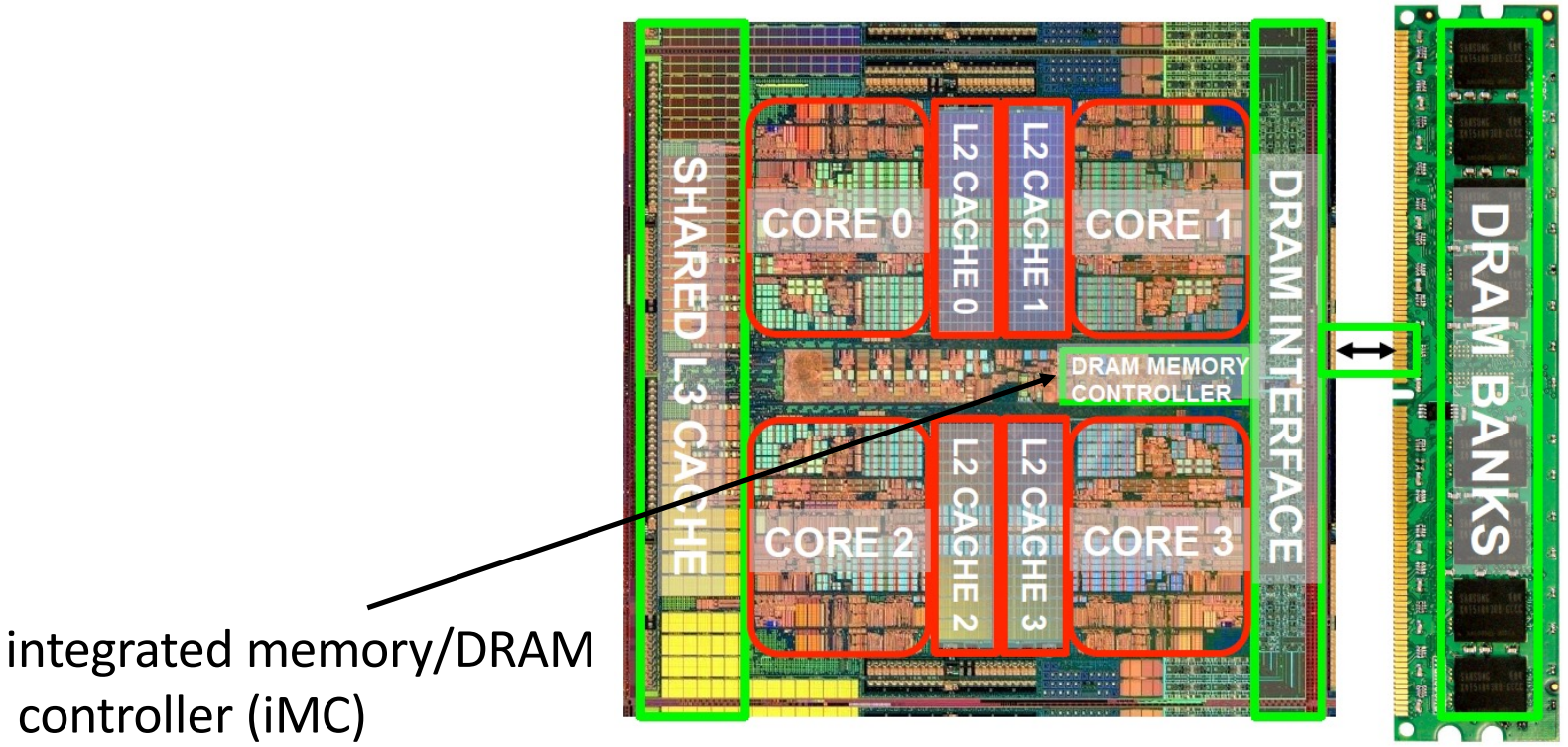
# COMP3710 (Class # 5176)
# Special Topics in Computer Science
# Computer Microarchitecture

Convener: Shoaib Akram
shoaib.akram@anu.edu.au


Australian
National
University

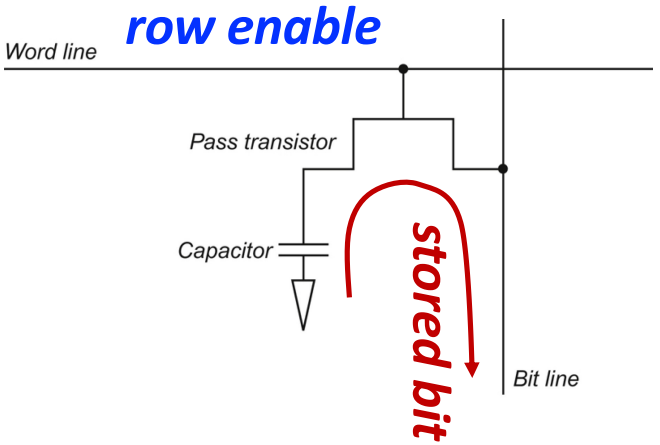
# The Big Picture

- Processor → Exploits ILP and MLP to deliver high performance
- Cache hierarchy → Exploits non-blocking caches enable parallelism
- Main memory → Also exploits parallelism (today)



# DRAM Storage Cell

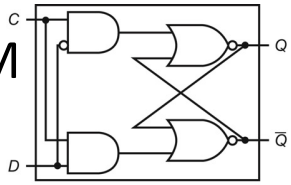
- Dynamic Random Access Memory (DRAM)
  - One pass/access transistor + one capacitor (**1T1C**)
  - Capacitor **holds** the charge (information)
  - Pass transistor **controls** loading/storing charge to/from the capacitor



# DRAM Cell Fact Sheet

- Capacitors lose charge over time and must be **refreshed** periodically (that's why it's **dynamic**)
- DRAM is denser than SRAM (1T1C versus 4T – 8T)
- DRAM has higher latency than SRAM
  - Capacitor takes time to charge/discharge
- DRAM is not compatible with CMOS logic processes
  - SRAM is used for registers & caches
  - DRAM is used for building off-chip main memory
- DRAM vs SRAM energy is complex tradeoff
  - DRAM requires refreshing
  - SRAM's cross-coupled inverters constantly draw current

SRAM Cell

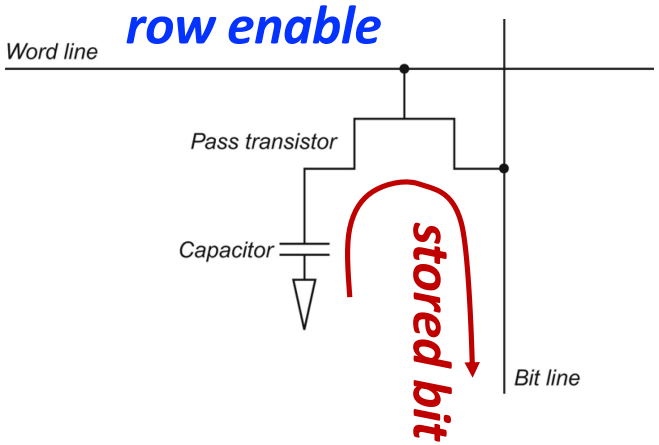


# Memory Wall: Three Aspects

- Capacity
- Latency
- Bandwidth

# Memory Wall: Capacity

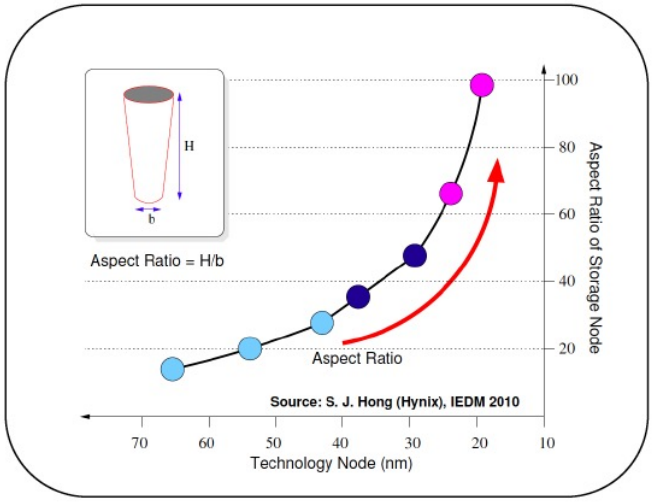
- Below 10 nm node, manufacturing complexity of 1T1C cell is complicated
- Capacitor aspect ratios have increased exponentially
  - Placement & routing issues
  - Increased error rates





**Scaling wall**

DRAM does not scale well to small feature sizes (sub 1x nm)

