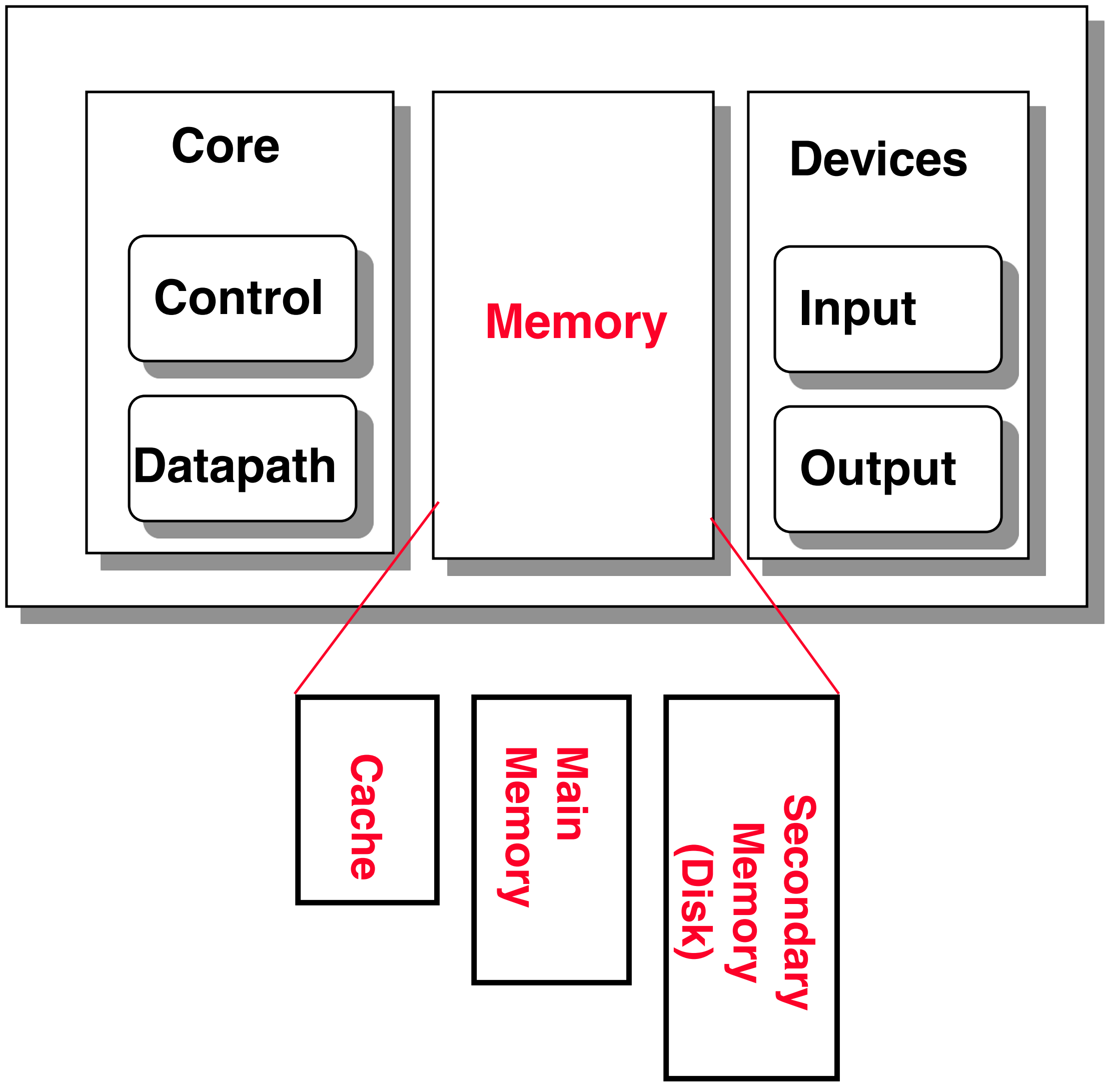
CMPEN 431
Computer Architecture
Fall 2017
Exploiting the Memory Hierarchy: Main Memory

Mahmut Taylan Kandemir (www.cse.psu.edu/~kandemir)

[Adapted from *Computer Organization and Design, 5th Edition*,
Patterson & Hennessy, © 2014, MK
With additional thanks/credits to Onur Mutlu and Jack Sampson

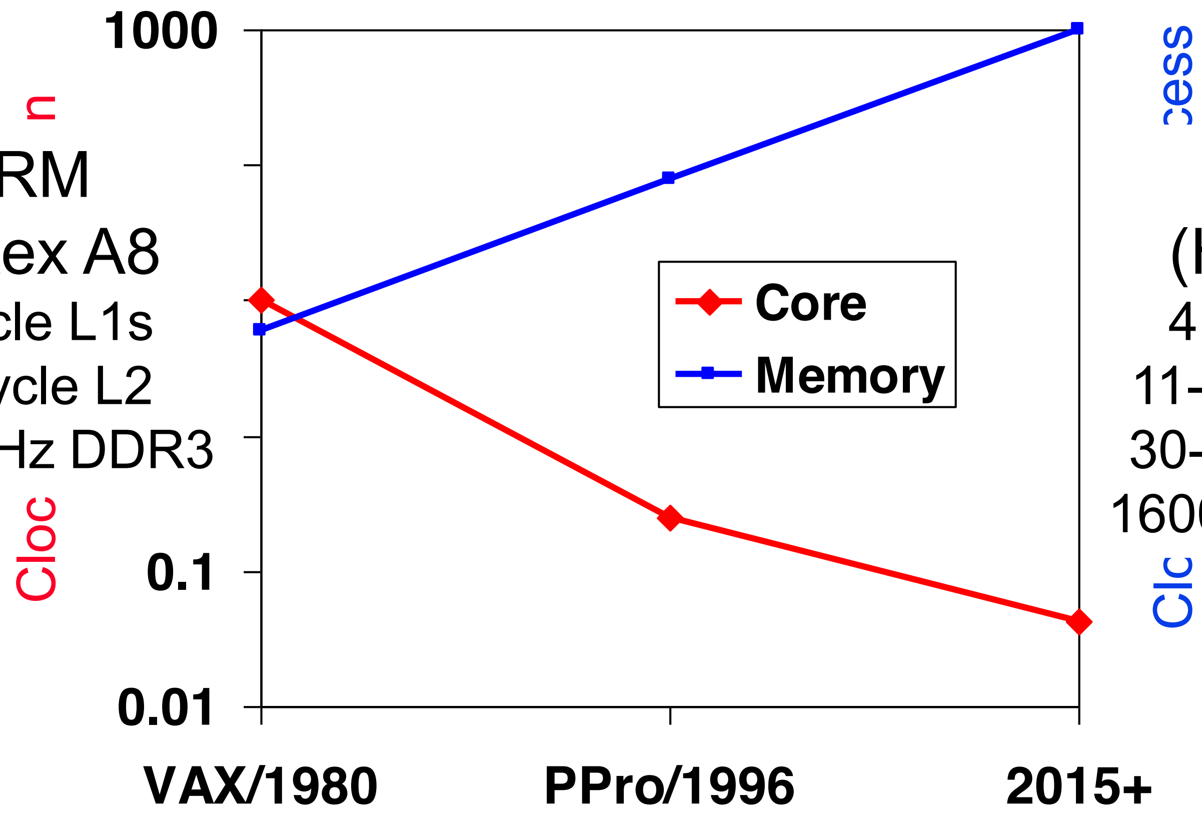
Review: Major Components of a Computer



The "Memory Wall"

- Core vs DRAM speed disparity continues to grow

ARM
Cortex A8
1 cycle L1s
11 cycle L2
1600MHz DDR3



Intel i7
(Haswell)
4 cycle L1s
11-16 cycle L2
30-55 cycle L3
1600MHz DDR3

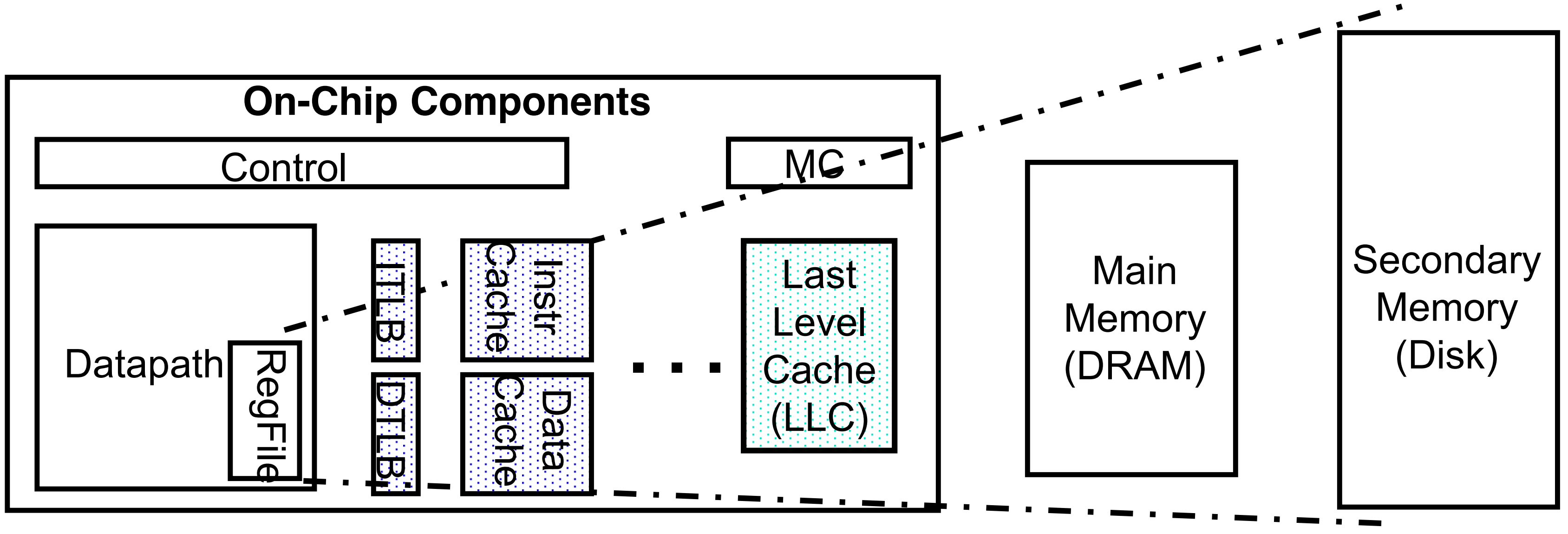
- Good memory hierarchy (cache) design is increasingly **important** to overall performance

The Memory Hierarchy Goal

- Fact: Large memories are slow and fast memories are small
- Fact: On-chip (with the cores) memories are much faster to respond than off-chip memories (even when they are the same size) because of much faster interconnect
- How do we create a memory that gives the illusion of being large, cheap and fast (most of the time)?
  - With hierarchy
  - With parallelism

A Typical Memory Hierarchy

- Take advantage of the principle of locality to present the user with as much memory as is available in the *cheapest* technology at the speed offered by the *fastest* technology



Speed (%cycles):	½'s	1's	10's	100's	10,000's
Size (bytes):	100's	10K's	M's	G's	T's
Cost:	highest				lowest

The Memory Hierarchy: Why Does it Work?

