TU Graz

Hochschule Hof
University of Applied Sciences

Real-World Rowhammer: Understanding and Addressing the Challenges to Rowhammer Attacks

Martin Heckel

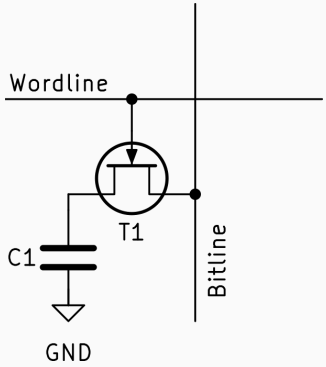
April 01, 2026

Background

DRAM Cells

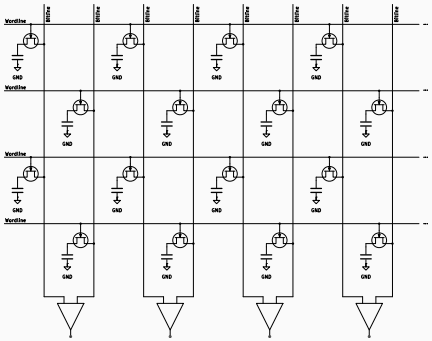
- A single cell consists of:
  - Capacitor C_1: Store data as electric charge
  - Transistor T_1: Control access to the capacitor
- Capacitor looses chage over time

$\Rightarrow$ Refresh cells periodically (64 ms according to standard [2])



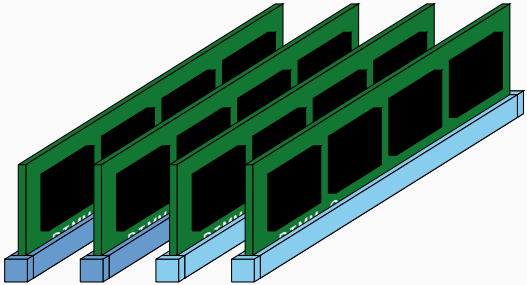
DRAM Array

- Cells are organized in an array
- Wordlines connect rows of cells (only entire rows can be enabled)
- Bitlines connect columns of cells (shared between rows)

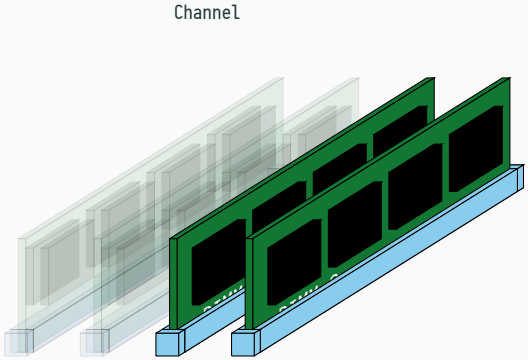


Physical Architecture of DRAM

System DRAM

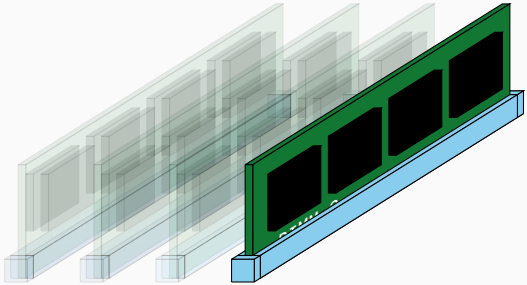


Physical Architecture of DRAM

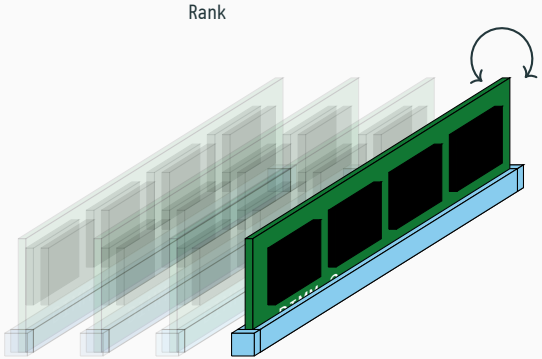


Physical Architecture of DRAM

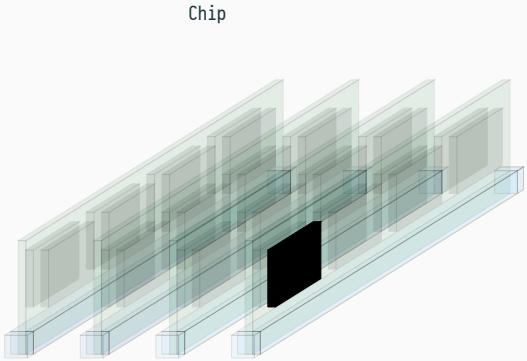
DIMM



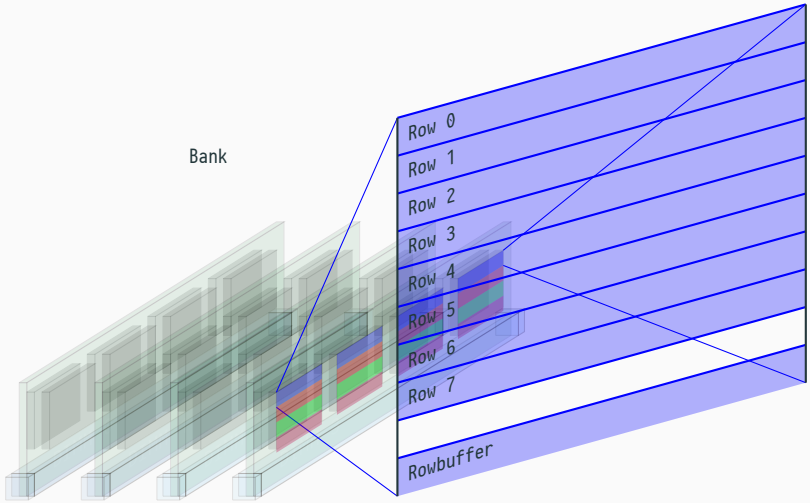
Physical Architecture of DRAM



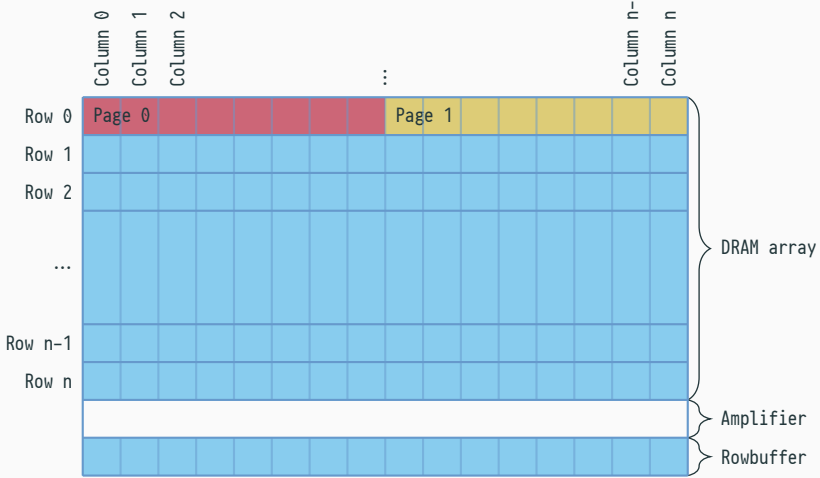
Physical Architecture of DRAM



Physical Architecture of DRAM



Structure within a DRAM Bank



DRAM Addressing

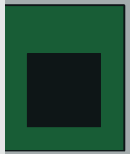
- Data is stored in physical memory:
  - Channel
  - DIMM
  - Rank
  - Bank
  - Row
  - Column
- The Memory Controller translates physical addresses to memory locations

DRAM Addressing

- Data is stored in physical memory:
  - C
  - D
  - R
  - B
  - R
  - C
- The M
  addresses to memory locations

decode-dimms

```
---=== Memory Characteristics ===---
Maximum module speed                    1333 MT/s (PC3-10600)
Size                                    4096 MB
Banks x Rows x Columns x Bits           8 x 15 x 10 x 64
Ranks                                   2
```



DRAM Addressing Functions

Physical Address (32 bit, $2^{32} = 4$ GiB)

DRAM Addressing Functions

Physical Address (32 bit, $2^{32} = 4$ GiB)

$2^{15} = 32\,768$ Rows

DRAM Addressing Functions

Physical Address (32 bit, $2^{32} = 4\,\text{GiB}$)

$2^{15} = 32\,768$ Rows

$2^{10} = 1024$ Columns

DRAM Addressing Functions

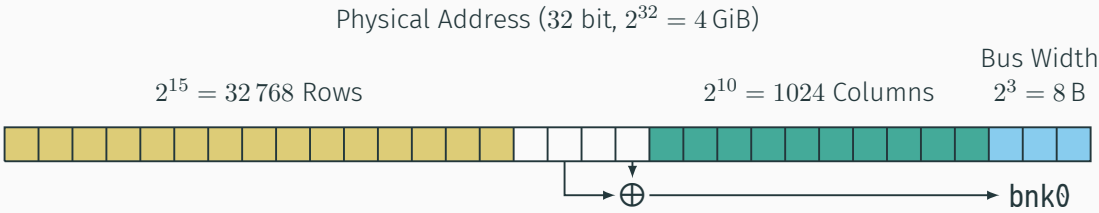
Physical Address (32 bit, $2^{32} = 4\,\text{GiB}$)

