Module 9 – Main Memory

(ECE 209) Secure and Advanced Computer Architecture

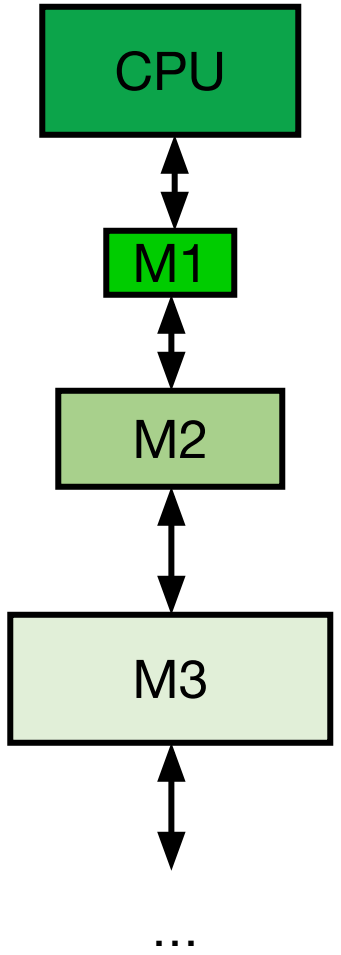
Nader Sehatbakhsh
Department of Electrical and Computer Engineering
University of California, Los Angeles

Review

- Out-of-order execution
- Memory hierarchy, cache design
- Paper reading and presentations
- and …

Memory Hierarchy

★ ***Idea:*** *have multiple layers of memory to balance between size and speed!*



Summary

- Miss Rate
  - Increase block size
  - Increase associativity
  - Increase cache size
  - Prefetching
  - Victim cache
  - Compiler
  - Replacement Policy
- Miss Penalty
  - Write buffer
  - Early restart with critical block first
  - Adding more levels
  - Sub-blocking
- Hit Time
  - Set associative cache
  - Add more levels
  - Parallel lookup
  - Speculative loads

Replacement Policy

- Strategies to maximize hit rate.
- The basic idea is that we should evict something that is least useful.
- Invalid lines first, but what if are lines are valid and potentially useful?

Three Main Questions

- How to *select* which line to evict?

- What position to *install* the new line
  - LRU
  - MRU
  - Somewhere in between?

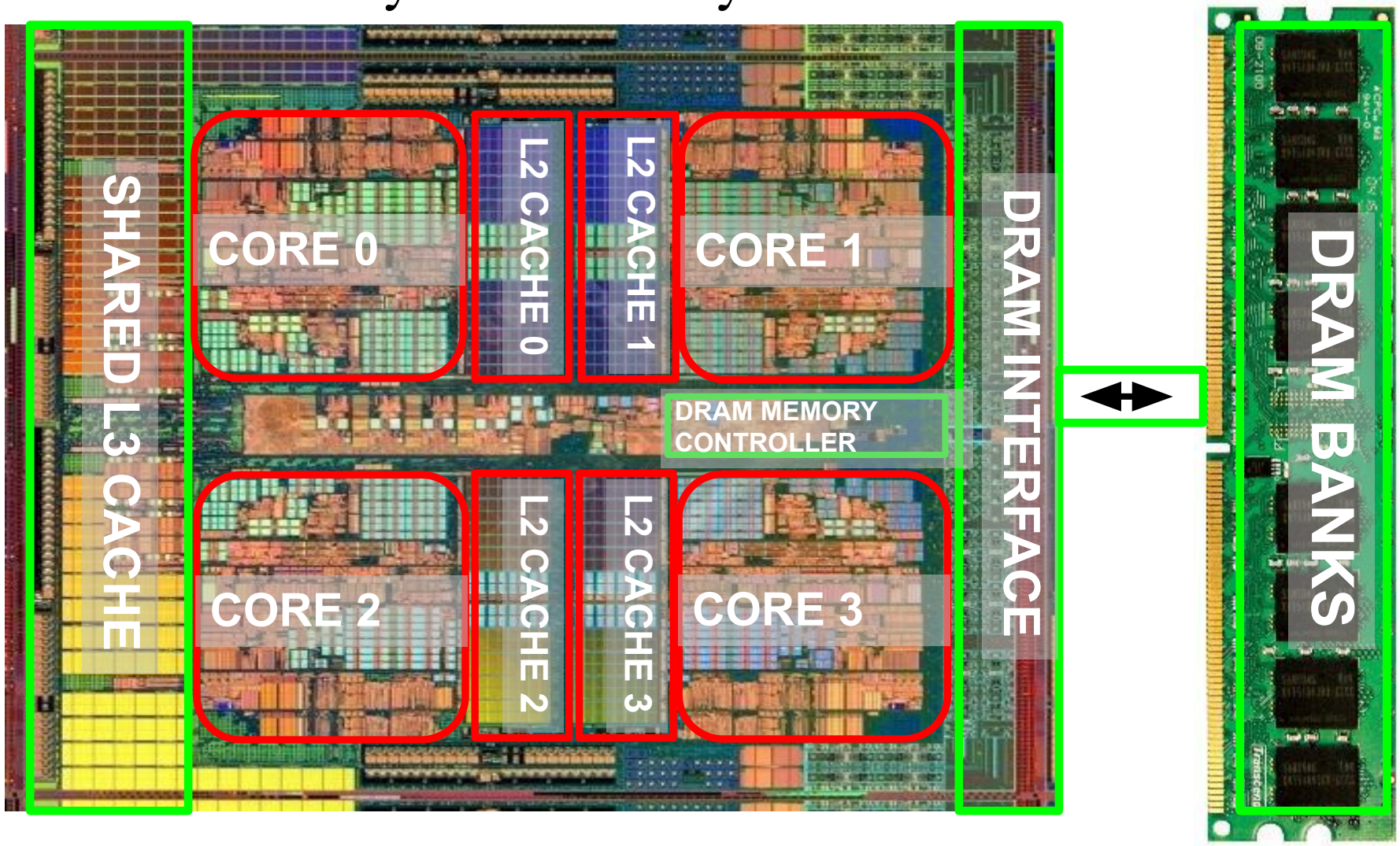
- What to do on a hit?
  - *Promote?* how much?

ECE 8873A
Advanced Topics in Memory Systems

Prof. Moinuddin Qureshi
Georgia Institute of Technology

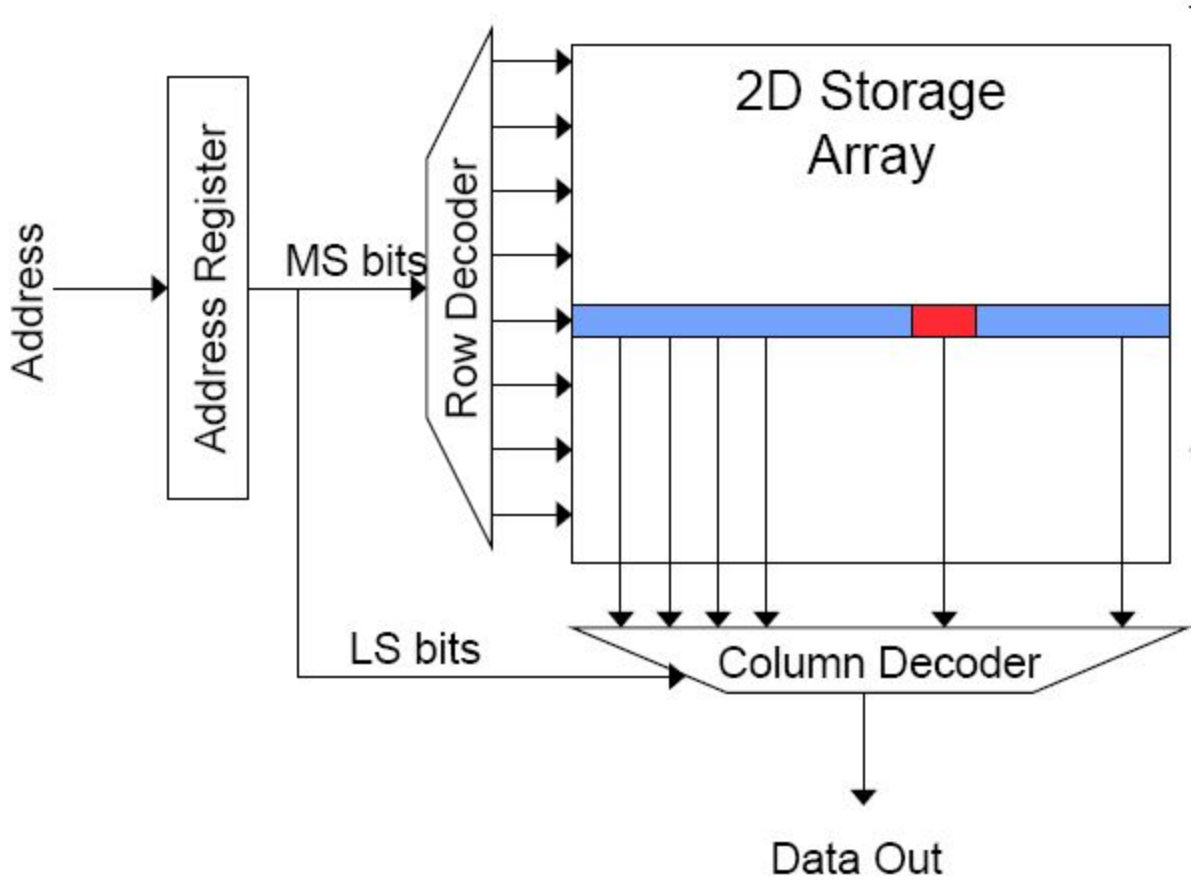
-Thanks to Prof. Mutlu (CMU)

Main Memory in the System



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Memory Bank Organization



- Read access sequence:

1. Decode row address & drive word-lines

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

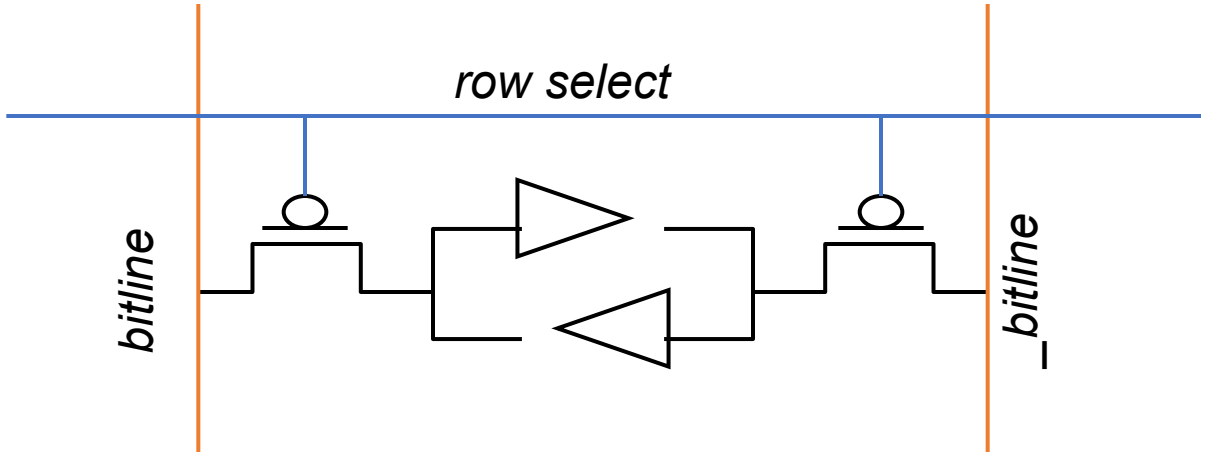
3. Amplify row data

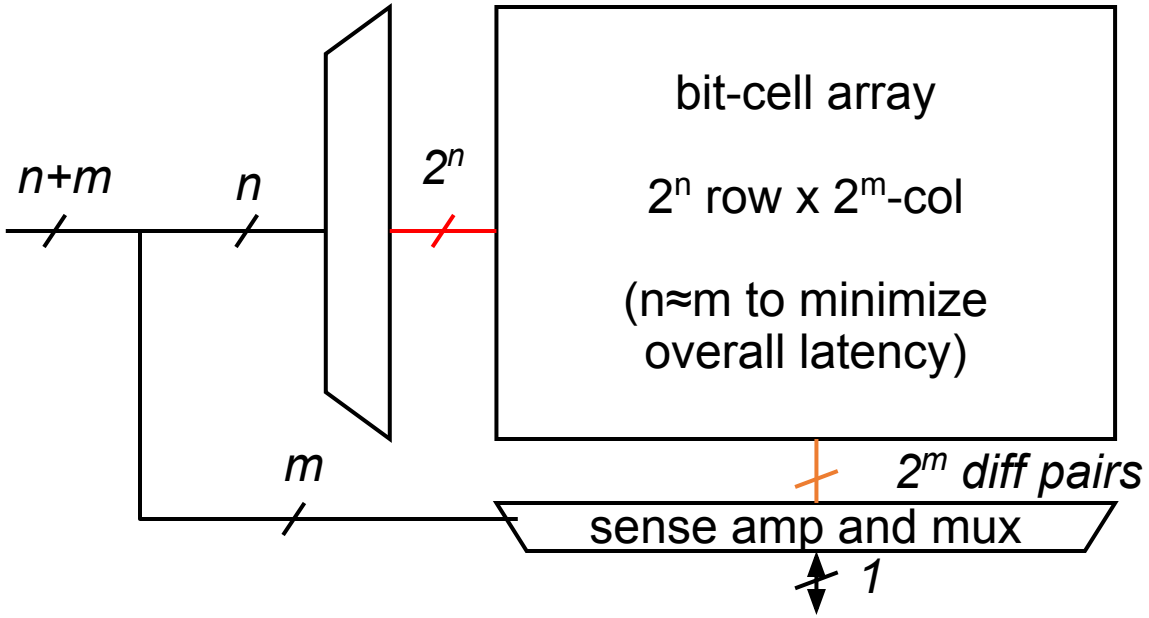
4. Decode column address & select subset of row
   - Send to output

5. Precharge bit-lines
   - For next access

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

SRAM (Static Random Access Memory)





