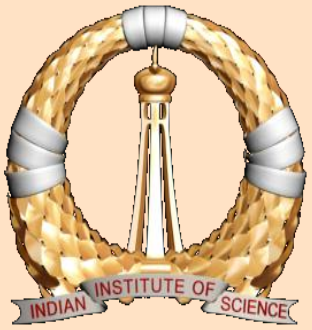
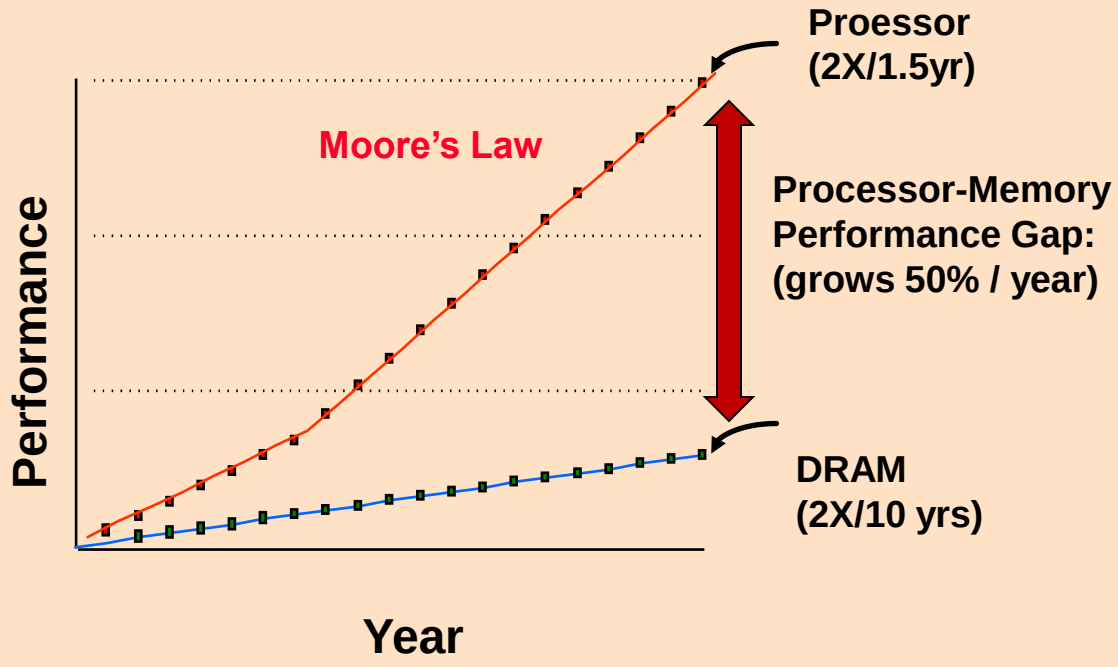
Understanding DRAM Architecture

R. Govindarajan
Computer Science & Automation
Supercomputer Edn. & Res. Centre
Indian Institute of Scinece, Bangalore
govind@iisc.ac.in

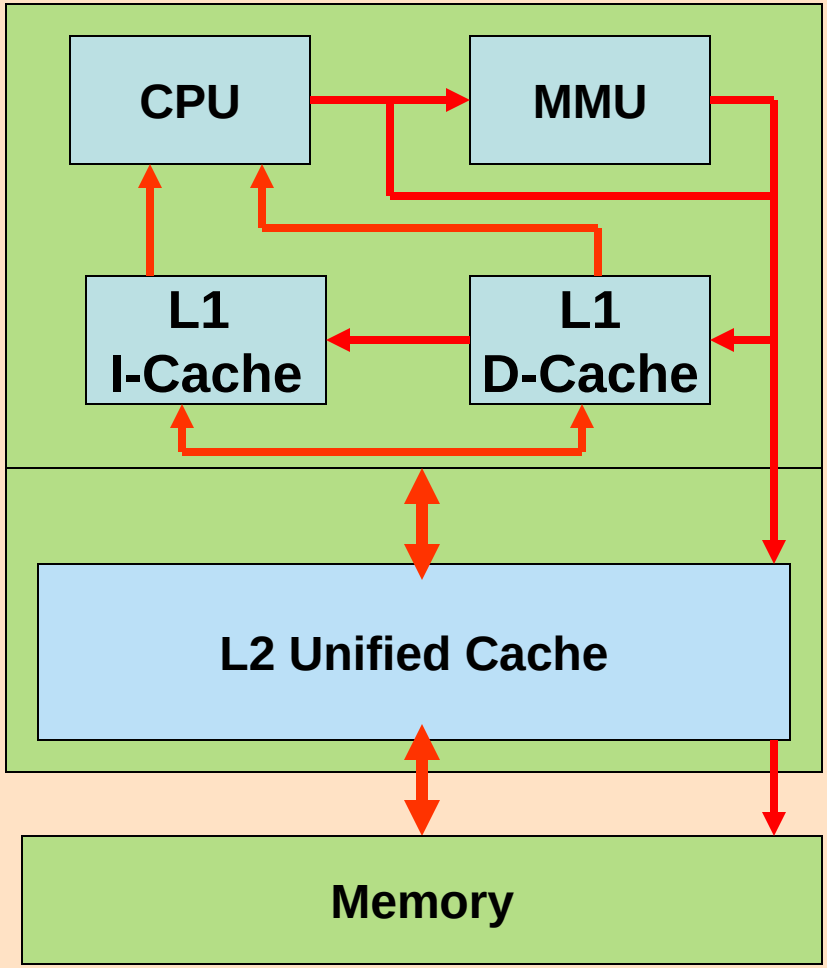


Why Study Memory System?

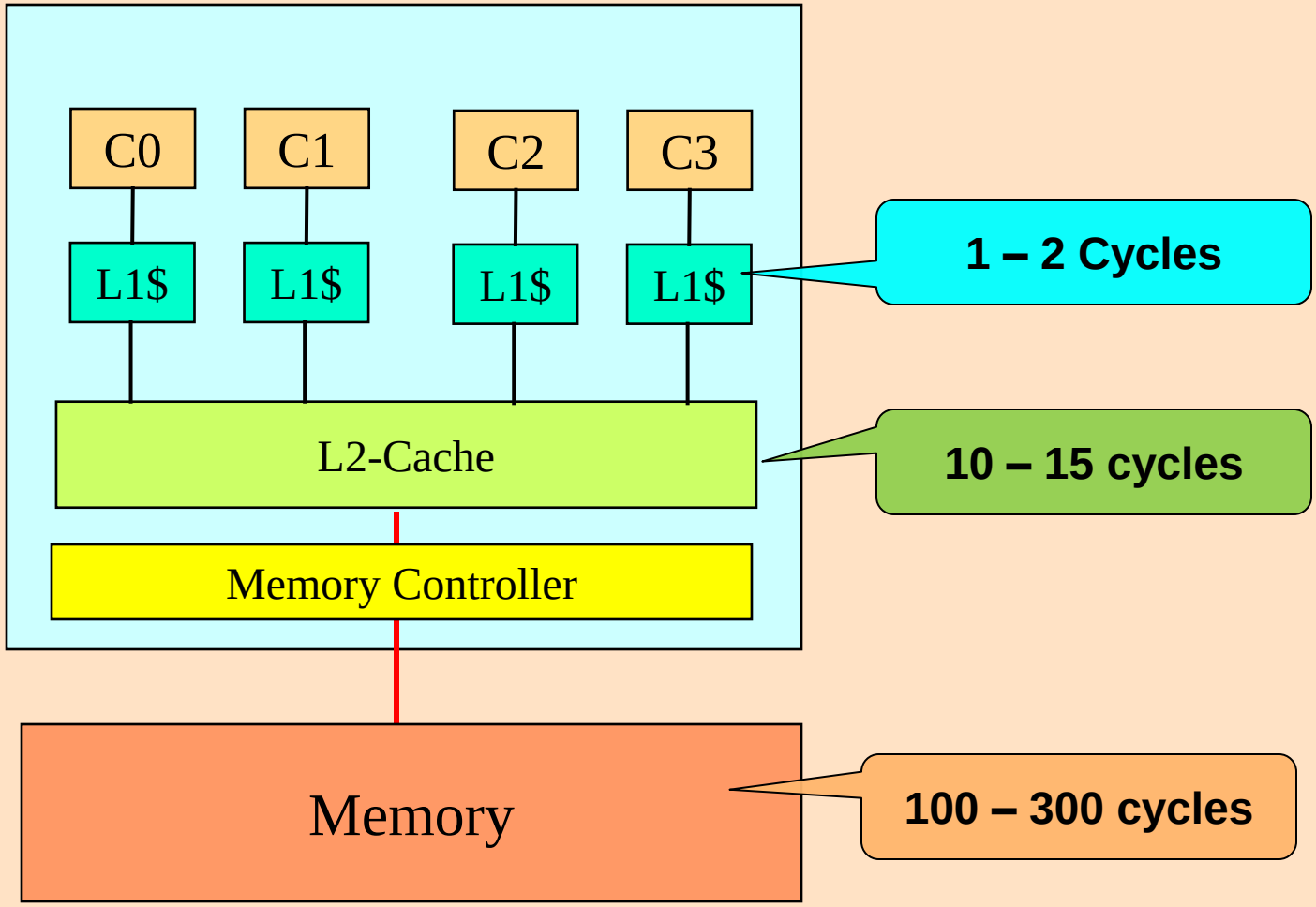


- Memory Wall [McKee'94]
  - CPU-Memory speed disparity
  - 100's of cycles for off-chip access

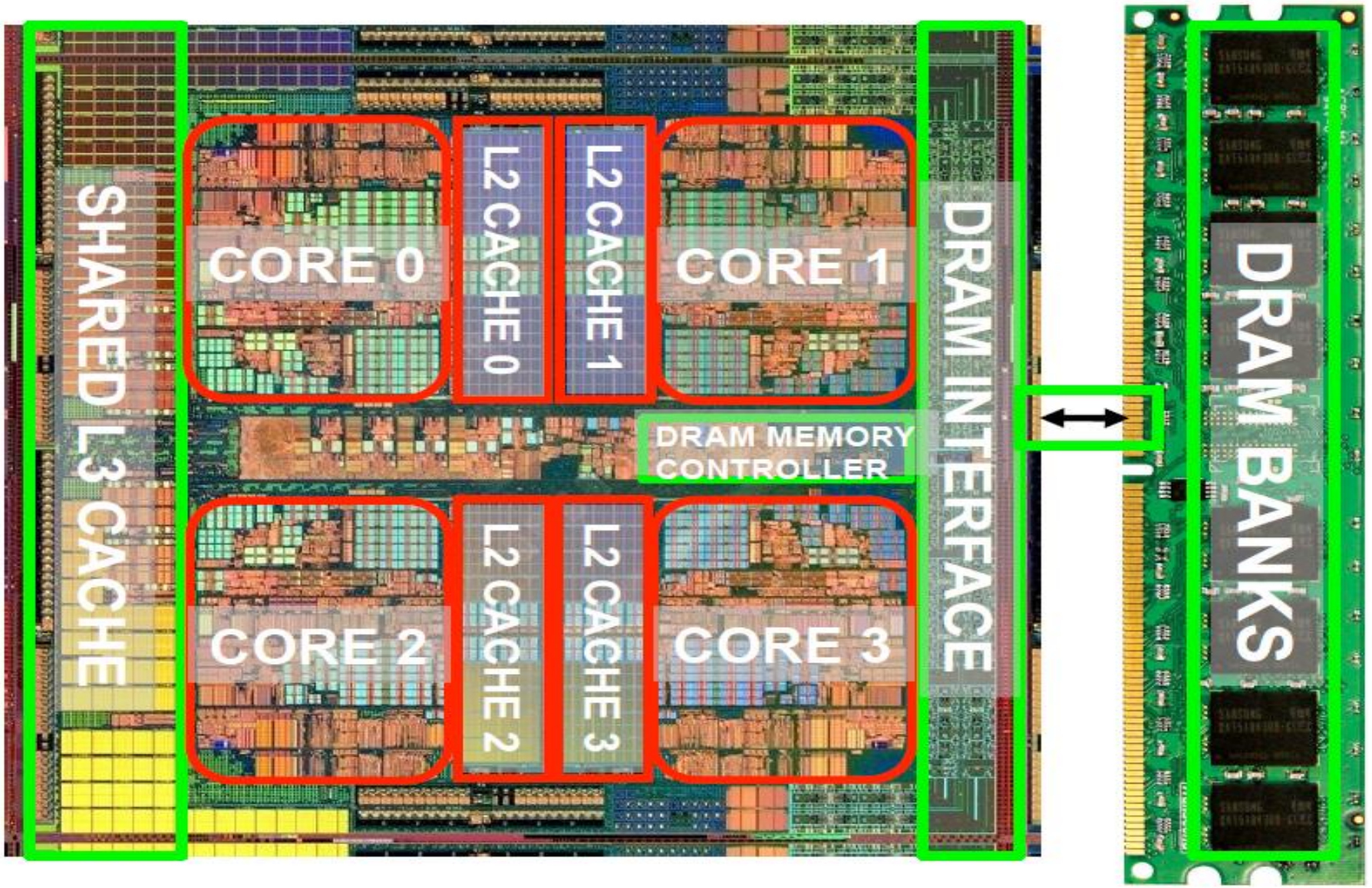
Memory Hierarchy : Recap



Memory Hierarchy in Multicore



Memory Hierarchy in Multicore



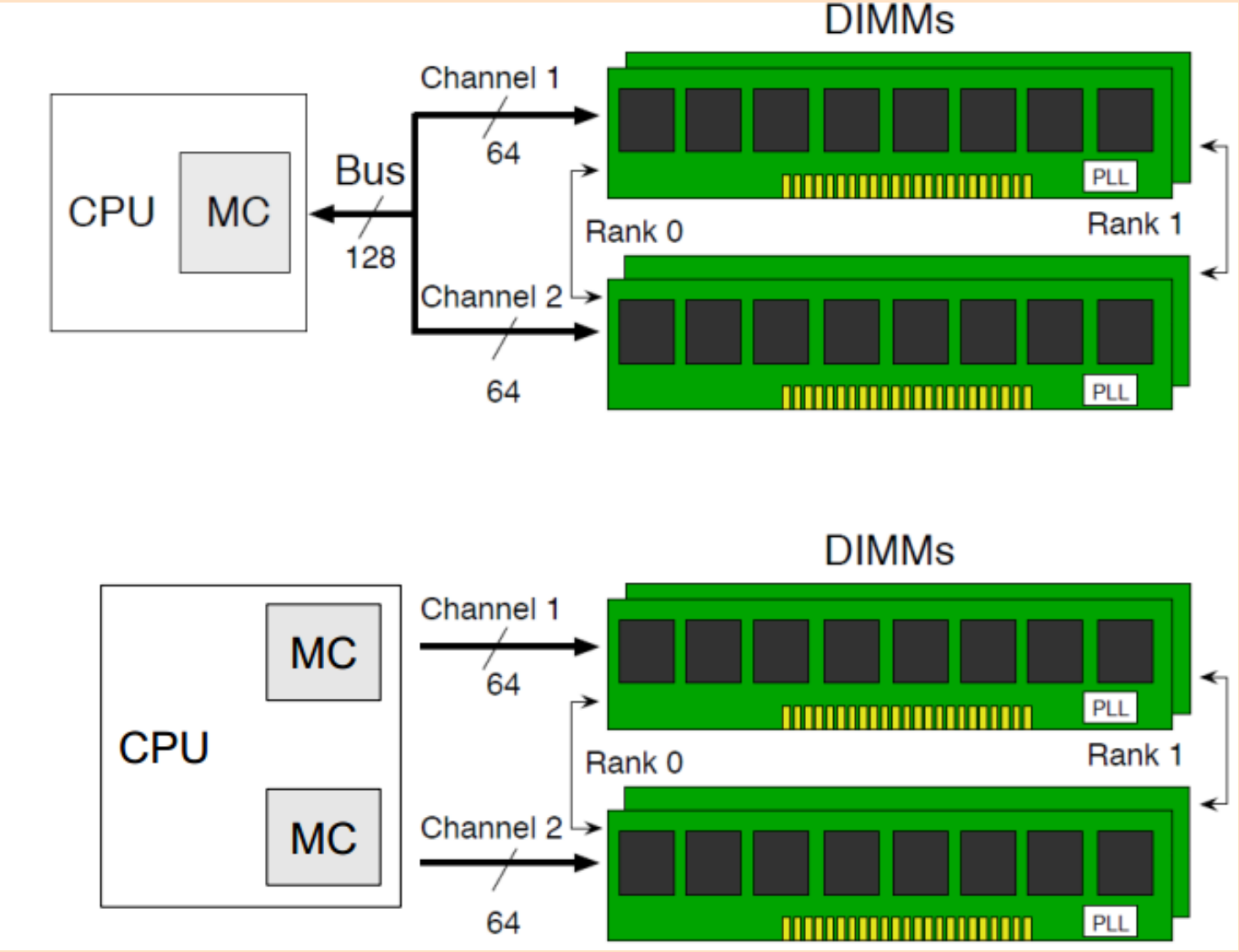
Memory Bandwidth Demand for Multicores

- Memory Wall [McKee'94]
  - CPU-Memory speed disparity
  - 100's of cycles for off-chip access
- Bandwidth Wall [ISCA'09]
  - More cores and limited off-chip bandwidth
  - Cores double every 18 months
  - Pincount grows only by 10%

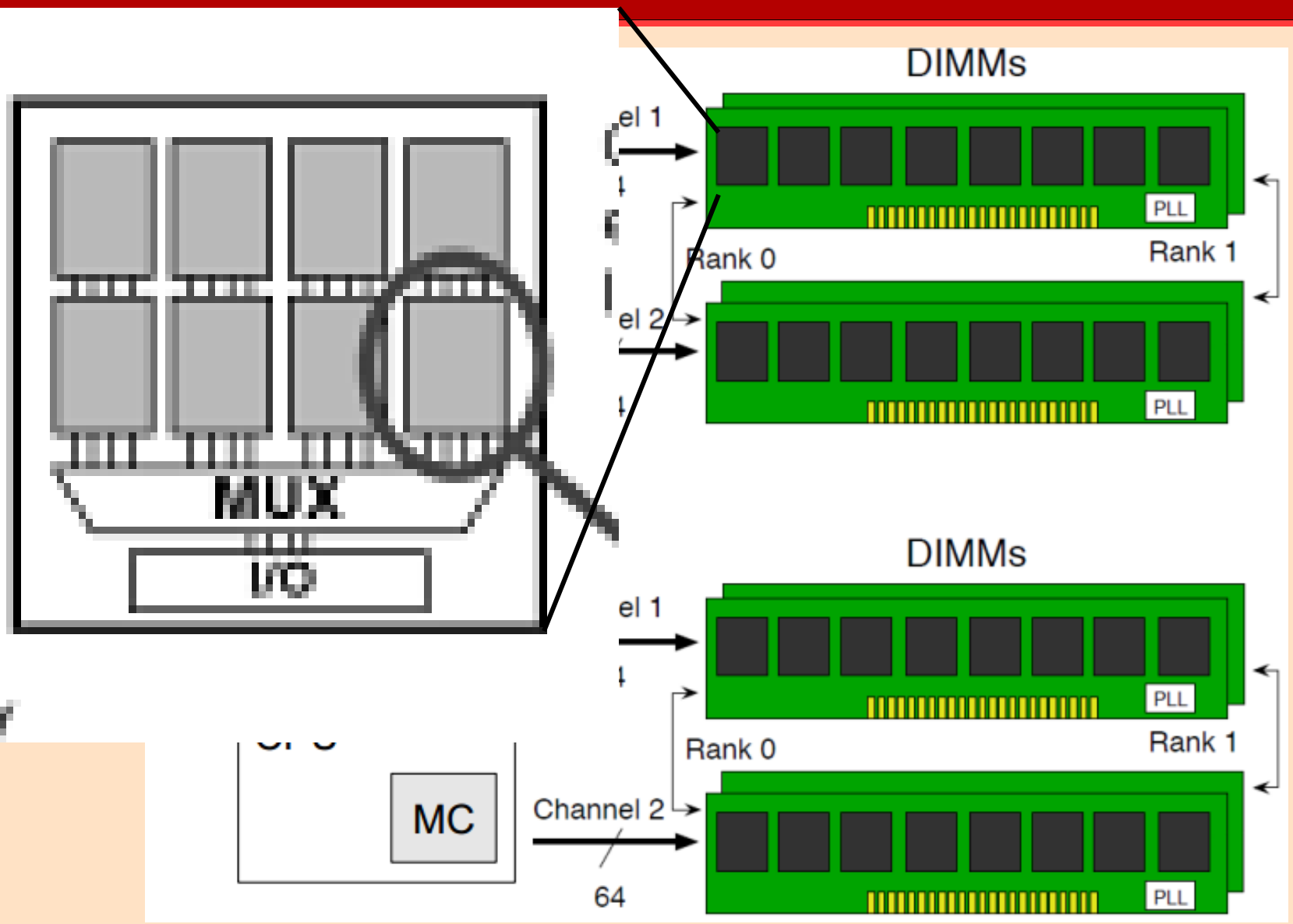
Off-chip accesses are expensive !

Memory System Performance is Critical

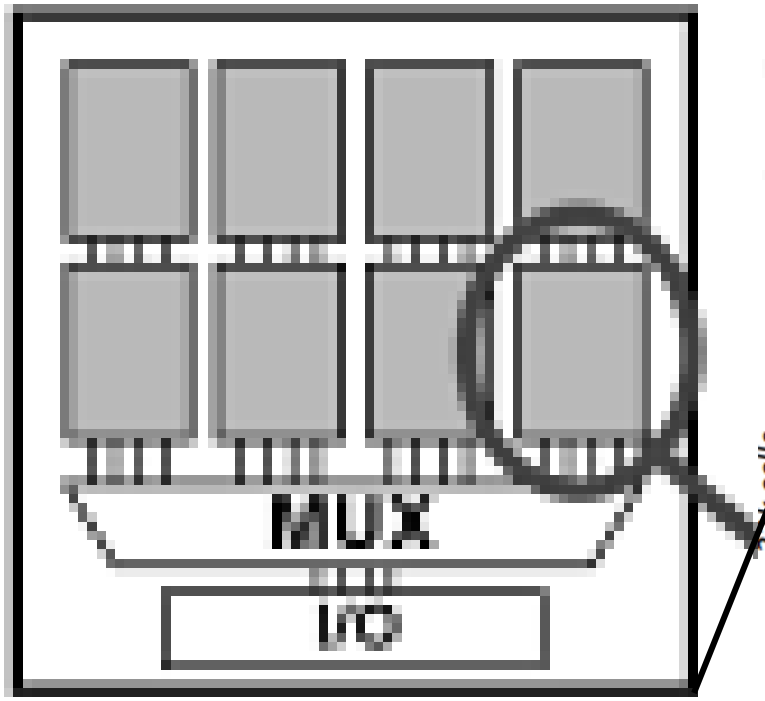
Big Picture of Memory

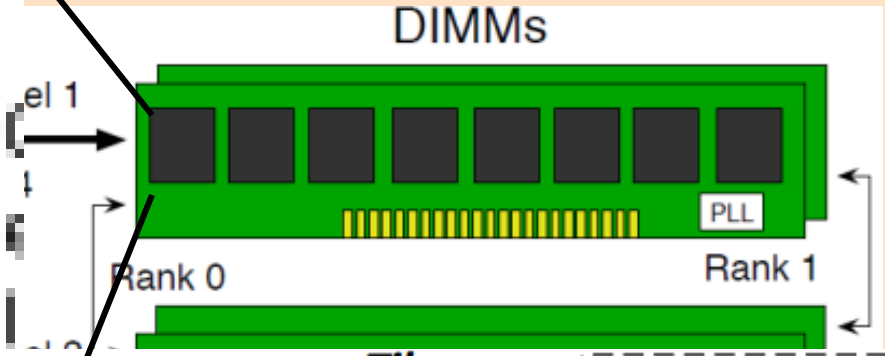


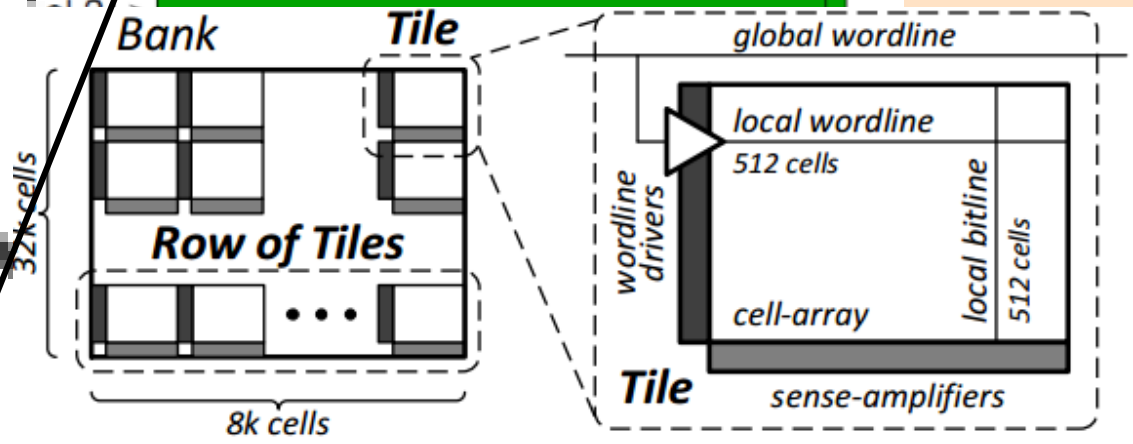
Big Picture of Memory



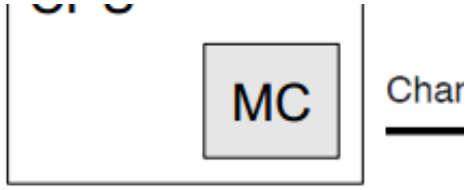
Big Picture of Memory

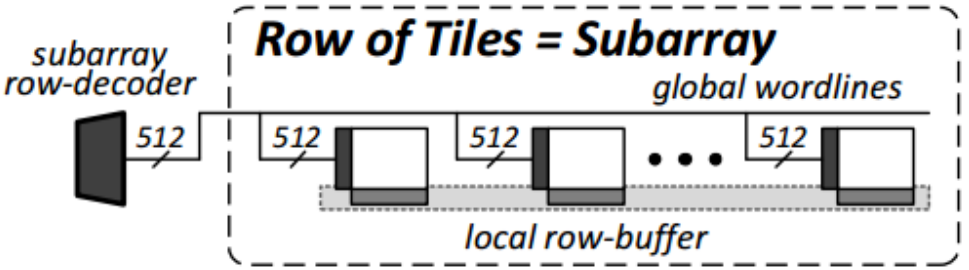






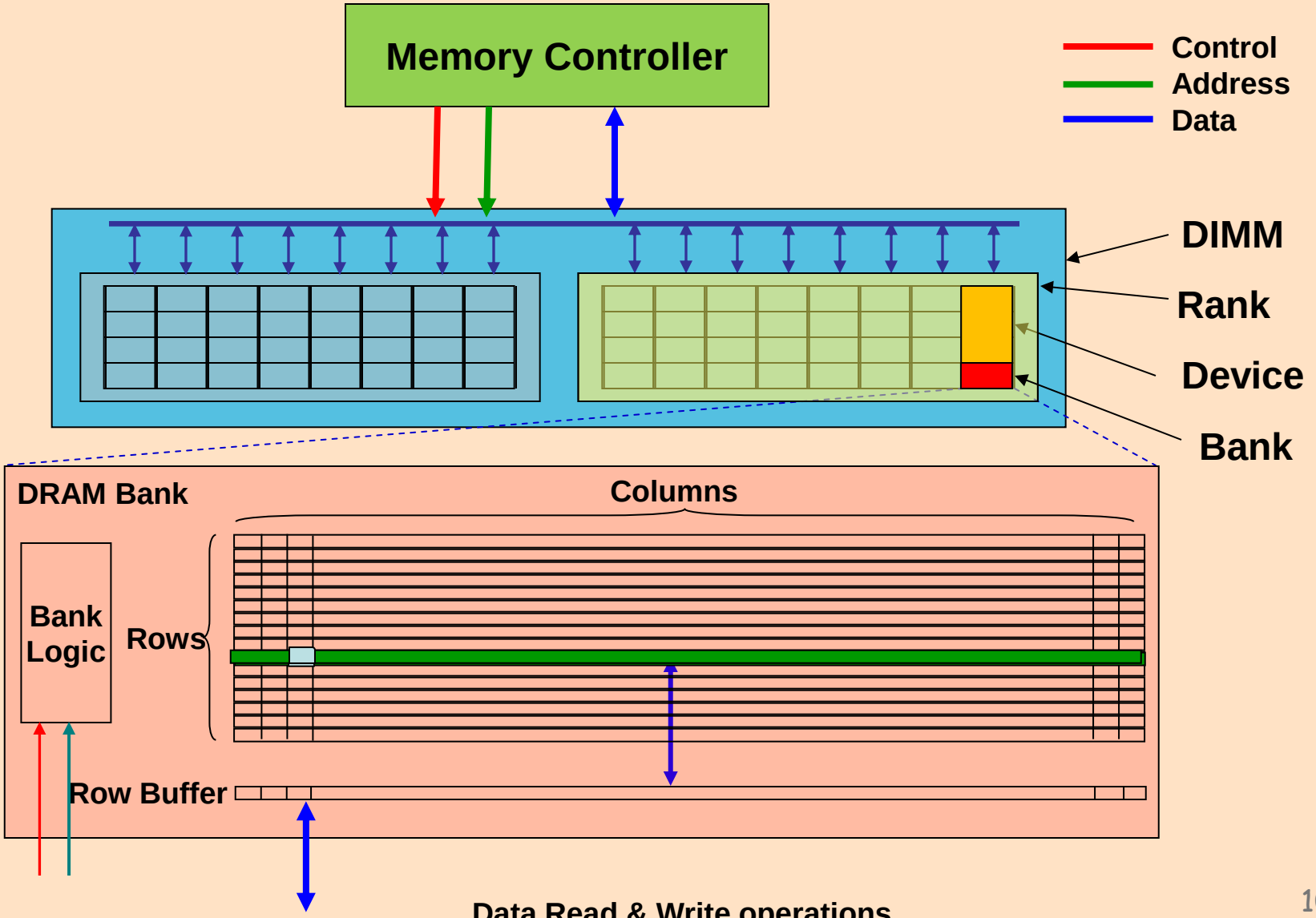
(a) A DRAM bank is divided into tiles.





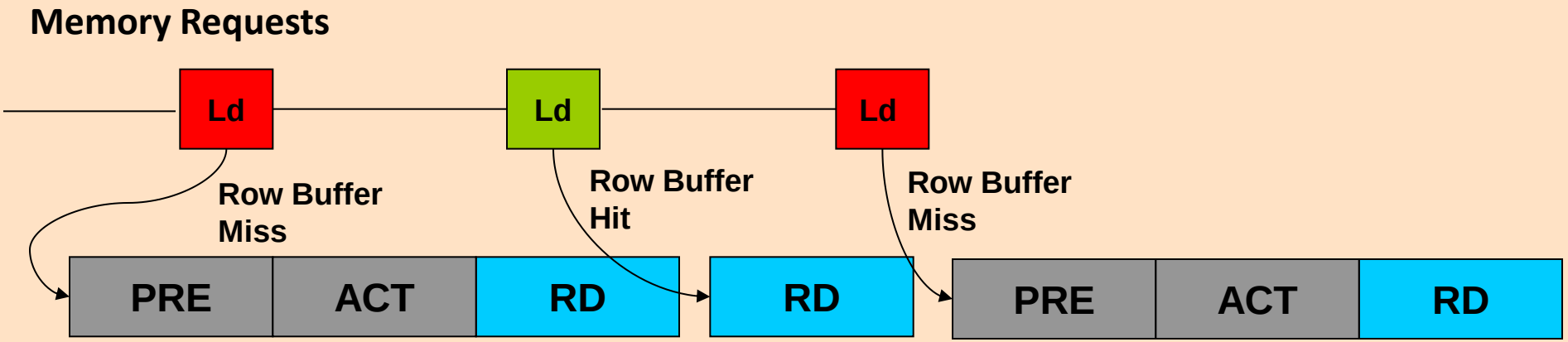
(b) Subarray: A row of tiles that operate in lockstep.

