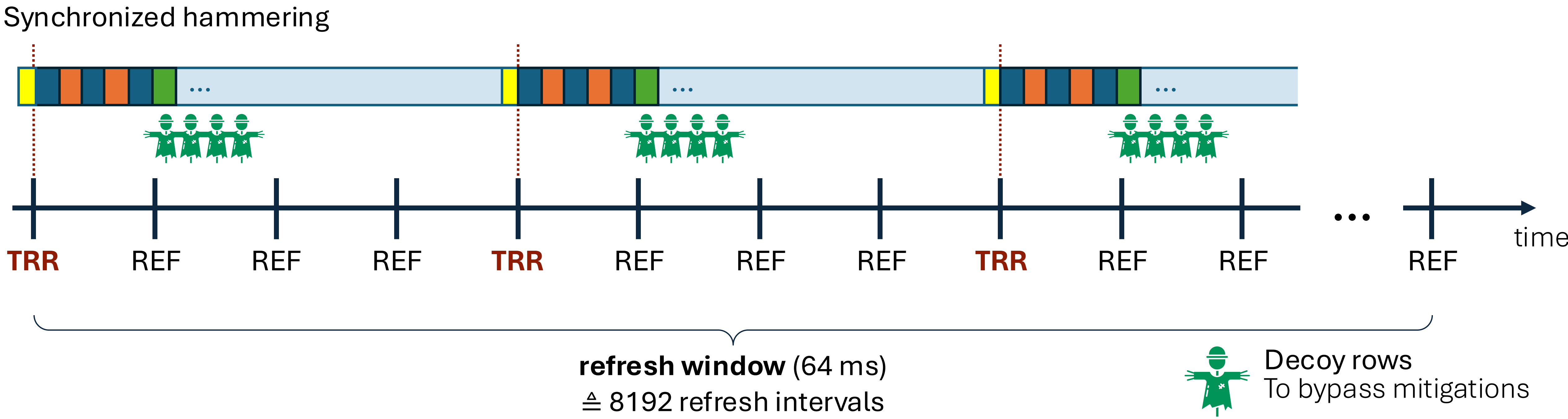
- **Solution**: Continuous, non-repeating refresh synchronization.



C2 Adapting timing-based REF synchronization



In-DRAM TRR: activation count



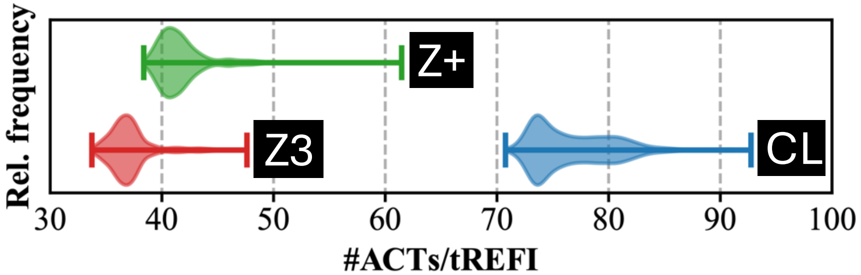
Requirements for doing Rowhammer:
⇨ sufficient **activation count** to the aggressors (C3)

C3 Increasing the ACT rate and preserving order

- On average, **ACTs/tREFI** on Z+ (41.9) and Z3 (37.2) are **halved** compared to CL (76.8).

> 40 ACTs/tREFI gives 36K ACTs with 18 aggs.
> ⇨ too low for many devices

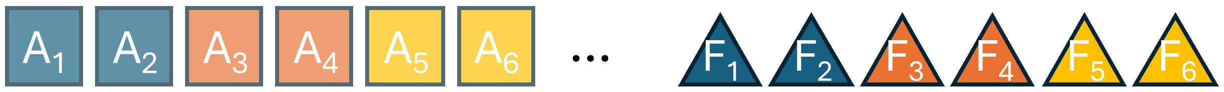
- **Systematic testing** of different hammering instruction sequences:
  - Cache flushing (e.g., CLFLUSH/CLFLUSHOPT, gathered/scattered)
  - Memory barriers (e.g., mfence, lfence, sfence)
  - Access types (e.g., load vs store)
  - Vector instructions (e.g., vpgatherdd)



