Flipper: Rowhammer on Steroids

Martin Heckel[1,2] and Florian Adamsky[2]

February 19, 2025

[1] Graz University of Technology
[2] Hof University of Applied Sciences

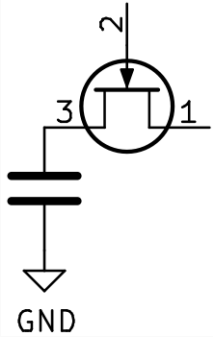
NeRAM
FWF Der Wissenschaftsfonds.
DFG Deutsche Forschungsgemeinschaft

Outline

Introduction

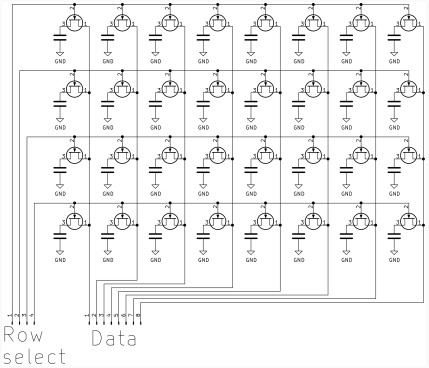
DRAM – Cells

- A single cell consists of a capacitor storing the actual data and a transistor controlling the access
- Reading procedure: Enable the control pin and read the voltage at the access pin
- Writing procedure: Apply the level that should be written to the access pin and enable the control pin



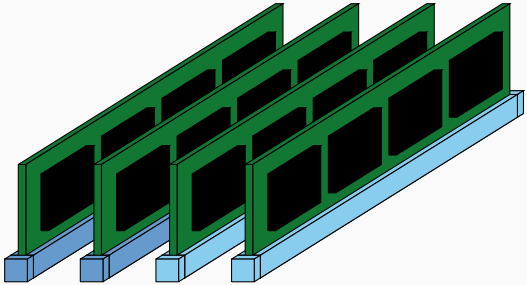
DRAM – Array

- Multiple of these cells organized is array
- Control pins of the cells connected in rows (only entire rows can be enabled)
- Access pins of the cells conneted in columns
- Capacitors loose chage over time, so it is required to refresh the cells periodically

