

Comp 790-184: Hardware Security and Side-Channels

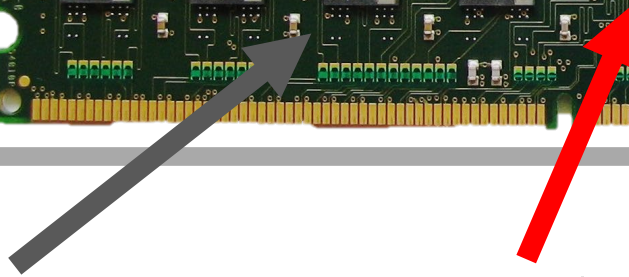
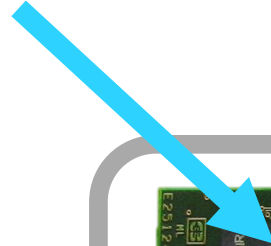
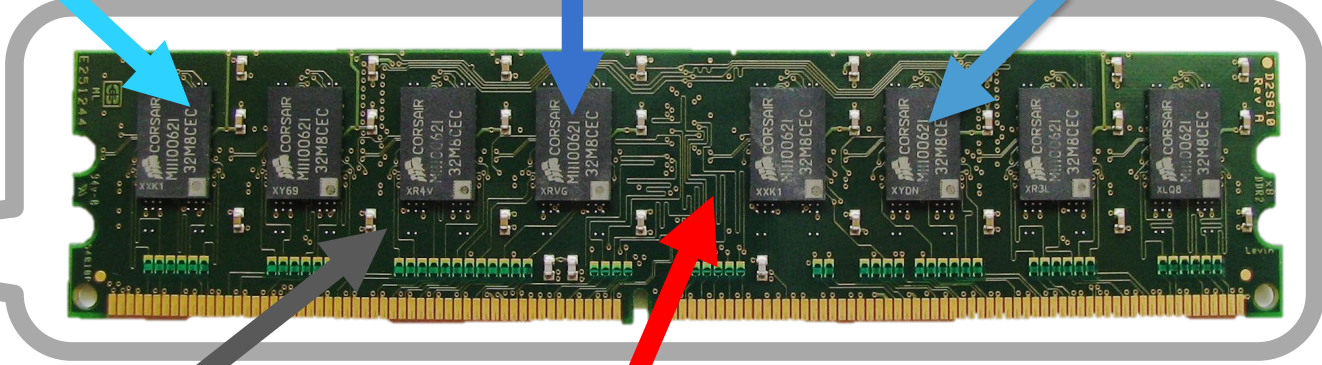
Lecture 6: Rowhammer

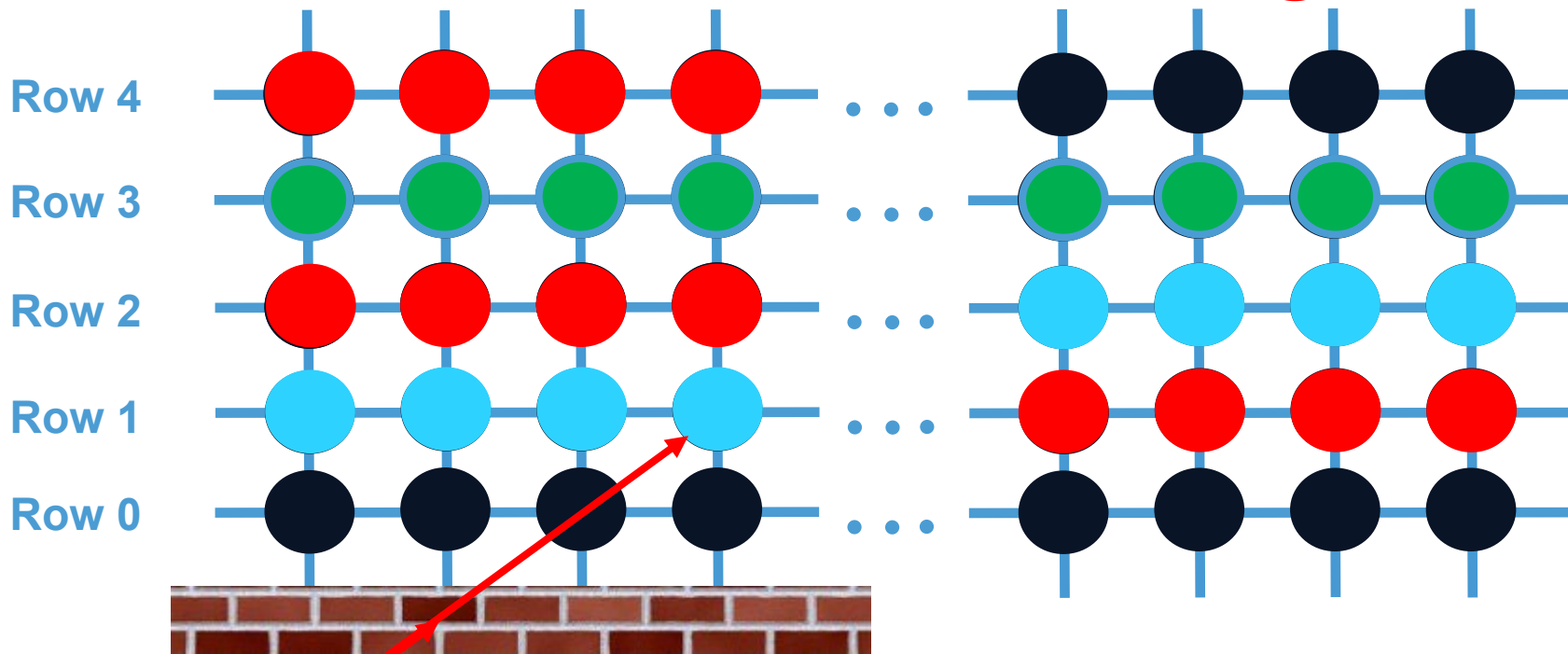
March 4, 2025
Andrew Kwong



THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

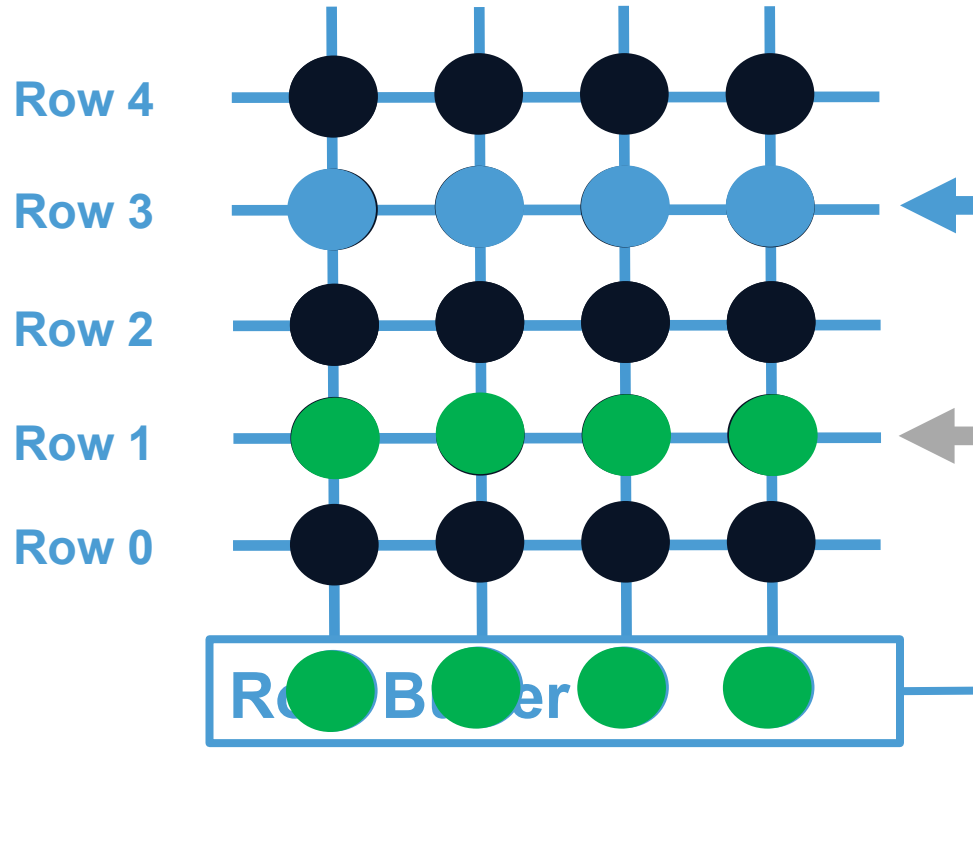
Slides adapted from Mengjia
Yan (shd.mit.edu)





