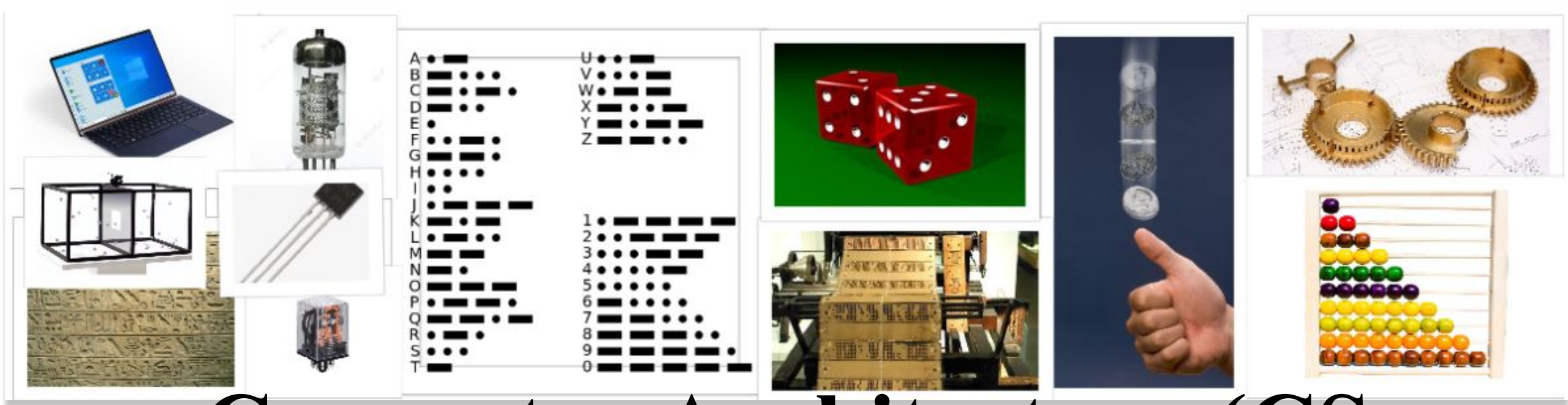


Computer Architecture (CS F342)

Main Memory Organization and Architecture

Review of last class: Cache's Design Parameters

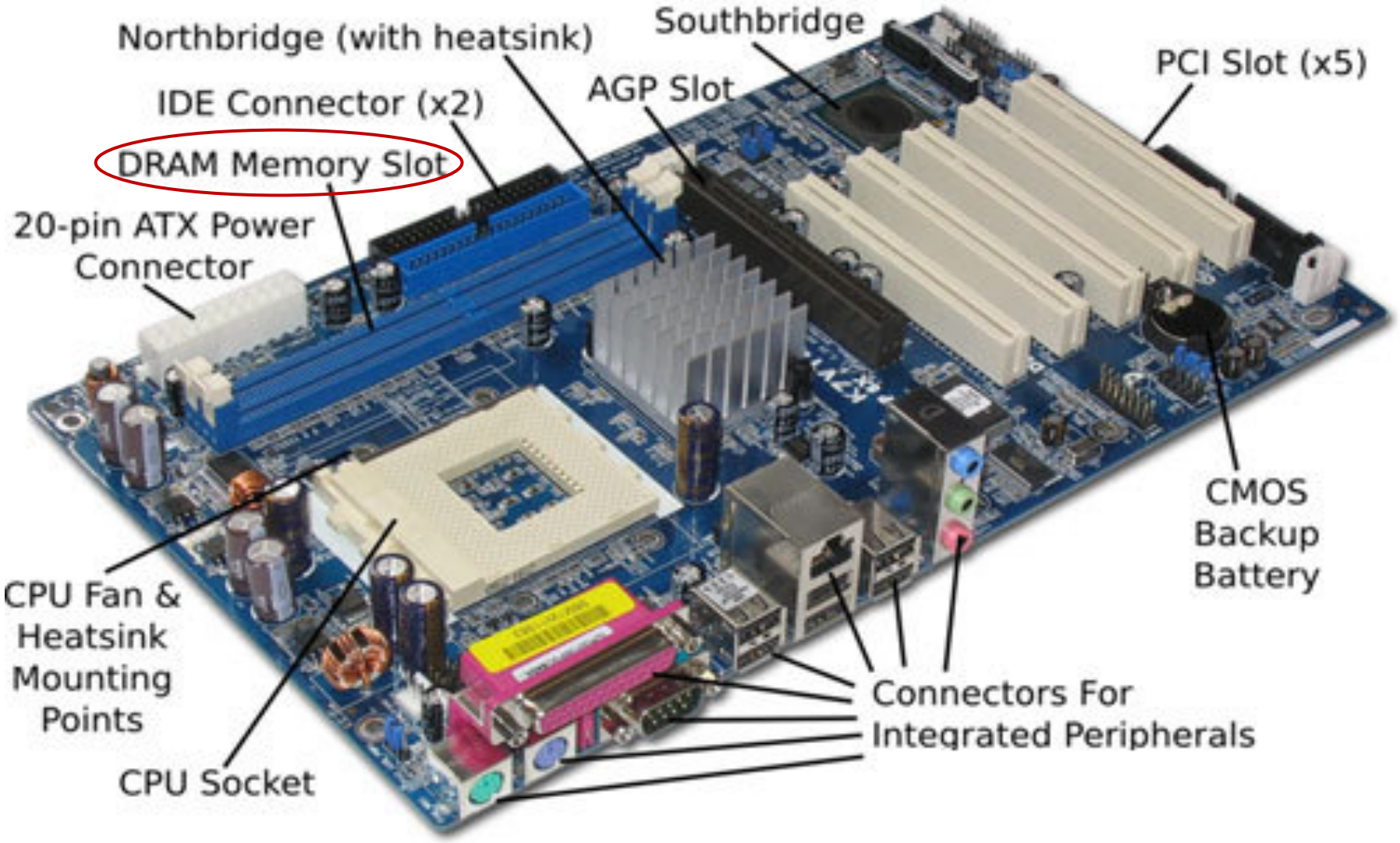
Cache design

- Block size, Associativity
- Block organization
  - Direct-mapped
  - Fully associative
  - Set-associative
- Block replacement policy
  - Random
  - FIFO
  - LRU or Pseudo-LRU
- Write policy
  - Writeback
  - Write-through
    - Write-allocate
    - Write-no-allocate
- Read policy
  - Look aside
  - Look through

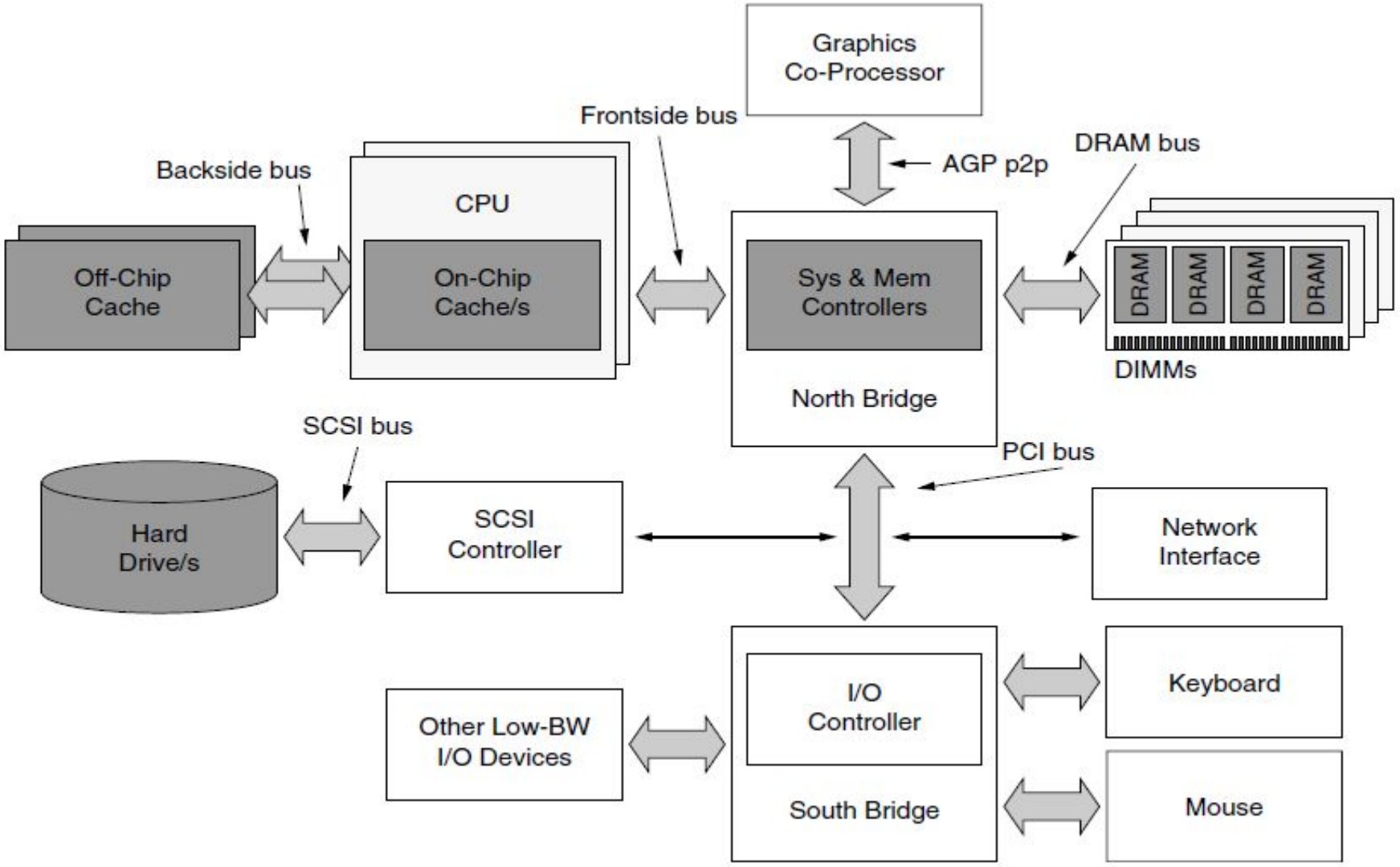
Cache design

- Inclusive cache
- Exclusive cache
- Non-Inclusive cache

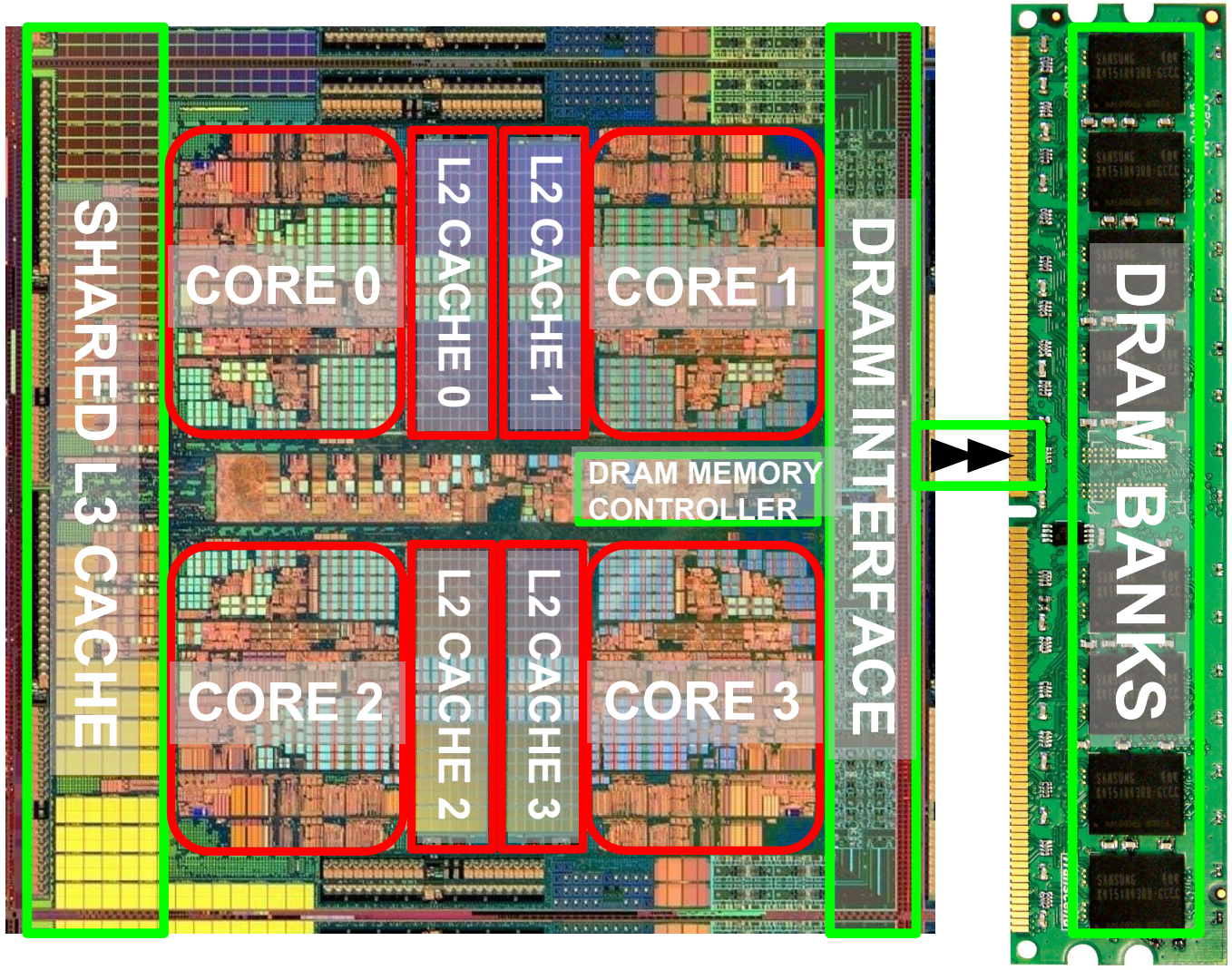
Location of Main Memory in Motherboard



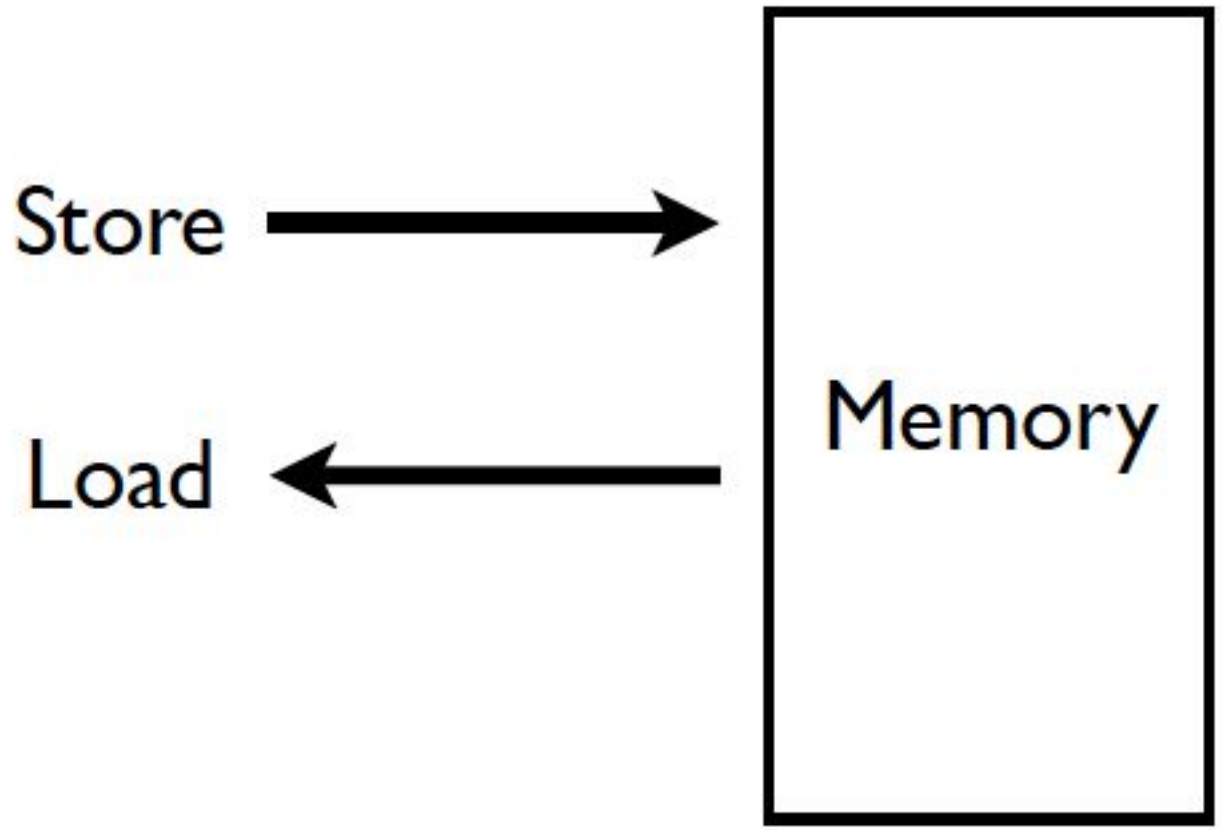
Location of Main Memory in Motherboard



Location of Main Memory



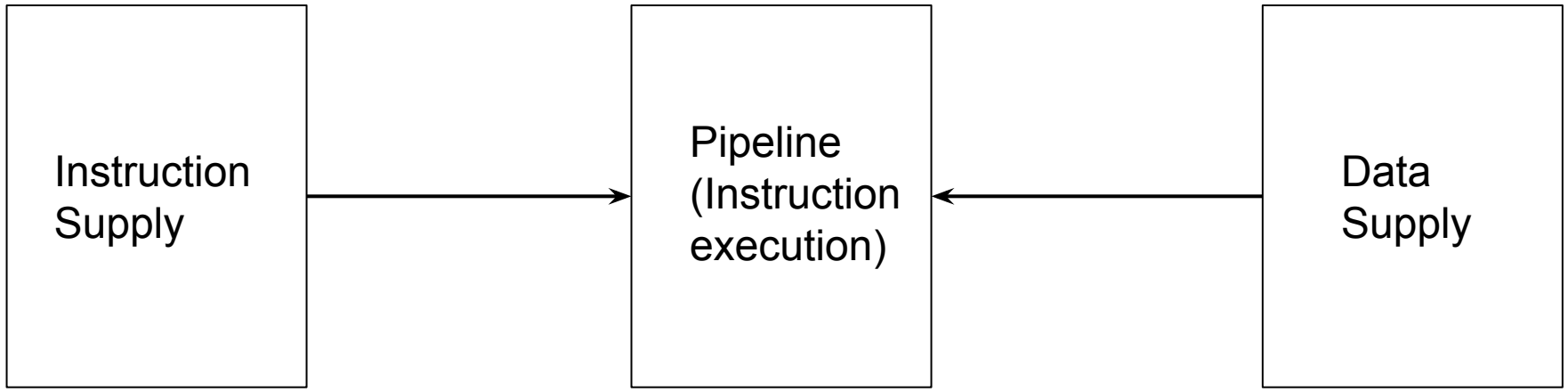
Programmer's view



Memory Idealism

- Zero access time (latency)
- Infinite capacity
- Zero cost
- Infinite bandwidth (to support multiple accesses in parallel)

Idealism



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

 No pipeline stalls

Perfect data flow (reg/memory dependencies)

 Zero-cycle interconnect (operand communication)

 Enough functional units

 Zero latency compute

 Zero latency access

 Infinite capacity

- Infinite bandwidth

 Zero cost

Problem with Memory Idealism

- Ideal memory's requirements oppose each other

- Bigger is slower
  - Bigger Takes longer to determine the location

- Faster is more expensive
  - Memory technology: SRAM vs. DRAM vs. Disk vs. Tape

- Higher bandwidth is more expensive
  - Need more banks, more ports, higher frequency, or faster technology

Problem with Memory Idealism

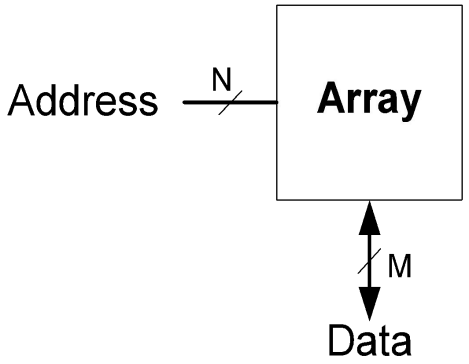
- Bigger is slower
  - SRAM, 512 Bytes, sub-nanosec
  - SRAM, KByte~MByte, ~nanosec
  - DRAM, Gigabyte, ~50 nanosec
  - Hard Disk, Terabyte, ~10 millisec
- Faster is more expensive (dollars and chip area)
  - SRAM, < 10$ per Megabyte
  - DRAM, < 1$ per Megabyte
  - Hard Disk < 1$ per Gigabyte
  - These sample values (circa ~2011) scale with time
- Other technologies have their place as well
  - Flash memory (mature), PC-RAM, MRAM, RRAM (not mature yet)

How does one store data?

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

Array Organization of Memories

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

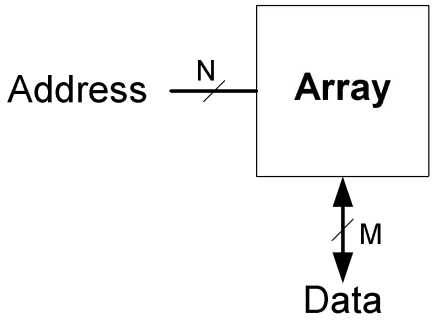


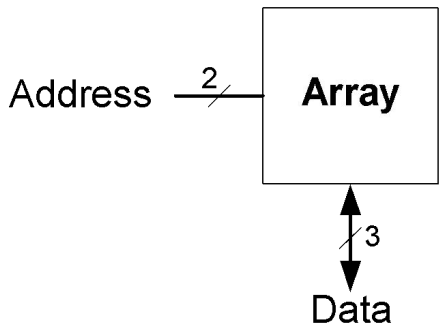
- An M-bit value can be read or written at each unique N-bit address
  - All values can be accessed, but only M-bits at a time
  - Access restriction allows more compact organization

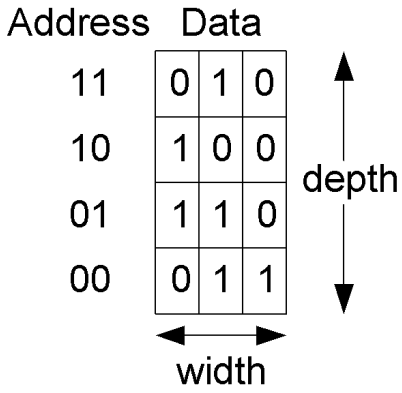
Memory arrays

- Two-dimensional array of bit cells
  - Each bit cell stores one bit

- An array with N address bits and M data bits:
  - 2^N rows and M columns
  - Depth: number of rows (number of words)
  - Width: number of columns (size of word)
  - Array size: depth × width = 2^N × M

