18-447
Computer Architecture
Lecture 22: Main Memory

Prof. Onur Mutlu
Carnegie Mellon University
Spring 2013, 3/26/2014

Enabling High Bandwidth Memories

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?

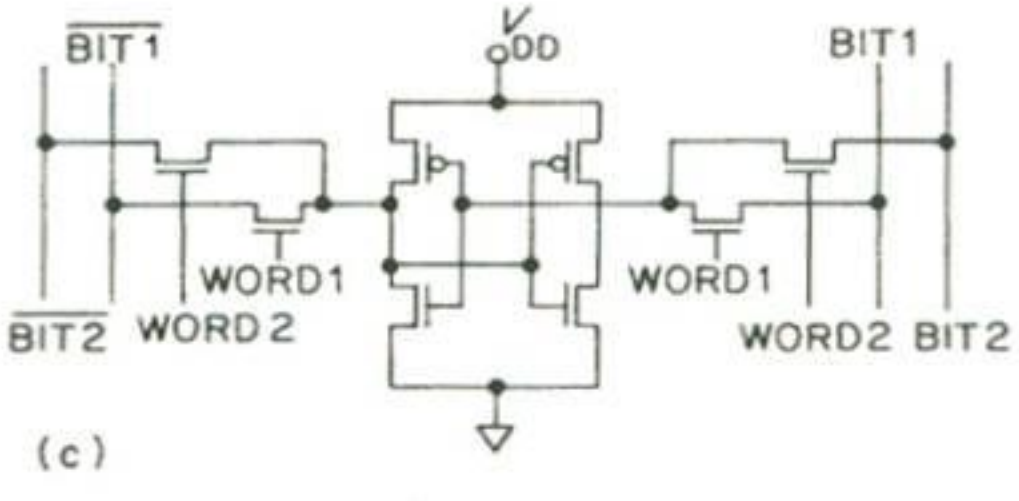
- Solutions:
  - true multi-porting
  - virtual multi-porting (time sharing a port)
  - multiple cache copies
  - banking (interleaving)

Handling Multiple Accesses per Cycle (I)

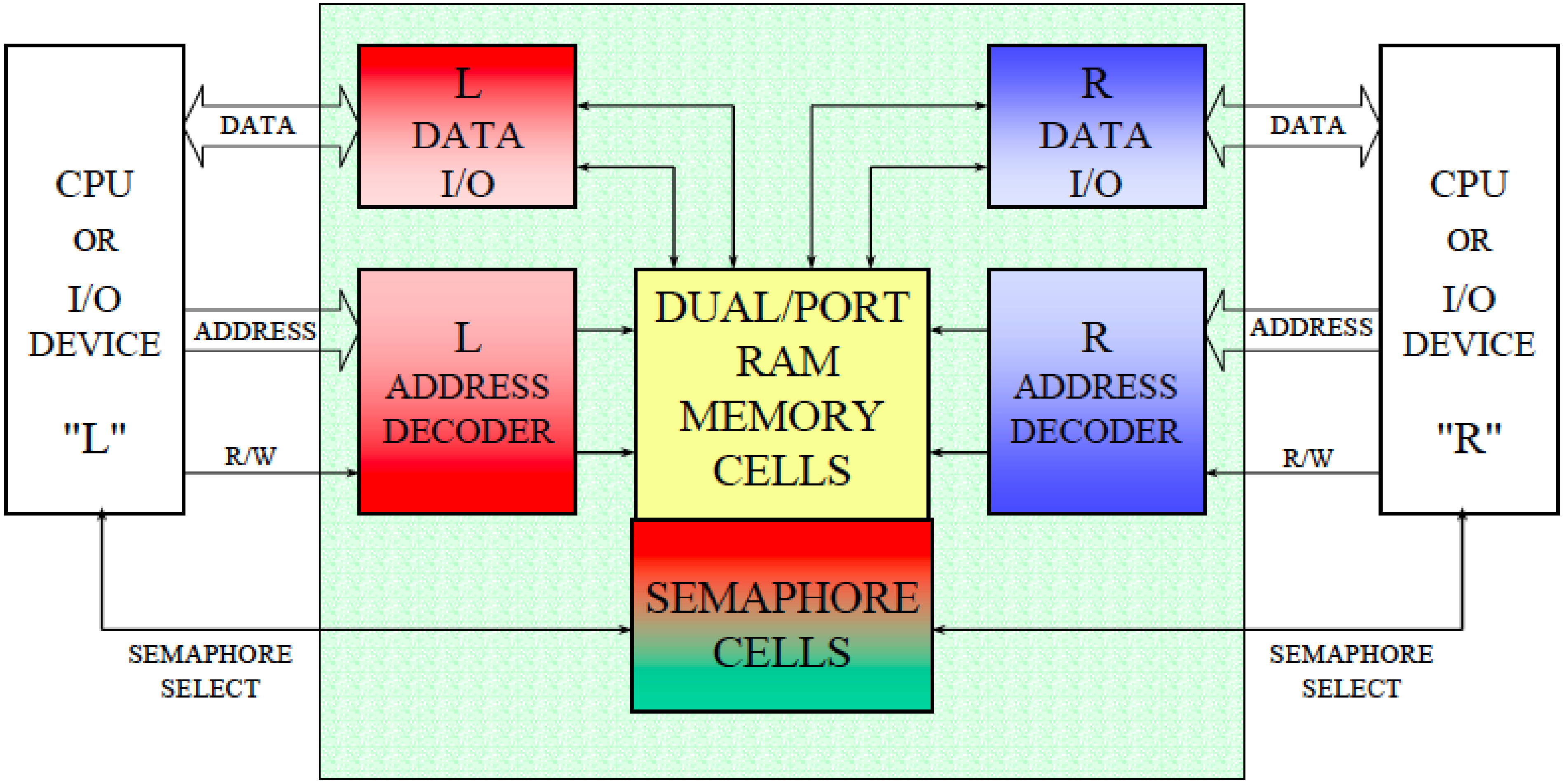
- True multiporting
  - Each memory cell has multiple read or write ports

  + Truly concurrent accesses (no conflicts regardless of address)

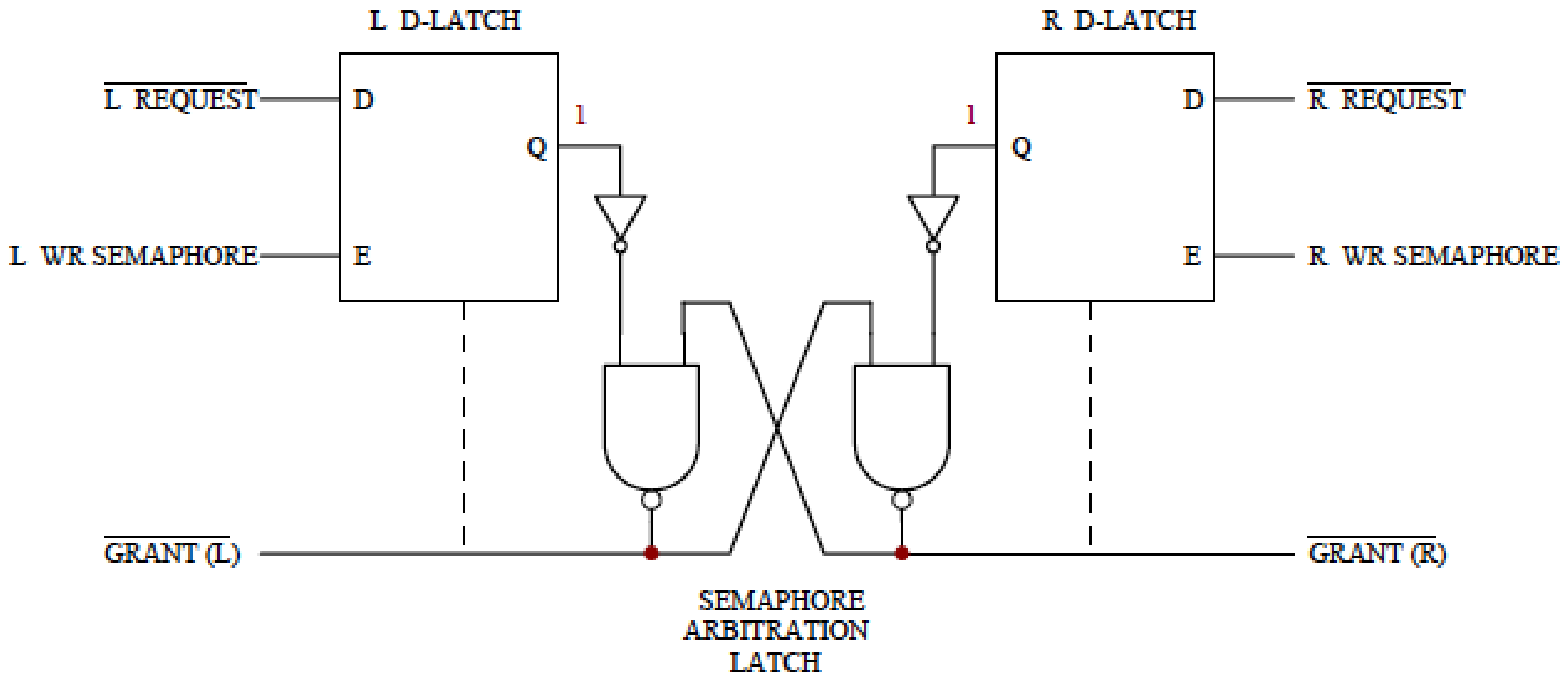
  -- Expensive in terms of latency, power, area
  - What about read and write to the same location at the same time?
    - Peripheral logic needs to handle this



Peripheral Logic for True Multiporting



Peripheral Logic for True Multiporting


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

Handling Multiple Accesses per Cycle (I)

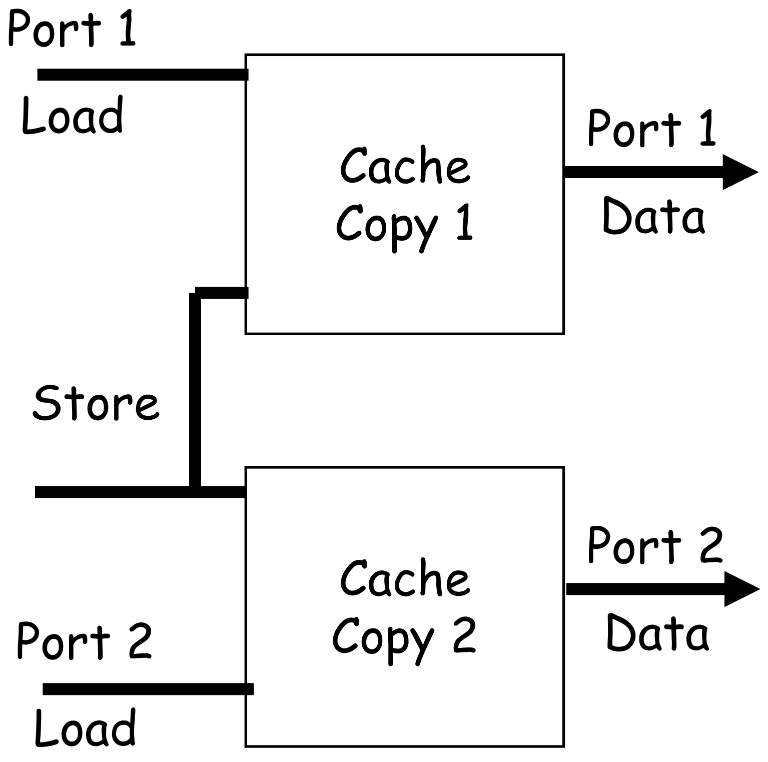
- Virtual multiporting
  - Time-share a single port
  - Each access needs to be (significantly) shorter than clock cycle
  - Used in Alpha 21264
  - Is this scalable?

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"



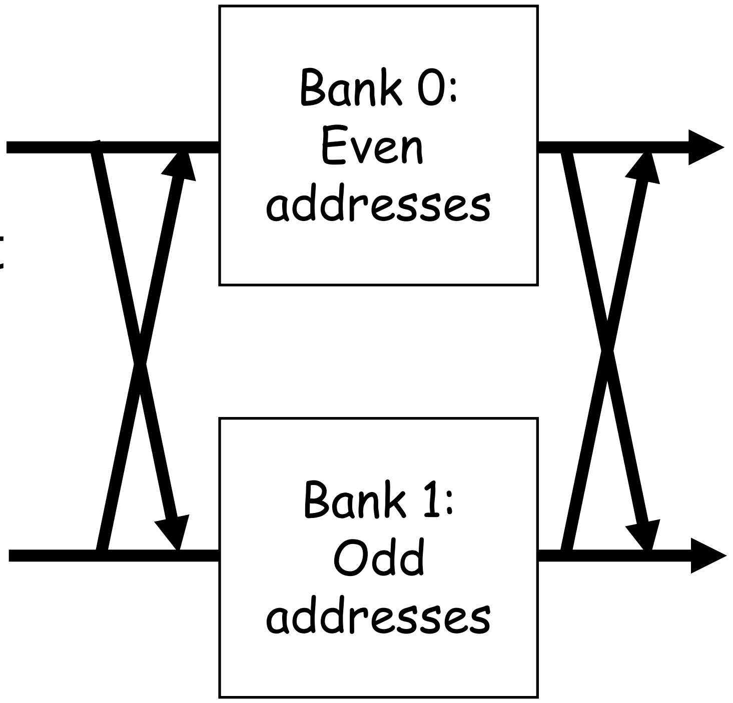
Handling Multiple Accesses per Cycle (III)

- 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?)

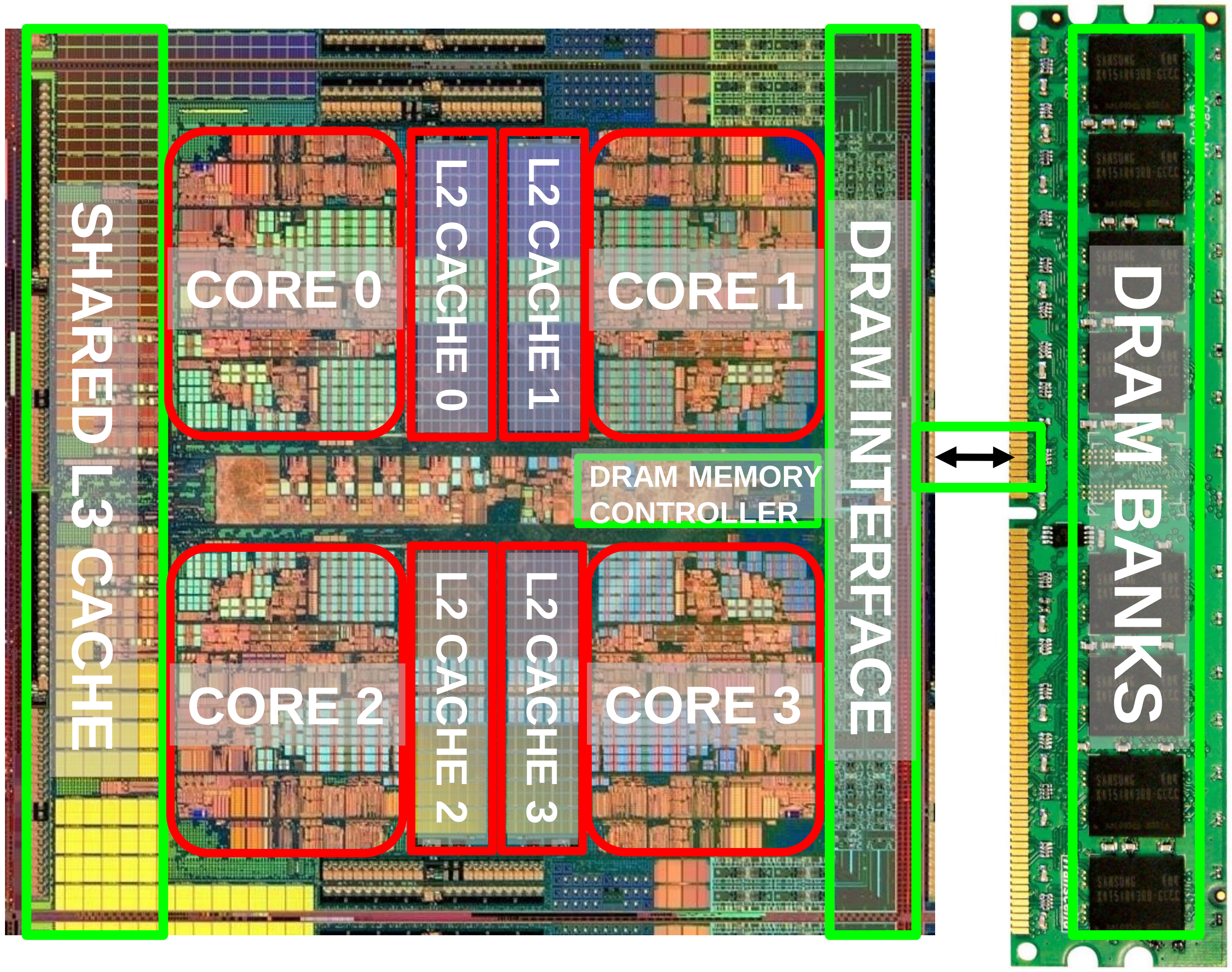
Further Readings on Caching and MLP

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

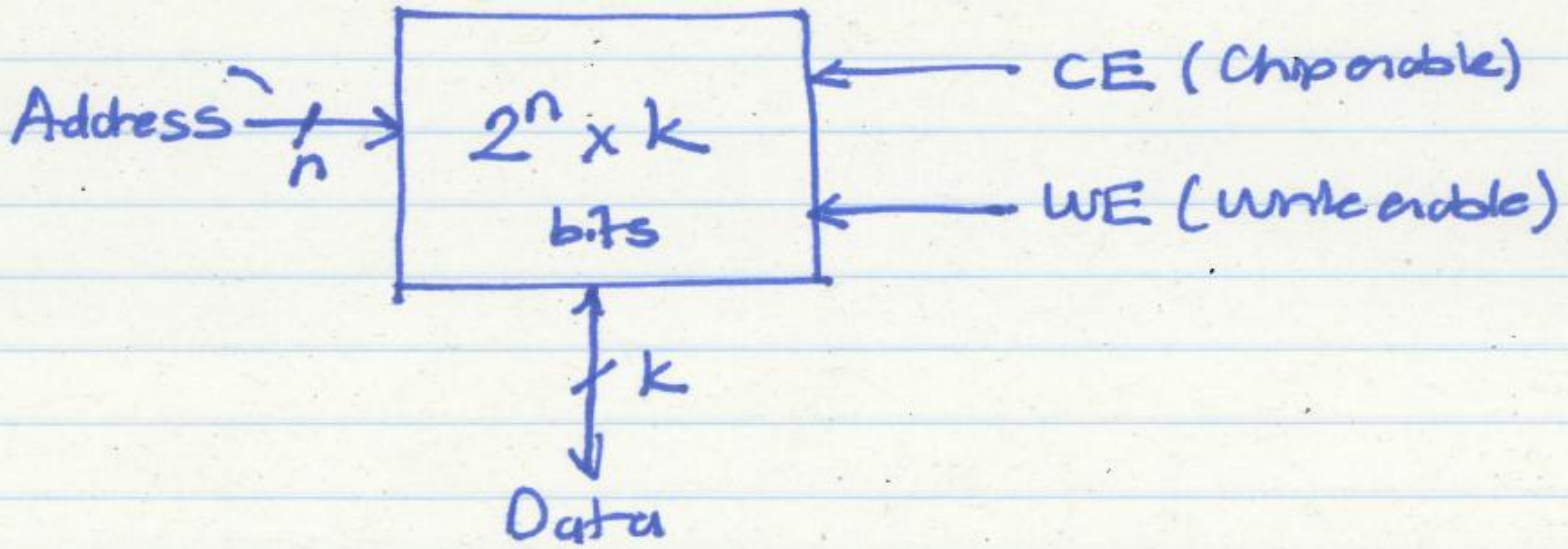
- Glew, "MLP Yes! ILP No!," ASPLOS Wild and Crazy Ideas Session, 1998.

Main Memory

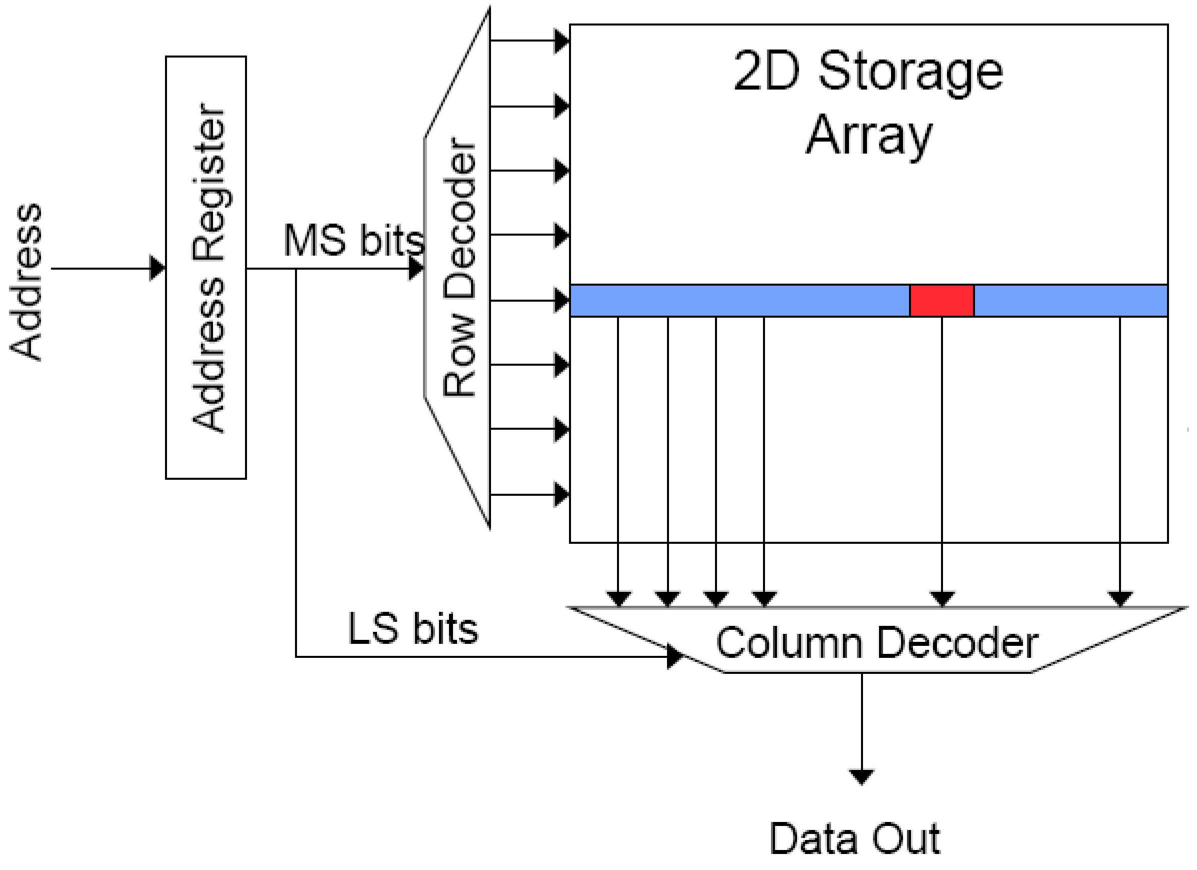
Main Memory in the System



The Memory Chip/System Abstraction



Review: Memory Bank Organization



- Read access sequence:

  1. Decode row address & drive word-lines

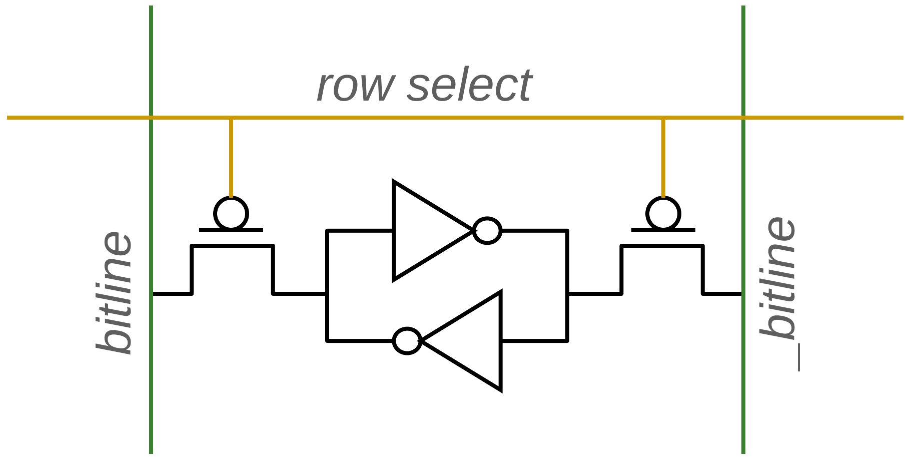
  2. Selected bits drive bit-lines
     - Entire row read

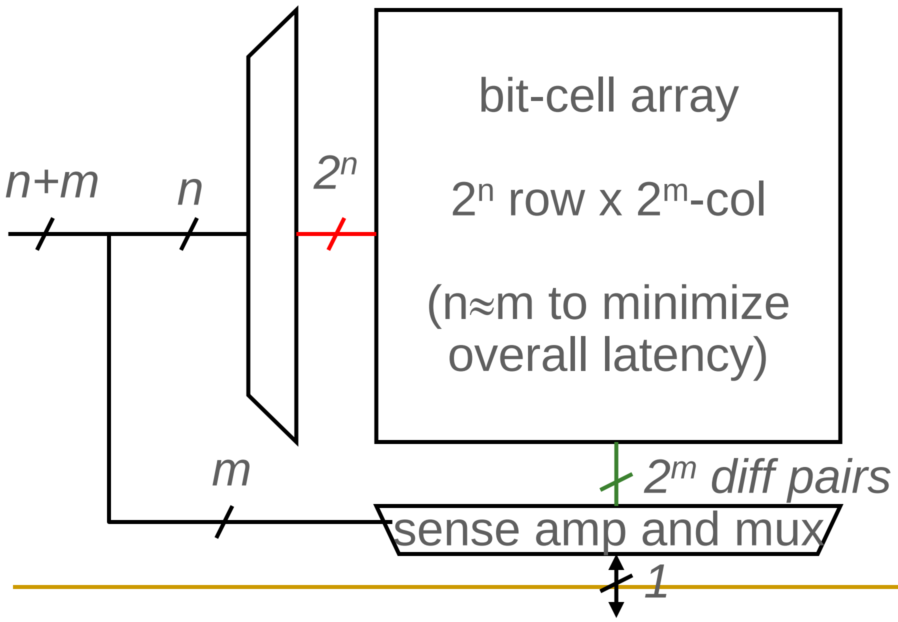
  3. Amplify row data

  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

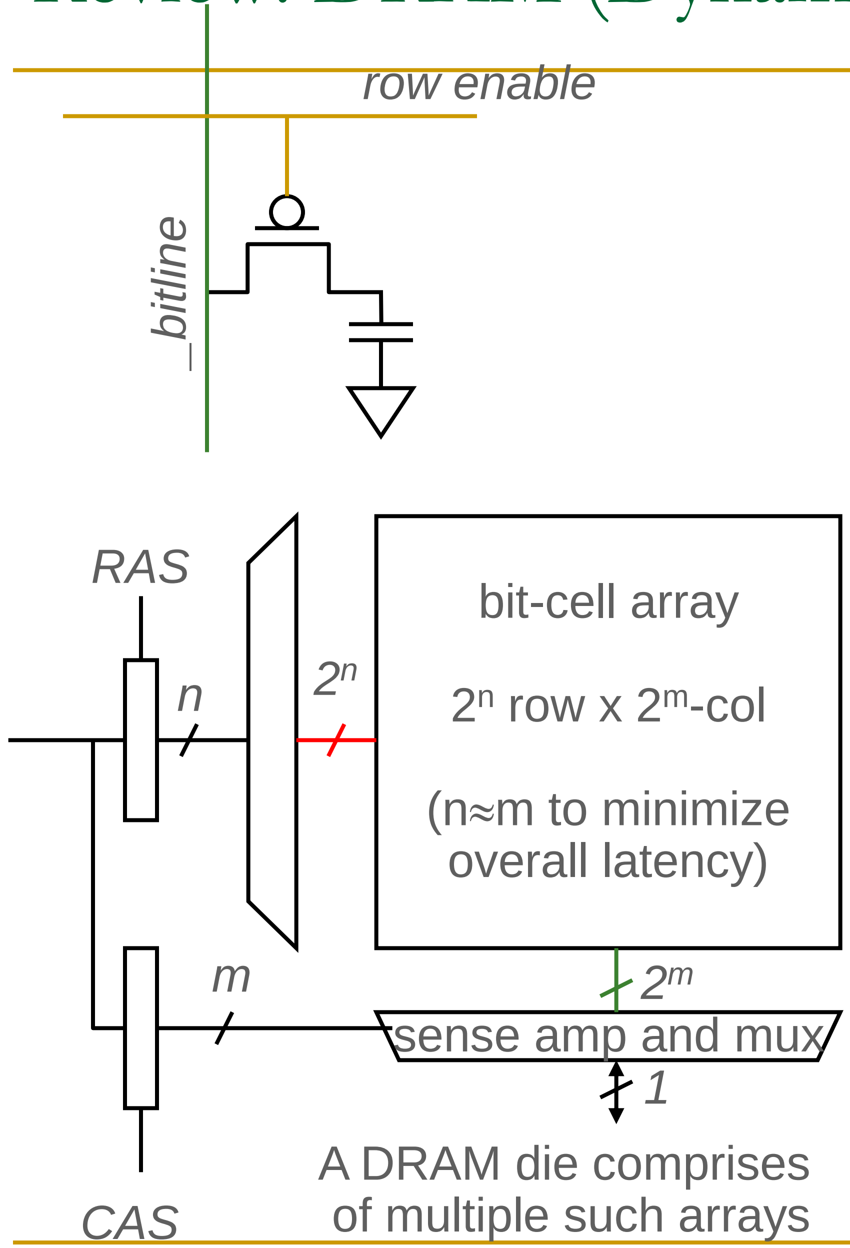
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. diff. sensing and col. select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

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?)

