Computer Architecture
Lecture 1: Introduction and Basics

Prof. Onur Mutlu
ETH Zurich
Fall 2017
20 September 2017
Question: What Is This?
Answer: The First Major Piece of a Famous Architect

- **Bahnhof Stadelhofen**: “The train station has several of the features that became signatures of his work; straight lines and right angles are rare.”

- **ETH Alumnus, Civil Engineering**

  **Santiago Calatrava Valls** (born 28 July 1951) is a Spanish architect, structural engineer, sculptor and painter, particularly known for his bridges supported by single leaning pylons, and his railway stations, stadiums, and museums, whose sculptural forms often resemble living organisms.[1] His best-known works include the Milwaukee Art Museum, the Turning Torso tower in Malmo, Sweden, the Margaret Hunt Hill Bridge in Dallas, Texas, and the Museum of Tomorrow in Rio de Janeiro,
Question 2: What Is This?

Answer: Masterpiece of a Famous Architect

Design  [edit]

Calatrava said that the Oculus resembles a bird being released from a child's hand. The roof was originally designed to mechanically open to increase light and ventilation to the enclosed space. Herbert Muschamp, architecture critic of The New York Times, compared the design to the Bethesda Terrace and Fountain in Central Park, and wrote in 2004:

Source: https://en.wikipedia.org/wiki/World_Trade_Center_station_(PATH)
Answer: Masterpiece of a Famous Architect

"Santiago Calatrava’s design for the World Trade Center PATH station should satisfy those who believe that buildings planned for ground zero must aspire to a spiritual dimension. Over the years, many people have discerned a metaphysical element in Mr. Calatrava’s work. I hope New Yorkers will detect its presence, too. With deep appreciation, I congratulate the Port Authority for commissioning Mr. Calatrava, the great Spanish architect and engineer, to design a building with the power to shape the future of New York. It is a pleasure to report, for once, that public officials are not overstating the case when they describe a design as breathtaking."[43]"
Design Constraints

However, Calatrava's original soaring spike design was scaled back because of security issues. The New York Times observed in 2005:

“In the name of security, Santiago Calatrava's bird has grown a beak. Its ribs have doubled in number and its wings have lost their interstices of glass.... [T]he main transit hall, between Church and Greenwich Streets, will almost certainly lose some of its delicate quality, while gaining structural expressiveness. It may now evoke a slender stegosaurus more than it does a bird.”[45]
Stegosaurus

From Wikipedia, the free encyclopedia

For the pachycephalosaurid of a similar name, see Stegoceras.

Stegosaurus (/stɛɡəˈsɔːrəs/[^1]) is a genus of armored dinosaur. Fossils of this genus date to the Late Jurassic period, where they are found in Kimmeridgian to early Tithonian aged strata, between 155 and 150 million years ago, in the western United States and Portugal. Several

Source: https://en.wikipedia.org/wiki/Stegosaurus

The Postcranial Skeleton of an Exceptionally Complete Individual of the Plated Dinosaur Stegosaurus stenops
Design Constraints: Noone is Immune

However, Calatrava's original soaring spike design was scaled back because of security issues. The *New York Times* observed in 2005:

“In the name of security, Santiago Calatrava’s bird has grown a beak. Its ribs have doubled in number and its wings have lost their interstices of glass.... [T]he main transit hall, between Church and Greenwich Streets, will almost certainly lose some of its delicate quality, while gaining structural expressiveness. It may now evoke a slender *stegosaurus* more than it does a bird."[45]

The design was further modified in 2008 to eliminate the opening and closing roof mechanism because of budget and space constraints.[46]

The Transportation Hub has been dubbed "the world's most expensive transportation hub" for its massive cost for reconstruction—$3.74 billion dollars.[48][58] By contrast, the proposed two-mile PATH extension
Question: What Is This?
Answer: Masterpiece of Another Famous Architect

Fallingwater

From Wikipedia, the free encyclopedia

Fallingwater or Kaufmann Residence is a house designed by architect Frank Lloyd Wright in 1935 in rural southwestern Pennsylvania, 43 miles (69 km) southeast of Pittsburgh. The home was built partly over a waterfall on Bear Run in the Mill Run section of Stewart Township, Fayette County, Pennsylvania, in the Laurel Highlands of the Allegheny Mountains.

