Digital Design & Computer Arch.

Lecture 22: Memory Overview, Organization & Technology

Prof. Onur Mutlu

ETH Zürich

Spring 2021

21 May 2021

Readings for This Lecture and Next

- Memory Hierarchy and Caches

- Required
  - H&H Chapters 8.1-8.3
  - Refresh: P&P Chapter 3.5

- Recommended
  - An early cache paper by Maurice Wilkes
    - Wilkes, "Slave Memories and Dynamic Storage Allocation," IEEE Trans. On Electronic Computers, 1965.

Extra Assignment 3: Amdahl's Law

- **Paper review**
  - G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," AFIPS 1967.

- **Optional Assignment – for 1% extra credit**
  - **Write a 1-page review**
  - Upload PDF file to Moodle – Deadline: June 15

- I strongly recommend that you follow my guidelines for (paper) review (see next slide)

We Are **Done** With This…

- Single-cycle Microarchitectures
- Multi-cycle and Microprogrammed Microarchitectures
- Pipelining
- Issues in Pipelining: Control & Data Dependence Handling, State Maintenance and Recovery, …
- Out-of-Order Execution
- Other Execution Paradigms

Approaches to (Instruction-Level) Concurrency

- Pipelining
- Fine-Grained Multithreading
- Out-of-order Execution
- Dataflow (at the ISA level)
- Superscalar Execution
- VLIW
- Systolic Arrays
- Decoupled Access Execute
- SIMD Processing (Vector and Array processors, GPUs)

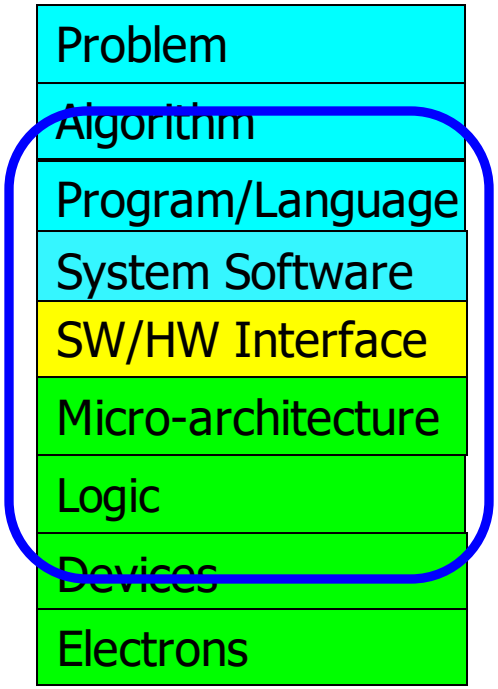
Now you are very familiar with many processing paradigms

Approaches to (Instruction-Level) Concurrency

- Pipelining
- Fine-Grained Multithreading
- Out-of-order Execution
- Dataflow (at the ISA level)
- Superscalar Execution
- VLIW
- Systolic Arrays
- Decoupled Access Execute
- SIMD Processing (Vector and Array processors, GPUs)

Food for thought:
tradeoffs of these different processing paradigms

Tradeoffs of Processing Paradigms



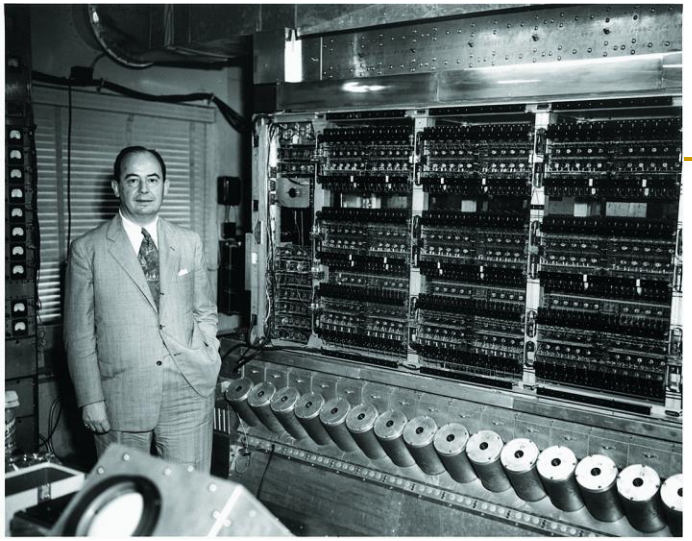
Food for thought:

tradeoffs of these different processing paradigms

Let Us Now Take A Step Back

A Computing System

- Three key components
- Computation
- Communication
- Storage/memory



Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.

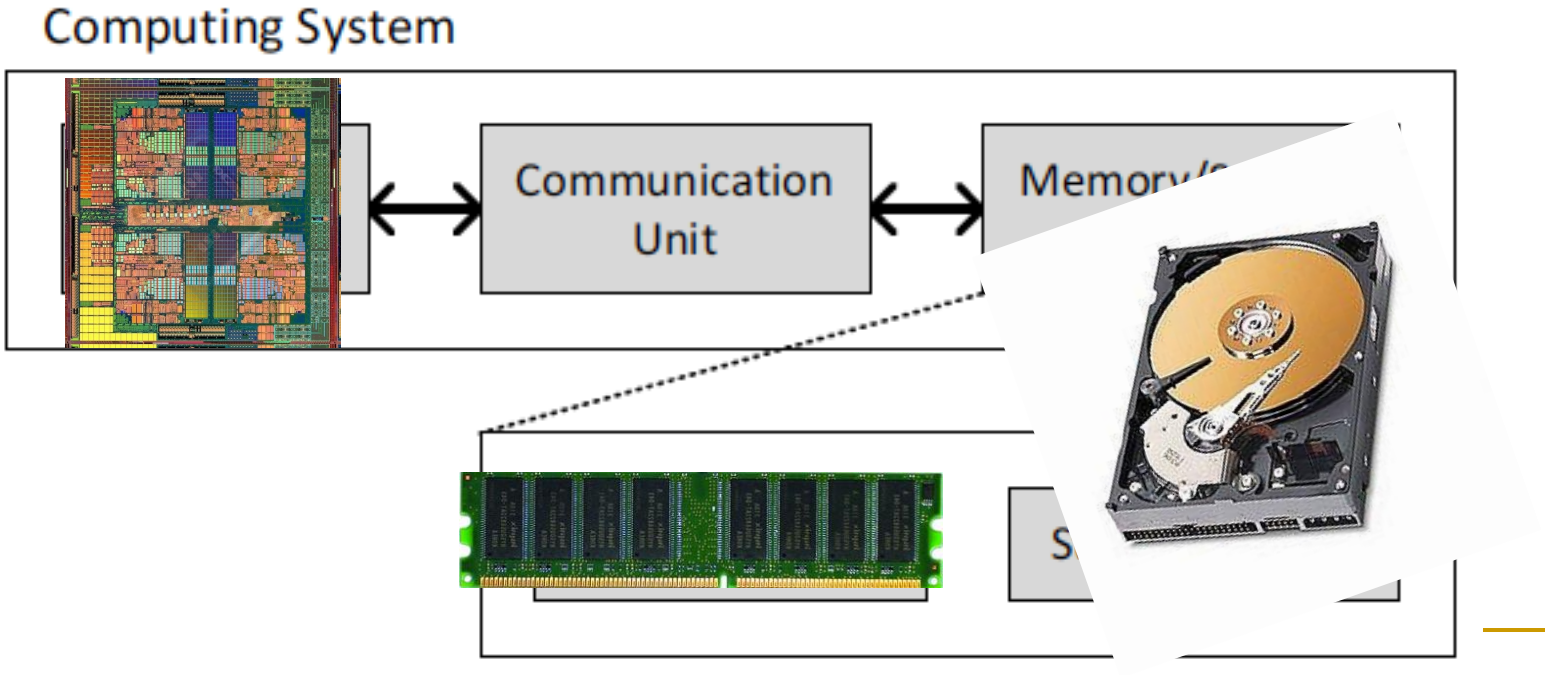
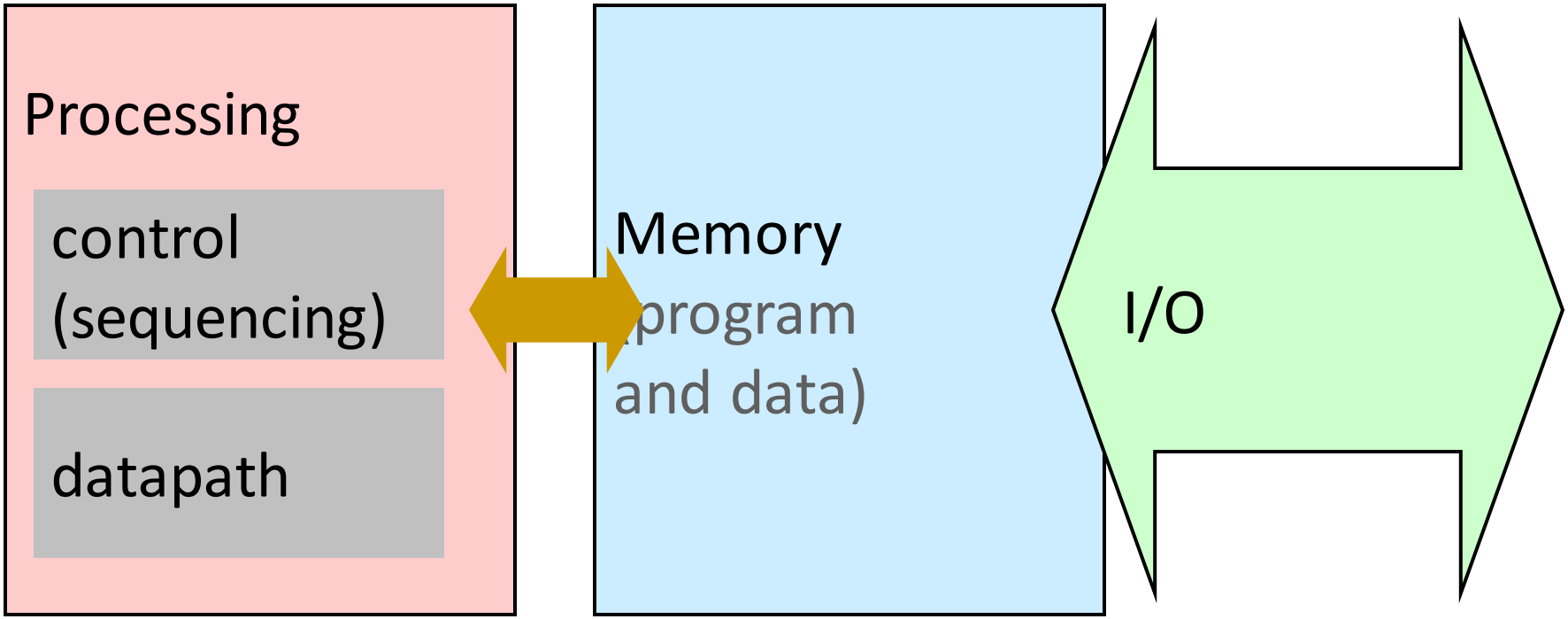


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

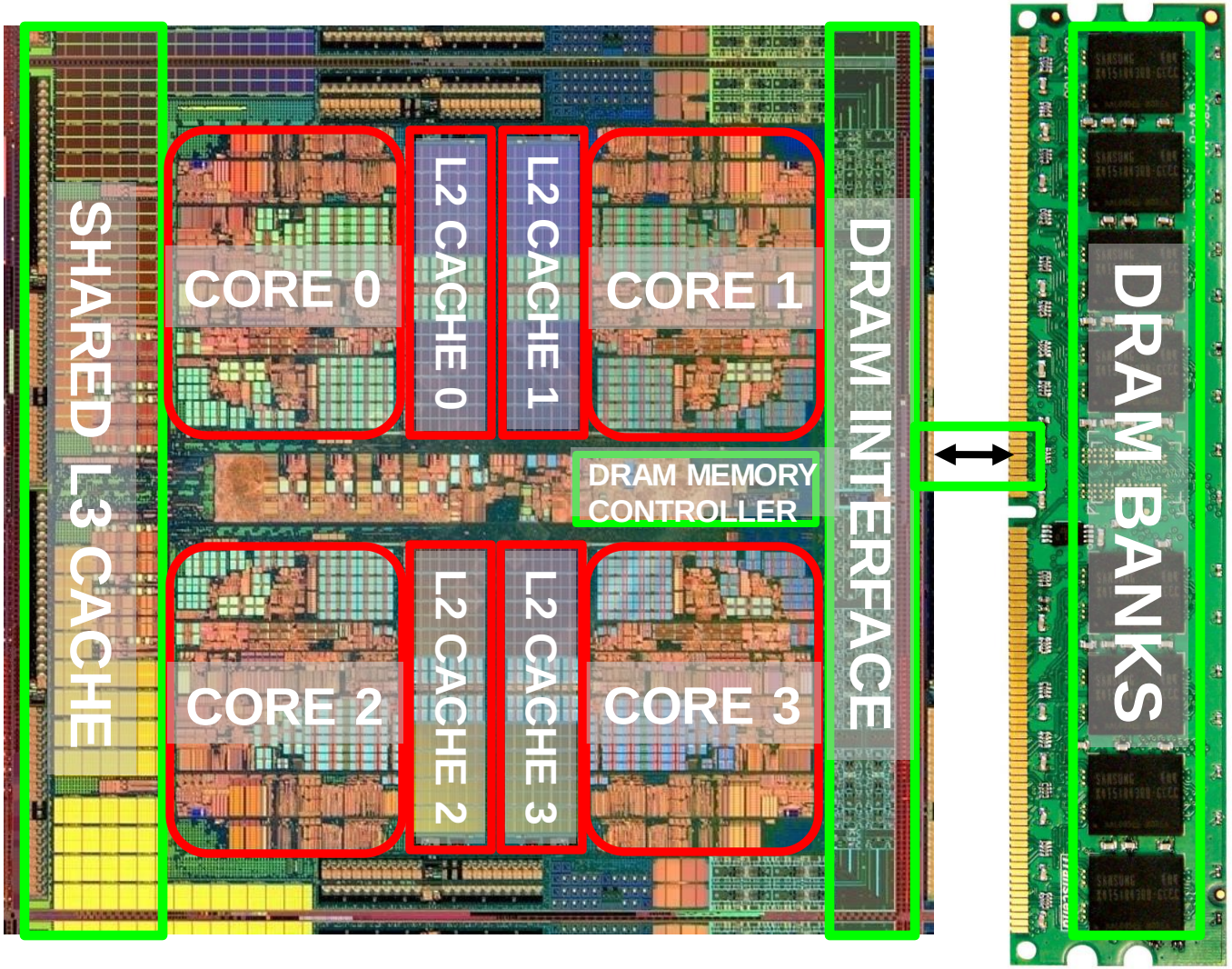
What is A Computer?

- We will cover all three components

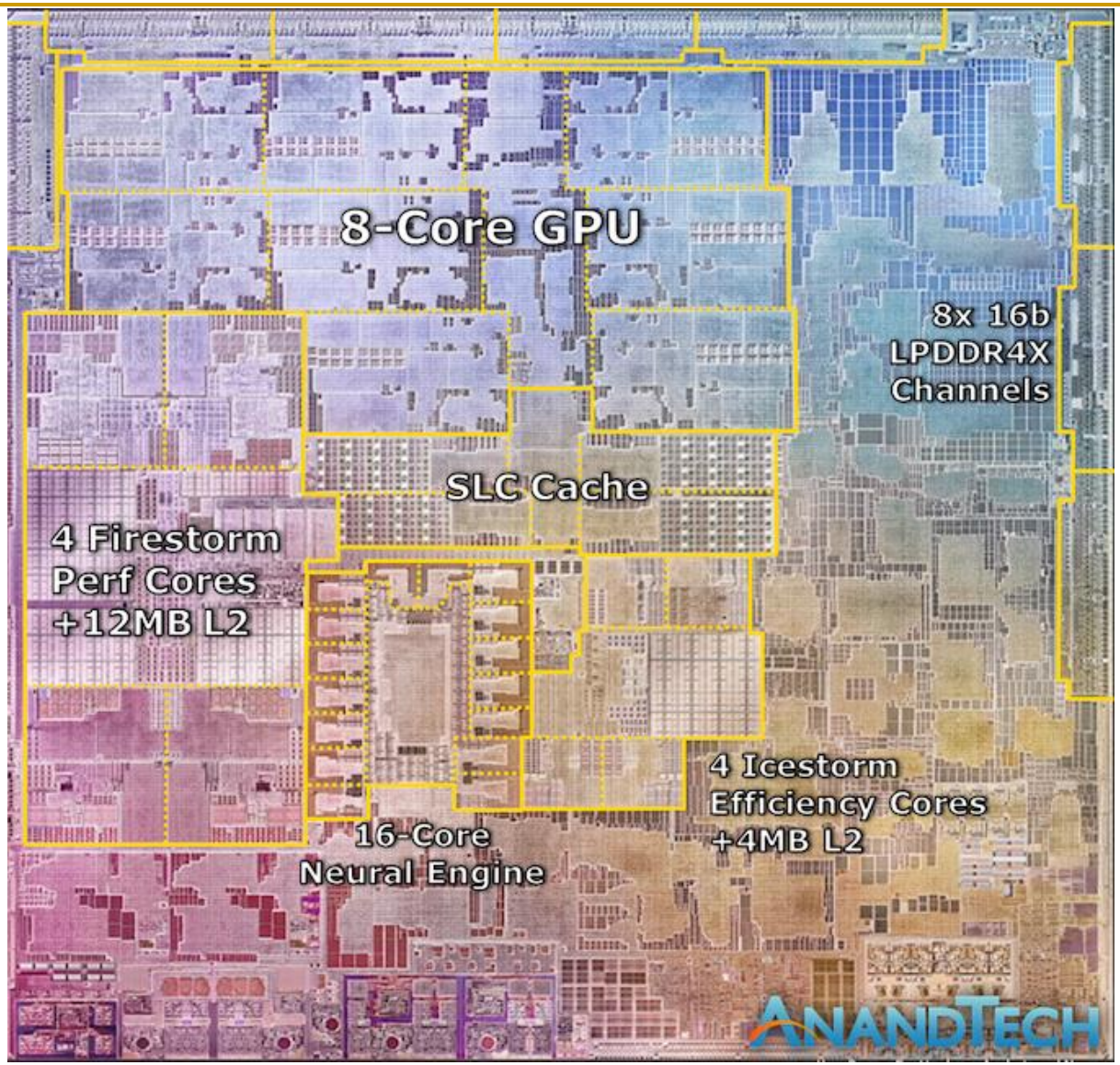


Memory Is Very Important

Memory in a Modern System

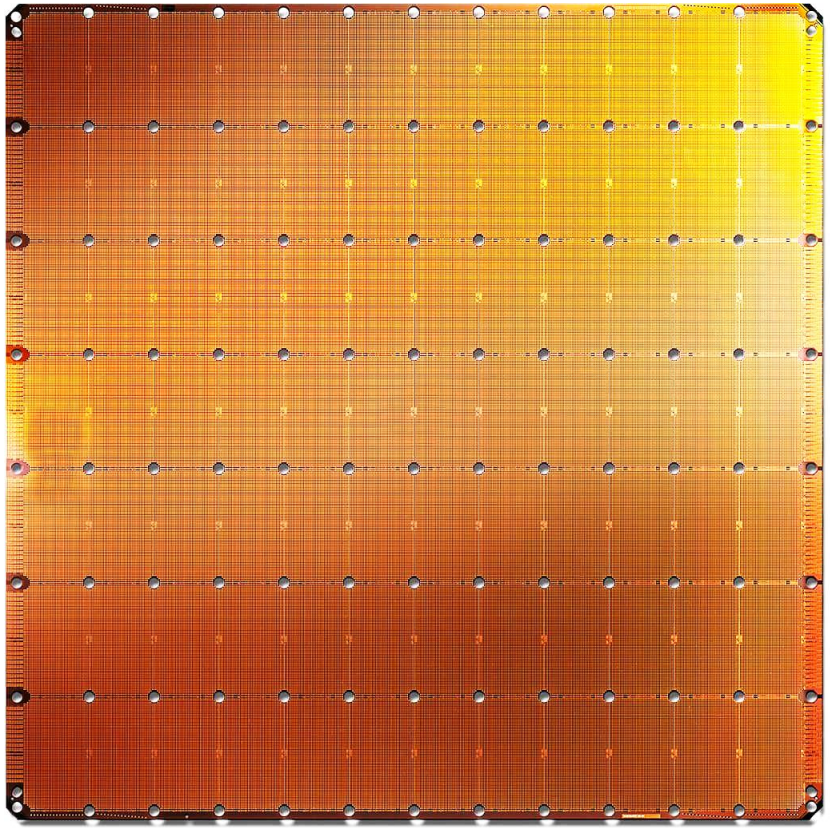


A Large Fraction of Modern Chips is Memory



Source: https://www.anandtech.com/show/16252/mac-mini-apple-m1-tested

Cerebras's Wafer Scale Engine (2019)



- The largest ML accelerator chip
- 400,000 cores
- **18 GB of on-chip memory**
- **9 PB/s memory bandwidth**

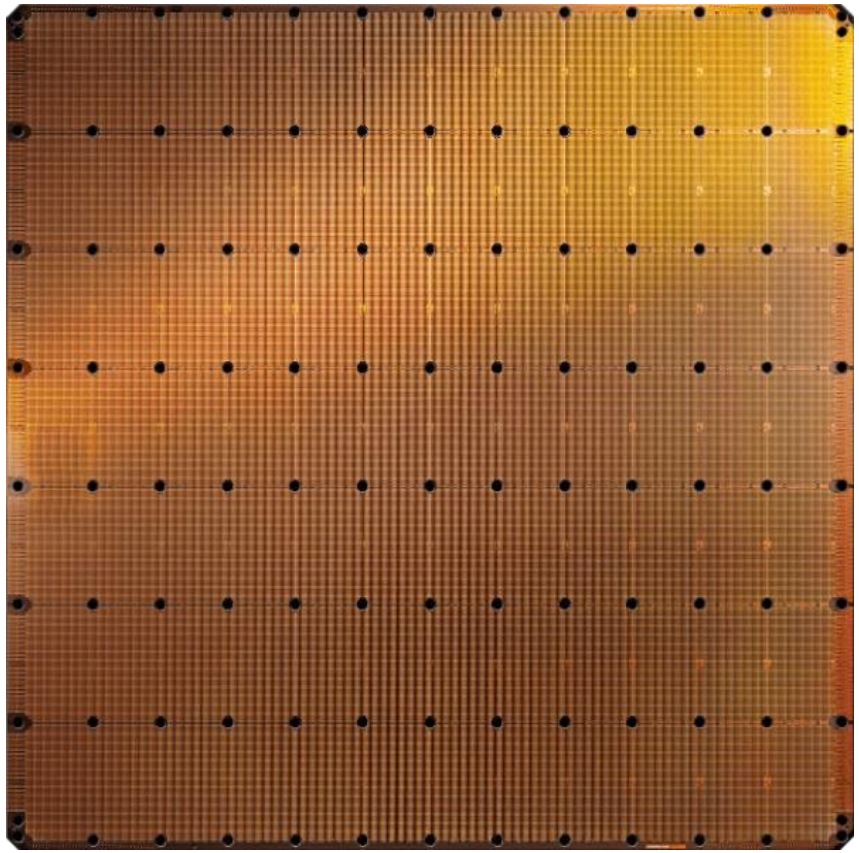
Cerebras WSE
1.2 Trillion transistors
46,225 mm^2

Largest GPU
21.1 Billion transistors
815 mm^2
NVIDIA TITAN V

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Cerebras's Wafer Scale Engine-2 (2021)



- The largest ML accelerator chip
- 850,000 cores
- **40 GB of on-chip memory**
- **20 PB/s memory bandwidth**

Cerebras WSE-2
2.6 Trillion transistors
46,225 mm^2

Largest GPU
54.2 Billion transistors
826 mm^2
NVIDIA Ampere GA100

https://cerebras.net/product/#overview

Memory is Critical for Performance

- We have seen it many times in this course

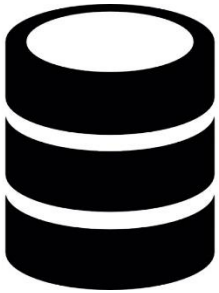
- Load-related stalls in pipelining
  - Even with magic "1-cycle" memory assumption
- Load/store handling in OoO execution processors
- OoO execution and memory latency tolerance
- VLIW stalls due to long-latency memory operations
- VLIW memory bank disambiguation
- Many memory banks needed in SIMD processors
  - SIMD vector processing performance example
- GPU register files and memory systems
- Fine-grained multithreading to tolerate memory latency
- …

Computation is Bottlenecked by Memory

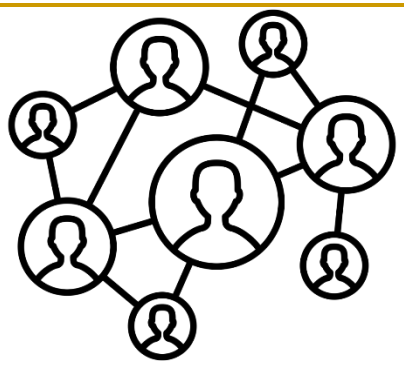
- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
  - We can generate more than we can process

Application Perspective

Memory Is Critical for Performance (I)

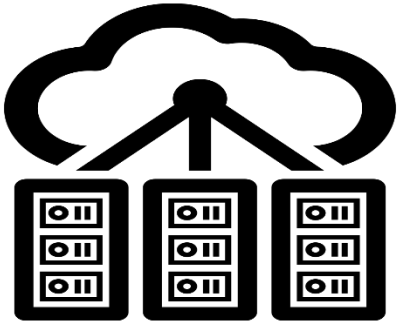


In-memory Databases
[Mao+, EuroSys'12;
Clapp+ (**Intel**), IISWC'15]



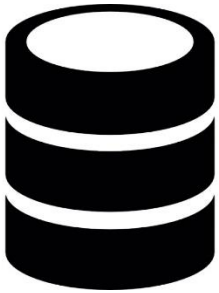
Graph/Tree Processing
[Xu+, IISWC'12; Umuroglu+, FPL'15]

In-Memory Data Analytics
[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]

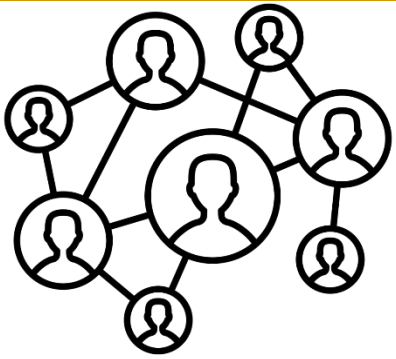


Datacenter Workloads
[Kanev+ (**Google**), ISCA'15]

