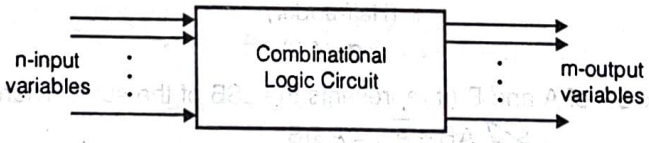
COMBINATION LOGIC 4

Logic circuits for digital systems may be combinational or sequential. A combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. A combinational circuit performs a specific information processing operation fully specified logically by a set of Boolean functions sequential circuits employ momery elements in addition to logic gates. Their outputs are a function of the inputs and the state of the memory elements. The state of memory elements, in turn, is a function of previous inputs. As a consequence, the outputs of a sequential circuit depend not only on present inputs, but also on past inputs, and the circuit behaviour must be specified by a time sequence of inputs and internal states.

The block diagram of a combinational circuit is shown in figure (4.1). The n-input binary variables come from an external source and the m output varibles go to an external destination. For n input variables, there are 2^n possible combinations of binary input values. For each possible input combination, there is one and only one possible output combination.

A combinational circuit can be described by m Boolean funtions one for each output varaibles. Each output function is expressed in terms of the n-input variables. Usually the inputs come from flip-flops and outputs go to flip-flips. So both the variable and its complement are assumed to be available.



(The block diagram of a combinational circuit)

Fig. (4.1)

Design Procedure

The design of combinational circuits starts from the verbal description of the problem and ends in a logic circuit diagram or a set of Boolean functions from which the logic diagram can be easily obtained. The procedure involves the following steps:

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

Adders

The most basic arithmatic operation is the addition of two binary digits. This simple addition consists of four possible operations, namely,

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

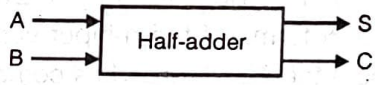
The first three operations produce a sum whose length is one digit, but when both augend and addend bits are equal to 1, the binary sum consists of two digits. The higher significant bit of this result is called a carry. When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits. A combinational circuit that performs the addition of two bits is called a **half adder.** One that performs the addition of three bits (two significant bits and previous carry) is called a **full adder.** The name of the former stems from the fact that two half-adders can be employed to implement a full-adder.

Half-Adder

A half-adder is a combinational circuit with two binary inputs (A and B) and produces the sum (S) and the carry (C) bits. It is an arithmetic circuit used to perform the arithmetic operation of addition of two single bit words. The truth table and block diagram of a half-adder are shown in figure (4.2).

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Truth table



(b) Block diagram

(Half-adder)

Fig. (4.2)

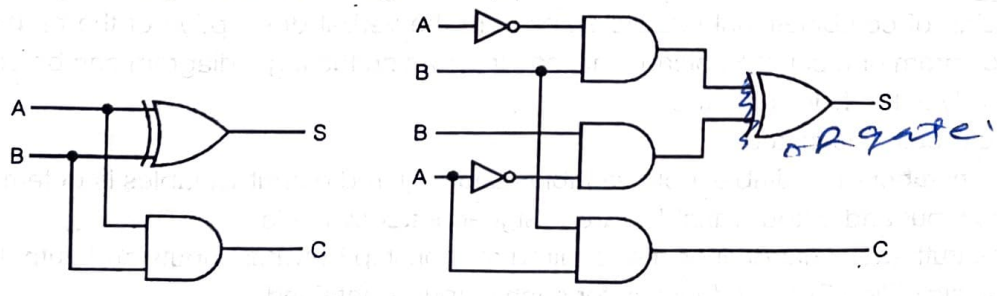
The sum (S) is the Ex-OR of A and B (it represents the LSB of the sum). Therefore,

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

The carry (C) is the AND of A and B (It is 0 unless both the inputs are 1). Therefore,

$$C = AB$$

A half-adder can, therefore, be realized by using one Ex-OR gate and one AND gate as shown in figure 4.3(a). Realization using AOI logic is shown in figure 4.3(b).



(a) (b)

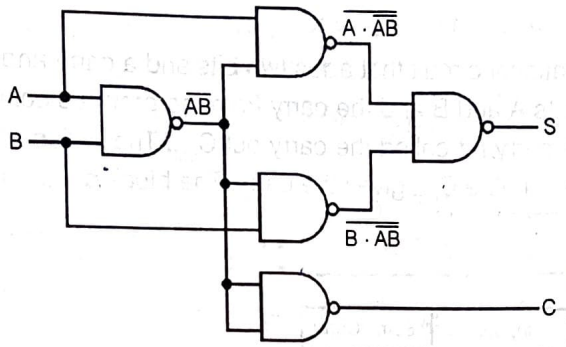
(Logic diagrams of half-adder)

Fig. (4.3)

A half-adder can also be realized in universal logic by using either only NAND gates or only NOR gates as shown in figures (4.4) and (4.5) respectively.

NAND Logic

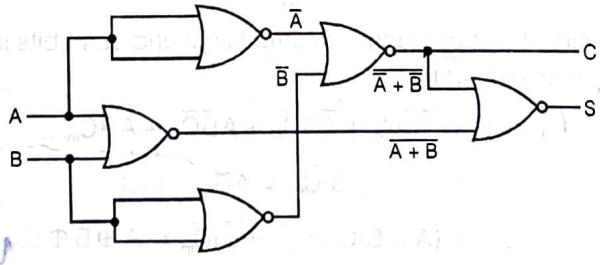
$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + B) + B(\bar{A} + \bar{B}) = A \cdot \overline{A\bar{B}} + B \cdot \overline{A\bar{B}}$$
$$= \overline{A \cdot \overline{A\bar{B}} \cdot B \cdot \overline{A\bar{B}}}$$
$$C = AB = \overline{\overline{AB}}$$



(Logic daigram of a half-adder using only 2-input NAND gates)
Fig. (4.4)

NOR Logic

$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$
$$= (A + B)(\bar{A} + \bar{B}) = \overline{\overline{A + B} + \overline{\bar{A} + \bar{B}}}$$
$$C = AB = \overline{\overline{AB}} = \overline{\bar{A} + \bar{B}}$$



(Logic diagram of a half-adder using only 2-input NOR gates)
Fig. (4.5)

Example 4.1

The addition of two binary variables A and B results into a SUM and a CARRY output. Consider the following expressions for the SUM and CARRY outputs.

1. SUM = $A \cdot B + \bar{A} \cdot \bar{B}$
2. SUM = $A \cdot \bar{B} + \bar{A} \cdot B$
3. CARRY = $A \cdot B$
4. CARRY = $A + B$

Which of these expressions are correct?

(a) 1 and 3 (b) 2 and 3

(c) 2 and 4 (d) 1 and 4

[IES-2003]

Solution: (b)

In Half adder,

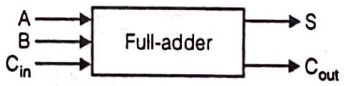
$$\text{Sum } S = A\bar{B} + \bar{A}B$$

$$\text{Carry } C = AB$$

The Full-Adder

A full-adder is a combinational circuit that adds two bits and a carry and outputs a sum bit and a carry bit. The full-adder adds that 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}. The variable 'S' gives the value of the least significant bit of the sum. The variable C_{out} gives the carry. The block diagram and truth table of a full-adder are shown in figure (4.6).

Inputs			Sum	Carry
A	B	C_{in}	S	C_{out}
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



(b) Block diagram

Full-adder

Fig. (4.6)

From the truth table, a circuit that produce the correct sum and carry bits in response to every possible combination of A, B and C_{in} is described by

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

$$= (A\bar{B} + \bar{A}B)\bar{C}_{in} + (AB + \bar{A}\bar{B})C_{in}$$

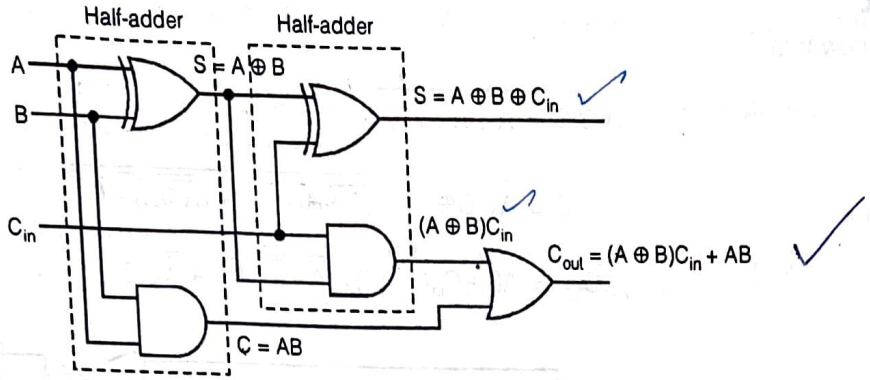
$$= (A \oplus B)\bar{C}_{in} + (\overline{A \oplus B})C_{in} = A \oplus B \oplus C_{in}$$

and

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$

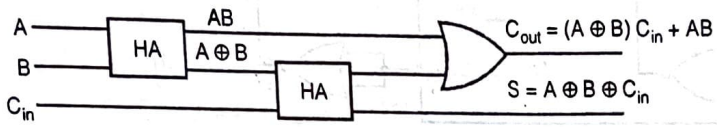
$$= AB + (A \oplus B)C_{in}$$

The sum term of the full-adder is the Ex-OR of A, B and C_{in}, i.e. the sum bit is the modulo sum of the data bits in that column and the carry from the previous column. The logic diagram of the full-adder using two Ex-OR gates and two AND gates (i.e. two half-adders) and one OR gate is shown in figure (4.7).



Logic diagram of a full-adder using two half-adders
Fig. (4.7)

The block diagram of a full-adder using two half-adders is shown in figure (4.8).



Block diagram of a full-adder using two half-adders
Fig. (4.8)

The full-adder can also be realized using universal logic gates as shown in figures (4.9) and (4.10) respectivley.

NAND Logic

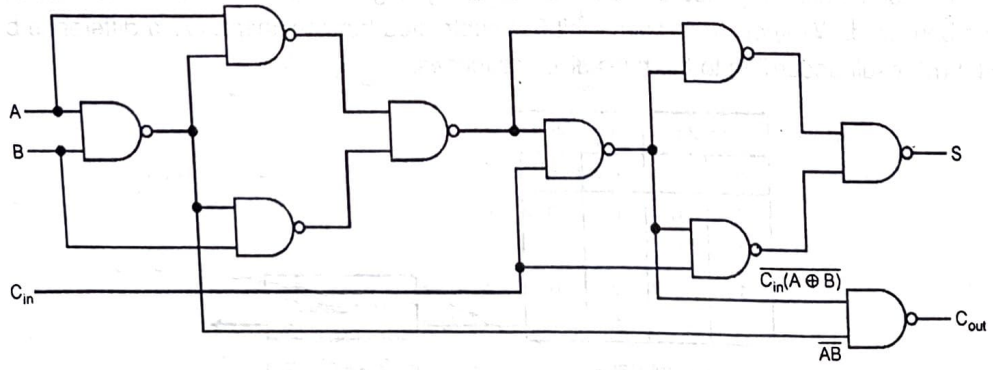
We know that,

$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

Then,

$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) \cdot \overline{(A \oplus B)C_{in}}} \cdot \overline{C_{in} \cdot \overline{(A \oplus B)C_{in}}}}$$

$$C_{out} = C_{in}(A \oplus B) + AB = \overline{\overline{C_{in}(A \oplus B)} \cdot \overline{AB}}$$



(Logic diagram of a full-adder using only 2-input NAND gates)
Fig. (4.9)

NOR Logic

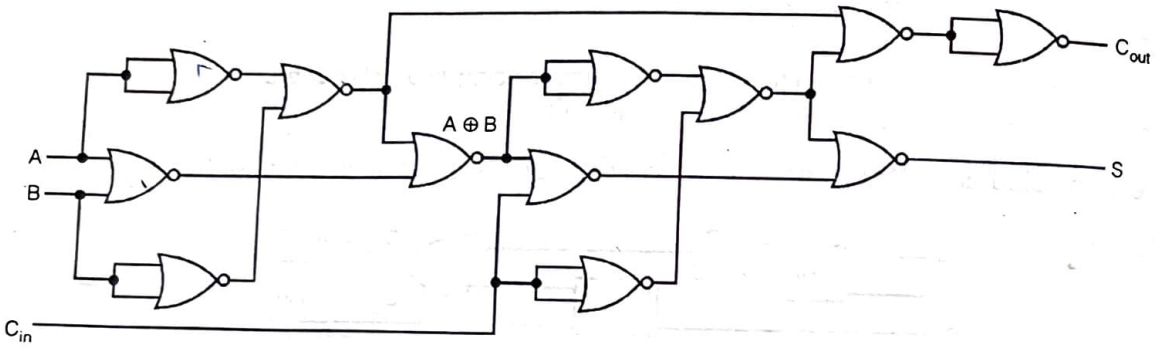
We know that,

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

Then,

$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) + C_{in}} + \overline{(A \oplus B) + \overline{C_{in}}}}$$

$$C_{out} = AB + C_{in}(A \oplus B) = \overline{\overline{A}+\overline{B}} + \overline{\overline{C_{in}} + \overline{A \oplus B}}$$



(Logic diagram of a full-adder using only 2-input NOR gates)

Fig. (4.10)

Subtractors

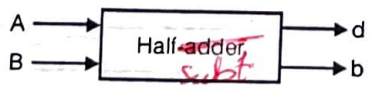
The subtraction of two binary numbers may be accomplished by taking the complement of the subtrahend and adding it to the minuend. By this method, the subtraction operation becomes an addition operation and instead of having a separate circuit for subtraction, the adder itself can be used to perform subtraction. This results in reduction of hardware.

The Half-Subtractor

A half-subtractor shown in figure (4.11) is a comibational circuit with two inputs A and B and two outputs d and b. d indicates the difference and b is the output signal generated that informs the next stage that a 1 has been borrowed. We know that, when a bit B is subtracted from another bit A, a difference bit (d) and a borrow bit (b) result according to the rules given as follows.

Inputs		Outputs	
A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

(a) Truth table



(b) Block diagram

(Half-subtractor)

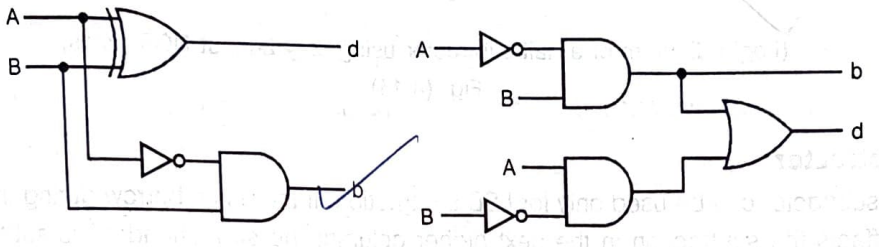
Fig. (4.11)

The output borrow b is a 0 as long as A ≥ B. It is a 1 for A = 0 and B = 1. The d output is the result of the arithmetic opeation A – B.

A circuit that produces the correct difference and borrow bits in response to every possible combination of the two 1-bit numbers is, therefore, described by

$$d = A\bar{B} + \bar{A}B = A \oplus B \quad \text{and} \quad b = \bar{A}B$$

Figure (4.12) shows two logic diagrams of a half-subtractor – one using an Ex-OR gate together with one each NOT gate and AND gate and the other using the AOI gates. Note that the logic for d is exactly the same as the logic for output S in the half-adder.



(Logic diagrams of a half-subtractor)

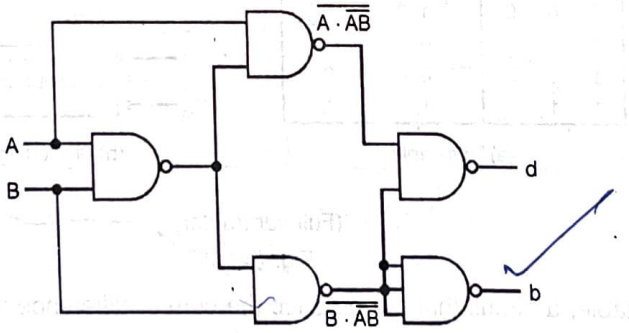
Fig. (4.12)

A half-subtractor can also be realized using universal logic gates as shown in figure (4.13) and (4.14) respectively.

NAND Logic

$$d = A \oplus B = \overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}$$

$$b = \bar{A}B = B(\bar{A} + \bar{B}) = B(\overline{AB}) = \overline{B \cdot \overline{AB}}$$



(Logic diagram of a half-subtractor using only 2-input NAND gates)

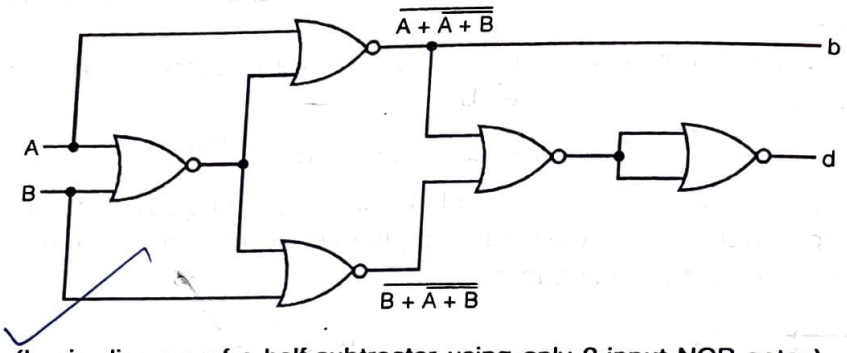
Fig. (4.13)

NOR Logic

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

$$= \bar{B}(A + B) + \bar{A}(A + B) = \overline{B + \overline{A + B}} + \overline{A + \overline{A + B}}$$

$$b = \bar{A}B = \bar{A}(A + B) = \overline{\bar{A}(A + B)} = \overline{A + \overline{(A + B)}}$$



(Logic diagram of a half-subtractor using only 2-input NOR gates)

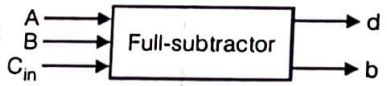
Fig. (4.14)

The Full-Subtractor

The half-subtractor can be used only for LSB subtraction. If there is a borrow during the subtraction the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from th minuend bit, considering the borrow from that column used for the subtraction in the preceding column. Suc a subtraction is performed by a full-subtractor. The truth table and the block diagram of a full-subtractor ar shown in the figure (4.15).

Inputs			Difference	Borrow
A	B	b_i	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

(a) Truth table



(b) Block diagram

(Full-subtractor)

Fig. (4.15)

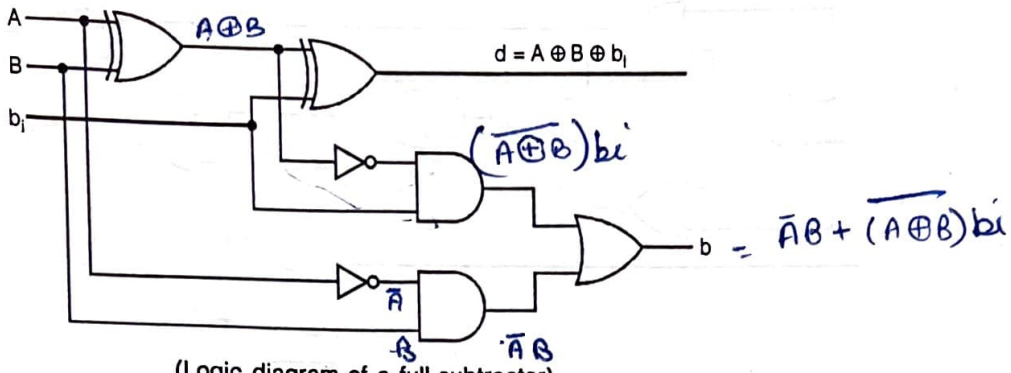
From the truth table, a circuit that will produce the correct difference and borrow bits in response to every possible combination of A, B and b_i is described by

$$d = \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + A\bar{B}\bar{b}_i + ABb_i$$
$$= b_i(AB + \bar{A}\bar{B}) + \bar{b}_i(A\bar{B} + \bar{A}B)$$
$$= b_i(\overline{A \oplus B}) + b_i(A \oplus B) = A \oplus B \oplus b_i$$

and

$$b = \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + \bar{A}Bb_i + ABb_i$$
$$= \bar{A}B(b_i + \bar{b}_i) + (AB + \bar{A}\bar{B})b_i$$
$$= \bar{A}B + (\overline{A \oplus B})b_i$$

A full-subtractor can, therefore, be realized using Ex-OR gates and AOI gates as shown in figure (4.16).



(Logic diagram of a full-subtractor)

Fig. (4.16)

The full-subtractor can also be realized in universal logic gates shown in figure (4.17) and (4.18).

