How Do Problems Get Solved by Electrons?
Recall: The Transformation Hierarchy

- Problem
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture
- Logic
- Devices
- Electrons

Computer Architecture (expanded view)

Computer Architecture (narrow view)
Levels of Transformation

“The purpose of computing is [to gain] insight” (Richard Hamming)

We gain and generate insight by solving problems

How do we ensure problems are solved by electrons?

Algorithm

Step-by-step procedure that is **guaranteed to terminate** where each step is precisely stated and can be carried out by a computer

- Finiteness
- Definiteness
- Effective computability

Many algorithms for the same problem

ISA (Instruction Set Architecture)

Interface/contract between SW and HW.

What the programmer assumes hardware will satisfy.

Microarchitecture

An implementation of the ISA

Digital logic circuits

Building blocks of micro-arch (e.g., gates)
Aside: A Famous Work By Hamming


- Introduced the concept of Hamming distance
  - number of locations in which the corresponding symbols of two equal-length strings is different

- Developed a theory of codes used for error detection and correction

- Also see:
Levels of Transformation, Revisited

- A user-centric view: computer designed for users

- The entire stack should be optimized for user
The Power of Abstraction

- **Levels of transformation create abstractions**
  - Abstraction: A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
  - E.g., high-level language programmer does not really need to know what the ISA is and how a computer executes instructions

- **Abstraction improves productivity**
  - No need to worry about decisions made in underlying levels
  - E.g., programming in Java vs. C vs. assembly vs. binary vs. by specifying control signals of each transistor every cycle

- Then, why would you want to know what goes on underneath or above?
Crossing the Abstraction Layers

- As long as everything goes well, not knowing what happens underneath (or above) is not a problem.

- What if
  - The program you wrote is running slow?
  - The program you wrote does not run correctly?
  - The program you wrote consumes too much energy?
  - Your system just shut down and you have no idea why?
  - Someone just compromised your system and you have no idea how?

- What if
  - The hardware you designed is too hard to program?
  - The hardware you designed is too slow because it does not provide the right primitives to the software?

- What if
  - You want to design a much more efficient and higher performance system?
Two goals of this course (especially the second half) are

- to understand how a processor works underneath the software layer and how decisions made in hardware affect the software/programmer

- to enable you to be comfortable in making design and optimization decisions that cross the boundaries of different layers and system components
Some Example “Mysteries”
Four Mysteries: Familiar with Any?

- Meltdown & Spectre (2017-2018)
- Rowhammer (2012-2014)
- Memories Forget: Refresh (2011-2012)
- Memory Performance Attacks (2006-2007)
Mystery #1:
  Meltdown & Spectre
What Are These?

MELTDOWN

SPECTRE

Source: J. Masters, Redhat, FOSDEM 2018 keynote talk.
Meltdown and Spectre Attacks

- Someone can steal secret data from the system even though
  - your program and data are perfectly correct and
  - your hardware behaves according to the specification and
  - there are no software vulnerabilities/bugs
Meltdown and Spectre

- Hardware security vulnerabilities that essentially affect almost all computer chips that were manufactured in the past two decades

- They exploit “speculative execution”
  - A technique employed in modern processors for high performance

- Speculative execution: Doing something before you know that it is needed
  - We do it all the time in life, to save time & be faster
    - Guess what will happen and act based on that guess
  - Processors do it, too, to run programs fast
    - They guess and execute code before they know it should be executed
Speculative Execution (I)

- Modern processors “speculatively execute” code to improve performance:

```java
if (account-balance <= 0) {
    // do something
} else if (account-balance < 1M) {
    // do something else
} else {
    // do something else
}
```

Guess what code will be executed and execute it speculatively
- Improves performance, if it takes a long time to access account-balance

If the guess was wrong, flush the wrong instructions and execute the correct code
Speculative Execution is Invisible to the User