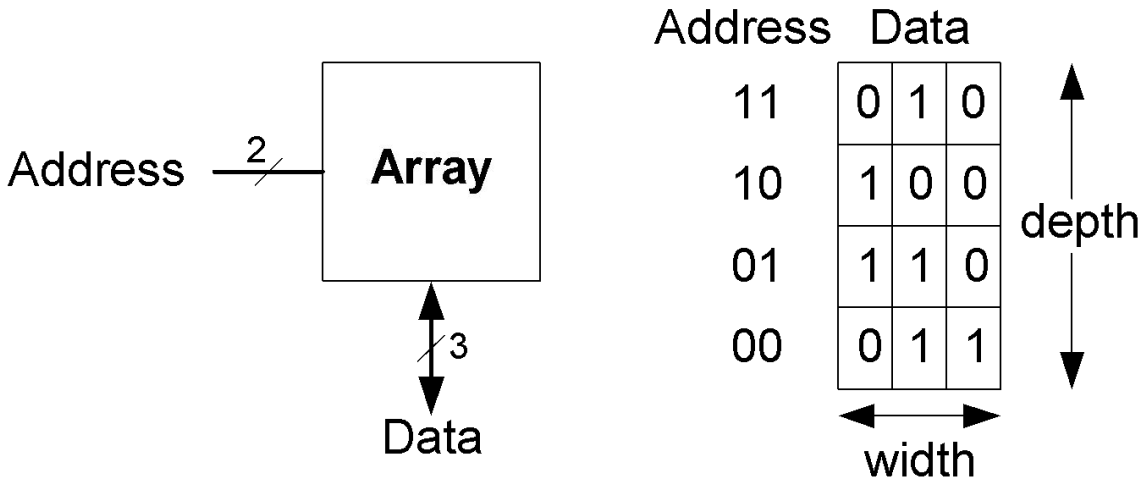




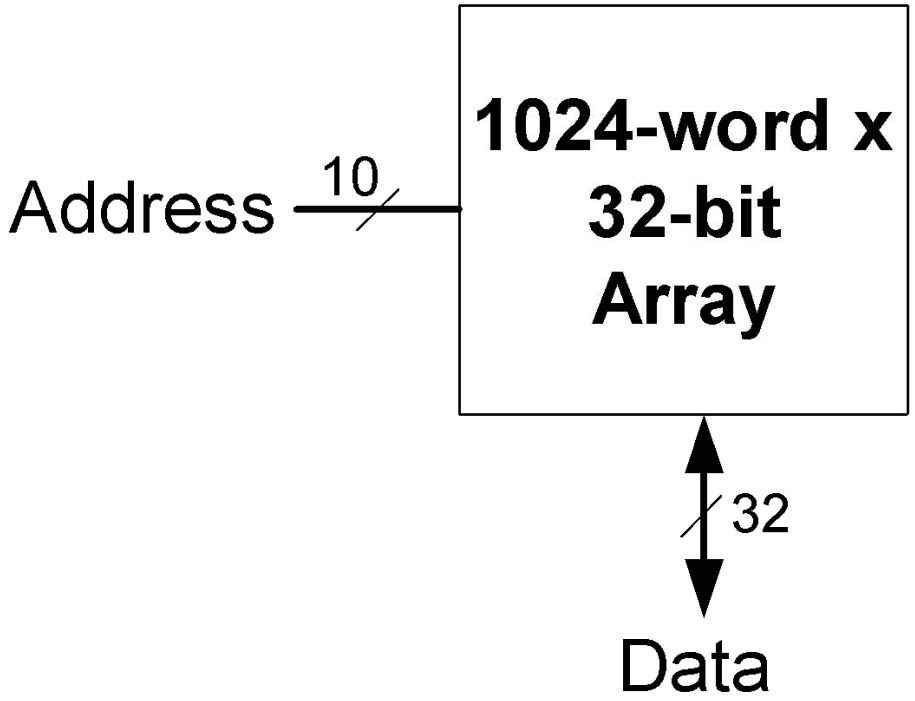


Memory Array Example

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

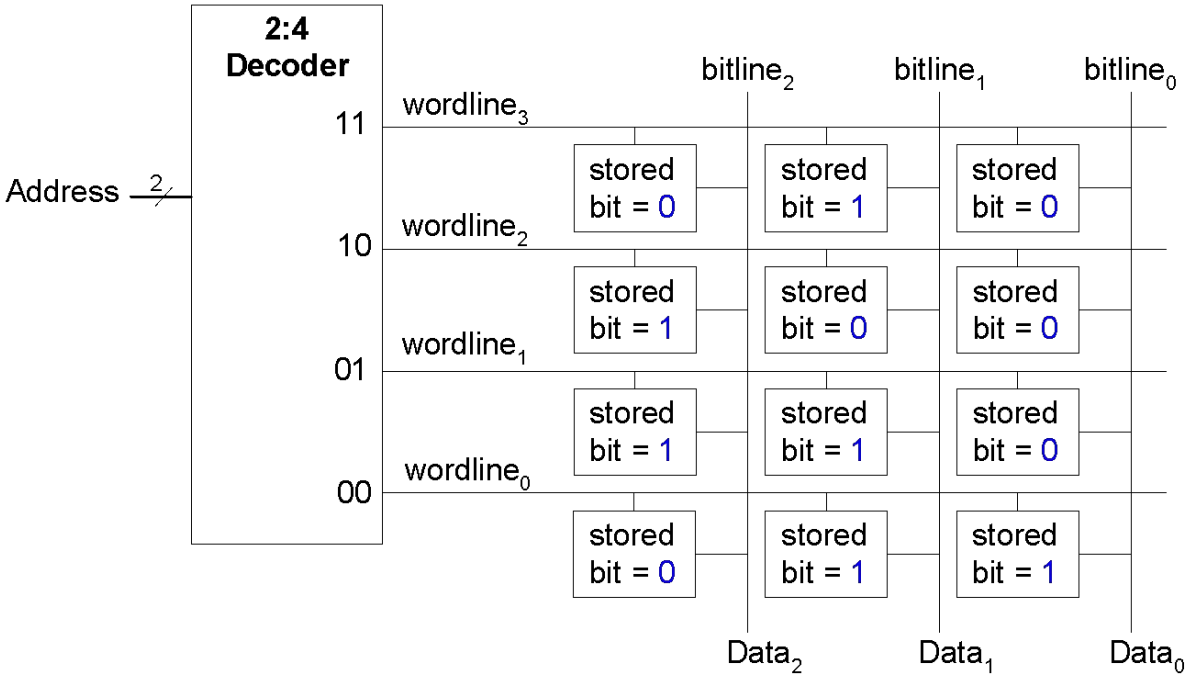


Larger and Wider Memory Array Example



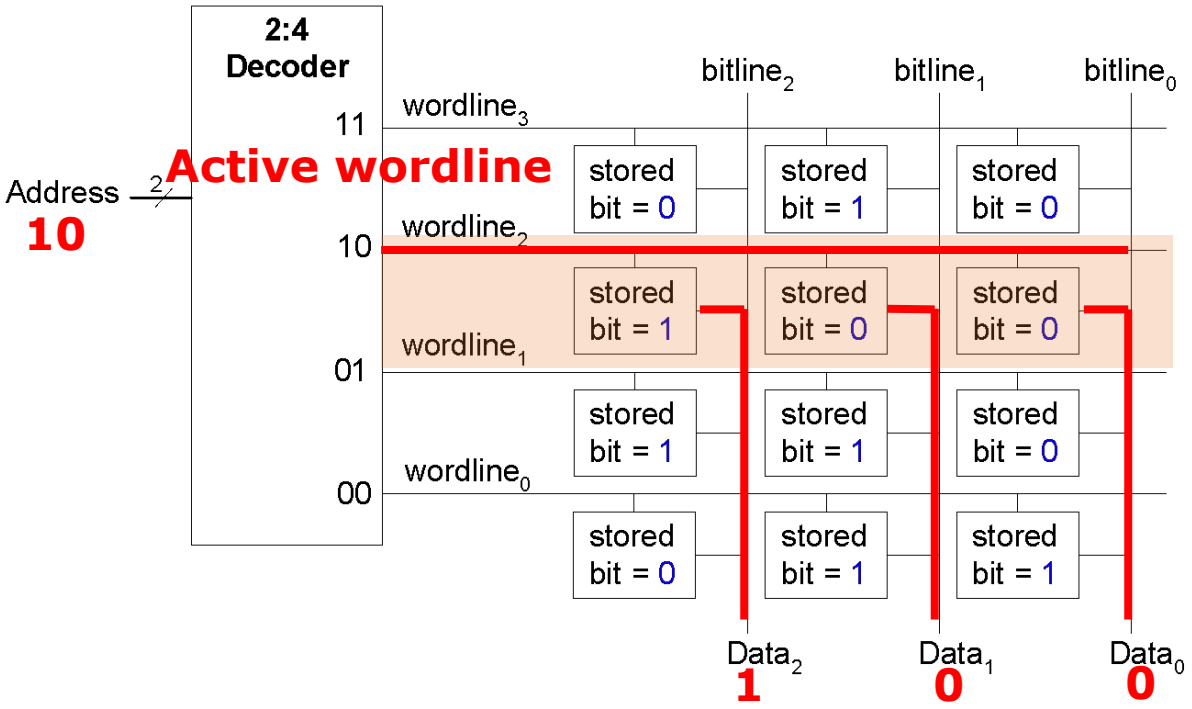
Memory Array Organization

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



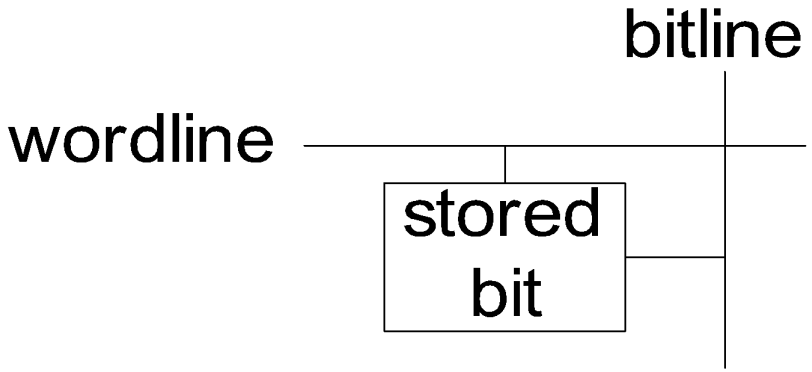
Memory Array Organization

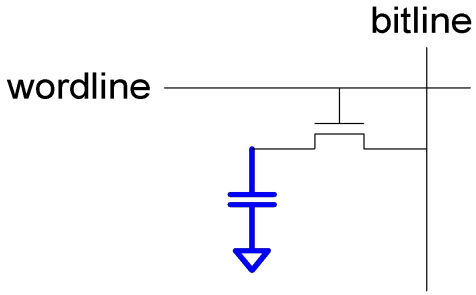
- Storage nodes in one column connected to one bitline
- Address decoder activates only ONE wordline
- Content of one line of storage available at output



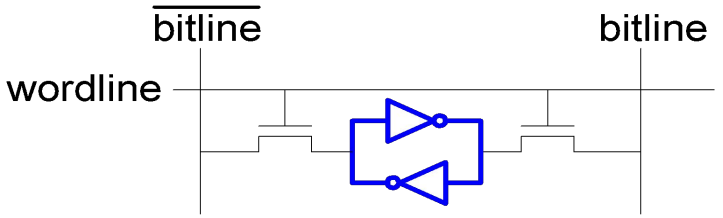
How is Access Controlled?

- Access transistors configured as switches connect the bit storage to the *bitline*
- Access controlled by the *wordline*





DRAM



SRAM

Building Larger Memories

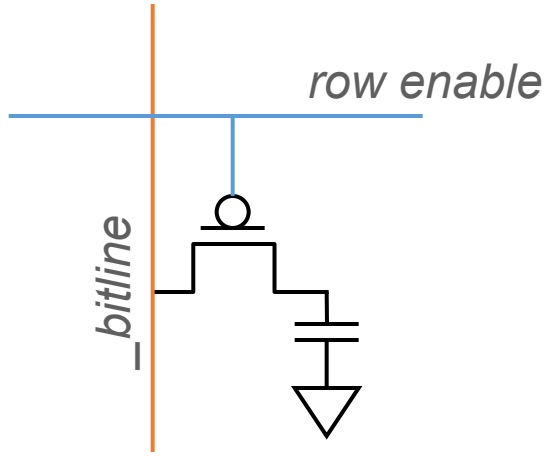
- Requires larger memory arrays
- Large □ slow
- How do we make the memory large without making it very slow?
- Idea: Divide the memory into smaller arrays and interconnect the arrays to input/output buses
  - Large memories are hierarchical array structures
  - DRAM: Channel □ Rank □ Bank □ Subarrays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

General Principle: Interleaving (Banking)

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

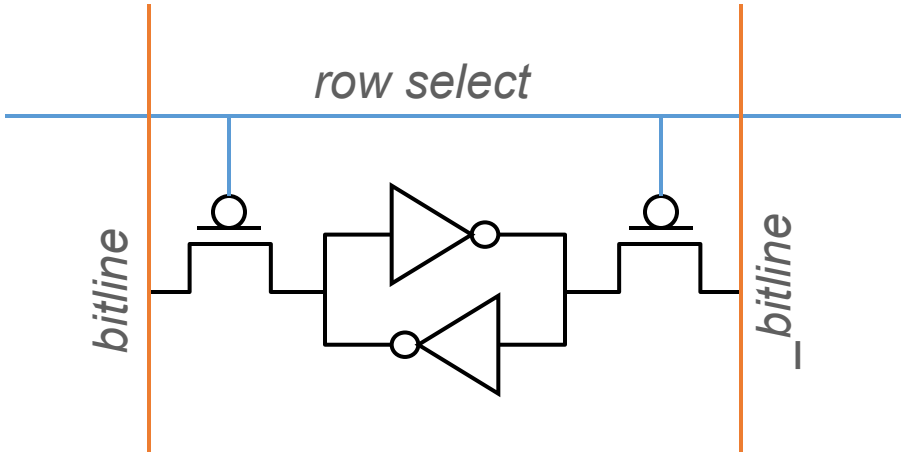
Memory Technology: DRAM

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

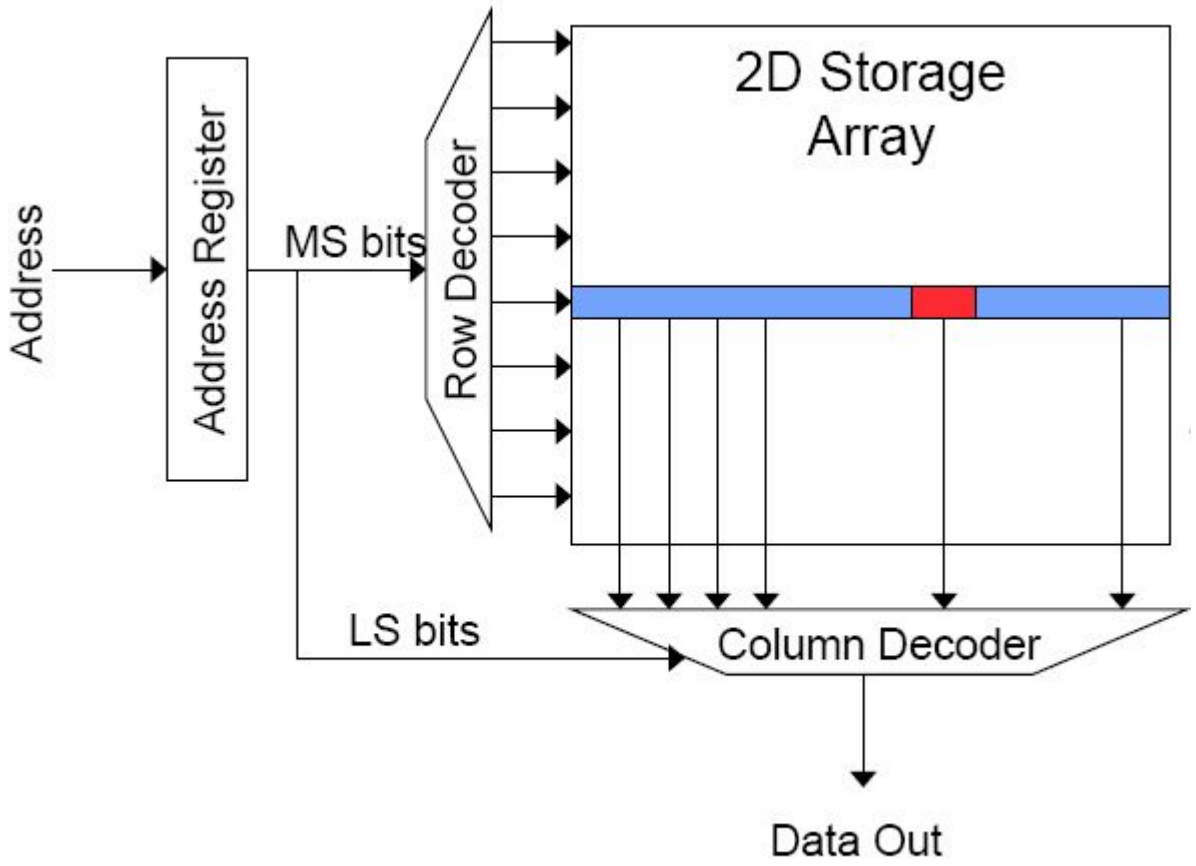


Memory Technology: SRAM

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



Memory Bank Organization and Operation



- Read access sequence:

1. Decode row address & drive word-lines

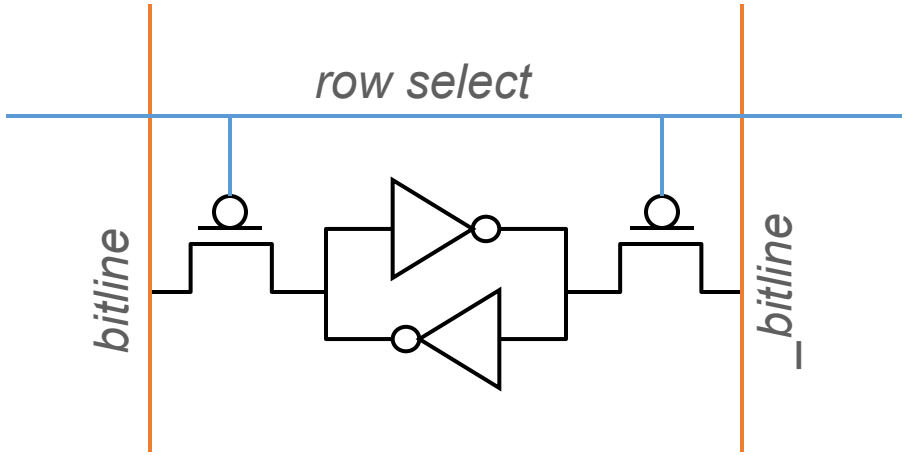
2. Selected bits drive bit-lines
   - Entire row read

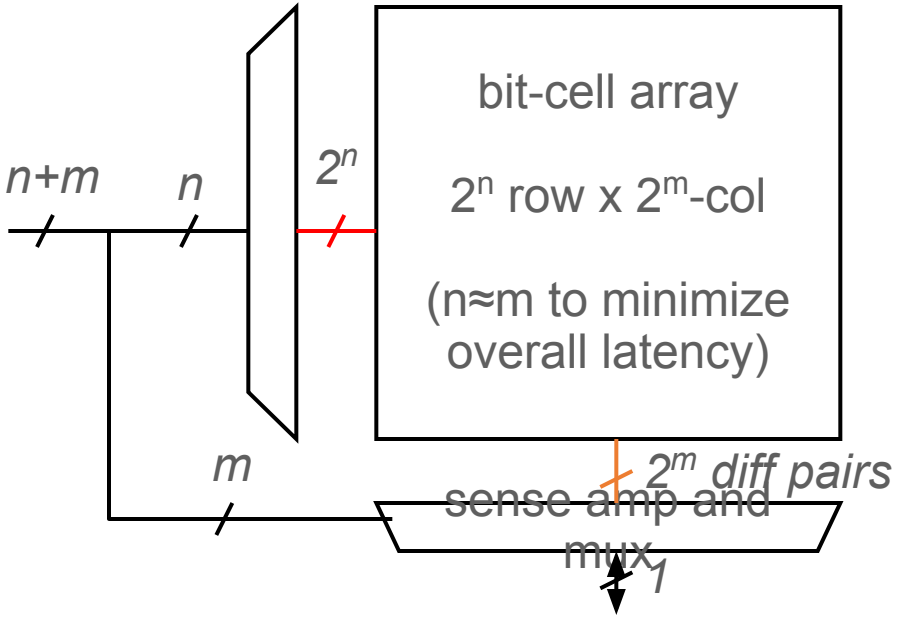
3. Amplify row data

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

5. Precharge bit-lines
   - For next access

SRAM (Static Random Access Memory)





Read Sequence

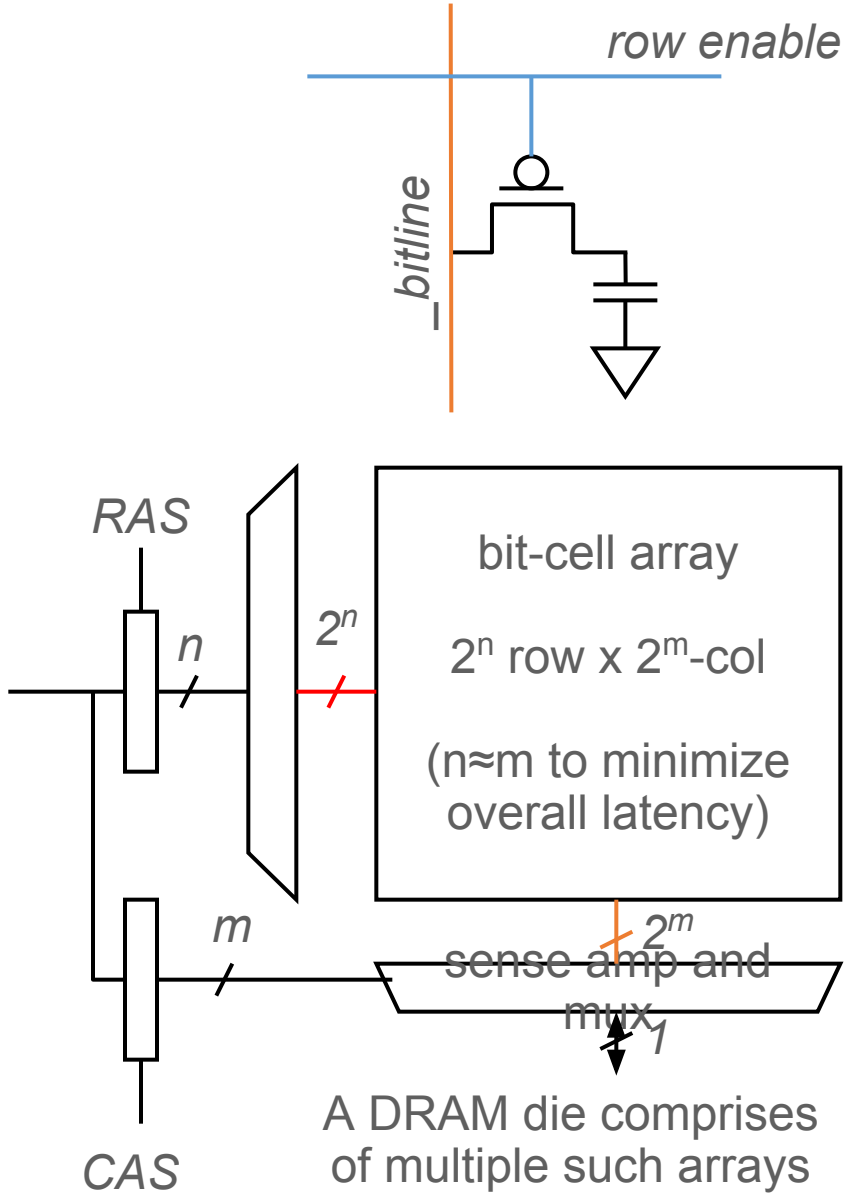
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. differential sensing and column select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

Access latency dominated by steps 2 and 3

Cycling time dominated by steps 2, 3 and 5

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

DRAM (Dynamic Random Access Memory)



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

Destructive reads

Charge loss over time

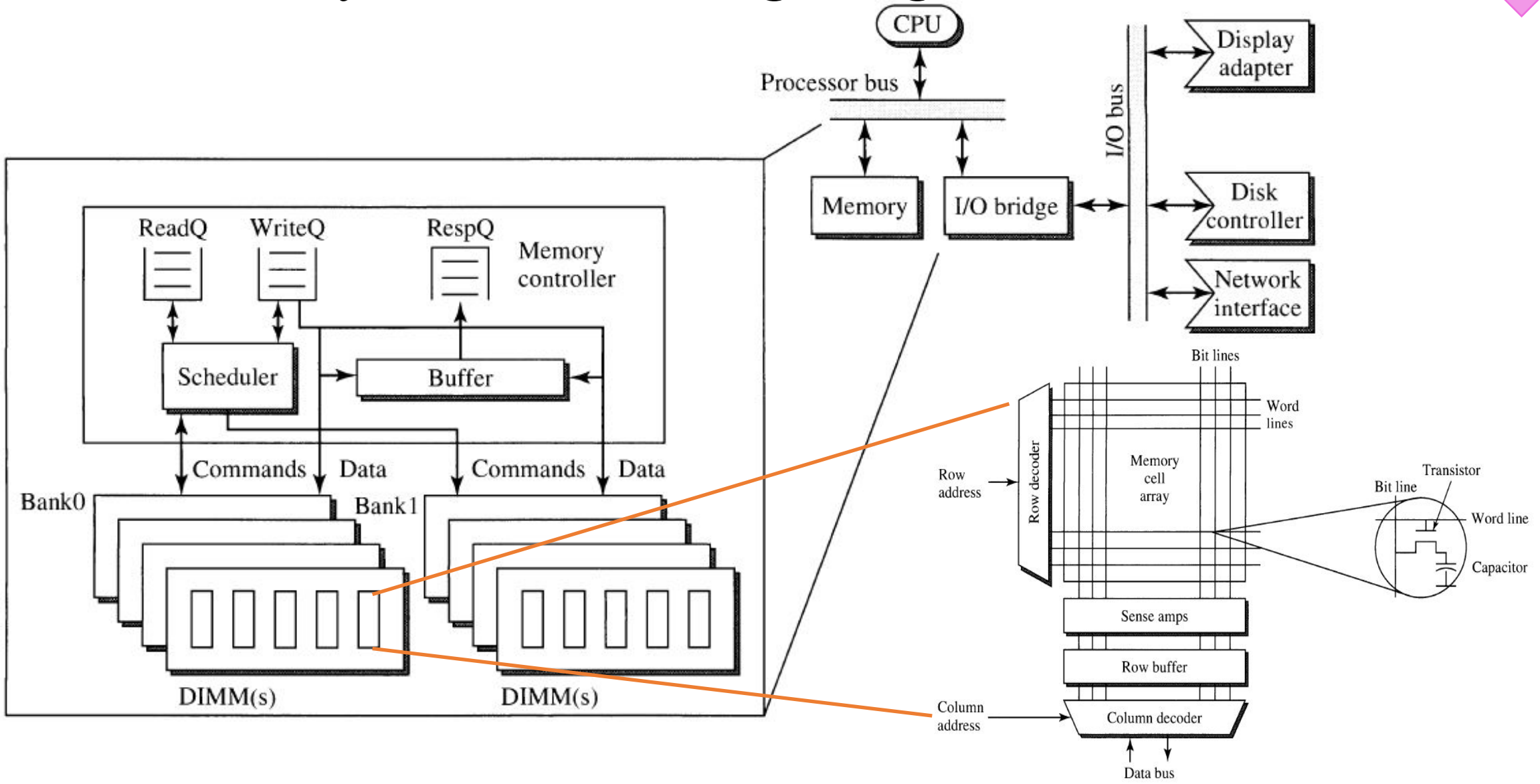
Refresh: A DRAM controller must periodically read each row within the allowed refresh time (10s of ms) such that charge is restored

DRAM vs. SRAM

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

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

Main memory/DRAM storage organization

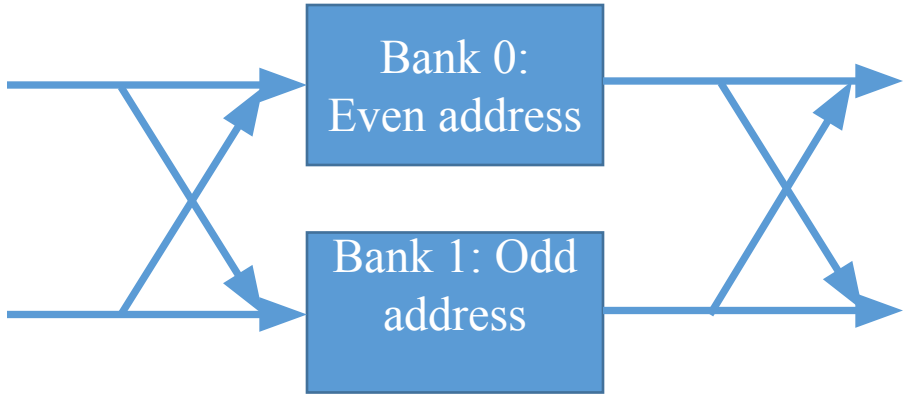


General Principle: Interleaving (Banking)

