Seminar in Computer Architecture
The Story of RowHammer

Prof. Onur Mutlu

ETH Zürich
Fall 2022
27 October 2022
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
How Safe & Secure Are Our Platforms?

Security is about preventing unforeseen consequences

Source: https://taxistartup.com/wp-content/uploads/2015/03/UK-Self-Driving-Cars.jpg
What Is 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**

**WIRED**

**FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS**
An “Early” Position Paper [IMW’13]

- Onur Mutlu, "Memory Scaling: A Systems Architecture Perspective"
  Proceedings of the 5th International Memory Workshop (IMW), Monterey, CA, May 2013. Slides (pptx) (pdf)
  EETimes Reprint

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

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.

![Graph showing relative server failure rate vs. chip density (Gb)]

**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]
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.com/CMU-SAFARI/SoftMC
SoftMC: Open Source DRAM Infrastructure

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

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

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

$^1$ETH Zürich  $^2$TOBB University of Economics & Technology  $^3$Carnegie Mellon University
$^4$University of Virginia  $^5$Microsoft Research  $^6$NVIDIA Research
Retention Time Profile of DRAM looks like this:

- Location dependent
- Stored value pattern dependent
- Time dependent

64-128ms

>256ms

128-256ms
RAIDR: Heterogeneous Refresh [ISCA’12]

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
Mitigation of Retention Issues [SIGMETRICS’14]

- 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"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, June 2016. [Slides (pptx) (pdf)]
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)]
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"
  *Proceedings of the 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, June 2017.*
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

- 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
Mitigation of Retention Issues [DSN'19]

[Source Code for EINSim, the Error Inference Simulator] Best paper award.
Mitigation of Retention Issues [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"


[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (15 minutes)]
[Lightning Talk Video (1.5 minutes)]

*Best paper award.*

---

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
Mitigation of Retention Issues [MICRO’21]

- Minesh Patel, Geraldo F. de Oliveira Jr., and Onur Mutlu, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes"

Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (1.5 minutes)]
[HARP Source Code (Officially Artifact Evaluated with All Badges)]

HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes
More on DRAM Refresh & Data Retention

Variable Retention Time

A 2Gb chip family

https://www.youtube.com/watch?v=v702wUnaWGE&list=PL5Q2soXY2Zi9xidyIqBxUz7xRPS-wisBN&index=3
SoftMC: Enabling DRAM Infrastructure


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

github.com/CMU-SAFARI/SoftMC
A Curious Phenomenon
One can predictably induce errors in most DRAM memory chips
DRAM RowHammer

A simple hardware failure mechanism can create a widespread system security vulnerability

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS
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

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 RowHammer 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 \( \mathbf{X} \) from cache

