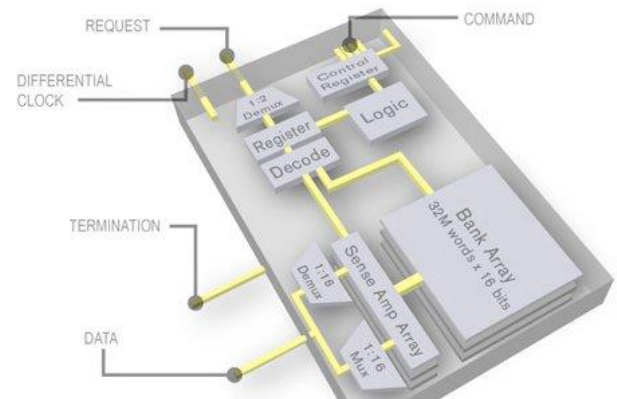


---

# Logic and Computer Design Fundamentals

## Chapter 7 – Memory Basics



Ming Cai

cm@zju.edu.cn

College of Computer Science and Technology  
Zhejiang University

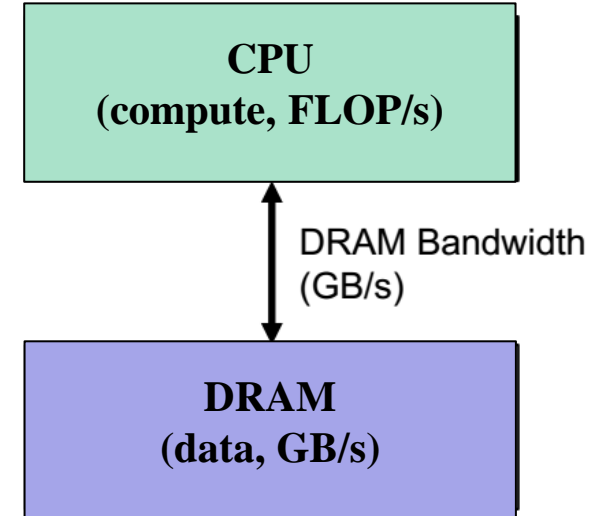
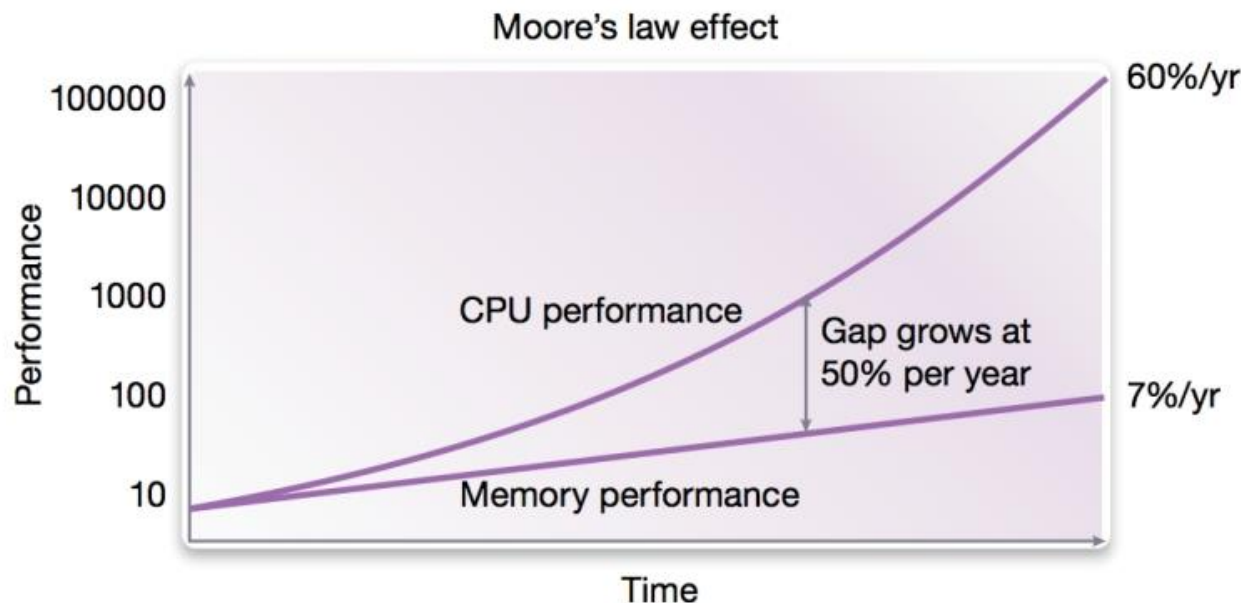
# Overview

---

- **Memory definitions**
- **Random Access Memory (RAM)**
- **Static RAM (SRAM) integrated circuits**
  - Cells and slices
  - Cell arrays and coincident selection
- **Arrays of SRAM integrated circuits**
- **Dynamic RAM (DRAM) integrated circuits**
- **DRAM Types**
  - Synchronous (SDRAM)
  - Double-Data Rate (DDR SDRAM)
  - RAMBUS DRAM (RDRAM)
- **Arrays of DRAM integrated circuits**

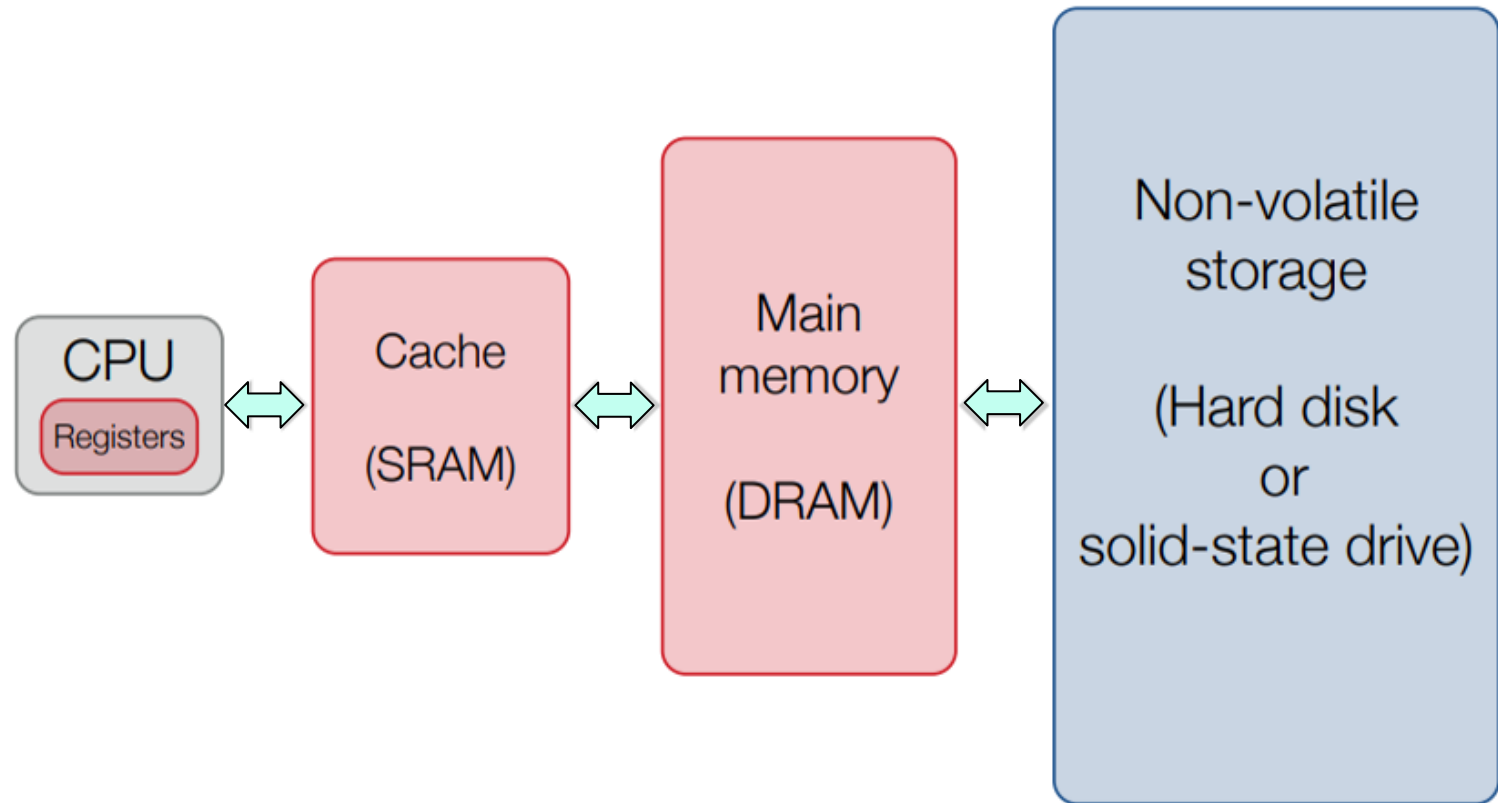
# Memory Wall

- The speed disparity between CPU and memory continues to grow.
- The processor performance is limited by memory (**memory wall**).



# Memory Hierarchy in Computers

- **Good memory hierarchy design is important to the overall performance of a computer system.**

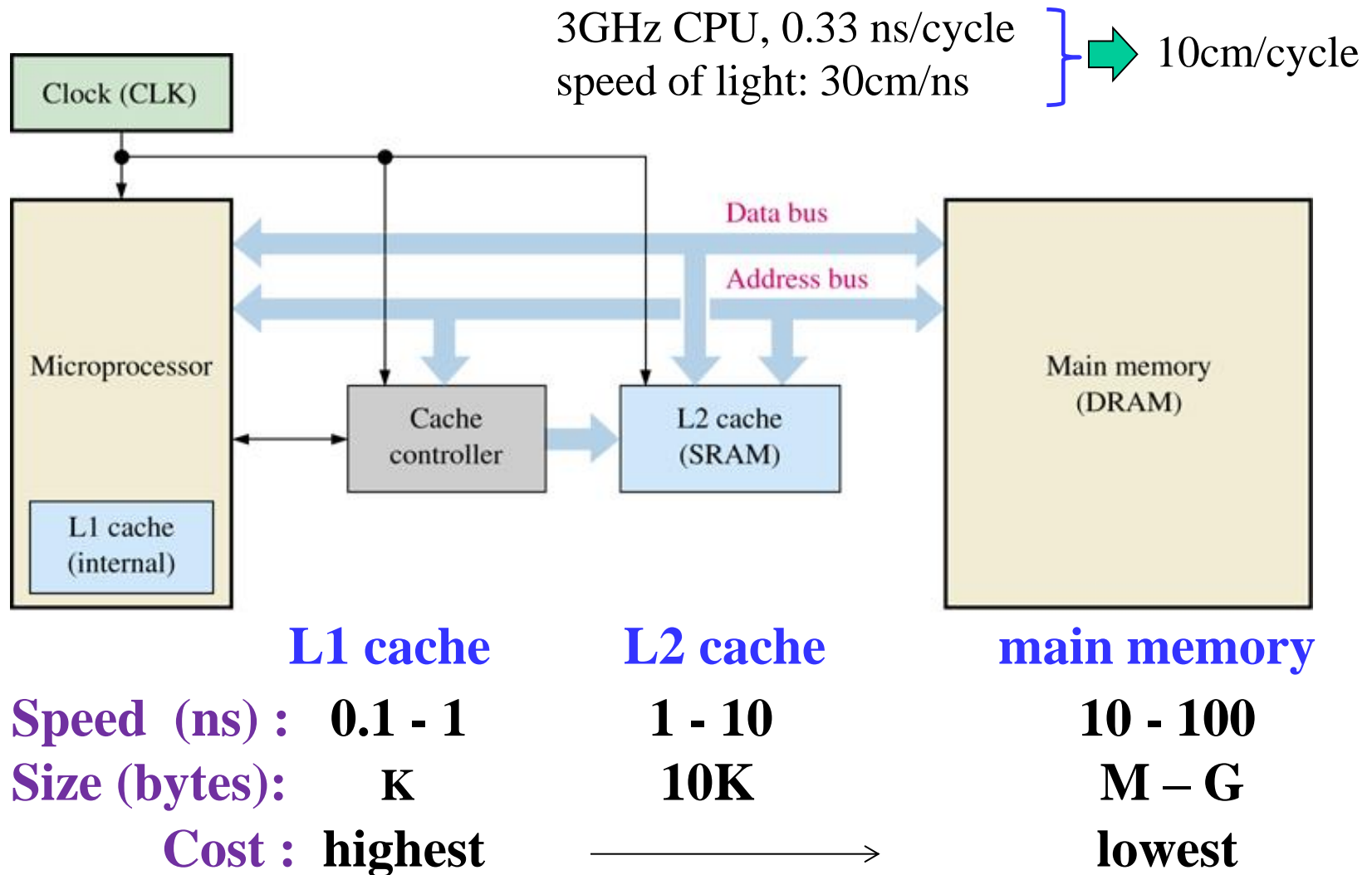


Fast access to small  
amount of data



Slow access to large  
amount of data

# Memory Hierarchy in Computers (continued)



# Latency Comparison Numbers

1	Latency Comparison Numbers (~2012)		
2	-----		
3	L1 cache reference	0.5	ns
4	Branch mispredict	5	ns
5	L2 cache reference	7	ns
6	Mutex lock/unlock	25	ns
7	Main memory reference	100	ns
8	Compress 1K bytes with Zippy	3,000	ns
9	Send 1K bytes over 1 Gbps network	10,000	ns
10	Read 4K randomly from SSD*	150,000	ns
11	Read 1 MB sequentially from memory	250,000	ns
12	Round trip within same datacenter	500,000	ns
13	Read 1 MB sequentially from SSD*	1,000,000	ns
14	Disk seek	10,000,000	ns
15	Read 1 MB sequentially from disk	20,000,000	ns
16	Send packet CA->Netherlands->CA	150,000,000	ns

from <https://gist.github.com/jboner/2841832>

# Memory Definitions

---

- **Memory** — A collection of storage cells together with the necessary circuits to transfer information to and from them.
- **Memory Organization** — the basic architectural structure of a memory in terms of how data is accessed.
- **Random Access Memory (RAM)** — a memory organized such that data can be transferred to or from any cell (or collection of cells) in a time that is not dependent upon the particular cell selected.
- **Memory Operations** — operations on memory data supported by the memory unit. Typically, *read* and *write* operations over some data element (bit, byte, word, etc.).

# Memory Definitions (Continued)

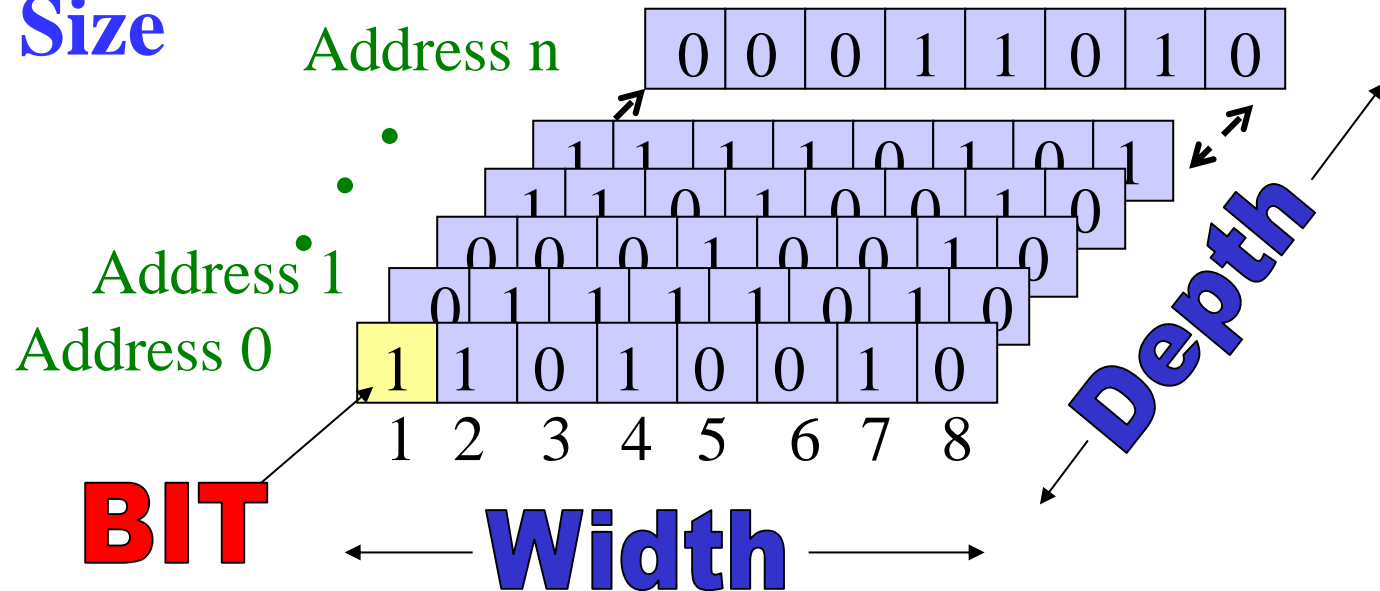
---

- **Typical data elements are:**
  - bit — a single binary digit
  - byte — a collection of eight bits accessed together
  - word — a collection of binary bits whose size is a typical unit of access for the memory. It is a power of two multiple of bytes (e.g., 2 bytes, 4 bytes, etc.)
- Memory Data — a bit or a collection of bits to be stored into or accessed from memory cells.
- Memory Address — A vector of bits that identifies a particular memory element.
- Memory Size — address width  $\times$  word width, e.g.,  $2K \times 8$ ,  $32M \times 16$ .



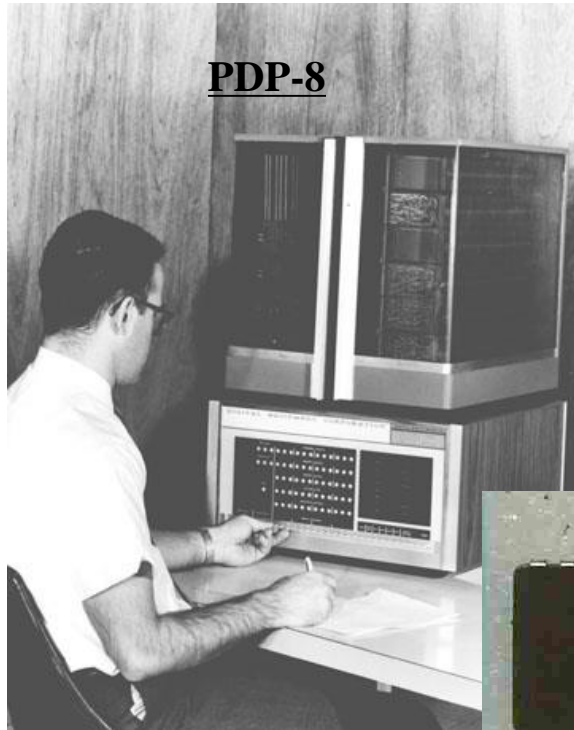
# Memory Definitions (Continued)

- **Memory Depth** (Words #)
- **Data Element**: bit, byte, word
- **Memory Width** (Bits # per word)
- **Memory Size**



$$\text{Memory Size} = \text{Memory Depth} \times \text{Memory Width}$$

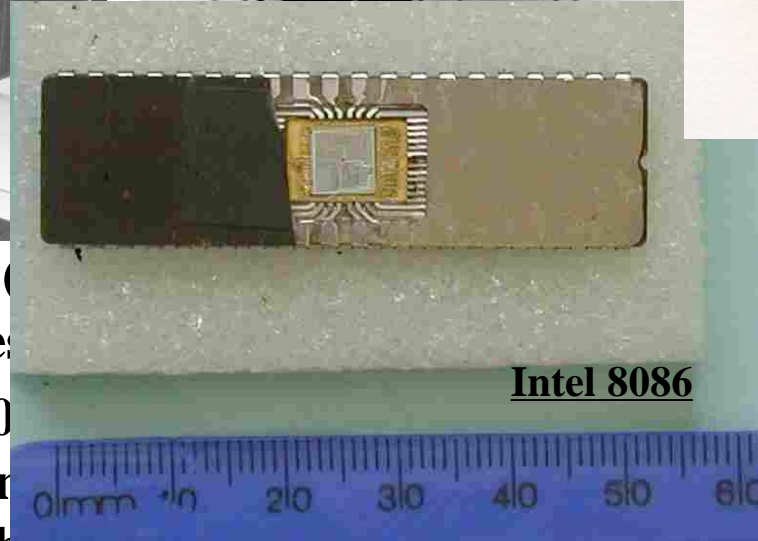
# Memory Organization



PDP-8



- IBM 360 (1964) used 8-bit bytes
- Intel 8080 (1974) used a 12-bit address 16,777,216
- current Intel 8086 and the 8086 and the address to address 65,536 8-bit bytes.