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

Handling multiple access per cycle

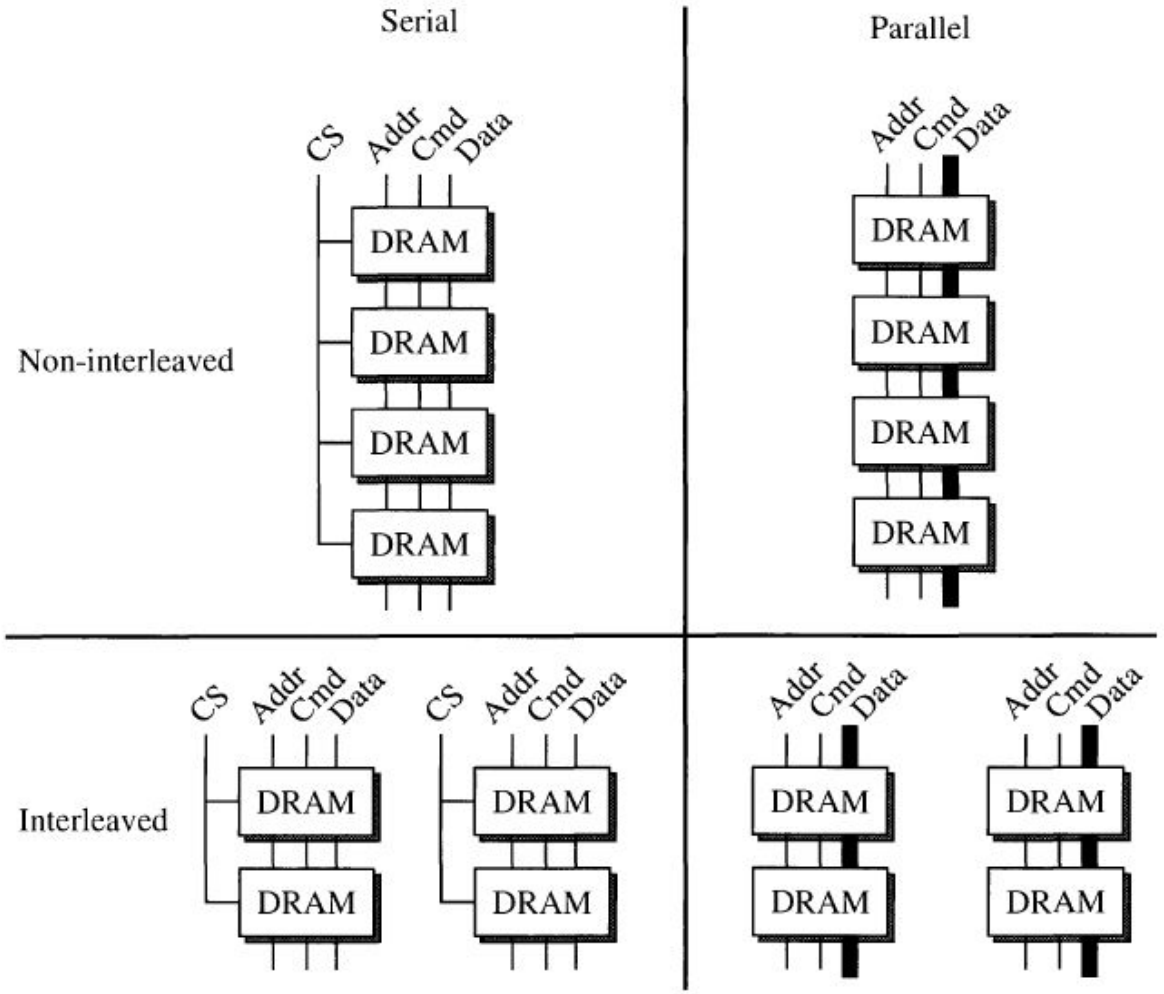
- Interleaving (banking)
  - Address space partitioned into separate banks
  - No increase in data store area
  - Bits in address determines which bank an address maps to
  - Cannot satisfy multiple access to the same bank
  - Crossbar interconnect input-output
  - Bank conflict: more than access to the same difficult to handle



Memory Module Organization

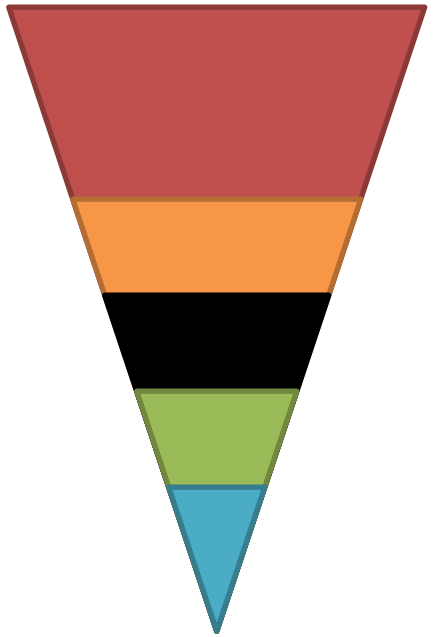
Four options:

- Serial Vs. Parallel
- Non-interleaved Vs. Interleaved

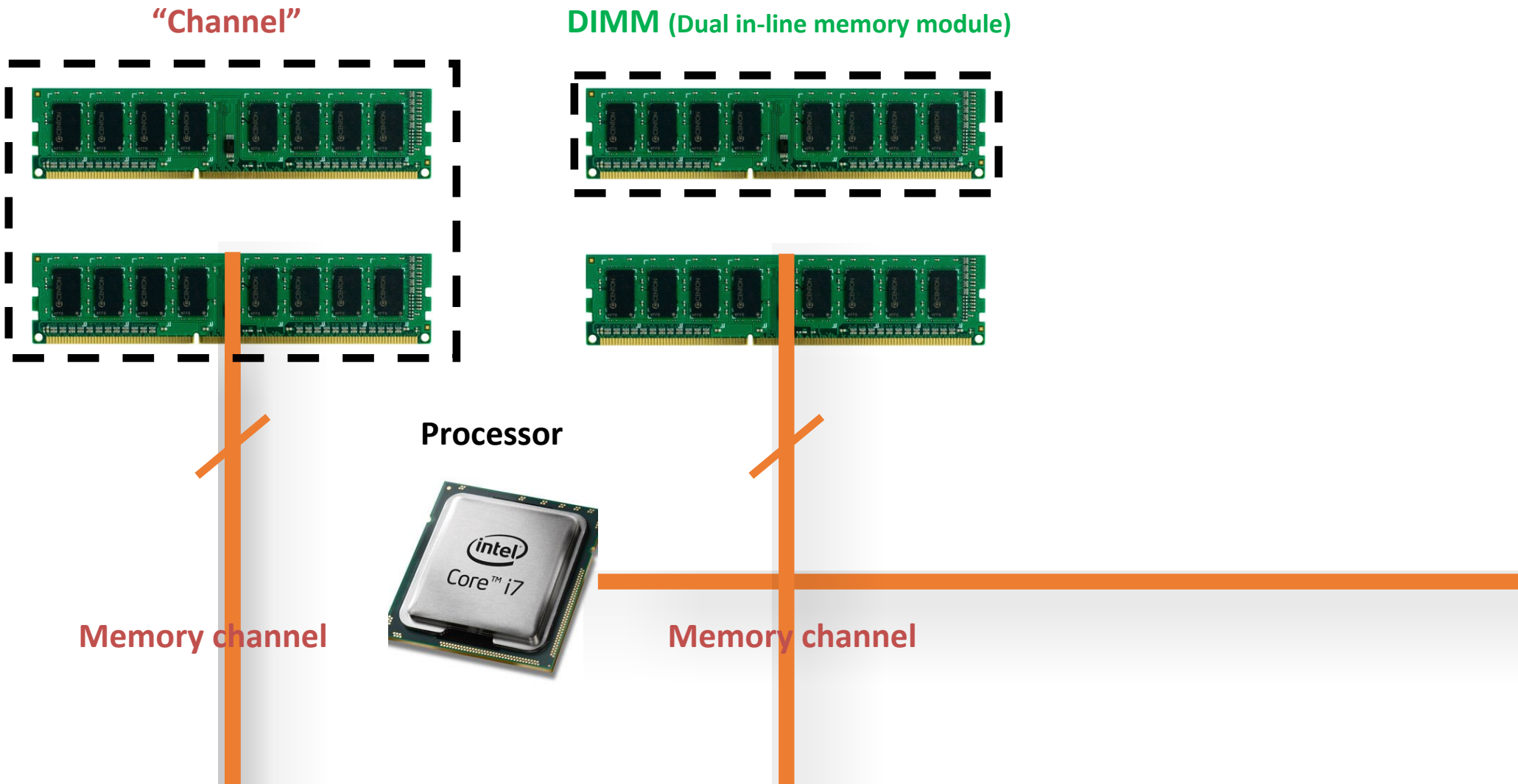


DRAM Subsystem Organization

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



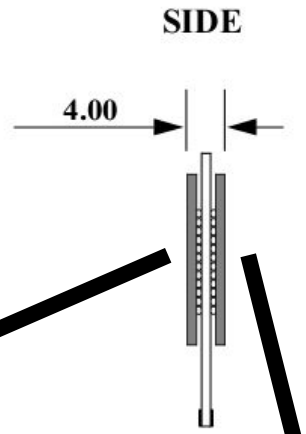
The DRAM subsystem



Breaking down a DIMM

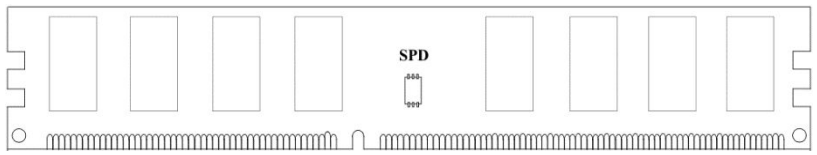
DIMM (Dual in-line memory module)

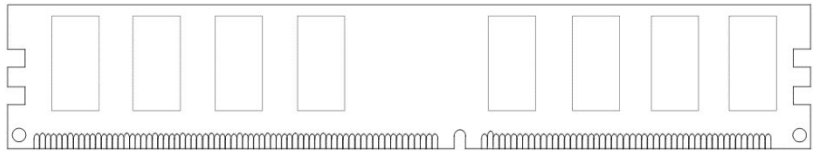
Side view



Front of DIMM

Back of DIMM

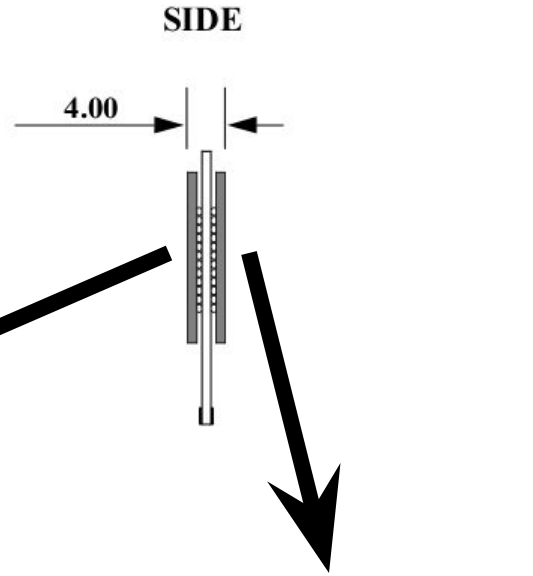




Breaking down a DIMM

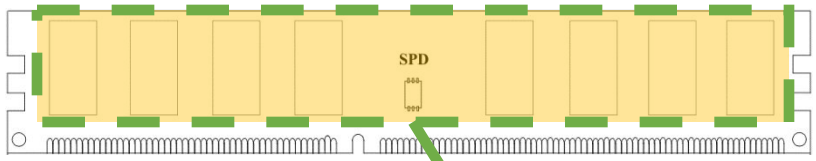
DIMM (Dual in-line memory module)

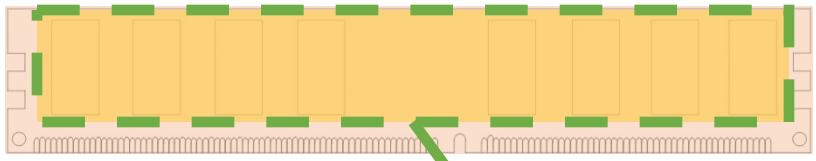
Side view



Front of DIMM

Back of DIMM

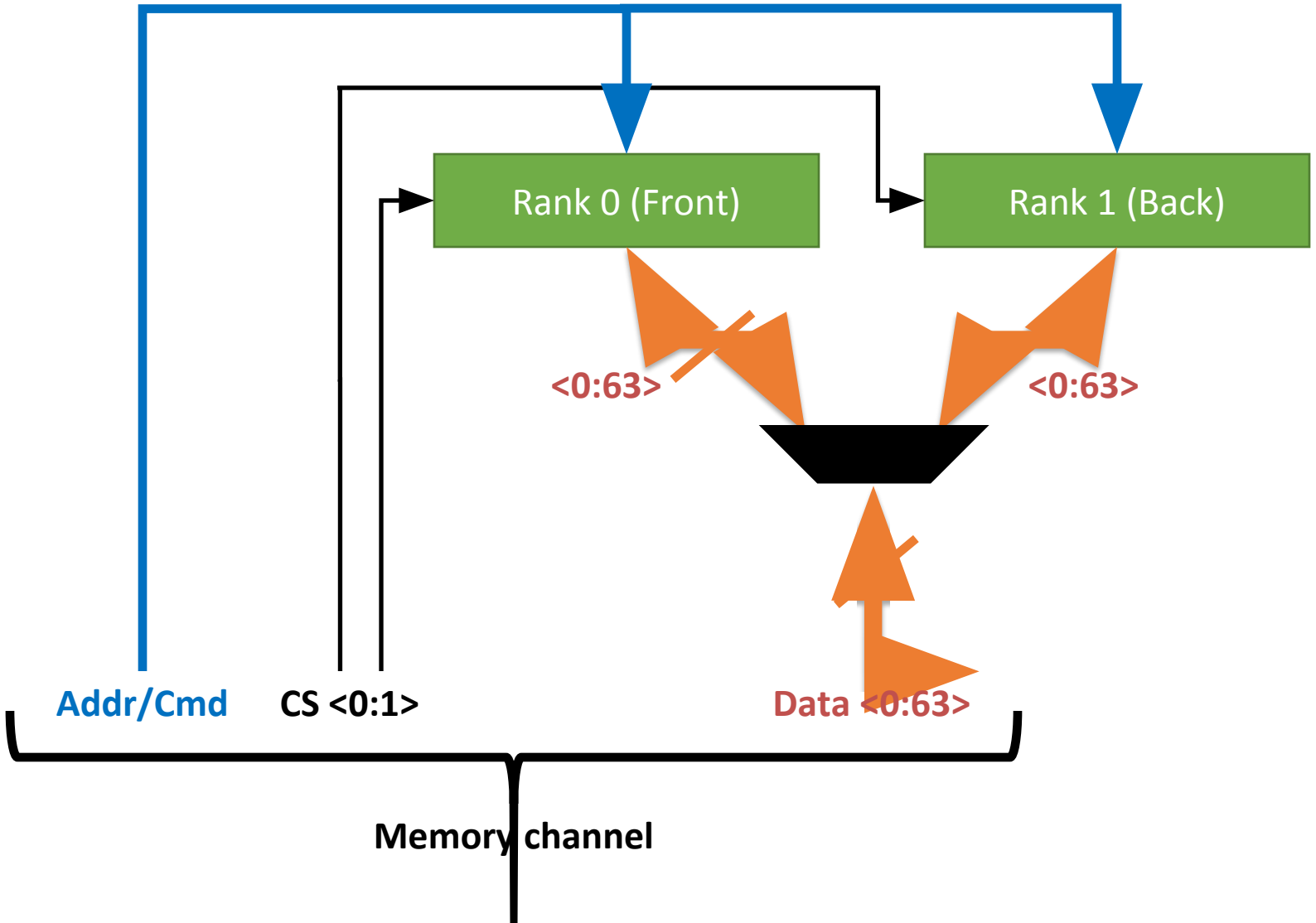




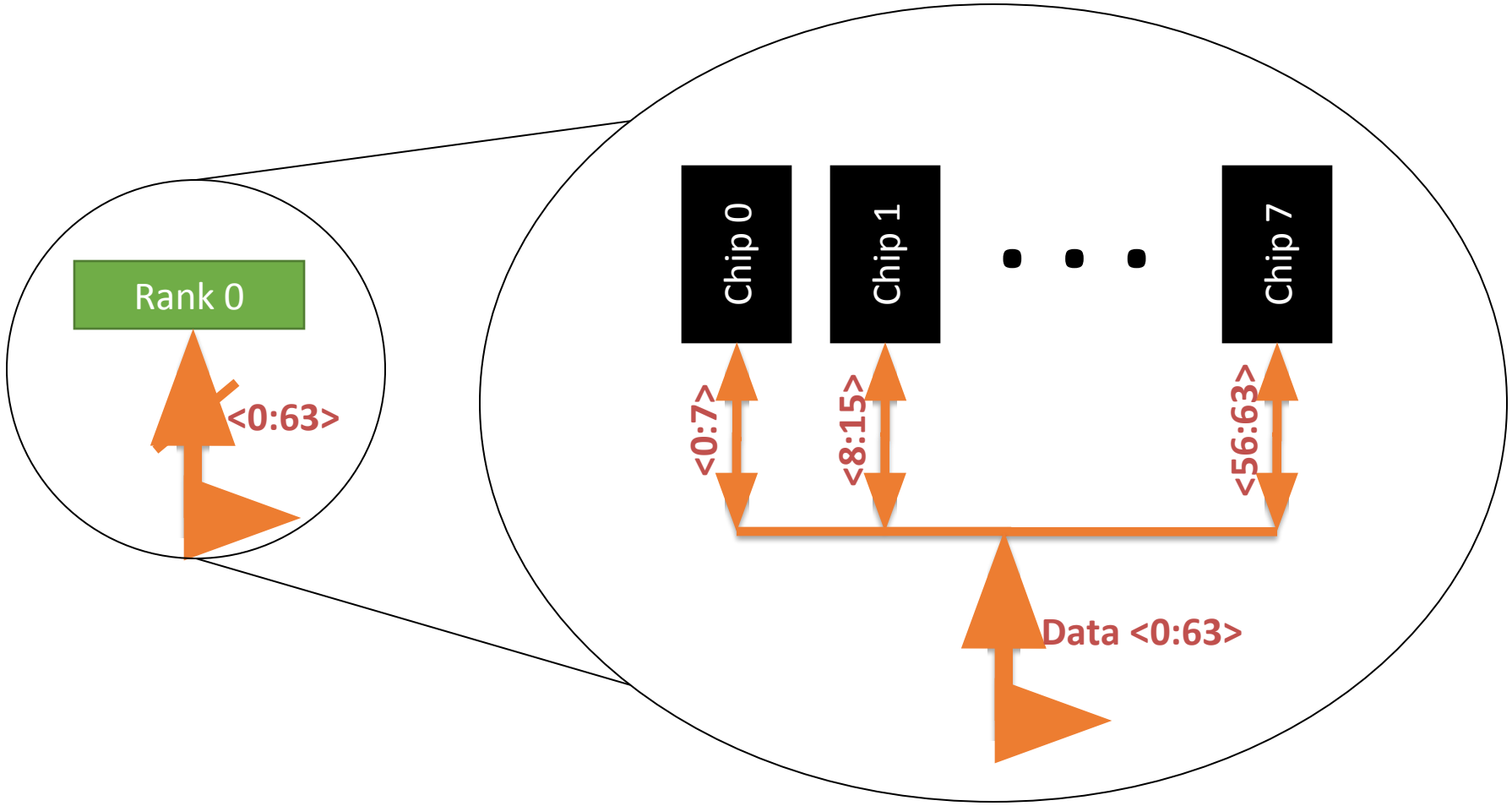
Rank 0: collection of 8 chips

Rank 1

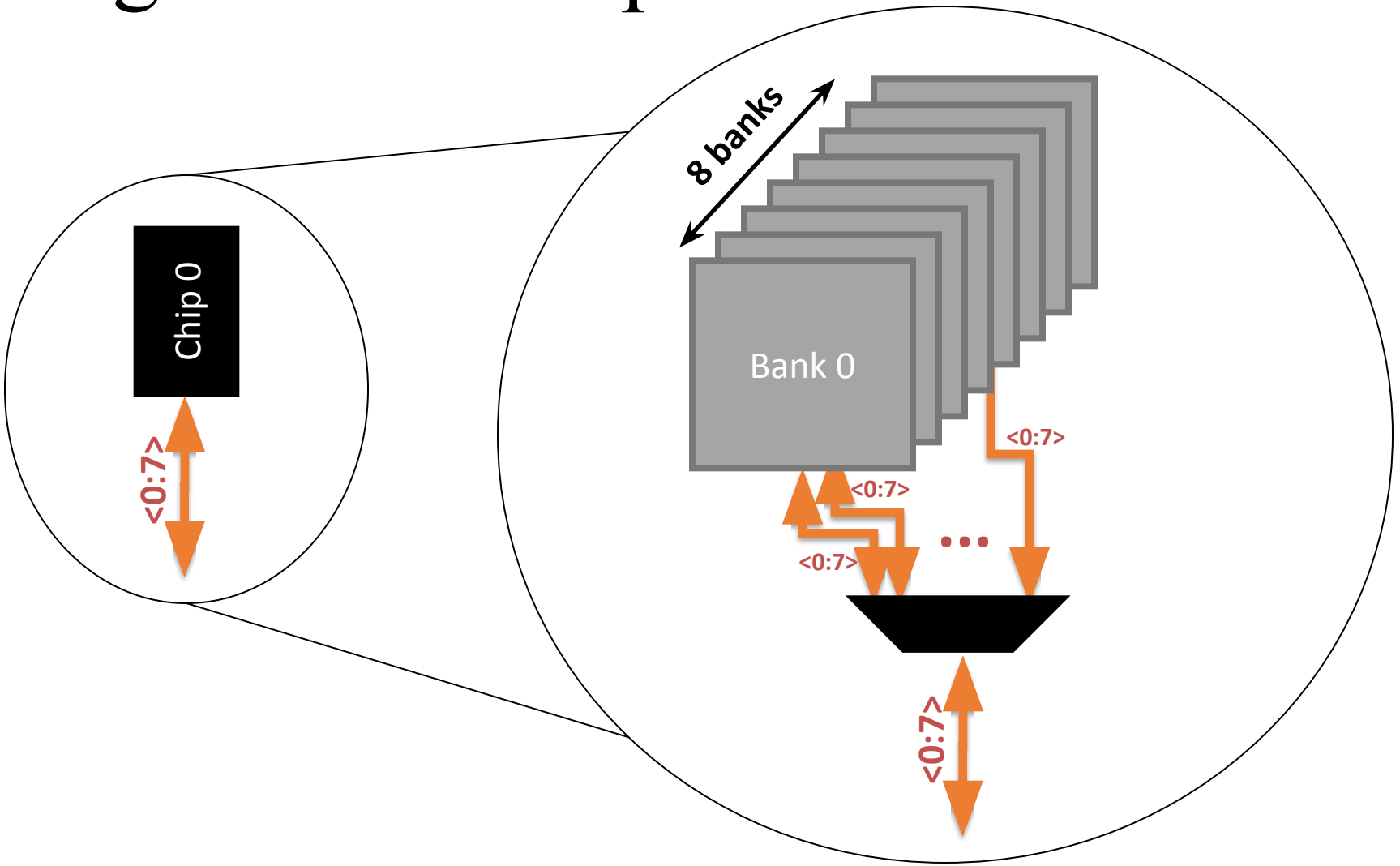
Rank


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

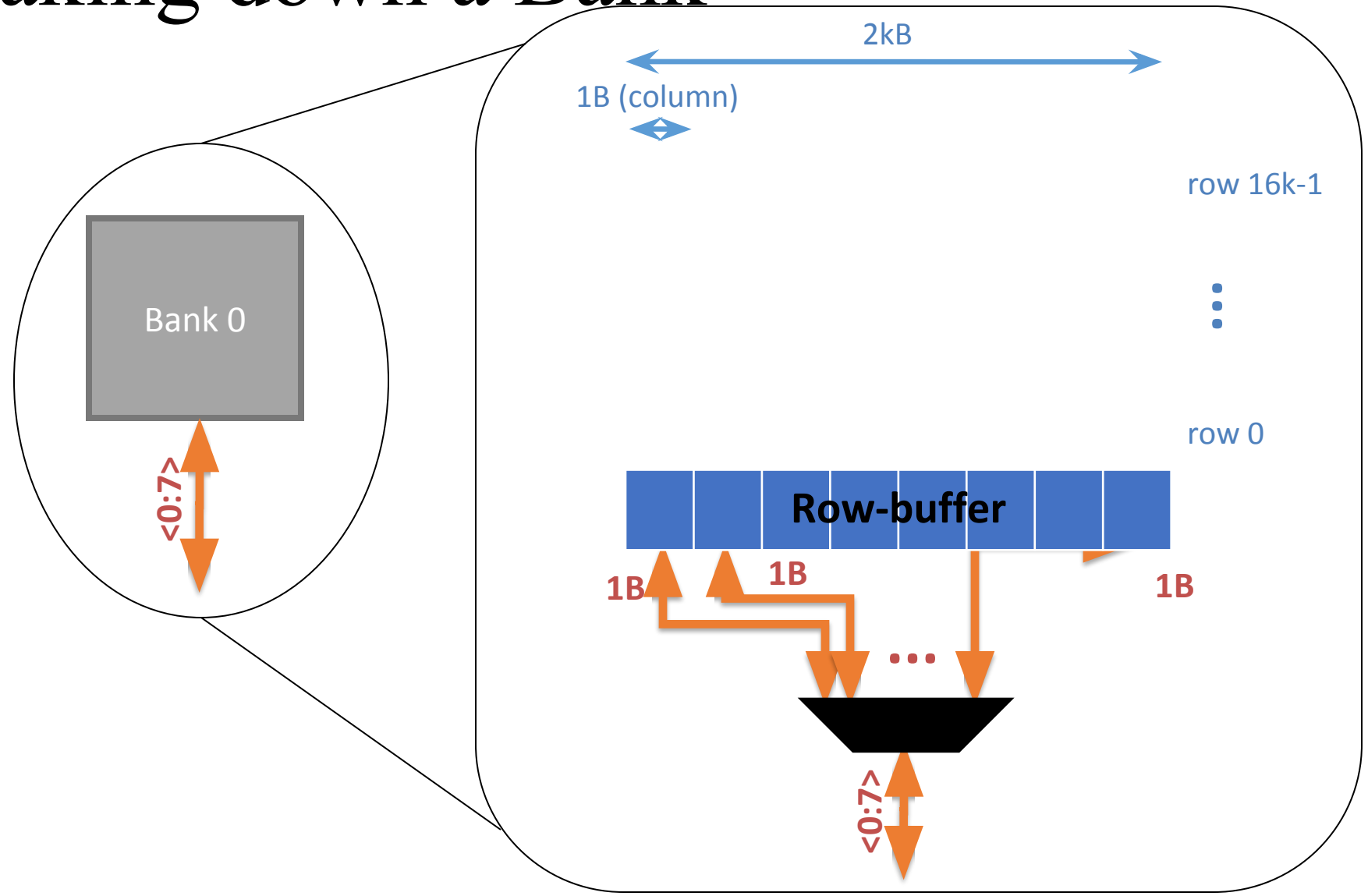
Breaking down a Rank



Breaking down a Chip

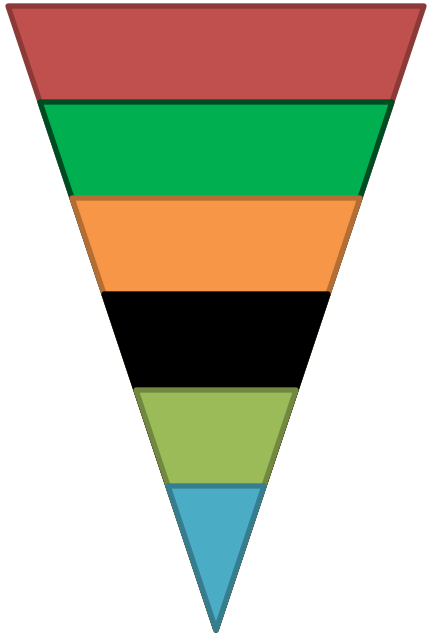


Breaking down a Bank



DRAM Subsystem Organization

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



Why do we need such a complex organization?
Provides bandwidth through pipeline requests or parallel access
Concurrency at rank-level & bank-level

Example

- Consider a 2-rank DIMM with each rank having four 8-Gb devices in it and each device implements eight x 16-sub-banks. What is the size of a row, if a bank has 32K rows in it?

