CSC 220: Computer Organization

Unit 9
Counters & RAM

Prepared by:

Md Saiful Islam, PhD

Department of Computer Science

College of Computer and Information Sciences

Overview

- Counter
  - Asynchronous (Ripple) Counters
  - Program Counter
- RAM
  - Introduction
  - Reading a writing RAM
  - Size

Chapter-6, 7

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

Binary Counters

Introduction to binary counter:

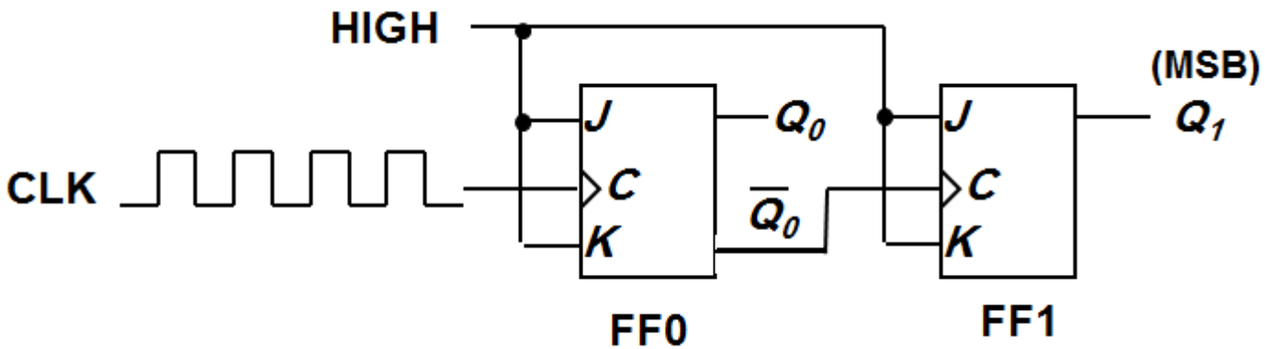
- Counter is a kind of register
- Goes through a prescribed sequence of binary numbers
- A *n*-bit binary counter consists of *n* flip-flops
- Counts: $0 – (2^n-1)$

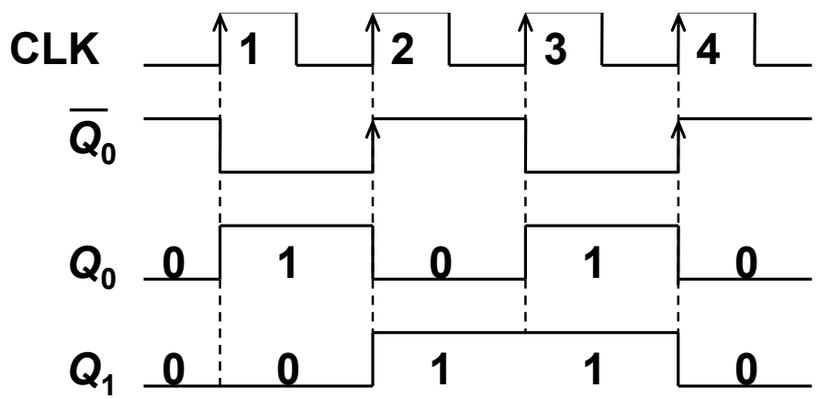
  Example: 2 bit binary counter: $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00...$
- Categories of Counter
  - **Ripple/asynchronous**: clock pulses are **different** for flip-flops
  - **Synchronous:** all flip-flops receive the **same clock pulse**

Ripple Counters

Design of a 2-bit binary ripple counter:

- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



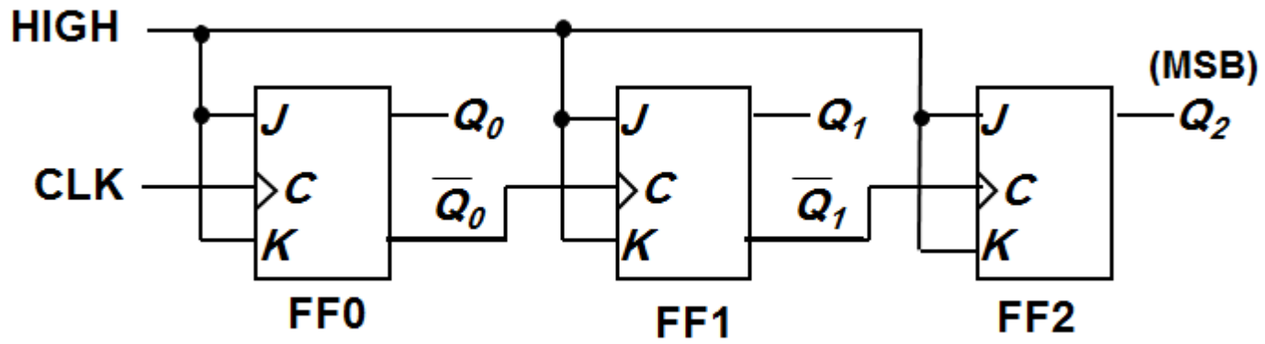


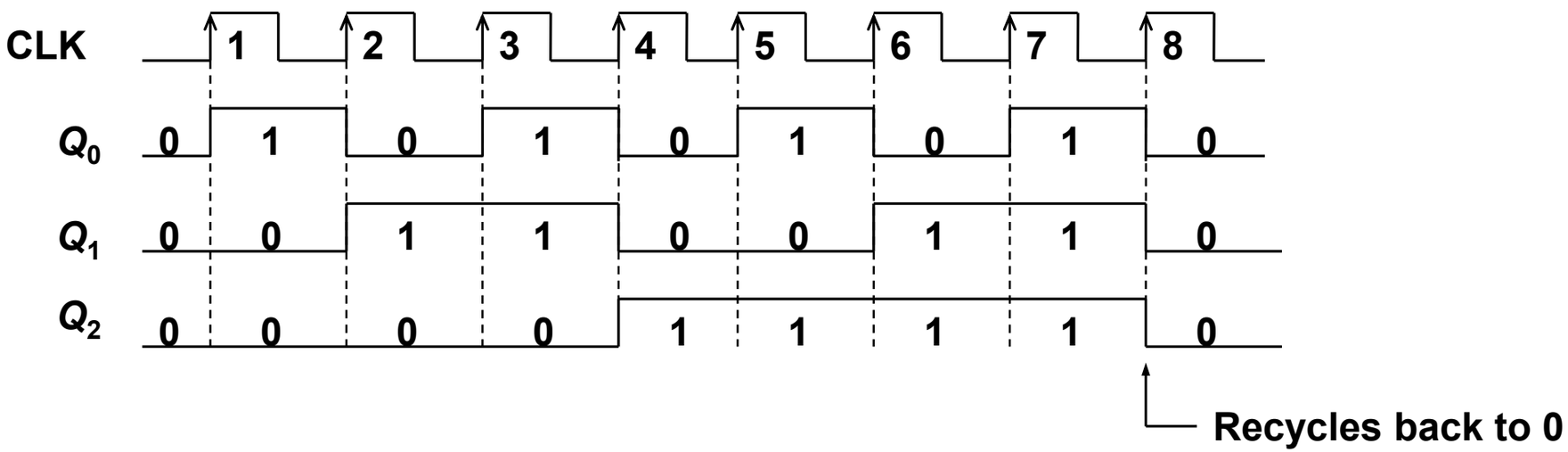
Timing diagram

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00...$

Ripple Counters …

Design of a 3-bit ripple binary counter:

