Review of Digital Design Fundamentals

□ **Bit:** A *bi*nary dig*it*; can have a value of 0 or 1

□ **Logic gate:** A digital circuit that manipulates bits. A logic gate takes one or more bits as input(s) and generates one bit as the output.

■ A logic gate can be represented pictorially by its logic symbol. The function performed by a logic gate can also be expressed *algebraically*, or in terms of a *Truth Table*.

□ **Logic diagram:** A diagram showing an interconnection of logic symbols.

Truth table

- **Truth table:** The truth table gives the input-output relation of a logic gate or logic circuit in tabular form.

- It specifies the output bit(s) for each possible input bit combination.

- A circuit with n binary inputs has 2^n different input combinations.

- A binary value of 0 is sometimes referred to as L (low) or F (false).

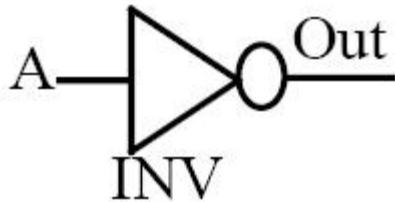- A binary value of 1 is sometimes referred to as H (high) or T (true).

Minterm

- **Minterm:** One specific combination of input bits, out of the 2^n different input combinations.

- A truth table of n binary inputs has 2^n minterms and an output is specified for each minterm.

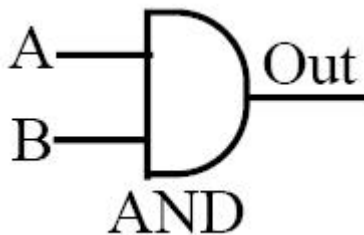# Logic Gates

Logic Symbols:

A	out
0	1
1	0

$\overline{A}, \sim A, \; A', \neg A$

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

$A \cdot B, \; AB, \; A*B, \; A \wedge B$

Logic Gates
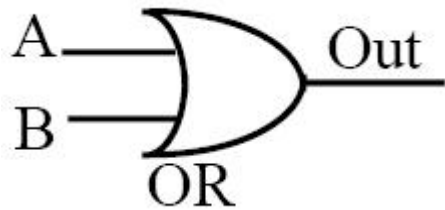
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

A—
B—
OR
Out

A+B, $A \vee B$

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

A—
B—
XOR
Out

$A \oplus B$
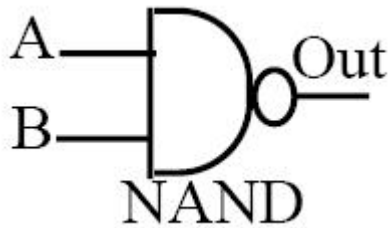
Logic Gates

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

$$\overline{AB},\ \overline{A.B},\ \overline{A*B},\ (A \wedge B)'$$
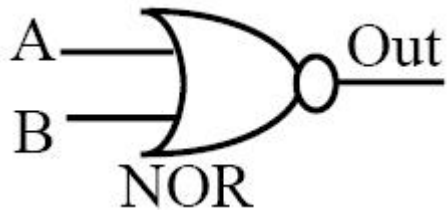
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

$$\overline{A+B},\ (A \vee B)'$$

Logic Gates

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

$$\overline{A \oplus B}$$

Digital Circuit Representation

- The truth table, logic diagram and algebraic expression are three different ways of representing a digital circuit and given one form the other representations of the circuit can be derived.

- The truth table representation of a Boolean function is unique, but the same function may have more than one logic or algebraic representation.

- EXAMPLE: Given the following logic diagram, obtain the corresponding truth table and algebraic expression.

Digital Circuit Representation
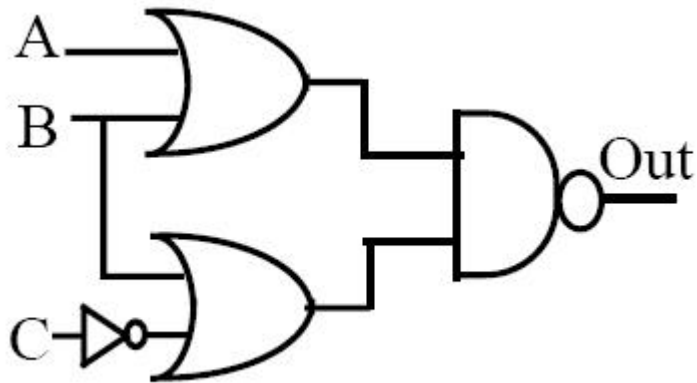
$$\text{Out} = \overline{(A+B)(B+\overline{C})}$$

A B C	Out
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	0

Basic Identities of Boolean algebra

$x + 0 = x$	$x \cdot 0 = 0$
$x + 1 = 1$	$x \cdot 1 = x$
$x + x = x$	$x \cdot x = x$
$x + x' = 1$	$x \cdot x' = 0$
$x + y = y + x$	$x \cdot y = y \cdot x$
$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + y \cdot z = (x + y) \cdot (x + z)$
$(x + y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
$(x')' = x$	

K-Map

- The complexity of a digital circuit depends on the complexity of the corresponding algebraic expression.

- The karnaugh map (k-map) provides a simple straightforward procedure for simplifying Boolean expressions and thereby obtaining simpler digital circuits.

K-Map

- A diagram made up of squares, where each square represents a minterm.

- The output (0 or 1) for a specific minterm is inserted in the corresponding square in the k-map.

- A function with *n* variables has a kmap with 2^n squares.

Properties of k-maps

□ Each row (or column) in the k-map is labelled by one or more bits, representing the values of the corresponding variables for that row (or column).

□ The minterm corresponding to a particular square in the k-map (belonging to the ith column and j^{th} row) is obtained by taking the values of the variables associated with the ith column and j^{th} row.

For example the shaded squares in figure 1 (a) and (b) correspond to minterms 111 and 0110 respectively.

Rules for simplifying k-maps

1. Plot a Boolean function on to a k-map by inserting 1's in those squares where the corresponding minterm has an output of 1.

2. Combine *adjacent* 1's into groups such that:

    i) a group contains only 1's

    ii) the number of squares in a group is a power of 2

    iii) the group is not part of a single larger group

Rules for simplifying k-maps

3. Keep choosing additional groups until all the 1's in the k-map are covered i.e. each 1 is part of at least one group. Choose the groups in such a way that the total number of groups needed to cover all the 1's is minimized.

4. Obtain an algebraic product term for each group.

5. Obtain the final solution by a logical OR of all the terms from step 4.

Some definitions:
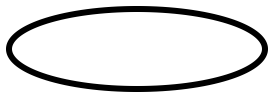
- Implicant: A group of 2^k adjacent 1's in a k-map.

- Prime Implicant (PI): An implicant which is not completely covered by a single larger implicant.

- Essential Prime Implicant: A prime implicant where at least one minterm is not covered by any other prime implicant.

Examples

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

AB

C	00	01	11	10
0	1	1	0	0
1	1	0	0	1

A`B`

F=A`C`+B`C
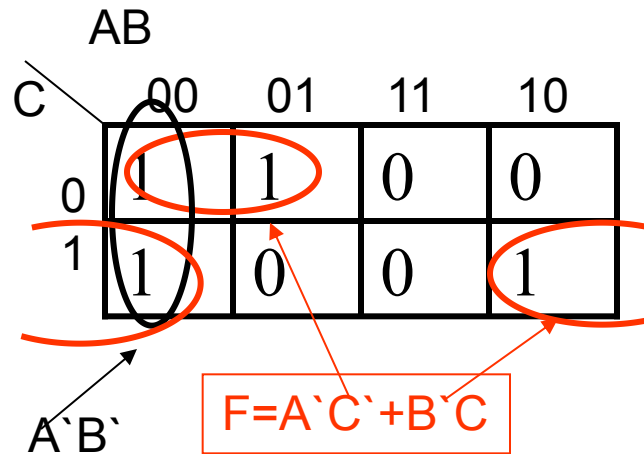
Essential PI   Non-Essential PI

Example

A B C D	F
0 0 0 0	0
0 0 0 1	0
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	0
0 1 1 0	1
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	1
1 0 1 1	1
1 1 0 0	1
1 1 0 1	1
1 1 1 0	0
1 1 1 1	1

