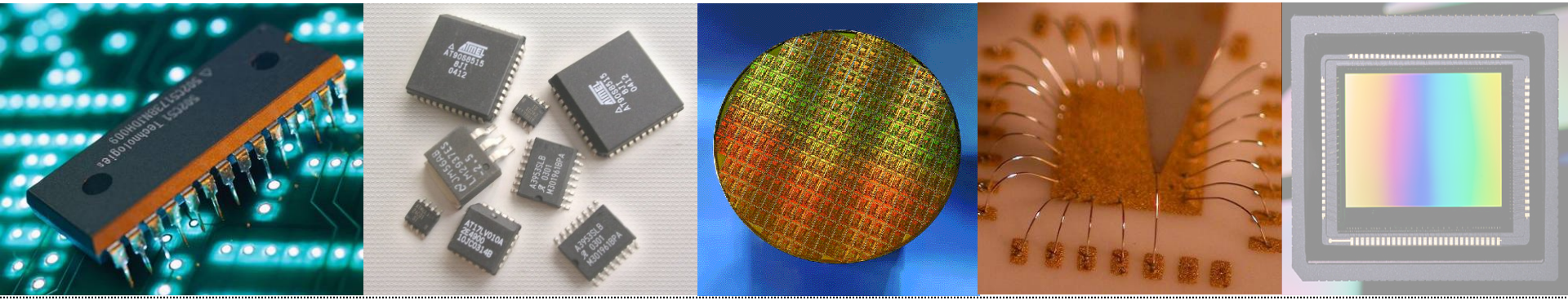


6. Registers, Counters and Memory

6.1 REGISTERS

Register- a group of binary cells suitable for holding binary information.

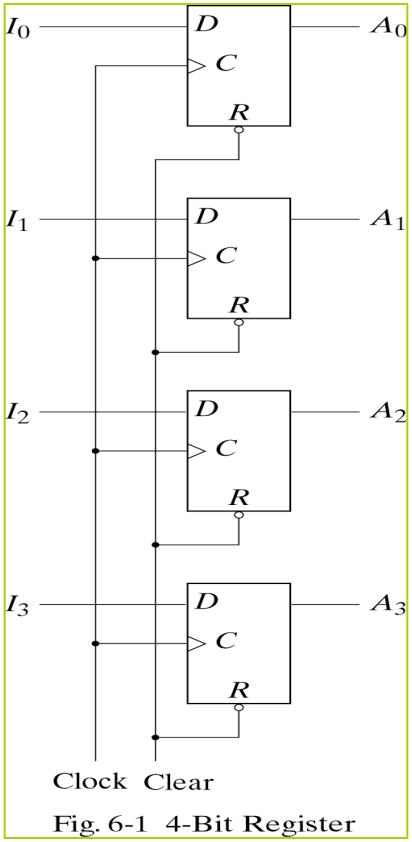


Fig. 6-1 4-Bit Register

- Clock=1 ;input information transferred
- Clock=0 ;unchanged
- Clear=0 ;clearing the register to all 0's prior to its clocked operation.

6.1 REGISTERS - Register with Parallel Load

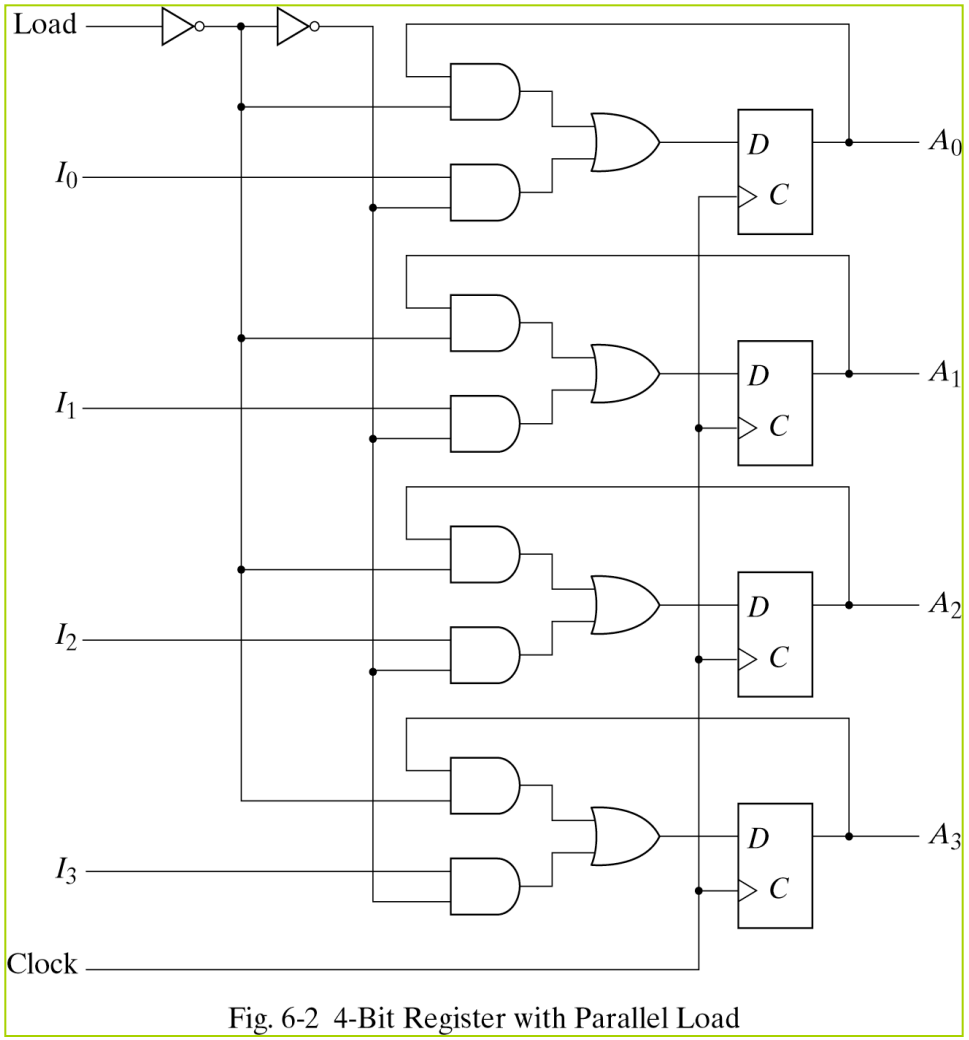


Fig. 6-2 4-Bit Register with Parallel Load

- Clock=1 ;input information ->loading
- Clock=0 ;the content of the register ->unchanged
- Load input=1 ; the I inputs are transferred into the register
- Load input=0 ; maintain the content of the register

6.2 SHIFT REGISTERS

Shift register-capable of shifting its binary information in one or both dir ections

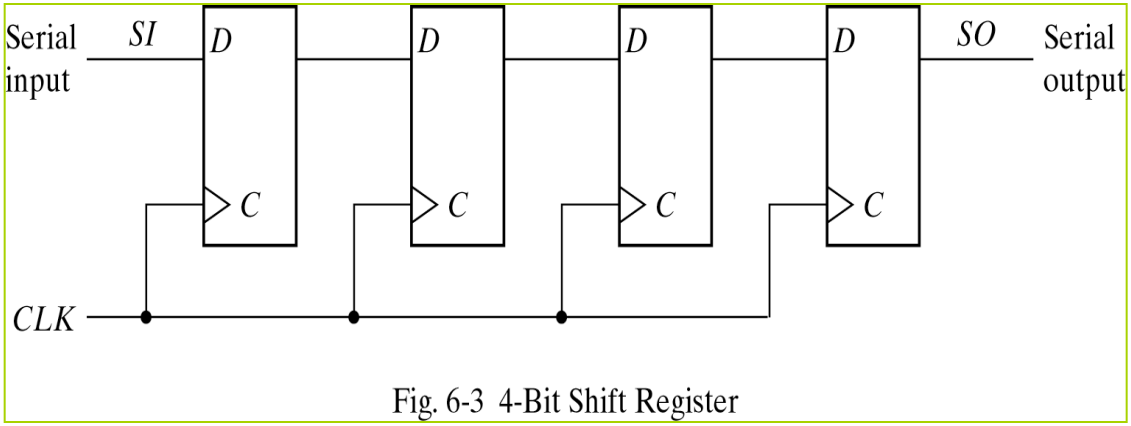


Fig. 6-3 4-Bit Shift Register

The simplest shift register

6.2 SHIFT REGISTERS - Serial Transfer

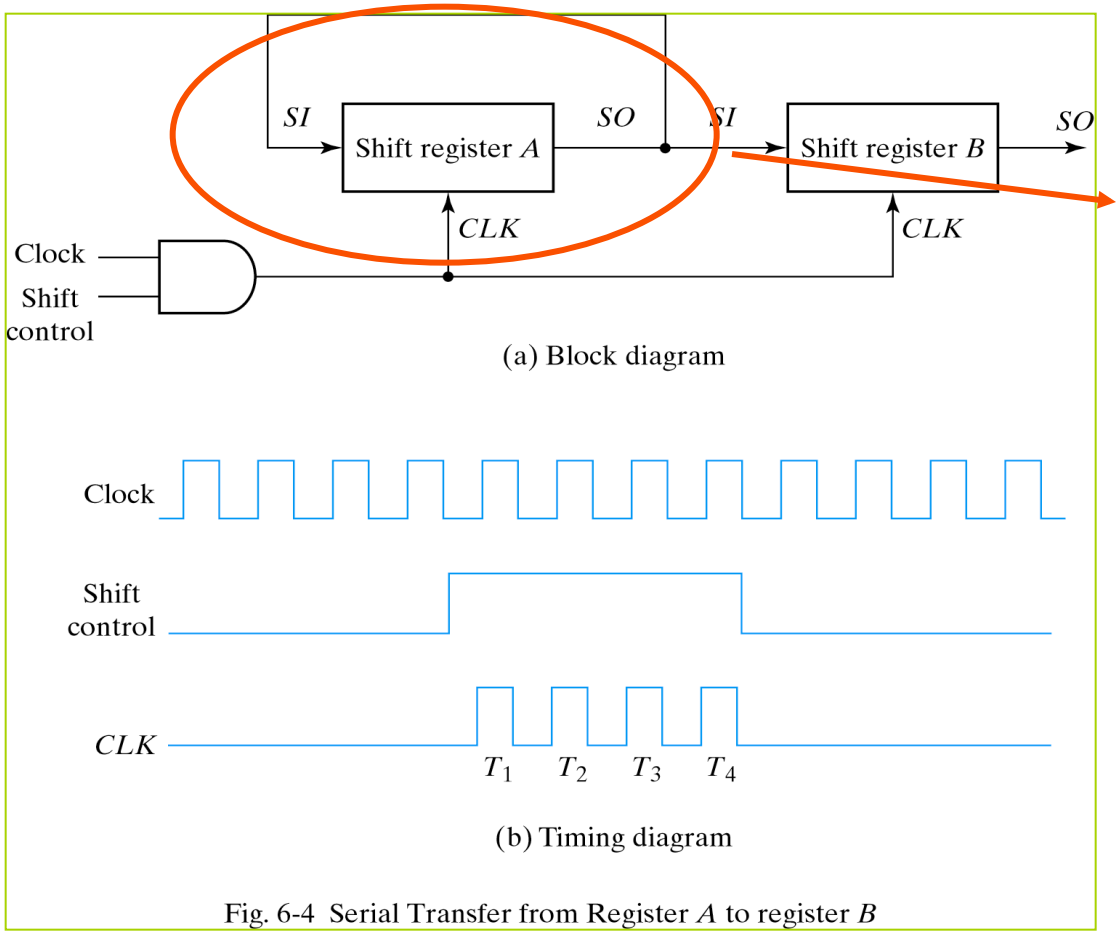
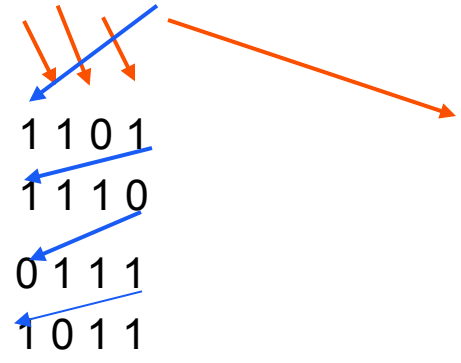