2. Avoid *row hits* to \( \mathbf{X} \)
   - Read \( \mathbf{Y} \) in another row

Download from: 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
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

```
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

One Can Take Over an Otherwise-Secure System

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

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology...

Project Zero

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

News and updates from the Project Zero team at Google

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges
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.

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn & Dullien, 2015)
Security Implications

Rowhammer
Security Implications

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after.
"We can gain unrestricted access to systems of website visitors."

Not there yet, but ...

**ROWHAMMER.js**

ROOT privileges for web apps!

Source: [https://lab.dsst.io/32c3-slides/7197.html](https://lab.dsst.io/32c3-slides/7197.html)
More Security Implications (II)

“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. IEEE S&P 2018

"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: 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)

USENIX Security 2019

Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks

Sanghyun Hong, Pietro Frigo†, Yiğitcan Kaya, Cristiano Giuffrida†, Tudor Dumitraș

University of Maryland, College Park
†Vrije Universiteit Amsterdam

A Single Bit-flip Can Cause Terminal Brain Damage to DNNs
One specific bit-flip in a DNN’s representation leads to accuracy drop over 90%

Our research found that a specific bit-flip in a DNN’s bitwise representation can cause the accuracy loss up to 90%, and the DNN has 40-50% parameters, on average, that can lead to the accuracy drop over 10% when individually subjected to such single bitwise corruptions...

Read More
More Security Implications (VIII)

USENIX Security 2020

DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao  
University of Central Florida  
fan.yao@ucf.edu

Adnan Siraj Rakin  
Arizona State University  
asrakin@asu.edu

Deliang Fan  

Degrade the inference accuracy to the level of Random Guess

Example: ResNet-20 for CIFAR-10, 10 output classes
Before attack, Accuracy: 90.2%  After attack, Accuracy: ~10% (1/10)
More Security Implications (IX)

- 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?
A RowHammer Survey Across the Stack

- Onur Mutlu and Jeremie Kim, *RowHammer: A Retrospective*
  

  [Preliminary arXiv version]

  [Slides from COSADE 2019 (pptx)]

  [Slides from VLSI-SOC 2020 (pptx) (pdf)]

  [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu§‡  
§ETH Zürich

Jeremie S. Kim‡§  
‡Carnegie Mellon University
Understanding RowHammer
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] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).

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

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹ Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs
RowHammer Infrastructure (2012-2014)

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Module</th>
<th>Date* (yy-mm)</th>
<th>Timing¹</th>
<th>Organization</th>
<th>Chip</th>
<th>Victims-per-Module</th>
<th>R_{30} (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>10-08</td>
<td>1066</td>
<td>50.625</td>
<td>0.5</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A2</td>
<td>10-20</td>
<td>1066</td>
<td>50.625</td>
<td>0.5</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>10-20</td>
<td>1066</td>
<td>50.625</td>
<td>0.5</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A7</td>
<td>11-24</td>
<td>1066</td>
<td>49.125</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A12</td>
<td>11-12</td>
<td>1066</td>
<td>49.125</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A14</td>
<td>11-50</td>
<td>1066</td>
<td>49.125</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>12-22</td>
<td>1600</td>
<td>49.125</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>A17</td>
<td>12-26</td>
<td>1600</td>
<td>49.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B1</td>
<td>08-49</td>
<td>1066</td>
<td>50.625</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>09-49</td>
<td>1066</td>
<td>50.625</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>B3</td>
<td>10-19</td>
<td>1066</td>
<td>50.625</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>B4</td>
<td>11-13</td>
<td>1333</td>
<td>49.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B5</td>
<td>11-16</td>
<td>1066</td>
<td>50.625</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>B6</td>
<td>11-19</td>
<td>1066</td>
<td>50.625</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>B7</td>
<td>11-25</td>
<td>1333</td>
<td>49.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B8</td>
<td>11-37</td>
<td>1333</td>
<td>49.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B9</td>
<td>13-14</td>
<td>1333</td>
<td>49.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B10</td>
<td>12-01</td>
<td>1866</td>
<td>47.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B11</td>
<td>12-10</td>
<td>1866</td>
<td>47.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B12</td>
<td>12-25</td>
<td>1600</td>
<td>48.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B13</td>
<td>14-07</td>
<td>1600</td>
<td>48.125</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>

Total of 43 Modules

| C1          | 08-18  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C2          | 08-20  | 1066         | 50.625  | 2            | 8     | 16                | 0          |
| C3          | 08-22  | 1066         | 50.625  | 2            | 8     | 16                | 0          |
| C4          | 10-26  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C5          | 10-43  | 1333         | 49.125  | 1            | 8     | 16                | 0          |
| C6          | 11-12  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C7          | 11-19  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C8          | 11-31  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C9          | 11-42  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C10         | 11-48  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C11         | 12-08  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C12         | 12-12  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C13         | 12-20  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C14         | 12-24  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C15         | 12-27  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C16         | 12-32  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C17         | 12-37  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C18         | 13-11  | 1333         | 49.125  | 2            | 8     | 16                | 0          |
| C19         | 13-15  | 1333         | 49.125  | 2            | 8     | 16                | 0          |

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 32GB.
§ We report DRAM die versions marked on the chip packages, which typically progress in the following manner: A → 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)

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

Less frequent accesses $\rightarrow$ Fewer errors
Note: Using three modules with the most errors (only first bank)

More frequent refreshes → Fewer errors
3 Data Pattern

Errors affected by data stored in other cells
6. Other Key Observations [ISCA’14]

• **Victim Cells ≠ Retention-Weak Cells**
  – Almost no overlap between them

• **Errors are repeatable**
  – Across ten iterations of testing, >70% of victim cells had errors in every iteration

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

• **Cells affected by two aggressors on either side**
  – Double sided hammering
Major RowHammer Characteristics (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"
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data] [Lecture Video (1 hr 49 mins), 25 September 2020]
One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).
RowHammer is Getting Much Worse (2020)

- 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
Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
"A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (21 minutes)]
[Lightning Talk Video (1.5 minutes)]
[arXiv version]
Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices

A. Giray Yağlıkçı¹, Haocong Luo¹, Geraldo F. de Oliveira¹, Ataberk Olgun¹, Minesh Patel¹, Jisung Park¹, Hasan Hassan¹, Jeremie S. Kim¹, Lois Orosa¹,² and Onur Mutlu¹

¹ETH Zürich  ²Galicia Supercomputing Center (CESGA)
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 (ISCA 2014)

- Make better DRAM chips  
  - Cost

- Refresh frequently  
  - Power, Performance

- Sophisticated ECC  
  - Cost, Power

- Access counters  
  - Cost, Power, Complexity
Apple’s Security Patch for RowHammer

- https://support.apple.com/en-gb/HT204934

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 Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP, Lenovo, and many 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** (done today)
  – Enough slack in timing and refresh 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**
  – Need coordination between 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] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).

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

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹ Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs

SAFARI
A Takeaway

Main Memory Needs

Intelligent Controllers

for Security, Safety, Reliability, Scaling
Aside: Intelligent Controller for NAND Flash

HAPS-52 Mother Board

USB Daughter Board

Virtex-V FPGA (NAND Controller)

USB Jack

Virtex-I Pro (USB controller)

1x-nm NAND Flash

NAND Daughter Board


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
Detailed Lectures on RowHammer

- Computer Architecture, Fall 2021, Lecture 5
  - RowHammer (ETH Zürich, Fall 2021)
  - https://www.youtube.com/watch?v=7wVKnPj3NVw&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=5

- Computer Architecture, Fall 2021, Lecture 6
  - RowHammer and Secure & Reliable Memory (ETH Zürich, Fall 2021)
  - https://www.youtube.com/watch?v=HNd4skQrt6I&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=6

https://www.youtube.com/onurmutlulectures
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] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).

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

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹ Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs
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

  - [Preliminary arXiv version]
  - [Slides from COSADE 2019 (pptx)]
  - [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  - [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu$^{§‡}$  Jeremie S. Kim$^{‡§}$

$^{§}$ETH Zürich  $^{‡}$Carnegie Mellon University
RowHammer in 2020-2022
Revisiting RowHammer
RowHammer is Getting Much Worse

- 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
Key Takeaways from 1580 Chips

• Newer DRAM chips are much more vulnerable to RowHammer (more bit flips, happening earlier)

• There are new chips 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.

• Existing mitigation mechanisms are NOT effective at future technology nodes
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
## 1580 DRAM Chips Tested

<table>
<thead>
<tr>
<th>DRAM type-node</th>
<th>Number of Chips (Modules) Tested</th>
<th>Mfr. A</th>
<th>Mfr. B</th>
<th>Mfr. C</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-old</td>
<td>56 (10)</td>
<td>88 (11)</td>
<td>28 (7)</td>
<td></td>
<td>172 (28)</td>
</tr>
<tr>
<td>DDR3-new</td>
<td>80 (10)</td>
<td>52 (9)</td>
<td>104 (13)</td>
<td></td>
<td>236 (32)</td>
</tr>
<tr>
<td>DDR4-old</td>
<td>112 (16)</td>
<td>24 (3)</td>
<td>128 (18)</td>
<td></td>
<td>264 (37)</td>
</tr>
<tr>
<td>DDR4-new</td>
<td>264 (43)</td>
<td>16 (2)</td>
<td>108 (28)</td>
<td></td>
<td>388 (73)</td>
</tr>
<tr>
<td>LPDDR4-1x</td>
<td>12 (3)</td>
<td>180 (45)</td>
<td>N/A</td>
<td></td>
<td>192 (48)</td>
</tr>
<tr>
<td>LPDDR4-1y</td>
<td>184 (46)</td>
<td>N/A</td>
<td>144 (36)</td>
<td></td>
<td>328 (82)</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

*Safari*
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**.
Newer chips from each DRAM manufacturer are more vulnerable to RowHammer
5. First RowHammer Bit Flips per Chip

In a DRAM type, $HC_{\text{first}}$ 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

Newer chips from a given DRAM manufacturer more vulnerable to RowHammer
Mitigation Mechanism Evaluation

Ideal mechanism is significantly better than any existing mechanism for $H_{C_{\text{first}}} < 1024$

Significant opportunity for developing a RowHammer solution with low performance overhead that supports low $H_{C_{\text{first}}}$
RowHammer is Getting Much Worse

- 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)]
Computer Architecture, Fall 2020, Lecture 5b

- RowHammer in 2020: Revisiting RowHammer (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=gR7XR-Eepcg&list=PL5Q2soXY2Zi9idyIgBxUz7xRPS-wisBN&index=10

https://www.youtube.com/onurmutlulectures
TRResplass
Industry-Adopted Solutions Do Not Work

- 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)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Lecture Video (59 minutes)]
[Source Code]
[Web Article]

Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

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* Kaveh Razavi*

*Vrije Universiteit Amsterdam  †ETH Zürich  ‥Qualcomm Technologies Inc.
TRRespass

- First work to show that TRR-protected DRAM chips are vulnerable to RowHammer in the field
  - Mitigations advertised as secure are not secure

- Introduces the Many-sided RowHammer attack
  - Idea: Hammer many rows to bypass TRR mitigations (e.g., by overflowing proprietary TRR tables that detect aggressor rows)

- (Partially) reverse-engineers the TRR and pTRR mitigation mechanisms implemented in DRAM chips and memory controllers

- Provides an automatic tool that can effectively create many-sided RowHammer attacks in DDR4 and LPDDR4(X) chips
Example Many-Sided Hammering Patterns

Fig. 12: Hammering patterns discovered by TRRespass. Aggressor rows are in red (■) and victim rows are in blue (□).
BitFlips vs. Number of Aggressor Rows

Fig. 10: Bit flips vs. number of aggressor rows. Module $C_{12}$: Number of bit flips in bank 0 as we vary the number of aggressor rows. Using SoftMC, we refresh DRAM with standard tREFI and run the tests until each aggressor rows is hammered 500K times.

Fig. 11: Bit flips vs. number of aggressor rows. Module $A_{15}$: Number of bit flips in bank 0 as we vary the number of aggressor rows. Using SoftMC, we refresh DRAM with standard tREFI and run the tests until each aggressor rows is hammered 500K times.

Fig. 13: Bit flips vs. number of aggressor rows. Module $A_{10}$: Number of bit flips triggered with $N$-sided RowHammer for varying number of $N$ on Intel Core i7-7700K. Each aggressor row is one row away from the closest aggressor row (i.e., VAVAVA... configuration) and aggressor rows are hammered in a round-robin fashion.
Infrastructures to Understand Such Issues

Components of In-DRAM TRR

**Sampler**
- Tracks aggressor rows activations
- Design options:
  - Frequency based (record every $N^{th}$ row activation)
  - Time based (record first $N$ row activations)
  - Random seed (record based on a coin flip)
- Regardless, the sampler has a limited size

**Inhibitor**
- Prevents bit flips by refreshing victim rows
  - The latency of performing victim row refreshes is squeezed into slack time available in $tRFC$ (i.e., the latency of regular Refresh command)
# TRRespass Vulnerable DRAM Modules

## TABLE II: TRRespass results

We report the number of patterns found and bit flips detected for the 42 DRAM modules in our set.

<table>
<thead>
<tr>
<th>Module</th>
<th>Date (yy-ww)</th>
<th>Freq. (MHz)</th>
<th>Size (GB)</th>
<th>Organization</th>
<th>MAC</th>
<th>Found Patterns</th>
<th>Best Pattern</th>
<th>Corruptions</th>
<th>Double Refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ranks Banks  Pins</td>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td>1 → 0 0 → 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{0,1,2,3}$</td>
<td>16-37</td>
<td>2132</td>
<td>4</td>
<td>1 16 x8</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>$A_{4}$</td>
<td>16-51</td>
<td>2132</td>
<td>4</td>
<td>1 16 x8</td>
<td>UL</td>
<td>4</td>
<td>9-sided</td>
<td>7956</td>
<td>4008</td>
</tr>
<tr>
<td>$A_{5}$</td>
<td>18-51</td>
<td>2400</td>
<td>4</td>
<td>1 8 x16</td>
<td>UL</td>