Read Sequence

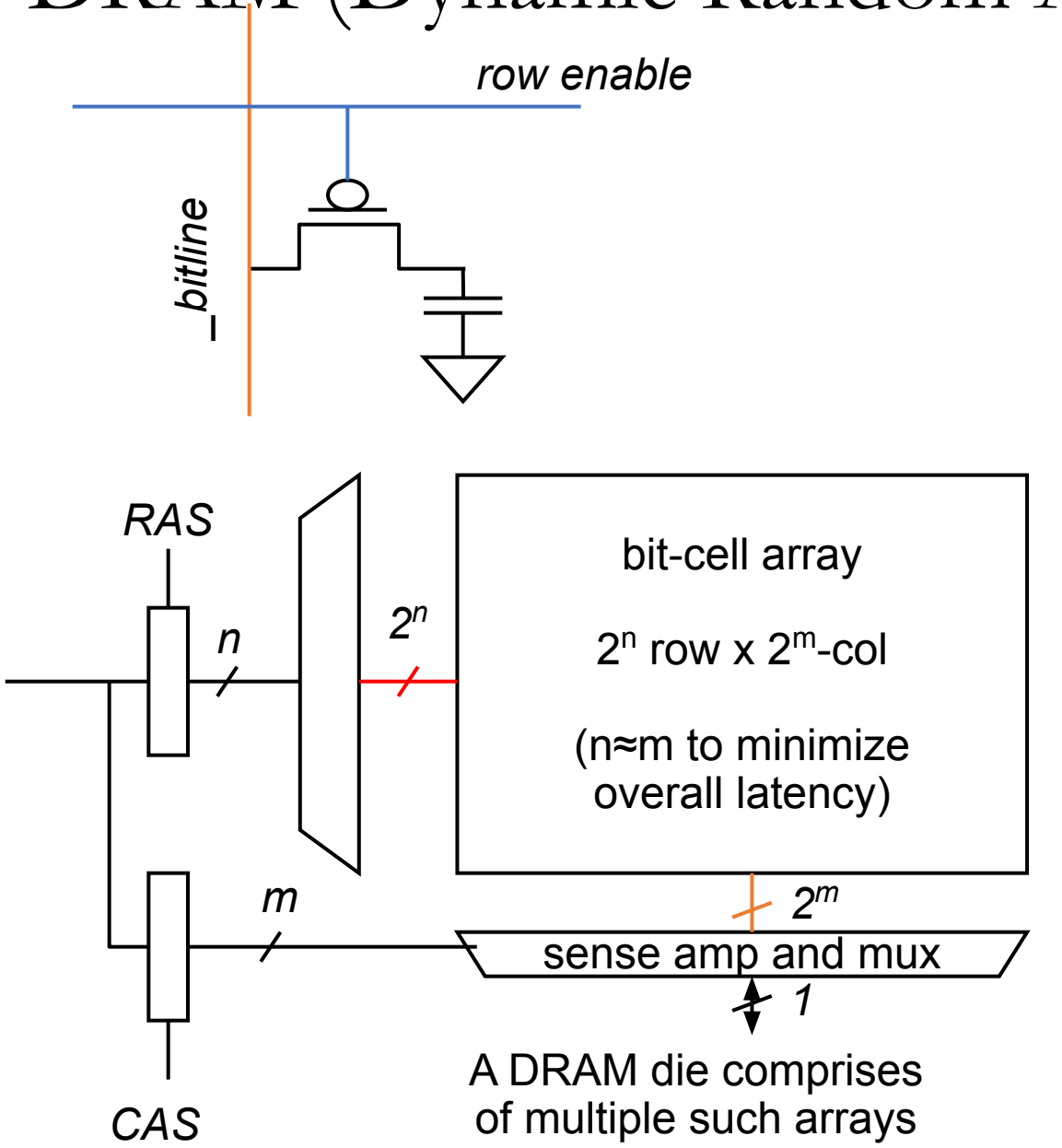
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. diff. sensing and col. select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

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

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

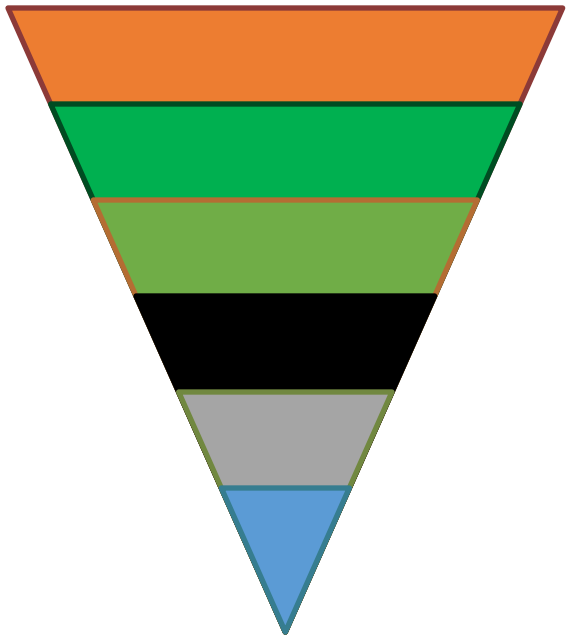
Refresh: A DRAM controller must periodically read all rows within the allowed refresh time (10s of ms) such that charge is restored in cells

SRAM vs. DRAM

- SRAM is preferable for register files and L1/L2 caches
  - Fast access
  - No refreshes
  - Simpler manufacturing (compatible with logic process)
  - Lower density (6 transistors per cell)
  - Higher cost
- DRAM is preferable for stand-alone memory chips
  - Much higher capacity
  - Higher density
  - Lower cost

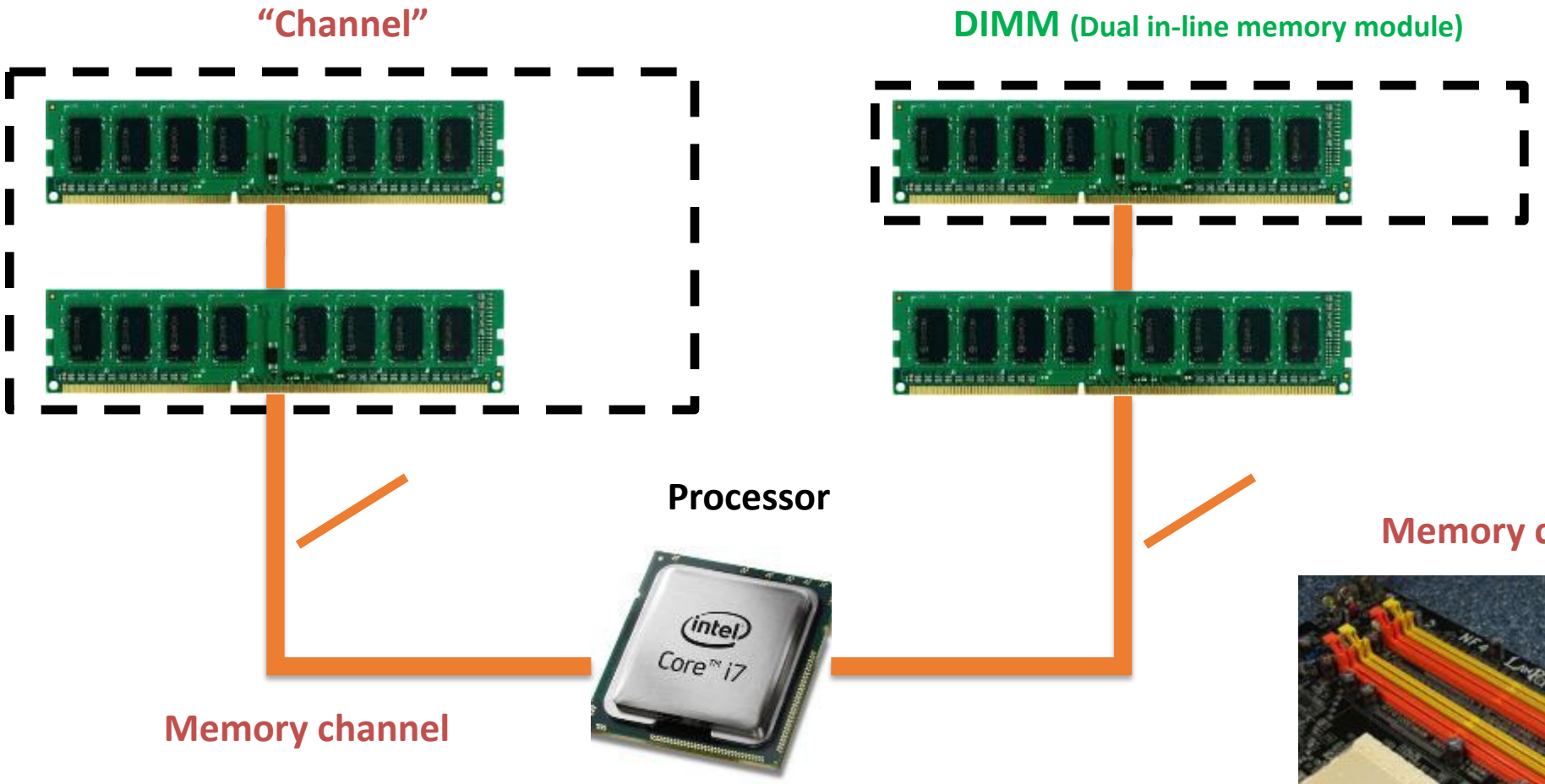
Memory subsystem organization

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



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

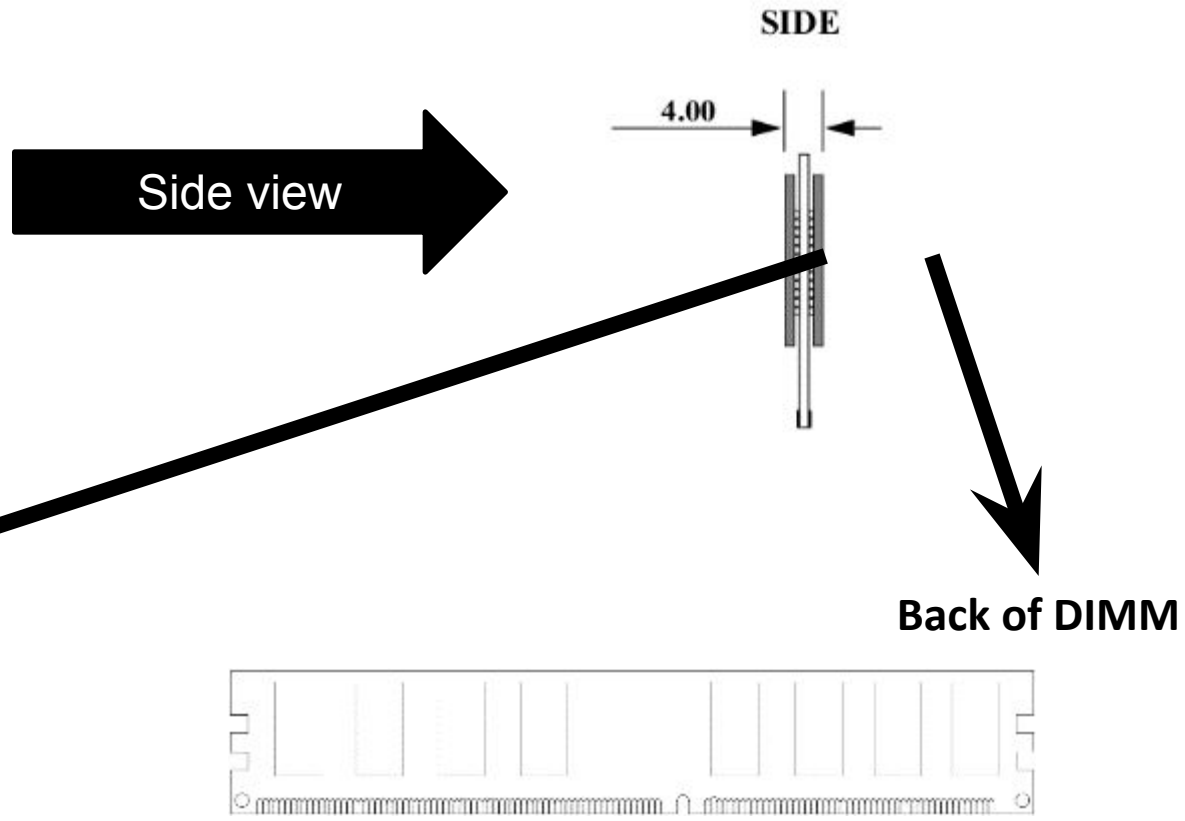
Memory subsystem



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

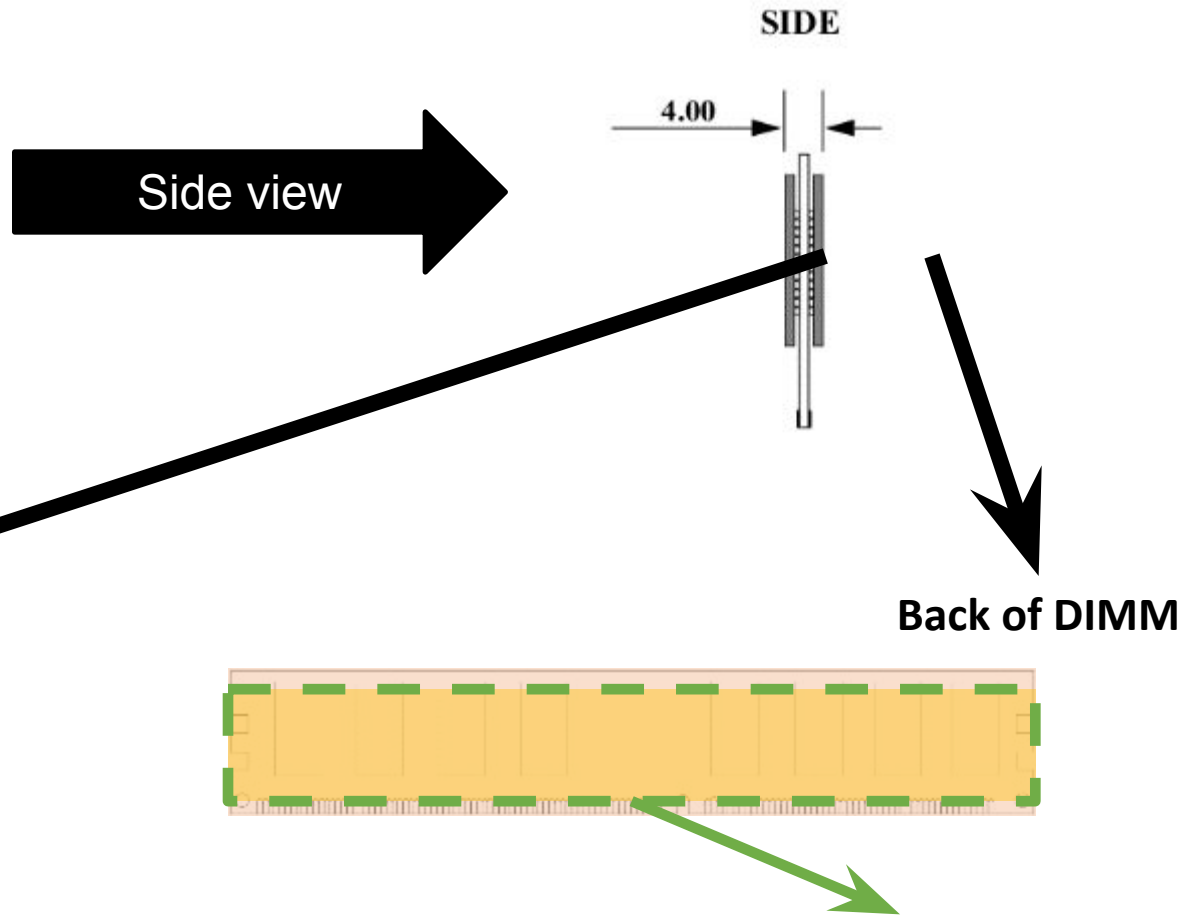
Breaking down a DIMM

DIMM (Dual in-line memory module)