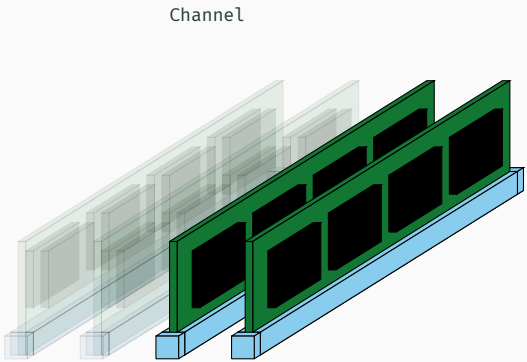


DRAM – physical architecture

System DRAM

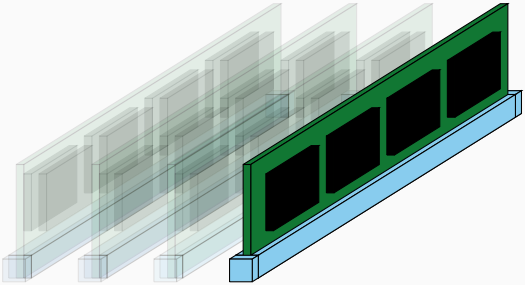


DRAM – physical architecture

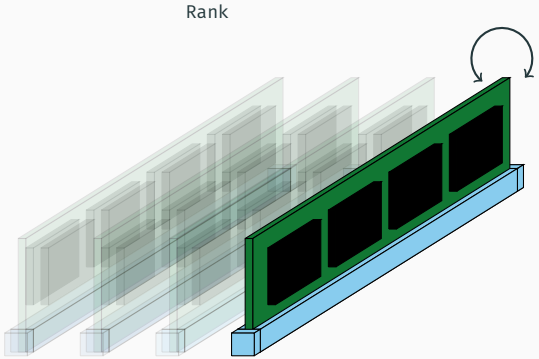


DRAM – physical architecture

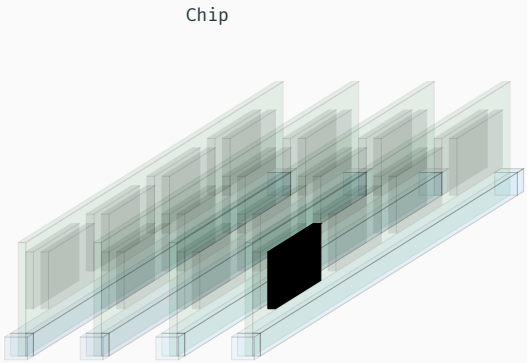
DIMM



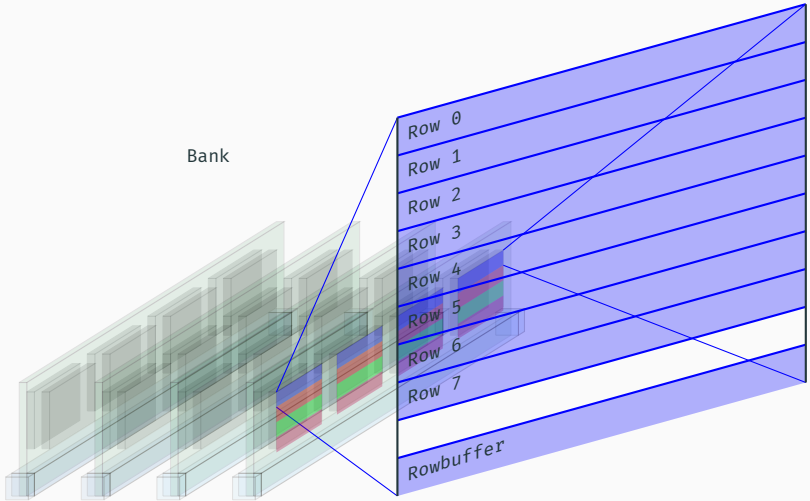
DRAM – physical architecture



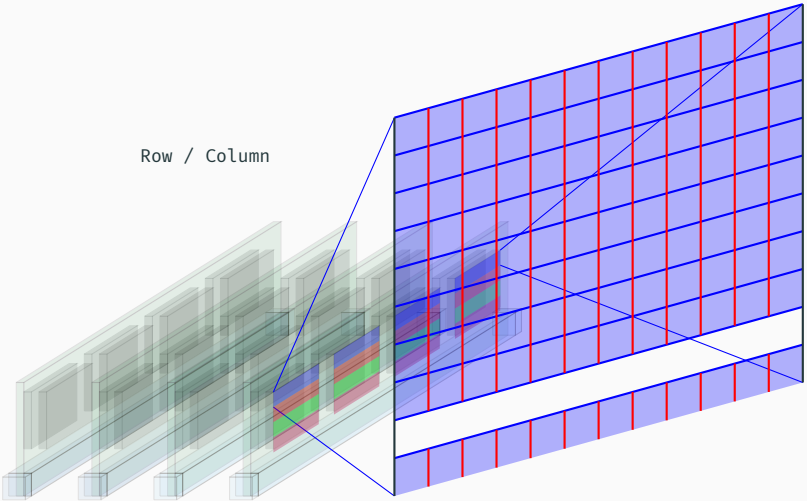
DRAM – physical architecture



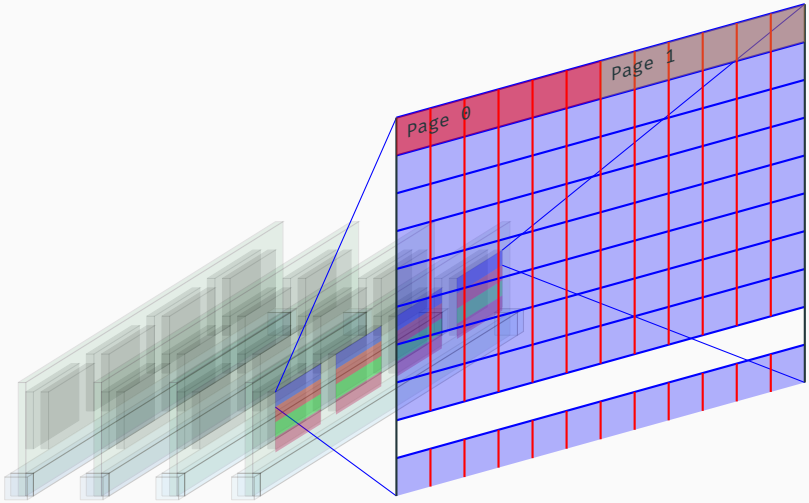
DRAM – physical architecture



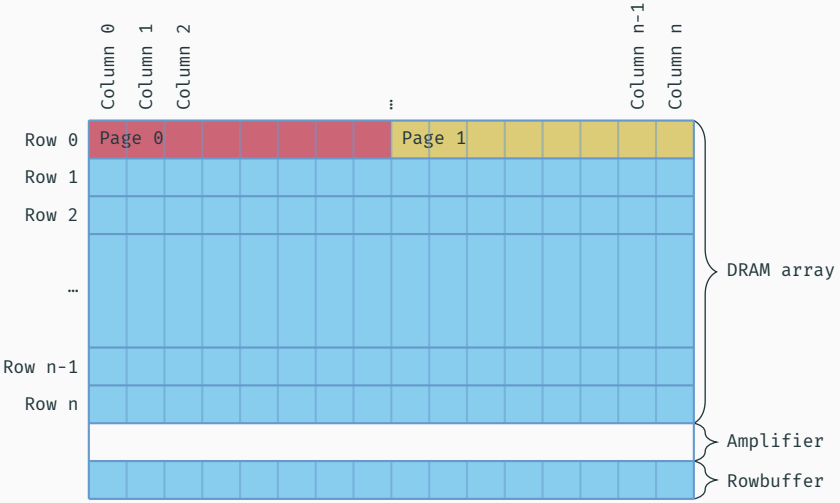
DRAM – physical architecture



DRAM – physical architecture

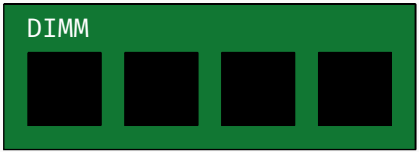


Structure within a DRAM bank



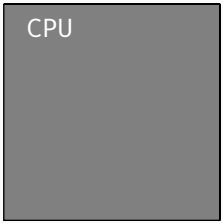
DRAM addressing

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



Rowhammer

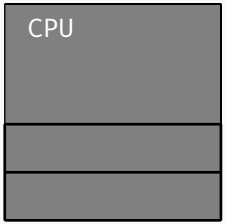
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