<table>
<thead>
<tr>
<th>Problem</th>
<th>ISA (Instruction Set Architecture)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Interface/contract between SW and HW.</td>
</tr>
<tr>
<td>Program/Language</td>
<td>What the programmer assumes hardware will satisfy.</td>
</tr>
<tr>
<td>System Software</td>
<td>Programmer assumes their code will be executed in sequential order</td>
</tr>
</tbody>
</table>

**Microarchitecture**

An implementation of the ISA

- Microarchitecture executes instructions in a different order, speculatively – but reports the results as expected by the programmer.
Meltdown and Spectre

- Someone can steal secret data from the system even though
  - your program and data are perfectly correct and
  - your hardware behaves according to the specification and
  - there are no software vulnerabilities/bugs

- Why?
  - Speculative execution leaves traces of secret data in the processor’s cache (internal storage)
    - It brings data that is not supposed to be brought/accessed if there was no speculative execution
  - A malicious program can inspect the contents of the cache to “infer” secret data that it is not supposed to access
  - A malicious program can actually force another program to speculatively execute code that leaves traces of secret data
Processor Cache as a Side Channel

- Speculative execution leaves traces of data in processor cache
  - Architecturally correct behavior w.r.t. specification
  - However, this leads to a side channel: a channel through which someone sophisticated can extract information

- Processor cache leaks information by storing speculatively-accessed data
  - A clever attacker can probe the cache and infer the secret data values
    - by measuring how long it takes to access the data
  - A clever attacker can force a program to speculatively execute code and leave traces of secret data in the cache
More on Meltdown/Spectre Side Channels

Project Zero

News and updates from the Project Zero team at Google

Wednesday, January 3, 2018

Reading privileged memory with a side-channel

Posted by Jann Horn, Project Zero

We have discovered that CPU data cache timing can be abused to efficiently leak information out of mis-speculated execution, leading to (at worst) arbitrary virtual memory read vulnerabilities across local security boundaries in various contexts.

Three Questions

- Can you figure out why someone stole your secret data if you do not know how the processor executes a program?

- Can you fix the problem without knowing what is happening “underneath”, i.e., inside the microarchitecture?

- Can you fix the problem well/fundamentally without knowing both software and hardware design?

- Can you construct this attack or similar attacks without knowing what is happening underneath?
Three Other Questions

- **What are the causes of Meltdown and Spectre?**

- **How can we prevent them (while keeping the performance benefits of speculative execution)?**
  - Software changes?
  - Operating system changes?
  - Instruction set architecture changes?
  - Microarchitecture/hardware changes?
  - Changes at multiple layers, done cooperatively?
  - ...

- **How do we design high-performance processors that do not leak information via side channels?**
Meltdown/Spectre Span Across the Hierarchy

Computer Architecture (expanded view)

Meltdown/Spectre problem and solution space

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Computer Architecture (narrow view)
Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
... and Also Understand/Critique Cartoons!

**THE MELTDOWN AND SPECTRE EXPLOITS USE**
**“SPECULATIVE EXECUTION?” WHAT’S THAT?**

YOU KNOW THE TROLLEY PROBLEM? WELL, FOR A WHILE NOW, CPUS HAVE BASICALLY BEEN SENDING TROLLEYS DOWN BOTH PATHS, QUANTUM-STYLE, WHILE AWAITING YOUR CHOICE. THEN THE UNNEEDED "PHANTOM" TROLLEY DISAPPEARS.

**THE PHANTOM TROLLEY ISN’T SUPPOSED TO TOUCH ANYONE. BUT IT TURNS OUT YOU CAN STILL USE IT TO DO STUFF. AND IT CAN DRIVE THROUGH WALLS.**

**THAT SOUNDS BAD.**
**HONESTLY, I’VE BEEN ASSUMING WE WERE DOOMED EVER SINCE I LEARNED ABOUT ROWHAMMER.**

**WHAT’S THAT?**
**IF YOU TOGGLE A ROW OF MEMORY CELLS ON AND OFF REALLY FAST, YOU CAN USE ELECTRICAL INTERFERENCE TO FLIP NEARBY BITS AND—**

**DO WE JUST SUCK AT...COMPUTERS?**
**YUP. ESPECIALLY SHARED ONES.**

**SO YOU’RE SAYING THE CLOUD IS FULL OF PHANTOM TROLLEYS ARMED WITH HAMMERS.**

...YES, THAT IS EXACTLY RIGHT. OKAY, I’LL, UH... INSTALL UPDATES?