Fig. 6-4 Serial Transfer from Register *A* to register *B*

To prevent the loss of information stored in the source register

- Serial-Transfer Example

Timing pulse	Shift register A	Shift register B	Serial output of B
Initial value	1 0 1 1	0 0 1 0	0
After T_1	1 1 0 1	1 0 0 1	1
After T_2	1 1 1 0	1 1 0 0	0
After T_3	0 1 1 1	0 1 1 0	0
After T_4	1 0 1 1	1 0 1 1	1



6.2 SHIFT REGISTERS - Serial Addition

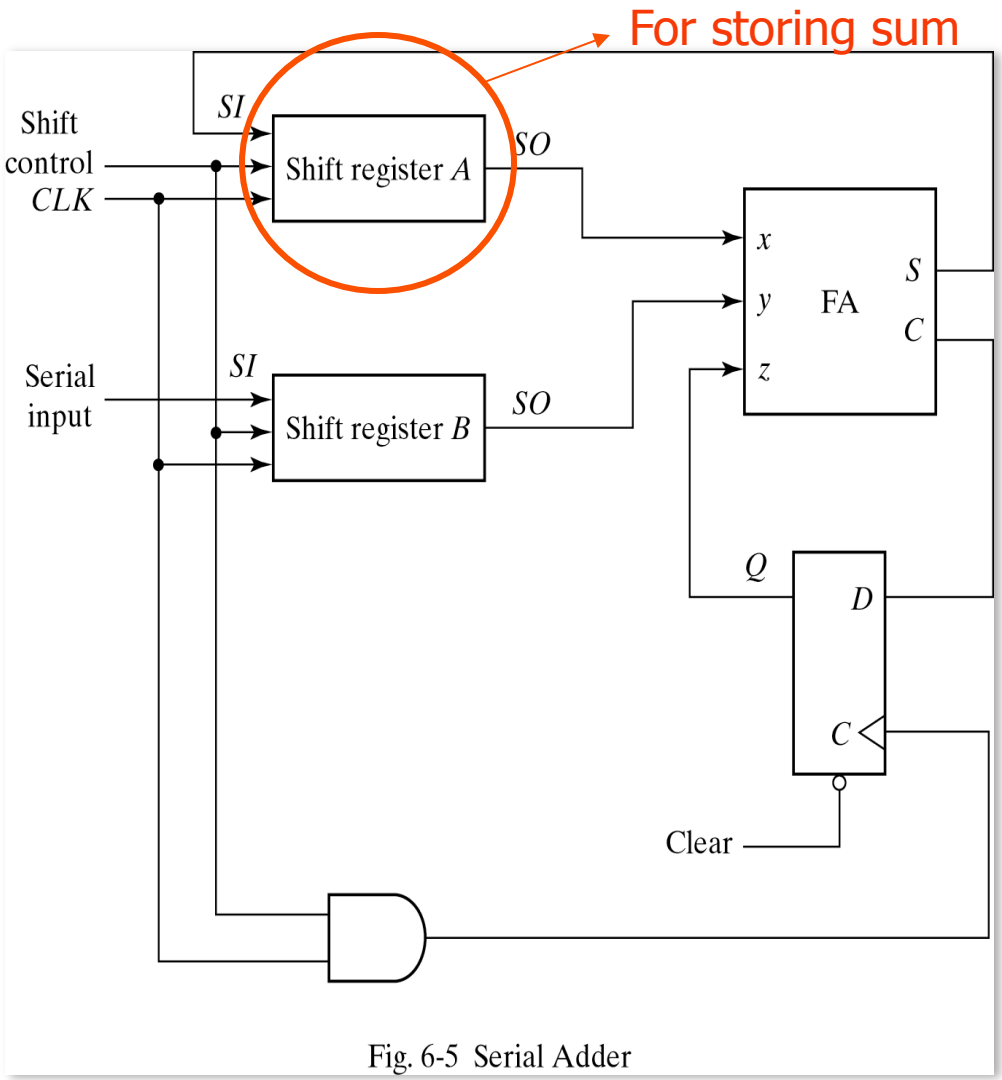


Fig. 6-5 Serial Adder

Operation

- the **A** register ->augend ,the **B** register ->addend ,carry ->0
- The **SO** of **A** and **B** provide a pair of significant bits for the **FA**
- Output **Q** gives the input carry at **z**
- The shift-right control enables both registers and the carry flip-flop.
- The sum bit from **S** enters the leftmost flip-flop of **A**

6.2 SHIFT REGISTERS - State Table for Serial Adder

Present value of carry

Output carry

Table 6-2
State Table for Serial Adder

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	X	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$$J_Q = xy$$
$$K_Q = x'y' = (x+y)'$$
$$S = x \oplus y \oplus Q$$