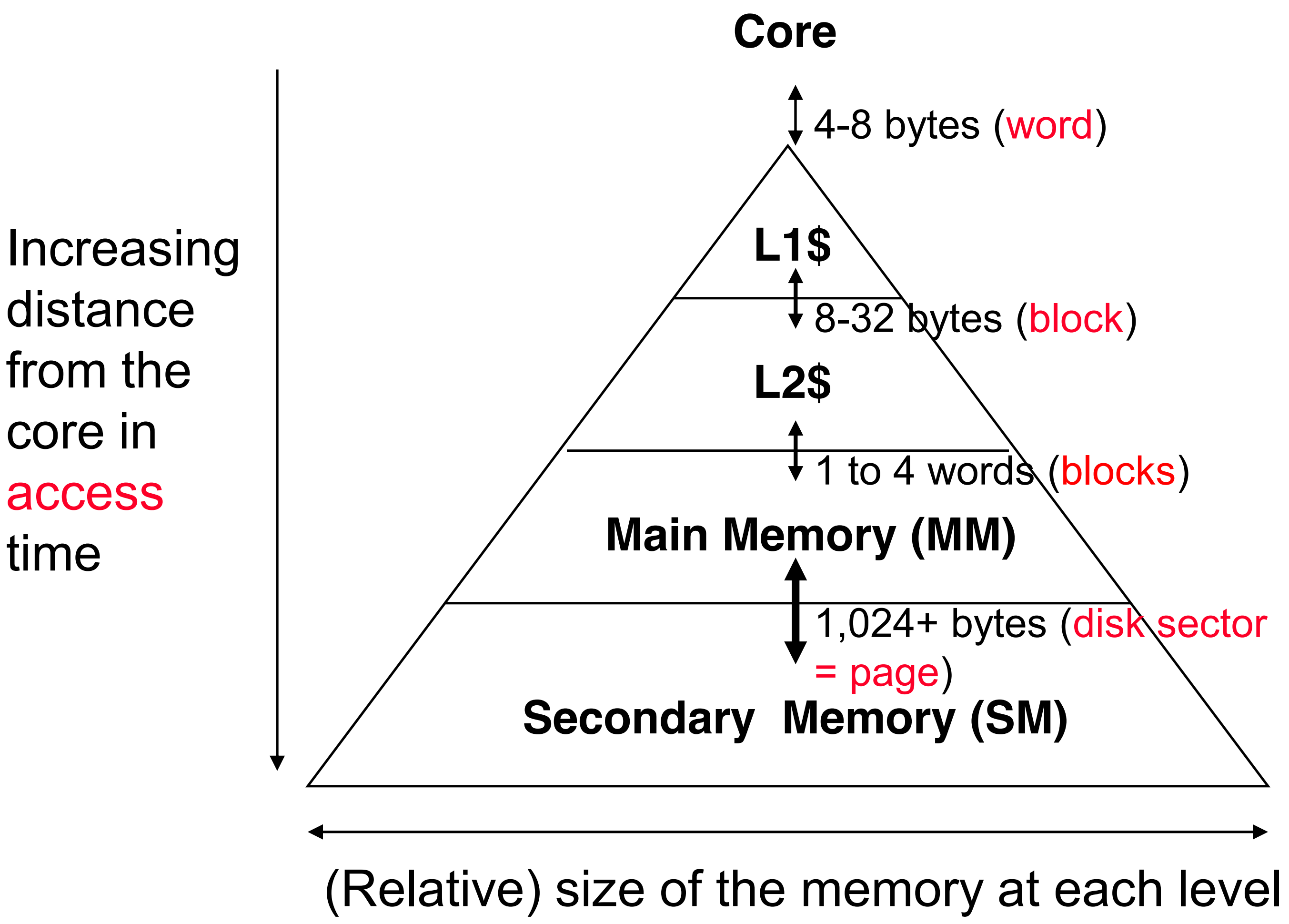
- Temporal Locality (locality in time)
  - If a memory location is referenced, then it will tend to be referenced again soon

  ⇒ Keep most recently accessed data items closer to the core

- Spatial Locality (locality in space)
  - If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon

  ⇒ Move blocks consisting of contiguous words closer to the core

Characteristics of the Memory Hierarchy



Core
4-8 bytes (word)
L1$
8-32 bytes (block)
L2$
1 to 4 words (blocks)
Main Memory (MM)
1,024+ bytes (disk sector = page)
Secondary Memory (SM)
Increasing distance from the core in access time
(Relative) size of the memory at each level

Inclusive caches – what is in L1$ is a subset of what is in L2$

Noninclusive caches – what is in L1$ is not a subset of what is in L2$

The Memory Hierarchy: Terminology

Hit Time << Miss Penalty

- Block (or line): the minimum unit of information that is present (or not) in a level of the memory hierarchy
- Hit Rate: the fraction of memory accesses found in a level of the memory hierarchy
  - Hit Time: Time to access that level which consists of

    Time to access the block + Time to determine hit/miss
- Miss Rate: the fraction of memory accesses *not* found in a level of the memory hierarchy $\Rightarrow$ 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in that level with the corresponding block from a lower level which consists of

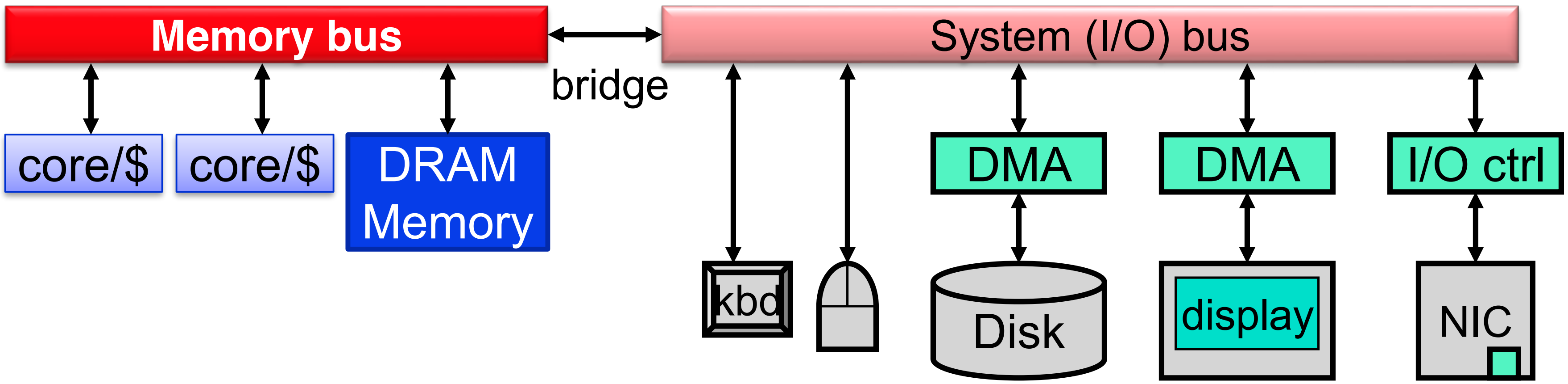
    Time to determine that there is a miss + Time to access that block in the lower level + Time to transmit that block to the level that experienced the miss + Time to insert the block in that level + Time to pass the block to the requestor

Memory Hierarchy Technologies

- Caches use ***SRAM*** for speed and technology compatibility with the core
  - Fast (typical access times of 100 psec to 2 nsec)
  - Lower density (6 transistor cells), higher power, expensive ($200 to $800 per GB)
  - Static: content will last "forever" (as long as power is left on)
- Main memory uses ***DRAM*** for size (density)
  - Slower (typical access times of 10 nsec to 70 nsec)
  - Higher density (1 transistor cells), lower power, cheaper ($5 to $10 per GB)
  - Dynamic: needs to be "refreshed" regularly (~ every 64 ms)
    - consumes ~ 1% of the active cycles of the DRAM
  - Addresses divided into 2 halves (row and column)
    - *RAS* or *Row Access Strobe* triggering the row decoder
    - *CAS* or *Column Access Strobe* triggering the column selector