- The capacity of each rank is (4 x 8)-Gb= 32-Gb
- The capacity of DIMM is 32 x 2-Gb= 64-Gb
- Total no. of banks in a rank is 8. A bank is equally distributed among all the four devices.
- The capacity of each bank is 32/8 Gb = 4-Gb
- The total number of rows in a bank is 32K, the row size is 4-Gb/32K =16KB

Basic DRAM Operations

- Access lines are multiplexed (?)
  - Send row address if *Row Address Strobe* (RAS) is asserted
  - Send column address if *Column Address Strobe* (CAS) is asserted
- To access data
  - Pre-charge (PRE): place the bank in a voltage state that is suitable for activation
  - Active (ACT): reads the entire row of bits into the row-buffer/sense amplifiers
  - Column Access (Rd/Wr): transfer data to/from the memory controller
- To retain data
  - Restore: DRAM reads are self-destructive
  - Refresh (REF): DRAM cells lose charge due to leakage
    - Periodic charge is needed (open the row & close it)
    - Retention: 64-ms for <85-deg-C and 32-ms for 85-95-deg-C temperature

Row-buffer Management Policy

- Row hit: if the requested row is in the row-buffer
- Row miss or conflict: if the requested row is not in the row-buffer
- Open-page row-buffer policy
  - Read latency can be either time taken by CAS(R) or time taken by the operation PRE (P), ACT (A) and CAS(R/W)[Column read/column write]
  - Better for high access locality
- Closed-page row-buffer policy
  - Read latency is the time taken by operations ACT and CAS(R)
  - Better for low access locality
- Hybrid row-buffer policy
  - Use a timer to control the sense amplifier
  - Timer is set to a predefine value when a row is activated or re-accessed
  - PRE issued if timer reaches 0
  - Better for dynamic access locality

Example

Time	Mem Read Req.	Page status	Memory Commands Issued		
			Open-page	Closed-page	Hybrid (50)
0	Row-0, Col-4	Empty	A, R	A, R	A, R
40	Row-0, Col-5	Hit	R	A, R	R
70	Row-1, Col-3	Miss	P, A, R	A, R	P, A, R
140	Row-2, Col-1	Miss	P, A, R	A, R	A, R

Conversion of requests to commands & Ordering

- Memory request: load (LW) & store (SW)
- Memory command: PRE(P), ACT(A) and CAS (R/W)[column read/column write]

S.No	Memory request
RQ1	LW R1, addrs-1 (Row-0, Col-4)
RQ2	LW R2, addrs-2 (Row-0, Col-5)
RQ3	LW R3, addrs-3 (Row-1, Col-1)
RQ4	SW R5, addrs-4 (Row-1, Col-3)

Are we missing any commands?
Refresh & etc.

Open page policy is maintained

O1: RQ1, RQ2, RQ3, RQ4

S.No	Memory Cmd
RQ1	P, A, R
RQ2	R
RQ3	P, A, R
RQ4	W

Total Cmd: 8

O2: RQ1, RQ2, RQ4, RQ3

S.No	Memory Cmd
RQ1	P, A, R
RQ2	R
RQ4	P, A, W
RQ3	R

Total Cmd: 8
Store-to-load latency

O3: RQ1, RQ4, RQ3, RQ2

S.No	Memory Cmd
RQ1	P, A, R
RQ4	P, A, W
RQ3	R
RQ2	P, A, R

Total Cmd: 10

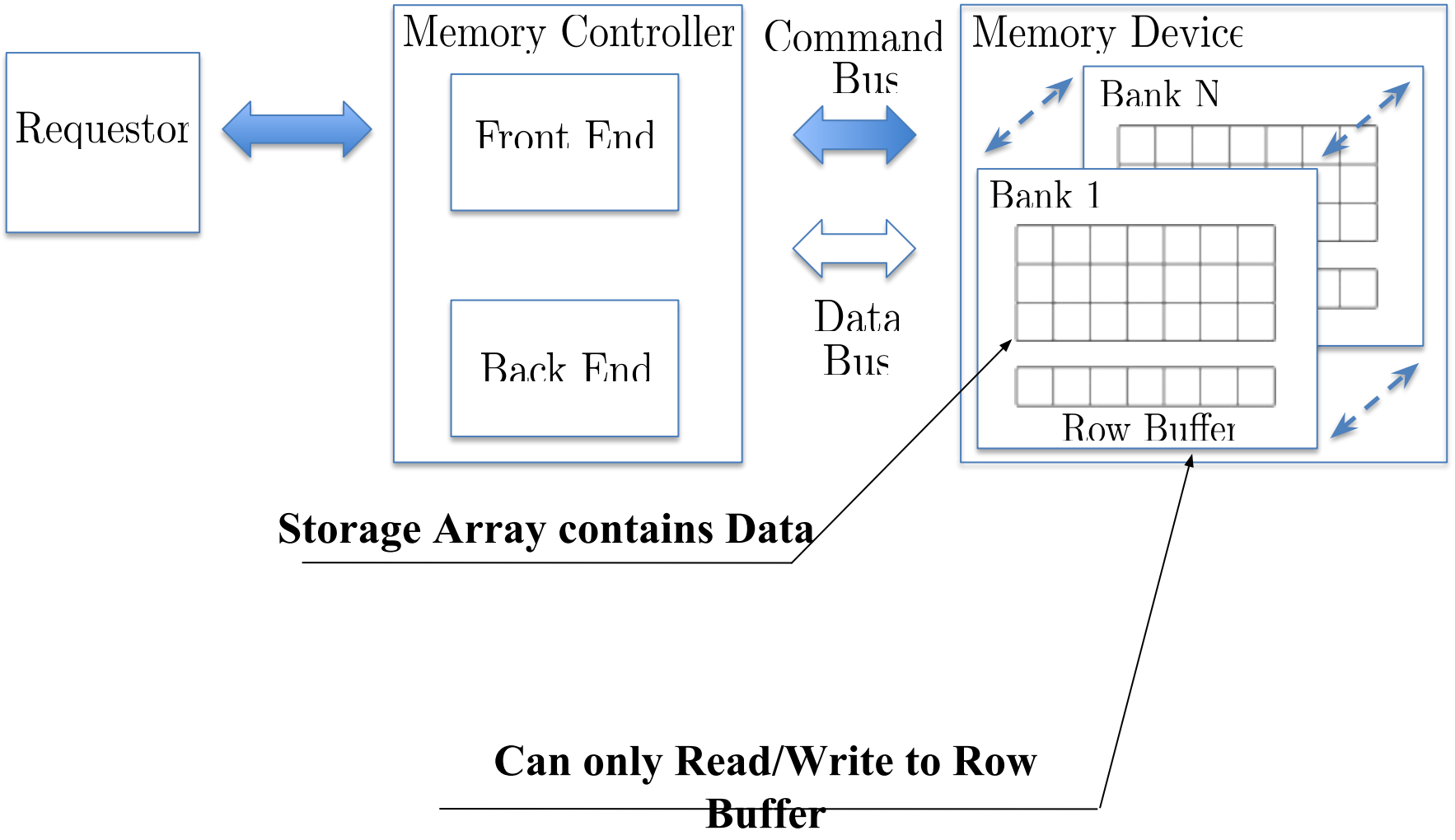
O4: RQ1, RQ4, RQ2, RQ3

S.No	Memory Cmd
RQ1	P, A, R
RQ4	P, A, W
RQ2	P, A, R
RQ3	P, A, R

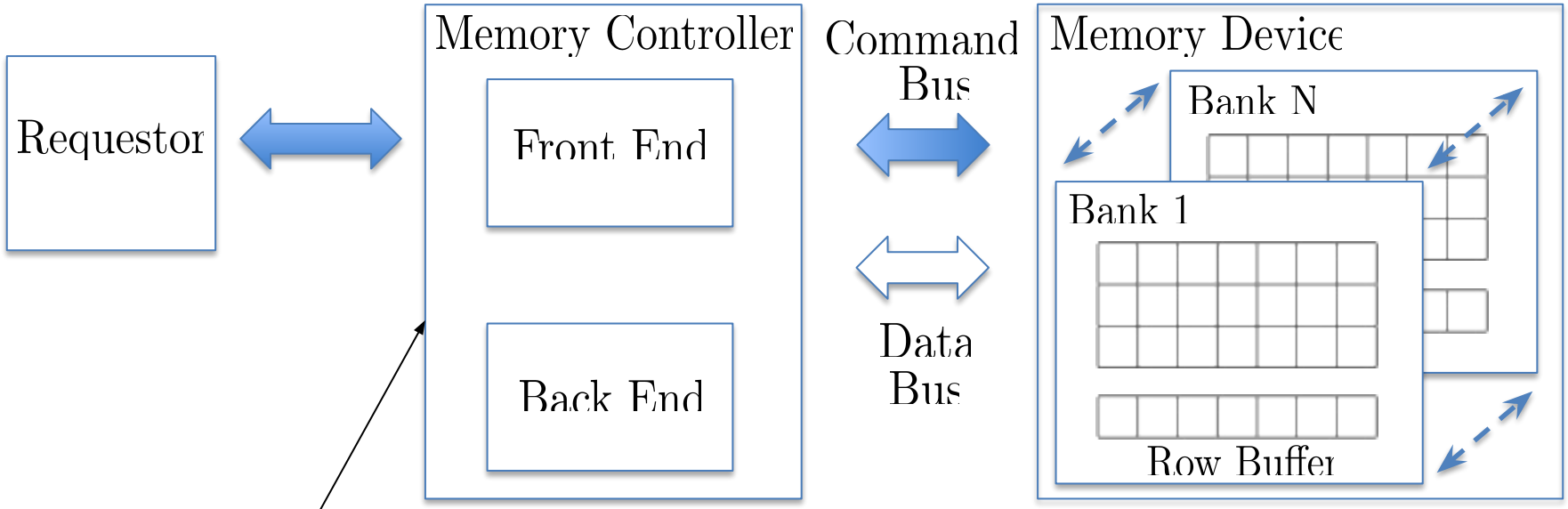
Total Cmd: 12
Locality?

Objective: Minimize the no. of Mem. Cmd.

DRAM's view

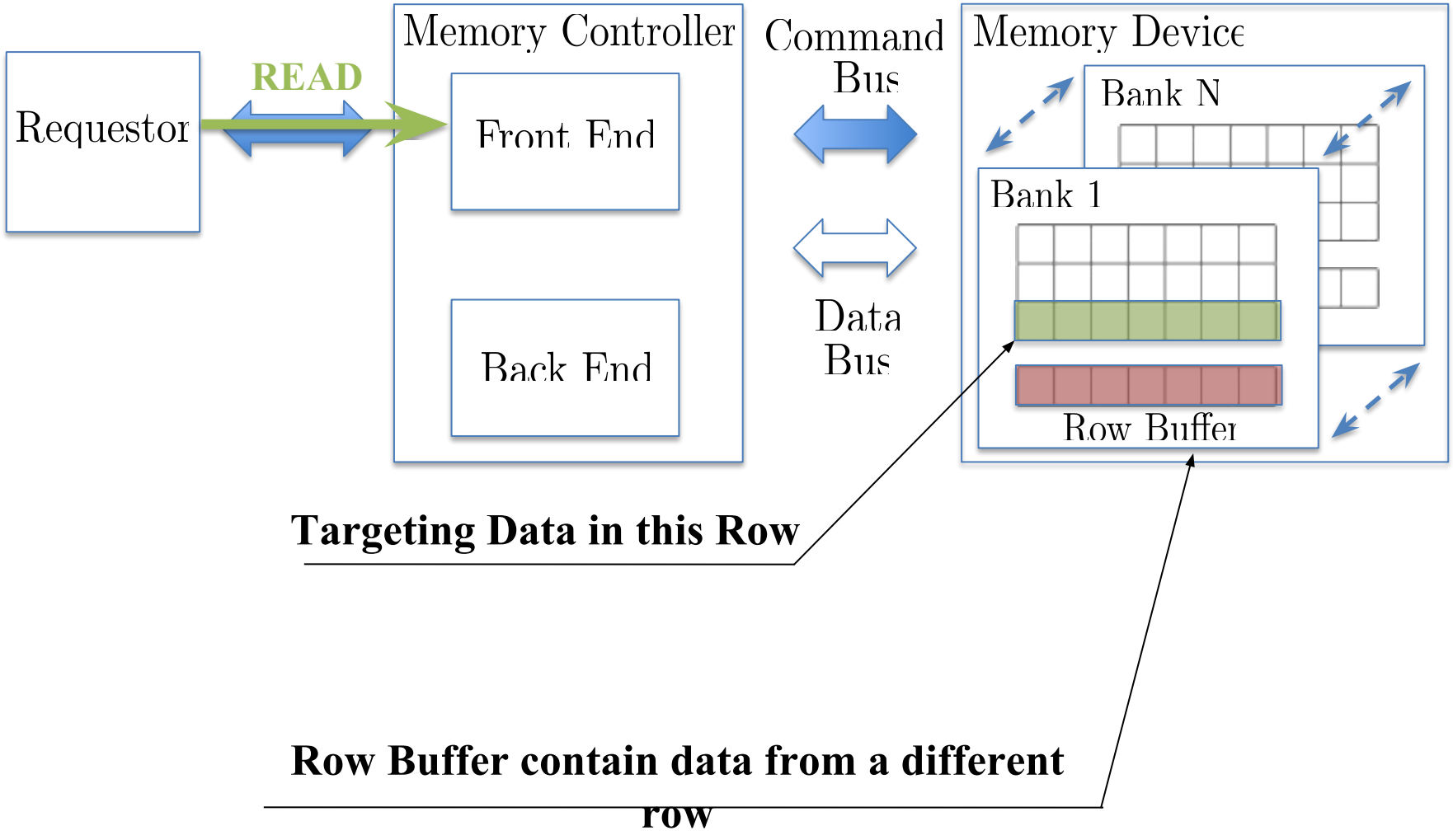

Requestor
Memory Controller
Front End
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Storage Array contains Data
Can only Read/Write to Row Buffer

