Cache and Main Memory

Review (and more)

* Slide title w/o “Review” are new materials

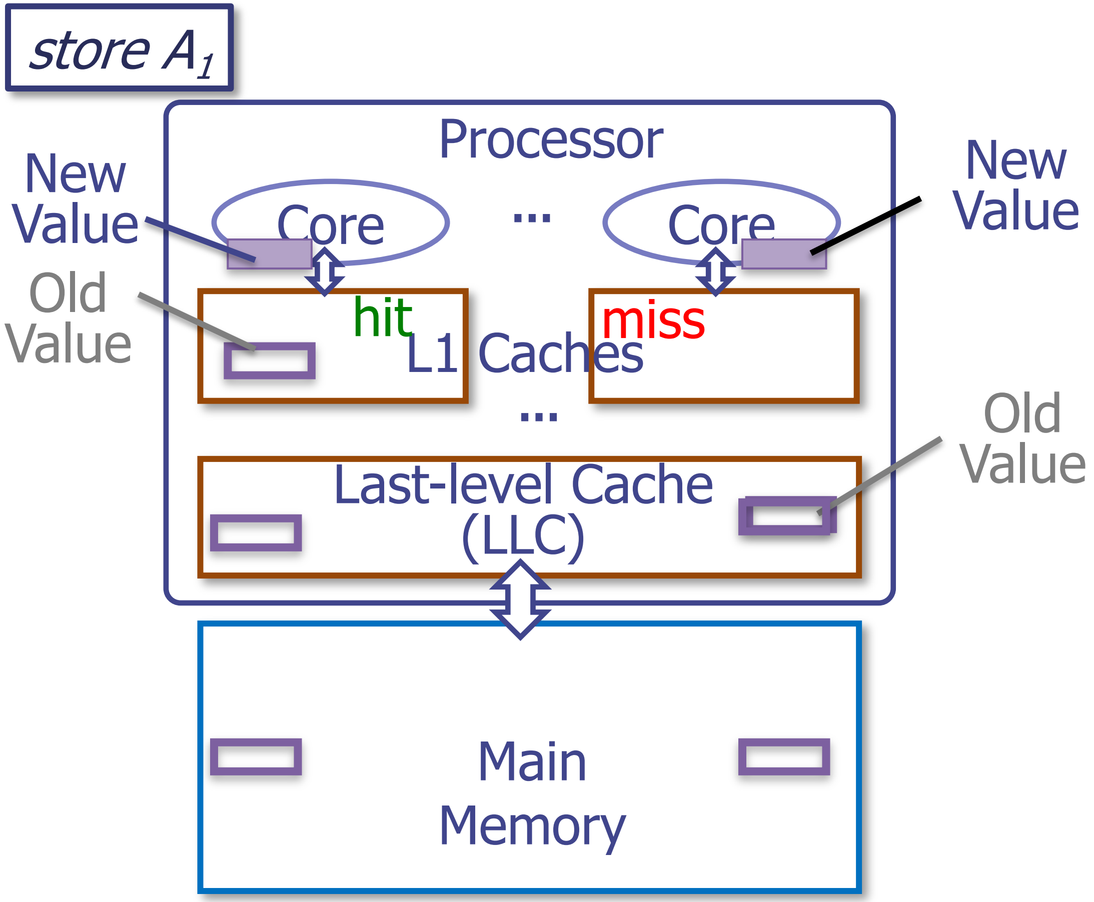
Review: Handling Cache Hits

- Read hits (I$ and D$)
  - this is what we want – no challenges
- Write hits (D$ only) -- Important!
  - Method 1: Allow cache and memory to be inconsistent → "write-back" cache
    - write the data only into the cache block (write-back the cache contents to the next level in the memory hierarchy when that cache block is "evicted")
    - (need a dirty bit for each data cache block to tell if it needs to be written back to memory when it is evicted)
  - Method 2: Require the cache and memory to be consistent
    - always write the data into both the cache block and the next level in the memory hierarchy (write-through) so don't need a dirty bit
    - writes run at the speed of the next level in the memory hierarchy – slow

Review:

Handle writes in write-back cache

"Write-back Write-allocate"

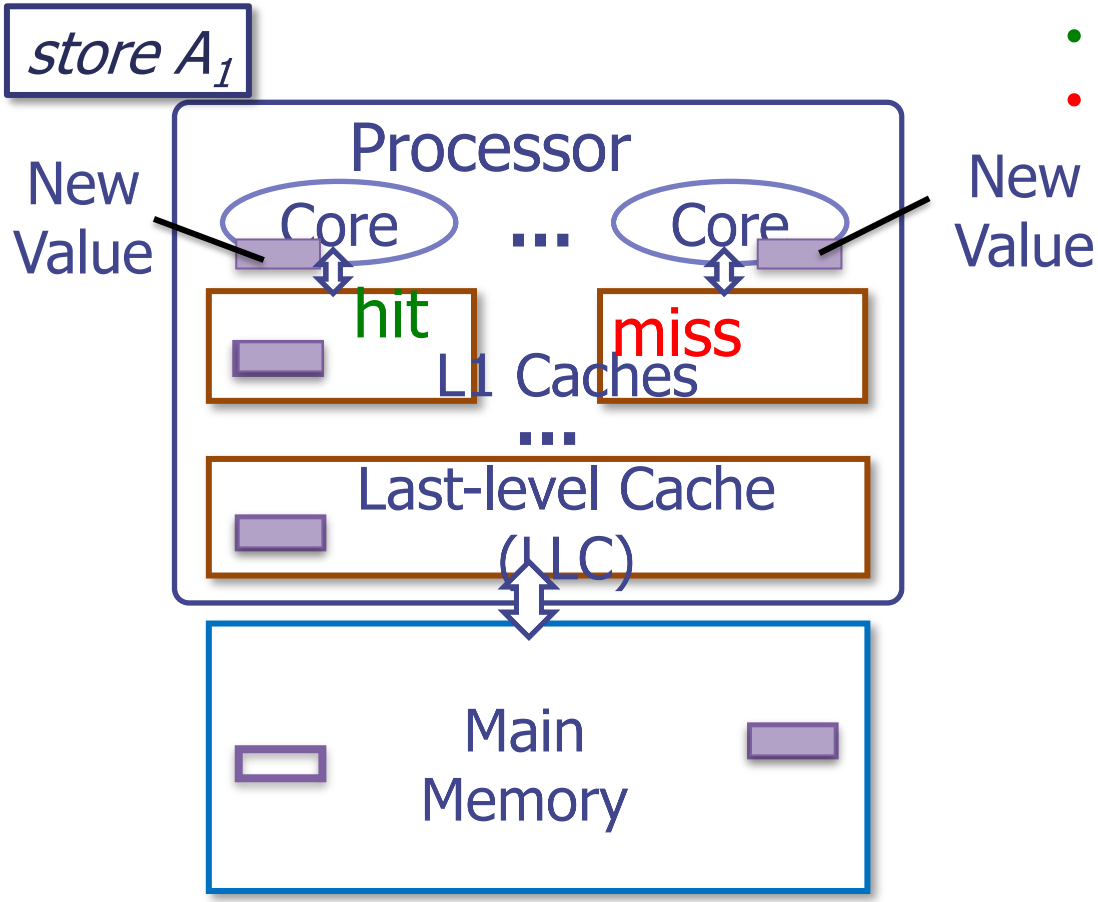


- On write hit: write-back
- On write miss: write allocate, i.e., copy the old value from a lower level all the way up to L1

Review:

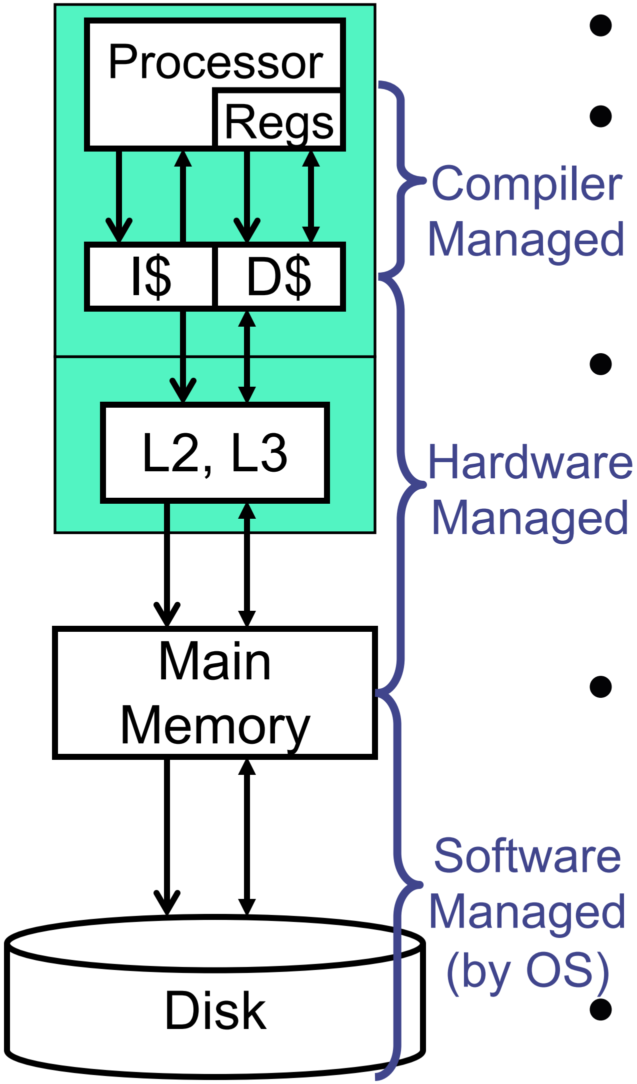
Handle writes in write-through cache

"Write-through Write-no-allocate"



- On write hit: write through
- On write miss: write-no-allocate, i.e., directly write to main memory, rather than bringing the cache block into caches (only bring cache blocks into caches on read misses)

Review: Concrete Memory Hierarchy



- 0th level: **Registers**
- 1st level: **Primary caches**
  - Split instruction (I$) and data (D$)
  - Typically 8KB to 64KB each
- 2nd level: **2nd and 3rd cache** (L2, L3)
  - On-chip, typically made of SRAM
  - 2nd level typically ~256KB to 512KB
  - "Last level cache" typically 4MB to 16MB
- 3rd level: **main memory**
  - Made of DRAM ("Dynamic" RAM)
  - Typically 1GB to 4GB for desktops/laptops
    - Servers can have >1 TB
- 4th level: **disk (swap and files)**
  - Uses magnetic disks or flash drives

