Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs *work done while at Carnegie Mellon University

ROWHAMMER

Background, Problems & Goals

Novelty

Key Approach & Ideas

Test Mechanics

Takeaway

Summary

Strengths & Weaknesses

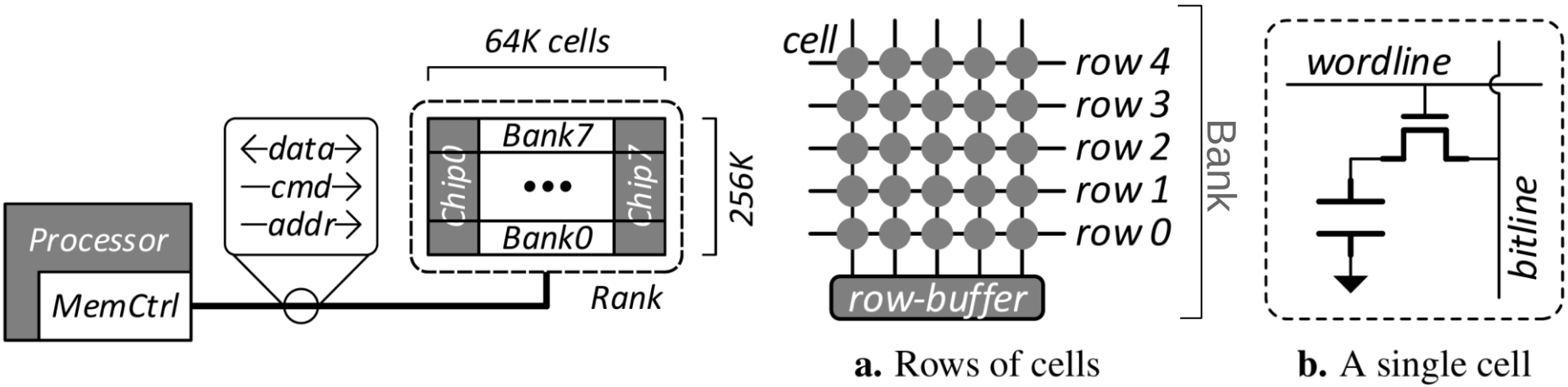
Further Work, Thoughts, Ideas

Discussion

Summary

- As DRAM cells get smaller and placed closer together, physical properties can make them interact in undesired ways
  - Toggling the same row in a short time can induce bit flips
  - Most modules from the 3 main manufacturers affected (2014)
- Serious security implications
  - Memory isolation is a key property of a secure system
- Proposed solution:
  - Probabilistic adjacent row activation (PARA)

DRAM Background

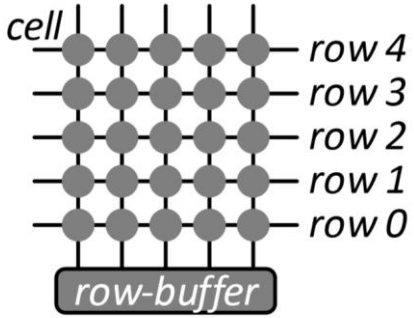


2GB Rank of 8 chips
8 Banks of 32K rows
64K cells/row (8KB)

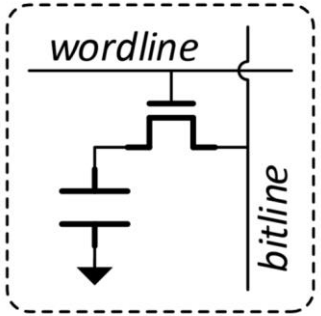
Accessing DRAM

1. `ACTIVATE`: raise wordline -> transfers charge to row-buffer
2. Read/Write: read/write into row-buffer
3. `PRECHARGE`: before reading new row, lower wordline and clear row-buffer