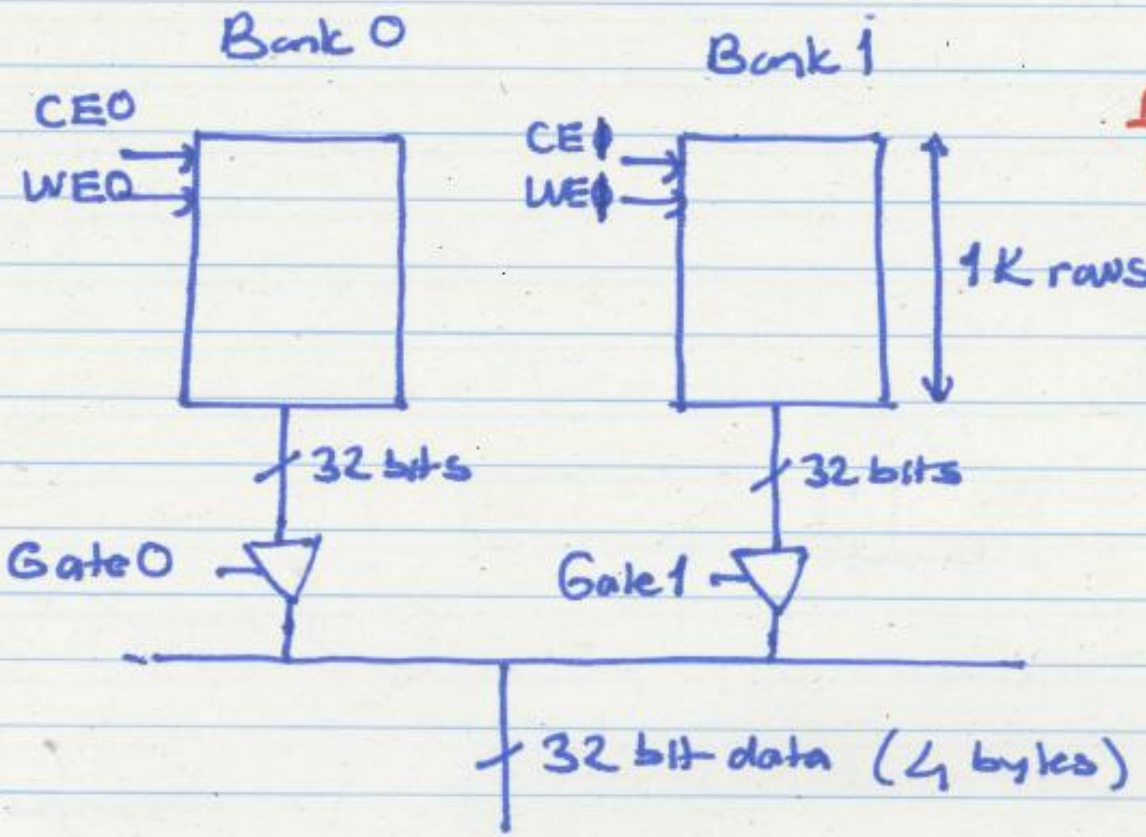
Interleaving

Interleaving (Example)

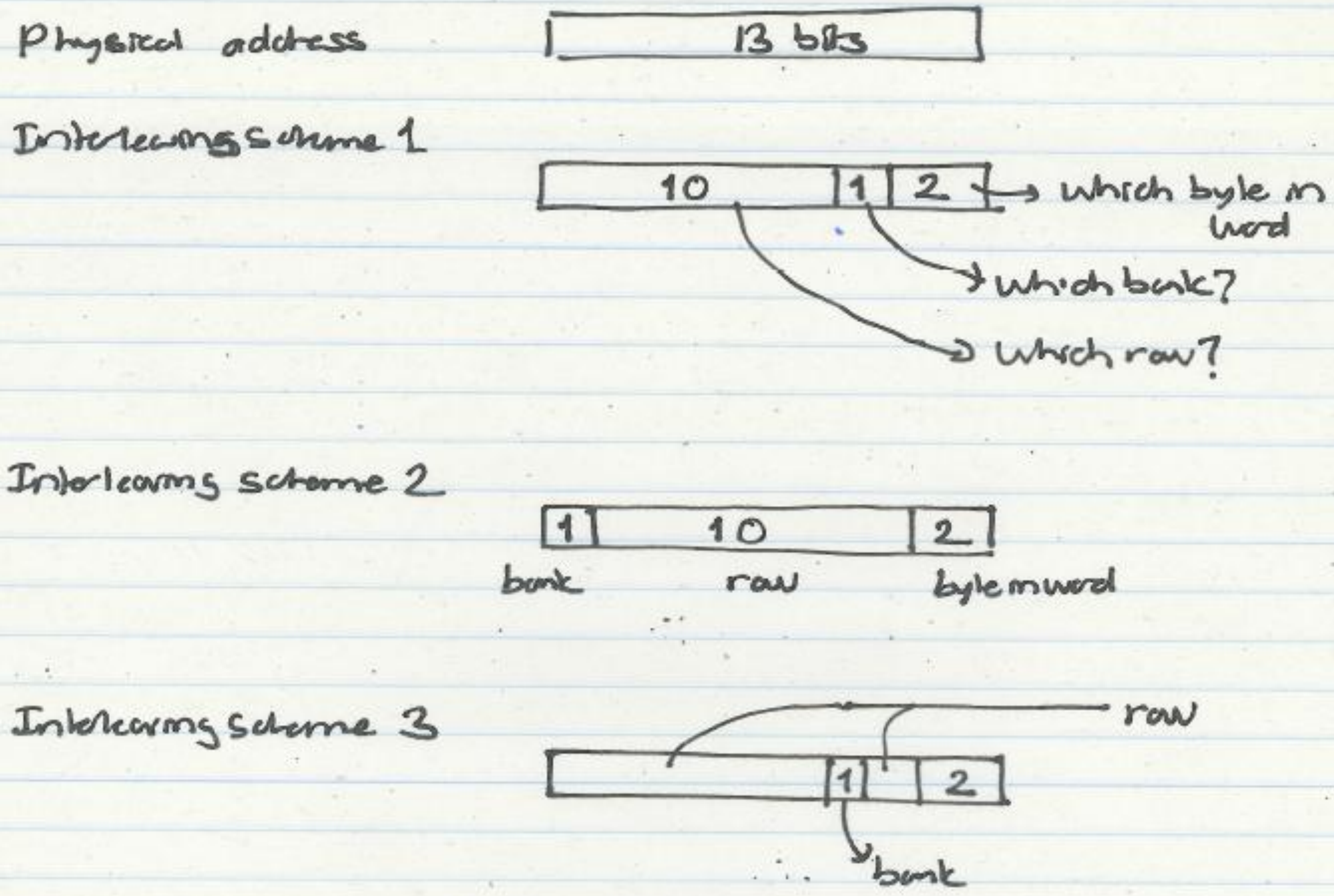
Assume each bank supplies a word.

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

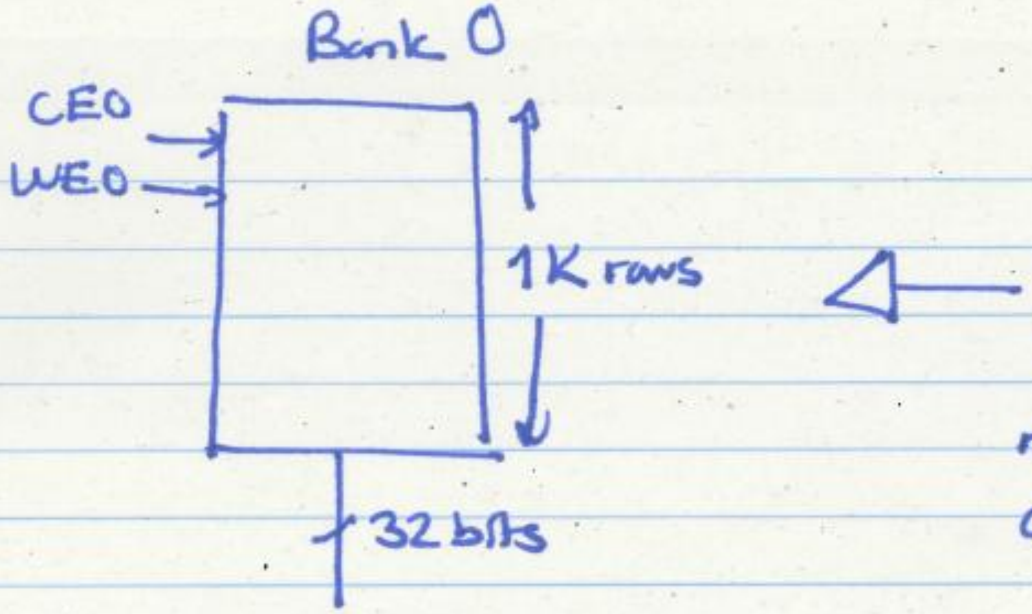
- Remember CRAY-1 with 16 banks
  - 11 cycle bank latency
  - Consecutive words in memory in consecutive banks (word interleaving)
  - 1 access can be started (and finished) per cycle

- 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

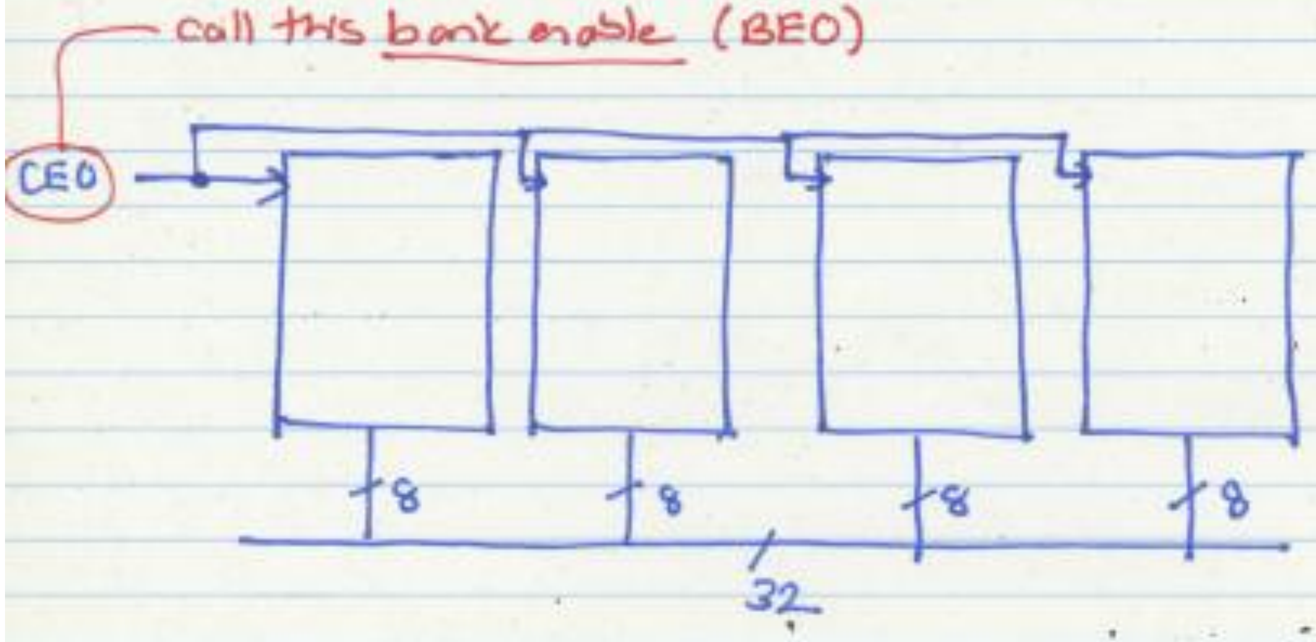
The Bank Abstraction



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

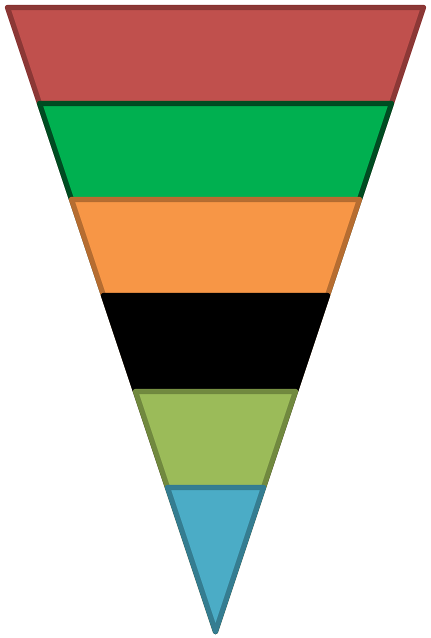
Why? Producing an 8-bit/pin chip cheaper than producing a 32-bit/pin chip

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.

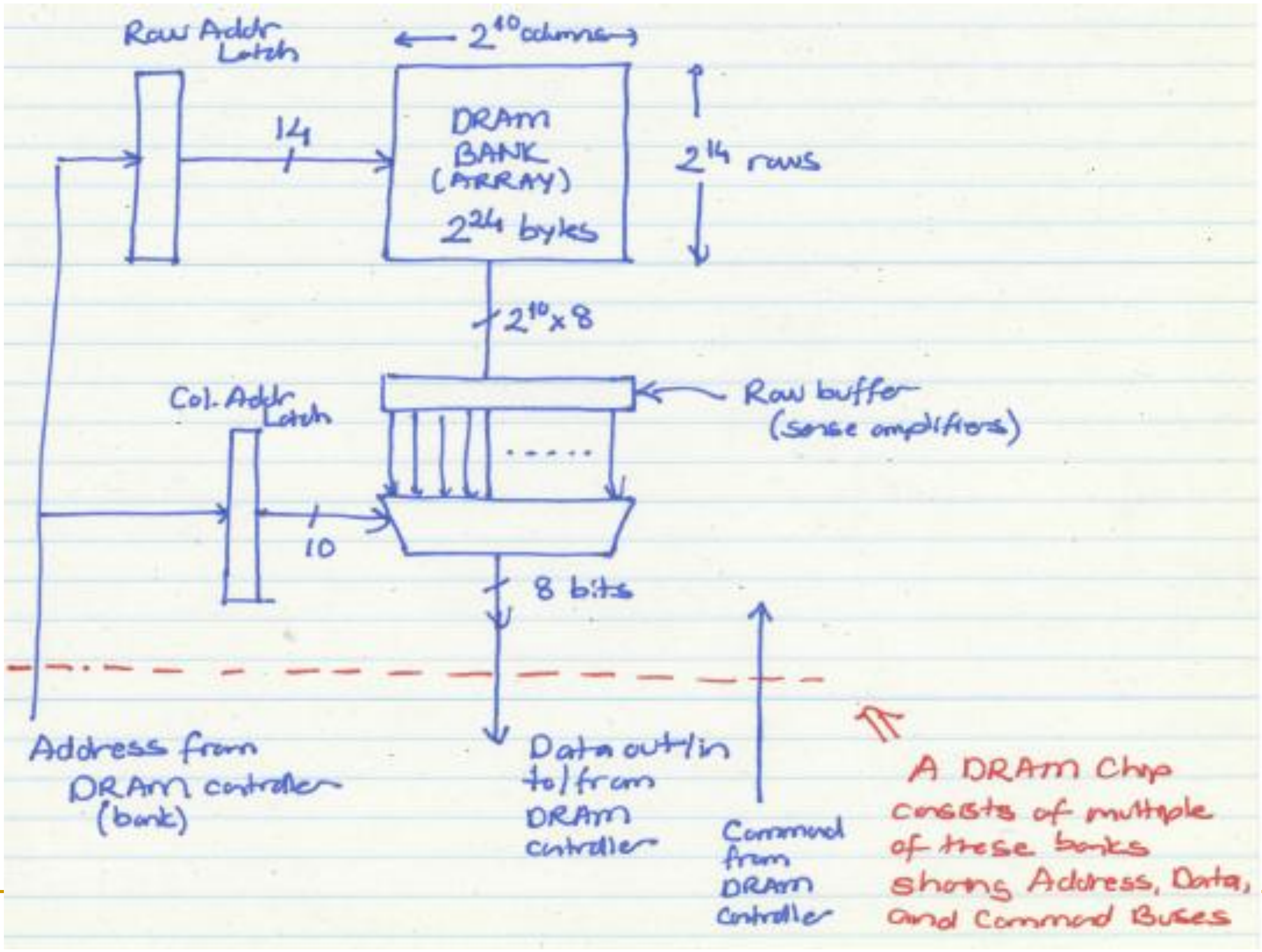
The DRAM Subsystem

DRAM Subsystem Organization

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



The DRAM Bank Structure


Row Addr Latch
2^{10} columns
14
DRAM BANK (ARRAY) 2^{24} bytes
2^{14} rows
$2^{10} \times 8$
Col. Addr Latch
Row buffer (sense amplifiers)
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

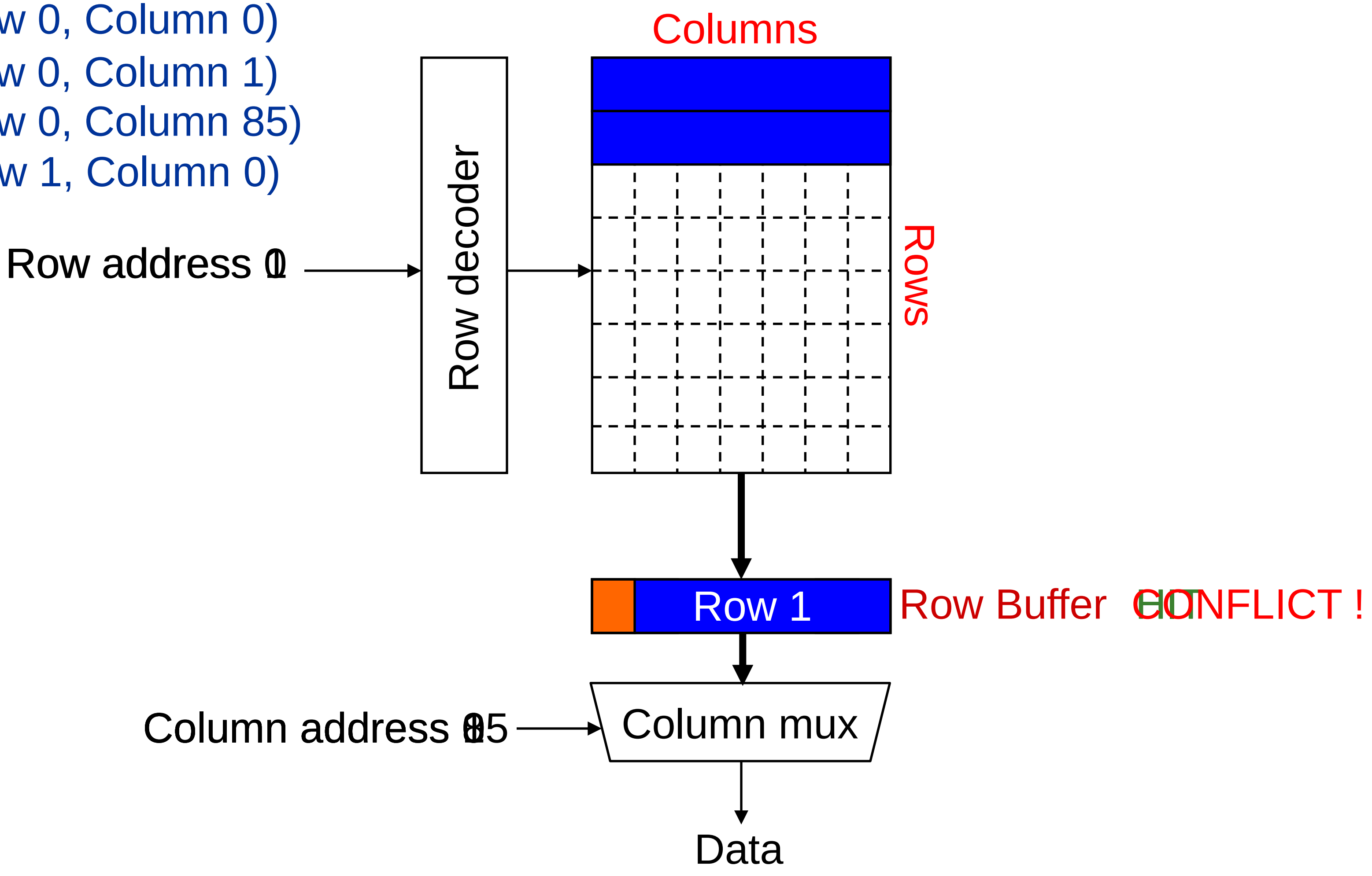
Page Mode DRAM

- 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

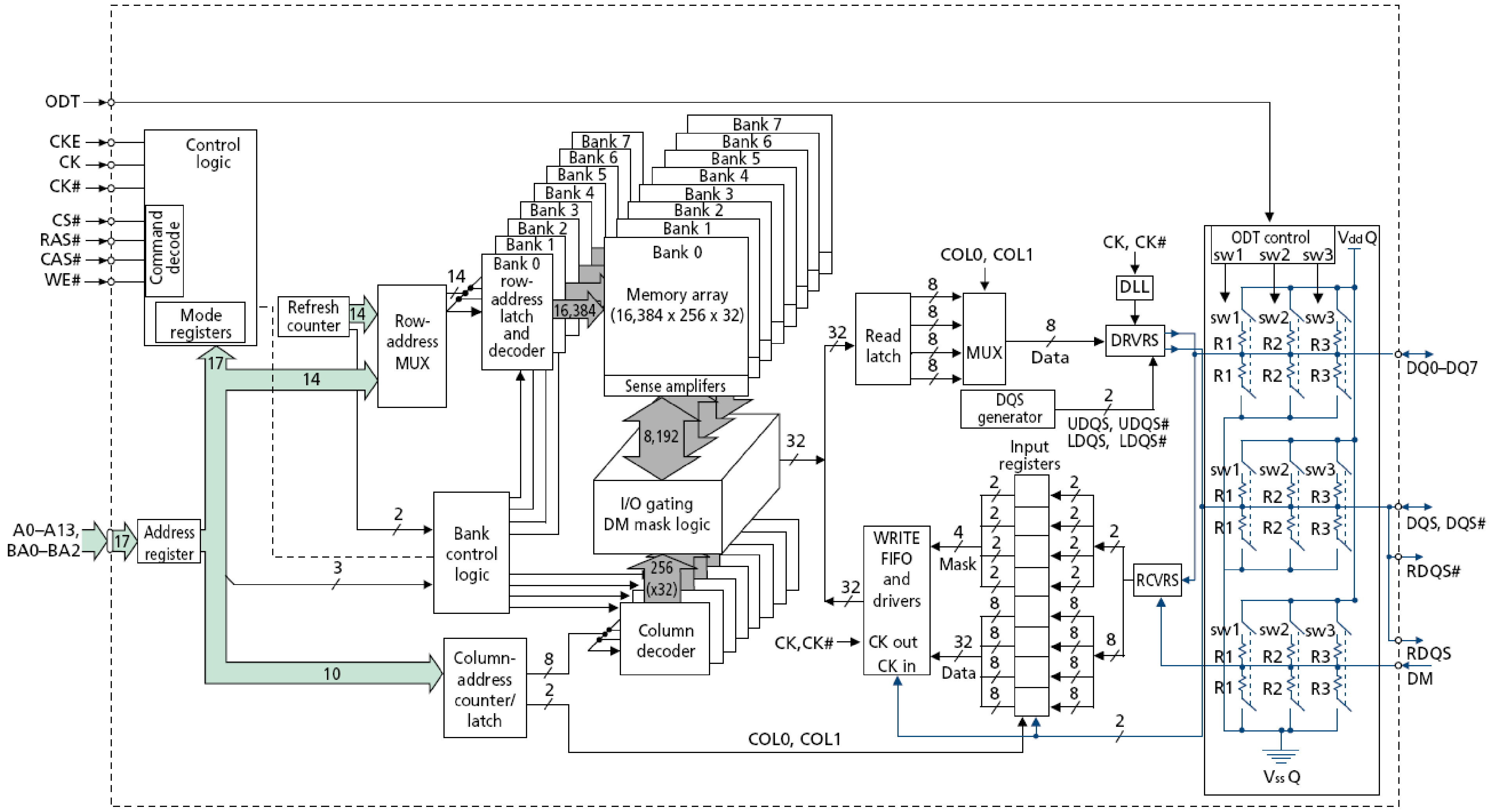
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