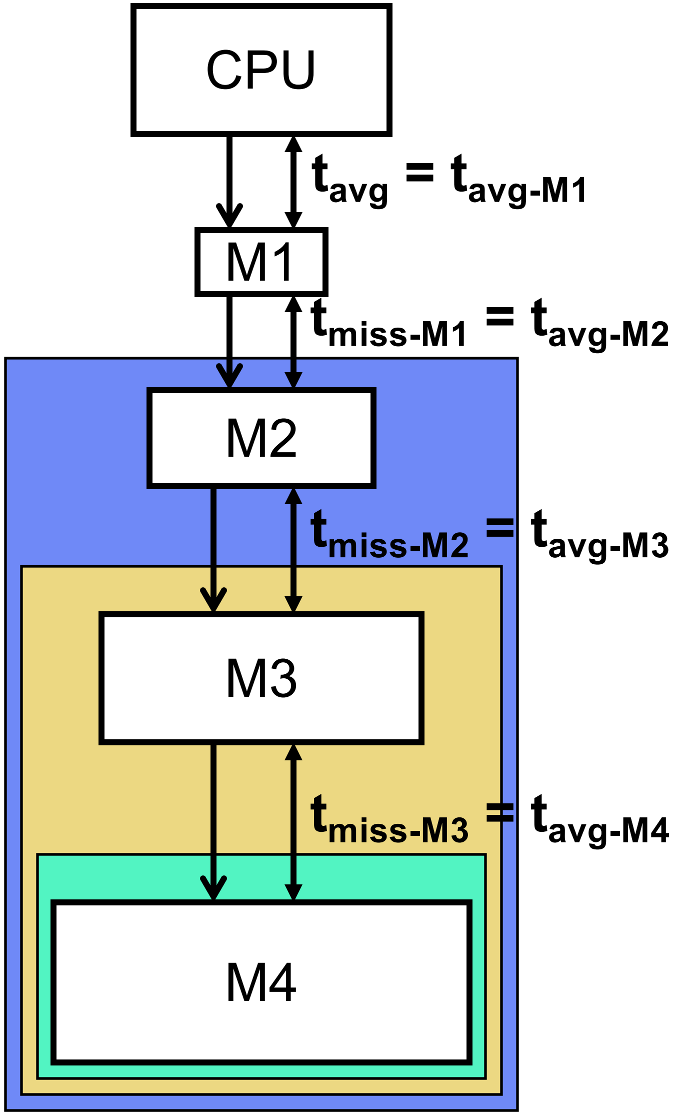
Review: Designing a cache hierarchy

- For any memory component: t_{hit} vs. $\%_{miss}$ tradeoff

- Upper components (I$, D$) emphasize low t_{hit}
  - **Frequent access → t_{hit} important**
  - t_{miss} is low → $\%_{miss}$ less important
  - Lower capacity and lower associativity (to reduce t_{hit})
  - Small-medium block-size (to reduce conflicts)
  - **Split instruction & data cache to allow simultaneous access**

- Moving down (L2, L3) emphasis turns to $\%_{miss}$
  - **Infrequent access → t_{hit} less important**
  - t_{miss} is high → $\%_{miss}$ important
  - High capacity, associativity, and block size (to reduce $\%_{miss}$)
  - **Unified insn+data caches to dynamically allocate capacity**

Review: Hierarchy Performance -- AMAT


CPU
$t_{avg} = t_{avg\text{-}M1}$
M1
$t_{miss\text{-}M1} = t_{avg\text{-}M2}$
M2
$t_{miss\text{-}M2} = t_{avg\text{-}M3}$
M3
$t_{miss\text{-}M3} = t_{avg\text{-}M4}$
M4

t_{avg}

$t_{avg\text{-}M1}$

$= t_{hit\text{-}M1} + (\%_{miss\text{-}M1} * t_{miss\text{-}M1})$

$= t_{hit\text{-}M1} + (\%_{miss\text{-}M1} * t_{avg\text{-}M2})$

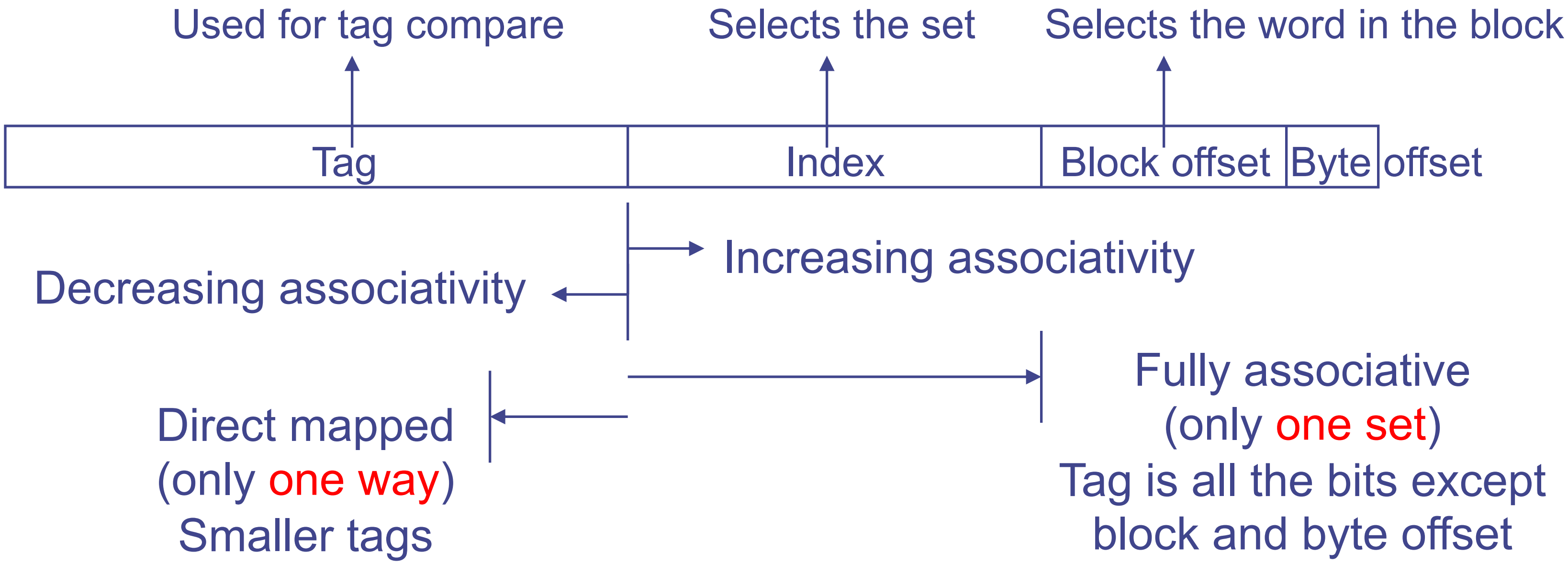
$= t_{hit\text{-}M1} + (\%_{miss\text{-}M1} * (t_{hit\text{-}M2} + (\%_{miss\text{-}M2} * t_{miss\text{-}M2})))$

$= t_{hit\text{-}M1} + (\%_{miss\text{-}M1} * (t_{hit\text{-}M2} + (\%_{miss\text{-}M2} * t_{avg\text{-}M3})))$

…

Review: Range of Set Associative Caches

- For a fixed size cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit

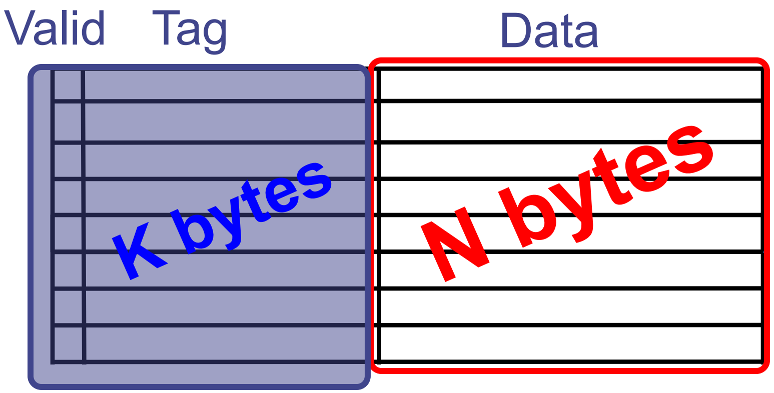


Review: 3C Model

- Divide cache misses into three categories
  - **Compulsory (cold)**: never seen this address before
    - **Would miss even in infinite cache**
  - **Capacity**: miss caused because cache is too small
    - **Would miss even in fully associative cache**
    - Identify? Consecutive accesses to block separated by access to at least N other distinct blocks (N is number of frames in cache)
  - **Conflict**: miss caused because cache associativity is too low
    - Identify? **All other misses**
  - **(Coherence)**: miss due to external invalidations
    - Only in shared memory multiprocessors (later)
- Calculated by multiple simulations
  - Simulate infinite cache, fully-associative cache, normal cache
  - Subtract to find each count