*Time* cited it after its completion as Wright's "most beautiful job"; it is listed among *Smithsonian*’s Life List of 28 places "to visit before you die." It was designated a National Historic Landmark in 1966. In 1991, members of the American Institute of Architects named the house the "best all-time work of American architecture" and in 2007, it was ranked twenty-ninth on the list of America’s Favorite Architecture according to the AIA.
Your First Comp Arch Assignment

- Go and visit Bahnhof Stadelhofen
  - Extra credit: Repeat for Oculus
  - Extra+ credit: Repeat for Fallingwater

- Appreciate the beauty & out-of-the-box and creative thinking
- Think about tradeoffs in the design of the Bahnhof
  - Strengths, weaknesses, goals of design
- Derive principles on your own for good design and innovation

- Due date: **Any time during this course**
  - Later during the course is better
  - Apply what you have learned in this course
  - Think out-of-the-box
But First, Today’s First Assignment

- Find The Differences Of This and That
Find The Differences of This and That
This
That

Source: http://cookiemagik.deviantart.com/art/Train-station-207266944
Many Tradeoffs Between Two Designs

- You can list them after you complete the first assignment...
Aside: Evaluation Criteria for the Designs

- Functionality (Does it meet the specification?)
- Reliability
- Space requirement
- Cost
- Expandability
- Comfort level of users
- Happiness level of users
- Aesthetics
- ...

- How to evaluate goodness of design is always a critical question.
A Key Question

- How was Calatrava able to design especially his key buildings?
- Can have many guesses
  - (Ultra) hard work, perseverance, dedication (over decades)
  - Experience
  - Creativity, Out-of-the-box thinking
  - A good understanding of past designs
  - Good judgment and intuition
  - Strong skill combination (math, architecture, art, engineering, ...)
  - Funding ($$$), luck, initiative, entrepreneurialism
  - Strong understanding of and commitment to fundamentals
  - Principled design
  - ...

- (You will be exposed to and hopefully develop/enhance many of these skills in this course)
Principled Design

“To me, there are two overriding principles to be found in nature which are most appropriate for building:

- one is the optimal use of material,
- the other the capacity of organisms to change shape, to grow, and to move.”

*Santiago Calatrava*

“Calatrava's constructions are inspired by natural forms like plants, bird wings, and the human body.”

Source: http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/
Gare do Oriente, Lisbon, Revisited

Source: http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/
A Principled Design

Zoomorphic architecture

From Wikipedia, the free encyclopedia

Zoomorphic architecture is the practice of using animal forms as the inspirational basis and blueprint for architectural design. "While animal forms have always played a role adding some of the deepest layers of meaning in architecture, it is now becoming evident that a new strand of biomorphism is emerging where the meaning derives not from any specific representation but from a more general allusion to biological processes."[1]

Some well-known examples of Zoomorphic architecture can be found in the TWA Flight Center building in New York City, by Eero Saarinen, or the Milwaukee Art Museum by Santiago Calatrava, both inspired by the form of a bird’s wings.^[3]
What Does This Remind You Of?

A Quote from The Other Famous Architect

“architecture [...] based upon principle, and not upon precedent” (Frank Lloyd Wright)

Source: http://www.fallingwater.org/
Organic architecture is a philosophy of architecture which promotes harmony between human habitation and the natural world through design approaches so sympathetic and well integrated with its site, that buildings, furnishings, and surroundings become part of a unified, interrelated composition.