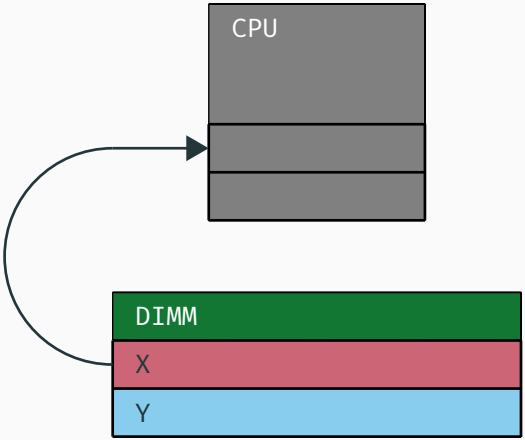
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

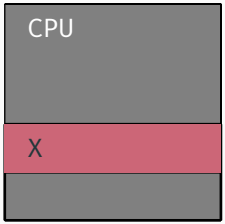
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

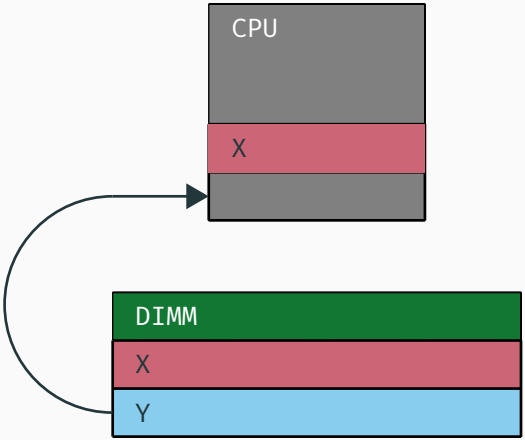
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

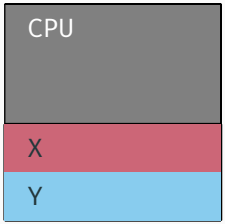
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

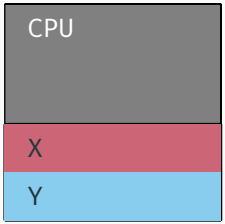
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

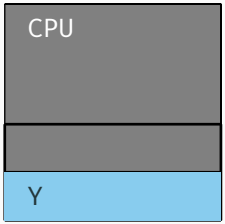
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

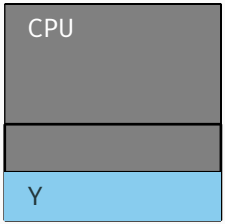
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

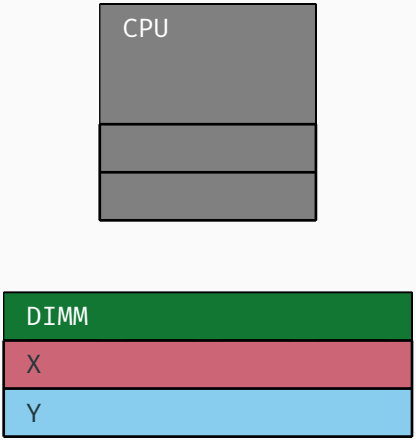
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

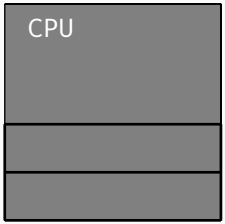
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