- Cells leak charge: Refresh every 64ms
- Refresh: `ACT; PRE`



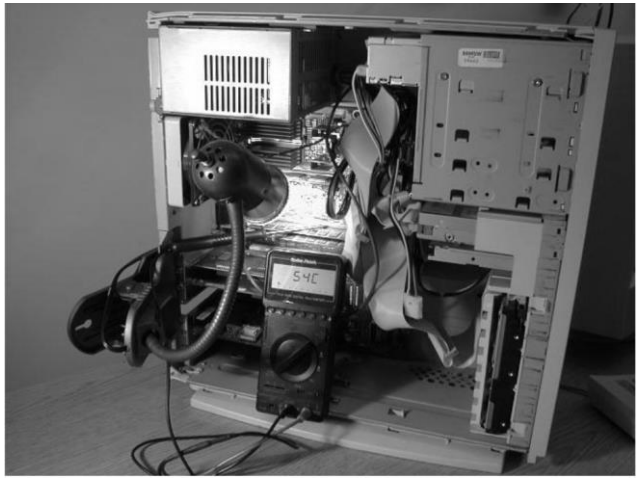
a. Rows of cells

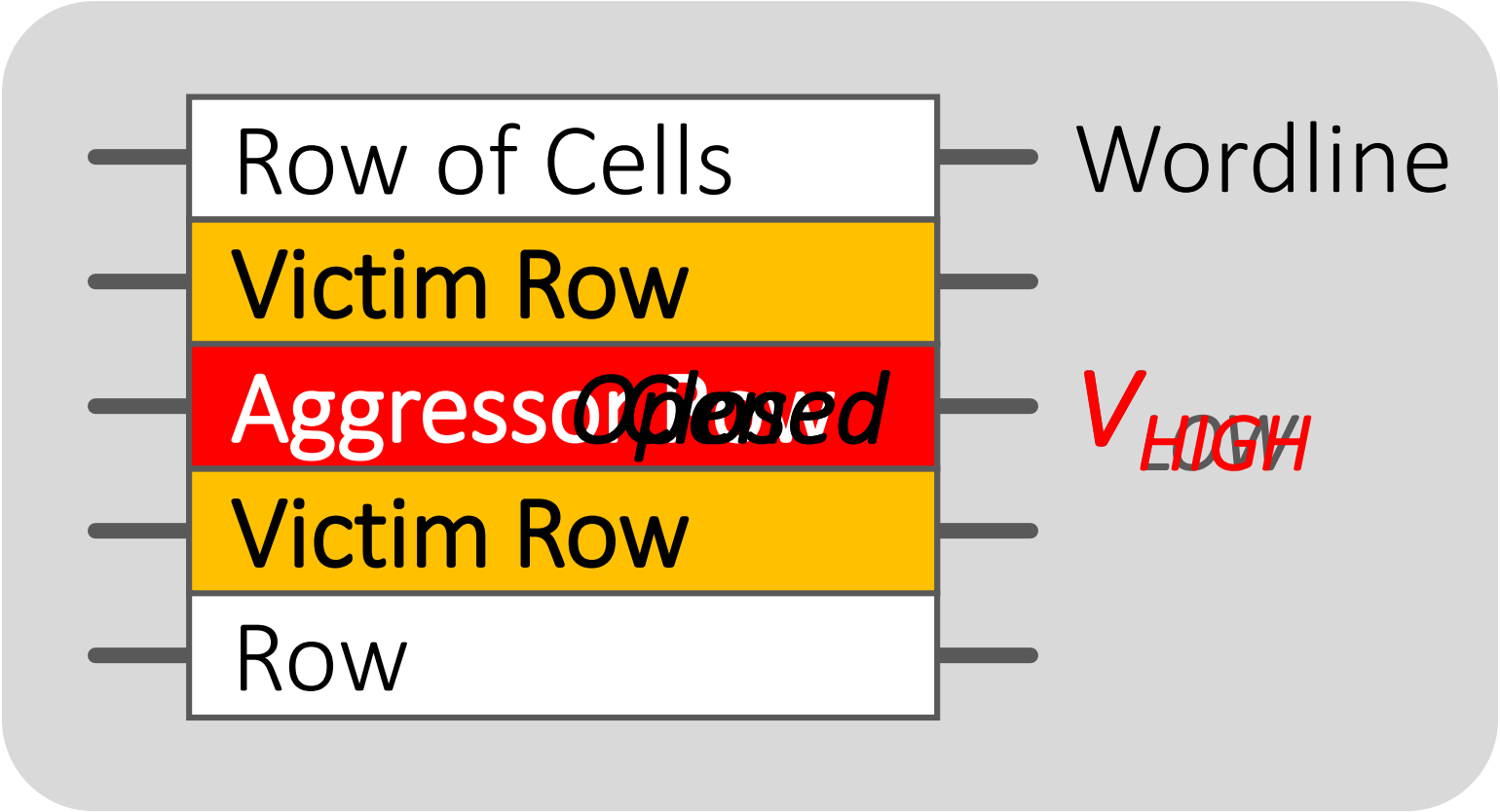


b. A single cell

Disturbance Errors

- Disturbance errors are not new
  - Smaller cells lead to smaller noise margin
  - Can be caused by radiation or even heat
- Cosmic and terrestrial single-event radiation effects in dynamic random access memories (L.W. Massengill, 1996)
- Using Memory Errors to Attack a Virtual Machine (Govindavajhala, Appel, 2003)
  - Escape a java VM using a lamp
- Rowhammer makes it possible without physical access