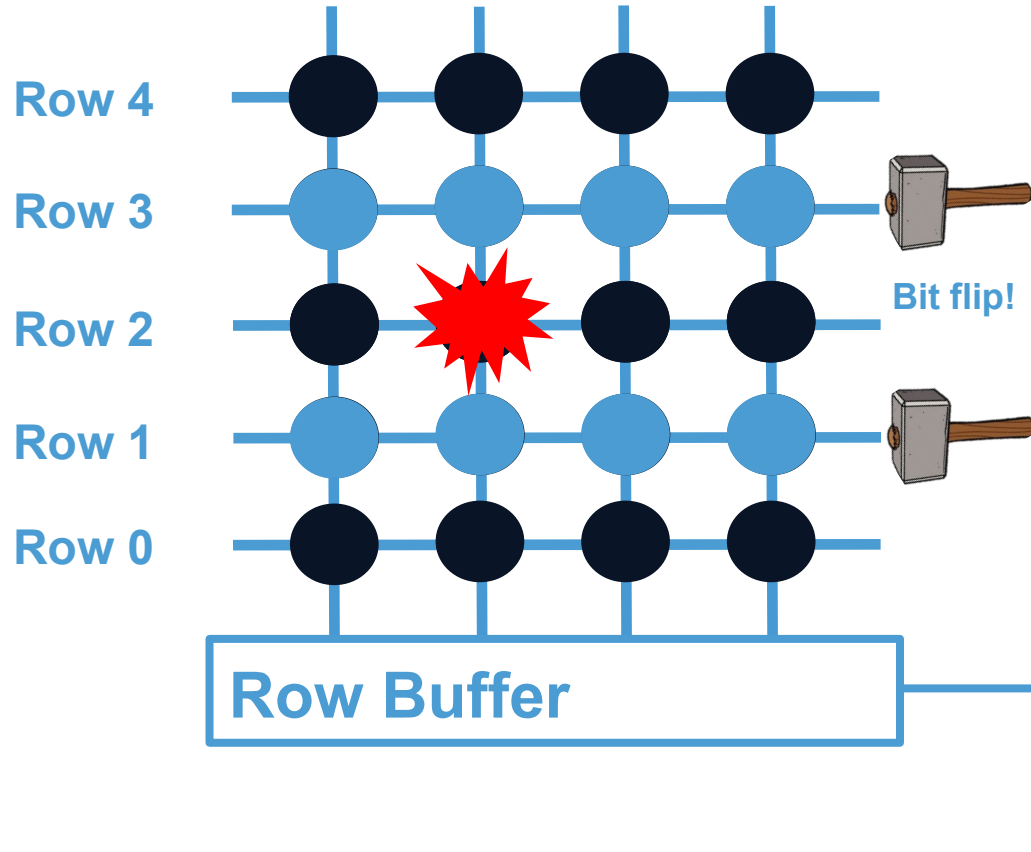
- OS enforces isolation
- **Can physics help us bypass OS and access memory across these boundaries?**

How DRAM works



- Data in row 3 is read
- Entire row is activated and stored in Row Buffer
- Forwarded to CPU

Rowhammer



- Activating a row drains charge from nearby capacitors
- Repeated activation of rows causes bit flips in nearby rows!
- Attacker that controls values in rows 1 and 3 writes to victim's memory in row 2



Why Should we Care About RowHammer?

- One can predictably induce bit flips in commodity DRAM chips
- An example of how a simple hardware failure mechanism can create a widespread system security vulnerability



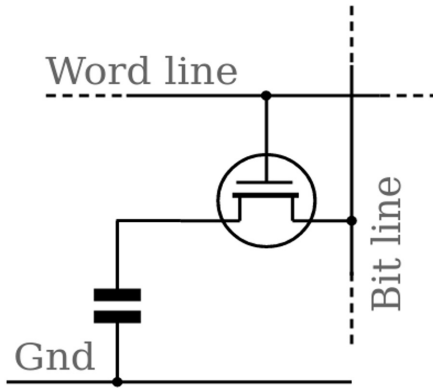
The image shows a screenshot of a Wired article. At the top left is the 'WIRED' logo. To its right is the article title 'Forget Software—Now Hackers Are Exploiting Physics'. Below the title is a navigation bar with categories: BUSINESS, CULTURE, DESIGN, GEAR, and SCIENCE. The author's name 'ANDY GREENBERG' is followed by 'SECURITY' and the date '08.31.16 7:00 AM'. On the left side, there is a 'SHARE' section with a Facebook icon and 'SHARE 18276' and a Twitter icon with 'TWEET'. The main headline of the article is 'FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS' in large, bold, black serif font.

Outline

- Why does RowHammer happen? What is its working mechanism?
- How to perform the attack in practice? Challenges?
- Attack consequences? Mitigations?

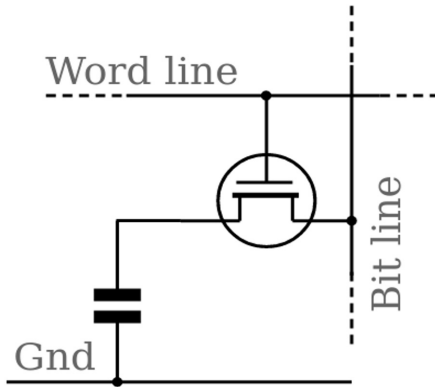
DRAM Basics

- Each bit in DRAM is stored in a “cell” using a *capacitor*
- *Read is destructive*
- DRAM cells lose their state over time (hence **Dynamic** RAM)
- Data stored in DRAM cells needs to be “**refreshed**” at a regular interval



DRAM Basics

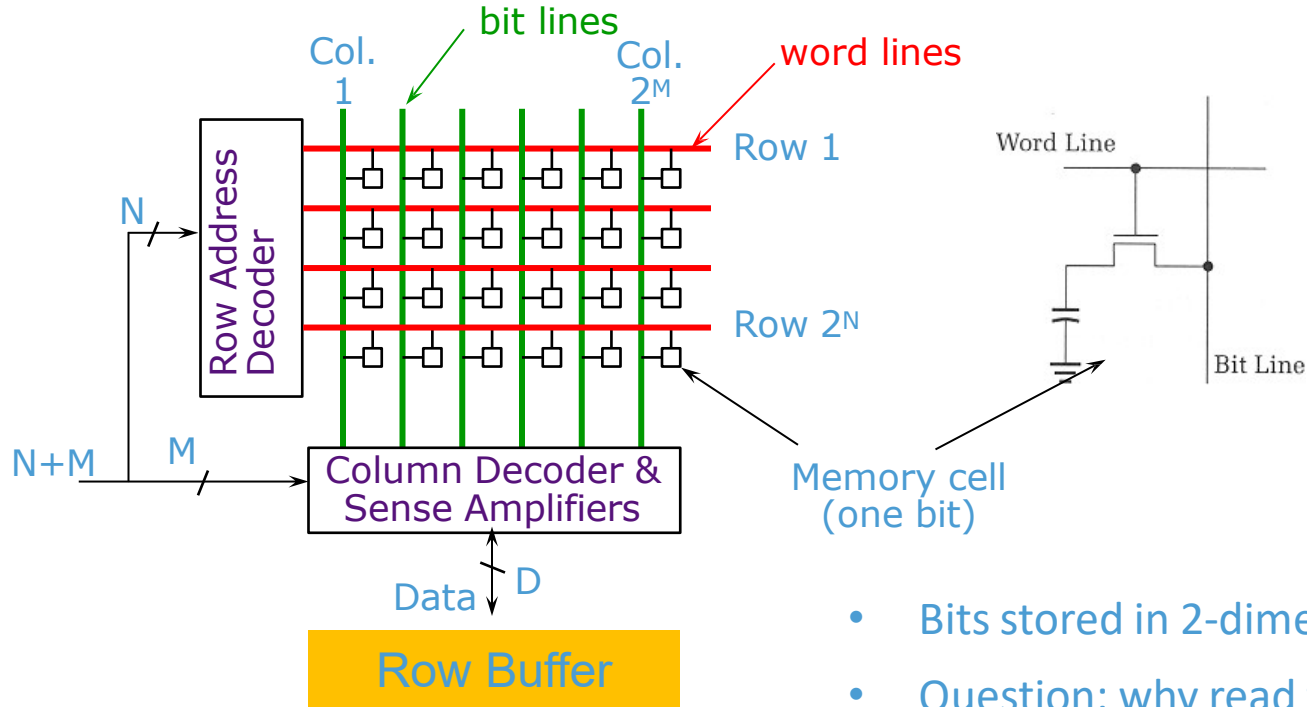
- Each bit in DRAM is stored in a “cell” using a *capacitor*
- *Read is destructive*
- DRAM cells lose their state over time (hence **Dynamic** RAM)
- Data stored in DRAM cells needs to be “refreshed” at a regular interval