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



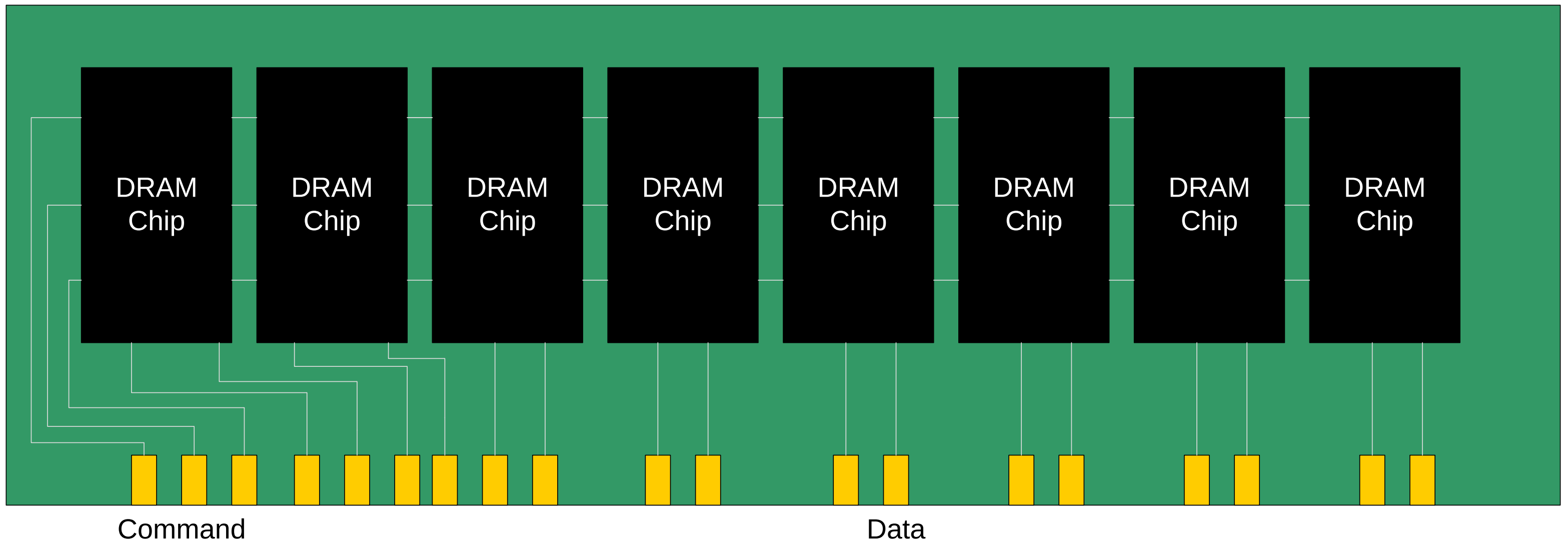
DRAM Rank and Module

- 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

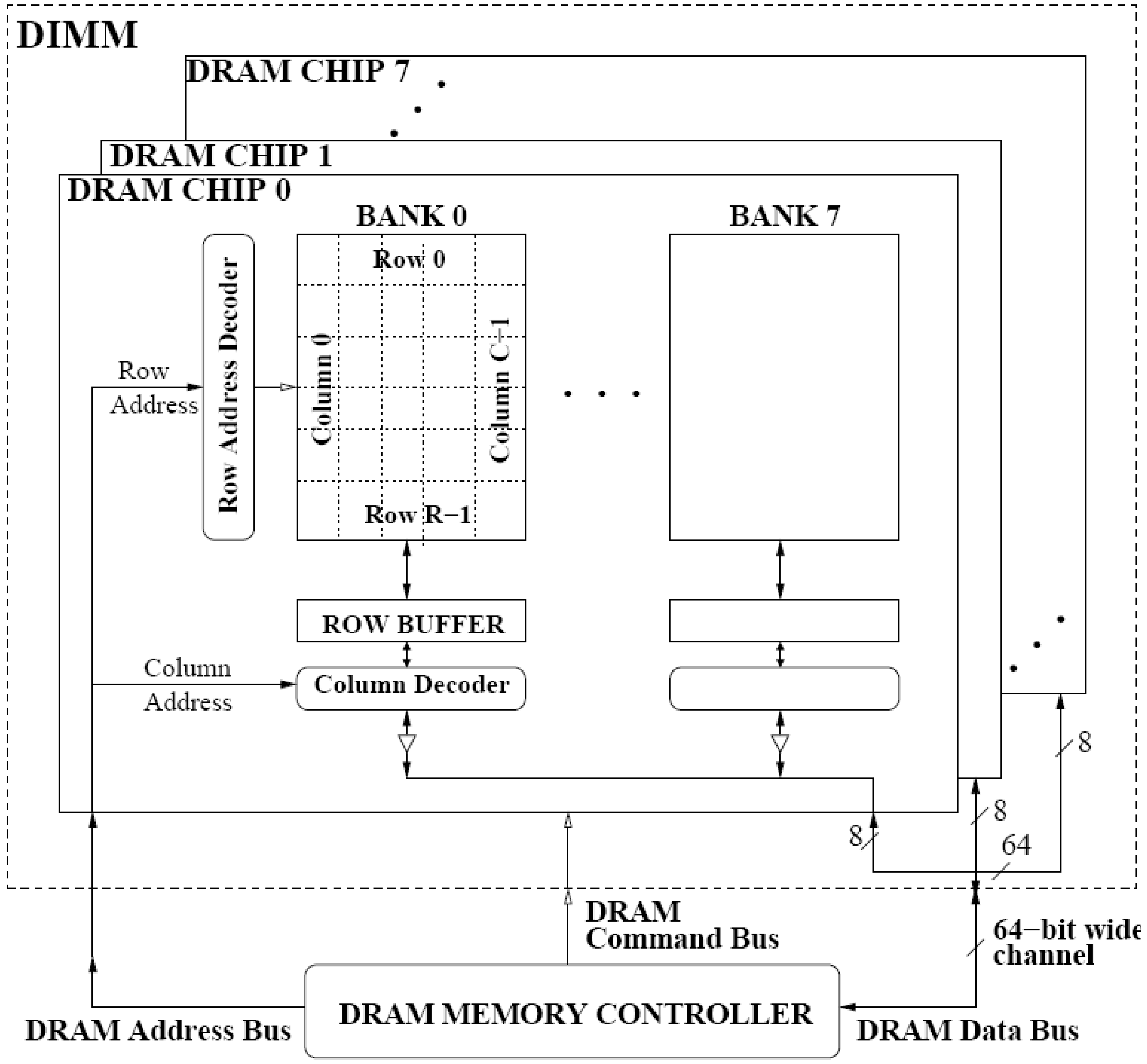
- A DRAM module consists of one or more ranks
  - E.g., DIMM (dual inline memory module)
  - This is what you plug into your motherboard

- 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)

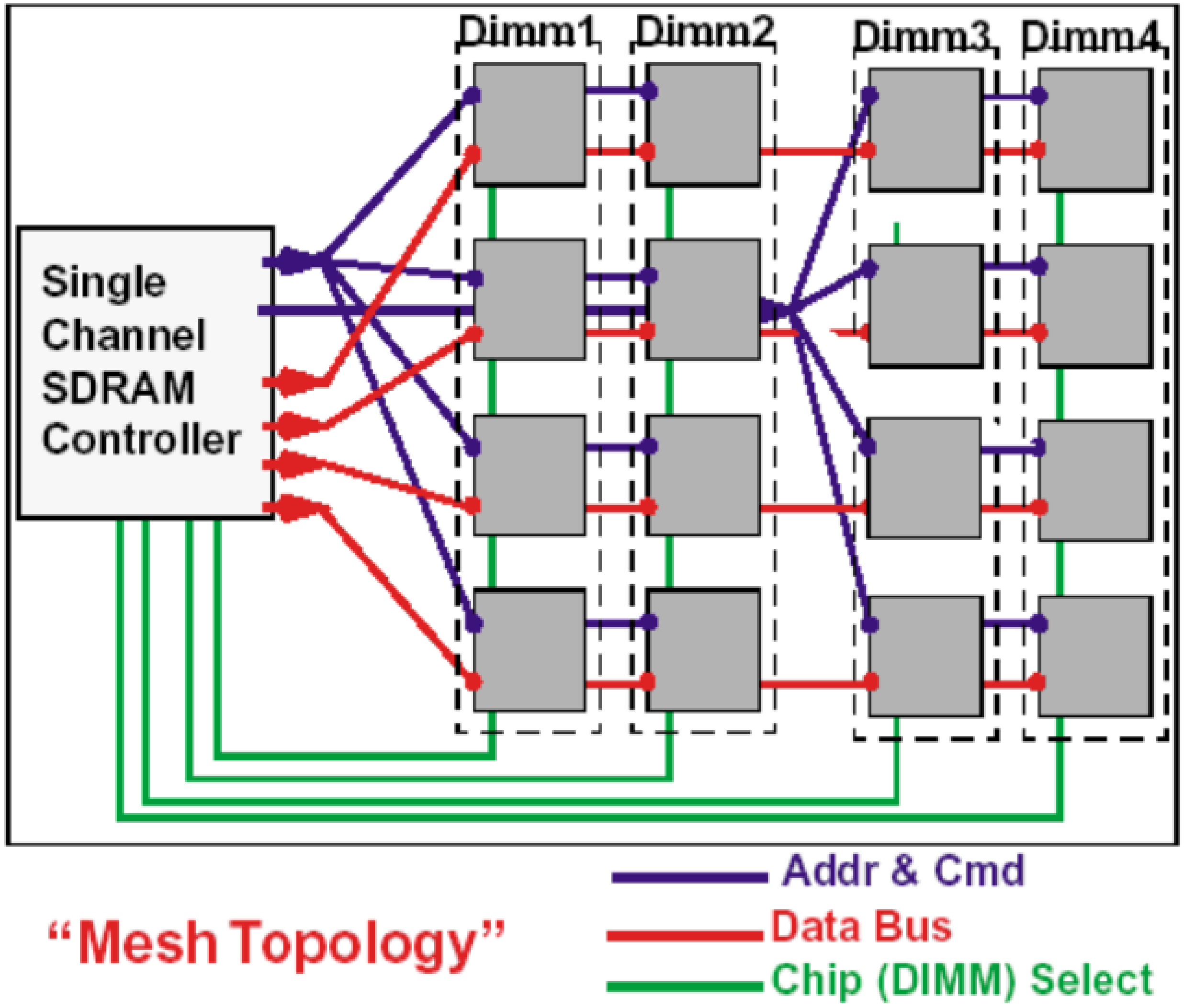


A 64-bit Wide DIMM (One Rank)



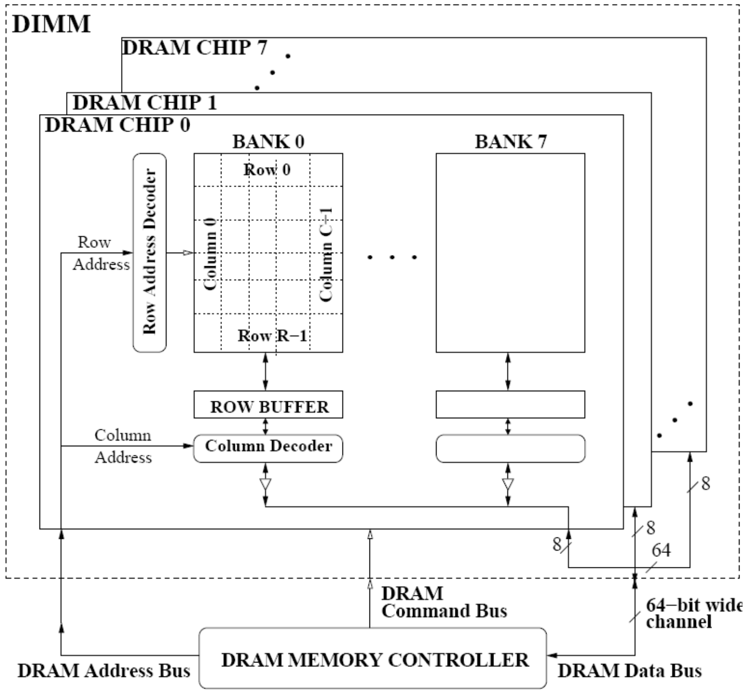
- Advantages:
  - Acts like a high-capacity DRAM chip with a wide interface
  - Flexibility: memory controller does not need to deal with individual chips
- Disadvantages:
  - Granularity: Accesses cannot be smaller than the interface width

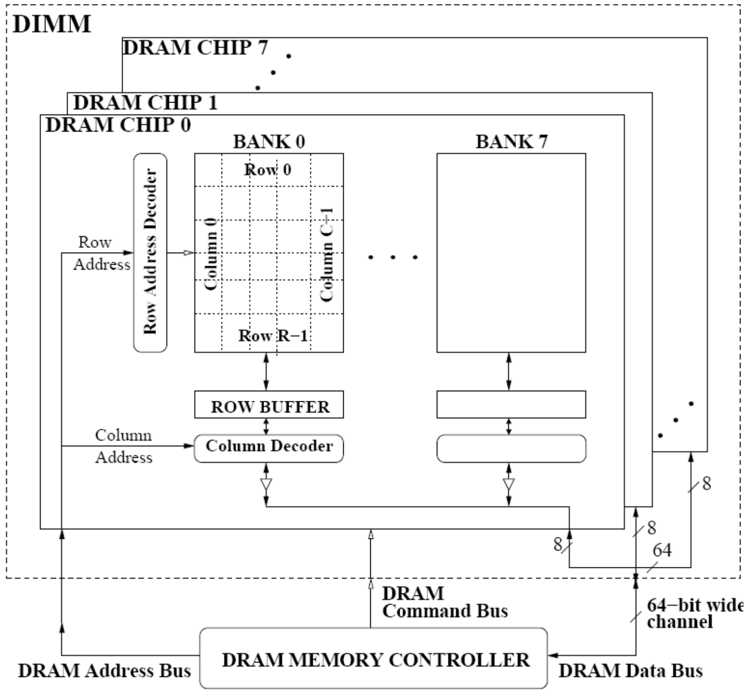
Multiple DIMMs



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

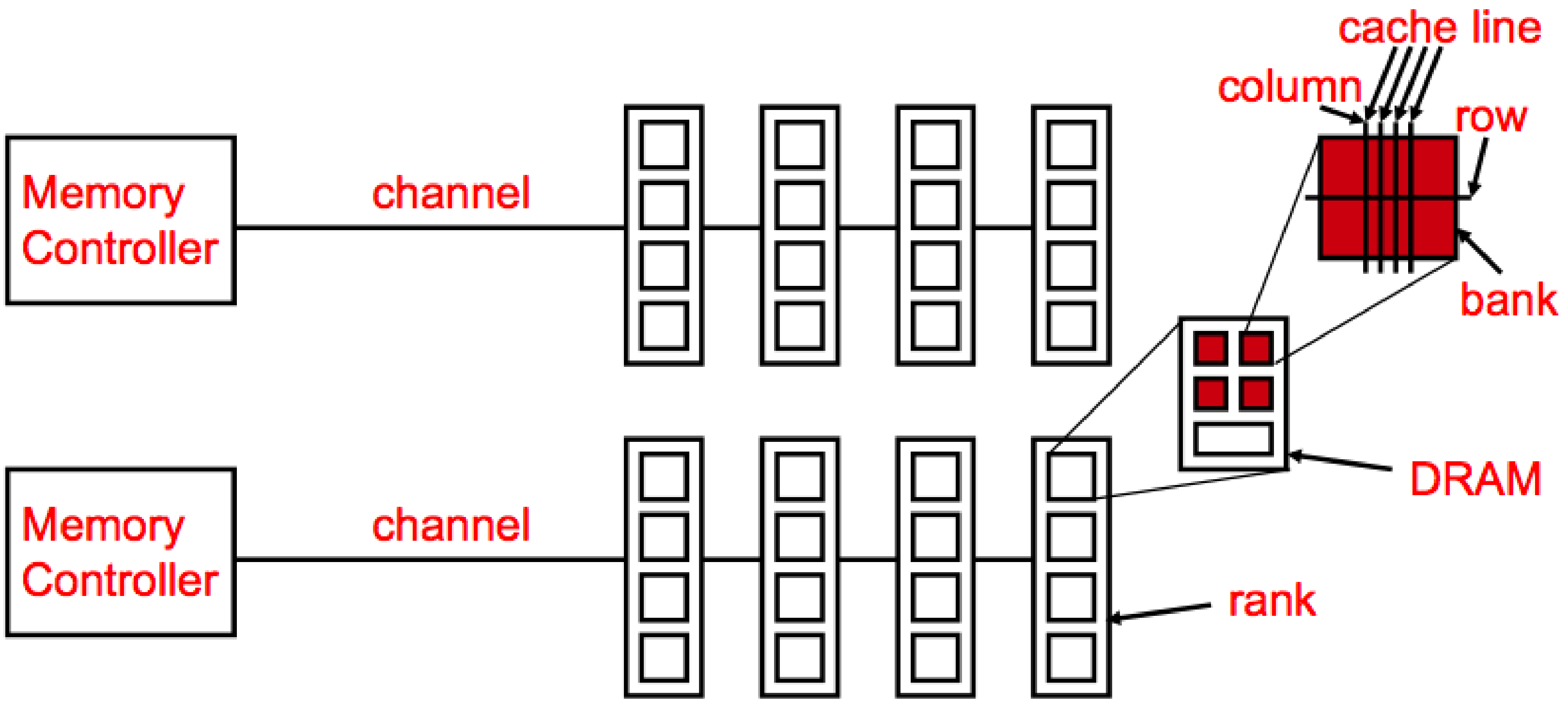
DRAM Channels



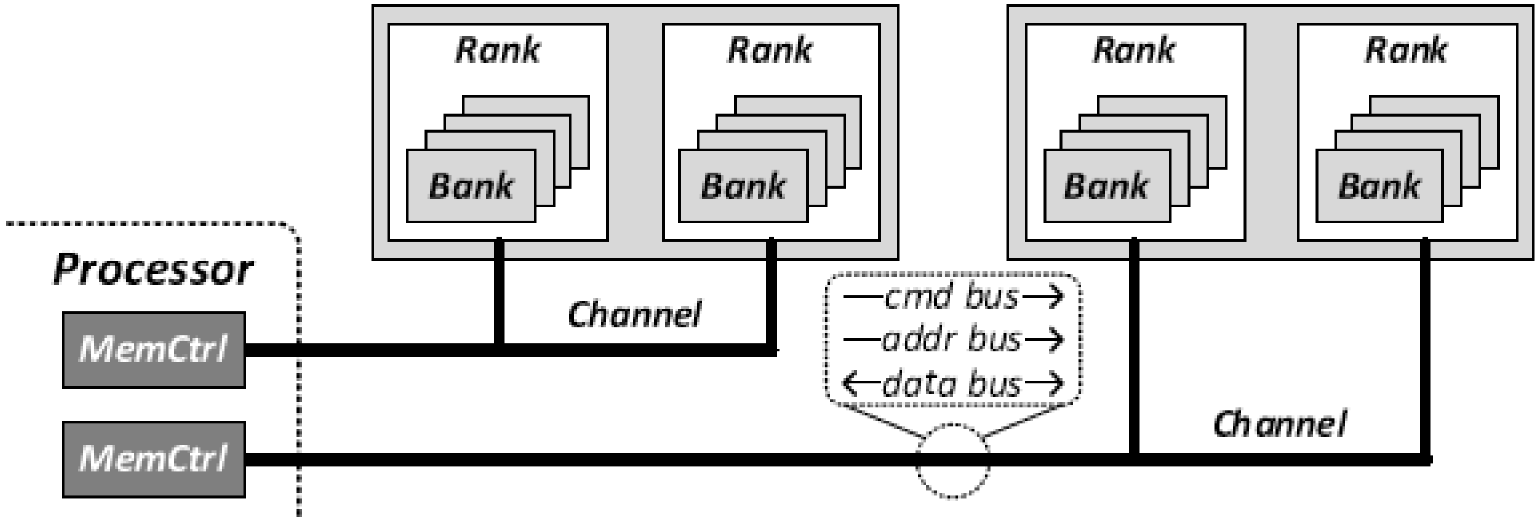


- 2 Independent Channels: 2 Memory Controllers (Above)
- 2 Dependent/Lockstep Channels: 1 Memory Controller with wide interface (Not Shown above)

Generalized Memory Structure



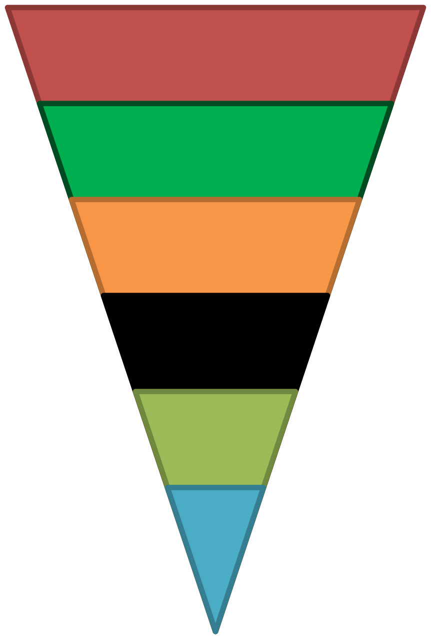
Generalized Memory Structure



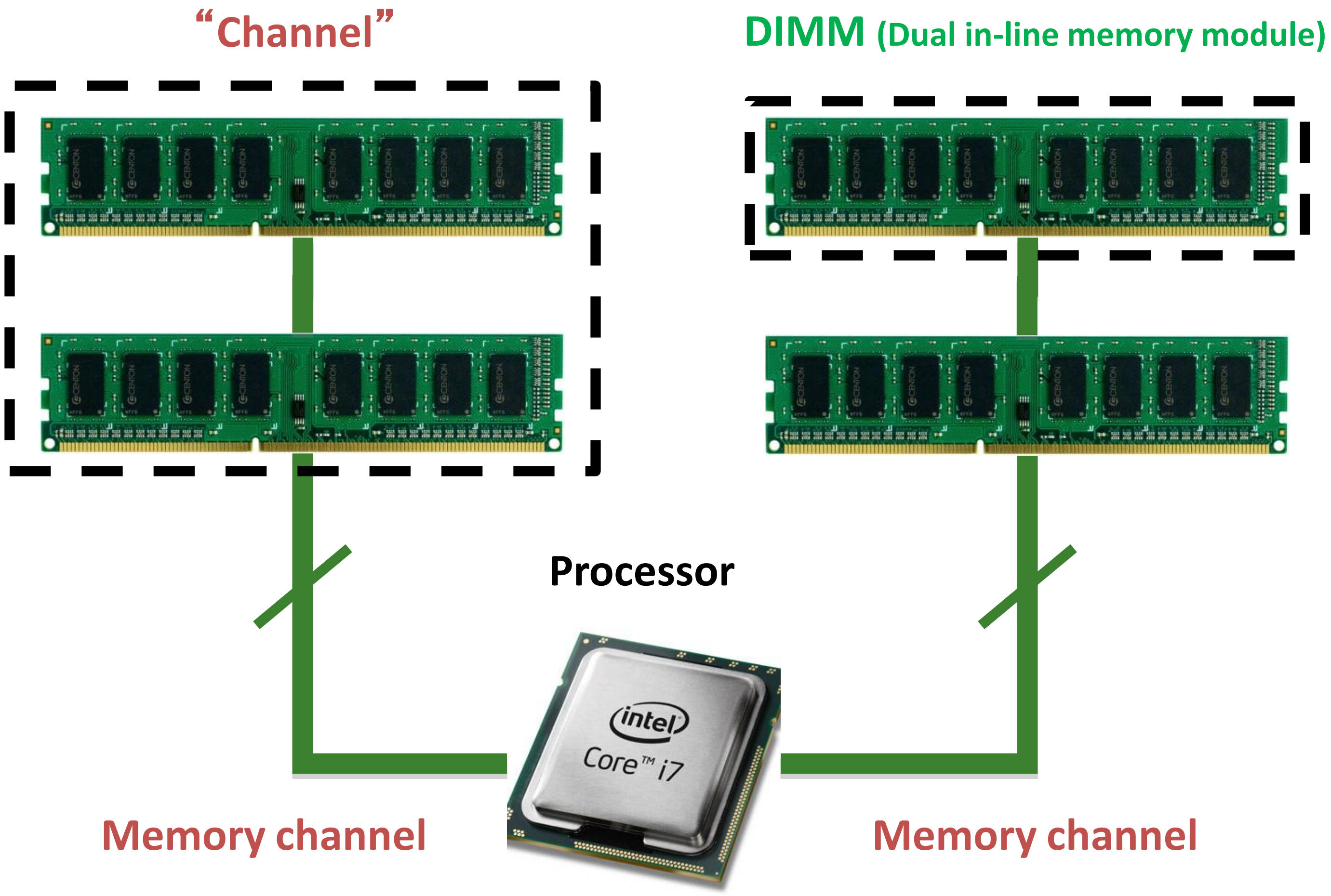
The DRAM Subsystem
The Top Down View

DRAM Subsystem Organization

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



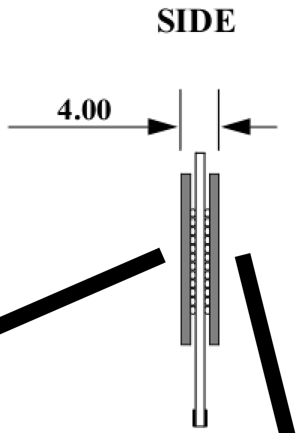
The DRAM subsystem



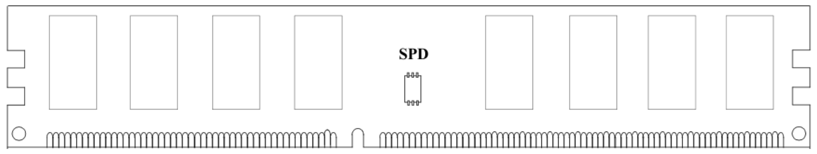
Breaking down a DIMM

DIMM (Dual in-line memory module)

