Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

- **UNIT-II: Combination circuits, binary adder & subtracter**

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

EMPOWERING THROUGH INNOVATIONS

MLRS

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

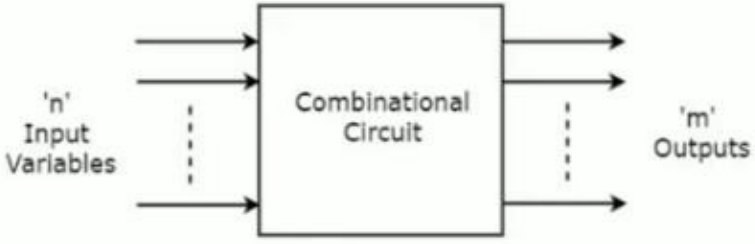
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Digital Circuits

Combinational Circuits

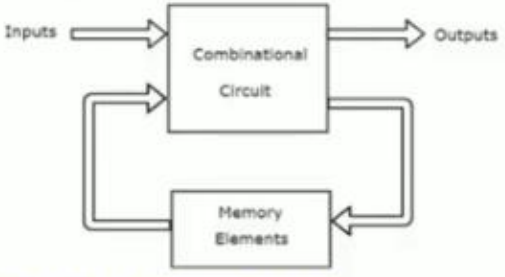
- Outputs of combinational circuit depend on the combination of present **inputs only**



- No Feedback
- No Memory
- Ex. Adder, Subtractor, Code Converter, Encoder, Decoder, Multiplexer and Demultiplexer

Sequential Circuits

- Outputs of Sequential circuit depend on the combination of **present inputs** and **previous (Past) Output**.



- Feedback is present
- Memory is present

Combinational Circuits

- Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer.

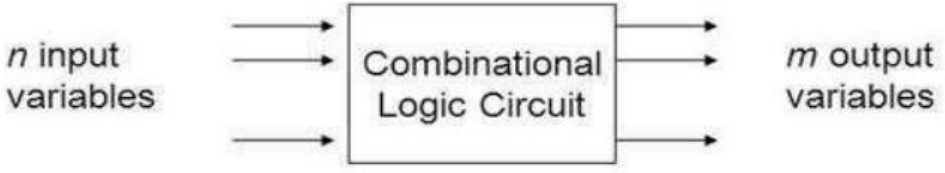
Some of the characteristics of combinational circuits are following:

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

COMBINATIONAL CIRCUITS

- **Block diagram:**

2^n possible combinations of input values.



- Specific functions :of combinational circuits

Adders,subtractors,multiplexers,comprators,encoder,Decoder.

MSI Circuits and standard cells

ANALYSIS PROCEDURE

Analysis procedure

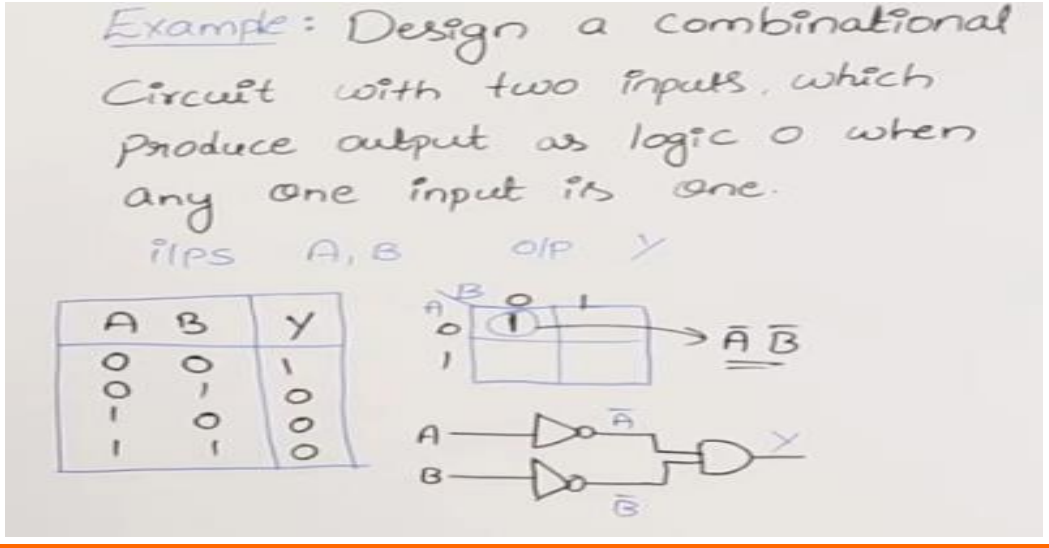
To obtain the output Boolean functions from a logic diagram, proceed as follows:

1. Label all gate outputs that are a function of input variables with arbitrary symbols. Determine the Boolean functions for each gate output.
2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.

DESIGN PROCEDURE

Design Procedure

1. The problem is stated
2. The number of available input variables and required output variables is determined.
3. The input and output variables are assigned letter symbols.
4. The truth table that defines the required relationship between inputs and outputs is derived.
5. The simplified Boolean function for each output is obtained.
6. The logic diagram is drawn.



Combinational circuits

Example:

Design a combinational logic circuit with three input variables that will produce a logic 1 output when more than one input variable are logic 1.

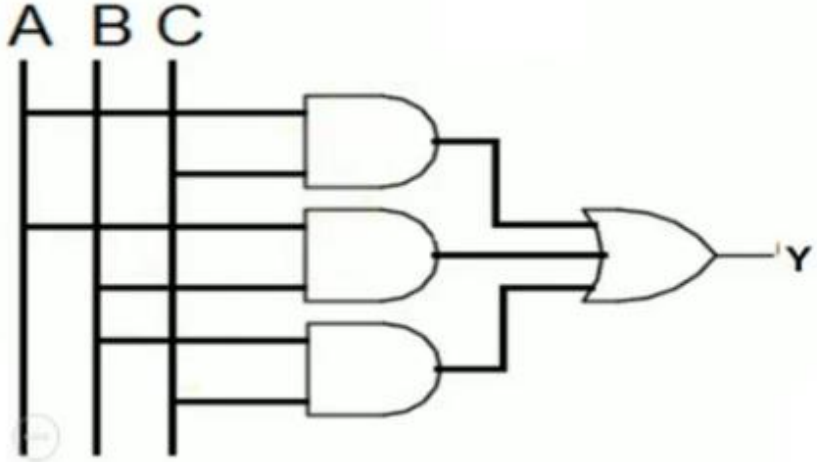
Truth Table for Circuit

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Combinational circuits

Y	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	0	0	1	0
A	0	1	1	1

$$Y = AC + AB + BC$$



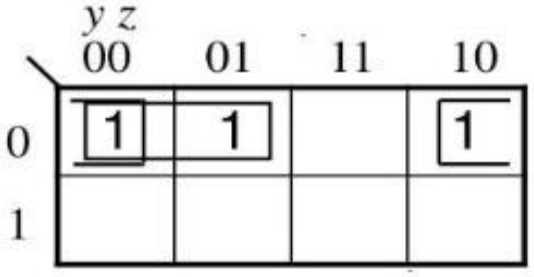
Truth Table for Circuit

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

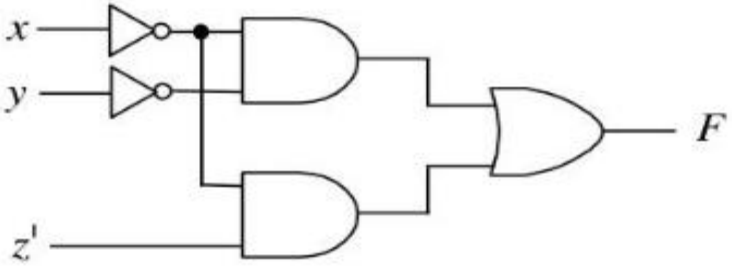
DESIGN OF COMBINATIONAL LOGIC

Example: *Design a combinational circuit with three inputs and one output. The output is a 1 when the binary value is less than three. The output is 0 otherwise.*

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



$$F = x'y' + x'z'$$

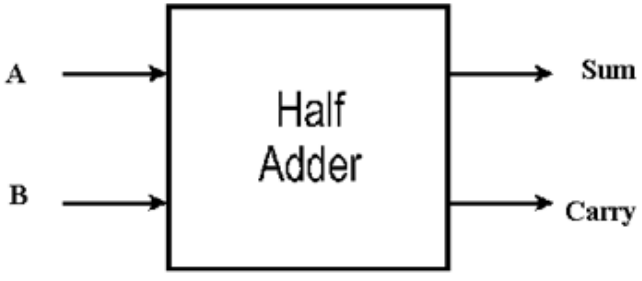


BINARY ADDERS

ADDERS

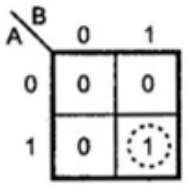
Half Adder

A Half Adder is a combinational circuit with two binary inputs (augends and addend bits and two binary outputs (sum and carry bits.) It adds the two inputs (A and B) and produces the sum (S) and the carry (C) bits.



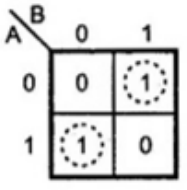
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

For Carry

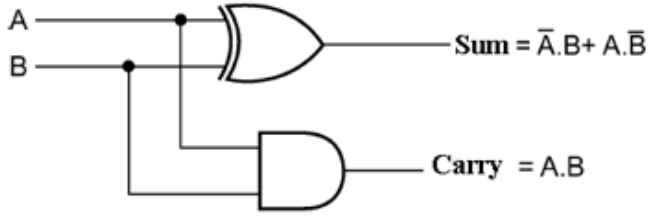


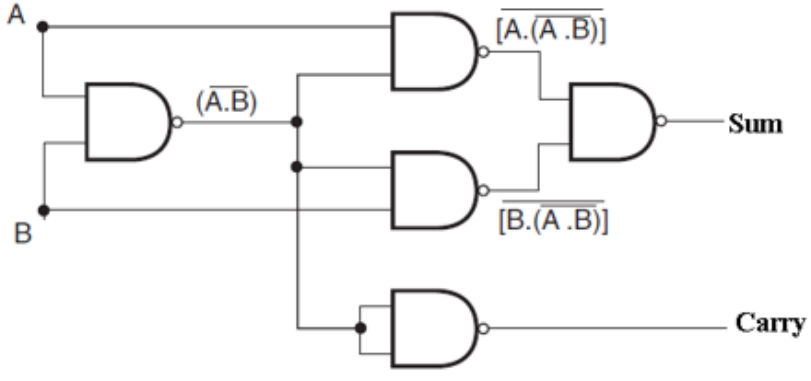
Carry = AB

For Sum



Sum = $A\bar{B} + \bar{A}B$
= $A \oplus B$





BINARY ADDERS

Full Adder

The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs the sum bit S and the carry bit called the carry-out C_{out} .

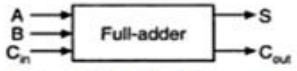


Fig : block diagram

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig :Truth table

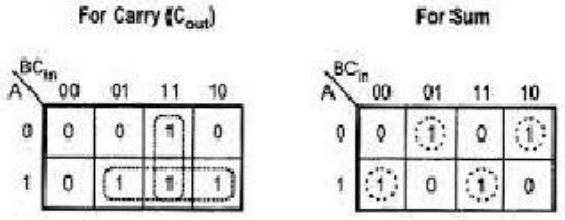


Fig. Maps for full-adder

$S=\bar{A}B\bar{C}+A\bar{B}\bar{C}+\bar{A}\bar{B}C+ABC$

$=A\,(\bar{B}\bar{C}+BC)+\bar{A}\,(B\bar{C}+\bar{B}C)$

$=A\,(B \text{ XNOR } C)+\bar{A}\,(B \text{ XOR } C)$

$=A \text{ XOR } B \text{ XOR } C$

$Cout= AB\bar{C}+ABC+\bar{A}BC+ABC+A\bar{B}C+ABC$

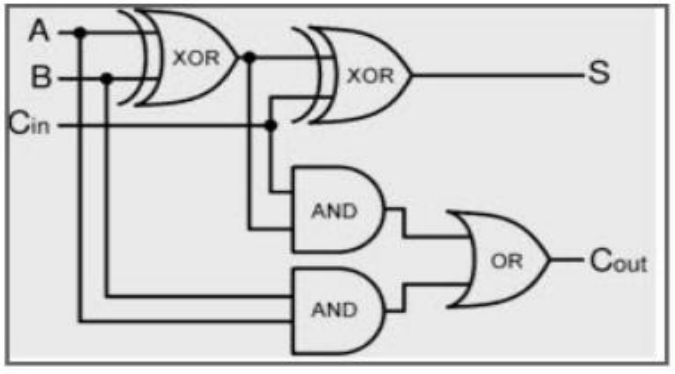
$=AB\bar{C}+ABC+\bar{A}BC+A\bar{B}C+(ABC+ABC)$

$=AB\,(\bar{C}+C)+C\,(\bar{A}B+A\bar{B})+ABC$

$=AB+C\,(A \text{ XOR } B)+ABC$

$=AB\,(1+C)+C\,(A \text{ XOR } B)$

$=AB+C\,(A \text{ XOR } B)$



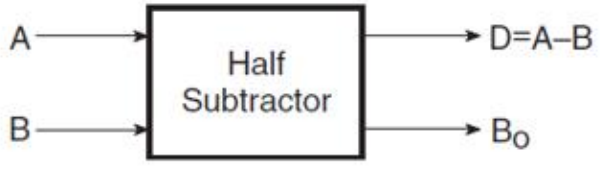
Full Adder Logical Diagram

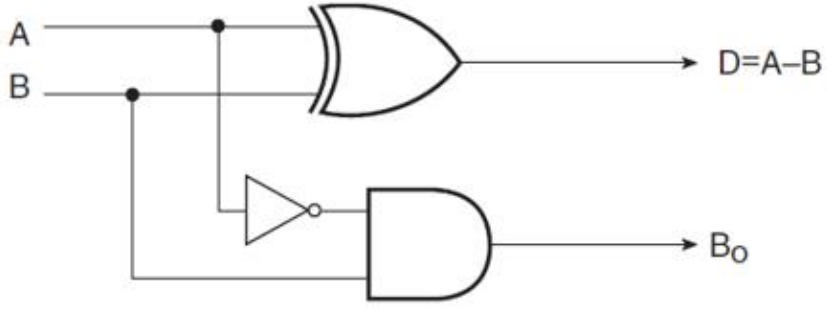
BINARY SUBTRACTORS

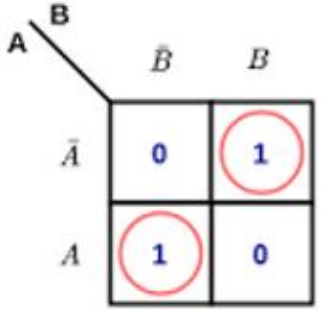
Half Subtractor

A Half-subtractor is a combinational circuit with two inputs A and B and two outputs difference and barrow.

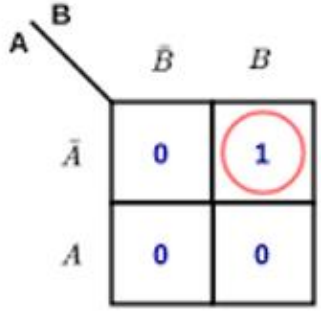
A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0







$D = A'B + AB' = A \oplus B$



$B_0 = \bar{A}.B$

BINARY SUBTRACTORS

Full subtractor

The full subtractor perform subtraction of three input bits: the minuend , subtrahend , and borrow in and generates two output bits difference and borrow out .

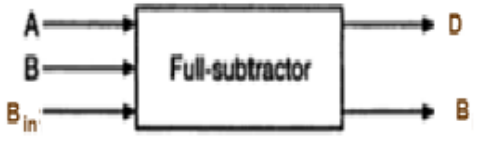


Fig :Block diagram

Inputs			Difference	Borrow
A	B	B_{in}	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

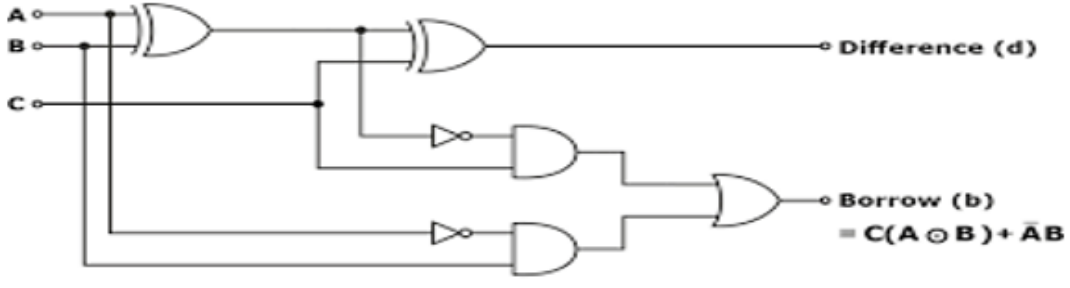
Fig : Truth table

For D:

$D = A \oplus B \oplus B_{in}$

For B_{in}:

$B_{out} = \bar{A}B + (\bar{A} + B)B_{in}$



Logic diagram of full subtractor in SOP form

Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

UNIT-II: Decimal adder, binary multiplier, Magnitude compotator

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

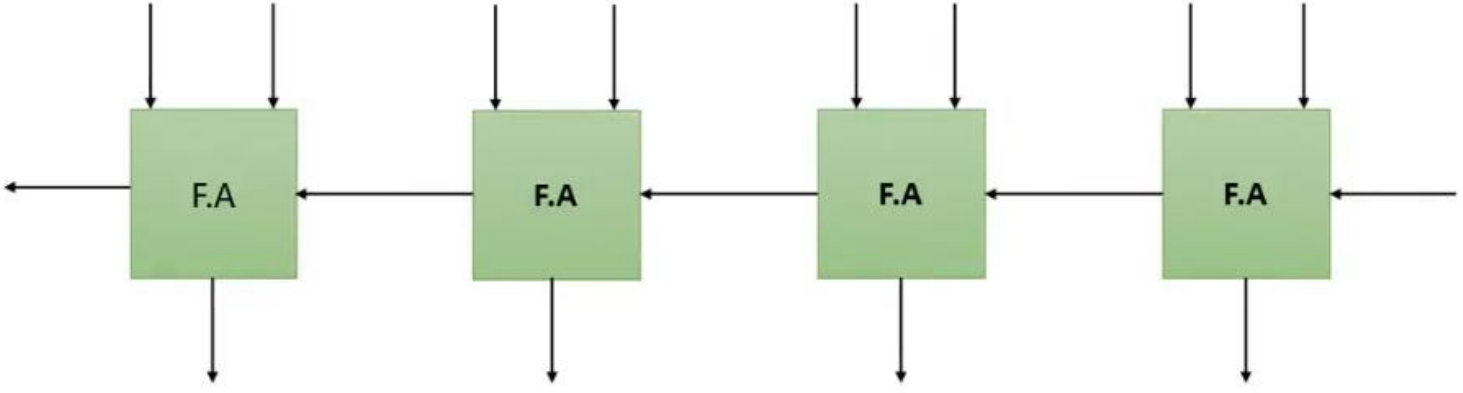
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Binary Parallel Adder

- By the use of single "Full-Adder", we are only capable of adding " two one-bit numbers" and an "input carry". So to add binary numbers that consist more than one bit , we have to use Parallel-Adder.

- The Binary Parallel adder consists of "Full-adders" connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder.

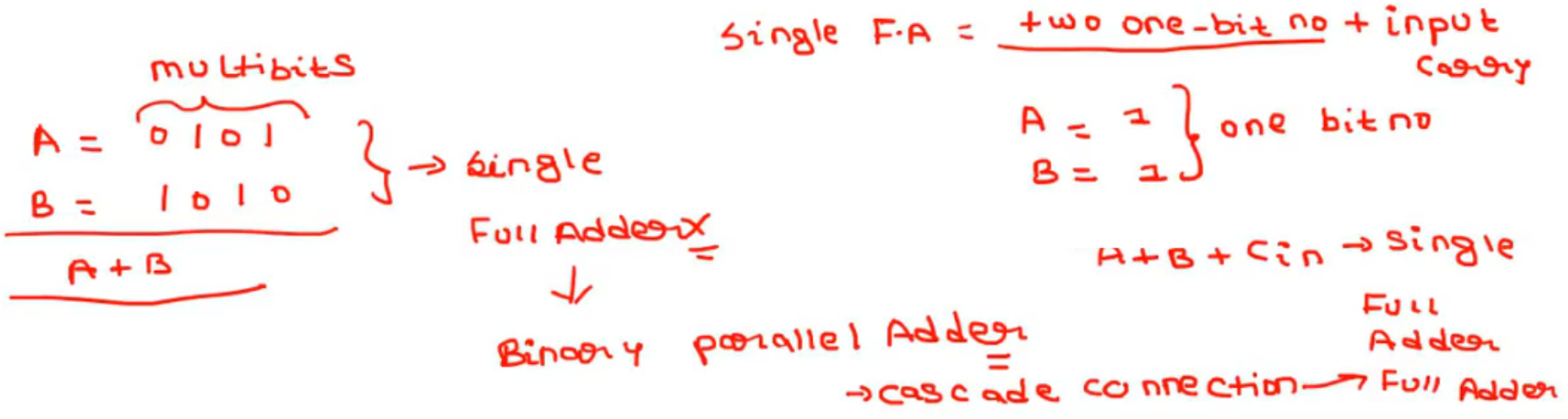
Block Diagram of Four-Bit Parallel Adder



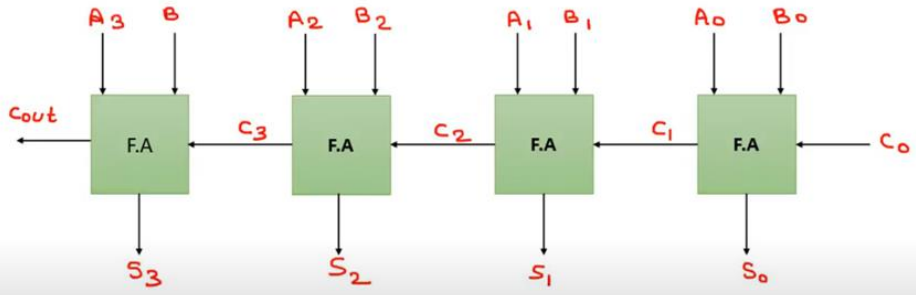
Binary Parallel Adder

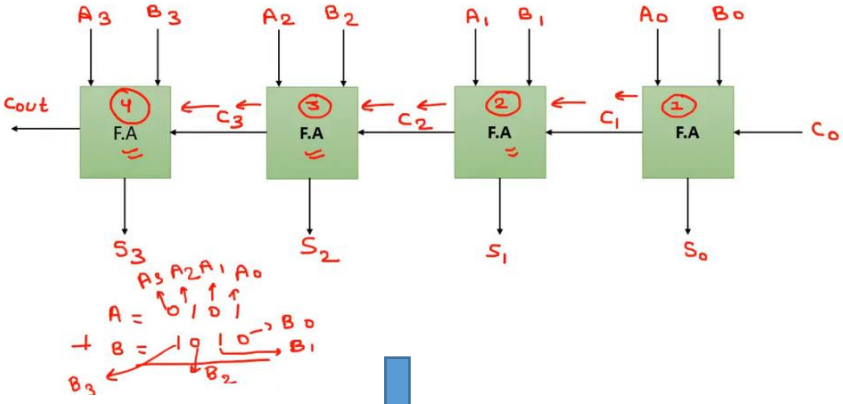
- By the use of single "Full-Adder", we are only capable of adding " two one-bit numbers" and an "input carry". So to add binary numbers that consist more than one bit , we have to use Parallel-Adder.

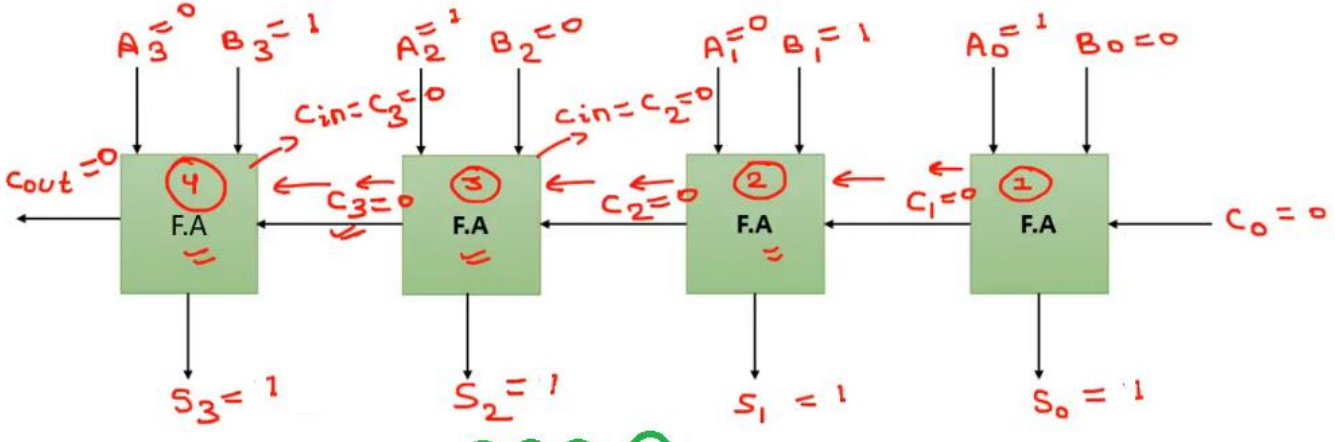
- The Binary Parallel adder consists of "Full-adders" connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder.



Block Diagram of Four-Bit Parallel Adder







A = 0 1 0 1
+ B = 1 0 1 0

C A B S

0 + 1 + 0 = 1

0 + 1 + 0 = 1

1 + 0 + 0 = 1

0 + 1 + 0 = 1

A + B = 1 1 1 1

Designing of BCD Adder

- A BCD adder adds two BCD digits and produces BCD digit.
- BCD cannot be greater than 9.

```
   7        0111
+  1      + 0001
------    ------
   8        1000

   7        0111
+  4      + 0100
------    ------
  11        1011
          + 0110
          ------
           10001

   9        1001
+  8      + 1000
------    ------
  17       10001
        +   0110
          ------
           10111
```

BCD ADDER:

A BCD adder should perform the following

- Add two 4-bit BCD number using straight Binary Addition.

- If the sum of two numbers is equal to or less than 9, then the sum is valid BCD number and no correction is required.

- If the sum of two numbers is greater than 9 or carry is generated from the sum, then the sum is not valid BCD number. Then add 0110 (6) to the sum, the result will be valid BCD number. If further a carry is generated then add 0110 to the result.

DESIGN:

- In fig. 1 $B_3B_2B_1B_0$ and $A_3A_2A_1A_0$ are the BCD inputs. $S_3S_2S_1S_0$ and C_{out} is the output of Adder 1.
- Checked the output of Adder 1, whether it is greater than or less than 9.
- If the sum of Adder 1 is greater than 9 then the output of combinational circuit should be 1 (i.e C_{out} should be high) and correction is required.
- Write the truth table and K-Map for combinational circuit.
- The Boolean Expression from K-Map $Y= S_3S_2+S_3S_1$
- The output of combinational circuit Y_2 is connected to B_2B_1 of Adder 2 and B_3B_0 are connected to the ground. Therefore $B_3=B_0=0$
- The output sum of Adder 1 is connected to Adder 2. the output of Adder 2 is the final result of BCD addition with Carry which can be ignored if any.

Correction:

BCD:

Decimal	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

$BCD_1 + BCD_2$

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \end{array}$$

1 0

TRUTH TABLE

INPUTS				OUTPUTS
S3	S2	S1	S0	Y_1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Sum is valid BCD number. Therefore Y=0

Sum is valid BCD number. Therefore Y=1

K-Map

S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$Y_1=S_3S_2+S_3S_1$

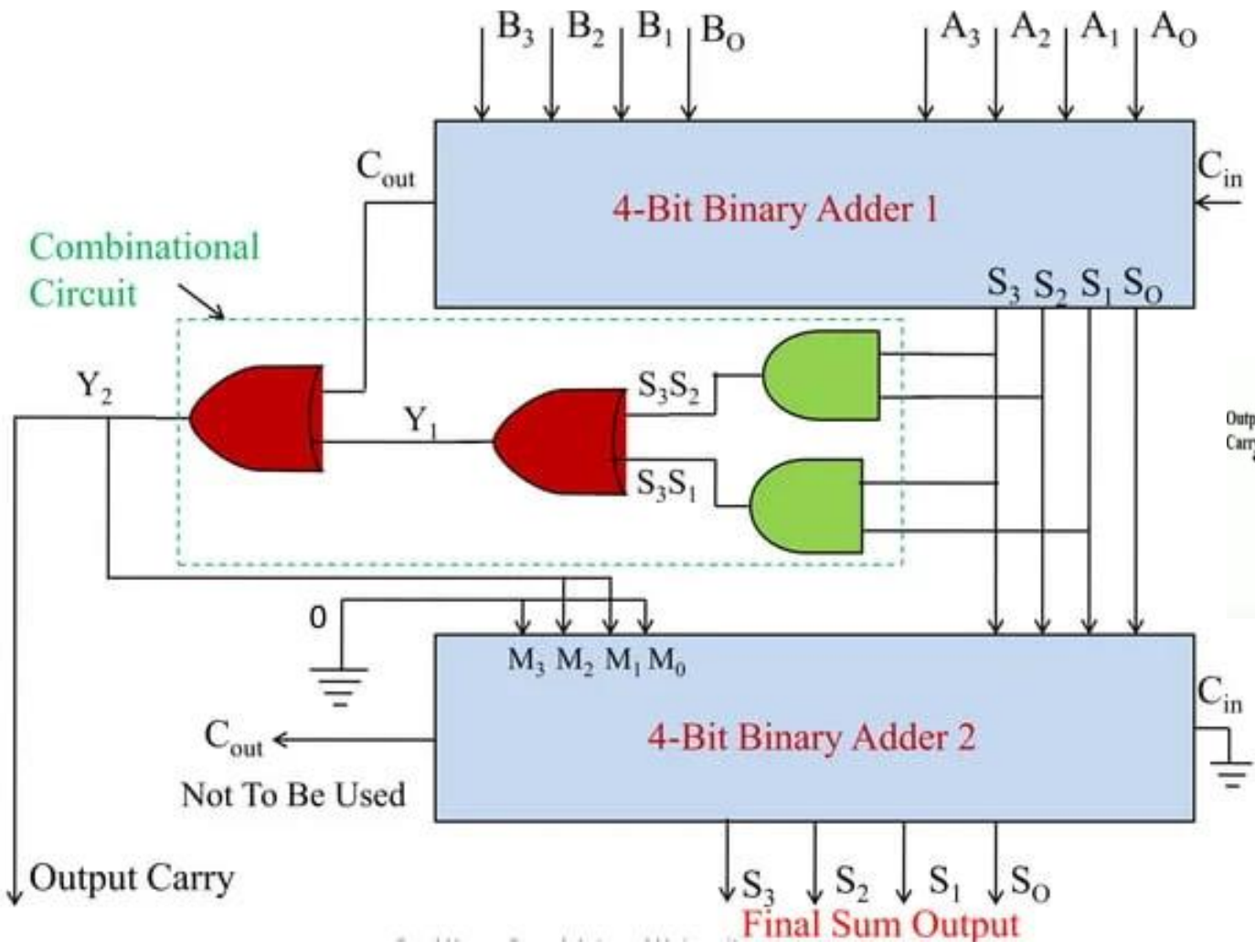
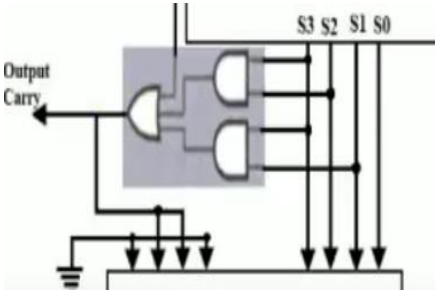


Fig. 1: BLOCK DIAGRAM OF BCD ADDER

or-else we can use a single OR gate



CASE 1: Sum equal to or less than 9 with carry 0

	DECIMAL	BCD
	2	0 0 1 0
	+ 1	0 0 0 1
Carry		0
SUM	3	0 0 0 1 1

Final Carry

Valid BCD Number

Output of combinational circuit $Y_2 = 0$, $M_3\ M_2\ M_1\ M_0 = 0$

Output of Adder 2 is same as the output of Adder 1

CASE 1:

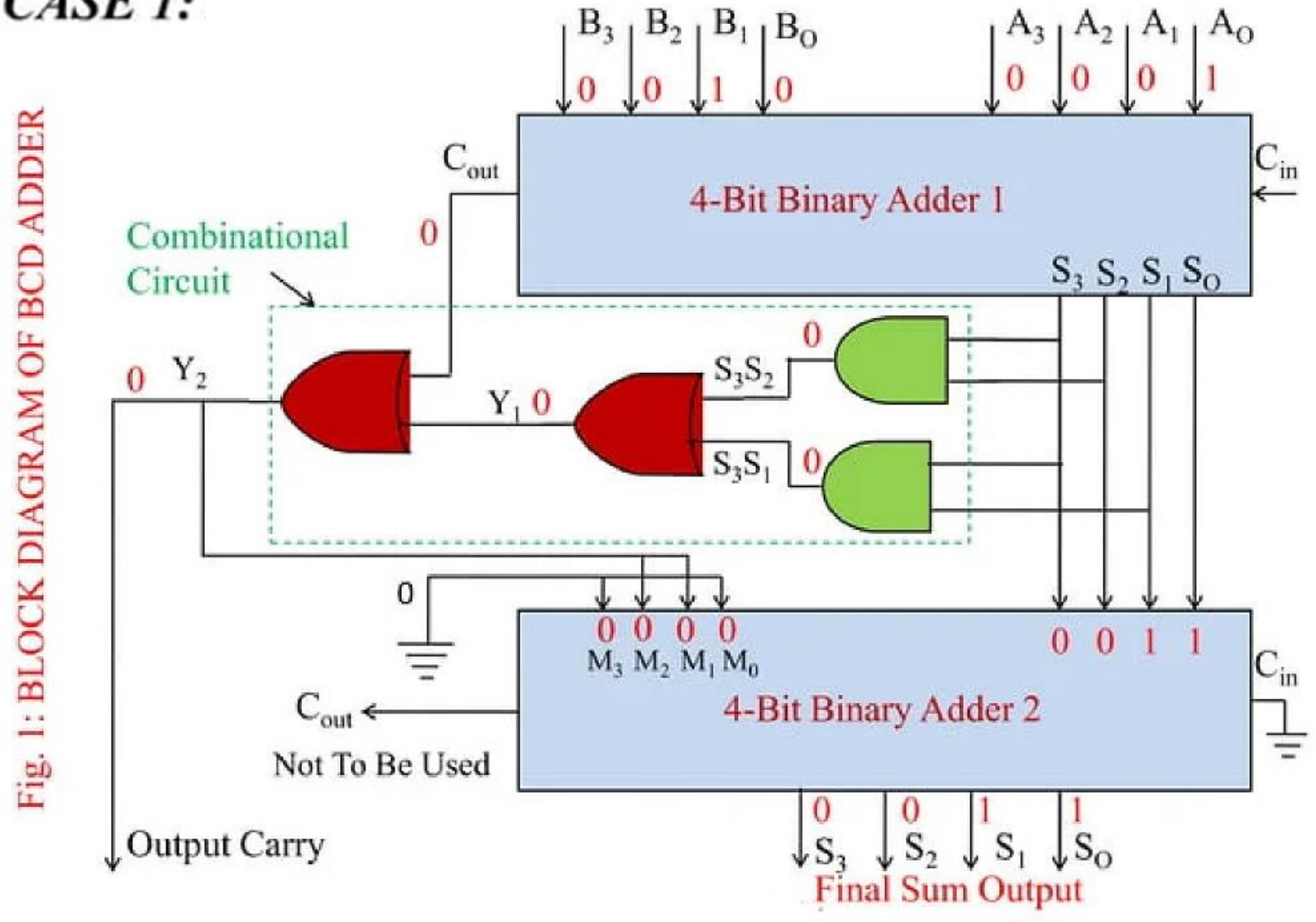


Fig. 1: BLOCK DIAGRAM OF BCD ADDER

CASE 2: Sum is equal to or less than 9 with carry 1

	DECIMAL	BCD	
	8	1 0 0 0	
	+ 8	1 0 0 0	
Carry		1	
SUM	16	1 0 0 0 0	*Invalid BCD Number*
		0 1 1 0	*Add 6 for correction*
Final result		1 0 1 1 0	*Valid BCD number with carry 1*

Final Carry

0 0 0 1 — 1

0 1 1 0 — 6

Output of combinational circuit $Y_2 = 1$, $M_3\ M_2\ M_1\ M_0 = 0\ 1\ 1\ 0$

CASE 2:

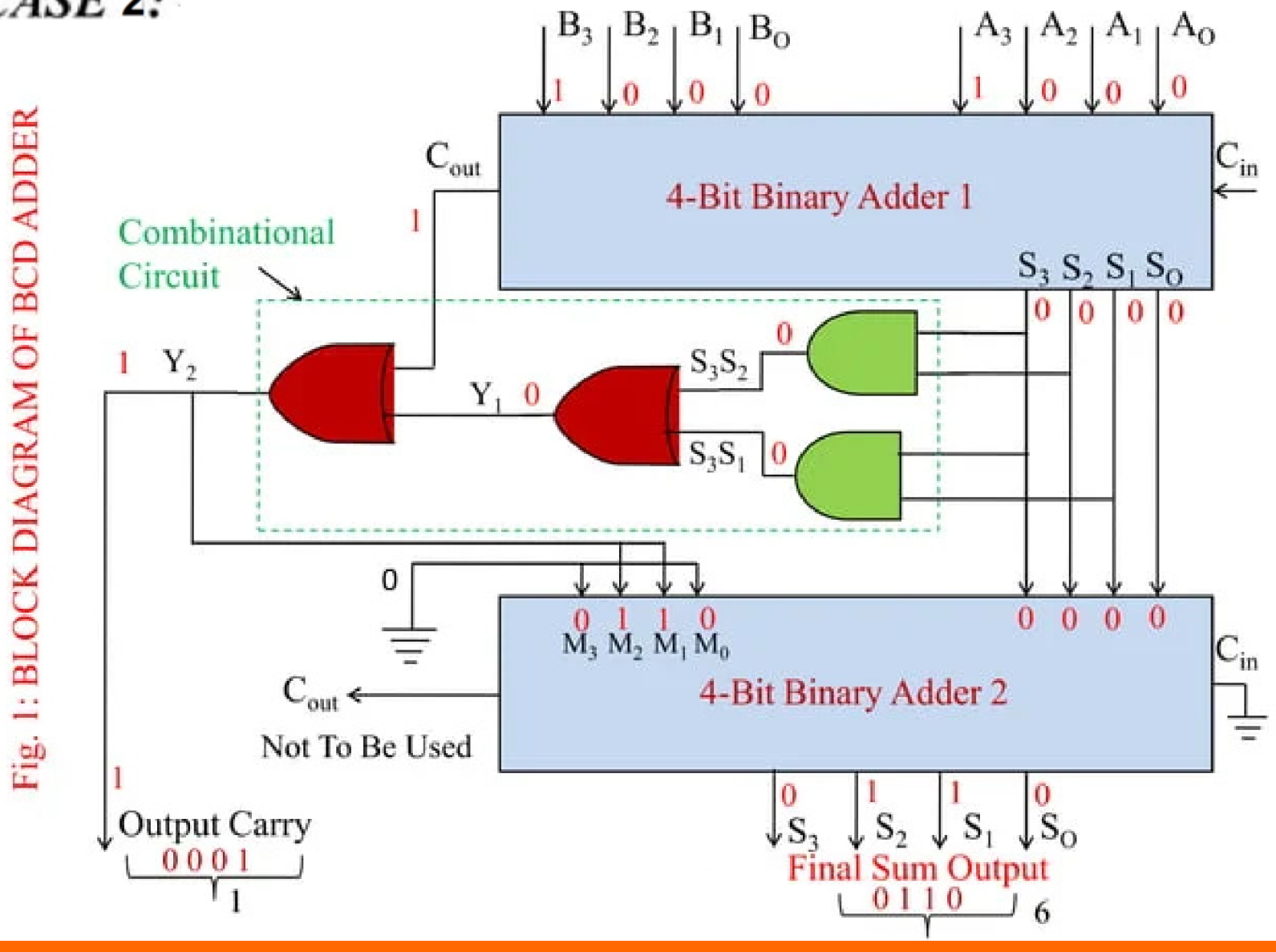


Fig. 1: BLOCK DIAGRAM OF BCD ADDER

Binary Multiplier

- A binary-multiplier is used multiply two binary numbers.
- It is built using "binary adders". The two numbers are known as "Multiplicand" and "Multiplier" and the result is known as "Product".