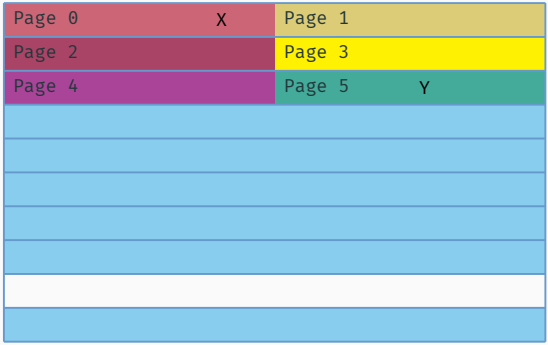
```
1  hammer:
2   mov eax, X
3   mov ebx, Y
4   clflush X
5   clflush Y
6   jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

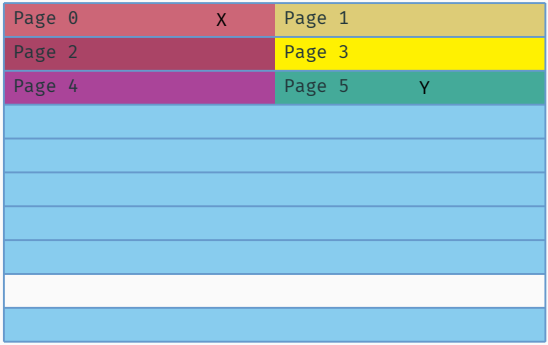
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

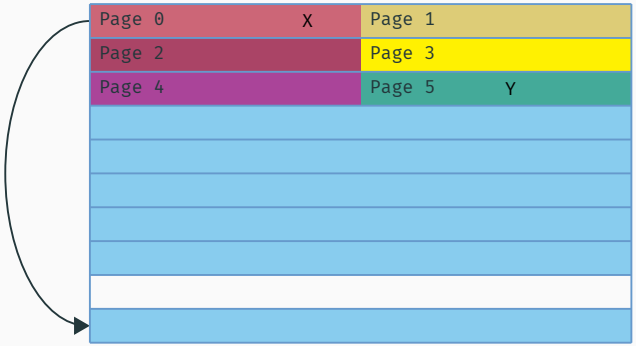
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

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



Source code from Kim et al. [1]

Rowhammer

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

Source code from Kim et al. [1]

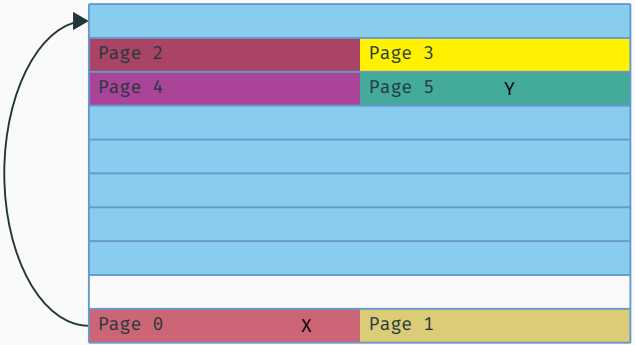
Rowhammer

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

Source code from Kim et al. [1]

Rowhammer

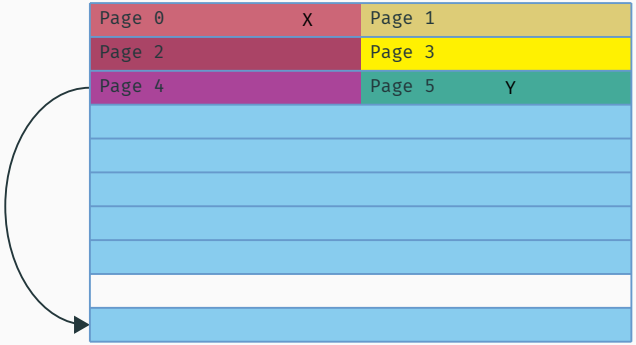
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

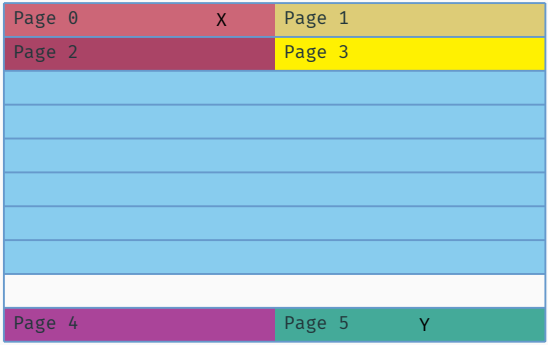
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

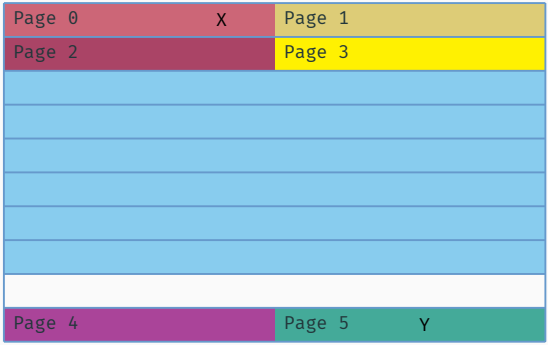
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

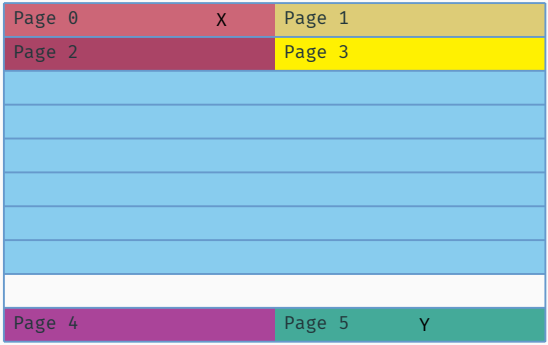
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

Rowhammer

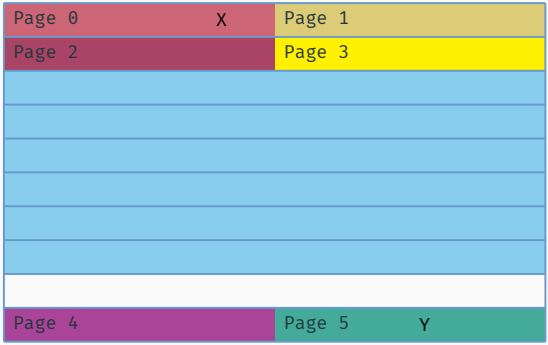
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]

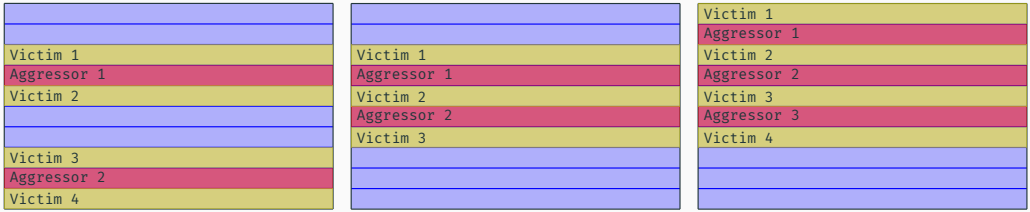
Rowhammer

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



Source code from Kim et al. [1]

Rowhammer



(a) Single Side (b) Double Side (c) Three Side

Figure 1: Examples of rowhammer patterns

Rowhammer Amplification Attacks

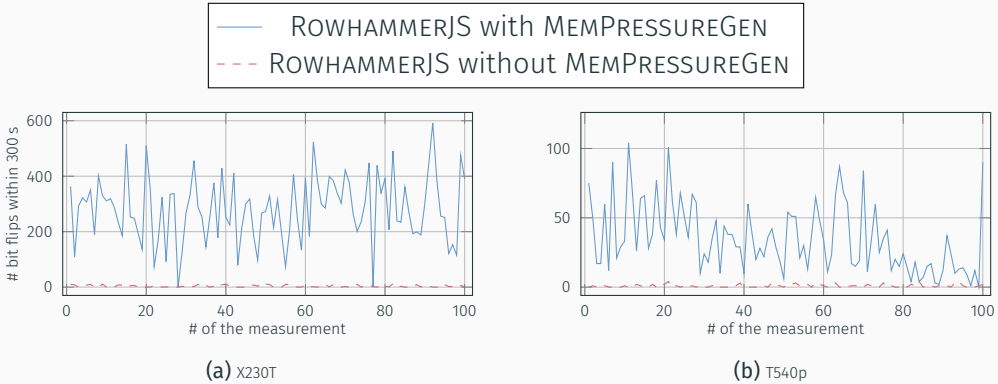
Additional Memory Accesses

- 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 MemPressureGen

- 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

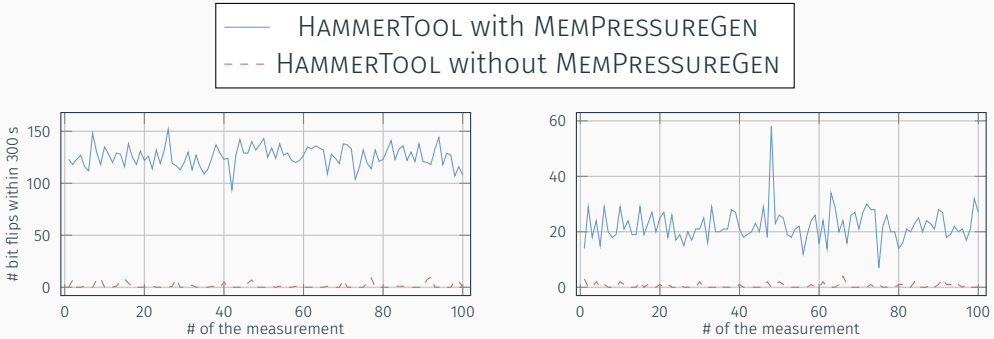