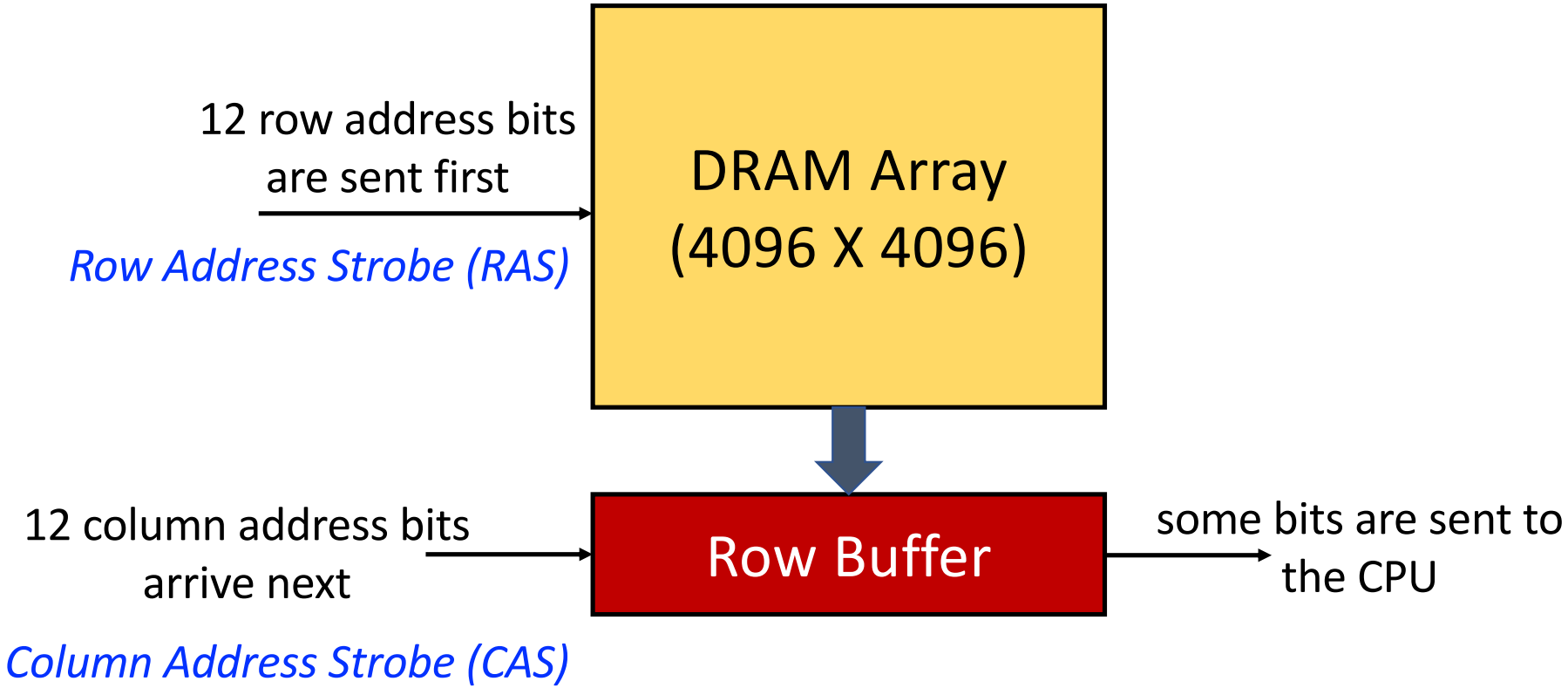


# Memory Wall: Latency & Bandwidth

- **Latency**
  - If 1T1C cell cannot be shrunk easily, then the memory latency cannot reduce anymore
  - The gap between processor & memory performance increases
- **Bandwidth**
  - Bandwidth is limited by the memory bus (interconnect) not scaling proportional to logic
- For OOO processors that exploit MLP both latency & bandwidth are critical
  - High DRAM latency ultimately stalls the OOO machinery
  - Low bandwidth (bytes/sec returned to caches) can also lead to stalls

# Two Level Decoding

- Large memory arrays use two-level decoding
  - Organize memory cells into rows and columns
  - 2D array of 1T1C cells



# Row Buffer

- A DRAM row is also called a DRAM page
  - No relation to the virtual memory page
- A DRAM page (row) is read into a row buffer
  - A collection of flip-flops + circuitry that senses/amplifies the charge on the bit lines
- Sense amplifier is the same as a row buffer
- A DRAM row can be:
  - Open
  - Closed

# Access to a Closed Row

- **Activate command**
  - Decode the address
  - Drive the row select
    - Selected bit cells drive the bit lines
    - Read of entire row into row buffer
    - Sense amplifier amplifies & regenerate the bit lines
- **Read/write command**
  - Read/write the columns in the row buffer (mux out data bits)
  - Restore the capacitor
- **Precharge** command
  - Prepares the DRAM bank for next access
  - Required for access to a different row

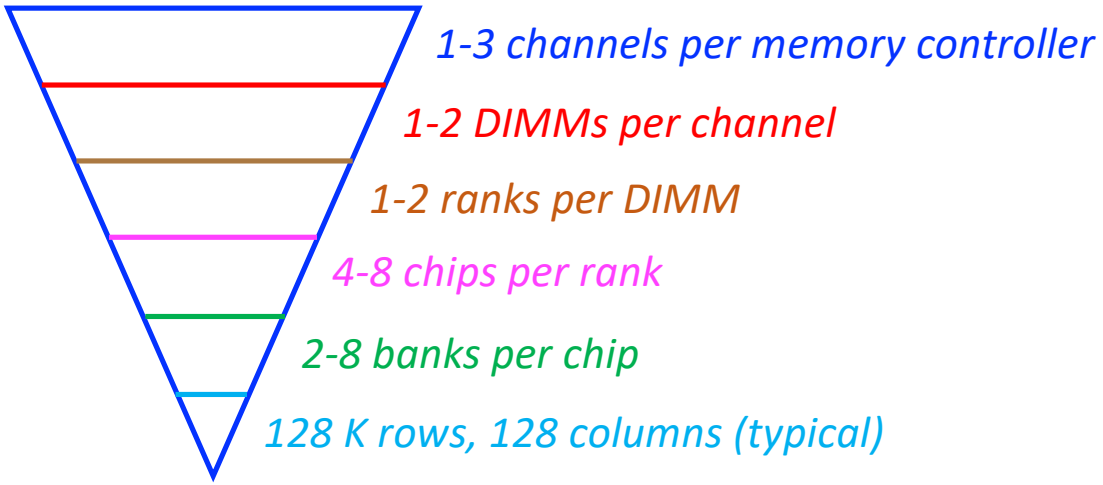
# Access to an Open Row

- No need for activate command

# Memory Organization

DRAM organization hierarchy

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

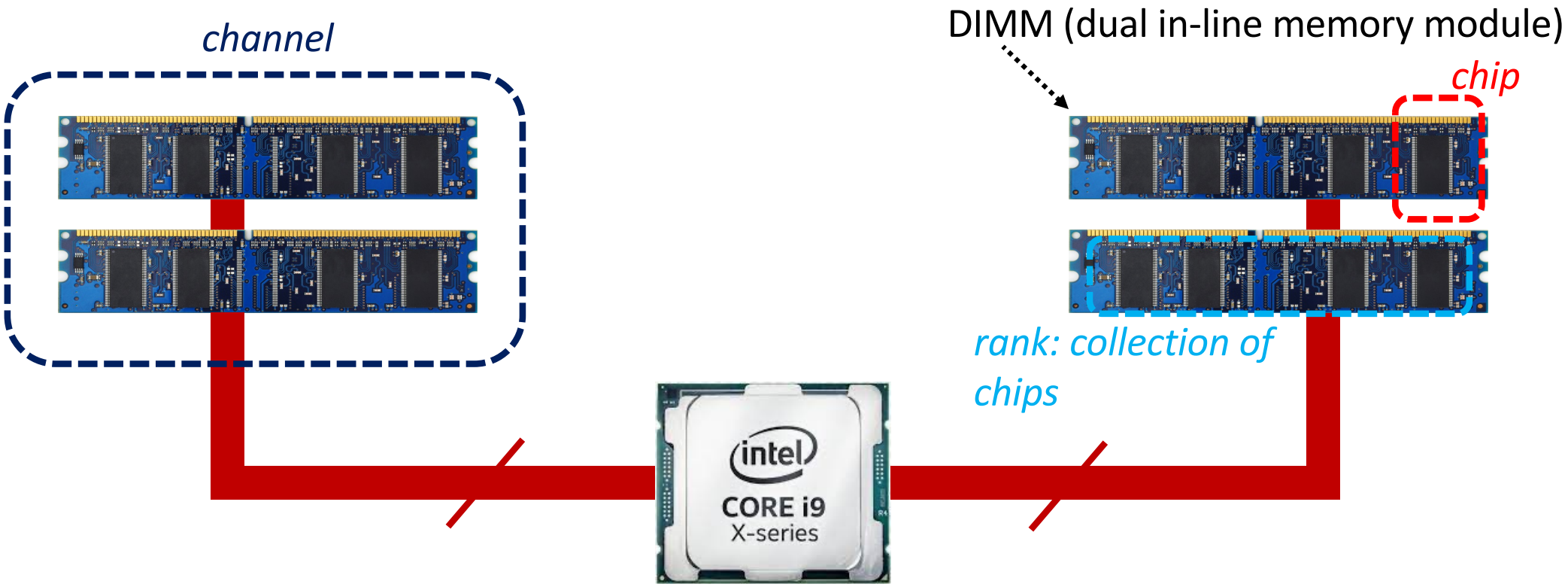



- **What do we need from a well-designed organization?**
  - Memory-level parallelism (MLP)
  - Capacity scaling
  - Low latency (10s of nanoseconds)
    - Limited ultimately by technology, but a good organization helps (via parallelism)
  - High bandwidth
    - Bytes/sec (30 – 50 GB/s typical for servers)