AB

CD

	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	1	1
10	1	1	0	1

F=AD+AB`+BC`D`+A`CD`
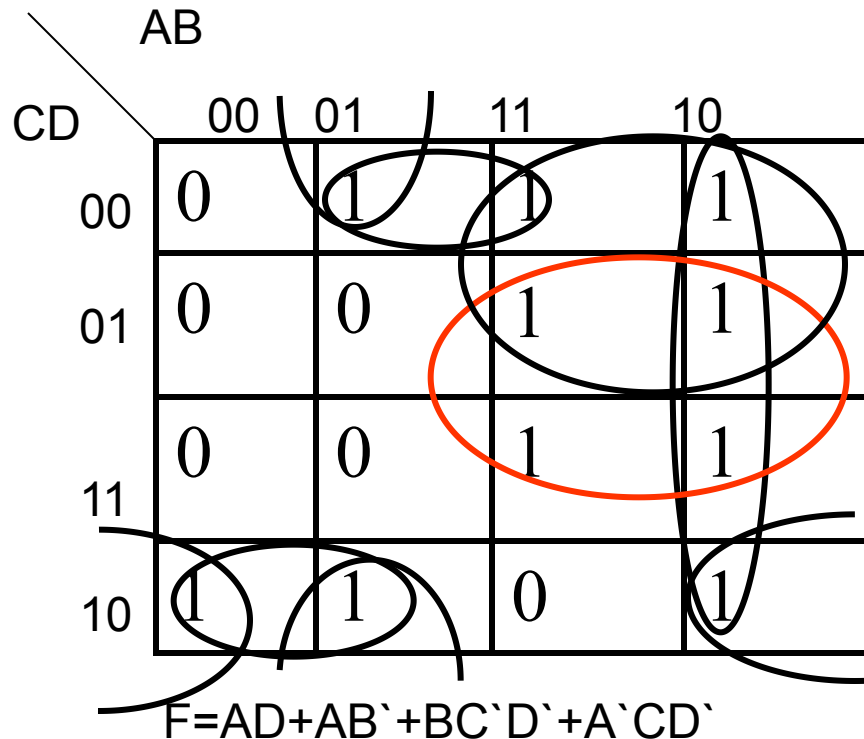
OR

F=AD+ AC`+ A`BD`+B`CD`

Non-Essential PI

Essential PI

Don't Cares

- A truth table can be expressed in compact form by simply specifying the minterms for which the output is 1.

- For some functions, there may be certain input conditions for which we don't care what output is. This situation is represented by putting a "d" in the output for the corresponding minterms.

- Don't care outputs may be treated as either a 0 or a 1, in order to obtain a minimized circuit.

- All minterms which are not in the given list(s) are assumed to have an output of 0.

Function with Don't cares

$$F(A,B,C) = \Sigma(0,2,6). \ d(A,B,C.) = \Sigma(1,3,5)$$

a) F has three inputs A,B, and C.

b) The outputs corresponding to minterms   0,2,6 are 1.

c) The outputs corresponding to minters 1,3,5 are don't cares

d) The outputs for all remaining minterms (4 and 7) are 0.

Examples:

- Obtain a minimized SOP expression for the following functions. Also, list all the prime implicants and indicate which ones are essential.

$$(i) \ F(A,B,C) = \Sigma(0,2,6), \ d(A,B,C,) = \Sigma(1,3,5).$$

F

AB

	00	01	11	10
C				
0	1	1	1	0
1	d	d	0	d

Prime Implicants
A'(essential)
BC'(essential)

F= A'+BC'

(ii) $F(A,B,C,D) = \Sigma(1,2,4,12), \ d(A,B,C,D) = \Sigma(3,5,9,11).$

AB

CD

	00	01	11	10
00	0	1	1	0
01	1	d	0	d
11	d	0	0	d
10	1	0	0	0

F = B'D + BC'D' + A'B'C

Prime Implicants

BC'D'(Essential)
A'B'C(Essential)
B'D
A'BC'
A'C'D

Product of sums (POS)

- A POS expression for a function F can be obtained by grouping the 0,s in the k-map for F and then

(i) Obtain a SOP expression for F' and complement it OR

(ii) Directly obtain the POS expression by examining the selected groups.

(i) $F(A,B,C,D) = \Sigma(0,2,4\text{-}6,8\text{-}10)$, $d(A,B,C,D) = \Sigma(1,7)$.

Prime Implicants of F'
AB (Essential)
CD(Essential)
A'B'D

CD \ AB	00	01	11	10
00	1	1	0	1
01	d	1	0	1
11	0	d	0	0
10	1	1	0	1

F'=AB+CD
F''=(AB+CD)'
F=(AB)'.(CD)'
F=(A'+B').(C'+D')

Final POS expression

The same final expression can also be obtained by directly examining the groups.

(ii) $F(A,B,C,D) = \Sigma(0\text{-}2,8\text{-}10,12\text{-}15)$.

Prime Implicants of F'
A'B (Essential)
B'CD(Essential)
A'CD

CD \ AB	00	01	11	10
00	1	0	1	1
01	1	0	1	1
11	0	0	1	0
10	1	0	1	1

F'=A'B+B'CD

F''=(A'B+B'CD)'
F=(A'B)'.(B'CD)'
F=(A+B').(B+C'+D')

Final POS expression

The same final expression can also be obtained by directly examining the groups.

Multi-output functions

- A multi-output function is treated similar to a single output function,

- A separate k-map and Boolean expression needs to be derived for each of the outputs.

- Example: full adder (FA) circuit which has 3 inputs - the 2 bits (A and B) to be added and a carry in (C_{in}) and has 2 outputs a sum (S) and a carry out to the next stage (C_{out}).

Full Adder

A	B	Cin		Cout	S
0	0	0		0	0
0	0	1		0	1
0	1	0		0	1
0	1	1		1	0
1	0	0		0	1
1	0	1		1	0
1	1	0		1	0
1	1	1		1	1

S=A'B'C + A'BC'+AB'C'+ABC

S= A $\oplus$ B $\oplus$ C

C_{out}=AB+BC+AC

S   AB

C	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Cout   AB

C	00	01	11	10
0	0	0	1	0
1	0	1	1	1

flip-flops and Sequential Circuits

- **Combinational circuits**: Digital circuits where the output depends only on the current inputs. They consist of a interconnection of logic gates.

- **Sequential circuits**: Output depends on the previous output state as well as the current inputs.

flip-flops and Sequential Circuits

- **Flip-Flops**: Basic storage element, capable of storing previous state (0 or 1). A flip-flop can store 1 bit of information and is the basic building block of sequential circuits.

flip-flops and Sequential Circuits

- **<u>Edge-triggered flip-flop</u>**: State changes occur only during a positive (0 to1) or negative (1 to 0) clock transition. The corresponding flip-flops are called *positive* or *negative* edge-triggered flip-flops respectively.
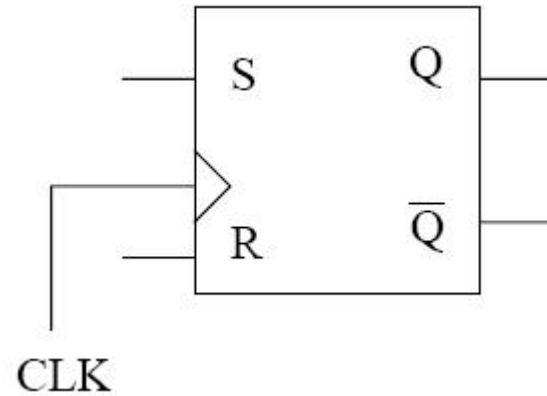
  The output (*Next State*) of a flip-flop depends on
  - Present State and
  - Current inputs and is described in a characteristics table.

<u>Characteristic table</u>: Specifies the next state, based on present state and current inputs.

SR-flip-flop
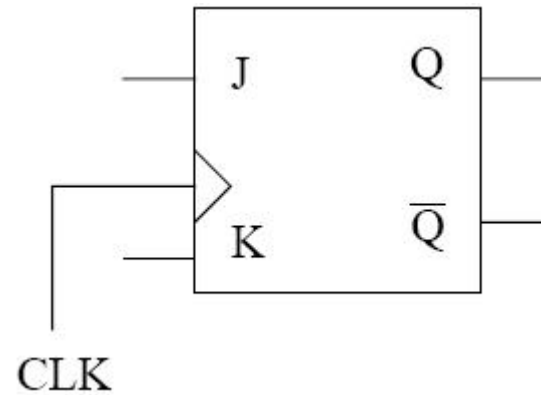
Input SR	Present StateQ(t)	Next State Q(t+1)
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	Indeterminate	

S   Q

R $\overline{Q}$

CLK

JK-flip-flop

Input S R	Present StateQ(t)	Next State Q(t+1)
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	1
11	1	0

CLK

D-flip-flop

Input D	Present State Q(t)	Next State Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

CLK
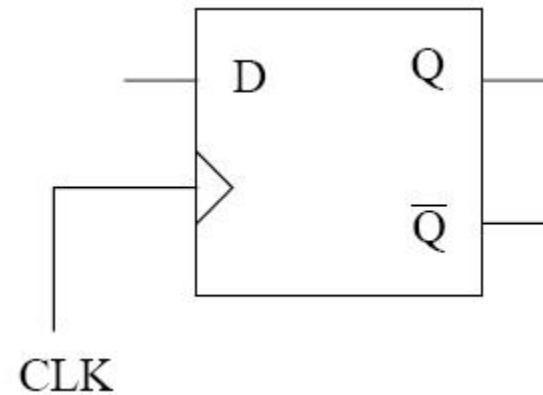
T-flip-flop

Input T	Present StateQ(t)	Next State Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

