18-447: Computer Architecture
Lecture 25: Main Memory

Prof. Onur Mutlu
Carnegie Mellon University
Spring 2013, 4/3/2013

Reminder: Homework 5 (Today)

- Due April 3 (Wednesday!)
- Topics: Vector processing, VLIW, Virtual memory, Caching

Reminder: Lab Assignment 5 (Friday)

- Lab Assignment 5
  - Due Friday, April 5
  - Modeling caches and branch prediction at the microarchitectural level (cycle level) in C
  - Extra credit: Cache design optimization
    - Size, block size, associativity
    - Replacement and insertion policies
    - Cache indexing policies
    - Anything else you would like

Heads Up: Midterm II in Two Weeks

- April 17

- Similar format as Midterm I

Last Lecture

- Wrap up virtual memory – cache interaction
  - Virtually-indexed physically-tagged caches
  - Solutions to the synonym problem

- Improving cache (and memory hierarchy) performance
  - Cheaper alternatives to more associativity
  - Blocking and code reorganization
  - Memory-level-parallelism (MLP) aware cache replacement

- Enabling multiple accesses in parallel

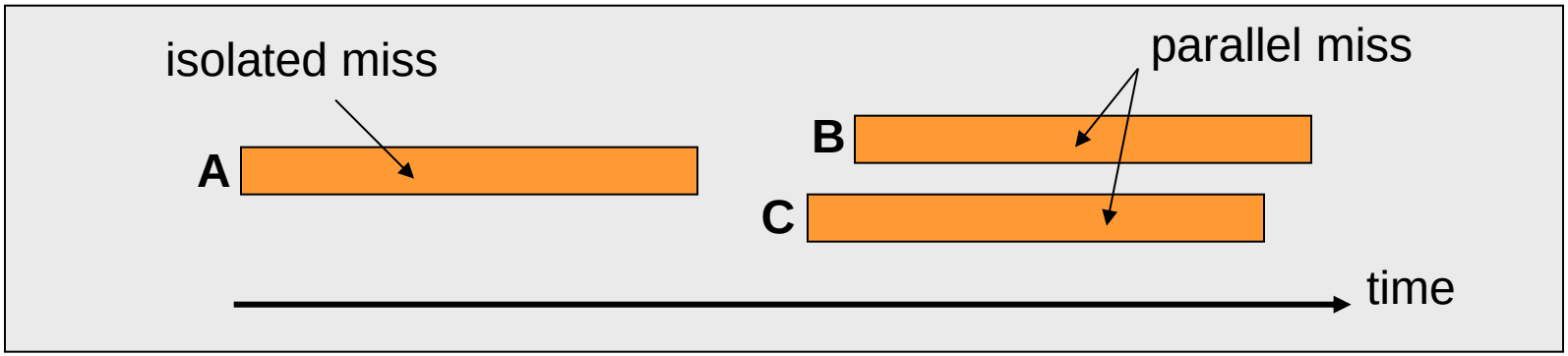
Today

- Enabling multiple accesses in parallel
- Main memory

Improving Basic Cache Performance

- Reducing miss rate
  - More associativity
  - Alternatives/enhancements to associativity
    - Victim caches, hashing, pseudo-associativity, skewed associativity
  - Better replacement/insertion policies
  - Software approaches
- Reducing miss latency/cost
  - Multi-level caches
  - Critical word first
  - Subblocking/sectoring
  - Better replacement/insertion policies
  - Non-blocking caches (multiple cache misses in parallel)
  - Multiple accesses per cycle
  - Software approaches

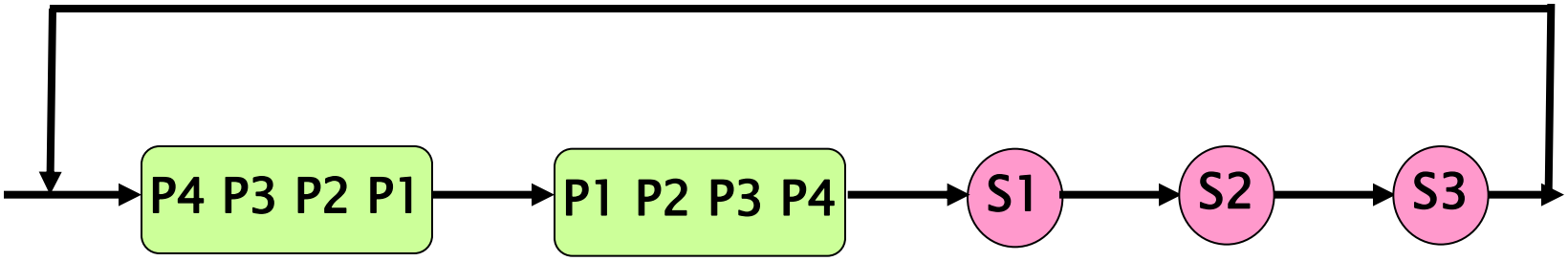
Review: Memory Level Parallelism (MLP)

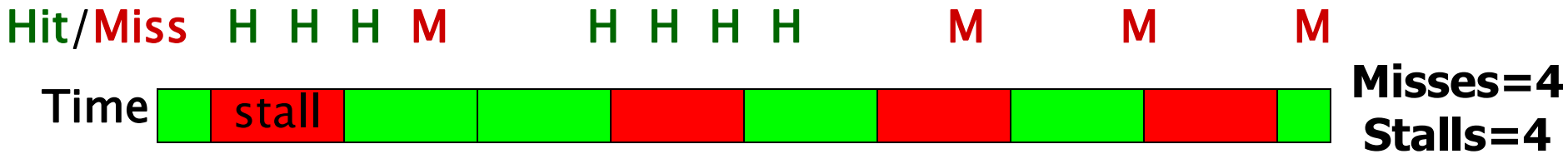


- Memory Level Parallelism (MLP) means generating and servicing multiple memory accesses in parallel [Glew' 98]
- Several techniques to improve MLP (e.g., out-of-order execution)
- MLP varies. Some misses are isolated and some parallel

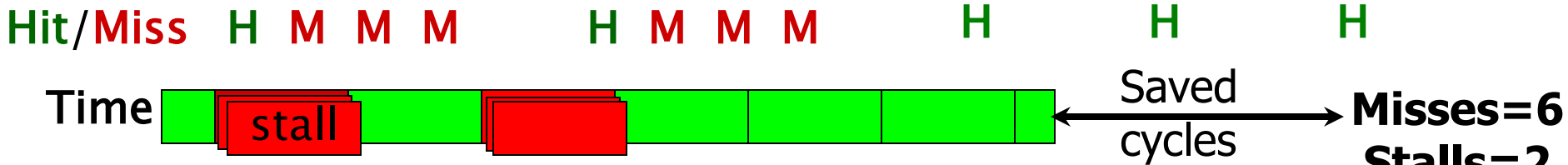
How does this affect cache replacement?

Review: Fewest Misses = Best Performance





Belady's OPT replacement



MLP-Aware replacement

Reading: MLP-Aware Cache Replacement

- How do we incorporate MLP into replacement decisions?

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

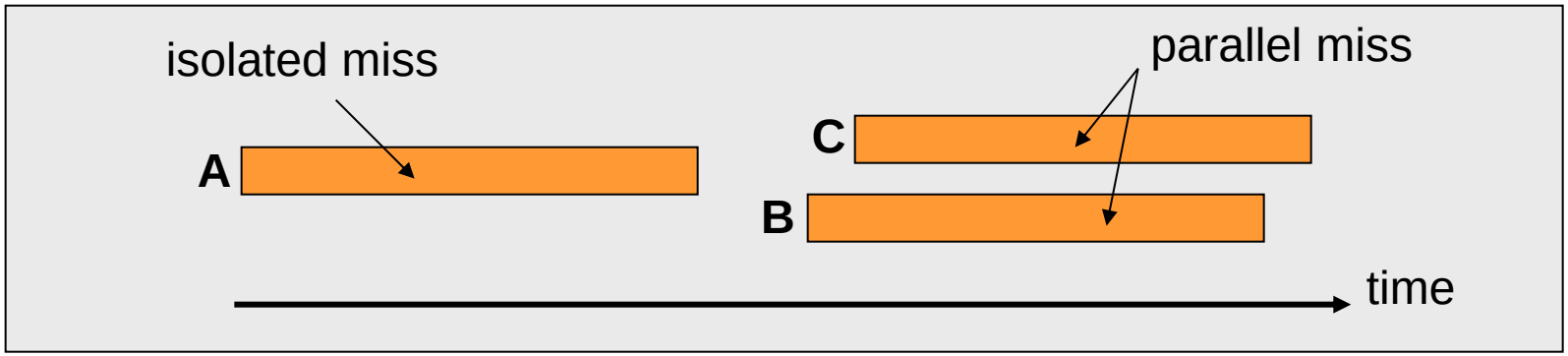
Enabling Multiple Outstanding Misses

Handling Multiple Outstanding Accesses

- Non-blocking or lockup-free caches
  - Kroft, "Lockup-Free Instruction Fetch/Prefetch Cache Organization," ISCA 1981.
- Question: If the processor can generate multiple cache accesses, can the later accesses be handled while a previous miss is outstanding?
- Idea: Keep track of the status/data of misses that are being handled in Miss Status Handling Registers (MSHRs)
  - A cache access checks MSHRs to see if a miss to the same block is already *pending.*
    - If pending, a new request is not generated
    - If pending and the needed data available, data forwarded to later load
  - Requires buffering of outstanding miss requests

Non-Blocking Caches (and MLP)

- Enable cache access when there is a pending miss
- Enable multiple misses in parallel
  - Memory-level parallelism (MLP)
    - generating and servicing multiple memory accesses in parallel
  - Why generate multiple misses?

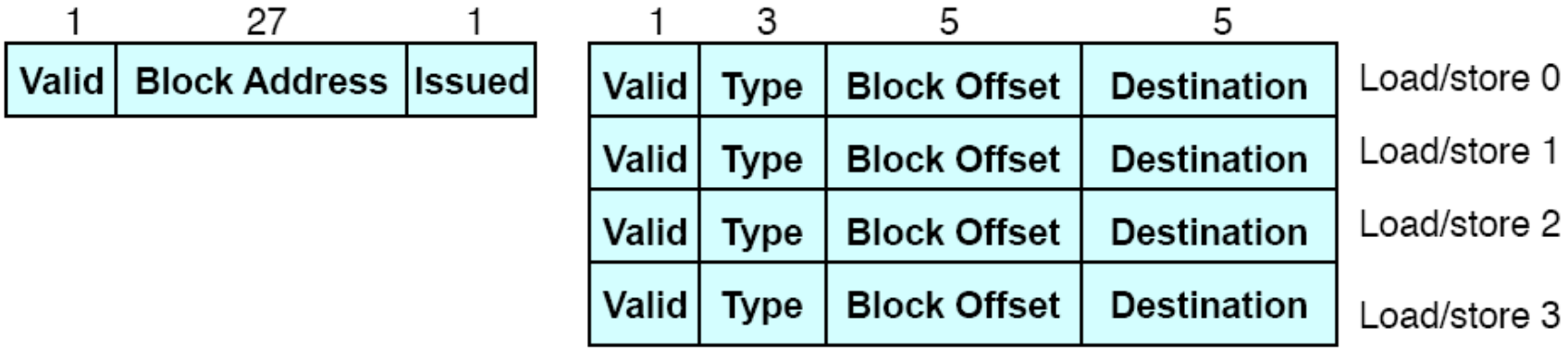


    - Enables latency tolerance: overlaps latency of different misses
  - How to generate multiple misses?
    - Out-of-order execution, multithreading, runahead, prefetching

Miss Status Handling Register

- Also called “miss buffer”
- Keeps track of
  - Outstanding cache misses
  - Pending load/store accesses that refer to the missing cache block
- Fields of a single MSHR entry
  - Valid bit
  - Cache block address (to match incoming accesses)
  - Control/status bits (prefetch, issued to memory, which subblocks have arrived, etc)
  - Data for each subblock
  - For each pending load/store
    - Valid, type, data size, byte in block, destination register or store buffer entry address

Miss Status Handling Register Entry



MSHR Operation

- On a cache miss:
  - Search MSHRs for a pending access to the same block
    - Found: Allocate a load/store entry in the same MSHR entry
    - Not found: Allocate a new MSHR
    - No free entry: stall
- When a subblock returns from the next level in memory
  - Check which loads/stores waiting for it
    - Forward data to the load/store unit
    - Deallocate load/store entry in the MSHR entry
  - Write subblock in cache or MSHR
  - If last subblock, dellaocate MSHR (after writing the block in cache)

Non-Blocking Cache Implementation

- When to access the MSHRs?
  - In parallel with the cache?
  - After cache access is complete?

- MSHRs need not be on the critical path of hit requests
  - Which one below is the common case?
    - Cache miss, MSHR hit
    - Cache hit

Enabling High Bandwidth Caches (and Memories in General)

Multiple Instructions per Cycle

- Can generate multiple cache accesses per cycle
- How do we ensure the cache 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)

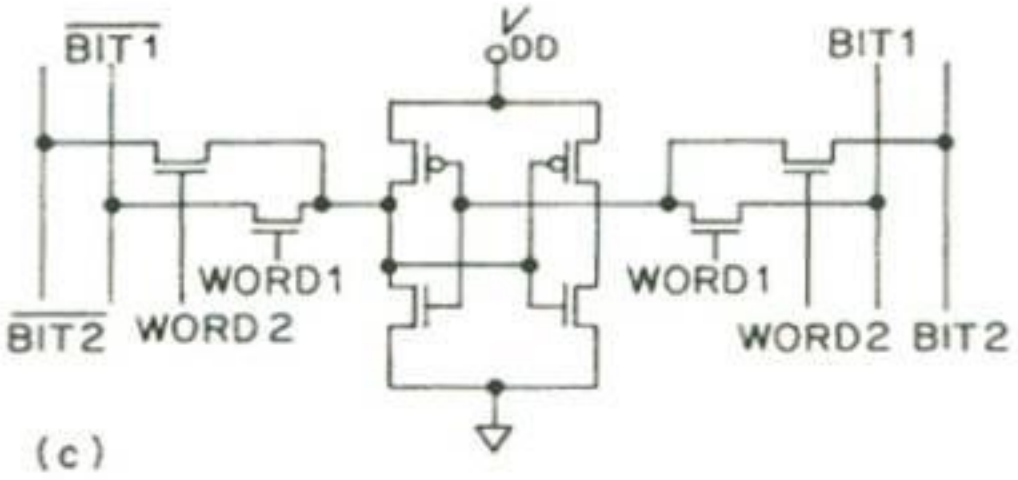
Handling Multiple Accesses per Cycle (I)

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

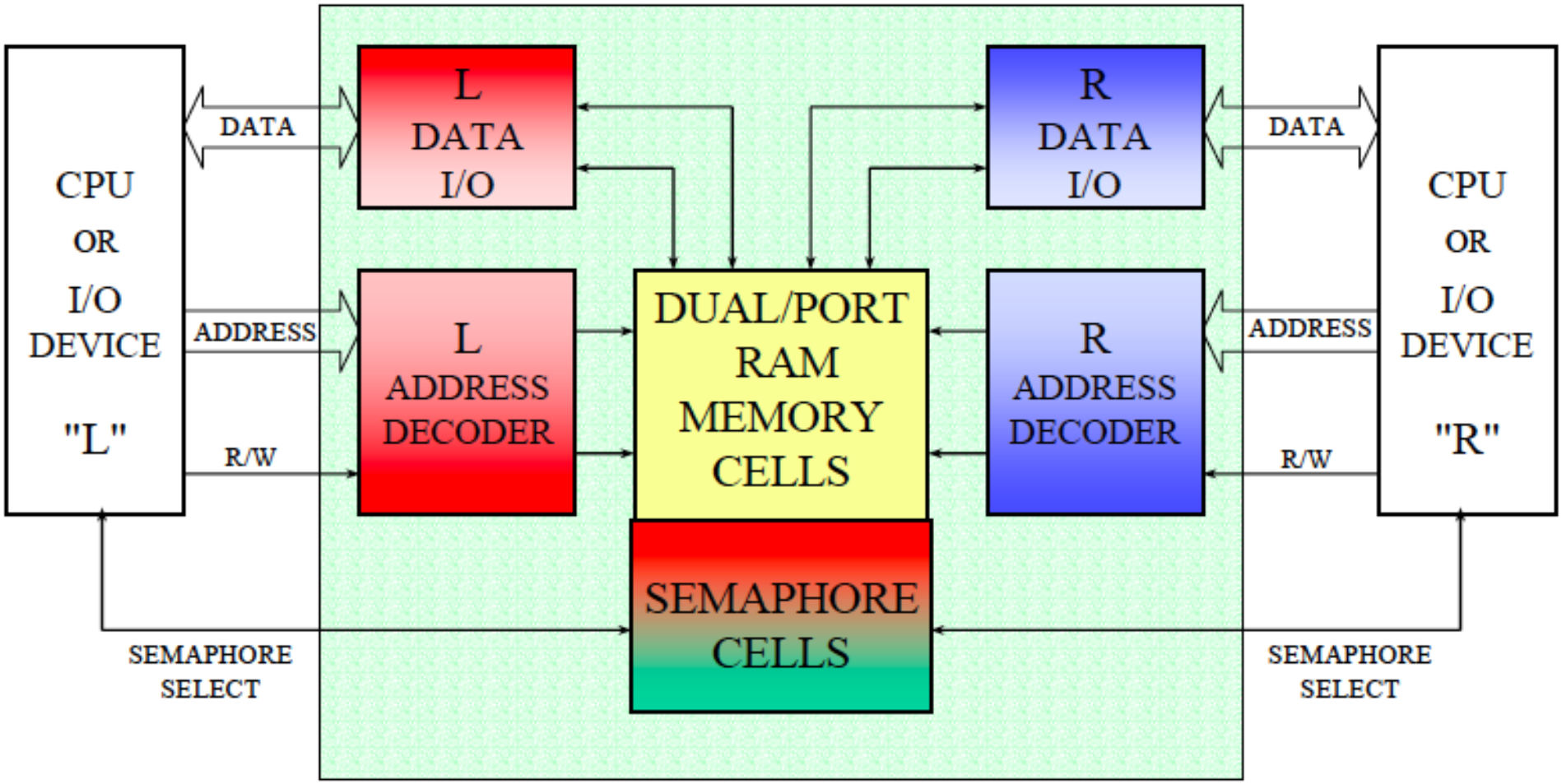
+ Truly concurrent accesses (no conflicts regardless of address)