DRAM's command generations

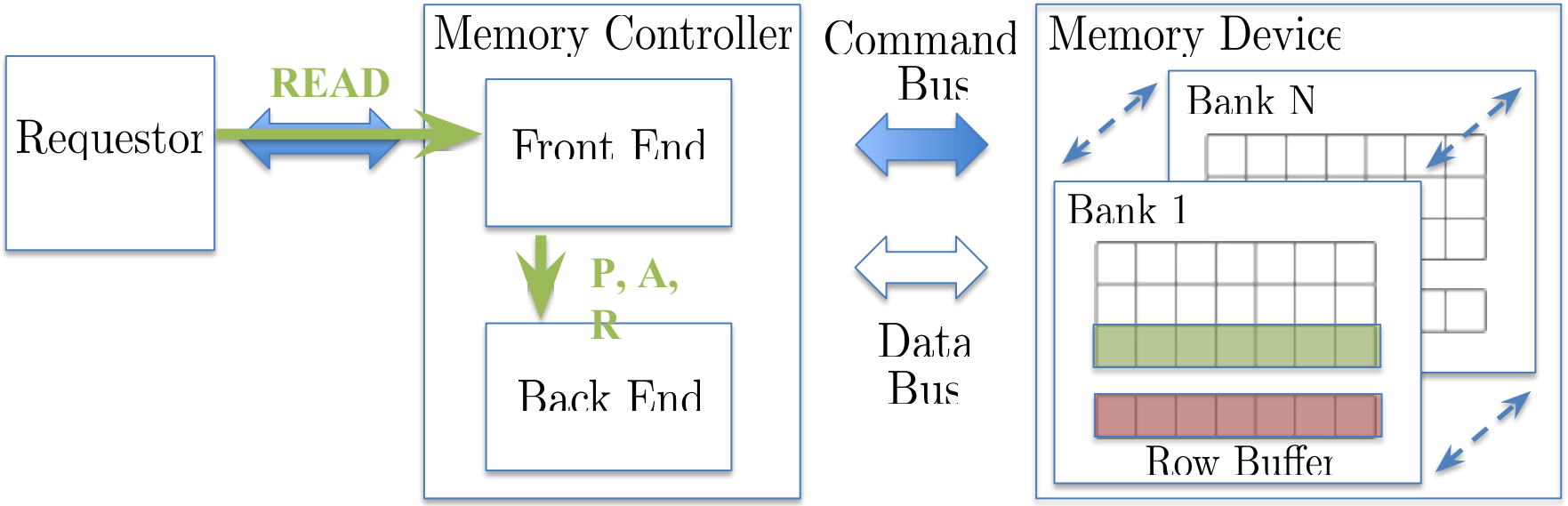


Front End generates the needed commands, Back End issues commands on command bus

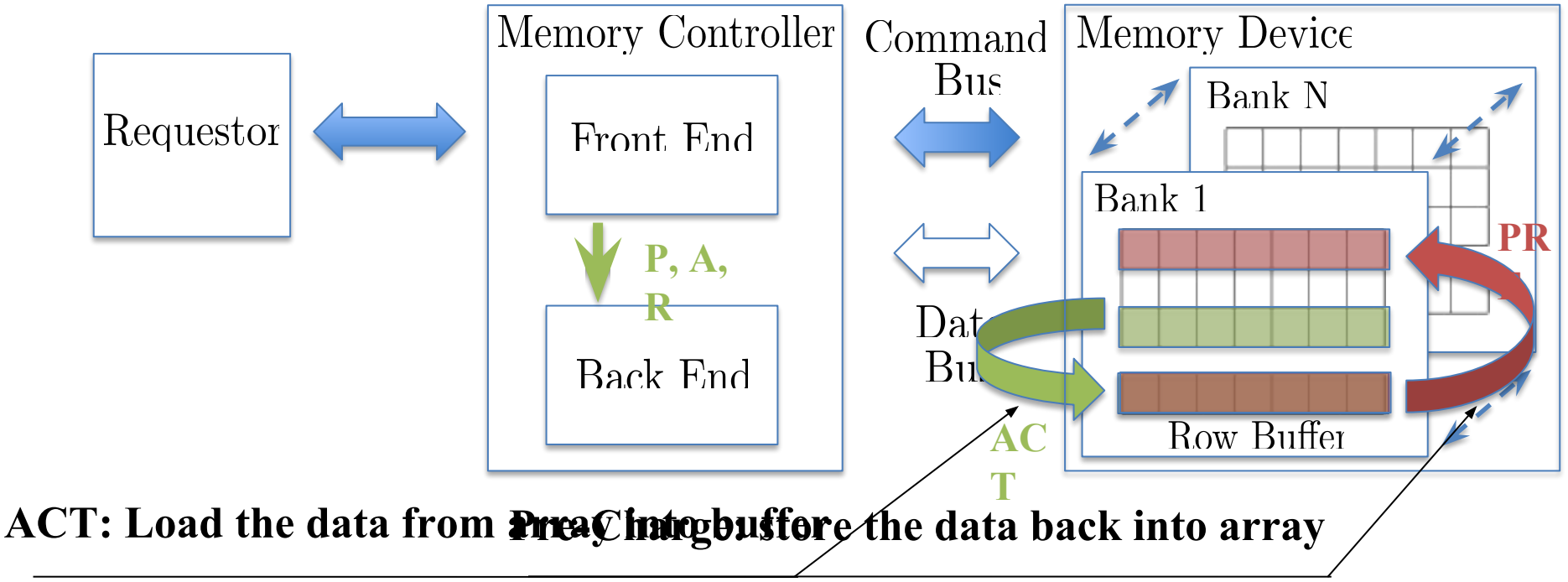
DRAM's Read operation



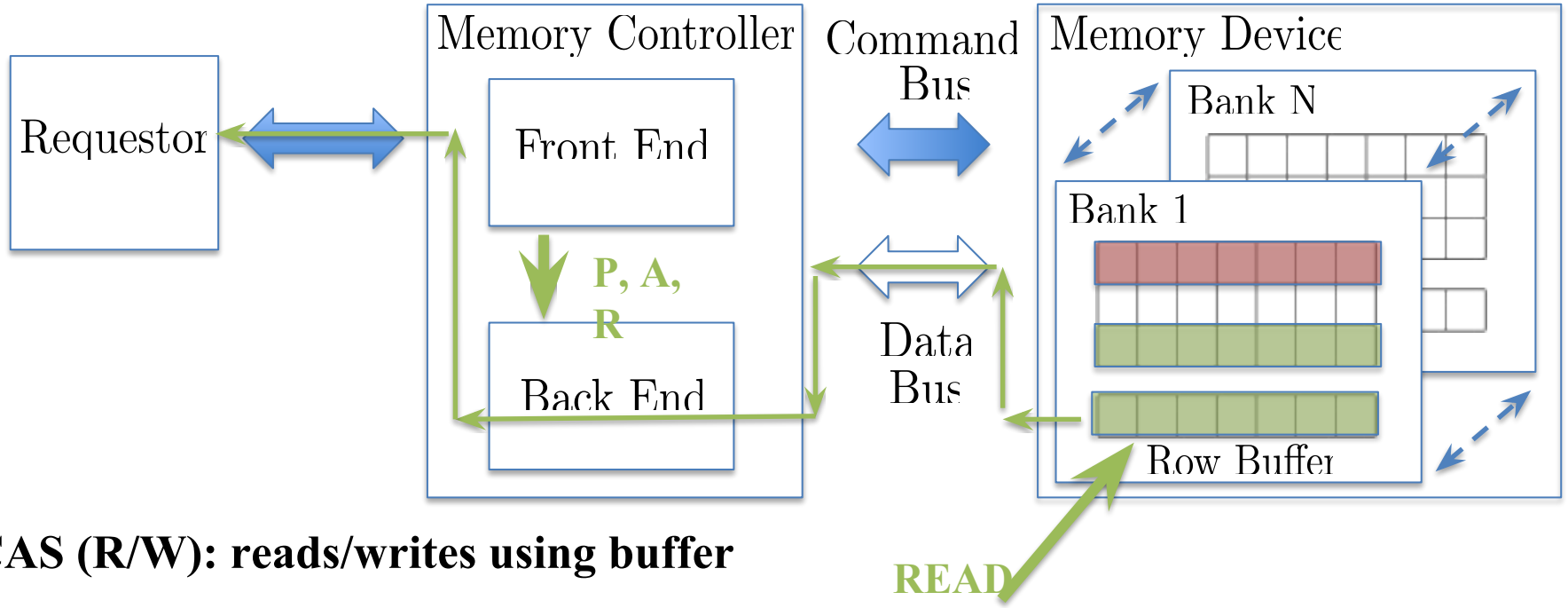
DRAM's read operation



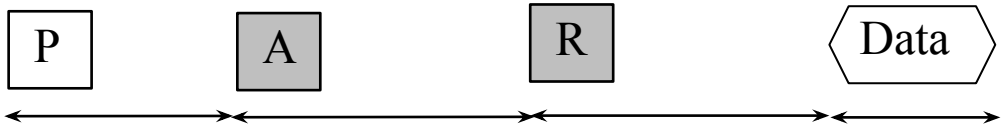
DRAM's read operation



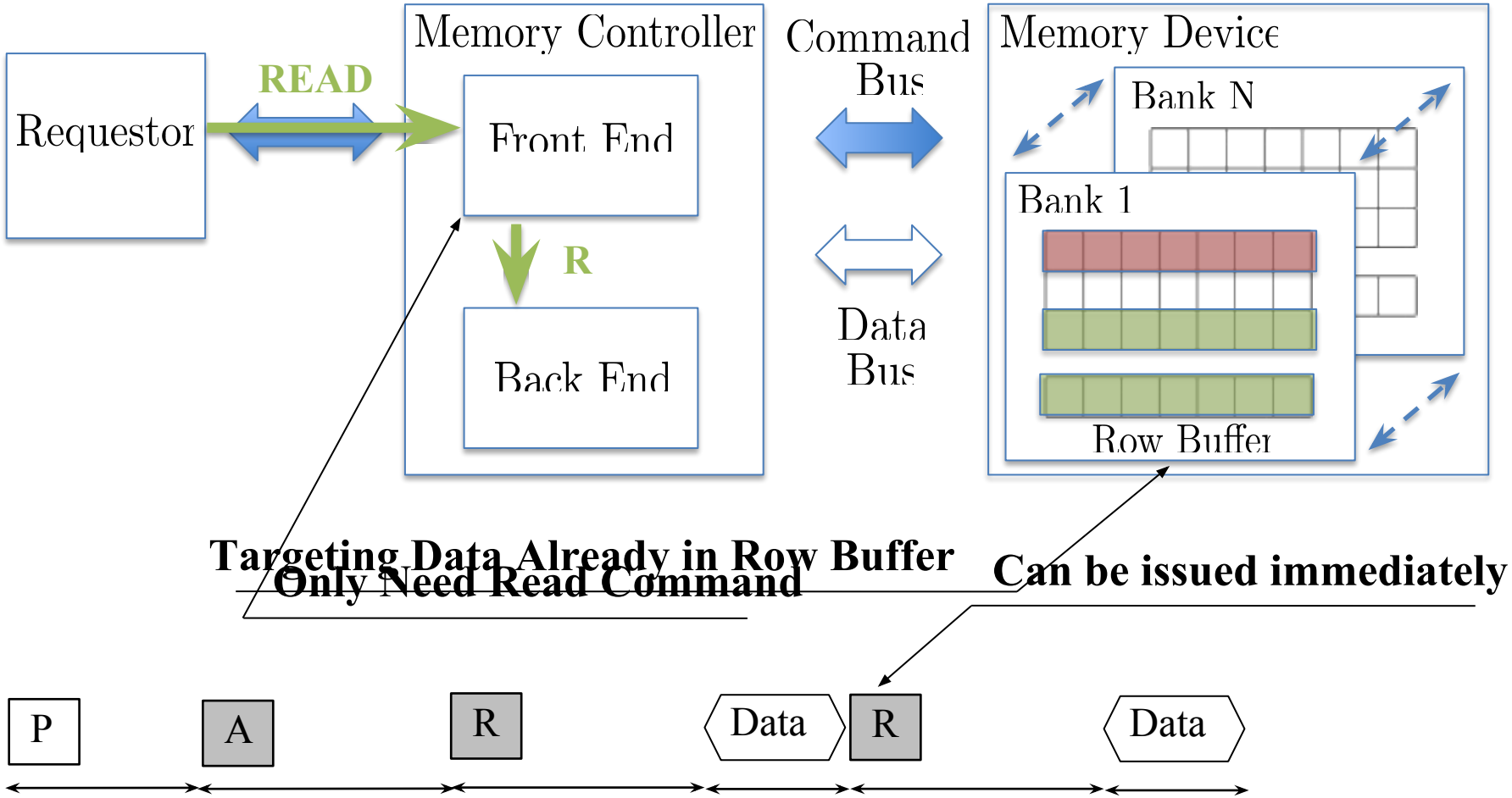
DRAM's read operation



CAS (R/W): reads/writes using buffer

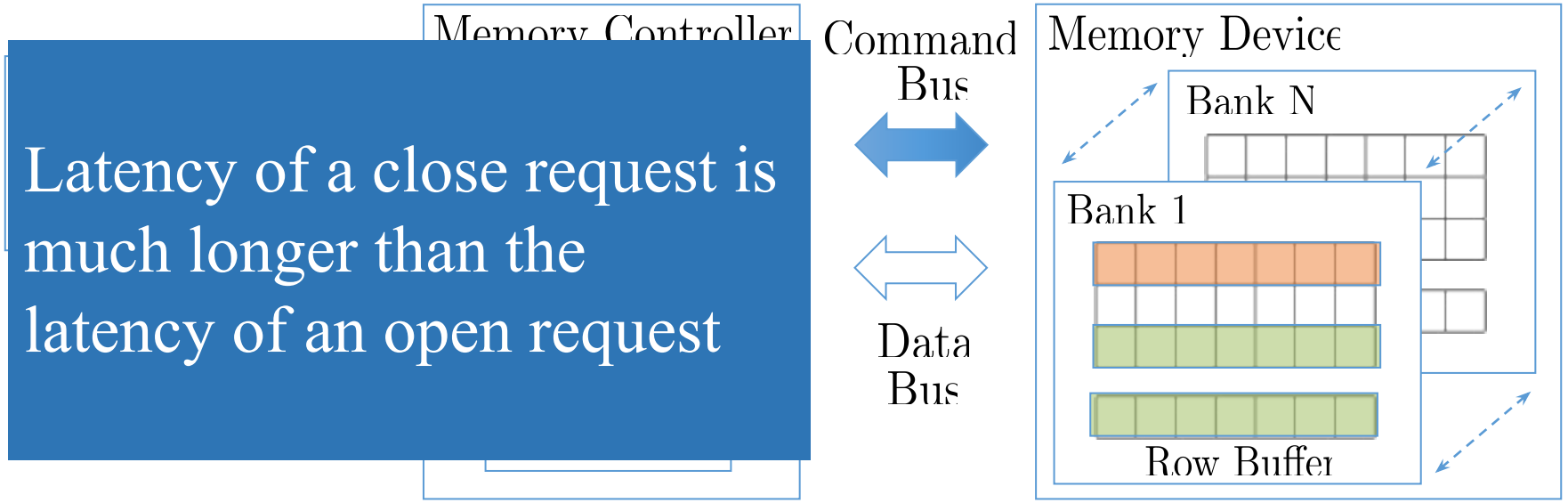


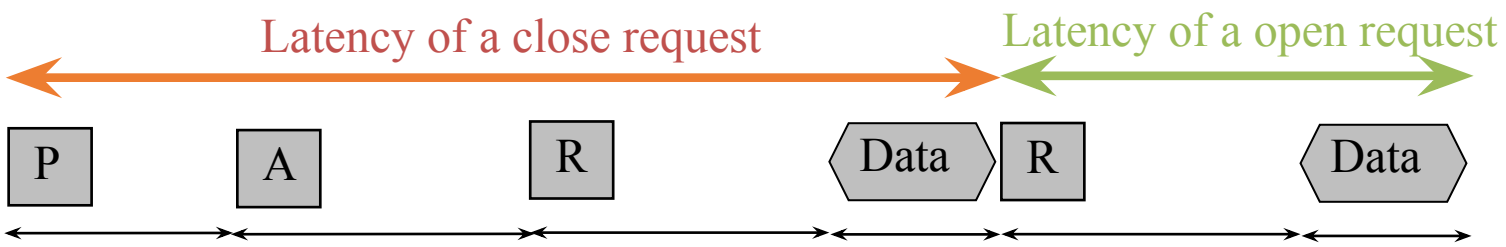
DRAM's read operation



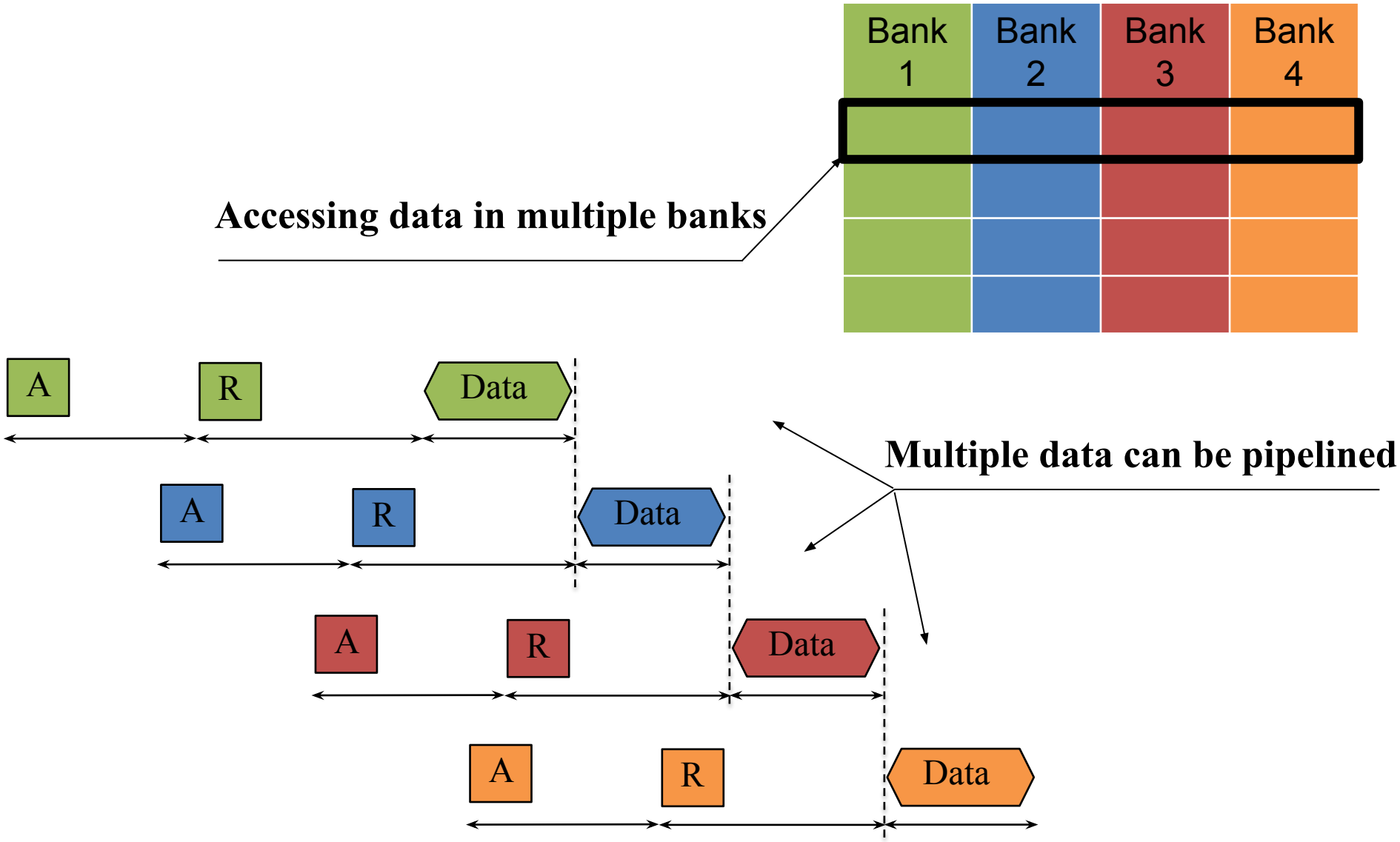
DRAM's read operation

Latency of a close request is much longer than the latency of an open request



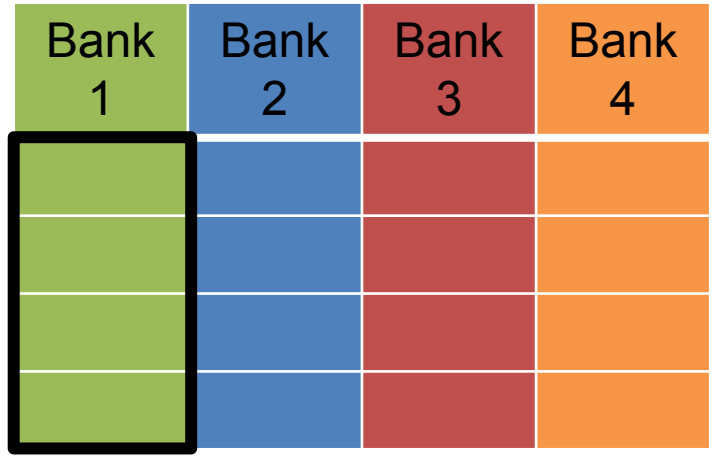


DRAM's Bank Parallelism Operation

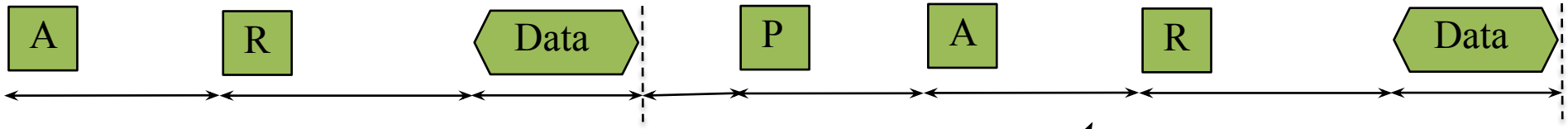


DRAM's Bank Parallelism

Requests should be spread over multiple banks



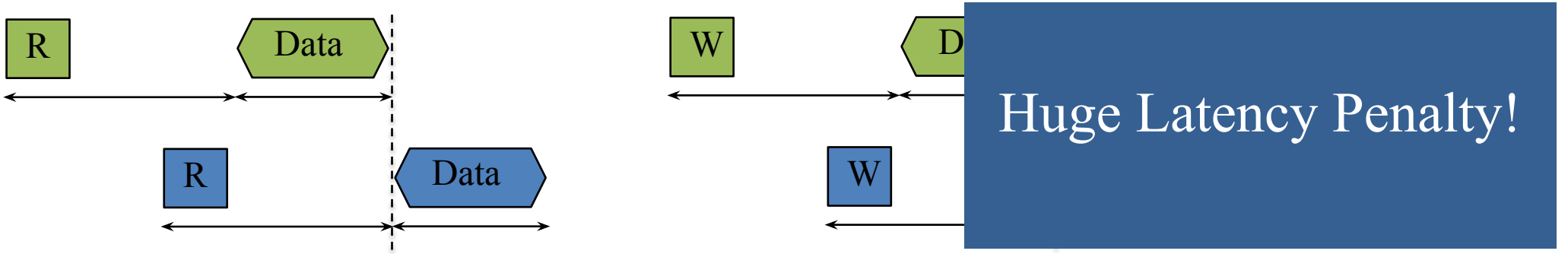
Accessing data in same bank, different rows



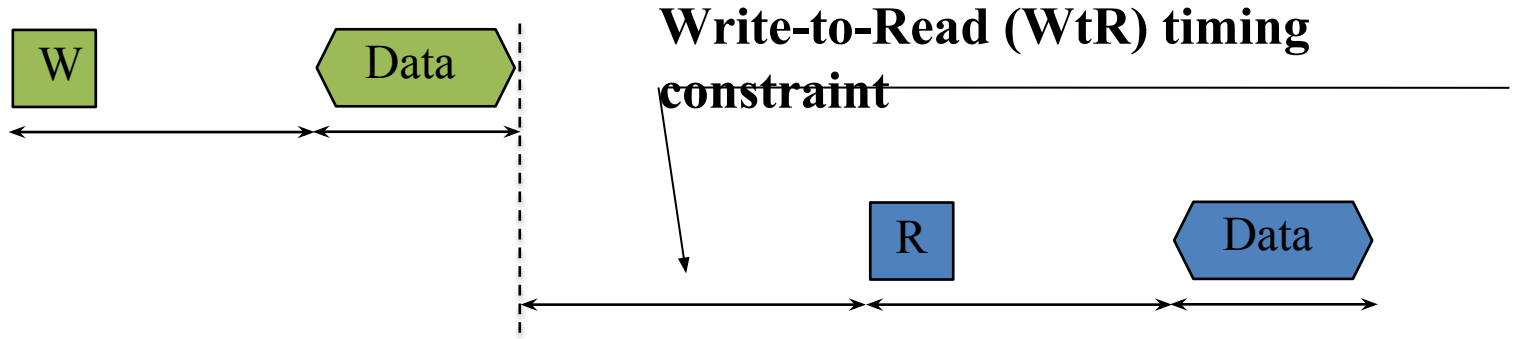
No pipelining – much larger delay

Write-to-Read Penalty

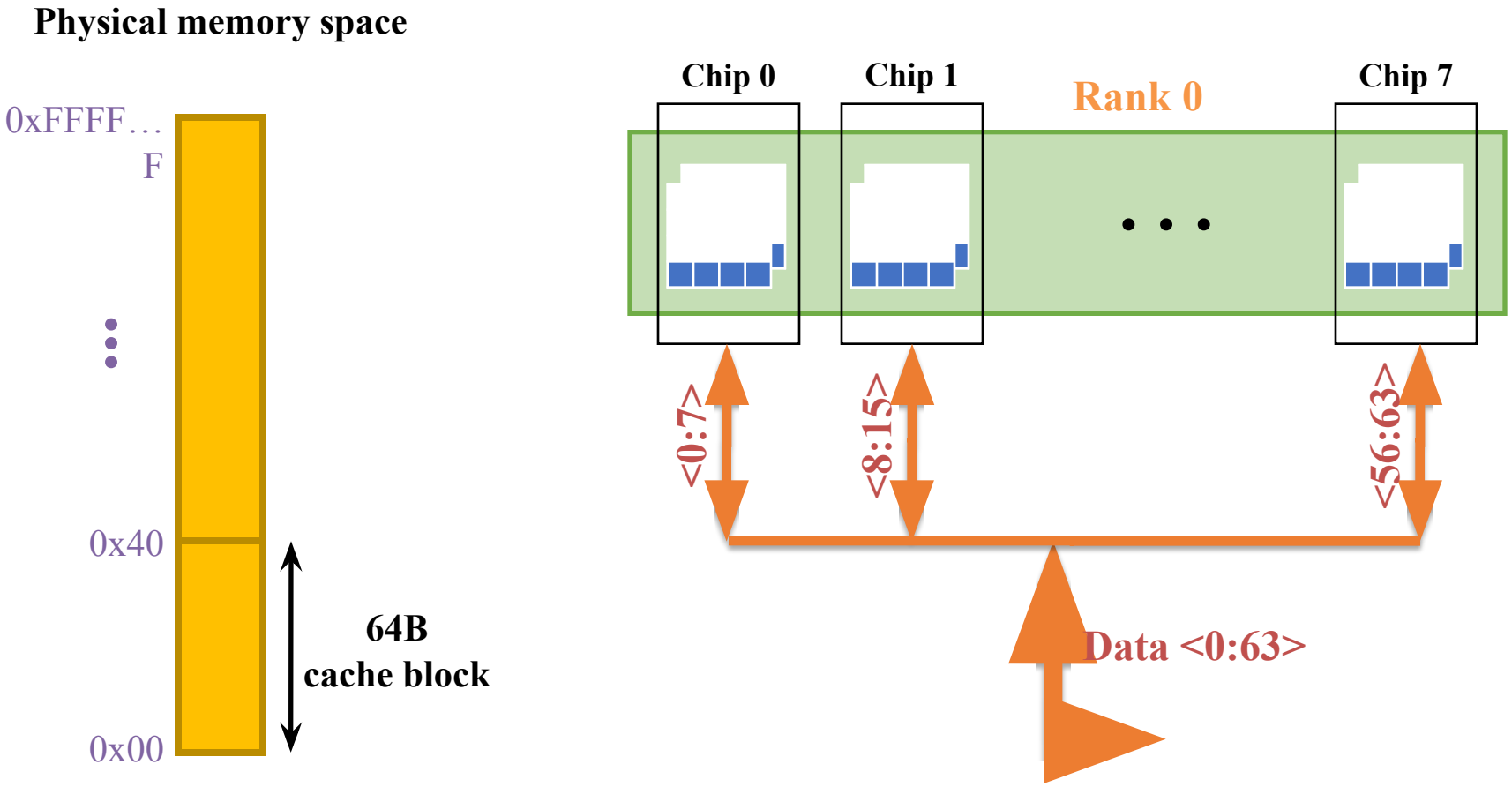
- Transactions of the same type can be pipelined…



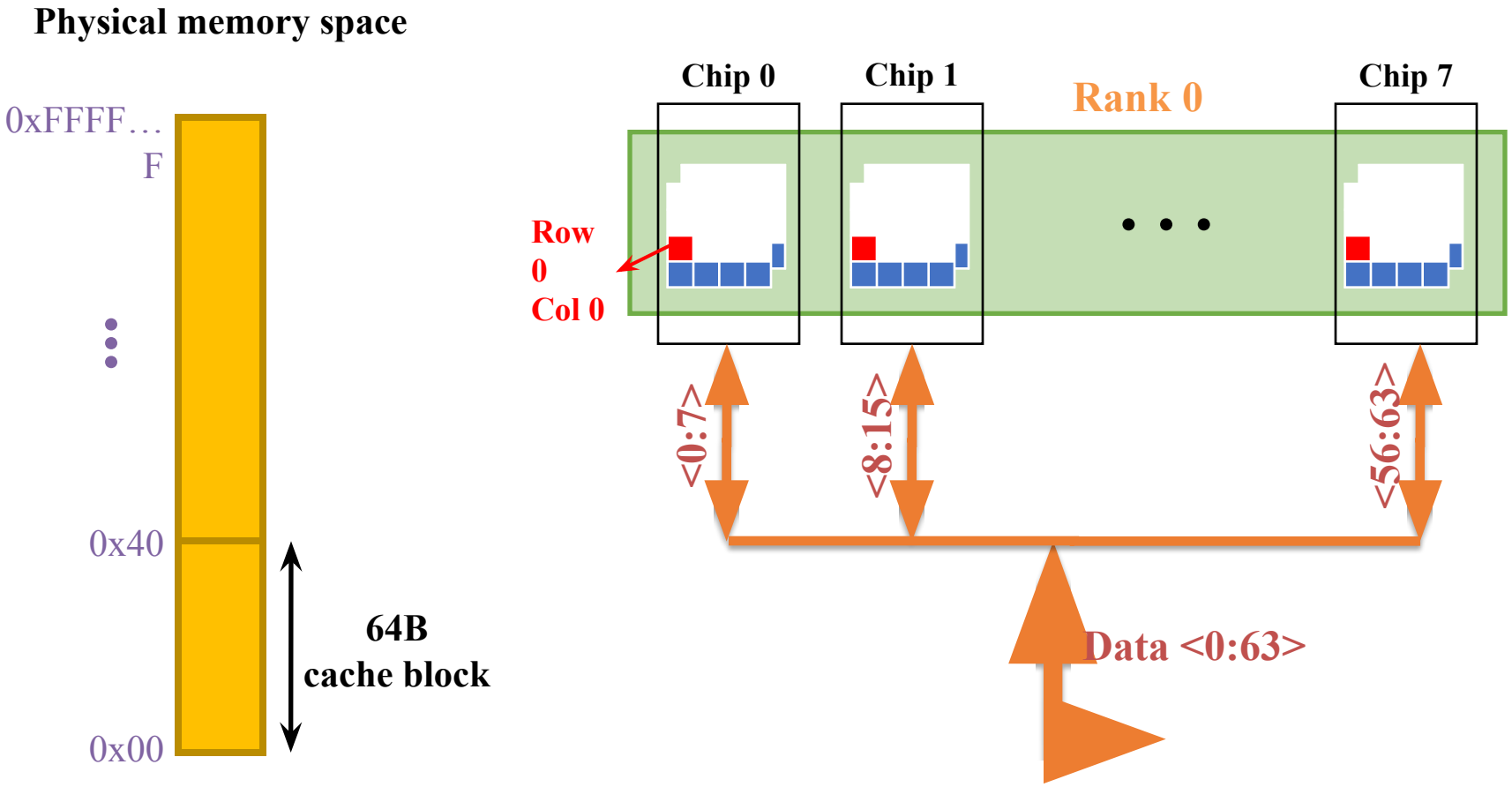
- … but a read after a write cannot.