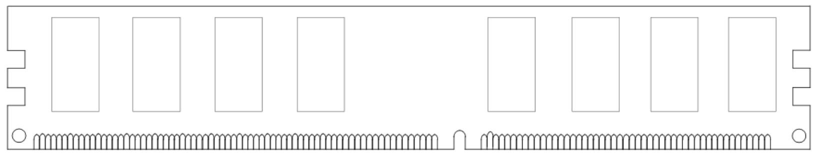
Side view



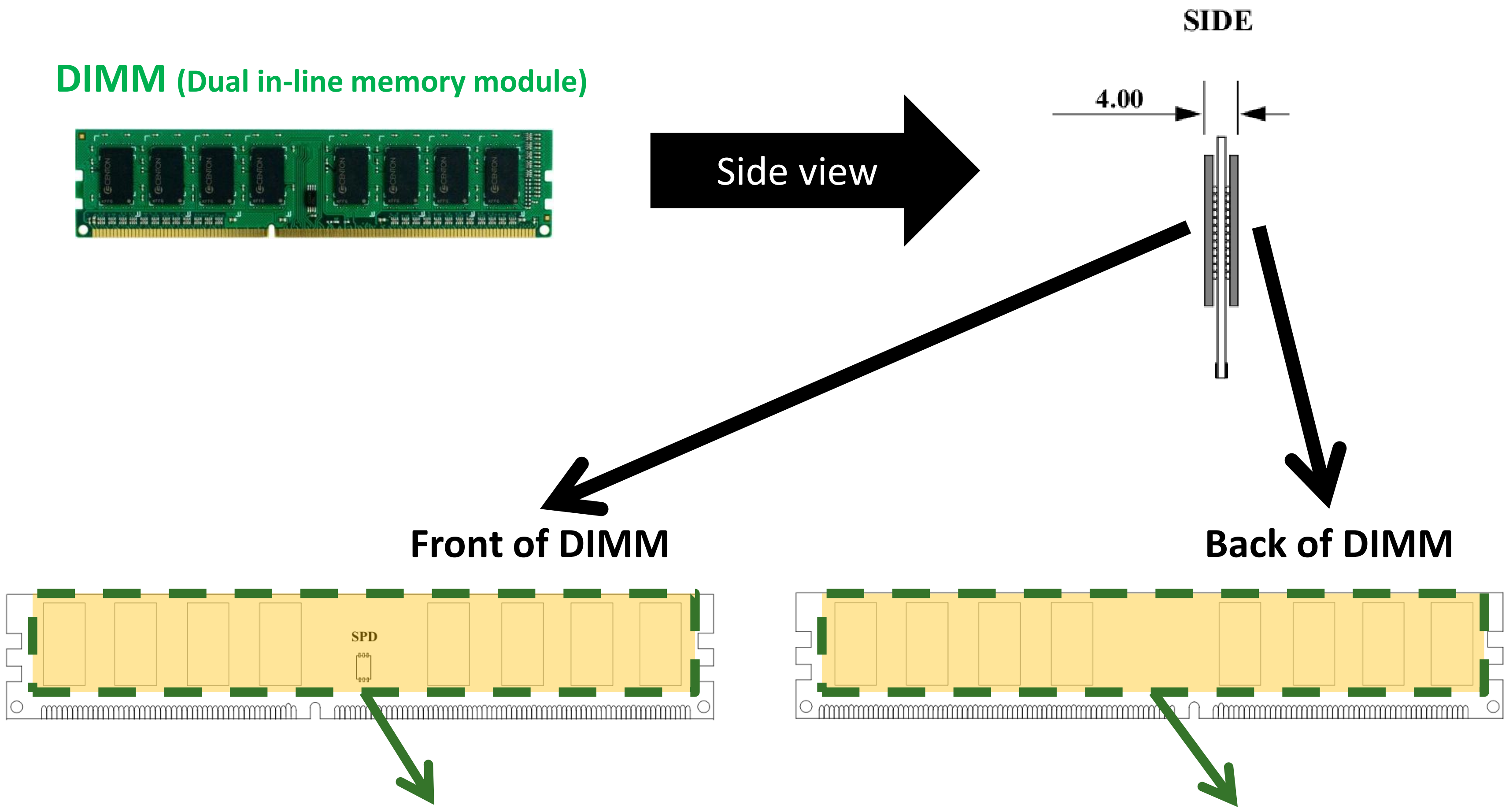
Front of DIMM



Back of DIMM



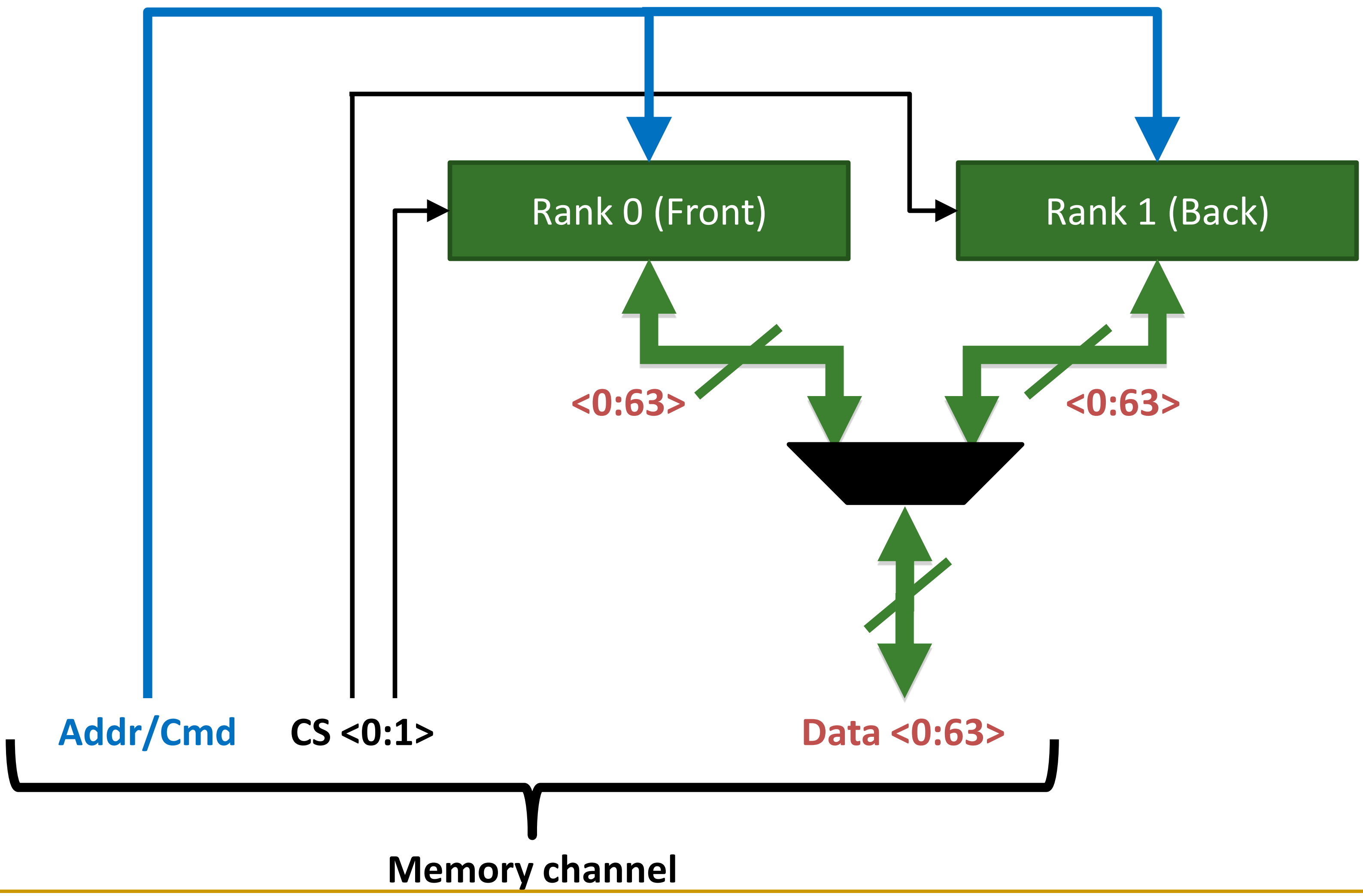
Breaking down a DIMM



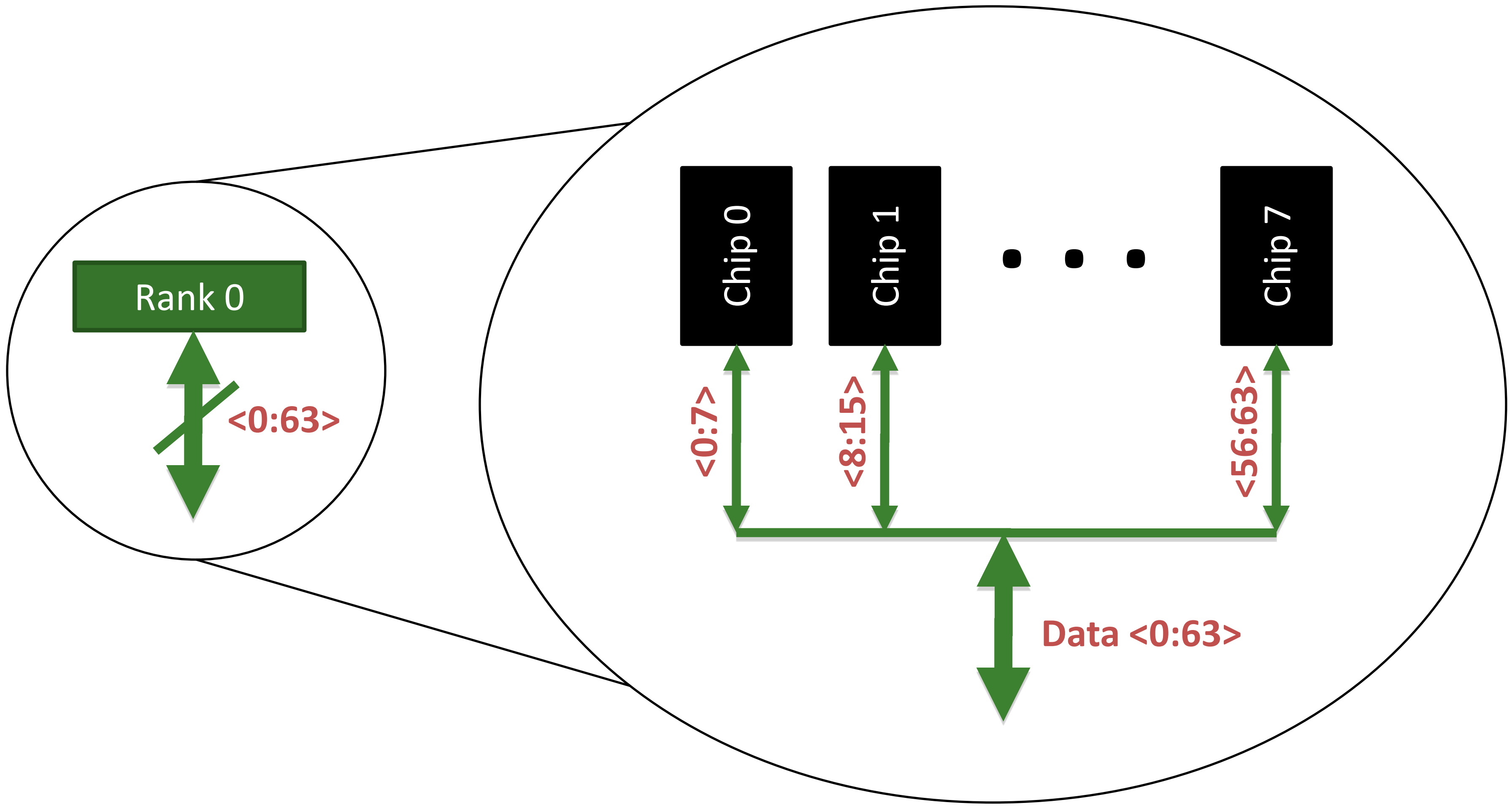
Rank 0: collection of 8 chips

Rank 1

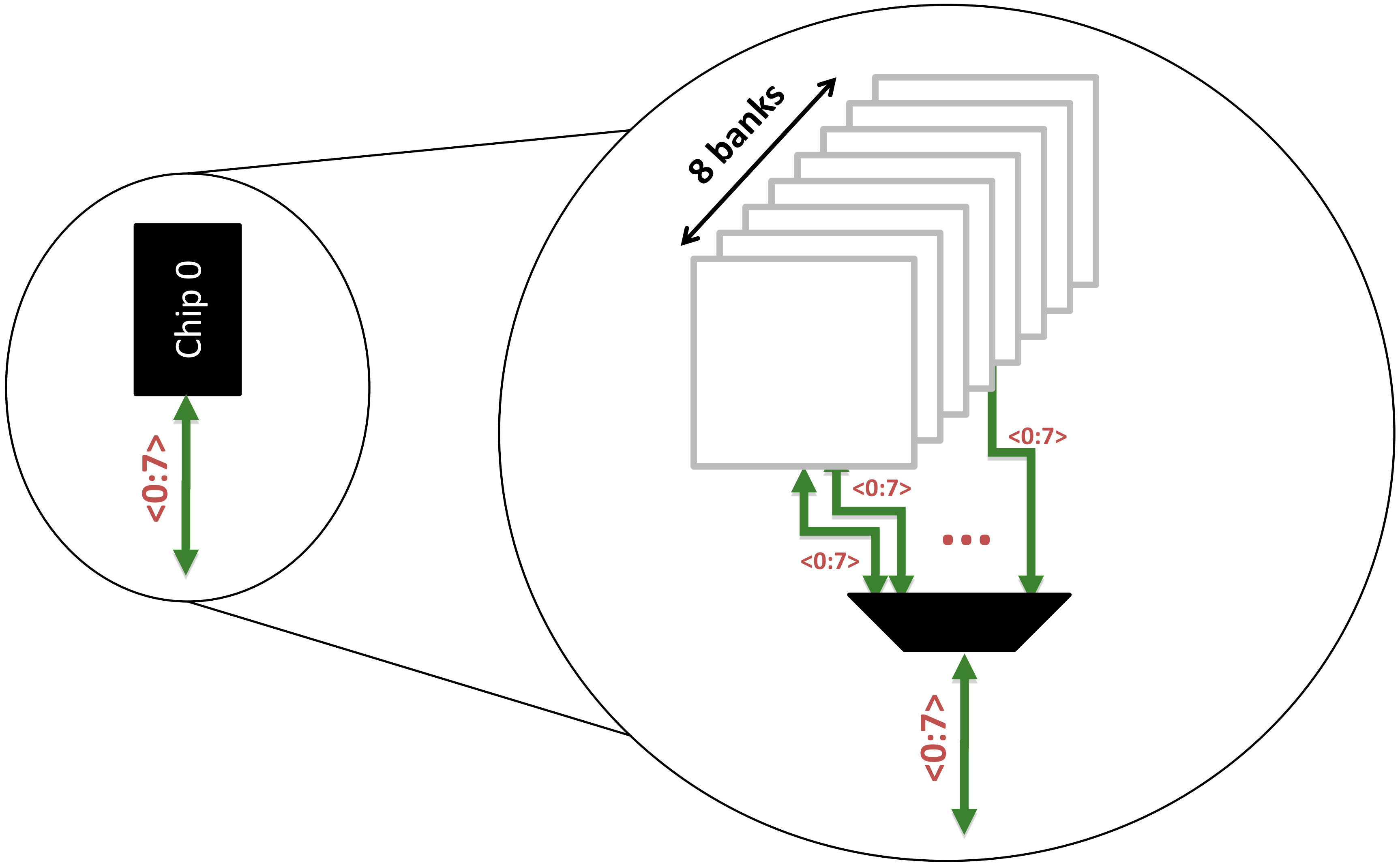
Rank


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

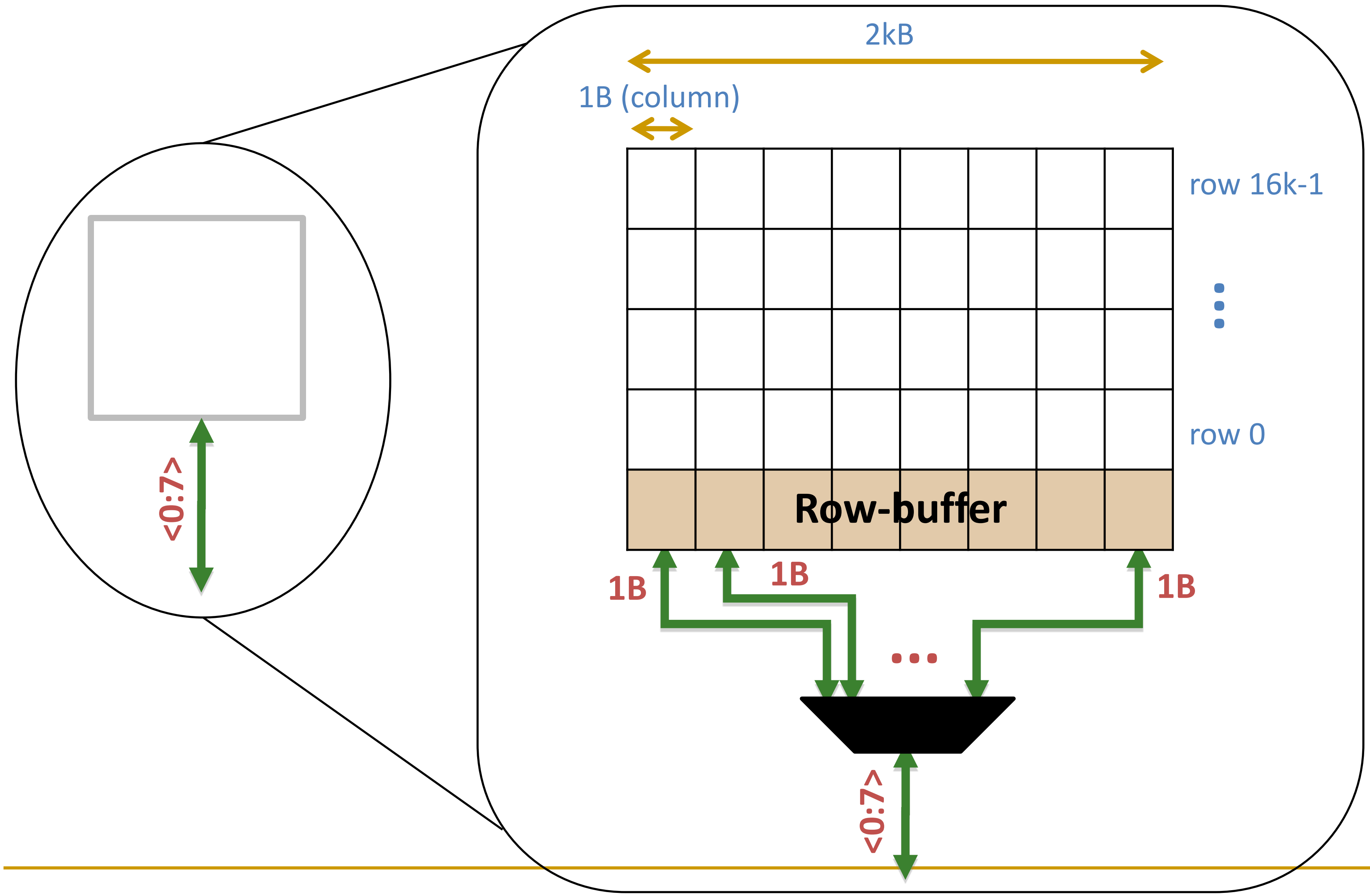
Breaking down a Rank



Breaking down a Chip

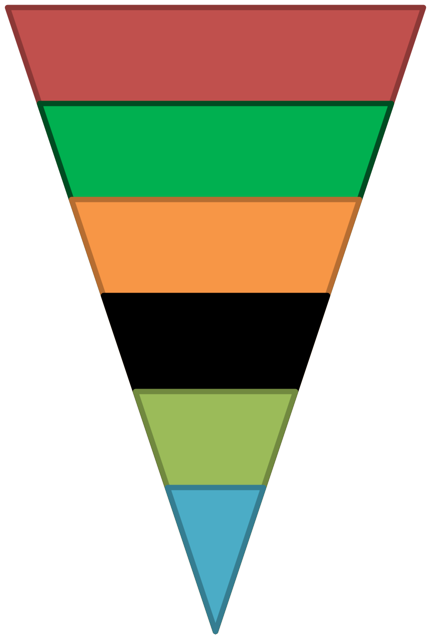


Breaking down a Bank

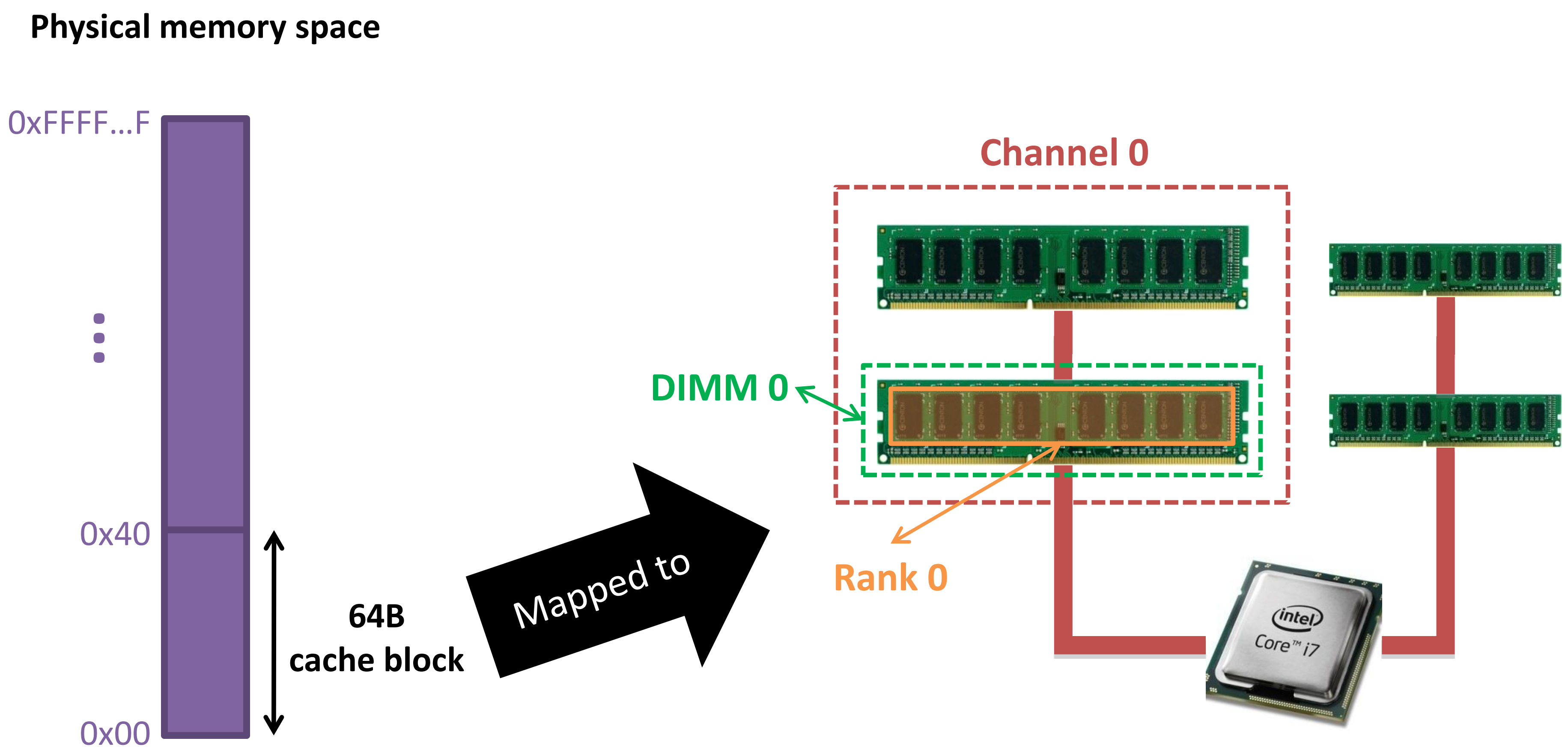


DRAM Subsystem Organization

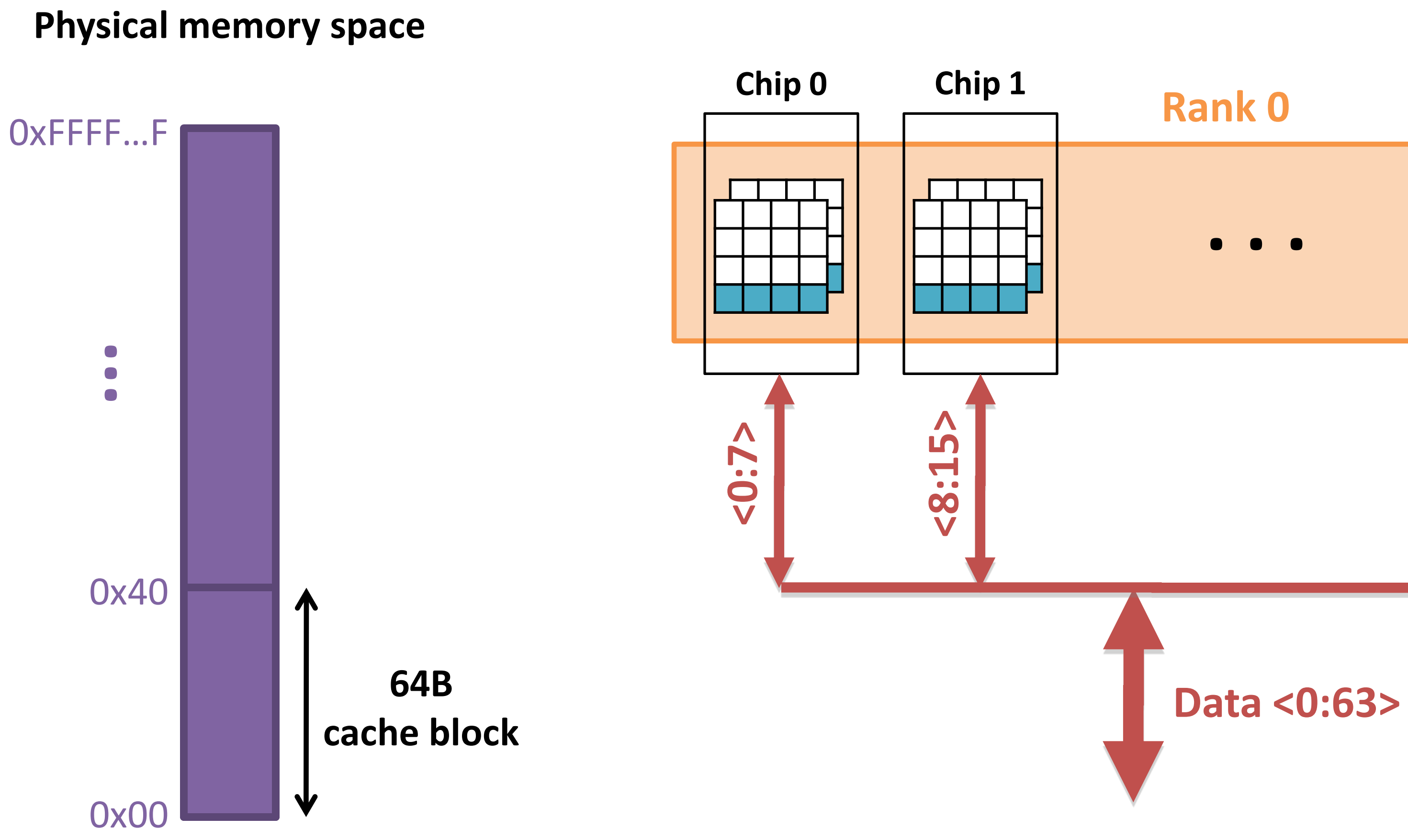
- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