Additional Memory Accesses – MemPressureGen with Rowhammer.JS



(a) X230T (b) T540p

Figure 2: Number of bit flips measured within 300 s with and without MemPressureGen running in parallel.

Additional Memory Accesses – MemPressureGen with HammerTool



(a) X230T (b) T540p

Figure 3: Number of bit flips measured within 300 s with and without MemPressureGen running in parallel.

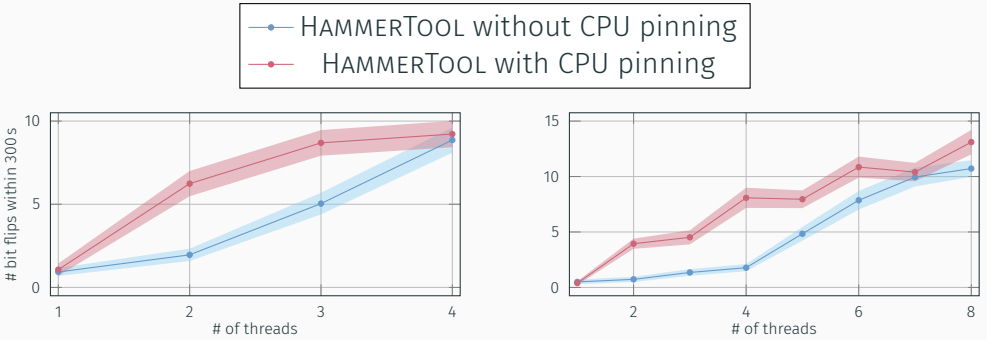
Multi-threaded Rowhammer

- 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
  - Performing accesses from a single thread using memory controller instruction parallelization
- We utilize multiple threads to hammer multiple banks in parallel

Multi-threaded Rowhammer – Implementation HammerTool

- 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

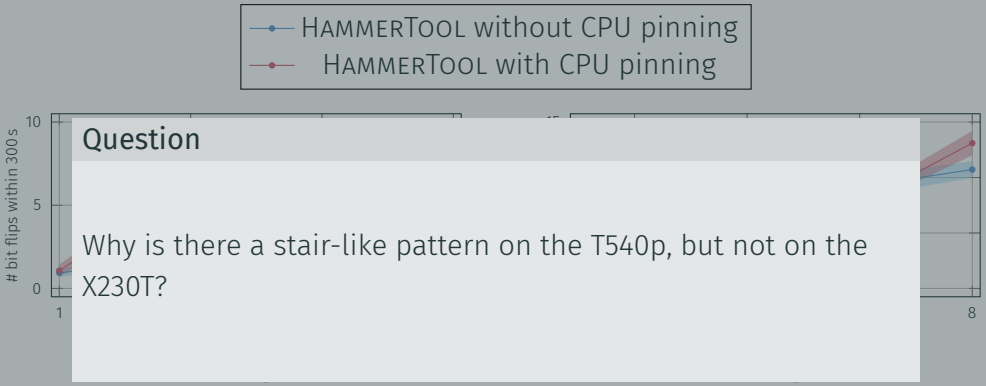
Multi-threaded Rowhammer – HammerTool



(a) X230T with 4 logical cores

(b) T540p with 8 logical cores

Figure 4: Number of bit flips found by HammerTool in 300 s without MemPressureGen running in parallel.



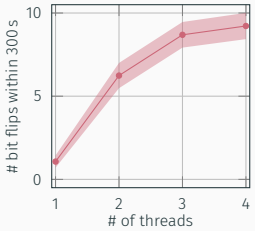
Question

Why is there a stair-like pattern on the T540p, but not on the X230T?

Figure 4: Number of bit flips found by HammerTool in 300 s without MemPressureGen running in parallel.

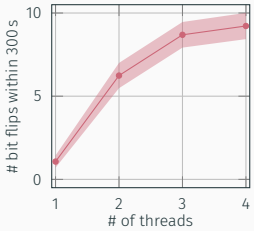
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge



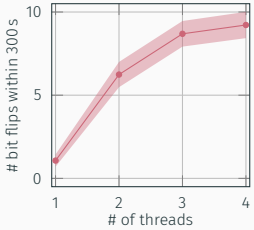
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge
  - Two 16 B loads and one 16 B store per cycle



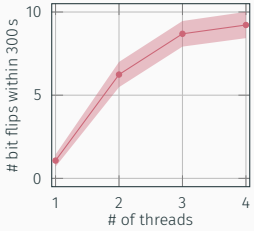
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge
  - Two 16 B loads and one 16 B store per cycle
  - L2 → L1 bandwith: 32 B per cycle



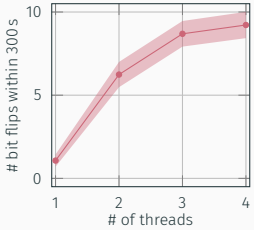