$2^{15} = 32\,768$ Rows

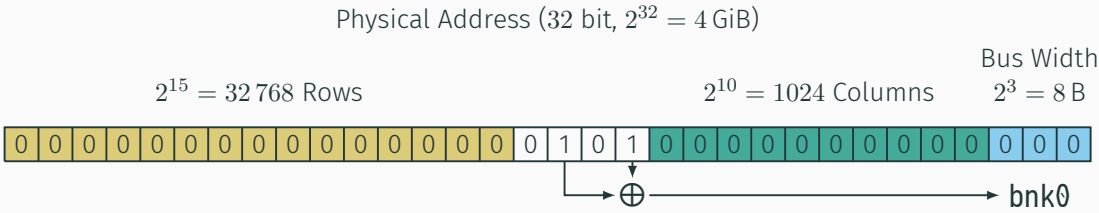
$2^{10} = 1024$ Columns

Bus Width
$2^3 = 8\,\text{B}$

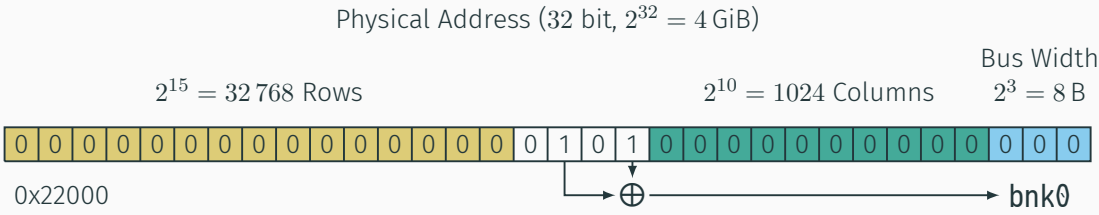
DRAM Addressing Functions



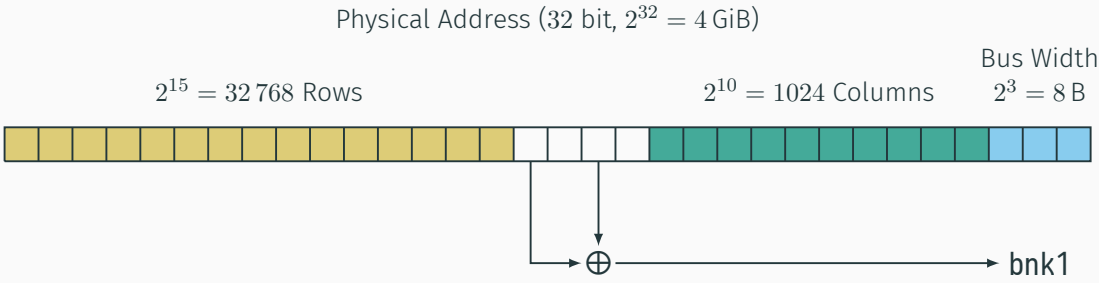
DRAM Addressing Functions



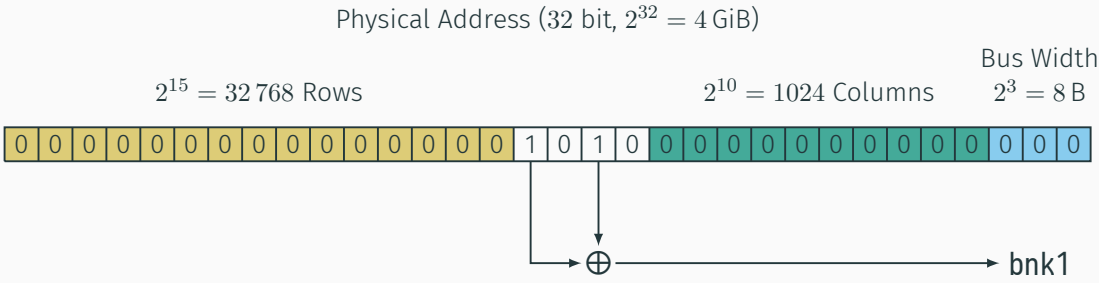
DRAM Addressing Functions



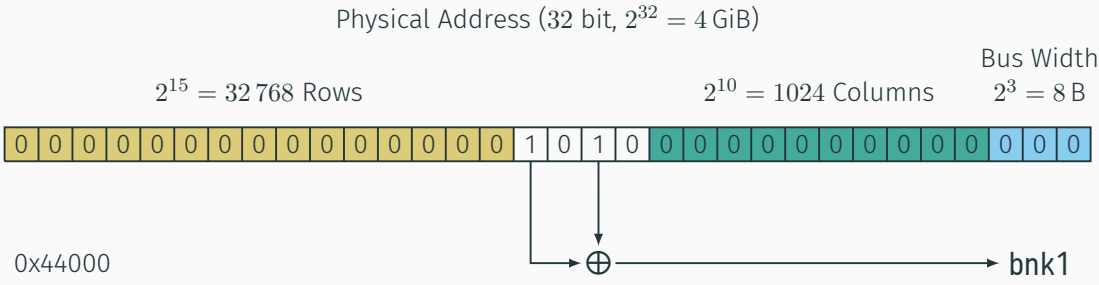
DRAM Addressing Functions



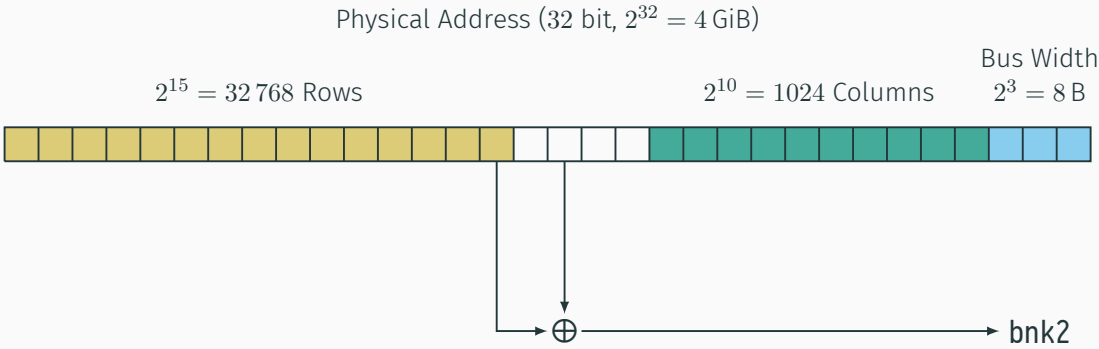
DRAM Addressing Functions



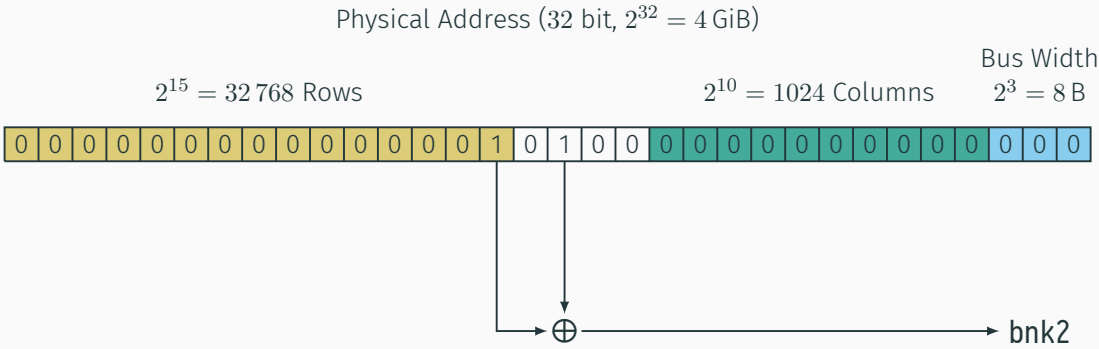
DRAM Addressing Functions



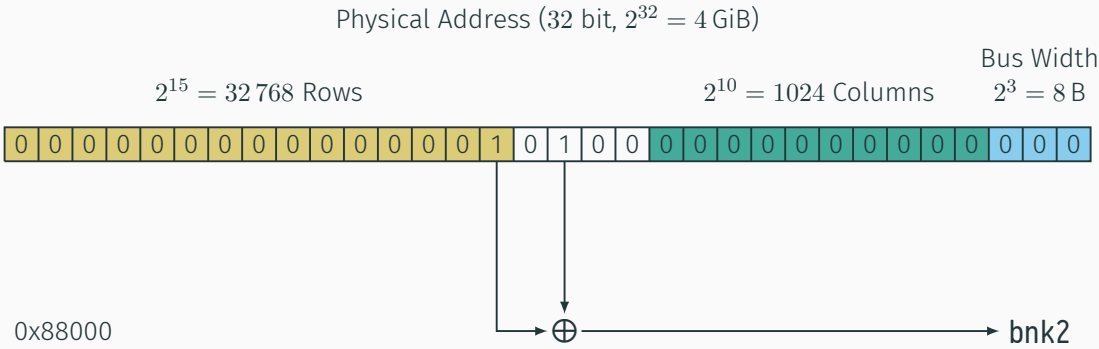
DRAM Addressing Functions



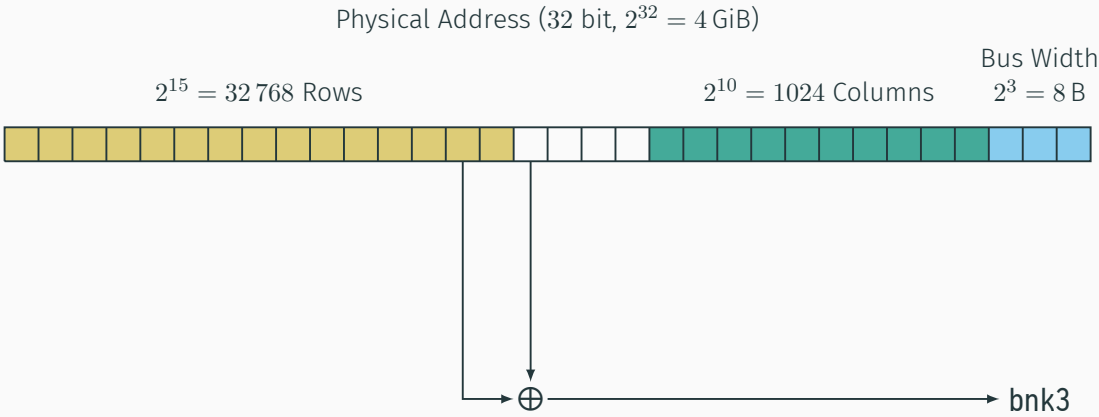
DRAM Addressing Functions



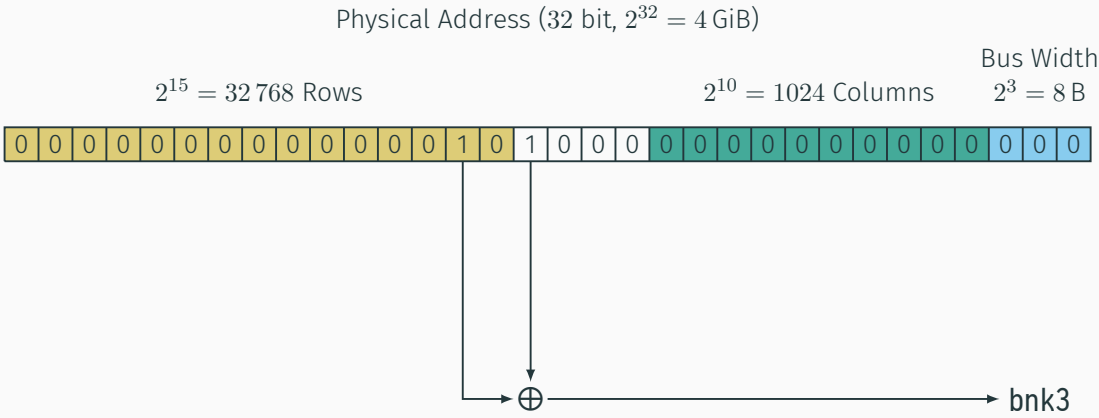
DRAM Addressing Functions



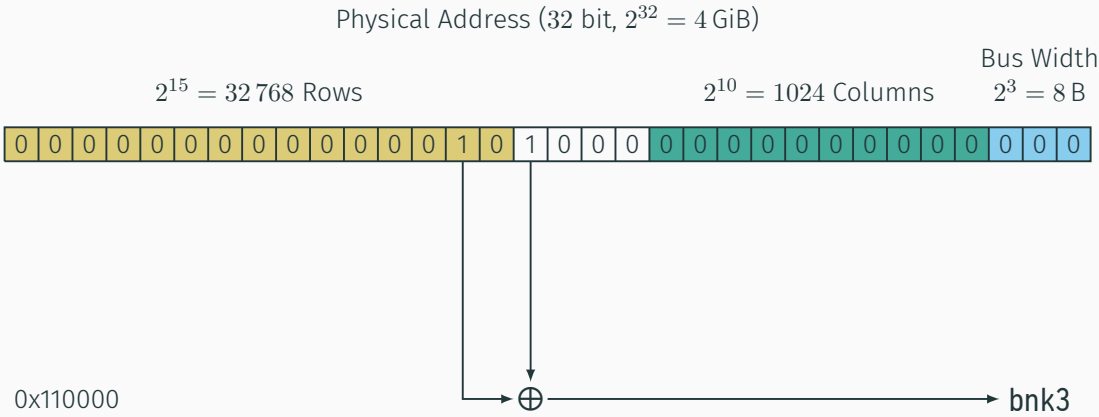
DRAM Addressing Functions



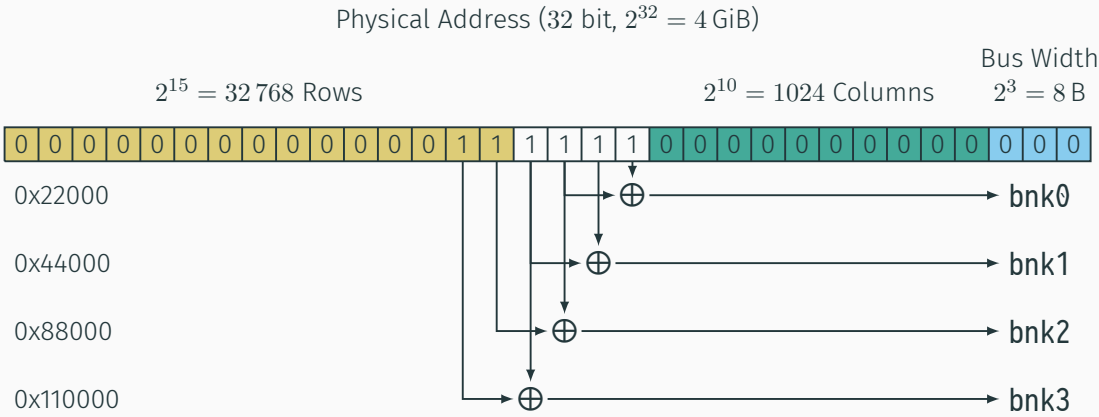
DRAM Addressing Functions



DRAM Addressing Functions

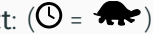


DRAM Addressing Functions



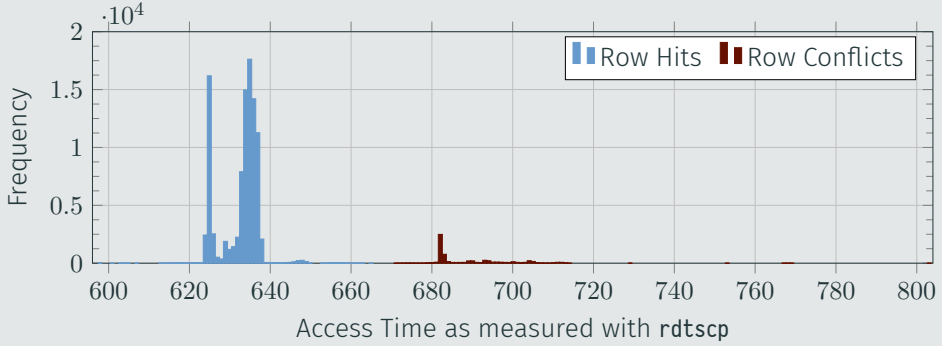
DRAM Access Times

- Row buffer is shared among all rows in one bank
- Reading a row from the DRAM array is destructive
- Accessessing different rows at the same bank

  $\Rightarrow$ **Row Conflict**: ( =)
- Accessessing the same row at the same bank

  $\Rightarrow$ **Row Hit**: ( =)

DRAM Access Times



Grouping Addresses based on DRAM Access Times

- Find threshold (⏲) between Row Hits and Row Conflicts
- Access addresses alternatingly and compare access time (⏱) to the threshold (⏲)
  - ⏱ < ⏲ ⇒  ⇒ Row Hit
  - ⏱ > ⏲ ⇒  ⇒ Row Conflict
- **Idea**: Group addresses with Row Conflicts (same bank) together

DRAM Addressing Function Reverse-Engineering

- **Idea**: brute-force addressing functions to match groups
- Solved for linear DRAM addressing functions by Pessl et al. [5]
- Not solved for nonlinear DRAM addressing functions

- Idea
- Solv
- Not

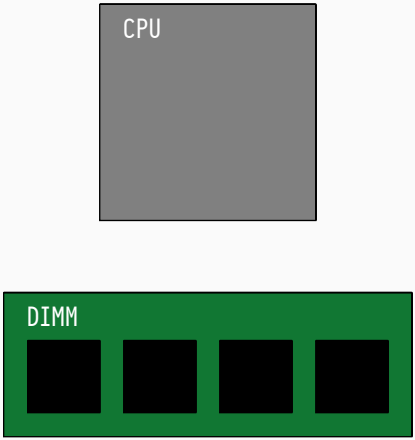
What about Rowhammer?

Rowhammer

- Fault Attack on DRAM
- Flipping bits in memory without accessing them

Rowhammer

```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```

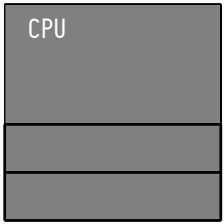


Source code from Kim et al. [4]

Rowhammer

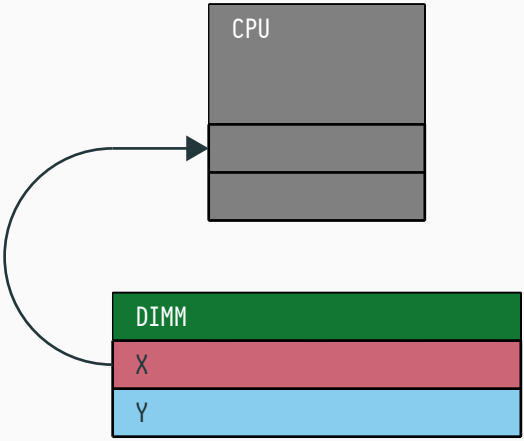
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```

Source code from Kim et al. [4]



Rowhammer

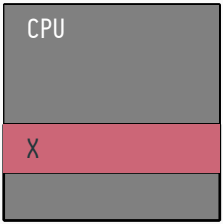
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

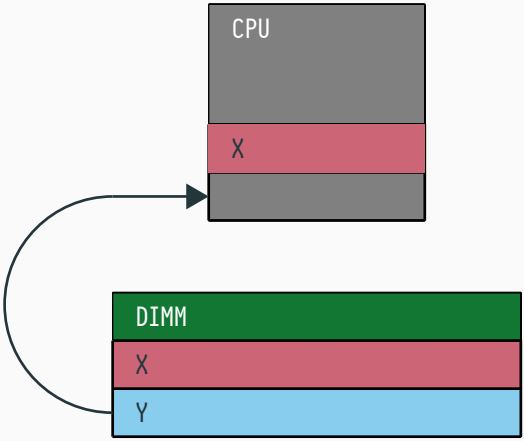
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

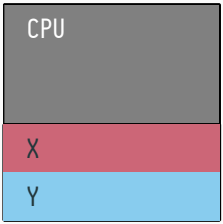
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

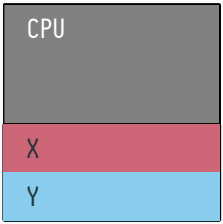
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

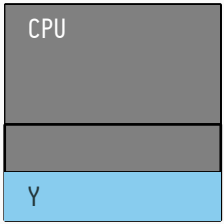
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

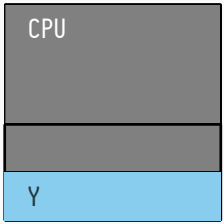
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

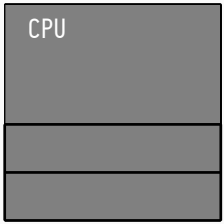
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

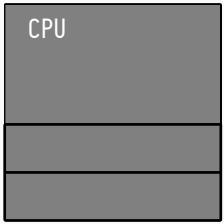
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

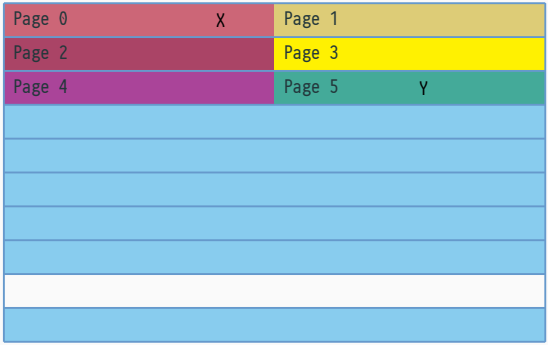
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

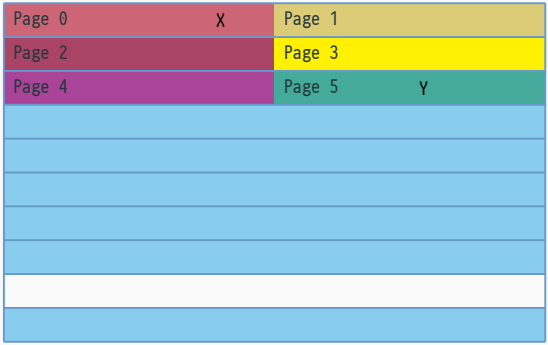
```
1  hammer:
2        mov eax, X
3        mov ebx, Y
4        clflush X
5        clflush Y
6        jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