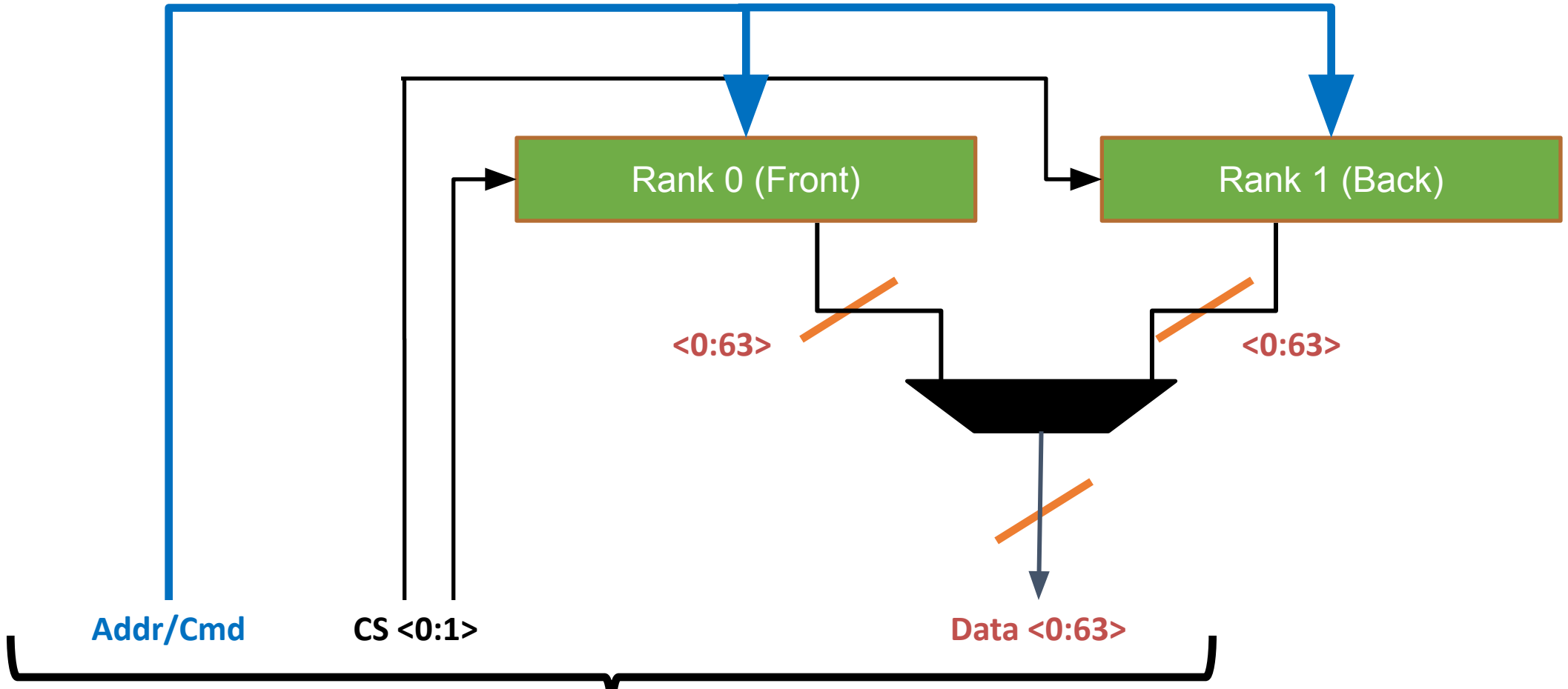
Breaking down a DIMM

DIMM (Dual in-line memory module)

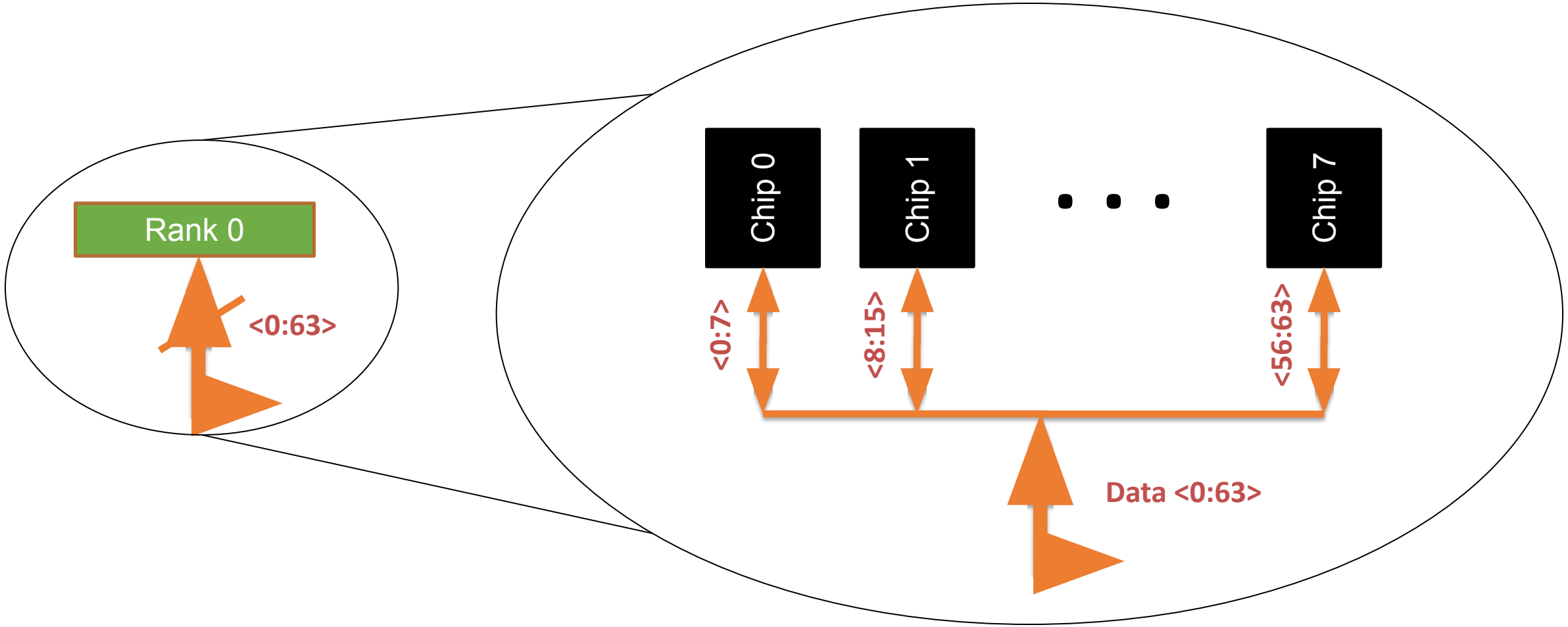


Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Rank

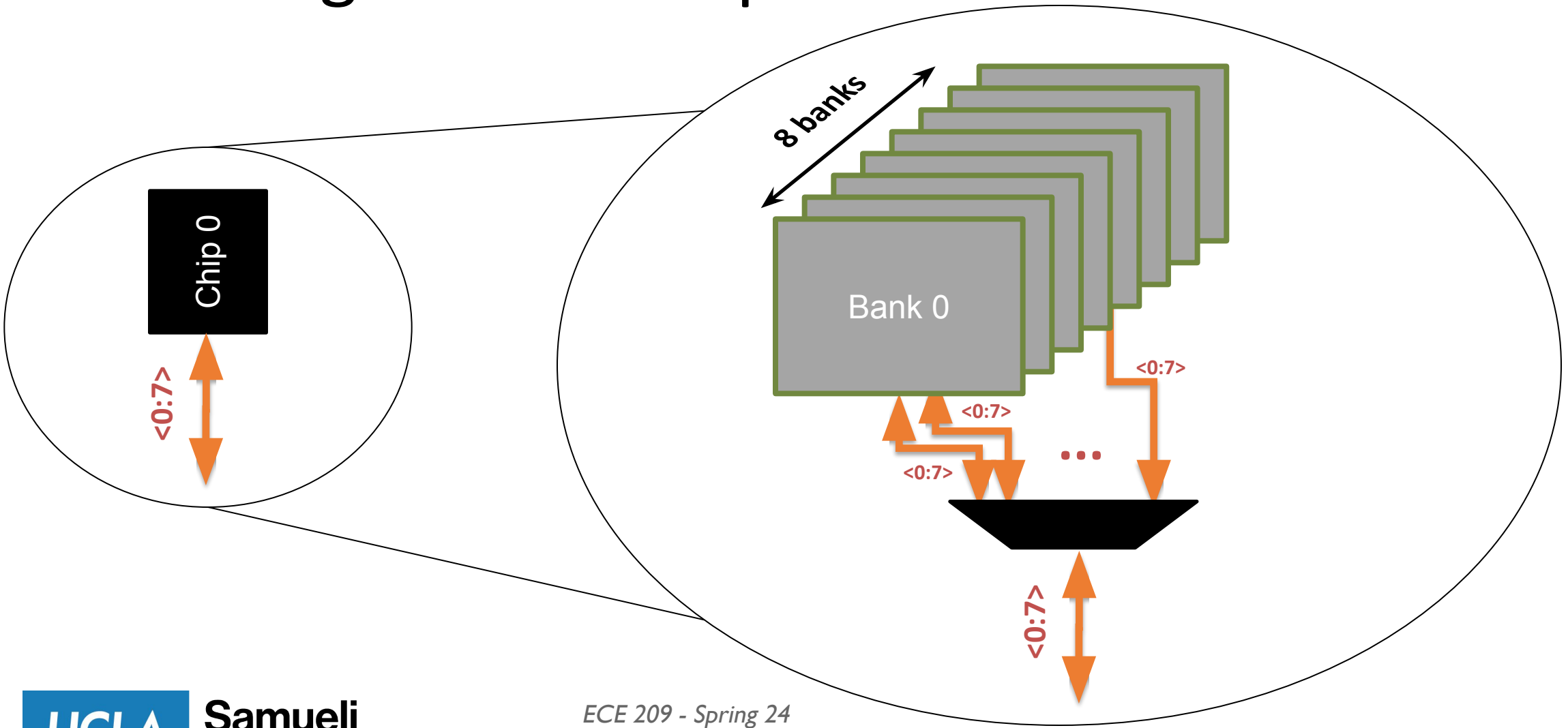

Rank 0 (Front)
Rank 1 (Back)
<0:63>
<0:63>
Addr/Cmd
CS <0:1>
Data <0:63>
Memory channel

Breaking down a Rank



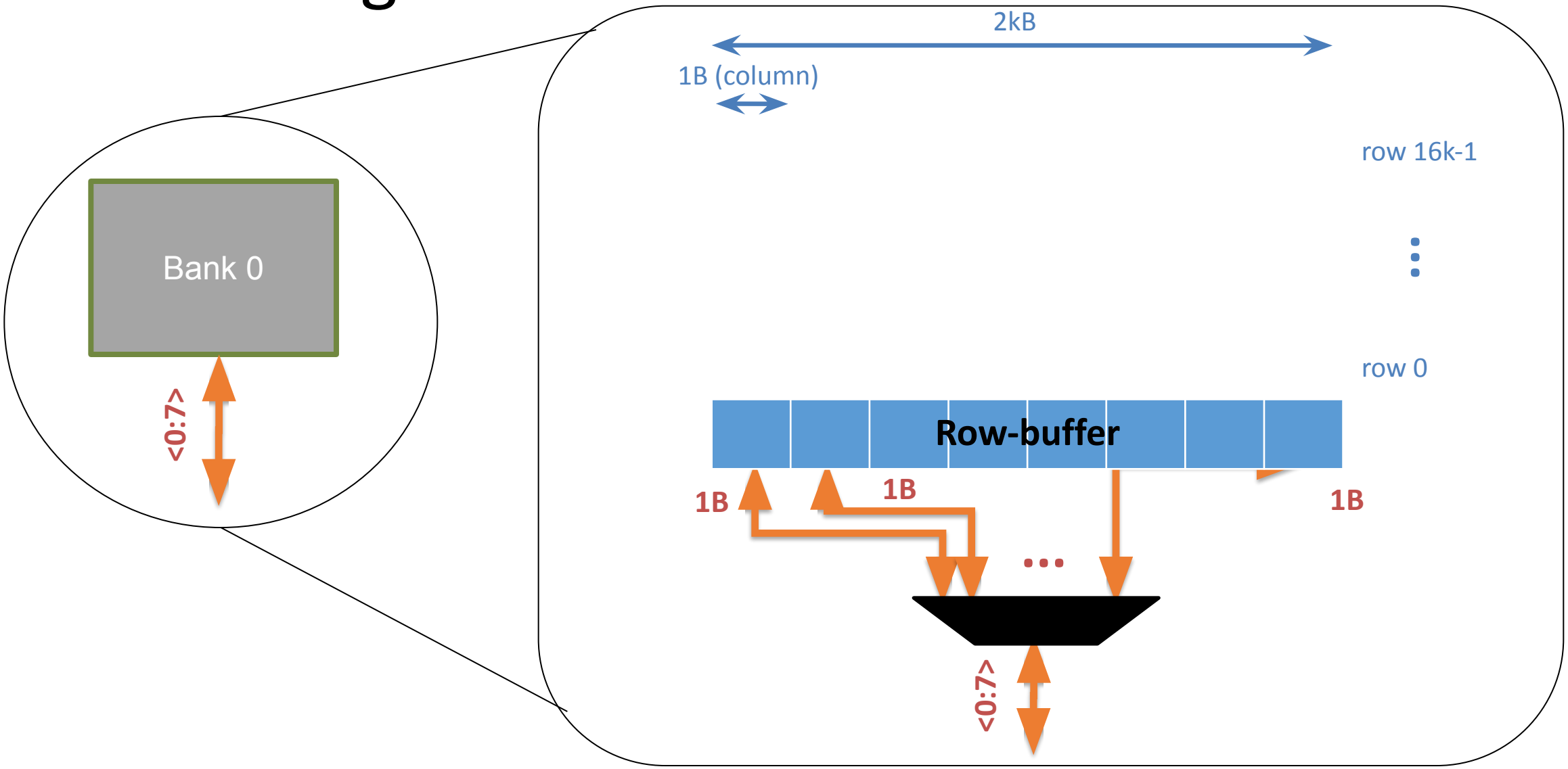
Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Breaking down a Chip