NAND Logic

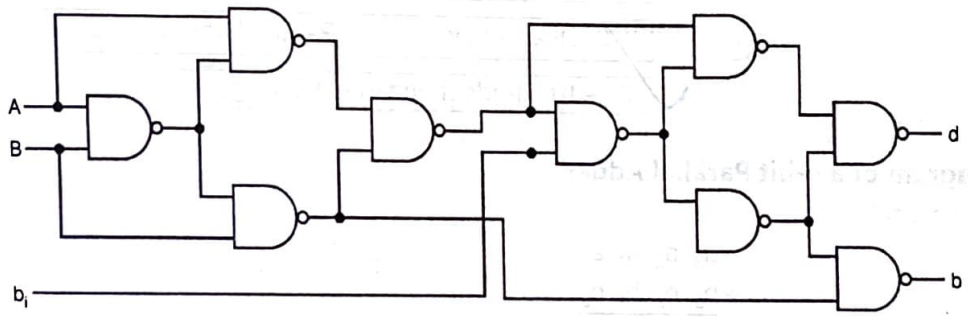
$$d = A\oplus B\oplus b_i = \overline{\overline{(A\oplus B)\oplus b_i}}$$

$$= \overline{\overline{(A\oplus B)}\,\overline{(A\oplus B)b_i}\cdot b_i\overline{(A\oplus B)b_i}}$$

$$b = \bar{A}B + b_i\overline{(A\oplus B)} = \overline{\overline{\bar{A}B + b_i\overline{(A\oplus B)}}}$$

$$= \overline{\overline{\bar{A}B}\cdot b_i\overline{(A\oplus B)}} = \overline{B(\bar{A}+\bar{B})\cdot b_i(\bar{b}_i + \overline{(A\oplus B)}}$$

$$= \overline{B\cdot\overline{A\bar{B}}\cdot b_i[b_i\cdot\overline{(A\oplus B)}]}$$



Logic diagram of a full-subtractor using 2-input NAND gates

Fig. (4.17)

NOR Logic

$$d = A\oplus B\oplus b_i = \overline{\overline{(A\oplus B)\oplus b_i}} = \overline{(A\oplus B)b_i + \overline{(A\oplus B)}\bar{b}_i}$$

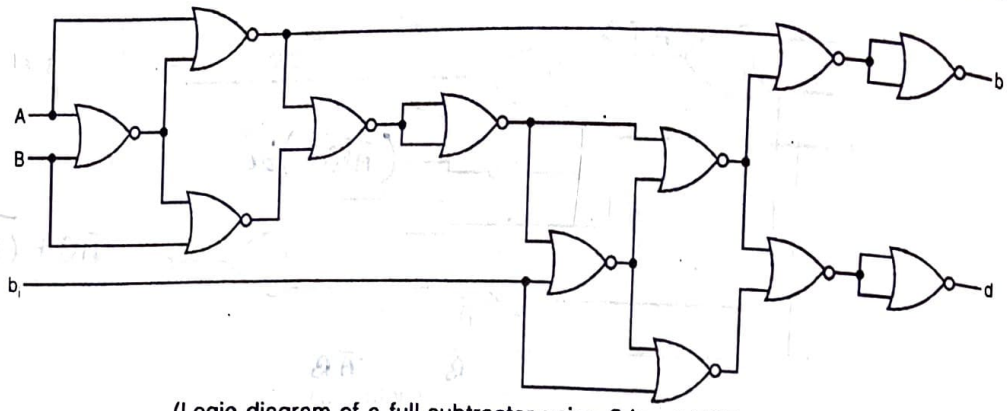
$$= \overline{\left[(A\oplus B) + \overline{(A\oplus B)}\bar{b}_i\right]\left[b_i + \overline{(A\oplus B)}\bar{b}_i\right]}$$

$$= \overline{(A\oplus B) + \overline{(A\oplus B)} + b_i} + \overline{b_i + \overline{(A\oplus B)} + b_i}$$

$$= \overline{\overline{(A\oplus B) + \overline{(A\oplus B)} + b_i} + \overline{b_i + \overline{(A\oplus B)} + b_i}}$$

$$b = \bar{A}B + b_i\,\overline{(A\oplus B)} = \bar{A}(A+B) + \overline{(A\oplus B)}\,[(A\oplus B)+b_i]$$

$$= \overline{\overline{A + \overline{(A+B)}} + \overline{(A\oplus B)} + \overline{(A\oplus B)} + b_i}$$



(Logic diagram of a full-subtractor using 2-input NOR gates)

Fig. (4.18)

Parallel Adder

It is a combinational digital circuit that performs the addition of two binary numbers in parallel.

It consists of full-adders connected in cascade, with the **output carry** from one adder connected to the **input carry** of the next adder. **"Parallel Adder"** is also called **"RIPPLE-CARRY ADDER"**. This adder is used to add **two n-bit numbers**, for this purpose number of full-adders required is **"n"**.

Hence, "n" full-adder

or, (n – 1) full-adder and 1 half-adder

or, (n – 1) {2 · half-addder + 1 – OR gate} and 1 half-adder

or, {2n – 2) half-adder + (n – 1) OR – gate} and 1 half-adder

or, (2n – 1) half-adder and (n – 1) OR – gate

Block Diagram of a 4-bit Parallel Adder

Let us consider,

$$\begin{array}{rl} & a_3\ a_2\ a_1\ a_0 \\ +\ & b_3\ b_2\ b_1\ b_0 \\ \hline \text{Carry} \Leftarrow c_4 & s_3\ s_2\ s_1\ s_0 \end{array}$$

Final SUM result

also if the initial input carry = C_0 then,

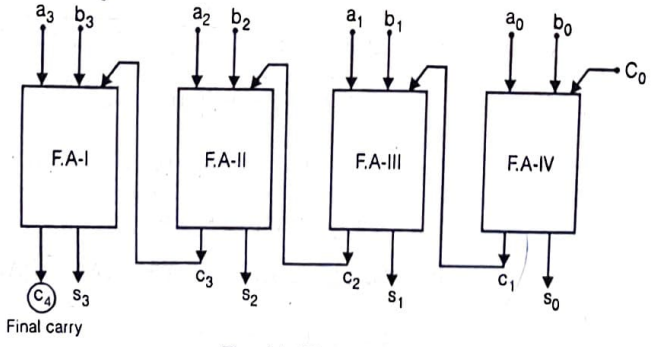


Fig. (4.19)

In parallel adder, carry propagation delay is present. In "n-bit parallel adder", minimum delay provided to the final result is equals to $2nt_{pd}$ (where t_{pd} = propagation delay of gate) that's why parallel adder becomes very slow. To overcome this difficulty a new type of adder is used which is called "Look Ahead Carry Adder (LCA)". This LCA is the Fastest carry adder among all Adders. (HA, FA, PA etc.) LCA is the most widely used technique employs for reducing the carry propagation time in a parallel adder. Consider the circuit of the full-adder shown in [figure 4.20(a)] and also we have,

$P_i = A_i \oplus B_i \Rightarrow$ carry propagate

$G_i = A_i B_i \Rightarrow$ carry generate

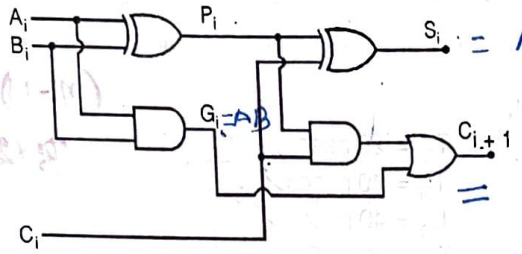


Fig. 4.20(a)

$S_i = P_i \oplus C_i$

and output carry $= C_{i+1} = G_i + P_i C_i$

$\therefore$

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1)$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

From above Boolean function it is clear that C_4 does not have to wait for C_3 and C_2 to propagate. Infact C_4 is propagated at the same time as C_2 and C_3. So that delay for output carry must be reduced. The construction of a 4-bit parallel adder with a Look-ahead carry scheme is shown in [figure 4.20(b)].

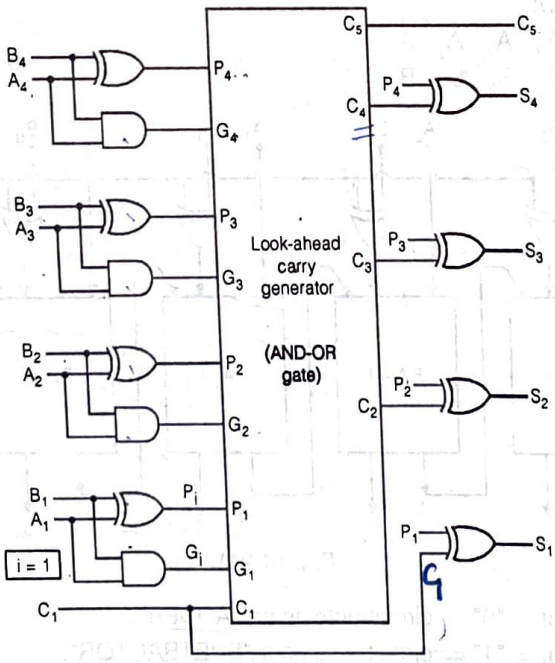


Fig. 4.20(b)

Each sum(s) output requires two EX-OR gates.
In LCA, to produce carry(C), it requires 3-gate delay and to produce sum it requires 4-gate delay.

Example 4.2

A 1-bit full adder takes 20 nsec to generate carry-out bit and 40 nsec for the sum bit. What is the maximum rate of addition per second when four 1-bit full adders are cascaded?

(a) 10^7 (b) 1.25×10^7
(c) 6.25×10^6 (d) 10^5

[IES-2005]

Solution: (a)

Here given that,

$T_{Ci} = 20$ nsec
$T_{Si} = 40$ nsec

It is clear that,

$T_{s0} = 40$ nsec
$T_{s1} = (40 + 20)$ nsec
$T_{s2} = (60 + 20)$ nsec
$T_{s3} = (80 + 20)$ nsec

Final, sum result will take 100 nsec

$$\therefore \text{ Rate of addition/sec} = \frac{1}{T_{s3}} = \frac{1}{100 \text{ nsec}}$$

$$\boxed{R = 10^7}$$

2's Complemented Adder

If we perform (A – B) operation, then we take it as A +(–B) i.e. A(as it is) + (2's complement of B) For this purpose we use a control circuit by using EX-OR gate. Consider a 4-bit 2's complement parallel adder as shown in figure (4.21) below:

Let $A = A_3\ A_2\ A_1\ A_0$
$B = B_3\ B_2\ B_1\ B_0$

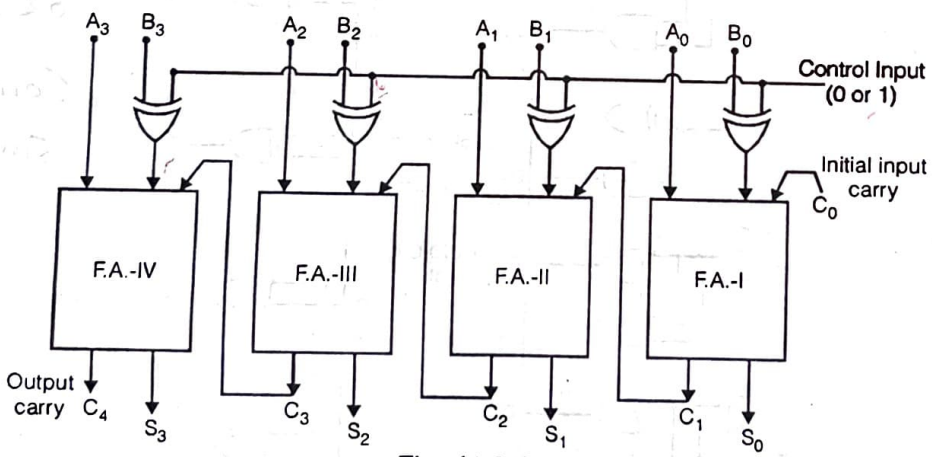


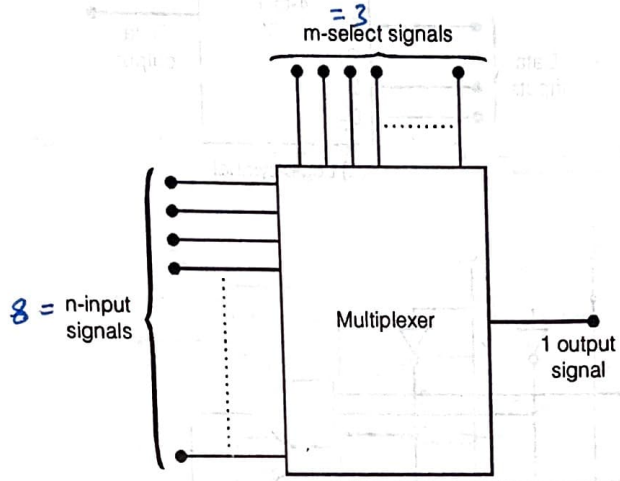
Fig. (4.21)

- When control input = "0" ⇒ circuit acts as an "ADDER".
- When control input = "1" ⇒ circuit acts as a "SUBTRACTOR".

Multiplexers

The term 'multiplex' means **"many into one"**. Mutliplexing is the process of transmitting a large number of information over a single line. A digital multiplexer (MUX) is a combinational circuit that selects one digital information from several sources and transmits the selected information on a single output line. A multiplexer is also called a **data seletor** since it selects one of many inputs and steers the information to the output.

The multiplexer has several data-input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines. The block diagram of a multiplexer with n input lines, m select signals and one output line is shown in figure (4.22). The selection lines decide the number of input lines of a particular multiplexer. If the number of n input lines is equal to 2^m, then m select lines are required to select one of the n input lines.



Block diagram of multiplexer

Fig. (4.22)

Basic Four-input Multiplexer

The logic symbol of a 4-to-1 multiplexer is shown in figure 4.23 (a). It has four data input lines ($D_0 - D_3$), a single output line (Y) and two select lines (S_0 and S_1) to select one of the four input lines. The truth table for a 4-to-1 multiplexer is shown in table.

From the truth table a logical expression for the output in terms of the data input and select inputs can be derived as follows:

The data output Y = data input D_0, if and only if $S_1 = 0$ and $S_0 = 0$.

Therefore, $$Y = D_0 \bar{S}_1 \bar{S}_0 = D_0 \bar{0}\bar{0} = D_0 1.1 = D_0$$

The data input Y = D_1, if and only if $S_1 = 0$ and $S_0 = 1$.

Therefore, $$Y = D_1 \bar{S}_1 S_0 = D_1, \text{ when } S_1 S_0 = 01$$

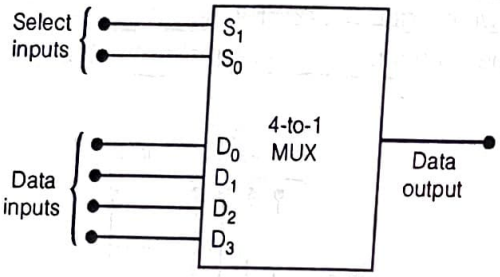
Similarly, $$Y = D_2 S_1 \bar{S}_0 = D_2, \text{ when } S_1 S_0 = 10 \text{ and}$$
$$Y = D_3 S_1 S_0 = D_3, \text{ when } S_1 S_0 = 11$$

If the above terms are ORed, then the final expression for the data output is given by

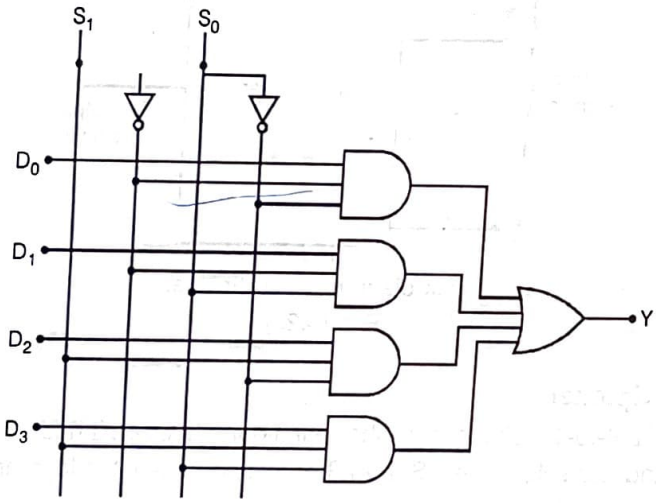
$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0$$

Using the above expression; the 4-to-1 multiplexer can be implemented using two NOT gates, four 3-input AND gates and one 4-input OR gate as shown in figure 4.23(b).

Data select inputs		Outputs
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



(a) Logic symbol



(b) Logic diagram

4-to-1 multiplier

Fig. (4.23)

2 : 1 Multiplexer

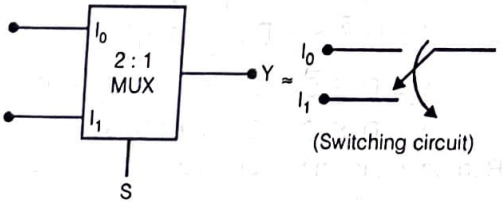


Fig. 4.24(a)

$$Y = \bar{S}I_0 + SI_1 \quad ...(5.2)$$

Logic diagram of 2 : 1 multiplexer:

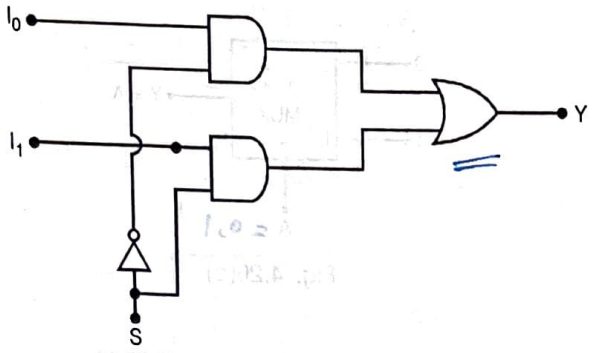


Fig. 4.24(b)

Implementation of higher order multiplexer using lower order multiplexer

4 : 1 MUX by 2 : 1 MUX

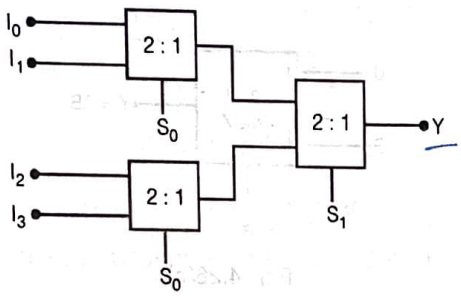


Fig. (4.25)

Total number of 2 : 1 MUX = 3

⇒ To implement 2^n : 1 MUX by using 2 : 1 MUX, the total number of 2 : 1 MUX required is $(2^n - 1)$.

Given MUX	To be implemented MUX	Required No. of MUX
4 : 1	16 : 1	4 + 1 = 5
4 : 1	64 : 1	16 + 4 + 1 = 21
8 : 1	64 : 1	8 + 1 = 9
8 : 1	256 : 1	32 + 4 + 1 = 37

Multiplexer circuit or ICs may have an enable input to control operation of the unit. When this input is in a given binary state the outputs are disabled and when in the other state, the circuit function as a normal 'multiplexer'. The multiplexer is a very useful MSI function and has a multitude of applications. It is used for connecting two or more sources to a single destination among computer units, and it is useful for constructing a common bus system.

MUX as an Universal Logic gate

Buffer

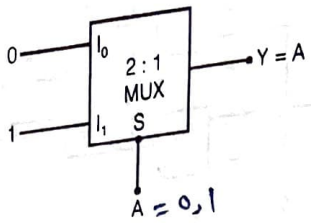


Fig. 4.26(a)

NOT-gate/Inverter

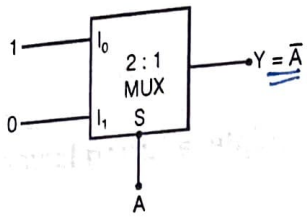


Fig. 4.26(b)

AND-gate

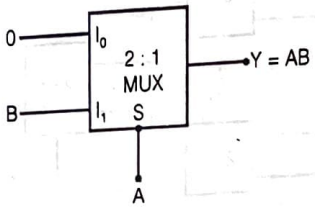


Fig. 4.26(c)

OR-gate

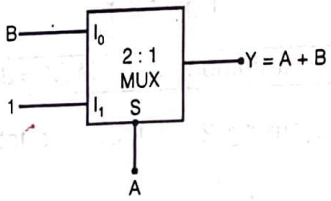


Fig. 4.26(d)

NOR-gate

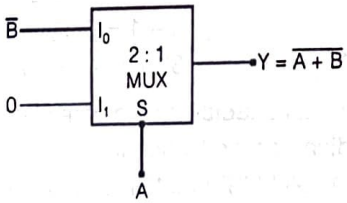


Fig. 4.26(e)

NAND-gate

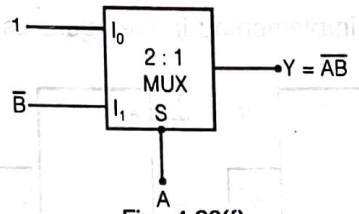


Fig. 4.26(f)

Ex-OR-gate

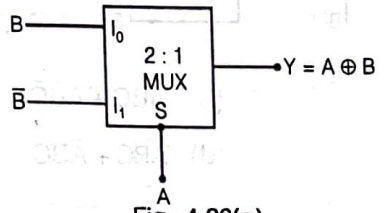


Fig. 4.26(g)

Ex-NOR gate

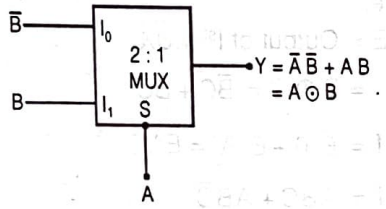


Fig. 4.26 (h)

Boolean Function Implementation by Using MUX

- For implementing any Boolean function of **n-variables** with a 2^{n-1} : 1 **MUX**, we require some general procedure.
- Express the function in its sum of minterms form.
- In the ordered sequence of **n variables**, connect the **(n-1)** variables to the select line and the single highest order position variable to the input line with complemented or uncomplemented form including 0 and 1.
- List the inputs of MUX (all the minterms) in two rows. The **Ist row** lists all those **minterms** where single variable is **complemented** and in **2nd row** with **uncomplemented** form.
- **Circle** all the minterms of the function and inspect each column separately.
- If two minterms in a column are **not circled**, apply **'0'** to the corresponding MUX input.
- If two minterms are **circled**, apply **'1'** to the corresponding MUX input.
- If the **one minterm** is **circled** (either upper row or lower row), then its **front value** is the corresponding MUX input.