T

Q

$\overline{Q}$

CLK
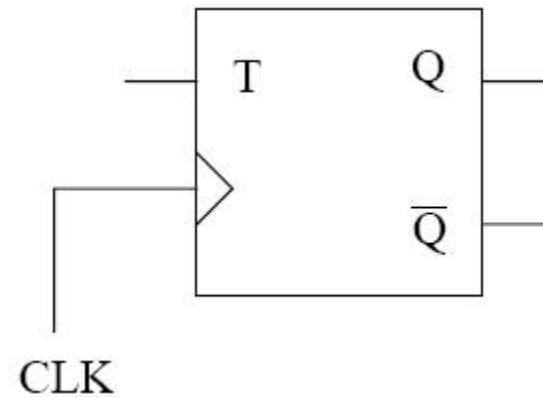
Sequential Circuit Analysis

Complete the state table, for the following circuit.

present state input			flip-flop inputs		next state	
A	B	X	D_A	D_B	A	B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1
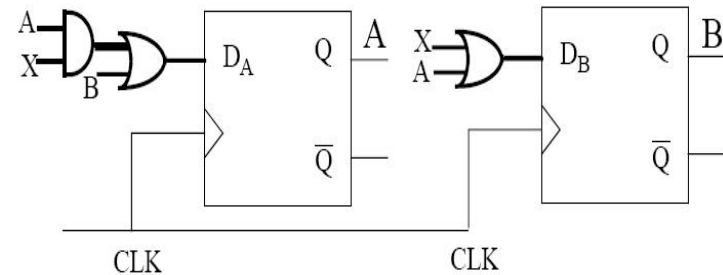
$D_A = AX + B; \; D_B = X + A;$

Sequential Circuit Analysis

present state input A B X			flip-flop inputs T_A T_B		next state A	B
0 0	0		0	0	0	0
0 0	1		0	1	0	1
0 1	0		1	0	1	1
0 1	1		1	1	1	0
1 0	0		0	1	1	1
1 0	1		1	1	0	1
1 1	0		1	1	0	0
1 1	1		1	1	0	0
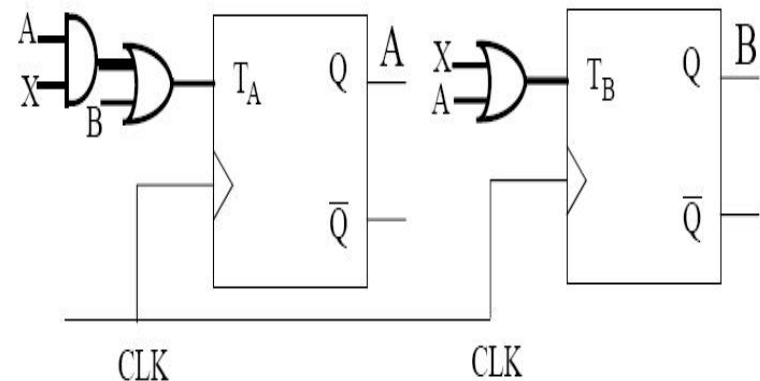
(ii) Complete the state table If T-flip-flops are used.

. $T_A = AX + B$; $T_B = X + A$;

SR-flip-flop

JK-flip-flop

Excitation Tables

Present State Q(t)	Next state Q(t+1)	Inputs S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

Present State Q(t)	Next state Q(t+1)	Inputs J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

D-flip-flop

T-flip-flop

Excitation Tables

Present State Q(t)	Next state Q(t+1)	Inputs D
0	0	0
0	1	1
1	0	0
1	1	1

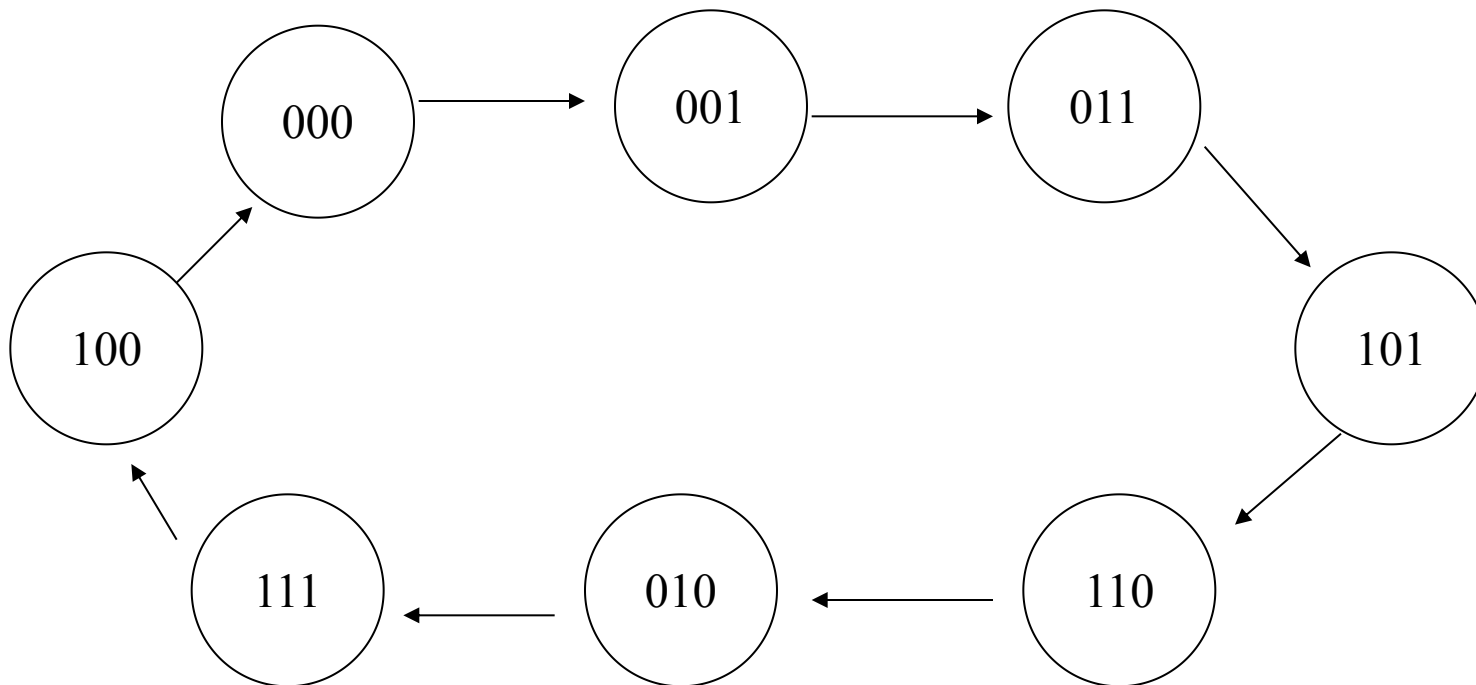| Present State Q(t)	Next state Q(t+1)	Inputs T
0	0	0
0	1	1
1	0	1
1	1	0

Sequential Circuit Design

- The excitation tables give the required inputs to the flip-flop for a specific change of state. The steps to be followed in the design of sequential circuits are as follows:

1. Draw the state diagram from the problem specification
2. Choose the type of flip-flop to be used.
3. Fill in the excitation table of the circuit using the selected flip-flops.
4. Obtain k-maps for each flip-flop input.
5. Simplify the k-maps to obtain Boolean expressions for each flip-flop input.
6. Draw the logic diagram of the circuit (if required).

Example

□ Design a sequential circuit going through the following sequence of states: 0 -> 1 -> 3 -> 5 -> 6 -> 2 -> 7 -> 4 -> 0

(a) Draw the state diagram of the circuit.

Excitation Table

Present state A B C			Next State A B C			flip-flop inputs T_A T_B T_C		
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	0
1	0	0	0	0	0	1	0	0
1	0	1	1	1	0	0	1	1
1	1	0	0	1	0	1	0	0
1	1	1	1	0	0	0	1	1

K-Map

T_A

AB

C	00	01	11	10
0	0	1	1	1
1	0	1	0	0

$$T_A = A'B + AC'$$

T_B

AB

C	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$$T_B = C$$

T_C

AB

C	00	01	11	10
0	1	1	0	0
1	0	0	1	1

$$T_C = A'C' + AC$$

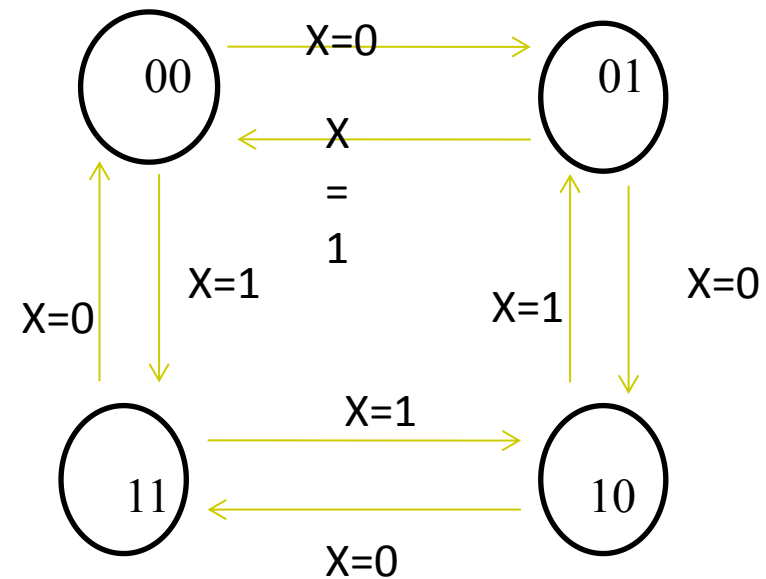| Present state A B C			Next State A B C			flip-flop inputs T_A T_B T_C		
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	0
1	0	0	0	0	0	1	0	0
1	0	1	1	1	0	0	1	1
1	1	0	0	1	0	1	0	0
1	1	1	1	0	0	0	1	1

More examples of sequential circuit synthesis.