Memory Is Critical for Performance (I)



In-memory Databases

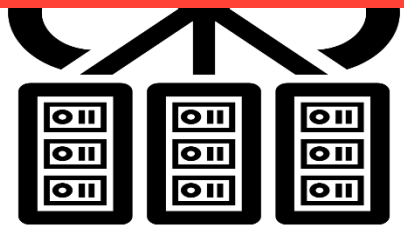


Graph/Tree Processing

Memory → bottleneck

In-Memory Data Analytics

[Clapp+ (**Intel**), IISWC'15; Awan+, BDCloud'15]



Datacenter Workloads

[Kanev+ (**Google**), ISCA'15]

Memory Is Critical for Performance (II)

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

VP9 YouTube

Video Playback

Google's **video codec**

VP9 YouTube

Video Capture

Google's **video codec**

Memory Is Critical for Performance (II)

Chrome

TensorFlow Mobile

Memory → bottleneck

VP9 YouTube

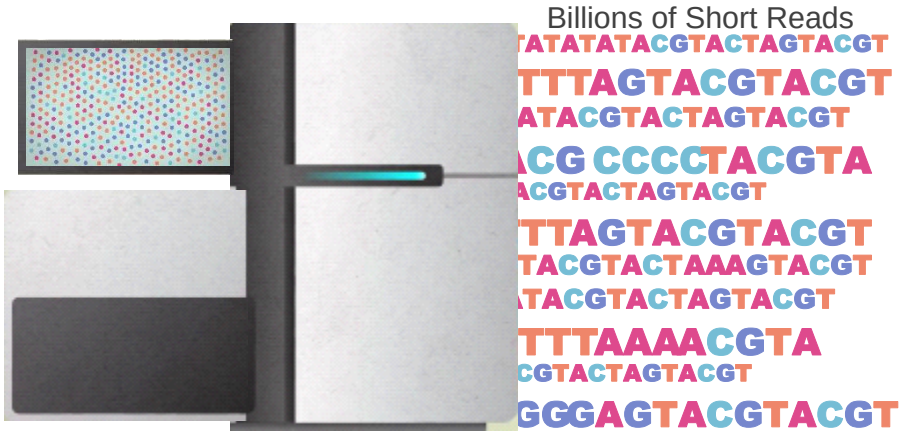
Video Playback

Google's **video codec**

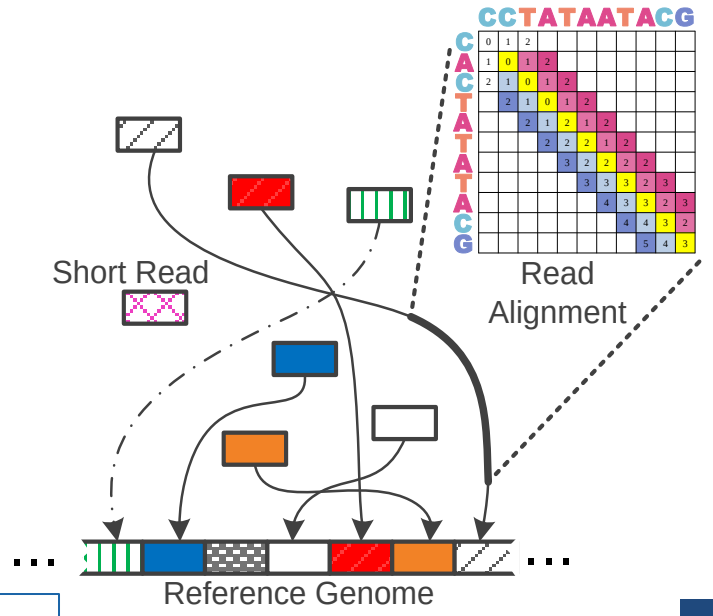
VP9 YouTube

Video Capture

Google's **video codec**



1 Sequencing



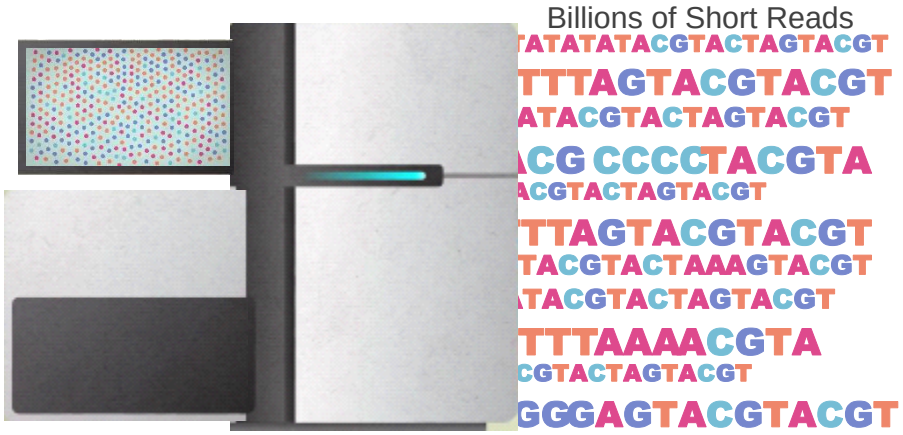
Read Mapping 2

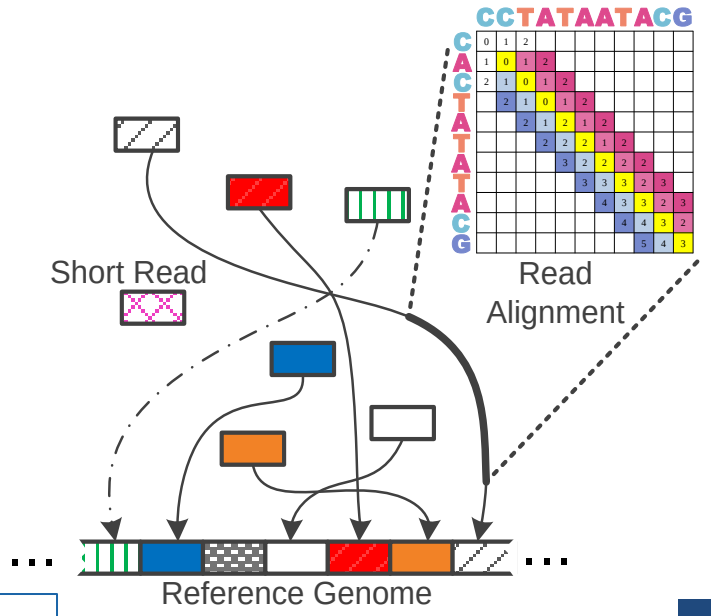
Genome Analysis

```
reference: TTTATCGCTTCCATGACGCAG
read1:       ATCGCATCC
read2:      TATCGCATC
read3:           CATCCATGA
read4:         CGCTTCCAT
read5:               CCATGACGC
read6:             TTCCATGAC
```

3 Variant Calling

Scientific Discovery 4





Memory → bottleneck

```
reference: TTTATCGCTTCCATGACGCAG
read1:       ATCGCATCC
read2:      TATCGCATC
read3:           CATCCATGA
read4:         CGCTTCCAT
read5:               CCATGACGC
read6:             TTCCATGAC
```

3 Variant Calling

Scientific Discovery 4

New Genome Sequencing Technologies

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 **Article history** ▾

Oxford Nanopore MinION

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions**," Briefings in Bioinformatics, 2018.
[Open arxiv.org version]

New Genome Sequencing Technologies

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 **Article history** ▾

Oxford Nanopore MinION

Memory → bottleneck

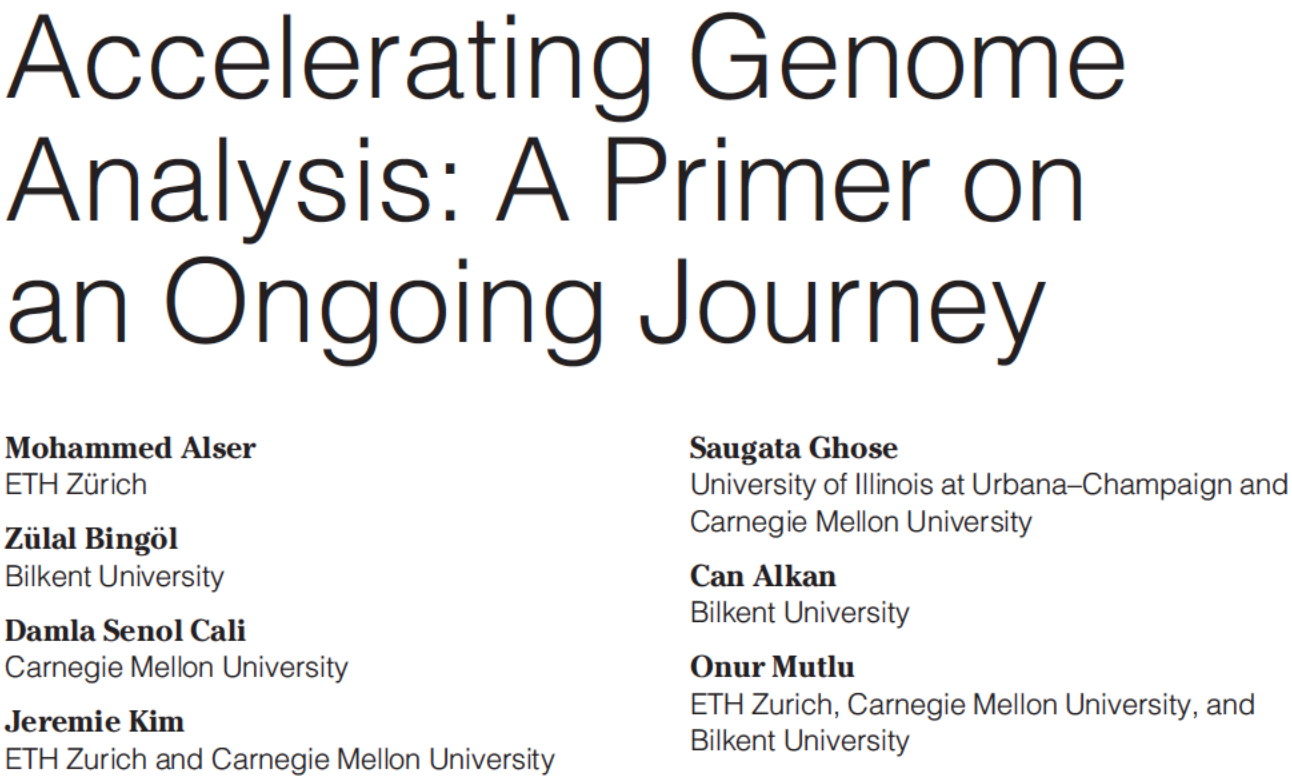
Future of Genome Sequencing & Analysis

MinION from ONT

SmidgION from ONT

Accelerating Genome Analysis

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  IEEE Micro (***IEEE MICRO***), Vol. 40, No. 5, pages 65-75, September/October 2020.
  [Slides (pptx)(pdf)]
  [Talk Video (1 hour 2 minutes)]



More on Fast & Efficient Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]

Detailed Lectures on Genome Analysis

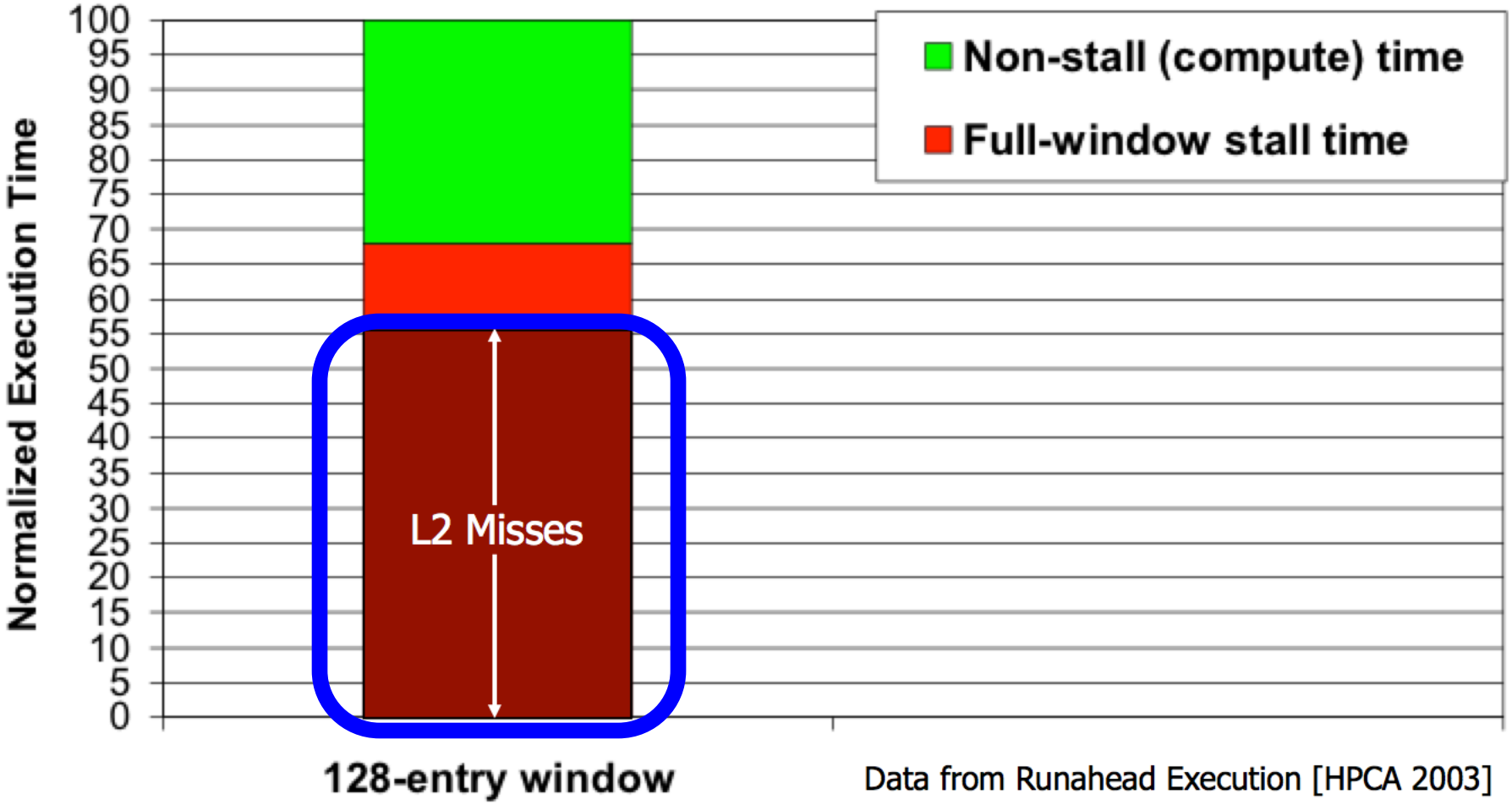
- Computer Architecture, Fall 2020, Lecture 3a
  - **Introduction to Genome Sequence Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5
- Computer Architecture, Fall 2020, Lecture 8
  - **Intelligent Genome Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14
- Computer Architecture, Fall 2020, Lecture 9a
  - **GenASM: Approx. String Matching Accelerator** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15
- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - **Accelerating Genomics** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId

Performance Perspective

Memory Bottleneck

I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.

- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)



The Performance Perspective

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt,
  "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"
  *Proceedings of the 9th International Symposium on High-Performance Computer Architecture (**HPCA**)*, pages 129-140, Anaheim, CA, February 2003. Slides (pdf)
  One of the 15 computer arch. papers of 2003 selected as Top Picks by IEEE Micro. HPCA Test of Time Award (awarded in 2021).
  [Lecture Slides (pptx) (pdf)]
  [Lecture Video (1 hr 54 mins)]
  [Retrospective HPCA Test of Time Award Talk Slides (pptx) (pdf)]
  [Retrospective HPCA Test of Time Award Talk Video (14 minutes)]

Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu § Jared Stark † Chris Wilkerson ‡ Yale N. Patt §

§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

†Microprocessor Research
Intel Labs
jared.w.stark@intel.com

‡Desktop Platforms Group
Intel Corporation
chris.wilkerson@intel.com

The Memory Bottleneck

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, **"Runahead Execution: An Effective Alternative to Large Instruction Windows"** *IEEE Micro, Special Issue: Micro's Top Picks from Microarchitecture Conferences* (***MICRO TOP PICKS***), Vol. 23, No. 6, pages 20-25, November/December 2003.

RUNAHEAD EXECUTION: AN EFFECTIVE ALTERNATIVE TO LARGE INSTRUCTION WINDOWS

The Memory Bottleneck

RICHARD SITES

It's the Memory, Stupid!

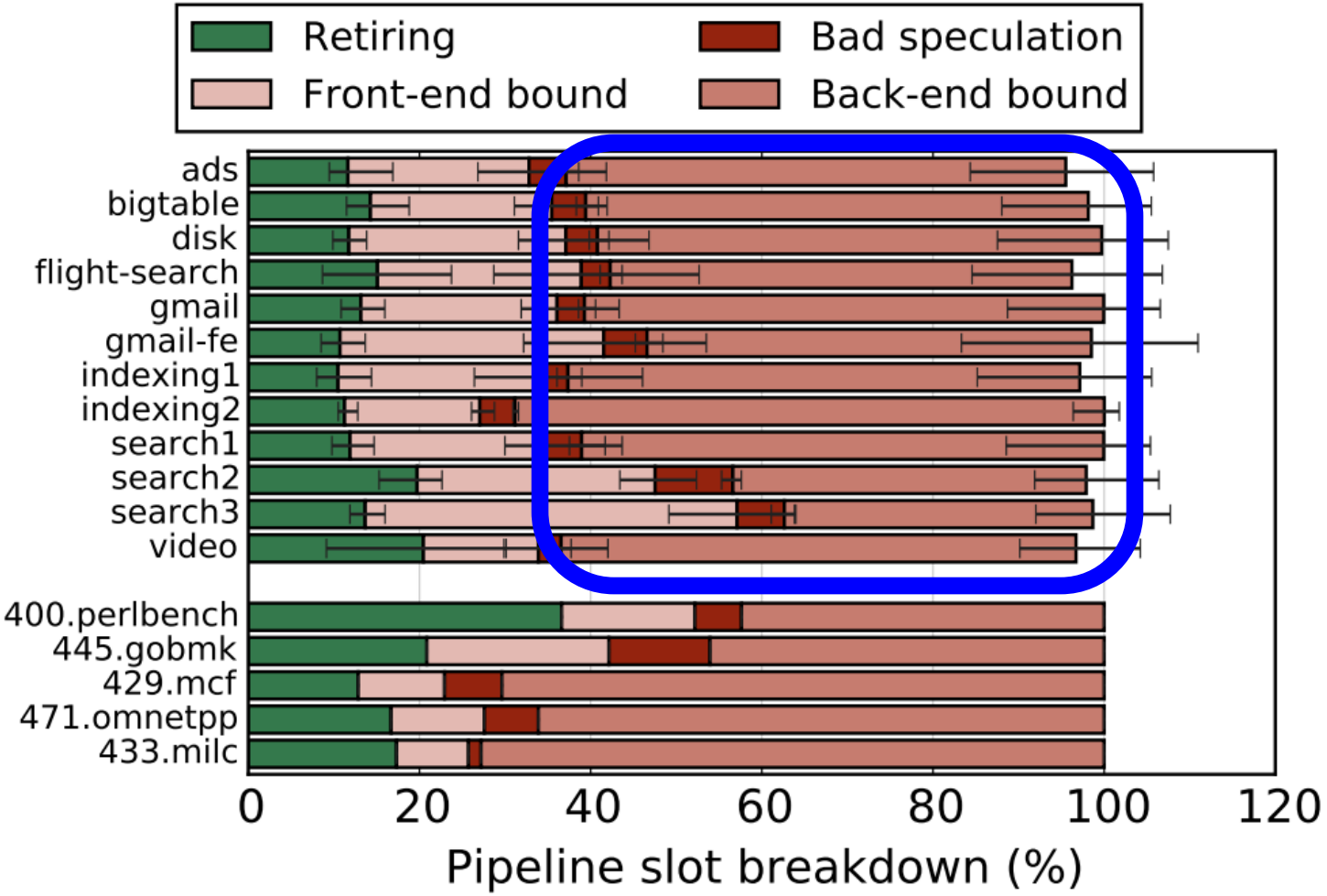
When we started the Alpha architecture design in 1988, we estimated a 25-year lifetime and a relatively modest 32% per year compounded performance improvement of implementations over that lifetime (1,000× total). We guestimated about 10× would come from CPU clock improvement, 10× from multiple instruction issue, and 10× from multiple processors.

5, 1996  MICROPROCESSOR REPORT

http://cva.stanford.edu/classes/cs99s/papers/architects_look_to_future.pdf

The Memory Bottleneck

- All of Google's Data Center Workloads (2015):



Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

The Memory Bottleneck

- All of Google's Data Center Workloads (2015):

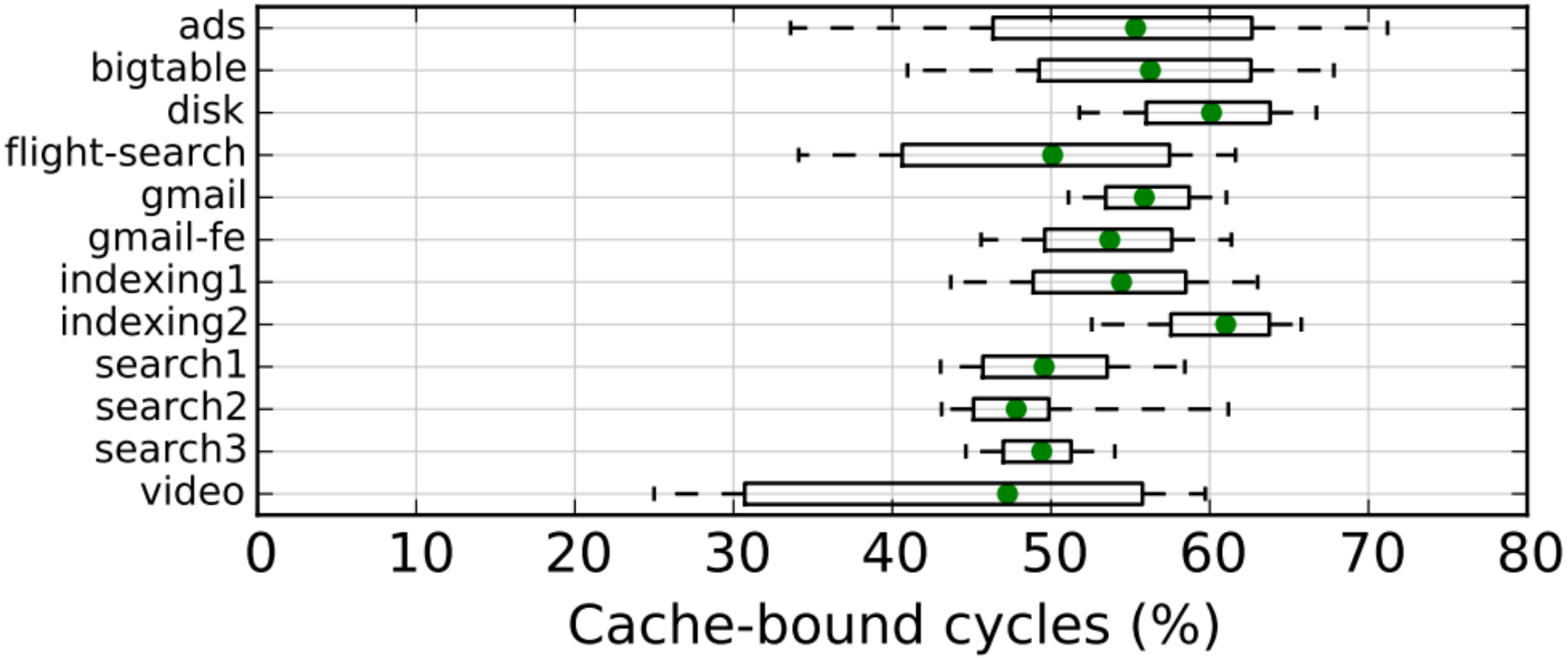
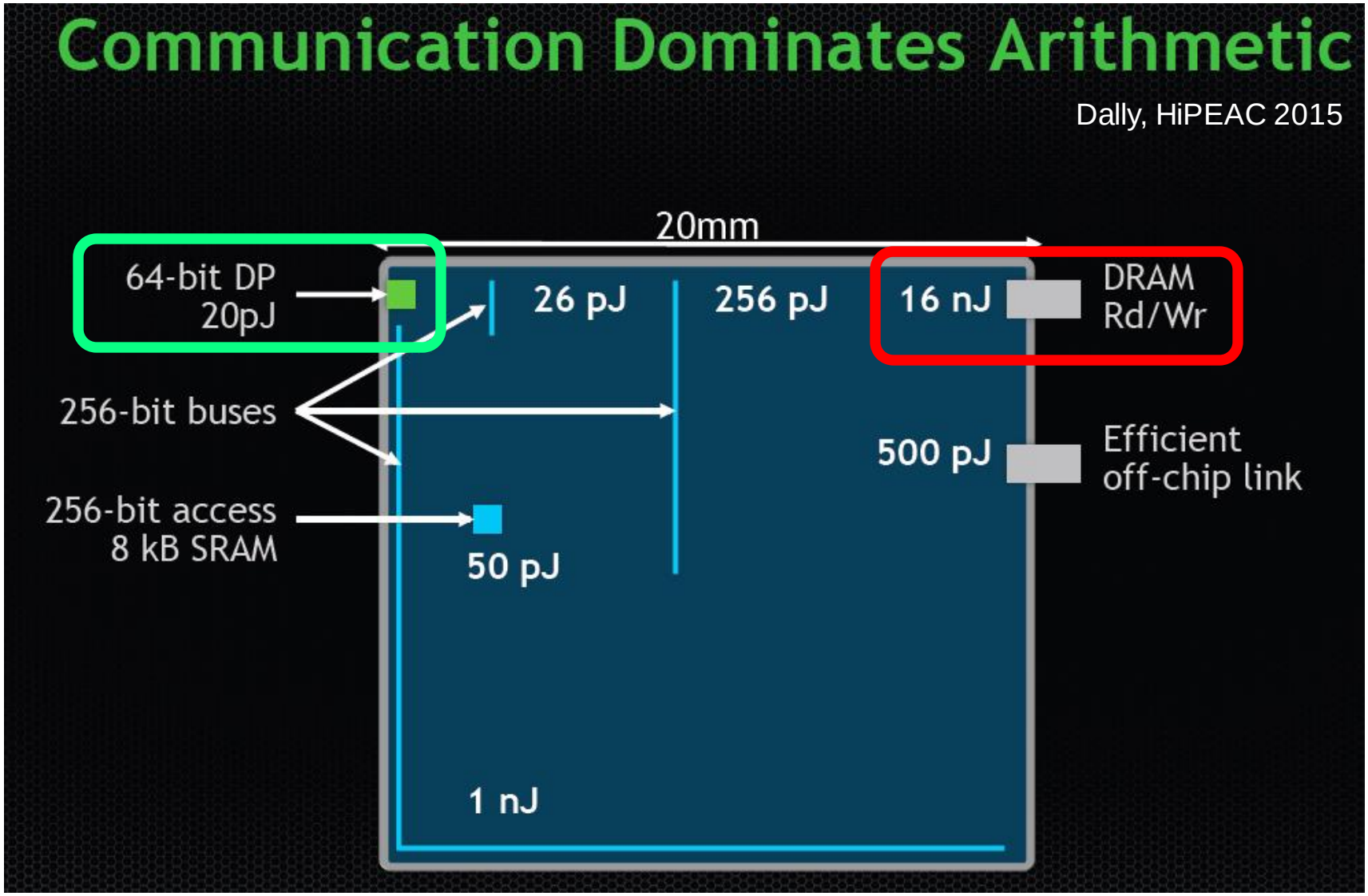


Figure 11: Half of cycles are spent stalled on caches.