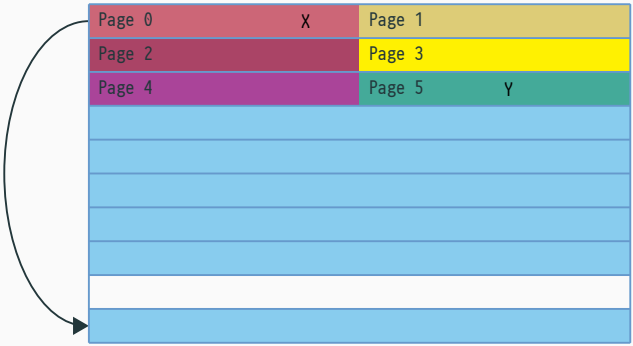
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```

Source code from Kim et al. [4]

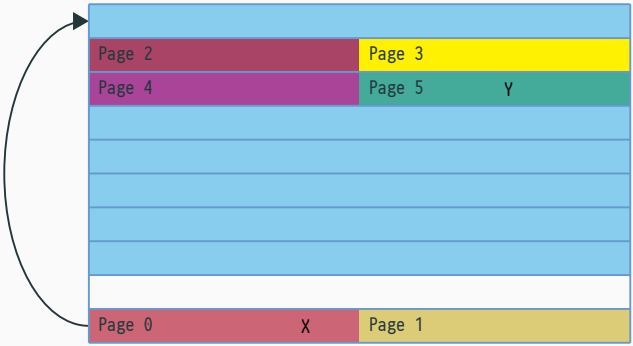
Rowhammer

```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```

Source code from Kim et al. [4]

Rowhammer

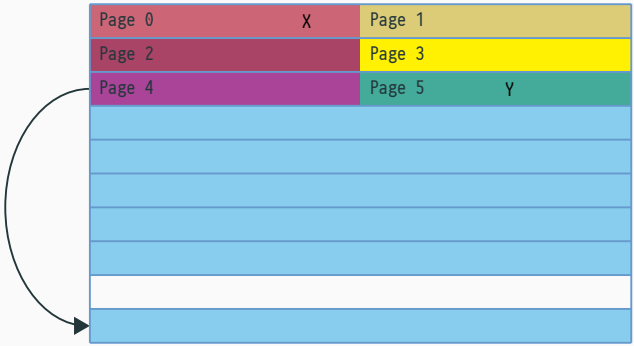
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

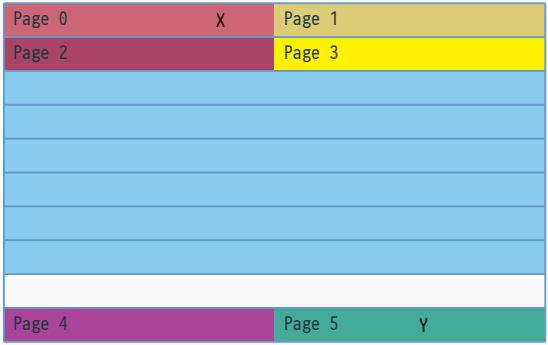
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

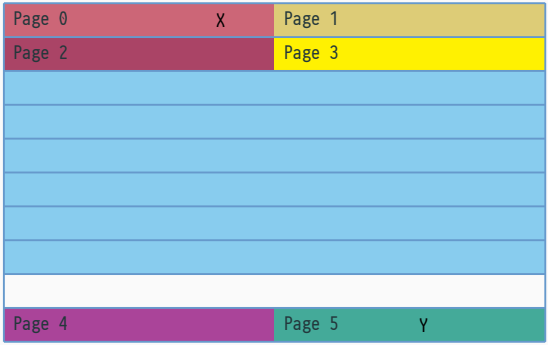
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

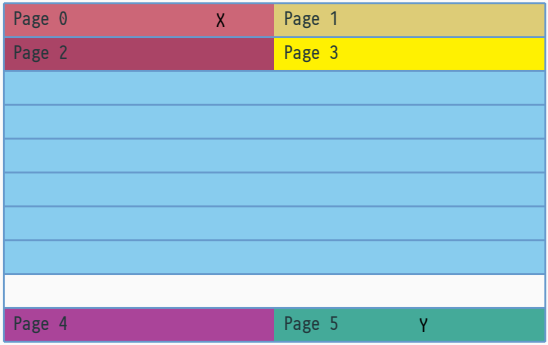
```
1  hammer:
2      mov eax, X
3      mov ebx, Y
4      clflush X
5      clflush Y
6      jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

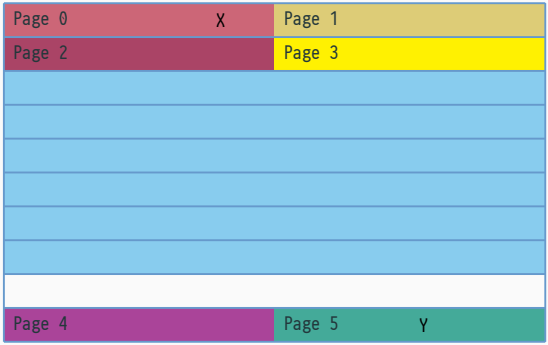
```
1   hammer:
2       mov eax, X
3       mov ebx, Y
4       clflush X
5       clflush Y
6       jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

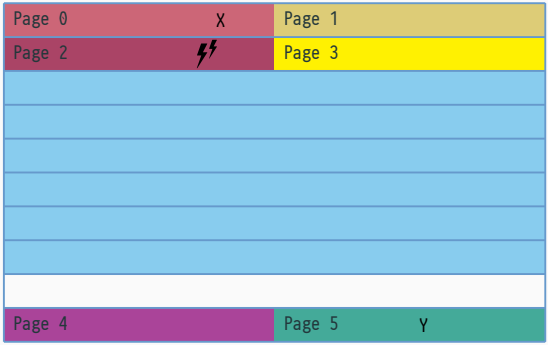
```
1   hammer:
2       mov eax, X
3       mov ebx, Y
4       clflush X
5       clflush Y
6       jmp hammer
7
```



Source code from Kim et al. [4]

Rowhammer