GOOD IDEA.

Source: [https://xkcd.com/1938/](https://xkcd.com/1938/)
An Important Note: Design Goal and Mindset

- Design goal of a system determines the design mindset and evaluation metrics

- Meltdown and Spectre are there because the design goal of cutting-edge processors (employed everywhere in our lives)
  - has mainly been focused on **high performance and low energy** (relatively recently)
  - has **not included security** (or information leakage) as an important constraint

- Incorporating security as a first-class constraint and “metric” into (hardware) design and education is critical in today’s world
Design Mindset

Security is about preventing unforeseen consequences

Source: https://s-media-cache-ak0.pinimg.com/originals/48/09/54/4809543a9c7700246a0cf8acdae27abf.jpg
Two Other Goals of This Course

- Enable you to **think critically**
- Enable you to **think broadly**
To Learn and Discover Further

- High-level Video by RedHat
  - https://www.youtube.com/watch?v=syAdX44pokE

- A bit lower-level, comprehensive explanation by Y. Vigfusson
  - https://www.youtube.com/watch?v=mgAN4w7LH2o

- Keep attending lectures and taking in all the material

- Go and talk with myself in the future
  - I have many bachelor’s/master’s projects on hardware security
  - “Fundamentally secure computing architectures” is a key direction of scientific investigation and design
Mystery #2: RowHammer
The Story of RowHammer

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

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

WIRED

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

ANDY GREENBERG  SECURITY  08.31.16  7:00 AM

SHARE

18276
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
Recent DRAM Is More Vulnerable

![Graph showing errors per 10^9 cells vs module vintage]
Recent DRAM Is More Vulnerable

All modules from 2012–2013 are vulnerable
Why Is This Happening?

- DRAM cells are too close to each other!
  - They are not electrically isolated from each other

- Access to one cell affects the value in nearby cells
  - due to *electrical interference* between
    - the cells
    - wires used for accessing the cells
  - Also called cell-to-cell coupling/interference

- Example: When we activate (apply high voltage) to a row, an adjacent row gets slightly activated as well
  - Vulnerable cells in that slightly-activated row lose a little bit of charge
  - If row hammer happens enough times, charge in such cells gets drained
Higher-Level Implications

- This simple circuit-level failure mechanism has enormous implications on upper layers of the transformation hierarchy.

![Diagram showing the transformation hierarchy from Problem to User with intermediate layers like Algorithm, Program/Language, Runtime System (VM, OS, MM), ISA (Architecture), Microarchitecture, Logic, Devices, and Electrons.](image-url)
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

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

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

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

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

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](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)
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

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)
Security Implications

Rowhammer

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

Not there yet, but ... 

ROOT privileges for web apps!

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

“Can gain control of a smart phone deterministically”

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/
More Security Implications (III)

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

"GRAND PWNING UNIT" —

Drive-by Rowhammer attack uses GPU to compromise an Android phone

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

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

Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU

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

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

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

Kaveh Razavi
Vrije Universiteit Amsterdam
kaveh@cs.vu.nl
More Security Implications (IV)

- Rowhammer over RDMA (I)

**THROWHAMMER** —

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
The Hacker News
Security in a serious way

Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests

Nethammer:
Inducing Rowhammer Faults through Network Requests

Moritz Lipp
Graz University of Technology

Misiker Tadesse Aga
University of Michigan

Michael Schwarz
Graz University of Technology

Daniel Gruss
Graz University of Technology

Clémentine Maurice
Univ Rennes, CNRS, IRISA

Lukas Raab
Graz University of Technology

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

- IEEE S&P 2020

RAMBleed

RAMBleed: Reading Bits in Memory Without Accessing Them

Andrew Kwong  
*University of Michigan*  
ankwong@umich.edu

Daniel Genkin  
*University of Michigan*  
genkin@umich.edu

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

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

- 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 Dumitraş

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  
dfan@asu.edu

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?
Where RowHammer Was Discovered...

