Four Key Problems + Directions

- Fundamentally **Secure/Reliable/Safe** Architectures

- Fundamentally **Energy-Efficient** Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally **Low-Latency and Predictable** Architectures

- Architectures for **AI/ML, Genomics, Medicine, Health**
The Story of RowHammer

- One can predictably induce bit flips in commodity DRAM chips
  - >80% of the tested DRAM chips are vulnerable

- First example of how a simple hardware failure mechanism can create a widespread system security vulnerability
Maslow’s (Human) Hierarchy of Needs


- We need to start with reliability and security...
How Reliable/Secure/Safe is This Bridge?

Source: http://www.technologystudent.com/struct1/tacom1.png
Collapse of the “Galloping Gertie”
How Secure Are These People?

Security is about preventing unforeseen consequences
The DRAM Scaling Problem

- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]

- DRAM capacity, cost, and energy/power hard to scale
As Memory Scales, It Becomes Unreliable

- Data from all of Facebook’s servers worldwide
- Meza+, “Revisiting Memory Errors in Large-Scale Production Data Centers,” DSN’15.

**Intuition:** quadratic increase in capacity
Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook’s server fleet

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field" Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.

[Slides (pptx) (pdf)] [DRAM Error Model]

Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

Justin Meza  Qiang Wu*  Sanjeev Kumar*  Onur Mutlu
Carnegie Mellon University   *Facebook, Inc.
Infrastructures to Understand Such Issues

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)
Infrastructures to Understand Such Issues

SoftMC: Open Source DRAM Infrastructure


- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  [GitHub Link](https://github.com/CMU-SAFARI/SoftMC)
SoftMC

- https://github.com/CMU-SAFARI/SoftMC

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan\textsuperscript{1,2,3}, Nandita Vijaykumar\textsuperscript{3}, Samira Khan\textsuperscript{4,3}, Saugata Ghose\textsuperscript{3}, Kevin Chang\textsuperscript{3}, Gennady Pekhimenko\textsuperscript{5,3}, Donghyuk Lee\textsuperscript{6,3}, Oguz Ergin\textsuperscript{2}, Onur Mutlu\textsuperscript{1,3}

\textsuperscript{1}ETH Zürich, \textsuperscript{2}TOBB University of Economics & Technology, \textsuperscript{3}Carnegie Mellon University, \textsuperscript{4}University of Virginia, \textsuperscript{5}Microsoft Research, \textsuperscript{6}NVIDIA Research
Data Retention in Memory [Liu et al., ISCA 2013]

- Retention Time Profile of DRAM looks like this:

  64-128ms

  >256ms

  128-256ms

Location dependent
Stored value pattern dependent
Time dependent
RAIDR: Heterogeneous Refresh [ISCA’12]

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh"
  Slides (pdf)

RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu    Ben Jaiyen    Richard Veras    Onur Mutlu
Carnegie Mellon University
Analysis of Data Retention Failures [ISCA’13]

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
  Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf)

An Experimental Study of Data Retention Behavior in Modern DRAM Devices:
Implications for Retention Time Profiling Mechanisms

- Jamie Liu*
  Carnegie Mellon University
  5000 Forbes Ave.
  Pittsburgh, PA 15213
  jamiel@alumni.cmu.edu

- Ben Jaiyen*
  Carnegie Mellon University
  5000 Forbes Ave.
  Pittsburgh, PA 15213
  bjaiyen@alumni.cmu.edu

- Yoongu Kim
  Carnegie Mellon University
  5000 Forbes Ave.
  Pittsburgh, PA 15213
  yoonguk@ece.cmu.edu

- Chris Wilkerson
  Intel Corporation
  2200 Mission College Blvd.
  Santa Clara, CA 95054
  chris.wilkerson@intel.com

- Onur Mutlu
  Carnegie Mellon University
  5000 Forbes Ave.
  Pittsburgh, PA 15213
  onur@cmu.edu
Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
Mitigation of Retention Issues [DSN’15]

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu,
  "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"

Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.

[Slides (pptx) (pdf)]

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi† Dae-Hyun Kim† Samira Khan‡ Prashant J. Nair† Onur Mutlu‡
†Georgia Institute of Technology
‡Carnegie Mellon University

\{moin, dhkim, pnair6\}@ece.gatech.edu
\{samirakhan, onur\}@cmu.edu
Mitigation of Retention Issues [DSN’16]

- Samira Khan, Donghyuk Lee, and Onur Mutlu,
  "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"

[Slides (pptx) (pdf)]

PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

Samira Khan* Donghyuk Lee†‡ Onur Mutlu*†
*University of Virginia †Carnegie Mellon University ‡Nvidia *ETH Zürich
Mitigation of Retention Issues [MICRO’17]

- Samira Khan, Chris Wilkerson, Zhe Wang, Alaa R. Alameldeen, Donghyuk Lee, and Onur Mutlu,
"Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content

Samira Khan* Chris Wilkerson† Zhe Wang† Alaa R. Alameldeen† Donghyuk Lee‡ Onur Mutlu*
*University of Virginia †Intel Labs ‡Nvidia Research *ETH Zürich
Mitigation of Retention Issues [ISCA’17]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
  "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"


- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER):
Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel$‡  Jeremie S. Kim‡$  Onur Mutlu$‡

§ETH Zürich  ‡Carnegie Mellon University
In-DRAM ECC Complicates Things \textsuperscript{[DSN’19]}

- Minesh Patel, Jeremie S. Kim, Hasan Hassan, and Onur Mutlu,
  "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices"


[Slides (pptx) (pdf)]
[Talk Video (26 minutes)]
[Full Talk Lecture (29 minutes)]
[Source Code for EINSim, the Error Inference Simulator]

*Best paper award.*

---

Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices

Minesh Patel\textsuperscript{†}  Jeremie S. Kim\textsuperscript{‡†}  Hasan Hassan\textsuperscript{†}  Onur Mutlu\textsuperscript{‡‡}

\textsuperscript{†}ETH Zürich  \textsuperscript{‡‡}Carnegie Mellon University
More on In-DRAM ECC [MICRO’20]

- Minesh Patel, Jeremie S. Kim, Taha Shahroodi, Hasan Hassan, and Onur Mutlu,

"Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics"


Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics

Minesh Patel† Jeremie S. Kim†† Taha Shahroodi† Hasan Hassan† Onur Mutlu††

†ETH Zürich ‡Carnegie Mellon University
A Curious Discovery [Kim et al., ISCA 2014]

One can predictably induce errors in most DRAM memory chips
A simple hardware failure mechanism can create a widespread system security vulnerability.
Modern DRAM is Prone to Disturbance Errors

Repeatedly reading a row enough times (before memory gets refreshed) induces disturbance errors in adjacent rows in most real DRAM chips you can buy today.

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)
Most DRAM Modules Are Vulnerable

A company
86%
(37/43)
Up to
1.0×10^7 errors

B company
83%
(45/54)
Up to
2.7×10^6 errors

C company
88%
(28/32)
Up to
3.3×10^5 errors

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)
Recent DRAM Is More Vulnerable
Recent DRAM Is More Vulnerable
Recent DRAM Is More Vulnerable

**All modules from 2012–2013 are vulnerable**
Why Is This 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
    - 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, charge in such cells gets drained
Higher-Level Implications

- This simple circuit level failure mechanism has enormous implications on upper layers of the transformation hierarchy.
A Simple Program Can Induce Many Errors

```
loop:
    mov (X), %eax
    mov (Y), %ebx
    clflush (X)
    clflush (Y)
    mfence
    jmp loop
```

Download from: https://github.com/CMU-SAFARI/rowhammer
A Simple Program Can Induce Many Errors

1. Avoid *cache hits*
   - Flush X from cache

2. Avoid *row hits* to X
   - Read Y in another row

Download from: [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)
A Simple Program Can Induce Many Errors

```
loop:
    mov (X), %eax
    mov (Y), %ebx
    clflush (X)
    clflush (Y)
    mfence
    jmp loop
```
A Simple Program Can Induce Many Errors

```assembly
loop:
    mov (X), %eax
    mov (Y), %ebx
    clflush (X)
    clflush (Y)
    mfence
    jmp loop
```