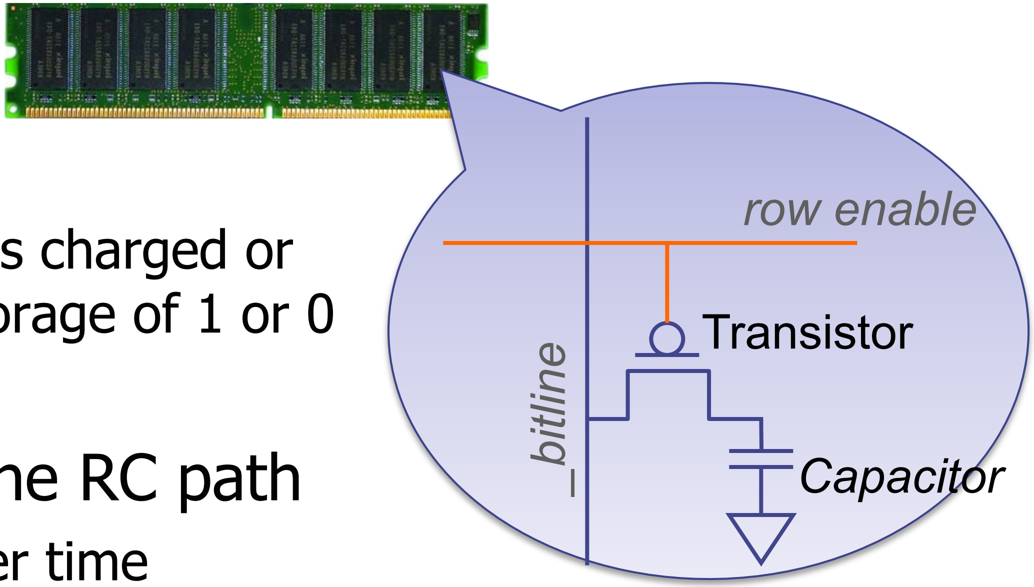
Review: How to calculate tag overhead

- Tag overhead = (K / N) * 100%



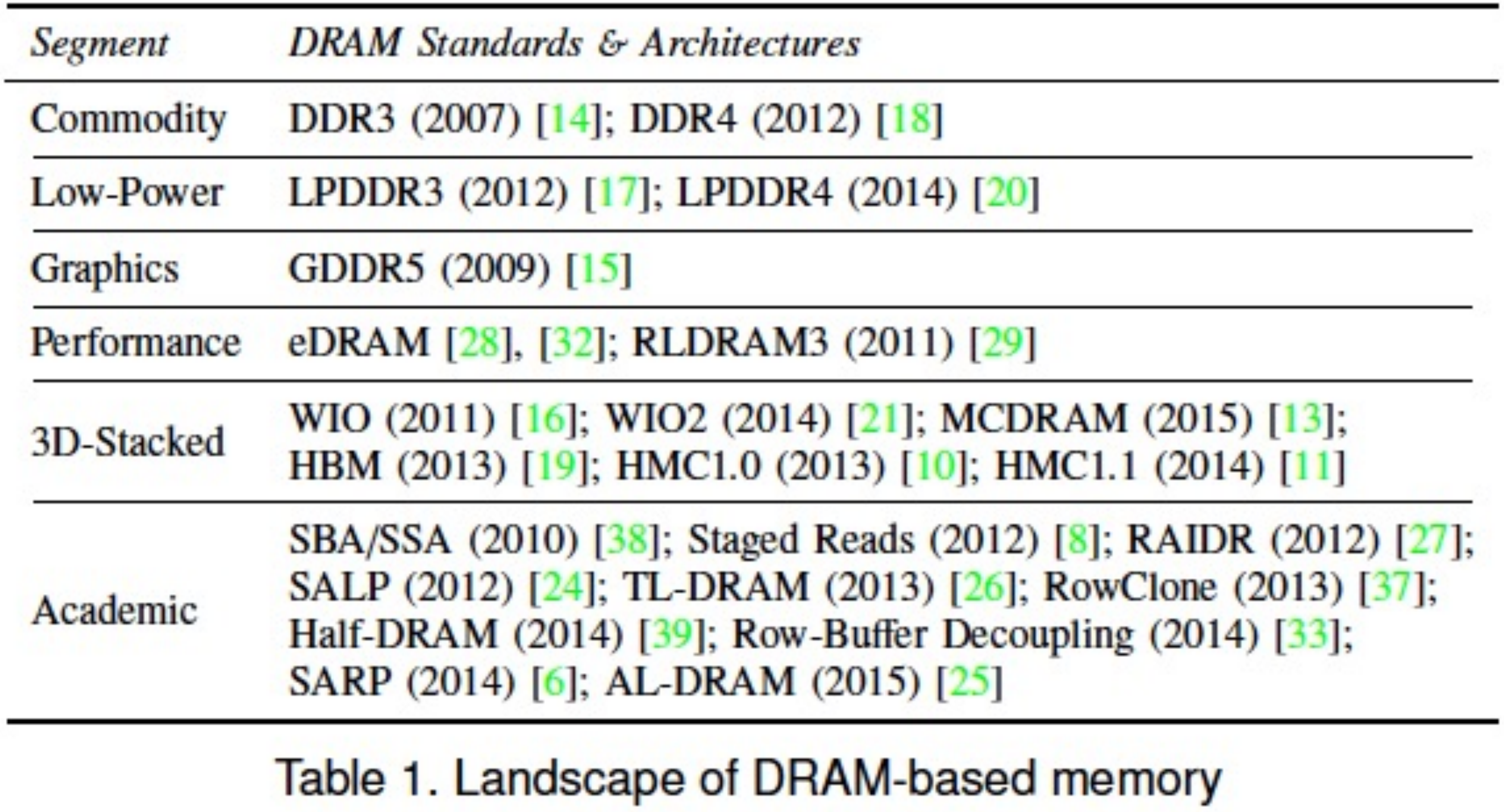
Review: Main Memory technology -- DRAM

- Dynamic random access memory
- Capacitor charge state indicates stored value
  - 1T1C
  - 1 access transistor
  - 1 capacitor
    - Whether the capacitor is charged or discharged indicates storage of 1 or 0



- Capacitor leaks through the RC path
  - DRAM cell loses charge over time
  - DRAM cell needs to be refreshed

DRAM Types



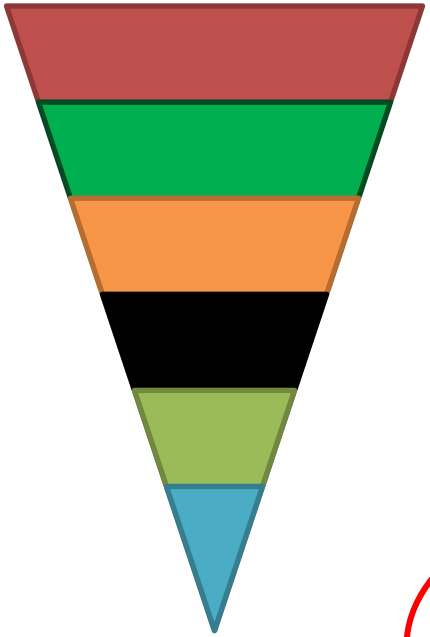
Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

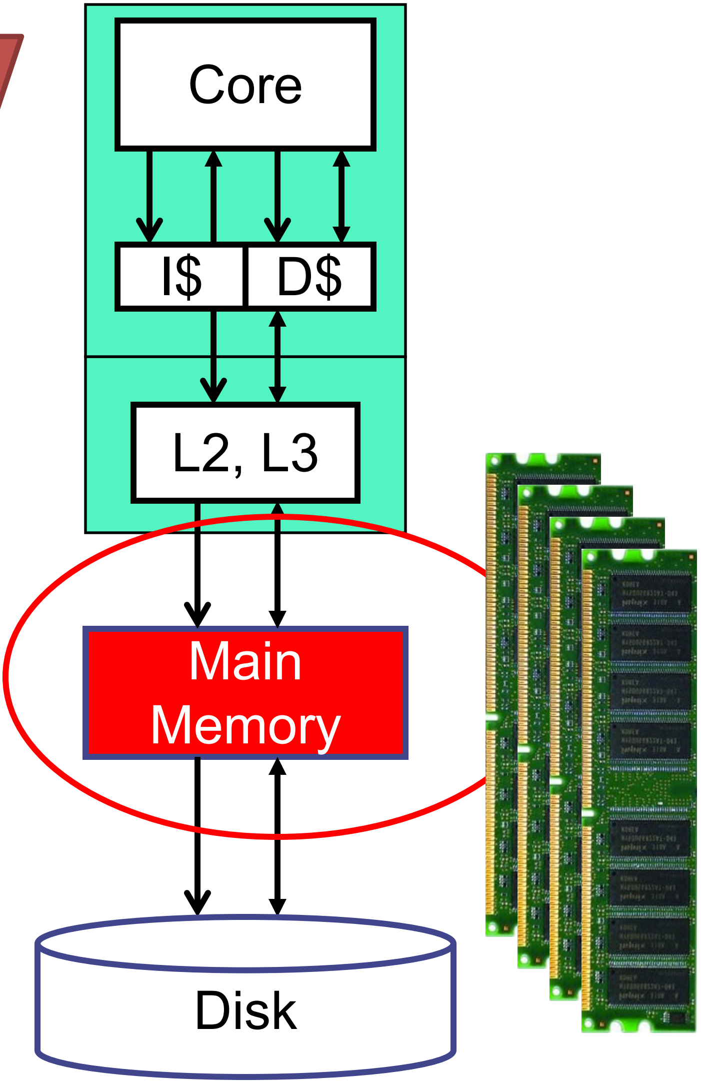
Table 1. Landscape of DRAM-based memory

Kim et al., “Ramulator: A Fast and Extensible DRAM Simulator,” IEEE Comp Arch Letters 2015.