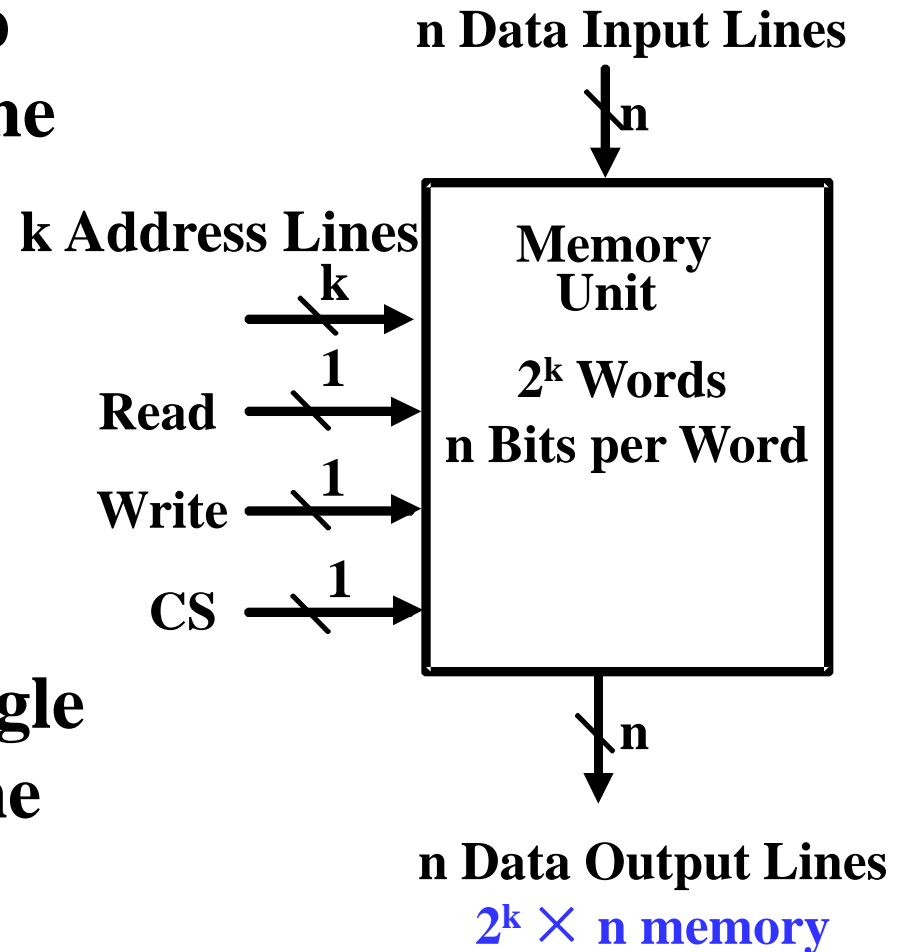
Intel 8086

**“640K ought to be enough for anybody.”**

**——Bill Gates (1981)**

# Memory Block Diagram

- **Chip select (CS)** or **chip enable (CE)** to enable the memory.
- **k address lines** are decoded to address  $2^k$  words of memory.
- Each word is  $n$  bits.
- **Read** and **Write** are single control lines defining the simplest of memory operations.



# Memory Organization Example

---

- **Example memory contents:**
  - A memory with 3 address bits and 8 data bits has:
  - $k = 3$  and  $n = 8$  so  $2^3 = 8$  addresses labeled 0 to 7.
  - $2^3 = 8$  words of 8-bit data

Memory Address		Memory Content
Binary	Decimal	
000	0	1 0 0 0 1 1 1 1
001	1	1 1 1 1 1 1 1 1
010	2	1 0 1 1 0 0 0 1
011	3	0 0 0 0 0 0 0 0
100	4	1 0 1 1 1 0 0 1
101	5	1 0 0 0 0 1 1 0
110	6	0 0 1 1 0 0 1 1
111	7	1 1 0 0 1 1 0 0

# Basic Memory Operations

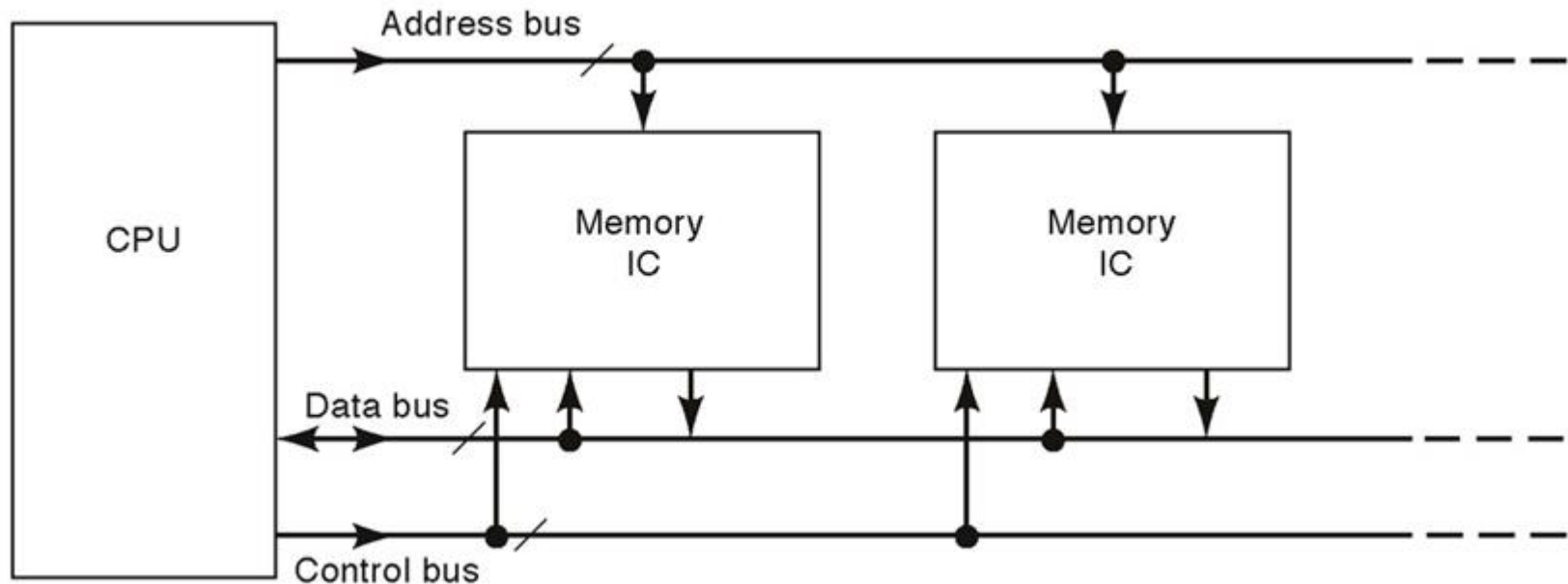
---

- **Memory operations require the following:**
  - ***Address*** — specifies the memory location to operate on. The address lines carry this information into the memory. Typically:  $n$  bits specify locations of  $2^n$  words.
  - ***Data*** — data written to, or read from, memory as required by the operation.
  - **An operation** — Information sent to the memory and interpreted as control information which specifies the type of operation to be performed. Typical operations are **READ** and **WRITE**. Others are READ followed by WRITE and a variety of operations associated with delivering blocks of data. Operation signals may also specify timing information.

# Basic Memory Operations

---

- **Three buses connect the memory to CPU:**
  - *Address bus* — specifies the memory location.
  - *Data bus* — contains data to/from memory location.
  - *Control bus* — additional control signals such as chip select, read/write, etc.



# Basic Memory Operations (continued)

- **Read Memory** — an operation that reads a data value stored in memory:

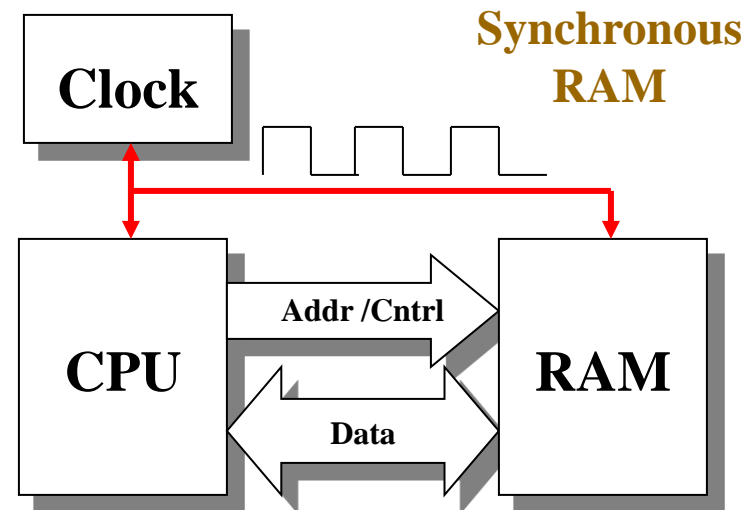
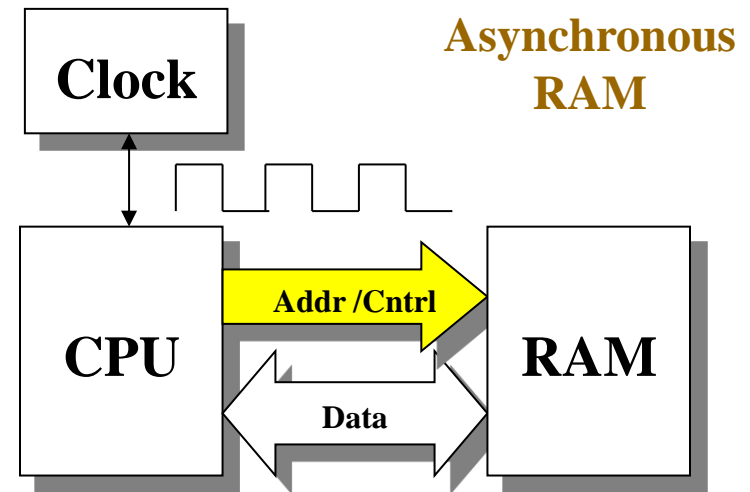
1. Place an address on the address lines.
2. Toggle the memory read control line.
3. Wait for the read data to become stable.

Chip Select CS	Read/Write R/W	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

- **Write Memory** — an operation that writes a data value to memory:
  1. Place an address on the address lines and valid data on the data lines.
  2. Toggle the memory write control line.
- Sometimes the read or write enable line is defined as a clock with precise timing information (e.g. Read Clock, Write Strobe).
  - Otherwise, it is just an interface signal.
  - Sometimes memory must acknowledge that it has completed the operation.

# Asynchronous vs. Synchronous RAM

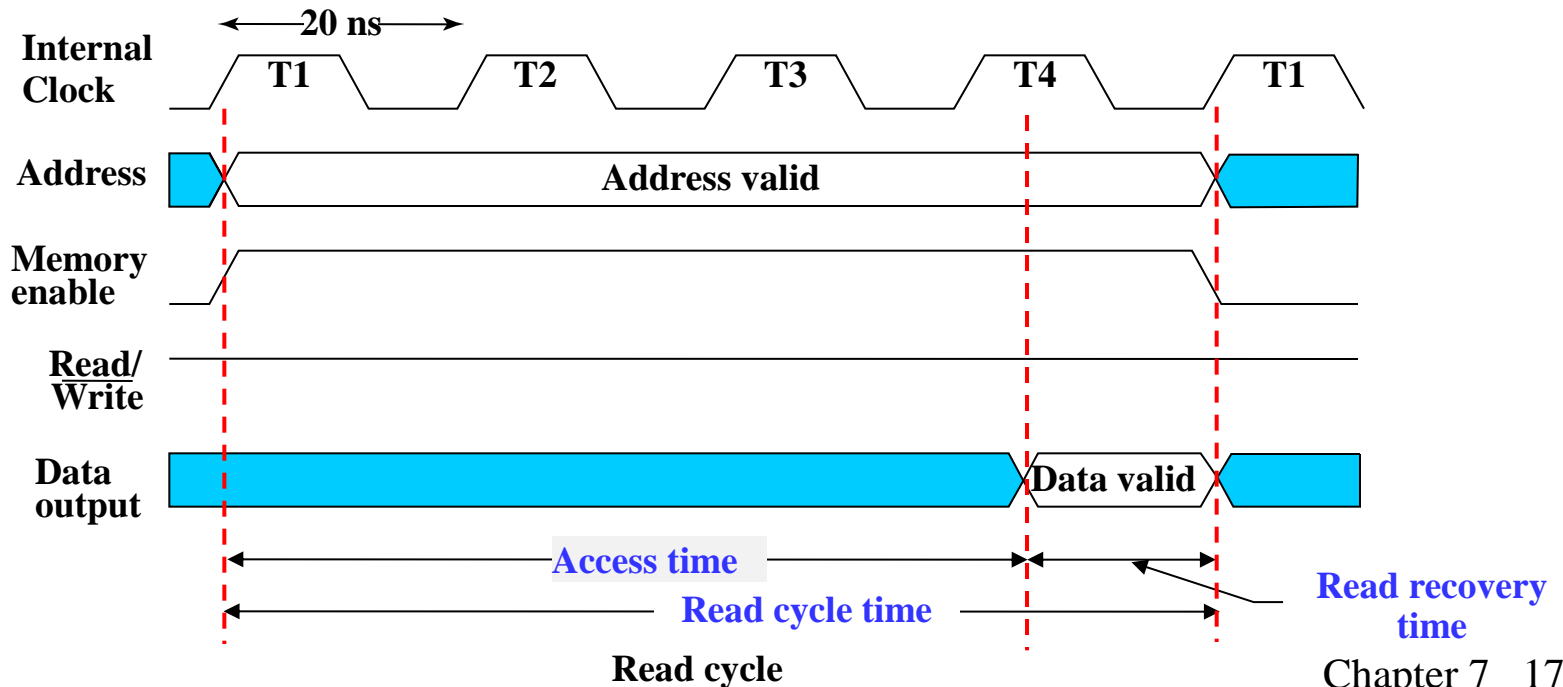
- An **asynchronous RAM** does not depend on the state of an external clock. It will read or write data as soon as it receives the instruction.
- The **synchronous RAM** is synchronized with an external clock. It will read and write data into the memory on particular states of the clock.





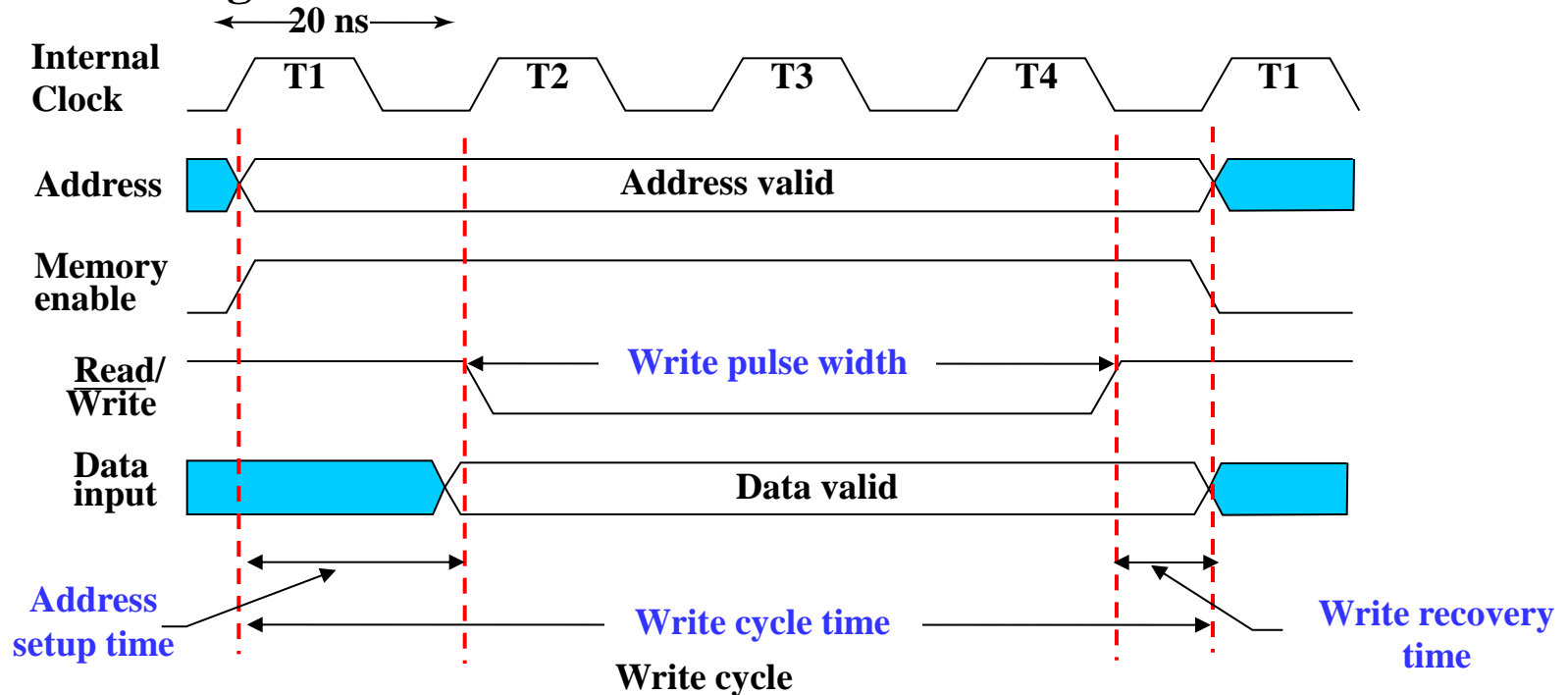
# Memory Operation Timing

- Most basic memories are **asynchronous**
  - Storage in latches or storage of electrical charge
  - No external clock
- Controlled by control inputs and address
- Timing of signal changes and data observation is critical to the operation
- Read timing:



# Memory Operation Timing

- Write timing:

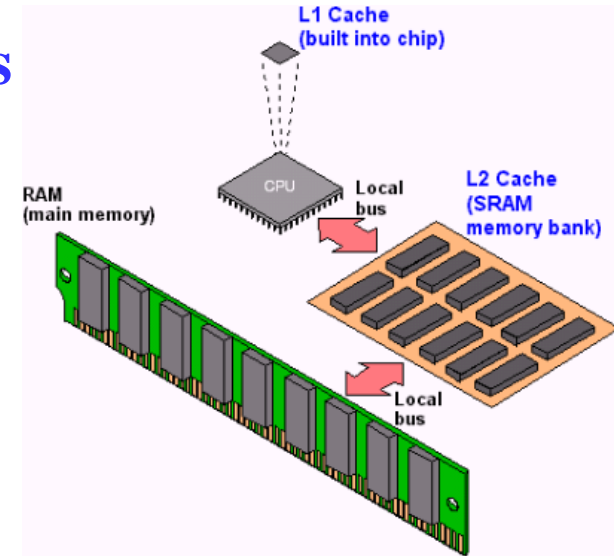


- Critical times measured with respect to edges of write pulse (1-0-1):

- Address must be established at least a specified time before 1-0 and held for at least a specified time after 0-1 to avoid disturbing stored contents of other addresses.
- Data must be established at least a specified time before 1-0 and held for at least a specified time after 0-1 to write correctly.

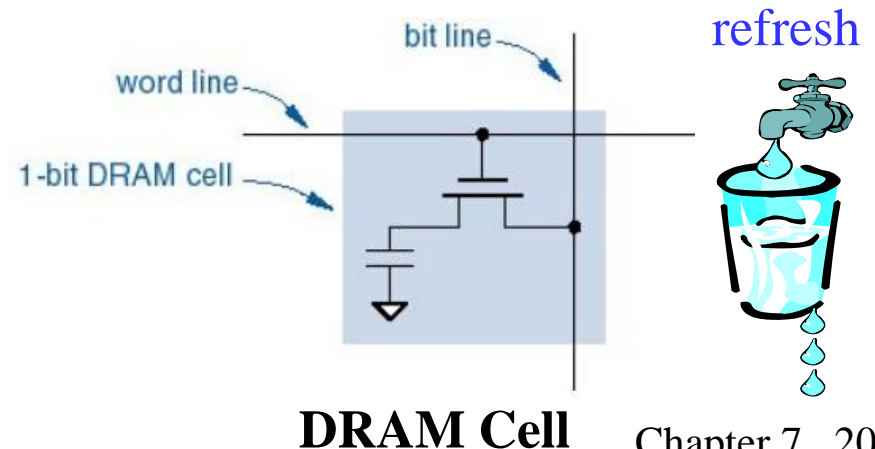
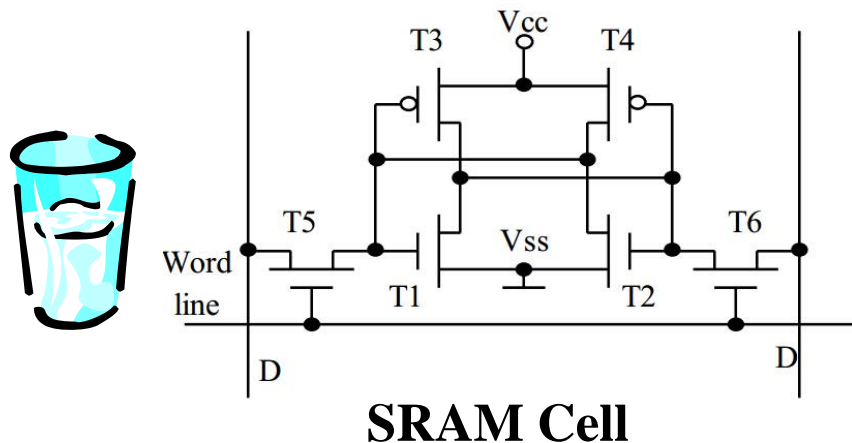
# RAM Integrated Circuits

- Types of random access memory
  - *Static* – information stored in **latches**
  - *Dynamic* – information stored as **electrical charges** on **capacitors**
    - Charge “leaks” off
    - Periodic *refresh* of charge required
- Dependence on Power Supply
  - *Volatile* – loses stored information when power turned off
  - *Non-volatile* – retains information when power turned off



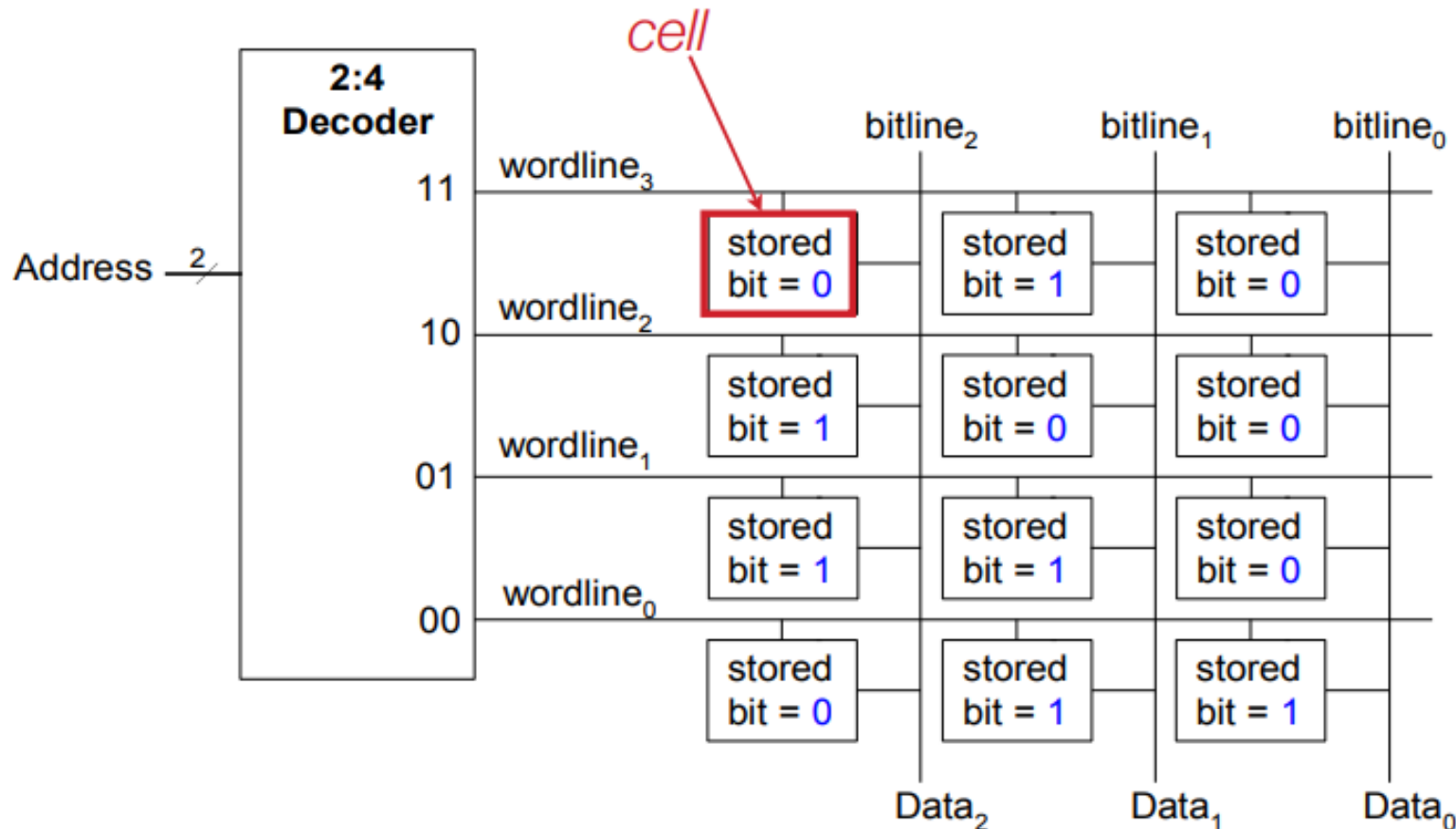
# Static vs. Dynamic RAM

Static RAM	Dynamic RAM
SRAM uses <b>transistor</b> to store data	DRAM uses <b>capacitor</b> to store data
SRAM does <b>not need periodic refreshment</b> to maintain data	DRAM needs <b>periodic refreshment</b> to maintain data
SRAM's structure is <b>complex</b>	DRAM's structure is <b>simple</b>
SRAM is <b>expensive</b>	DRAM is less <b>expensive</b>
SRAM is <b>faster</b>	DRAM is <b>slower</b>
SRAM is used in <b>Cache memory</b>	DRAM is used in <b>main memory</b>



# Memory Array Architecture

- Memory is a 2D array of bits. Each bit stored in a **cell**.