Why do we widely use DRAM given some of its unappealing properties?

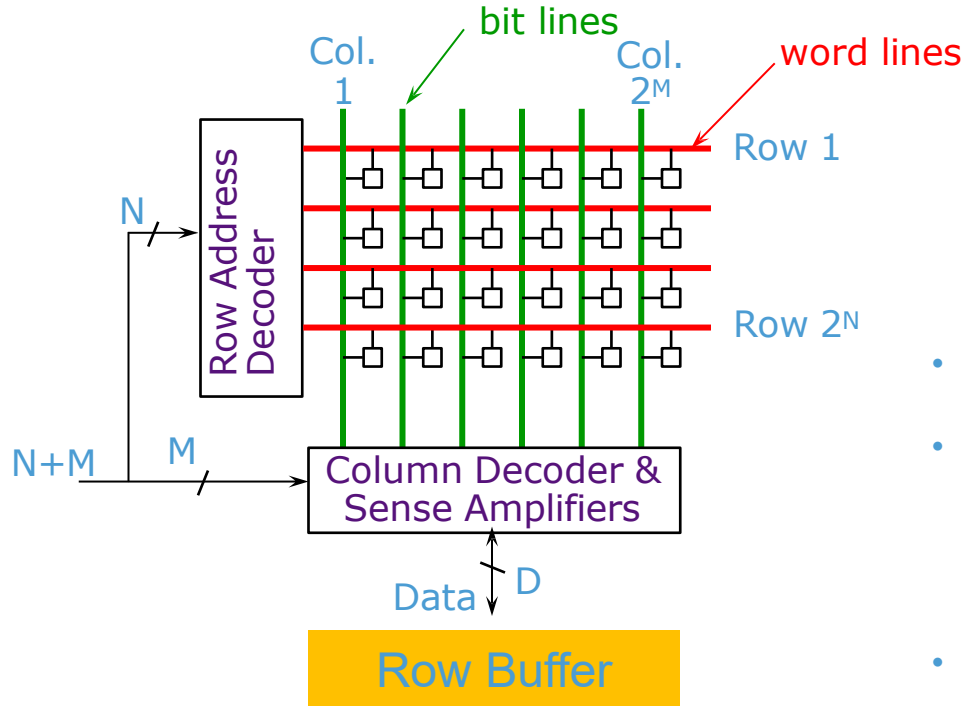
- Speed (2-10x slower than SRAM)
- Density (20x denser than SRAM)
- Cost (~100x cheaper per MB)

DRAM Architecture



- Bits stored in 2-dimensional arrays on chip
- Question: why read the entire row?

DRAM Refresh



- How do we refresh?
- Performance penalty of refresh
 - In an 8Gb memory, upwards of 10% of time is spent in refresh!
- The common refresh interval: **64ms**

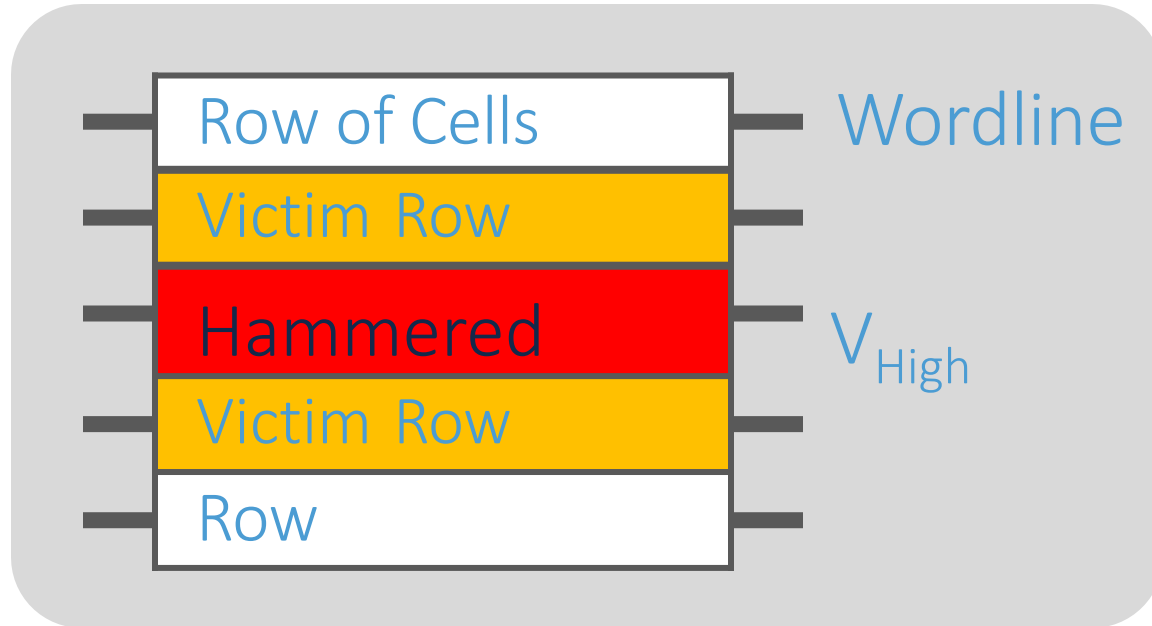
Aside: Cold Boot Attacks

	Seconds w/o power	Error % at operating temp.	Error % at -50°C
SDRAM (1999)	60	41	(no errors)
	300	50	0.000095
DDR (2001)	360	50	(no errors)
	600	50	0.000036
DDR (2003)	120	41	0.00105
	360	42	0.00144
DDR2 (2007)	40	50	0.025
	80	50	0.18



