18-447
Computer Architecture
Lecture 22: Main Memory

Prof. Onur Mutlu
Carnegie Mellon University

Spring 2013, 3/26/2014

Announcements

- Homework 5 due date pushed out by two days
  - March 28 is the new due date

- Get started early on Lab 5
  - Due April 6

- Upcoming events that I encourage you all to attend
  - Computer Architecture Seminars on Memory (April 3)
  - Carnegie Mellon Cloud Workshop (April 4)

Computer Architecture Seminars

- Seminars relevant to many topics covered in 447
  - Caching
  - DRAM

- 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

Upcoming Seminar on DRAM (April 3)

- April 3, Thursday, 4pm, this room (CIC Panther Hollow)
- Prof. Rajeev Balasubramonian, Univ. of Utah
- Memory Architectures for Emerging Technologies and Workloads
  - The memory system will be a growing bottleneck for many workloads running on high-end servers. Performance improvements from technology scaling are also expected to decline in the coming decade. Therefore, new capabilities will be required in memory devices and memory controllers to achieve the next big leaps in performance and energy efficiency. Some of these capabilities will be inspired by emerging workloads (e.g., in-memory big-data, approximate computing, co-scheduled VMs), some will be inspired by new memory technologies (e.g., 3D stacking). The talk will discuss multiple early-stage projects in the Utah Arch lab that focus on DRAM parameter variation, near-data processing, and memory security.

Cloud Workshop All Day on April 4

- http://www.industry-academia.org/event-carnegie-mellon-cloud-workshop.html
- You need to register to attend. Gates 6115. Many talks:
- Keynote: Prof. Onur Mutlu – Carnegie Mellon – "Rethinking Memory System Design for Data-Intensive Computing"
- Prof. Rajeev Balasubramonian – Utah – "Practical Approaches to Memory Security in the Cloud"
- Bryan Chin – Cavium – "Head in the Clouds - Building a Chip for Scale-out Computing"
- Dr. Joon Kim - SK Hynix – "The Future of NVM Memories"
- Prof. Andy Pavlo - Carnegie Mellon – "OLTP on NVM: YMMV"
- Dr. John Busch – SanDisk – "The Impact of Flash Memory on the Future of Cloud Computing"
- Keynote: Prof. Greg Ganger – Carnegie Mellon – "Scheduling Heterogeneous Resources in Cloud Datacenters"
- Paul Rad – Rackspace – "OpenStack-Based High Performance Cloud Architecture"
- Charles Butler – Ubuntu – "Cloud Service Orchestration with JuJu"
- Prof. Mor Harchol-Balter - Carnegie Mellon – "Dynamic Power Management in Data Centers"
- Prof. Eric Xing – Carnegie Mellon – "Petuum: A New Platform for Cloud-based Machine Learning to Efficiently Solve Big Data Problems"
- Majid Bemanian – Imagination Technologies – "Security in the Cloud and Virtualized Mobile Devices"
- Robert Broberg – Cisco – "Cloud Security Challenges and Solutions"

Cloud Career Fair on April 4

- http://www.industry-academia.org/event-carnegie-mellon-cloud-workshop.html
- Gates 6121, 11am-3pm
- Runs in Room 6121 in parallel to the Tech Forum, from 11am to 3PM. IAP members will have informational/recruiting tables on site. During the breaks in the technical presentations and lunch, the Tech Forum attendees can network on lining up an internship or that first full-time engineering job. Students who are only interested and/or able to attend the Career Fair are welcome to do so, but please indicate this specific interest on your registration application (see the "Register Here" button below).

Enabling High Bandwidth Memories

Multiple Instructions per Cycle

- Can generate multiple cache/memory accesses per cycle
- How do we ensure the cache/memory can handle multiple accesses in the same clock cycle?

- Solutions:
  - true multi-porting
  - virtual multi-porting (time sharing a port)
  - multiple cache copies
  - banking (interleaving)

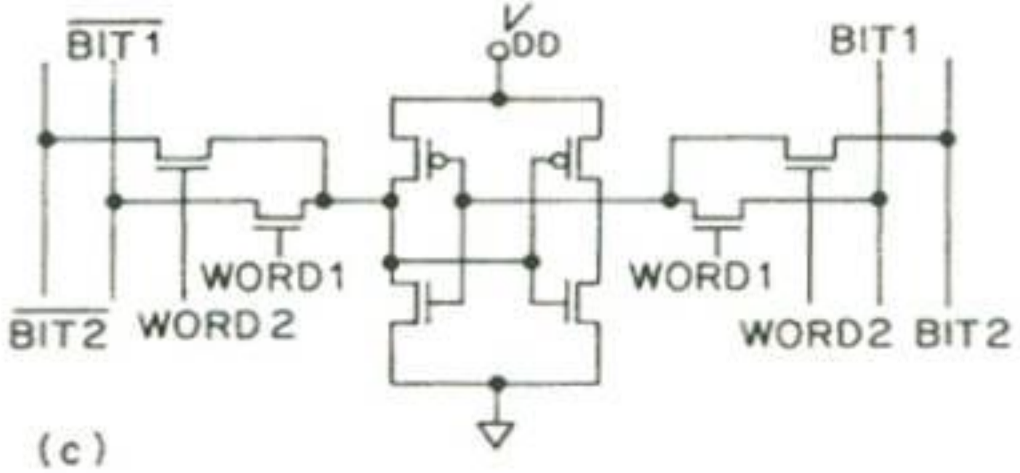
Handling Multiple Accesses per Cycle (I)

- True multiporting
  - Each memory cell has multiple read or write ports

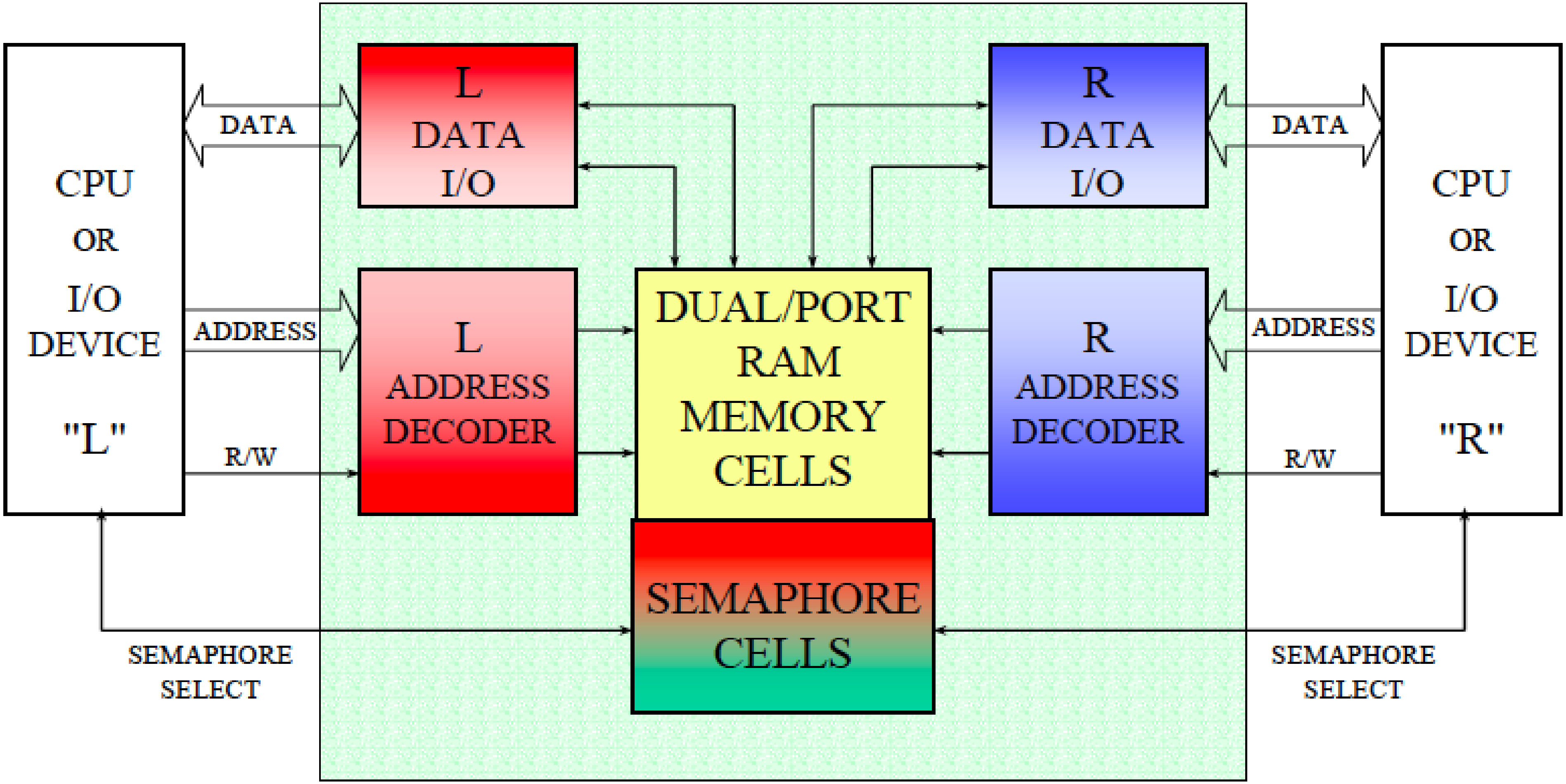
  + Truly concurrent accesses (no conflicts on read accesses)

  -- Expensive in terms of latency, power, area

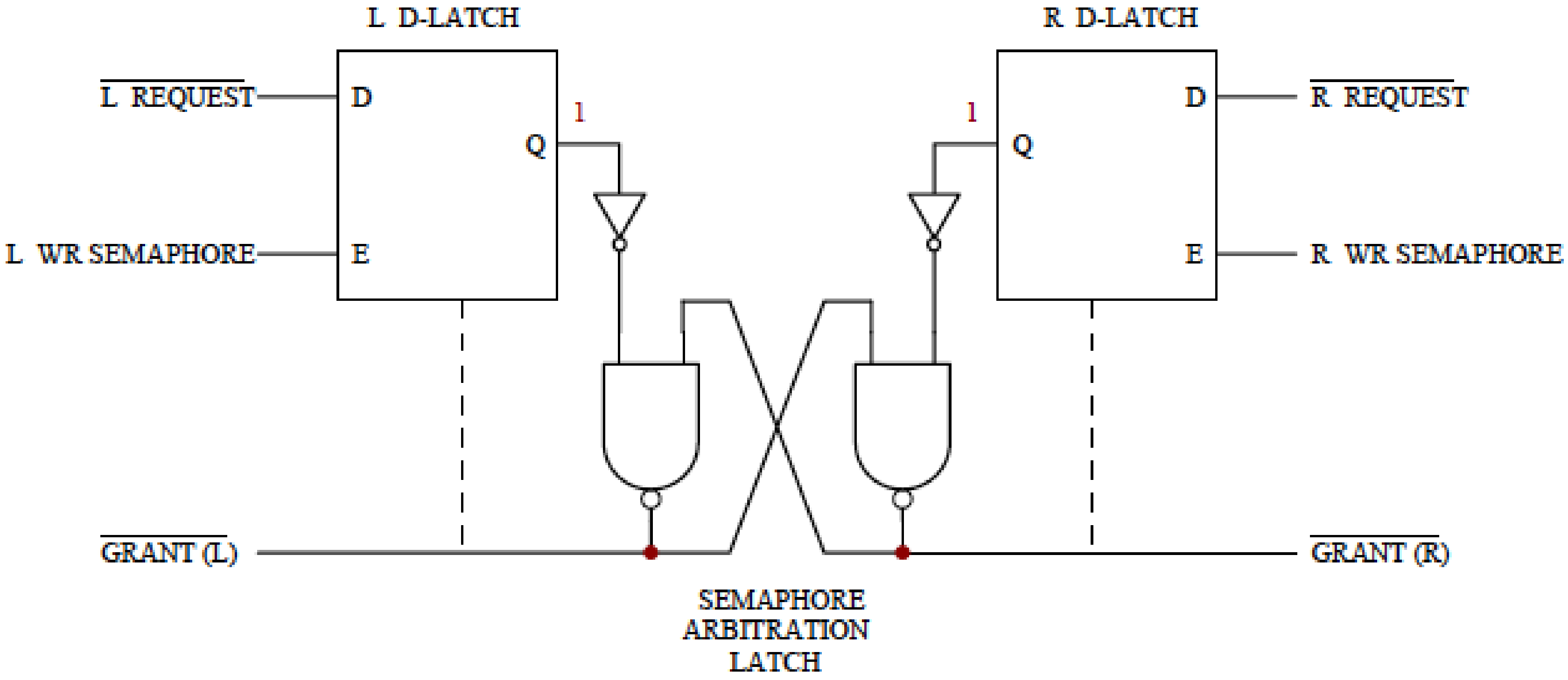
  - What about read and write to the same location at the same time?
    - Peripheral logic needs to handle this



Peripheral Logic for True Multiporting



Peripheral Logic for True Multiporting

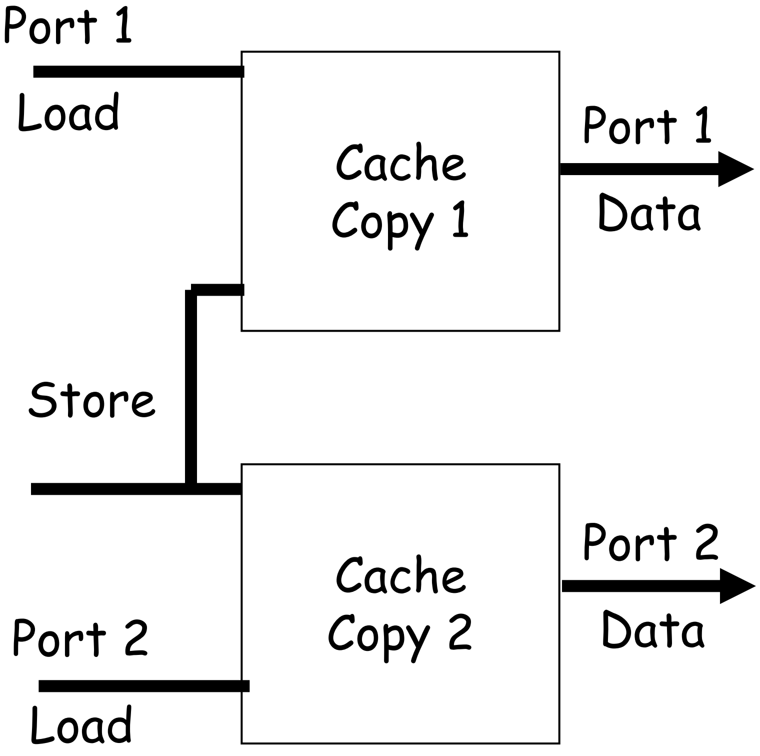

L D-LATCH
R D-LATCH
$\overline{\text{L REQUEST}}$
D
Q
1
L WR SEMAPHORE
E
$\overline{\text{R REQUEST}}$
D
1
Q
E
R WR SEMAPHORE
$\overline{\text{GRANT (L)}}$
$\overline{\text{GRANT (R)}}$
SEMAPHORE
ARBITRATION
LATCH

Handling Multiple Accesses per Cycle (II)

- Virtual multiporting
  - Time-share a single port
  - Each access needs to be (significantly) shorter than clock cycle
  - Used in Alpha 21264
  - Is this scalable?

Handling Multiple Accesses per Cycle (III)

- Multiple cache copies
  - Stores update both caches
  - Loads proceed in parallel
- Used in Alpha 21164
- Scalability?
  - Store operations form a bottleneck
  - Area proportional to "ports"



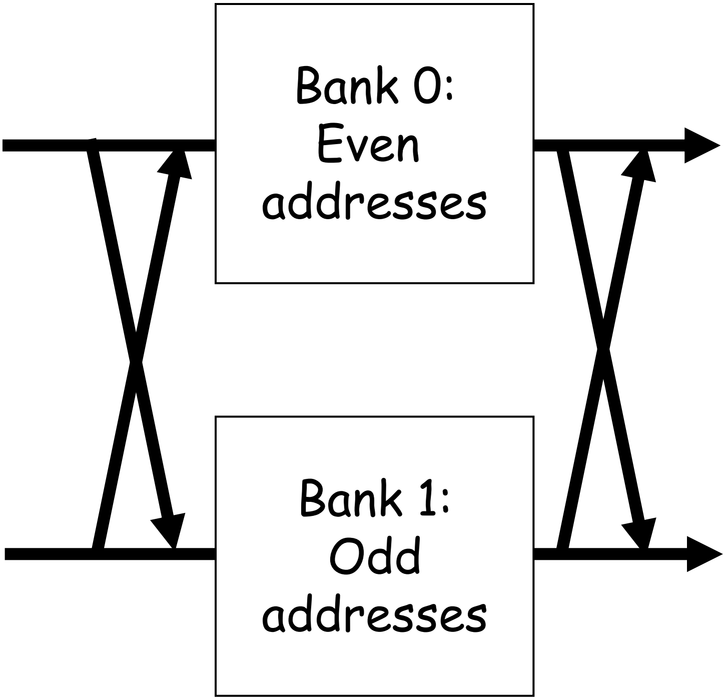
Handling Multiple Accesses per Cycle (III)

- Banking (Interleaving)
  - Bits in address determines which bank an address maps to
    - Address space partitioned into separate banks
    - Which bits to use for "bank address"?

  + No increase in data store area

  -- Cannot satisfy multiple accesses to the same bank

  -- Crossbar interconnect in input/output



- Bank conflicts
  - Two accesses are to the same bank
  - How can these be reduced?
    - Hardware? Software?

General Principle: Interleaving

- 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?)