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge
  - Two 16 B loads and one 16 B store per cycle
  - L2 → L1 bandwith: 32 B per cycle
  - L3 → L2 bandwith: 32 B per cycle



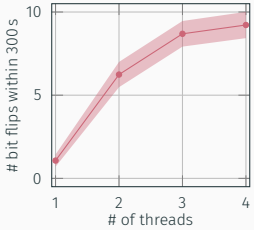
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge
  - Two 16 B loads and one 16 B store per cycle
  - L2 $\rightarrow$ L1 bandwith: 32 B per cycle
  - L3 $\rightarrow$ L2 bandwith: 32 B per cycle
  - Two hyperthreads $\Rightarrow$ 32 B load per cycle



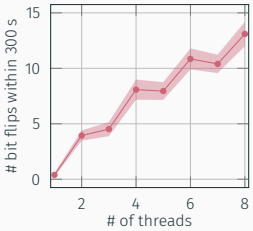
Stair-like pattern on the T540p – Our hypothesis

- Ivy Bridge
  - Two 16 B loads and one 16 B store per cycle
  - L2 → L1 bandwith: 32 B per cycle
  - L3 → L2 bandwith: 32 B per cycle
  - Two hyperthreads ⇒ 32 B load per cycle
  - **Bottleneck**: 32 B per cycle



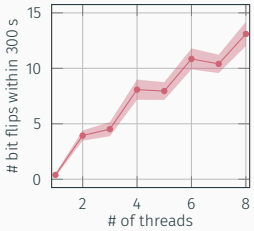
Stair-like pattern on the T540p – Our hypothesis

- Haswell



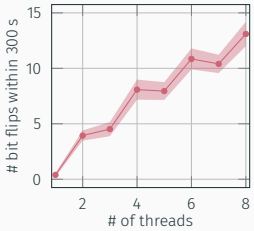
Stair-like pattern on the T540p – Our hypothesis

- Haswell
  - Two 32 B loads and one 32 B store per cycle



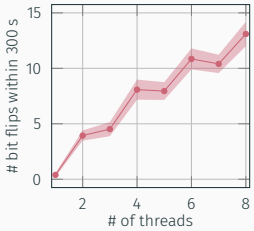
Stair-like pattern on the T540p – Our hypothesis

- Haswell
  - Two 32 B loads and one 32 B store per cycle
  - L2 → L1 bandwith: 64 B per cycle



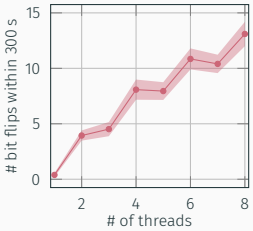
Stair-like pattern on the T540p – Our hypothesis

- Haswell
  - Two 32 B loads and one 32 B store per cycle
  - L2 → L1 bandwith: 64 B per cycle
  - L3 → L2 bandwith: 32 B per cycle



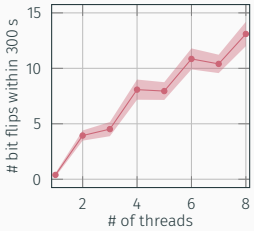
Stair-like pattern on the T540p – Our hypothesis

- Haswell
  - Two 32 B loads and one 32 B store per cycle
  - L2 $\rightarrow$ L1 bandwith: 64 B per cycle
  - L3 $\rightarrow$ L2 bandwith: 32 B per cycle