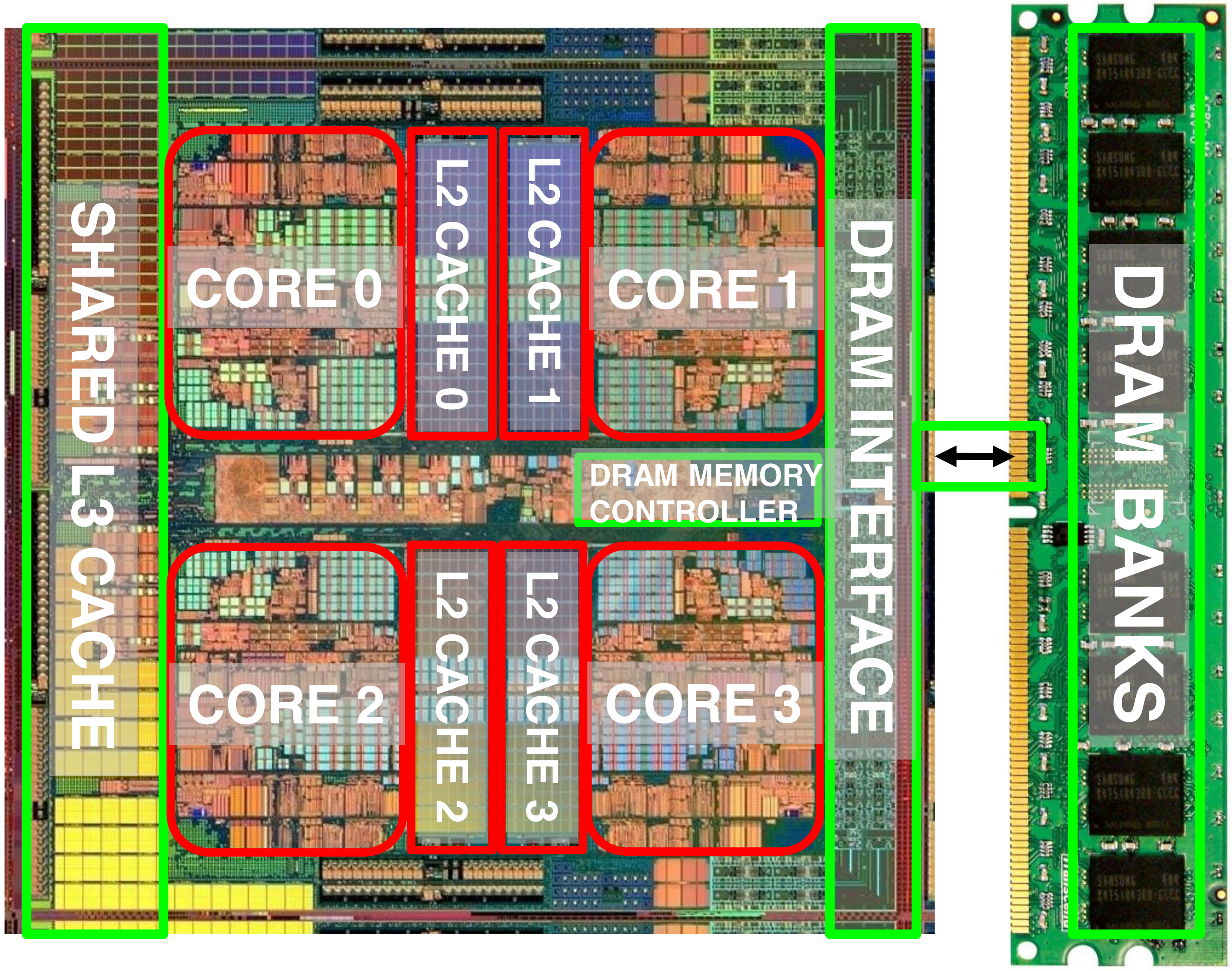
Memory Hierarchy

- Cores, caches and **main memory**
  - Connected by the **memory bus** (aka northbridge, ideally on-chip with the cores and caches)



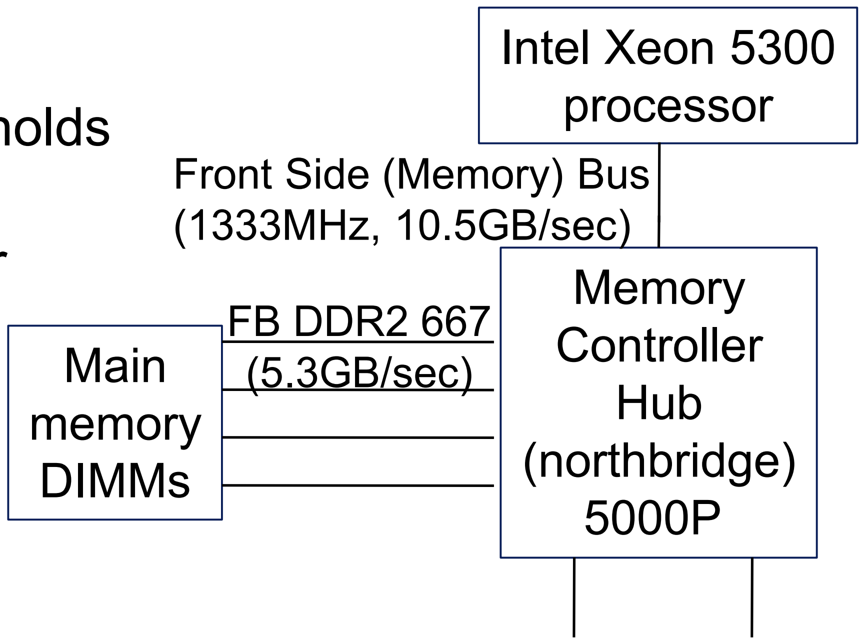
- I/O peripherals: storage, input, display, network, …
  - With separate or built-in DMA
  - Connected by the **system bus** (aka southbridge) which is connected to memory bus

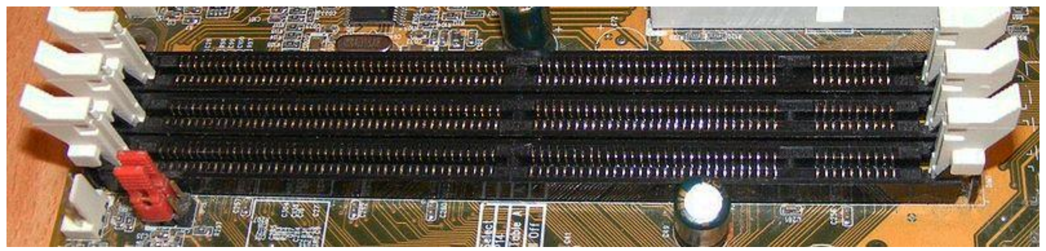
Main Memory in the System



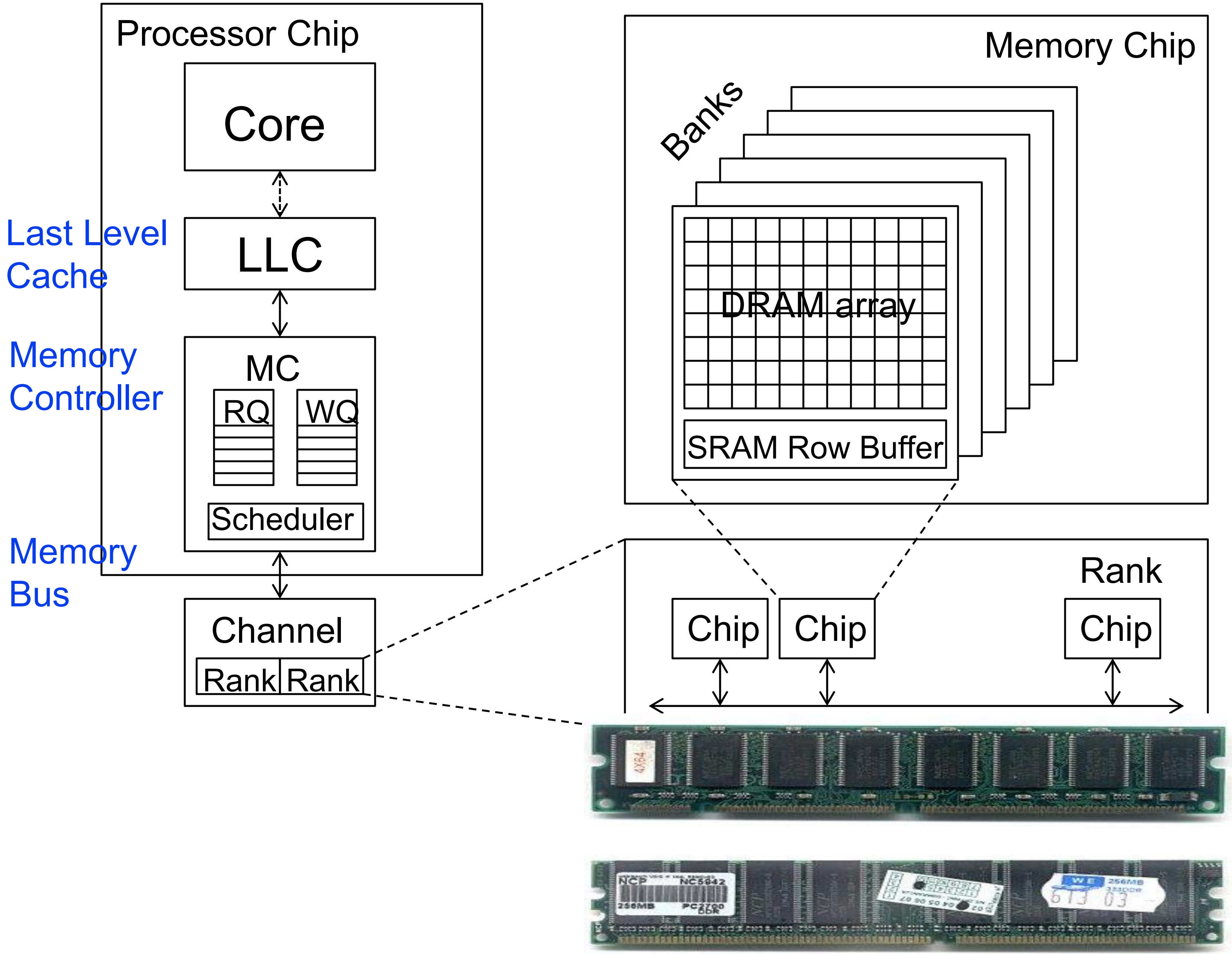
DRAM packaging - DIMMs

- Dual In-line Memory Modules
  - Small printed circuit board that holds DRAMs with a 64-bit datapath
  - Each contain eight “x4” (by 4) or “x8”(by 8) DRAM parts


Intel Xeon 5300 processor
Front Side (Memory) Bus (1333MHz, 10.5GB/sec)
FB DDR2 667 (5.3GB/sec)
Main memory DIMMs
Memory Controller Hub (northbridge) 5000P



From Core to DRAM



Channels, Memory Controllers, DIMMs, DRAM Chips

- To maximize memory bandwidth, there are multiple memory channels in the system. Each channel is managed by one memory controller (MC), and is connected to one or more DIMMs, each containing multiple DRAM chips.