<td>33</td>
<td>19-sided</td>
<td>20808</td>
<td>10289</td>
</tr>
<tr>
<td>$A_{6,7}$</td>
<td>18-15</td>
<td>2666</td>
<td>4</td>
<td>1 8 x16</td>
<td>UL</td>
<td></td>
<td>33</td>
<td>24854</td>
<td>12580</td>
</tr>
<tr>
<td>$A_{8}$</td>
<td>17-09</td>
<td>2400</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td>33</td>
<td>19-sided</td>
<td>11342</td>
<td>1809</td>
</tr>
<tr>
<td>$A_{9}$</td>
<td>17-31</td>
<td>2400</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td>523</td>
<td>10-sided</td>
<td>12830</td>
<td>1682</td>
</tr>
<tr>
<td>$A_{10}$</td>
<td>19-02</td>
<td>2400</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>488</td>
<td>10-sided</td>
<td>32723</td>
<td>16490</td>
</tr>
<tr>
<td>$A_{11}$</td>
<td>19-02</td>
<td>2400</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>120</td>
<td>14-sided</td>
<td>22397</td>
<td>12351</td>
</tr>
<tr>
<td>$A_{12,13}$</td>
<td>18-50</td>
<td>2666</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>2</td>
<td>23004</td>
<td>1320</td>
</tr>
<tr>
<td>$A_{14}$</td>
<td>19-08†</td>
<td>3200</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>2</td>
<td>9-sided</td>
<td>12900</td>
<td>8900</td>
</tr>
<tr>
<td>$A_{15}†$</td>
<td>17-08</td>
<td>2132</td>
<td>4</td>
<td>1 16 x8</td>
<td>UL</td>
<td>2</td>
<td>9-sided</td>
<td>12900</td>
<td>8900</td>
</tr>
<tr>
<td>$B_{0}$</td>
<td>18-11</td>
<td>2666</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>$B_{1}$</td>
<td>18-11</td>
<td>2666</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>$B_{2}$</td>
<td>18-49</td>
<td>3000</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$B_{3}$</td>
<td>19-08†</td>
<td>3000</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>1210</td>
<td>—</td>
</tr>
<tr>
<td>$B_{4,5}$</td>
<td>19-08†</td>
<td>2666</td>
<td>8</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>2</td>
<td>10000</td>
<td>—</td>
</tr>
<tr>
<td>$B_{6,7}$</td>
<td>19-08†</td>
<td>2400</td>
<td>4</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>5200</td>
<td>2600</td>
</tr>
<tr>
<td>$B_{8,9}$</td>
<td>19-08†</td>
<td>2400</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>1230</td>
<td>—</td>
</tr>
<tr>
<td>$B_{10,11}$</td>
<td>16-13†</td>
<td>2132</td>
<td>8</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>22397</td>
<td>12351</td>
</tr>
<tr>
<td>$C_{0,1}$</td>
<td>18-46</td>
<td>2666</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>$C_{2,3}$</td>
<td>19-08†</td>
<td>2800</td>
<td>4</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>10000</td>
<td>—</td>
</tr>
<tr>
<td>$C_{4,5}$</td>
<td>19-08†</td>
<td>3000</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>5200</td>
<td>2600</td>
</tr>
<tr>
<td>$C_{6,7}$</td>
<td>19-08†</td>
<td>3000</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>1230</td>
<td>—</td>
</tr>
<tr>
<td>$C_{8}$</td>
<td>19-08†</td>
<td>3200</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>2</td>
<td>10004</td>
<td>—</td>
</tr>
<tr>
<td>$C_{9}$</td>
<td>18-47</td>
<td>2666</td>
<td>16</td>
<td>2 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>10004</td>
<td>—</td>
</tr>
<tr>
<td>$C_{10,11}$</td>
<td>19-04</td>
<td>2933</td>
<td>8</td>
<td>1 16 x8</td>
<td>UL</td>
<td></td>
<td>3</td>
<td>694</td>
<td>239</td>
</tr>
<tr>
<td>$C_{12}†$</td>
<td>15-01†</td>
<td>2132</td>
<td>4</td>
<td>1 16 x8</td>
<td>UT</td>
<td>25</td>
<td>10-sided</td>
<td>190037</td>
<td>63904</td>
</tr>
<tr>
<td>$C_{13}†$</td>
<td>18-49</td>
<td>2132</td>
<td>4</td>
<td>1 16 x8</td>
<td>UT</td>
<td>3</td>
<td>9-sided</td>
<td>694</td>
<td>239</td>
</tr>
</tbody>
</table>

† The module does not report manufacturing date. Therefore, we report purchase date as an approximation.
‡ Analyzed using the FPGA-based SoftMC.
§ The system runs with double refresh frequency in standard conditions. We configured the refresh interval to be 64 ms in the BIOS settings.

UL = Unlimited  UT = Untested
**TABLE III: LPDDR4(X) results.** Mobile phones tested against TRRespass on ARMv8 sorted by production date. We found bit flip inducing RowHammer patterns on 5 out of 13 mobile phones.

<table>
<thead>
<tr>
<th>Mobile Phone</th>
<th>Year</th>
<th>SoC</th>
<th>Memory (GB)</th>
<th>Found Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Pixel</td>
<td>2016</td>
<td>MSM8996</td>
<td>4†</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 2</td>
<td>2017</td>
<td>MSM8998</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Samsung G960F/DS</td>
<td>2018</td>
<td>Exynos 9810</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Huawei P20 DS</td>
<td>2018</td>
<td>Kirin 970</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Sony XZ3</td>
<td>2018</td>
<td>SDM845</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>HTC U12+</td>
<td>2018</td>
<td>SDM845</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>LG G7 ThinQ</td>
<td>2018</td>
<td>SDM845</td>
<td>4†</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 3</td>
<td>2018</td>
<td>SDM845</td>
<td>4</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 4</td>
<td>2019</td>
<td>SM8150</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>OnePlus 7</td>
<td>2019</td>
<td>SM8150</td>
<td>8</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung G970F/DS</td>
<td>2019</td>
<td>Exynos 9820</td>
<td>6</td>
<td>✓</td>
</tr>
<tr>
<td>Huawei P30 DS</td>
<td>2019</td>
<td>Kirin 980</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>Xiaomi Redmi Note 8 Pro</td>
<td>2019</td>
<td>Helio G90T</td>
<td>6</td>
<td>—</td>
</tr>
</tbody>
</table>

† LPDDR4 (not LPDDR4X)
**TABLE IV: Time to exploit.** Time to find the first exploitable template on two sample modules from each DRAM vendor.

<table>
<thead>
<tr>
<th>Module</th>
<th>$\tau (ms)$</th>
<th>$PTE$</th>
<th>RSA-2048</th>
<th>sudo</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{14}$</td>
<td>188.7</td>
<td>4.9s</td>
<td>6m 27s</td>
<td>—</td>
</tr>
<tr>
<td>$A_4$</td>
<td>180.8</td>
<td>38.8s</td>
<td>39m 28s</td>
<td>—</td>
</tr>
<tr>
<td>$B_1$</td>
<td>360.7</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_2$</td>
<td>331.2</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{12}$</td>
<td>300.0</td>
<td>2.3s</td>
<td>74.6s</td>
<td>54m 16s</td>
</tr>
<tr>
<td>$C_{13}$</td>
<td>180.9</td>
<td>3h 15m</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

$\tau$: Time to template a single row: time to fill the victim and aggressor rows + hammer time + time to scan the row.
TRRespass Key Results

- 13 out of 42 tested DDR4 DRAM modules are vulnerable
  - From all 3 major manufacturers
  - 3-, 9-, 10-, 14-, 19-sided hammer attacks needed

- 5 out of 13 mobile phones tested vulnerable
  - From 4 major manufacturers
  - With LPDDR4(X) DRAM chips

- These results are scratching the surface
  - TRRespass tool is not exhaustive
  - There is a lot of room for uncovering more vulnerable chips and phones
RowHammer is still an open problem

Security by obscurity is likely not a good solution
Detailed Lecture on TRRespass

- Computer Architecture, Fall 2020, Lecture 5a
  - RowHammer in 2020: TRRespass (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9

https://www.youtube.com/onurmutlulectures
Industry-Adopted Solutions Do Not Work

- 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)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Lecture Video (59 minutes)]
[Source Code]
[Web Article]
Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

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*  Kaveh Razavi*

*Vrije Universiteit Amsterdam  †ETH Zürich  ¶Qualcomm Technologies Inc.
How to Guarantee That a Chip is RowHammer-Free?
Hard to Guarantee RowHammer-Free Chips

- 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
Uncovering TRR
Almost Completely
Industry-Adopted Solutions Are Very Poor

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,

"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"

Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (25 minutes)]
[Lightning Talk Video (100 seconds)]
[arXiv version]

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan†  Yahya Can Tuğrul†‡  Jeremie S. Kim†  Victor van der Veenσ
Kaveh Razavi†  Onur Mutlu†

†ETH Zürich  ‡TOBB University of Economics & Technology  σQualcomm Technologies Inc.
**U-TRR Summary & Key Results**

**Target Row Refresh (TRR):**
a set of obscure, undocumented, and proprietary RowHammer mitigation techniques

We cannot easily study the security properties of TRR

Is TRR fully secure? How can we validate its security guarantees?

**U-TRR**

A new methodology that leverages data retention failures to uncover the inner workings of TRR and study its security

15x Vendor A DDR4 modules
15x Vendor B DDR4 modules
15x Vendor C DDR4 modules

All 45 modules we test are vulnerable

99.9% of rows in a DRAM bank experience at least one RowHammer bit flip

Up to 7 RowHammer bit flips in an 8-byte dataword, making ECC ineffective

TRR does not provide security against RowHammer

U-TRR can facilitate the development of new RowHammer attacks and more secure RowHammer protection mechanisms

SAFARI
Overview of U-TRR

**U-TRR**: A new methodology to *uncover* the inner workings of TRR

**Key idea**: Use data retention failures as a side channel to detect when a row is refreshed by TRR
Analyzing TRR-Protected DDR4 Chips

*SoftMC [Hassan+, HPCA’17] enhanced for DDR4*
U-TRR Analysis Summary

15x Vendor A DDR4 DRAM modules

15x Vendor B DDR4 DRAM modules

15x Vendor C DDR4 DRAM modules

U-TRR

new RowHammer access patterns that circumvent TRR
Key Takeaways

All 45 modules we test are vulnerable

99.9% of rows in a DRAM bank experience at least one RowHammer bit flip

ECC is ineffective: up to 7 RowHammer bit flips in an 8-byte dataword

<table>
<thead>
<tr>
<th>Module</th>
<th>Date (yyww)</th>
<th>Chip Density (Gbit)</th>
<th>Organization</th>
<th>$HC_{first}^+$</th>
<th>Our Key TRR Observations and Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>19-50</td>
<td>8</td>
<td>1 16 8</td>
<td>16K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 4 73.3% 1.16</td>
</tr>
<tr>
<td>A1-5</td>
<td>19-36</td>
<td>8</td>
<td>1 8 16</td>
<td>13K-15K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 4 99.2% - 99.4% 2.32 - 4.73</td>
</tr>
<tr>
<td>A6-7</td>
<td>19-45</td>
<td>8</td>
<td>1 8 16</td>
<td>13K-15K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 4 99.3% - 99.4% 2.12 - 3.86</td>
</tr>
<tr>
<td>A8-9</td>
<td>20-07</td>
<td>8</td>
<td>1 16 8</td>
<td>12K-14K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 4 74.6% - 75.0% 1.96 - 2.96</td>
</tr>
<tr>
<td>A10-12</td>
<td>19-51</td>
<td>8</td>
<td>1 16 8</td>
<td>12K-13K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 4 74.6% - 75.0% 1.48 - 2.86</td>
</tr>
<tr>
<td>A13-14</td>
<td>20-31</td>
<td>8</td>
<td>1 8 16</td>
<td>11K-14K</td>
<td>$AT_{RR1}$ Counter-based 16 ✓ 1/9 2 94.3% - 98.6% 1.53 - 2.78</td>
</tr>
<tr>
<td>B0</td>
<td>18-22</td>
<td>4</td>
<td>1 16 8</td>
<td>44K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/4 2 99.9% 2.13</td>
</tr>
<tr>
<td>B1-4</td>
<td>20-17</td>
<td>4</td>
<td>1 16 8</td>
<td>159K-192K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/4 2 23.3% - 51.2% 0.06 - 0.11</td>
</tr>
<tr>
<td>B5-6</td>
<td>16-48</td>
<td>4</td>
<td>1 16 8</td>
<td>44K-50K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/4 2 99.9% 1.85 - 2.03</td>
</tr>
<tr>
<td>B7</td>
<td>19-06</td>
<td>8</td>
<td>2 16 8</td>
<td>20K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/4 2 99.9% 31.14</td>
</tr>
<tr>
<td>B8</td>
<td>18-03</td>
<td>4</td>
<td>1 16 8</td>
<td>43K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/4 2 99.9% 2.57</td>
</tr>
<tr>
<td>B9-12</td>
<td>19-48</td>
<td>8</td>
<td>1 16 8</td>
<td>42K-65K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/9 2 36.3% - 38.9% 16.83 - 24.26</td>
</tr>
<tr>
<td>B13-14</td>
<td>20-08</td>
<td>4</td>
<td>1 16 8</td>
<td>11K-14K</td>
<td>$BT_{RR1}$ Sampling-based 1 × 1/2 4 99.9% 16.20 - 18.12</td>
</tr>
<tr>
<td>C0-3</td>
<td>16-48</td>
<td>4</td>
<td>1 16 x8</td>
<td>137K-194K</td>
<td>$CT_{RR1}$ Mix Unknown ✓ 1/17 2 1.0% - 23.2% 0.05 - 0.15</td>
</tr>
<tr>
<td>C4-6</td>
<td>17-12</td>
<td>8</td>
<td>1 16 x8</td>
<td>130K-150K</td>
<td>$CT_{RR1}$ Mix Unknown ✓ 1/17 2 7.8% - 12.0% 0.06 - 0.08</td>
</tr>
<tr>
<td>C7-8</td>
<td>20-31</td>
<td>8</td>
<td>1 8 x16</td>
<td>40K-44K</td>
<td>$CT_{RR1}$ Mix Unknown ✓ 1/17 2 39.8% - 41.8% 9.66 - 14.56</td>
</tr>
<tr>
<td>C9-11</td>
<td>20-31</td>
<td>8</td>
<td>1 8 x16</td>
<td>42K-53K</td>
<td>$CT_{RR1}$ Mix Unknown ✓ 1/9 2 99.7% 9.30 - 32.04</td>
</tr>
<tr>
<td>C12-14</td>
<td>20-46</td>
<td>16</td>
<td>1 8 x16</td>
<td>6K-7K</td>
<td>$CT_{RR2}$ Mix Unknown ✓ 1/8 2 99.9% 4.91 - 12.64</td>
</tr>
</tbody>
</table>
Effect on Individual Rows

All 45 modules we tested are vulnerable to our new RowHammer access patterns.

Our RowHammer access patterns cause bit flips in more than 99.9% of the rows.
Bypassing ECC with New RowHammer Patterns

Modules from all three vendors have many **8-byte data chunks** with 3 and more (up to 7) RowHammer bit flips.

Conventional DRAM ECC cannot protect against our **new RowHammer access patterns**.
Many Observations & Results in the Paper

- More observations on the TRRs of the three vendors
- Detailed description of the crafted access patterns
- Hammers per aggressor row sensitivity analysis
- Observations and results for individual modules
- ...

<table>
<thead>
<tr>
<th>Module</th>
<th>Date (yy-ww)</th>
<th>Chip Density (Gbit)</th>
<th>Organization</th>
<th>$HC_{first}$</th>
<th>ATRR1</th>
<th>ATRR1</th>
<th>ATRR1</th>
<th>ATRR1</th>
<th>ATRR2</th>
<th>ATRR2</th>
<th>ATRR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>19-50</td>
<td>8</td>
<td>1 16 8</td>
<td>16K</td>
<td>16</td>
<td>1/9</td>
<td>4</td>
<td>73.3%</td>
<td>1.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1-5</td>
<td>19-36</td>
<td>8</td>
<td>1 8 16</td>
<td>13K-15K</td>
<td>16</td>
<td>1/9</td>
<td>4</td>
<td>99.2% - 99.4%</td>
<td>2.32 - 4.73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6-7</td>
<td>19-45</td>
<td>8</td>
<td>1 16 8</td>
<td>13K-15K</td>
<td>16</td>
<td>1/9</td>
<td>4</td>
<td>99.3% - 99.4%</td>
<td>2.12 - 3.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8-9</td>
<td>20-07</td>
<td>8</td>
<td>1 16 8</td>
<td>12K-14K</td>
<td>16</td>
<td>1/9</td>
<td>4</td>
<td>74.6% - 75.0%</td>
<td>1.96 - 2.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A10-12</td>
<td>19-51</td>
<td>8</td>
<td>1 16 8</td>
<td>12K-13K</td>
<td>16</td>
<td>1/9</td>
<td>4</td>
<td>74.6% - 75.0%</td>
<td>1.48 - 2.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A13-14</td>
<td>20-31</td>
<td>8</td>
<td>1 8 16</td>
<td>11K-14K</td>
<td>16</td>
<td>1/9</td>
<td>2</td>
<td>94.3% - 98.6%</td>
<td>1.53 - 2.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>18-22</td>
<td>4</td>
<td>1 16 8</td>
<td>44K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>99.9%</td>
<td>2.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B1-4</td>
<td>20-17</td>
<td>4</td>
<td>1 16 8</td>
<td>159K-192K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>23.3% - 51.2%</td>
<td>0.06 - 0.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5-6</td>
<td>16-48</td>
<td>4</td>
<td>1 16 8</td>
<td>44K-50K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>99.9%</td>
<td>1.85 - 2.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td>19-06</td>
<td>8</td>
<td>2 16 8</td>
<td>20K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>99.9%</td>
<td>31.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B8</td>
<td>18-03</td>
<td>4</td>
<td>1 16 8</td>
<td>43K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>99.9%</td>
<td>2.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B9-12</td>
<td>19-48</td>
<td>8</td>
<td>1 16 8</td>
<td>42K-65K</td>
<td>16</td>
<td>1/4</td>
<td>2</td>
<td>36.3% - 38.9%</td>
<td>16.83 - 24.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B13-14</td>
<td>20-08</td>
<td>4</td>
<td>1 16 8</td>
<td>11K-14K</td>
<td>16</td>
<td>1/2</td>
<td>4</td>
<td>99.9%</td>
<td>16.20 - 18.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0-3</td>
<td>16-48</td>
<td>4</td>
<td>1 16 x8</td>
<td>137K-194K</td>
<td>Mix</td>
<td>Unknown</td>
<td>1/17</td>
<td>2</td>
<td>1.0% - 23.2%</td>
<td>0.05 - 0.15</td>
<td></td>
</tr>
<tr>
<td>C4-6</td>
<td>17-12</td>
<td>8</td>
<td>1 16 x8</td>
<td>130K-150K</td>
<td>Mix</td>
<td>Unknown</td>
<td>1/17</td>
<td>2</td>
<td>7.8% - 12.0%</td>
<td>0.06 - 0.08</td>
<td></td>
</tr>
<tr>
<td>C7-8</td>
<td>20-31</td>
<td>8</td>
<td>1 8 x16</td>
<td>40K-44K</td>
<td>Mix</td>
<td>Unknown</td>
<td>1/17</td>
<td>2</td>
<td>39.8% - 41.8%</td>
<td>9.66 - 14.56</td>
<td></td>
</tr>
<tr>
<td>C9-11</td>
<td>20-31</td>
<td>8</td>
<td>1 8 x16</td>
<td>42K-53K</td>
<td>Mix</td>
<td>Unknown</td>
<td>1/9</td>
<td>2</td>
<td>99.7%</td>
