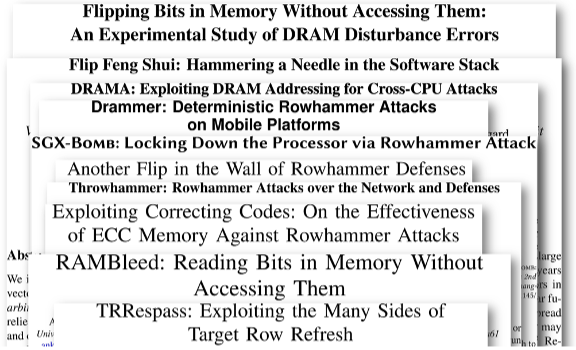
Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers

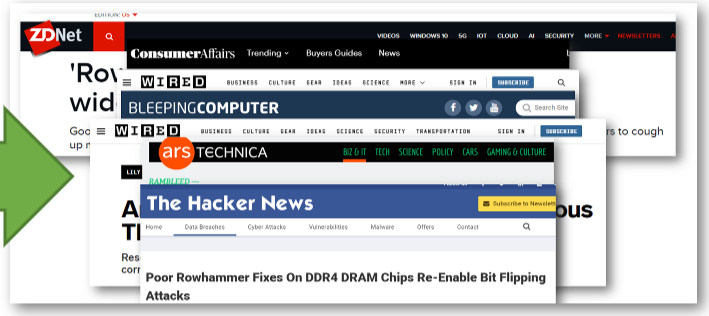
Lucian Cojocar, Jeremie Kim[§†], Minesh Patel[§], Lillian Tsai[‡], Stefan Saroiu, Alec Wolman, and Onur Mutlu[§†]

Microsoft Research, [§]ETH Zurich, [†]CMU, [‡]MIT

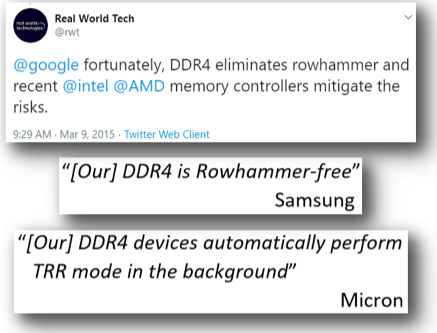
Landscape of Rowhammer



Research Community



Media Coverage



Memory Vendors

Cloud Providers

Our Work

- Develop an **end-to-end** methodology to test if **any** cloud server is susceptible to Rowhammer

- Must overcome two major technical challenges:
  1. Develop instruction sequence to **optimally hammer** DRAM
  2. Develop a strategy to **map row adjacency** inside DIMMs

- Apply methodology on three generations of Intel servers:
  - Broadwell, **Skylake**, Cascade Lake

