18-447
Computer Architecture
Lecture 21: Main Memory

Prof. Onur Mutlu
Carnegie Mellon University
Spring 2015, 3/23/2015

Assignment Reminders

- Lab 6: Due April 3
  - C-level simulation of caches and branch prediction

- HW 5: Due March 29
  - Will be out later today

- Midterm II: TBD

- The course will move quickly in the last 1.5 months
  - Please manage your time well
  - Get help from the TAs during office hours and recitation sessions
  - The key is learning the material very well

Upcoming Seminar on Flash Memory (March 25)

- March 25, Wednesday, CIC Panther Hollow Room, 4-5pm
- Yixin Luo, PhD Student, CMU
- Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery

- Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu,
  "Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery"
  Proceedings of the
  21st International Symposium on High-Performance Computer Architecture (***HPCA***), Bay Area, CA, February 2015.
  [Slides (pptx) (pdf)]
  Best paper session.

Computer Architecture Seminars

- Seminars relevant to many topics covered in 447
  - Caching
  - DRAM
  - Multi-core systems
  - …
- List of past and upcoming seminars are here:
  - https://www.ece.cmu.edu/~calcm/doku.php?id=seminars:seminars
- You can subscribe to receive Computer Architecture related event announcements here:
  - https://sos.ece.cmu.edu/mailman/listinfo/calcm-list

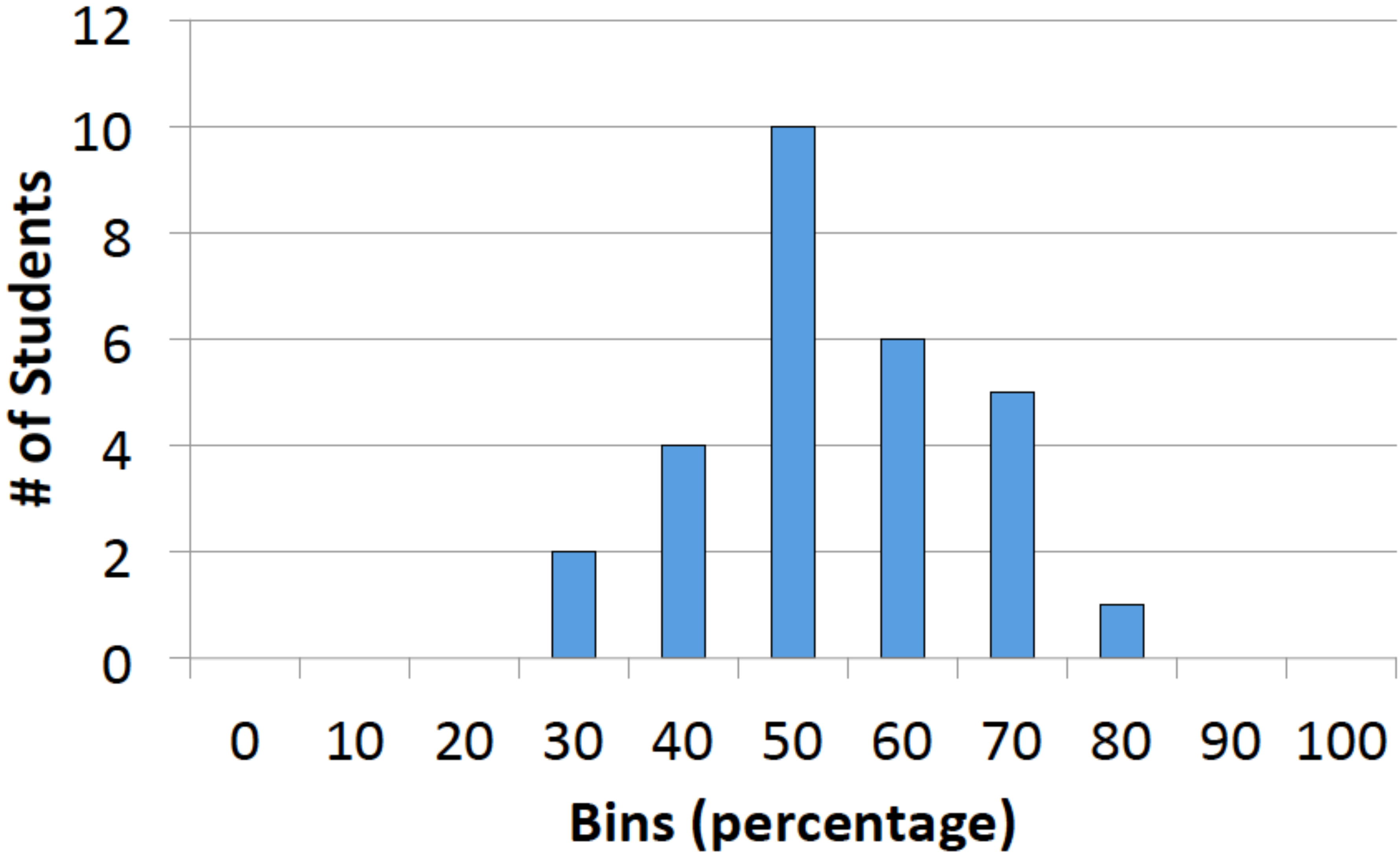
Midterm I Statistics: Average

- Out of 100:

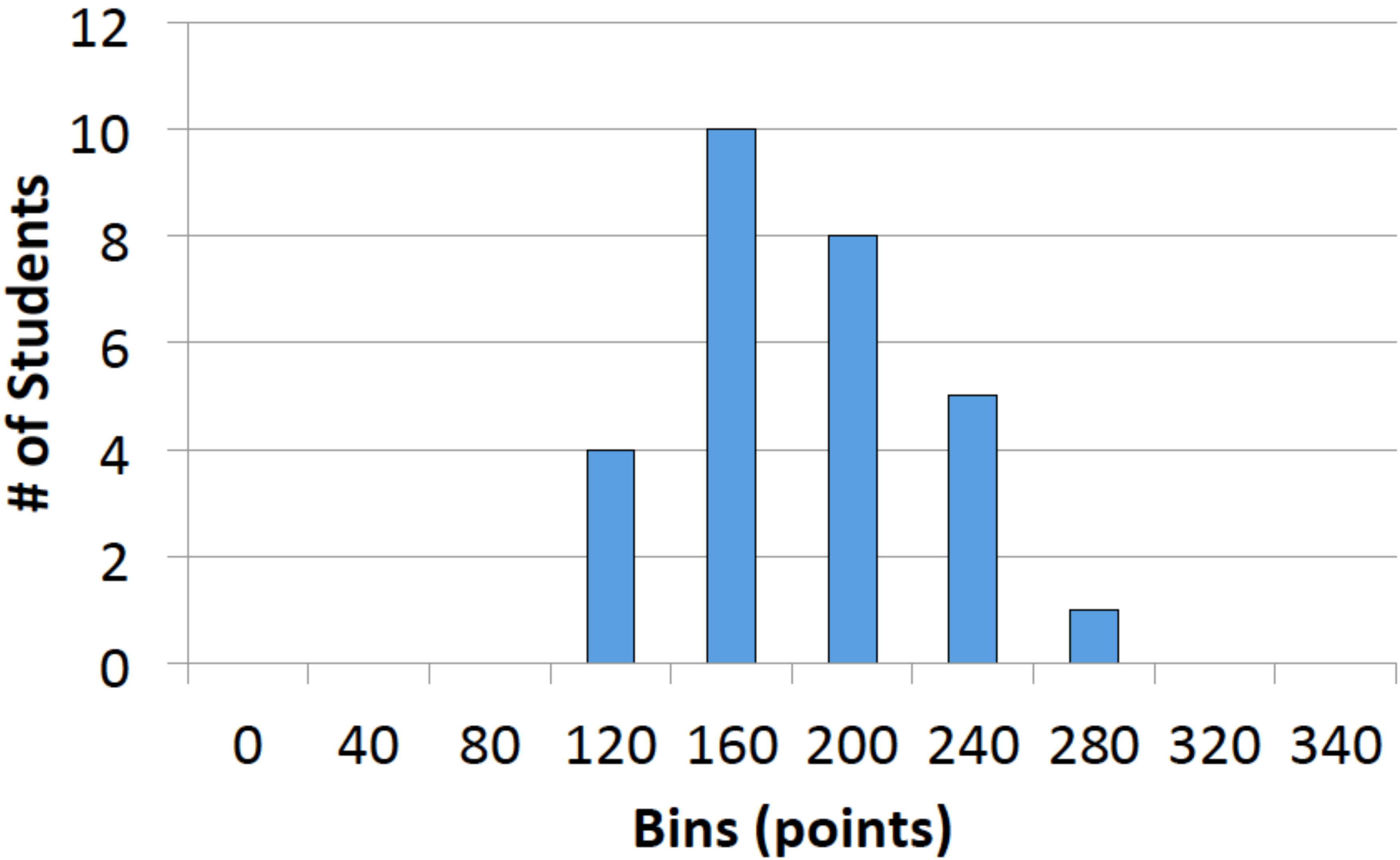
- MEAN 48.69
- MEDIAN 47.94
- STDEV 12.06

- MAX 76.18
- MIN 27.06

Midterm I Grade Distribution (Percentage)



Midterm I Grade Distribution (Absolute)



Grade Breakdowns per Question

- http://www.ece.cmu.edu/~ece447/s15/lib/exe/fetch.php?media=midterm_distribution.pdf

Going Forward

- What really matters is learning
  - And using the knowledge, skills, and ability to process information in the future
  - Focus less on grades, and put more weight into understanding

- Midterm I is only 12% of your entire course grade
  - Worth less than 2 labs + extra credit

- There are still Midterm II, Final, 3 Labs and 3 Homeworks
- There are many extra credit opportunities (great for learning by exploring your creativity)

Lab 3 Extra Credit Recognitions

- 4.00 bmperez (Brandon Perez)
- 3.75 junhanz (Junhan Zhou)
- 3.75 zzhao1 (Zhipeng Zhao)
- 2.50 terencea (Terence An)
- 2.25 rohitban (Rohit Banerjee)

Where We Are in Lecture Schedule

- The memory hierarchy
- Caches, caches, more caches
- Virtualizing the memory hierarchy: Virtual Memory
- Main memory: DRAM
- Main memory control, scheduling
- Memory latency tolerance techniques
- Non-volatile memory

- Multiprocessors
- Coherence and consistency
- Interconnection networks
- Multi-core issues

Main Memory

Required Reading (for the Next Few Lectures)

- Onur Mutlu, Justin Meza, and Lavanya Subramanian, **"The Main Memory System: Challenges and Opportunities"**
  Invited Article in
  Communications of the Korean Institute of Information Scientists and Engineers (***KIISE***), 2015.

Required Readings on DRAM

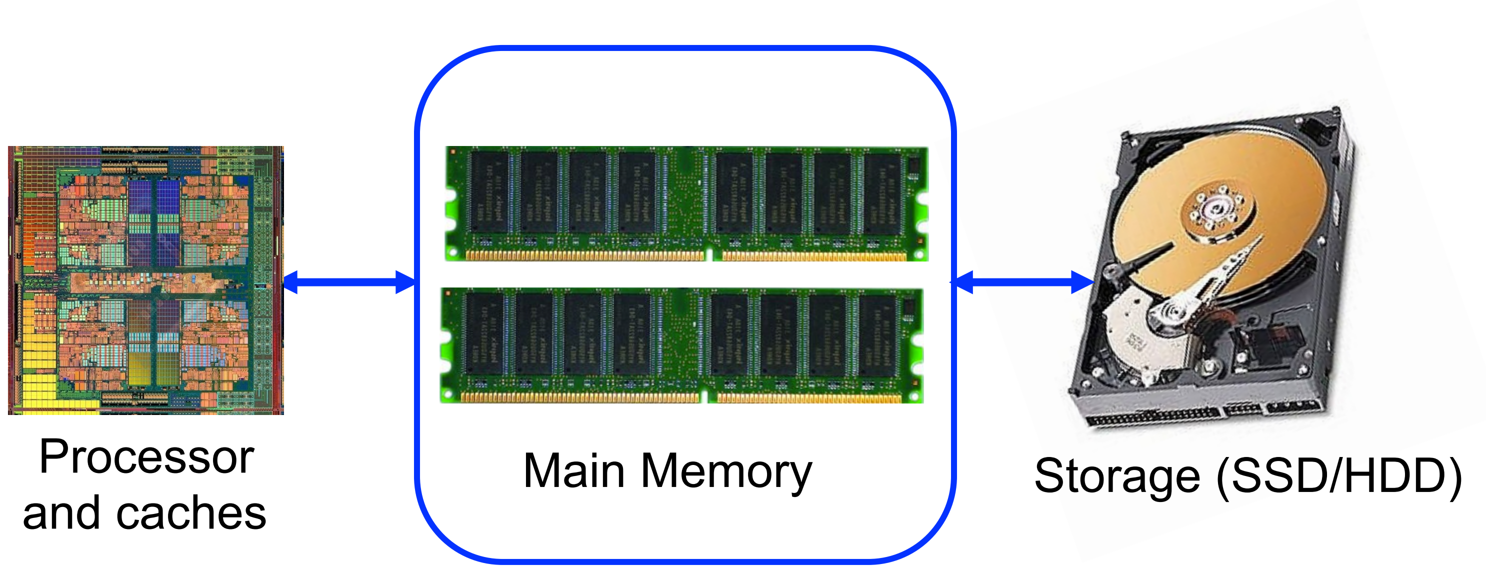
- DRAM Organization and Operation Basics
  - Sections 1 and 2 of: Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.
    http://users.ece.cmu.edu/~omutlu/pub/tldram_hpca13.pdf

  - Sections 1 and 2 of Kim et al., "A Case for Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
    http://users.ece.cmu.edu/~omutlu/pub/salp-dram_isca12.pdf

- DRAM Refresh Basics
  - Sections 1 and 2 of Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
    http://users.ece.cmu.edu/~omutlu/pub/raidr-dram-refresh_isca12.pdf

Why Is Memory So Important? (Especially Today)

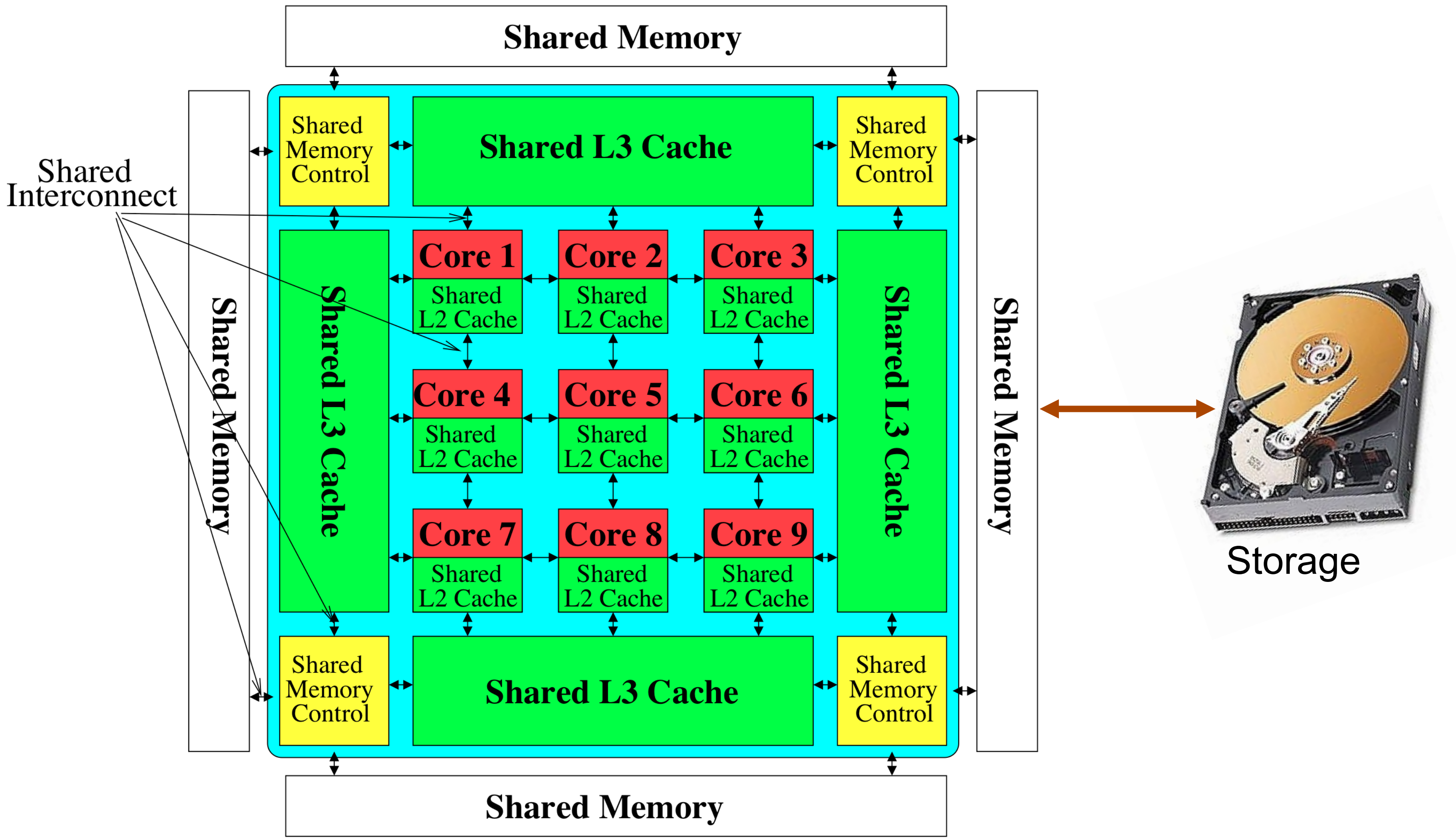
The Main Memory System



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

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

Memory System: A *Shared Resource* View



State of the Main Memory System

- Recent technology, architecture, and application trends
  - lead to new requirements
  - exacerbate old requirements

- DRAM and memory controllers, as we know them today, are (will be) unlikely to satisfy all requirements

- Some emerging non-volatile memory technologies (e.g., PCM) enable new opportunities: memory+storage merging

- We need to rethink/reinvent the main memory system
  - to fix DRAM issues and enable emerging technologies
  - to satisfy all requirements

Major Trends Affecting Main Memory (I)

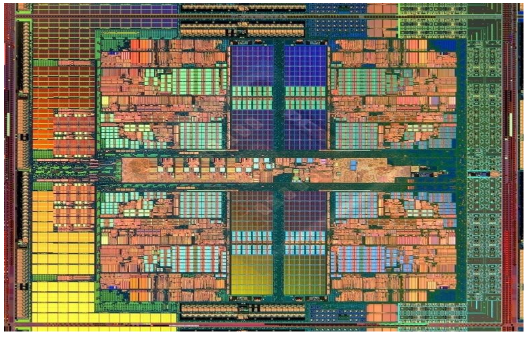
- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern

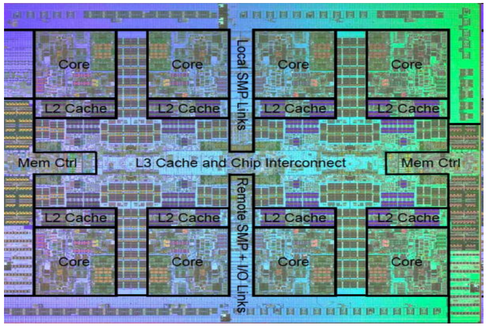
- DRAM technology scaling is ending

Demand for Memory Capacity

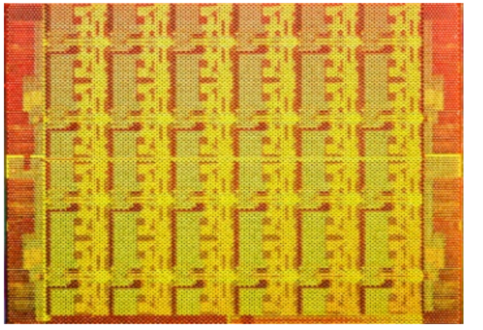
- More cores → More concurrency → Larger working set



AMD Barcelona: 4 cores



IBM Power7: 8 cores



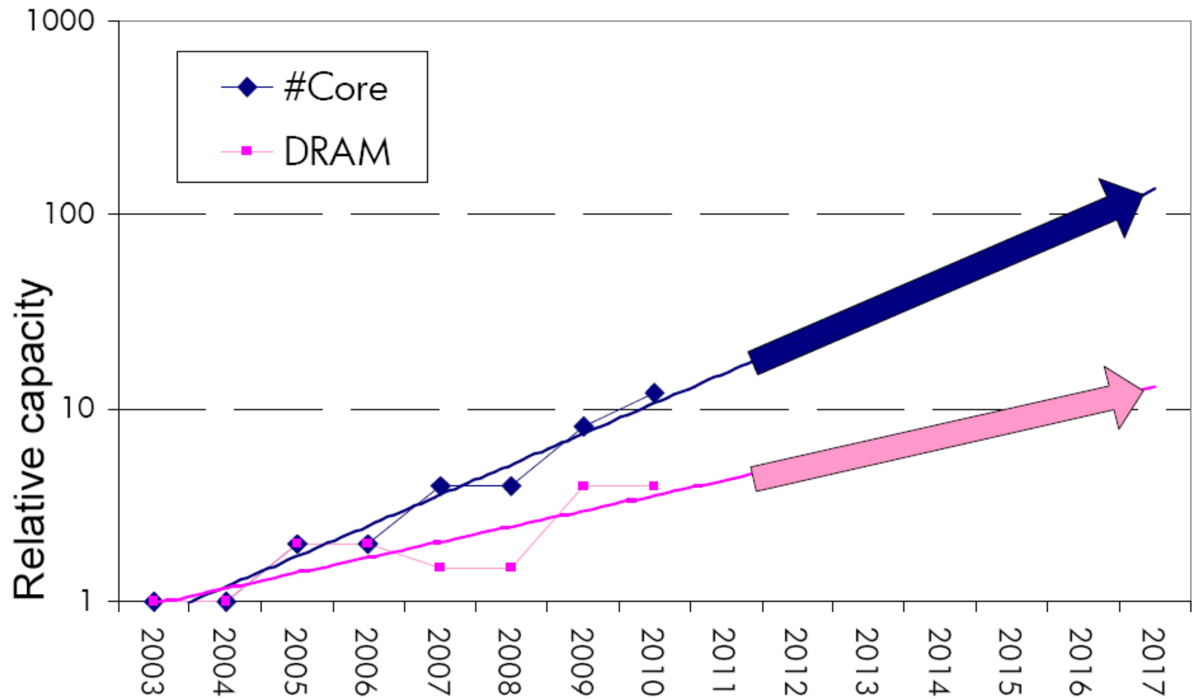
Intel SCC: 48 cores

- Modern applications are (increasingly) data-intensive

- Many applications/virtual machines (will) share main memory
  - Cloud computing/servers: Consolidation to improve efficiency
  - GP-GPUs: Many threads from multiple parallel applications
  - Mobile: Interactive + non-interactive consolidation
  - ...