"Rowhammer and Beyond“ - Onur Mutlu

Background, Problems & Goals

Novelty

Key Approach & Ideas

Test Mechanics

Results

Summary

Strengths & Weaknesses

Further Work, Thoughts, Ideas

Takeaway

Discussion

Novelty

- Demonstration on real, off-the shelf hardware
- Shows how widespread the problem of row-hammer is
- Called a lot of attention to this problem
- First example of a widespread hardware security issue
- Novel solution with probabilistic activation

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Takeaway

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

Demonstration on a real system

- X and Y on different rows in the same bank
- Test run on Sandy Bridge (2011), Ivy Bridge (2012), Haswell (2013) and AMD Piledriver (2012) using normal off the shelf DDR3 Ram sticks
- Errors observed in every one of these architectures

```
1 code1a:
2   mov (X), %eax
3   mov (Y), %ebx
4   clflush (X)
5   clflush (Y)
6   mfence
7   jmp code1a
```

a. Induces errors

Experimental Setup

- Test DRAM modules from the 3 main manufacturers
- Use an FPGA based test platform to evaluate the modules
- Identify under which condition errors occur
  - Manufactured date and process
  - Access patterns
  - Refresh timing
  - Data patterns
  - Temperature
- Find a correlation of aggressor and victim rows

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Takeaway

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

FPGA based testing platform

- Run DDR3 memory controller and a test engine on an FPGA in a controlled environment
- Test 129 memory modules from 3 main Manufacturers
- Most tests done with:
  - `AI=55ns, RI=64ms -> N=2'330'000`
- Further testing with the “worst” memory modules from each manufacturer

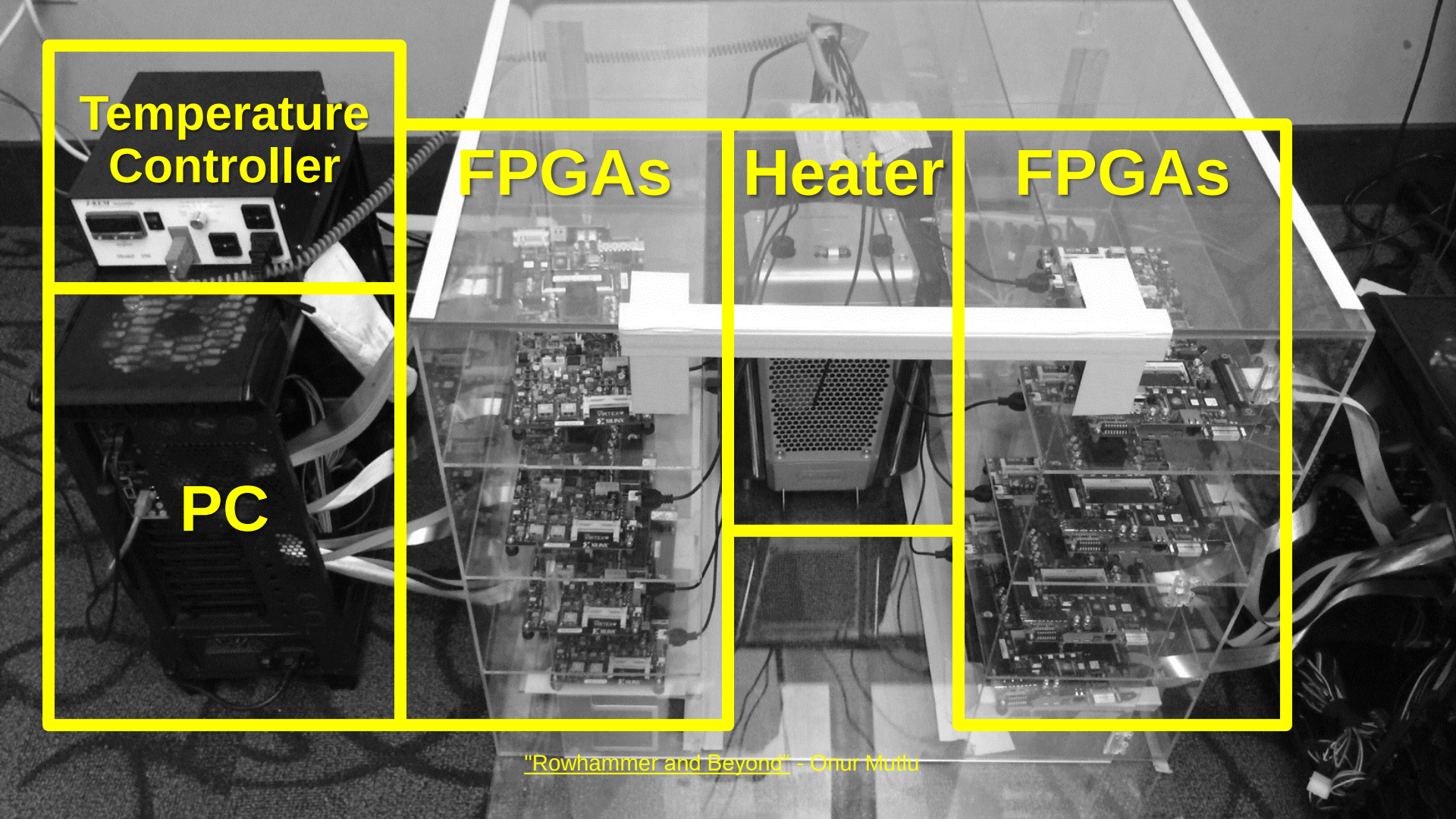
```
1  TestBulk(AI, RI, DP)
2    setAI(AI)
3    setRI(RI)
4    N ← (2 × RI)/AI
5
6    writeAll(DP)
7    for r ← 0···ROW_MAX
8      for i ← 0···N
9        ACT r^th row
10       READ 0^th col.
11       PRE r^th row
12   readAll()
13   findErrors()
```