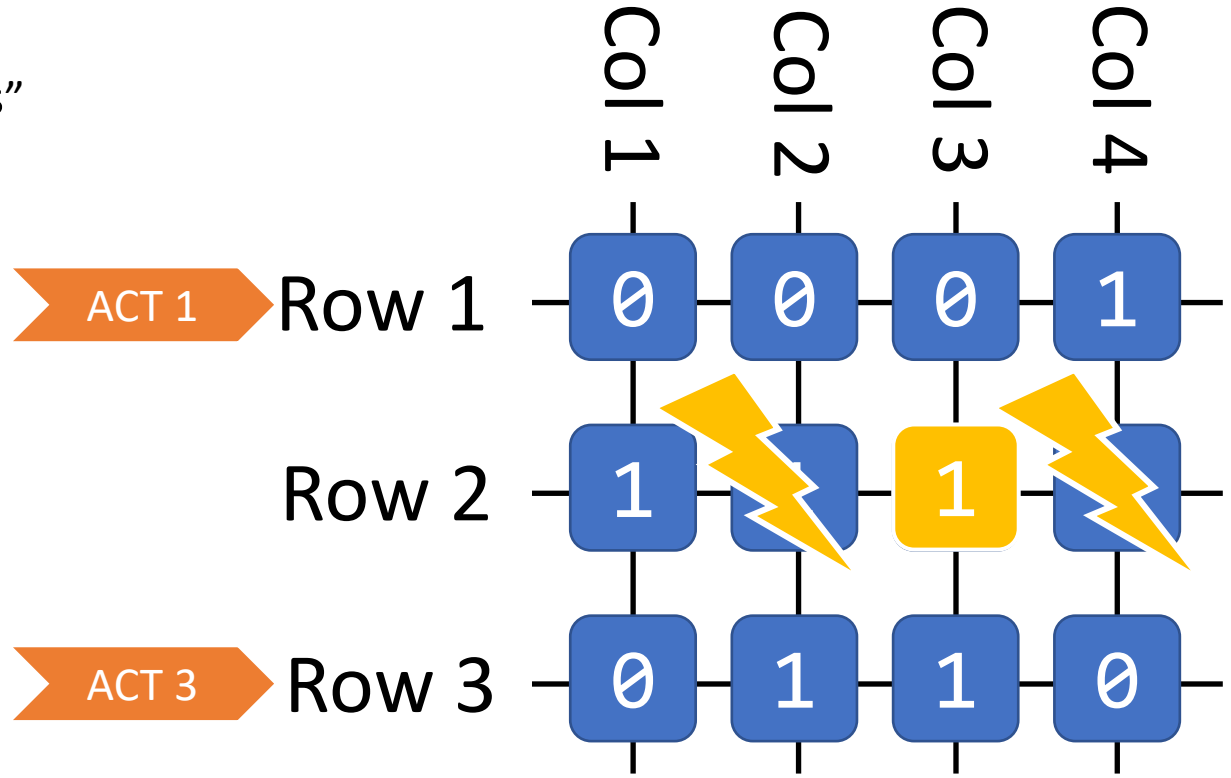
Rowhammer Primer

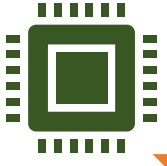
"Flipping Bits in Memory Without Accessing Them:
An Experimental Study of DRAM Disturbance Errors"
Kim et al., ISCA '14

```
loop:
  load A   // Row 1
  load B   // Row 3
  clflush A
  clflush B
```

PC

PC





ACT 3 ACT 1 ACT 3 ACT 1 ACT 3 ACT 1 ACT 3

CPU

DRAM

Two Key Requirements for Rowhammer Testing

To be successful, methodology must create **worst-case** testing conditions for DIMMs

1. Must generate **highest** rate of ACTs on a cloud server

2. Must hammer rows whose cells are **adjacent** inside DRAM

Challenges in Generating Highest Rate of ACTs

- How to measure rate of ACTs of instruction sequences?