<td>9.30 - 32.04</td>
<td></td>
</tr>
<tr>
<td>C12-14</td>
<td>20-46</td>
<td>16</td>
<td>1 8 x16</td>
<td>6K-7K</td>
<td>Mix</td>
<td>Unknown</td>
<td>1/8</td>
<td>2</td>
<td>99.9%</td>
<td>4.91 - 12.64</td>
<td></td>
</tr>
</tbody>
</table>
Uncovering TRR Can Help Future Solutions

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,

"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"

Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (25 minutes)]
[Lightning Talk Video (100 seconds)]
[arXiv version]
New RowHammer Characteristics
RowHammer Has Many Dimensions

Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
"A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (21 minutes)]
[Lightning Talk Video (1.5 minutes)]
[arXiv version]

A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses
Our Goal

Provide insights into **three fundamental properties**

- **Temperature**
- **Aggressor Row Active Time**
- **Victim DRAM Cell’s Physical Location**

To find **effective and efficient** attacks and defenses
DRAM Testing Infrastructures

Two separate testing infrastructures

1. DDR3: FPGA-based SoftMC (Xilinx ML605)
2. DDR4: FPGA-based SoftMC (Xilinx Virtex UltraScale+ XCU200)

Fine-grained control over DRAM commands, timing parameters and temperature (±0.1°C)
## DRAM Chips Tested

<table>
<thead>
<tr>
<th>Mfr.</th>
<th>DDR4 DIMMs</th>
<th>DDR3 SODIMMs</th>
<th># Chips</th>
<th>Density</th>
<th>Die</th>
<th>Org.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Micron)</td>
<td>9</td>
<td>1</td>
<td>144 (8)</td>
<td>8Gb (4Gb)</td>
<td>B (P)</td>
<td>x4 (x8)</td>
</tr>
<tr>
<td>B (Samsung)</td>
<td>4</td>
<td>1</td>
<td>32 (8)</td>
<td>4Gb (4Gb)</td>
<td>F (Q)</td>
<td>x8 (x8)</td>
</tr>
<tr>
<td>C (SK Hynix)</td>
<td>5</td>
<td>1</td>
<td>40 (8)</td>
<td>4Gb (4Gb)</td>
<td>B (B)</td>
<td>x8 (x8)</td>
</tr>
<tr>
<td>D (Nanya)</td>
<td>4</td>
<td>-</td>
<td>32 (-)</td>
<td>8Gb (-)</td>
<td>C (-)</td>
<td>x8 (-)</td>
</tr>
</tbody>
</table>

- **Two DRAM standards**
- **4 Major Manufacturers**
- **272 DRAM Chips in total**
Summary of The Study & Key Results

• 272 DRAM chips from four major manufacturers

• 6 major takeaways from 16 novel observations

• A RowHammer bit flip is more likely to occur
  1) in a bounded range of temperature
  2) if the aggressor row is active for longer time
  3) in certain physical regions of the DRAM module under attack

• Our novel observations can inspire and aid future work
  - Craft more effective attacks
  - Design more effective and efficient defenses
Example Attack Improvement 3: Bypassing Defenses with Aggressor Row Active Time

Activating aggressor rows as frequently as possible:

![Diagram showing activation of rows over time]

Keeping aggressor rows active for a longer time:

![Diagram showing longer active time for aggressor rows]

Reduces the minimum activation count to induce a bit flip by 36%

Bypasses defenses that do not account for this reduction
### Key Takeaways from Spatial Variation Analysis

#### Key Takeaway 5

RowHammer vulnerability *significantly varies* across DRAM rows and columns due to *design-induced* and *manufacturing-process-induced* variation.

#### Key Takeaway 6

The distribution of the minimum activation count to observe bit flips ($HC_{first}$) exhibits a *diverse set of values in a subarray* but *similar values across subarrays* in the same DRAM module.
Spatial Variation across Rows

The **minimum activation count** to observe bit flips \((HC_{first})\) across DRAM rows:

![Graph showing minimum activation count](image)

The RowHammer vulnerability **significantly varies** across DRAM rows.
Spatial Variation across Rows

The RowHammer vulnerability **significantly varies** across DRAM rows.
Spatial Variation across Rows

OBSERVATION 12

A small fraction of DRAM rows are significantly more vulnerable to RowHammer than the vast majority of the rows.
Spatial Variation across Columns

OBSERVATION 13

Certain columns are **significantly more vulnerable** to RowHammer than other columns.
Example Defense Improvements

• **Example 1: Leveraging variation across DRAM rows**
  
  - Breakdown of DRAM Rows
  
  - $10\%$ $\rightarrow HC_{first}$
  
  - $90\%$ $\rightarrow 2 \times HC_{first}$

  - Aggressiveness can be reduced:
    - **33\% area reduction** for BlockHammer [Yağlıkçı+, HPCA'21]
    - **80\% area reduction** for Graphene [Park+, MICRO'20]

• **Example 2: Leveraging variation with temperature**
  
  - A DRAM cell experiences **bit flips** within a **bounded temperature range**
    - no bit flips $\rightarrow$ **Vulnerable Temperature Range** $\rightarrow$ no bit flips

  - A row can be **disabled** within the row’s **vulnerable temperature range**
Many More Analyses In The Paper

Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,

"A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"

Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (21 minutes)]
[Lightning Talk Video (1.5 minutes)]
[arXiv version]
More RowHammer Analysis
RowHammer vs. Wordline Voltage (2022)

- A. Giray Yaşlıkçı, Haocong Luo, Geraldo F. de Oliviera, Ataberk Olgun, Minesh Patel, Jisung Park, Hasan Hassan, Jeremie S. Kim, Lois Orosa, and Onur Mutlu, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices"

Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Baltimore, MD, USA, June 2022.

[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[arXiv version]
[Talk Video (34 minutes, including Q&A)]
[Lightning Talk Video (2 minutes)]

Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices

A. Giray Yaşlıkçı¹ Haoccong Luo¹ Geraldo F. de Oliviera¹ Ataberk Olgun¹ Minesh Patel¹ Jisung Park¹ Hasan Hassan¹ Jeremie S. Kim¹ Lois Orosa¹,² Onur Mutlu¹

¹ETH Zürich ²Galicia Supercomputing Center (CESGA)
Updated DRAM Testing Infrastructure

FPGA-based SoftMC (Xilinx Virtex UltraScale+ XCU200)

Fine-grained control over DRAM commands, timing parameters (±1.5ns), temperature (±0.1°C), and wordline voltage (±1mV)

Summary
We provide the first RowHammer characterization under reduced wordline voltage

Experimental results with 272 real DRAM chips show that reducing wordline voltage:

1. Reduces RowHammer vulnerability
   - Bit error rate caused by a RowHammer attack reduces by 15.2% (66.9% max)
   - A row needs to be activated 7.4% more times (85.8% max) to induce the first bit flip

2. Increases row activation latency
   - More than 76% of the tested DRAM chips reliably operate using nominal timing parameters
   - Remaining 24% reliably operate with increased (up to 24ns) row activation latency

3. Reduces data retention time
   - 80% of the tested DRAM chips reliably operate using nominal refresh rate
   - Remaining 20% reliably operate by
     - Using single error correcting codes
     - Doubling the refresh rate for a small fraction (16.4%) of DRAM rows

Reducing wordline voltage can reduce RowHammer vulnerability without significantly affecting reliable DRAM operation
New RowHammer Solutions
BlockHammer Solution in 2021

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Intel Hardware Security Academic Awards Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]
[Intel Hardware Security Academic Awards Short Talk Video (2 minutes)]
[BlockHammer Source Code]

Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations)

---

BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

A. Giray Yağlıkçı¹  Minesh Patel¹  Jeremie S. Kim¹  Roknoddin Azizi¹  Ataberk Olgun¹  Lois Orosa¹  Hasan Hassan¹  Jisung Park¹  Konstantinos Kanellopoulos¹  Taha Shahroodi¹  Saugata Ghose²  Onur Mutlu¹

¹ETH Zürich  ²University of Illinois at Urbana–Champaign

SAFARI
RowHammer Solution Approaches

- More robust DRAM chips and/or error-correcting codes
- Increased refresh rate
- Physical isolation
- Reactive refresh
- Proactive throttling

Cost, Power, Performance, Complexity

Fewer activations possible in a refresh interval

Large-enough distance

Fewer activations allowed for aggressive applications
Two Key Challenges

1. Scalability with worsening RowHammer vulnerability

2. Compatibility with commodity DRAM chips
Our Goal

To prevent RowHammer *efficiently and scalably* *without knowledge of or modifications to* DRAM internals
BlockHammer
Key Idea

**Selectively throttle** memory accesses that may cause **RowHammer bit-flips**
A RowHammer attack hammers Row A

BlockHammer detects a RowHammer attack using area-efficient Bloom filters

BlockHammer selectively throttles accesses from within the memory controller

Bit flips do not occur

BlockHammer can optionally inform the system software about the attack

BlockHammer is compatible with commodity DRAM chips
No need for proprietary info or modifications to DRAM chips
BlockHammer
Overview of Approach

**RowBlocker**
Tracks row activation rates using area-efficient Bloom filters
Blacklists rows that are activated at a high rate
Throttles activations targeting a blacklisted row

No row can be activated at a high enough rate to induce bit-flips

**AttackThrottler**
Identifies threads that perform a RowHammer attack
Reduces memory bandwidth usage of identified threads

Greatly reduces the **performance degradation** and **energy wastage** a RowHammer attack inflicts on a system
Evaluation: BlockHammer
Scaling with RowHammer Vulnerability

- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

BlockHammer’s performance and energy overheads remain negligible (<0.6%)

BlockHammer scalably provides much higher performance (71% on average) and lower energy consumption (32% on average) than state-of-the-art mechanisms.
Key Results: BlockHammer

- **Competitive** with state-of-the-art mechanisms *when there is no attack*
- **Superior** performance and DRAM energy *when RowHammer attack present*
- **Better hardware area scaling with RowHammer vulnerability**
- **Security Proof**
- **Addresses Many-Sided Attacks**

- Evaluation of **14 mechanisms** across four desirable properties
  - Comprehensive Protection
  - Compatibility with Commodity DRAM Chips
  - Scalability with RowHammer Vulnerability
  - Deterministic Protection

<table>
<thead>
<tr>
<th>Approach</th>
<th>Mechanism</th>
<th>Comprehensive Protection</th>
<th>Compatable w/ Commodity DRAM Chips</th>
<th>Scalable with RowHammer Vulnerability</th>
<th>Deterministic Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased Refresh Rate</td>
<td>[2, 73]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Physical Isolation</td>
<td>CAT [14]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>GuardION [148]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>ZebRAM [78]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Reactive Refresh</td>
<td>ANVIL [5]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PARA [73]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PRoHiT [137]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>MRLoc [161]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CBT [132]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>TWiCe [84]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Graphene [113]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Proactive Throttling</td>
<td>Naive Thrott. [102]</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Thrott. Supp. [40]</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>BlockHammer</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**BlockHammer is the only solution that satisfies all four desirable properties**
More in the Paper: BlockHammer

- Using area-efficient Bloom filters for RowHammer detection
- Security Proof
  - Mathematically represent all possible access patterns
  - No row can be activated high-enough times to induce bit-flips
- BlockHammer prevents many-sided attacks
  - TRRespass [Frigo+, S&P’20]
  - U-TRR [Hassan+, MICRO’21]
  - BlackSmith [Jattke+, S&P’22]
  - Half-Double [Kogler+, USENIX Security’22]
- System Integration
  - BlockHammer can detect RowHammer attacks with high accuracy and inform system software
  - Measures RowHammer likelihood of each thread
- Hardware complexity analysis
Summary: BlockHammer

- BlockHammer is **the first work to practically enable throttling-based RowHammer mitigation**

- BlockHammer is implemented in **the memory controller** *(no proprietary information of / no modifications to DRAM chips)*

- BlockHammer is **both scalable with worsening RowHammer and compatible with commodity DRAM chips**

- BlockHammer is **open-source** along with **six state-of-the-art mechanisms**: [https://github.com/CMU-SAFARI/BlockHammer](https://github.com/CMU-SAFARI/BlockHammer)
BlockHammer Solution

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Intel Hardware Security Academic Awards Finalist (one of 4 finalists out of 34 nominations)]
A Takeaway

Main Memory Needs

Intelligent Controllers

for Security, Safety, Reliability, Scaling
More RowHammer in 2020-2022
RowHammer in 2020 (I)

Session 1A: Security & Privacy I

5:00 PM CEST – 5:15 PM CEST
Graphene: Strong yet Lightweight Row Hammer Protection
Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, Jae W. Lee (Seoul National University)

5:15 PM CEST – 5:30 PM CEST
Persist Level Parallelism: Streamlining Integrity Tree Updates for Secure Persistent Memory
Alexander Freij, Shougang Yuan, Huiyang Zhou (NC State University); Yan Solihin (University of Central Florida)

5:30 PM CEST – 5:45 PM CEST
PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses
Zhi Zhang (University of New South Wales and Data61, CSIRO, Australia); Yueqiang Cheng (Baidu Security); Dongxi Liu, Surya Nepal (Data61, CSIRO, Australia); Zhi Wang (Florida State University); Yuval Yarom (University of Adelaide and Data61, CSIRO, Australia)
RowHammer in 2020 (II)

Session #5: Rowhammer

Session chair: Michael Franz (UC Irvine)

**RAMBleed: Reading Bits in Memory Without Accessing Them**
Andrew Kwong (University of Michigan), Daniel Genkin (University of Michigan), Daniel Gruss (Data61)

**Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers**
Lucian Cojocar (Microsoft Research), Jeremie Kim (ETH Zurich, CMU), Minesh Patel (ETH Zurich, Microsoft Research), Onur Mutlu (ETH Zurich, CMU)

**Leveraging EM Side-Channel Information to Detect Rowhammer Attacks**
Zhenkai Zhang (Texas Tech University), Zihao Zhan (Vanderbilt University), Daniel Balasubram Peter Völgyesi (Vanderbilt University), Xenofon Koutsoukos (Vanderbilt University)

**TRRespass: Exploiting the Many Sides of Target Row Refresh**
Pietro Frigo (Vrije Universiteit Amsterdam, The Netherlands), Emanuele Vannacci (Vrije Universiteit Amsterdam, Qualcomm Technologies, Inc.), Onur Mutlu (ETH Zürich), Cristiano Giuffrida (Vrije Universiteit Amsterdam, The Netherlands), Kaveh Razavi (Vrije Universiteit Amsterdam, The Netherlands)
DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips
Fan Yao, University of Central Florida; Adnan Siraj Rakin and Deliang Fan, Arizona State University

AVAILABLE MEDIA □ ▬ ▶
Show details ▶
Stop! Hammer Time: Rethinking Our Approach to Rowhammer Mitigations
SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript
RowHammer in 2021 (III)

Session 10A: Security & Privacy III

Session Chair: Hoda Naghibjouybari (Binghamton)

9:00 PM CEST – 9:15 PM CEST
A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses
Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo (ETH Zurich); Ataberk Olgun (TOBB University of Economics and Technology); Jisung Park, Hasan Hassan, Miness Patel, Jeremie S. Kim, Onur Mutlu (ETH Zurich)

Paper

9:15 PM CEST – 9:30 PM CEST
Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications
Hasan Hassan (ETH Zurich); Yahya Can Tugrul (TOBB University of Economics and Technology); Jeremie S. Kim (ETH Zurich); Victor van der Veen (Qualcomm); Kaveh Razavi, Onur Mutlu (ETH Zurich)

Paper
RowHammer in 2022 (I)

43rd IEEE Symposium on Security and Privacy

BLACKSMITH: Scalable Rowhammering in the Frequency Domain

SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks

PROTRR: Principled yet Optimal In-DRAM Target Row Refresh

DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories
RowHammer in 2022 (II)

Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation between Aggressor and Victim Rows
RowHammer in 2022 (III)

**HPCA 2022**
The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28), Seoul, South Korea

**SafeGuard: Reducing the Security Risk from Row-Hammer via Low-Cost Integrity Protection**

**Mithril: Cooperative Row Hammer Protection on Commodity DRAM Leveraging Managed Refresh**
RowHammer in 2022 (IV)

The Price of Secrecy: How Hiding Internal DRAM Topologies Hurts Rowhammer Defenses

Stefan Saroiu, Alec Wolman, Lucian Cojocar
Microsoft
Half-Double: Hammering From the Next Row Over

Andreas Kogler\textsuperscript{1} Jonas Juffinger\textsuperscript{1,2} Salman Qazi\textsuperscript{3} Yoongu Kim\textsuperscript{3} Moritz Lipp\textsuperscript{4*}