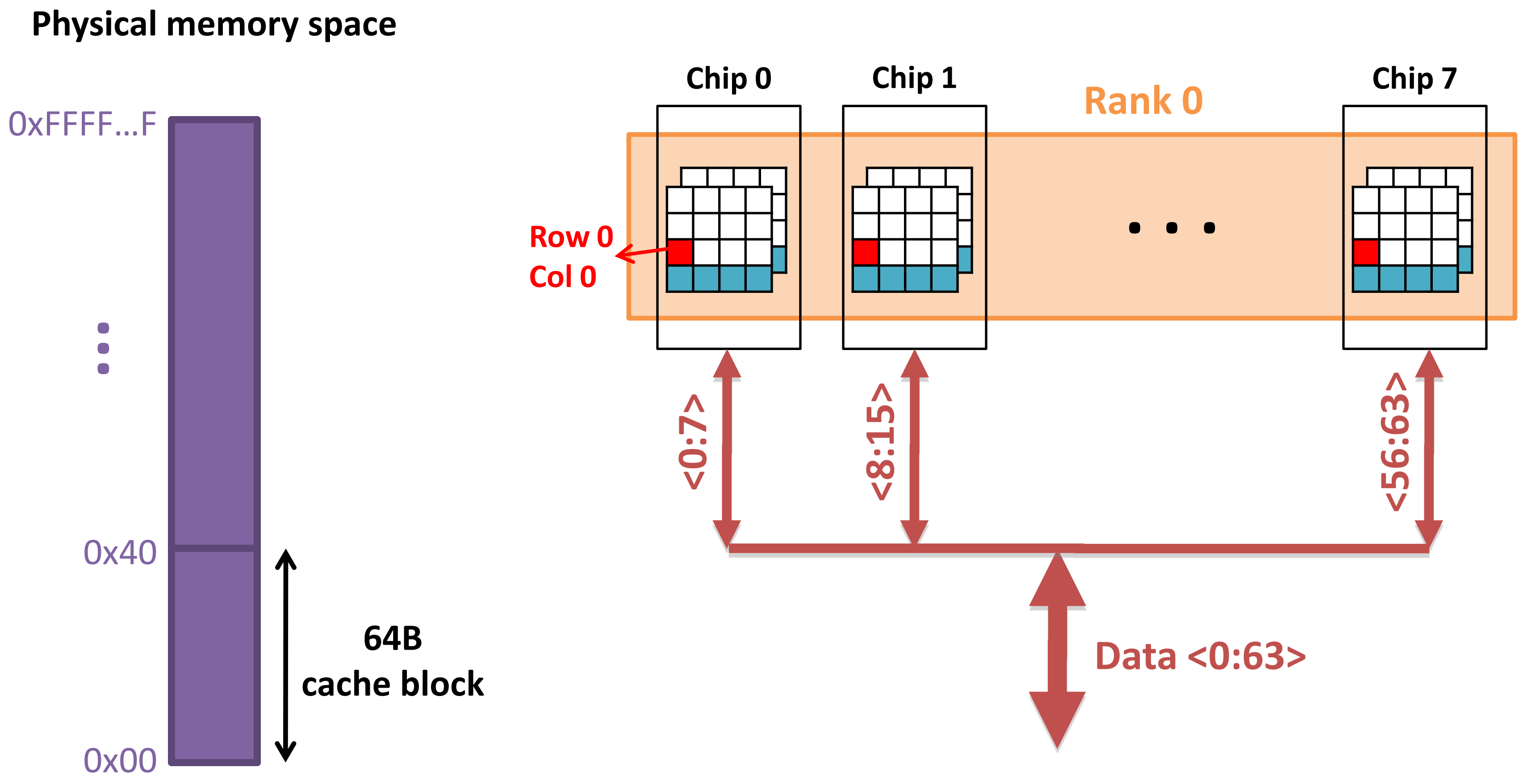
Example: Transferring a cache block



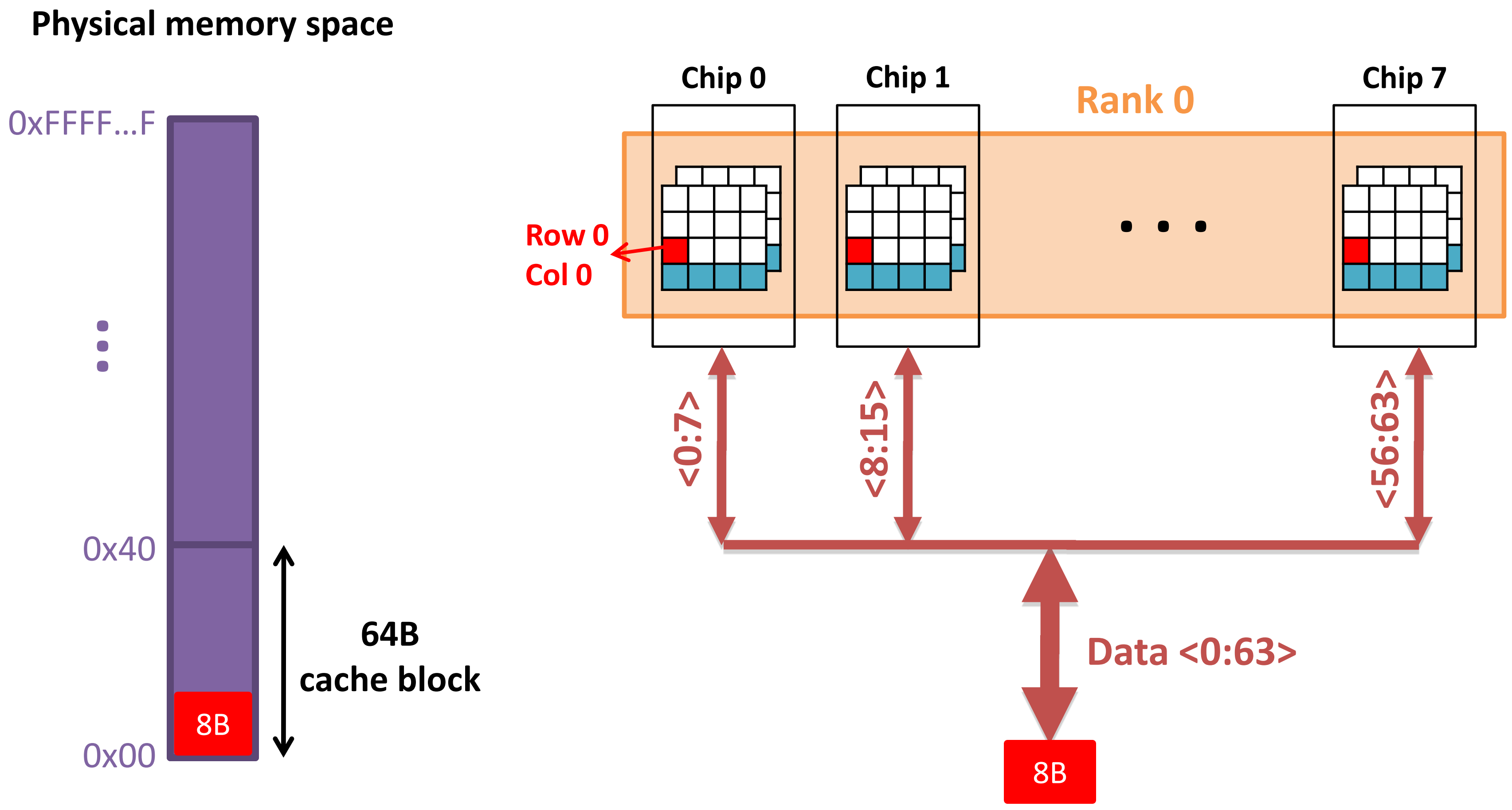
Example: Transferring a cache block



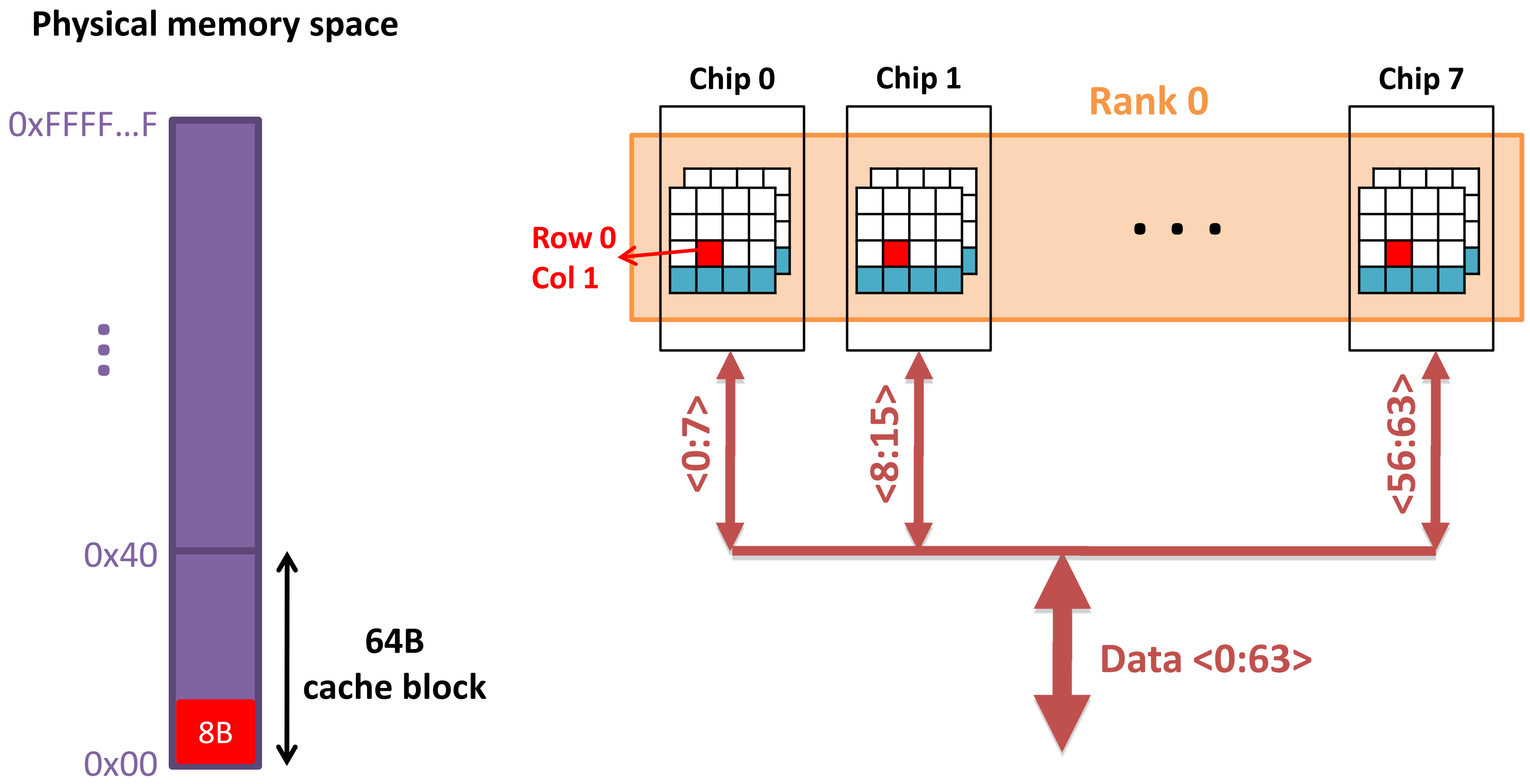
Example: Transferring a cache block



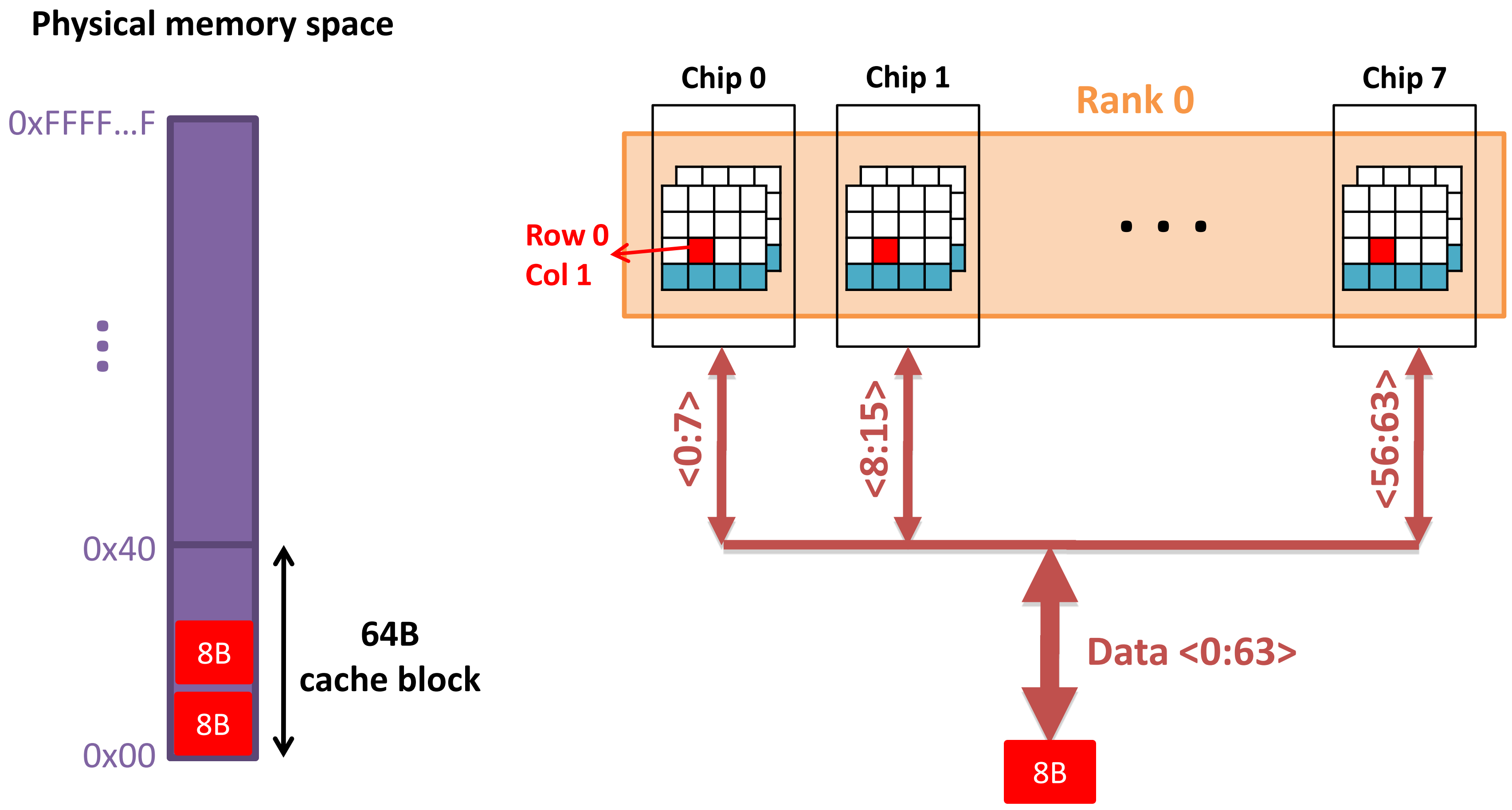
Example: Transferring a cache block



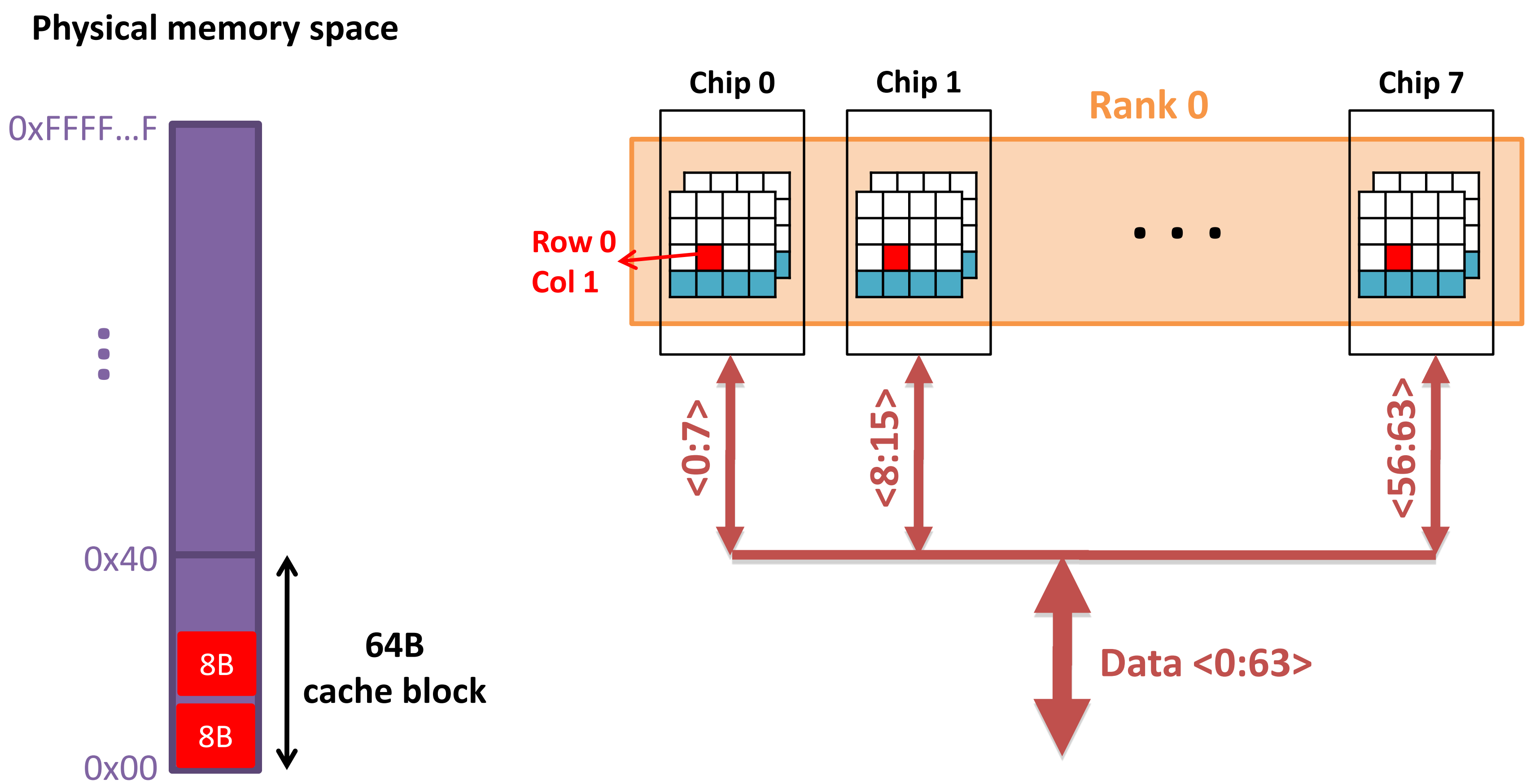
Example: Transferring a cache block



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

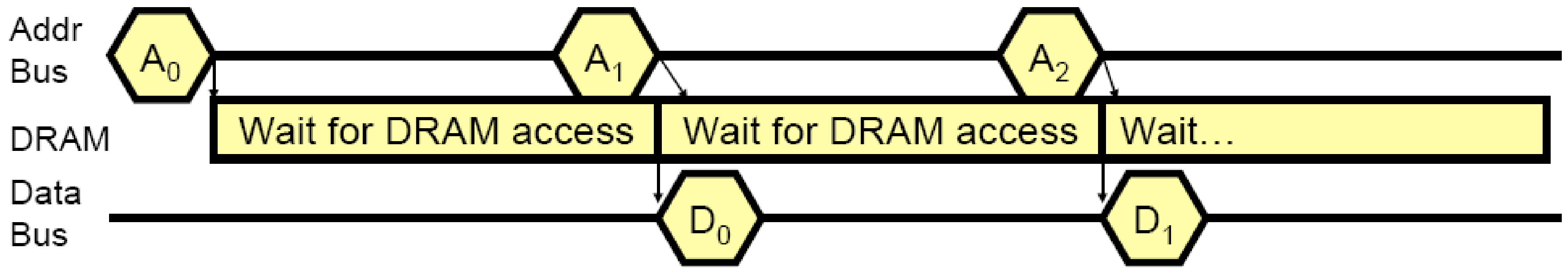
- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS if row is "open" OR
  - RAS + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)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

Multiple Banks (Interleaving) and Channels

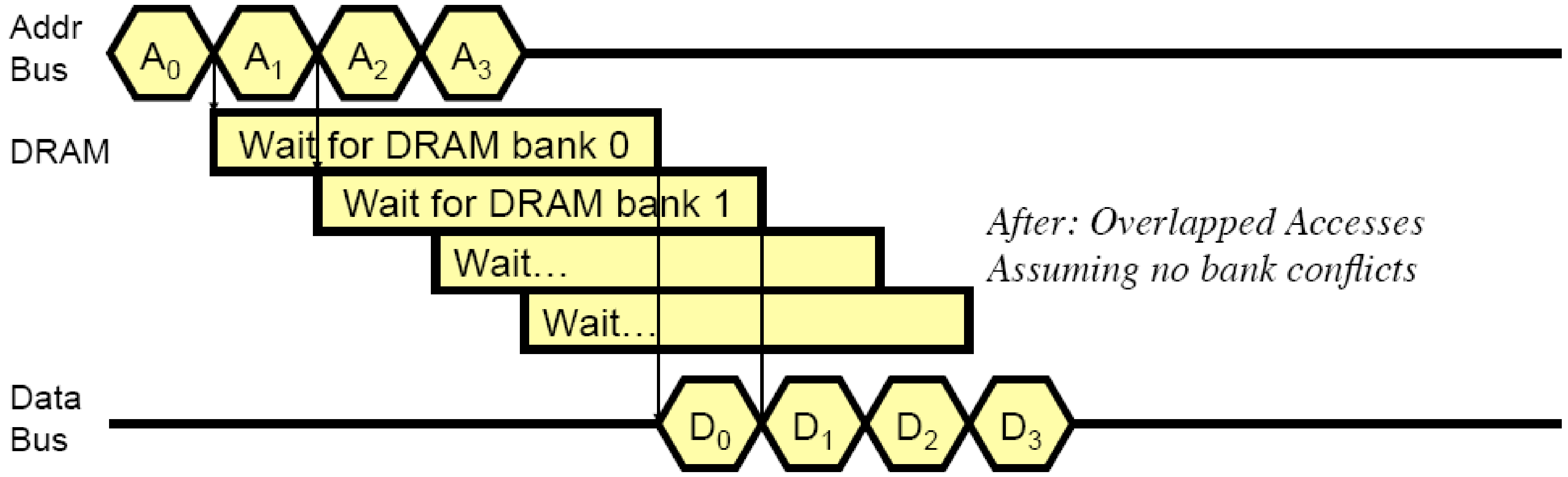
- Multiple banks
  - Enable concurrent DRAM accesses
  - Bits in address determine which bank an address resides in
- Multiple independent channels serve the same purpose
  - But they are even better because they have separate data buses
  - Increased bus bandwidth

- 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



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)

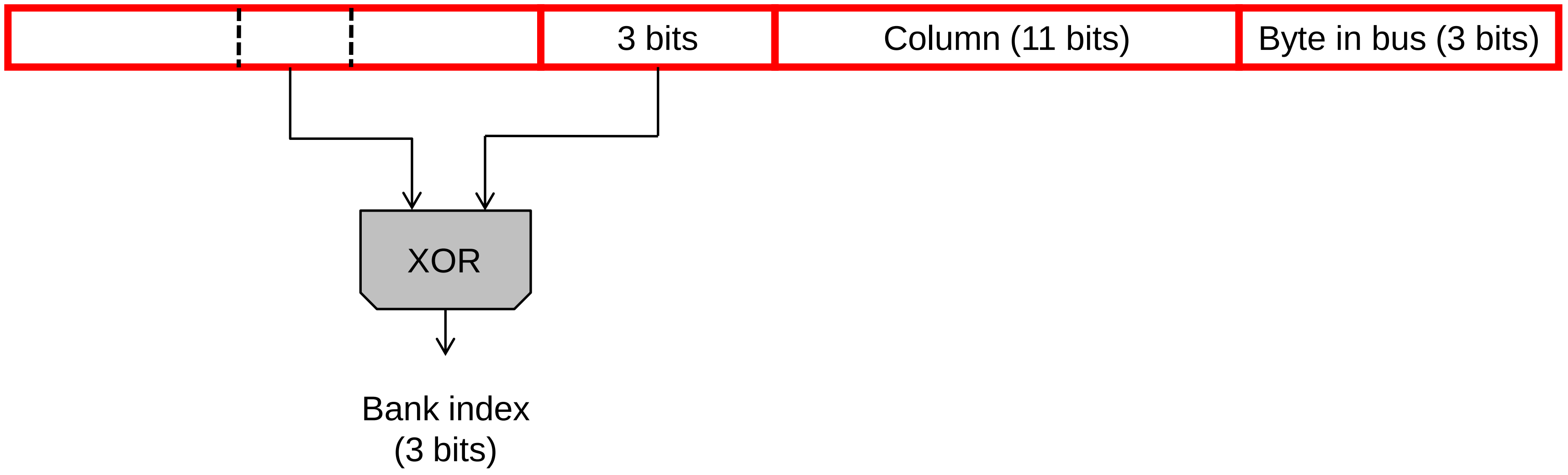
- Cache block interleaving
  - Consecutive cache block addresses in consecutive banks
  - 64 byte cache blocks

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
  - 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)