Further Readings on Caching and MLP

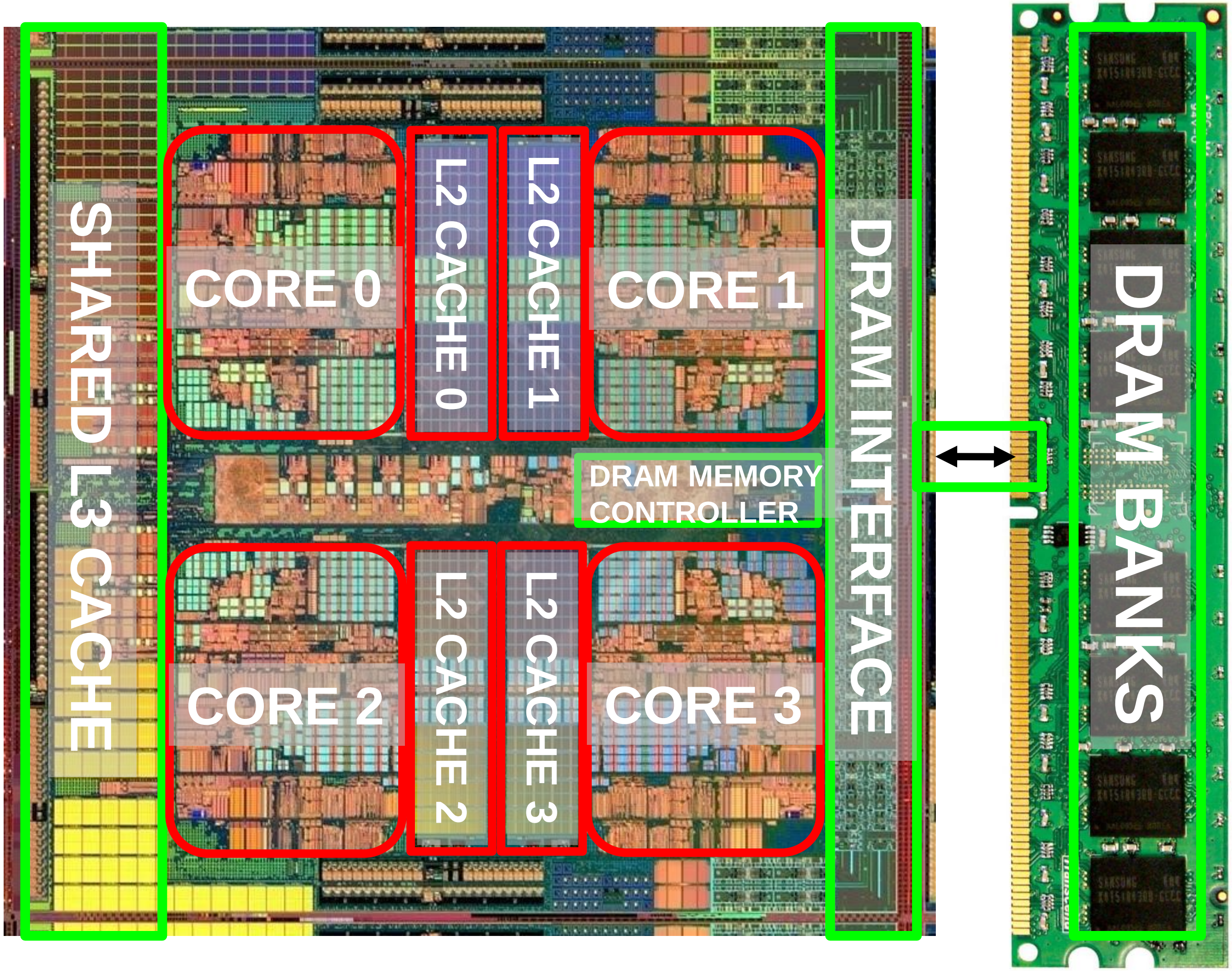
- Required: Qureshi et al., "A Case for MLP-Aware Cache Replacement," ISCA 2006.

- Glew, "MLP Yes! ILP No!," ASPLOS Wild and Crazy Ideas Session, 1998.

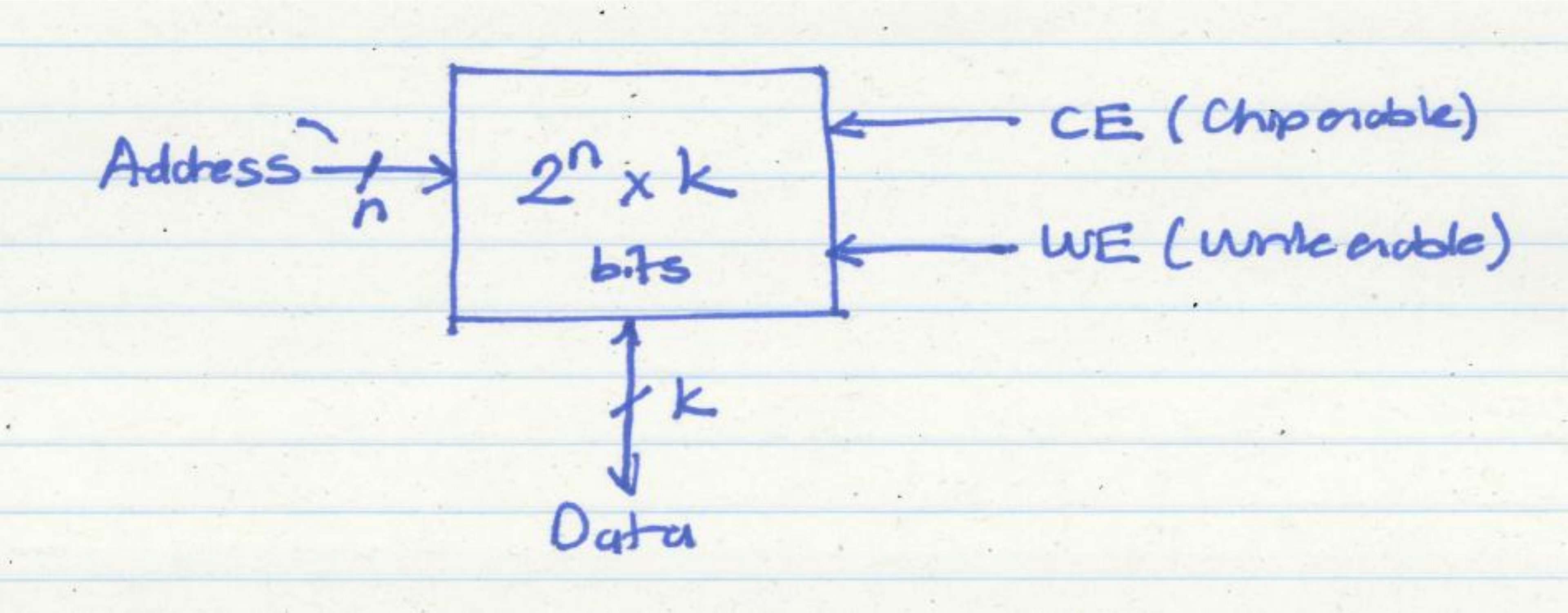
- Mutlu et al., "Runahead Execution: An Effective Alternative to Large Instruction Windows," IEEE Micro 2003.

Main Memory

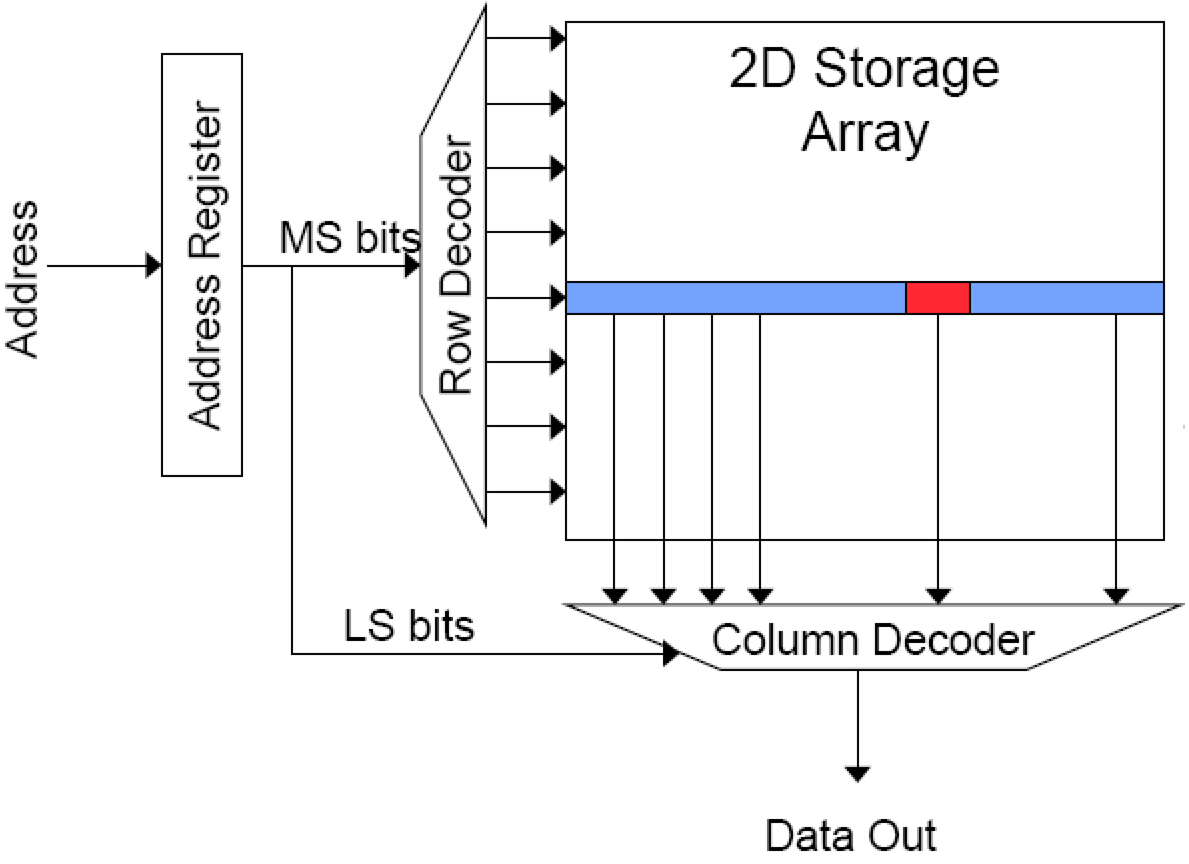
Main Memory in the System



The Memory Chip/System Abstraction



Review: Memory Bank Organization



- Read access sequence:

1. Decode row address & drive word-lines

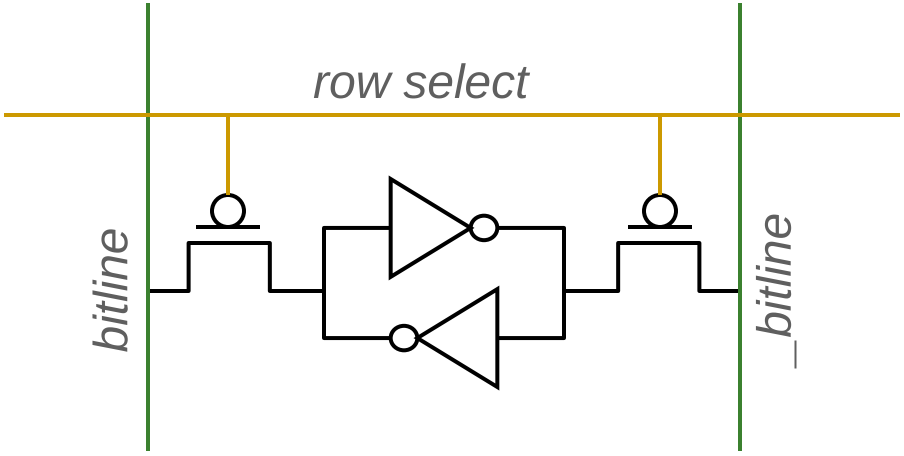
2. Selected bits drive bit-lines
   - Entire row read

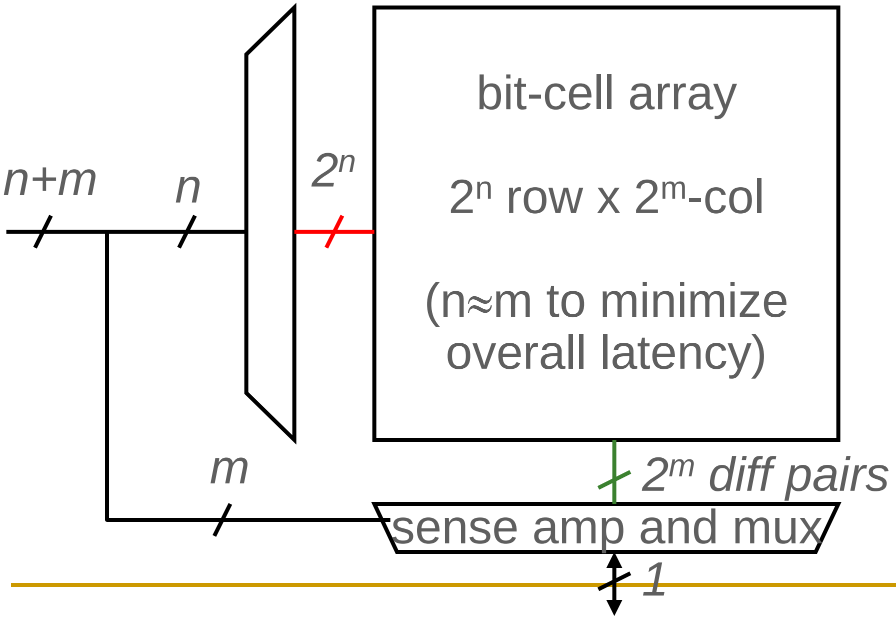
3. Amplify row data

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

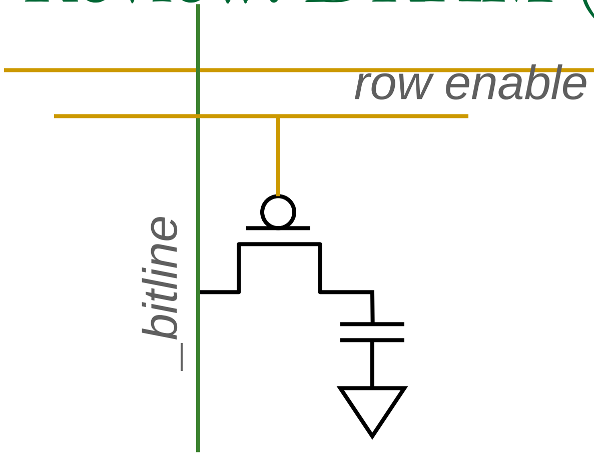
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. diff. sensing and col. select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

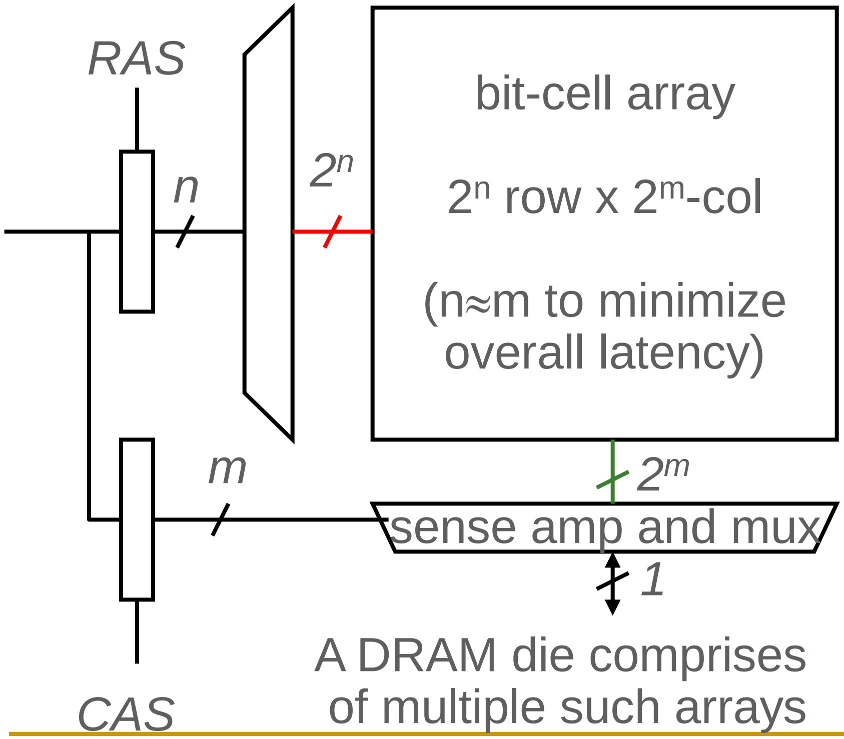
Access latency dominated by steps 2 and 3

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

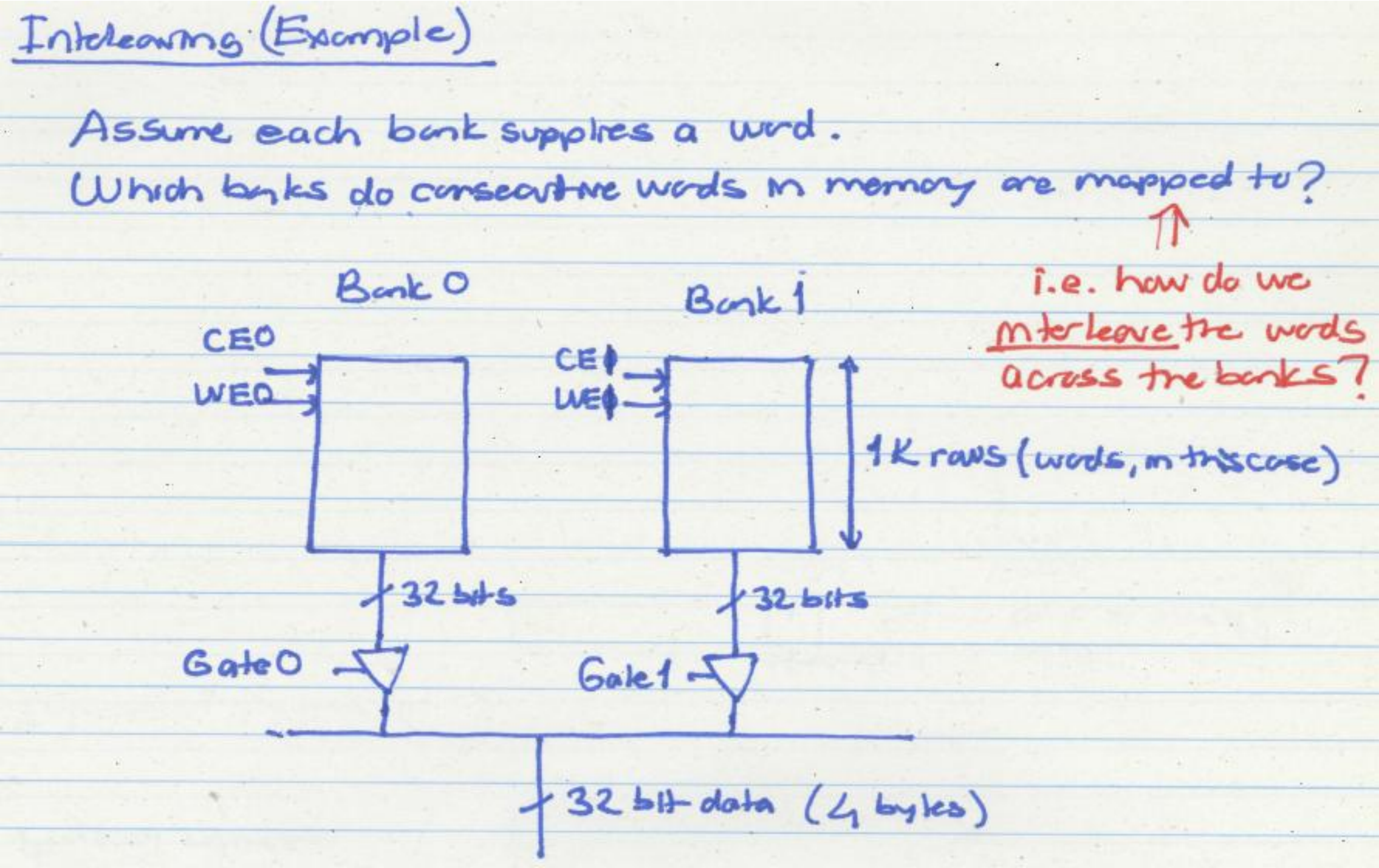
- Alignment
  - Does the hardware support unaligned access transparently to software?