Zen 3

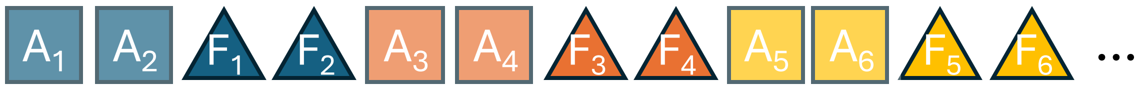
Access Type	Flushing Strategy	Fence Type	#Rows						
			1	2	4	8	**16**	**32**	**256**
MOV (load)	gather	`mfence`	24	49	71	91	100	110	114

Gathered flushes



BEST

Scattered flushes

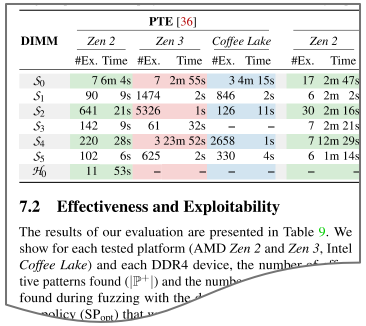


s

he

Evaluation: Best Rowhammer pattern

- Optimizations drastically increased #effective patterns.
- **Higher #bitflips** in 4 cases (Z2) and 5 cases (Z3) compared to Coffee Lake.
- Bit flips on DIMM $\mathbf{H_0}$ on Z2 where we found none on Coffee Lake.
- We also analyzed the **impact on exploitation**, see our paper for results!



	PTE [36]							
DIMM	*Zen 2*		*Zen 3*		*Coffee Lake*		*Zen 2*	
	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time
$\mathcal{S}_0$	7	6m 4s	7	2m 55s	3	4m 15s	17	2m 47s
$\mathcal{S}_1$	90	9s	1474	2s	846	2s	6	2m 2s
$\mathcal{S}_2$	641	21s	5326	1s	126	11s	30	2m 16s
$\mathcal{S}_3$	142	9s	61	32s	–	–	7	2m 21s
$\mathcal{S}_4$	220	28s	3	23m 52s	2658	1s	7	12m 29s
$\mathcal{S}_5$	102	6s	625	2s	330	4s	6	1m 14s
$\mathcal{H}_0$	11	53s	–	–	–	–	–	–

7.2 Effectiveness and Exploitability

The results of our evaluation are presented in Table 9. We show for each tested platform (AMD *Zen 2* and *Zen 3*, Intel *Coffee Lake*) and each DDR4 device, the number
tive patterns found ($|\mathbb{P}^+|$) and the numbe
found during fuzzing with the
policy (SP_{opt}) that

	Zen 2		**Zen 3**		**Coffee Lake**	
DIMM	#Patt.	#Bit Flips	#Patt.	#Bit Flips	#Patt.	#Bit Flips
S_0	51	6’945	31	17’775	122	6’782
S_1	26	1’758	25	15’613	102	10’106
S_2	97	12’893	45	79’306	782	1’708
S_3	8	2’020	1	667	3	0
S_4	60	1’183	43	13	47	18’357
S_5	25	1’911	26	10’741	155	5’860
H_0	6	182	0	0	0	0
H_1	0	0	0	0	24	0
M_0	0	0	0	0	0	0
M_1	0	0	0	0	16	2
	7/10 devices		**6/10 devices**		**8/10 devices**	
	5/10 devices		0/10 devices			

Demo: PTE Attack on AMD Zen 3

Evaluation: ZenHammer on DDR5

- Upon the request of reviewers, we extended our evaluation to **Zen 4.**
- We repeated all experiments and tested **10 random DDR5 DIMMs.**
  - **4x** SAMSUNG
  - **1x** SK hynix
  - **5x** Micron
- We found bit flips on **1/10** DIMMs:
  - **41’995 bit flips** during 256 MiB sweep

***Reviewer A:** Do you have any early results/thoughts on **Zen4** applicability?*

***Reviewer C:** However, the newest microarchitecture that is evaluated is Zen 3 from 2020. Since then, there have been […] **Zen 4** (2022) […]*

Microarch.	Release Date	CPU
Zen 4	September 2022	Ryzen 7 7700X
Zen 3	November 2020	Ryzen 5 5600G
Zen 2	July 2019	Ryzen 5 3600X
Zen+	April 2018	Ryzen 5 2600X

Conclusion

Current AMD Zen-based systems are equally **vulnerable to Rowhammer** as Intel systems.