```
1   hammer:
2       mov eax, X
3       mov ebx, Y
4       clflush X
5       clflush Y
6       jmp hammer
7
```



Source code from Kim et al. [4]

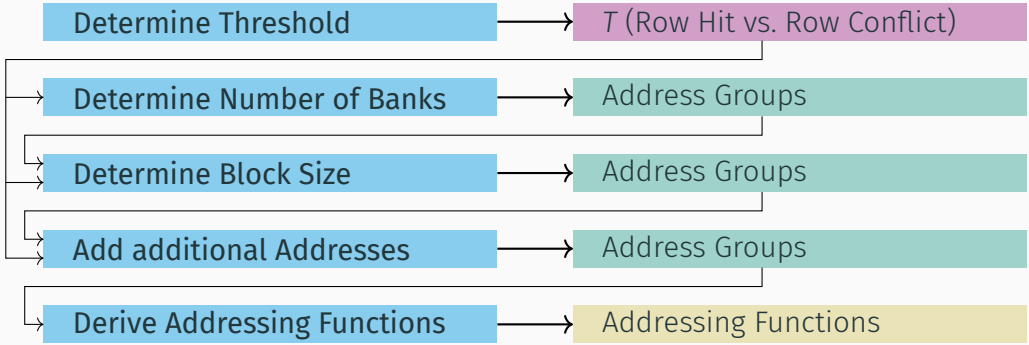
Peer-Reviewed Publications

Reverse-Engineering Bank Addressing Functions on AMD CPUs

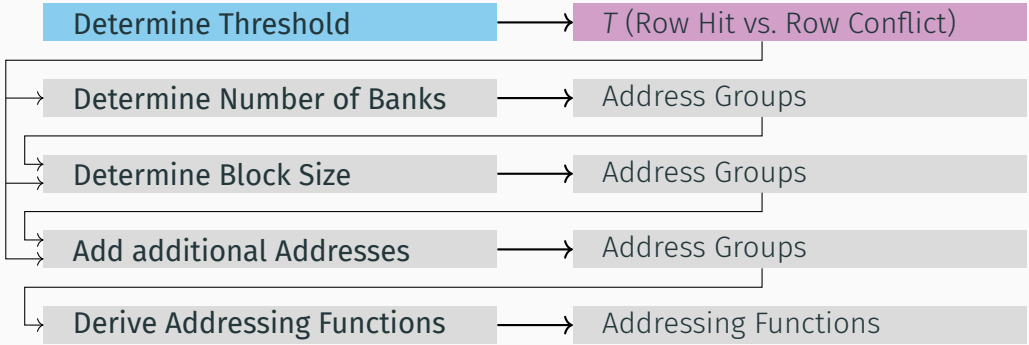
Martin Heckel and Florian Adamsky

DRAMSec, 2023

Overview of the Reverse-Engineering Process

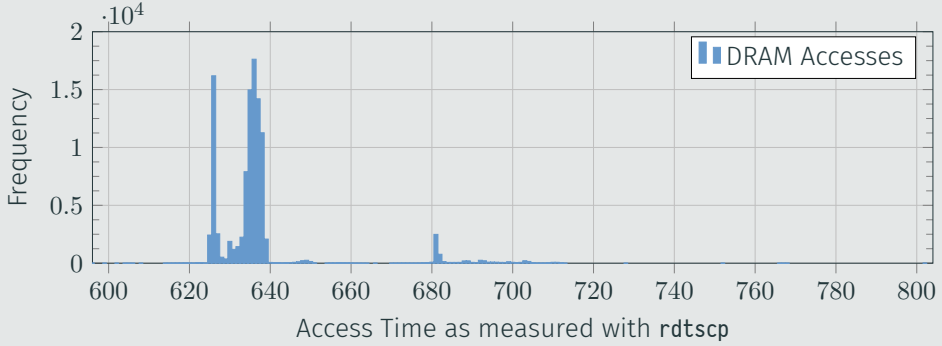


Determine Threshold between Row Hit and Row Conflict



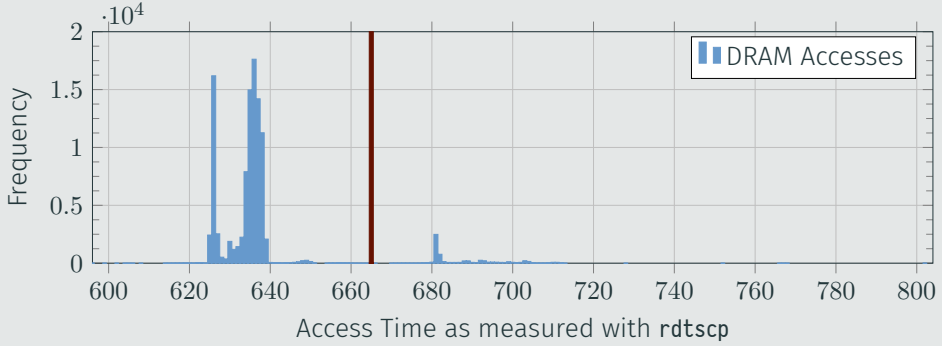
Determine Threshold between Row Hit and Row Conflict

Remember?

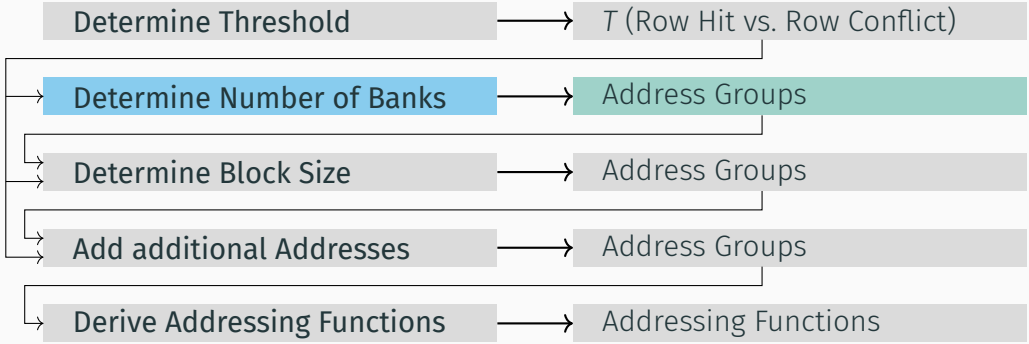


Determine Threshold between Row Hit and Row Conflict

Remember?

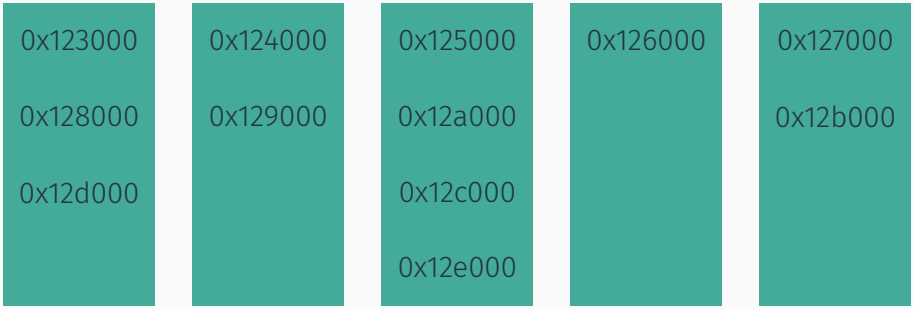


Determine Number of Banks

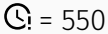


Determine Number of Banks

⏲ = 550



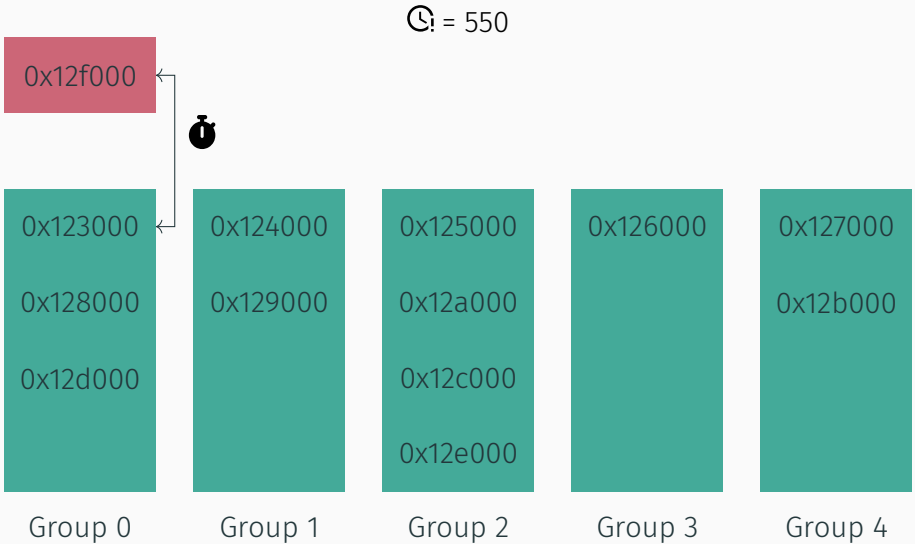
Determine Number of Banks



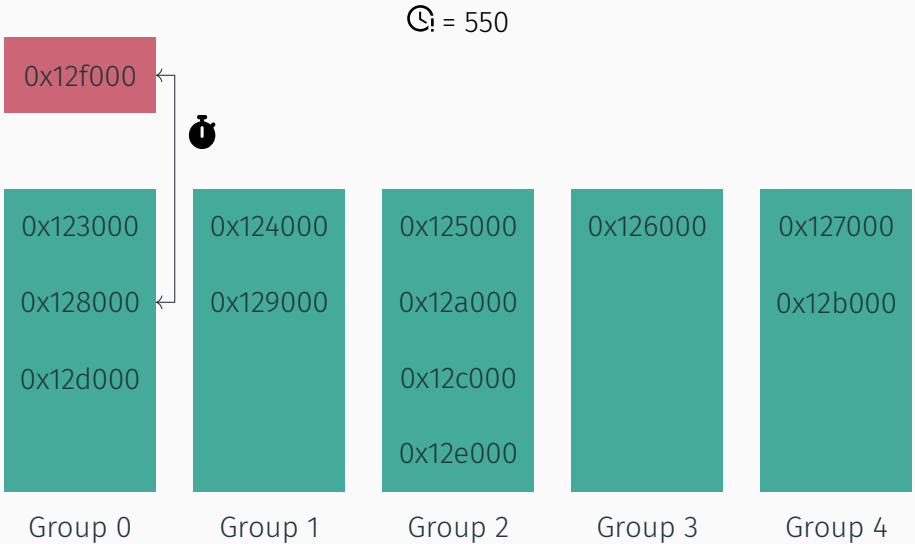
0x12f000

Group 0	Group 1	Group 2	Group 3	Group 4
0x123000	0x124000	0x125000	0x126000	0x127000
0x128000	0x129000	0x12a000		0x12b000
0x12d000		0x12c000		
		0x12e000		

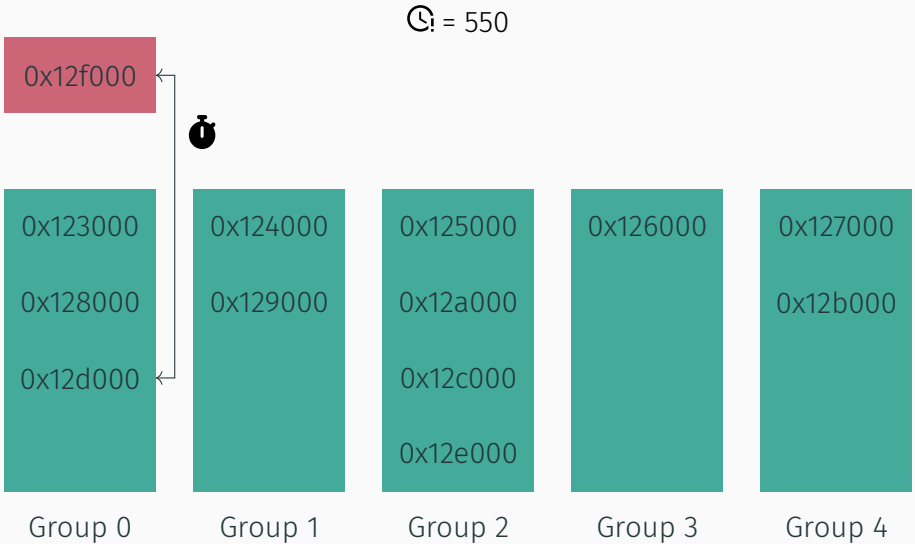
Determine Number of Banks



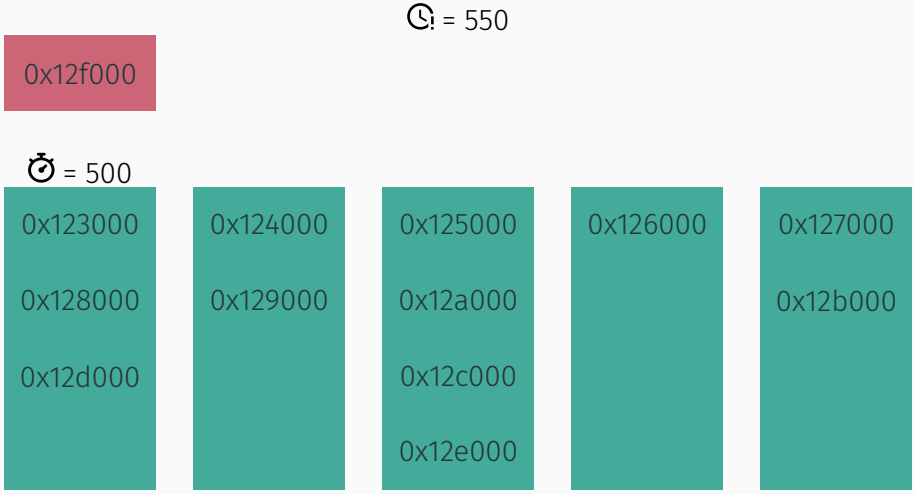
Determine Number of Banks



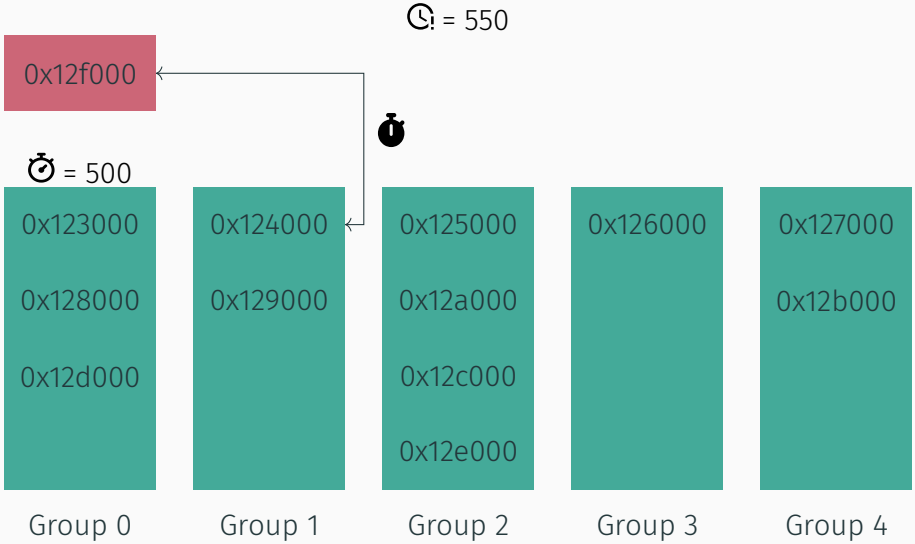
Determine Number of Banks



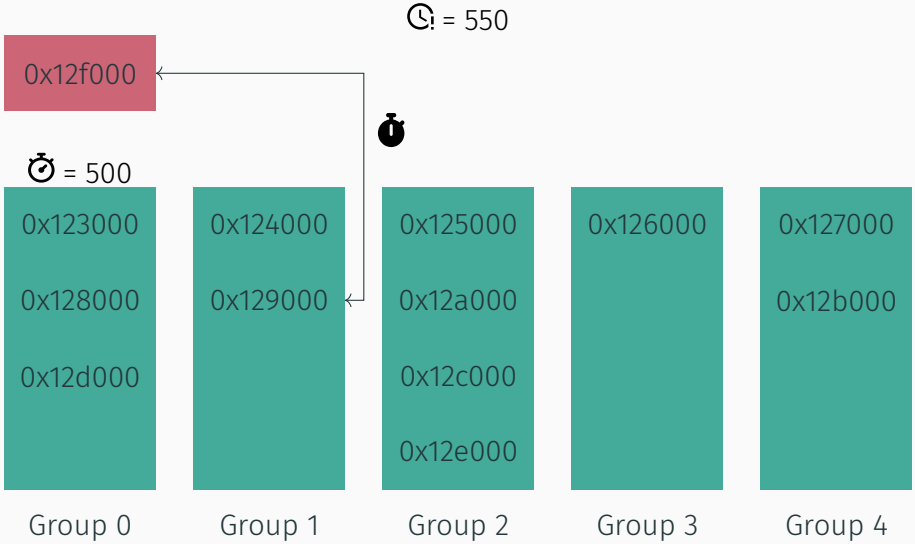
Determine Number of Banks



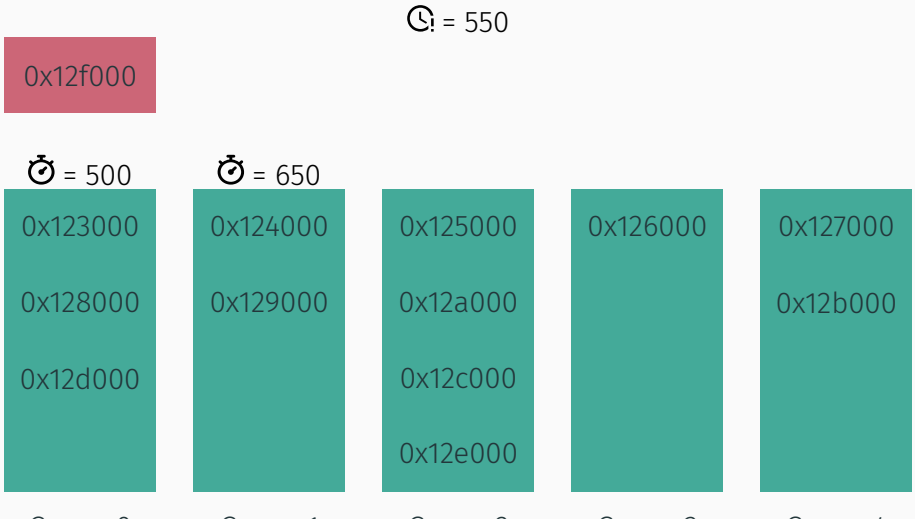
Determine Number of Banks



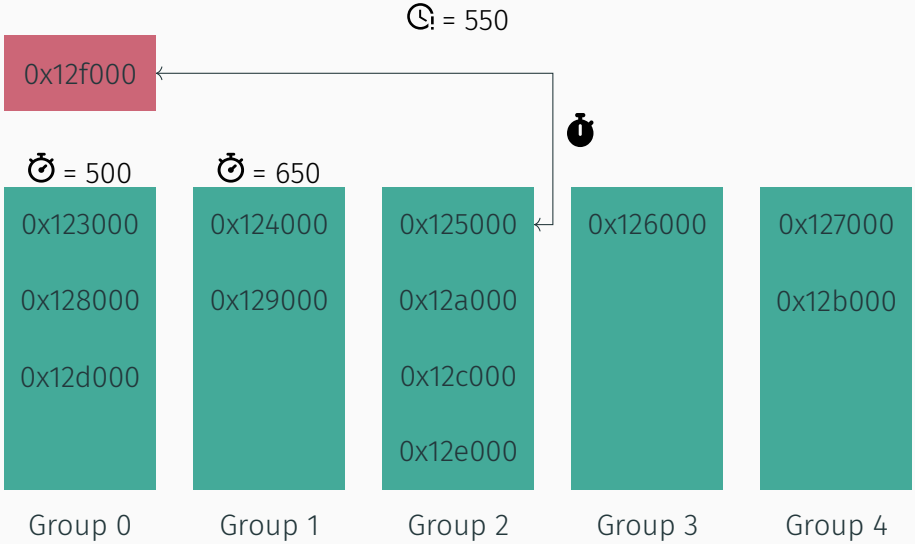
Determine Number of Banks



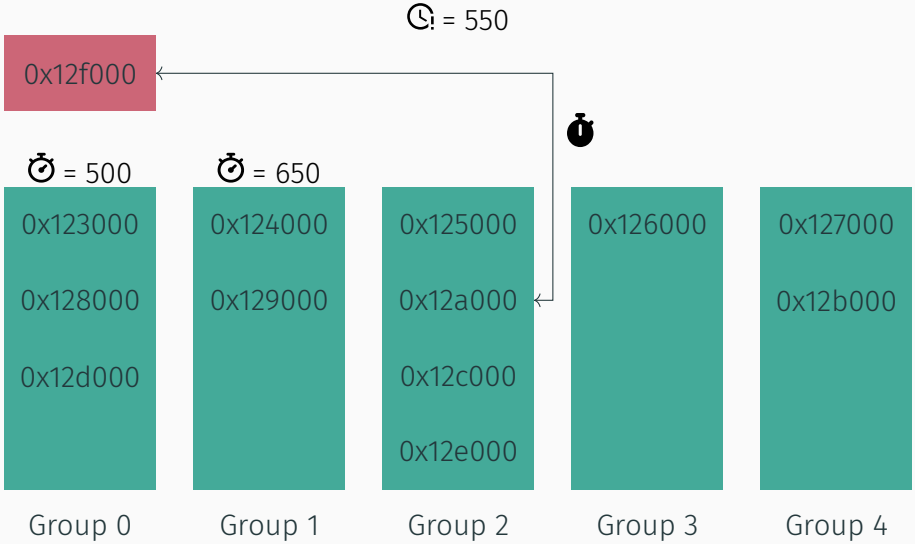
Determine Number of Banks



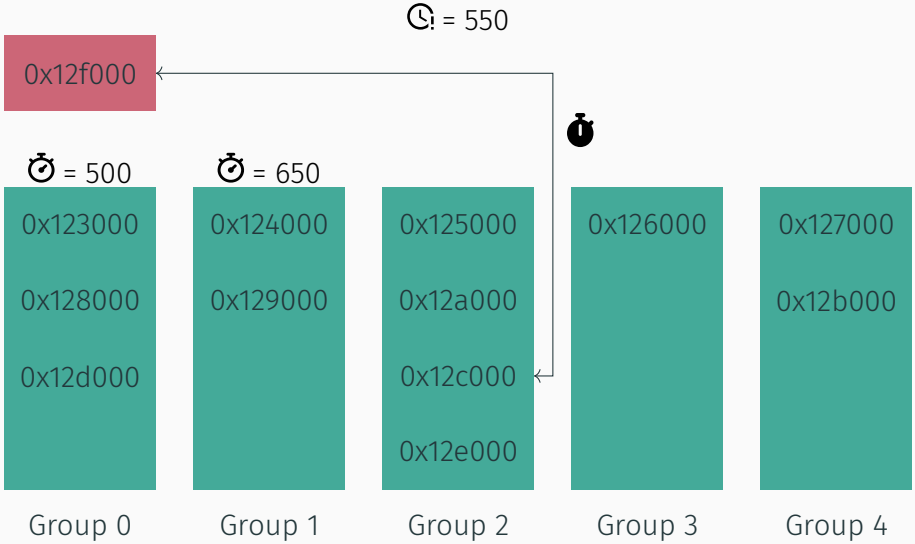
Determine Number of Banks



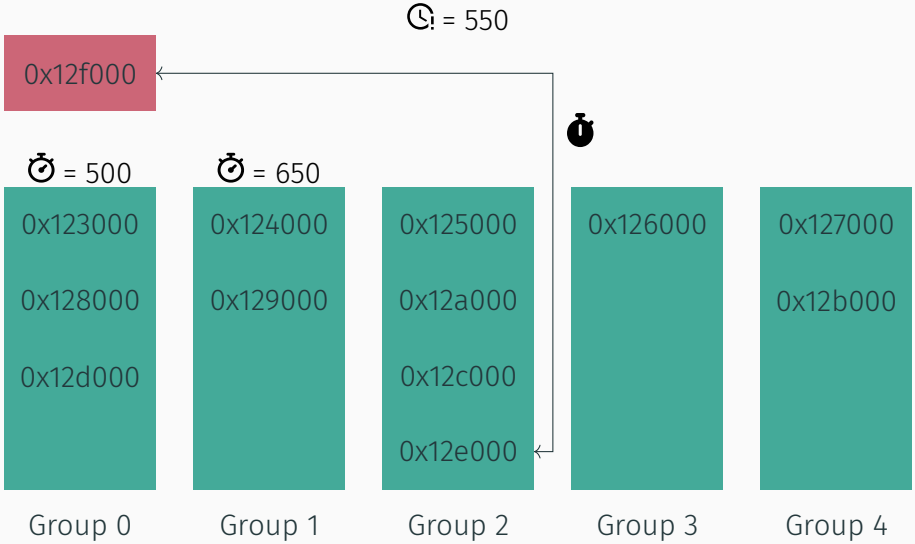
Determine Number of Banks



Determine Number of Banks



Determine Number of Banks

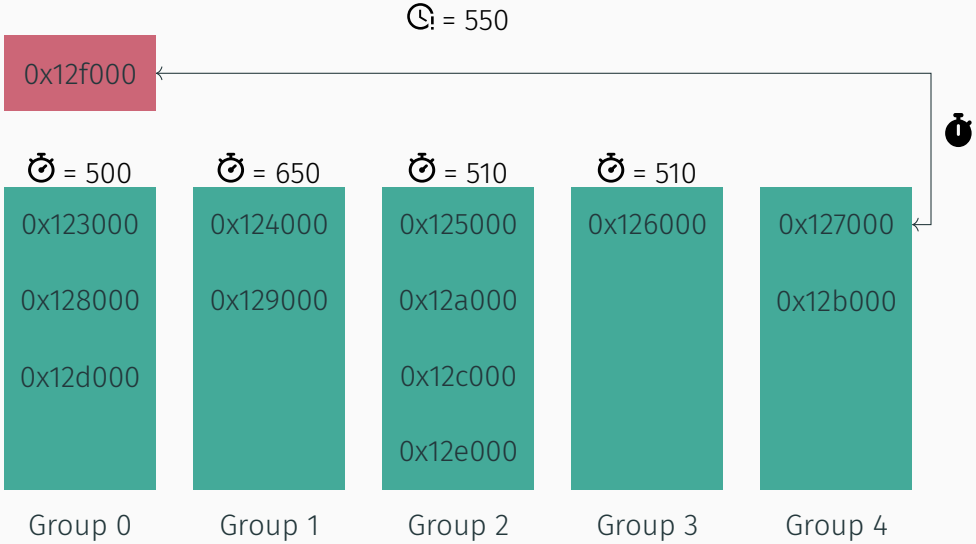


Determine Number of Banks

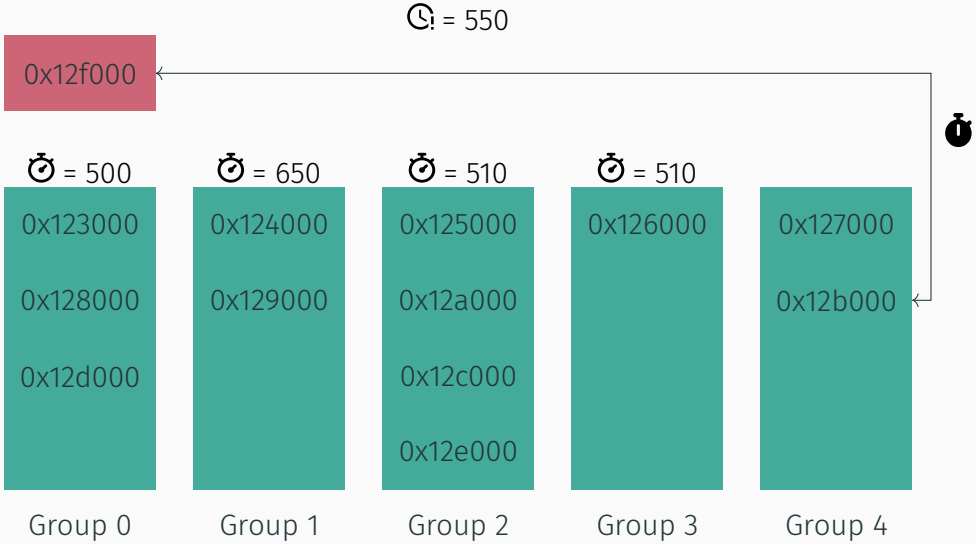
Determine Number of Banks

Determine Number of Banks

Determine Number of Banks



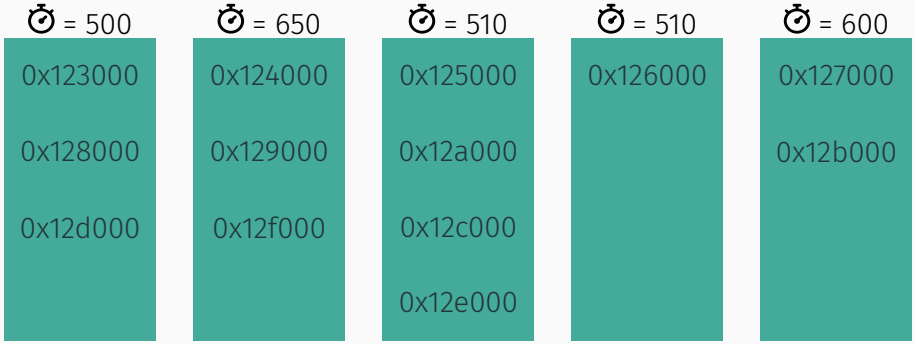
Determine Number of Banks



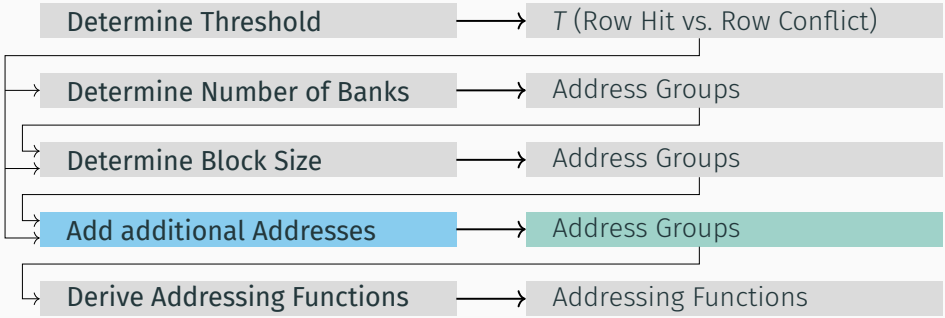
Determine Number of Banks

Determine Number of Banks

= 550



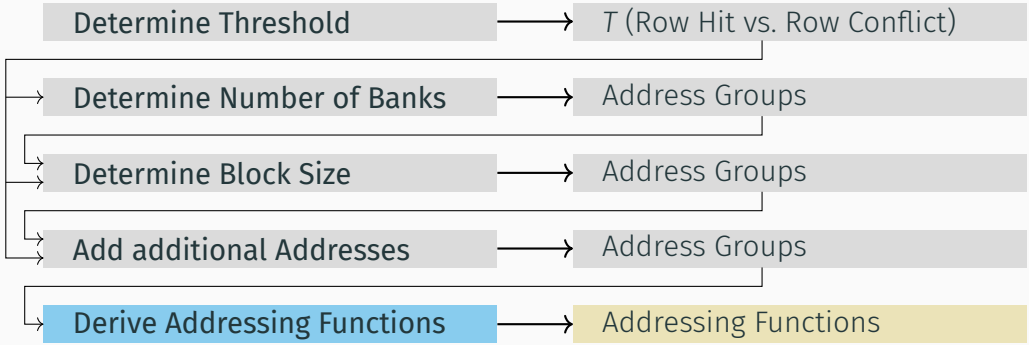
Add additional Addresses



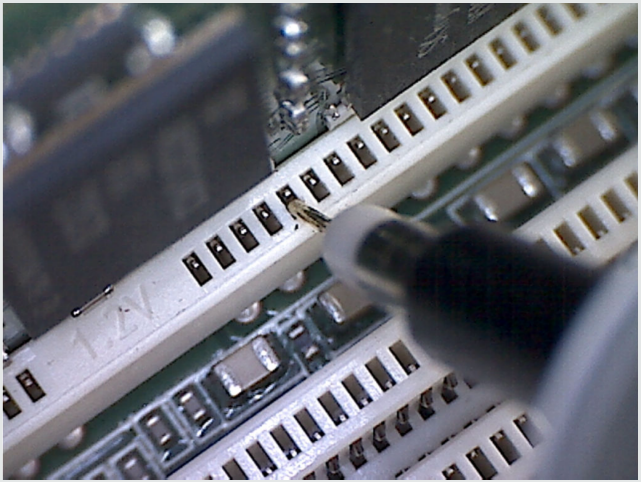
Add additional Addresses

- A number of transparent hugepages is allocated
- The addresses of the transparent hugepages are added to the existing groups (no new groups are created)

Derive Addressing Functions



Verification with Physical Probing as used by Pessl et al. [5]



Verifying DRAM Addressing in Software

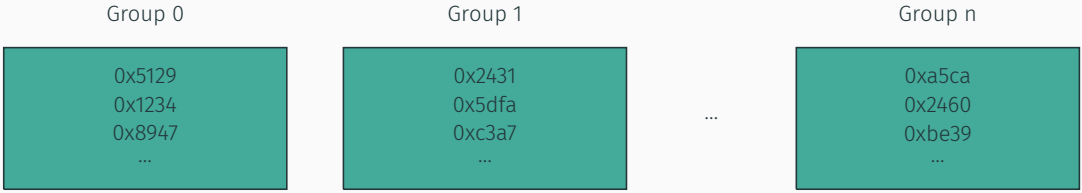
Martin Heckel, Florian Adamsky, Jonas Juffinger, Fabian Rauscher, and Daniel Gruss

ESORICS, 2025

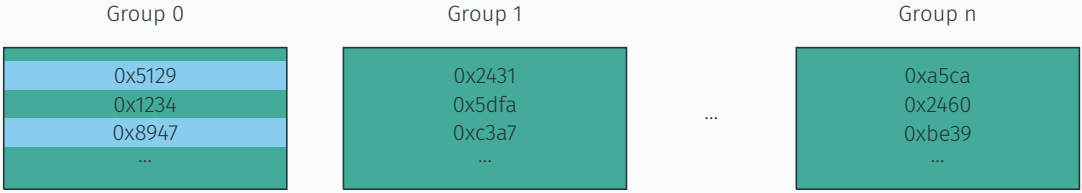
Idea: Assume correct Functions and verify this Assumption

- Group DRAM addresses based on the DRAM addressing functions that should be verified
- If correctly grouped:
  - Row Hits should be measured between randomly selected addresses of different groups
  - Row Conflicts should be measured between randomly selected addresses of the same group

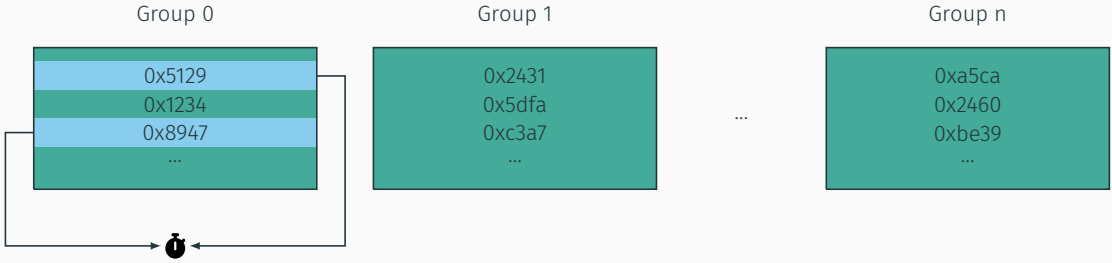
Idea: Assume correct Functions and verify this Assumption



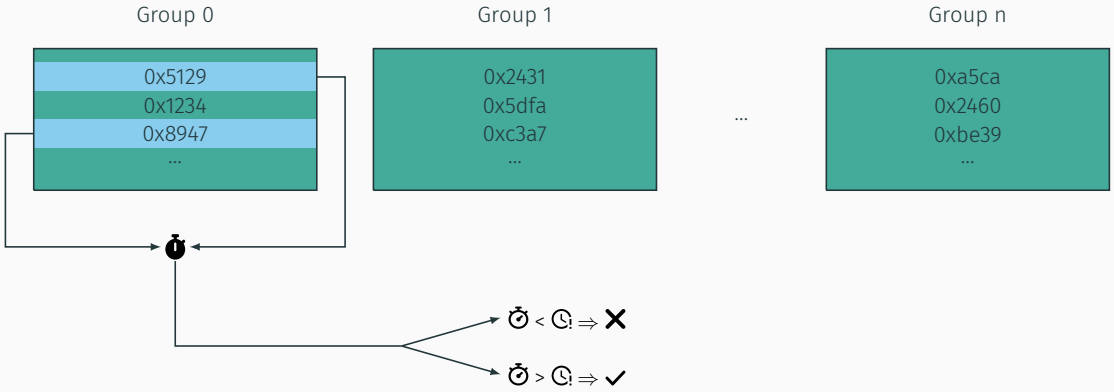