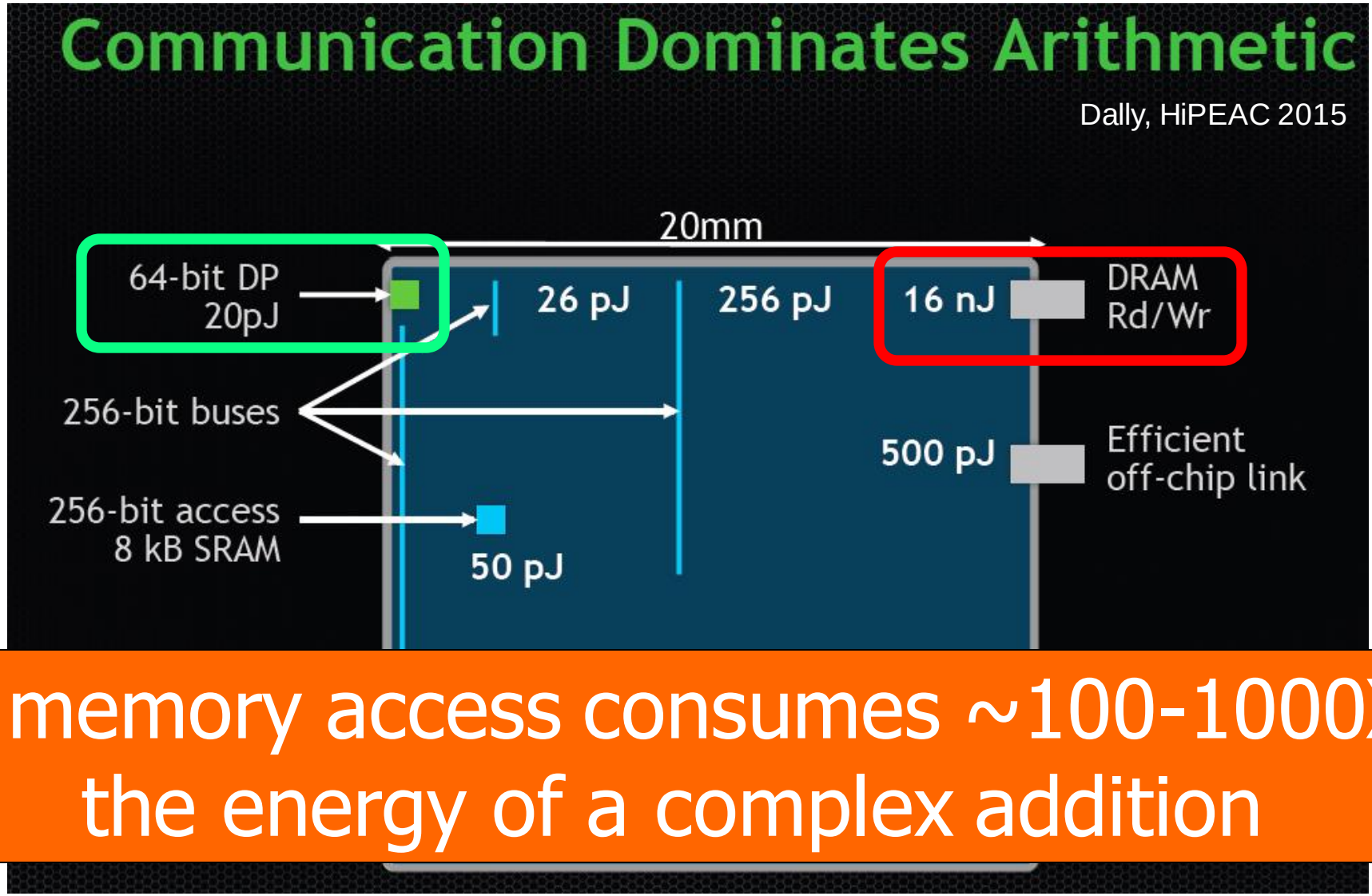
Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

Energy Perspective

Data Movement vs. Computation Energy

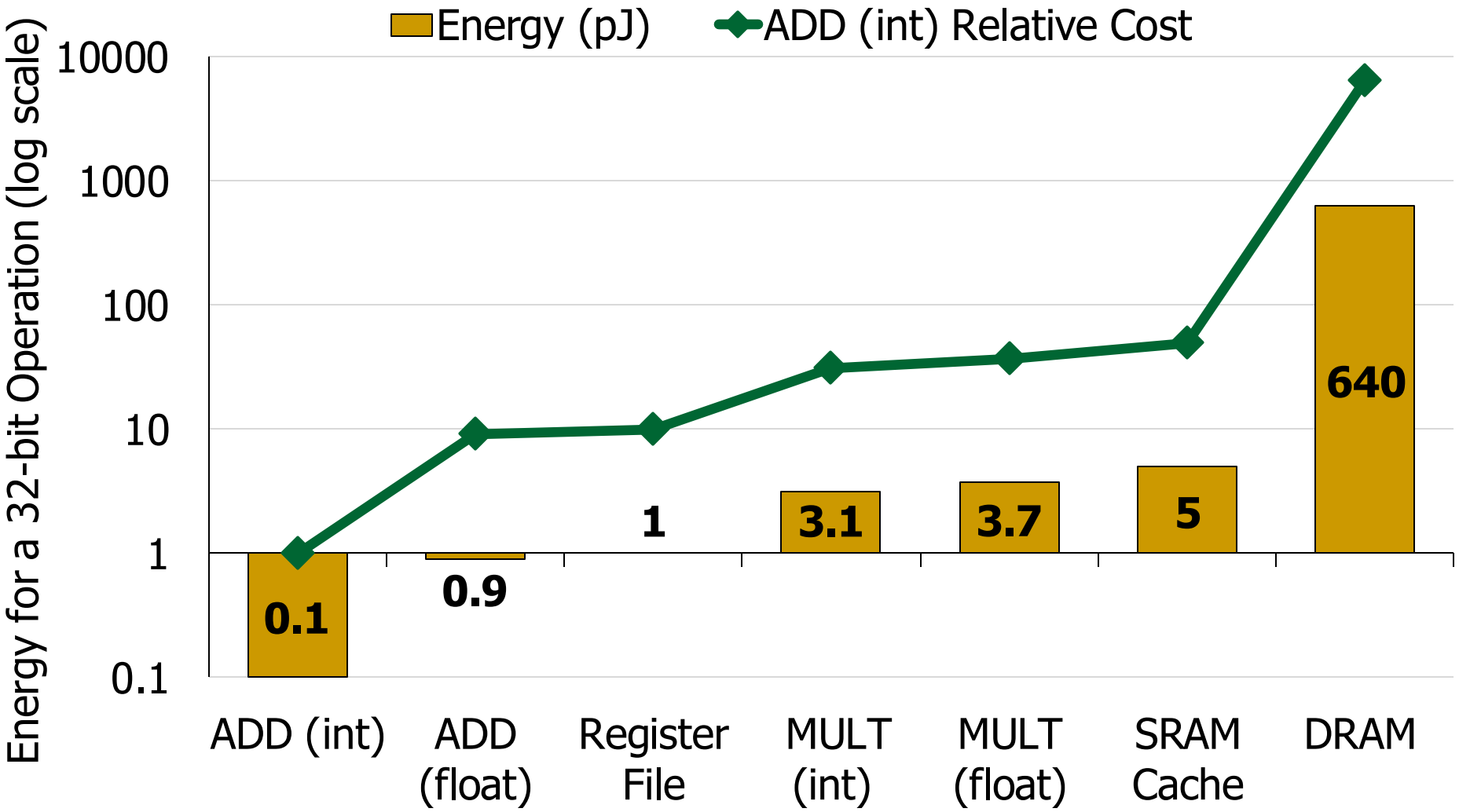


Data Movement vs. Computation Energy



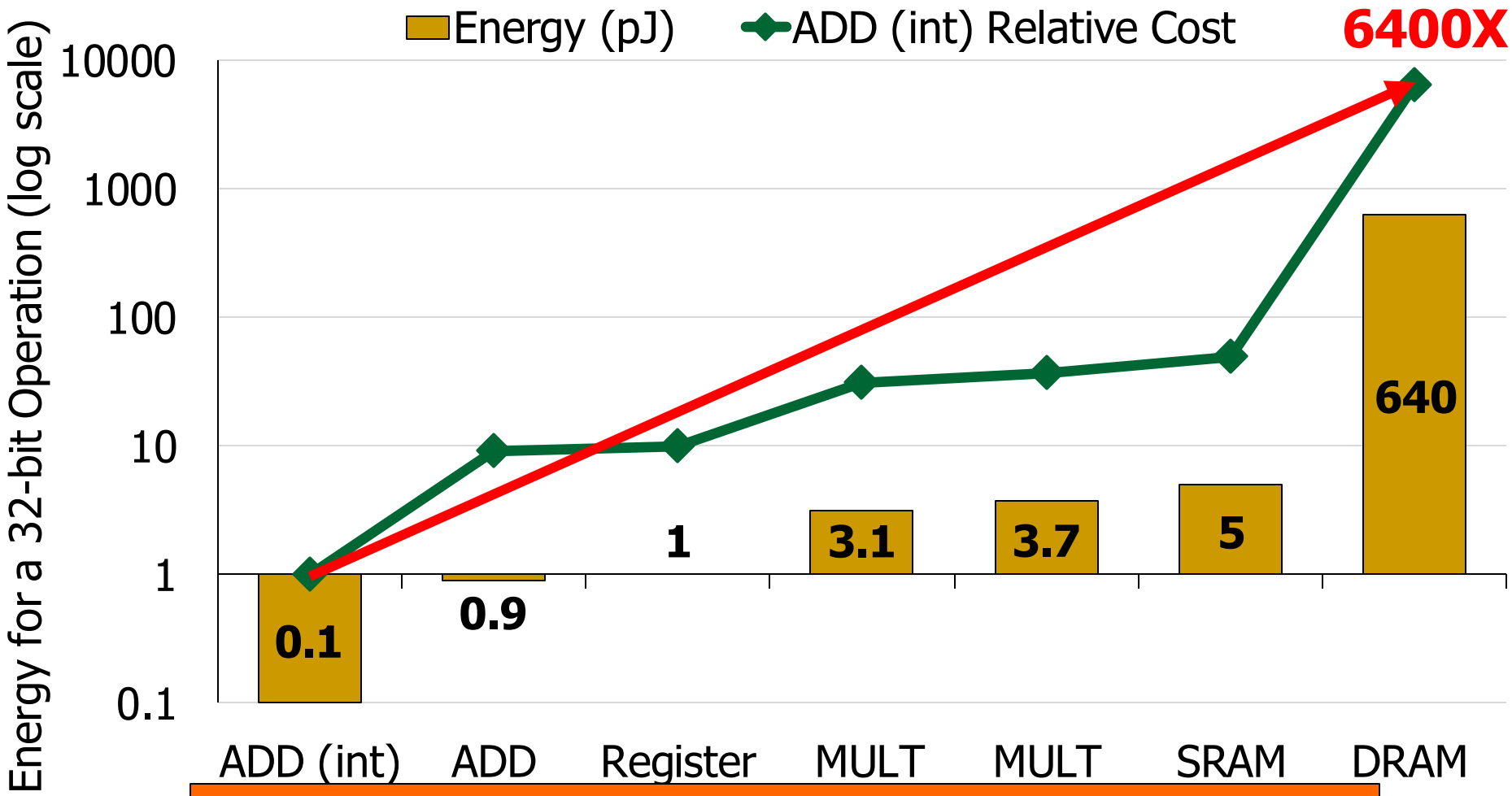
A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy



Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.

Data Movement vs. Computation Energy



A memory access consumes ~6400X the energy of an integer addition

Data Movement vs. Computation Energy

32-bit Operation	Energy (pJ)	ADD (int) Relative Cost
ADD (int)	0.1	1
ADD (float)	0.9	9
Register File	1	10
MULT (int)	3.1	31
MULT (float)	3.7	37
SRAM Cache	5	50
DRAM	**640**	**6400**

A memory access consumes ~6400X the energy of an integer addition

Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.

Memory is Critical for Energy

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.

62.7% of the total system energy is spent on data movement

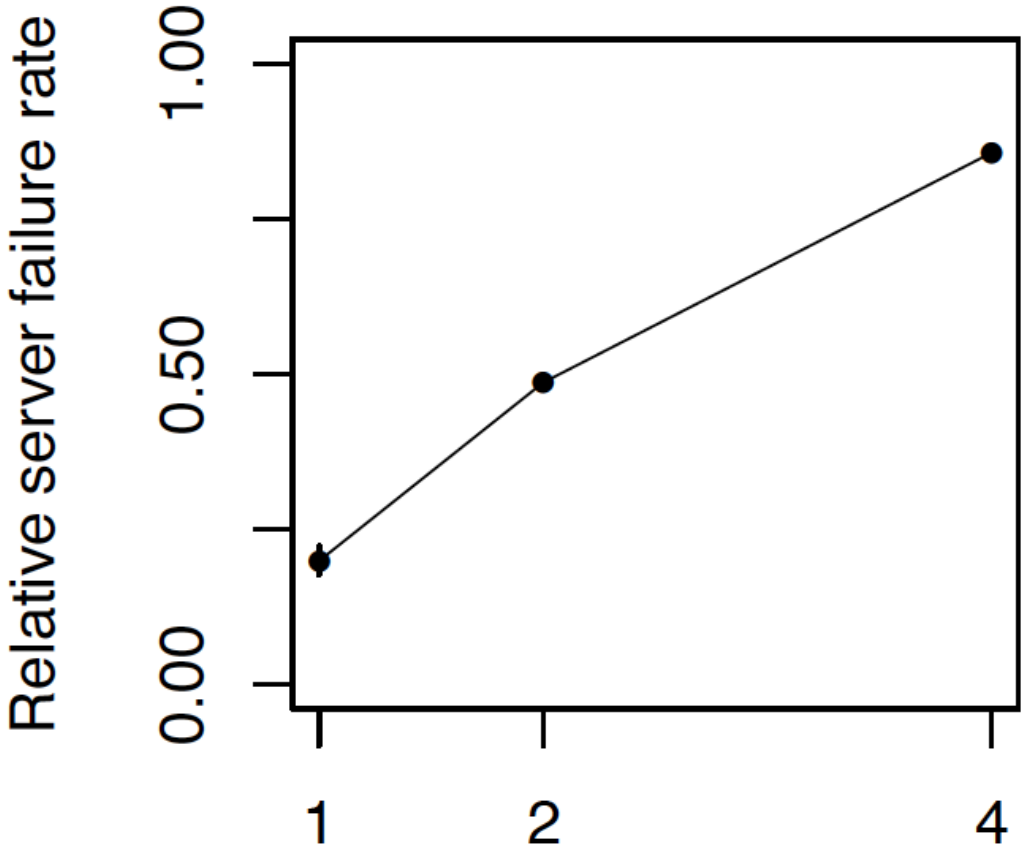
Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Reliability & Security Perspectives

Memory is Critical for Reliability

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

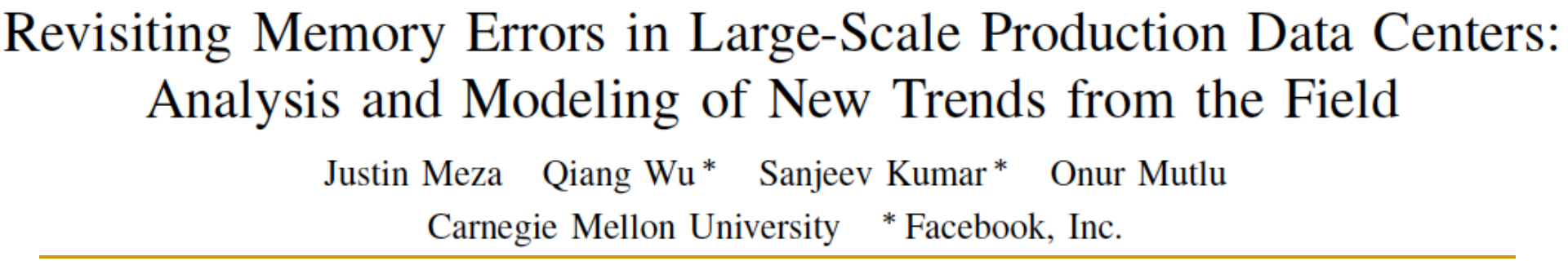


As memory capacity increases, system reliability reduces

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.

A Curious Discovery [Kim et al., ISCA 2014]

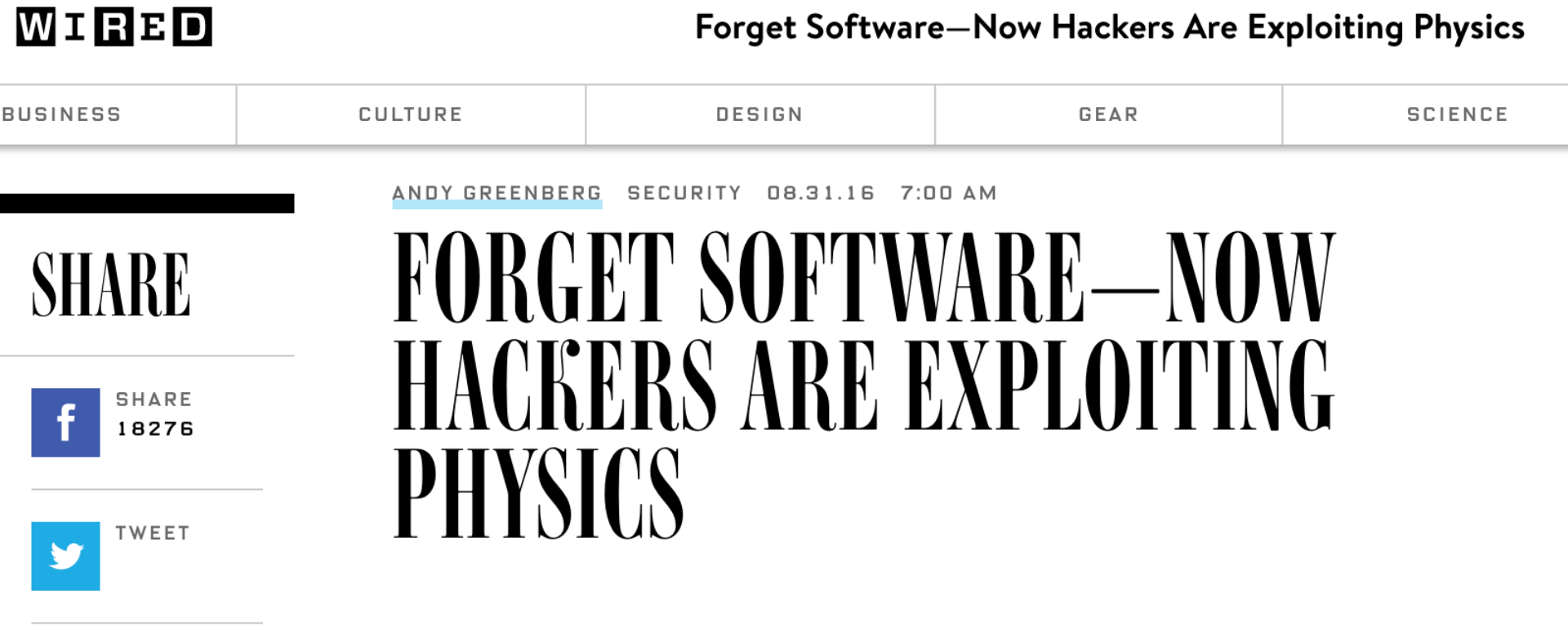
One can

predictably induce errors

in most DRAM memory chips

DRAM RowHammer

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



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

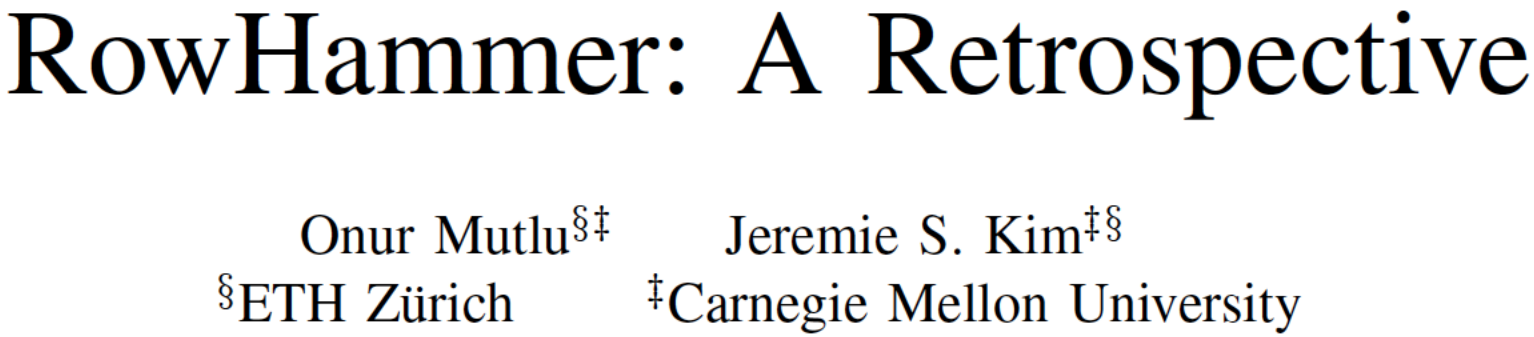
Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

A Recent RowHammer Retrospective

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (***TCAD***) *Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (30 minutes)]


RowHammer: A Retrospective

Onur Mutlu[§‡] Jeremie S. Kim[‡§]
[§]ETH Zürich [‡]Carnegie Mellon University

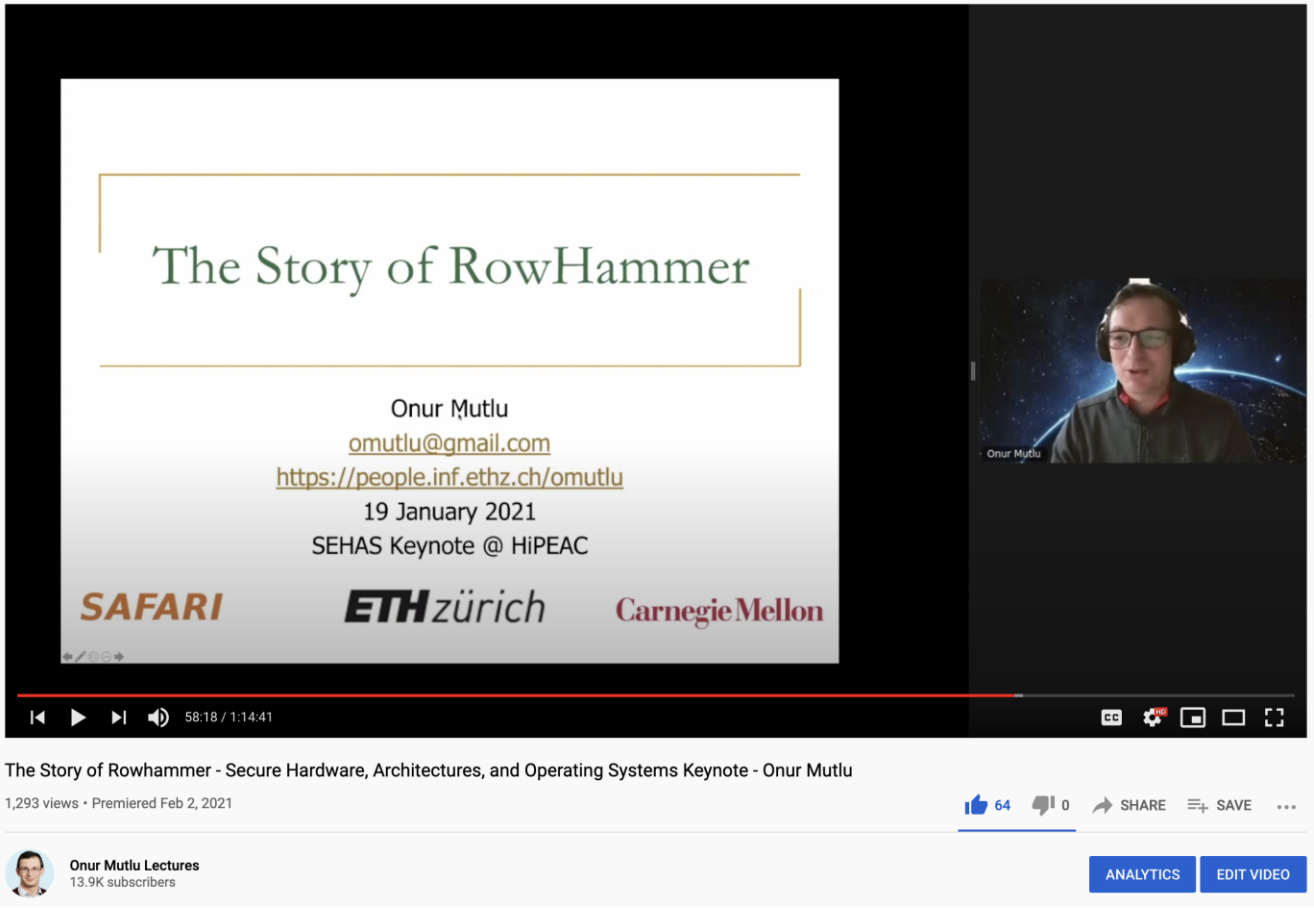
Memory is Critical for Security

Detailed Lectures on RowHammer

- Computer Architecture, Fall 2020, Lecture 4b
  - RowHammer (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=KDy632z23UE&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
- 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
- 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