By k-map

6.2 SHIFT REGISTERS - Second form of Serial Adder

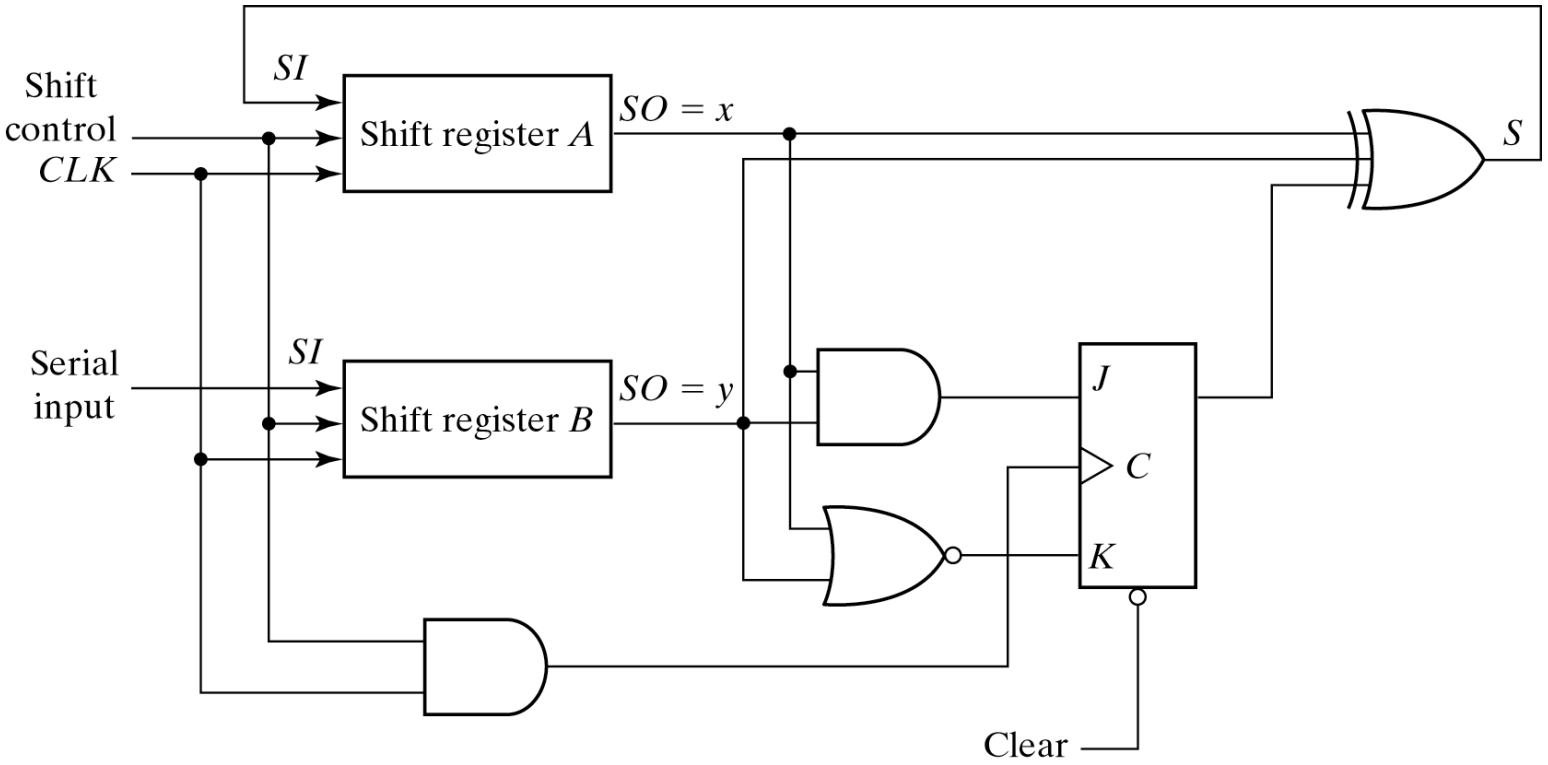


Fig. 6-6 Second form of Serial Adder

Universal Shift Register

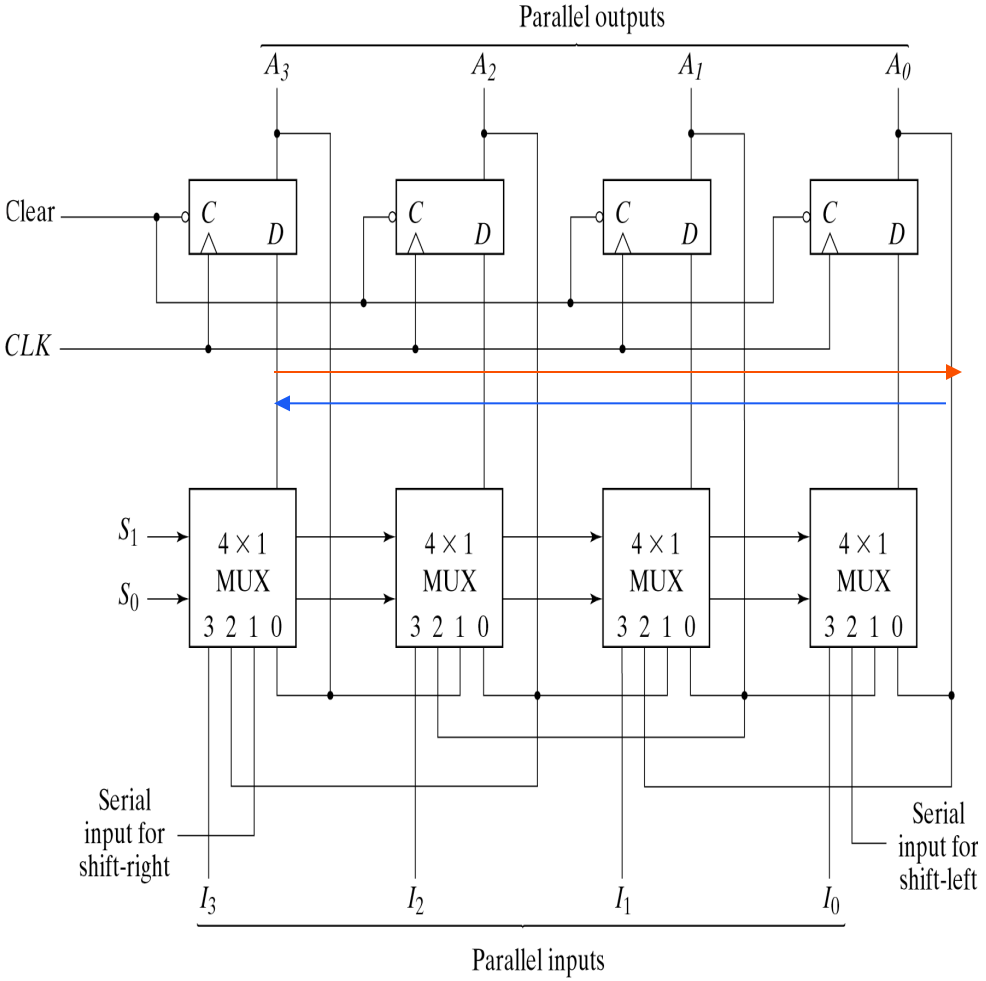
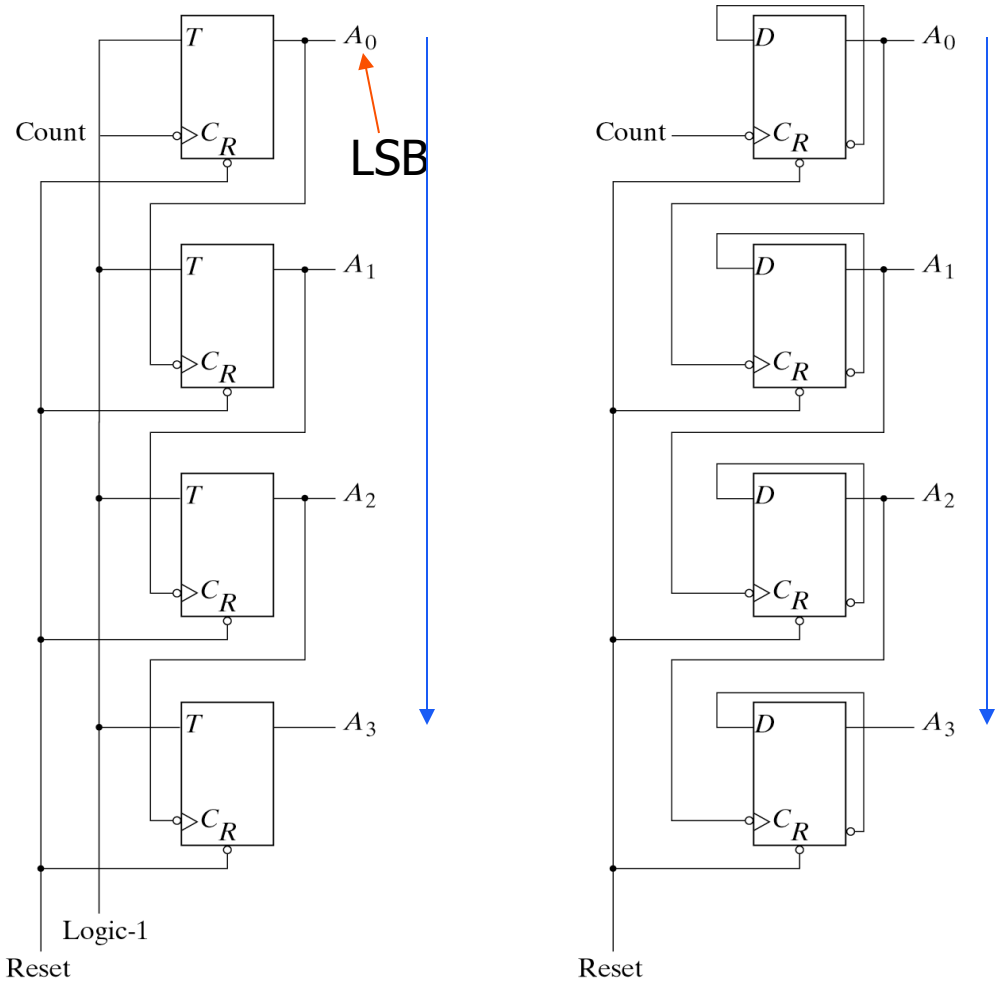


Fig. 6-7 4-Bit Universal Shift Register

- **S_1, S_0 -> 0, 0 ;No change**
- **S_1, S_0 -> 0, 1 ;Shift right**
- **S_1, S_0 -> 1, 0 ;Shift left**
- **S_1, S_0 -> 1, 1 ;Parallel load**

6.3 RIPPLE COUNTERS



(a) With T flip-flops (b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

6.3 RIPPLE COUNTERS - Binary Ripple Counter

Count sequence A_3 A_2 A_1 A_0		Conditions for Complementing
0 0 0 0	Complement A_0	
0 0 0 1	Complement A_0	A_0 will go from 1 to 0 and complement A_1
0 0 1 0	Complement A_0	
0 0 1 1	Complement A_0	A_0 will go from 1 to 0 and complement A_1 ;
0 1 0 0	Complement A_0	A_1will go from 1 to 0 and complement A_2
0 1 0 1	Complement A_0	
0 1 1 0	Complement A_0	A_0 will go from 1 to 0 and complement A_1
0 1 1 1	Complement A_0	
.....		
1 0 0 0	and so on…	

6.3 RIPPLE COUNTERS - BCD Ripple Counter

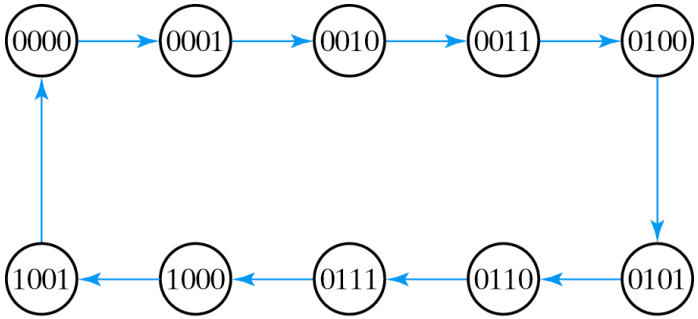


Fig. 6-9 State Diagram of a Decimal BCD-Counter

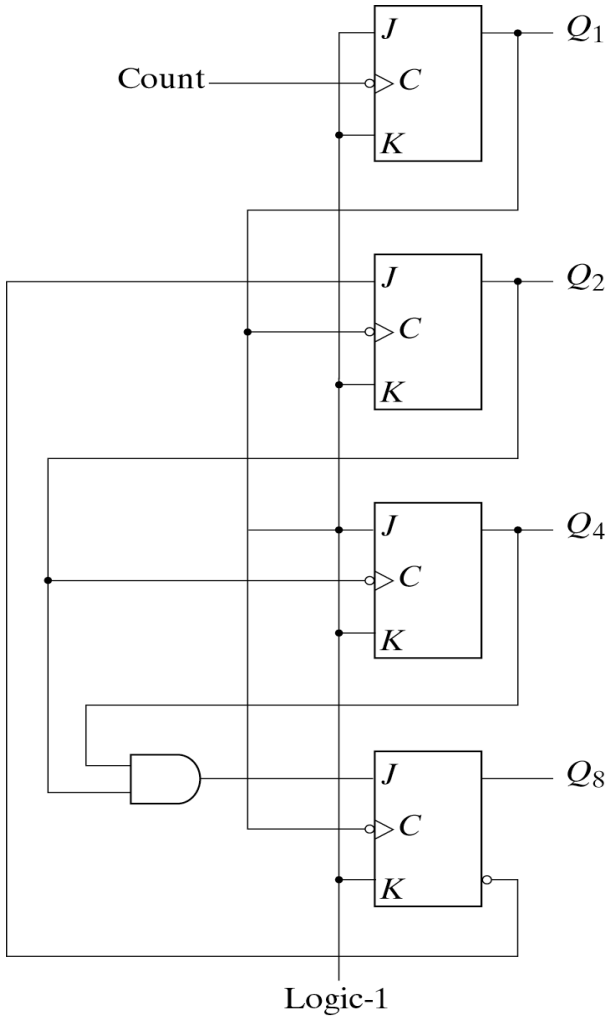


Fig. 6-10 BCD Ripple Counter

Operation

1. Q_1 is complemented on the negative edge of every co unt pulse.

2. Q_2 is complemented if Q_8=0 and Q_1 goes from 1 to 0. Q_2 is cleared if Q_8=1 and Q_1 goes from 1 to 0.

3. Q_4 is complemented when Q_2 goes from 1 to 0.

4 .Q_8 is complemented when Q_4Q_2=11 and Q_1 goes from 1 to 0. Q_8 is cleared if either Q_4 or Q_2 is 0 and Q_1 goes fro m 1 to 0