Idea: Assume correct Functions and verify this Assumption



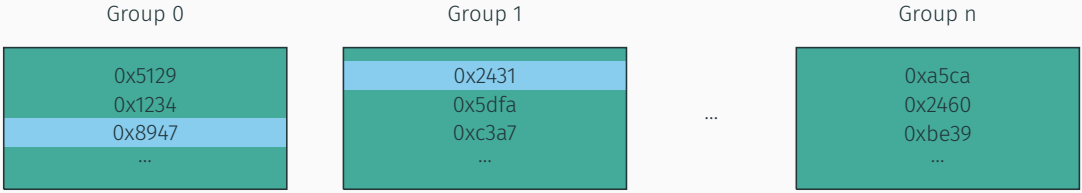
Idea: Assume correct Functions and verify this Assumption



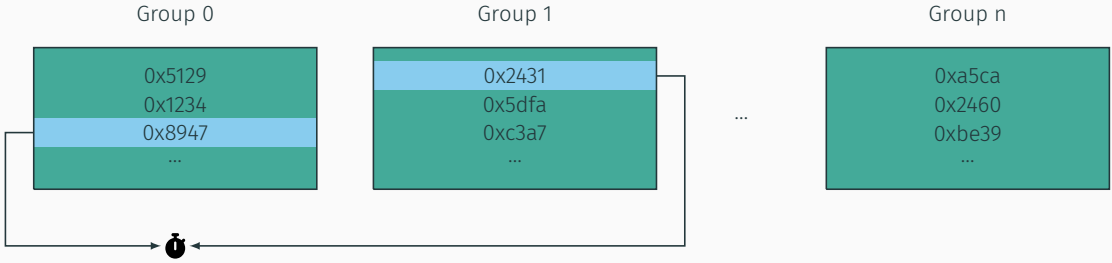
Idea: Assume correct Functions and verify this Assumption



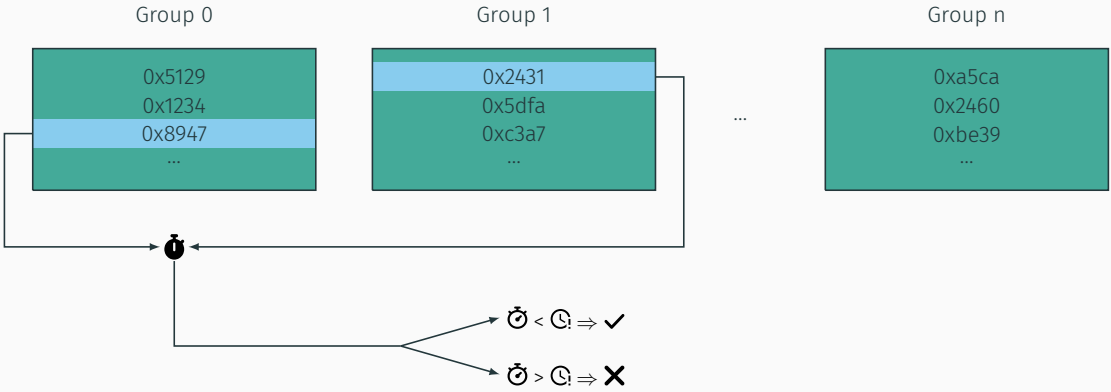
Idea: Assume correct Functions and verify this Assumption



Idea: Assume correct Functions and verify this Assumption



Idea: Assume correct Functions and verify this Assumption

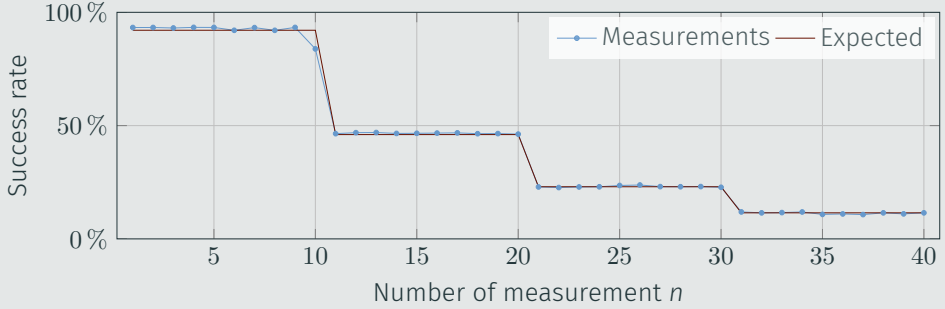


Verify single DRAM Addressing Functions

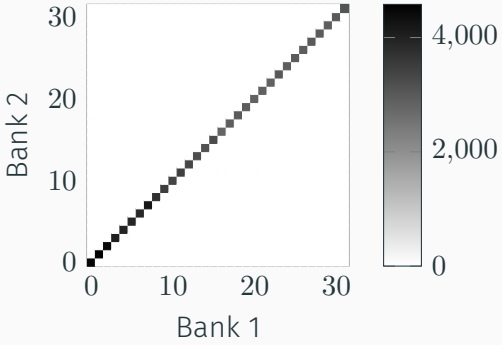
- Each DRAM bank addressing function returns one bit of the DRAM bank
- Remove function
  - $\Rightarrow$ half of the banks can not be addressed anymore
- DRAM bank number should be incorrect for half of the addresses

Verify single DRAM Addressing Functions

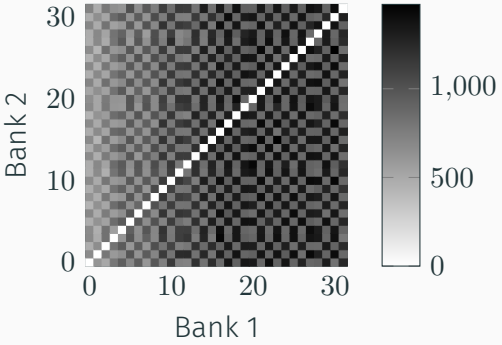
Overall Correctness when removing Functions



Number of wrongly classified DRAM banks



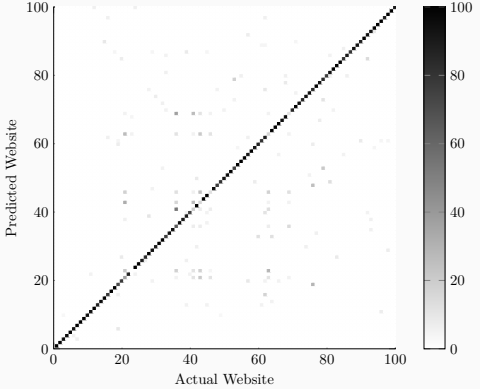
(a) Too fast Row Conflicts



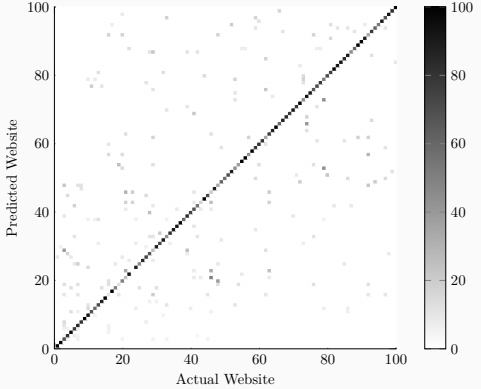
(b) Too slow Row Hits

Website Fingerprinting Attack

- Based on the row hit vs row conflic covert channel shown by Pessl et al. [5]
- We reached an F_1 score of $84\,\%$ on DDR4 and an F_1 score of $74\,\%$ on DDR5



(a) Website confusion matrix on DDR4, with F_1 score $84.0\,\%$.



(b) Website confusion matrix on DDR5, with F_1 score $74.0\,\%$.

- Bas
- We

t al. [5]
DDR5

What about Rowhammer?

(a) Website confusion matrix on DDR4, with F_1 score $84.0\,\%$.

(b) Website confusion matrix on DDR5, with F_1 score $74.0\,\%$.

Flipper

Rowhammer on Steroids

Martin Heckel and Florian Adamsky

uASC, 2025

Additional Memory Accesses

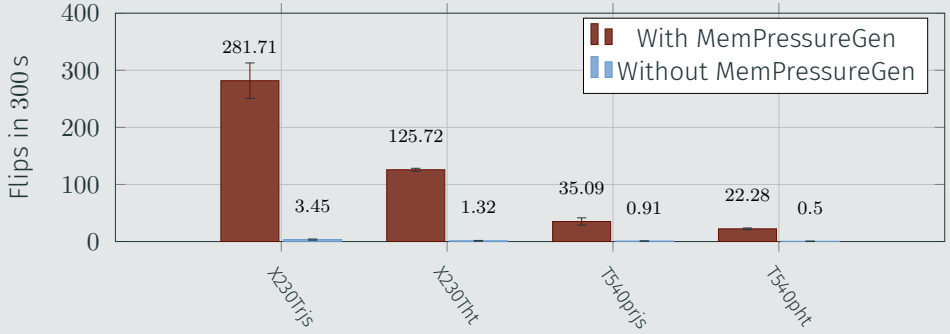
- Utilizing the x86 `cmpsb` instruction used for optimized memory comparison in combination with `repe`
  - Repeats `cmpsb` as long as the `ECX` register is not zero and the `ZF` flag is set
- 2.65 times faster than the “native” implementation using pointers

Additional Memory Accesses – Implementation

- Allocate 1024 MiB of memory
- Initialize all pages in the memory region with the same (randomly generated) content
- Compare the first page to each other page in a loop
- Repeat until the process is terminated

Additional Memory Accesses – Implementation

Results



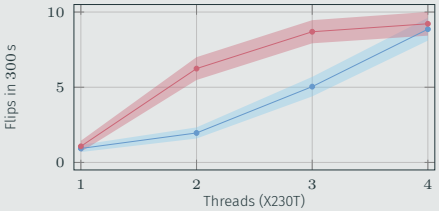
Multi-threaded Rowhammer

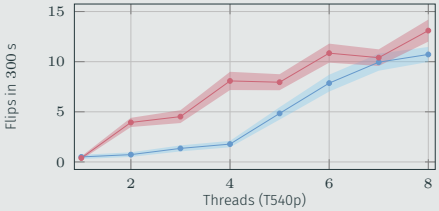
- As shown in previous work, Rowhammer can be executed in parallel to multiple banks.
- There are two basic approaches:
  - Accessing multiple banks from different threads [1]
  - Performing accesses from a single thread using memory controller instruction parallelization [3]
- We utilize multiple threads to hammer multiple banks in parallel

Multi-threaded Rowhammer – Implementation

- Two modes:
  - **CPU pinning**: Each thread is pinned to one logical CPU core
  - **No CPU pinning**: The threads are not pinned to CPU cores
- Threads handle one bank after until no banks are left

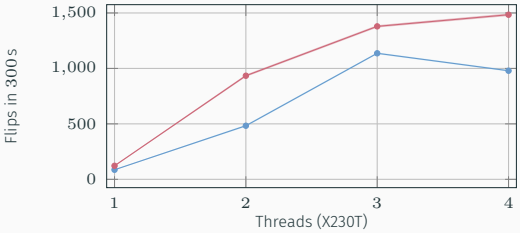
Results

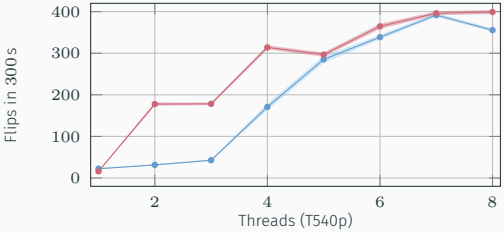

Flips in 300 s
10
5
0
1
2
3
4
Threads (X230T)


Flips in 300 s
15
10
5
0
2
4
6
8
Threads (T540p)


Without CPU pinning
With CPU pinning