A well-known example of organic architecture is Fallingwater, the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy dialog with the rushing water and the steep site. The horizontal striations of stone masonry with daring cantilevers of colored beige concrete blend with native rock outcroppings and the wooded environment.
Another View

Source: https://roadtrippers.com/stories/falling-water
Yet Another View

Major High-Level Goals of This Course

- Understand the principles
- Understand the precedents

Based on such understanding:
  - Enable you to evaluate tradeoffs of different designs and ideas
  - Enable you to develop principled designs
  - Enable you to develop novel, out-of-the-box designs

The focus is on:
  - Principles, precedents, and how to use them for new designs

In Computer Architecture
Role of the (Computer) Architect

Role of the Architect

-- Look Backward (Examine old code)
-- Look forward (Listen to the dreamers)
-- Look Up (Nature of the problems)
-- Look Down (Predict the future of technology)

from Yale Patt’s lecture notes
Role of The (Computer) Architect

- **Look backward (to the past)**
  - Understand tradeoffs and designs, upsides/downsides, past workloads. Analyze and evaluate the past.

- **Look forward (to the future)**
  - Be the dreamer and create new designs. Listen to dreamers.
  - Push the state of the art. Evaluate new design choices.

- **Look up (towards problems in the computing stack)**
  - Understand important problems and their nature.
  - Develop architectures and ideas to solve important problems.

- **Look down (towards device/circuit technology)**
  - Understand the capabilities of the underlying technology.
  - Predict and adapt to the future of technology (you are designing for N years ahead). Enable the future technology.
Takeaways

- Being an architect is not easy
- You need to consider many things in designing a new system + have good intuition/insight into ideas/tradeoffs

- But, it is fun and can be very technically rewarding
- And, enables a great future
  - E.g., many scientific and everyday-life innovations would not have been possible without architectural innovation that enabled very high performance systems
  - E.g., your mobile phones

- This course will teach you how to become a good computer architect
So, I Hope You Are Here for This

- How does an assembly program end up executing as digital logic?
- What happens in-between?
- How is a computer designed using logic gates and wires to satisfy specific goals?

“C” as a model of computation
Programmer’s view of how a computer system works

Architect/microarchitect’s view: How to design a computer that meets system design goals.

*Choices critically affect both the SW programmer and the HW designer*

HW designer’s view of how a computer system works
Digital logic as a model of computation
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:
  - [http://www.cs.virginia.edu/~robins/YouAndYourResearch.html](http://www.cs.virginia.edu/~robins/YouAndYourResearch.html)
Levels of Transformation, Revisited

- A user-centric view: computer designed for users

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Program/Language</th>
<th>Runtime System (VM, OS, MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>Microarchitecture</td>
<td>Logic</td>
<td>Devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Electrons</td>
</tr>
</tbody>
</table>

- 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?
Crossing the Abstraction Layers

- Two key goals of this course 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
An Example: 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 Memory Performance Hog

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: Rephrased & Concise

- 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?
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

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

Row address 0

Column address 85

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

// initialize large arrays A, B
for (j=0; j<N; j++) {
    index = j*linesize; /* streaming */
    A[index] = B[index]; (in sequence)
    ...
}

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 \((8\text{KB}/64\text{B})\) 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
Reading on Memory Performance Attacks


- One potential reading for your Homework 1 assignment

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

Thomas Moscibroda    Onur Mutlu
Microsoft Research
{moscitho,onur}@microsoft.com
If You Are Interested … Further Readings


- 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)]

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
Another Example

- DRAM Refresh
DRAM in the System

Multi-Core Chip

*Die photo credit: AMD Barcelona
A DRAM cell consists of a capacitor and an access transistor. It stores data in terms of charge in 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 8 ExaByte DRAM ($2^{63}$ bytes)
- Assume a row size of 8 KiloBytes ($2^{13}$ 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...
- 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 consumption, verification complexity, cost

- How do we determine this profile information?
  - Also, who determines it?
Retention Time of DRAM Rows