Example: The Memory Capacity Gap

Core count doubling ~ every 2 years
DRAM DIMM capacity doubling ~ every 3 years



- *Memory capacity per core* expected to drop by 30% every two years
- Trends worse for *memory bandwidth per core*!

Major Trends Affecting Main Memory (II)

- Need for main memory capacity, bandwidth, QoS increasing
  - Multi-core: increasing number of cores/agents
  - Data-intensive applications: increasing demand/hunger for data
  - Consolidation: Cloud computing, GPUs, mobile, heterogeneity

- Main memory energy/power is a key system design concern

- DRAM technology scaling is ending

Major Trends Affecting Main Memory (III)

- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern
  - IBM servers: ~50% energy spent in off-chip memory hierarchy [Lefurgy, IEEE Computer 2003]
  - DRAM consumes power when idle and needs periodic refresh

- DRAM technology scaling is ending

Major Trends Affecting Main Memory (IV)

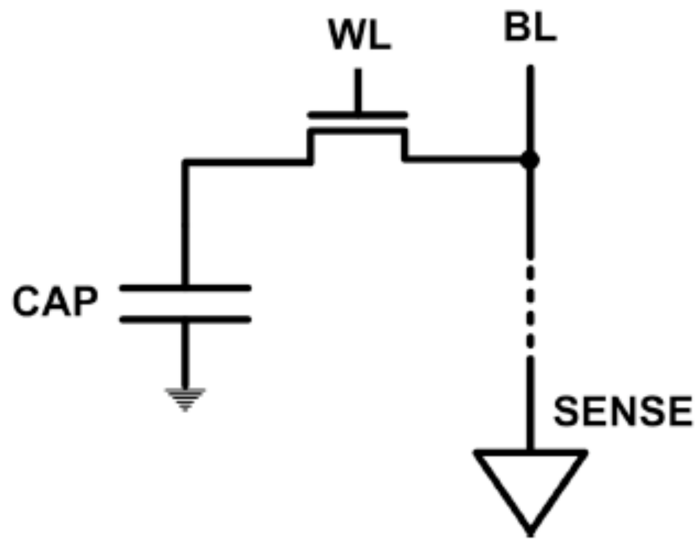
- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern

- DRAM technology scaling is ending
  - ITRS projects DRAM will not scale easily below X nm
  - Scaling has provided many benefits:
    - higher capacity, higher density, lower cost, lower energy

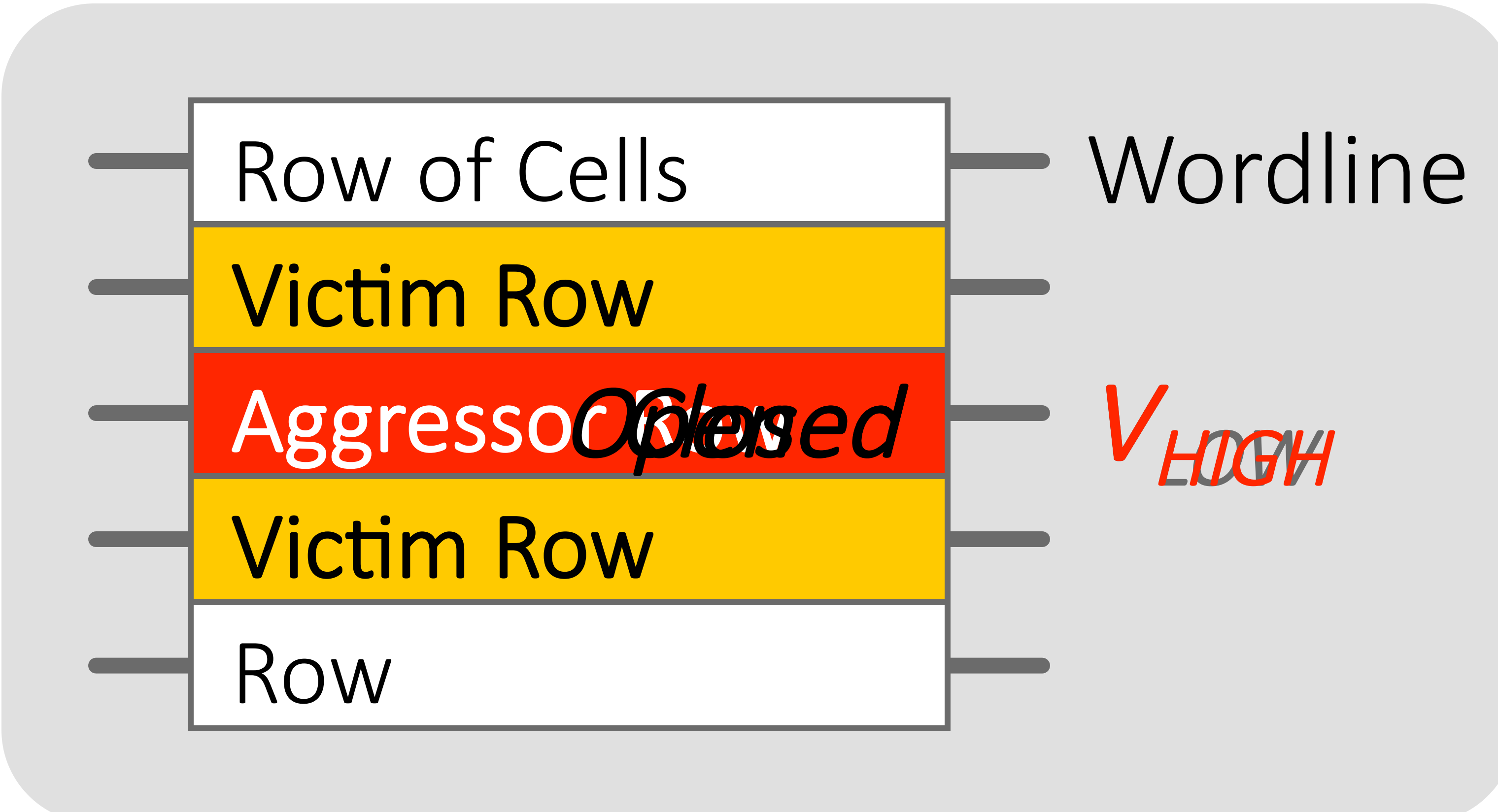
The DRAM Scaling Problem

- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



- DRAM capacity, cost, and energy/power hard to scale

Evidence of the DRAM Scaling Problem

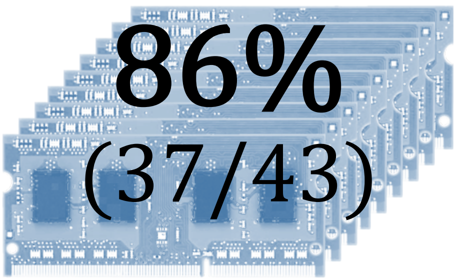


Repeatedly opening and closing a row enough times within a refresh interval induces disturbance errors in adjacent rows in most real DRAM chips you can buy today

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

Most DRAM Modules Are At Risk

A company

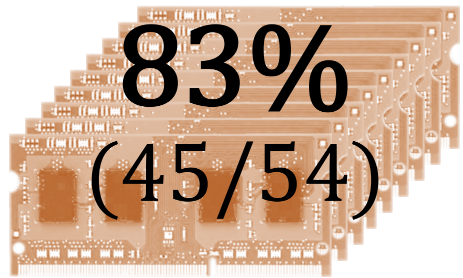


Up to

1.0×10^7

errors

B company

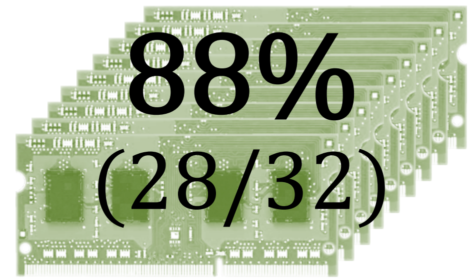


Up to

2.7×10^6

errors

C company



Up to

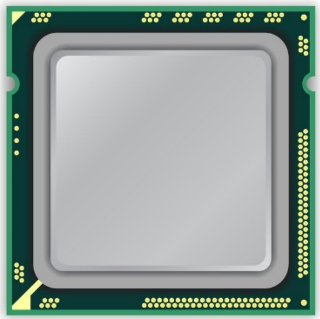
3.3×10^5

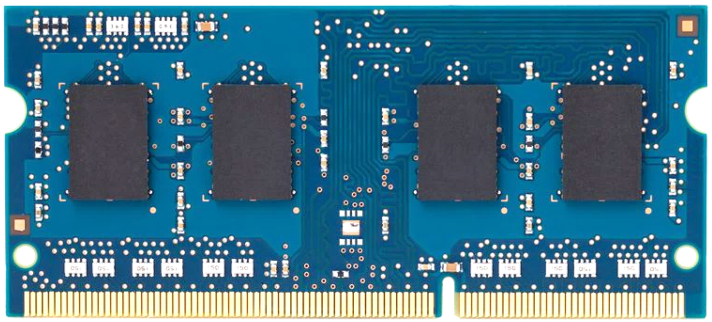
errors

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

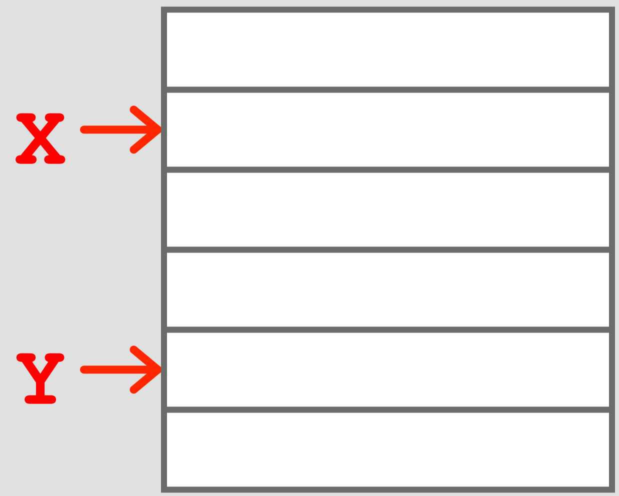
x86 CPU

DRAM Module

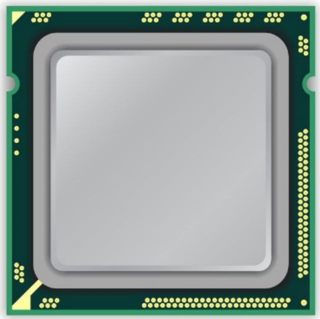




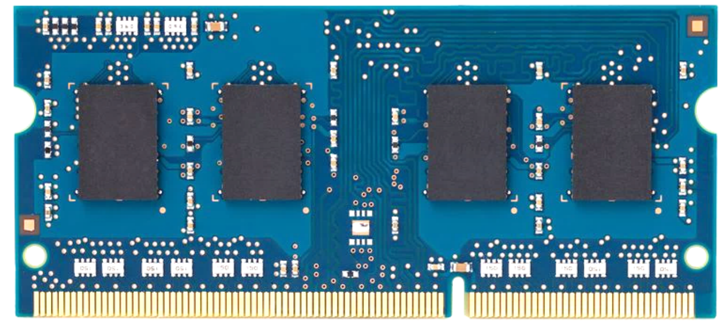
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



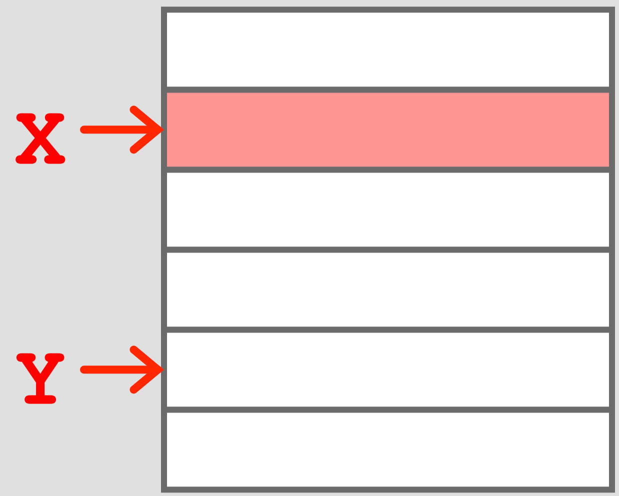
x86 CPU



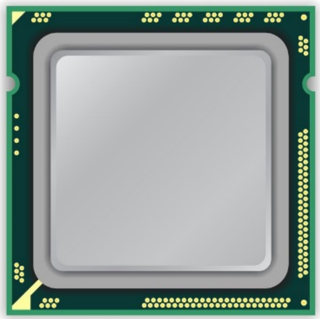
DRAM Module



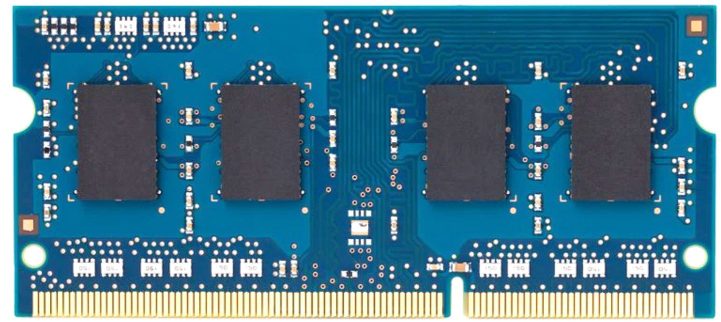
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



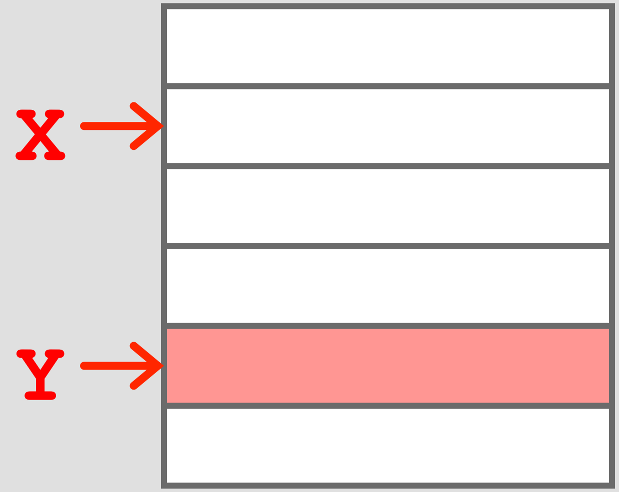
x86 CPU



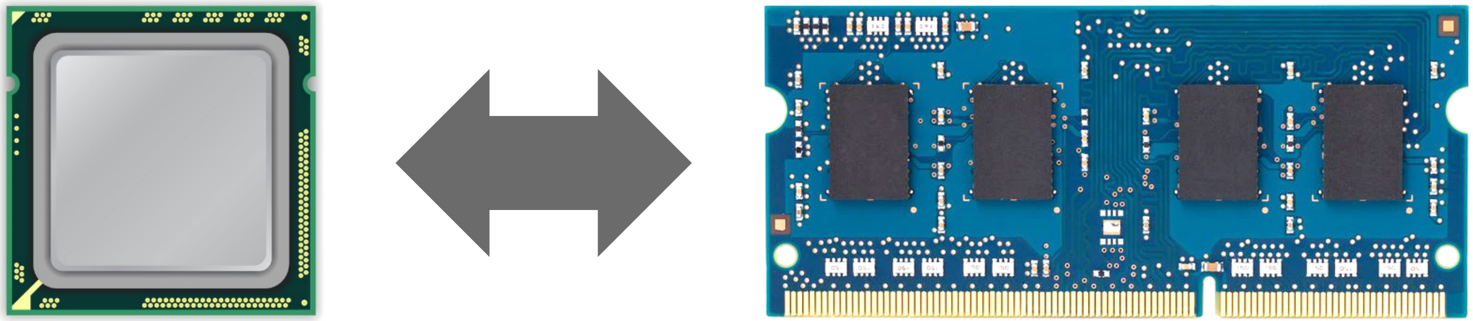
DRAM Module



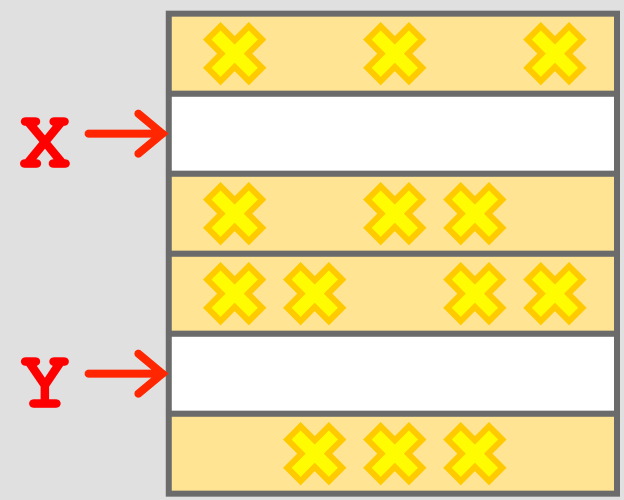
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



x86 CPU   DRAM Module



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



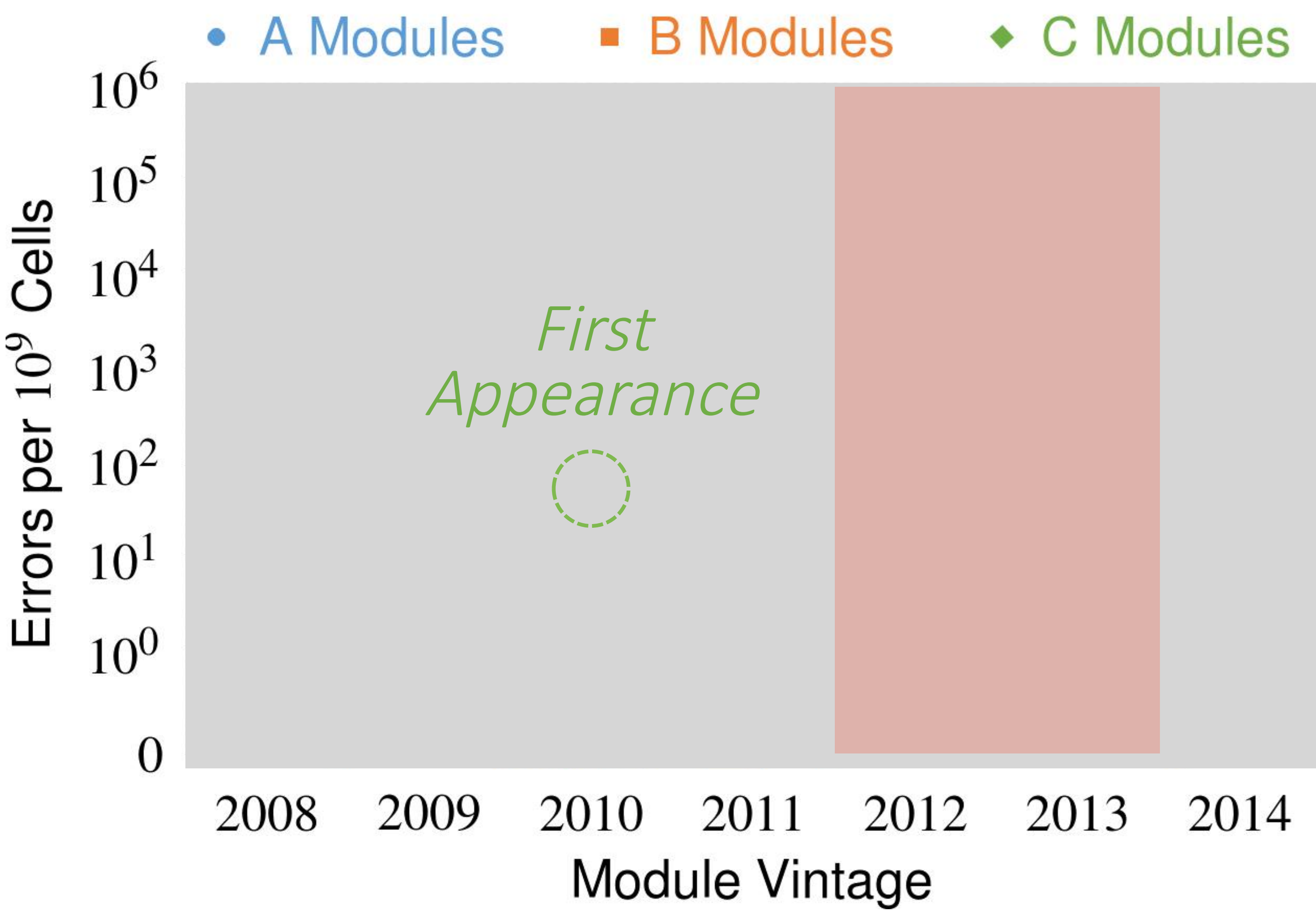
Observed Errors in Real Systems

CPU Architecture	Errors	Access-Rate
Intel Haswell (2013)	22.9K	12.3M/sec
Intel Ivy Bridge (2012)	20.7K	11.7M/sec
Intel Sandy Bridge (2011)	16.1K	11.6M/sec
AMD Piledriver (2012)	59	6.1M/sec

- *A real reliability & security issue*
- *In a more controlled environment, we can induce as many as ten million disturbance errors*

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

Errors *vs.* Vintage



All modules from 2012–2013 are vulnerable

Security Implications (I)

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

http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_isca14.pdf

Project Zero

News and updates from the Project Zero team at Google

http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

Security Implications (II)

- "Rowhammer" is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows.
- 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.
- 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.