# Memory Organization

- DRAM organization/hierarchy ***delivers*** MLP
  - We have channel, DIMM, rank, chip, and bank-level parallelism
  - Once a bank is locked up, another request needs to wait
- DRAM organization/hierarchy ***delivers*** capacity scaling
  - Scaling up (user): Buy extra DIMMs
  - Scaling up (processor manufacturer): Add more ctlrs/channels to increase bandwidth
  - Scaling up capacity (DRAM manufacturer): Add more banks, increase per-bank capacity, add more rows/columns per bank, store more bits per column

# High-Level Organization

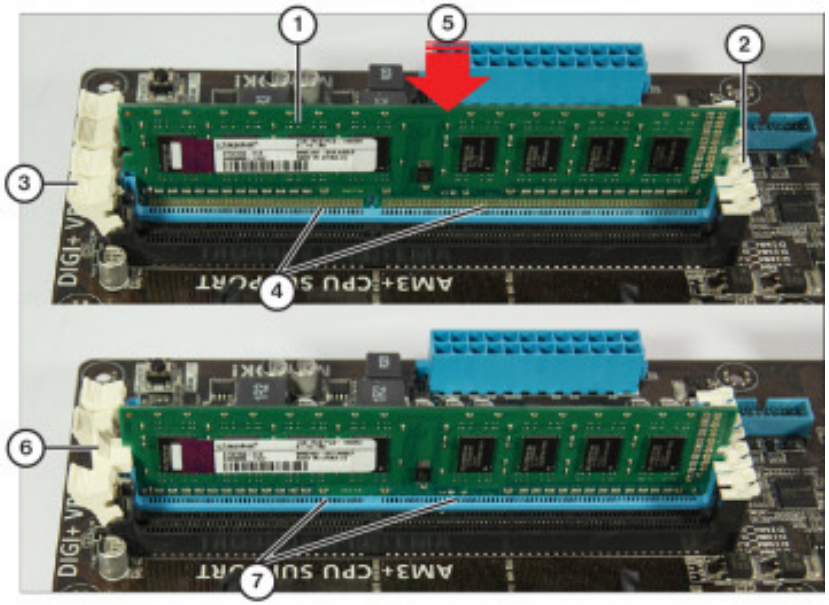



Hypothetical example

- One channel per controller (two controllers)
- Two DIMMs per channel

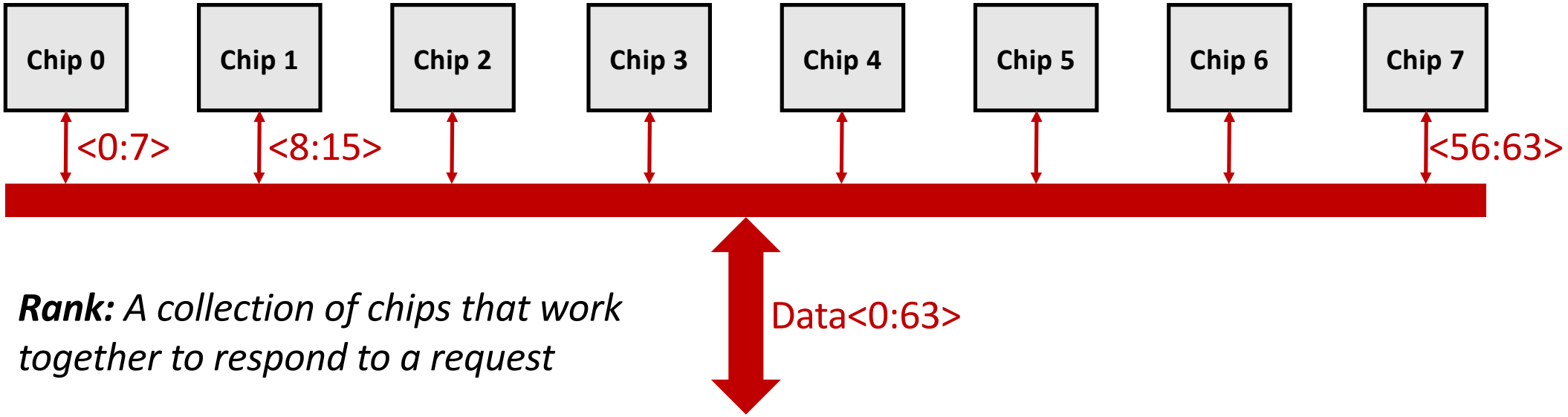
# DIMM Overview

- Front-Side Rank
- Back-Side Rank

# Rank (with x8 chips)

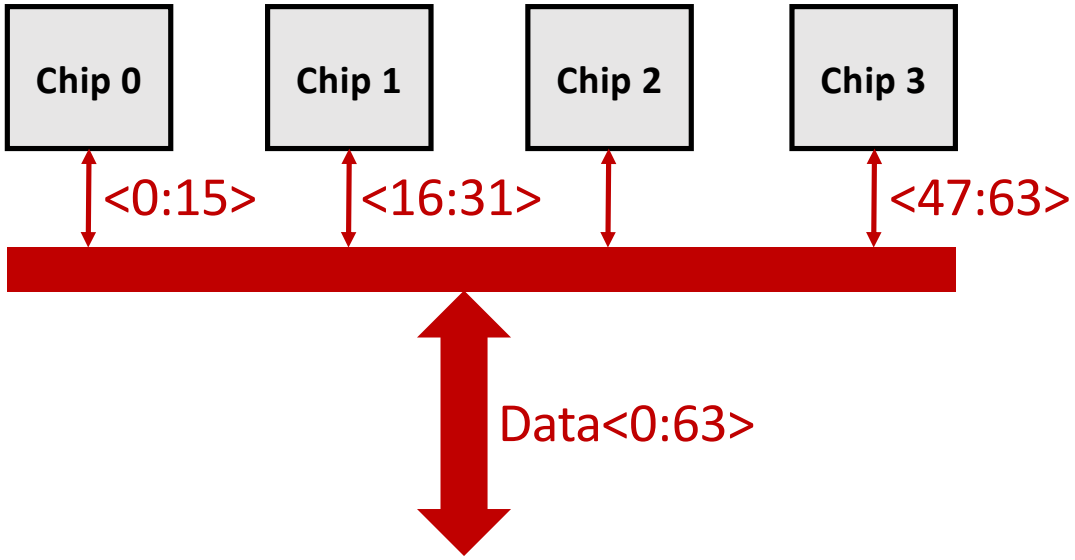
- In x8 DIMMs, there are eight DRAM chips in a single rank
  - Here, x8 refers to the width of the data bus per chip
  - A 64-bit word is interleaved across eight chips
  - Each chip delivers 8 bits out of 64 bits
- Typical organization for high-end servers


***Rank:*** *A collection of chips that work together to respond to a request*

# Rank (with x16 chips)

- In x16 DIMMs, there are four DRAM chips in a single rank
  - Here, x16 refers to the width of the data bus per chip
  - A 64-bit word is interleaved across four chips
  - Each chip delivers 16 bits out of 64 bits
- Used in space-limited scenarios
  - Handhelds (scaling up capacity is a problem)