- These DRAM chips are logically arranged into multiple ranks. Internally, each DRAM chip is partitioned into banks that are accessed in parallel.

- In each memory controller, there is one read queue and one write queue to buffer the outstanding memory requests. A scheduler (e.g., FCFS) determines which request should be serviced next. For each channel, only one request can be sent to the memory through the bus at each clock cycle.

DIMM vs Rank

- Sometimes memory modules are designed with two or more independent sets of DRAM chips connected to the same address and data buses
- Each such set is called a rank
- Since all ranks share the same buses, only one rank may be accessed at any given time
- DIMMs are currently being commonly manufactured with up to four ranks per module
- Consumer DIMM vendors have recently begun to distinguish between single and dual ranked DIMMs

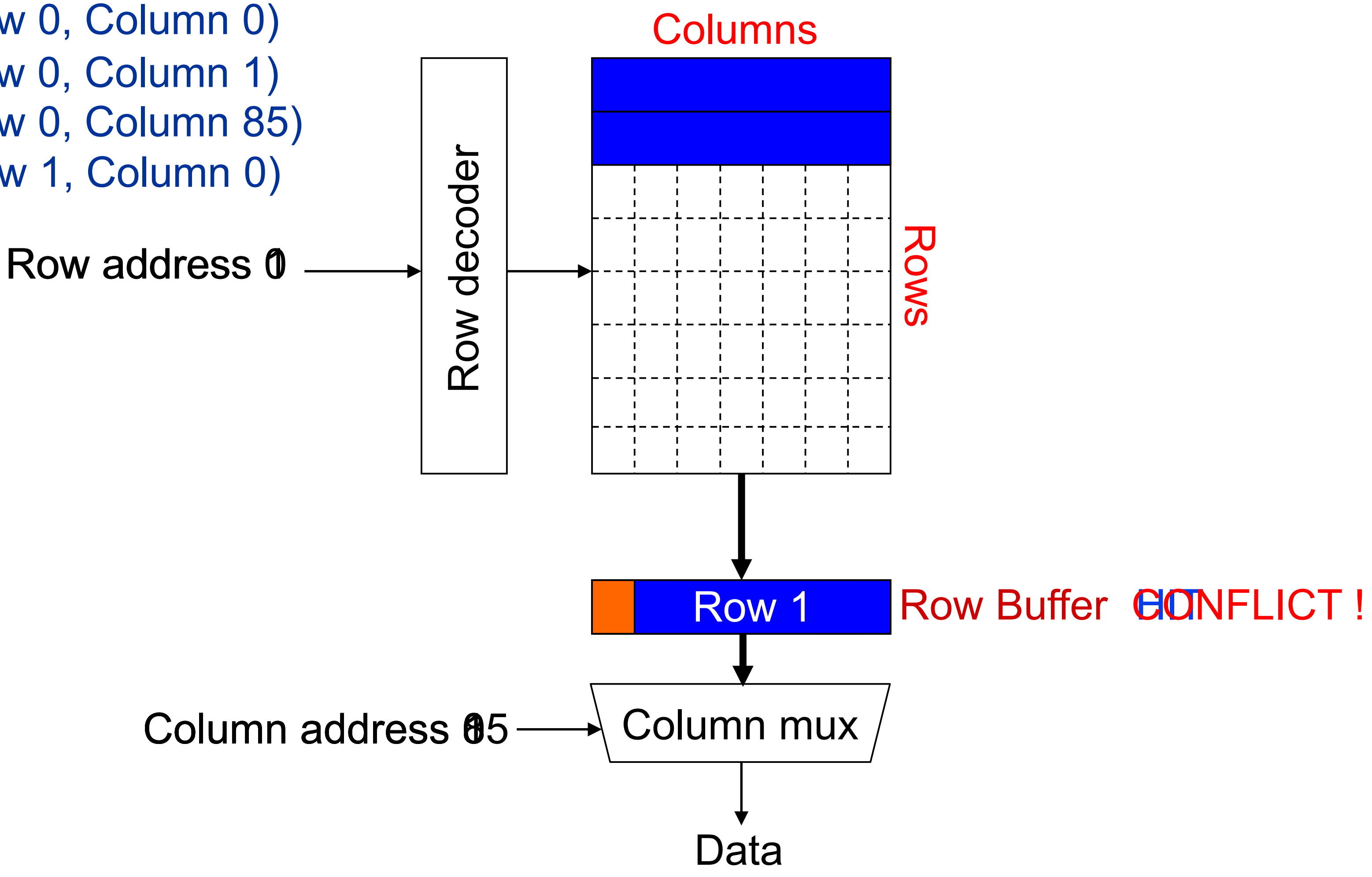
DRAM Access

- A DRAM bank is a 2D array of cells: rows x columns
- A "DRAM row" is also called a "DRAM page"
- A row is stored in "row buffer"

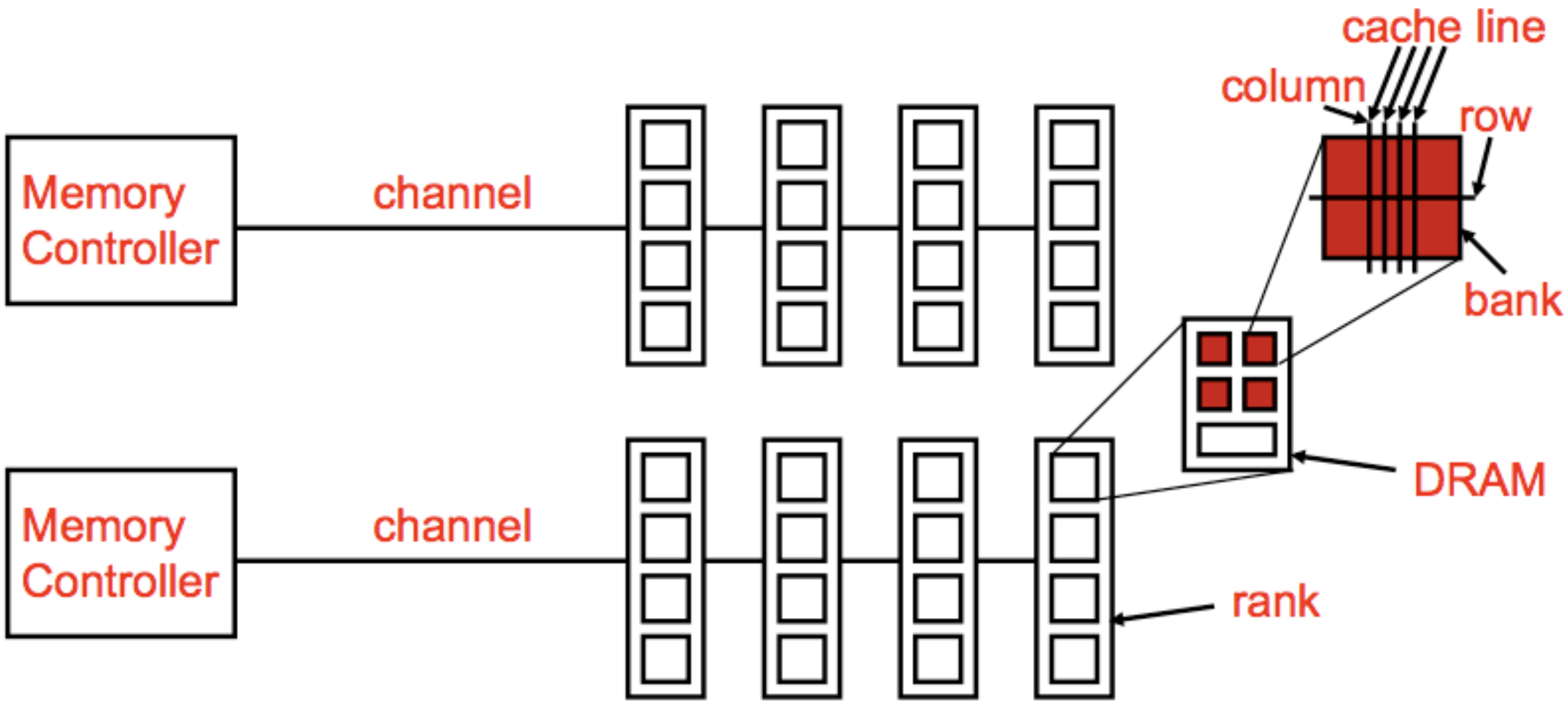
- Each address is a <row,column> pair
- Access to a "closed row"
  - Activate command opens row (placed into row buffer)
  - Read/write command reads/writes column in the row buffer
  - Precharge command closes the row and prepares the bank for next access
- Access to an "open row"
  - No need for activate command