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	

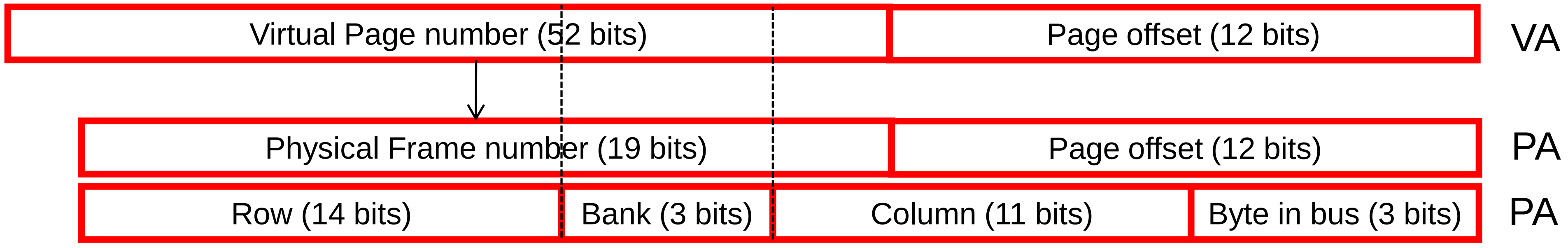
Row (14 bits)	High Column	C	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
	8 bits			3 bits	

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 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

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 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

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

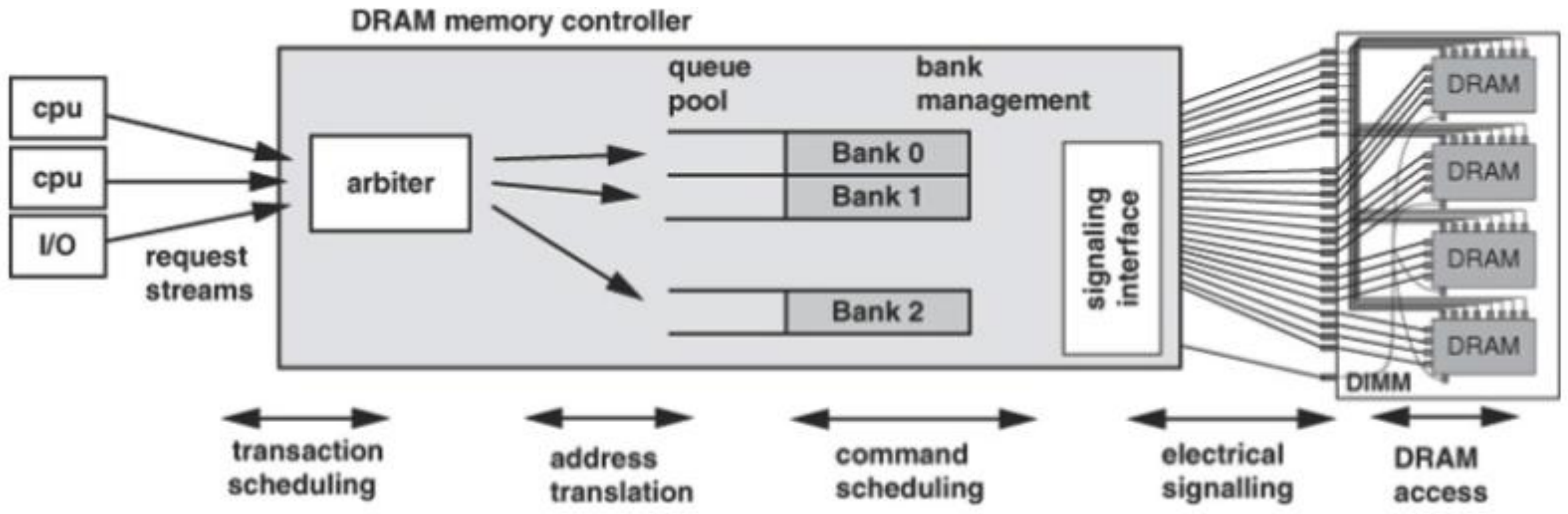
- Manage power consumption and thermals in DRAM
  - Turn on/off DRAM chips, manage power modes

DRAM Controller: Where to Place

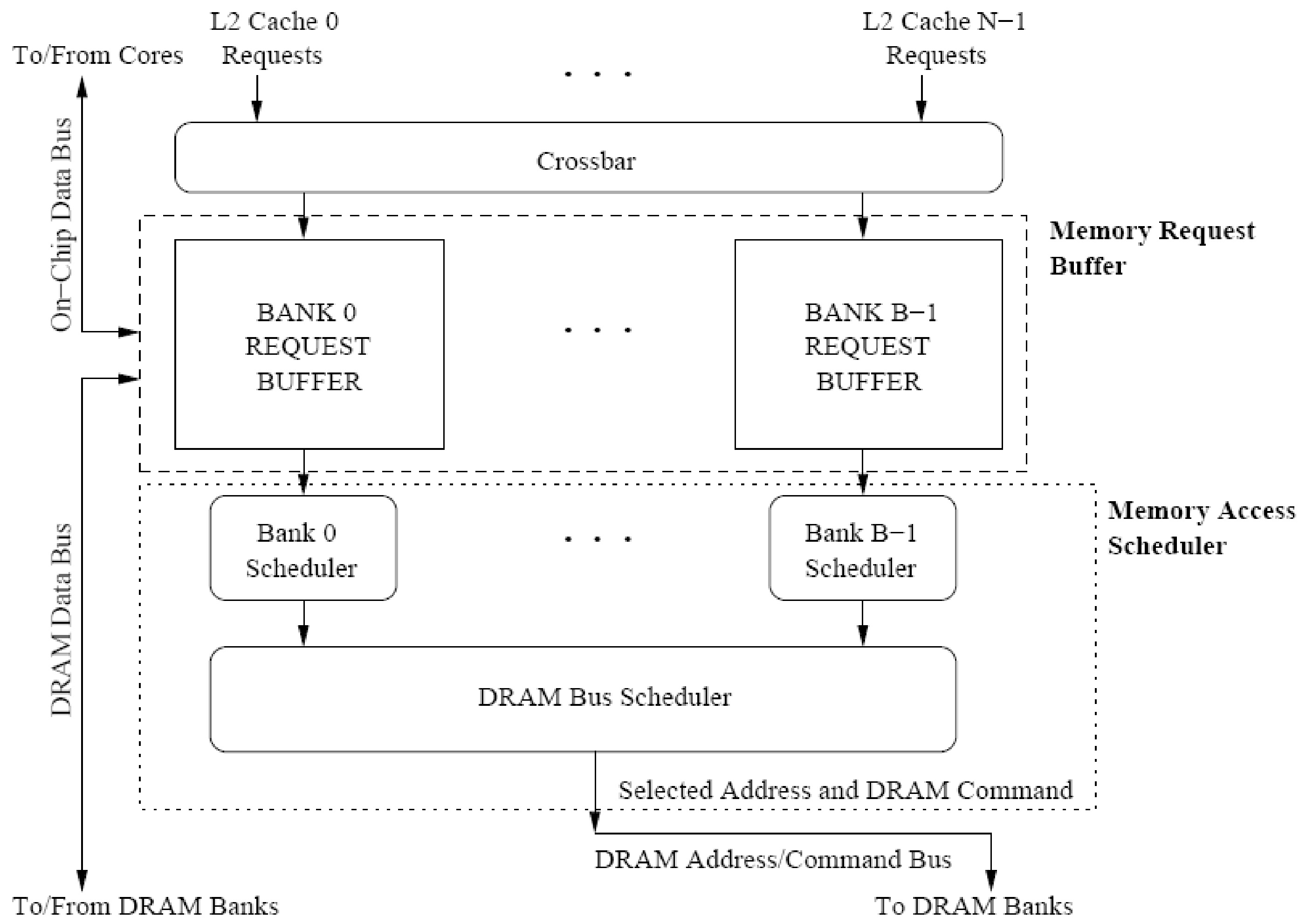
- In chipset
  + More flexibility to plug different DRAM types into the system
  + Less power density in the CPU chip

- 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)

DRAM Controller (II)



A Modern DRAM Controller



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?

Row Buffer Management Policies

- Open row
  - Keep the row open after an access

  + Next access might need the same row → row hit

  -- Next access might need a different row → row conflict, wasted energy

- Closed row
  - Close the row after an access (if no other requests already in the request buffer need the same row)

  + Next access might need a different row → avoid a row conflict

  -- Next access might need the same row → extra activate latency

- Adaptive policies
  - Predict whether or not the next access to the bank will be to the same row

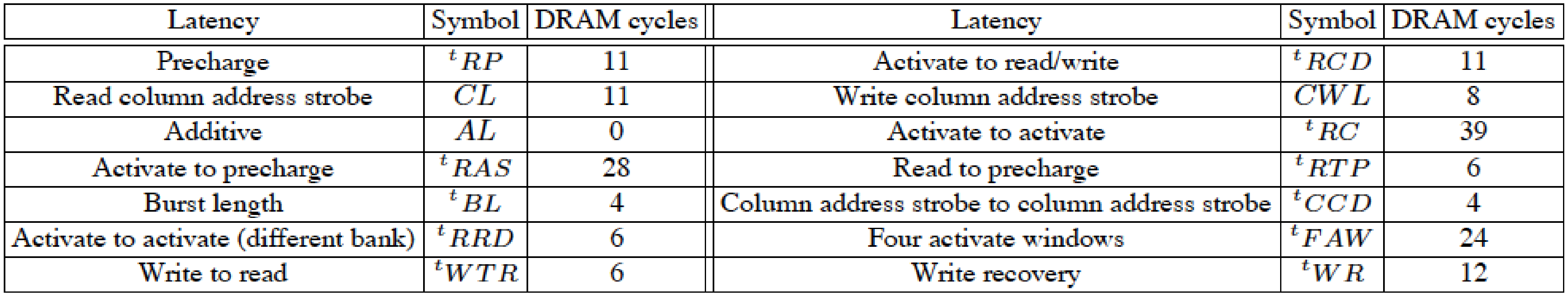
Open vs. Closed Row Policies

Policy	First access	Next access	Commands needed for next access
Open row	Row 0	Row 0 (row hit)	Read
Open row	Row 0	Row 1 (row conflict)	Precharge + Activate Row 1 + Read
Closed row	Row 0	Row 0 – access in request buffer (row hit)	Read
Closed row	Row 0	Row 0 – access not in request buffer (row closed)	Activate Row 0 + Read + Precharge
Closed row	Row 0	Row 1 (row closed)	Activate Row 1 + Read + Precharge

Why are DRAM Controllers Difficult to Design?

- Need to obey DRAM timing constraints for correctness
  - There are many (50+) timing constraints in DRAM
  - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued
  - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
  - …
- Need to keep track of many resources to prevent conflicts
  - Channels, banks, ranks, data bus, address bus, row buffers
- Need to handle DRAM refresh
- Need to optimize for performance (in the presence of constraints)
  - Reordering is not simple
  - Predicting the future?

Many DRAM Timing Constraints



Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Precharge	^tRP	11	Activate to read/write	^tRCD	11
Read column address strobe	CL	11	Write column address strobe	CWL	8
Additive	AL	0	Activate to activate	^tRC	39
Activate to precharge	^tRAS	28	Read to precharge	^tRTP	6
Burst length	^tBL	4	Column address strobe to column address strobe	^tCCD	4
Activate to activate (different bank)	^tRRD	6	Four activate windows	^tFAW	24
Write to read	^tWTR	6	Write recovery	^tWR	12

Table 4. DDR3 1600 DRAM timing specifications

- From Lee et al., "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," HPS Technical Report, April 2010.

More on DRAM Operation

- Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
- Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

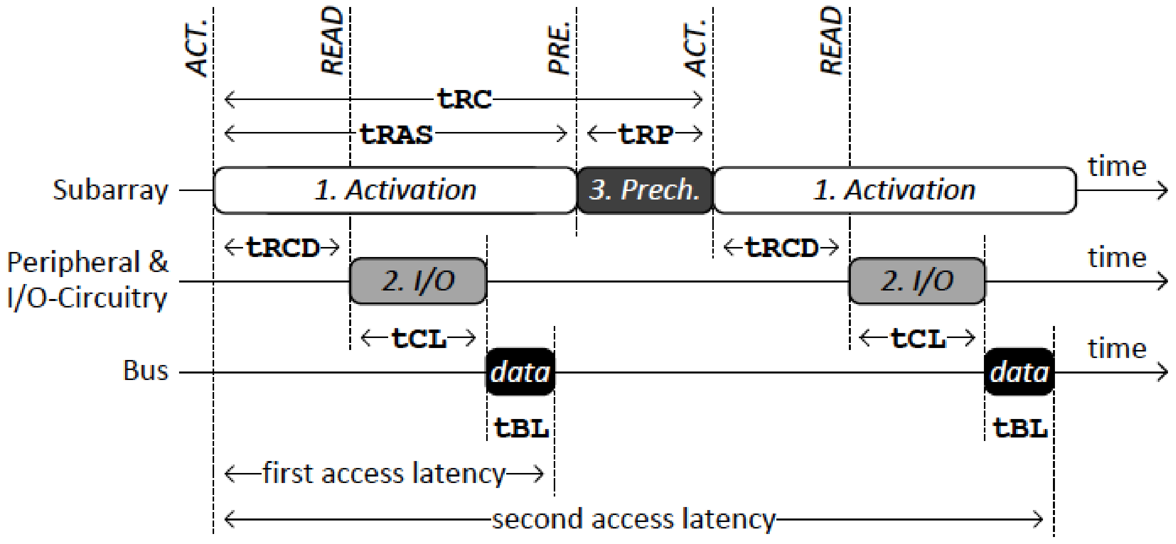


Figure 5. Three Phases of DRAM Access

Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ ACT → WRITE	tRCD	15ns
	ACT → PRE	tRAS	37.5ns
2	READ → *data* WRITE → *data*	tCL tCWL	15ns 11.25ns
	data burst	tBL	7.5ns
3	PRE → ACT	tRP	15ns
1 & 3	ACT → ACT	tRC (tRAS+tRP)	52.5ns

DRAM Power Management

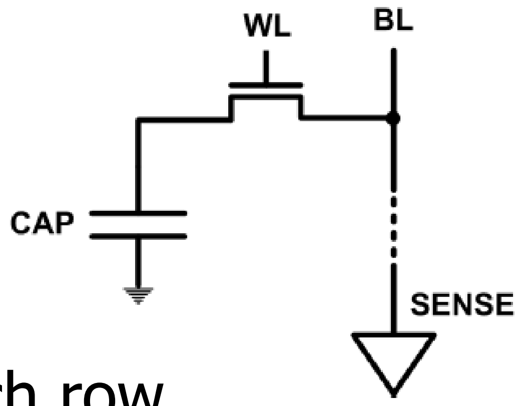
- DRAM chips have power modes
- Idea: When not accessing a chip power it down

- Power states
  - Active (highest power)
  - All banks idle
  - Power-down
  - Self-refresh (lowest power)

- State transitions incur latency during which the chip cannot be accessed

DRAM Refresh

DRAM Refresh



- DRAM capacitor charge leaks over time
- The memory controller needs to refresh each row periodically to restore charge
  - Read and close each row every N ms
  - Typical N = 64 ms
- Downsides of refresh

  -- Energy consumption: Each refresh consumes energy

  -- Performance degradation: DRAM rank/bank unavailable while refreshed

  -- QoS/predictability impact: (Long) pause times during refresh

  -- Refresh rate limits DRAM capacity scaling

DRAM Refresh: Performance

- Implications of refresh on performance

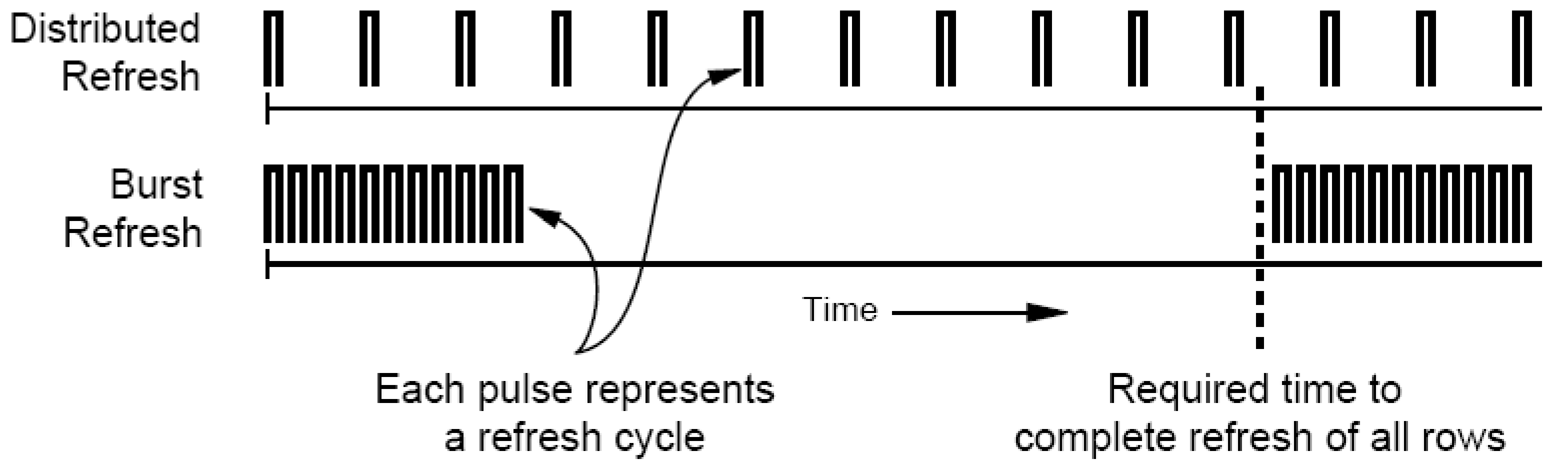
  -- DRAM bank unavailable while refreshed

  -- Long pause times: If we refresh all rows in burst, every 64ms the DRAM will be unavailable until refresh ends

- Burst refresh: All rows refreshed immediately after one another

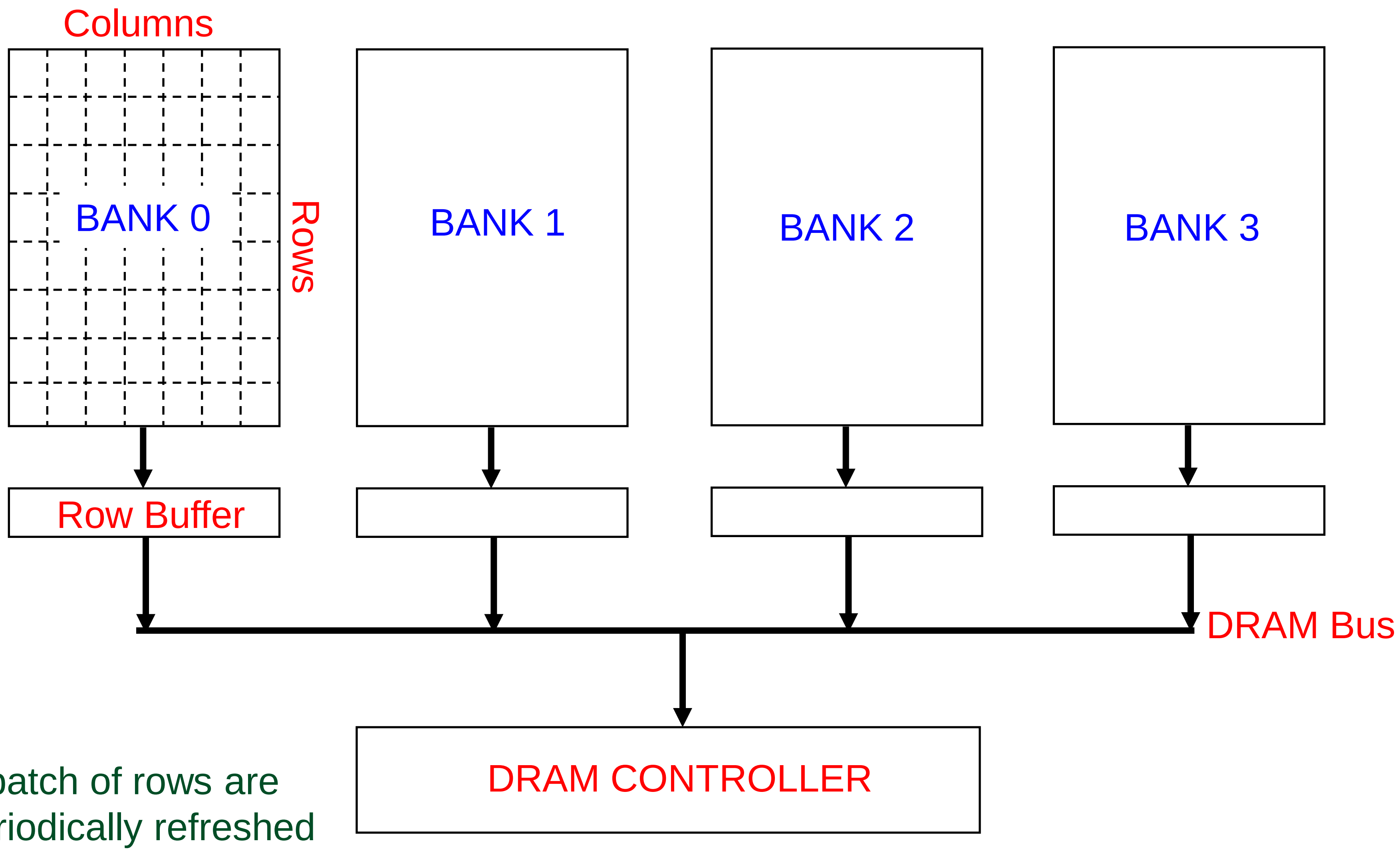
- Distributed refresh: Each row refreshed at a different time, at regular intervals

Distributed Refresh


Distributed Refresh
Burst Refresh
Time
Each pulse represents a refresh cycle
Required time to complete refresh of all rows

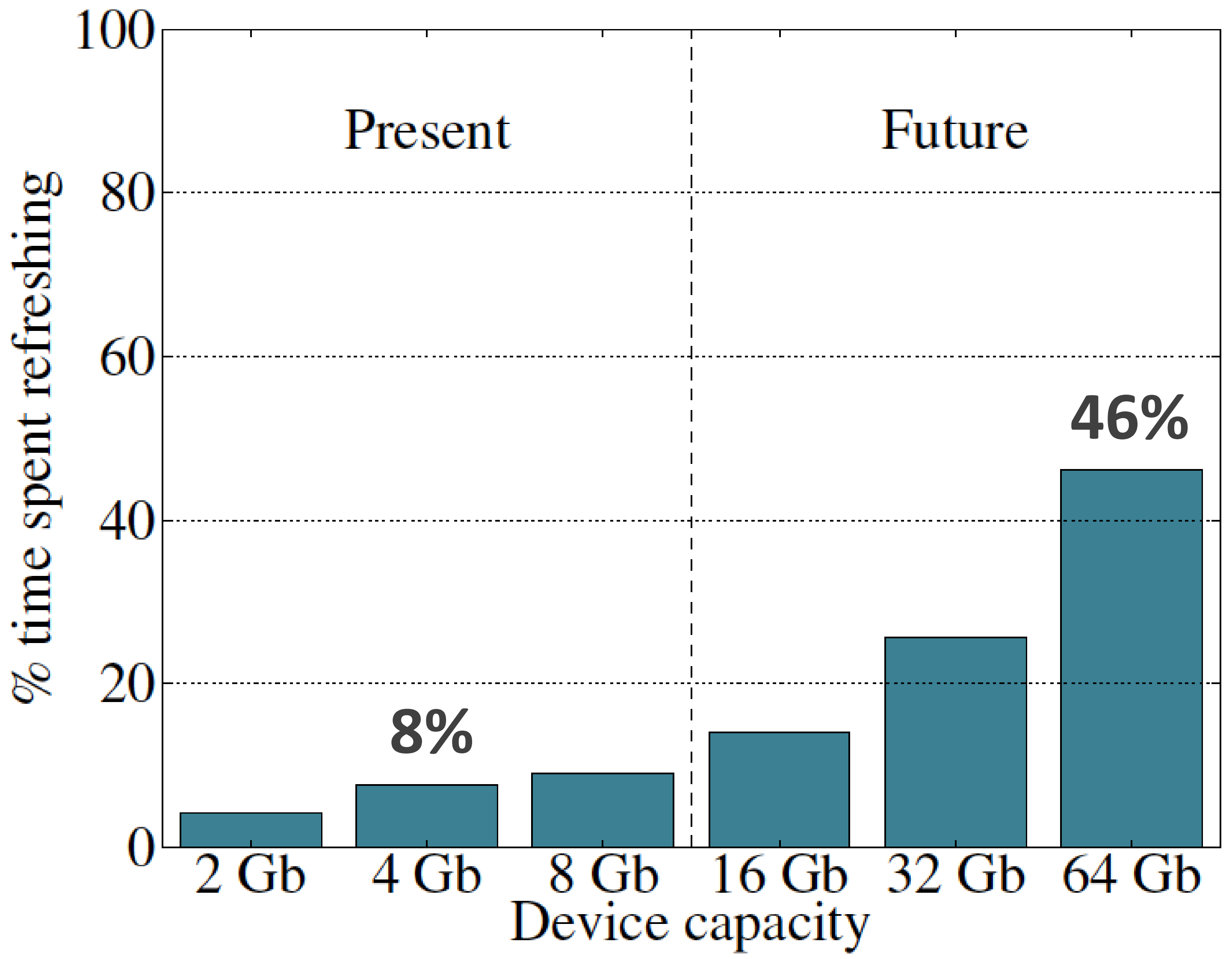
- Distributed refresh eliminates long pause times
- How else can we reduce the effect of refresh on performance/QoS?
- Does distributed refresh reduce refresh impact on energy?
- Can we reduce the number of refreshes?