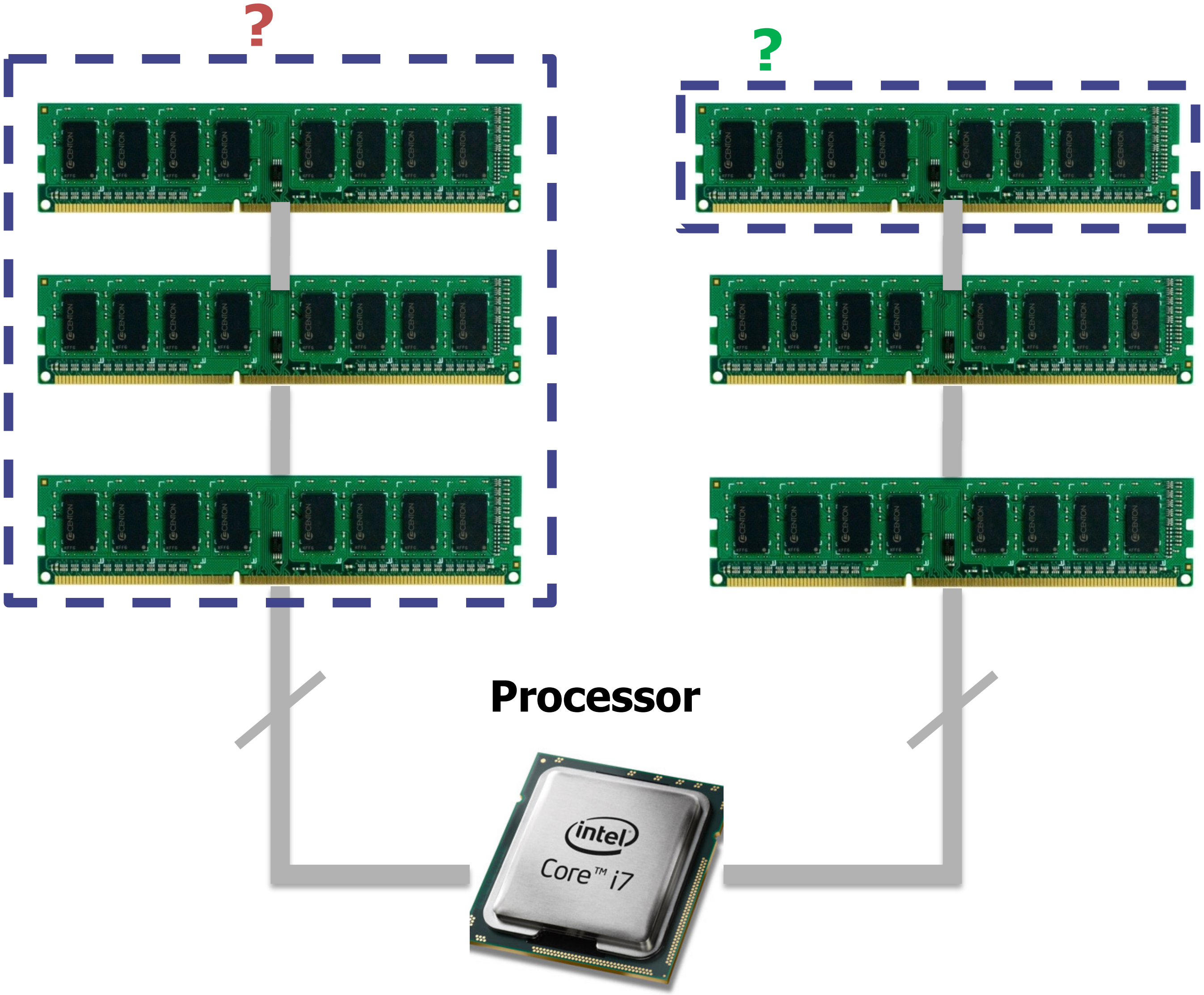
Review: Main memory organization

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



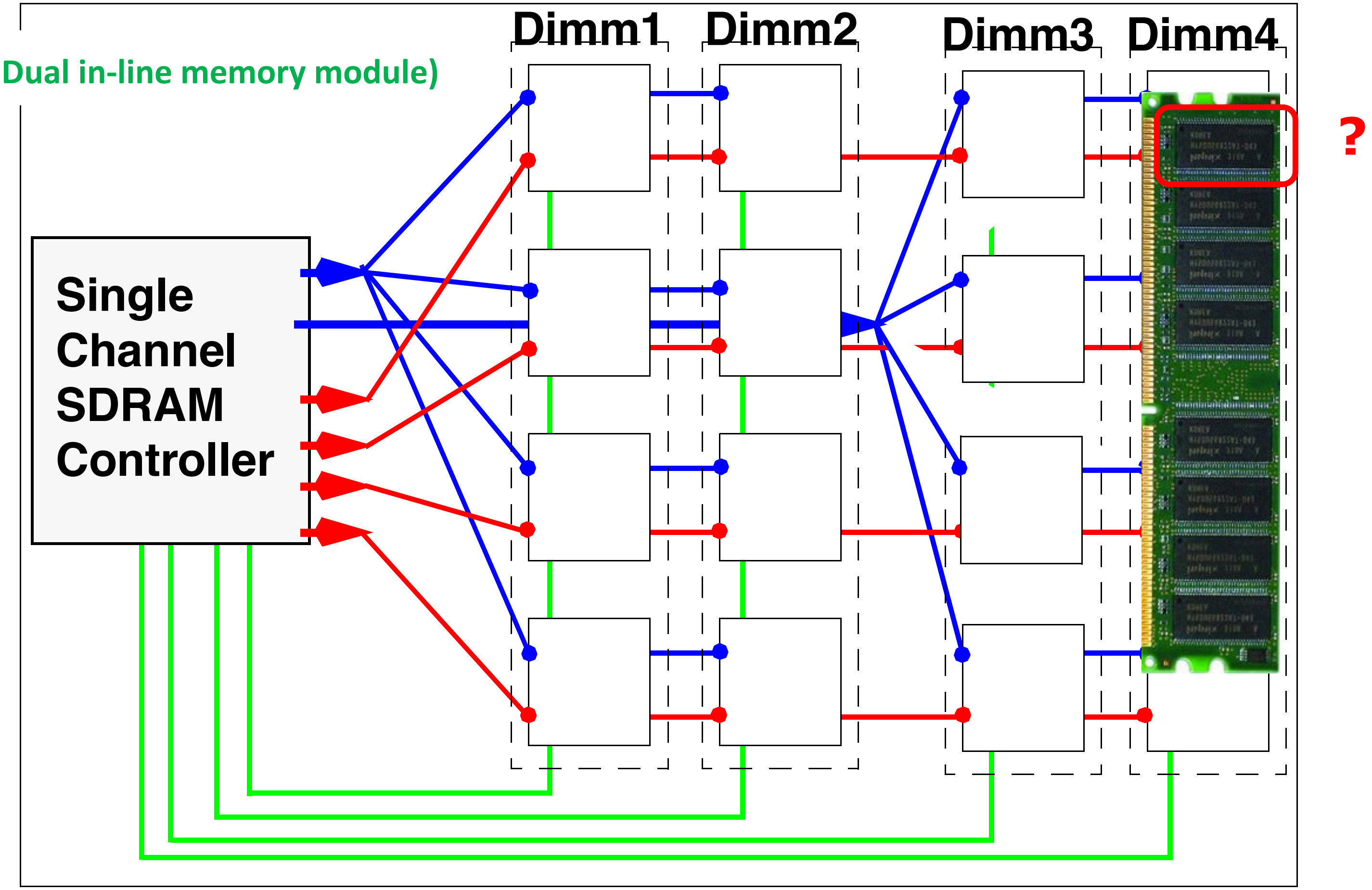


Review: Test yourself



Review: Test yourself

DIMM (Dual in-line memory module)



“Mesh Topology”

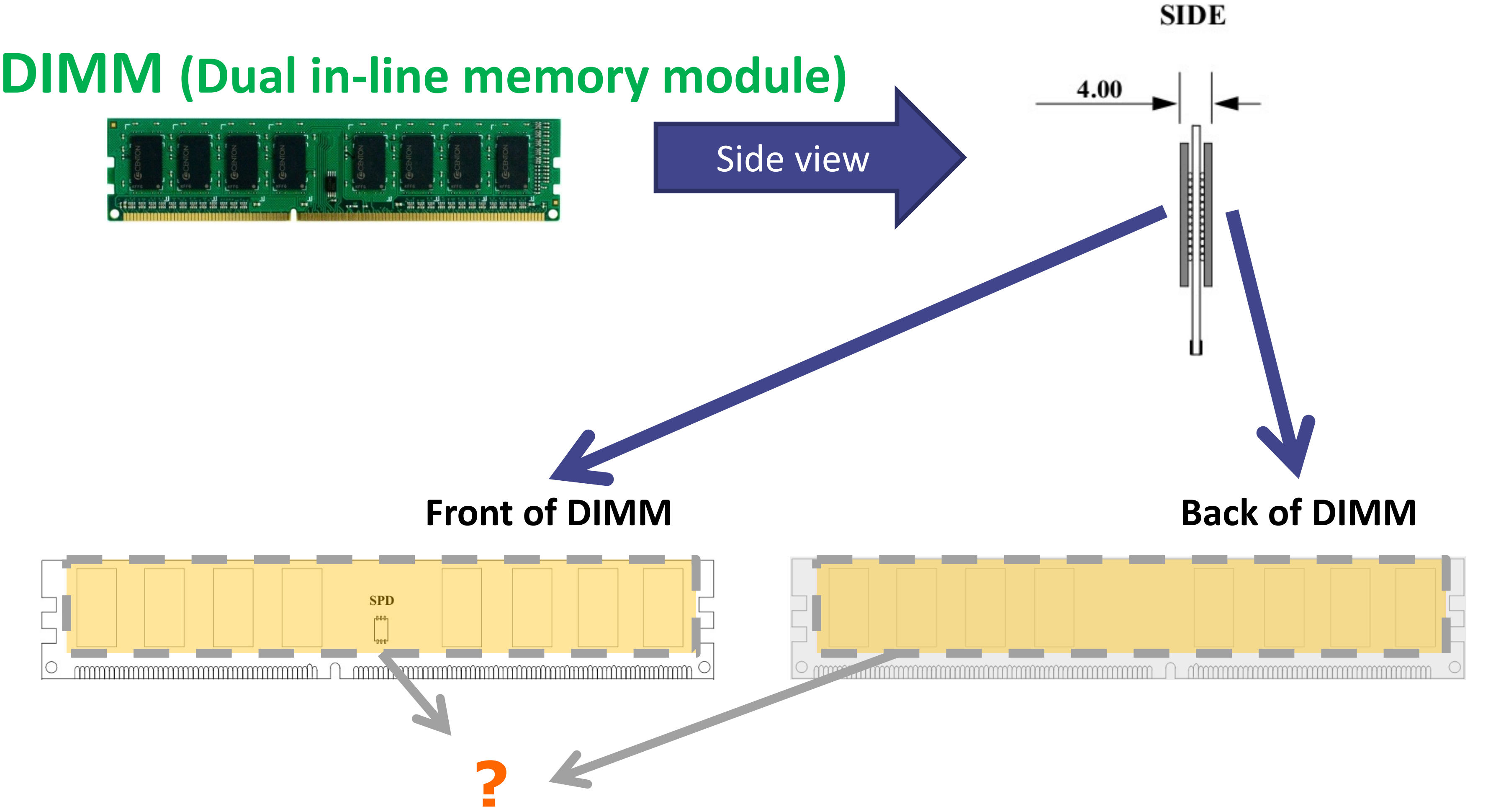
— Addr & Cmd

— Data Bus

— Chip (DIMM) Select

Source: Bruce Jacob & David Wang

Review: Test yourself



DIMM (Dual in-line memory module)