Overview of a DRAM Memory



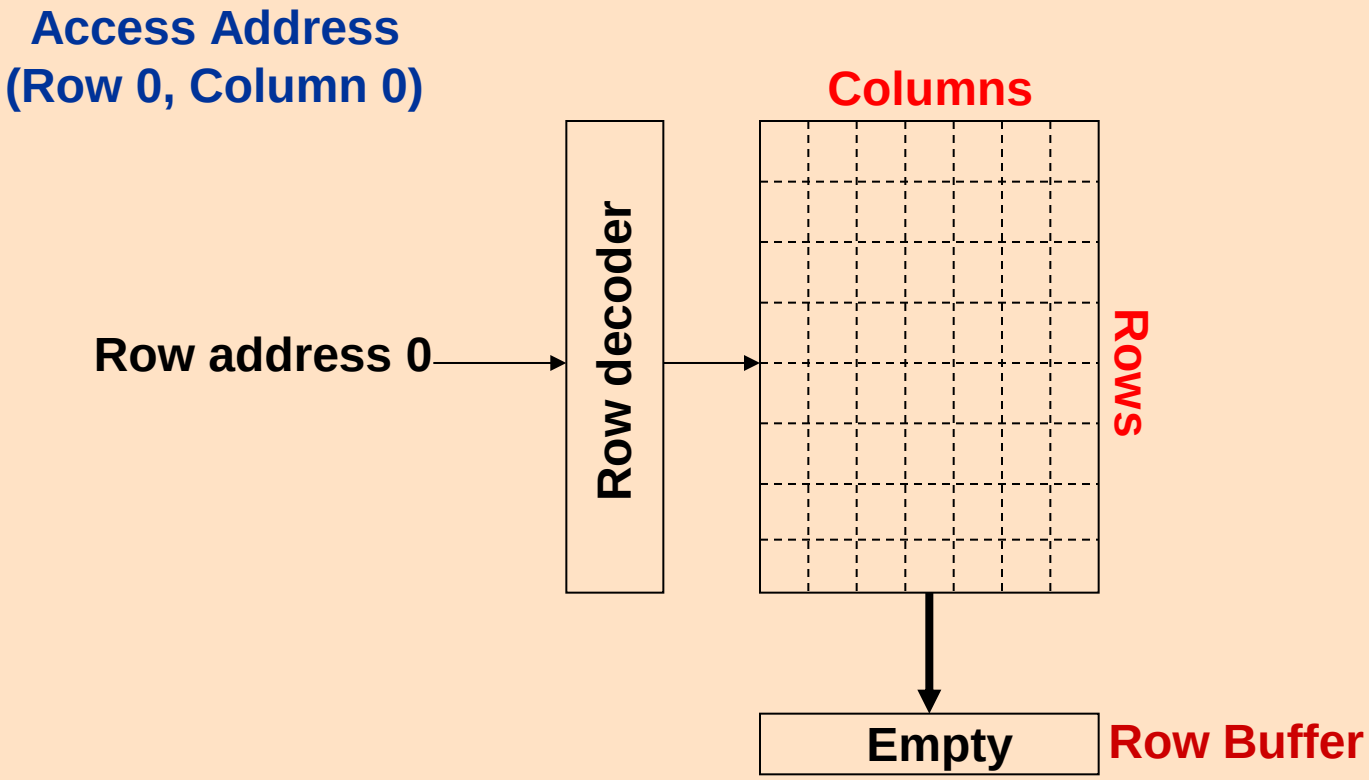
Basic DRAM Operations

- ACTIVATE → Bring data from DRAM core into the row-buffer
- READ/WRITE → Perform read/write operations on the contents in the row-buffer
- PRECHARGE → Store data back to DRAM core (ACTIVATE discharges capacitors), put cells back at neutral voltage



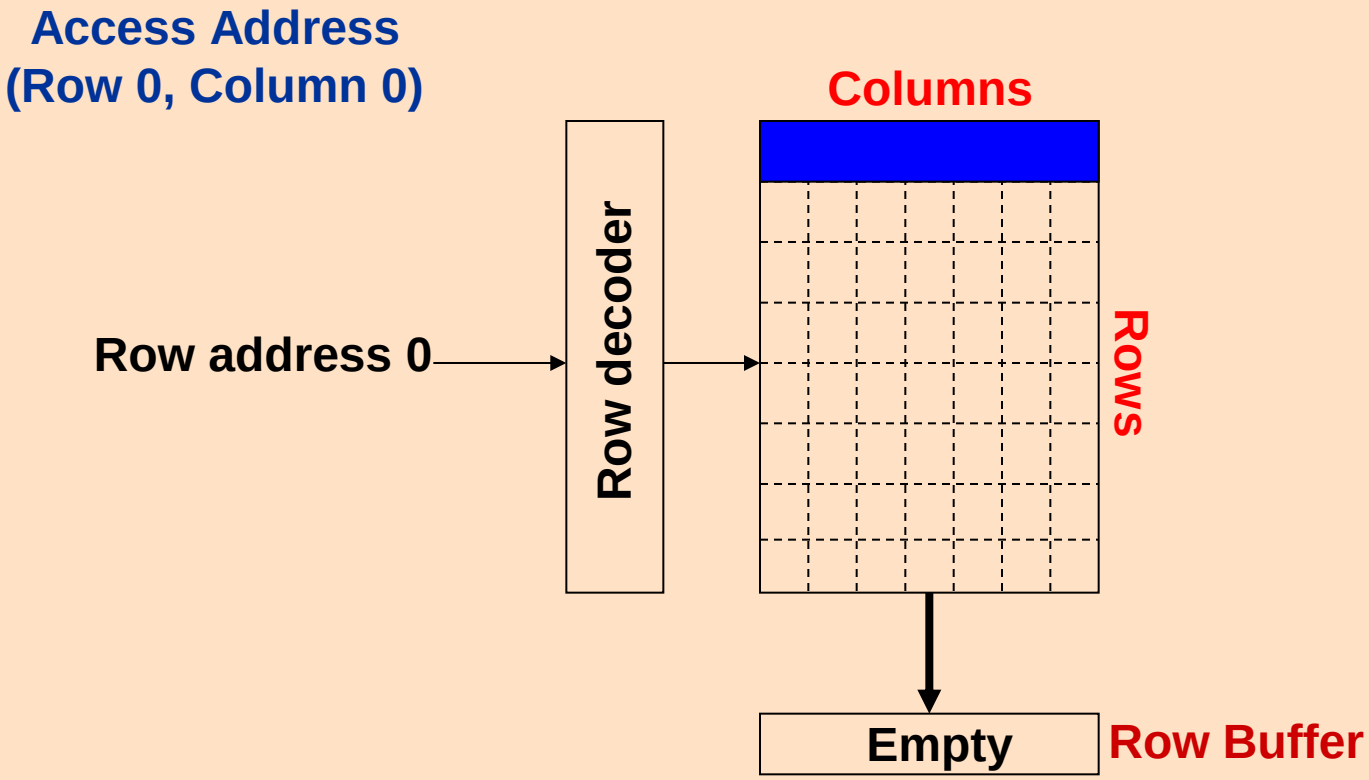
Row buffer hits are faster and consume less power

DRAM Bank Operation



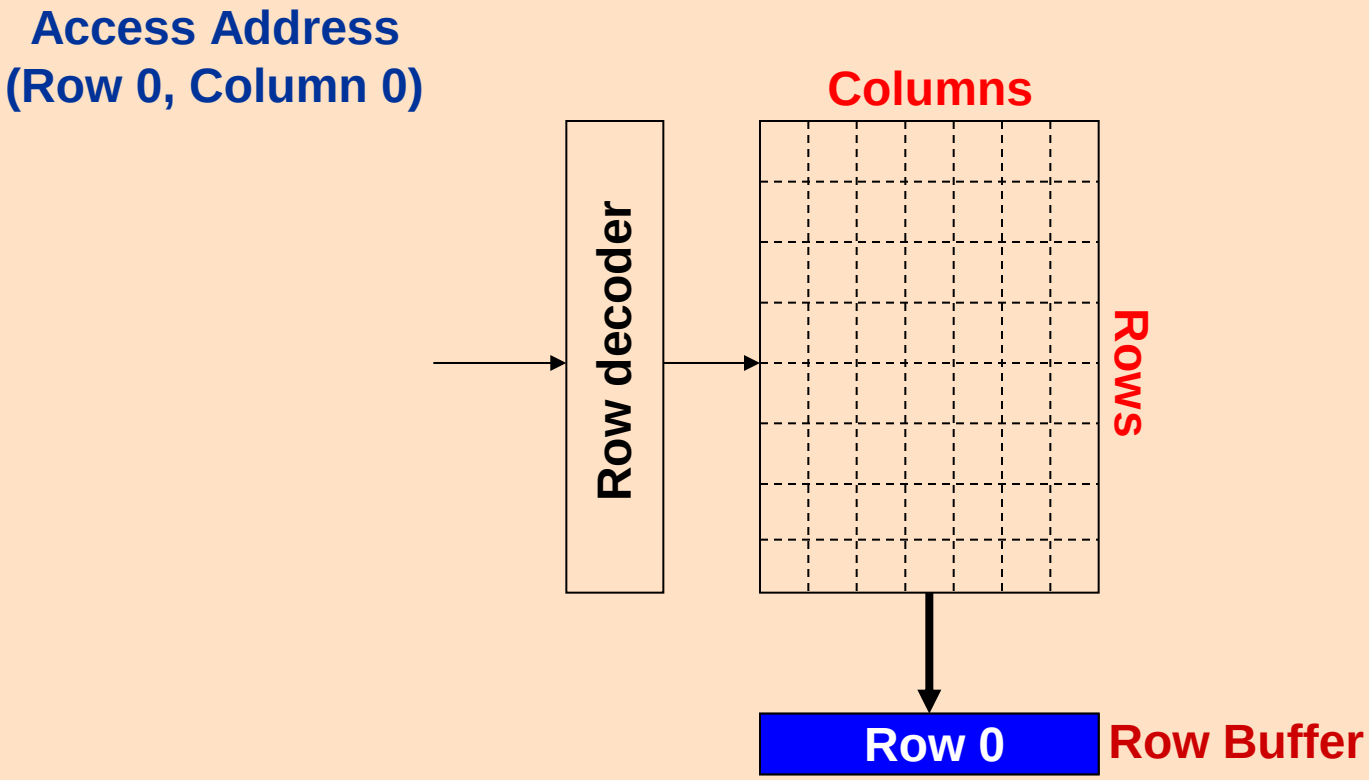
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



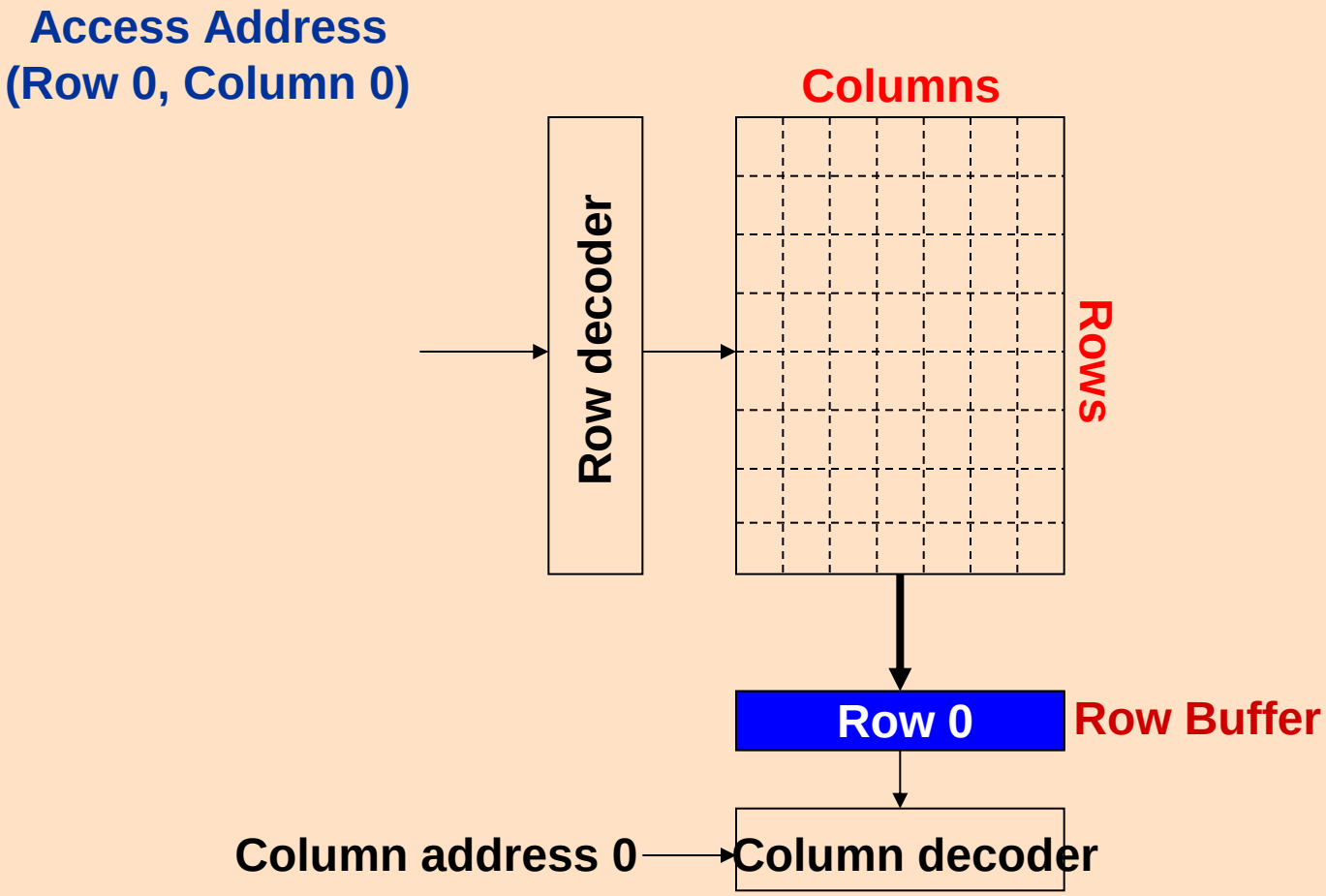
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



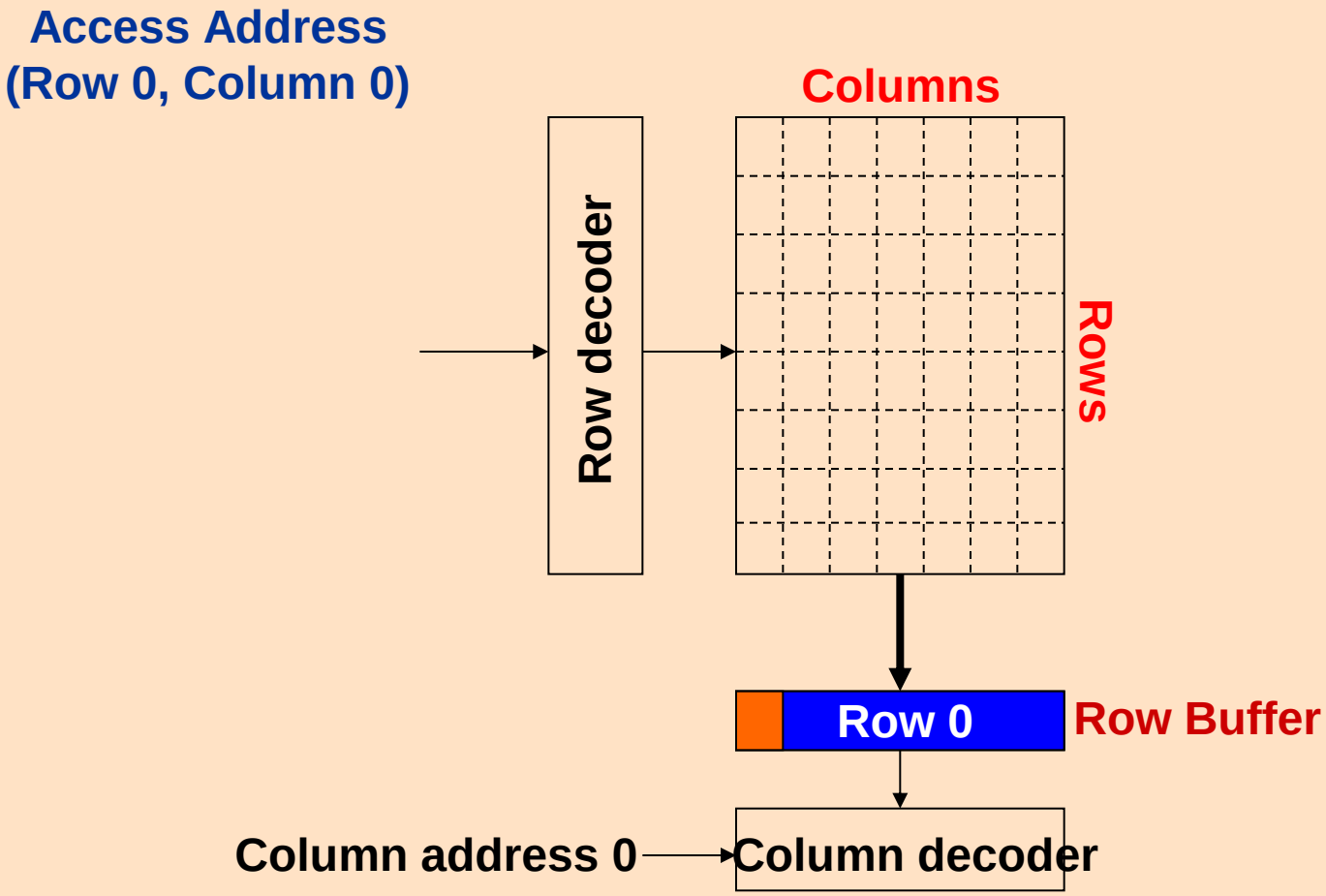
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



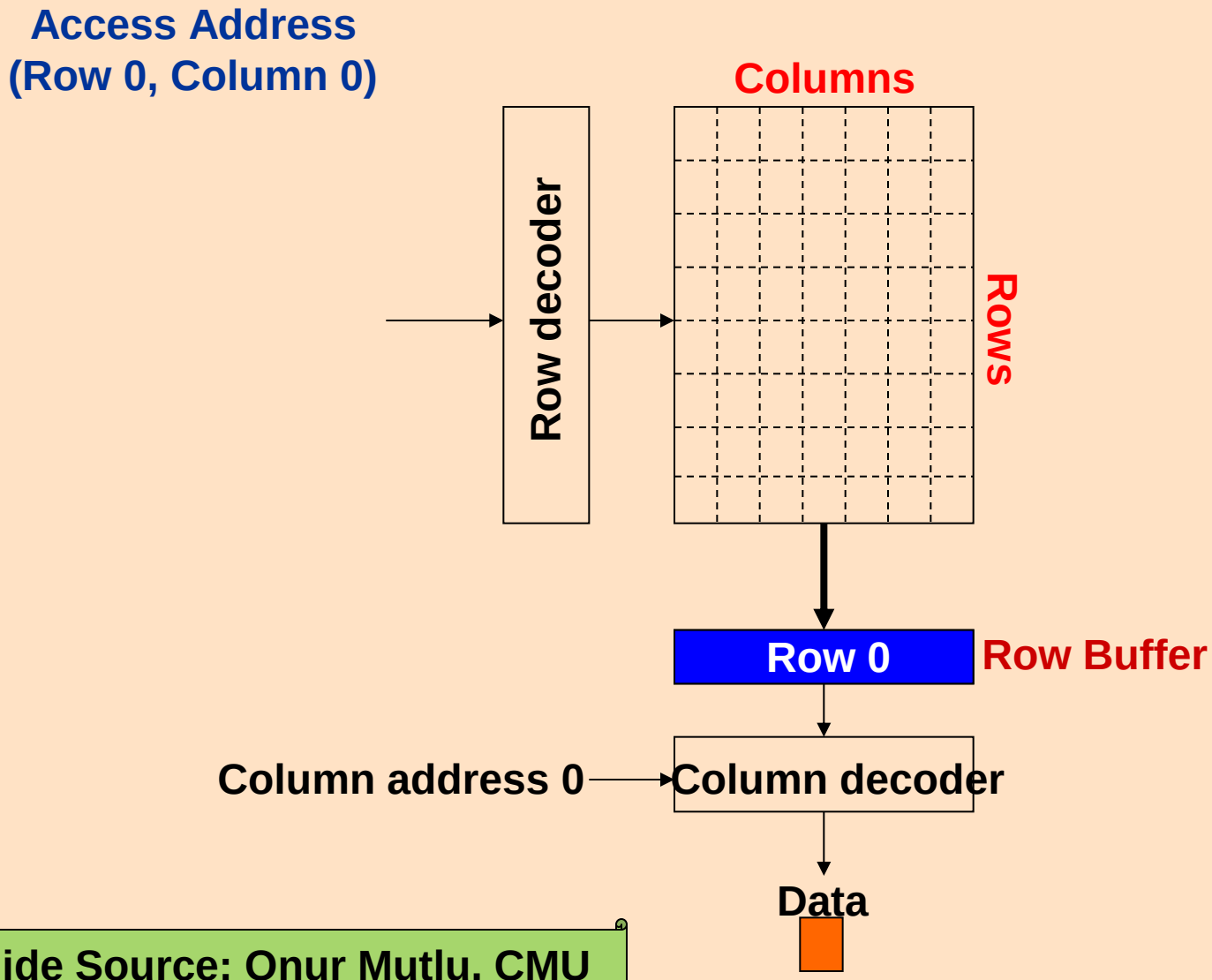
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



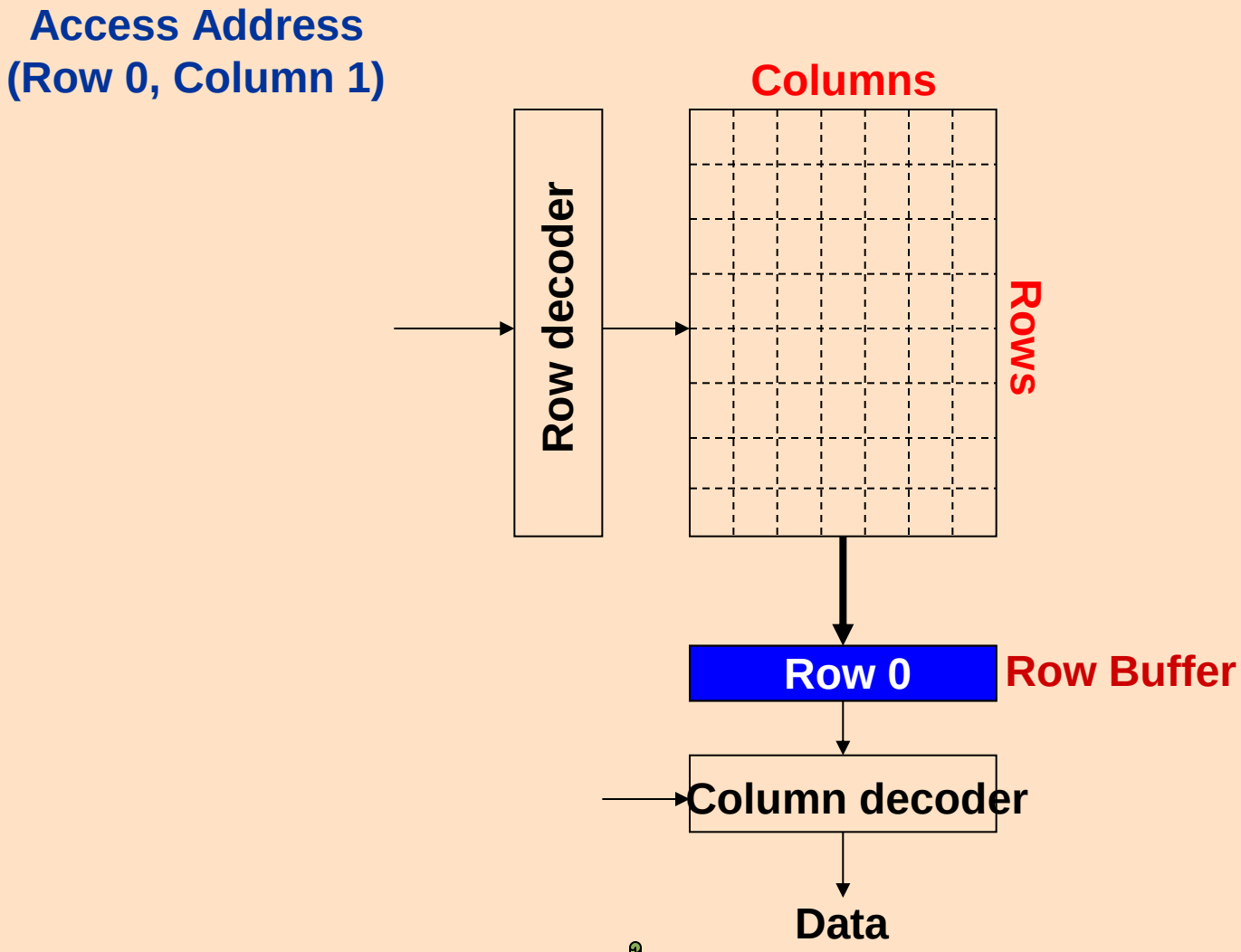
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



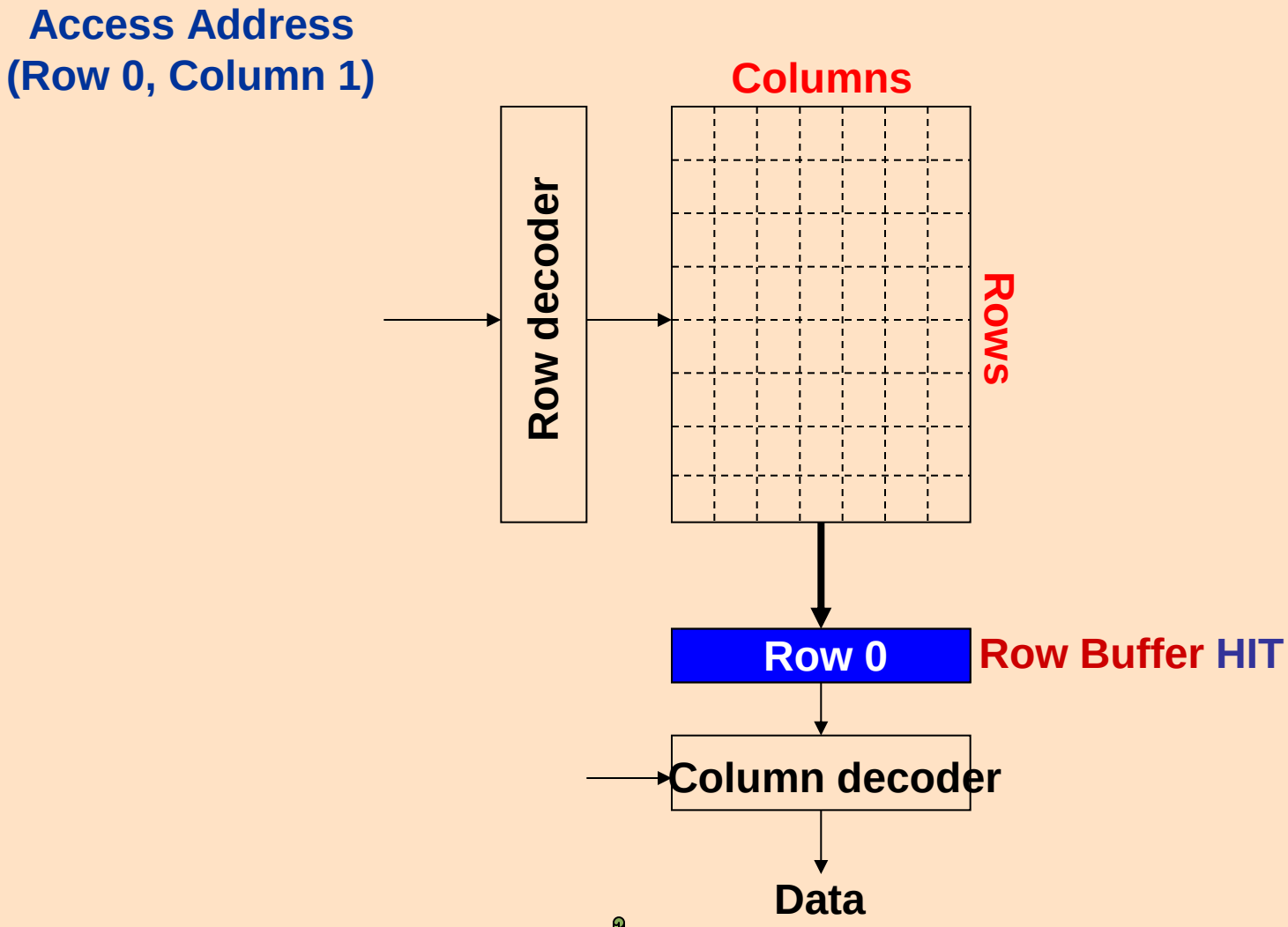
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



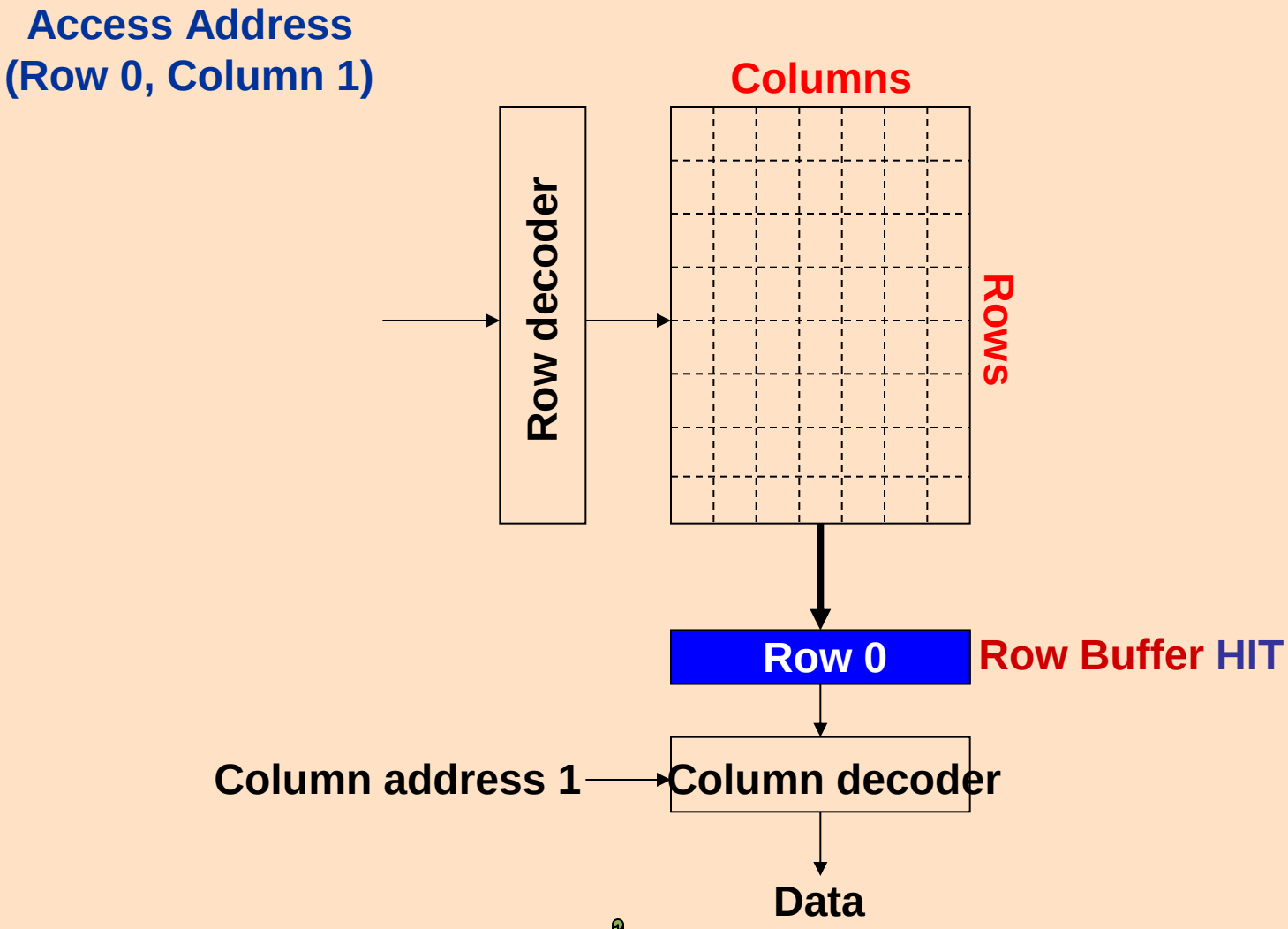
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



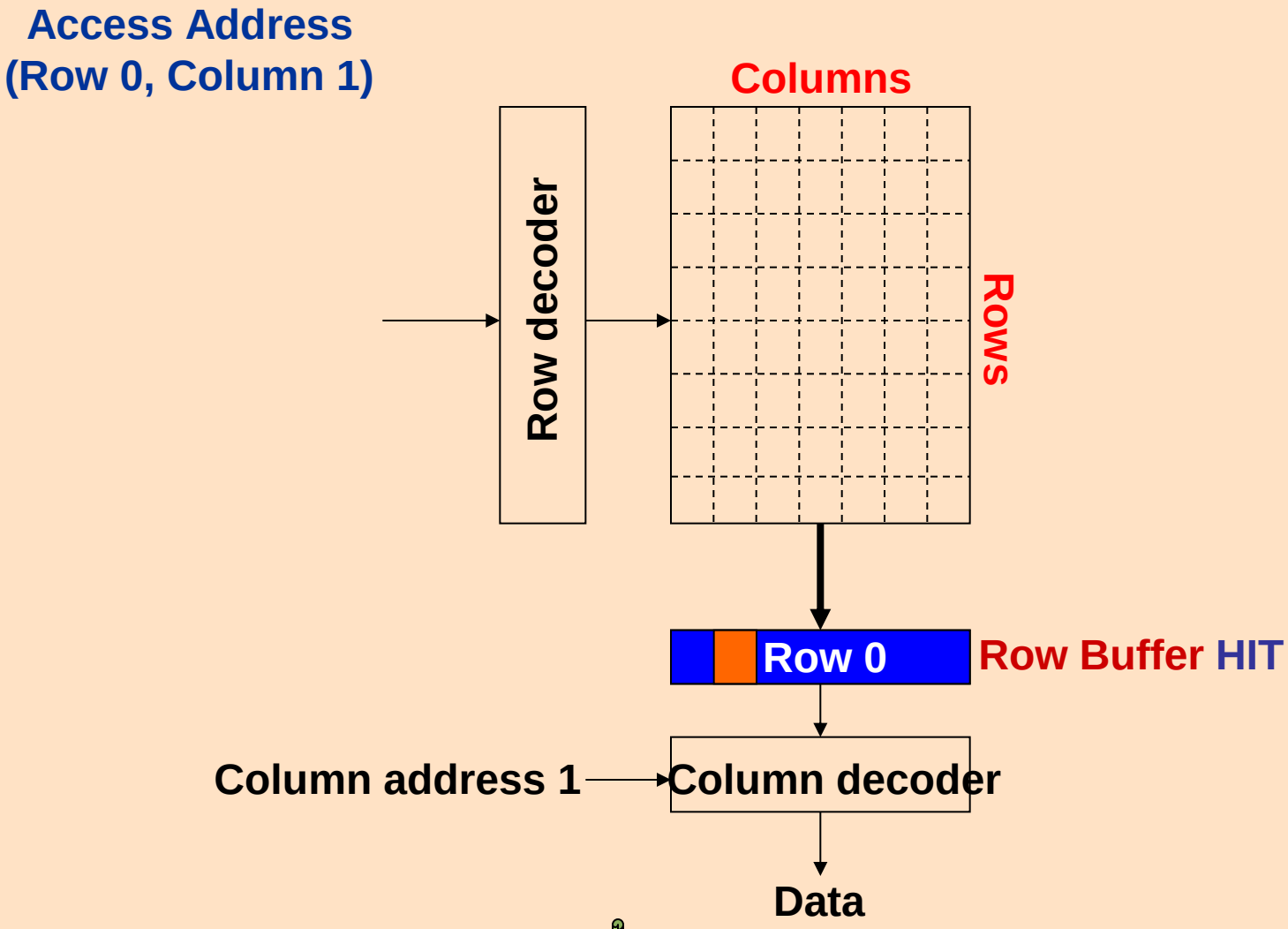
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



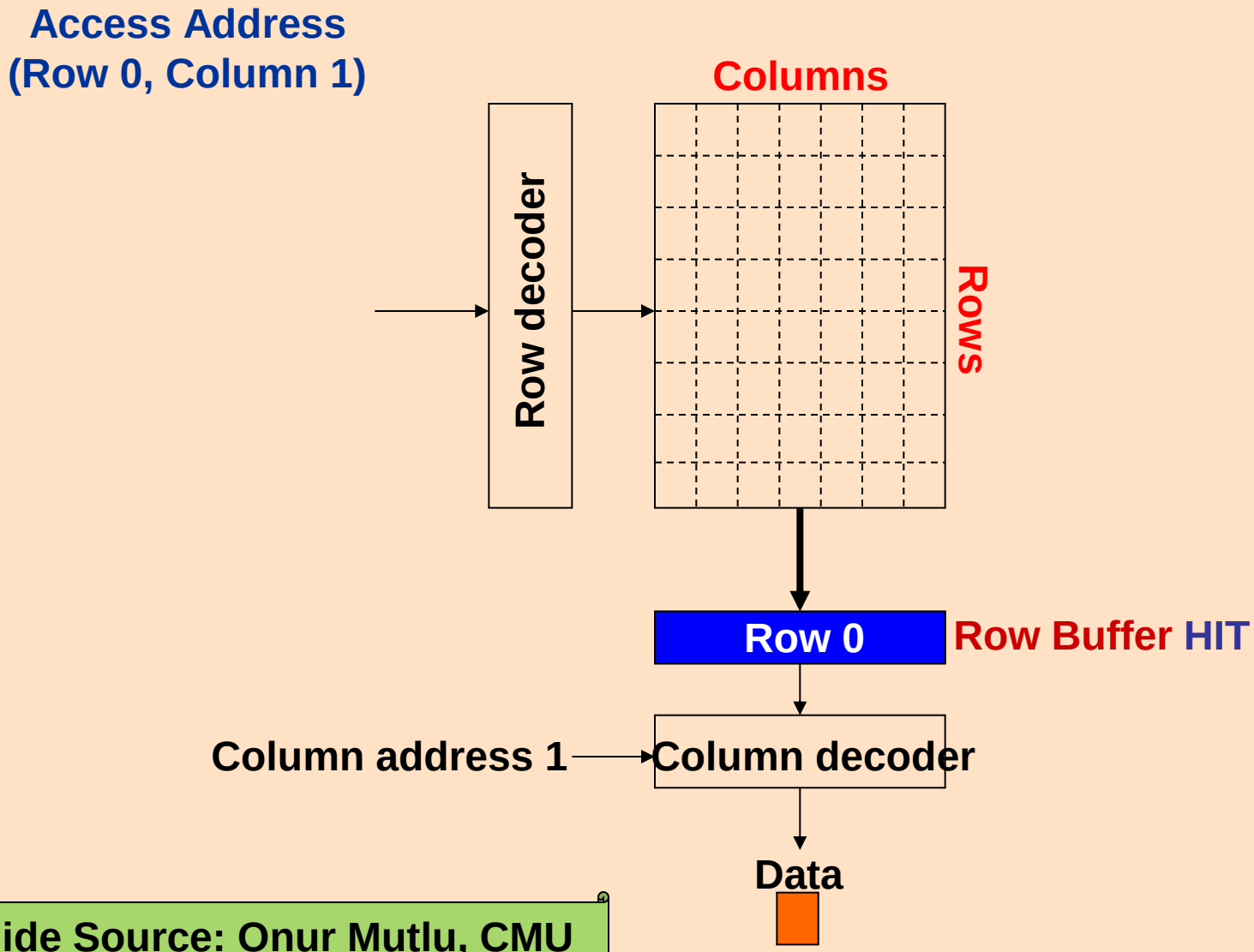
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



