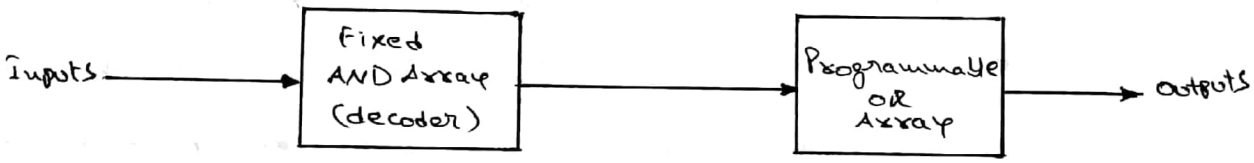
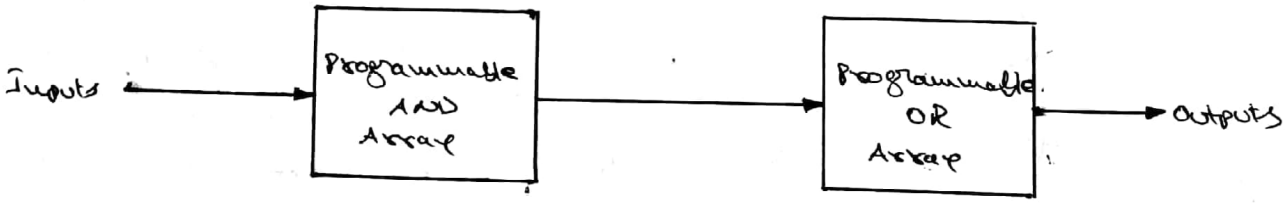
PROGRAMMABLE LOGIC DEVICES (PLD)

A combinational PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation. There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array.



(a) Programmable Read-Only Memory (PROM)

(b) Programmable array logic (PAL)



(c) Programmable logic Array (PLA)

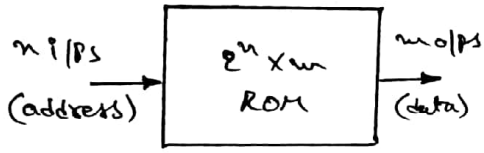
Fig 1: Basic Configuration of Three PLDs

Figure 1 shows the configuration of the three PLDs. The Programmable read-only memory (PROM) has a fixed AND array constructed as a decoder & programmable OR array. The programmable OR gates implement the Boolean functions in Sum of

in terms. The programmable array logic (PAL) has a Programmable AND array & a fixed OR array. The AND gates are programmed to provide the product terms of for the Boolean functions, which are logically summed in each OR gate. The most flexible PLD is the programmable logic array (PLA), where both the AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum & products implementation.

READ ONLY MEMORY

A Read only Memory (ROM) is essentially a memory device in which permanent binary information is stored The binary information must be specified by the designer and is then embedded in the unit to form the required interconnection pattern. Once the pattern is established, it stays within the unit even when power is turned OFF & ON again



The fig shows block diagram of a ROM. It consists of n i/Ps & m o/Ps. The i/Ps provide the address for the memory & the o/Ps give the data bits of the stored word which is selected by the address. The no. of words in a ROM is determined from the fact that n address i/p lines are needed to specify 2^n words.

Consider for Example a 32X8 ROM. The unit consists of 32 words of 8 bits each. There are five i/p lines that form the binary numbers from 0 through 31 for the address

The figure 3 shows the internal logic construction of the ROM. The five i/ps are decoded into 32 distinct o/ps by means of a 5×32 decoder. Each o/p of the decoder represents a memory address. The 32 o/ps of the decoder are connected to each of the eight OR gates. The Fig-4 shows the array logic convention used in complex circuits.

Fig-4

Each OR gate must be considered as having 32 i/ps. Each o/p of the decoder is connected to one of the i/ps of each OR gate. Since each OR gate has 32 i/p connections & there are 8 OR gates, the ROM contains 32×8 = 256 connections. In general, a $2^n \times m$ ROM will have an internal $m \times 2^n$ decoder & m OR gates. Each OR gate has 2^n i/ps, which are connected to each of the o/p of the decoder.

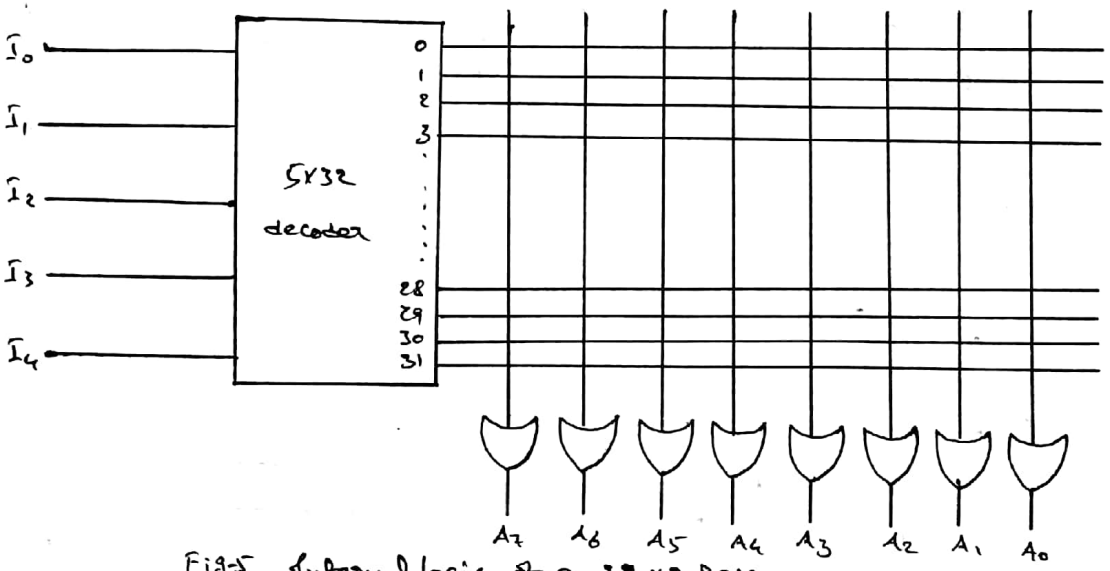


Fig-5 Internal logic of a 32×8 ROM

The 256 intersections in fig-5 are programmable. A programmable connection between two lines is logically equivalent to a switch that can be altered to either be close (meaning that the two lines are connected) or open (meaning that the two lines are disconnected). The programmable intersection between two lines is sometimes called a crosspoint. Various physical devices are used to implement crosspoint switches. One of the simplest technologies

employes a fuse that normally connects the two points, but is opened a "blown" by applying a high voltage to the fuse.

Combinational circuit implementation using ROM

1. Design full adder circuit using ROM

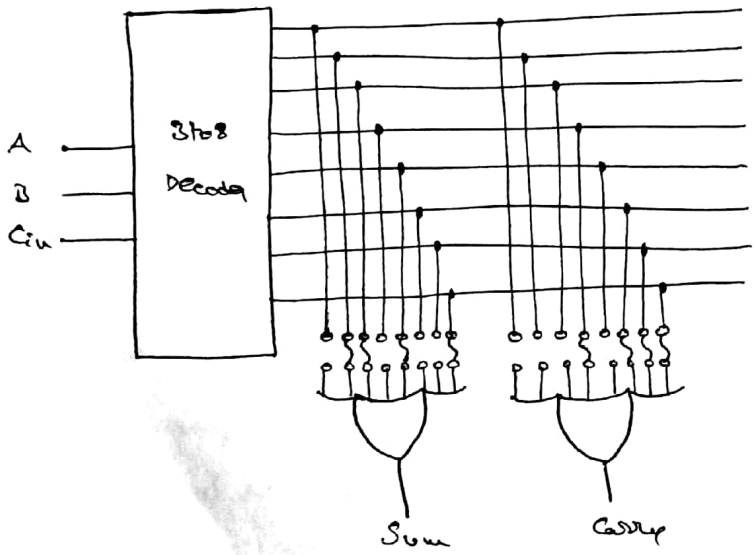
Full adder

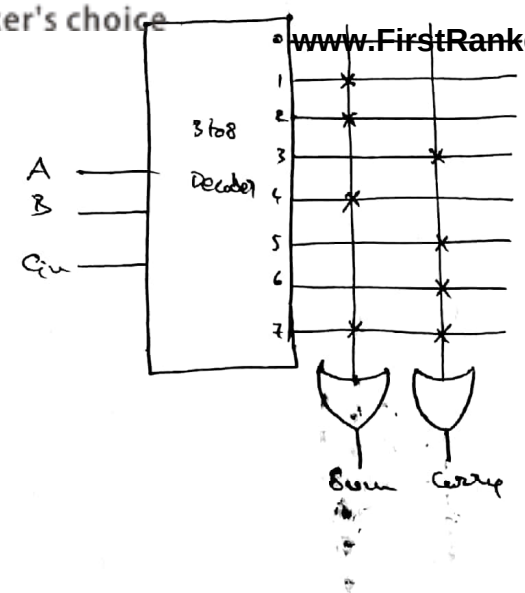
Truth table for full adder

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

$Sum(A, B, Cin) = \Sigma m(1, 2, 4, 7)$

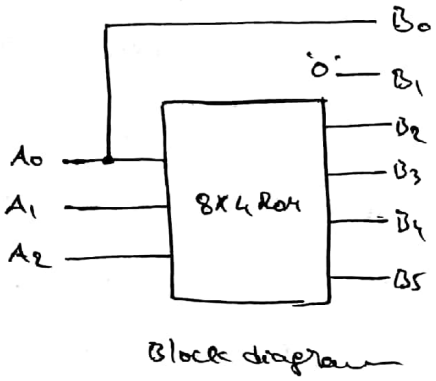
$Carry(A, B, Cin) = \Sigma m(3, 5, 6, 7)$





(2) Design a combinational circuit using a ROM. The circuit accepts a 3-bit number & generates an o/p binary number equal to the square of the i/p number.

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	$0^2 = 0$
0	0	1	0	0	0	0	0	1	$1^2 = 1$
0	1	0	0	0	0	1	0	0	$2^2 = 4$
0	1	1	0	0	1	0	0	1	$3^2 = 9$
1	0	0	0	1	0	0	0	0	$4^2 = 16$
1	0	1	0	1	1	0	0	1	$5^2 = 25$
1	1	0	1	0	0	1	0	0	$6^2 = 36$
1	1	1	1	1	0	0	0	1	$7^2 = 49$



Block diagram

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

ROM truth table.

PROGRAMMABLE LOGIC ARRAY (PLA)

The programmable logic array (PLA) is similar to the PROM in concept except that the PLA does not provide full decoding of the variables & does not generate all the minterms. The decoder is replaced by an array of AND gates that can be programmed to generate any product term of the i/p variables. The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.

The internal logic of a PLA shown in fig.5 to demonstrate the typical logic configuration of a PLA.

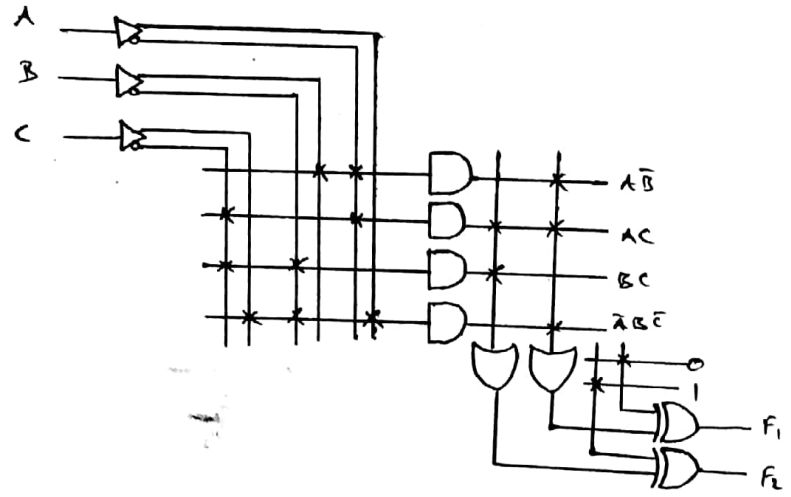


Fig-5: PLA with 3 i/ps, 4 product terms, & 2 o/ps.