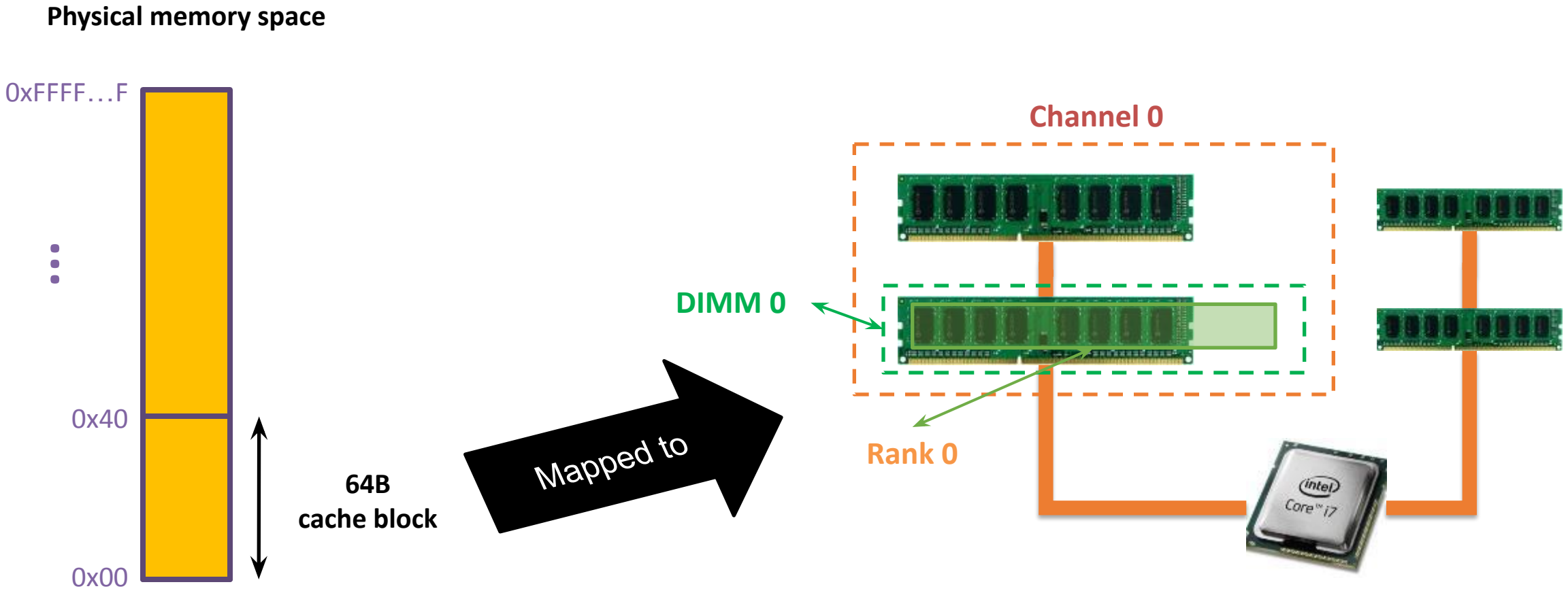


Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Breaking down a Bank

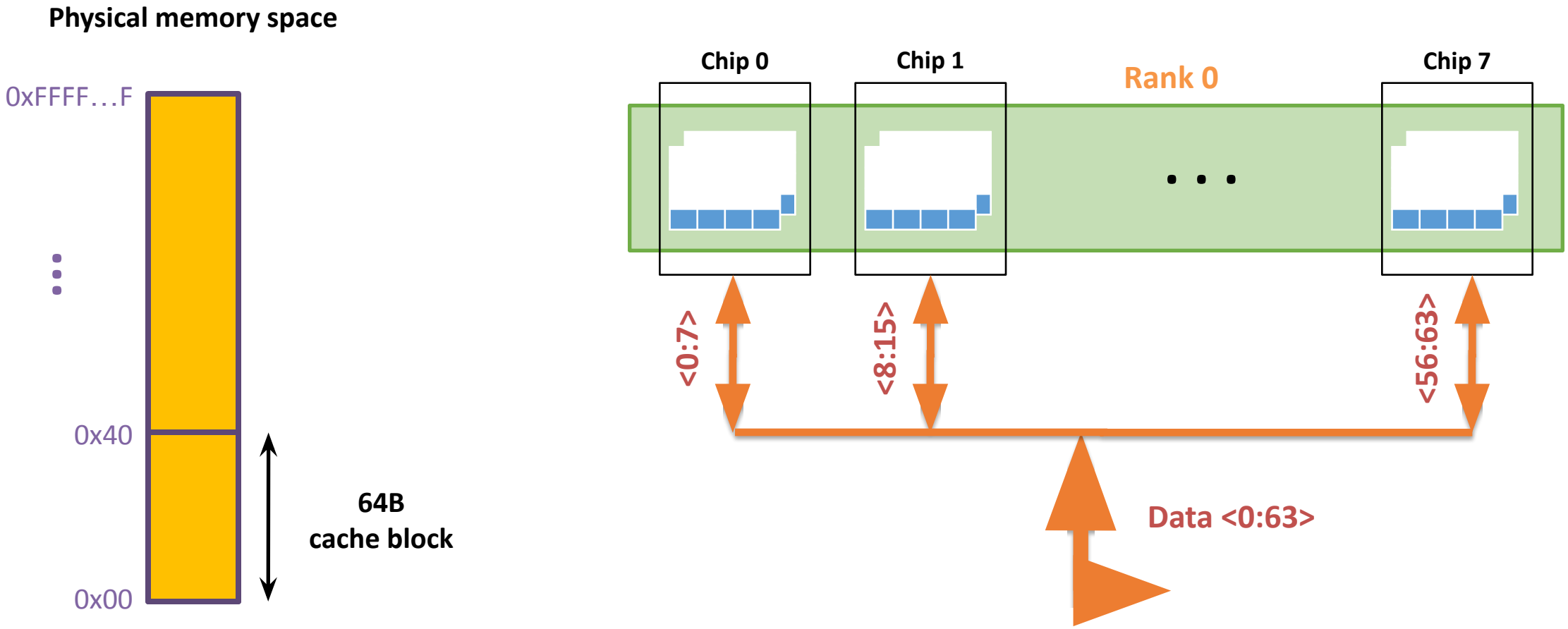


Example: Transferring a cache block

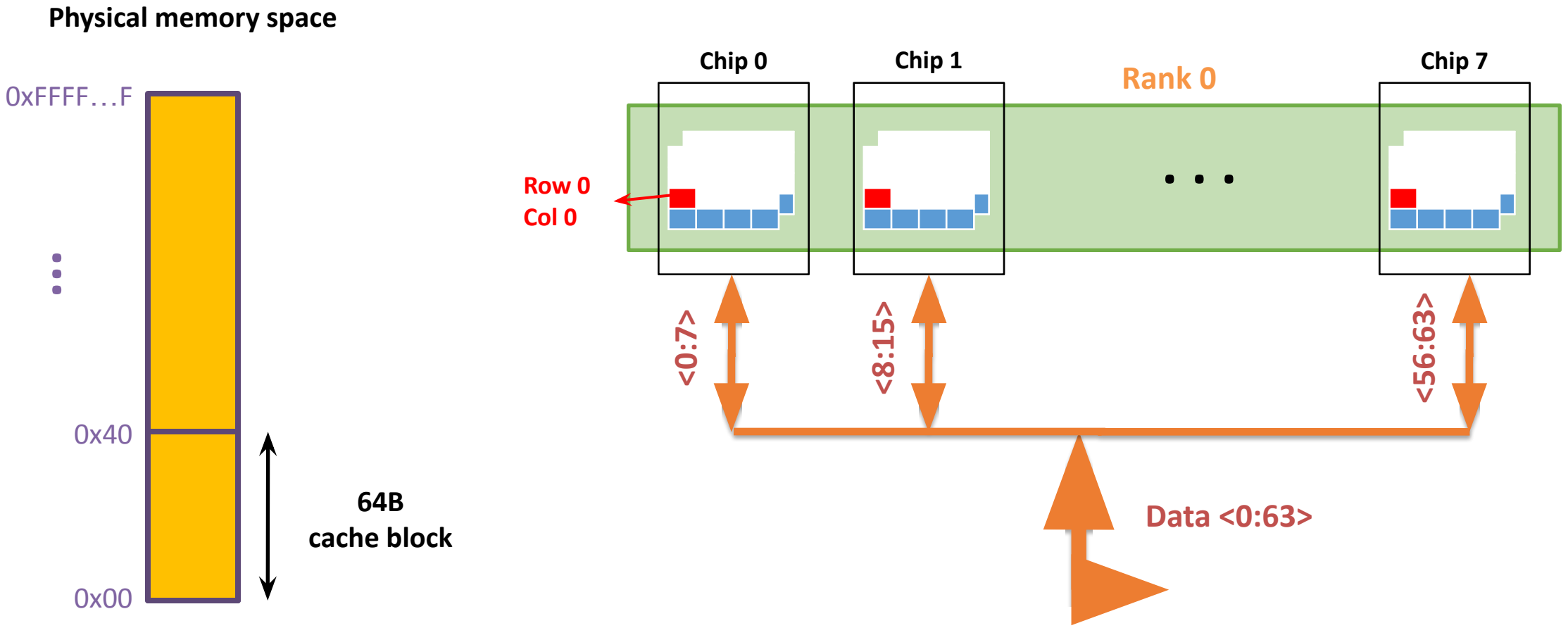


Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Example: Transferring a cache block