➢We used a DDR bus analyzer

DIMM & Bus Analyzer

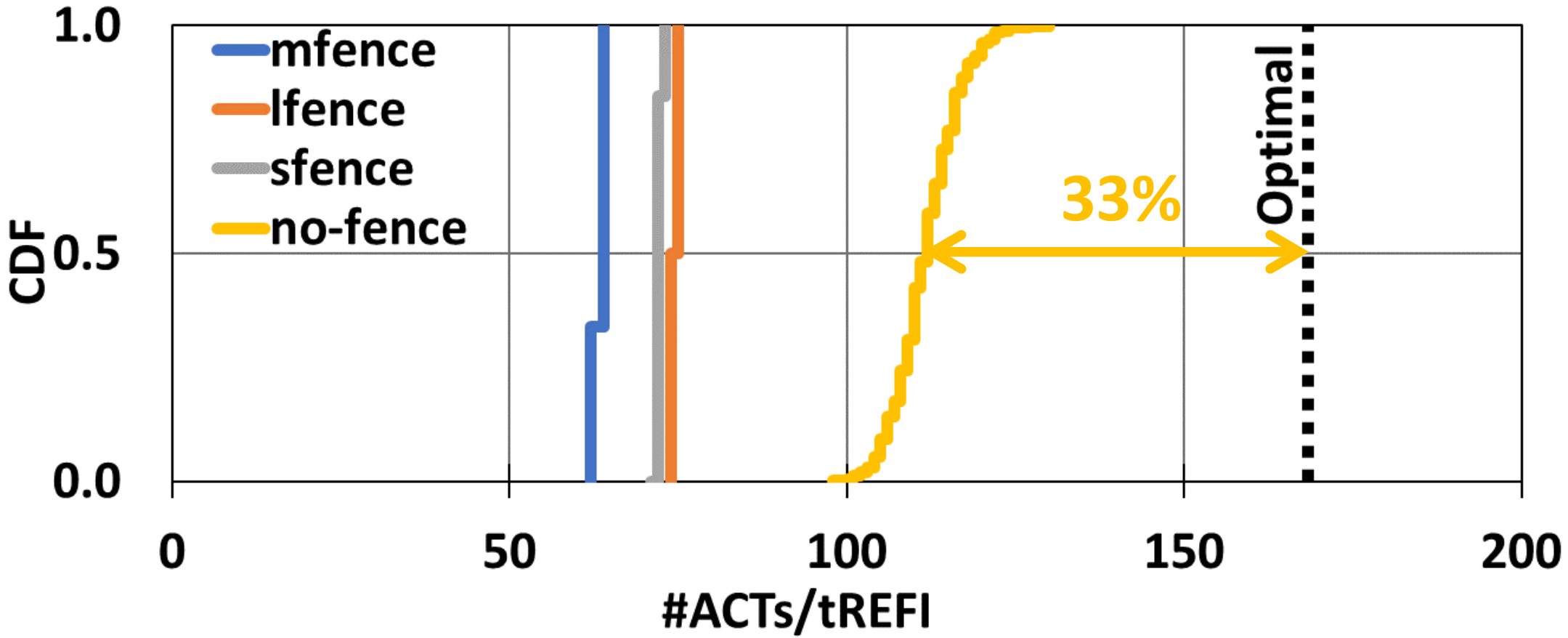
- Effectiveness of instruction sequence subject to many factors:
  - Out of order execution
  - Cache hierarchy
  - Memory controller's DRAM scheduling
  - Memory traffic interference from OS and processes

Typical Rowhammer Instruction Sequence

Dozens of variations found in dozens of papers:

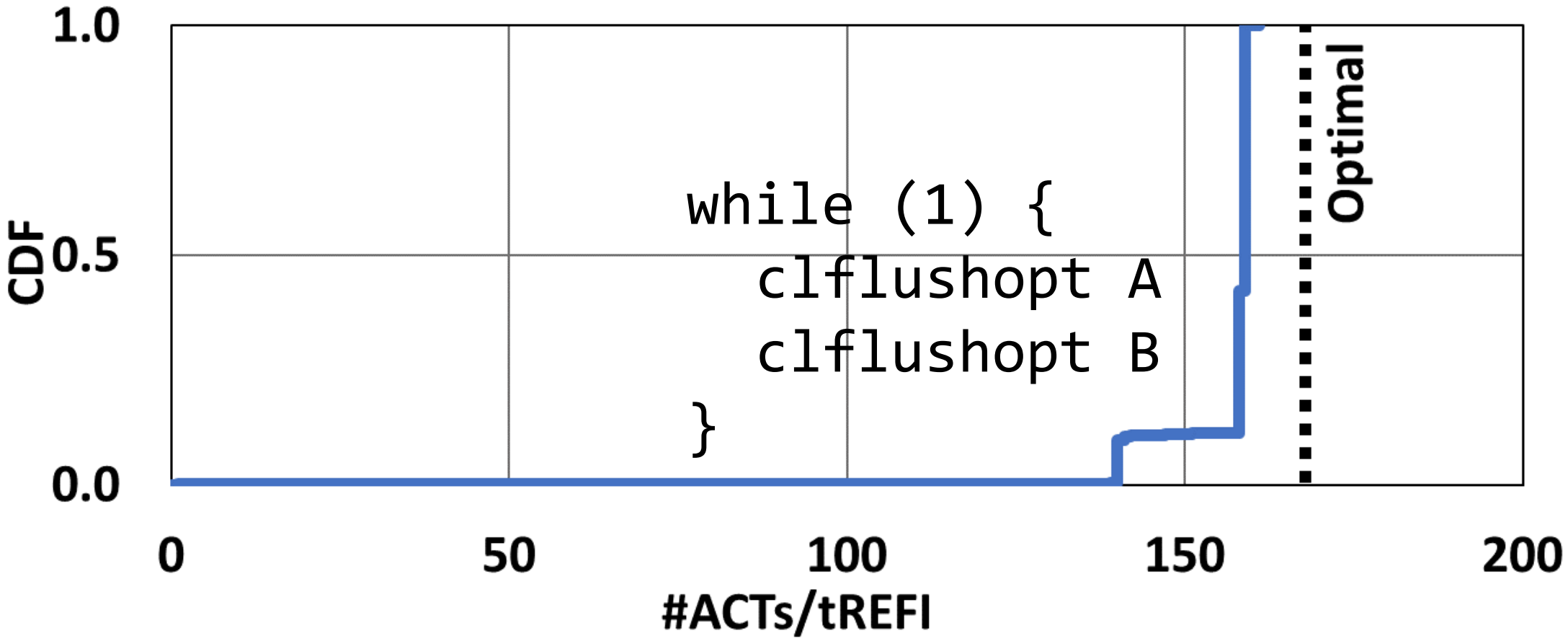
```
loop:
    movzx rax, BYTE PTR [rcx]
    movzx rax, BYTE PTR [rdx]
    clflush{opt} BYTE PTR [rcx]
    clflush{opt} BYTE PTR [rdx]
    {m|l|s|no}-fence
```

