Digital Logic

(1) Range of n bit 2's complement number :-

$$-2^{(n-1)} \text{ to } (2^{(n-1)} - 1)$$

(2) Range of n bit 1's complement number :-

$$-(2^{(n-1)} - 1) \text{ to } (2^{(n-1)} - 1)$$

(3) Implicant :-

A normal product term that implies Y.

$Y = AB + ABC + BC$ → Implicants

(4) Prime Implicants :-

An implicant of Y such that if any variable is removed from the implicant, the resulting term does not imply Y.

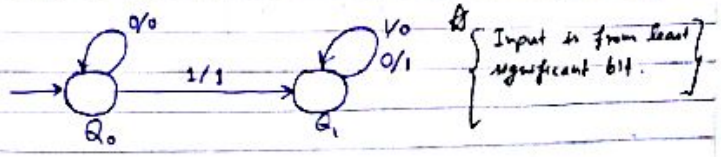
$Y = AB + BC + ABC$ → Not a prime implicant.

AB, BC → Prime Implicant

(5) Essential Prime Implicant :-

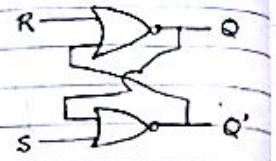
If a minterm is covered only by a prime implicant, then this prime implicant is called an essential prime implicant.

(6) Finite State Machine to calculate 2's complement :-



S-R (Set - Reset) Flip-Flop :-

S(t)	R(t)	Q(t)	Q(t+∈)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	– } inputs not allowed
1	1	1	– }



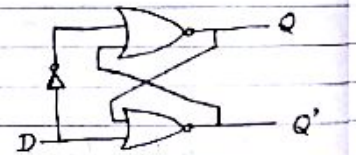
Characteristic Equation :-

$$Q^+ = S + R'Q$$

$(SR = 0)$
↳ Condition for both of the i/ps to be complimentary

D (delay) Flip Flop :-

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

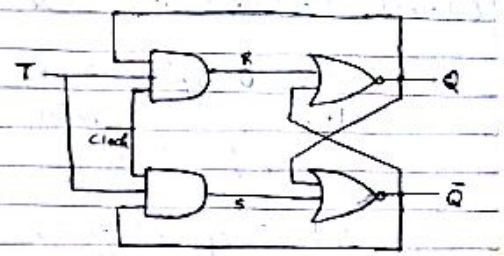


Characteristic Equation :-

$$Q^+ = D$$

T (Trigger) Flip-Flop :-

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

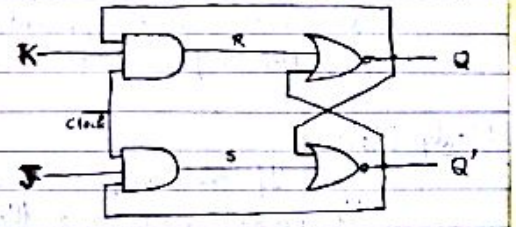


Characteristic Equation :-

$$Q^+ = T'Q + TQ' = T \oplus Q$$

JK Flip-Flop :-

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



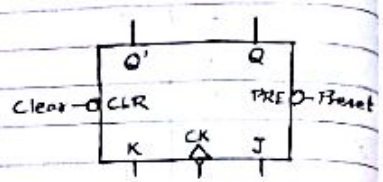
Characteristic Equation :-

$$Q^+ = QK' + Q'J$$

Characteristic Equation :-

An equation which expresses the next state of a flip-flop in terms of its present state and inputs.

Clocked Flip-Flops with Clear and Preset Inputs :-



A 0 applied to the clear input will reset the flip-flop to Q = 0

A 0 applied to the preset input will set the flip-flop to Q = 1

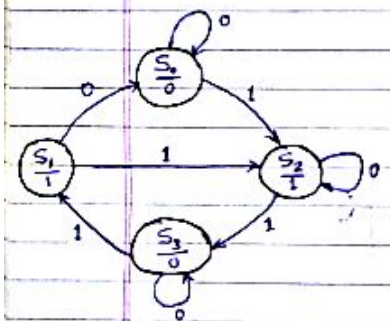
These inputs override the clock and J K inputs

Moore Machine :-

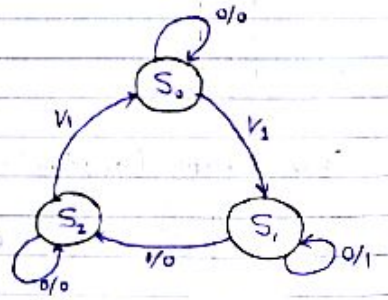
If the output of a sequential network is a function of the present state only, the network is often referred to as a Moore Machine.

Mealy Machine :-

If the output is a function of both the present state and the input, the network is referred to as a Mealy Machine.



Moore Machine



Mealy Machine

Positive Logic :-

+V represent a logic 1 and 0 volts represent a logic 0.