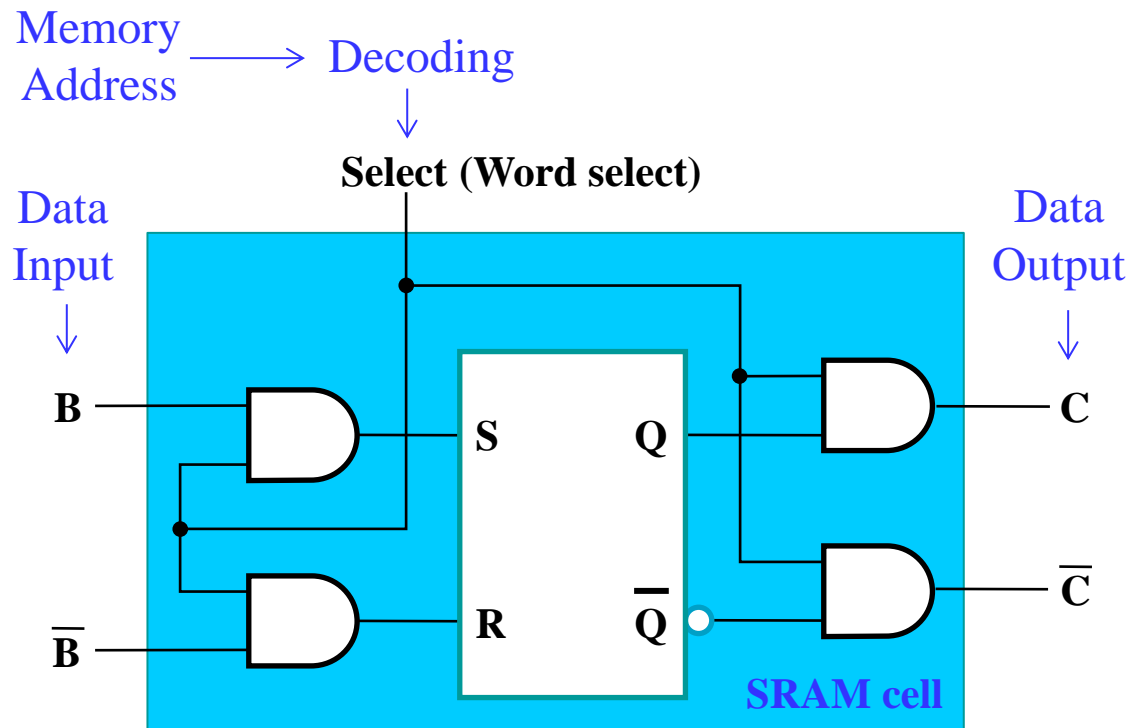


# Static RAM — Cell

- Array of storage cells used to implement static RAM

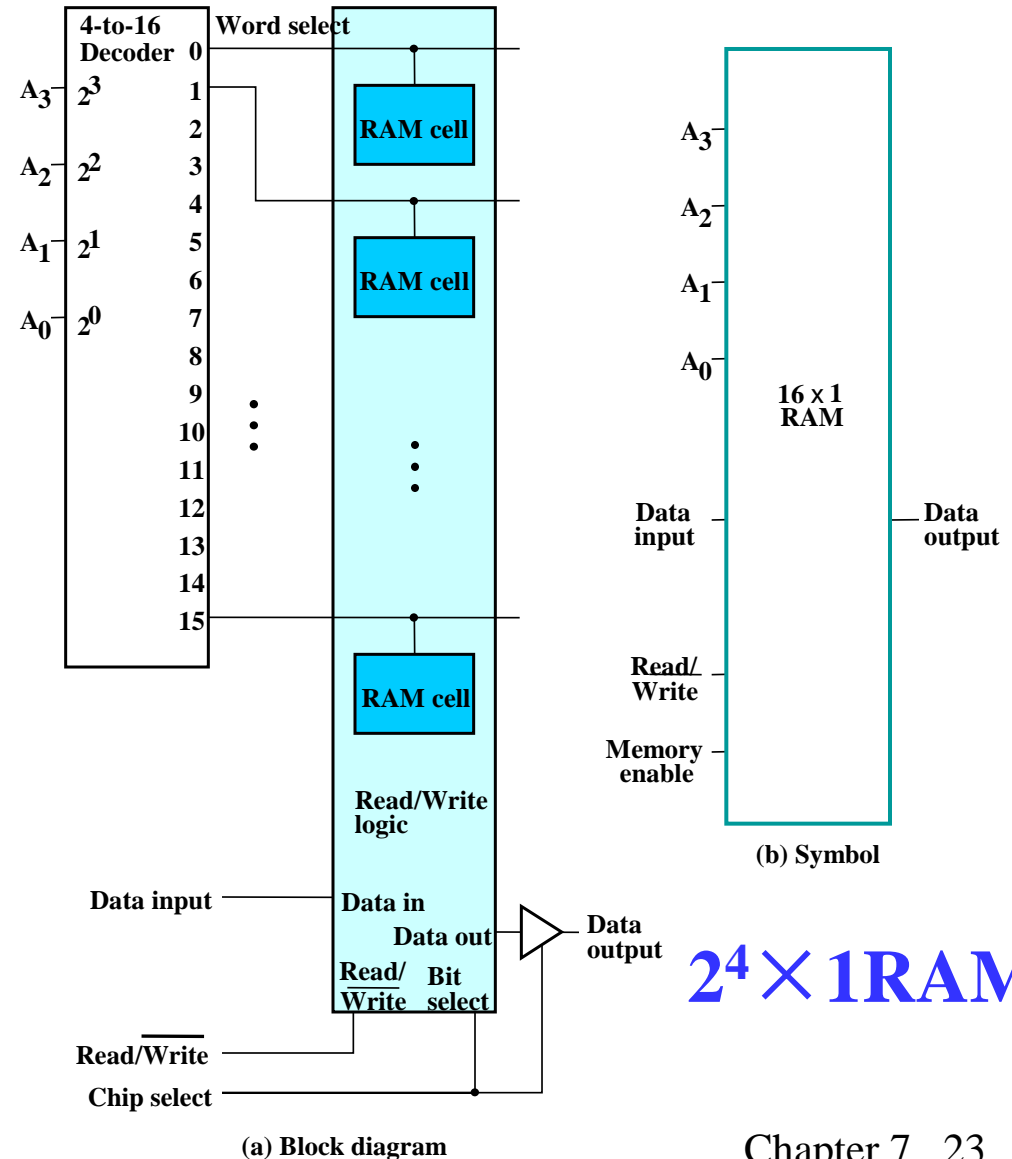
- Storage Cell

- SR Latch
- Select input for control
- Dual Rail Data Inputs B and  $\bar{B}$
- Dual Rail Data Outputs C and  $\bar{C}$



# $2^n$ -Word $\times$ 1-Bit RAM IC

- To build a RAM IC from a RAM slice, we need:
  - Decoder — decodes the  $n$  address lines to  $2^n$  word select lines
  - A 3-state buffer on the data output permits RAM ICs to be combined into a RAM with  $2^n$  words



# Static RAM — Bit Slice

- Represents all circuitry that is required for  $2^n$  1-bit words

- Multiple RAM cells

- Control Lines:

- Word select  $i$   
– one for each word

- Bit Select

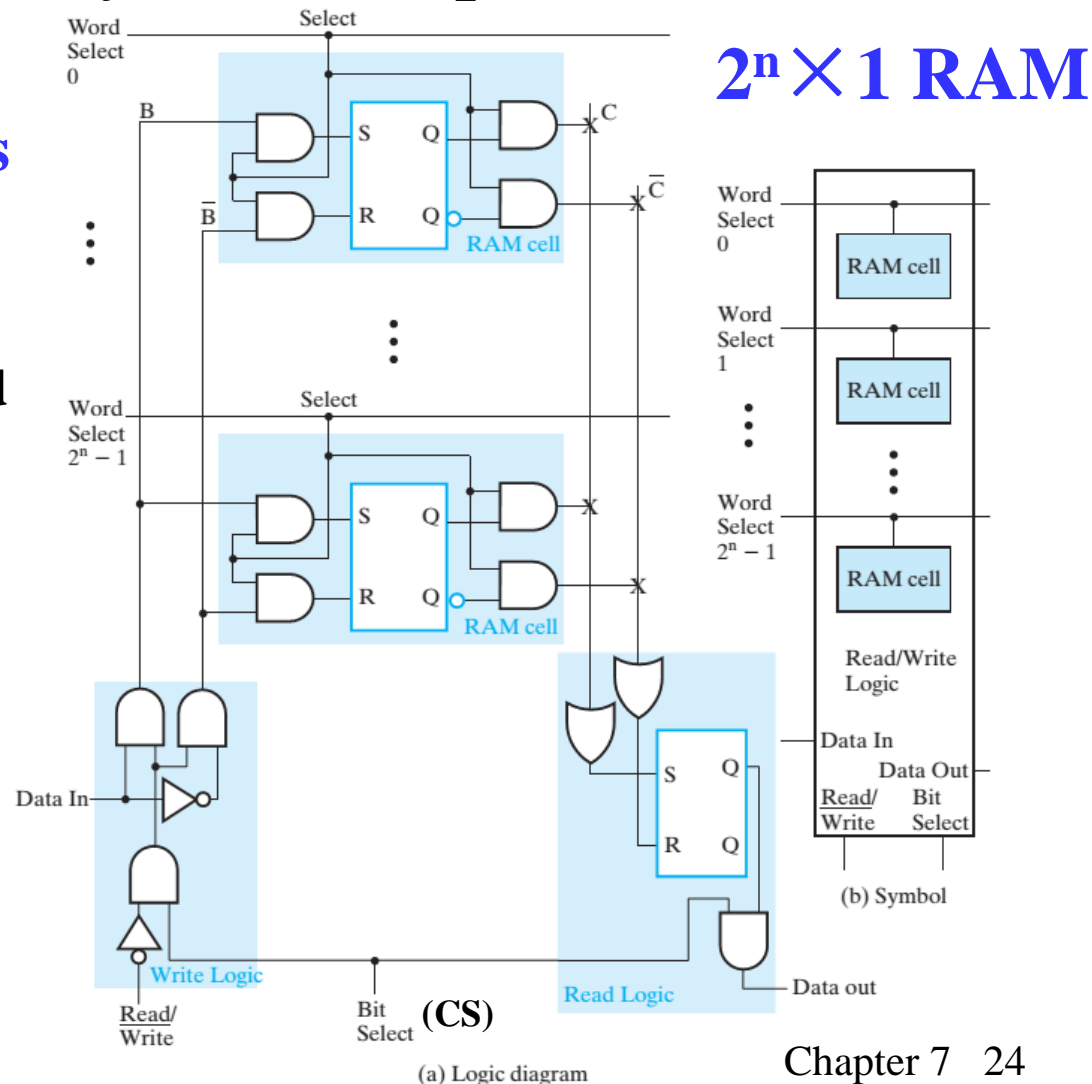
- Read/Write

- Data Lines:

- Data in

- Data out

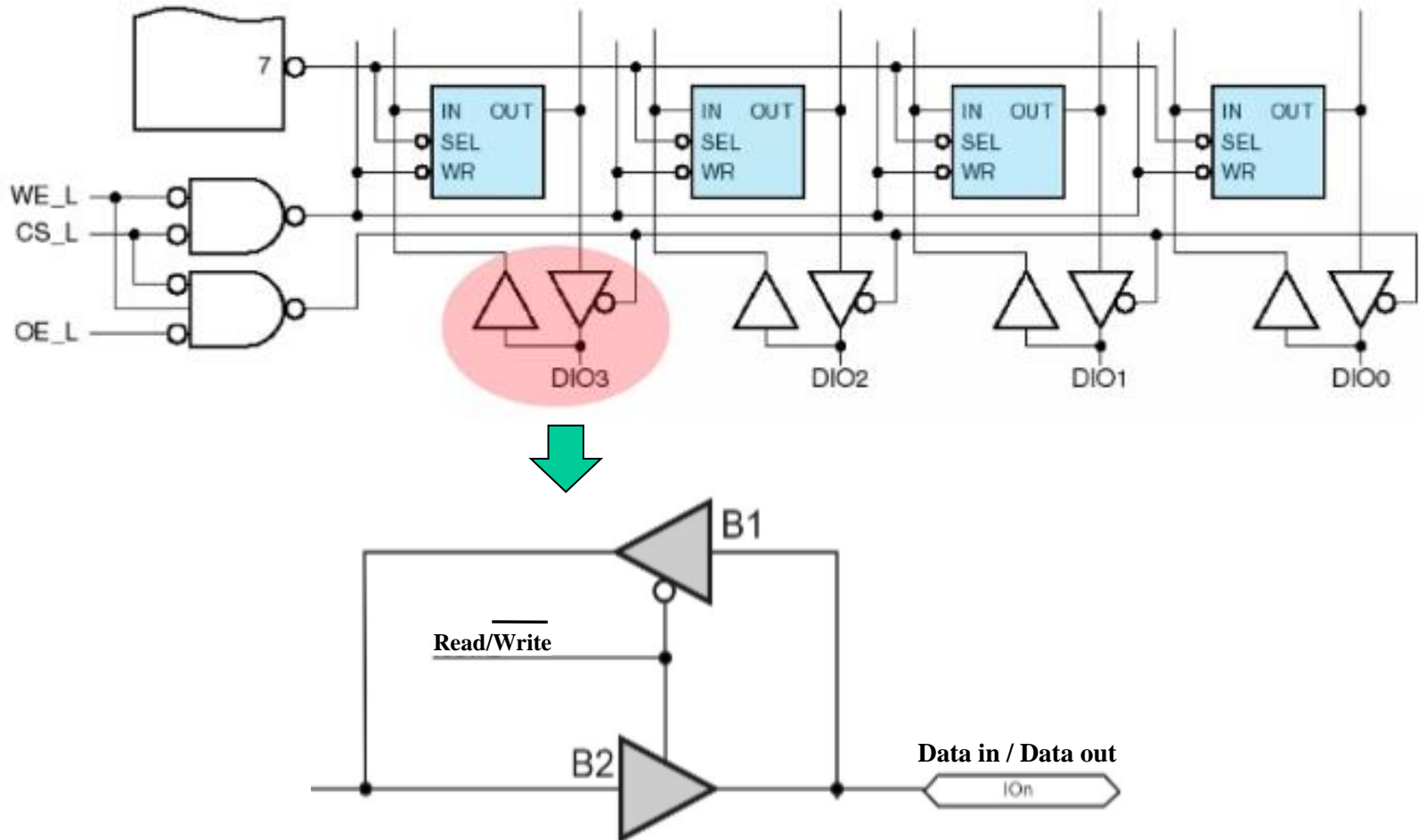
bidirectional pins





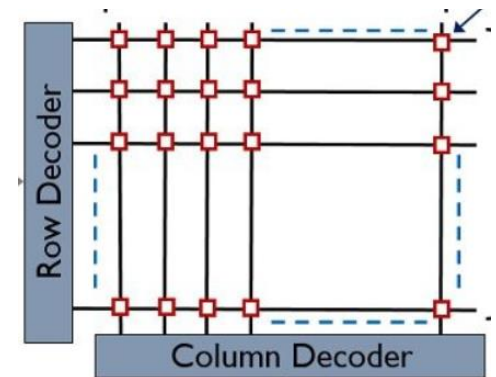
# Bidirectional In/Out Data Pins

- Uses the same data pins for reads and writes



# Cell Arrays and Coincident Selection

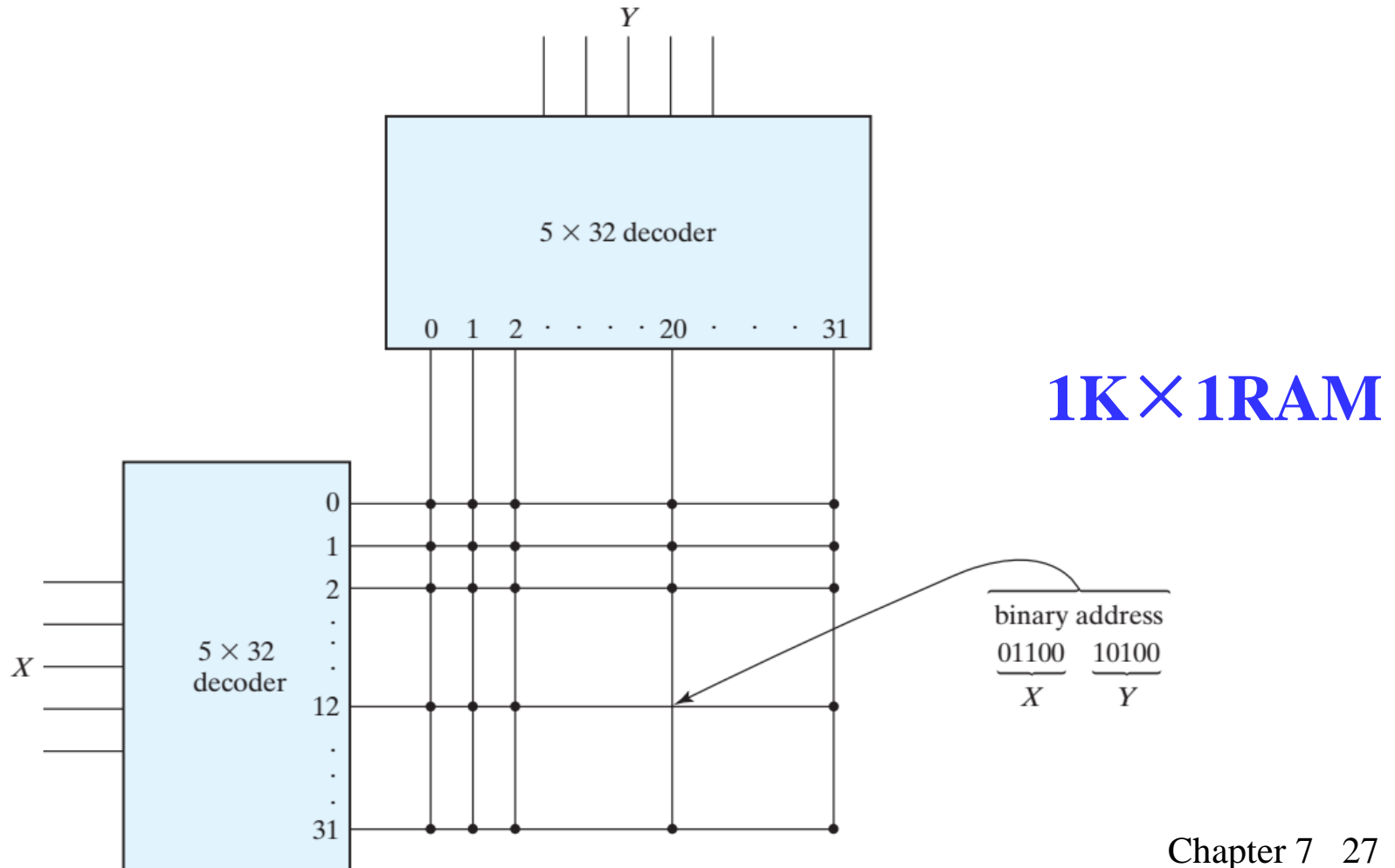
- Memory arrays can be very large =>
  - Large decoders
  - Large fanouts for the bit lines
  - The decoder size and fanouts can be reduced by approximately  $\sqrt{n}$  by using a **coincident selection** in a **2-dimensional array**
  - Uses **two decoders, one for words and one for bits**
  - Word select becomes Row select
  - Bit select becomes Column select
- For 1K\*1 memory, with a single 10-to-1024-line decoder, we need 1,024 AND gates with 10 inputs.