- ☐ Design a two bit sequential circuit with an external input x, such that the circuit counts up when x=0 and counts down when x=1. Use D flip-flops.
  - ■ a) Draw state diagram

Excitation Table

| Present state | | | Next State | | flip-flop Inputs | |
A	B	x	A	B	DA	DB
0	0	0	0	1	0	1
0	0	1	1	1	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	1	0	0	0	0	0
1	1	1	1	0	1	0

K-maps

DA

AB   00   01   11   10

x

	00	01	11	10
0	0	1	0	1
1	1	0	1	0

DA= A'B'x+ A'Bx'+AB'x'+ABx

DB

AB   00   01   11   10

x

	00	01	11	10
0	1	0	0	1
1	1	0	0	1

DB= B'

Present state			Next State		flip-flop Inputs	
A	B	x	A	B	DA	DB
0	0	0	0	1	0	1
0	0	1	1	1	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	1	0	0	0	0	0
1	1	1	1	0	1	0

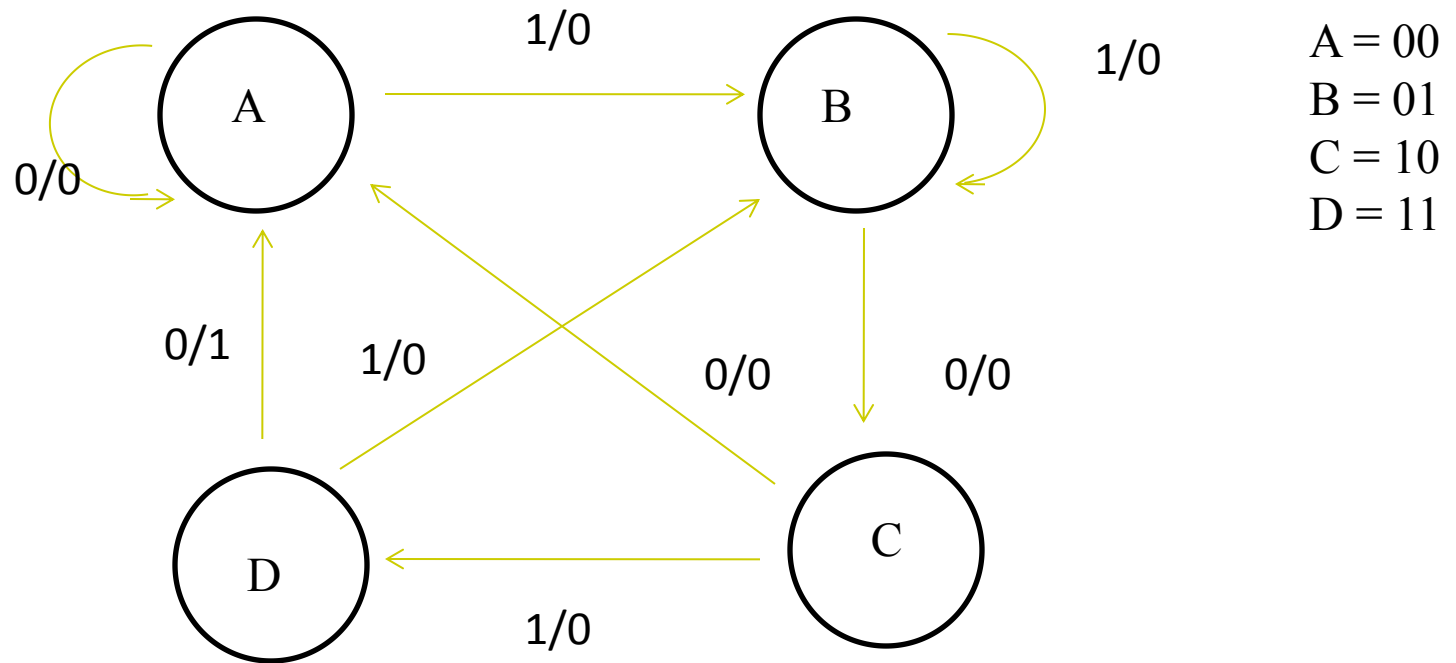# Draw the logic diagram.
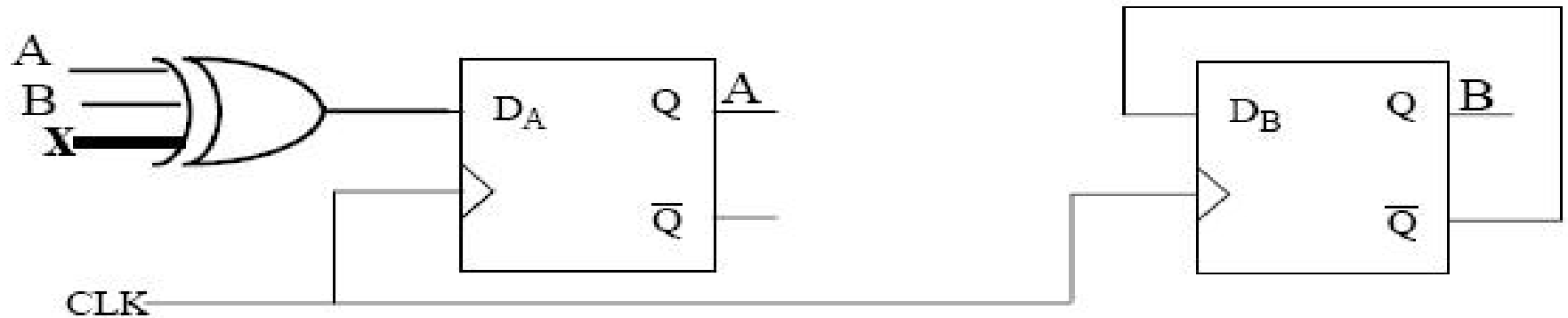
2. Design a sequential circuit, using T flip-flops, that has the following state diagram

$A = 00$
$B = 01$
$C = 10$
$D = 11$

Fill in the excitation table corresponding to the above sequential circuit, using T flip-flops.

Present State			Next State		flip-flop Inputs		Output
Q1	Q2	X	Q1	Q2	T1	T2	
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0
1	0	1	1	1	0	1	0
1	1	0	0	0	1	1	1
1	1	1	0	1	1	0	0
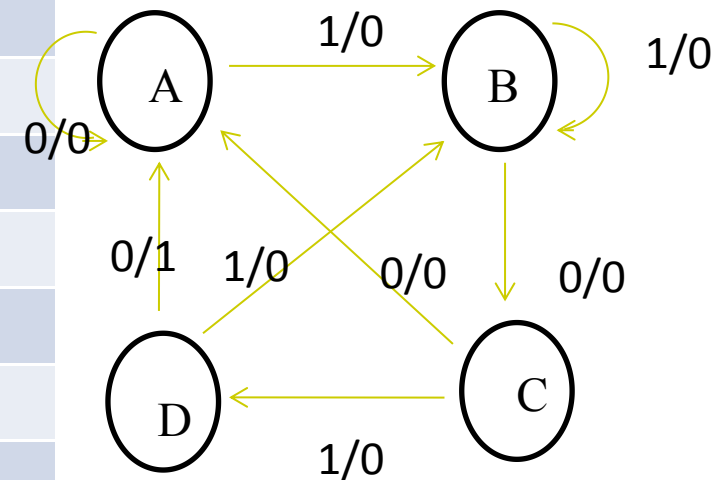
Fill in the K-maps and obtain a Boolean expression for each flip-flop input.