- 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



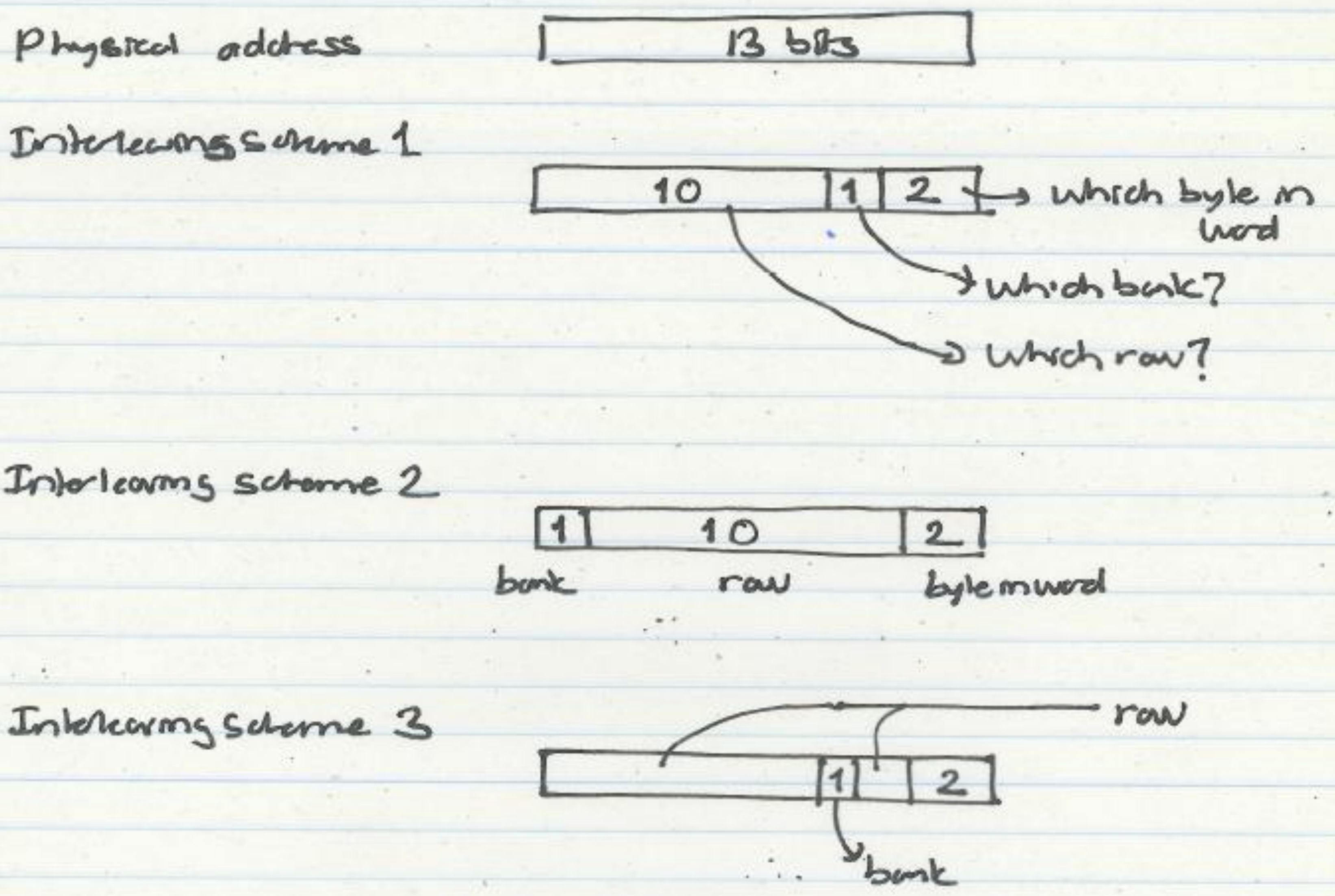
Interleaving (Example)

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

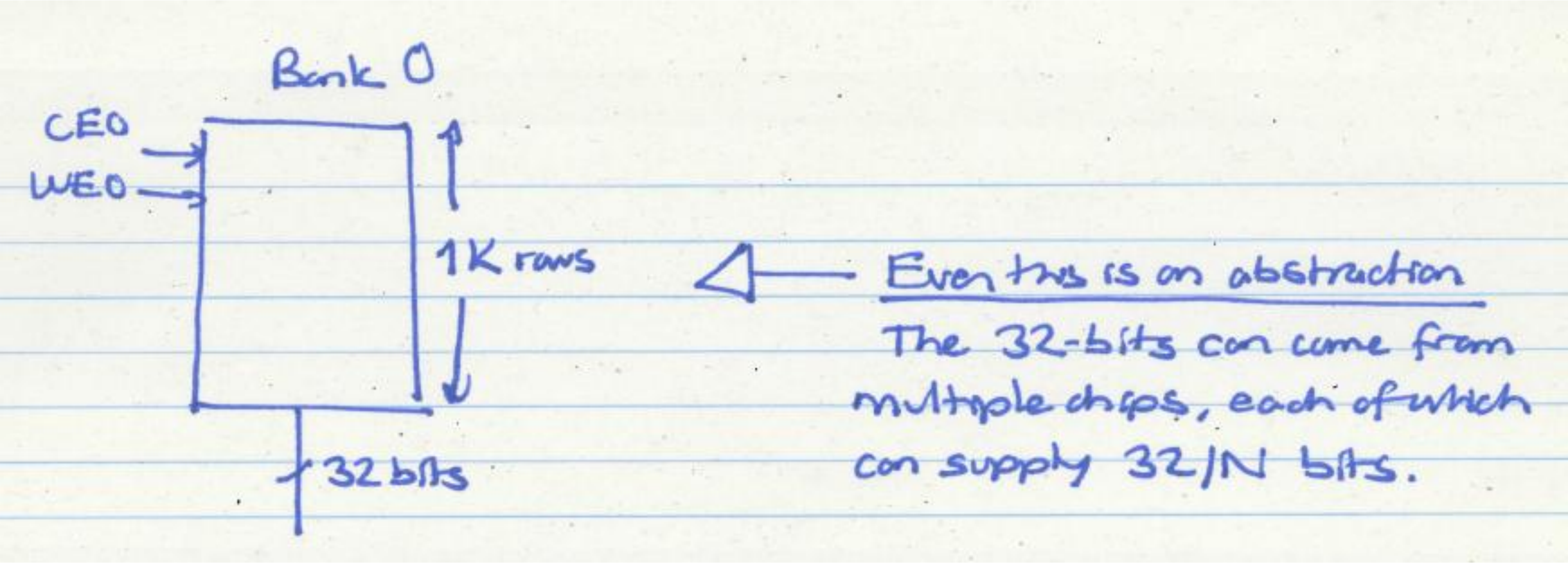
- Remember CRAY-1 with 16 banks
  - 11 cycle bank latency
  - Consecutive words in memory in consecutive banks (word interleaving)
  - 1 access can be started (and finished) per cycle

- Can banks be operated *fully* in parallel?
  - Multiple accesses started per cycle?

- What is the cost of this?
  - We have seen it earlier (today)

- Modern superscalar processors have L1 data caches with multiple, fully-independent banks; DRAM banks share buses

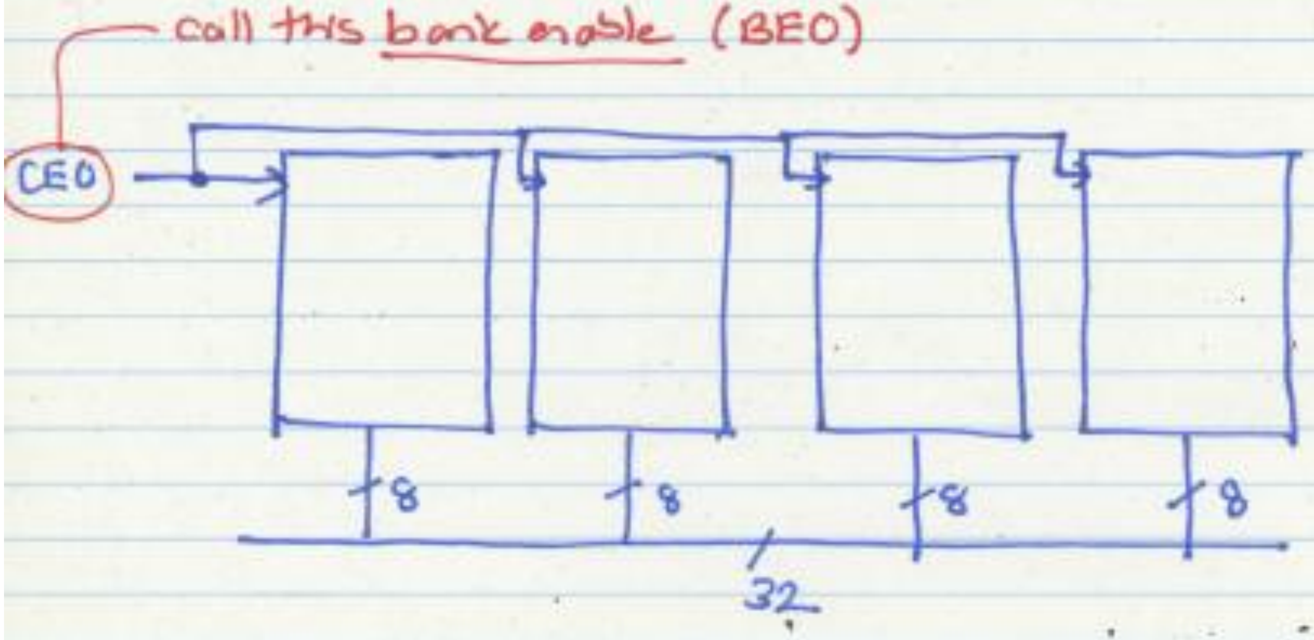
The Bank Abstraction



Even this is an abstraction

The 32-bits can come from multiple chips, each of which can supply 32/N bits.

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

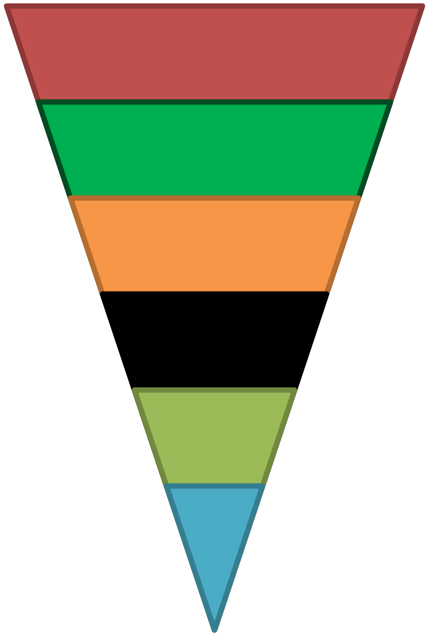
Why? Producing an 8-bit/pin chip cheaper than producing a 32-bit/pin chip

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.

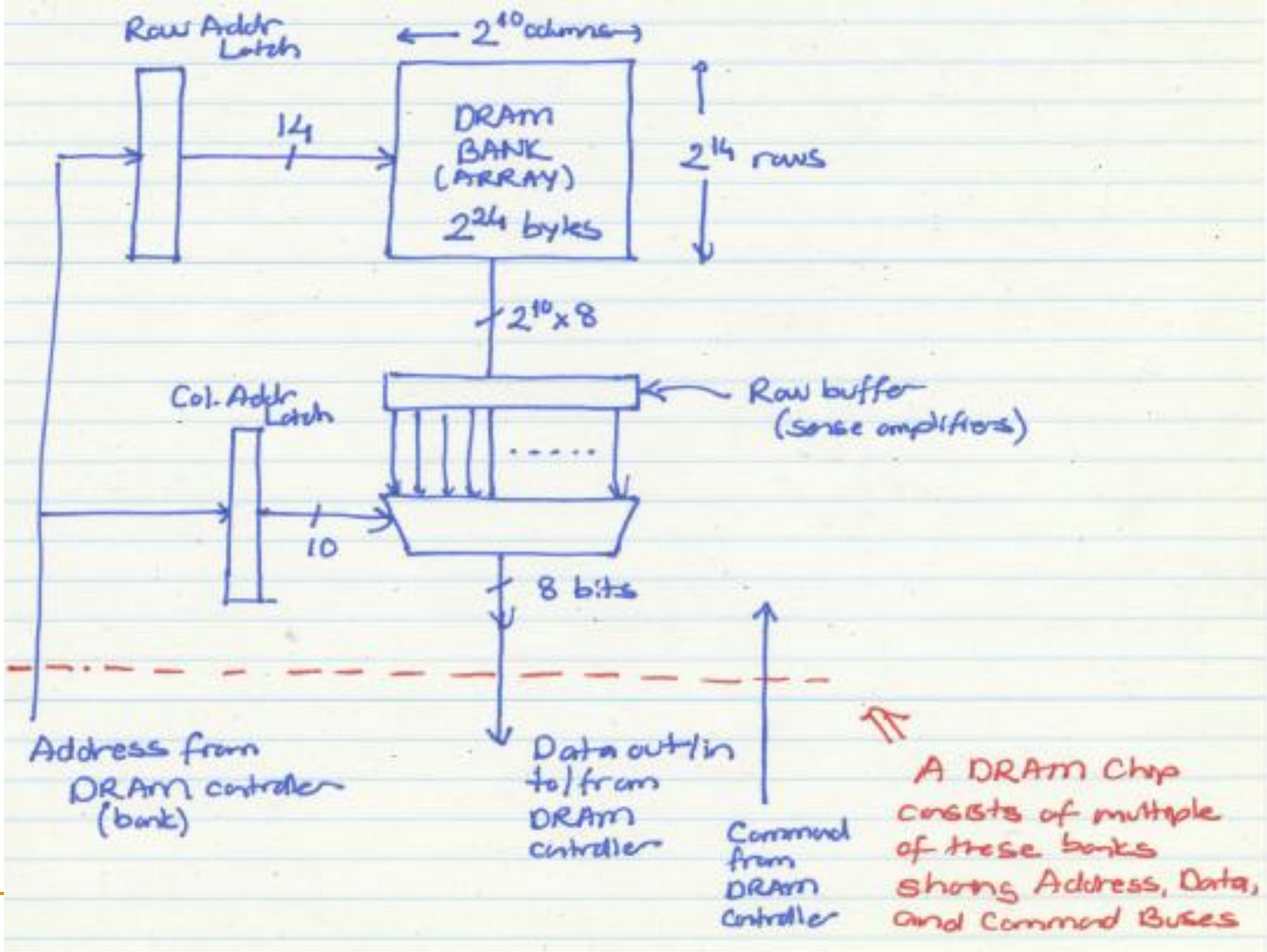
The DRAM Subsystem

DRAM Subsystem Organization

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



The DRAM Bank Structure


Row Addr Latch
2^10 columns
DRAM BANK (ARRAY)
2^24 bytes
14
2^14 rows
2^10 x 8
Row buffer (sense amplifiers)
Col. Addr Latch
10
8 bits
Address from DRAM controller (bank)
Data out/in to/from DRAM controller
Command from DRAM Controller
A DRAM Chip consists of multiple of these banks showing Address, Data, and Command Buses