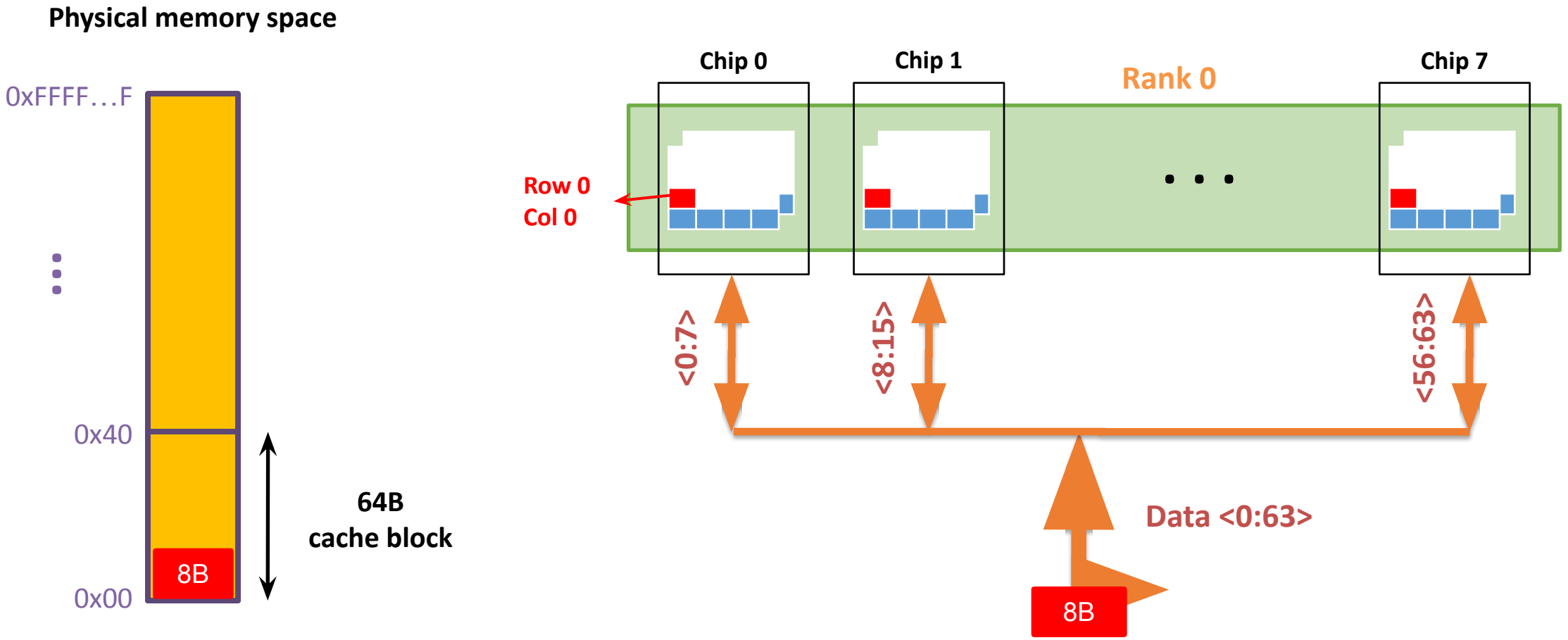


Example: Transferring a cache block

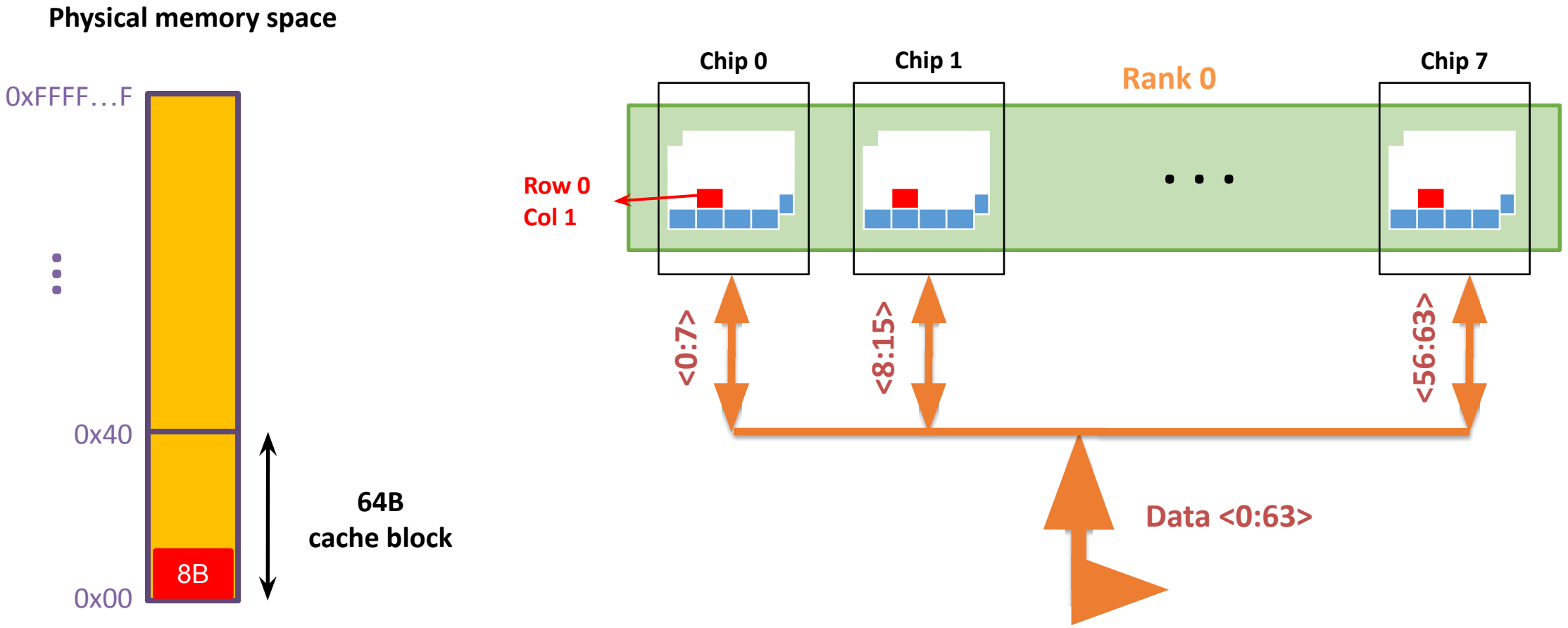


Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Example: Transferring a cache block

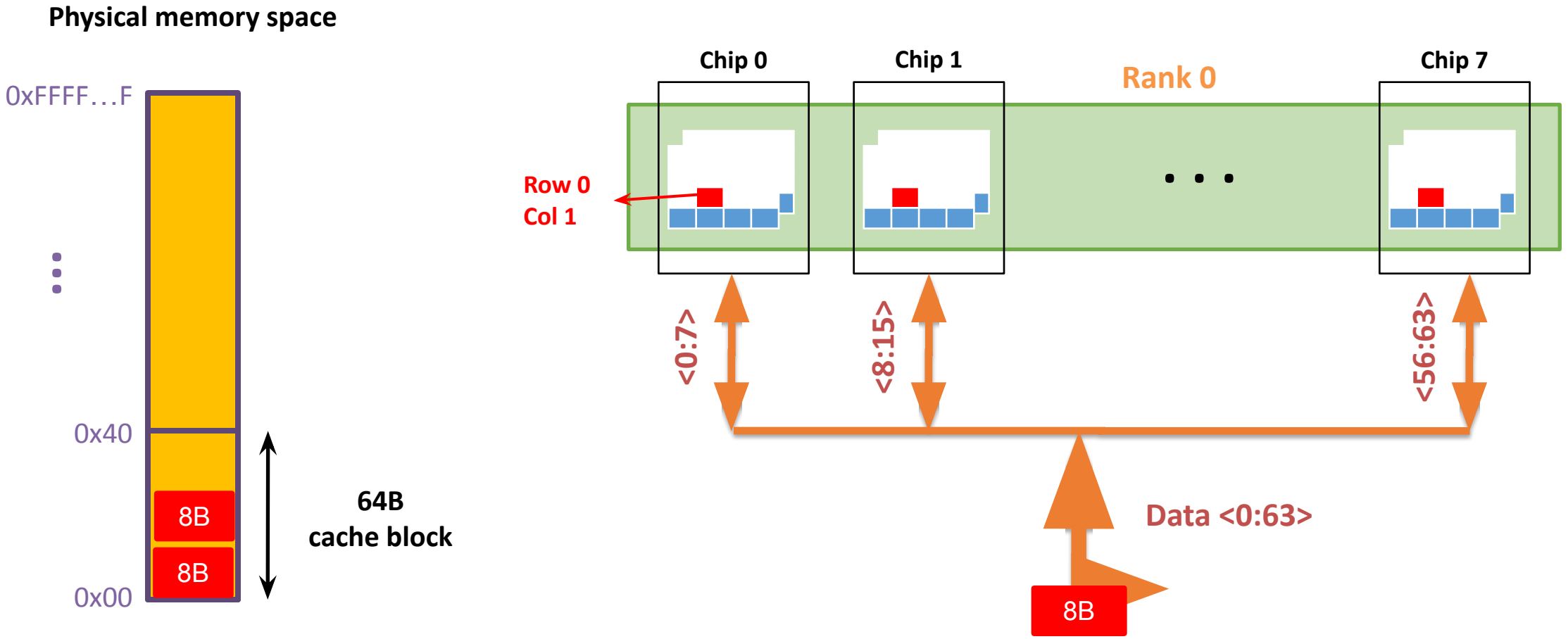


Example: Transferring a cache block



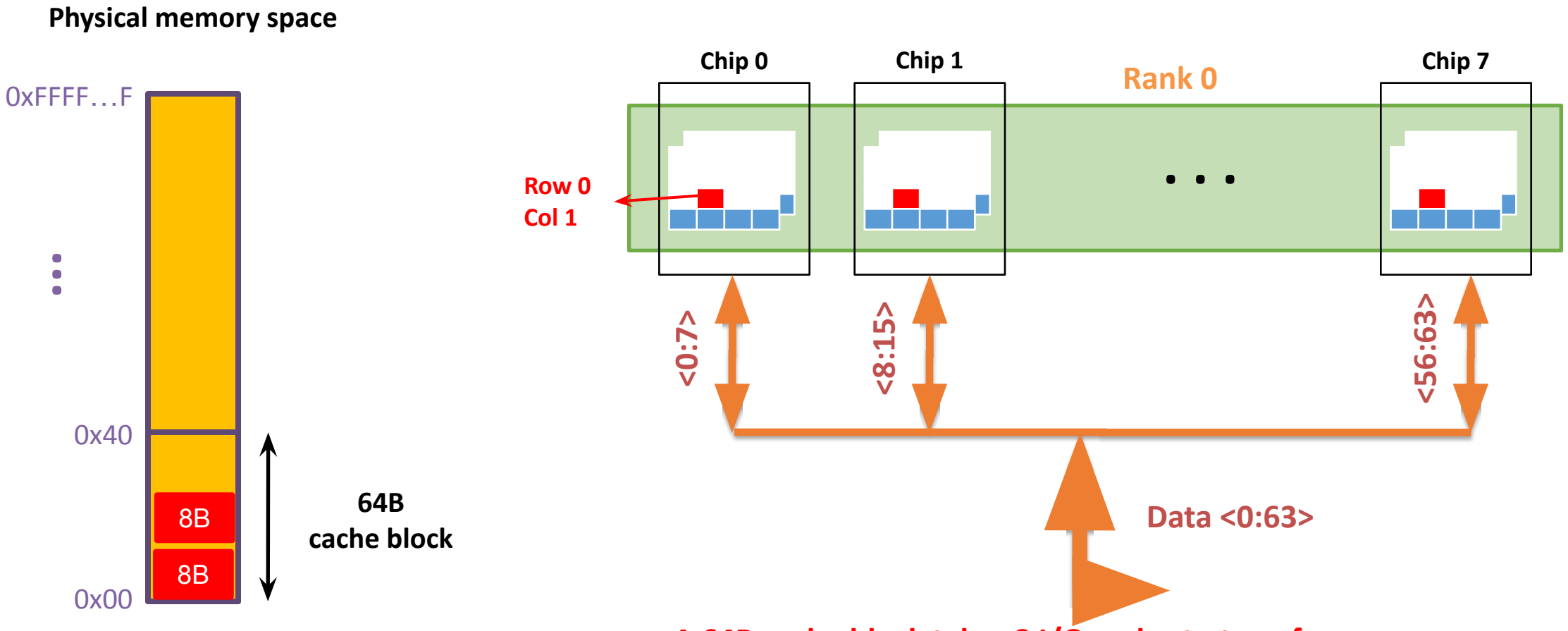
Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Example: Transferring a cache block



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Example: Transferring a cache block



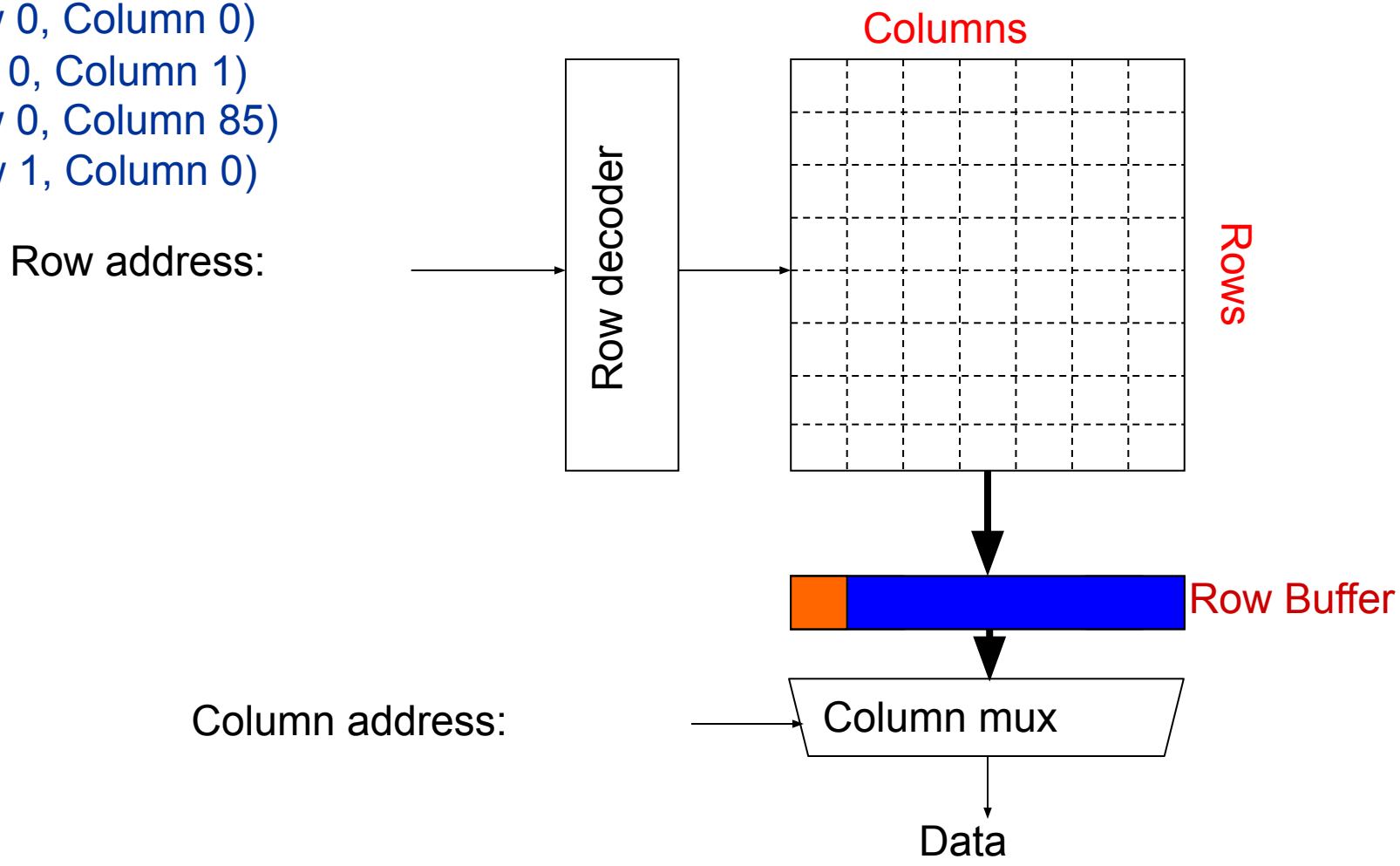
Page Mode DRAM

- 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)



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 is row is "open" OR
  - RAS + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)
- DRAM → CPU transfer time (through controller)

Open Page vs. Close Page Policies

- Open Page: Keep the row opened until a conflict
  - A row buffer hit will take only CAS delay: say 12.5 ns
  - A row buffer conflict will take PRE+RAS+CAS: say 37.5 ns

- Close Page: Close the row buffer after access
  - A row buffer miss take RAS + CAS: say 12.5 + 12.5 = 25ns

- Right policy depends on row buffer hit rate
  - Server processors typically use close page policy

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

How to schedule? When to close row?