T1

Q1Q2

x

	00	01	11	10
0	0	1	1	1
1	0	0	1	0

T1= Q1Q2 + Q1x' + Q2x'

T2

Q1Q2

x

	00	01	11	10
0	0	1	1	0
1	1	0	0	1

T2= Q2x' + Q2'x

Present State		Next State	flip-flop Inputs		Output
Q1 Q2 X		Q1	T1	T2	
0 0 0	0	0	0	0	0
0 0 1	0	1	0	1	0
0 1 0	1	0	1	1	0
0 1 1	0	1	0	0	0
1 0 0	0	0	1	0	0
1 0 1	1	1	0	1	0
1 1 0	0	0	1	1	1
1 1 1	0	1	1	0	0

Draw the logic diagram.
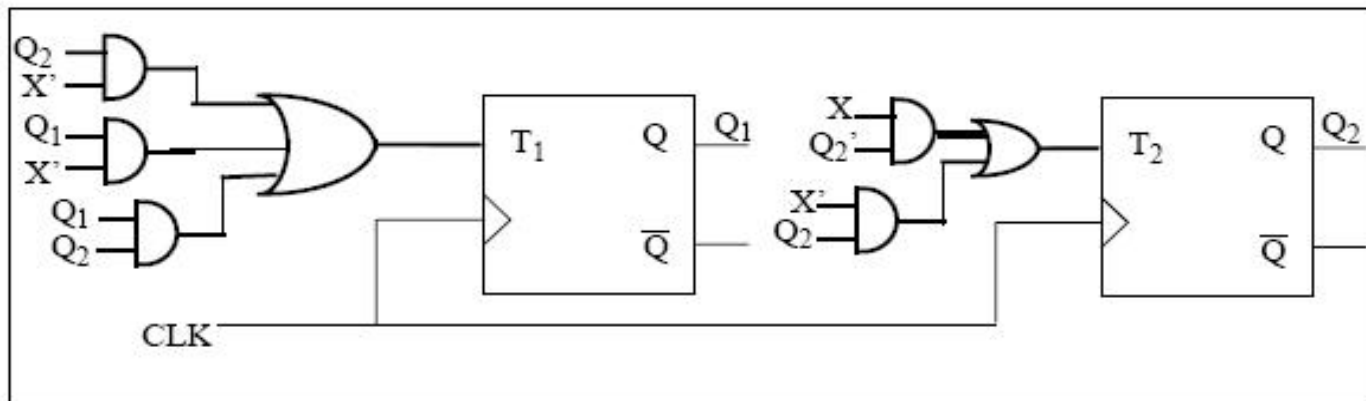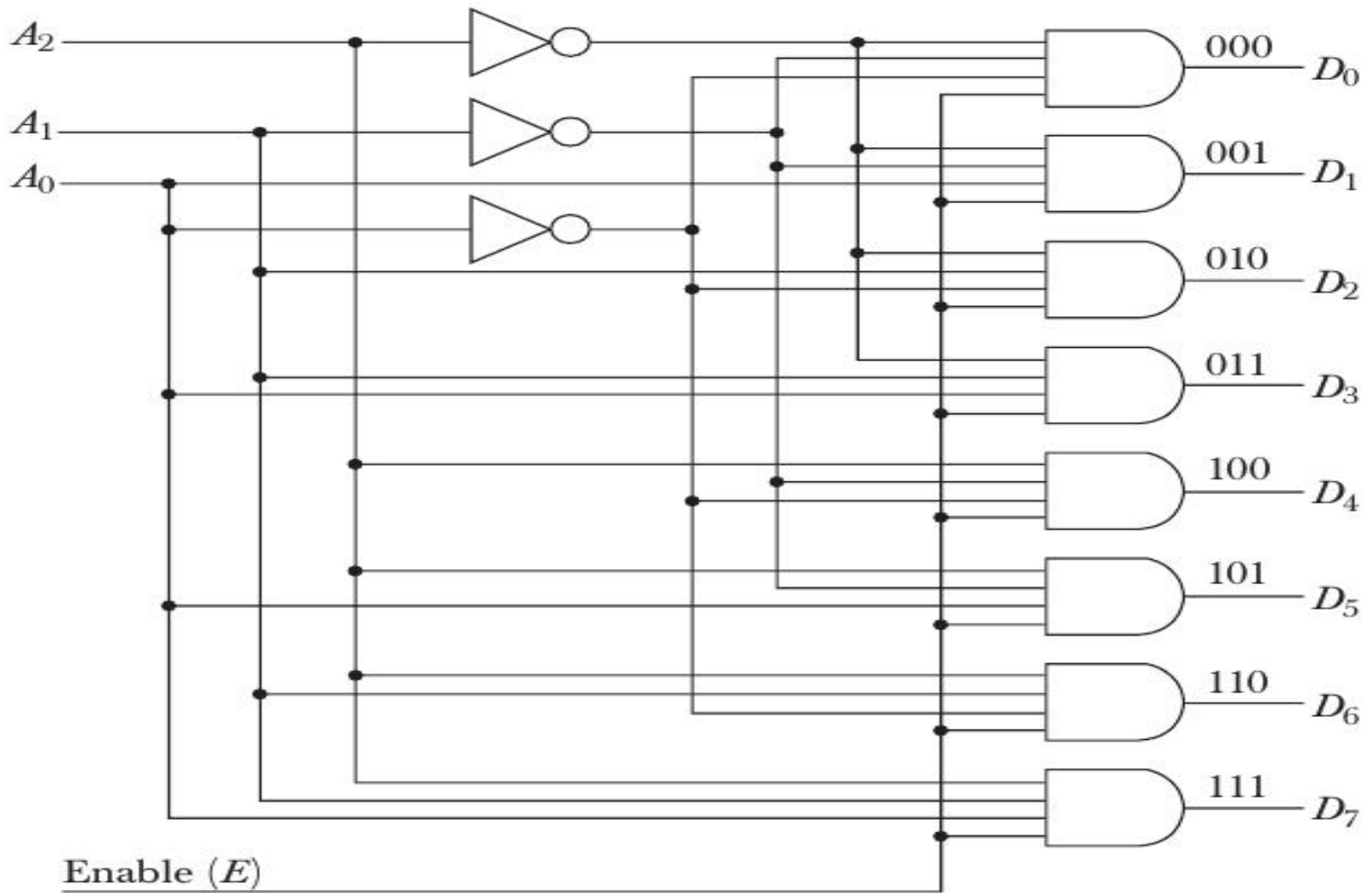
Decoder

- Converts binary information from n inputs to upto 2^n outputs.
- If some input combinations are unused, the number of outputs may be less.
- For each input combination, only one output is *active* and all other outputs are *inactive*.
- The decoders presented here are called n-to-m-line decoders, where m<=2^n.

A decoder may also have an enable input (E), as shown above.

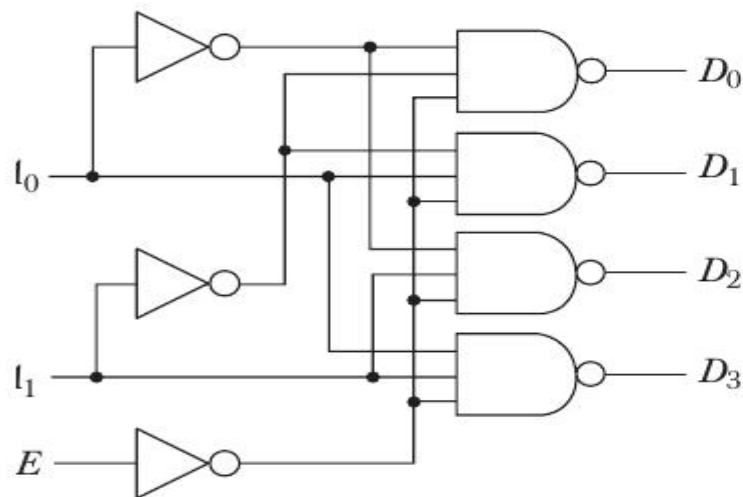**TABLE 2-1** Truth Table for 3-to-8-Line Decoder

Enable	Inputs			Outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

If E=0, all outputs are 0
If E=1, decoder functions normally
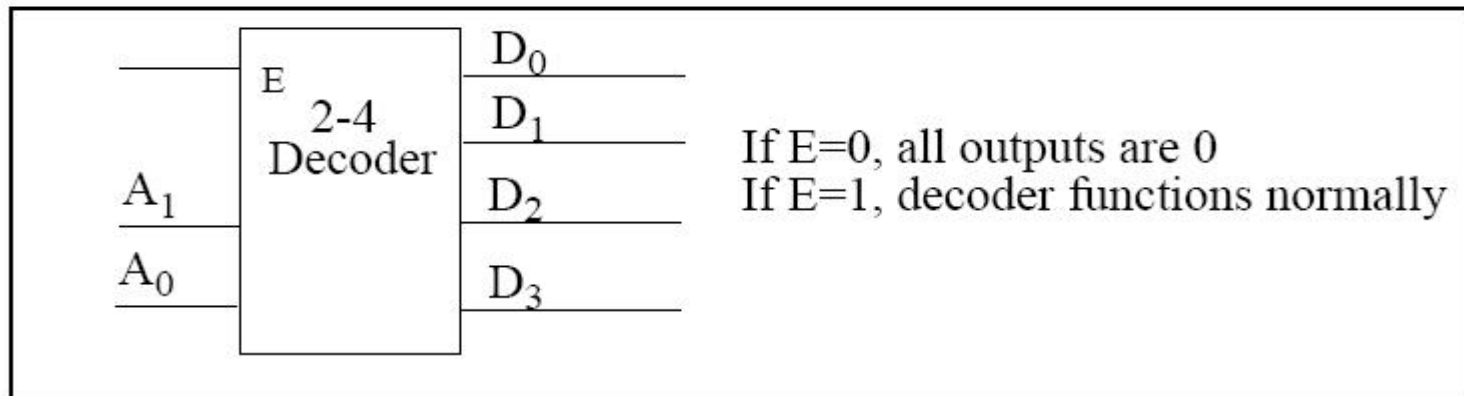
Figure 2-3   2-to-4-line decoder with NAND gates.