DRAM addr. mappings
for Zen 2/3/4 incl. offsets.

ZenHammer bit flips
DDR4
DDR5
Zen 2: 7/10 DIMMs
Zen 3: 6/10 DIMMs
Zen 4: 1/10 DIMMs

First ever reported DDR5 bit flips!

Up to 46x more bit flips
on Zen 3 compared to Coffee Lake.

Exploitation
in the best case (PTE) in just 6s (Zen 2) and 2s (Zen 3).

End-to-end PTE exploit on Zen 3.

Check out our paper for more information!

Are current AMD Zen-based platforms vulnerable to Rowhammer attacks?

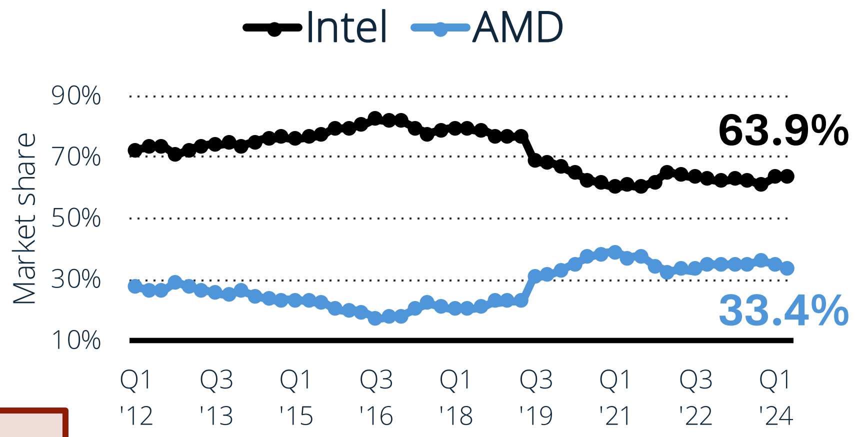
Our test systems

Microarchitecture	Release Date	CPU
Zen 3	November 2020	Ryzen 5 5600G
Zen 2	July 2019	Ryzen 5 3600X
Zen+	April 2018	Ryzen 5 2600X

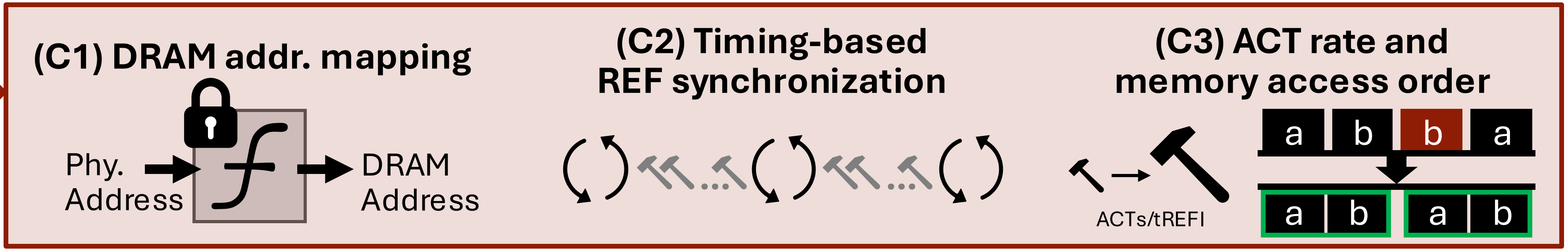
Executive Summary

- Today, every third sold x86 CPU is from AMD
  ⇨ not reflected in Rowhammer attack research

19x Intel **vs.** **1x** AMD



Are current AMD Zen* platforms vulnerable to Rowhammer?