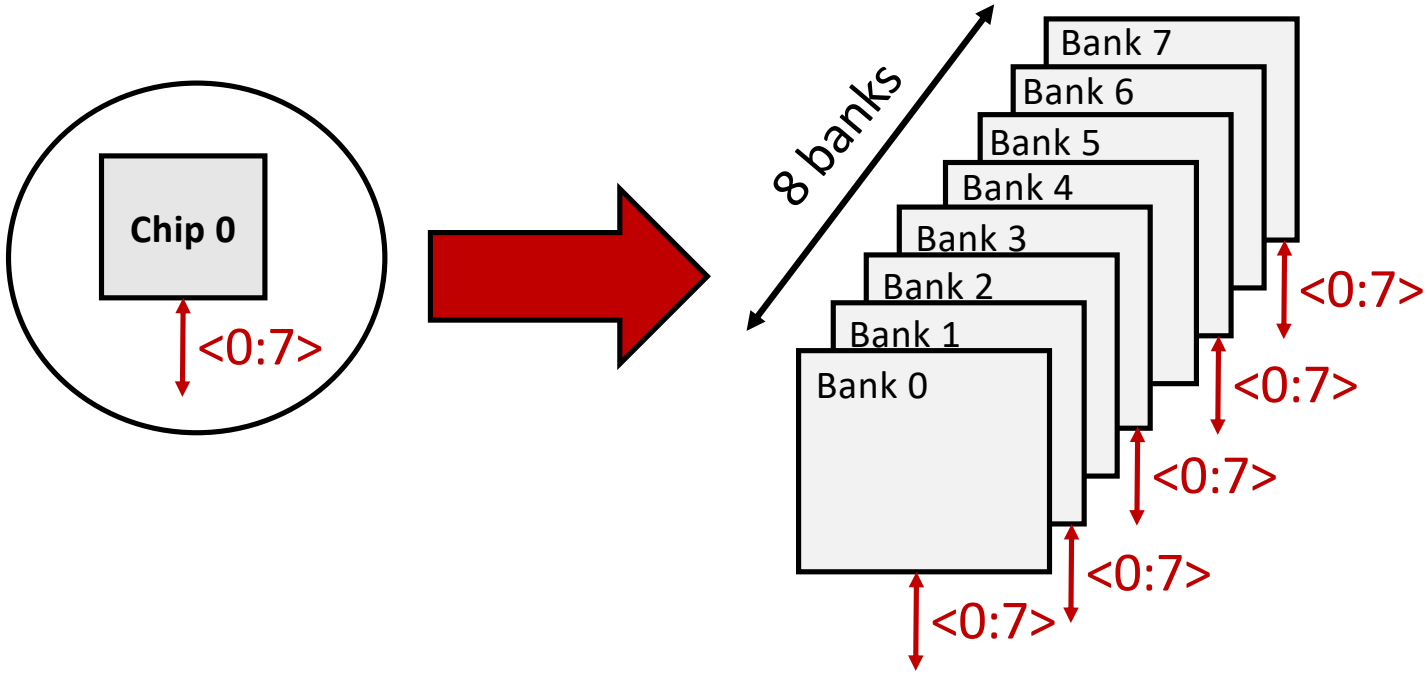



# Rank (with x4 chips)

- In x4 DIMMs, there are sixteen DRAM chips in a single rank
  - Here, x4 refers to the width of the data bus per chip
  - A 64-bit word is interleaved across sixteen chips
  - Each chip delivers 4 bits out of 64 bits
- Used in scenarios where very high capacity is a requirement
  - Very high-end servers
  - *Signal integrity is a problem*

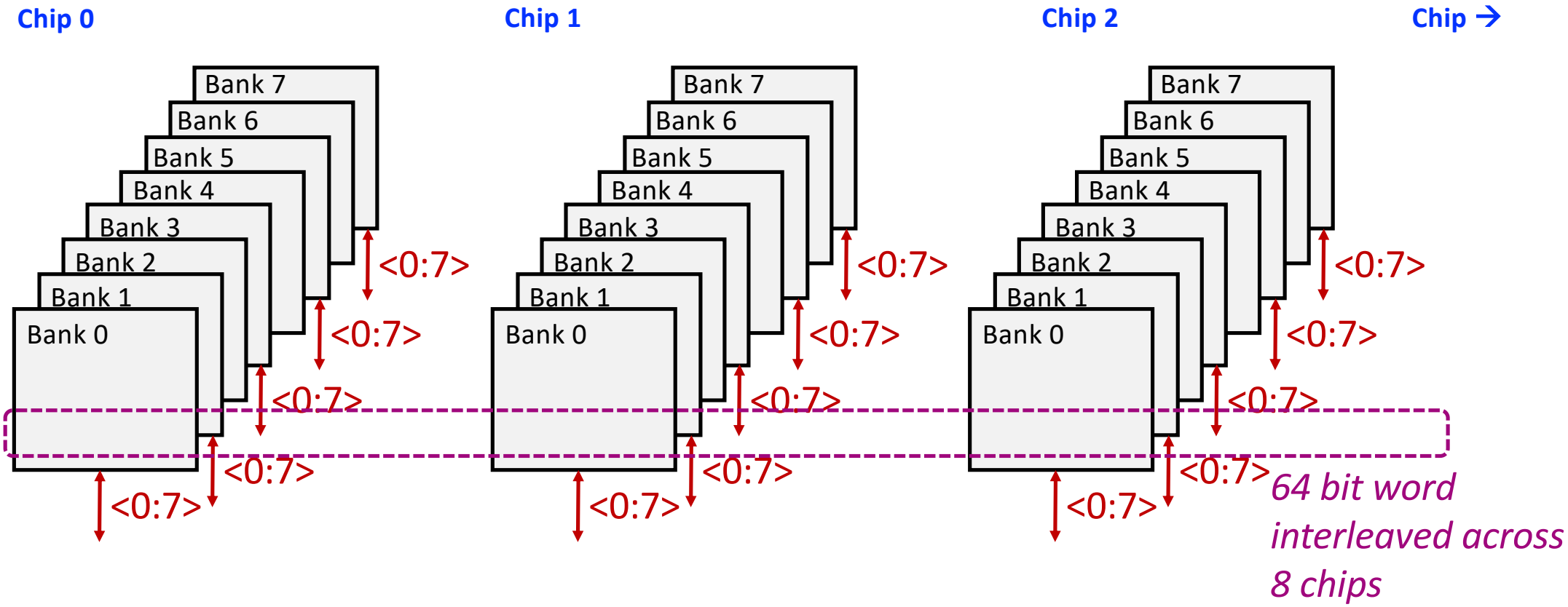
# Breaking Down a Chip

- **Bank-level parallelism:** *Different banks serve different requests concurrently*
- The DRAM chip (e.g., x8) only has eight data output pins, so there is some delay due to bank-level sharing of pins
- *There is one row buffer per bank in the chip*

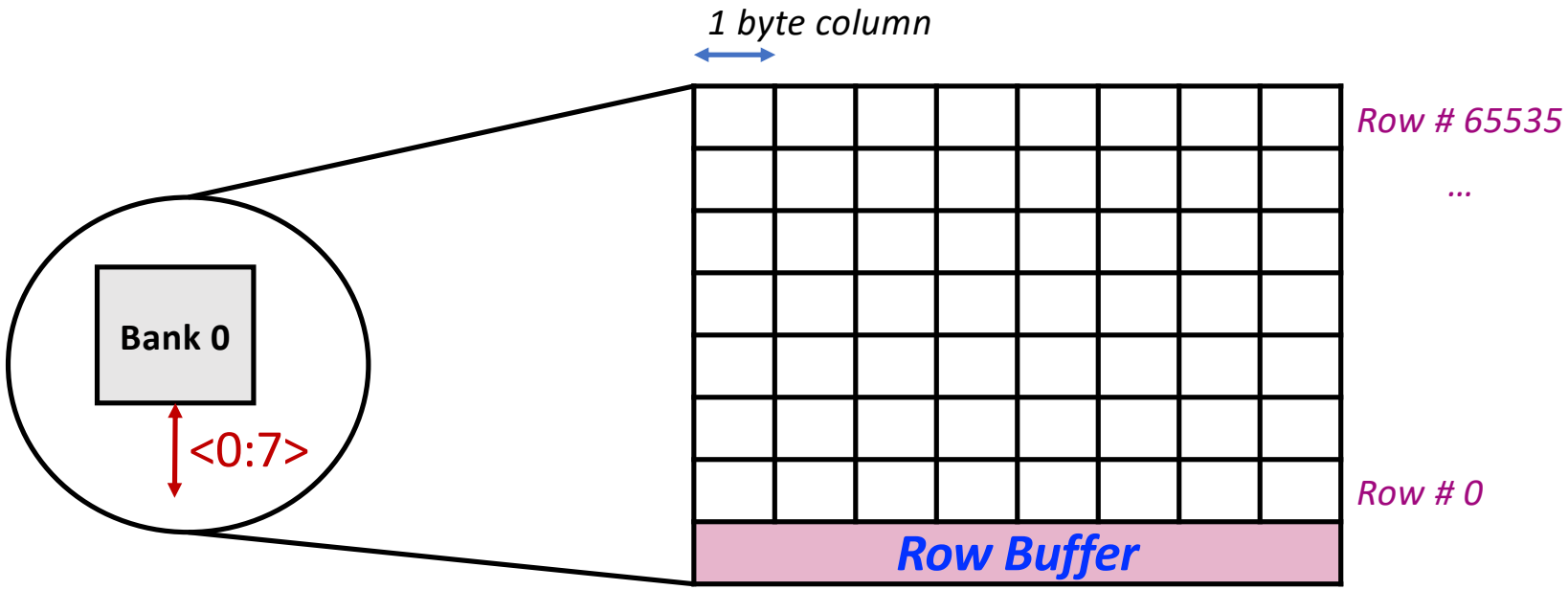


# Breaking Down a Rank of Chips

Each bank (e.g., Bank 0) in chips<0:7> receive the same address request and respond with eight bits of the 64-bit word (chip 0 with bits<7:0> and chip 1 with bits<15:8> and so on