Recap: The DRAM Scaling Problem

DRAM Process Scaling Challenges

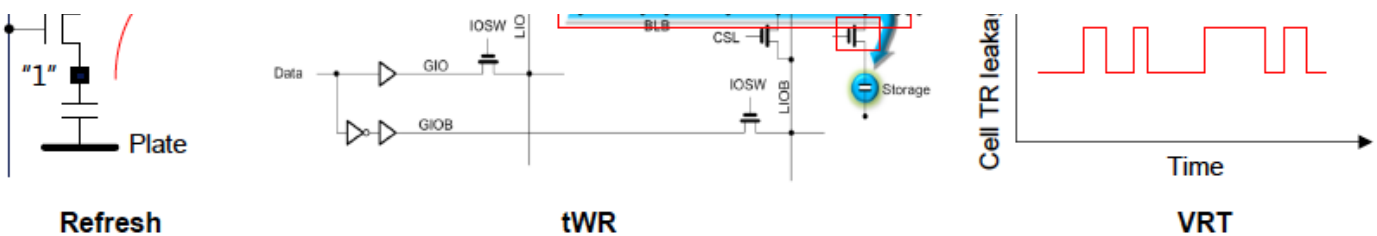
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

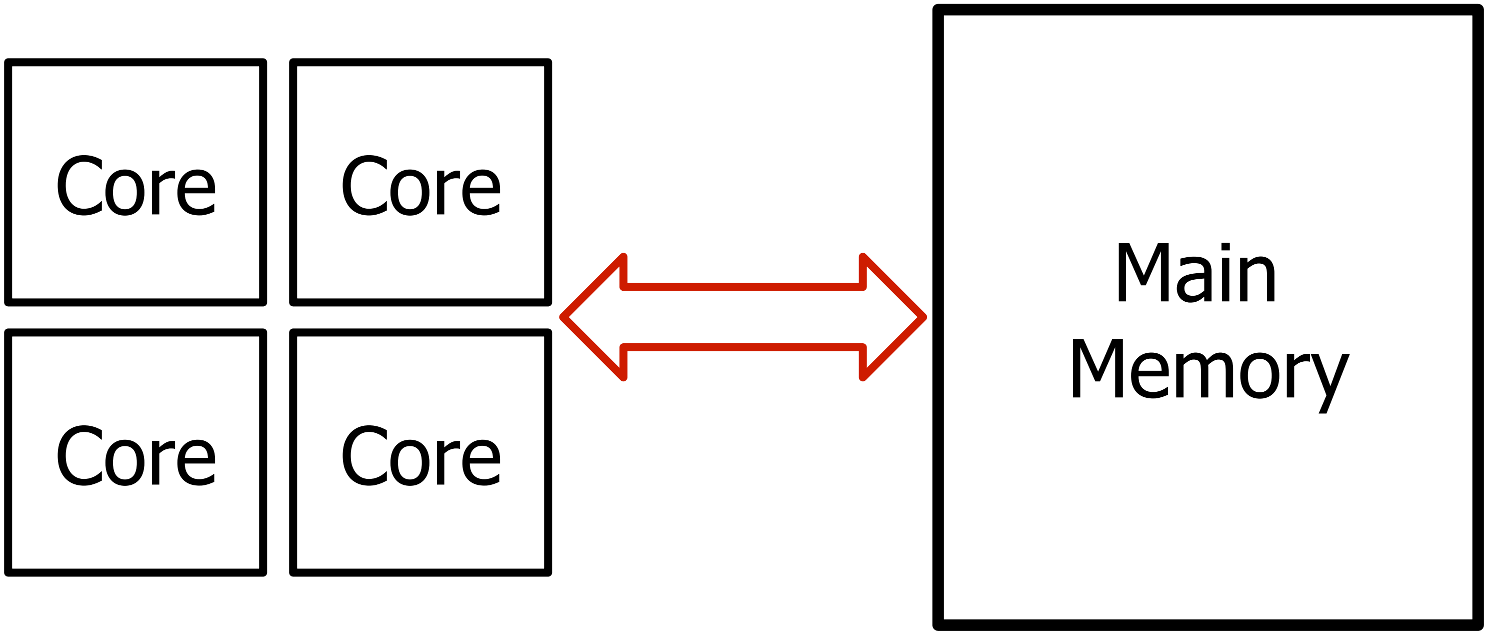
*Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel*



The Memory Forum

SAMSUNG intel

An Orthogonal Issue: Memory Interference



Cores' interfere with each other when accessing shared main memory

Uncontrolled interference leads to many problems (QoS, performance)

Major Trends Affecting Main Memory

- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern

- DRAM technology scaling is ending

How Can We Fix the Memory Problem & Design (Memory) Systems of the Future?

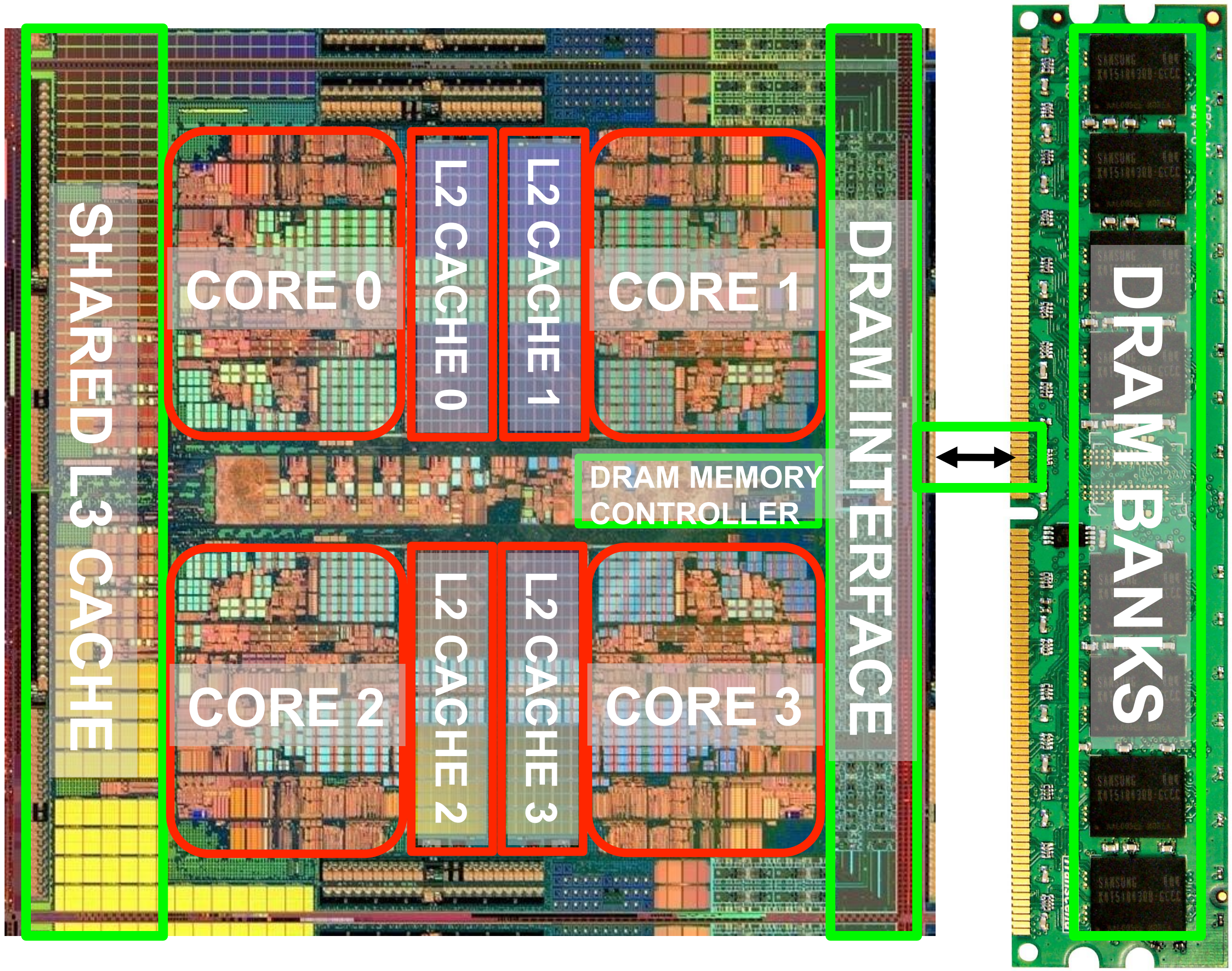
Look Backward to Look Forward

- We first need to understand the principles of:
  - Memory and DRAM
  - Memory controllers
  - Techniques for reducing and tolerating memory latency
  - Potential memory technologies that can compete with DRAM

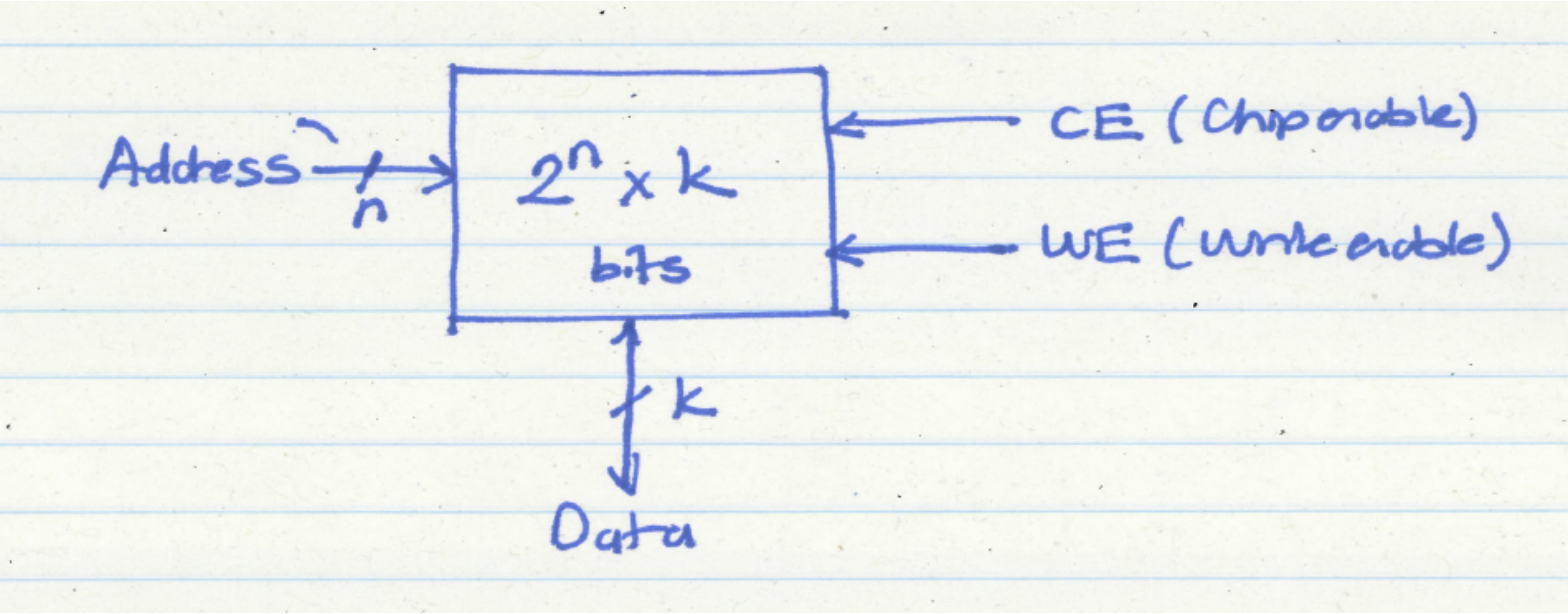
- This is what we will cover in the next few lectures

Main Memory

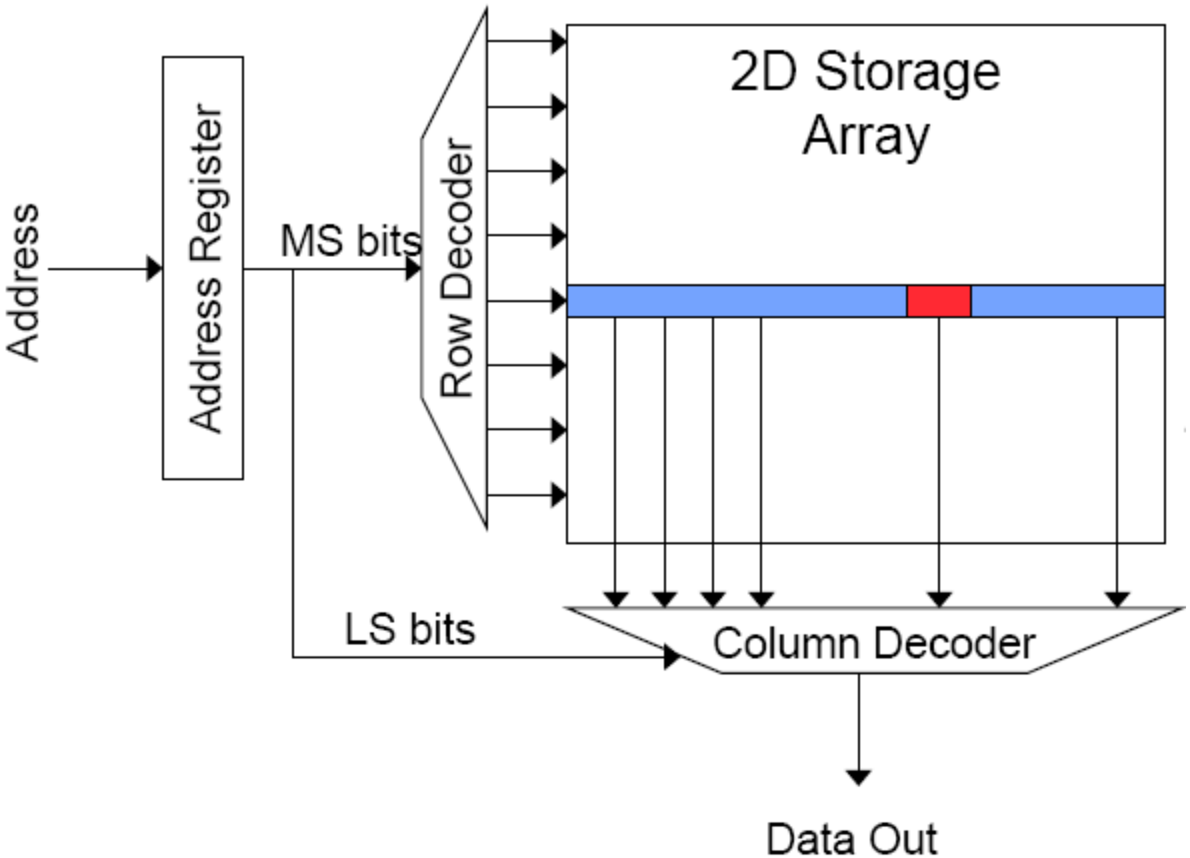
Main Memory in the System



The Memory Chip/System Abstraction



Review: Memory Bank Organization



- Read access sequence:

  1. Decode row address & drive word-lines

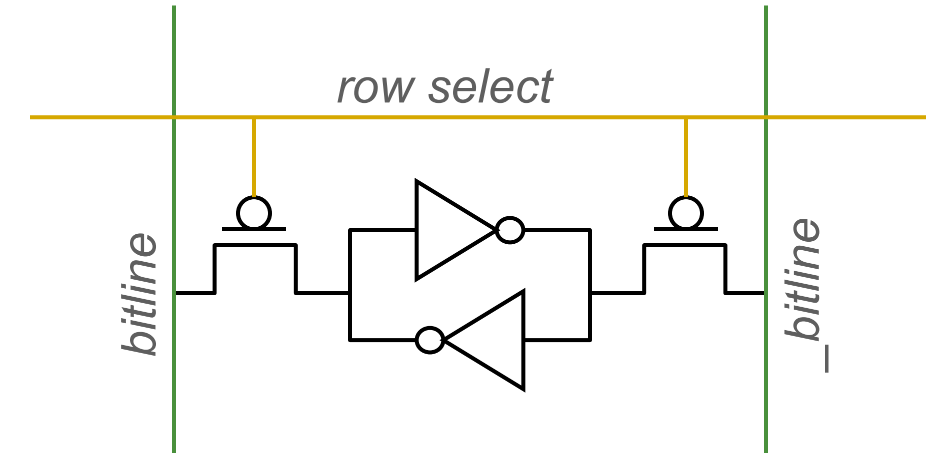
  2. Selected bits drive bit-lines
     - Entire row read

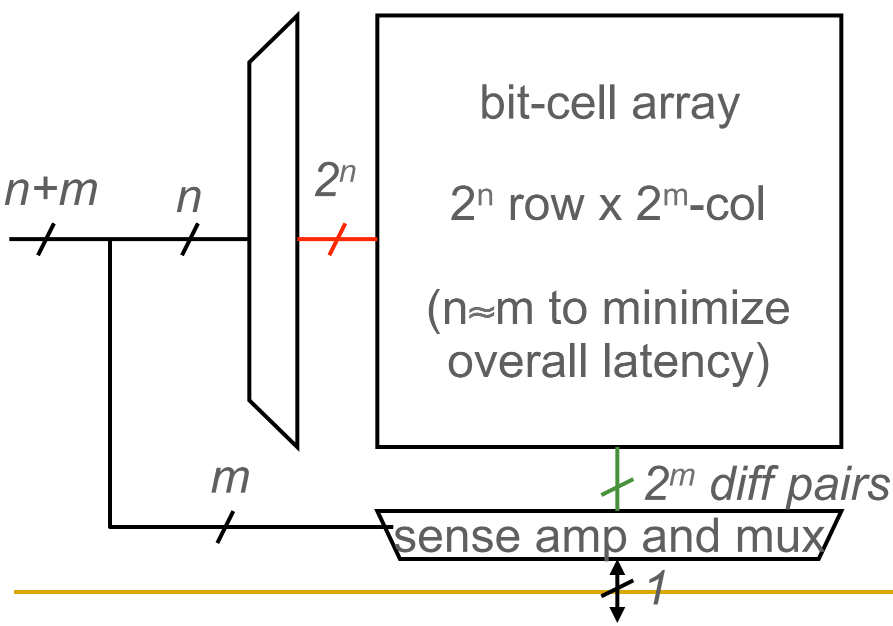
  3. Amplify row data

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

  5. Precharge bit-lines
     - For next access

Review: SRAM (Static Random Access Memory)





Read Sequence

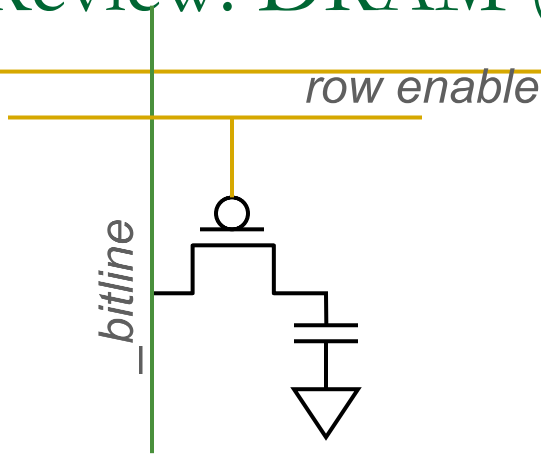
1. address decode
2. drive row select
3. selected bit-cells drive bitlines (entire row is read together)
4. diff. sensing and col. select (data is ready)
5. precharge all bitlines (for next read or write)

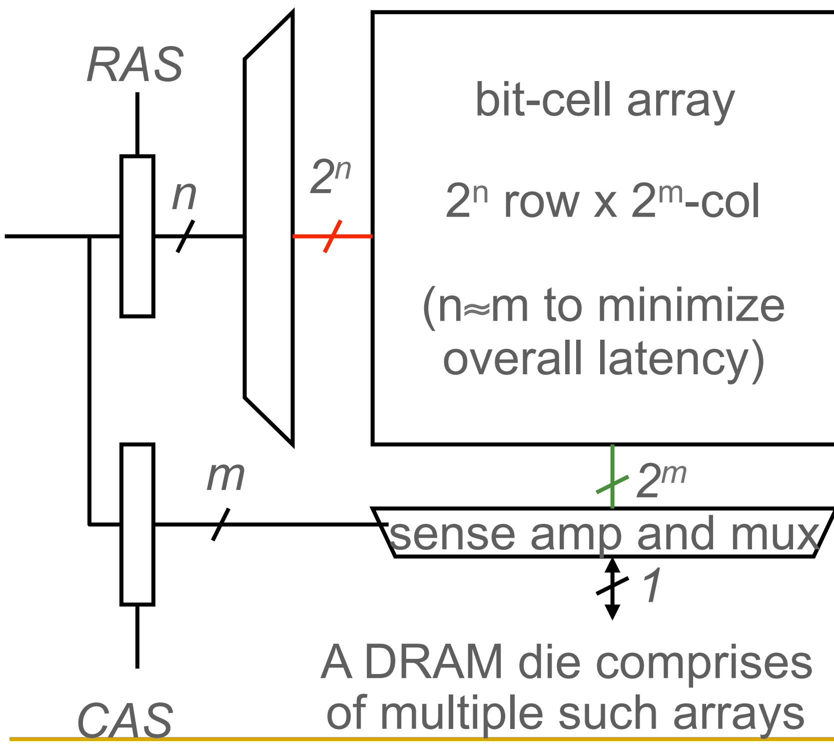
Access latency dominated by steps 2 and 3

Cycling time dominated by steps 2, 3 and 5

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

Review: DRAM (Dynamic Random Access Memory)





A DRAM die comprises of multiple such arrays

Bits stored as charges on node capacitance (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

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

Review: 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)

Some Fundamental Concepts (I)

- Physical address space
  - Maximum size of main memory: total number of uniquely identifiable locations
- Physical addressability
  - Minimum size of data in memory can be addressed
  - Byte-addressable, word-addressable, 64-bit-addressable
  - Microarchitectural addressability depends on the abstraction level of the implementation
- Alignment
  - Does the hardware support unaligned access transparently to software?
- Interleaving

Some Fundamental Concepts (II)

- Interleaving (banking)
  - Problem: a single monolithic 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 the 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?)