A → Multiplicand
× B → multiplier
A×B → product

Binary multiplication:

0 × 0 = 0 1 × 0 = 0
0 × 1 = 0 1 × 1 = 1

Shift and Add principle

```
   0 1 1
 × 1 1 0
 -------
   0 0 0
 1 1 0 x   ← left Shift
 1 1 0 x x → left Shift
 -------
 1 0 1 0   → product
 -------
```

2*2 Multiplier

Multiplicand $A = A_1 A_0$

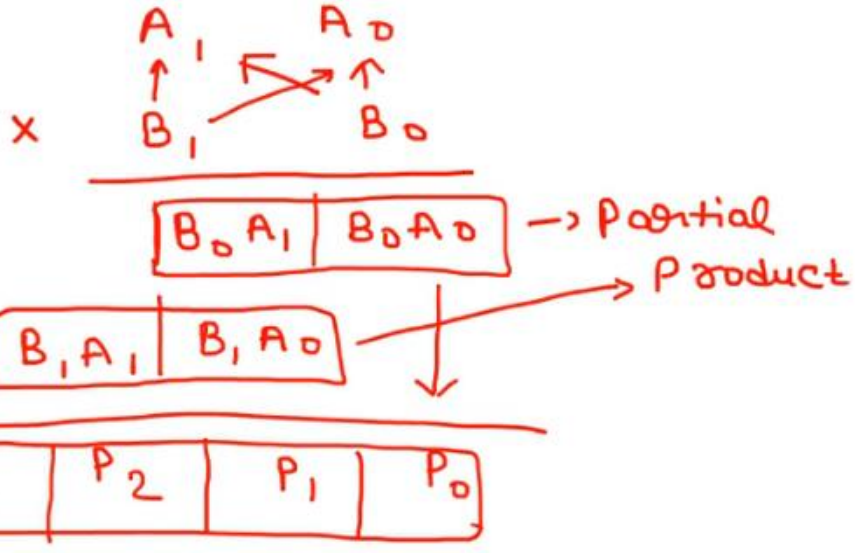
Multiplier $B = B_1 B_0$

$P_0 = B_0 A_0$

$P_1 = B_0 A_1 + B_1 A_0$

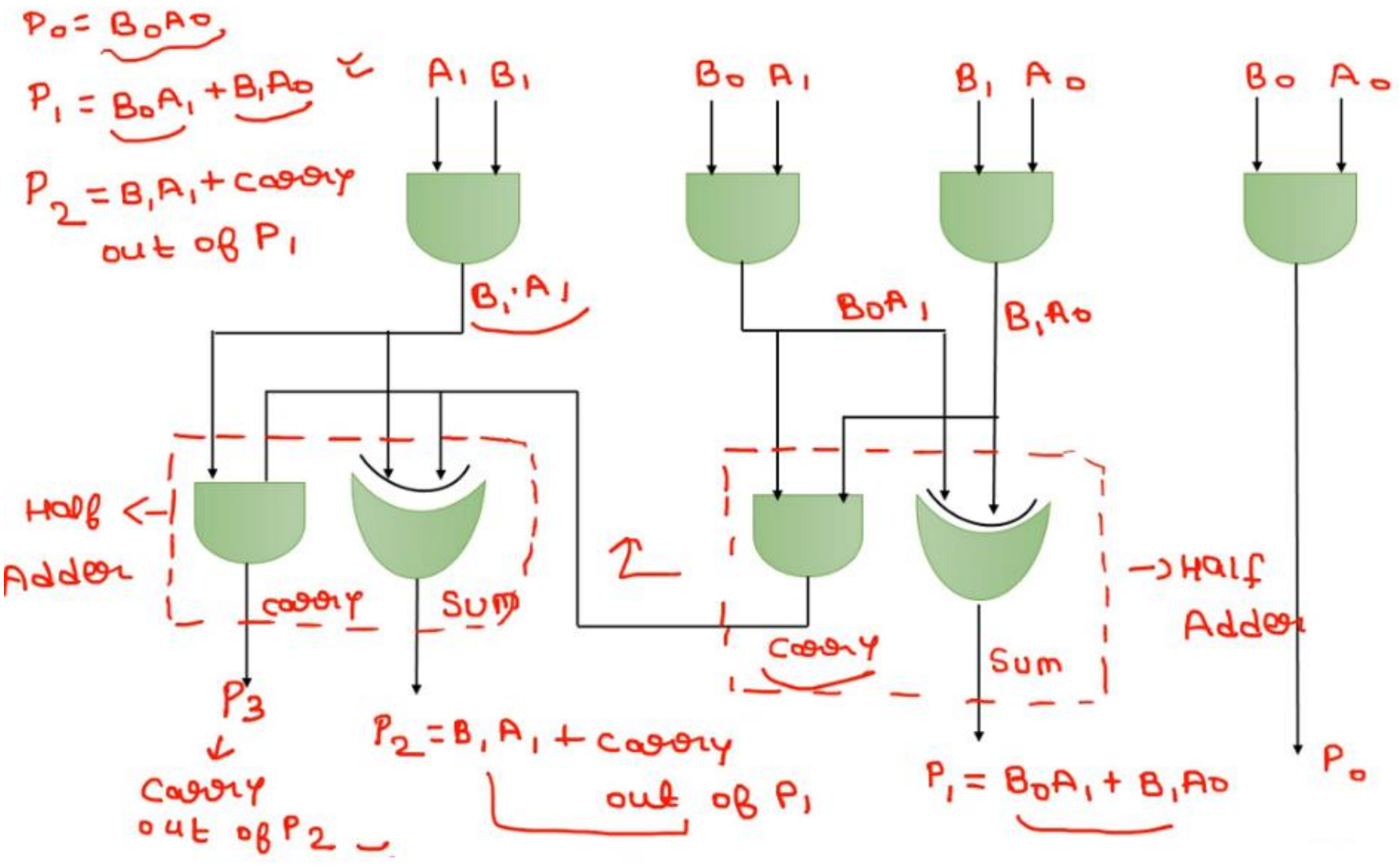
$P_2 = B_1 A_1 +$ carry out of P_1

$P_3 =$ carry out of P_2



Partial product -> using AND gate

Partial product Add -> Full Adder/ Half Adder


$P_0 = B_0A_0$
$P_1 = B_0A_1 + B_1A_0$
$P_2 = B_1A_1 + carry$ out of P_1
A_1 B_1
B_0 A_1
B_1 A_0
B_0 A_0
$B_1 \cdot A_1$
B_0A_1
B_1A_0
Half Adder
carry
Sum
Half Adder
carry
Sum
P_3
carry out of P_2
$P_2 = B_1A_1 + carry$ out of P_1
$P_1 = B_0A_1 + B_1A_0$
P_0

MAGNITUDE COMPARATOR

Magnitude comparator takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number.

1-Bit Magnitude Comparator

A comparator used to compare two bits is called a single bit comparator.

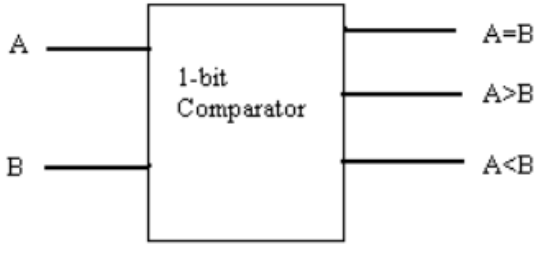


Fig :1 Block diagram

1-bit Comparator

Truth Table

A	B	A = B	A > B	A < B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

Output = $\overline{A}\,\overline{B} + A\,B$

Truth Table

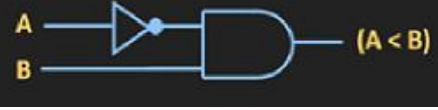
A	B	A = B	A > B	A < B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

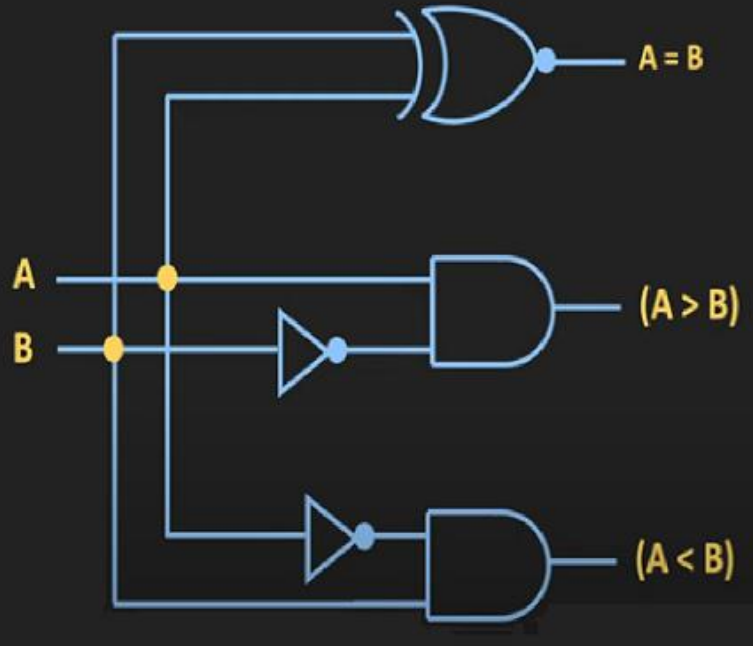
$(A > B) = A\,\overline{B}$

Truth Table

A	B	A = B	A > B	A < B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

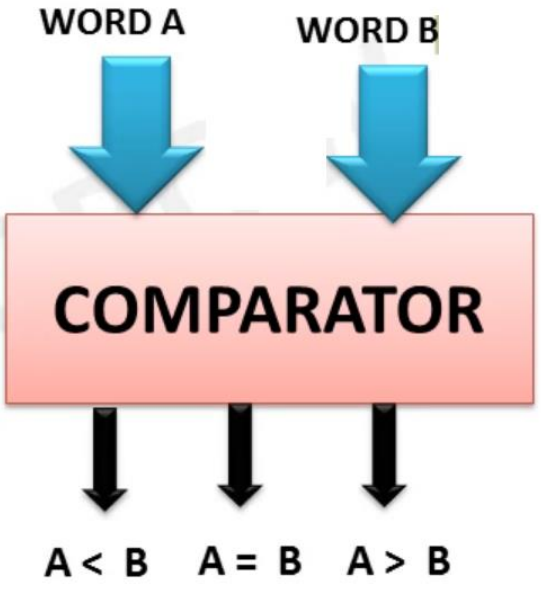
$(A < B) = \overline{A}\,B$





2 BIT COMPARATOR

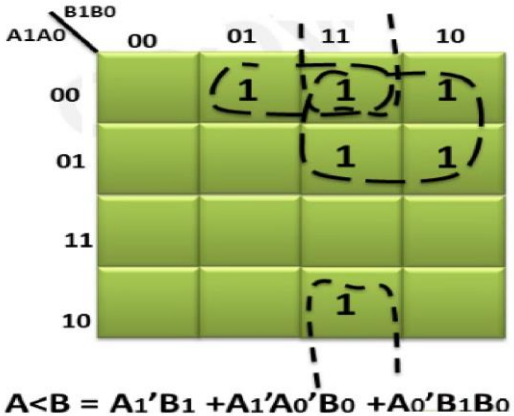
A1	A0	B1	B0	A < B	A = B	A > B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0



2 BIT COMPARATOR

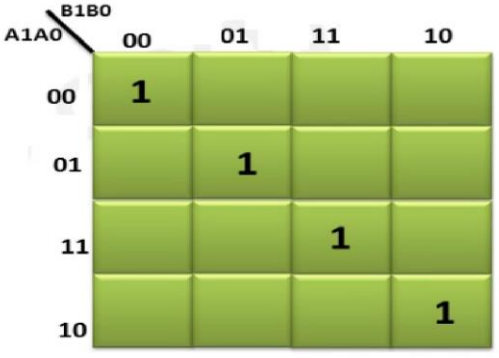
A1	A0	B1	B0	A < B	A = B	A > B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

A < B



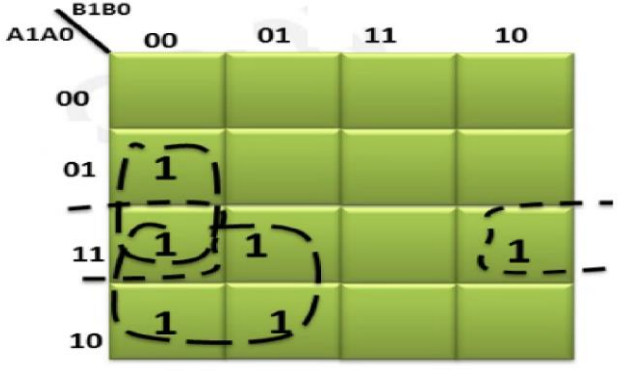
$A<B = A_1'B_1 + A_1'A_0'B_0 + A_0'B_1B_0$

A = B



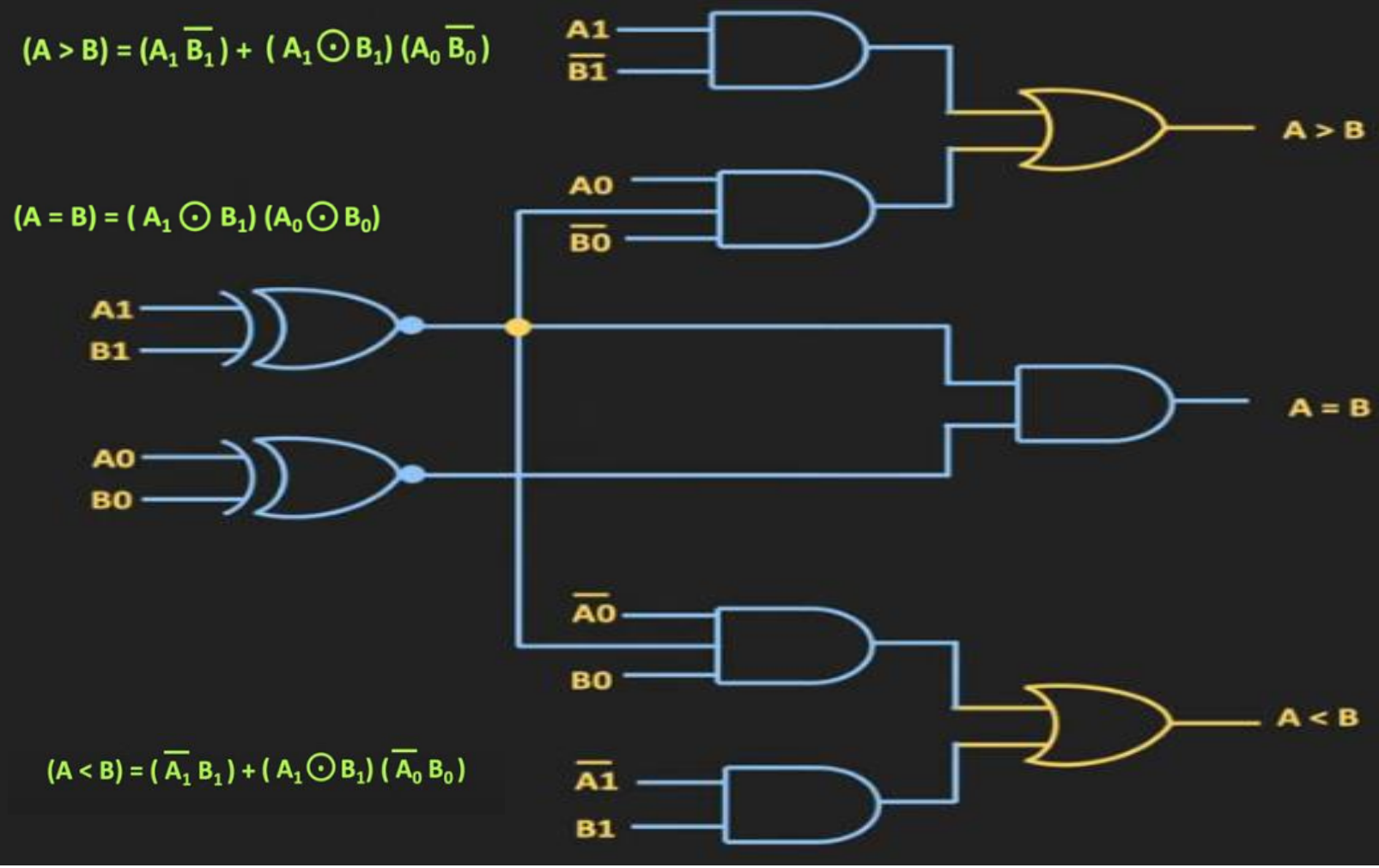
$A=B = (A_1 \odot B_1)\ (A_0 \odot B_0)$

A > B



$A>B = A_1B_1'B_0' + A_1A_0B_0' + A_1B_1'$

2-bit Comparator



Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

- **UNIT-II: Decoders, Encoders, Multiplexers**

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

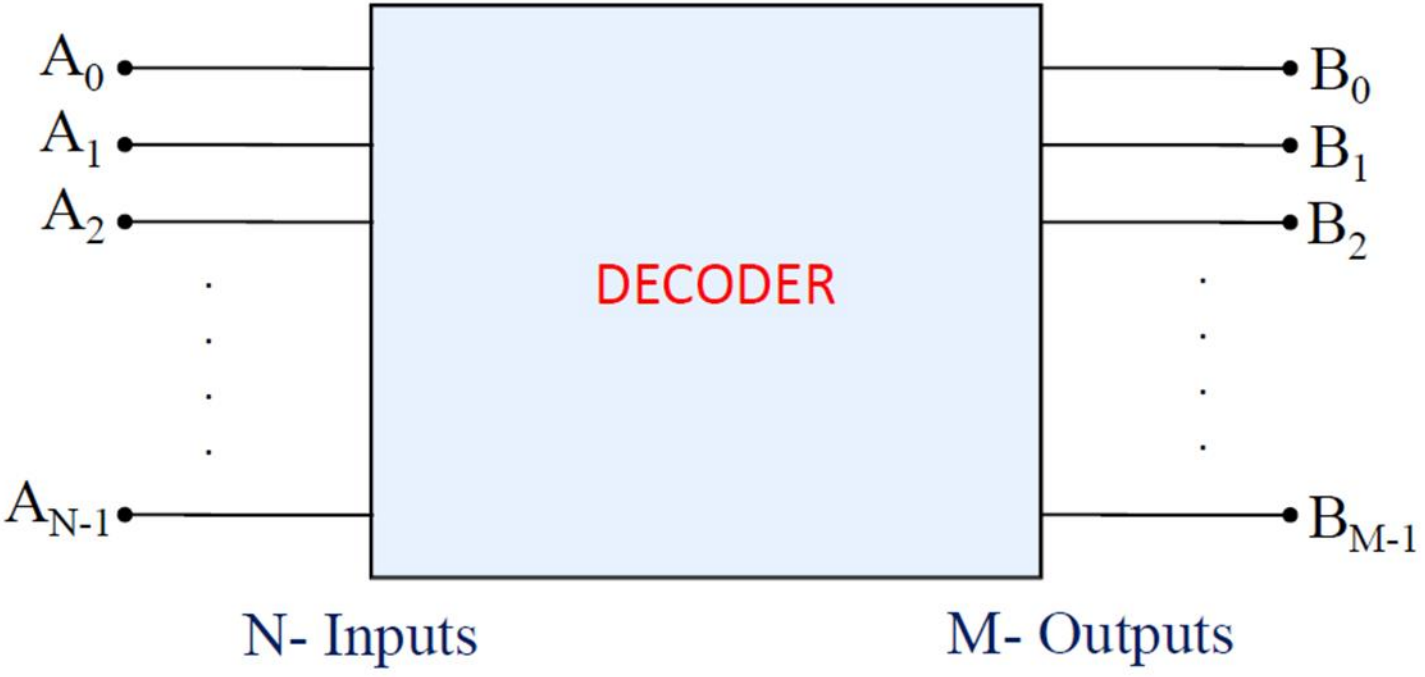
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DECODER

- A decoder is a combinational circuit.
- A decoder accepts a set of inputs that represents a binary number and activates only that output corresponding to the input number. All other outputs remain inactive.
- Fig. 1 shows the block diagram of decoder with 'N' inputs and 'M' outputs.
- There are 2^N possible input combinations, for each of these input combination only one output will be HIGH (active) all other outputs are LOW
- Some decoder have one or more ENABLE (E) inputs that are used to control the operation of decoder.

BLOCK DIAGRAM OF DECODER



Only one output is High for each input

Fig. 1

2 to 4 Line Decoder:

- Block diagram of 2 to 4 decoder is shown in fig. 2
- A and B are the inputs. (No. of inputs =2)
- No. of possible input combinations: $2^2=4$
- No. of Outputs : $2^2=4$, they are indicated by D_0, D_1, D_2 and D_3
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.

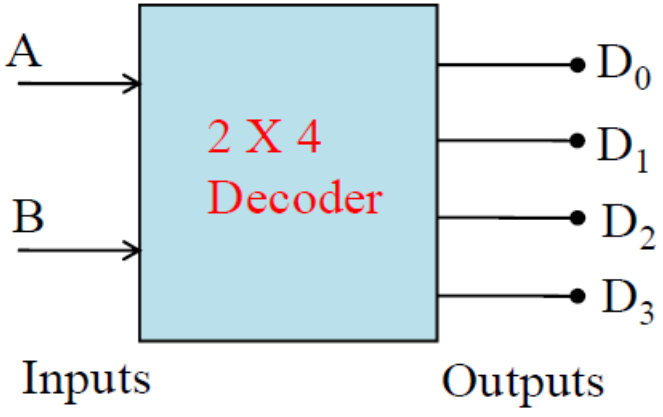


Fig. 2

TRUTH TABLE

INPUTS		OUTPUTS			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

BOOLEAN EXPRESSION:

From Truth Table

$$D_0 = \overline{A}\,\overline{B} \qquad D_1 = \overline{A}\,B$$

$$D_2 = A\,\overline{B} \qquad D_3 = AB$$

LOGIC DIAGRAM:

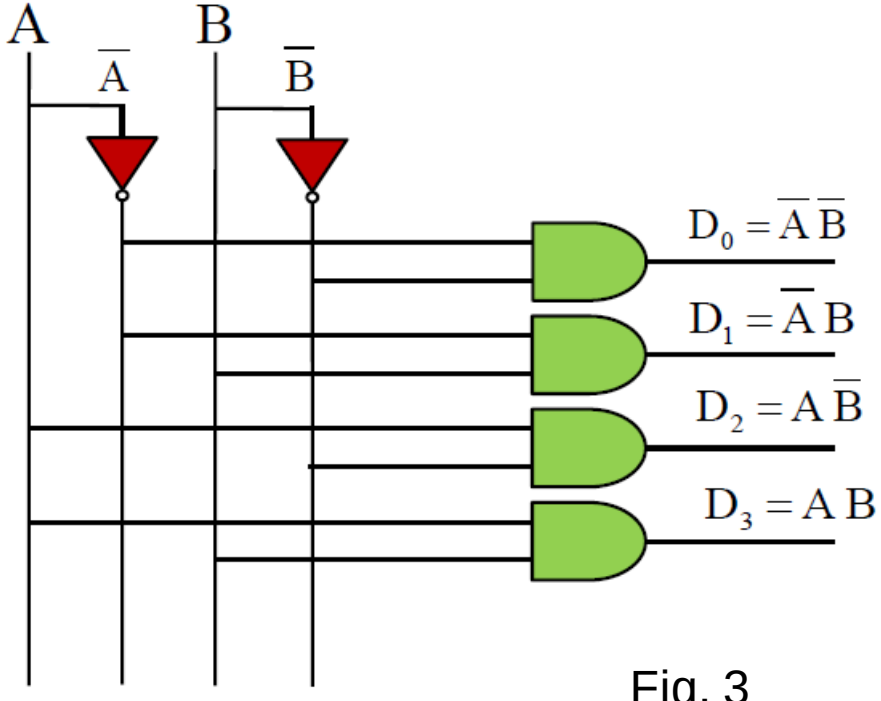


Fig. 3

TRUTH TABLE

INPUTS		OUTPUTS			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

3 to 8 Line Decoder:

- Block diagram of 3 to 8 decoder is shown in fig. 4
- A , B and C are the inputs. (No. of inputs =3)
- No. of possible input combinations: $2^3=8$
- No. of Outputs : $2^3=8$, they are indicated by D_0 to D_7
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.

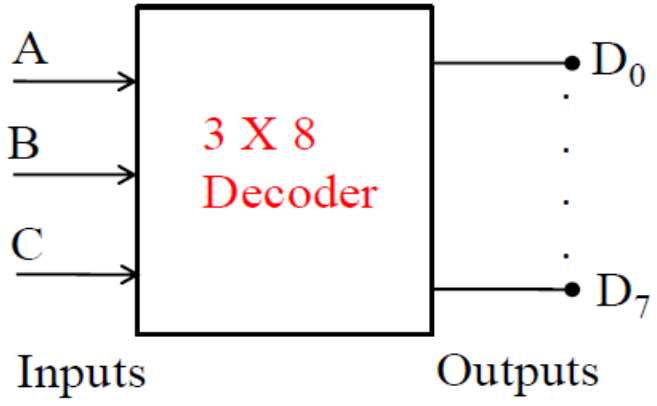


Fig. 4

TRUTH TABLE FOR 3 X 8 DECODER:

INPUTS			OUTPUTS								
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 = \overline{A}\,\overline{B}\,\overline{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \overline{A}\,\overline{B}\,C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \overline{A}\,B\,\overline{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \overline{A}\,B\,C$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\,\overline{B}\,\overline{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\,\overline{B}\,C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = A\,B\,\overline{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = A\,B\,C$

LOGIC DIAGRAM OF 3 X 8 DECODER:

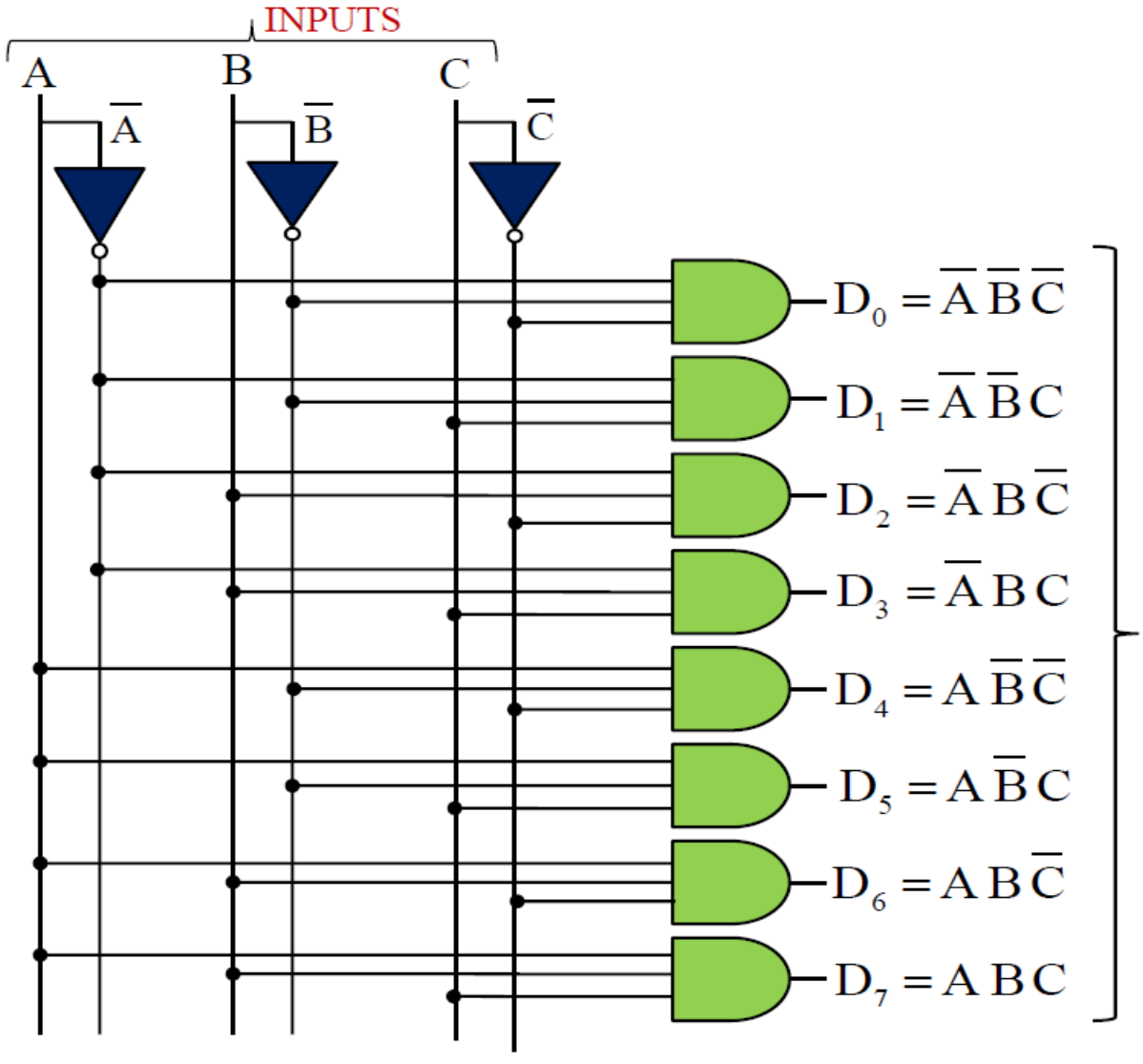


Fig. 5

3 x 8 Decoder From 2 x 4 Decoder:

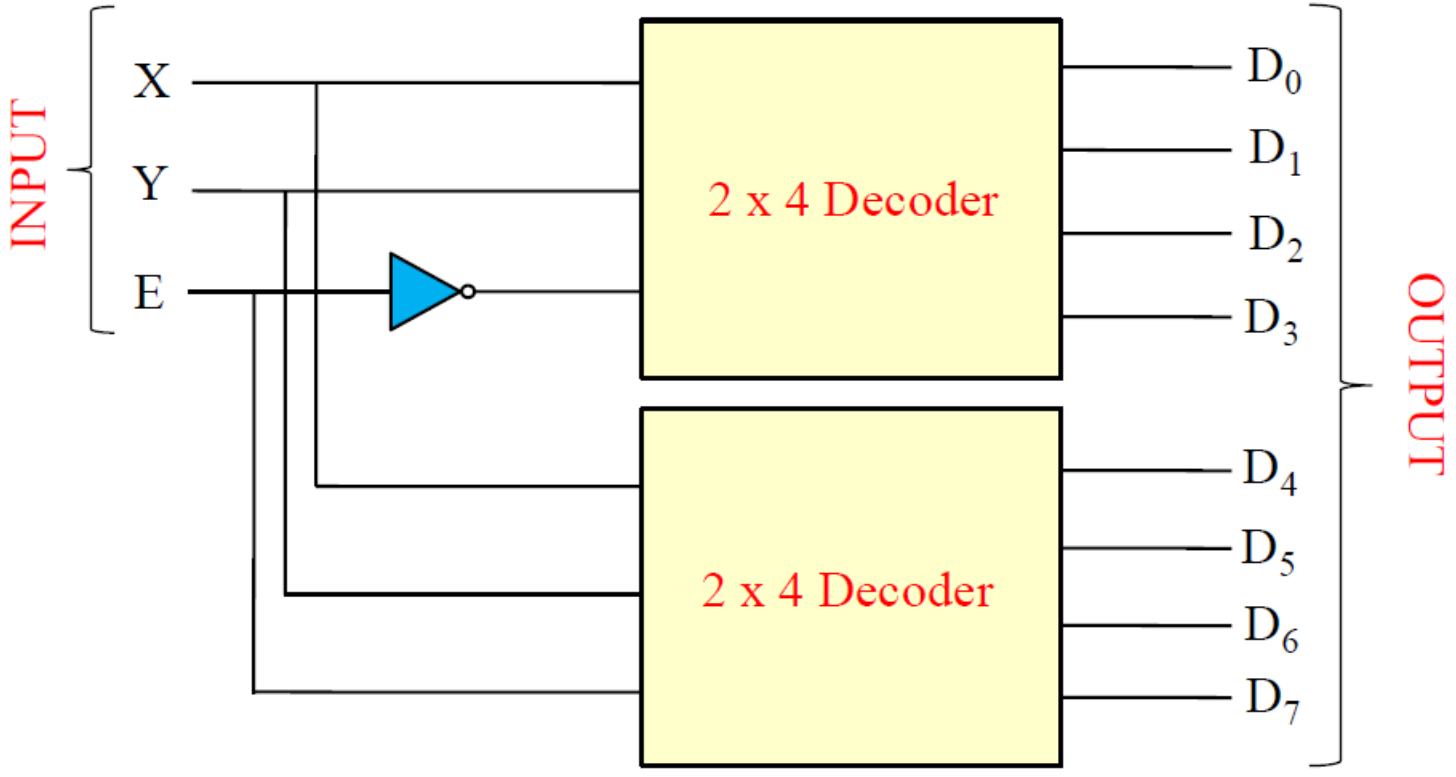


Fig. 6

Example: Implement the following multiple output function using a suitable Decoder.

$f_1(A, B, C) = \sum m(0,4,7) + d(2,3)$

$f_2(A, B, C) = \sum m(1,5,6)$

$f_3(A, B, C) = \sum m(0,2,4,6)$

Solution: f1 consists of don't care conditions. So we consider them to be logic 1.

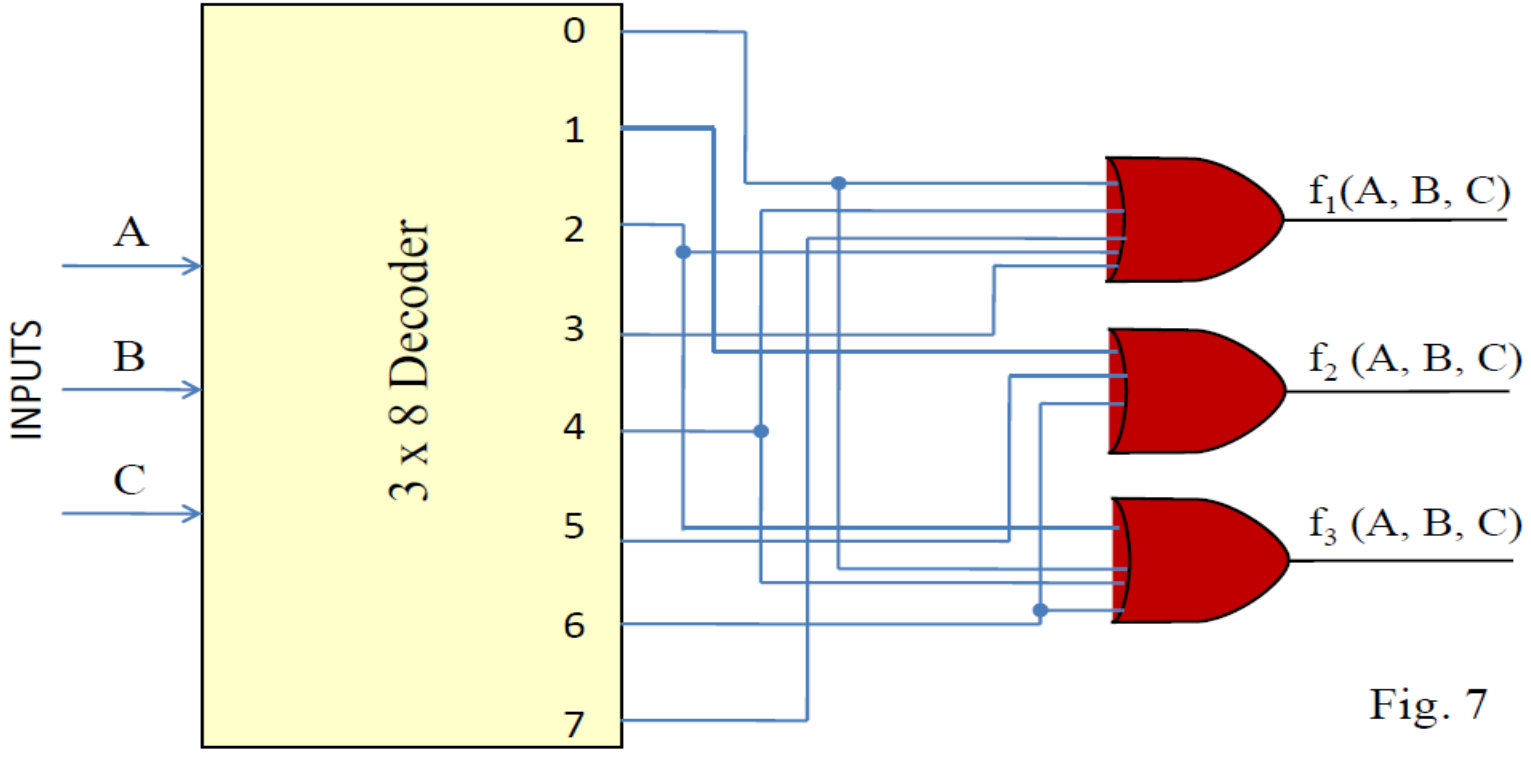


Fig. 7

EXAMPLE: A combinational circuit is defined by the following Boolean function. Design circuit with a Decoder and external gate.