-- Expensive in terms of latency, power, area

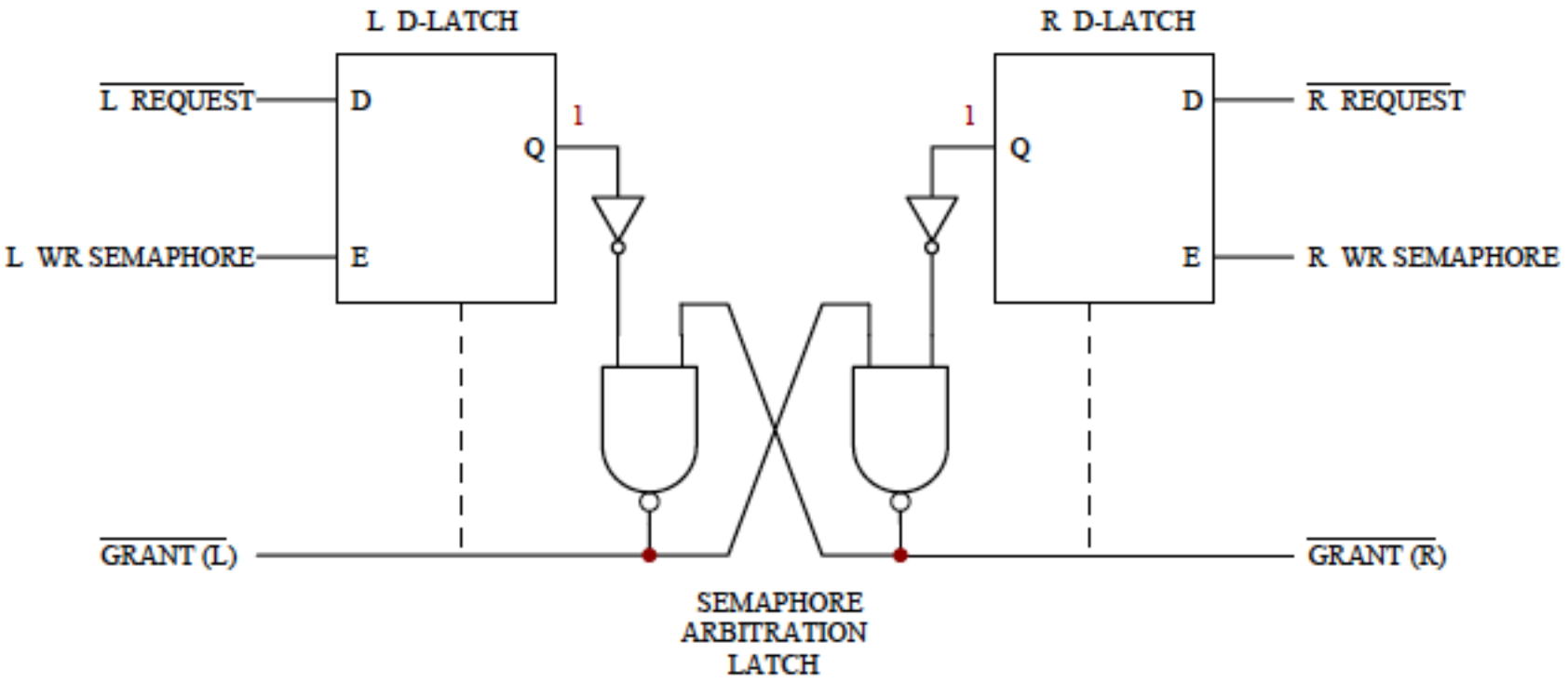
  - What about read and write to the same location at the same time?
    - Peripheral logic needs to handle this



Peripheral Logic for True Multiporting



Peripheral Logic for True Multiporting

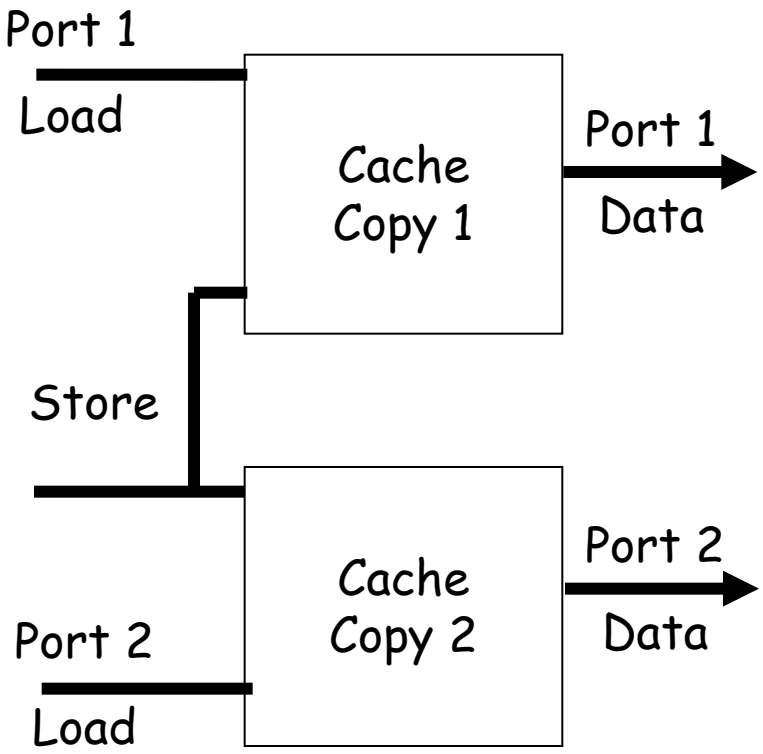


Handling Multiple Accesses per Cycle (I)

- 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 (II)

- 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”



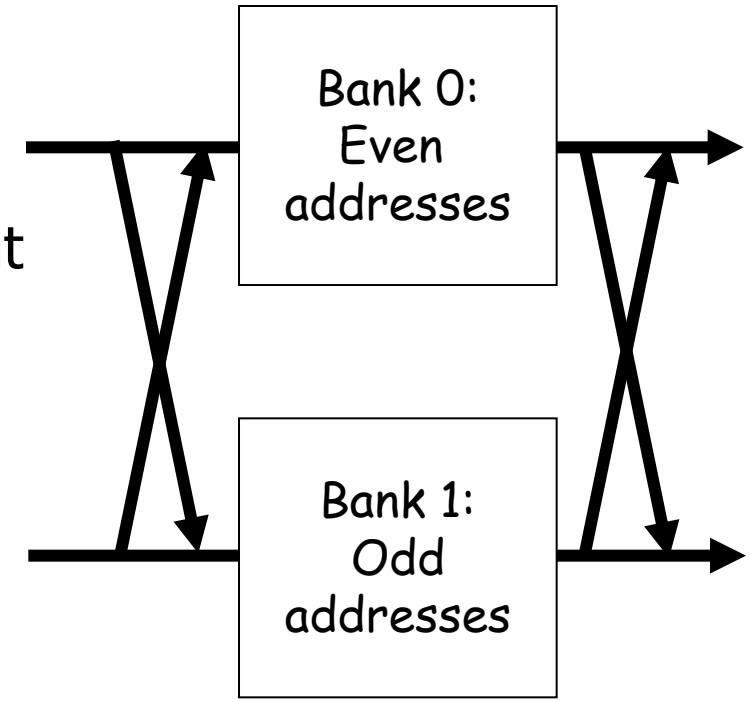
Handling Multiple Accesses per Cycle (III)

- 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



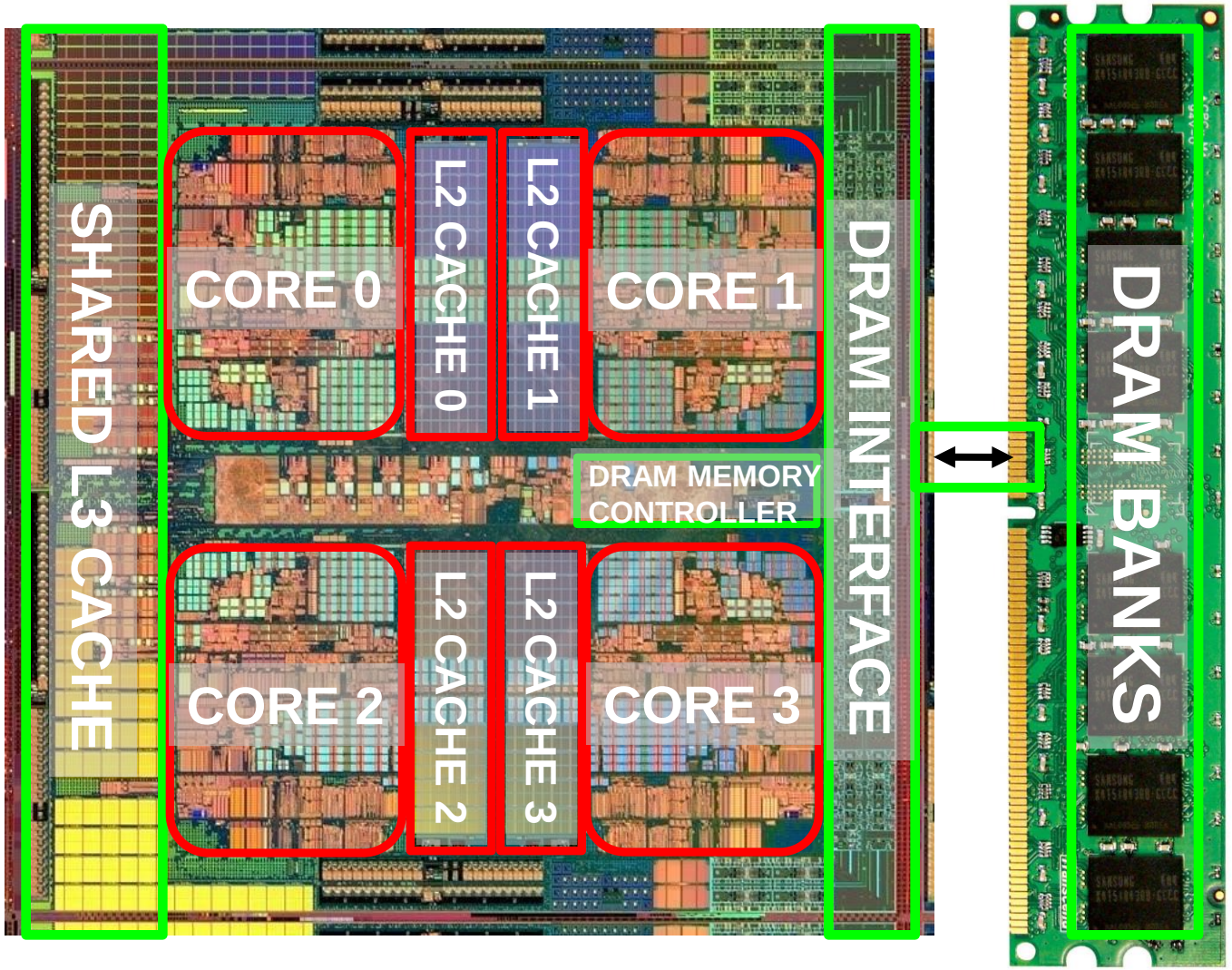
- Bank conflicts
  - Two accesses are to the same bank
  - How can these be reduced?
    - Hardware? Software?

General Principle: Interleaving

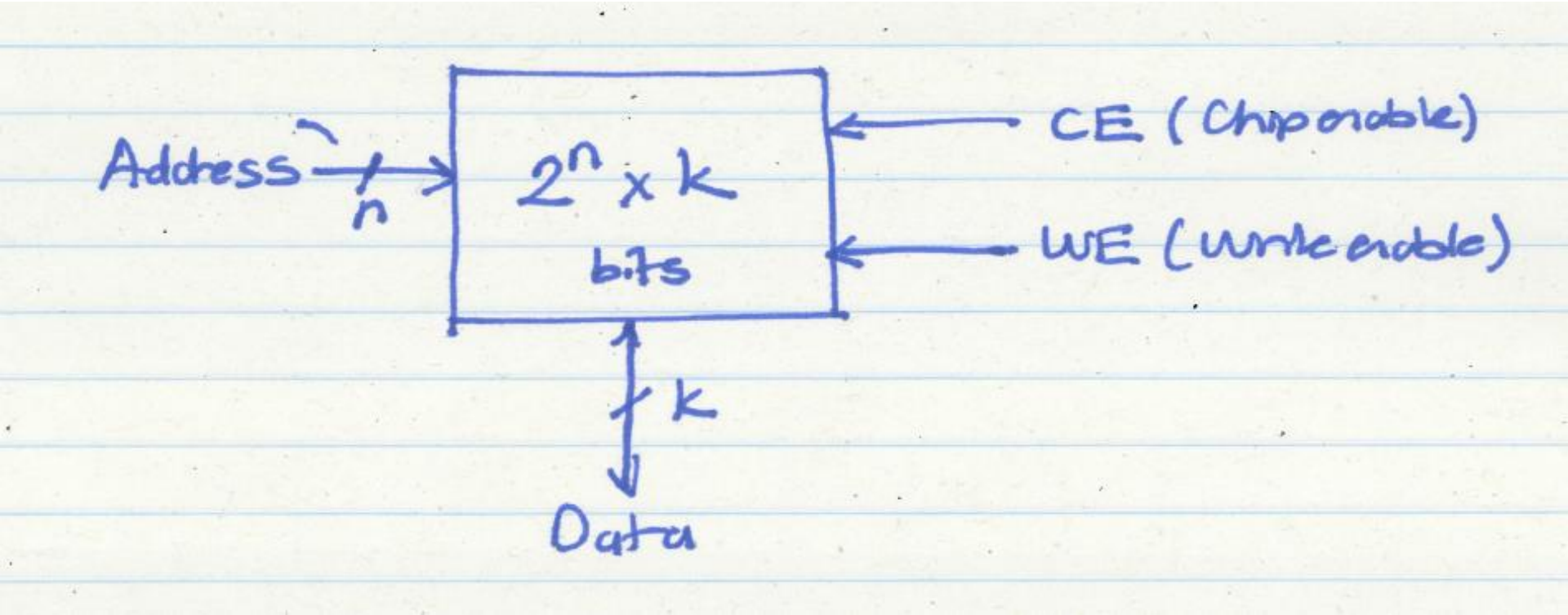
- Interleaving (banking)
  - Problem: a single monolithic memory array takes long to access and does not enable multiple accesses in parallel
  - Goal: Reduce the latency of memory array access and enable multiple accesses in parallel
  - Idea: Divide the array into multiple banks that can be accessed 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
  - Issue: How do you map data to different banks? (i.e., how do you interleave data across banks?)

Main Memory

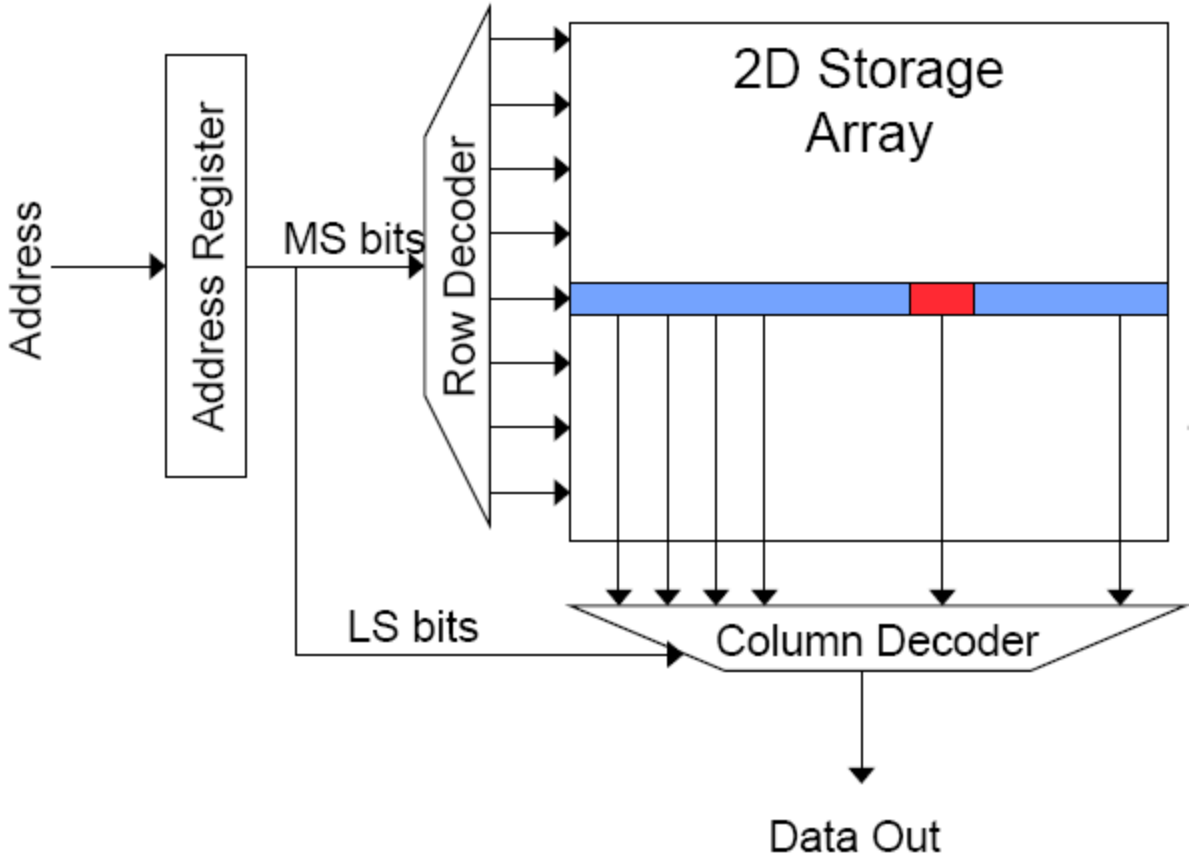
Main Memory in the System



The Memory Chip/System Abstraction



Review: Memory Bank Organization



- Read access sequence:

1. Decode row address & drive word-lines

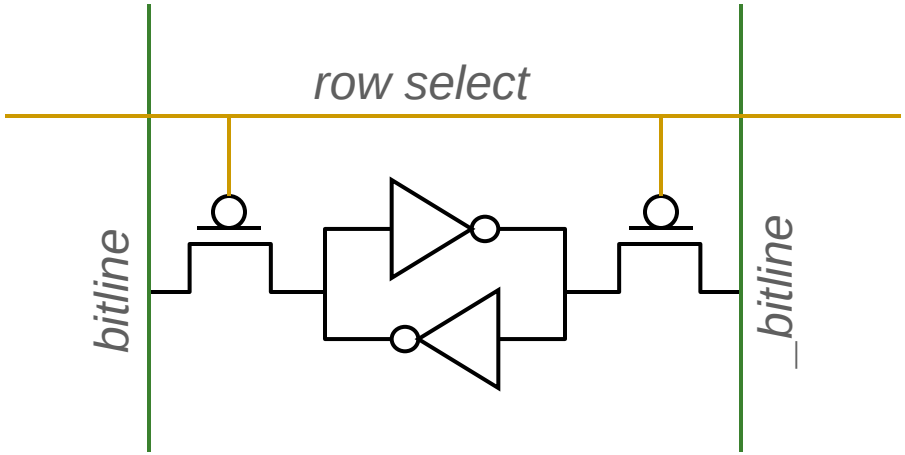
2. Selected bits drive bit-lines
   - Entire row read