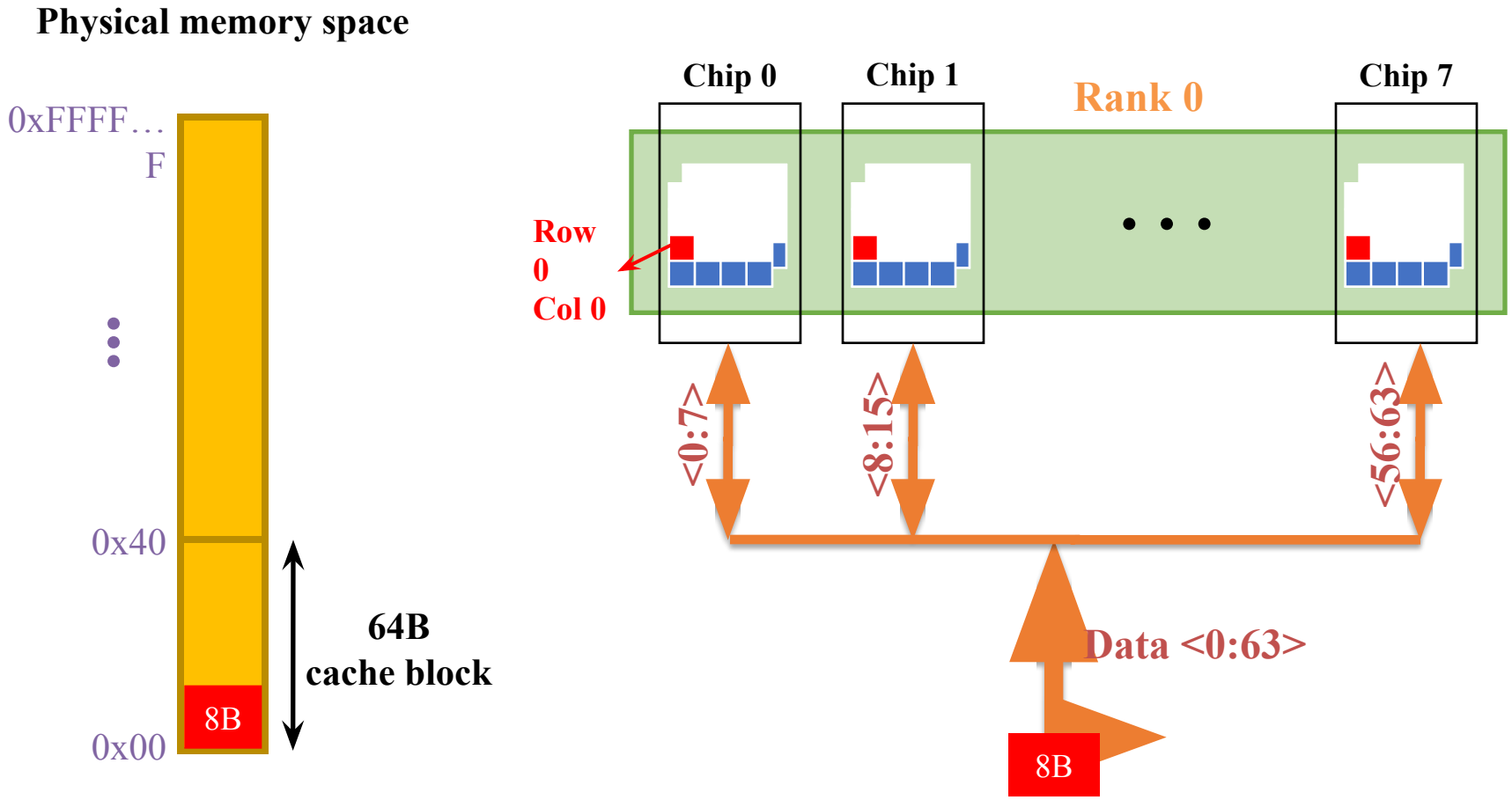
Example: Transferring a cache block



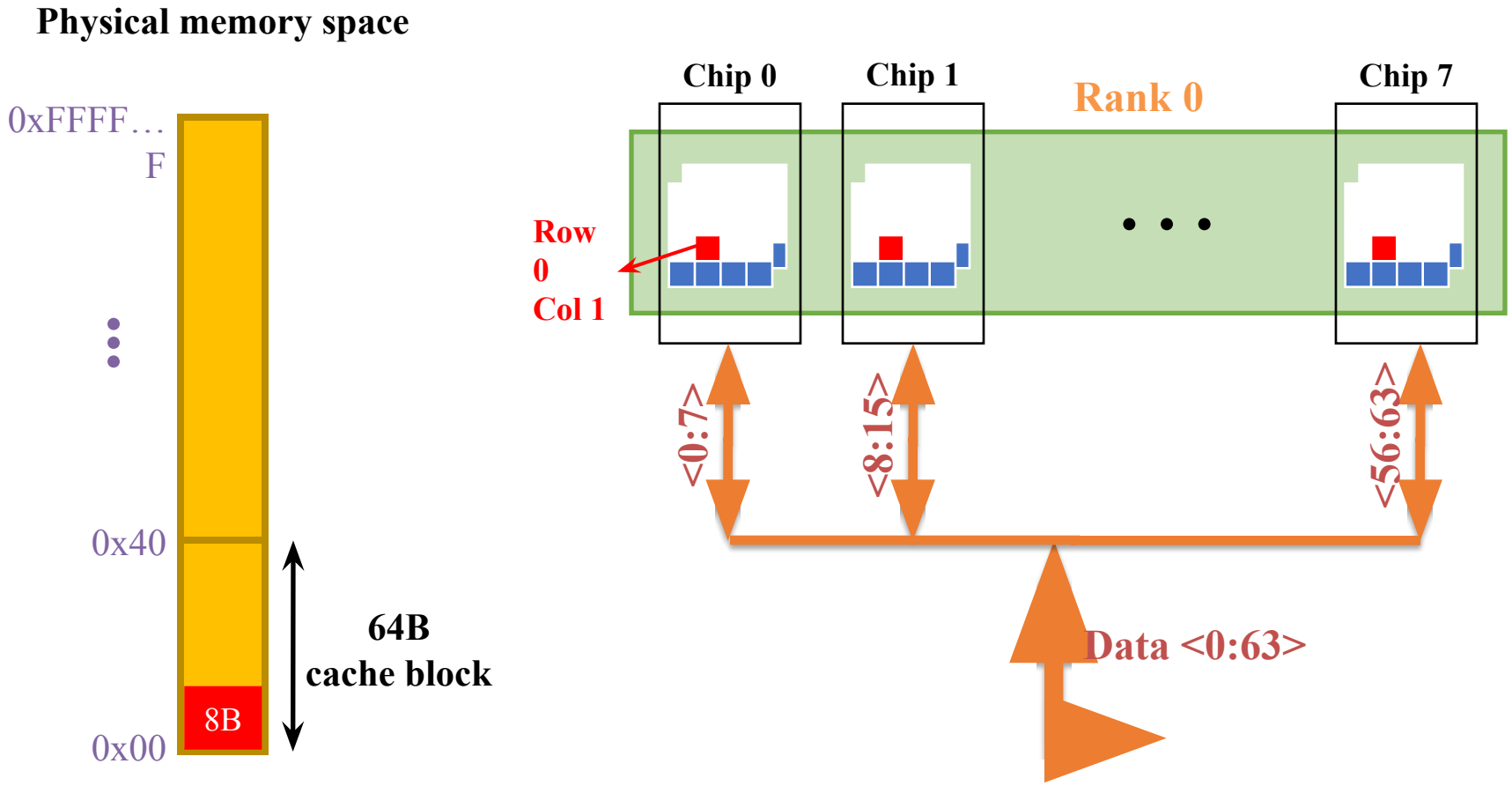
Example: Transferring a cache block



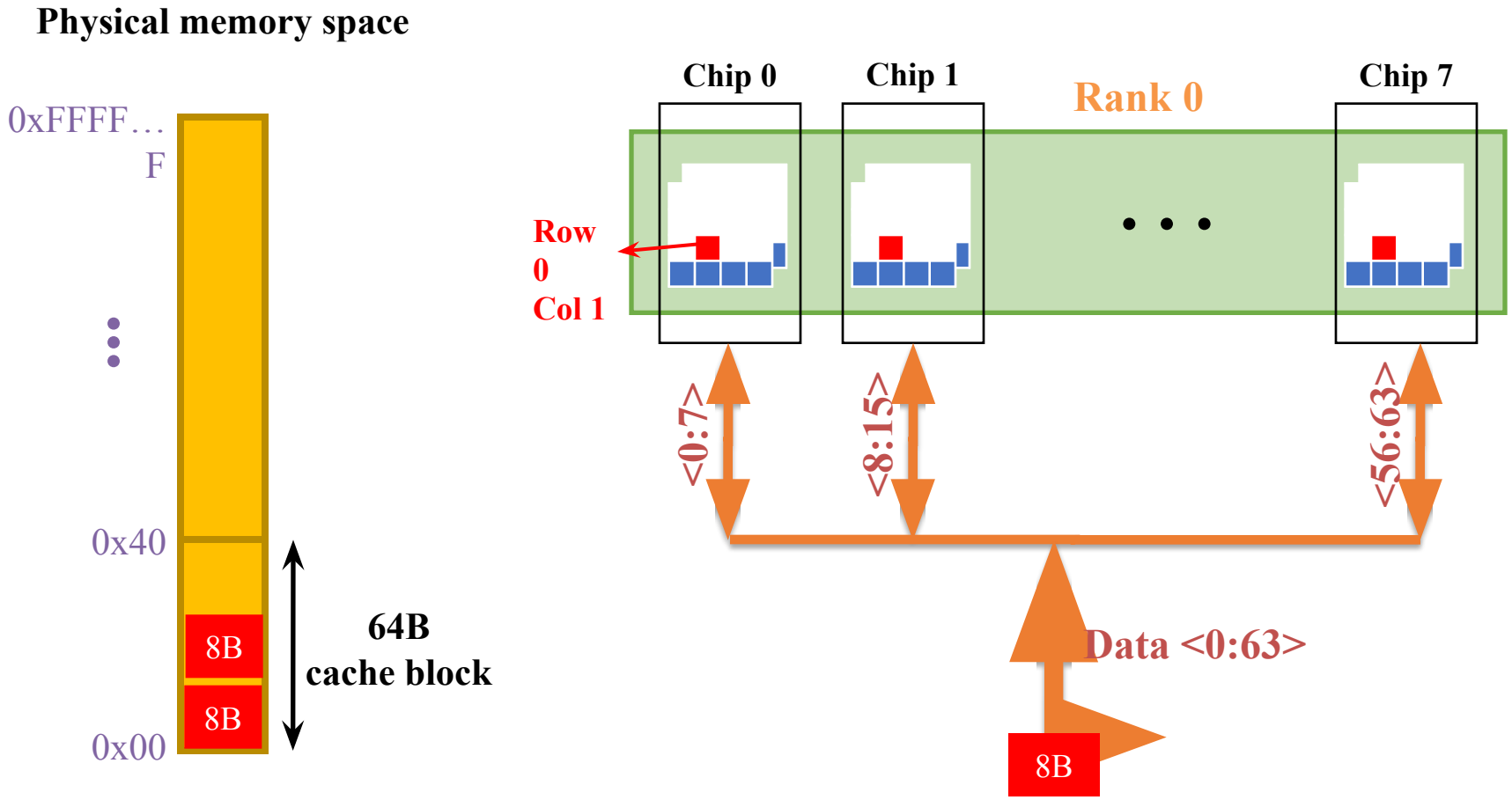
Example: Transferring a cache block



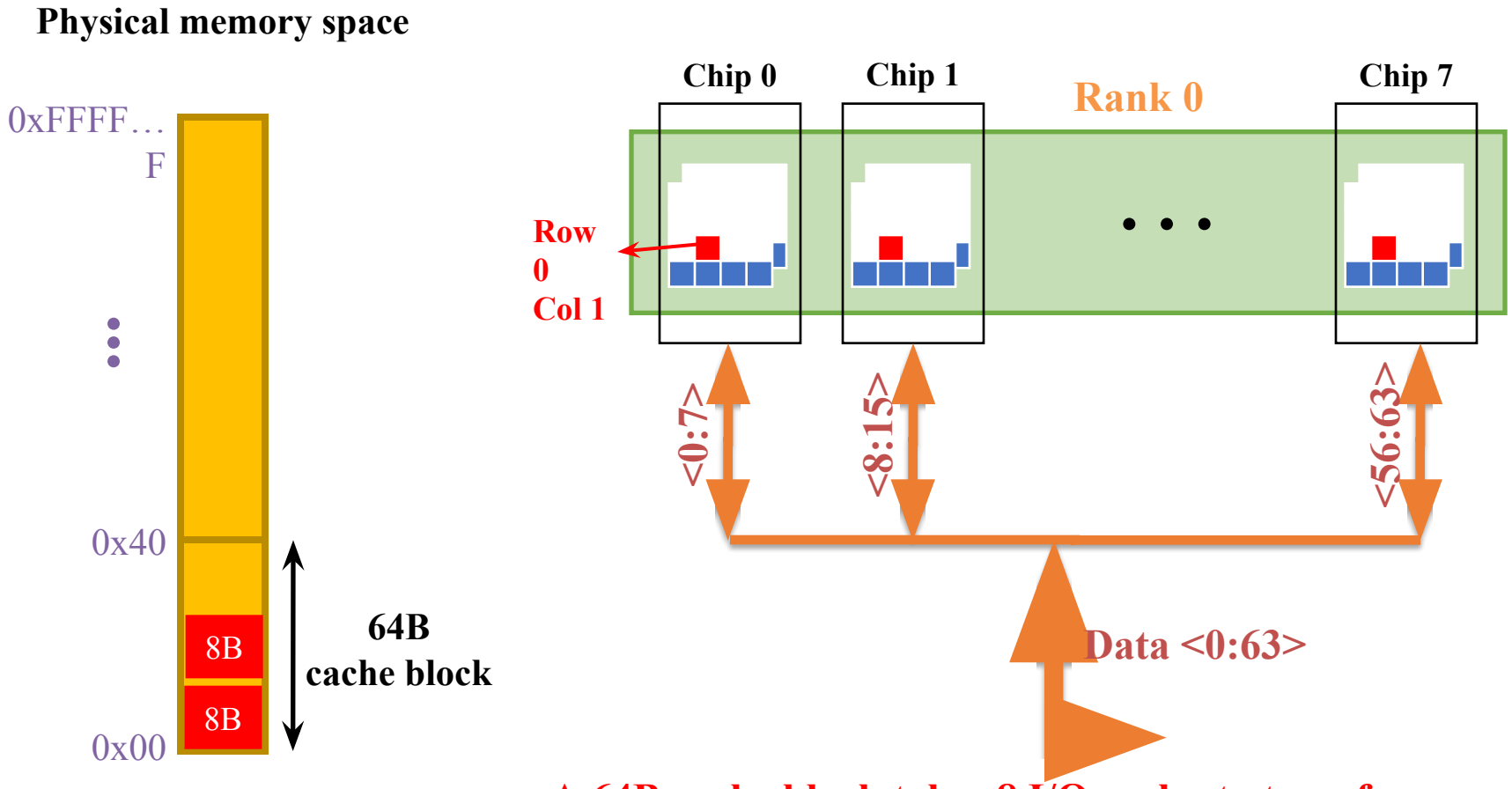
Example: Transferring a cache block



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.

Address mapping schemes

- Why do we need to study Address Mapping Scheme?
- To improve DRAM's performance
- Minimize the bank conflicts in temporally adjacent requests & maximize the parallelism in memory system
- Parallelism can be done at various levels:
  - Channels: highest
  - Ranks: rank-to-rank switching penalties
  - Banks: if shared resources are available, consecutive requests can be sent to different banks of a given rank
  - Rows: only one row per bank can be active at any given time
  - Columns: map the column address to the lower (higher) address bits of a given physical address, in open (closed)-paged system

Address Mapping Schemes

- Let's consider size of memory is K * L * B * R * C * V, where
- K: # of channels (=2^k)
- L: # of ranks per channels (=2^l)
- B: # of banks per rank (=2^b)
- R: # of rows per bank (=2^r)
- C: # of columns per row (=2^c)
- V: # of bytes per column (=2^v)
- The number of bytes per row (C * V) can be expressed as N * Z, where
  - N: # of cache line per row (=2^n)
  - Z: # of bytes per cache line (=2^z)
- Open-page address mapping scheme is r : l : b : n : k : z
- Closed-page address mapping scheme is r : n : l : b : k : z

		B0	B1	B0	B1	B0	B1	B0	B1
C0	R0	0	8	2	10	4	12	6	14
	R1	16	18	20	22	24	26	28	30
C1	R0	1	9	3	11	5	12	7	15
	R1	17	19	21	23	25	27	29	31
(r : l : b : n : k : z), keep i & i+1 cache line in same bank									

R_i=Ranks, C_i=Channels, B_i=Banks, #CacheLine:32,

		B0	B1	B0	B1	B0	B1	B0	B1
C0	R0	0	2	4	6	8	10	12	14
	R1	16	18	20	22	24	26	28	30
C1	R0	1	3	5	7	9	11	13	15
	R1	17	19	21	23	25	27	29	31
(r : n : l : b : k : z), keep i & i+1 cache line in separate bank									

What if we use row interleaving

Consecutive rows of memory in consecutive banks

Cache block interleaving:

Consecutive cache block addresses in consecutive banks

What if we have one channel?

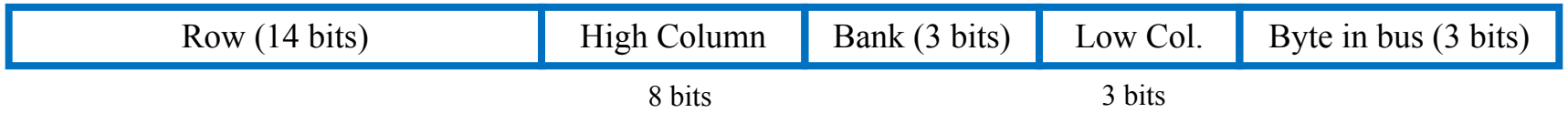
Address Mapping (Single Channel)

- 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

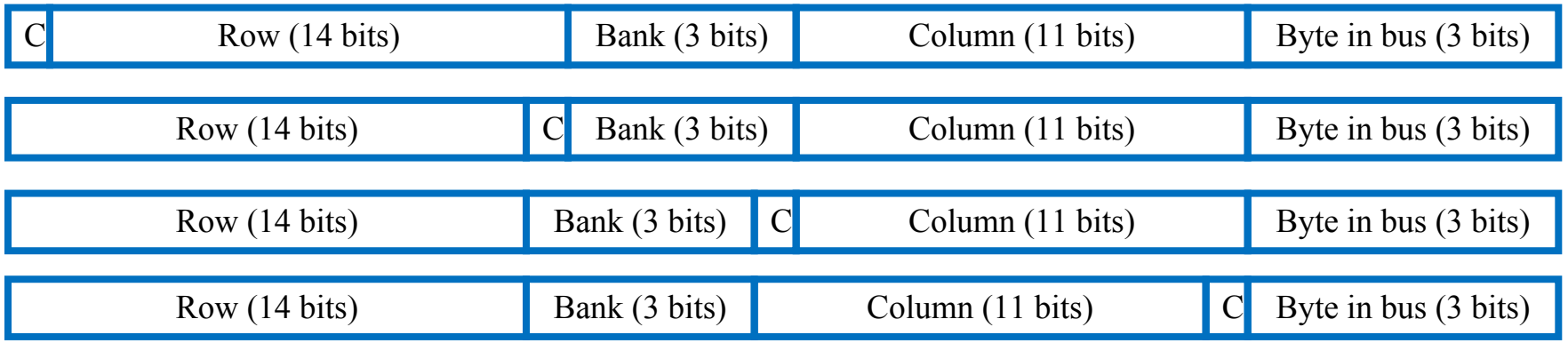
  - Accesses to consecutive cache blocks serviced in a pipelined manner

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

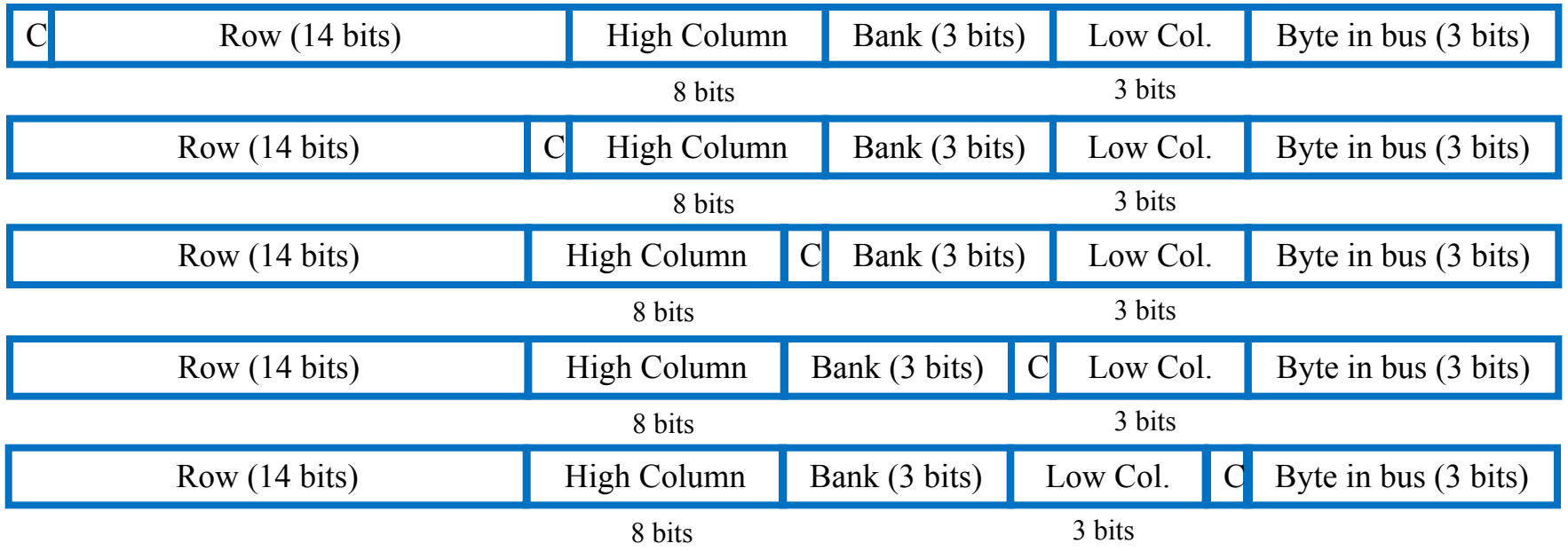


  - Accesses to consecutive cache blocks can be serviced in parallel

Address Mapping (Multiple Channels)



- Where are consecutive cache blocks?



Example: Where is the data?

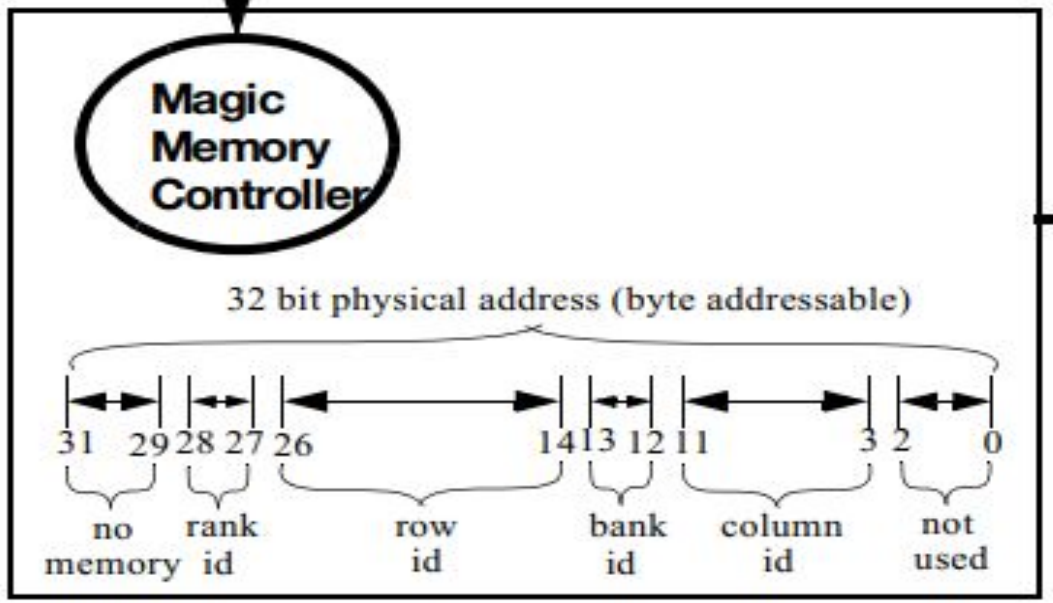
Read Request

Physical Address:

0x0AC75C38 = 0b 0000 1010 1100 0111 0101 1100 0011 1000

= 000 01 0101100011101 01 110000111 000

rank row bank column



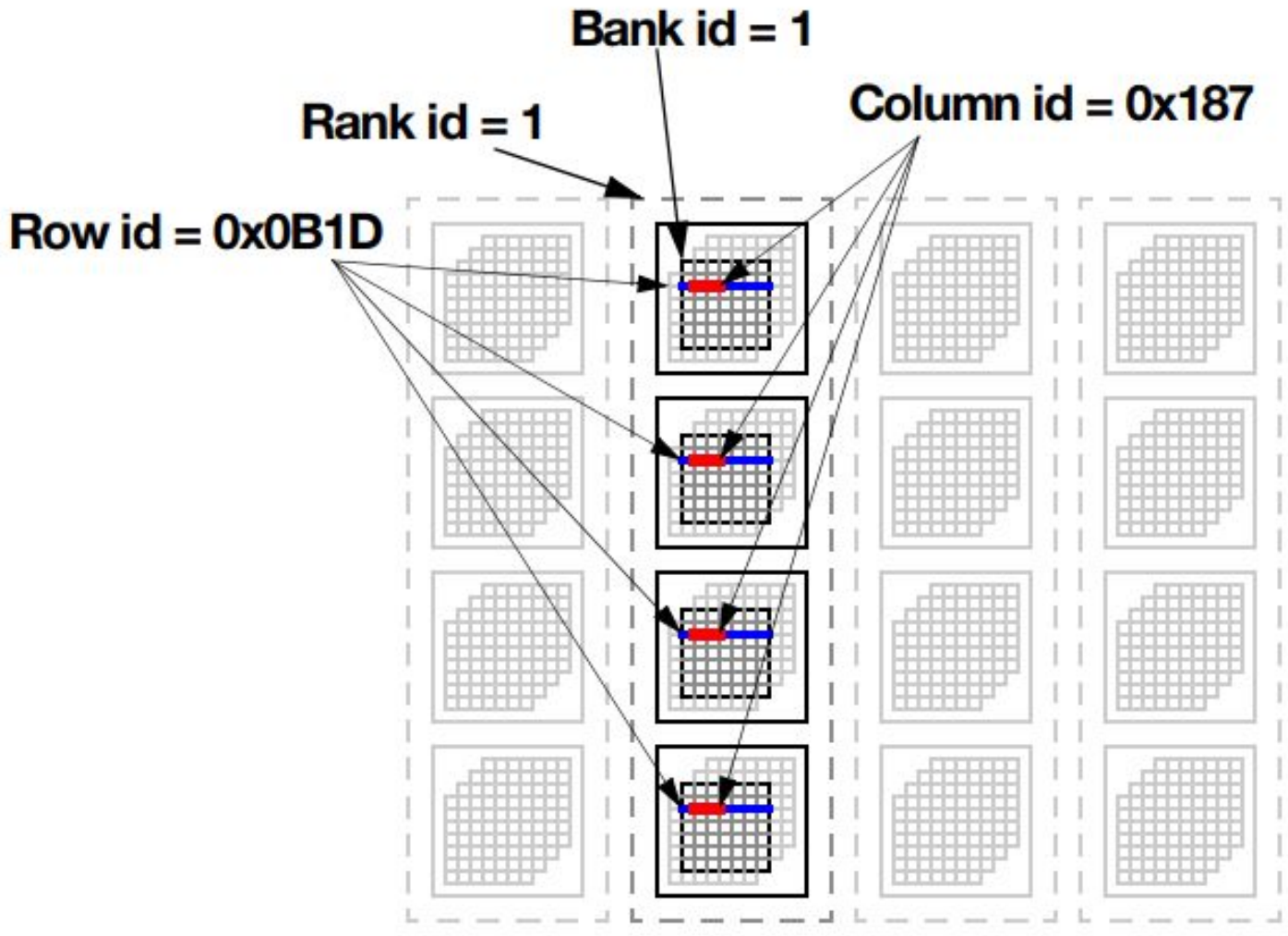
Rank id = 1

Bank id = 1

Row id = 0x0B1D

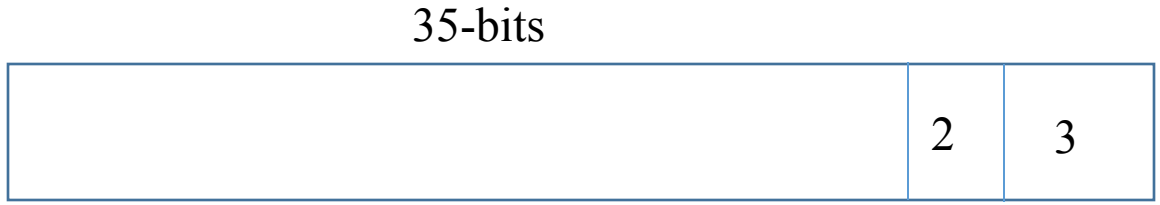
Column id = 0x187

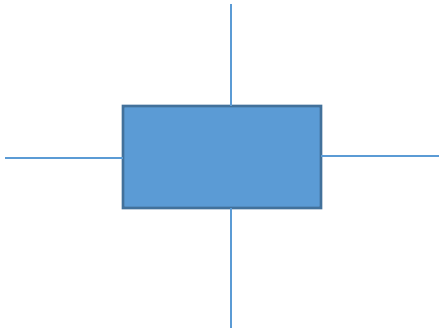
Example: Where is the data?