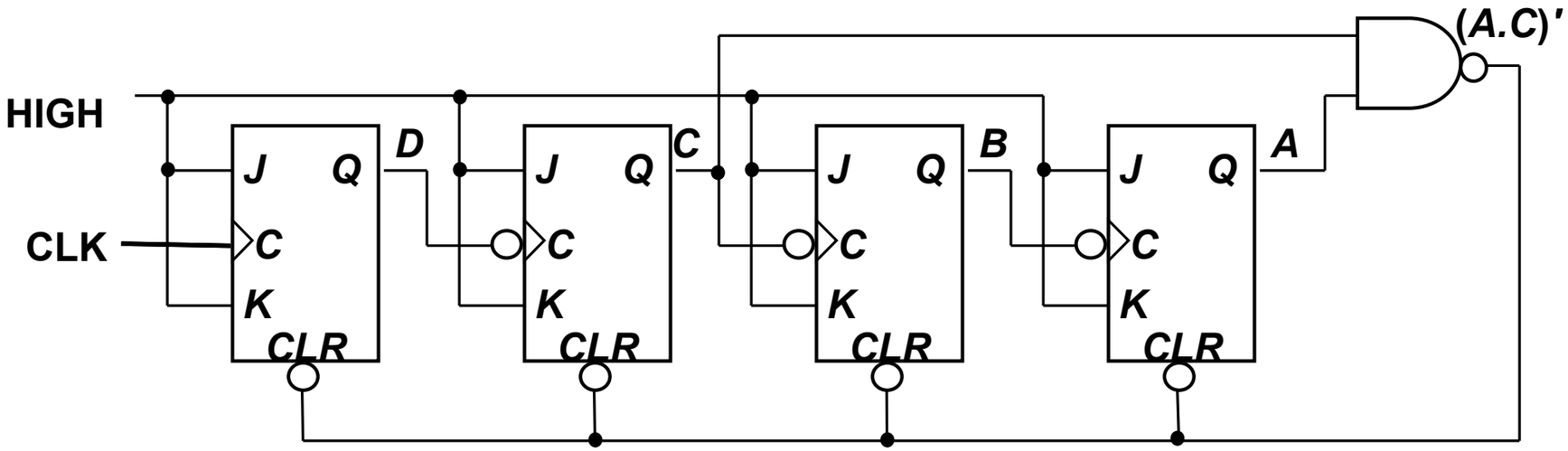




Ripple Counters …

Asynchronous Counters with MOD no. < 2^n

- Design of an asynchronous MOD-10 counter : counts (0-9)



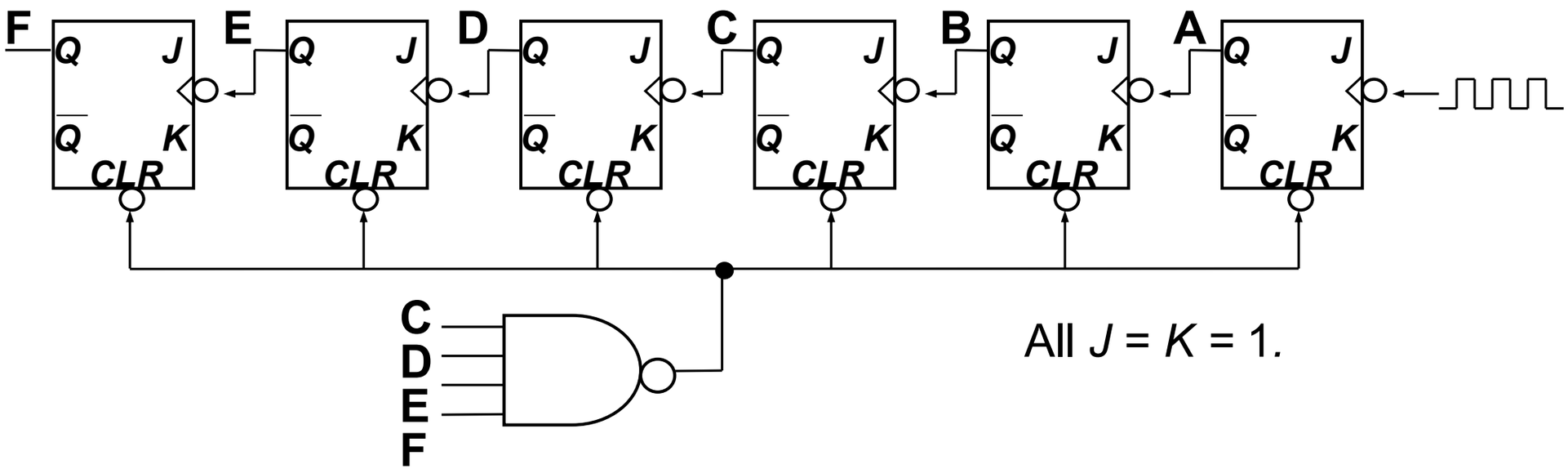
- **MOD-10 counter:** known as decade counter or BCD counter
- Counter with 10 states (modulus-10) in their sequence
- They are commonly used in daily life (e.g.: utility meters, odometers, etc.).

Ripple Counters …

Asyn. Counters with MOD no. < 2^n

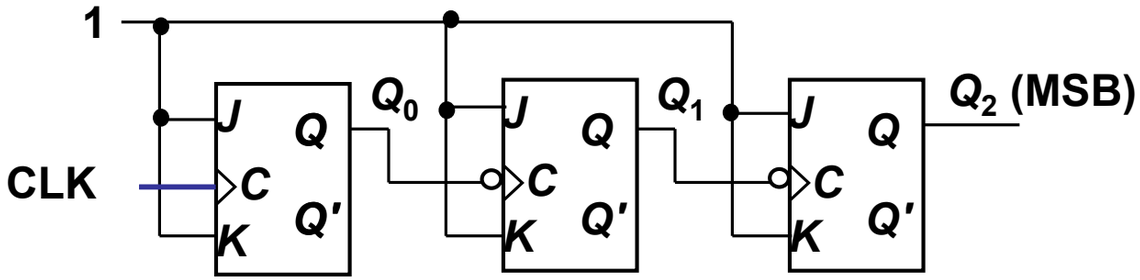
- **Exercise:** How to construct an asynchronous MOD-5 counter? MOD-7 counter? MOD-12 counter?

- **Question:** The following is a MOD-? counter?



Up and Down Counters …

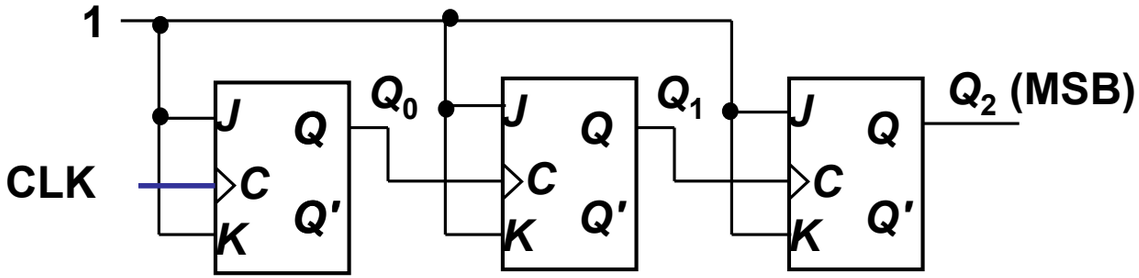
- *Up counters:* count upward from zero to a maximum value, and repeat



3-bit binary up counter

- *Down counters:* count downward from a maximum value to zero, and repeat.