# Cell Arrays and Coincident Selection

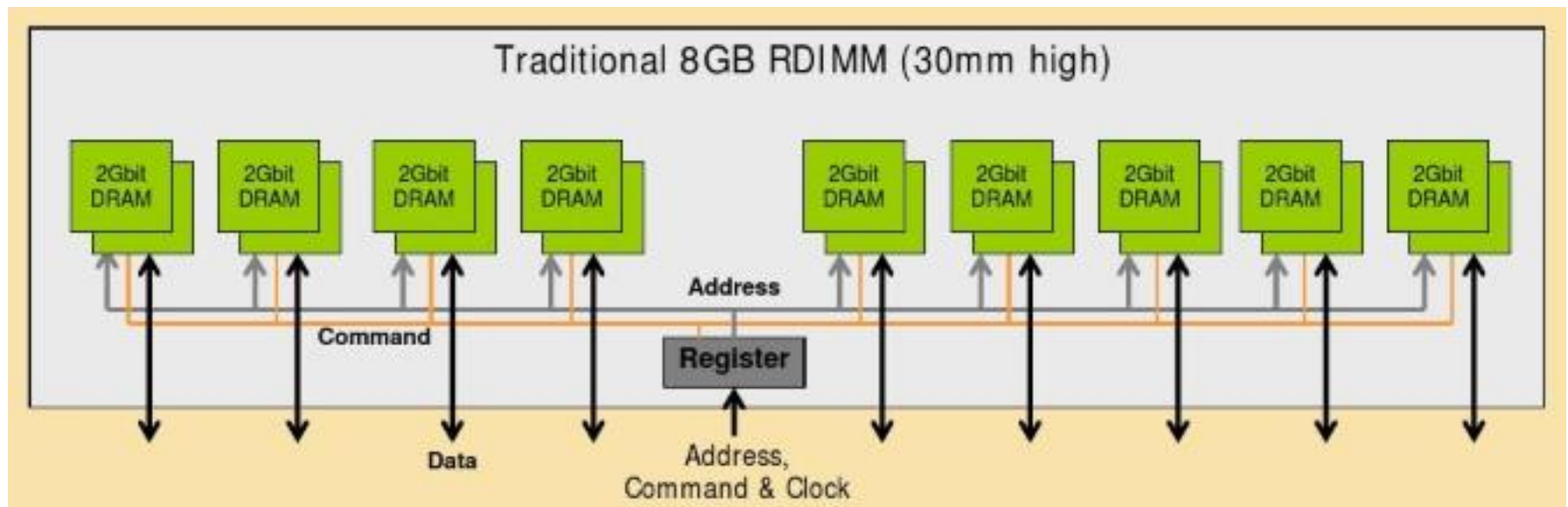
(continued)

- For 1K\*1 memory, with two 5-to-32-line decoders, we need  $32 \times 2 = 64$  AND gates with 5 inputs in each.



# RAM ICs with $> 1$ Bit/Word

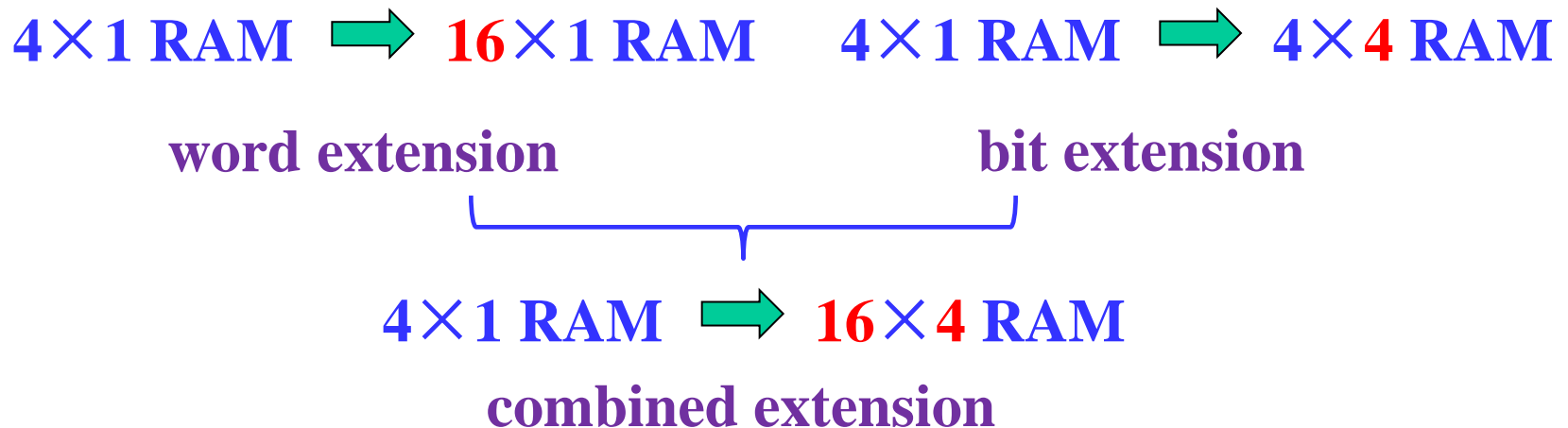
- To better balance the number of words and word length, use ICs with  $> 1$  bit/word



# RAM ICs with $> 1$ Bit/Word

---

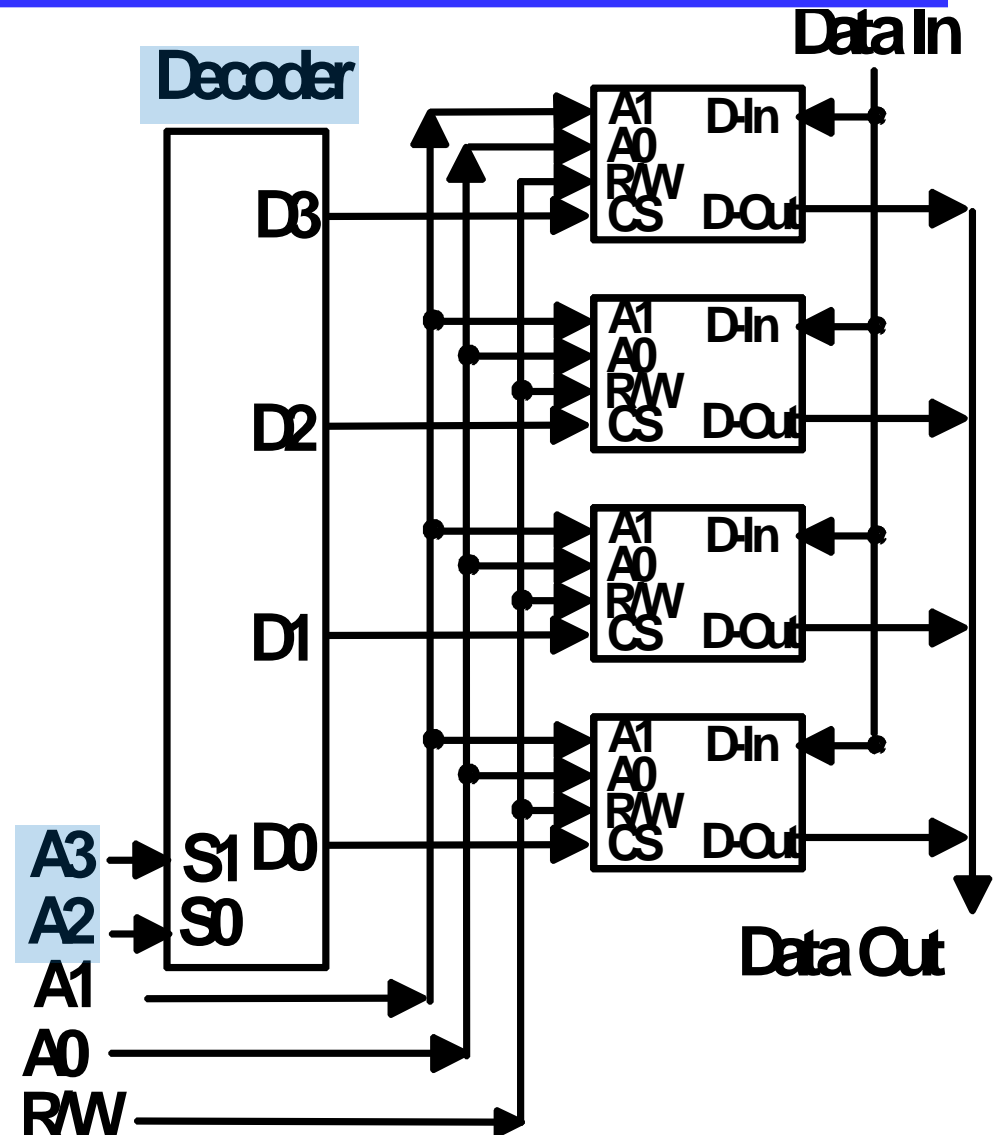
- **Memory expansion method**
  - Address space expansion: **word extension**  
(address bits are increased)
  - Word width expansion: **bit extension**



# Making Larger Memories: Word extension

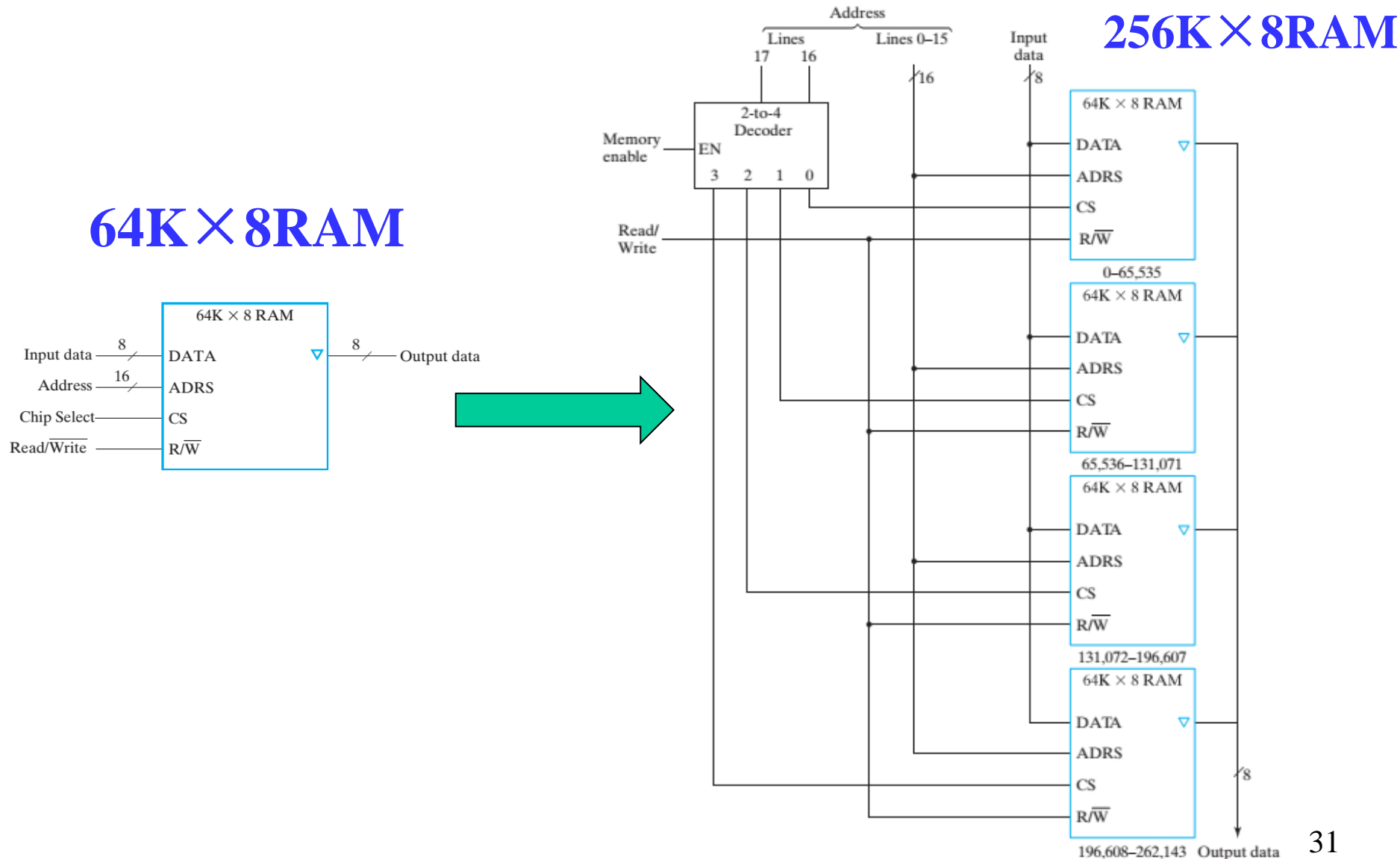
- Using the CS lines, we can make larger memories from smaller ones by tying all address, data, and R/W lines in parallel, and using the decoded higher order address bits to control CS.
- Using the 4-Word by 1-Bit memory from before, we construct a 16-Word by 1-Bit memory.  $\Rightarrow$

$4 \times 1 \text{ RAM} \Rightarrow 16 \times 1 \text{ RAM}$



# Making Larger Memories: Word extension

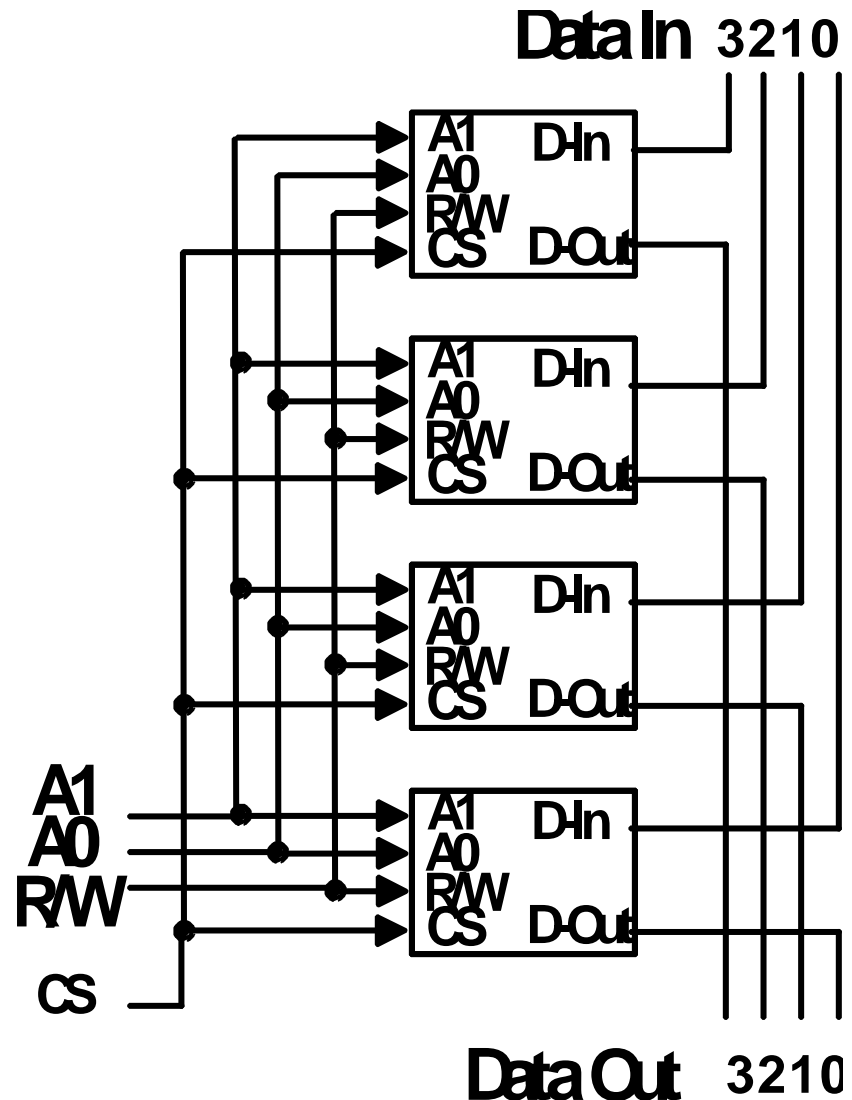
## (continued)



# Making Wider Memories: Bit extension

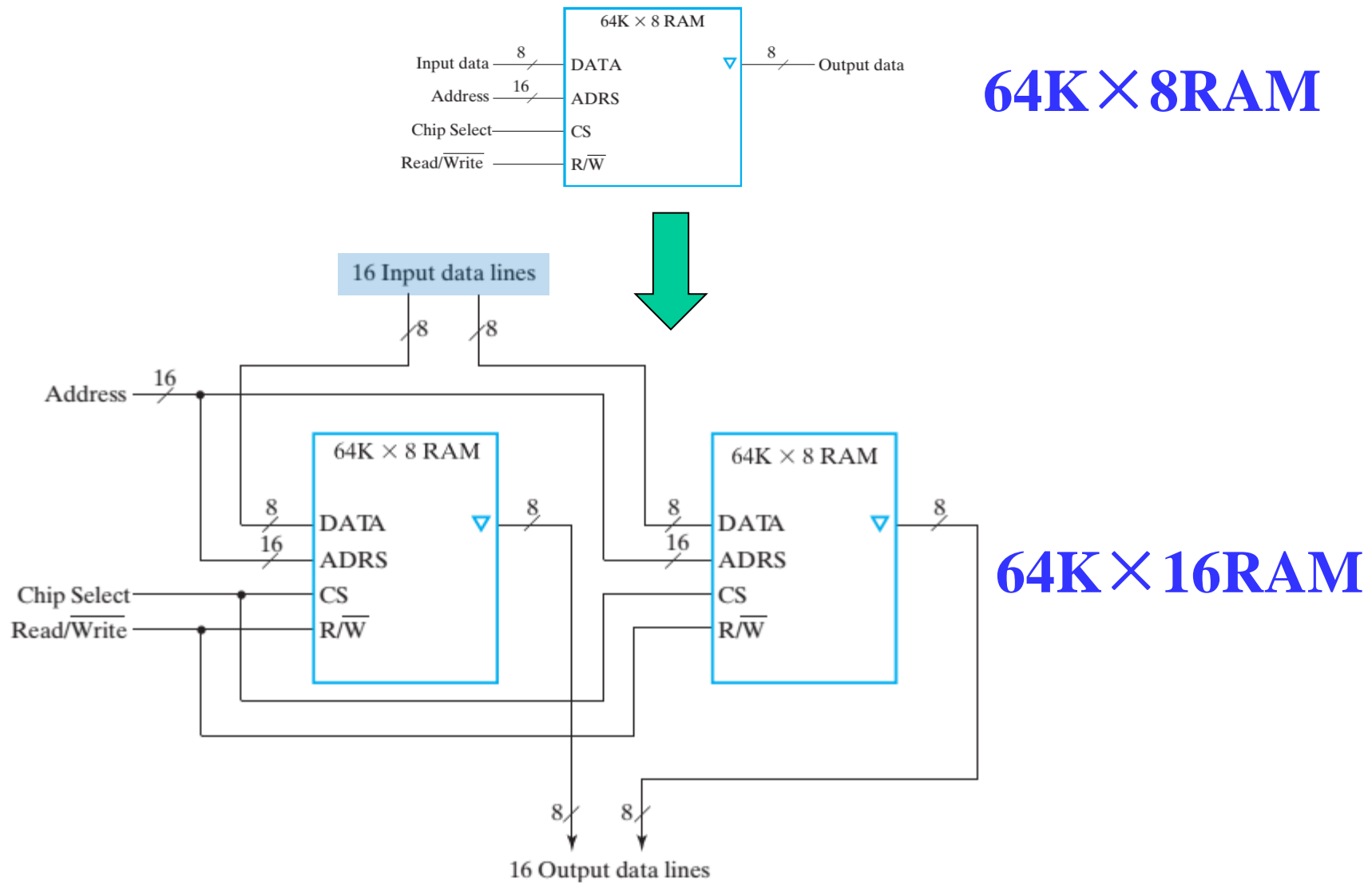
- To construct wider memories from narrow ones, we tie the address and control lines in parallel and keep the data lines separate.
- For example, to make a 4-word by 4-bit memory from 4, 4-word by 1-bit memories  $\Rightarrow$
- Note: Both  $16 \times 1$  and  $4 \times 4$  memories take 4-chips and hold 16 bits of data.