DRAM Scheduling Policies

- FCFS (first come first served)
  - Oldest request first

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

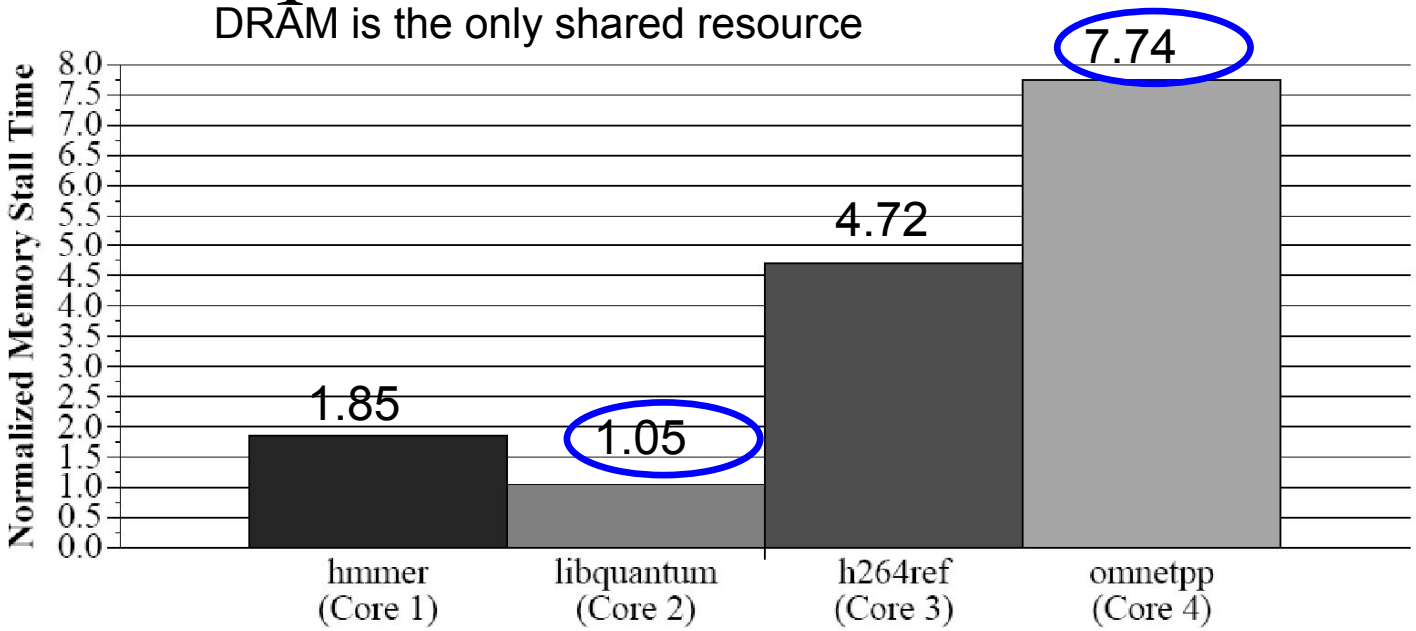
  1. Row-hit first
  2. Oldest first

  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

  FR-FCFS is unfair when multiple threads share the DRAM system

Consequences of Unfairness in DRAM



- Vulnerability to denial of service [Moscibroda & Mutlu, Usenix Security'07]
- System throughput loss
- Priority inversion at the system/OS level
- Poor performance predictability

Fairness in Shared DRAM Systems

- A thread's DRAM performance dependent on its inherent
  - Row-buffer locality
  - Bank parallelism
- Interference between threads can destroy either or both
- A fair DRAM scheduler should take into account all factors affecting each thread's DRAM performance
  - Not solely bandwidth or solely request latency

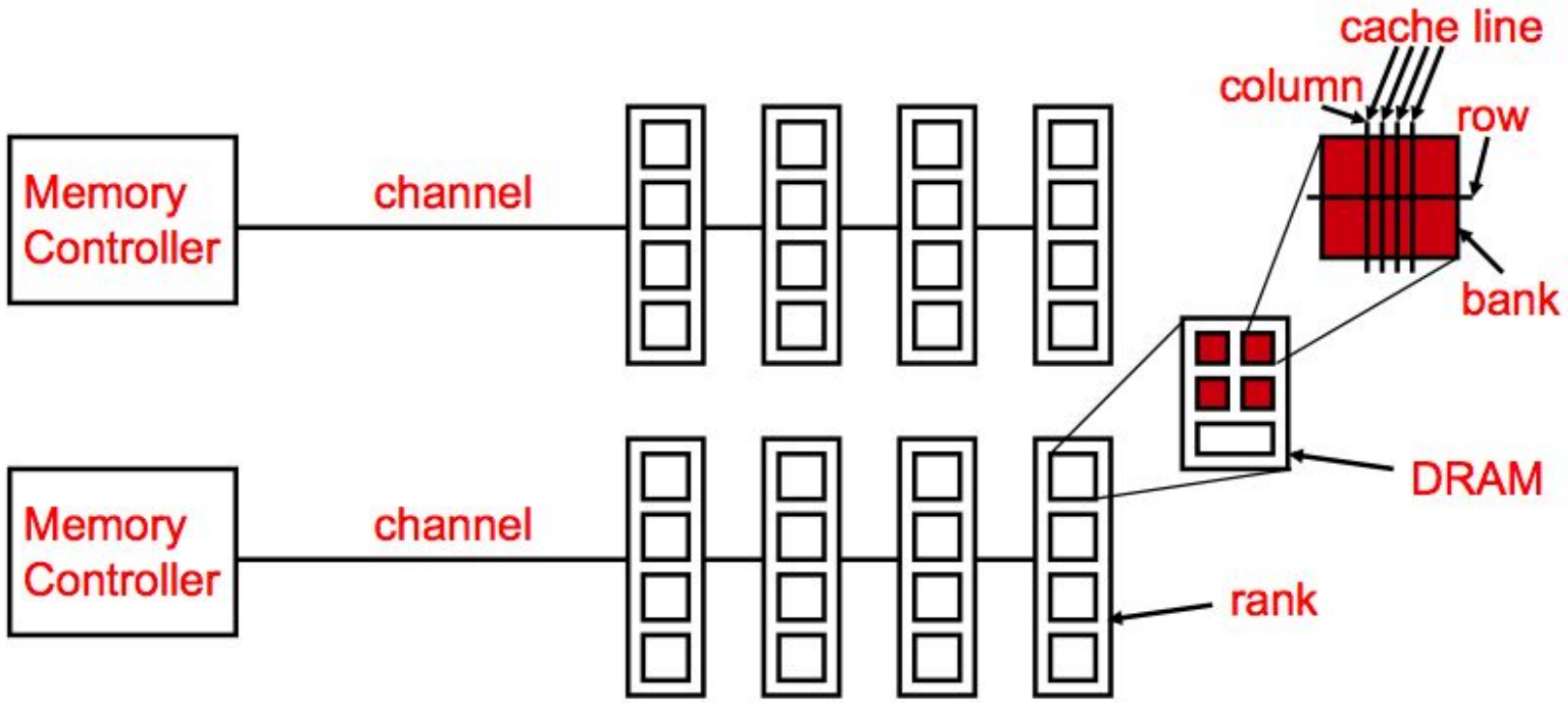
- Observation: A thread's performance degradation due to interference in DRAM mainly characterized by the extra memory-related stall-time due to contention with other threads

What other strategies one can use?

- Round robin
- Random
- Power saving
- …

➔ CA3 will be about designing a scheduler.

Generalized Memory Structure



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

A DRAM Chip and DIMM

- 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)

- Multiple chips are put together to form a wide interface
  - Called a module
  - DIMM: Dual Inline Memory Module
  - All chips in one side of a DIMM are operated the same way (rank)
    - Respond to a single command
    - Share address and command buses, but provide different data

- If we have chips with 8-bit interface, to read 8 bytes in a single access, use 8 chips in a DIMM

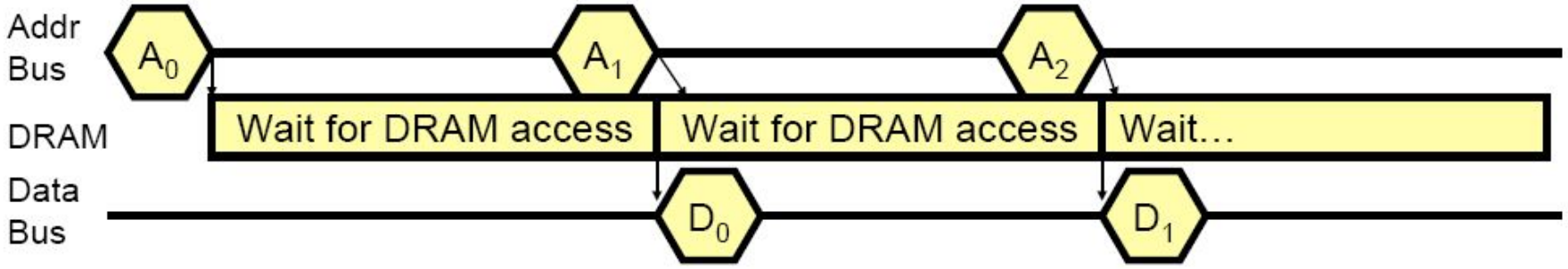
Multiple Banks (Interleaving) and Channels

- 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

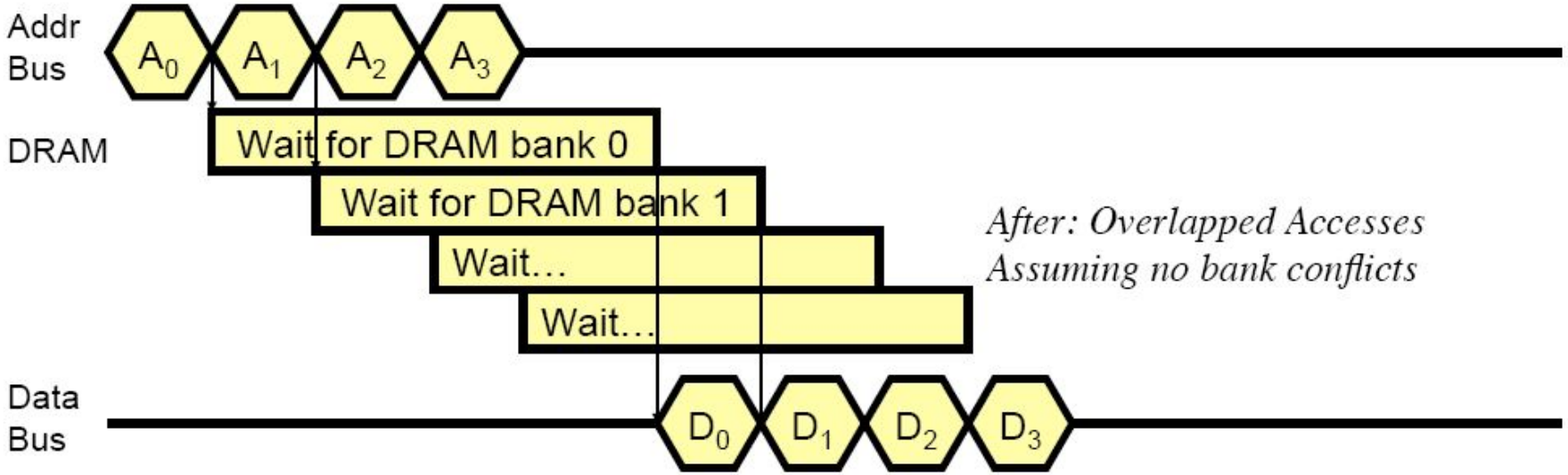
- Enabling more concurrency requires reducing
  - Bank conflicts
  - Channel conflicts
- How to select/randomize bank/channel indices in address?
  - Lower order bits have more entropy
  - Randomizing hash functions (XOR of different address bits)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

How Multiple Banks/Channels Help



Before: No Overlapping
Assuming accesses to different DRAM rows



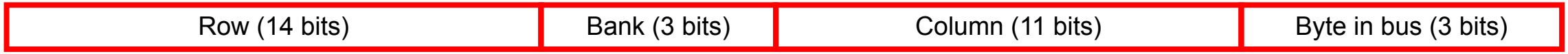
After: Overlapped Accesses
Assuming no bank conflicts

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)

- 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



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

  - Accesses to consecutive cache blocks can be serviced in parallel
  - How about random accesses? Strided accesses?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Bank Mapping Randomization

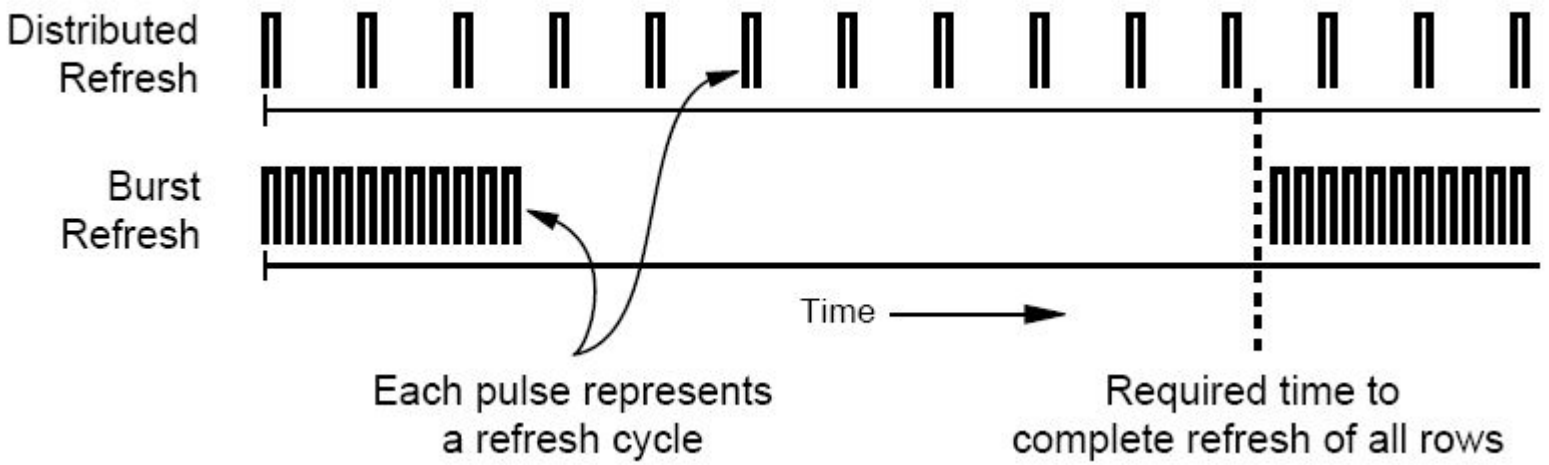
- DRAM controller can randomize the address mapping to banks so that bank conflicts are less likely

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

DRAM Refresh (I)

- DRAM capacitor charge leaks over time
- The memory controller needs to read each row periodically to restore the charge
  - Activate + precharge each row every N ms
  - Typical N = 64 ms
- Implications 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

DRAM Refresh (II)



- Distributed refresh eliminates long pause times
- How else we can reduce the effect of refresh on performance?
  - Can we reduce the number of refreshes?

What about security?

- What if someone can flip some data?

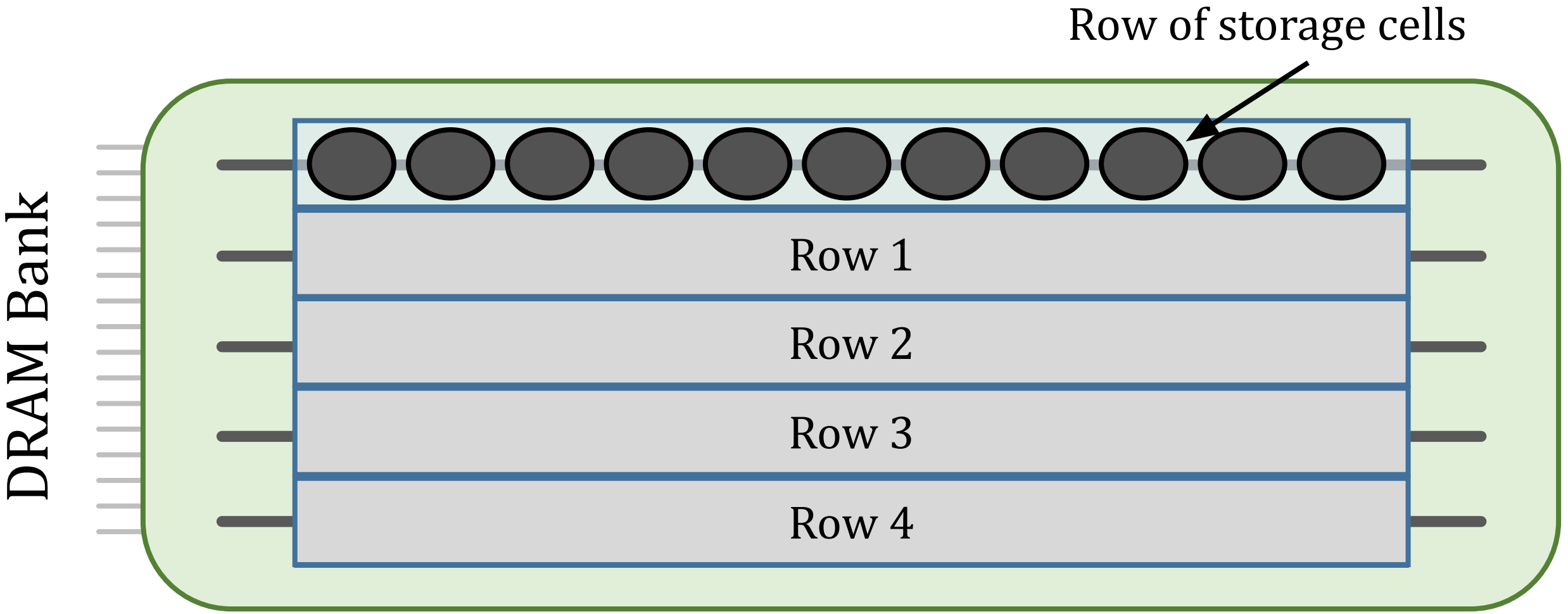
- How?
  - Physical attacks

What is RowHammer attack?

- Attack on modern DRAM systems!

- The key idea is that consecutive accesses to a few nearby rows could cause bit flips to adjacent rows! and hence create a fault!

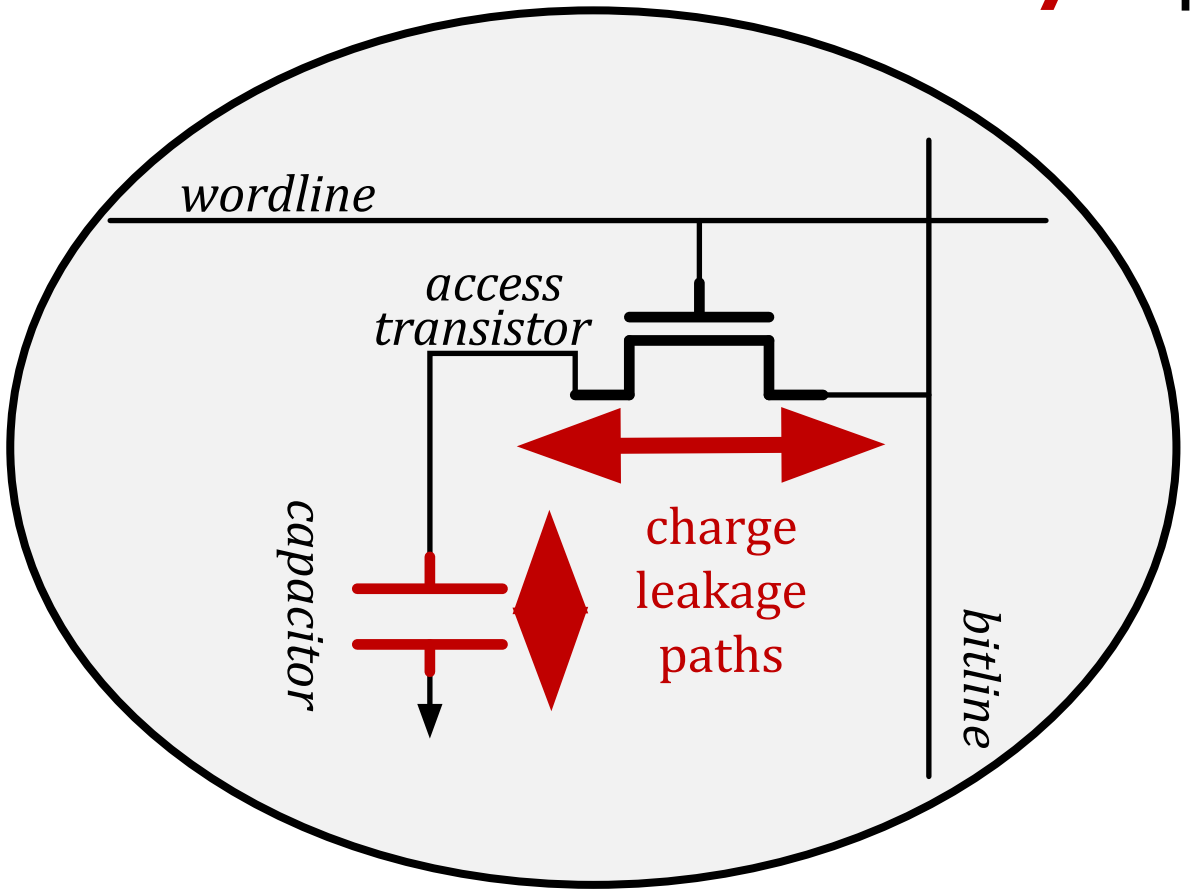
DRAM Organization



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

DRAM Cell Leakage

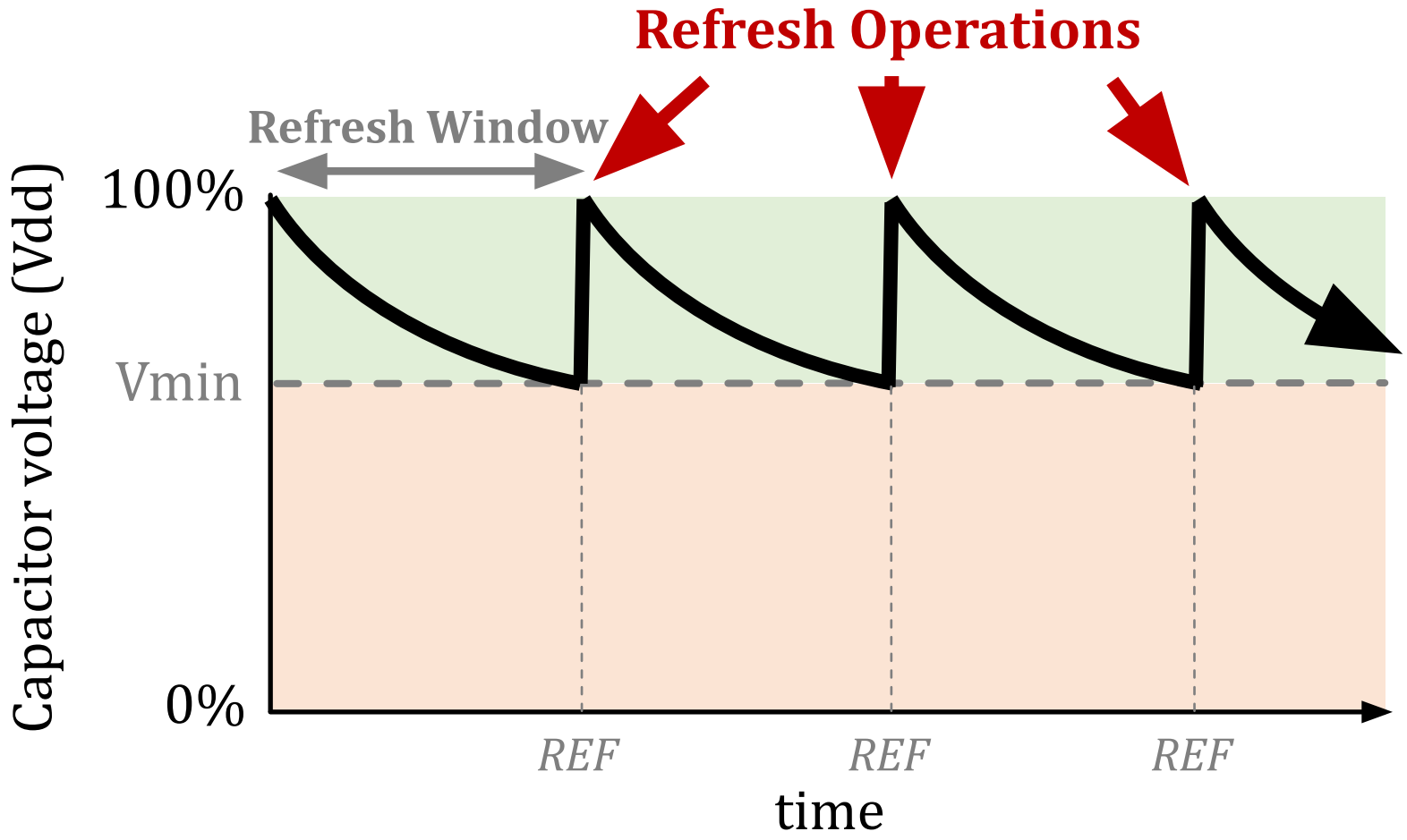
Each cell encodes information in **leaky** capacitors



Stored data is **corrupted** if too much charge leaks (i.e., the capacitor voltage degrades too much)

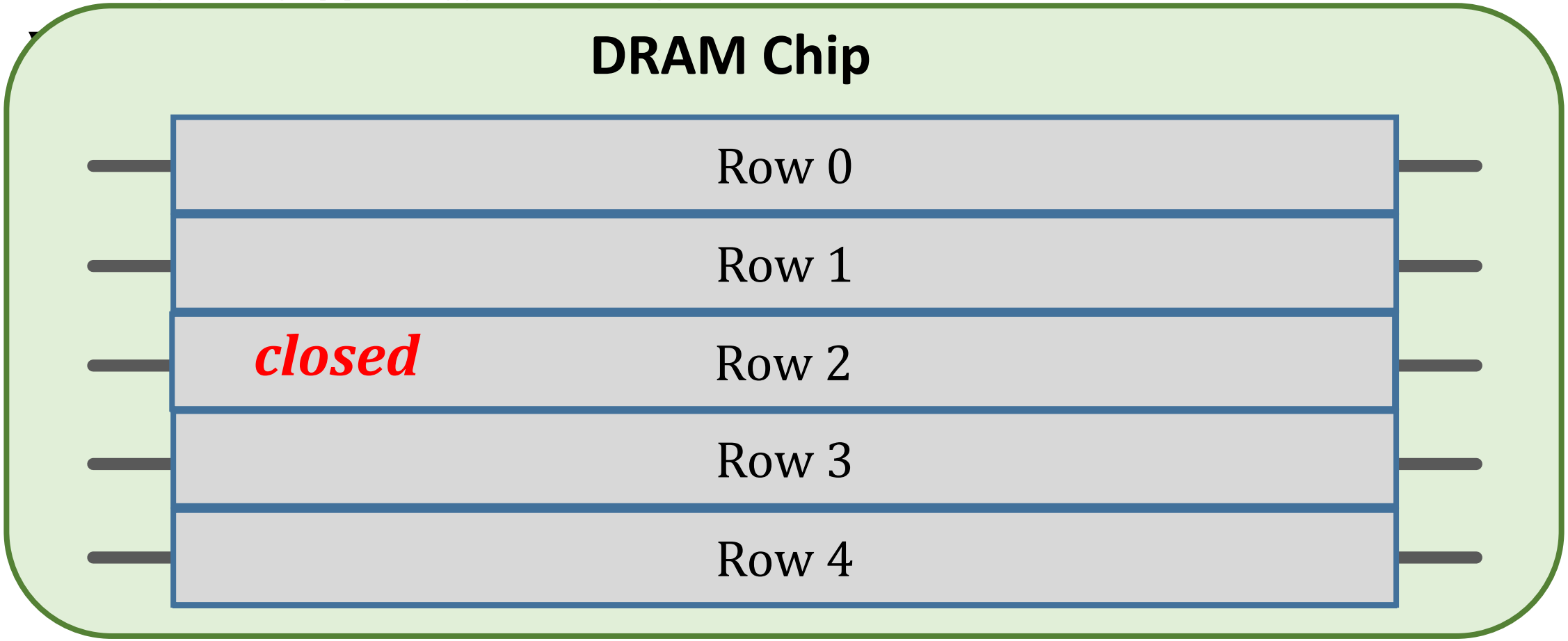
[Patel+, ISCA'17]