How Do We Fix The Problem?
Some Potential Solutions

- Make better DRAM chips  
  - Cost

- Refresh frequently  
  - Power, Performance

- Sophisticated Error Correction  
  - 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
A Cheaper 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 provide an arbitrarily strong protection against errors
Probabilistic Activation in Real Life (I)

https://twitter.com/isislovecruft/status/1021939922754723841
Some Thoughts on RowHammer

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

- How to find, exploit and fix the vulnerability requires a strong understanding across the transformation layers
  - And, a strong understanding of tools available to you

- Fixing needs to happen for two types of chips
  - Existing chips (already in the field)
  - Future chips

- Mechanisms for fixing are different between the two types
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

LESLE 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
Really Interested?

- 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-SAFA/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
RowHammer: Seven Years Ago…

- 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]

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: Now and Beyond…

- 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 (30 minutes)]
Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
RowHammer in 2020
RowHammer in 2020 (I)

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

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

Jeremie S. Kim§†, Minesh Patel§, A. Giray Yaglıkçı§
Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§†
§ETH Zürich †Carnegie Mellon University
RowHammer in 2020 (II)

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi,
  "TRRespass: Exploiting the Many Sides of Target Row Refresh"
  [Slides (pptx) (pdf)]
  [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.
RowHammer in 2020 (III)

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

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

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

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

Lucian Cojocar, Jeremie Kim§†, Minesh Patel§, Lillian Tsai†, Stefan Saroiu, Alec Wolman, and Onur Mutlu§†
Microsoft Research, §ETH Zürich, †CMU, ‡MIT
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"


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
The Story of RowHammer Lecture …

- Onur Mutlu, "The Story of RowHammer"
  Keynote Talk at Secure Hardware, Architectures, and Operating Systems Workshop (SeHAS), held with HiPEAC 2021 Conference, Virtual, 19 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]
Detailed Lectures on RowHammer

- **Computer Architecture, Fall 2020, Lecture 4b**
  - RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=K Dy632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8](https://www.youtube.com/watch?v=K Dy632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8)

- **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/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9)

- **Computer Architecture, Fall 2020, Lecture 5b**
  - RowHammer in 2020: Revisiting RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=gR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10](https://www.youtube.com/watch?v=gR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10)

- **Computer Architecture, Fall 2020, Lecture 5c**
  - Secure and Reliable Memory (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11](https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11)

[SAFARI](https://www.youtube.com/onurmutlulectures)
Maslow’s (Human) Hierarchy of Needs


- We need to start with reliability and security...
How Reliable/Secure/Safe is This Bridge?
Collapse of the “Galloping Gertie”

Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm
How Secure Are These People?

Security is about preventing unforeseen consequences

Source: https://s-media-cache-ak0.pinimg.com/originals/48/09/54/4809543a9c7700246a0cf8acdae27abf.jpg
RowHammer: Retrospective

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

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

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

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

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

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

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

RowHammer: A Retrospective

Onur Mutlu$^{\S\S}$  Jeremie S. Kim$^{\S\S}$

$^{\S}$ETH Zürich       $^{\S\S}$Carnegie Mellon University
Mystery #3: DRAM Refresh
DRAM in the System

*Die photo credit: AMD Barcelona*
A DRAM cell consists of a capacitor and an access transistor.

It stores data in terms of charge status of the capacitor.

A DRAM chip consists of (10s of 1000s of) rows of such cells.
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**
First, Some Analysis

- Imagine a system with 1 ExaByte DRAM \(2^{60} \text{ bytes}\)
- Assume a row size of 8 KiloBytes \(2^{13} \text{ bytes}\)

- How many rows are there?
- How many refreshes happen in 64ms?
- What is the total power consumption of DRAM refresh?
- What is the total energy consumption of DRAM refresh during a day?

- A good exercise... Optional homework...
- Brownie points from me if you do it...
Refresh Overhead: Performance

Refresh Overhead: Energy

How Do We Solve the Problem?