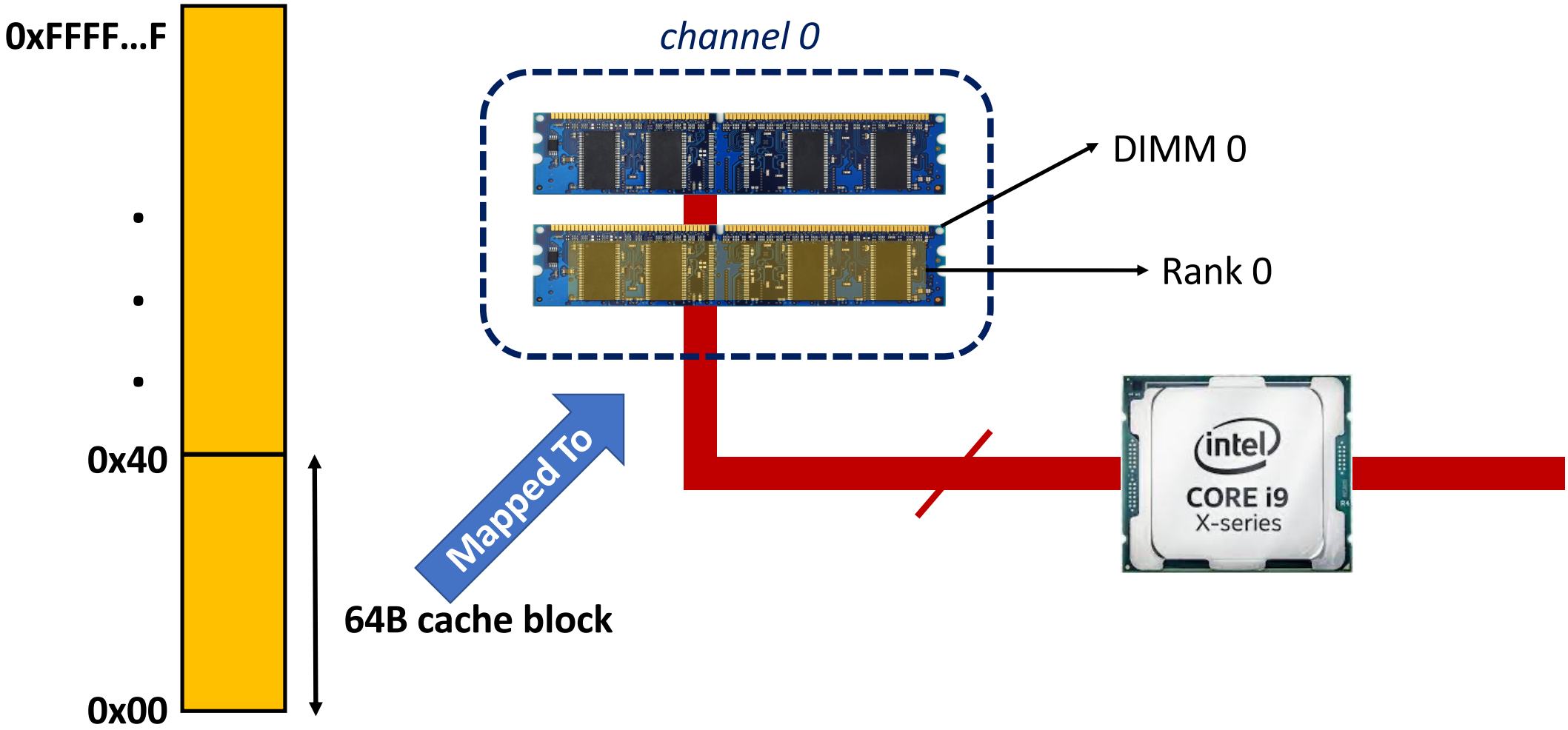


# Breaking Down a Bank

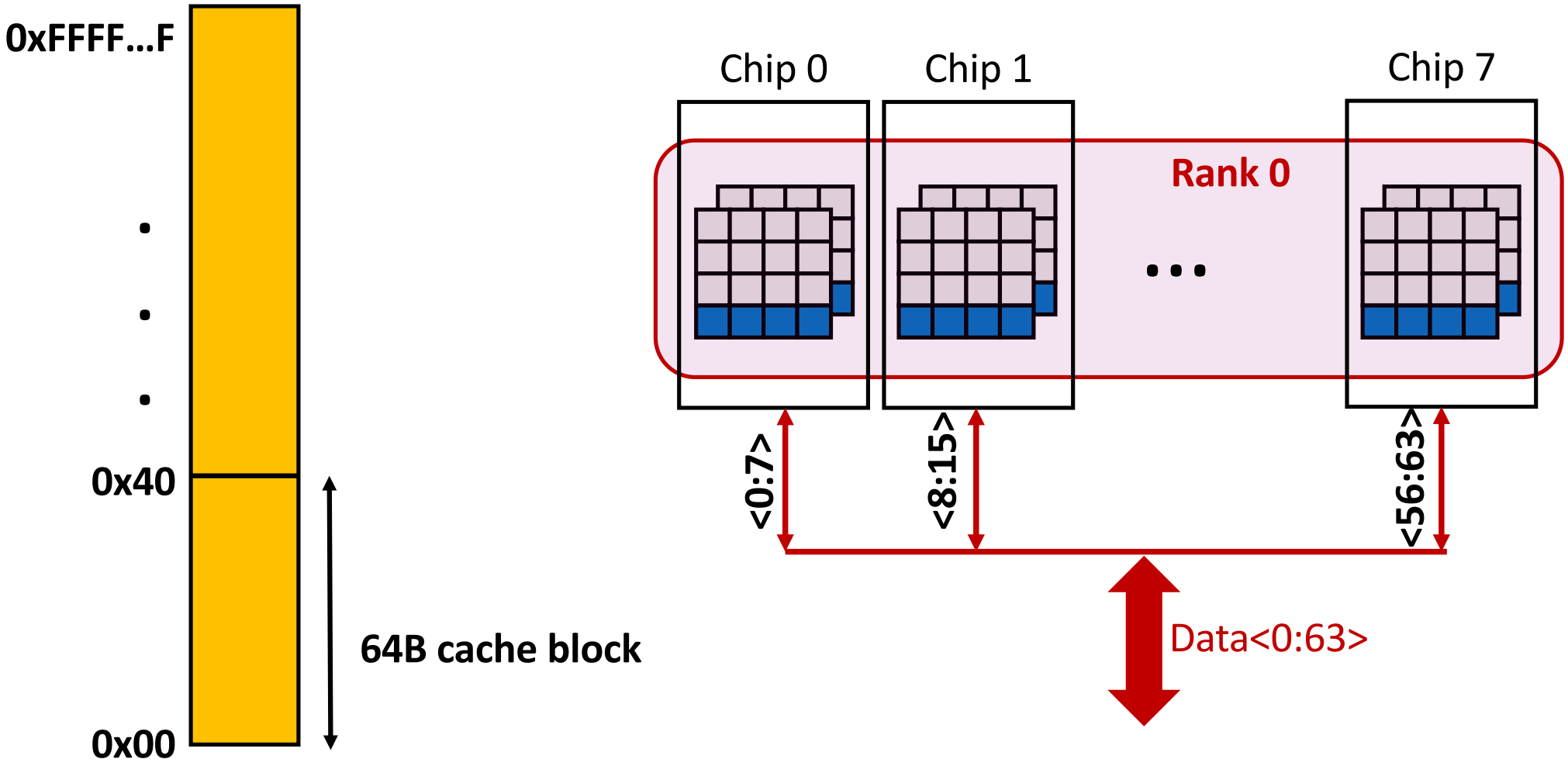



- Terminology: Row is also called an array or a DRAM array
- Bank is made up of multiple *wide* arrays (rows)
- Multiple arrays are also referred to as mats (*matrix of rows and columns*)