a. Test all rows at once

```
1  TestEach(AI, RI, DP)
2    setAI(AI)
3    setRI(RI)
4    N ← (2 × RI)/AI
5
6    for r ← 0···ROW_MAX
7      writeAll(DP)
8      for i ← 0···N
9        ACT r^th row
10       READ 0^th col.
11       PRE r^th row
12     readAll()
13     findErrors()
```

b. Test one row at a time

`AI`: Activation Interval
`RI`: Refresh Interval
`DP`: Data Pattern
`N`: Number of Activations

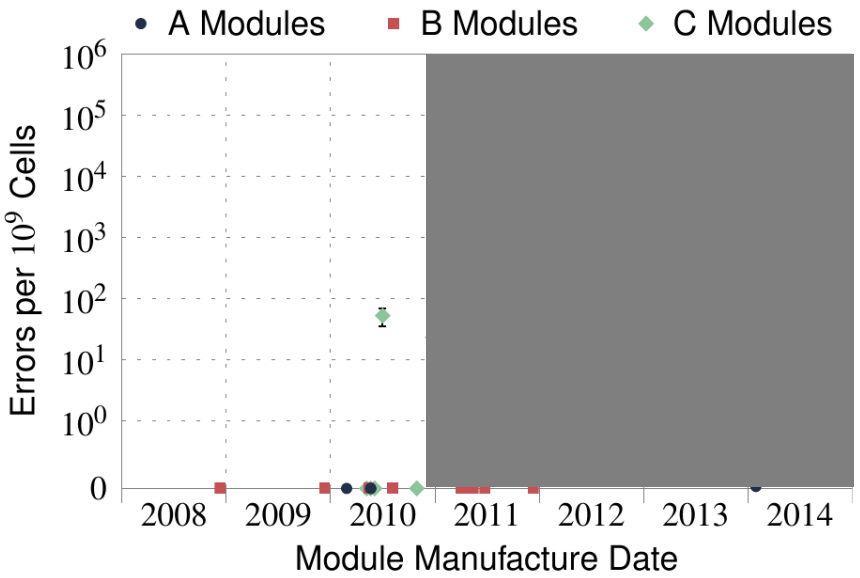

Temperature Controller
PC
FPGAs
Heater
FPGAs

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Key Results
Takeaway

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

Disturbance Errors are a recent problem

All modules from 2012 – 2013 are vulnerable



Access Pattern

Repeatedly toggling the same wordline is the cause of disturbance errors

Access Pattern	Disturbance Errors?
1. $(open–read–close)^N$	**Yes**
2. $(open–write–close)^N$	**Yes**
3. $open–read^N–close$	No
4. $open–write^N–close$	No

```
1 code1a:
2   mov (X), %eax
3   mov (Y), %ebx
4   clflush (X)
5   clflush (Y)
6   mfence
7   jmp code1a
```