Example

- A 32-GB DRAM systems have 4-channels(C0-C4) and 4096-columns per row. The data bus is of size 8-B for transferring data from DRAM to last level cache. If adjacent memory words are mapped on to adjacent memory channels, which channel hold the data for physical address 0x3488A752?
- 32-GB = 2^35 B; 35-bits address line





CC Byte

0x3488A752 = …0101_0010

Channel = (10)b = CC-3

Example

- 64-GB DRAM, 4 channels, each channel with 2-DIMMs, each DIMM having 2-ranks, 8-banks and in each bank the number of rows and columns is same. The width of data, for communicating with last-level cache, bus is 8-B. Each channel is connected to 16-GB of continuous memory. Find out the bits needed to identify channel, DIMM, Rank, Bank, Rows, Cols.
- 64-GB = 2^(6+30); 36-bits for address

Channel	DIMM	Rank	Bank	Row	Col	Byte
2	1	1	3	13	13	3

Example

- 64-GB DRAM, 4 channels, each channel with 2-DIMMs, each DIMM having 2-ranks, 8-banks and in each bank the number of rows and columns is same. The width of data, for communicating with last-level cache, bus is 8-B. Each channel is connected to 16-GB of continuous memory. Find out the bits needed to identify channel, DIMM, Rank, Bank, Rows, Cols.
- If the system is using row interleaving in banking, which is the bank no. and channel no. for the physical address 0x844332255
- 64-GB = 2^(6+30); 36-bits for address

Channel	DIMM	Rank	Row	Bank	Col	Byte
2	1	1	13	3	13	3

0x844332255
Bank:3 (011)
Channel: 2 (10)

Example

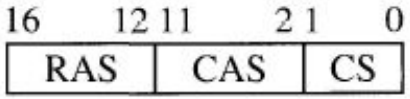
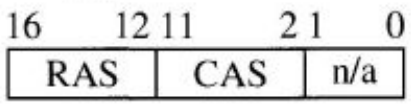
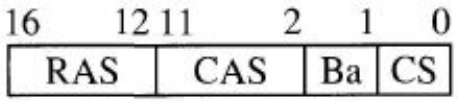
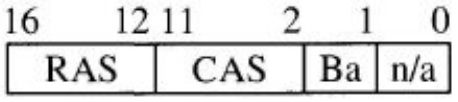
- 64-GB DRAM, 4 channels, each channel with 2-DIMMs, each DIMM having 2-ranks, 8-banks and in each bank the number of rows and columns is same. The width of data, for communicating with last-level cache, bus is 8-B. Each channel is connected to 16-GB of continuous memory. Find out the bits needed to identify channel, DIMM, Rank, Bank, Rows, Cols.
- If the system is using a 128-B last level cache with cache block interleaving, what is the bank no. & channel no. for the physical address 0x844332255?
- 64-GB = 2^(6+30); 36-bits for address

Channel	DIMM	Rank	Row	Col. High	Bank	Col. Low	Byte
2	1	1	13	9	3	4	3

0x844332255
255 (0010_0101_0101)
Bank:4 (100)
Channel: 2 (10)

Example: Homework

Refer to Slide#30 Serial Vs. Parallel, non-interleaving Vs. interleaving

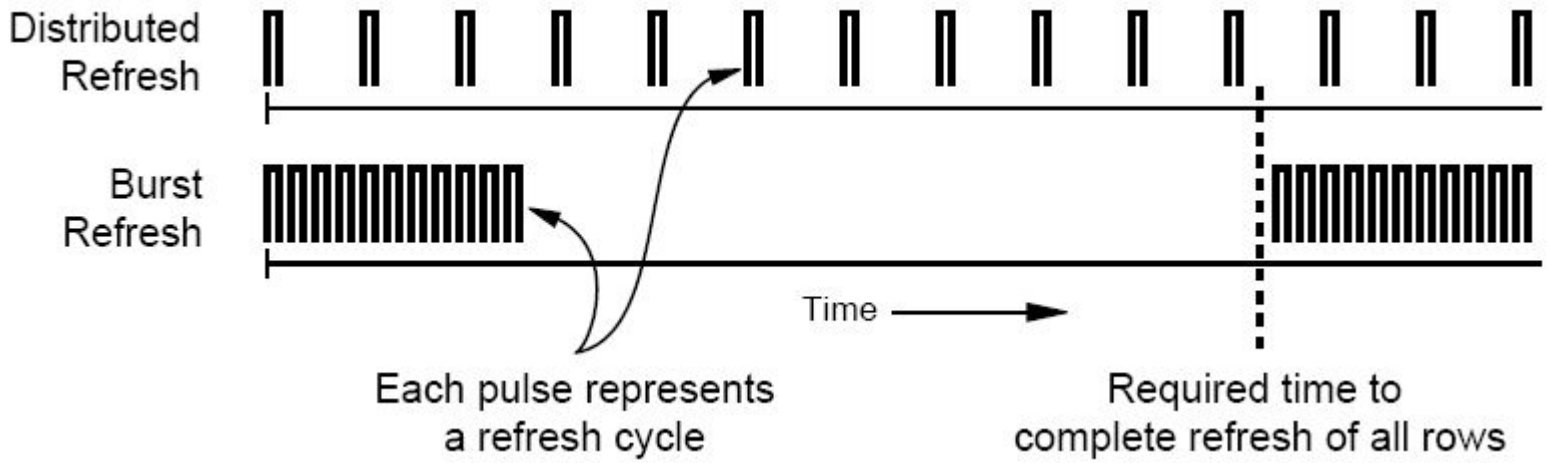
Organization	DRAM Address Breakdown	Example Physical Address	Corresponding DRAM Address
Serial non-interleaved, 8-bit data bus		0x4321	RAS: 0x4, CAS: 0xC8, CS: 0x1
Parallel non-interleaved, 32-bit data bus		0x4320	RAS: 0x4, CAS: 0xC8
Serial interleaved, 8-bit data bus		0x8251	RAS: 0x4, CAS: 0x94, Bank: 0x0, CS: 0x0
Parallel interleaved, 16-bit data bus		0x8254	RAS: 0x4, CAS: 0x95, Bank: 0x0

*Examples assume 4 × 256-kbit DRAM with 8-bit data path and 8-kbit row, for a total of 128 kB of addressable memory.

Is it a row interleaving scheme?

Refresh Operation

- Read each row-data into row-buffer and restore it into its location, on time
- Consume bandwidth & power
- Minimize: controller's complexity or bandwidth or power consumption
- All-bank concurrent refresh: a single refresh command to the DRAM device
  - Refresh address register to store the address of the last refreshed row
- For an example, 8192 refresh commands issued by MC for every 64-ms
- Distributed refresh:

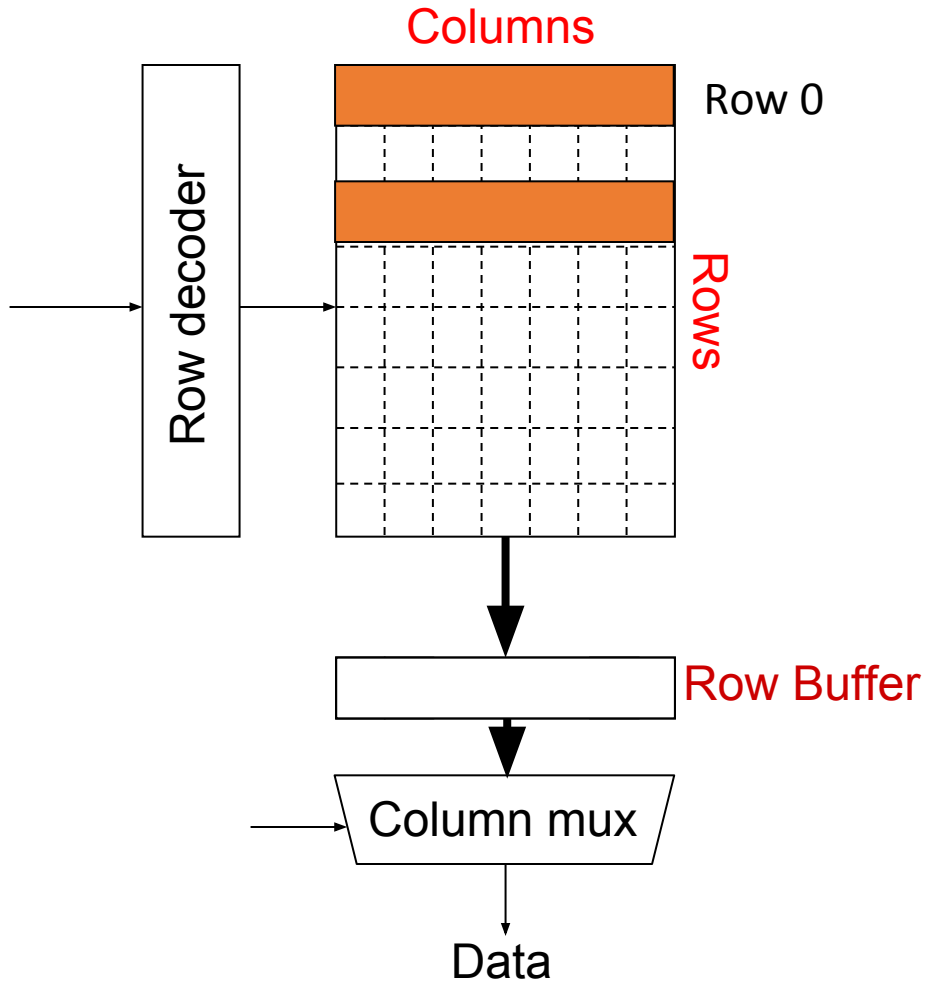


Memory Access Scheduling

- Scheduling at
  - Request level
  - Command level
- Two separate queues to hold reads and writes
  - Memory controller works in read major mode and write major mode
- Request scheduler can select request in out-of-order fashion also
- Command scheduler selects a command based on Round-Robin, First Ready First Serve (FRFS), or aged-based policies
- Command scheduler picks the commands from command queue in FIFO order

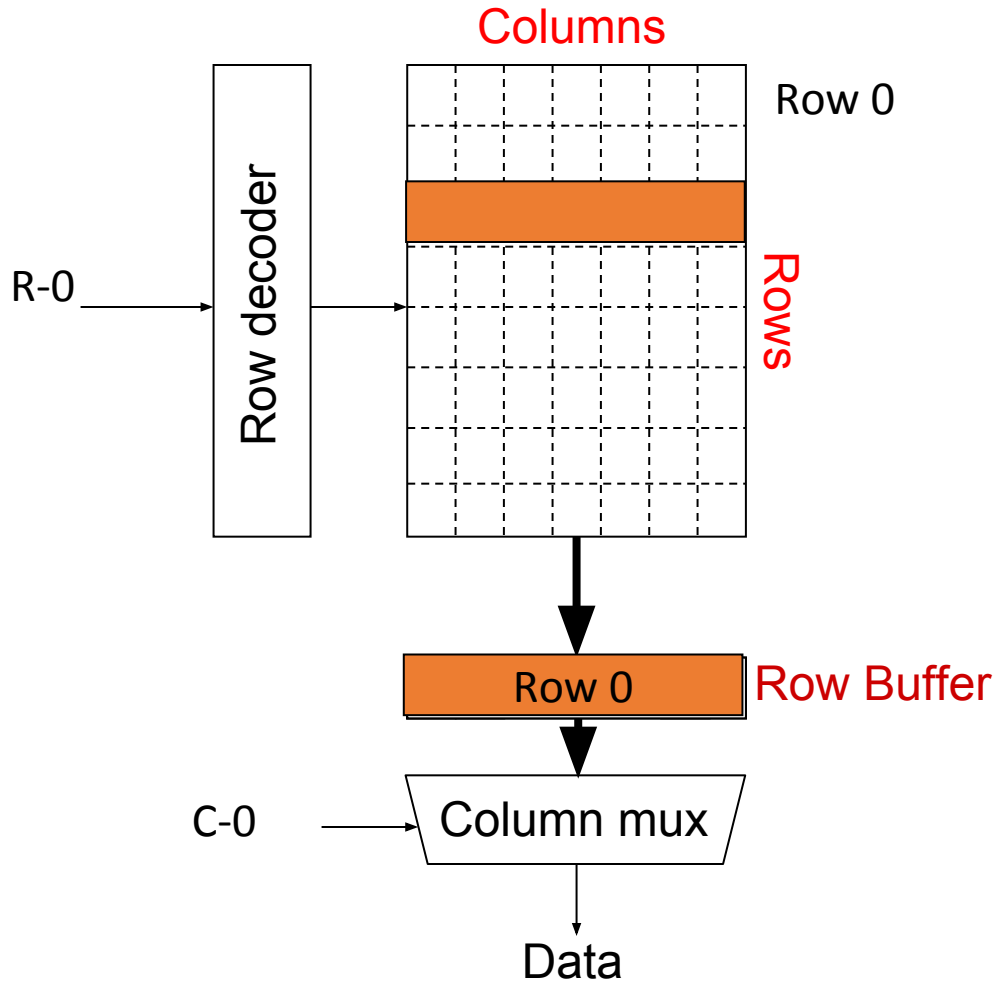
Memory Access Scheduling: FCFS

(R-0, C-4)
(R-2, C-0)
(R-0, C-1)
(R-0, C-0)



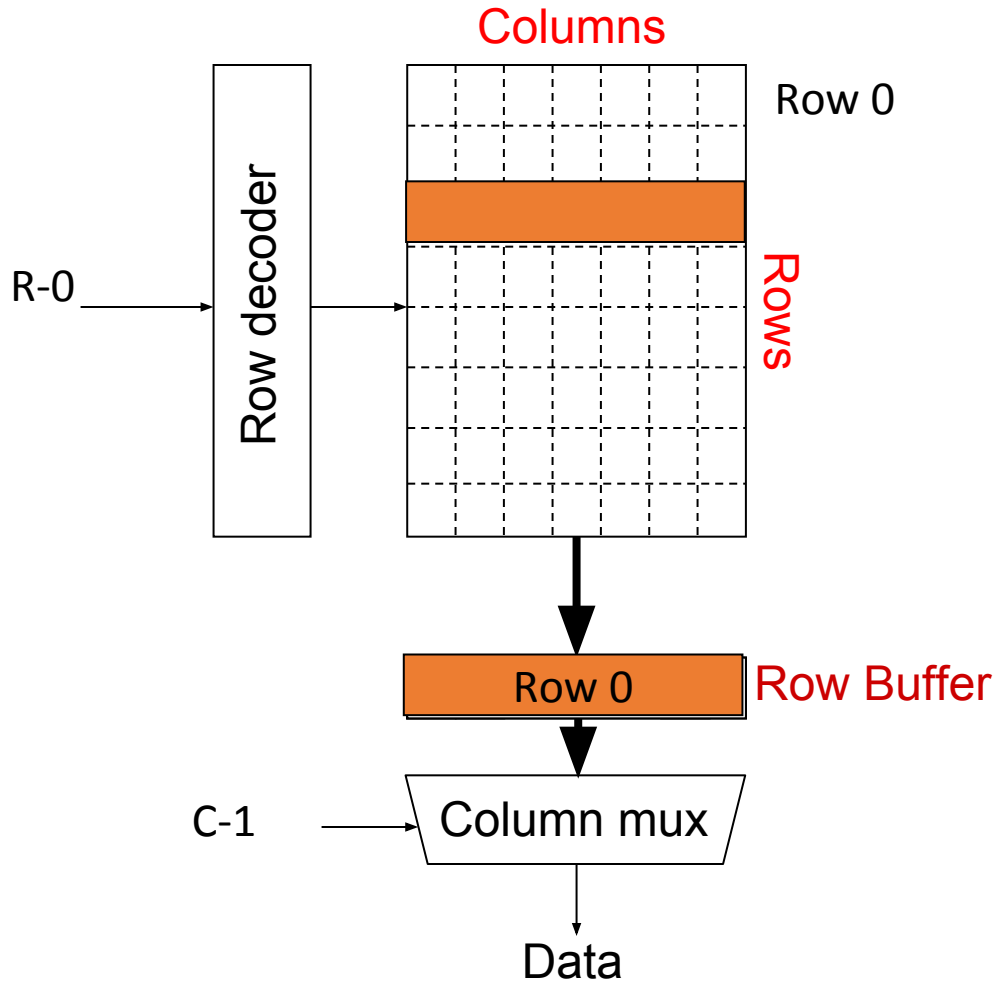
Memory Access Scheduling: FCFS

(R-0, C-4)
(R-2, C-0)
(R-0, C-1)



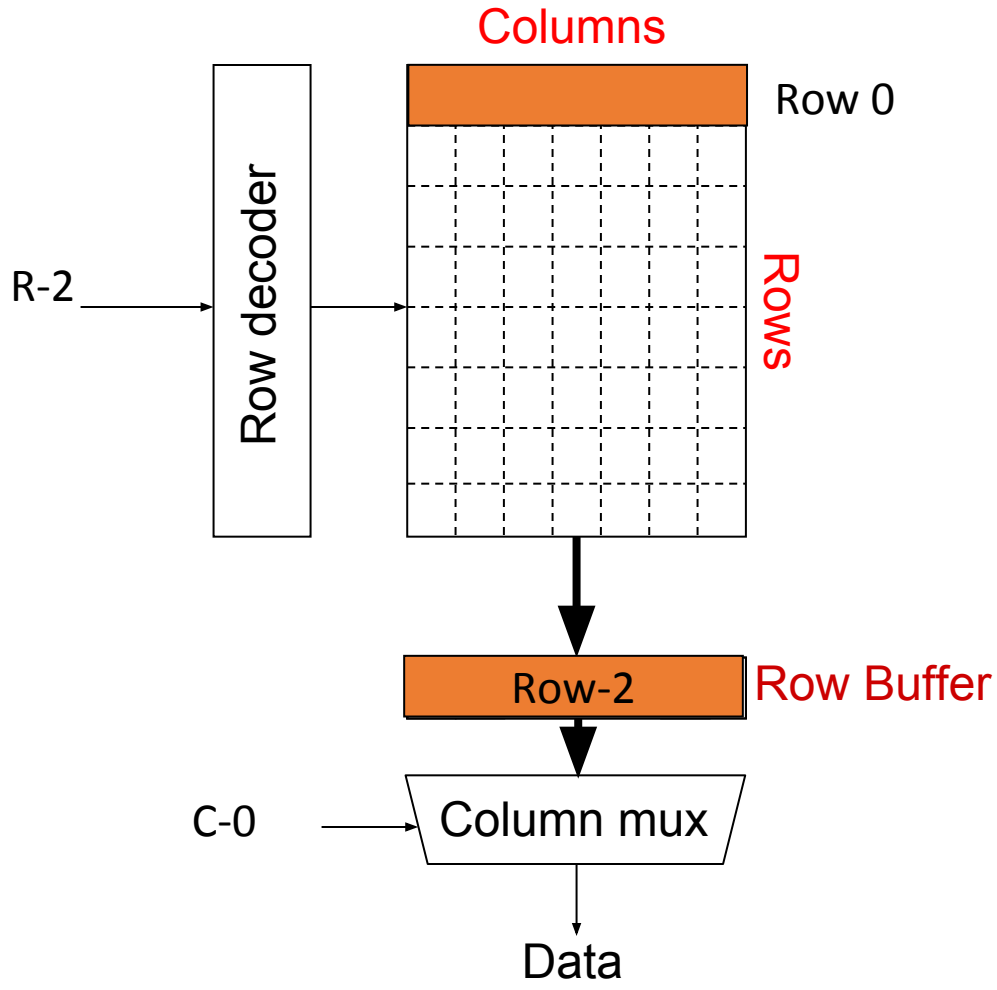
Memory Access Scheduling: FCFS

(R-0, C-5)
(R-0, C-4)
(R-2, C-0)



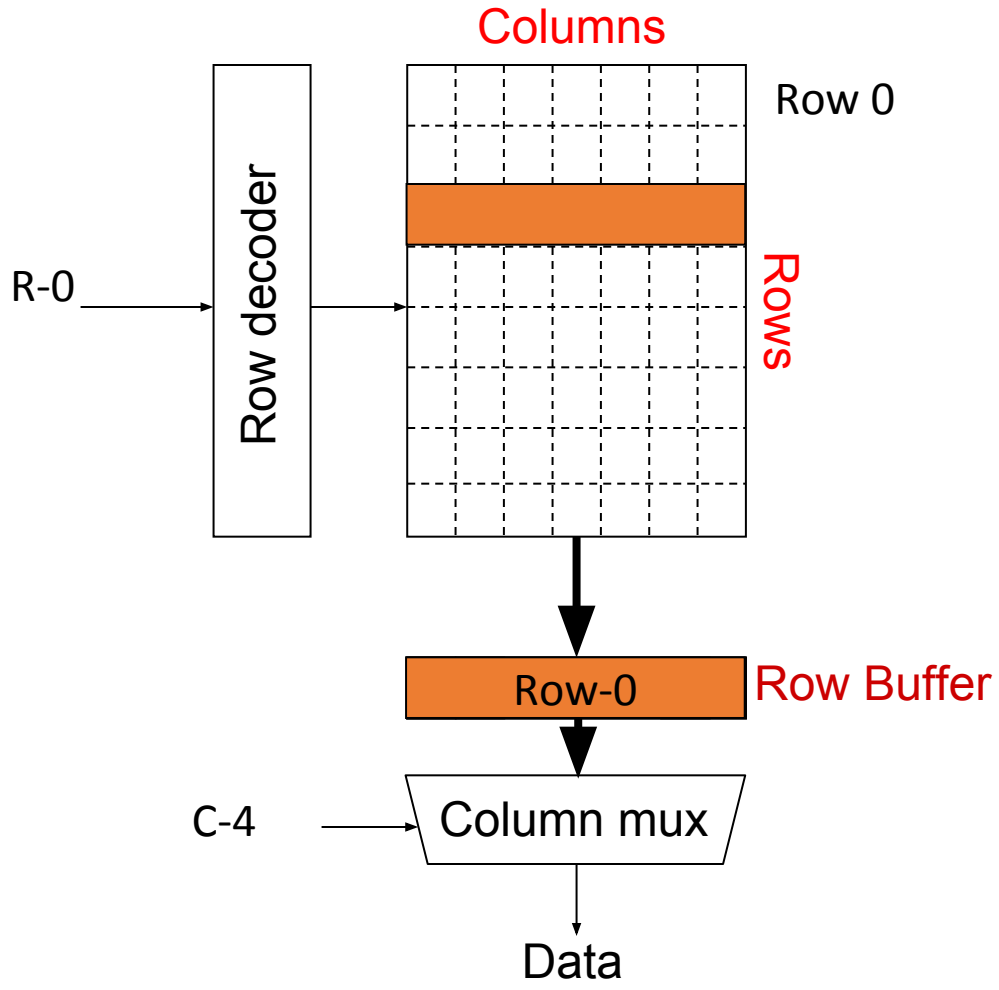
Memory Access Scheduling: FCFS

(R-0, C-5)
(R-0, C-4)

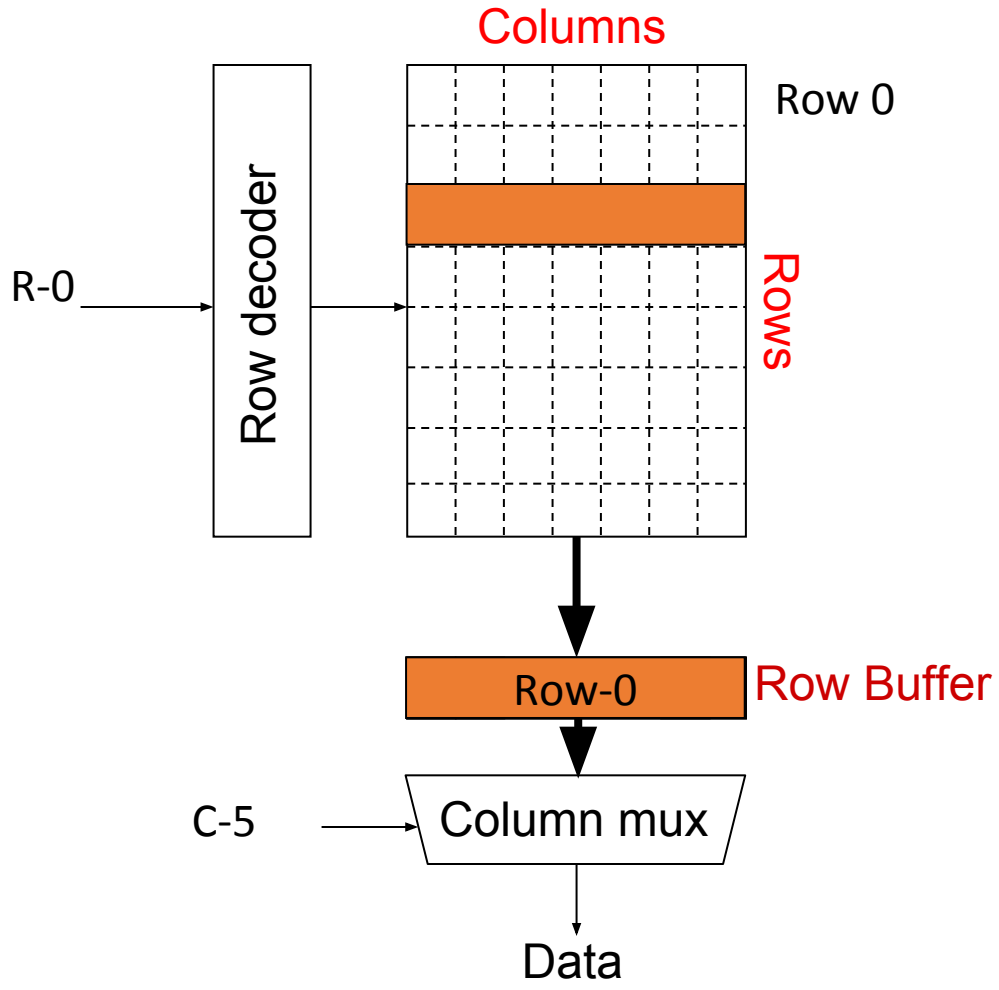


Memory Access Scheduling: FCFS

(R-0, C-5)



Memory Access Scheduling: FCFS

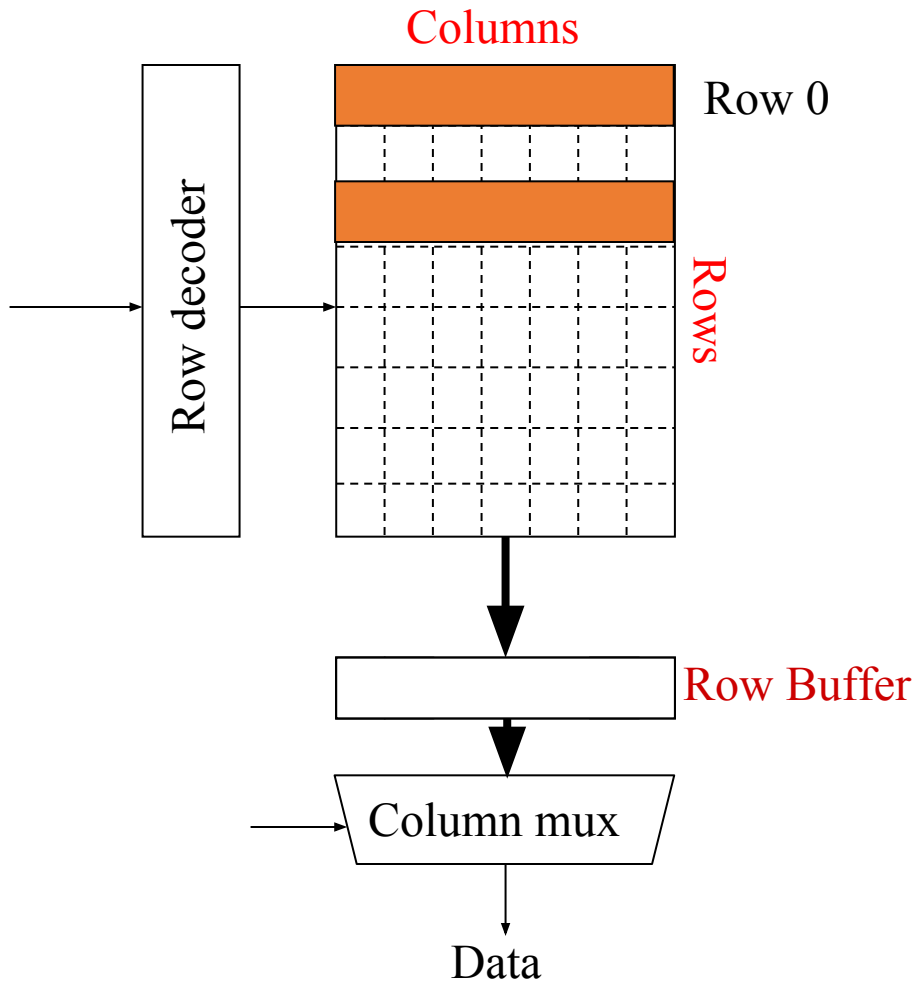


Memory Access Scheduling: FR-FCFS

- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer
- Commonly used scheduling policy (FR-FCFS)
  1. Row-hit (column) first: Serve row-hit memory accesses request first
  2. Oldest-first: Then serve older memory accesses request first
- This scheduling policy aims to maximize DRAM throughput