Each i/p & its complement are connected to the i/ps of each AND gate as indicated by the intersections b/w the vertical & horizontal lines. The o/ps of the AND gates are connected to the i/ps of each OR gate. The o/p of the OR gate goes to an XOR gate where the other i/p can be programmed to receive a signal equal to either logic '1' or '0'. The o/p is inverted when the XOR i/p is connected to '1'. The o/p does not change when the XOR i/p is connected to '0'.

The product terms generated in each AND gate are listed along the o/p of the gate in the diagram. The product term is determined from the i/p whose crosspoints are connected & marked with a X. The output of an OR gate gives the logic sum of the selected product terms. The o/p may be complemented or left in its form depending on the connection for one of the XOR gate i/ps.

The fuse map of a PLA can be specified in a tabular form called PLA programming table. It consists of three sections. The first section lists the product terms numerically. The second section specifies the required paths between i/ps & AND gates. The third section specifies the paths between the AND & OR gates. For each o/p variable, we may have a T (true) or C (complement) for programming the XOR gate. The product terms listed on the left are not part of the table; they are included for reference only. For each product term, the i/ps are marked with 1, 0, or – (dash). If a variable in the product term appears in its true form, the corresponding i/p variable is marked with a '1'. If it appears complemented, the corresponding i/p variable is marked with a '0'. If the variable is ~~absent~~ absent in the product term, it is marked with a dash.

→ Implement the following two Boolean functions with a PLA:

$$F_1(A,B,C) = \Sigma(0,1,2,4)$$
$$F_2(A,B,C) = \Sigma(0,5,6,7)$$

Sol Simplify the given Boolean functions using K-Maps.

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$F_1 = \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C}$

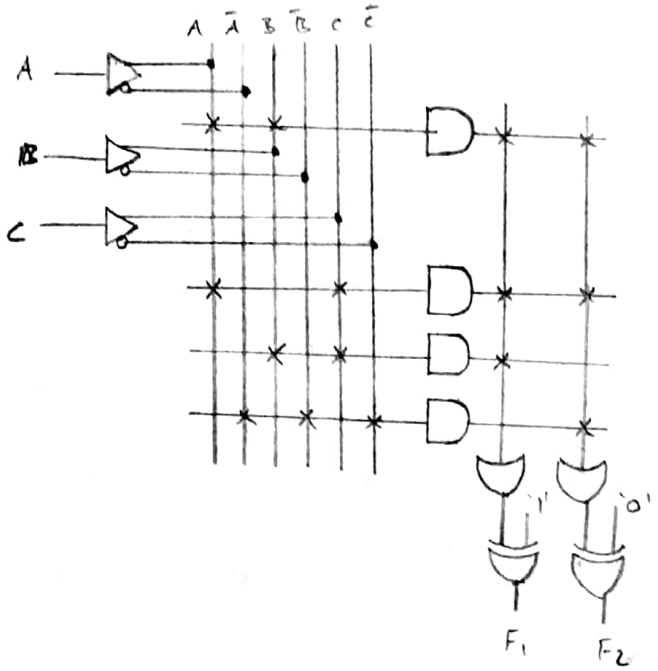
$\bar{F}_1 = (AB + AC + B\bar{C})$

A \ BC	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$F_2 = AB + AC + \bar{A}\bar{B}\bar{C}$

$\bar{F}_2 = (\bar{A}C + \bar{A}B + A\bar{B}\bar{C})$

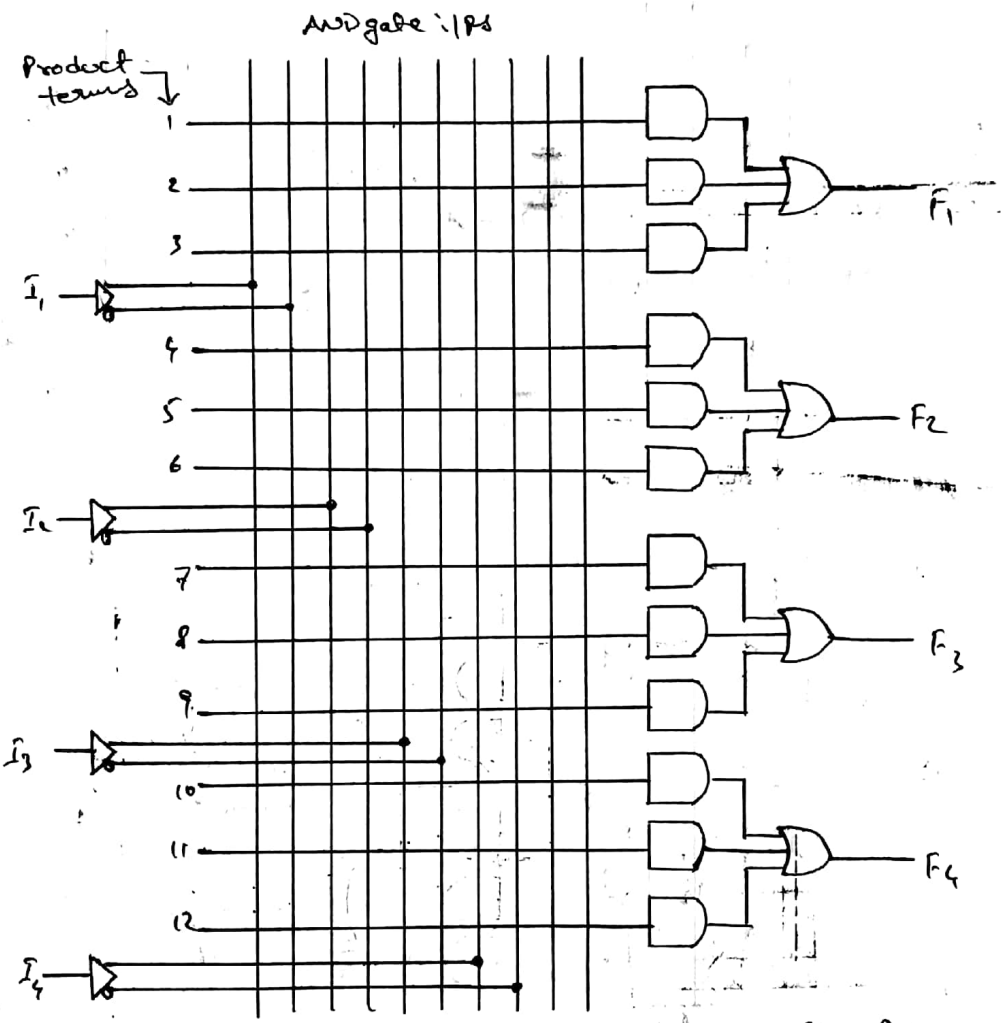
Product term		Inputs A	B	C	outputs (C) F1	(T) F2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A\bar{B}\bar{C}$	4	0	0	0	–	1



PROGRAMMABLE ARRAY LOGIC (PAL)

The programmable array logic (PAL) is a programmable logic device with a fixed OR array & a programmable AND Array. Because only the AND gates are programmable, the PAL is easier to program, but is not as flexible as the PLA. Fig shows the logic configuration of a typical PAL. It has four i/ps & four o/ps. Each i/p has a buffer - inverter gate and each o/p is generated by a fixed OR gate. There are four sections in the unit, each being composed of a three-wide AND-OR array. This is the term used to indicate that there are three programmable

AND gates in each section and a fixed OR gate. Each AND gate has 10 programmable i/p connections. This is shown in the diagram by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple-i/p configuration of the AND gate. One of the o/p is connected to a buffer-inverter gate and then fed back into two i/ps of the AND gate.



PAL 4 i/ps Four o/ps, & Three wide AND-OR, Structure.

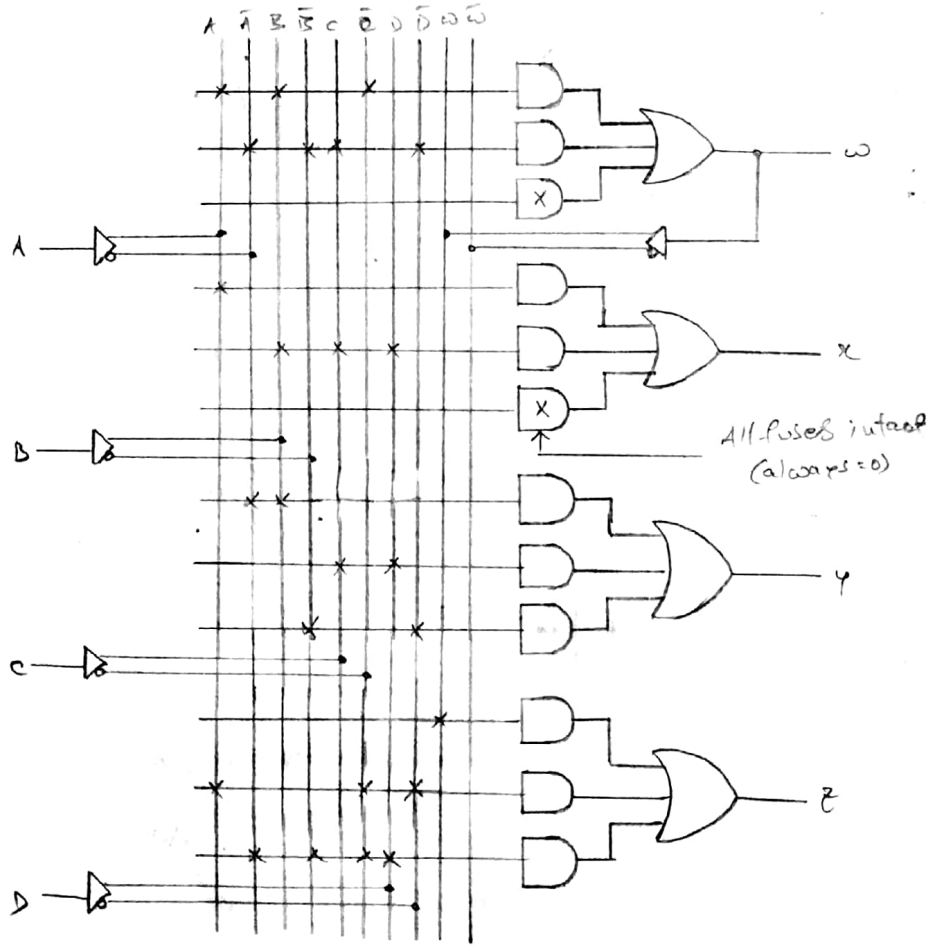
When designing with a PAL, the Boolean functions must be simplified to fit into each section. Unlike the PLA, a product term cannot be shared among two or more OR gates. Therefore, each function can be simplified by itself without regard to common product term. The no. of product terms in each section is fixed, and if the no. of terms in the function is too large, it may be necessary to use two sections to implement one Boolean function.