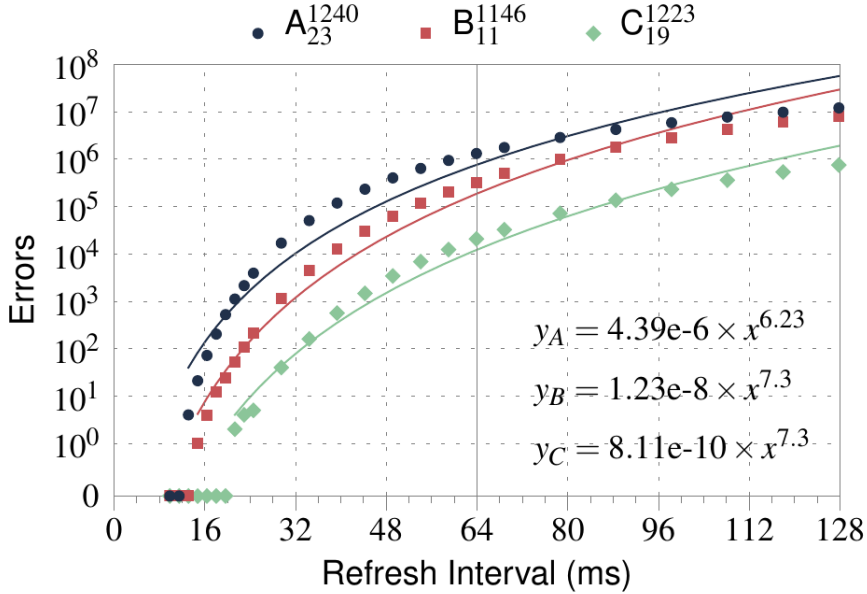
a. Induces errors

```
1 code1b:
2   mov (X), %eax
3   clflush (X)
4
5
6   mfence
7   jmp code1b
```