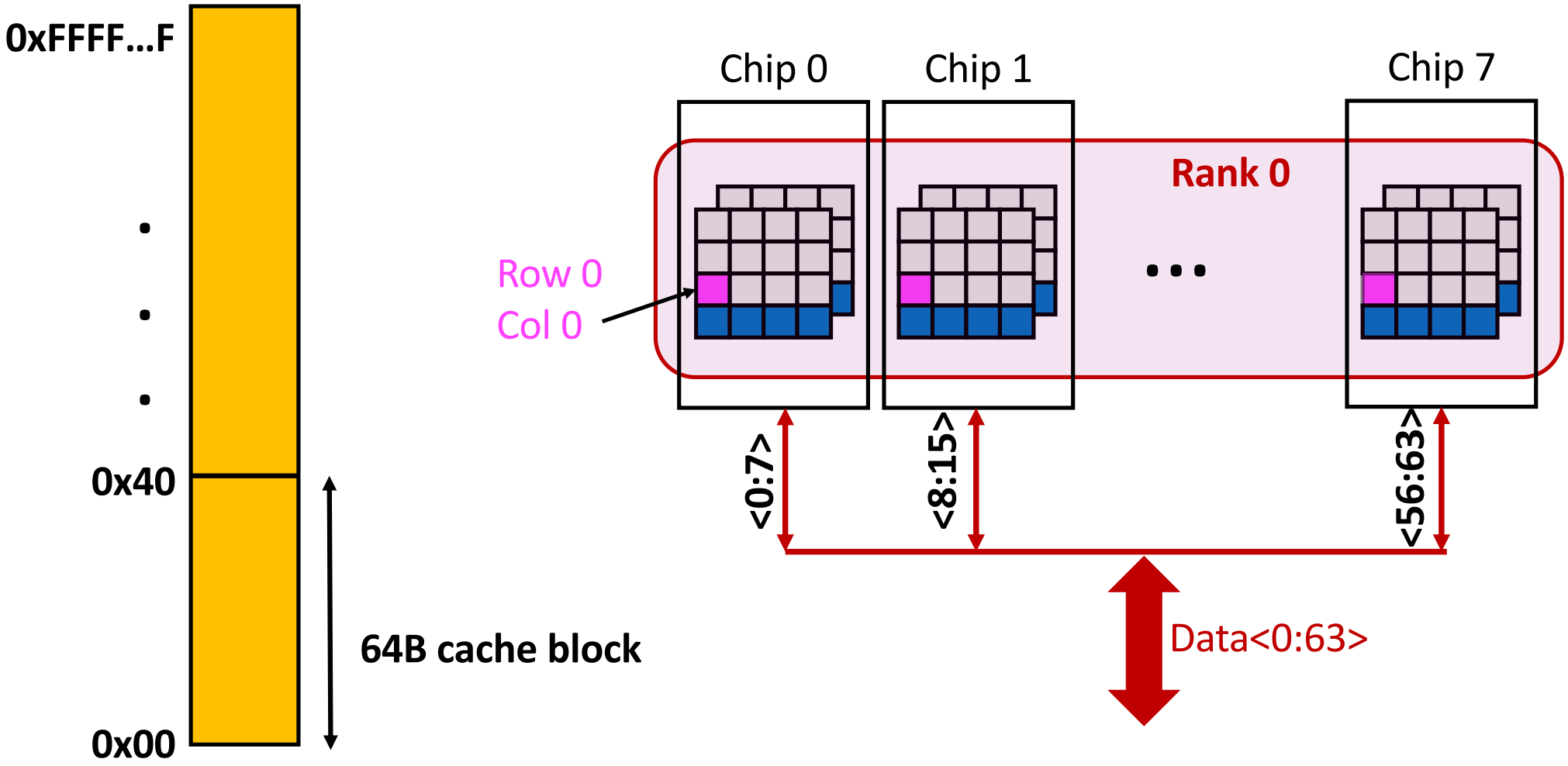
# Example: Transferring a Cache Block



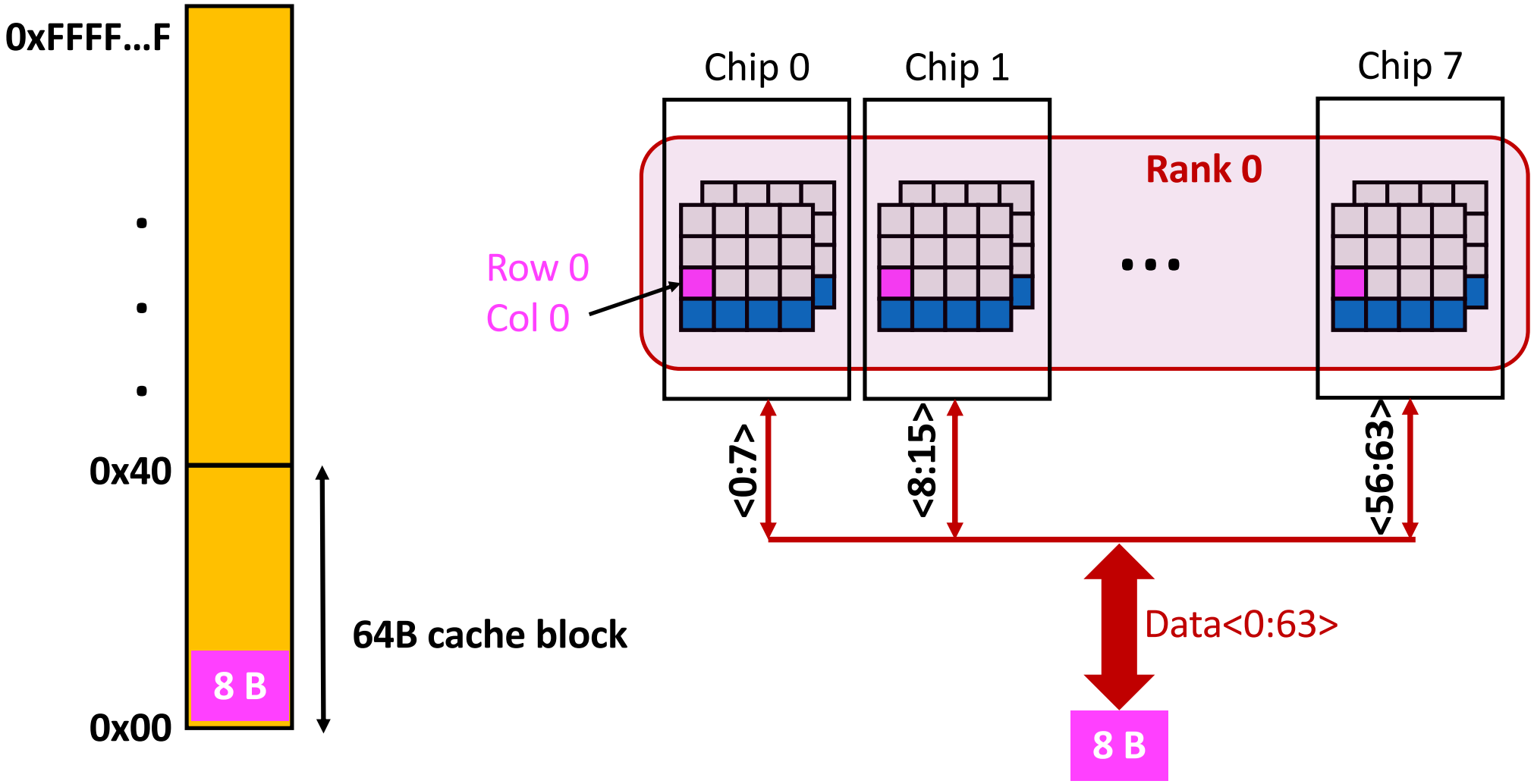
# Example: Transferring a Cache Block



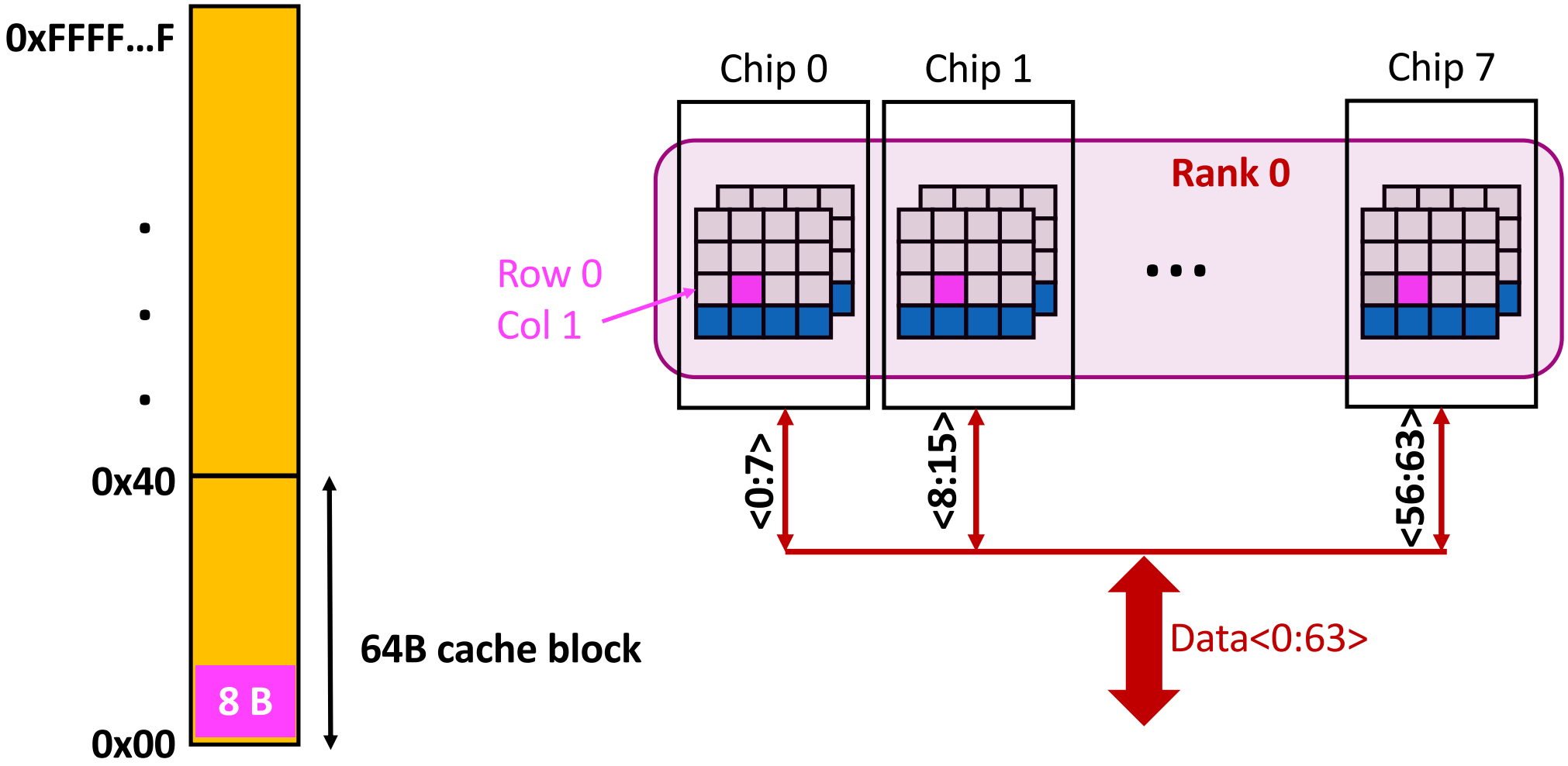
# Example: Transferring a Cache Block



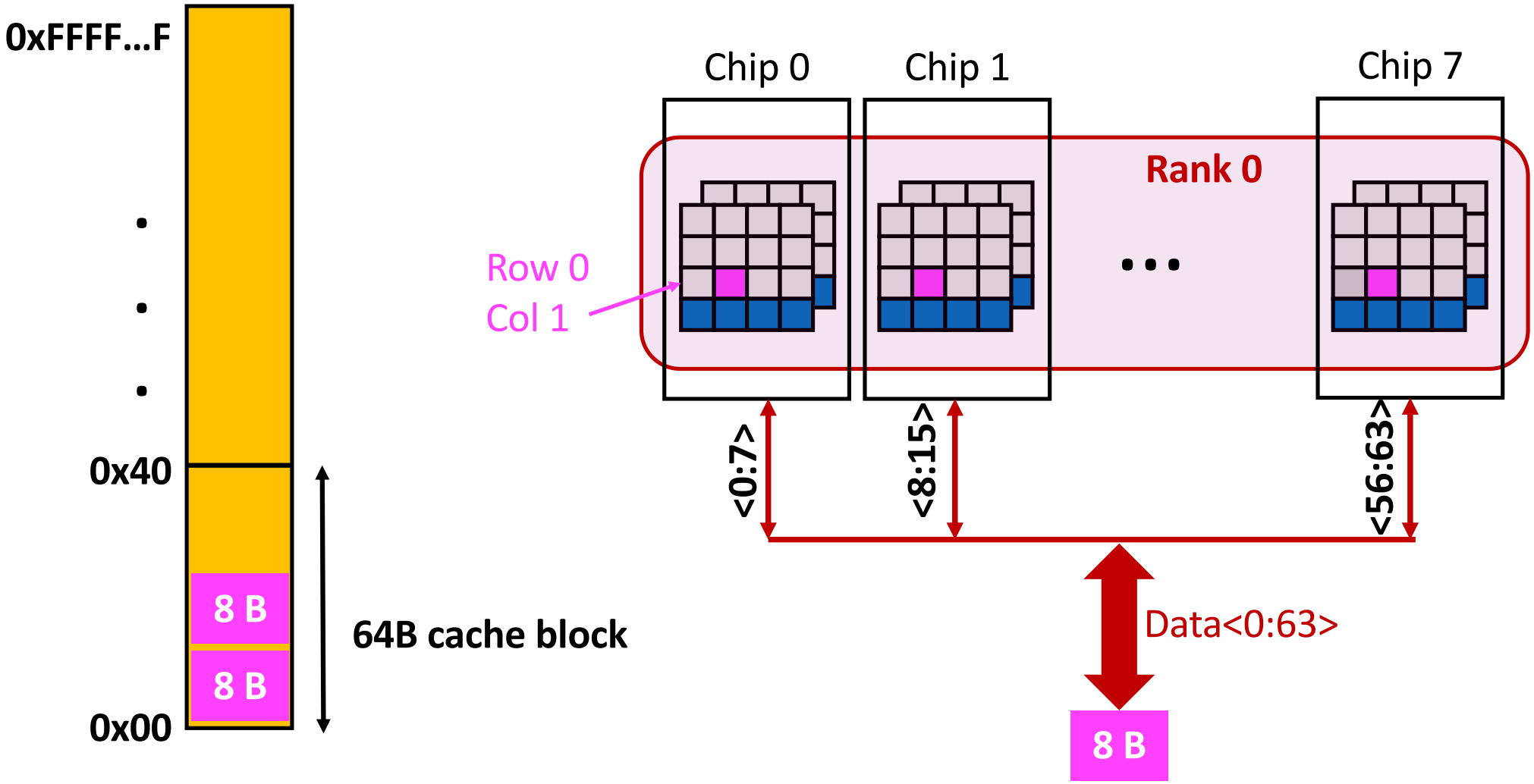
# Example: Transferring a Cache Block



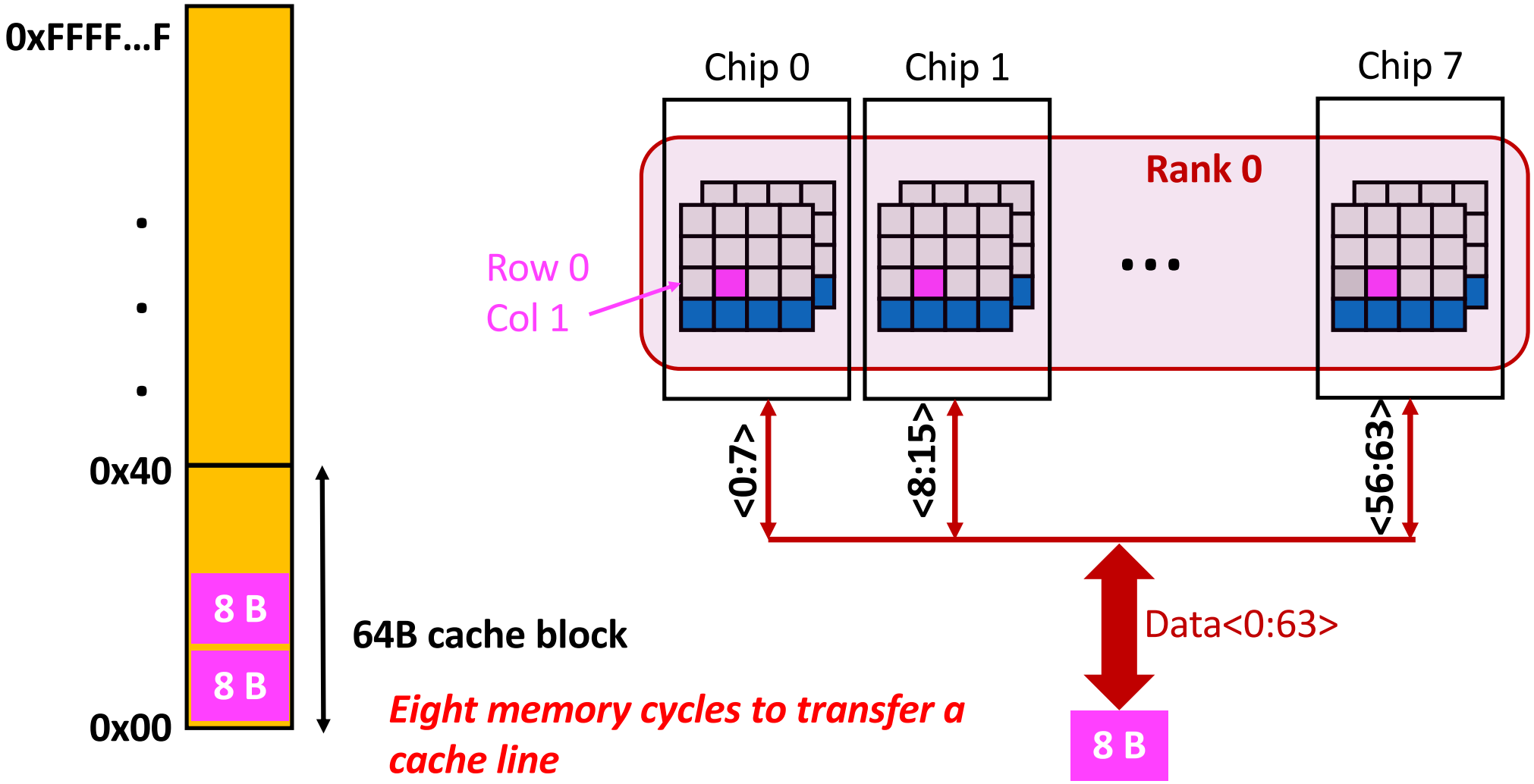
# Example: Transferring a Cache Block



# Example: Transferring a Cache Block



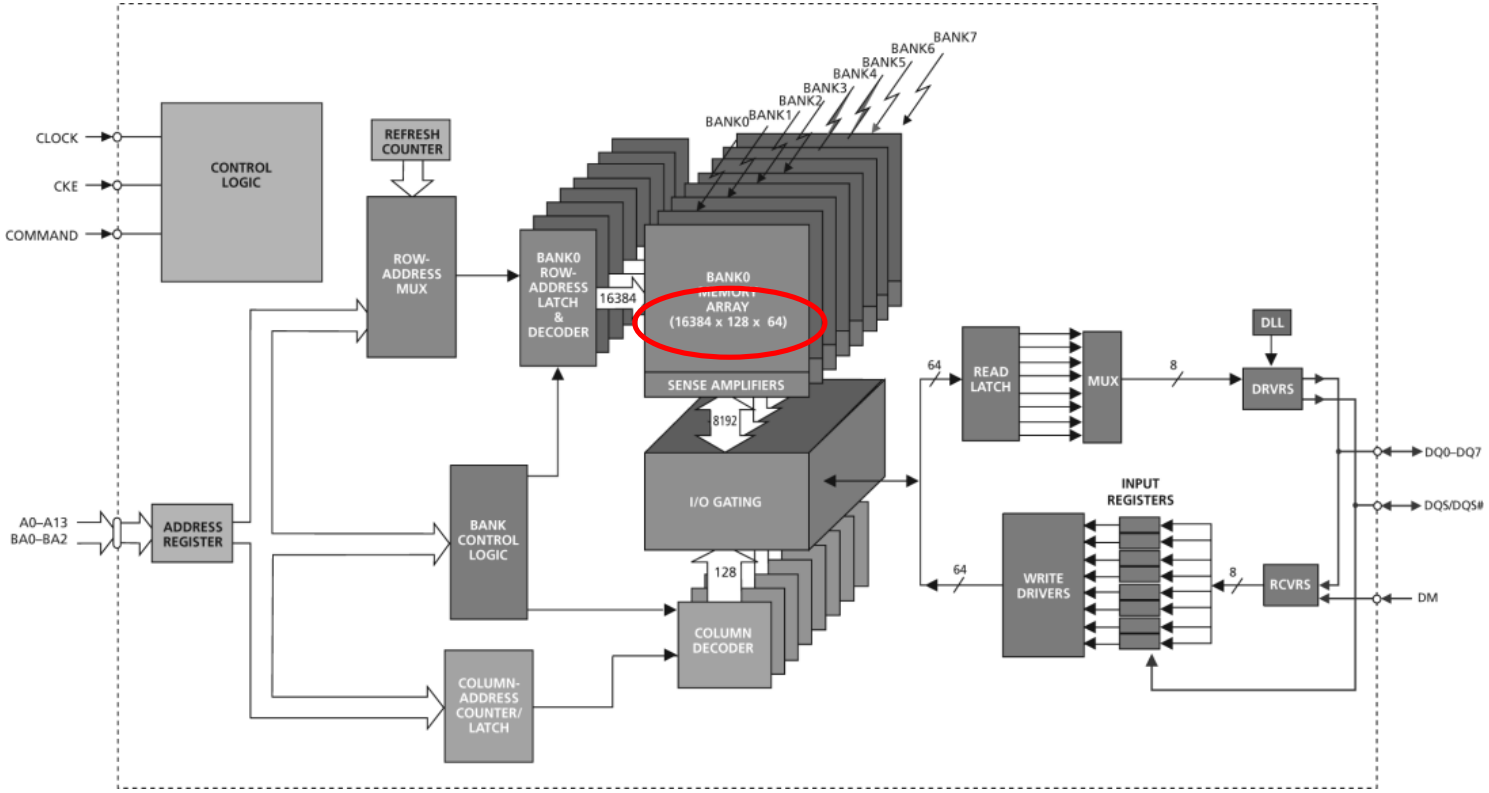
# Example: Transferring a Cache Block



# Organization Tradeoffs

- DIMM, rank, bank, array → form a hierarchy
- Electrical constraints govern DIMM count attached to a bus (channel)
  - Typicall a few (1 – 2)
- One DIMM can have 1 – 4 ranks
- When do we use wide-output DRAM chips?
  - Energy efficiency: Activate only 4 x16 chips instead of 16 x4 chips
- When do we use narrow-output DRAM chips?
  - Capacity scaling: 16 x4 delivers larger capacity than 4 x16

# Micron 128 M x8 DRAM Chip




- There are 64 bits per colum in a bank (think 3D) instead of just 8
  - Send 8 bits out at a time. Read 64 bits from array (8n prefetch)
- 8 x8 chips → 64 B cache line

# Some Resources

*Reading data sheets and understanding engineering details and tradeoffs*

https://en.bmstu.wiki/index.php?title=DDR3_SDRAM&mobileaction=toggle_view_mobile
https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn41_01ddr3_power.pdf
https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/16gb_ddr4_sdram.pdf
https://www.youtube.com/watch?v=w2bFzQTQ9aI&ab_channel=ActuallyHardcoreOverclocking

# Row Buffer Mgmt. Policies

- Two policies for managing the row buffer
  - **Open page:** Keep the current row/page open
  - **Closed page:** Precharge the bit lines right away

# Open Page Policy

- **Access to a closed row**
  - PRECHARGE command closes the row and prepares the bank for the access
    - This precharge is on the critical path because it precedes read/write access
  - ACTIVATE command opens the row (bring data into the row buffer)
  - READ/WRITE command accesses the column in the row buffer

- Access to an open row
  - **READ/WRITE** command accesses the column in the row buffer