  - Two hyperthreads $\Rightarrow$ 64 B load per cycle



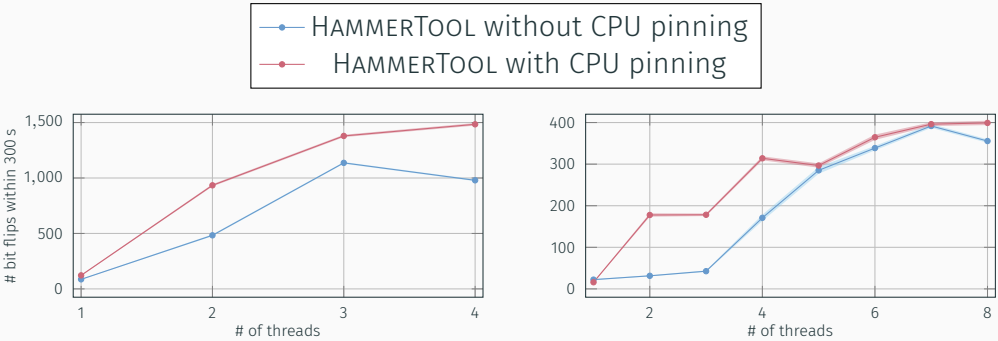
Stair-like pattern on the T540p – Our hypothesis

- Haswell
  - Two 32 B loads and one 32 B store per cycle
  - L2 → L1 bandwith: 64 B per cycle
  - L3 → L2 bandwith: 32 B per cycle
  - Two hyperthreads ⇒ 64 B load per cycle
  - **Bottleneck**: 32 B per cycle



Experimental Evaluation

Combining both Primitives – Flipper

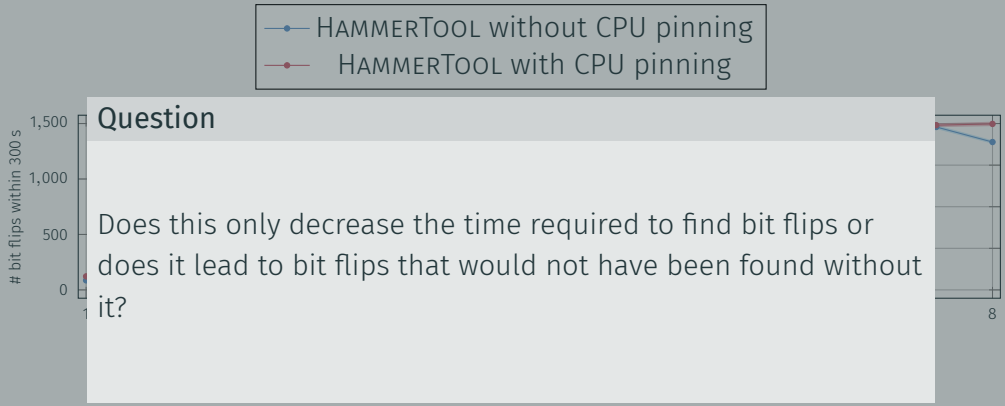


(a) X230T with 4 logical cores

(b) T540p with 8 logical cores

Figure 5: Number of bit flips found by HammerTool in 300 s with MemPressureGen running in parallel.

Combining both Primitives – Flipper



Question

Does this only decrease the time required to find bit flips or does it lead to bit flips that would not have been found without it?

Figure 5: Number of bit flips found by HammerTool in 300 s with MemPressureGen running in parallel.

Unique Bit Flips

- A bit flip is considered to be unique if:
  - **Aggressor rows** that were hammered when the bit flip was triggered;
  - **Victim row** that contained the actual bit flip;
  - **Offset** in the victim row, which specifies the byte that contained the bit flip;
  - **Flip mask** which specifies the bit/bits that flipped within the byte; and
  - **Flip direction** which specifies if a bit flipped from 1 to 0 or vice versa.

Unique Bit Flips – Evaluation

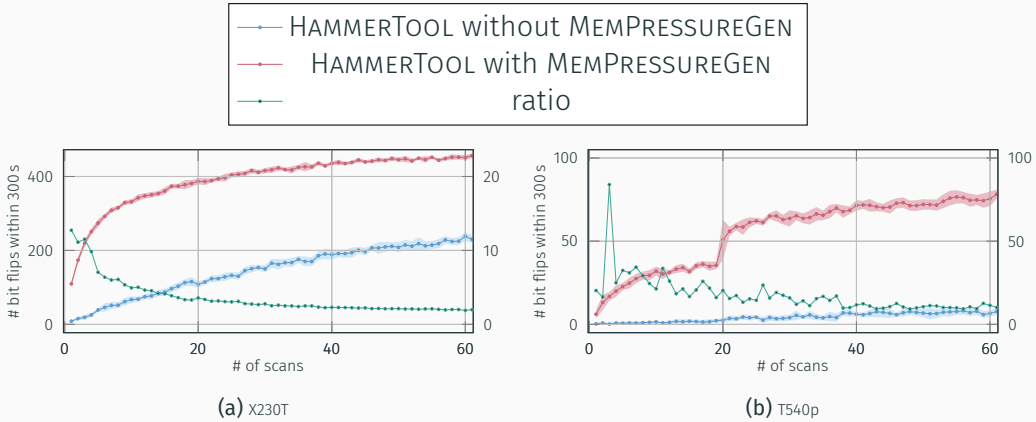


Figure 6: Number and Ratio of unique bit flips found by HammerTool with and without MemPressureGen running at the same time when scanning the same THP *x* times.

Measurement Time

- We performed most of our measurements for 300 s each and repeated these measurements multiple times