- Observation: All DRAM rows are refreshed every 64ms.

- Critical thinking: Do we need to refresh all rows every 64ms?

- What if we knew what happened underneath and exposed that information to upper layers?
Underneath: Retention Time Profile of DRAM

64-128ms

>256ms

128-256ms

Aside: Why Do We Have Such a Profile?

- Answer: Manufacturing is not perfect
- Not all DRAM cells are exactly the same
- Some are more leaky than others
- This is called Manufacturing Process Variation
Opportunity: Taking Advantage of This Profile

- Assume we know the retention time of each row exactly

- What can we do with this information?

- Who do we expose this information to?

- How much information do we expose?
  - Affects hardware/software overhead, power, verification complexity, cost

- How do we determine this profile information?
  - Also, who determines it?
Observation: Overwhelming majority of DRAM rows can be refreshed much less often without losing data.

Key Idea of RAIDR: Refresh weak rows more frequently, all other rows less frequently.

RAIDR: Eliminating Unnecessary DRAM Refreshes

Liu, Jaiyen, Veras, Mutlu,
RAIDR: Retention-Aware Intelligent DRAM Refresh
ISCA 2012.
RAIDR: Mechanism

1. Profiling: Identify the retention time of all DRAM rows

64-128ms

1.25KB storage in controller for 32GB DRAM memory

→ check the bins to determine refresh rate of a row

>256ms

128-256ms

RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; Various workloads
- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%
- Benefits increase as DRAM scales in density
Reading for the Really Interested


RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu  Ben Jaiyen  Richard Veras  Onur Mutlu
Carnegie Mellon University
{jamiel,bjaiyen,rveras,onur}@cmu.edu
Really Interested? … Further Readings

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  Technical talk at MemCon 2013 (MEMCON), Santa Clara, CA, August 2013.
  Slides (pptx) (pdf) Video

- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
  "Improving DRAM Performance by Parallelizing Refreshes with Accesses"
Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
Mystery #4: Memory Performance Attacks
Multi-Core Systems

*Die photo credit: AMD Barcelona
A Trend: Many Cores on Chip

- Simpler and lower power than a single large core
- Parallel processing on single chip → faster, new applications

AMD Barcelona
4 cores

Intel Core i7
8 cores

IBM Cell BE
8+1 cores

IBM POWER7
8 cores

Sun Niagara II
8 cores

Nvidia Fermi
448 “cores”

Intel SCC
48 cores, networked

Tilera TILE Gx
100 cores, networked
Many Cores on Chip

- What we want:
  - N times the system performance with N times the cores

- What do we get today?
Unexpected Slowdowns in Multi-Core

Three Questions

- Can you figure out why the applications slow down if you do not know the underlying system and how it works?

- Can you figure out why there is a disparity in slowdowns if you do not know how the system executes the programs?

- Can you fix the problem without knowing what is happening “underneath”?
Three Questions

- Why is there any slowdown?
- Why is there a disparity in slowdowns?
- How can we solve the problem if we do not want that disparity?
  - What do we want (the system to provide)?
Why Is This Important?

- We want to execute applications in parallel in multi-core systems → consolidate more and more
  - Cloud computing
  - Mobile phones

- We want to mix different types of applications together
  - those requiring QoS guarantees (e.g., video, pedestrian detection)
  - those that are important but less so
  - those that are less important

- We want the system to be controllable and high performance
Why the Disparity in Slowdowns?

Multi-Core Chip

Shared DRAM Memory System

CORE 1

L2 CACHE

INTERCONNECT

DRAM MEMORY CONTROLLER

CORE 2

L2 CACHE

DRAM Bank 0

DRAM Bank 1

DRAM Bank 2

DRAM Bank 3
Why the Disparity in Slowdowns?

Multi-Core Chip

Shared DRAM Memory System

unfairness
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)

Row address 0 → Row decoder

This view of a bank is an abstraction.