Download from: [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)
A Simple Program Can Induce Many Errors

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```

Download from: https://github.com/CMU-SAFARI/rowhammer
## Observed Errors in Real Systems

<table>
<thead>
<tr>
<th>CPU Architecture</th>
<th>Errors</th>
<th>Access-Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Haswell (2013)</td>
<td>22.9K</td>
<td>12.3M/sec</td>
</tr>
<tr>
<td>Intel Ivy Bridge (2012)</td>
<td>20.7K</td>
<td>11.7M/sec</td>
</tr>
<tr>
<td>Intel Sandy Bridge (2011)</td>
<td>16.1K</td>
<td>11.6M/sec</td>
</tr>
<tr>
<td>AMD Piledriver (2012)</td>
<td>59</td>
<td>6.1M/sec</td>
</tr>
</tbody>
</table>

A real reliability & security issue

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

Flipping Bits in Memory Without Accessing Them:
An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)
RowHammer Security Attack Example

- “Rowhammer” is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows (Kim et al., ISCA 2014).
  - Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

- We tested a selection of laptops and found that a subset of them exhibited the problem.
- We built two working privilege escalation exploits that use this effect.
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)

- One exploit uses rowhammer-induced bit flips to gain kernel privileges on x86-64 Linux when run as an unprivileged userland process.
- When run on a machine vulnerable to the rowhammer problem, the process was able to induce bit flips in page table entries (PTEs).
- It was able to use this to gain write access to its own page table, and hence gain read-write access to all of physical memory.
Security Implications

Rowhammer
Security Implications

Rowhammer

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after...
We present an experimental study showing that soft memory errors can lead to serious security vulnerabilities in Java and .NET virtual machines, or in any system that relies on type-checking of untrusted programs as a protection mechanism. Our attack works by sending to the JVM a Java program that is designed so that almost any memory error in its address space will allow it to take control of the JVM. All conventional Java and .NET virtual machines are vulnerable to this attack. The technique of the attack is broadly applicable against other language-based security schemes such as proof-carrying code.

We measured the attack on two commercial Java Virtual Machines: Sun’s and IBM’s. We show that a single-bit error in the Java program’s data space can be exploited to execute arbitrary code with a probability of about 70%, and multiple-bit errors with a lower probability.

Our attack is particularly relevant against smart cards or tamper-resistant computers, where the user has physical access (to the outside of the computer) and can use various means to induce faults; we have successfully used heat. Fortunately, there are some straightforward defenses against this attack.

7 Physical fault injection

If the attacker has physical access to the outside of the machine, as in the case of a smart card or other tamper-resistant computer, the attacker can induce memory errors. We considered attacks on boxes in form factors ranging from a credit card to a palmtop to a desktop PC.

We considered several ways in which the attacker could induce errors.4
Before RowHammer (II)

Using Memory Errors to Attack a Virtual Machine

Sudhakar Govindavajhala * Andrew W. Appel
Princeton University
{sdhakar,appel}@cs.princeton.edu

Figure 3. Experimental setup to induce memory errors, showing a PC built from surplus components, clip-on gooseneck lamp, 50-watt spotlight bulb, and digital thermometer. Not shown is the variable AC power supply for the lamp.

Our first detailed study: Rowhammer analysis and solutions (June 2014)
- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"

Our Source Code to Induce Errors in Modern DRAM Chips (June 2014)
- https://github.com/CMU-SAFAI/rowhammer

Google Project Zero’s Attack to Take Over a System (March 2015)
- Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)
- https://github.com/google/rowhammer-test
- Double-sided Rowhammer
Selected Readings on RowHammer (II)

- Remote RowHammer Attacks via JavaScript (July 2015)
  - https://github.com/IAIK/rowhammerjs
  - Gruss et al., DIMVA 2016.
  - CLFLUSH-free Rowhammer
    - “A fully automated attack that requires nothing but a website with JavaScript to trigger faults on remote hardware.”
    - “We can gain unrestricted access to systems of website visitors.”

- ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks (March 2016)
  - http://dl.acm.org/citation.cfm?doid=2872362.2872390
  - Aweke et al., ASPLOS 2016
  - CLFLUSH-free Rowhammer
    - Software based monitoring for rowhammer detection
Selected Readings on RowHammer (III)

- Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector (May 2016)
  - Exploits Rowhammer and Memory Deduplication to overtake a browser
  - “We report on the first reliable remote exploit for the Rowhammer vulnerability running entirely in Microsoft Edge.”
  - “[an attacker] ... can reliably “own” a system with all defenses up, even if the software is entirely free of bugs.”

- CAn’t Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory (August 2017)
  - Partitions physical memory into security domains, user vs. kernel; limits rowhammer-induced bit flips to the user domain.
Selected Readings on RowHammer (IV)

- **A New Approach for Rowhammer Attacks** (May 2016)
  - Qiao et al., HOST 2016
  - **CLFLUSH-free RowHammer**
  - “Libc functions memset and memcpy are found capable of rowhammer.”
  - Triggers RowHammer with malicious inputs but benign code

- **One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation** (August 2016)
  - “Technique that allows a malicious guest VM to have read and write accesses to arbitrary physical pages on a shared machine.”
  - Graph-based algorithm to reverse engineer mapping of physical addresses in DRAM
Selected Readings on RowHammer (V)

- Curious Case of RowHammer: Flipping Secret Exponent Bits using Timing Analysis (August 2016)
  - Bhattacharya et al., CHES 2016
  - Combines timing analysis to perform rowhammer on cryptographic keys stored in memory

- DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks (August 2016)
  - Pessl et al., USENIX Security 2016
  - Shows RowHammer failures on DDR4 devices despite TRR solution
  - Reverse engineers address mapping functions to improve existing RowHammer attacks
Selected Readings on RowHammer (VI)

- **Flip Feng Shui: Hammering a Needle in the Software Stack** (August 2016)
  - Razavi et al., USENIX Security 2016.
  - Combines memory deduplication and RowHammer
  - “A malicious VM can gain unauthorized access to a co-hosted VM running OpenSSH.”
  - Breaks OpenSSH public key authentication

- **Drammer: Deterministic Rowhammer Attacks on Mobile Platforms** (October 2016)
  - [http://dl.acm.org/citation.cfm?id=2976749.2978406](http://dl.acm.org/citation.cfm?id=2976749.2978406)
  - Van Der Veen et al., ACM CCS 2016
  - Can take over an ARM-based Android system deterministically
  - Exploits predictable physical memory allocator behavior
    - Can deterministically place security-sensitive data (e.g., page table) in an attacker-chosen, vulnerable location in memory
Selected Readings on RowHammer (VII)

- **When Good Protections go Bad: Exploiting anti-DoS Measures to Accelerate Rowhammer Attacks** (May 2017)
  - Aga et al., HOST 2017
  - “A virtual-memory based cache-flush free attack that is sufficiently fast to **rowhammer with double rate refresh.**”
  - Enabled by Cache Allocation Technology

- **SGX-Bomb: Locking Down the Processor via Rowhammer Attack** (October 2017)
  - [https://dl.acm.org/citation.cfm?id=3152709](https://dl.acm.org/citation.cfm?id=3152709)
  - Jang et al., SysTEX 2017
  - “Launches the Rowhammer attack against enclave memory to trigger the processor lockdown.”
  - **Running unknown enclave programs on the cloud can shut down servers shared with other clients.**
Selected Readings on RowHammer (VIII)

- Another Flip in the Wall of Rowhammer Defenses (May 2018)
  - Gruss et al., IEEE S&P 2018
  - A new type of Rowhammer attack which only hammers one single address, which can be done without knowledge of physical addresses and DRAM mappings
  - Defeats static analysis and performance counter analysis defenses by running inside an SGX enclave

- GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM (June 2018)
  - https://link.springer.com/chapter/10.1007/978-3-319-93411-2_5
  - Van Der Veen et al., DIMVA 2018
  - Presents RAMPAGE, a DMA-based RowHammer attack against the latest Android OS
Selected Readings on RowHammer (IX)

- Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU (May 2018)
  - The first end-to-end remote Rowhammer exploit on mobile platforms that use our GPU-based primitives in orchestration to **compromise browsers on mobile devices in under two minutes**.