→ Using a PAL, the design of a combinational circuit, consider the following Boolean functions, given in Sum of minterms.

$$\omega(A,B,C,D) = \Sigma\,(2,12,13)$$
$$x(A,B,C,D) = \Sigma\,(7,8,9,10,11,12,13,14,15)$$
$$y(A,B,C,D) = \Sigma\,(0,2,3,4,5,6,7,8,10,11,15)$$
$$z(A,B,C,D) = \Sigma\,(1,2,8,12,13)$$

Sol

$$\omega = A\bar{B}\bar{C} + \bar{A}\bar{B}C\bar{D}$$
$$x = A + \bar{B}CD$$
$$y = \bar{A}B + CD + \bar{B}\bar{D}$$
$$z = A\bar{B}\bar{C} + \bar{A}\bar{B}C\bar{D} + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$
$$= \omega + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$



Tut

Comparison of programmable logic devices

PROM	PLA	PAL
1. AND array is fixed and OR array is programmable	1. Both AND & OR arrays are programmable.	1. OR array is fixed and AND array is programmable.
2. Cheaper & Simple to use	2. Costliest and more complex than PALs & PROMs	2. Cheaper & Simpler.
3. All minterms are decoded.	3. AND array can be programmed to get desired minterms.	3. AND array can be programmed to get desired minterms.
4. Only Boolean functions in standard SOP form can be implemented using PROM	4. Any Boolean function in SOP form can be implemented using PLA.	4. Any Boolean function in SOP form can be implemented PAL

UNIT-V

SEQUENTIAL CIRCUITS I

Classification Of Sequential Circuits (Synchronous And Asynchronous):

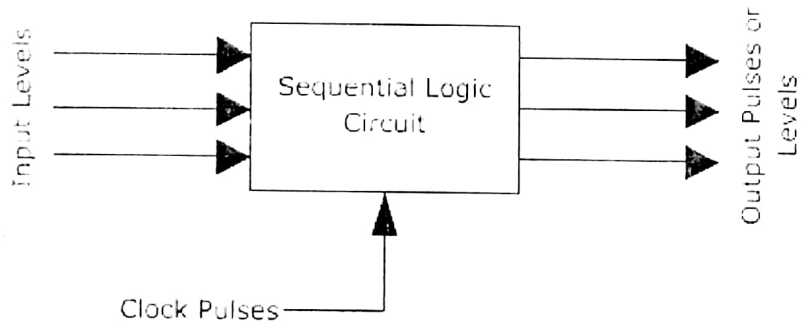
Classifications Of Sequential Circuits:

The sequential circuits are classified on the basis of timing of their signals into two types. They are.

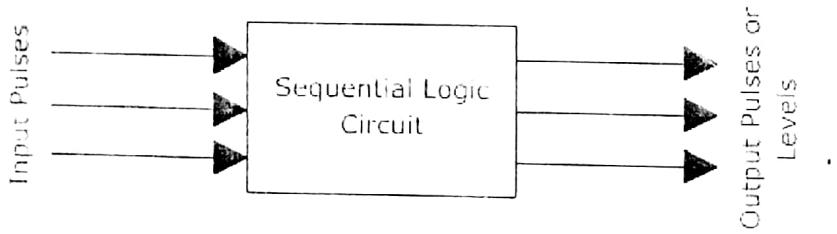
1) Synchronous sequential circuit.

2) Asynchronous sequential circuit.

There are two types of sequential circuit, synchronous and asynchronous.
Synchronous types use pulsed or level inputs and a clock input to drive the circuit (with restrictions on pulse width and circuit propagation)



Asynchronous sequential circuits do not use a clock signal as synchronous circuits do. Instead the circuit is driven by the pulses of the inputs.



A pulsed output is an output that lasts for the duration of a particular input pulse but can be less in some cases. For the clocked sequential circuits, the output pulse is the same duration as the clock pulse.
A level output refers to an output that changes state at the start of an input pulse or clock pulse and remains in that state until the next input or clock pulse.

This implies that if we had the outputs from the collectors of both the transistors, then the two outputs would be complementary.

In the flip-flops of various types that are available in IC form, we will see that all these devices offer complementary outputs usually designated as Q and Q'.

The R-S flip-flop is the most basic of all flip-flops. The letters R and S here stand for RESET and SET.

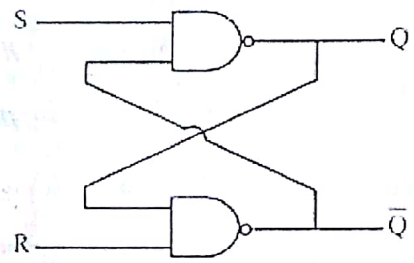
When the flip-flop is SET, its Q output goes to a 1 state, and when it is RESET it goes to a 0 state. The Q output is the complement of the Q output at all times.

Operation of RS flip-flop:

- When R input is low and S input is high, the Q output of flip-flop is set.
- When R input is high and S input is low, the Q output of flip-flop is reset.
- When both the inputs R and S are low, the output does not change.
- When both the inputs R and S are high, the output is unpredictable.

Truth Table:

			NAND Gate		
S	R	Q(t + 1)	A	B	Y
0	0	Indeterminate	0	0	1
0	1	Set	0	1	1
1	0	Reset	1	0	1
1	1	NC	1	1	0



JK flip flop:

A J-K flip-flop behaves in the same fashion as an R-S flip-flop except for one of the entries in the function table.

In case of R-S flip-flop, its input combination S = R = 1 (in the case of a flip-flop with active HIGH inputs) and the input combination S = R = 0 (in the case of a flip-flop with active LOW inputs) are prohibited.

In the case of a J-K flip-flop with active HIGH inputs, the output of the flip-flop toggles, that is, it goes to the other state, for J = K = 1.

The output toggles for J = K = 0 in the case of the flip-flop having active LOW inputs. Thus, the J-K flip-flop can overcome the problem of a forbidden input combination of the R-S flip-flop.

Figures below show the circuit symbol of level-triggered J-K flip-flops with active HIGH and active LOW inputs respectively with their function tables.

The characteristic tables for a J-K flip-flop with active HIGH J and K inputs and a J-K flip-flop with active LOW J and K inputs respectively are shown in both Figures (a) and (b).

The corresponding Karnaugh maps are shown in below figure for the characteristics table .

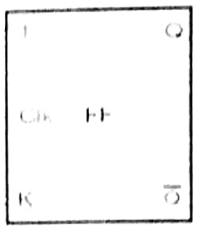
The characteristic equations for the Karnaugh maps of below figure is shown next.

$$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$$

$$Q_{n+1} = \overline{J}\overline{Q_n} + KQ_n$$

Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

(a)



Operation Mode	J	K	Clk	Q_{n+1}
SET	0	1	1	1
RESET	1	0	1	0
NO CHANGE	1	1	1	Q_n
TOGGLE	0	0	1	$\overline{Q_n}$

(b)

a) JK flip flop with active high inputs, b) JK flip flop with active low inputs

JK flip-flop is an universal flip-flop:

When configured in various ways, JK flip flop is capable of operating like most other types of flip - flops.

Operation of JK flip-flop:

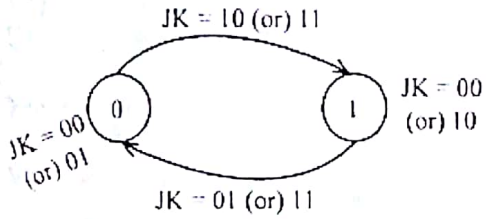
When K input is low and J input is high, the Q output of flip-flop is set.

When K input is high and J input is low, the Q output of flip-flop is reset.

When both the inputs K and J are low, the output does not change.

When both the inputs K and J are high, it is possible to set or reset the flip-flop (i.e) the output toggle on the next positive clock edge.

State diagram:

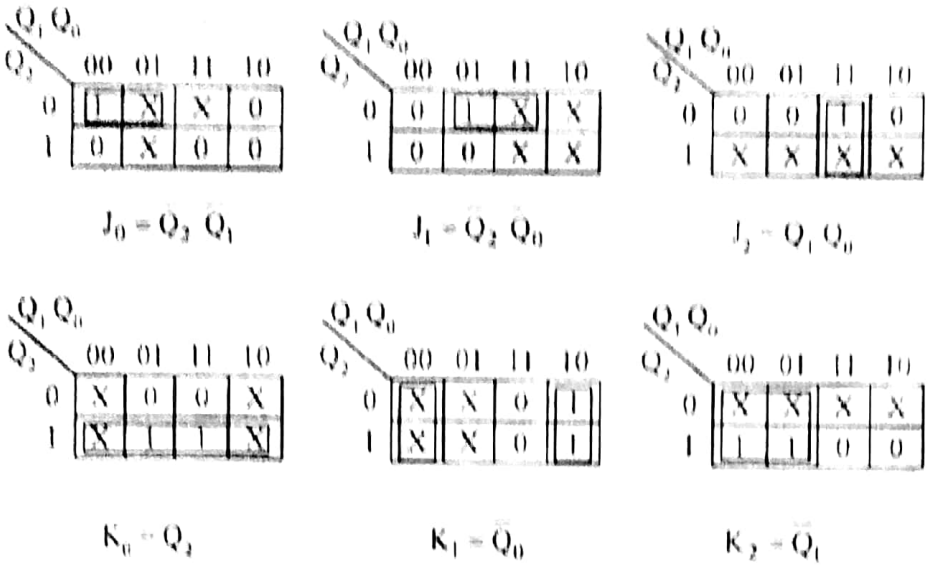


State table:

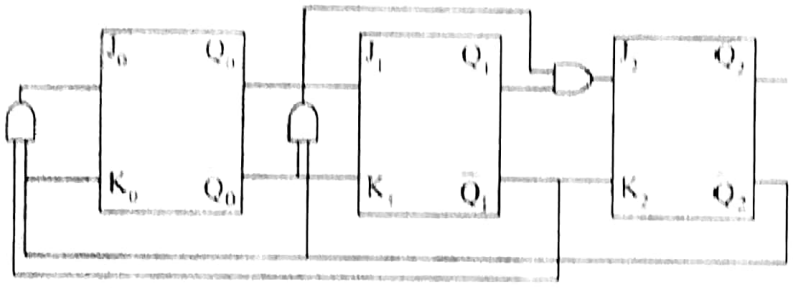
Q	J	K	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Counter using JK flip flop:

Q_2	Q_1	Q_0
0	0	0
0	0	1



Logic Diagram:



Problems

Let us consider that The following sequence is to be realized by a counter consisting of 3 JK Flip flops.

A_1	0	0	0	0	1	1	0
A_2	0	1	1	0	0	1	0
A_3	0	1	0	1	1	0	0

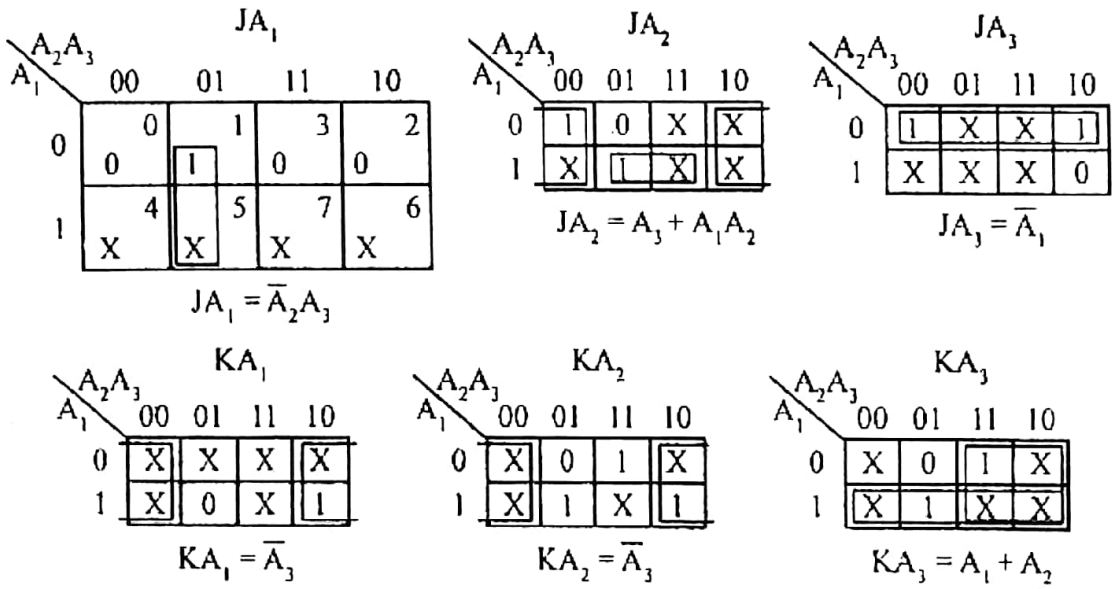
Here we will design the counter.