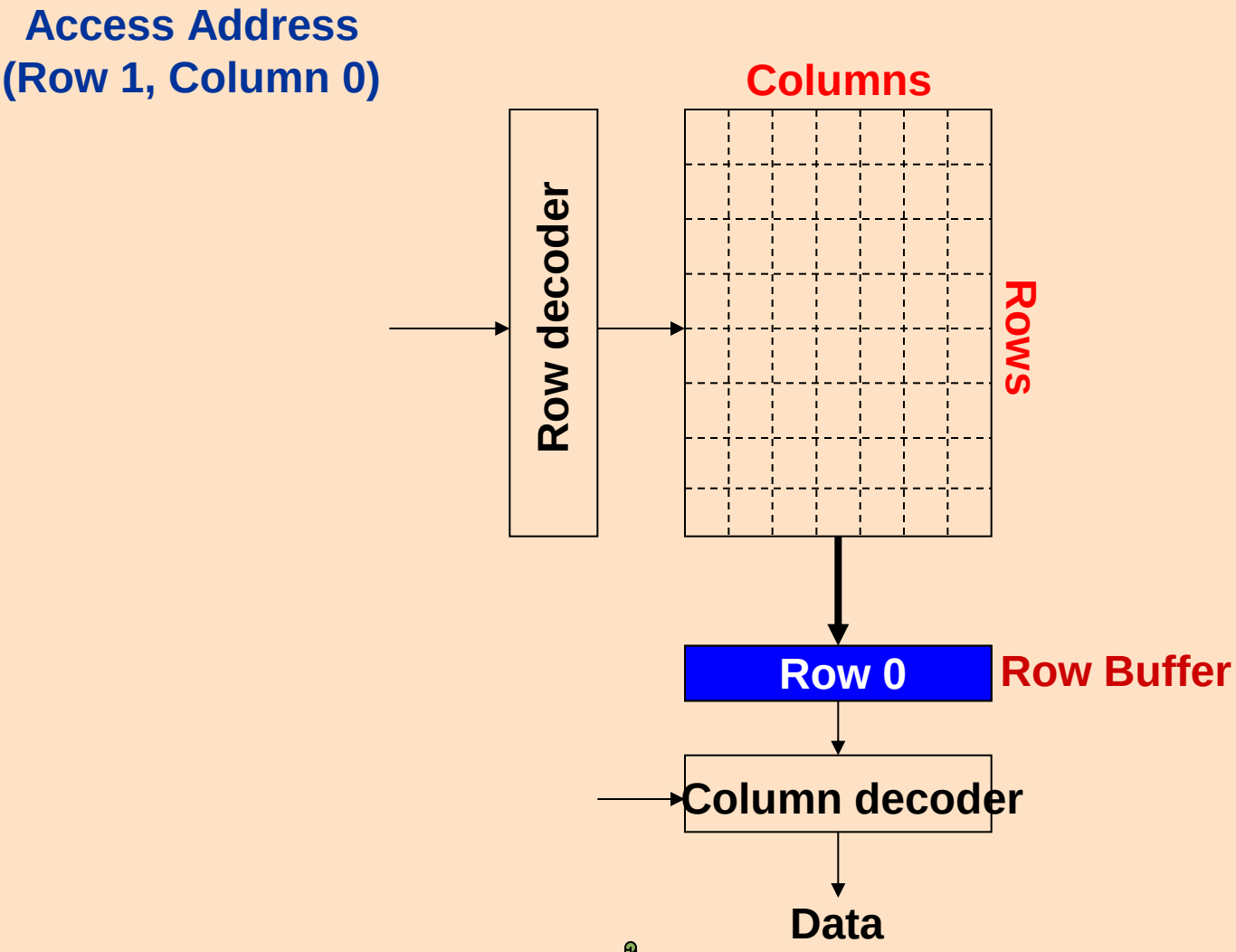
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



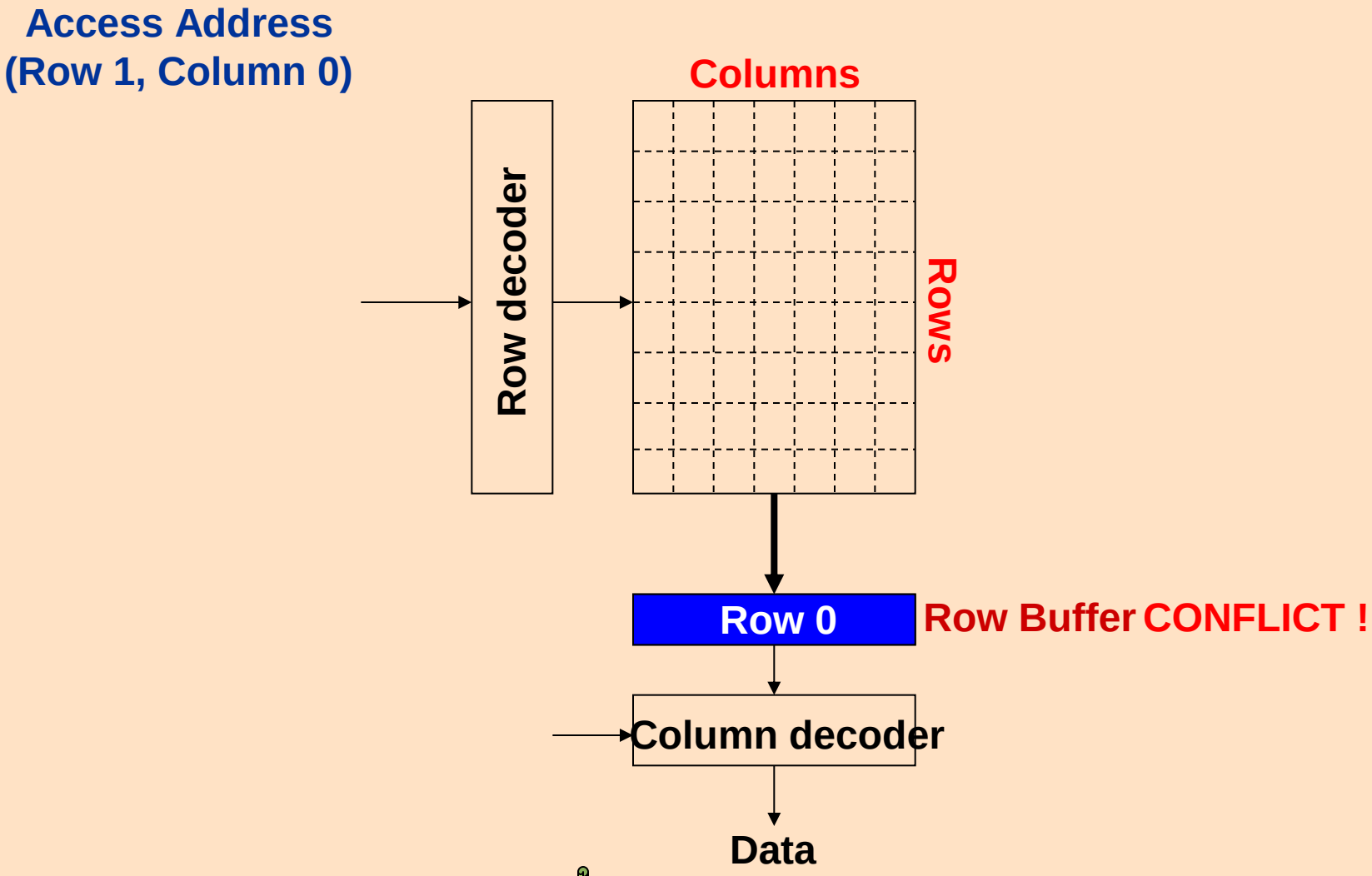
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



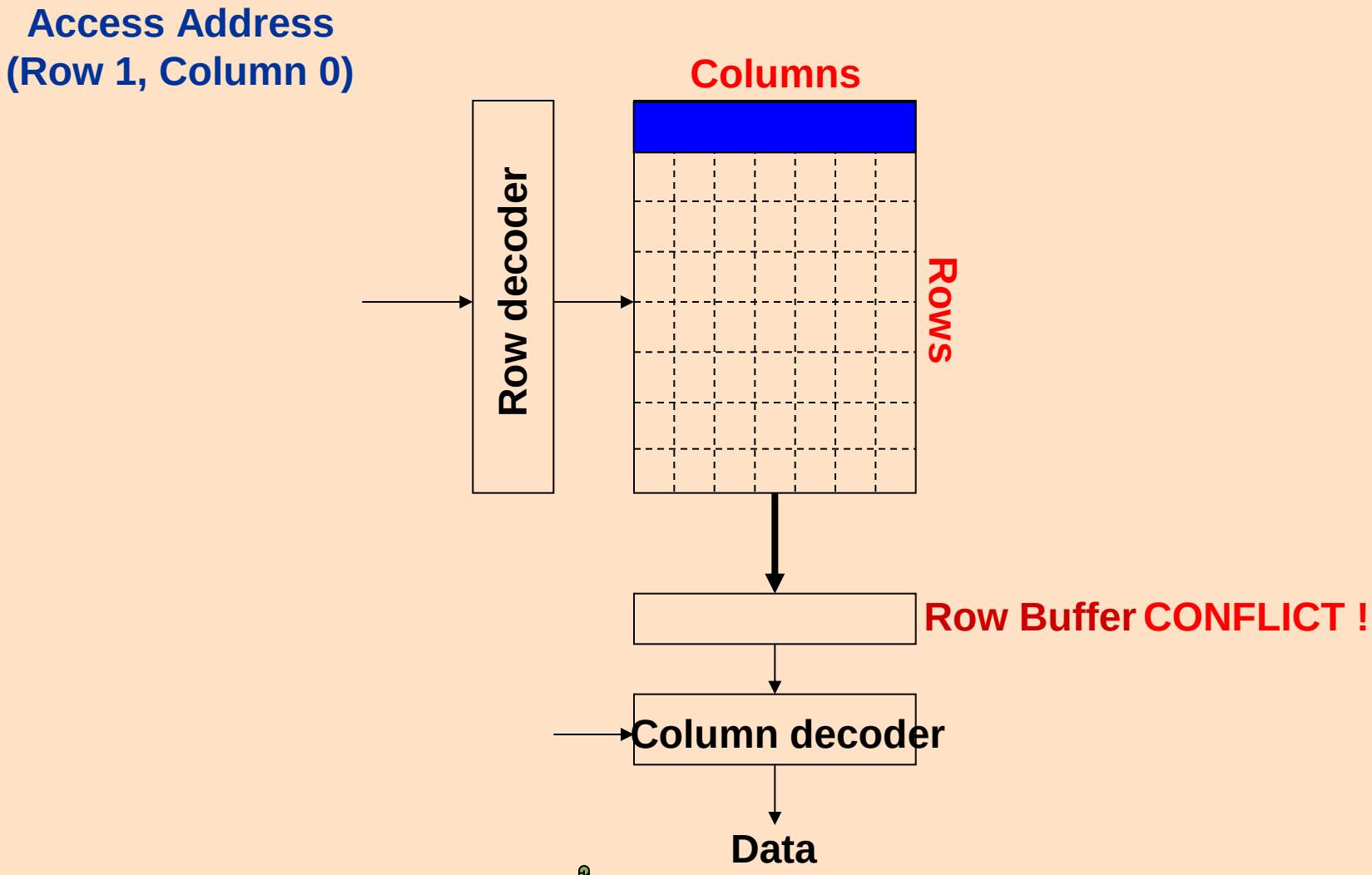
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



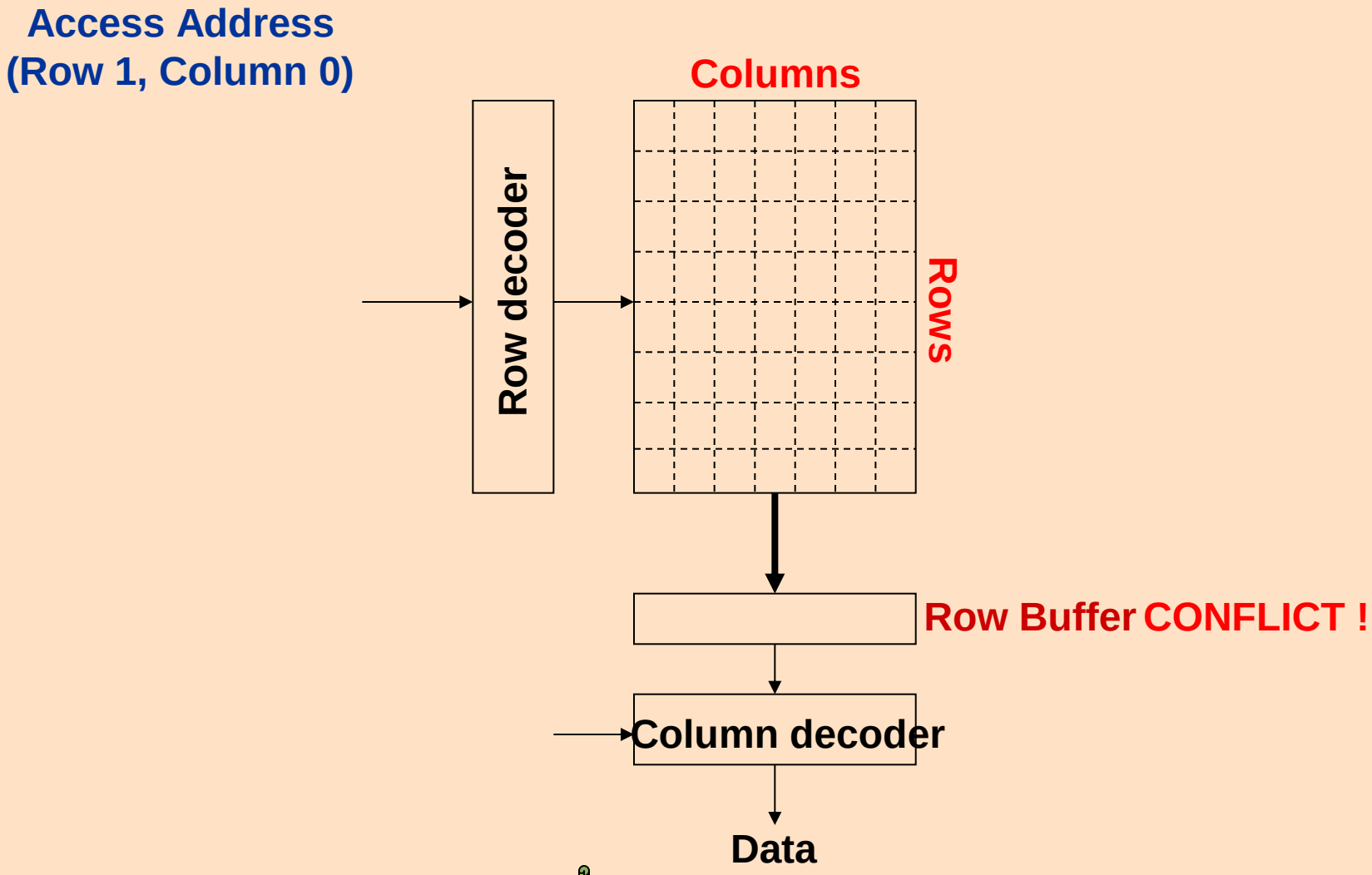
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



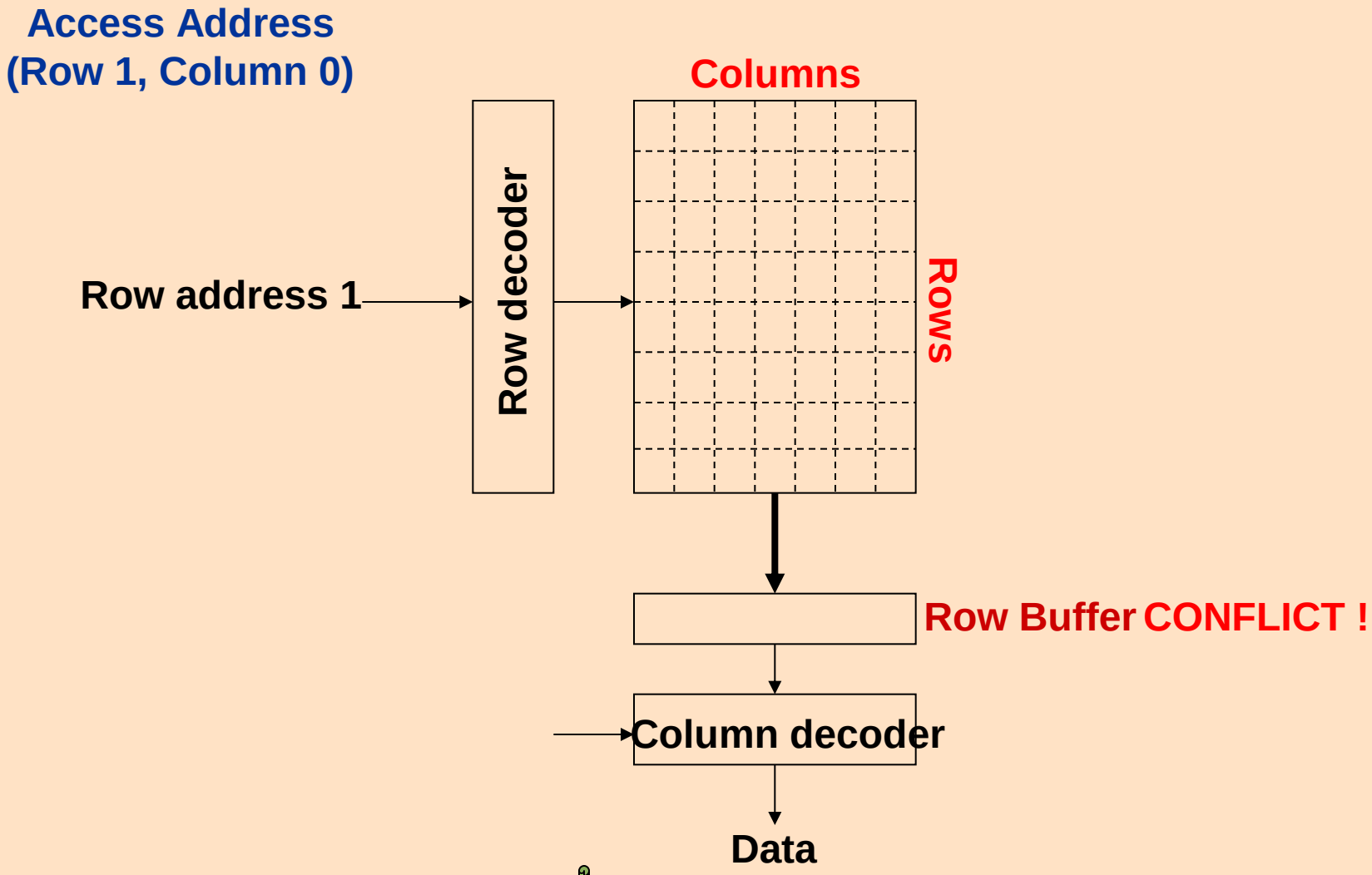
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



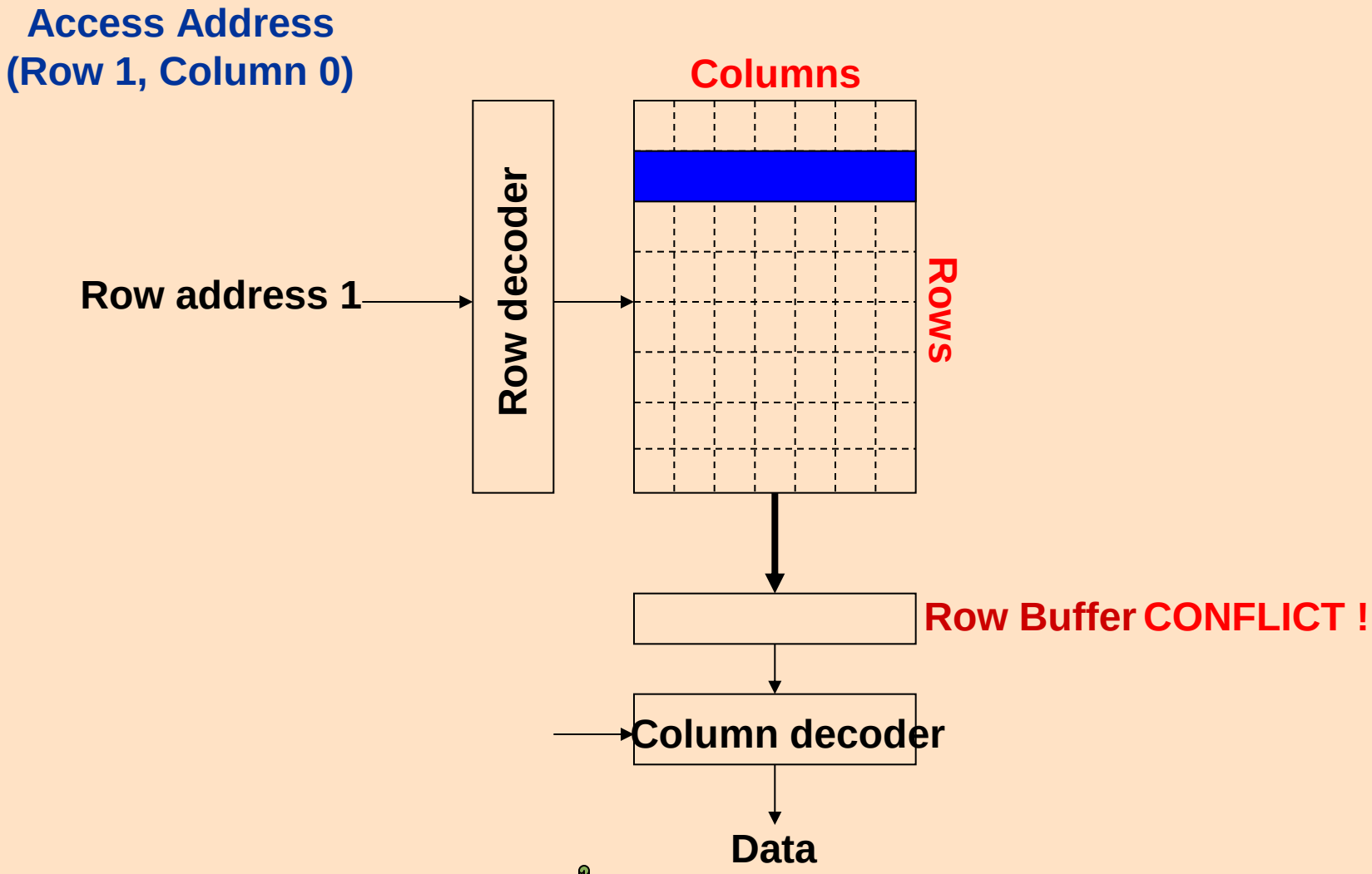
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



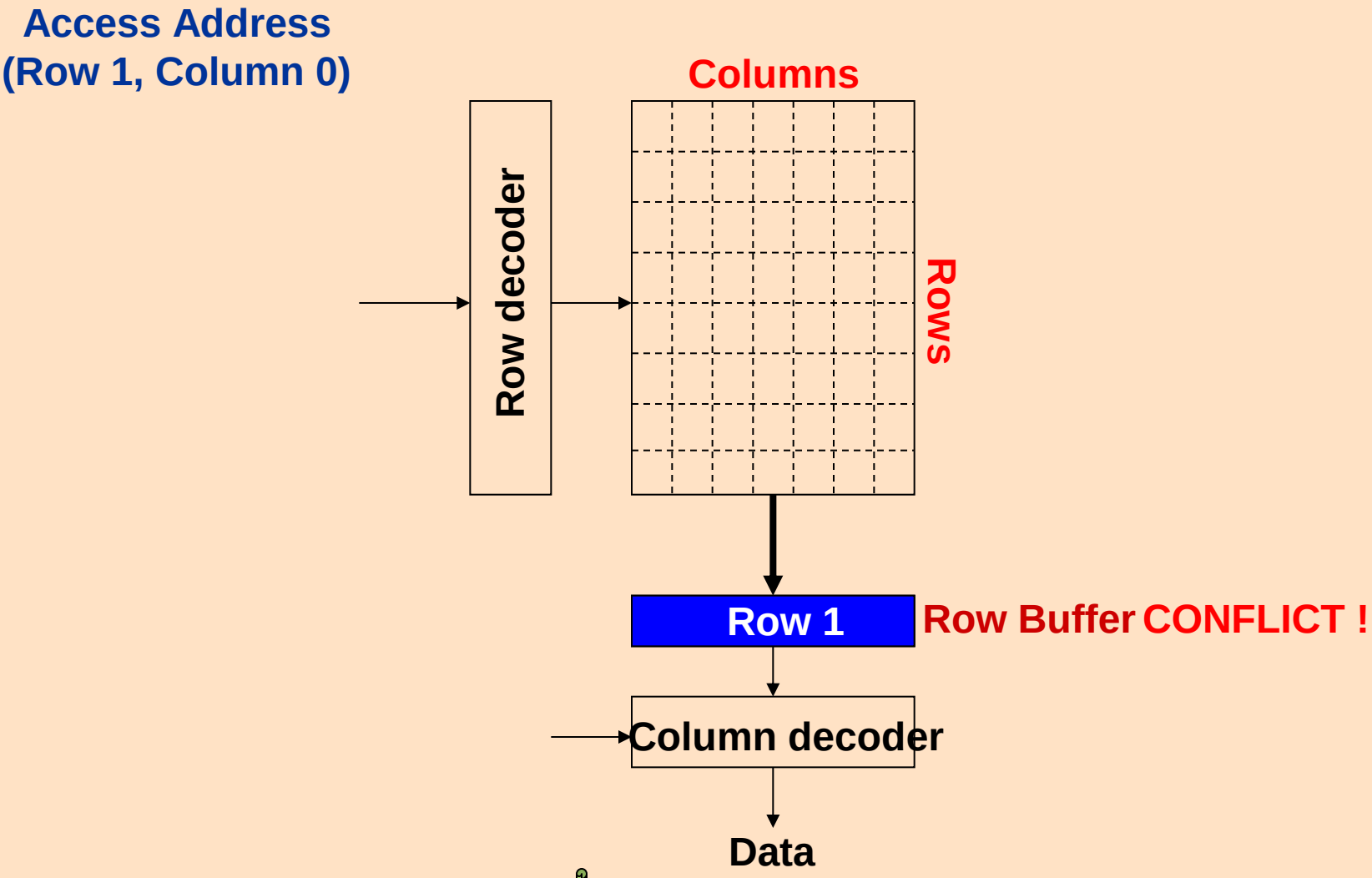
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



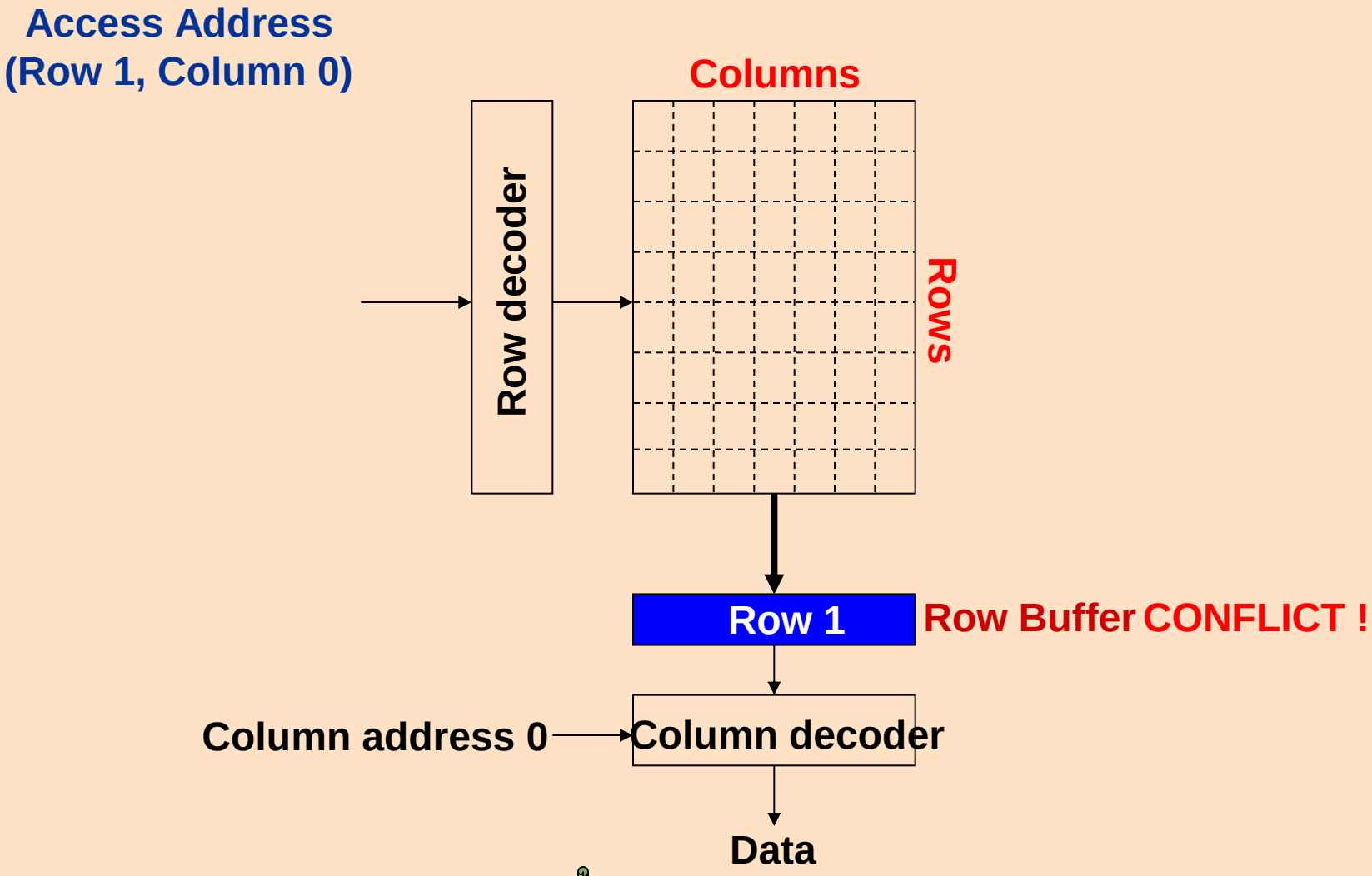
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



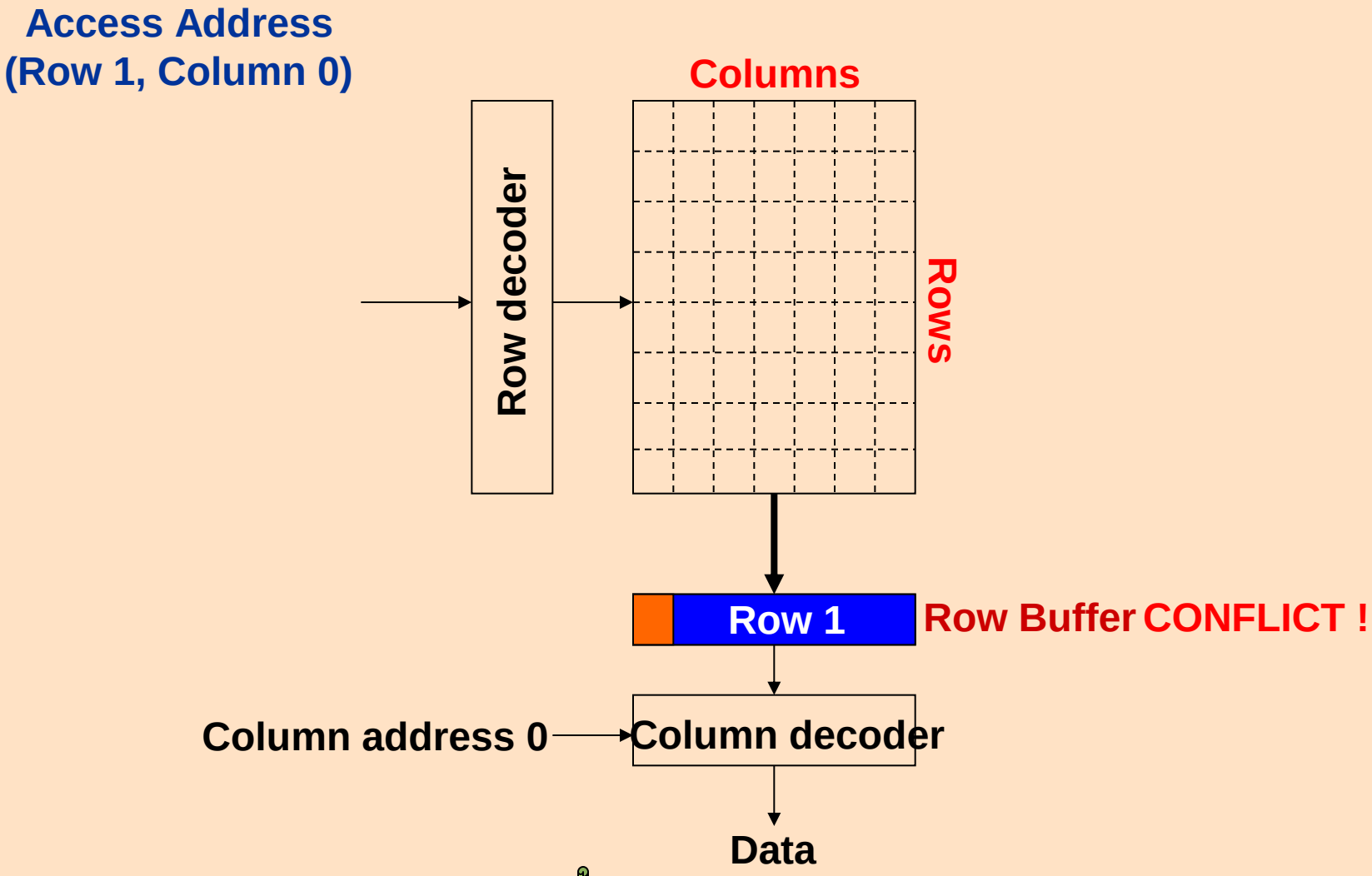
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



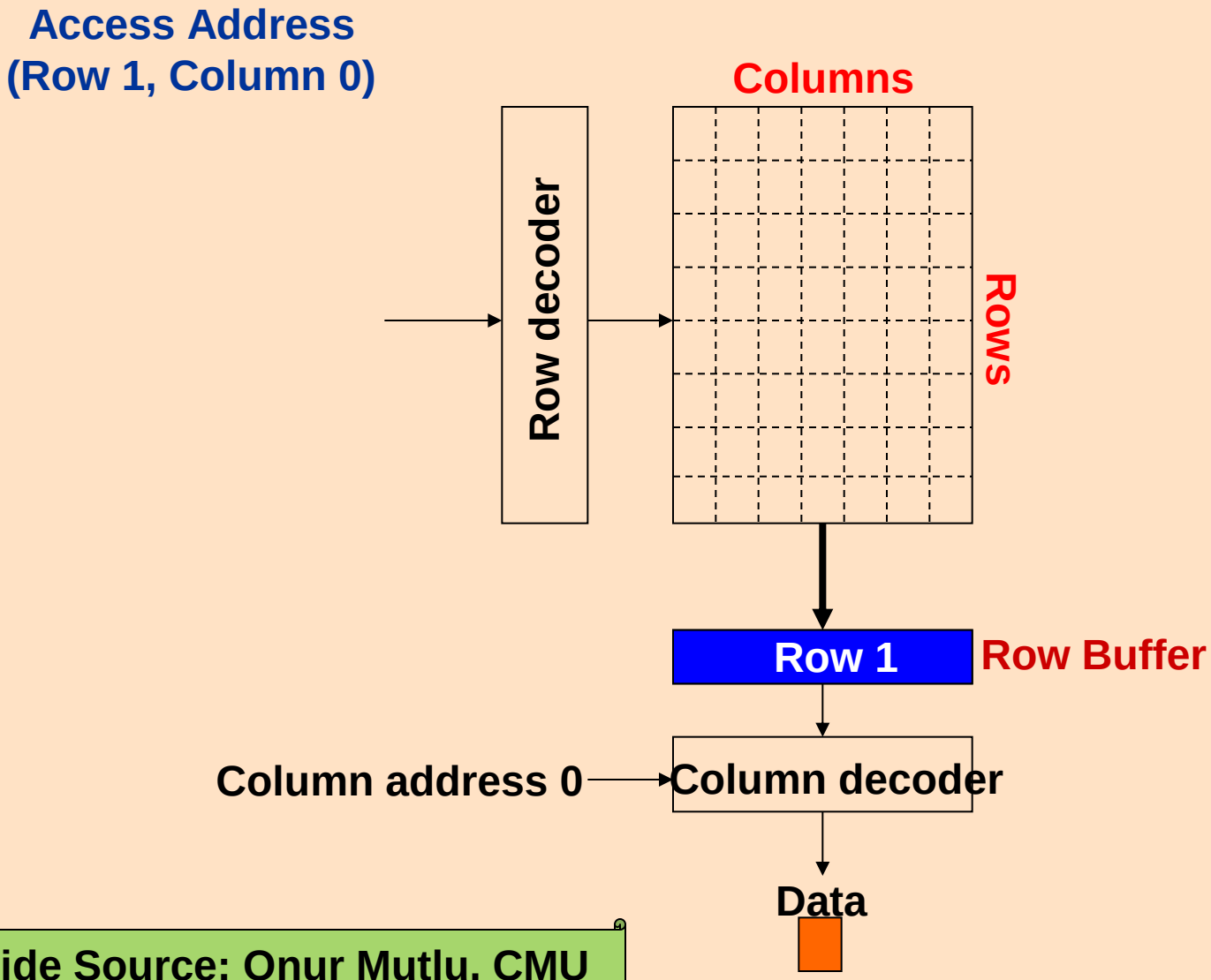
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