Nicolas Boichat\textsuperscript{3} Eric Shiu\textsuperscript{5} Mattias Nissler\textsuperscript{3} Daniel Gruss\textsuperscript{1}

\textsuperscript{1}Graz University of Technology \textsuperscript{2}Lamarr Security Research \textsuperscript{3}Google
\textsuperscript{4}Amazon Web Services \textsuperscript{5}Rivos
HAMMERSCOPE: Observing DRAM Power Consumption Using Rowhammer

When Frodo Flips: End-to-End Key Recovery on FrodoKEM via Rowhammer
AQUA: Scalable Rowhammer Mitigation by Quarantining Aggressor Rows at Runtime
Anish Saxena, Gururaj Saileshwar (Georgia Institute of Technology); Prashant J. Nair (University of British Columbia); Moinuddin Qureshi (Georgia Institute of Technology)

HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips
Abdullah Giray Yaglikci (ETH Zürich); Ataberk Olgun (TOBB University of Economics and Technology); Lois Orosa, Minesh Patel, Haocong Luo, Hasan Hassan (ETH Zürich); Oguz Ergin (TOBB University of Economics and Technology); Onur Mutlu (ETH Zürich)
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı¹  Ataberk Olgun¹,²  Minesh Patel¹  Haocong Luo¹  Hasan Hassan¹
Lois Orosa¹,³  Oğuz Ergin²  Onur Mutlu¹

¹ETH Zürich  ²TOBB University of Economics and Technology  ³Galicia Supercomputing Center (CESGA)
A Case for Transparent Reliability in DRAM Systems

Minesh Patel† Taha Shahroodi†† Aditya Manglik† A. Giray Yağlıkçı†
Ataberk Olgun† Haocong Luo† Onur Mutlu†
†ETH Zürich ‡TU Delft


Hasan Hassan       Ataberk Olgun       A. Giray Yağlıkçı
Haocong Luo       Onur Mutlu

ETH Zürich

Self-Managing DRAM (SMD) enables autonomous in-DRAM maintenance operations.

### Key Idea:
Prevent the memory controller from accessing DRAM regions that are under maintenance by rejecting row activation (ACT) commands.

Leveraging the ability to reject an ACT, a maintenance operation can be implemented completely within a DRAM chip.
# SMD-Based Maintenance Mechanisms

<table>
<thead>
<tr>
<th>DRAM Refresh</th>
<th>Fixed Rate (SMD-FR)</th>
<th>Variable Rate (SMD-VR)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>uniformly</em> refreshes all DRAM rows with a <em>fixed</em> refresh period</td>
<td><em>skips</em> refreshing rows that can <em>retain their data for longer</em> than the default refresh period</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RowHammer Protection</th>
<th>Probabilistic (SMD-PRP)</th>
<th>Deterministic (SMD-DRP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs <em>neighbor row refresh</em> with a <em>small probability</em> on every row activation</td>
<td><em>keeps track</em> of most frequently activated rows and performs <em>neighbor</em> row refresh when activation count threshold is exceeded</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Scrubbing</th>
<th>Periodic Scrubbing (SMD-MS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>periodically scans the entire</em> DRAM for errors and corrects them</td>
</tr>
</tbody>
</table>
Self-Managing DRAM: Summary

The three major DRAM maintenance operations:
- Refresh
- RowHammer Protection
- Memory Scrubbing

Implementing new **maintenance mechanisms** often requires **difficult-to-realize changes**

**Our Goal**
1. Ease the process of enabling new DRAM maintenance operations
2. Enable more efficient in-DRAM maintenance operations

**Self-Managing DRAM (SMD)**
Enables implementing new **in-DRAM** maintenance mechanisms with **no further changes** in the **DRAM interface** and **memory controller**

SMD-based refresh, RowHammer protection, and scrubbing achieve **9.2% speedup** and **6.2% lower DRAM energy** vs. conventional DRAM
Talk on Self-Managing DRAM

Problem: The Rigid DRAM Interface

The Memory Controller manages DRAM maintenance operations

Changes to maintenance operations are often reflected to the memory controller design, DRAM interface, and other system components.

Implementing new maintenance operations (or modifying the existing ones) is difficult-to-realize.

https://www.youtube.com/watch?v=mGa6-vpExbE
A Case for Self-Managing DRAM Chips:
Improving Performance, Efficiency, Reliability, and Security
via Autonomous in-DRAM Maintenance Operations

Hasan Hassan  Ataberk Olgun  A. Giray Yağlıkçı
Haocong Luo  Onur Mutlu

ETH Zürich

RowHammer in 2023

CSI: Rowhammer – Cryptographic Security and Integrity against Rowhammer

Jonas Juffinger*, Lukas Lamster†, Andreas Kogler†, Maria Eichlseder†, Moritz Lipp‡, Daniel Gruss*†

* Lamarr Security Research, † Graz University of Technology, ‡ Amazon Web Services
More to Come…
Future Memory
Reliability/Security Challenges
Future of Main Memory Security

- 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
Future of Main Memory Security

- DRAM

- Flash memory

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

SAFARI
**Many Errors and Their Mitigation [PIEEE’17]**

**Table 3** List of Different Types of Errors Mitigated by NAND Flash Error Mitigation Mechanisms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shadow Program Sequencing [35,40] (Section V-A)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neighbor-Cell Assisted Error Correction [36] (Section V-B)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refresh [34,39,67,68] (Section V-C)</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Read-Retry [33,72,107] (Section V-D)</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Voltage Optimization [37,38,74] (Section V-E)</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hot Data Management [41,63,70] (Section V-F)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Adaptive Error Mitigation [43,65,77,78,82] (Section V-G)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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
A Takeaway

Main Memory Needs
Intelligent Controllers
for Security, Safety, Reliability, Scaling
The Takeaway

Intelligent Memory Controllers Can Avoid Many Failures & Enable Better Scaling
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
A Case for Transparent Reliability in DRAM Systems

Minesh Patel† Taha Shahroodi†† Aditya Manglik† A. Giray Yağlıkçı†
Ataberk Olgun† Haocong Luo† Onur Mutlu†
†ETH Zürich ‡TU Delft

Better Coordination of DRAM & Controller


Hasan Hassan  Ataberk Olgun  A. Giray Yağlıkçı
Haocong Luo  Onur Mutlu

ETH Zürich

Understand and Model with Experiments (DRAM)

Understand and Model with Experiments (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
Collapse of the “Galloping Gertie” (1940)

Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm
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)

Source: Morry Gash/AP,
https://www.npr.org/2017/08/01/540669701/10-years-after-bridge-collapse-america-is-still-crumbling?t=1535427165809
A More Recent Example (2018)

A Most Recent Example (2022)

A Most Recent Example (2022)

A Most Recent Example (2022)

[Image of a collapsed bridge with a sign indicating a weight limit of 26 tons and a speed limit of 35.]
A Most Recent Example (2022)

[Image of a damaged bridge during winter]

https://www.npr.org/2022/01/28/1076343656/pittsburgh-bridge-collapse-biden-visit
In-Field Patch-ability
(Intelligent Memory)
Can Avoid Such Failures
An Early Proposal for Intelligent Controllers [IMW’13]

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  Proceedings of the 5th International Memory Workshop (IMW), Monterey, CA, May 2013. Slides (pptx) (pdf)
  EETimes Reprint

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

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
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
Using Memory Errors to Attack a Virtual Machine

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

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

IEEE S&P 2003

Using Memory Errors to Attack a Virtual Machine

Sudhakar Govindavajhala* Andrew W. Appel
Princeton University
{sudhakar,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.
A simple memory error can be induced by software.
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 & architecture 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 ASPLOS 2022)
  - 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…
Conclusion
Summary: RowHammer

- Memory 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
A RowHammer Survey Across the Stack

- Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu§‡
§ETH Zürich
Jeremie S. Kim‡§
‡Carnegie Mellon University
Detailed Lectures on RowHammer

- Computer Architecture, Fall 2021, Lecture 5
  - RowHammer (ETH Zürich, Fall 2021)
  - https://www.youtube.com/watch?v=7wVKnPj3NVw&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=5

- Computer Architecture, Fall 2021, Lecture 6
  - RowHammer and Secure & Reliable Memory (ETH Zürich, Fall 2021)
  - https://www.youtube.com/watch?v=HNd4skQrt6I&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=6

https://www.youtube.com/onurmutlulectures
Funding Acknowledgments

- Alibaba, AMD, ASML, Google, Facebook, Hi-Silicon, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware, Xilinx
- NSF
- NIH
- GSRC
- SRC
- CyLab
- EFCL

Thank you!
Acknowledgments

Think BIG, Aim HIGH!

https://safari.ethz.ch
SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-january-2021/

40+ Researchers

Think BIG, Aim HIGH!

https://safari.ethz.ch
SAFARI Research Group

- https://safari.ethz.ch/safari-newsletter-december-2021/

Think Big, Aim High
Comp Arch (Fall 2021)

- **Fall 2021 Edition:**
  - https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule

- **Fall 2020 Edition:**

- **Youtube Livestream (2021):**
  - https://www.youtube.com/watch?v=4yfkM_5EFgo&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF

- **Youtube Livestream (2020):**
  - https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN

- Master’s level course
  - Taken by Bachelor’s/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 5 Simulator-based Lab Assignments
  - Potential research exploration
  - Many research readings

https://www.youtube.com/onurmutlulectures
**DDCA (Spring 2022)**

- **Spring 2022 Edition:**

- **Spring 2021 Edition:**

- **Youtube Livestream (Spring 2022):**
  - [https://www.youtube.com/watch?v=cpXdE3HwvK0&list=PL5Q2soXY2Zi97Ya5DEUpMpO2bbAoAG7c6](https://www.youtube.com/watch?v=cpXdE3HwvK0&list=PL5Q2soXY2Zi97Ya5DEUpMpO2bbAoAG7c6)

- **Youtube Livestream (Spring 2021):**
  - [https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7L1IN](https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7L1IN)

- **Bachelor’s course**
  - 2nd semester at ETH Zurich
  - Rigorous introduction into “How Computers Work”
  - Digital Design/Logic
  - Computer Architecture
  - 10 FPGA Lab Assignments

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
RowHammer & DRAM Exploration (Fall 2022)

- **Fall 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=softmc](https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=softmc)

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=softmc](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=softmc)

- **Youtube Livestream (Spring 2022):**
  - [https://www.youtube.com/watch?v=r5QxuoJWttg&list=PL5Q2soXY2Zi_1trfCckr6PTN8WR72icUO](https://www.youtube.com/watch?v=r5QxuoJWttg&list=PL5Q2soXY2Zi_1trfCckr6PTN8WR72icUO)

- **Bachelor’s course**
  - Elective at ETH Zurich
  - Introduction to DRAM organization & operation
  - Tutorial on using FPGA-based infrastructure
  - Verilog & C++
  - Potential research exploration

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
Exploration of Emerging Memory Systems (Fall 2022)

- **Fall 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=ramulator

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=ramulator

- **Youtube Livestream (Spring 2022):**
  - https://www.youtube.com/watch?v=aMI1XQd3s&list=PL5Q2soXY2Zi_TIlmLGw_Z8hBo2925ZApqV

- **Bachelor’s course**
  - Elective at ETH Zurich
  - Introduction to memory system simulation
  - Tutorial on using Ramulator
  - C++
  - Potential research exploration

https://www.youtube.com/onurmutlulectures
Seminar in Computer Architecture

The Story of RowHammer

Prof. Onur Mutlu

ETH Zürich
Fall 2022
27 October 2022
Backup Slides for Further Info
RowHammer Review History
Some More Historical Perspective

- RowHammer is the first example of a circuit-level failure mechanism causing a widespread system security vulnerability.

- It led to a large body of work in security attacks, mitigations, architectural solutions, analyses, ...

- Work building on RowHammer still continues
  - See MICRO 2022, S&P 2022, and many top venues in 2020-2023

- Initially, it was dismissed by some reviewers
  - Rejected from MICRO 2013 conference
#66 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**
We demonstrate the vulnerability of commodity DRAM chips to disturbance errors. By repeatedly reading from one DRAM address, we show that it is possible to corrupt the data stored.

**Authors**
Y. Kim, R. Daly, J. Lee, J. Kim, C. Fallin, C. Wilkerson, O. Mutlu

**Keywords:** DRAM; errors

**Topics**

<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>
Reviewer A -- Security is Not “Realistic”

**Review #66A**  Modified Friday 5 Jul 2013 3:59:18am PDT

**OVERALL MERIT (?)**

1. Reject

**PAPER SUMMARY**

This work tests and studies the disturbance problem in DRAM arrays in isolation.

**PAPER STRENGTHS**

+ Many results and observations.
+ Insights on how the may happen

**PAPER WEAKNESSES**

- 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

**NOVELTY (?)**


**WRITING QUALITY (?)**

4. Well-written
Reviewer A -- Security is Not “Realistic”

COMMENTS FOR AUTHORS

I found the paper very well written and organized, easy to understand. The topic is interesting and relevant. However, I'm not fully convinced that the disturbance problem is going to be an issue in a realistic DRAM usage scenario (main memory with caches). In that scenario the 64ms refresh interval might be enough. Overall, the work presented, the experimentation and the results are not enough to justify/claim that disturbance may be an issue for future systems, and that microarchitectural solutions are required.

I really encourage the authors to address this issue, to run the new set of experiments; if the results are positive, the work is great and will be easily accepted in a top notch conference. Test scenario in the paper (open-read-close a row many times consecutively) that is used to create disturbances is not likely to show up in a realistic usage scenario (check also rebuttal question).
WILL IT AFFECT REAL WORKLOADS ON REAL SYSTEMS?

Malicious workloads and pathological access-patterns can bypass/thrash the cache and access the same DRAM row a very large number of times. While these workloads may not be common, they are just as real. Using non-temporal
Reviewer A -- Demands

To make sure that correct information and messages are given to the research community, it would be good if the conclusions drawn in the paper were verified with the actual DRAM manufacturers, although I see that it can be difficult to do. In addition, knowing the technology node of each tested DRAM would make the paper stronger and would avoid speculative guesses.

**Reviewer expertise (?)**

4. Expert in area, with highest confidence in review.
Reviewer C – No Architectural Content

Review #66C  Modified Friday 12 Jul 2013 7:38:57am PDT

OVERALL MERIT (?)
2. Weak reject

PAPER SUMMARY
This paper presents a rigorous study of DRAM module errors which are observed to be caused through repeated access to the same address in the DRAMs.

PAPER STRENGTHS
The paper's measurement methodology is outstanding, and the authors very thoroughly dive into different test scenarios, to isolate the circumstances under which the observed errors take place.

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

NOVELTY (?)
3. Incremental improvement.

WRITING QUALITY (?)
5. Outstanding

QUESTIONS TO ADDRESS IN THE REBUTTAL
My primary concern with this paper is that it doesn't have (micro-)architectural content, and may not spur on future work.
Reviewer C -- Leave It to DRAM Vendors

Comments for authors
This is an extremely well-written analysis of DRAM behavior, and the authors are to be commended on establishing a robust and flexible characterization platform and methodology.

That being said, disturb errors have occurred repeatedly over the course of DRAM's history (which the authors do acknowledge). History has shown that particular disturbances, and in particular hammer errors, are short-lived, and are quickly solved by DRAM manufacturers. Historically, once these types of errors occur at a particular lithography node/DRAM density, they must be solved by the DRAM manufacturers, because even if a solution for a systemic problem could be asserted for particular markets (e.g., server, where use of advanced coding techniques, extra chips, etc. is acceptable), there will always be significant DRAM chip volume in single-piece applications (e.g., consumer devices, etc.) where complex architectural solutions aren't an option. The authors have identified a contemporary disturb sensitivity in DRAMs, but as non-technologists, our community can generally only observe, not correct, such problems.