b. Does not induce errors

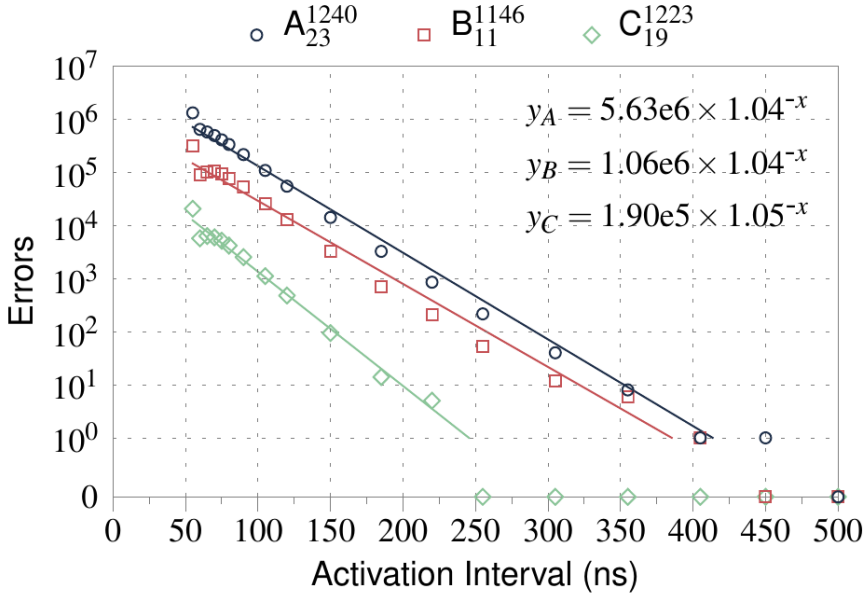
Refresh interval

Faster refresh intervals lead to fewer errors



Activation Interval

Less frequent row activations lead to fewer errors



Errors per row

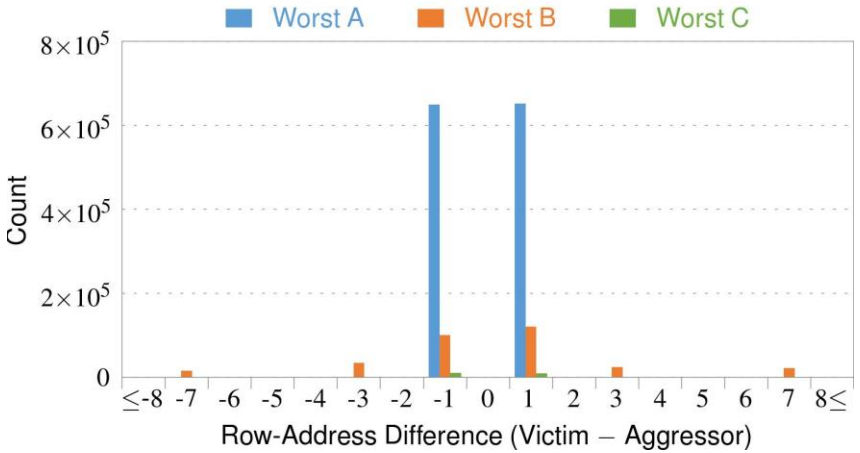
Some rows contain 2 or more victim cells

- Serious implications for ECC

Module	*Number of 64-bit words with X errors*			
	$X = 1$	$X = 2$	$X = 3$	$X = 4$
A_{23}	9,709,721	**181,856**	**2,248**	**18**
B_{11}	2,632,280	**13,638**	**47**	0
C_{19}	141,821	**42**	0	0

Victim and aggressor addresses

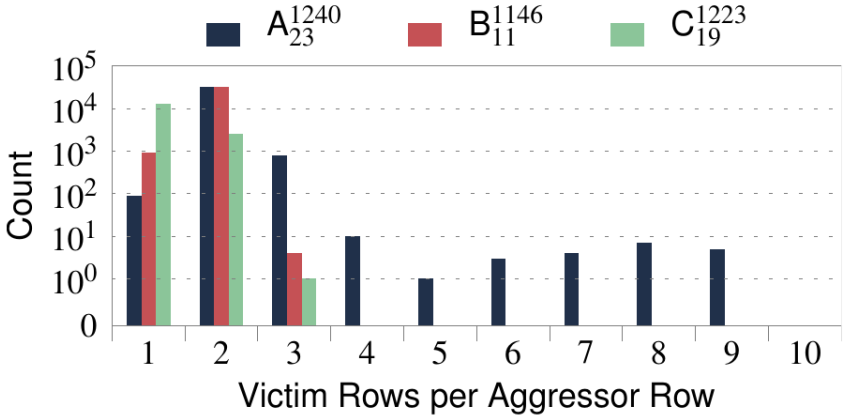
Victim and aggressor rows are most likely adjacent



"Rowhammer and Beyond" - Onur Mutlu

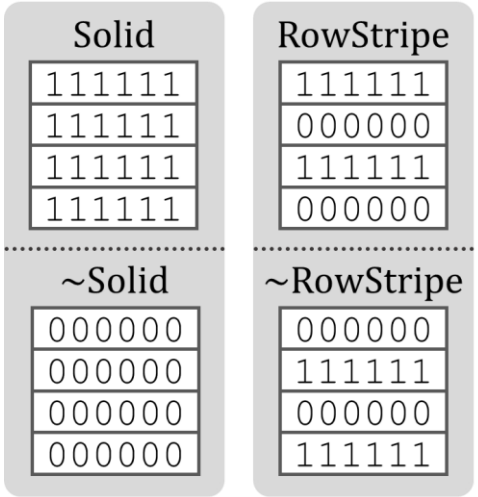
Victim and Aggressor rows

Most aggressor rows have one or two victim rows



Data Pattern dependence

- Charge always depleted
  - `1` → `0` for true cells (charged = `1`)
  - `0` → `1` for anti cells (charged = `0`)
- Cells flips in `RowStripe` ↔ does not flip in `~RowStripe`
- `RowStripes` have 10x more errors



Other results

- Errors not temperature dependent
- Victim cells are not the same as “leaky” cells
- Errors are repeatable (>70% of cells had errors in every test iteration)