Rate of ACTs from Previous Instruction Sequences



Far from optimal

Our Near-Optimal Instruction Sequence on Skylake



Instruction sequence leverages micro-architectural side-effects

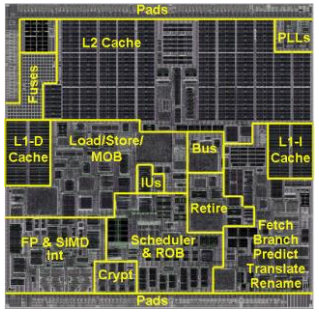
Two Key Requirements for Rowhammer Testing

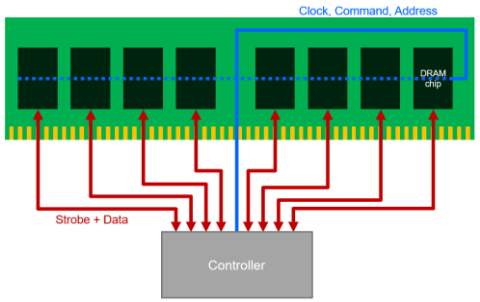
To be successful, methodology must create **worst-case** testing conditions for DIMMs

1. Must generate **highest** rate of ACTs on a cloud server

2. Must hammer rows whose cells are **adjacent** inside DRAM

Determining Physically Adjacent Rows





System Virtual Address

System Physical Address

DDR Bus Address