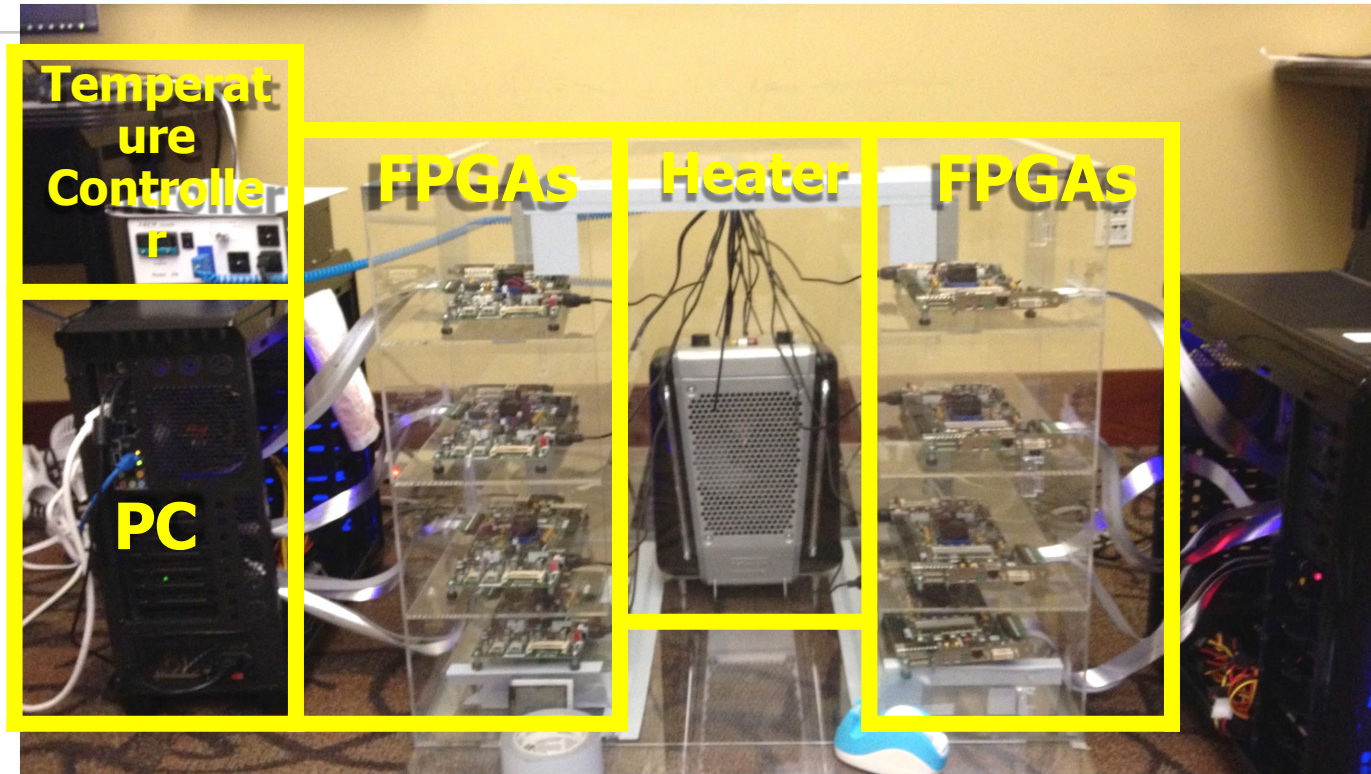
Halderman et al.; Lest We Remember: Cold Boot Attacks on Encryption Keys; USENIX Security'08

See RowHammer Again



Observation: Repeatedly accessing a row enough times **between refreshes** can cause disturbance errors in nearby rows

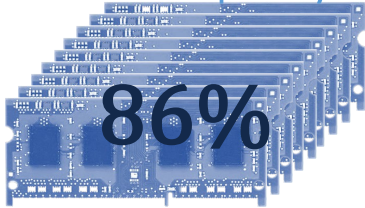
Infrastructures to Understand Rowhammer



Kim et al; Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors; ISCA'14

Most DRAM Modules Are Vulnerable

A company

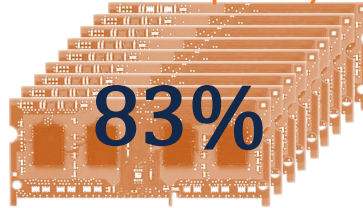


(37/43)

Up to

1.0×10^7 errors

B company

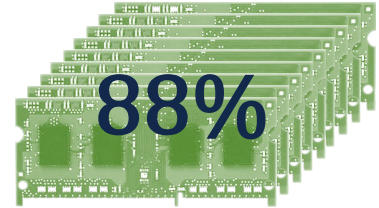


(45/54)

Up to

2.7×10^6 errors

C company



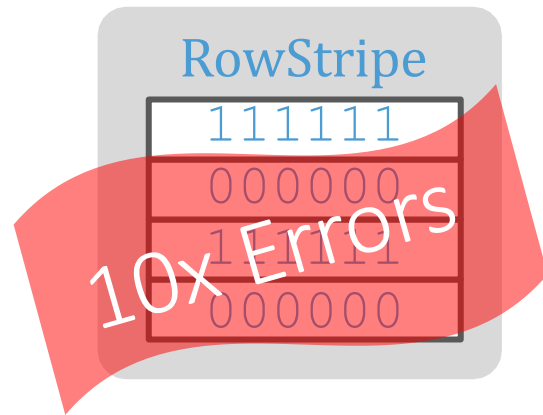
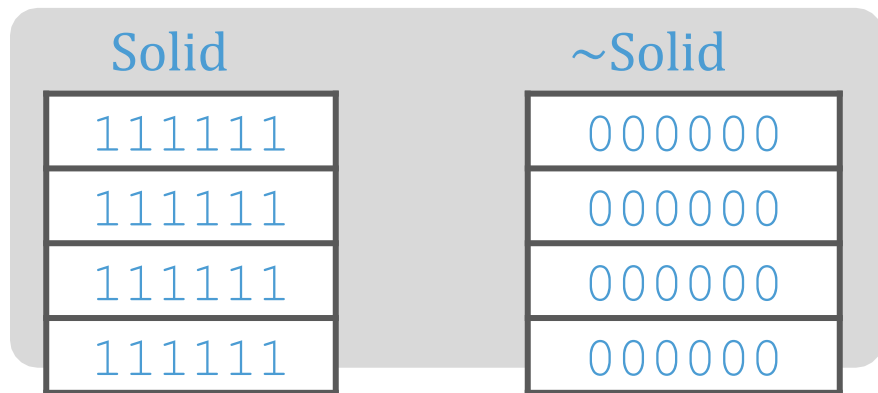
(28/32)

Up to

3.3×10^5 errors

RowHammer Characteristics

- Highly local nature of the bit-flipping capability
- Bit flips are reproducible
- The probability of bitflips are data-dependent

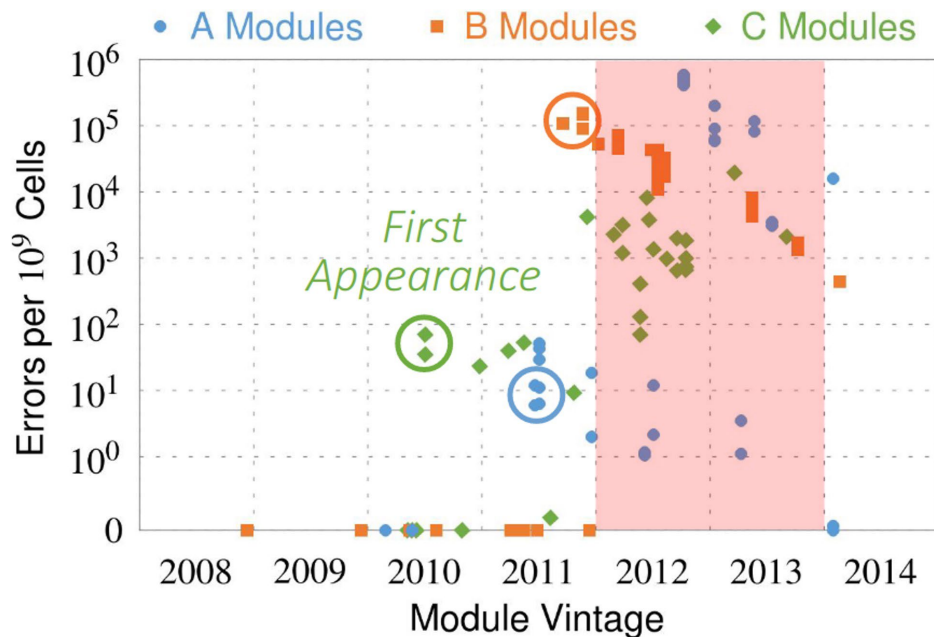


Study RowHammer Characteristics

- Highly local nature of the bit-flipping capability
- Bit flips are reproducible
- The probability of bitflips are data-dependent