$4 \times 1$  RAM  $\Rightarrow$   $4 \times 4$  RAM





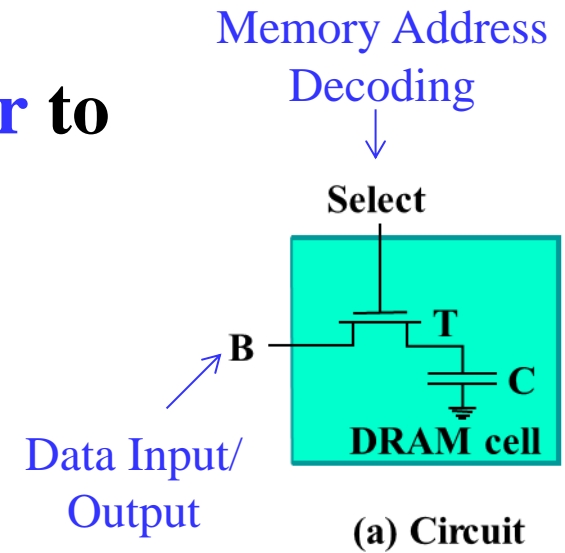
# Making Wider Memories: Bit extension (continued)



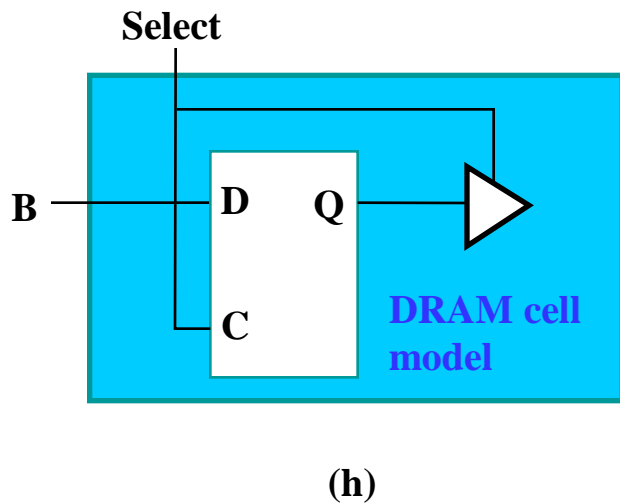
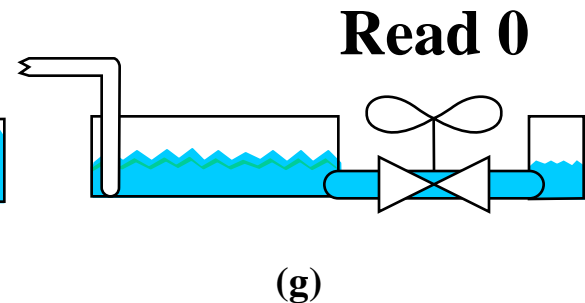
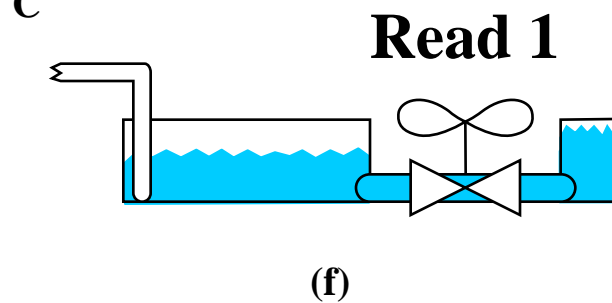
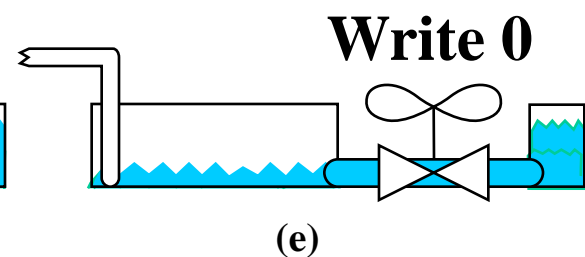
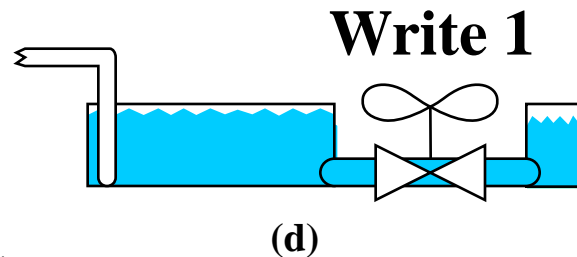
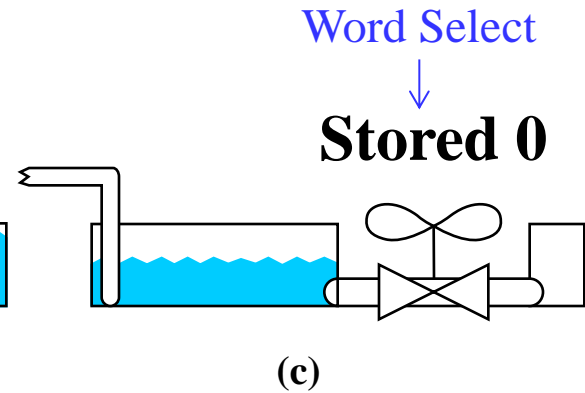
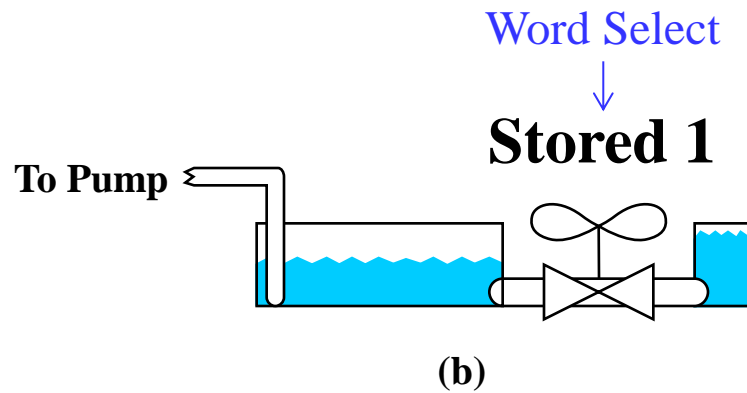
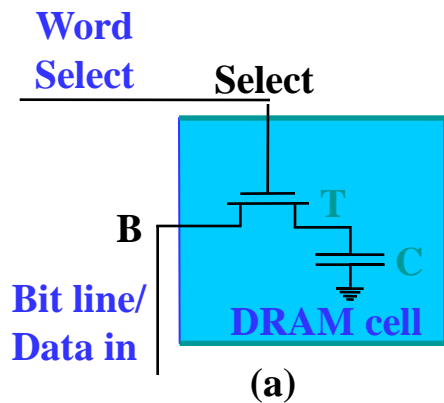
# Dynamic RAM (DRAM)

---

- **Basic Principle: Storage of information on capacitors.**
- **Charge and discharge of capacitor to change stored value**
- **Use of transistor as “switch” to:**
  - Store charge
  - Charge or discharge
- **Each row of a DRAM chip requires refreshing within a specified maximum refresh time**



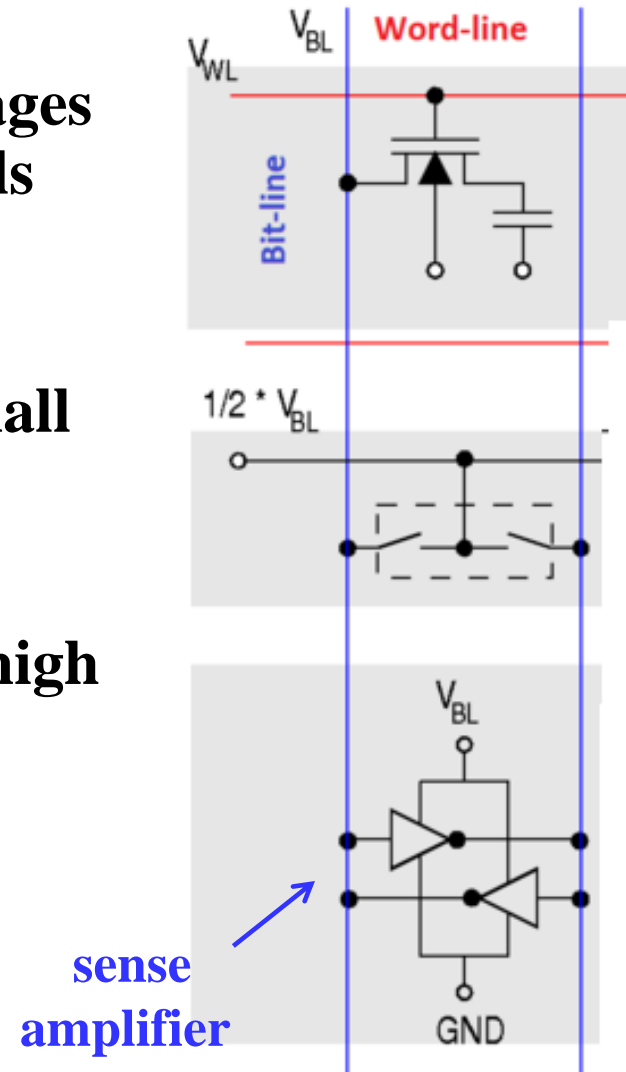
# Dynamic RAM (continued)



hydraulic analogy of cell operation

# Dynamic RAM - Basic Operation

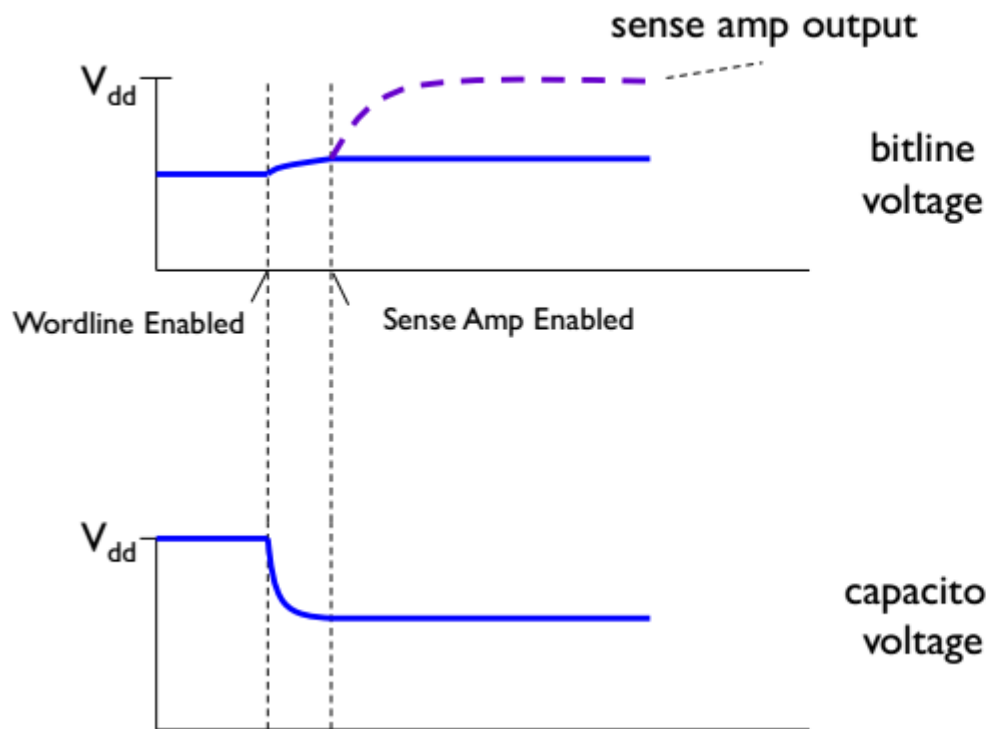
- **Read from a DRAM:**
  - bit-lines are precharged to voltages between high and low logic levels
  - word-line goes high to open the row
  - **sense amplifier** amplifies the small voltage
- **Write to a DRAM**
  - sense amplifier is forced to the high or low voltage state
  - open the row to charge or discharge the cell



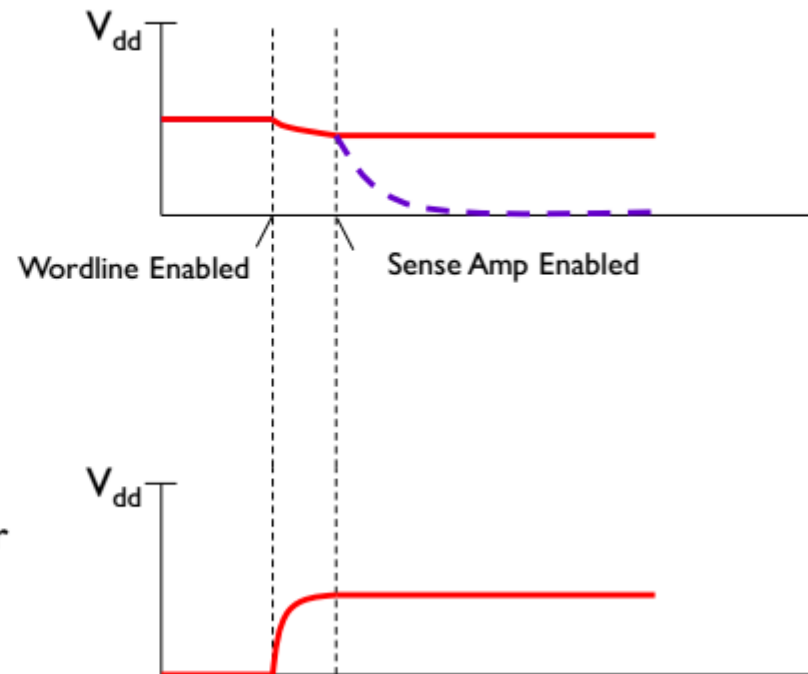
# Dynamic RAM - Destructive Read

- Reads are destructive: contents are erased by reading.

read of 1



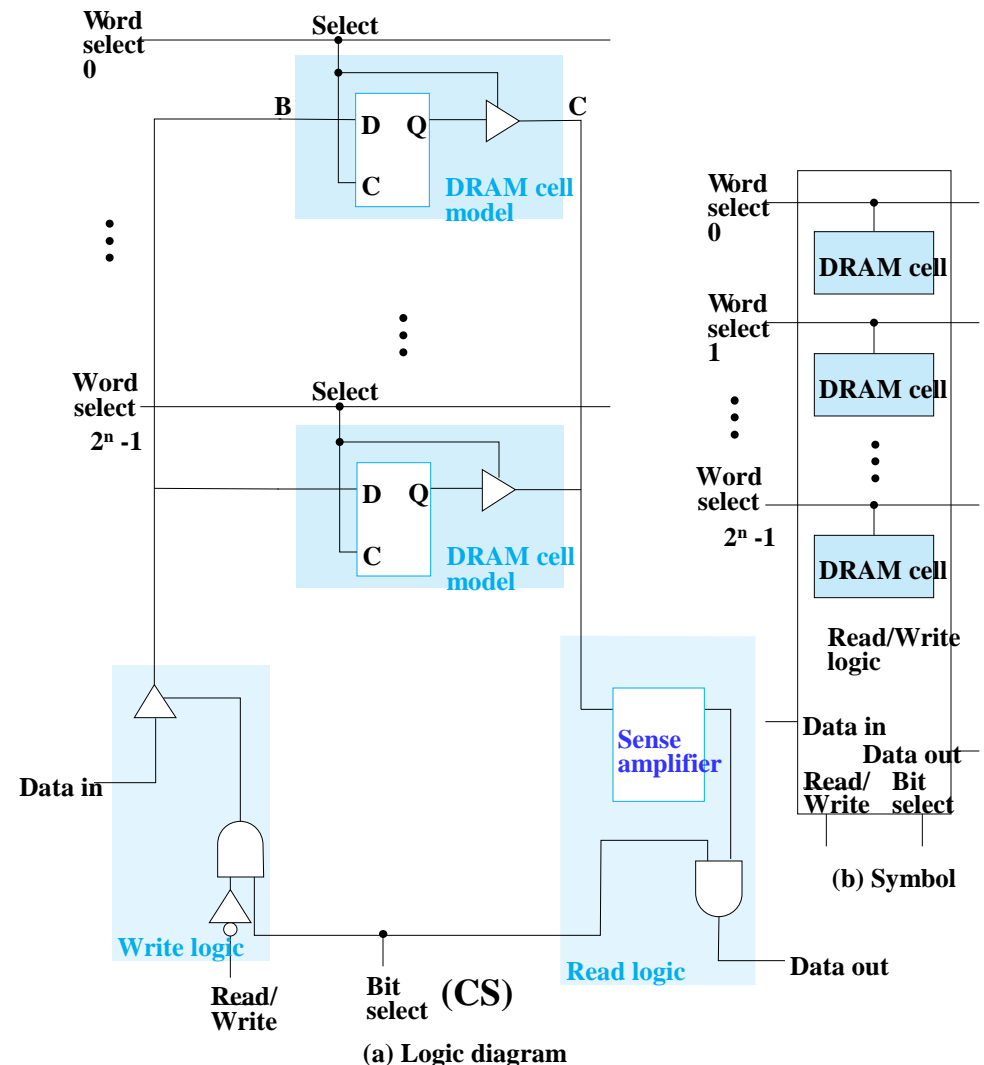
read of 0



- After read of 0 or 1, cell contents close to  $\frac{1}{2}$ .

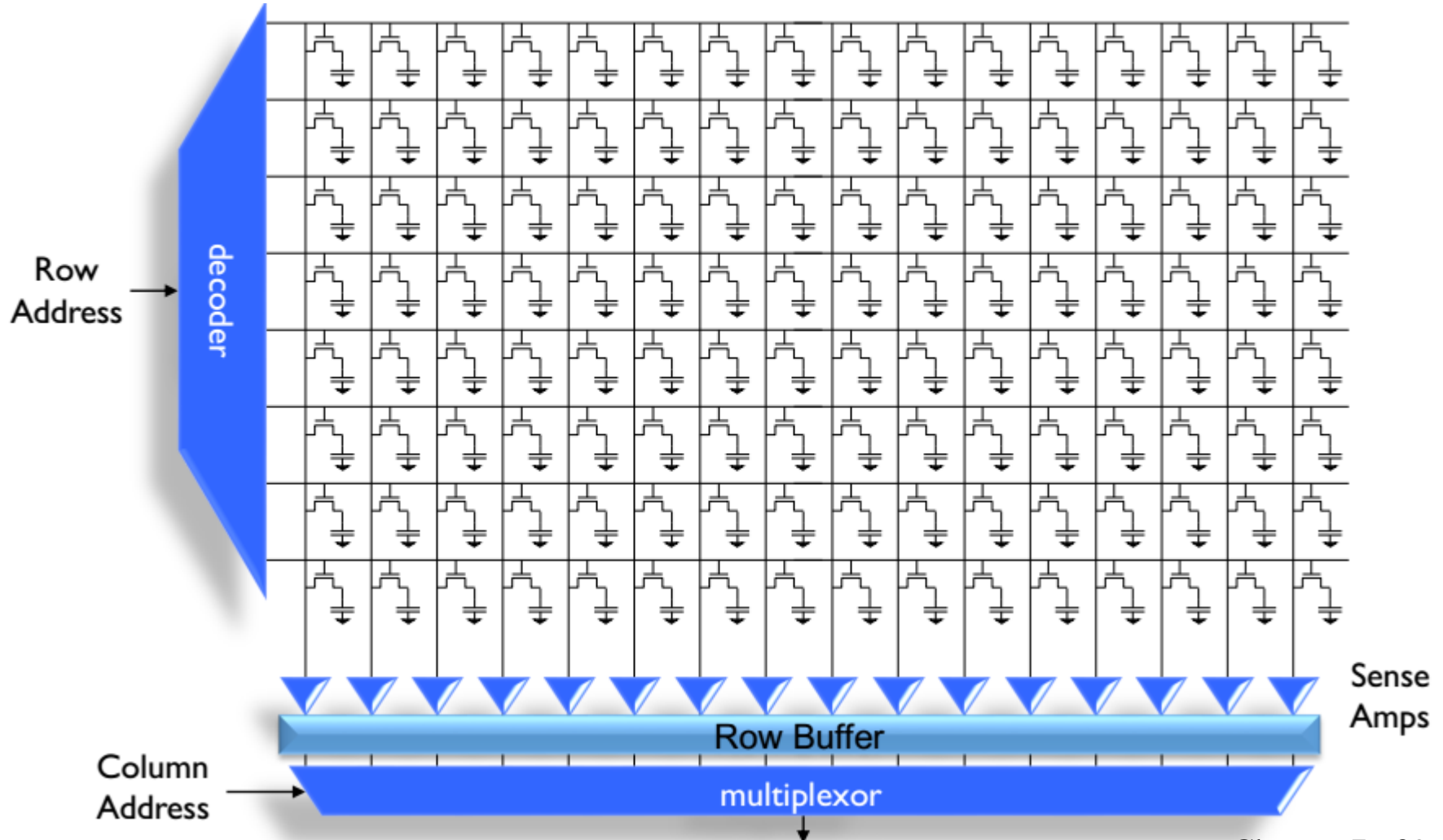
# Dynamic RAM - Bit Slice

- C is driven by 3-state drivers
- **Sense amplifier** is used to change the small voltage change on C into H or L
- In the electronics, B, C, and the sense amplifier output are connected to make destructive read into non-destructive read



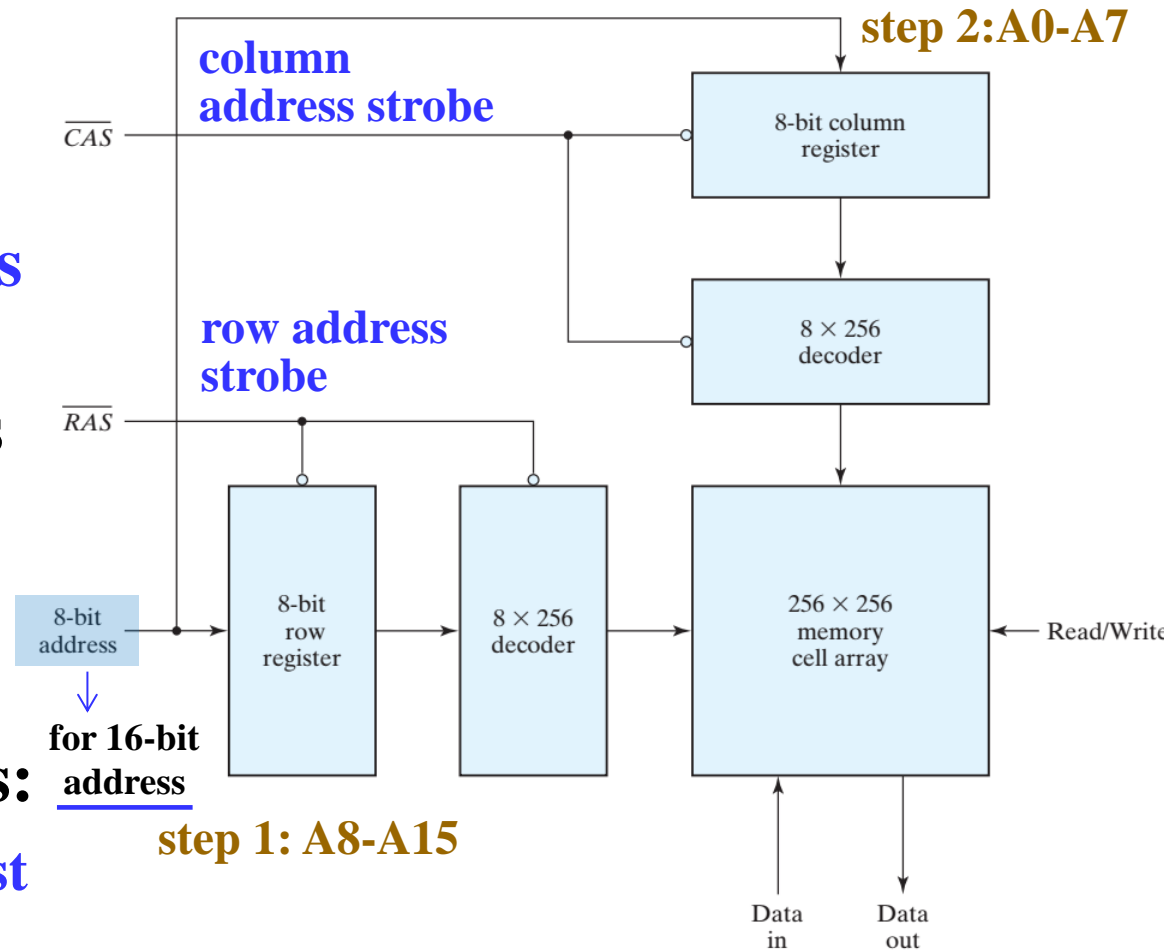
# Dynamic RAM - Address Multiplexing (1/2)

- DRAM is much denser than SRAM.