Example 4.3

Construct a 4-input multiplexer using four 3-input AND gats, an OR gate and three inverters. Show the input, output and select lines and write a table showing the outputs for various select inputs.

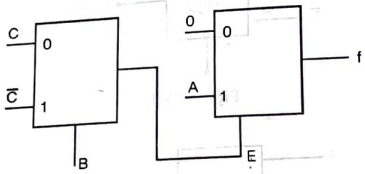
[IES-2005 : 10 marks]

Solution:

Refer preceding articles.

Example 4.4

The Boolean function f implemented in the figure using two input multiplexers is



(a) $A\bar{B}C + AB\bar{C}$ (b) $ABC + A\bar{B}\bar{C}$

(c) $\bar{A}BC + \bar{A}\bar{B}\bar{C}$ (d) $\bar{A}BC + \bar{A}B\bar{C}$

[GATE-2005]

Solution: (a)

From the figure we have,

E = Output of Ist MUX

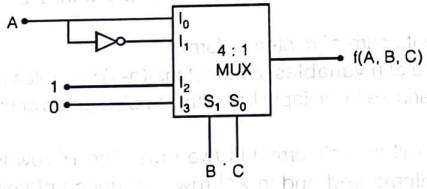
$= B \oplus C = \bar{B}C + B\bar{C}$

and, $f = \bar{E}\cdot 0 + E\cdot A = EA$

$f = A\bar{B}C + AB\bar{C}$

Example 4.5

The output of the given MUX equals to?



(a) $f(A, B, C) = \Sigma m(1, 2, 4, 6)$ (b) $f(A, B, C) = \Sigma m(1, 2, 6)$

(c) $f(A, B, C) = \Sigma m(2, 4, 5, 6)$ (d) $f(A, B, C) = \Sigma m(1, 5, 6)$

Solution: (a)

From the given 4 : 1 MUX we get,

$$f(A, B, C) = A\bar{B}\bar{C} + \bar{A}\bar{B}C + B\bar{C}\cdot 1$$

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

$$= A\bar{B}\bar{C} + \bar{A}\bar{B}C + AB\bar{C} + \bar{A}B\bar{C}$$

$$\approx 100,\quad 001,\quad 110,\quad 010$$

$$\boxed{f(A, B, C) \approx \Sigma m(1, 2, 4, 6)}$$

Example 4.6

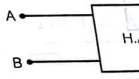
How many minimum number of 2 : 1 M

(a) 2 (b)

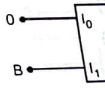
(c) 4 (d

Solution: (b)

We know, H.A. has a SUM = A ⊕ B a variables i.e.



⇒ For carry, we require an AND-



⇒ For SUM, we require an EX

But we don't available $\bar{B}$ as i

Finally, to implement a hal

[For SUM ⇒ 2 and for CA

Example 4.7

What are the minimu

2-input AND gate and

(a) 1 and 2

(c) 1 and 1

Solution: (a)

Problem should be d

So, for AND gate ⇒

Example 4.6

How many minimum number of 2 : 1 MUX required to implement the Half Adder (H.A)

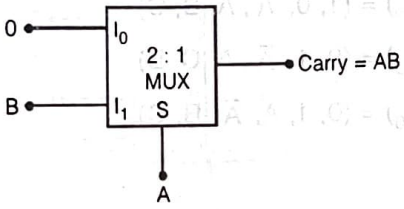
(a) 2 (b) 3

(c) 4 (d) None of these

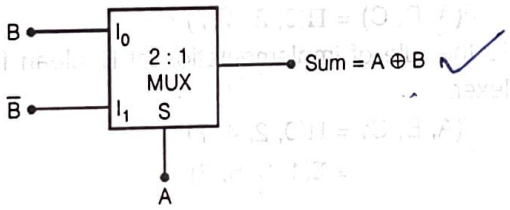
Solution: (b)

We know, H.A. has a SUM = A ⊕ B and a CARRY = AB, when A and B are the two input variables i.e.

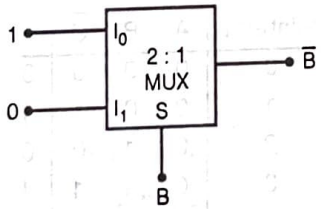
⇒ For **carry**, we require an **AND-gate** as,



⇒ For **SUM**, we require an **EX-OR gate** as,



But we don't available $\bar{B}$ as input, so we require an inverter circuit also as,



Finally, to implement a half-adder we require three 2 : 1 MUX.

[For SUM ⇒ 2 and for CARRY ⇒ 1].

Example 4.7

What are the minimum number of 2-to-1 multiplexers required to generate a 2-input AND gate and a 2-input EX-OR gate?

(a) 1 and 2 (b) 1 and 3

(c) 1 and 1 (d) 2 and 2

[GATE-2009]

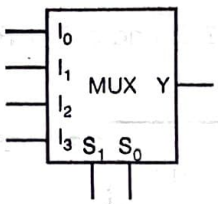
Solution: (a)

Problem should be done by the concept which has been already seen in the example 5.3.

So, for AND gate ⇒ 2 : 1 MUX required = 1 and for EXOR gate ⇒ 2 : 1 MUX required = 2.

Example 4.8

The Boolean function $F(A, B, C) = \Pi(0, 2, 4, 7)$ is to be implemented using a 4×1 multiplexer shown in figure. Which one of the following choices of inputs to multiplexer will realize the Boolean function?



(a) $(I_0, I_1, I_2, I_3, S_1, S_0) = (1, 0, \bar{A}, A, C, B)$

(b) $(I_0, I_1, I_2, I_3, S_1, S_0) = (1, 0, \bar{A}, A, B, C)$

(c) $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, \bar{A}, A, C, B)$

(d) $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, A, \bar{A}, B, C)$

[BSNL (JTO)-2009]

Solution: (d)

Given Boolean function is,

$$F(A, B, C) = \Pi(0, 2, 4, 7)$$

According to the rule of implementation of Boolean function by multiplexer, we required 4 : 1 multiplexer.

also, $F(A, B, C) = \Pi(0, 2, 4, 7)$

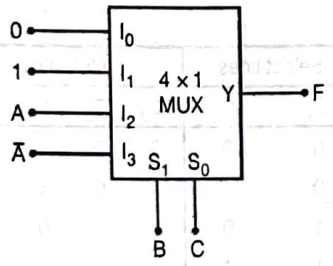
$$\equiv \Sigma(1, 3, 5, 6)$$

Truth table:

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

	I_0	I_1	I_2	I_3 → MUX input connection
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$ → Corresponding MUX input

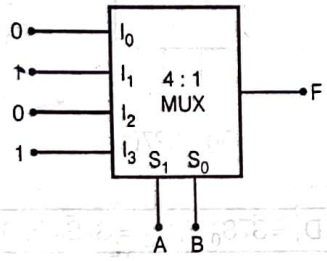
Now, multiplexer implementation is,



So, $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, A, \bar{A}, B, C)$

Example 4.9

Find output of MUX given below:



(a) A (b) $\bar{A}B$

(c) B (d) $A\bar{B} + \bar{A}B$

[IES-2008]

Solution: (c)

$$f = \bar{A}B + AB$$
$$= B(A + \bar{A}) = B$$

Demultiplexer

A "Demultiplexer" is a combinational circuit that receives information on a **single line** and transmits this on one of 2^n **possible output lines.** The selection of a specific output line is controlled by the bit values of "n" selection lines. A "Decoder" with an enable input can function as a "Demultiplexer". It is also known as "Data Distributor". It is used to perform the reverse operation of MUX.

1 : 2 Demultiplexer

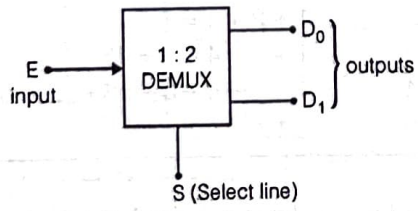


Fig. (4.27)

$$D_0 = \bar{S}E \text{ and } D_1 = SE$$

1 : 4 Demultiplexer

Truth table:

Select lines		Outputs			
S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

Circuit diagram:

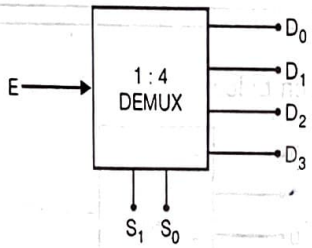


Fig. 4.27(a)

From the truth table we have,

$$D_0 = \bar{S}_1 \bar{S}_0 E, D_1 = \bar{S}_1 S_0 E, D_2 = S_1 \bar{S}_0 E, D_3 = S_1 S_0 E$$

Logic gate circuit:

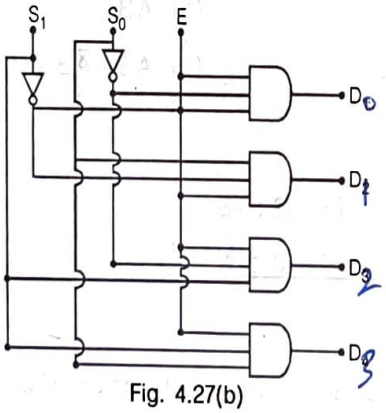


Fig. 4.27(b)

To implement higher order demultiplexer using lower order demultiplexer, we can use the required number of demultiplexers as per following analysis:

Given Demux	To be implemented DEMUX	Required no. of DEMUX
1 : 2	1 : 4	1 + 2 = 3
1 : 2	1 : 8	1 + 2 + 4 = 7
1 : 2	1 : 16	1 + 2 + 4 + 8 = 15
1 : 2	1 : 64	1 + 2 + 4 + 8 + 16 + 32 = 63
1 : 4	1 : 16	1 + 4 = 5

Example 4.10

What is the number of select lines required in a single input and 'n' output DEMUX.

(a) 2 (b) n

(c) $\log_2 n$ (d) 3

[IES-2006]

Solution: (c)

Let number of select lines required = m

We know, $n \approx 2^m$

$\Rightarrow \log_2 n = \log_2 2^m$

$\Rightarrow m \log_2 2 = \log_2 n$

$\Rightarrow m = \log_2 n$

Decoders

A "Decoder" have **many inputs** and **many outputs line.** It is a combinational circuit that converts binary information from **n-input lines** to a maximum 2^n **unique output lines.** If the **n-bit decoded** information has **unused** or **don't care** combinations, the decoder output will have less than 2^n outputs. **i.e. if n = total number of input lines and m = total number of output lines.**

then, $\boxed{m \le 2^n}$

"Decoders" are widely used in memory system of computers.

2 × 4 Decoder

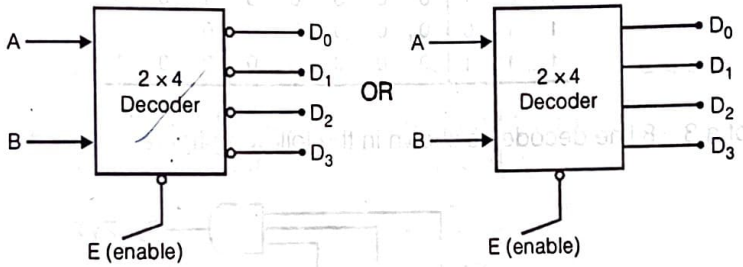


Fig. 4.28(a) Fig. 4.28(b)

For the figure 4.28 (a), we have

Truth table:

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Logic diagram:

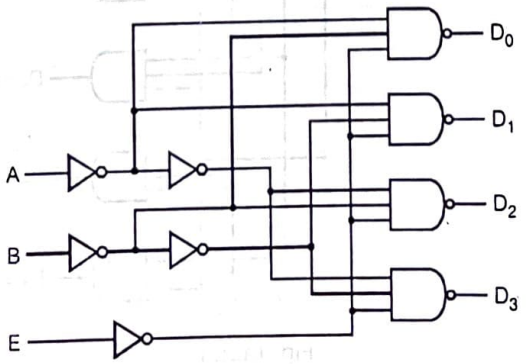


Fig. 4.28 (c)

Note:

⇒ 2 × 4 - decoder may acts like an 1 : 4 demultiplexer and vice-versa.

⇒ Decoder and demultiplexer circuits are almost same.

⇒ Decoder contains AND-gates or NAND-gates.

⇒ A half-adder or half-subtractor circuit can be implemented by the 2 × 4 decoder and an OR-gate

3 × 8 Decoder

Let x, y, z are the three inputs and D_0, D_1, D_2,.....D_7 are eight outputs then truth table of a 3 × 8 lin decoder is as given below:

Input			Output							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Logic diagram of a 3 × 8 line decoder is shown in the following figure.

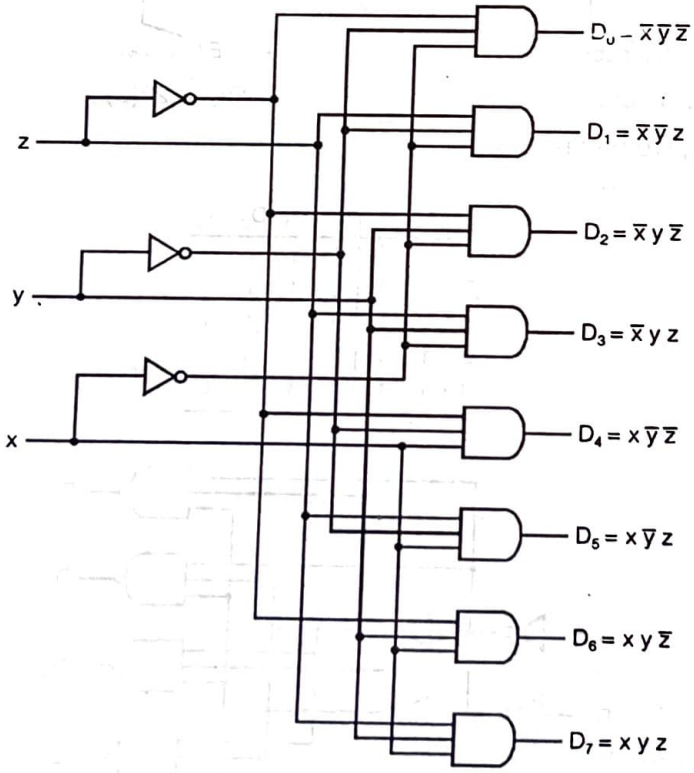


Fig. (4.29)

Note:

⇒ 3×8 line decoder may act like an 1 : 8 demultiplexer and vice-versa.

⇒ To implement a full-adder or full-subtractor we require a 3×8 line decoder and 2-OR-gate.

Since, we know a 3-input (A, B, C) full-adder has SUM = $\Sigma m(1, 2, 4, 7)$ and CARRY = $\Sigma m(3, 5, 6, 7)$, so circuit is as shown below:

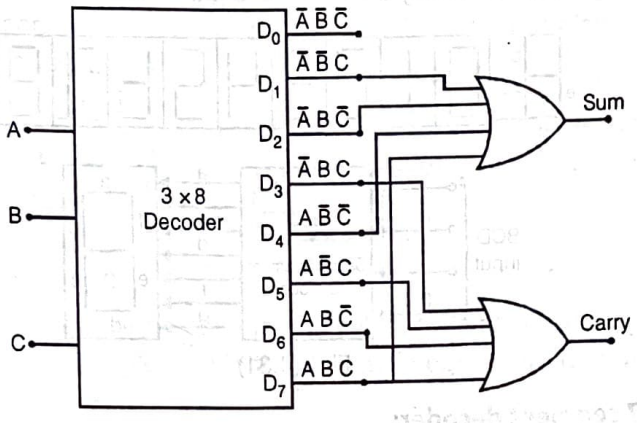


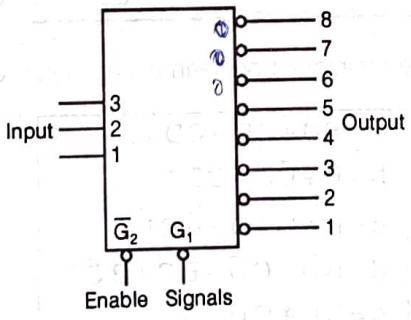
Fig. (4.30)

Following analysis will help us while implementary higher order decoders with lower order decoders:

Given Decoder	To be implemented Decoder	Required number of Decoder
(2×4)	(4×16)	1 + 4 = 5
(2×4)	(3×8)	2 + 1 Not gate
(4×16)	(8×256)	1 + 16 = 17

Example 4.11

A 3-to-8 decoder is shown below:



All the output lines of the chip will be high, when all the inputs 1, 2 and 3

(a) are high; and G_1, G_2 are low

(b) are high; and G_1 is low, G_2 is high

(c) are high; and G_1, G_2 are high

(d) are high; and G_1 is high, G_2 is low

[IES-2002(EE)]

Solution: (b)

BCD-To-7-Segment Decoder/Drivers

One of the most popular and simplest methods for displaying numerical digits uses a "7-segment display" [in figure (4.31)] to form the decimal characters '0' through '9' and some times the hexadecimal characters 'A' through 'F'. It is used in the eletronic calculator, Digital multimeter, watches (Wrist and Railways). It is used to take a 4-bit BCD input and provide the outputs that will pass current through the appropriate segments to display the decimal digit.

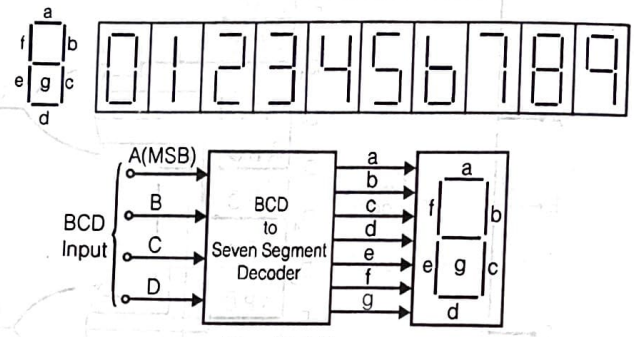


Fig. (4.31)

Truth table of BCD-to-7 segment decoder:

Decimal digit displayed	Input				Output						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

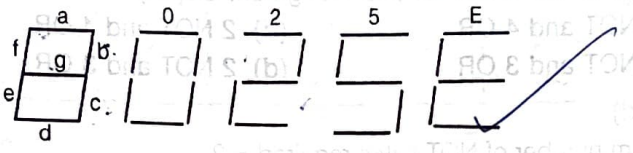
After using K-MAPs, the simplified minimum boolean expression are given by

$$a = \bar{B}\bar{D} + BD + CD + A$$
$$b = \bar{B} + \bar{C}\bar{D} + CD$$
$$c = B + \bar{C} + D = \overline{\bar{B}C\bar{D}}$$
$$d = \bar{B}\bar{D} + C\bar{D} + \bar{B}C + B\bar{C}D$$
$$e = \bar{B}\bar{D} + C\bar{D}$$
$$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$$
$$g = A + B\bar{C} + \bar{B}C + C\bar{D}$$

The outputs of display has AND-OR logic, so it can be implemented by "24" NAND-gates.

Statement for Linked Answer:
Examples 4.12 (a) and 4.12 (b).
Two products are sold from a vending machine, which has two push buttons P_1 and P_2.

When a button is pressed, the price of the corresponding product is displayed in a 7-segment display.
If no buttons are pressed, '0' is displayed, signifying 'Rs. 0'
If only P_1 is pressed, '2' is displayed, signifying 'Rs.2'
If only P_2 is pressed, '5' is displayed, signifying 'Rs.5'
If both P_1 and P_2 are pressed, 'E' is displayed, signifying 'Error'
The names of the segments in the 7-segment display, and the glow of the display for '0', '2', '5' and 'E' are shown below.



[GATE-2009]

Consider:
(i) push button pressed/not pressed is equivalent to logic 1/0 respectively,
(ii) a segment glowing/not glowing in the display is equivalent to logic 1/0 respectively

Example 4.12
(a) If segments a to g are considered as functions of P_1 and P_2, then which of the following is correct?

(a) $g = \bar{P}_1 + P_2, d = c + e$ (b) $g = P_1 + P_2, d = c + e$

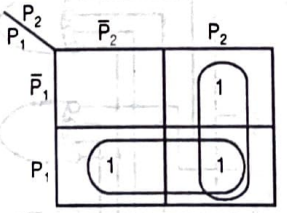
(c) $g = \bar{P}_1 + P_2, e = b + c$ (d) $g = P_1 + P_2, e = b + c$

Solution: (b)

We have two input variables here as P_1 and P_2. Firstly we draw a truth table according to statement of problem.

Inputs		Outputs							
P_1	P_2	a	b	c	d	e	f	g	
0	0	1	1	1	1	1	1	0	→ display '0'
0	1	1	0	1	1	0	1	1	→ display '5'
1	0	1	1	0	1	1	0	1	→ display '2'
1	1	1	0	0	1	1	1	1	→ display 'E'

K-map for 'g':



$\therefore$ $\boxed{g = P_1 + P_2}$

also, we have

$a = 1, b = \bar{P}_2, c = \bar{P}_1, d = 1, e = P_1 + \bar{P}_2, f = \bar{P}_1 + P_2$

since, $\boxed{e \neq b + c}$

but, $d = c + e$

$= P_1 + \bar{P}_2 + \bar{P}_1$

$= 1 + \bar{P}_2 = 1$

so, $\boxed{g = P_1 + P_2 \text{ and } d = c + e}$

(b) What are the minimum numbers of NOT gates and 2-input OR gates required to design the logic of the driver for this 7-segment display?