Solution:

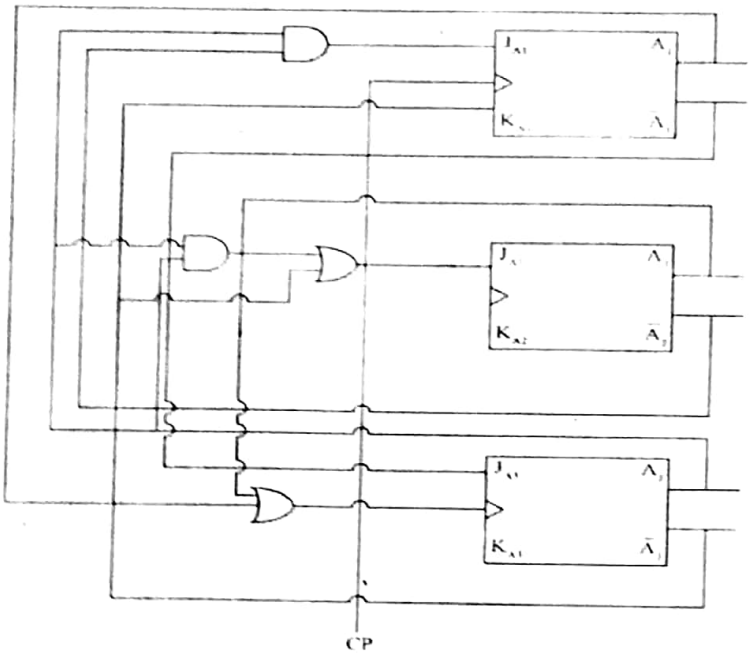
Truth table:

Present State			Next State			Flip-Flop Inputs					
A_1	A_2	A_3	A_1	A_2	A_3	JA_1	KA_1	JA_2	KA_2	JA_3	KA_3
0	0	0	0	1	1	0	X	1	X	1	X
0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	0	1	1	0	1	1	X	0	X	X	0
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

K - map:



State Diagram:



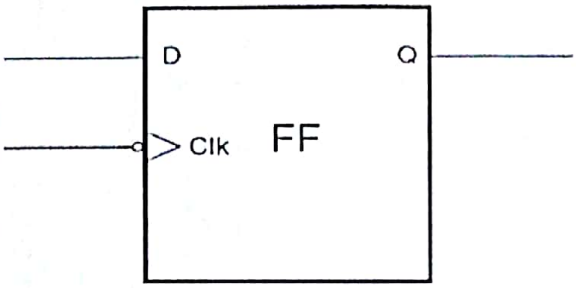
D flipflop:

A D flip-flop also called a delay flip-flop can be used to provide temporary storage of one bit of information.

Figure shows the symbol of the circuit and the function table of a negative edge-triggered D flip-flop.

When the clock is active, the data bit (0 or 1) present at the D input is transferred to the output.

In the D flip-flop, the data transfer from D input to Q output occurs on the negative-going (HIGH-to-LOW) transition of the clock input. D input can acquire new status.



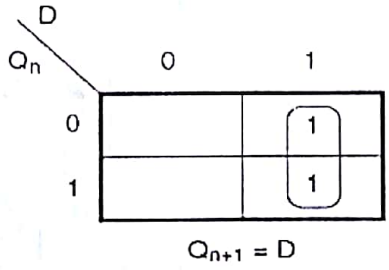
(a)

D	Clk	Q
0	↓	0
1	↓	1

(b)

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

(c)



$Q_{n+1} = D$

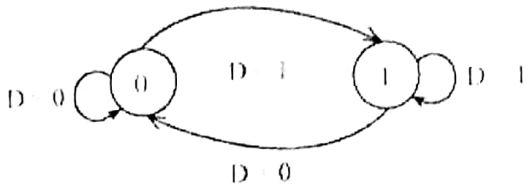
(d)

(a) Symbol, (b) Function table, (c) Truth table and (d) K-Map

Operation of D flip-flop:

In D flip-flop during the occurrence of clock pulse if D=1, the output Q is set and if D=0, the output is reset.

State diagram:



State table:

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

Flip-Flop excitation tables for D flip-flop:

In D flip-flop, its next state is always equal to the D input and it is independent of the present state.

D is 0 if Q_{n+1} has to 0 and if Q_{n+1} has to be 1 regardless the value of Q_n.

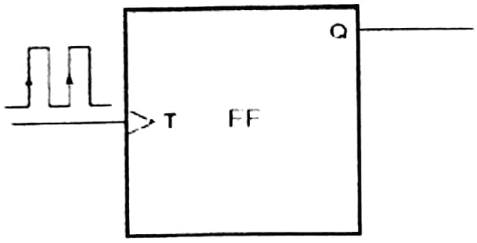
T flipflop:

The output of a toggle flip-flop also called a T flip-flop changes state every time it is triggered at its T input called the toggle input. That is, its output becomes 1, if it was 0 and 0 if it was 1.

If we consider the T input as active when HIGH, the characteristic table of such a flip-flop is shown in the figure. The Karnaugh maps for the characteristic tables of Figures are shown . The characteristic equations as written from the Karnaugh maps are as follows:

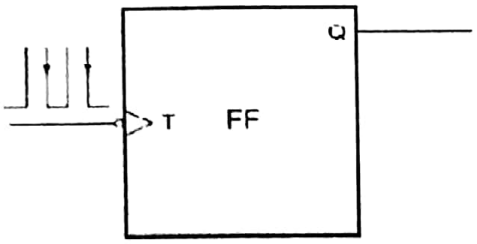
$$Q_{n+1} = T.\overline{Q_n} + \overline{T}.Q_n$$

$$Q_{n+1} = \overline{T}.\overline{Q_n} + T.Q_n$$



T	Q_n	Q_{n+1}
↑	0	1
↑	1	0

(a)



T	Q_n	Q_{n+1}
↓	0	1
↓	1	0

(b)

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

(c)

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

(d)

D flip-flop

Operation of T flip-flop:

T flip-flop is also known as Toggle flip-flop.

When the value of T=0, there is no change in the output.

When the value of T=1, the output switch to the complement state (i.e) the output toggles.

Flip-flop excitation tables for T flip-flop:

When input T=1 the state of the flip-flop is complemented.

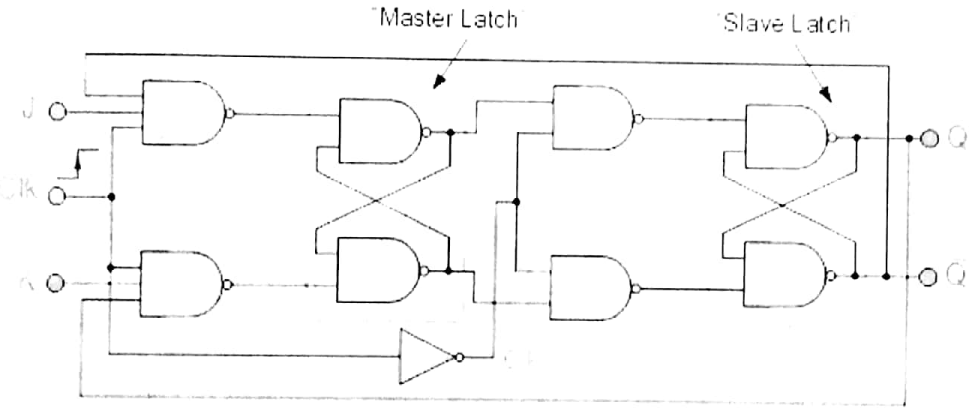
when T=0, the state of the Flip-flop remains unchanged. Therefore, for 0_0 and 1_1 transitions T must be 0 and for 0_1 and 1_0 transitions must be 1.

Master-Slave JK Flip-flop:

A master-slave flip-flop consists of two flip-flops where one circuit serves as a master and the other as a slave.

The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse.

The outputs from Q and Q from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip flop as shown below.



Master-Slave JK Flip Flop

The input signals J and K are connected to the gated "master" SR flip flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1".

As the clock input of the "slave" flip flop is the inverse (complement) of the "master" clock input, the "slave" SR flip flop does not toggle.

The outputs from the "master" flip flop are only "seen" by the gated "slave" flip flop when the clock input goes "LOW" to logic level "0".

When the clock is "LOW", the outputs from the "master" flip flop are latched and any additional changes to its inputs are ignored.

The gated "slave" flip flop now responds to the state of its inputs passed over by the "master" section.

Then ,on the "Low-to-High" transition of the clock pulse ,the inputs of the "master" flip flop are fed to the gated inputs of the "slave" flip flop and on the "High-to-Low" transition, the same inputs are reflected on the output of the "slave" making this type of flip flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is "HIGH" and passes the data to the output on the falling-edge of the clock signal.

In other words, the Master-Slave JK Flip flop is a "Synchronous" device as it only passes data with the timing of the clock signal.

Application of Master Slave FF:

- Frequency dividers
- Shift registers
- Parallel data storage
- Counters

Conversion from One Flip-Flop To Flip-Flop:

JK Flip Flop to SR Flip Flop:

S and R will be the external inputs to J and K. As shown in the logic diagram below, J and K values will be the outputs of the combinational circuit. Thus, the values of J and K have to be obtained in terms of S, R and Qp. The logic diagram is shown below.

A conversion table is to be written using S, R, Q_p, Q_{p+1}, J and K. For two inputs S and R, eight combinations can be made. For each combination, its corresponding Q_{p+1} outputs are found out.

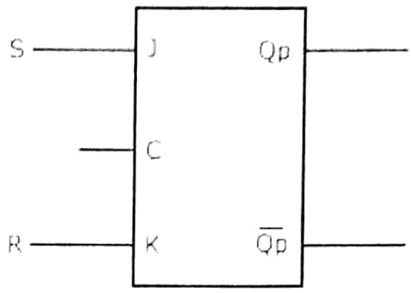
The outputs for the combinations of S=1 and R=1 are not permitted for an SR flip flop. Hence, the outputs are considered as invalid and the J and K values are taken as "don't cares".