# Dynamic RAM - Address Multiplexing (2/2)

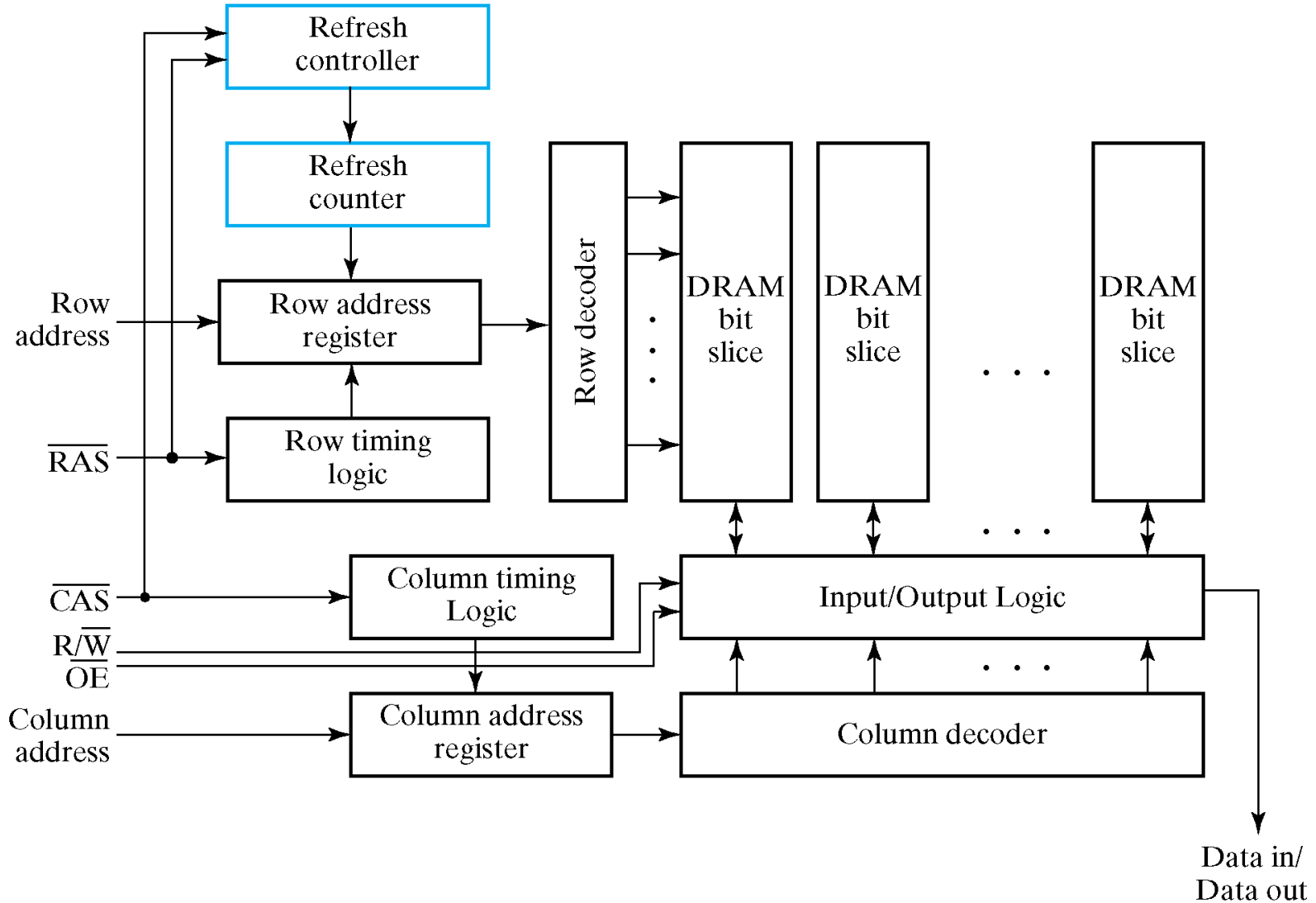
- To reduce the number of pins, DRAMs utilize **multiplexed address** whereby one set of pins accommodates all the address components.
- The address is applied in two steps:
  - the row address first
  - the column address second



**64K x 1 DRAM**

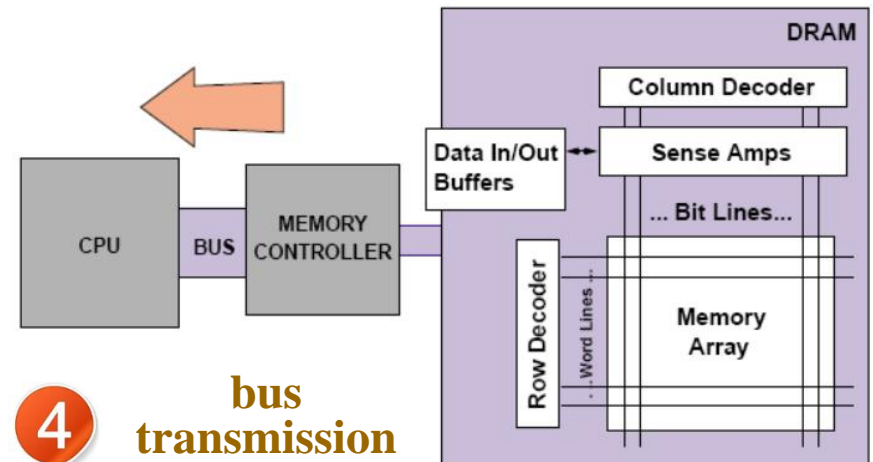
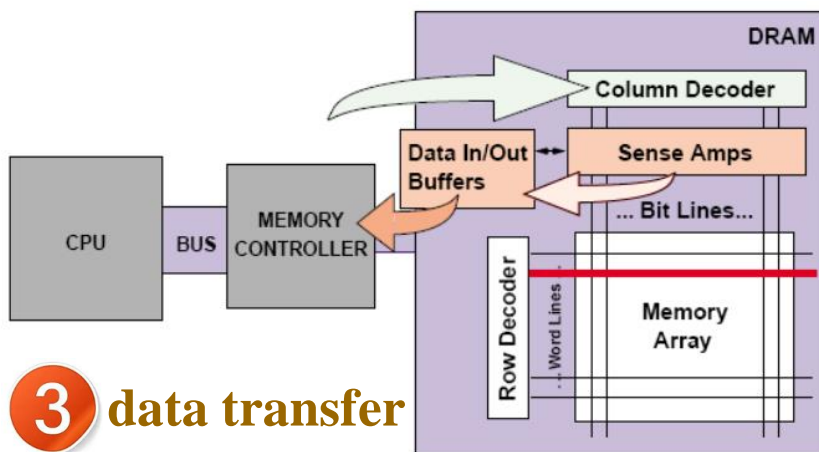
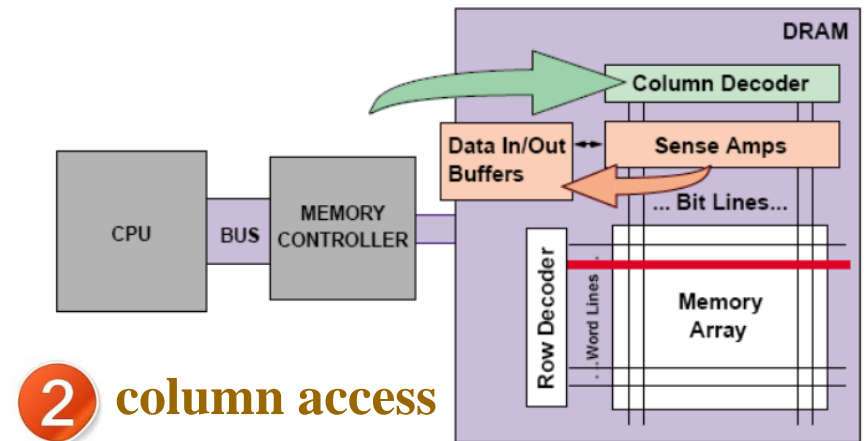
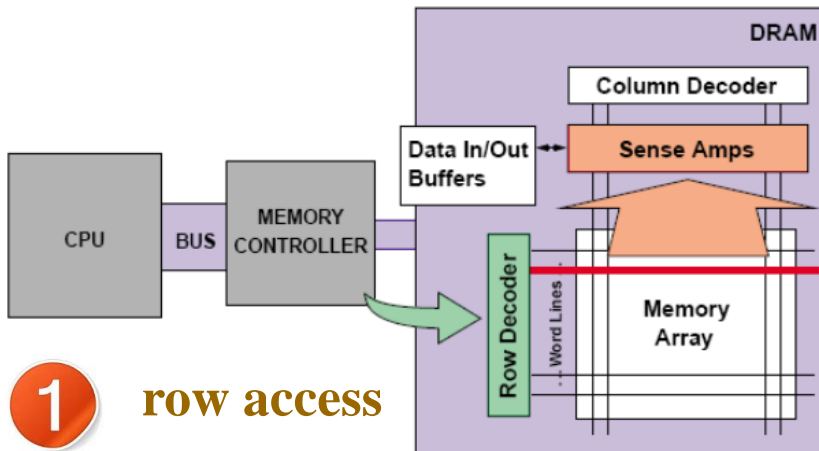


# Dynamic RAM - Block Diagram

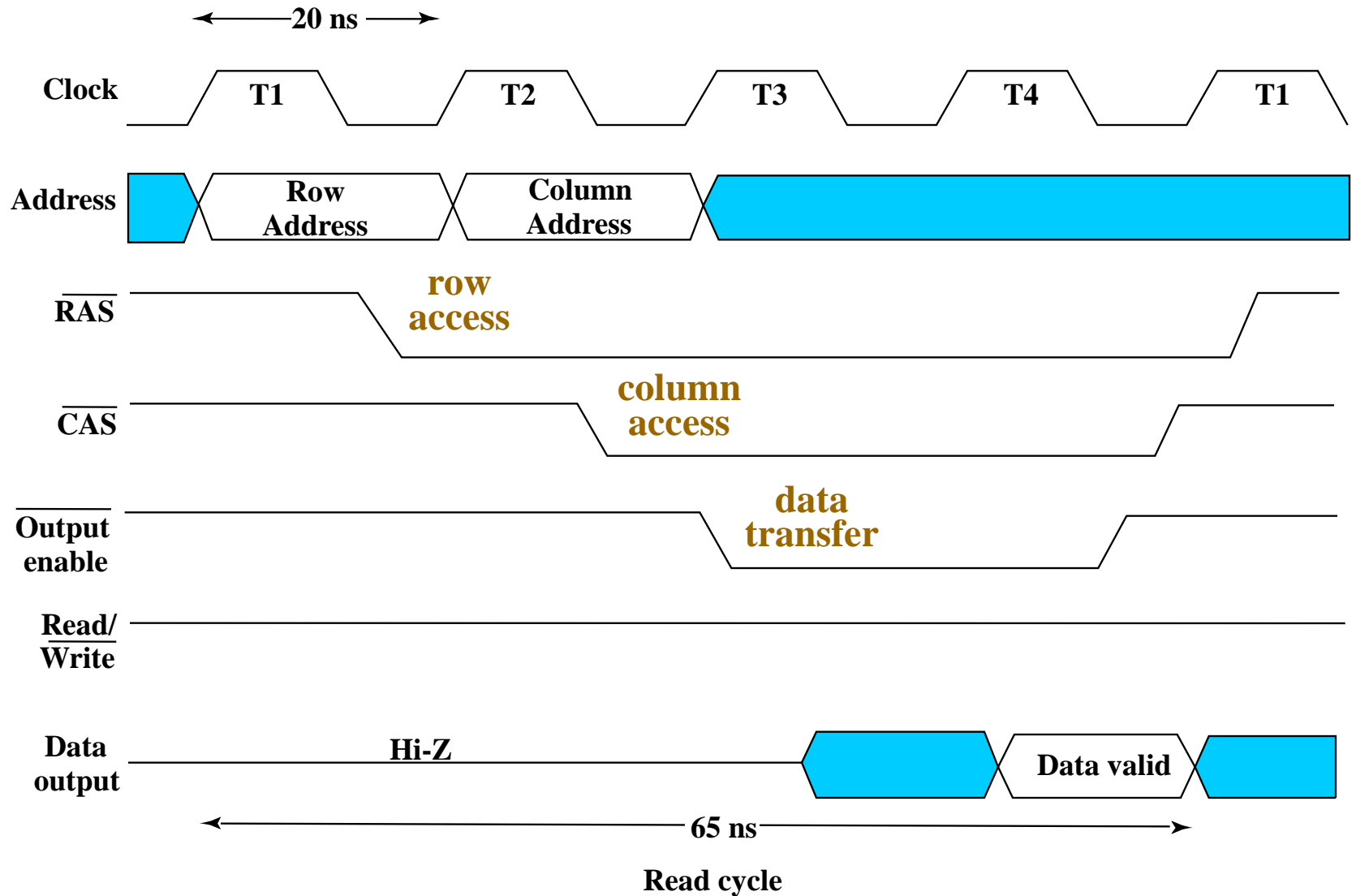


# Dynamic RAM - DRAM Access Sequence

## ■ Why is the row address applied first?

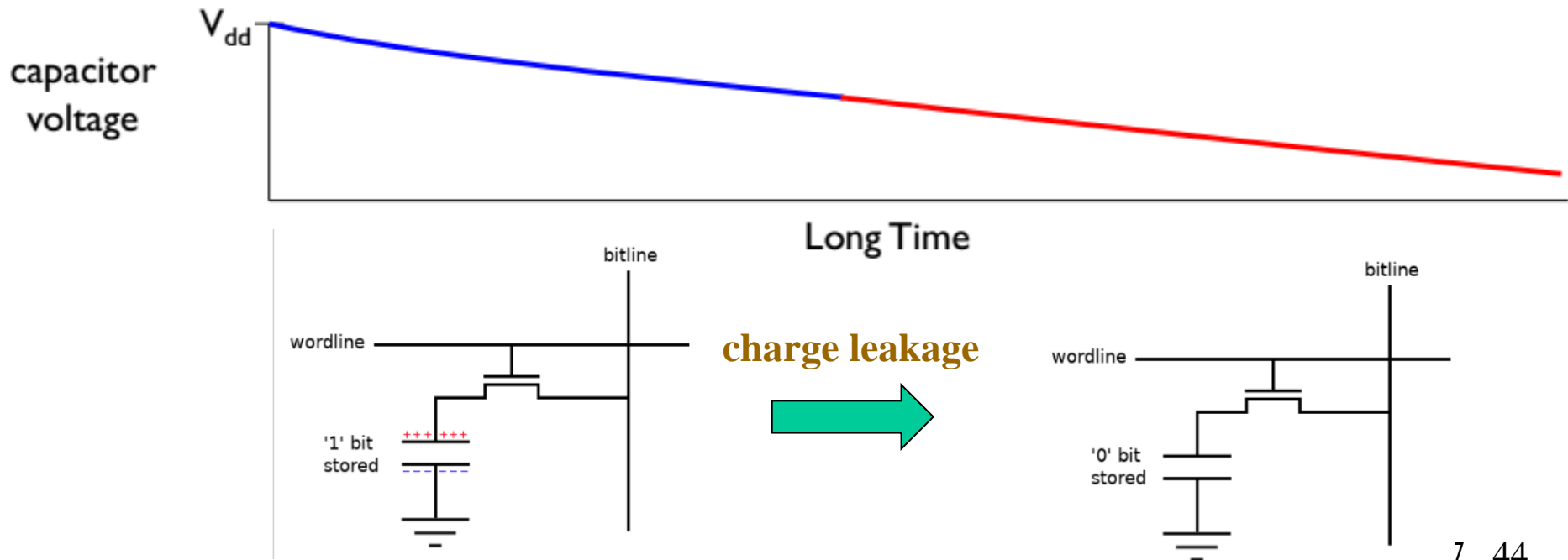


# Dynamic RAM Read Timing



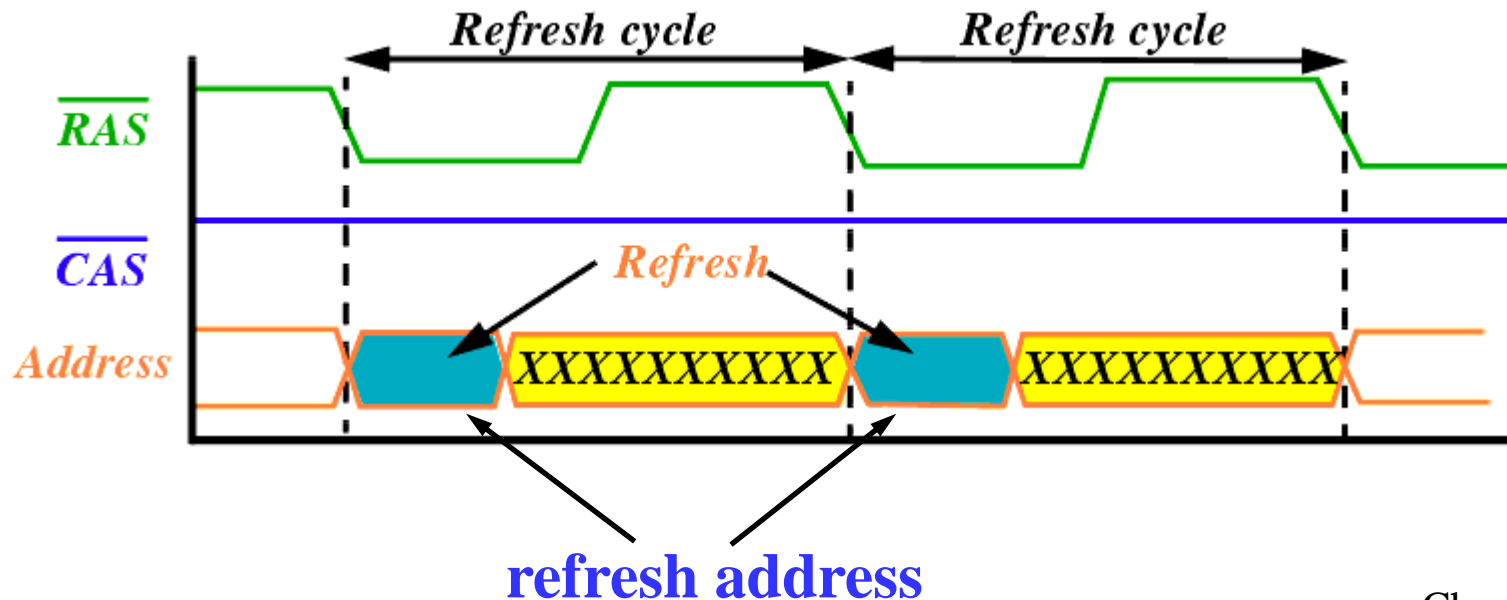
# Dynamic RAM – DRAM Refresh

- Gradually, DRAM cell loses contents
  - Even if it is not accessed
  - This is why DRAM is called “dynamic”
- DRAM must be regularly read and re-written
  - What to do if no read/write to row for long time?



# Dynamic RAM – Refresh Policies

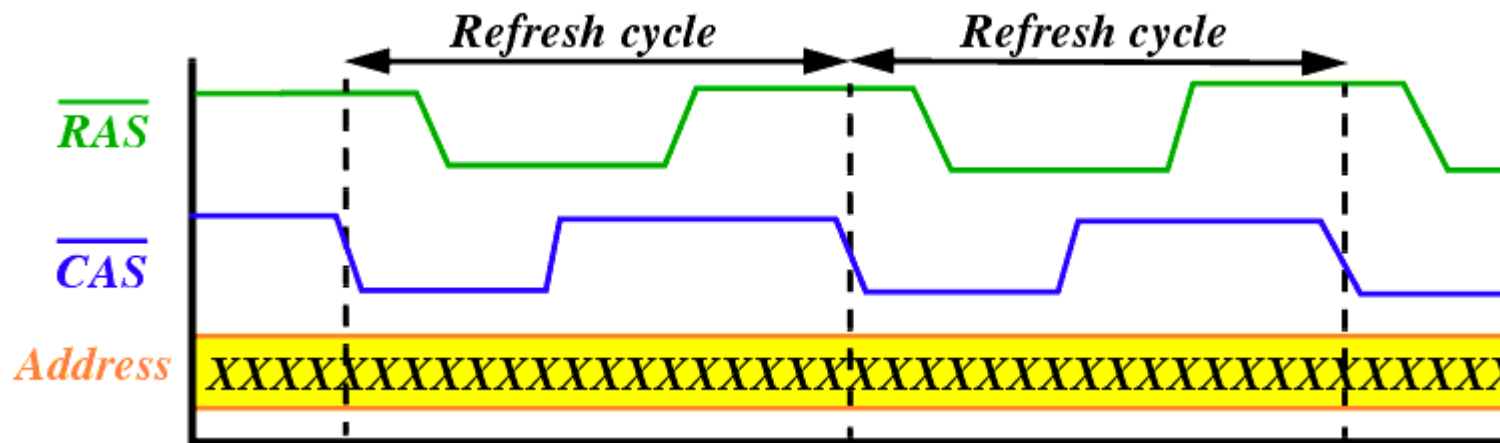
- **RAS-only refresh (row access only without column access)**
  - RAS is activated and a row address (**refresh address**) is applied to the DRAM, CAS inactive.
  - DRAM internally reads one row and amplifies the read data. Not transferred to the output pins as CAS is disabled.
  - The main disadvantage of this refresh method is that an external logic device is required to generate the row addresses in succession.