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + x\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,z$$

(UPTU, 2004-05)

SOLUTION: STEP 1: Write the given function F_1 in SOP form

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + (y + \bar{y})\,x\,z$$

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + x\,y\,z + x\,\bar{y}\,z$$

$$F_1(x, y, z) = \Sigma m\,(0,5,7)$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + (y + \bar{y})\,\bar{x}\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,y\,z + \bar{x}\,\bar{y}\,z$$

$$F_2(x, y, z) = \Sigma m\,(1,3,6)$$

Boolean Function using Decoder:

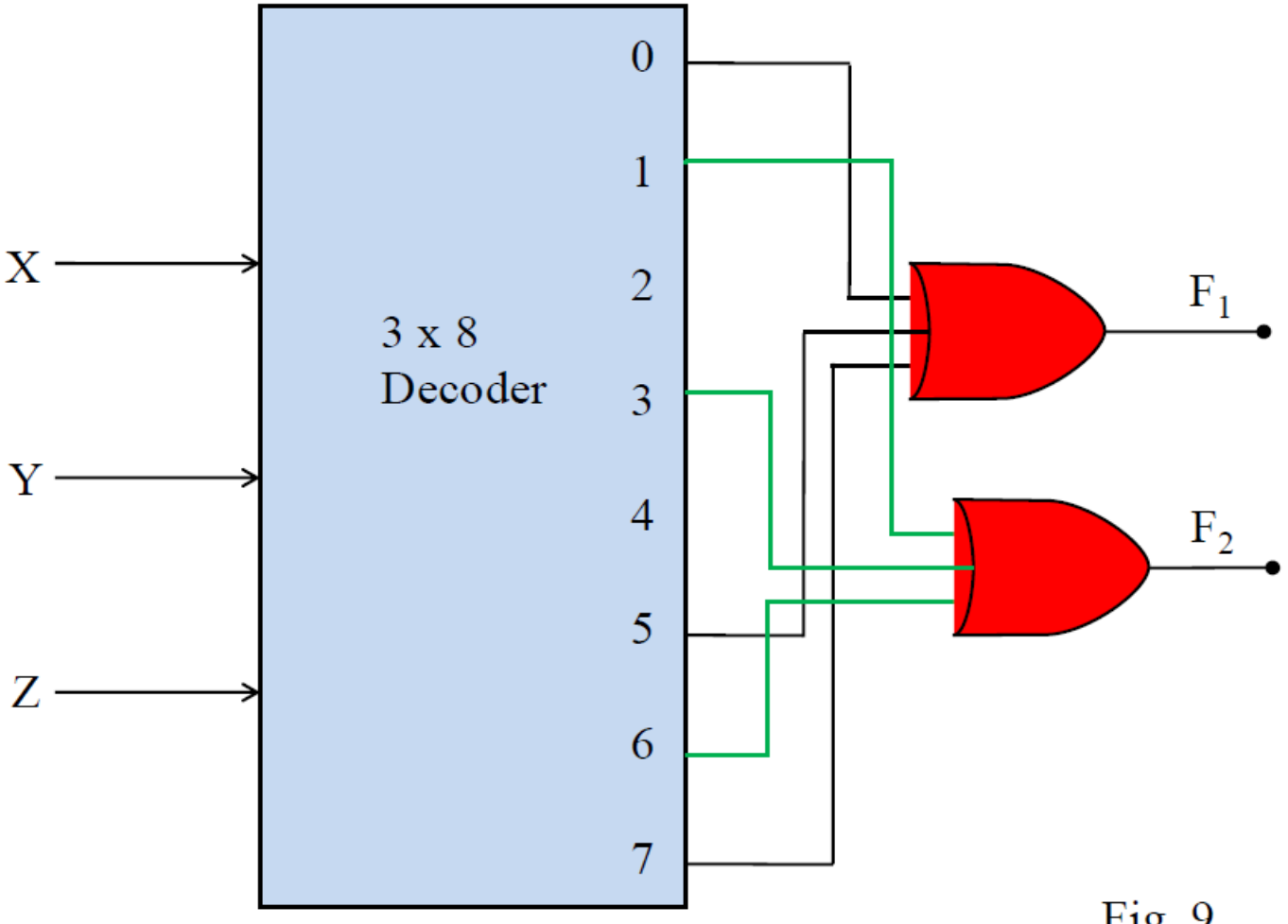


Fig. 9

ENCODER

- An Encoder is a combinational logic circuit.
- It performs the inverse operation of Decoder.
- The opposite process of decoding is known as Encoding.
- An Encoder converts an active input signal into a coded output signal.
- Block diagram of Encoder is shown in Fig.10. It has 'M' inputs and 'N' outputs.
- An Encoder has 'M' input lines, only one of which is activated at a given time, and produces an N-bit output code, depending on which input is activated.

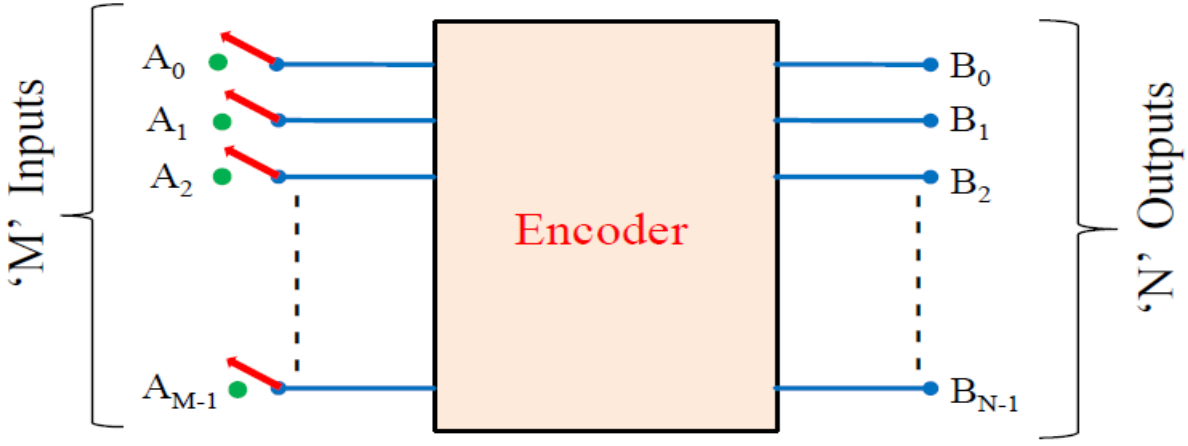


Fig. 10

- Encoders are used to translate the rotary or linear motion into a digital signal.
- The difference between Decoder and Encoder is that Decoder has Binary Code as an input while Encoder has Binary Code as an output.
- Encoder is an Electronics device that converts the analog signal to digital signal such as BCD Code.
- Types of Encoders

i. Priority Encoder
ii. Decimal to BCD Encoder
iii. Octal to Binary Encoder
iv. Hexadecimal to Binary Encoder

ENCODER

M=4
$M=2^2$
$M=2^N$
'M' is the input and
'N' is the output

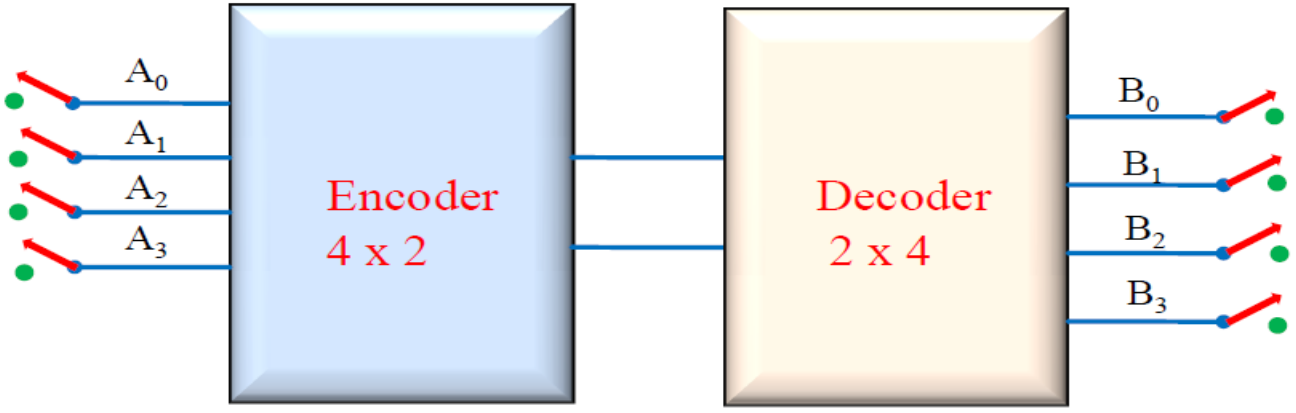


Fig. 11

ENCODER

M=4
$M=2^2$
$M=2^N$
'M' is the input and
'N' is the output

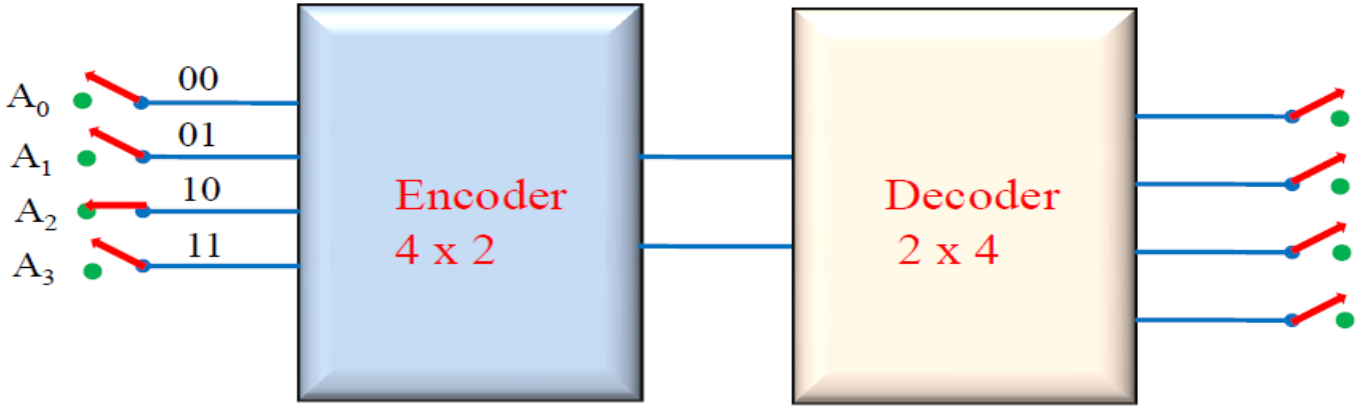


Fig. 12

ENCODER

M=4
$M=2^2$
$M=2^N$
'M' is the input and
'N' is the output

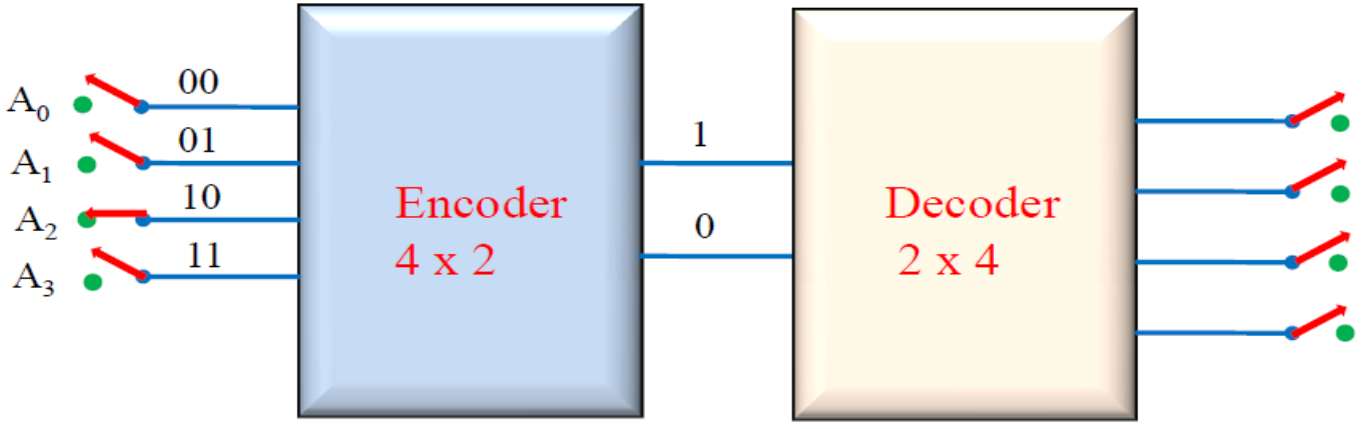


Fig. 13

ENCODER

M=4
$M=2^2$
$M=2^N$
'M' is the input and
'N' is the output

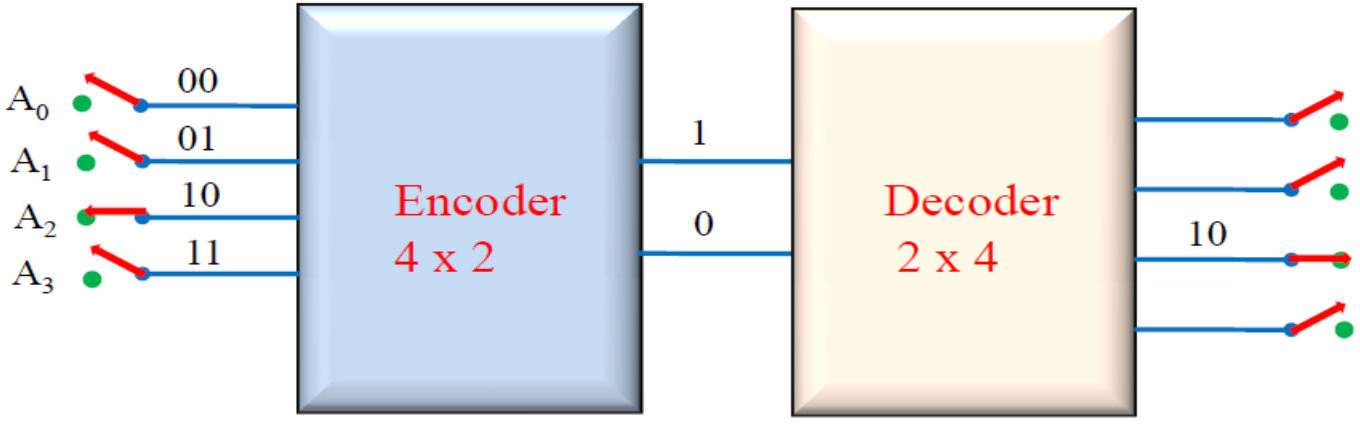
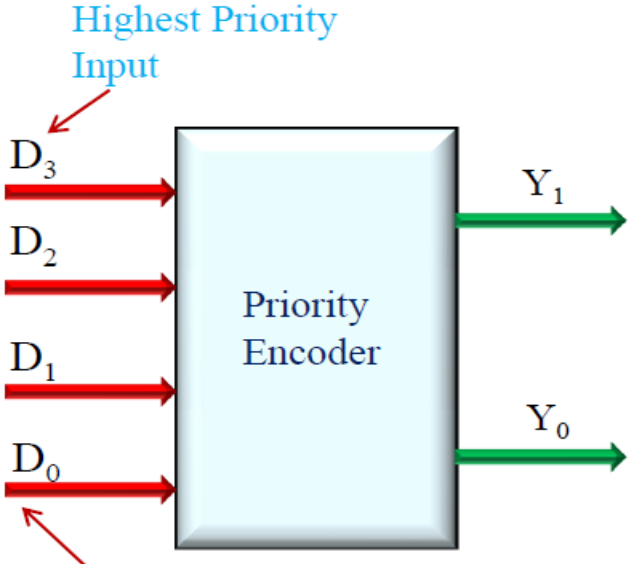


Fig. 14

PRIORITY ENCODER:

- As the name indicates, the priority is given to inputs line.
- If two or more input lines are high at the same time i.e 1 at the same time, then the input line with high priority shall be considered.
- Block diagram and Truth table of Priority Encoder are shown in fig.15



Block Diagram of Priority Encoder

Fig.15

TRUTH TABLE:

INPUTS				OUTPUTS		V
D_3	D_2	D_1	D_0	Y_1	Y_0	
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

- There are four inputs D_0, D_1,D_2, D_3 and two outputs Y_1 and Y_2.
- D_3 has highest priority and D_0 is at lowest priority.
- If D_3=1 irrespective of other inputs then output Y_1Y_0=11.
- D_3 is at highest priority so other inputs are considered as don't care.

K-map for Y_1 and Y_0

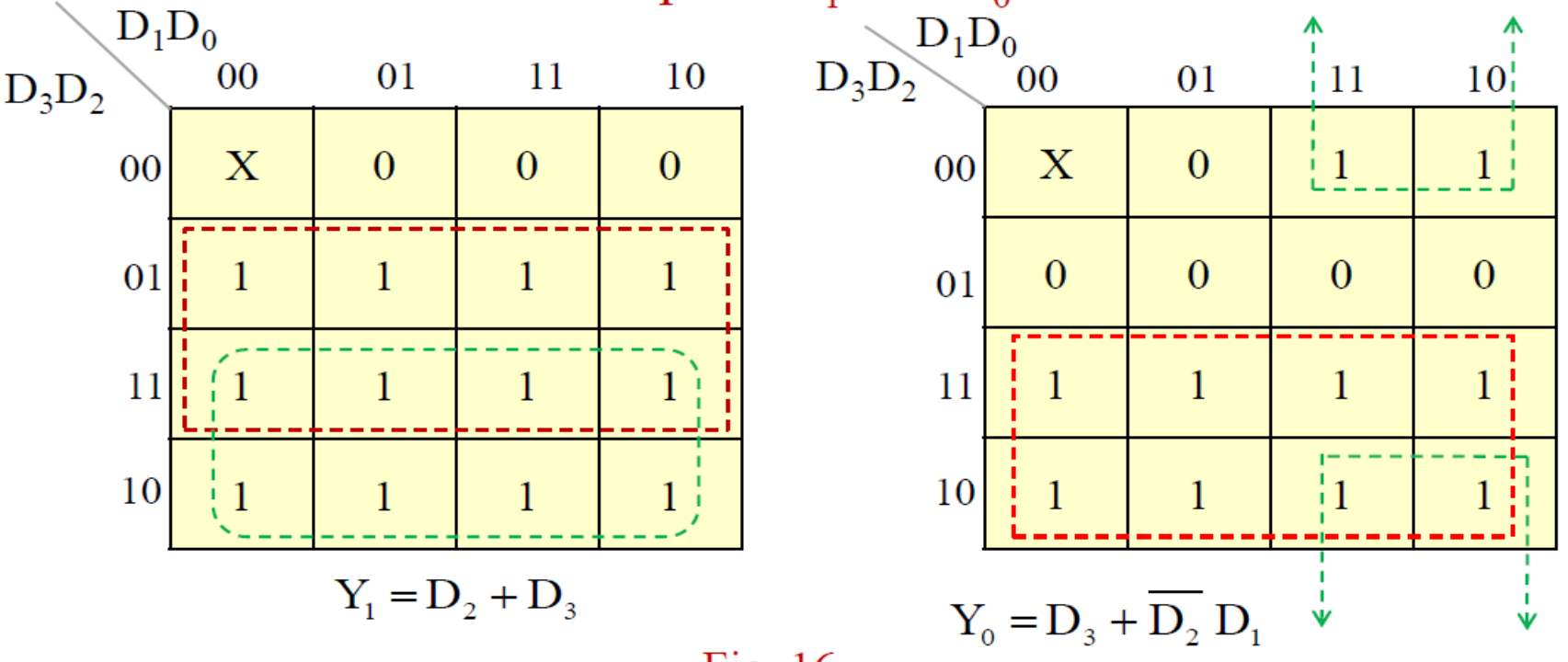


Fig. 16

LOGIC DIAGRAM OF PRIORITY ENCODER:

$$Y_1 = D_2 + D_3$$

$$Y_0 = D_3 + \overline{D_2}\, D_1$$

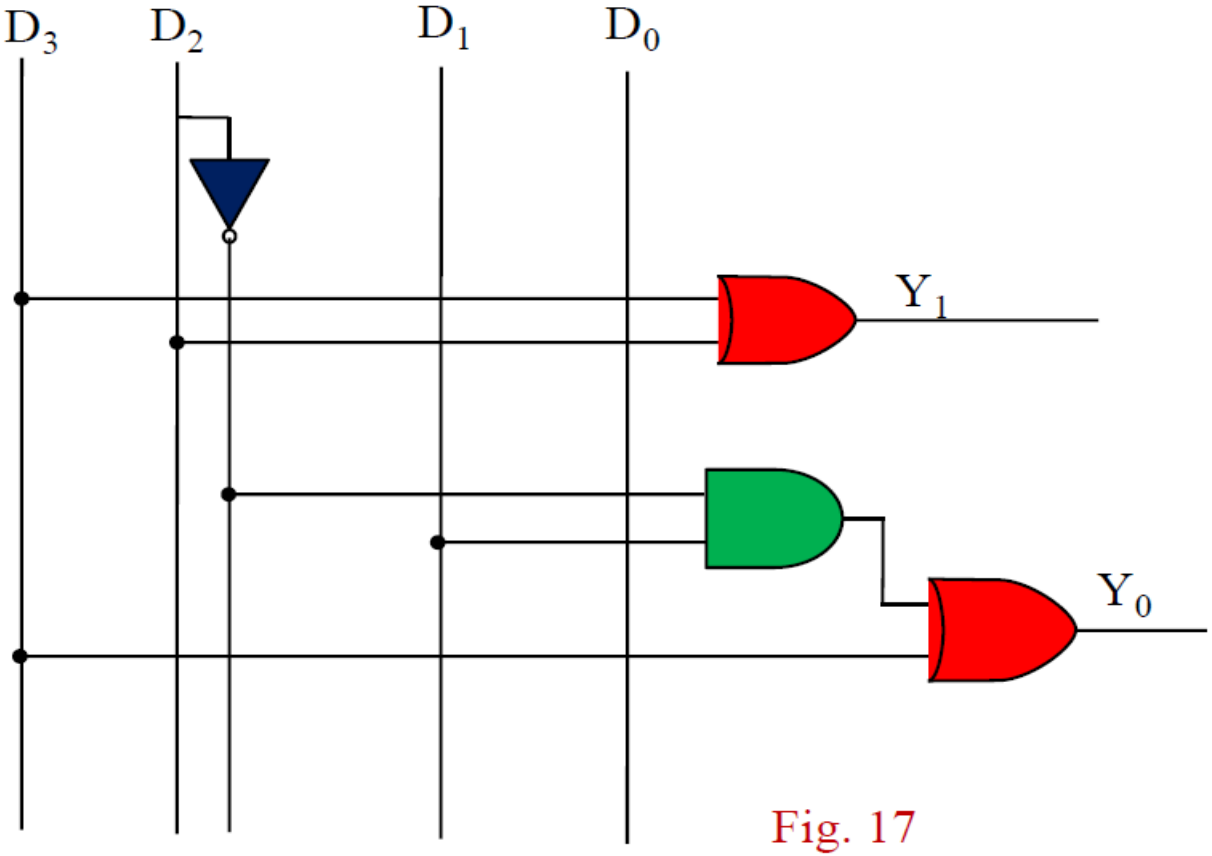


Fig. 17

DECIMAL TO BCD ENCODER:

- It has ten inputs corresponding to ten decimal digits (from 0 to 9) and four outputs (A,B,C,D) representing the BCD.
- The block diagram is shown in fig.18 and Truth table in fig.19

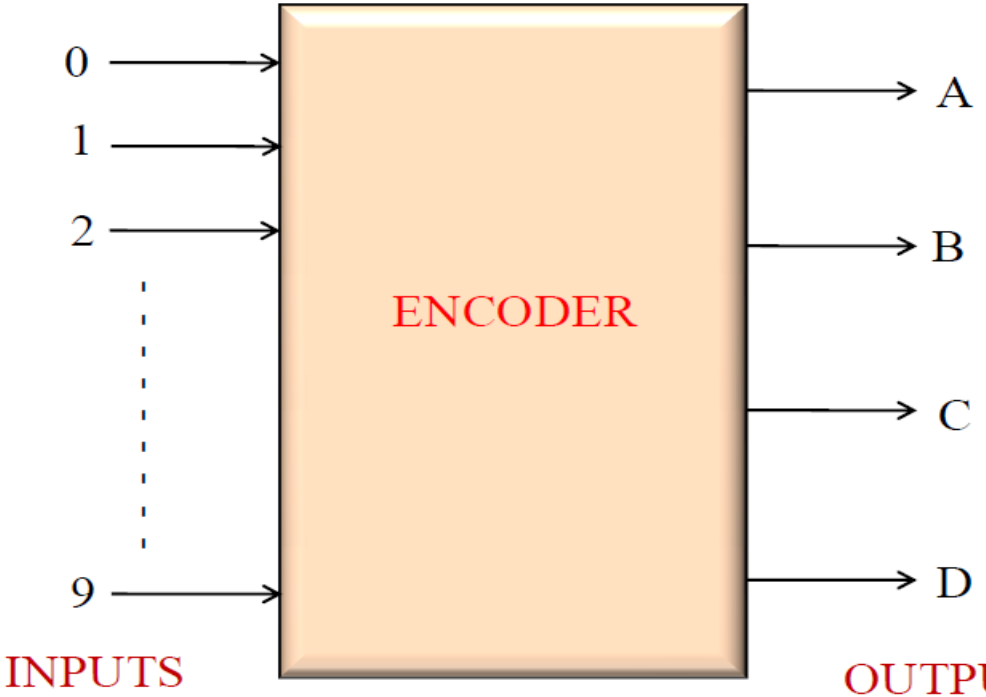


Fig. 18

Truth table:

INPUTS										BCD OUTPUTS			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Fig. 19

- From Truth Table it is clear that the output A is HIGH when input is 8 OR 9 is HIGH

  Therefore A=8+9
- The output B is HIGH when 4 OR 5 OR 6 OR 7 is HIGH

  Therefore B=4+5+6+7
- The output C is HIGH when 2 OR 3 OR 6 OR 7 is HIGH

  Therefore C=2+3+6+7
- Similarly D=1+3+5+7+9

  Logic Diagram is shown in fig.20

DECIMAL TO BCD ENCODER

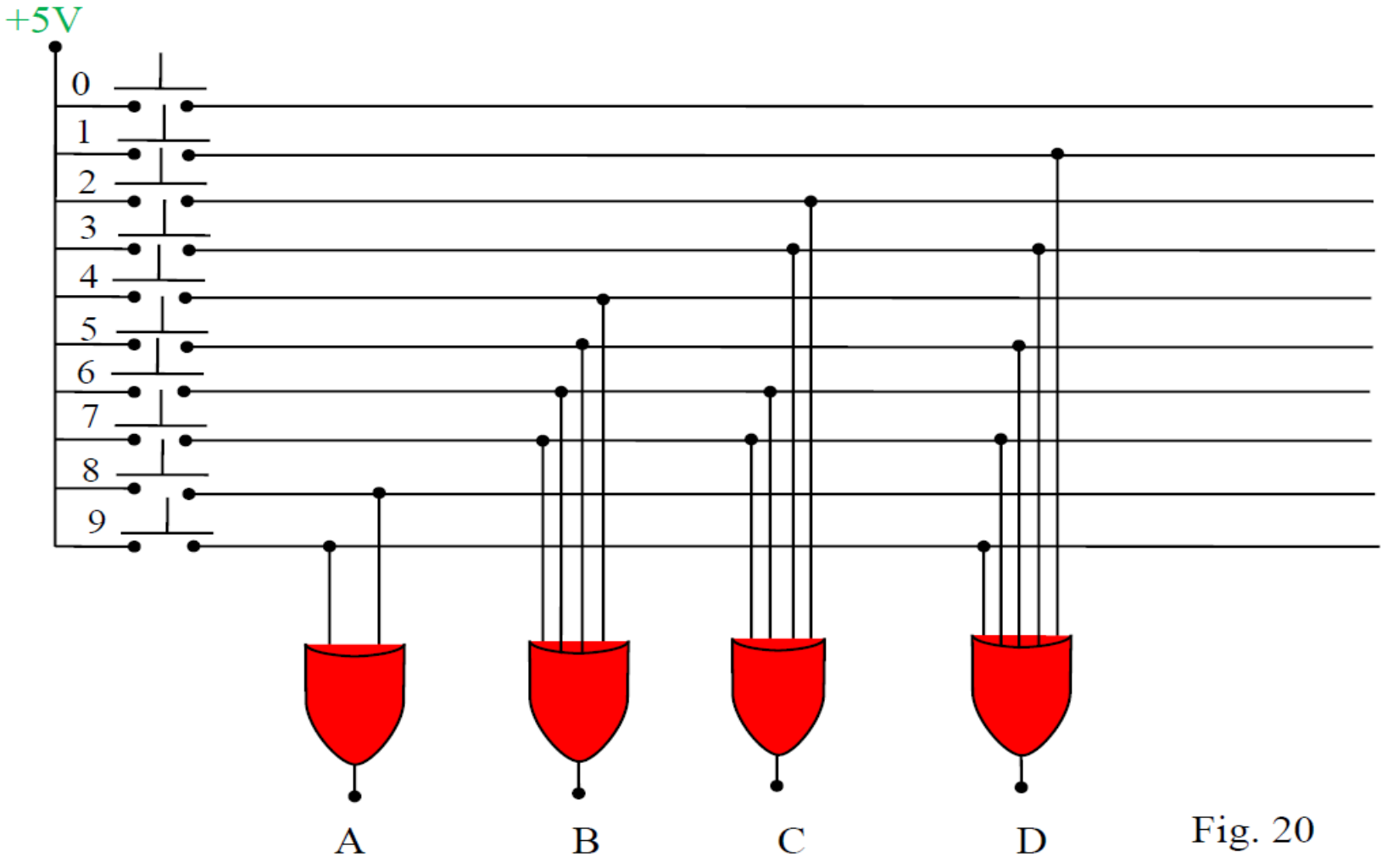


Fig. 20

OCTAL TO BINARY ENCODER:

- Block Diagram of Octal to Binary Encoder is shown in Fig. 21
- It has eight inputs and three outputs.
- Only one input has one value at any given time.
- Each input corresponds to each octal digit and output generates corresponding Binary Code.

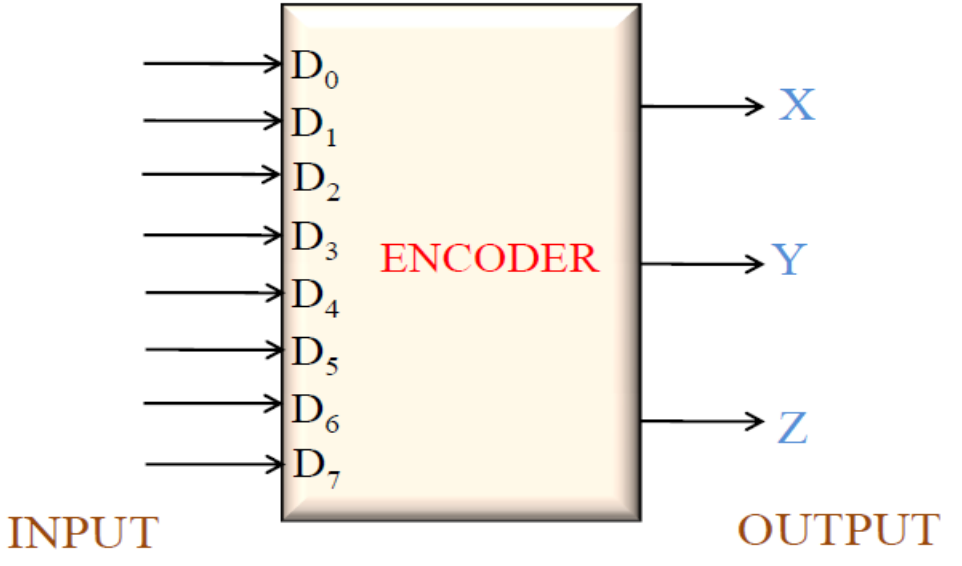


Fig. 21

TRUTH TABLE:

INPUT								OUTPUT		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Fig. 22

From Truth table:

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

- It is assume that only one input is HIGH at any given time. If two outputs are HIGH then undefined output will produced. For example D_3 and D_6 are HIGH, then output of Encoder will be 111. This output neither equivalent code corresponding to D_3 nor to D_6.
- To overcome this problem, priorities should be assigned to each input.
- Form the truth table it is clear that the output X becomes 1 if any of the digit D_4 or D_5 or D_6 or D_7 is 1.
- D_0 is considered as don't care because it is not shown in expression.
- If inputs are zero then output will be zero. Similarly if D_0 is one, the output will be zero.

$$X = D_4 + D_5 + D_6 + D_7$$
$$Y = D_2 + D_3 + D_6 + D_7$$
$$Z = D_1 + D_3 + D_5 + D_7$$

LOGIC DIAGRAM:

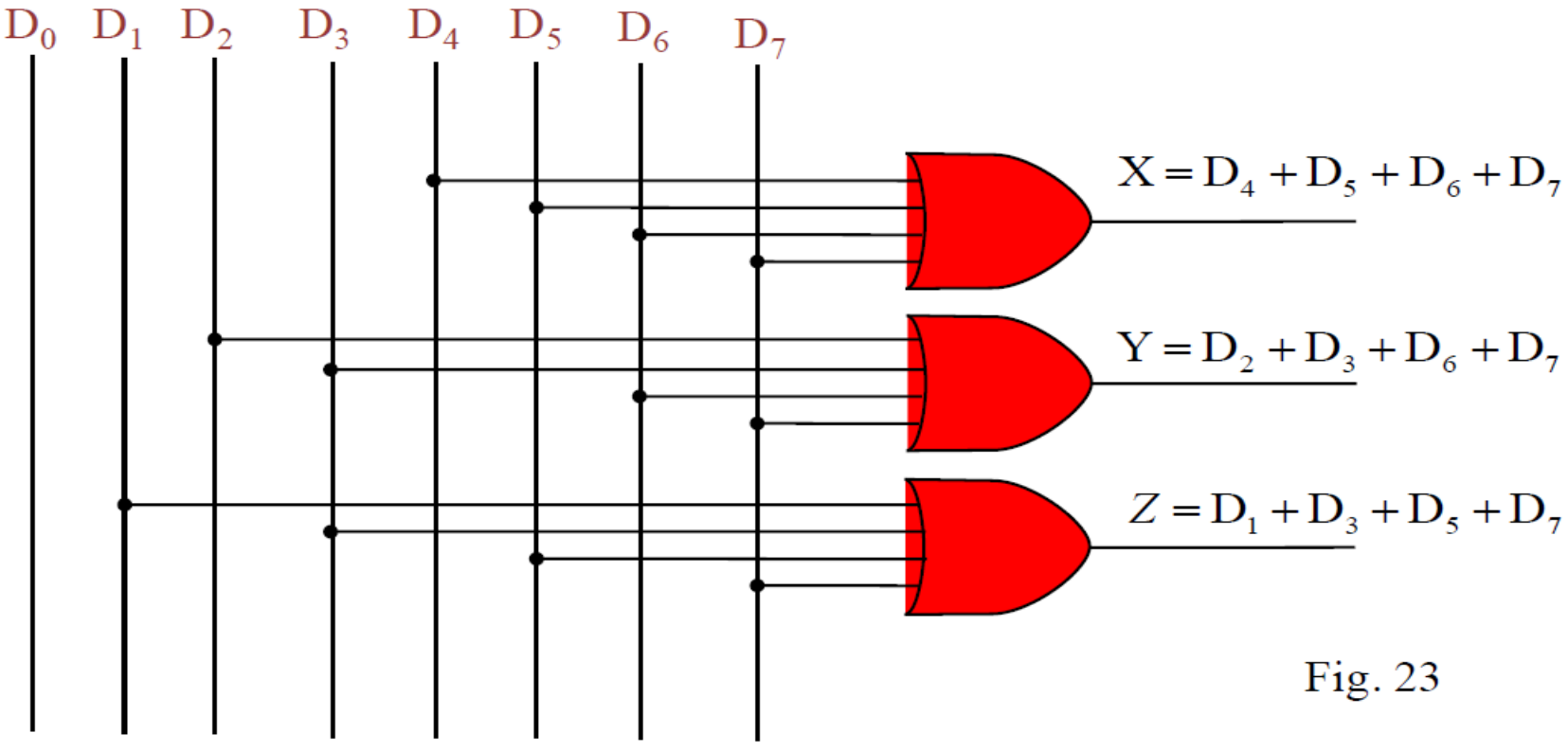


Fig. 23

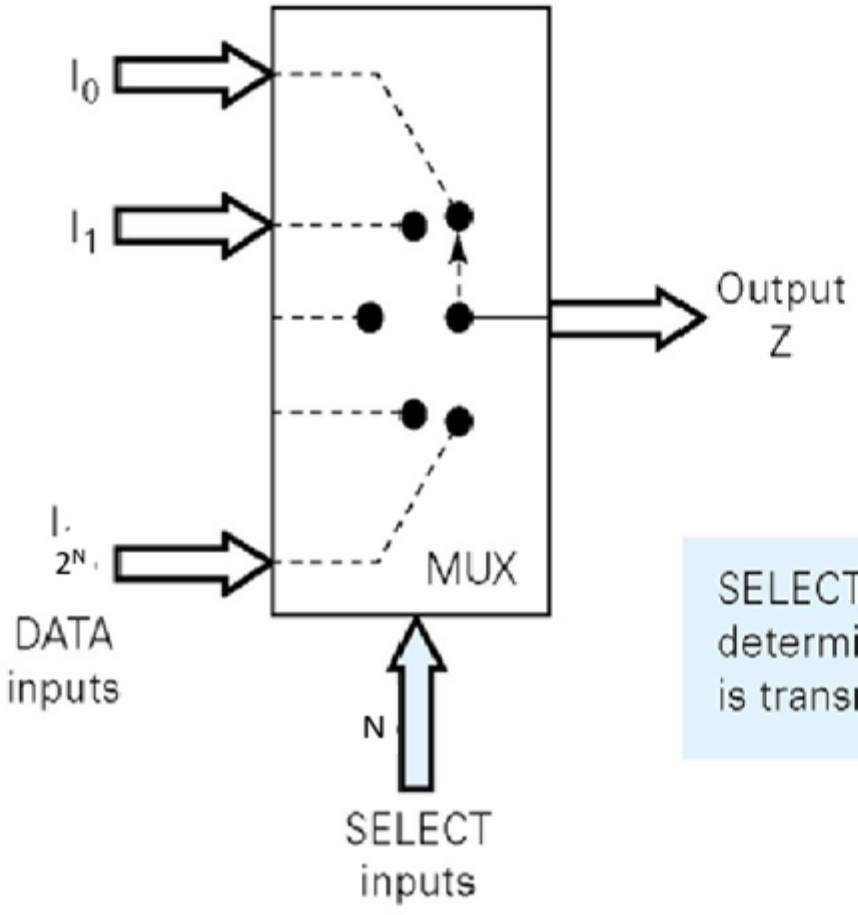
Multiplexers

A multiplexers (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.

The basic multiplexers has several data input lines and a single output line. It also has data- select inputs, which permit digital data on any one of the inputs to be switched to the output line.