Internally, a bank consists of many cells (transistors & capacitors) and other structures that enable access to cells.
DRAM Controllers

- A row-conflict memory access takes significantly longer than a row-hit access

- Current controllers take advantage of this fact

- Commonly used scheduling policy (FR-FCFS) [Rixner 2000]*
  1. *Row-hit first:* Service row-hit memory accesses first
  2. *Oldest-first:* Then service older accesses first

- This scheduling policy aims to maximize DRAM throughput

The Problem

- Multiple applications share the DRAM controller
- DRAM controllers designed to maximize DRAM data throughput

- **DRAM scheduling policies are unfair to some applications**
  - Row-hit first: unfairly prioritizes apps with high row buffer locality
    - Threads that keep on accessing the same row
  - Oldest-first: unfairly prioritizes memory-intensive applications

- **DRAM controller vulnerable to denial of service attacks**
  - Can write programs to exploit unfairness
A Memory Performance Hog

STREAM
- Sequential memory access
- Very high row buffer locality (96% hit rate)
- Memory intensive

RANDOM
- Random memory access
- Very low row buffer locality (3% hit rate)
- Similarly memory intensive

What Does the Memory Hog Do?

Row size: 8KB, request size: 64B
128 (8KB/64B) requests of STREAM serviced before a single request of RANDOM

Now That We Know What Happens Underneath

- How would you solve the problem?

- What is the right place to solve the problem?
  - Programmer?
  - System software?
  - Compiler?
  - Hardware (Memory controller)?
  - Hardware (DRAM)?
  - Circuits?

- Two other goals of this course:
  - Enable you to think critically
  - Enable you to think broadly
For the Really Interested…


Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems

Thomas Moscibroda    Onur Mutlu
Microsoft Research
{moscitho, onur}@microsoft.com
Onur Mutlu and Thomas Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors" 
Proceedings of the 40th International Symposium on Microarchitecture (MICRO), pages 146-158, Chicago, IL, December 2007. Slides (ppt)

Onur Mutlu and Thomas Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems" 
Proceedings of the 35th International Symposium on Computer Architecture (ISCA) [Slides (ppt)]

Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda, "Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning" 
Proceedings of the 44th International Symposium on Microarchitecture (MICRO), Porto Alegre, Brazil, December 2011. Slides (pptx)
Takeaway I

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
Cooperation between multiple components and layers can enable more effective solutions and systems
Recap: Mysteries No Longer!

- Meltdown & Spectre (2017-2018)
- Rowhammer (2012-2014)
- Memories Forget: Refresh (2011-2012)
- Memory Performance Attacks (2006-2007)
Takeaways
Takeaways

- It is an exciting time to be understanding and designing computing architectures.

- Many challenging and exciting problems in platform design:
  - That no one has tackled (or thought about) before.
  - That can have huge impact on the world’s future.

- Driven by huge hunger for data (Big Data), new applications (ML/AI, graph analytics, genomics), ever-greater realism, ...
  - We can easily collect more data than we can analyze/understand.

- Driven by significant difficulties in keeping up with that hunger at the technology layer:
  - Five walls: Energy, reliability, complexity, security, scalability.
Computer Architecture as an Enabler of the Future
Assignment: Required Lecture Video

- Why study computer architecture? Why is it important?
- Future Computing Platforms: Challenges & Opportunities

Required Assignment
- Watch one of Prof. Mutlu’s lectures and analyze either (or both)
  - https://www.youtube.com/watch?v=kgiZlSOcGFM (May 2017)
  - https://www.youtube.com/watch?v=mskTeNnf-i0 (Feb 2021)

Optional Assignment – for 1% extra credit
- Write a 1-page summary of one of the lectures and email us
  - What are your key takeaways?
  - What did you learn?
  - What did you like or dislike?
  - Submit your summary to Moodle
Backup Slides For Your Benefit. Not Covered in Lecture.
Bloom Filters
Approximate Set Membership

- Suppose you want to quickly find out:
  - whether an element belongs to a set

- And, you can tolerate mistakes of the sort:
  - The element is actually not in the set, but you are incorrectly told that it is → false positive