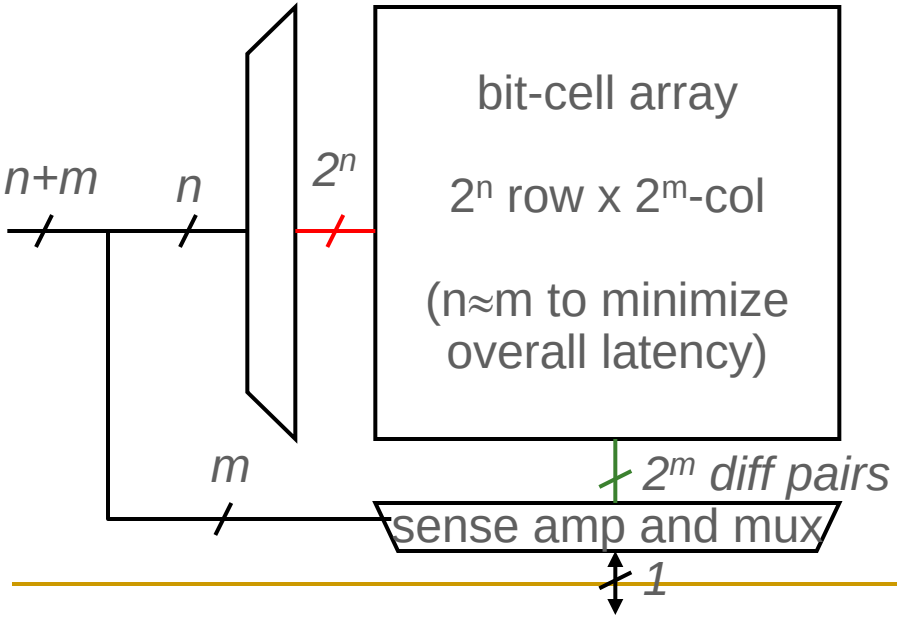
3. Amplify row data

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

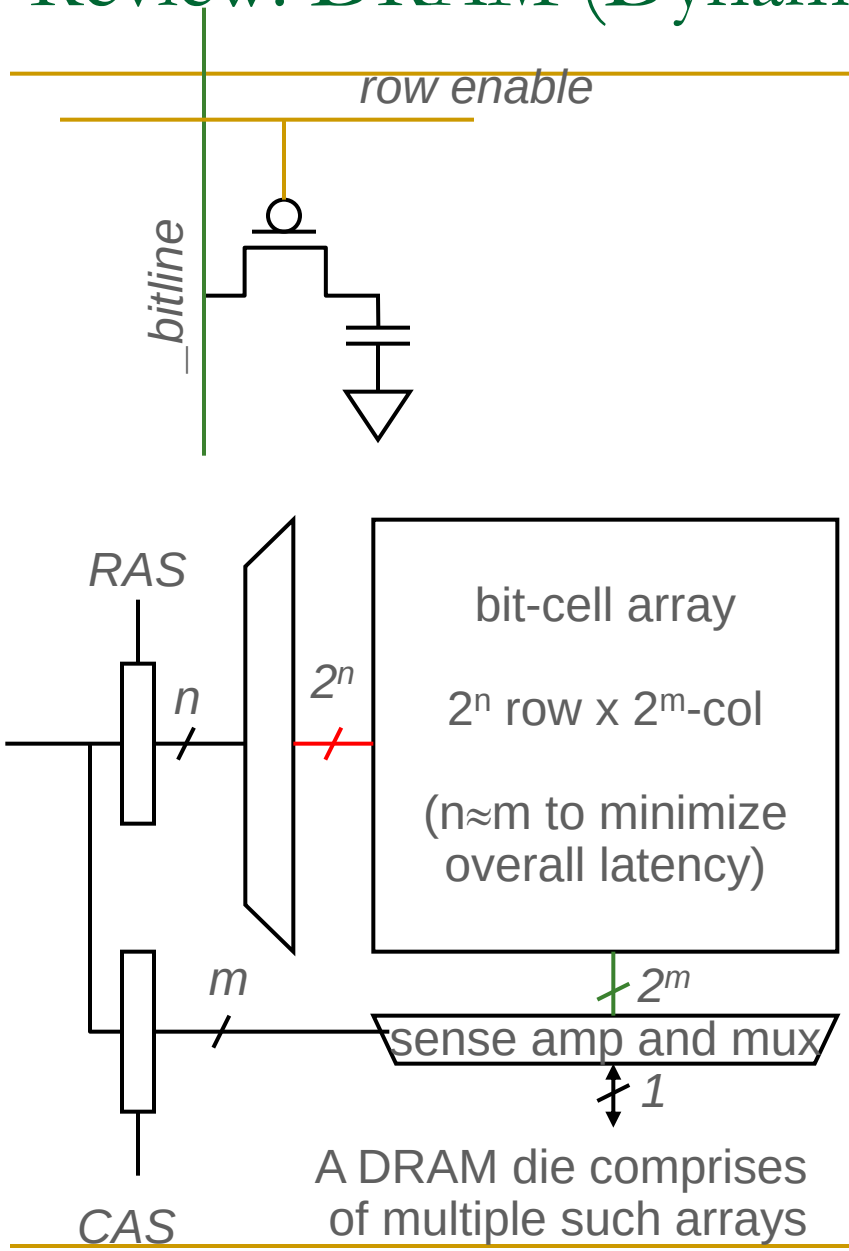
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. diff. sensing and col. select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

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)



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)

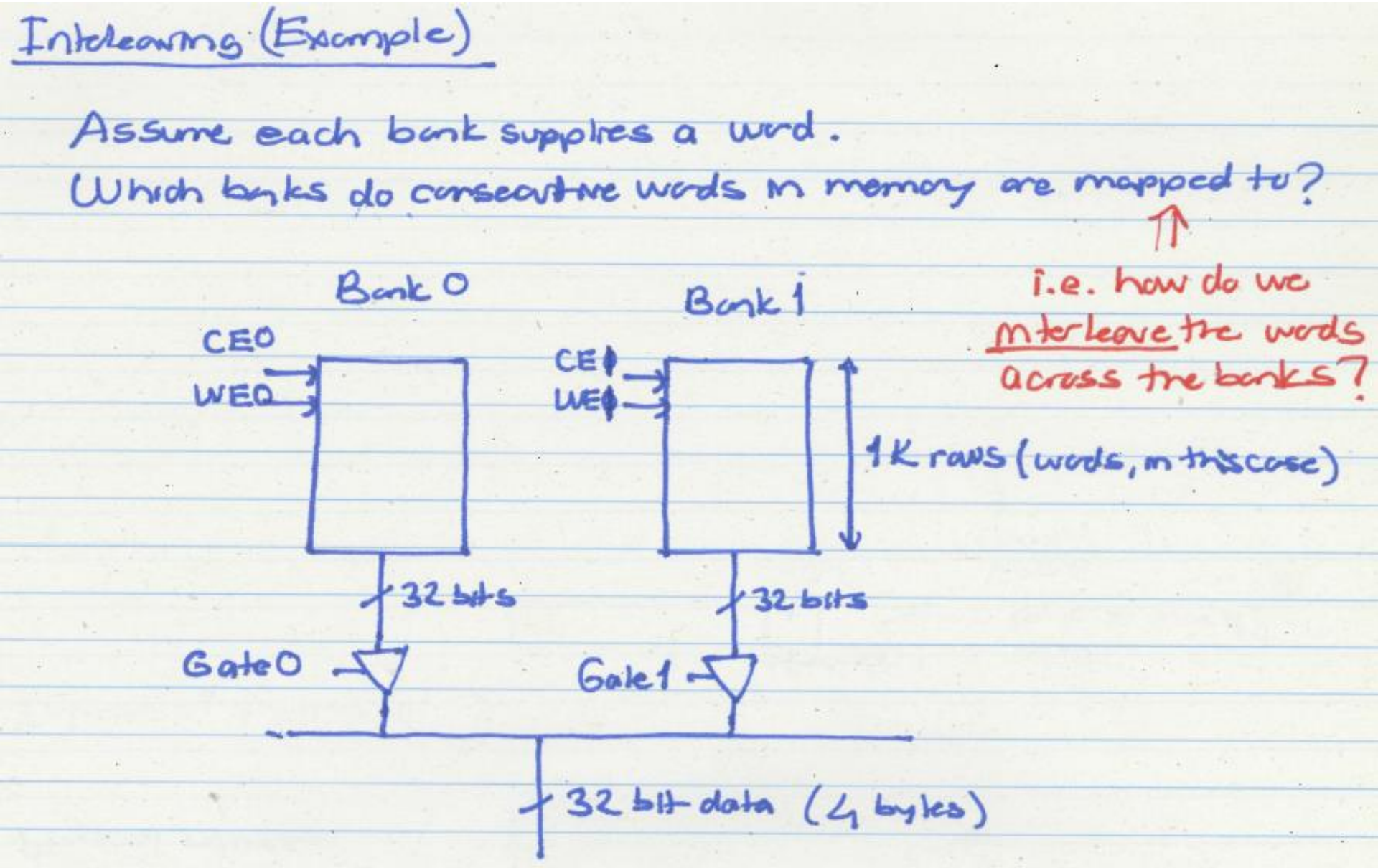
Some Fundamental Concepts (I)

- 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
  - Addressability depends on the abstraction level of the implementation
- Alignment
  - Does the hardware support unaligned access transparently to software?
- Interleaving

Some Fundamental Concepts (II)

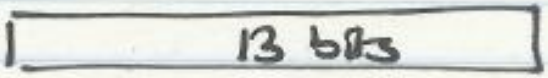
- Interleaving (banking)
  - Problem: a single monolithic memory array takes long to access and does not enable multiple accesses in parallel
  - Goal: Reduce the latency of memory array access and enable multiple accesses in parallel
  - Idea: Divide the array into multiple banks that can be accessed 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
  - Issue: How do you map data to different banks? (i.e., how do you interleave data across banks?)

Interleaving

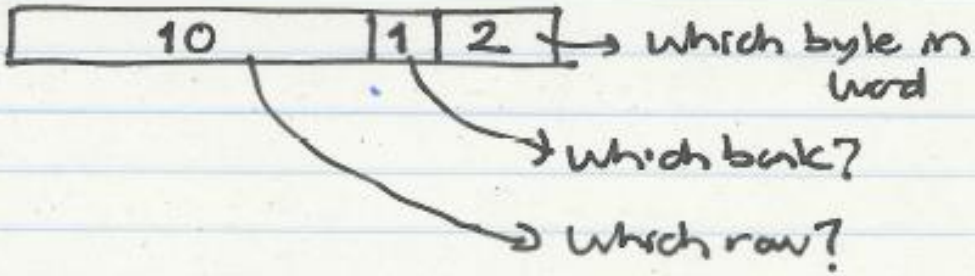


Interleaving Options

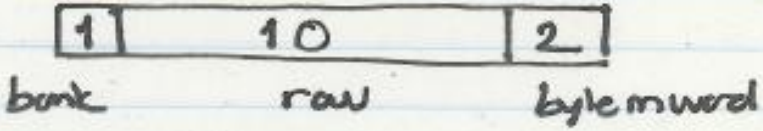
Physical address



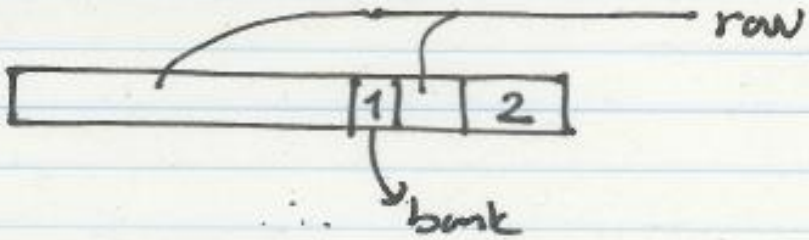
Interleaving scheme 1



Interleaving scheme 2



Interleaving scheme 3

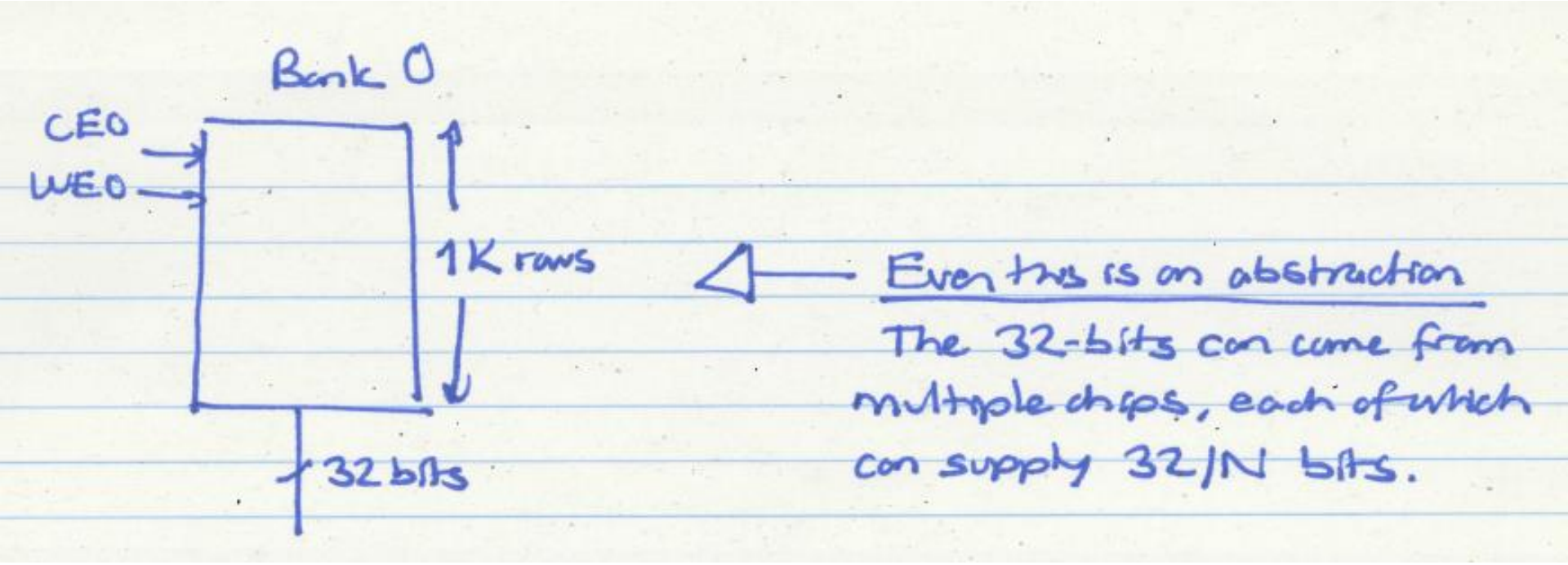


Where (which bank) do consecutive words in memory are mapped to?

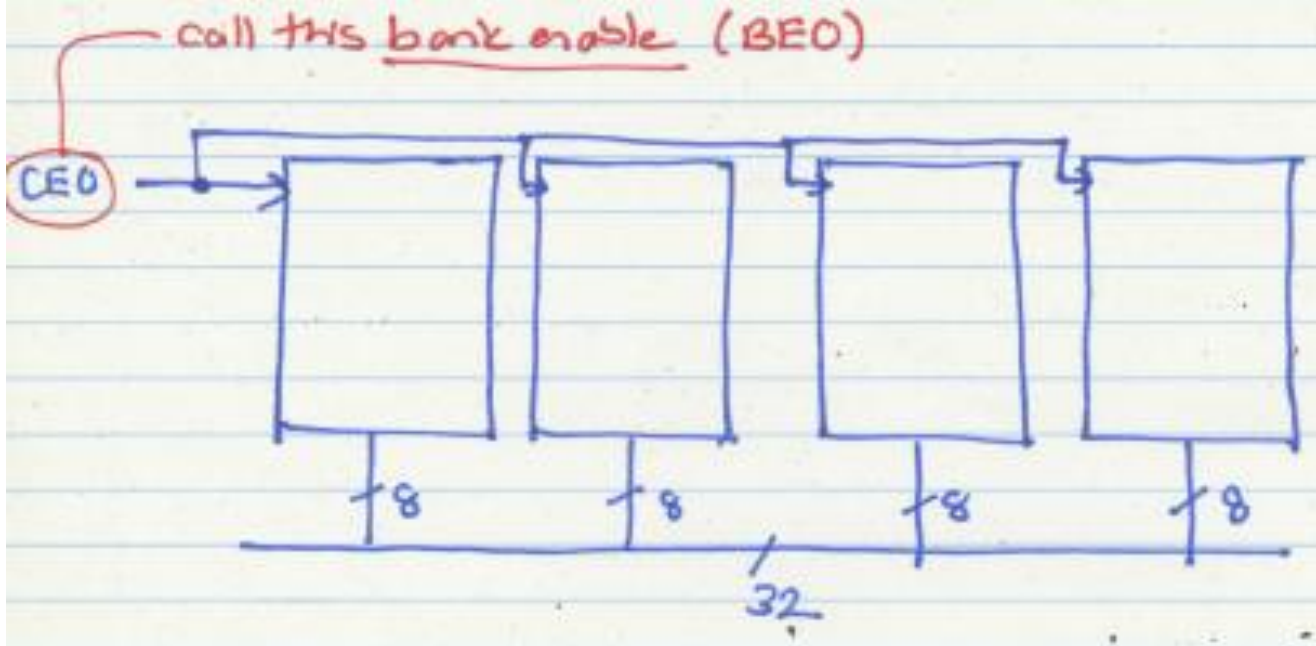
Some Questions/Concepts

- 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

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

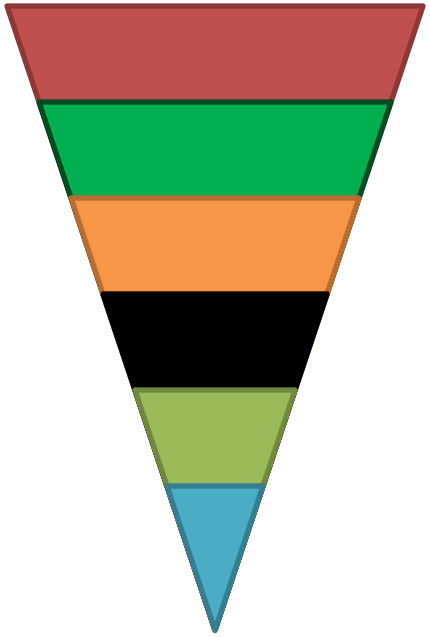
Why? Producing an 8-bit/pin chip cheaper than producing a 32-bit/pin chip

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.