DRAM Bank Operation

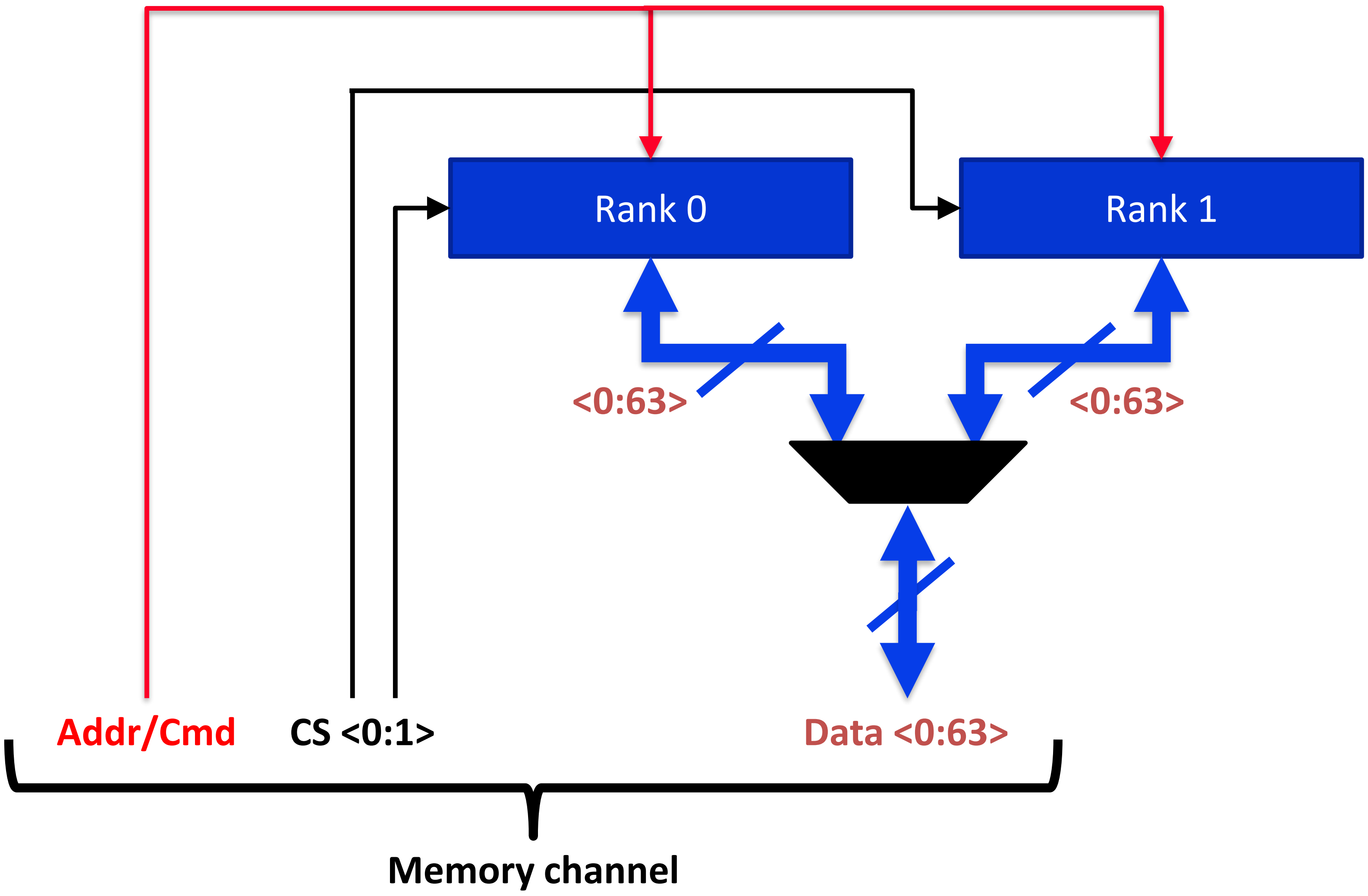
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