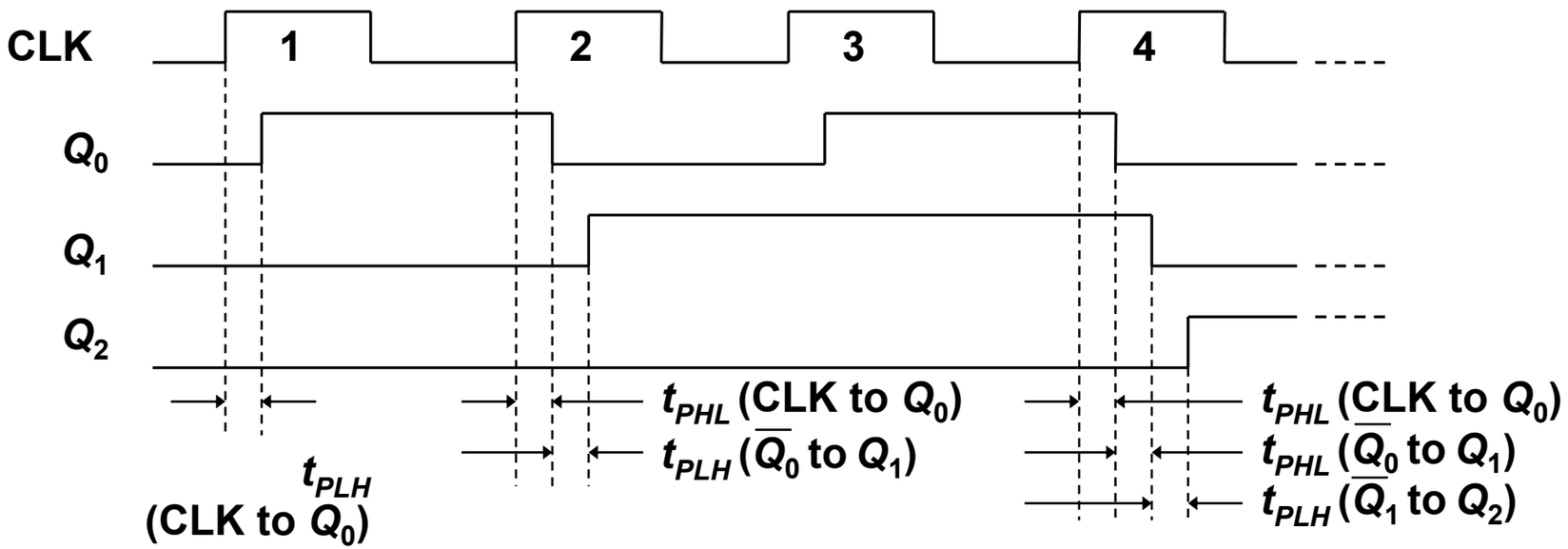
Example: A 3-bit binary down counter.



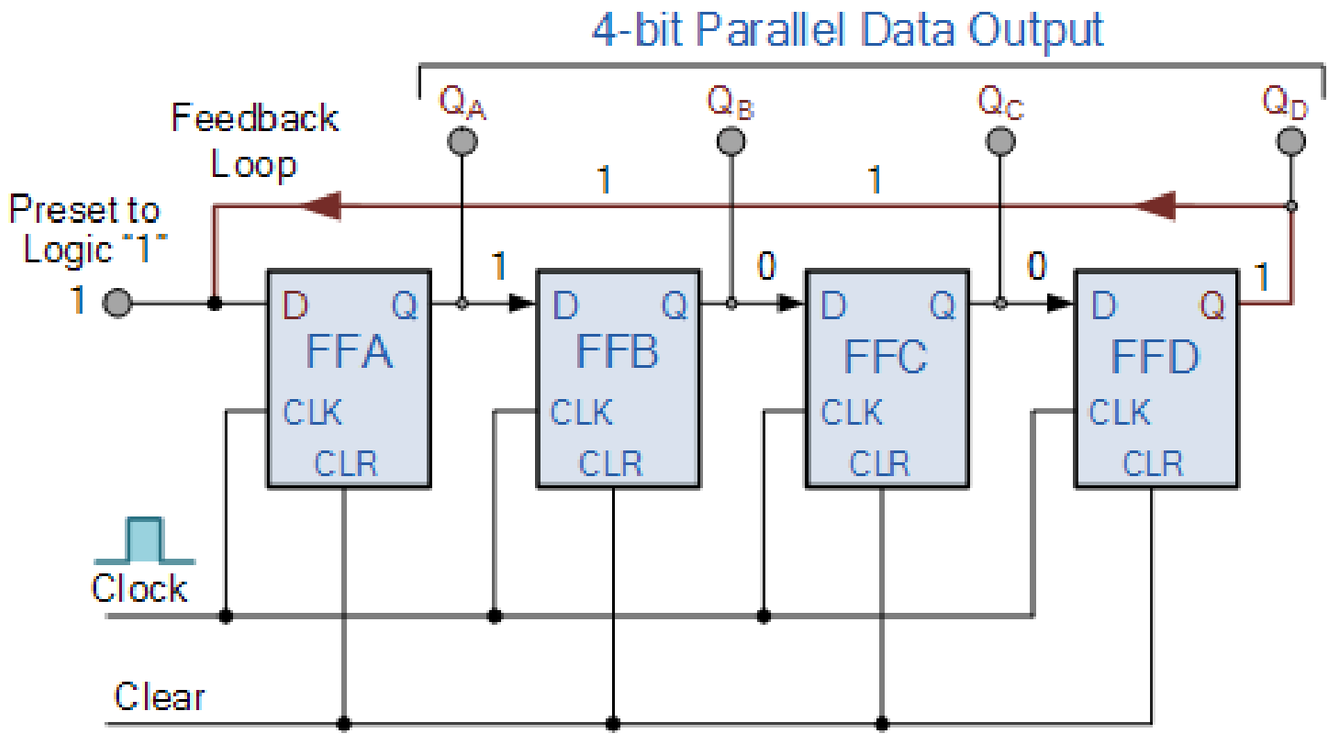
Ripple Counters …

Propagation delays in an asynchronous (ripple) counter:

- If the accumulated delay is greater than the clock pulse, some counter states may be misrepresented!



Synchronous Counter



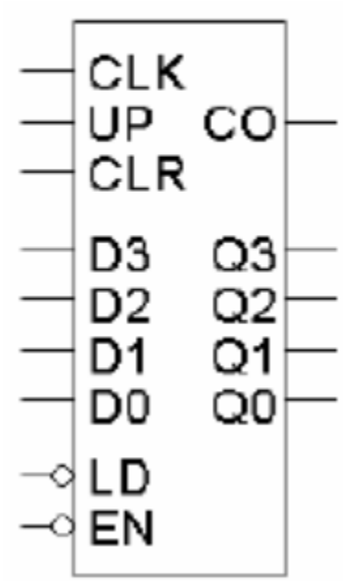
4-bit Ring Counter

- A “CLEAR” signal and a “PRESET” pulse is applied to initialize the counter.
- On each successive clock pulse, the counter circulates the same data bit between the four flip-flops over and over again around the “ring” every fourth clock cycle.

Program Counter

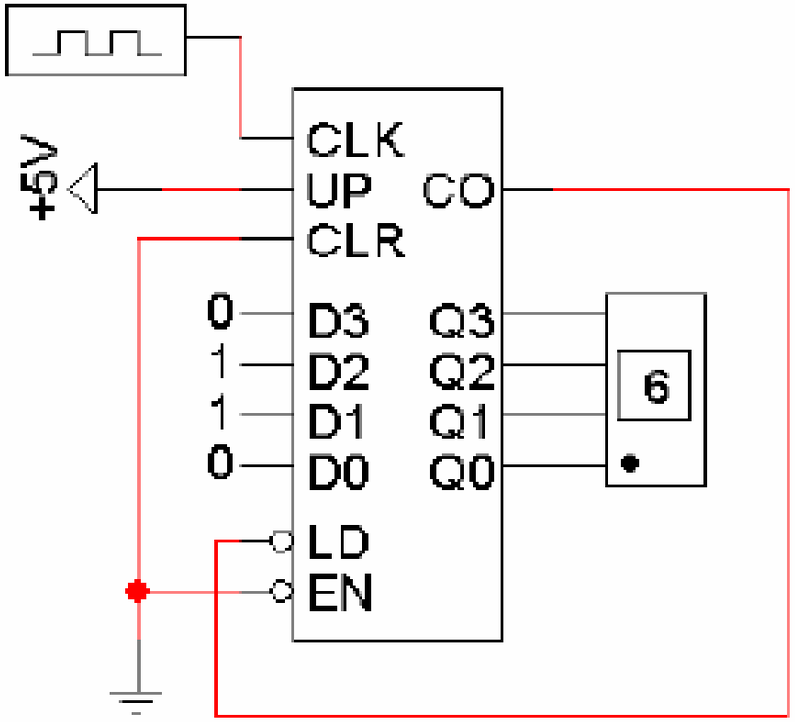
More complex counters : A Register with Parallel Load + Counting

- More complex counters are also possible. The full-featured LogicWorks Counter-4 device below has several functions.
  - It can count up or down, depending on whether the UP input is 1 or 0.
  - You can clear the counter to 0000 asynchronously by setting CLR = 1.
  - You can perform a parallel load of D3-D0 when LD = 0.
  - The active-low EN input enables or disables the counter.
  - The "counter out" CO is normally 1, but becomes 0 when the counter reaches its maximum value of 1111 (if UP = 1) or 0000 (if UP = 0).