(a) 3 NOT and 4 OR (b) 2 NOT and 4 OR

(c) 1 NOT and 3 OR (d) 2 NOT and 3 OR

Solution: (d)

Minimum number of NOT-gates required = 2

Minimum number of OR-gates required = 3

ENCODER

The opposite of the **"Decoding"** process is called **"Encoding"** and is performed by a logic circuit called an **"Encoder"**. An **"Encoder"** has 2^n **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. Figure 4.32 (a) shows the general diagram for an **"Encoder"** with **M-inputs** and **N-outputs**. Here the inputs are **active-HIGH**, which means they are normally **LOW**.

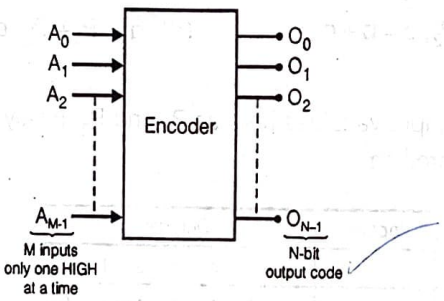


Fig. 4.32(a)

Octal-to-Binary encoder

8-Line-to-3-Line Encoder

Figure 4.32 (b) shows the logic circuit and truth table for **8 × 3 encoder** with **active-low inputs.**

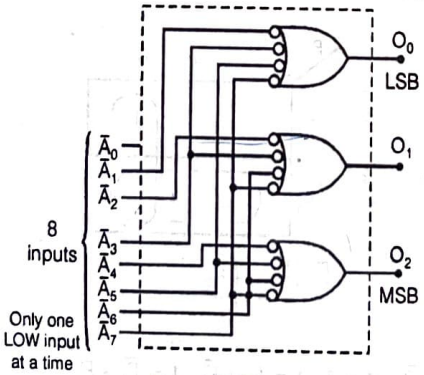


Fig. 4.32(b)

Truth table:

Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

From the truth table we get,

$$O_0 = \bar{A}_1 + \bar{A}_3 + \bar{A}_5 + \bar{A}_7$$

$$O_1 = \bar{A}_2 + \bar{A}_3 + \bar{A}_6 + \bar{A}_7$$

$$O_2 = \bar{A}_4 + \bar{A}_5 + \bar{A}_6 + \bar{A}_7$$

⇒ It contains only 3-OR gates.

Priority Encoders

This is the modified version of simple encoder which eleminates the draw back of encoder (when more than one input is activated at a time). A "Priority Encoder", includes the necessary logic to ensure that when two or more inputs are activated, the output code will correspond to the "highest-number input". For example, (as in truth table when, $\bar{A}_3$ and $\bar{A}_5$ are simultaneously LOW the output code will be 101(5). The 74148, 74LS148 and 74HC148 are all octal to binary priority encoders.

Magnitude Comparators

It is a combinational logic circuit that compares two input binary quantities and generates outputs to indicate which one has the greater magnitude. Consider an **"one-bit comparator"** whose inputs are A and B and the truth table is given as:

Inputs		Outputs		
A	B	A < B	A = B	A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

From truth table have,

⇒ $\boxed{A < B; \text{output} = \bar{A}B}$

⇒ $\boxed{A = B; \text{output} = \bar{A}\bar{B} + AB = A \odot B}$

⇒ $\boxed{A > B; \text{output} = A\bar{B}}$

IEEE/ANSI diagram

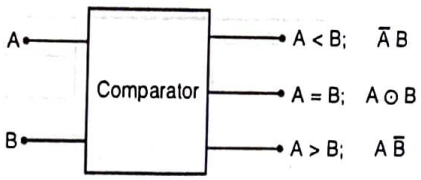


Fig. 4.33(a)

Equivalent logic gate diagram is;

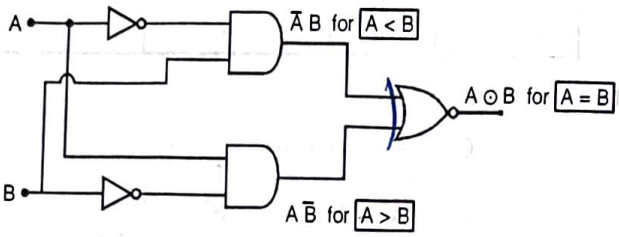


Fig. 4.33(b)

Applications of Magnitude Comparators

These are often used as part of the address decoding circuitry used in a computer to select a specific "input-output device" or area of memory for the storage or retrieval of data. It is also useful in control applications where a binary number representing the physical variable being controlled (e.g. position, speed) is compared with a reference value.

Code Converters

A "Code coverter" is a logic circuit which changes data presented in one type of binary code to another type of binary code. A conversion circuit must be inserted between two systems if each uses different codes for the same information.

BCD to Excess-3 code

Let input variables of BCD code is A, B, C, D and output variables of excess-3 is W, X, Y, Z then required truth table (BCD + 0011 = excess-3-code) is given below:

Inputs (BCD)				Output (Excess-3 code)			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

From the above truth table, the minimised Boolean function is

$$Z = \bar{D}$$
$$Y = CD + \bar{C}\bar{D}$$
$$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$$
$$W = A + BC + BD$$

Logic circuit diagram:

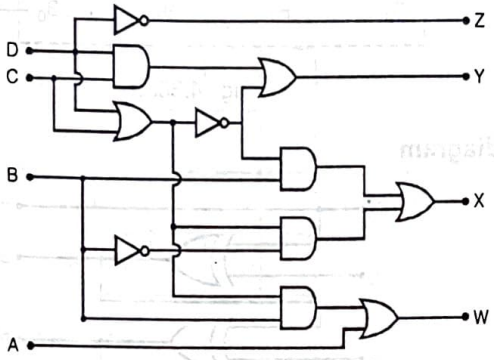


Fig. (4.34)

Binary-to-Gray code converter

The basic conversion concept is shown below:

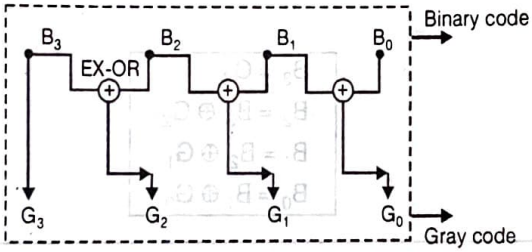


Fig. 4.35(a)

Equivalent logical gate diagram

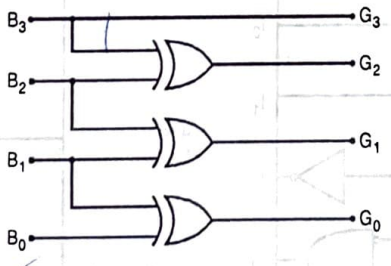


Fig. 4.35(b)

We have,

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0$$

Gray to Binary Converter

The basic conversion concept is shown below:

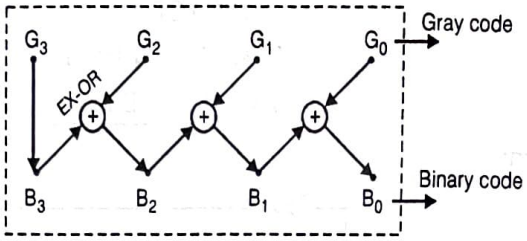


Fig. 4.36(a)

Equivalent logical gate diagram

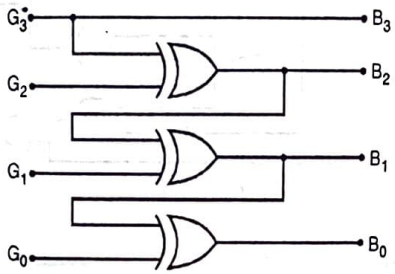


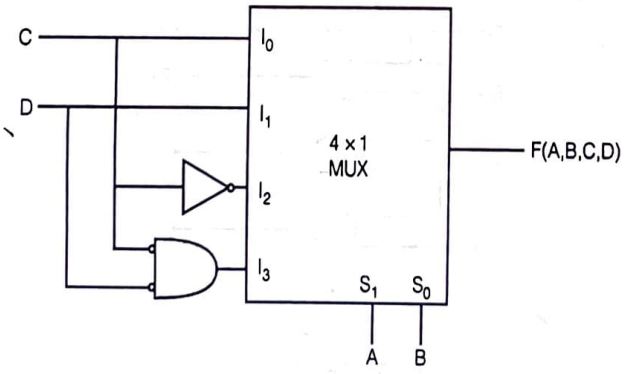
Fig. 4.36(b)

We have,

$$B_3 = G_3$$
$$B_2 = B_3 \oplus G_2$$
$$B_1 = B_2 \oplus G_1$$
$$B_0 = B_1 \oplus G_0$$

Example 4.13

The Boolean function realized by the logic circuit shown is



(a) $F = \Sigma m(0, 1, 3, 5, 9, 10, 14)$ (b) $F = \Sigma m(2, 3, 5, 7, 8, 12, 13)$

(c) $F = \Sigma m(1, 2, 4, 5, 11, 14, 15)$ (d) $F = \Sigma m(2, 3, 5, 7, 8, 9, 12)$

[GATE-2010]

Solution: (d)

$$F(A, B, C, D) = \bar{A}\bar{B}C + \bar{A}BD + A\bar{B}\bar{C} + AB(\bar{C}\bar{D})$$

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

Placing above minterms in Karnaugh map,

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

So, $F = \Sigma m(2, 3, 5, 7, 8, 9, 12)$

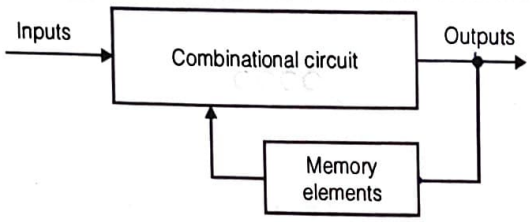
○○○○

SEQUENTIAL CIRCUITS

5

The logic circuits whose outputs at any instant of time depend not only on the present inputs but also on the past outputs are called **sequential circuits.** In sequential circuits, the output signals are fed back to the input side. Thus, an output signal is a function of the present input signals and a sequence of past input signals i.e. the present output signals.

Figure (5.1) shows a block diagram of a sequential circuit. The memory elements are connected to the combinational circuit as a feedback path.



(Block diagram of a sequential circuit)

Fig. (5.1)

The information stored in the memory element at any given time defines the present state of the sequential circuit. The present state and the external inputs determine the outputs and the next state of the sequential circuit. Thus, we can specify the sequential circuit by a time sequence of external inputs, internal states (present state and next state) and outputs. Table (5.1) presents the comparison between combinational and sequential circuits.

Combinational circuits	Sequential circuits
1. In combinational circuits, the output variables at any instant of time are dependent only on the present variables.	1. In sequential circuits, the output variables at any instant of time are dependent not only on the present input variables, but also on the present state, i.e. on the past history of the system.
2. Memory unit is not required in combinational circuits.	2. Memory unit is required to store the past history of the input variables in sequential circuits.
3. Combinational circuits are faster because the delay between the input and the output is due to propagation delay of gates only.	3. Sequential circuits are slower than combinational circuits.
4. Combinational circuits are easy to design.	4. Sequential circuits are comparatively harder to design.

Classification of Sequential Circuits

The sequential circuits may be classified as **synchronous sequential circuits** and **asynchronous sequential circuits** depending on the timing of their signals. The sequential circuits which are controlled by a clock are called synchronous sequential circuits. These circuits will be active only when clock signal is present. The sequential circuits which are not controlled by a clock are called asynchronous sequential circuits, i.e. the sequential circuits in which events can take place any time the inputs are applied are called asynchronous sequential circuits. Table (5.2) shows the comparison between synchronous and asynchronous sequential circuits.

Synchronous sequential circuits	Asynchronous sequential circuits
1. In synchronous circuits, memory elements are clocked FFs.	1. In asynchronous circuits, memory elements are either unclocked FFs, or time delay elements.
2. In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.	2. In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
3. The maximum operating speed of the clock depends on time delays involved.	3. Because of the absence of the clock, asynchronous circuits can operate faster than synchronous circuits.
4. Easier to design.	4. More difficult to design.

To design a sequential circuit, a storage device is required to know what has happened in the past. The basic unit of storage is the **flip-flop**.

Flip-Flops

The memory elements used in "clocked sequential circuits" are called **"flip-flops"**. **It is a basic memory elements**. These are binary cells and can store "one-bit of information". It has two outputs (Q and $\bar{Q}$), which are complemented to each other. It has two stable states which are known as the **"1-state"** and the **"0-state"**. This flip-flop circuit is also known as **"Bistable Multivibrator"** or **"Latch"**. A flip-flop circuit can be constructed from two **NAND-gates** or two **NOR-gates**.

Direct-coupled RS Flip-flop or SR-Latch

It is a cross-coupled **NAND** or **NOR-gates**. The R and S are the first letters of the two input names.

NAND-gate Latch

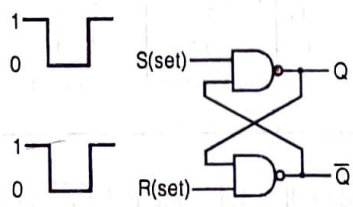


Fig. (5.2)

Truth table:

S	R	Q	$\bar{Q}$	
0	0	1	1	(Invalid)
0	1	1	0	
1	0	0	1	
1	1	0	1	(Previous state)
		1	0	

After the set input returns to **'1'**, a momentary **'0'** to the reset input causes a transition to the clear stat When both inputs go to '0' both outputs go to '1'; a condition avoided in normal flip-flop operation.

NOR-gate Latch

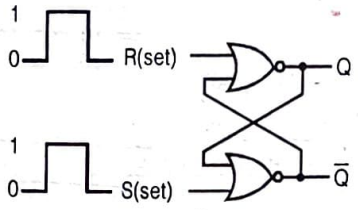


Fig. (5.3)

Truth table:

S	R	Q	$\bar{Q}$	
0	0	0	1	Previous state
		1	0	
0	1	0	1	
1	0	1	0	
1	1	0	0	Invalid state

The **NOR** gate latch operates exactly like the **NAND** latch except that the **Set** and **Reset** inputs are **Active High** rather than **Active Low** and the **normal resisting state** is $\boxed{S = R = 0}$.

Finally we get,

Inputs		Latch Gates	
S	R	NAND	NOR
		Q	Q
0	0	Invalid	No change or Previous
0	1	1	0
1	0	0	1
1	1	No change or Previous	Invalid

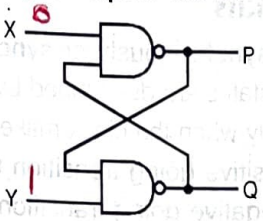
Q will be set **High** by a **High** pulse on the **Set** input and it will be **cleared Low** by a **High** pulse on the **Reset** input in the NOR latch.

Example 5.1

The following binary values were applied to the X and Y inputs of the NAND latch shown in the figure in the sequence indicated below:

X = 0, Y = 1; X = 0, Y = 0; X = 1, Y = 1.

The corresponding stable P, Q outputs will be



(a) P = 1, Q = 0; P = 1, Q = 0; P = 1, Q = 0 or P = 0, Q = 1

(b) P = 1, Q = 0; P = 0, Q = 1 or P = 0, Q = 1; P = 0, Q = 1

(c) P = 1, Q = 0; P = 1, Q = 1; P = 1, Q = 0 or P = 0, Q = 1

(d) P = 1, Q = 0; P = 1, Q = 1; P = 1, Q = 1

[GATE-2007]

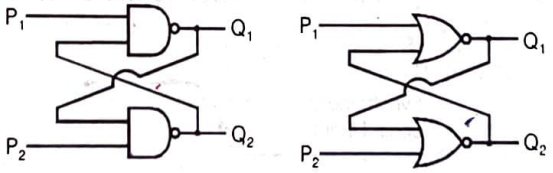
Solution: (c)

As given in the truth table of NAND-gate latch circuit, we get

Inputs			Outputs		
X	Y		P	Q	
0	1	⇒	1	0	
0	0	⇒	1	1	(Invalid)
1	1	⇒	1	0	(Previous)
			0	1	

Example 5.2

Refer to the NAND and NOR latches shown in the figure. The inputs (P_1, P_2) for both the latches are first made (0,1) and then, after a few seconds, made (1,1). The corresponding stable outputs (Q_1, Q_2) are



(a) NAND: first (0,1) then (0,1) NOR: first(1,0) then (0,0)

(b) NAND: first (1,0) then (1,0) NOR: first (1,0) then (1,0)

(c) NAND: first (1,0) then (1,0) NOR: first (1, 0) then (0,0)

(d) NAND: first (1,0) then (1,1) NOR: first (0,1) then (0,1)

[GATE-2009]

Solution: (c)

From the truth tables we have, for NAND-latch,

P_1	P_2		Q_1	Q_2
0	1	→	1	0
1	1	→	Previous	

i.e. $Q_1 = 1$

$Q_2 = 0$

	P_1	P_2	Q_1	Q_2	
For NOR-latch,	0	1	1	0	
	1	1	0	0	(Invalid)

Clocked Flip-flops and Clock Signals

Digital systems can operate either **asynchronously** or **synchronously**. In synchronous system, exact time at which any output can change states are determined by a signal commonly known as the "clock signals". The outputs can change states only when the clock makes a transition (also called edges).

⇒ Clock changes from 0 to 1 → Positive going transition (PGT).

⇒ Clock changes from 1 to 0 → Negative going transition (NGT).

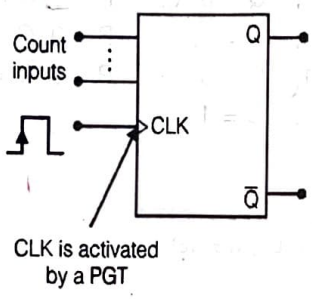


Fig. 5.4(a)

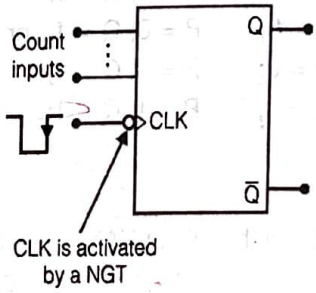


Fig. 5.4(b)

Clocked flip-flops have a clock input i.e. typically labelled CLK, CK or CP. This is indicated by a sm triangle on the CLK input. Ultimately we can say that the **control inputs** get the flip-flop outputs ready change. While the active transition at the CLK **inputs** actually triggers the change i.e. the control inp control the **"What" (what state the output will go to)**; the CLK input determines the **"When"**.

Setup Time (t_s)

It is the time interval immediately preceding the active transition of CLK signal during which t synchronous input has to be maintained at the proper level. IC manufacturers usually specify the minimu setup time, $(t_s)_{min}$.

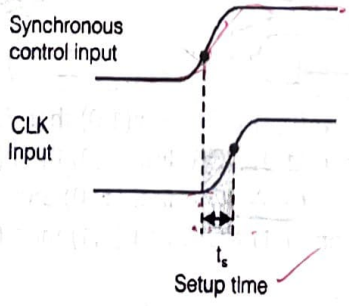


Fig. 5.5(a)

Hold Time (t_H)

It is the time interval immediately following the active transition of CLK signal during which the synchronou input has to be maintained at the proper level. It should be minimum. If this requirement is not met, the flip flop will not trigger reliably.

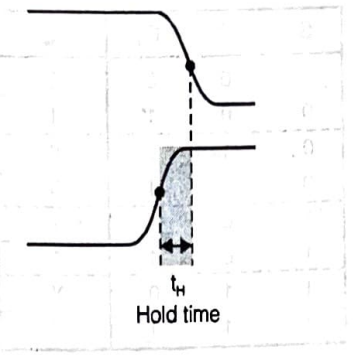


Fig. 5.5(b)

⇒ In IC flip-flops the minimum allowable t_S and t_H values are in the nanosecond range.

Clocked S-R Flip-flop Using NAND Latch

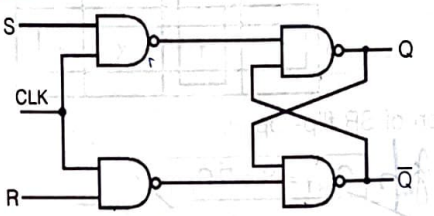


Fig. (5.6)

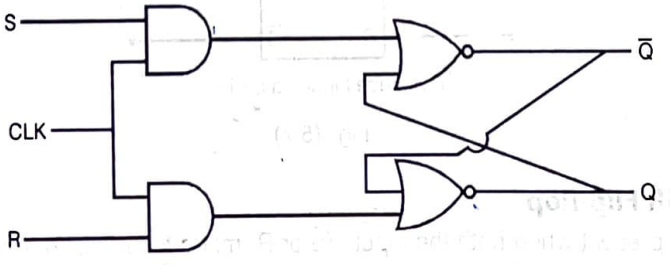
Truth table of S-R FF

Clock	S_n	R_n	Q_{n+1}	
0	X	X	Q_n	
1	0	0	Q_n	→ HOLD state
1	0	1	0	→ RESET state
1	1	0	1	→ SET state
1	1	1	Invalid	→ FORBIDDEN state

⇒ Here S_n and R_n denotes the inputs and Q_n the output during the bit time **'n'**.

⇒ 'Q_{n+1}' denotes the output Q after CLK passes, i.e. in bit time **(n + 1)**.