(C1) DRAM addr. mapping

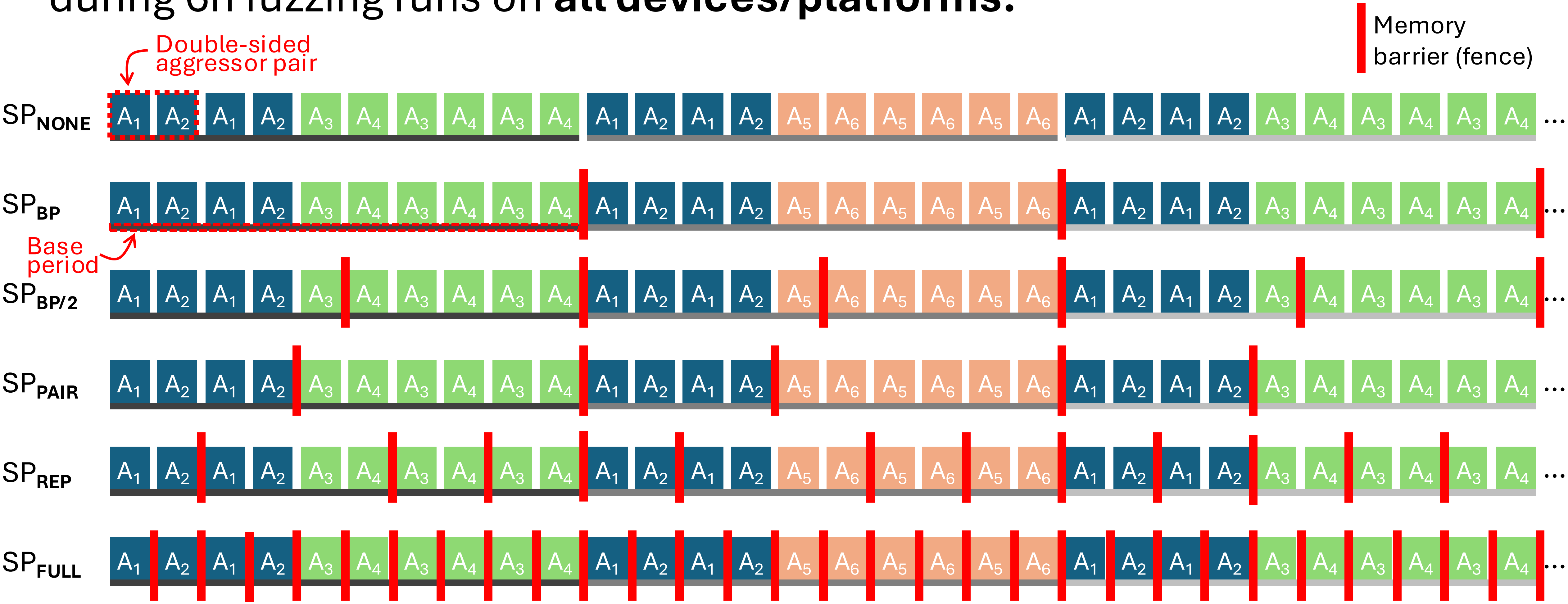
(C2) Timing-based REF synchronization

(C3) ACT rate and memory access order

- We find bit flips on **7/6 DIMMs** on Zen 2/3
- **46x more bit flips** on Zen 3 than on Coffee Lake ⇨ devices are easier exploitable
- First bit flips on one **DDR5 DIMM** on Zen 4

C3 Increasing the ACT rate and preserving order

We designed and evaluated six fence scheduling policies during 6h fuzzing runs on **all devices/platforms.**