DRAM Internal Address

KNOWN → **KNOWN** → **REVERSE-ENGINEERED** → **UNKNOWN**

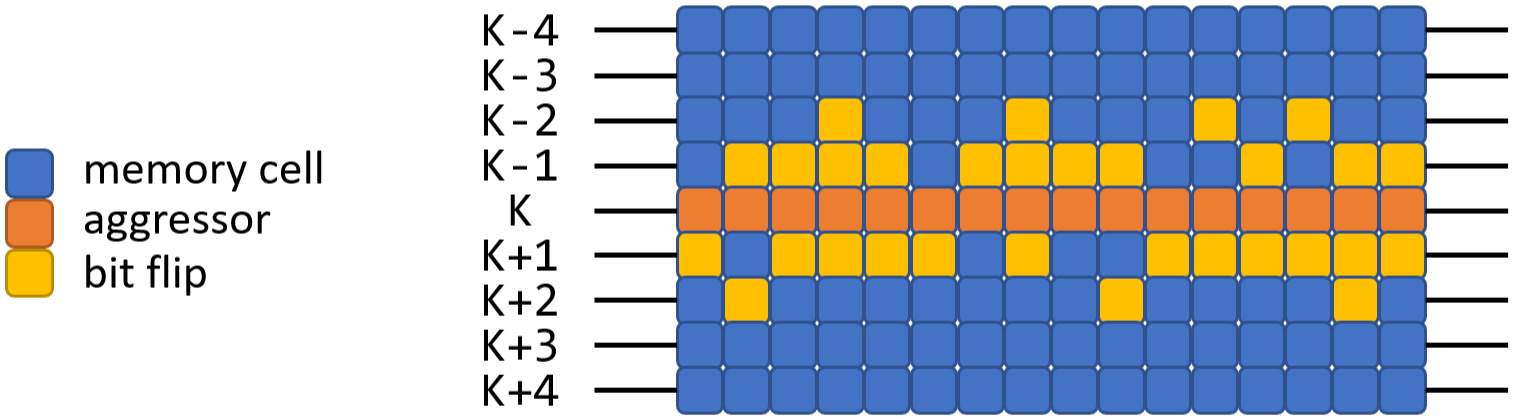
Straightforward

Chipset-specific, Underspecified documentation

DIMM-specific, Trade secret

Previous Reverse-Engineering Methodology Relies on Rowhammer

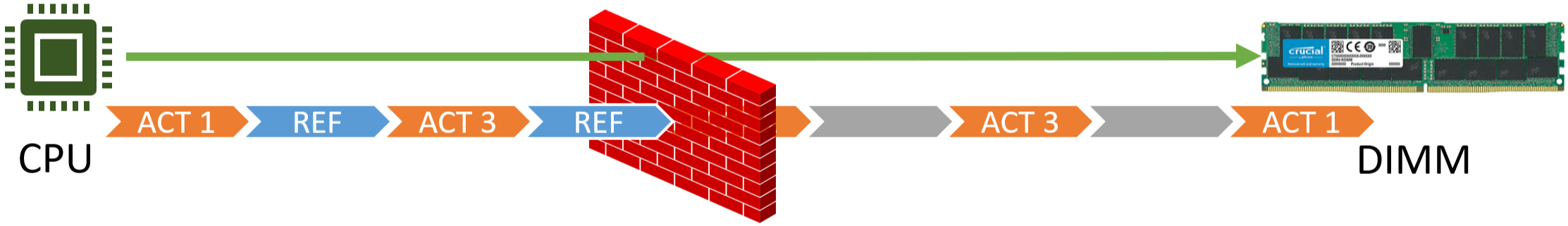
- Hammering one row yields many bit flips in adjacent rows
  - Fewer bits flipped as the distance from the hammered row increases



- Technique works only for DIMMs that succumb to Rowhammer
  - Instance of chicken-and-egg problem

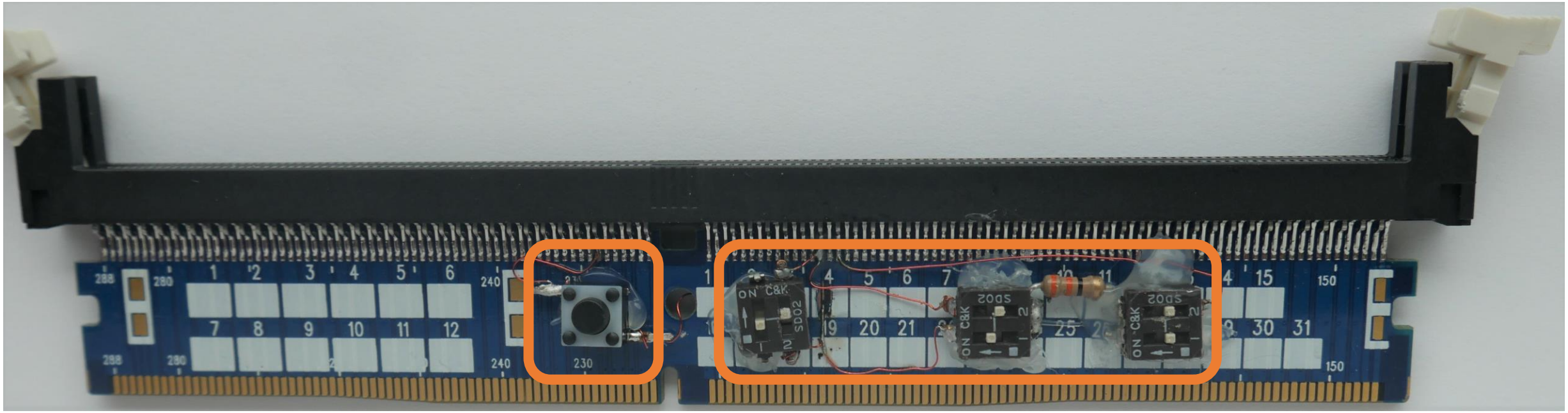
Mount a Devastating Rowhammer by Masking Refreshes (REF)