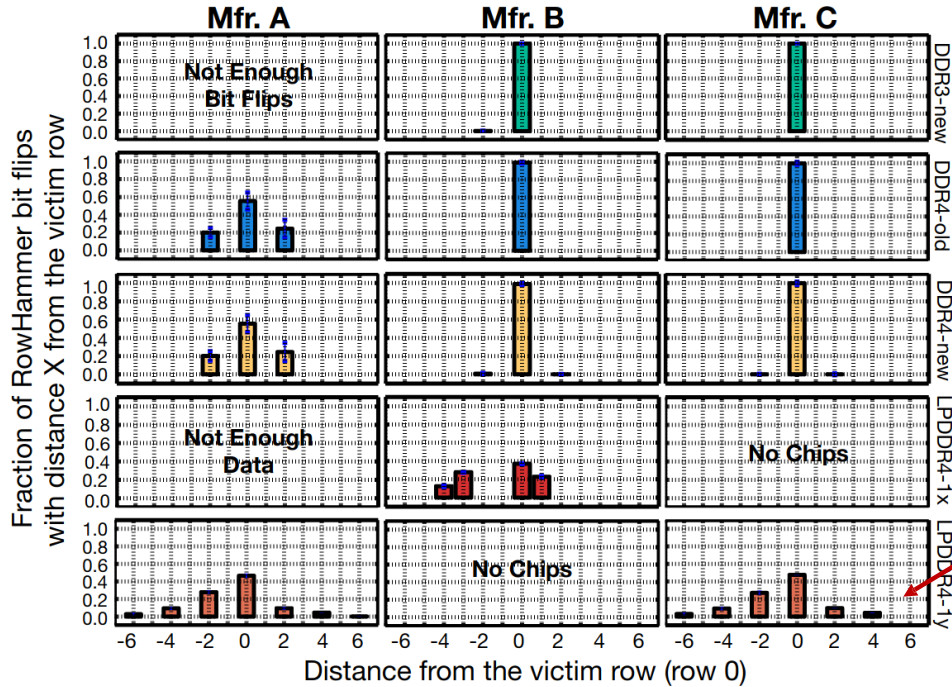
- More advanced DRAM technologies suffer more from this disturbance effect

Density Trends



- As DRAM gets physically denser, it becomes even **more vulnerable!**
 - Trend continues with DDR4
- Only a few thousand hammer iterations are required on modern DRAM to cause a bit-flip

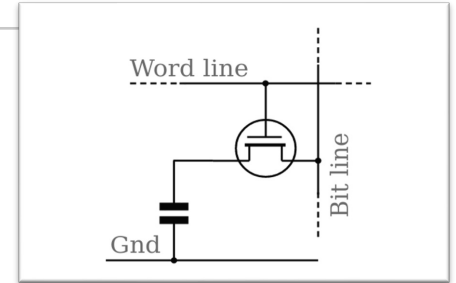
Density Trends



Denser DRAM also can result in flips in rows which are *not directly adjacent* to the attacker

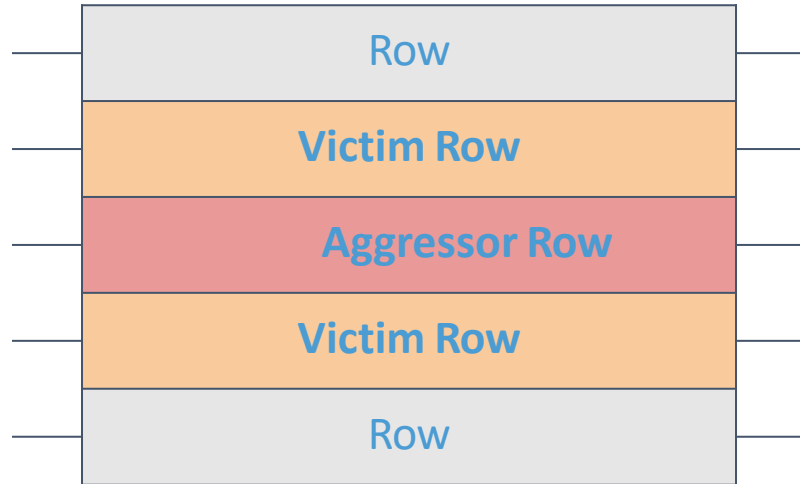
Why Is Rowhammer Happening?

- DRAM cells are too close to each other
 - They are not electrically isolated from each other
- Access to one cell affects the value in nearby cells
 - Due to **electrical interference** between the cells and wires used for accessing the cells
 - Also called cell-to-cell coupling/interference
- Example: When we activate (apply high voltage) to a row, an adjacent row gets slightly activated as well
 - Vulnerable cells in that slightly-activated row lose a little bit of charge
 - If row hammer happens enough times, capacitor's charge in such cells gets drained



RowHammer Attacks in Practice

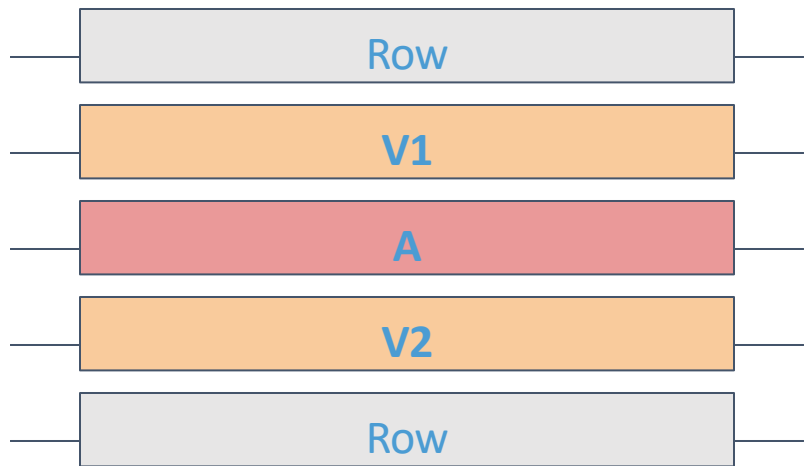
- Aggressor Row = Hammered Row



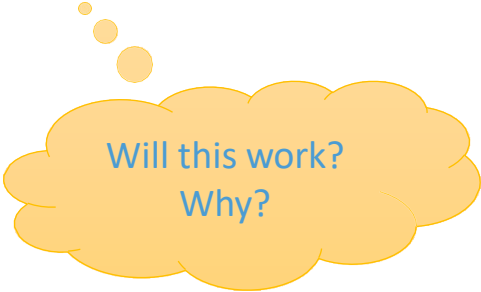
Challenges:

1. How to hammer? Need to access aggressor row enough times between refreshes.
2. Address mapping. How can we find addresses that map to neighboring rows?
3. How do we map victim's data to vulnerable cells?

Hammer Attempt #1: repeat accesses



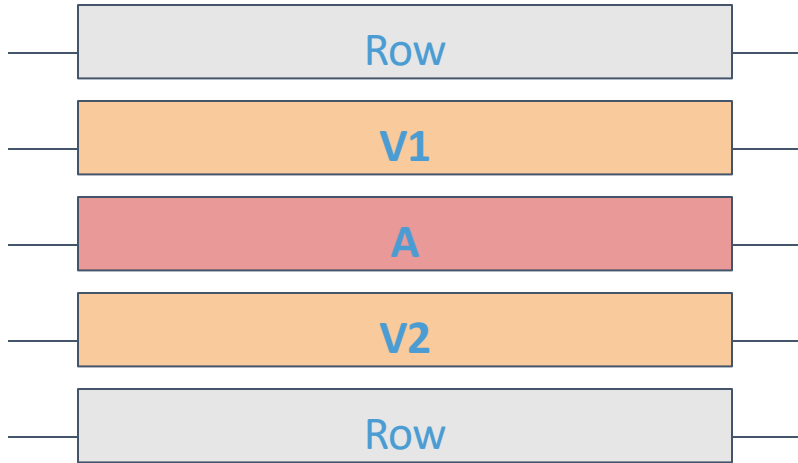
```
loop:  
  mov (A), %eax  
  
  mfence  
  jmp loop
```



Will this work?
Why?