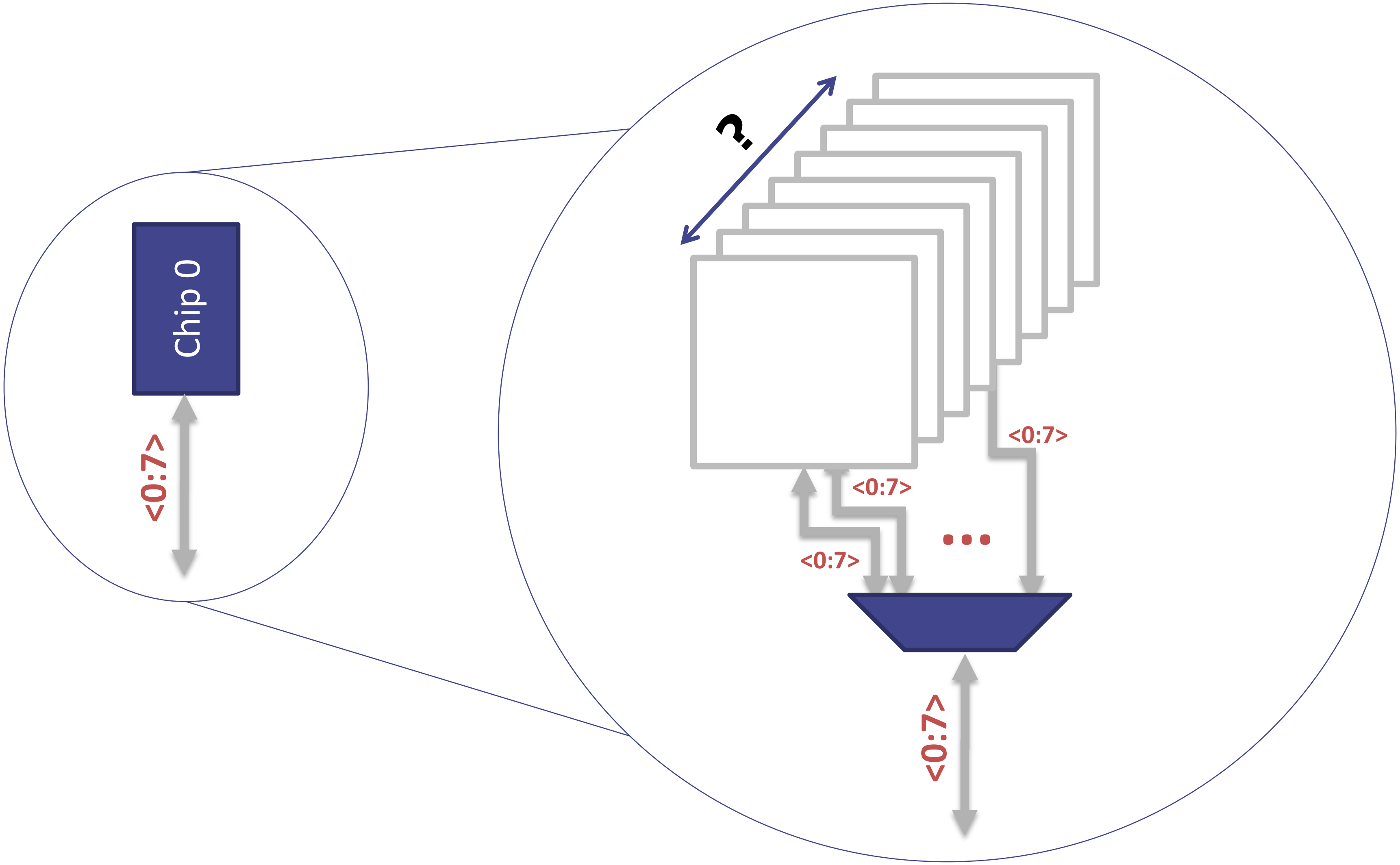
Side view

Front of DIMM

Back of DIMM

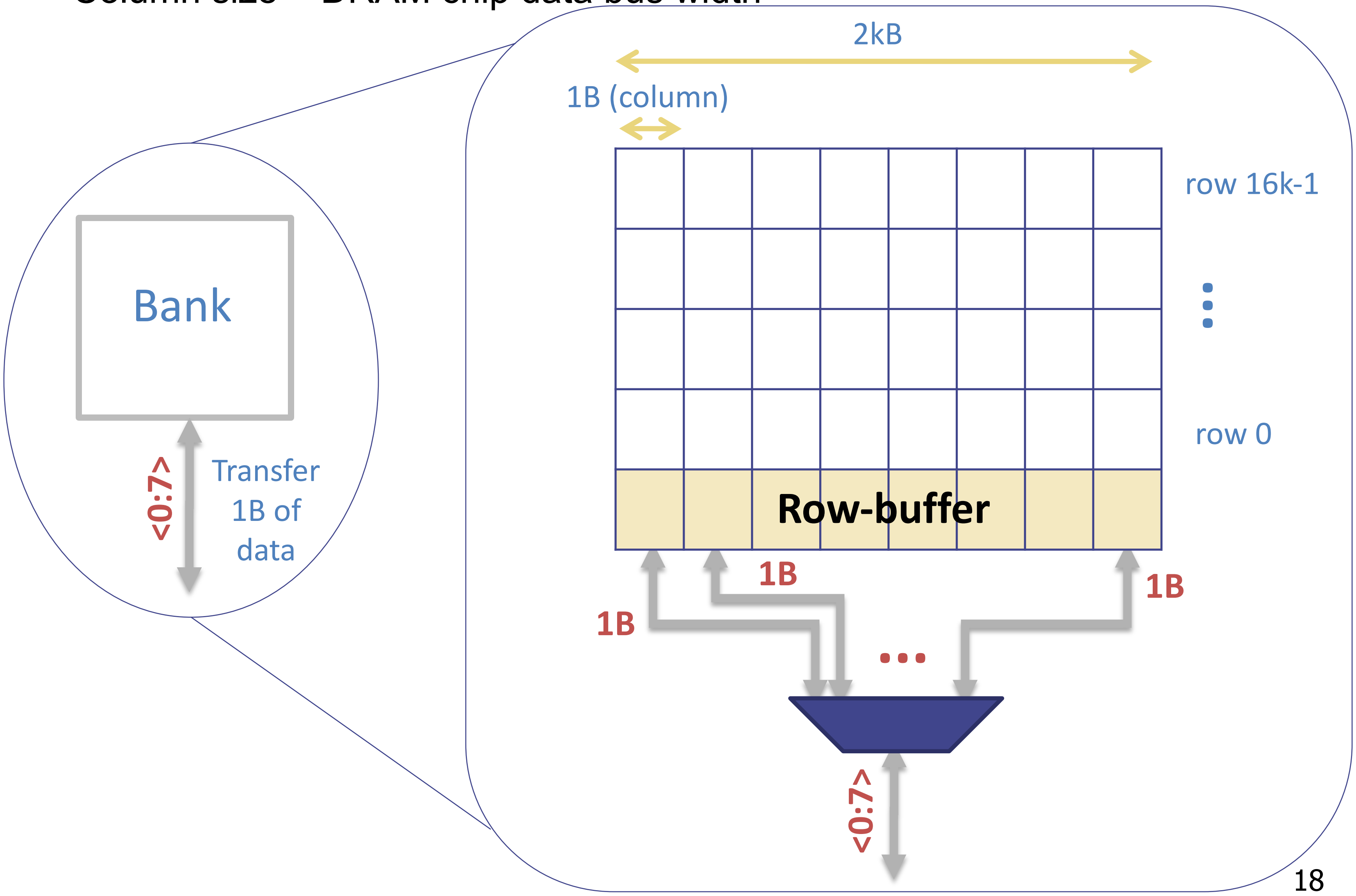
?

Review: Test yourself



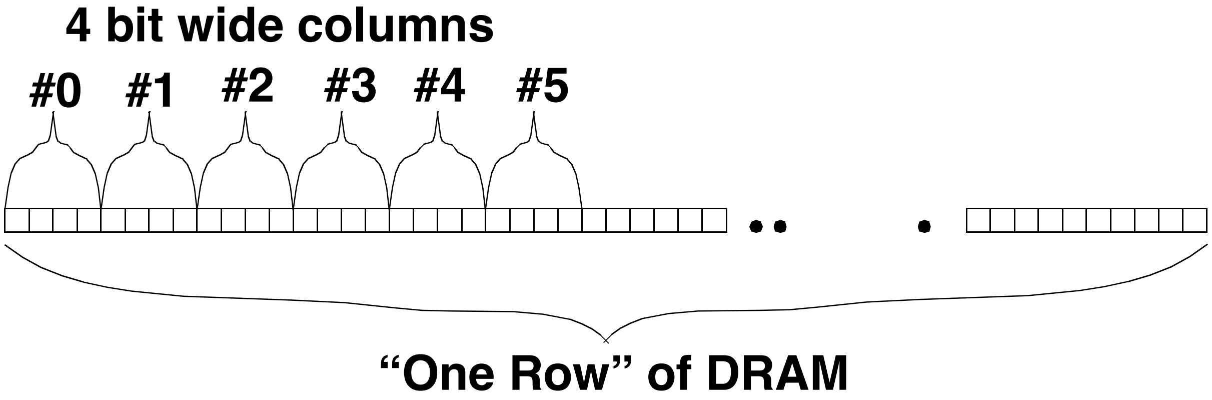
Review: Rows and columns

Column size = DRAM chip data bus width



Review: Burst length

- Column: Smallest addressable quantity of DRAM chip
  - 1 column: Column size == DRAM chip data bus width (4, 8,16, 32)
    - (i.e., 4 bits, 8 bits, 16 bits, 32 bits)
  - Get "n" columns per access. n = (1, 2, 4, 8)
    - "Burst length" == n

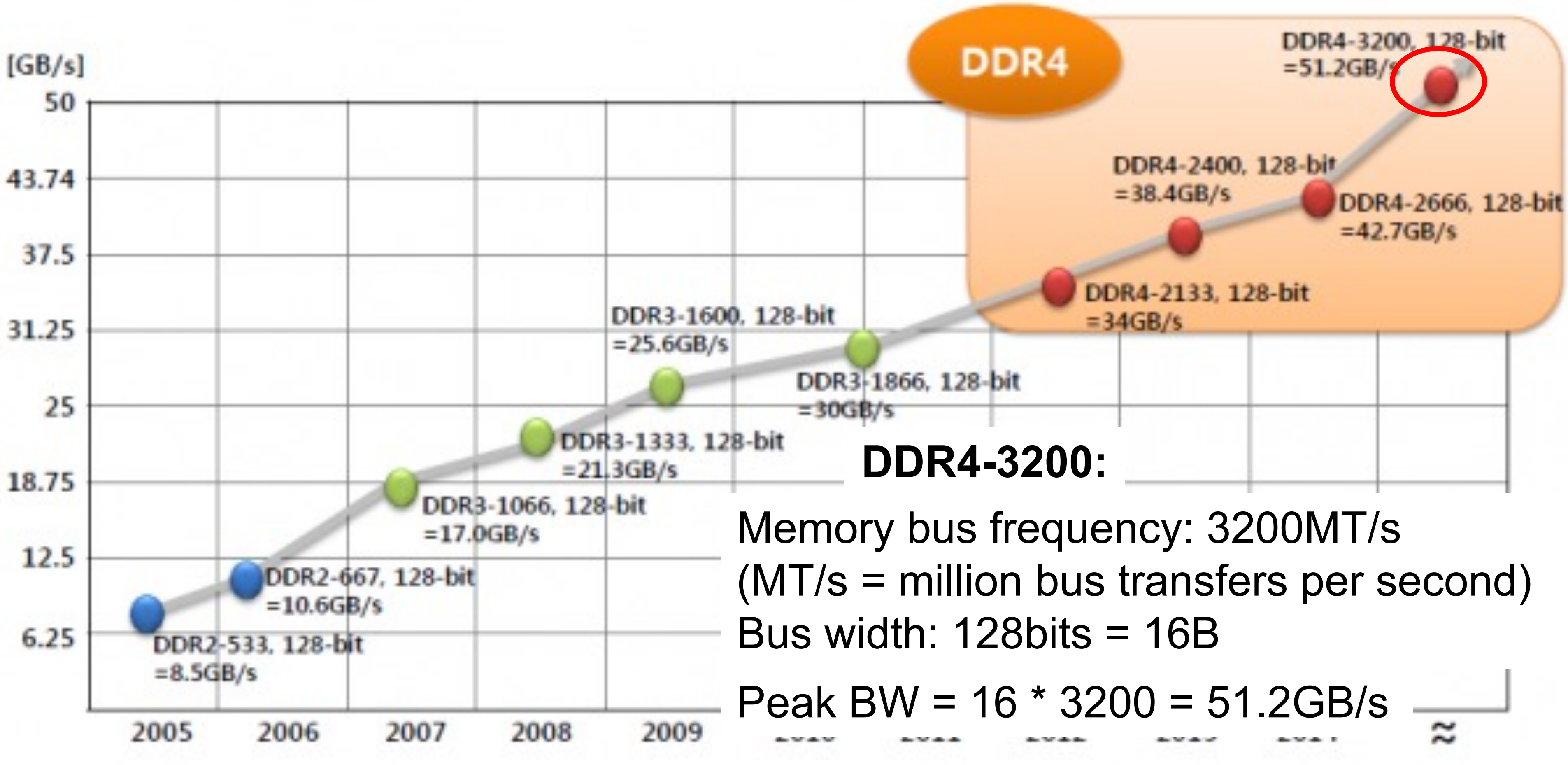


Review: Memory bandwidth

- Memory performance: Latency -- we already know
- What about memory throughput? → called "memory bandwidth"
  - How many bytes main memory can send per second?
  - Unit: GB/s or MB/s
  - Read bandwidth & write bandwidth
- "Peak memory bandwidth"
  - The maximum bandwidth a main memory device can achieve
  - This is what vendors like to advertise about