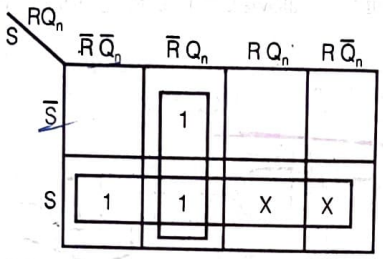
SR Flip-Flop using NOR Latch



Characteristics Table of SR Flip-flop

S	R	Q_n	Q_{n+1}	
0	0	0	0	→ Q_n
0	0	1	1	
0	1	0	0	→ 0
0	1	1	0	
1	0	0	1	→ 1
1	0	1	1	
1	1	0	X	→ Invalid
1	1	1	X	

K-Map



∴ Characteristics equation of SR flip-flop is,

$$Q_{n+1} = S + \bar{R}Q_n \quad \text{...(5.1)}$$

and

$$SR = 0 \quad \text{...(5.2)}$$
$$SR \neq 1$$

The characteristic equation is an algebraic expression for the binary information of the characteristic table. It specifies the value of the next state as a function of the present state and the inputs.

Graphic Symbol

SR flip-flop has 3 inputs – S, R and CP

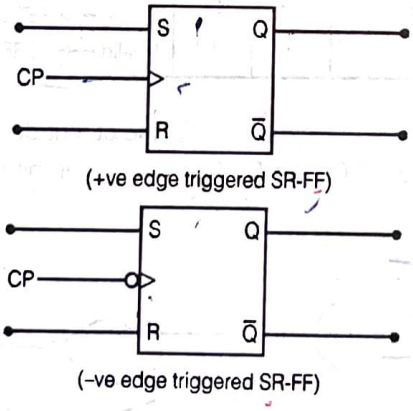


Fig. (5.7)

Disadvantages of SR Flip-flop

Invalid states are present when both the inputs (S or R) made to HIGH. To avoid this difficulty we have to use J-K flip-flop.

J-K Flip-Flop

A JK flip-flop is a refinement of the SR flip-flop in that the indeterminate (or invalid) state of the SR-type is defined in the JK-type. The data inputs are J and K which are ANDed with $\bar{Q}$ and Q respectively, to obtain S and R inputs i.e.,

$$S = J\bar{Q} \text{ and } R = KQ \qquad ...(5.3)$$

This JK flip-flop is called an universal flip-flop because the flip-flops like D flip-flops, SR flip-flop and T flip-flop can be derived from it.

Logic diagram:

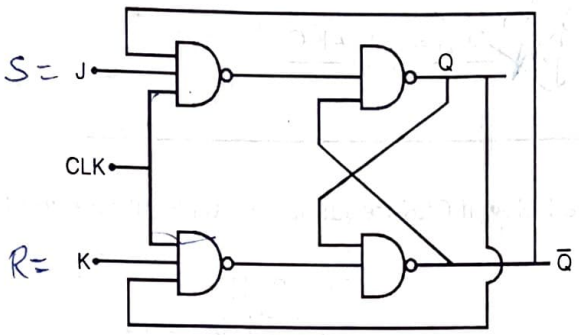


Fig. 5.8(a)

Graphical diagram:

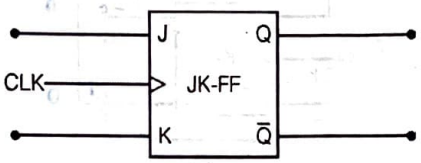


Fig. 5.8(b)

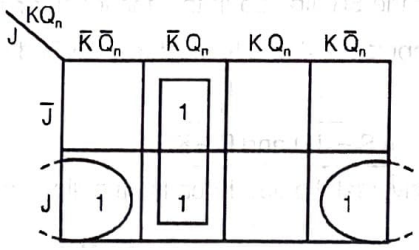
Truth Table of JK flip-flop:

Clock	J	K	Q_{n+1}	
0	X	X	Q_n	
1	0	0	Q_n	→ HOLD state
1	0	1	0	→ RESET state
1	1	0	1	→ SET state
1	1	1	$\bar{Q}_n$	→ TOGGLE state

Characteristic Table

J	K	Q_n	Q_{n+1}	
0	0	0	0	→ Q_n
0	0	1	1	
0	1	0	0	→ 0
0	1	1	0	
1	0	0	1	→ 1
1	0	1	1	
1	1	0	1	→ $\bar{Q}_n$
1	1	1	0	

K-Map



∴ Characteristic equation is,

$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n \qquad \text{...(5.4)}$$

[IES-2007, 2006, 2003]

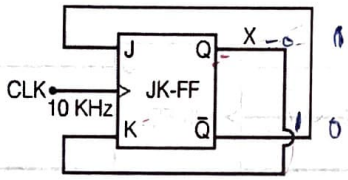
Example 5.3

In the given figure below if CLK frequency is 10 kHz then output frequency at output X is;

(a) 5 kHz (b) 10 kHz

(c) 20 kHz (d) 2.5 kHz

[IES-2003]



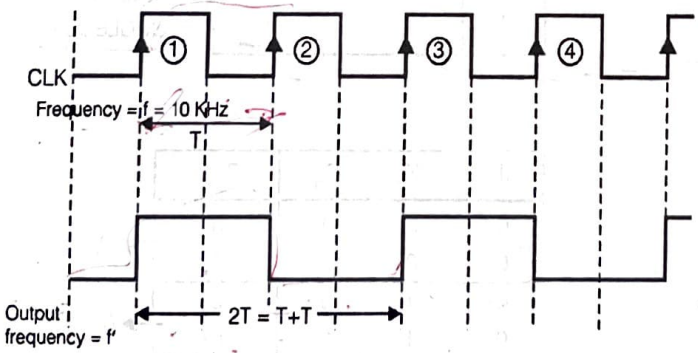
Solution: (a)

Let, initially Q = 0 i.e, X = 0 So, $\bar{Q}$ = 1 then, J = 1 and K = 0. After triggering the first CLK, we get Q = X = 1.

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

after 2nd CLK, we get Q = X = 0

∴ sequence of output Q = 0 1 0 1 0.....



∴ output frequency $f' = \frac{1}{2T} = \frac{f}{2} = 5$ kHz.

Disadvantage of JK flip-flop

Race-Around Condition

The difficulty of both inputs '1' i.e. S = R = 1 being not allowed in S-R flip-flop is eliminated in a JK-flip-flop by using the feedback connections from output to the input. When J = K = 1 and Q = 0, and a CLK as given in figure (5.10) is applied at the CLK input. After a time "$t_{pd(FF)}$" (Propagation delay through two NAND gates in series), the output will change to Q = 1, after another "$t_{pd\ (FF)}$" output will change back to Q = 0. For the duration "t_{pw}" of the CP, the 'Q' will oscillate back and forth between 0 and 1. At the end of the CP, the value of 'Q' is uncertain. This situation is referred to as the "Race-around condition".

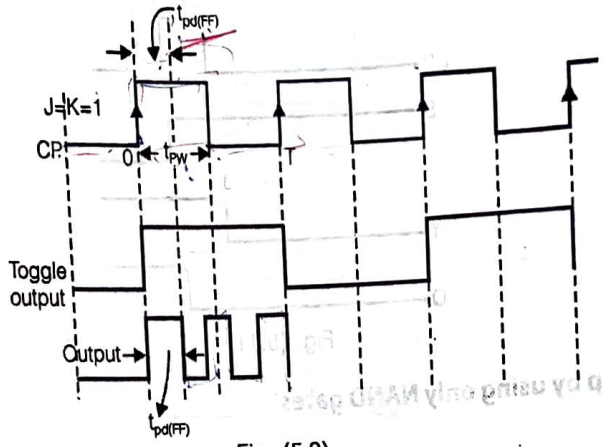


Fig. (5.9)

Note:

⇒ The "Race-around condition" will occur when $J = K = 1$ and $t_{pd(FF)} < t_{pw}$.

⇒ To avoid this, we should maintain, $t_{pw} < t_{pd(FF)} < T$.

⇒ A more practical method for overcoming this "Race-around condition" is the use of the "Master-slave configuration".

Master-Slave Flip-flop

A "Master-slave flip-flop" is basically constructed from 2 flip-flops (a master and a slave) and an 'inverter'. On the rising edge of CLK (i.e. +ve edge CLK pulse) the control inputs are used to determine the output of the master, when the CLK goes low (i.e. –ve edge CLK pulse), the state of master is transferred to the slave, whose outputs are Q and $\overline{Q}$. In the master-slave flip-flop output fully depends upon the output of slave flip-flop.

Logic diagram of Master-slave flip-flop:

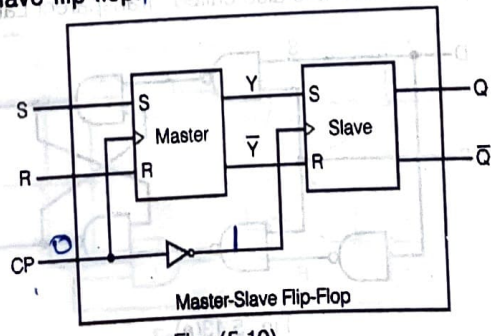


Fig. (5.10)

Operation

When clock pulse CP is 0, the output of the inverter is 1. Since the clock input of the slave is 1, t flop is enabled and output Q is equal to Y, while $\bar{Q}$ is equal to $\bar{Y}$. The master flip-flop is disabled b CP = 0. When the pulse becomes 1, the information then at the external R and S inputs is transmitted master flip-flop. The slave flip-flop, however, is isolated as long as the pulse is at its 1 level, becau output of the inverter is 0. When the pulse returns to 0, the master flip-flop is isolated, which prever external inputs from affecting it. The slave flip-flop then goes to the same state as the master flip-flop.

Timing diagram in Master-slave flip-flop:

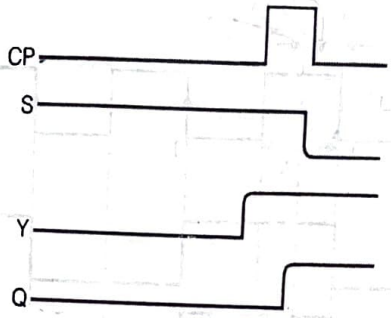


Fig. (5.11)

Clocked-M-S JK flip-flop by using only NAND gates

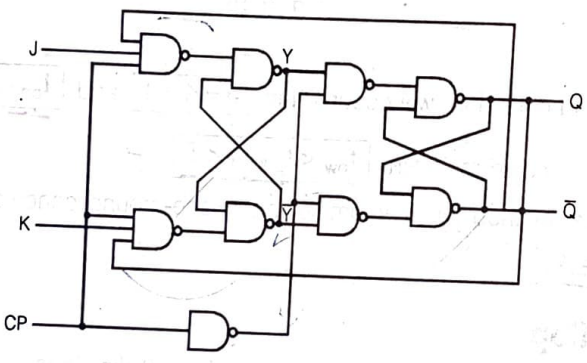


Fig. (5.12)

D-Flip-Flop

The D flip-flop as shown in figure 5.13(a) is a modification of the clocked SR flip-flop. It is a flip-flop with a delay equal to exactly one cycle of CLK. It is also called "Transparent Latch".

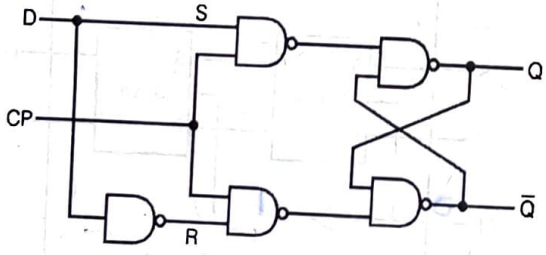


Fig. 5.13(a)

Graphical diagram:

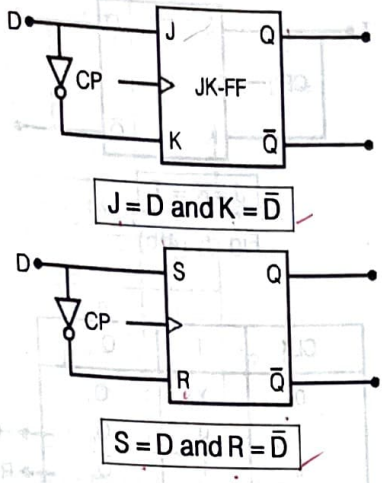


Fig. 5.13(b)

Truth table:

CLK	D	Q_{n+1}
0	X	Q_n
1	0	0
1	1	1

Characteristic table:

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

After using K-map we get, characteristic equation,

$$Q_{n+1} = D\,\bar{Q}_n + D\,Q_n = D(\bar{Q}_n + Q_n) = D$$

$$\therefore \quad \boxed{Q_{n+1} = D} \qquad \ldots(5.5)$$

Toggle Flip-Flop

[IES-2000]

The T flip-flop is a single input version of the JK flip-flop. **T flip-flop can be obtained from a JK flip-flop if J and K are tied together as in figure 5.14(a).** The designation 'T' comes from the ability of the flip-flop to "Toggle" or "Change state".

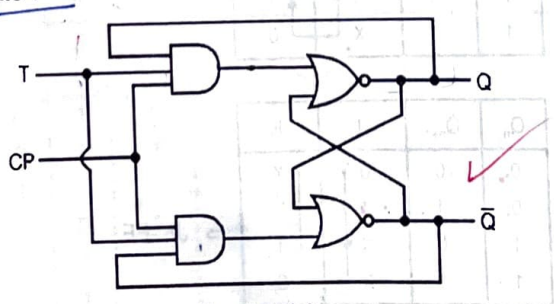
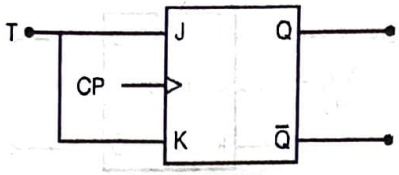


Fig. 5.14(a)

Graphical symbol of T flip-flop:



$J = K = T$

Fig. 5.14(b)

Truth table:

CLK	T	Q_{n+1}	
0	X	Q_n	
1	0	Q_n	→ HOLD state
1	1	$\bar{Q}_n$	→ RESET state

Characteristic table:

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

After using K-map we get, characteristic equation,

$$Q_{n+1} = \bar{T} Q_n + T \bar{Q}_n$$

[BSNL-2009, ISRO-2009, IES-2001]

$$Q_{n+1} = T \oplus Q_n \quad \text{...(5.6)}$$

Flip-Flops Excitation Table

During the design process of flip-flops, we usually know the transition from present state to next state and wish to find the flip-flop input condition that will cause the required transition. For this purpose, we need a table that lists the required inputs for a given change of state. Such a list is called an "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

⇒ SR-FF

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

⇒ JK-FF

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

⇒ D-FF

⇒ It represents the decimal odd number detector.

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

⇒ T-FF

⇒ It represents the odd number of 1's detector.

Example 5.4

The present output Q_n of an edge triggered JK flip-flop is logic 0. If J = 1, then Q_{n+1}

(a) cannot be determined (b) will be logic 0

(c) will be logic 1 (d) will race around [GATE-2005]

Solution: (c)

Given, $Q_n = 0$, J = 1, then, $Q_{n+1} = ?$

From the excitation table of JK flip-flop, we get,

$$\boxed{Q_{n+1} = 1}$$

Conversion of flip-flops

- Firstly write the characteristic table of required flip-flop.
- Now write the excitation table of available or given flip-flop.
- Write excitation equation
- Minimize excitation equation
- Implement logic circuit

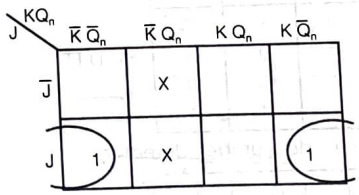
SR flip-flop to JK flip-flops

Here required flip-flop ⇒ JK flip-flop

and given flip-flop ⇒ SR flip-flop

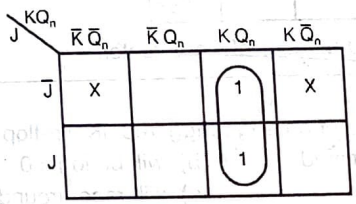
Characteristic Table				Excitation Table	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Now K-Map for S:



$$S = J\bar{Q}_n$$

K-Map for R:



$$R = KQ_n$$

Now final circuit diagram is,

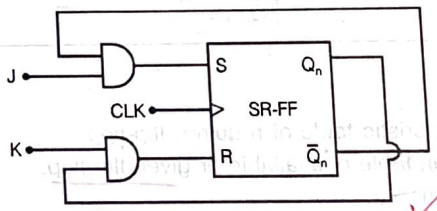


Fig. (5.15)

SR flip-flop to D flip-flop

$$S = D \text{ and } R = \bar{D}$$

SR flip-flop to T flip-flop

$$S = T\bar{Q} \text{ and } R = TQ$$

JK flip-flop to SR flip-flop

$$J = S \text{ and } K = R$$

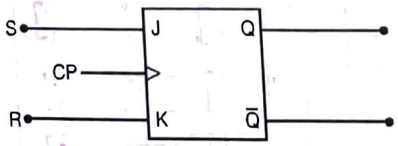


Fig. (5.16)

JK flip-flop to D flip-flop

JK flip-flop to T flip-flop

D flip-flop to other flip-flops

D flip-flop to SR flip-flop

From equation (5.1) and (5.5) we

Q_{n+1}

and Q_{n+}

then,

D flip-flop to JK flip-flop

D flip-flop to T flip-flop

JK flip-flop to D flip-flop

$$\boxed{J = D \text{ and } K = \bar{D}}$$

[IES-2005]

JK flip-flop to T flip-flop

$$\boxed{J = K = T}$$

D flip-flop to other flip-flops

D flip-flop to SR flip-flop

From equation (5.1) and (5.5) we get,

$$Q_{n+1} = D$$

and $$Q_{n+1} = S + \bar{R}Q_n$$

then, $$\boxed{D = S + \bar{R}Q_n}$$

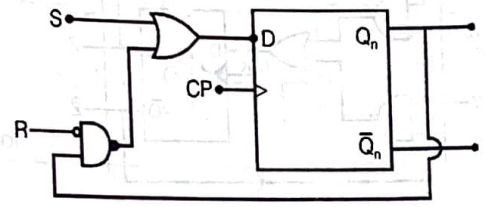


Fig. (5.17)

D flip-flop to JK flip-flop

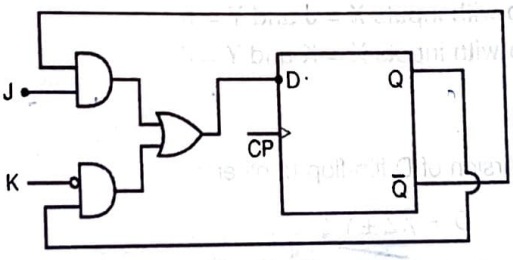
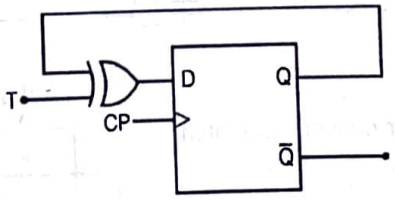


Fig. (5.18)

$$\boxed{D = J\bar{Q} + \bar{K}Q}$$

D flip-flop to T flip-flop



$$\boxed{D = T \oplus Q}$$

Fig. (5.19)

T flip-flop to other flip-flop

T flip-flop to JK flip-flop

$$\boxed{T = J\bar{Q} + KQ}$$

T flip-flop to SR flip-flop

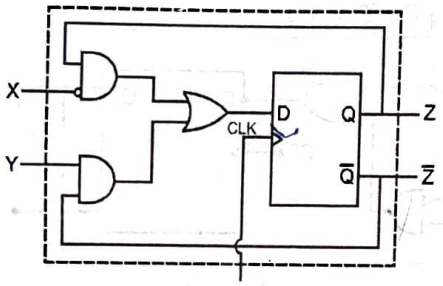
$$\boxed{T = S\bar{Q} + RQ}$$

T flip-flop to D flip-flop

$$\boxed{T = D \oplus Q}$$

Example 5.5

A sequential circuit using D flip-flop and logic gates is shown in the figure, where X and Y are the inputs and Z is the output. The circuit is



(a) S – R flip-flop with inputs X = R and Y = S
(b) S – R flip-flop with inputs X = S and Y = R
(c) J – K flip-flop with inputs X = J and Y = K
(d) J – K flip-flop with inputs X = K and Y = J

[GATE-2000]

Solution: (d)

Here we use conversion of D flip-flop to others:

Here, $D = \bar{X}Z + Y\bar{Z}$ ...(i)

also when D flip-flop to JK flip-flop, we have,

$$D = J\bar{Q} + \bar{K}Q \approx \bar{K}Z + J\bar{Z} \quad ...(ii)$$

From (i) and (ii)

$$\boxed{X = K \text{ and } Y = J}$$

Applications of flip-flops

- Bounce elimination switch or chatterless switch
- Latch
- Registers
- Counters
- Memory etc.

Design of Counters for Random Sequence

A sequential circuit that goes through a prescribed sequence of states upon the application of input pulses is called a **"Counter"**.

Design Procedure

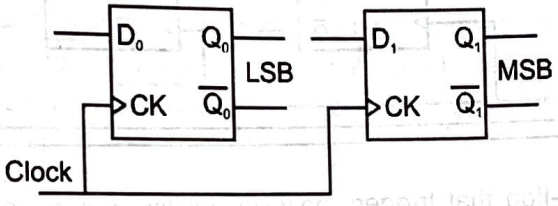
- Write the state table (this table provides the relationship between present state and next state).
- Now write the excitation table with respect to the designing flip-flop(D flip-flop, JK flip-flop etc.).
- Write logical expression with respect to the designing flip-flop and taking the present state.
- Finally implement the circuit according to final Boolean expression.