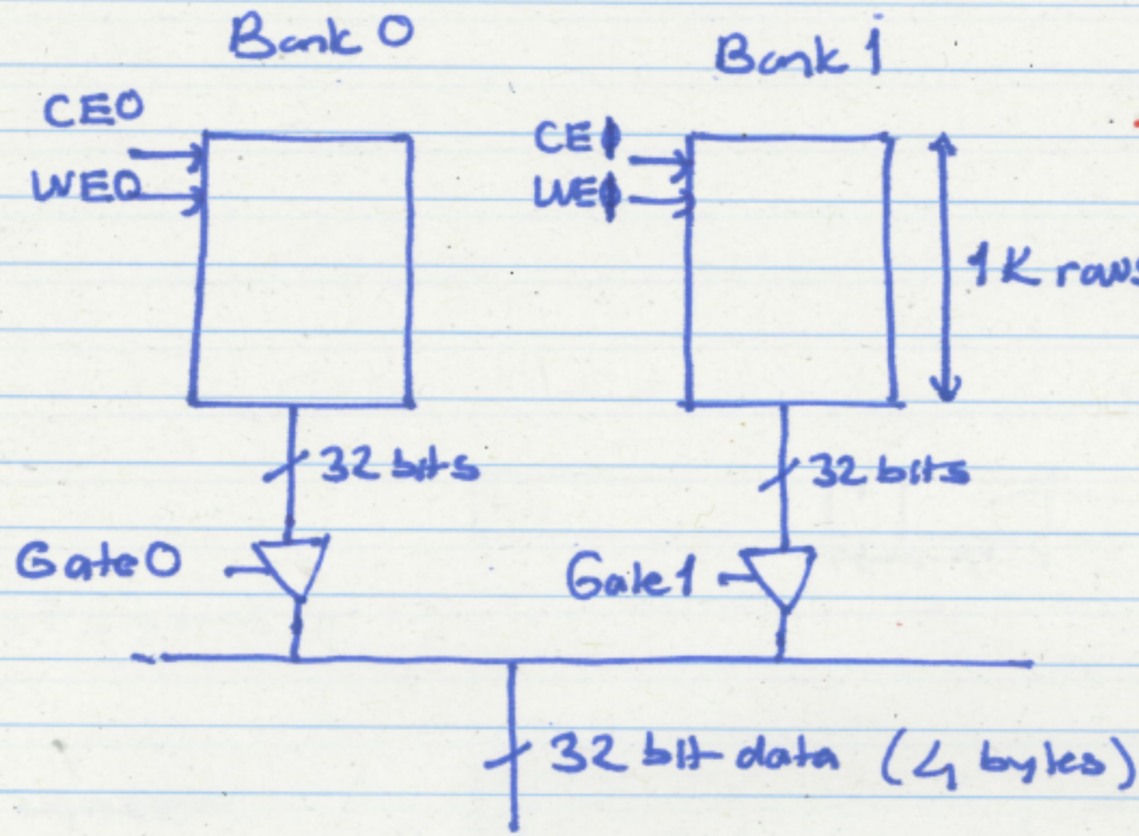
Interleaving

Interleaving (Example)

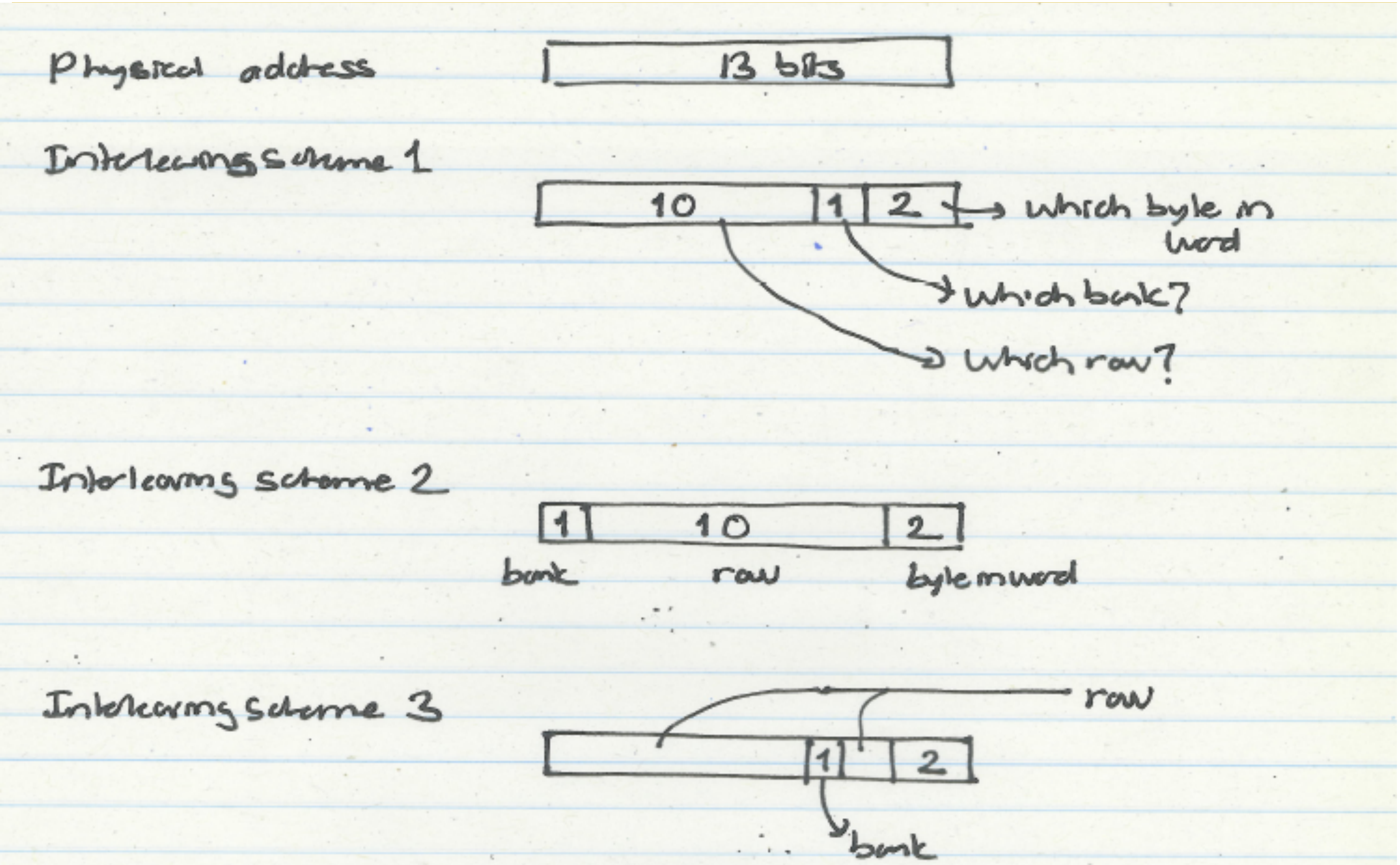
Assume each bank supplies a word.

Which banks do consecutive words in memory are mapped to?

i.e. how do we interleave the words across the banks?



Interleaving Options

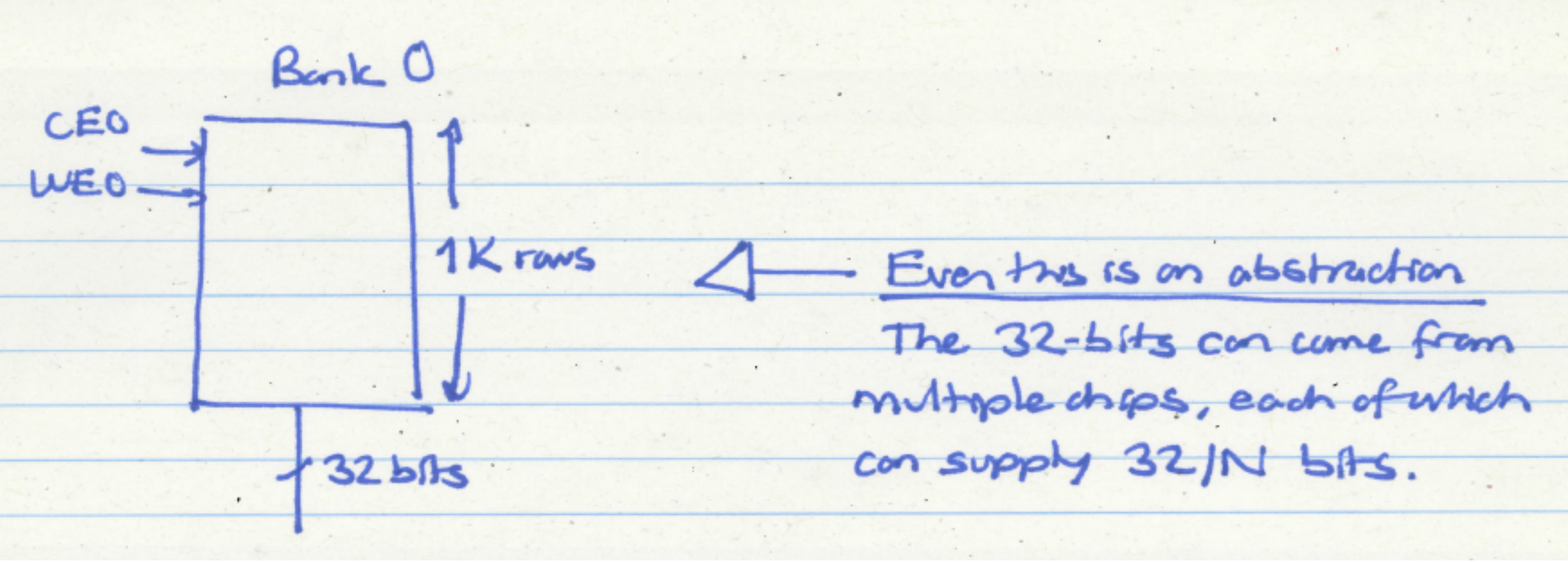


Where (which bank) do consecutive words in memory are mapped to?

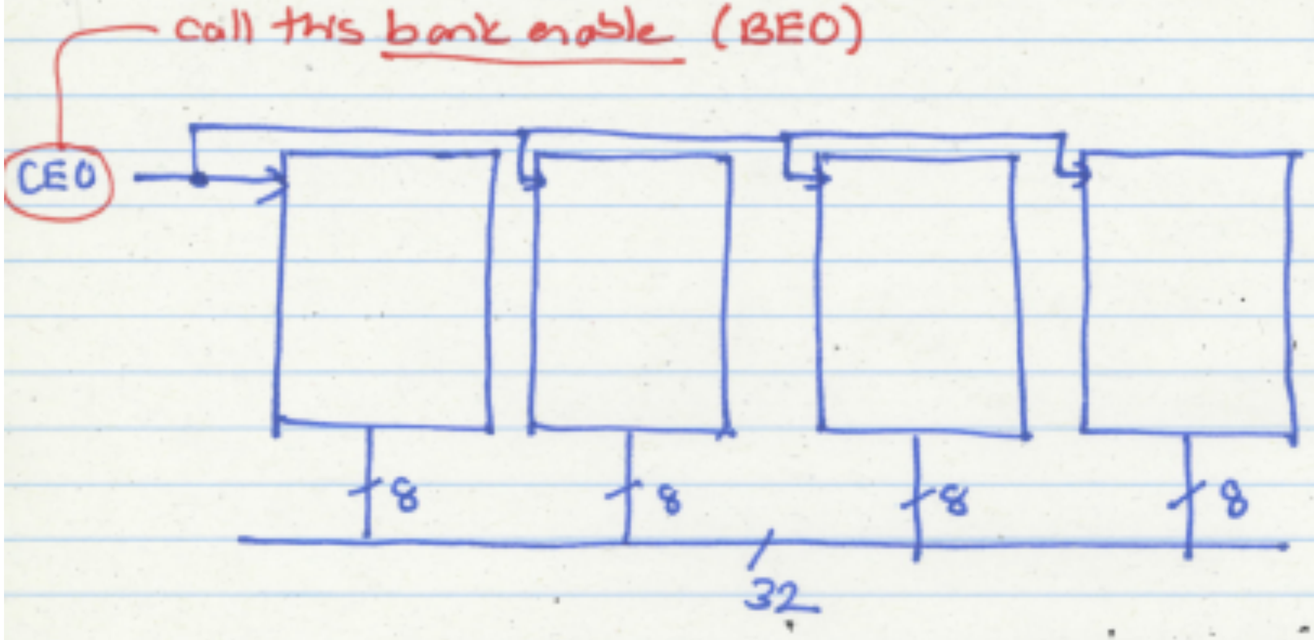
Some Questions/Concepts

- Remember CRAY-1 with 16 banks
  - 11 cycle bank latency
  - Consecutive words in memory in consecutive banks (word interleaving)
  - 1 access can be started (and finished) per cycle
- Can banks be operated *fully* in parallel?
  - Multiple accesses started per cycle?
- What is the cost of this?
  - We have seen it earlier
- Modern superscalar processors have L1 data caches with multiple, fully-independent banks; DRAM banks share buses

The Bank Abstraction



Rank



This is called a "rank." (only bank 0 of the rank shown here)

Rank: A set of chips that respond to the same command & same address at the same time with different pieces of the requested data

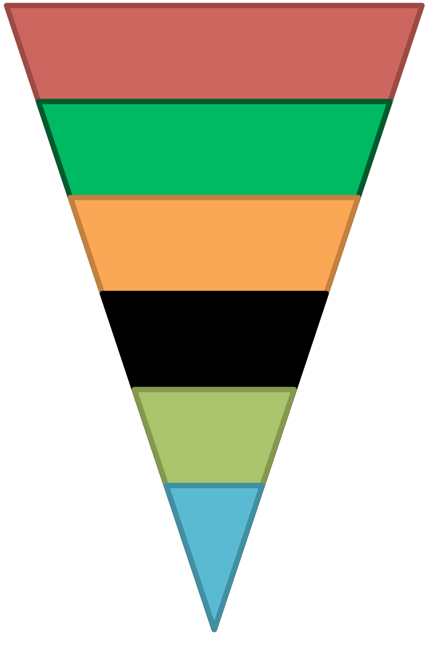
Why? Producing an 8-bit/pin chip cheaper than producing a 32-bit/pin chip

Idea: Produce an 8-bit/pin chip, but control/operate them as a rank so that we can get 32 bits in a single read.

The DRAM Subsystem

DRAM Subsystem Organization

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

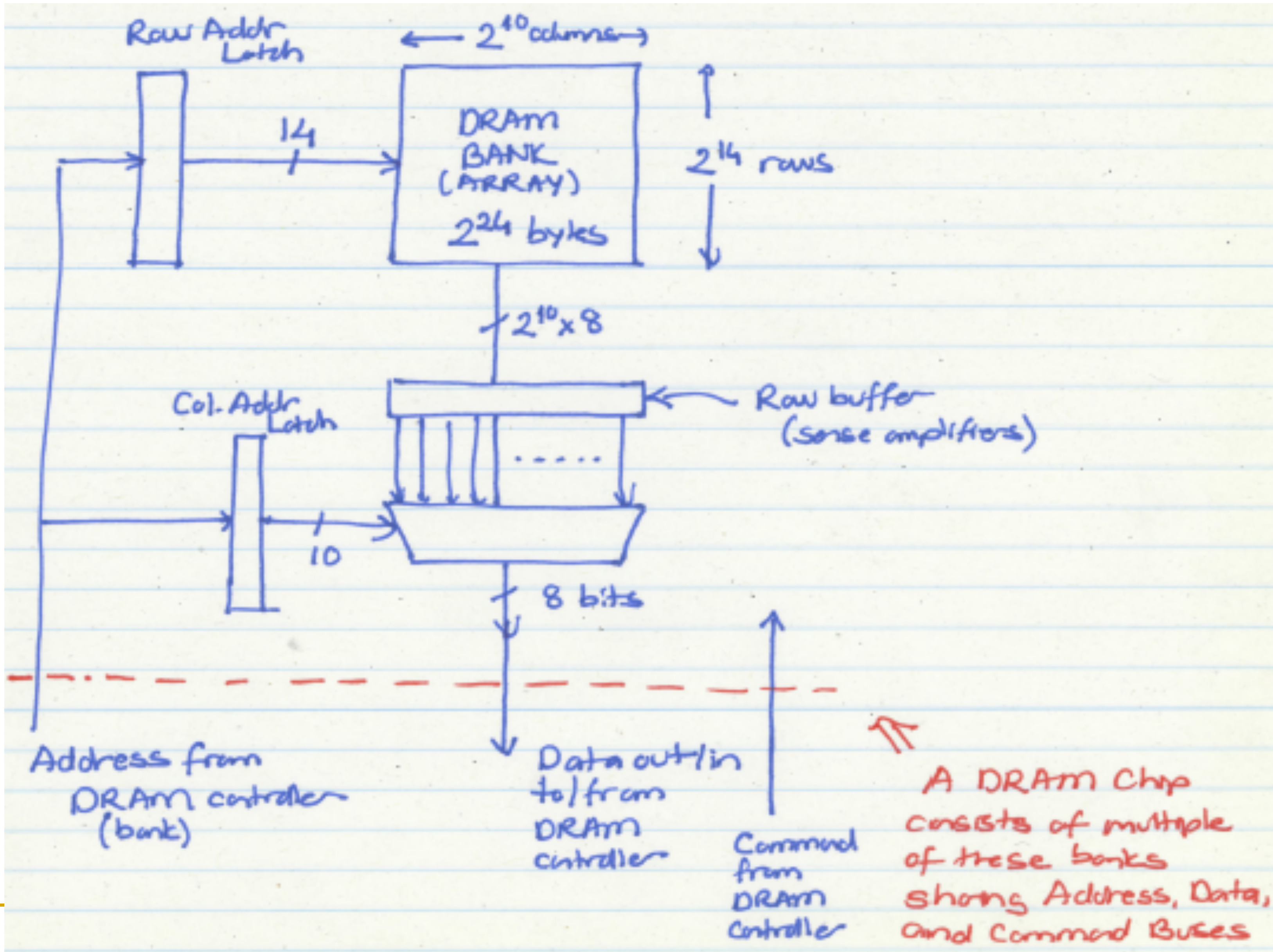


Page Mode DRAM

- A DRAM bank is a 2D array of cells: rows x columns
- A “DRAM row” is also called a “DRAM page”
- “Sense amplifiers” also called “row buffer”

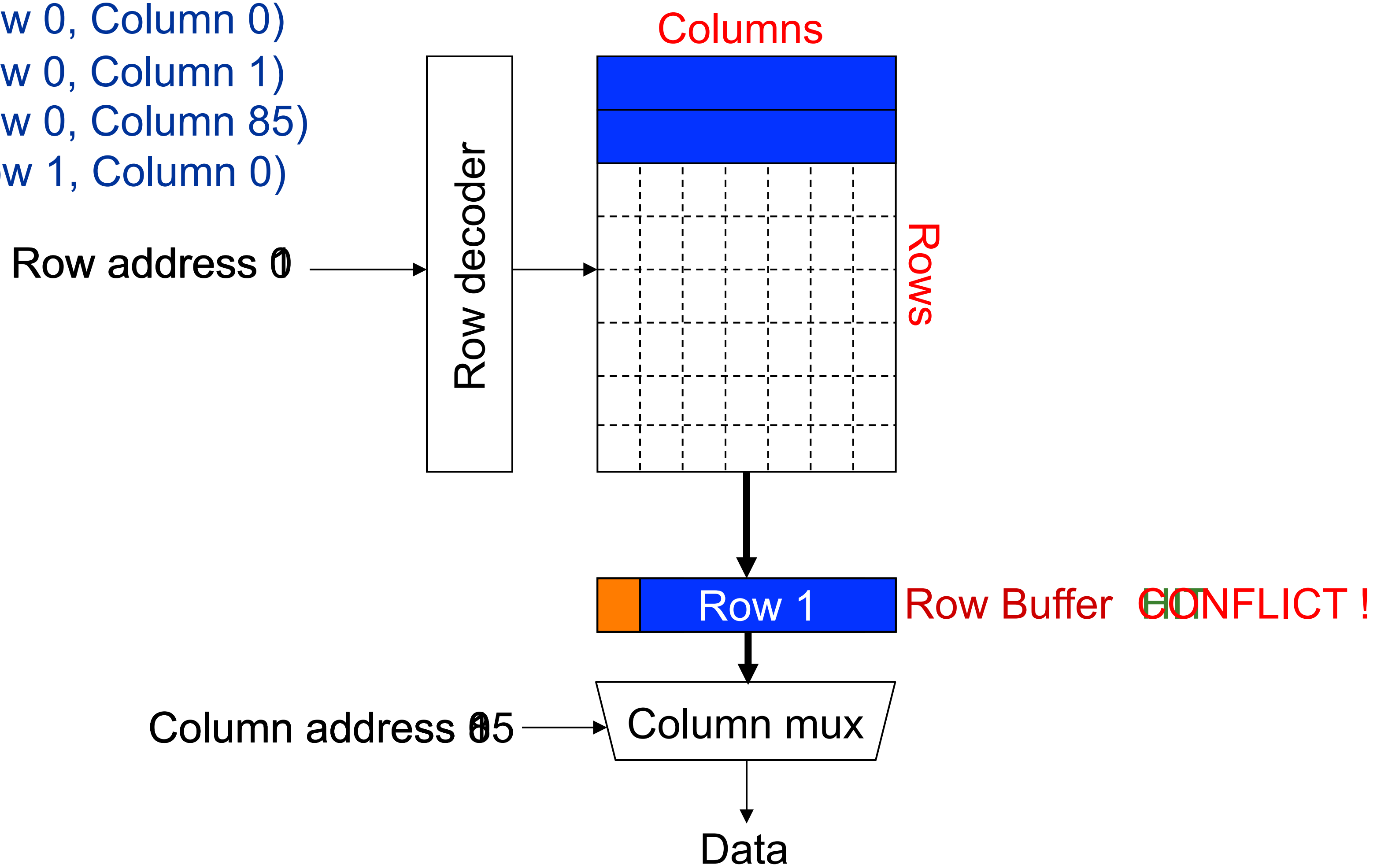
- Each address is a <row,column> pair
- Access to a “closed row”
  - Activate command opens row (placed into row buffer)
  - Read/write command reads/writes column in the row buffer
  - Precharge command closes the row and prepares the bank for next access
- Access to an “open row”
  - No need for an activate command