DRAM Refresh



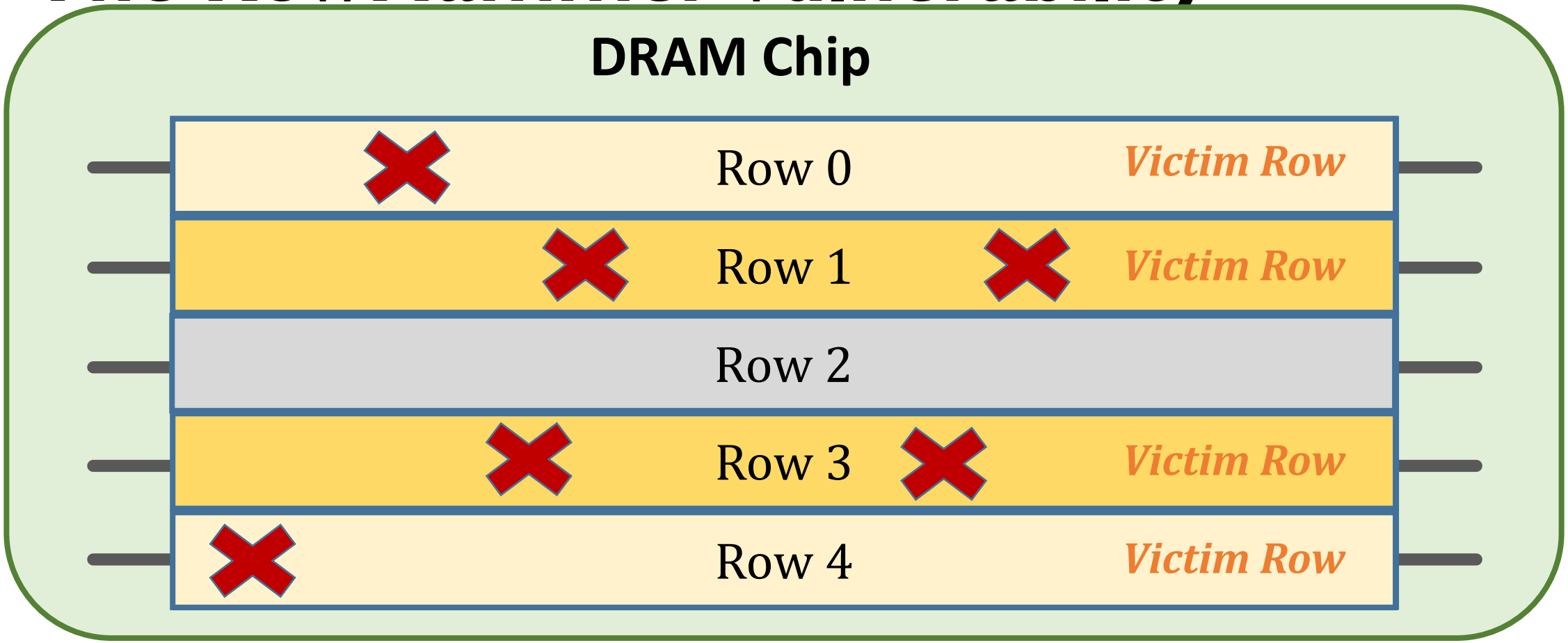
Periodic **refresh operations** preserve stored data

The RowHammer



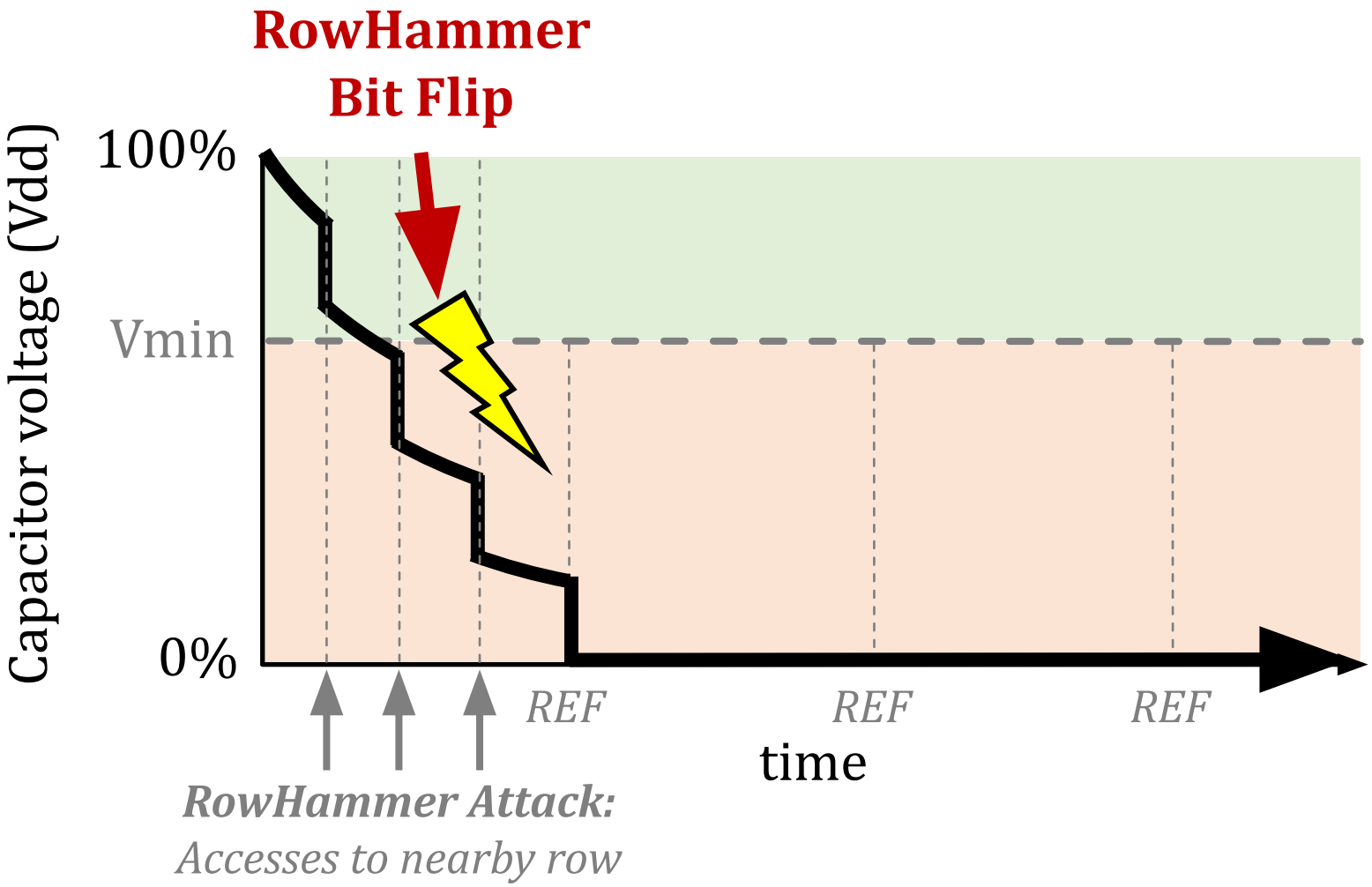
Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **RowHammer bit flips** in nearby cells

The RowHammer Vulnerability



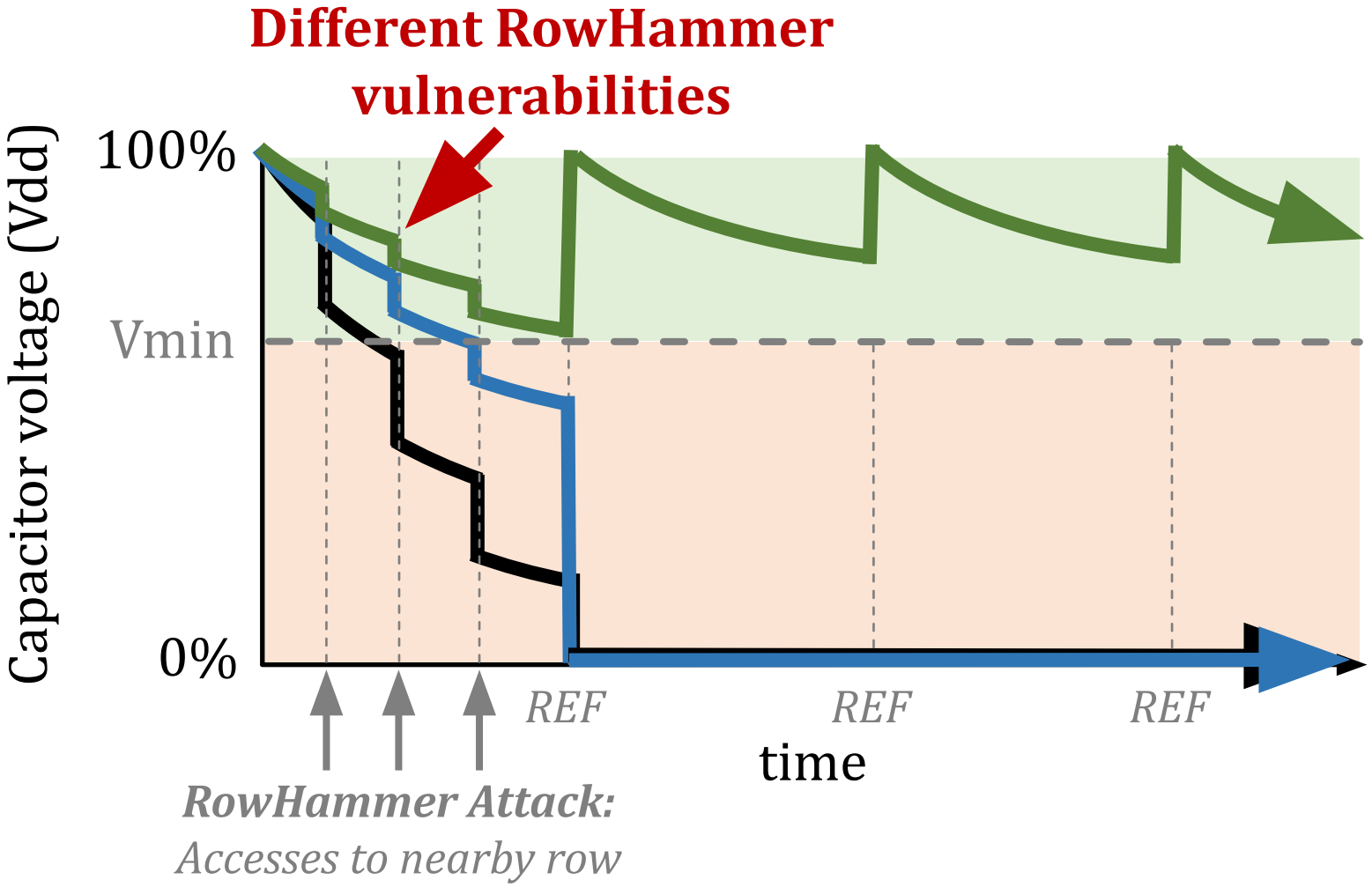
Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **RowHammer bit flips** in nearby cells

RowHammer Bit Flips



Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Cell-to-Cell Variation



Some cells are more vulnerable due to **process variation**

Challenges

- Finding the address mapping
- Skipping the row buffer hit
- Single sided vs double sided
- What to do with it?
  - Privilege escalation for a page
  - Data-dependent activities
  - …

Research in Rowhammer

- New attacks
  - Innovative methods to cause more damage
  - How to leverage bitflips to do something useful at software layer

- New defenses: how to protect
  - Software methods
  - Error correction
  - Detection and prevention methods
  - Changing hardware

Defense for RowHammer

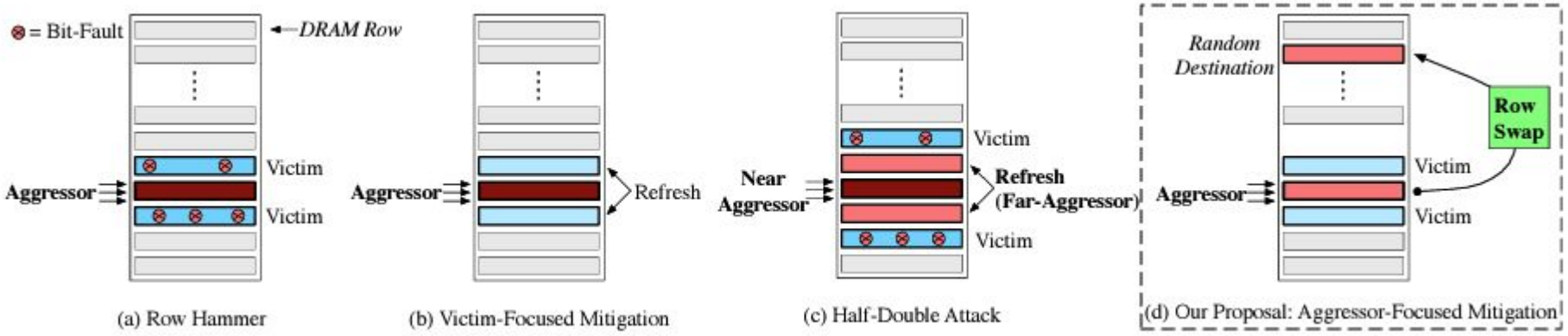


Image was taken from [2]

Beyond Rowhamer

- Rowhammer is a member of a bigger class of attacks called fault attacks. The idea is to leverage a physical phenomenon to inject faults (i.e., mostly flipping bits) in CPU and/or memory.

- Other examples:
  - Physical attacks through laser, magnetic, etc.
  - Changing voltage and frequency (generally operation) of the devices.
  - …

End of Presentation

UCLA Samueli School of Engineering

Acknowledgement

- This course is partly inspired by the following courses created by my colleagues:
  - 6.888 MIT (Yan)
  - CS 598 UIUC (Fletcher)
  - CS 7290 Georgia Tech (Kim)
  - ECE 752 Wisconsin (Lipasti)
  - ECE 7103A Georgia Tech (Qureshi)