- Three-Decade Decimal BCD Counter

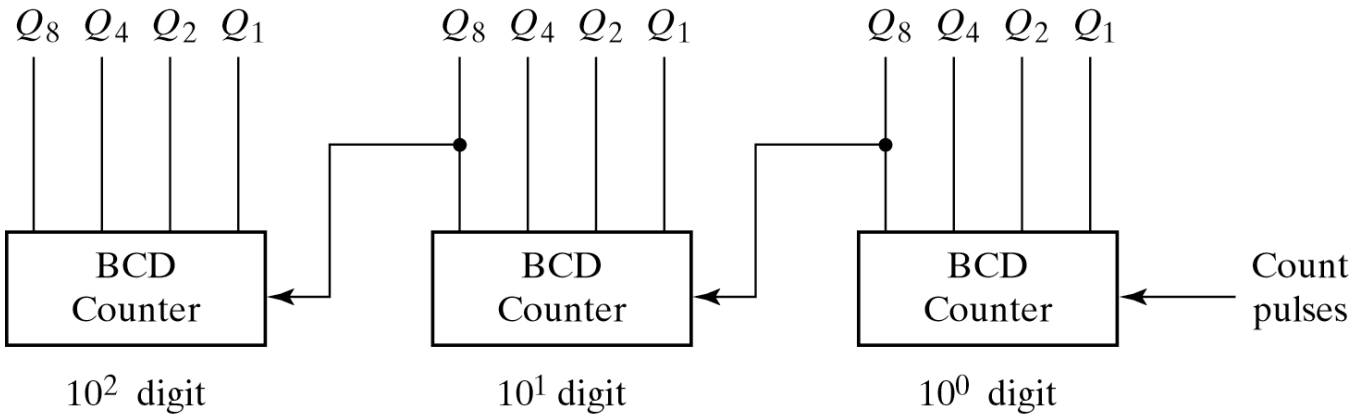


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

- To count from **0** to **999**, We need a three-decade counter.

6.4 SYNCHRONOUS COUNTERS

- Synchronous Counters
  - Binary Counter
  - Up-Down Binary Counter
  - BCD Counter
  - Binary Counter with Parallel Load
  - Other Counter

6.4 SYNCHRONOUS COUNTERS - Binary Counter

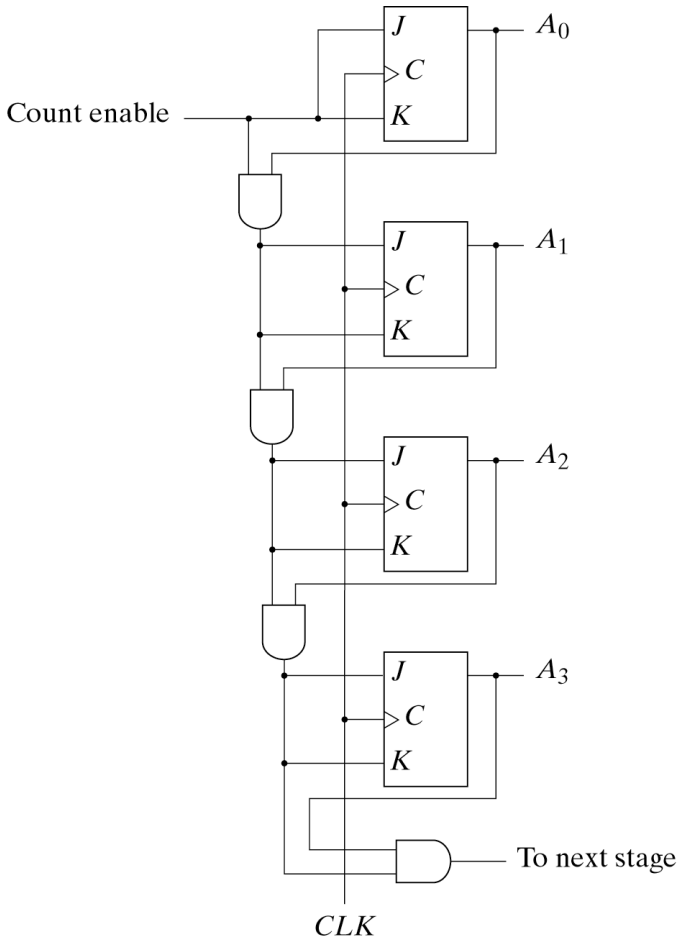


Fig. 6-12 4-Bit Synchronous Binary Counter

- The first stage **A_0** has its **J** and **K** equal to **1** if the counter is enabled .
- The other **J** and **K** inputs are equal to **1** if all previous low-order bits are equal to **1** and the count is enabled.

6.4 SYNCHRONOUS COUNTERS - Up-Down Binary Counter

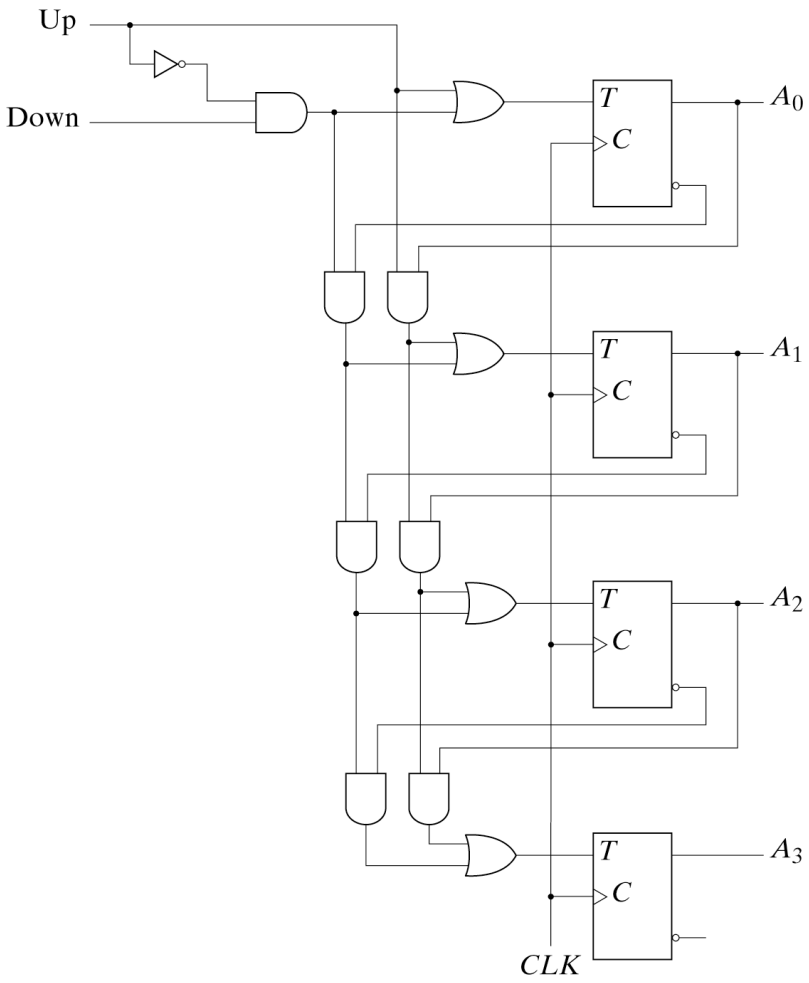


Fig. 6-13 4-Bit Up-Down Binary Counter

- **Up** input control=**1** ;count up (the **T** inputs receive their signals from the values of the previous normal outputs of the flip-flops.)
- **Down** input control=**1**, **up** input control=**0** ; count down
- **Up=down**=0 ;unchanged state
- **Up=down**=1 ;count up

6.4 SYNCHRONOUS COUNTERS - BCD Counter

Table 6-5
State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

6.4 SYNCHRONOUS COUNTERS - Binary Counter with Parallel Load

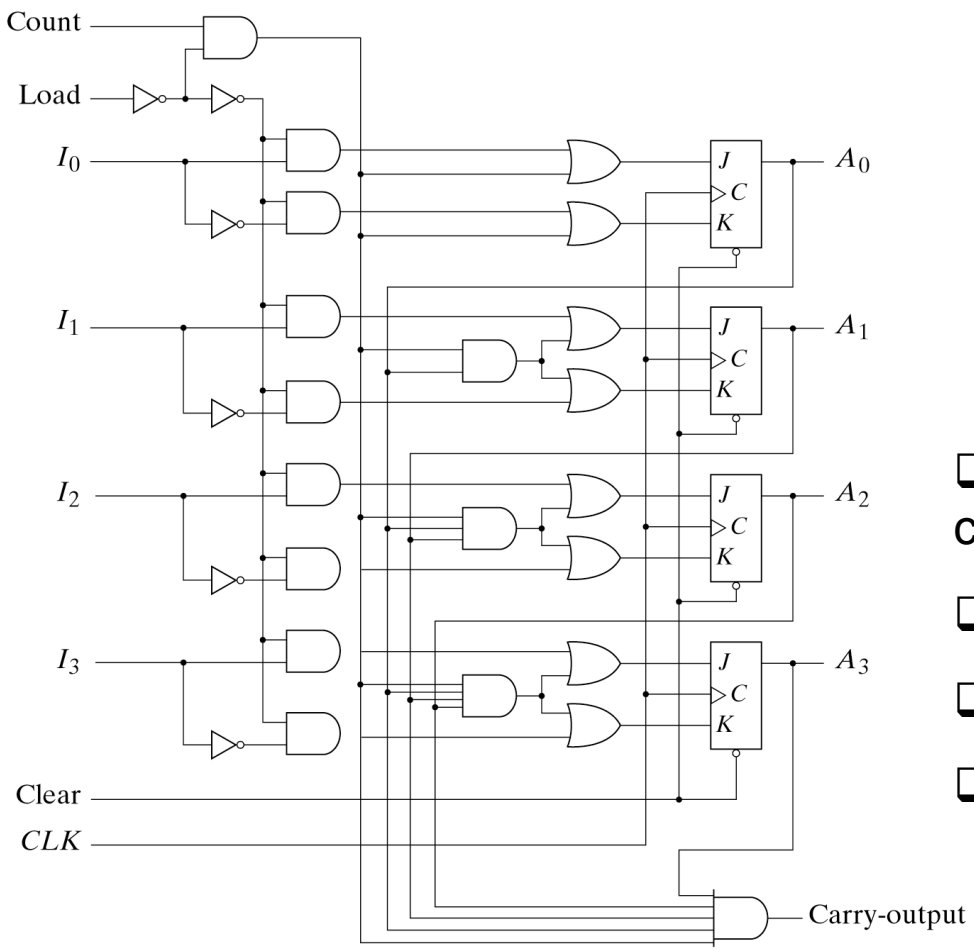


Fig. 6-14 4-Bit Binary Counter with Parallel Load

- Input **load** control=1 ; disables the count sequence ,data transfer
- **Load** =0 and **count**=1 ;count
- **Load**=0 and **count**=0 ;unchanged
- **Carry out**=1(all flip-flop=1)