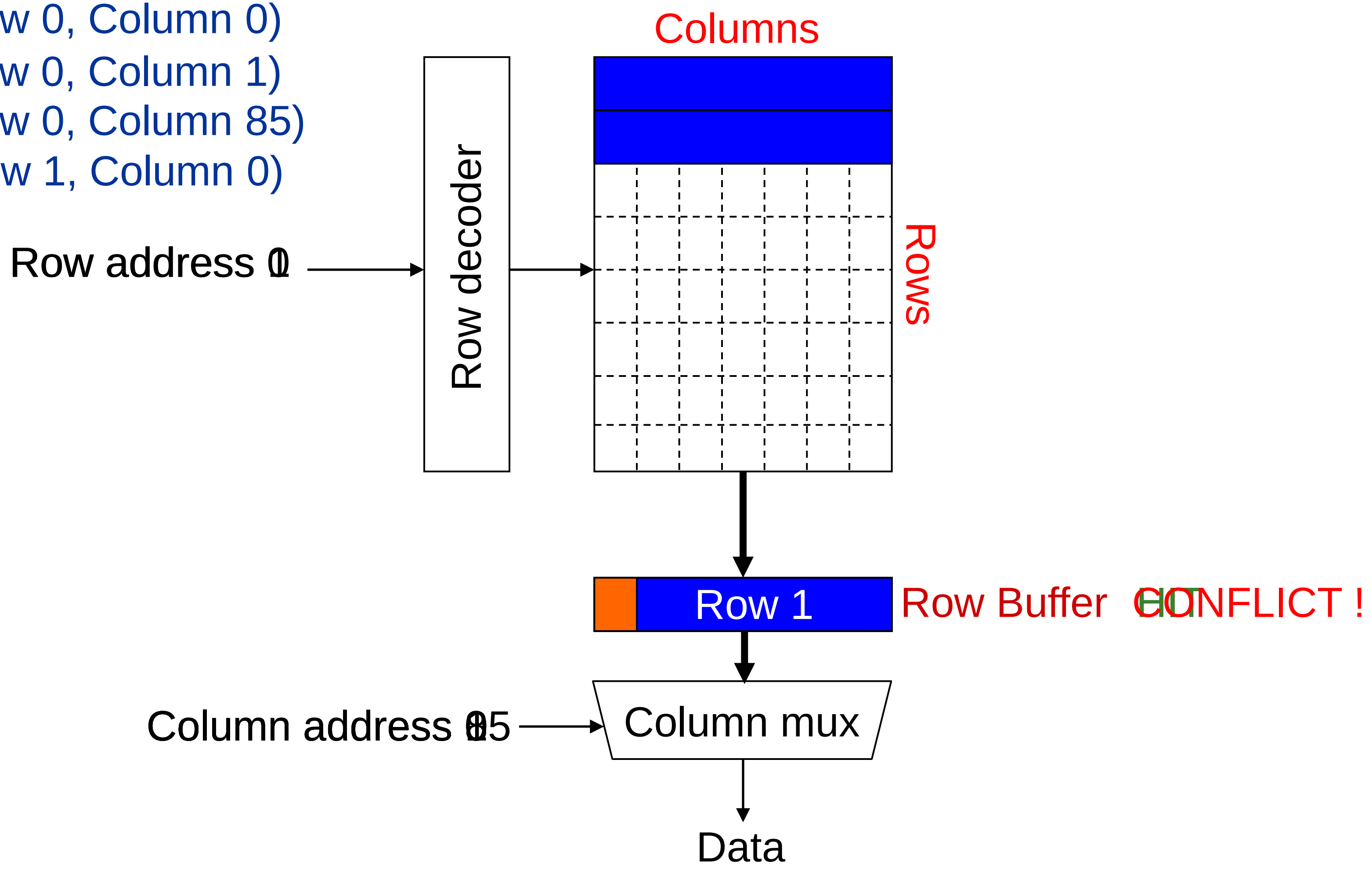
Page Mode DRAM

- 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 activate command

DRAM Bank Operation

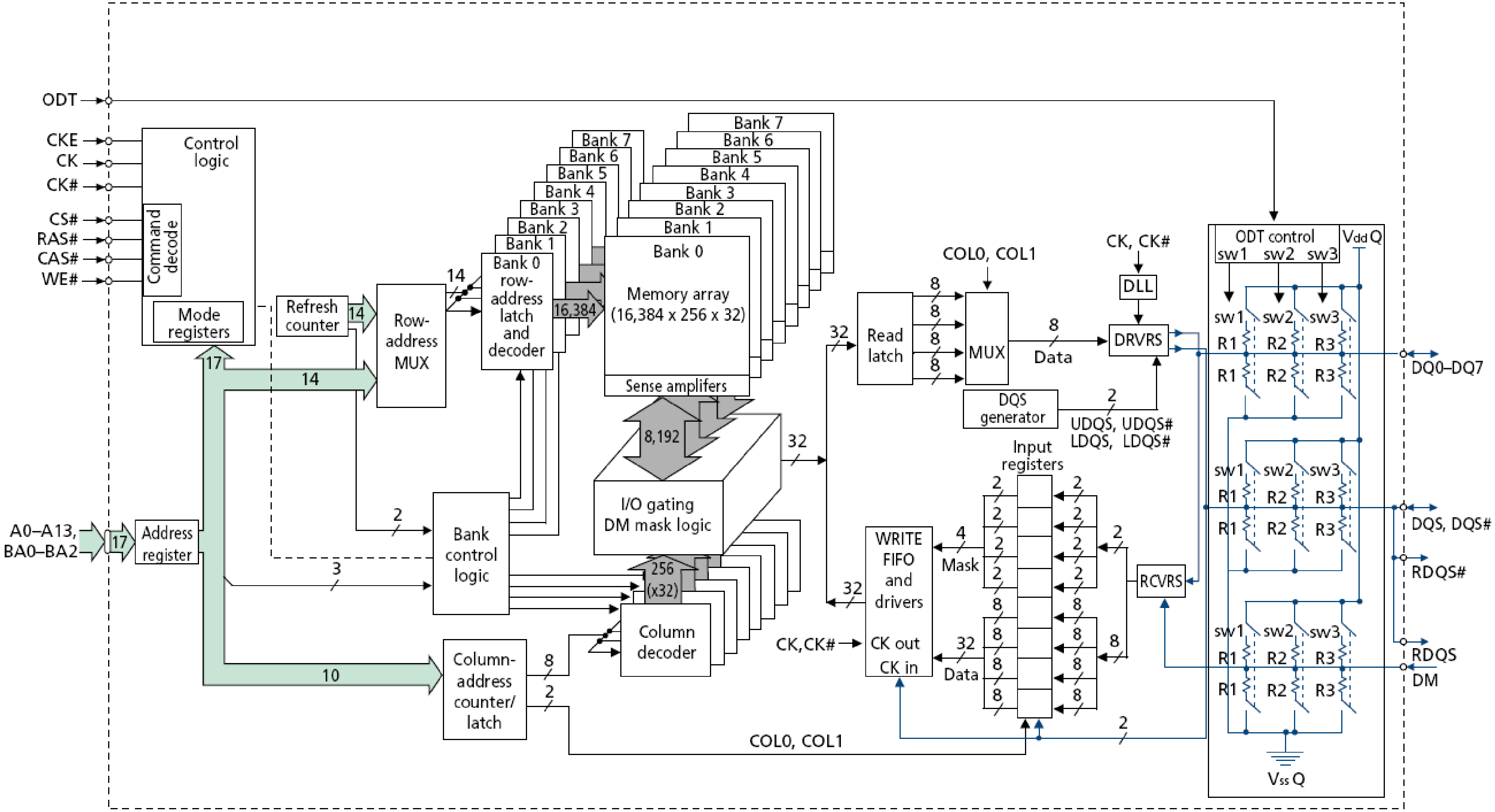
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



The DRAM Chip

- Consists of multiple banks (2-16 in Synchronous DRAM)
- Banks share command/address/data buses
- The chip itself has a narrow interface (4-16 bits per read)

128M x 8-bit DRAM Chip



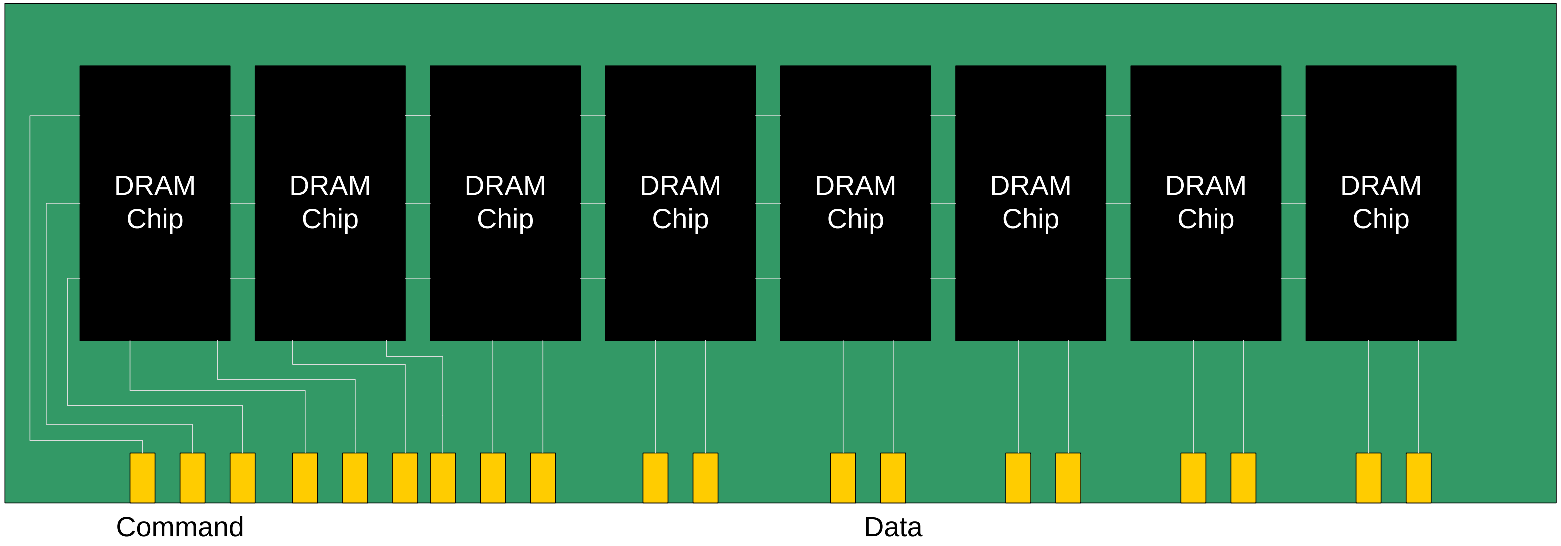
DRAM Rank and Module

- 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

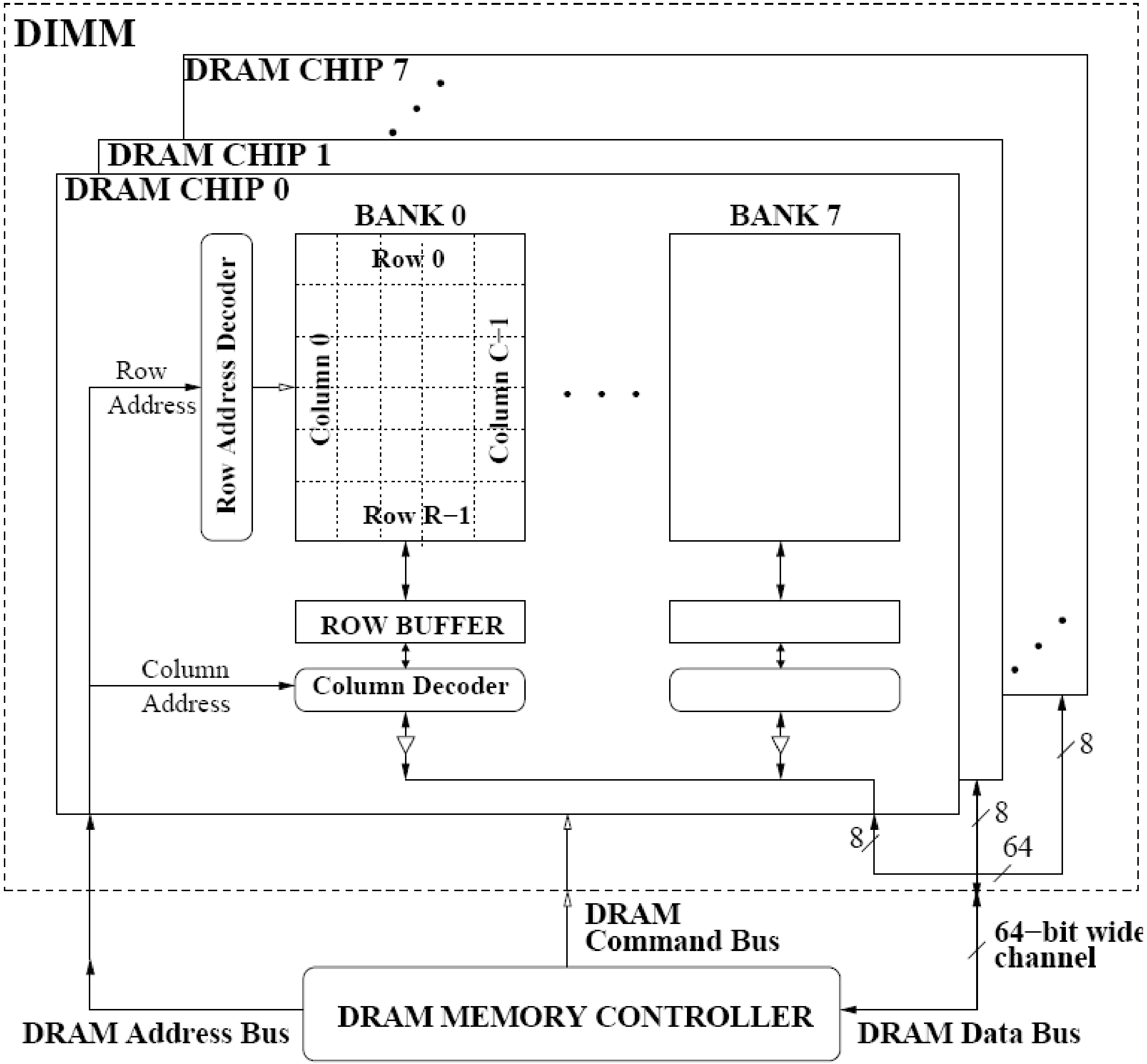
- A DRAM module consists of one or more ranks
  - E.g., DIMM (dual inline memory module)
  - This is what you plug into your motherboard

- 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)

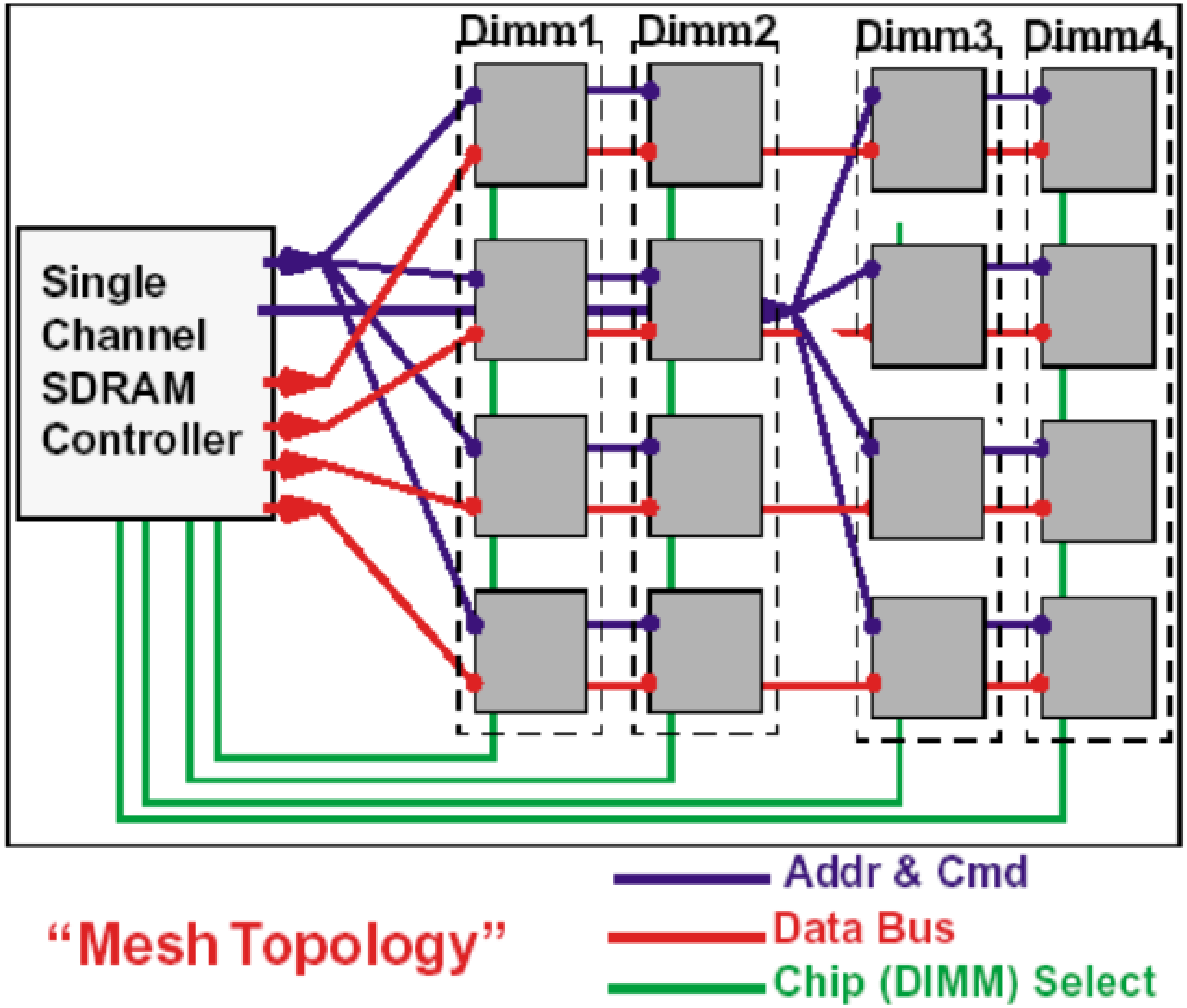


A 64-bit Wide DIMM (One Rank)



- 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

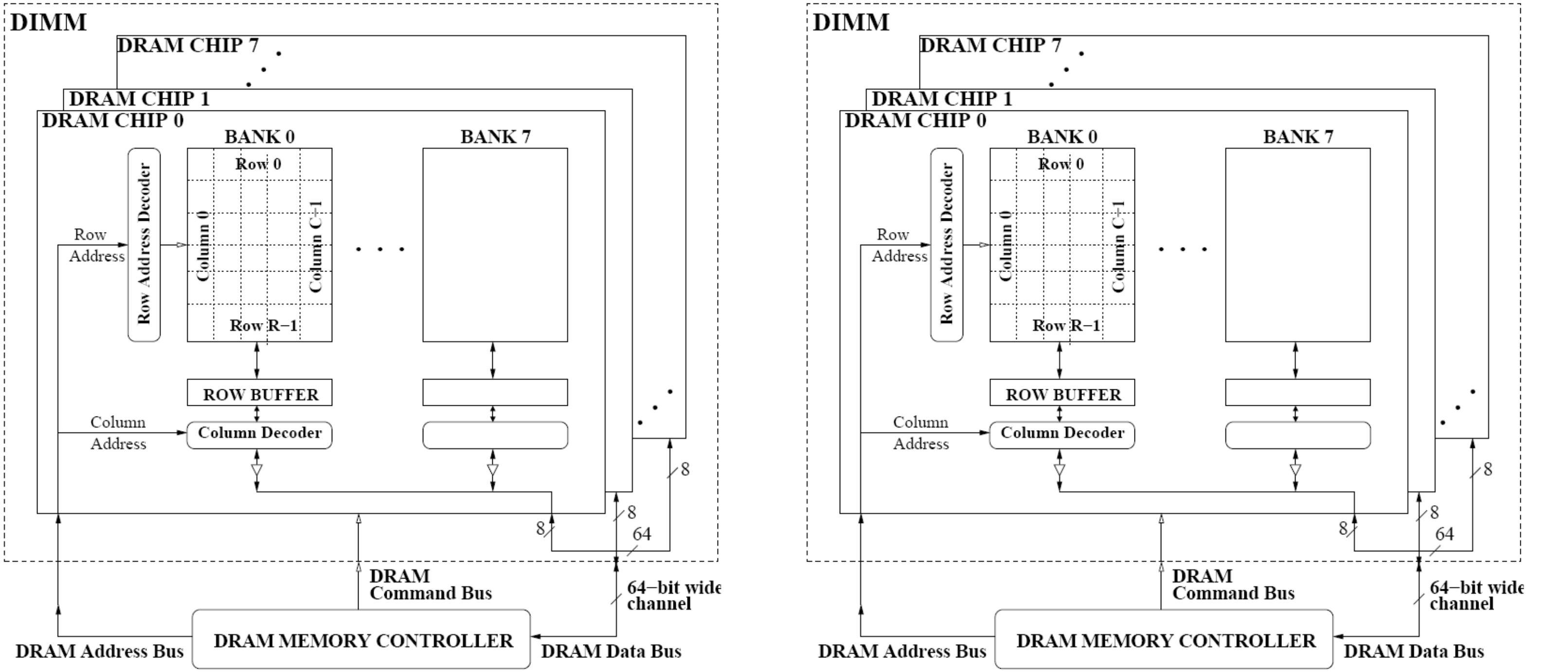
Multiple DIMMs



"Mesh Topology"