The DRAM Bank Structure



DRAM Bank Operation

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

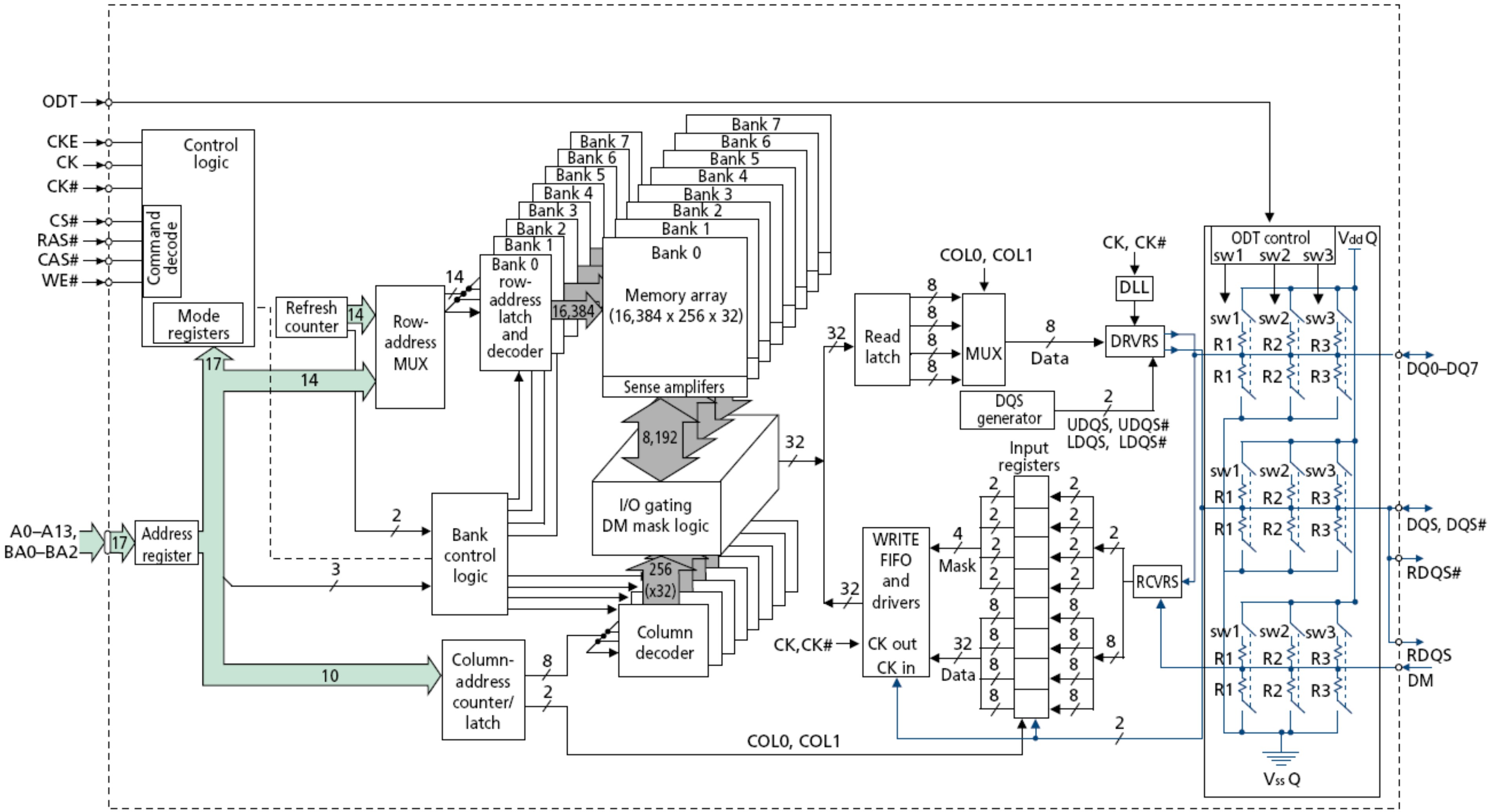


The DRAM Chip

- Consists of multiple banks (8 is a common number today)
- Banks share command/address/data buses
- The chip itself has a narrow interface (4-16 bits per read)

- Changing the number of banks, size of the interface (pins), whether or not command/address/data buses are shared has significant impact on DRAM system cost

128M x 8-bit DRAM Chip



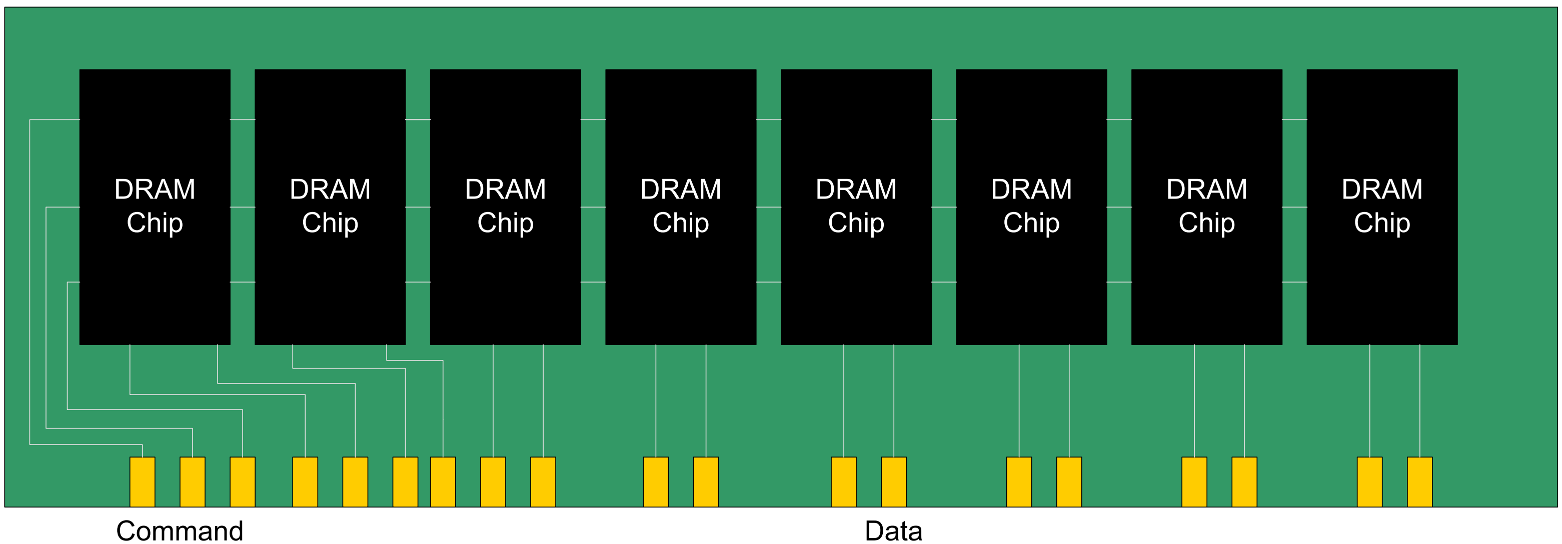
DRAM Rank and Module

- Rank: Multiple chips operated together to form a wide interface
- All chips comprising a rank are controlled at the same time
  - Respond to a single command
  - Share address and command buses, but provide different data

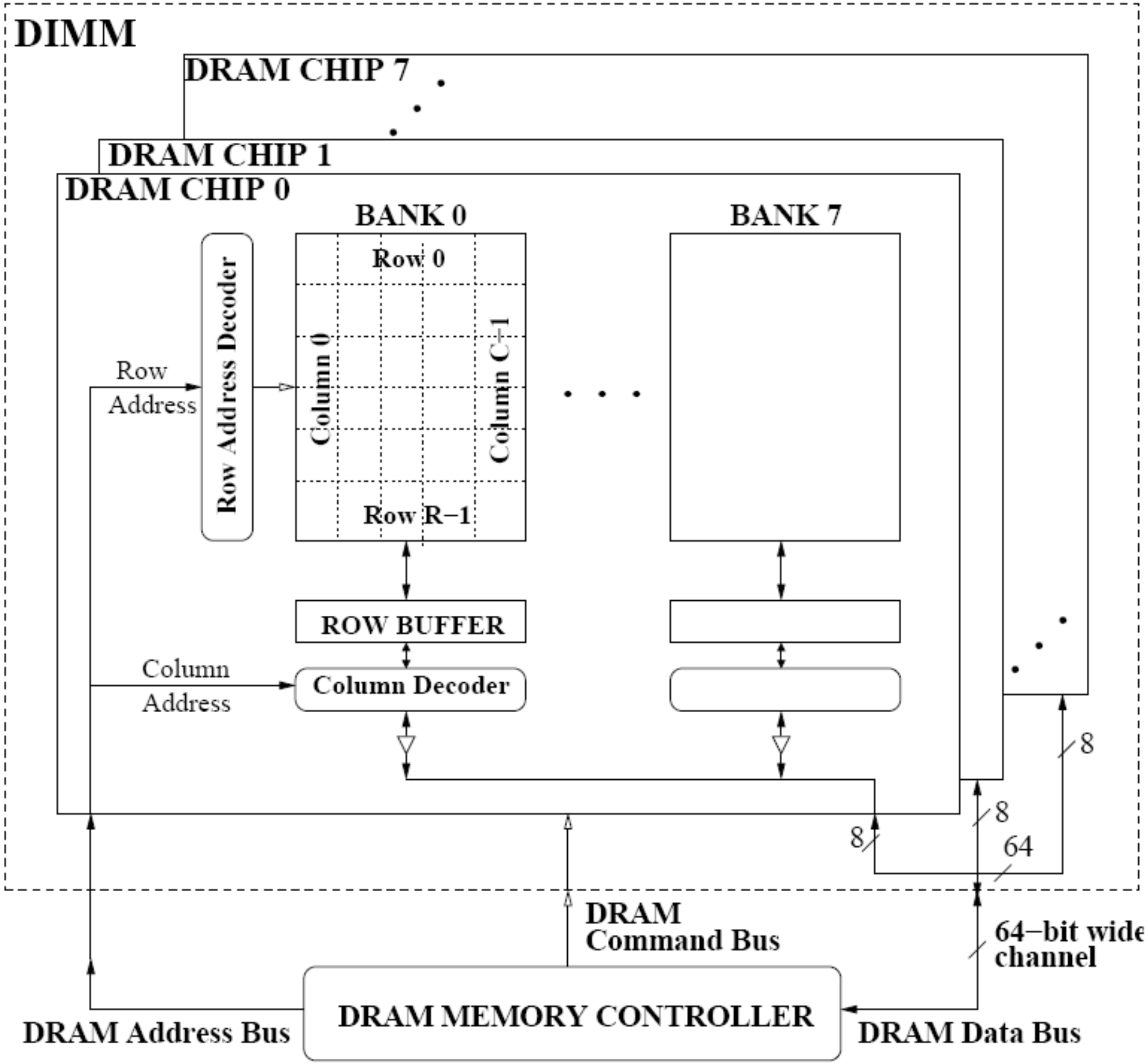
- A DRAM module consists of one or more ranks
  - E.g., DIMM (dual inline memory module)
  - This is what you plug into your motherboard

- If we have chips with 8-bit interface, to read 8 bytes in a single access, use 8 chips in a DIMM

A 64-bit Wide DIMM (One Rank)

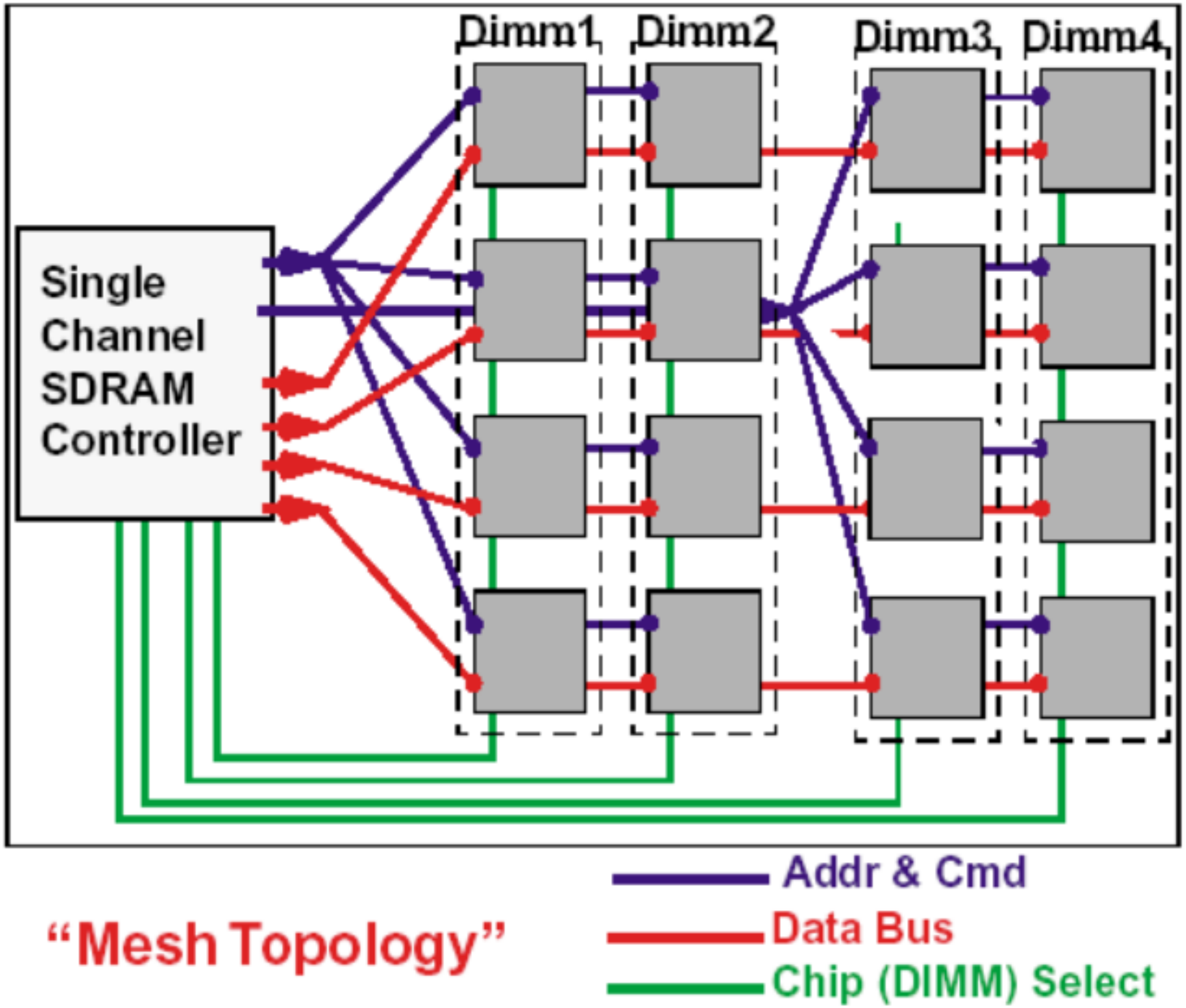


A 64-bit Wide DIMM (One Rank)



- Advantages:
  - Acts like a high-capacity DRAM chip with a wide interface
  - Flexibility: memory controller does not need to deal with individual chips
- Disadvantages:
  - Granularity: Accesses cannot be smaller than the interface width

Multiple DIMMs

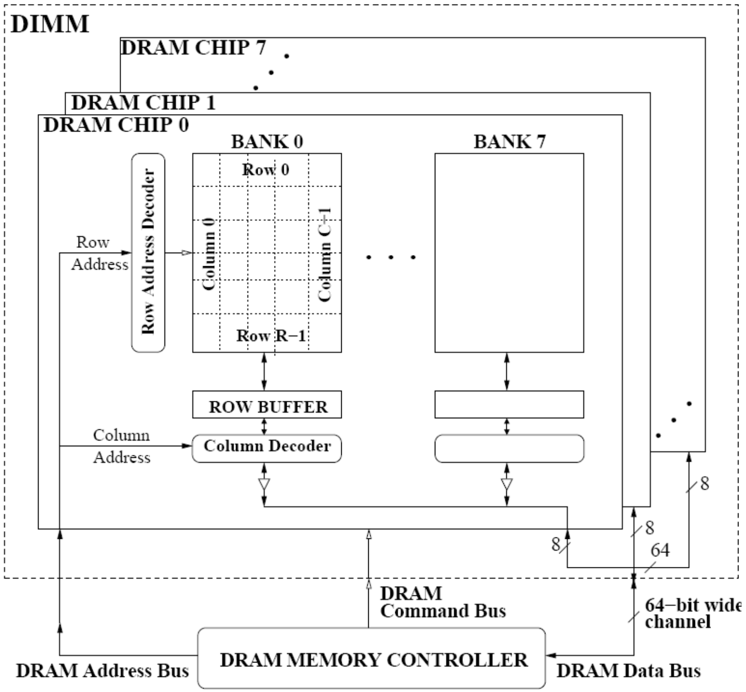


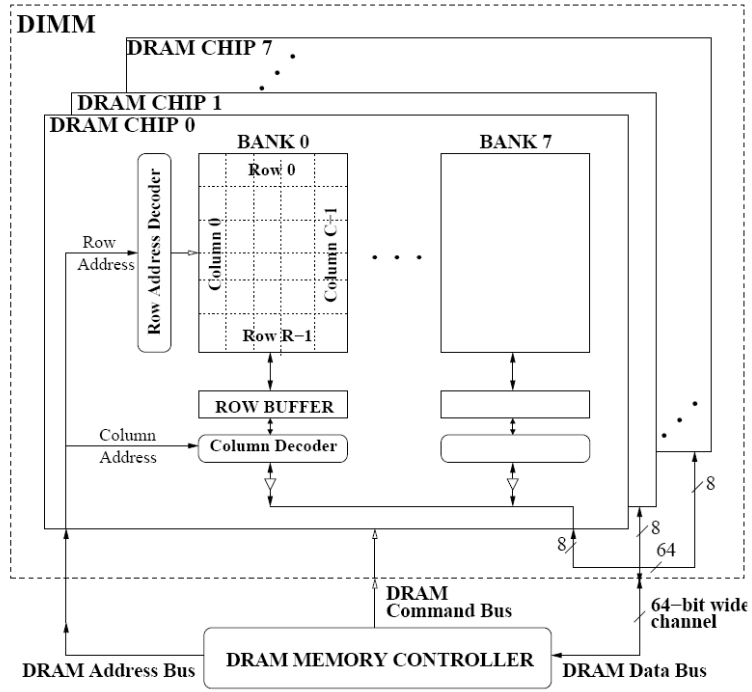
"Mesh Topology"

- Advantages:
  - Enables even higher capacity
- Disadvantages:
  - Interconnect complexity and energy consumption can be high

    → Scalability is limited by this

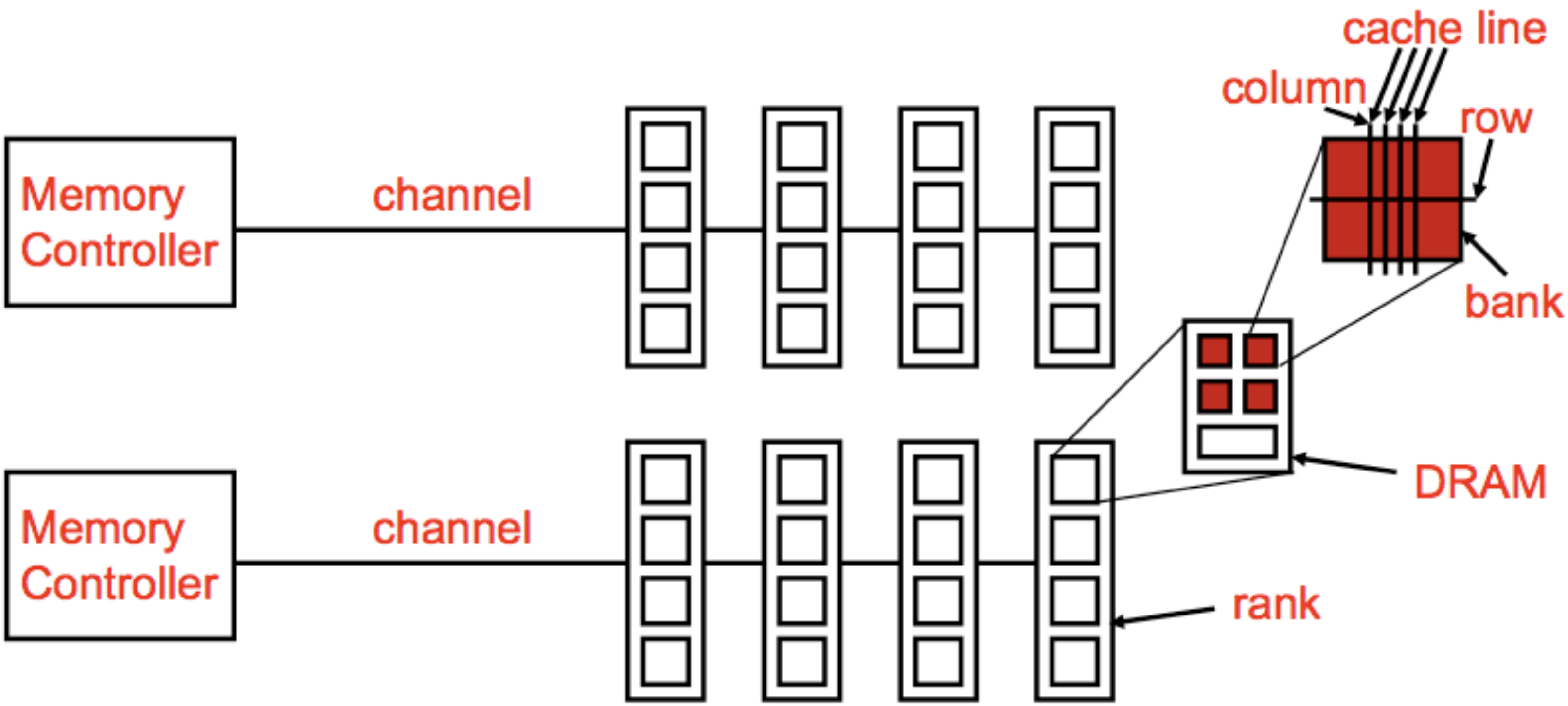
DRAM Channels



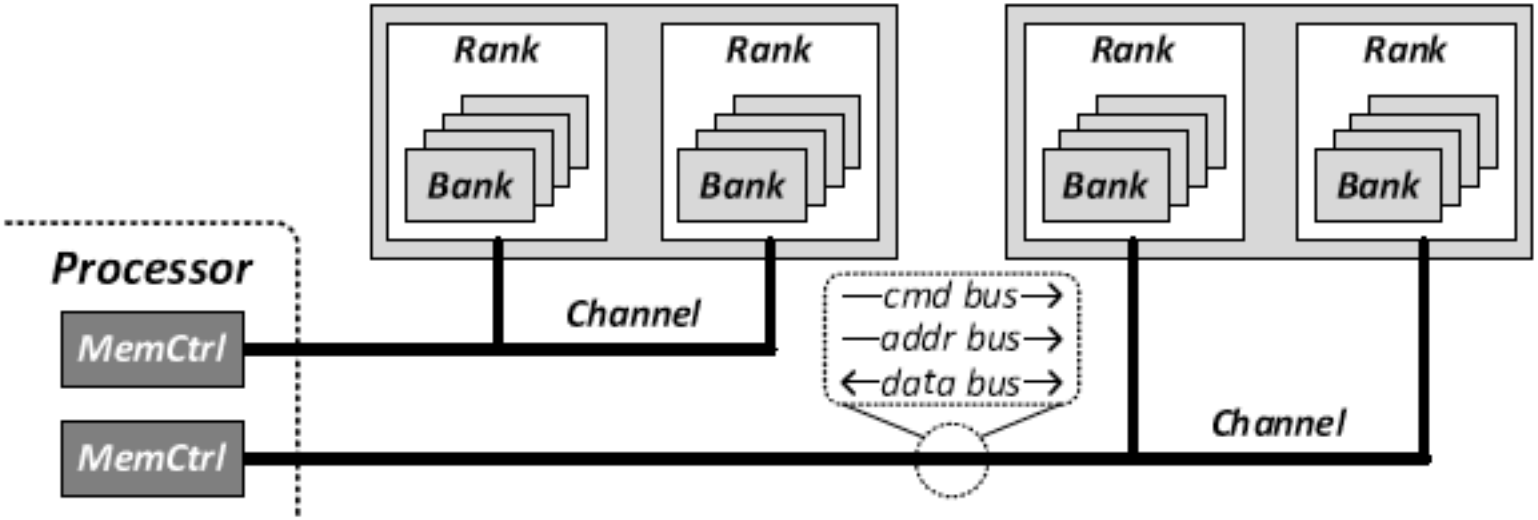


- 2 Independent Channels: 2 Memory Controllers (Above)
- 2 Dependent/Lockstep Channels: 1 Memory Controller with wide interface (Not Shown above)