BCD COUNTER using Binary Counter with Parallel Load

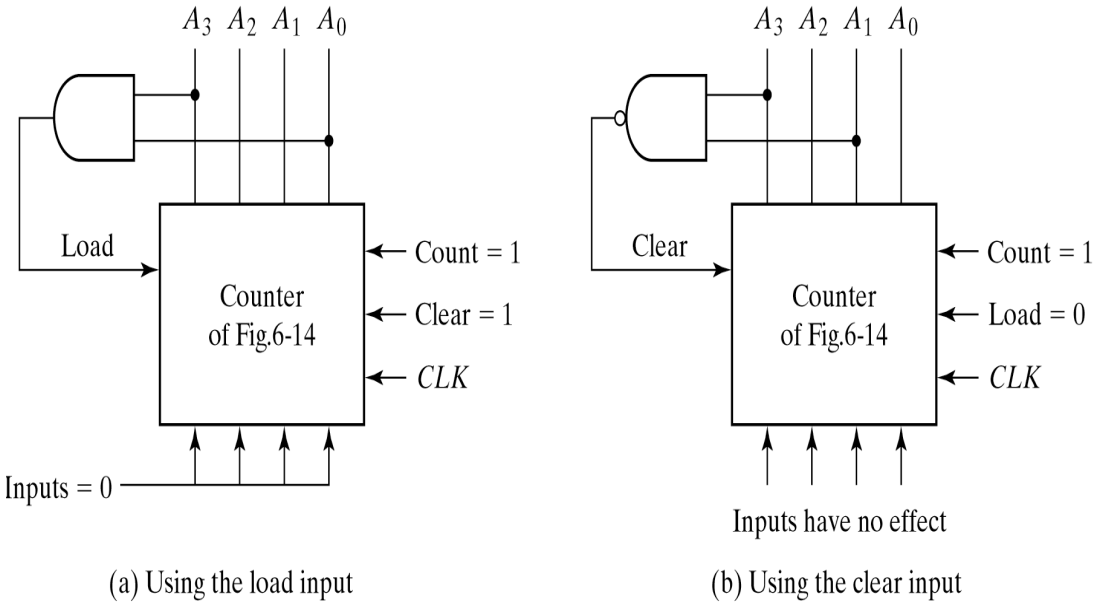


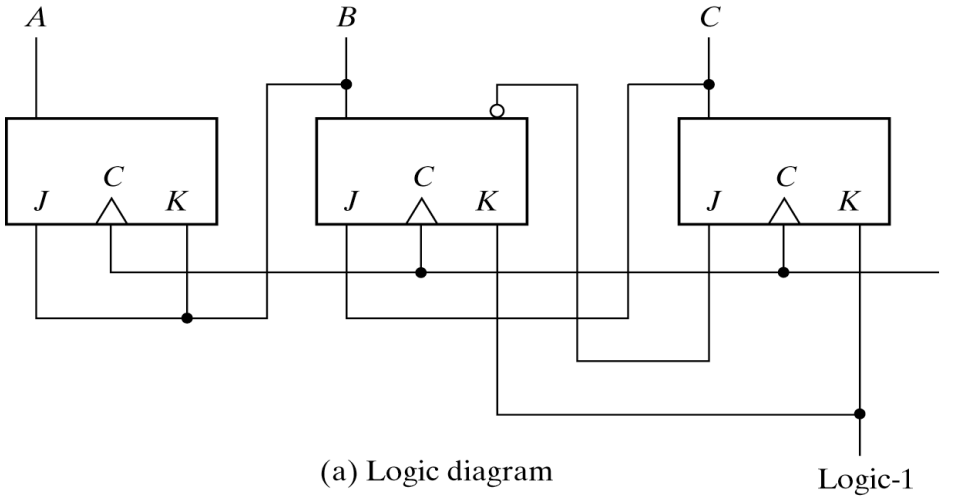
Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

- The **AND** gate detects the occurrence of state **1001 (9)** in the output. In this state, the load input is enabled and all-**0**'s input is loaded into register.
- The **NAND** gate detects the count of **1010(10)** , as soon as this count occurs the register is **cleared.**
- A momentary **spike** occurs in output A2 as the count goes from **1001** to **1010** and immediately to **0000**

6.5 OTHER COUNTERS

- Counter with Unused States
  - Don’t care conditional Counter
- Ring Counter
- Johnson Counter

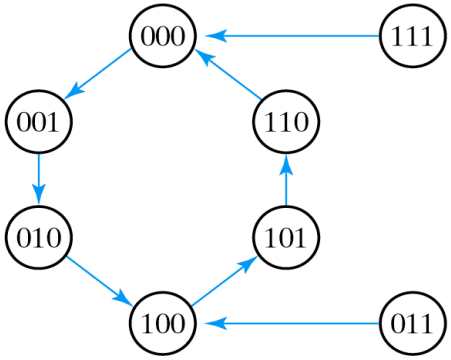
6.5 OTHER COUNTERS - Counter with Unused States



(a) Logic diagram

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = B' \qquad K_C = 1$$

Except **011 ,111**



(b) State diagram

Fig. 6-16 Counter with Unused States

Table 6-7
State Table for Counter

Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

6.5 OTHER COUNTERS - Ring Counter

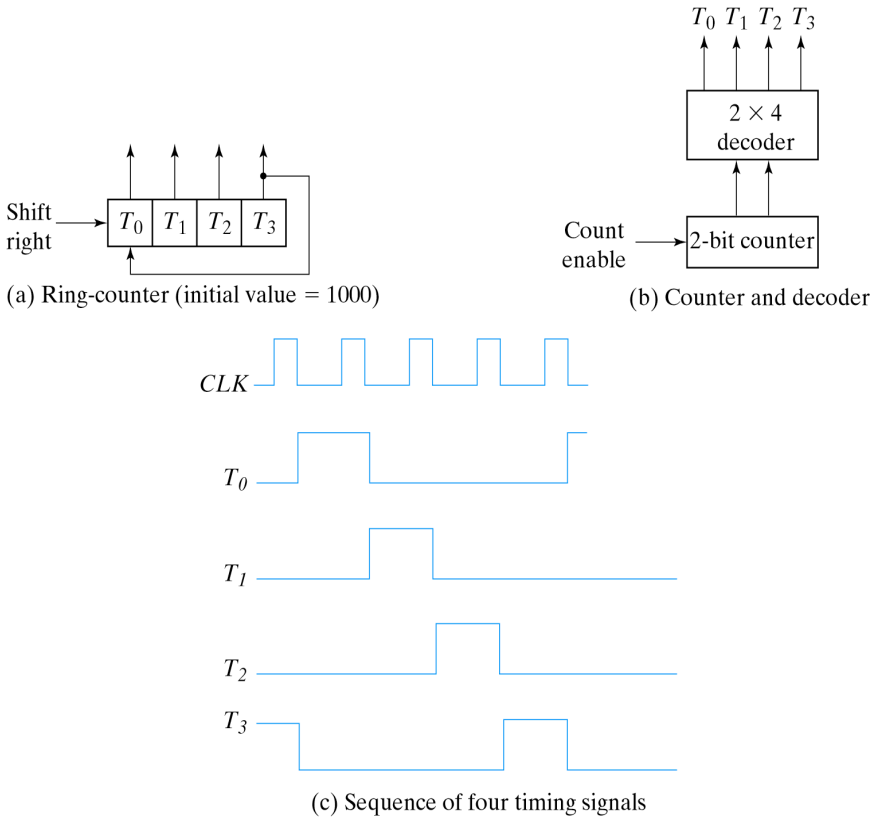
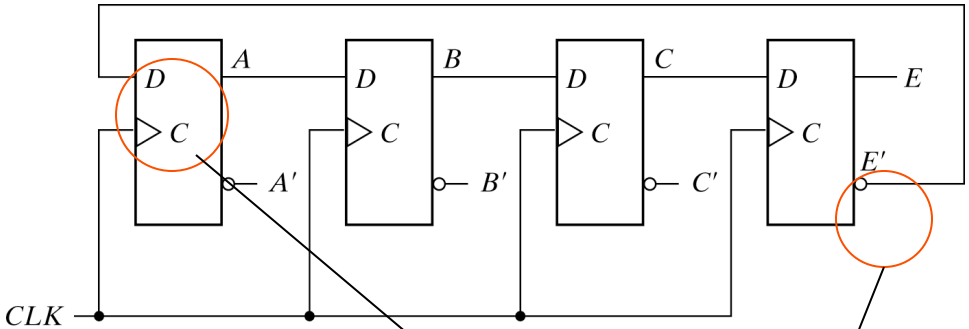


Fig. 6-17 Generation of Timing Signals

- A circular sift register with only one flip-flop being set at any **par ticular time**. ; all others are clear ed.
- **The single bit** is shifted from o ne flip-flop to the other.

6.5 OTHER COUNTERS - Johnson Counter



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(b) Count sequence and required decoding

Fig. 6-18 Construction of a Johnson Counter

- A circular shift register with **the c omplement output of the last flip-flop** connected to the input of the fir st flip-flop.