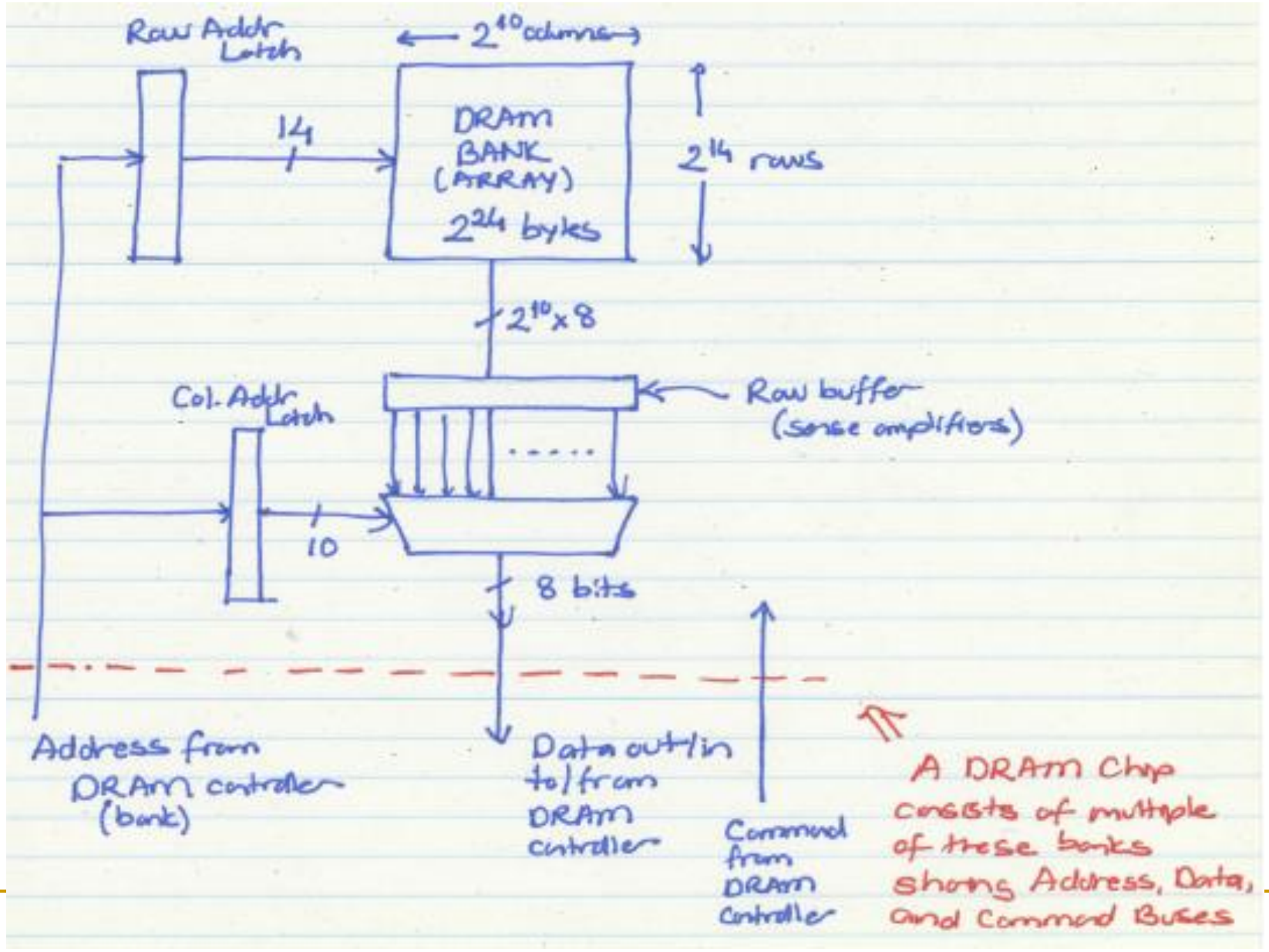
The DRAM Subsystem

DRAM Subsystem Organization

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



The DRAM Bank Structure



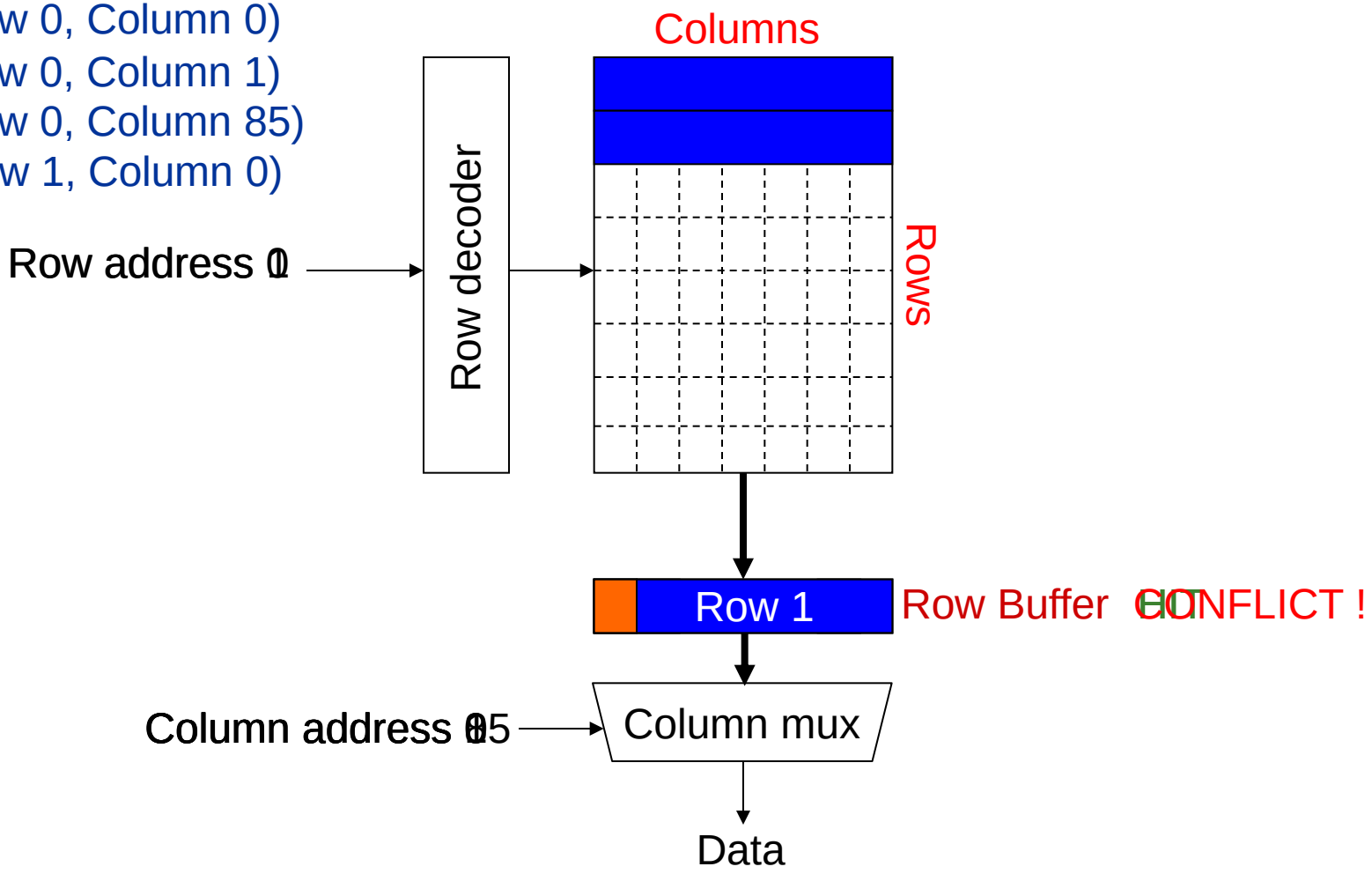
Page Mode DRAM

- 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

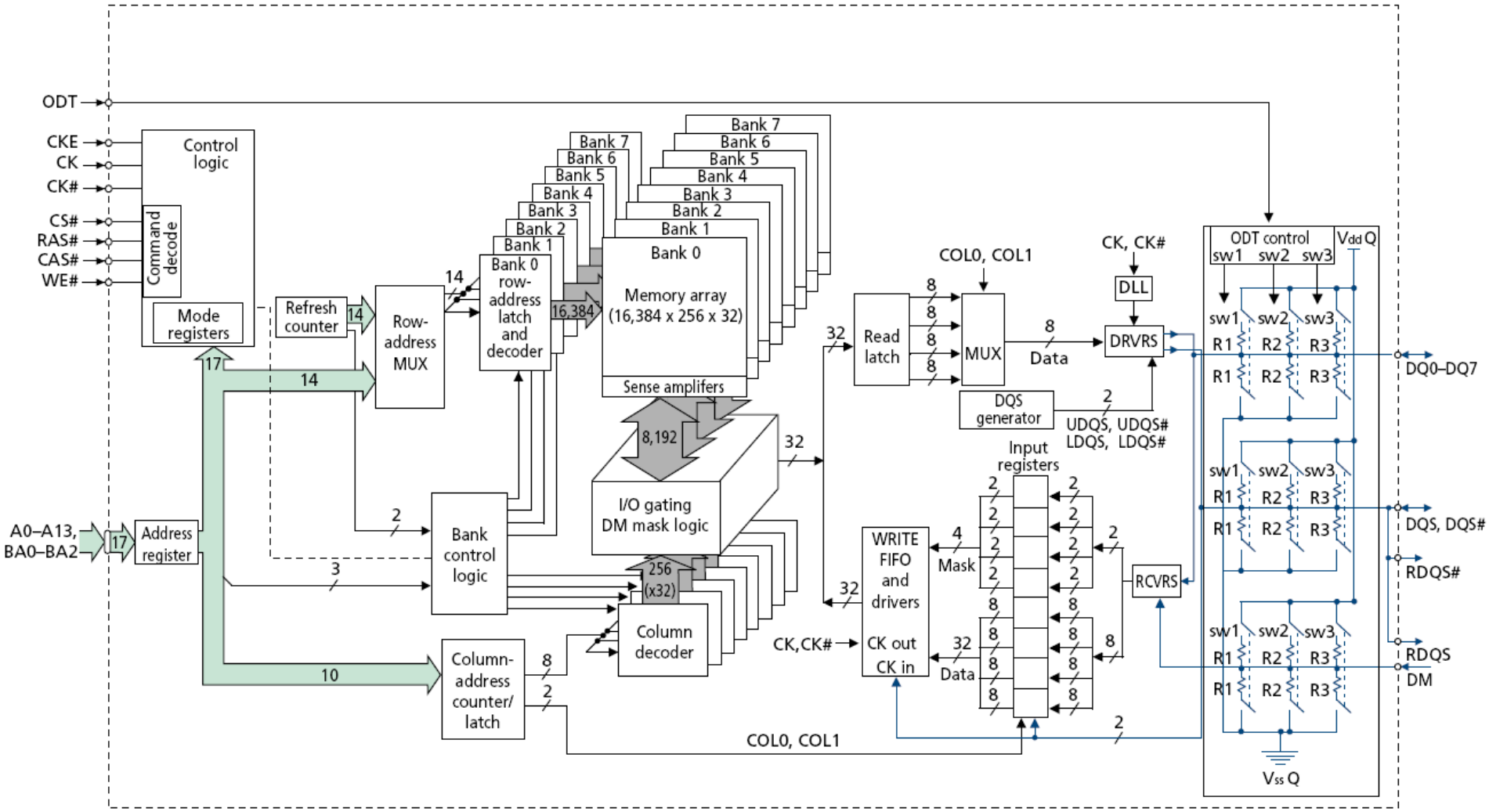
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



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)

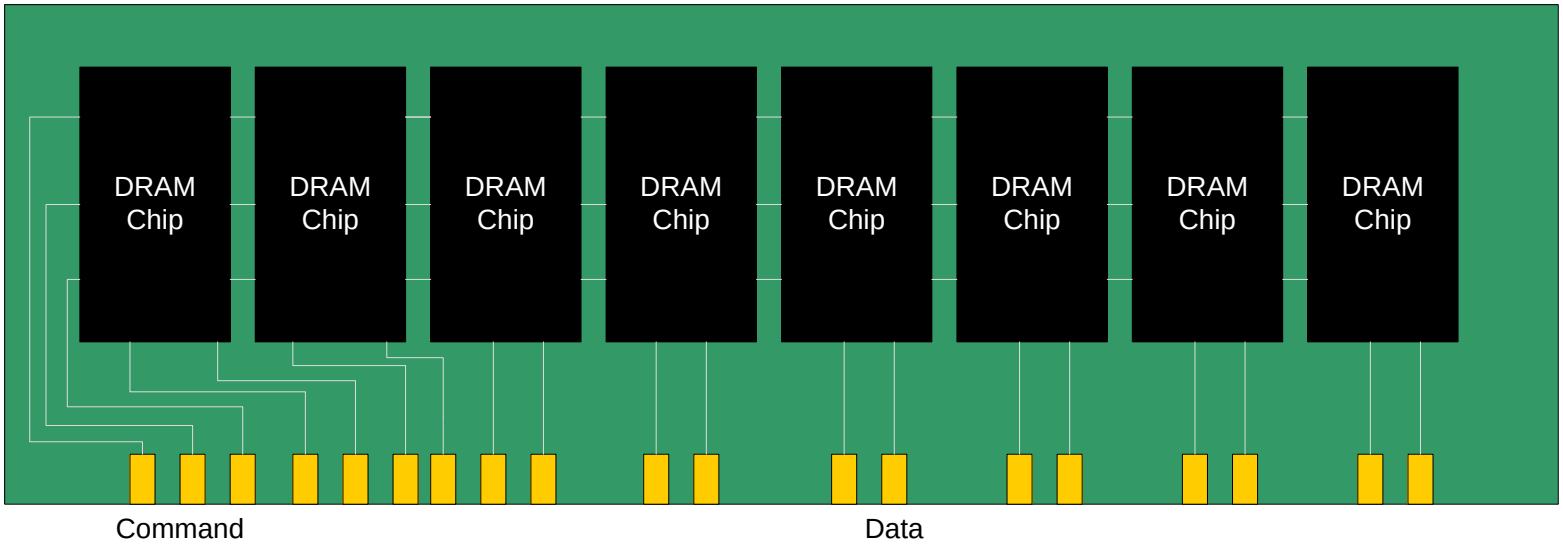
128M x 8-bit DRAM Chip



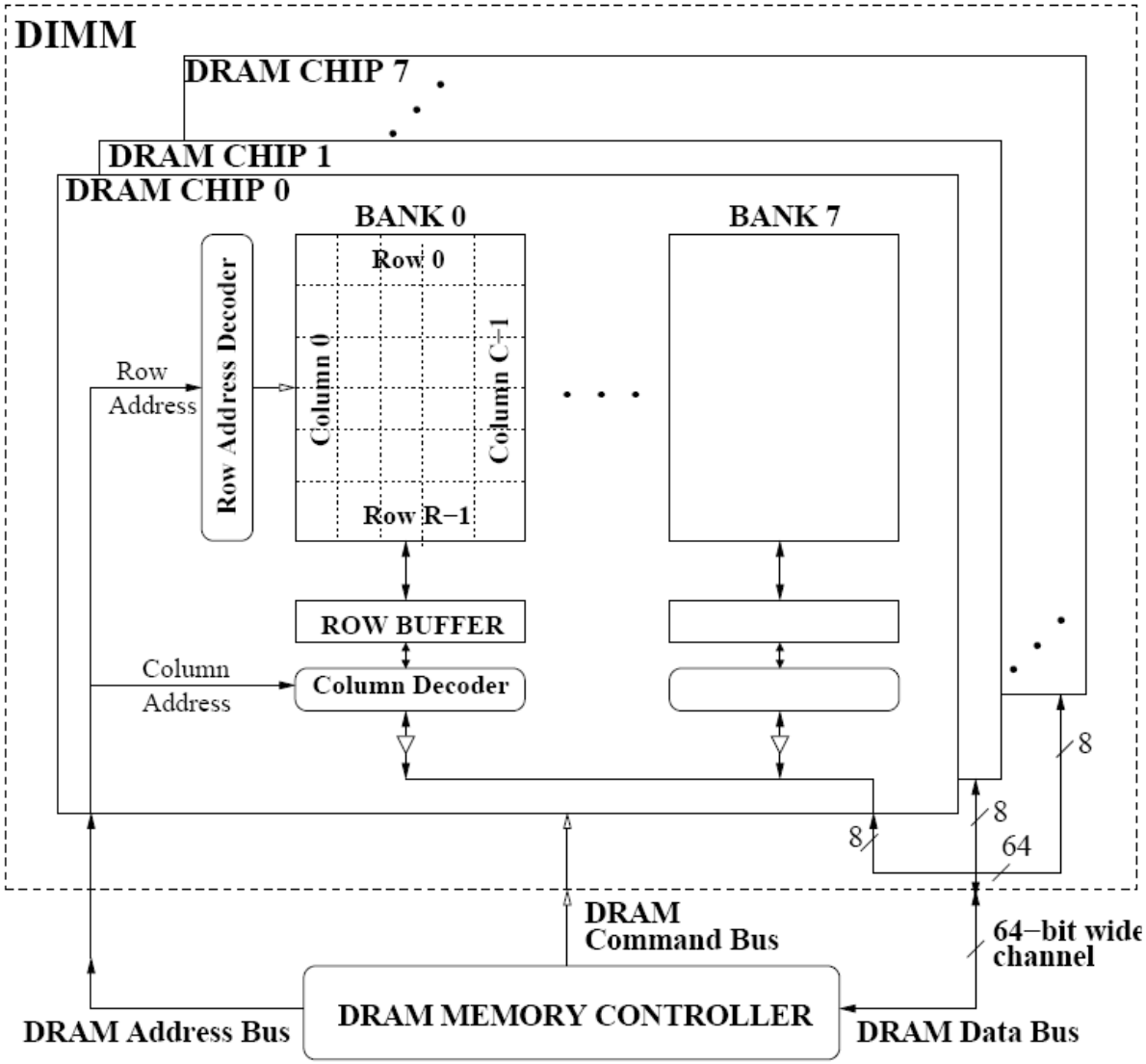
DRAM Rank and Module

- 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)