Review: Peak memory bandwidth calculation

Peak memory bandwidth = memory bus width * memory bus frequency



DDR4-3200:

Memory bus frequency: 3200MT/s
(MT/s = million bus transfers per second)
Bus width: 128bits = 16B

Peak BW = 16 * 3200 = 51.2GB/s

Review: Page Mode DRAM

- [Known] A DRAM bank is a 2D array of cells:
  - rows x columns
- A "DRAM row" is also called a "DRAM page"
- [Known] "Sense amplifiers" also called "row buffer"

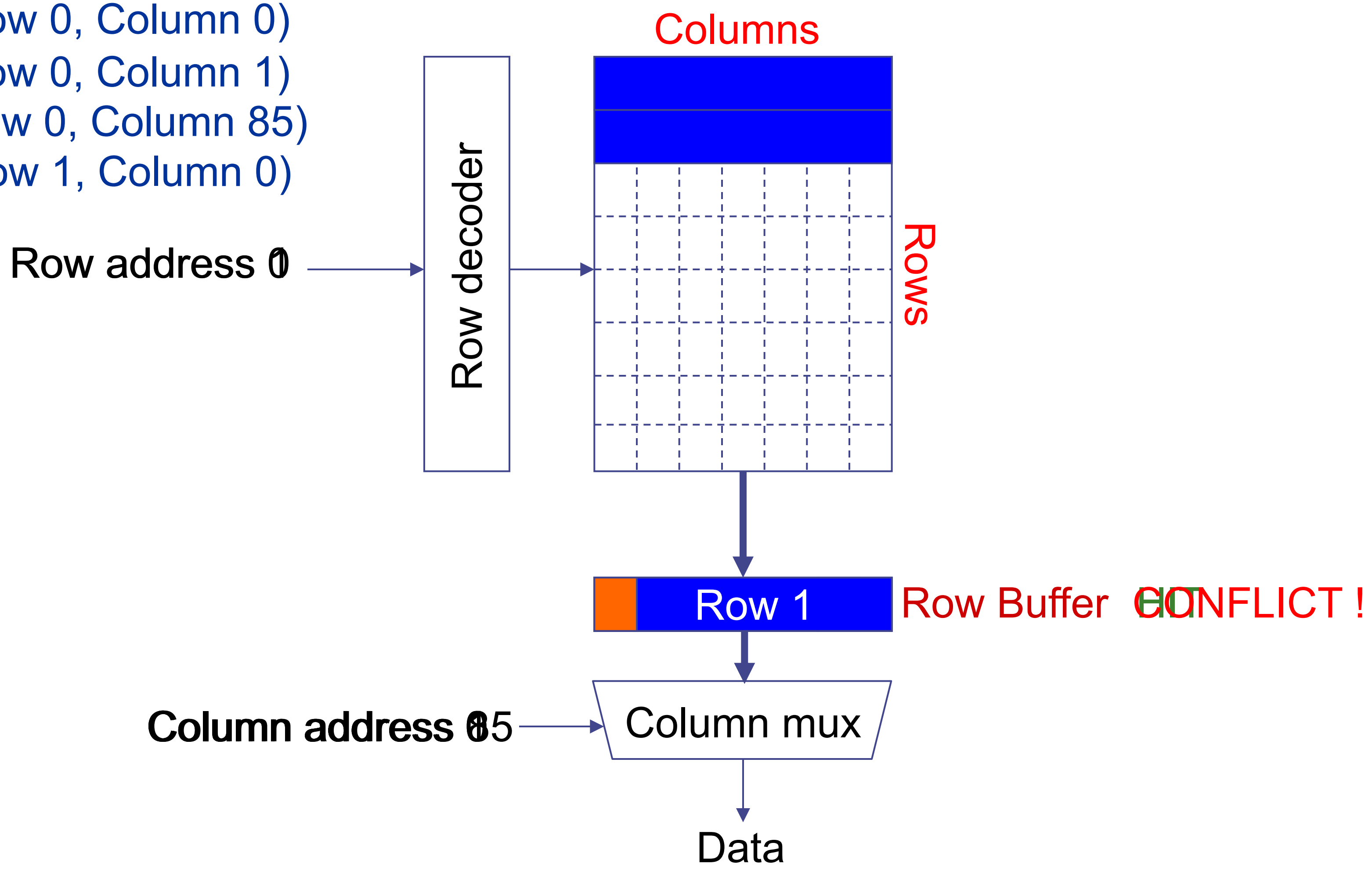
- **A closed row vs. an open row**
  (analogy: write-through vs. write-back cache)
  - A closed row: data are not buffered in the row buffer after each access (similar to "write-through")
  - An open row: data are buffered in the row buffer after access (similar to "write-back")
  - Can only have one open row at a time ← only one row buffer in a bank

Review: Open row vs. closed row

- Access to an "open row" – data in the row buffer
  - Read/write command reads/writes column in the row buffer
- Access to a "closed row" – data not in the row buffer
  - 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

DRAM Bank Operation (open row policy)

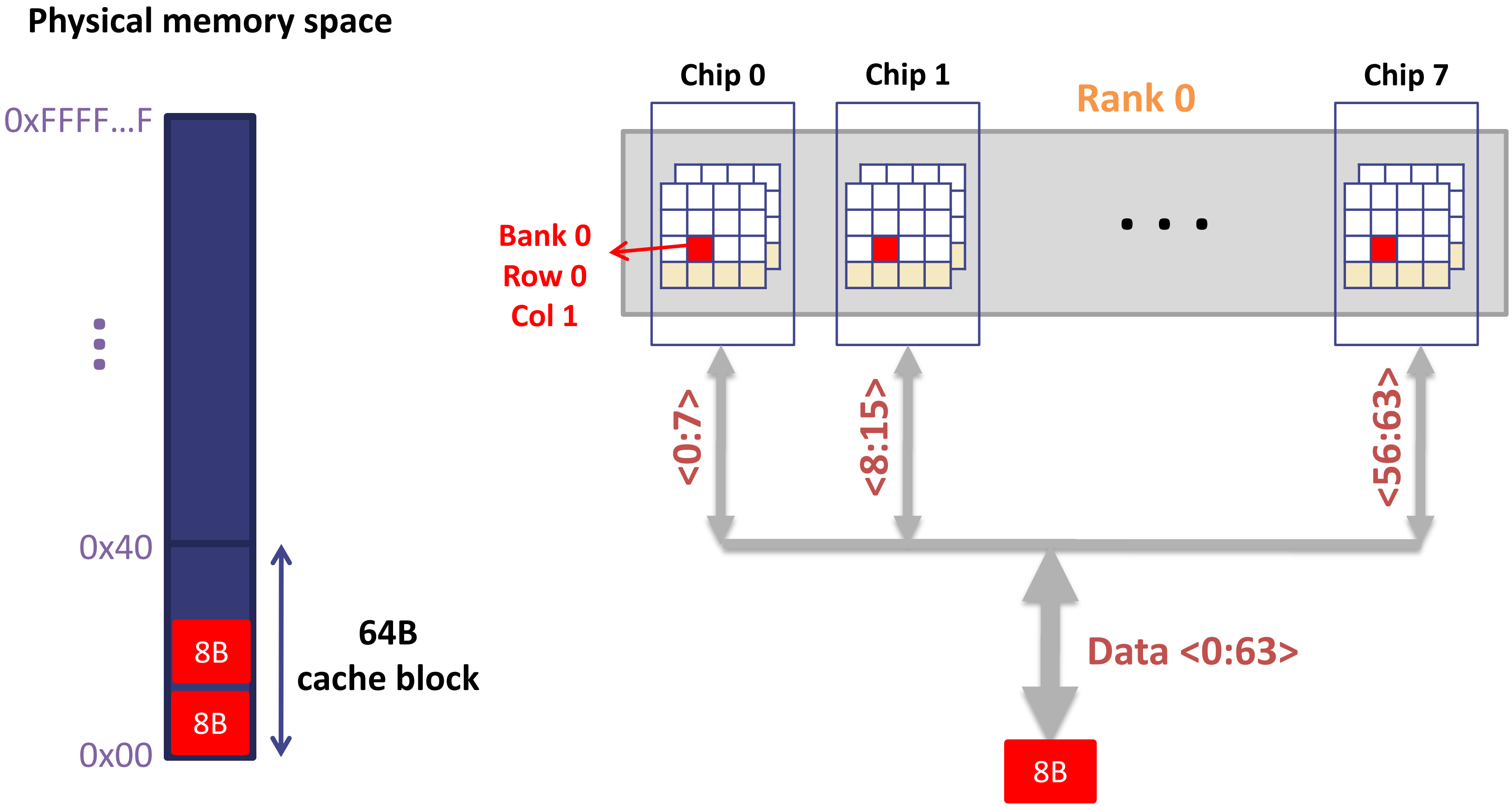
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



Row address 0

Column address 85

Review: Transferring a cache block

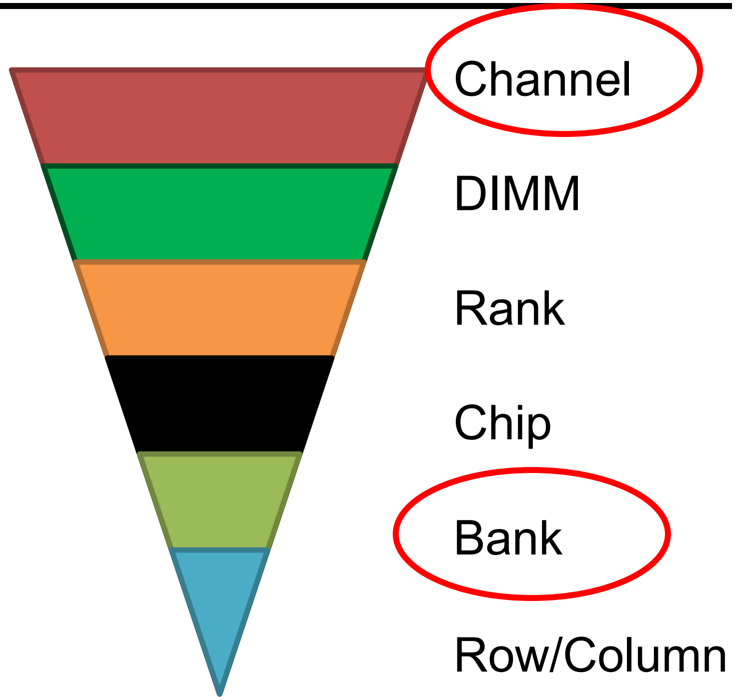


A 64B cache block takes 8 I/O cycles to transfer.