Negative Logic :-

+V represent a logic 0 and 0 volts represent a logic 1.

Finding the minimum number of Gates required to solve an expression :-

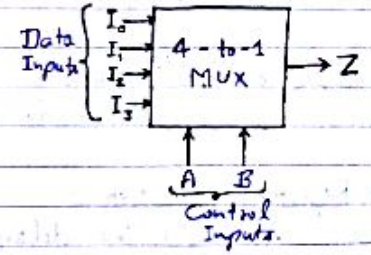
To find a minimum solution, one must find both the network with the AND-gate output and the one with the OR-gate output.

Multiplexers :→

A multiplexer has a group of data inputs and a group of control inputs.

The control inputs are used to select one of the data inputs and connect it to the output terminal.

$$Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$$

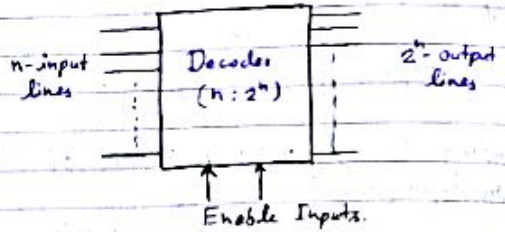


A 2^n to 1 multiplexer can be used to realize any (n+1) variable function with no added gates.

n of the variables are used as control inputs and the remaining variable is used as required on the data inputs.

Decoders :-

A decoder is a combinational logic circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.



Encoders :-

An encoder is a combinational logic circuit. It is reverse of decoder function. It has 2^n (or fewer) input lines and n output lines.

Multiplexer Applications :-

(1) Data Routing
(2) Function Generator
(3) Parallel to Serial Conversion

Demultiplexer Applications :-

(1) Serial to Parallel Conversion.

Shift Register Applications :-

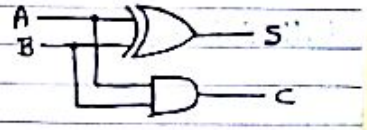
(1) Time delay
(2) Serial to parallel data converter
(3) Parallel to serial data converter

Canonical Logic Forms :-

If each term in SOP and POS forms contain all the literals then these are known as Canonical (Standard) SOP and POS expression.

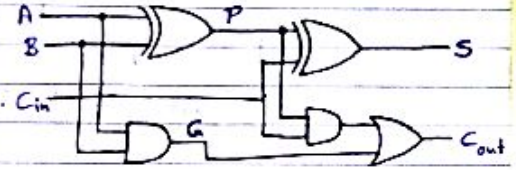
Half Adder :-

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



Full Adder :-

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



Look Ahead Carry Adder :-

$$S_i = P_i \oplus C_i \qquad \begin{cases} P_i = A_i \oplus B_i \\ G_i = A_i \cdot B_i \end{cases}$$
$$C_{i+1} = G_i + P_i C_i$$

$i = 0$

$$C_1 = G_0 + P_0 C_0$$

$i = 1$

$$C_2 = G_1 + P_1 C_1$$
$$= G_1 + P_1 (G_0 + P_0 C_0)$$

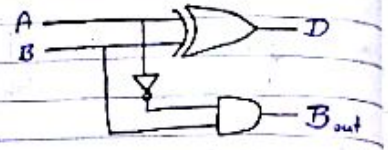
$i = 2$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0))$$

$i = 3$

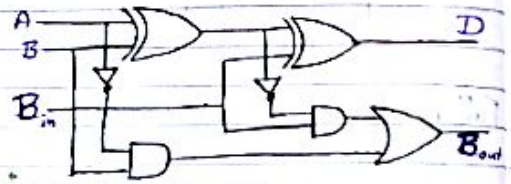
$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 (G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0)))$$

Half Substractor :-



$$D = A \oplus B$$
$$B_{out} = \bar{A}B$$

Full Substractor :-



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

Binary Code to Gray Code Conversion :-

Binary Code - 1 1 0 1 1 1

Gray Code - 1 0 1 1 0 0

Same

Gray Code to Binary Code Conversion :-

Gray Code - 1 0 1 1 0 0

Binary Code - 1 1 0 1 1 1

Same

Reduction of State Tables :-

(1) Elimination of Redundant States
(2) Determination of State Equivalence using an Implication Table.

B	~~A-B~~ B-C	A≡B if A≡D & B≡C ✓
C	~~A-B~~ B-B	A-A B-C
D	X	X X C≠D same output differ
	A	B C

Implication Table

(Present State)	0	1 ← I/p
A	D, 0	B, 1
B	A, 0	C, 1
C	A, 0	B, 1
D	A, 1	C, 1

Next State, o/p

$$B \equiv C$$

New State Table :-

	0	1
A	D, 0	B, 1
B	A, 0	B, 1
D	A, 1	B, 1