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

Onur Mutlu
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu
April 21, 2017
CFAED Workshop on Resilient Systems Keynote

Systems@ETH Zürich
SAFARI
ETH Zürich
The Main Memory System

- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits
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]

- As DRAM cell becomes smaller, it becomes more vulnerable
Testing DRAM Scaling 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)
Repeatedly opening and closing a row enough times within a refresh interval 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
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](https://github.com/CMU-SAFARI/rowhammer)
A Simple Program Can Induce Many Errors

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

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

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

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

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

```
loop:
mov (X), %eax
mov (Y), %ebx
clflush (X)
clflush (Y)
mfence
jmp loop
```
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
- In a more controlled environment, we can induce as many as ten million disturbance errors

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

Flipping 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)
“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, 2015)
Rowhammer

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after.
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 (Seaborne+, 2015)
  - https://github.com/google/rowhammer-test
  - Double-sided Rowhammer
Selected Readings on RowHammer (II)

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

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

- **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., 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
More Security Implications

Not there yet, but ...

ROWHAMMERJS

ROOT privileges for web apps!

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript

Source: https://lab.dsst.io/32c3-slides/7197.html
More Security Implications

Hammer And Root

Millions of Androids

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/
More Security Implications?
Root Causes of Disturbance Errors

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

• **Cause 2: Conductive bridges**

• **Cause 3: Hot-carrier injection**

Confirmed by at least one manufacturer
Experimental DRAM Testing Infrastructure

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

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

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

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

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

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

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

Most aggressors & victims are adjacent
Access Interval (Aggressor)

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

Less frequent accesses → Fewer errors
2 Refresh Interval

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

More frequent refreshes → Fewer errors
Data Pattern

- **Solid**
  - 111111
  - 111111
  - 111111
  - 111111

- **RowStripe**
  - 111111
  - 000000
  - 111111
  - 000000

- **~Solid**
  - 000000
  - 000000
  - 000000
  - 000000

- **RowStripe**
  - 000000
  - 111111
  - 000000
  - 111111

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

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

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

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

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

• **Number of cells & rows affected by aggressor**
  – Victims cells per aggressor: $\leq 110$
  – Victims rows per aggressor: $\leq 9$

• **Cells affected by two aggressors on either side**
  – Very small fraction of victim cells ($<100$) have an error when either one of the aggressors is toggled
Some Potential Solutions

- Make better DRAM chips  Cost
- Refresh frequently  Power, Performance
- Sophisticated ECC  Cost, Power
- Access counters  Cost, Power, Complexity
Naive Solutions

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

2. **Refresh more frequently**
   - Shorten refresh-interval by \( \sim 7x \)

Both naive solutions introduce significant overhead in performance and power
Apple’s 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 and Lenovo released similar patches
Our Solution

• PARA: **Probabilistic Adjacent Row Activation**

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

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

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

- **PARA is stateless**
  - Low cost
  - Low complexity

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

- If implemented in **DRAM chip**
  - Enough slack in timing parameters
  - Plenty of slack today:
    - Lee et al., “Design-Induced Latency Variation in Modern DRAM Chips,” SIGMETRICS 2017.

- If implemented in **memory controller**
  - Better coordination between memory controller and DRAM
  - Memory controller should know which rows are physically adjacent
More on RowHammer Analysis

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

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

Future of Main Memory

- DRAM is becoming less reliable $\rightarrow$ more vulnerable
Large-Scale Failure Analysis of DRAM Chips

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

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

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

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

Intuition: quadratic increase in capacity
Future of Main Memory

- DRAM is becoming less reliable → more vulnerable

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

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

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

- Determining the data retention time of a cell/row is getting more difficult

- Retention failures may already be slipping into the field
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms
Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time

- Variable Retention Time (VRT) phenomenon
Two Challenges to Retention Time Profiling

- **Challenge 1: Data Pattern Dependence (DPD)**
  - Retention time of a DRAM cell depends on its value and the values of cells nearby it.
  - When a row is activated, all bitlines are perturbed simultaneously.
Data Pattern Dependence