- Throwhammer: Rowhammer Attacks over the Network and Defenses (July 2018)
  - Tatar et al., USENIX ATC 2018.
  - “[We] show that **an attacker can trigger and exploit Rowhammer bit flips directly from a remote machine by only sending network packets.**”
Selected Readings on RowHammer (X)

- **Nethammer: Inducing Rowhammer Faults through Network Requests** (July 2018)
  - Lipp et al., arxiv.org 2018.
  - “Nethammer is the first truly remote **Rowhammer attack**, without a single attacker-controlled line of code on the targeted system.”

- **ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks** (October 2018)
  - Konoth et al., OSDI 2018
  - A new pure-software protection mechanism against RowHammer.
Selected Readings on RowHammer (XI.A)

- PassMark Software, memtest86, since 2014
  - https://www.memtest86.com/troubleshooting.htm#hammer

Why am I only getting errors during Test 13 Hammer Test?

The Hammer Test is designed to detect RAM modules that are susceptible to disturbance errors caused by charge leakage. This phenomenon is characterized in the research paper Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors by Yoongu Kim et al. According to the research, a significant number of RAM modules manufactured 2010 or newer are affected by this defect. In simple terms, susceptible RAM modules can be subjected to disturbance errors when repeatedly accessing addresses in the same memory bank but different rows in a short period of time. Errors occur when the repeated access causes charge loss in a memory cell, before the cell contents can be refreshed at the next DRAM refresh interval.

Starting from MemTest86 v6.2, the user may see a warning indicating that the RAM may be vulnerable to high frequency row hammer bit flips. This warning appears when errors are detected during the first pass (maximum hammer rate) but no errors are detected during the second pass (lower hammer rate). See MemTest86 Test Algorithms for a description of the two passes that are performed during the Hammer Test (Test 13). When performing the second pass, address pairs are hammered only at the rate deemed as the maximum allowable by memory vendors (200K accesses per 64ms). Once this rate is exceeded, the integrity of memory contents may no longer be guaranteed. If errors are detected in both passes, errors are reported as normal.

The errors detected during Test 13, albeit exposed only in extreme memory access cases, are most certainly real errors. During typical home PC usage (e.g. web browsing, word processing, etc.), it is less likely that the memory usage pattern will fall into the extreme case that make it vulnerable to disturbance errors. It may be of greater concern if you were running highly sensitive equipment such as medical equipment, aircraft control systems, or bank database servers. It is impossible to predict with any accuracy if these errors will occur in real life applications. One would need to do a major scientific study of 1000 of computers and their usage patterns, then do a forensic analysis of each application to study how it makes use of the RAM while it executes. To date, we have only seen 1-bit errors as a result of running the Hammer Test.
Selected Readings on RowHammer (XI.B)

- PassMark Software, memtest86, since 2014
  - https://www.memtest86.com/troubleshooting.htm#hammer

Detection and mitigation of row hammer errors

The ability of MemTest86 to detect and report on row hammer errors depends on several factors and what mitigations are in place. To generate errors adjacent memory rows must be repeatedly accessed. But hardware features such as multiple channels, interleaving, scrambling, Channel Hashing, NUMA & XOR schemes make it nearly impossible (for an arbitrary CPU & RAM stick) to know which memory addresses correspond to which rows in the RAM. Various mitigations might also be in place. Different BIOS firmware might set the refresh interval to different values (tREFI). The shorter the interval the more resistant the RAM will be to errors. But shorter intervals result in higher power consumption and increased processing overhead. Some CPUs also support pseudo target row refresh (pTRR) that can be used in combination with pTRR-compliant RAM. This field allows the RAM stick to indicate the MAC (Maximum Active Count) level which is the RAM can support. A typical value might be 200,000 row activations. Some CPUs also support the Joint Electron Design Engineering Council (JEDEC) Targeted Row Refresh (TRR) algorithm. The TRR is an improved version of the previously implemented pTRR algorithm and does not inflict any performance drop or additional power usage. As a result the row hammer test implemented in MemTest86 maybe not be the worst case possible and vulnerabilities in the underlying RAM might be undetectable due to the mitigations in place in the BIOS and CPU.
Security Implications (ISCA 2014)

• **Breach of memory protection**
  – OS page (4KB) fits inside DRAM row (8KB)
  – Adjacent DRAM row $\rightarrow$ Different OS page

• **Vulnerability: disturbance attack**
  – By accessing its own page, a program could corrupt pages belonging to another program

• **We constructed a proof-of-concept**
  – Using only user-level instructions
More Security Implications (I)

“We can gain unrestricted access to systems of website visitors.”

Not there yet, but ...

ROOT privileges for web apps!

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA’16)

Source: https://lab.dsst.io/32c3-slides/7197.html
“Can gain control of a smart phone deterministically”

Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS’16
More Security Implications (III)

- Using an integrated GPU in a mobile system to remotely escalate privilege via the WebGL interface

"GRAND PWNING UNIT" — Drive-by Rowhammer attack uses GPU to compromise an Android phone

JavaScript based GLitch pwns browsers by flipping bits inside memory chips.

DAN GOODIN - 5/3/2018, 12:00 PM

Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU

Pietro Frigo
Vrije Universiteit
Amsterdam
p.frigo@vu.nl

Cristiano Giuffrida
Vrije Universiteit
Amsterdam
giuffrida@cs.vu.nl

Herbert Bos
Vrije Universiteit
Amsterdam
herbertb@cs.vu.nl

Kaveh Razavi
Vrije Universiteit
Amsterdam
kaveh@cs.vu.nl
Packets over a LAN are all it takes to trigger serious Rowhammer bit flips.

The bar for exploiting potentially serious DDR weakness keeps getting lower.

DAN GOODIN - 5/10/2018, 5:26 PM

Throwhammer: Rowhammer Attacks over the Network and Defenses

Andrei Tatar  
VU Amsterdam

Radhesh Krishnan  
VU Amsterdam

Elias Athanasopoulos  
University of Cyprus

Cristiano Giuffrida  
VU Amsterdam

Herbert Bos  
VU Amsterdam

Kaveh Razavi  
VU Amsterdam
More Security Implications (V)

- Rowhammer over RDMA (II)

**Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests**

**Nethammer:**
**Inducing Rowhammer Faults through Network Requests**

- Moritz Lipp
  Graz University of Technology

- Misiker Tadesse Aga
  University of Michigan

- Michael Schwarz
  Graz University of Technology

- Daniel Gruss
  Graz University of Technology

- Clémentine Maurice
  Univ Rennes, CNRS, IRISA

- Lukas Raab
  Graz University of Technology

- Lukas Lamster
  Graz University of Technology
More Security Implications (VI)

IEEE S&P 2020

RAMBleed

RAMBleed: Reading Bits in Memory Without Accessing Them

Andrew Kwong  
University of Michigan  
ankwong@umich.edu

Daniel Genkin  
University of Michigan  
genkin@umich.edu

Daniel Gruss  
Graz University of Technology  
daniel.gruss@iaik.tugraz.at

Yuval Yarom  
University of Adelaide and Data61  
yval@cs.adelaide.edu.au
More Security Implications (VII)

- Rowhammer on MLC NAND Flash (based on [Cai+, HPCA 2017])

Security

Rowhammer RAM attack adapted to hit flash storage

Project Zero's two-year-old dog learns a new trick

By Richard Chirgwin 17 Aug 2017 at 04:27

From random block corruption to privilege escalation:
A filesystem attack vector for rowhammer-like attacks