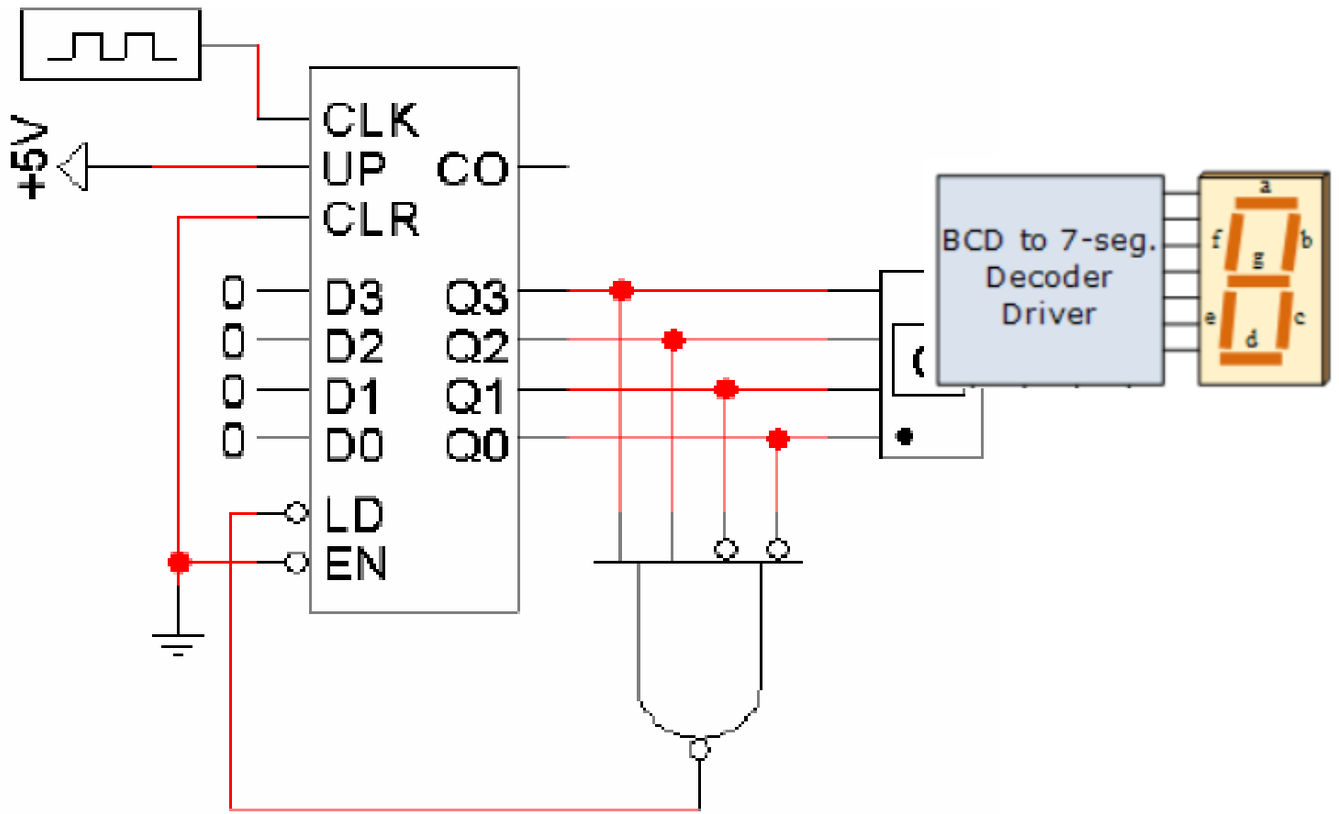
A restricted 4-bit counter

- We can also make a counter that "starts" at some value besides 0000.
- In the diagram below, when CO = 0 the LD signal forces the next state to be loaded from D3-D0.
- The result is this counter wraps from 1111 to 0110 (instead of 0000).



Another restricted counter

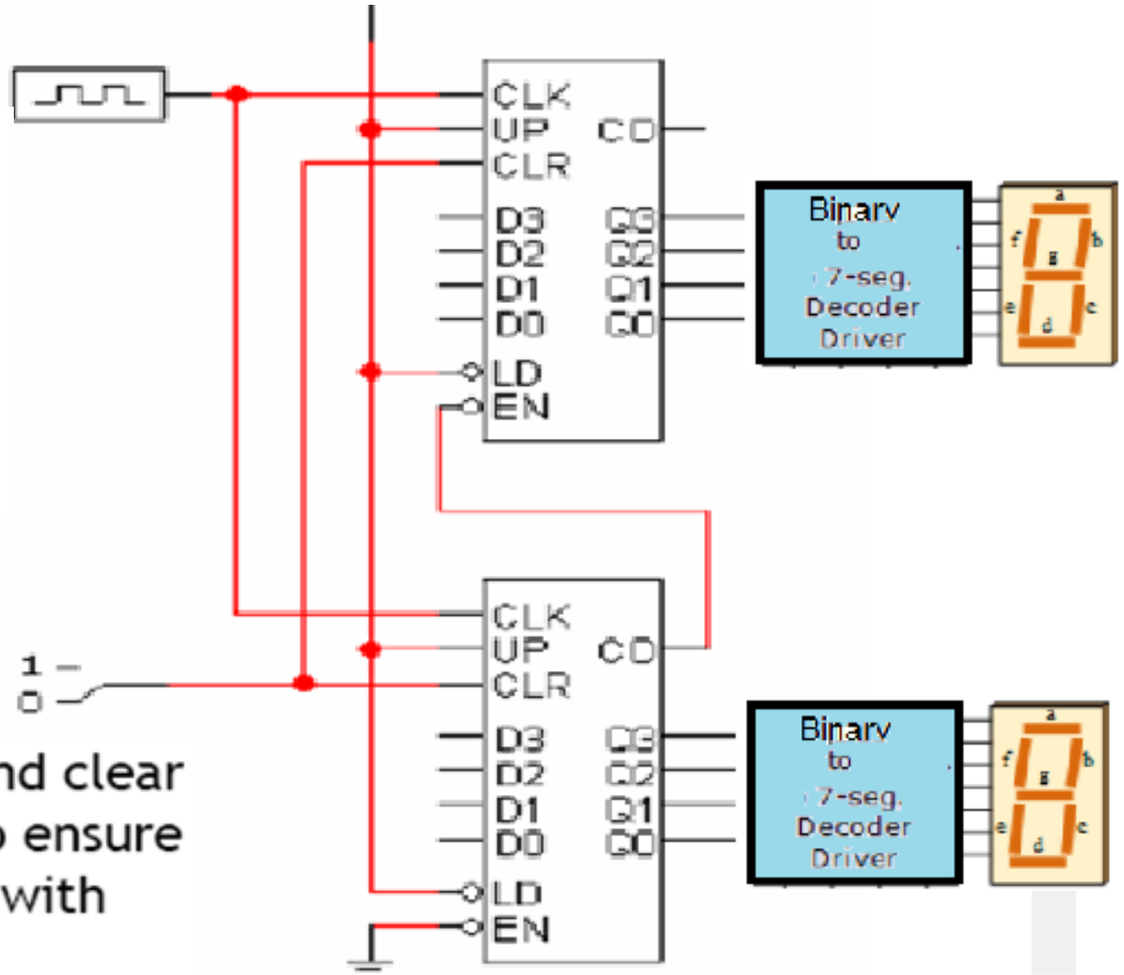
- We can also make a circuit that counts up to only 1100, instead of 1111.
- Here, when the counter output reaches 1100, the NAND gate forces the counter to load, so the next state becomes 0000.