Generalized Memory Structure



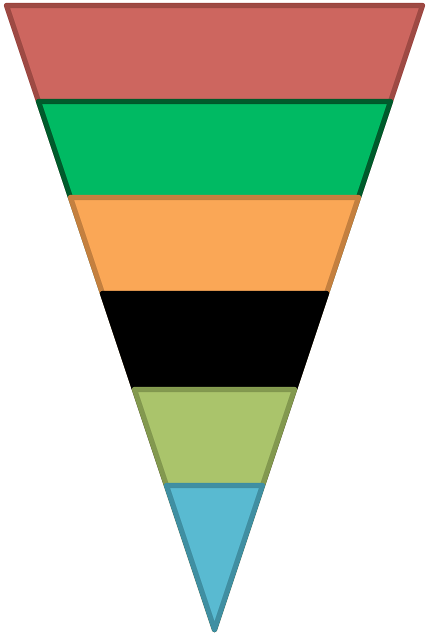
Generalized Memory Structure



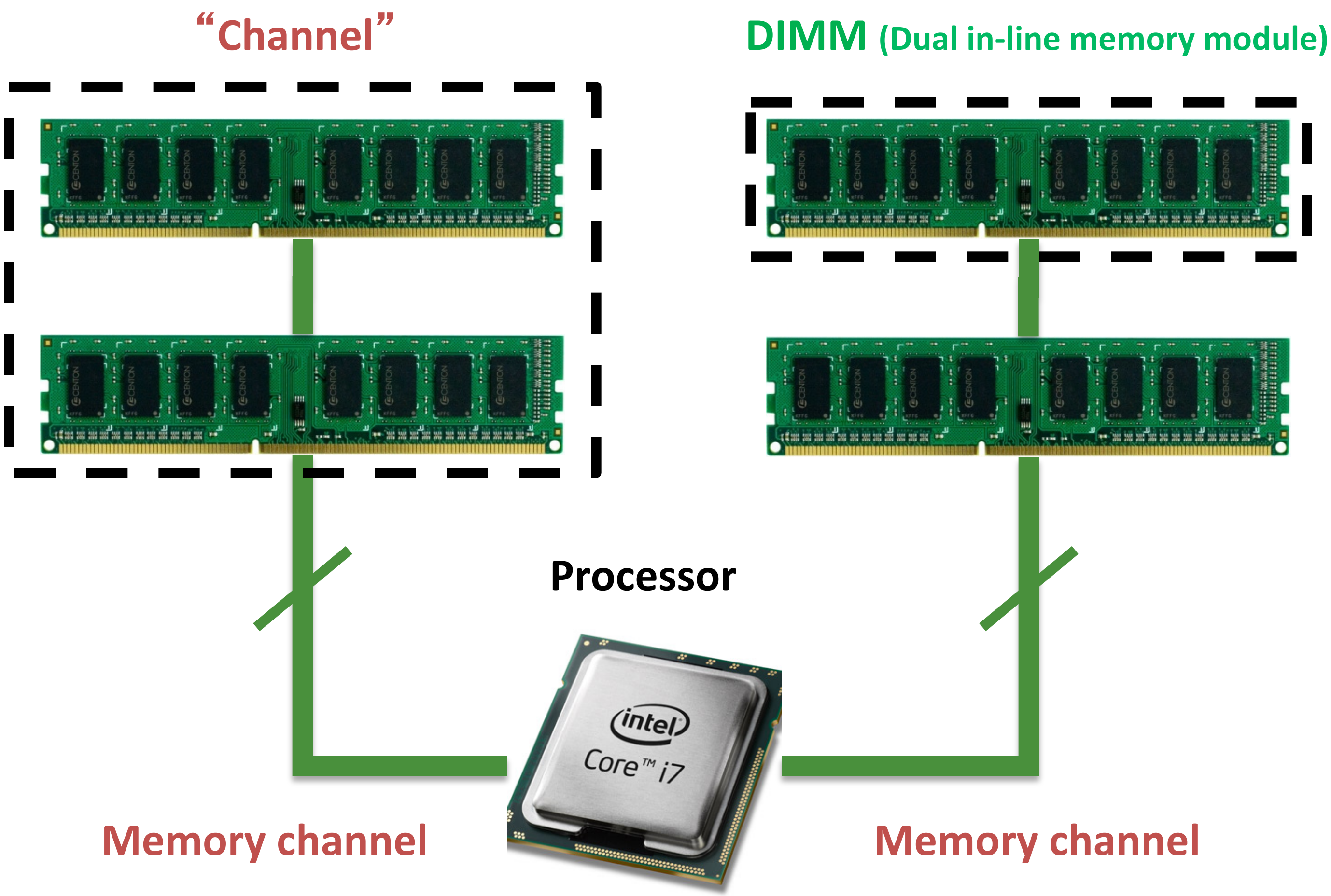
The DRAM Subsystem
The Top Down View

DRAM Subsystem Organization

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



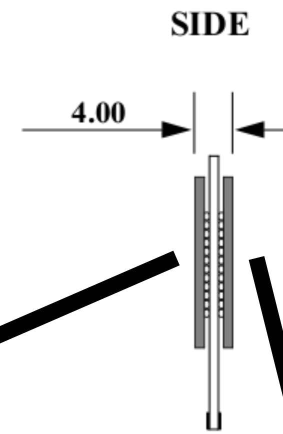
The DRAM subsystem



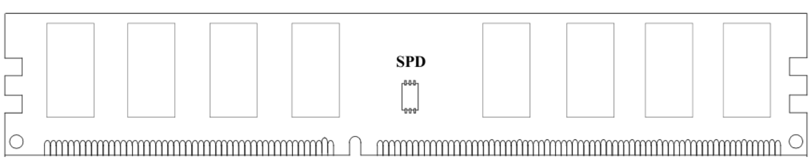
Breaking down a DIMM

DIMM (Dual in-line memory module)