7.1 INTRODUCTION

- **Memory**

  RAM(random access memory): read and write operation
  ROM(read only memory) : only read operation

- **Programmable Logic Device**

  PLD(programmable logic device)
  PLA(programmable logic array)
  PAL(programmable array logic)
  FPGA(field programmable gate array)

Fig. 7-1 Conventional and Array Logic Diagrams for OR Gate

7.2 RANDOM-ACCESS MEMORY

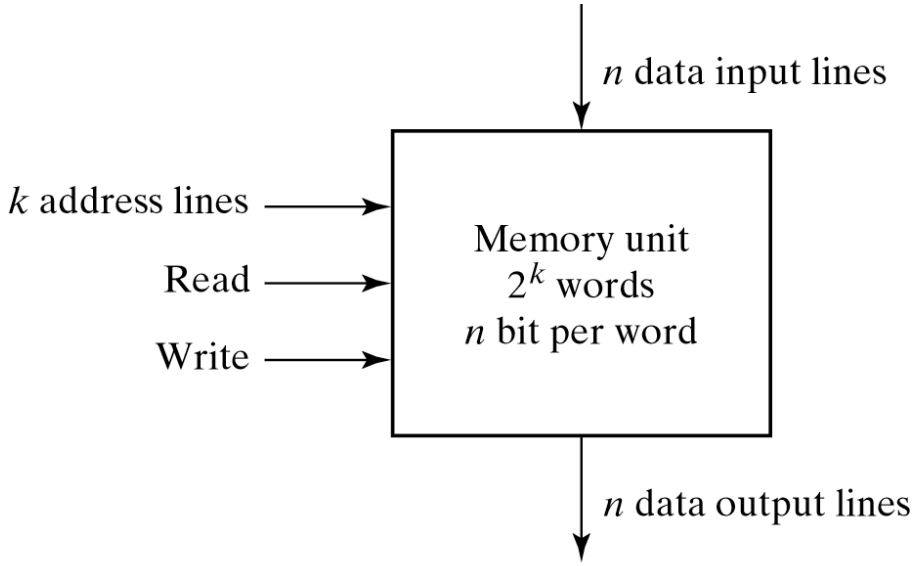


Fig. 7-2 Block Diagram of a Memory Unit

- The address lines select one particular word.
- A decoder inside the memory accepts this address and opens the paths needed to select the word specified.

- 1024x16 Memory

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

The 1K * 16 memory has 10 bits in th e address and 16 bits in each word.

7.2 RANDOM-ACCESS MEMORY - Write and Read Operations

- Write operation

1. Transfer the binary address of the desired word to the address lines.
2. Transfer the data bits that must be stored in memory to the data input lines.
3. Activate the write input

- Read operation

1. Transfer the binary address of the desired word to the address lines.
2. Activate the read input.

Table 7.1
Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

7.2 RANDOM-ACCESS MEMORY - Timing Waveforms

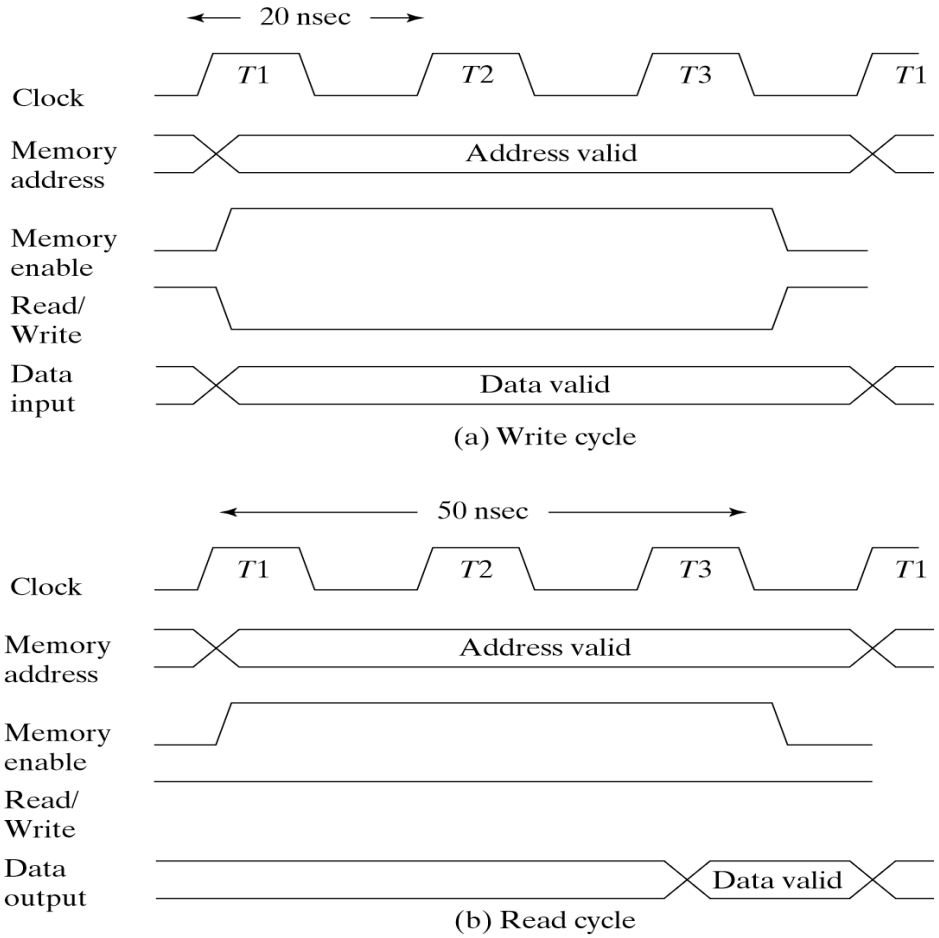


Fig. 7-4 Memory Cycle Timing Waveforms

7.2 RANDOM-ACCESS MEMORY - Types of Memories

- **Random access Memory** -each word occupy one particular location.the acces s time is always the same
- **Sequential access Memory**-information is read out only when the requir ed word has been reached. the access time is variable.
- Volatile –Memory units lose the stored information when power is t urned off.
- **Nonvolatile**-A nonvolatile memory retains its stored information after remova l of power. ex) magnetic disk ,ROM(Read Only Memory)
- **Static RAM**-The stored information remains valid as long as power is a pplied to the unit. Static **RAM** is easier to use and has shorter read and wri te cycles.
- **Dynamic RAM**-The capacitors must be periodically recharged by refreshing th e dynamic memory. Dynamic **RAM** offers reduced power consumption and larger storage capacity

7.3 MEMORY DECODING - Internal Construction

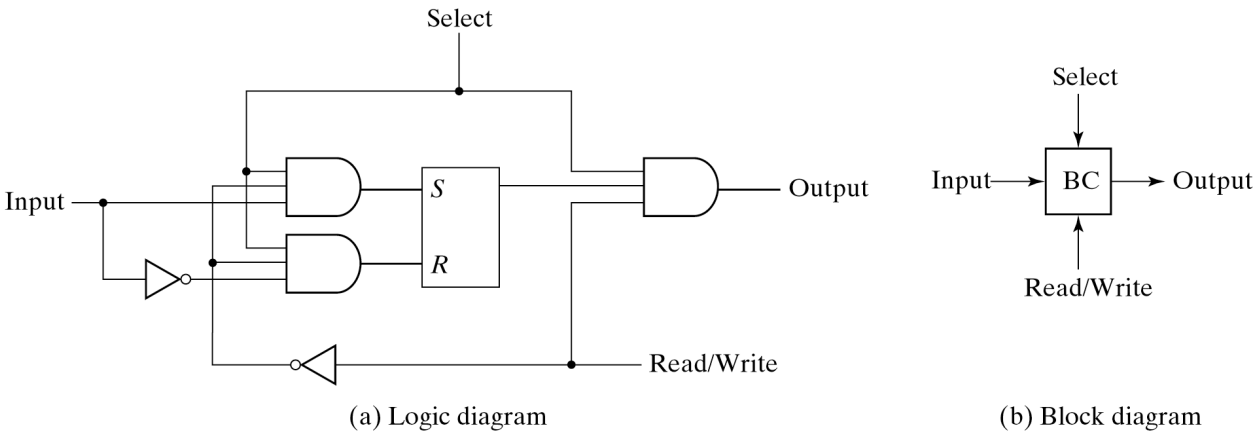


Fig. 7-5 Memory Cell

- The equivalent logic of a binary cell that stores one bit of information
- The binary cell stores one bit in its internal flip-flop
- It has three inputs and one output. The read/write input determines the cell operation when it is selected.

7.3 MEMORY DECODING - Internal Construction

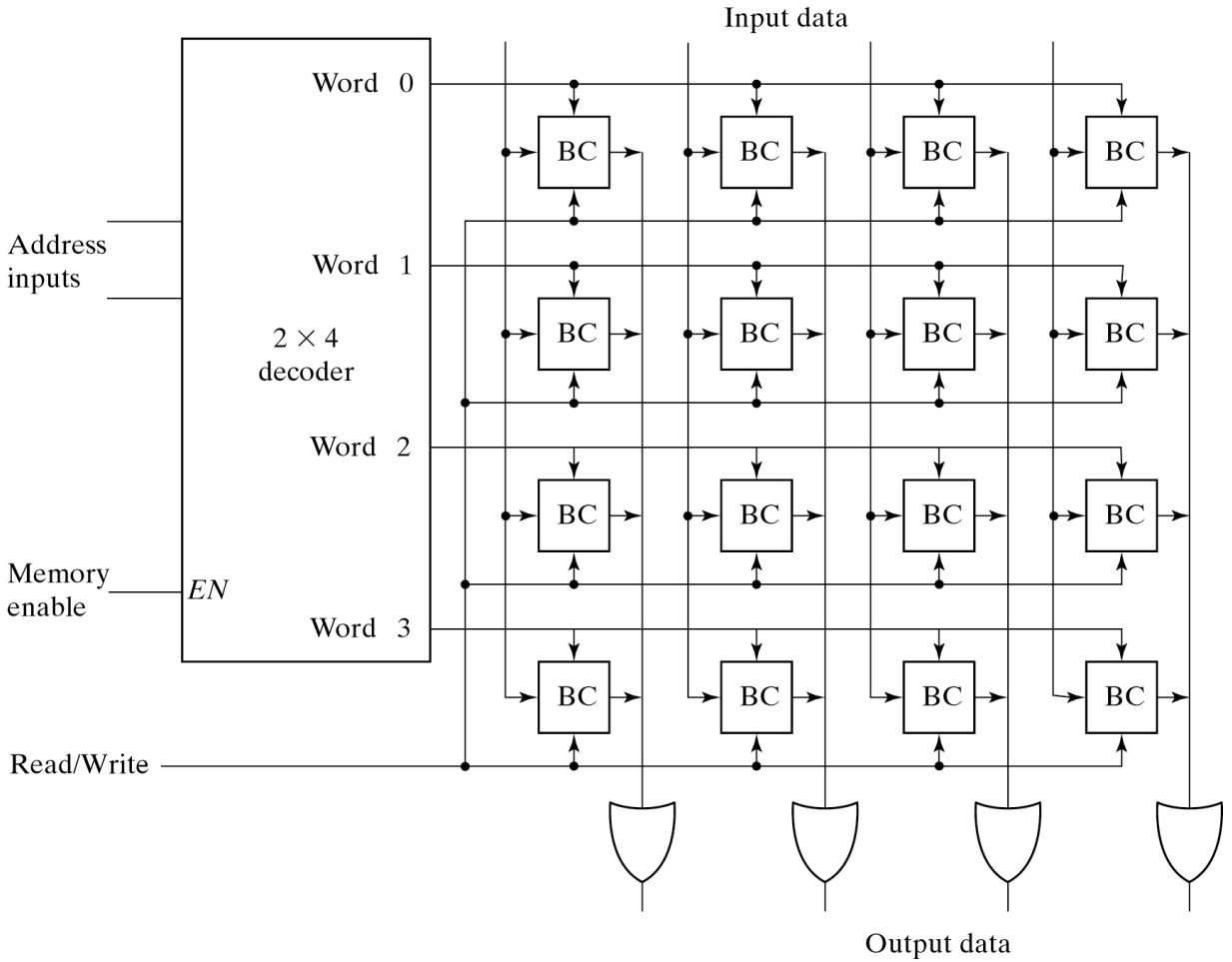


Fig. 7-6 Diagram of a 4 × 4 RAM

7.3 MEMORY DECODING - Internal Construction

- It has 16 binary cells
- Memory enable=0; all outputs of the decoder =0 , none of the m emory words are selected.
- Memory enable=1; one of the four words is selected. The read/ write input determines the operation.
- During the read operation, the four bits of the selected word go t hrough **OR** gates to the output terminals.
- During the write operation, the data available in the input lines are transferred into the four binary cells of the selected word.