Example 5.6

Two D-flip-flops, as shown below, are to be connected as a synchronous counter that goes through the following Q_1Q_0 sequence

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

The inputs D_0 and D_1 respectively should be connected as



(a) $\overline{Q_1}$ and Q_0 (b) $\overline{Q_0}$ and Q_1

(c) $\overline{Q_1}Q_0$ and $\overline{Q_1}Q_0$ (d) $\overline{Q_1}\,\overline{Q_0}$ and Q_1Q_0

[GATE-2006]

Solution: (a)

The given random sequence is,

$[Q_1\ Q_0 = 00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00.....]$

(a) Write state table:

Present State		Next State	
Q_1	Q_0	Q_{1+}	Q_{0+}
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0
0	0		

(b) Write excitation table for D flip-flops:

Q_0	Q_{0+}	D_0
0	1	1
1	1	1
1	0	0
0	0	0

Q_1	Q_{1+}	D_1
0	0	0
0	1	1
1	1	1
1	0	0

(c) Logical expression of D flip-flop:

$$D_0 = \bar{Q}_1\bar{Q}_0 + \bar{Q}_1 Q_0$$

$$= \bar{Q}_1(Q_0 + \bar{Q}_0)$$

$$\therefore \quad \boxed{D_0 = \bar{Q}_1}$$

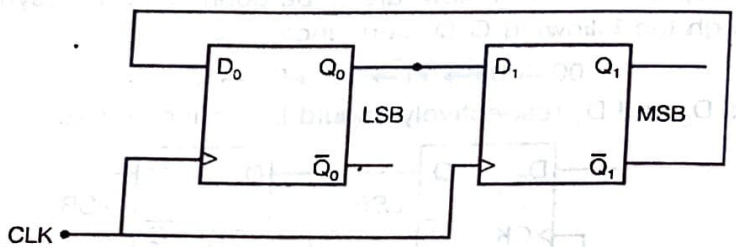
and

$$D_1 = \bar{Q}_1 Q_0 + Q_1 Q_0$$

$$= Q_0(\bar{Q}_1 + Q_1)$$

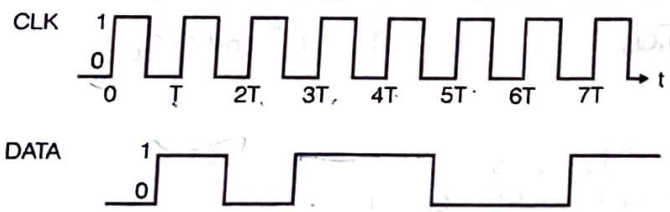
$$\therefore \quad \boxed{D_1 = Q_0}$$

$\Rightarrow$ Final circuit for the example 6.6 is as,



Example 5.7

Consider a D flip-flop that triggers only on positive going transitions. Write its truth table and draw the output at Q for given D-input and clock wave-forms as shown.

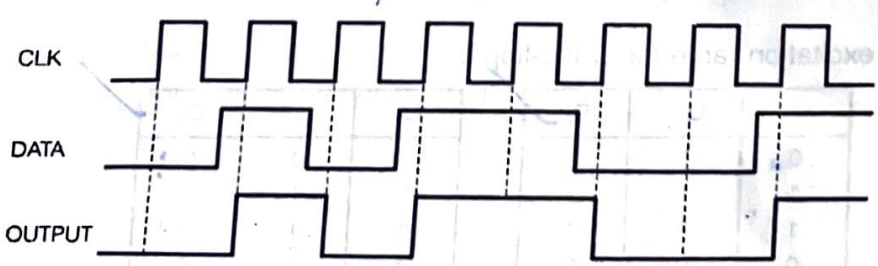


[IES-2005]

Solution:

Truth table of D flip-flop:

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1



Example 5.8

A master-slave flip-flop has the characteristic that

(a) change in the input immediately reflected in the output

(b) change in the output occurs when the state of the master is affected

(c) change in the output occurs when the state of the slave is affected

(d) both the master and the slave states are affected at the same time

[GATE-2004]

Solution: (b)

Example 5.9

What two types of input does a clocked flip-flop have? Explain. What is meant by edge-triggering? Define set-up time and hold-time for a clocked flip-flop.

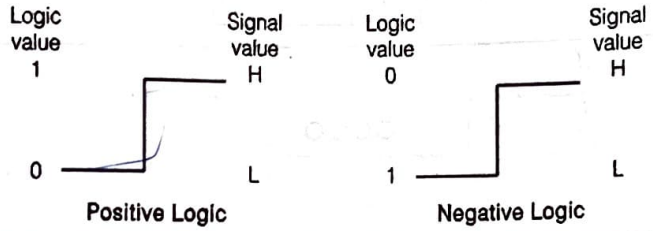
[IES-2006 : 10 marks]

Solution:

A clocked flip-flop has two types of inputs:

(i) **Positive logic system:** The higher level is designated by H and the lower signal level by L. Choosing the high-level H to represent logic-1 defines a positive logic system.

(ii) **Negative logic system:** Choosing the low-level L to represent logic-1 defines a negative logic systems.



Edge triggering:

Edge triggering means that the clock input is activated by a single transition. This is indicated by the presence of a small triangle on the clock input.

Set-up time:

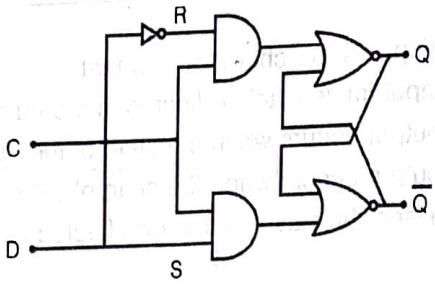
The set-up time (t_S), is the time interval immediately preceding the acting transition of the CLK signal during which the control input must be maintained at the proper level.

Hold time:

The hold time (t_H), is the time interval immediately following the active transition of the CLK signal during which the synchronous control input must be maintained at the proper level.

Example 5.10

A flip-flop is shown in the figure. Explain its operation and construct a truth table to characterize its performance.



[IES-2006 : 10 marks]

Solution:

The given circuit shows the implementation of D flip-flop using RS flip-flop. As long as the pulse input is at 0, the circuit cannot change state regardless of the value of D. The D input is sampled when C = 1. If D = 1, the output Q goes to 1, placing the circuit in the set state. If D is 0, the output Q goes to 0 and the circuit switches to the clear state.

Truth table of D flip-flop:

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

OOOO

SHIFT REGISTERS 6

Registers

A register is composed of a group of flip-flops to store a group of bits (word). For storing an N-bit word, the number of flip-flops required is N (one flip-flop for each bit). Also we can say, a register is a group of binary storage cells suitable for holding binary information. In addition to the flip-flops, a register may have combinational gates that perform certain data processing tasks. In the broadest definition, a register consists of a group of flip-flops and gates that effect their transition. The flip-flops hold binary information and the gates control WHEN and HOW new information is transferred into the register. A group of flip-flops sensitive to the pulse duration is usually called "**gated-latch**", where as a group of flip-flops sensitive to pulse transition is called a "register". In storage registers, mostly D-flip-flops are used. The simplest possible register is one that consists of only flip-flops without any external gates, is shown in figure (6.1).

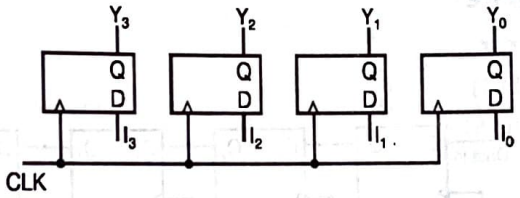


Fig. (6.1)

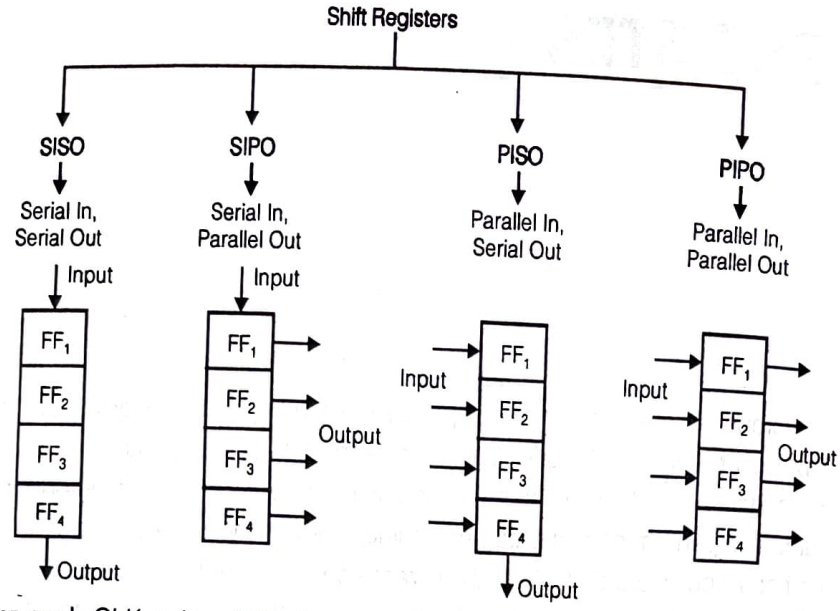
The data can be entered in serial (one-bit at a time) or in parallel form (all the bits simultaneously) and can be retrieved in the serial or parallel form.

- Data in serial form ⇒ Temporal code.
- Data in parallel form ⇒ Spacial code.

Shift Registers

A register capable of shifting its binary information either to the right (right shift register) or to the Left (Left-shift register) is called the "Shift register". The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of next flip-flop. All flip-flops receive a common CLK pulse which cause the shift from one state to the next.

Classification of shift registers depending upon the way in which data are entered retrieved



In shift register each CLK pulse shifts the contents of register one-bit position to the right or left. T "serial input" determines what goes into the leftmost flip-flop during the shift. The "serial output" is taken fr the output of the right most flip-flop prior to the application of a pulse.

Serial-in Serial-out Register

4-bit right-shift SISO register

Let we have information **1101**.

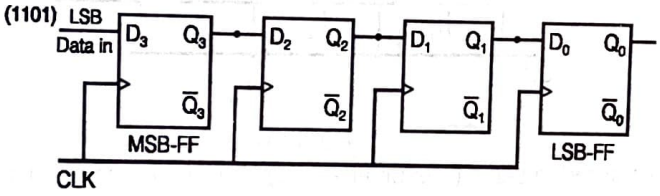
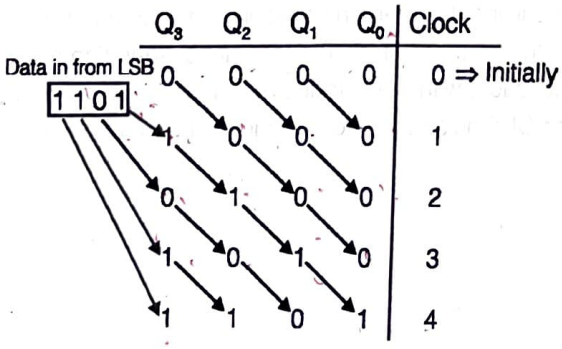


Fig. 6.2(a)

In right shift SISO register, LSB data is applied at the MSB flip-flop'(D-flip-flop). In 'n' bit register, t enter 'n' bit data, it requires 'n' clock pulses in serial form. If 'n' bit data is stored in SISO register then outpu is taken serially; for this it requires (n – 1) clock pulse. SISO register is used to provide 'n' clock pulse delay to the input data. If 'T' is the time period of clock pulse, then delay provided by SISO is nT.



⇒ From truth table it is clear that the data 1101 stores from LSB in the right shift way after 4 clock pulses.

4-bit left-shift SISO register

Let we have information 1101.

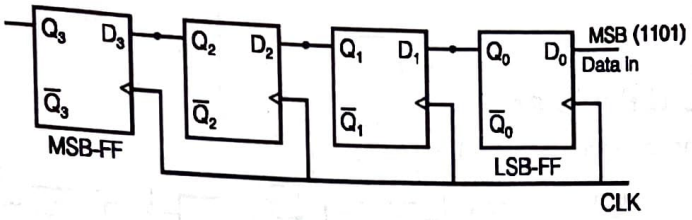
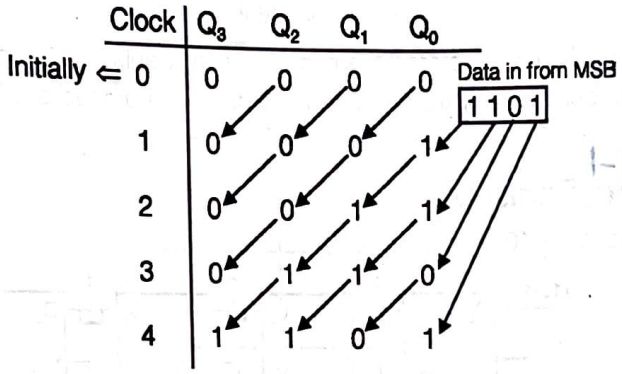


Fig. 6.2(b)

In this above SISO register MSB data is applied to the LSB flip-flop(D-flip-flop). To enter the 'n' bit data in serial form we require 'n' clock pulse. To exit or getting output of 'n' bit data as serially we require (n – 1) clock pulse.



⇒ From truth table it is clear that the data 1101 stores from MSB in the left shift way after 4 clock pulses.

Bi-Directional Shift Register

Such register which are capable of shifting the information (data) to right and left both is called "Bi-directional shift register".

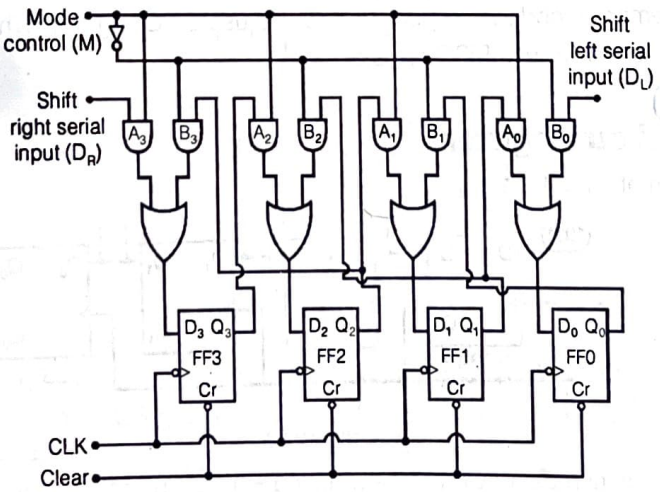


Fig. (6.3)

Operation

When the Mode control M = 1, all the 'A' AND gates are enabled and the data at D_R is shifted to the right when clock pulses are applied. When M = 0, the A gates are inhibited and B gates are enabled allowing the data at D_L to be shifted to the left. M should be changed only when CLK = 0, otherwise the data stored in the register may be altered.

Serial-in Parallel-out Register

Let we have information (group of bits) = 1101.

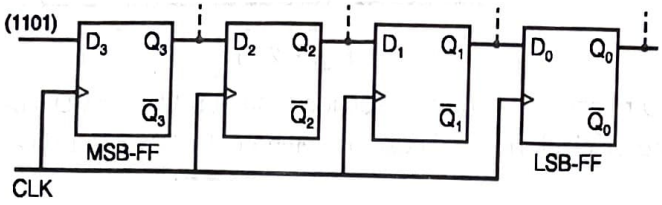


Fig. (6.4)

For n bit- serial input data to be stored the number of CLK pulse required = n. For n bit-parallel output data to be stored the number of CLK pulse required = 0 (there is no need of CLK pulse).

Parallel-in Serial-out Register

Let we have information = 1011.

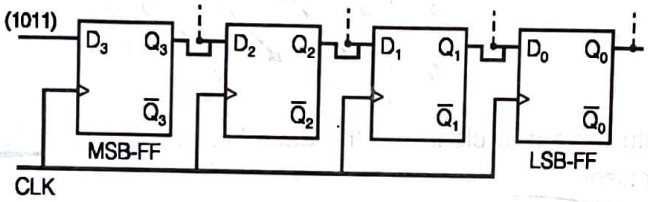


Fig. (6.5)

To store parallel in data, if we store n bit then the number of CLK pulse required = 1 CLK pulse. To store serial out data if we store n bit then the number of CLK pulse required = (n – 1).

Note:

To convert temporal code into spacial code, we use SIPO register. While to convert spacial code into temporal code we use PISO register.

Parallel-in Parallel-out Register

Let we have information = 1011.

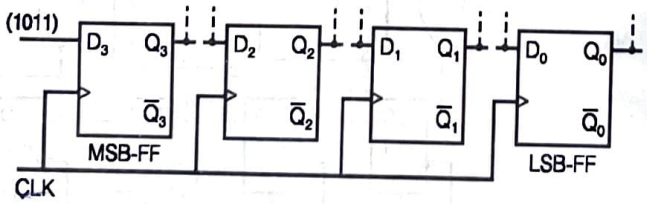
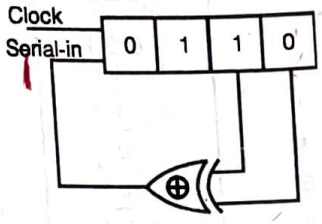


Fig. (6.6)

For parallel in data the number of CLK pulse required = 1 CLK pulse. For parallel out data the number of CLK pulse required = 0 CLK pulse.

Example 6.1

Consider the following shift rights register:



The initial contents of the 4-bit serial-in parallel-out, shift right register shown above are 0110. What will be the contents of the register after 3-clock pulses are applied?

(a) 0000 (b) 0101
(c) 1010 (d) 1111

[IES-2001, 2003, 2004, BSNL(JTO)-2002]

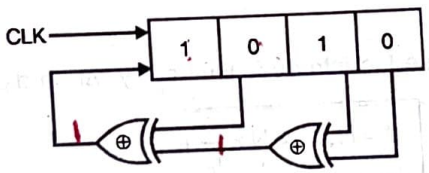
Solution: (c)

Storage bit = 0110 when CLK = 0 since it is Right-shift register and we know, output of Ex-OR gate is HIGH (1) only when both inputs are different.

CLK	DATA
Initially	0 1 1 0
After first CLK	1 0 1 1
After second CLK	0 1 0 1
After Third CLK	1 0 1 0

Example 6.2

The shift right register shown below initially loaded with 1010. If CLK are applied continously then after how many CLK pulse the contents of the shift register becomes 1010 again.



(a) 3 (b) 7
(c) 15 (d) 11

[GATE-2003, IES-2006]

Solution: (b)

We know that, Ex-OR gate with 3-inputs acts as the odd number of 1's detector.

Final output of both Ex-OR gate depends on the given 3-input from LSB.

No. of CLK	Data
Initially = 0	1 0 1 0
1	1 1 0 1
2	0 1 1 0
3	0 0 1 1
4	0 0 0 1
5	1 0 0 0
6	0 1 0 0
7	1 0 1 1

same data

⇒ After 7 CLK pulse, we get the same data.

Difference between Serial and Parallel Transfer

S.No.	Parallel Transfer	Serial Transfer
1.	During single pulse, all the data is transferred simultaneously.	For complete transfer of n-bits of data it requires n-CLK pulses.
2.	It is much faster.	It is relatively slower.
3.	It requires more inter connections between sending register and the receiving register.	It is relatively less inter-connections.
4.	The circuit is more complex.	Circuit is relatively simple

Applications of Shift Registers

The primary uses of shift registers are temporary data storage and bit manipulations. Despite of this, other common applications are given below:

Time-Delay

A SISO shift registers may be used to introduce time delay "Δt" in digital signals given by:

$$\Delta t = N \times T = N \times \frac{1}{f_c}$$

where,

N = Number of flip-flops

T = Time period of CLK pulse

f_C = CLK frequency

The amount of delay can be controlled by the "f_c" or number of flip-flops in the shift register.

Data Conversion

- Data in the serial form can be converted into parallel form by using SIPO-shift register.
- Data in the parallel form can be converted into serial form by using PISO-shift register.

Ring Counter

If the output of the shift register is connnected back to input then an injected pulse will keep circulating, this circuit is referred to as the "Ring counter".

Sequence Generator

A circuit which generates a prescribed sequence of bits, in synchronism with a CLK, is referred to as a "sequence generator".

Arithmetic Operation

Shift registers are used to perform various arithmetic operations e.g. "serial adder" adds two binary numbers.

Example 6.3

Consider the following statements regarding registers and latches:

1. Registers are made of edge-triggered flip-flop$_S$, whereas latches are made from level-triggered flip-flop$_S$.
2. Registers are temporary storage devices whereas latches are not.
3. A latch employs cross-coupled feedback connection.
4. A register stores a binary word whereas a latch does not.

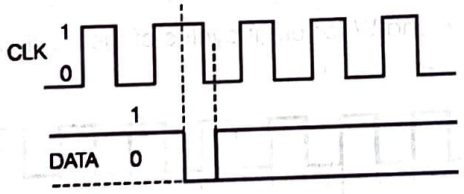
Which of the statements given above are correct?

(a) 1 and 2 (b) 1 and 3
(c) 2 and 3 (d) 3 and 4 [IES-2004]

Solution: (b)

Example 6.4

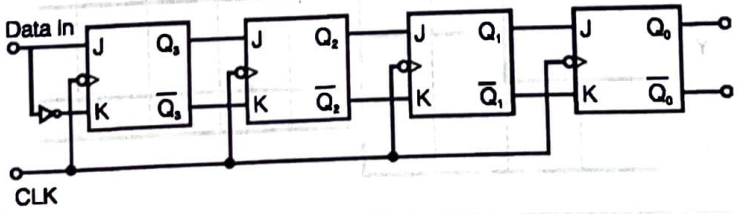
Draw the diagram of a 4-bit shift register using JK flip-flops. Each flip-flop triggers on the negative going transition. Draw the output waveforms for all flip-flops when the input data and clock signals are as shown.