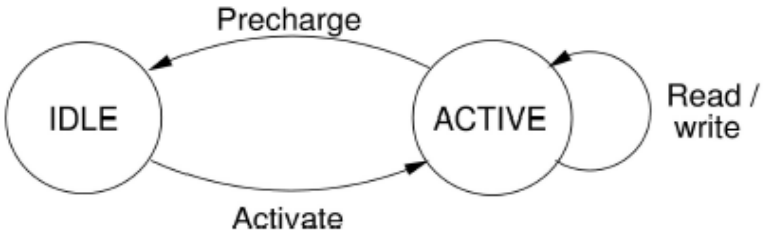
Slide Source: Onur Mutlu, CMU

DRAM Bank Operation



Slide Source: Onur Mutlu, CMU

DRAM Command Summary

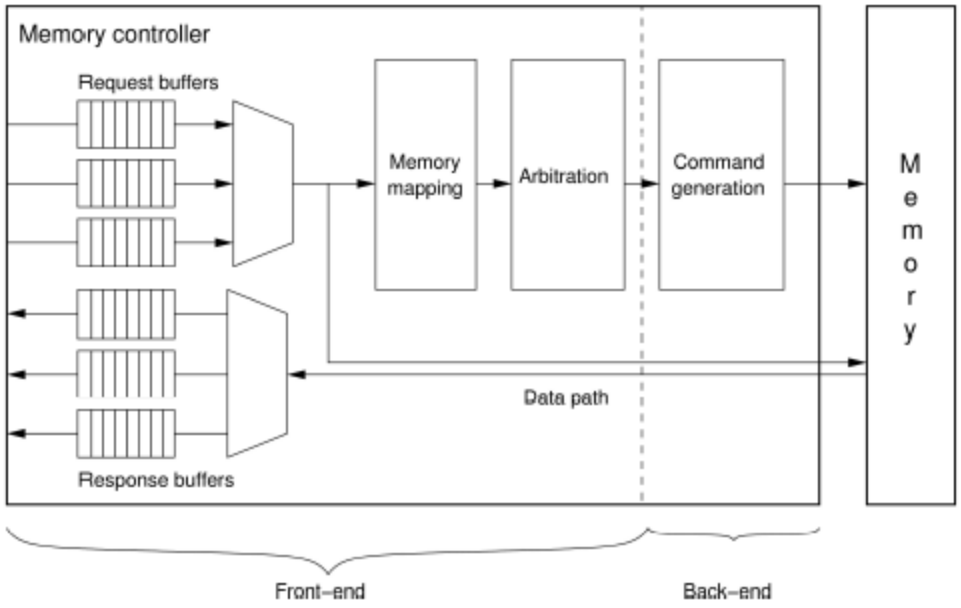


No operation	NOP	Ignores all inputs
Activate	ACT	Activate a row in a particular bank
Read	RD	Initiate a read burst to an active row
Write	WR	Initiate a write burst to an active row
Precharge	PRE	Close a row in a particular bank
Refresh	REF	Start a refresh operation

Slide Source: S. Rixner

DRAM Memory Controller

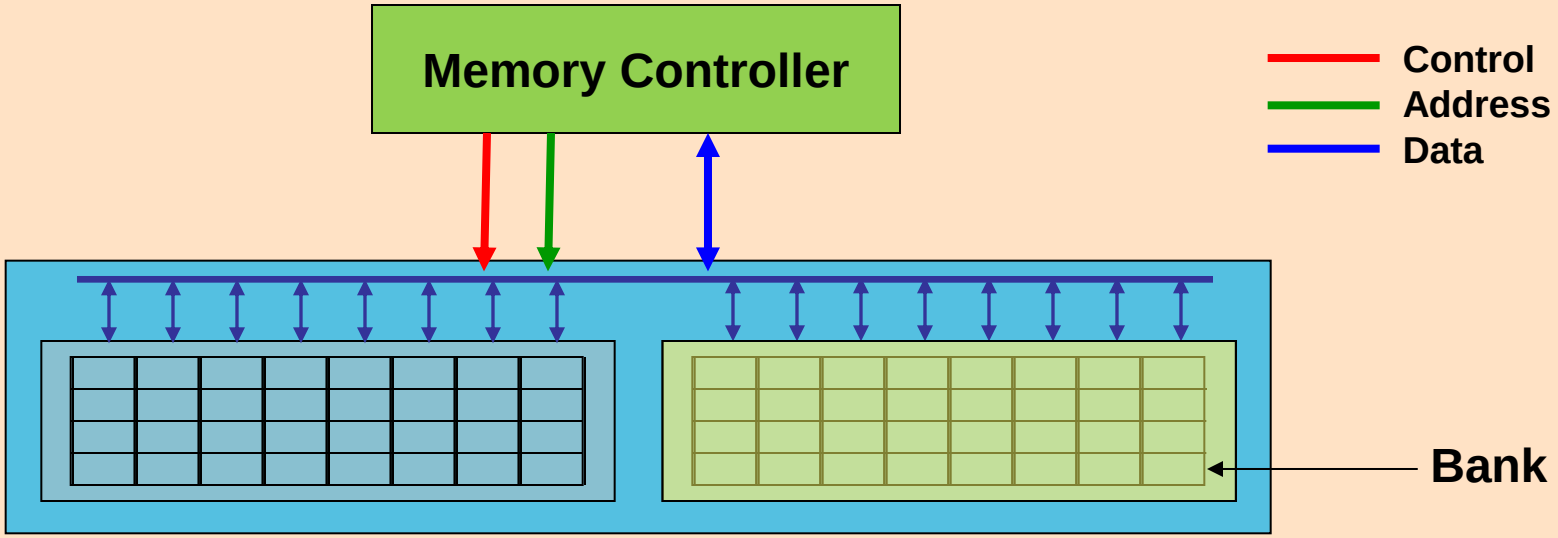
- Frontend
  - Request/Response Buffers
  - Memory mapping
  - Arbiter
- Controller Backend
  - Command Generator
- Timing to be obeyed



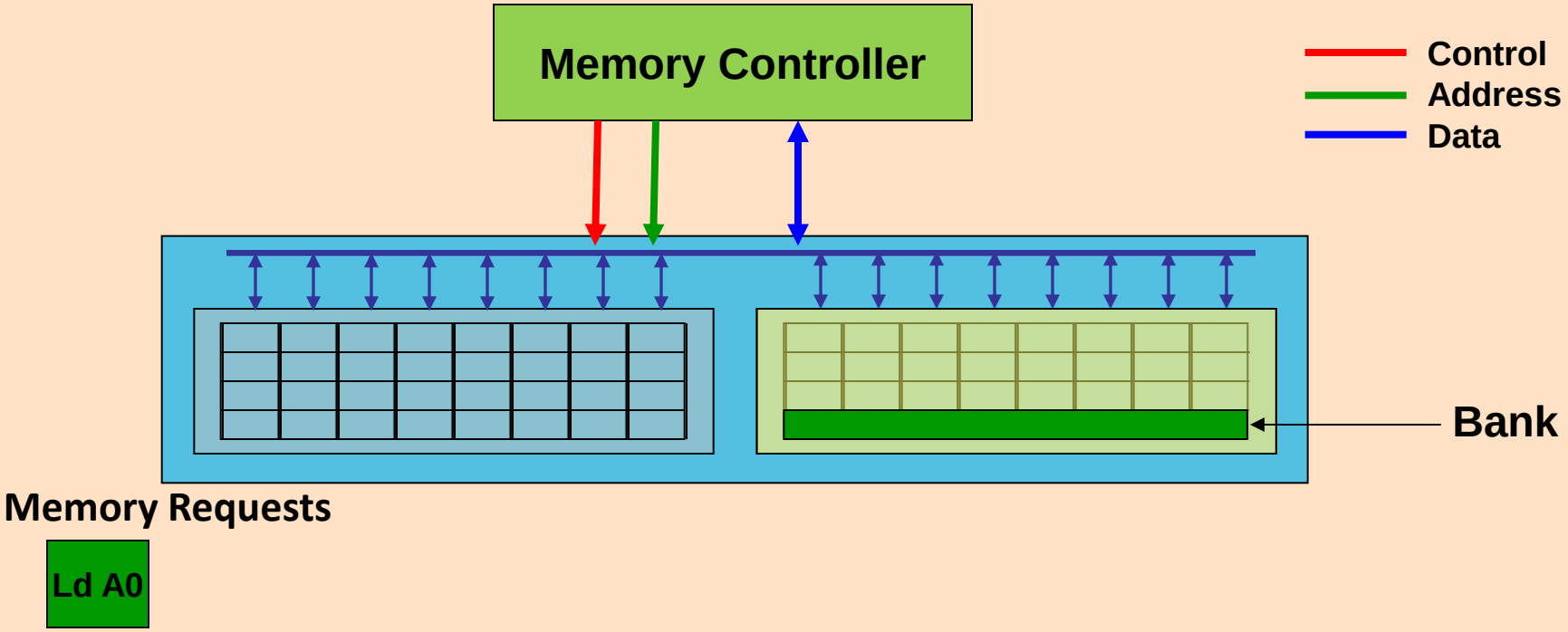
Command delay	Cycles
ACT to PRE	9
ACT to ACT (same bank)	12
ACT to ACT (diff. bank)	2
ACT to RD/WR	3
WR to RD turn-around	2

Slide Source: S. Rixner

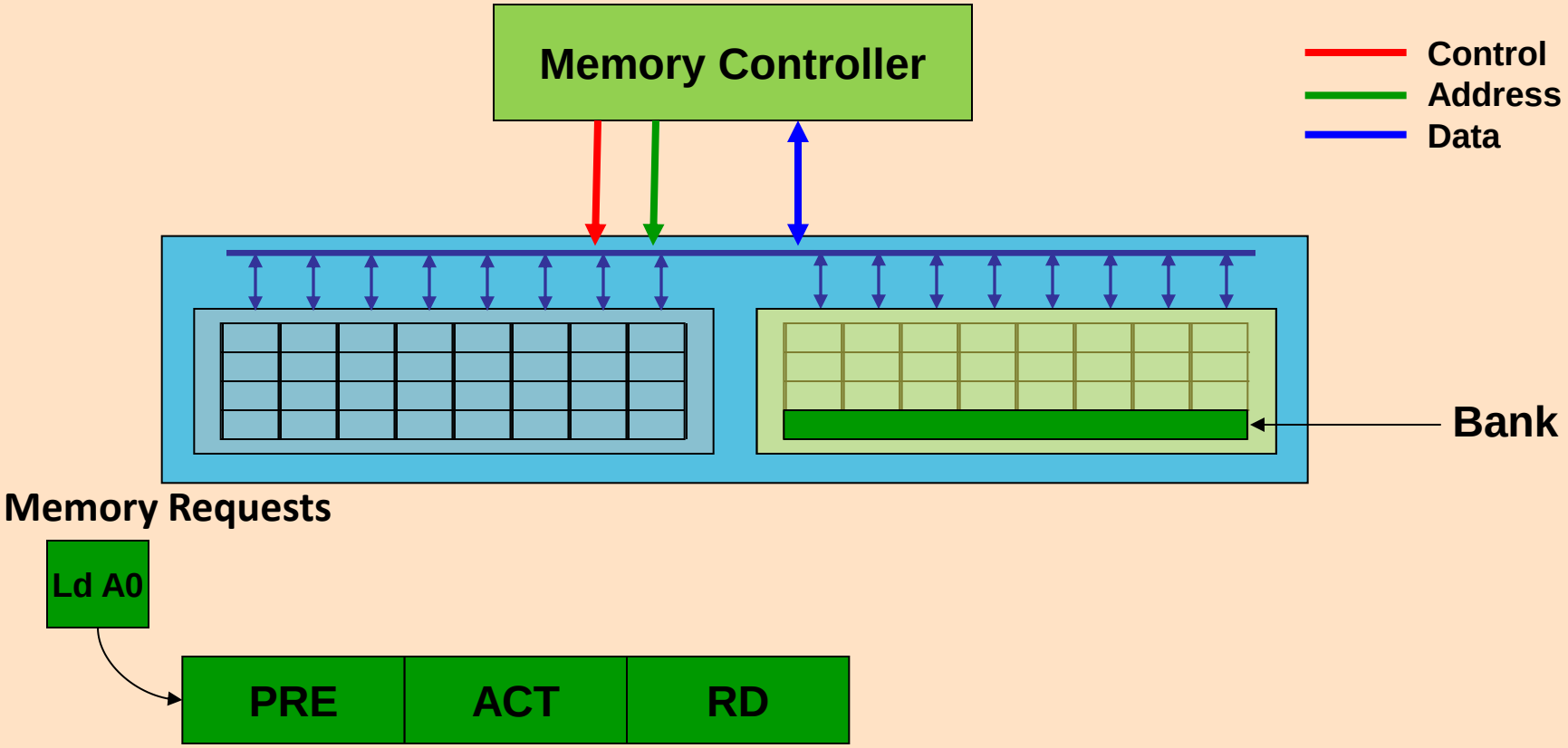
Bank Level Parallelism in DRAM



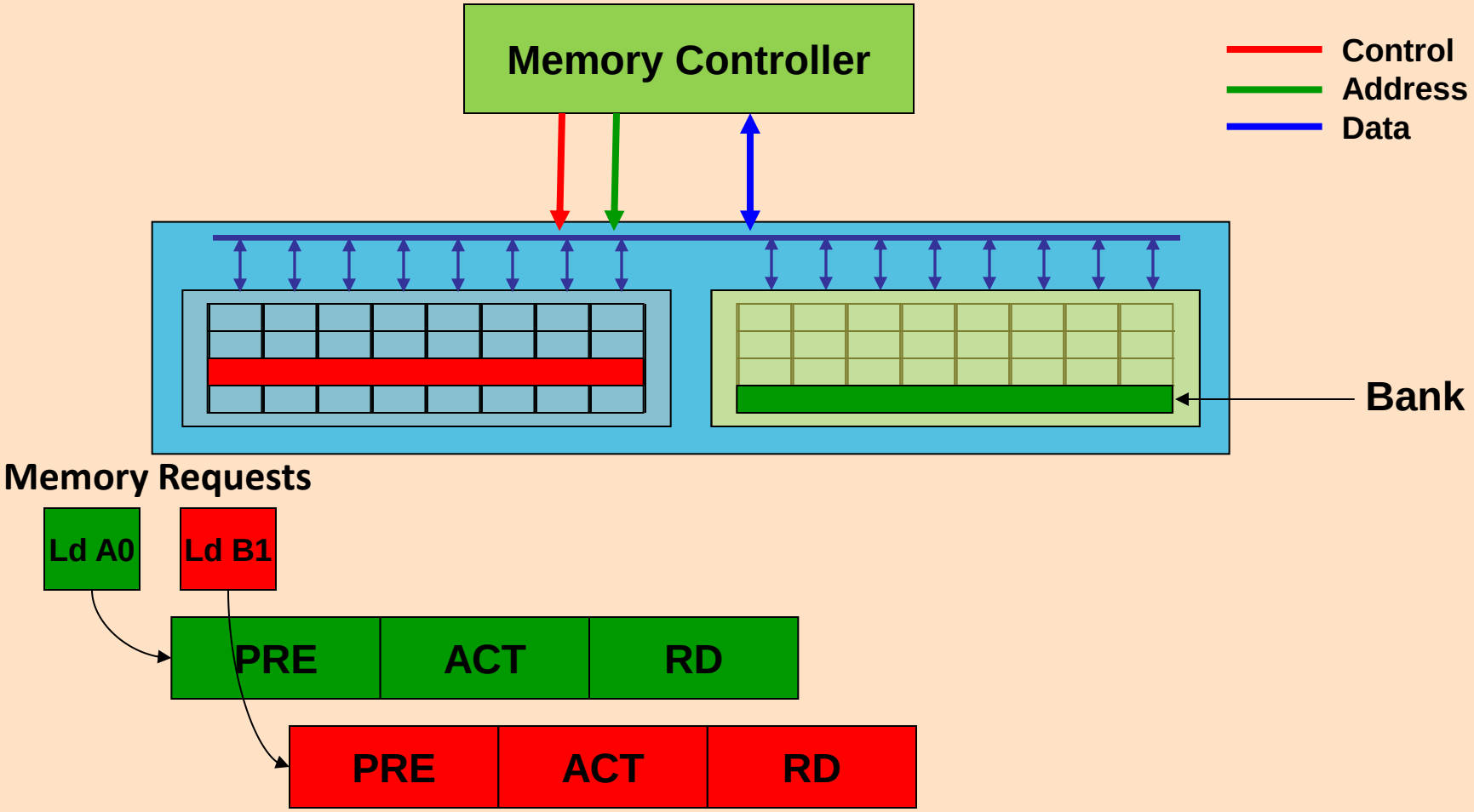
Bank Level Parallelism in DRAM



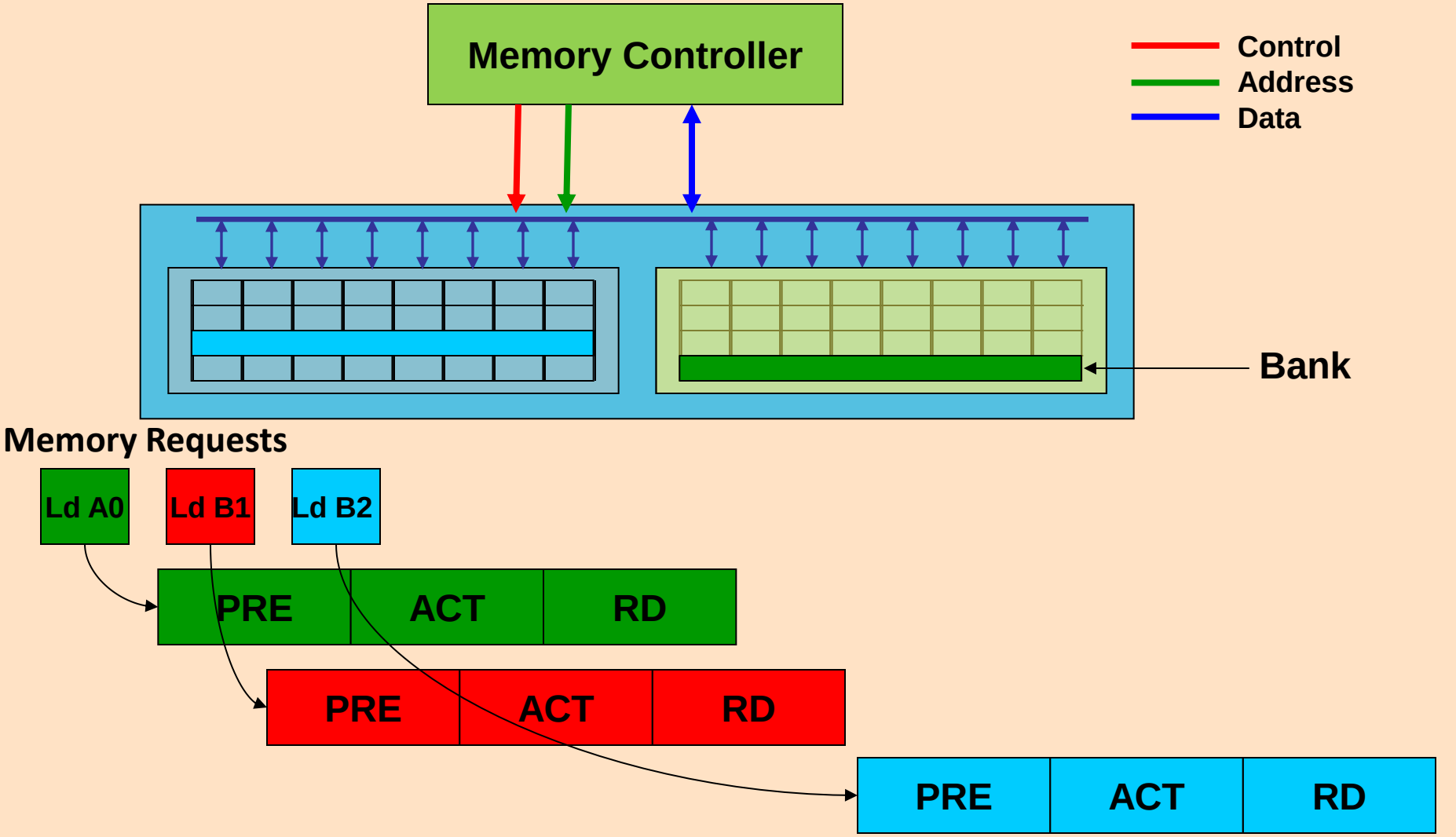
Bank Level Parallelism in DRAM



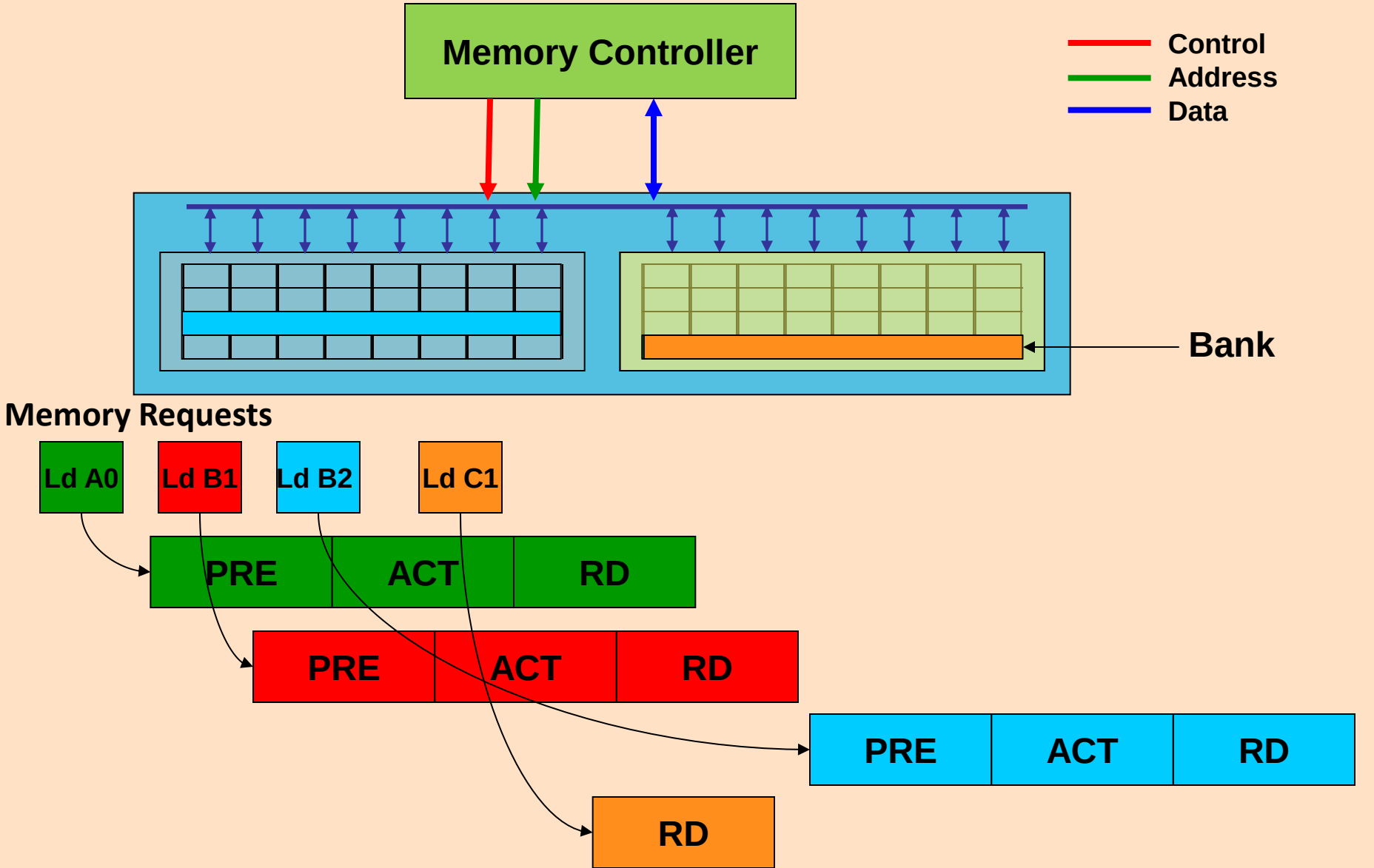
Bank Level Parallelism in DRAM



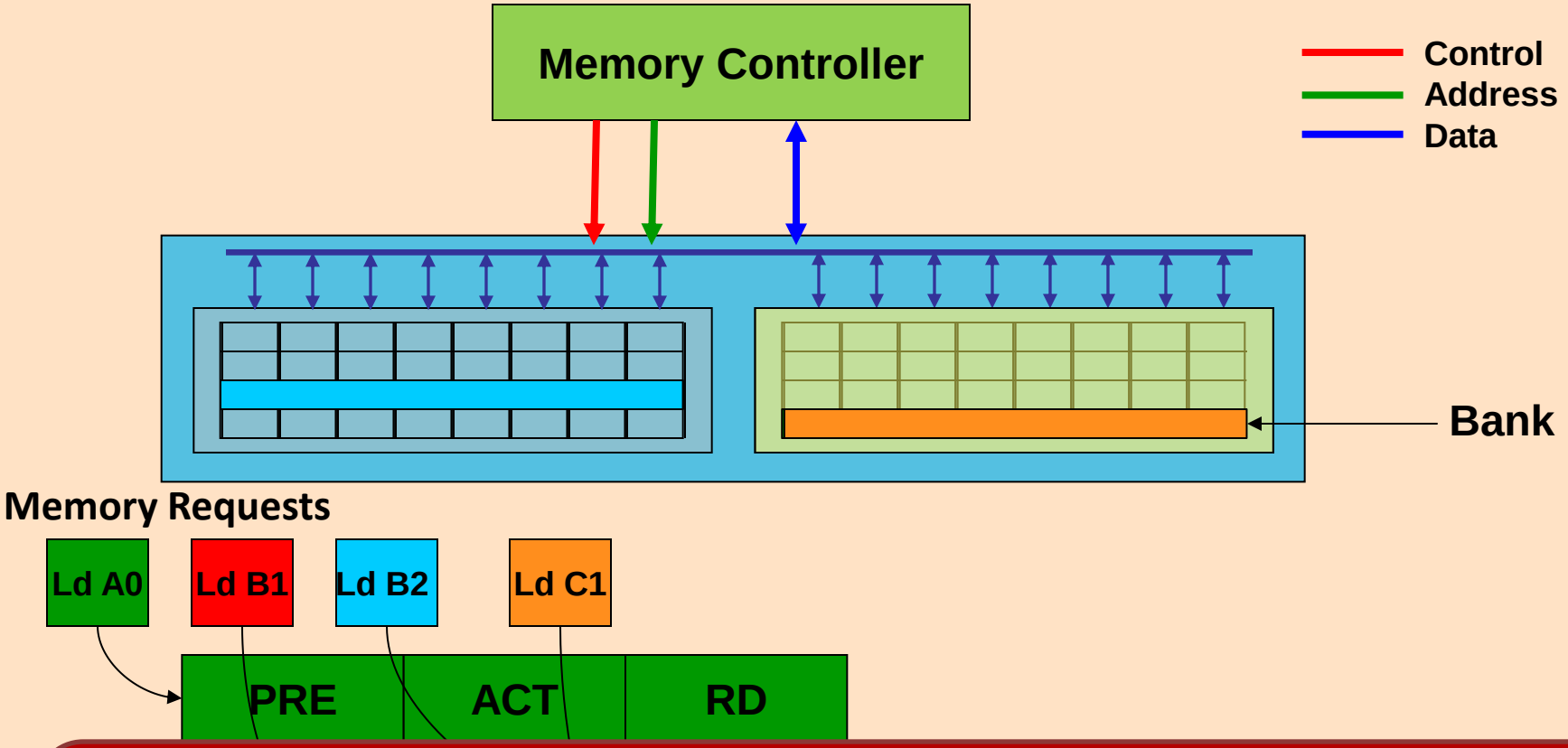
Bank Level Parallelism in DRAM



Bank Level Parallelism in DRAM



Bank Level Parallelism in DRAM



Bank Level Parallelism

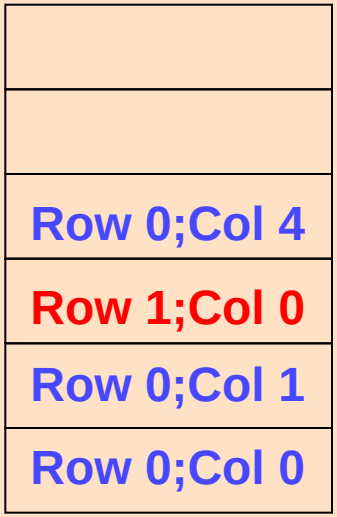
- Improves perf. with Parallelism and Row Buffer Hit
- Hurts perf. due to bank-to-bank switch delay

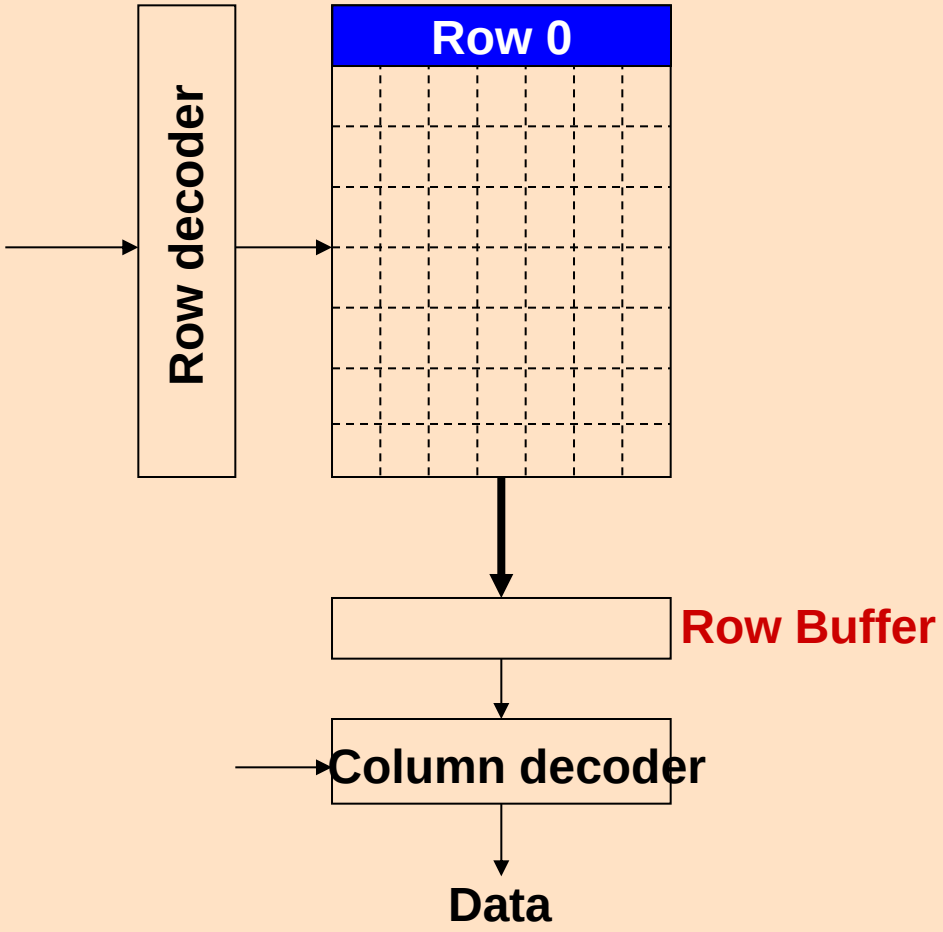
DRAM Refresh

- Capacitors leak and lose charge → Need periodic restoration of charge
- JEDEC Spec: At normal temp, cell retention time limit is 64ms. At high (extended) temp, retention time halves to 32ms.
- The memory controller issues refresh operations periodically.

- Assume 4GB DRAM with 2KB pages, organized as 16 banks
- 2M pages total, 128K pages per bank
- Refreshing a page takes 20ns (ACTIVATE+PRECHARGE)
- Refreshing all pages in a bank → 2.6ms!
- → 2.6/64 = 4% overhead!