The Story of RowHammer Lecture …

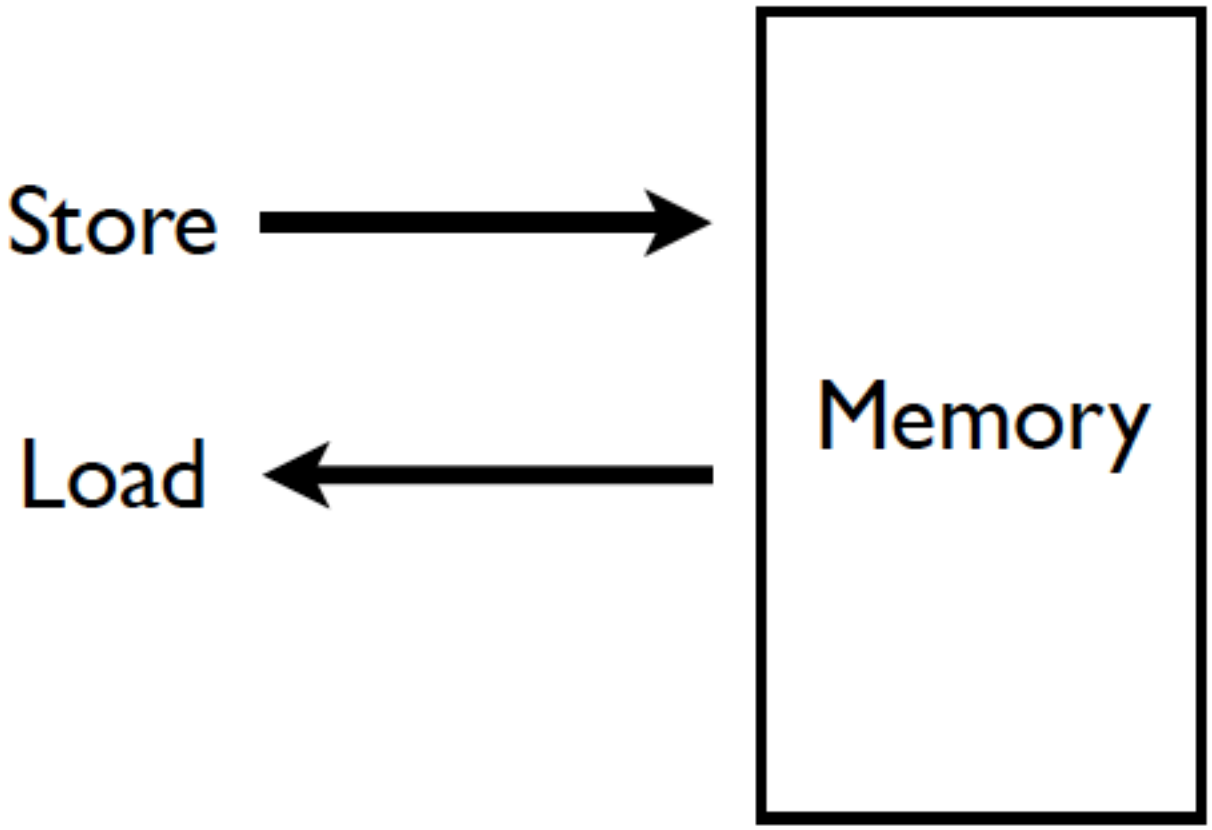
- Onur Mutlu,
  "The Story of RowHammer"
  Keynote Talk at *Secure Hardware, Architectures, and Operating Systems Workshop* (***SeHAS***), *held with HiPEAC 2021 Conference*, Virtual, 19 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]



Memory Fundamentals

Memory Organization & Technology

Memory (Programmer's View)



Abstraction: Virtual vs. Physical Memory

- Programmer sees virtual memory
  - Can assume the memory is "infinite"
- Reality: Physical memory size is much smaller than what the programmer assumes
- The system (system software + hardware, cooperatively) maps virtual memory addresses to physical memory
  - The system automatically manages the physical memory space transparently to the programmer

+ Programmer does not need to know the physical size of memory nor manage it → A small physical memory can appear as a huge one to the programmer → Life is easier for the programmer

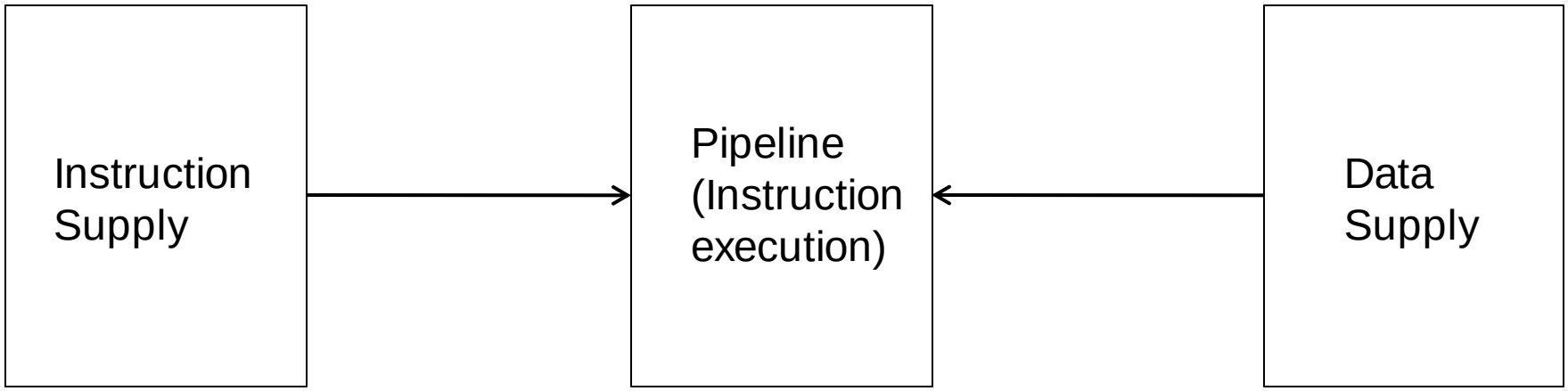
-- More complex system software and architecture

A classic example of the programmer/(micro)architect tradeoff

(Physical) Memory System

- You need a larger level of storage to manage a small amount of physical memory automatically
  → Physical memory has a backing store: disk
- We will first start with the physical memory system
- For now, ignore the virtual→physical indirection
- We will get back to it later, if time permits…

Idealism



- Zero latency access
- Infinite capacity
- Zero cost
- Perfect control flow

- No pipeline stalls
- Perfect data flow (reg/memory dependencies)
- Zero-cycle interconnect (operand communication)
- Enough functional units
- Zero latency compute

- Zero latency access
- Infinite capacity
- Infinite bandwidth
- Zero cost

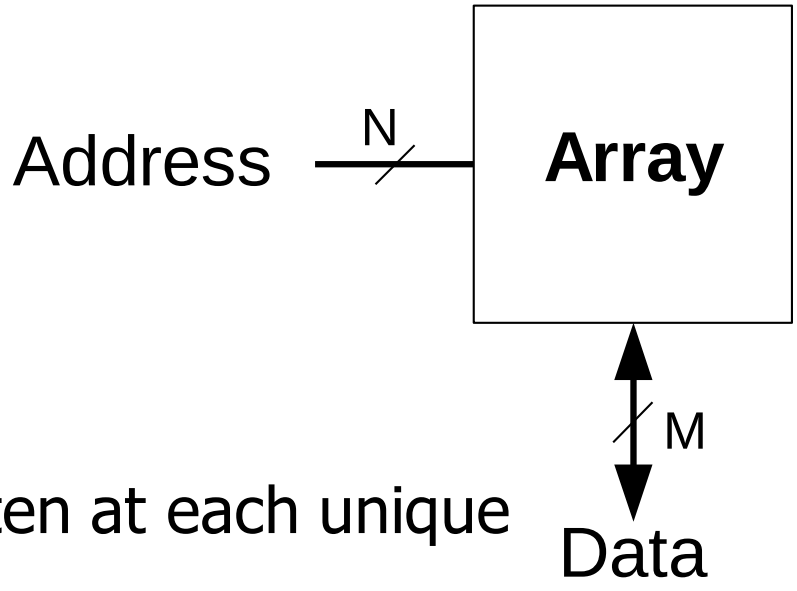
Quick Overview of Memory Arrays

How Can We Store Data?

- Flip-Flops (or Latches)
  - Very fast, parallel access
  - Very expensive (one bit costs tens of transistors)
- Static RAM (we will describe them in a moment)
  - Relatively fast, only one data word at a time
  - Expensive (one bit costs 6+ transistors)
- Dynamic RAM (we will describe them in a moment)
  - Slower, one data word at a time, reading destroys content (refresh), needs special process for manufacturing
  - Cheap (one bit costs only one transistor plus one capacitor)
- Other storage technology (flash memory, hard disk, tape)
  - Much slower, access takes a long time, non-volatile
  - Very cheap (one transistor stores many bits or no transistors involved)

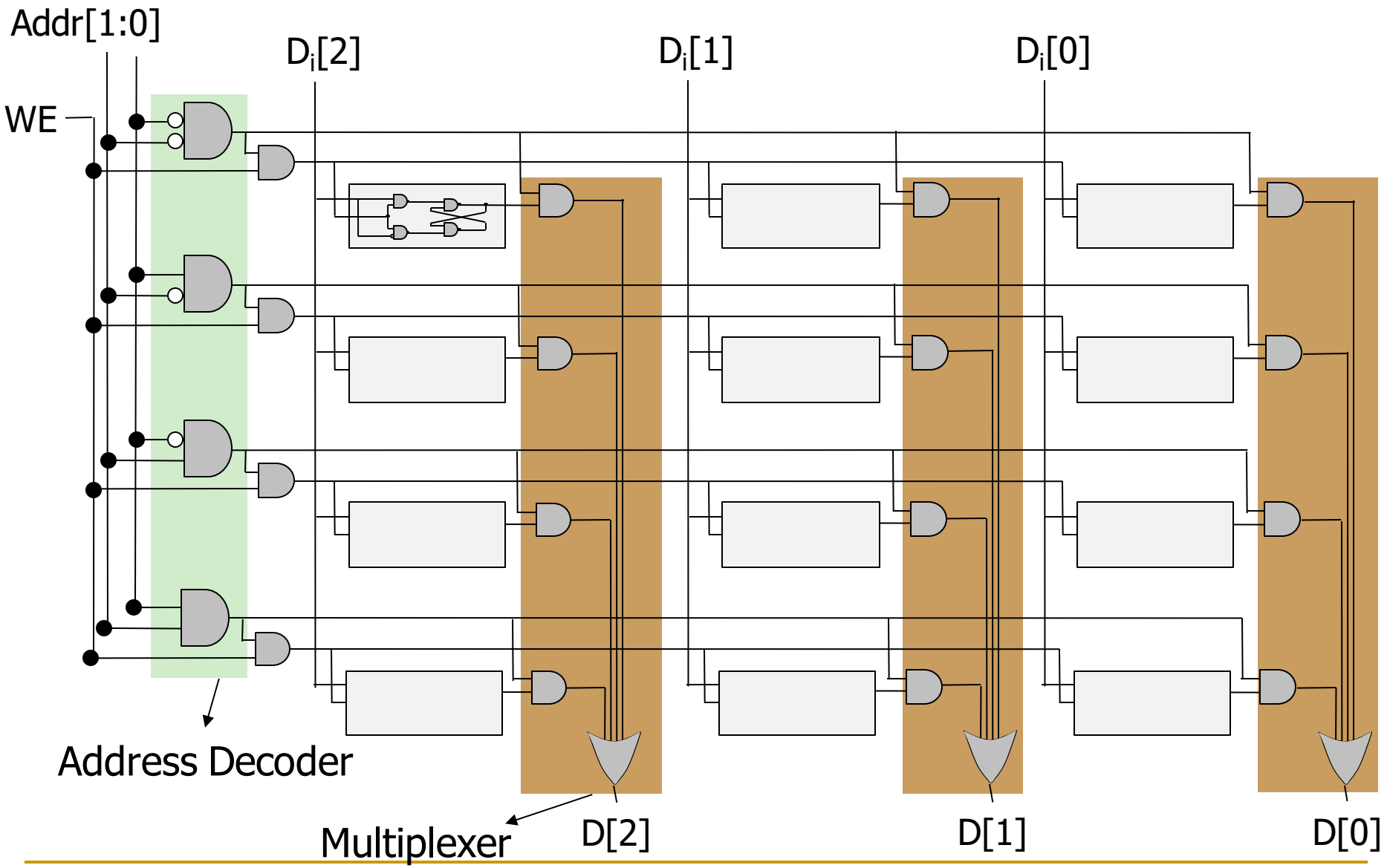
Array Organization of Memories

- Goal: Efficiently store large amounts of data
  - A memory array (stores data)
  - Address selection logic (selects one row of the array)
  - Readout circuitry (reads data out)



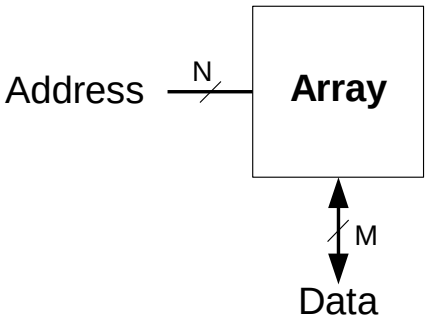
- An M-bit value can be read or written at each unique N-bit address
  - All values can be accessed, but only M-bits at a time
  - Access restriction allows more compact organization

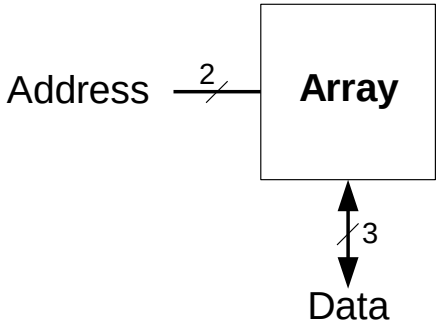
Recall: A Bigger Memory Array (4 locations X 3 bits)

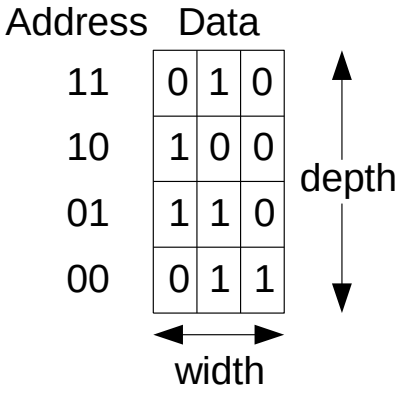


Memory Arrays

- Two-dimensional array of bit cells
  - Each bit cell stores one bit
- An array with N address bits and M data bits:
  - 2^N rows and M columns
  - Depth: number of rows (number of words)
  - Width: number of columns (size of word)
  - Array size: depth × width = 2^N × M

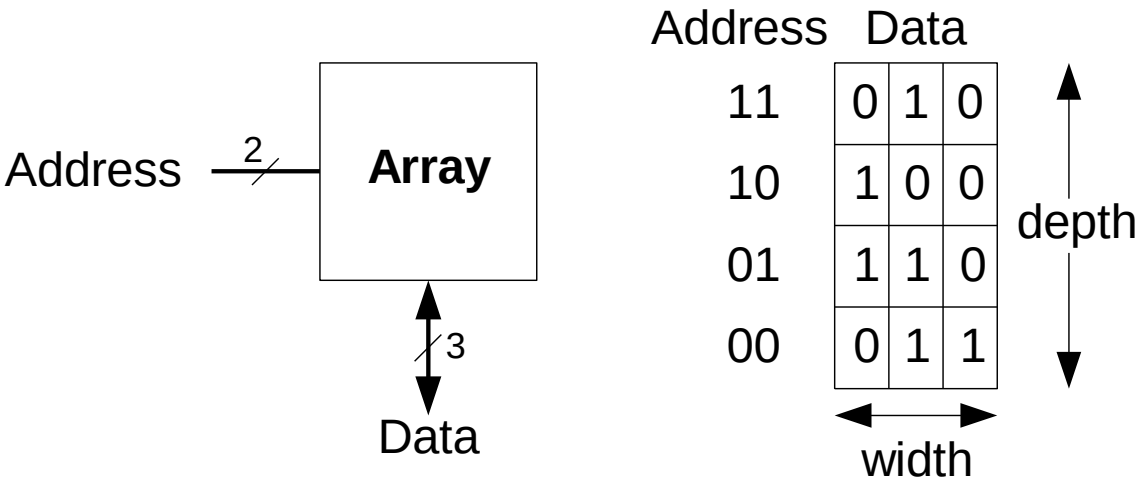




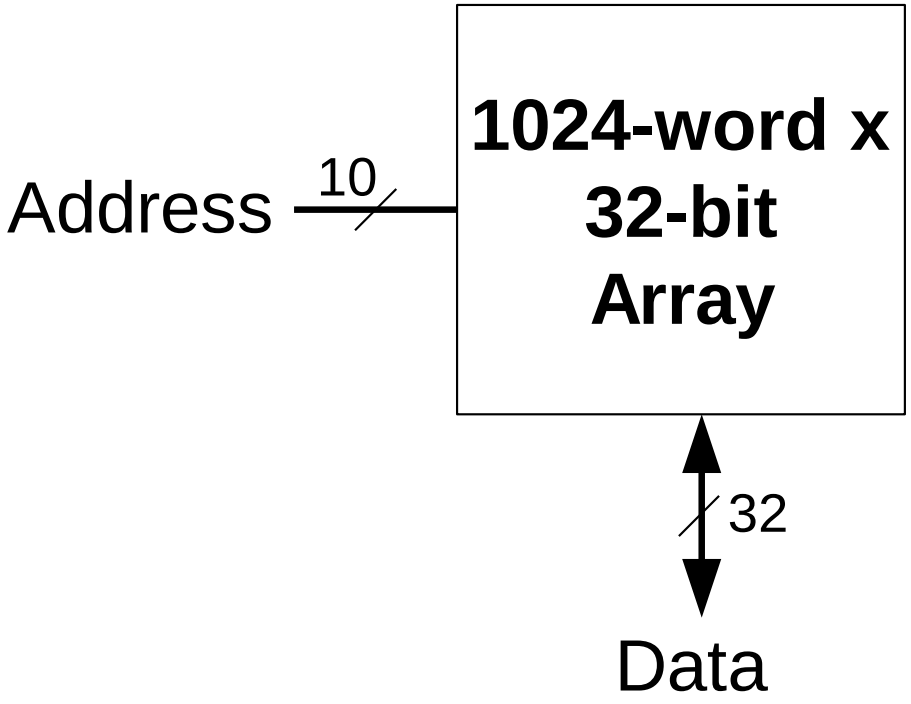


Memory Array Example

- $2^2 \times$ 3-bit array
- Number of words: 4
- Word size: 3-bits
- For example, the 3-bit word stored at address 10 is 100

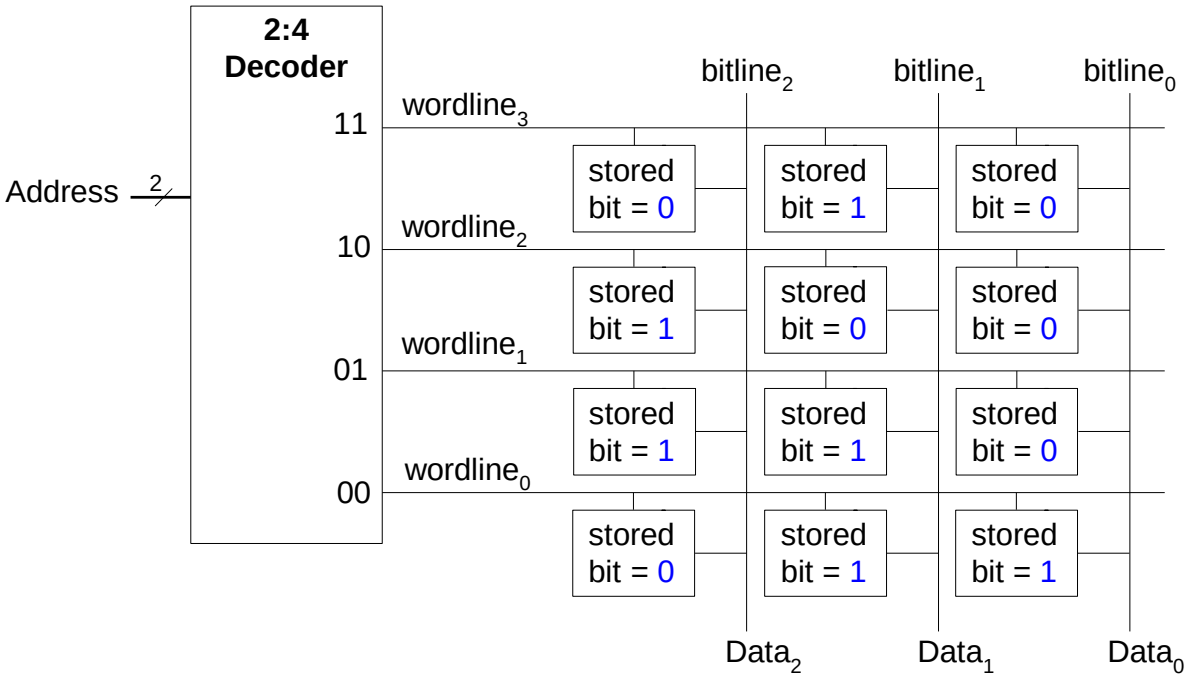


Larger and Wider Memory Array Example



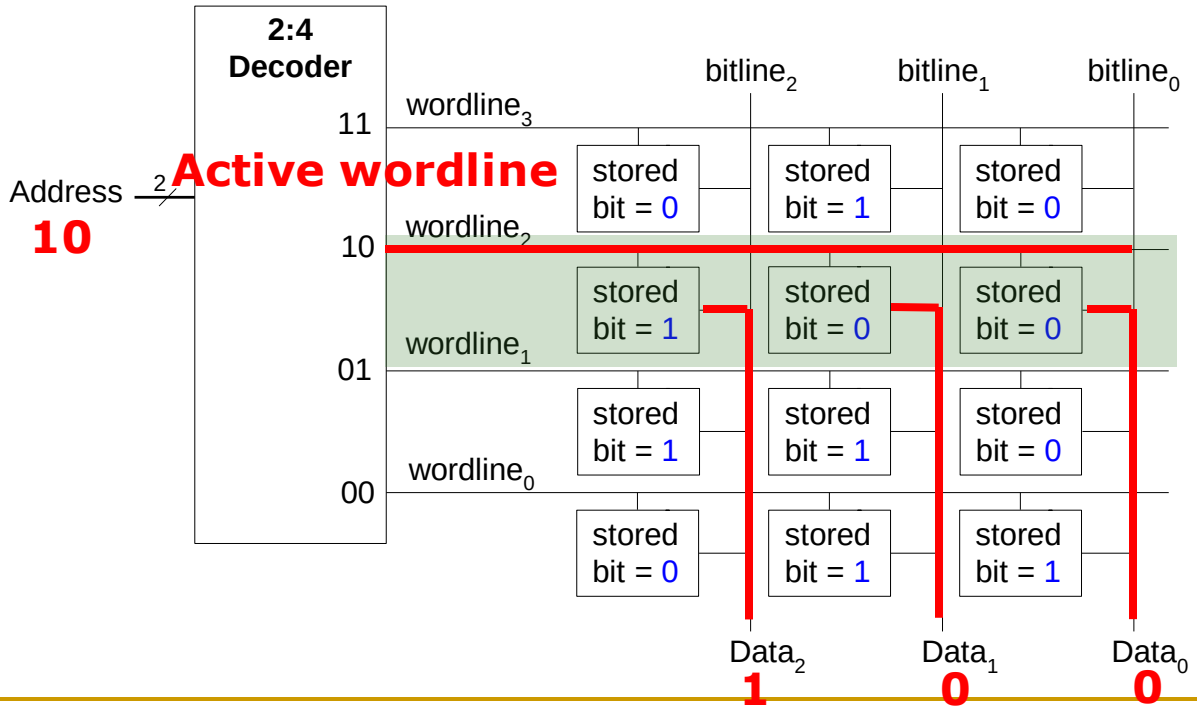
Memory Array Organization (I)

- Storage nodes in one column connected to one bitline
- Address decoder activates only ONE wordline
- Content of one line of storage available at output



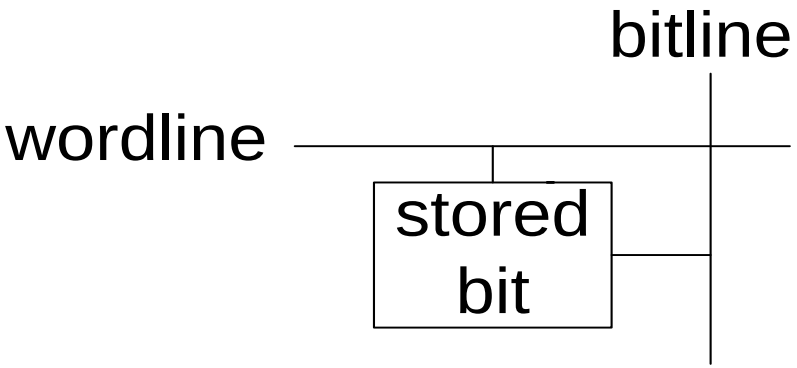
Memory Array Organization (II)

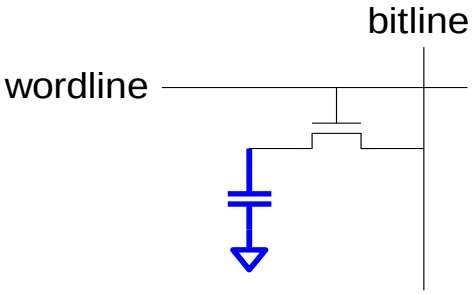
- Storage nodes in one column connected to one bitline
- Address decoder activates only ONE wordline
- Content of one line of storage available at output



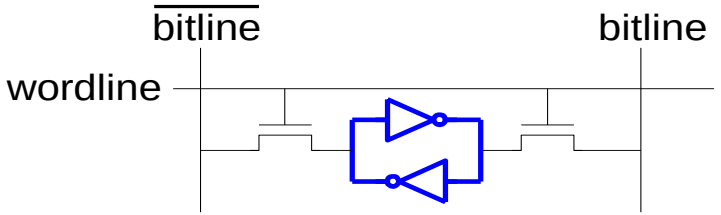
How is Access Controlled?