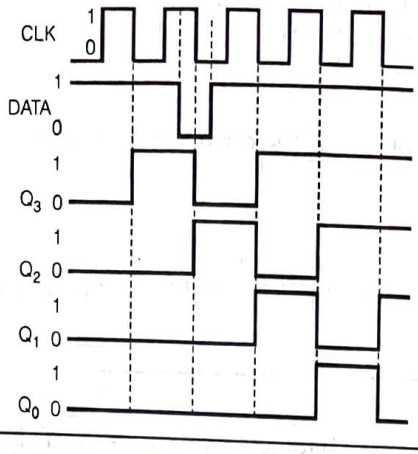
[IES-2003]

Solution:

The diagram of a 4-bit shift register using JK flip-flops is shown below:

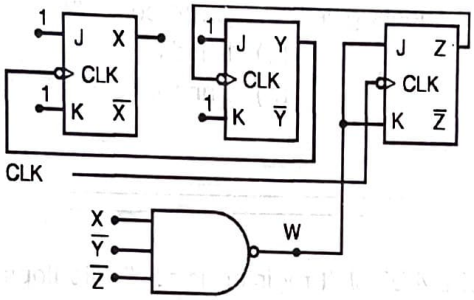


The output waveforms of all flip-flops as for the given data are shown below:



Example 6.5

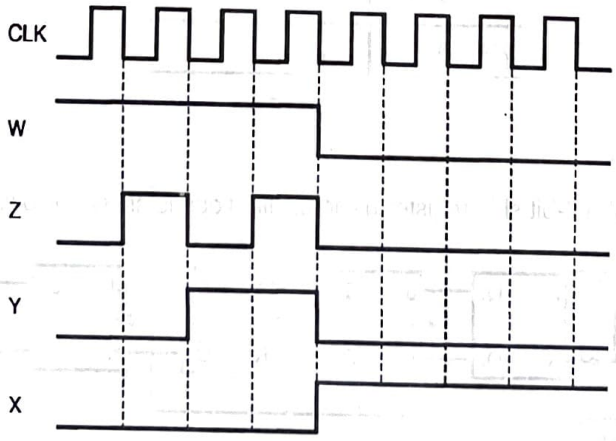
Consider the circuit shown below. Initially all the flip-flop outputs X, Y and Z are in the 0 state before the clock pulses are applied. Determine and sketch the waveforms at Z, Y, X and W for eight cycles of the clock input.



[IES-2005]

Solution:

The waveforms at Z, Y, X and W for eight cycles of the clock input are shown below:



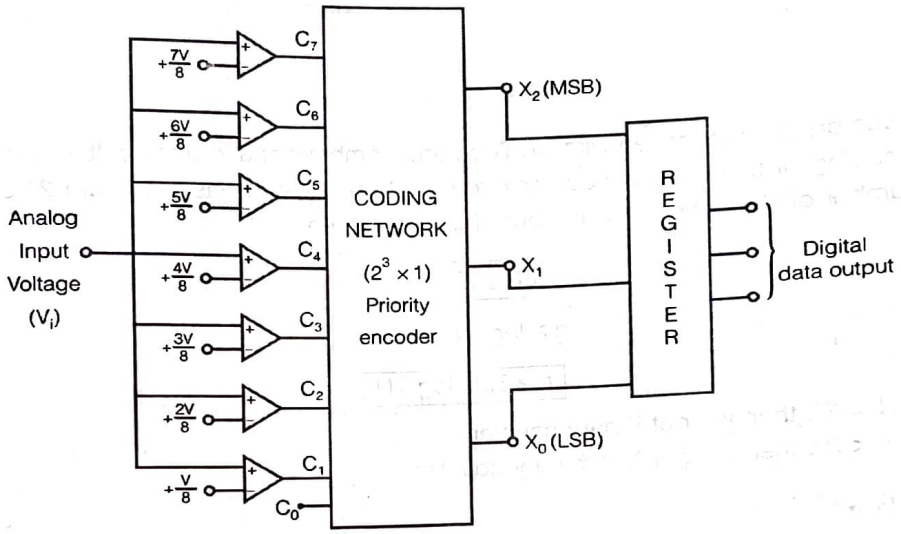
Example 6.6

Describe the operation of a 3-bit flash type ADC with the help of a suitable diagram and a table.

[IES-2003 : 10 marks]

Solution:

3-bit flash type ADC: The block diagram of 3-bit flash type ADC is shown below:



$\Rightarrow$ When, $V_{in} > V_{ref}$ then comparator output is high.

The comparator output and 3-bit digital output for each interval is given in the table below:

Analog Input	Comparator Outputs							Digital Output		
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	X_2	X_1	X_0
$0 < V_{in} < V/8$	0	0	0	0	0	0	0	0	0	0
$V/8 < V_{in} < 2V/8$	0	0	0	0	0	0	1	0	0	1
$2V/8 < V_{in} < 3V/8$	0	0	0	0	0	1	1	0	1	0
$3V/8 < V_{in} < 4V/8$	0	0	0	0	1	1	1	0	1	1
$4V/8 < V_{in} < 5V/8$	0	0	0	1	1	1	1	1	0	0
$5V/8 < V_{in} < 6V/8$	0	0	1	1	1	1	1	1	0	1
$6V/8 < V_{in} < 7V/8$	0	1	1	1	1	1	1	1	1	0
$7V/8 < V_{in} < V$	1	1	1	1	1	1	1	1	1	1

oooo

COUNTER

7

The counters are composed with flip-flops and combinational elements. It is a sequential circuit forming by the cascading of flip-flops. A counter circuit with n-flip-flops has maximum 2^n possible states. Also, if N = total number of states and n = number of flip-flops then,

$$\boxed{N \le 2^n} \quad \text{...(7.1)}$$

$$\Rightarrow \quad n \ge \log_2 N$$

$$\Rightarrow \quad \boxed{n \ge 3.32 \log_{10} N} \quad \text{...(7.2)}$$

- If $N = 2^n$, then we get Binary counter.
- If $N < 2^n$, then we get Non-binary counter.

MOD Number

The "MOD-number" indicates the number of states in counting sequence. For n-flip-flops, counter will have 2^n different states and then this counter is said to be "MOD-2^n counter". MOD number indicates the frequency division obtained from the last flip-flop. **It would be capable of counting upto $(2^n - 1)$ before returning to zero state.**

Based upon the application of CLK pulse, counters are of two types:

1. Asynchronous counter (Ripple counter). or Serial Counter
2. Synchronous counter. Or Parallel Counter

Synchronous Counter	Asynchronous Counter
1. All flip-flops are triggered with different CLK.	1. Different flip-flops are run with same CLK.
2. Operation is faster.	2. Operation is slower.
3. Any required sequence can be designed.	3. Only fixed sequence can be designed.
4. **No decoding error occurs.**	4. Decoding error occurs due to "t_{pd}".
5. Its design is complex.	5. Its design is relatively easy.

UP/Down Counter

If a counter counts in such a way that the decimal equivalent of output increases with successive CLK pulses, is called as "UP counter" and if the decimal equivalent of the output decreases with successive CLK pulses then it is called "Down counter" while an "UP/Down counter" can count in any direction depending upon the control input.

Application of Counters

- To count the number of CLK pulses.
- To count the number of items in industry.
- As a "Frequency divider".
- In time measurement.
- For distance measurement in Radar system.
- In Analog to Digital converter (ADC).
- In measurement of PRI (Pulse Repetition Interference).

Note:

⇒ In "MOD-N counter", if applied input frequency is "f", then output frequency is f/N.

⇒ If two counters are cascaded with MOD-M followed by MOD-N, then number of overall states of combined counter is (M × N) and counter is called "MOD-MN" counter.

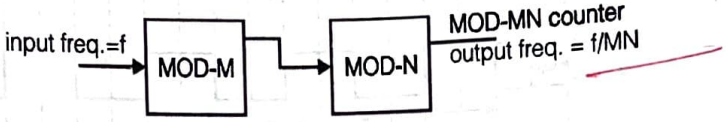


Fig. (7.1)

Asynchronous Counter (or Ripple Counter)

Here we have discussed the "Binary counter" and also used Toggled mode flip-flops (i.e. J-K or T-flip-flop). Figure 7.2(a) shows the 3-bit binary ripple counter which consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the CP input of the next higher-order flip-flop. The flip-flop holding the LSB receives the incoming count pulses.

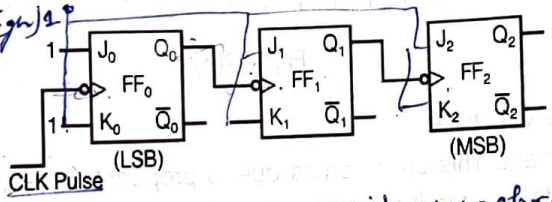


Fig. 7.2(a) 3bit asynchronous Counter or MOD-8 Counter.

- Q_0 will change its state in every CLK pulses.
- Q_1 will changes its state when Q_0 will changes from 1 to 0.
- Q_2 will changes its state when $\overline{Q}_1$ will changes from 1 to 0.

Truth table:

CLK	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 ⇒	0	0	0

A single F.F is Mod 2 Counter.

Modulus Counter → The no of states through which the counter passes before returning to the starting state or initial state is called modulus counter. Since a 2 bit counter has 4 states, →

- Initially all flip-flops are set to zero.
- Maximum possible states = 8 (from 0 to 7).
- Fixed sequence follows, so it is an "UP Counter".
- If input frequency is 'f' then here output frequency = f/8.
- In ripple counter with n-flip-flops there are 2^n possible states.
- With n-flip-flops the maximum count that can be counted by this counter is $2^n - 1$.
- It is also called (2^n : 1) scalar counter.

Timing Diagram

We use (–ve) edge triggering.

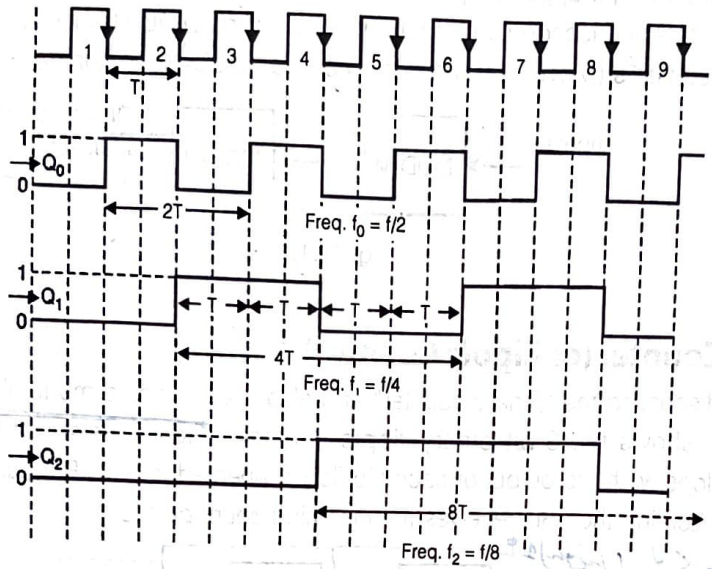


Fig. 7.2(b)

Disadvantage of Ripple Counter

Decoding error is present. This error occurs due to propagation delay of flip-flops i.e. $t_{pd(flip\text{-}flop)}$. For proper operation of the ripple counter; it should be noted that,

$$T_{CLK} \geq n t_{pd(FF)} \quad \ldots(7.3)$$

$$\therefore \quad f_{CLK} \leq \frac{1}{n t_{pd(FF)}} \quad \ldots(7.4)$$

$$\Rightarrow \text{Maximum CLK frequency} = \frac{1}{n t_{pd(FF)}} \quad \ldots(7.5)$$

To overcome the decoding error in ripple counter, we may use "strobe Input".

For determination of UP/Down Counter

Triggering with	CLK connection in	Access as
(–ve) edge	Q	UP Counter
(–ve) edge	$\bar{Q}$	Down Counter
(+ve) edge	Q	Down Counter
(+ve) edge	$\bar{Q}$	UP Counter

Non-Binary Ripple Counter

Decade Counter or Mod-10 Counter

Decade counter require four T-flip-flops. As we know that with four flip-flops, there will be 16-states in total; so here used states are equal to 10 while remaining 6-states are unused. Truth table for decade counter is given below:

CLK	Q_3	Q_2	Q_1	Q_0
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
	⇓	⇓	⇓	⇓
	0	0	0	0
	$\Rightarrow Q_3$	$\overline{Q}_2$	Q_1	$\overline{Q}_0$

⇒ After 10 CLK same count to be obtained, so it is called "MOD-10 Counter".

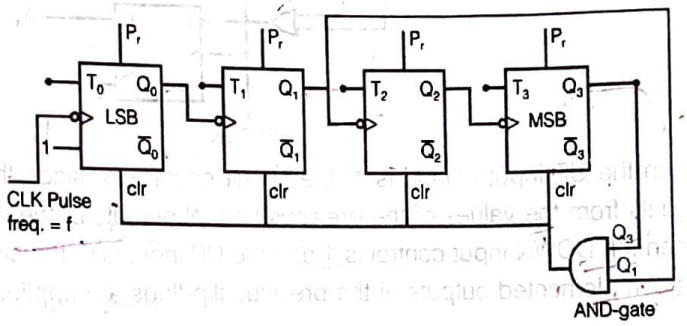


Fig. (7.3)

- Output frequency of MOD-10 counter = f/10.
- **If there is no feedback present at Q_3** then output frequency = f/16.
- When Decade counter counts from 0 to 9 the it is known as BCD Counter.

Conclusions

For making Non-binary counter, if "clr" is present and CLK connected with output Q, then we use AND-gate. (as seen in fig. 7.3) similarly we can say,

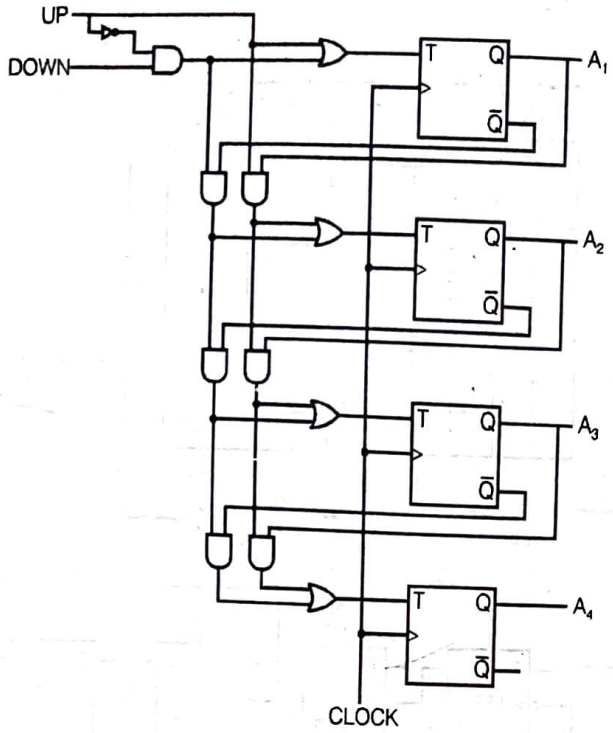
$\text{clr} \Rightarrow Q \Rightarrow$ AND – gate
$\text{clr} \Rightarrow \overline{Q} \Rightarrow$ NOR – gate
$\overline{\text{clr}} \Rightarrow Q \Rightarrow$ NAND – gate
$\overline{\text{clr}} \Rightarrow \overline{Q} \Rightarrow$ OR – gate

Example 7.1

Design a 4-bit binary UP/DOWN ripple counter with a control for UP/DOWN counting. [IES-2005]

Solution:

The 4-bit binary UP/DOWN ripple counter is shown below:



⇒ When the UP input control is 1, the circuit counts up, since the T inputs receive their signals from the values of the previous normal outputs of the flip-flops.

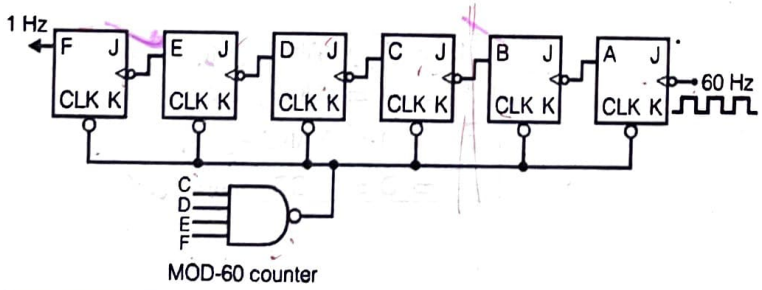
⇒ When the DOWN input control is 1 and the UP input is 0, the circuit counts down, since the complemented outputs of the previous flip-flops are applied to the T inputs.

Example 7.2

MOD-60 counter was needed to divide the 60-Hz line frequency down to 1 Hz. Construct an appropriate MOD-60 counter.

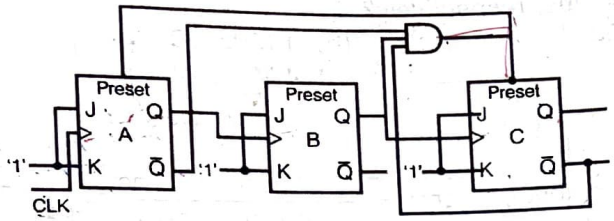
Solution:

$2^5 = 32$ and $2^6 = 64$, and so we need six flip-flops. The counter is to be cleared when it reaches the count of sixty (111100). Thus, the outputs of flip-flops C, D, E and F must be connected to the NAND gate. The output of flip-flop F will have a frequency of 1 Hz.



Example 7.3

The ripple counter shown in the figure works as a



(a) mod-3 up counter
(b) mod-5 up counter
(c) mod-3 down counter
(d) mod-5 down counter

[GATE-1999]

Solution: (d)

From the given circuit, it is +ve edge triggered and Q as CLK, so it is a **down counter.**

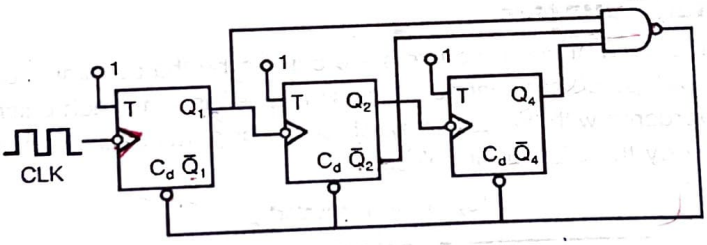
Note: When preset is connected as the output of AND gate, then mod-number is counted in reverse sequence.

Initially ⇒	7	1	1	1	(Preset)
	6	1	1	0	2
	5	1	0	1	3
	4	1	0	0	4
	3	0	1	1	5
Finally ⇒	2	0	1	0	

MOD No. = 5

Example 7.4

The circuit given below is that of a



(a) Mod-5 counter
(b) Mod-6 counter
(c) Mod-7 counter
(d) Mod-8 counter

[IES-2005]

Solution: (a)

Output sequence = $Q_4\ Q_2\ Q_1$

1 0 1 = 5

It is a Mod-5 counter.

⇒ It is also an **UP Counter.**

Example 7.5

12 MHz clock frequency is applied to a cascaded counter of modulus-3 counter, modulus-4 counter and modulus-5 counter. What are the lowest output frequency and the overall modulus, respectively?

(a) 200 kHz, 60 (b) 1 MHz, 60
(c) 3 MHz, 12 (d) 4 MHz, 12

[IES-2005]

Solution: (a)

$$\text{Lowest output frequency} = \frac{\text{input clock frequency}}{\text{Modulus number}}$$

also, mod. number $= 3 \times 4 \times 5 = 60$

and $$f_0 = \frac{12\,\text{MHz}}{60} = \frac{1}{5} \times 10^6 = 200 \times 10^3\,\text{Hz}$$

$$f_0 = 200\,\text{kHz}$$

Example 7.6

A 4-bit modulo-16 ripple counter uses J-K flip-flop. If the propagation delay of each flip-flop is 50 nanoseconds, the maximum clock frequency that can be used is equal to

(a) 20 MHz (b) 10 MHz
(c) 8 MHz (d) 5 MHz

[IES-2003(EE)]

Solution: (d)

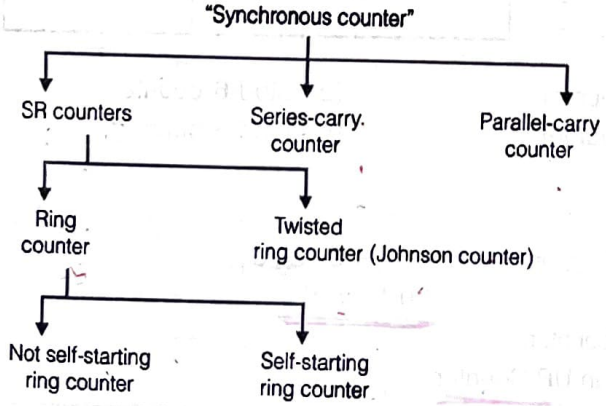
$$f_{max} = \frac{1}{n t_{pd(FF)}} = \frac{1}{4 \times 50\text{ns}}$$

$$= \frac{1 \times 10^9}{4 \times 50} = \frac{1}{20} = 10^8 = 5 \times 10^6$$

$$f_{max} = 5\,\text{MHz}$$

Synchronous (Parallel) Counters

The problems encountered with ripple counters are caused by the accumulated flip-flops propagation delay. In other words, the flip-flops do not change states simultaneously in synchronism with the input pulses. These limitations can be overcome with the use of synchronous or parallel counters in which all the flip-flops are triggered simultaneously by the CLK input pulses.