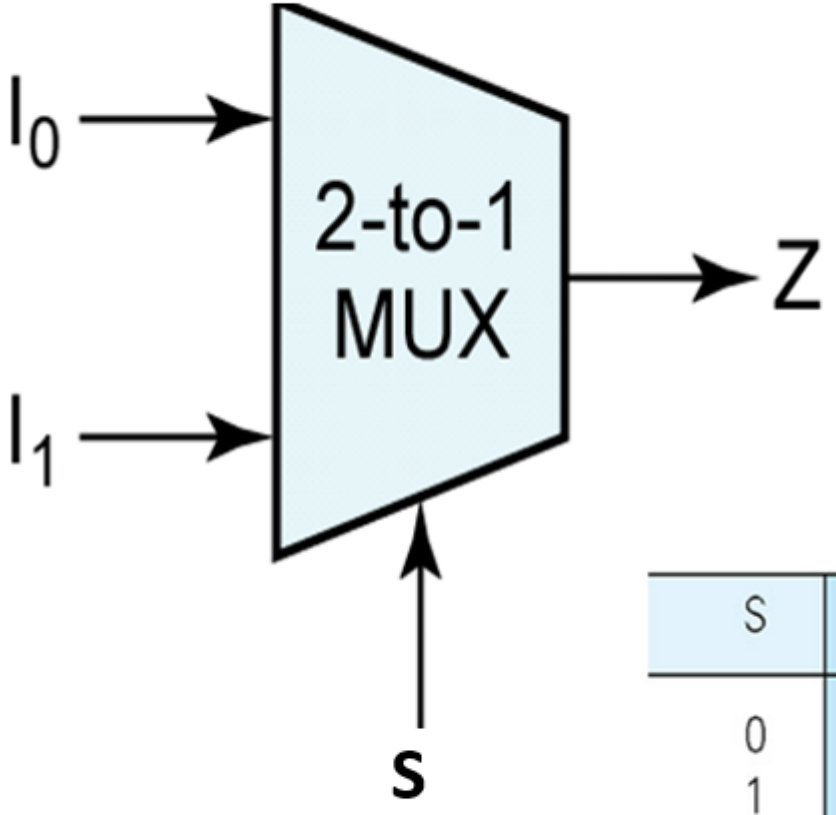
Block Diagram

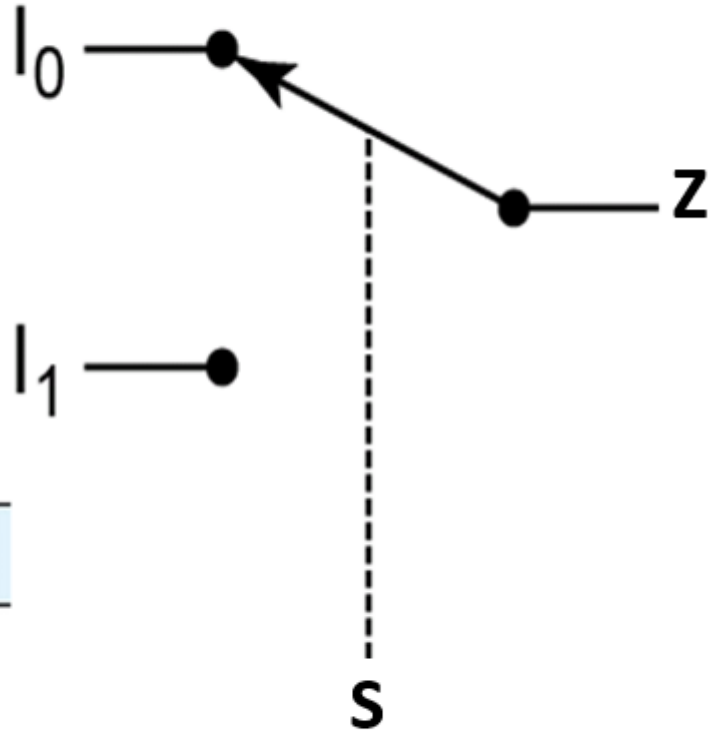
- A multiplexer has
  1. N control/select inputs
  2. 2^N data inputs
  3. 1 output
- A multiplexer routes (or connects) the selected data input to the output.
- The value of the control inputs determines the data input that is selected.



SELECT input code determines which input is transmitted to output Z.

2 to 1 mux

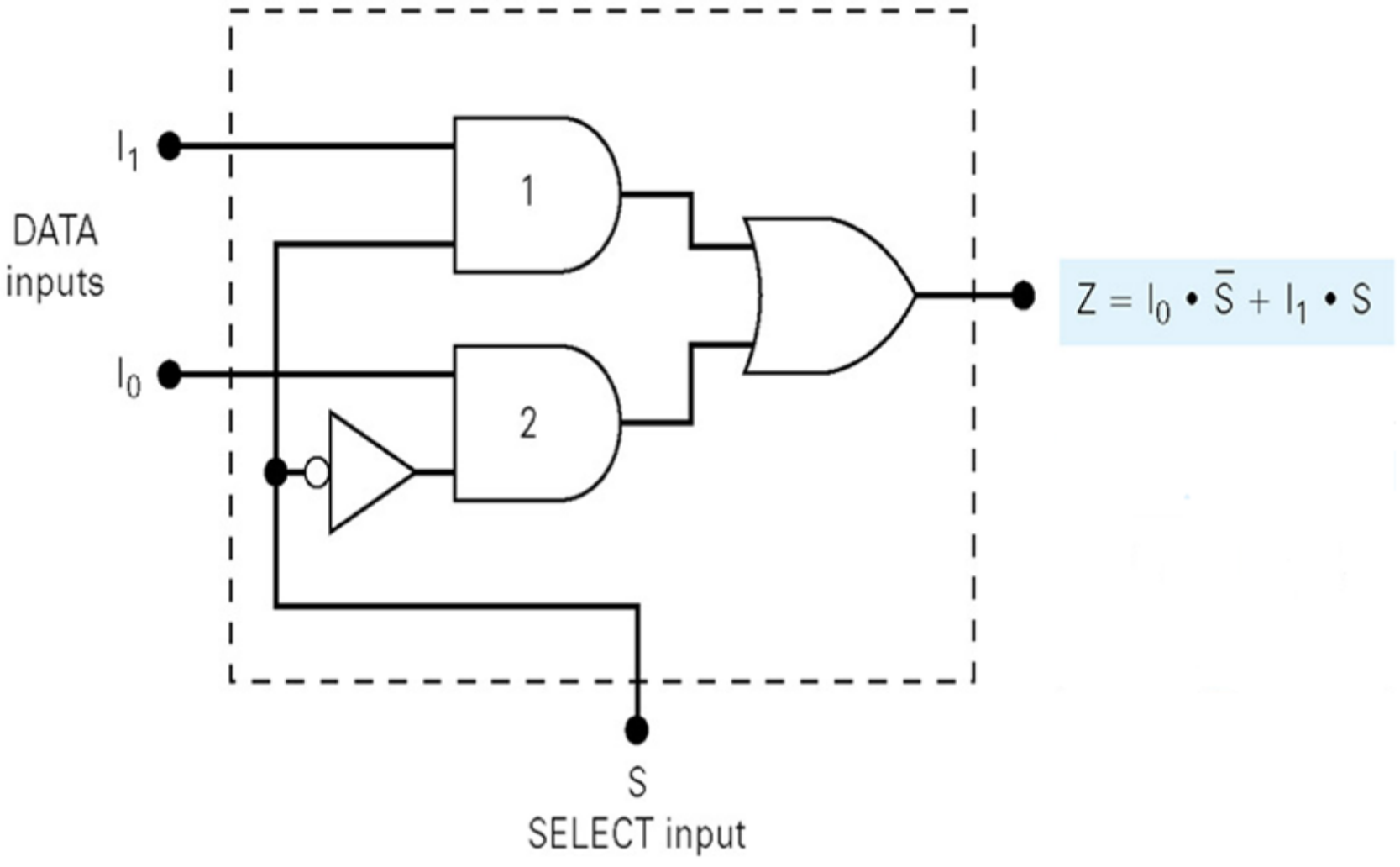




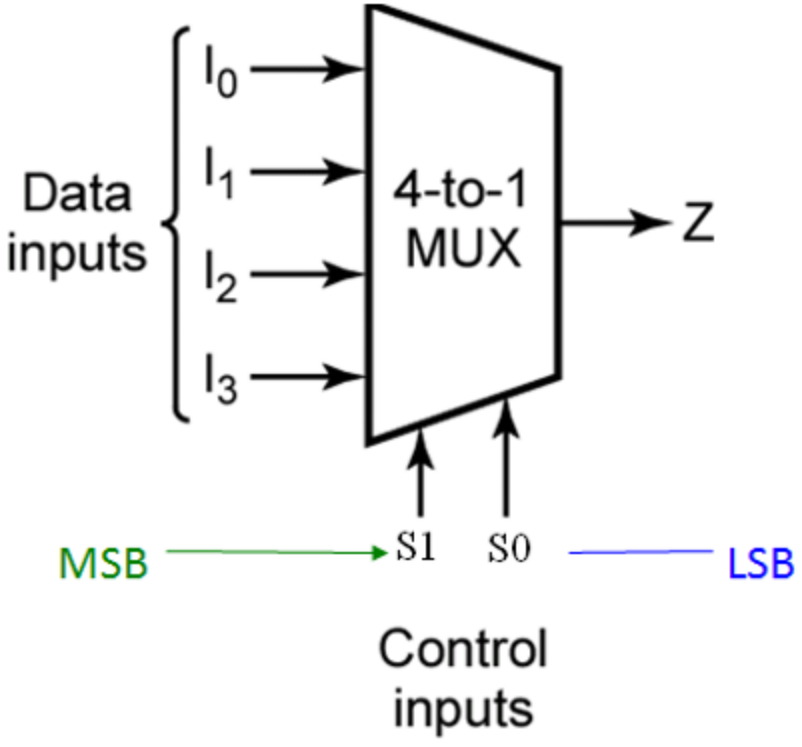
S	Output
0	$Z = I_0$
1	$Z = I_1$

$$Z = S'.I_0 + S.I_1$$

2 to 1 multiplexer



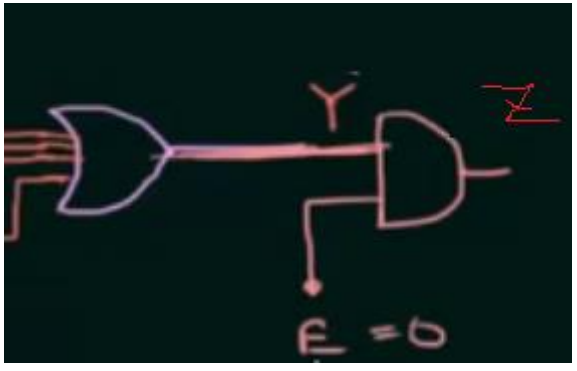
4 to 1Multiplexer



S1	S0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

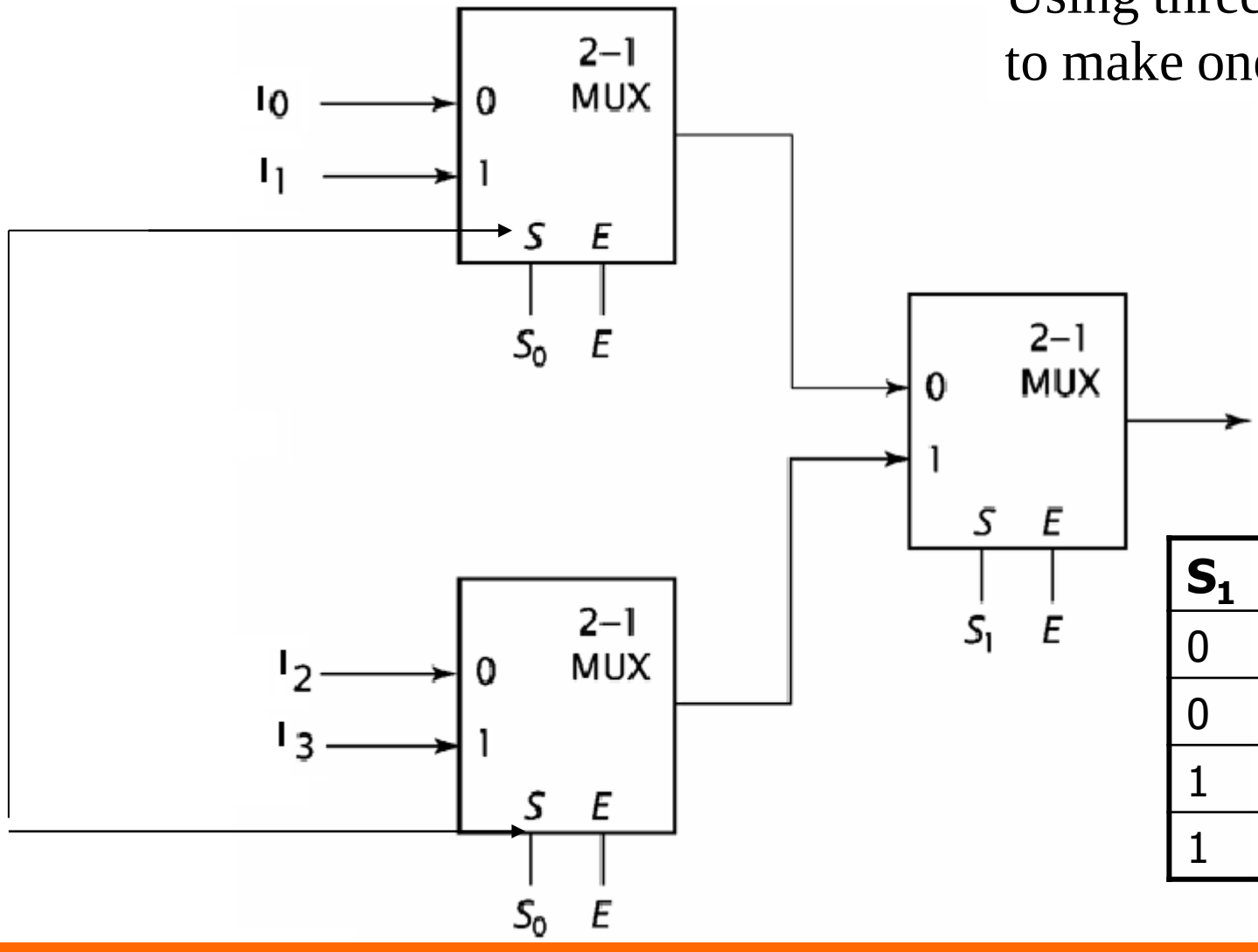
$$Z = S1'.S0'.I_0 + S1'.S0.I_1 + S1.S0'.I_2 + S1.S0.I_3$$

4 to 1 multiplexer



Cascading multiplexers

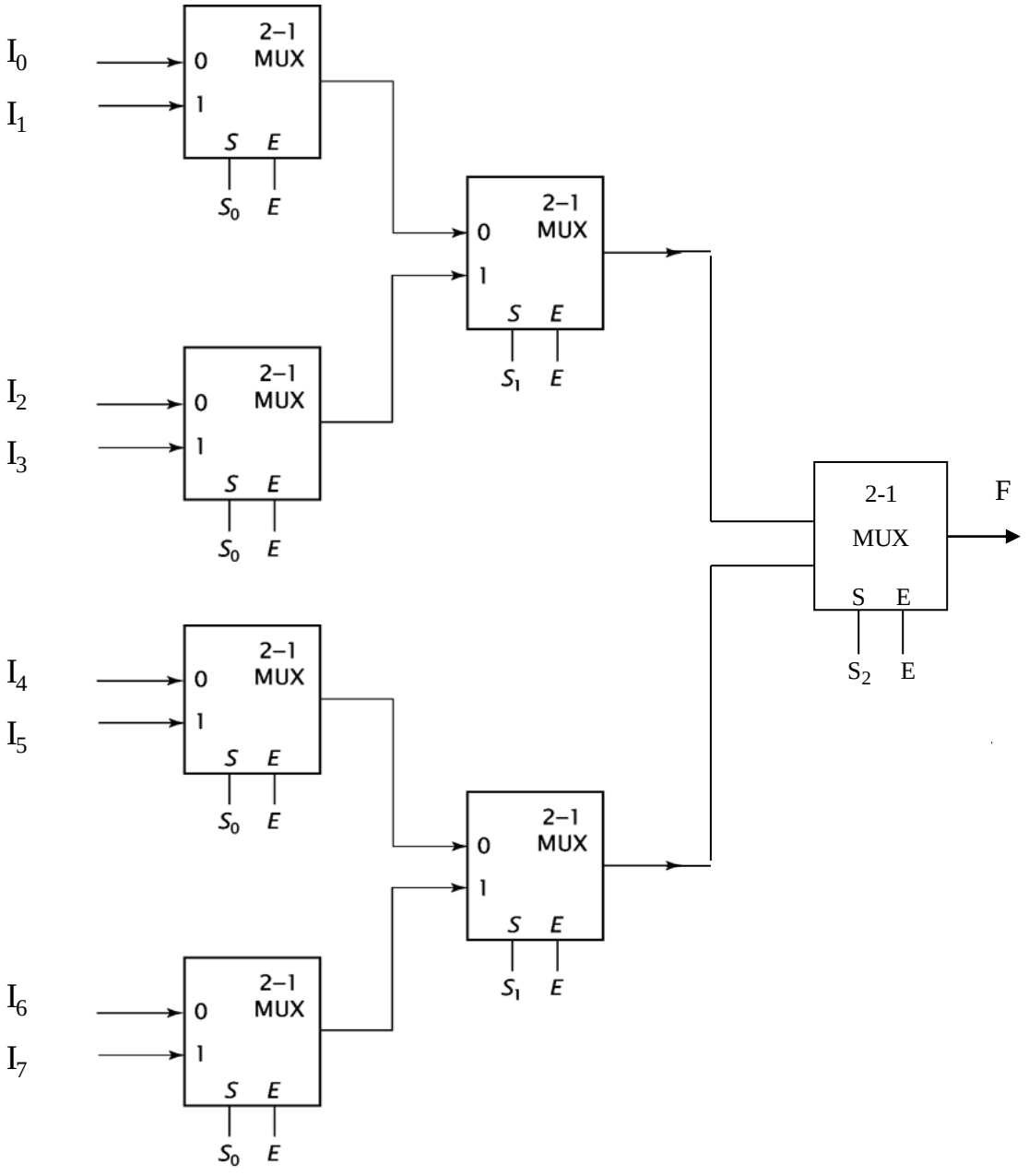
Using three 2-1 MUX
to make one 4-1 MUX



S_1	S_0	F
0	0	I0
0	1	I1
1	0	I2
1	1	I3

8-to-1 multiplexer using 2-to-1 multiplexers.

S_2	S_1	S_0	F
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

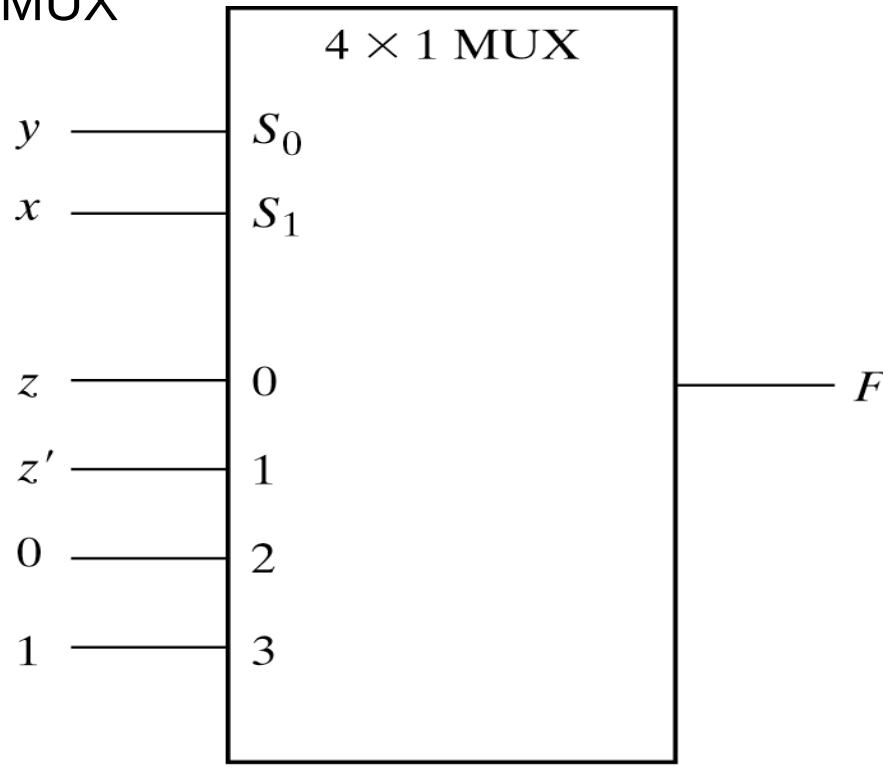


LOGIC FUNCTION GENERATION USING MUX

Implement the Boolean function F=x'y'z +x'yz' + xyz' + xyz using a 4 to 1 MUX

x	y	z	F		
0	0	0	0	$F = z$	0
0	0	1	1		
0	1	0	1	$F = z'$	1
0	1	1	0		
1	0	0	0	$F = 0$	2
1	0	1	0		
1	1	0	1	$F = 1$	3
1	1	1	1		

(a) Truth table



(b) Multiplexer implementation

Applications of Multiplexer

- Multiplexer are used in various fields where multiple data need to be transmitted using a single line. Following are some of the applications of multiplexers
- Communication system
- Telephone network
- Computer memory
- Transmission from the computer system of a satellite

Applications of De-Multiplexer

- De-multiplexer is used to connect a single source to multiple destinations. The main application area of de-multiplexer is communication system where multiplexer are used.
- Communication System
- ALU (Arithmetic Logic Unit)
- Serial to parallel converter

Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

- **UNIT-II: Sequential circuits, latches, Flip-Flops**

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

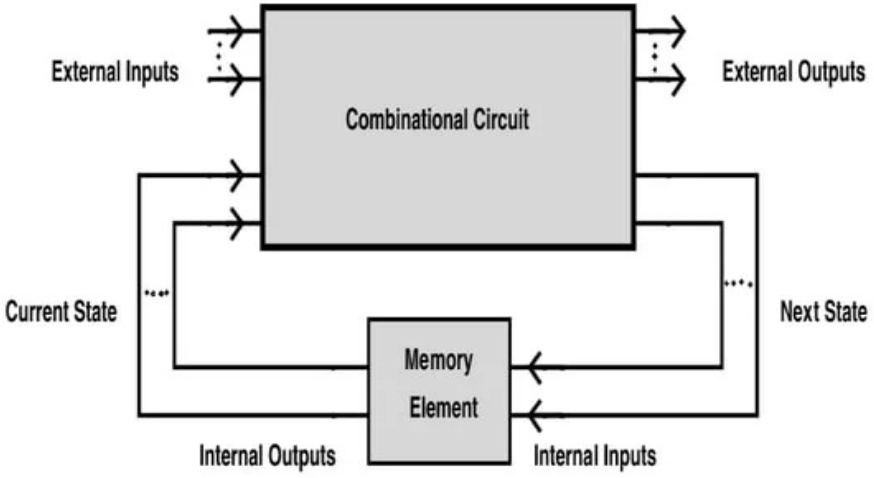
(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Sequential Circuits

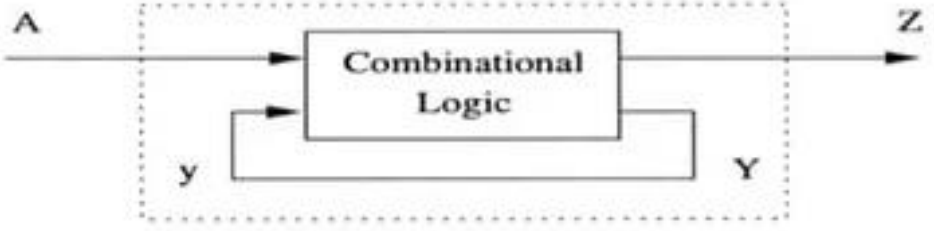
- Output depends on current as well as past inputs
- Depends on the history
- Have "memory" property
- Sequential circuit consists of
  - » Combinational circuit
  - » Feedback circuit
- Past input is encoded into a set of state variables
  - » Uses feedback (to feed the state variables)
    - – Simple feedback
    - – Uses flip flops



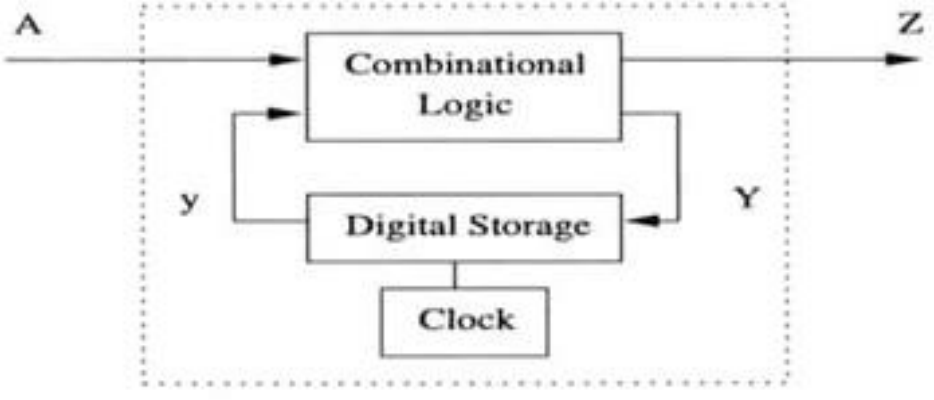
Types of Sequential Circuits

In Asynchronous sequential circuits the output of the logic circuit can change state at any time, as soon as any input changes its state whereas in the case of synchronous systems a signal namely clock signal is used to determine/control the exact time at which any output can change its state. These are also called as clocked sequential circuits.

Asynchronous

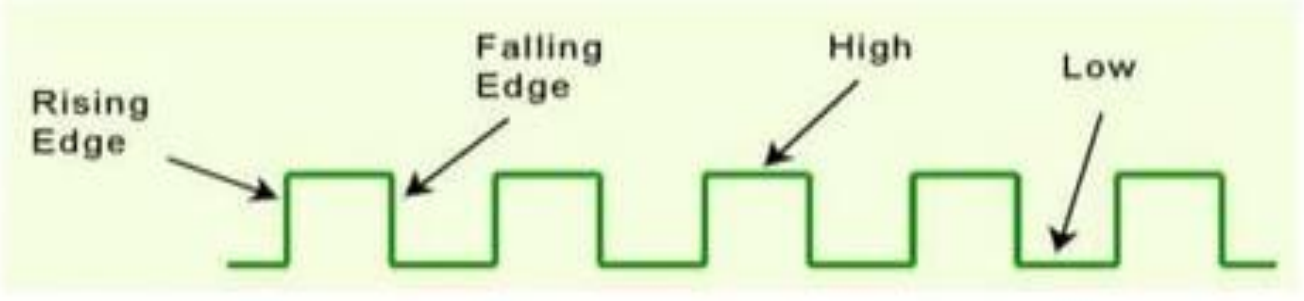


Synchronous



Sequential circuit= Combinational logic + Memory Elements

- The **memory element** is a device which can remember value indefinitely, or change value on command from its inputs.
- The memory element also has a clock input which provides timing for changing states.
- The feedback path is required for the circuit to have memory.
- State changes are controlled by clocks.
  A "clock" is a special circuit that sends electrical pulses through a circuit.
- Clocks produce electrical waveforms in generally some form of square wave
- Circuits can change state on the rising edge, falling edge, or when the clock pulse reaches its highest voltage.

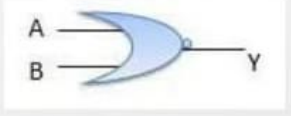


STORAGE ELEMENTS:

LATCHES

- are storage elements that operate with signal levels (rather than signal transitions)
- useful for storing binary information and for the design of asynchronous sequential circuits
- are the building blocks of flip-flops

Quick Revision
NOR gate & Truth Table:

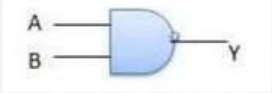


Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

SR LATCHES

- The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates,
- The two inputs are labeled **S** for *set* and **R** for *reset*.

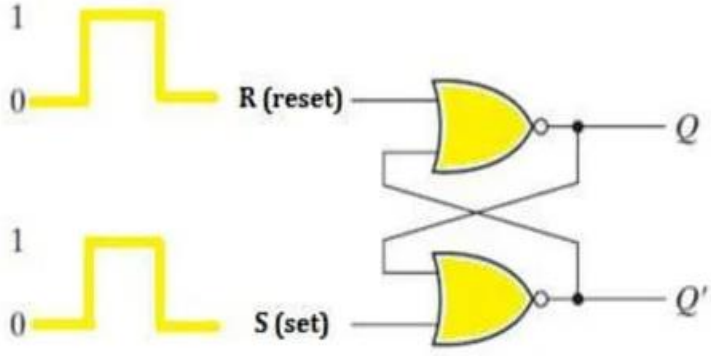
Quick Revision
NAND gate & Truth Table:



Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

SR LATCHES (WITH NOR GATES)

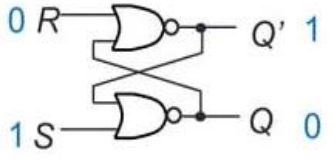
1



Logic Diagram

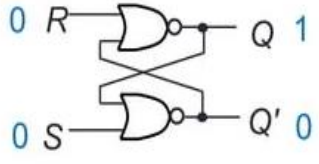
2

- when Q = 1 and Q' = 0, the latch is said to be in the set state
- *S*=HIGH (and *R*=LOW)



S	R	Q	Q'	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

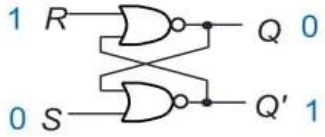
3 The inputs must go back to their normal conditions (S=0, R=0) before any changes occur



S	R	Q	Q'	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

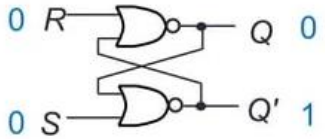
4

- When Q = 0 and Q' = 1, it is in the reset state
- *S*=LOW (and *R*=HIGH)



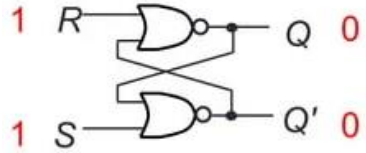
S	R	Q	Q'	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

5 The inputs must go back to their normal conditions (S=0, R=0) before any changes occur



S	R	Q	Q'	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

6 If both inputs HIGH ⇨ Q and Q' both LOW (invalid)!



S	R	Q	Q'	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

SR Latch with NOR gate (Conclusion)

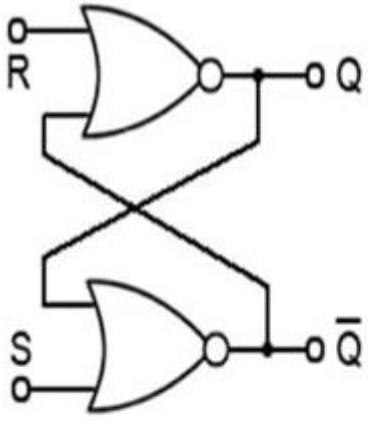
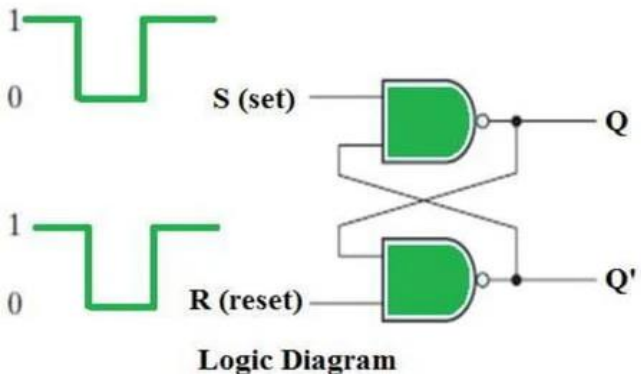


Table					
	R	S	Q	$\overline{Q}$	Comments
1.	0	1	1	0	Q is set to 1 by 1 on S
2.	0	0	1	0	No Change after set
3.	1	0	0	1	Q is reset to 0 by 1 on R
4.	0	0	0	1	No Change after reset
5.	1	1	0	0	Not allowed (both outputs at 0)
6.	0	0	?	?	If both inputs change from 11 to 00 outputs will be INDETERMINATE

- **When S = 0 and R = 0:** If we assume Q = 1 and Q' = 0 as initial condition, then output Q after input is applied would be Q = (R + Q')' = 1 and Q' = (S + Q)' = 0. Assuming Q = 0 and Q' = 1 as initial condition, then output Q after the input applied would be Q = (R + Q')' = 0 and Q' = (S + Q)' = 1. So it is clear that when both S and R inputs are LOW, the output is retained as before the application of inputs. (i.e. there is no state change).
- **When S = 1 and R = 0:** If we assume Q = 1 and Q' = 0 as initial condition, then output Q after input is applied would be Q = (R + Q')' = 1 and Q' = (S + Q)' = 0. Assuming Q = 0 and Q' = 1 as initial condition, then output Q after the input applied would be Q = (R + Q')' = 1 and Q' = (S + Q)' = 0. So in simple words when S is HIGH and R is LOW, output Q is HIGH.
- **When S = 0 and R = 1:** If we assume Q = 1 and Q' = 0 as initial condition, then output Q after input is applied would be Q = (R + Q')' = 0 and Q' = (S + Q)' = 1. Assuming Q = 0 and Q' = 1 as initial condition, then output Q after the input applied would be Q = (R + Q')' = 0 and Q' = (S + Q)' = 1. So in simple words when S is LOW and R is HIGH, output Q is LOW.
- **When S = 1 and R =1 :** No matter what state Q and Q' are in, application of 1 at input of NOR gate always results in 0 at output of NOR gate, which results in both Q and Q' set to LOW (i.e. Q = Q'). LOW in both the outputs basically is wrong, so this case is invalid

SR LATCHES (WITH NAND GATES)

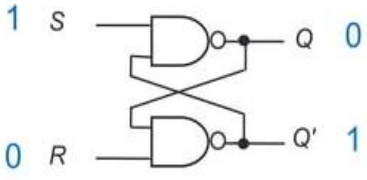
1



Logic Diagram

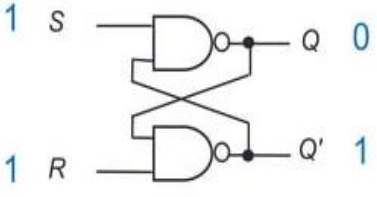
2

- when Q = 0 and Q' = 1, the latch is said to be in the set state
- *R=LOW* (and *S*=HIGH)



S	R	Q	Q'	
1	0	0	1	initial
1	1	0	1	(afer *S*=1, *R*=0)
0	1	1	0	
1	1	1	0	(after *S*=0, *R*=1)
0	0	1	1	invalid!

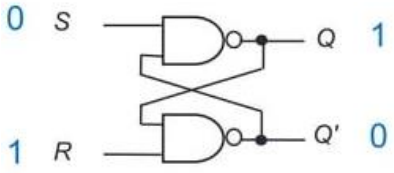
3 The inputs must go back to their normal conditions (S=1, R=1)before any changes occur



S	R	Q	Q'	
1	0	0	1	initial
1	1	0	1	(afer *S*=1, *R*=0)
0	1	1	0	
1	1	1	0	(after *S*=0, *R*=1)
0	0	1	1	invalid!

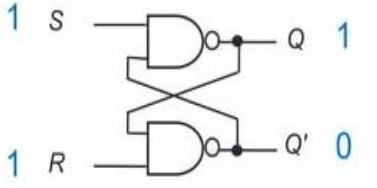
4

- When Q = 0 and Q' = 1, it is in the reset state
- *R=LOW* (and *S*=HIGH)



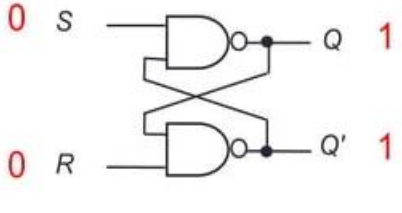
S	R	Q	Q'	
1	0	0	1	initial
1	1	0	1	(afer *S*=1, *R*=0)
0	1	1	0	
1	1	1	0	(after *S*=0, *R*=1)
0	0	1	1	invalid!

5 The inputs must go back to their normal conditions (S=1, R=1)before any changes occur



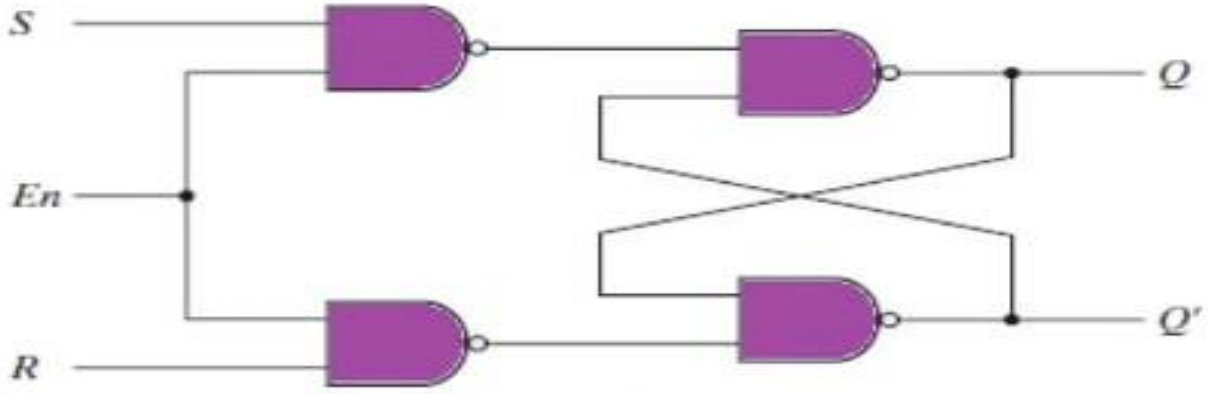
S	R	Q	Q'	
1	0	0	1	initial
1	1	0	1	(afer *S*=1, *R*=0)
0	1	1	0	
1	1	1	0	(after *S*=0, *R*=1)
0	0	1	1	invalid!

6 If both inputs LOW ⇨ *Q* and *Q'* both LOW (invalid)!



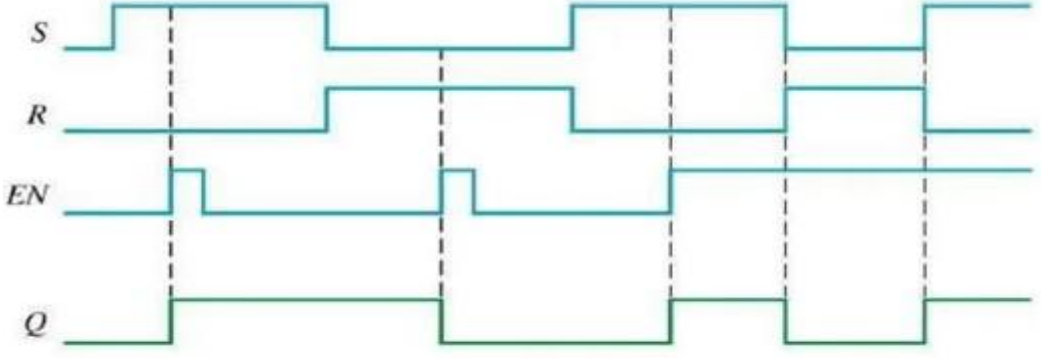
S	R	Q	Q'	
1	0	0	1	initial
1	1	0	1	(afer *S*=1, *R*=0)
0	1	1	0	
1	1	1	0	(after *S*=0, *R*=1)
0	0	1	1	invalid!