- Access transistors (that are configured as switches) connect the bit storage to the bitline
- Access controlled by the wordline





DRAM



SRAM

Building Larger Memories

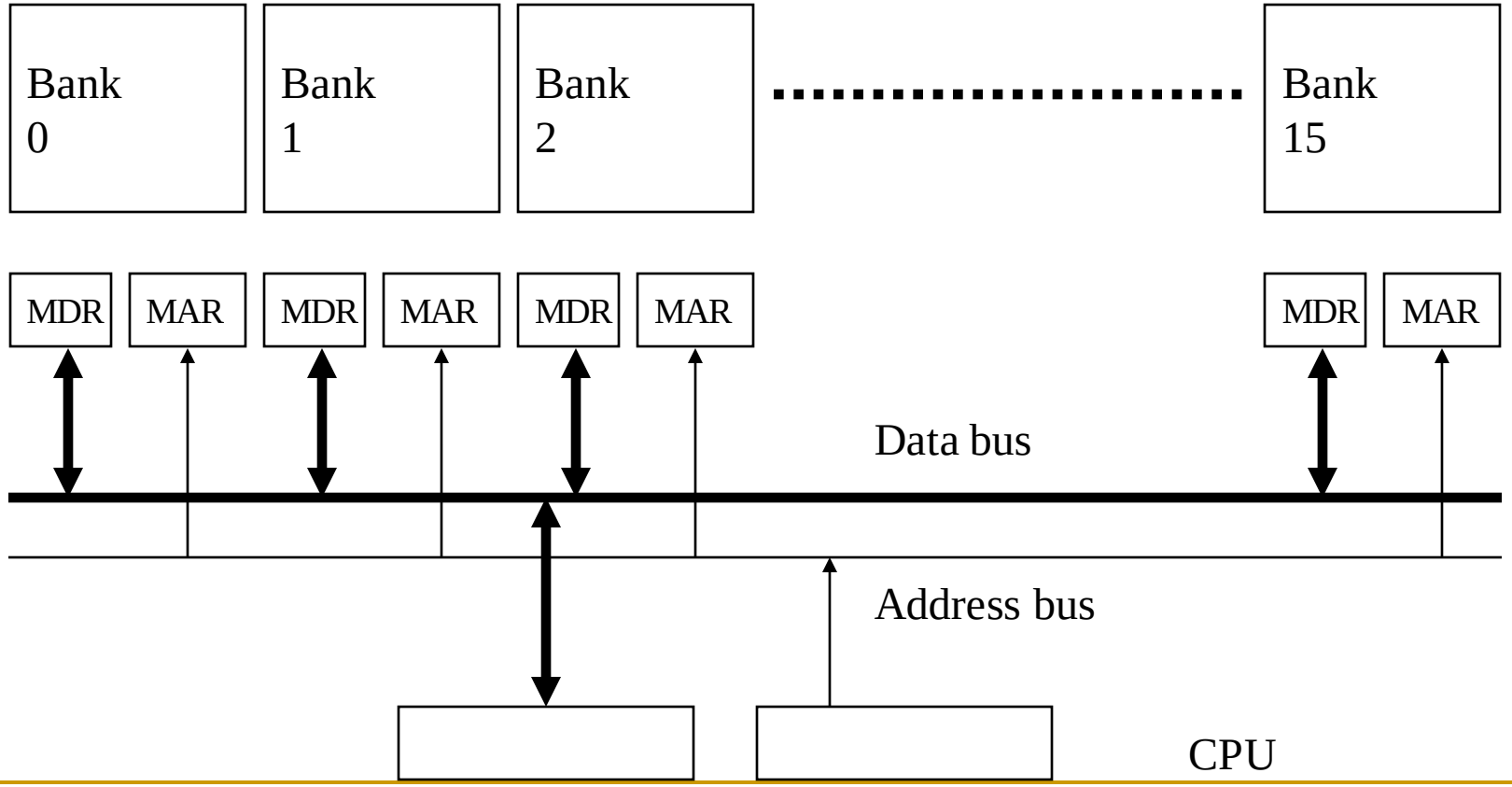
- Requires larger memory arrays
- Large → slow
- How do we make the memory large without making it too slow?
- Idea: Divide the memory into smaller arrays and interconnect the arrays to input/output buses
  - Large memories are hierarchical array structures
  - DRAM: Channel → Rank → Bank → Subarrays → Mats

General Principle: Interleaving (Banking)

- Interleaving (banking)
  - Problem: a single monolithic large memory array takes long to access and does not enable multiple accesses in parallel
  - Goal: Reduce the latency of memory array access and enable multiple accesses in parallel
  - Idea: Divide a large array into multiple banks that can be accessed independently (in the same cycle or in consecutive cycles)
    - Each bank is smaller than the entire memory storage
    - Accesses to different banks can be overlapped
  - A Key Issue: How do you map data to different banks? (i.e., how do you interleave data across banks?)

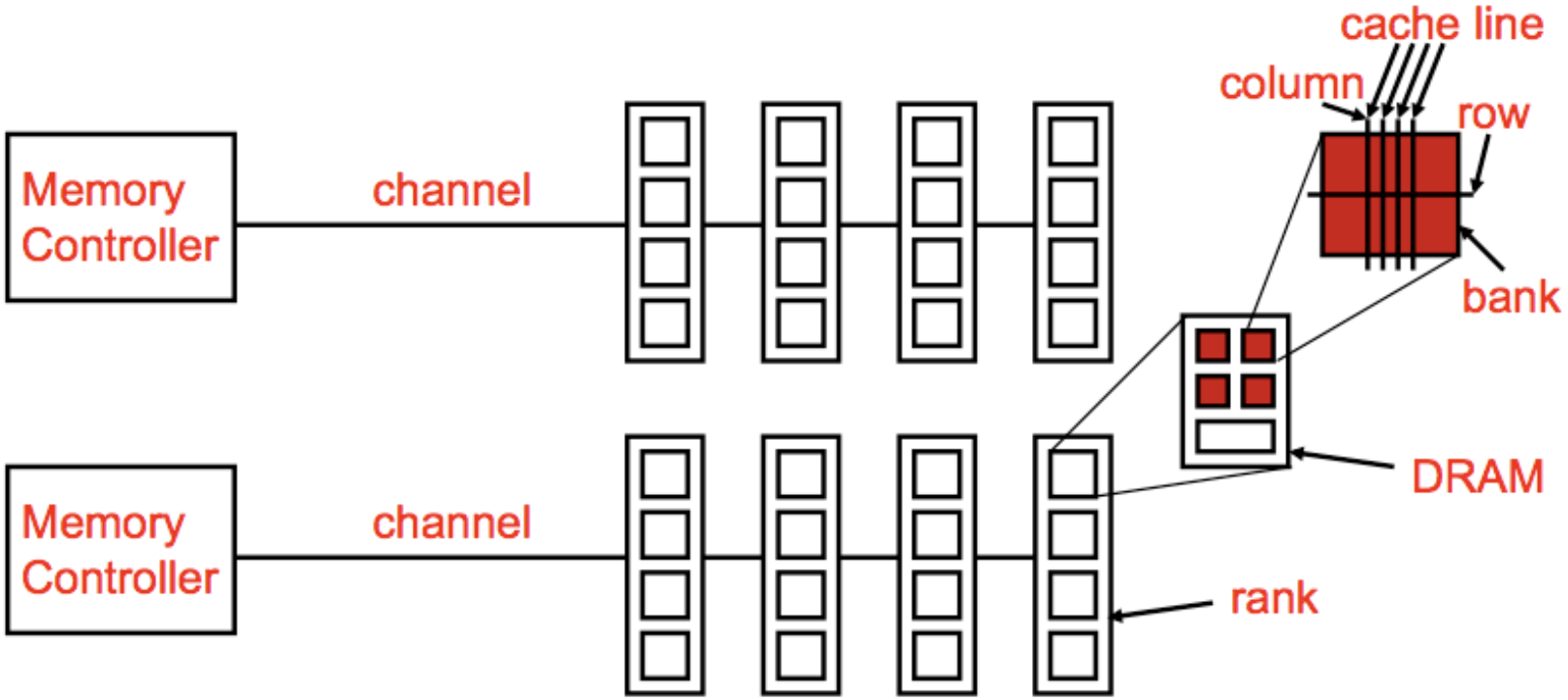
Recall: Memory Banking

- Memory is divided into banks that can be accessed independently; banks share address and data buses (to minimize pin cost)
- Can start and complete one bank access per cycle
- Can sustain N concurrent accesses if all N go to different banks

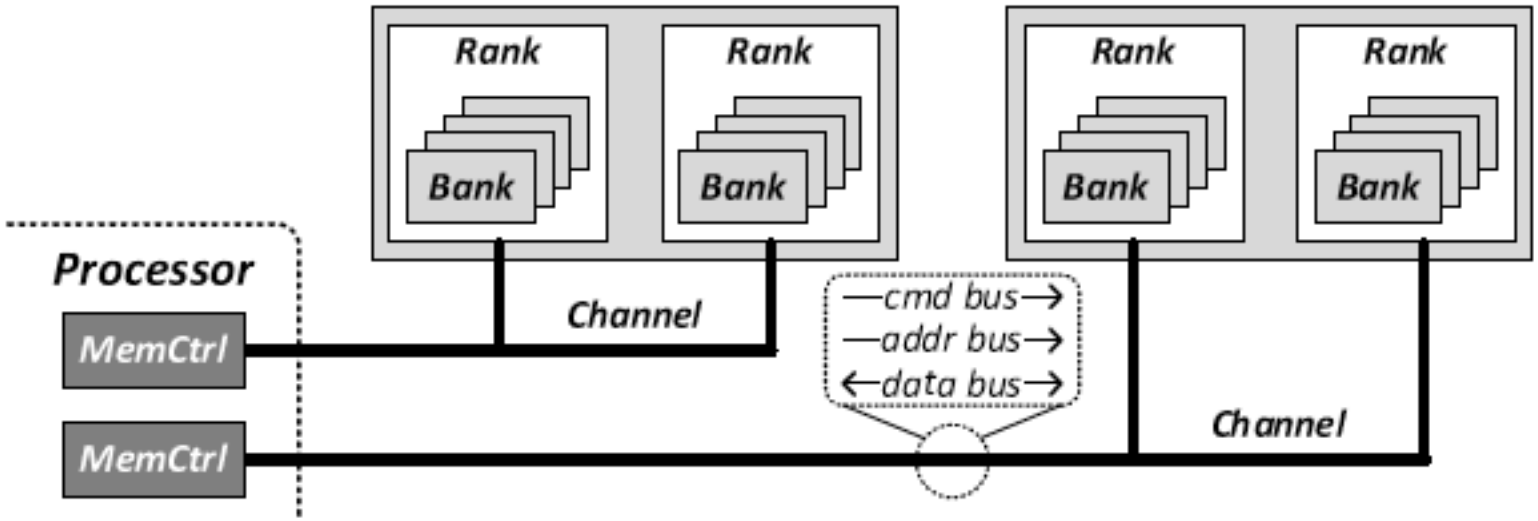


Picture credit: Derek Chiou

Generalized Memory Structure



Generalized Memory Structure



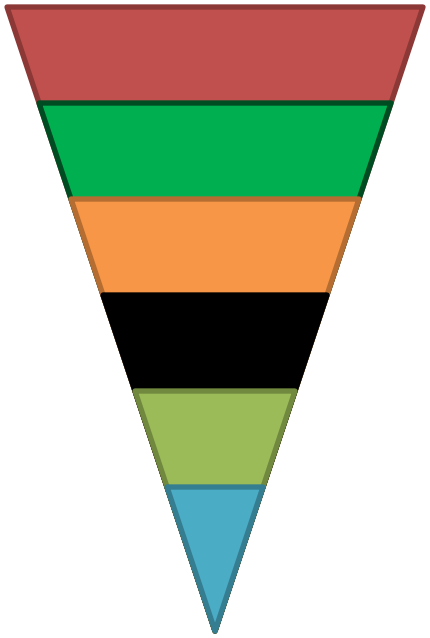
Kim+, "A Case for Exploiting Subarray-Level Parallelism in DRAM," ISCA 2012.

Lee+, "Decoupled Direct Memory Access," PACT 2015.

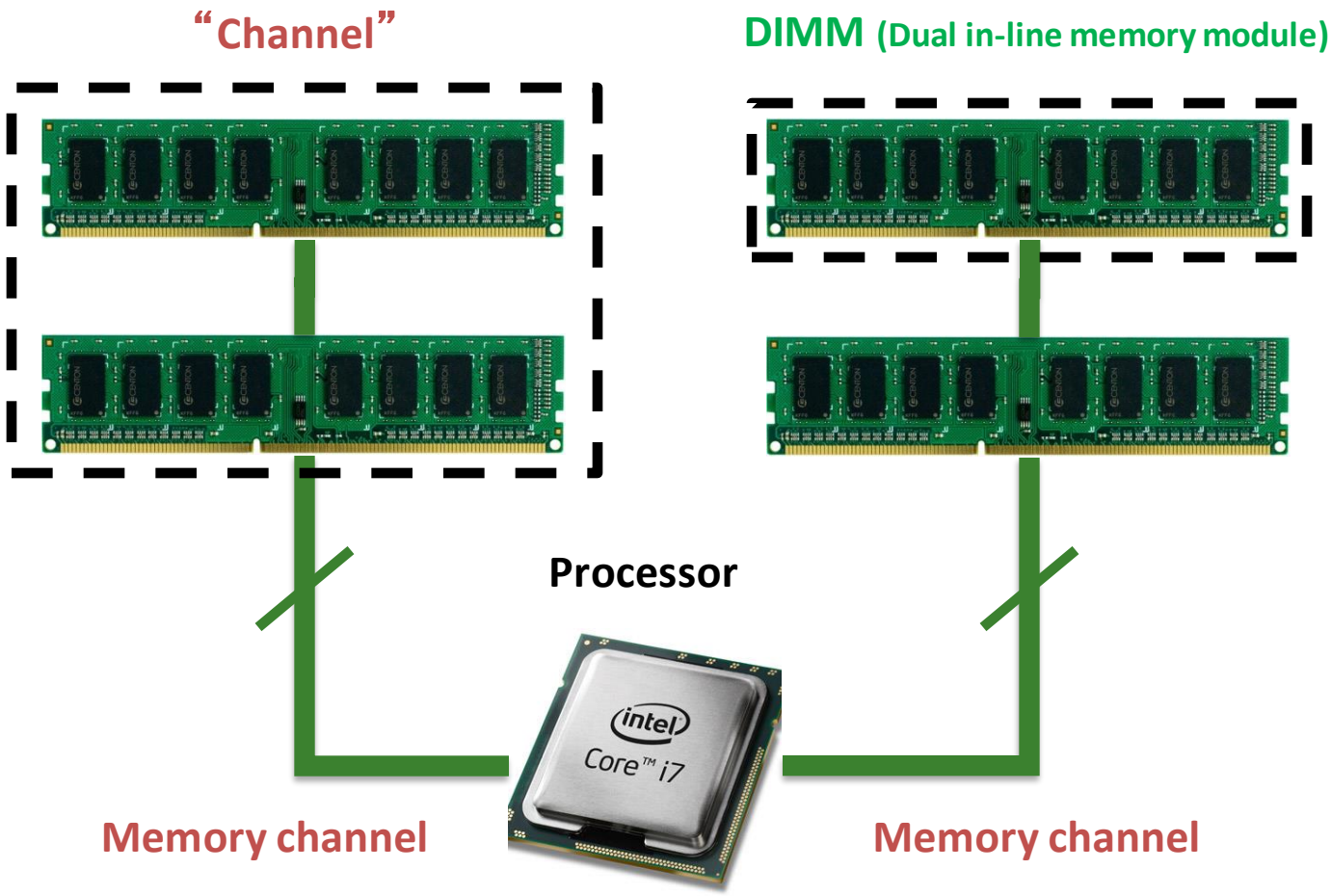
The DRAM Subsystem
The Top-Down View

DRAM Subsystem Organization

- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



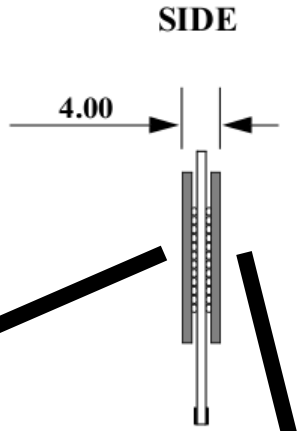
The DRAM Subsystem



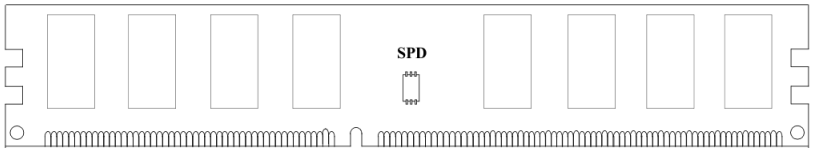
Breaking down a DIMM (module)

DIMM (Dual in-line memory module)

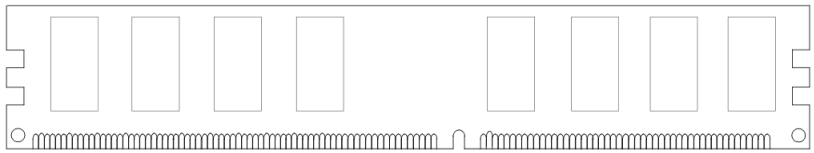
Side view



Front of DIMM



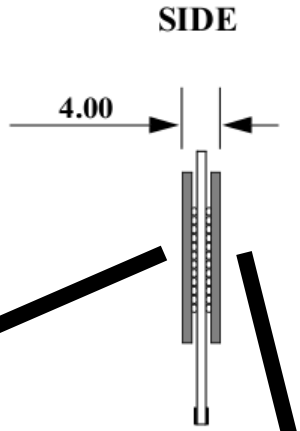
Back of DIMM



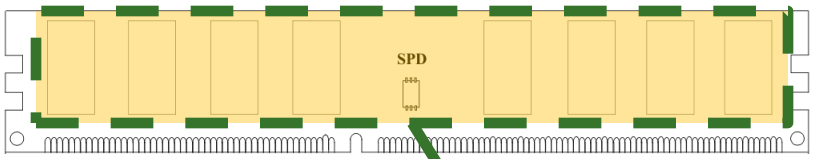
Breaking down a DIMM (module)

DIMM (Dual in-line memory module)

Side view

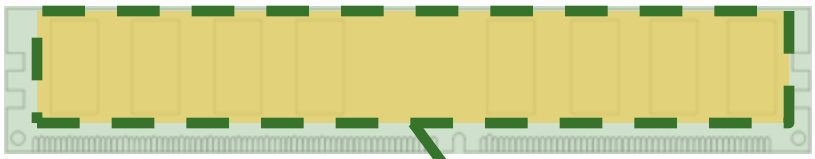


Front of DIMM



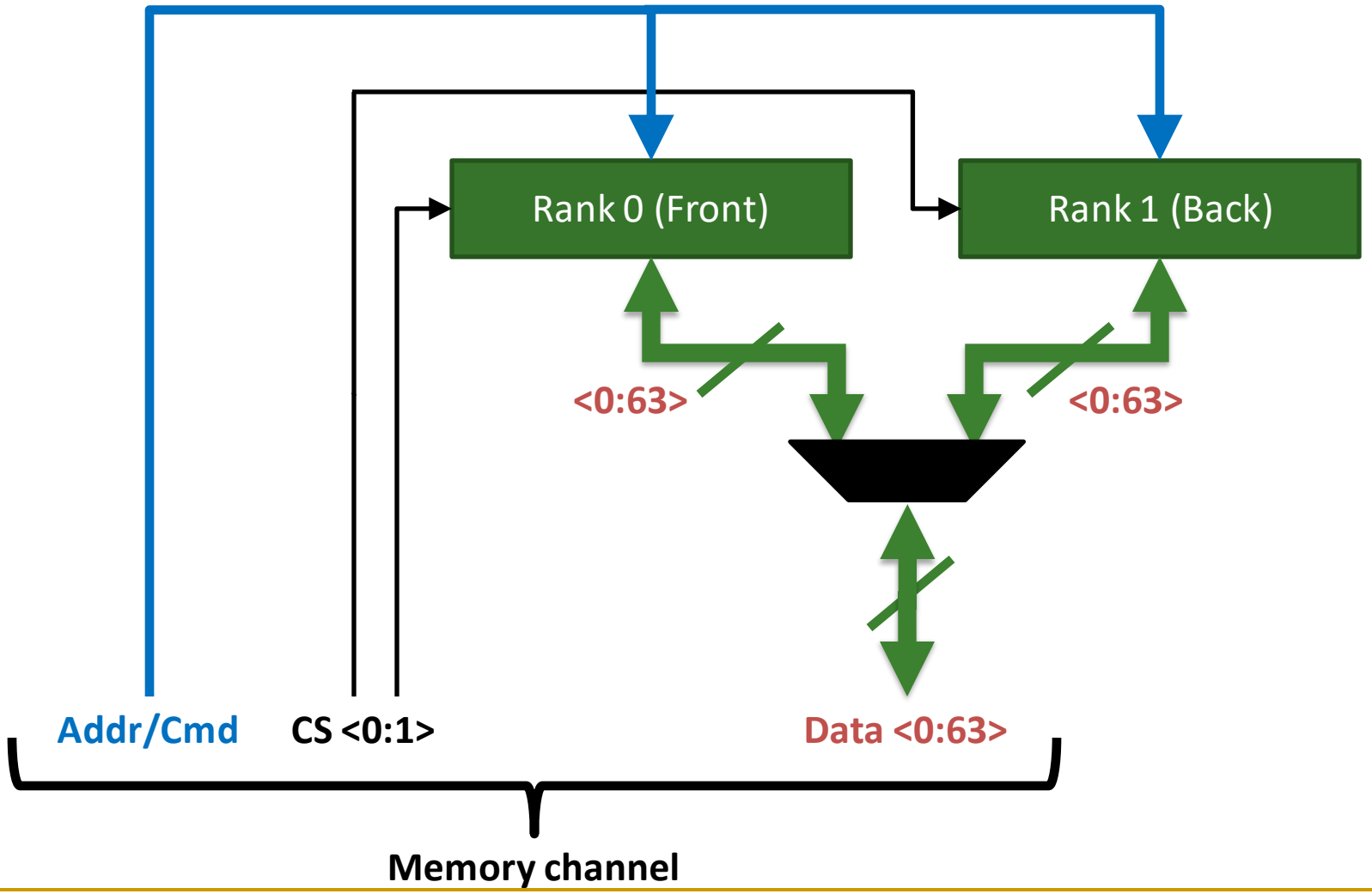
Rank 0: collection of 8 chips

Back of DIMM

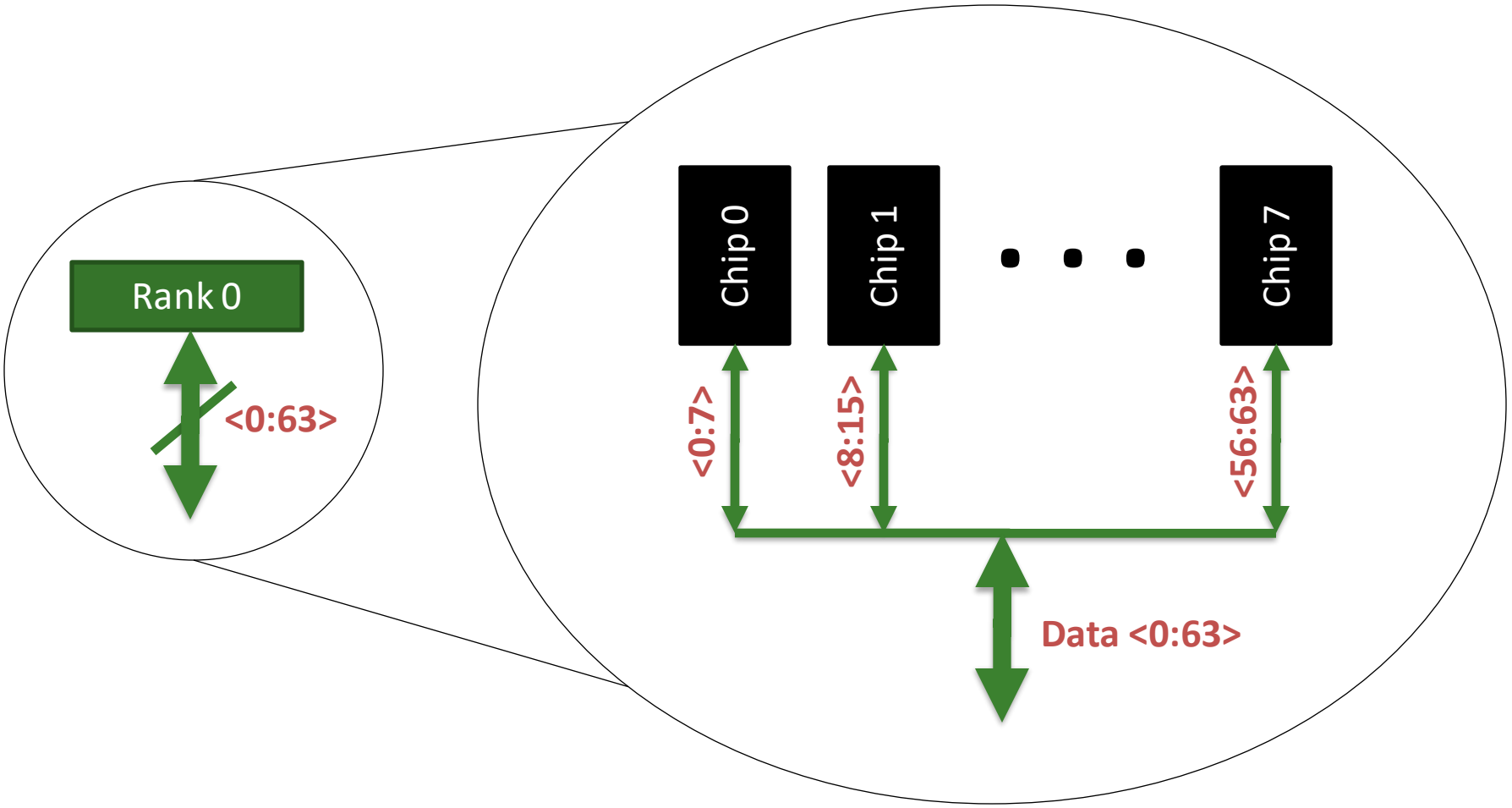


Rank 1

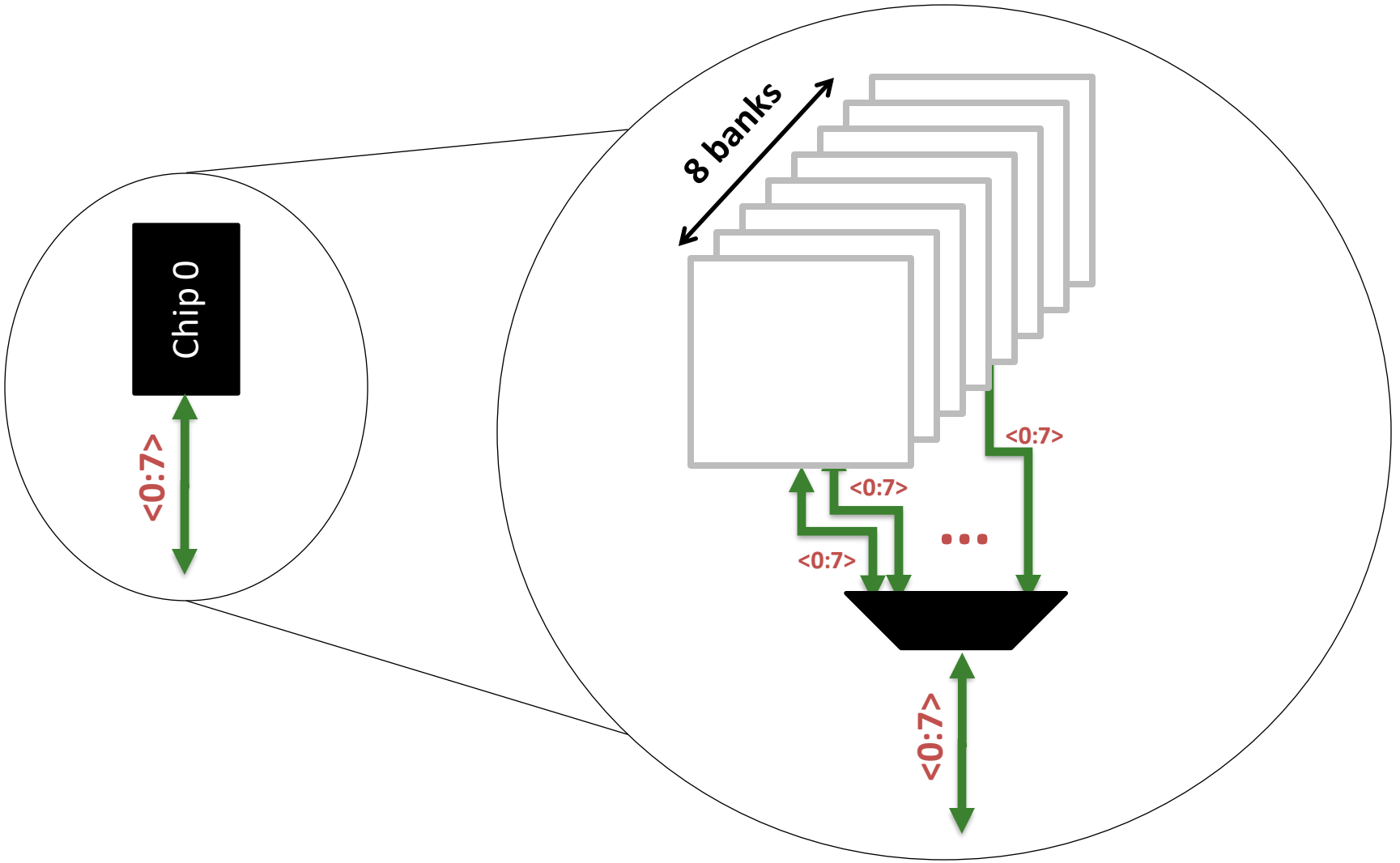
Rank


Rank 0 (Front)
Rank 1 (Back)
<0:63>
<0:63>
Addr/Cmd
CS <0:1>
Data <0:63>
Memory channel