8-bit Counter …

Here is an 8-bit counter made from two 4-bit

- The bottom device represents the least significant four bits, while the top counter represents the most significant four bits.
- When the bottom counter reaches 1111 (i.e., when CO = 0), it enables the top counter for one cycle.



The two four-bit counters share clock and clear inputs. Sharing the clock is important to ensure that the two counters are synchronized with respect to each other.

We've used Hex Display units here to view the four-bit output as a single hexadecimal digit.

Introduction to RAM

- Random-access memory, or RAM, provides large quantities of temporary storage in a computer system.
- Remember the basic capabilities of a memory.
  - It should be able to store a value.
  - You should be able to read the value that was saved.
  - You should be able to change the stored value.
- A RAM is similar, except that it can store *many* values.
  - An address will specify which memory value we're interested in.
  - Each value can be a multiple-bit word (e.g., 32 bits).
- We'll refine the memory properties as follows.

A RAM should be able to:

1. Store many words, one per address
2. Read the word that was saved at a particular address
3. Change the word that's saved at a particular address

Block diagram of RAM

CS	RW	Memory operation
0	x	None
1	1	Read from ADRS
1	0	Write DATA to ADRS

- This block diagram introduces the main interface to RAM.
  - A Chip Select, CS, enables or disables the RAM.
  - ADRS specifies the address or location to read from or write to.
  - RW selects between reading from or writing to the memory.
    - To read from memory, RW should be set to 1.
      OUT will be the *n*-bit value stored at ADRS.
    - To write to memory, we set RW = 0.
      DATA is the *n*-bit value to save in memory.
- This interface makes it easy to combine RAMs together, as we'll see.

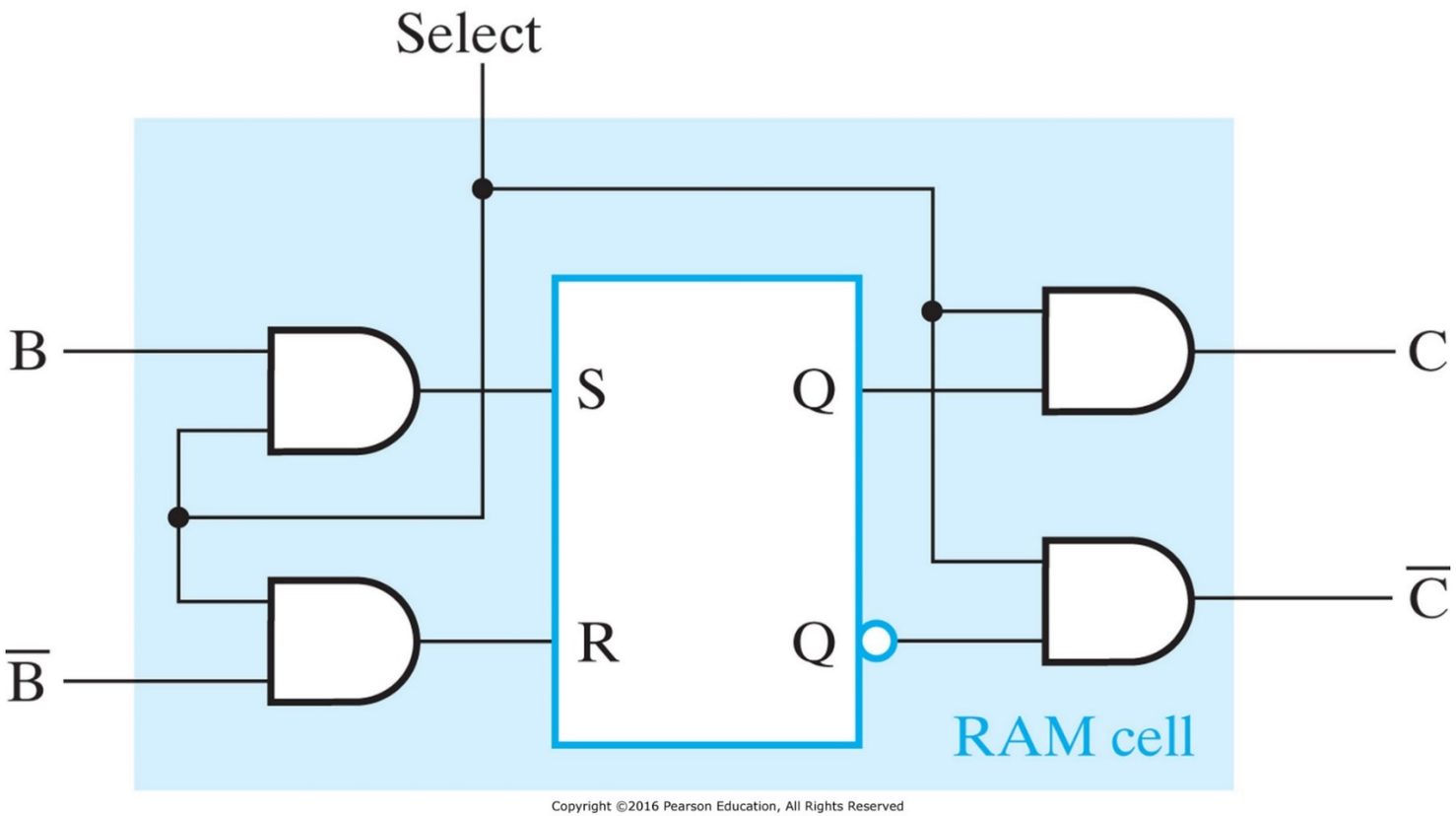
Example: Contents of a 1024 × 16 Memory

- 1K words
- 16 bits in each

Memory Address		
Binary	Decimal	Memory Contents
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
	.	.
	.	.
	.	.
	.	.
	.	.
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

A Static RAM Cell

- If Select = 0
  - S = 0, R = 0 → No change in SR Latch
  - C = 0, C′ = 0
- If Select = 1
  - S = B, R = B′ → Change in SR Latch
  - C = Q, C′ = Q′