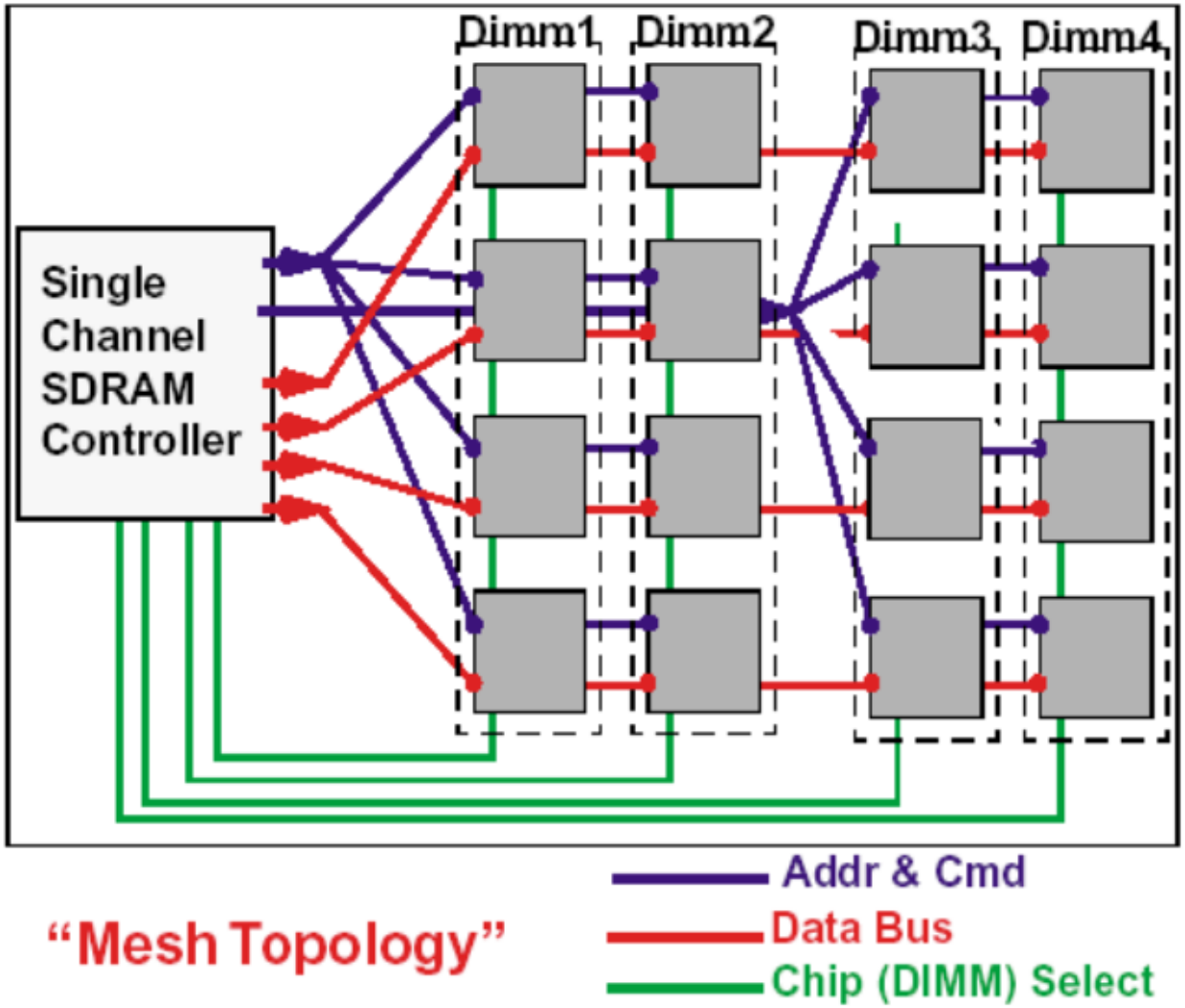


A 64-bit Wide DIMM (One Rank)



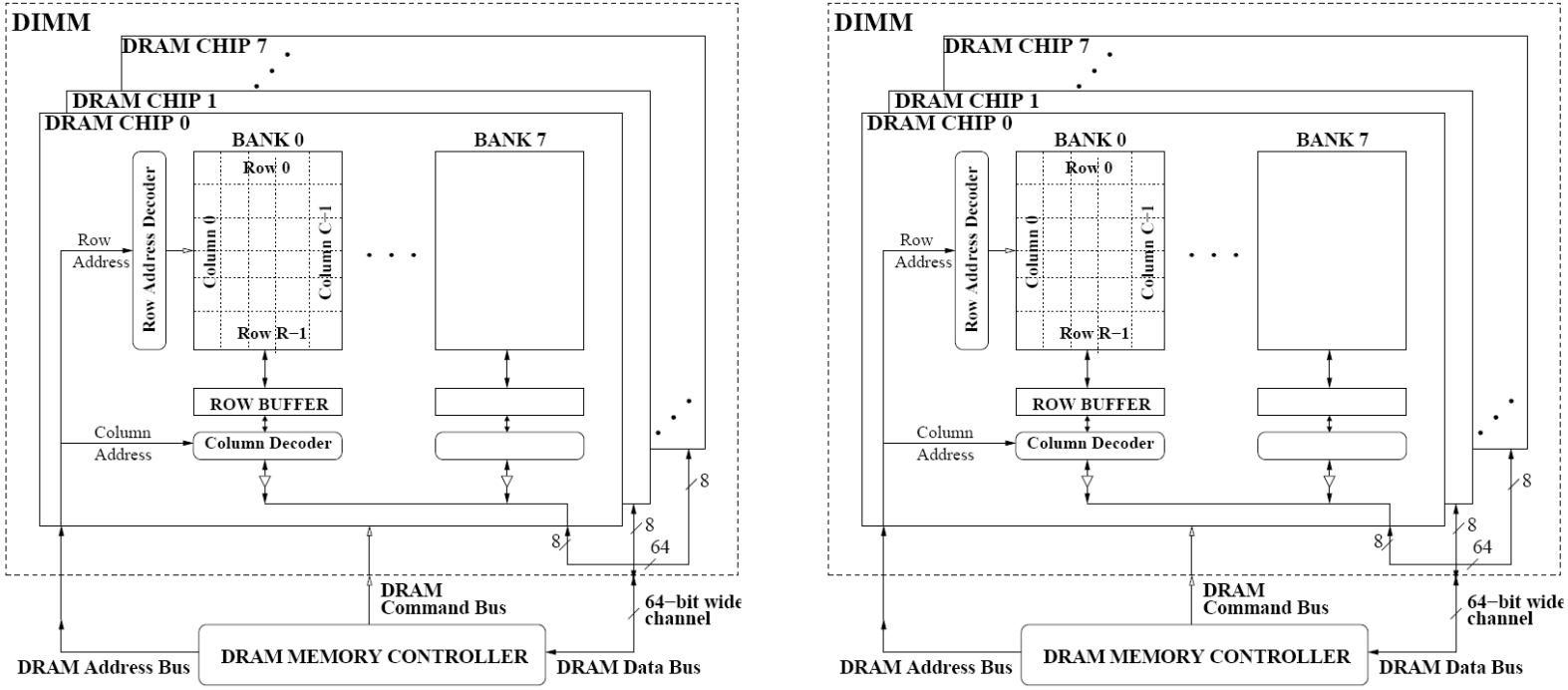
- Advantages:
  - Acts like a high-capacity DRAM chip with a wide interface
  - Flexibility: memory controller does not need to deal with individual chips
- Disadvantages:
  - Granularity: Accesses cannot be smaller than the interface width

Multiple DIMMs



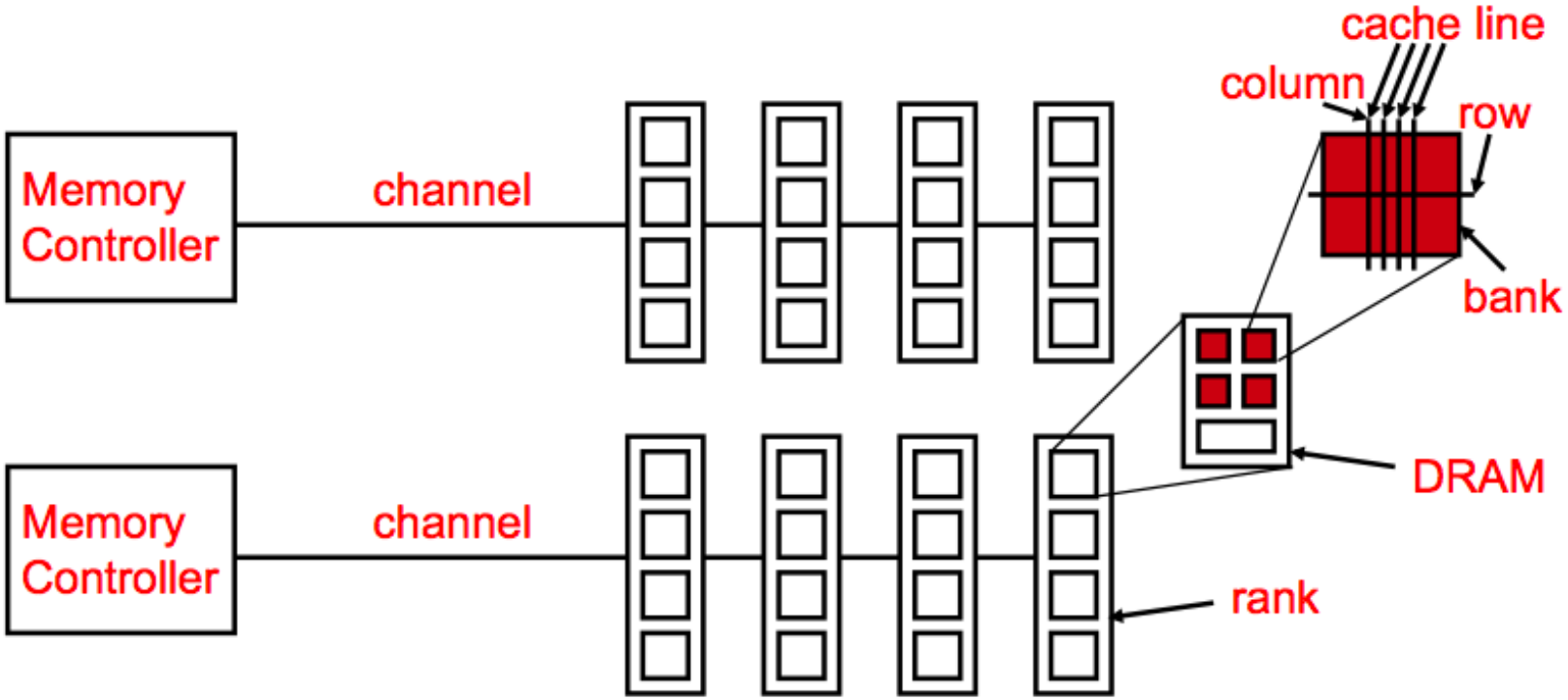
- Advantages:
  - Enables even higher capacity
- Disadvantages:
  - Interconnect complexity and energy consumption can be high

DRAM Channels

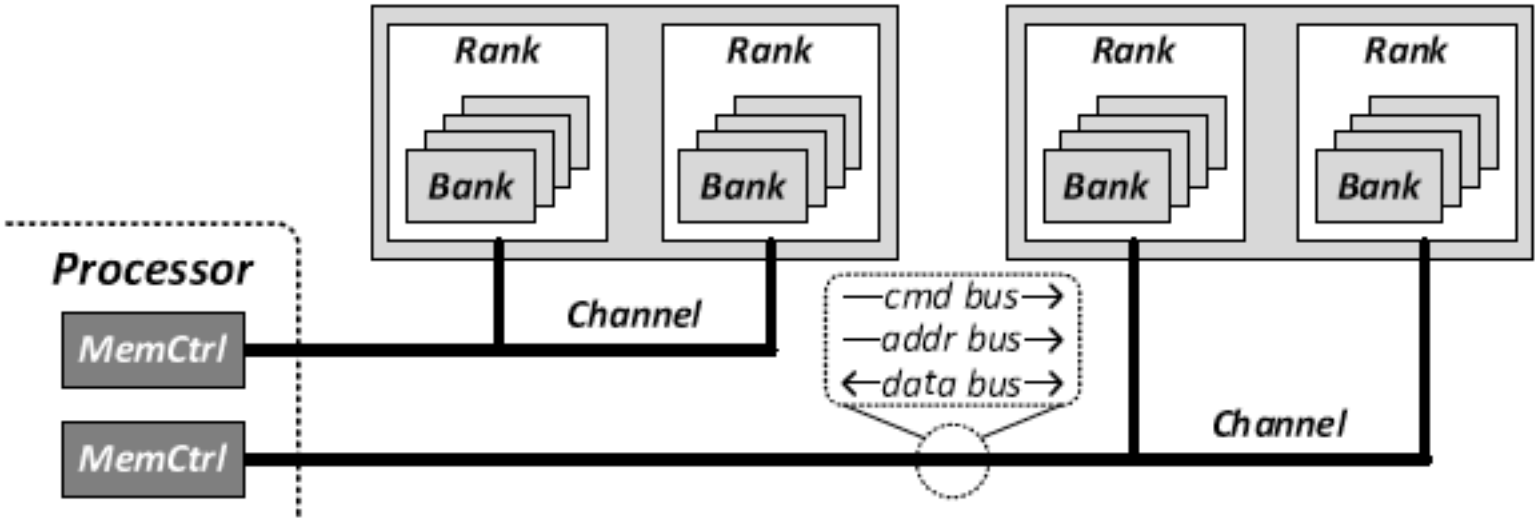


- 2 Independent Channels: 2 Memory Controllers (Above)
- 2 Dependent/Lockstep Channels: 1 Memory Controller with wide interface (Not Shown above)

Generalized Memory Structure



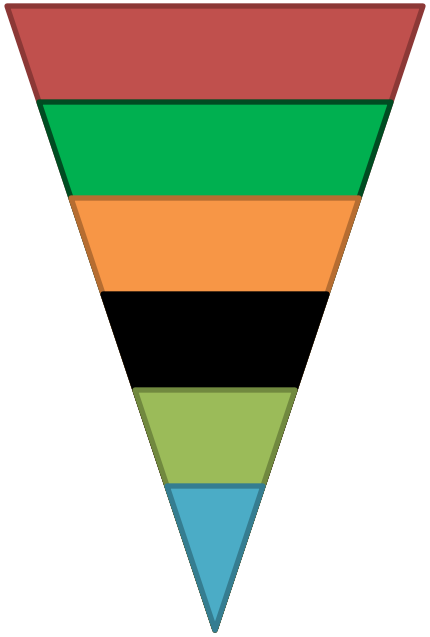
Generalized Memory Structure



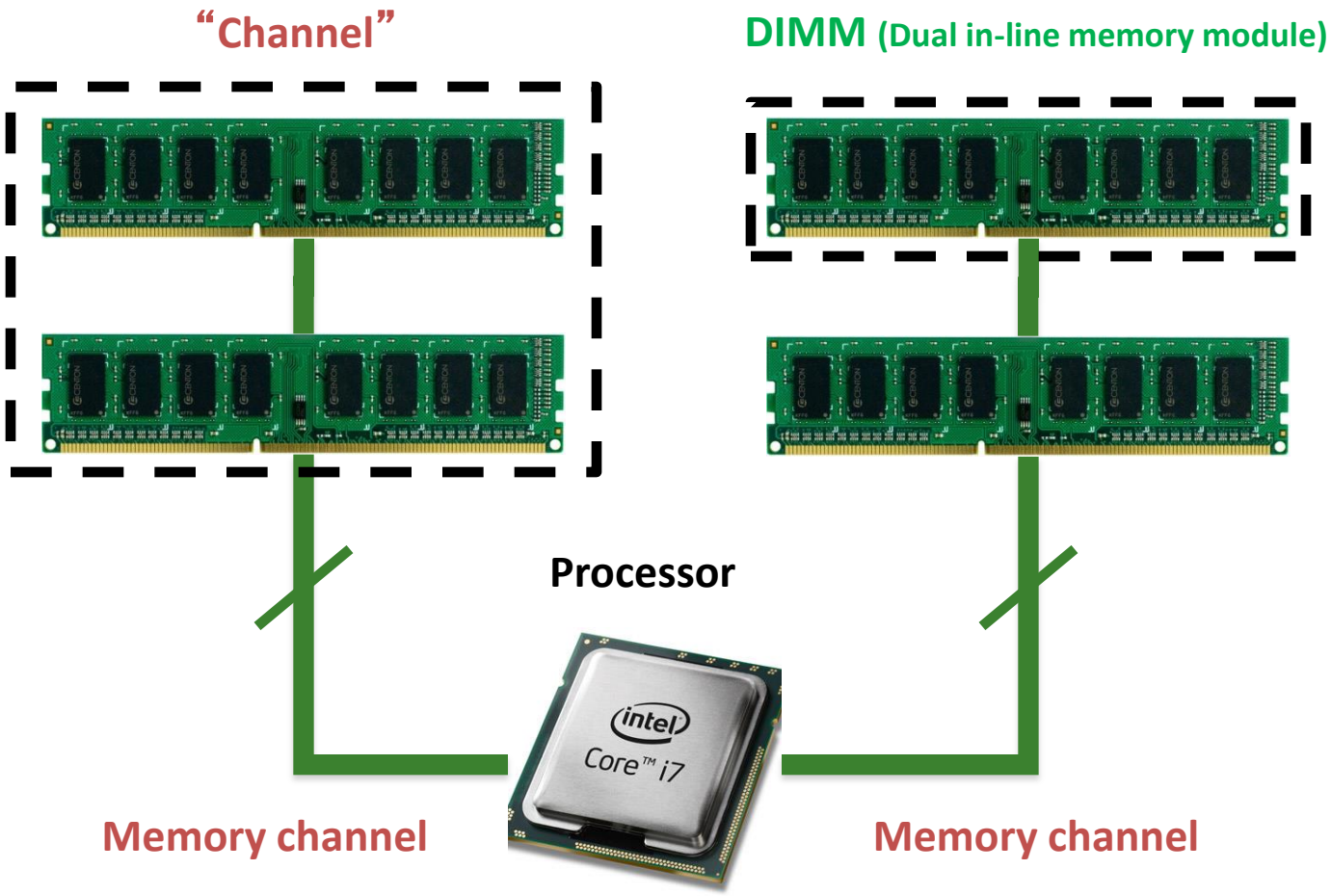
The DRAM Subsystem
The Top Down View

DRAM Subsystem Organization

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



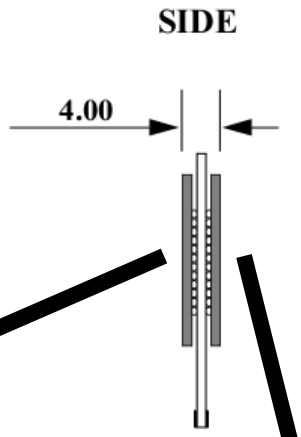
The DRAM subsystem



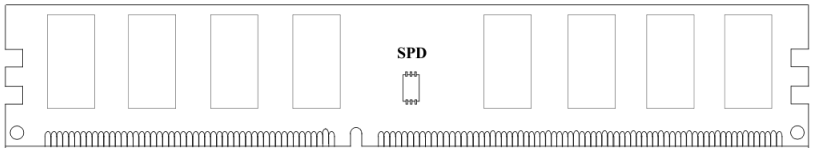
Breaking down a DIMM

DIMM (Dual in-line memory module)

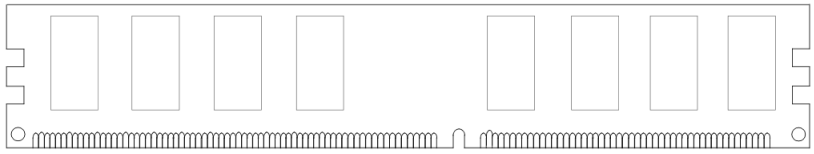
Side view



Front of DIMM



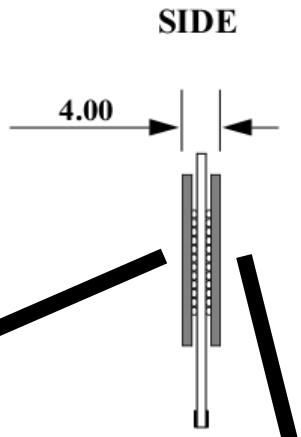
Back of DIMM



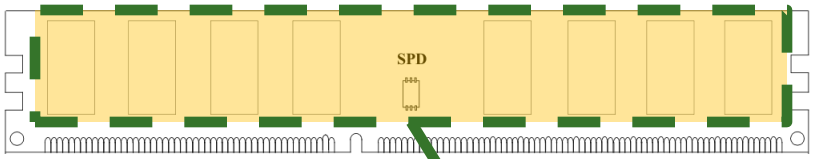
Breaking down a DIMM

DIMM (Dual in-line memory module)