- In the absence of REFs, any DIMM easily succumbs to Rowhammer



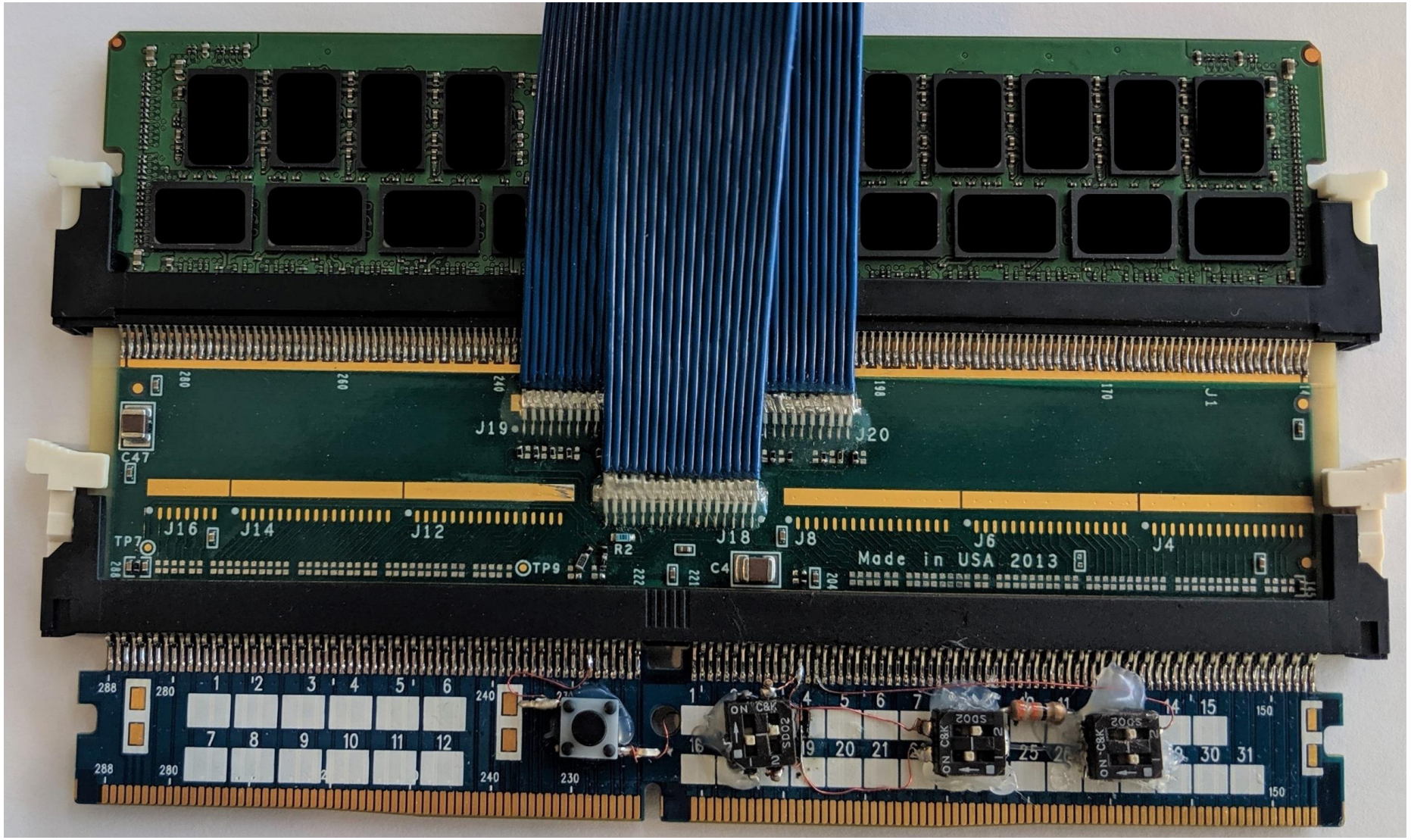
➢ Design a device to **mask REFs** but allow Rowhammer traffic