# Dynamic RAM – Refresh Policies

## ■ CAS before RAS refresh

- DRAM has its own refresh logic with an address counter. When the sequence is applied on CAS and RAS, the internal refresh logic generates an address and refreshes the associated cells.
- After every cycle, the internal address counter is incremented.
- The memory controller just needs to issue the signals.

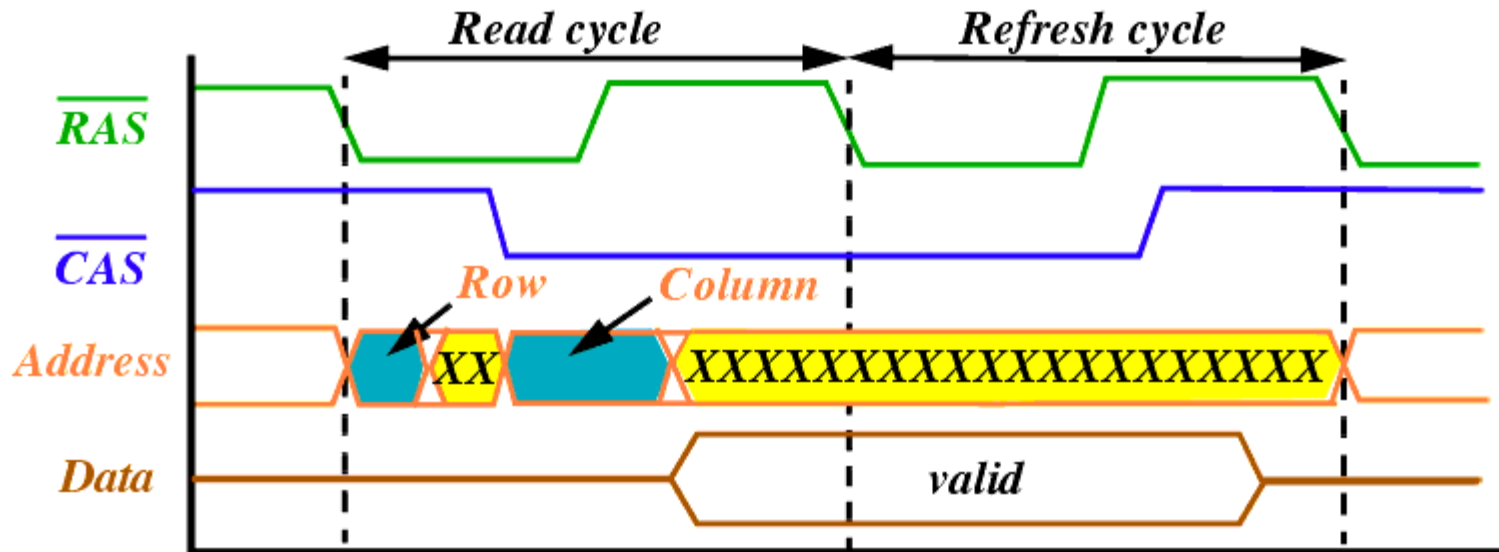


**No address is involved !**

# Dynamic RAM – Refresh Policies

## ■ Hidden refresh

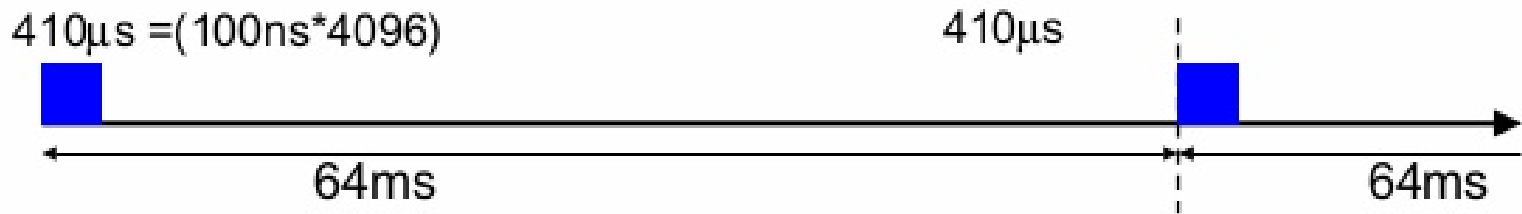
- Following a normal read or write, CAS is left at 0 and RAS is cycled, effectively performing a CAS-before-RAS refresh. During a hidden refresh, the output data from the prior read remains valid. Thus, the refresh is hidden.



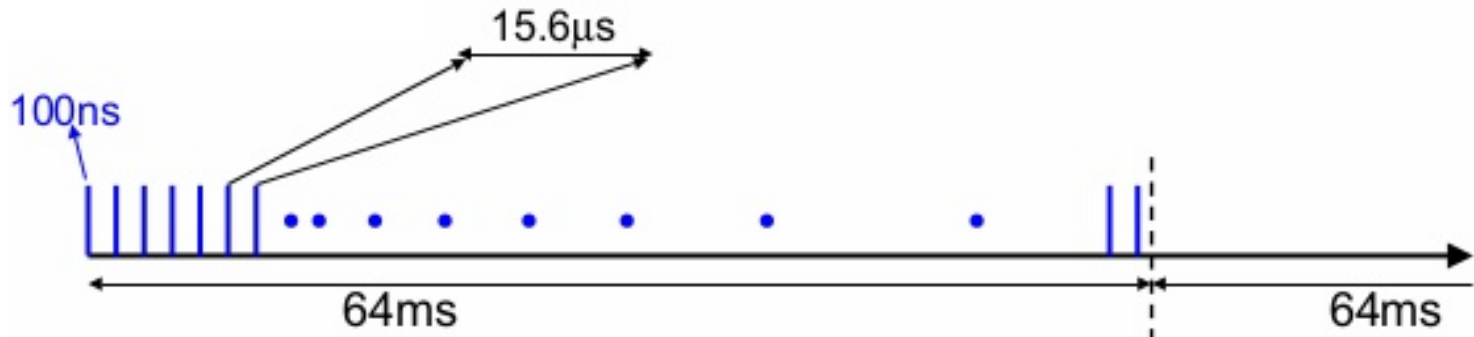
# Dynamic RAM – Refresh (continued)

## ■ Refresh mode

- **Burst mode:** stop the work and refresh all memory (e.g., per 16~64ms)



- **Distributed mode:** space out refresh one row at a time, thus avoid blocking memory for a long time (e.g.,  $15.6\mu s$ )





# DRAM Types

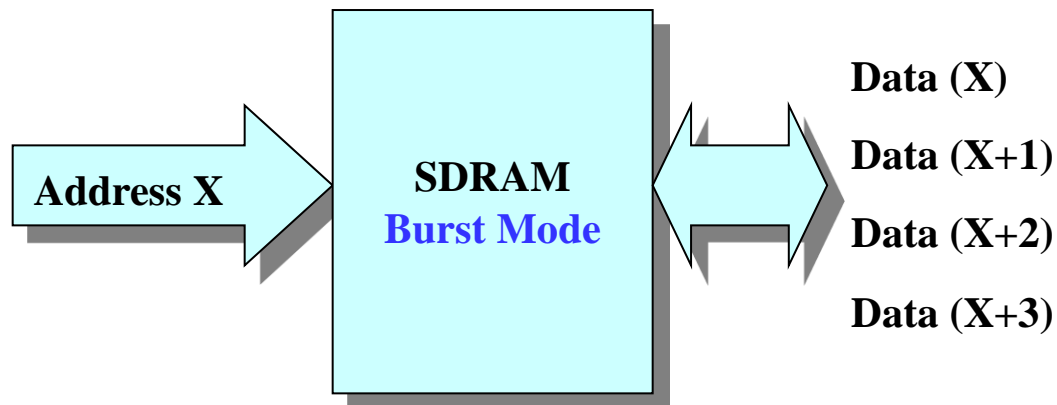
---

- **Types to be discussed**
  - Synchronous DRAM (**SDRAM**)
  - Double Data Rate SDRAM (**DDR SDRAM**)
  - RAMBUS® DRAM (**RDRAM**)
- **Justification for effectiveness of these types**
  - DRAM often used as a part of a memory hierarchy (Detail discussed in Computer Architecture)
  - Reads from DRAM bring data into lower levels of the hierarchy
  - Transfers from DRAM involve multiple consecutively addressed words
  - Many words are internally read within the DRAM ICs using a single row address and captured within the memory
  - This read involves a fairly long delay

# DRAM Types (continued)

---

- **Justification for effectiveness of these types (continued)**
  - These words are then transferred out over the memory data bus using a series of clocked transfers
  - These transfers have a low delay, so several can be done in a short time
  - The column address is captured and used by a synchronous counter within the DRAM to provide consecutive column addresses for the transfers
- *burst read* – the resulting multiple word read from consecutive addresses

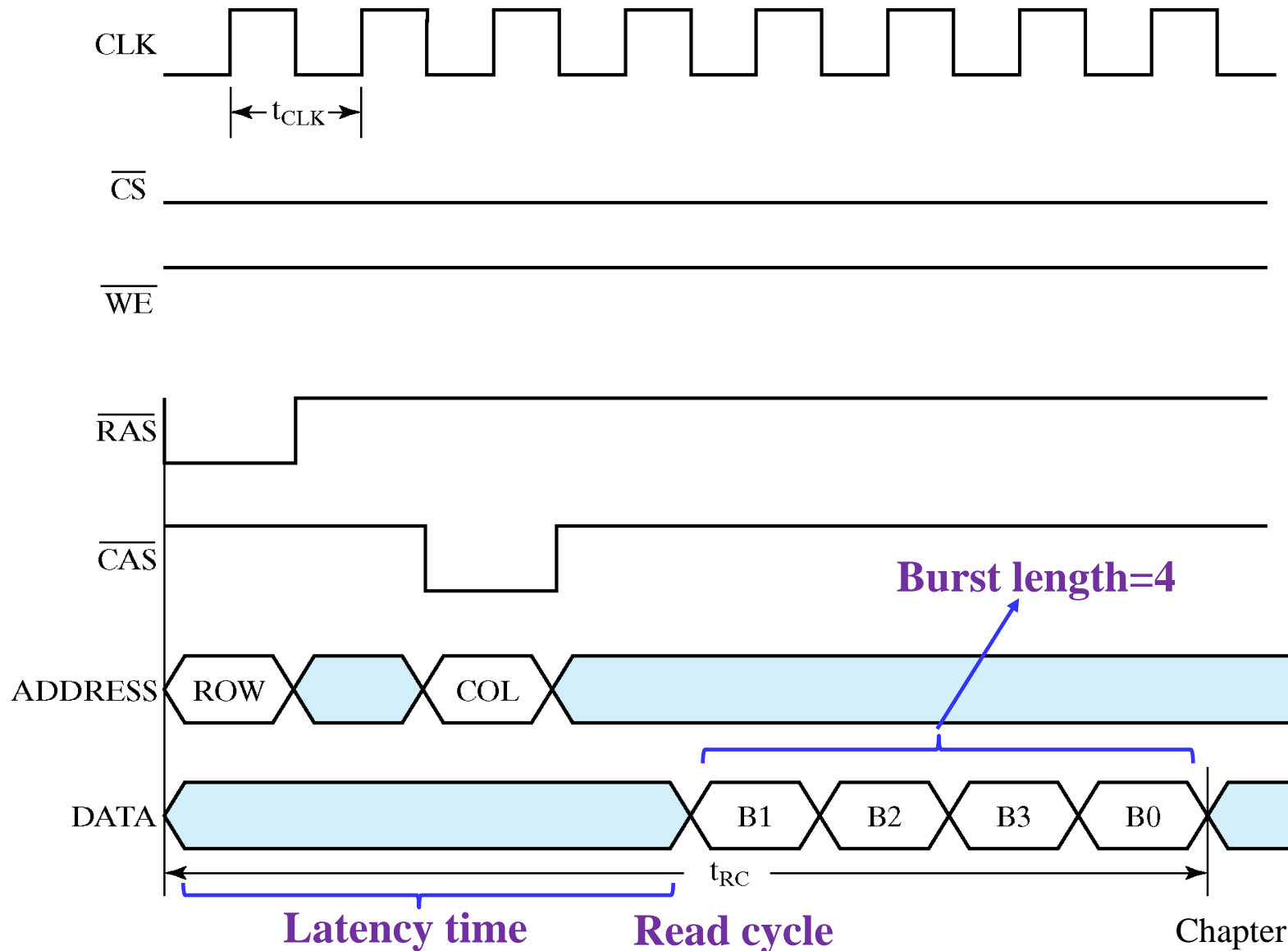


# Synchronous DRAM

---

- All signals are referenced to an **external clock**
  - Makes timing more precise with other system (CPU)
  - Synchronous registers appear on address input, data input and data output
- Burst oriented read and write accesses
  - **Column address counter** is used for addressing internal data to be transferred on each clock cycle
  - Beginning with the column address counts up to column address + burst length – 1
  - **Burst length** is programmable (e.g., 1,2,4,8)
- A user programmable mode register
  - CAS latency, burst length, burst type

# SDRAM burst time——burst length=4



# Synchronous DRAM (Continued)

---

- **Memory bandwidth** is the speed of the memory. It is the maximum rate at which data can be read from or stored into a memory by the processor.
- Memory bandwidth is measured in megabytes per second (MB/s) or gigabytes per second (GB/s).
- The memory bandwidth of SDRAM is related with burst size.

# Synchronous DRAM (Continued)

---

- **For example**
  - **Memory data path width: 1 byte**
  - **Memory clock period: 7.5 ns**
  - **Latency time (from application of row address until first word available): 4 clock cycles**
  
- **If burst size: 8 bytes**
  - **Read cycle time:  $(4 + 8) * 7.5 \text{ ns} = 90 \text{ ns}$**
  - **Memory Bandwidth:  $8 / (90 * 10^{-9}) = 88.89 \text{ Mbytes/sec}$**
  
- **If burst size: 2048 bytes**
  - **Read cycle time:  $(4 + 2048) * 7.5 \text{ ns} = 15390 \text{ ns}$**
  - **Memory Bandwidth:  $2048 / (15390 * 10^{-9}) = 133.07 \text{ Mbytes/sec}$**

# Synchronous DRAM (Continued)

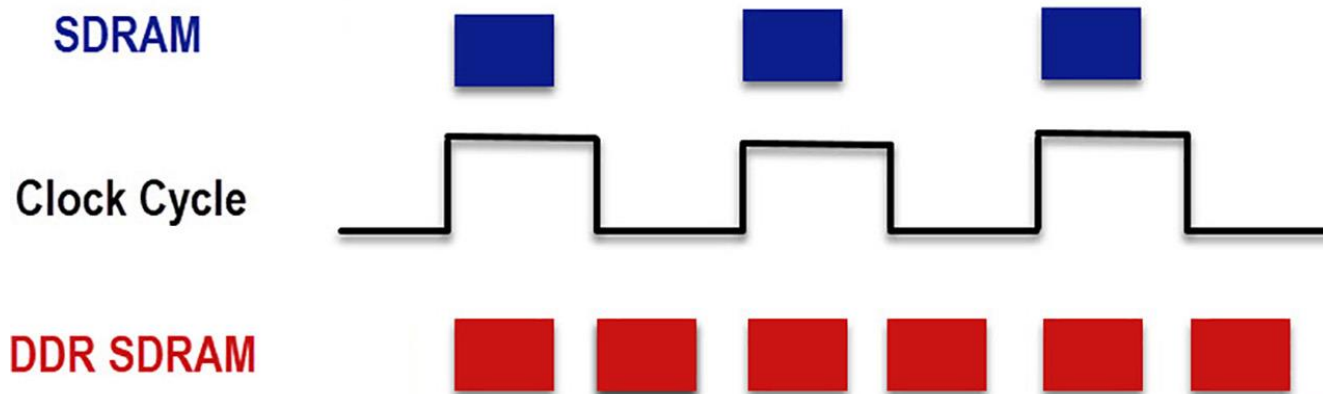
---

## ■ Example

- **Memory data path width: 1 word = 4 bytes**
- **Burst size: 8 words = 32 bytes**
- **Memory clock frequency: 5 ns**
- **Latency time: 4 clock cycles**
- **Read cycle time:  $(4 + 8) * 5 \text{ ns} = 60 \text{ ns}$**
- **Memory Bandwidth:  $32 / (60 * 10^{-9}) = 533 \text{ Mbytes/sec}$**

# Double Data Rate Synchronous DRAM

- Transfers data on **both edges of the clock**
- Provides a transfer rate of 2 data words per clock cycle

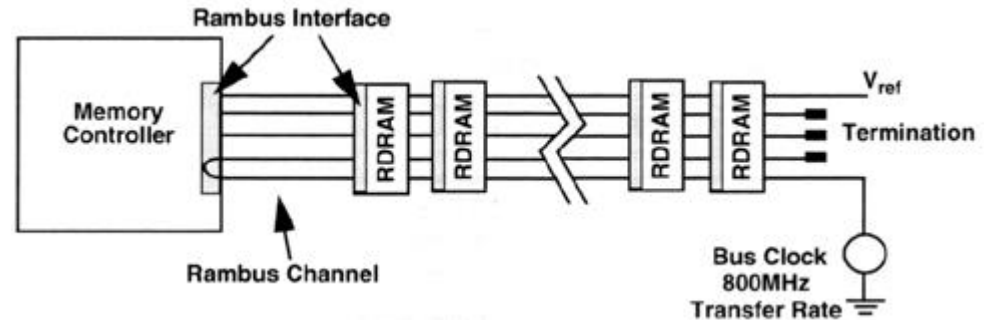


- **Example: Same as for synchronous DRAM**
  - Read cycle time = 60 ns
  - Burst size: 8 — **8 \* 2 words** =  $8 * 4 * 2 = 64$  bytes
  - Memory Bandwidth: **64** /  $(60 * 10^{-9}) = 1.066$  Gbytes/sec



# RAMBUS DRAM (RDRAM)

- Uses a **packet-based bus** for interaction between the RDRAM ICs and the memory bus to the processor
- The bus consists of:
  - A 3-bit row address bus
  - A 5-bit column address bus
  - A 16 or 18-bit (for error correction) data bus
- The bus is synchronous and transfers on both edges of the clock
- Packets are 4-clock cycles long giving 8 transfers per packet representing:
  - A 12-bit row address packet
  - A 20-bit column address packet
  - A 128 or 144-bit data packet
- Multiple memory banks are used to permit concurrent memory accesses with different row addresses
- The electronic design is sophisticated permitting very fast clock speeds

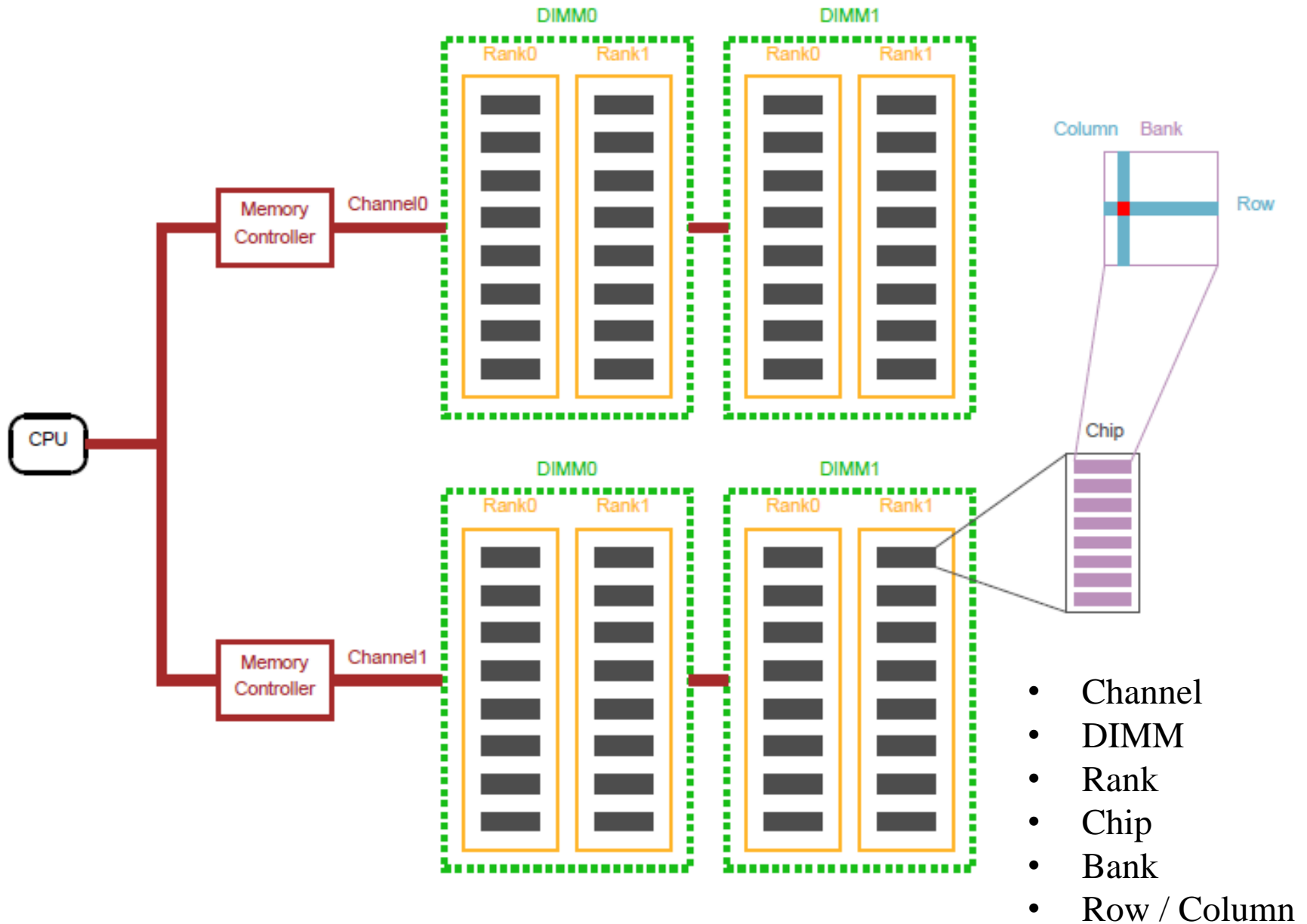


# Arrays of DRAM Integrated Circuits

---

- Similar to arrays of SRAM ICs, but there are differences typically handled by an IC called a *DRAM controller*:
  - Separation of the address into row address and column address and timing their application
  - Providing  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  and timing their application
  - Performing refresh operations at required intervals
  - Providing status signals to the rest of the system (e.g., indicating whether or not the memory is active or is busy performing refresh)

# DRAM Organization



# Assignments

---

## Reading:

- 7.1-7.7

## Problem assignment:

- 7-1、 7-4、 7-5、 7-8

# Appendix A: Roofline Performance Model (1/7)

---

- We hope to attain peak performance (FLOP/s), however, finite locality (reuse) and memory bandwidth limit performance.
- The **Roofline model** is a performance model seeking to give the limitations of a specific hardware in terms of algorithm implementation.