- We assume that the number of bit flips should be linearly related to the time, e. g., measuring for the double time should result in the double number of bit flips

Measurement Time – Evaluation

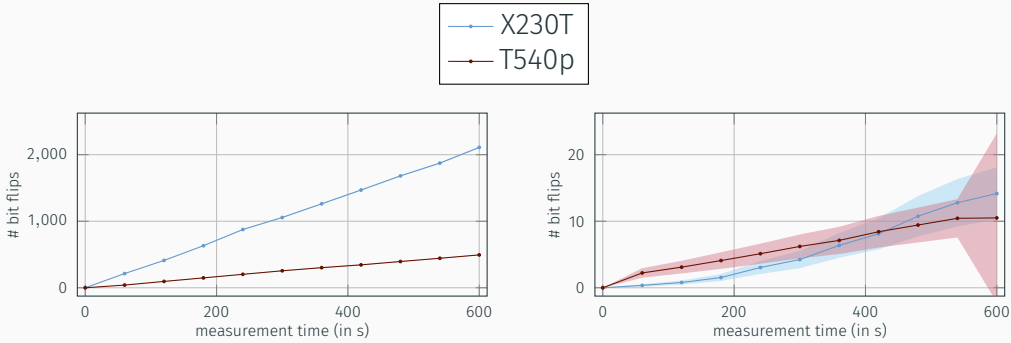


Figure 7: Number of bit flips found with and without MemPressureGen in relation to the time of the measurement.

Overall Amplification Factor

- Which amplification can be reached by comparing both primitives?
- **Note**: We performed all measurements on systems with the double refresh rate mitigation

Overall Amplification Factor – Evaluation

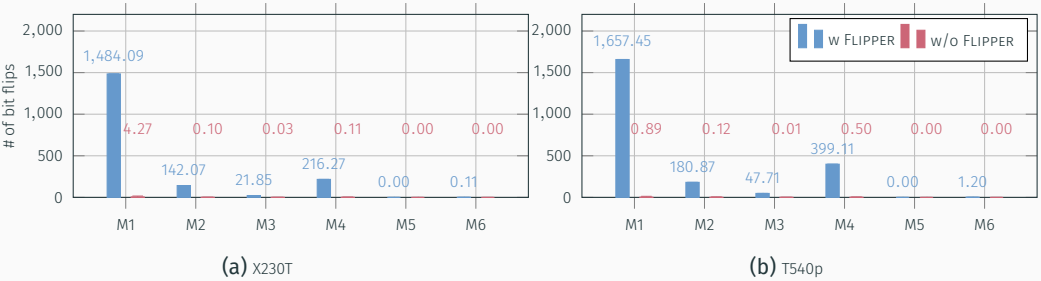


Figure 8: Comparison of the number of bit flips with and without Flipper

Conclusion

Conclusion

- We performed a systematic evaluation of the previously know effect of multi-threaded hammering
- We introduced new instructions that increase the number of bit flips occuring in a given time significantly
- We showed that FLIPPER, the combination of both approaches leads to bit flips that would not have occurred otherwise
- We calculated an amplification factor of 837.5 compared to ROWHAMMERJS

Flipper: Rowhammer on Steroids

Martin Heckel[1,2] and Florian Adamsky[2]

February 19, 2025

[1] Graz University of Technology
[2] Hof University of Applied Sciences

NeRAM
FWF Der Wissenschaftsfonds.
DFG Deutsche Forschungsgemeinschaft

Additional Memory Accesses – Different access types

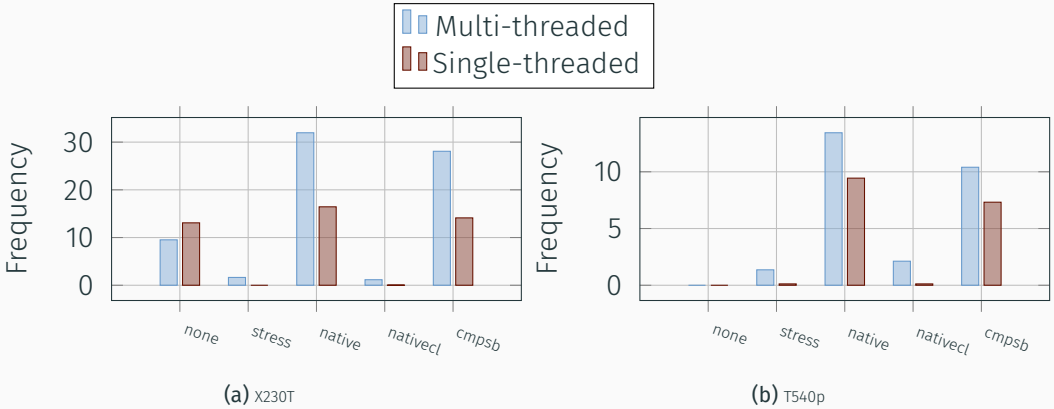


Figure 9: Number of bit flips identified by HammerTool in 300 s based on different stress techniques. An average is shown over 25 measurements.

References i

[1] Yoongu Kim et al. **“Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors”.** In: *ISCA*. 2014. DOI: 10.1109/ISCA.2014.6853210.