- If an access stream has a high locality, open page policy is helpful
  - Row buffer hits are cheap (it's made out of SRAM cells)
  - Row buffer miss is expensive because precharge is now on the critical path

# Closed Page Policy

- Access to a closed row
  - **ACTIVATE** command opens row (→ row buffer)
  - **READ/WRITE** command accesses the column in the row buffer
  - **PRECHARGE** command closes the row and prepares the bank for the next access
    - This precharge can happen in the background and is not on the critical path

- If an access stream has low locality, precharging the bit lines immediately after access is helful for performance

Modern memory controllers use (proprietary) policies somewhere between the two extremes

# DRAM Latency: Five Components

1. CPU to DRAM controller (request) transfer time after a load request misses everywhere in the cache hierarchy
2. DRAM Controller latency
   - Queuing and scheduling delay
   - Load/store request translation to DRAM COMMANDS
3. Request transfer time from the DRAM Controller to the selected channel/DIMM
4. Bank latency (*if there is no conflict*)
   - Row buffer hit: 20 ns (move data from row buffer to output pins)
   - Row buffer miss: 60 ns (PRECHARGE + READ + move data to pins)
   - Empty row buffer: 40 ns (READ + move data to pins) *closed page policy, precharge immediately*

   *open page policy*
5. Other delays
   - Bank conflict
   - Respecting timing constraints
   - Interconnect/wire

# Address Mapping

**Problem:** We have a 32-bit address. We need to decide the bits we use for selecting the channel, bank, rows, column.

Hypothetical Example:

32-bit address 

Example solution: Use bits <3:5> for selecting one of the eight banks
Use bits <X:Y> for selecting one of the R rows

**Bottomline:** Depending on the address mapping, contiguous chunks of program data in virtual memory can end up in different banks or the same bank. Recall accesses across banks benefit from bank-level parallelism.

# Two Concrete Policies

**Assumptions:** One channel, x8, 2 GB, 8 banks, 16 K rows, 2K columns
The channel bit is assumed 0 in the scenarios below (only 31 bits shown)