Side view

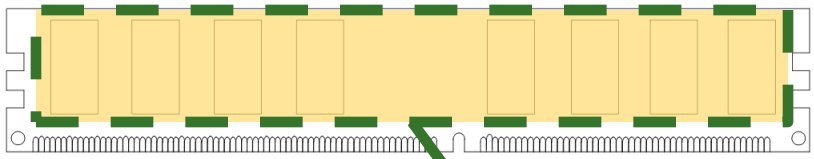


Front of DIMM



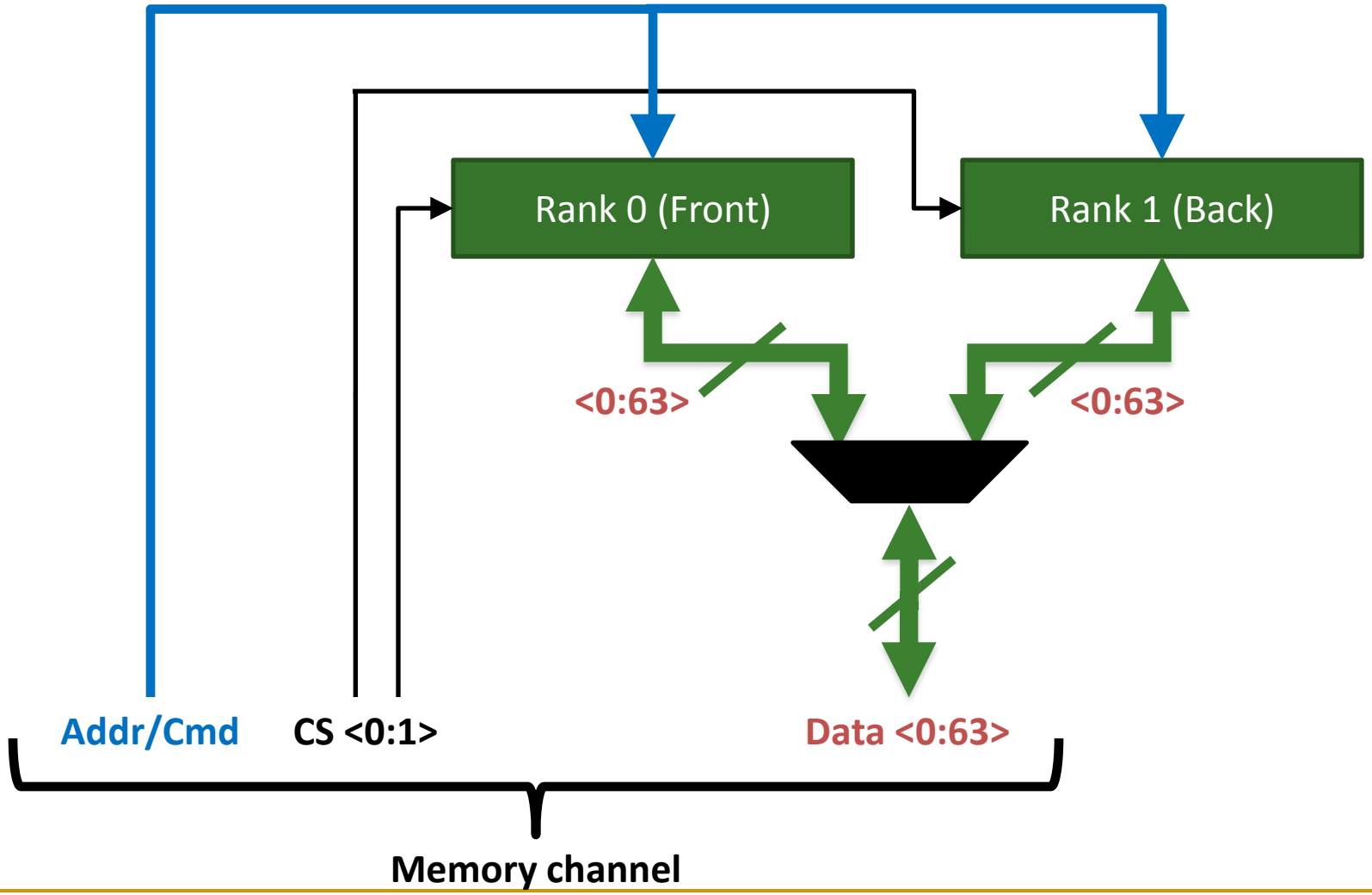
Rank 0: collection of 8 chips

Back of DIMM

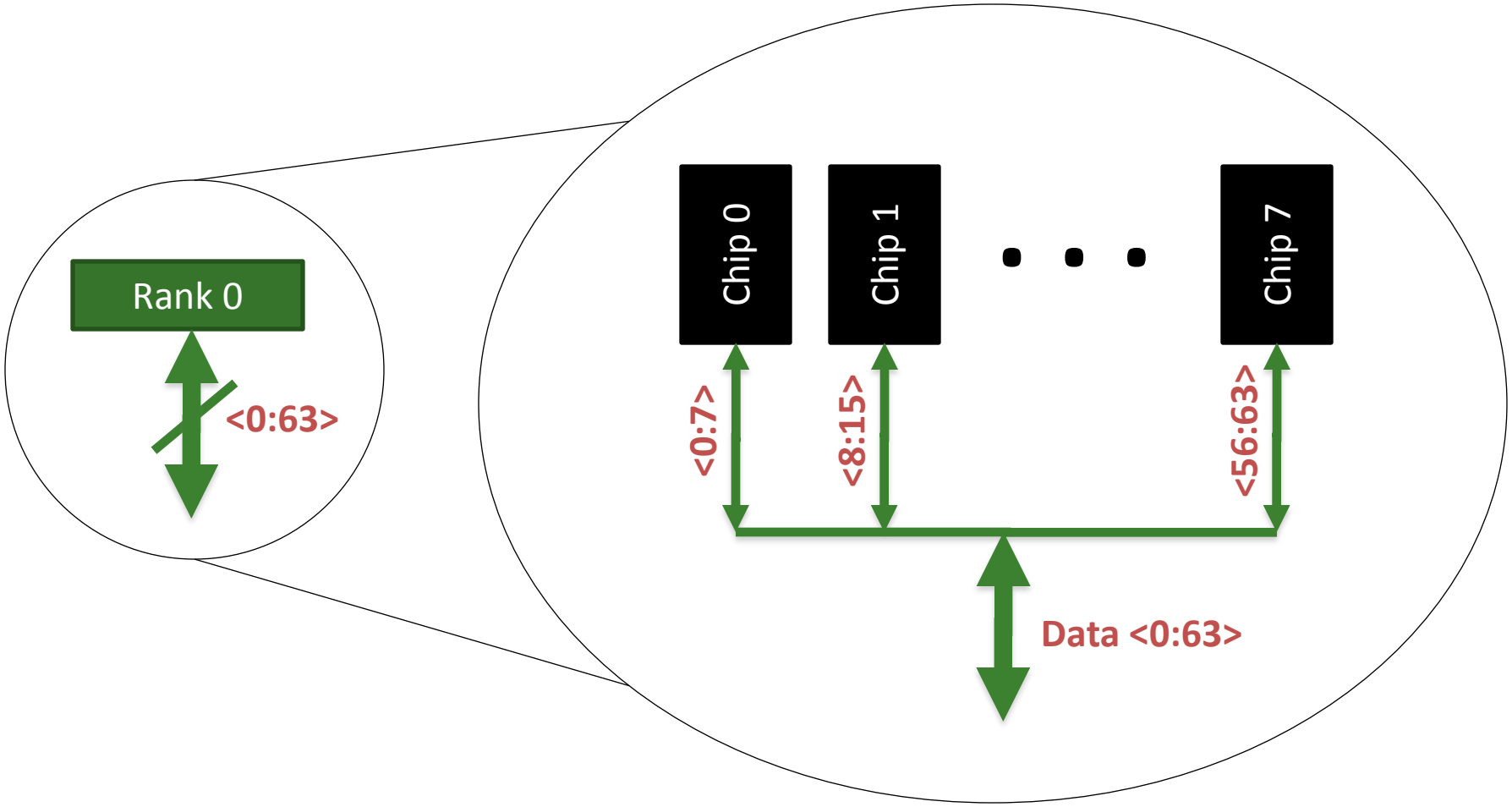


Rank 1

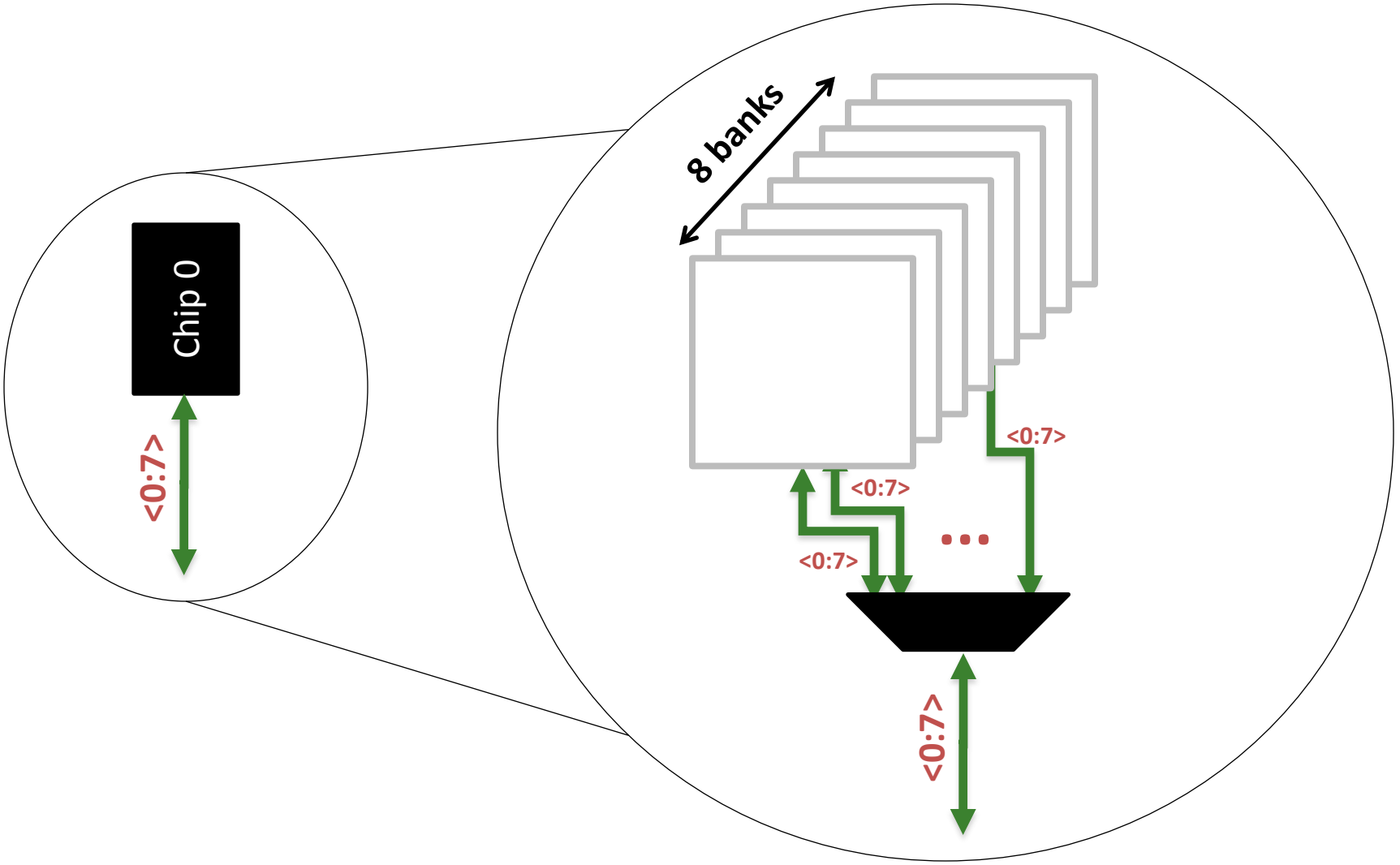
Rank


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

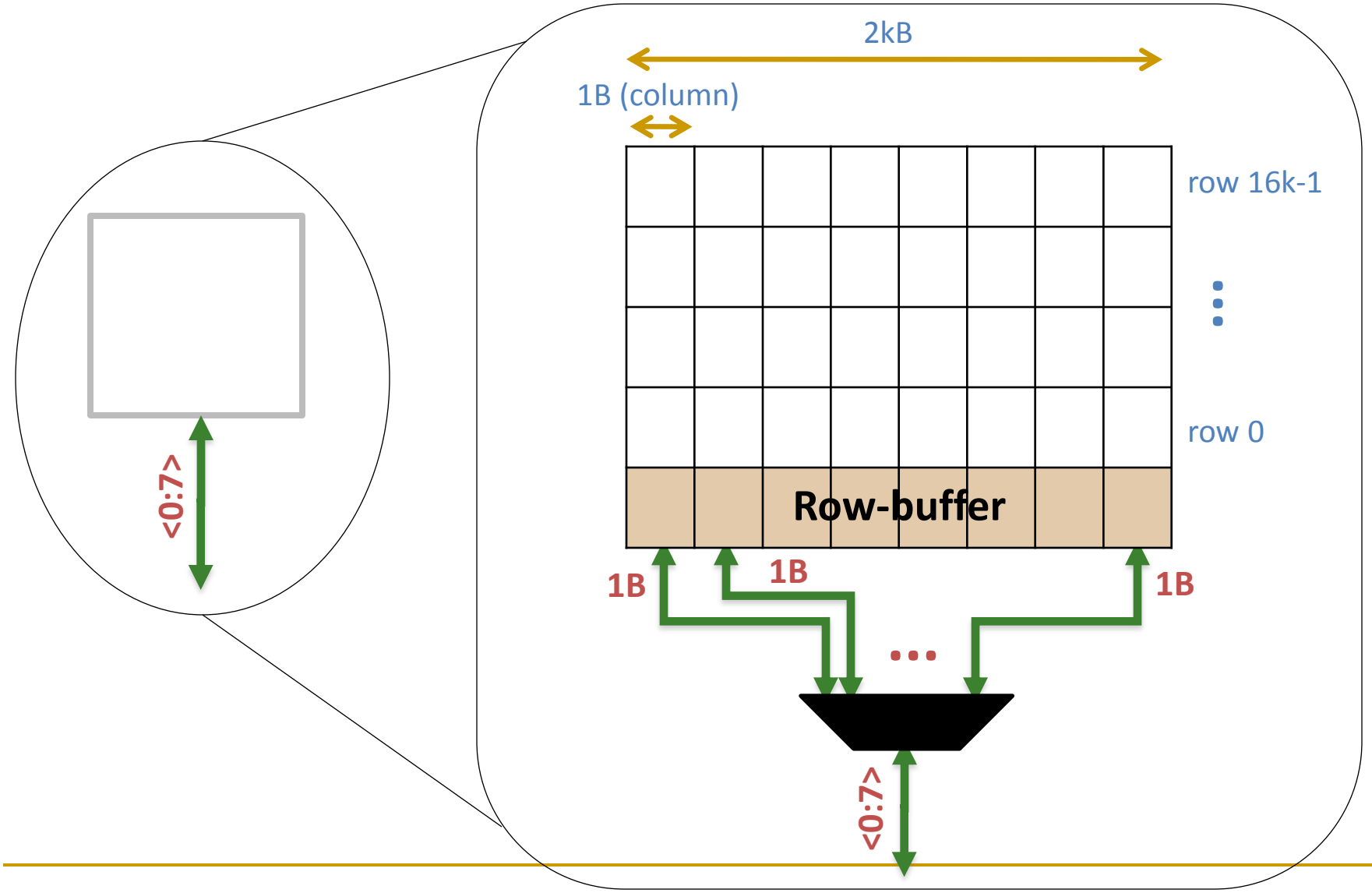
Breaking down a Rank



Breaking down a Chip

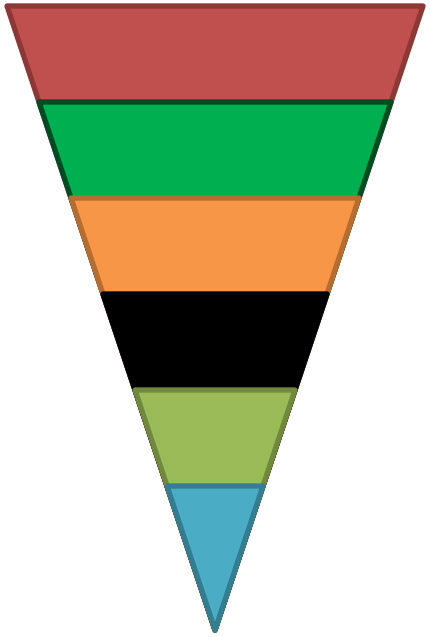


Breaking down a Bank

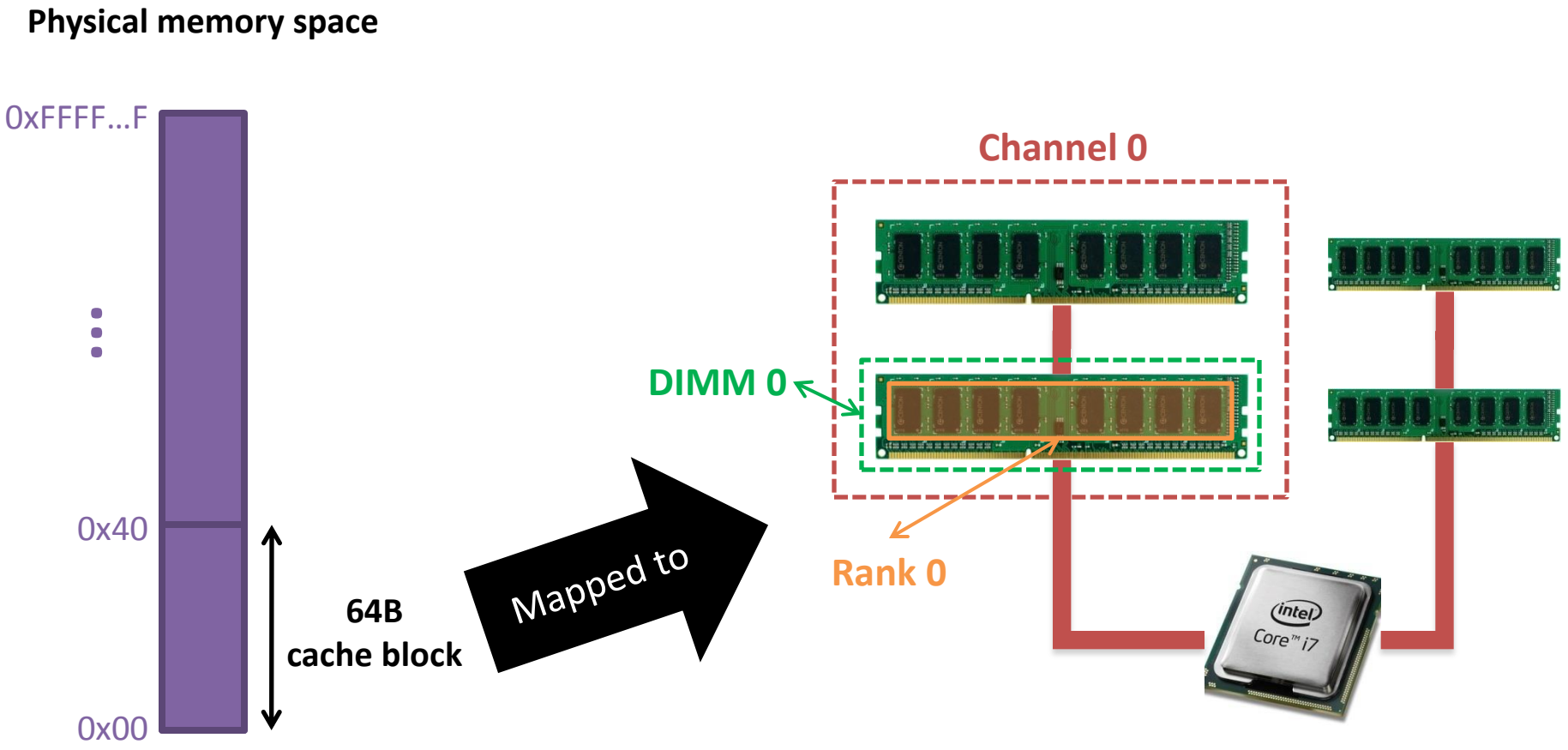


DRAM Subsystem Organization

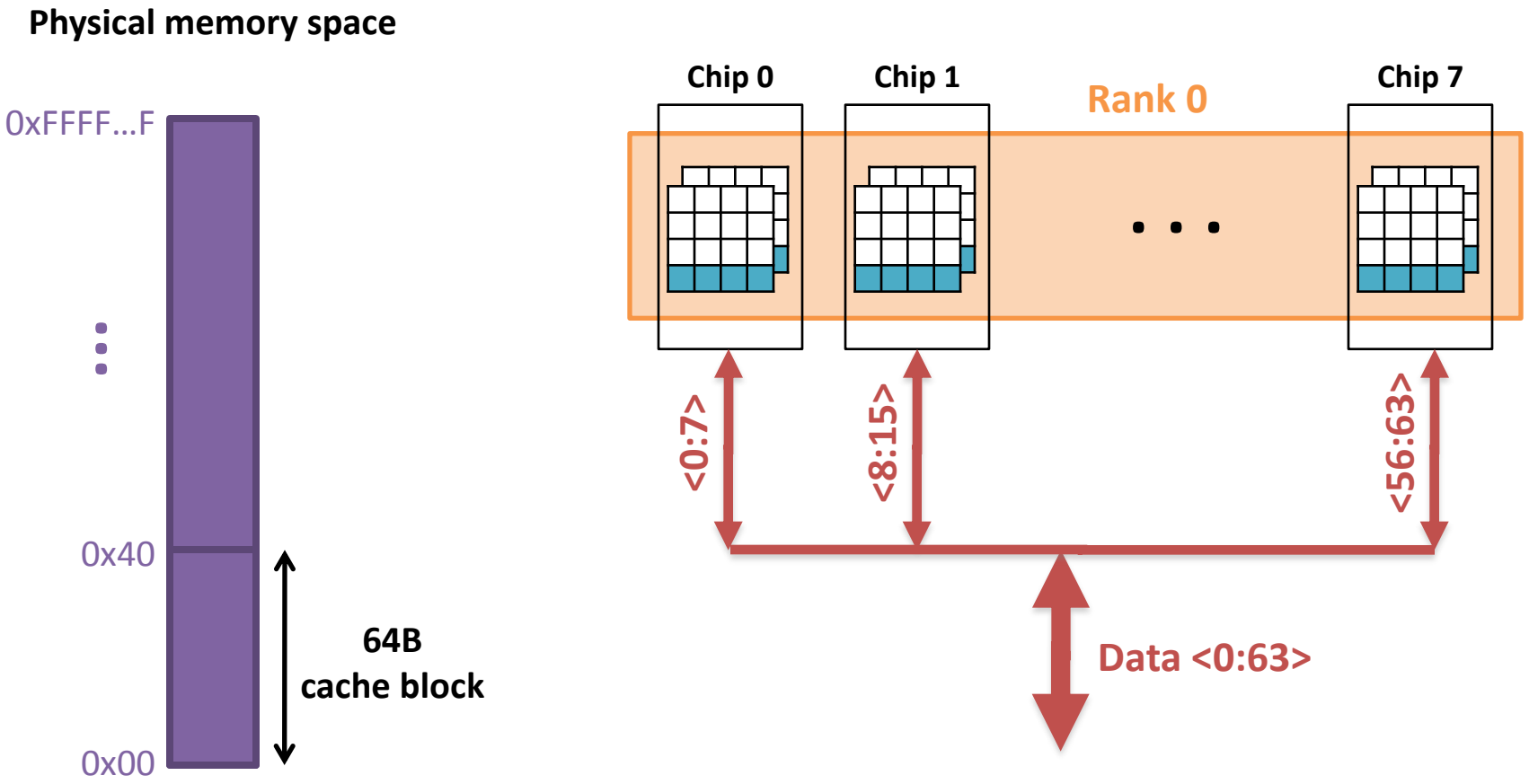
- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