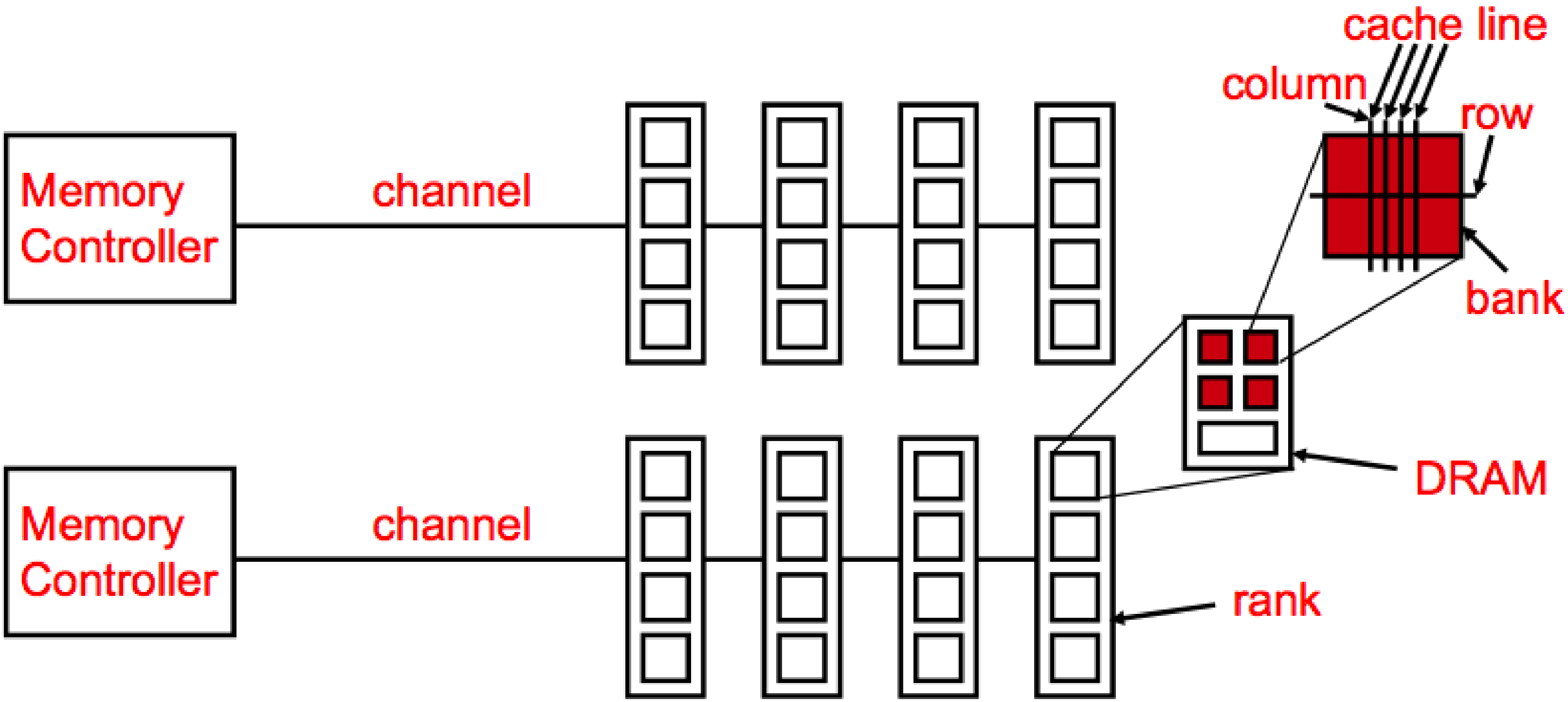
- Advantages:
  - Enables even higher capacity
- Disadvantages:
  - Interconnect complexity and energy consumption can be high

DRAM Channels

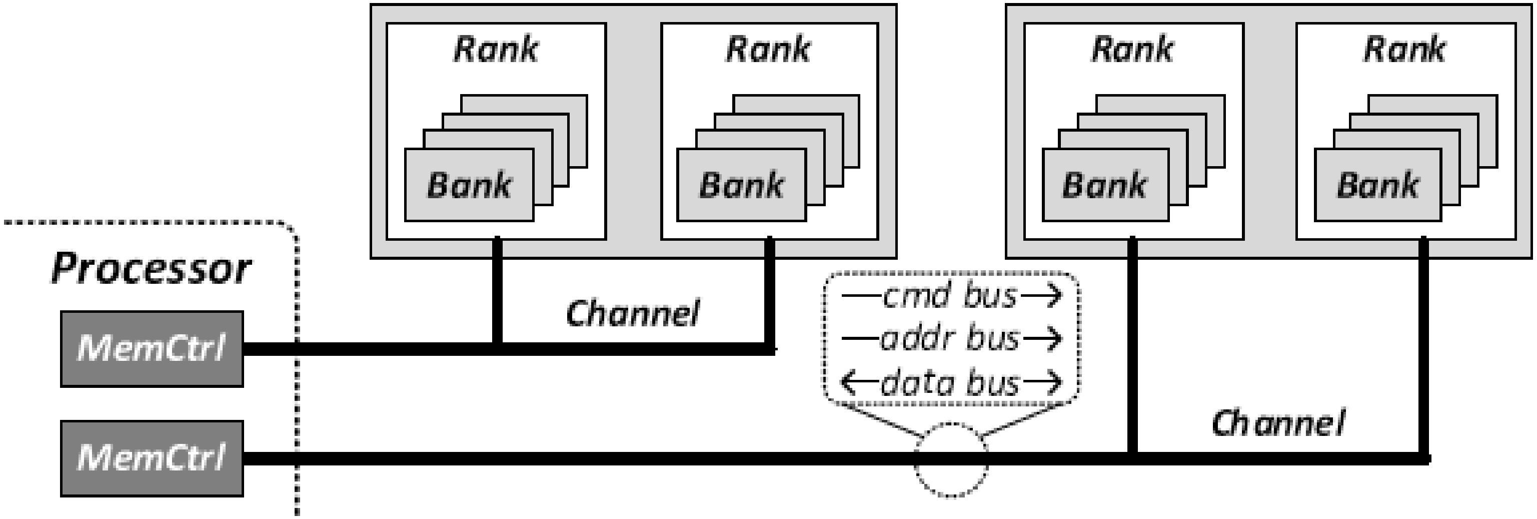


- 2 Independent Channels: 2 Memory Controllers (Above)
- 2 Dependent/Lockstep Channels: 1 Memory Controller with wide interface (Not Shown above)

Generalized Memory Structure



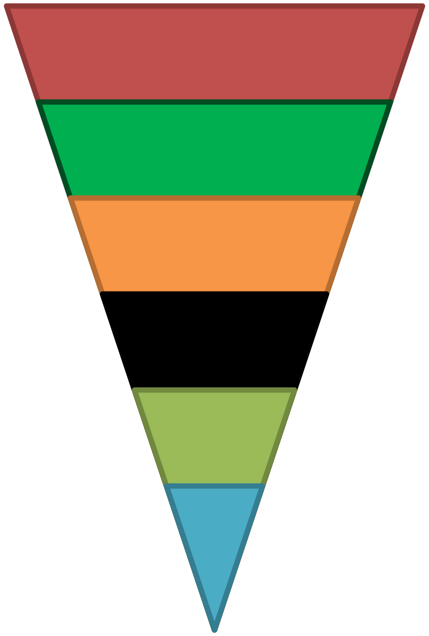
Generalized Memory Structure



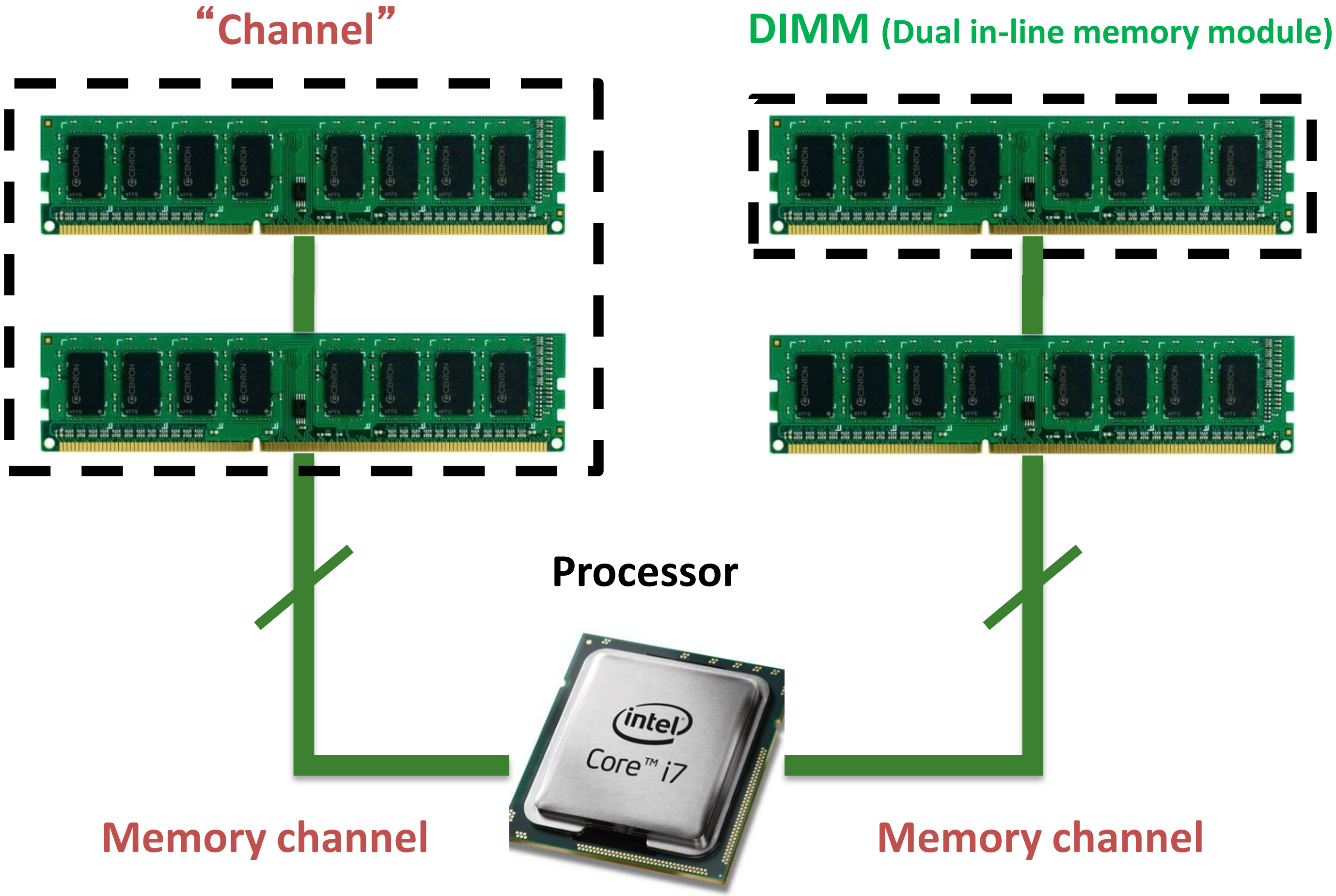
The DRAM Subsystem
The Top Down View

DRAM Subsystem Organization

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



The DRAM subsystem



“Channel”

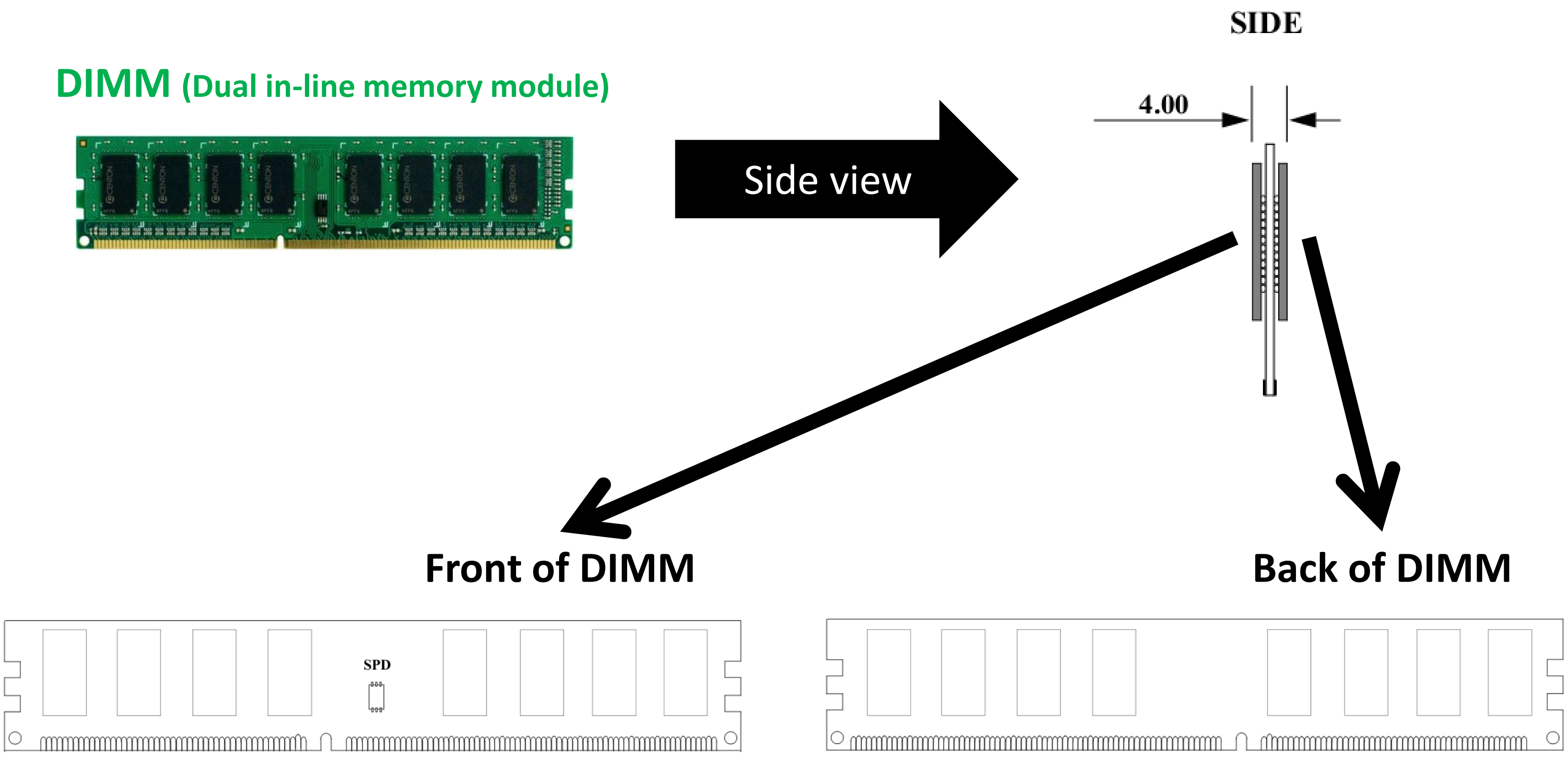
DIMM (Dual in-line memory module)

Processor

Memory channel

Memory channel

Breaking down a DIMM



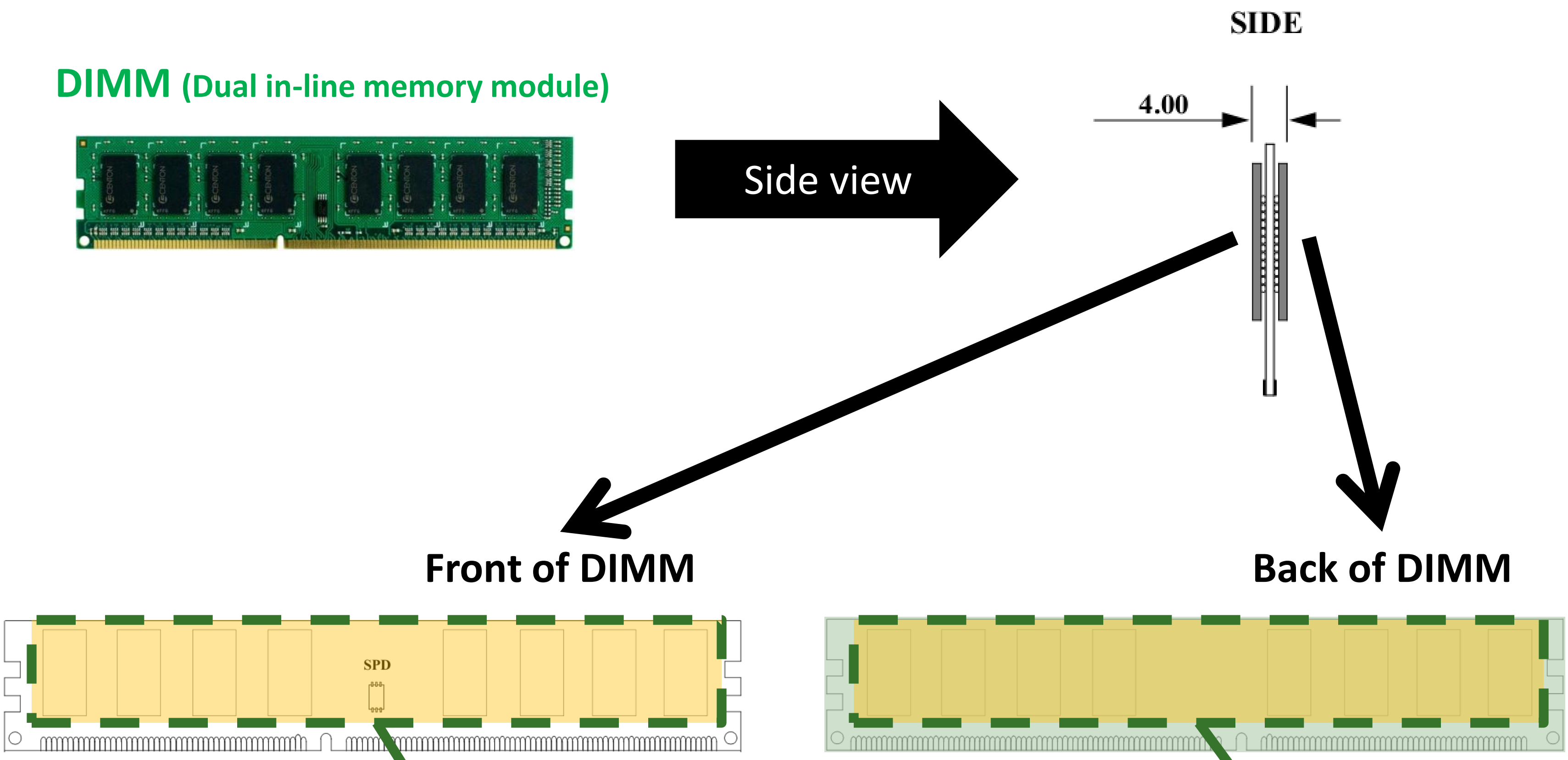
DIMM (Dual in-line memory module)