Refresh Today: Auto Refresh

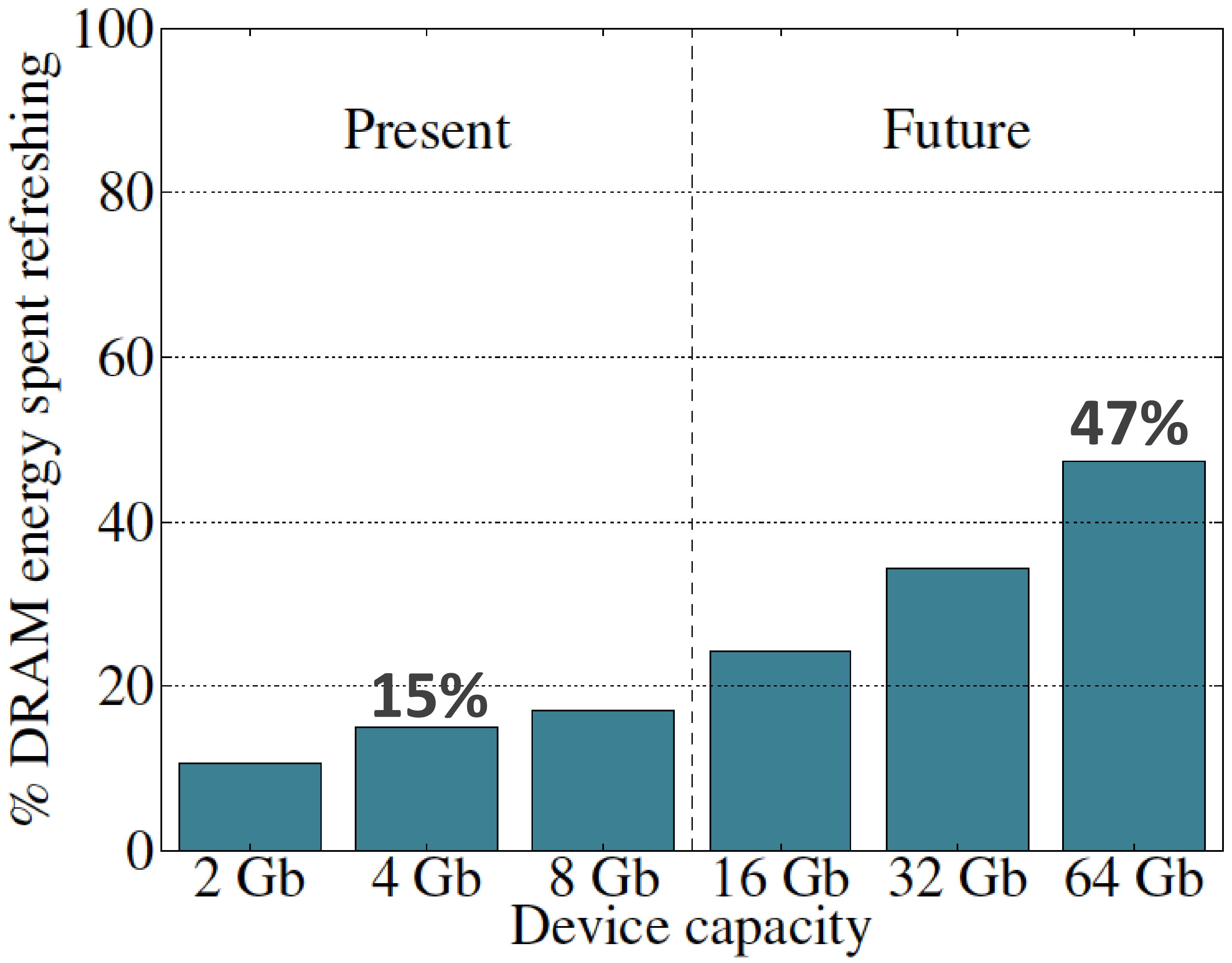


A batch of rows are periodically refreshed via the auto-refresh command

Refresh Overhead: Performance

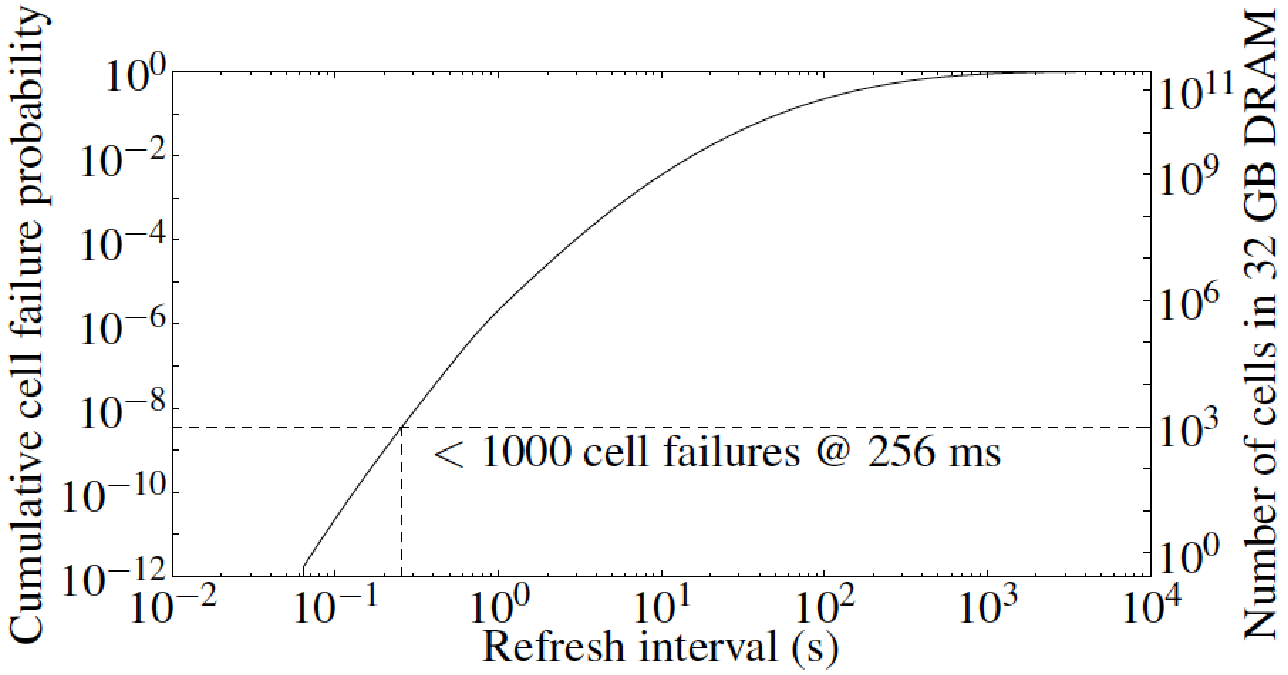


Refresh Overhead: Energy



Problem with Conventional Refresh

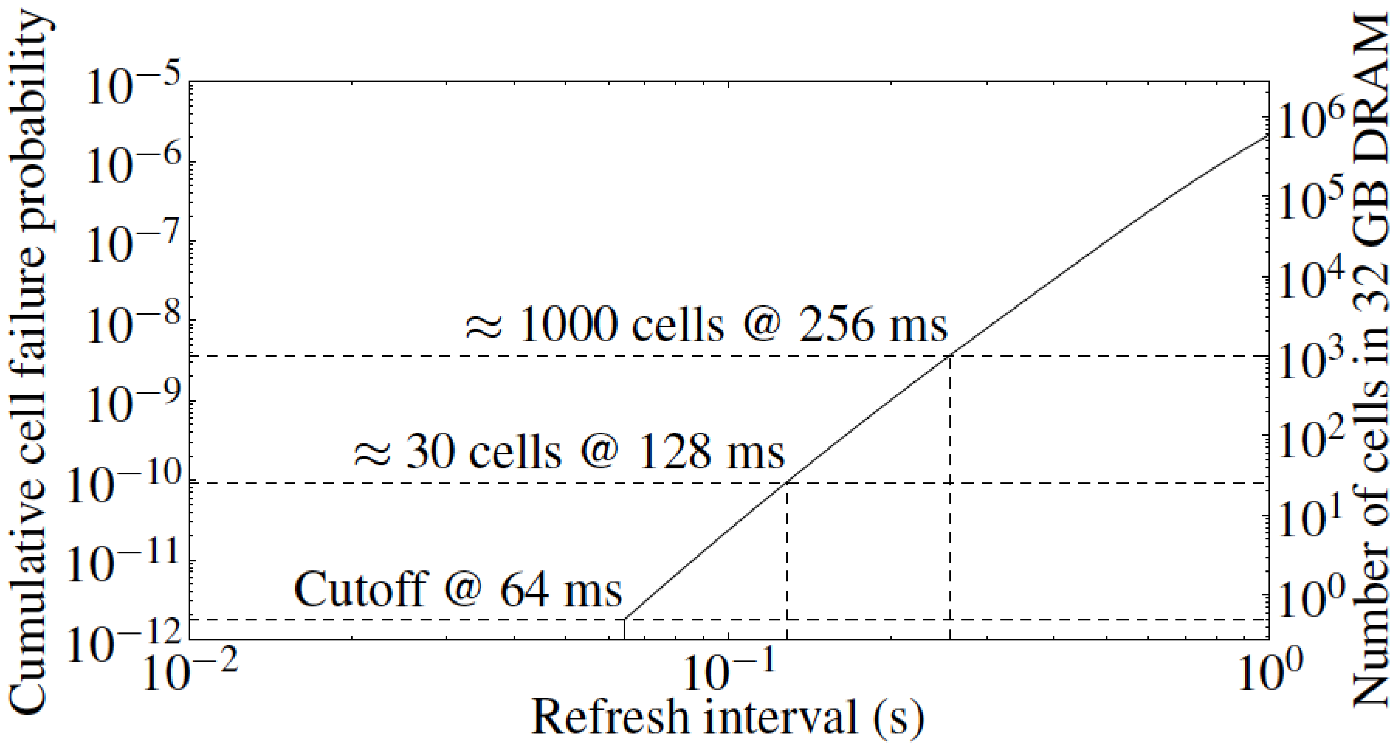
- Today: Every row is refreshed at the same rate



- Observation: Most rows can be refreshed much less often without losing data [Kim+, EDL'09]
- Problem: No support in DRAM for different refresh rates per row

Retention Time of DRAM Rows

- Observation: Only very few rows need to be refreshed at the worst-case rate



- Can we exploit this to reduce refresh operations at low cost?

Reducing DRAM Refresh Operations

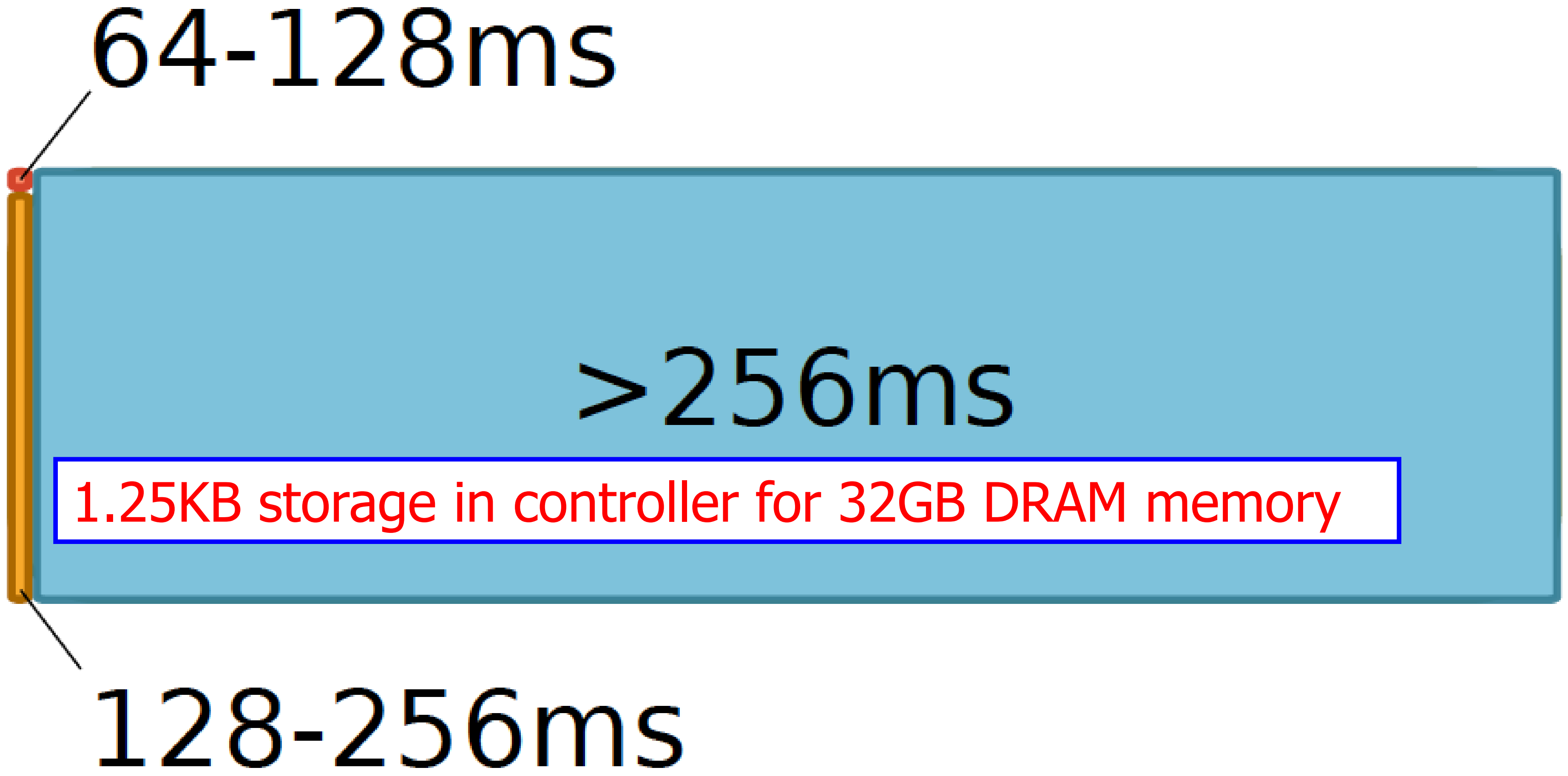
- Idea: Identify the retention time of different rows and refresh each row at the frequency it needs to be refreshed

- (Cost-conscious) Idea: Bin the rows according to their minimum retention times and refresh rows in each bin at the refresh rate specified for the bin
  - e.g., a bin for 64-128ms, another for 128-256ms, …

- Observation: Only very few rows need to be refreshed very frequently [64-128ms] → Have only a few bins → Low HW overhead to achieve large reductions in refresh operations

- Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

RAIDR: Mechanism



1.25KB storage in controller for 32GB DRAM memory

bins at different rates

→ probe Bloom Filters to determine refresh rate of a row

1. Profiling

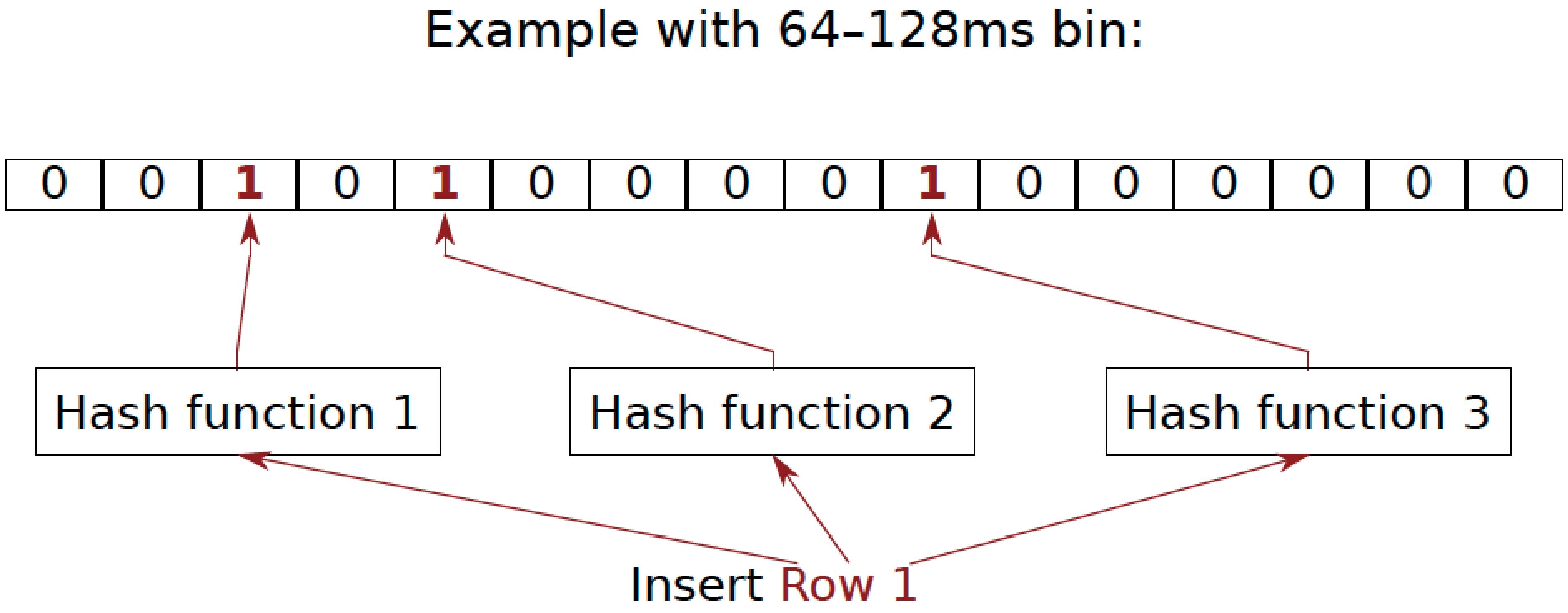
To profile a row:

1. Write data to the row
2. Prevent it from being refreshed
3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111... (64–128ms)	11111111...	11111111...
After 256 ms		11111011... (128–256ms)	11111111... (>256ms)

2. Binning

- How to efficiently and scalably store rows into retention time bins?
- Use Hardware Bloom Filters [Bloom, CACM 1970]



Bloom Filter

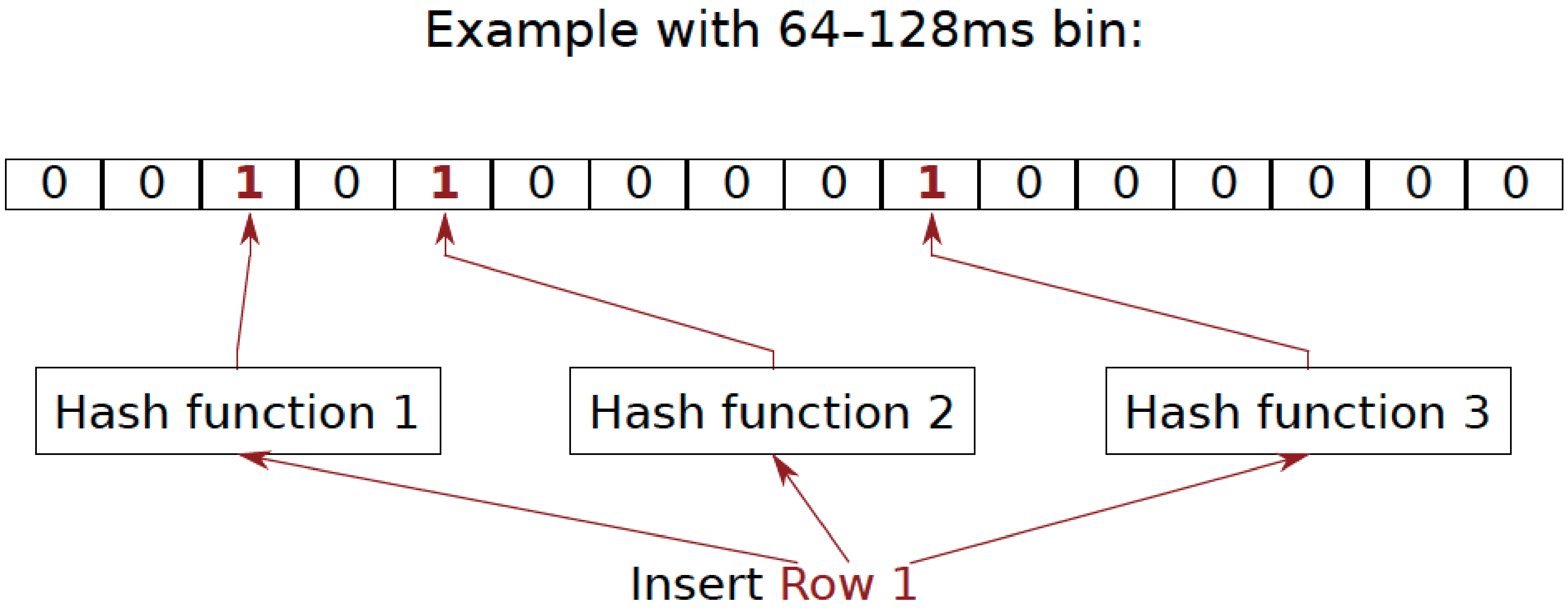
- [Bloom, CACM 1970]
- Probabilistic data structure that compactly represents set membership (presence or absence of element in a set)

- Non-approximate set membership: Use 1 bit per element to indicate absence/presence of each element from an element space of N elements
- Approximate set membership: use a much smaller number of bits and indicate each element's presence/absence with a subset of those bits
  - Some elements map to the bits also mapped to other elements

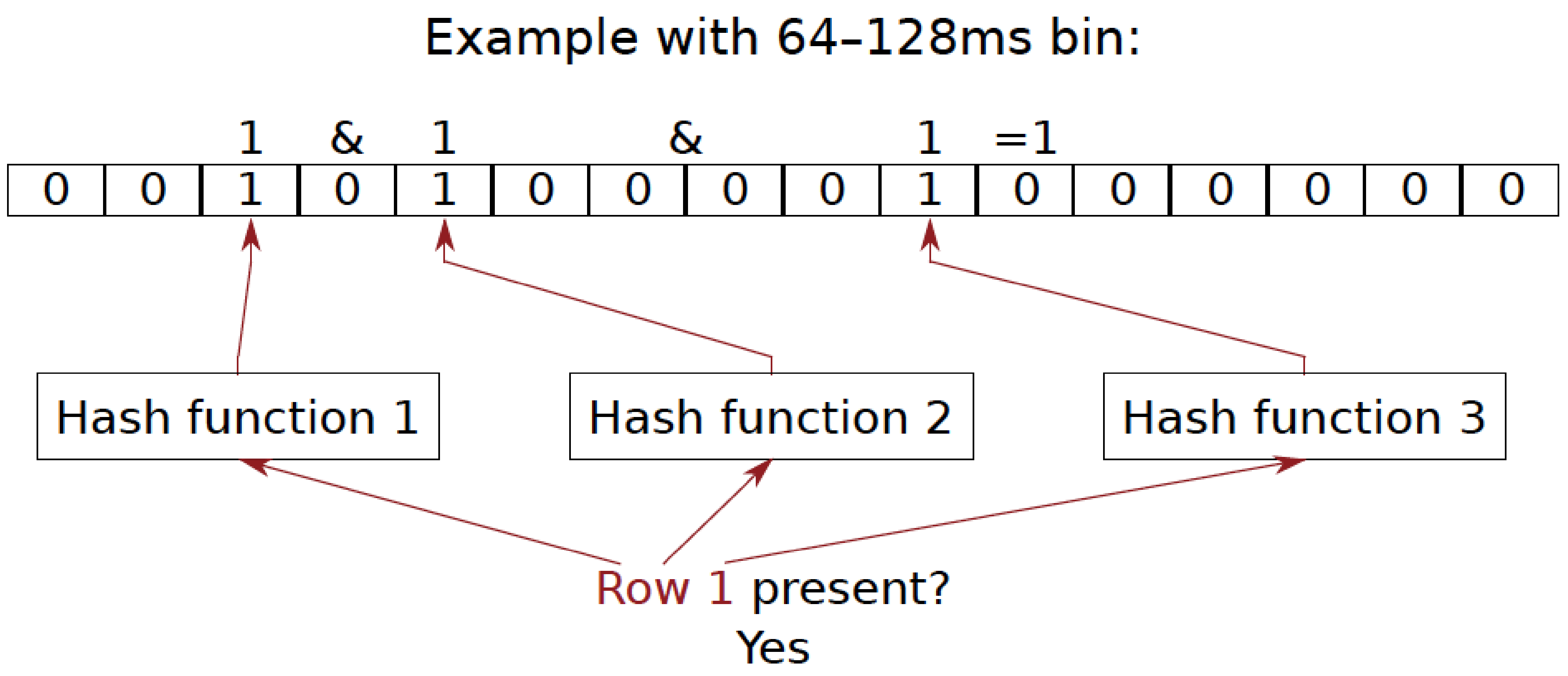
- Operations: 1) insert, 2) test, 3) remove all elements