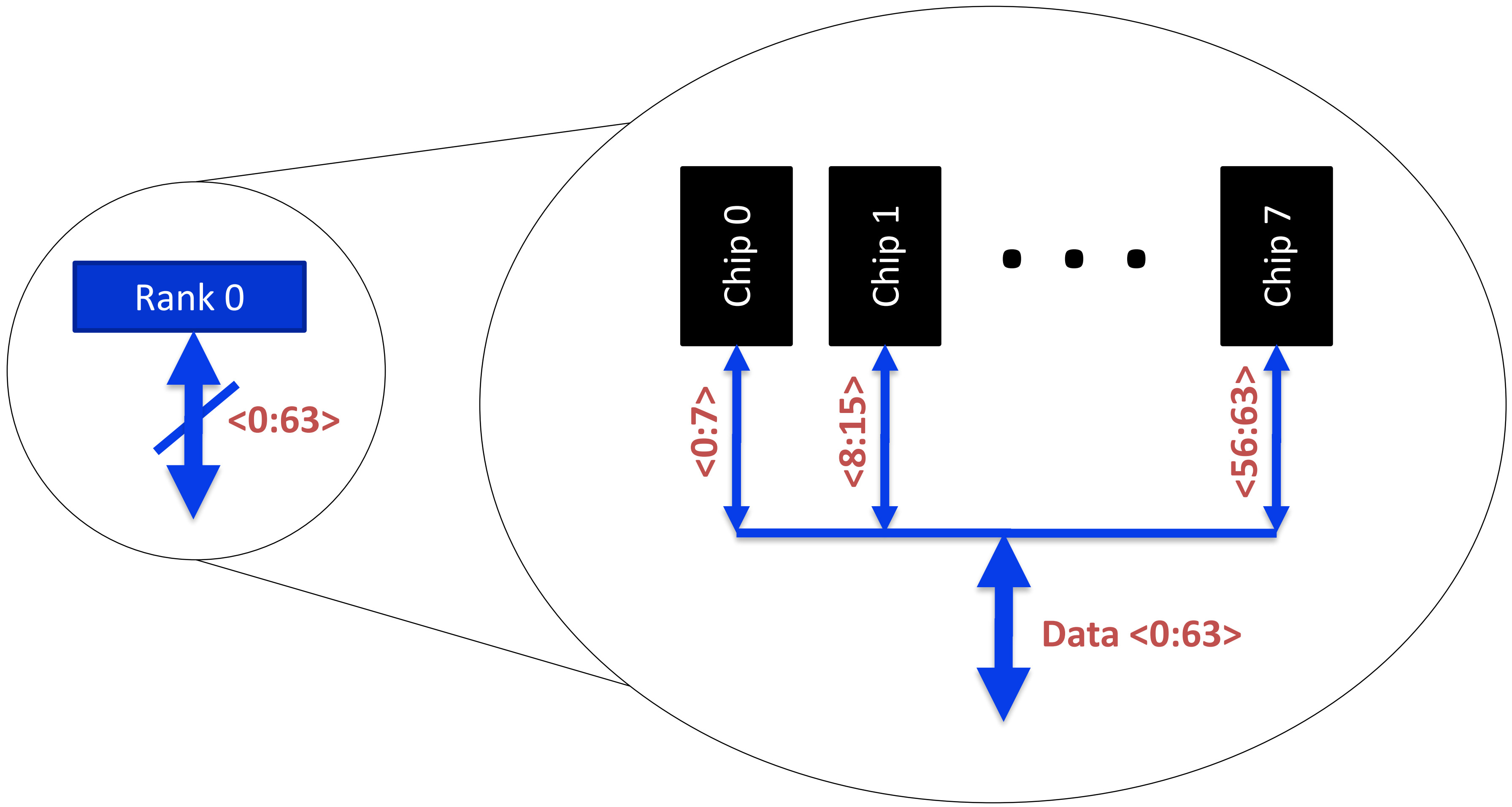
Generalized Memory Structure



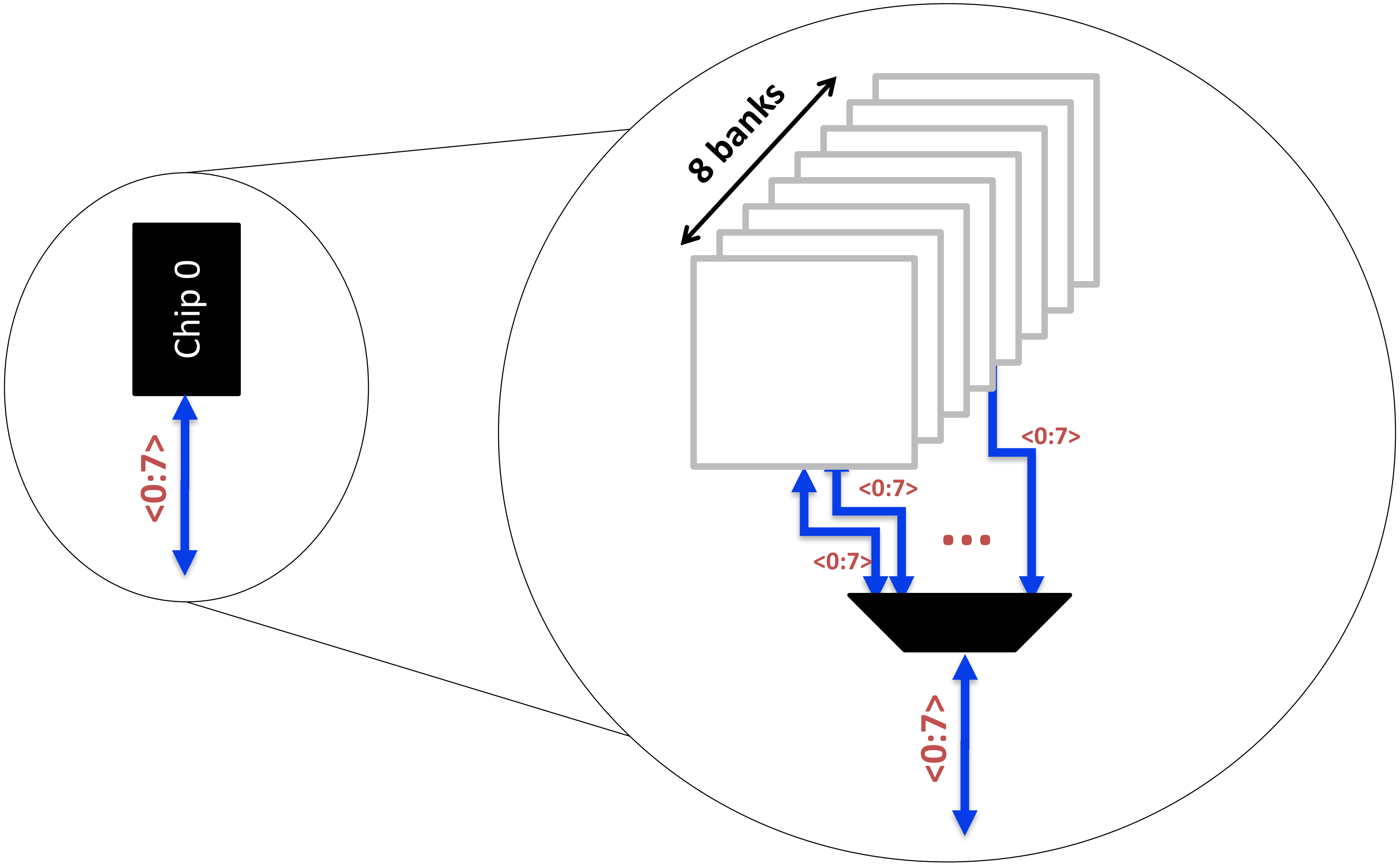
Ranks



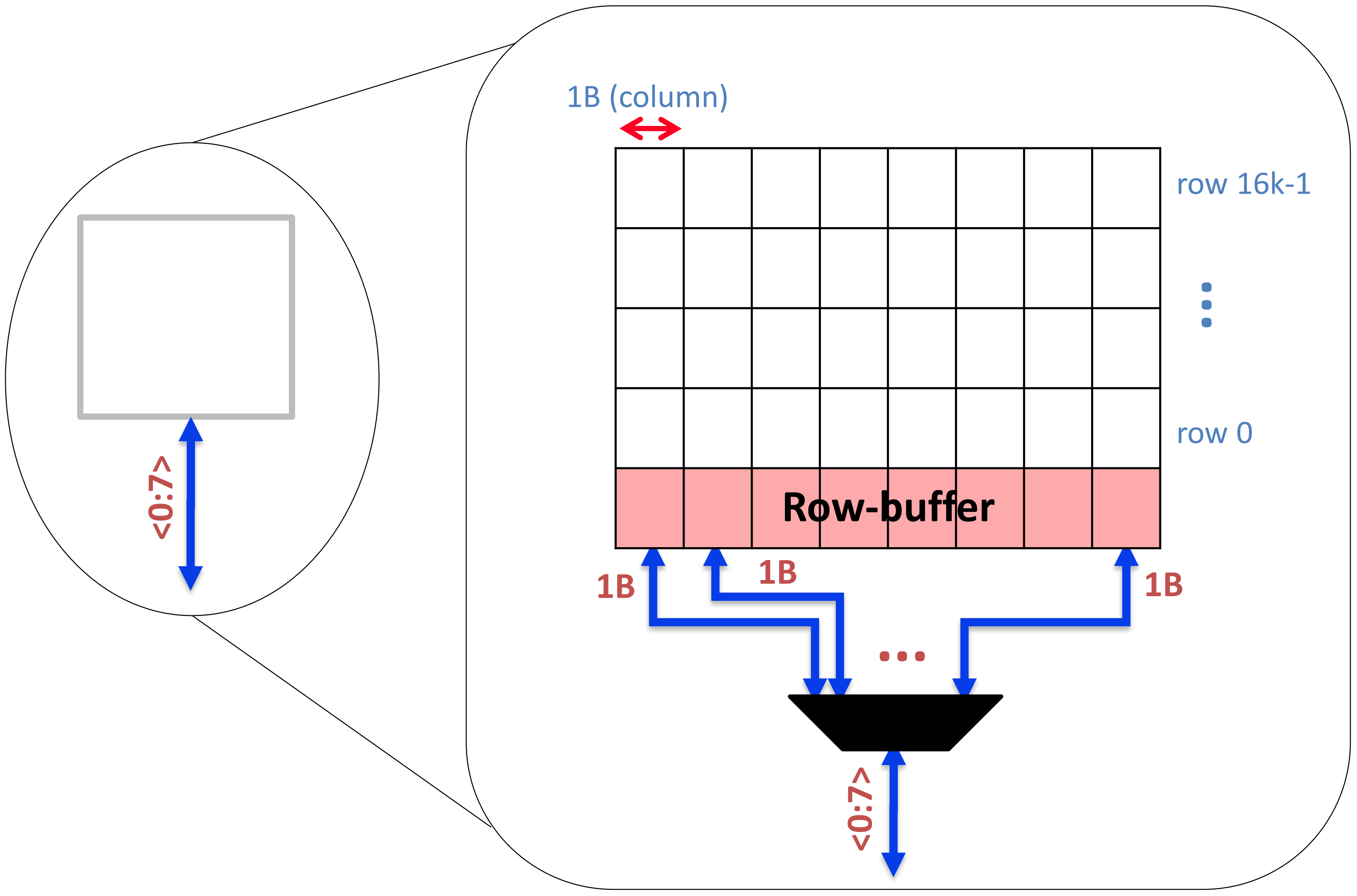
Breaking down a Rank



Breaking down a Chip



Breaking down a Bank



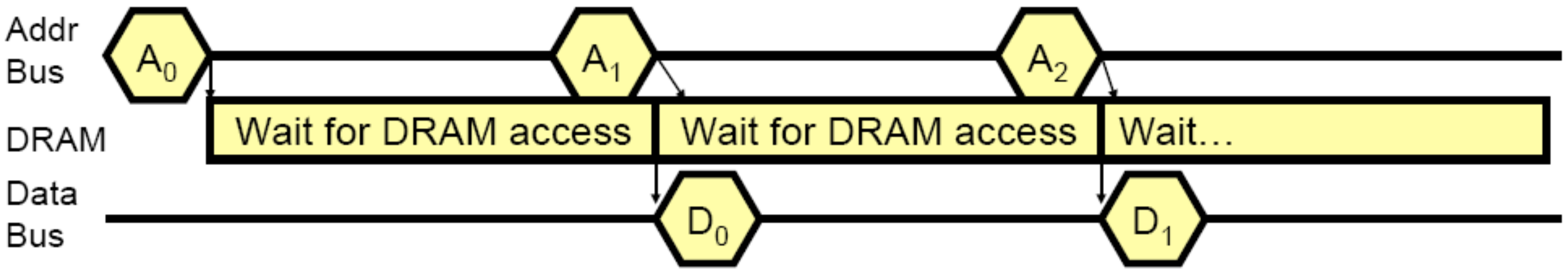
Latency Components: Basic DRAM Operation

- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
- 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)

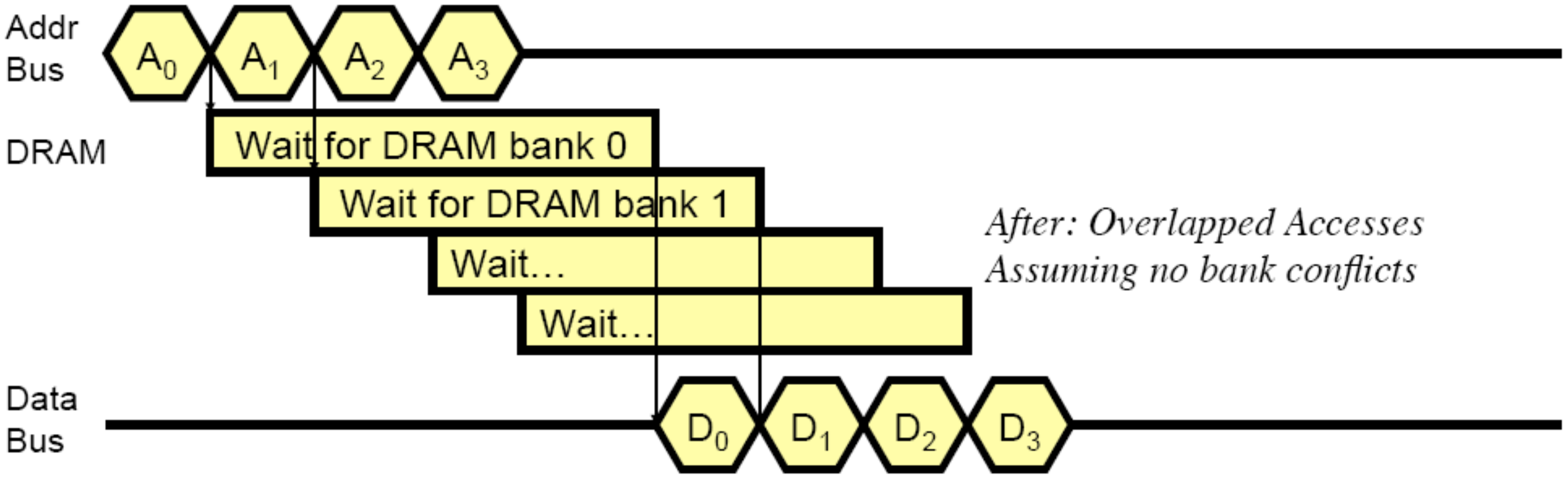
Multiple Banks (Interleaving) and Channels

- 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 Multiple Banks/Channels Help