Side view

Front of DIMM

Back of DIMM

Breaking down a DIMM



DIMM **(Dual in-line memory module)**

Side view

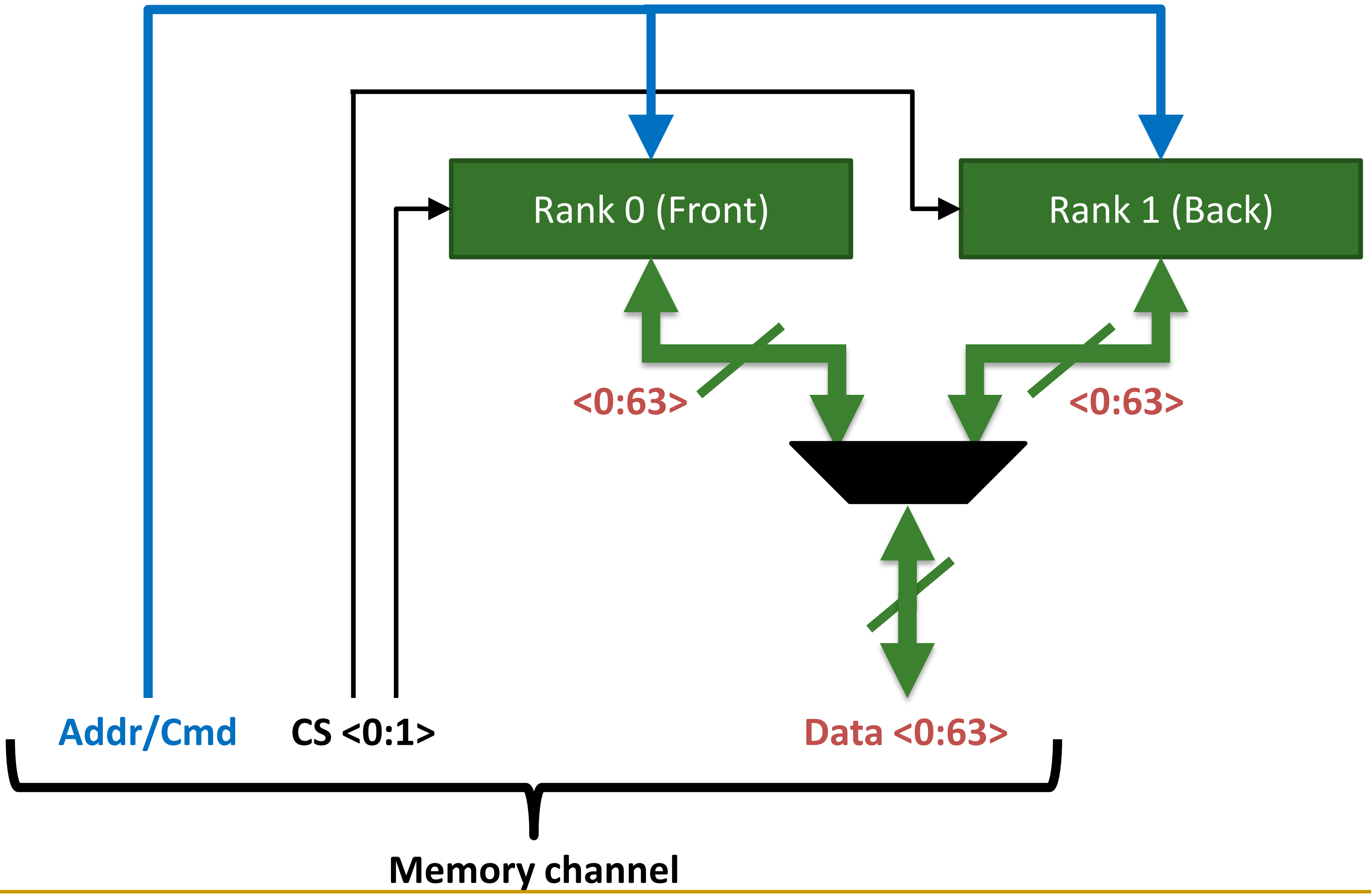
Front of DIMM

Back of DIMM

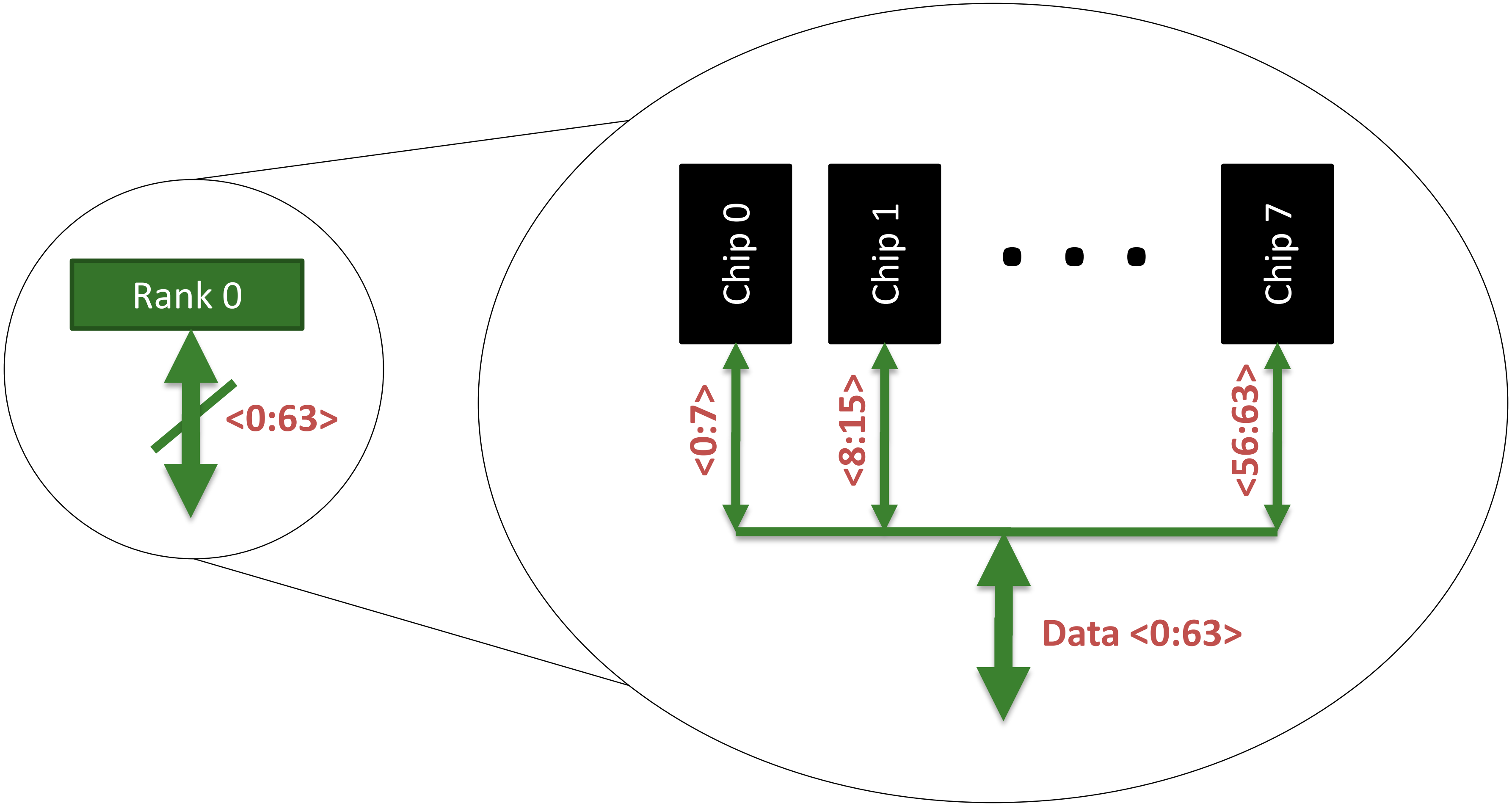
Rank 0: collection of 8 chips

Rank 1

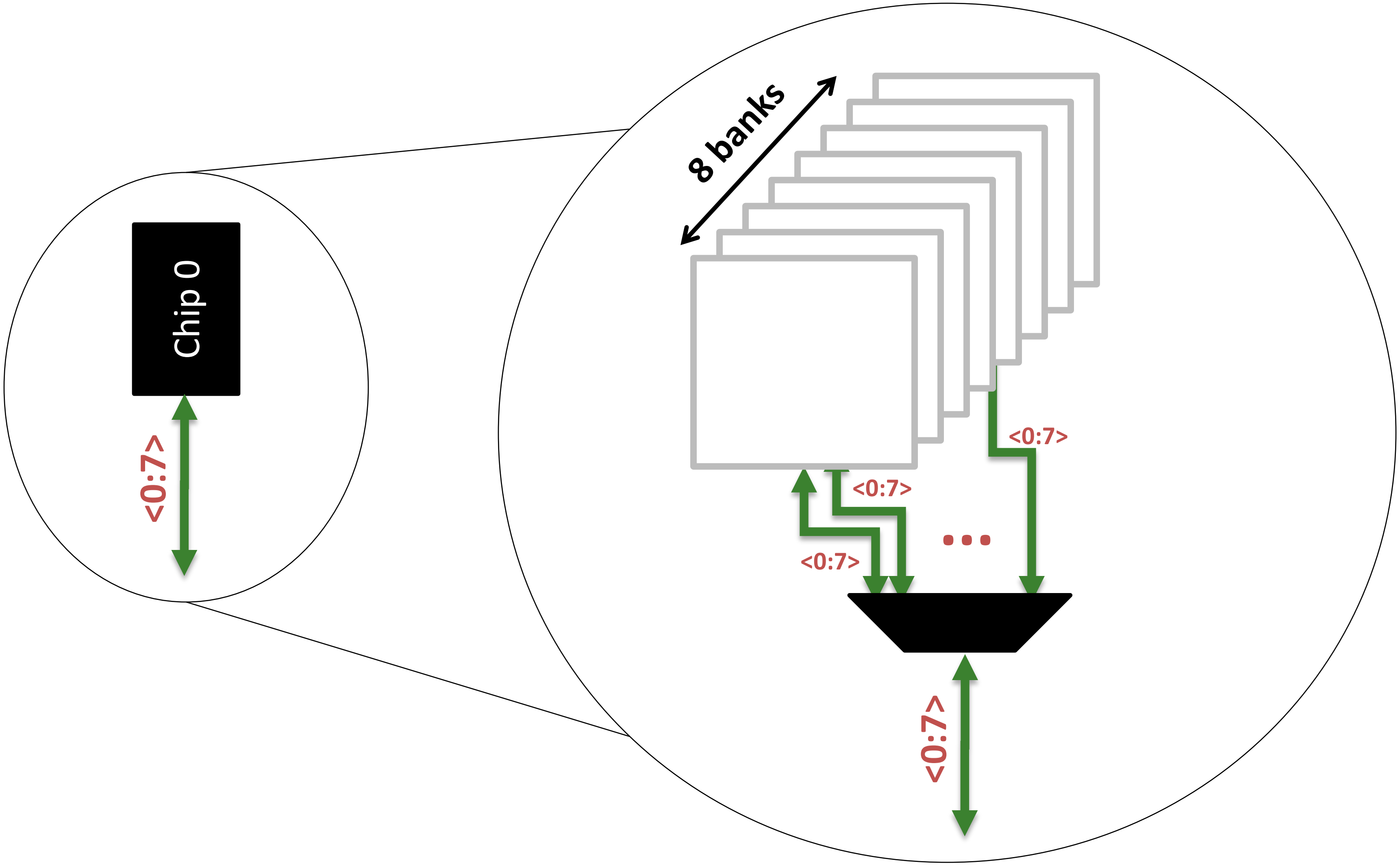
Rank


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

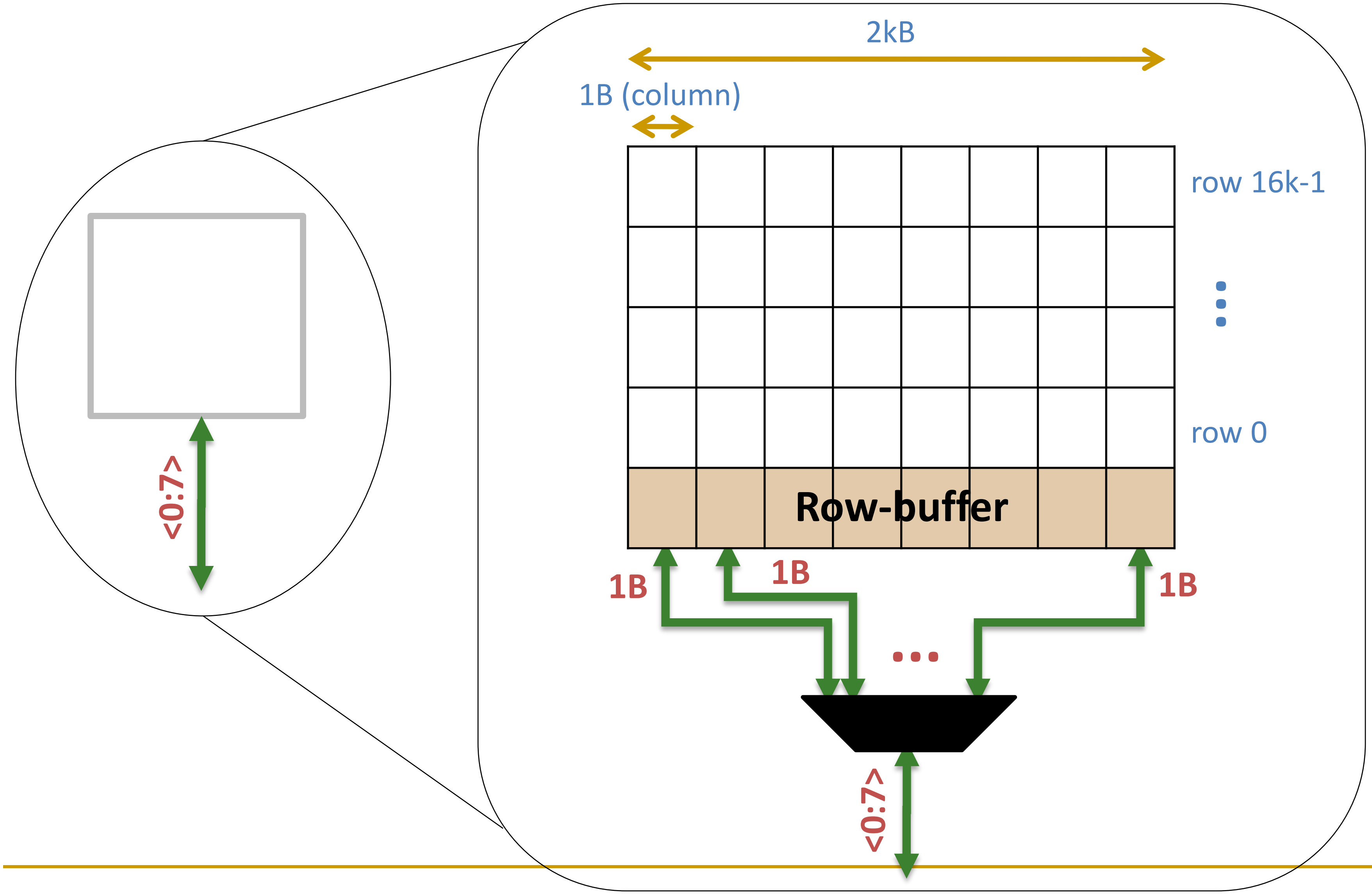
Breaking down a Rank



Breaking down a Chip

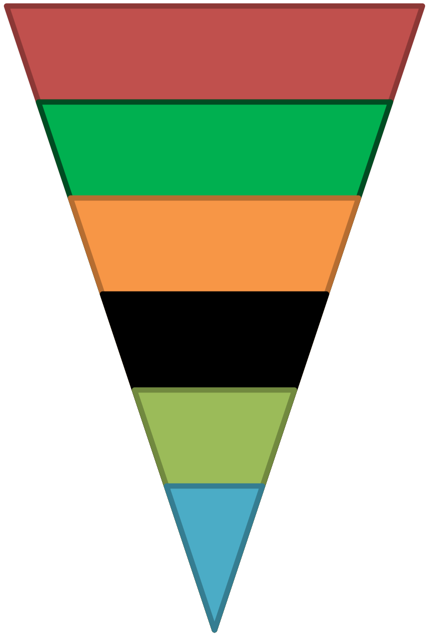


Breaking down a Bank



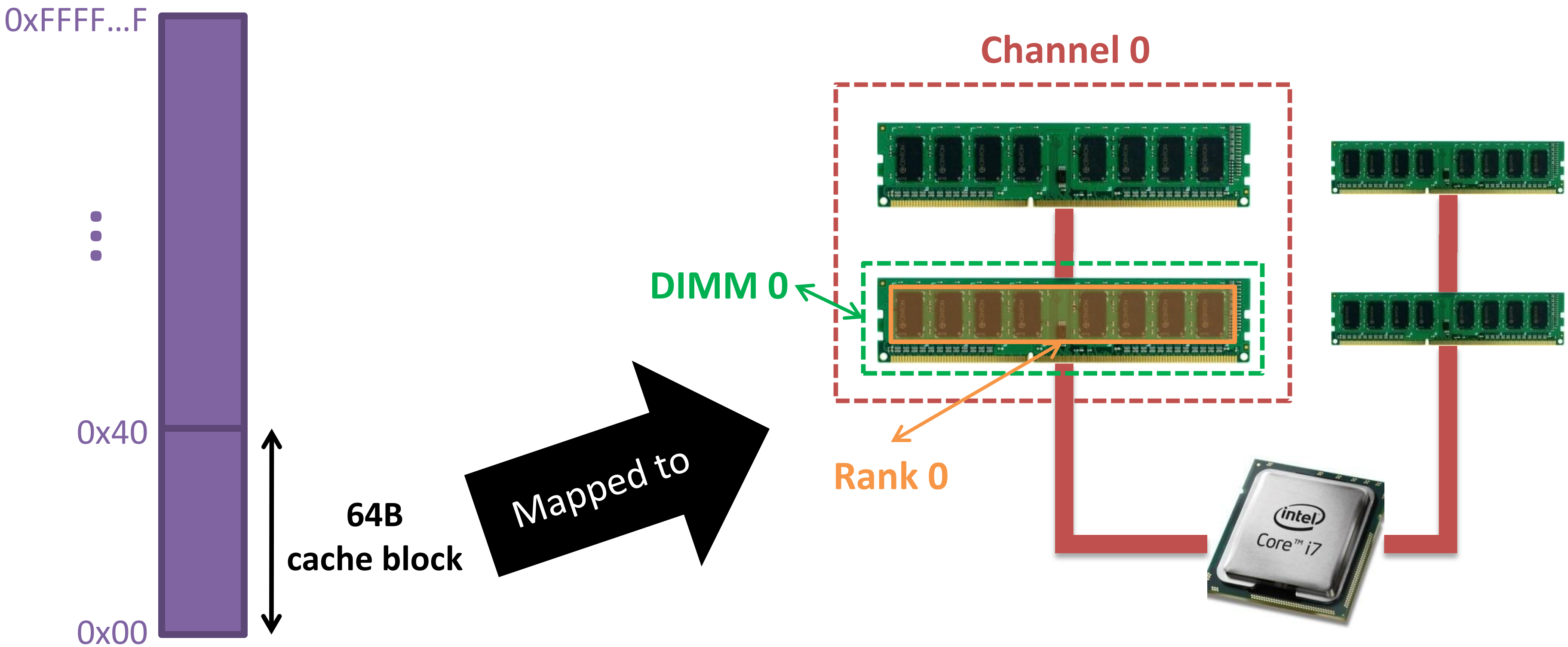
DRAM Subsystem Organization

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



Example: Transferring a cache block

Physical memory space



0xFFFF…F

0x40

0x00

64B
cache block

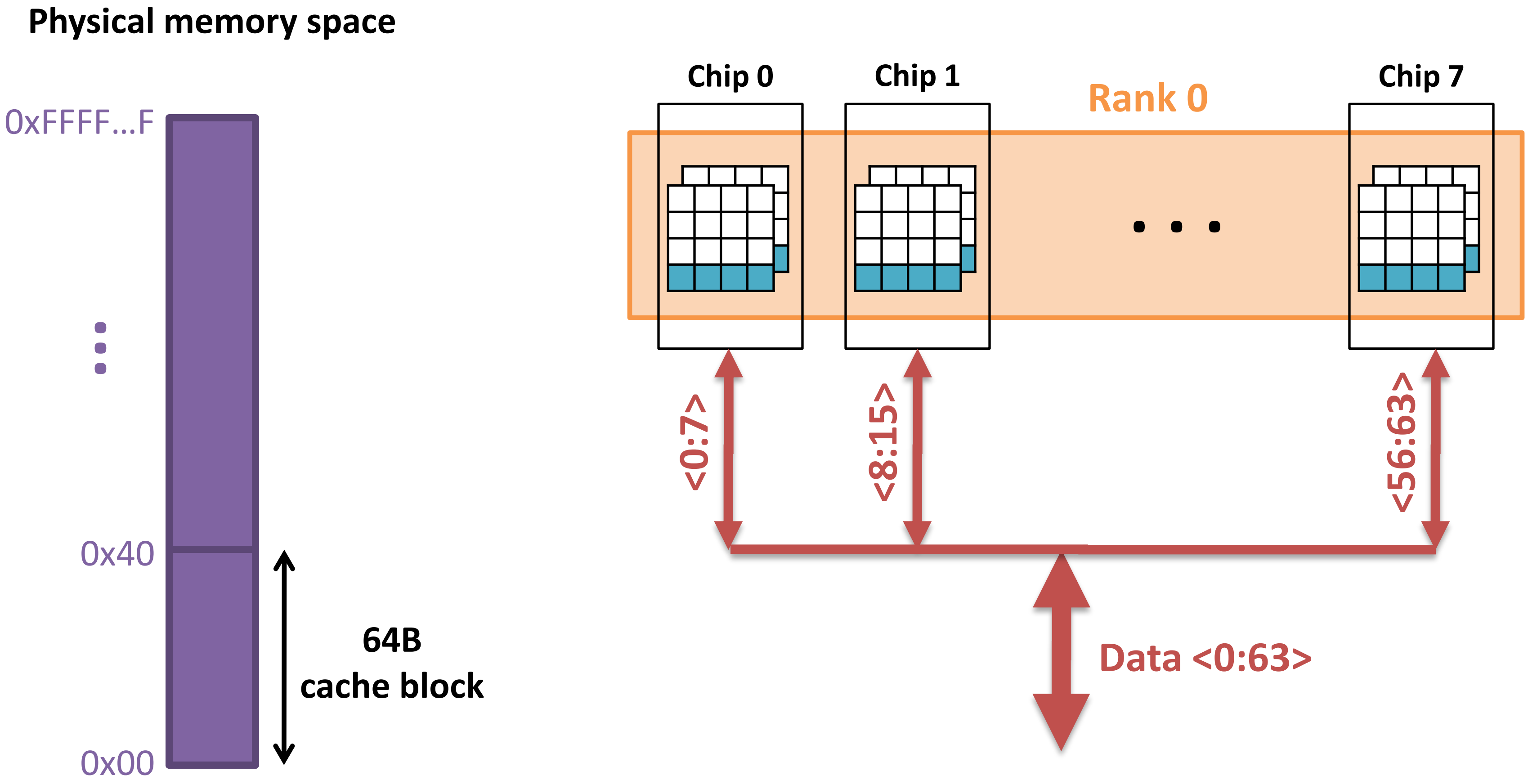
Mapped to

Channel 0

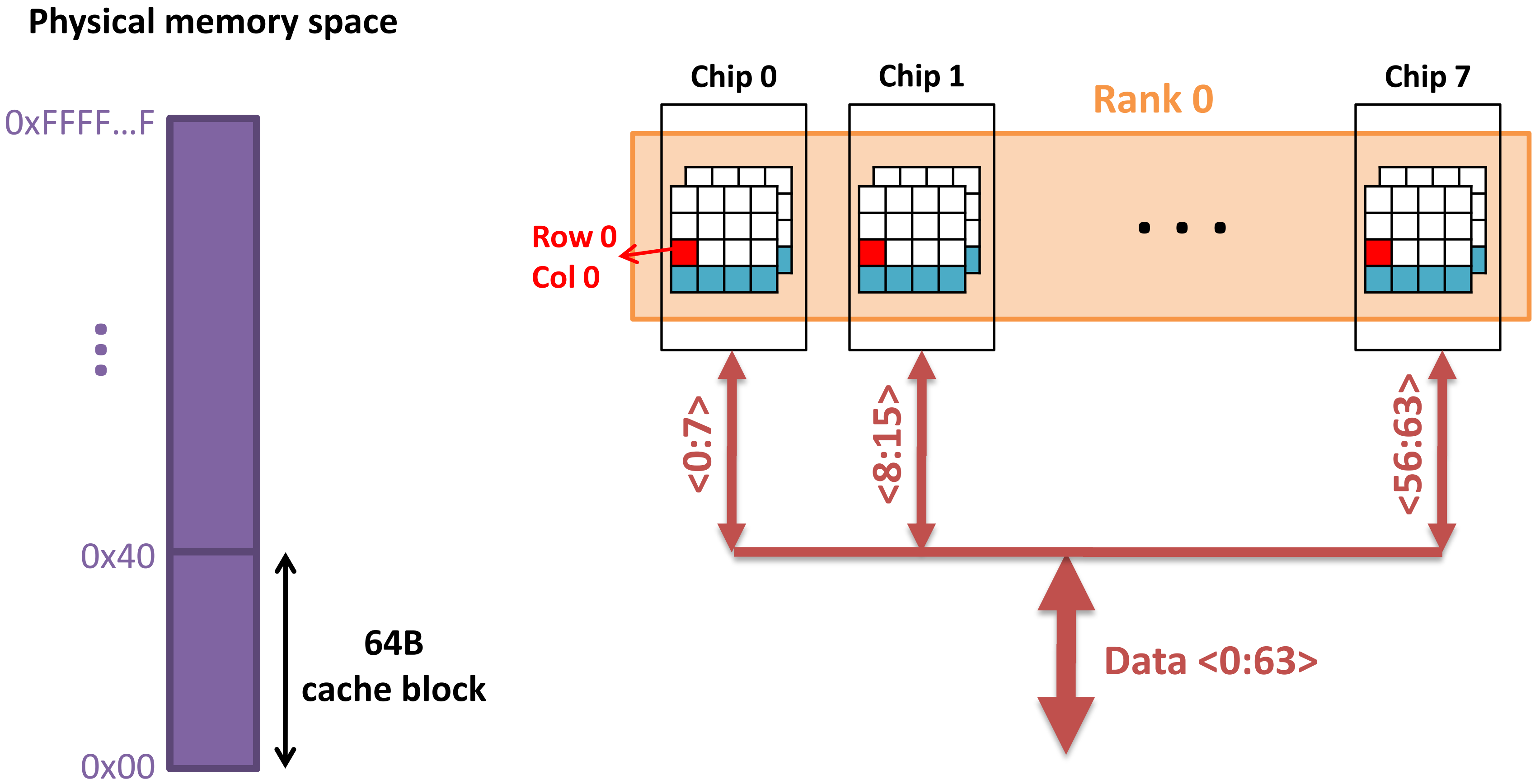
DIMM 0

Rank 0

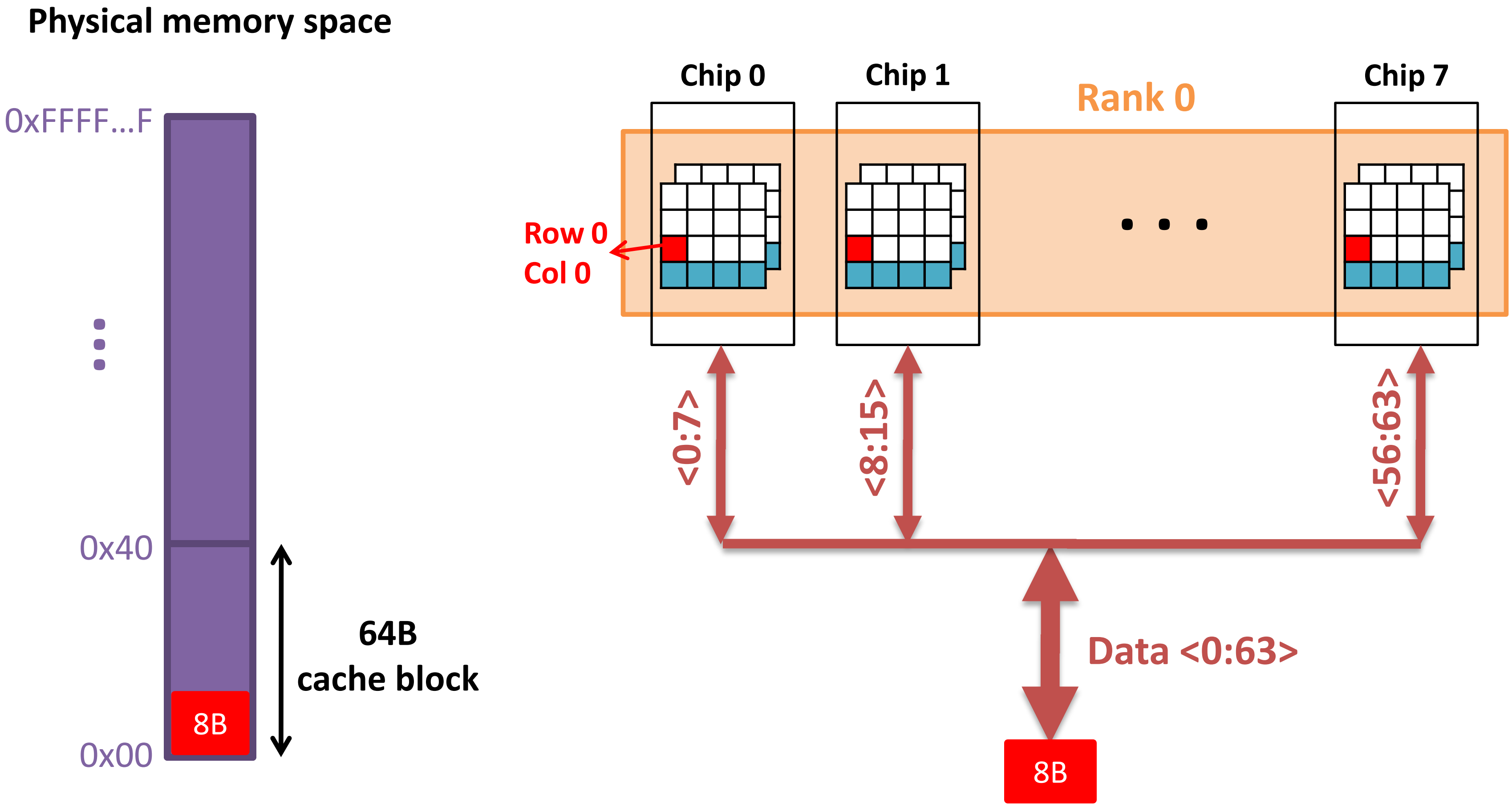
Example: Transferring a cache block



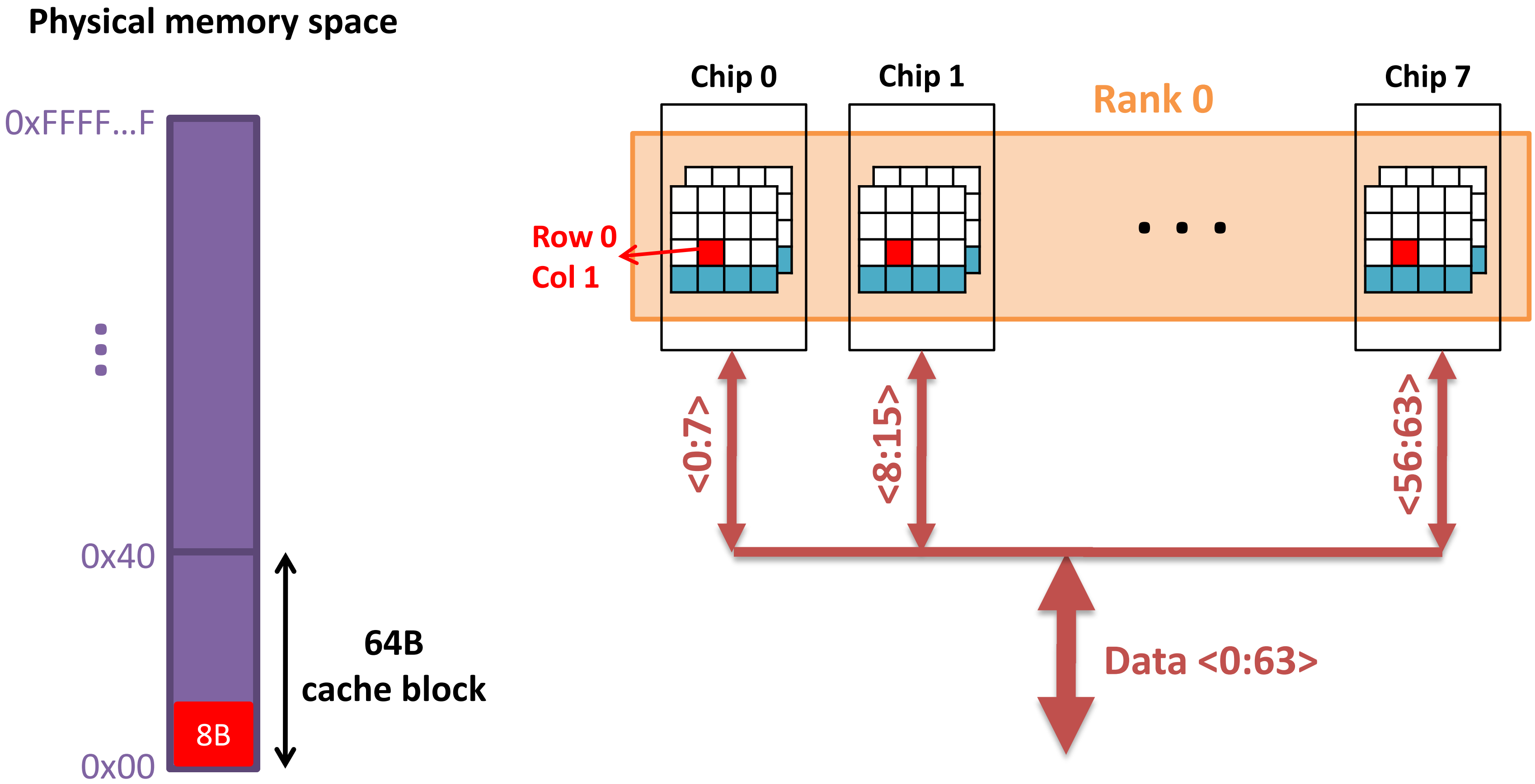
Example: Transferring a cache block



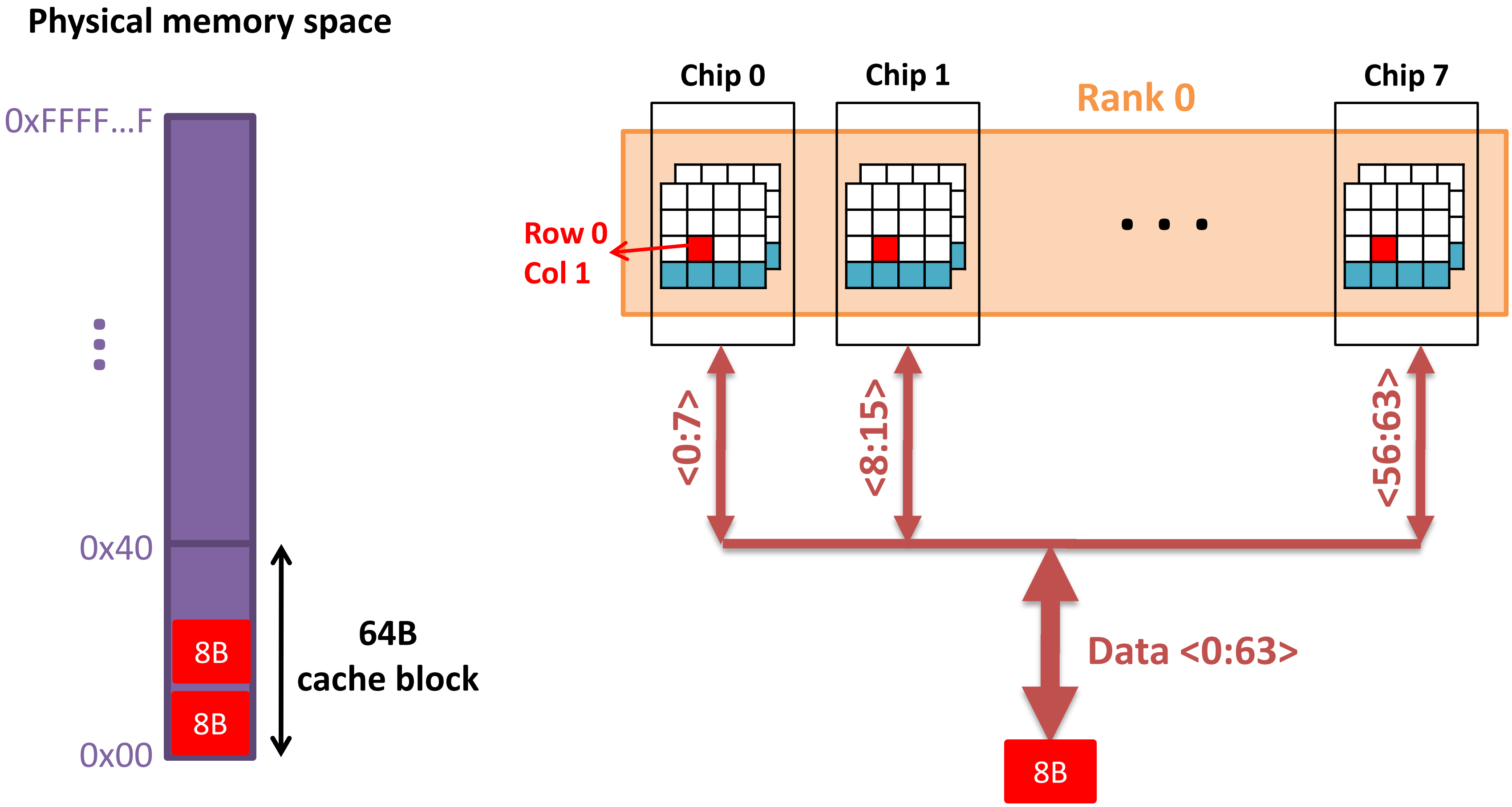
Example: Transferring a cache block



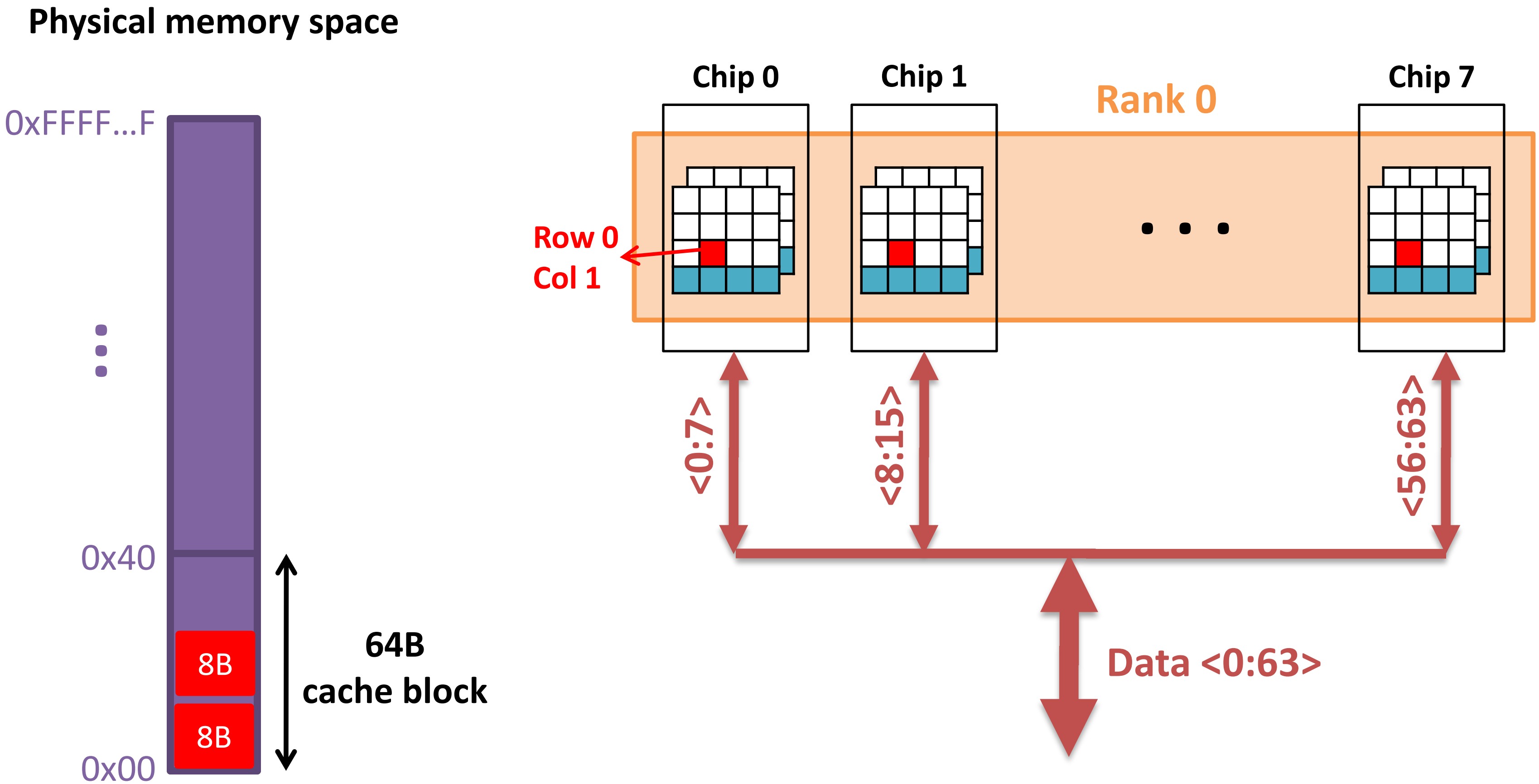
Example: Transferring a cache block



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

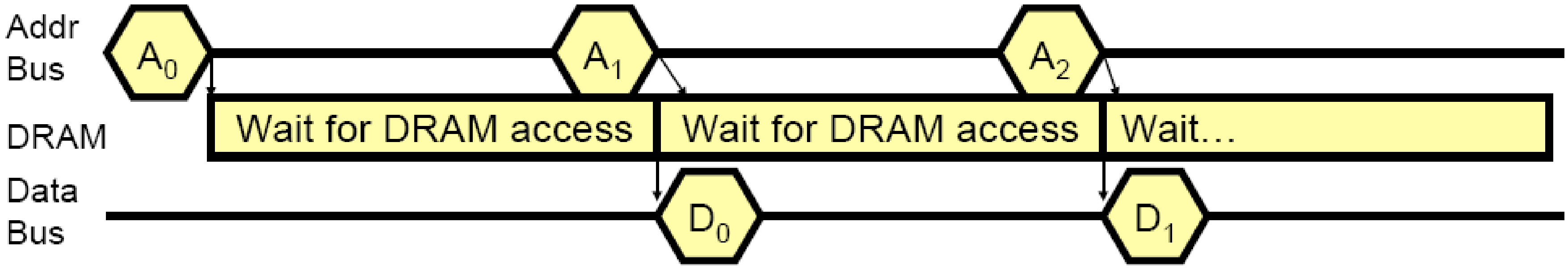
- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS (column address strobe) if row is “open” OR
  - RAS (row address strobe) + CAS if array precharged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

Multiple Banks (Interleaving) and Channels