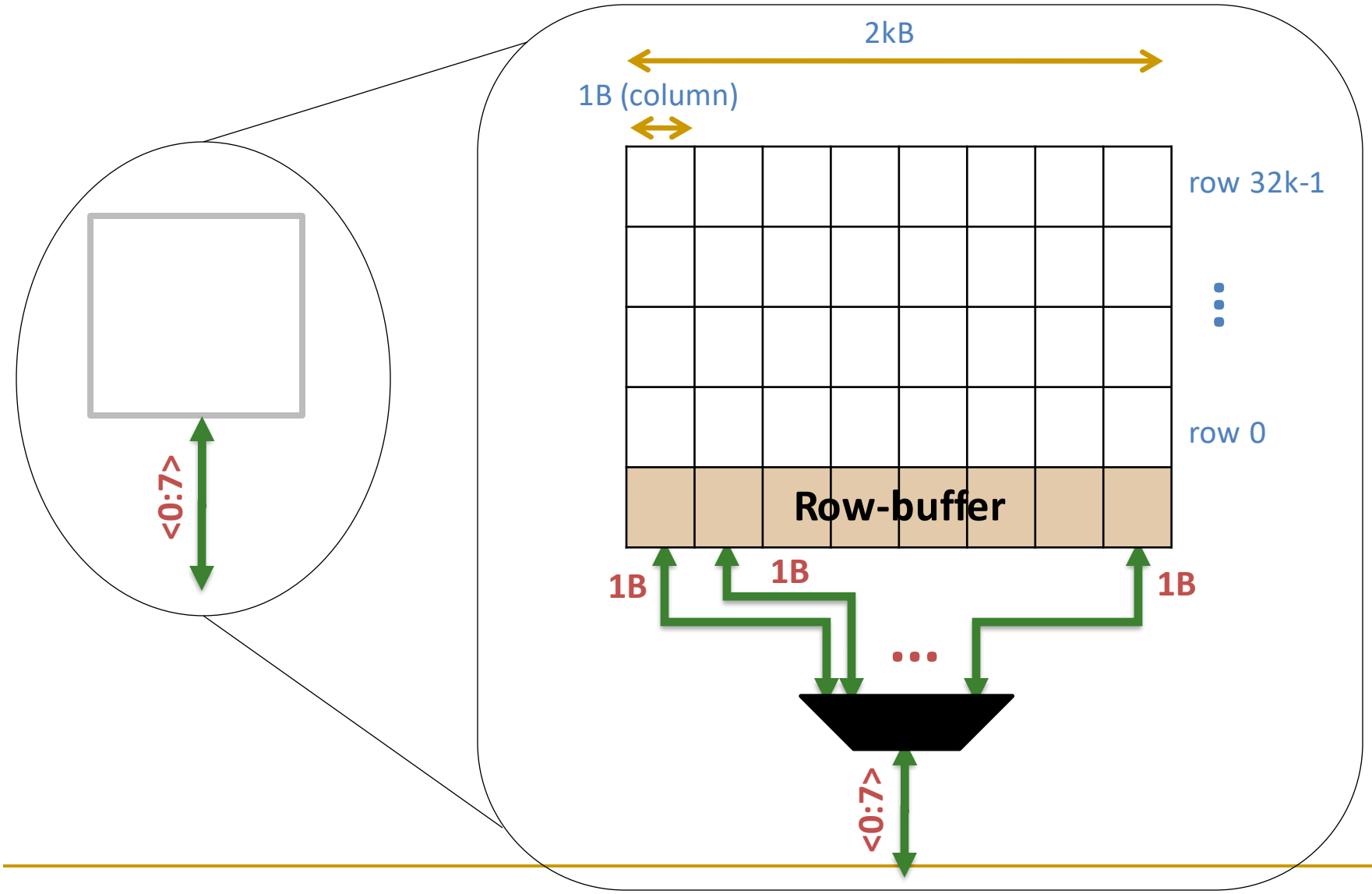
Breaking down a Rank



Breaking down a Chip



Breaking down a Bank



A DRAM Bank Internally Has Sub-Banks

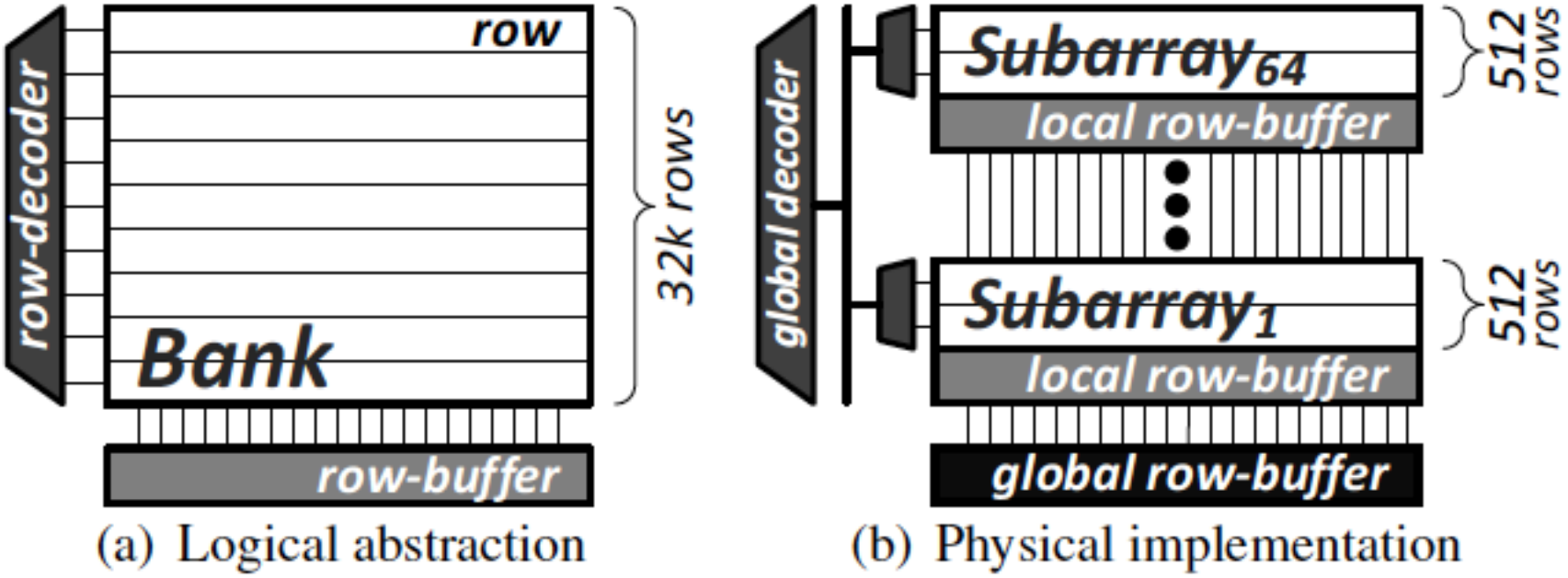
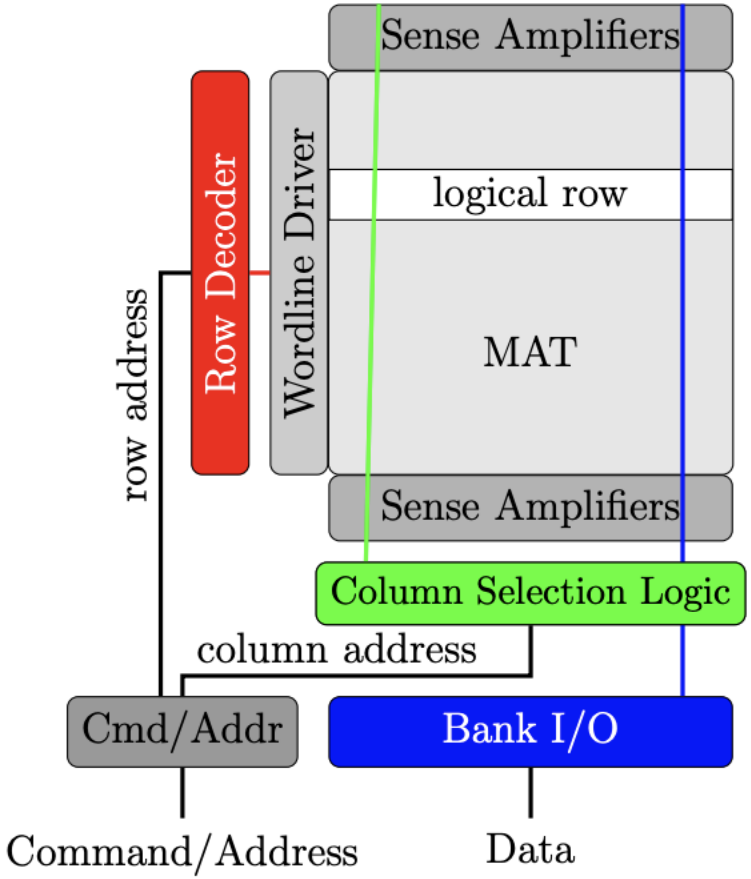
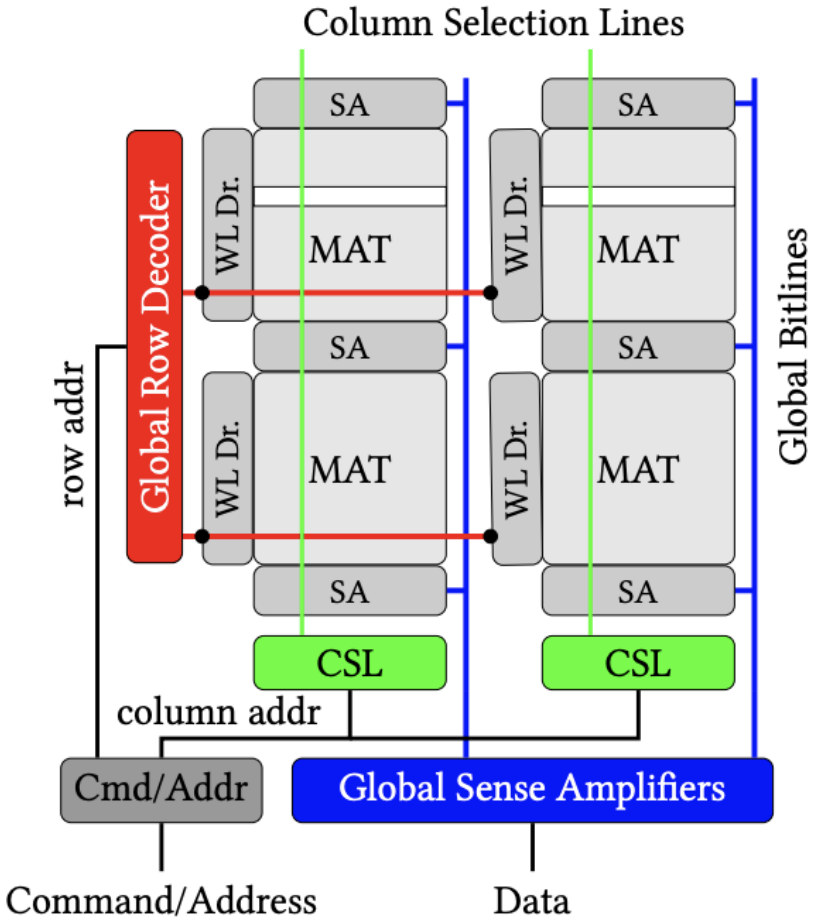


Figure 1. DRAM bank organization

Another View of a DRAM Bank



Logical Abstraction



Physical View

Seshadri+, "In-DRAM Bulk Bitwise Execution Engine," ADCOM 2020.

More on DRAM Basics & Organization

- Vivek Seshadri and Onur Mutlu,
  "In-DRAM Bulk Bitwise Execution Engine"
  Invited Book Chapter in Advances in Computers, 2020.
  [Preliminary arXiv version]

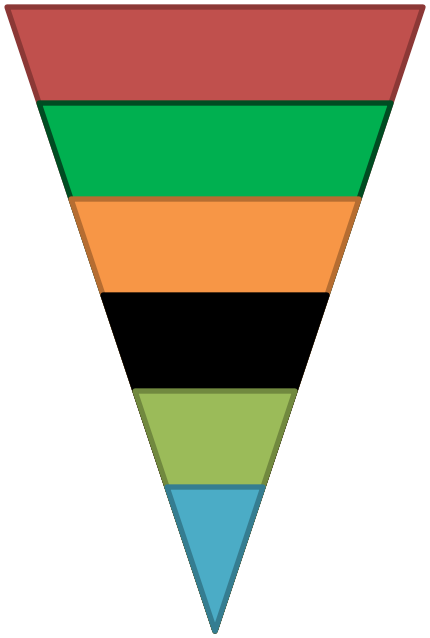
In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

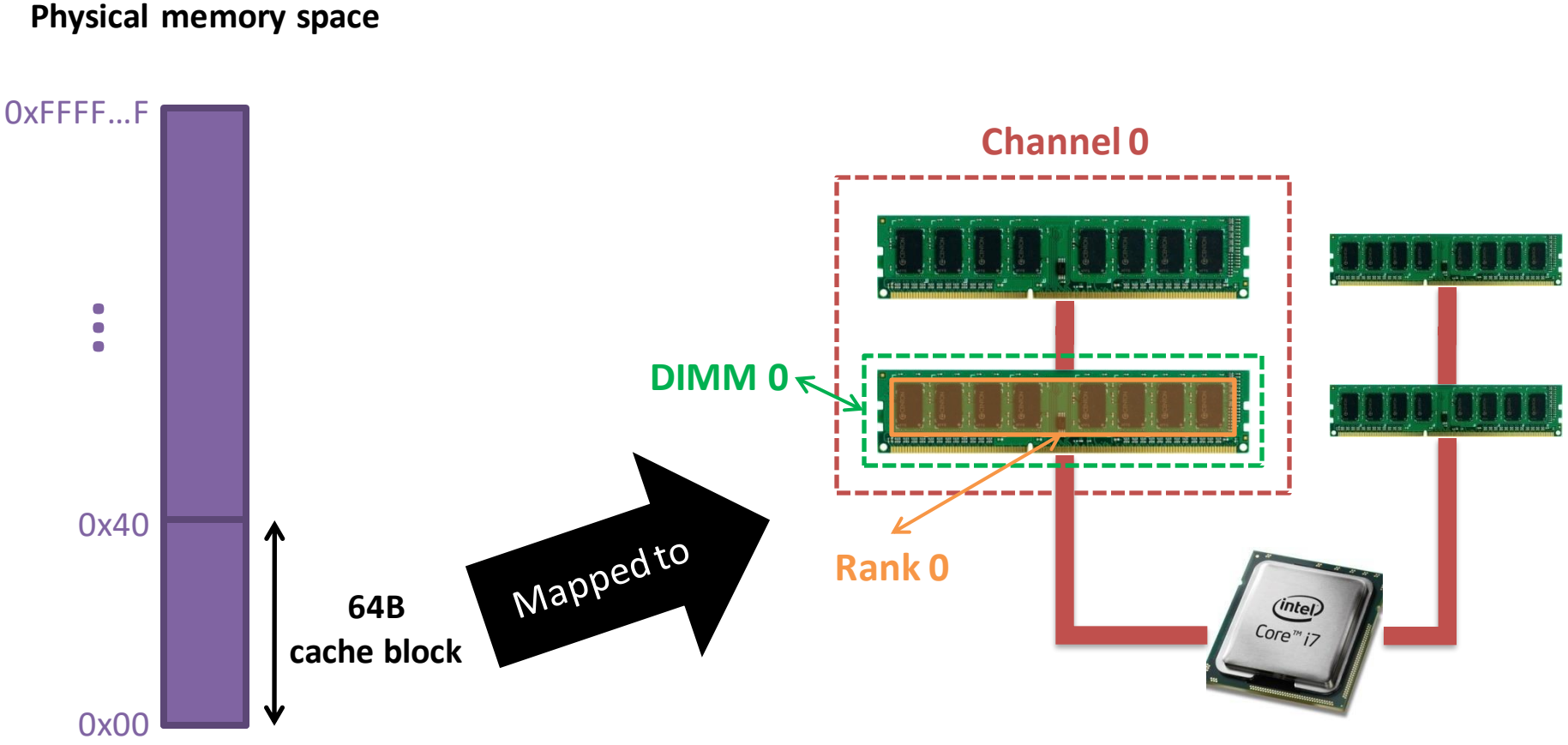
Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch

DRAM Subsystem Organization

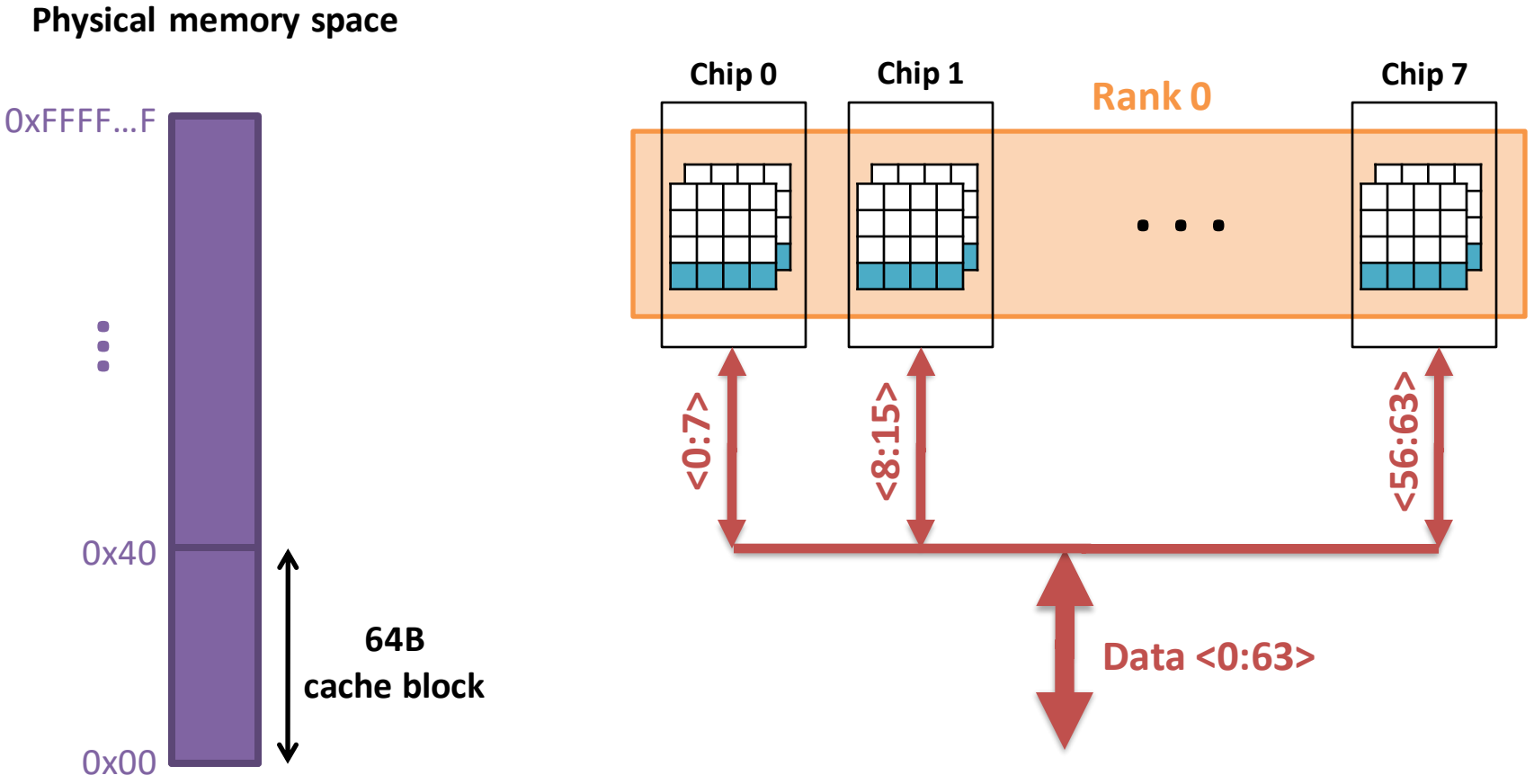
- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