## Row interleaving

- Large contiguous chunks (rows) of program data in consecutive banks
- Start filling the row and move to a different bank after $2^{14}$ bytes (16 KB)


## Cache block interleaving

- Consecutive cache blocks (64 B) in consecutive banks
- Place eight 64-bit words (see first 6 bits below) in a one row/bank and then move to a different bank (see next three bits)

# Interaction with Virtual Memory

- Operating system (OS) influences where a virtual page ends up in DRAM

- OS can place consecutive virtual pages in the same row or different rows in different banks

# DRAM Refresh

- Leaky capacitor must be refreshed
  - Typical interval is 64 ms
  - ACTIVATE + PRECHARGE each row in a bank every 64 ms
  - Responsibility of the memory controller
- Implications for performance
  - DRAM bank is unavailable while refreshed
  - Program can experience long pause times
  - Optimizations in literature

# Memory/DRAM Controller

- **Placement**
  - Today: on the processor die (integrated memory controller or iMC)
    - Low latency but consumes extra die area
    - DRAM standards and processor must evolve together
  - Old days: outside the processor die (part of the chipset)
    - Decouple DRAM standards/types from the processor
    - Higher access latency
- **Functions**
  - Respect DRAM timing constraints & ensure correct operation (refresh)
  - Two popular scheduling policies
    - **FCFS (first come, first serve)**
      - Issue the first read/write in the queue that is ready for issue
    - **FR-FCFS (first ready-first come, first server)**
      - First, prioritize row buffer hits, then prioritize oldest requests

# Cutting-Edge Research

- **Emerging memory technologies**
  - Phase-change memory (PCM)
  - Carbon nanotubes
  - Spin-torque transfer RAM (STT-RAM)
- **Processing in memory (PIM)**
  - Old approach: Bring data close to the processor
    - Data movement is slow and consumes energy
  - Next “big” thing: Take processing where the data is
- **Fast storage**
  - PCIe NVMe SSDs are closer to the processor
  - Can serve our needs for capacity expansion
- All active areas of research in my group: https://shbakram.github.io/
- The End!