Fault Injector that Masks Refresh Commands

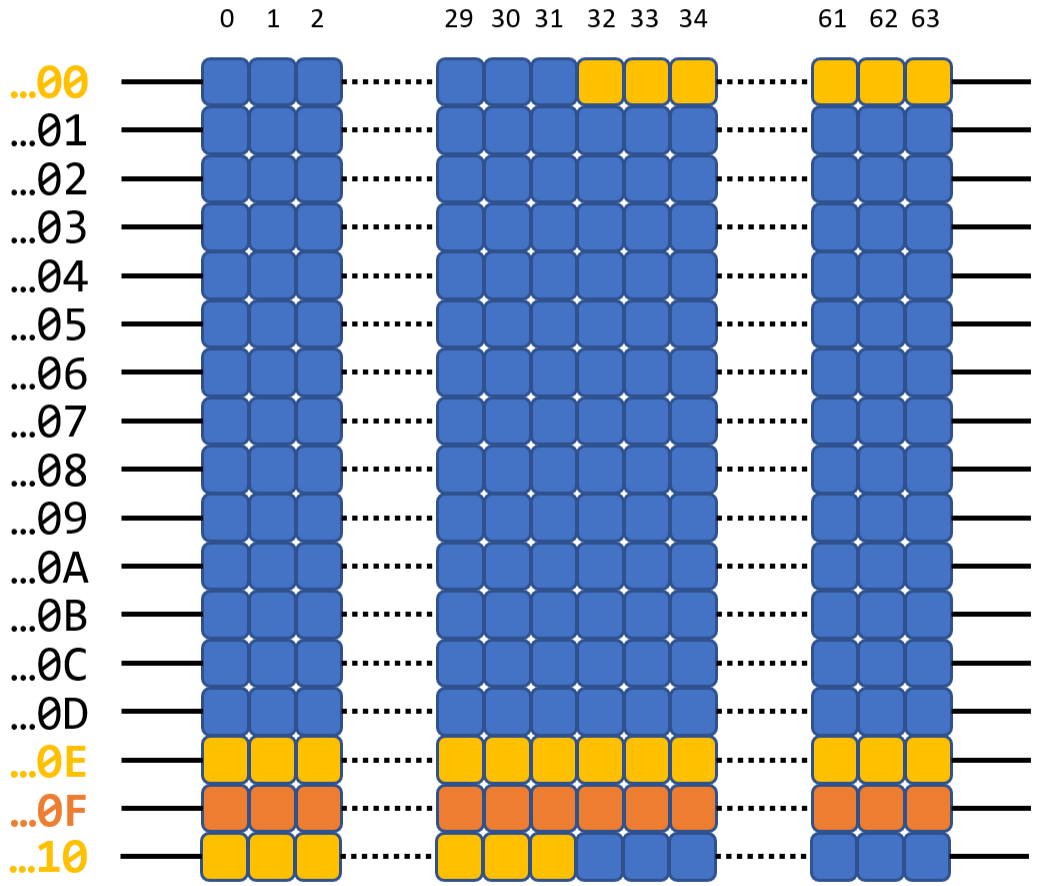


- Fault the A14 signal to mask refreshes

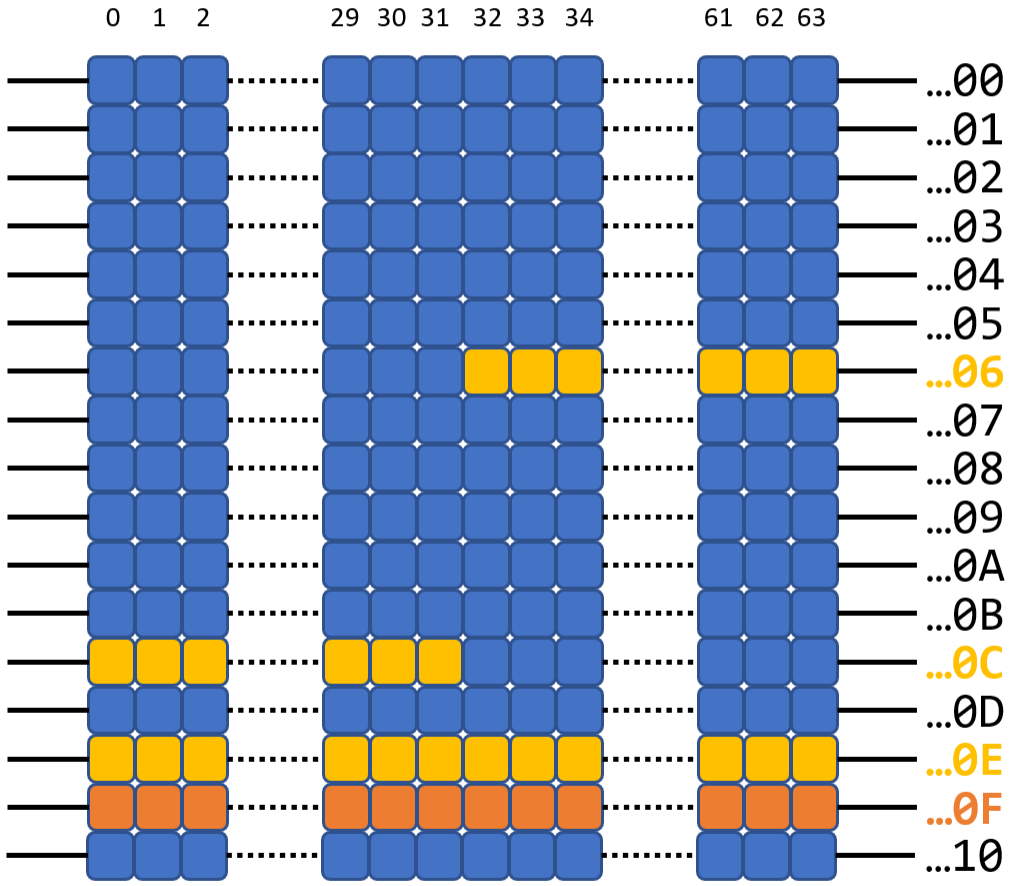
Our Complete Hardware Stack



Row Adjacency Map



Adjacency for row **0x1140F** in one DIMM from Vendor 1



Adjacency for row **0x1140F** in one DIMM from Vendor 2

Key Takeaways

- Logical rows do not always map linearly inside DRAM devices
- Words in half-rows flip fewer bits than words in whole-rows
- All bits in the whole-rows are equally susceptible
- Some (but not all) bits in half-rows are equally susceptible
- Most, but not all, bits flip from 1 to 0
- Adjacency map differs from one vendor to another
- DIMMs sourced from different vendors have different # of bit flips

An end-to-end methodology to test if any cloud server is susceptible to Rowhammer

- An instruction sequence to **optimally hammer** DRAM
  - All previous sequences have sub-optimal ACT rates
  - Two `clflushopt` instructions hammers at near-optimal rate
- A strategy to map **row adjacency** inside DIMMs
  - A DDR4 fault injector that masks refresh commands
  - Logical rows do not always map linearly
- Applied the methodology to three generations of Intel servers
  - Broadwell, Skylake, Cascade Lake