Anil Kurmus    Nikolas Ioannou    Matthias Neugschwandtner    Nikolaos Papandreou
Thomas Parnell
IBM Research – Zurich
More Security Implications?
Understanding RowHammer
Root Causes of Disturbance Errors

• **Cause 1: Electromagnetic coupling**
  – Toggling the wordline voltage briefly increases the voltage of adjacent wordlines
  – Slightly opens adjacent rows \(\rightarrow\) Charge leakage

• **Cause 2: Conductive bridges**

• **Cause 3: Hot-carrier injection**

*Confirmed by at least one manufacturer*
Experimental DRAM Testing Infrastructure

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)
Experimental DRAM Testing Infrastructure

### Tested DRAM Modules

(129 total)

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Module</th>
<th>Date* (yy-mm)</th>
<th>Timing†</th>
<th>Organization</th>
<th>Chip</th>
<th>Victims-per-Module</th>
<th>R_{t_{A}} (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A_1</td>
<td>10-08</td>
<td>1066</td>
<td>50.625</td>
<td>0.5</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>A_2</td>
<td>10-20</td>
<td>50.625</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_3</td>
<td>10-20</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_4</td>
<td>11-24</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_5</td>
<td>11-24</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_6</td>
<td>11-50</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_7</td>
<td>12-22</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_8</td>
<td>12-26</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_9</td>
<td>12-30</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_10</td>
<td>12-30</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_11</td>
<td>13-02</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_12</td>
<td>13-14</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_13</td>
<td>13-14</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_14</td>
<td>13-20</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_15</td>
<td>13-20</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_16</td>
<td>14-04</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>A_17</td>
<td>14-04</td>
<td>65.025</td>
<td>0.5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Total of 43 Modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| B            | B_1    | 08-49         | 1066    | 50.625       | 1     | 8                 | 16             |
| B_2          | 09-49  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_3          | 10-19  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_4          | 11-13  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_5          | 11-16  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_6          | 11-19  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_7          | 11-25  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_8          | 11-37  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_9          | 12-48  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_10         | 12-52  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_11         | 13-31  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_12         | 13-31  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_13         | 13-40  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| B_14         | 14-07  | 1066          | 50.625  | 1           | 8     | 1                 | 16             |
| Total of 54 Modules | | | | | | | |

| C            | C_1    | 10-18         | 1333    | 49.125       | 2     | 8                 | 16             |
| C_2          | 10-20  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_3          | 10-22  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_4          | 10-26  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_5          | 10-30  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_6          | 10-43  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_7          | 11-12  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_8          | 11-19  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_9          | 11-25  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_10         | 11-31  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_11         | 11-42  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_12         | 11-48  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_13         | 12-08  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_14         | 12-12  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_15         | 12-15  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_16         | 12-20  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_17         | 12-23  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_18         | 12-24  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_19         | 12-28  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_20         | 12-32  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_21         | 12-37  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_22         | 12-38  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_23         | 13-11  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| C_24         | 13-25  | 1333          | 49.125  | 2           | 8     | 1                 | 16             |
| Total of 32 Modules | | | | | | | |

* We report the manufacture date marked on the chip packages, which is more accurate than other dates that can be gleaned from a module.
† We report timing constraints stored in the module’s on-board ROM [33], which is read by the system BIOS to calibrate the memory controller.
‡ The maximum DRAM chip size supported by our testing platform is 3Gb.
§ We report DRAM die versions marked on the chip packages, which typically progress in the following manner: A → B → C → ⋯.

Table 3. Sample population of 129 DDR3 DRAM modules, categorized by manufacturer and sorted by manufacture date.
RowHammer Characterization Results

1. Most Modules Are at Risk
2. Errors vs. Vintage
3. Error = Charge Loss
4. Adjacency: Aggressor & Victim
5. Sensitivity Studies
6. Other Results in Paper
7. Solution Space

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)
4. Adjacency: Aggressor & Victim

Most aggressors & victims are adjacent

Note: For three modules with the most errors (only first bank)

Most aggressors & victims are adjacent
1. **Access Interval (Aggressor)**

![Graph showing access interval and errors](image)

**Note:** For three modules with the most errors (only first bank)

*Less frequent accesses → Fewer errors*
2 Refresh Interval

Note: Using three modules with the most errors (only first bank)

More frequent refreshes → Fewer errors
## Data Pattern

<table>
<thead>
<tr>
<th>Solid</th>
<th>RowStripe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>~Solid</td>
<td>~RowStripe</td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Errors affected by data stored in other cells
6. Other Results (in Paper)

- **Victim Cells ≠ Weak Cells (i.e., leaky cells)**
  - Almost no overlap between them

- **Errors not strongly affected by temperature**
  - Default temperature: 50°C
  - At 30°C and 70°C, number of errors changes <15%

- **Errors are repeatable**
  - Across ten iterations of testing, >70% of victim cells had errors in every iteration
6. Other Results (in Paper) cont’d

• As many as 4 errors per cache-line
  – Simple ECC (e.g., SECDED) cannot prevent all errors

• Number of cells & rows affected by aggressor
  – Victims cells per aggressor: ≤110
  – Victims rows per aggressor: ≤9

• Cells affected by two aggressors on either side
  – Very small fraction of victim cells (<100) have an error when either one of the aggressors is toggled
First RowHammer Analysis

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]
RowHammer Solutions
Two Types of RowHammer Solutions

- **Immediate**
  - To protect the vulnerable DRAM chips in the field
  - Limited possibilities

- **Longer-term**
  - To protect future DRAM chips
  - Wider range of protection mechanisms

- Our ISCA 2014 paper proposes both types of solutions
  - Seven solutions in total
  - PARA proposed as best solution → already employed in the field
Some Potential Solutions

- Make better DRAM chips
  - Cost

- Refresh frequently
  - Power, Performance

- Sophisticated ECC
  - Cost, Power

- Access counters
  - Cost, Power, Complexity
Naive Solutions

1. *Throttle accesses to same row*
   - Limit access-interval: $\geq 500\text{ns}$
   - Limit number of accesses: $\leq 128\text{K} (=64\text{ms}/500\text{ns})$

2. *Refresh more frequently*
   - Shorten refresh-interval by $\sim 7\times$

*Both naive solutions introduce significant overhead in performance and power*
Apple’s Patch for RowHammer


Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. **This issue was mitigated by increasing memory refresh rates.**

CVE-ID

CVE-2015-3693: Mark Seaborne and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP, Lenovo, and other vendors released similar patches
Our Solution to RowHammer

• **PARA:** Probabilistic Adjacent Row Activation

• Key Idea
  – After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: \( p = 0.005 \)

• Reliability Guarantee
  – When \( p = 0.005 \), errors in one year: \( 9.4 \times 10^{-14} \)
  – By adjusting the value of \( p \), we can vary the strength of protection against errors
Advantages of PARA

• *PARA refreshes rows infrequently*
  – Low power
  – Low performance-overhead
    • Average slowdown: 0.20% (for 29 benchmarks)
    • Maximum slowdown: 0.75%

• *PARA is stateless*
  – Low cost
  – Low complexity

• *PARA is an effective and low-overhead solution to prevent disturbance errors*
Requirements for PARA

• If implemented in **DRAM chip**
  – Enough slack in timing parameters
  – Plenty of slack today:
    • Chang et al., “Understanding Latency Variation in Modern DRAM Chips,” SIGMETRICS 2016.
    • Lee et al., “Design-Induced Latency Variation in Modern DRAM Chips,” SIGMETRICS 2017.
    • Chang et al., “Understanding Reduced-Voltage Operation in Modern DRAM Devices,” SIGMETRICS 2017.