No. Because we will hit the cache.

Hammer Attempt #2: use cflush



```
loop:
```

```
    mov (A), %eax
```

```
    cflush (A)
```

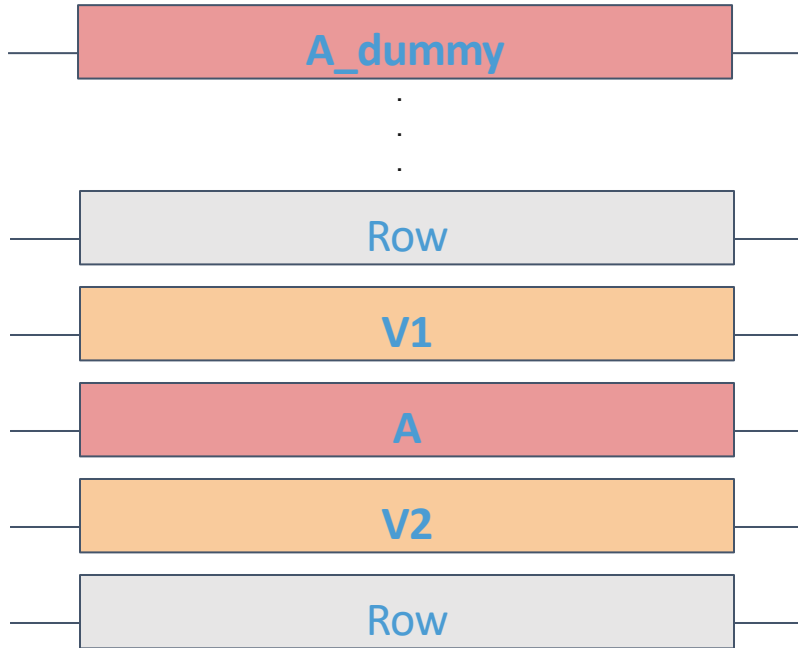
```
    mfence
```

```
    jmp loop
```

Will this work?
Why?

No. Because we will hit the row buffer.

Hammer Attempt #3: force row open/close



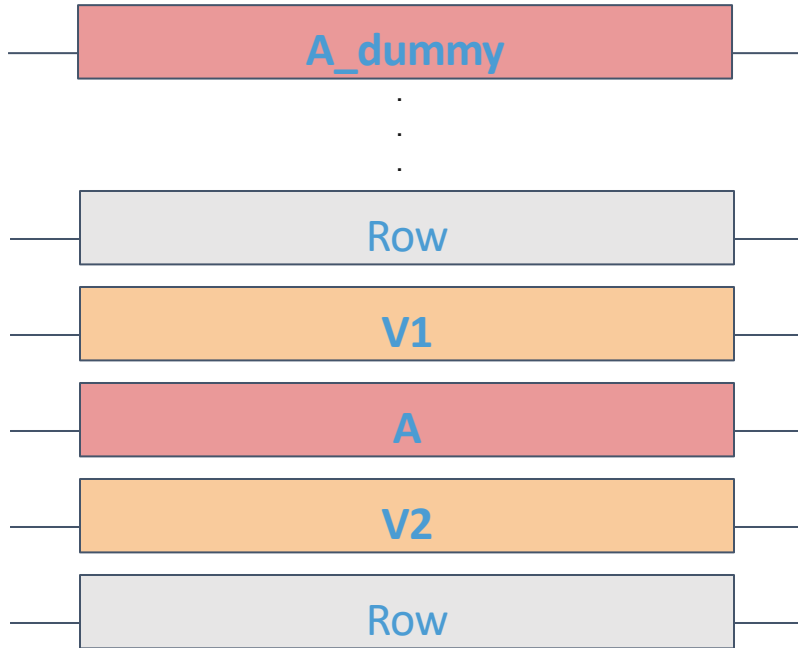
loop:

```
mov (A), %eax  
mov (A_dummy), %ecx
```

```
clflush (A)  
clflush (A_dummy)
```

```
mfence  
jmp loop
```


“Single-Sided” Rowhammer



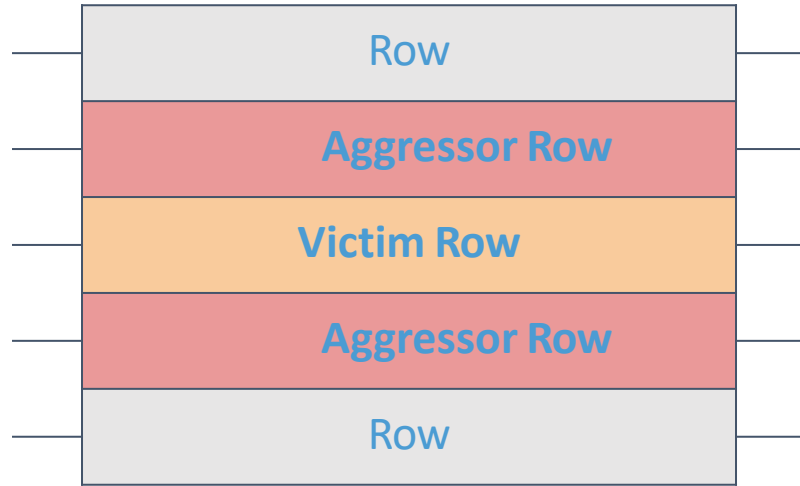
loop:

```
mov (A), %eax  
mov (A_dummy), %ecx
```

```
clflush (A)  
clflush (A_dummy)
```