(a) Logic diagram

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

(b) Truth table

Decoder Expansion:

- Smaller decoders can be combined to form a larger one.
- Example: Construct a 3-8 decoder using 2-4 decoders.
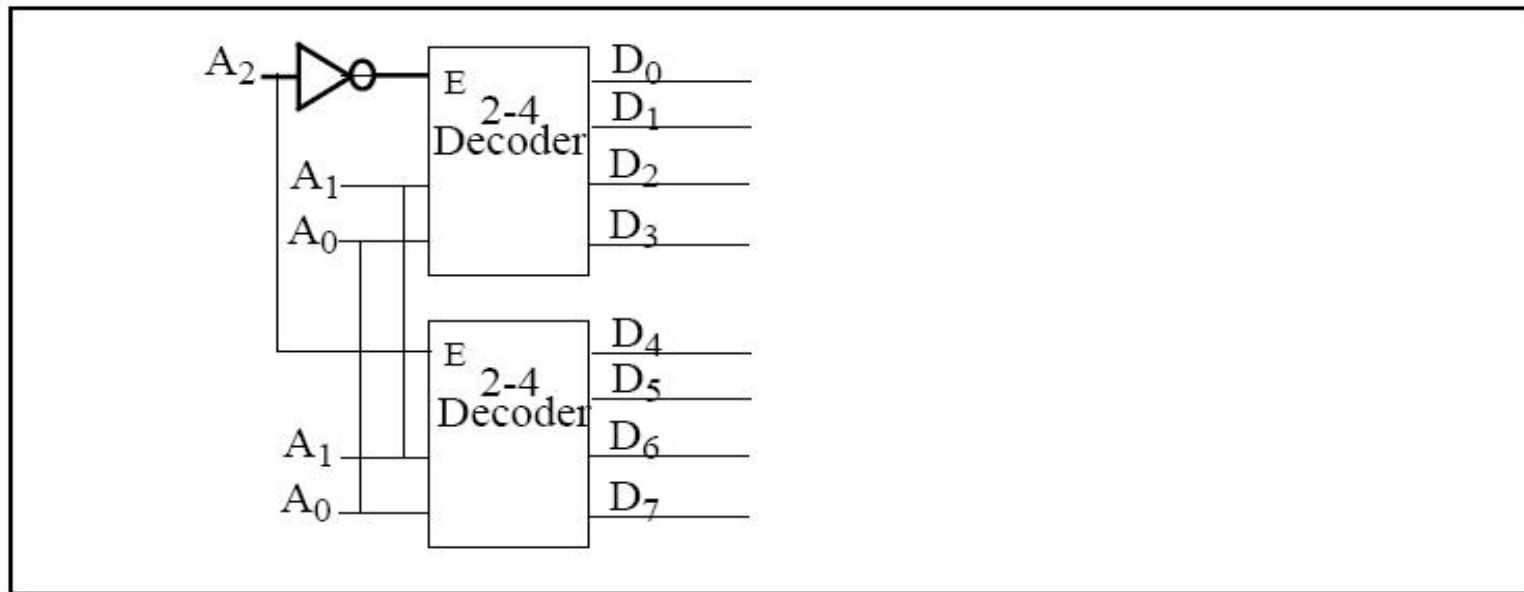
Encoders

- An encoder performs the inverse operation of a decoder.

- It has (up to) 2^n inputs and n outputs.

- The output lines generate the binary code corresponding to the input value.

- Only one input line can be high at any given time.

TABLE 2-2 Truth Table for Octal-to-Binary Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

Multiplexer

- A combinational circuit which takes information from one of 2^n input lines and transfers it to a single output line.

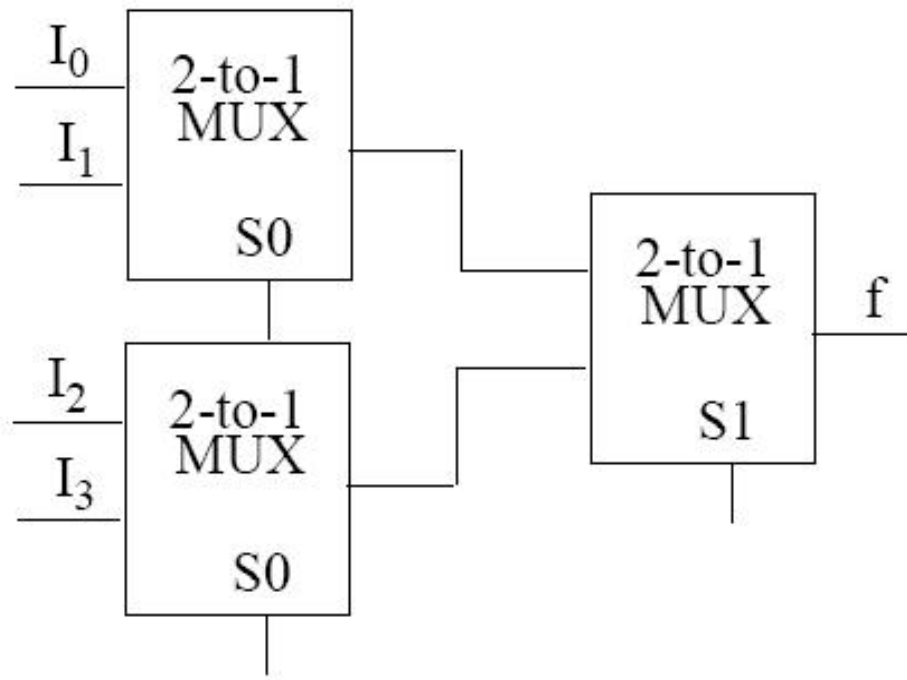- The particular input line chosen is determined by n select lines.

Figure: A 4-to-1 Multiplexer
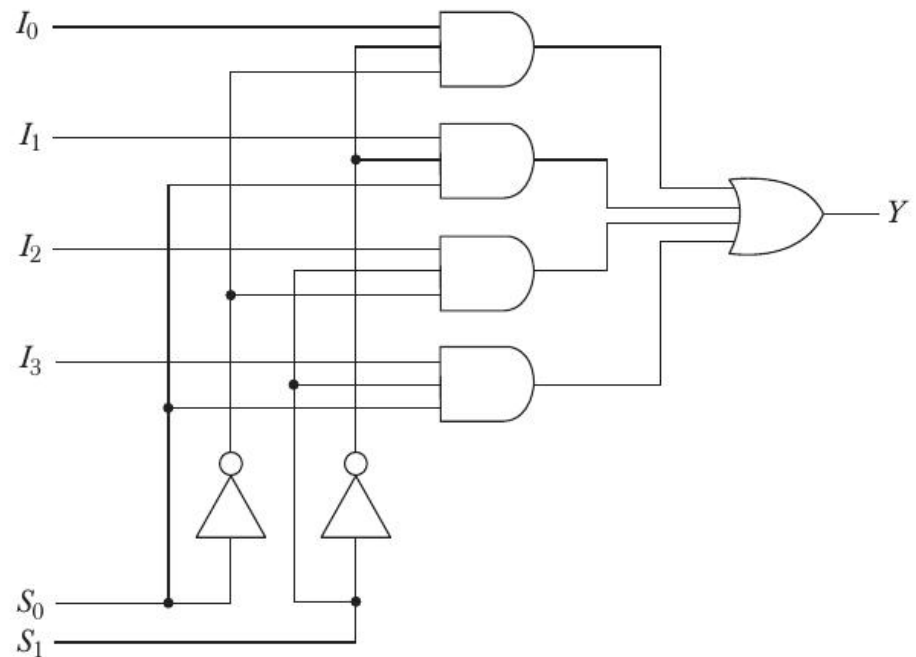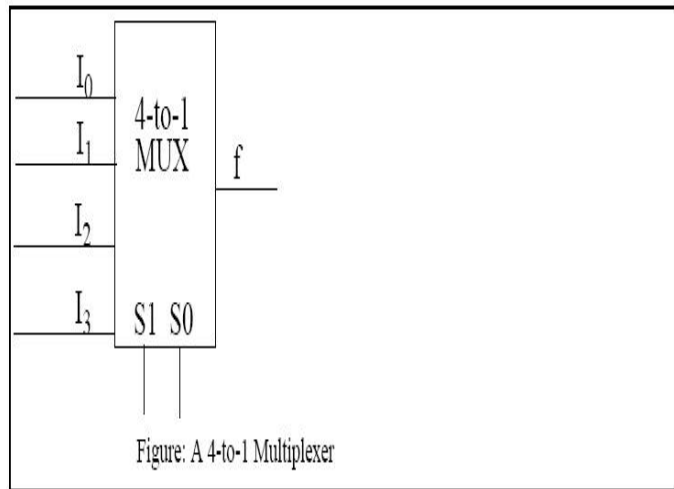
TABLE 2-3 Function Table for 4-to-1-Line Multiplexer

Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Multiplexer Expansion

- Smaller multiplexers can be combined to form a larger one.
- Example: Construct a 4-to-1 multiplexer using 2-to-1 multiplexers.
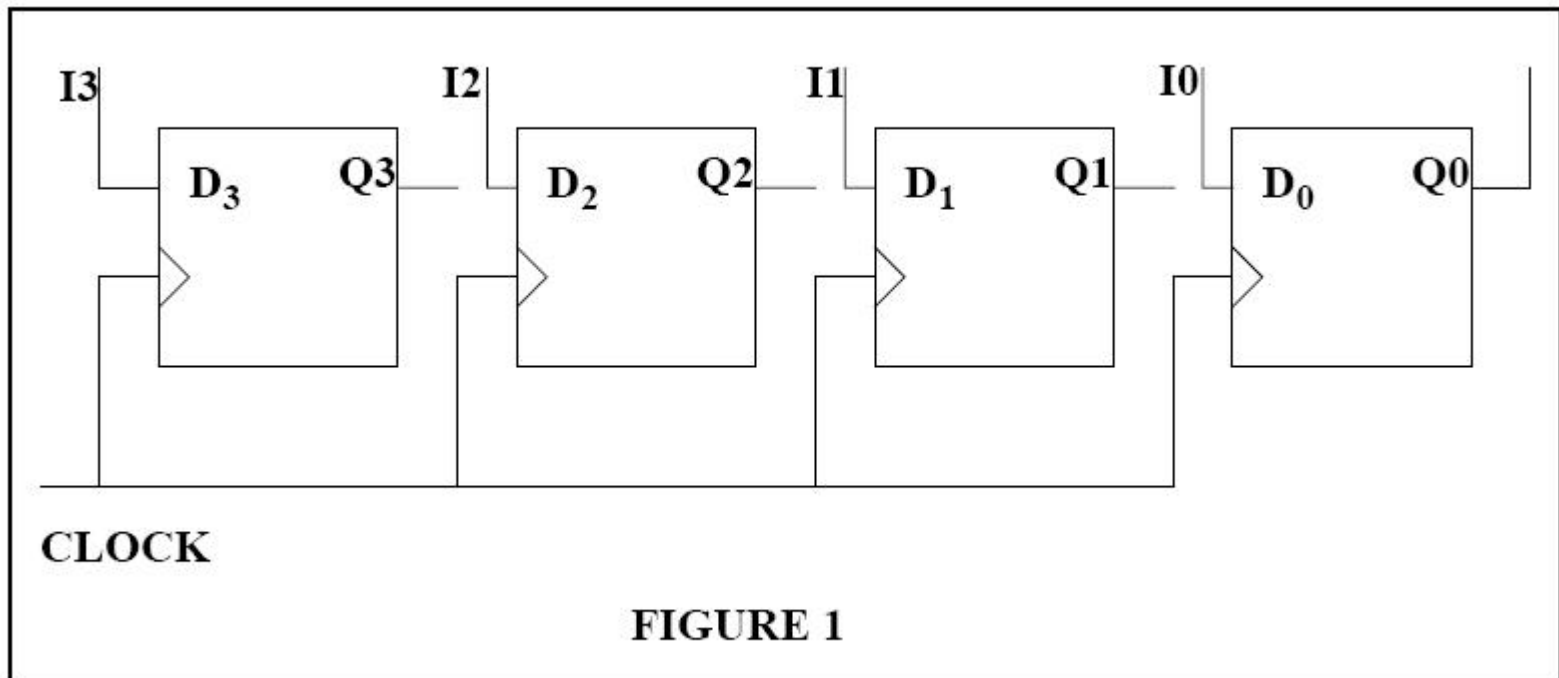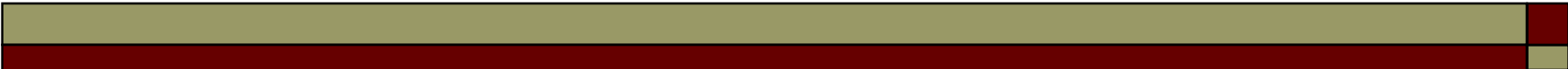
Registers

- A group of flip-flops, where each flip-flop stores 1 bit of information.
- A n-bit register consists of n flip-flops and can store any binary information of n bits.
- May have combinational circuits associated with each flip-flop, for simple data processing operations such as LOAD, INR, INV etc.
- The flip-flops hold binary information and the combinational circuits control how and when new information is transferred to the register.

Registers

□ Figure 1 shows a simple 4-bit register with parallel load. A positive clock transition will load all 4 values I3 - I0 into the register.

FIGURE 1

- In a digital system a clock generator supplies a continuous set of clock pulses.

- A separate control signal (LOAD) is required to determine which clock pulse affects the data in a register.

- When LOAD = 1, a new value is loaded into the register.

- When LOAD = 0, the contents of the register remains unchanged.

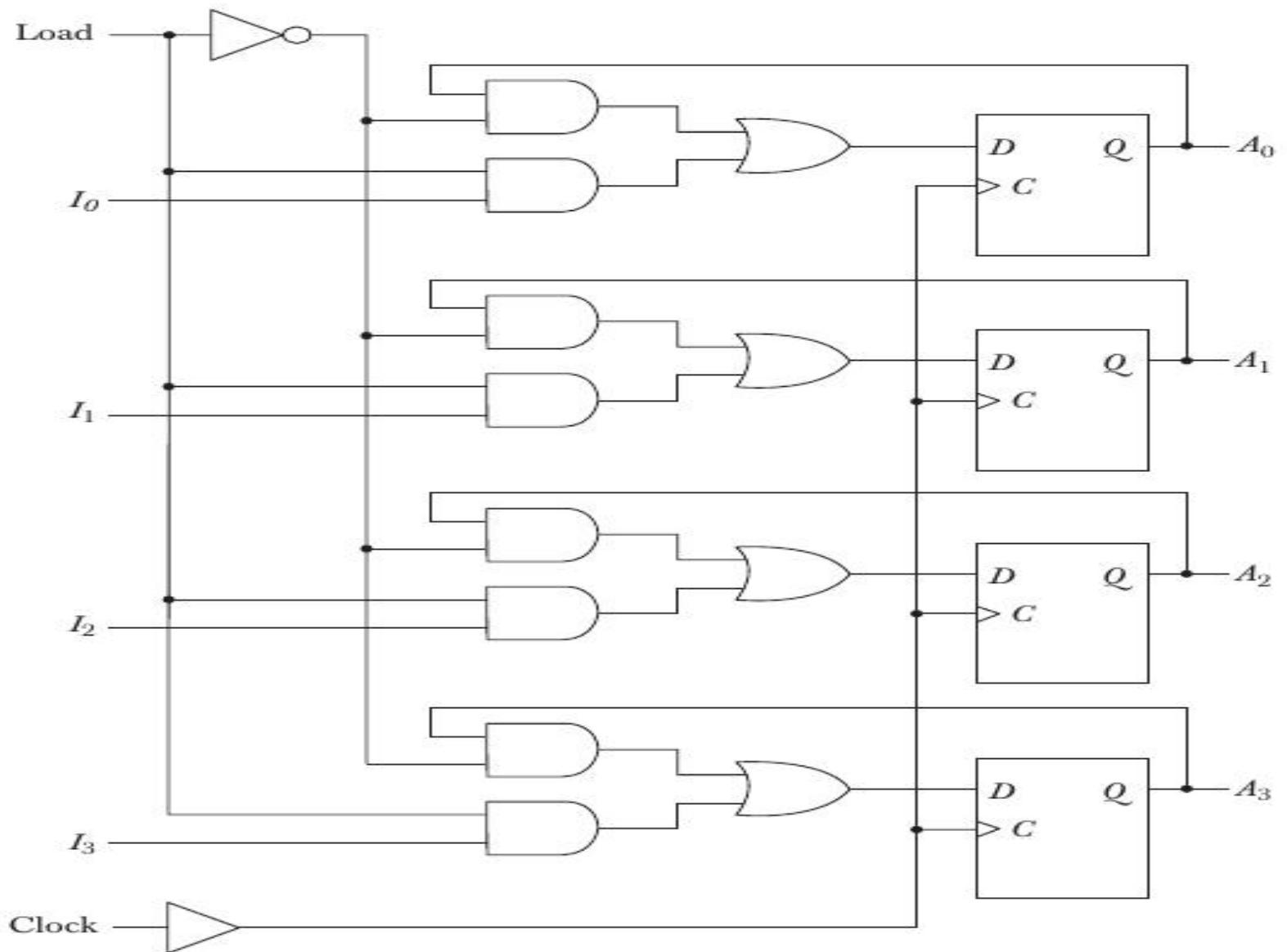**Figure 2-8** 4-bit register with parallel load.

Shift Registers

- A register capable of shifting binary information in one or both directions.

- It consists of a chain of flip-flops in cascade, with the output of one stage connected to the input of the next stage.

- All flip-flops receive a common clock pulse.

- A shift register that can shift in both directions is called a *bidirectional* shift register.