- Electrical noise on the bitline affects reliable sensing of a DRAM cell
- The magnitude of this noise is affected by values of nearby cells via
  - Bitline-bitline coupling → electrical coupling between adjacent bitlines
  - Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline

- Retention time of a cell depends on data patterns stored in nearby cells
  - Need to find the worst data pattern to find worst-case retention time
  - This pattern is location dependent
Two Challenges to Retention Time Profiling

- **Challenge 2: Variable Retention Time (VRT)**
  - Retention time of a DRAM cell changes randomly over time
    - a cell alternates between multiple retention time states
  - Leakage current of a cell changes sporadically due to a charge trap in the gate oxide of the DRAM cell access transistor
  - When the trap becomes occupied, charge leaks more readily from the transistor’s drain, leading to a short retention time
    - Called *Trap-Assisted Gate-Induced Drain Leakage*
  - This process appears to be a random process
  - Worst-case retention time depends on a random process
    → need to find the worst case despite this
Newer device families have more weak cells than older ones
Likely a result of technology scaling
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

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

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
The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study

Samira Khan†∗
samirakhan@cmu.edu

Donghyuk Lee†
donghyuk1@cmu.edu

Yoongu Kim†
yoongukim@cmu.edu

Alaa R. Alameldeen∗
alaa.r.alameldeen@intel.com

Chris Wilkerson∗
chris.wilkerson@intel.com

Onur Mutlu†
onur@cmu.edu

†Carnegie Mellon University  ∗Intel Labs
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
Handling Data-Dependent Failures [DSN’16]

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

Samira Khan* Donghyuk Lee†‡ Onur Mutlu*†
*University of Virginia †Carnegie Mellon University ‡Nvidia *ETH Zürich
How Do We Keep Memory Secure?

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

SAFARI
How Do We Keep Memory Secure?

- **Understand**: Solid methodologies for failure modeling and discovery
  - Modeling based on real device data – small scale and large scale

- **Architect**: Principled co-architecting of system and memory
  - Good partitioning of duties across the stack

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

Understand with Experiments (Flash)

Flash memory is widening its range of applications

- Portable consumer devices, laptop PCs and enterprise servers
Flash Challenges: Reliability and Endurance

E. Grochowski et al., “Future technology challenges for NAND flash and HDD products”, Flash Memory Summit 2012

- **P/E cycles (provided)**
  - A few thousand

- **P/E cycles (required)**
  - Writing the full capacity of the drive 10 times per day for 5 years (STEC)
  - > 50k P/E cycles

E. Grochowski et al., “Future technology challenges for NAND flash and HDD products”, Flash Memory Summit 2012
NAND Flash Memory is Increasingly Noisy

Write → Noisy NAND → Read
Our Goals:

Build reliable error models for NAND flash memory
Design efficient reliability mechanisms based on the model
NAND Flash Error Model

Experimentally characterize and model dominant errors


Luo et al., “Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory”, JSAC 2016


Cai et al., “Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques”, HPCA 2017


Cai et al., “Neighbor-Cell Assisted Error Correction in MLC NAND Flash Memories”, SIGMETRICS 2014

Cai et al., “Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation”, DSN 2015


Cai et al., “Error Analysis and Retention-Aware Error Management for NAND Flash Memory”, ITJ 2013

Cai et al., “Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery”, HPCA 2015
Our Goals and Approach

- **Goals:**
  - Understand error mechanisms and develop reliable predictive models for MLC NAND flash memory errors
  - Develop efficient error management techniques to mitigate errors and improve flash reliability and endurance

- **Approach:**
  - Solid experimental analyses of errors in real MLC NAND flash memory → drive the understanding and models
  - Understanding, models, and creativity → drive the new techniques
Experimental Testing Platform

Cai et al., FPGA-based Solid-State Drive prototyping platform, FCCM 2011.
NAND Flash Error Types

- Four types of errors [Cai+, DATE 2012]

- Caused by common flash operations
  - Read errors
  - Erase errors
  - Program (interference) errors

- Caused by flash cell losing charge over time
  - Retention errors
    - Whether an error happens depends on required retention time
    - Especially problematic in MLC flash because threshold voltage window to determine stored value is smaller
Observations: Flash Error Analysis