• If implemented in **memory controller**
  – Better coordination between memory controller and DRAM
  – Memory controller should know which rows are physically adjacent
Probabilistic Activation in Real Life (I)

https://twitter.com/isislovecruft/status/1021939922754723841
Probabilistic Activation in Real Life (II)

https://twitter.com/isislovecruft/status/1021939922754723841
Seven RowHammer Solutions Proposed

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]
Main Memory Needs
Intelligent Controllers for Security
Industry Is Writing Papers About It, Too

**DRAM Process Scaling Challenges**

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing

- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing

- **VRT**
  - Occurring more frequently with cell capacitance decreasing
Call for Intelligent Memory Controllers

DRAM Process Scaling Challenges

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel
Aside: Intelligent Controller for NAND Flash

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642
First RowHammer Analysis

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]
Retrospective on RowHammer & Future

Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
[Slides (pptx) (pdf)]

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu
A More Recent RowHammer Retrospective

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  [Preliminary arXiv version]

RowHammer: A Retrospective

Onur Mutlu§‡
§ETH Zürich

Jeremie S. Kim‡§
‡Carnegie Mellon University
A Key Takeaway

Main Memory Needs

Intelligent Controllers
RowHammer in 2020
RowHammer in 2020 (I)

Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,

"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim§†, Minesh Patel§, A. Giray Yaglıkçısı§, Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§†

§ETH Zürich, †Carnegie Mellon University
TRRespass: Exploiting the Many Sides of Target Row Refresh

Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi,

"TRRespass: Exploiting the Many Sides of Target Row Refresh"

Best paper award.
RowHammer in 2020 (III)

- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu,
"Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers"
[Slides (pptx) (pdf)]
[Talk Video (17 minutes)]

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 Zürich, †CMU, ‡MIT
Slides for Next Lecture
Revisiting RowHammer
An Experimental Analysis of Modern Devices and Mitigation Techniques

Jeremie S. Kim  Minesh Patel
A. Giray Yağlıkçı  Hasan Hassan
Roknoddin Azizi  Lois Orosa  Onur Mutlu

SAFARI

ETH Zürich
Carnegie Mellon
Executive Summary

- **Motivation:** Denser DRAM chips are more vulnerable to RowHammer but no characterization-based study demonstrates how vulnerability scales.

- **Problem:** Unclear if existing mitigation mechanisms will remain viable for future DRAM chips that are likely to be more vulnerable to RowHammer.

- **Goal:**
  1. Experimentally demonstrate how vulnerable modern DRAM chips are to RowHammer and study how this vulnerability will scale going forward.
  2. Study viability of existing mitigation mechanisms on more vulnerable chips.

- **Experimental Study:** First rigorous RowHammer characterization study across a broad range of DRAM chips:
  - 1580 chips of different DRAM {types, technology node generations, manufacturers}.
  - We find that RowHammer vulnerability worsens in newer chips.

- **RowHammer Mitigation Mechanism Study:** How five state-of-the-art mechanisms are affected by worsening RowHammer vulnerability:
  - Reasonable performance loss (8% on average) on modern DRAM chips.
  - Scale poorly to more vulnerable DRAM chips (e.g., 80% performance loss).

- **Conclusion:** It is critical to research more effective solutions to RowHammer for future DRAM chips that will likely be even more vulnerable to RowHammer.
Motivation

- Denser DRAM chips are more vulnerable to RowHammer.

- Three prior works [Kim+, ISCA’14], [Park+, MR’16], [Park+, MR’16], over the last six years provide RowHammer characterization data on real DRAM.

- However, there is no comprehensive experimental study that demonstrates how vulnerability scales across DRAM types and technology node generations.

- It is unclear whether current mitigation mechanisms will remain viable for future DRAM chips that are likely to be more vulnerable to RowHammer.
Goal

1. Experimentally demonstrate how vulnerable modern DRAM chips are to RowHammer and predict how this vulnerability will scale going forward.

2. Examine the viability of current mitigation mechanisms on more vulnerable chips.
DRAM Testing Infrastructures

Three separate testing infrastructures

1. **DDR3**: FPGA-based SoftMC [Hassan+, HPCA’17] (Xilinx ML605)

2. **DDR4**: FPGA-based SoftMC [Hassan+, HPCA’17] (Xilinx Virtex UltraScale 95)

3. **LPDDR4**: In-house testing hardware for LPDDR4 chips

All provide fine-grained control over DRAM commands, timing parameters and temperature
### DRAM Chips Tested

<table>
<thead>
<tr>
<th>DRAM type-node</th>
<th>Number of Chips (Modules) Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-old</td>
<td>56 (10)</td>
</tr>
<tr>
<td>DDR3-new</td>
<td>80 (10)</td>
</tr>
<tr>
<td>DDR4-old</td>
<td>112 (16)</td>
</tr>
<tr>
<td>DDR4-new</td>
<td>264 (43)</td>
</tr>
<tr>
<td>LPDDR4-1x</td>
<td>12 (3)</td>
</tr>
<tr>
<td>LPDDR4-1y</td>
<td>184 (46)</td>
</tr>
</tbody>
</table>

1580 total DRAM chips tested from 300 DRAM modules

- **Three** major DRAM manufacturers {A, B, C}
- **Three** DRAM types or standards {DDR3, DDR4, LPDDR4}
  - LPDDR4 chips we test implement on-die ECC
- **Two** technology nodes per DRAM type {old/new, 1x/1y}
  - Categorized based on manufacturing date, datasheet publication date, purchase date, and characterization results

**Type-node**: configuration describing a chip’s type and technology node generation: **DDR3-old/new, DDR4-old/new, LPDDR4-1x/1y**
Effective RowHammer Characterization

To characterize our DRAM chips at **worst-case** conditions, we:

1. Prevent sources of interference during core test loop
   - We disable:
     - **DRAM refresh**: to avoid refreshing victim row
     - **DRAM calibration events**: to minimize variation in test timing
     - **RowHammer mitigation mechanisms**: to observe circuit-level effects
   - Test for **less than refresh window (32ms)** to avoid retention failures

2. **Worst-case** access sequence
   - We use **worst-case** access sequence based on prior works’ observations
   - For each row, **repeatedly access the two directly physically-adjacent rows as fast as possible**

**SAFARI** [More details in the paper]
Testing Methodology

<table>
<thead>
<tr>
<th>Row 0</th>
<th>Aggressor Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFRESH</td>
<td></td>
</tr>
<tr>
<td>Row 1</td>
<td>Victim Row</td>
</tr>
<tr>
<td>Row 2</td>
<td>Aggressor Row</td>
</tr>
<tr>
<td>Row 3</td>
<td></td>
</tr>
<tr>
<td>Row 4</td>
<td>Row</td>
</tr>
<tr>
<td>Row 5</td>
<td>Row</td>
</tr>
</tbody>
</table>

**DRAM_RowHammer_Characterization():**

```c
foreach row in DRAM:
    set victim_row to row
    set aggressor_row1 to victim_row - 1
    set aggressor_row2 to victim_row + 1
    Disable DRAM refresh
    Refresh victim_row
```

```c
for n = 1 -> HC: // core test loop
    activate aggressor_row1
    activate aggressor_row2
    Enable DRAM refresh
    Record RowHammer bit flips to storage
    Restore bit flips to original values
```

| Disable refresh to **prevent interruptions** in the core loop of our test **from refresh operations**
| Induce RowHammer bit flips on a **fully charged row** |
## Testing Methodology

### Algorithm 1: DRAM_RowHammer_Characterization()

- **foreach** row in DRAM:
  - set victim_row to row
  - set aggressor_row1 to victim_row - 1
  - set aggressor_row2 to victim_row + 1
  - Disable DRAM refresh
  - Refresh victim_row

- **for** \( n = 1 \rightarrow HC \) // core test loop
  - activate aggressor_row1
  - activate aggressor_row2
  - Enable DRAM refresh
  - Record RowHammer bit flips to storage
  - Restore bit flips to original values

### Table: Row Addressing and RowHammer Activation

<table>
<thead>
<tr>
<th>Row</th>
<th>Aggressor Row</th>
<th>Victim Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Disable refresh to **prevent interruptions** in the core loop of our test **from refresh operations**
- Induce RowHammer bit flips on a **fully charged row**
- Core test loop where we alternate accesses to adjacent rows
- **1 Hammer (HC) = two accesses**
- Prevent further retention failures
- Record bit flips for analysis
Key Takeaways from 1580 Chips

• Chips of newer DRAM technology nodes are more vulnerable to RowHammer

• There are chips today whose weakest cells fail after only 4800 hammers

• Chips of newer DRAM technology nodes can exhibit RowHammer bit flips 1) in more rows and 2) farther away from the victim row.
1. RowHammer Vulnerability

Q. Can we induce RowHammer bit flips in all of our DRAM chips?

All chips are vulnerable, except many DDR3 chips

- A total of 1320 out of all 1580 chips (84%) are vulnerable
- Within DDR3-old chips, only 12% of chips (24/204) are vulnerable
- Within DDR3-new chips, 65% of chips (148/228) are vulnerable

Newer DRAM chips are more vulnerable to RowHammer
2. Data Pattern Dependence

Q. Are some data patterns more effective in inducing RowHammer bit flips?

- We test several data patterns typically examined in prior work to identify the worst-case data pattern.