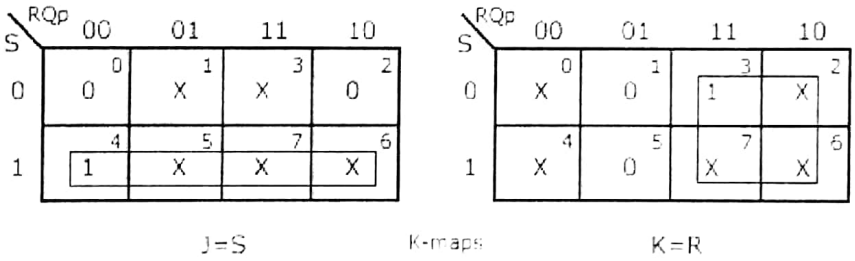
J-K Flip Flop to S-R Flip Flop

Conversion Table

S-R Inputs S	R	Outputs Qp	Qp+1	J-K Inputs J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	

Logic Diagram



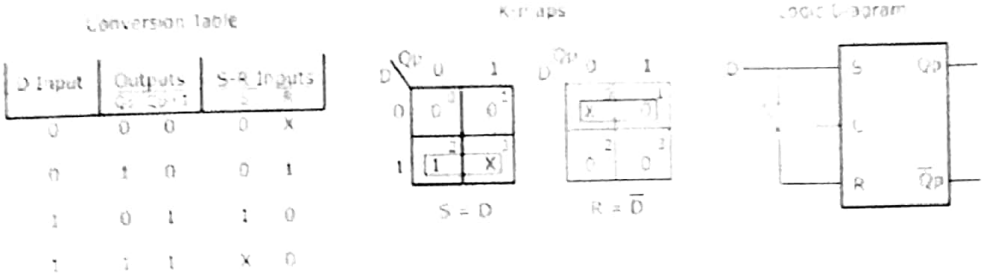


Conversion table, K-Map and Logic diagram

SR Flip Flop to D Flip Flop:

As shown in the figure, S and R are the actual inputs of the flip flop and D is the external input of the flip flop. The four combinations, the logic diagram, conversion table and the K-map for S and R in terms of D and Qp are shown below.

S-R Flip Flop to D Flip Flop



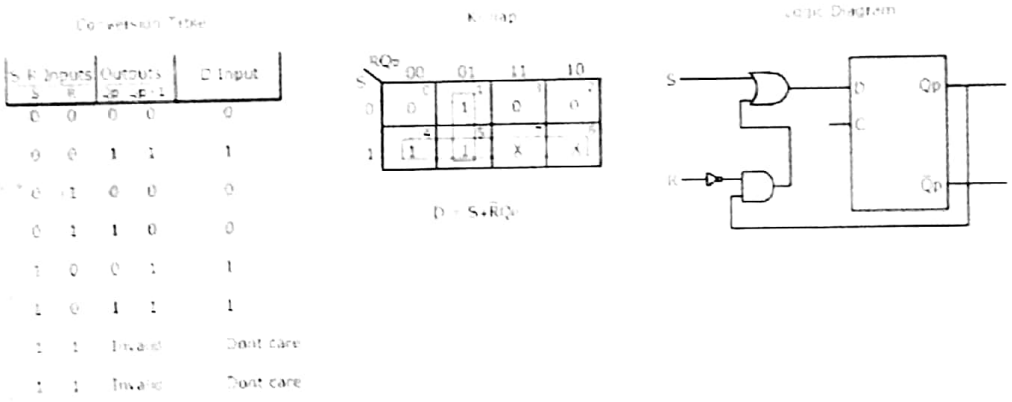
Conversion table, K-Map and Logic diagram

D Flip Flop to SR Flip Flop:

D is the actual input of the flip flop and S and R are the external inputs. Eight possible combinations are achieved from the external inputs S, R and Qp. Since the combination of S=1 and R=1 are invalid, the values of Qp+1 and D are considered as "don't cares".

The logic diagram showing the conversion from D to SR, and the K-map for D in terms of S, R and Qp are shown below.

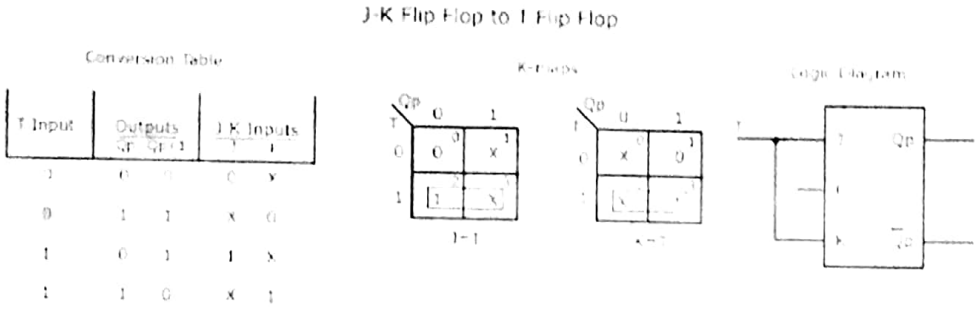
D Flip Flop to S-R Flip Flop



Conversion table, K-Map and Logic diagram

JK Flip Flop to T Flip Flop:

J and K are the actual inputs of the flip flop and T is taken as the external input for conversion. Four combinations are produced with T and Qp. J and K are expressed in terms of T and Qp.



Conversion table, K-Map and Logic diagram

Design Of Ripple Counters, Design Of Synchronous Counters, Johnson Counter, Ring Counter:

A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous counters are also called ripple-counters because of the way the clock pulse ripples it way through the flip-flops.

The MOD of the ripple counter or asynchronous counter is 2^n if n flip-flops are used. For a 4-bit counter, the range of the count is 0000 to 1111 (2^4-1). A counter may count up or count down or count up and down depending on the input control. The count sequence usually repeats itself. When counting up, the count sequence goes from 0000, 0001, 0010, ... 1110 , 1111 , 0000, 0001, ... etc. When counting down the count sequence goes in the opposite manner: 1111, 1110, ... 0010, 0001, 0000, 1111, 1110, ... etc.

The complement of the count sequence counts in reverse direction. If the uncomplemented output counts up, the complemented output counts down. If the uncomplemented output counts down, the complemented output counts up.

There are many ways to implement the ripple counter depending on the characteristics of the flip flops used and the requirements of the count sequence.

Clock Trigger: Positive edged or Negative edged

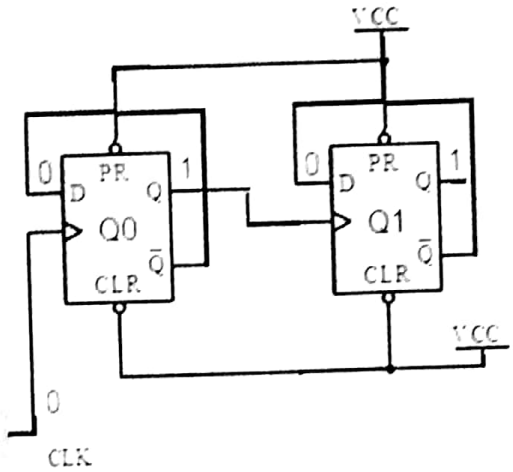
JK or D flip-flops

Count Direction: Up, Down, or Up/Down

Asynchronous counters are slower than synchronous counters because of the delay in the transmission of the pulses from flip-flop to flip-flop. With a synchronous circuit, all the bits in the count change synchronously with the assertion of the clock. Examples of synchronous counters are the Ring and Johnson counter.

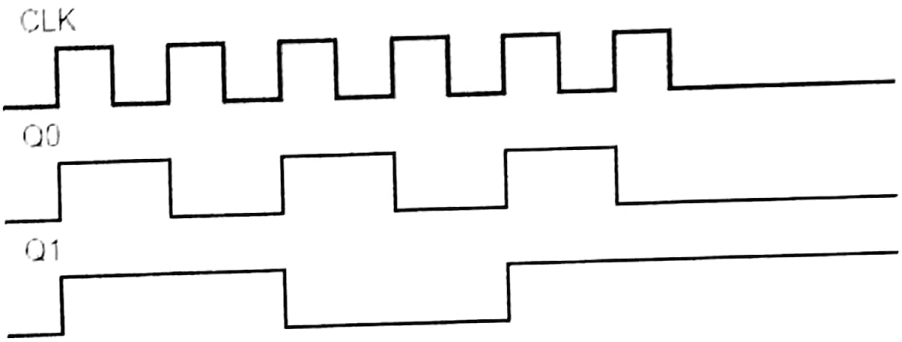
It can be implemented using D-type flip-flops or JK-type flip-flops.

The circuit below uses 2 D flip-flops to implement a divide-by-4 ripple counter ($2^n = 2^2 = 4$). It counts down.



STATE TABLE

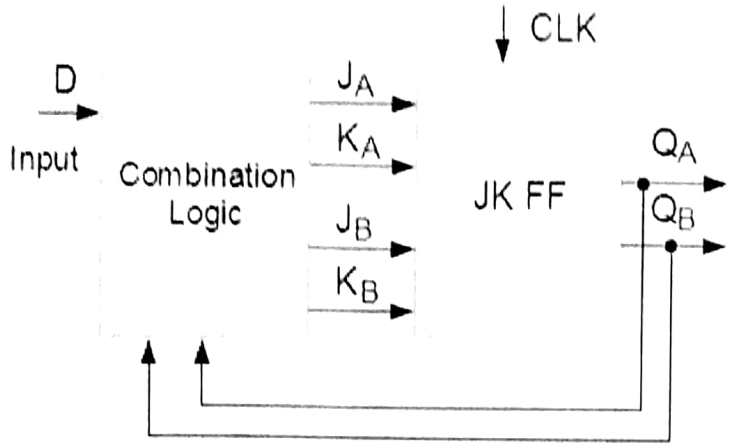
COUNT	Q1	Q0
3	1	1
2	1	0
1	0	1
0	0	0



Synchronous Counter Design

A finite-state machine determines its outputs and its next state from its current inputs and current state. A synchronous finite state machine changes state only when the appropriate clock edge occurs.

The following diagram shows a sequential circuit that consists of a combinational logic block and a memory block. For simplicity, we limit the design to one input and 2 JK flip flops. You will learn to derive the combination logic that meets the design specifications.

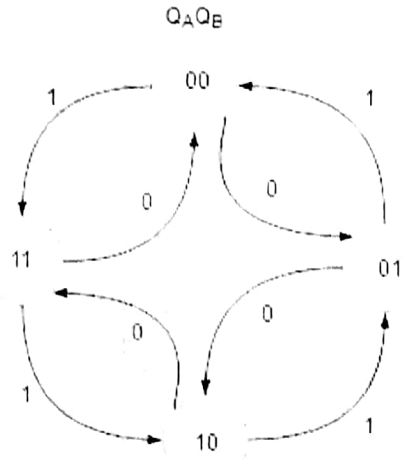


The steps to design a Synchronous Counter using JK flip flops are:

Let us describe a general sequential circuit in terms of its basic parts and its input and outputs.

Let us design a 2 bit up/down counter with an input D which determines the up/down function. Thus when D=0, the count sequence is 00,01,10,11,00 ... when D=1, the count sequence is 00,11,10,01,00 ...

Draw the state diagram for the given sequence.



Develop a next-state table for the specific counter sequence. Using the state diagram as a reference, fill up the present state and next state columns. For this interactive table, we can modify the next state.

Present State	Next State	JK flip flop inputs					

D	QA	QB	QA	QB	JA	KA	JB	KB
0	0	0	0	1	0	X	1	X
0	0	1	1	0	1	X	X	1
0	1	0	1	1	X	0	1	X
0	1	1	0	0	X	1	X	1
1	0	0	1	1	1	X	1	X
1	0	1	0	0	0	X	X	1
1	1	0	0	1	X	1	1	X
1	1	1	1	0	X	0	X	1

JK Flip Flop Truth Table

J	K	Q
0	0	Q_0
0	1	0
1	0	1
1	1	Toggle

JK Flip Flop Transition Table

QN	QN+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Use K-map to derive the logic equations.