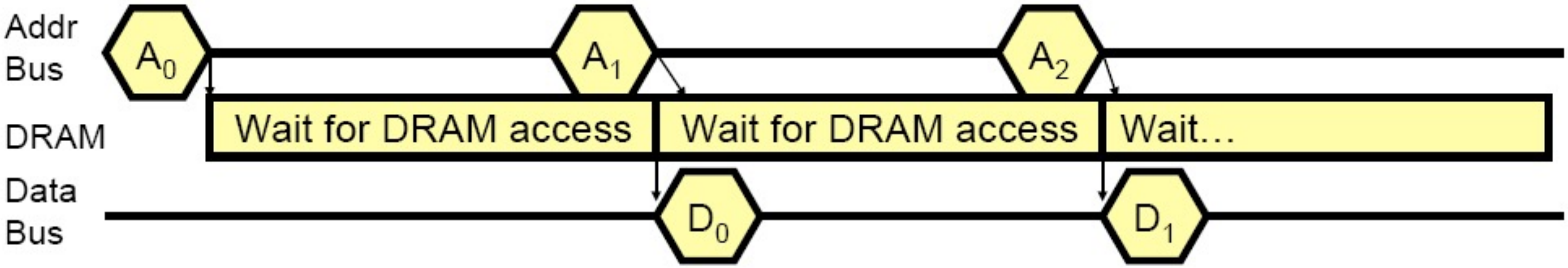
During the process, 8 columns are read sequentially.

Review: Test yourself

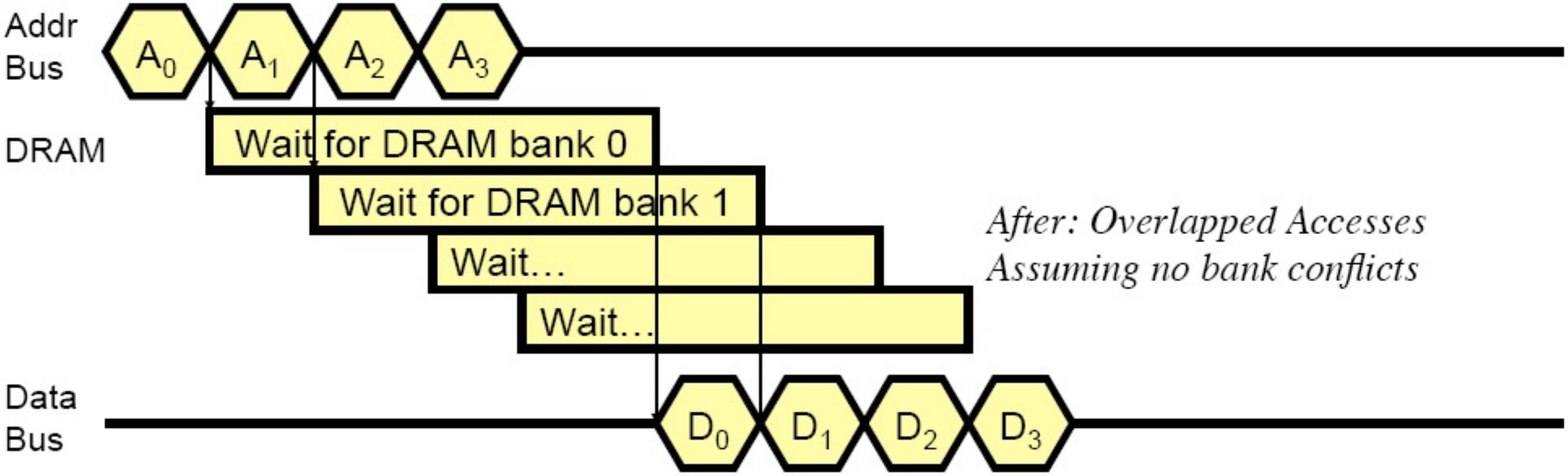
- List the methods you’ve learned for improving parallelism of memory access
  - Multiple banks
  - Multiple channels



Review: How Multiple Banks Help

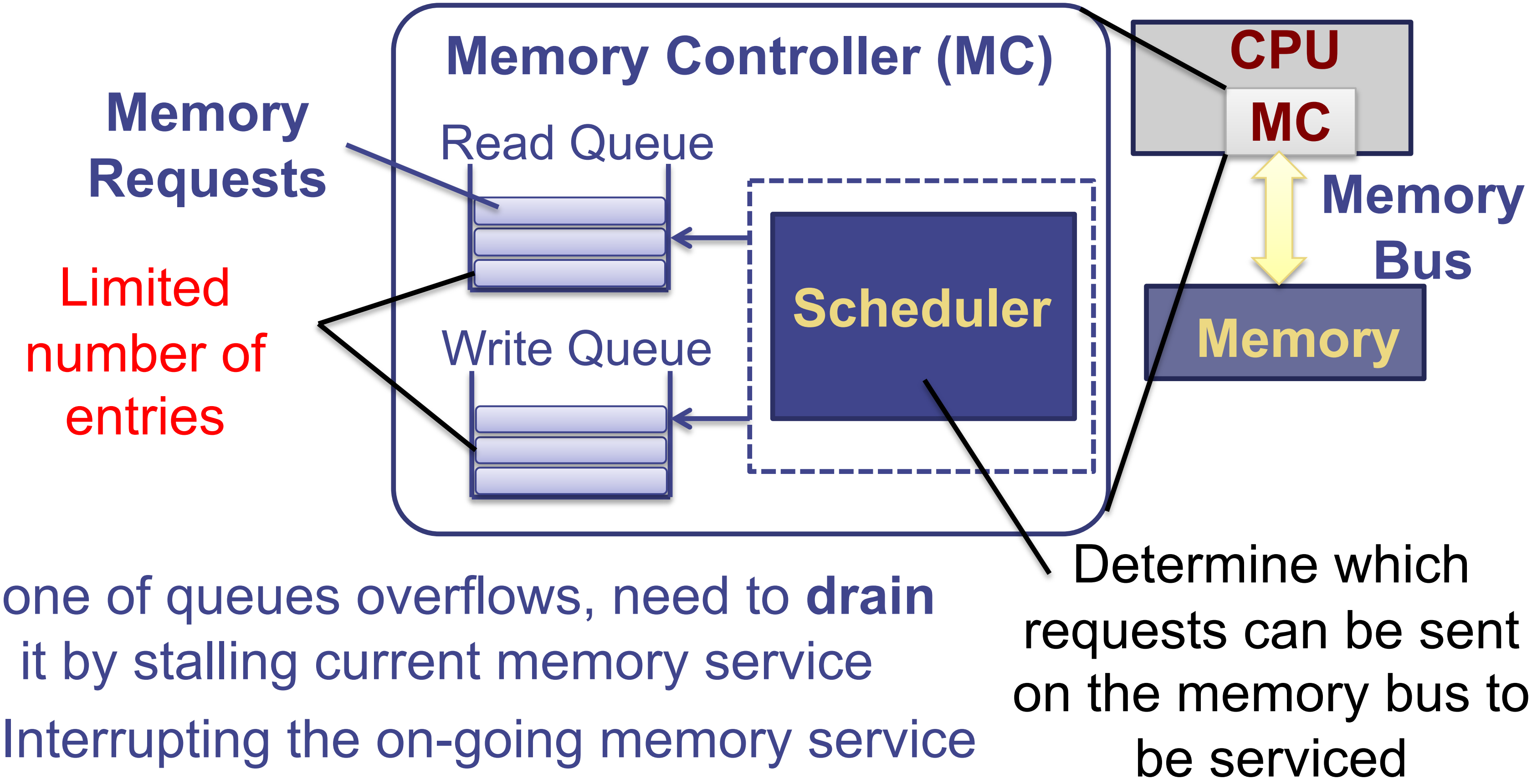


Before: No Overlapping
Assuming accesses to different DRAM rows



After: Overlapped Accesses
Assuming no bank conflicts

Memory controller



Memory Requests

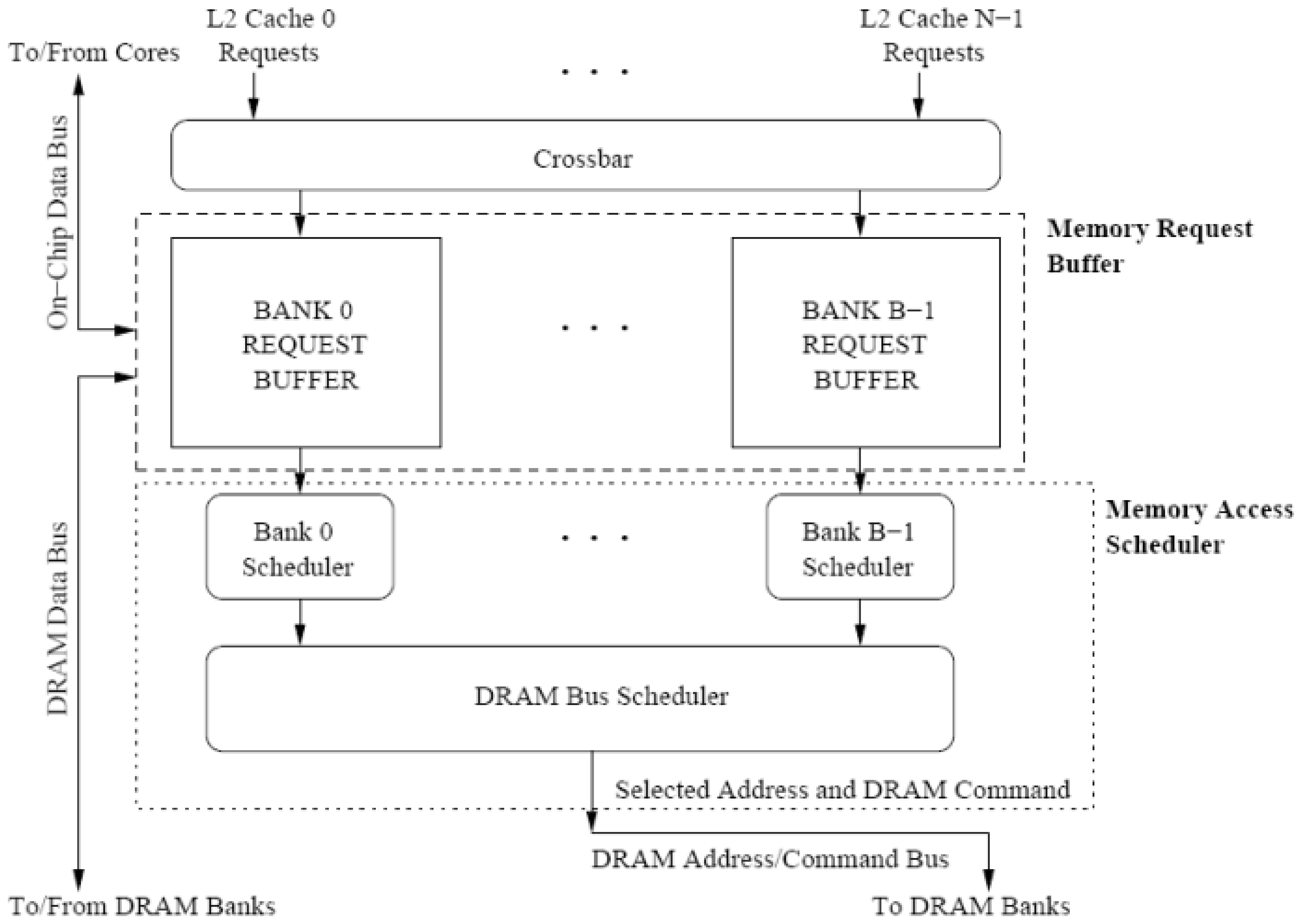
Limited number of entries

Determine which requests can be sent on the memory bus to be serviced

If one of queues overflows, need to **drain** it by stalling current memory service