- The worst-case data pattern is consistent across chips of the same manufacturer and DRAM type-node configuration.

- We use the worst-case data pattern per DRAM chip to characterize each chip at worst-case conditions and minimize the extensive testing time.

[More detail and figures in paper]
3. Hammer Count (HC) Effects

Q. How does the Hammer Count affect the number of bit flips induced?

![Graph showing the relationship between Hammer Count (HC) and Bit Flip Rate for Mfr. A DDR4-New.]

Hammer Count = 2 Accesses, one to each adjacent row of victim
3. Hammer Count (HC) Effects

RowHammer bit flip rates increase when going from old to new DDR4 technology node generations

RowHammer bit flip rates (i.e., RowHammer vulnerability) increase with technology node generation
4. Spatial Effects: Row Distance

Q. Where do RowHammer bit flips occur relative to aggressor rows?

The number of RowHammer bit flips that occur in a given row decreases as the distance from the **victim row (row 0)** increases.
4. Spatial Effects: Row Distance

We normalize data by inducing a bit flip rate of $10^{-6}$ in each chip.
4. Spatial Effects: Row Distance

We plot this data for each DRAM type-node configuration per manufacturer.
4. Spatial Distribution of Bit Flips

Q. How are RowHammer bit flips spatially distributed across a chip?

We normalize data by inducing a bit flip rate of $10^{-6}$ in each chip. The distribution of RowHammer bit flip density per word changes significantly in LPDDR4 chips from other DRAM types.

At a bit flip rate of $10^{-6}$, a 64-bit word can contain up to 4 bit flips. Even at this very low bit flip rate, a very strong ECC is required.
4. Spatial Distribution of Bit Flips

We plot this data for each DRAM type-node configuration per manufacturer.

[More analysis in the paper]
5. First Row Hammer Bit Flips per Chip

What is the minimum Hammer Count required to cause bit flips ($HC_{first}$)?

- **Whisker Q3:** 75% point
- **Median:** 50%
- **Q1:** 25% point
- **Whisker**

![Diagram showing Hammer Count needed for the first bit flip ($HC_{first}$) for different memory types and versions.](image)
5. First RowHammer Bit Flips per Chip

What is the minimum Hammer Count required to cause bit flips ($HC_{first}$)?

We note the different DRAM types on the x-axis: DDR3, DDR4, LPDDR4.

We focus on trends across chips of the same DRAM type to draw conclusions.
5. First RowHammer Bit Flips per Chip

Newer chips from a given DRAM manufacturer more vulnerable to RowHammer
In a DRAM type, $HC_{first}$ reduces significantly from old to new chips, i.e., DDR3: 69.2k to 22.4k, DDR4: 17.5k to 10k, LPDDR4: 16.8k to 4.8k.

There are chips whose weakest cells fail after only 4800 hammers.
Key Takeaways from 1580 Chips

• Chips of newer DRAM technology nodes are more vulnerable to RowHammer

• There are chips today whose weakest cells fail after only 4800 hammers

• Chips of newer DRAM technology nodes can exhibit RowHammer bit flips 1) in more rows and 2) farther away from the victim row.
Evaluation Methodology

• **Cycle-level simulator:** Ramulator [Kim+, CAL’15]
  https://github.com/CMU-SAFARI/ramulator
  - 4GHz, 4-wide, 128 entry instruction window
  - 48 8-core workload mixes randomly drawn from SPEC CPU2006 (10 < MPKI < 740)

• **Metrics to evaluate mitigation mechanisms**
  1. **DRAM Bandwidth Overhead:** fraction of total system DRAM bandwidth consumption from mitigation mechanism
  2. **Normalized System Performance:** normalized weighted speedup to a 100% baseline
Evaluation Methodology

• We evaluate **five state-of-the-art mitigation mechanisms**: 
  - Increased Refresh Rate [Kim+, ISCA’14]
  - PARA [Kim+, ISCA’14]
  - ProHIT [Son+, DAC’17]
  - MRLoc [You+, DAC’19]
  - TWiCe [Lee+, ISCA’19]

• and **one ideal refresh-based mitigation mechanism**: 
  - Ideal

• **More detailed descriptions in the paper on:**
  - Descriptions of mechanisms in our paper and the original publications
  - How we scale each mechanism to more vulnerable DRAM chips (lower $HC_{first}$)
Mitigation Mech. Eval. (Increased Refresh)

Substantial overhead for high $HC_{\text{first}}$ values.

This mechanism does not support $HC_{\text{first}} < 32k$ due to the prohibitively high refresh rates required.
Mitigation Mechanism Evaluation (PARA)

\[ HC_{first} \text{ (number of hammers required to induce first RowHammer bit flip)} \]

Normalized System Performance (%)

- Low Performance Overhead
- High Performance Overhead

80% performance loss

Increased Refresh Rate

Mfr. A
Mfr. B
Mfr. C
Mitigation Mechanism Evaluation (ProHIT)

$HC_{\text{first}}$ (number of hammers required to induce first RowHammer bit flip)
Mitigation Mechanism Evaluation (MRLoc)

<table>
<thead>
<tr>
<th>Mitigation Mechanism</th>
<th>Evaluation</th>
<th>DRAM bandwidth overhead of RowHammer mitigation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HC first</td>
<td>Normalized</td>
<td>System Performance (%)</td>
</tr>
</tbody>
</table>

Models for scaling ProHIT and MRLoc for \( HC_{first} < 2k \) are not provided and how to do so is not intuitive.
Mitigation Mechanism Evaluation (TWiCe)

**TWiCe does not support** $HC_{\text{first}} < 32k$.

*We evaluate an ideal scalable version (TWiCe-ideal) assuming it solves two critical design issues.*
Mitigation Mechanism Evaluation (Ideal)

Normalized System Performance (%) vs. $HC_{first}$ (number of hammers required to induce first RowHammer bit flip)

**Ideal mechanism** issues a refresh command to a row only right before the row can potentially experience a RowHammer bit flip.
Mitigation Mechanism Evaluation

**PARA, ProHIT, and MRLoc** mitigate RowHammer bit flips in **worst chips** today with reasonable system performance (92%, 100%, 100%)
Mitigation Mechanism Evaluation

Only PARA’s design scales to low HC\textsubscript{first} values but has very low normalized system performance

\begin{itemize}
  \item Increased Refresh Rate
  \item PARA
  \item ProHIT
  \item MRLoc
  \item TWiCe
  \item TWiCe-ideal
\end{itemize}

\textit{HC\textsubscript{first}} (number of hammers required to induce first RowHammer bit flip)
Mitigation Mechanism Evaluation

**Ideal mechanism is significantly better than any existing mechanism for $HC_{first} < 1024$**

**Significant opportunity** for developing a RowHammer solution with low performance overhead that supports low $HC_{first}$
Key Takeaways from Mitigation Mechanisms

• Existing RowHammer mitigation mechanisms can prevent RowHammer attacks with reasonable system performance overhead in DRAM chips today

• Existing RowHammer mitigation mechanisms do not scale well to DRAM chips more vulnerable to RowHammer

• There is still significant opportunity for developing a mechanism that is scalable with low overhead
Additional Details in the Paper

- Single-cell RowHammer bit flip probability
- More details on our data pattern dependence study
- Analysis of Error Correcting Codes (ECC) in mitigating RowHammer bit flips
- Additional observations on our data
- Methodology details for characterizing DRAM
- Further discussion on comparing data across different infrastructures
- Discussion on scaling each mitigation mechanism
RowHammer Solutions Going Forward

Two promising directions for new RowHammer solutions:

1. **DRAM-system cooperation**
   - We believe the DRAM and system should cooperate more to provide a holistic solution can prevent RowHammer at low cost

2. **Profile-guided**
   - Accurate *profile of RowHammer-susceptible cells* in DRAM provides a powerful substrate for building *targeted* RowHammer solutions, e.g.:
     - Only increase the refresh rate for rows containing RowHammer-susceptible cells
   
   - A *fast and accurate* profiling mechanism is a key research challenge for developing low-overhead and scalable RowHammer solutions
Conclusion

• We characterized 1580 DRAM chips of different DRAM types, technology nodes, and manufacturers.