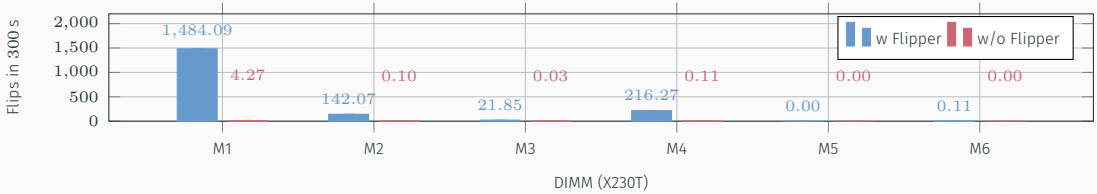
Combining both Primitives

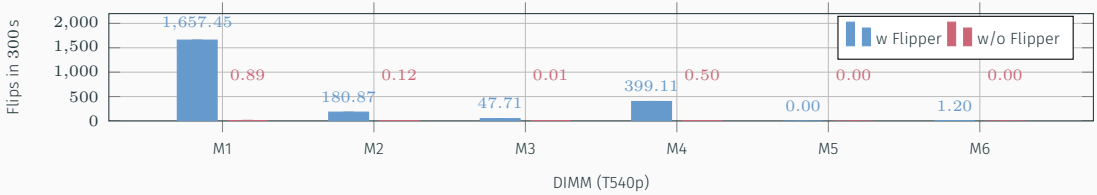




Without CPU pinning
With CPU pinning

Overall Amplification Factor





Epistemology of Rowhammer Attacks

Threats to Rowhammer Research Validity

Martin Heckel, Hannes Weissteiner, Florian Adamsky, and Daniel Gruss

ESORICS, 2025

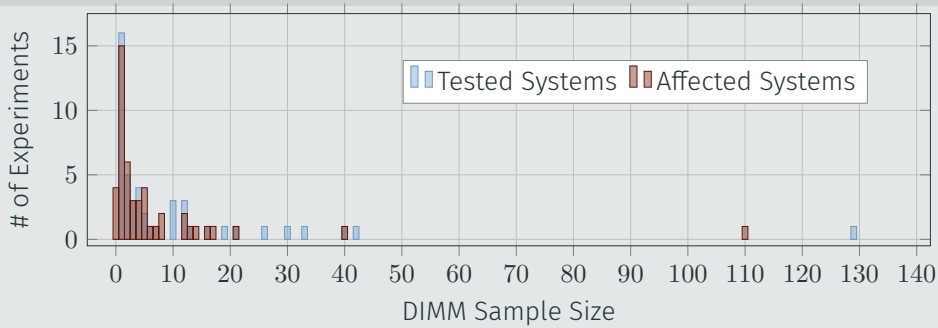
Methodology

- Google Scholar search for the word *Rowhammer*: 2509 publications
- Publications with ≥ 5 mentions of the word *Rowhammer*: 463 publications
- Peer-reviewed papers that perform Rowhammer attacks: 55 publications
- Papers at A or A* conferences: 22 publications
- Added other relevant papers: 32 publications with 48 experimental evaluations

$\mathcal{T}1$ Sample Sizes Too Small

- Multiple potential causes for bit flips:
  -  Bad memory cells
  -  Temperature fluctuations
  -  Cosmic rays
  -  Voltage fluctuations
  -  Manufacturing variations
  -  Electrical properties of the motherboard

Sample Size of experimental Evaluations



$\mathcal{T}1$ Sample Sizes Too Small

$\mathcal{R}1$: DIMMs used in empirical research must be tested for other problems, e.g., using Memtest86 (except for integrated Rowhammer tests), to ensure that no other (non-Rowhammer) problems are present.

$\mathcal{R}2$: Increase the sample size to ≥ 30 DIMMs total, spread across 3 major vendors, each with at least 2 different capacities.

$\mathcal{R}3$: Do more reproduction studies of published work to gain more insights regarding the prevalence. More venues should accept reproduction studies.

$\mathcal{T}2$ Dependence on Elevated Attacker Privileges

- Seaborn and Dullien [6] demonstrated two exploits based on Rowhammer in 2015
- Following, virtual-to-physical address mapping was made privileged
- Newer attacks use uncached memory, THP, 1GB Hugepages, etc.
- Many prequisites of exploits have been mitigated as a reaction to the publication of these techniques

$\mathcal{T}2$ Dependence on Elevated Attacker Privileges

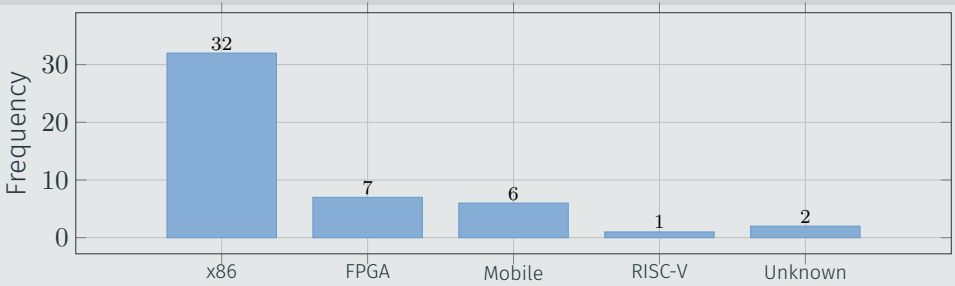
$\mathcal{R}4$: Attacks should only be classified as such when assessed under realistic attack scenarios, and there should be a more apparent distinction between actual attacks and potential (theoretical) attacks.

$\mathcal{T}3$ Uncertain Practical Applicability

- Some experiments are performed on:
  - Specialized hardware
  - Commodity hardware with extreme parameters
  - Rowhammer simulators
- While essential for understanding the Rowhammer effect, these results cannot be directly applied to real-world attacks
- $\mathcal{R}4$ applies again

$\mathcal{T}3$ Uncertain Practical Applicability

Frequencies of different experimental Setups



$\mathcal{T}4$ Comparability across Publications

- The position and number of bit flips depends on environmental parameters and the system and DIMMs that are evaluated
- In some publications, the experimental setup is not described sufficiently
- Even DIMMs that are the same model are affected differently by Rowhammer
- Hard to compare novel and existing attacks

$\mathcal{T}5$ Unspecified Age and Wear of Hardware

- Aging affects the reliability of DRAM
- Bit flips induced by Rowhammer can “burn in”
- The implementation of on-DIMM mitigations like TRR strongly depends on the vendor and model of the DIMM