SR LATCHES WITH CONTROL INPUT

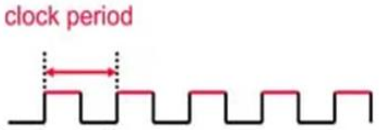


Logic Diagram

En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate



Clock:-



- A clock is a special device that whose output continuously alternates between 0 and 1.
- The time it takes the clock to change from 1 to 0 and back to 1 is called the clock period, or clock cycle time.
- The clock frequency is the inverse of the clock period. The unit of measurement for frequency is the hertz.
- Clocks are often used to synchronize circuits.

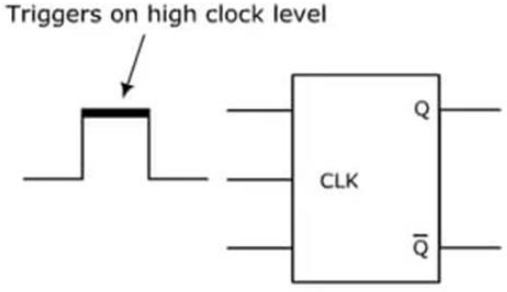
Triggering

- Sequential circuits are dependant on clock pulses applies to their inputs.
- The result of flip-flop responding to a clock input is called **clock pulse triggering**, of which there are four types. Each type responds to a clock pulse in one of four ways :-

1. High level triggering
2. Low level triggering
3. Positive edge triggering
4. Negative edge triggering

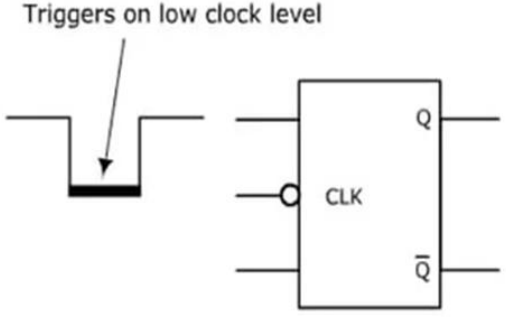
High Level Triggering

One type of flip-flop responds to a clock signal during the time at which it is in the logic High state. This type is identified by a straight lead at the clock input, as shown below.



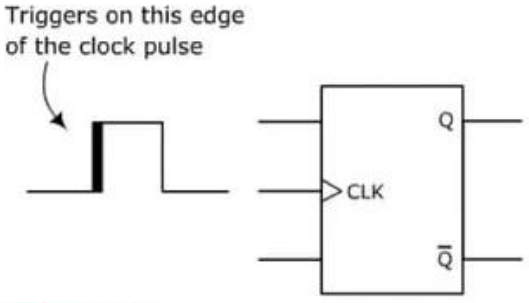
Low Level Triggering

Another type of flip-flop responds to a clock signal during the time at which it is in the logic Low state. This type is identified by a clock input lead with a low-state indicator bubble, as shown below.



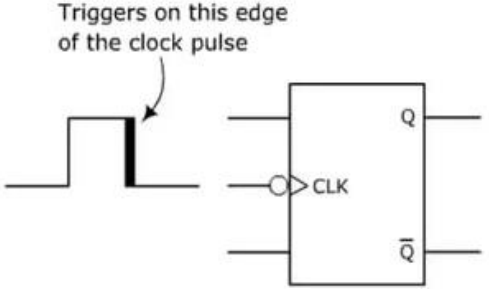
Positive Edge Triggering

A third type of flip-flop responds to a clock signal during the low-to-high transition of a clock pulse. This type is identified by a clock input lead with a triangle, as shown below.



Negative Edge Triggering

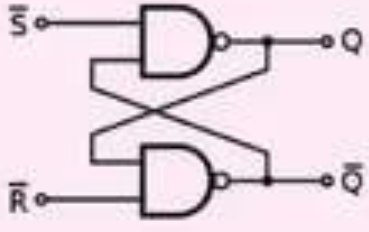
The fourth type of flip-flop responds to a clock signal during the high-to-low transition of a clock pulse. This type is identified by a clock input lead with a low-state indicator and triangle, as shown below.



Difference between both....?

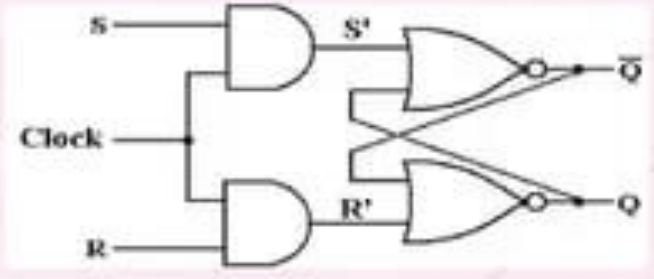
Latches..

- Both are same but there is a little difference between both.
- Latches are the building blocks of sequential circuits.
- latches can be built from gates.
- latch does not have a *clock signal.*



Flip Flop..

- flip-flops are also the building blocks of sequential circuits.
- Flip-flops can be built from latches.
- A flip-flop always has a *clock Signal*

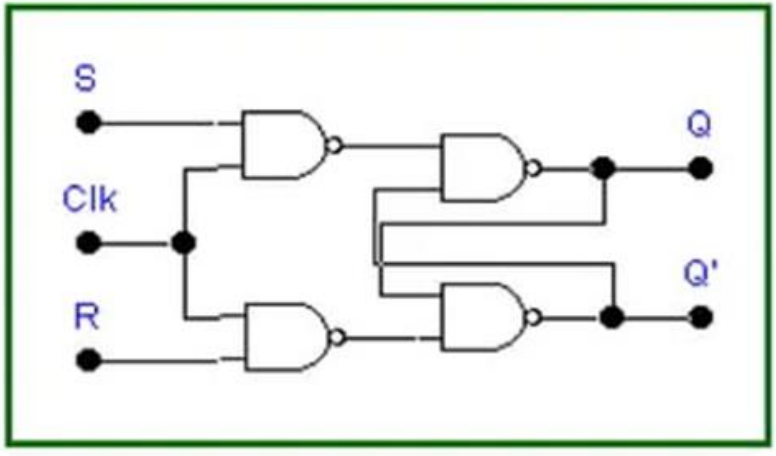


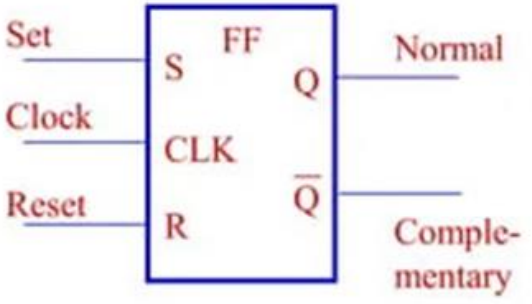
Flip-flops

- A flip-flop is a bi-stable device, with inputs, that remains in a given state as long as power is applied and until input signals are applied to cause its output to change.
- There are four basic different types of flip-flops:
  - SR
  - D
  - JK
  - T

CLOCKED R-S FLIP-FLOP

Symbols:



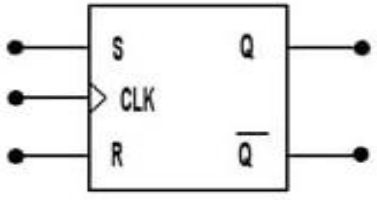


Truth Table:

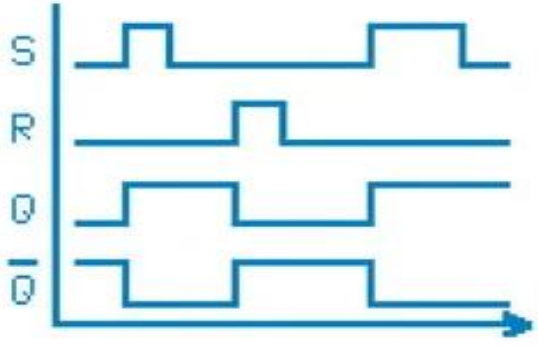
Clk	S	R	Q_n	$\overline{Q_n}$
1 ↑ ↓	0	0	No change	
1 ↑ ↓	0	1	0	1
1 ↑ ↓	1	0	1	0
1 ↑ ↓	1	1	Not Allowed	

POSITIVE EDGE TRIGGERED R-S FLIP-FLOP

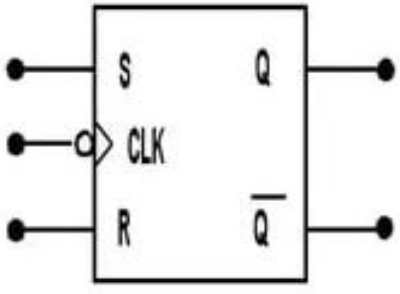
TIMING DIAGRAMS



CLK	R	S	Q
↑	0	0	NO CHG
↑	0	1	SET
↑	1	0	RESET
↑	1	1	ILLEGAL



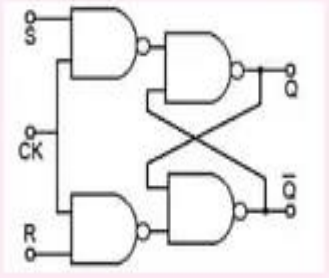
NEGATIVE EDGE TRIGGERED R-S FLIP-FLOP



CLK	R	S	Q
↓	0	0	NO CHG
↓	0	1	SET
↓	1	0	RESET
↓	1	1	ILLEGAL

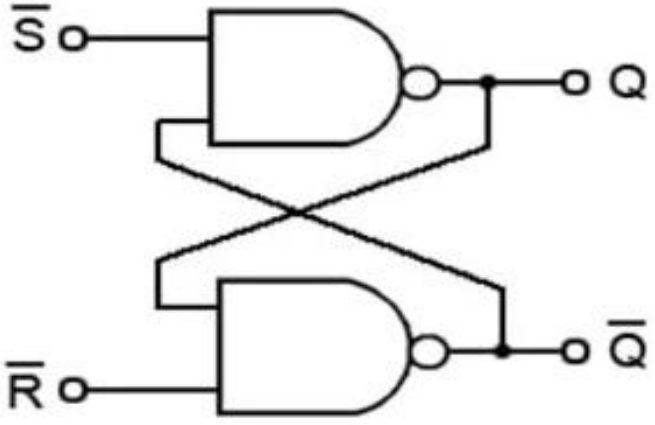
CLOCKED R-S FLIP-FLOP or SR Flip-flop

Logic Diagram:



Truth Table:

Clk	S	R	Q	Q'	
1	1	1	1	1	Not Allowed
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	0	0	Q	Q'	Memory state
0	0	0	1	0	Memory state

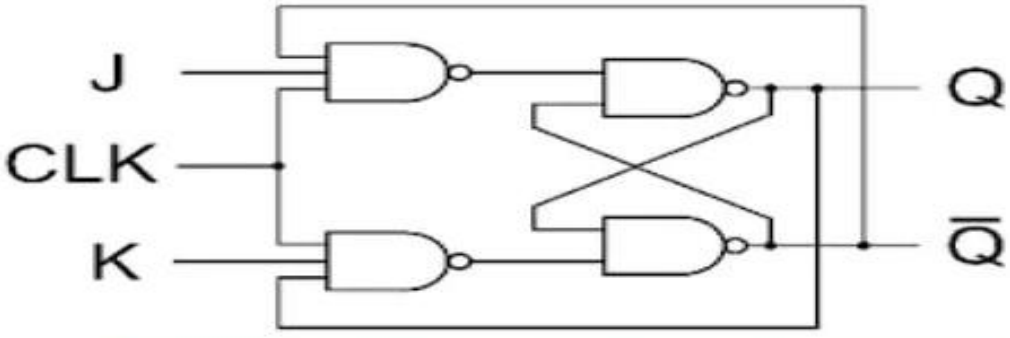


SR Flip-flop

- The SR (Set-Reset) flip-flop is one of the simplest sequential circuits and consists of two gates connected .
- The output of each gate is connected to one of the inputs of the other gate.
- The circuit has two active low inputs marked S' and R', as well as two outputs, Q and Q'.

Table					
	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Comments
1.	0	1	1	0	Q is set to 1 by 0 on $\overline{S}$
2.	1	1	1	0	No change, (1 on Q is remembered)
3.	1	0	0	1	Q is reset to 0 by 0 on $\overline{R}$
4.	1	1	0	1	No change, (0 on Q is remembered)
5.	0	0	1	1	Both inputs at 0 – both outputs are at 1 (Non-allowed state)
6.	1	1	?	?	Inputs change from 0,0 to 1,1 together - outputs will be INDETERMINATE

JK Flip Flop



- One of the most useful and versatile flip flop is the JK flip flop the unique features of a JK flip flop are:
  - If the J and K input are both at 1 and the clock pulse is applied, then the output will change state, regardless of its previous condition.
  - If both J and K inputs are at 0 and the clock pulse is applied there will be no change in the output.
  - There is no indeterminate condition, in the operation of JK flip flop i.e. it has no ambiguous state.

Same as for SR Latch	Clock	Input		Output		Description
	Clk	J	K	Q	Q'	
	X	0	0	1	0	Memory no change
	X	0	0	0	1	
	↓	0	1	1	0	Reset Q>>0
	X	0	1	0	1	
	↓	1	0	0	1	Set Q>>1
	X	1	0	1	0	
Toggle action	↓	1	1	0	1	Toggle
	↓	1	1	1	0	

- **When J = 0 and K = 0,**

  These J and K inputs disable the NAND gates, therefore clock pulse have no effect on the flip flop. In other words, Q returns it last value.

- **When J = 0 and K = 1,**

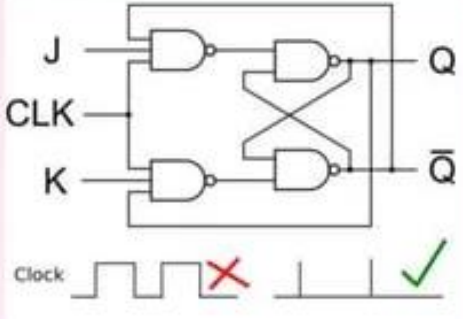
  The upper NAND gate is disabled the lower NAND gate is enabled if Q is 1 therefore, flip flop will be reset (Q = 0 , Q' =1)if not already in that state.

- **When J = 1 and K = 0**

  The lower NAND gate is disabled and the upper NAND gate is enabled if is at 1, As a result we will be able to set the flip flop (Q = 1, Q' = 0) if not already set.

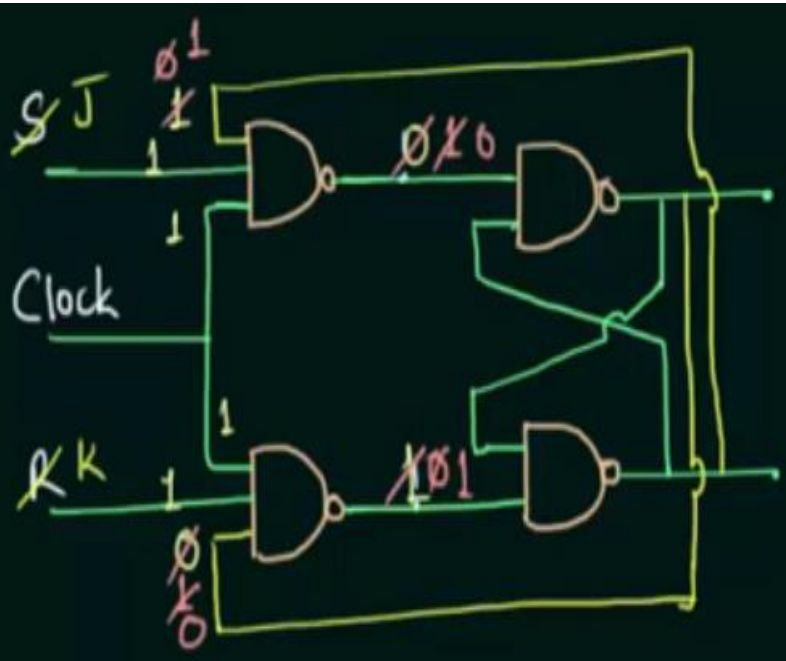
- **When J and K are both high**, the clock pulses cause the JK flip flop to toggle.

Circuit Diagram:



Clk	J	K	Q	Q'	State
1	0	0	Q	Q'	No change in state
1	0	1	0	1	Resets Q to 0
1	1	0	1	0	Sets Q to 1
1	1	1	-	-	Toggles

Racing explained



Clk	S	R	Q_{n+1}
0	x	x	Q_n (memory)
1	0	0	Q_n (memory)
1	0	1	0
1	1	0	1
1	1	1	Not used.

Clk = 0 Memory

Clk = 1 J = 1, K = 0, Q = 1, $\bar{Q}$ = 0

Clk = 1 J = 0, K = 1, Q = 0, $\bar{Q}$ = 1

Clk = 1 J = 1, K = 1

Assume Q = 0 & $\bar{Q}$ = 1

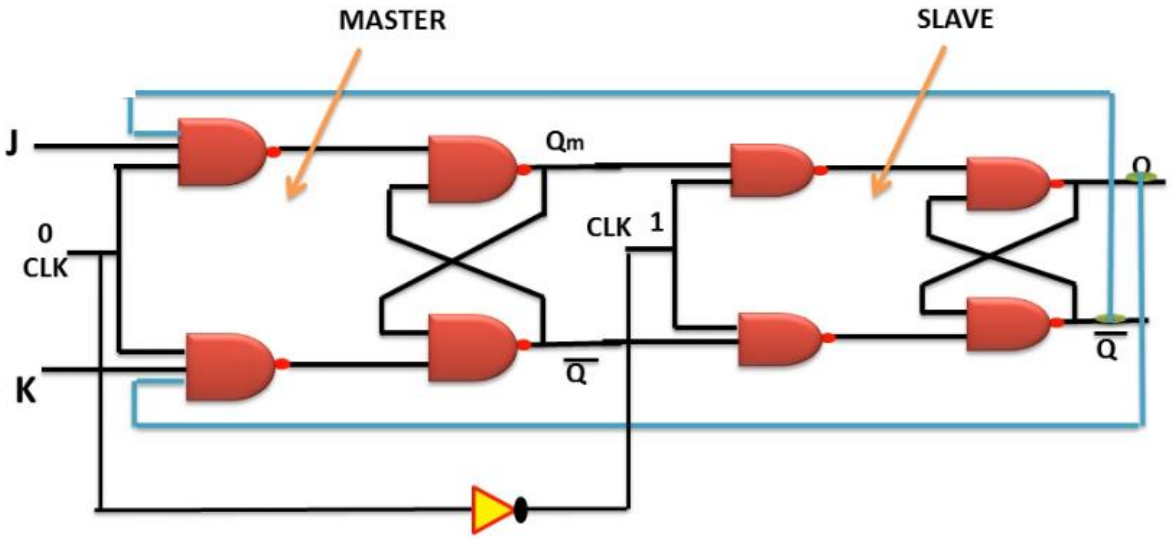
Q = 0, 1, 0, 1 ---

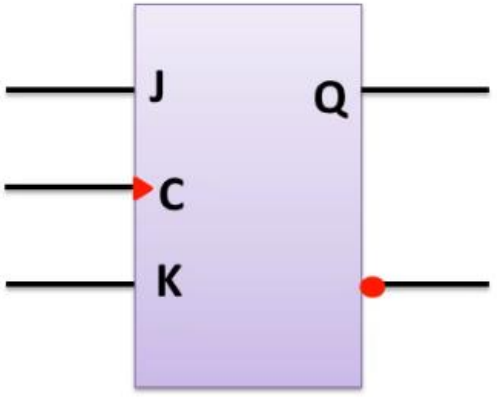
$\bar{Q}$ = 1, 0, 1, 0 ---

$Q_{n+1} = 0$

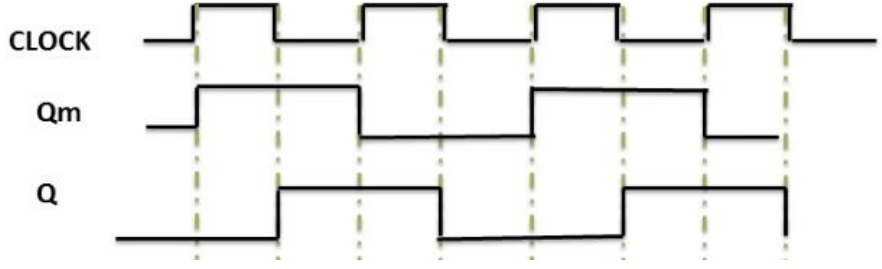
$Q_n \; \bar{Q}_n$ 1 0

JK OR MASTER SLAVE FLIP FLOP





Graphic symbol

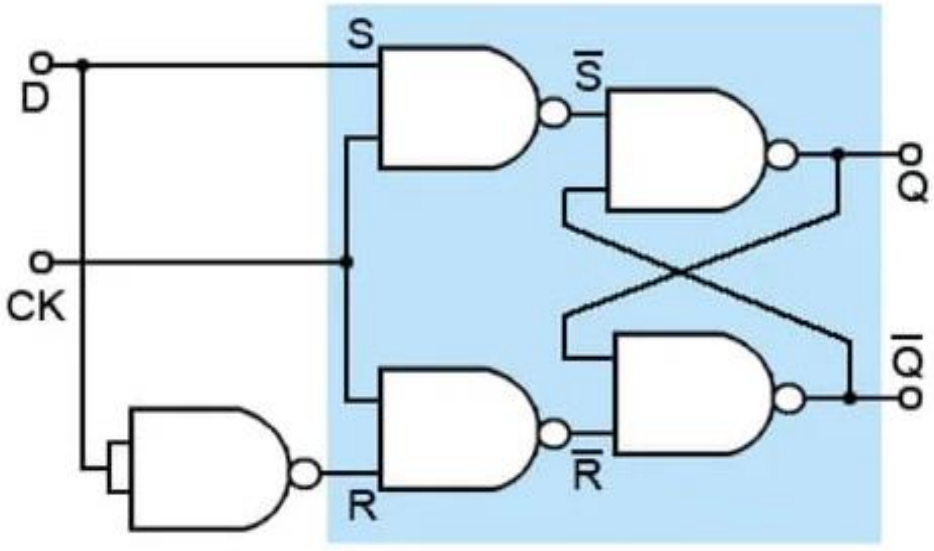


J	K	Q(t+1)
0	0	Q(t) no change
0	1	0 clear to 0
1	0	1 set to 1
1	1	Q'(t)

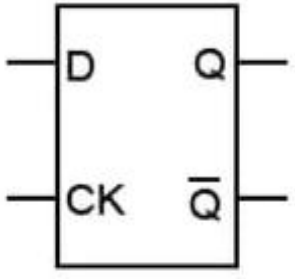
Master slave configuration is **mainly used to eliminate the race around the condition and get rid of unstable oscillation in the flip flop.**

D-Flip Flop

- The D type flip-flop has one data input 'D' and a clock input. The circuit edge triggers on the clock input. The flip-flop also has two outputs Q and Q' (where Q' is the reverse of Q).
- Such type of flip flop is a modification of clocked RS flip flop gates from a basic Latch flip flop and NOR gates modify it in to a clock RS flip flop.
- The D input goes directly to S input and its complement through NOT gate, is applied to the R input.

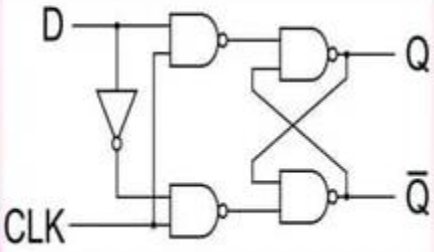


Inputs		Outputs	
CK	D	Q	$\bar{Q}$
0	X	No change	
1	0	0	1
1	1	1	0



- When the clock is low, both AND gates are disabled, therefore D can change values without affecting the value of Q.
- When the clock is high, both AND gates are enabled. In this case, Q is forced equal to D
- When the clock again goes low, Q retains or stores the last value of D

Logic Diagram:

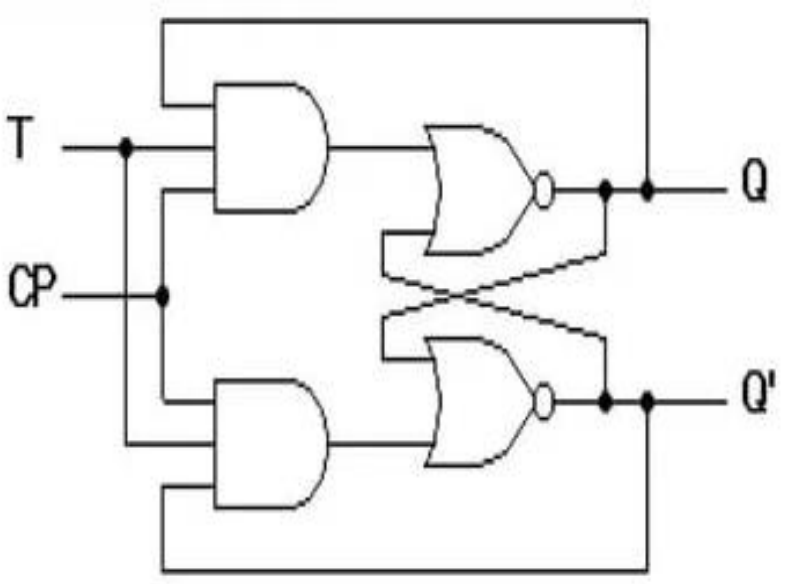


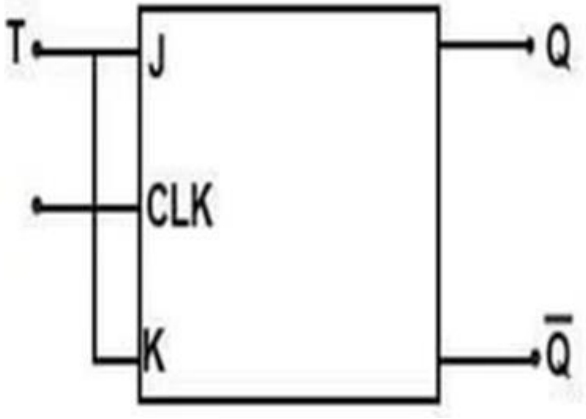
Truth Table:

Cp	D	Q	Q'
0	0	Memory state	
1	1	Set state	
1	0	Reset state	

Toggle Flip Flop or T-Flip Flop

- The T flip flop has only the Toggle and Hold **or NO Change** Operation.
- If Toggle mode operation. The output will toggle from 1 to 0 or vice versa.

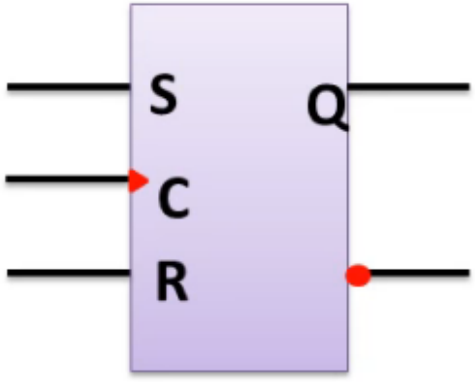




clk	T	Q	$\overline{Q}$	comments
↑	0	Q	$\overline{Q}$	No change
↑	1	$\overline{Q}$	Q	toggle

EXCITATION TABLE SR FLIP FLOP

Characteristics table of flip flop show the next state when input and present state are known.



S	R	Q(n+1)
0	0	Q(n) no change
0	1	0 clear to 0
1	0	1 set to 1
1	1	invalid

Characteristics table

Q(n)	S	R	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	x

Q(n) = pervious state
Q(n+1) = next state
Clock = 1

EXCITATION TABLE SR FLIP FLOP

Characteristics table

Q(n)	S	R	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	x

Excitation table

Q(n)	Q(n+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

S	R	Q(n+1)
0	0	Q(n) no change
0	1	0 clear to 0
1	0	1 set to 1
1	1	invalid

Q(n) = pervious state

Q(n+1) = next state

Clock = 1

EXCITATION TABLE OF JK OR MASTER SLAVE FLIP FLOP

Characteristics table of flip flop show the next state when input and present state are known.

C	J	K	Q(n+1)
0	0	0	Q(n)
1	0	0	Q(n)
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q(n)}$

Characteristics table

Q(n)	J	K	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Q(n) = pervious state

Q(n+1) = next state

Clock = 1

EXCITATION TABLE OF JK OR MASTER SLAVE FLIP FLOP

Characteristics table

Q(n)	J	K	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation table

Q(n)	Q(n+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

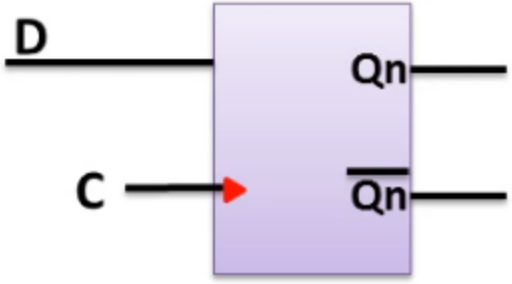
Q(n) = pervious state

Q(n+1) = next state

Clock = 1

EXCITATION TABLE OF D FLIP FLOP

Characteristics table of flip flop show the next state when input and present state are known.



C	D	Q(n+1)
0	X	Q(n)
1	0	0
1	1	1

Q(n) = pervious state
Q(n+1) = next state
Clock = 1

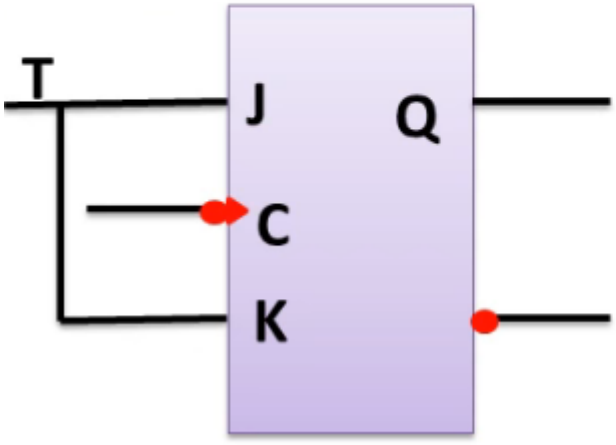
Characteristics table

Qn	D	Q(n+1)
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table

Qn	Q(n+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

EXCITATION TABLE OF T FLIP FLOP



C	T	Q(n+1)
0	X	Q(n)
1	0	Q(n)
1	1	$\overline{Q(n)}$

Characteristics table

Qn	T	Q(n+1)	
0	0	0	
0	1	1	ODD ONE DETECTOR
1	0	1	
1	1	0	

$Q(n+1) = Qn \oplus T$

Excitation table

Qn	Q(n+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

- **UNIT-II: Analysis of clocked sequential circuits, State Reduction and Assignment, Design Procedure.**

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Analysis of Clocked Sequential Circuit

Outputs & state of a clocked Sequential circuit is function of inputs and previous state.

Analysis of clocked sequential logic circuit consists of obtaining

- **state table:** Also called transition table consists of four sections; inputs, present state, next state and output.
- **state diagram:** Graphical representation of state table. A state is represented by a circle and transition between states is represented by a directed line. Input and output values of the transition is put on the directed line
- **state equation:** Specifies next state in terms of present state and inputs. Output equation is also described in terms of present state and inputs

Analysis Steps

State & output equation

- Assign variable names to input, outputs, flip flops input and output
- Determine FF's input equations in terms of input variable and present state.
- Determine next state equation of flip flop's using FF characteristic equation (state equation). Determine circuit's output if any, in terms of input variable/ present state (output equation).

State Table

- List all binary combination of inputs and present states
- List next state of FF's and outputs using state equation and output equation.

State Diagram

- Mark FF states and connect them with directed lines to next state with input / output values on the line. Each row of state table correspond to a directed line.

Analysis : Example -1

Example circuit has one input X, two D FF with outputs A & B and one output Y.

Input to FF's;

$D_A = AX + BX$

$D_B = \overline{A}X$

Using characteristic equation of D FF; $Q_{n+1} = D$

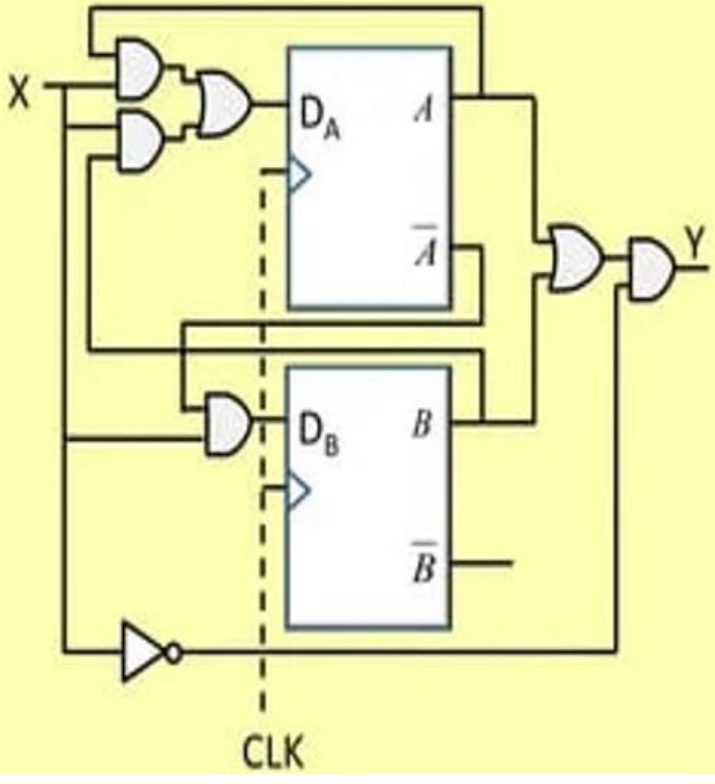
State equation;

$A_{n+1} = AX + BX$

$B_{n+1} = \overline{A}X$

Output equation;

$Y = (A+B)\overline{X}$



Analysis: example -1

State/ output equation;

$$A_{n+1} = (A+B)X$$

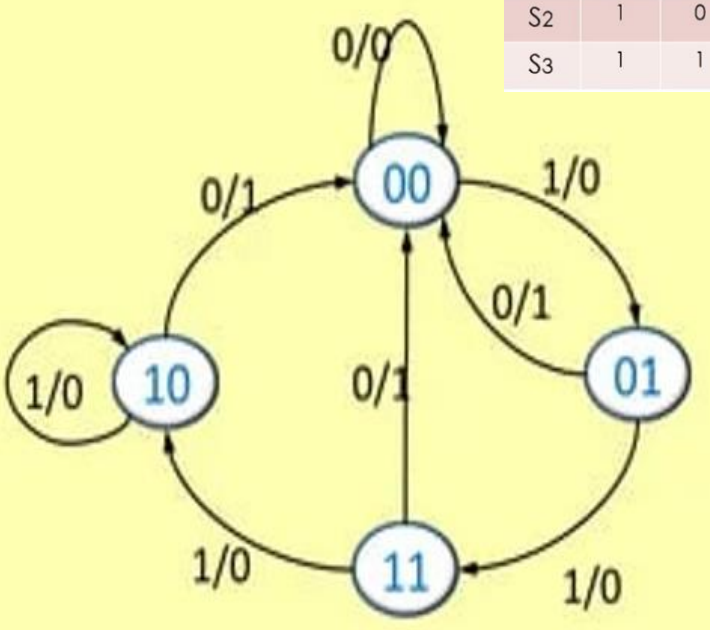
$$B_{n+1} = \overline{A}X$$

$$Y = (A+B)\overline{X}$$

State Table

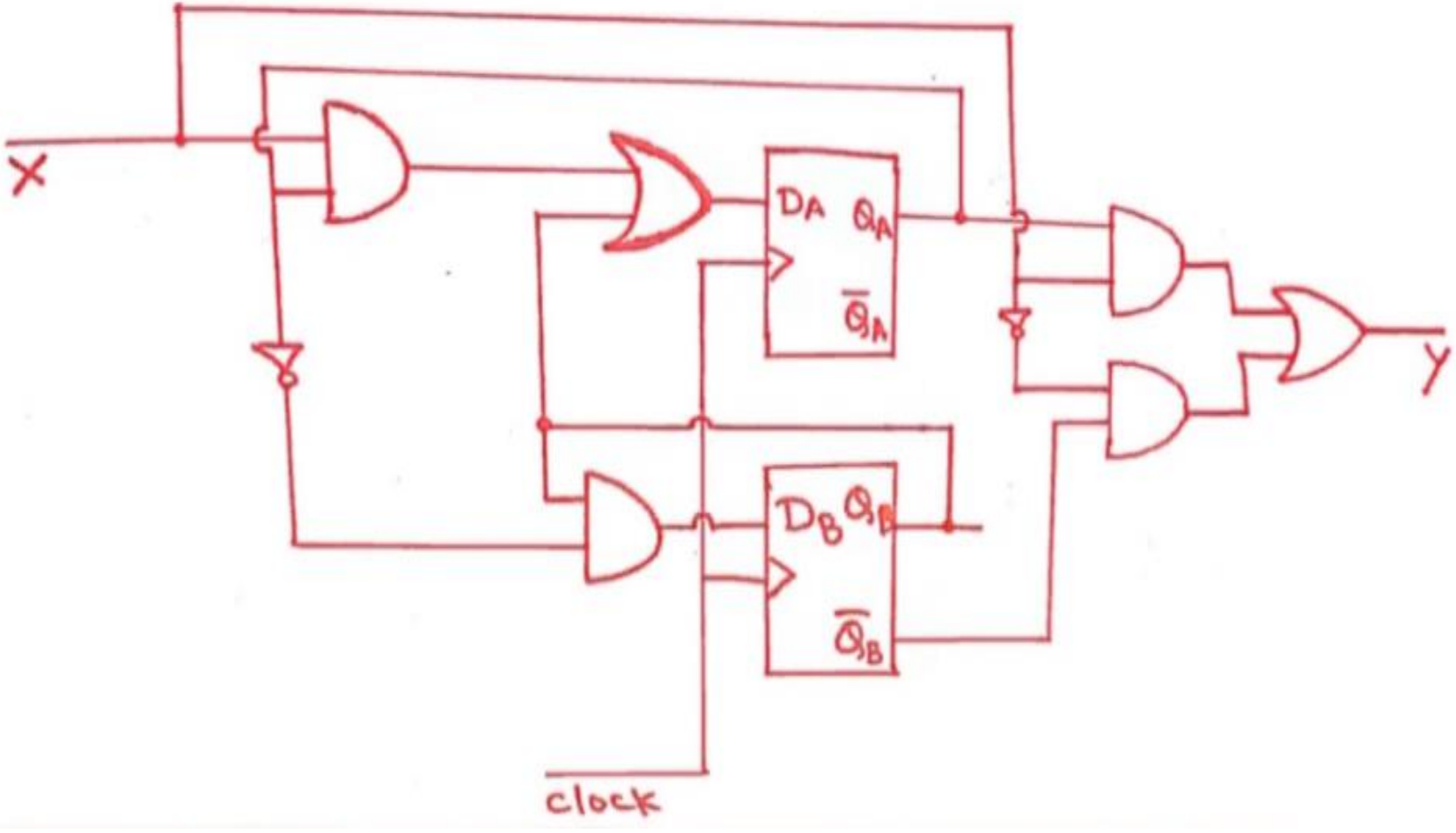
input	Present state		Next state		Output
X	A	B	A_{n+1}	B_{n+1}	Y
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	0

	A	B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

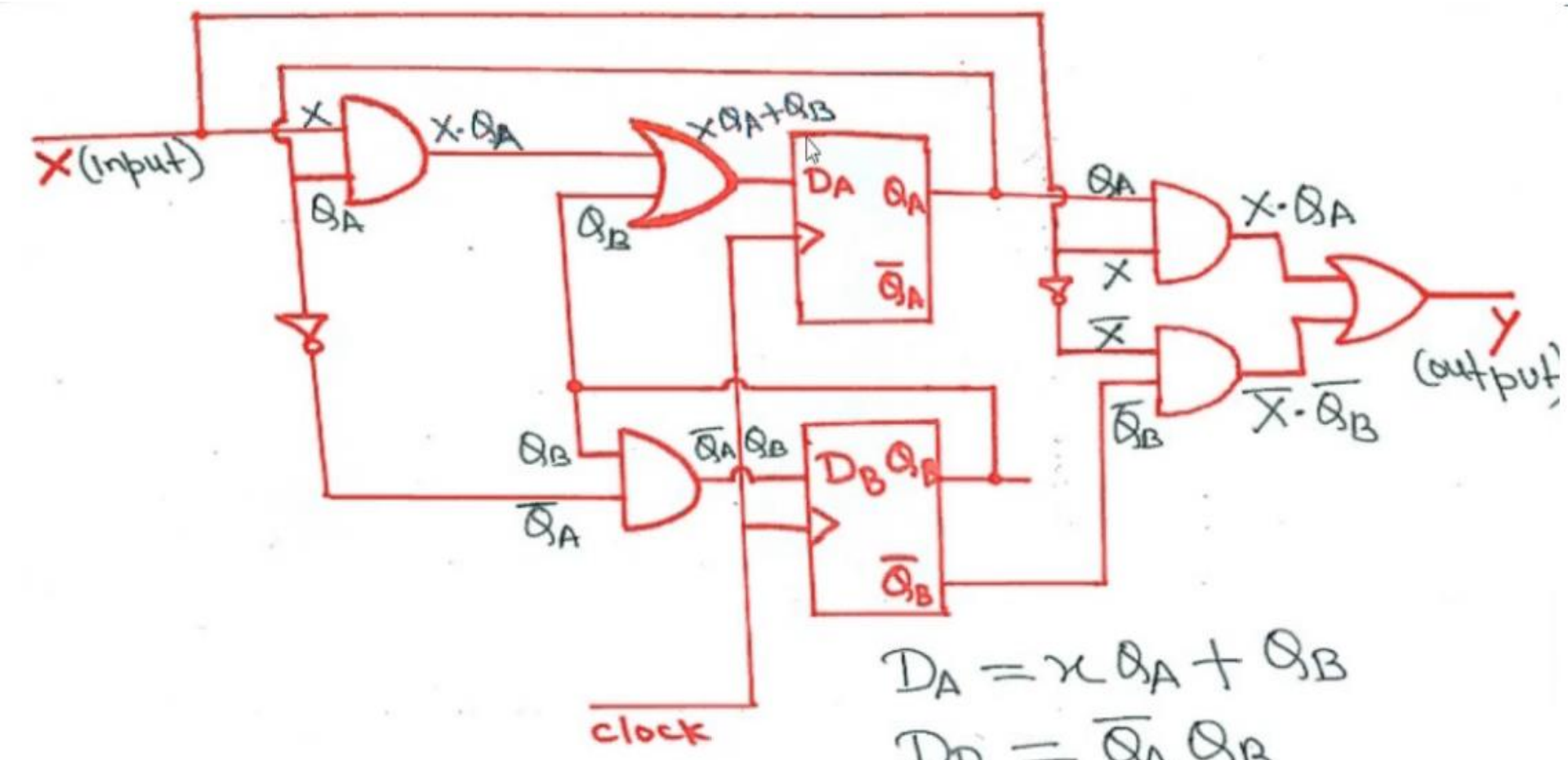


Example - 2

IN THE GIVEN CIRCUIT BELOW, TWO FLIP FLOP ARE USED, BOTH ARE D FLIP FLOP. HERE GATES ARE USED AS A COMBITIONAL LOGIC(INPUT COMBITIONAL LOGIC AND OUTPUT COMBITIONAL LOGIC).