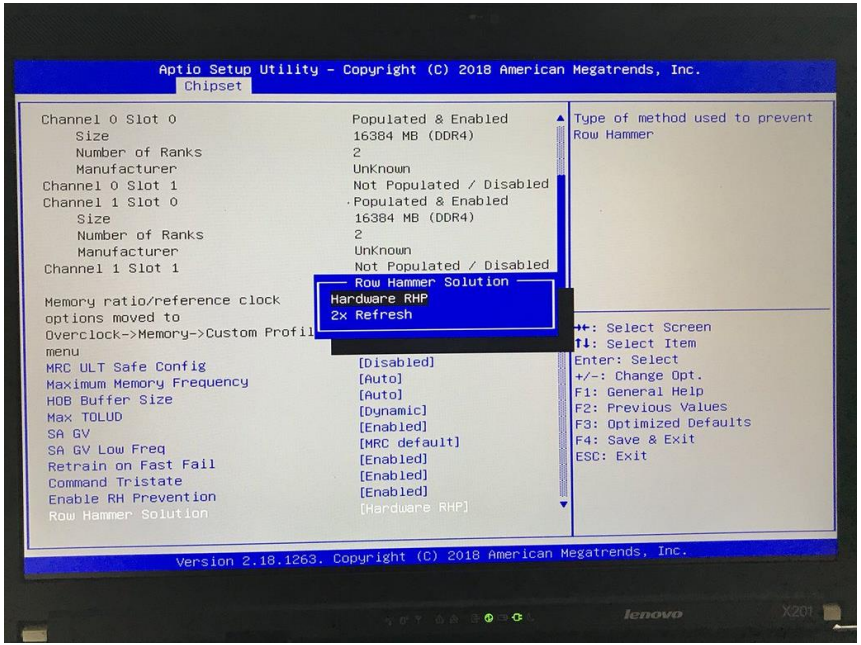
Potential solutions

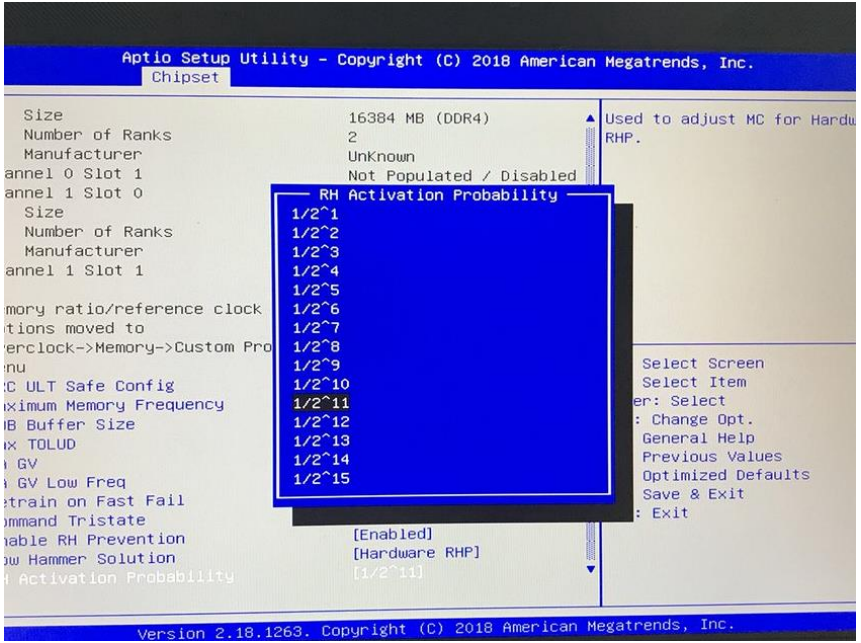
1. Make better Chips
2. Correct Errors
3. Refresh all rows frequently
4. Retire cells (manufacturer)
5. Retire cells (user)
6. Identify “hot” rows and refresh neighbours
7. probabilistic adjacent row activation (PARA)

Probabilistic adjacent row activation (PARA)

- On row close, refresh adjacent rows with a low probability ($p \ll 1$)
- Extremely low error probability
- Stateless
- No expensive hardware structures
- Low performance overhead in simulation ($p=0.005$, 0.192% avg. overhead)
- Memory controller has to know the physical layout

Probabilistic mitigation in the wild





Lenovo x201 with 8th gen intel motherboard

https://twitter.com/isislovecruft/status/1021939922754723841

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Takeaway

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

Takeaway

- Software security can be broken by hardware
- Intrinsic physical property of the hardware design
- To totally mitigate this, a hardware redesign is probably necessary