- Raw bit error rate increases exponentially with P/E cycles
- Retention errors are dominant (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement

Cai et al., Error Patterns in MLC NAND Flash Memory, DATE 2012.
More on Flash Error Analysis

Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"

Proceedings of the
Design, Automation, and Test in Europe Conference
(DATE), Dresden, Germany, March 2012. Slides (ppt)
Solution to Retention Errors

- Refresh periodically
- Change the period based on P/E cycle wearout
  - Refresh more often at higher P/E cycles
- Use a combination of in-place and remapping-based refresh

Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai\textsuperscript{1}, Gulay Yalcin\textsuperscript{2}, Onur Mutlu\textsuperscript{1}, Erich F. Haratsch\textsuperscript{3}, Adrian Cristal\textsuperscript{2}, Osman S. Unsal\textsuperscript{2} and Ken Mai\textsuperscript{1}

\textsuperscript{1}DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
\textsuperscript{2}Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain
\textsuperscript{3}LSI Corporation, 1110 American Parkway NE, Allentown, PA
One Issue: Read Disturb in Flash Memory

- All scaled memories are prone to read disturb errors
NAND Flash Memory Background

Flash Memory

Block 0
Read
Pass
Pass
Pass

......

Block N

Flash Controller
Flash Cell Array

Block X
Page Y

Row
Column

Sense Amplifiers
Flash Cell

Floating Gate Transistor (Flash Cell)
Flash Read

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

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

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

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

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

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

\[ V_{\text{th}} = 3 \text{ V} \]
Read from Flash Cell Array

- **V_{\text{pass}} = 5.0 V**
  - **Pass (5V)**
  - Page 1

- **V_{\text{read}} = 2.5 V**
  - **Read (2.5V)**
  - Page 2

- **V_{\text{pass}} = 5.0 V**
  - **Pass (5V)**
  - Page 3

- **V_{\text{pass}} = 5.0 V**
  - **Pass (5V)**
  - Page 4

Correct values for page 2:

```
0 0 1 1
```
Read Disturb Problem: “Weak Programming” Effect

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

Incorrect values from page 2:

High pass-through voltage induces “weak-programming” effect.
Executive Summary

- **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%
More on Flash Read Disturb 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
Large-Scale Flash SSD Error Analysis

- First large-scale field study of flash memory errors

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu,
  "A Large-Scale Study of Flash Memory Errors in the Field"

Proceedings of the
ACM International Conference on Measurement and Modeling of
Computer Systems (SIGMETRICS), Portland, OR, June 2015.
[Slides (pptx) (pdf)] [Coverage at ZDNet] [Coverage on The Register]
[Coverage on TechSpot] [Coverage on The Tech Report]

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

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
Another Time: NAND Flash Vulnerabilities

- Onur Mutlu,
  "Error Analysis and Management for MLC NAND Flash Memory"
  Technical talk at Flash Memory Summit 2014 (FMS), Santa Clara, CA, August 2014. Slides (ppt) (pdf)

Cai+，“Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis,” DATE 2012.
Cai+，“Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation,” ICCD 2013.
Cai+，“Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery,” HPCA 2015.
Cai+，“Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation,” DSN 2015.
Luo+，“WARM: Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management,” MSST 2015.
Meza+，“A Large-Scale Study of Flash Memory Errors in the Field,” SIGMETRICS 2015.
Luo+，“Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory,” IEEE JSAC 2016.
Flash Memory Programming Vulnerabilities

Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

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

Summary

- Memory reliability is reducing
- Reliability issues open up security vulnerabilities
  - Very hard to defend against
- Rowhammer is an example
  - Its implications on system security research are tremendous & exciting

**Good news: We have a lot more to do.**

- **Understand:** Solid methodologies for failure modeling and discovery
  - Modeling based on real device data – small scale and large scale
- **Architect:** Principled co-architecting of system and memory
  - Good partitioning of duties across the stack
- **Design & Test:** Principled electronic design, automation, testing
  - High coverage and good interaction with system reliability methods
The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu
April 21, 2017
CFAED Workshop on Resilient Systems Keynote
More on DRAM Data Retention
DRAM Refresh