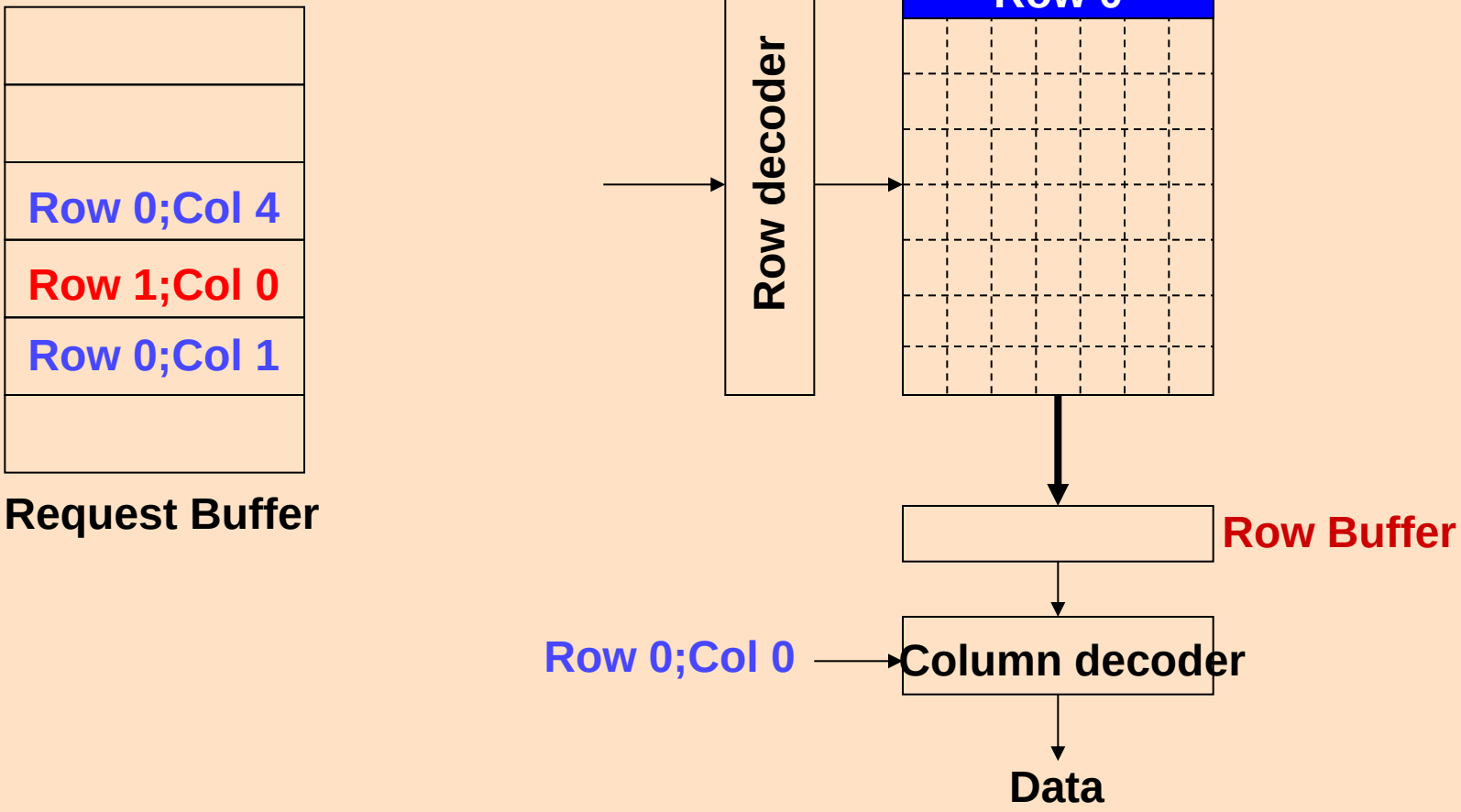
Memory Access Scheduling: FCFS





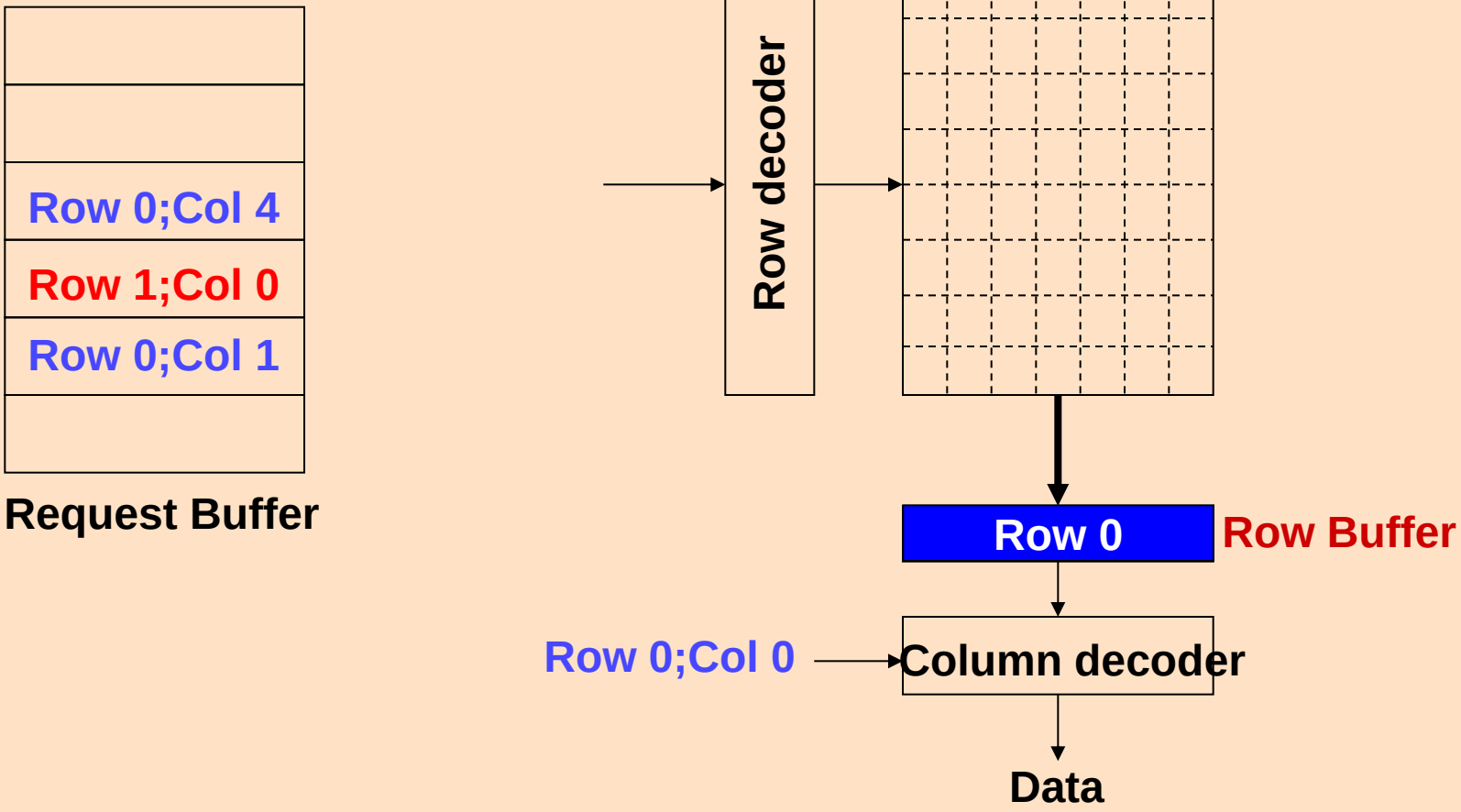
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



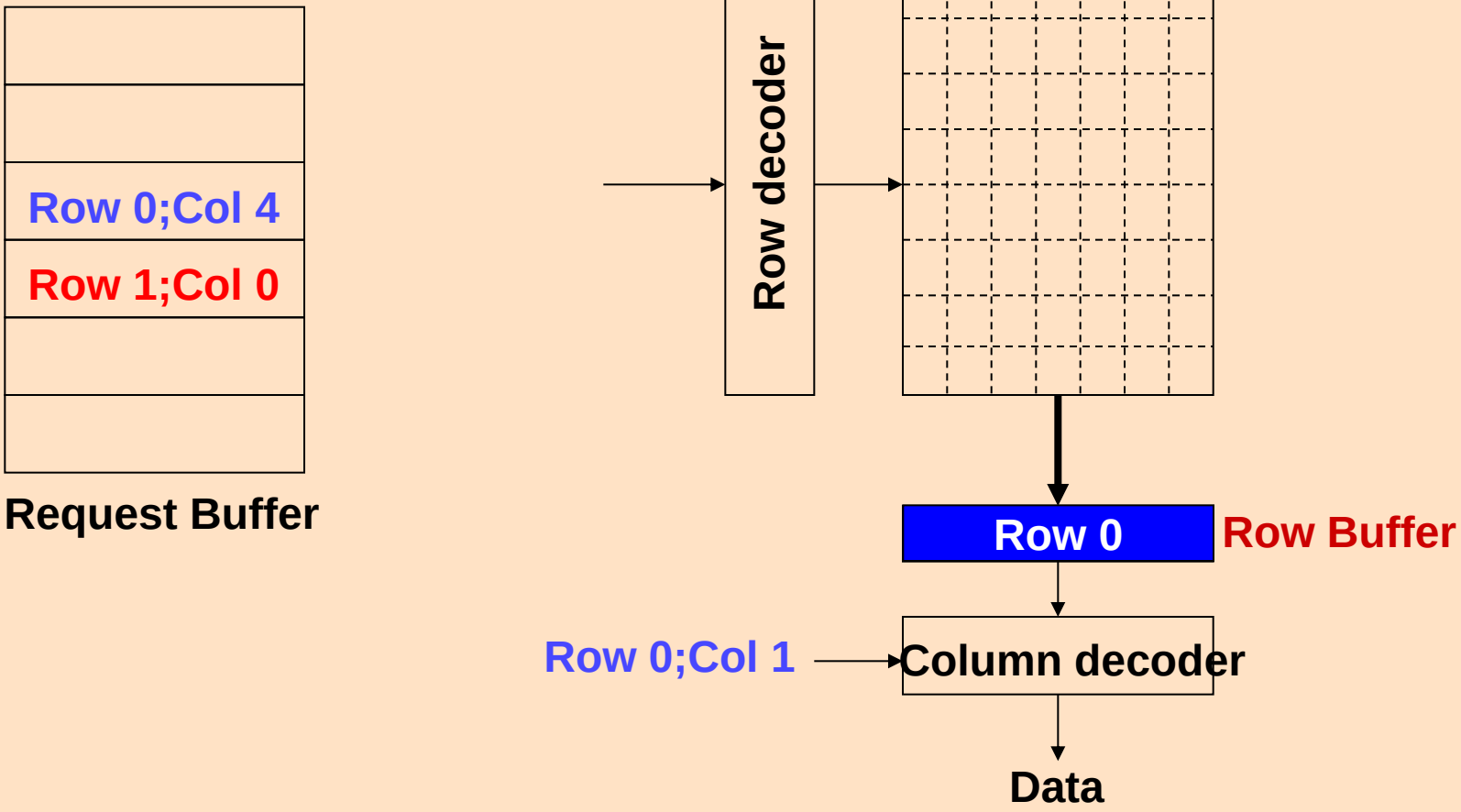
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



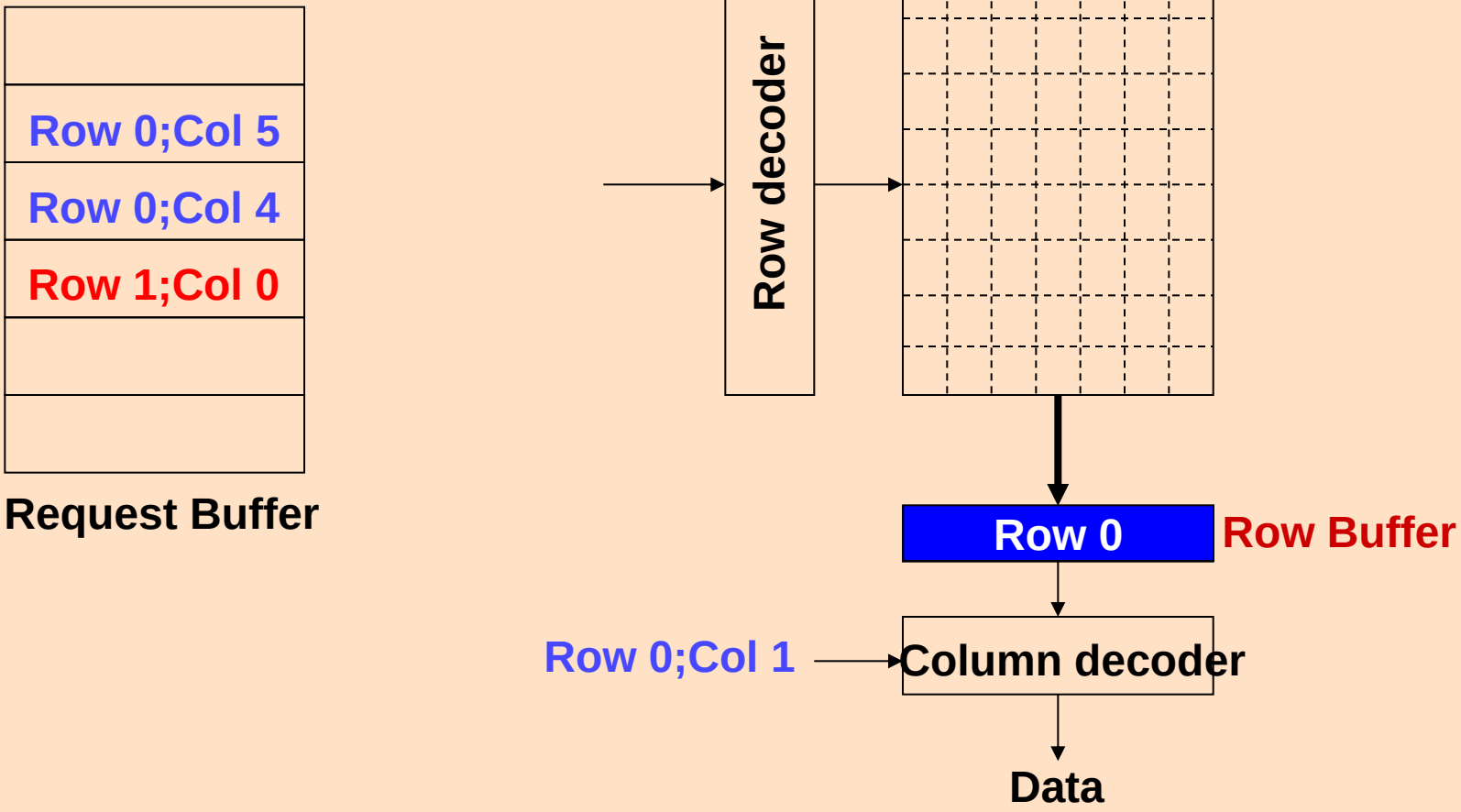
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



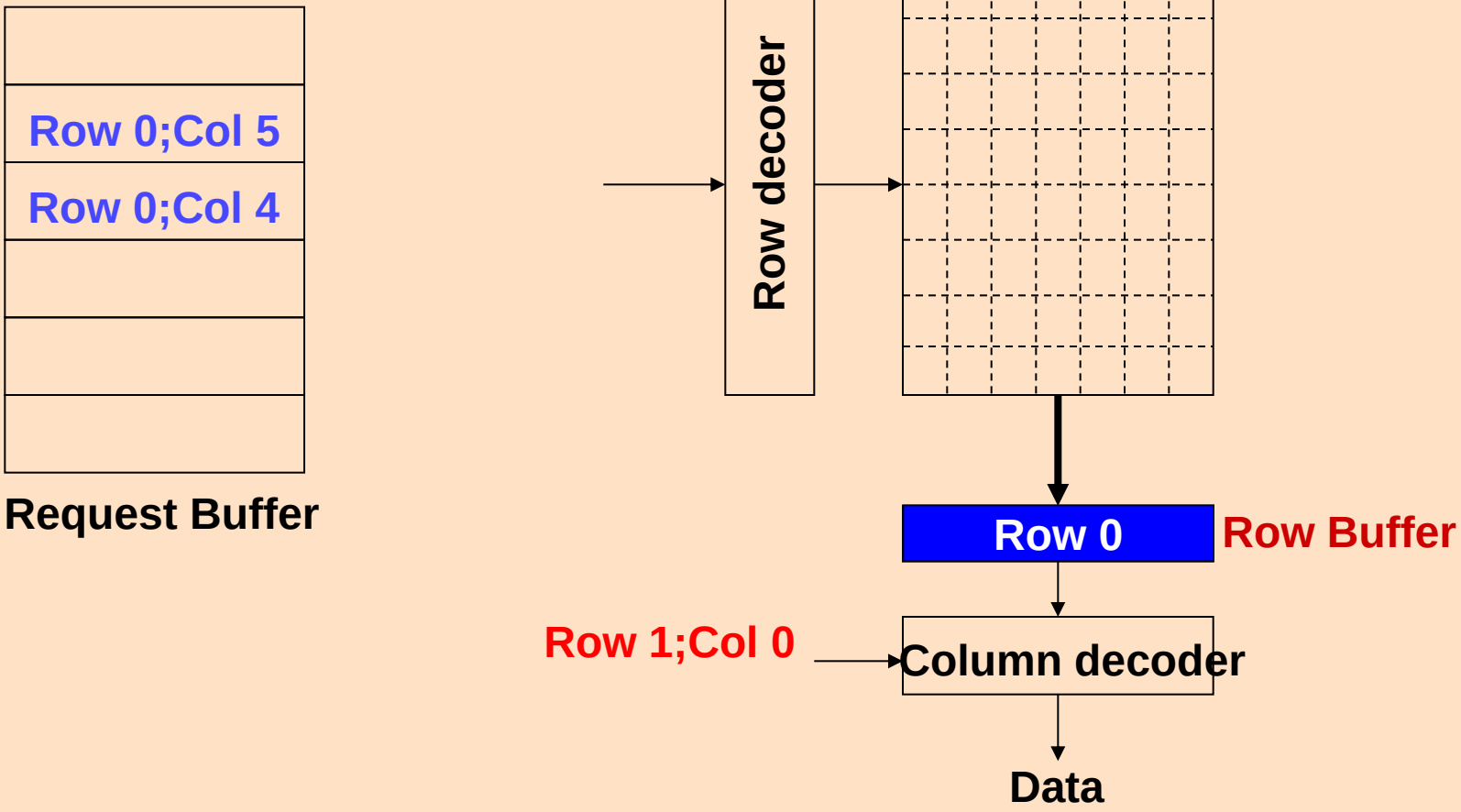
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



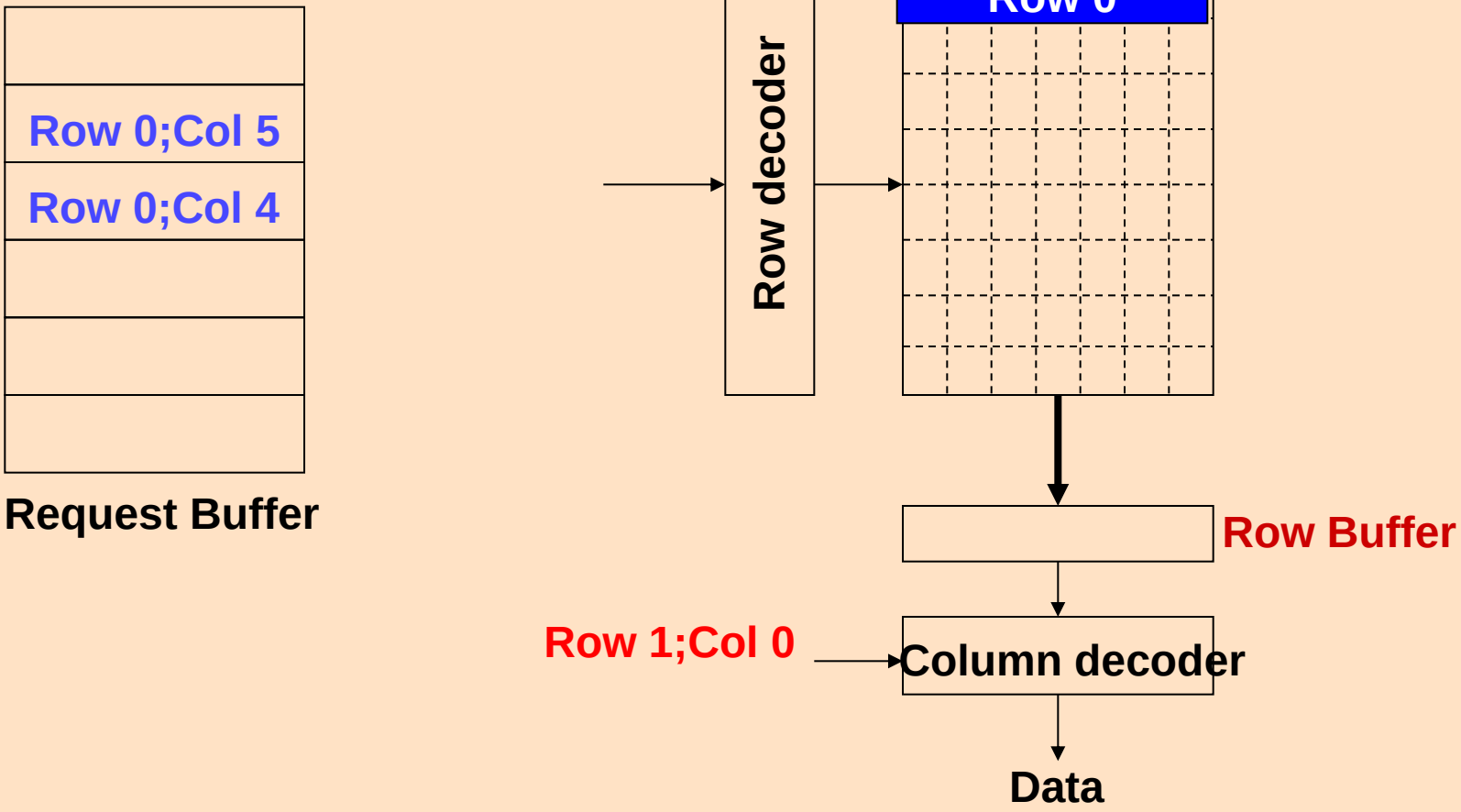
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



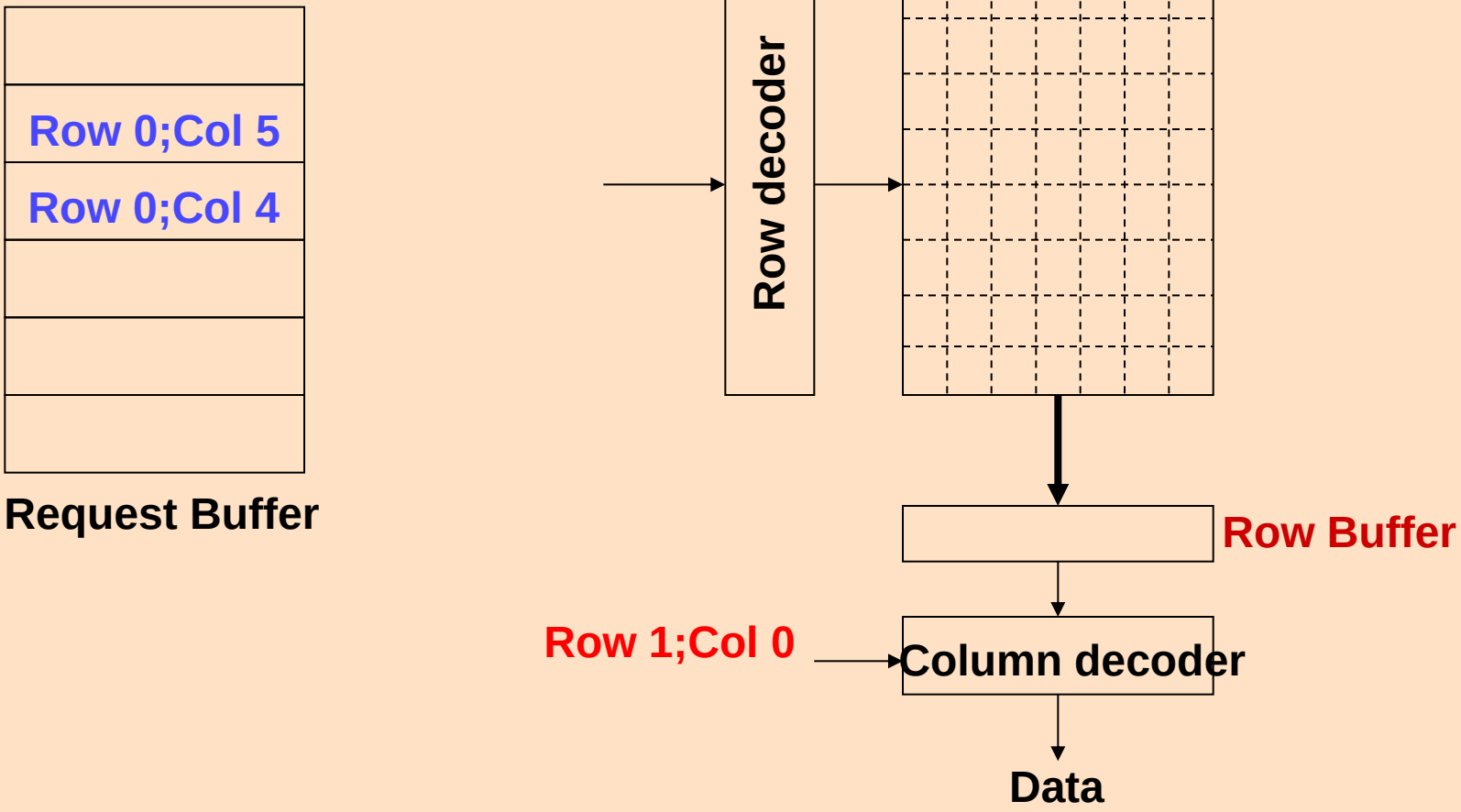
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



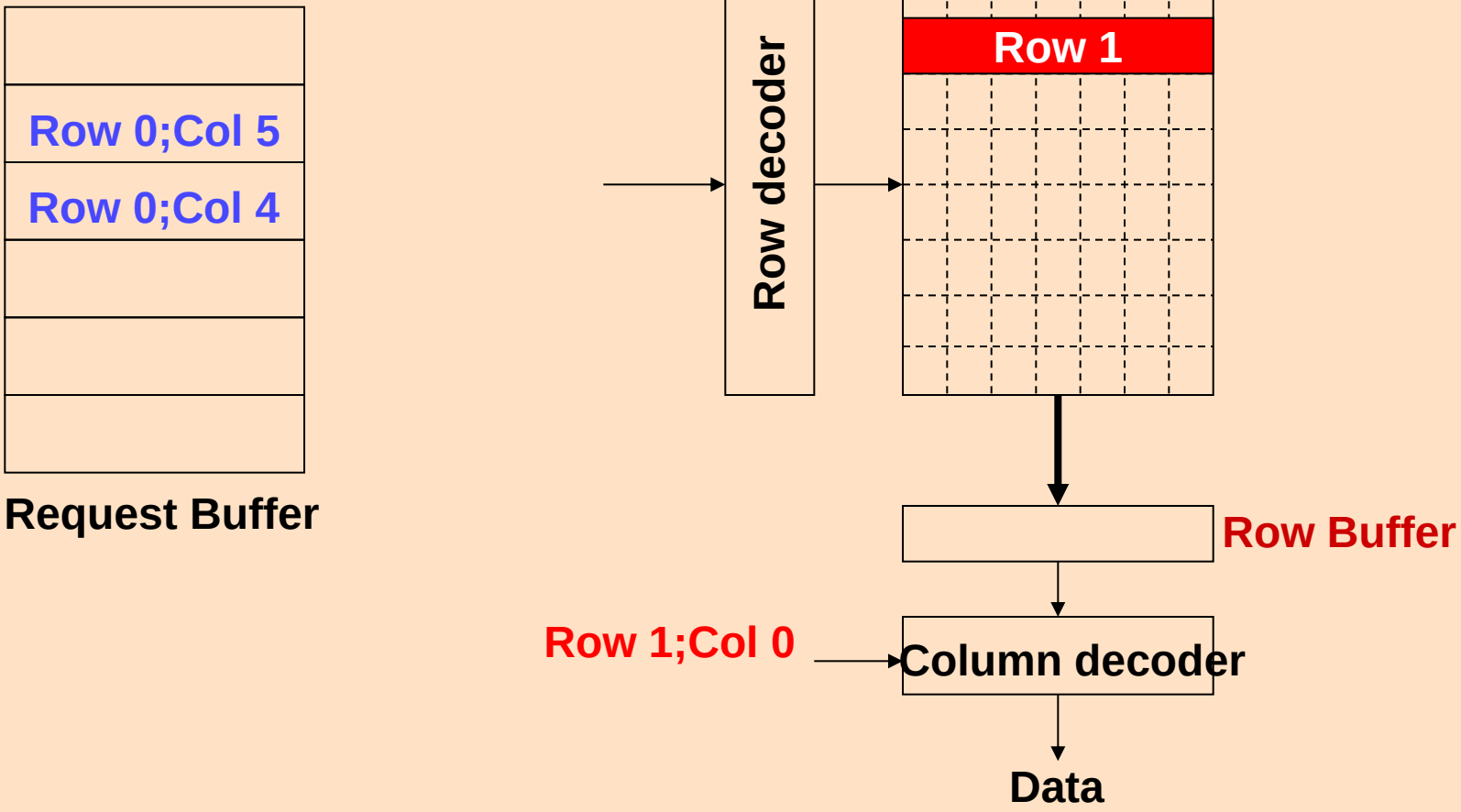
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



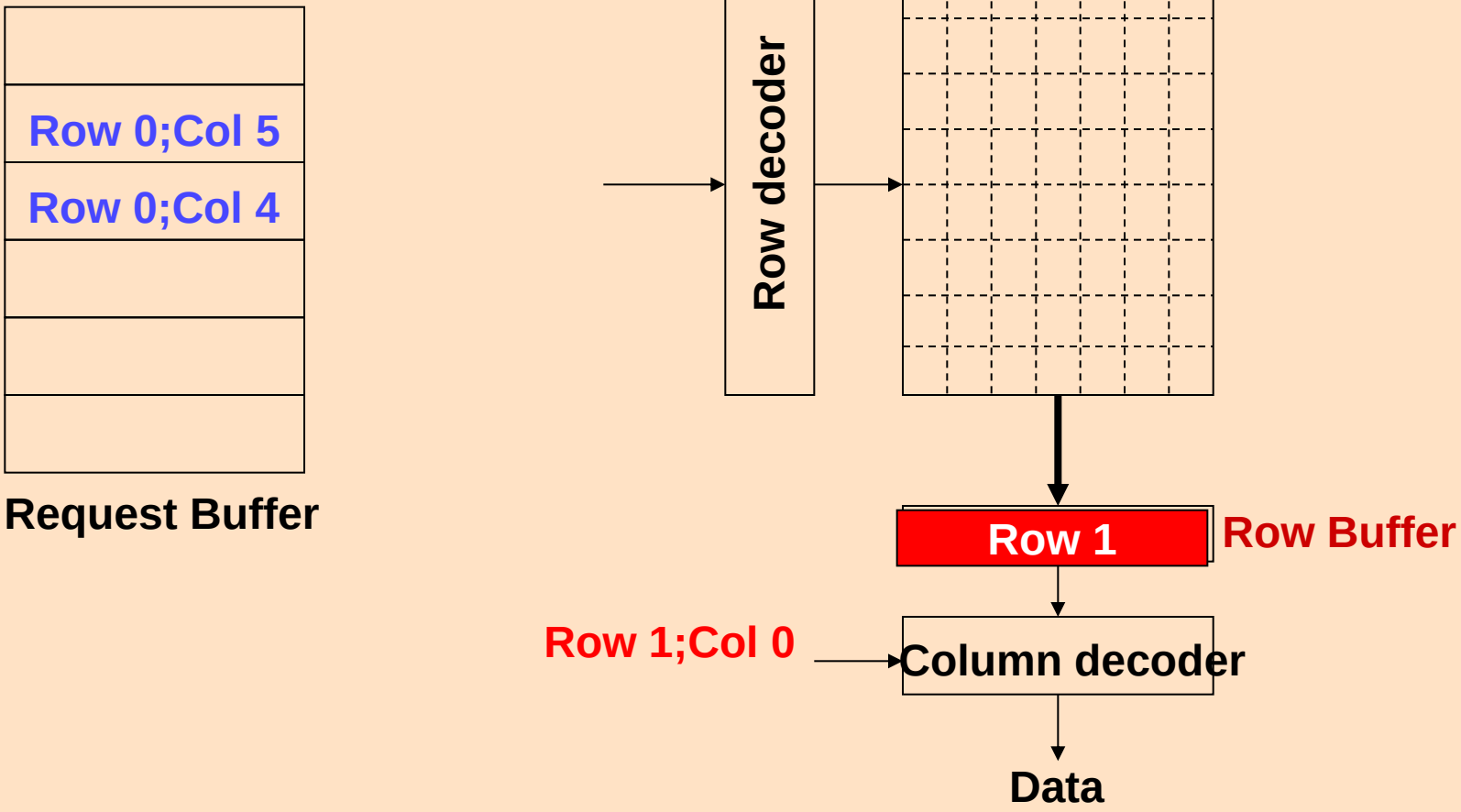
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



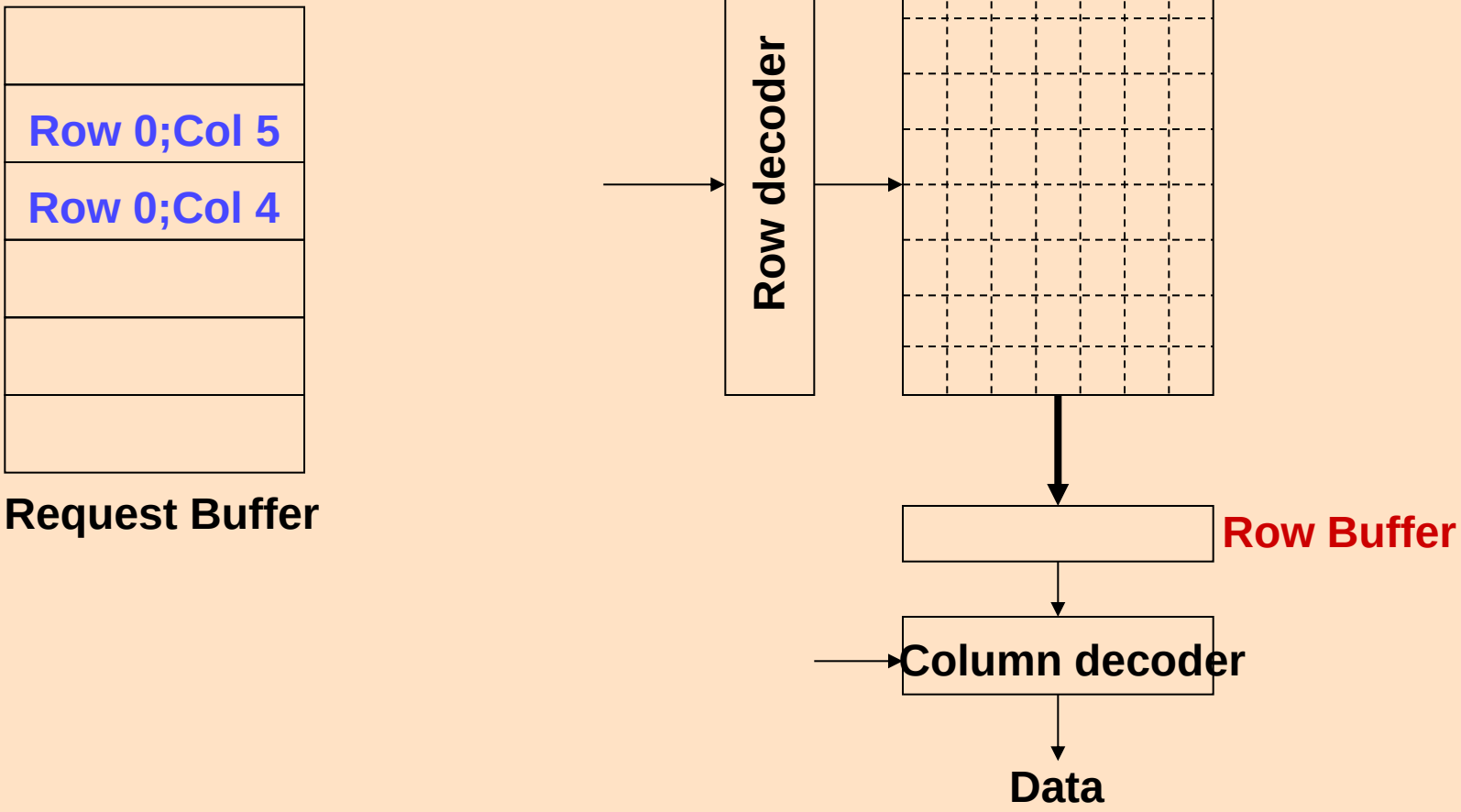
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



Slide Source: Onur Mutlu, CMU

Memory Access Scheduling: FCFS



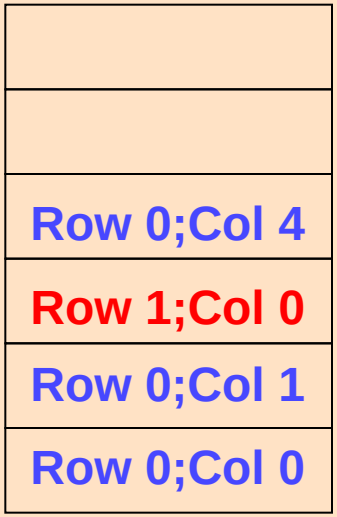
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS

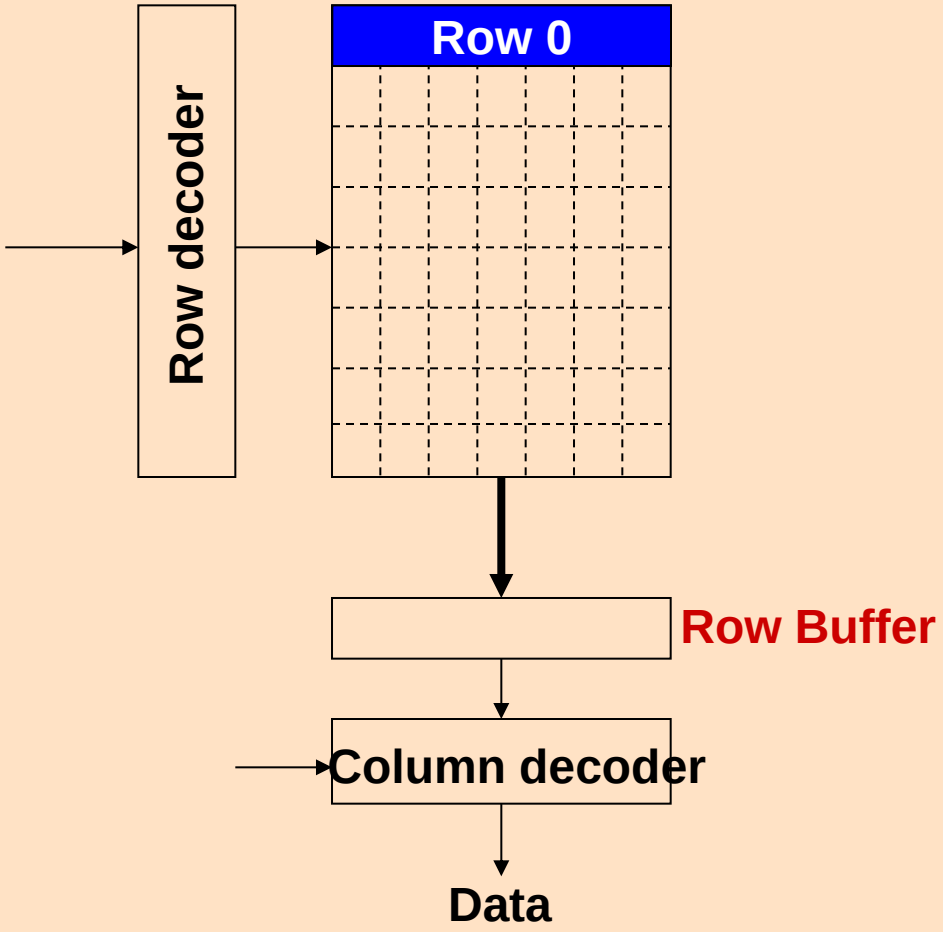
- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer
- Commonly used scheduling policy (FR-FCFS) [Rixner, ISCA'00]
  - (1) Row-hit (column) first: Service row-hit memory accesses first
  - (2) Oldest-first: Then service older accesses first
- This scheduling policy aims to maximize DRAM throughput

Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS

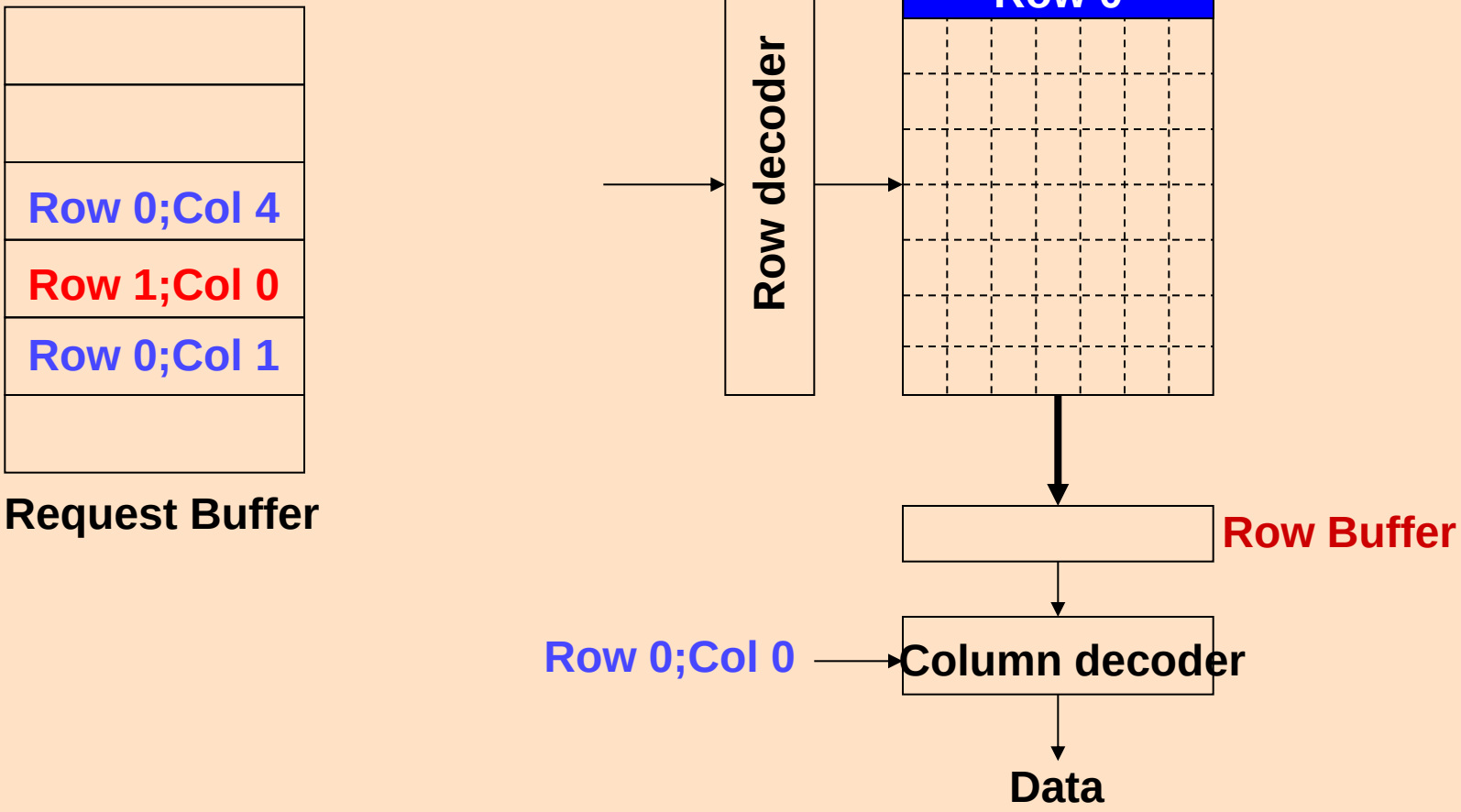


Request Buffer



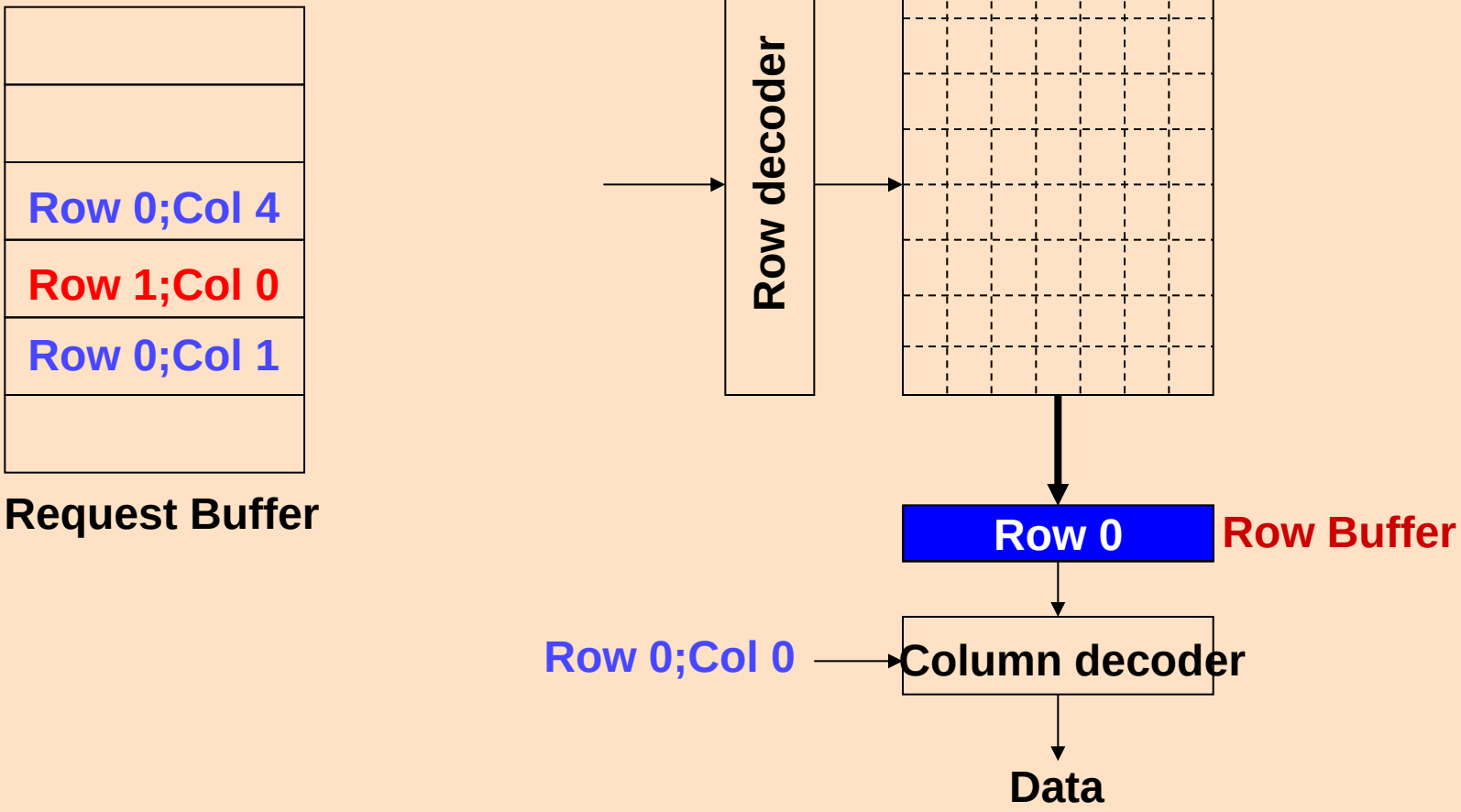
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



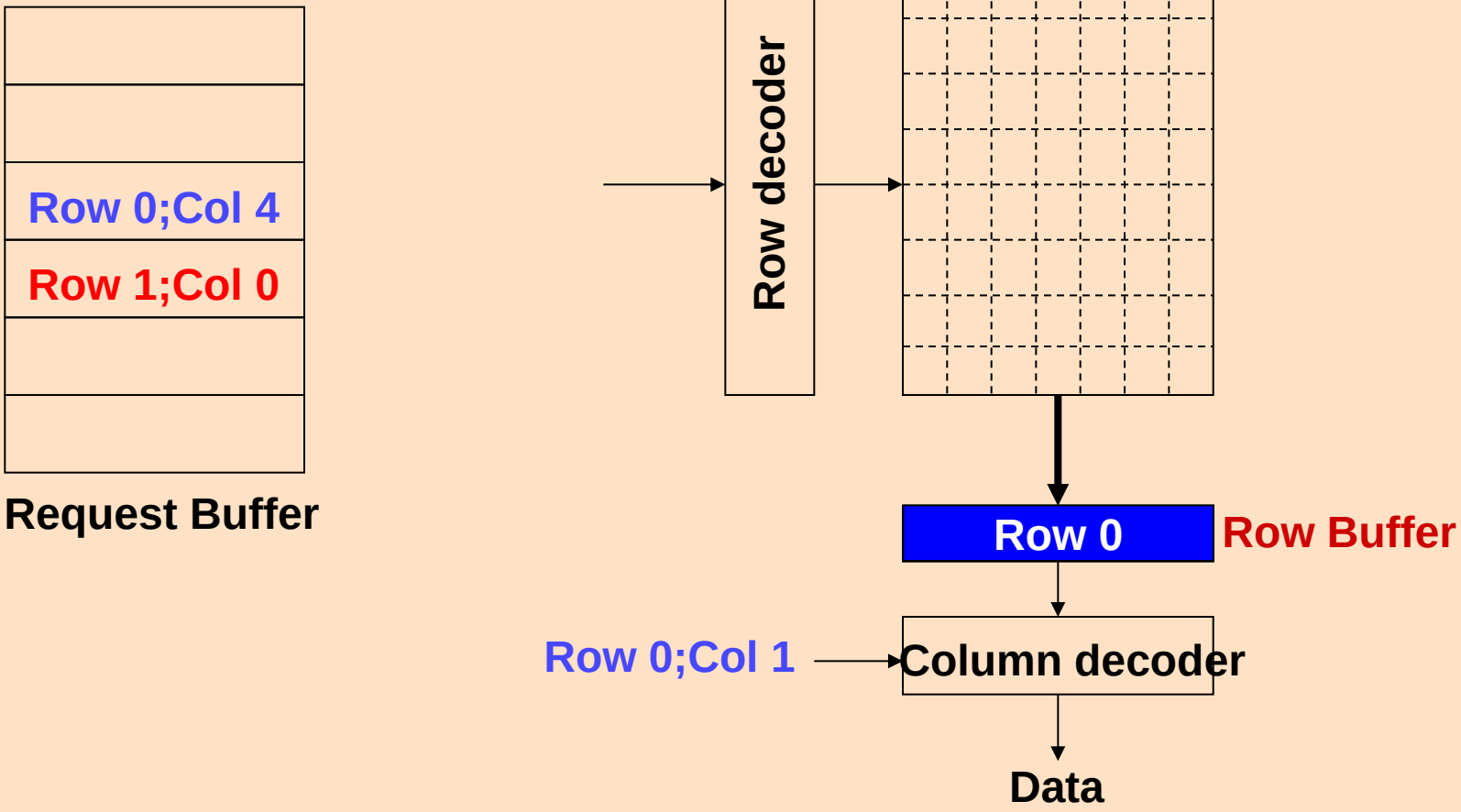
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



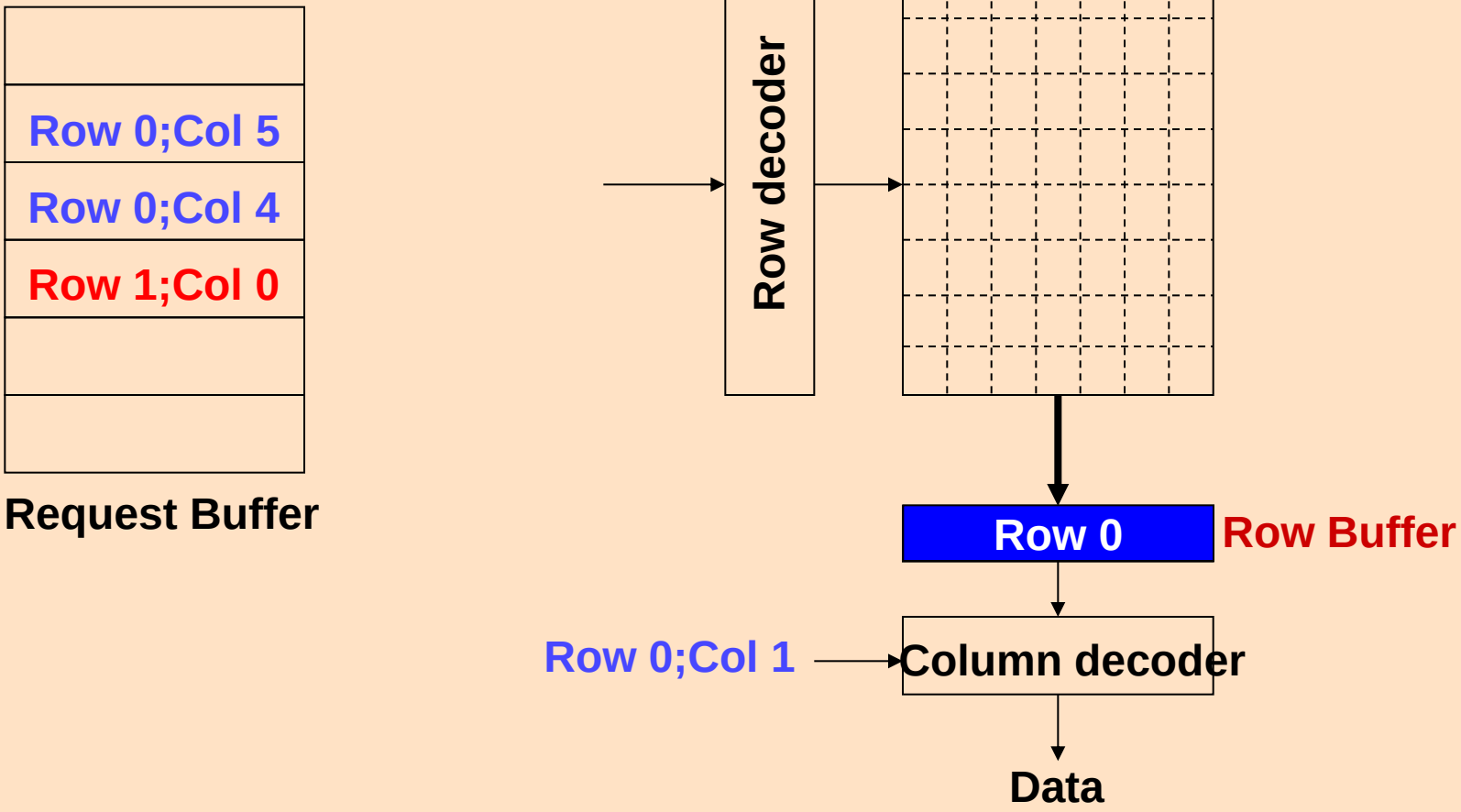
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



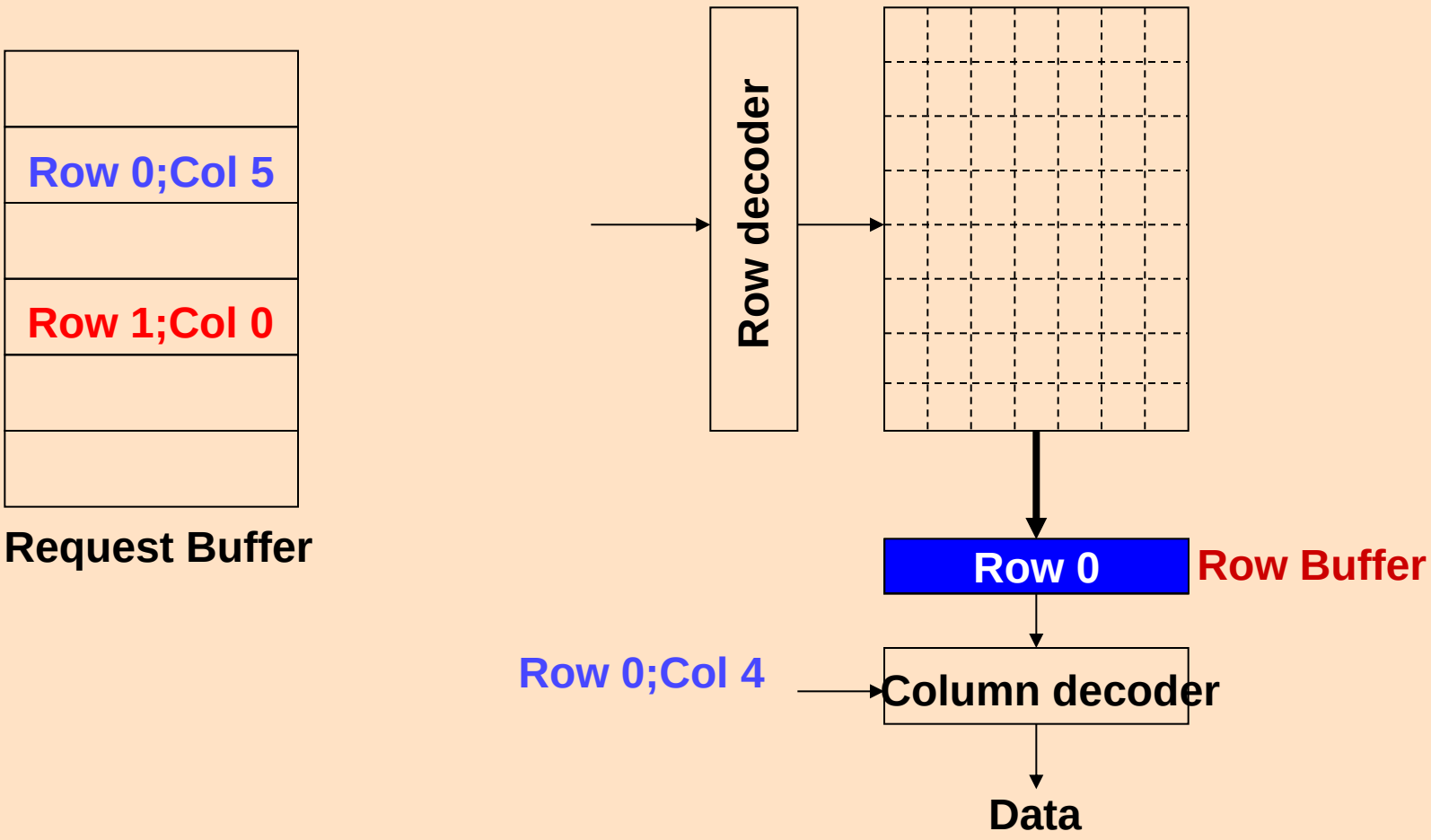
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



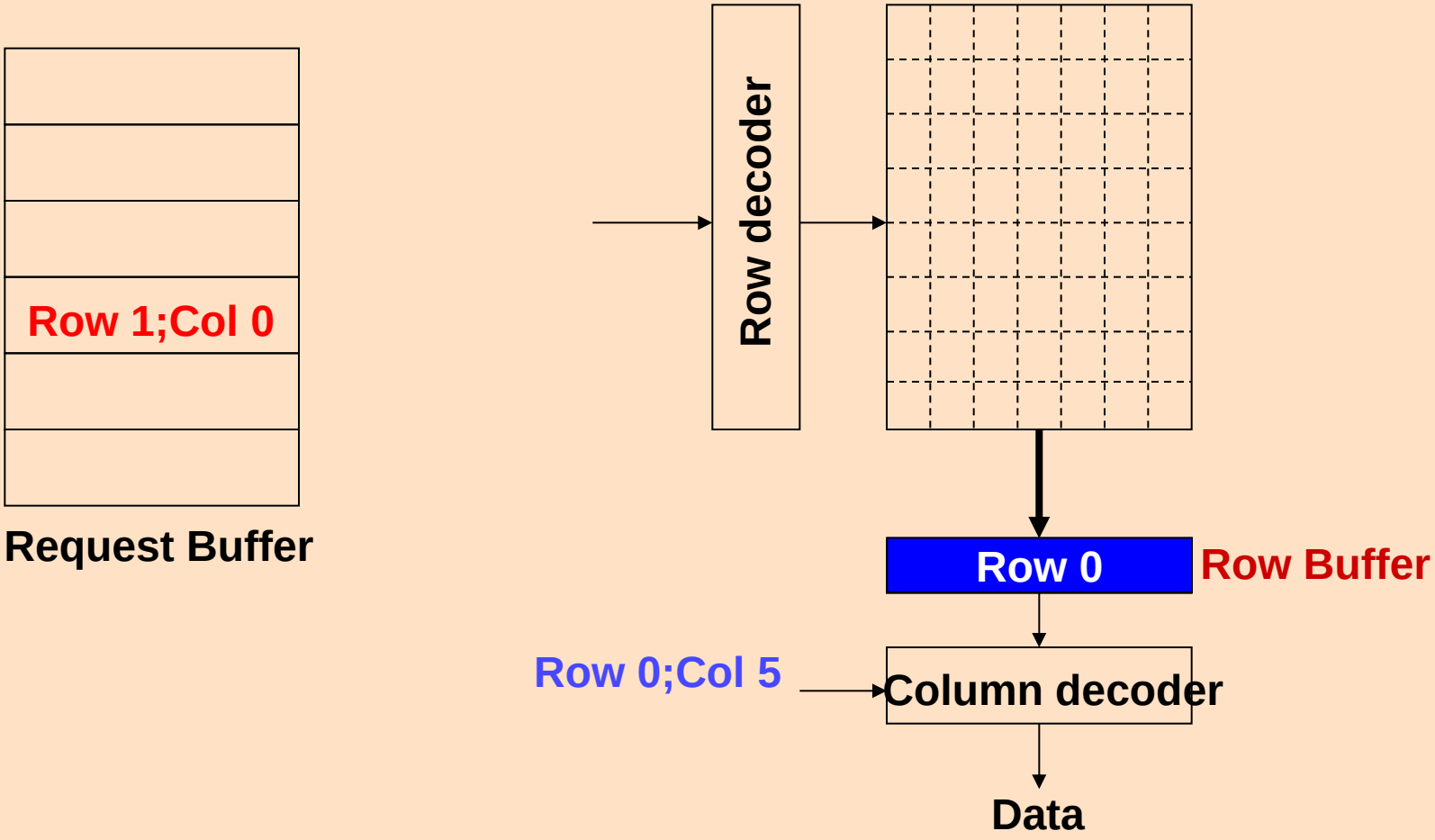
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



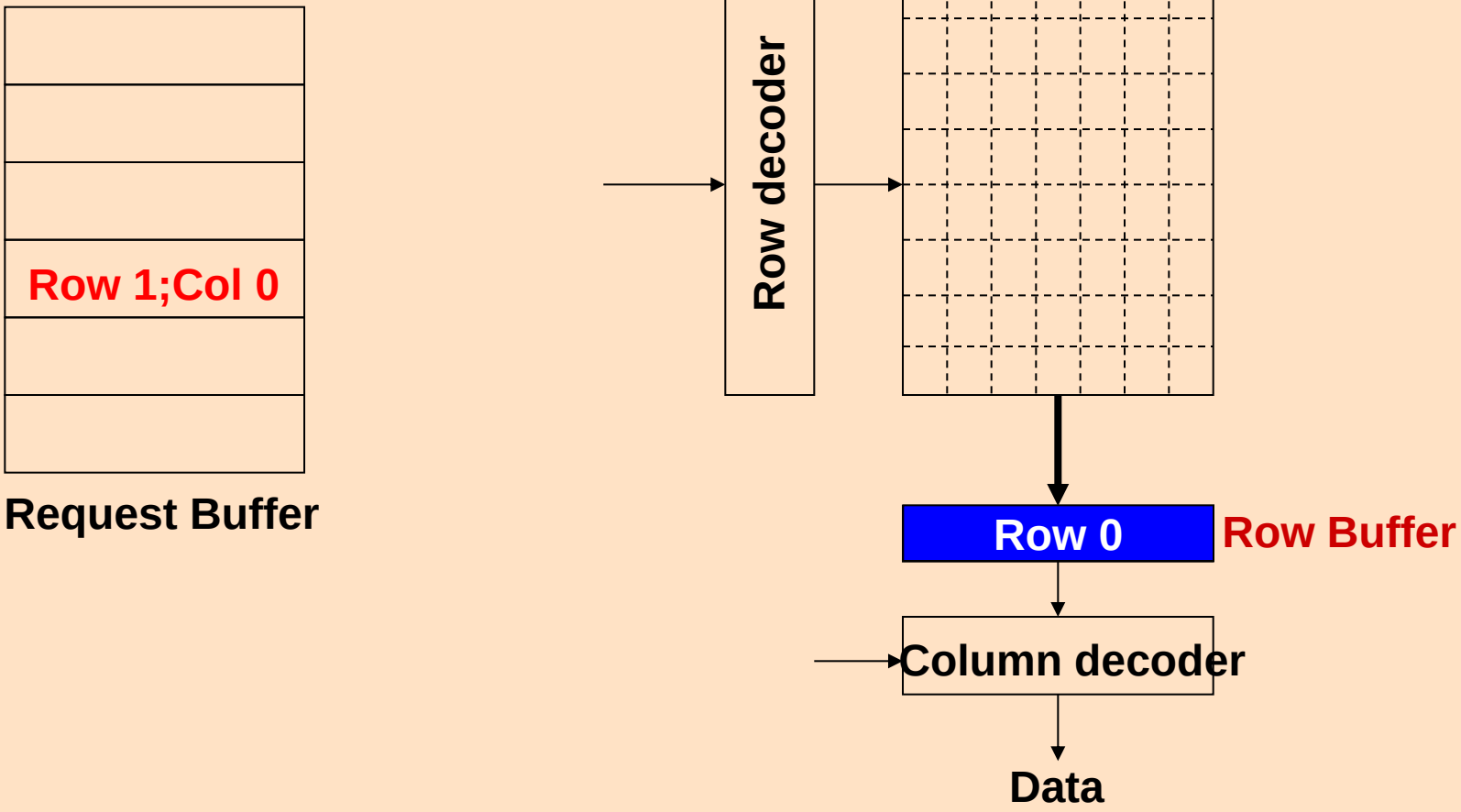
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



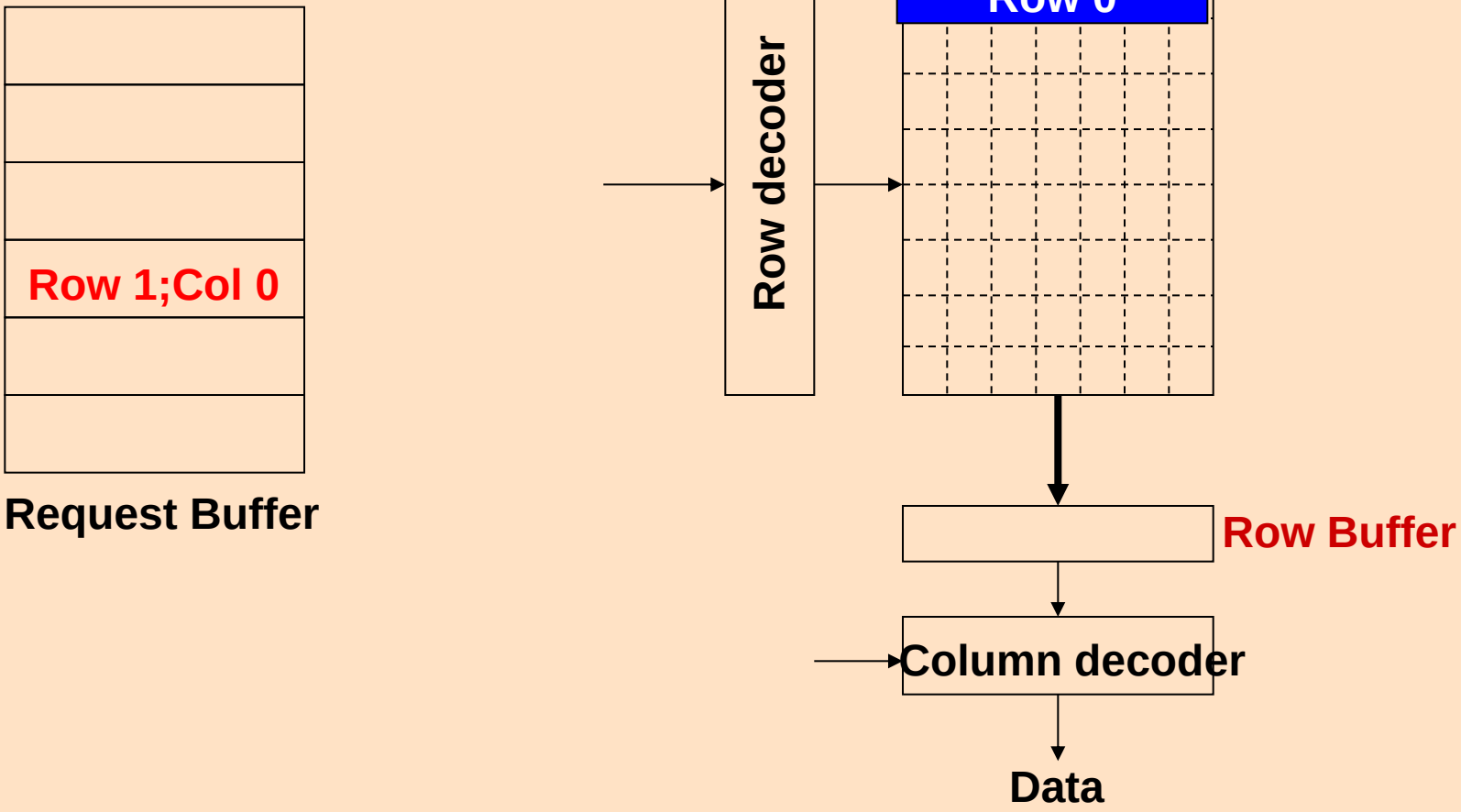
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



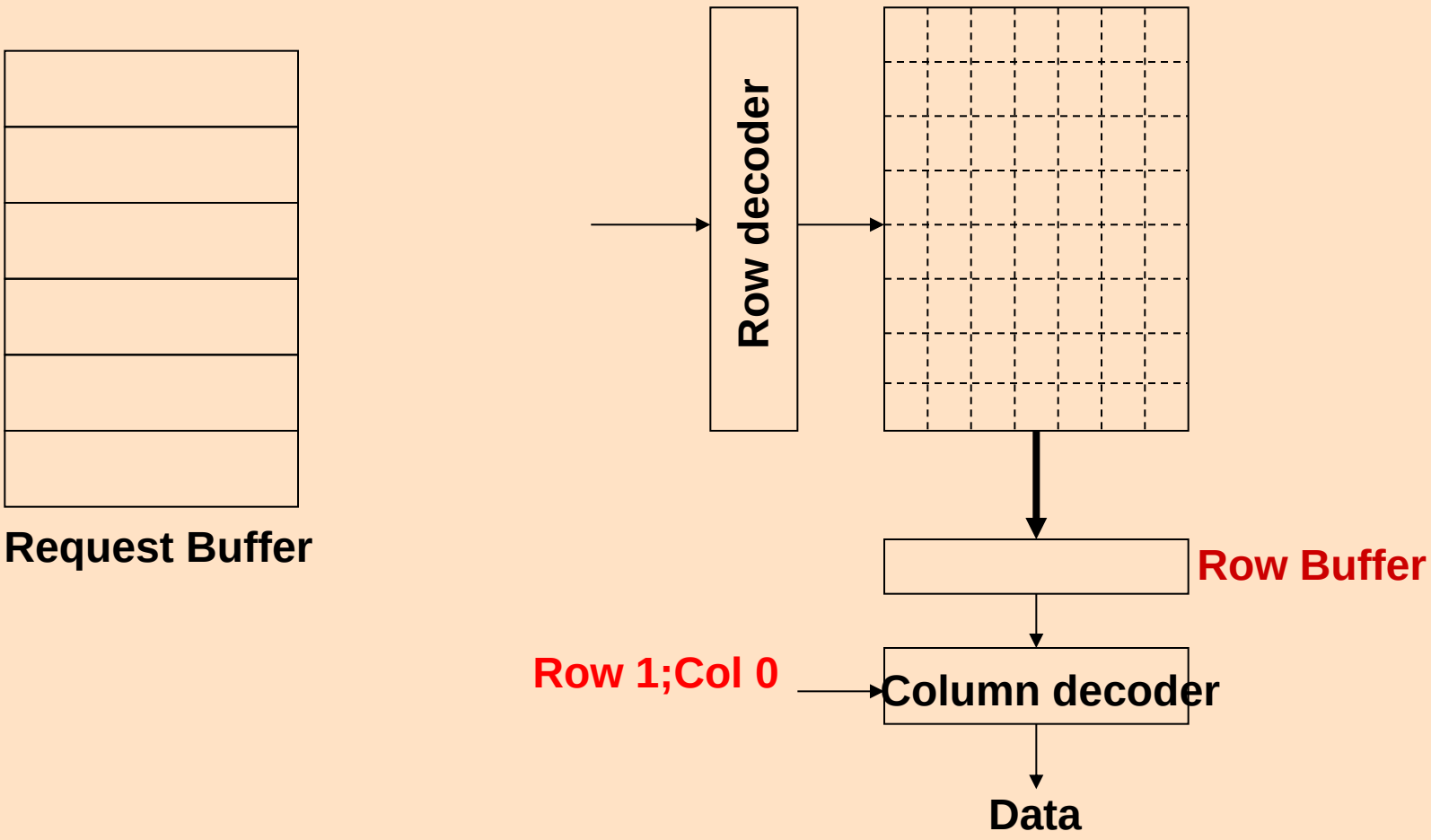
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



