# ADVANCED MEMORY SYSTEMS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

THE UNIVERSITY OF UTAH

## Overview

- Announcement
  - Homework 5 will be released tonight (the last one ☺)
- This lecture
  - Memory addressing/scheduling
  - DRAM refresh
  - Emerging technologies

# Recall: DRAM Control Tasks

- Refresh management
  - Periodically replenish the DRAM cells (burst vs. distributed)
- Address mapping
  - Distribute the requests to destination banks (load balancing)
- Request scheduling
  - Generate a sequence of commands for memory requests
    - Reduce overheads by eliminating unnecessary commands
- Power management
  - Keep the power consumption under a cap
- Error detection/correction
  - Detect and recover corrupted data

## Address Mapping

- A memory request

|  |  |  |
|---|---|---|
| Type | Address | Data |

- Address is used to find the location in memory
  - Channel, rank, bank, row, and column IDs
- Example physical address format

|  |  |  |  |  |
|---|---|---|---|---|
| Row ID | Channel ID | Rank ID | Bank ID | Column ID |

- A 4GB channel, 2 ranks, 4 banks/rank, 8KB page

# Address Mapping

- A memory request

| Type | Address | Data |
|---|---|---|

- Address is used to find the location in memory
  - Channel, rank, bank, row, and column IDs
- Example physical address format

| Row ID | Channel ID | Rank ID | Bank ID | Column ID |
|---|---|---|---|---|
| 16 | 0 | 1 | 2 | 13 |

- A 4GB channel, 2 ranks, 4 banks/rank, 8KB page

## Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)

*addr*

00000010

20000001

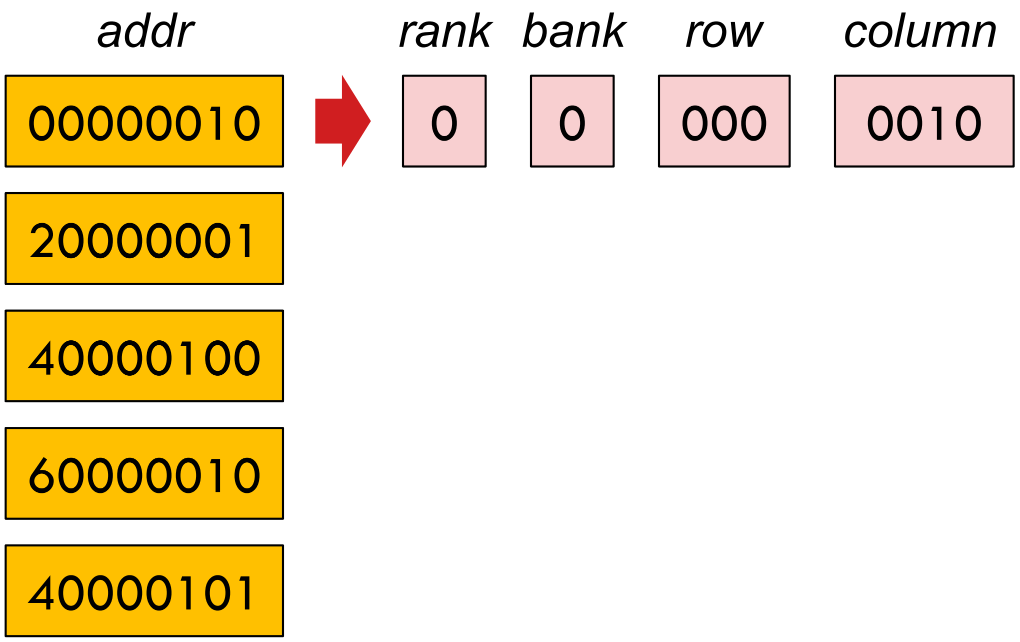
40000100

60000010

40000101

## Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)



## Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)

| addr | | rank | bank | row | column |
|---|---|---|---|---|---|
| 00000010 | ➡ | 0 | 0 | 000 | 0010 |
| 20000001 | ➡ | 0 | 0 | 200 | 0001 |
| 40000100 | ➡ | 0 | 0 | 400 | 0100 |
| 60000010 | ➡ | 0 | 0 | 600 | 0010 |
| 40000101 | ➡ | 0 | 0 | 400 | 0101 |

# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)

| *addr* | *rank* | *bank* | *row* | *column* | *commands* |
|---|---|---|---|---|---|
| 00000010 | 0 | 0 | 000 | 0010 | |
| 20000001 | 0 | 0 | 200 | 0001 | |
| 40000100 | 0 | 0 | 400 | 0100 | |
| 60000010 | 0 | 0 | 600 | 0010 | |
| 40000101 | 0 | 0 | 400 | 0101 | |

# Example Problem

- Start with empty row buffers, find the total number of commands if all the request are served in order
  - Address= row(12):channel(0):rank(1):bank(3):column(16)

| *addr* | *rank* | *bank* | *row* | *column* | *commands* | | |
|---|---|---|---|---|---|---|---|
| 00000010 | 0 | 0 | 000 | 0010 | | ACT | RD |
| 20000001 | 0 | 0 | 200 | 0001 | PRE | ACT | RD |
| 40000100 | 0 | 0 | 400 | 0100 | PRE | ACT | RD |
| 60000010 | 0 | 0 | 600 | 0010 | PRE | ACT | RD |
| 40000101 | 0 | 0 | 400 | 0101 | PRE | ACT | RD |

## Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)

*addr*

| |
|---|
| 00000010 |
| 20000001 |
| 40000100 |
| 60000010 |
| 40000101 |

## Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)

| *addr* | *rank* | *bank* | *row* | *column* |
|---|---|---|---|---|
| 00000010 | 0 | 0 | 000 | 0010 |
| 20000001 | 0 | 1 | 000 | 0001 |
| 40000100 | 0 | 2 | 000 | 0100 |
| 60000010 | 0 | 3 | 000 | 0010 |
| 40000101 | 0 | 2 | 000 | 0101 |

## Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)

| *addr* | *rank* | *bank* | *row* | *column* | *commands* |
|---|---|---|---|---|---|
| 00000010 | 0 | 0 | 000 | 0010 | |
| 20000001 | 0 | 1 | 000 | 0001 | |
| 40000100 | 0 | 2 | 000 | 0100 | |
| 60000010 | 0 | 3 | 000 | 0010 | |
| 40000101 | 0 | 2 | 000 | 0101 | |

# Example Problem

- Find the total number of commands using the following address mapping scheme
  - Address= bank(3):rank(1):channel(0):row(12):column(16)

| *addr* | *rank* | *bank* | *row* | *column* | *commands* | |
|---|---|---|---|---|---|---|
| 00000010 | 0 | 0 | 000 | 0010 | ACT | RD |
| 20000001 | 0 | 1 | 000 | 0001 | ACT | RD |
| 40000100 | 0 | 2 | 000 | 0100 | ACT | RD |
| 60000010 | 0 | 3 | 000 | 0010 | ACT | RD |
| 40000101 | 0 | 2 | 000 | 0101 | | RD |

## Command Scheduling

- Write buffering
  - Writes can wait until reads are done
- Controller queues DRAM commands
  - Usually into per-bank queues
  - Allows easily reordering ops. meant for same bank
- Common policies
  - First-Come-First-Served (FCFS)
  - First-Ready First-Come-First-Served (FR-FCFS)

# Command Scheduling

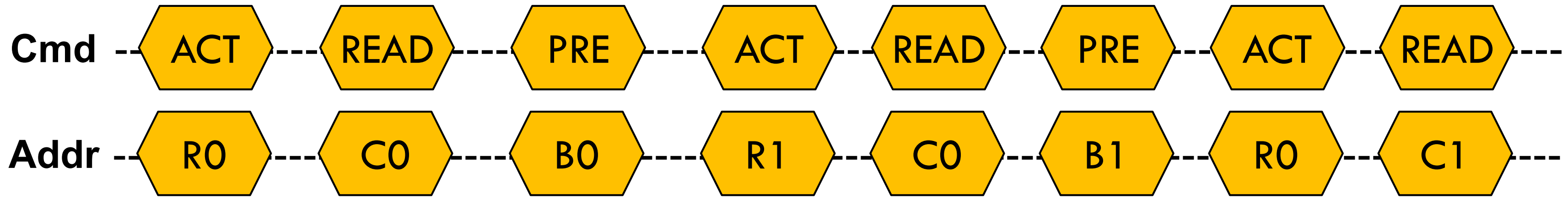
- First-Come-First-Served
  - Oldest request first
- First-Ready First-Come-First-Served
  - Prioritize column changes over row changes
  - Skip over older conflicting requests
  - Find row hits (on queued requests)
    - Find oldest
    - If no conflicts with in-progress request → good
    - Otherwise (if conflicts), try next oldest

## FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS

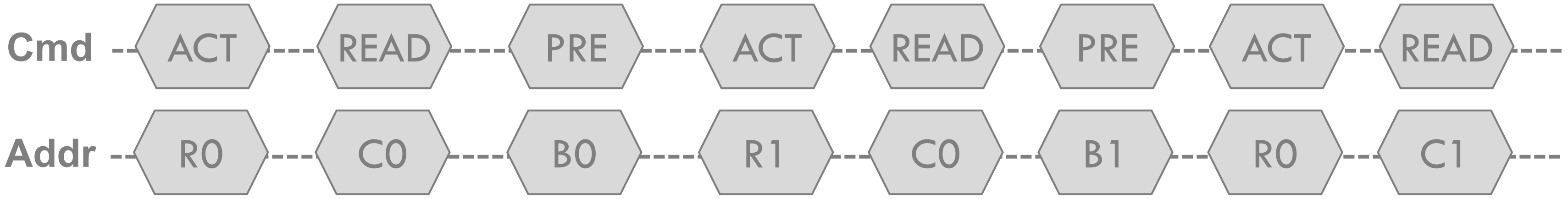
# FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS



# FCFS vs. FR-FCFS

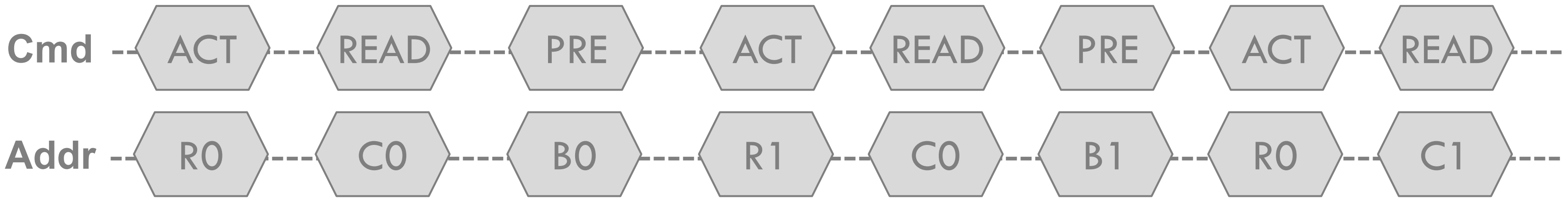
- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS



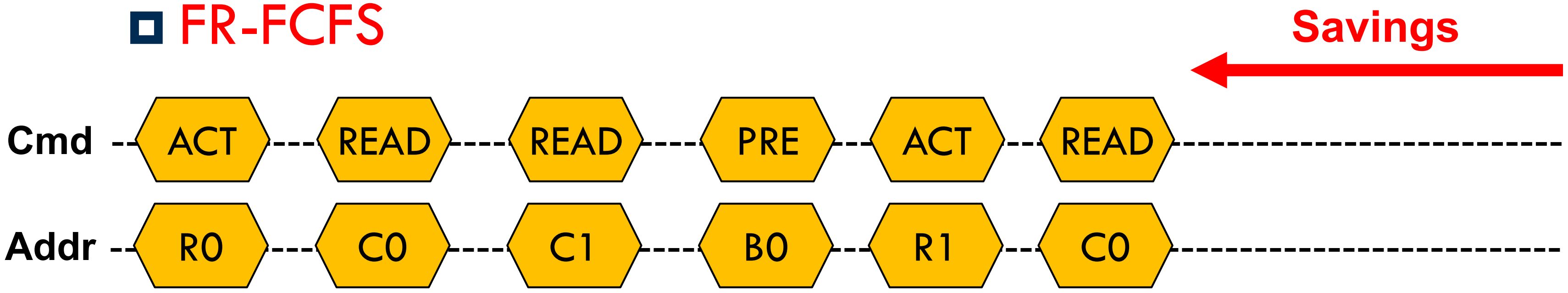
  - FR-FCFS

# FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS



  - FR-FCFS



## Row Buffer Management Policies

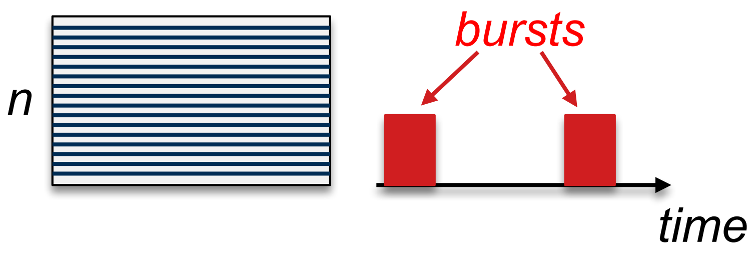
- Open-page policy
  - After access, keep page in DRAM row buffer
  - If access to different page, must close old one first
    - Good if lots of locality
- Close-page policy
  - After access, immediately close page in DRAM row buffer
  - If access to different page, old one already closed
    - Good if no locality (random access)

## DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically

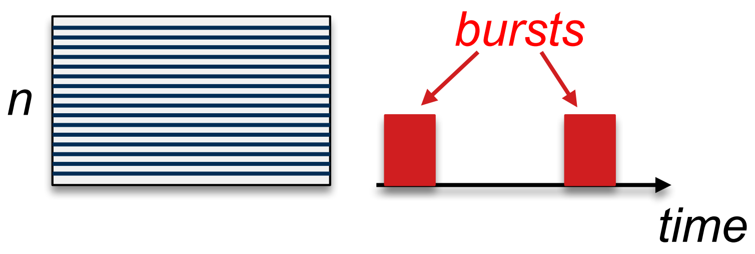
# DRAM Refresh Management

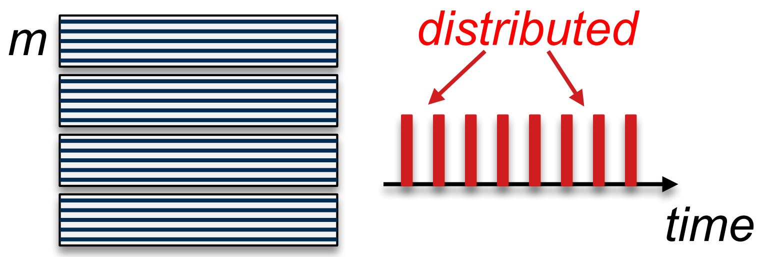
- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism



# DRAM Refresh Management

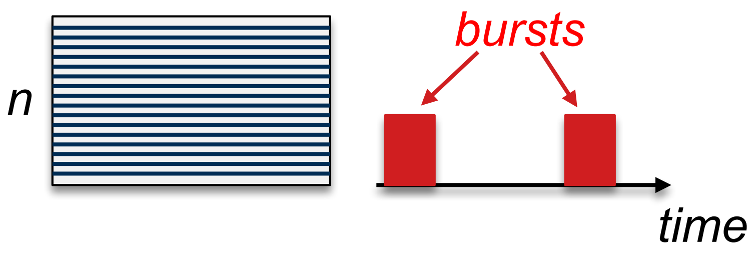
- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism
  - Distributed refresh: a group of cells are refreshed
    - Avoid blocking memory for a long time

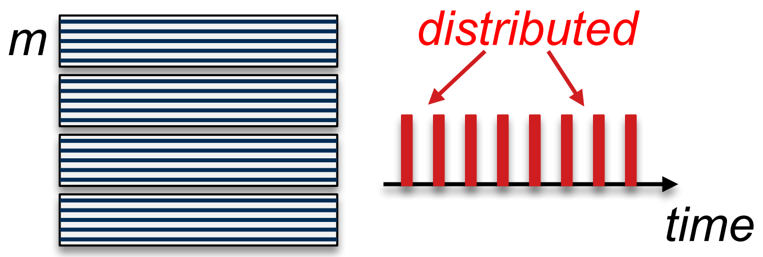




# DRAM Refresh Management

- DRAM requires the cells' contents to be read and written periodically
  - Burst refresh: refresh all of the cells each time
    - Simple control mechanism
  - Distributed refresh: a group of cells are refreshed
    - Avoid blocking memory for a long time
- Recently accessed rows need not to be refreshed
  - Smart refresh



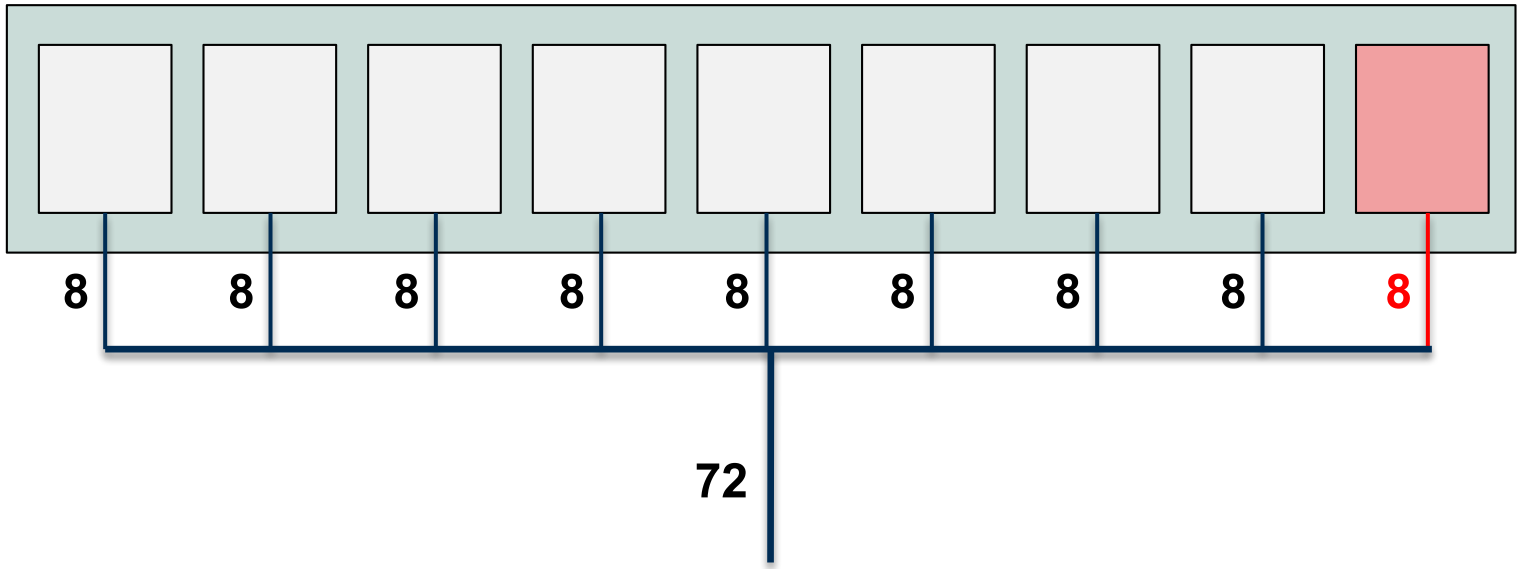


# Error Detection/Correction

- Data in memory may be corrupted
  - Many reasons: leakage, alpha particles, hard errors
- Can errors be detected?
  - Error detection codes: additional parity bits
- Can errors be corrected?
  - Error correction codes: ECC bits are added to data
- Single-Error Correction, Double-Error Detection
  - Commonly used in memory systems

## ECC DIMM

- An additional DRAM chip is used for storing SECDED ECC bits for error correction

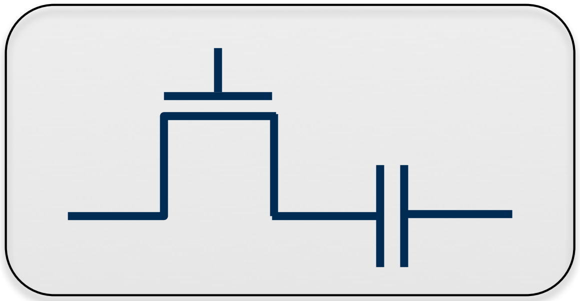


Hamming Code (72,64)

## Emerging Technologies

## DRAM Cell Structure

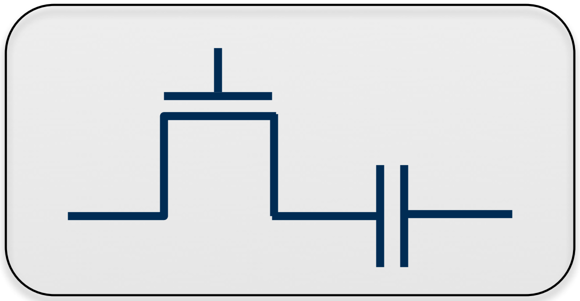
- One-transistor, one-capacitor
  - Realizing the capacitor is challenging



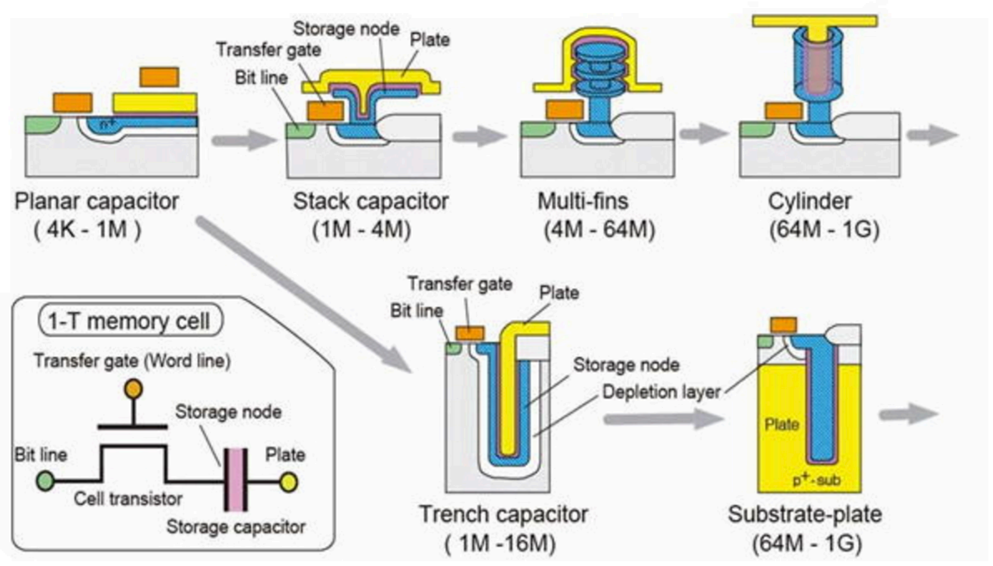
- 1T-1C DRAM
- Charge based sensing
- Volatile

# DRAM Cell Structure

- One-transistor, one-capacitor
  - Realizing the capacitor is challenging



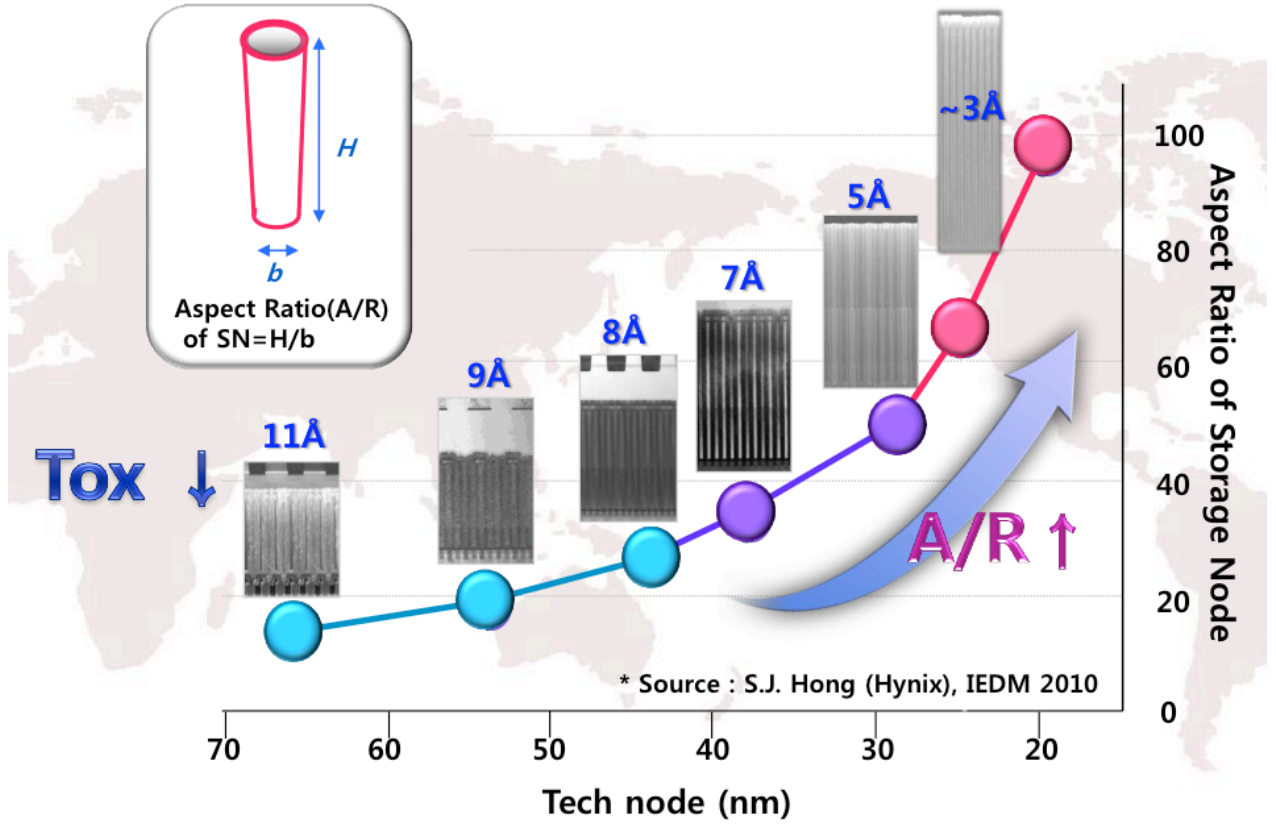
- 1T-1C DRAM
- Charge based sensing
- Volatile



# Memory Scaling in Jeopardy

Scaling of semiconductor memories greatly challenged beyond 20nm

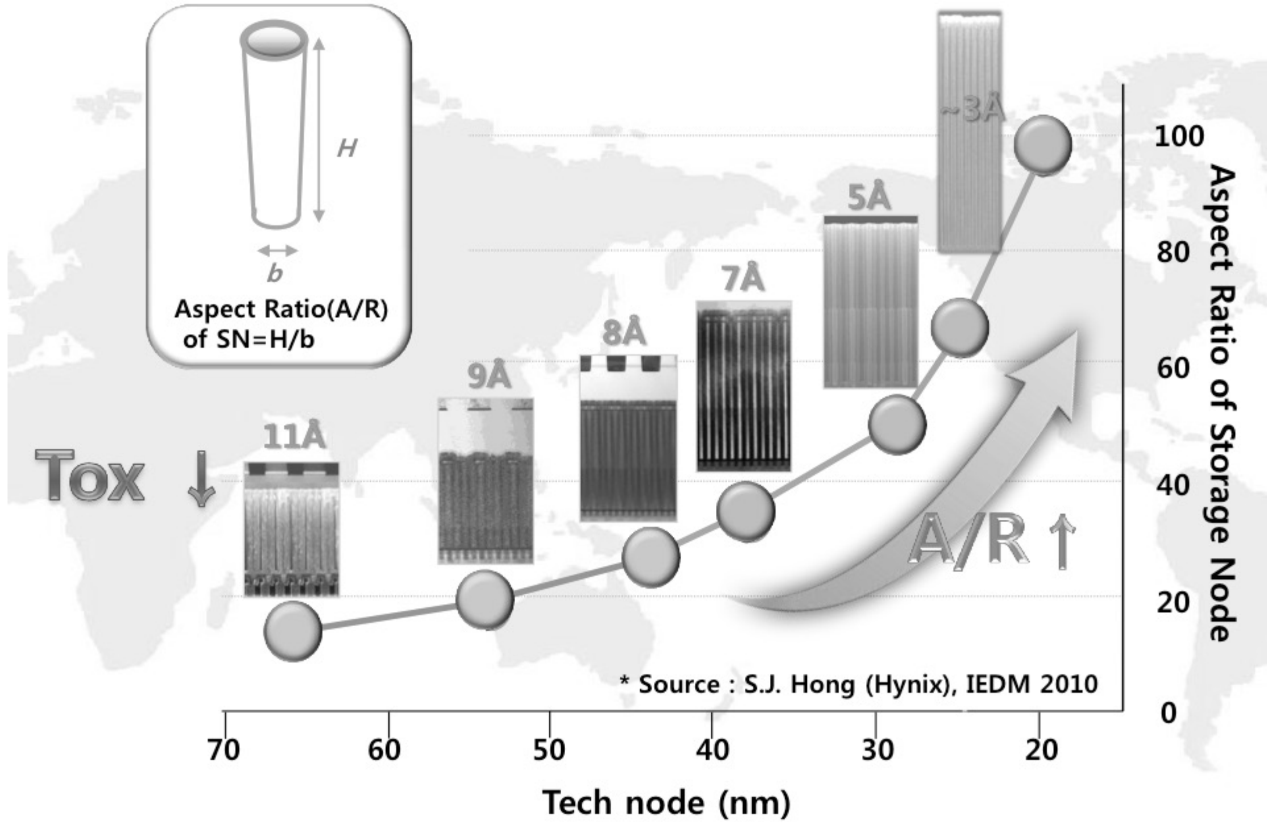
Example: DRAM

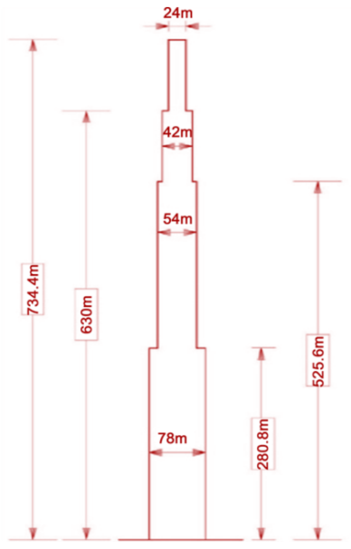


# Memory Scaling in Jeopardy

Scaling of semiconductor memories greatly challenged beyond 20nm

Example: DRAM



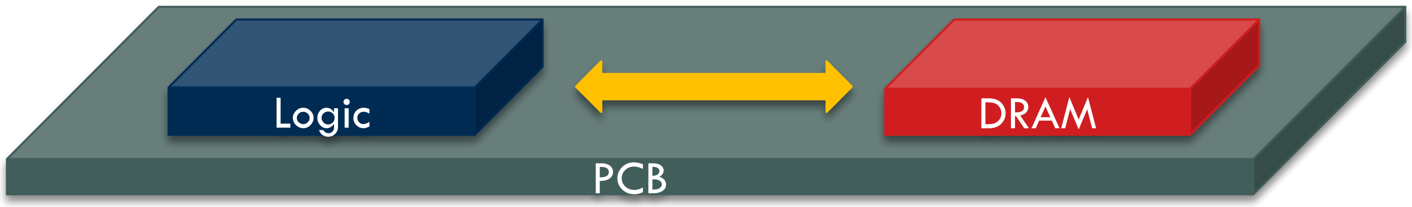


A/R < 10

# Why DRAM Slow?

- Logic VLSI Process: optimized for better transistor performance

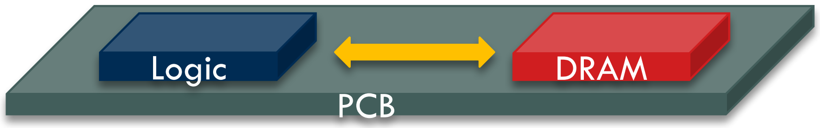
- DRAM VLSI Process: optimized for low cost and low leakage

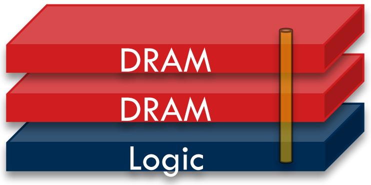


**How to reduce distance?**

## 3D Die-Stacking

- Different devices are stacked on top of each other
- Layers are connected by through-silicon vias (TSVs)

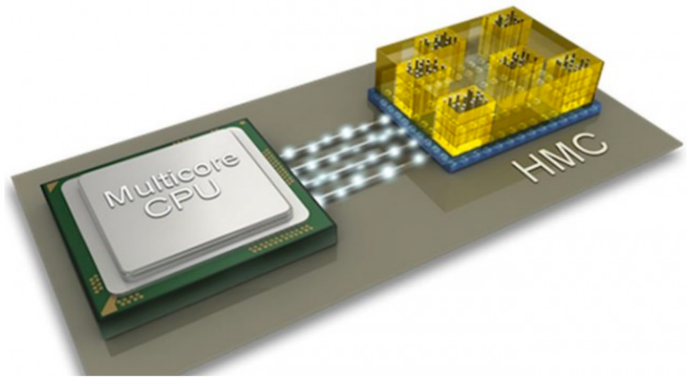




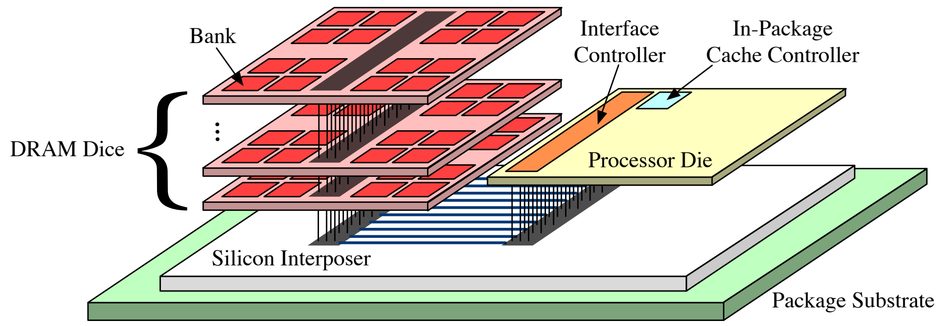
- Why?
  - Communication between devices bottlenecked by limited I/O pins
  - Integrating heterogeneous elements on a single wafer is expensive and suboptimal

# 3D Stacked Memory

- Hybrid Memory Cube (HMC)
  - A logic layer at the bottom



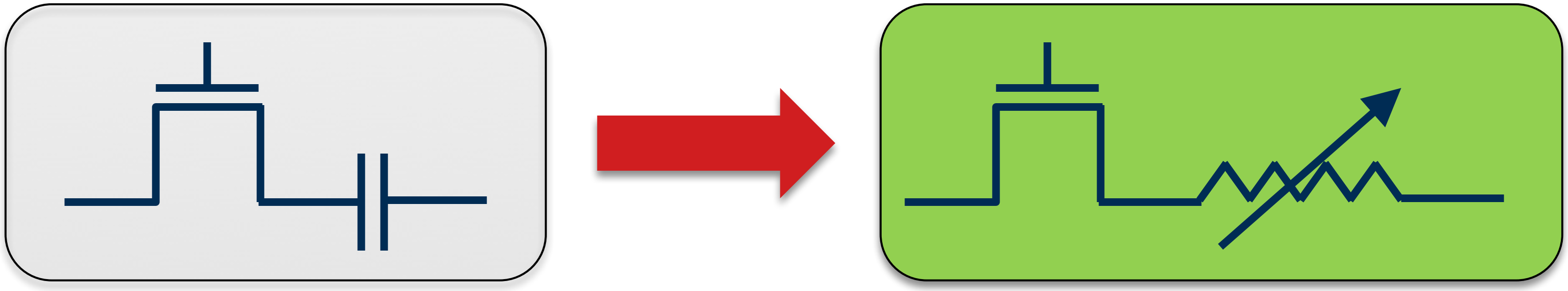
- High Bandwidth Memory (HBM)
  - Silicon interposer at the bottom



## Emerging Non Volatile Memory

## Resistive Memory Technologies

- Key concept: replace DRAM cell capacitor with a programmable resistor

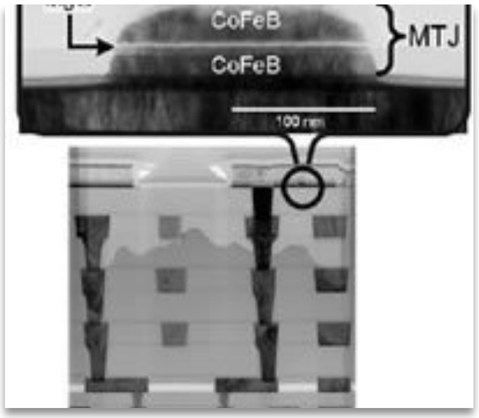


- 1T-1C DRAM
- Charge based sensing
- Volatile

- 1T-1R STT-MRAM, PCM, RRAM
- Resistance based sensing
- Non-volatile

# Leading Contenders

## STT-MRAM



[Halupka, et al. ISSCC'10]