- In many publications, these information are not submitted, which increases the difficulty of reproducing results

$\mathcal{T}5$ Unspecified Age and Wear of Hardware

$\mathcal{R}5$: Authors should publish the manufacturing date of the DIMMs used in experimental evaluation.

$\mathcal{R}6$: Authors should submit information about the DIMMs' wear in experimental evaluation.

$\mathcal{T}6$ Suboptimal Metrics for Comparison

- There are different metrics for the suscepbitility of systems:
  - Absolute number of bit flips in a given time or memory area
  - Minimal number of aggressor activations until the first bit flip
  - Percentage of times a bit flipped at a tested location
  - Time until the first (exploitable) bit flip is observed
- Different metrics are hard to compare
- Some metrics strongly depend on definitions, e.g., of *exploitable*

$\mathcal{T}6$ Suboptimal Metrics for Comparison

$\mathcal{R}7$: Authors should use multiple metrics for bit flips to allow for better comparisons to other works.

FlippyRAM

A Large-Scale Study of Rowhammer Prevalence

Martin Heckel, Nima Sayadi, Jonas Juffinger, Carina Fiedler, Daniel Gruss, and Florian Adamsky

NDSS, 2026

Study Overview

Study Overview

User Agreements
Privacy Policy
Risk Agreement

Study Overview

User Agreements
Privacy Policy
Risk Agreement

System Information
Retrieval

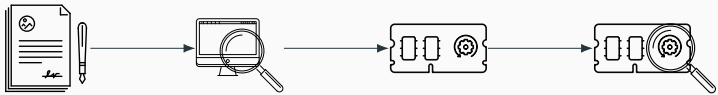
Study Overview

User Agreements
Privacy Policy
Risk Agreement

System Information
Retrieval

Reverse Engineering
of the Address-
ing Functions

Study Overview



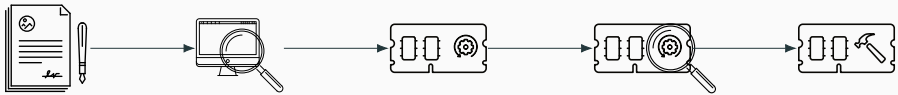
User Agreements
Privacy Policy
Risk Agreement

System Information Retrieval

Reverse Engineering of the Addressing Functions

Verification & Injection of the Addressing Functions in the Rowhammer PoCs

Study Overview



User Agreements
Privacy Policy
Risk Agreement

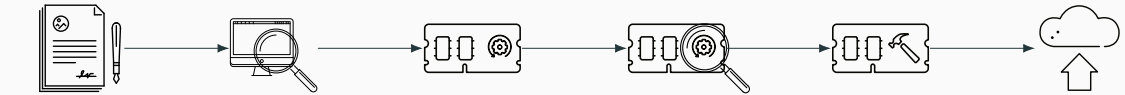
System Information Retrieval

Reverse Engineering of the Addressing Functions

Verification & Injection of the Addressing Functions in the Rowhammer PoCs

Running different Rowhammer Tools

Study Overview



User Agreements
Privacy Policy
Risk Agreement

System Information Retrieval

Reverse Engineering of the Addressing Functions

Verification & Injection of the Addressing Functions in the Rowhammer PoCs

Running different Rowhammer Tools

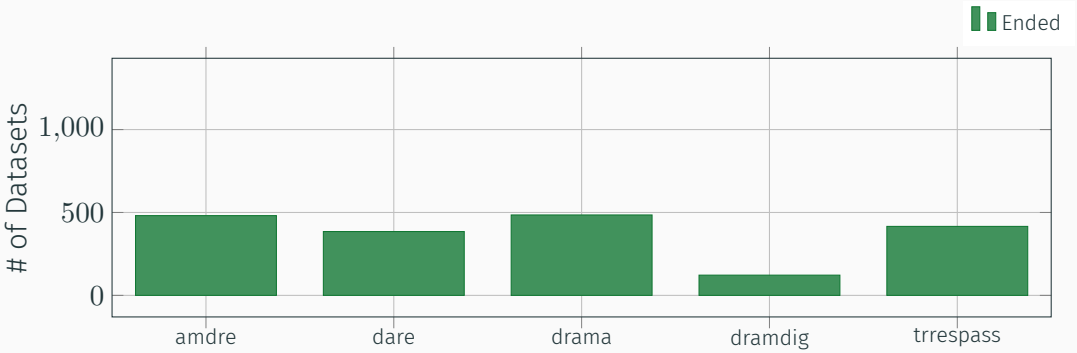
Optional uploading the data to our server

How could Users participate?

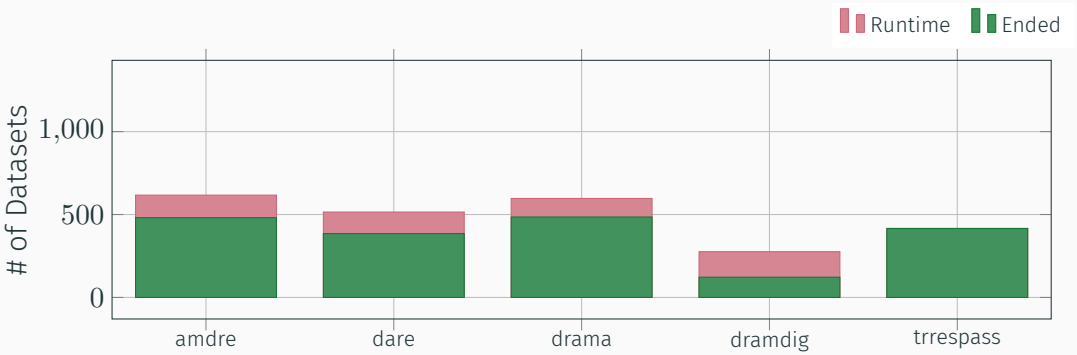
- Get a free bootable USB stick from us
- Download bootable ISO from `https://FlippyR.am`
- Clone repository and build bootable ISO themselves
- Clone repository and run docker file

How could Users participate?

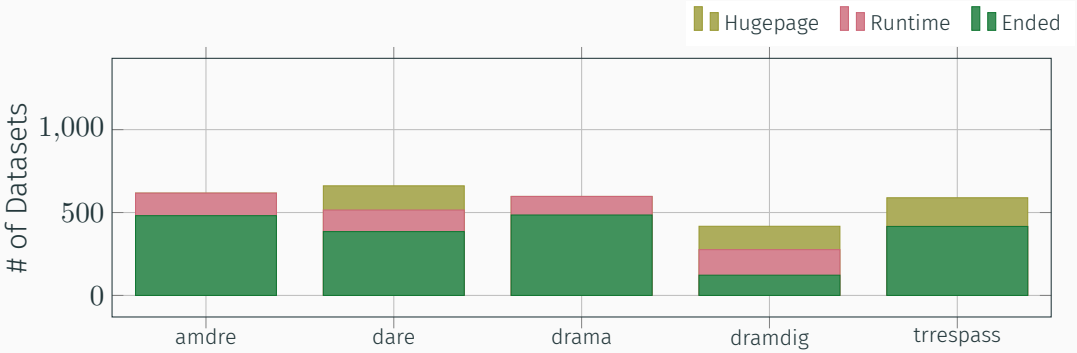
Reverse Engineering of the Addressing Functions



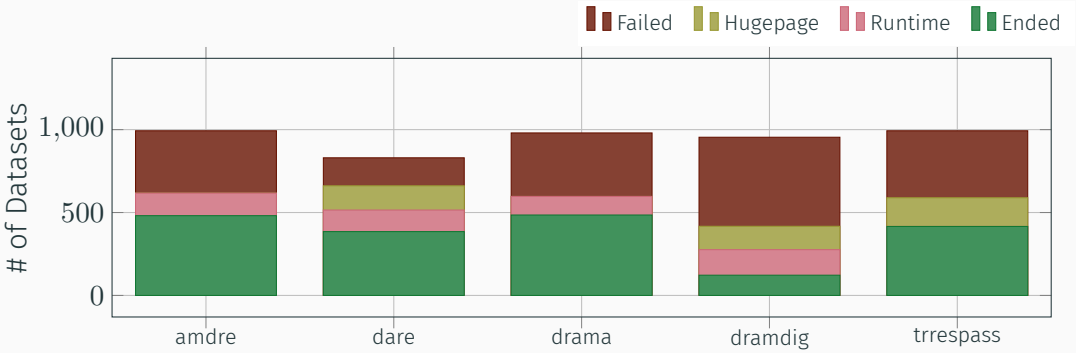
Reverse Engineering of the Addressing Functions



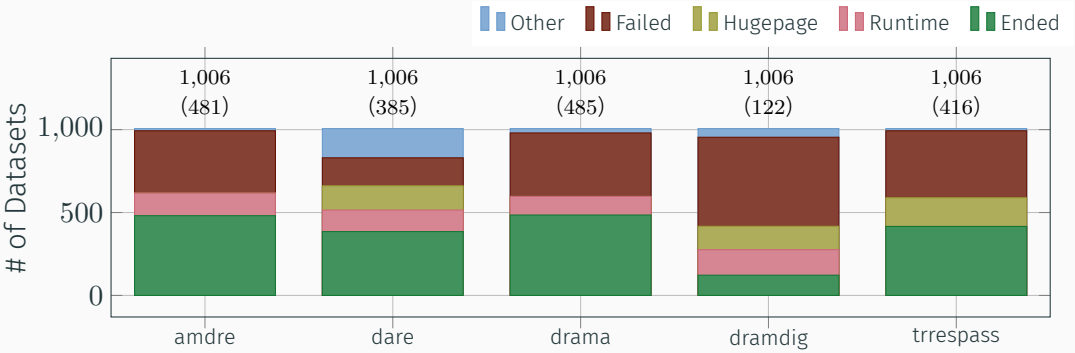
Reverse Engineering of the Addressing Functions



Reverse Engineering of the Addressing Functions

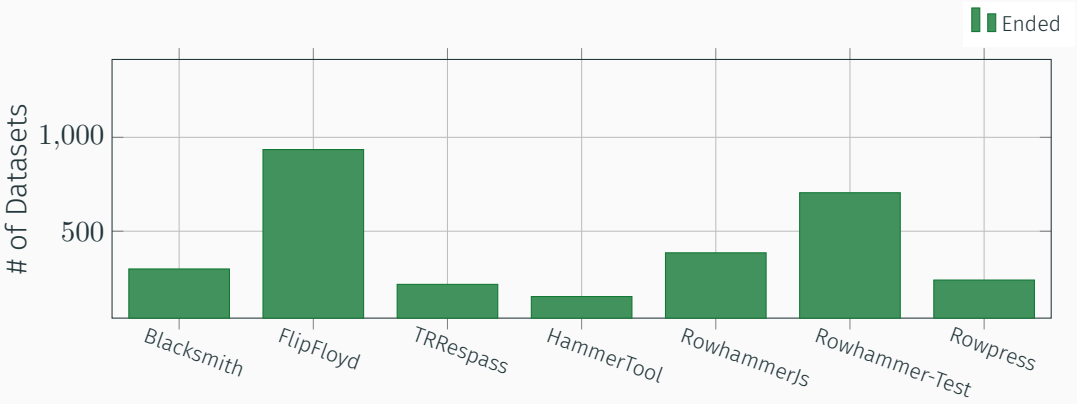


Reverse Engineering of the Addressing Functions

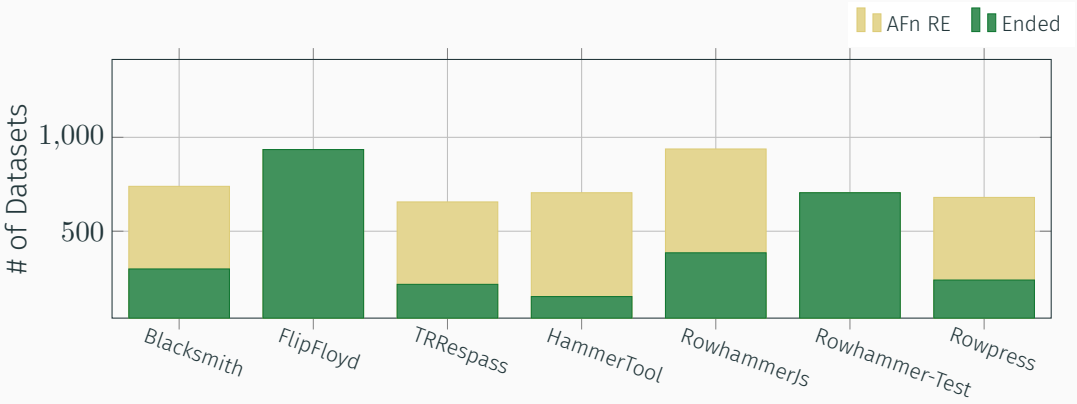


👁 (#1): Majority of cases: reverse-engineering tools fail, crash, or exceed time limits!

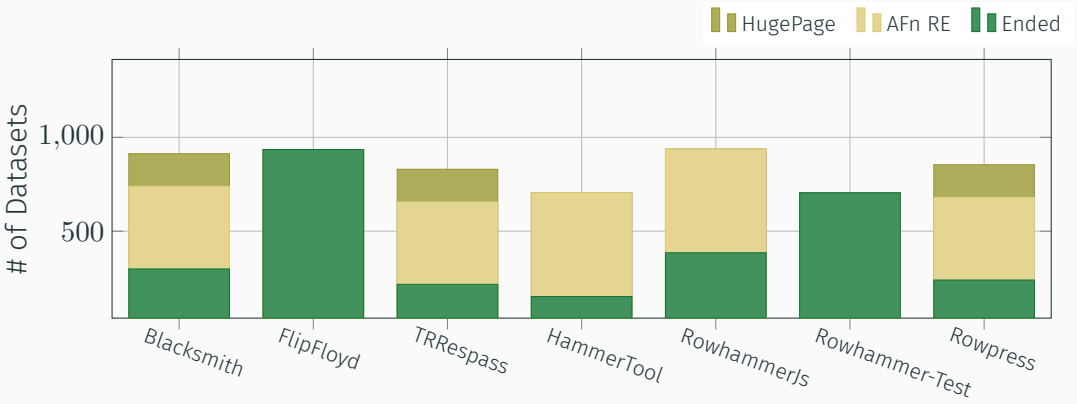