Reviewer expertise (?)
4. Expert in area, with highest confidence in review.
Reviewer D -- Nothing New in RowHammer

**Review #66D** Modified Thursday 18 Jul 2013 12:51pm

**Overall merit (?)**

1. Reject

**Reviewer expertise (?)**

4. Expert in area, with highest confidence in review.

**Paper summary**

The authors demonstrate that repeated activate-precharge operations on one wordline of a DRAM can disturb a few cells on adjacent wordlines. They showed that such a behavior can be caused for most DRAMs and all DRAMs of recent manufacture they tested.

**Paper strengths**

DRAM errors are getting more likely with newer generations and it is necessary to investigate their cause and mitigation in computer systems, as such the paper addresses a subtopic of a relevant problem.

**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.

**Novelty (?)**

2. Insignificant novelty. Virtually all of the ideas are published or known.

**Writing quality (?)**

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

You are an author of this paper.

Abstract:
Memory isolation is a key property of a reliable and secure computing system --- an access to one memory address should not have unintended side effects on data stored in other.

+ Authors:
Y. Kim, R. Daly, J. Kim, J. Lee, C. Fallin, C. Wilkerson, O. Mutlu
[details]

+ Topics:

<table>
<thead>
<tr>
<th>Review</th>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevConAnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>#41A</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>#41B</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>#41C</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>#41D</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>#41E</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>#41F</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
**Review #41D**

**Reviewer D**

**OVERALL MERIT (?)**

2. Reject

**PAPER SUMMARY**

The authors
1) characterize disturbance error in commodity DRAM
2) identify the root cause such errors (but it's already a well know problem in DRAM community).
3) propose a simple architectural technique to mitigate such errors.

**PAPER STRENGTHS**

The authors demonstrated the problem using the real systems.

**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-qOSnBcdo](http://www.youtube.com/watch?v=i3-qOSnBcdo)

2) The architectural contribution of this study is too insignificant.
**Novelty (?)**

2. Insignificant novelty. Virtually all of the ideas are published or known.

**Writing Quality (?)**

5. Outstanding

**Reviewer Confidence and Expertise (?)**

4. Expert in area, with highest confidence in review.

**Questions for Authors**

1. There are other sources of disturbance errors. How can you guarantee the errors observed by you are not from such errors?

2. You did your best on explaining why we have much fewer 1->0 error but not quite satisfied. Any other explanation?

3. Can you elaborate why we have more disturbed cells over rounds while you claim that disturbed cells are not weak cells? I'm sure this is related to device again issues.

**Detailed Comments**

This is a well written and executed paper (in particular using real systems), but I have many concerns:

1) this is a well-known problem to the DRAM community (so no novelty there); in DRAM community people use
2) what you did to incur disturbance is is so called "row hammering" issues - please see
http://www.youtube.com/watch?v=i3-qQSnBcdo - a demonstration video for capturing this problem...

3) the relevance of this paper to ISCA. I feel that this paper (most part) is more appropriate to conferences like
International Test Conference (ITC) or VLSI Test Symposium or Dependable Systems and Networks (DSN)
at most. This is because the authors mainly dedicated the effort to the DRAM circuit characterization and test
method in my view while the architectural contribution is very weak - I'm not even sure this can be published to
these venues since it's a well known problem! I also assume techniques proposed to minimize disturbance
error in STT-RAM and other technology can be employed here as well.
Rebuttal to Reviewer D

1. As we acknowledge in the paper, it is true that different types of DRAM coupling phenomena have been known to the DRAM circuits/testing community. However, there is a clear distinction between circuits/testing techniques confined to the *foundry* versus characterization/solution of a problem out in the *field*. The three citations (from 10+ years ago) do *not* demonstrate that disturbance errors exist in DIMMs sold then or now. They do *not* provide any real data (only simulated ones), let alone a large-scale characterization across many DIMMs from multiple manufacturers. They do *not* construct an attack on real systems, and they do *not* provide any solutions. Finally, our paper *already* references all three citations, or their more relevant equivalents. (The second/third citations provided by the reviewer are on bitline-coupling, whereas we cite works from the same authors on wordline-coupling [2, 3, 37].)

2. We were aware of the video from Teledyne (a test equipment company) and have *already* referenced slides from the same company [36]. In terms of their content regarding "row hammer", the video and the slides are identical: all they mention is that "aggressive row activations can corrupt adjacent rows". (They then advertise how their test equipment is able to capture a timestamped DRAM access trace, which can then be post-processed to identify when the number of activations exceeds a user-set threshold.) Both the video and slides do *not* say that this is a real problem affecting DIMMs on the market now. They do *not* provide any quantitative data, *nor* real-system demonstration, *nor* solution.
Review #41E  Modified 7 Feb 2014 11:08:04pm CST A Plain text

Reviewer E

This paper studies the row disturbance problem in DRAMs. The paper includes a thorough quantitative characterization of the problem and a qualitative discussion of the source of the problem and potential solutions.

Paper Strengths
+ The paper provides a detailed quantitative characterization of the “row hammering” problem in memories.

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.

Novelty (?)
2. Insignificant novelty. Virtually all of the ideas are published or known.

Writing Quality (?)
3. Adequate

Reviewer Confidence and Expertise (?)
3. Knowledgeable in area, and significant confidence in the paper's evaluation.
but there are numerous mentions of hammering in the literature, and clearly industry has studied this problem for many years. In particular, Intel has a patent application on a memory controller technique that addresses this exact problem, with priority date June 2012:


The patent application details sound very similar to solution 6 in this paper, so a more thorough comparison with solution 7 seems mandatory.

My overall feeling is that while the reliability characterization is important and interesting, a better target audience for the characterization work would be in a testing/reliability venue. The most interesting part of this paper from the ISCA point of view are the proposed solutions, but all of these are discussed in a very qualitative manner. My preference would be to see a much shorter characterization section with a much stronger and quantitative evaluation and comparison of the proposed solutions.
*Nevertheless*, we were able to induce a large number of DRAM disturbance errors on all the latest Intel/AMD platforms that we tested: Haswell, Ivy Bridge, Sandy Bridge, and Piledriver. (At the time of submission, we had tested only Sandy Bridge.) Importantly, the patents do *not* provide quantitative characterization *nor* real-system demonstration.

[R1] "Row Hammer Refresh Command." US20140006703 A1
[R2] "Row Hammer Condition Monitoring." US20140006704 A1

After our paper was submitted, two patents that had been filed by Intel were made public (one is mentioned by the reviewer [R1]). Together, the two patents describe what we posed as the *sixth* potential solution in our paper (Section 8). Essentially, the memory controller maintains a table of counters to track the number of activations to recently activated rows [R2]. And if one of the counters exceeds a certain threshold, the memory controller notifies the DRAM chips using a special command [R1]. The DRAM chips would then refresh an entire "region" of rows that includes both the aggressor and its victim(s) [R1]. For the patent [R1] to work, DRAM manufacturers must cooperate and implement this special command. (It is a convenient way of circumventing the opacity in the logical-physical mapping. If implemented, the same command can also be used for our *seventh* solution.) The limitation of this *sixth* solution is the storage overhead of the counters and the extra power required to associatively search through them on every activation (Section 8). That is why we believe our *seventh* solution to be more attractive. We will cite the patents and include a more concrete comparison between the two solutions.
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
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
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; its 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/disk/s/…) faster; our CPUs/disk/s (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.
I poked around a bit and DRAM vendors have already solved this problem. DRAM row hammering appears to be a known problem.

**CHANCE OF IMPACT (??)**

3. Minor impact

**OVERALL MERIT (??)**

2. Weak reject (Happy to discuss but unlikely to be chosen.)

**COMMENTS FOR AUTHOR**

Interesting paper for those interested in DRAM issues. I wonder if it is possible to gain an insight into why this happens.

I seem to remember that, during the presentation at ISCA, it was pointed out that DRAM manufacturers have already fixed the problem. So where is the novelty and long term impact?
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 or writing papers
- **Be constructive**, not destructive
- **Enable heterogeneity, but do not** have double standards...

**Do not block or delay scientific progress for non-reasons**
Suggestion to Community

We Need to Fix the Reviewer Accountability Problem
Takeaway

Main Memory Needs
Intelligent Controllers
Takeaway

Research Community Needs

Accountable Reviewers
An Interview on Research and Education

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

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

SAFARI  
https://www.youtube.com/onurmutlulectures
Suggestion to Researchers: Principle: Passion

Follow Your Passion
(Do not get derailed by naysayers)
Suggestion to Researchers: Principle: Resilience

Be Resilient
Focus on learning and scholarship
Principle: Learning and Scholarship

The quality of your work defines your impact
Principle: Work Hard

Work Hard to Enable Your Passion
You can make a good impact on the world
Recommended Interview on Research & Education

- **Computing Research and Education (@ ISCA 2019)**
  - [https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz](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](https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15)

- **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"]
Recommended Interview

https://www.youtube.com/watch?v=8ffSEKZhmvo
A Talk on Impactful Research & Education

Applying to Grad School & Doing Impactful Research

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
13 June 2020
Undergraduate Architecture Mentoring Workshop @ ISCA 2021

Arch. Mentoring Workshop @ISCA’21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu
1,563 views • Premiered Jun 16, 2021

Panel talk at Undergraduate Architecture Mentoring Workshop at ISCA 2021
(https://sites.google.com/wisc.edu/uar...)
Richard Hamming
``You and Your Research''

Transcription of the
Bell Communications Research Colloquium Seminar
7 March 1986

Hidden Row Activation
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yaşlıkçı
Ataberk Olgun  Minesh Patel  Haocong Luo  Hasan Hassan
Lois Orosa  Oğuz Ergin  Onur Mutlu

SAFARI

ETH zürich  CESGA  TOBB ETÜ
Executive Summary

- **Problem**: Periodic and preventive refreshes cause increasingly significant performance degradation as DRAM chip density increases

- **Goal**: Reduce the performance overhead of periodic and preventive refreshes in off-the-shelf DRAM chips

- **Key Idea**: Refresh a DRAM row concurrently with refreshing or activating another row in the same bank, leveraging subarray-level parallelism

- **HiRA**: Hidden Row Activation
  - Concurrently opens two rows in electrically isolated subarrays in quick succession
  - Refreshes a DRAM row concurrently with refreshing or activating any of the 32% of the rows in the same bank
  - 51.4% reduction on the overall latency of refreshing two rows

- **HiRA-MC**: Buffers refresh requests for a time slack to leverage the parallelism HiRA provides
  - 12.6% speedup by reducing periodic refresh’s performance overheads
  - 3.73x speedup by reducing preventive refresh’s performance overheads
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yaşlıkçı¹ Ataberk Olgun¹ Minesh Patel¹ Haocong Luo¹ Hasan Hassan¹
Lois Orosa¹,³ Oğuz Ergin² Onur Mutlu¹
¹ETH Zürich ²TOBB University of Economics and Technology ³Galicia Supercomputing Center (CESGA)

DRAM is the building block of modern main memory systems. DRAM cells must be periodically refreshed to prevent data loss. Refresh operations degrade system performance by interfering with memory accesses. As DRAM chip density increases with technology node scaling, refresh operations also increase because: 1) the number of DRAM rows in a chip increases; and 2) DRAM cells need additional refresh operations to mitigate bit failures caused by RowHammer, a failure mechanism that becomes worse with technology node scaling. Thus, it is critical to enable refresh operations at low performance overhead. To this end, we propose a new operation, Hidden Row Activation (HiRA), and the HiRA Memory Controller (HiRA-MC) to perform HiRA operations.

As DRAM density increases with technology node scaling, the performance overhead of refresh also increases due to three major reasons. First, as the DRAM chip density increases, more DRAM rows need to be periodically refreshed in a DRAM chip [55, 57–61]. Second, as DRAM technology node scales down, DRAM cells become smaller and thus can store less amount of charge, requiring them to be refreshed more frequently [10, 20, 67, 102, 103, 118, 122–124]. Third, with increasing DRAM density, DRAM cells are placed closer to each other, exacerbating charge leakage via a disturbance error mechanism called RowHammer [79, 84, 119, 120, 133, 134, 167, 180, 183], and thus requiring additional refresh operations (called preventive refreshes) to avoid data corruption due to RowHammer [2, 3, 5–7, 29, 33, 42, 63, 66, 76, 82, 84, 97, 98, 107, 135, 141].
HiRA: Hidden Row Activation
for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yağlıkçı
Ataberk Olgun    Minesh Patel    Haocong Luo    Hasan Hassan
Lois Orosa    Oğuz Ergin    Onur Mutlu

SAFARI
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 → Charge leakage

• **Cause 2: Conductive bridges**

• **Cause 3: Hot-carrier injection**

*Confirmed by at least one manufacturer*
Revisiting RowHammer
Revisiting RowHammer in 2020

- 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 Yağılkıçı§
Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§†

§ETH Zürich †Carnegie Mellon University
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
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

SAFARI
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>
<th></th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-old</td>
<td>56 (10)</td>
<td>88 (11)</td>
<td>28 (7)</td>
<td>172 (28)</td>
<td></td>
</tr>
<tr>
<td>DDR3-new</td>
<td>80 (10)</td>
<td>52 (9)</td>
<td>104 (13)</td>
<td>236 (32)</td>
<td></td>
</tr>
<tr>
<td>DDR4-old</td>
<td>112 (16)</td>
<td>24 (3)</td>
<td>128 (18)</td>
<td>264 (37)</td>
<td></td>
</tr>
<tr>
<td>DDR4-new</td>
<td>264 (43)</td>
<td>16 (2)</td>
<td>108 (28)</td>
<td>388 (73)</td>
<td></td>
</tr>
<tr>
<td>LPDDR4-1x</td>
<td>12 (3)</td>
<td>180 (45)</td>
<td>N/A</td>
<td>192 (48)</td>
<td></td>
</tr>
<tr>
<td>LPDDR4-1y</td>
<td>184 (46)</td>
<td>N/A</td>
<td>144 (36)</td>
<td>328 (82)</td>
<td></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></td>
</tr>
<tr>
<td>Row 5</td>
<td></td>
</tr>
</tbody>
</table>

**DRAM_RowHammer_Characterization()**:

```plaintext
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 → 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

### DRAM_RowHammer_Characterization()

```python
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 → 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**
- 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?

Hammer Count = 2 Accesses, one to each adjacent row of victim

SAFARI
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.

Chips of newer DRAM technology nodes can exhibit RowHammer bit flips 1) in **more rows** and 2) **farther away** from the victim row.
4. Spatial Effects: Row Distance

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

[More analysis in the paper]
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.

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 RowHammer Bit Flips per Chip

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

![Box plot diagram showing Hammer Count needed for the first bit flip ($HC_{\text{first}}$) for different DDR types.]

- Whisker
- Q3: 75% point
- Median: 50%
- Q1: 25% point
- Whisker
5. First RowHammer Bit Flips per Chip

What is the minimum Hammer Count required to cause bit flips ($H_{C_{\text{first}}}$)?

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, $H_{C\text{first}}$ 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.

Newer chips from a given DRAM manufacturer more vulnerable to RowHammer.
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 ($10 < \text{MPKI} < 740$)

• **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 $HC_{first}$)
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)

![Graph showing normalized system performance vs. HC<sub>first</sub> (number of hammers required to induce first RowHammer bit flip). The graph is divided into two regions: Low Performance Overhead and High Performance Overhead. In the Low Performance Overhead region, the normalized system performance remains above 80% for HC<sub>first</sub> values up to 10<sup>5</sup>. In the High Performance Overhead region, the performance drops significantly, indicating an 80% performance loss. The PARA line is depicted, showing its effectiveness in mitigating RowHammer.]
Mitigation Mechanism Evaluation (ProHIT)

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

Normalized System Performance (%) vs. $HC_{\text{first}}$
Mitigation Mechanism Evaluation (MRLoc)

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

The graph shows the normalized system performance (%) as a function of $HC_{\text{first}}$ (number of hammers required to induce first RowHammer bit flip). TWiCe is shown in green, supported, and TWiCe-ideal in red, not supported. The graph indicates that TWiCe does not support $HC_{\text{first}} < 32k$.

We evaluate an ideal scalable version (TWiCe-ideal) assuming it solves two critical design issues.

SAFARI
Mitigation Mechanism Evaluation (Ideal)

**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 $H_{C_{\text{first}}}$ values but has very low normalized system performance
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

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$^\dagger$, Minesh Patel$,^\dagger$ A. Giray Yaşlıkçı$,^\dagger$
Hasan Hassan$,^\dagger$ Roknoddin Azizi$,^\dagger$ Lois Orosa$,^\dagger$ Onur Mutlu$,^\dagger$

$^\dagger$ETH Zürich $^\dagger$Carnegie Mellon University
TRRespas
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)]
  [Lecture Slides (pptx) (pdf)]
  [Talk Video (17 minutes)]
  [Lecture Video (59 minutes)]
  [Source Code]
  [Web Article]
  Best paper award. Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

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*  Kaveh Razavi*

*Vrije Universiteit Amsterdam  §ETH Zürich  ¶Qualcomm Technologies Inc.
TRRespass

- First work to show that TRR-protected DRAM chips are vulnerable to RowHammer in the field
  - Mitigations advertised as secure are not secure

- Introduces the Many-sided RowHammer attack
  - Idea: Hammmer many rows to bypass TRR mitigations (e.g., by overflowing proprietary TRR tables that detect aggressor rows)

- (Partially) reverse-engineers the TRR and pTRR mitigation mechanisms implemented in DRAM chips and memory controllers

- Provides an automatic tool that can effectively create many-sided RowHammer attacks in DDR4 and LPDDR4(X) chips
Example Many-Sided Hammering Patterns

(a) Assisted double-sided    (b) 4-sided

Fig. 12: Hammering patterns discovered by TRRespass. Aggressor rows are in red (■) and victim rows are in blue (□).
Target Row Refresh (TRR)

• How does it work?

  1. *Track* activation count of each DRAM row

  2. *Refresh* neighbor rows if row activation count exceeds a threshold

• Many possible implementations in practice

• Security through obscurity

• In-DRAM TRR

• Embedded in the DRAM circuitry, i.e., not exposed to the memory controller
Timeline of TRR Implementations

- **pTRR DDR3**: Intel reports pTRR on DDR3 server systems.
- **In-DRAM TRR**: Earliest manufacturing date of RH-free DRAM modules.
- **pTRR DDR4**: First DDR4 generation is pTRR protected.

- **'12**: pTRR DDR3
- **'13**: Intel reports pTRR on DDR3 server systems
- **'14**: In-DRAM TRR
- **'15**: Earliest manufacturing date of RH-free DRAM modules
- **'16**: pTRR DDR4
- **'17**: First DDR4 generation is pTRR protected
- **'18**
- **'19**: Last generation DIMMs we focus on
Our Goals

• Reverse engineer in-DRAM TRR to demystify how it works

• Bypass TRR protection
  • A Novel hammering pattern: The Many-sided RowHammer
  • Hammering up to 20 aggressor rows allows bypassing TRR

• Automatically test memory devices: TRRespass
  • Automate hammering pattern generation
Infrastructures to Understand Such Issues

SoftMC: Open Source DRAM Infrastructure


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

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 \quad \textsuperscript{2}TOBB University of Economics & Technology \quad \textsuperscript{3}Carnegie Mellon University
\textsuperscript{4}University of Virginia \quad \textsuperscript{5}Microsoft Research \quad \textsuperscript{6}NVIDIA Research
Components of In-DRAM TRR

- **Sampler**
  - Tracks aggressor rows activations
  - Design options:
    - Frequency based (record every N\textsuperscript{th} row activation)
    - Time based (record first N row activations)
    - Random seed (record based on a coin flip)
  - Regardless, the sampler has a limited size

- **Inhibitor**
  - Prevents bit flips by refreshing victim rows
    - The latency of performing victim row refreshes is squeezed into slack time available in $t_{RFC}$ (i.e., the latency of regular Refresh command)
Case Study: Vendor C

How big is the sampler?
- Pick \( N \) aggressor rows
- Perform a series of hammers (i.e., activations of aggressors)
  - 8K activations
- After each series of hammers, issue \( R \) refreshes
- 10 Rounds
Case Study: Vendor C
Case Study: Vendor C
Case Study: Vendor C

1. The TRR mitigation acts on a refresh command
Case Study: Vendor C
2. The mitigation can sample more than one aggressor per refresh interval.
3. The mitigation can refresh only a single victim within a refresh operation.
Case Study: Vendor C

#Corruptions

![Heatmap showing #Corruptions](chart.png)
4. Sweeping the number of refresh operations and aggressor rows while hammering reveals the sampler size
Many-Sided Hammering

![Bit Flips vs. Refreshes Diagram](image)

**Fig. 9: Refreshes vs. Bit Flips.** Module $C_{12}$: Number of bit flips detected when sending $r$ refresh commands to the module. We report this for different number of aggressor rows ($n$). For example, when hammering 5 rows, followed by sending 2 refreshes, we find 1,710 bit flips. This figure shows that the number of bit flips stabilizes for $r \geq 4$, implying that the size of the sampler may be 4.
Some Observations

**Observation 1:** The TRR mitigation acts (i.e., carries out a targeted refresh) on every refresh command.

**Observation 2:** The mitigation can sample more than one aggressor per refresh interval.

**Observation 3:** The mitigation can refresh only a single victim within a refresh operation (i.e., time $t_{RFC}$).

**Observation 4:** Sweeping the number of refresh operations and aggressor rows while hammering reveals the sampler size.

![Diagram](image)

(a) Assisted double-sided  (b) 4-sided

**Fig. 12:** Hammering patterns discovered by TRRespass. Aggressor rows are in red (■) and victim rows are in blue (■).
Case Study: Vendor C

Hammering using the default refresh rate

\[ t_{REFI} = 7.8 \mu s \]
BitFlips vs. Number of Aggressor Rows

**Fig. 10:** Bit flips vs. number of aggressor rows. Module $C_{12}$: Number of bit flips in bank 0 as we vary the number of aggressor rows. Using SoftMC, we refresh DRAM with standard tREFI and run the tests until each aggressor rows is hammered 500K times.

**Fig. 11:** Bit flips vs. number of aggressor rows. Module $A_{15}$: Number of bit flips in bank 0 as we vary the number of aggressor rows. Using SoftMC, we refresh DRAM with standard tREFI and run the tests until each aggressor rows is hammered 500K times.

**Fig. 13:** Bit flips vs. number of aggressor rows. Module $A_{10}$: Number of bit flips triggered with $N$-sided RowHammer for varying number of $N$ on Intel Core i7-7700K. Each aggressor row is one row away from the closest aggressor row (i.e., VAVAVA... configuration) and aggressor rows are hammered in a round-robin fashion.
TABLE II: TRRespass results. We report the number of patterns found and bit flips detected for the 42 DRAM modules in our set.

<table>
<thead>
<tr>
<th>Module</th>
<th>Date (yy-ww)</th>
<th>Freq. (MHz)</th>
<th>Size (GB)</th>
<th>Organization</th>
<th>MAC</th>
<th>Found Patterns</th>
<th>Best Pattern</th>
<th>Corruptions</th>
<th>Double Refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ranks</td>
<td>Banks</td>
<td>Pins</td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>$A_{0,1,2,3}$</td>
<td>16-37</td>
<td>2132</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$A_4$</td>
<td>16-51</td>
<td>2132</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>4</td>
<td>9-sided</td>
</tr>
<tr>
<td>$A_5$</td>
<td>18-51</td>
<td>2400</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>$\times 16$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$A_{6,7}$</td>
<td>18-15</td>
<td>2666</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>$\times 16$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$A_8$</td>
<td>17-09</td>
<td>2400</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>33</td>
<td>19-sided</td>
</tr>
<tr>
<td>$A_9$</td>
<td>17-31</td>
<td>2400</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>33</td>
<td>19-sided</td>
</tr>
<tr>
<td>$A_{10}$</td>
<td>19-02</td>
<td>2400</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>488</td>
<td>10-sided</td>
</tr>
<tr>
<td>$A_{11}$</td>
<td>19-02</td>
<td>2400</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>523</td>
<td>10-sided</td>
</tr>
<tr>
<td>$A_{12,13}$</td>
<td>18-50</td>
<td>2666</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$A_{14}$</td>
<td>19-08†</td>
<td>3200</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>120</td>
<td>14-sided</td>
</tr>
<tr>
<td>$A_{15}$‡</td>
<td>17-08‡</td>
<td>2132</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>2</td>
<td>9-sided</td>
</tr>
<tr>
<td>$B_0$</td>
<td>18-11</td>
<td>2666</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
</tr>
<tr>
<td>$B_1$</td>
<td>18-11</td>
<td>2666</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
</tr>
<tr>
<td>$B_2$</td>
<td>18-49</td>
<td>3000</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
</tr>
<tr>
<td>$B_3$</td>
<td>19-08†</td>
<td>3000</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_{4,5}$</td>
<td>19-08†</td>
<td>2666</td>
<td>8</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_{6,7}$</td>
<td>19-08†</td>
<td>2400</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_8$</td>
<td>19-08†</td>
<td>2400</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_9$</td>
<td>19-08†</td>
<td>2400</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>2</td>
<td>3-sided</td>
</tr>
<tr>
<td>$B_{10,11}$</td>
<td>16-13†</td>
<td>2132</td>
<td>8</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{0,1}$</td>
<td>18-46</td>
<td>2666</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{2,3}$</td>
<td>19-08†</td>
<td>2800</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{4,5}$</td>
<td>19-08†</td>
<td>3000</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{6,7}$</td>
<td>19-08†</td>
<td>3000</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_9$</td>
<td>19-08†</td>
<td>3200</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_9$</td>
<td>18-47</td>
<td>2666</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{10,11}$</td>
<td>19-04</td>
<td>2933</td>
<td>8</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UL</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{12}$‡</td>
<td>15-01‡</td>
<td>2132</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UT</td>
<td>25</td>
<td>10-sided</td>
</tr>
<tr>
<td>$C_{13}$‡</td>
<td>18-49‡</td>
<td>2132</td>
<td>4</td>
<td>1</td>
<td>16</td>
<td>$\times 8$</td>
<td>UT</td>
<td>3</td>
<td>9-sided</td>
</tr>
</tbody>
</table>

† The module does not report manufacturing date. Therefore, we report purchase date as an approximation.
‡ Analyzed using the FPGA-based SoftMC.

UL = Unlimited
UT = Untested
The system runs with double refresh frequency in standard conditions. We configured the refresh interval to be 64 ms in the BIOS settings.
TABLE III: LPDDR4(X) results. Mobile phones tested against TRRespass on ARMv8 sorted by production date. We found bit flip inducing RowHammer patterns on 5 out of 13 mobile phones.

<table>
<thead>
<tr>
<th>Mobile Phone</th>
<th>Year</th>
<th>SoC</th>
<th>Memory (GB)</th>
<th>Found Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Pixel</td>
<td>2016</td>
<td>MSM8996</td>
<td>4†</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 2</td>
<td>2017</td>
<td>MSM8998</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Samsung G960F/DS</td>
<td>2018</td>
<td>Exynos 9810</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Huawei P20 DS</td>
<td>2018</td>
<td>Kirin 970</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>Sony XZ3</td>
<td>2018</td>
<td>SDM845</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>HTC U12+</td>
<td>2018</td>
<td>SDM845</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>LG G7 ThinQ</td>
<td>2018</td>
<td>SDM845</td>
<td>4†</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 3</td>
<td>2018</td>
<td>SDM845</td>
<td>4</td>
<td>✓</td>
</tr>
<tr>
<td>Google Pixel 4</td>
<td>2019</td>
<td>SM8150</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>OnePlus 7</td>
<td>2019</td>
<td>SM8150</td>
<td>8</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung G970F/DS</td>
<td>2019</td>
<td>Exynos 9820</td>
<td>6†</td>
<td>✓</td>
</tr>
<tr>
<td>Huawei P30 DS</td>
<td>2019</td>
<td>Kirin 980</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>Xiaomi Redmi Note 8 Pro</td>
<td>2019</td>
<td>Helio G90T</td>
<td>6</td>
<td>—</td>
</tr>
</tbody>
</table>

† LPDDR4 (not LPDDR4X)
TABLE IV: Time to exploit. Time to find the first exploitable template on two sample modules from each DRAM vendor.

<table>
<thead>
<tr>
<th>Module</th>
<th>$\tau$ (ms)</th>
<th>PTE [81]</th>
<th>RSA-2048 [79]</th>
<th>sudo [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{14}$</td>
<td>188.7</td>
<td>4.9s</td>
<td>6m 27s</td>
<td>—</td>
</tr>
<tr>
<td>$A_{4}$</td>
<td>180.8</td>
<td>38.8s</td>
<td>39m 28s</td>
<td>—</td>
</tr>
<tr>
<td>$B_{1}$</td>
<td>360.7</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$B_{2}$</td>
<td>331.2</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$C_{12}$</td>
<td>300.0</td>
<td>2.3s</td>
<td>74.6s</td>
<td>54m 16s</td>
</tr>
<tr>
<td>$C_{13}$</td>
<td>180.9</td>
<td>3h 15m</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

$\tau$: Time to template a single row: time to fill the victim and aggressor rows + hammer time + time to scan the row.
TRRespass Key Results

- 13 out of 42 tested DDR4 DRAM modules are vulnerable
  - From all 3 major manufacturers
  - 3-, 9-, 10-, 14-, 19-sided hammer attacks needed

- 5 out of 13 mobile phones tested vulnerable
  - From 4 major manufacturers
  - With LPDDR4(X) DRAM chips

- These results are scratching the surface
  - TRRespass tool is not exhaustive
  - There is a lot of room for uncovering more vulnerable chips and phones
RowHammer is still an open problem

Security by obscurity is likely not a good solution
More on TRRespass

- 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)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Lecture Video (59 minutes)]
[Source Code]
[Web Article]

Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

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*  Kaveh Razavi*

*Vrije Universiteit Amsterdam  §ETH Zürich  ¶Qualcomm Technologies Inc.
BlockHammer Solution
BlockHammer Solution in 2021

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Intel Hardware Security Academic Awards Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]
[Intel Hardware Security Academic Awards Short Talk Video (2 minutes)]
[BlockHammer Source Code]

Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations)

---

BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

A. Giray Yağlıkçı¹  Minesh Patel¹  Jeremie S. Kim¹  Roknoddin Azizi¹  Ataberk Olgun¹  Lois Orosa¹  Hasan Hassan¹  Jisung Park¹  Konstantinos Kanellopoulos¹  Taha Shahroodi¹  Saugata Ghose²  Onur Mutlu¹

¹ETH Zürich ²University of Illinois at Urbana–Champaign
BlockHammer
Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

Abdullah Giray Yağlıkçı
Minesh Patel  Jeremie S. Kim  Roknoddin Azizi
Ataberk Olgun  Lois Orosa  Hasan Hassan  Jisung Park
Konstantinos Kanellopoulos  Taha Shahroodi
Saugata Ghose*  Onur Mutlu

SAFARI

ETH Zürich
Executive Summary

- **Motivation**: RowHammer is a worsening DRAM reliability and security problem

- **Problem**: Mitigation mechanisms have limited support for current/future chips
  - **Scalability** with worsening RowHammer vulnerability
  - **Compatibility** with commodity DRAM chips

- **Goal**: Efficiently and **scalably** prevent RowHammer bit-flips without knowledge of or modifications to DRAM internals

- **Key Idea**: Selectively throttle memory accesses that may cause RowHammer bit-flips

- **Mechanism**: BlockHammer
  - Tracks activation rates of all rows by using area-efficient Bloom filters
  - Throttles row activations that could cause RowHammer bit flips
  - Identifies and throttles threads that perform RowHammer attacks

- **Scalability with Worsening RowHammer Vulnerability**: Competeive with state-of-the-art mechanisms when there is no attack
  - Superior performance and DRAM energy when a RowHammer attack is present

- **Compatibility with Commodity DRAM Chips**:
  - No proprietary information of DRAM internals
  - No modifications to DRAM circuitry
# Outline

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM and RowHammer Background</td>
<td></td>
</tr>
<tr>
<td>Motivation and Goal</td>
<td></td>
</tr>
<tr>
<td>BlockHammer</td>
<td></td>
</tr>
<tr>
<td>RowBlocker</td>
<td></td>
</tr>
<tr>
<td>AttackThrottler</td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td></td>
</tr>
<tr>
<td>Conclusion</td>
<td></td>
</tr>
</tbody>
</table>
# Outline

<table>
<thead>
<tr>
<th>DRAM and RowHammer Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation and Goal</td>
</tr>
<tr>
<td>BlockHammer</td>
</tr>
<tr>
<td>RowBlocker</td>
</tr>
<tr>
<td>AttackThrottler</td>
</tr>
<tr>
<td>Evaluation</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Organizing and Accessing DRAM Cells

A DRAM cell consists of a capacitor and an access transistor.

A row needs to be activated to access its content.
DRAM Refresh

Periodic refresh operations preserve stored data

[Patel+ ISCA’17, Kim+ ISCA’20]
The RowHammer Phenomenon

Repeatedly opening (activating) and closing (precharging) a DRAM row causes RowHammer bit flips in nearby cells.
# Outline

- DRAM and RowHammer Background
- Motivation and Goal
- BlockHammer
- RowBlocker
- AttackThrottler
- Evaluation
- Conclusion
RowHammer Mitigation Approaches

• Increased refresh rate

- REF-to-REF time reduces
- Fewer activations can fit

• Physical isolation

- Large-enough distance

• Reactive refresh

- Refresh
- Rapidly activated (hammered)

• Proactive throttling

SAFARI

- Fewer activations can be performed
Two Key Challenges

1. **Scalability**
   with worsening RowHammer vulnerability

2. **Compatibility**
   with commodity DRAM chips
Scalability with Worsening RowHammer Vulnerability

- DRAM chips are more vulnerable to RowHammer today
- RowHammer bit-flips occur at much lower activation counts (more than an order of magnitude decrease):
  - 139.2K [Y. Kim+, ISCA 2014]
  - 9.6K [J. S. Kim+, ISCA 2020]

- RowHammer blast radius has increased by 33%:
  - 9 rows [Y. Kim+, ISCA 2014]
  - 12 rows [J. S. Kim+, ISCA 2020]

- In-DRAM mitigation mechanisms are ineffective [Frigo+, S&P 2020]

RowHammer is a more serious problem than ever
Mitigation Approaches with Worsening RowHammer Vulnerability

- **Increased refresh rate**
  - REF-to-REF time further reduces
  - Even fewer activations can fit

- **Physical isolation**
  - Larger distance
  - More isolation rows

- **Reactive refresh**
  - Refresh more frequently
  - Refresh more rows

- **Proactive throttling**
  - More aggressively throttles row activations
Mitigation Approaches with Worsening RowHammer Vulnerability

- Increased refresh rate
  - REF-to-REF time further reduces
  - Even fewer activations can fit

- Physical isolation
  - Mitigation mechanisms face the challenge of scalability with worsening RowHammer

- Reactive refresh
  - Refresh more frequently
  - Refresh more rows

- Proactive throttling
  - More aggressively throttles row activations

SAFARI
Two Key Challenges

1. **Scalability**
   with worsening RowHammer vulnerability

2. **Compatibility**
   with commodity DRAM chips
Compatibility with Commodity DRAM Chips

- **Application Level**
  - Virtual Memory Address

- **System Level**
  - Physical Memory Address
  - **Memory Controller**
    - DRAM Bus Addresses
      - (Channel, Rank, Bank Group, Bank, Row, Col)

- **DRAM Chip**
  - In-DRAM Mapping
  - Physical Rows and Columns
Compatibility
with Commodity DRAM Chips

Vendors apply in-DRAM mapping for two reasons:

• **Design Optimizations:** By simplifying DRAM circuitry to provide better density, performance, and power

• **Yield Improvement:** By mapping faulty rows and columns to redundant ones

• In-DRAM mapping scheme includes insights into **chip design** and **manufacturing quality**

In-DRAM mapping is proprietary information