- DRAM capacitor charge leaks over time

- The memory controller needs to refresh each row periodically to restore charge
  - Activate each row every N ms
  - Typical N = 64 ms

- Downsides of refresh
  - Energy consumption: Each refresh consumes energy
  - Performance degradation: DRAM rank/bank unavailable while refreshed
  - QoS/predictability impact: (Long) pause times during refresh
  - Refresh rate limits DRAM capacity scaling
Refresh Overhead: Performance

Refresh Overhead: Energy

Retention Time Profile of DRAM

64-128ms

\( >256 \text{ms} \)

128-256ms
RAIDR: Eliminating Unnecessary Refreshes

- **Observation:** Most DRAM rows can be refreshed much less often without losing data [Kim+, EDL’09][Liu+ ISCA’13]

- **Key idea:** Refresh rows containing weak cells more frequently, other rows less frequently
  
  1. **Profiling:** Profile retention time of all rows
  2. **Binning:** Store rows into bins by retention time in memory controller
     
     *Efficient storage with Bloom Filters* (only 1.25KB for 32GB memory)
  3. **Refreshing:** Memory controller refreshes rows in different bins at different rates

- **Results:** 8-core, 32GB, SPEC, TPC-C, TPC-H
  
  - 74.6% refresh reduction @ 1.25KB storage
  - ~16%/20% DRAM dynamic/idle power reduction
  - ~9% performance improvement
  - Benefits increase with DRAM capacity

RowHammer in Popular Sites and Press

- [https://twitter.com/hashtag/rowhammer?f=realtime](https://twitter.com/hashtag/rowhammer?f=realtime)
- [https://www.youtube.com/watch?v=H63dUfGBpxE](https://www.youtube.com/watch?v=H63dUfGBpxE)
- [http://www.wired.com/2015/03/google-hack-dram-memory-electric-leaks/](http://www.wired.com/2015/03/google-hack-dram-memory-electric-leaks/)
Recap: The DRAM Scaling Problem

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
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
Towards an Online Profiling System

Key Observations:

• **Testing** alone cannot detect all possible failures

• **Combination** of ECC and other mitigation techniques is much more **effective**
  – But degrades performance

• **Testing** can help to reduce the **ECC strength**
  – Even when starting with a **higher strength ECC**

Towards an Online Profiling System

1. Initially Protect DRAM with Strong ECC
2. Periodically Test Parts of DRAM
3. Mitigate errors and reduce ECC

Run tests periodically after a short interval at smaller regions of memory
Online Mitigating of DRAM Failures

- 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"

Proceedings of the
ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Austin, TX, June 2014. [Slides (pptx) (pdf)] [Poster (pptx) (pdf)] [Full data sets]
Memory Errors in Facebook Fleet

- 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.

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

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

Error/failure occurrence

Page offlining at scale
Technology scaling
New reliability trends
Modeling errors
Architecture & workload
Findings

Error/failure occurrence

Errors follow a *power-law distribution* and a large number of errors occur due to *sockets/channels*
Findings

We find that newer cell fabrication technologies have higher failure rates.
Findings

Chips per DIMM, transfer width, and workload type (not necessarily CPU/memory utilization) affect reliability.
Findings

We have made publicly available a statistical model for assessing server memory reliability.
Findings

Error/failure occurrence

Page offlining at scale

First large-scale study of page offlining; real-world limitations of technique

Reliability trends

Modeling errors

Architecture & workload
Server error rate

- Correctable errors (CE)
- Uncorrectable errors (UCE)

Fraction of servers:
- 0.040
- 0.035
- 0.030
- 0.025
- 0.020
- 0.015
- 0.010
- 0.005
- 0.000

Month:
- 7/13
- 8/13
- 9/13
- 10/13
- 11/13
- 12/13
- 1/14
- 2/14
- 3/14
- 4/14
- 5/14
- 6/14
- 7/14
- 8/14
Memory error distribution

Number of logged errors

Normalized device number

Measured
Power law
Memory error distribution

\[ \Pr(\text{logged errors} > x) \]

- Measured
- Pareto \( (R^2 = 0.97) \)

Decreasing hazard rate
Errors in Flash Memory (I)

1. **Retention noise study and management**

2. **Flash-based SSD prototyping and testing platform**
3. **Overall flash error analysis**


4. **Program and erase noise study**

5. Cell-to-cell interference characterization and tolerance
8) Yu Cai, Onur Mutlu, Erich F. Haratsch, and Ken Mai, 
Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation, ICCD 2013.

9) Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Osman Unsal, Adrian Cristal, and Ken Mai, 
Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories, SIGMETRICS 2014.

6. Read disturb noise study
10) 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, DSN 2015.

7. Flash errors in the field
11) Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, 
A Large-Scale Study of Flash Memory Errors in the Field, SIGMETRICS 2015.
More on Flash Retention Errors

- Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu, "Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery"
  [Slides (pptx) (pdf)]

Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery

Yu Cai, Yixin Luo, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *LSI Corporation
yucaicai@gmail.com, yixinluo@cs.cmu.edu, erich.haratsch@lsi.com, {kenmai, omutlu}@ece.cmu.edu
More on Flash Read Disturb 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
More on Flash Error Analysis

Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"


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

Yu Cai¹, Erich F. Haratsch², Onur Mutlu¹ and Ken Mai¹

¹Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
²LSI Corporation, 1110 American Parkway NE, Allentown, PA
¹{yucai, onur, kenmai}@andrew.cmu.edu, ²erich.haratsch@lsi.com
More Detail on Flash Error Analysis

- Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai,
  "Error Analysis and Retention-Aware Error Management for NAND Flash Memory"
Google’s RowHammer Attack

The following slides are from Mark Seaborn and Thomas Dullien’s BlackHat 2015 talk

Kernel exploit

- x86 page tables entries (PTEs) are **dense and trusted**
  - They control access to physical memory
  - A bit flip in a PTE’s physical page number can give a process access to a different physical page
- Aim of exploit: Get access to a page table
  - Gives access to all of physical memory
- Maximise chances that a bit flip is useful:
  - Spray physical memory with page tables
  - Check for useful, repeatable bit flip first

This slide is from Mark Seaborn and Thomas Dullien’s BlackHat 2015 talk
x86-64 Page Table Entries (PTEs)

- Page table is a 4k page containing array of 512 PTEs
- Each PTE is 64 bits, containing:

![Diagram of 4-Kbyte PTE—Long Mode]

- Could flip:
  - “Writable” permission bit (RW): 1 bit → 2% chance
  - Physical page number: 20 bits on 4GB system → 31% chance

This slide is from Mark Seaborn and Thomas Dullien’s BlackHat 2015 talk

What happens when we map a file with read-write permissions?
What happens when we map a file with read-write permissions? Indirection via page tables.
What happens when we repeatedly map a file with read-write permissions?
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.

If a bit in the right place in the PTE flips ...
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.

If a bit in the right place in the PTE flips …

… the corresponding virtual address now points to a wrong physical page - with RW access.
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.

If a bit in the right place in the PTE flips …

… the corresponding virtual address now points to a wrong physical page - with RW access.

Chances are this wrong page contains a page table itself.
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.

If a bit in the right place in the PTE flips ...

... the corresponding virtual address now points to a wrong physical page - with RW access.

Chances are this wrong page contains a page table itself.

An attacker that can read / write page tables ...
What happens when we repeatedly map a file with read-write permissions?

PTEs in physical memory help resolve virtual addresses to physical pages.

We can fill physical memory with PTEs.

Each of them points to pages in the same physical file mapping.

If a bit in the right place in the PTE flips …

… the corresponding virtual address now points to a wrong physical page - with RW access.

Chances are this wrong page contains a page table itself.

An attacker that can read / write page tables can use that to map any memory read-write.
Exploit strategy

Privilege escalation in 7 easy steps …
1. Allocate a large chunk of memory
2. Search for locations prone to flipping
3. Check if they fall into the “right spot” in a PTE for allowing the exploit
4. Return that particular area of memory to the operating system
5. Force OS to re-use the memory for PTEs by allocating massive quantities of address space
6. Cause the bitflip - shift PTE to point into page table
7. Abuse R/W access to all of physical memory

In practice, there are many complications.