JA = D QB + D QB

QAQB \ D	0	1
00	0	1
01	1	0
11	X	X
10	X	X

KA = D QB + D QB

QAQB \ D	0	1

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

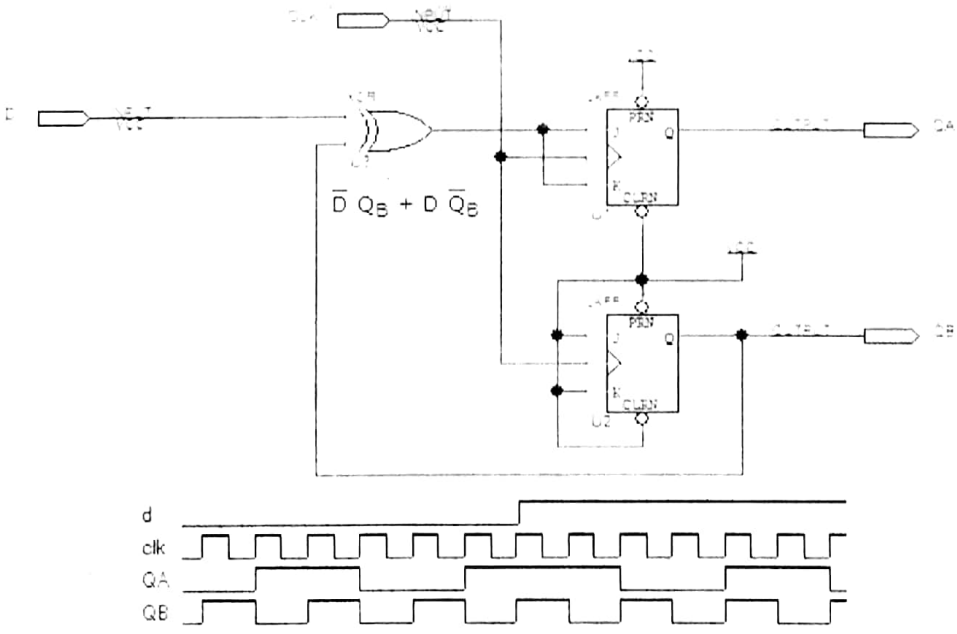
JB = 1

QAQB \ D	0	1
00	1	1
01	X	X
11	X	X
10	1	1

KB = 1

QAQB \ D	0	1
00	X	X
01	1	1
11	1	1
10	X	X

Use the boolean expressions to implement the counter



Shift register counters - Ring counter

A shift register can also be used as a counter. A shift register with the serial output connection back to the serial input is called Shift register counter.

There are 2 types of shift Register counters are:

i) Ring counter

ii) Johnson counter

Ring counter:

A ring counter is a circular shift register with only one flip flop being set at any particular time, all others are cleared.

A ring counter is a counter that counts up and when it reaches the last number that is designed to count up , it will reset itself back to the first number.

For example, a ring counter that is designed using 3 JK Flip Flops will count starting from 001 to 010 to 100 and back to 001. It will repeat itself in a 'Ring'

Johnson counters:

The Johnson counter is a K-bit switch-tail ring counter with 2k decoding gates to provide outputs for 2 k timing signals

A Johnson counter is a special case of shift register where, the output from the last stage is inverted and fed back as input to the first stage. A pattern of bits equal in length to the shift register thus circulates indefinitely. These counters are sometimes called "walking ring" counters.

It is used in specialist applications including those similar to the decade counter, digital to analogue conversion, etc.and thus the name Ring Counter.

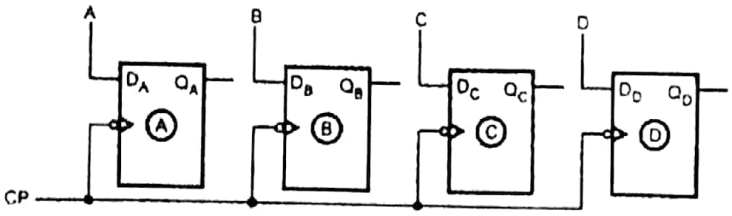
Design Of Registers - Buffer Register, Control Buffer Register, Shift Register, Bi-Directional Shift Register, Universal Shift Register:

Buffer Register:

Figure below shows the simplest register constructed with four D flip-flops. This register is also called buffer register. Each D flip-flop is triggered with a common negative edge clock pulse. The input X bits set up the flip-flops for loading.

Therefore, when the first negative clock edge arrives, the stored binary information becomes,

$Q_A Q_B Q_C Q_D$ = ABCD



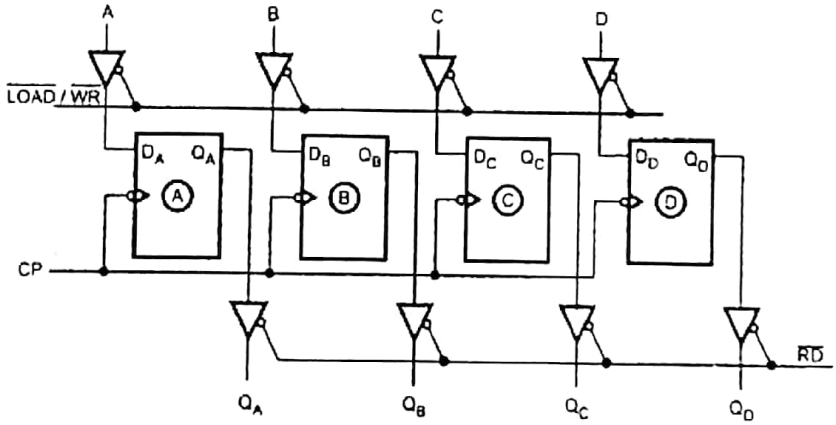
Buffer register:

In this register, four D flip-flops are used. So it can store 4-bit binary information. Thus the number of flip-flop stages in a register determines its total storage capacity.

Controlled Buffer Register:

We can control input and output of the register by connecting tristate devices at the input and output sides of register as shown in Figure below .So this register is called 'controlled buffer register'. Here, tristate switches are used to control the operation. When we want to store data in the register, we have to make LOAD or WR signal low to activate the tristate buffers.

When we want the data at the output, we have to make RD signal low to activate the buffers.



Controlled Buffer Register

Controlled buffer registers are commonly used for temporary storage of data within a digital system.

Shift registers:

A register capable of shifting its binary information in one or both directions from state to state within the register or into or out of the register upon application of clock pulses is called a shift register.

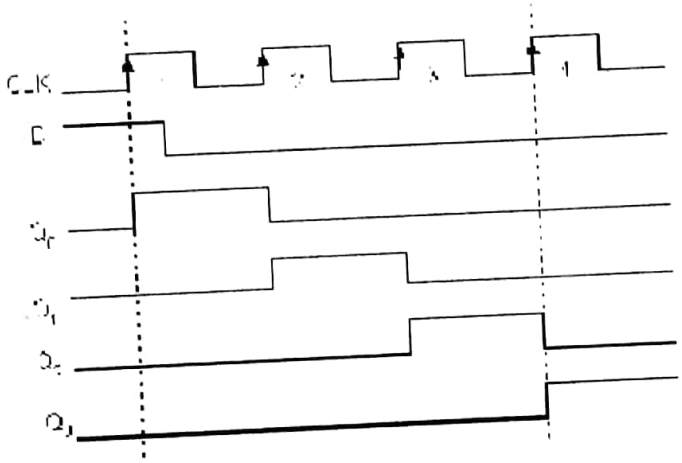
A register that is capable of shifting data, one bit at a time is called a shift register. One of the uses of a shift register is to convert between serial and parallel interfaces.

This is useful as many circuits work on groups of bits in parallel, but serial interfaces are simpler to construct.

Shift registers can be used as simple delay circuits and also as pulse extenders. A serial shift register consists of a chain of flip-flops connected in cascade with the output of one flip-flop being connected to the input of its neighbor.

The operation of the shift register is synchronous; thus each flip-flop is connected to a common clock.

D flip-flops forms the simplest type of shift-registers. The basic data movements possible within a four-bit shift register is shown in figure.



Timing diagram of shift register

The types of shift registers are:

1) Serial In Serial Out (SISO)

2) Serial In Parallel Out (SIPO)

3) Parallel In Serial Out (PISO)

4) Parallel In Parallel Out (PIPO)

Serial-In, Serial-Out:

Destructive Readouts

In destructive readout - each datum is lost once it has been shifted out of the right-most bit.

These are the simplest kind of shift register. The data string is presented at 'Data In', and is shifted right one stage each time 'Data Advance' is brought high.

At each advance, the bit on the far left (i.e. 'Data In') is shifted into the first flip-flop's output. The bit on the far right (i.e. 'Data Out') is shifted out and lost.

Non-destructive readout

Non-destructive readout can be achieved if another input line is added - the Read/Write Control.

When this is high (i.e. write) then the shift register behaves as normal, advancing the input data one place for every clock cycle and data can be lost from the end of the register.

However, when the R/W control is set low (i. e. read), any data shifted out of the register at the right becomes the next input at the left and is kept in the system.

Therefore, as long as the R/W control is set low, none of the data can be lost from the system.

Serial-In, Parallel-Out: This configuration allows conversion from serial to parallel format.

Data are input serially and once the data has been input, it may be either read off at each output simultaneously or it can be shifted out and replaced.

Parallel-In, Serial-Out:

This configuration has the data input in parallel format. To write the data to the register, the Write/Shift control line must be held LOW.

To shift the data, the W/S control line is brought HIGH and the registers are clocked.

As long as the number of clock cycles is not more than the length of the data-string, the Data Output, Q, will be the parallel data read off in order.

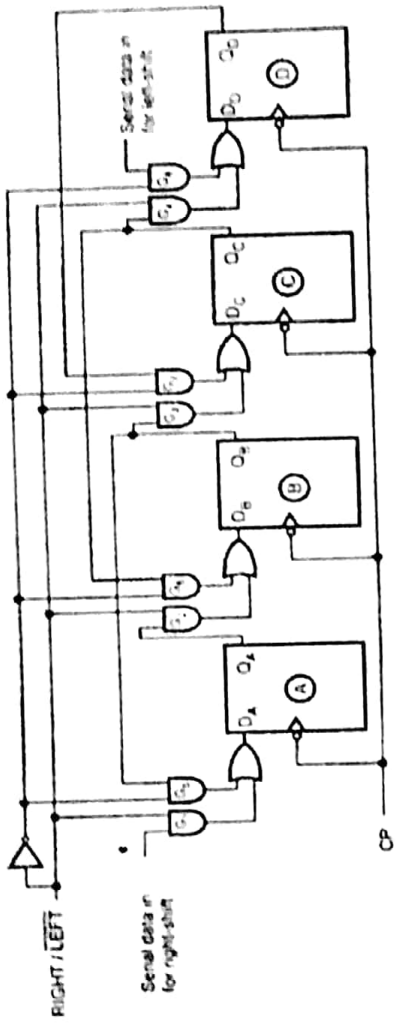
Parallel-In, Parallel-Out:

This kind of shift register takes the data from the parallel inputs (D0-D3) and shifts it to the corresponding output (Q0-Q3) when the registers are clocked.

It can be used as a kind of 'history ' retaining old information as the input in another part of the system until ready for new information, where upon the registers are clocked and the new data are 'let through'.

Bidirectional Shift Register:

This type of register allows shifting of data either to the left or to the right side. It can be implemented by using logic gate circuitry that enables the transfer of data from one stage to the next stage to the right or to the left, depending on the level of a control line. Figure below shows a four-bit bidirectional register.



4-bit bi-directional shift register

The RIGHT/LEFT is the control input signal which allows data shifting either towards right or towards left. A high on this line enables the shifting of data towards right and a low enables it towards left. When RIGHT/LEFT signal is high, gates GI, G2, G3, G4 are enabled. The state of the Q output of each flip-flop is passed through the D input of the following flip-flop.

When a clock pulse arrives, the data are shifted one place to the right. When the RIGHT/LEFT signal is low, gates G5, Gs, G7, G5 are enabled. The Q output of each flip-flop is passed through the D input of the preceding flip-flop. When clock pulse arrives, the data are shifted one place to

the left. Bidirectional Shift Register with Parallel Load We have seen that shift register can be used for converting serial data into parallel data, and vice-versa.