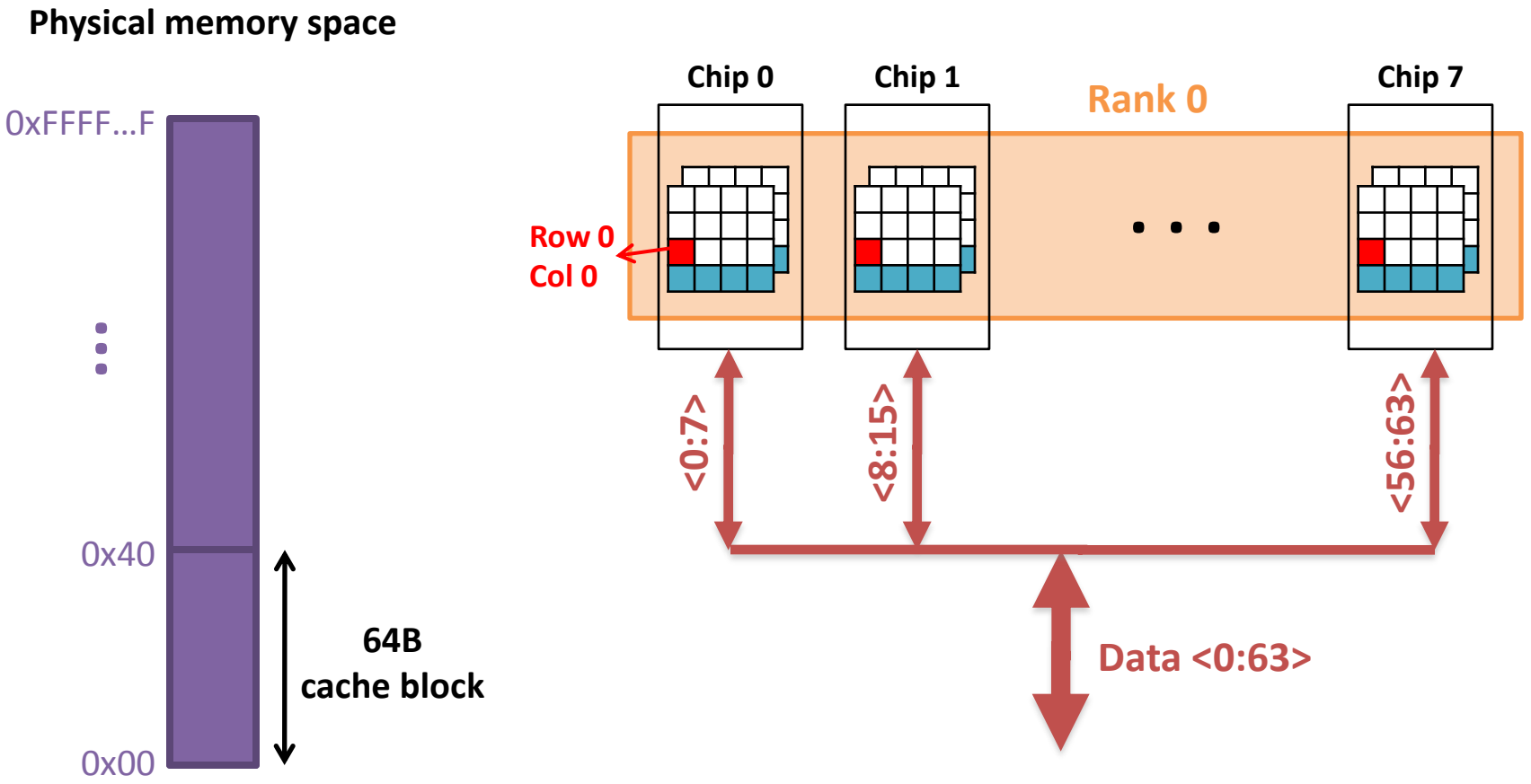
Example: Transferring a cache block



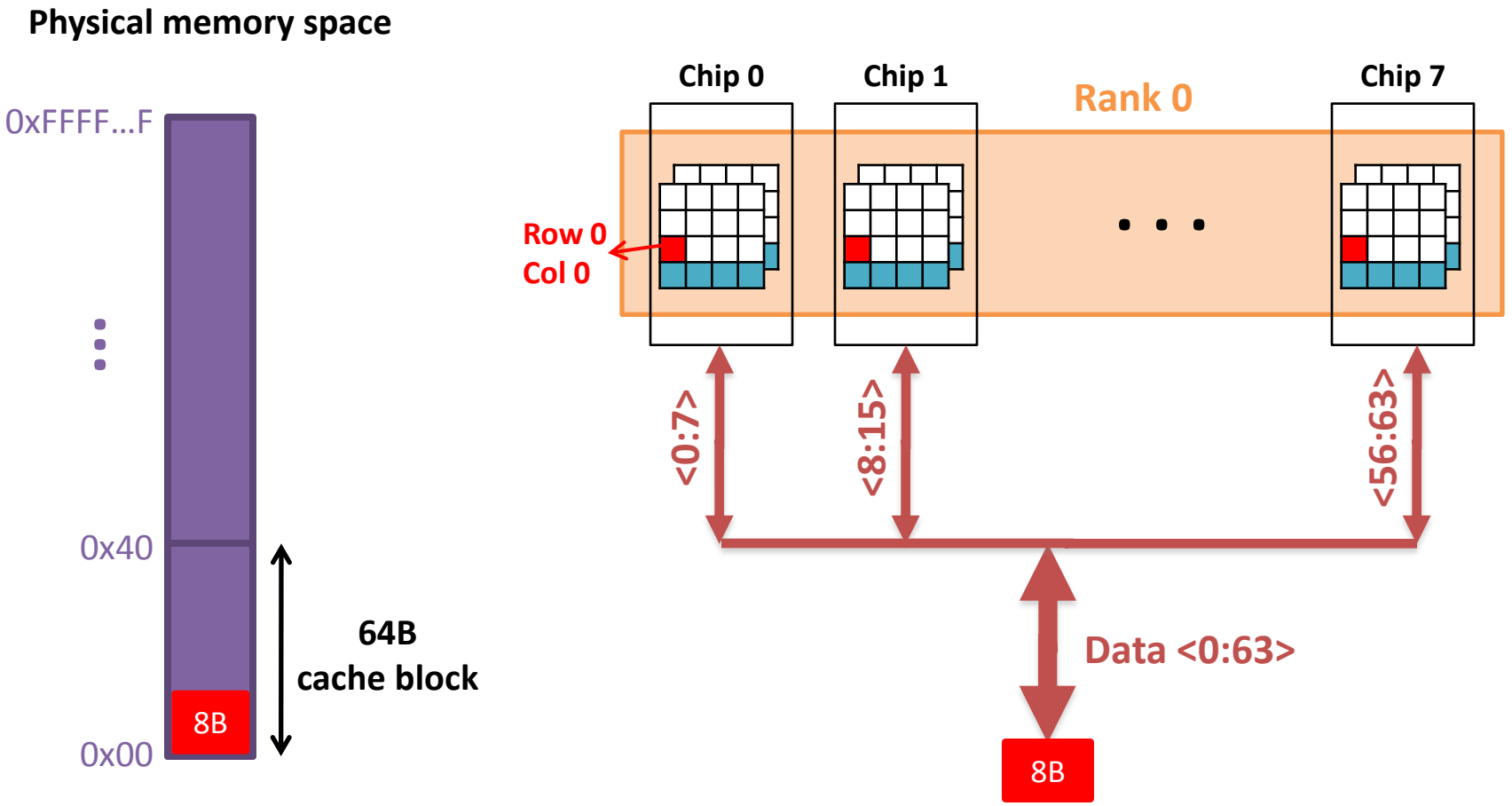
Example: Transferring a cache block



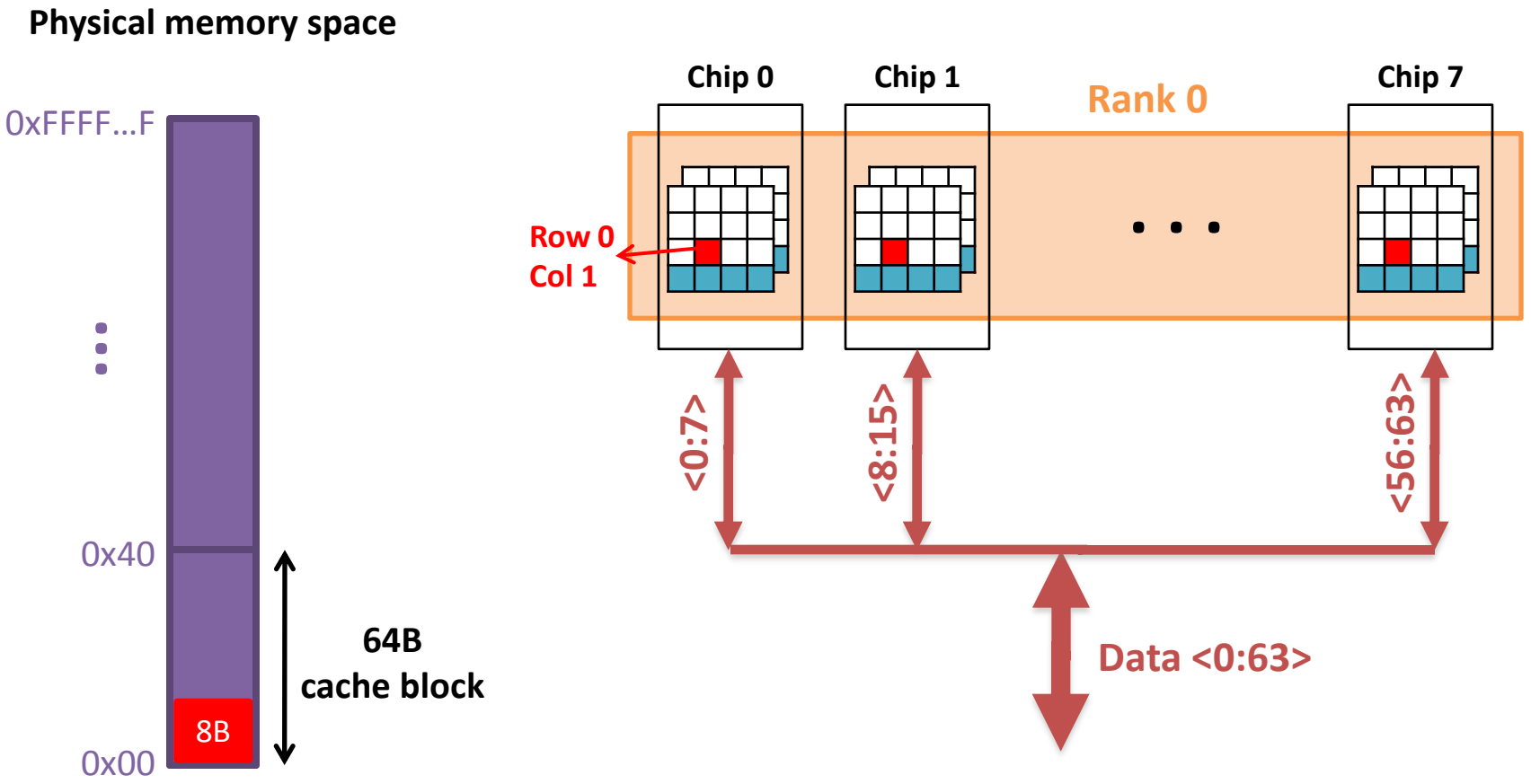
Example: Transferring a cache block



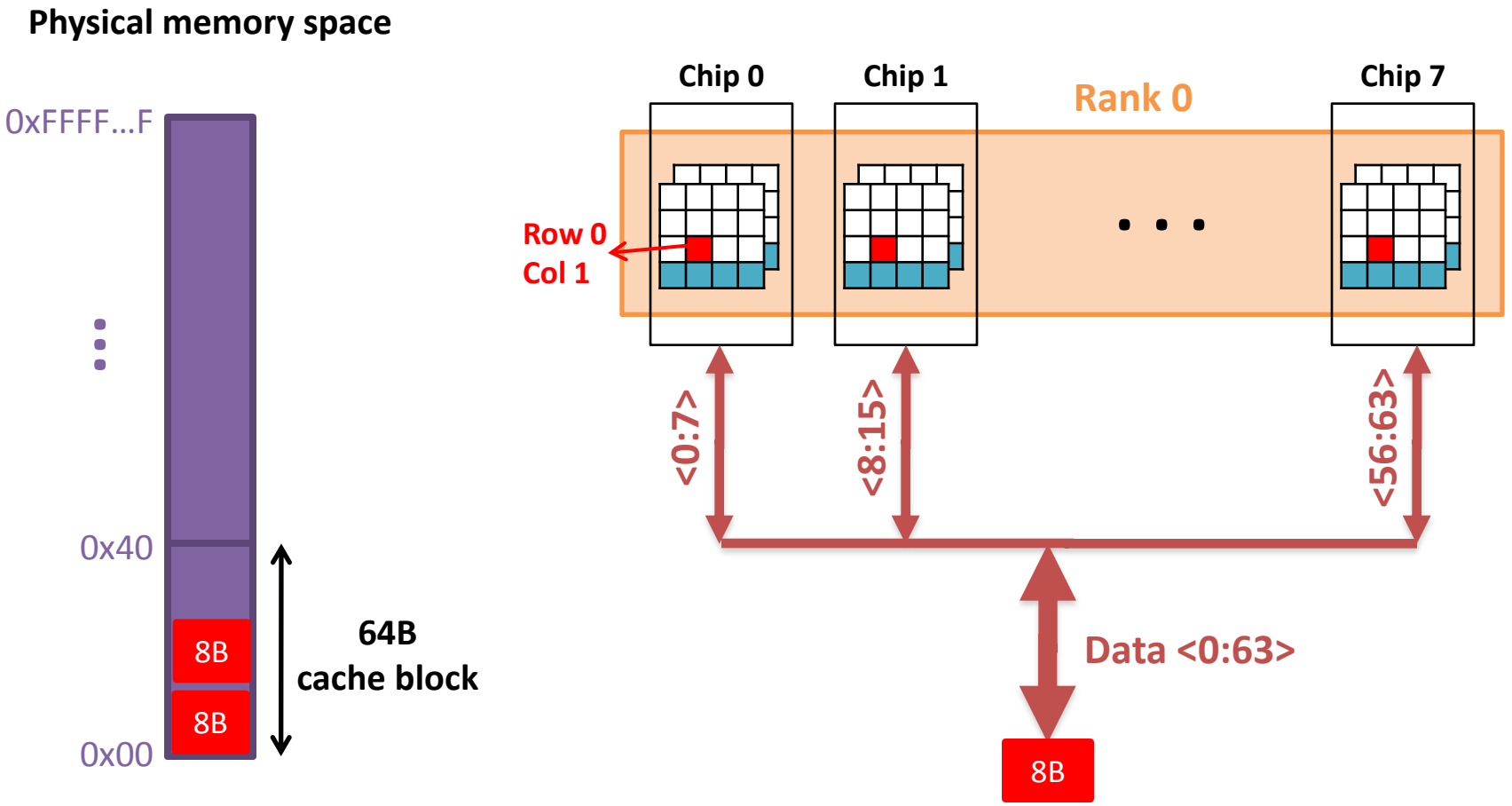
Example: Transferring a cache block



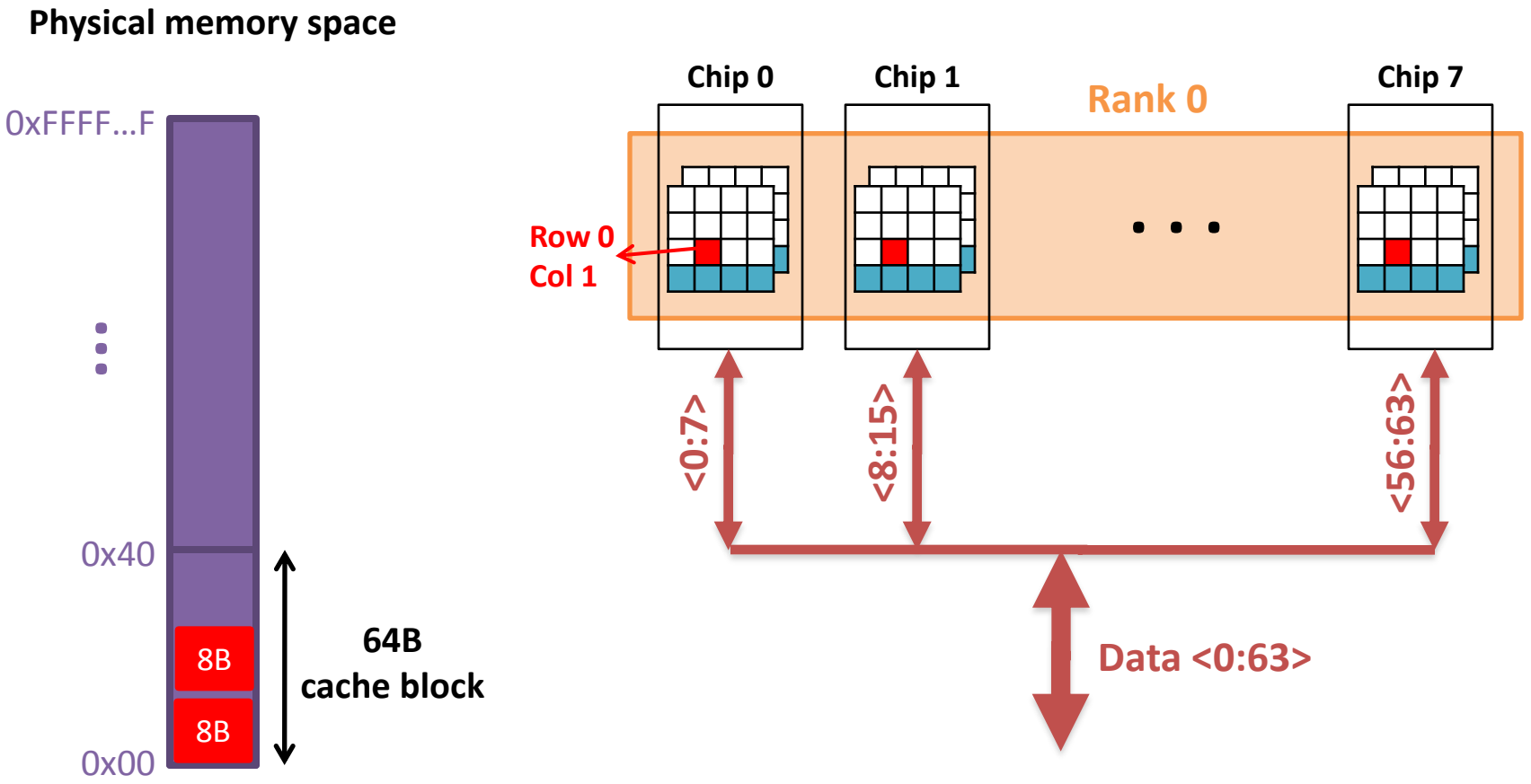
Example: Transferring a cache block



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.

Latency Components: Basic DRAM Operation

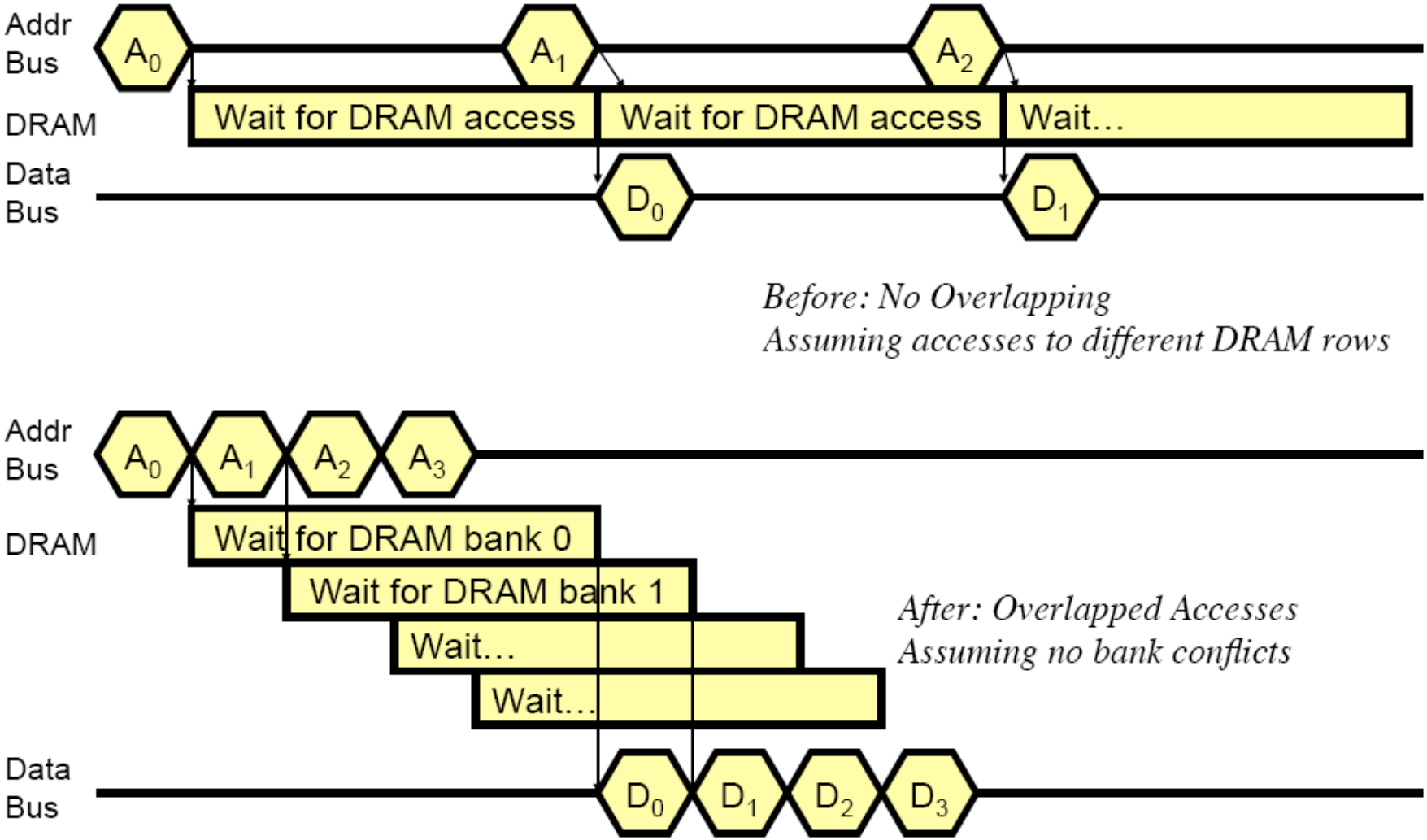
- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS if row is "open" OR
  - RAS + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)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

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



Multiple Channels

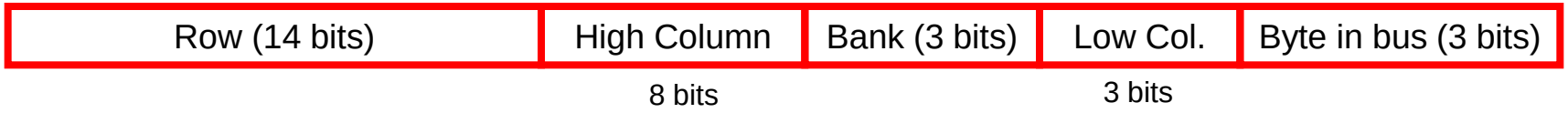
- Advantages
  - Increased bandwidth
  - Multiple concurrent accesses (if independent channels)