Example - 2

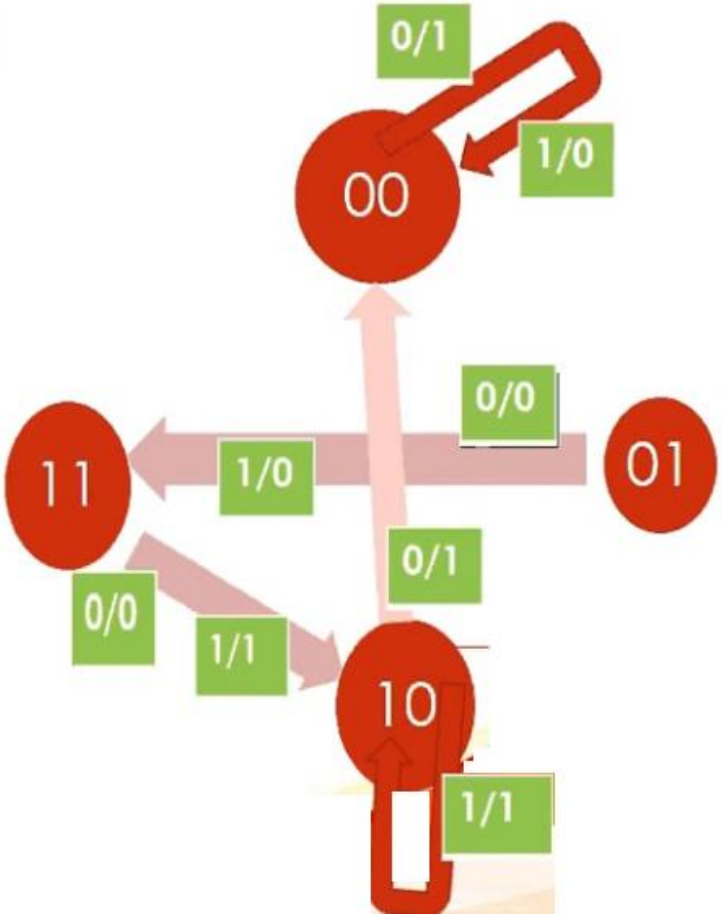


$$D_A = x Q_A + Q_B$$

$$D_B = \overline{Q}_A Q_B$$

$$Y = \overline{X}\,\overline{Q}_B + X \cdot Q_A$$

	QA	QB
S0	0	0
S1	0	1
S2	1	0
S3	1	1



Present state			Next state		
Q_A	Q_B	X	Q_A+	Q_B+	Y
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	0	1

$$Q_A+ = D_A = x Q_A + Q_B$$

$$Q_B+ = D_B = \overline{Q_A} Q_B$$

$$Y = \overline{x}\,\overline{Q_B} + x \cdot Q_A$$

Example – 3 (with different state table consideration)

State Equations

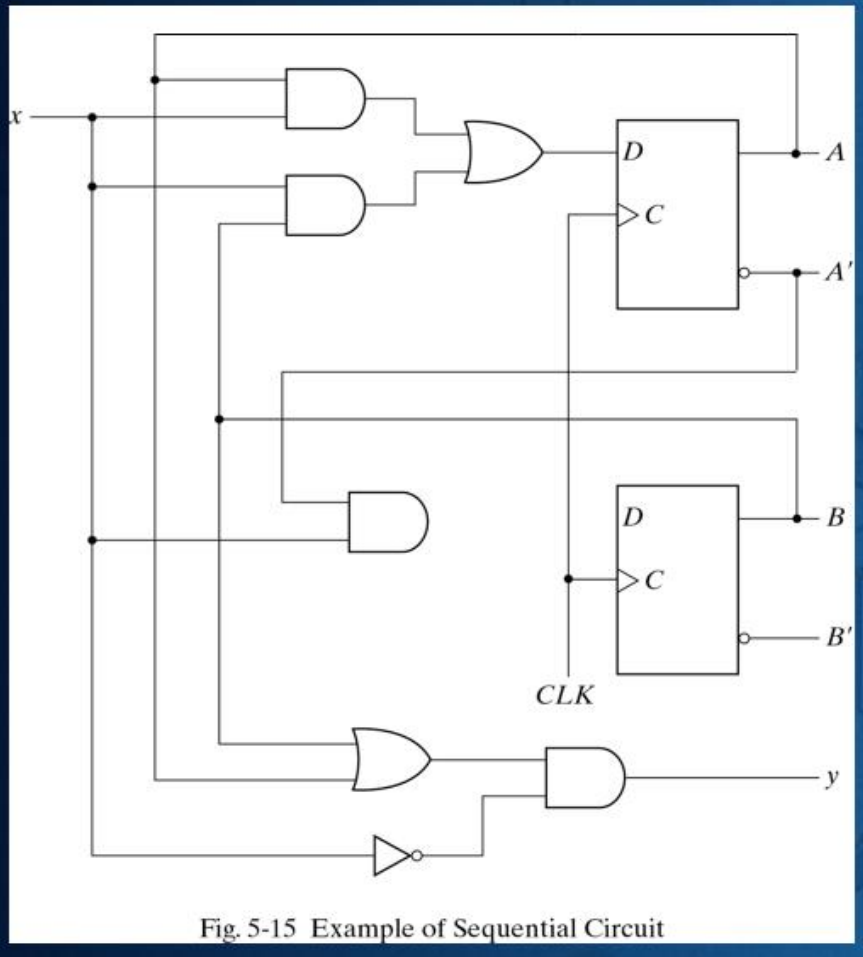


Fig. 5-15 Example of Sequential Circuit

A state equation shows the next state as a function of the current state and inputs

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

$$B(t+1) = A'(t)x(t)$$

$$y(t) = [A(t) + B(t)]x'(t)$$

$$A(t+1) = Ax + Bx$$

$$B(t+1) = A'x$$

$$y = (A + B)x'$$

Example – 3 (with different state table consideration)

State Table

Present State AB	Next State x=0 AB	Next State x=1 AB	Output x=0 y	Output x=1 y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

$$A(t+1) = Ax + Bx$$

$$B(t+1) = A'x$$

$$y = (A+B)x'$$

State Diagram

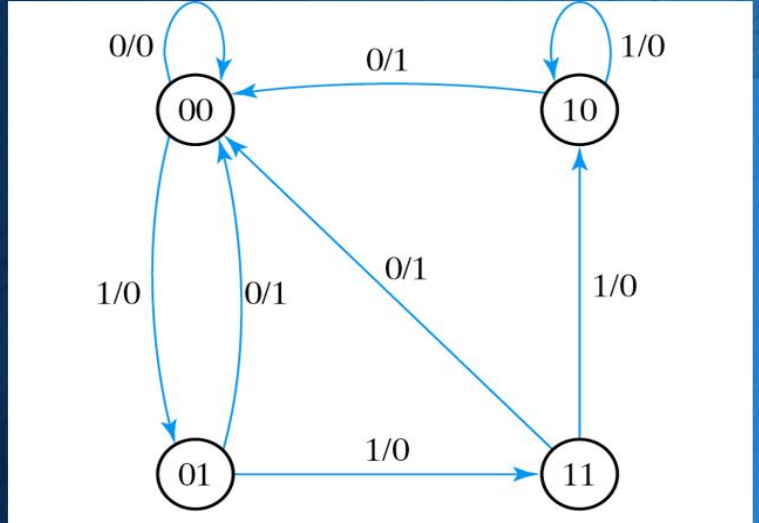
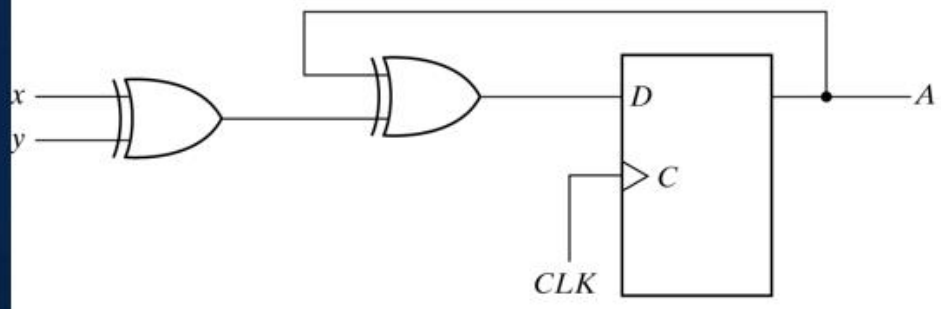


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

Example – 4

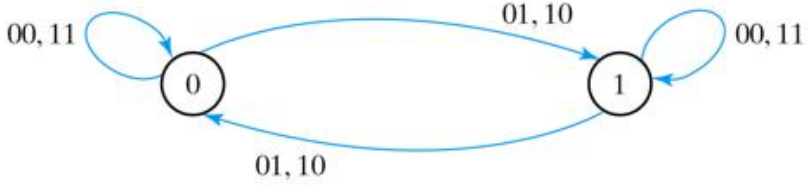
Analysis with D Flip-Flops



(a) Circuit diagram

Present state	Inputs		Next state
A	x	y	$A(t+1)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table



(c) State diagram

Fig. 5-17 Sequential Circuit with *D* Flip-Flop

$$D_A = A \oplus x \oplus y$$

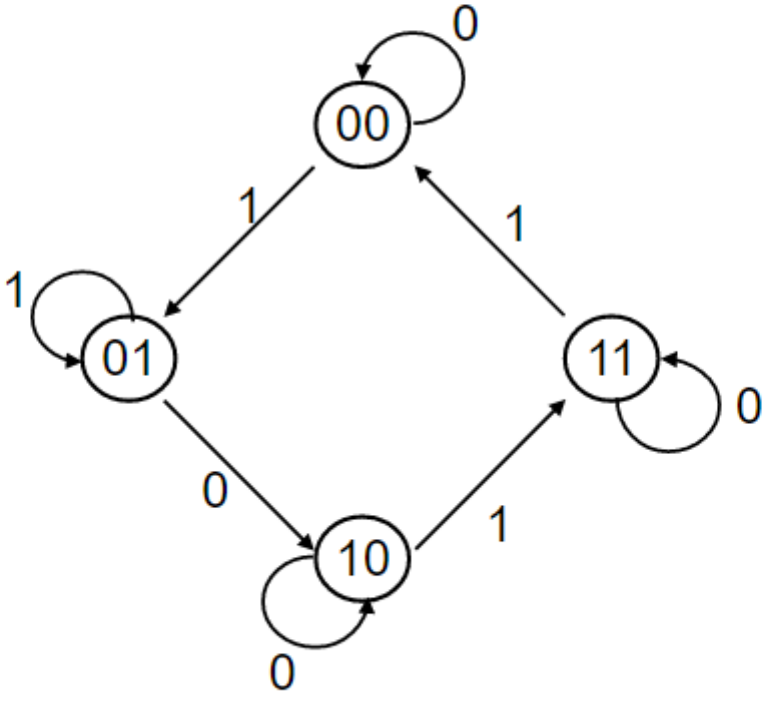
$$A(t+1) = A \oplus x \oplus y$$

Sequential Circuit Design

- Design procedure:
  - Start with circuit specifications – description of circuit behavior.
  - Derive the state table.
  - Perform state reduction if necessary.
  - Perform state assignment.
  - Determine number of flip-flops and label them.
  - Choose the type of flip-flop to be used.
  - Derive circuit excitation and output tables from the state table.
  - Derive circuit output functions and flip-flop input functions.
  - Draw the logic diagram.

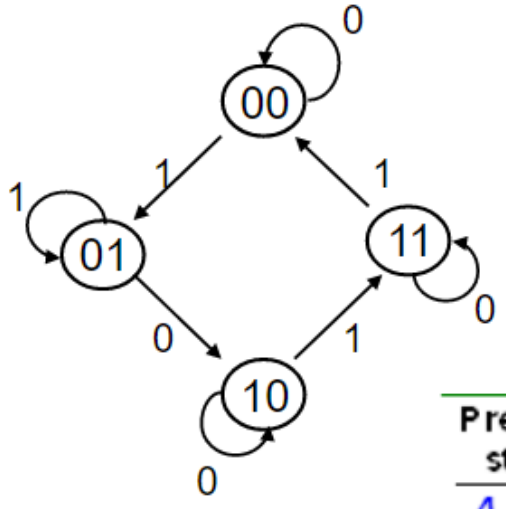
Design: Example #1

- Given the following state diagram, design the sequential circuit using JK flip-flops.



Design: Example #1

- Circuit state/excitation table, using JK flip-flops.



Present State	Next State x=0	Next State x=1
AB	A^+B^+	A^+B^+
00	00	01
01	10	01
10	10	11
11	11	00

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

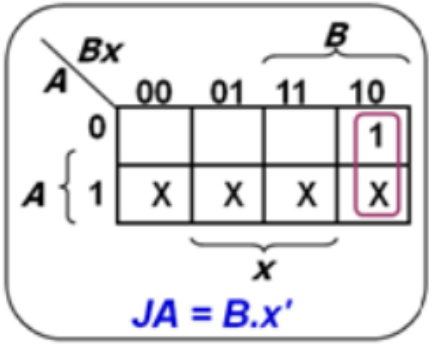
JK Flip-flop's excitation table.

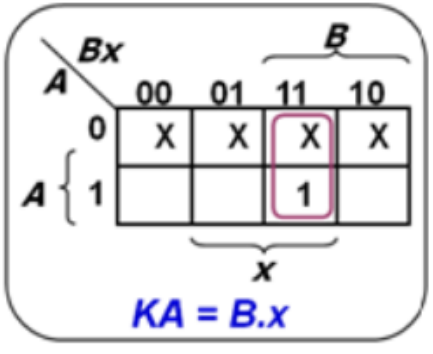
Present state A	Present state B	Input x	Next state A^+	Next state B^+	Flip-flop inputs JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

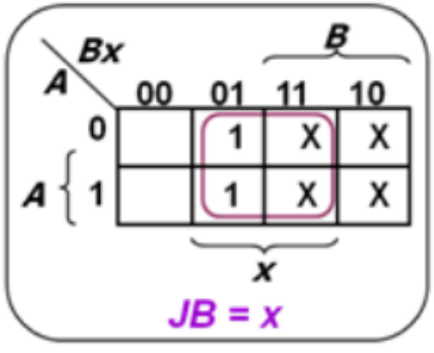
Design: Example #1

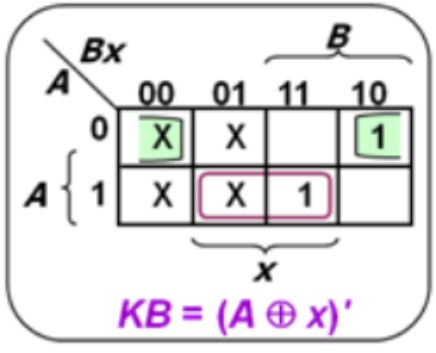
- From state table, get flip-flop input functions.

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1









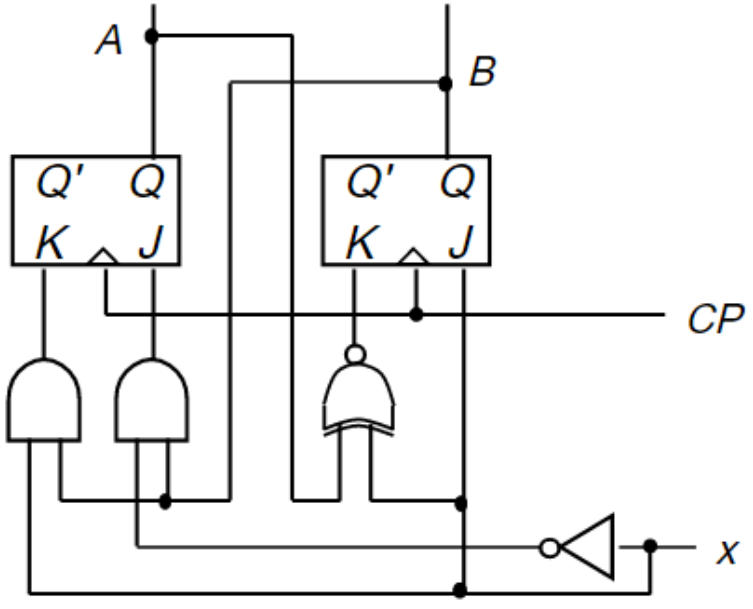
Design: Example #1

- Flip-flop input functions.

$JA = B.x'$ $\qquad JB = x$

$KA = B.x$ $\qquad KB = (A \oplus x)'$

- Logic diagram:



Example #2

- Design a circuit (with D flip-flops) that detects three or more consecutive 1's in a string of bits coming through an input line

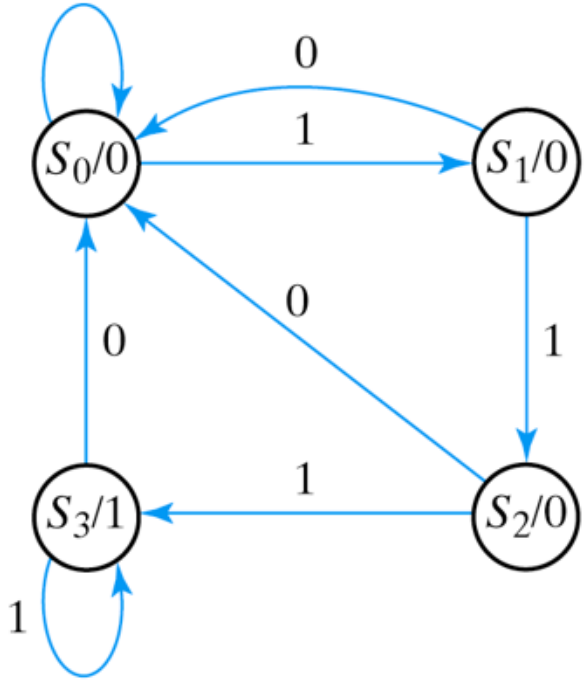
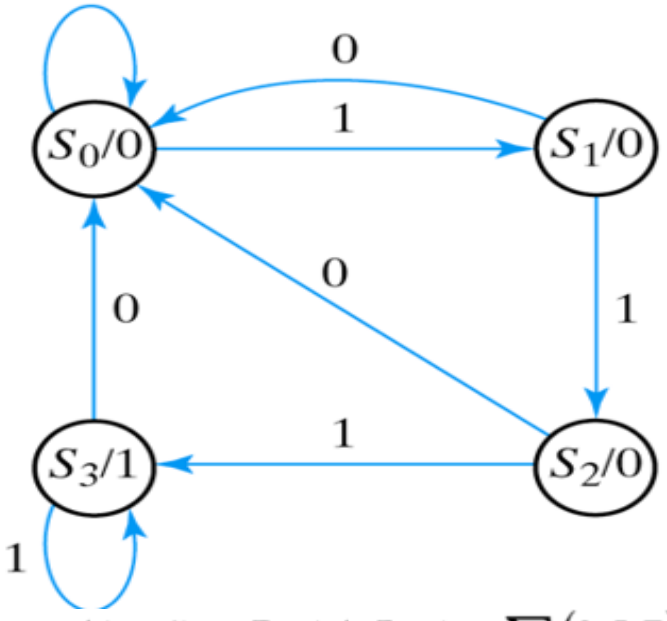


Fig. 5-24 State Diagram for Sequence Detector

Example (Cont.)

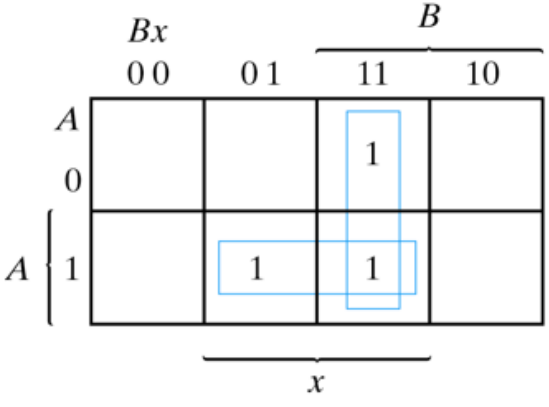
Present State A	Present State B	Input x	Next State A	Next State B	Output y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



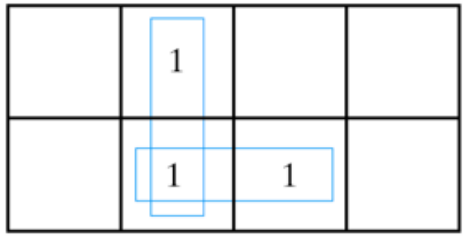
$$A(t+1) = D_A(A,B,x) = \sum(3,5,7)$$

$$B(t+1) = D_B(A,B,x) = \sum(1,5,7)$$

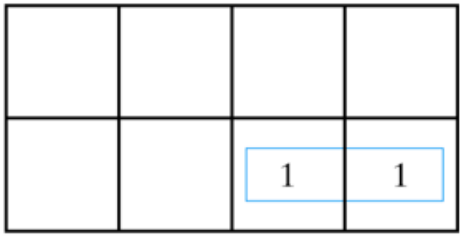
$$y(A,B,x) = \sum(6,7)$$



$D_A = Ax + Bx$



$D_B = Ax + B'x$



$y = AB$

Fig. 5-25 Maps for Sequence Detector

Example (Cont.)

$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$

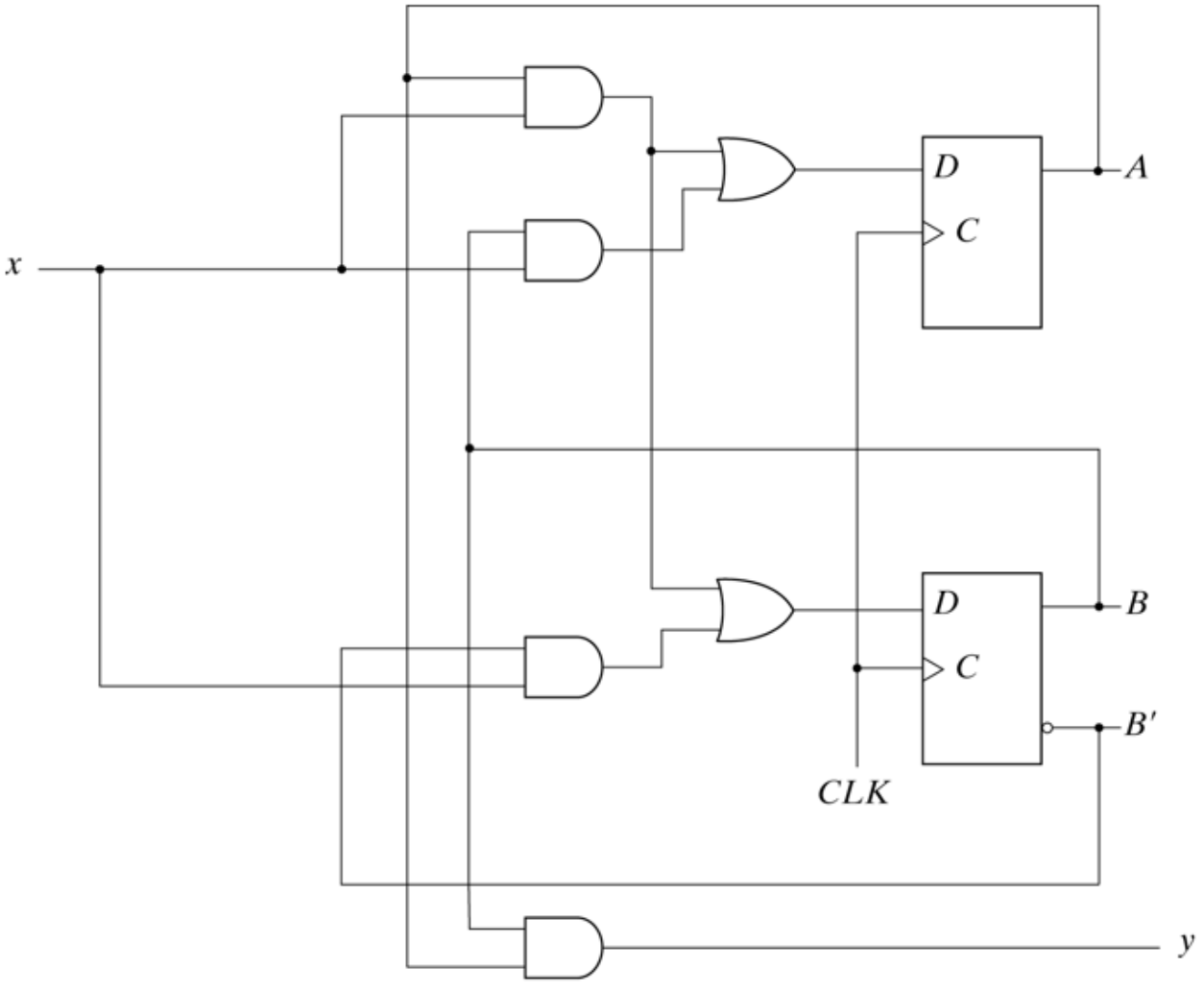


Fig. 5-26 Logic Diagram of Sequence Detector

State Reduction and State Assignment

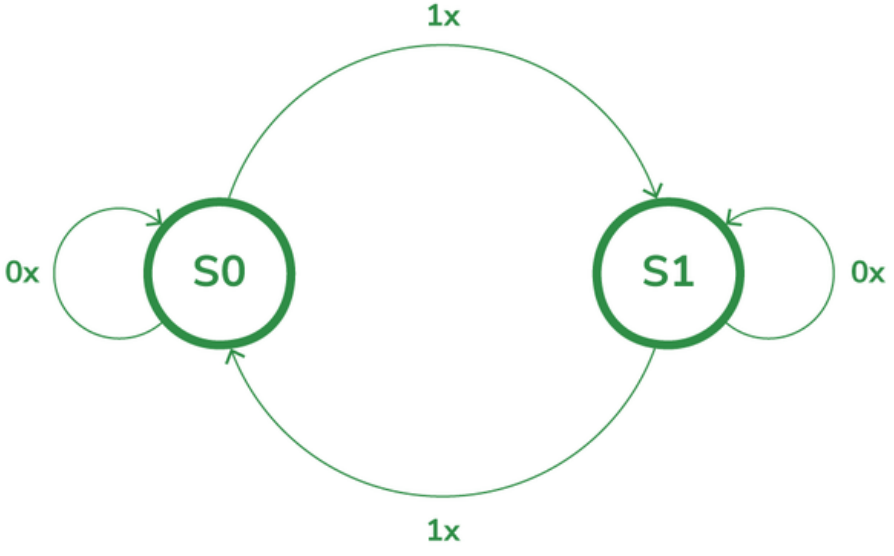
To illustrate the process of state reduction and state assignment first we have to know the concepts of the state diagram, state table, and state equation. In this article, we are going to learn all the topics related to state reduction and assignment.

State diagram: The state graph or state diagram is a pictorial representation of the relationships between the present state, the input state, the next state, and the output state of a sequential circuit i.e. A state diagram is a graphical representation of a sequential circuit's behavior.

Example: Consider an excitation table of J-K flip-flop

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

The state diagram of the above table is



State Diagram of J-K flip-flop

State table: Even though the behavior of a sequential circuit can be conveniently described using a state diagram, for its implementation the information contained in the state diagram is to be translated into a state table. The tabular form of the state diagram is the state table. The present state, the next state, and the output are the three sections of the diagram.

The state table of JK flip-flop is:

Inputs		Present state	Output
J	K	Q	Q+ (Output)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

State equation: $Q_{n+1} = Q_n \text{ bar } J + Q_n K \text{ bar}$

State equation: $Q_{n+1} = Q_n \text{ bar } J + Q_n K \text{ bar}$

State reduction:

The state reduction technique generally prevents the addition of duplicate states. The reduction in redundant states reduces the number of flip-flops and logic gates, reducing the cost of the final circuit. Two states are said to be equivalent if every possible set of inputs generates exactly the same output and the same next state. When two states are equal, one of them can be eliminated without changing the input-output relationship. The state reduction algorithm is applied in the state table to reduce equivalent states.

State assignment:

State assignment refers to the process of assigning binary values to the states of a sequential machine. The binary values should be given to the states in such a way that flip-flop input functions may be implemented with a minimum number of logic gates.

State assignment rules are as follows:

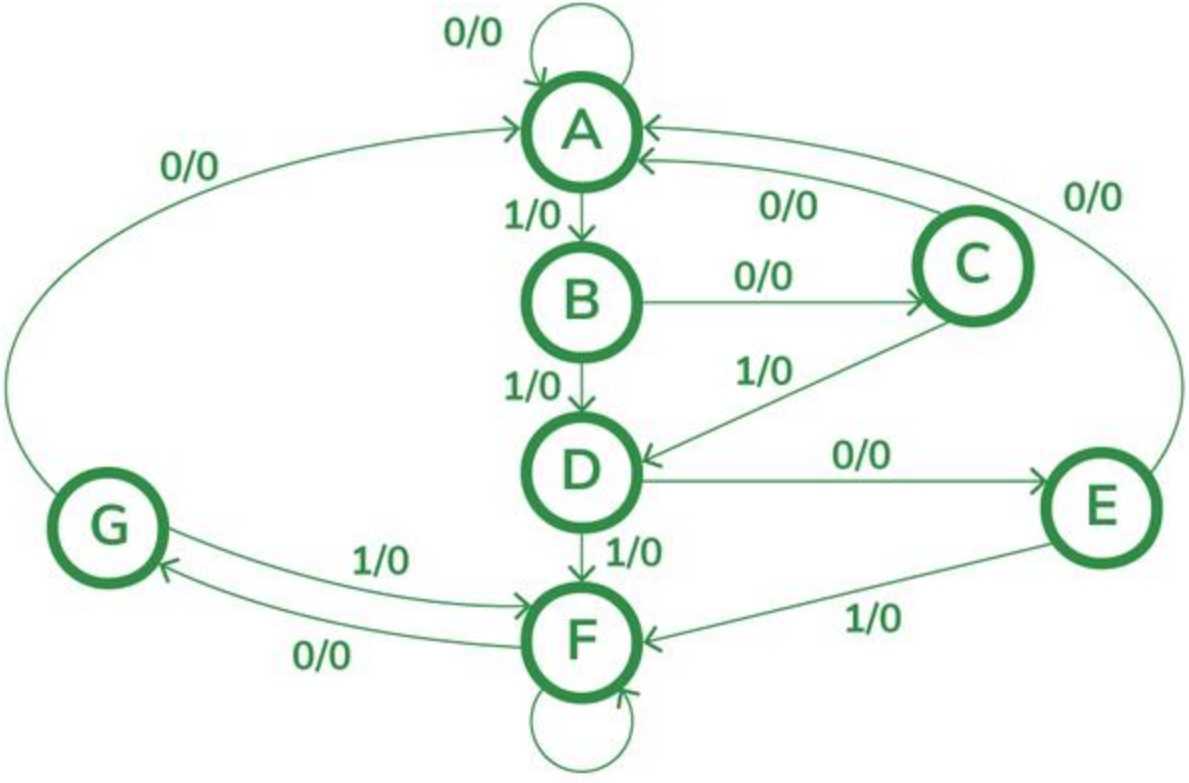
Rule 1: States having the same next state for a given input condition should have assignments that can be grouped into logically adjacent cells in a K-map.

Rule 2: States that are the next states of a single state should have assignments that can be grouped into logically adjacent cells in a K-map.

Example 1: To explain the concept of state reduction let us consider the state table as

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

The state diagram for the above state table is



State diagram before reduction

Step1: First here we are supposed to identify two or more similar states in the next state and output state. In the above table if we observe states of e and g are having the same next state and output values for all combinations of input i.e. X=0 and X=1.

So eliminate the g state in the state table and wherever g is present replace it with e. Because e and g both are the same i.e. e=g.

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e(g=e)	f	0	1

Step 2: Again check if any two states have similar values or not. If any two states have the same next state and output then eliminate one state.

Here d and f are having the same next state value and output. So eliminate f and wherever f is present replace it with d. Because both are the same d=f

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d(d=f)	0	1
e	a	d(d=f)	0	1

Step 3: Further observe if any similar states are present or not. The states c and e are having same next states but they are having different outputs. So we can not consider it a reduction state.

Step 4: If you observed the state table, the states are represented by using the alphabet. We can not proceed further if we are having alphabets, so, assigning binary numbers to alphabets is called a state assignment.

To assign binary numbers to the state we have to consider the minimum number of bits.

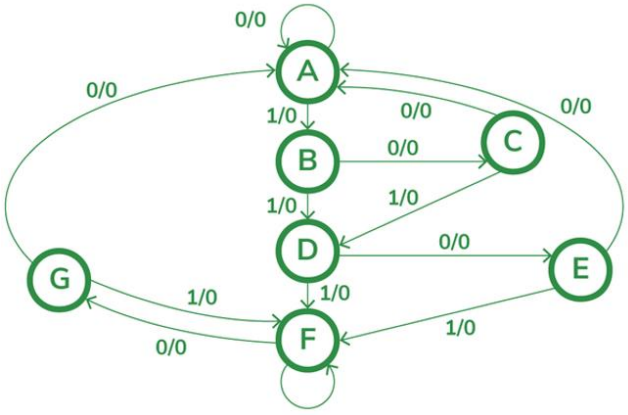
The codes must contain n bits for a circuit with m states, where $2^n >= m$. In this case, each state requires $2^3 >=5=>3$ bits to be represented. With three bits, there are eight possible combinations, of which five can be used to represent the states.

State	Assignment 1
	Binary
a	000
b	001
c	010
d	011
e	100

State	Assignment 1
	Binary
a	000
b	001
c	010
d	011
e	100

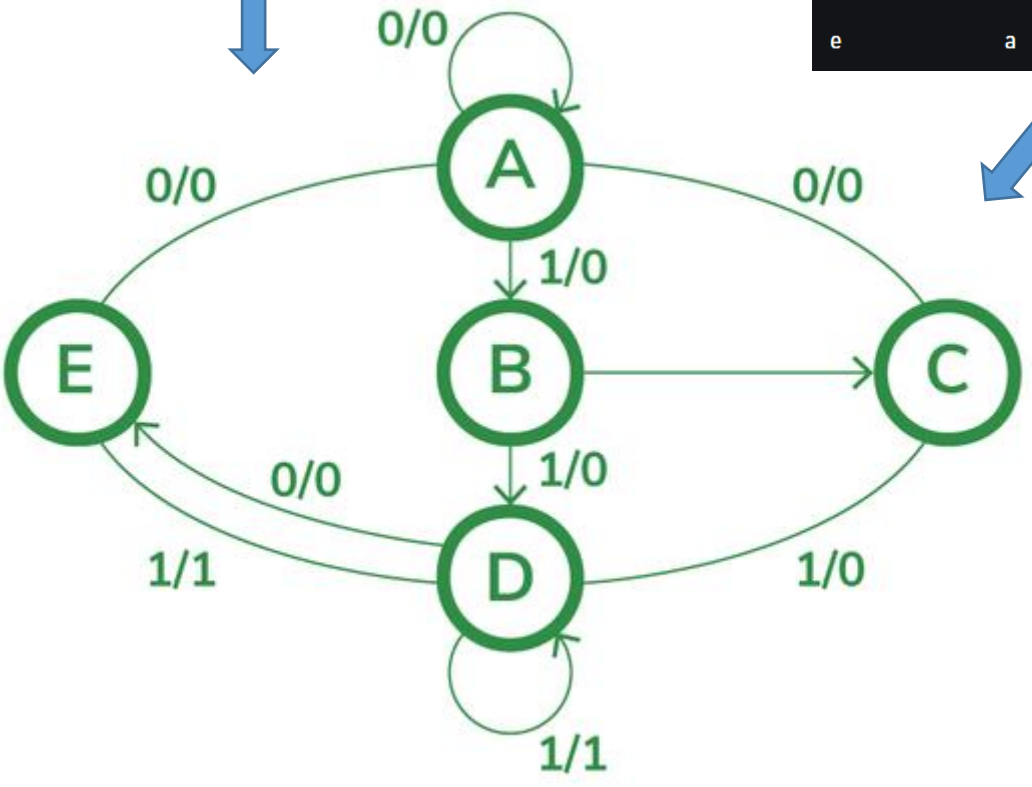
Step 5: Replacing the alphabets with binary numbers.