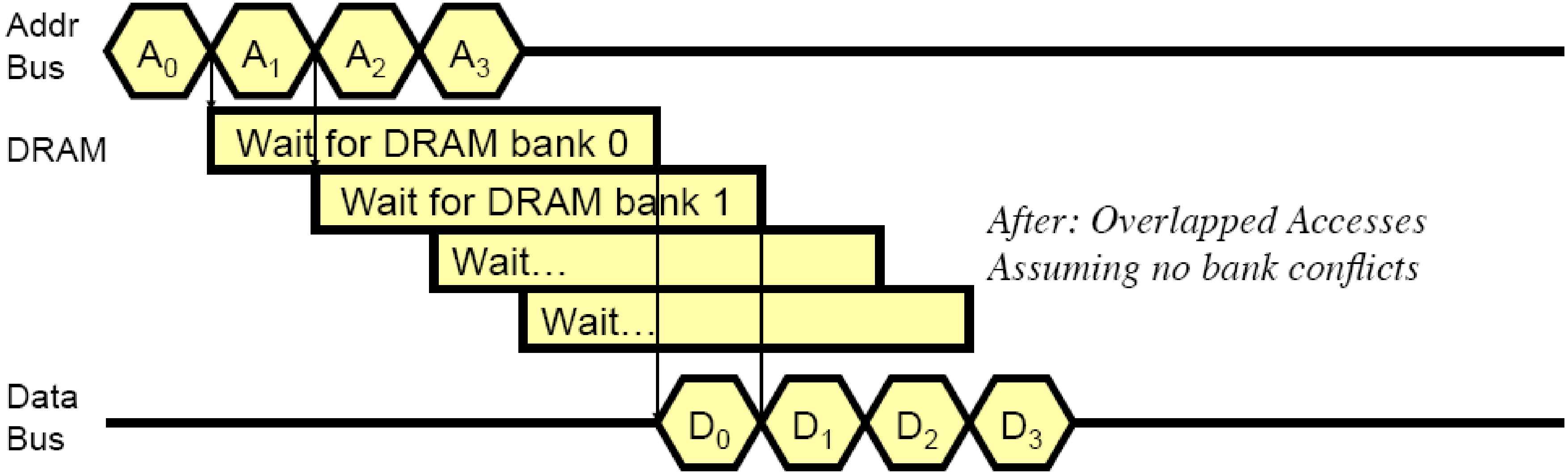
- Multiple banks
  - Enable concurrent DRAM accesses
  - Bits in address determine which bank an address resides in
- Multiple independent channels serve the same purpose
  - But they are even better because they have separate data buses
  - Increased bus bandwidth

- 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

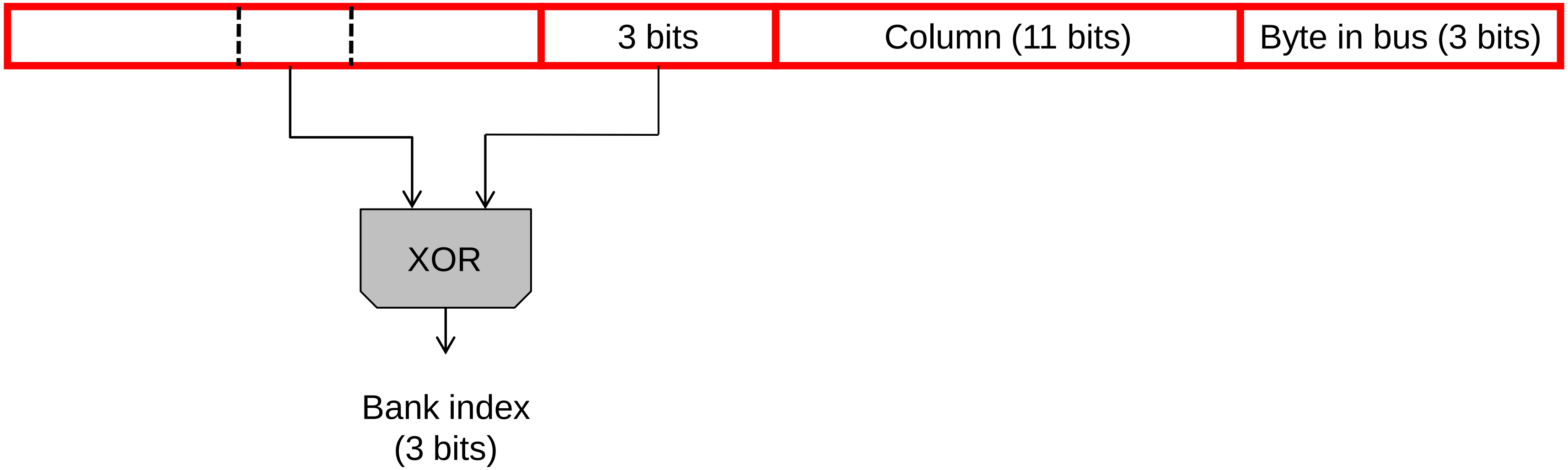
- Cache block interleaving
  - Consecutive cache block addresses in consecutive banks
  - 64 byte cache blocks

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	

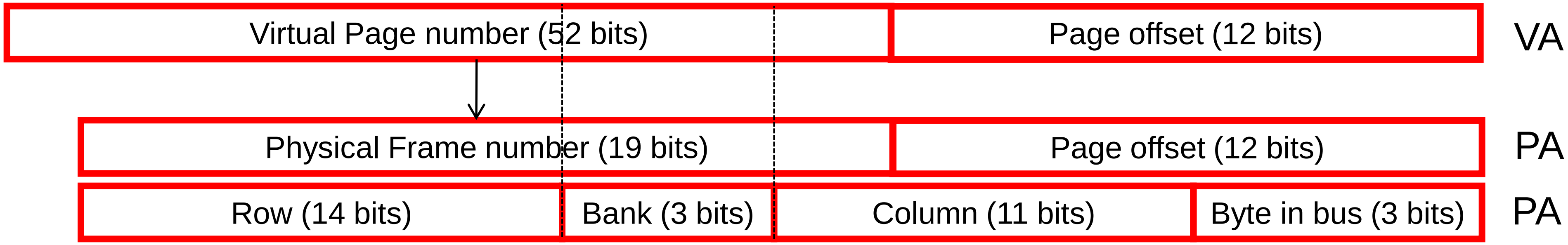
Row (14 bits)	High Column	C	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
	8 bits			3 bits	

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]**

Memory Controllers

DRAM versus Other Types of Memories

- Long latency memories have similar characteristics that need to be controlled.

- The following discussion will use DRAM as an example, but many scheduling and control issues are similar in the design of controllers for other types of memories
  - Flash memory
  - Other emerging memory technologies
    - Phase Change Memory
    - Spin-Transfer Torque Magnetic Memory
  - These other technologies can place other demands on the controller

DRAM Controller: Functions

- Ensure correct operation of DRAM (refresh and timing)

- Service DRAM requests while obeying timing constraints of DRAM chips
  - Constraints: resource conflicts (bank, bus, channel), minimum write-to-read delays
  - Translate requests to DRAM command sequences