Williams, Samuel, Andrew Waterman, and David Patterson. **Roofline: an insightful visual performance model for multicore architectures.**  
*Communications of the ACM*, 2009

# Roofline Performance Model (2/7)

---

- **Performance Objective (Flops/s)**
  - **Flops/s** is a measurement standing for floating point operations per second.
  - This is the number of mathematical operations (i.e. +, \*, ...) that a given algorithm does per second.
- E.g., the number of Flops for the **daxpy** loop ( $y = ax + y$ )

```
void daxpy(double *x, double *y, double a, int n)
{
    int i;
    for (i = 0; i < n; i++)
        y[i] = a*x[i] + y[i];
}
```

 **Flops # =  $2 \times n = 2n$**

# Roofline Performance Model (3/7)

- **Memory Traffic (Bytes/s)**

- **Bytes/s** is the access speed of the memory at which data can be loaded or stored.

- **E.g., the number of memory traffic (Bytes) for the **daxpy** loop ( $y = ax + y$ )**

```
void daxpy(double *x, double *y, double a, int n)
{
```

```
    int i;
```

```
    for (i = 0; i < n; i++)
```

```
        y[i] = a*x[i] + y[i];
```

```
}
```

↑  
1 store

↖ ↗  
2 loads



**Bytes #** = ((2 loads)(8 bytes/load)  
+ (1 store)(8 bytes/store))  
× n = 24n

# Roofline Performance Model (4/7)

## ■ Arithmetic Intensity (AI)

- AI is the measure of how many operations are done per bytes loaded or stored from memory.

$$\text{AI} = \frac{\text{CPU performance}}{\text{Memory Traffic}} = \frac{\text{Flops/s}}{\text{Bytes/s}} = \frac{\text{Flops \#}}{\text{Bytes \#}}$$

- A **high AI value** indicates a lot of computation per load, while a **low AI value** means the algorithm is often waiting on data to be loaded before it can do anything.

## ■ E.g., the AI value for the **daxpy** loop ( $y = ax + y$ )

```
void daxpy(double *x, double *y, double a, int n)
```

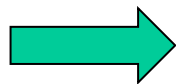
```
{
```

```
    int i;
```

```
    for (i = 0; i < n; i++)
```

```
        y[i] = a*x[i] + y[i];
```

```
}
```

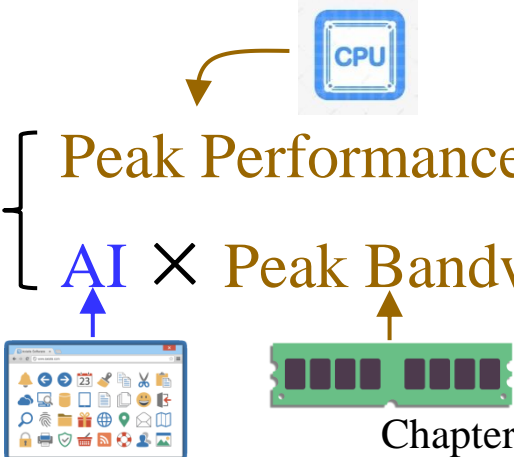


$$\text{AI} = \frac{\text{Flops \#}}{\text{Bytes \#}} = \frac{2n}{24n} = \frac{1}{12}$$



# Roofline Performance Model (5/7)

- The Roofline model finds the upper bound on performance by using the **peak performance** and **peak bandwidth**.
  - **Peak Performance** (flops/second): The floating point max performance of the processor.
  - **Peak Bandwidth** (bytes/second): The fastest the processor can load data from memory.
- A formula for calculating **attainable performance** (Gflops/s)

$$\text{Attainable Performance (AI)} = \min \left\{ \begin{array}{l} \text{Peak Performance} \\ \text{AI} \times \text{Peak Bandwidth} \end{array} \right.$$


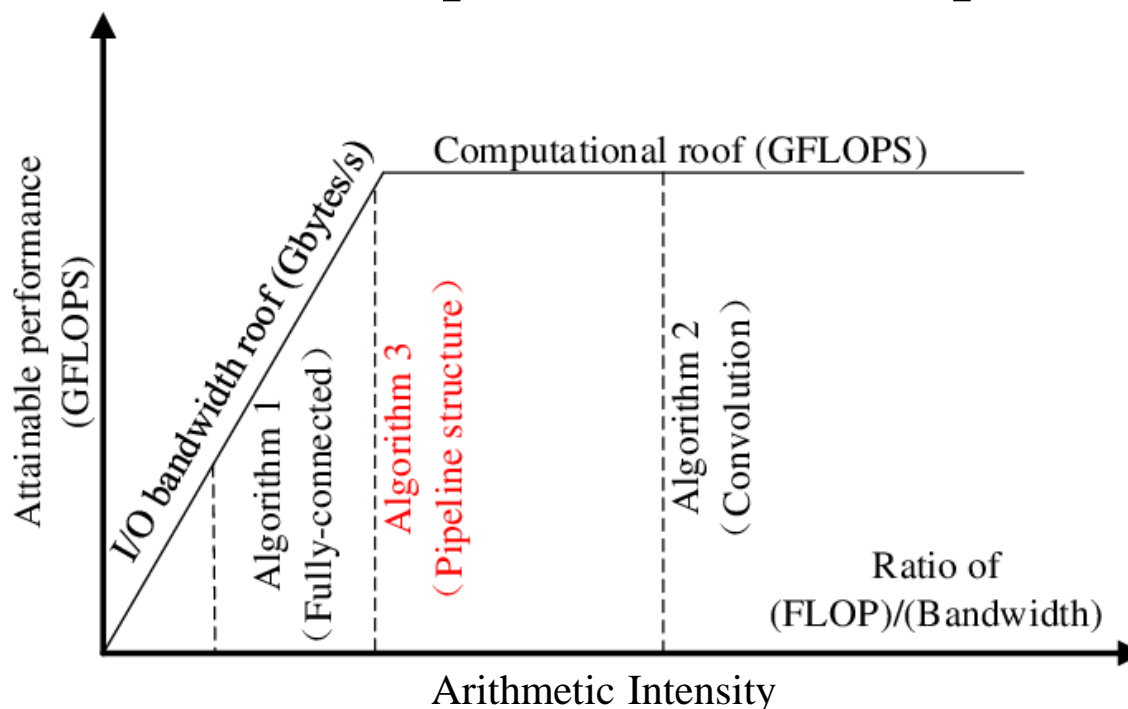
Chapter 7 65

# Roofline Performance Model (6/7)

$$\text{Attainable Performance (AI)} = \min \begin{cases} \text{Peak Performance} \\ \text{AI} \times \text{Peak Bandwidth} \end{cases}$$

## ■ Roofline model

- **X-axis:** Arithmetic Intensity (AI)
- **Y-axis:** Attainable performance (Gflops/s)

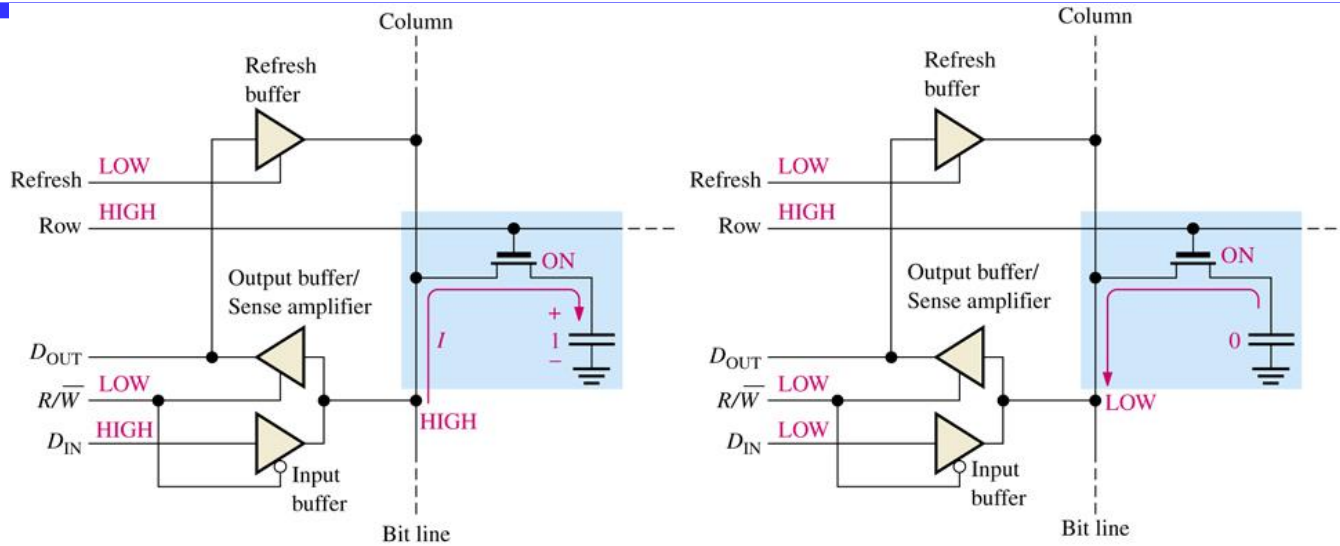


# Roofline Performance Model (7/7)

---

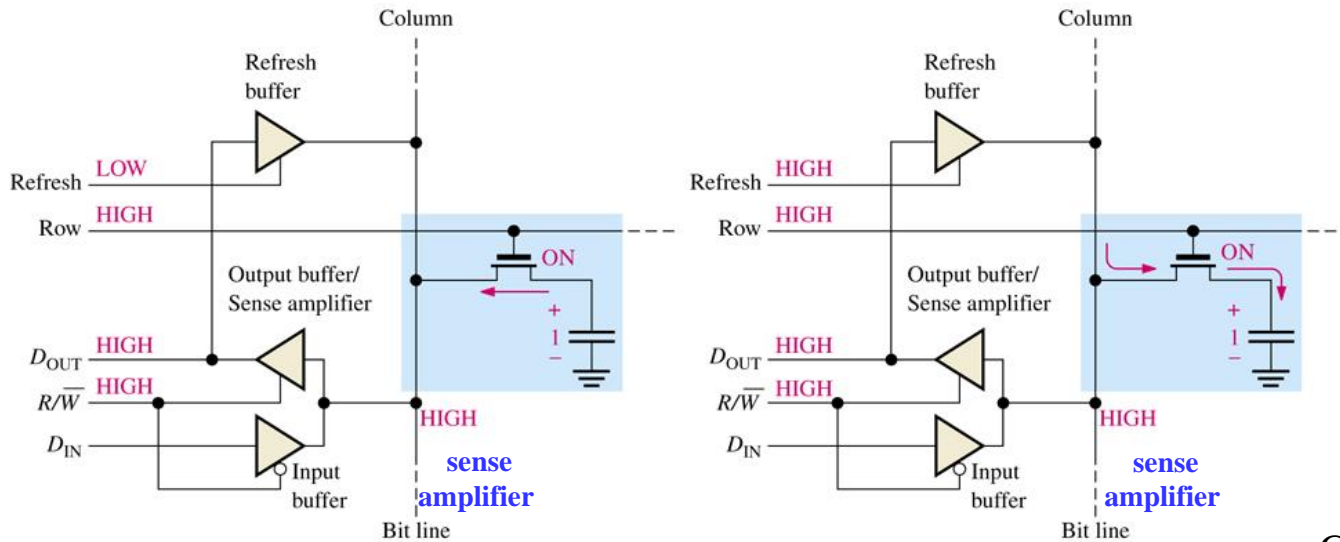
- For example, consider that a processor has
  - a peak performance of 64 Gflops/s and
  - a peak bandwidth of 16 GB/s
- If  $AI = \frac{1}{8}$ , then
  - Attainable Performance =  $\min \{64, \frac{1}{8} \times 16\} = 2$  Gflops/s
  - The attainable performance is limited by memory operations.
- If  $AI = 8$ , then
  - Attainable Performance =  $\min \{64, 8 \times 16\} = 64$  Gflops/s
  - The attainable performance is limited by CPU's computation.

# Appendix B: Dynamic RAM - Basic Operation



(a) Writing a 1 into the memory cell

(b) Writing a 0 into the memory cell



(c) Reading a 1 from the memory cell

(d) Refreshing a stored 1

# Appendix C: Synchronous DRAM

- SDRAM has a **mode control word** that can be loaded from the address bus.

Mode Register

BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	OP Code	0	0	CAS Latency			BT	Burst Length		

OP Code

A9	Write Mode
0	Burst Read and Burst Write
1	Burst Read and Single Write

Burst Type

A3	Burst Type
0	Sequential
1	Interleave

CAS Latency

A6	A5	A4	CAS Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	2
0	1	1	3
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Burst Length

A2	A1	A0	Burst Length	
			A3 = 0	A3 = 1
0	0	0	1	1
0	0	1	2	2
0	1	0	4	4
0	1	1	8	8
1	0	0	Reserved	Reserved
1	0	1	Reserved	Reserved
1	1	0	Reserved	Reserved
1	1	1	Full Page	Reserved

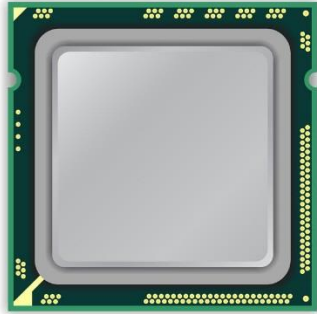
# Synchronous DRAM (continued)

- The order of the word retrieved depends on the least three significant bits of the column address and burst type (sequential or interleaved)

Burst Length	Starting Column Address			Order of Accesses Within a Burst	
				Type = Sequential	Type = Interleaved
2			A0		
			0	0-1	0-1
			1	1-0	1-0
4		A1	A0		
			0	0-1-2-3	0-1-2-3
			1	1-2-3-0	1-0-3-2
			0	2-3-0-1	2-3-0-1
			1	3-0-1-2	3-2-1-0
8	A2	A1	A0		
			0	0-1-2-3-4-5-6-7	0-1-2-3-4-5-6-7
			1	1-2-3-4-5-6-7-0	1-0-3-2-5-4-7-6
			0	2-3-4-5-6-7-0-1	2-3-0-1-6-7-4-5
			1	3-4-5-6-7-0-1-2	3-2-1-0-7-6-5-4
			0	4-5-6-7-0-1-2-3	4-5-6-7-0-1-2-3
			1	5-6-7-0-1-2-3-4	5-4-7-6-1-0-3-2
			0	6-7-0-1-2-3-4-5	6-7-4-5-2-3-0-1
			1	7-0-1-2-3-4-5-6	7-6-5-4-3-2-1-0

## Appendix D: DRAM-row hammer

# x86 CPU



# DRAM Module



# DDR3

1. Avoid *cache hits*
  - Flush **X** from cache
2. Avoid *row hits* to **X**
  - Read **Y** in another row

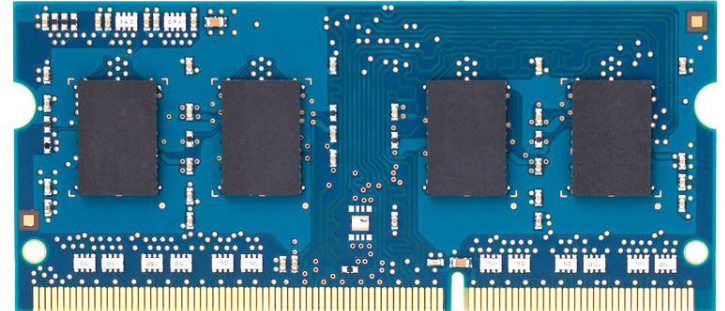
The diagram illustrates the input stage of a neural network. It shows two input vectors, **X** and **Y**, each of size 1x9. The vectors are represented as rows of nine '1's. The vector **X** is shown as a row of nine '1's, and the vector **Y** is shown as a row of nine '1's. The vectors are fed into the network, which is represented by a stack of layers.

# How to Induce Errors

x86 CPU



DRAM Module



```
loop:
```

```
mov  (X), %eax
```

```
mov  (Y), %ebx
```

```
clflush (X)
```

```
clflush (Y)
```

```
mfence
```

```
jmp  loop
```

X →

001110111

1111 | 1111

1011111101

110001011

Y →

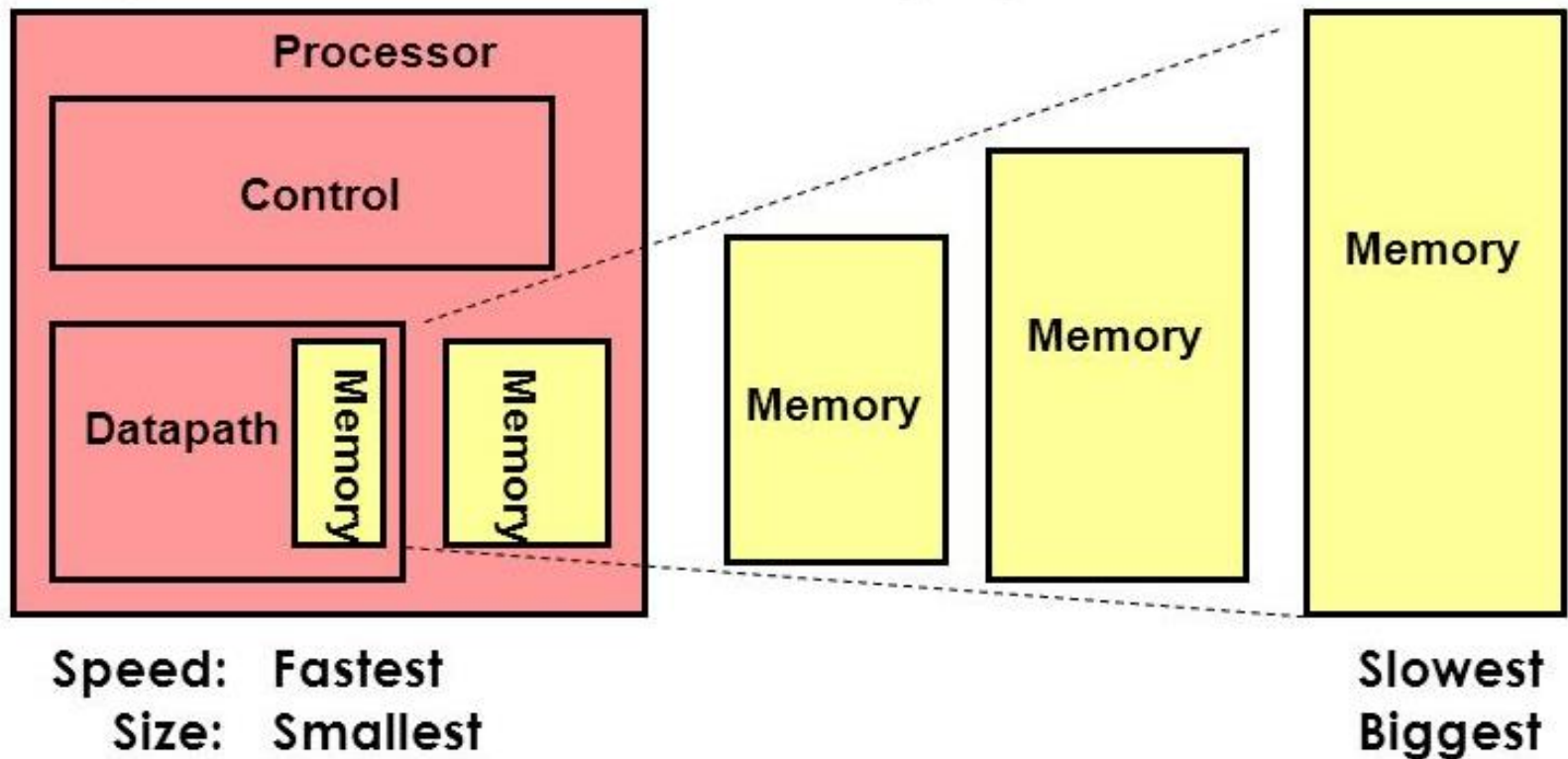
1111 | 1111

011011110



# Memory Hierarchy in Computers

- **Good memory hierarchy design is important to the overall performance of a computer system.**



# Memory Hierarchy in Computers (continued)