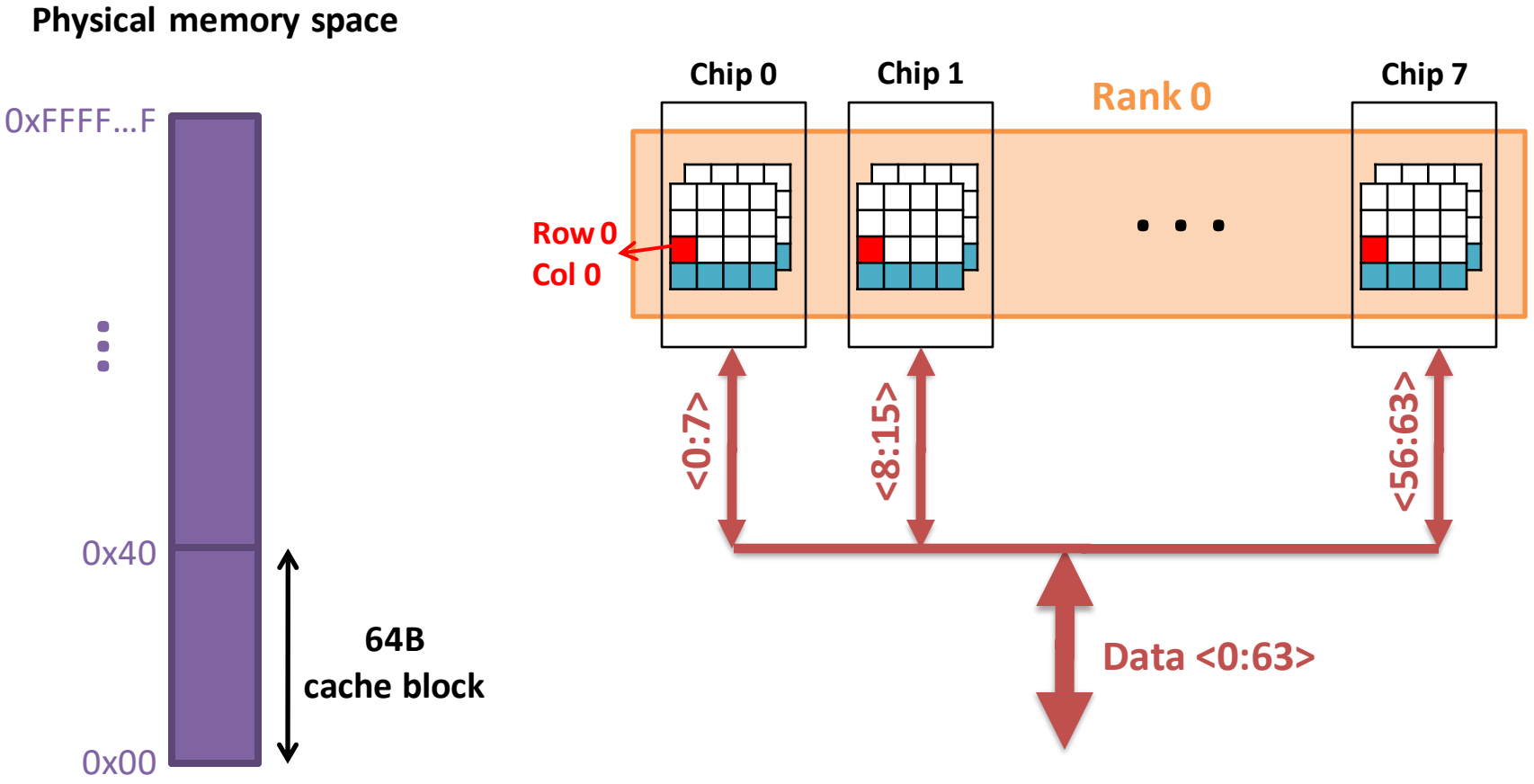
Example: Transferring a cache block



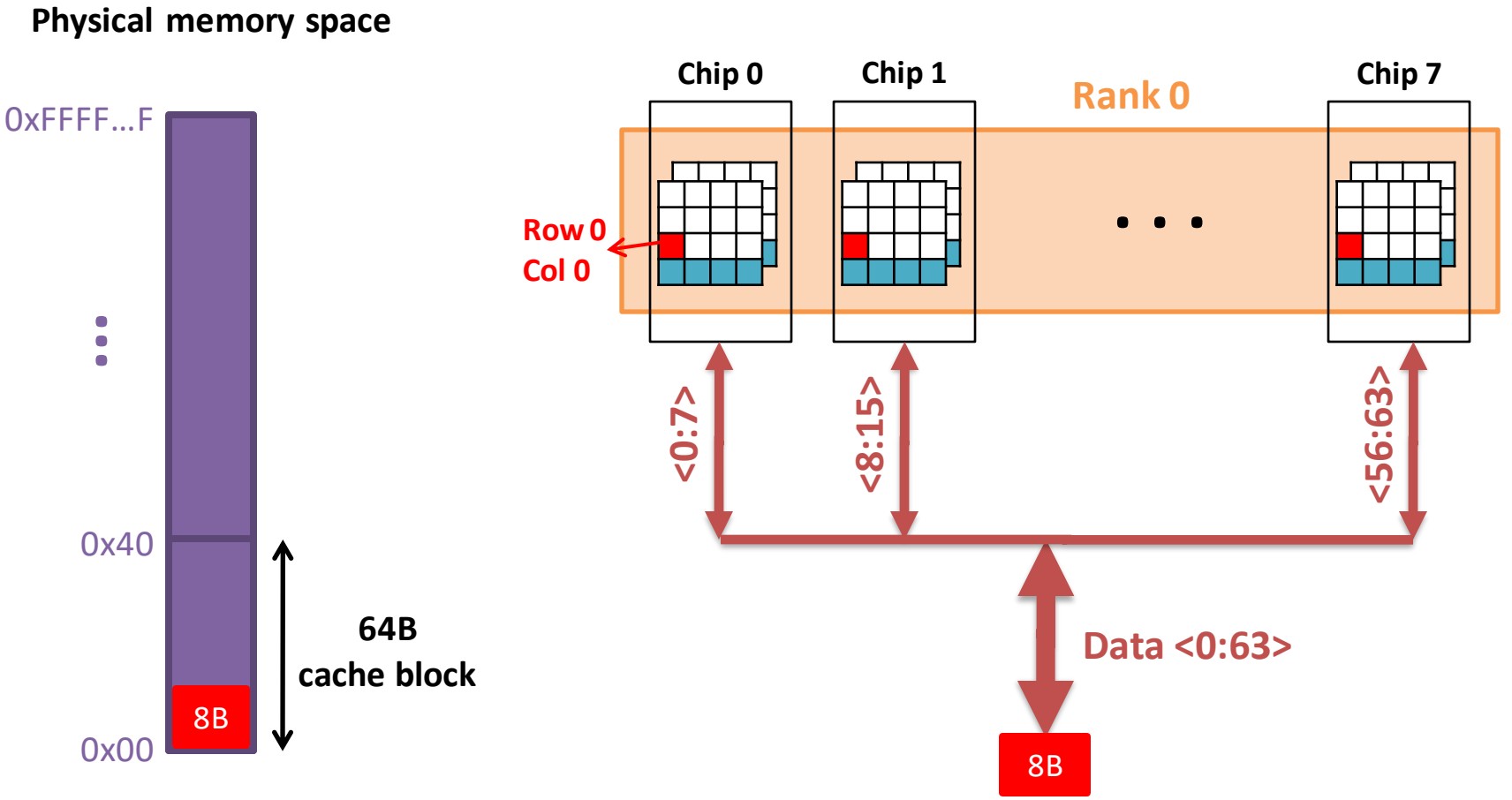
Example: Transferring a cache block



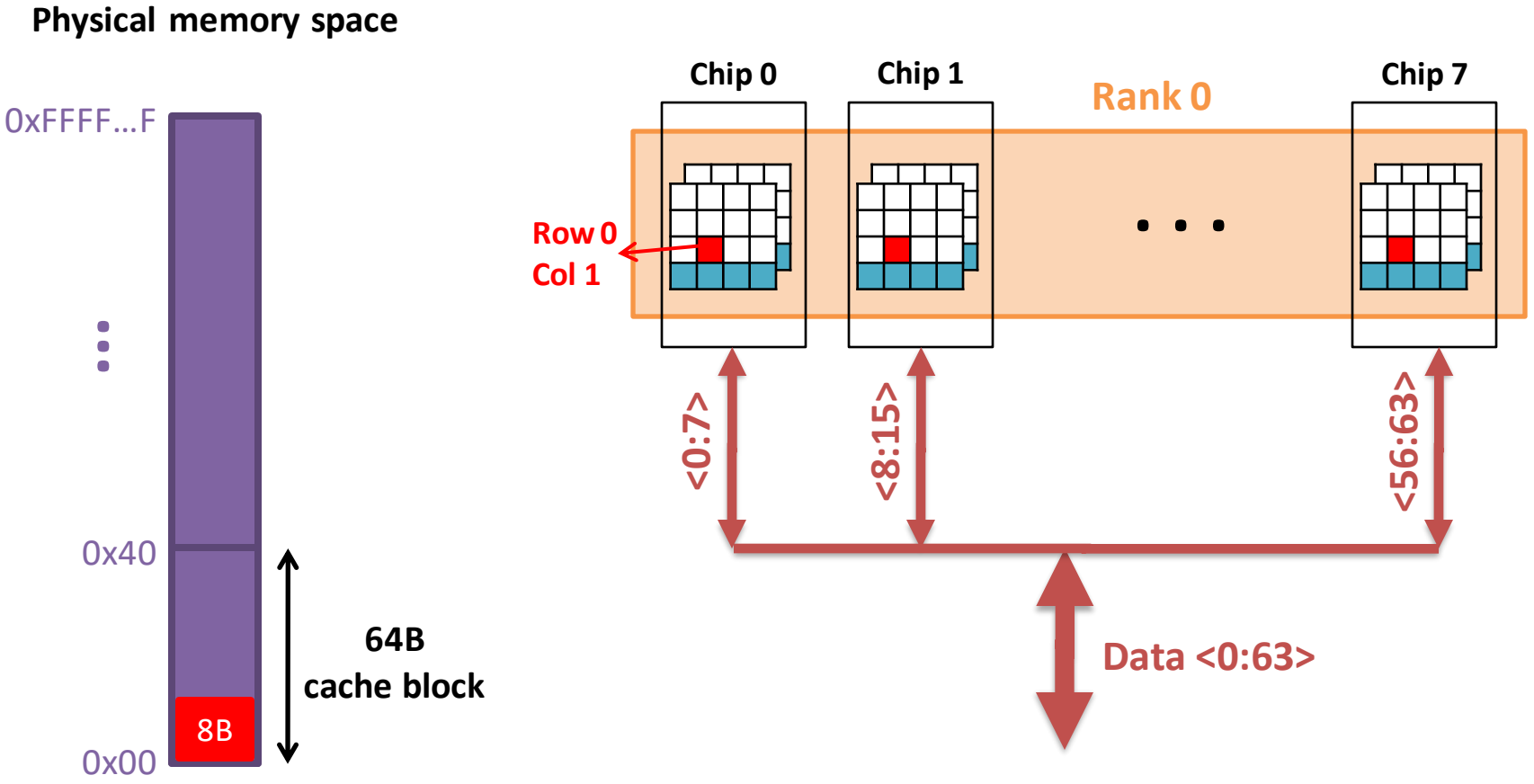
Example: Transferring a cache block



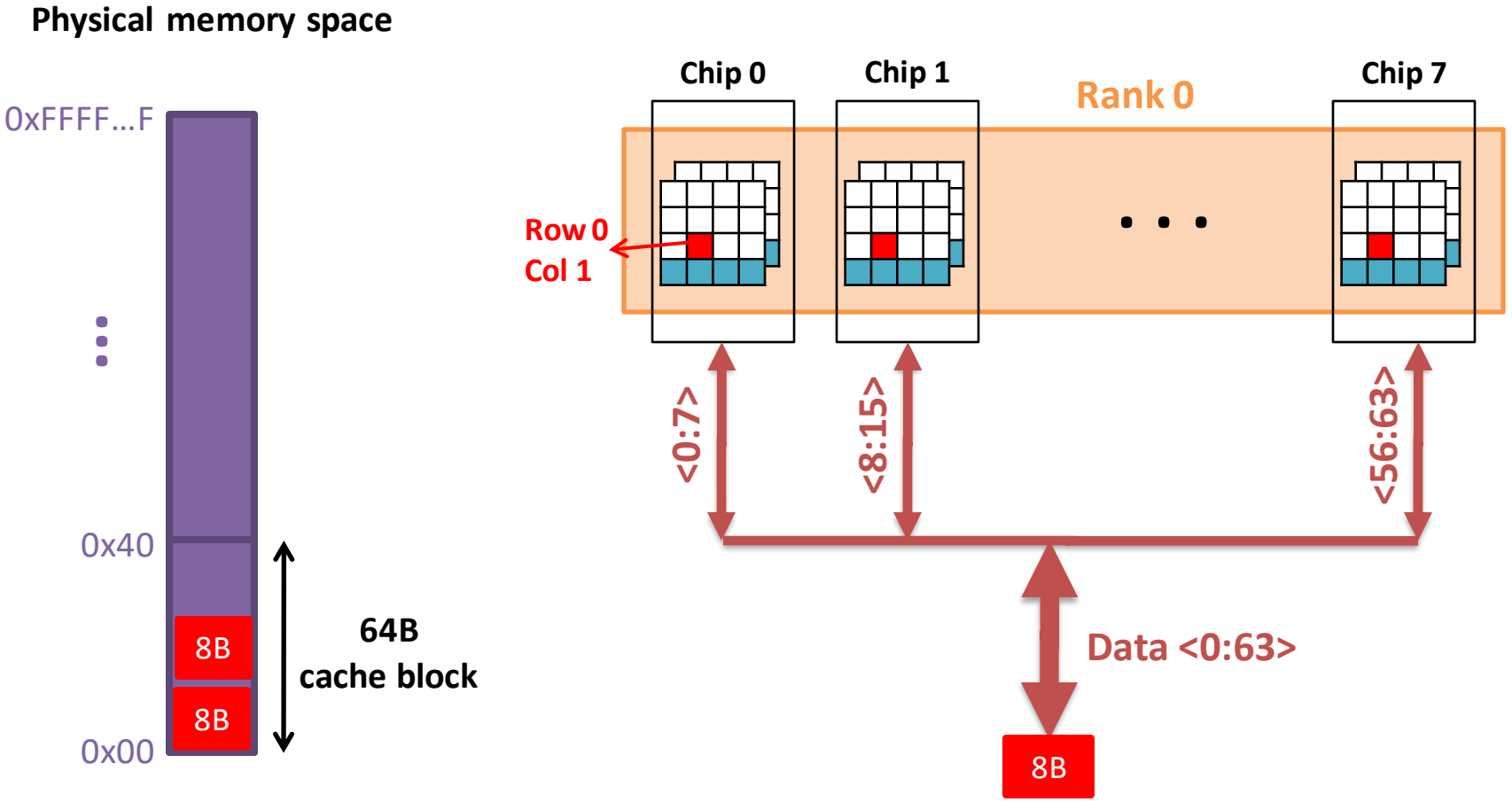
Example: Transferring a cache block



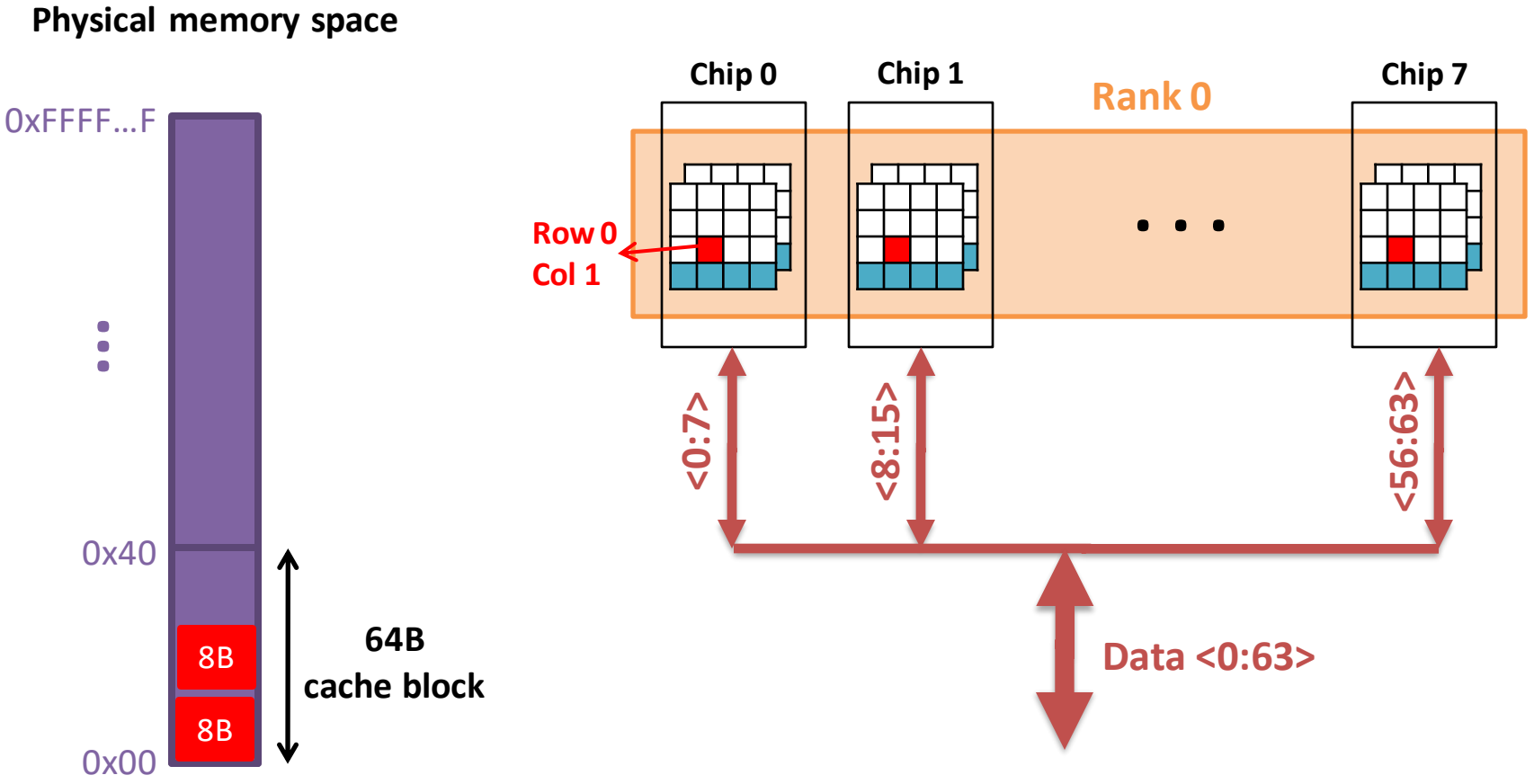
Example: Transferring a cache block



Example: Transferring a cache block



Example: Transferring a cache block



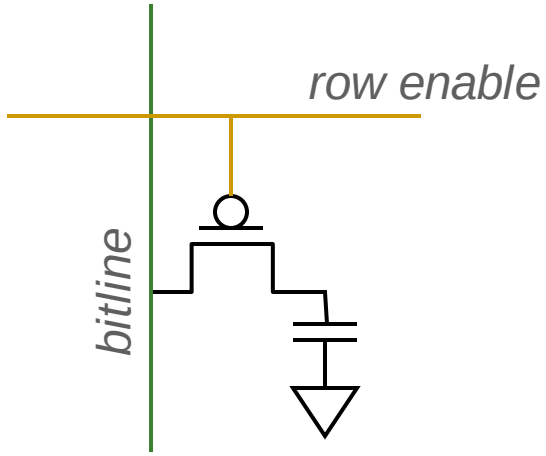
A 64B cache block takes 8 I/O cycles to transfer.

During the process, 8 columns are read sequentially.

Memory Technology: DRAM and SRAM

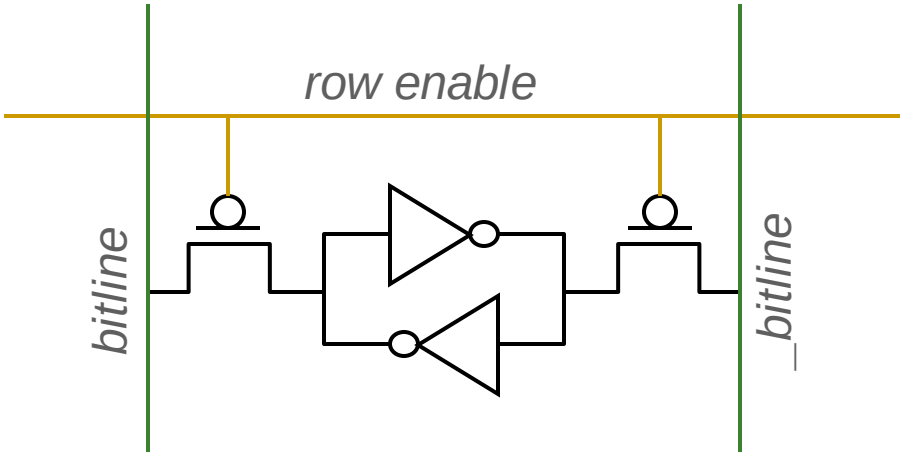
Memory Technology: DRAM

- Dynamic random access memory
- Capacitor charge state indicates stored value
  - Whether the capacitor is charged or discharged indicates storage of 1 or 0
  - 1 capacitor
  - 1 access transistor
- Capacitor leaks through the RC path
  - DRAM cell loses charge over time
  - DRAM cell needs to be refreshed

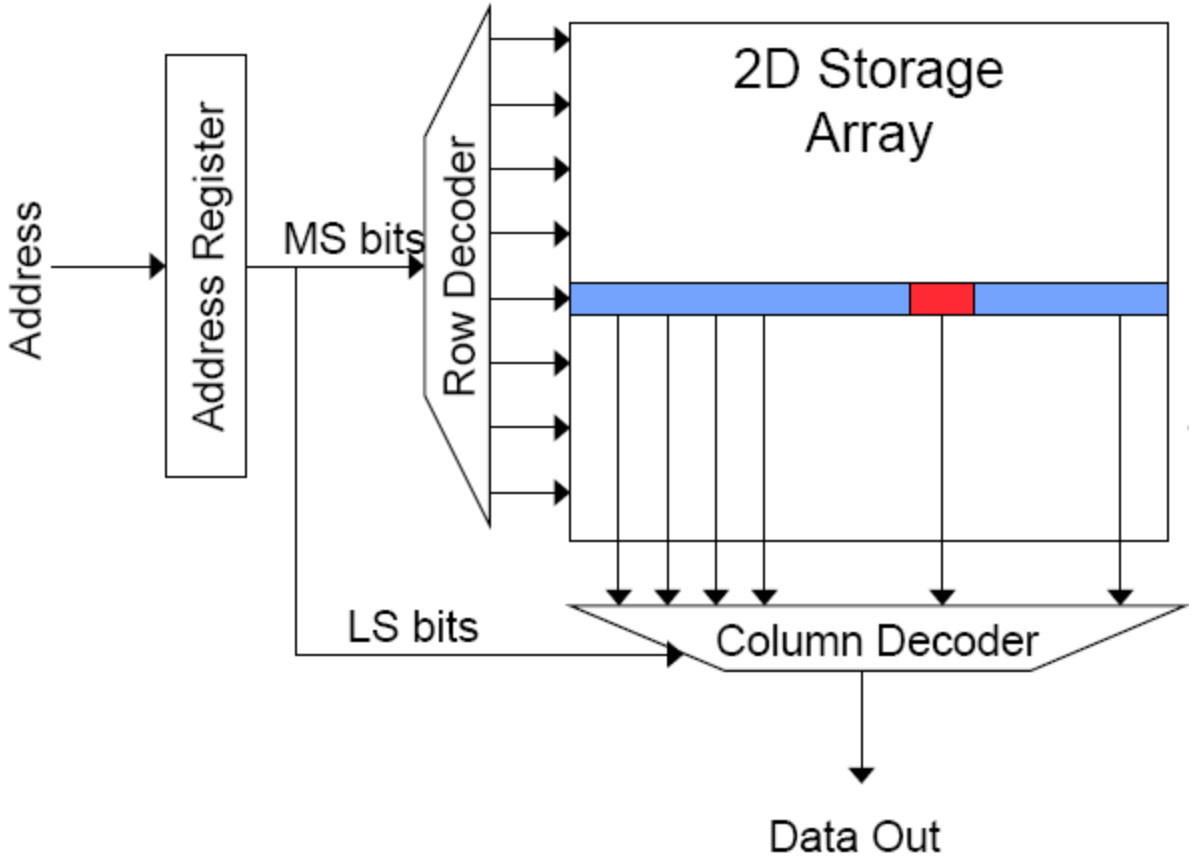


Memory Technology: SRAM

- Static random access memory
- Two cross coupled inverters store a single bit
  - Feedback path enables the stored value to persist in the "cell"
  - 4 transistors for storage
  - 2 transistors for access



Memory Bank Organization and Operation



- Read access sequence:

1. Decode row address & drive word-lines

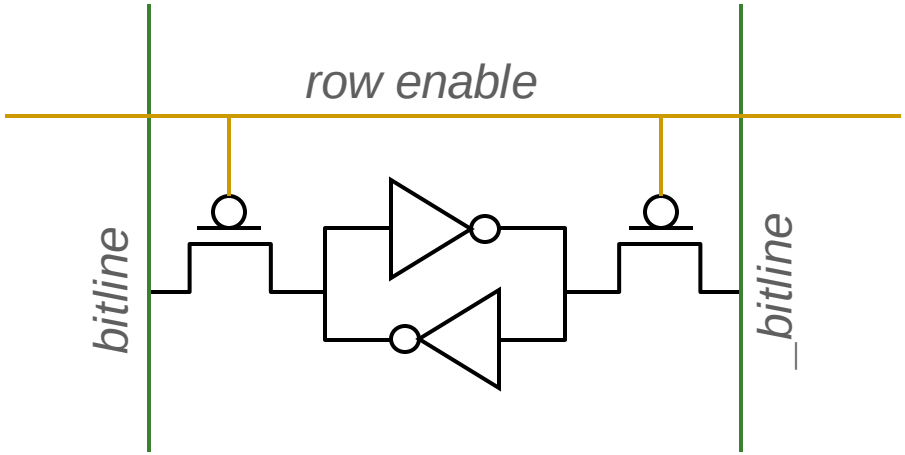
2. Selected bits drive bit-lines
   - Entire row read

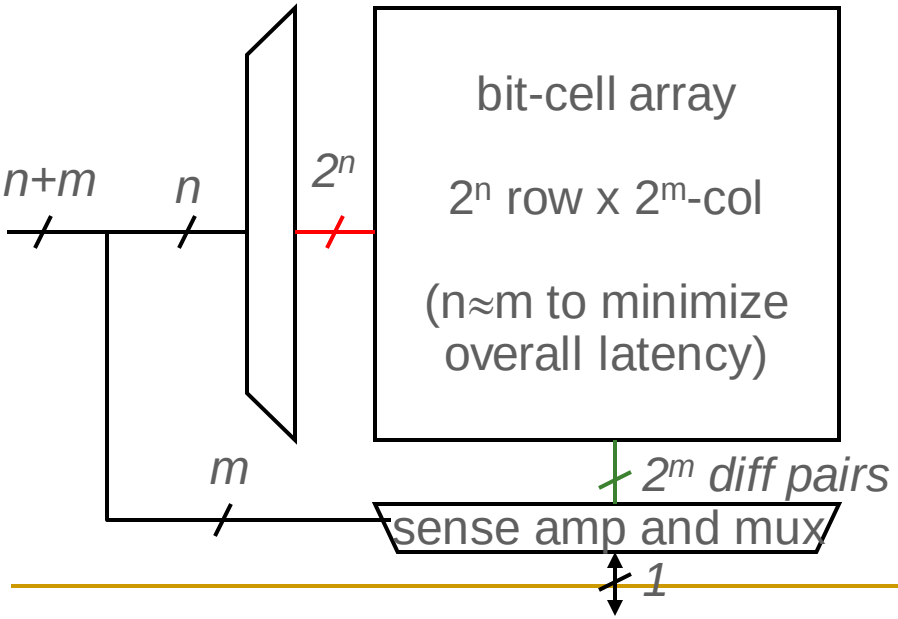
3. Amplify row data

4. Decode column address & select subset of row
   - Send to output

5. Precharge bit-lines
   - For next access

SRAM (Static Random Access Memory)