RowHammer Mitigation Approaches

- Increased refresh rate
  - REF-to-REF time reduces
  - Fewer activations can fit

- Physical isolation
  - Identifying victim and isolation rows requires proprietary knowledge of in-DRAM mapping

- Reactive refresh
Our Goal

To prevent RowHammer *efficiently and scalably* *without* knowledge of or modifications to DRAM internals
Outline

DRAM and RowHammer Background
Motivation and Goal
BlockHammer
  RowBlocker
  AttackThrottler
Evaluation
Conclusion
BlockHammer

Key Idea

Selectively throttle memory accesses that may cause RowHammer bit-flips
BlockHammer
Overview of Approach

**RowBlocker**
Tracks row activation rates using area-efficient Bloom filters
Blacklists rows that are activated at a high rate
Throttles activations targeting a blacklisted row

No row can be activated at a high enough rate to induce bit-flips

**AttackThrottler**
Identifies threads that perform a RowHammer attack
Reduces memory bandwidth usage of identified threads

Greatly reduces the performance degradation and energy wastage a RowHammer attack inflicts on a system
## Outline

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM and RowHammer Background</td>
</tr>
<tr>
<td>Motivation and Goal</td>
</tr>
<tr>
<td>BlockHammer</td>
</tr>
<tr>
<td>RowBlocker</td>
</tr>
<tr>
<td>AttackThrottler</td>
</tr>
<tr>
<td>Evaluation</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
RowBlocker

- Modifies the memory request scheduler to throttle row activations
- **Blacklists** rows with a high activation rate and **delays** subsequent activations targeting blacklisted rows
RowBlocker

- Blocks a row activation if the row is both blacklisted and recently activated
RowBlocker

- When a row activation is performed, both RowBlocker-BL and RowBlocker-HB are updated with the row activation information.
RowBlocker-BL
Blacklisting Logic

• **Blacklists** a row when the row’s activation count in a time window exceeds a threshold

• Employs **two counting Bloom filters** for area-efficient activation rate tracking
Counting Bloom Filters

• Blacklisting logic counts activations using counting Bloom filters
• A row’s activation count
  - can be observed more than it is (false positive)
  - cannot be observed less than it is (no false negative)
• To avoid saturating counters, we use a time-interleaving approach
RowBlocker-BL
Blacklisting Logic

- Blacklisting logic employs **two counting Bloom filters**
- A new row activation is **inserted in both filters**
- Only one filter (**active filter**) responds to test queries
- The active filter changes at every epoch
RowBlocker-BL
Blacklisting Logic

- Blacklisting logic employs two counting Bloom filters
- A new row activation is inserted in both filters
- Only one filter (active filter) responds to test queries
- The active filter changes at every epoch
- Blacklists a row if its activation count reaches the blacklisting threshold ($N_{BL}$)

Assume that the row is activated at a high rate
Assume that the row is not activated at a high rate
Limiting the Row Activation Rate

- The activation rate is **RowHammer-safe** if it is smaller than or equal to **RowHammer threshold** \( N_{RH} \) activations in a refresh window \( t_{REFW} \)

- RowBlocker limits the **activation count** \( N_{CBF} \) in a CBF’s lifetime \( t_{CBF} \)

Activations Rate in a \( t_{CBF} \leq N_{RH} \) activations in a refresh window \( t_{REFW} \)
Limiting the Row Activation Rate

- The activation rate is RowHammer-safe if it is smaller than or equal to RowHammer threshold ($N_{RH}$) activations in a refresh window ($t_{REFW}$).
- RowBlocker limits the activation count ($N_{CBF}$) in a CBF’s lifetime ($t_{CBF}$).

Activation Rate in a $t_{CBF} \leq N_{RH}$ activations in a refresh window ($t_{REFW}$).

RowHammer Safety Constraint

$$\frac{N_{CBF}}{t_{CBF}} \leq \frac{N_{RH}}{t_{REFW}}$$

SAFARI
RowBlocker-HB
Limiting the Row Activation Rate

- Ensures that all rows experience a RowHammer-safe activation rate

\[
\frac{N_{CBF}}{t_{CBF}} \leq \frac{N_{RH}}{t_{REFW}}
\]

- We limit \( N_{CBF} \) by configuring \( t_{Delay} \):

\[
N_{CBF} \leq N_{BL} + \frac{t_{CBF} - (t_{RC} \times N_{BL})}{t_{Delay}}
\]
RowBlocker-HB

Delaying Row Activations

• RowBlocker-HB ensures no subsequent blacklisted row activation is performed sooner.

• RowBlocker-HB implements a history buffer for row activations that can fit in a $t_{\text{Delay}}$ time window.

• A blacklisted row activation is blocked as long as a valid activation record of the row exists in the history buffer.

No row can be activated at a high enough rate to induce bit-flips.
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM and RowHammer Background</td>
</tr>
<tr>
<td>Motivation and Goal</td>
</tr>
<tr>
<td>BlockHammer</td>
</tr>
<tr>
<td>RowBlocker</td>
</tr>
<tr>
<td>AttackThrottler</td>
</tr>
<tr>
<td>Evaluation</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
AttackThrottler

- Tackles a RowHammer attack’s **performance degradation** and **energy wastage** on a system

- A RowHammer attack intrinsically keeps activating **blacklisted rows**

- **RowHammer Likelihood Index (RHLI):** Number of activations that target blacklisted rows (normalized to maximum possible activation count)

  - **RHLI is larger** when the thread’s access pattern is more **similar to a RowHammer attack**
**AttackThrottler**

- Applies a smaller quota to a thread’s in-flight request count as RHLI increases.
  
  
<table>
<thead>
<tr>
<th>RHLI</th>
<th>Benign application</th>
<th>RowHammer attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>No blacklisted row activations</td>
<td>Blacklisted row activation count approaches RowHammer threshold</td>
</tr>
<tr>
<td>1.0</td>
<td><em>No quota applied</em></td>
<td><em>No request is allowed</em></td>
</tr>
</tbody>
</table>

- Reduces a RowHammer attack’s memory bandwidth consumption, enabling a larger memory bandwidth for concurrent benign applications.

  *Greatly reduces the performance degradation and energy wastage a RowHammer attack inflicts on a system.*

- RHLI can also be used as a RowHammer attack indicator by the system software.
Outline

DRAM and RowHammer Background
Motivation and Goal
BlockHammer
RowBlocker
AttackThrottler
Evaluation
Conclusion
Evaluation
BlockHammer’s Hardware Complexity

• We analyze six state-of-the-art mechanisms and BlockHammer
• We calculate area, access energy, and static power consumption*

<table>
<thead>
<tr>
<th>Mitigation Mechanism</th>
<th>SRAM KB</th>
<th>CAM KB</th>
<th>Area mm²</th>
<th>%CPU</th>
<th>Access Energy pJ</th>
<th>Static Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockHammer</td>
<td>51.48</td>
<td>1.73</td>
<td>0.14</td>
<td>0.06</td>
<td>20.30</td>
<td>22.27</td>
</tr>
<tr>
<td>PARA [73]</td>
<td>-</td>
<td>-</td>
<td>&lt;0.01</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ProHIT [137]</td>
<td>-</td>
<td>0.22</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>3.67</td>
<td>0.14</td>
</tr>
<tr>
<td>MRLoc [161]</td>
<td>-</td>
<td>0.47</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>4.44</td>
<td>0.21</td>
</tr>
<tr>
<td>CBT [132]</td>
<td>16.00</td>
<td>8.50</td>
<td>0.20</td>
<td>0.08</td>
<td>9.13</td>
<td>35.55</td>
</tr>
<tr>
<td>TWiCe [84]</td>
<td>23.10</td>
<td>14.02</td>
<td>0.15</td>
<td>0.06</td>
<td>7.99</td>
<td>21.28</td>
</tr>
<tr>
<td>Graphene [113]</td>
<td>-</td>
<td>5.22</td>
<td>0.04</td>
<td>0.02</td>
<td>40.67</td>
<td>3.11</td>
</tr>
</tbody>
</table>

BlockHammer is low cost and competitive with state-of-the-art mechanisms

*Assuming a high-end 28-core Intel Xeon processor system with 4-channel single-rank DDR4 DIMMs with a RowHammer threshold (NRH) of 32K
## Evaluation
### BlockHammer’s Hardware Complexity

<table>
<thead>
<tr>
<th>Mitigation Mechanism</th>
<th>SRAM KB</th>
<th>CAM KB</th>
<th>Area mm²</th>
<th>%CPU</th>
<th>Access Energy pJ</th>
<th>Static Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockHammer</td>
<td>51.48</td>
<td>1.73</td>
<td>0.14</td>
<td>0.6</td>
<td>20.30</td>
<td>22.27</td>
</tr>
<tr>
<td>PARA [73]</td>
<td>-</td>
<td>-</td>
<td>&lt;0.01</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ProHIT [137]</td>
<td>-</td>
<td>0.22</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MRLoc [161]</td>
<td>-</td>
<td>0.47</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBT [132]</td>
<td>16.00</td>
<td>8.50</td>
<td>0.20</td>
<td>0.08</td>
<td>3.67</td>
<td>4.4</td>
</tr>
<tr>
<td>TWiCe [84]</td>
<td>23.10</td>
<td>14.02</td>
<td>0.15</td>
<td>0.06</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>Graphene [113]</td>
<td>-</td>
<td>5.22</td>
<td>0.04</td>
<td>0.02</td>
<td>4.44</td>
<td>0.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mitigation Mechanism</th>
<th>SRAM KB</th>
<th>CAM KB</th>
<th>Area mm²</th>
<th>%CPU</th>
<th>Access Energy pJ</th>
<th>Static Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockHammer</td>
<td>441.33</td>
<td>55.58</td>
<td>1.57</td>
<td>0.64</td>
<td>99.64</td>
<td>220.99</td>
</tr>
<tr>
<td>PARA [73]</td>
<td>-</td>
<td>-</td>
<td>&lt;0.01</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ProHIT [137]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MRLoc [161]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CBT [132]</td>
<td>512.00</td>
<td>272.00</td>
<td>3.95</td>
<td>1.60</td>
<td>127.93</td>
<td>535.50</td>
</tr>
<tr>
<td>TWiCe [84]</td>
<td>738.32</td>
<td>448.27</td>
<td>5.17</td>
<td>2.10</td>
<td>124.79</td>
<td>631.98</td>
</tr>
<tr>
<td>Graphene [113]</td>
<td>-</td>
<td>166.03</td>
<td>1.14</td>
<td>0.46</td>
<td>917.55</td>
<td>93.96</td>
</tr>
</tbody>
</table>

BlockHammer’s hardware complexity **scales more efficiently** than state-of-the-art mechanisms.
Evaluation
Performance and DRAM Energy

- Cycle-level simulations using Ramulator and DRAMPower

- System Configuration:
  - Processor: 3.2 GHz, \{1,8\} core, 4-wide issue, 128-entry instr. window
  - LLC: 64-byte cacheline, 8-way set-associative, \{2,16\} MB
  - Memory scheduler: FR-FCFS
  - Address mapping: Minimalistic Open Pages
  - DRAM: DDR4 1 channel, 1 rank, 4 bank group, 4 banks per bank group
  - RowHammer Threshold: 32K

- Single-Core Benign Workloads:
  - 22 SPEC CPU 2006
  - 4 YCSB Disk I/O
  - 2 Network Accelerator Traces
  - 2 Bulk Data Copy with Non-Temporal Hint (movnti)

- Randomly Chosen Multiprogrammed Workloads:
  - 125 workloads containing 8 benign applications
  - 125 workloads containing 7 benign applications and 1 RowHammer attack thread
Evaluation
Performance and DRAM Energy

• We classify single-core workloads into three categories based on row buffer conflicts per thousand instructions

• No application’s row activation count exceeds BlockHammer’s blacklisting threshold ($N_{BL}$)

BlockHammer does not incur performance or DRAM energy overheads for single-core benign applications
Evaluation
Performance and DRAM Energy

- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

BlockHammer introduces **very low** performance (<0.5%) and DRAM energy (<0.4%) overheads

BlockHammer **significantly increases** benign application performance (by 45% on average) and **reduces** DRAM energy consumption (by 29% on average)
Evaluation
Scaling with RowHammer Vulnerability

- System throughput (weighted speedup)
- Job turnaround time (harmonic speedup)
- Unfairness (maximum slowdown)
- DRAM energy consumption

BlockHammer’s performance and energy overheads remain negligible (<0.6%)

BlockHammer scalably provides much higher performance (71% on average) and lower energy consumption (32% on average) than state-of-the-art mechanisms.
More in the Paper

• Security Proof
  - Mathematically represent all possible access patterns
  - We show that no row can be activated high-enough times to induce bit-flips when BlockHammer is configured correctly

• Addressing Many-Sided Attacks

• Evaluation of 14 mechanisms representing four mitigation approaches
  - Comprehensive Protection
  - Compatibility with Commodity DRAM Chips
  - Scalability with RowHammer Vulnerability
  - Deterministic Protection

<table>
<thead>
<tr>
<th>Approach</th>
<th>Mechanism</th>
<th>Comprehensive Protection</th>
<th>Compatible w/ Commodity DRAM Chips</th>
<th>Scaling with RowHammer Vulnerability</th>
<th>Deterministic Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased Refresh Rate</td>
<td>[2, 73]</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Physical Isolation</td>
<td>CATT [14]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>GuardIOn [148]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>ZebRAM [78]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Reactive Refresh</td>
<td>ANVIL [5]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PARA [73]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PRoHIT [137]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>MRLoc [161]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CBT [132]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>TWiCe [84]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Graphene [113]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Proactive Throttling</td>
<td>Naive Thrott. [102]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Thrott. Supp. [40]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BlockHammer</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusion

- **Motivation**: RowHammer is a worsening DRAM reliability and security problem

- **Problem**: Mitigation mechanisms have limited support for current/future chips
  - **Scalability** with worsening RowHammer vulnerability
  - **Compatibility** with commodity DRAM chips

- **Goal**: Efficiently and **scalably** prevent RowHammer bit-flips without knowledge of or modifications to DRAM internals

- **Key Idea**: Selectively throttle memory accesses that may cause RowHammer bit-flips

- **Mechanism**: BlockHammer
  - **Tracks** activation rates of all rows by using area-efficient Bloom filters
  - **Throttles** row activations that could cause RowHammer bit flips
  - **Identifies and throttles** threads that perform RowHammer attacks

- **Scalability with Worsening RowHammer Vulnerability**:  
  - **Competitive** with state-of-the-art mechanisms when there is no attack
  - **Superior** performance and DRAM energy when a RowHammer attack is present

- **Compatibility with Commodity DRAM Chips**:  
  - No proprietary information of DRAM internals
  - No modifications to DRAM circuitry
BlockHammer
Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

Abdullah Giray Yağlıkçı
Minesh Patel  Jeremie S. Kim  Roknoddin Azizi
Ataberk Olgun  Lois Orosa  Hasan Hassan  Jisung Park
Konstantinos Kanellopoulos  Taha Shahroodi
Saugata Ghose*  Onur Mutlu

SAFARI

ETH Zürich

University of Illinois Urbana-Champaign
More on BlockHammer

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Intel Hardware Security Academic Awards Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]
[Intel Hardware Security Academic Awards Short Talk Video (2 minutes)]
[BlockHammer Source Code]

Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations)

BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

A. Giray Yağlıkçı¹ Minesh Patel¹ Jeremie S. Kim¹ Roknoddin Azizi¹ Ataberk Olgun¹ Lois Orosa¹ Hasan Hassan¹ Jisung Park¹ Konstantinos Kanellopoulos¹ Taha Shahroodi¹ Saugata Ghose² Onur Mutlu¹

¹ETH Zürich ²University of Illinois at Urbana–Champaign
Read Disturb in Flash Memory
Experimental Testing Platform

HAPS-52 Mother Board

USB Daughter Board

Virtex-V FPGA (NAND Controller)

USB Jack

Virtex-II Pro (USB controller)

1x-nm NAND Flash

NAND Daughter Board


NAND Flash Usage and Error Model

Start

P/E cycle 0

... 

P/E cycle i

... 

P/E cycle n

End of life

Erase Errors

Program Errors

Erase Block

Program Page

Retention Errors

Read Errors

Retention1
(t₁ days)

Read Page

Retention j
(tⱼ days)

Read Page
More on Flash Error Analysis


Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis

Yu Cai\textsuperscript{1}, Erich F. Haratsch\textsuperscript{2}, Onur Mutlu\textsuperscript{1} and Ken Mai\textsuperscript{1}
\textsuperscript{1}Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
\textsuperscript{2}LSI Corporation, 1110 American Parkway NE, Allentown, PA
\textsuperscript{1}\{yucai, onur, kenmai\}@andrew.cmu.edu, \textsuperscript{2}erich.haratsch@lsi.com
Many Errors and Their Mitigation [PIEEE'17]

### Table 3 List of Different Types of Errors Mitigated by NAND Flash Error Mitigation Mechanisms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shadow Program Sequencing [35,40] (Section V-A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neighbor-Cell Assisted Error Correction [36] (Section V-B)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refresh [34,39,67,68] (Section V-C)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Read-Retry [33,72,107] (Section V-D)</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Voltage Optimization [37,38,74] (Section V-E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Hot Data Management [41,63,70] (Section V-F)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Adaptive Error Mitigation [43,65,77,78,82] (Section V-G)</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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
One Issue: Read Disturb in Flash Memory

- All scaled memories are prone to read disturb errors
- DRAM
- SRAM
- Hard Disks: Adjacent Track Interference
- NAND Flash
NAND Flash Memory Background

Flash Memory

Block 0

Read
Pass
Pass
Pass

......

Block N

Flash Controller
Flash Cell

Floating Gate Transistor (Flash Cell)

$V_{th} = 2.5 \text{ V}$
Flash Read

\[ V_{\text{read}} = 2.5 \, \text{V} \]

\[ V_{\text{th}} = 2 \, \text{V} \]

\[ V_{\text{read}} = 2.5 \, \text{V} \]

\[ V_{\text{th}} = 3 \, \text{V} \]
Flash Pass-Through

\[ V_{\text{pass}} = 5 \text{ V} \]

\[ V_{\text{th}} = 2 \text{ V} \]

\[ V_{\text{pass}} = 5 \text{ V} \]

\[ V_{\text{th}} = 3 \text{ V} \]

Gate
More on Flash Read Disturb Errors [DSN’15]

- Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch, Ken Mai, and Onur Mutlu,

"Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation"

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

Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery

Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *Seagate Technology
yucaicai@gmail.com, {yixinluo, ghose, kenmai, onur}@cmu.edu
Read from Flash Cell Array

$V_{pass} = 5.0$

Read (2.5V)

$V_{read} = 2.5$

Pass (5V)

Correct values for page 2:

\[\begin{array}{c}
0 \\
0 \\
1 \\
1 \\
\end{array}\]
Read Disturb Problem: “Weak Programming” Effect

Repeatedly read page 3 (or any page other than page 2)
Read Disturb Problem: “Weak Programming” Effect

$V_{\text{pass}} = 5.0 \text{ V}$

Page 1

$V_{\text{read}} = 2.5 \text{ V}$

Page 2

Incorrect values from page 2: 0 0 0 1

High pass-through voltage induces “weak-programming” effect.
Executive Summary [DSN’15]

- **Read disturb errors** limit flash memory lifetime today
  - Apply a *high pass-through voltage* \( (V_{\text{pass}}) \) to multiple pages on a read
  - Repeated application of \( V_{\text{pass}} \) can alter stored values in unread pages

- We **characterize read disturb** on real NAND flash chips
  - Slightly lowering \( V_{\text{pass}} \) greatly reduces read disturb errors
  - Some flash cells are more prone to read disturb

- **Technique 1:** **Mitigate** read disturb errors online
  - \( V_{\text{pass Tuning}} \) dynamically finds and applies a lowered \( V_{\text{pass}} \) per block
  - Flash memory lifetime improves by 21%

- **Technique 2:** **Recover** after failure to prevent data loss
  - **Read Disturb Oriented Error Recovery** (RDR) selectively corrects cells more susceptible to read disturb errors
  - Reduces raw bit error rate (RBER) by up to 36%
Read Disturb Prone vs. Resistant Cells

Disturb-R Resistant

Disturb-P Prone

N read disturbs

Normalized $V_{th}$

PDF
Observation 2: Some Flash Cells Are More Prone to Read Disturb

After 250K read disturbs:

Disturb-prone cells have higher threshold voltages

Disturb-resistant cells have lower threshold voltages

Disturb-prone $\rightarrow$ ER state

Disturb-resistant $\rightarrow$ P1 state
Read Disturb Oriented Error Recovery (RDR)

• Triggered by an uncorrectable flash error
  – Back up all valid data in the faulty block
  – Disturb the faulty page 100K times (more)
  – Compare $V_{th}$’s before and after read disturb
  – Select cells susceptible to flash errors ($V_{ref} - \sigma < V_{th} < V_{ref} - \sigma$)
  – Predict among these susceptible cells
    • Cells with more $V_{th}$ shifts are disturb-prone $\rightarrow$ Higher $V_{th}$ state
    • Cells with less $V_{th}$ shifts are disturb-resistant $\rightarrow$ Lower $V_{th}$ state

Reduces total error count by up to 36% @ 1M read disturbs
ECC can be used to correct the remaining errors
Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch, Ken Mai, and Onur Mutlu, "Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation"

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

Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery

Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *Seagate Technology
yucaicai@gmail.com, {yixinluo, ghose, kenmai, onur}@cmu.edu
Data Retention in Flash Memory

Large-Scale SSD Error Analysis [SIGMETRICS'15]

- First large-scale field study of flash memory errors


[A Large-Scale Study of Flash Memory Errors in the Field](https://www.sigmetrics.org/archives/conferences/2015/)

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
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
Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu,
"Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery"
[Preliminary arxiv.org version]

Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery

YU CAI, SAUGATA GHOSE
Carnegie Mellon University

ERICH F. HARATSCH
Seagate Technology

YIXIN LUO
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University
More on Flash Memory Issues

Decreasing Endurance with Flash Scaling

- Endurance of flash memory decreasing with scaling and multi-level cells

https://www.youtube.com/watch?v=rninK6KWBeM&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=47
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
Some Selected Readings
Selected Readings on RowHammer (I)

- 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-SAFARI/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
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
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
  - 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.
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](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 (eg. 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.
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
Keeping Future Memory Secure