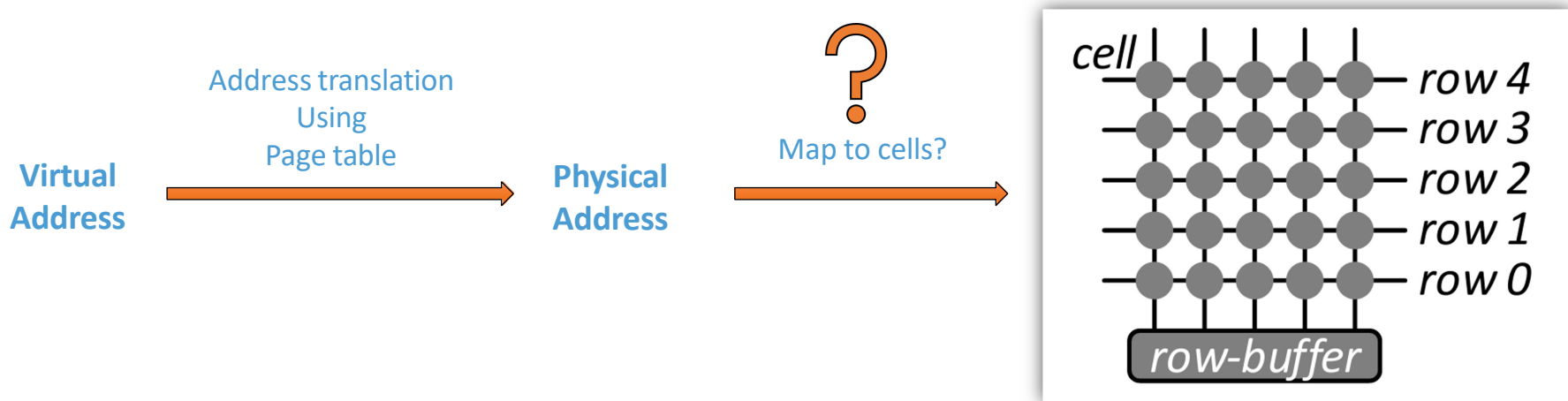
```
mfence  
jmp loop
```

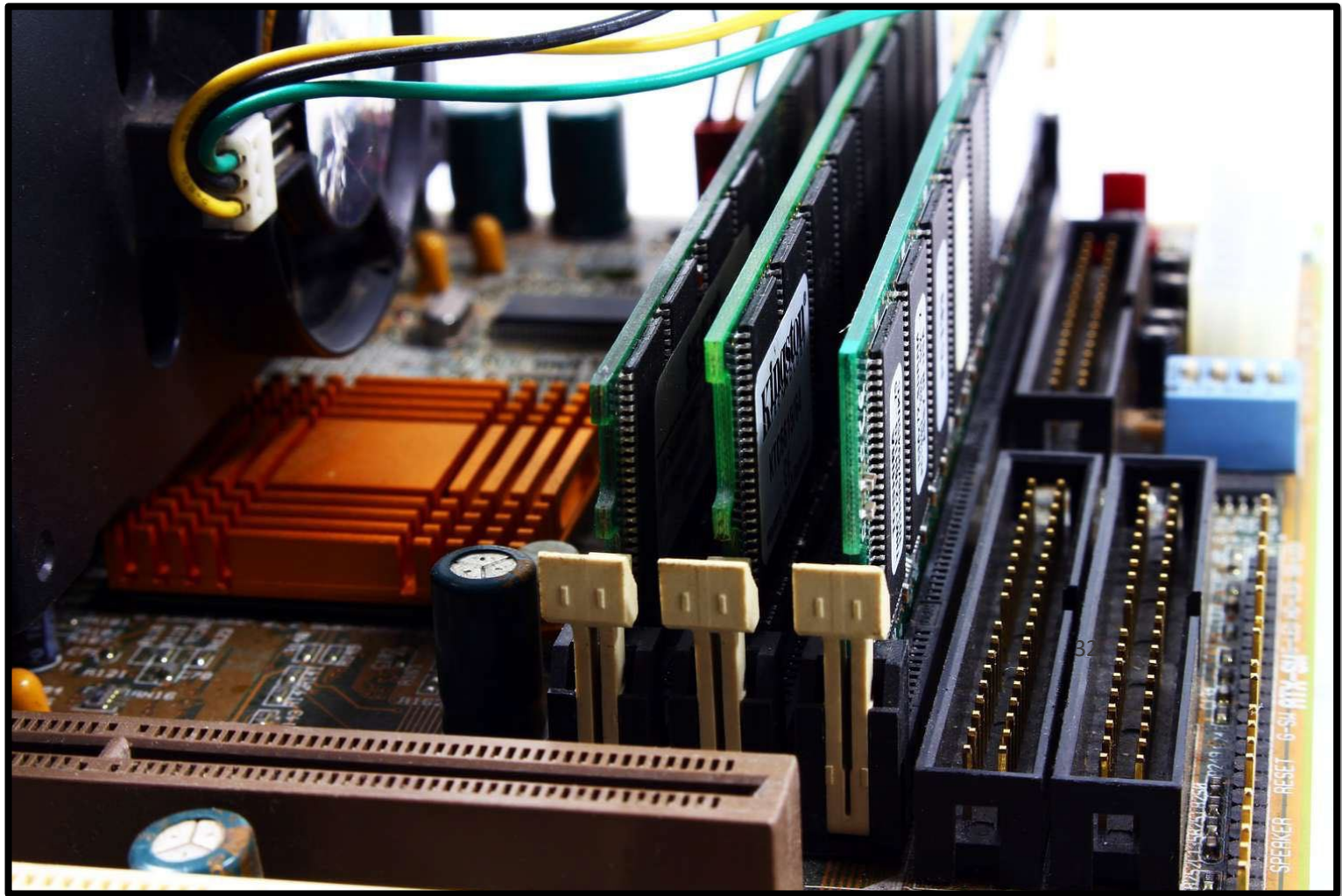
“Double-Sided” Rowhammer



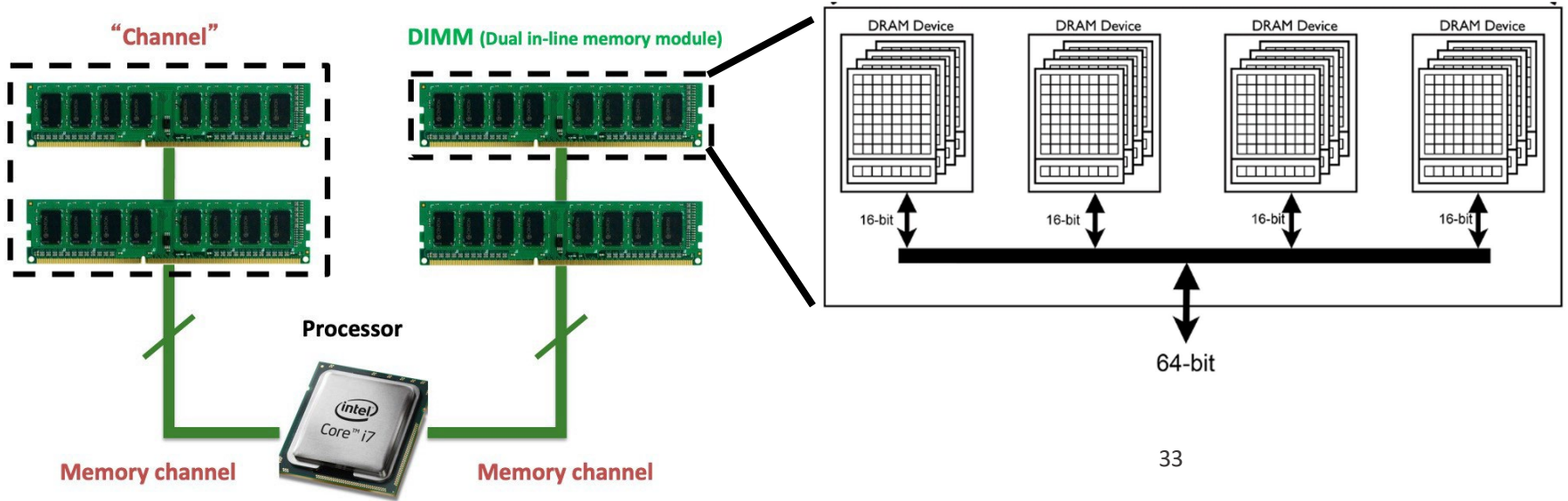
- Increase the stress:
- Repeatedly accessing both adjacent rows *dramatically* increases the error rate of the victim row