Read Sequence

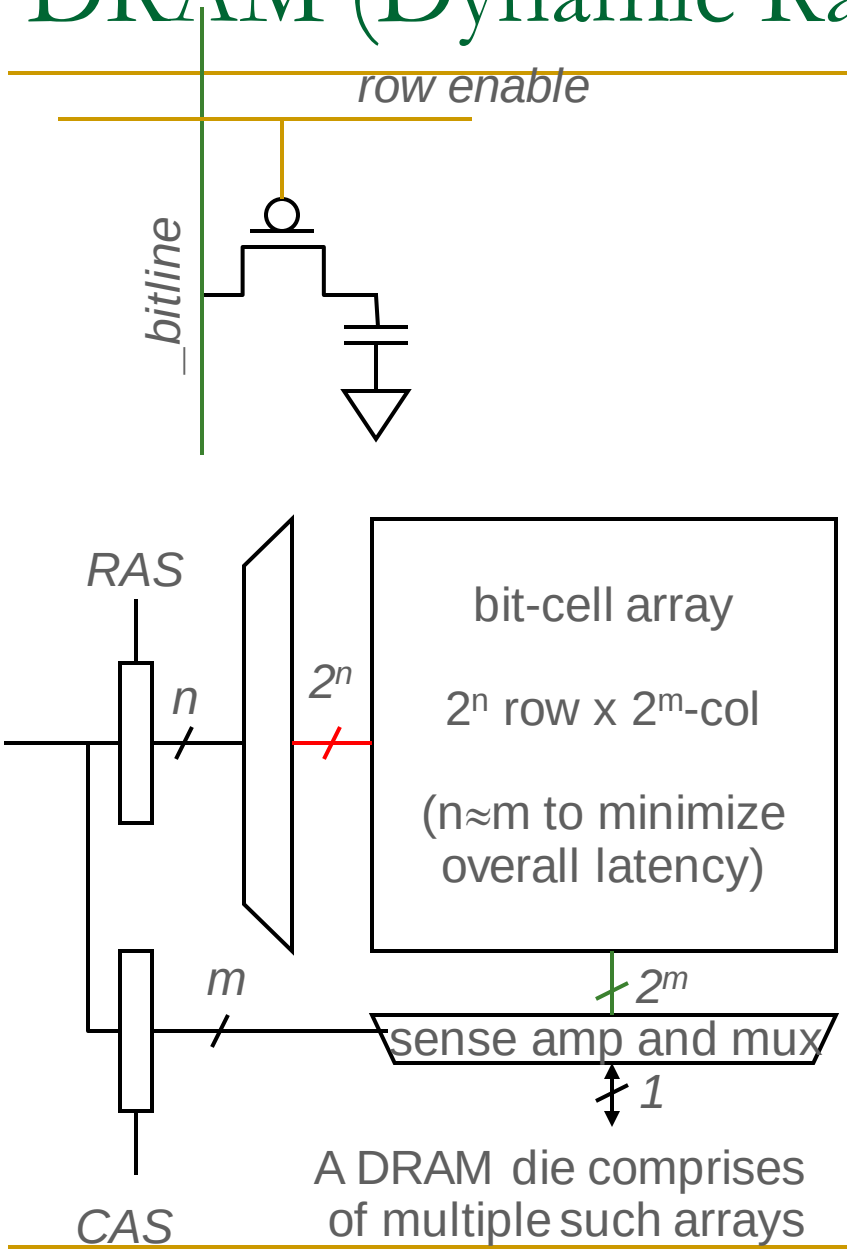
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. differential sensing and column select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

Access latency dominated by steps 2 and 3

Cycling time dominated by steps 2, 3 and 5

- step 2 proportional to 2^m
- step 3 and 5 proportional to 2^n

DRAM (Dynamic Random Access Memory)



Bit stored as charge on node capacitor (non-restorative)

- bit cell loses charge when read
- bit cell loses charge over time

Read Sequence

1~3 same as SRAM

4. a "flip-flopping" sense amp amplifies and regenerates the bitline, data bit is mux'ed out

5. precharge all bitlines

Destructive reads

Charge loss over time

Refresh: A DRAM controller must periodically read each row within the allowed refresh time (10s of ms) such that charge is restored

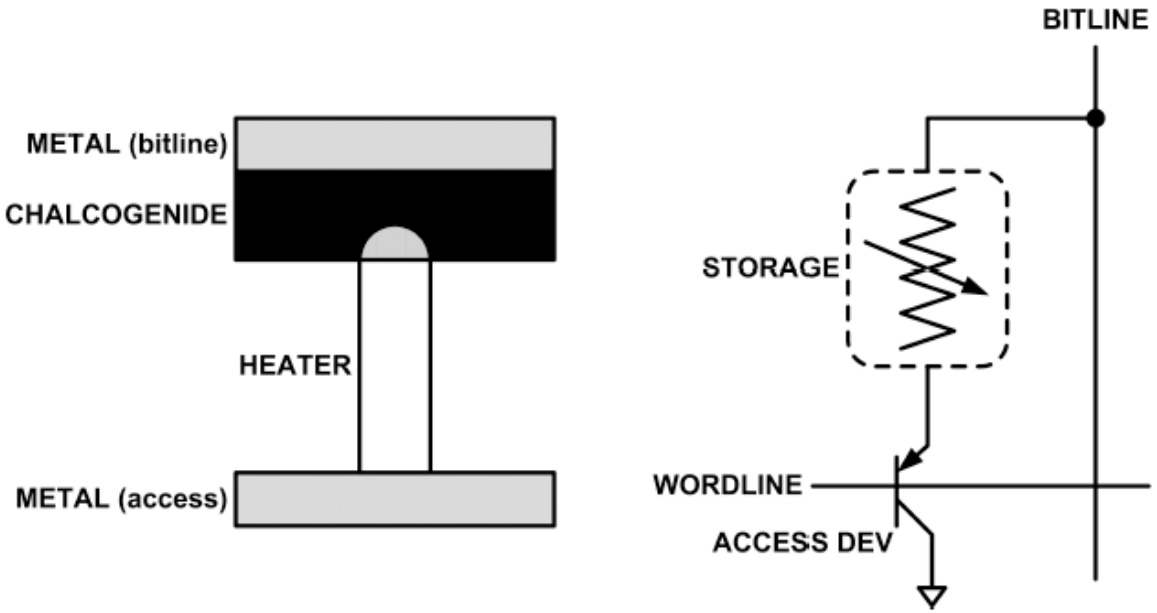
DRAM vs. SRAM

- DRAM
  - Slower access (capacitor)
  - Higher density (1T 1C cell)
  - Lower cost
  - Requires refresh (power, performance, circuitry)
  - Manufacturing requires putting capacitor and logic together

- SRAM
  - Faster access (no capacitor)
  - Lower density (6T cell)
  - Higher cost
  - No need for refresh
  - Manufacturing compatible with logic process (no capacitor)

An Aside: Phase Change Memory

- Phase change material (chalcogenide glass) exists in two states:
  - Amorphous: Low optical reflexivity and high electrical resistivity
  - Crystalline: High optical reflexivity and low electrical resistivity

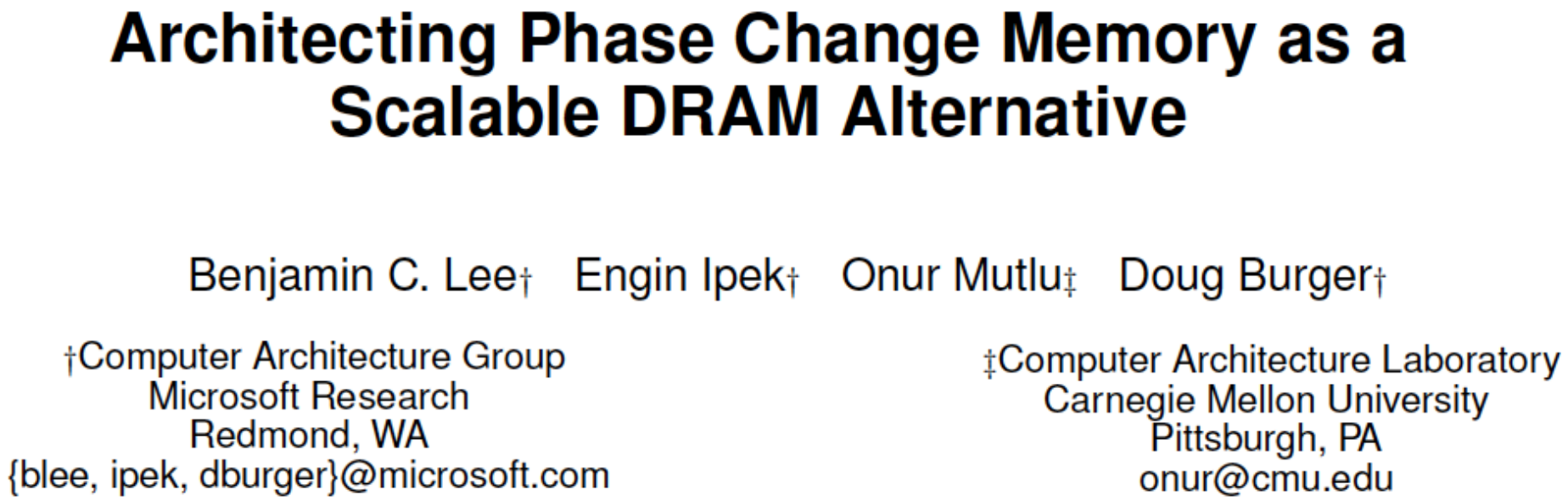


PCM is resistive memory: High resistance (0), Low resistance (1)

Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.

Reading: PCM As Main Memory

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, **"Architecting Phase Change Memory as a Scalable DRAM Alternative"**
  *Proceedings of the 36th International Symposium on Computer Architecture (**ISCA**)*, pages 2-13, Austin, TX, June 2009. Slides (pdf)



Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek† Onur Mutlu‡ Doug Burger†

†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

Reading: More on PCM As Main Memory

- Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger, **"Phase Change Technology and the Future of Main Memory"** *IEEE Micro, Special Issue: Micro's Top Picks from 2009 Computer Architecture Conferences (**MICRO TOP PICKS**)*, Vol. 30, No. 1, pages 60-70, January/February 2010.

Intel Optane Persistent Memory (2019)

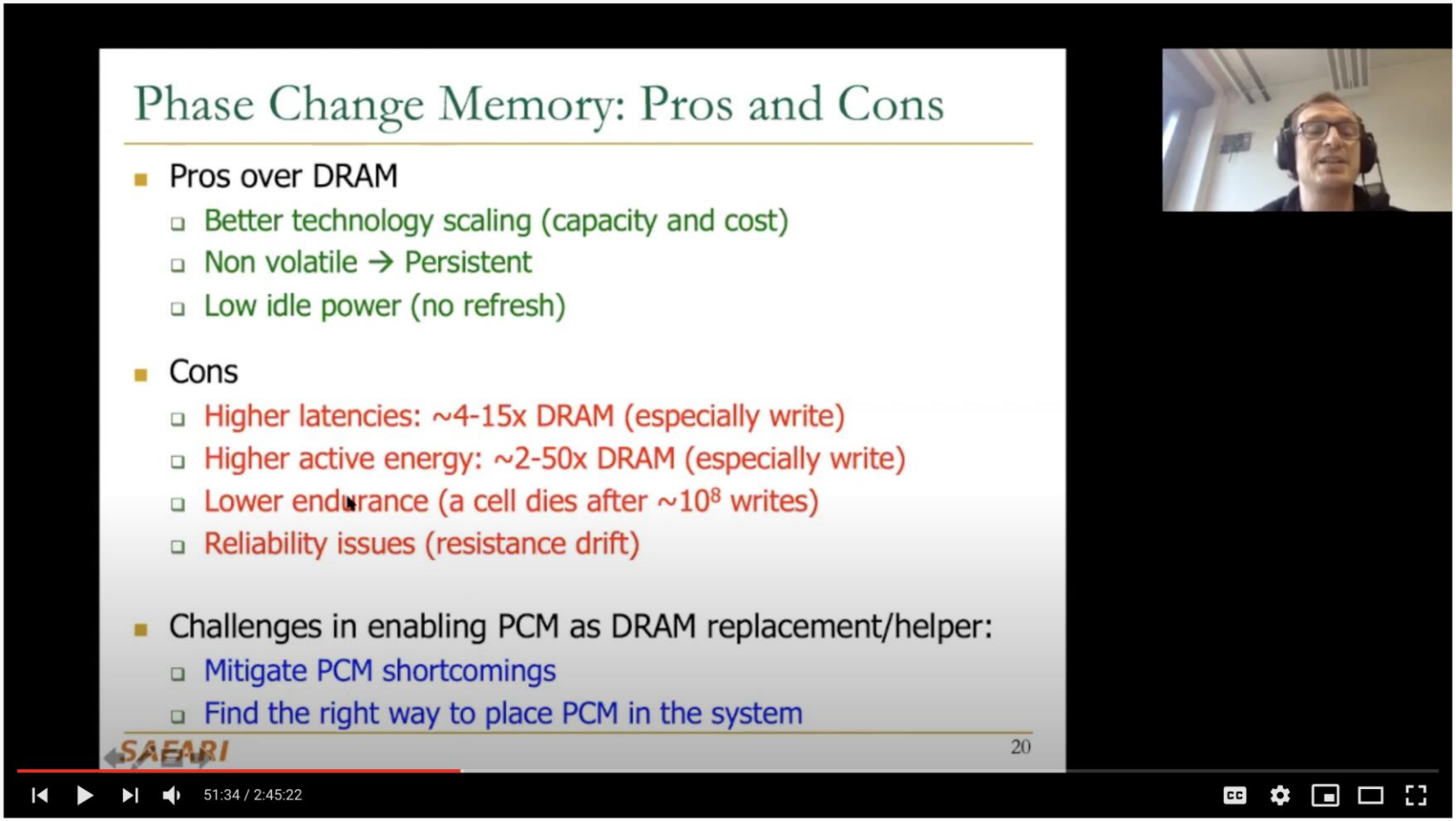
- Non-volatile main memory
- Based on 3D-XPoint Technology

https://www.storagereview.com/intel_optane_dc_persistent_memory_module_pmm

DRAM vs. PCM

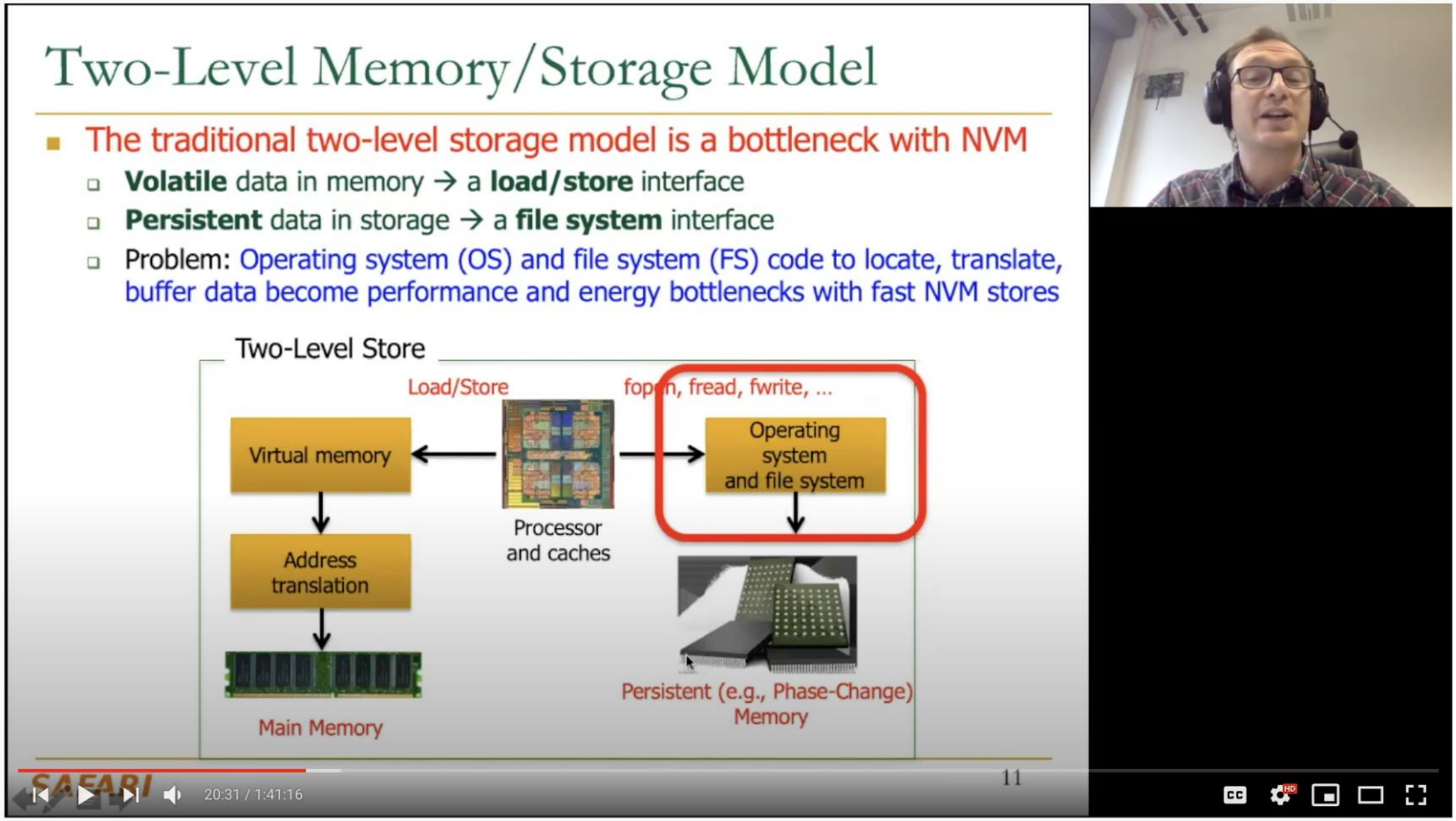
- DRAM
  - Faster access (capacitor)
  - Lower density (capacitor less scalable) → higher cost
  - Requires refresh (power, performance, circuitry)
  - Manufacturing requires putting capacitor and logic together
  - Volatile (loses data at loss of power)
  - No endurance problems
  - Lower access energy
- PCM
  - Slower access (no capacitor)
  - Higher density (phase change material more scalable) → lower cost
  - No need for refresh
  - Manufacturing requires less conventional processes – less mature
  - Non-volatile (does not lose data at loss of power)
  - Endurance problems (a cell cannot be used after N writes)
  - Higher access energy

More on Emerging Memory Technologies



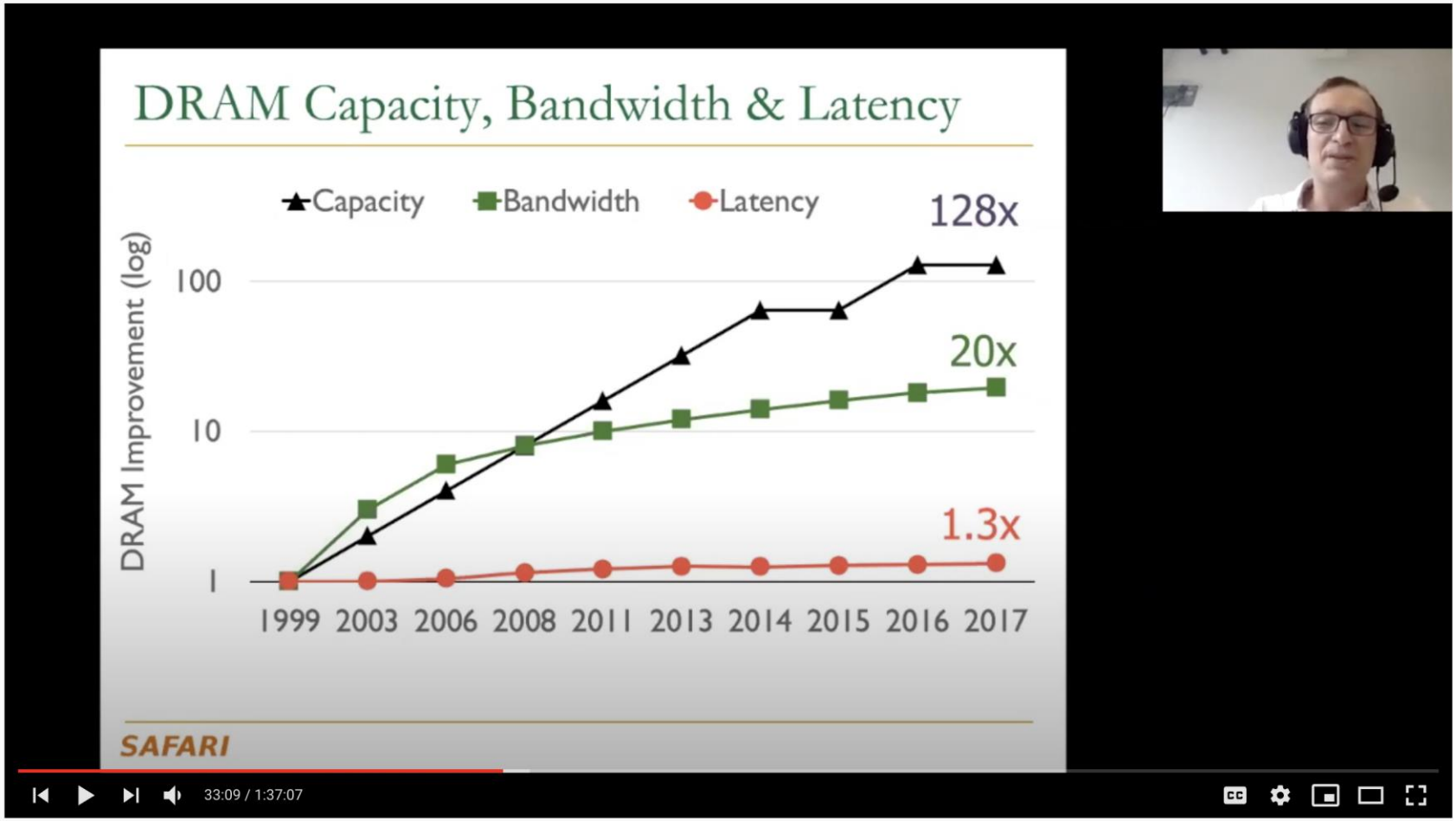
https://www.youtube.com/watch?v=AlE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28

More on Emerging Memory Technologies



https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29

More on Memory Technologies



https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29

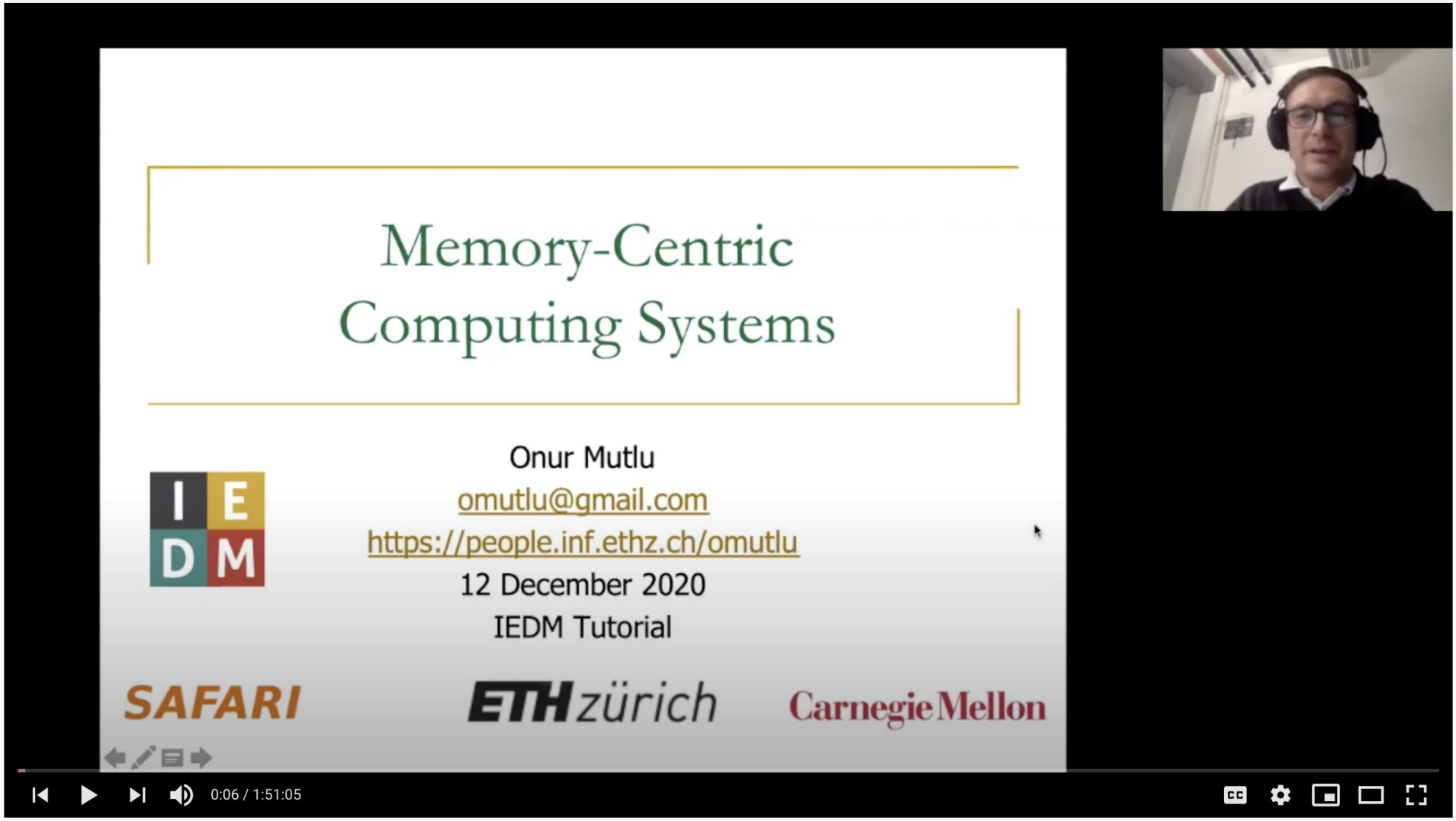
Lectures on Memory Technologies

- Computer Architecture, Fall 2020, Lecture 15
  - Emerging Memory Technologies (ETH, Fall 2020)
  - https://www.youtube.com/watch?v=AlE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28
- Computer Architecture, Fall 2020, Lecture 16a
  - Opportunities & Challenges of Emerging Memory Tech (ETH, Fall 2020)
  - https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29
- Computer Architecture, Fall 2020, Lecture 3b
  - Memory Systems: Challenges & Opportunities (ETH, Fall 2020)
  - https://www.youtube.com/watch?v=Q2FbUxD7GHs&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=6

A Tutorial on Memory-Centric Systems

- Onur Mutlu,
  "Memory-Centric Computing Systems"
  Invited Tutorial at *66th International Electron Devices Meeting (**IEDM**)*, Virtual, 12 December 2020.
  [Slides (pptx) (pdf)]
  [Executive Summary Slides (pptx) (pdf)]
  [Tutorial Video (1 hour 51 minutes)]
  [Executive Summary Video (2 minutes)]
  [Abstract and Bio]
  [Related Keynote Paper from VLSI-DAT 2020]
  [Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE



IEDM 2020 Tutorial: Memory-Centric Computing Systems, Onur Mutlu, 12 December 2020

1,641 views • Dec 23, 2020

48 0 SHARE SAVE

Onur Mutlu Lectures
13.9K subscribers

ANALYTICS EDIT VIDEO

https://www.youtube.com/onurmutlulectures

Digital Design & Computer Arch.

Lecture 22: Memory Overview, Organization & Technology

Prof. Onur Mutlu

ETH Zürich

Spring 2021

21 May 2021