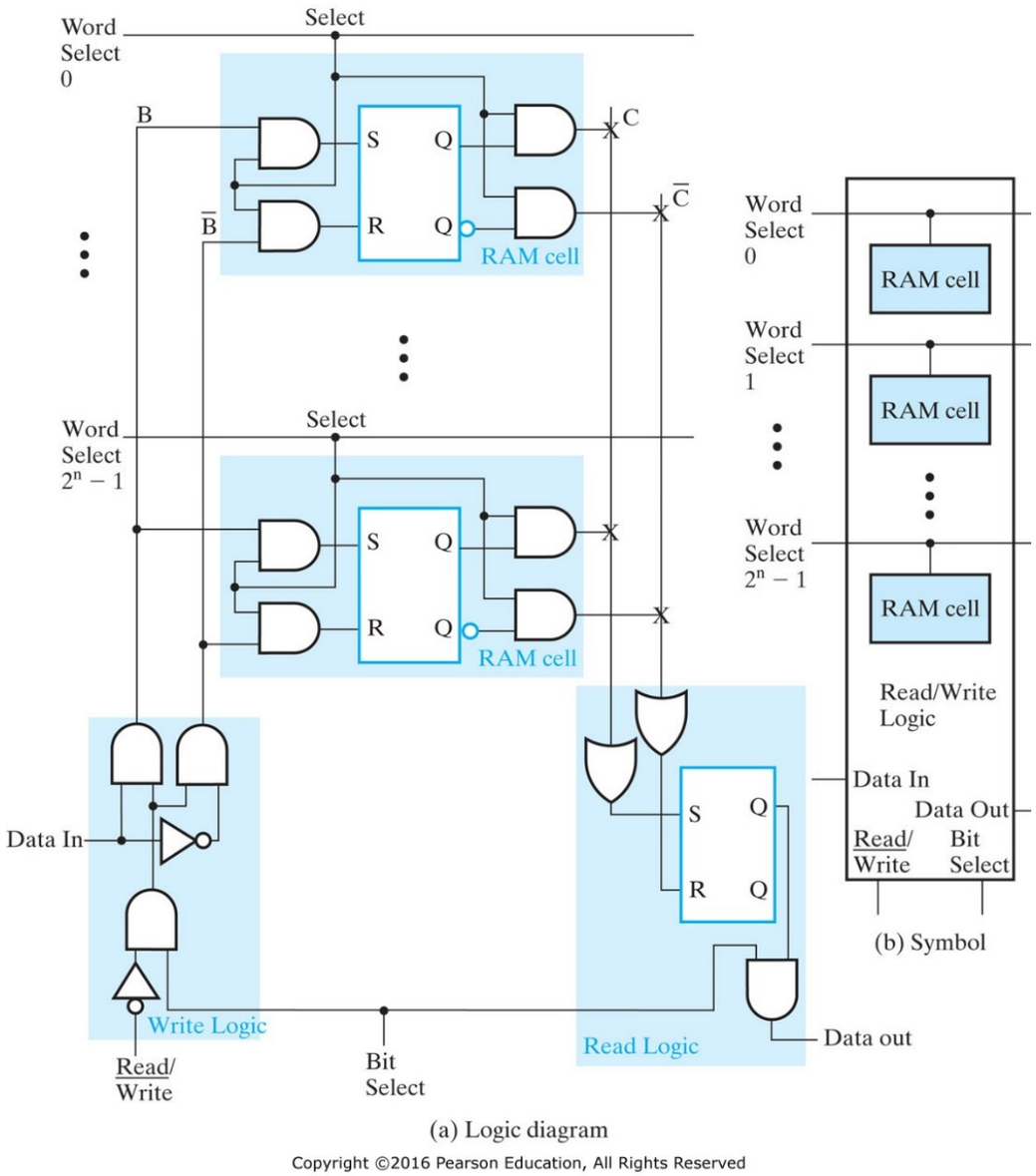
RAM Bit Slice Model

One of the words is selected at a time

- Bit Select = 0, RW = X
  B = 0, B′ = 0, Data out = 0;
- Bit Select = 1, RW = 1 (read)
  B = 0, B′ = 0, Data out = Q;
- Bit Select = 1, RW = 0 (write)
  B =Data In, B′=(Data In)′

TABLE 7-1
Control Inputs to a Memory Chip

Chip Select CS	Read/$\overline{\text{Write}}$ R/$\overline{\text{W}}$	Memory Operation
0	×	None
1	0	Write to selected word
1	1	Read from selected word



16-Word by 1-Bit RAM Chip

- 1 bit/word
- 4-to-16 Decoder is used to select a word

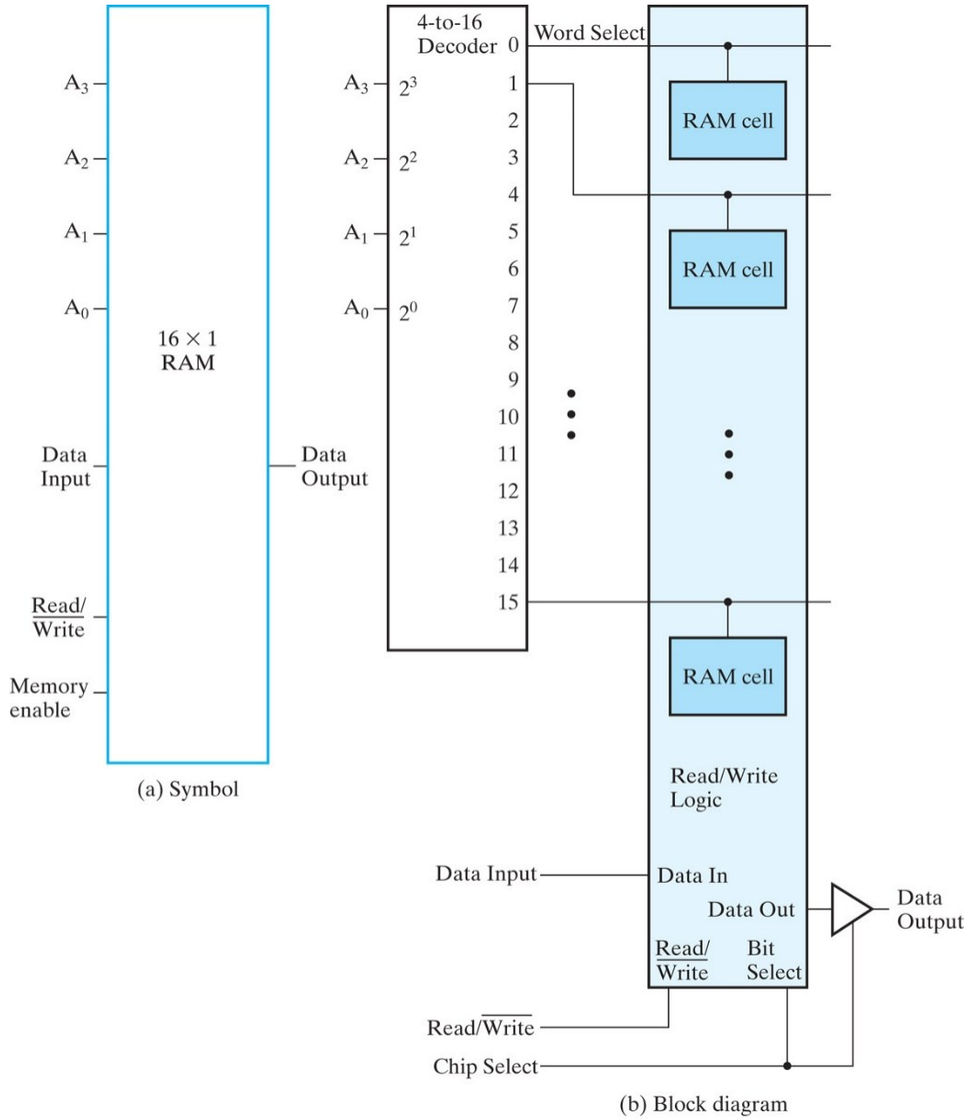
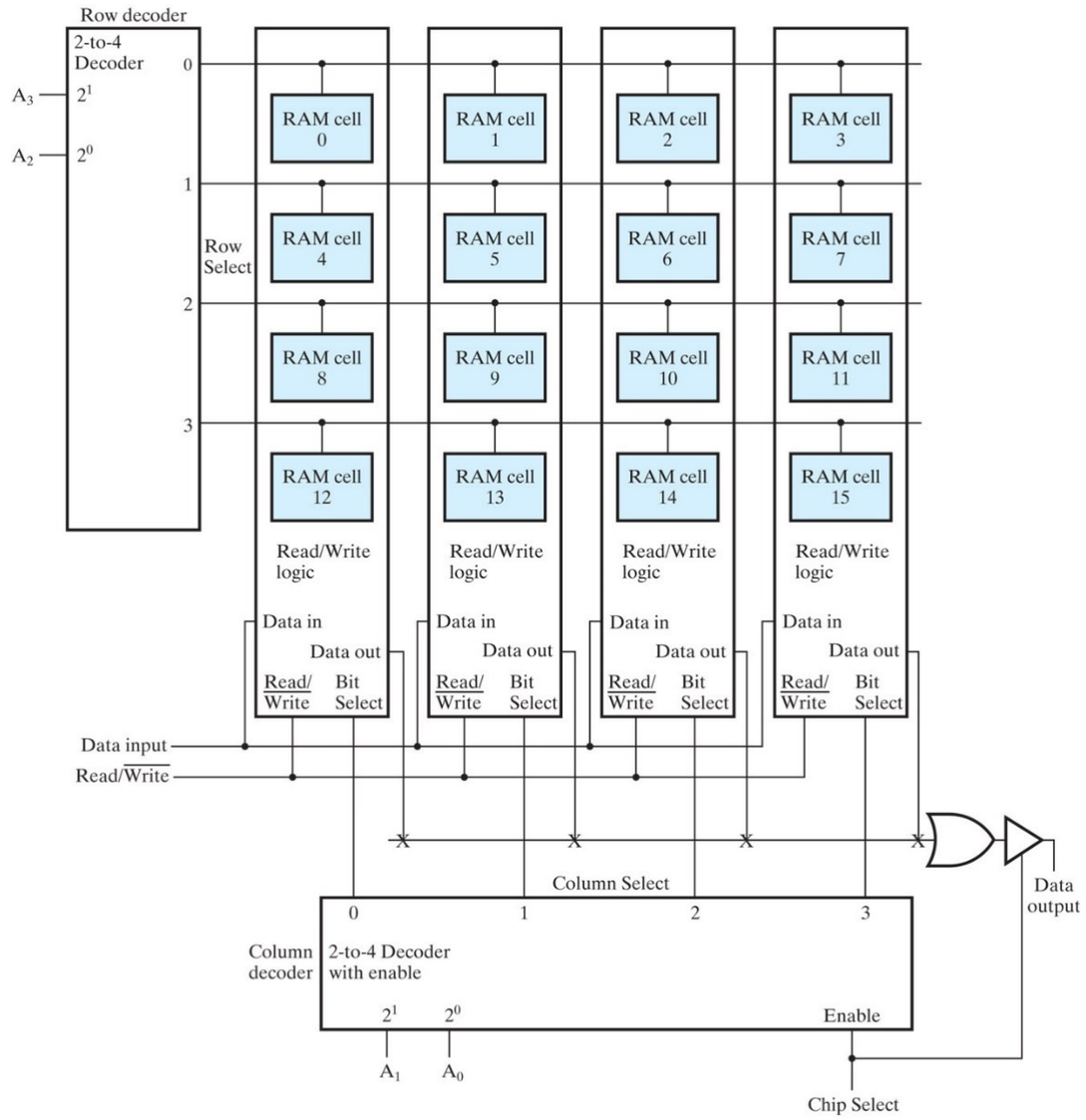