- Buffer and schedule requests to improve performance
  - Reordering, row-buffer, bank, rank, bus management

- Manage power consumption and thermals in DRAM
  - Turn on/off DRAM chips, manage power modes

DRAM Controller: Where to Place

- In chipset

  + More flexibility to plug different DRAM types into the system

  + Less power density in the CPU chip

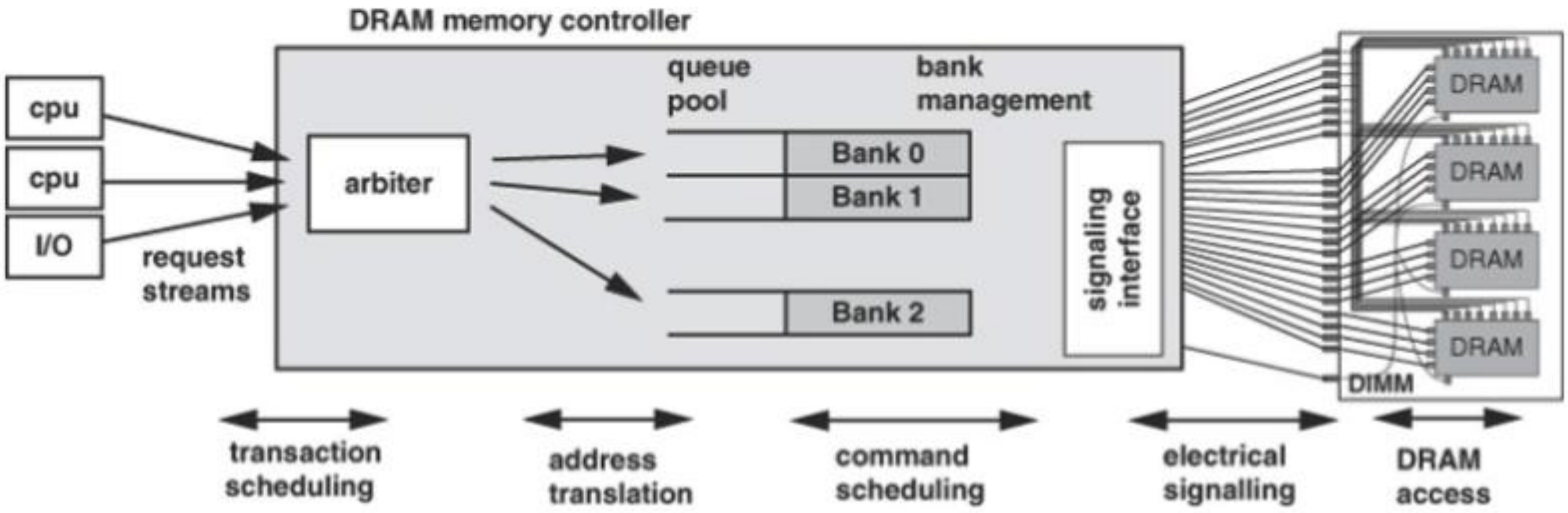
- On CPU chip

  + Reduced latency for main memory access

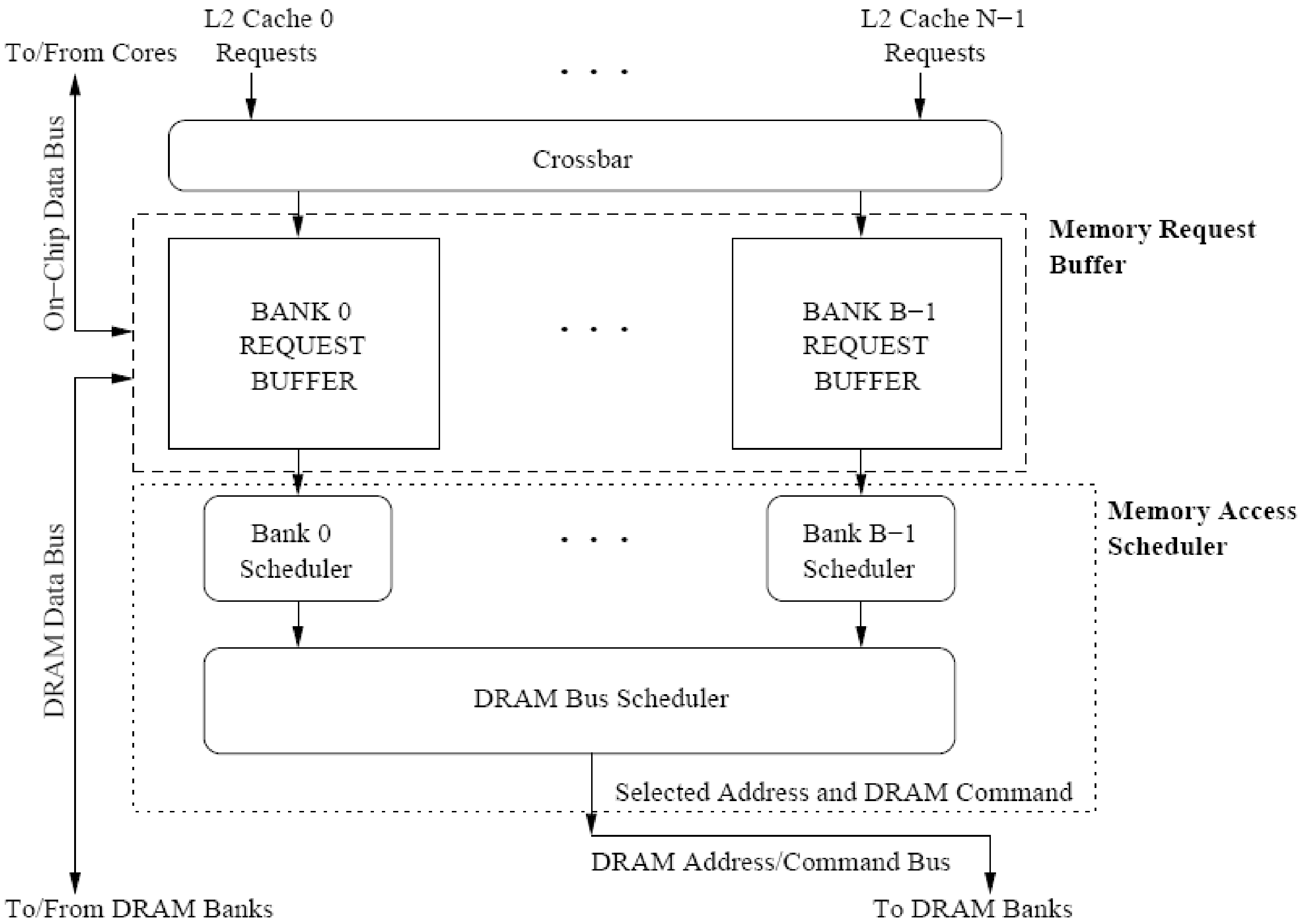
  + Higher bandwidth between cores and controller

  - More information can be communicated (e.g. request's importance in the processing core)

A Modern DRAM Controller (I)



A Modern DRAM Controller (II)



DRAM Scheduling Policies (I)

- FCFS (first come first served)
  - Oldest request first

- FR-FCFS (first ready, first come first served)

  1. Row-hit first
  2. Oldest first

  Goal: Maximize row buffer hit rate → maximize DRAM throughput

  - Actually, scheduling is done at the command level
    - Column commands (read/write) prioritized over row commands (activate/precharge)
    - Within each group, older commands prioritized over younger ones

DRAM Scheduling Policies (II)

- A scheduling policy is essentially a prioritization order

- Prioritization can be based on
  - Request age
  - Row buffer hit/miss status
  - Request type (prefetch, read, write)
  - Requestor type (load miss or store miss)
  - Request criticality
    - Oldest miss in the core?
    - How many instructions in core are dependent on it?