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Takeaway

Strengths & Weaknesses

Further Work, Thoughts, Ideas
Discussion

Strengths

- First paper to show how widespread Rowhammer is
- Thorough tests on a very wide range of commodity DRAM chips
- A practical long-term solution is proposed
  - Low overhead and hardware cost
- Sparked a lot of further research in the topic
- Well written paper

Weaknesses

- Para:
  - The controller needs to know the physical row layout of the chips, which is not the case at the moment
  - Needs hardware redesign
  - Probabilistic solution does not guarantee protection
- The only immediate solution presented is refreshing the rows 8 times more often, which has considerable performance overhead
- No software solutions mentioned
- Does not show any specific exploitation

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Takeaway

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

Exploiting the DRAM Rowhammer bug to gain kernel privileges (2015, Seaborn et al., Google Project Zero)

- First real life exploitation of Rowhammer
- Double-sided hammering
- Modify page table entry to gain kernel privileges

ECCploit: Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks (Cojocar et. al 2019)

- ECC DDR3 Memory exploit
- Flip multiple bits in order to circumvent ECC
- ECC is not a Solution to Rowhammer

Rowhammer.js: A remote software induced fault attack in js (D. Gauss et al. 2015)

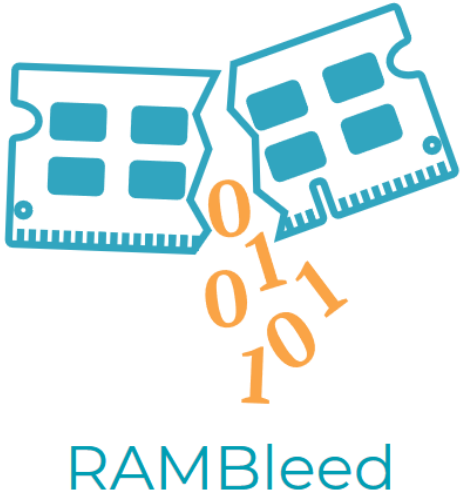
- Cause cache flush without using `CLFLUSH`
- Fast cache eviction with normal memory access
- A malicious website can induce bit flips

```
1 code1a:
2   mov (X), %eax
3   mov (Y), %ebx
4   clflush (X)
5   clflush (Y)
6   mfence
7   jmp code1a
```

a. Induces errors

RAMbleed: Reading Bits in Memory Without Accessing Them (Kwong et al., 2020)

- Read Memory by data dependence of bit flips
- Demonstration: Extracted an RSA-2048 Private Key from OpenSSH



Other security exploits

- One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation (Y. Xiaou et al., USENIX Sec. 2016)
- Flip Feng Shui: Hammering a Needle in the Software Stack (K. Razavi et al. 2016)
- Drammer: Deterministic Rowhammer attacks on mobile platforms (V. van der Veen et al., CCS, 2016)
- Another Flip in the Wall of Rowhammer Defense (D. Gruss et al., 2018)
- Throwhammer: Rowhammer Attacks over the Network and Defenses (A. Tatar et al., USENIX Sec., 2018)

Protection against Rowhammer

- Short term solution: double the refresh rate of DRAM (Apple, HP, Cisco, Lenovo, IBM)
  - Higher power draw, performance overhead
- TRR (targeted row refresh) in DDR4
  - Refresh specific victim rows

ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks (Z. B. Aweke et al., 2016)

- Track DRAM access using existing hardware performance counters
- Detects aggressors and refreshes nearby rows

CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory (F. Brasser et al., 2017)

- Physically isolate memory of kernel- and userspace

GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM (V. van der Veen et al., 2018)

- Mitigate attacks using guard rows on DMA allocated Memory

TWiCe: Time Window Counter Based Row Refresh to Prevent Row-Hammering (E. Lee et al., 2017)

- Low hardware overhead row counters
- Needs new DRAM chips

Background, Problems & Goals
Novelty
Key Approach & Ideas
Test Mechanics
Results
Summary

Strengths & Weaknesses
Further Work, Thoughts, Ideas
Discussion

Discussion

- What do you think of the proposed solutions
- New Solutions?
- New Attacks?
- Have you read another interesting paper on rowhammer?
- Do you think this can be exploited further? (other than dram)

New Solutions

- Add smaller “indicator” cells
  - Get depleted faster than normal cells
  - Indicate a hammer attack and refresh neighbors

- Instead of counters, use capacitors on each row which get charged with each activation
  - When charge exceeds certain threshold, trigger refresh
  - Capacitor also leaks charge if the row is not activated frequently