• We studied five state-of-the-art RowHammer mitigation mechanisms and an ideal refresh-based mechanism

• We made two key observations
  1. **RowHammer is getting much worse.** It takes much fewer hammers to induce RowHammer bit flips in newer chips
     • e.g., DDR3: 69.2k to 22.4k, DDR4: 17.5k to 10k, LPDDR4: 16.8k to 4.8k
  2. **Existing mitigation mechanisms do not scale** to DRAM chips that are more vulnerable to RowHammer
     • e.g., 80% performance loss when the hammer count to induce the first bit flip is 128

• We conclude that it is critical to do more research on RowHammer and develop scalable mitigation mechanisms to prevent RowHammer in future systems
Revisiting RowHammer
An Experimental Analysis of Modern Devices and Mitigation Techniques

Jeremie S. Kim    Minesh Patel
A. Giray Yağlıkçı    Hasan Hassan
Roknoddin Azizi    Lois Orosa    Onur Mutlu

SAFARI
Revisiting RowHammer in 2020 (I)

Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,
"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim§†, Minesh Patel§, A. Giray Yağlıkçı§
Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§†
§ETH Zürich, †Carnegie Mellon University
Future Memory
Reliability/Security Challenges
Future of Main Memory

- DRAM is becoming less reliable → more vulnerable
Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook’s server fleet

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field" Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.
[Slides (pptx) (pdf)] [DRAM Error Model]

Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

Justin Meza  Qiang Wu*  Sanjeev Kumar*  Onur Mutlu
Carnegie Mellon University  *Facebook, Inc.
DRAM Reliability Reducing

Intuition: quadratic increase in capacity

Meza+, “Revisiting Memory Errors in Large-Scale Production Data Centers,” DSN’15.
Aside: SSD Error Analysis in the Field

- First large-scale field study of flash memory errors


A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
Future of Main Memory

- DRAM is becoming less reliable → more vulnerable

- Due to difficulties in DRAM scaling, other problems may also appear (or they may be going unnoticed)

- Some errors may already be slipping into the field
  - Read disturb errors (Rowhammer)
  - **Retention errors**
  - Read errors, write errors
  - ...

- These errors can also pose security vulnerabilities
DRAM Data Retention Time Failures

- Determining the data retention time of a cell/row is getting more difficult
- Retention failures may already be slipping into the field
Analysis of Data Retention Failures [ISCA’13]

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu,
  "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
  Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf)

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time

- Variable Retention Time (VRT) phenomenon

https://www.youtube.com/watch?v=v702wUnaWGE
Industry Is Writing Papers About It, Too

**DRAM Process Scaling Challenges**

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing

- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing

- **VRT**
  - Occurring more frequently with cell capacitance decreasing
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

- Refresh
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel
Keeping Future Memory Secure
How Do We Keep Memory Secure?

- DRAM
- Flash memory
- Emerging Technologies
  - Phase Change Memory
  - STT-MRAM
  - RRAM, memristors
  - ...

SAFARI
Solution Direction: Principled Designs

Design fundamentally secure computing architectures

Predict and prevent such safety issues
Architecting Future Memory for Security

- **Understand:** Methods for vulnerability modeling & discovery
  - Modeling and prediction based on real (device) data and analysis
  - Understanding vulnerabilities
  - Developing reliable metrics

- **Architect:** Principled architectures with security as key concern
  - Good partitioning of duties across the stack
  - Cannot give up performance and efficiency
  - Patch-ability in the field

- **Design & Test:** Principled design, automation, (online) testing
  - Design for security
  - High coverage and good interaction with system reliability methods
Understand and Model with Experiments (DRAM)

SAFARI

Understand and Model with Experiments (Flash)


Understanding Flash Memory Reliability

Proceedings of the IEEE, Sept. 2017

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

SAFARI

https://arxiv.org/pdf/1706.08642
Understanding Flash Memory Reliability


[Slides (pptx) (pdf)] [Coverage at ZDNet] [Coverage on The Register] [Coverage on TechSpot] [Coverage on The Tech Report]

A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onor@cmu.edu
Vulnerabilities in MLC NAND Flash Memory Programming:
Experimental Analysis, Exploits, and Mitigation Techniques

Yu Cai† Saugata Ghose† Yixin Luo‡† Ken Mai† Onur Mutlu§† Erich F. Haratsch‡
†Carnegie Mellon University ‡Seagate Technology §ETH Zürich

Modern NAND flash memory chips provide high density by storing two bits of data in each flash cell, called a multi-level cell (MLC). An MLC partitions the threshold voltage range of a flash cell into four voltage states. When a flash cell is programmed, a high voltage is applied to the cell. Due to parasitic capacitance coupling between flash cells that are physically close to each other, flash cell programming can lead to cell-to-cell program interference, which introduces errors into neighboring flash cells. In order to reduce the impact of cell-to-cell interference on the reliability of MLC NAND flash memory, flash manufacturers adopt a two-step programming method, which programs the MLC in two separate steps. First, the flash memory partially programs the least significant bit of the MLC to some intermediate threshold voltage. Second, it programs the most significant bit to bring the MLC up to its full voltage state.

In this paper, we demonstrate that two-step programming exposes new reliability and security vulnerabilities. We experi-
HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness

Yixin Luo†, Saugata Ghose†, Yu Cai‡, Erich F. Haratsch‡, and Onur Mutlu§

†Carnegie Mellon University  ‡Seagate Technology  §ETH Zürich
Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation

Yixin Luo†, Saugata Ghose†, Yu Cai‡, Erich F. Haratsch‡, and Onur Mutlu§†
†Carnegie Mellon University ‡Seagate Technology §ETH Zürich
Recall: Collapse of the “Galloping Gertie”
Another Example (1994)

Source: By 최광모 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35197984
Yet Another Example (2007)

A More Recent Example (2018)

The Takeaway, Again

In-Field Patch-ability (Intelligent Memory) Can Avoid Such Failures
Final Thoughts on RowHammer
Aside: Byzantine Failures

- This class of failures is known as Byzantine failures

- Characterized by
  - Undetected erroneous computation
  - Opposite of “fail fast (with an error or no result)”

- “erroneous” can be “malicious” (intent is the only distinction)

- Very difficult to detect and confine Byzantine failures

- Do all you can to avoid them

Aside: Byzantine Generals Problem

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Categories and Subject Descriptors: C.2.4. [Computer-Communication Networks]: Distributed Systems—network operating systems; D.4.4 [Operating Systems]: Communications Management—network communication; D.4.5 [Operating Systems]: Reliability—fault tolerance

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

https://dl.acm.org/citation.cfm?id=357176
RowHammer, Revisited

- One can predictably induce bit flips in commodity DRAM chips
  - >80% of the tested DRAM chips are vulnerable

- First example of how a simple hardware failure mechanism can create a widespread system security vulnerability