7.3 MEMORY DECODING - Coincident Decoding

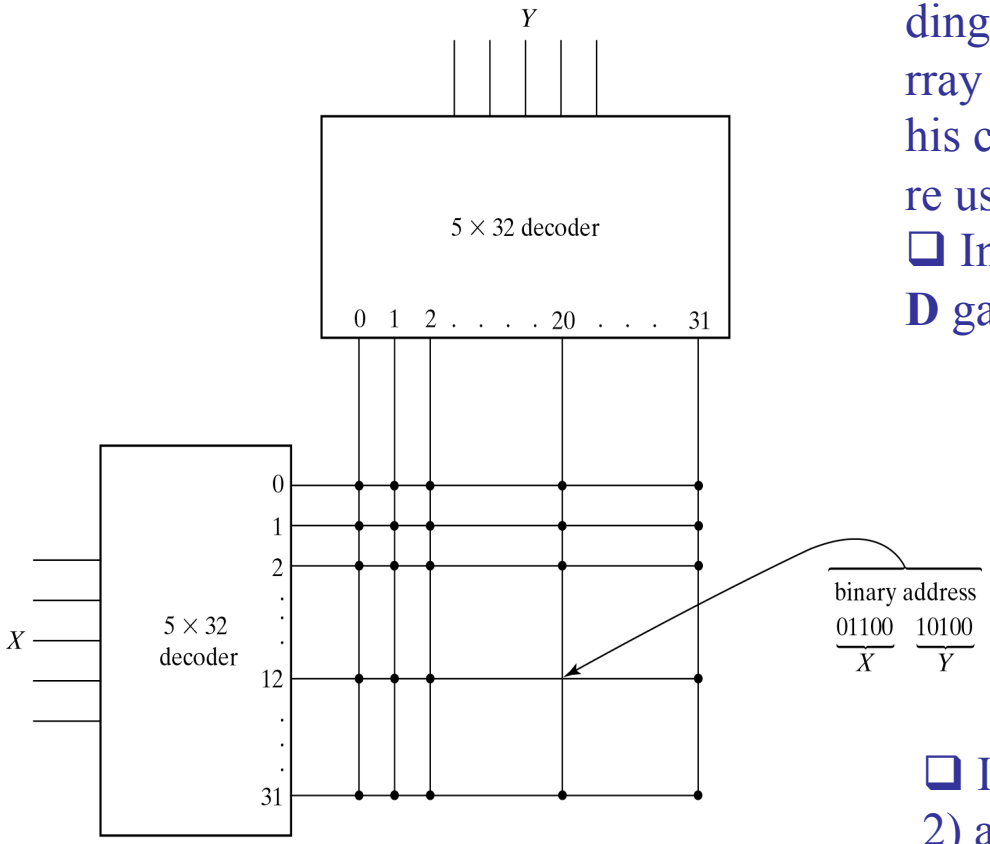


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

- The basic idea in two-dimensional decoding is to arrange the memory cells in an array that is close as possible as square. In this configuration, two k/2-input decoders are used instead of one k-input decoder.
- In the two-decoder case,we need 64 **AND** gates with five inputs in each.

- If the word address is 404, X=01100 (12) and Y=10100(20).

7.3 MEMORY DECODING - Address Multiplexing

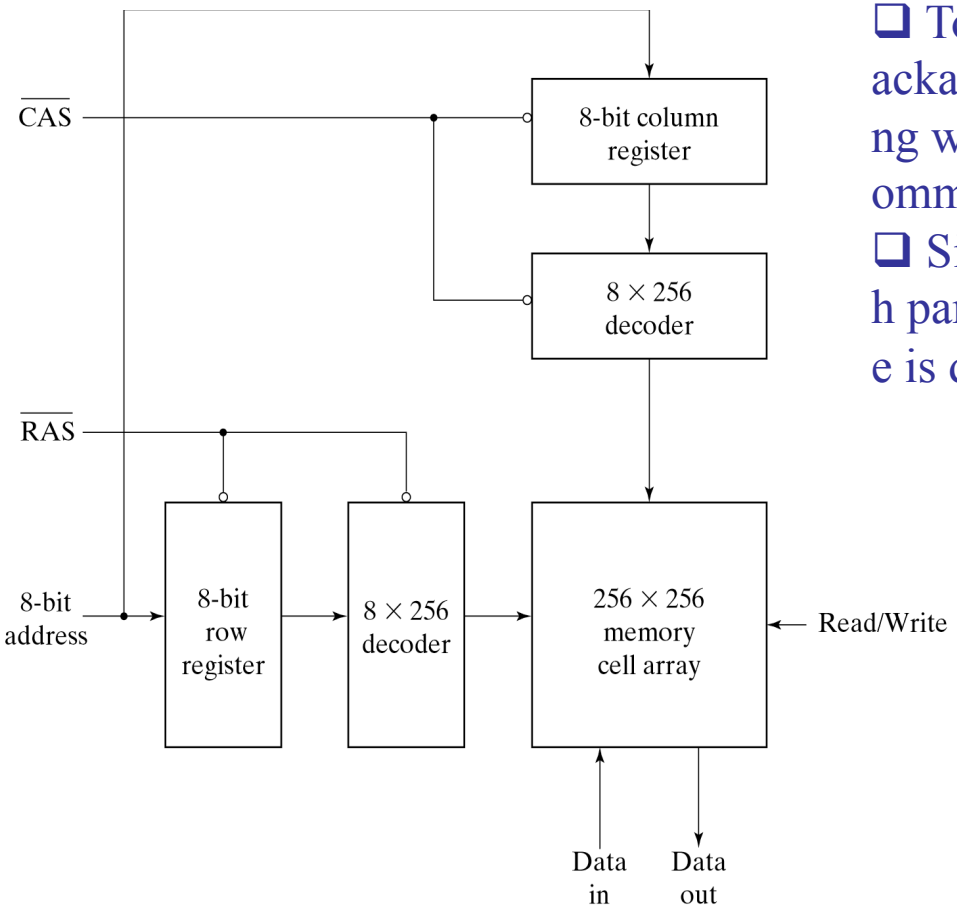


Fig. 7-8 Address Multiplexing for a 64K DRAM

- To reduce the number of pins in the **IC** package, designers utilize address multiplexing whereby one set of address input pins accommodates the address components.
- Since the same set of pins is used for both parts of the address, the size of the package is decreased significantly.

7.4 ERROR DETECTION AND CORRECTION

8-bit data word =>11000100

Bit position 1 2 3 4 5 6 7 8 9 10 11 12

P_1 P_2 1 P_4 1 0 0 P_8 0 1 0 0

P_1=XOR of bits(3,5,7,9,11)=0 (**XOR** performs the odd function.)

P_2=XOR of bits(3,6,7,10,11)=0

P_4=XOR of bits(5,6,7,12)=1

P_8=XOR of bits(9,10,11,12)=1

In memory, **Bit position** 1 2 3 4 5 6 7 8 9 10 11 12

0 0 1 1 1 0 0 1 0 1 0 0

When the 12 bits are read from memory ,The four check bits

C_1=XOR of bits (1,3,5,7,9,11) **C_2=XOR** of bits (2,3,6,7,10,11)

C_3=XOR of bits (4,5,6,7,12) **C_4=XOR** of bits (8,9,10,11,12)

7.4 ERROR DETECTION AND CORRECTION - Hamming code

Since the bits were stored with even parity $C=C_8C_4C_2C_1=0000$ (No error)

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	
	0	0	1	1	1	0	0	1	0	1	0	0	**No error**
	1	0	1	1	1	0	0	1	0	1	0	0	**Error in bit 1**
	0	0	1	1	0	0	0	1	0	1	0	0	**Error in bit 5**

Check bits	C_8	C_4	C_2	C_1
With error in bit 1 :	0	0	0	0
With error in bit 5:	0	1	0	1

The error can be corrected by complementing the corresponding bit.

The **Hamming code** can detect and correct only a single error. Multiple errors are not detected.

To correct a single error and detect double errors ,we **include the additional parity bit.**

If C=0 and P=0 No error occurred

If C=1 and p=1 A single error occurred , which can be corrected

If C=1 and P=0 A double error occurred, which is detected but cannot be corrected

If C=0 and P=1 An error occurred in the P_{13} bit

7.5 READ-ONLY MEMORY

ROM=Decoder + **OR** gates
-permanent binary information is stored.

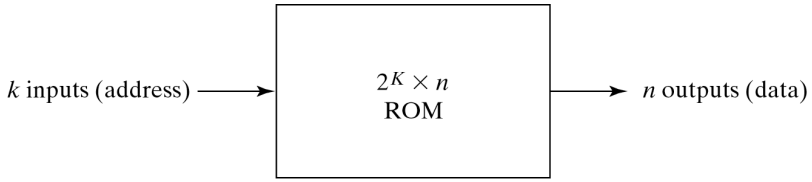


Fig. 7-9 ROM Block Diagram

k input lines and n output lines
The number of output lines ,n= the number of bits per word

7.5 READ-ONLY MEMORY

ROM = **AND** gates connected as a decoder + a number of **OR** gates
5 X 32 decoder has 32 **AND** gates and 5 inverters.