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1



State diagram before reduction

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d(d=f)	0	1
e	a	d(d=f)	0	1



State diagram After reduction

State Reduction Example

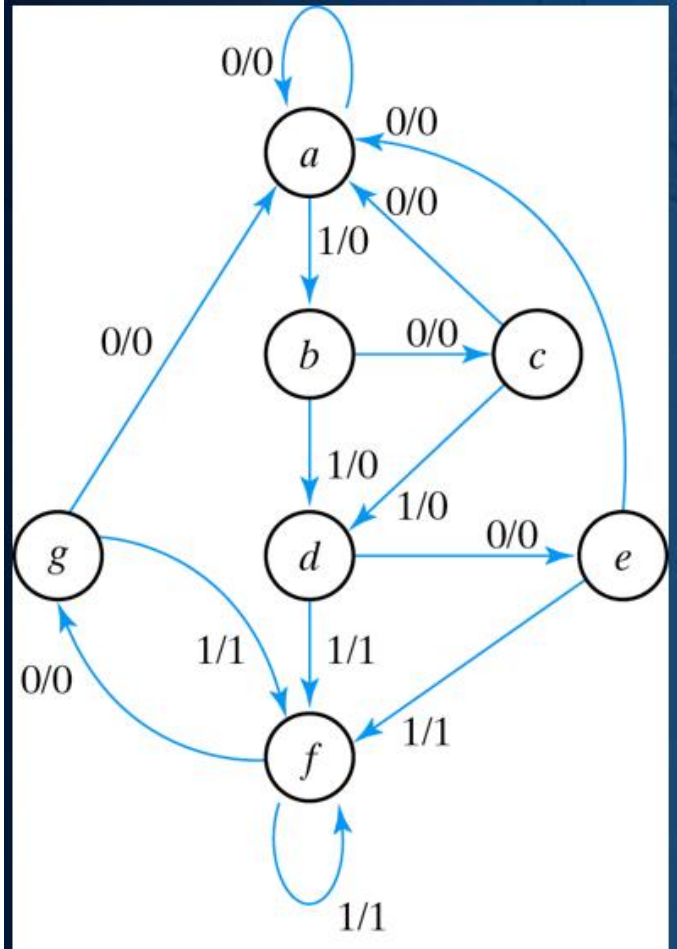


Fig. 5-22 State Diagram

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Find the states for which the next states and outputs are the same

Example (Cont.)

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
~~d~~	~~e~~	~~f~~	~~0~~	~~1~~
e	a	f	0	1
~~f~~	~~c~~	~~f~~	~~0~~	~~1~~

In the next state, g is replaced with e

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

In the next state, f is replaced with d

Example (Cont.)

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

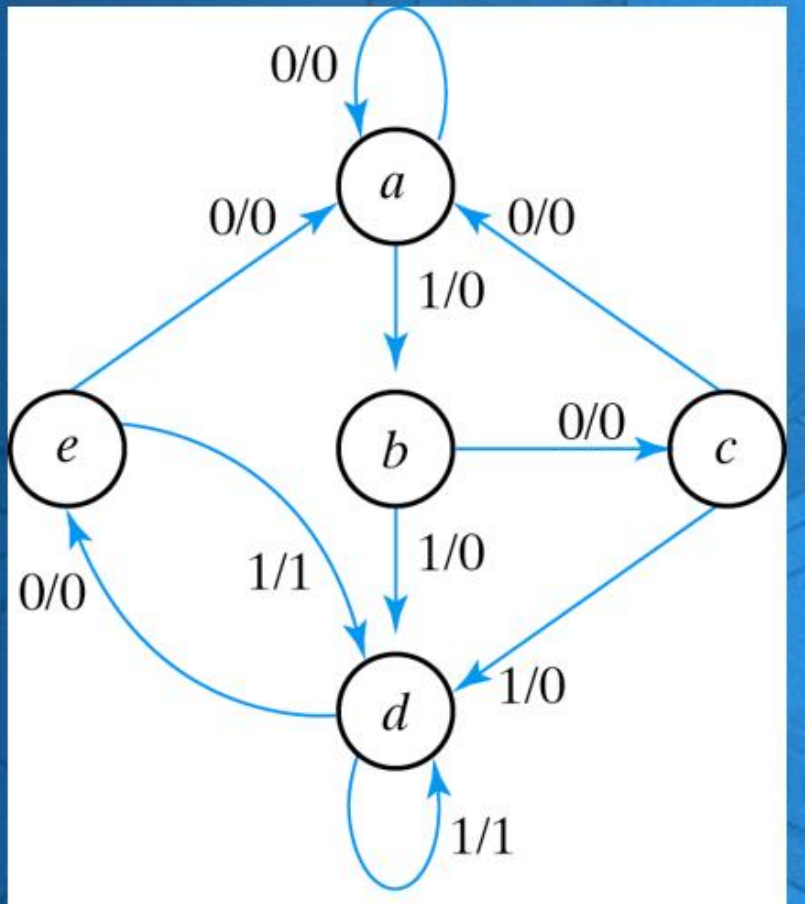


Fig. 5-23 Reduced State Diagram

State Assignments

- You need to assign binary values for each state so that they can be implemented
- You need to use enough number of bits to cover all the states

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Department of Electronics and Communication Engineering

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

EMPOWERING THROUGH INNOVATIONS

MLRS

DIGITAL LOGIC DESIGN AND COMPUTER ORGANIZATION

- **UNIT-II: Registers, Shift Registers, Ripple counters, Synchronous counters**

Presentation By

Dr. Bapi Debnath

Assoc. Prof.

Department of ECE

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AN AUTONOMOUS INSTITUTION)

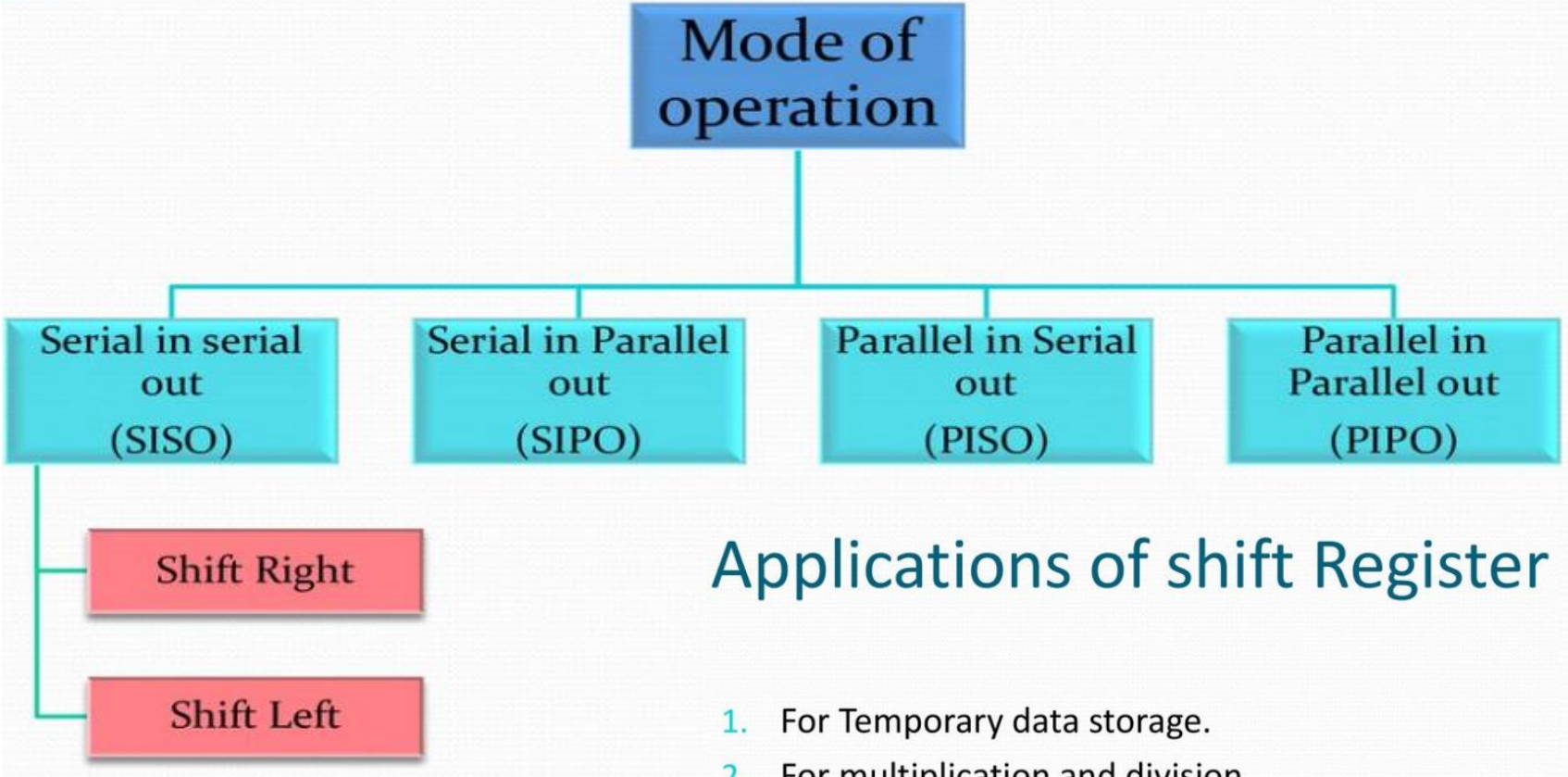
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

INTRODUCTION: REGISTERS

- Register is an important application of Flip-Flop.
- Flip- Flop is a 1 bit memory cell which can be used for storing the digital data.
- To increase the ***storage capacity in terms of number of bits,*** we have to use a group of flip-flop. ***Such a groups of flip-flop is known as a "Register".***
- Thus register is a group of flip-flops. ***The "n-bit" register will consist of "n" number of flip flops.***

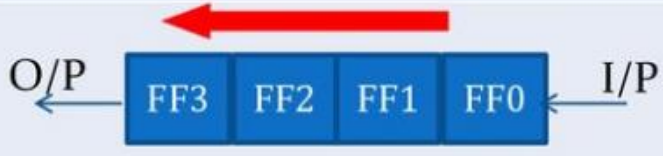
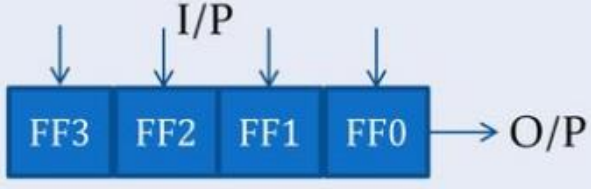
Classification of Registers :



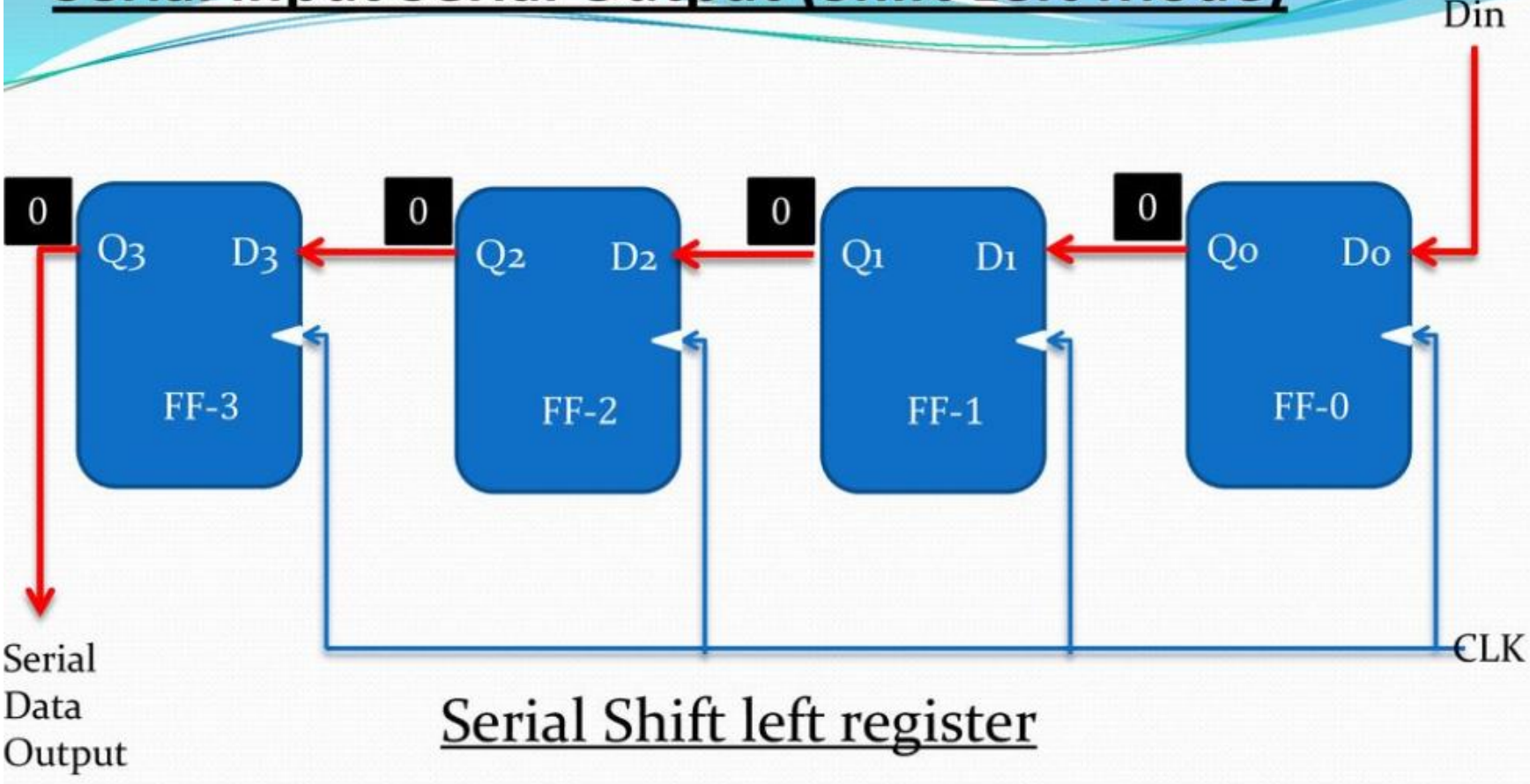
Applications of shift Register

1. For Temporary data storage.
2. For multiplication and division
3. As a delay line
4. Ring Counter
5. Parallel to serial converter

Shift Register

Sr. No.	Mode	Illustrative Diagram	Comments
1.	SISO (Shift Right)		Data bits shift from Left to Right by 1 position per clock cycle.
2.	SISO (Shift Left)		Data bits shift from Right to Left by 1 position per clock cycle.
3.	SIPO		All o/p bits are made aval. simult. after **4-clcok pulse**
4.	PISO		All i/p bits are applied simult and. After **4-clk pulse** the required o/p is available serially.

Serial input Serial Output (Shift Left Mode)



Serial Shift left register

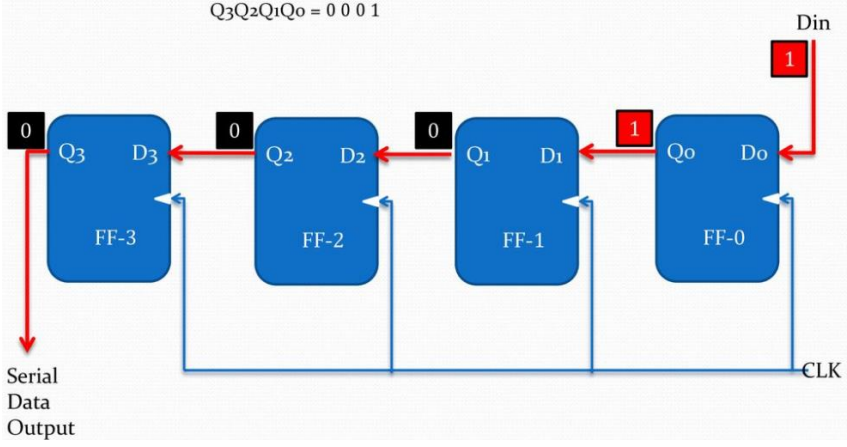
- Before application of clock let assume all outputs are zero and apply MSB bit of the number to entered to Din. So Din = Do = 1.

SISO ----- 1111

1

- Apply the clock . On the first falling edge of clock, the FF-0 is SET and the stored data in the register is

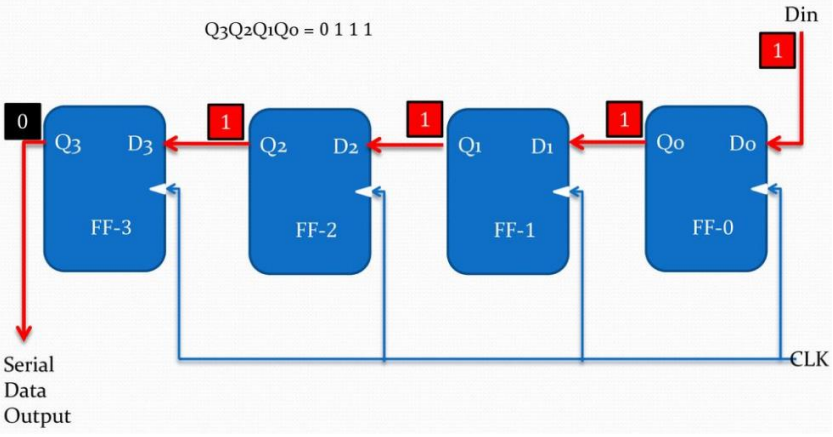
$Q_3Q_2Q_1Q_0 = 0\ 0\ 0\ 1$



3

- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 2will SET and the stored data changes to,

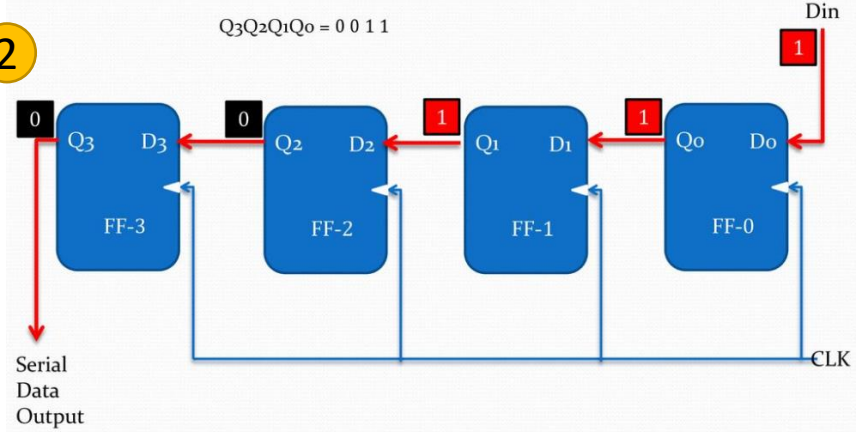
$Q_3Q_2Q_1Q_0 = 0\ 1\ 1\ 1$



- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 1 will SET and the stored data changes to,

2

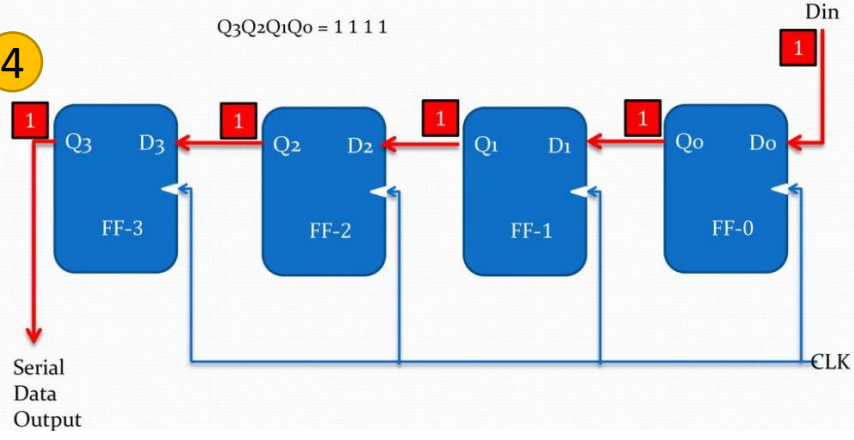
$Q_3Q_2Q_1Q_0 = 0\ 0\ 1\ 1$



- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 3will SET and the stored data changes to,

4

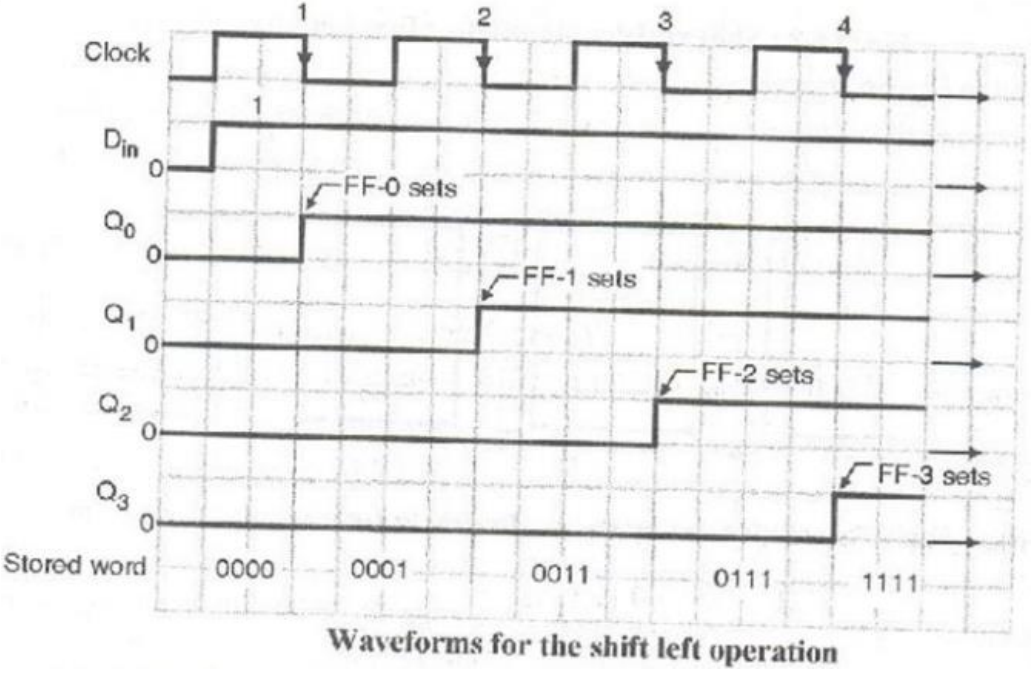
$Q_3Q_2Q_1Q_0 = 1\ 1\ 1\ 1$



Summary of shift left operation

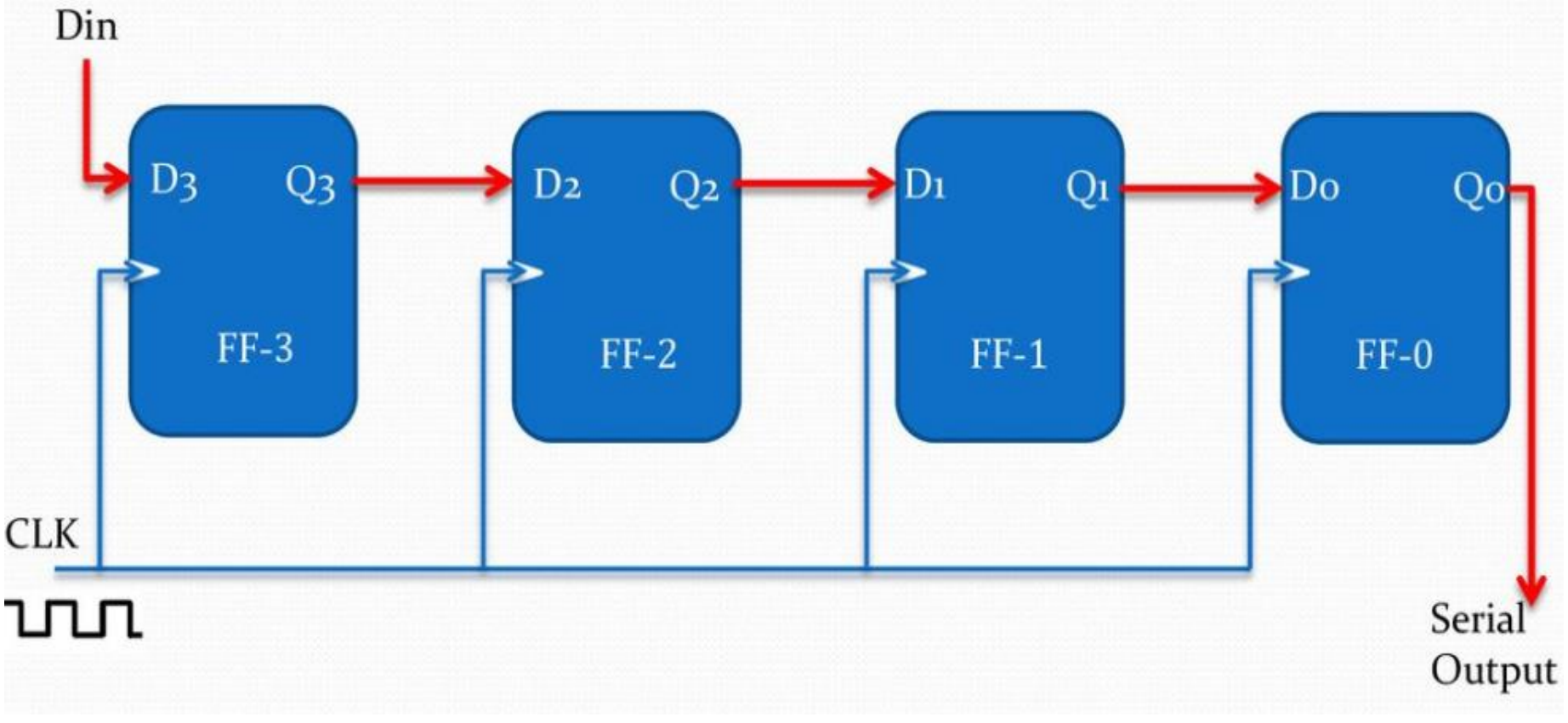
	CLK	Q3	Q2	Q1	Q0	SERIAL INPUT Din = Do
INITIALLY		0	0	0	0	
1st	↓	0	0	0	1	1
2nd	↓	0	0	1	1	1
3rd	↓	0	1	1	1	1
4th	↓	1	1	1	1	1

Direction of data travel ←



Waveforms for the shift left operation

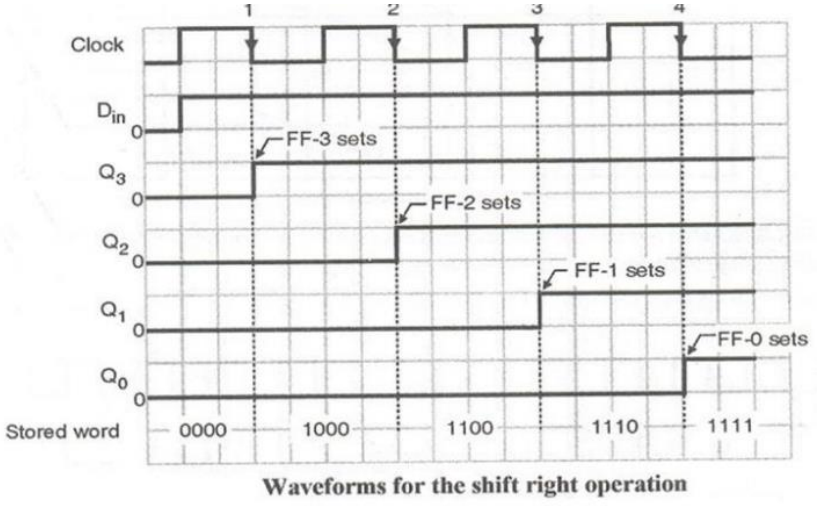
Serial input Serial Output (Shift Right Mode)



shift right operation

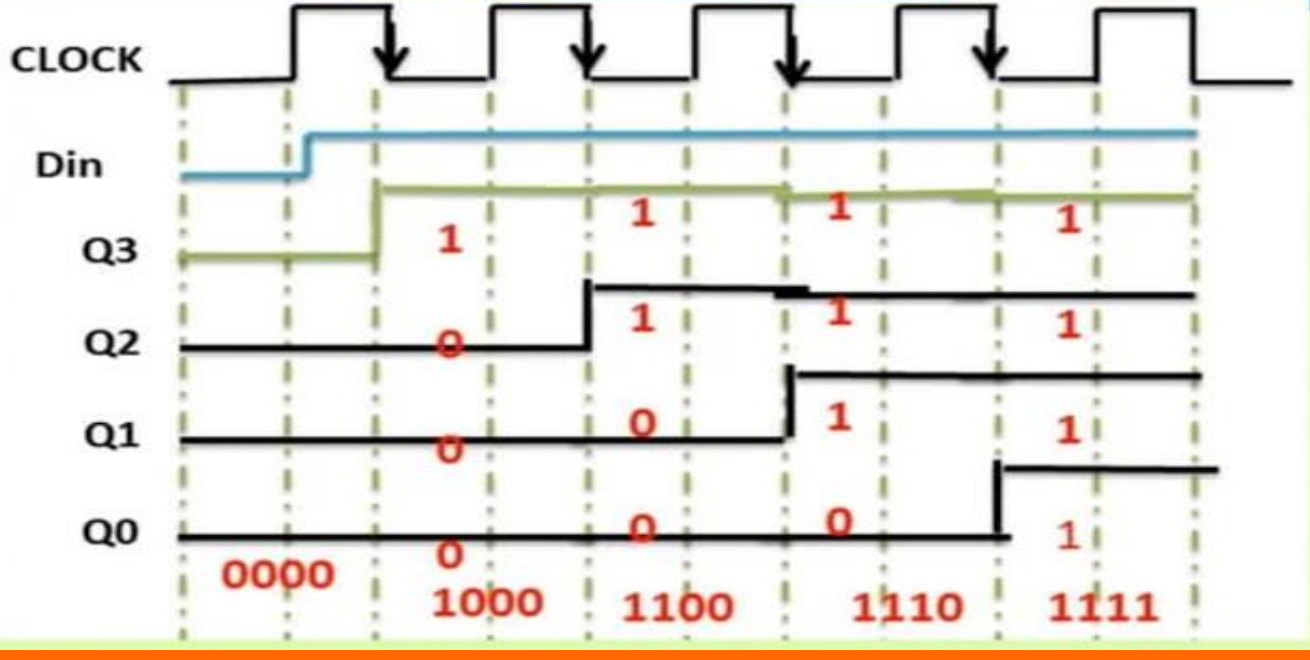
	CLK	$D_{in} = Q_3$	$Q_3 = D_2$	$Q_2 = D_1$	$Q_1 = D_0$	Q_0
Initially			0	0	0	0
1st	↓	1	1	0	0	0
2nd	↓	1	1	1	0	0
3rd	↓	1	1	1	1	0
4th	↓	1	1	1	1	1

Direction of data travel



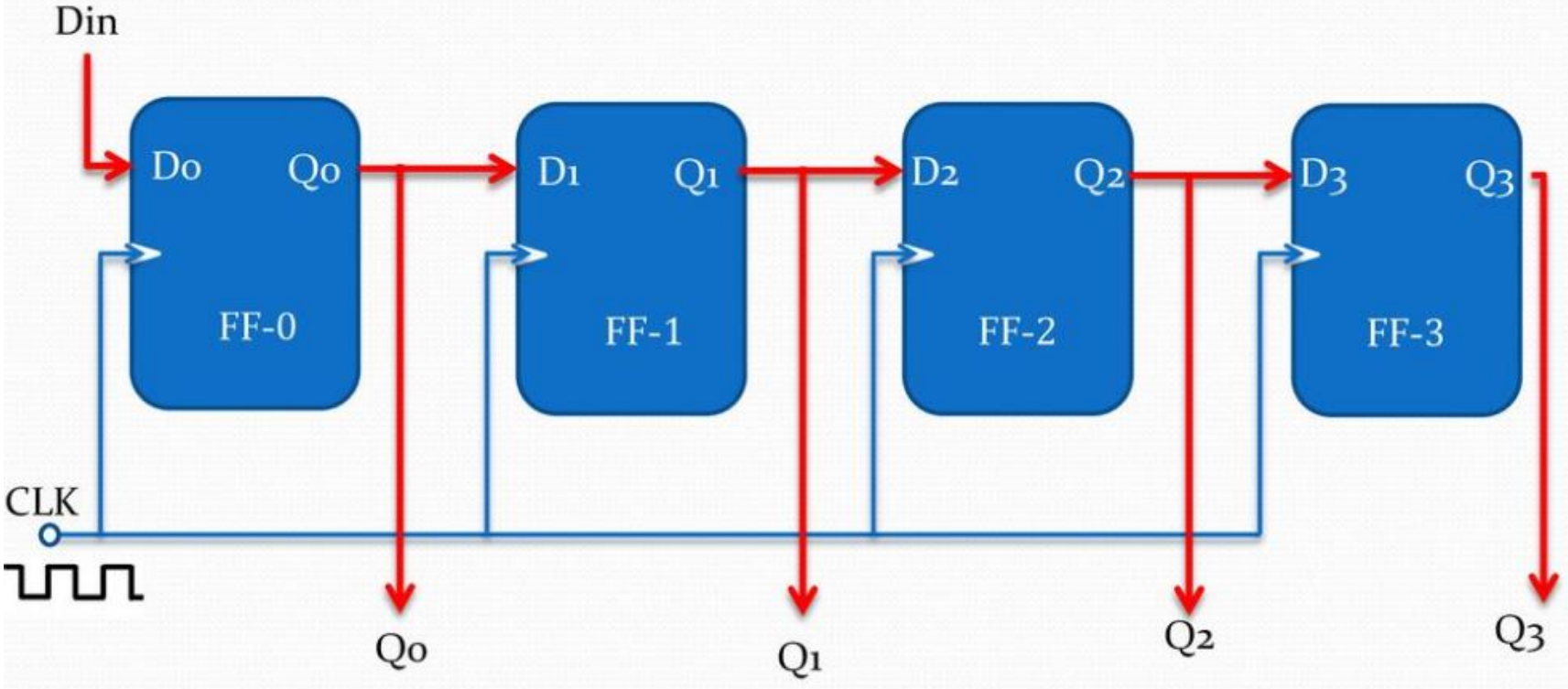
Waveforms for the shift right operation

Waveform (SISO)



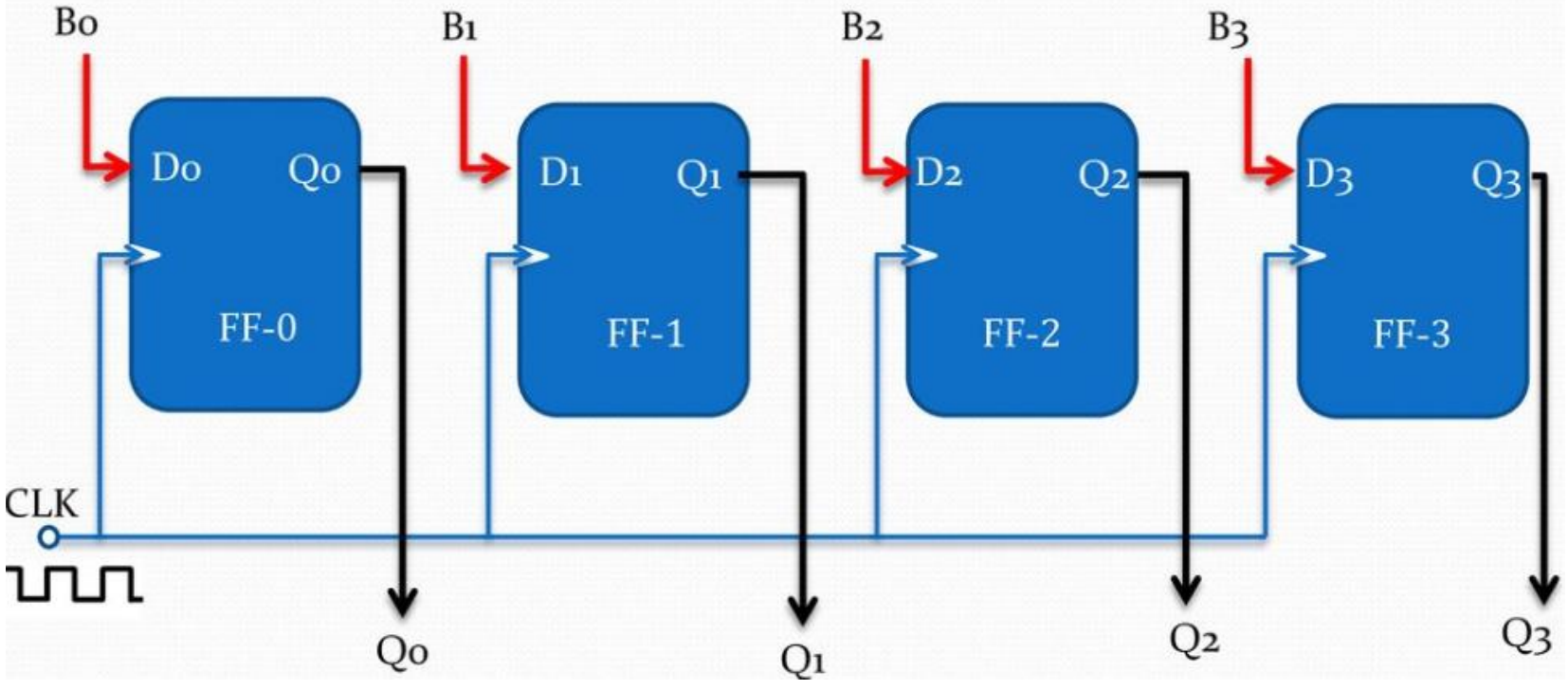
Serial input Parallel Output (SIPO)

- In this operation the data is entered serially and taken out in parallel.
- That means first the data is loaded bit by bit. The output are disabled as the loading is taking place.
- Number of clock cycles required to load a four bits data is 4. Hence the speed of operation of SIPO mode is same as that of SISO mode.