- **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
   - >256ms

   1.25KB storage in controller for 32GB DRAM memory

   → check the bins to determine refresh rate of a row

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 on RAIDR

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

- One potential reading for your Homework 1 assignment

RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu    Ben Jaiyen    Richard Veras    Onur Mutlu
Carnegie Mellon University
{jamiel,bjaiyen,rveras,onur}@cmu.edu
If You Are 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 1

Breaking the abstraction layers (between components and transformation hierarchy levels)

and knowing what is underneath enables you to understand and solve problems
Takeaway 2

Cooperation between multiple components and layers can enable more effective solutions and systems.
"Good ideas are a dime a dozen"

"Making them work is oftentimes the real contribution"
Recall: RAIDR: Mechanism

1. Profiling: Identify the retention time of all DRAM rows
   ➔ can be done at design time or during operation

2. Binning: Store rows into bins by retention time
   ➔ use Bloom Filters for efficient and scalable storage
   1.25KB storage in controller for 32GB DRAM memory

3. Refreshing: Memory controller refreshes rows in different bins at different rates
   ➔ check the bins to determine refresh rate of a row

1. Profiling

To profile a row:
1. Write data to the row
2. Prevent it from being refreshed
3. Measure time before data corruption

<table>
<thead>
<tr>
<th>Time</th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>111111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 64 ms</td>
<td>111111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>110111111... (64–128ms)</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 256 ms</td>
<td>11111011... (128–256ms)</td>
<td>111111111... (&gt;256ms)</td>
<td>111111111...</td>
</tr>
</tbody>
</table>
Q: Is it really this easy?

A: Ummm, not really...
Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time
- Variable Retention Time (VRT) phenomenon
An Example VRT Cell

A cell from E 2Gb chip family
VRT: Implications on Profiling Mechanisms

- **Problem 1:** There does not seem to be a way of determining if a cell exhibits VRT without actually observing a cell exhibiting VRT
  - VRT is a memoryless random process [Kim+ JJAP 2010]

- **Problem 2:** VRT complicates retention time profiling by DRAM manufacturers
  - Exposure to very high temperatures can induce VRT in cells that were not previously susceptible
    - can happen during soldering of DRAM chips
    - manufacturer’s retention time profile may not be accurate

- **One option for future work:** Use ECC to continuously profile DRAM online while aggressively reducing refresh rate
  - Need to keep ECC overhead in check
More on DRAM Retention Analysis


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
2. Binning

- How to efficiently and scalably store rows into retention time bins?
- Use Hardware Bloom Filters [Bloom, CACM 1970]
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

Bloom Filter Operation Example

Example with 64-128ms bin:

Hash function 1

Hash function 2

Hash function 3

Insert Row 1

Bloom Filter Operation Example

Example with 64-128ms bin:

| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 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:

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

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 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 Rate Bins

- **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
Use of Bloom Filters in Hardware

- Useful when you can tolerate false positives in set membership tests

- See the following recent examples for clear descriptions of how Bloom Filters are used
3. Refreshing (RAIDR Refresh Controller)

- Choose a refresh candidate row
- Determine which bin the row is in
- Determine if refreshing is needed
3. Refreshing (RAIDR Refresh Controller)

Memory controller chooses each row as a refresh candidate every 64ms

Row in 64-128ms bin? —— Row in 128-256ms bin?
(First Bloom filter: 256B) (Second Bloom filter: 1KB)

Refresh the row
Every other 64ms window, refresh the row
Every 4th 64ms window, refresh the row

RAIDR: Baseline Design

Refresh control is in DRAM in today’s auto-refresh systems.
RAIDR can be implemented in either the controller or DRAM.
Overhead of RAIDR in DRAM controller:
1.25 KB Bloom Filters, 3 counters, additional commands issued for per-row refresh (all accounted for in evaluations)
Overhead of RAIDR in DRAM chip:
Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)
Total overhead: 1.25KB Bloom Filters, 64 counters (32 GB DRAM)
RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; SPEC, TPC-C, TPC-H 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

![Energy per access graph](image)

![Weighted speedup graph](image)
DRAM Refresh: More Questions

- What else can you do to reduce the impact of refresh?