Double-sided aggressor pair

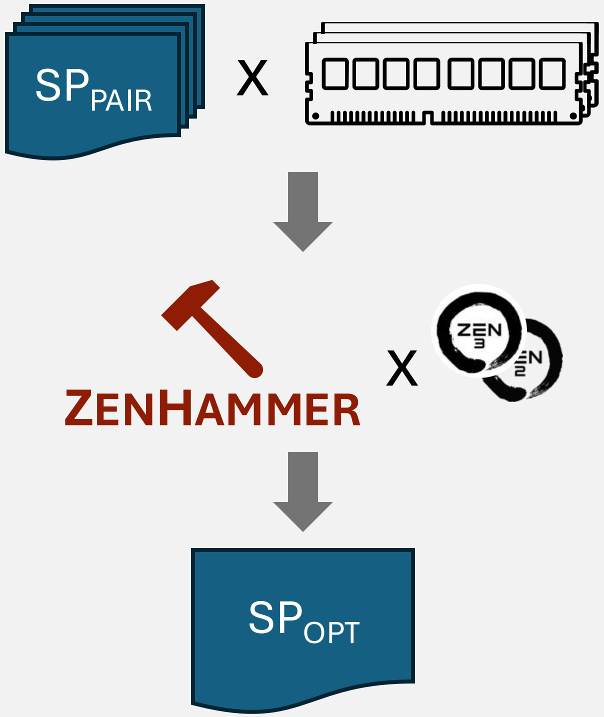
Base period

Memory barrier (fence)

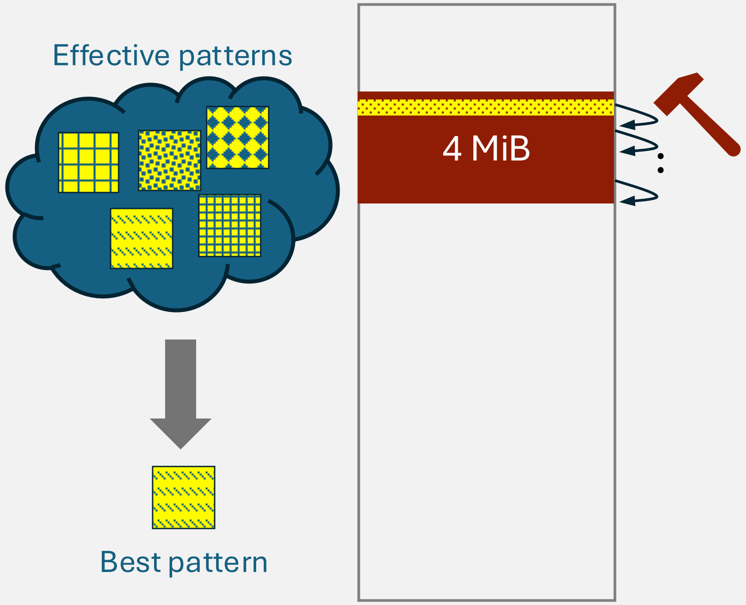
Evaluation: Best Rowhammer pattern

- ZenHammer fuzzing in three stages

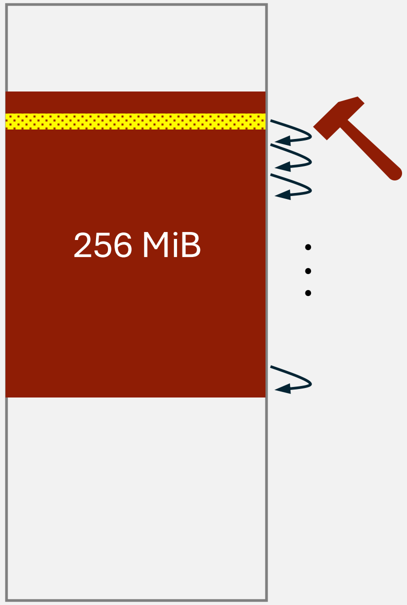
1 Short **fuzzing** with each fence scheduling policy ⇨ **SP$_{OPT}$**