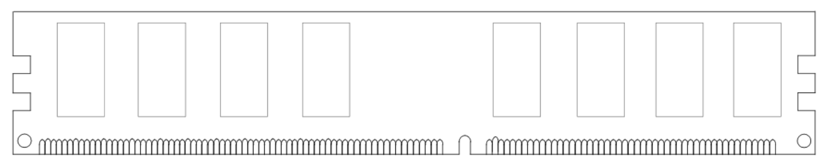
Side view



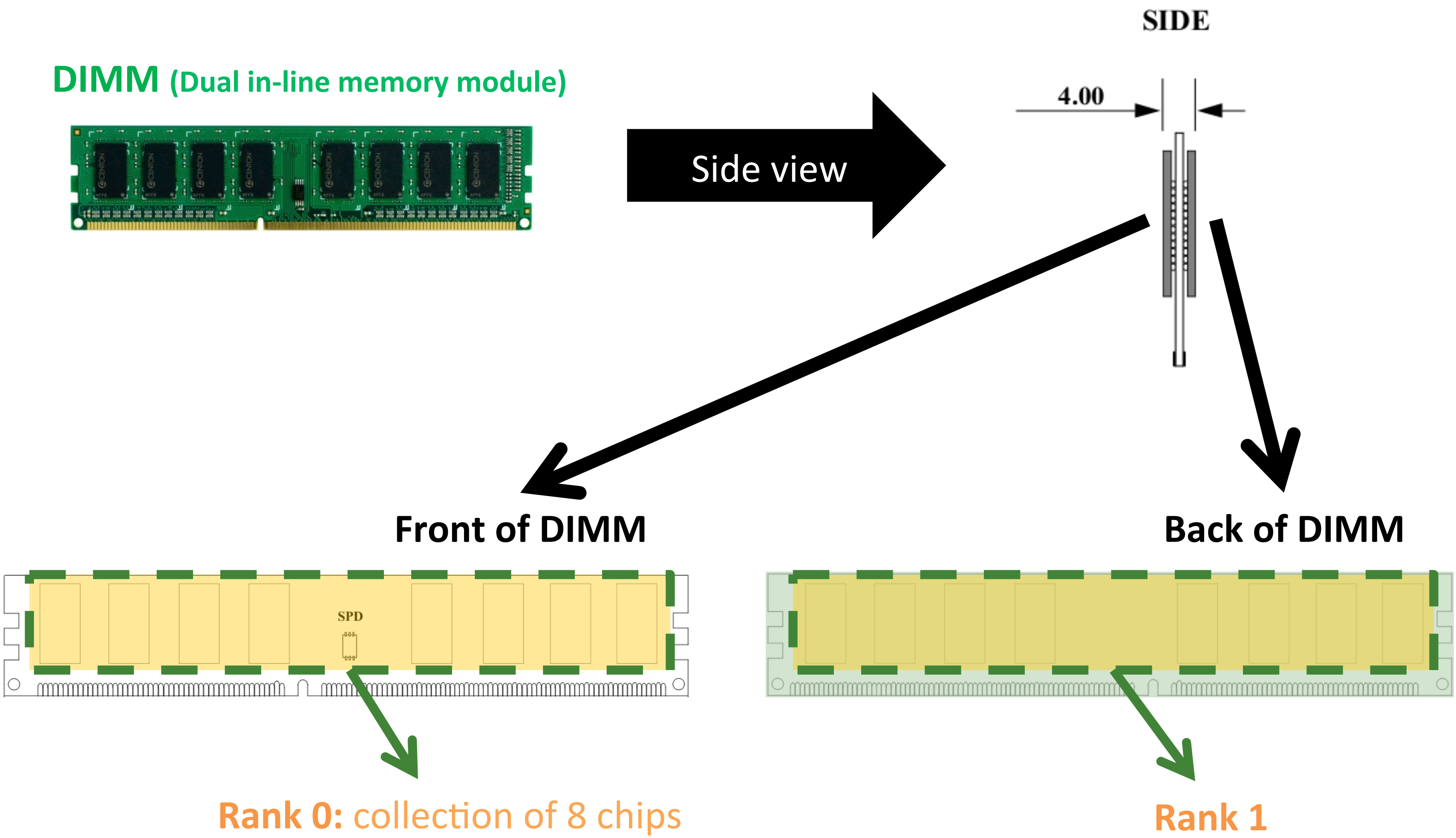
Front of DIMM



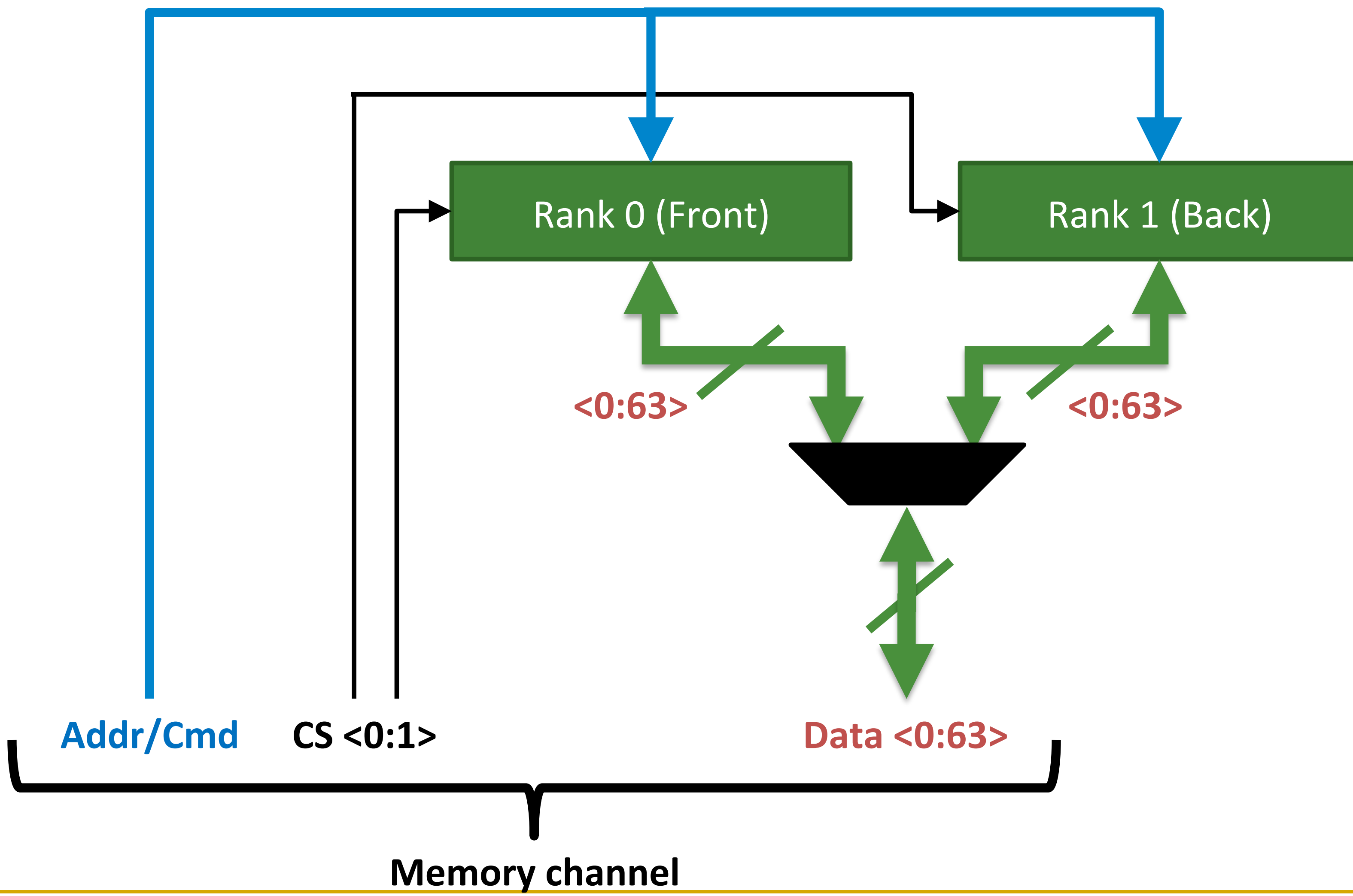
Back of DIMM



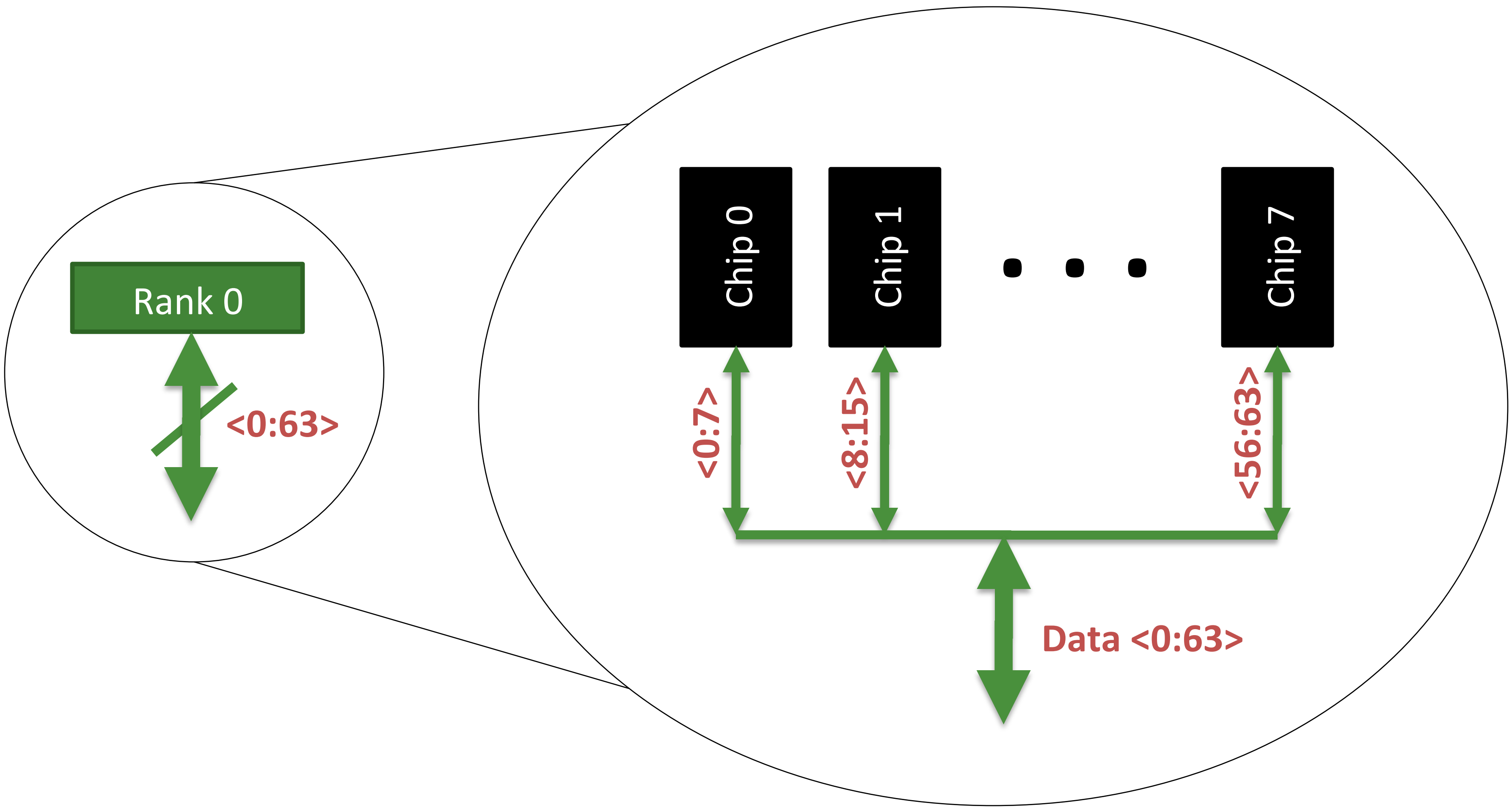
Breaking down a DIMM



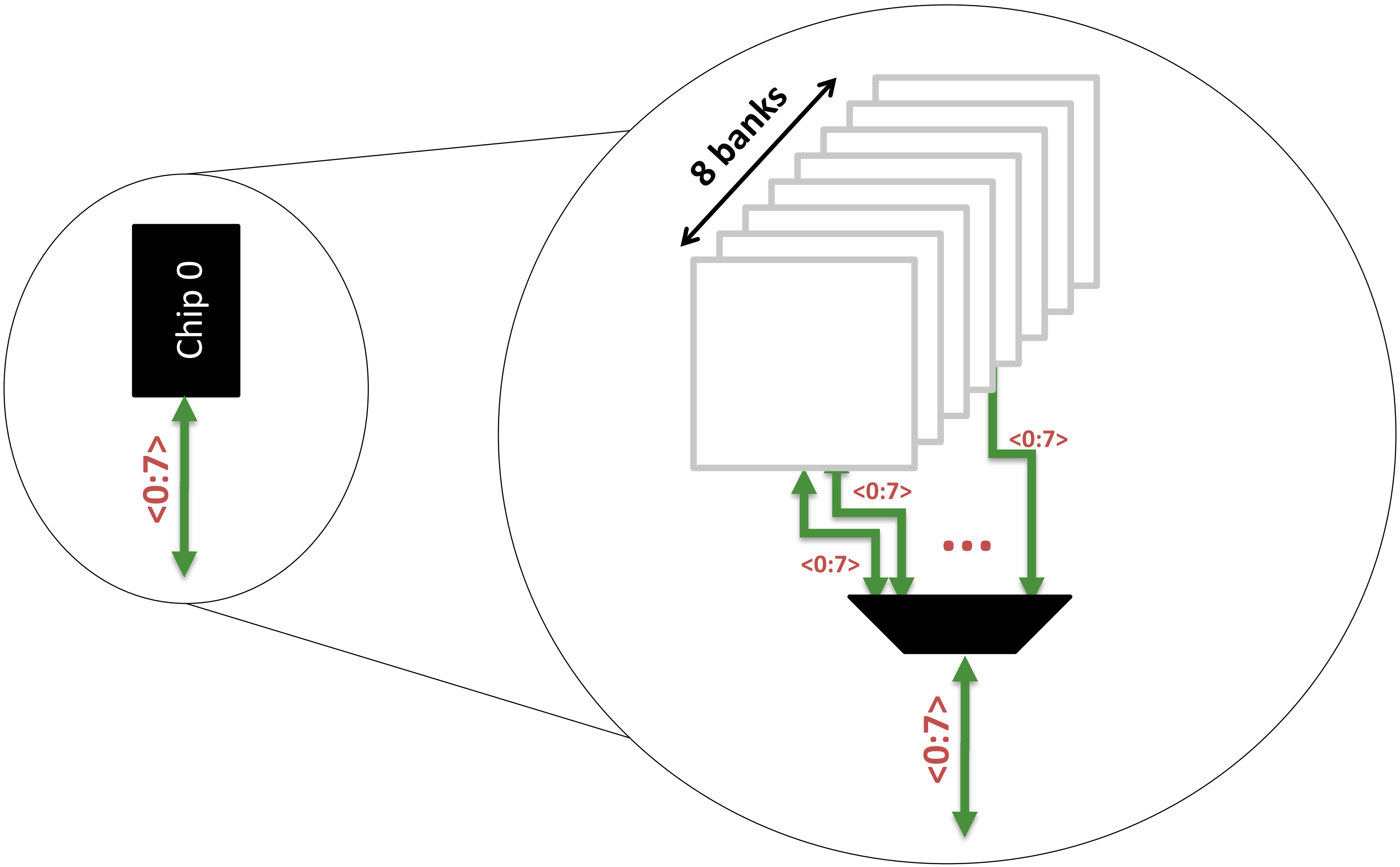
Rank


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

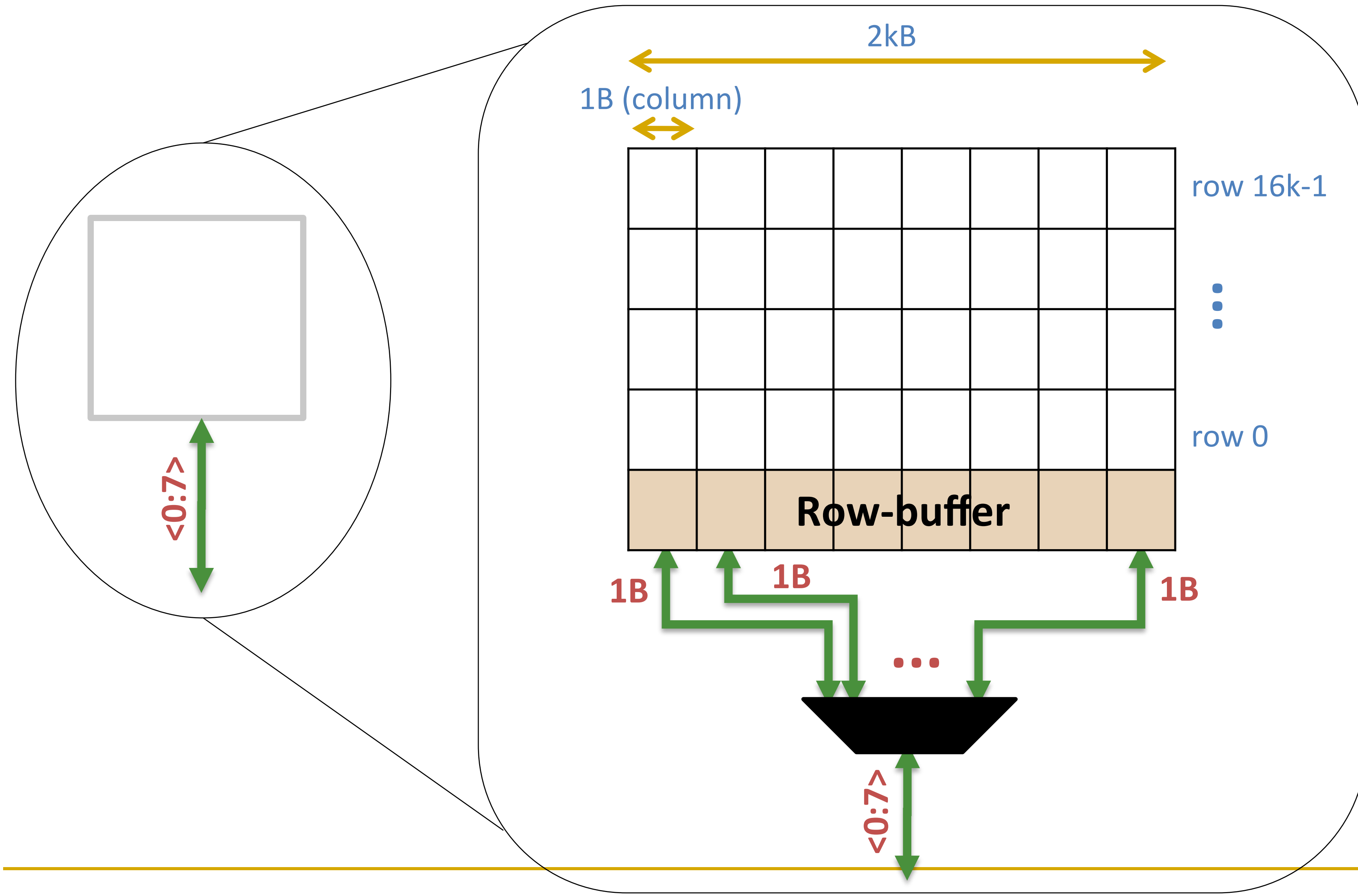
Breaking down a Rank



Breaking down a Chip

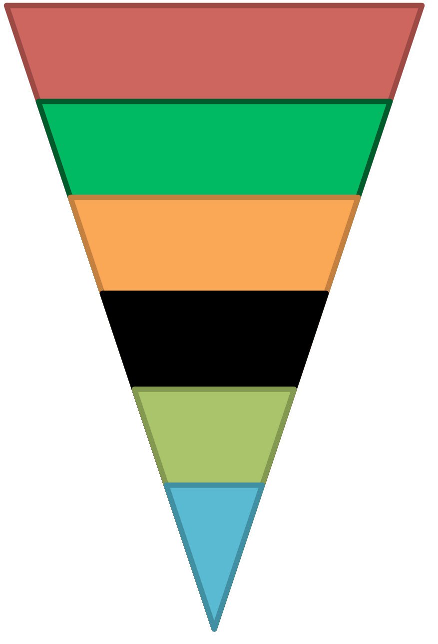


Breaking down a Bank

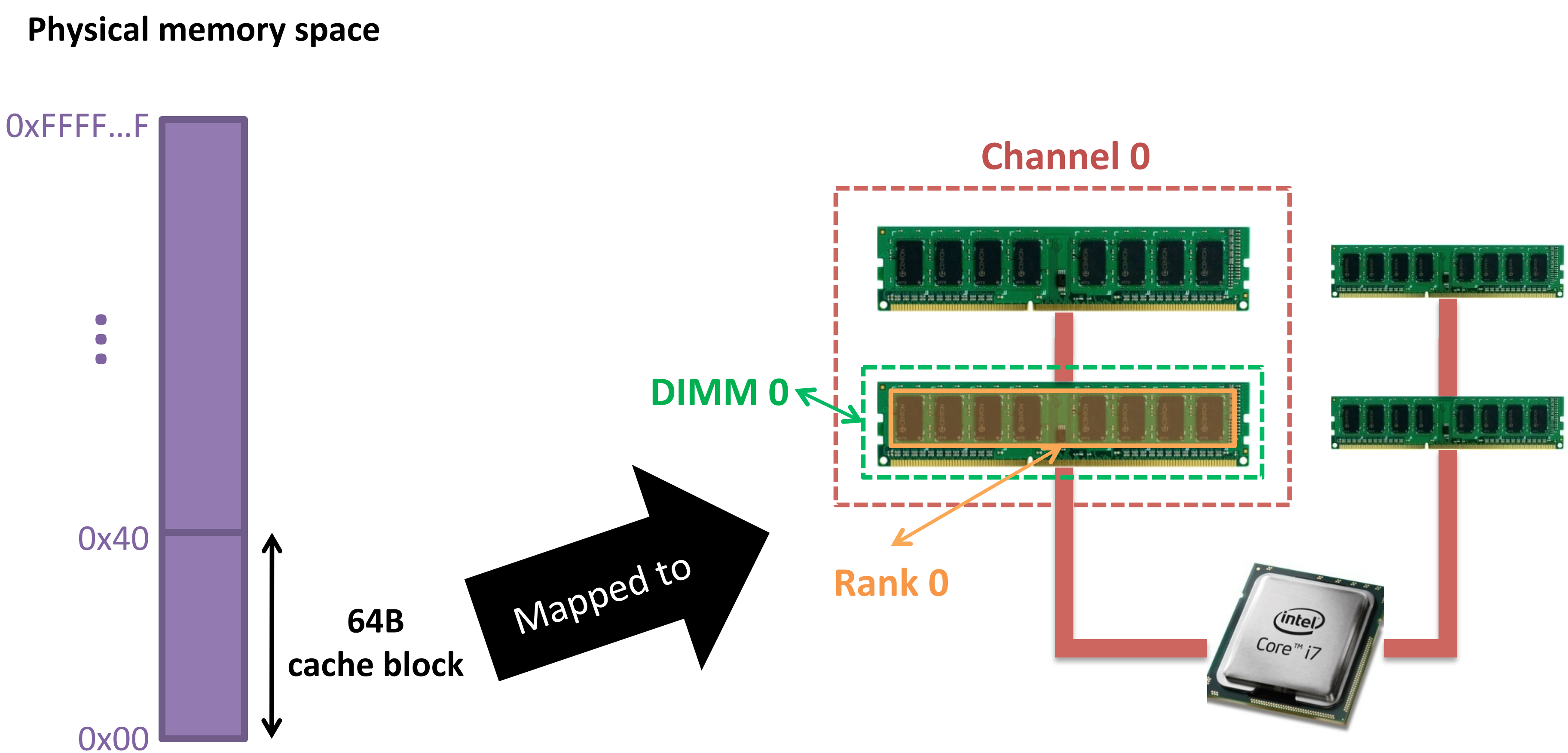


DRAM Subsystem Organization

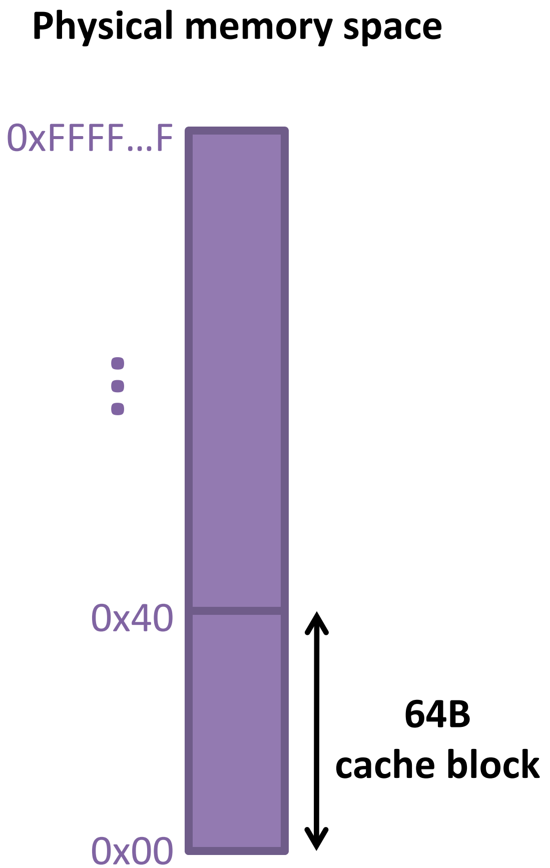
- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column
- Cell