- What else can you do if you know the retention times of rows?

- How can you accurately measure the retention time of DRAM rows?

- Recommended reading:
We Will Dig Deeper More
In This Course

“Good ideas are a dime a dozen”

“Making them work is oftentimes the real contribution”
Yet Another Example

- DRAM Row Hammer (or, DRAM Disturbance Errors)
- How a simple hardware failure mechanism can create a widespread system security vulnerability
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
$1.0 \times 10^7$
errors

B company

83%
(45/54)

Up to
$2.7 \times 10^6$
errors

C company

88%
(28/32)

Up to
$3.3 \times 10^5$
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 row hammer happens enough times, charge in such cells gets drained
Higher-Level Implications

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

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

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

1. Avoid *cache hits* – Flush \( X \) from cache

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

Download from: https://github.com/CMU-SAFARI/rowhammer
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

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

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

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

Download from: [https://github.com/CMU-SAFARI/rowhammer](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 Security Attack Example

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

- We tested a selection of laptops and found that a subset of them exhibited the problem.

- We built two working privilege escalation exploits that use this effect.
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

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

Rowhammer

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after.
More Security Implications

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

Not there yet, but ...

ROOT privileges for web apps!

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

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

“Can gain control of a smart phone deterministically”

Hammer And Root

Millions of Androids

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/
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


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
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
Some Thoughts on RowHammer

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

- How to 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

More 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](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](https://github.com/google/rowhammer-test)
  - Double-sided Rowhammer
More on RowHammer (II)

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

One potential reading for your Homework 1 assignment
Retrospective on RowHammer & Future

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

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

- 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, 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
  - Three walls: Energy, reliability, complexity
Increasingly Demanding Applications

- Dream, and they will come
Increasingly Diverging/Complex Tradeoffs

Communication Dominates Arithmetic

- 64-bit DP 20pJ
- 256-bit buses
- 256-bit access 8 kB SRAM
- 20mm
- 26 pJ
- 256 pJ
- 16 nJ
- 500 pJ
- 50 pJ
- 1 nJ

DRAM Rd/Wr
Efficient off-chip link

Dally, HiPEAC 2015
Increasingly Diverging/Complex Tradeoffs

Communication Dominates Arithmetic

A memory access consumes $\sim 1000$X the energy of a complex addition
Increasingly Complex Systems

Past systems

- Microprocessor
- Main Memory
- Storage (SSD/HDD)
Recap: Some Goals of This Course

- **Teach/enable/empower you to:**
  - Understand how a computing platform works
  - Understand how decisions made in hardware affect the software/programmer as well as hardware designer
  - Think critically (in solving problems)
  - Think broadly across the levels of transformation
  - Understand how to analyze and make tradeoffs in design
  - Apply the above in several design/research projects and HWs
Course Info: Who Are We?

- **Onur Mutlu**
  - Professor @ ETH Zurich CS, since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-...
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - [https://people.inf.ethz.ch/omutlu/](https://people.inf.ethz.ch/omutlu/)
  - [omutlu@gmail.com](mailto:omutlu@gmail.com) (Best way to reach me)
  - Office hours: TBD

- **Research and Teaching in:**
  - Computer architecture, computer systems, bioinformatics
  - Memory and storage systems
  - Hardware security
  - Fault tolerance
  - Hardware/software cooperation
  - ...

139
Course Info: Who Are We?

- **Teaching Assistants**
  - Hasan Hassan
  - Juan Gomez Luna
  - Lois Orosa
  - Mohammad Sadr
  - Arash Tavakkol

- Get to know them and their research
Review: Major High-Level Goals of This Course

- Understand the principles
- Understand the precedents

Based on such understanding:
- Enable you to evaluate tradeoffs of different designs and ideas
- Enable you to develop principled designs
- Enable you to develop novel, out-of-the-box designs

The focus is on:
- Principles, precedents, and how to use them for new designs

In Computer Architecture