NOT-Self Starting Ring Counter

It is nothing but SISO shift register. It is also known as "End-Carry Counter" and is a synchronous counter.

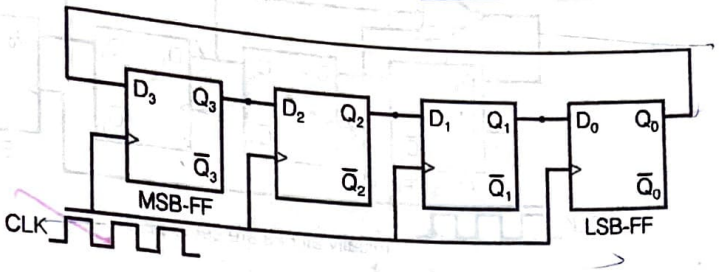
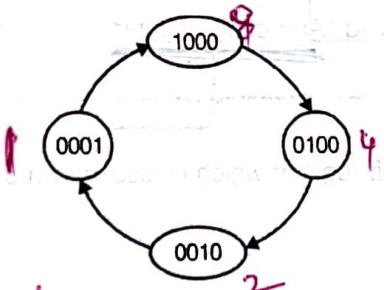


Fig. 7.4(a)

In this counter LSB-flip-flop output "Q_0" is connected to the MSB-flip-flop input (D_3). In this only one bit is high and circulates among all the flip-flops.

(if Initially $D_3 = 1$)

Clock	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	
1	1	0	0	0	
2	0	1	0	0	4-states
3	0	0	1	0	
4	0	0	0	1	
5	1	0	0	0	
6	0	1	0	0	4-states
7	0	0	1	0	
8	0	0	0	1	
9	1	0	0	0	



(State diagram)

Fig. 7.4(b)

⇒ with n flip-flops, there are n-states present in ring counter.

⇒ with n flip-flops, maximum count possible in ring counter is (2^{n-1}).

⇒ Decoding is very easy in ring counter, because there is no aid of extra circuit.

Applications of ring counter

- A/D-converter
- Stepper motors
- In controlled signal generation such as interrupts

Self-Starting Ring Counter

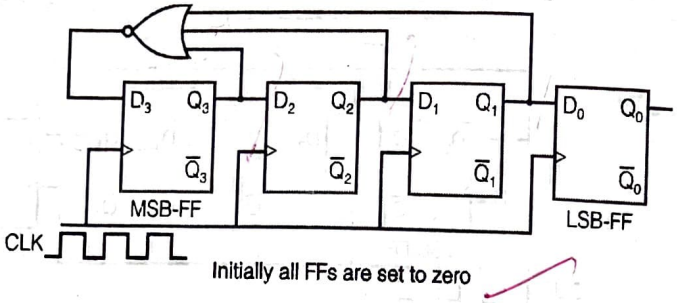


Fig. (7.5)

Clock	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	
1	1	0	0	0	4-states
2	0	1	0	0	
3	0	0	1	0	
4	0	0	0	1	
5	1	0	0	0	4-states
6	0	1	0	0	
7	0	0	1	0	
8	0	0	0	1	
9	1	0	0	0	

⇒ In 4-bit ring counter the used states = 4 and the unused input = $2^4 - 4 = 12$.

⇒ In any counter if CLK frequency is "f" the flip-flops output frequency is "f/N" (where N = No. of states).

⇒ This counter is also called "And carry counter".

Twisted-Ring Counter

This counter is nothing but a shift register which is also known as:

- Johnson Counter
- Switch Tail Ring Counter
- Mobies Counter
- Creeping Counter

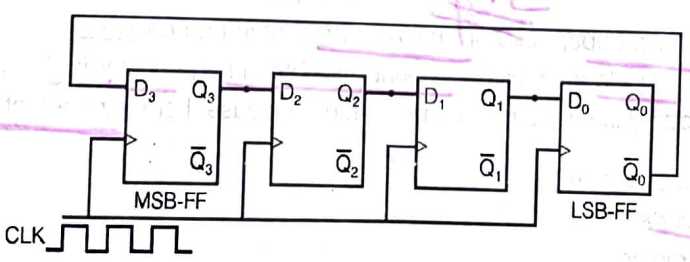


Fig. (7.6)

Clock	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	8-states
1	1	0	0	0	
2	1	1	0	0	
3	1	1	1	0	
4	1	1	1	1	
5	0	1	1	1	
6	0	0	1	1	
7	0	0	0	1	
8	0	0	0	0	

⇒ with n flip-flops there are 2n states in this counter.
⇒ with n flip-flops the maximum count by this counter is $(2^n - 1)$.
⇒ In normal "Johnson Counter" with n flip-flops and the input frequency is 'f' then output frequency of flip-flops is "f/2n" (where N = No. of states = 2n).

When a counter is entered into the unused state and counter is functioning in only on unused states then this counter is called "Lock-out Stage Counter". In a counter if a feedback connection is used the number of possible states will decreases.

Synchronous-Series Carry Counter

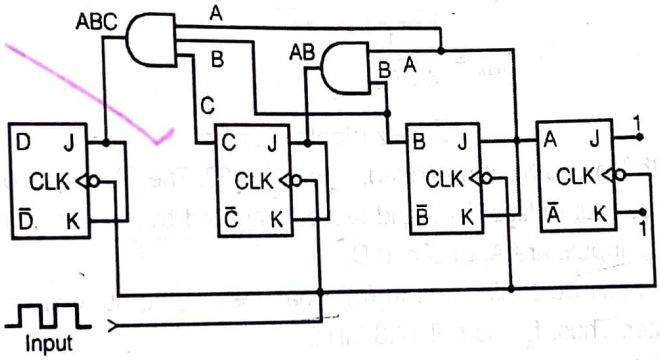


Fig. (7.7)

Circuit Operation

The basic principle of operation of this synchronous counter is "The J and K inputs of the flip-flops are connected so that only those flip-flops that are supposed to toggle on a given negative going transition will have J = K = 1 when that negative going transition occurs". In this counter,

$$T_{CLK} \geq t_{pd(FF)} + (n-2)\, t_{pd(AND\text{-}gate)} \quad ...(7.6)$$

Most important advantages of this counter is that it reduces the "decoding error". Total delay of this counter is much lower than an asynchronous counter with same number of flip-flops.

i.e. $$\text{total delay} = FF\, t_{pd(FF)} + t_{pd(AND\ gate)} \quad ...(7.7)$$

Synchronous-Parallel Carry Counter

It is physically same as that the counter shown in figure 7.7, but it is logically different. It is the "Fastes Counter". Clock frequency for this counter is given as,

$$f_{CLK} = \frac{1}{t_{pd(FF)} + t_{pd(AND\ gate)}} \quad \text{...(7.8}$$

Example 7.7

(a) Determine f_{max}, if t_{pd} for each flip-flop is 50ns and t_{pd} for each AND gate is 20ns. Compare this with f_{max} for a MOD-16 ripple counter.

(b) What has to be done to convert this counter to MOD-37?

(c) Determine f_{max} for the MOD-32 parallel counter.

Solution:

(a) The total delay that must be allowed between input clock pulses is equal to flip-flop t_{pd} + AND gate t_{pd}. Thus, $T_{clock} \geq 50 + 20 = 70ns$, and so the parallel counter has

$$f_{max} = \frac{1}{70ns}$$

$$= 14.3 \text{ MHz (parallel counter)}$$

A MOD-16 ripple counter uses four flip-flops $t_{pd} = 50$ ns. Thus, f_{max} for the ripple counter is

$$f_{max} = \frac{1}{4 \times 50ns}$$

$$= 5 \text{ MHz (ripple counter)}$$

(b) A fifth flip-flop must be added, since $2^5 = 32$. The CLK input of this flip-flop is also tied to the input pulses. Its J and K inputs are fed by the output of a four-input AND gate whose inputs are A, B, C and D.

(c) f_{max} is still determined as in (a) regardless of the number of flip-flops in the parallel counter. Thus, f_{max} is still 14.3 MHz.

Synchronous Counter Design

Synchrounous counters for any given count sequence and modulus can be designed in the following way:

- Find the number of flip-flops required.
- Write the count sequence in the tabular form.
- Determine the flip-flop inputs which must be present for the desired next state from the present state using the excitation table of the flip-flops.
- Prepare K-map for each flip-flop input in terms of flip-flop outputs as the input variables. Simplify the K-maps and obtain the minimized expressions.
- Connect the circuit using flip-flops and other gates corresponding to the minimized expressions.

Example 7.8

Design a 3-bit synchronous counter using J-K flip-flops.

Solution:

The number of flip-flops required is 3. Let the flip-flops be FF0, FF1 and FF2 and their inputs and outputs are given below:

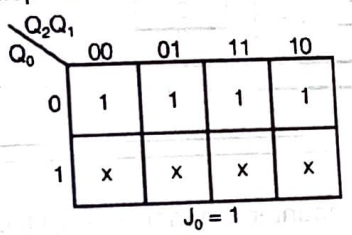
Flip – flop	Inputs	Output
FF0	J_0, K_0	Q_0
FF1	J_1, K_1	Q_1
FF2	J_2, K_2	Q_2

Also add the table for the next state as shown:

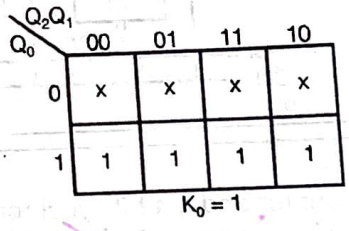
Present state			Next state			Flip-flop inputs					
						FF_0		FF_1		FF_2	
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_0	K_0	J_1	K_1	J_2	K_2
0	0	0	0	0	1	1	x	0	x	0	x
0	0	1	0	1	0	x	1	1	x	0	x
0	1	0	0	1	1	1	x	x	0	0	x
0	1	1	1	0	0	x	1	x	1	1	x
1	0	0	1	0	1	1	x	0	x	x	0
1	0	1	1	1	0	x	1	1	x	x	0
1	1	0	1	1	1	1	x	x	0	x	0
1	1	1	0	0	0	x	1	x	1	x	1
0	0	0									

through Excitation Table of J-K

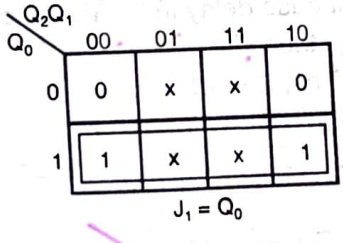
The count sequence and the required inputs of flip-flops are given in table. The inputs to the flip-flops are determined by using K-MAP.



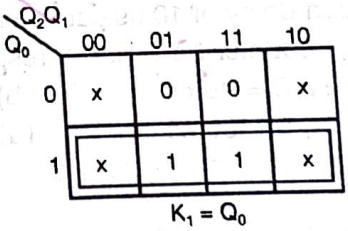
$J_0 = 1$



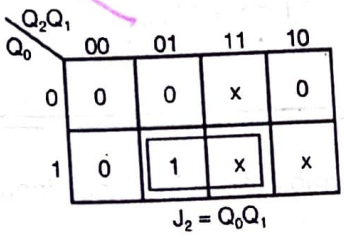
$K_0 = 1$



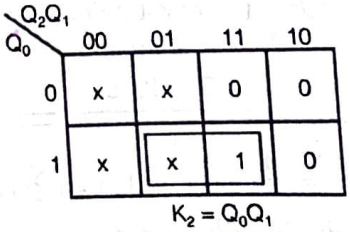
$J_1 = Q_0$



$K_1 = Q_0$



$J_2 = Q_0Q_1$



$K_2 = Q_0Q_1$

Consider one column of the counter state at a time and start from the first row.

⇒ We consider Q_0. Before the first pulse is applied, $Q_0 = 0$ and it is required to be 1 at the end of the first clock pulse.

⇒ Therefore, to achieve this condition, the values of J_0 and K_0 are 1 and x respectively (from the excitation table). These are entered in the table in the row corresponding to 0 pulse.

⇒ When the second clock pulse is applied Q_0 is to change from 1 to 0, therefore, the required inputs are

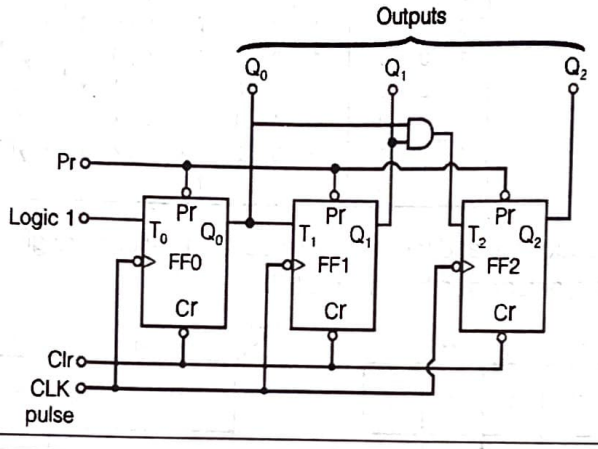
$$J_0 = x, K_0 = 1$$

⇒ In a similar manner inputs of each flip-flop are determined.

⇒ Now, we prepare the K-maps with Q_2, Q_1 and Q_0 as input variables and flip-flop inputs are output variables as:

$$J_0 = 1, \quad K_0 = 1$$
$$J_1 = Q_0, \quad K_1 = Q_0$$
$$J_2 = Q_0 Q_1, \quad K_2 = Q_0 Q_1$$

Now the final circuit of 3-bit synchronous counter is,



Example 7.9

A 4 bit ripple counter and a 4 bit synchronous counter are made using flip flops having a propagation delay of 10 ns each. If the worst case delay in the ripple counter and the synchronous counter be R and S respectively, then

(a) R = 10 ns, S = 40 ns (b) R = 40 ns, S = 10 ns

(c) R = 10 ns, S = 30 ns (d) R = 30 ns, S =10 ns

[GATE-2003]

Solution: (b)

Propagation delay (t_{pd}) of 4-bit ripple counter = 4 × 10ns = 40 ns

∴ R = 40ns

and S = 10ns

Example 7.10

A 0 to 6 counter consists of 3 flip flops and a combination circuit of 2-input gate(s). The combination circuit consists of

(a) one AND gate
(b) one OR gate
(c) one AND gate and one OR gate
(d) two AND gates

[GATE-2003]

Solution: (d)
From the circuit diagram of figure (7.7), we can say
It consists of 2-AND gates.

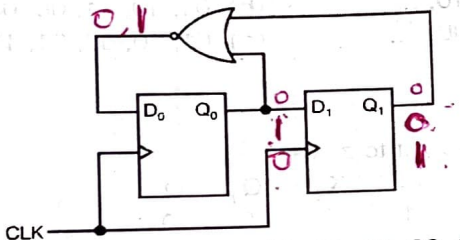
Example 7.11
Choose the correct one from among the alternatives A, B, C, D after matching an item from Group-1 with the most appropriate item in Group-2.

Group-1	Group-2
P. Shift register	1. Frequency division
Q. Counter	2. Addressing in memory chips
R. Decoder	3. Serial to parallel data conversion

(a) P – 3, Q – 2, R–1
(b) P – 3, Q – 1, R – 2
(c) P – 2, Q – 1, R – 3
(d) P – 1, Q – 2, R – 2

Solution: (b)

Example 7.12
For the circuit shown, the counter state (Q_1Q_0) follows the sequence



(a) 00, 01, 10, 11, 00 ...
(b) 00, 01, 10, 00, 01 ...
(c) 00, 01, 11, 00, 01 ...
(d) 00, 10, 11, 00, 10 ...

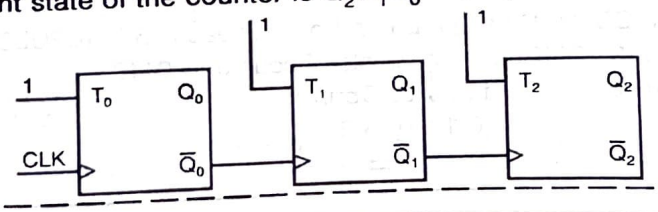
[GATE-2003]

Solution: (b)

Q_1	Q_0
0	0
0	1
1	0
0	0

$\therefore$ $(Q_1\ Q_0) \rightarrow$ 00, 01, 10, 00, 10.....

Example 7.13
The given figure shows a ripple counter using positive edge triggered flip-flops. If the present state of the counter is $Q_2Q_1Q_0 = 011$, then its next state $(Q_2Q_1Q_0)$ will be



(a) 010 (b) 100
(c) 111 (d) 101

[GATE-2005]

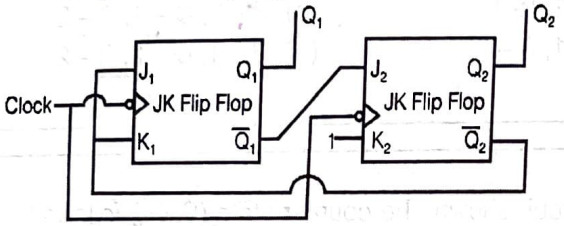
Solution: (b)

From this given circuit, CLK is (+)ve edge triggered and $\bar{Q}$ is connected as output so that counter is an "UP Counter".

	Q_2	Q_1	Q_0
Present state	0	1	1
	↓	↓	↓
Next state	1	0	0

Example 7.14

What are the counting states (Q_1, Q_2) for the counter shown in the figure below?



(a) 11, 10, 00, 11, 10, ... (b) 01, 10, 11, 00, 01 ....
(c) 00, 11, 01, 10, 00, ... (d) 01, 10, 00, 01, 10, ...

[GATE-2009]

Solution: (a)

Initially all flip-flops are set to zero.

CLK	J_1	K_1	J_2	K_2	Q_1	Q_2
0	1	1	1	1	0	0
1	1	1	1	1	1	1
2	0	0	0	1	1	0
3	1	1	0	1	1	1

Correct sequence is, $\boxed{00}$, 11, 10, 00, 11, 10,.....
⇩
Initial

Example 7.15

The initial state of MOD-16 down counter is 0110. After 37 clock pulses, the state of the counter will be

(a) 1011 (b) 0110
(c) 0101 (d) 0001

[IES-1999]

Solution: (d)

Since, this is a **MOD-16 Down Counter**, So, after each 16 CLK PULSE, the counter contents = 0110 also after 32 CLK pulse, contents of counter = 0110.

No. of CLK Pulse	Contents of Counter
32nd	0 1 1 0 ⇒ 6
33rd	0 1 0 1 ⇒ 5

34th	
35th	0100 ⇒ 4
36th	0011 ⇒ 3
37th	0010 ⇒ 2
	0001 ⇒ 1

Example 7.16

Write the counting sequence of a 4-bit down synchronous counter. Design the same by using (–)ve edge triggered J-K flip-flops. [IES-2009 : 10 marks]

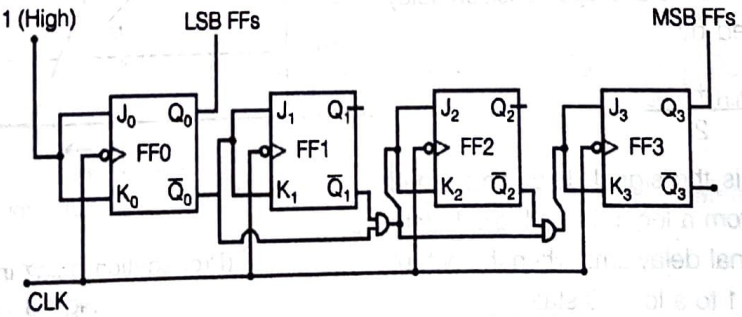
Solution:

The counting sequence of a 4-bit down synchronous counter is as initially 0000, then after first clock pulse 1111, then 1110 and so on, which can be listed by truth table.

CLOCK PULSE	STATES OF THE COUNTERS				
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	→ Initially
1	1	1	1	1	
2	1	1	1	0	
3	1	1	0	1	
4	1	1	0	0	
5	1	0	1	1	
6	1	0	1	0	
7	1	0	0	1	
8	1	0	0	0	
9	0	1	1	1	
10	0	1	1	0	
11	0	1	0	1	
12	0	1	0	0	
13	0	0	1	1	
14	0	0	1	0	
15	0	0	0	1	
16	0	0	0	0	
17	1	1	1	1	→ Again repeat

For designing this counter we require 4 JK-flip-flops. From the above truth table it is clear that Q_0 changes its own state after each clock pulses, so FF0 i.e. LSB-flip-flop is in toggled mode i.e. $J_0 = K_0 = 1$.

Again Q_1 changes states only when $Q_0 = 0$ i.e. FF1 toggles only when $\overline{Q}_0 = 1$, so connect $\overline{Q}_0$ to J_1 and K_1. Q_2 changes its state only when $Q_0 = 0$ and $Q_1 = 0$ i.e. $\overline{Q}_0 = 1$ and $\overline{Q}_1 = 1$ i.e. FF2 toggles only when $\overline{Q}_0\,\overline{Q}_1 = 1$, so connect $\overline{Q}_0\,\overline{Q}_1$ to J_2 and K_2 and again in similar fasion to obtained the 4-bit synchronous down counter as shown below:



(Logic gate diagram of a 4-bit down counter using J-K flip-flops)

OOOO