Shift registers

FIGURE 3

SRI = Shift Right Input

SRO = Shift Right Output

CLOCK
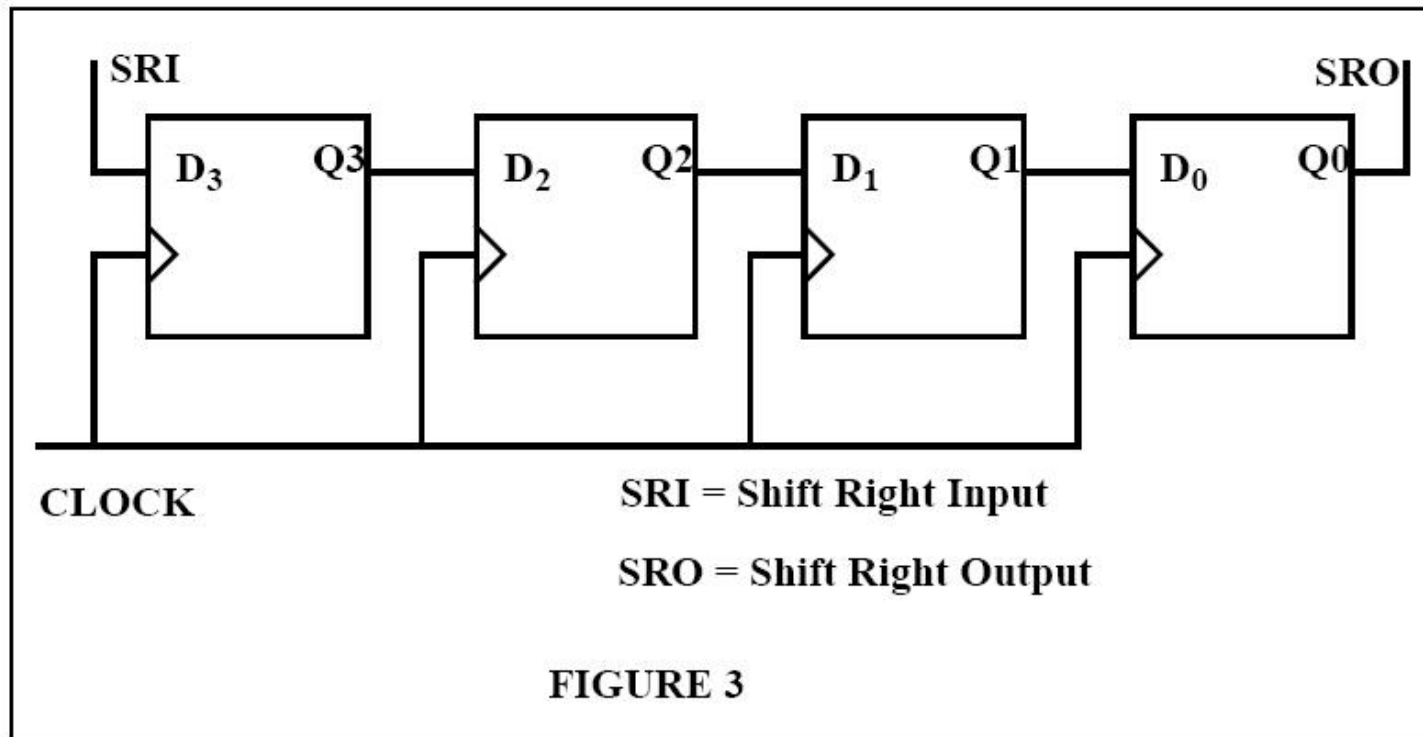
Bidirectional Shift Registers with Parallel Load

- Unidirectional and Bidirectional Shift Registers
- The most general shift register has all the capabilities listed below. Others may have some of these capabilities, with at least one shift operation.
  - An input for clock pulses to synchronize all operations.
  - A shift-right operation and a serial input line associated with the shift-right.
  - A shift-left operation and a serial input line associated with the shift-left.
  - A parallel load operation and n input lines associated with the parallel transfer.
  - n parallel output lines.
  - A control state that leaves the information in the register unchanged even though clock pulses are applied continuously.
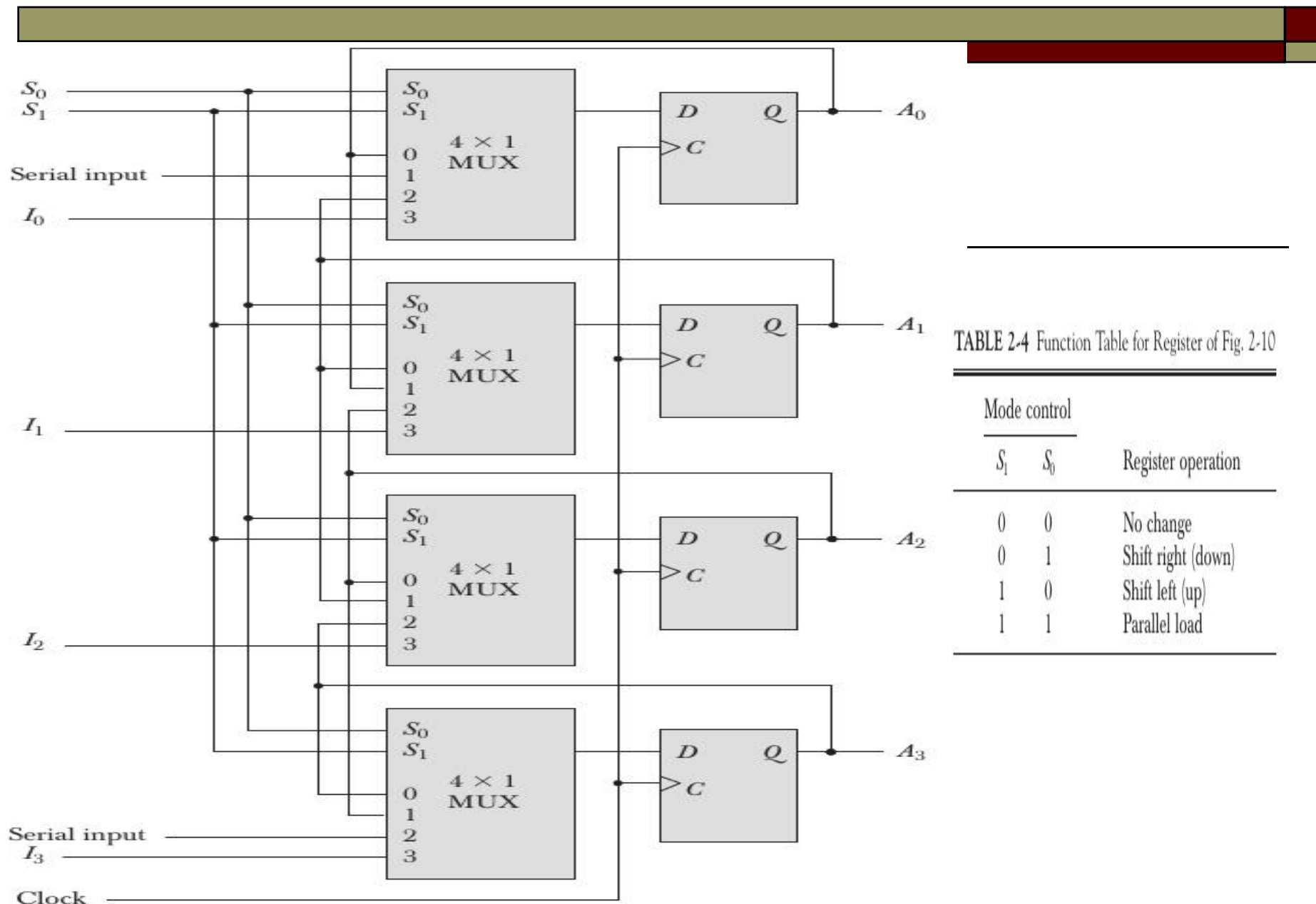
TABLE 2-4 Function Table for Register of Fig. 2-10

Mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift right (down)
1	0	Shift left (up)
1	1	Parallel load

Figure 2-10   Bidirectional shift register with parallel load.

Counters

- A register that can go through a predetermined sequence of states upon the application of input pulses is called a *counter*.

- Counters are widely used in digital systems for counting number of occurrences of events, generating timing and control information etc.

Binary Counters

- A binary counter is one which follows a simple binary sequence.
- A n-bit binary counter can count from 0 to $2^n - 1$.
- A down counter counts in the reverse order.

Capacity of Memory

- ☐ Total number of bytes that can be stored in the memory.

- ☐ Capacity = no. of words X no. of bytes per word.

- ☐ Address lines are used to select one particular word in memory.

- ☐ A memory with 2^k locations requires k address lines.

Examples

(i) How many address and data lines[*] are needed for a 64k X 8 bit memory.

$64k = 2^6 \times 2^{10} = 2^{16}$.

So, 16 address lines are needed.

wordlength = 8 bits. So, 8 data lines are needed.

(ii) How many address and data lines are needed for a 16M X 4 byte memory.

$16M = 2^4 \times 2^{20} = 2^{24}$.

So, 24 address lines are needed.

wordlength = 4 bytes = 32 bits. So, 32 data lines are needed

[*]Assume the entire word is accessed as a unit.

Random Access Memory (RAM)

- Memory cells from any location can be accessed directly.

- Process of locating a word in memory is the same and takes the same amount of time for each location.

- RAM is capable of both READ and WRITE operations.

Steps for accessing a memory location in a RAM

- 1) Apply address to address lines
- 2) Apply data bits to input data lines (for WRITE operation only)
- 3) Activate READ/WRITE control line
- 4) Read data from data output lines (for READ operation only)