Challenge #2: DRAM Addressing

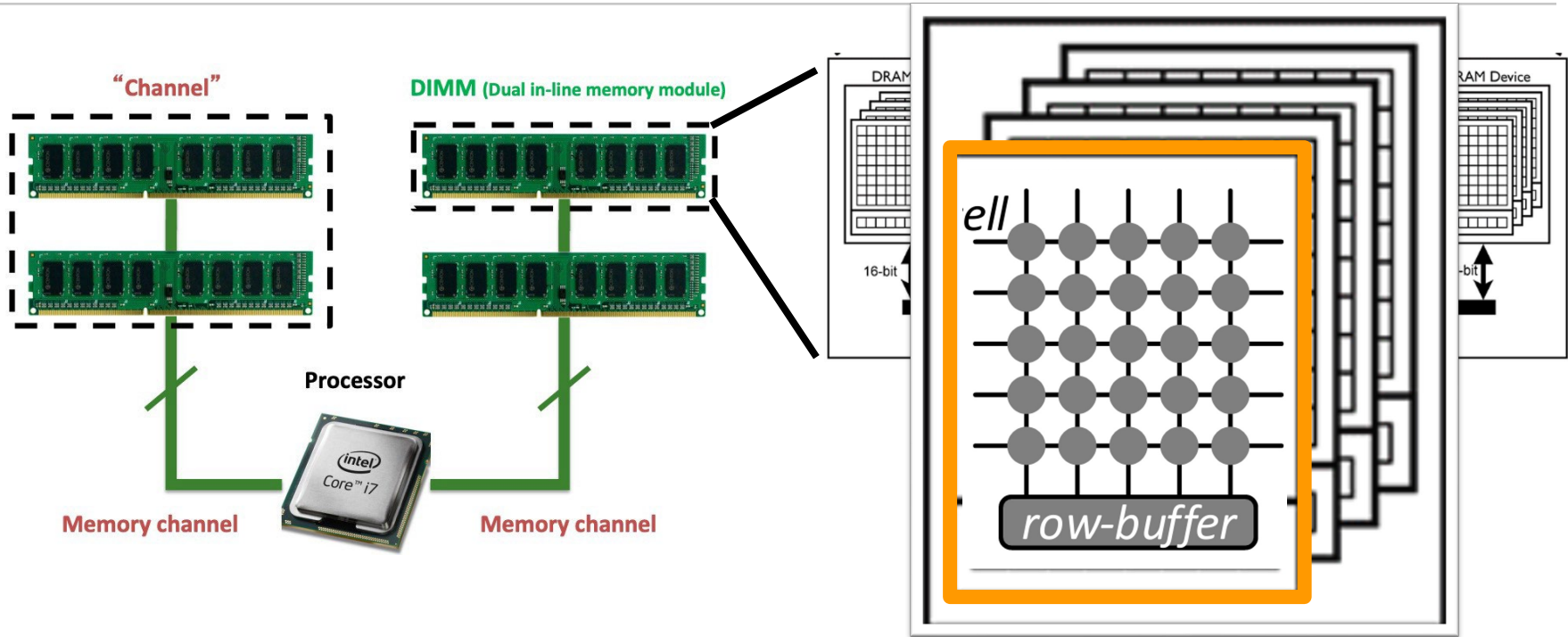




DRAM Organization: Top-down View



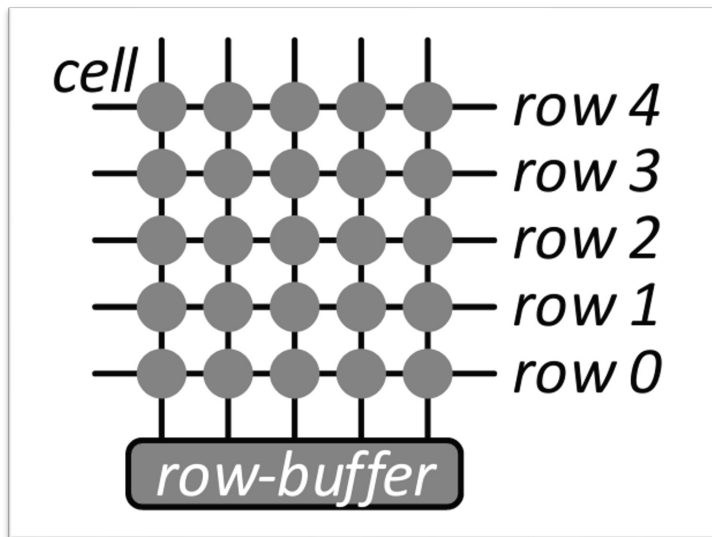
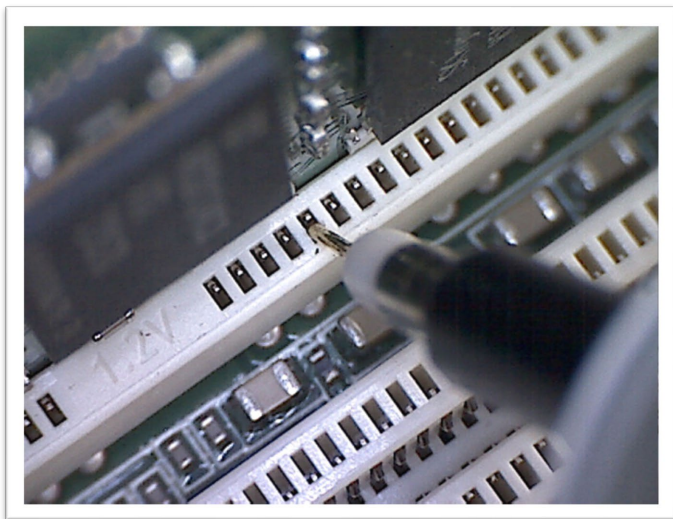
DRAM Organization: Top-down View



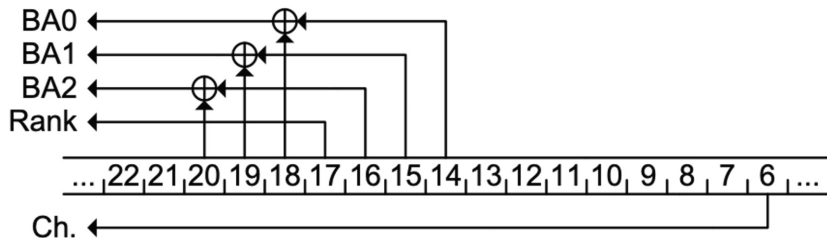
Channel -> DIMM -> Rank -> Bank -> Row/Column

Reverse Engineer the Mapping

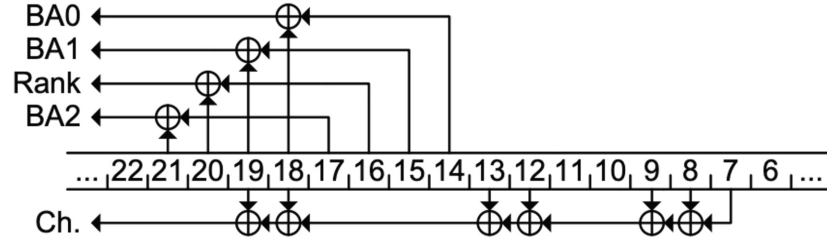
- Approach #1: Physical Probe
- Approach #2: Timing Side Channel via Row Buffer



Address Mapping Examples



(a) Sandy Bridge – DDR3 [23].



(b) Ivy Bridge / Haswell – DDR3.

Rowhammer Attacks

Native Client (NaCl) Sandbox Escape

- NaCl is a sandbox for running native code (C/C++)
- Runs a “safe” subset of x86, statically verifying an executable
- Use bit flips to make an instruction sequence unsafe!

Example “Safe” Code:

```
andl $~31, %eax // Truncate address to 32 bits
                // and mask to be 32-byte-aligned.
addq %r15, %rax // Add %r15, the sandbox base address.
jmp  *%rax      // Indirect jump.
```

Native Client (NaCl) Sandbox Escape

We can flip bits to allow for (unsafe) non 32-byte-aligned jumps!

Exploited “Safe” Code:

```
andl $~31, %ecx // Truncate address to 32 bits
                // and mask to be 32-byte-aligned.
addq %r15, %rax // Add %r15, the sandbox base address.
jmp *%rax       // Indirect jump.
```


Other Attacks

- Virtual machine takeover
 - Use page de-duplication to corrupt host machine
- OpenSSH attacks
 - Overwrite internal public key with attacker controlled one
 - Read private key directly (RAMBleed)
- Drammer
 - Rowhammer privilege escalation on ARM
 - Utilizes determinism in page allocation to target vulnerable DRAM rows
- Rowhammer.js
 - Remote takeover of a server vulnerable to rowhammer

Without memory integrity, *any* software-based security mechanism is insecure!

Rowhammer Mitigations?

- Manufacturing “better” chips

cost

- Increasing refresh rate

Performance, power

- Error Correcting Codes

cost, power

- Targeted row refresh (TRR) - Used in DDR4!

cost, power, complexity

- Retiring vulnerable cells

cost, power, complexity

- Static binary analysis

security

- User/kernel space isolation in physical memory

Error Correcting Codes (ECC)

- **Basic Idea:** Store extra *redundant* bits to be used in case of a flip!
- **Naive Implementation:** Store multiple copies and compare
- **Actual Implementation:** Hamming codes

Hamming codes allow for *single-error* correction, double error detection (aka **SECDED**)

How about more than 2-bit flips?



Takeaways

Reliability Concerns → Security Implications



THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL