Memory Hierarchy

Chap. 5.1 – 5.2

Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

Seoul National University

Fall 2022

Memory Technologies

- Static RAM (SRAM)
  - Each cell stores a bit with a four or six-transistor circuit
  - Retains value indefinitely, as long as it is kept powered
  - Faster and more expensive than DRAM
- Dynamic RAM (DRAM)
  - Each cell stores a bit with a capacitor. One transistor is used for access
  - Value must be refreshed every 10 – 100 ms
  - Slower and cheaper than SRAM

Technology	Typical access time	$ Per GiB in 2020
SRAM	0.5 – 2.5ns	$500 – $1000
DRAM	50 – 70ns	$3 – $6
Flash	5μs – 50μs	$0.06 – $0.12
Disk	5ms – 20ms	$0.01 – $0.02

Non-volatile Memories

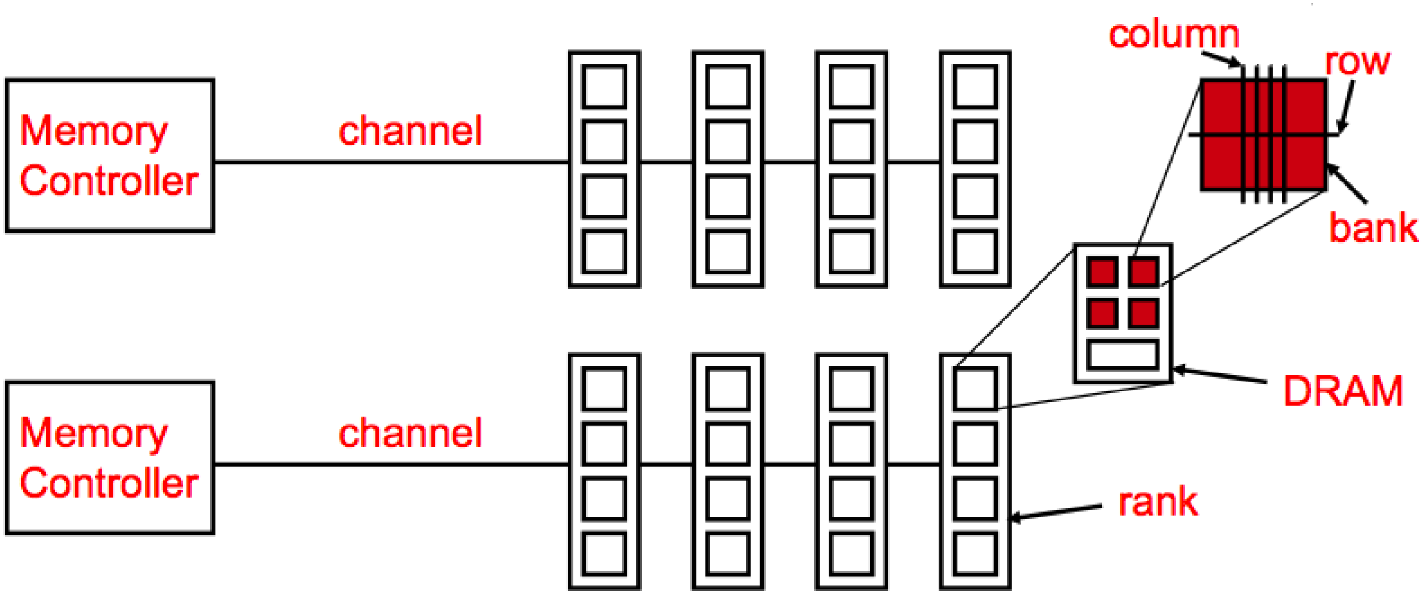
- Nonvolatile memories retain value even if powered off
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Eraseable PROM (EPROM): can be bulk erased (UV, X-ray)
  - Electrically eraseable PROM (EEPROM): electronic erase capability
  - Flash memories: EEPROMs with partial (block-level) erase capability (NOR vs. NAND)
  - Intel Optane memory: slower than DRAM, denser and better cost/GiB than DRAM

- Uses for nonvolatile memories
  - Firmware programs stored in a ROM (BIOS, Disk/network/graphics controllers, …)
  - USB drives, smartphones, tablets, SSDs (Solid-State Drives), disk caches, …
  - Main memory?

DRAM Technology

- Data stored as a charge in a capacitor
  - Single transistor used to access the charge

- Must periodically be refreshed
  - Read contents and write back
  - Performed on a DRAM "row"



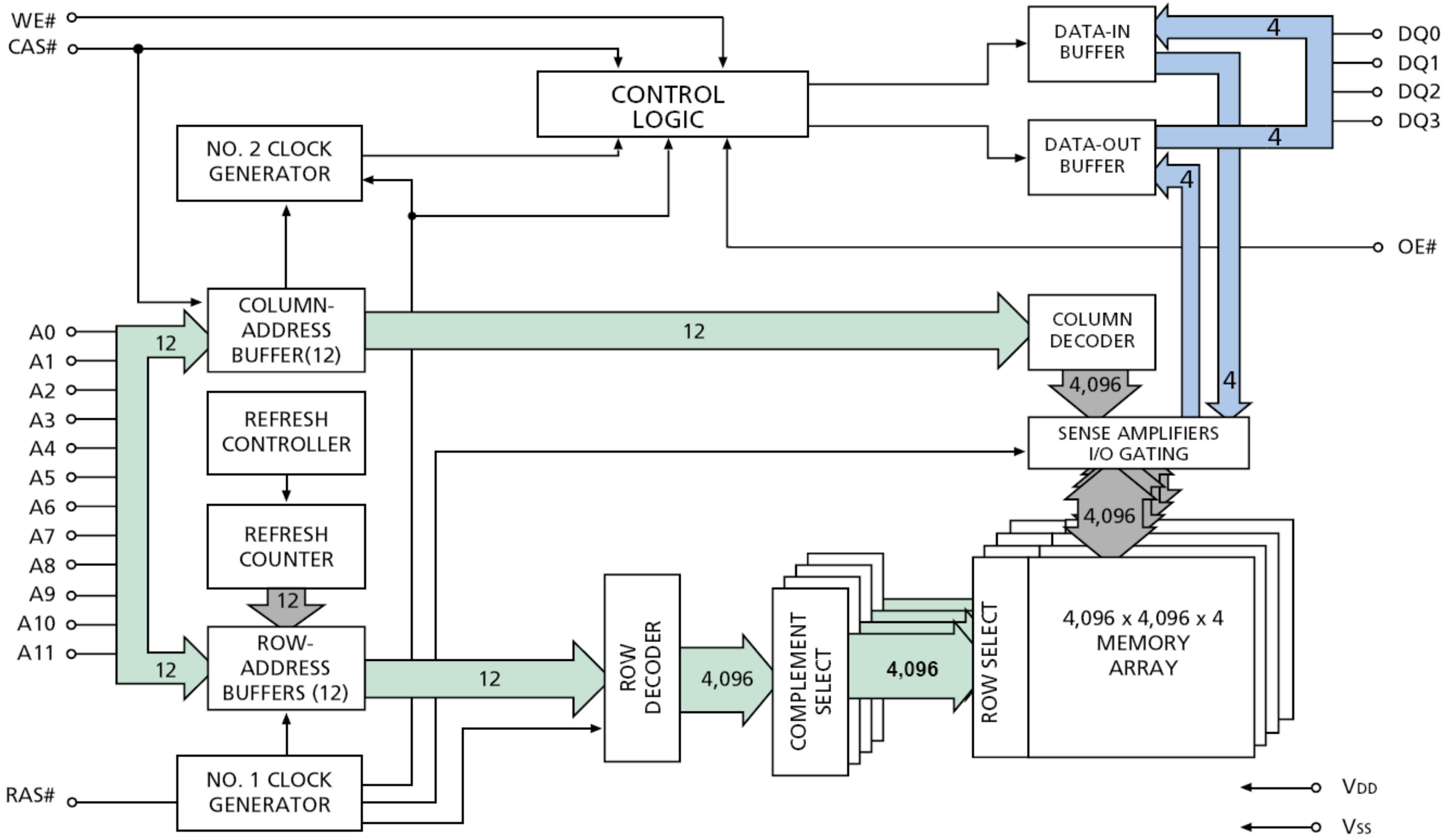
Source: *Onur Mutlu, Memory Overview, Organization & Technology, 2021*

DRAM Configuration

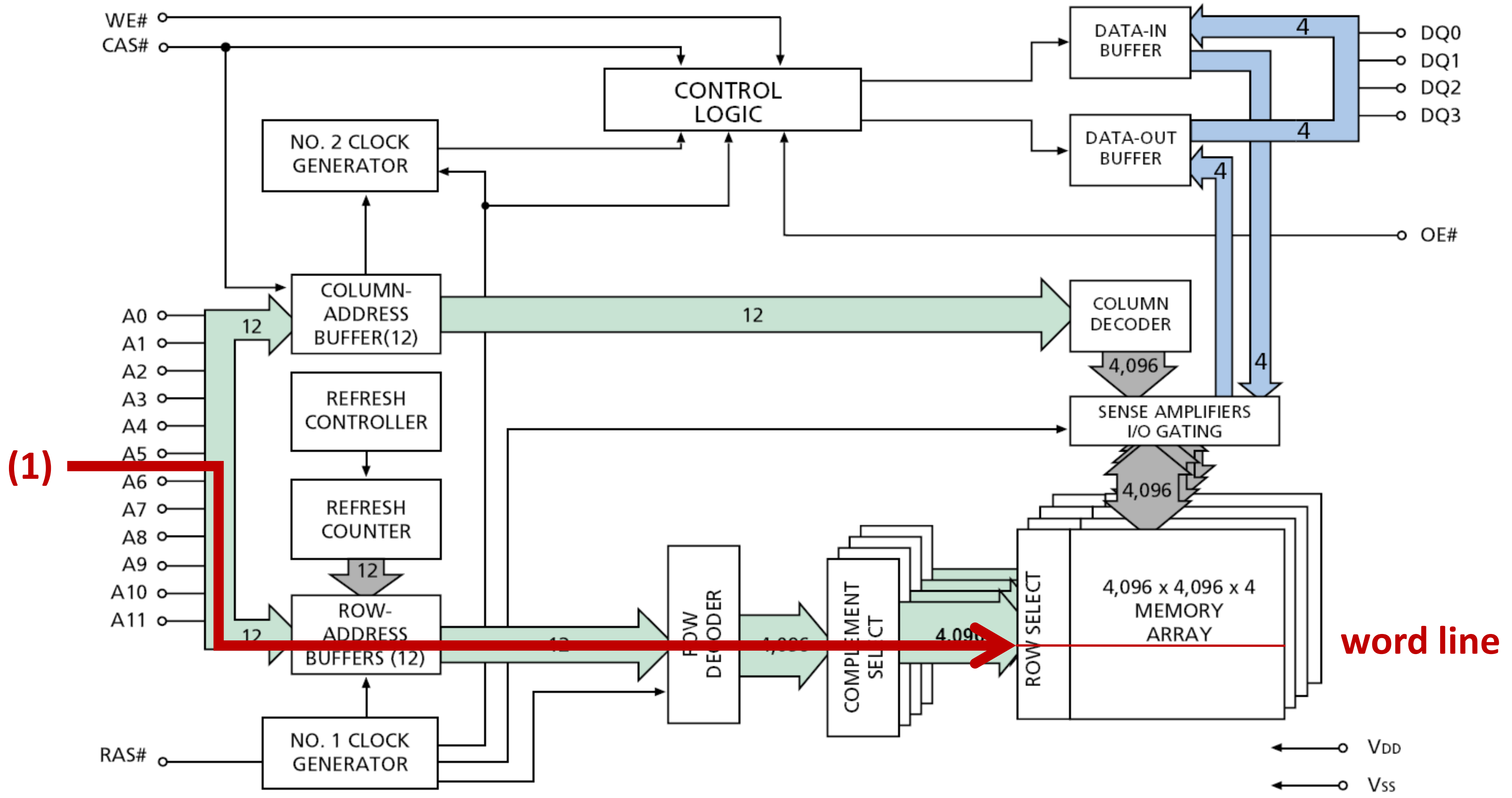
- Asynchronous: no clock
- Large capacity: 1 – 32 Gb
  - Arranged as 2D matrix
  - Minimizes wire length
  - Maximizes refresh efficiency
- Narrow data interface: 1 – 16 bits (x1, x4, x8, x16)
  - Cheap packages → few bus pins
  - Pins are expensive
- Narrow address interface
  - Multiplexed address lines: row and column address
  - Signaled by RAS# and CAS# respectively

DRAM Organization

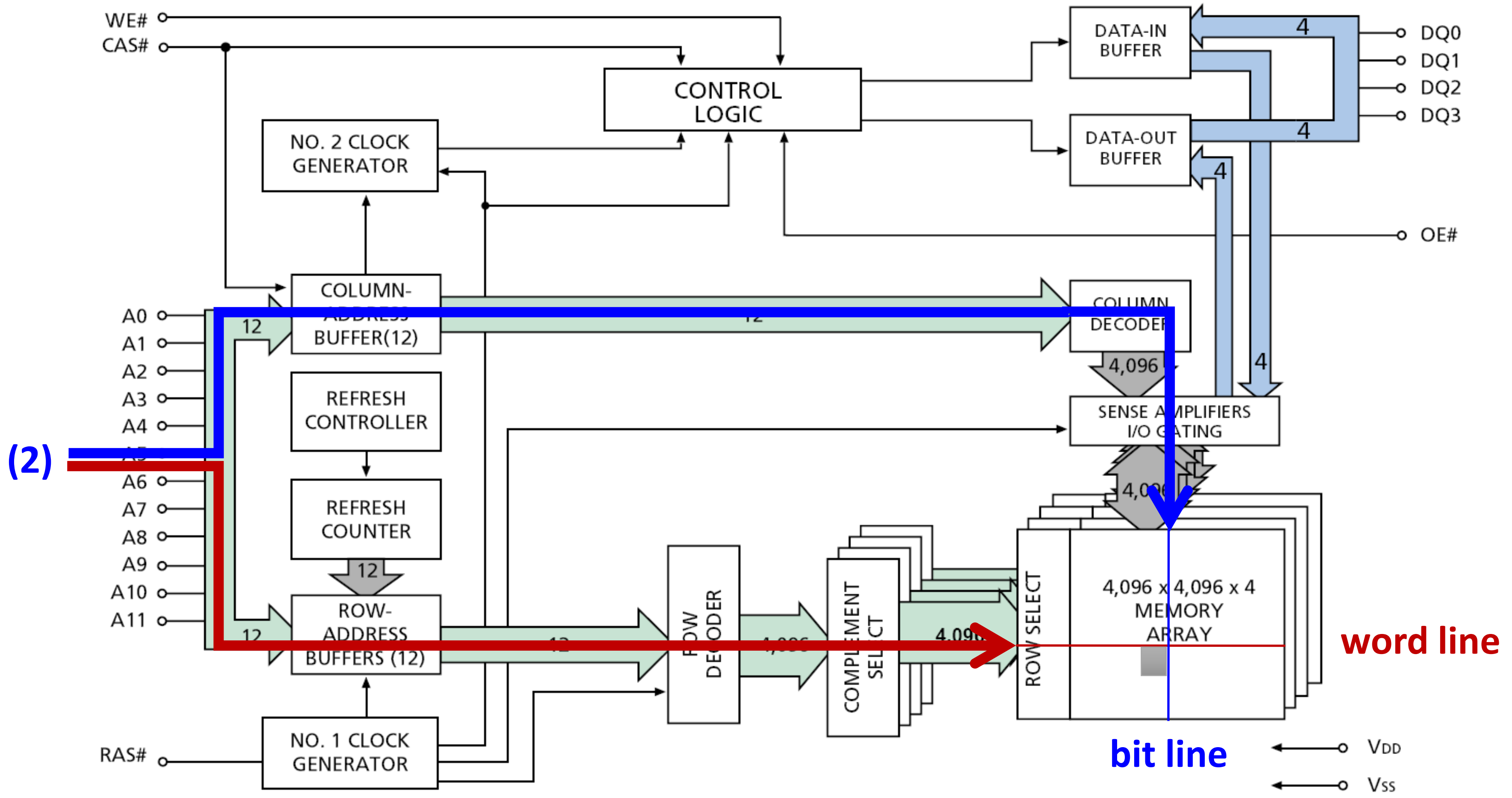
- Micron MT4LC16M4T8 (16M x 4bit)



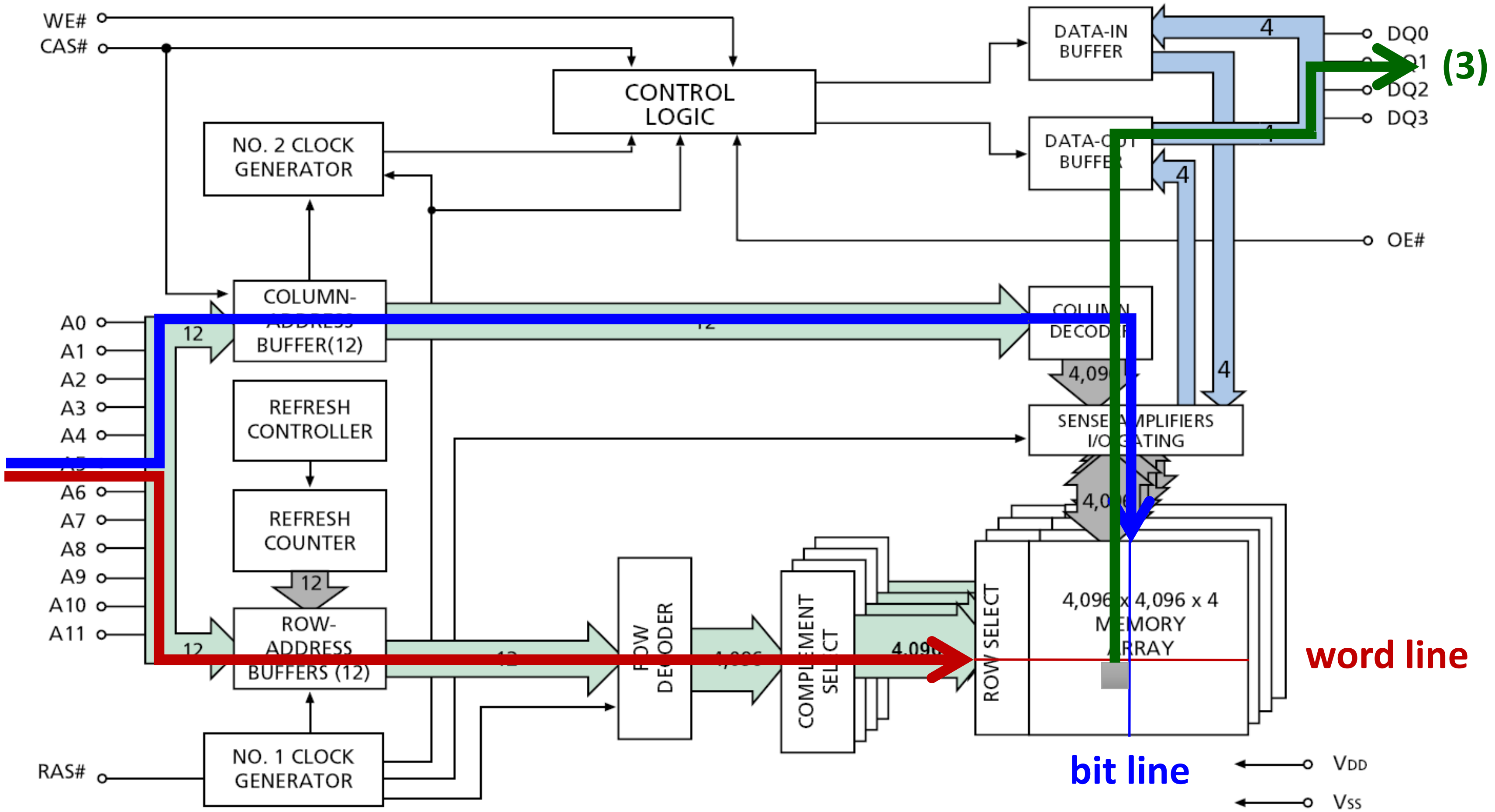
DRAM Read Operation (1)



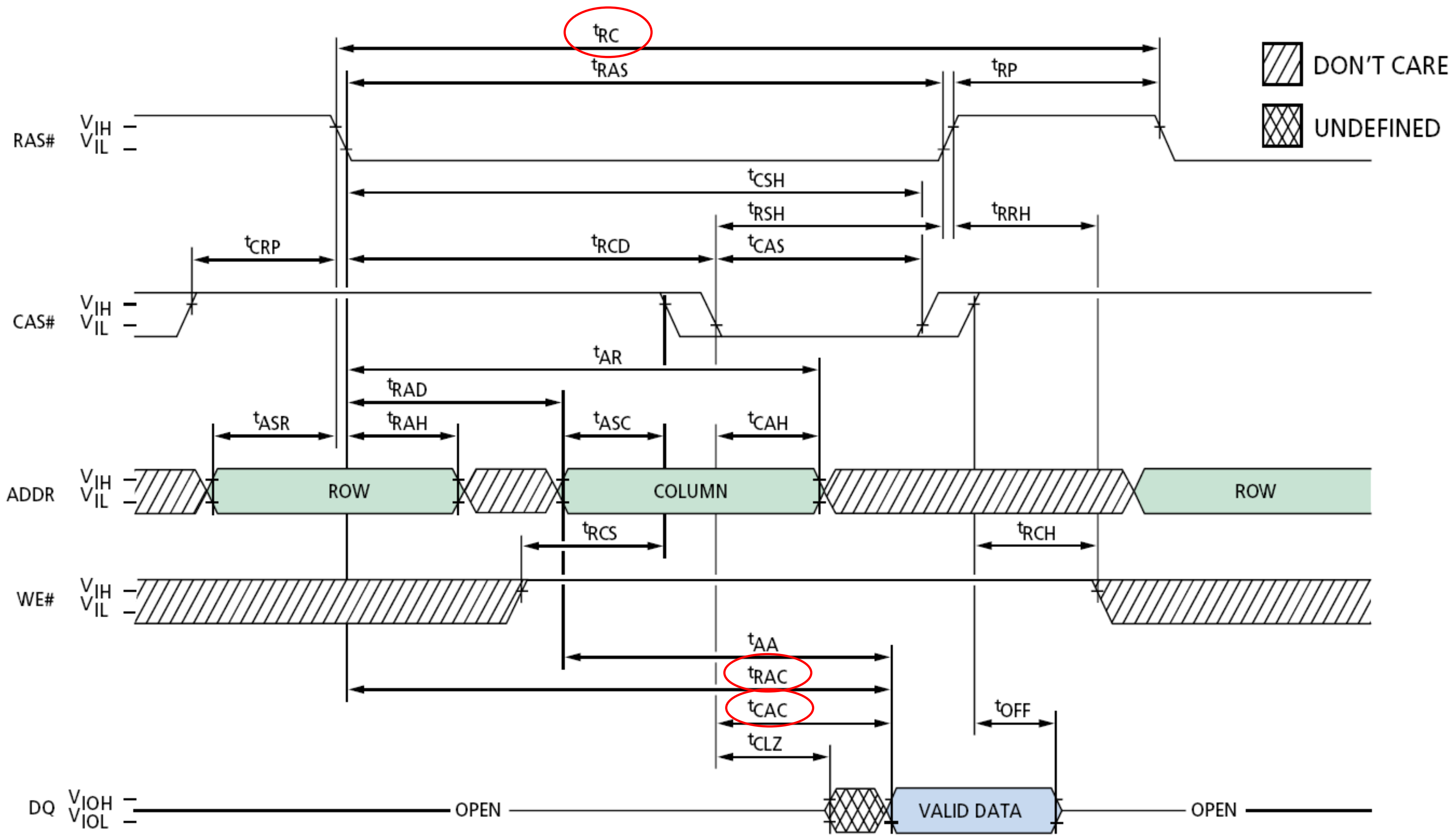
DRAM Read Operation (2)



DRAM Read Operation (3)



DRAM Read Cycle

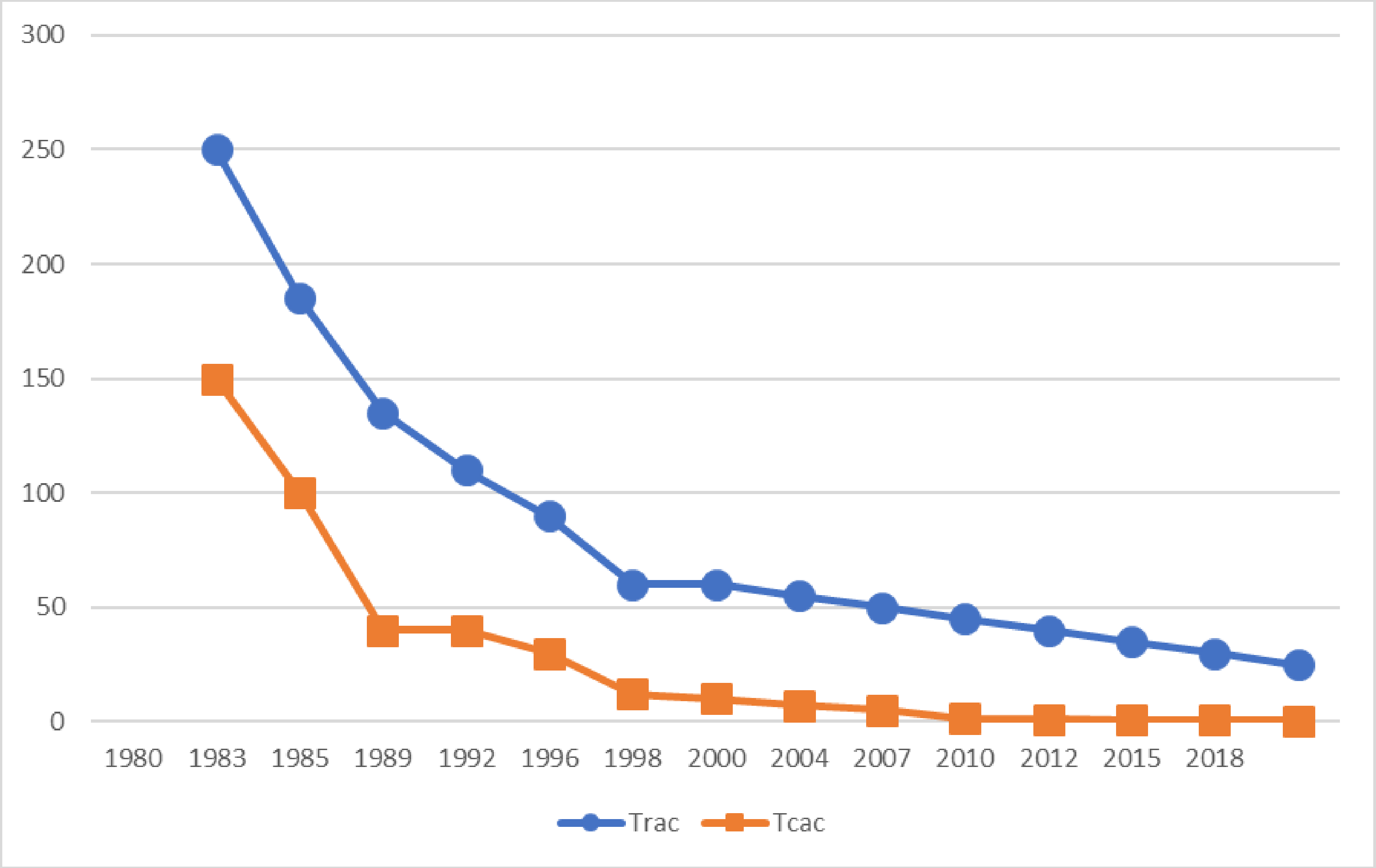


DRAM Timing Parameters

- t_{RC}
  - Minimum time from the start of one row access to the start of the next
  - “Cycle time”
- t_{RAC}
  - Minimum time from RAS# line falling to the valid data output
  - “Access time”
- t_{CAC}
  - Minimum time from CAS# line falling to valid data output

DRAM Generations

Year	Capacity	$/GB
1980	64 Kbit	$6,480,000
1983	256 Kbit	$1,980,000
1985	1 Mbit	$720,000
1989	4 Mbit	$128,000
1992	16 Mbit	$30,000
1996	64 Mbit	$9,000
1998	128 Mbit	$900
2000	256 Mbit	$840
2004	512 Mbit	$150
2007	1 Gbit	$40
2010	2 Gbit	$13
2012	4 Gbit	$5
2015	8 Gbit	$7
2018	16 Gbit	$6

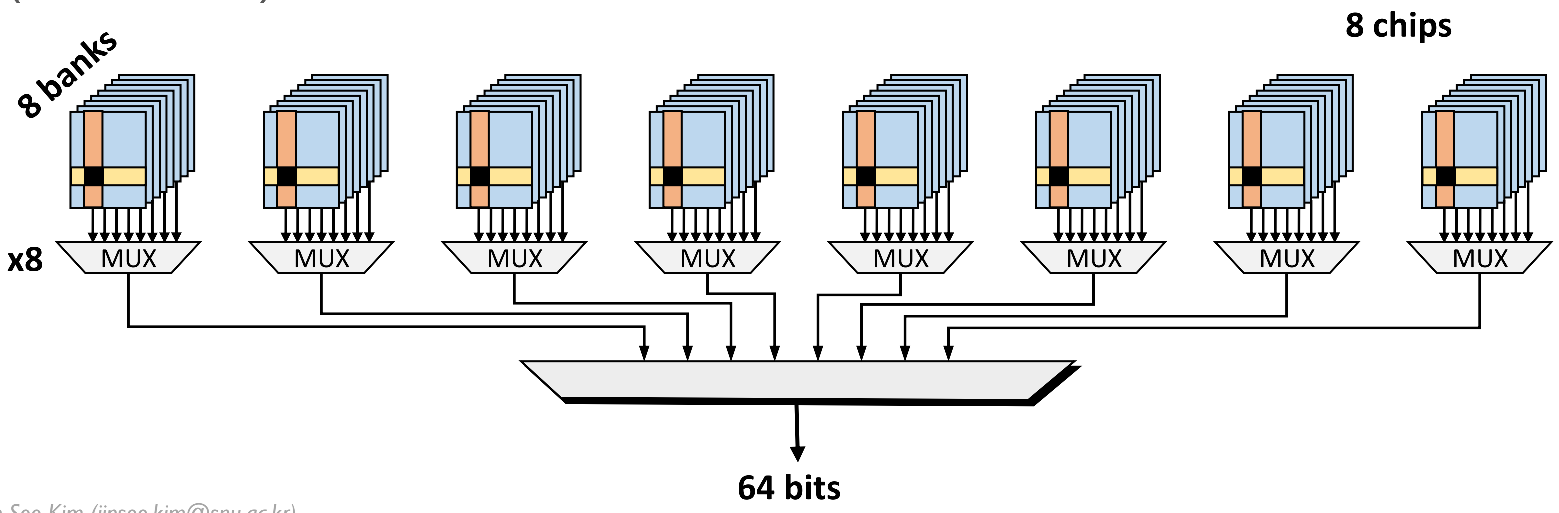


DRAM Performance Factors

- Row buffer
  - Allows several words to be read and refreshed in parallel
- Synchronous DRAM
  - Allows for consecutive accesses in bursts without needing to send each address
  - Improves bandwidth
- DRAM banking
  - Allows simultaneous access to multiple DRAMs
  - Improves bandwidth

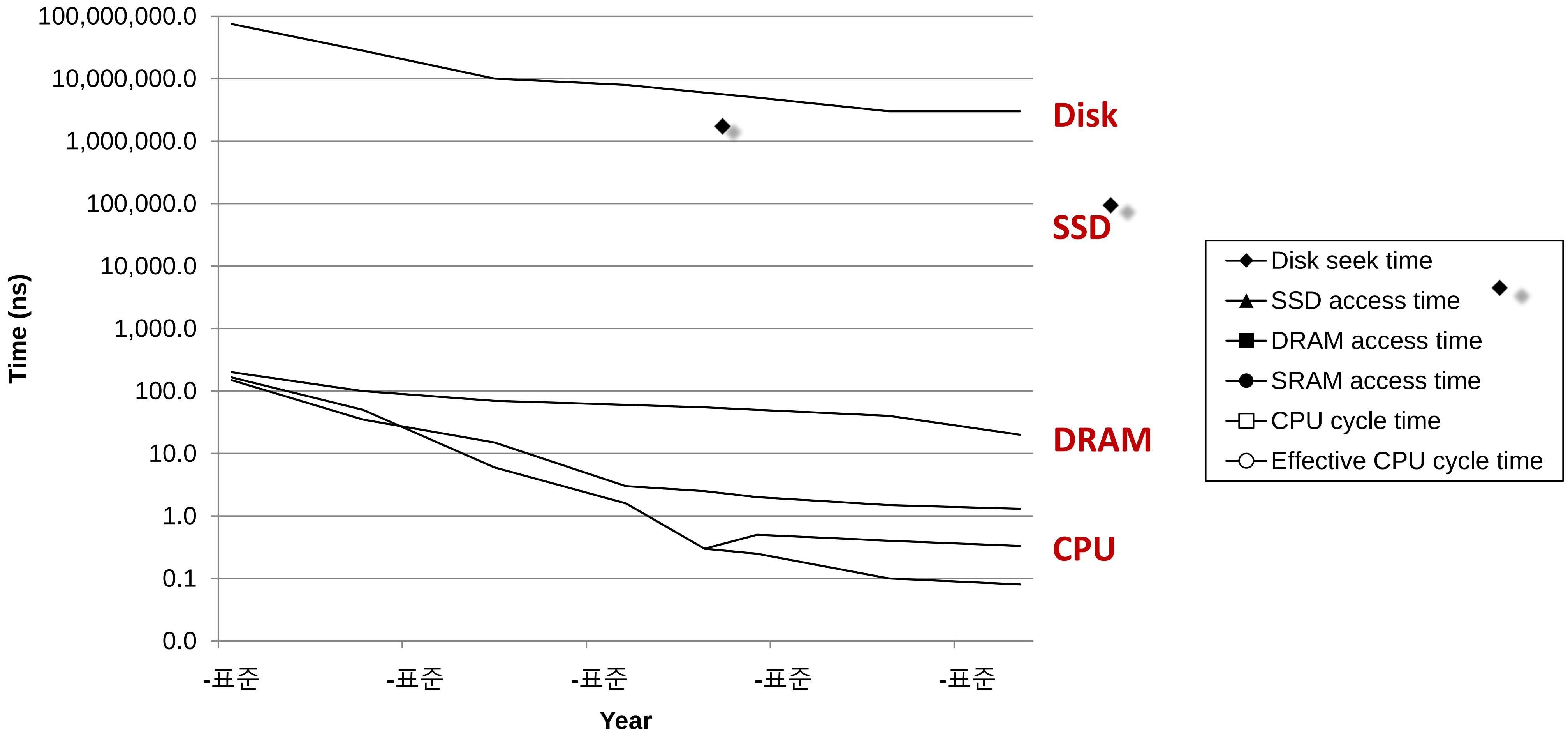
Advanced DRAM Organization

- Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency
- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
  - DDR4-3200 (PC25600): 64bits * 1600MHz * 2 = 25600MB/s



The CPU-Memory Gap

- The gap widens between DRAM, disk, and CPU speeds



Locality to the Rescue!

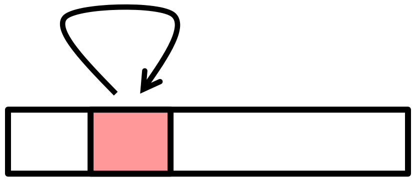
- Question:
  How can we make a memory as fast as SRAM and as cheap as DRAM (or even disk)?

- The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality

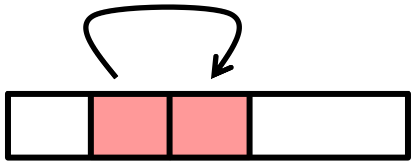
Principle of Locality

- Programs tend to use data and instructions with addresses near or equal to those they have used recently
- Programs access a small portion of their address space at any time

- Temporal locality
  - Recently referenced items are likely to be accessed again soon e.g., instructions in a loop, induction variables



- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - e.g., sequential instruction access, array data



Principle of Locality: Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- **Data**
  - Reference array elements in succession (stride-1 reference pattern) — Spatial locality
  - Reference `sum` each iteration — Temporal locality
- **Instructions**
  - Reference instructions in sequence — Spatial locality
  - Cycle through loop repeatedly — Temporal locality

Memory Hierarchy (1)

- Some fundamental and enduring properties of hardware and software
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!)
  - The gap between CPU and main memory speed is widening
  - Well-written programs tend to exhibit good locality

- These fundamental properties complement each other beautifully

- They suggest an approach for organizing memory and storage systems known as a memory hierarchy

Memory Hierarchy (2)

> **"We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."**
>
> *-- A. W. Burks, H. H. Goldstein, J. von Neumann, Preliminary Discussion of the Logical Design of Electronic Computing Instrument, June 1946.*

- Taking advantage of locality
  - Store everything on disk
  - Copy recently accessed (and nearby) items from disk to smaller DRAM memory (main memory)
  - Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory (cache memory)

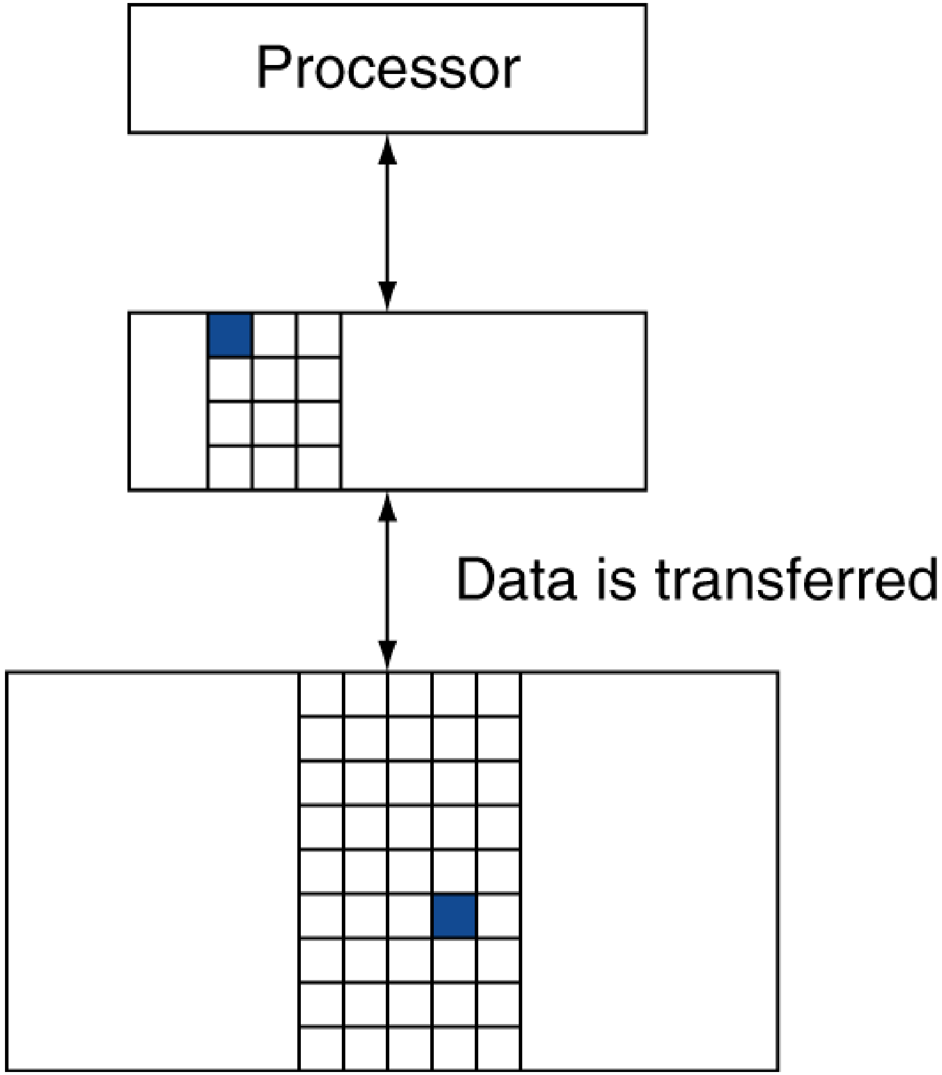
Caches

- A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device
- Fundamental idea of a memory hierarchy
  - For each *k*, the faster, smaller device at level *k* serves a cache for the larger, slower device at level *k*+*1*
- Why do memory hierarchies work?
  - Because of locality, programs tend to access the data at level *k* more often than they access the data at level *k*+*1*

Big Idea: The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top

Memory Hierarchy Levels

- Block (or line): a unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
  - Hit ratio: hits / accesses
- If accessed data is absent
  - Miss: block copied from lower level
  - Then accessed data supplied from upper level
  - Time taken: miss penalty
  - Miss ratio: misses / accesses = 1 – hit ratio



Memory Hierarchy