- But, you cannot tolerate mistakes of the sort:
  - The element is actually in the set, but you are incorrectly told that it is not → false negative

- Example task: You want to quickly identify all Mobile Phone Model X owners among all possible people in the world
  - Perhaps you want to give them free replacement phones
Example Task

- **World population**
  - \(~8\) billion (and growing)
  - 1 bit per person to indicate Model X owner or not
  - \(2^{33}\) bits needed to represent the entire set accurately
    - 8 Gigabits → large storage cost, slow access

- **Mobile Phone Model X owner population**
  - Say 1 million (and growing)

- **Can we represent the Model X owner set approximately, using a much smaller number of bits?**
  - Record the ID’s of owners in a much smaller Bloom Filter
Example Task II

- DRAM row population
  - ~8 billion (and growing)
  - 1 bit per row to indicate Refresh-often or not
  - $2^{33}$ bits needed to represent the entire set accurately
    - 8 Gigabits $\rightarrow$ large storage cost, slow access

- Refresh-often population
  - Say 1 million

- Can we represent Refresh-often set approximately, using a much smaller number of bits?
  - Record the ID’s of Refresh-Often rows in a much smaller Bloom Filter
Bloom Filter

- [Bloom, CACM 1970]
- Probabilistic data structure that compactly represents set membership (presence or absence of element in a set)

- Non-approximate set membership: Use 1 bit per element to indicate absence/presence of each element from an element space of N elements

- Approximate set membership: use a much smaller number of bits and indicate each element’s presence/absence with a subset of those bits
  - Some elements map to the bits other elements also map to

- Operations: 1) insert, 2) test, 3) remove all elements

Example with 64–128ms bin:

Insert Row 1

Bloom Filter Operation Example

Example with 64-128ms bin:

| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Hash function 1
- Hash function 2
- Hash function 3

Row 1 present?
Yes
Bloom Filter Operation Example

Example with 64-128ms bin:

```
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
```

Hash function 1
Hash function 2
Hash function 3

Row 2 present? No
Bloom Filter Operation Example

Example with 64–128ms bin:

```
0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0
```

- **Hash function 1**
- **Hash function 2**
- **Hash function 3**

**Insert Row 4**
Bloom Filter Operation Example

Example with 64-128ms bin:

0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 1 1 0 1 0 1 0 1 0

Hash function 1
Hash function 2
Hash function 3

Row 5 present?
Yes (false positive)
Bloom Filters

Space/Time Trade-offs in Hash Coding with Allowable Errors

In such applications, it is envisaged that overall performance could be improved by using a smaller core resident hash area in conjunction with the new methods and, when necessary, by using some secondary and perhaps time-consuming test to "catch" the small fraction of errors associated with the new methods. An example is discussed which illustrates possible areas of application for the new methods.

Burton H. Bloom

In this paper trade-offs among certain computational factors in hash coding are analyzed. The paradigm problem considered is that of testing a series of messages one-by-one for membership in a given set of messages. Two new hash-coding methods are examined and compared with a particular conventional hash-coding method. The computational factors considered are the size of the hash area (space), the time required to identify a message as a nonmember of the given set (reject time), and an allowable error frequency.

Bloom Filters: Pros and Cons

- Advantages
  + Enables storage-efficient representation of set membership
  + Insertion and testing for set membership (presence) are fast
  + No false negatives: If Bloom Filter says an element is not present in the set, the element must not have been inserted
  + Enables tradeoffs between time & storage efficiency & false positive rate (via sizing and hashing)

- Disadvantages
  -- False positives: An element may be deemed to be present in the set by the Bloom Filter but it may never have been inserted
  
  Not the right data structure when you cannot tolerate false positives

Benefits of Bloom Filters as Refresh RateBins

- **False positives**: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Refresh some rows more frequently than needed

- **No false negatives**: rows are never refreshed less frequently than needed (no correctness problems)

- **Scalable**: a Bloom filter never overflows (unlike a fixed-size table)

- **Efficient**: No need to store info on a per-row basis; simple hardware → 1.25 KB for 2 filters for 32 GB DRAM system