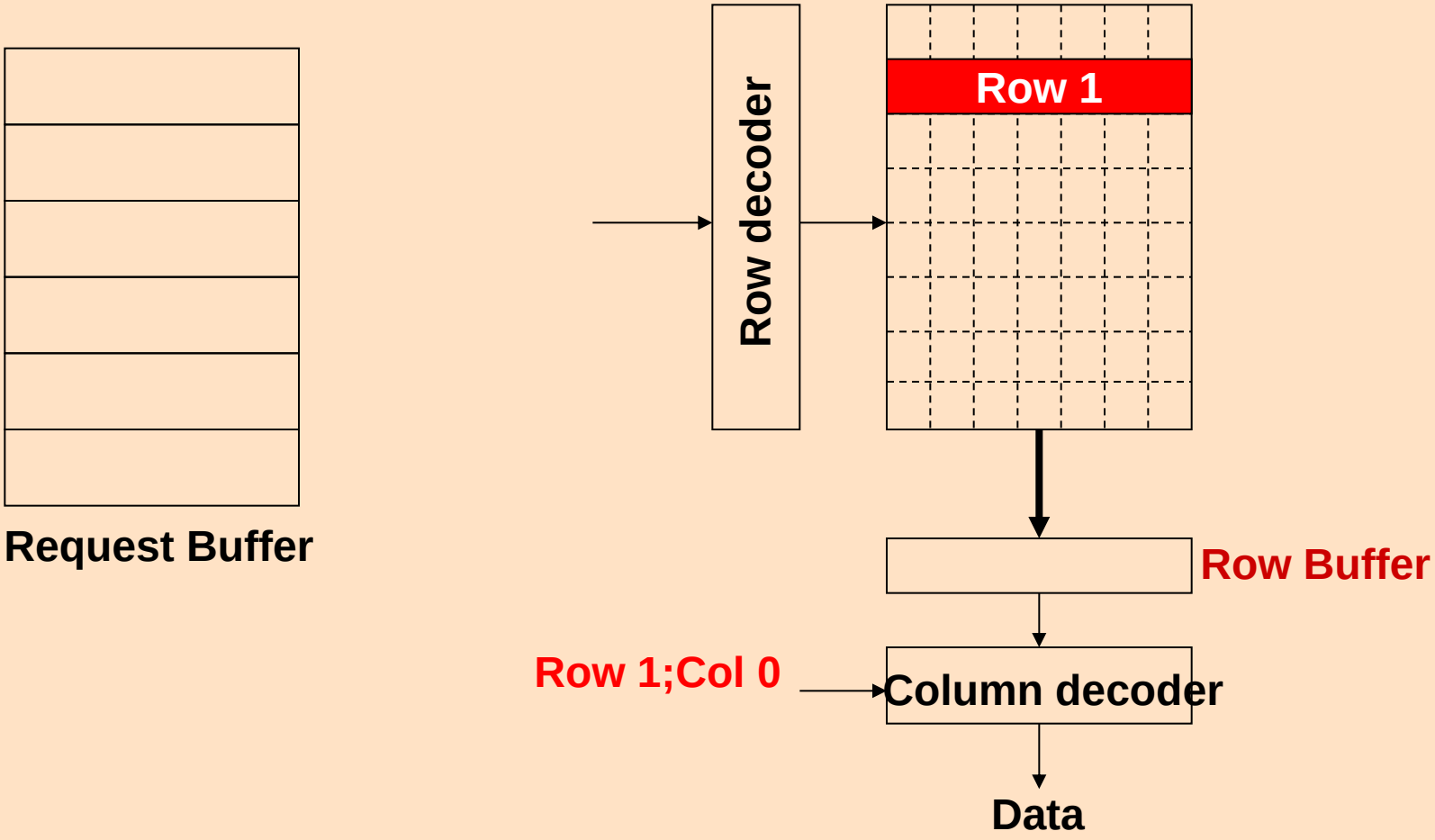
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



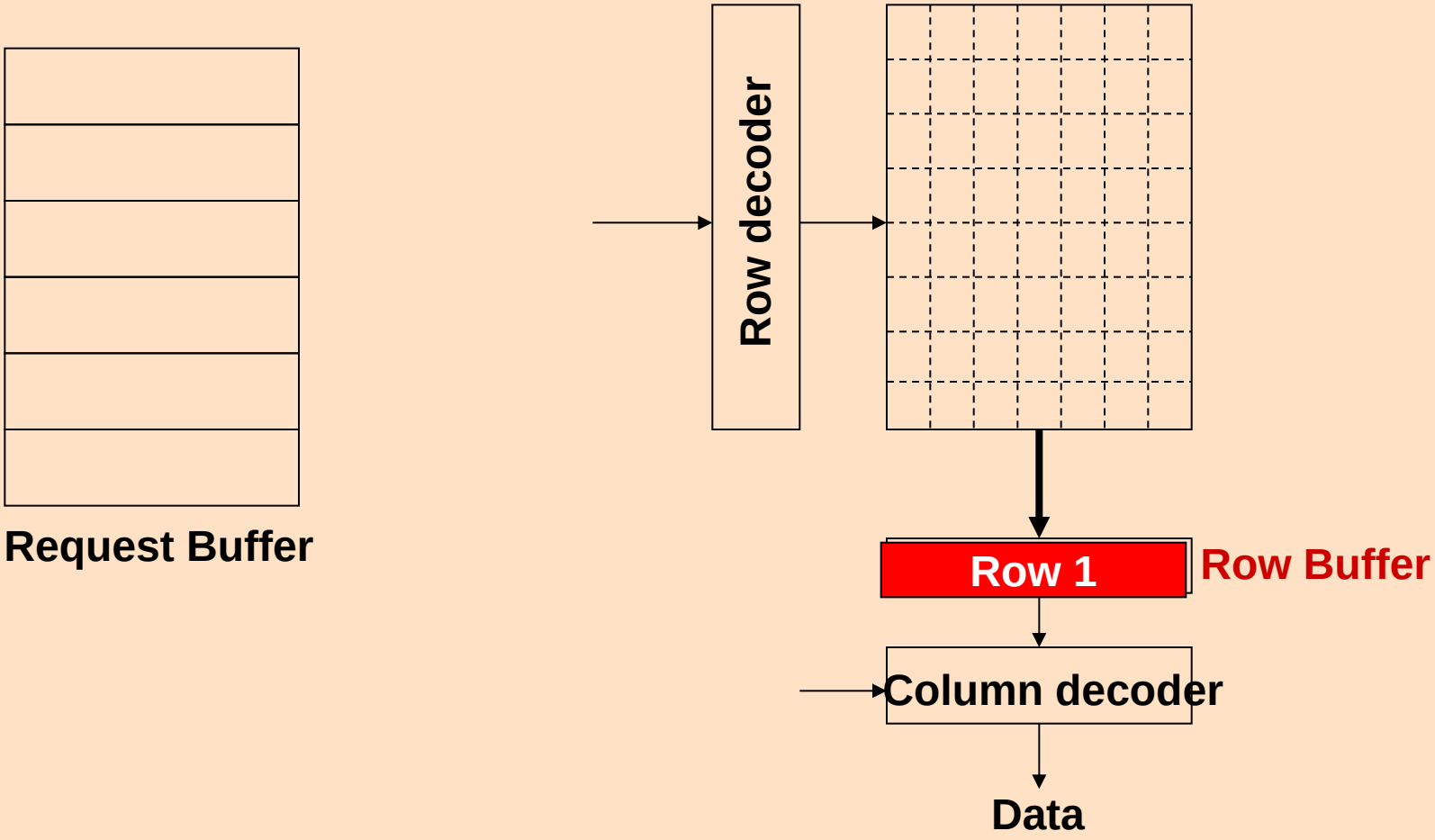
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



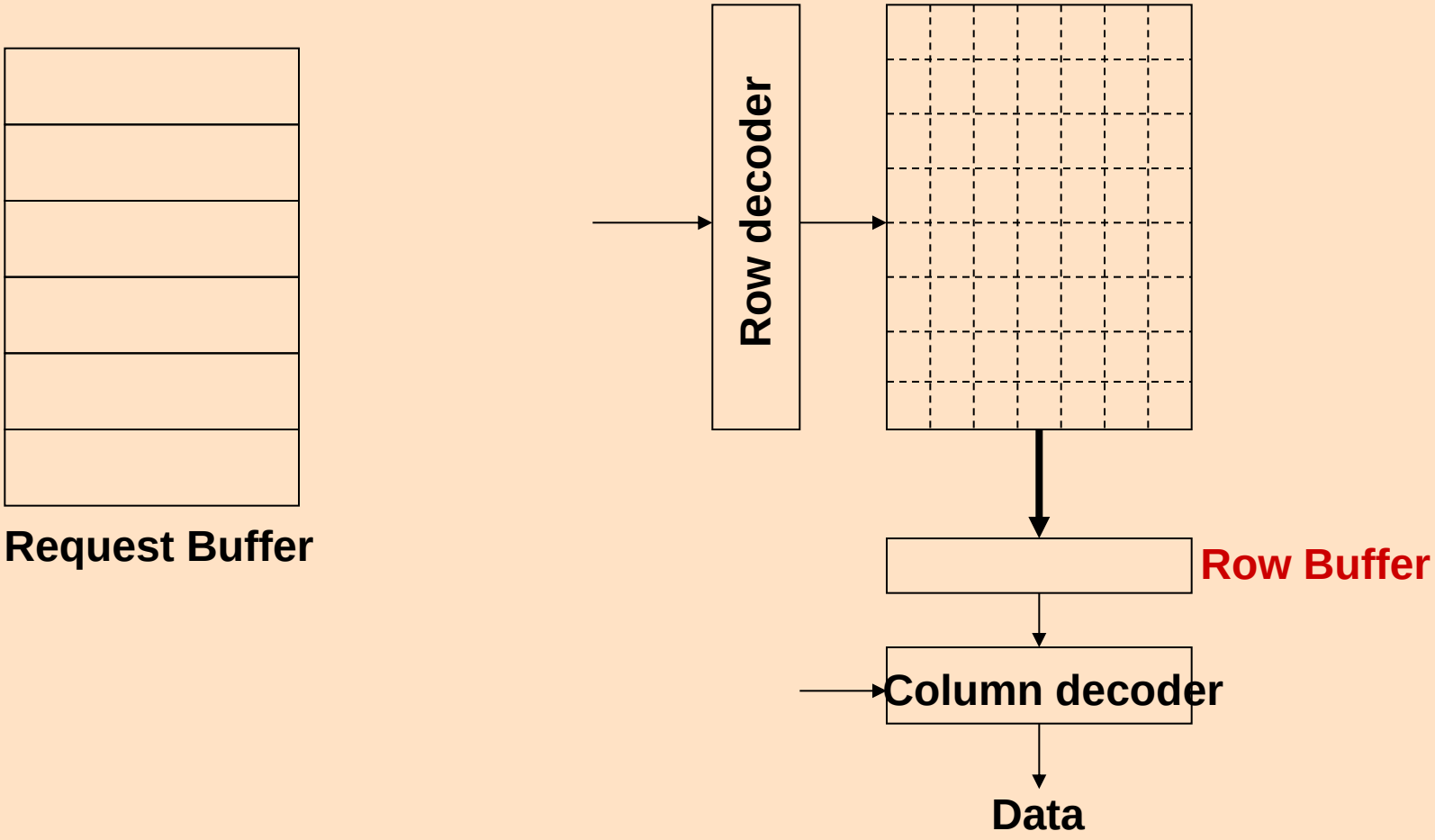
Slide Source: Onur Mutlu, CMU

Memory Access Scheduling : FR-FCFS



Slide Source: Onur Mutlu, CMU

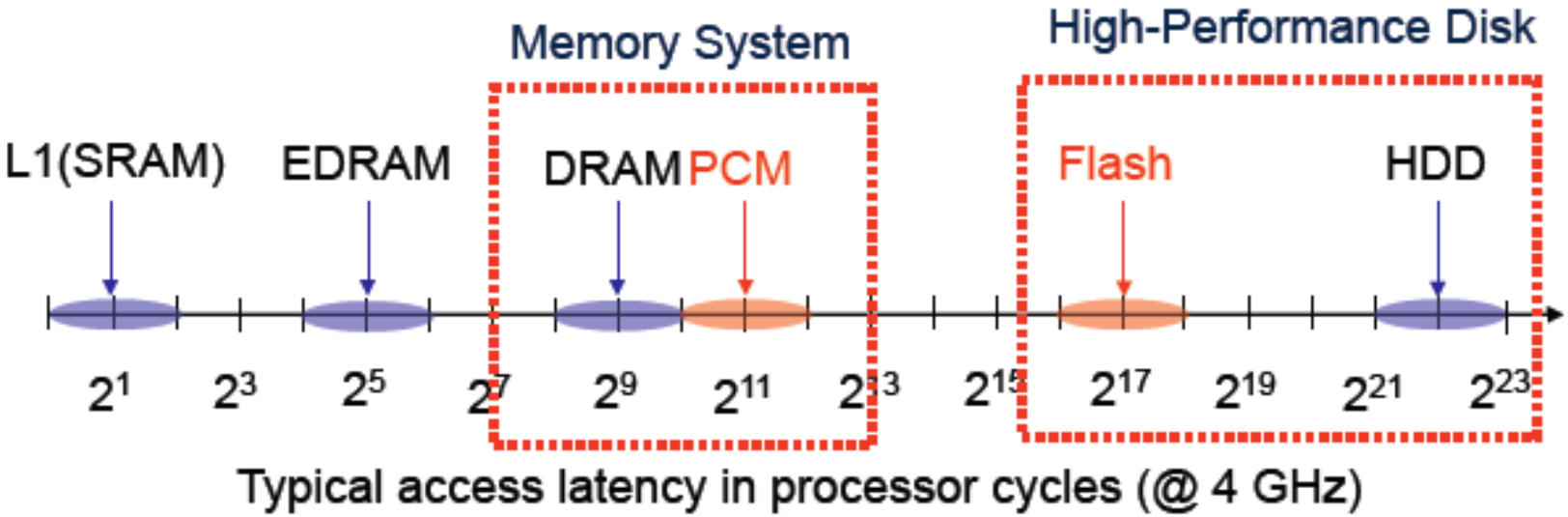
Memory Access Scheduling : FR-FCFS



Slide Source: Onur Mutlu, CMU

Emerging Memory Technology

- Non-Volatile Memory technology
  - Phase Change Memory (PCM), Magnetic RAM (MRAM), Resistive RAM (RRAM), Spin Torque Transfer RAM (STT-RAM), …



Slide Source: Moin Quereshi, Georgia Tech.

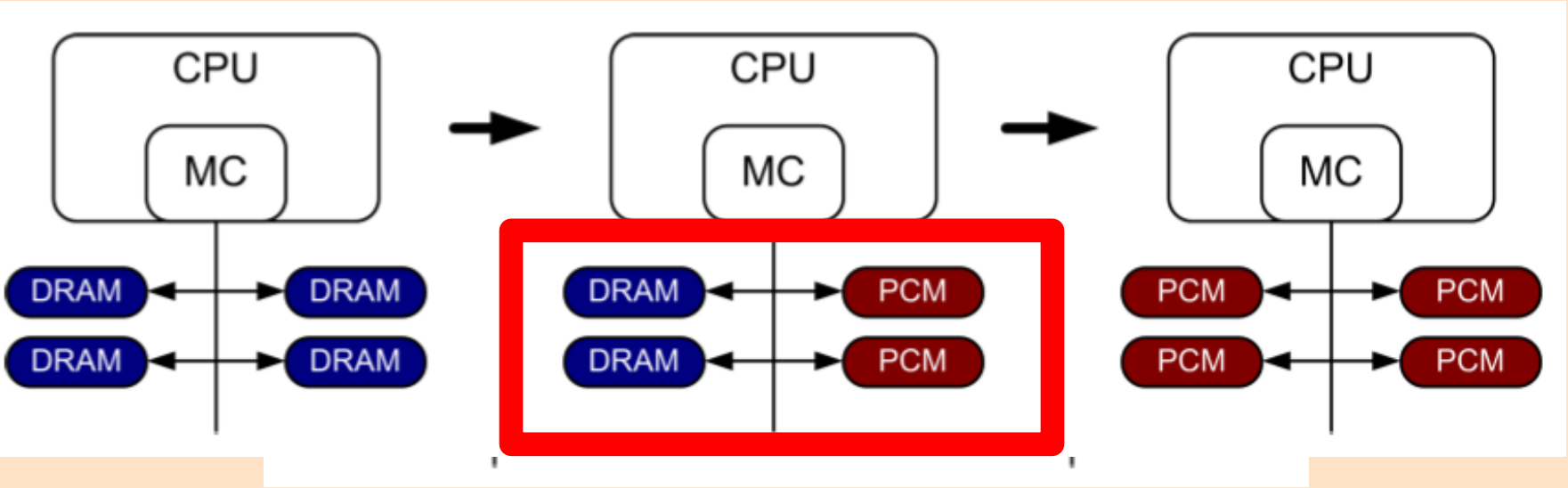
Emerging Memory Technology

- Phase Change Memory
  - Data stored by changing phase of special material
  - Data read by detecting material's resistance
  - Phase change material (chalcogenide glass) exists in two states:
    1. Amorphous: high resistivity – reset state or 0
    2. Crystalline: low resistivity – set state or 1
  - Non-volatality and low idle power (no refresh)
  - Expected to scale (to 9nm), denser than DRAM, and can store multiple bits/cell
  - Higher Write latency and write-energy
  - Endurance issues (cell dies after 10^8 writes)

Slide Source: Onur Mutlu, CMU

DRAM – PCM Hybrid Memory

- PCM-based (main) memory be organized?

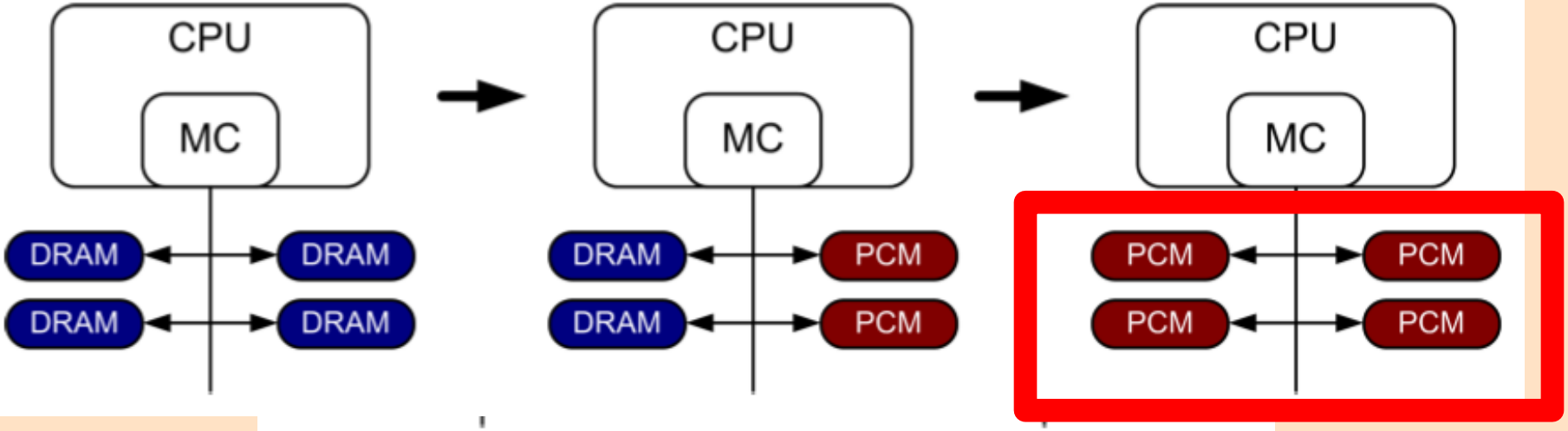


- Hybrid PCM+DRAM
  - How to partition/migrate data between PCM and DRAM
  - Is DRAM a cache for PCM or part of main memory?
  - How to design the hardware and software

Slide Source: Onur Mutlu, CMU

PCM-based Main Memory

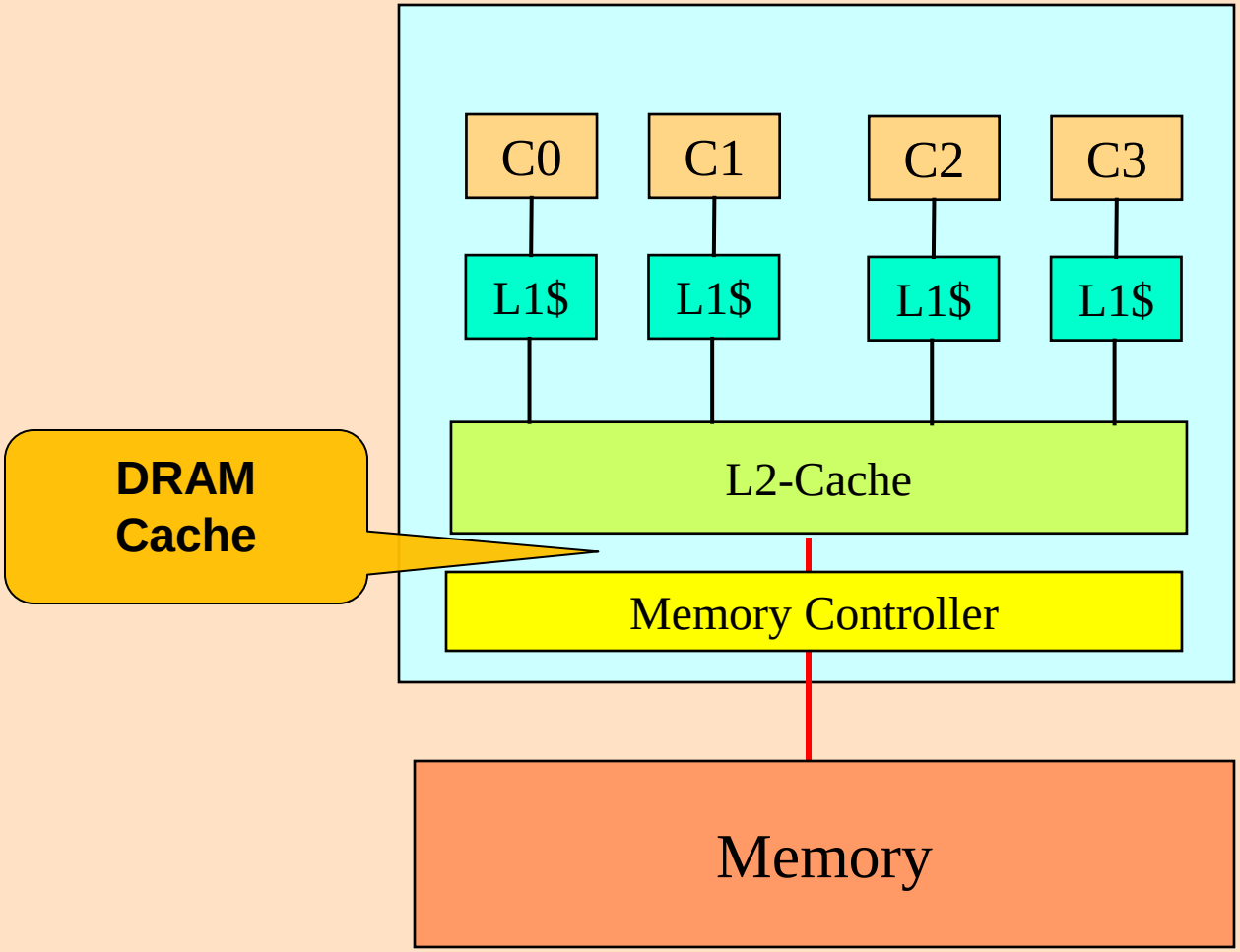
- How should PCM-based (main) memory be organized?



- Pure PCM main memory [Lee et al., ISCA'09, Top Picks'10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings

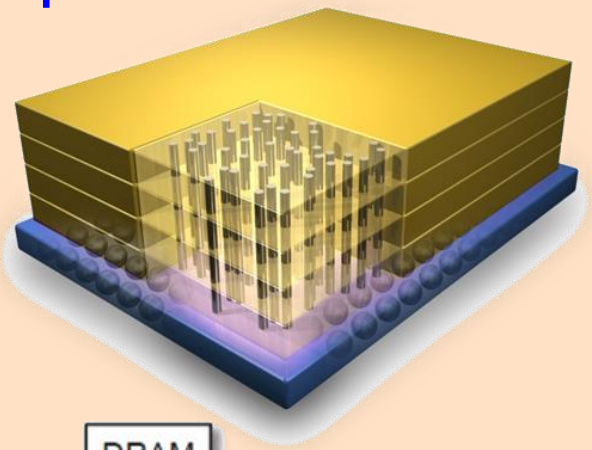
Slide Source: Onur Mutlu, CMU

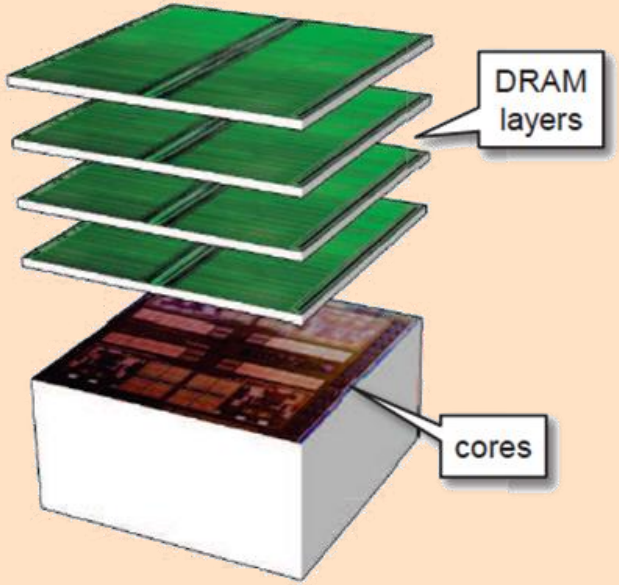
Expanding the Multicore Memory Hierarchy



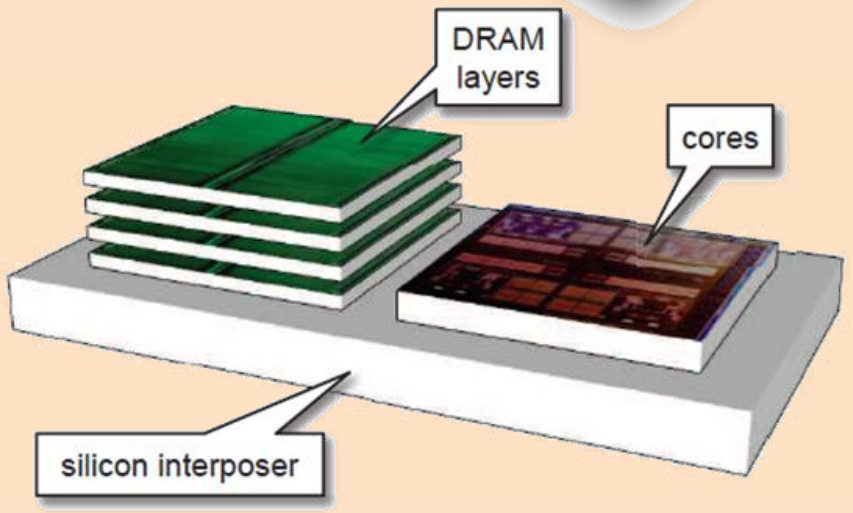
Stacked DRAM

- DRAM vertically stacked over the processor die.
- Stacked DRAMs offer
  - High bandwidth
  - Large capacity
  - Same or slightly lower latency.





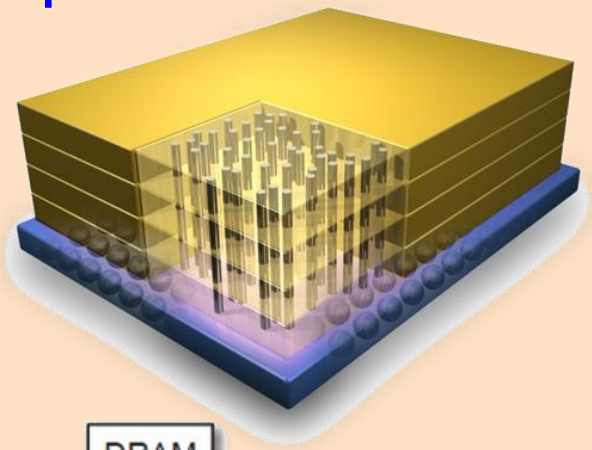
3-D Stacked DRAM

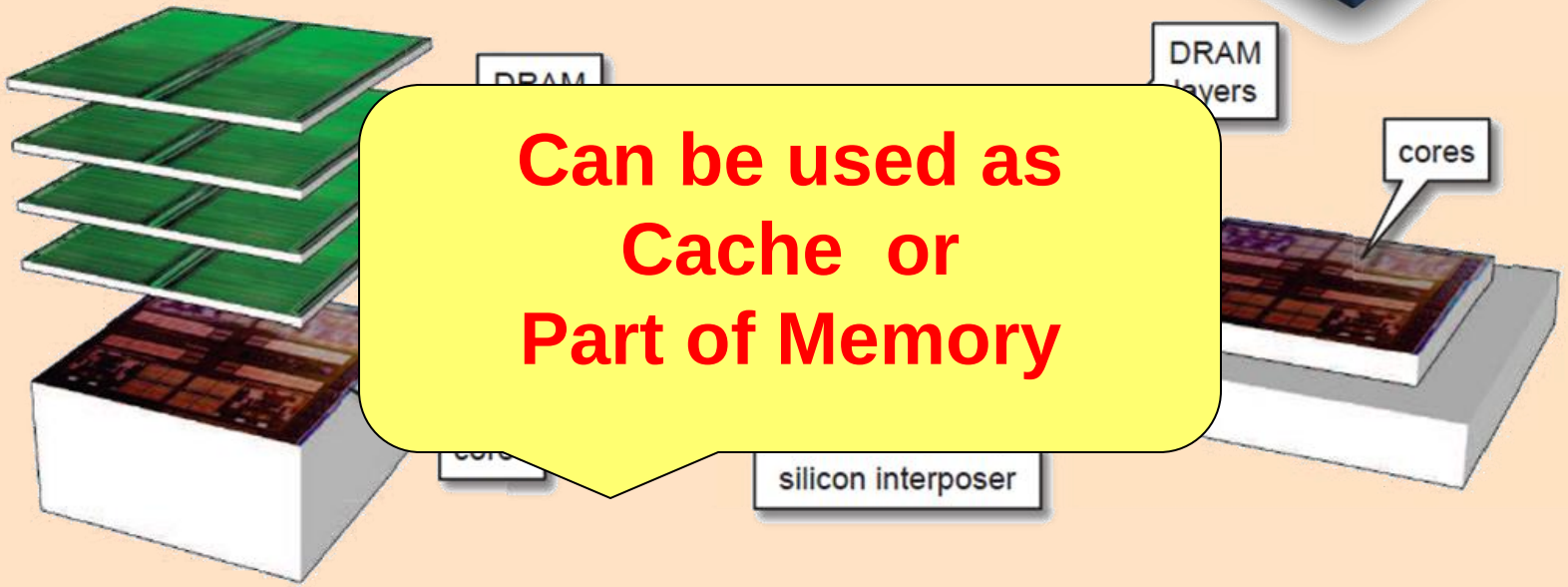


2.5-D Stacked DRAM

Stacked DRAM

- DRAM vertically stacked over the processor die.
- Stacked DRAMs offer
  - High bandwidth
  - Large capacity
  - Same or slightly lower latency.



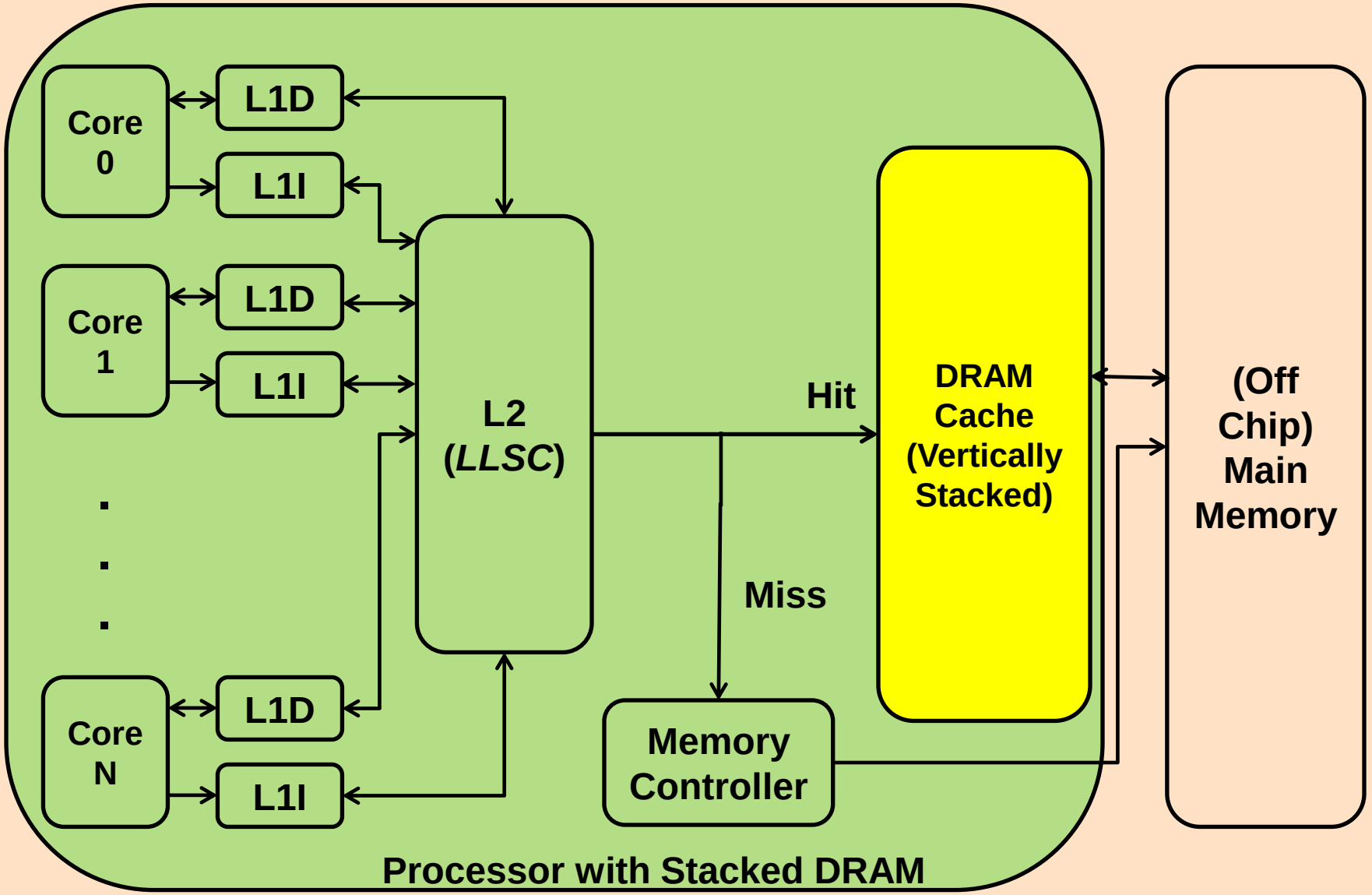


Can be used as Cache or Part of Memory

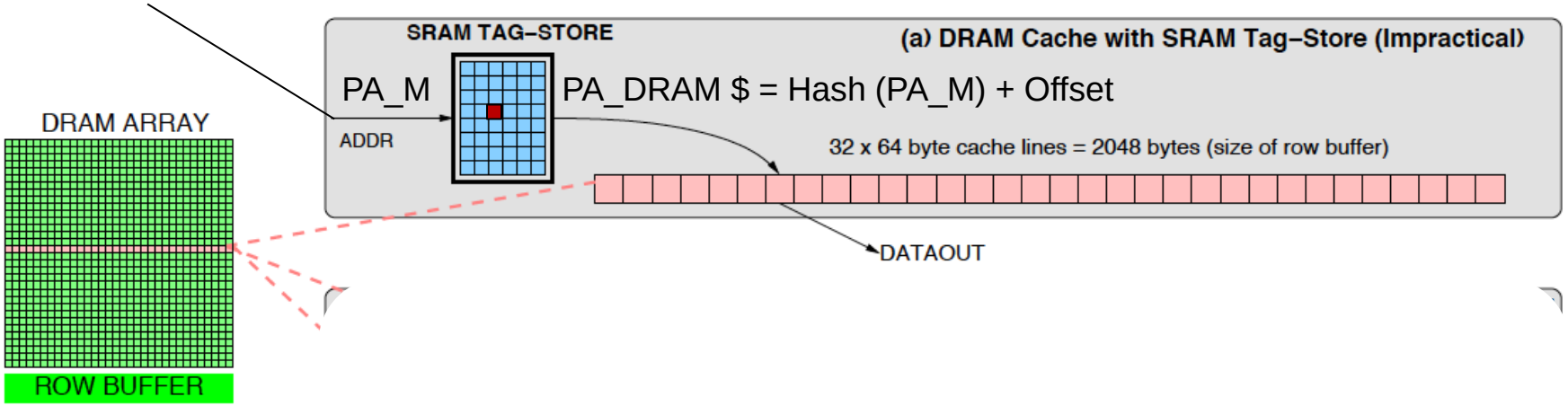
3-D Stacked DRAM

2.5-D Stacked DRAM

Multicore With DRAM Cache



Overview of Different Designs



Problems in Architecting Large Caches

- Organizing at cache line granularity (64 B) reduces wasted space and wasted bandwidth
- **Problem:** Cache of hundreds of MB needs tag-store of tens of MB
- E.g. 256MB DRAM cache needs ~20MB tag store (5 bytes/line)
- But big blocks have their own issues
  - Wasted off-chip bandwidth
  - Wasted cache space

Problems in Architecting Large Caches

- Organizing at cache line granularity (64 B) reduces wasted space and wasted bandwidth
- **Problem:** Cache of hundreds of MB needs tag-store of tens of MB
- E.g. 256MB DRAM cache needs ~20MB tag store (5 bytes/line)
- But big blocks have their own issues
  - Wasted off-chip bandwidth