When parallel load capability is added to the shift register, the data entered in parallel can be taken out in serial fashion by shifting the data stored in the register. Such a register is called bidirectional shift register with parallel load. Figure below shows bidirectional shift register with parallel load.

As shown in the Figure (b) , the D input of each flip-flop has three sources :

Output of left adjacent flip-flop

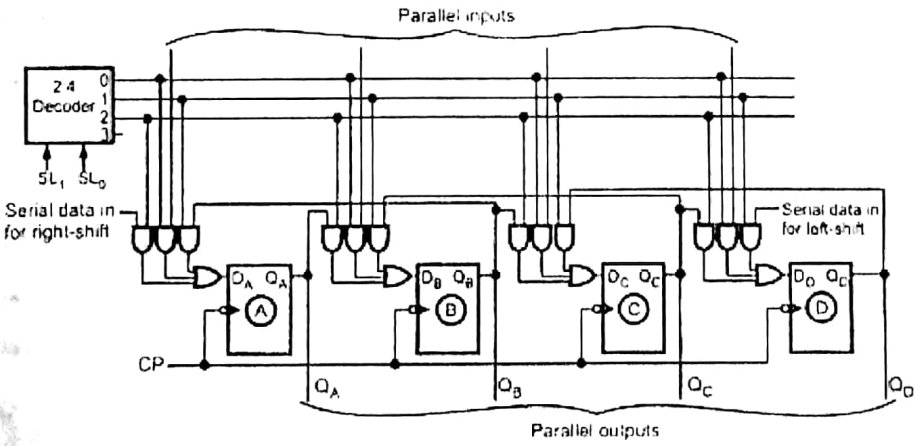
output of right adjacent flip-flop and

Parallel input

Out of these three sources one source is selected at a time and it is done with the help of decoder. The decoder select lines (SL_1 and SL_0) select the one source out of three as shown in the table below.

SL_1	SL_0	Selected Source
0	0	Parallel input
0	1	Output of right adjacent FF
1	0	Output of left adjacent FF
1	1	

When select lines are 00 (i.e. $SL_1 = 0$ and $SL_0 = 0$), data from the parallel inputs is loaded into the 4-bit register. When select lines are 01 (i.e. $SL_1 = 0$ and $SL_0 = 1$), data within the register is shifted 1-bit left. When select lines are 10 (i.e. $SL_1 = 1$ and $SL_0 = 0$), data within the register is shifted 1-bit right.

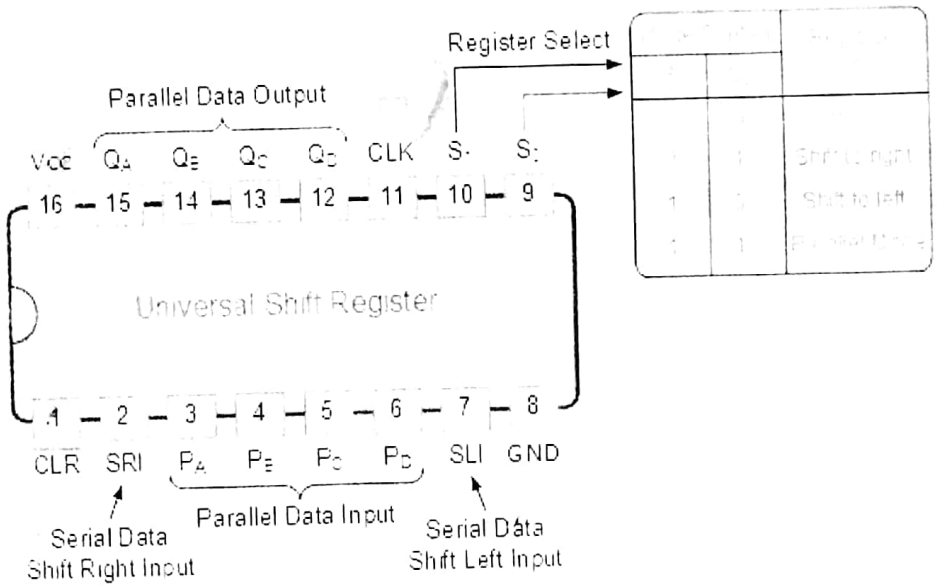


(b) 4-bit bi-directional shift register with parallel load

Universal shift registers:

Today, there are many high speed bi-directional "universal" type Shift Registers available such as the TTL 74LS194, 74LS195 or the CMOS 4035 which are available as 4-bit multi-function devices that can be used in either serial-to-serial, left shifting, right shifting, serial-to-parallel, parallel-to-serial or as a parallel-to-parallel multi-function data register, hence the name "Universal".

These universal shift registers can perform any combination of parallel and serial input to output operations but require additional inputs to specify desired function and to pre-load and reset the device. A commonly used universal shift register is the TTL 74LS194.



4-bit Universal Shift Register 74LS194

Universal shift registers are very useful digital devices. They can be configured to respond to operations that require some form of temporary memory storage or for the delay of information such as the SISO or PIPO configuration modes or transfer data from one point to another in either a serial or parallel format.

Universal shift registers are frequently used in the arithmetic operations to shift data to the left or right for multiplication or division.

Sequential circuits - II

Finite state Machine:-

→ Finite state Machine is an Abstract model describing the synchronous sequential Machine.

→ the behaviour of FSM is described as a sequence of events that occur at discrete instants designated as $t = 1, 2, 3 \ldots$ etc.

→ the machine has been receiving i/p signal & also responding by producing o/p signals.

→ if, now at time t, we apply i/p signal $x(t)$ to machine, the response $z(t)$ would depend on $x(t)$ as well as on the past i/p's to the machine & since a given machine might have infinite varieties of possible histories. it would need infinite capacity for storing them

→ In practice, it is impossible to store have infinite storage capabilities,

→ So we will concentrate on those machines whose past histories can affect their future behaviour only in a finite no of ways. These are called Finite state Machines ie, machine with fixed no of states

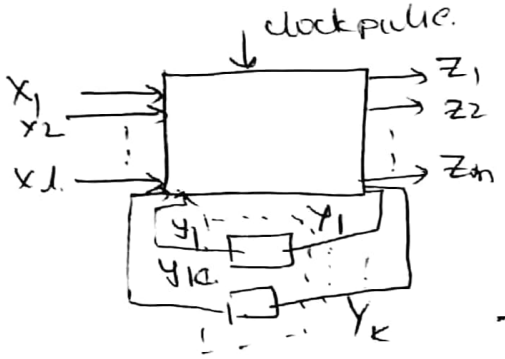


Fig 1. Block diagram of finite state model.

→ schematic diagram of synchronous sequential machine.

→ the ckt have a finite no of 'l' i/p's & have set $(x_1, x_2, \ldots x_l)$ i/p variables

$P = 2^l$ (possible combination - P)

→ the ckt has finite no of m o/p terminals with set of $(Z_1, Z_2 \ldots Z_m)$ o/p variables.

$q = 2^m$ (possible combination - q)

output $O = [O_1, O_2 \ldots O_q]$

→ signal value at o/p is state variable $y_1, y_2 \ldots y_k$

→ $x_1 \ldots x_l$, $y_1 \ldots y_k$ are i/p's o/p $z_1 \ldots z_m$, & $Y_1, Y_2 \ldots Y_k \ldots$

→ The synchronous & clocked sequential ckts are represented by two models

1. Moore circuit :- In this model the o/p depends only on present state of the flipflops.
2. Mealy circuit :- the o/p depends on both the present state of the flipflop(s) & input(s)

Analysis of clocked sequential circuits:

→ It is the process of determining the functional Relation that exists b/n its o/p's, its i/p's & its internal states.

→ The contents of all the flipflops in the ckt combined determine the internal state of the circuit.

→ If the ckt contains n flops, it can be in one of the '2' states. Knowing the present state of the circuit and the i/p values at any time t, we should be able to derive its next state & the o/p produced by the circuit at t.

→ A sequential circuit can be described completely by a state table that is very similar to the ones shown for flipflops

→ For a circuit with n flipflops, 2^n rows in the statetable.

→ If there are m i/p's to the circuit, there will be 2^m columns in the state table.

→ A state diagram is graphical representation of state table, in which each state is represented by a circle and the state transitions are represented by arrows b/n the circles.

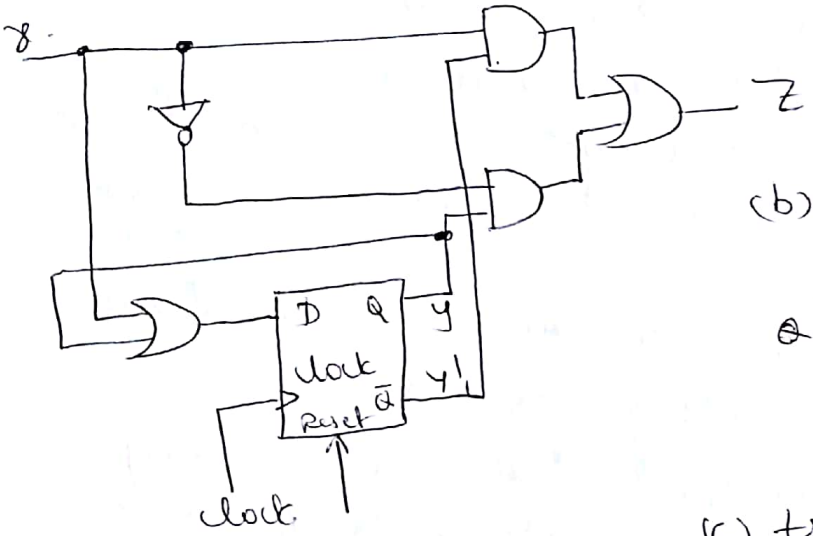
→ The i/p combination that brings about the transition and the corresponding o/p information are shown on the arrow.

→ Analysis of a sequential circuit thus corresponds to generating the state table of the state diagram for the circuit.

→ The state table or state diagram can be used to determine the o/p sequence generated by the circuit for a len i/p sequence if the initial state is known.

→ It is important to note that for proper operation, a sequential circuit must be in its initial state before the i/p's to it can be applied.

→ usually power-up clrs are used to initialize the circuit to the appropriate state when the power is turned on



(a) circuit

Q \ x	0	1
0	0	1
1	1	1

Q(t+1)

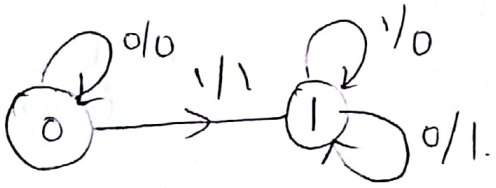
Q \ x	0	1
0	0	1
1	1	0

Z

(b) next state output table:

Q \ x	0	1
0	0/0	1/1
1	1/1	1/0

(c) transition table:

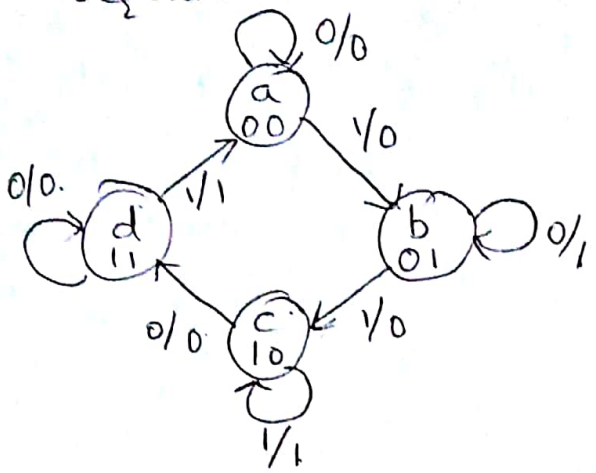


(d) state diagram

State Diagram:-

→ State Diagram or state graph is a pictorial representation of the relationships b/n the present state, i/p, & next state and the o/p of sequential circuit ie, the state Diagram

pictorial representation of the behaviour of a sequential circuit.