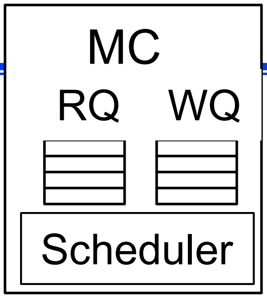


Before: No Overlapping
Assuming accesses to different DRAM rows



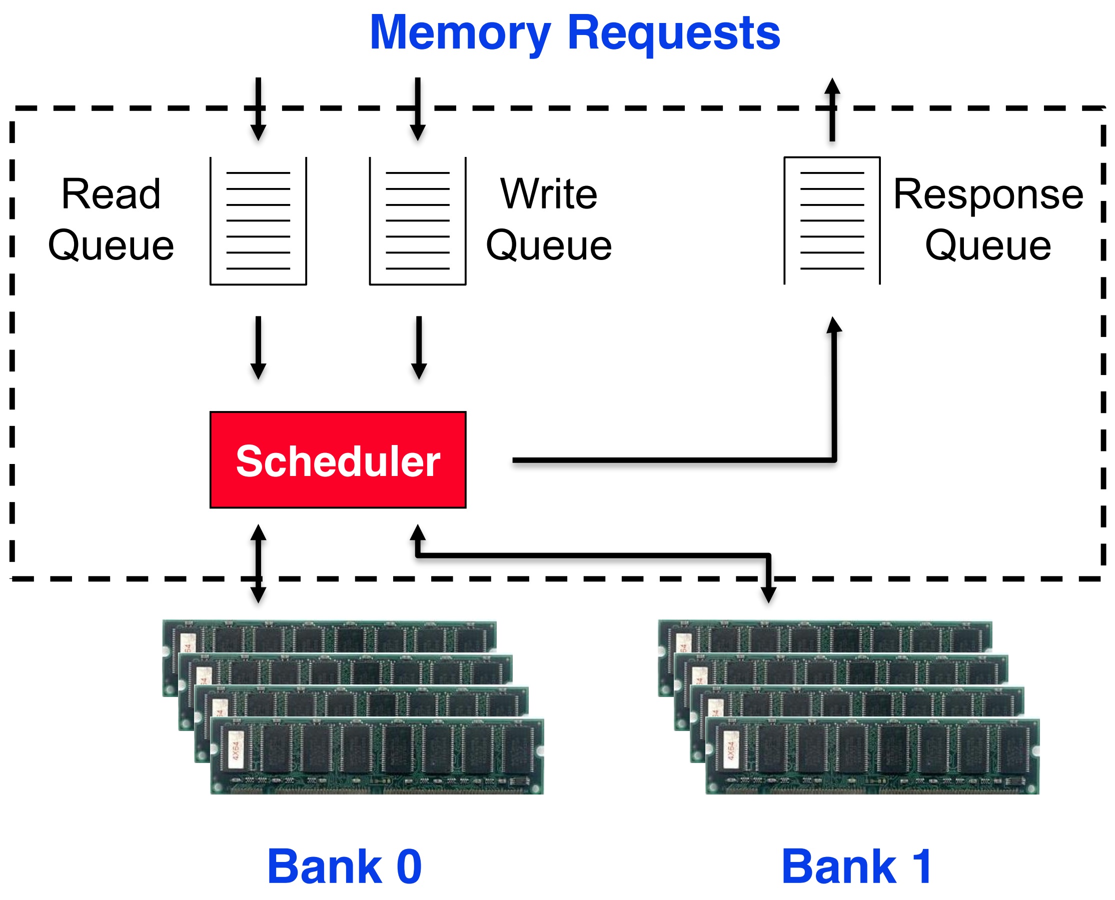
After: Overlapped Accesses
Assuming no bank conflicts

On-Chip Memory Controller



- A front-end buffers requests to the DRAM from the LLC (is independent of DRAM type)
  - Decodes the address into bank, row, column
  - Schedules requests to DRAM for max bandwidth
    1. Reorders bursts to minimize bank conflicts
    2. Prioritizes reads over writes (a core is waiting for the read data)
  - Sends responses from the DRAM back to the LLC
- Back-end interface to the target DRAM memory (and is depending on DRAM type)

Memory Controller



DRAM Request Scheduling (I)

- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer
  - Exploit row buffer locality as much as possible
- 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

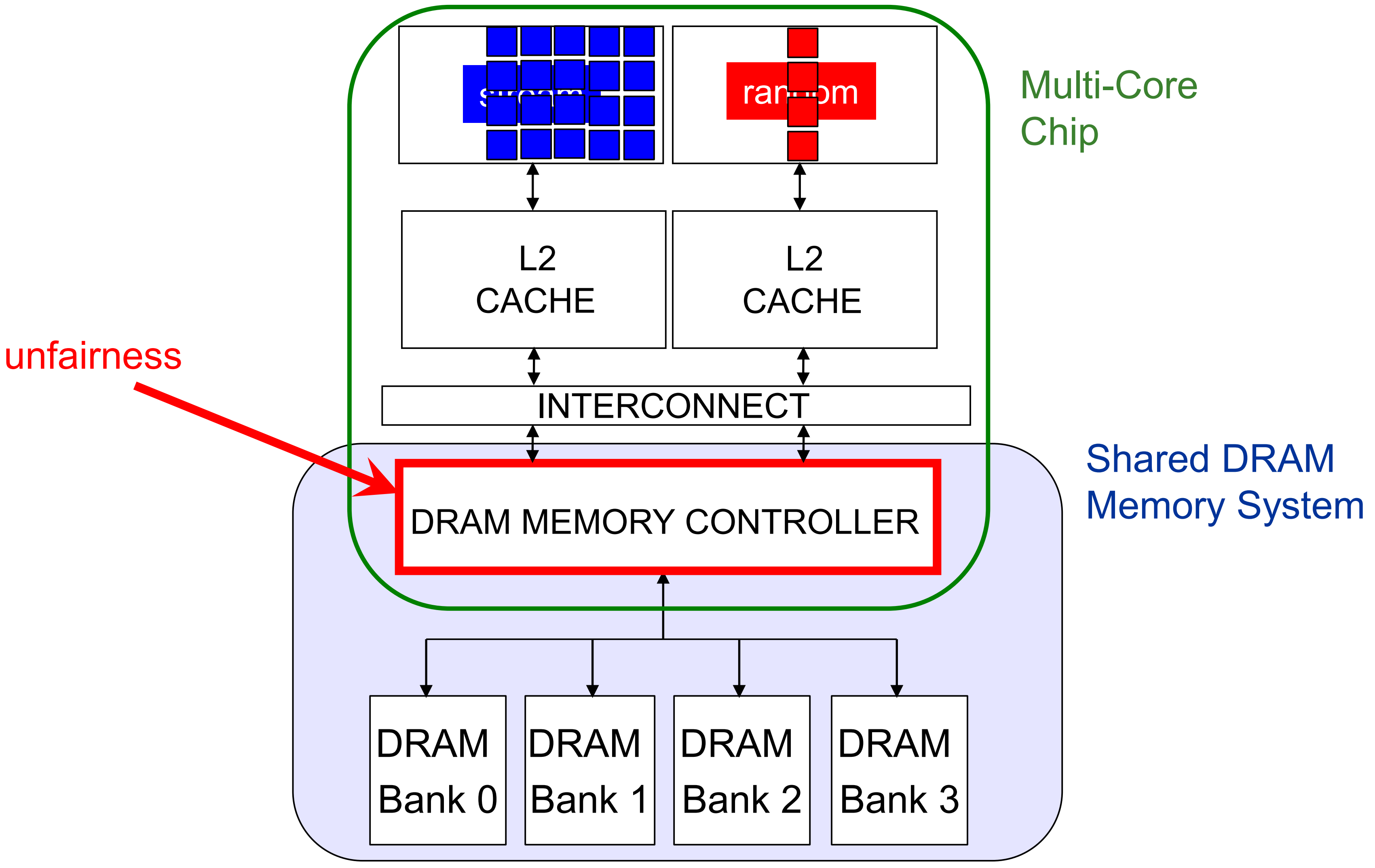
  What do we have these two criteria?
- This scheduling policy aims to maximize DRAM throughput
  - Note that this policy may be unfair when multiple threads/applications share the DRAM system

DRAM Request Scheduling (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?

Uncontrolled Interference: An Example



A Memory Performance Hog

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = j*linesize;    streaming
    A[index] = B[index];
    …
}
```

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = rand();    random
    A[index] = B[index];
    …
}
```

STREAM

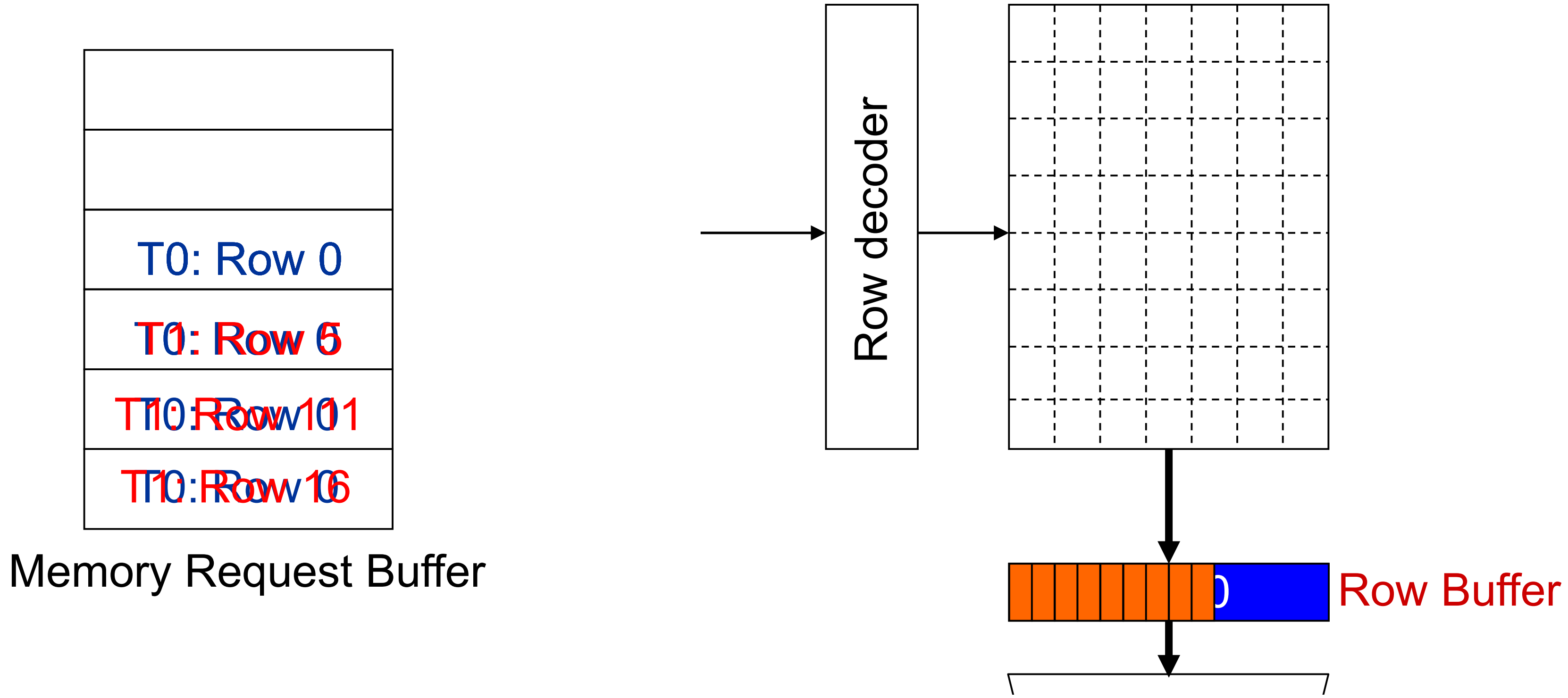
- Sequential memory access
- Very high row buffer locality (96% hit rate)
- Memory intensive

RANDOM

- Random memory access
- Very low row buffer locality (3% hit rate)
- Similarly memory intensive

What Does the Memory Hog Do?