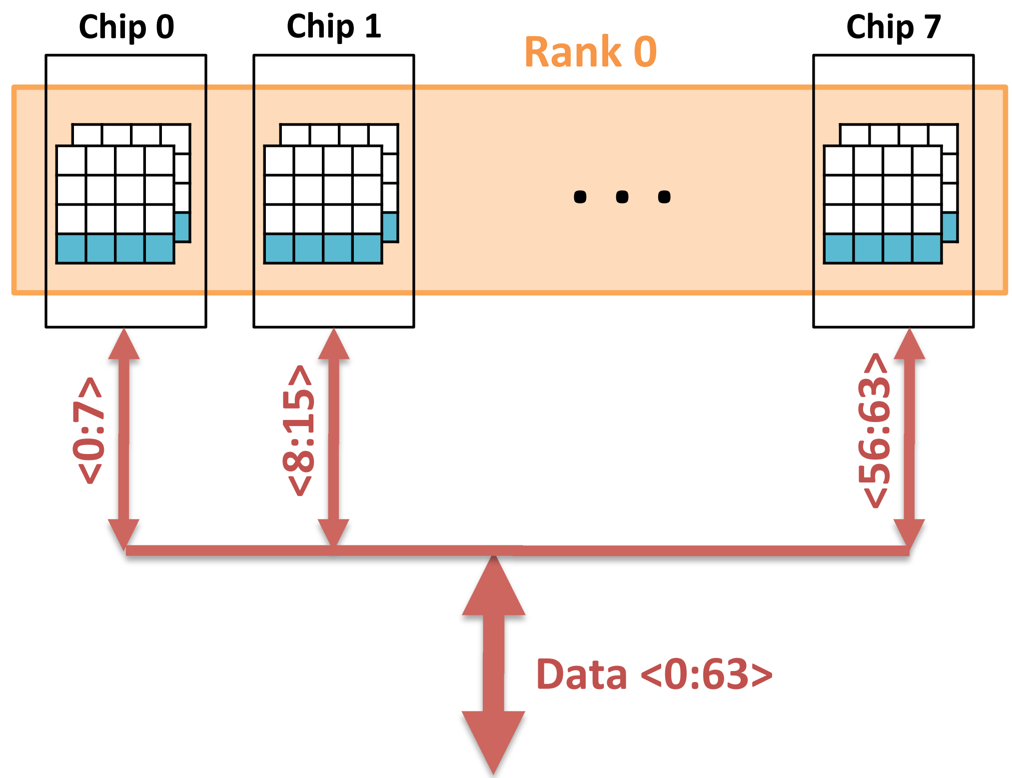


Example: Transferring a cache block

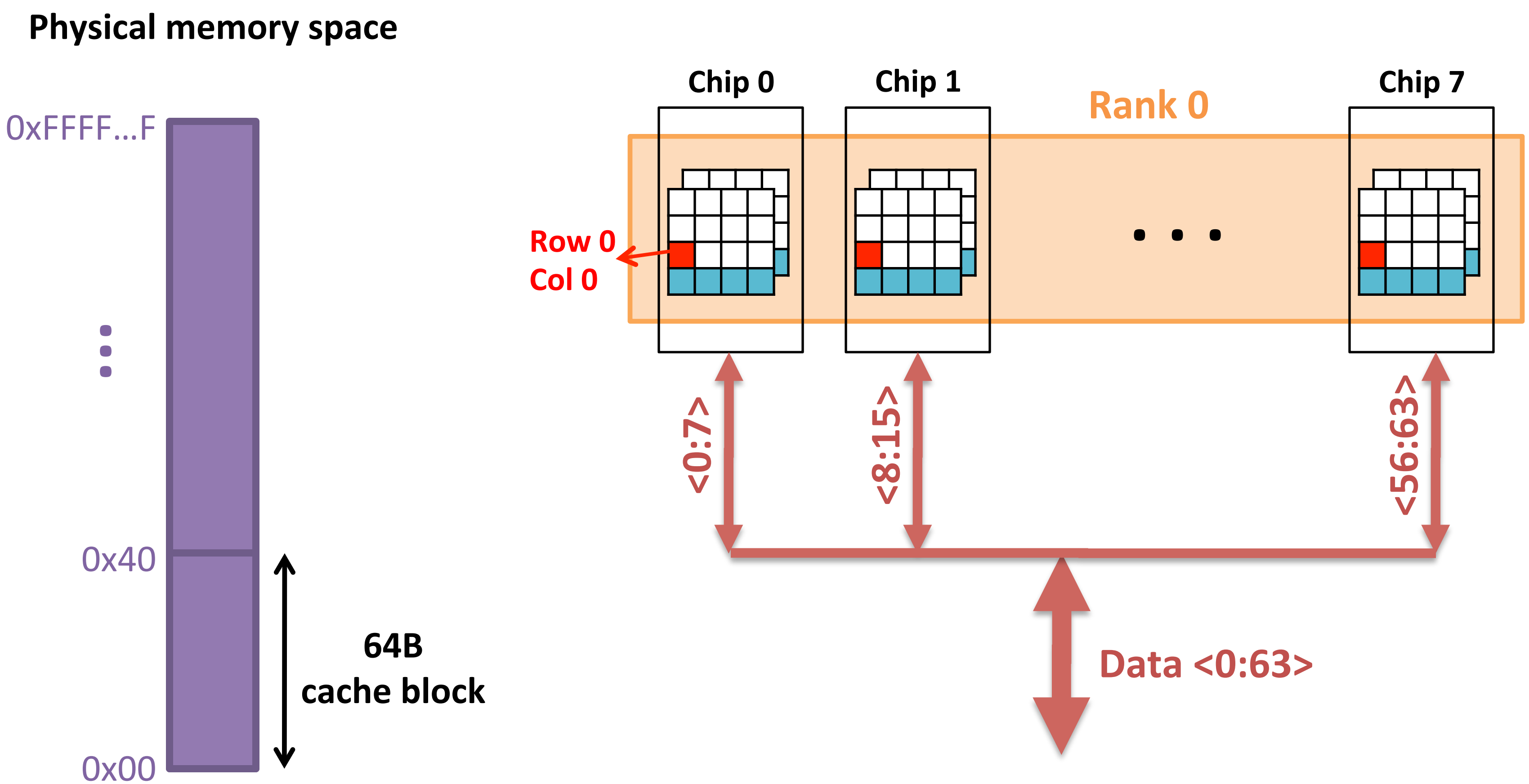


Example: Transferring a cache block

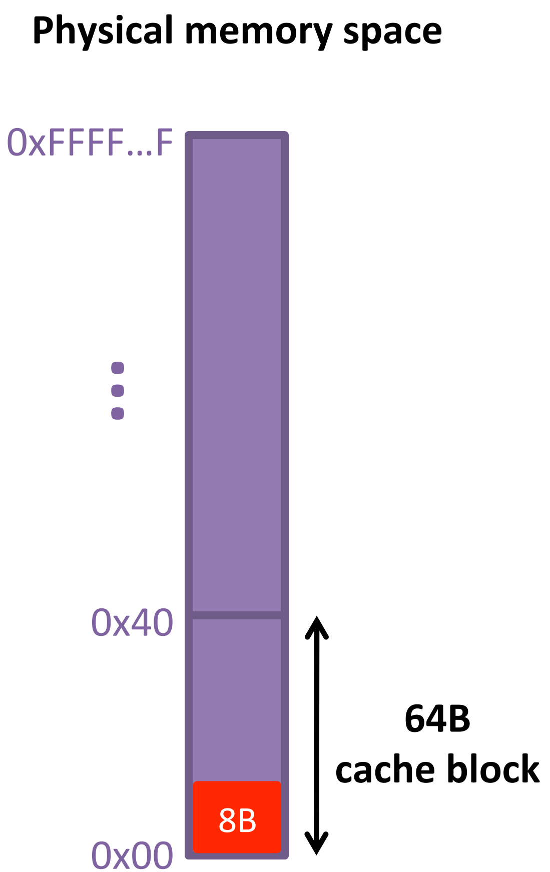


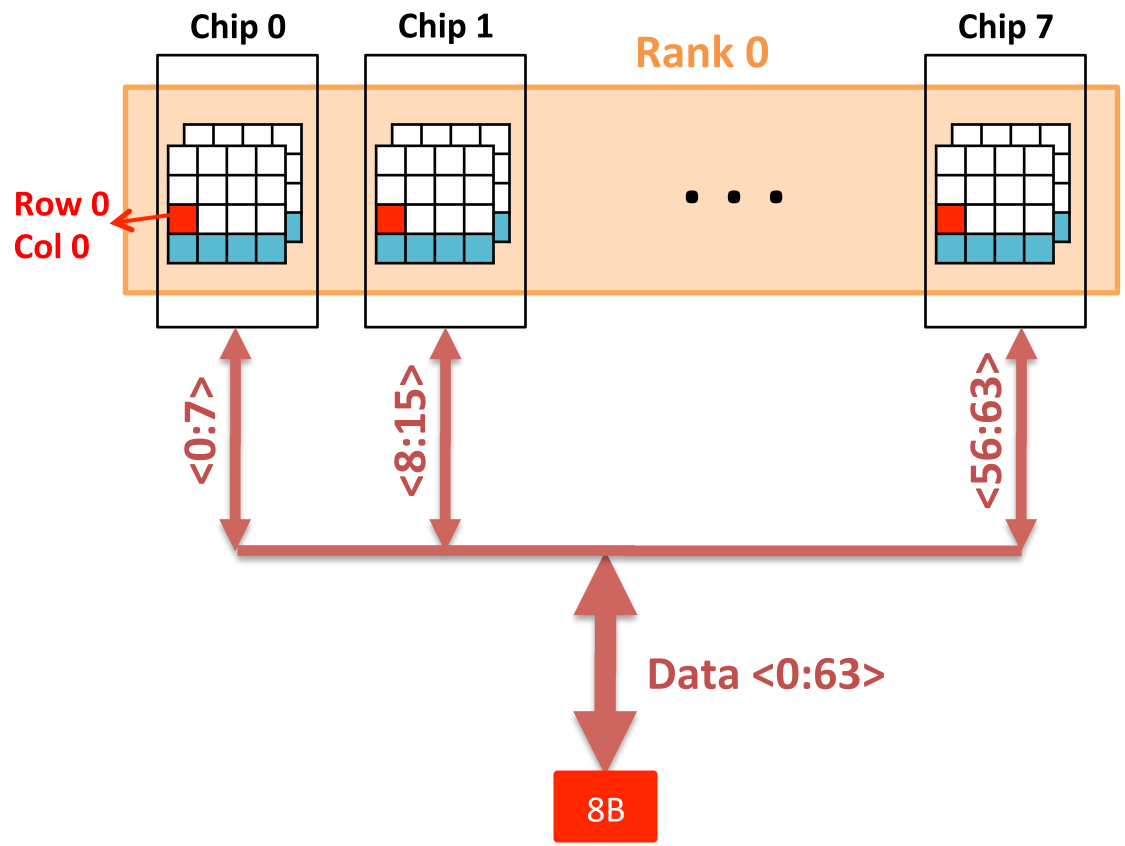


Example: Transferring a cache block

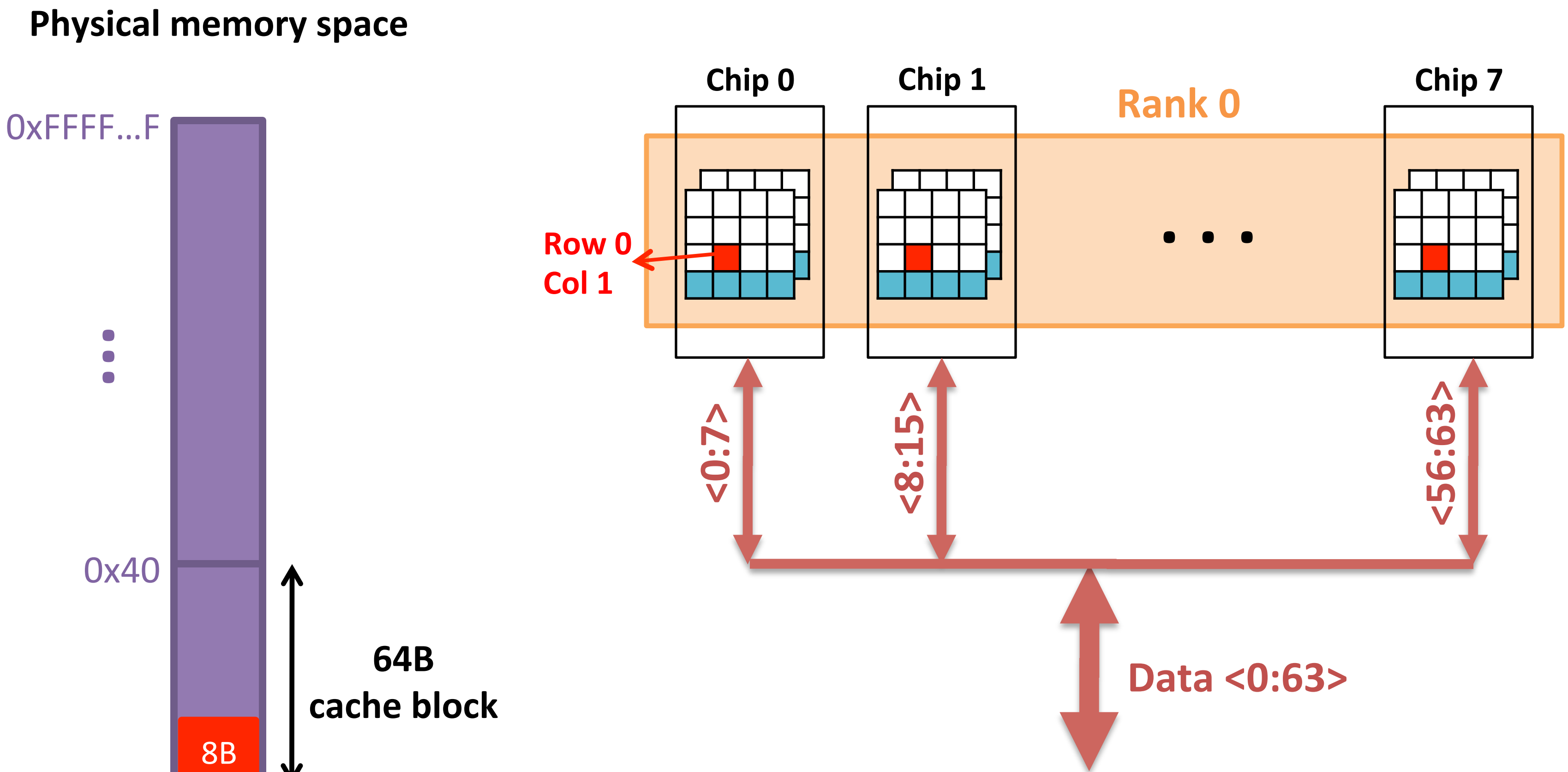


Example: Transferring a cache block

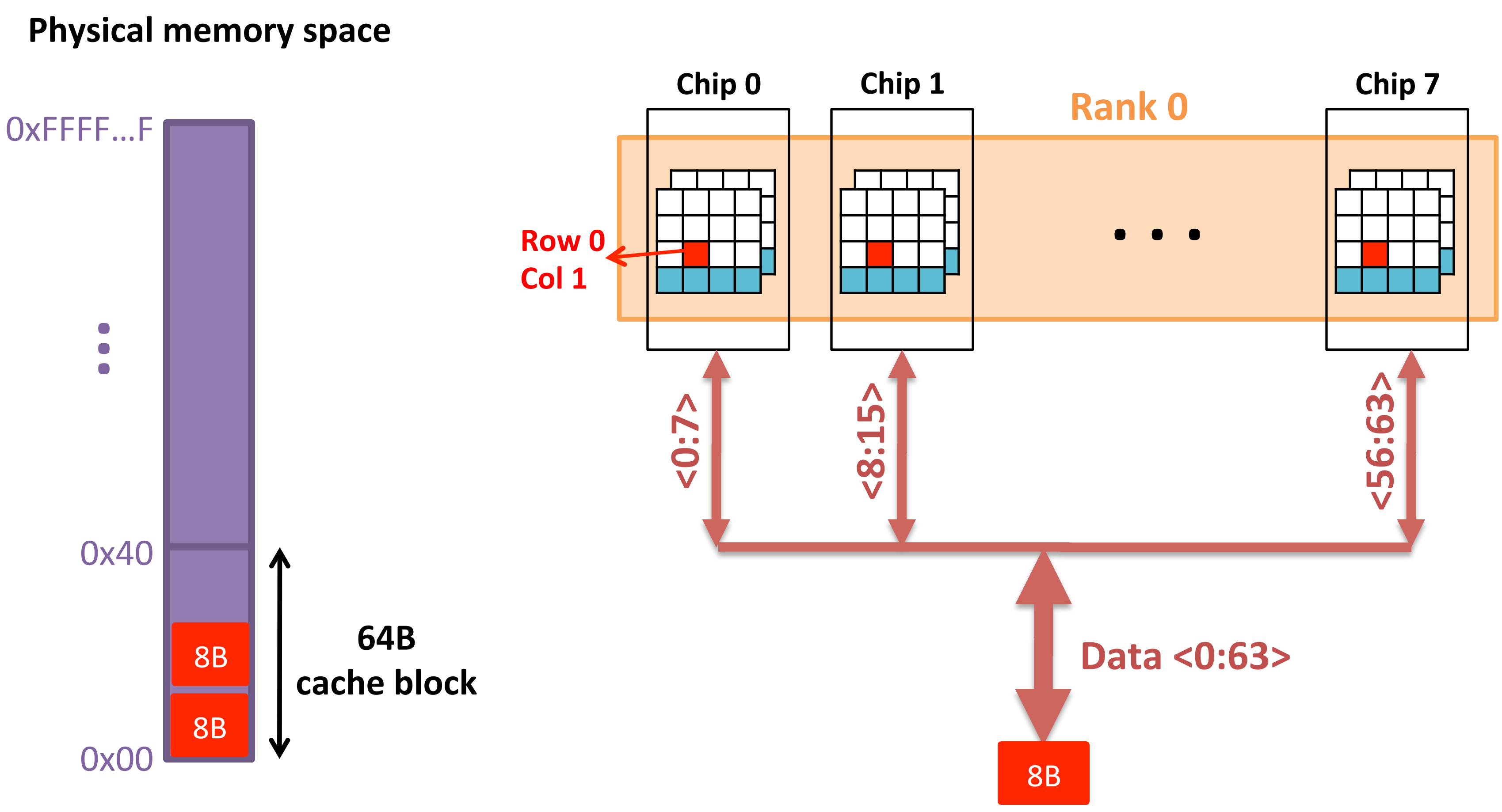




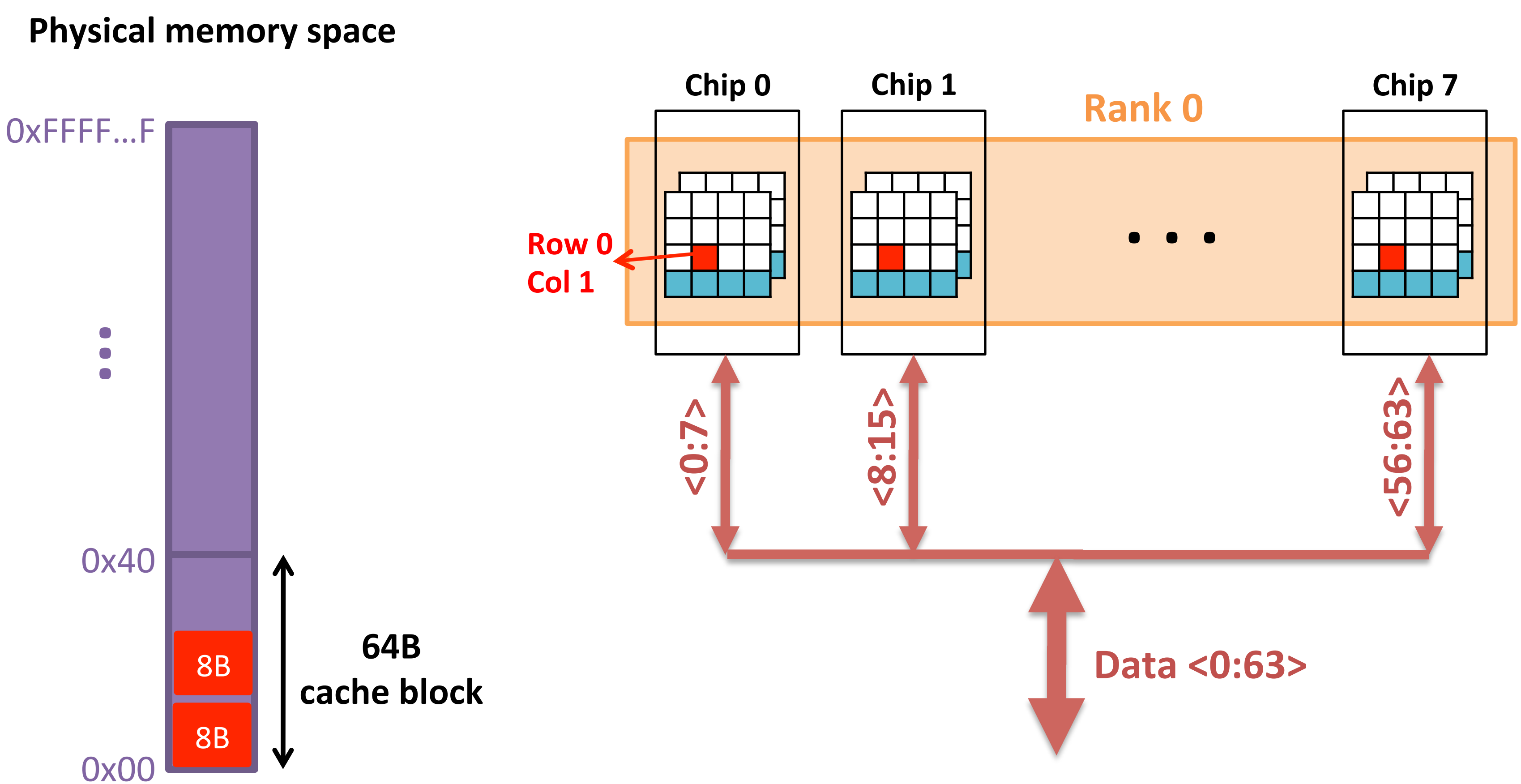
Example: Transferring a cache block



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.

Latency Components: Basic DRAM Operation

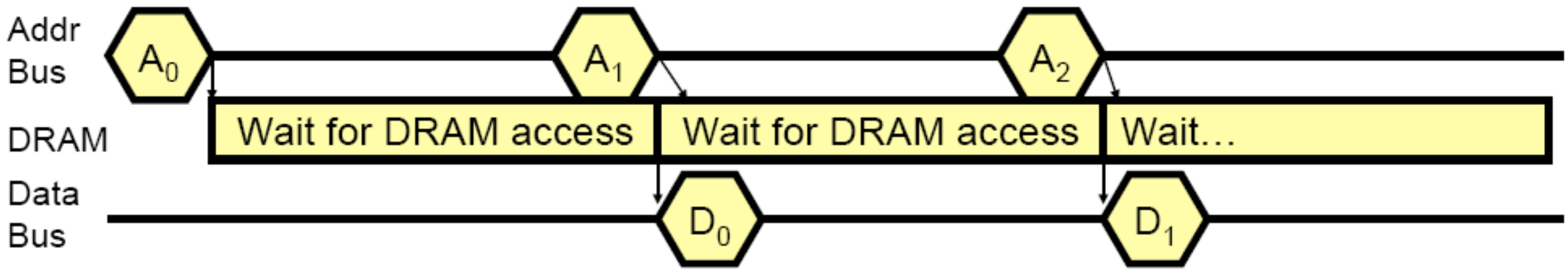
- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS (column address strobe) if row is “open” OR
  - RAS (row address strobe) + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)
- DRAM → Controller transfer time
  - Bus latency (BL)
- Controller to CPU transfer time

Multiple Banks (Interleaving) and Channels

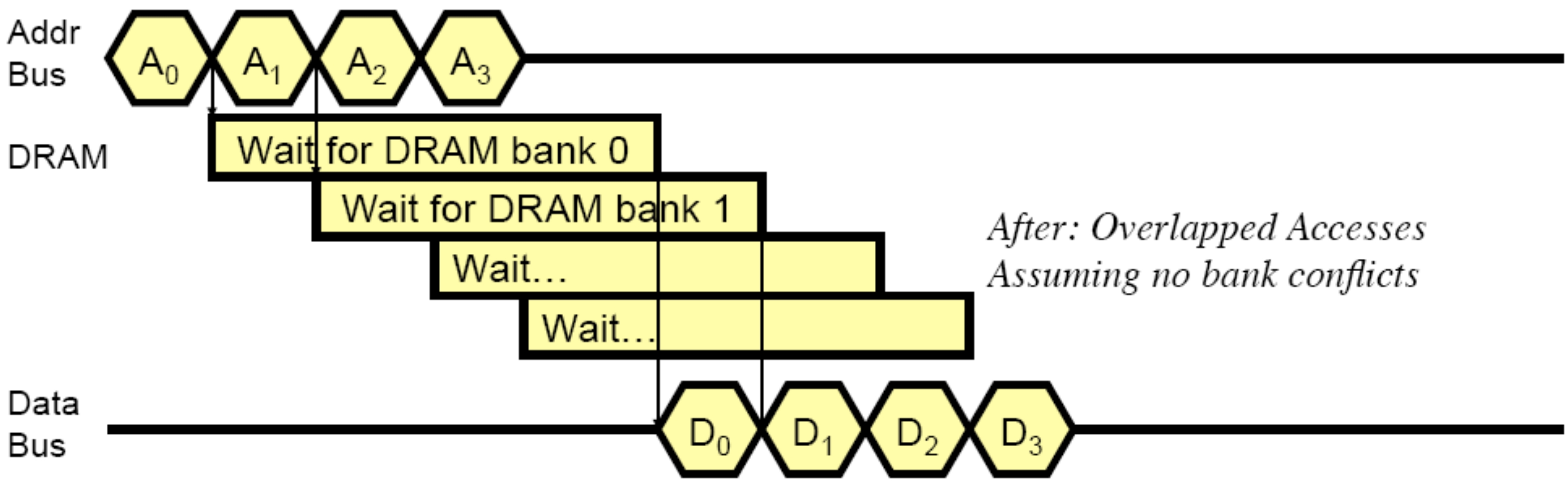
- Multiple banks
  - Enable concurrent DRAM accesses
  - Bits in address determine which bank an address resides in
- Multiple independent channels serve the same purpose
  - But they are even better because they have separate data buses
  - Increased bus bandwidth

- Enabling more concurrency requires reducing
  - Bank conflicts
  - Channel conflicts
- How to select/randomize bank/channel indices in address?
  - Lower order bits have more entropy
  - Randomizing hash functions (XOR of different address bits)

How Multiple Banks/Channels Help



Before: No Overlapping
Assuming accesses to different DRAM rows



After: Overlapped Accesses
Assuming no bank conflicts

Multiple Channels

- Advantages
  - Increased bandwidth
  - Multiple concurrent accesses (if independent channels)

- Disadvantages
  - Higher cost than a single channel
    - More board wires
    - More pins (if on-chip memory controller)

Address Mapping (Single Channel)

- Single-channel system with 8-byte memory bus
  - 2GB memory, 8 banks, 16K rows & 2K columns per bank

- Row interleaving
  - Consecutive rows of memory in consecutive banks

Row (14 bits)	Bank (3 bits)	Column (11 bits)	Byte in bus (3 bits)

  - Accesses to consecutive cache blocks serviced in a pipelined manner

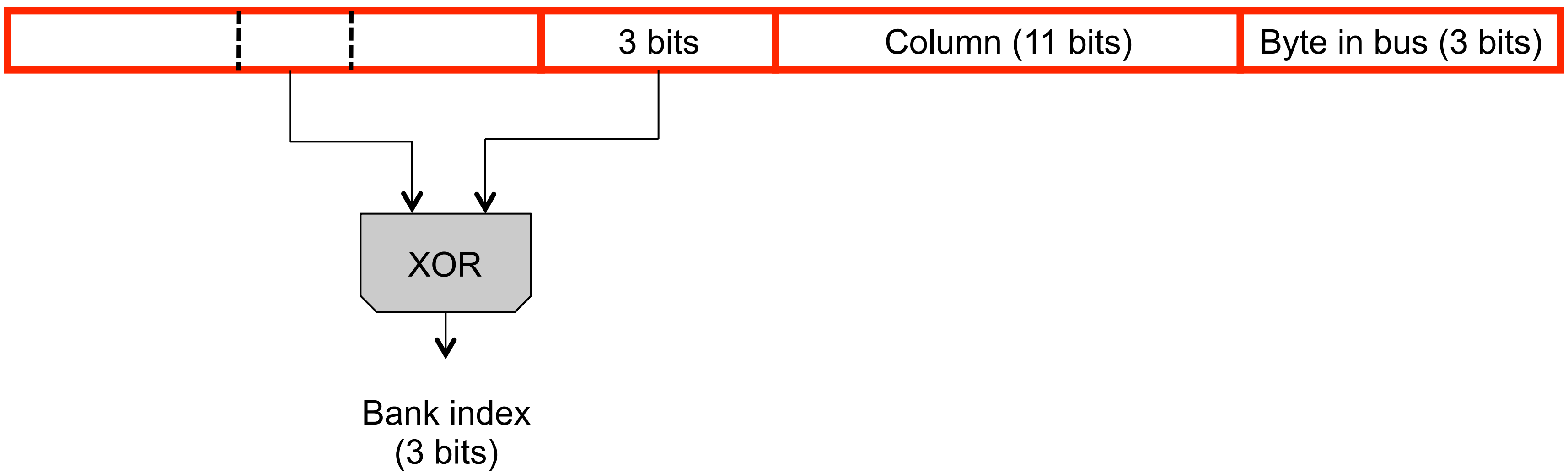
- Cache block interleaving
  - Consecutive cache block addresses in consecutive banks
  - 64 byte cache blocks

Row (14 bits)	High Column	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
	8 bits		3 bits	

  - Accesses to consecutive cache blocks can be serviced in parallel

Bank Mapping Randomization

- DRAM controller can randomize the address mapping to banks so that bank conflicts are less likely



Address Mapping (Multiple Channels)

C	Row (14 bits)	Bank (3 bits)	Column (11 bits)	Byte in bus (3 bits)

Row (14 bits)	C	Bank (3 bits)	Column (11 bits)	Byte in bus (3 bits)

Row (14 bits)	Bank (3 bits)	C	Column (11 bits)	Byte in bus (3 bits)

Row (14 bits)	Bank (3 bits)	Column (11 bits)	C	Byte in bus (3 bits)

- Where are consecutive cache blocks?

C	Row (14 bits)	High Column	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
		8 bits		3 bits	

Row (14 bits)	C	High Column	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
		8 bits		3 bits	

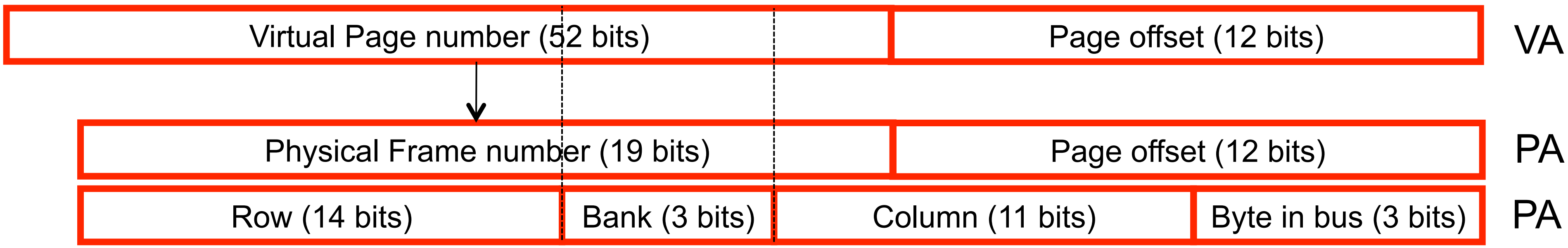
Row (14 bits)	High Column	C	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
	8 bits			3 bits	

Row (14 bits)	High Column	Bank (3 bits)	C	Low Col.	Byte in bus (3 bits)
	8 bits			3 bits	

Row (14 bits)	High Column	Bank (3 bits)	Low Col.	C	Byte in bus (3 bits)
	8 bits		3 bits		

Interaction with Virtual→Physical Mapping

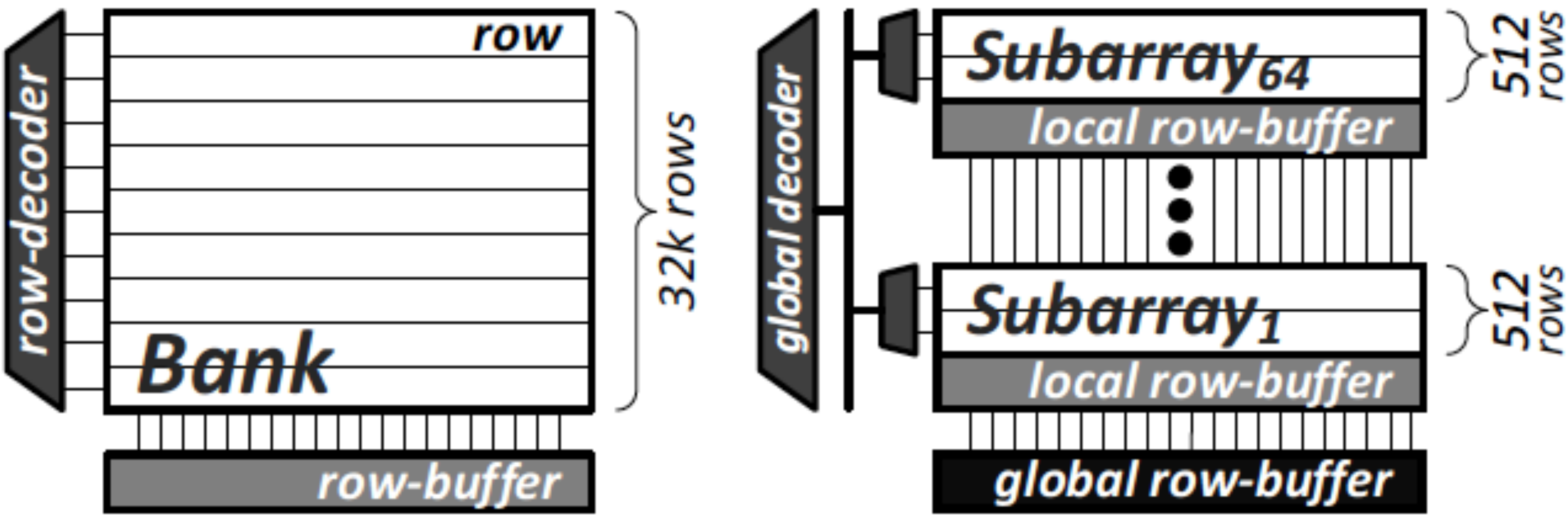
- Operating System influences where an address maps to in DRAM



- Operating system can influence which bank/channel/rank a virtual page is mapped to.
- It can perform page coloring to
  - Minimize bank conflicts
  - Minimize inter-application interference **[Muralidhara+ MICRO'11]**

More on Reducing Bank Conflicts

- Read Sections 1 through 4 of:
  - Kim et al., "A Case for Exploiting Subarray-Level Parallelism in DRAM," ISCA 2012.



(a) Logical abstraction (b) Physical implementation

Figure 1. DRAM bank organization