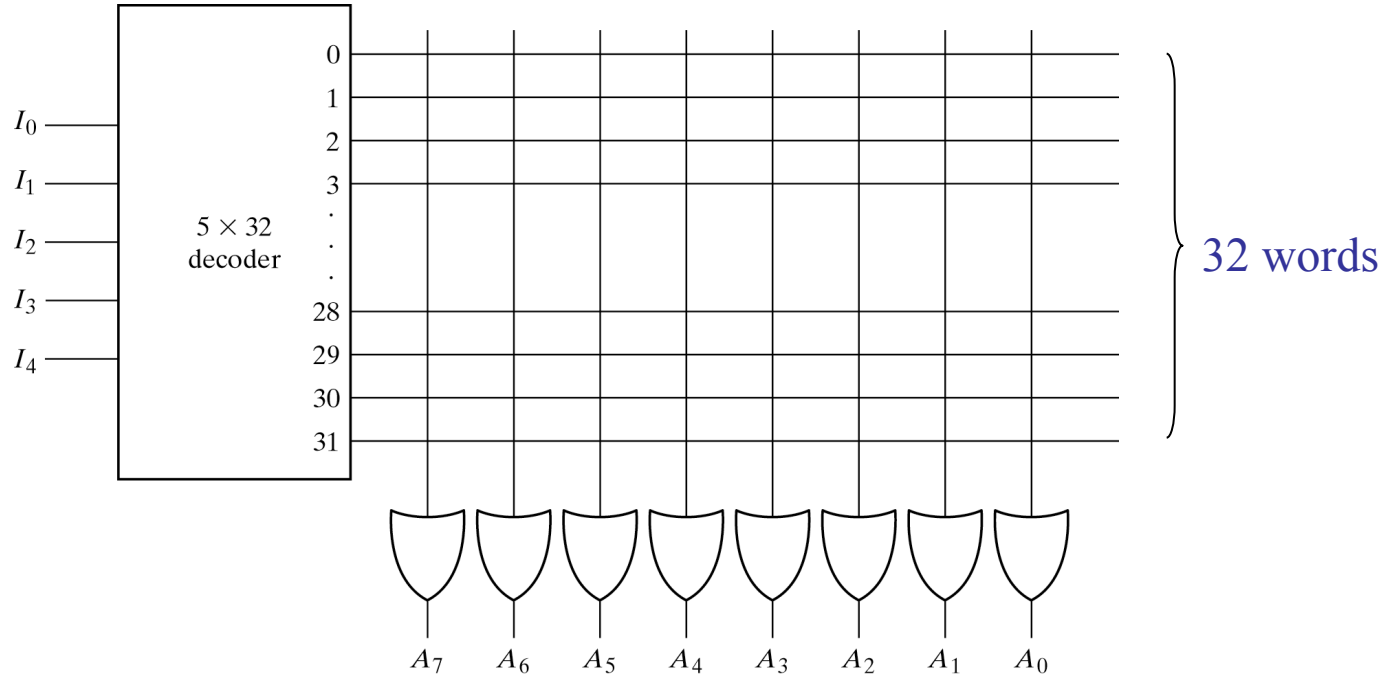


Fig. 7-10 Internal Logic of a 32 × 8 ROM

7.5 READ-ONLY MEMORY

- ROM Truth Table(Partial)

Table 7-3
ROM Truth Table (Partial)

Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

7.5 READ-ONLY MEMORY - Combinational Circuit Implementation

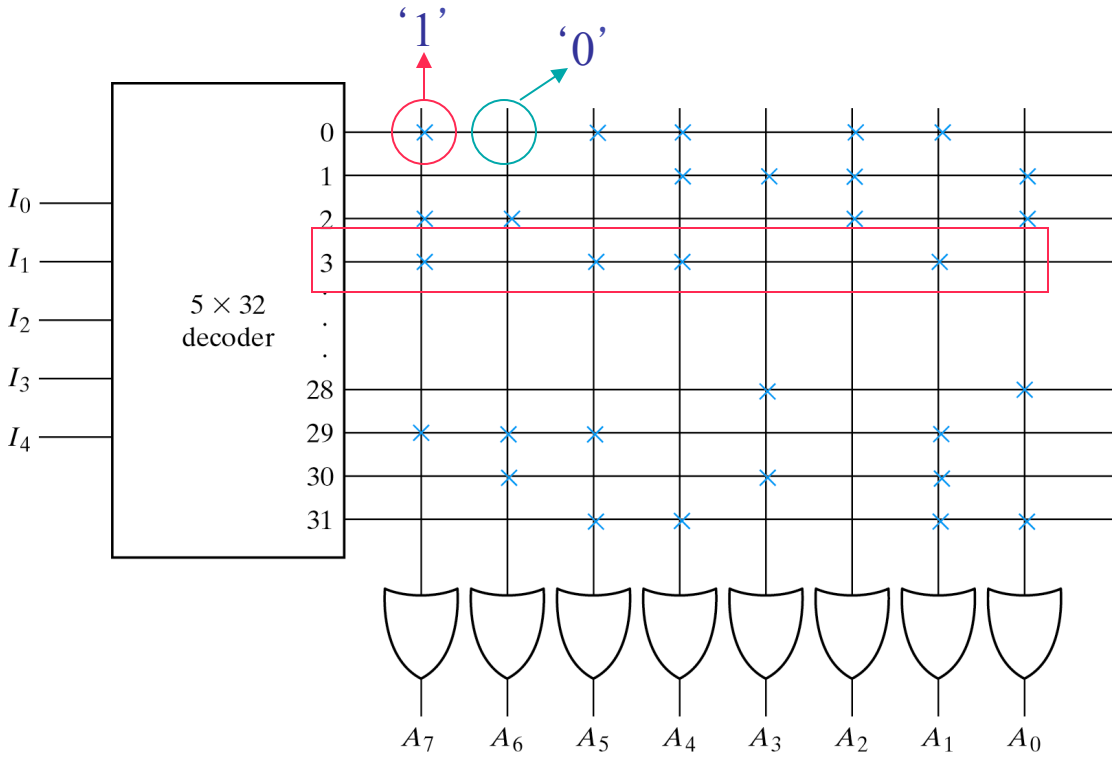


Fig. 7-11 Programming the ROM According to Table 7-3

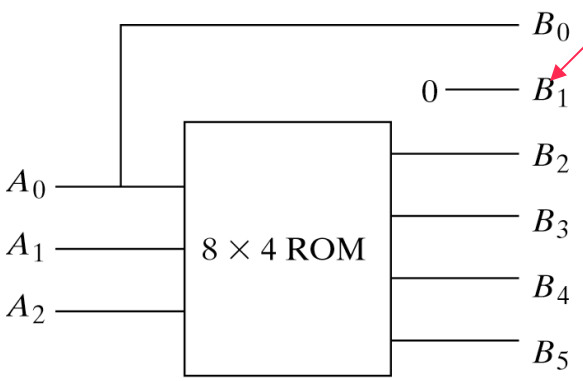
$A_7(I_4,I_3,I_2,I_0)$=Sum of minterms(0,2,3,…,29)

Input->00011(3)
Others-> all '0'
Output->10110010

EXAMPLE 7-1

Table 7-4
Truth Table for Circuit of Example 7-1

Inputs			Outputs						
A_1	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49



(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

Fig. 7-12 ROM Implementation of Example 7-1

7.5 READ-ONLY MEMORY - Types of ROMs

For large quantities**, mask programming** is economical .
For small quantities , programmable read-only memory(**PROM**) is more economical.(all '**1**'s)

PROM- irreversible and permanent
EPROM(Erasable PROM)- can be restructured to the initial value(**UV** light->discharge)
EEPROM(Electrically erasable PROM)- can be erased with electrical signals instead ultra violet light.

7.5 READ-ONLY MEMORY - Combinational PLDs

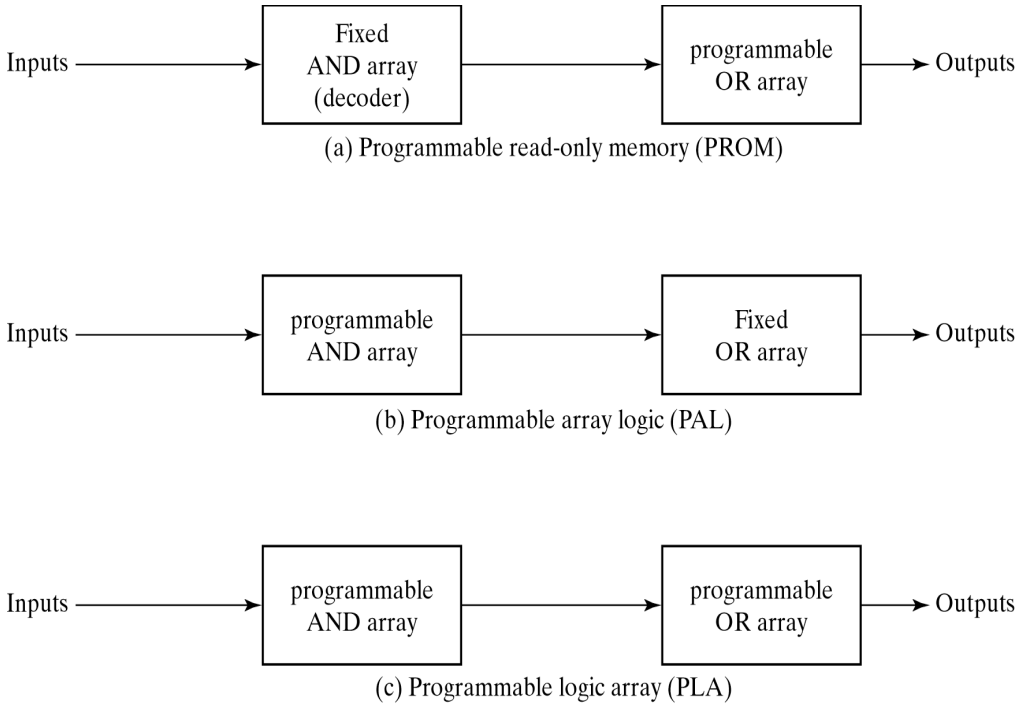


Fig. 7-13 Basic Configuration of Three PLDs

- A combinational **PLD** is an **IC** with programmable gates divided into an **AND** array and an **OR** array to provide an **AND-OR** sum of product implementation.
- Three major types **PLD**s

(a) fixed **AND** array + programmable **OR** array

(b) programmable **AND** array + fixed **OR** array

(c) programmable **AND** array + programmable **OR** array