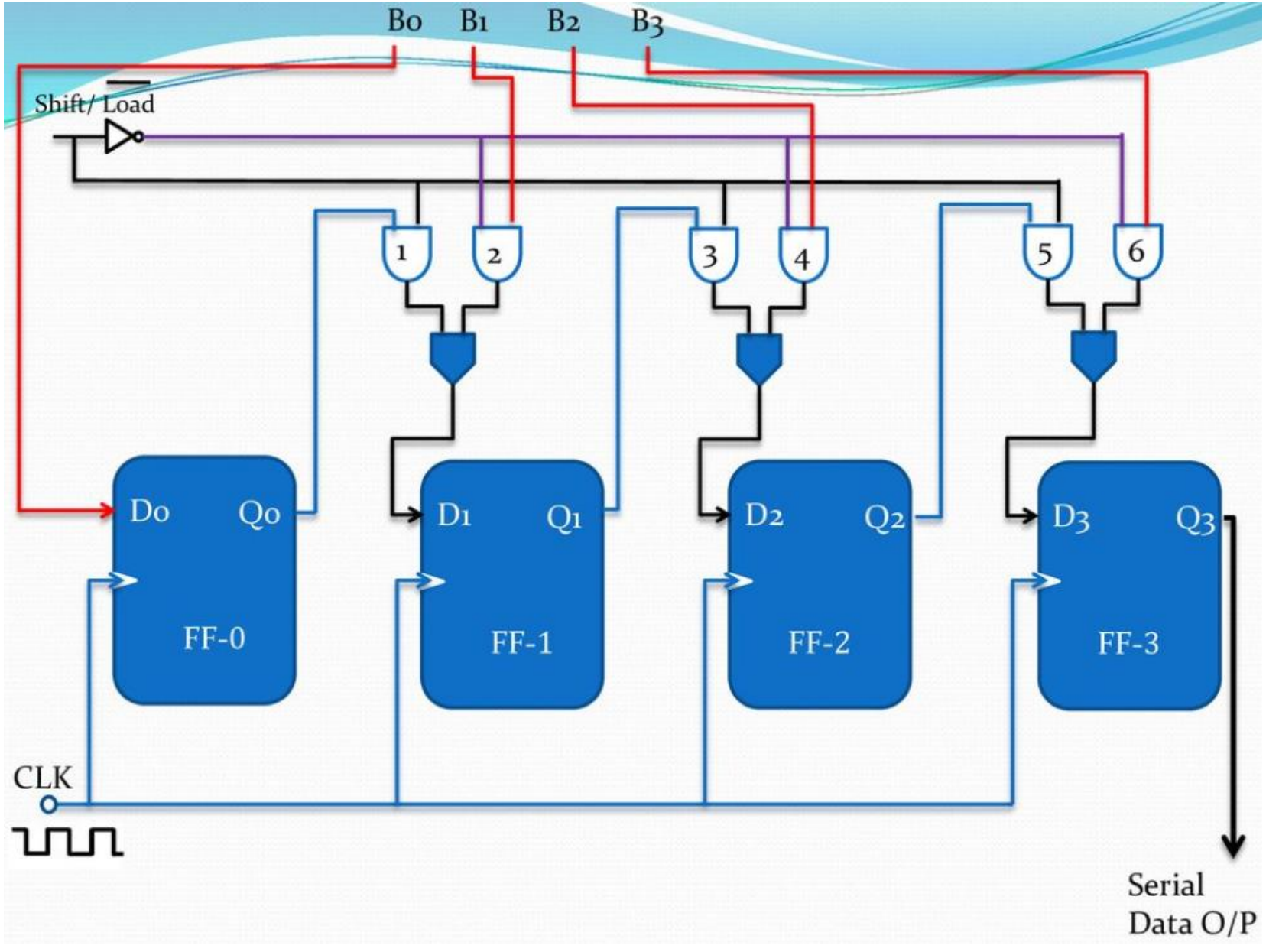
Parallel input Parallel Output (PIPO)

- In this operation the data are entered parallel.
- The 4-bit binary input B_0, B_1, B_2, B_3 is applied to data inputs D_0, D_1, D_2 and D_3 respectively of the four flip-flops.
- As soon as a positive clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously.
- The loaded bits will appear simultaneously to the output side. **ONLY ONE CLOCK IS ESSENTIAL TO LOAD ALL THE BITS.**



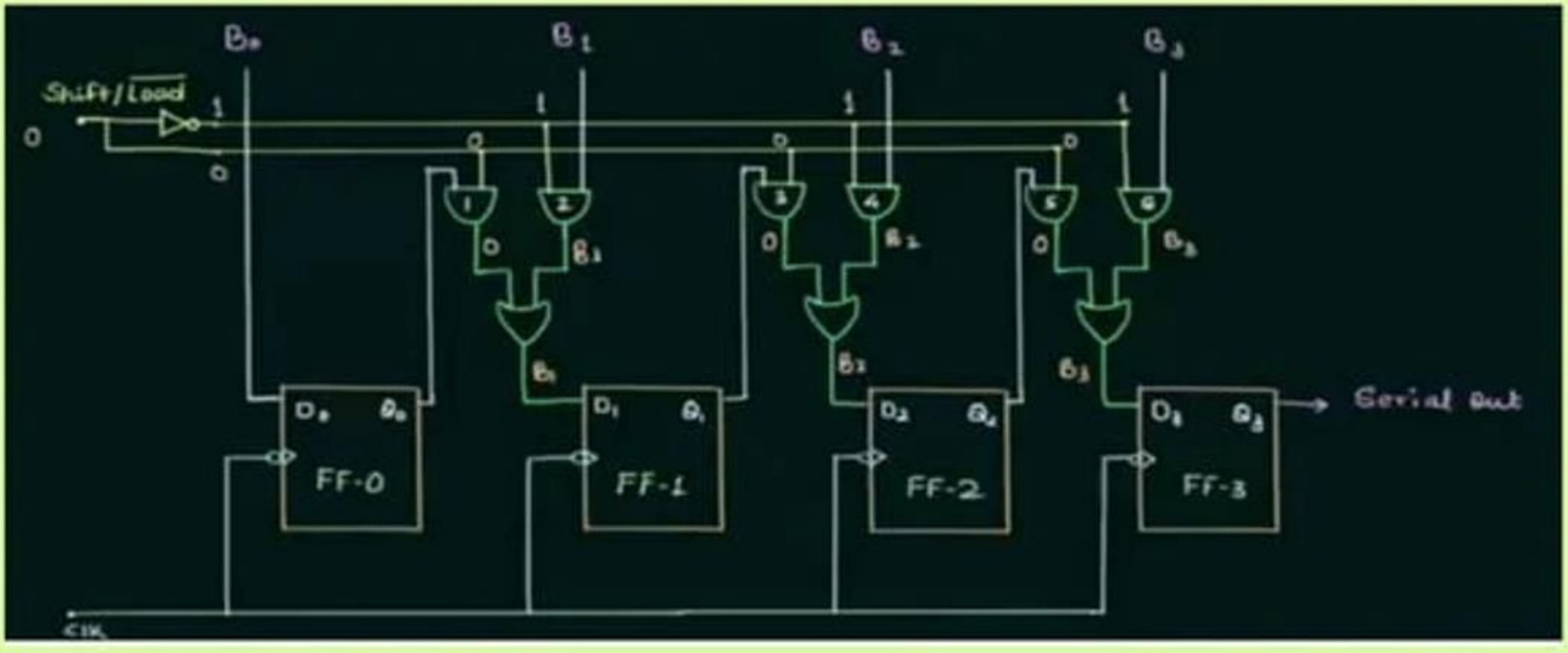
Parallel input Serial Output (PISO)

- In this operation the data are entered parallel.
- Output of pervious FF is connected to the input of the next via a combinational circuit.
- The binary input data B_0, B_1, B_2, B_3 is applied through the same the combinational circuit.
- There are two modes in which this circuit can work namely **shift mode or load mode**.


B0
B1
B2
B3
Shift/ Load
1
2
3
4
5
6
D0
Q0
FF-0
D1
Q1
FF-1
D2
Q2
FF-2
D3
Q3
FF-3
CLK
Serial
Data O/P

Load mode:-

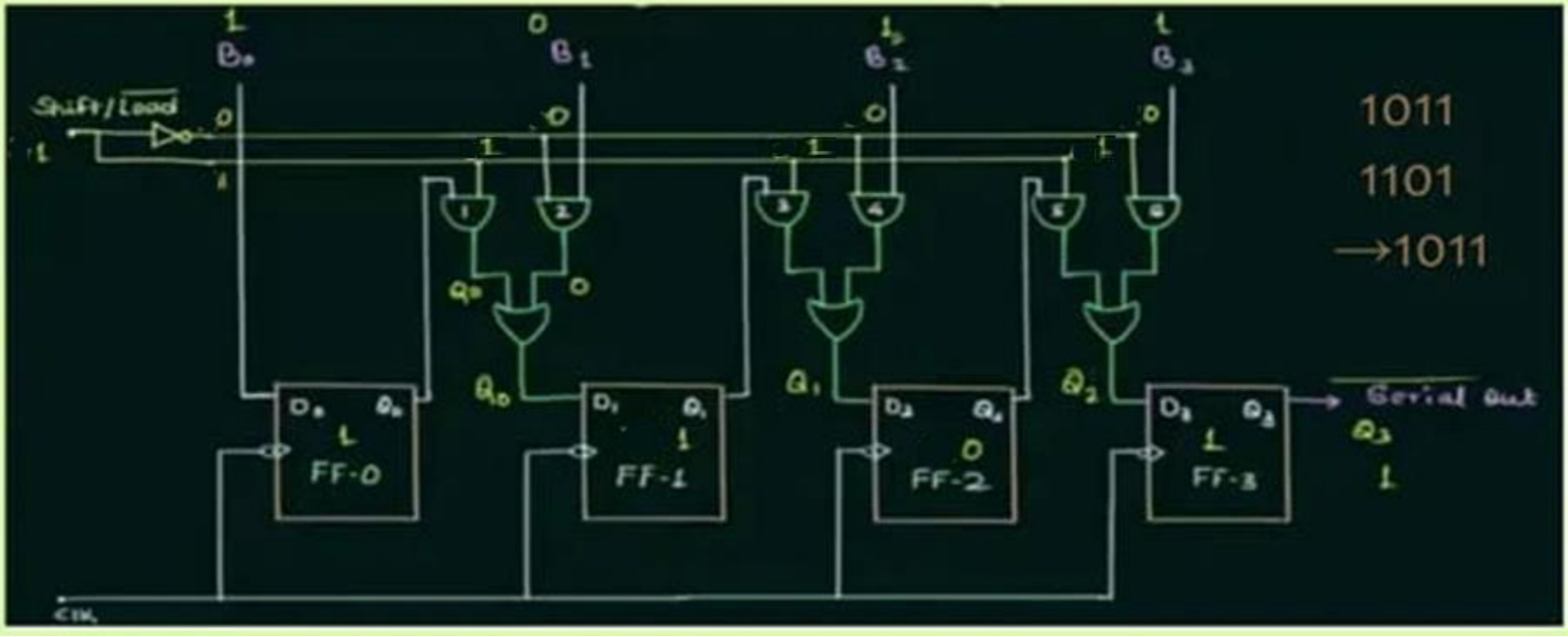
By the load mode we have the parallel input of data in which we have enter or load the data in each of the flip flops simultaneously i.e. in the following example flipflop-0 has bit o, flipflop- 1 has bit-1, flipflop-2 has bit -2 and flipflop-3 has bit-3. It can be achieve by providing logic 0 at the input.



Shift mode:-

The second aim of serial output is achieve mode with this operation. It is responsible for serial movement of data i.e. the data will go from flipflop- 0 to flipflop- 1, flipflop- 1 to flipflop- 2, flipflop- 2 to flipflop- 3 and the data store in flipflop -3 is act as serial output.

It can be achieve by providing logic 1 at the input.

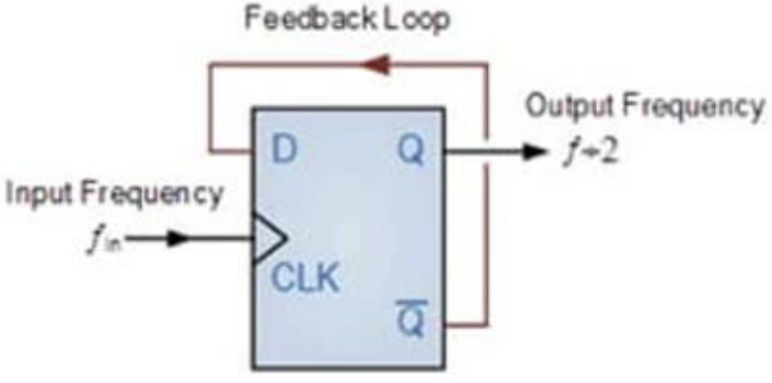


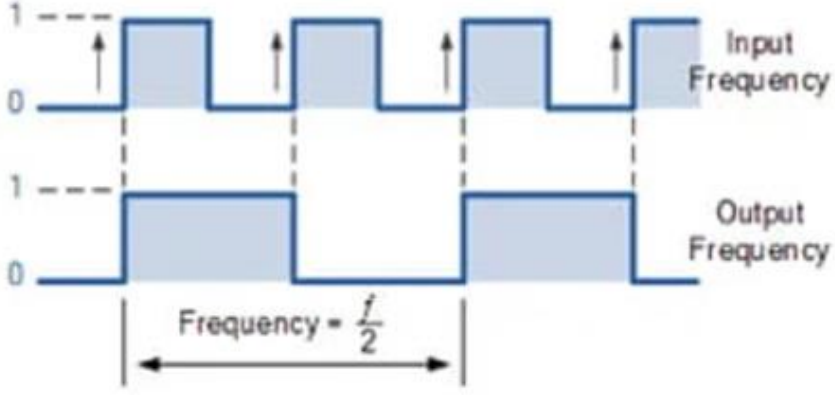
Introduction –COUNTERS

- A *counter* is a register that goes through a predetermined sequence of states upon the application of clock pulses.
  - Asynchronous counters
  - Synchronous counters
- **Asynchronous Counters** (or *Ripple counters*)
  - the clock signal (CLK) is only used to clock the first FF.
  - Each FF (except the first FF) is clocked by the preceding FF.
- **Synchronous Counters**,
  - the clock signal (CLK) is applied to all FF, which means that all FF shares the same clock signal,
  - Modulus (MOD) – the *number of states it counts* in a complete cycle before it goes back to the initial state.
  - Thus, the number of flip-flops used depends on the MOD of the counter (ie; MOD-4 use 2 FF (2-bit), MOD-8 use 3 FF (3-bit), etc..)
  - Example: MOD-4 Ripple/Asynchronous Up-Counter.

Frequency Division :

Frequency division is one of the main propose of counters, the division process is used to reduce the frequency of the clock input waveform.





Divide by 2 counter :

- This type of counters reduces the frequency of the clock input exactly to the half for each flip-flop, this type of counters is called binary ripple counters. For n^{th} FFs , the input frequency reduces by the factor $\frac{f_{in}}{2^n}$.
- For example, the output frequency :
  - After 4 FFs = $\frac{f_{in}}{2^4} = \frac{f_{in}}{16}$
  - After 8 FFs = $\frac{f_{in}}{256}$
  - After 12 FFs = $\frac{f_{in}}{4096}$

Asynchronous (Ripple) Counter :

- Definition : Type of counters in which each flip flop output serves as the clock input signal for the next FF in the sequence.
- In Asynchronous counters , the input of some or all FFs are triggered not by the common clock pulse, but rather by the transition that occurs in other FF output.
- The FFs don't change states at exactly the same time as they don't have a common clock pulse.
- For a ripple counter consists of n FFs , the number of its states equals 2^n , and It can count from 0 to $(2^n - 1)$.

Notes :

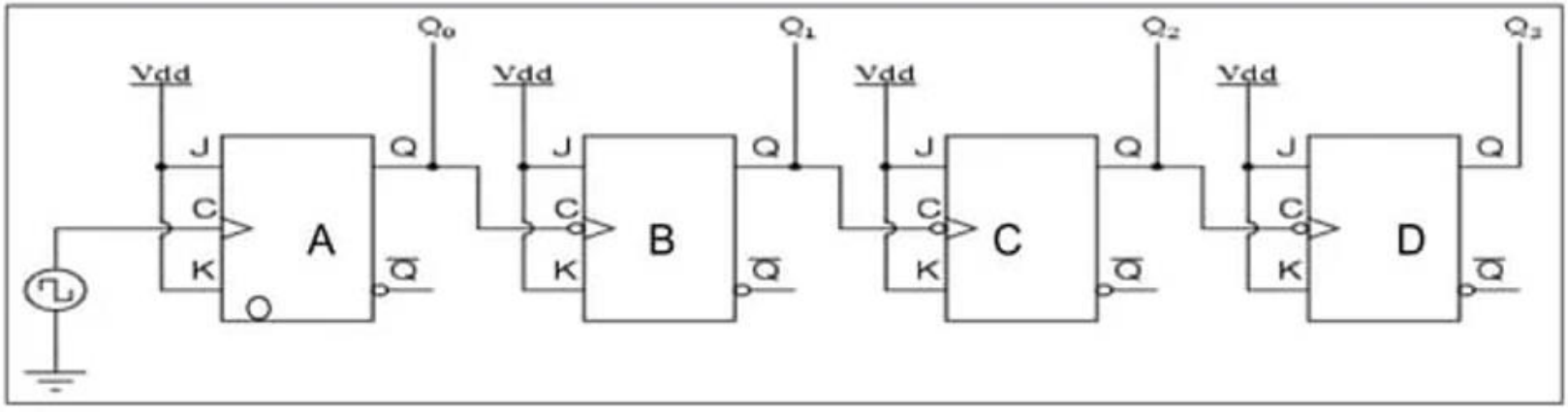
- To operate the toggle mode , the FFs must be connected with J=K=1
- Remember that the FFs are connected in series in Asynchronous systems.

Binary Ripple Counter :

- Binary counter : Group of FFs connected in a special arrangement in which the states of FF represent the binary number equivalent to the number of pulses that have occurred at the input of the counter.
- Binary Ripple counter is called also, a Mod-X counter, where X is the number of counter states and its equal to 2^n , for n FFs .
- The first FF in the counter is called LSB , and the last FF is called MSB.
- The output of MSB divides the input clock frequency by X

- The figure below shows a 4-bit binary counter (Mod-16) :

1. The clock pulse applied only to the Clk input of flip flop A.
2. The input for each of the next FFs is the output of the previous FF. So the output of FF A is the input of FF B , the output of FF B is the input of FF C , and the output of FF C is the input of FF D.
3. FF A will toggle each time the clock pulse make a transition.
4. Since The output of FF A is the input of FF B ,FF B will toggle each time the output of FF A goes from 1 to 0 , and so on.



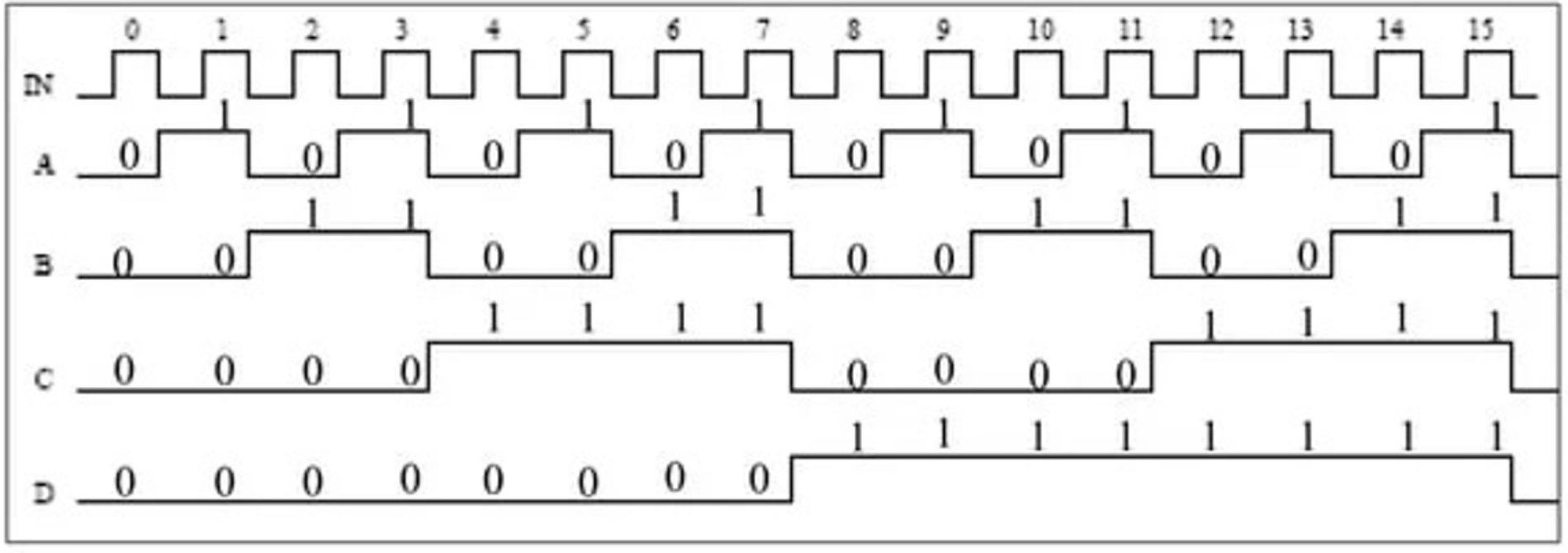
5. This means only FF A responds to the clock pulses. FF B has to wait for FF A to change states before its toggled , and so on.
6. Thus, all FFs don't changes states in exact synchronism with the clock pulse. And because of that it is called asynchronous counters.
7. Because of the phenomenon we talked about in 5 , there is a delay between the response of sequential FFs.
8. So , this type of counters is also commonly called a ripple counter. (simply we can say that the input clock ripples through the counter.)
9. After 15 clock pulse , the counter has finished its first complete cycle (0000 through 1111), and in its 16 clock pulse it will recycle back to (0000) .

Truth table for 4-bit binary counter

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

Note : . A counter may count up or count down or count up and down depending on the input control.

Timing diagram of a 4-bit binary ripple counter



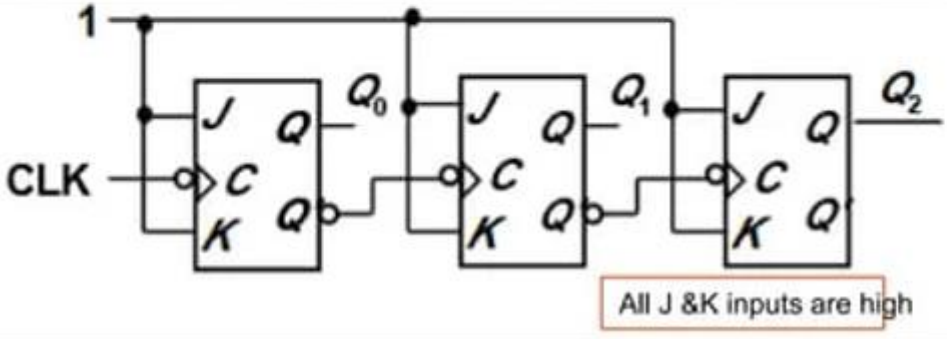
Asynchronous Down Counters :

- The ripple down counters, will count downward from maximum count to 0.
- It works in the same method that the ripple up counters works.

State	CBA
7	111
6	110
5	101
4	100
3	011
2	010
1	001
0	000

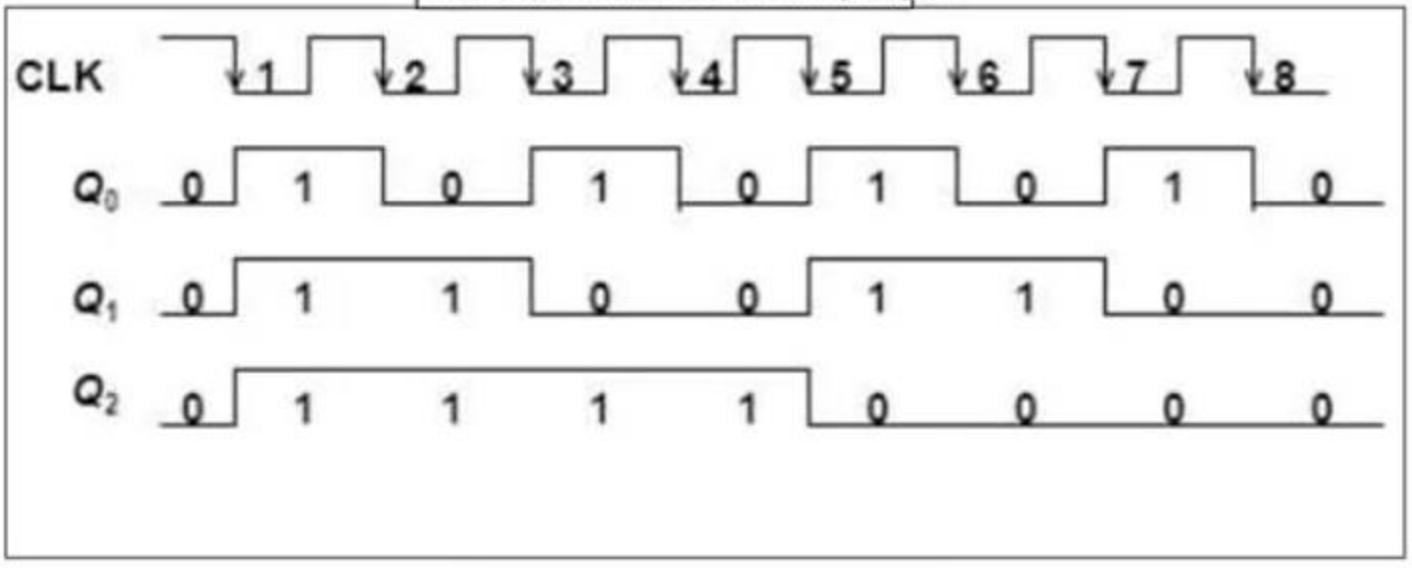
❖ Remark :

Note that when the clK input of FFs is connected to negative edge , the inverted output (Q) of FF0 will be connected to the input clk input for the FF1 , and so on. While , as you see previously, on the up counters when the clK input of FFs is connected to negative edge , the output (Q) of FF0 will be connected to the input clk input for the FF1

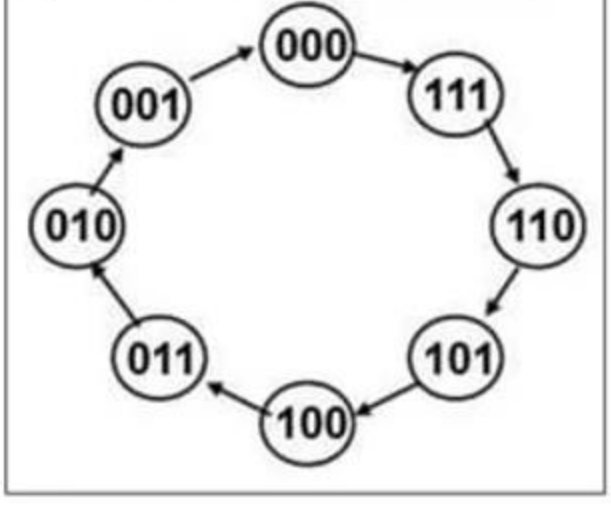


Mod-8 down ripple counter

Mod-8 down ripple counter timing diagram



Mod-8 down ripple counter state diagram



Asynchronous Counters advantages and disadvantages:

- Advantages :

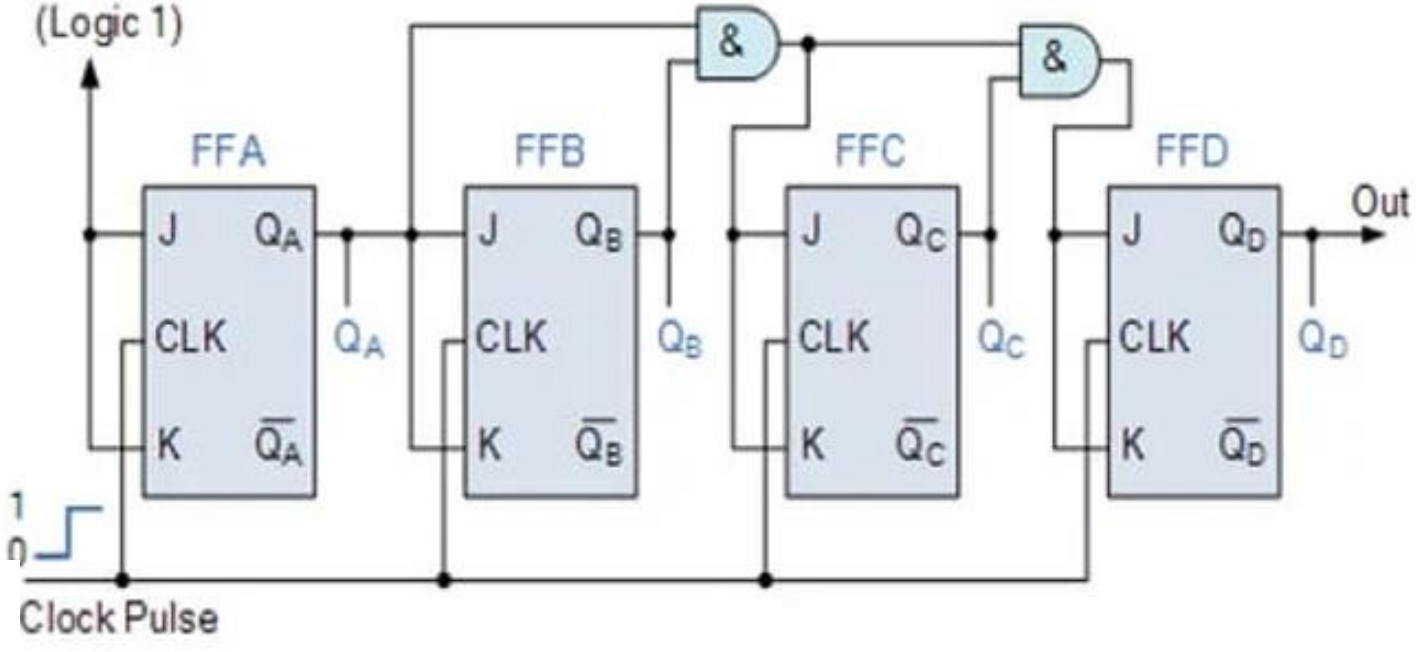
1. Asynchronous Counters can easily be made from Toggle or D-type flip-flops.
2. They are called "Asynchronous Counters" because the clock input of the flip-flops are not all driven by the same clock signal.
3. Each output in the chain depends on a change in state from the previous flip-flops output.
4. Asynchronous counters are sometimes called ripple counters because the data appears to "ripple" from the output of one flip-flop to the input of the next.
5. They can be implemented using "divide-by-n" counter circuits.
6. Truncated counters can produce any modulus number count.

- Disadvantages :

1. An extra "re-synchronizing" output flip-flop may be required.
2. To count a truncated sequence not equal to 2^n, extra feedback logic is required.
3. Counting a large number of bits, propagation delay by successive stages may become undesirably large.
4. This delay gives them the nickname of "Propagation Counters".
5. Counting errors occur at high clocking frequencies.
6. Synchronous Counters are faster and more reliable as they use the same clock signal for all flip-flops.

synchronous binary counters :

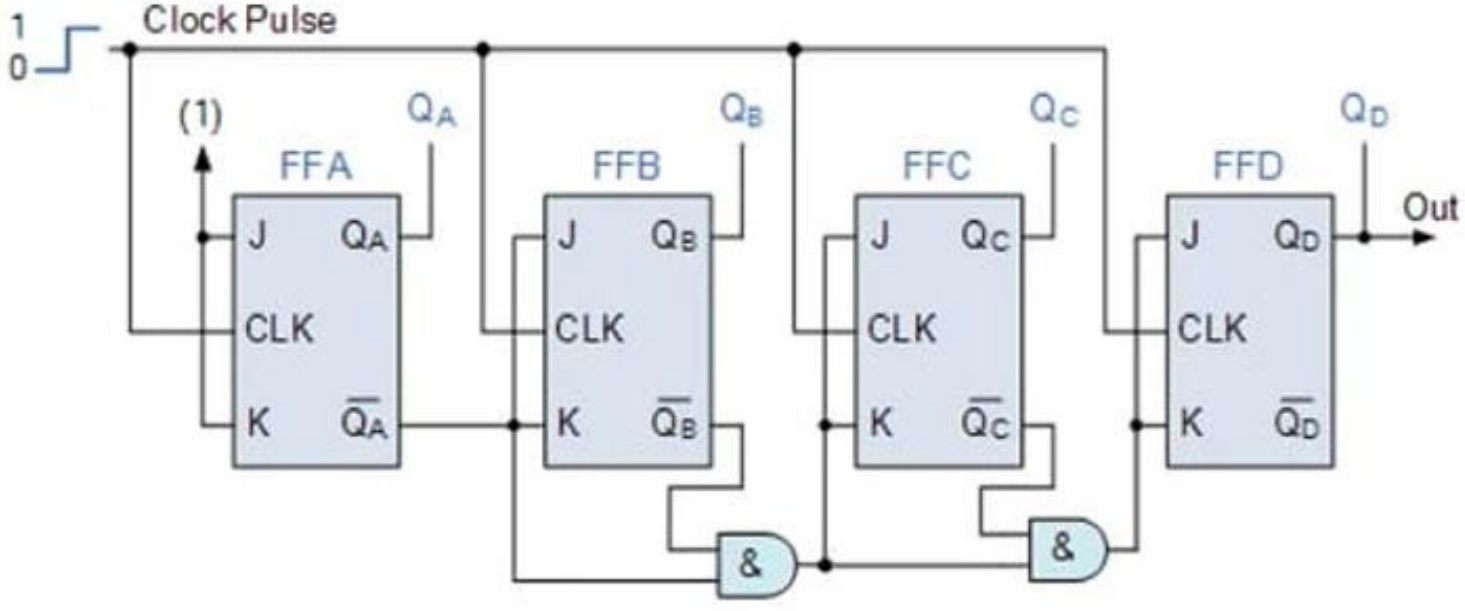
Binary 4-bit Synchronous Up Counter



- the external clock pulses are fed directly to each of the J-K flip flops in the counter chain and that both the J and K inputs are all tied together in toggle mode.
- but only in the first flip-flop, flip-flop FFA (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.
- The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flop FFA
- The J and K inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage.
- And the additional AND gates generate the required logic for the JK inputs of the next stage.

- we can easily construct a 4-bit Synchronous Down Counter by connecting the AND gates to the Q output of the flip-flops.

Binary 4-bit Synchronous Down Counter



Advantages of synchronous counters over Asynchronous :

- In parallel counter all the FFs will change states simultaneously .Thus ,unlike the asynchronous counters , the propagation delays of the FFs do not add thogather to produce the overall delay .

- The total response time of a synchronous counter is the time it take one FF to toggle plus the time for the new logic levels to propagate through a signal AND gate to reach the J,K inputs . that is ,(total delay = FF tpd + AND gate tpd) .

- This total delay is the same no matter how many FFs are in the counter , and it will generally be much lower than an asynchronous counter with the same number of FFs.
- A synchronous counter can operate at much higher frequency , but the circuitry is more complex than that of the asynchronous counter .

How To Design Synchronous Counter

- For synchronous counters, all the flip-flops are using the same CLOCK signal. Thus, the output would change synchronously.
- Procedure to design synchronous counter are as follows:-

STEP 1: Obtain the State Diagram.

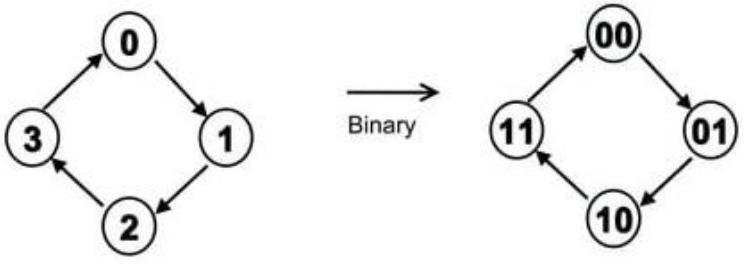
STEP 2: Obtain the Excitation Table using state transition table for any particular FF (JK or D). Determine number of FF used.

STEP 3: Obtain and simplify the function of each FF input using K-Map.

STEP 4: Draw the circuit.

- Design a MOD-4 synchronous up-counter, using JK FF.

STEP 1: Obtain the State transition Diagram



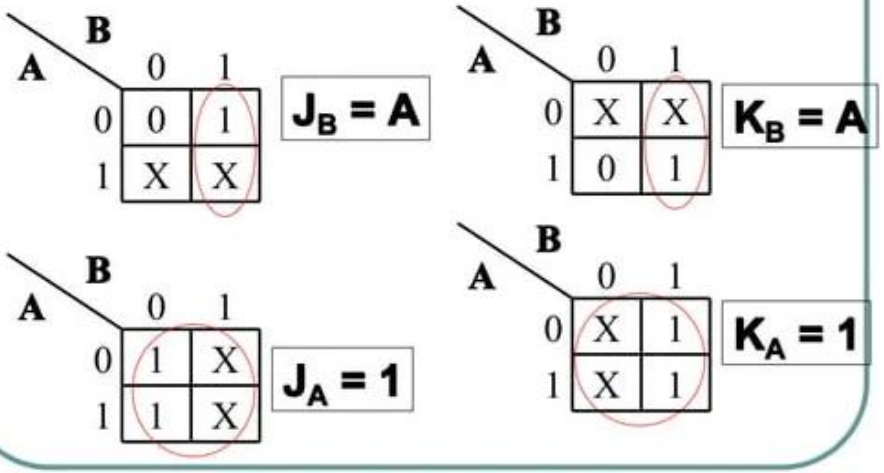
STEP 2: Obtain the Excitation table. Two JK FF are used.

OUTPUT TRANSITION Q_N	Q_{N+1}	FF INPUT J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

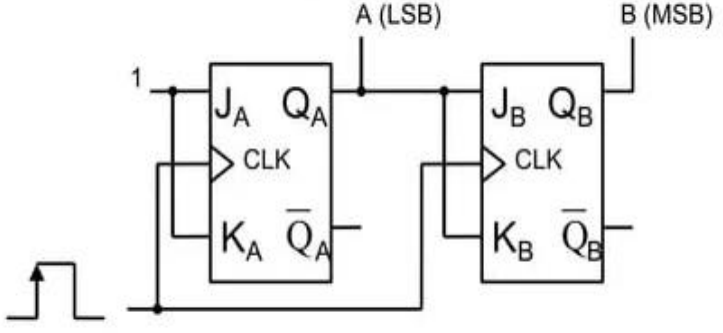
Excitation table

Present State B A	Next State B A	Input, J K J_B K_B	J_A K_A
0 0	0 1	0 X	1 X
0 1	1 0	1 X	X 1
1 0	1 1	X 0	1 X
1 1	0 0	X 1	X 1

STEP 3: Obtain the simplified function using K-Map



STEP 4: Draw the circuit diagram.



(MOD-4 synchronous up-counter)