Diagram of a 16 × 1 RAM

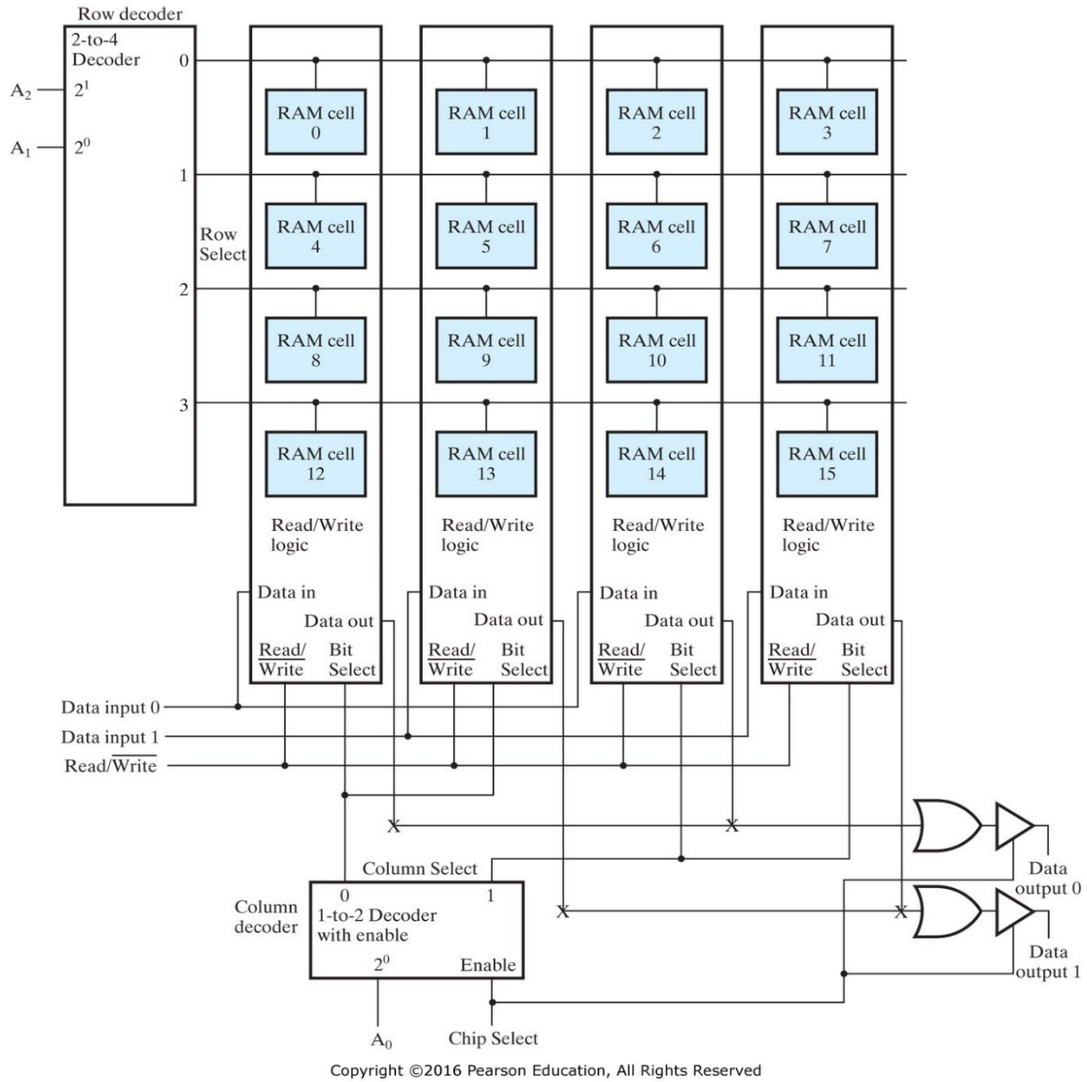
- Using a 4 × 4 RAM Cell Array



8 × 2 RAM

Block Diagram of an 8 × 2 RAM

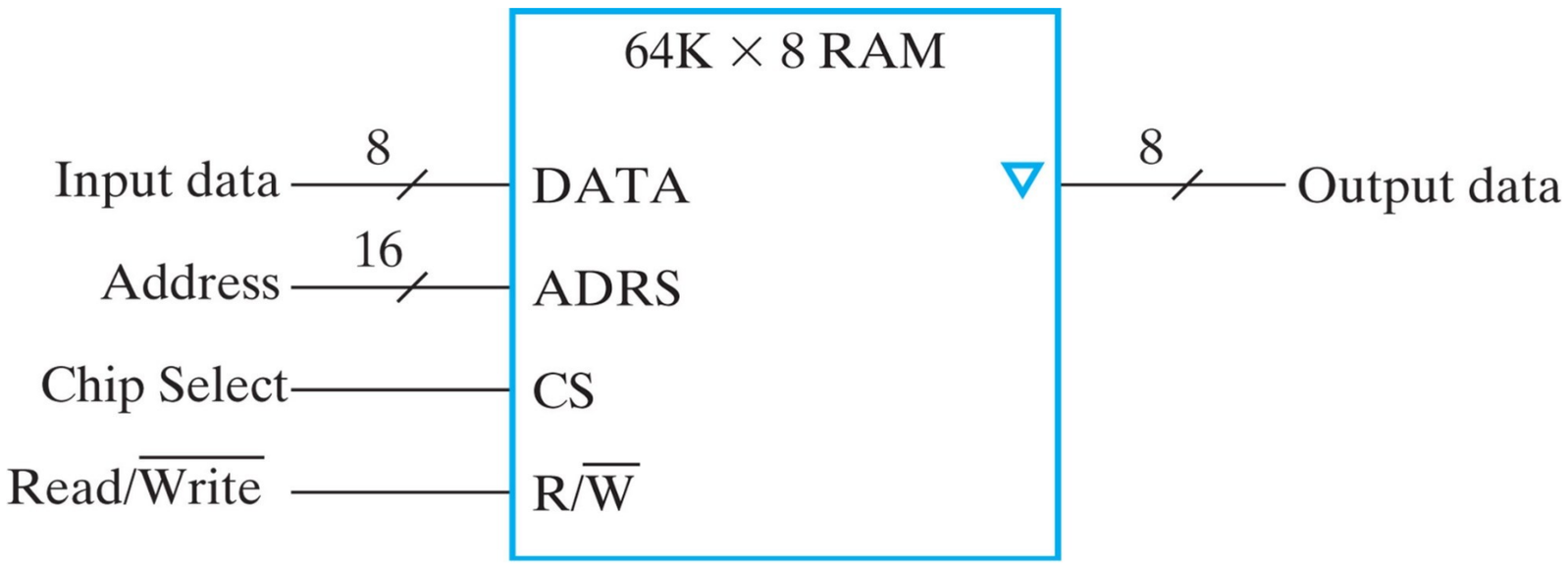
- Using a 4 × 4 RAM Cell Array



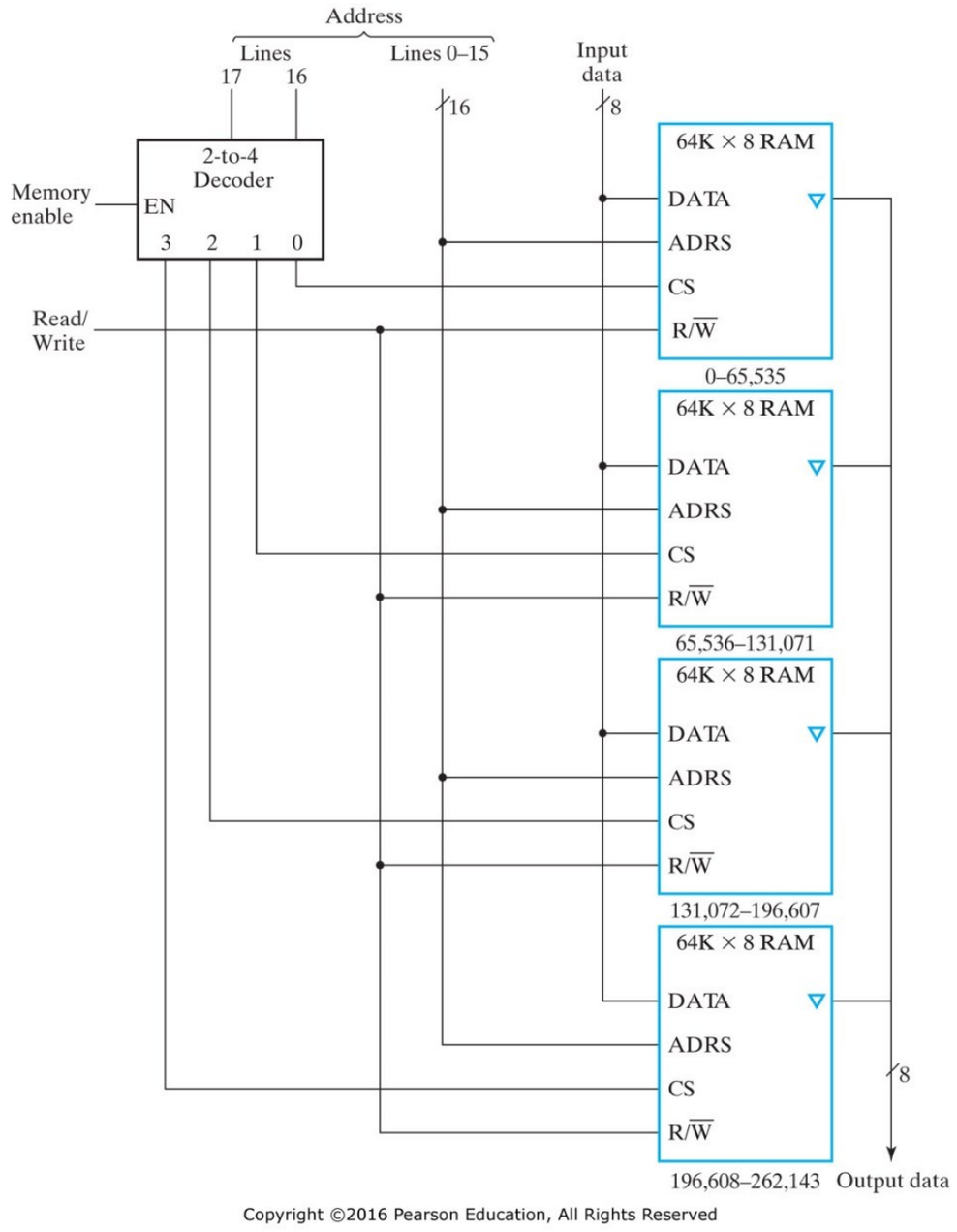
Exercise: Design a 4 × 4 RAM Using a 4 × 4 RAM Cell Array

A 64K × 8 RAM Chip

Symbol for a 64K × 8 RAM Chip



Block Diagram of a 256K × 8 RAM



Exercise: Design a 512 × 8 RAM Using 64 × 8 RAM chips

Memory sizes

- We refer to this as a $2^k \times n$ memory.
  - There are k address lines, which can specify one of 2^k addresses.
  - Each address contains an n-bit word.

- For example, a $2^{24} \times 16$ RAM contains 2^{24} = 16M words, each 16 bits long.
  - The RAM would need 24 address lines.
  - The total storage capacity is $2^{24} \times 16 = 2^{28}$ bits.

Size matters!

- Memory sizes are usually specified in numbers of bytes (8 bits).
- The 2^{28}-bit memory on the previous page has a capacity of 2^{25} bytes.

$$2^{28} \text{ bits} \,/\, 8 \text{ bits per byte} = 2^{25} \text{ bytes}$$

- With the abbreviations below, this is equivalent to 32 megabytes.

$$2^{25} \text{ bytes} = 2^5 \times 2^{20} \text{ bytes} = 32 \text{ MB}$$