- Interrupting the on-going memory service
  - Bus turnaround overhead

A Modern DRAM Controller



Scheduling Policy for Single-Core Systems

- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer

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

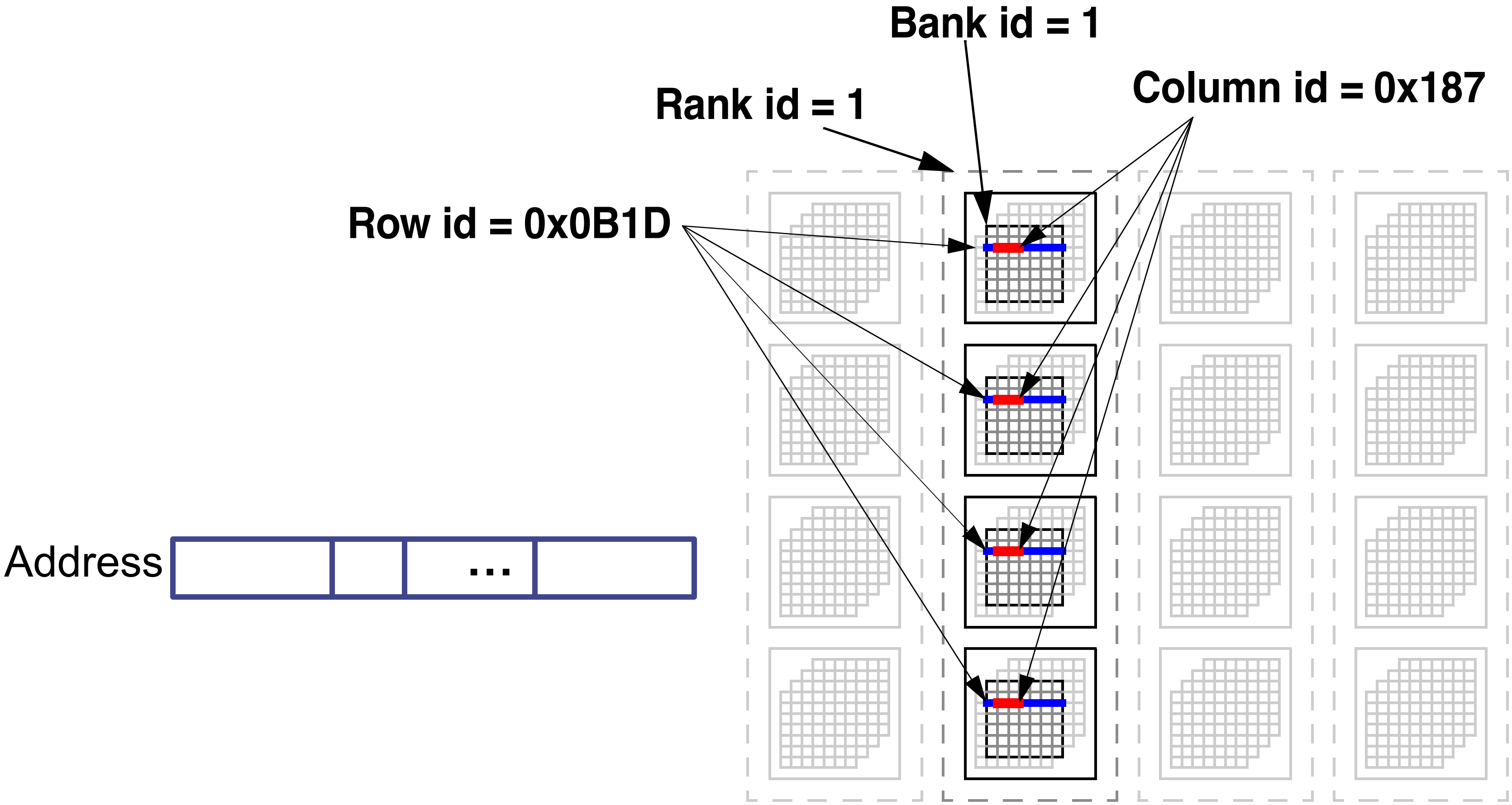
  1. Row-hit first
  2. Oldest first

  Goal 1: Maximize row buffer hit rate → maximize DRAM throughput

  Goal 2: Prioritize older requests → ensure forward progress

- Is this a good policy in a multi-core system?
  - Mutlu and Moscibroda, Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors, MICRO'07.
  - Mutul and Moscibroda, Parallelism-Aware Batch Scheduling:Enhancing both Performance and Fairness of Shared DRAM Systems, ISCA'08.

How do we encode the address: Address mapping



Bank id = 1

Rank id = 1

Column id = 0x187

Row id = 0x0B1D

Address

Address mapping

- Single-channel system with 8-byte memory bus
  - 2GB memory, 8 banks, 16K rows & 2048 columns per bank, 64B cache blocks

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

- Row interleaving
  - Ensure consecutive rows of memory mapped to consecutive (i.e., different) banks

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

  - Accesses to consecutive rows serviced in a pipelined manner

- Cache block interleaving
  - Ensure consecutive cache block addresses mapped to consecutive banks

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

  - Accesses to consecutive cache blocks serviced in pipelined manner

Example/Exercise: Address mapping

- Single-channel, 8-byte memory bus, 2GB memory, 8 banks, 16K rows & 2048 columns (i.e., 16K * 2KB) per bank, 64B cache blocks
  - How many bits needed in address? Answer: 31 bits
  - How many bits needed in address to identify each byte in one bus transfer? Answer: 3 bits
  - How many bits needed in address to identify each bus transfer for transferring one row of data? Answer: 11 bits
  - How many bits needed in address to identify each row in one bank? Answer: 14 bits
  - How many bits needed in address to identify each bank? Answer: 3 bits
- A program has the following physical address access sequence:
  - 0x0000 0001 *Bank 0, Row 0, Col. 0, Byte 1*
  - 0x0000 4001 *Bank 0, Row 1, Col. 0, Byte 1*
  - 0x0000 8002 *Bank 0, Row 2, Col. 0, Byte 2*

Row Conflicts

No address mapping

Bank ID	Row ID	Col. ID	Byte in one bus transfer
3 bits	14 bits	11 bits	3 bits

Example/Exercise: Address mapping

- Single-channel, 8-byte memory bus, 2GB memory, 8 banks, 16K rows & 2048 columns (i.e., 16K * 2KB) per bank, 64B cache blocks
  - How many bits needed in address? Answer: 31 bits
  - How many bits needed in address to identify each byte in one bus transfer? Answer: 3 bits
  - How many bits needed in address to identify each bus transfer for transferring one row of data? Answer: 11 bits
  - How many bits needed in address to identify each row in one bank? Answer: 14 bits
  - How many bits needed in address to identify each bank? Answer: 3 bits
- A program has the following physical address access sequence:
  - 0x0000 0001 *Row 0,* *Bank 0,* *Col. 0,* *Byte 1*
  - 0x0000 4001 *Row 0,* *Bank 1,* *Col. 0,* *Byte 1*
  - 0x0000 8002 *Row 0,* *Bank 2,* *Col. 0,* *Byte 2*
  - 0x0A00 800A *Row 0x500,* *Bank 2,* *Col. 1,* *Byte 2*

No row conflicts!

No address mapping

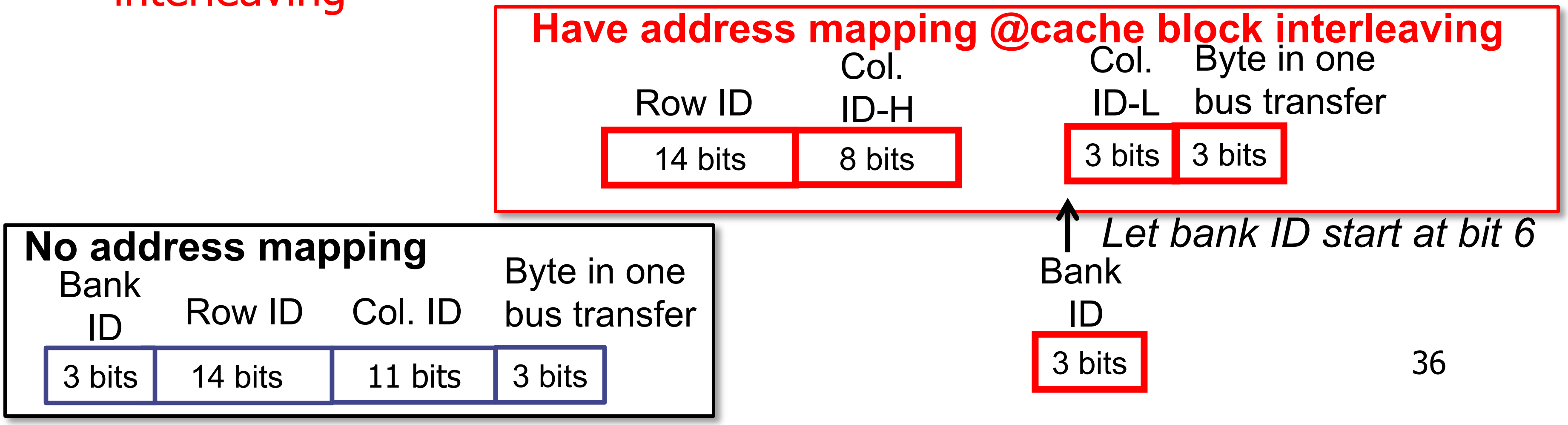
Bank ID	Row ID	Col. ID	Byte in one bus transfer
3 bits	14 bits	11 bits	3 bits

Have address mapping @row interleaving

Row ID	Bank ID	Col. ID	Byte in one bus transfer
14 bits	3 bits	11 bits	3 bits

Exercise: Address mapping

- Single-channel, 8-byte memory bus, 2GB memory, 8 banks, 16K rows & 2048 columns (i.e., 16K * 2KB) per bank, 64B cache blocks
  - How many bits needed in address? Answer: 31 bits
  - How many bits needed in address to identify each byte in one bus transfer? Answer: 3 bits
  - How many bits needed in address to identify each bus transfer for transferring one row of data? Answer: 11 bits
  - How many bits needed in address to identify each row in one bank? Answer: 14 bits
  - How many bits needed in address to identify each bank? Answer: 3 bits
- A program has the following physical address access sequence:
  - Come up with an address sequence yourself to evaluate cache block interleaving



Have address mapping @cache block interleaving

Row ID	Col. ID-H	Col. ID-L	Byte in one bus transfer
14 bits	8 bits	3 bits	3 bits

Let bank ID start at bit 6

Bank ID: 3 bits

No address mapping

Bank ID	Row ID	Col. ID	Byte in one bus transfer
3 bits	14 bits	11 bits	3 bits