Running different Rowhammer Tools



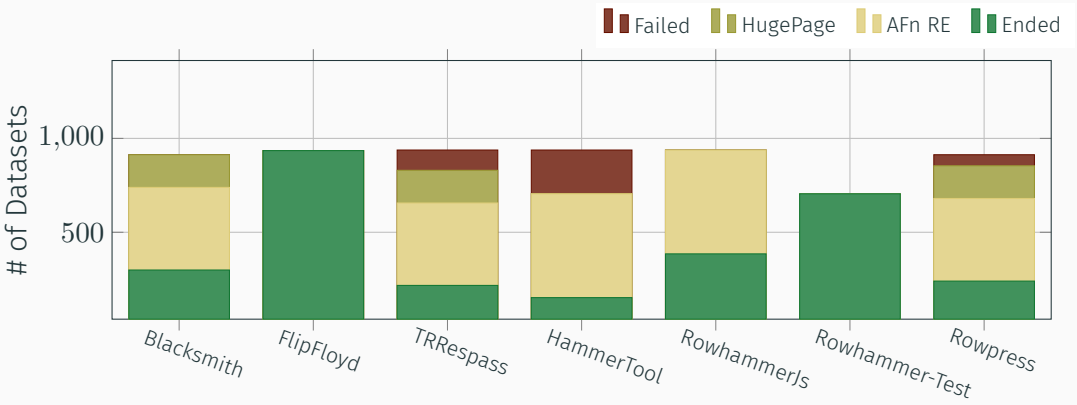
Running different Rowhammer Tools



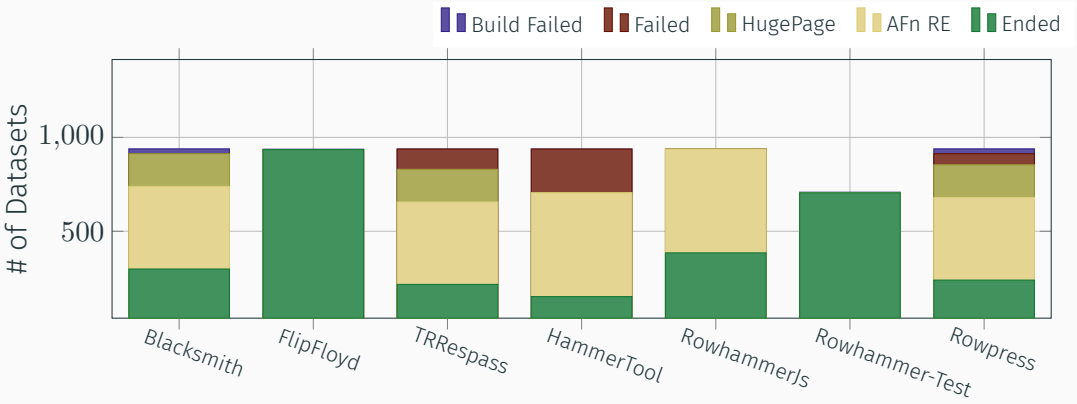
Running different Rowhammer Tools



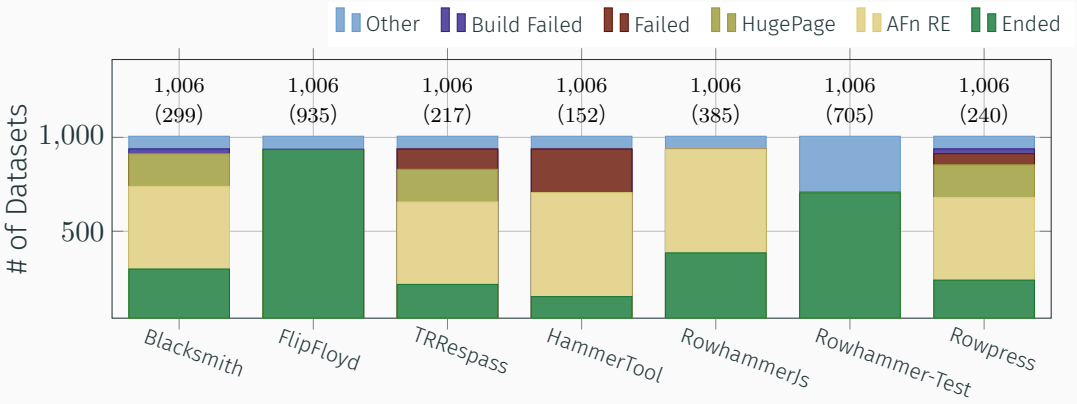
Running different Rowhammer Tools



Running different Rowhammer Tools

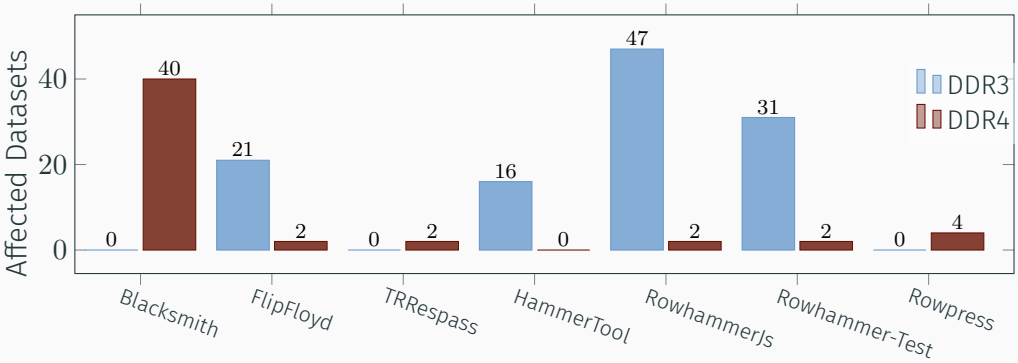


Running different Rowhammer Tools



👁 (#2): Many Rowhammer tools failed because of missing DRAM functions or 1 GiB hugepages.

Affected Datasets by Tool

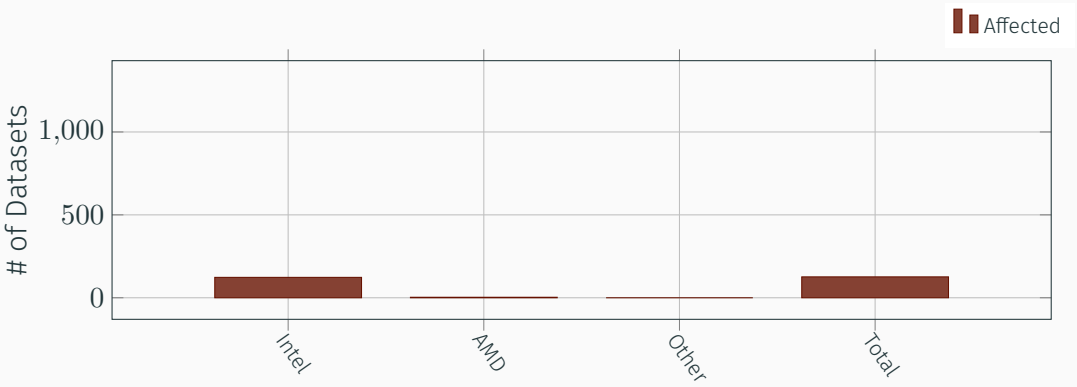


👁 (#3): 126 ($12.5\,\%$) out of 1006 datasets are vulnerable to fully-automated Rowhammer attacks!

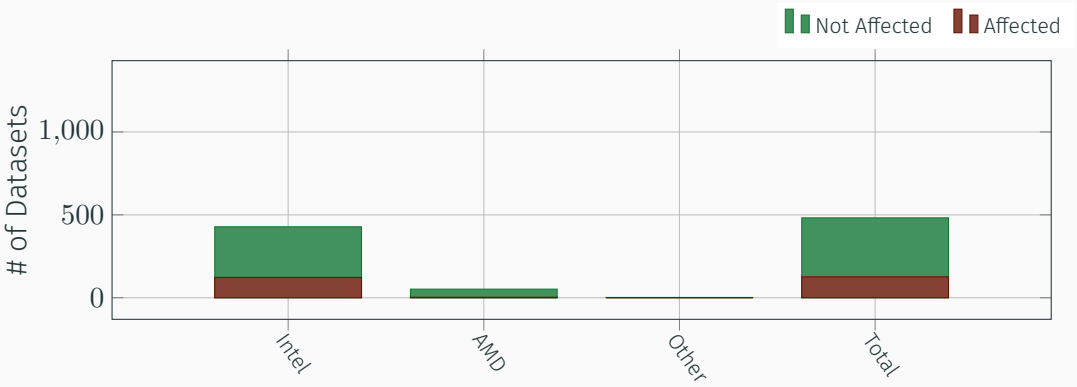
👁 (#4): DDR3 $\rightarrow$ simple fast patterns (RowhammerJS); DDR4 with TRR $\rightarrow$ pattern fuzzing for non-uniform patterns (Blacksmith)

👁 (#5): The minimum time to the first bit flip was between $0\,\mathrm{min}$ and $115\,\mathrm{min}$ on average, which is a practical time frame for real-world attacks.

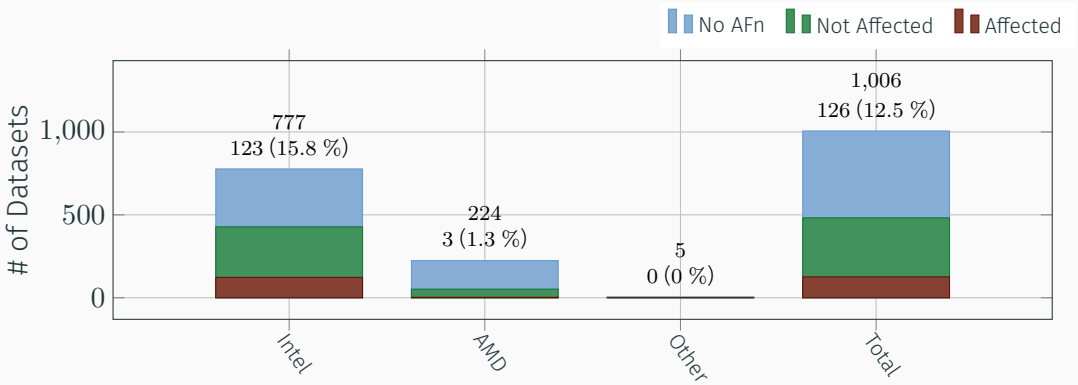
Affected Datasets by CPU Vendor



Affected Datasets by CPU Vendor

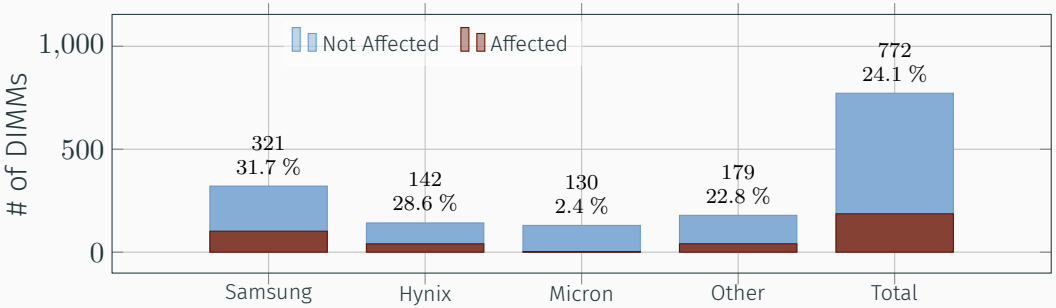


Affected Datasets by CPU Vendor



👁 (#6): Mainly tools for Intel, fewer AMD tools, especially when we started the study $\rightarrow$ we expect AMD to be equally affected.

Affected DIMMs by DRAM Vendor



👁 (#7): DRAM from Samsung, Hynix, and third-party resellers similarly affected by Rowhammer but only $2.4\,\%$ of Micron DIMMs?

Conclusion

Conclusion

- First approach to reverse-engineer DRAM bank addressing functions on AMD-based systems
- Novel approach for DRAM bank addressing function verification
- First identification of Rank addressing functions
- First cover channel based on row conflicts on DDR5 (including website fingerprinting)
- Approach for Rowhammer amplification on DDR3, bypassing double refresh rate mitigation

Conclusion

- Identification of six threats to Rowhammer research validity, seven recommendations to improve
- First Large-Scale Study of Rowhammer prevalence with multiple insights:
  - $12.5\,\%$ of datasets are susceptible to out fully-automated Rowhammer framework
  - Main reasons for systems not being tested: DRAM addressing function reverse-engineering, support on more systems
  - Probably, more systems are affected

TU Graz

Hochschule Hof
University of Applied Sciences

Real-World Rowhammer: Understanding and Addressing the Challenges to Rowhammer Attacks

Martin Heckel

April 01, 2026

References i

[1] Wei He et al. **“WhistleBlower: A System-Level Empirical Study on RowHammer.”** In: *IEEE Transactions on Computers* (2023).

[2] JEDEC Solid State Technology Association. ***DDR4 SDRAM Standard.*** 2021. URL: https://www.jedec.org/standards-documents/docs/jesd79-4a.

[3] Ingab Kang et al. **“SledgeHammer: Amplifying Rowhammer via Bank-level Parallelism.”** In: *USENIX Security*. 2024.

[4] Yoongu Kim et al. **“Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors.”** In: *ISCA*. 2014.

[5] Peter Pessl et al. **“DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks.”** In: *USENIX Security*. 2016.

References ii

[6] Mark Seaborn and Thomas Dullien. *Exploiting the DRAM rowhammer bug to gain kernel privileges.* 2015. URL: http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html.