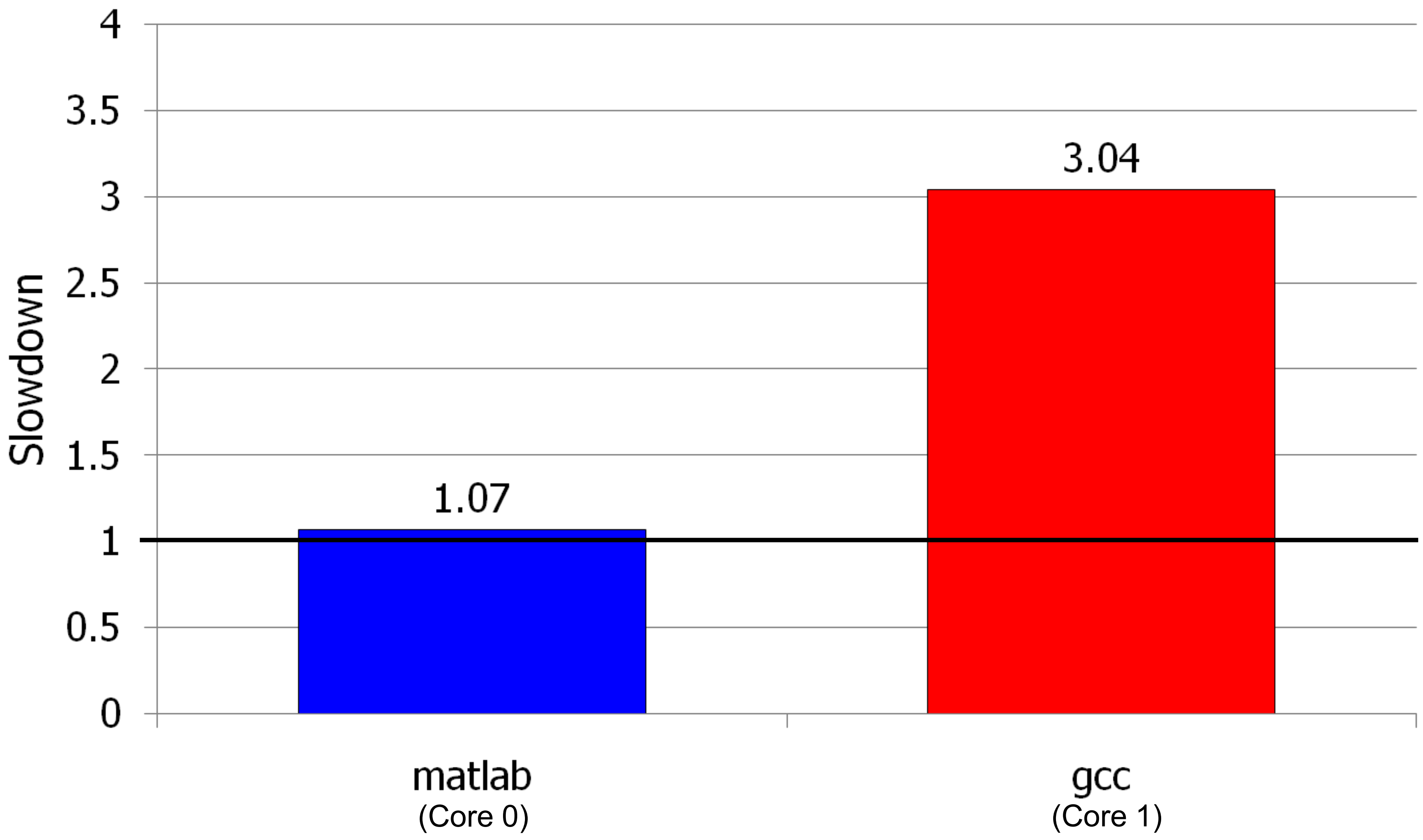
important



Row size: 8KB, cache block size: 64B

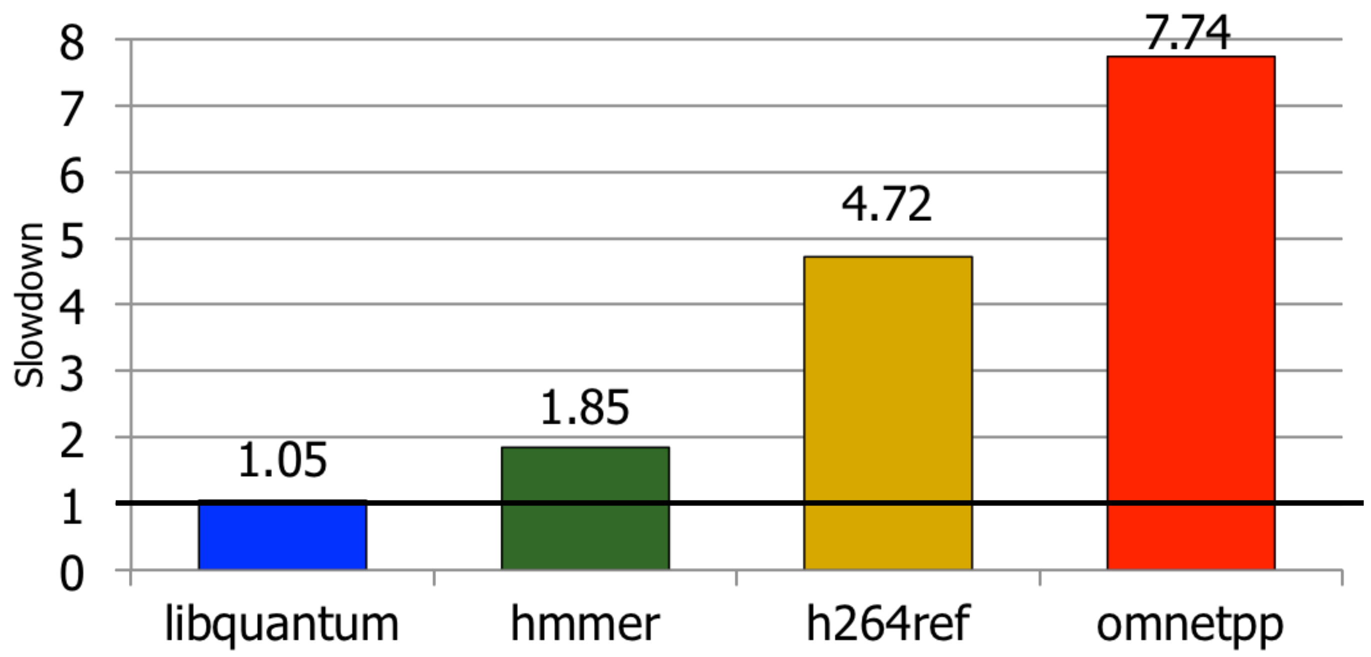
128 (8KB/64B) requests of T0 serviced before T1

Slowdown in Multi-Core Systems



Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

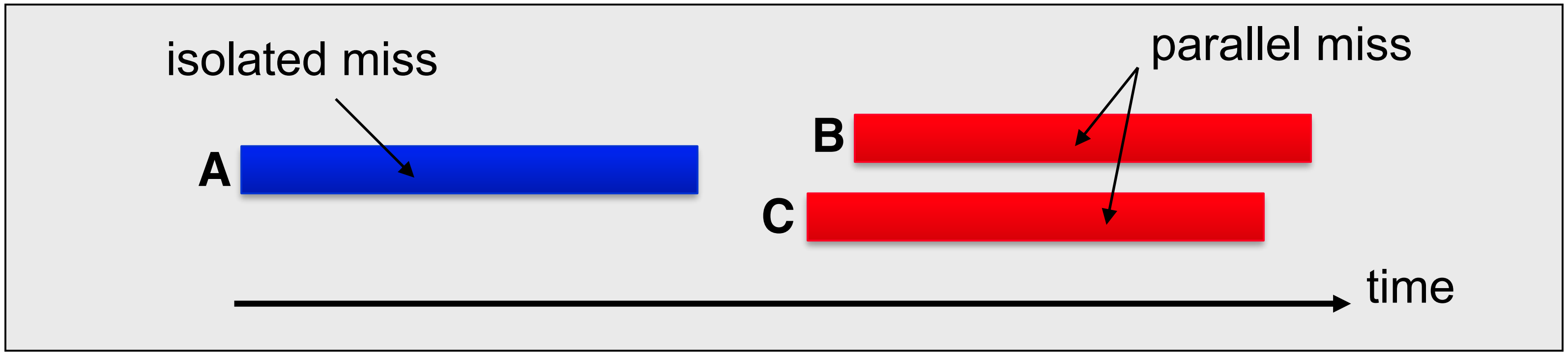
The more the cores, the greater the problem



- Vulnerable to denial of service [Usenix Security'07]
- Unable to enforce priorities or SLAs [MICRO'07,'10,'11, ISCA'08'11, ASPLOS'10]
- Low system performance [MICRO'07,'10,'11, ISCA'08,'11, HPCA'10, ASPLOS'10]

Uncontrollable, unpredictable system

Memory Level Parallelism (MLP)



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

DRAM Memory System Summary

- Its important to match the cache characteristics
  - caches access one block at a time (usually more than one word)

with the DRAM characteristics

  - use DRAMs that support fast multiple word accesses, preferably ones that match the block size of the cache

with the memory-bus characteristics

  - make sure the memory-bus can support the DRAM access rates and patterns
  - with the goal of increasing the Memory-Bus to Cache bandwidth

- And to have a well-tuned, on-chip Memory Controller (or several in the case of multicores)