2 Quick **sweep** of each effective pattern ⇨ **best pattern**

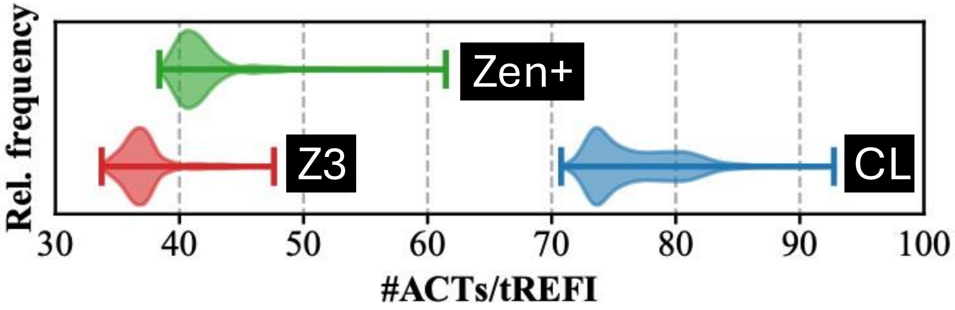


3 Large **sweep** of the best pattern ⇨ **# Bit flips**



C3 Increasing the ACT rate and preserving order

- On average, **ACTs/tREFI** on Z+ (41.9) and Z3 (37.2) are **halved** compared to CL (76.8).
  - 40 ACTs/tREFI gives HC of 36K for n=18 ⇨ too low for many devices.



- **Systematic testing** of different hammering instruction sequences:
  - Cache flushing R1 R2 (e.g., CLFLUSH vs **CLFLUSHOPT**, gather vs **scatter**)
  - Memory barriers (e.g., mfence, lfence, sfence)
  - Access types R3 (e.g., **load** vs store)
  - Vector instructions (e.g., vpgatherdd)

Recommendations

O4. Memory loads following a CLFLUSH(OPT) never incur cache hits on Zen 3 but on Zen+/2.

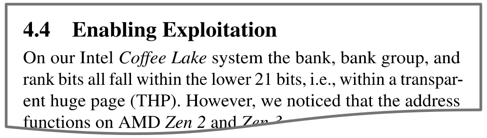
C3 Increasing the ACT rate and preserving order

- We designed six **fence scheduling** policies and evaluated them during 6h fuzzing runs on **all devices/platforms.**

Stronger ordering

Policy	Fencing Frequency	Pattern-Aware	Cache-Avoiding	**Optimal Policy SP$_{OPT}$**
SP$_{NONE}$	No fences	✗	✗	
SP$_{BP}$	Every base period	✓	✗	
SP$_{BP/2}$	Every half base period	✓	✗	
SP$_{PAIR}$	Between different aggressor pairs	✓	✗	⇨ Zen 2 (75%), Zen 3 (43%)
SP$_{REP}$	Between aggressor pair repetitions	✓	✓	
SP$_{FULL}$	Every access *(Blacksmith default)*	✗	✓	⇨ Coffee Lake (100%)

Check out our paper
for details on the PoC!



4.4 **Enabling Exploitation**

On our Intel *Coffee Lake* system the bank, bank group, and rank bits all fall within the lower 21 bits, i.e., within a transparent huge page (THP). However, we noticed that the address functions on AMD *Zen 2* and *Zen 3*

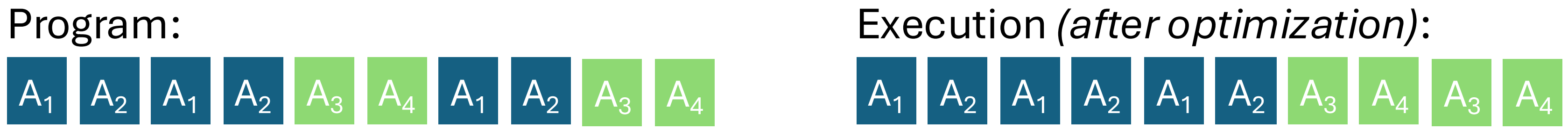
Evaluation: Exploitation

- We simulate attacks using Hammertime: page table flipping (**PTE**;+PoC), flip feng shui (**RSA-2048**), sudo binary (**sudo**).
- High number of bit flips **significantly reduces** the time for exploitation and increases the number of exploitable devices.