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

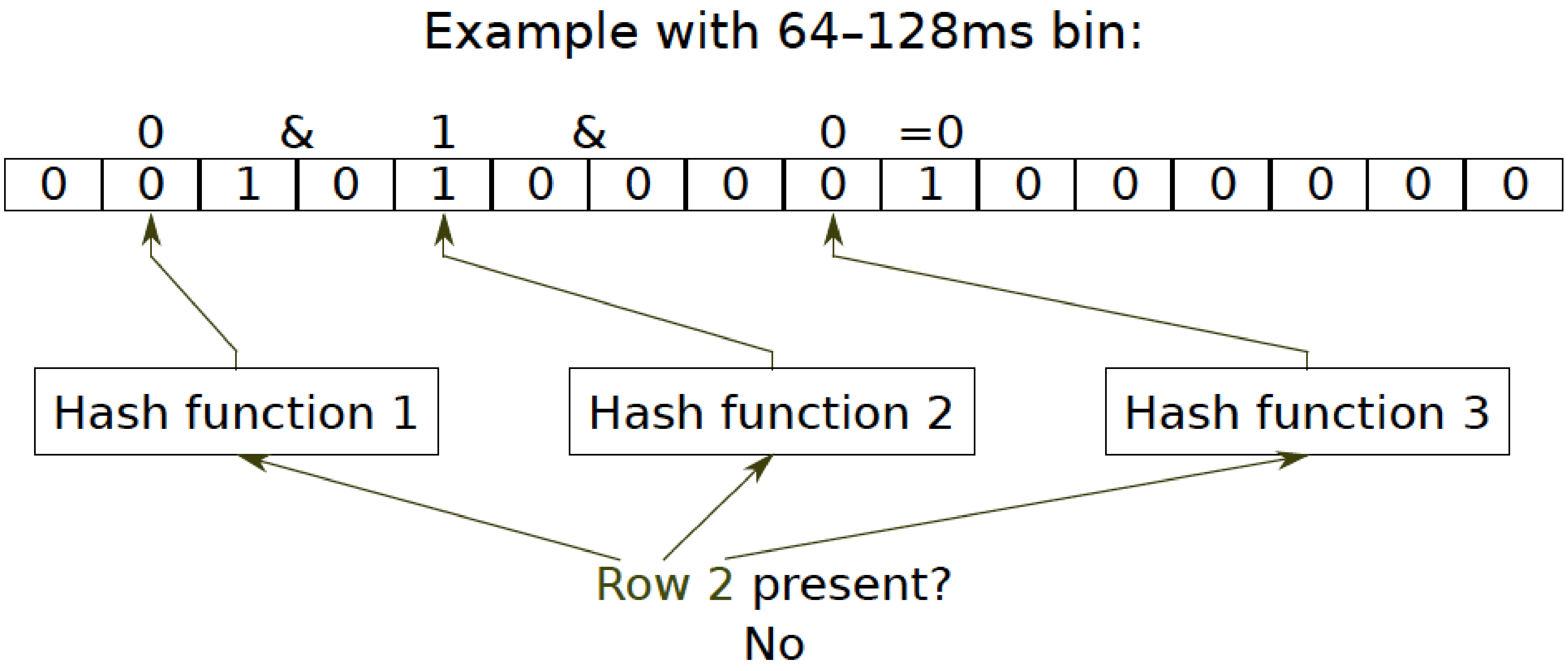
Bloom Filter Operation Example



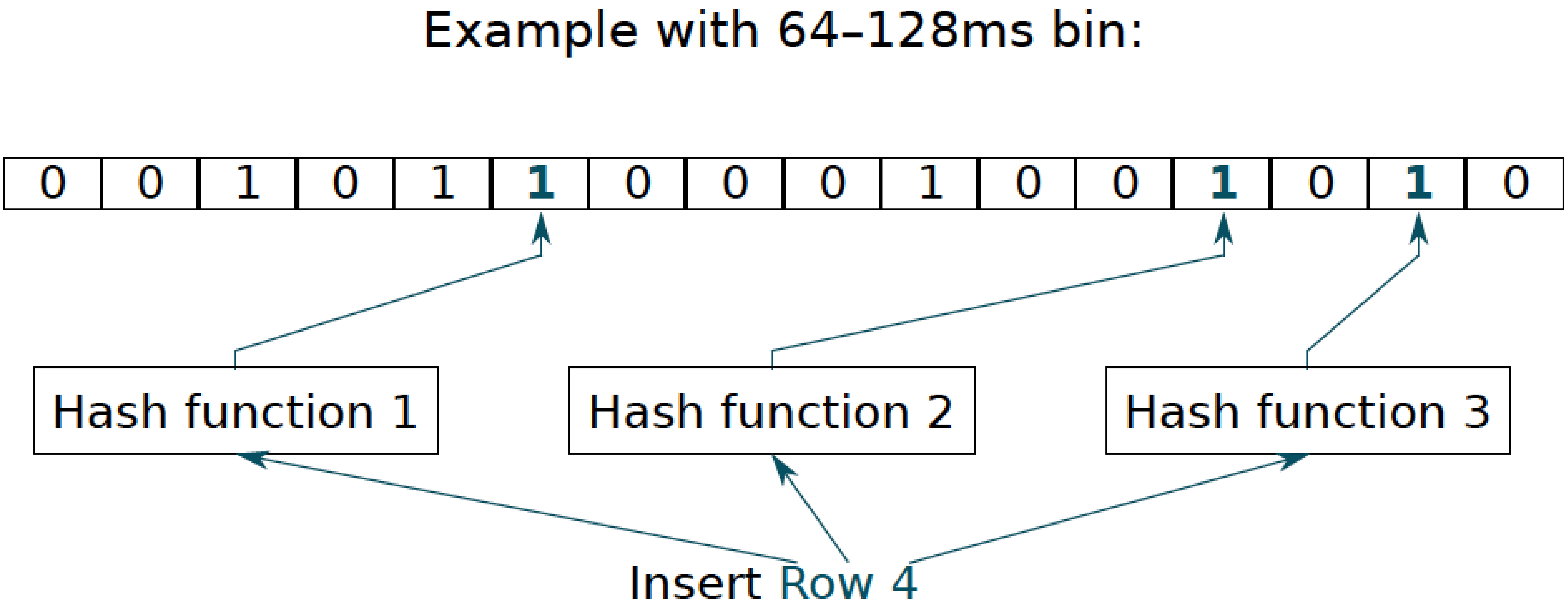
Bloom Filter Operation Example



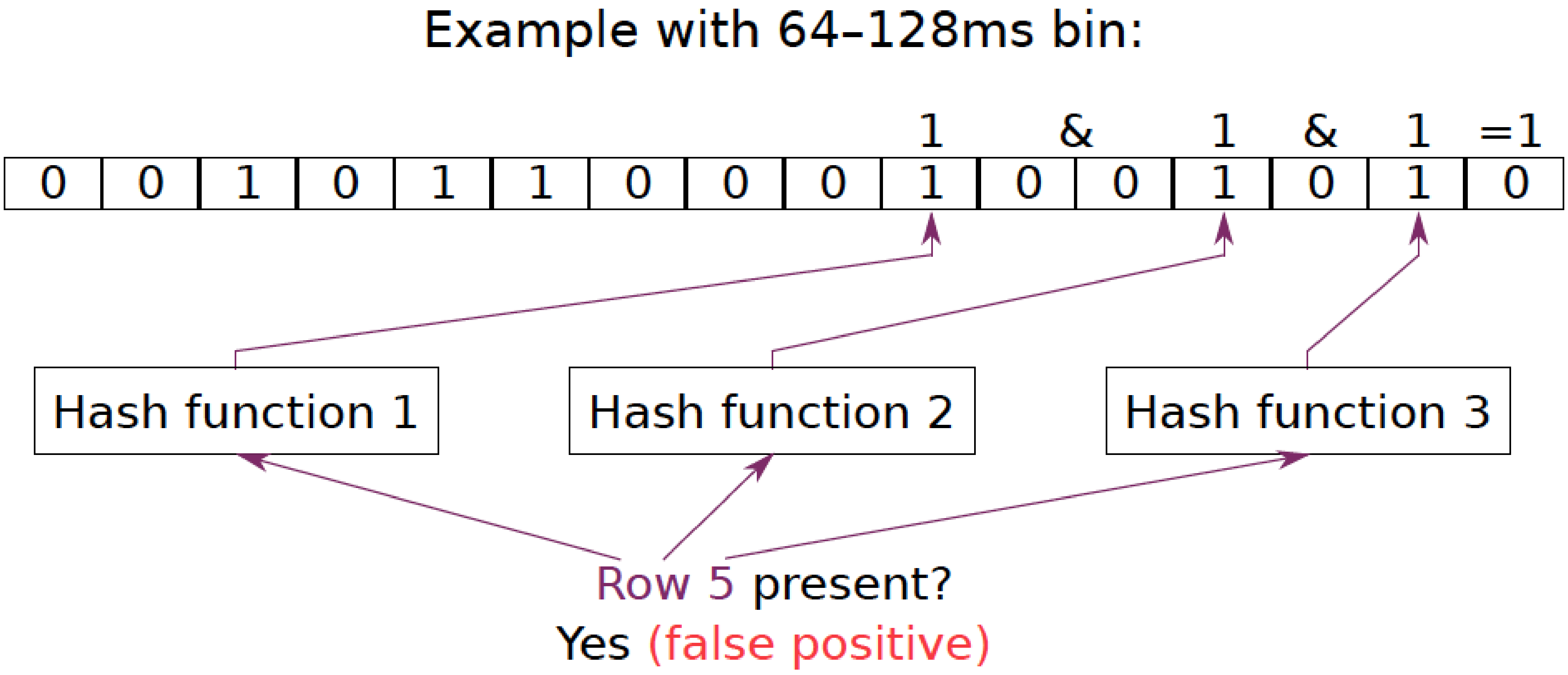
Bloom Filter Operation Example



Bloom Filter Operation Example



Bloom Filter Operation Example



Benefits of Bloom Filters as Bins

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Refresh some rows more frequently than needed

- No false negatives: rows are never refreshed less frequently than needed (no correctness problems)

- Scalable: a Bloom filter never overflows (unlike a fixed-size table)

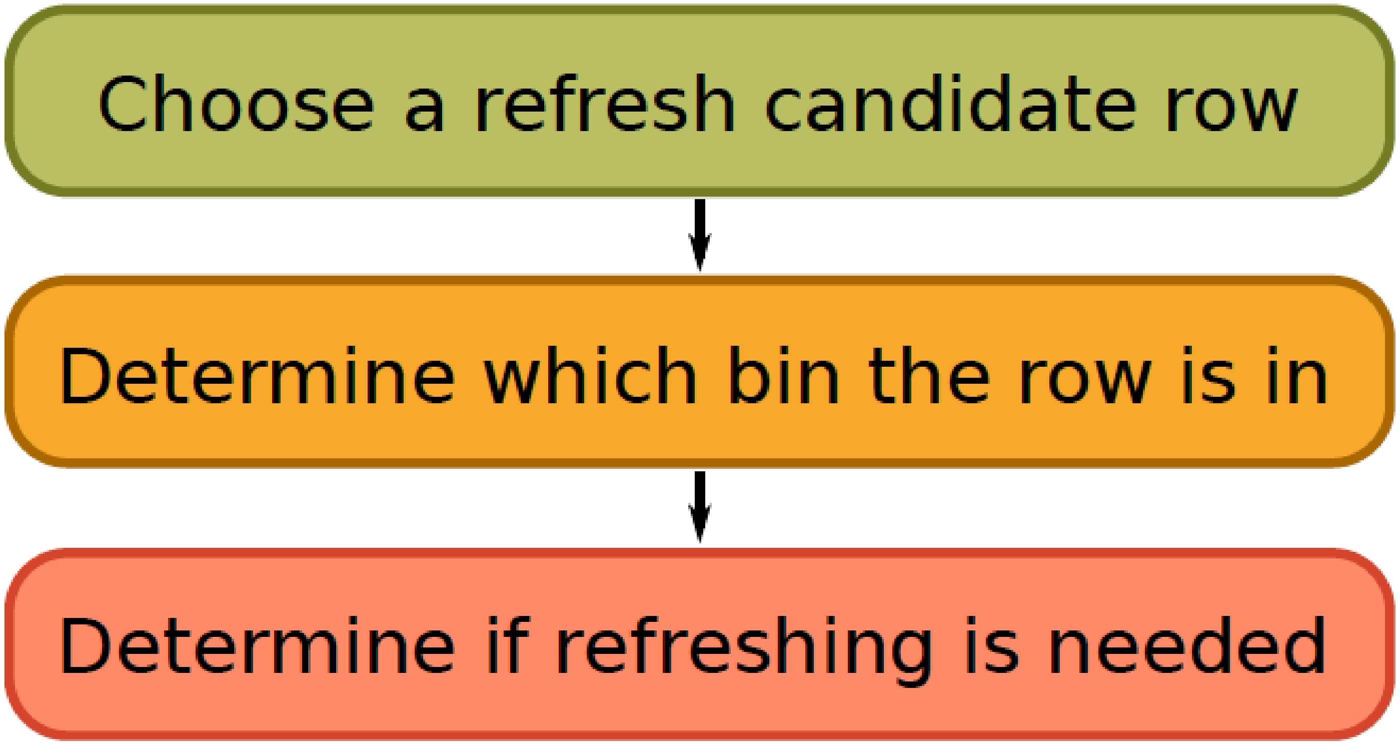
- Efficient: No need to store info on a per-row basis; simple hardware → 1.25 KB for 2 filters for 32 GB DRAM system

Use of Bloom Filters in Hardware

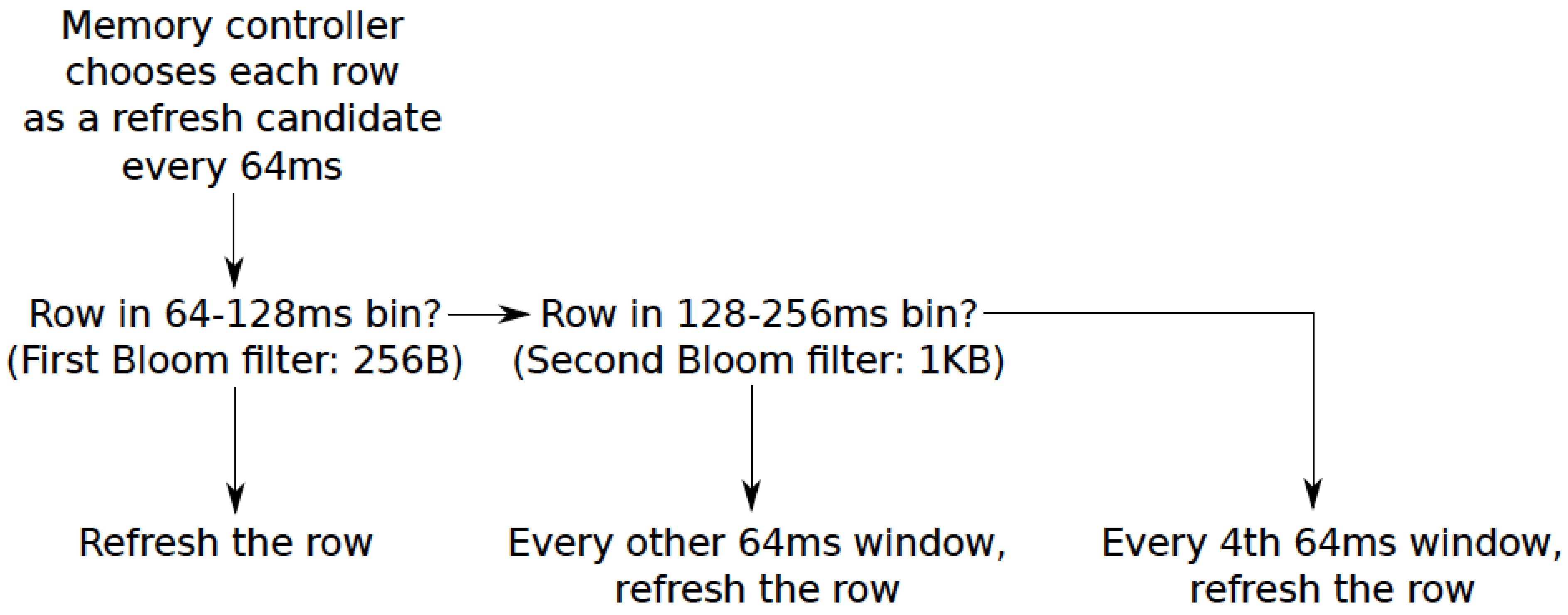
- Useful when you can tolerate false positives in set membership tests

- See the following recent examples for clear descriptions of how Bloom Filters are used
  - Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
  - Seshadri et al., "The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing," PACT 2012.

3. Refreshing (RAIDR Refresh Controller)

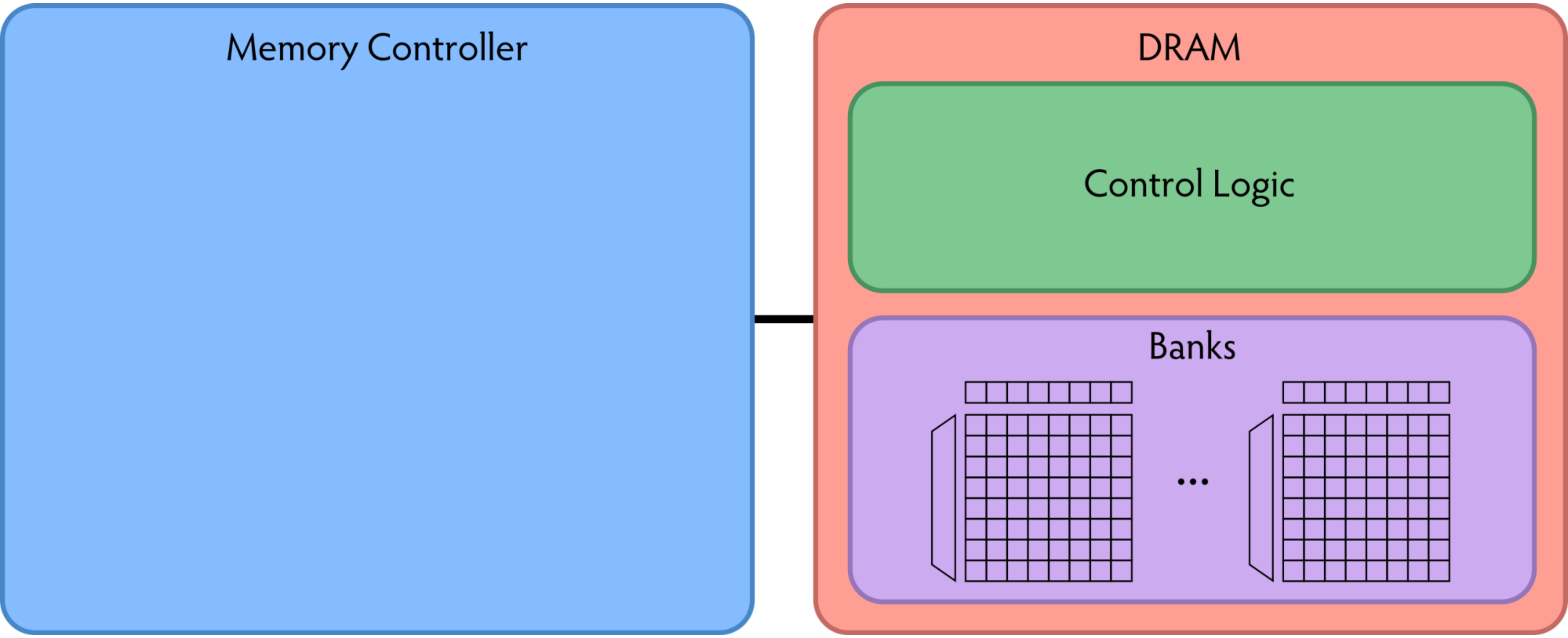

Choose a refresh candidate row
Determine which bin the row is in
Determine if refreshing is needed

3. Refreshing (RAIDR Refresh Controller)


Memory controller chooses each row as a refresh candidate every 64ms
Row in 64-128ms bin? (First Bloom filter: 256B)
Row in 128-256ms bin? (Second Bloom filter: 1KB)
Refresh the row
Every other 64ms window, refresh the row
Every 4th 64ms window, refresh the row

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

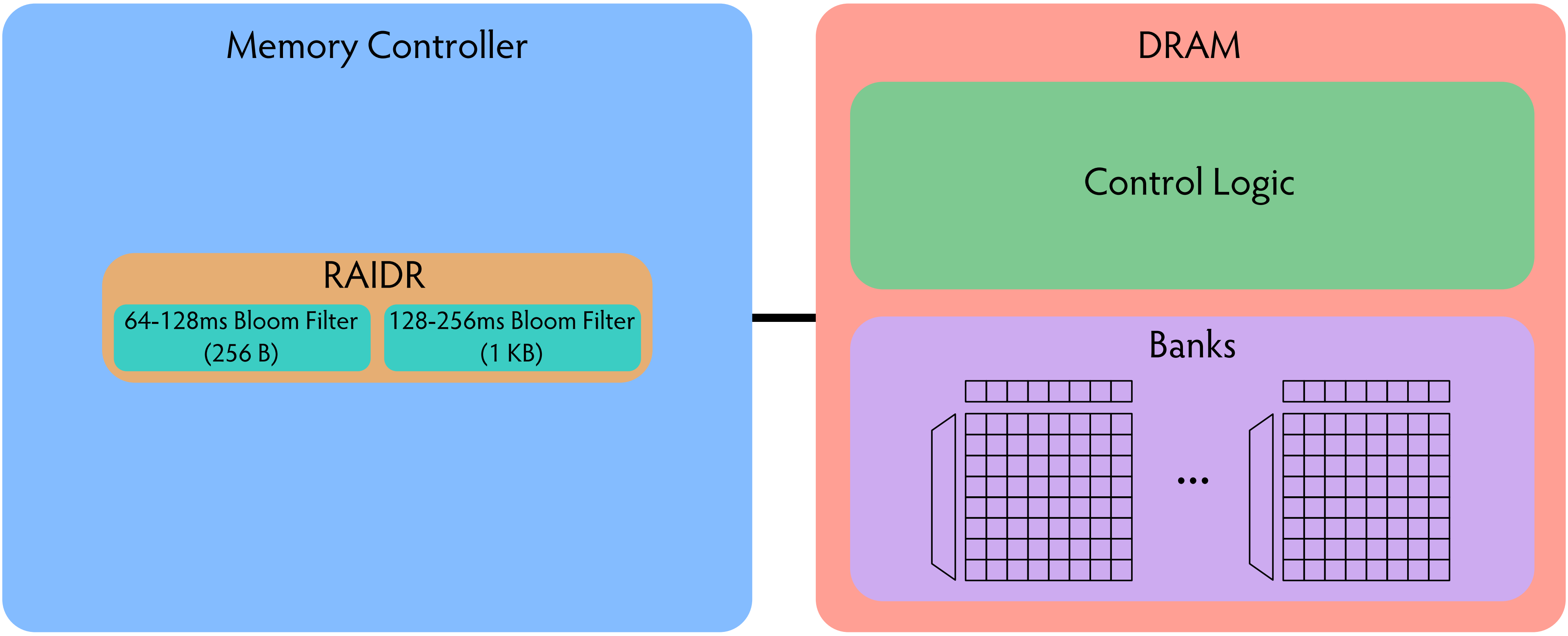
RAIDR: Baseline Design



Refresh control is in DRAM in today's auto-refresh systems

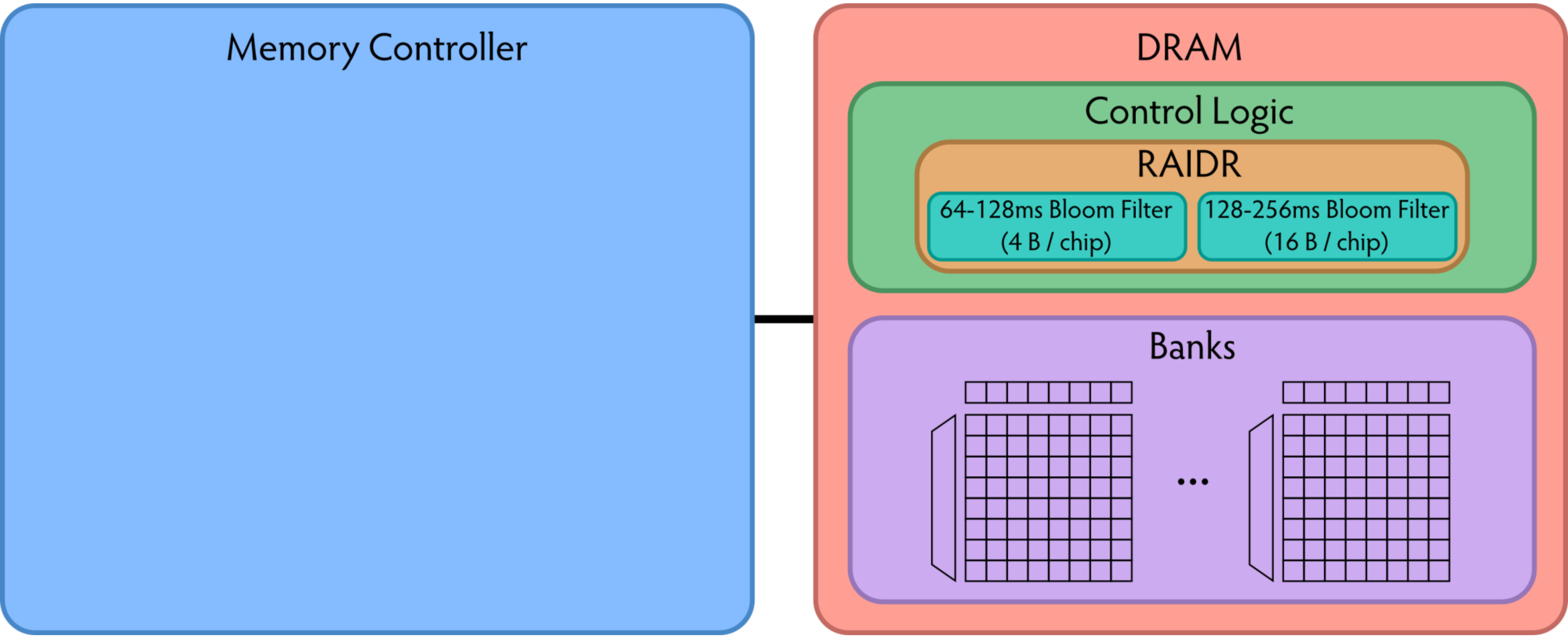
RAIDR can be implemented in either the controller or DRAM

RAIDR in Memory Controller: Option 1



Overhead of RAIDR in DRAM controller:
1.25 KB Bloom Filters, 3 counters, additional commands issued for per-row refresh (all accounted for in evaluations)

RAIDR in DRAM Chip: Option 2


Memory Controller
DRAM
Control Logic
RAIDR
64-128ms Bloom Filter (4 B / chip)
128-256ms Bloom Filter (16 B / chip)
Banks
...

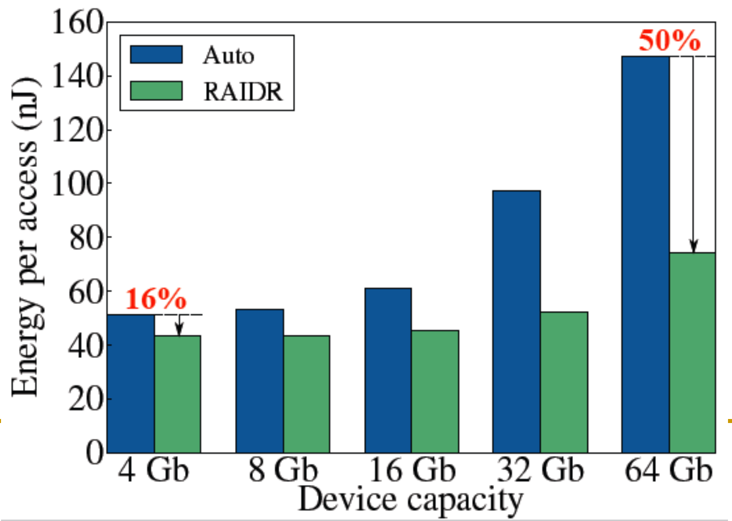
Overhead of RAIDR in DRAM chip:
Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)
Total overhead: 1.25KB Bloom Filters, 64 counters (32 GB DRAM)

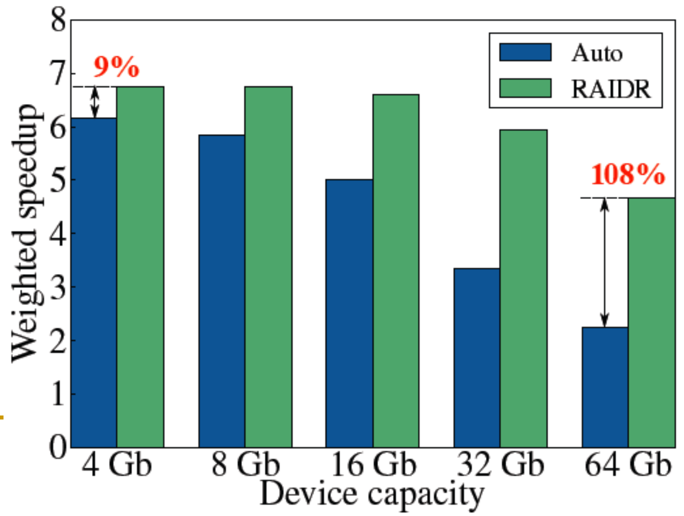
RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; SPEC, TPC-C, TPC-H workloads

- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%

- Benefits increase as DRAM scales in density





DRAM Refresh: More Questions

- What else can you do to reduce the impact of refresh?
- What else can you do if you know the retention times of rows?
- How can you accurately measure the retention time of DRAM rows?
- Recommended reading:
  - Liu et al., "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," ISCA 2013.