	Prefix	Base 2	Base 10
K	Kilo	$2^{10} = 1{,}024$	$10^3 = 1{,}000$
M	Mega	$2^{20} = 1{,}048{,}576$	$10^6 = 1{,}000{,}000$
G	Giga	$2^{30} = 1{,}073{,}741{,}824$	$10^9 = 1{,}000{,}000{,}000$

RTL and Memory

TABLE 6-1
Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	$AR, R2, DR, IR$
Parentheses	Denotes a part of a register	$R2(1), R2(7{:}0), AR(L)$
Arrow	Denotes transfer of data	$R1 \leftarrow R2$
Comma	Separates simultaneous transfers	$R1 \leftarrow R2, R2 \leftarrow R1$
Square brackets	Specifies an address for memory	$DR \leftarrow M[AR]$

- Memory to register transfers are called ***read*** **operations**,
- Register to memory transfers are called ***write*** **operations**.
- Both require specification of the memory location to be used (which can be done through a special register (AR) or a special bus (Address bus).

RTL and Memory

- RTL expressions for a **Read** operation, assuming the use of an address registers:

$$AR \leftarrow address$$
$$DR \leftarrow M[AR]$$

- RTL expressions for a **Write** operation, assuming use of a data register:

$$AR \leftarrow address$$
$$DR \leftarrow value$$
$$M[AR] \leftarrow DR$$