Option 1: SRAM Tags

Fast, But Impractical (Not enough transistors)

Option 2: Tags in DRAM

Naïve design has 2x latency (Two accesses -- tag and data)

Stacked DRAM Caches

Tags-On-SRAM

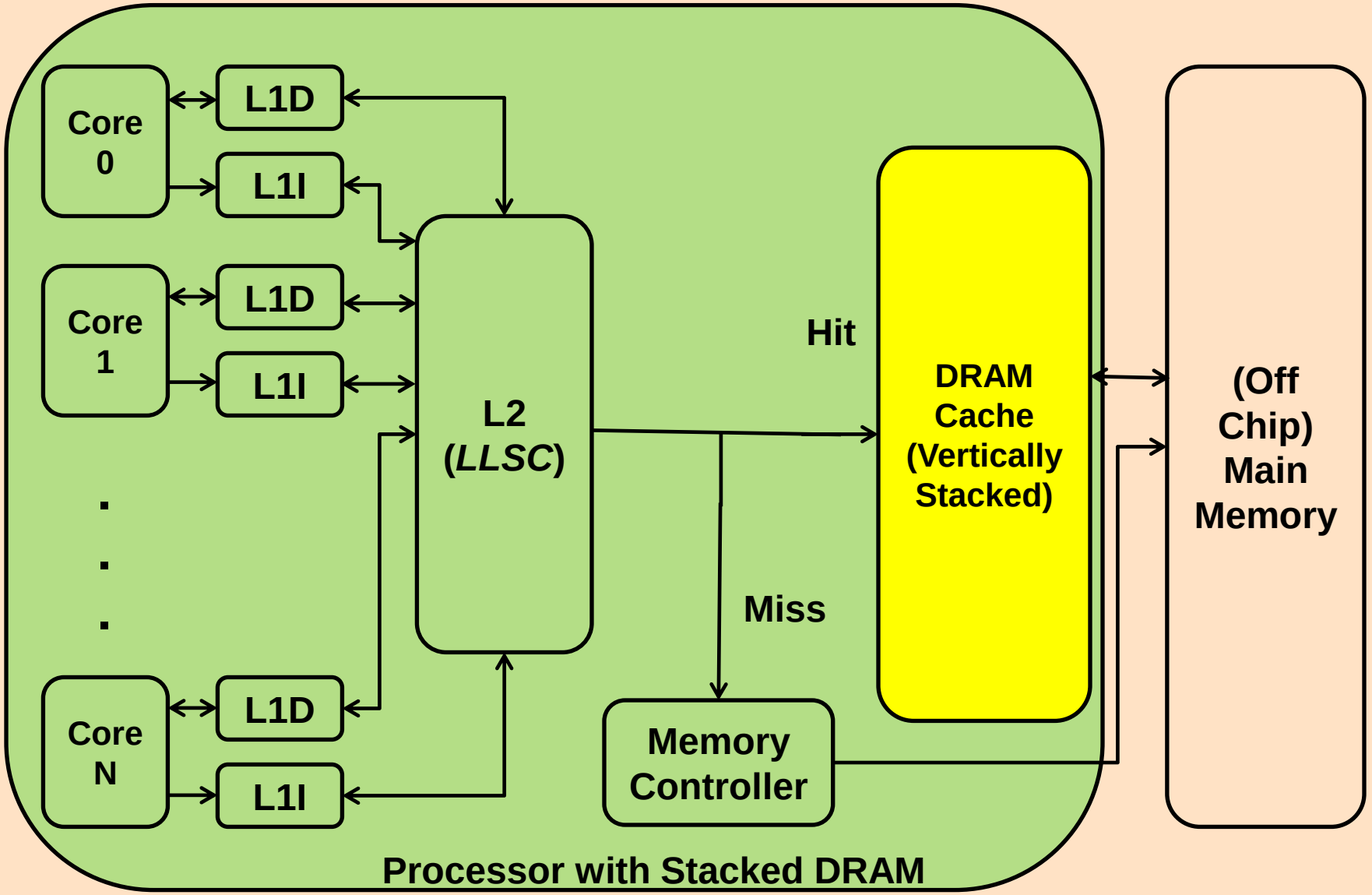
- Cache tags on SRAM
- Expensive SRAM
- Large storage overhead
- So typically uses larger block sizes to reduce overhead

  (~ 1KB)
- Off-chip bandwidth and cache utilization are concerns
- Several recent proposals (*FootPrintCache*, *CHOP*)

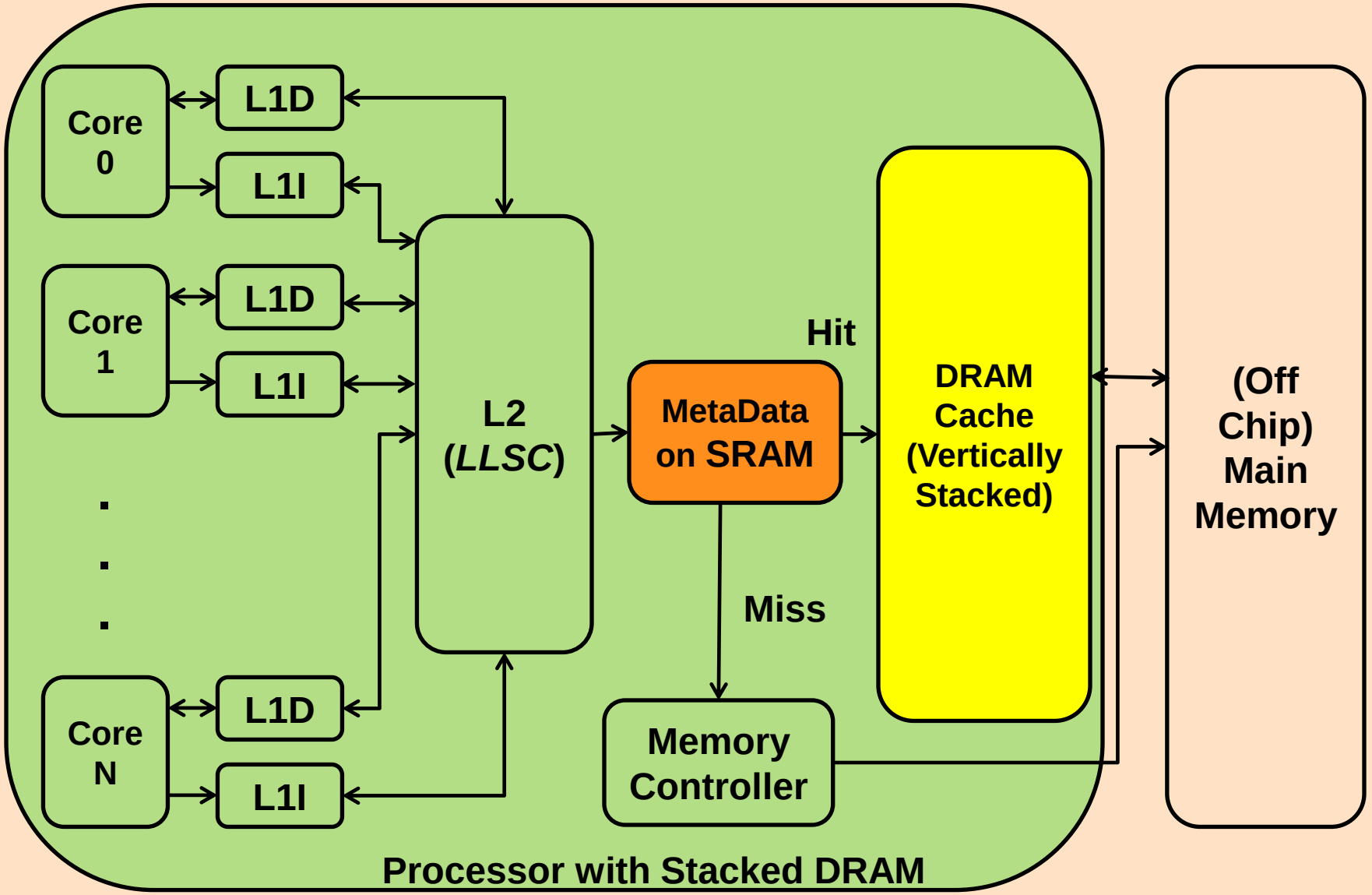
Tags-On-DRAM

- Cache tags on DRAM itself
- Typically 64B blocks
- Due to overhead of tag access from DRAM, requires some form of predictor/cache in SRAM
- Several recent proposals (*Loh-Hill, AlloyCache*, *ATCache, Bi-Modal*)

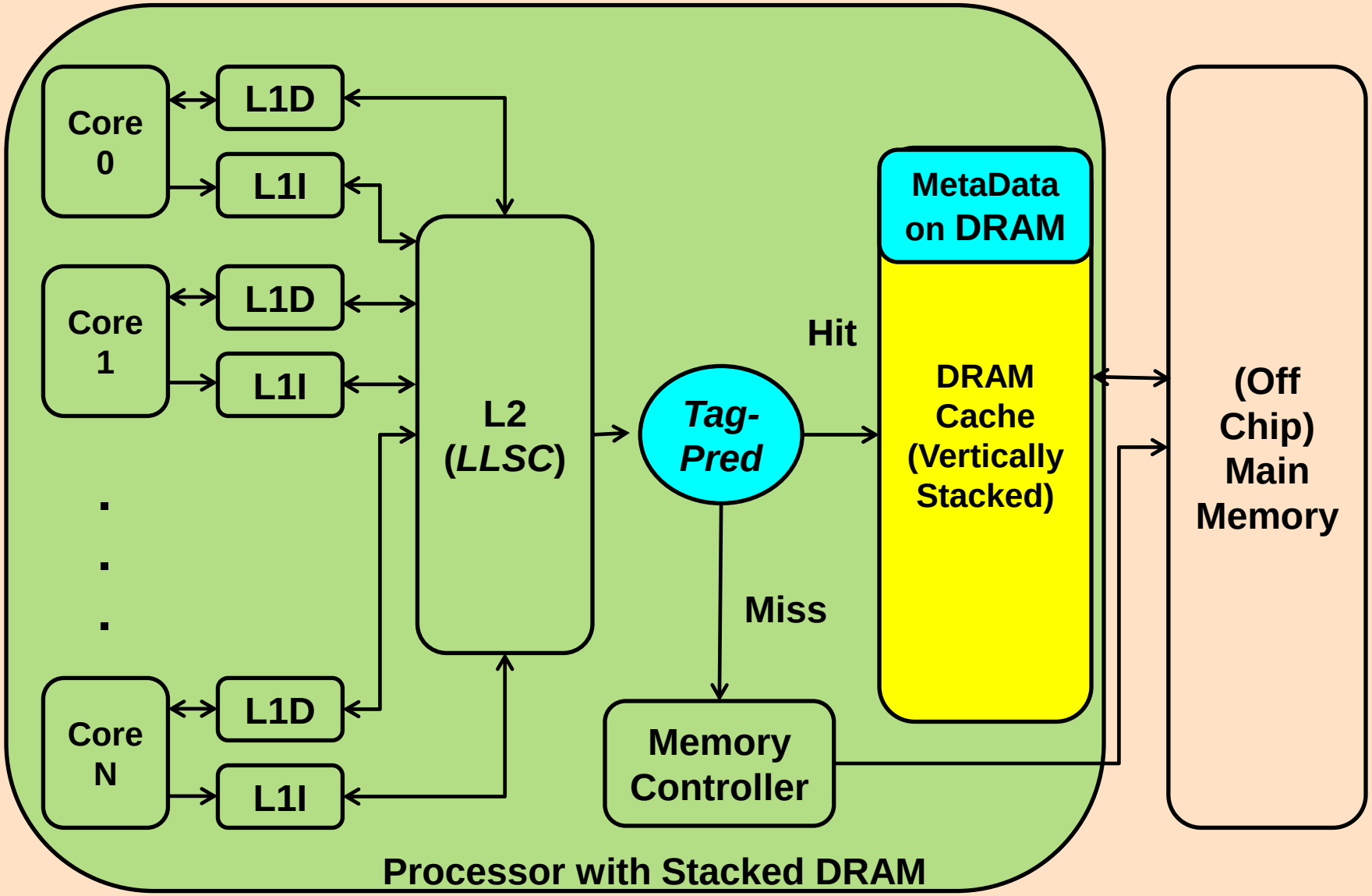
Stacked DRAM Cache Orgn.



Stacked DRAM Cache Orgn.

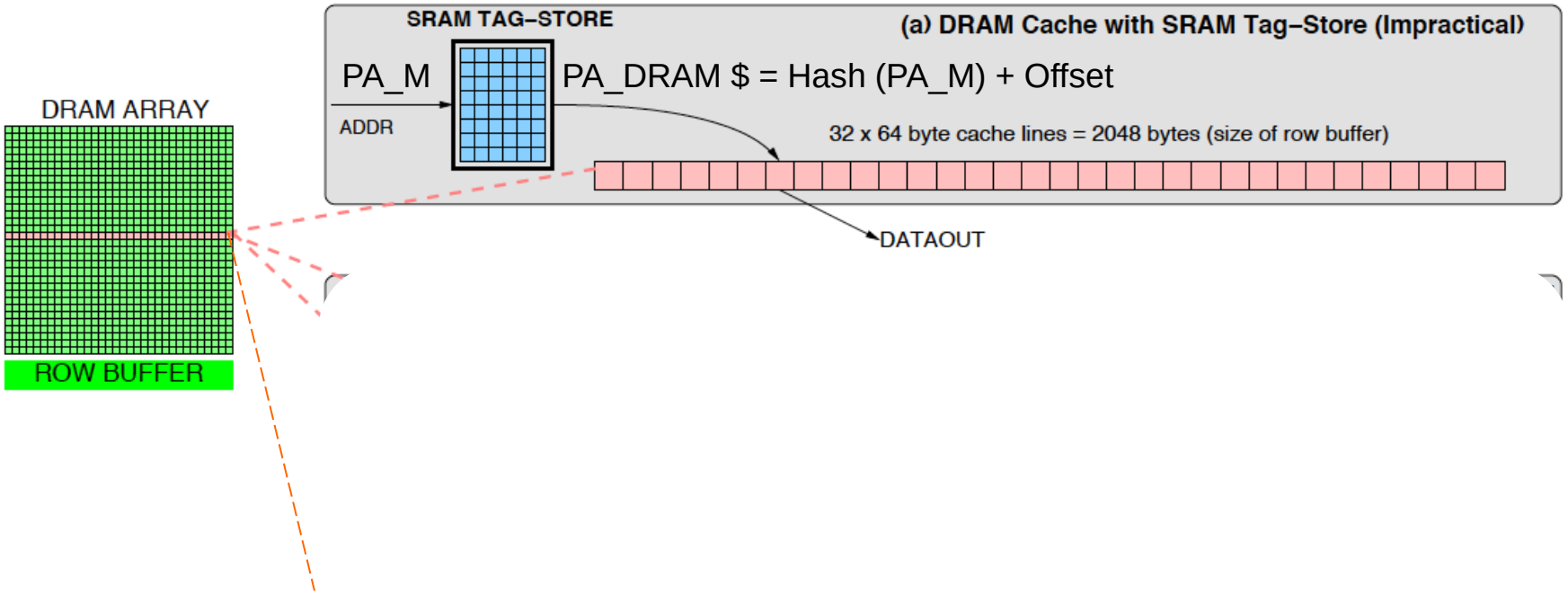


Stacked DRAM Cache Orgn.



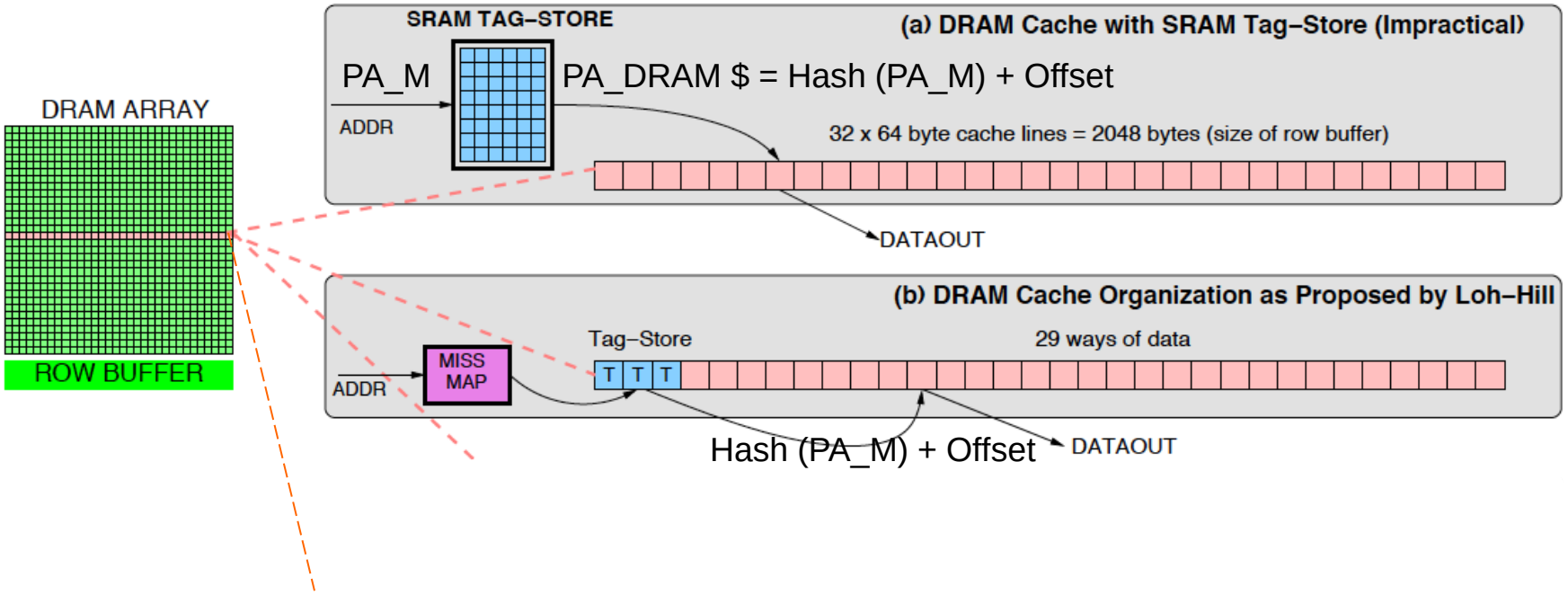
Overview of Different Designs

- For DRAM caches, critical to optimize first for latency, then hit-rate



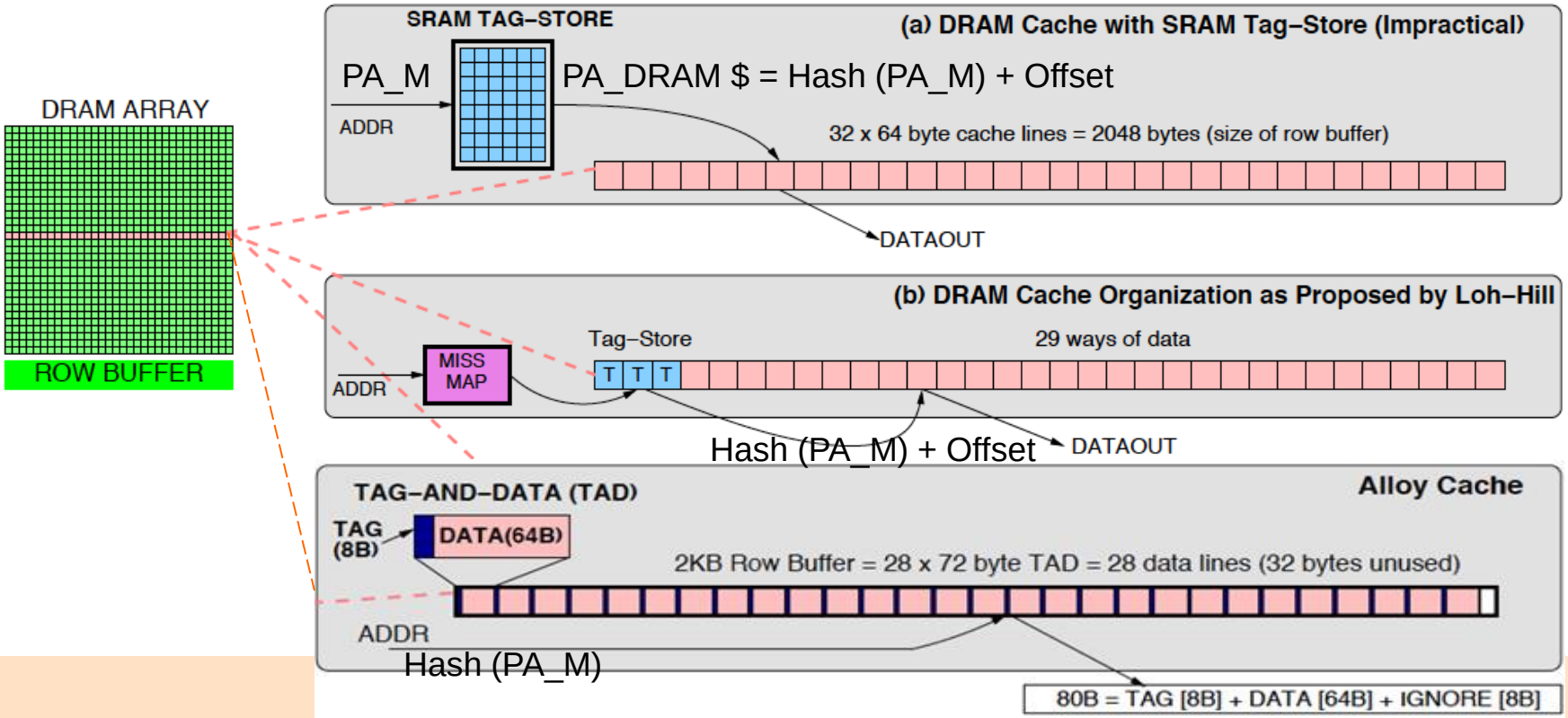
Overview of Different Designs

- For DRAM caches, critical to optimize first for latency, then hit-rate



Overview of Different Designs

- For DRAM caches, critical to optimize first for latency, then hit-rate



Overview of Bi-Modal Cache

- Tags-In-DRAM organization

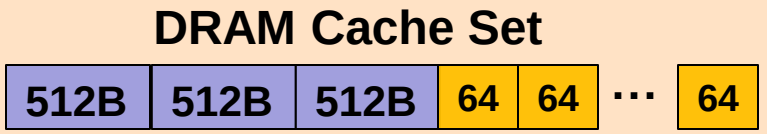
- With 3 new organizational features:
  1) Cache Sets are *Bi-Modal* – they can hold a combination of big (512B) and small (64B) blocks → **Improves Hit Rate And Reduces Off-Chip Bandwidth**
  2) Parallel Tag and Data Accesses
  3) Eliminating Most Tag Accesses via a small SRAM based *Way Locator*

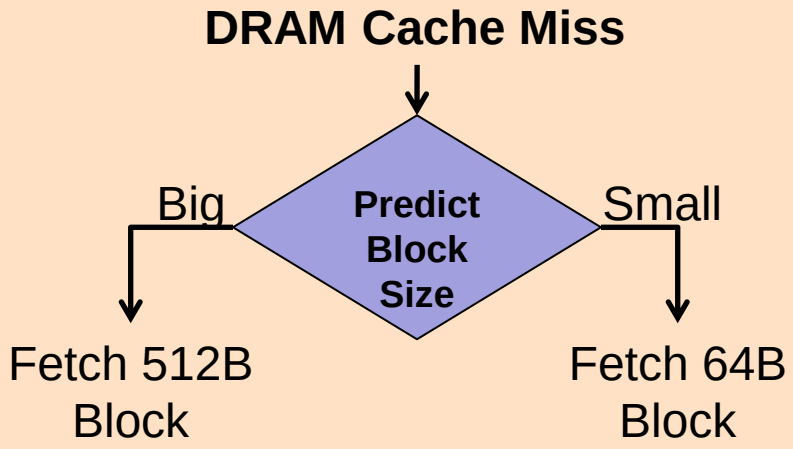
  → **Reduce Hit Latency**

Supporting Bi-Modal Block Sizes

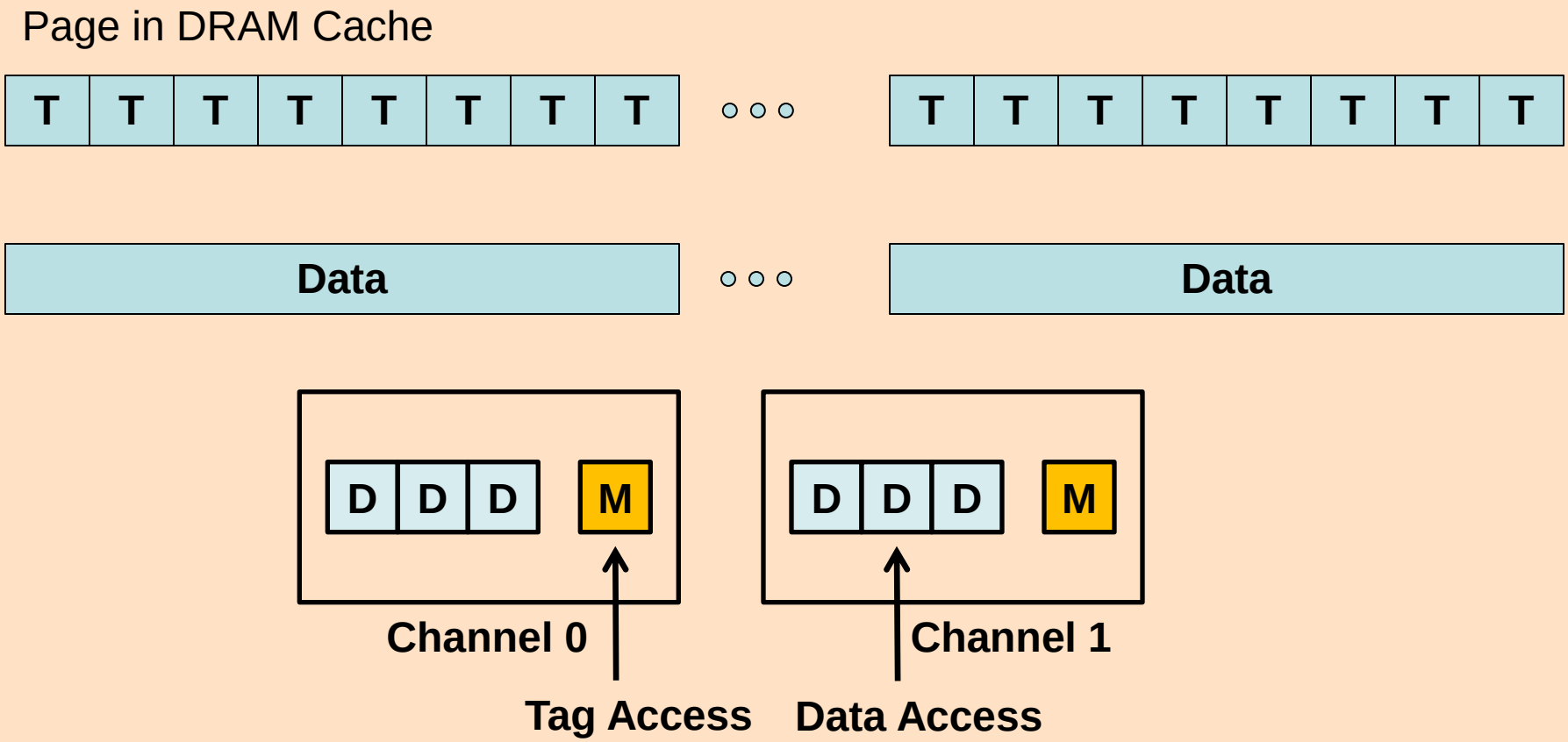
- Each Set can hold some big (512B) and some small (64B) blocks.

- Block Size Predictor
  - Blocks with high spatial reuse
    - ➔ fetch 512B
  - Blocks with little spatial reuse
    - ➔ fetch 64B

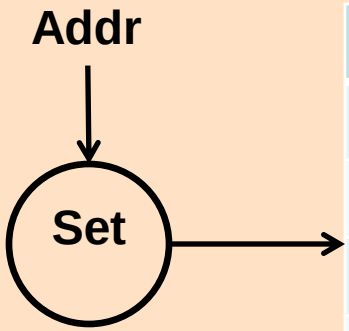




Parallel Tag and Data Accesses



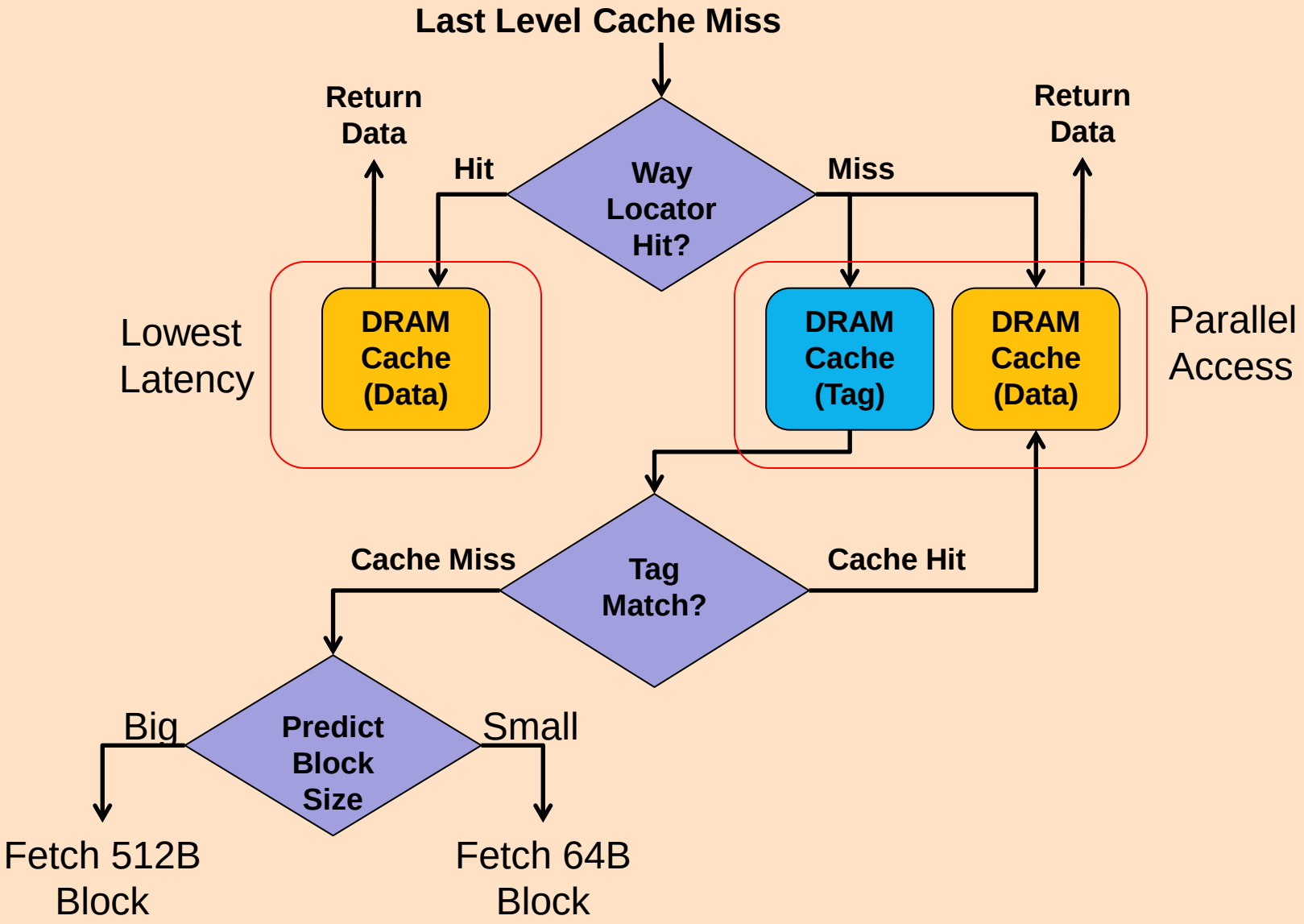
Eliminating a Majority of Tag Accesses using the *Way Locator*



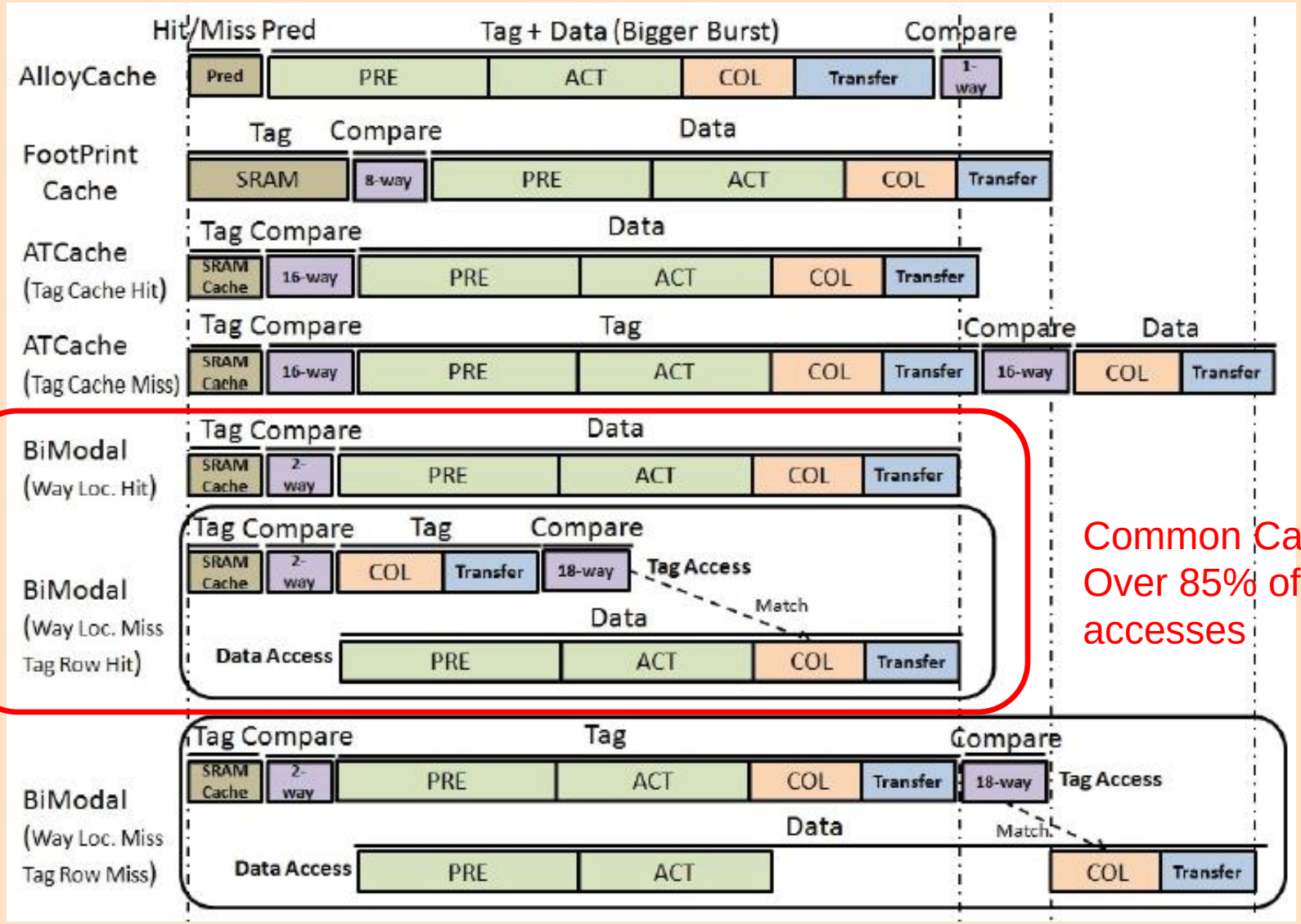
Set	MRU: Tag	and Way	MRU-1: Tag and Way	
Set 0	Tag a1	Way 3	Tag a2	Way 1
Set 1	Tag a3	Way 2	Tag a4	Way 0
Set 2	Tag a5	Way 0	Tag a6	Way 3
…				
Set N	Tag am	Way x	Tag an	Way y

2-way Set Associative Cache
Each entry specifies tag and associated way (DRAM column) where data is stored

Putting them together



Hit Latency Improvement



Common Case: Over 85% of accesses