Forget Software—Now Hackers Are Exploiting Physics
RowHammer: Retrospective

- New mindset that has enabled a renewed interest in HW security attack research:
  - Real (memory) chips are vulnerable, in a simple and widespread manner → this causes real security problems
  - Hardware reliability → security connection is now mainstream discourse

- Many new RowHammer attacks...
  - Tens of papers in top security venues
  - More to come as RowHammer is getting worse (DDR4 & beyond)

- Many new RowHammer solutions...
  - Apple security release; Memtest86 updated
  - Many solution proposals in top venues (latest in ISCA 2019)
  - Principled system-DRAM co-design (in original RowHammer paper)
  - More to come...
Perhaps Most Importantly…

- RowHammer enabled a shift of mindset in mainstream security researchers
  - General-purpose hardware is fallible, in a widespread manner
  - Its problems are exploitable

- This mindset has enabled many systems security researchers to examine hardware in more depth
  - And understand HW’s inner workings and vulnerabilities

- It is no coincidence that two of the groups that discovered Meltdown and Spectre heavily worked on RowHammer attacks before
  - More to come…
Summary: RowHammer

- DRAM reliability is reducing
- Reliability issues open up security vulnerabilities
  - Very hard to defend against
- **Rowhammer is a prime example**
  - First example of how a simple hardware failure mechanism can create a widespread system security vulnerability
  - Its implications on system security research are tremendous & exciting
- **Bad news**: RowHammer is getting worse.

- **Good news**: We have a lot more to do.
  - We are now fully aware hardware is easily fallible.
  - We are developing both attacks and solutions.
  - We are developing principled models, methodologies, solutions.
For More on RowHammer...

- Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"
  [Preliminary arXiv version]

RowHammer: A Retrospective

Onur Mutlu§‡  Jeremie S. Kim‡§
§ETH Zürich  ‡Carnegie Mellon University
RowHammer in 2020 (I)

- Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,

"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim§†, Minesh Patel§, A. Giray Yağlıkçı§, Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§†

§ETH Zürich, †Carnegie Mellon University
RowHammer in 2020 (II)

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi, "TRRespass: Exploiting the Many Sides of Target Row Refresh"


[Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Source Code]
[Web Article]

*Best paper award.*

TRRespass: Exploiting the Many Sides of Target Row Refresh
RowHammer in 2020 (III)

- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu,

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


[Slides (pptx) (pdf)]
[Talk Video (17 minutes)]

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 Zürich, †CMU, ‡MIT
Rowhammer
Some History
Initial RowHammer Reviews

Disturbance Errors in DRAM: Demonstration, Characterization, and Prevention

Rejected (R2) 863kB  Friday 31 May 2013 2:00:53pm PDT

You are an author of this paper.

+ Abstract

<table>
<thead>
<tr>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevExp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review #66A</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Review #66B</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Review #66C</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Review #66D</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Review #66E</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Review #66F</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Missing the Point  Reviews from Micro 2013

**Paper Weaknesses**

This is an excellent test methodology paper, but there is no micro-architectural or architectural content.

- Whereas they show disturbance may happen in DRAM array, authors don't show it can be an issue in realistic DRAM usage scenario.
- Lacks architectural/microarchitectural impact on the DRAM disturbance analysis.

**Paper Weaknesses**

The mechanism investigated by the authors is one of many well known disturb mechanisms. The paper does not discuss the root causes to sufficient depth and the importance of this mechanism compared to others. Overall the length of the sections restating known information is much too long in relation to new work.
More …

Reviews from ISCA 2014

Paper Weaknesses

1) The disturbance error (a.k.a. coupling or cross-talk noise induced error) is a known problem to the DRAM circuit community.

2) What you demonstrated in this paper is so called DRAM row hammering issue - you can even find a Youtube video showing this! - http://www.youtube.com/watch?v=i3-gQSnBcdo

2) The architectural contribution of this study is too insignificant.

Paper Weaknesses

- Row Hammering appears to be well-known, and solutions have already been proposed by industry to address the issue.

- The paper only provides a qualitative analysis of solutions to the problem. A more robust evaluation is really needed to know whether the proposed solution is necessary.
Final RowHammer Reviews

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

Accepted 639kB 21 Nov 2013 10:53:11pm CST |
f039be2735313b39304ae1c6296523867a485610

You are an author of this paper.

<table>
<thead>
<tr>
<th>Review #41A</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review #41B</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review #41C</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review #41D</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review #41E</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review #41F</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Suggestions to Reviewers

- Be fair; you do not know it all
- Be open-minded; you do not know it all
- Be accepting of diverse research methods: there is no single way of doing research
- Be constructive, not destructive
- Do not have double standards...

Do not block or delay scientific progress for non-reasons
An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
  - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15
More Thoughts and Suggestions

- Onur Mutlu,
  "Some Reflections (on DRAM)"
  Award Speech for *ACM SIGARCH Maurice Wilkes Award*, at the *ISCA* Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  *57th Design Automation Conference Early Career Workshop (DAC)*, Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]
Aside: A Recommended Book

Even if the performance analysis is correctly done and presented, it may not be enough to persuade your audience—the decision makers—to follow your recommendations. The list shown in Box 10.2 is a compilation of reasons for rejection heard at various performance analysis presentations. You can use the list by presenting it immediately and pointing out that the reason for rejection is not new and that the analysis deserves more consideration. Also, the list is helpful in getting the competing proposals rejected!

There is no clear end of an analysis. Any analysis can be rejected simply on the grounds that it requires more analysis. This is the first reason listed in Box 10.2. The second most common reason for rejection of an analysis and for endless debate is the workload. Since workloads are always based on the past measurements, their applicability to the current or future environment can always be questioned. Actually workload is one of the four areas of discussion that lead a performance presentation into an endless debate. These “rat holes” and their relative sizes in terms of time consumed are shown in Figure 10.26. Presenting this cartoon at the beginning of a presentation helps to avoid these areas.

Box 10.2 Reasons for Not Accepting the Results of an Analysis

1. This needs more analysis.
2. You need a better understanding of the workload.
3. It improves performance only for long I/O's, packets, jobs, and files, and most of the I/O's, packets, jobs, and files are short.
4. It improves performance only for short I/O's, packets, jobs, and files, but who cares for the performance of short I/O's, packets, jobs, and files; it’s the long ones that impact the system.
5. It needs too much memory/CPU/bandwidth and memory/CPU/bandwidth isn’t free.
6. It only saves us memory/CPU/bandwidth and memory/CPU/bandwidth is cheap.
7. There is no point in making the networks (similarly, CPUs/disks/—) faster; our CPUs/disks (any component other than the one being discussed) aren’t fast enough to use them.
8. It improves the performance by a factor of \( x \), but it doesn’t really matter at the user level because everything else is so slow.
9. It is going to increase the complexity and cost.
10. Let us keep it simple stupid (and your idea is not stupid).
11. It is not simple. (Simplicity is in the eyes of the beholder.)
12. It requires too much state.
13. Nobody has ever done that before. (You have a new idea.)
14. It is not going to raise the price of our stock by even an eighth. (Nothing ever does, except rumors.)
15. This will violate the IEEE, ANSI, CCITT, or ISO standard.
16. It may violate some future standard.
17. The standard says nothing about this and so it must not be important.
18. Our competitors don’t do it. If it was a good idea, they would have done it.
19. Our competition does it this way and you don’t make money by copying others.
20. It will introduce randomness into the system and make debugging difficult.
21. It is too deterministic; it may lead the system into a cycle.
22. It’s not interoperable.
23. This impacts hardware.
24. That’s beyond today’s technology.
25. It is not self-stabilizing.
26. Why change—it’s working OK.
A similar process of professionalization has transformed other parts of the scientific landscape. (Central Press/Getty Images)

THE WALL STREET JOURNAL.

**Could Einstein get published today?**

3 min read. Updated: 25 Sep 2020, 11:51 AM IST

The Wall Street Journal

Scientific journals and institutions have become more professionalized over the last century, leaving less room for individual style