- 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

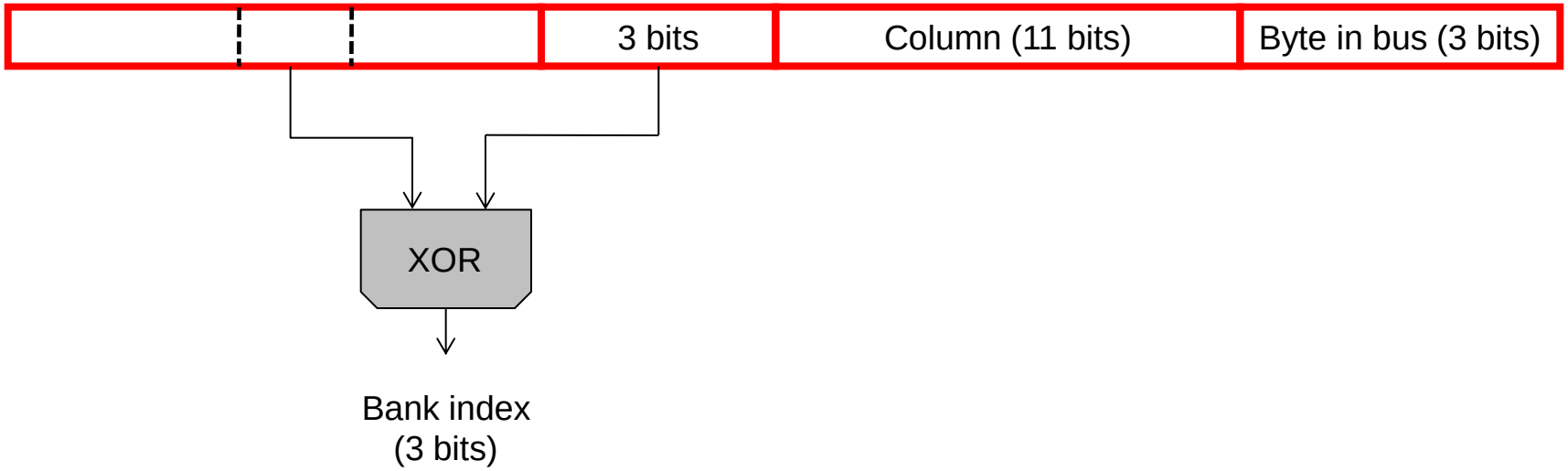
- Cache block interleaving
  - Consecutive cache block addresses in consecutive banks
  - 64 byte cache blocks



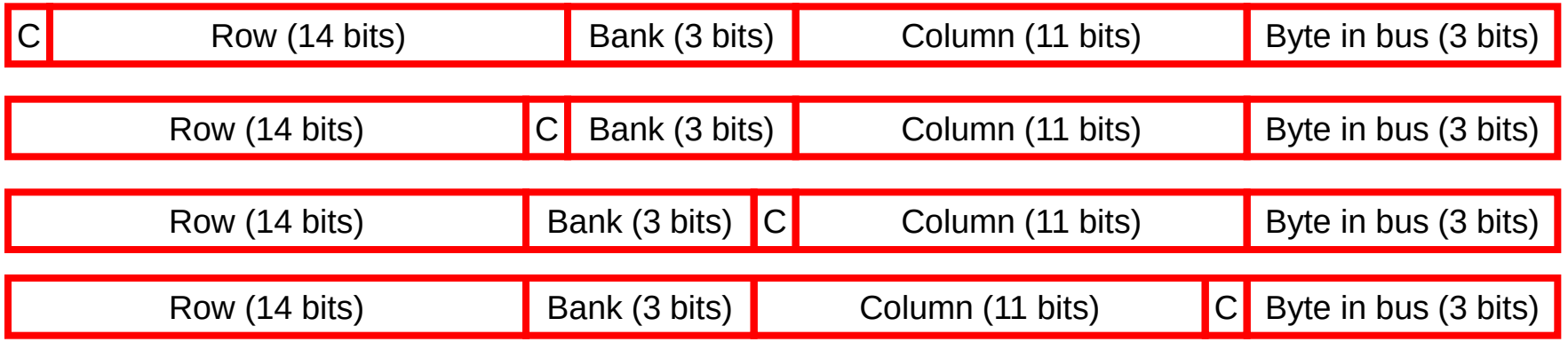
  - Accesses to consecutive cache blocks can be serviced in parallel
  - How about random accesses? Strided accesses?

Bank Mapping Randomization

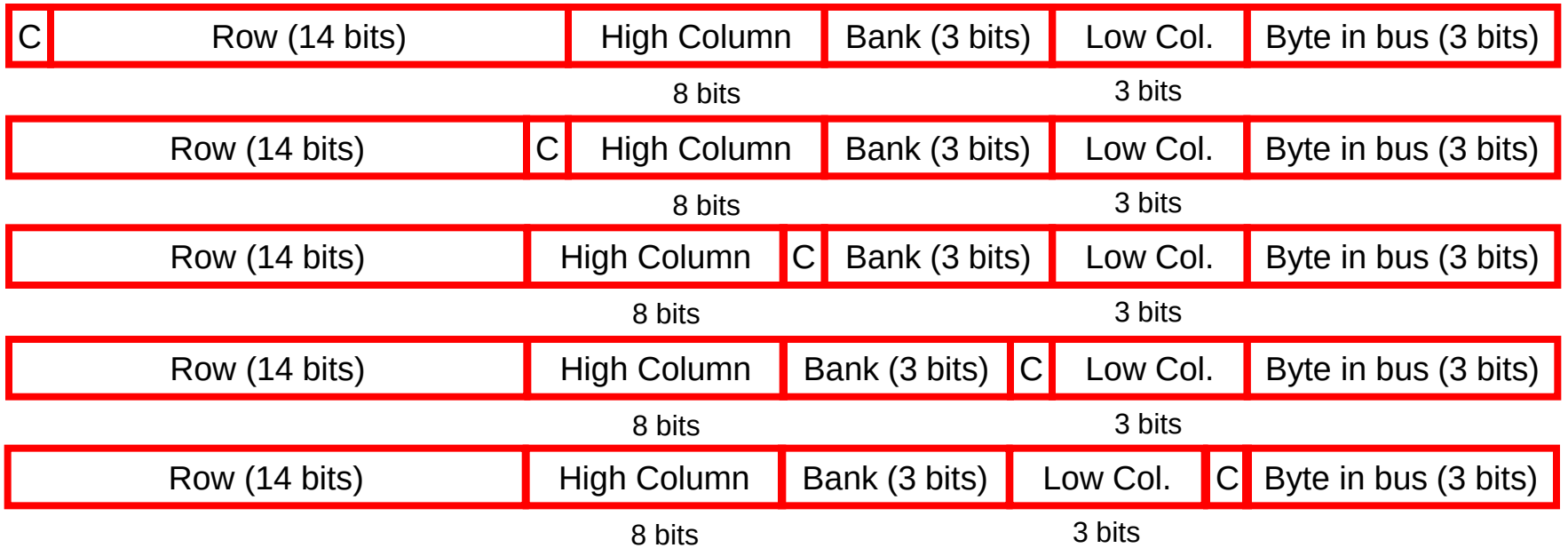
- DRAM controller can randomize the address mapping to banks so that bank conflicts are less likely



Address Mapping (Multiple Channels)

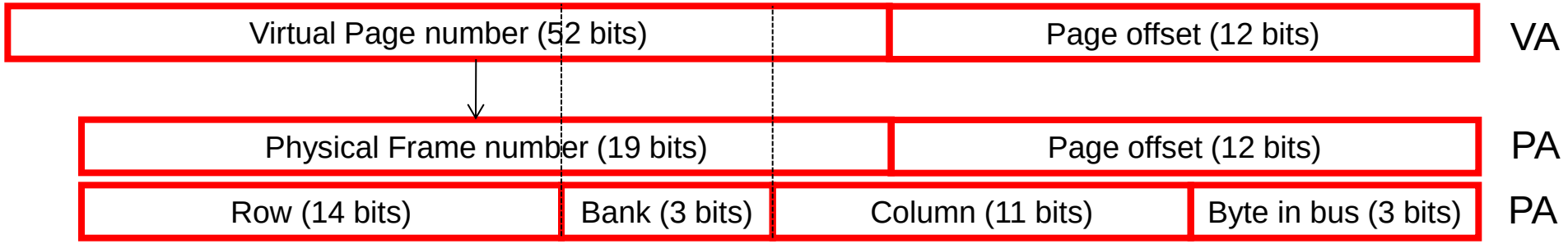


- Where are consecutive cache blocks?



Interaction with Virtual→Physical Mapping

- Operating System influences where an address maps to in DRAM



- Operating system can control which bank/channel/rank a virtual page is mapped to.

- It can perform page coloring to minimize bank conflicts
- Or to minimize inter-application interference