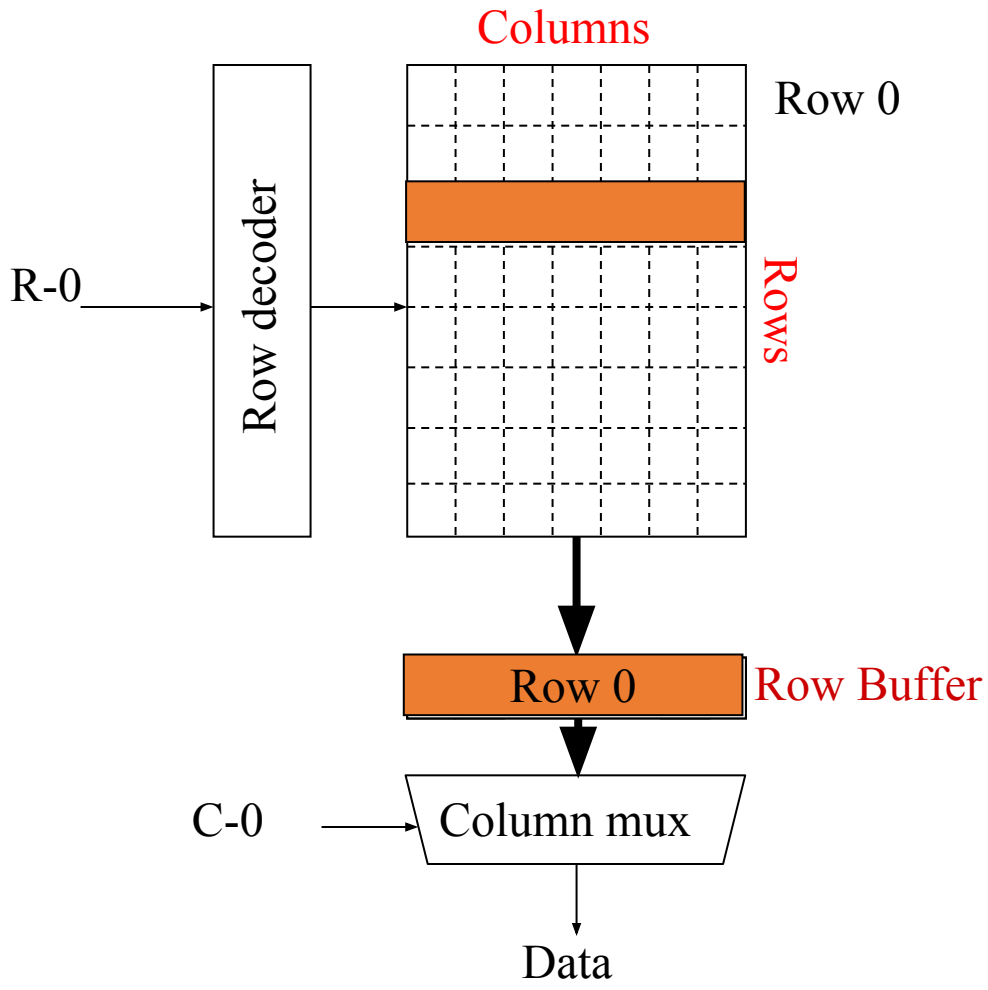
Memory Access Scheduling: FR-FCFS

(R-0, C-4)
(R-2, C-0)
(R-0, C-1)
(R-0, C-0)



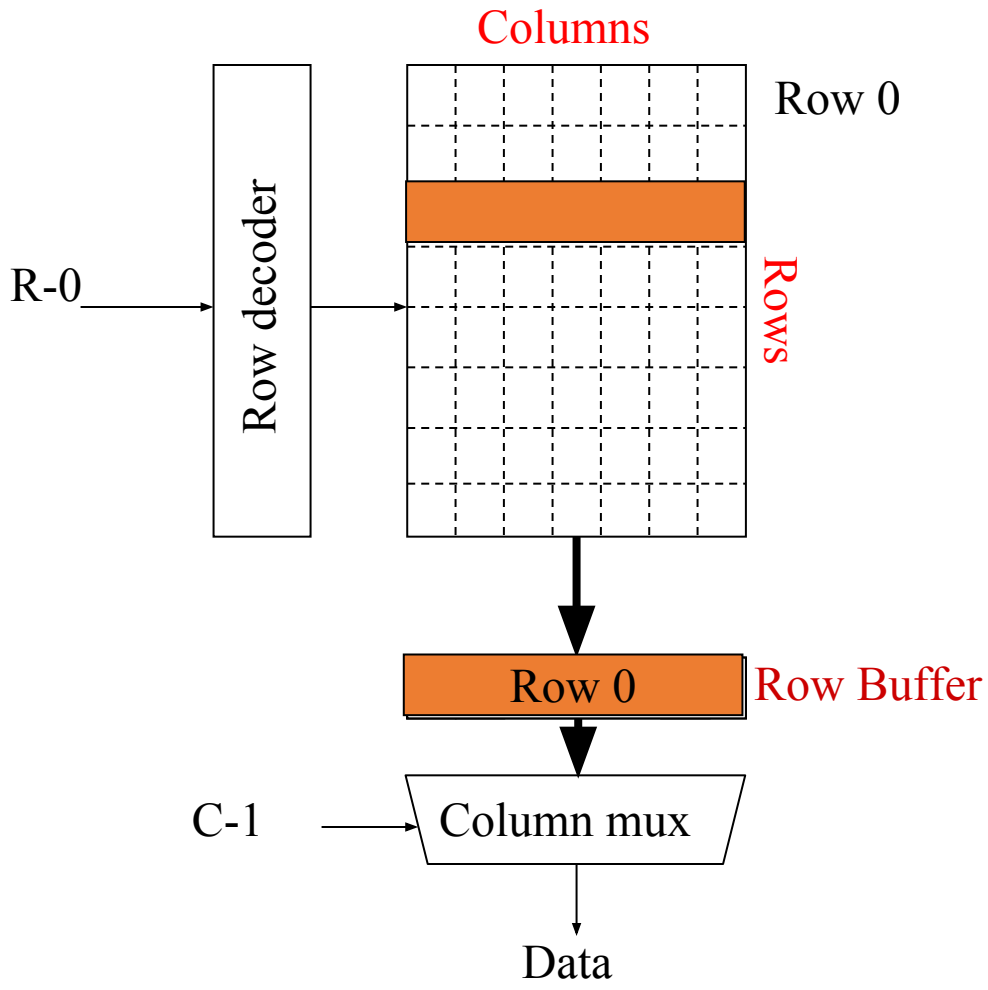
Memory Access Scheduling: FR-FCFS

(R-0, C-4)
(R-2, C-0)
(R-0, C-1)



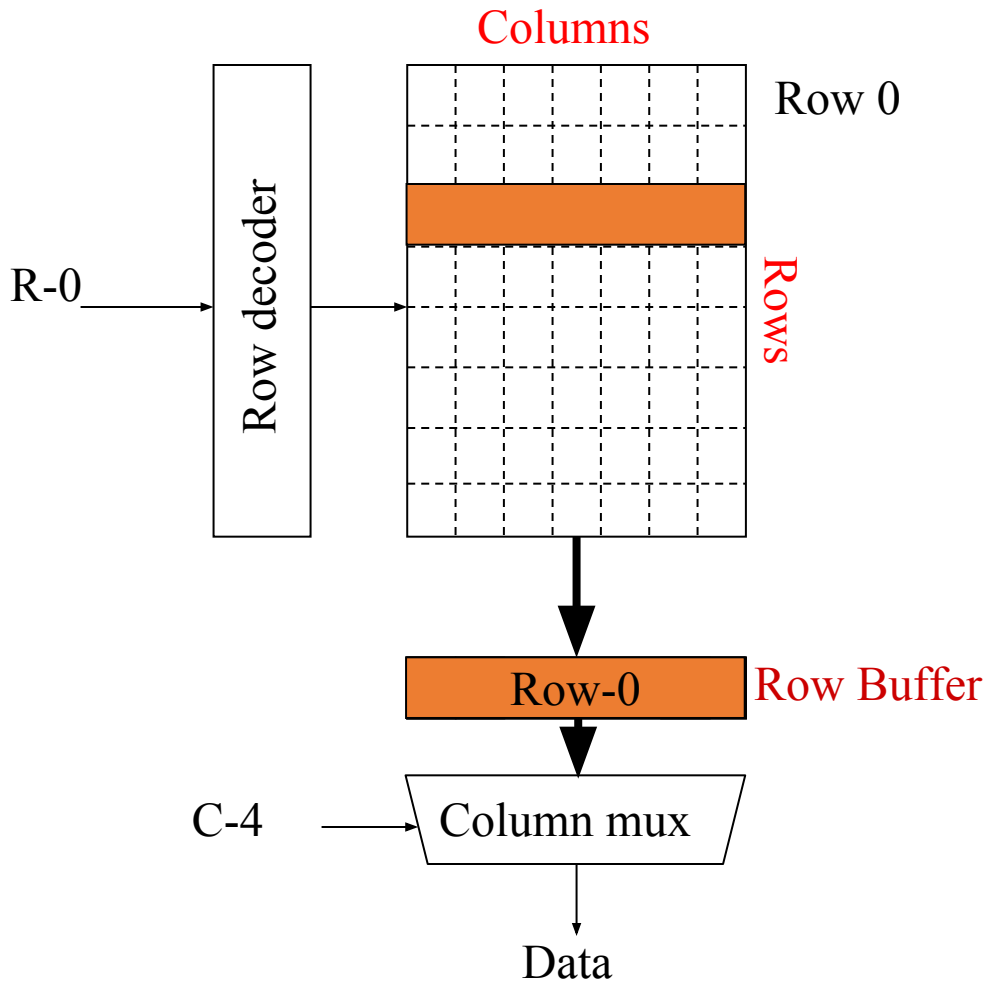
Memory Access Scheduling: FR-FCFS

(R-0, C-5)
(R-0, C-4)
(R-2, C-0)

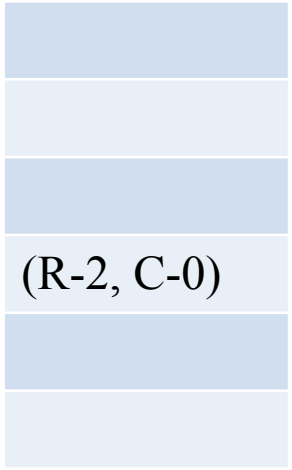


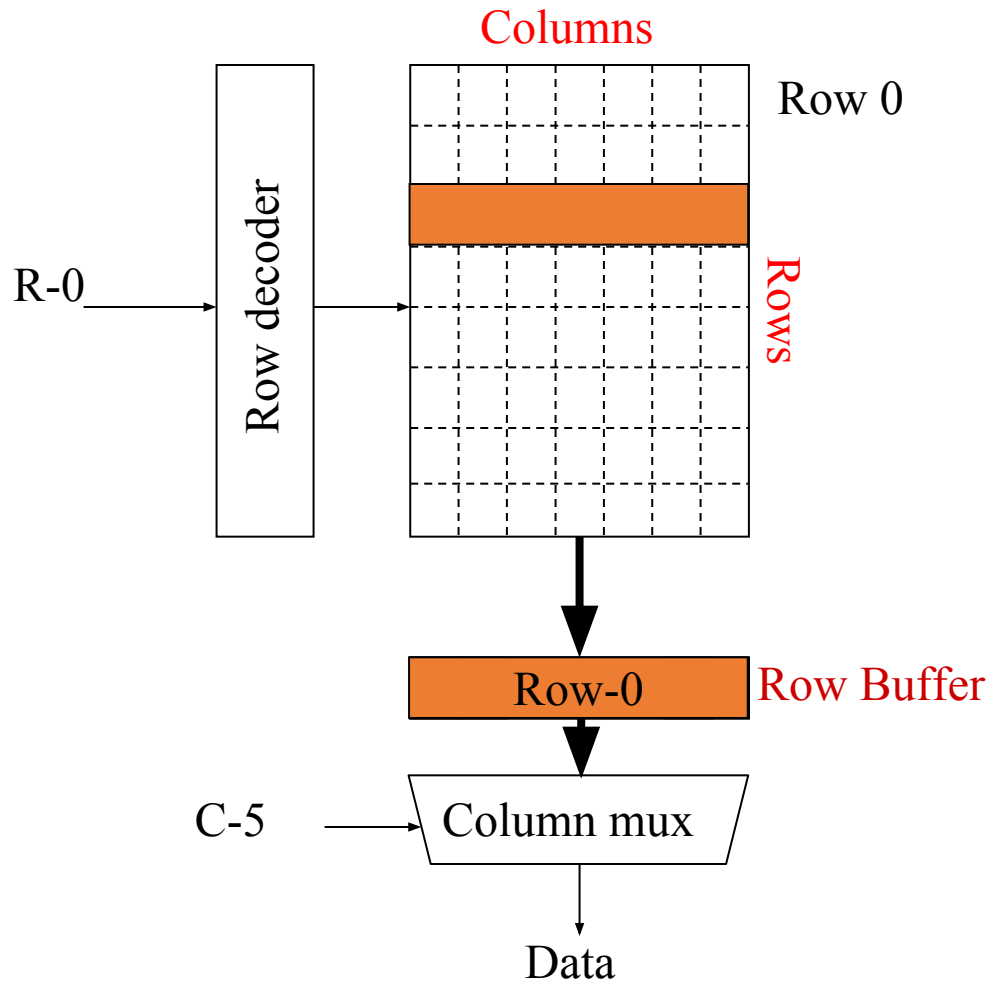
Memory Access Scheduling: FR-FCFS

(R-0, C-5)
(R-2, C-0)

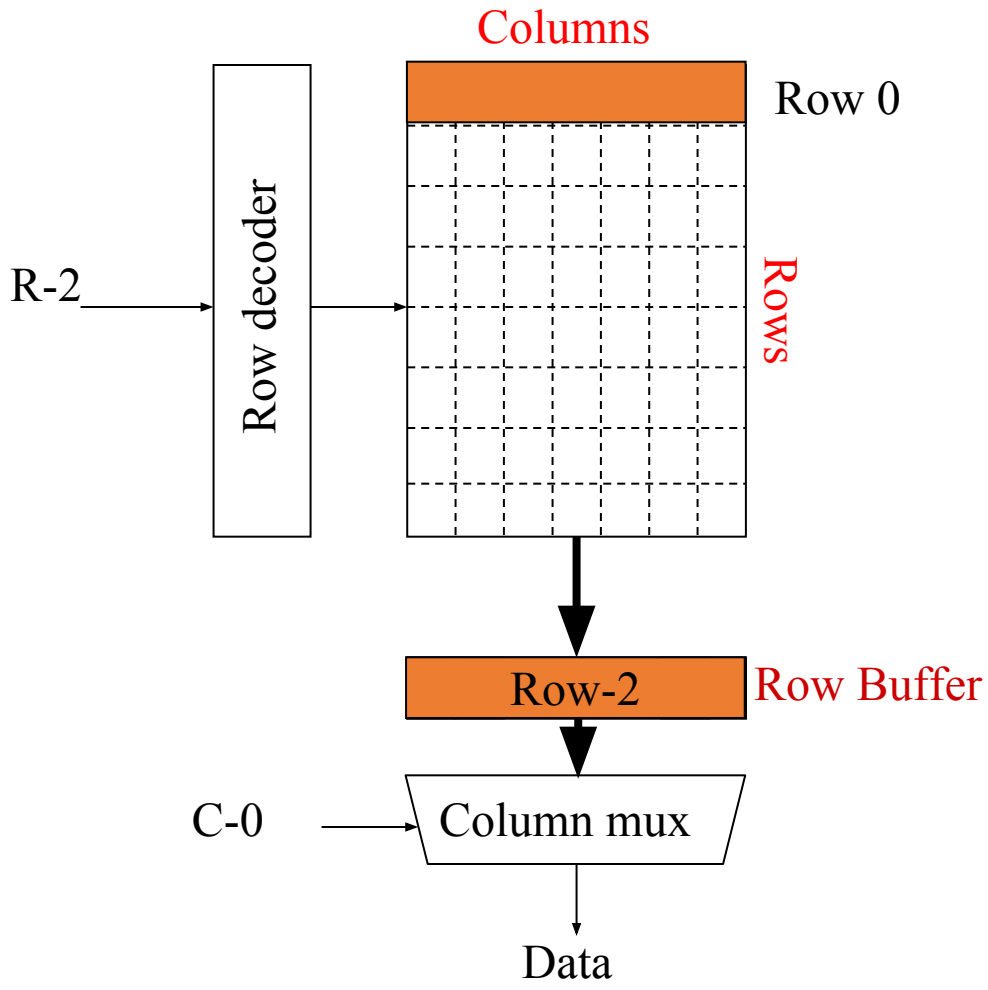


Memory Access Scheduling: FR-FCFS





Memory Access Scheduling: FR-FCFS



Example

- A DRAM controller is using a closed page row buffer management policy. Assume that it takes 20-cycles for a row to be transferred from the storage array to row buffer after ACT (A) is enabled, and 10 cycles for a data to be moved through data bus to memory controller after CAS is enabled. Minimum 15-cycles are needed between a PRE (P) and ACT (A) signal. The controller is using FR-FFCFS scheduling algorithm. Consider the following memory requests that came to bank 2 of the DRAM controller. Each request is denoted by (R<n>,C<m>,T<p>), Where the request is arrived at p-th clock foe word at n-th row and m-th column.
- (R15C78T10), (R20C50T20), (R24C20T35), (R20C78T80), (R2C12T150), (R24C50T180), (R35C79T200), (R10C50T210), (R24C14T220), (R25C18T250)

Example

Order	Time	R,C	PRE(P)	ACT(A)	CAS	PRE(P)	Total_Count
1	10	15, 78		10	30		40
2	20	20, 50	40	55	75		85
4	35	24, 20	95	110	130	140	140
3	80	20, 78			85		95
5	150	2, 12		155	175		185
6	180	24, 50	185	200	220		230
8	200	35, 79	240	255	275		285
9	210	10, 50	285	300	320		330
7	220	24, 14			230		240
10	250	25, 18	330	345	365		375

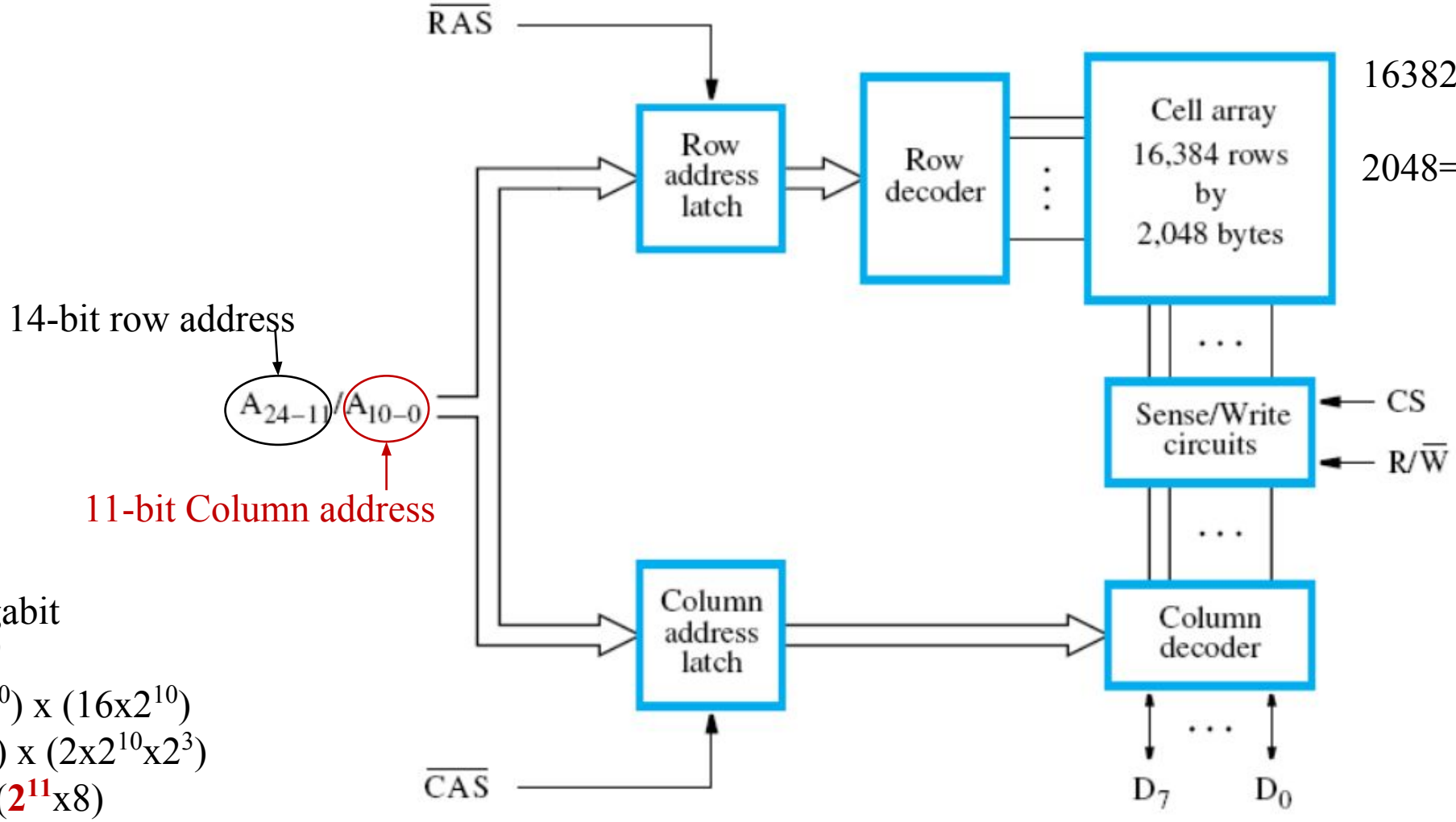
Array to Buffer after ACT 20-cycles

Data bus to mem. Controller 10-cycles

Time needed between a PRE & ACT is 15-cycles

140 + 15 = 155

Internal organization of a 256-Megabit DRAM (32M × 8) dynamic memory chip

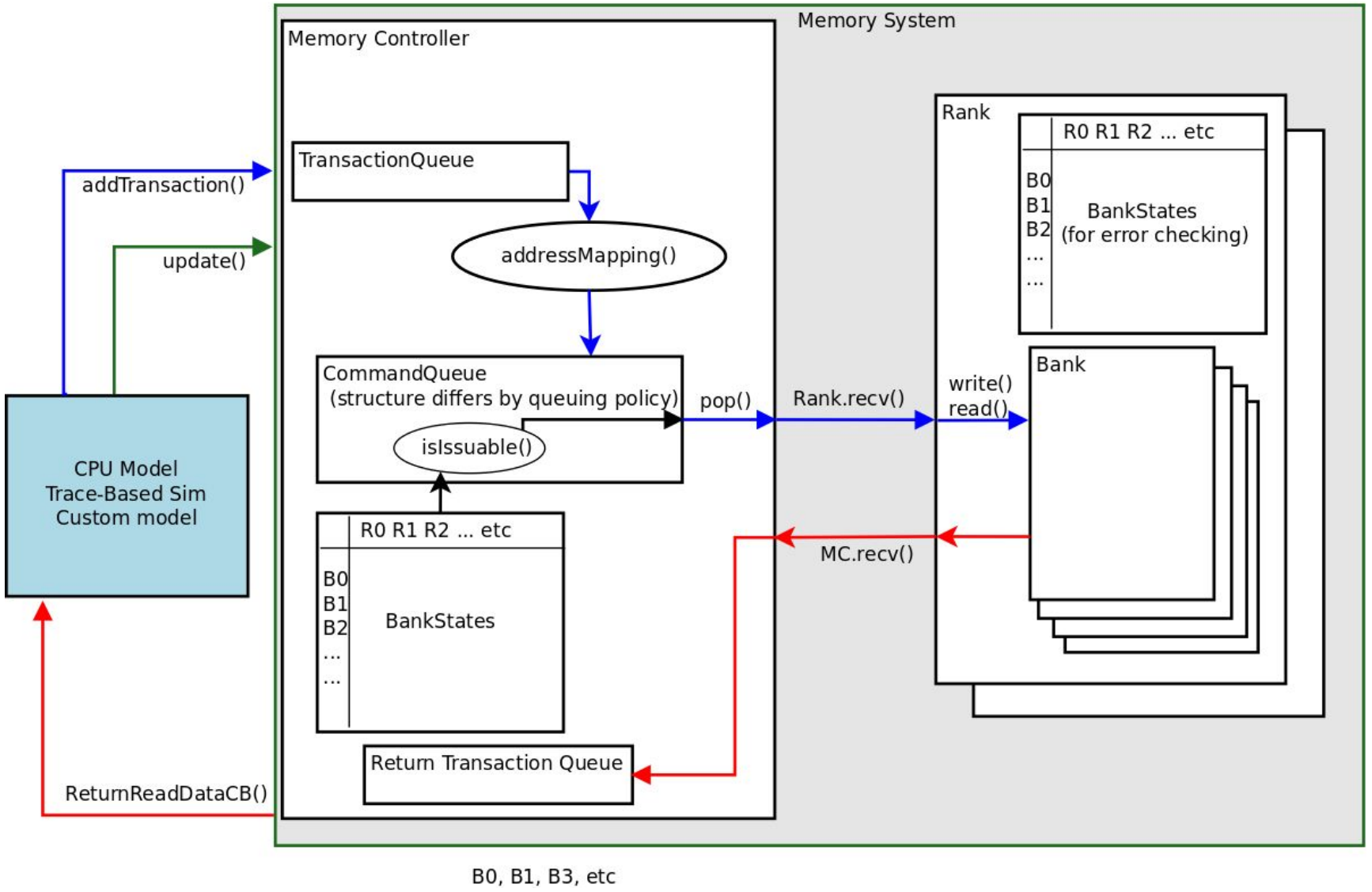


16382= 16x1024 = 2^4x2^{10}

2048=2x1024=2x2^{10}

256 Megabit
=2^8 x 2^{20}
=(16x 2^{10}) x (16x2^{10})
=(2^4x2^{10}) x (2x2^{10}x2^3)
=(**2^{14}**) x (**2^{11}**x8)

Overall flow



Summary

- Memory idealism
- Array organization of memory
- How to store data
- SRAM
- DRAM
- DRAM’s Organization
- DRAM’s request to commands
- Address mapping
- Refreshing
- Scheduling