	PTE						RSA-2048						sudo					
	Zen 2		Zen3		Coffee Lake		Zen 2		Zen3		Coffee Lake		Zen 2		Zen3		Coffee Lake	
DIMM	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time
S_0	7	6m 4s	7	2m 55s	3	15s	17	2m 47s	37	46s	14	1m 36s	–	–	4	3m 13s	1	23m 49s
S_1	90	9s	1’474	2s	846	**2s**	6	2m 2s	27	30s	21	26s	–	–	1	6m 50s	1	**1m 20s**
S_2	641	21s	5’326	**1s**	126	11s	30	2m 16s	170	**6s**	6	1m 59s	–	–	12	**1m 17s**	–	–
S_3	142	9s	61	32s	–	–	7	2m 21s	–	–	–	–	–	–	–	–	–	–
S_4	220	28s	3	23m 52s	2’658	1s	7	12m 29s	1	23m 52s	53	26s	–	–	–	–	4	5m 16s
S_5	102	**6s**	625	2s	330	4s	6	**1m 14s**	28	33s	11	**5s**	–	–	2	5m 58s	3	2m 34s
H_0	11	53s	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Median		**21s**		**17s**		**4s**		**2m 19s**		**33s**		**1m 5s**				**4m 36s**		**3m 55s**

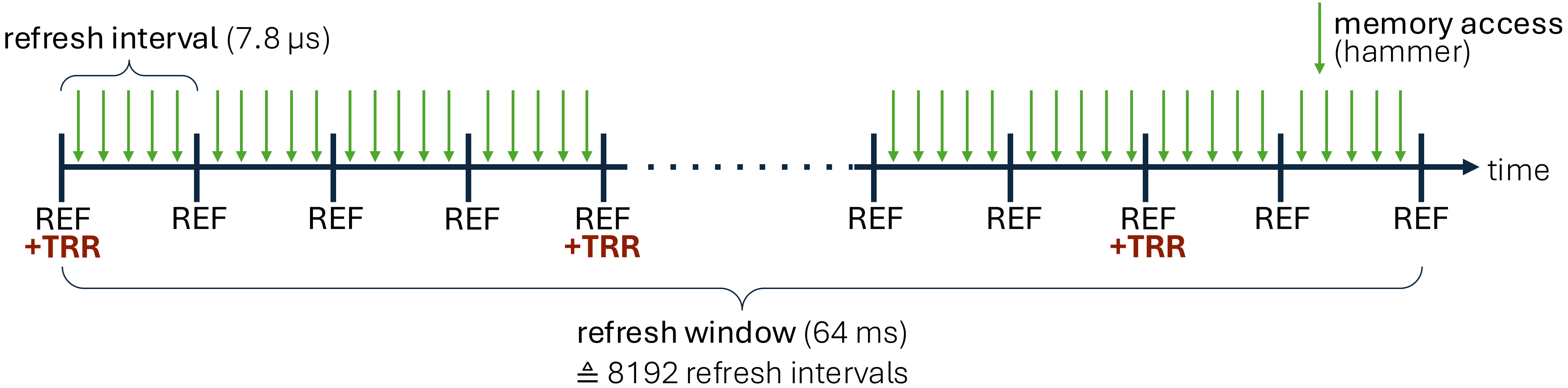
In-DRAM TRR: Order of accesses and ACT rate

- **Issue**: the memory controller reorders accesses
  ⇨ enforce order by adding memory fences: which fence? where?



Program:
A_1 A_2 A_1 A_2 A_3 A_4 A_1 A_2 A_3 A_4
Execution *(after optimization)*:
A_1 A_2 A_1 A_2 A_1 A_2 A_3 A_4 A_3 A_4

- **Issue**: too few ACTs due to hammering ”too slowly”
  ⇨ the ACT rate should be **maximized** to make bit flips more likely



Conclusion

- AMD Zen-based systems are equally **vulnerable to Rowhammer** as Intel systems.
- We disclose the secret **DRAM mappings** for AMD Zen-based systems including their address offsets.
- We found bit flips on **7 DIMMs** (Zen 2) and **6 DIMMs** (Zen 3) compared to 8 DIMMs on Intel Coffee Lake.
- We show **46x more** bit flips on Zen 3 than on Coffee Lake ⇨ devices are easier exploitable
- In the best case, we only need **6s** (Zen 2) and **2s** (Zen 3) to mount an attack (PTE).

Check out our paper for more information!