- Limited to single-level cell
- 3D un-stackable

+ High endurance (~$10^{15}$)
+ ~4ns switching time
+ ~50uW switching power

## PCM-RAM

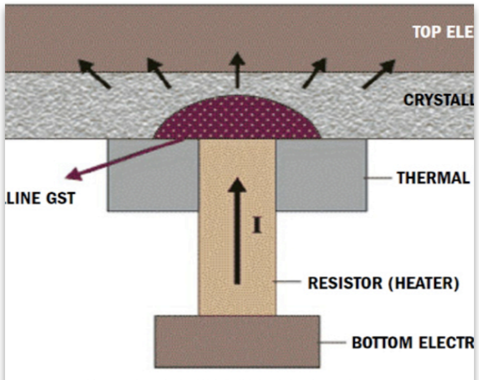


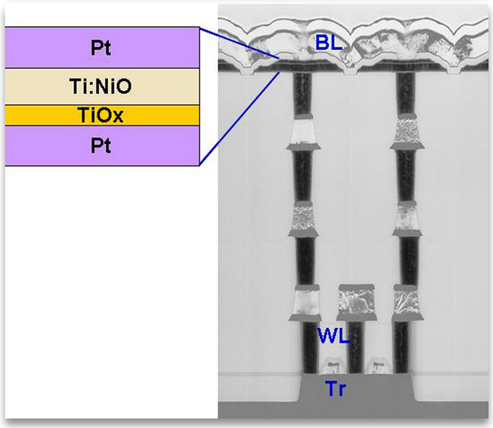
Figure 1. Typical PCM device structure.

[Pronin. EETime'13]

+ Multi-level cell capable
+ $4F^2$ 3D-stackable cell

- Endurance: ~$10^9$ writes
- ~100ns switching time
- ~300uW switching power

## R-RAM



[Henderson. InfoTracks'11]

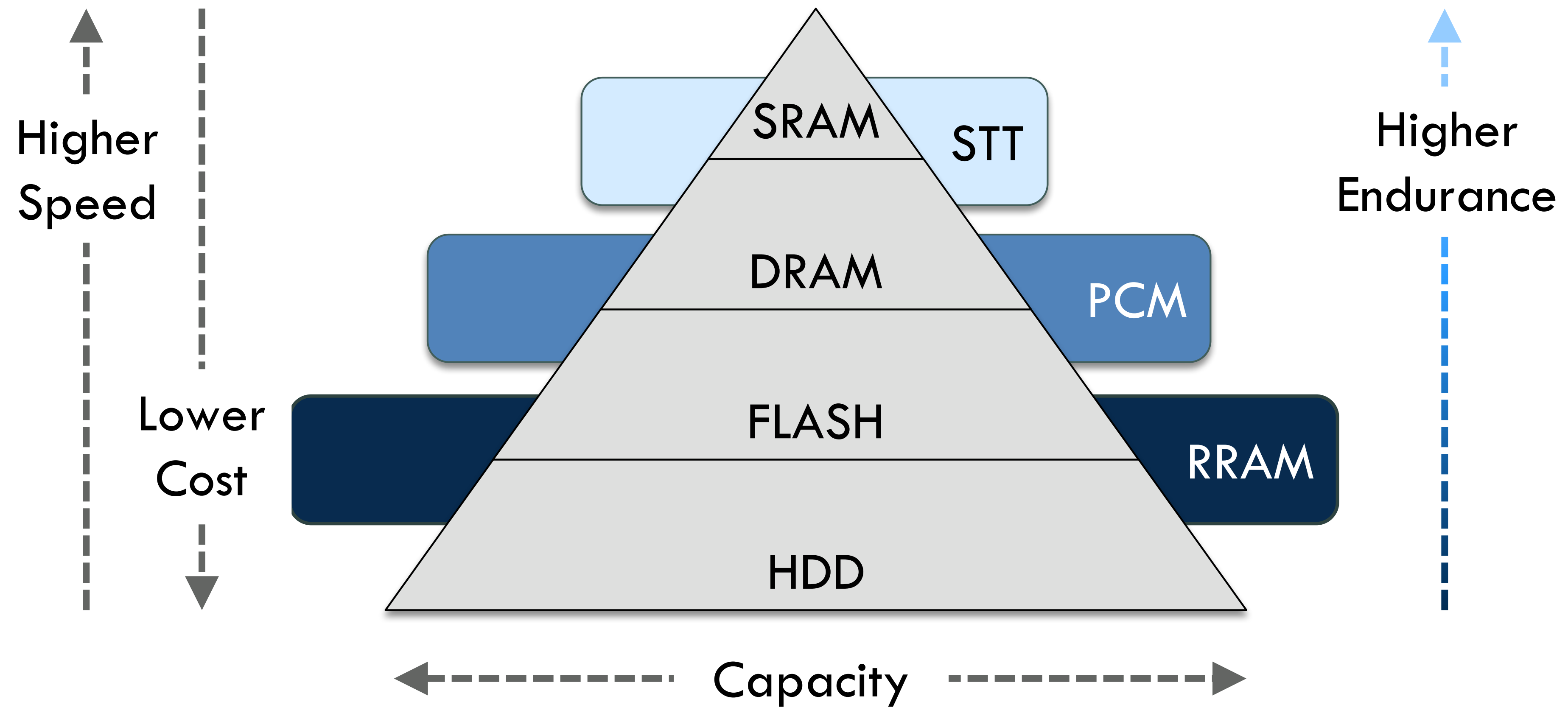
+ Multi-level cell capable
+ $4F^2$ 3D-stackable cell

- Endurance: $10^6$~$10^{12}$ writes

+ ~5ns switching time
+ ~50uW switching power

[ITRS'13]

# Positioning of Resistive Memories


Higher Speed
Lower Cost
SRAM
STT
DRAM
PCM
FLASH
RRAM
HDD
Higher Endurance
Capacity