(a) state Diagram

PS	NS, OP Input x	
	x=0	x=1
a	a,0	b,0
b	b,1	c,0
c	d,0	c,1
d	d,0	a,1

(b) state table

Fig :- delay circuit.

Fig(a) shows the state diagram of a delay circuit.

→ The state is represented by a circle also called the node & vertex and the transition b/n states is indicated by directed lines connecting the circles

→ A directed line connecting circle with itself indicates that the next state is the same as present state.

→ the binary number inside each circle identifies the state represented by the circle.

→ The directed lines are labelled with two binary numbers separated by a symbol /. The i/p value that causes state transition is labelled 1st & the o/p value that occurs when the i/p is applied during the present state is labelled after the symbol /.

→ In the case of a moore circuit, the directed lines are labelled with only one binary number representing the i/p that causes the state transition.

→ The o/p is indicated within the circle below the P.S, because the o/p depends only on P.S and not on the i/p.

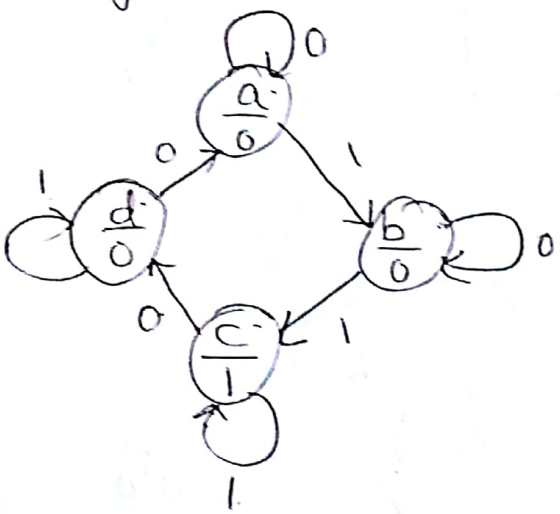
State table :-

→ eventhough the behaviour of sequential ckt can be conveniently described using a state diagram, for its implementation the information contained in the state diagram is to be translated into a state table.

→ the state table is a tabular representation of the state diagram.

→ the P.S designates the state of the flip-flops before the application of clock pulse.

→ the next state is the state of the flipflops after the application of the clock pulse & the o/p section gives the value of o/p variable during the P.S



(a) State Diagram

PS	NS, input x: x=0	x=1	o/p
a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	0

(b) State table.

→ Both state diagram, state table contain the same information & the choice b/n two representations is a matter of convenience.

→ Both having the advantages of precise, unambiguous, and thus, more suitable for describing the op'n of sequential machine than that by verbal description

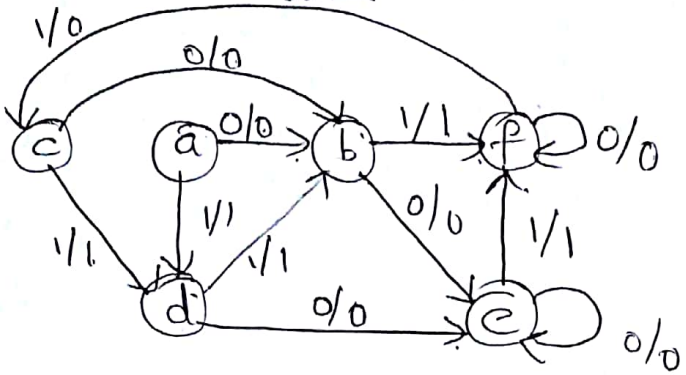
Reduction of State tables :-

→ it avoids the introduction of Redundant states.

→ the Reduction in redundant state reduces the no of flipflops & logic gates, reducing the cost of final ckt.

→ Two states are said to be equivalent if every possible set of i/p's generate exactly the same o/p and the same next state.

→ When two states are equivalent, one of them can be removed without altering the i/p-o/p relationship



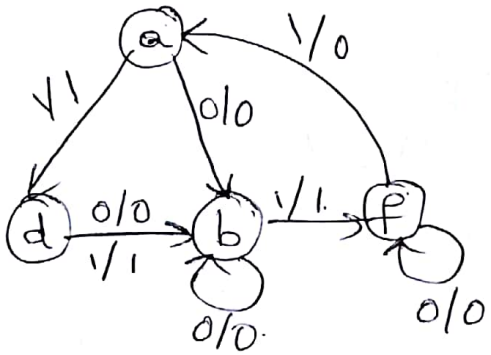
(a) state diagram

PS	NS		o/p	
	x=0	x=1	x=0	x=1
a	b	d	0	1
b	e	f	0	1
c	b	d	0	1
d	e	b	0	1
e	e	f	0	1
f	f	c	0	0

(b) state table

PS	NS		o/p	
	x=0	x=1	x=0	x=1
a	b	d	0	1
b	b	f	0	1
d	b	b	0	1
f	f	a	0	0

(a) state table.



(b) state diagram.

State Assignment

→ The process of assigning binary values to the states of the sequential machine is known as state Assignment.

→ The binary values are to be assigned to the states in such a way that it is possible to implement flipflop i/p function using minimum logic gates

→ The o/p values of the physical device are referred as state variables

→ Design procedure:

→ The word description of the ckt behavior is stated. The may be accompanied by a state diagram, a timing diagram & other pertinent information

→ from the given information about the circuit, obtain the state table

→ the no of states may be reduced by state-reduction methods if the sequential circuit can be characterized by i/p-o/p relationships independent of no of states.

→ Assign binary values to each state

→ Determine the no of flipflops needed and assign a letter symbol to each

→ choose the type of flipflop to be used

→ from state table, derive the circuit excitation & o/p tables

→ using the map or any other simplification method, derive the circuit o/p functions and the flipflop i/p functions

→ Draw the logic diagram.

→ Realization of circuits using various flipflops:-

Let us consider that the following sequence is to be realized by a counter consisting of 3 JK flipflops

(e.g.)

A_1	0	0	0	0	1	1	0
A_2	0	1	1	0	0	1	0
A_3	0	1	0	1	1	0	0

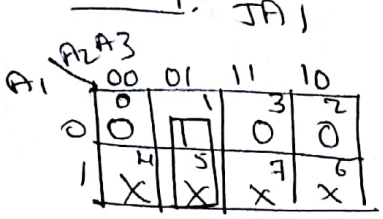
Here we will design the counter.

Solution:

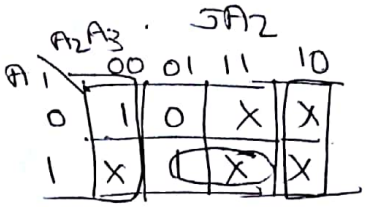
Truth table:

present state			Next state			Flip-flop inputs					
A_1	A_2	A_3	A_1	A_2	A_3	JA_1	KA_1	JA_2	KA_2	JA_3	KA_3
0	0	0	0	1	1	0	X	1	X	1	X
0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	0	1	1	0	1	1	X	0	X	X	0
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

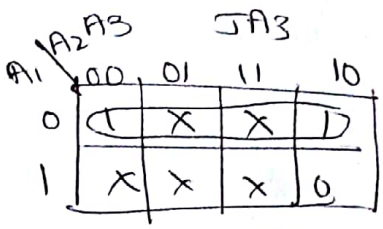
K-map:



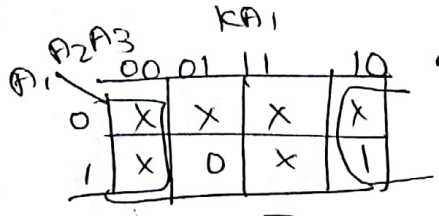
$JA_1 = \bar{A}_2 A_3$



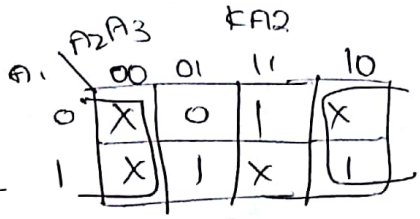
$JA_2 = A_3 + A_1 A_2$



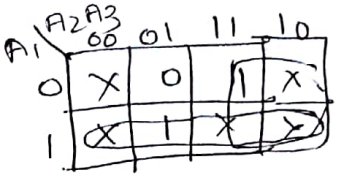
$JA_3 = \bar{A}_1 \; KA_3$



$KA_1 = \bar{A}_3$

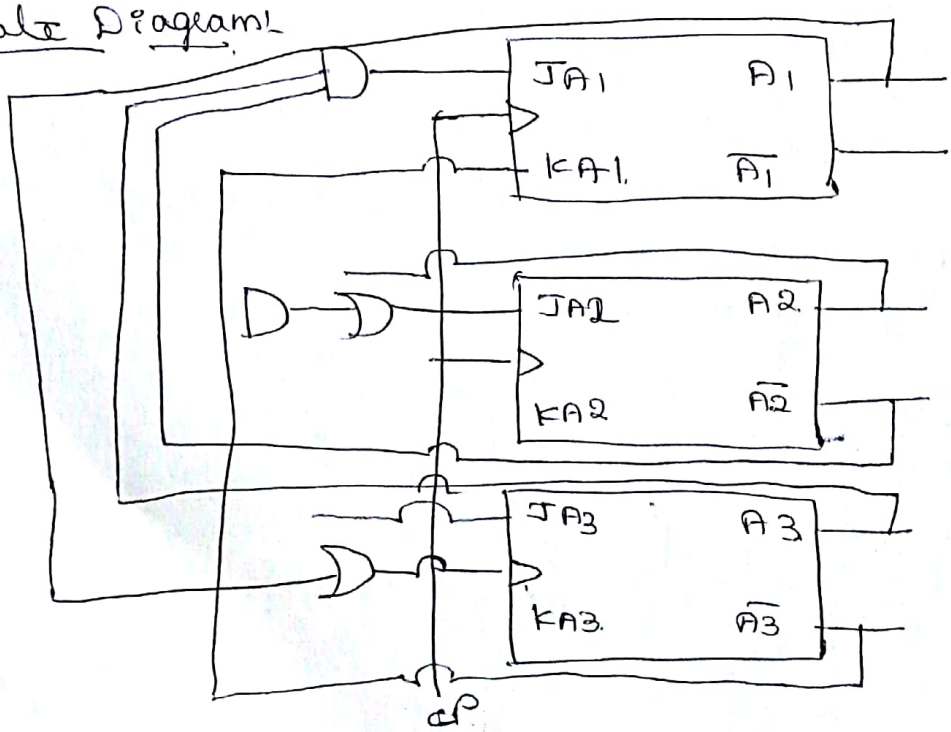


$KA_2 = \bar{A}_3$



$KA_3 = A_1 + A_2$

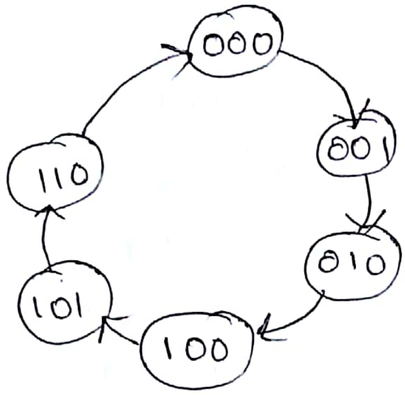
State Diagram:



→ Let us Design the synchronous sequential circuit using JK flip-flop for 0,1,2,4,5,6

0 → 1 → 2 → 4 → 5 → 6

State Diagram



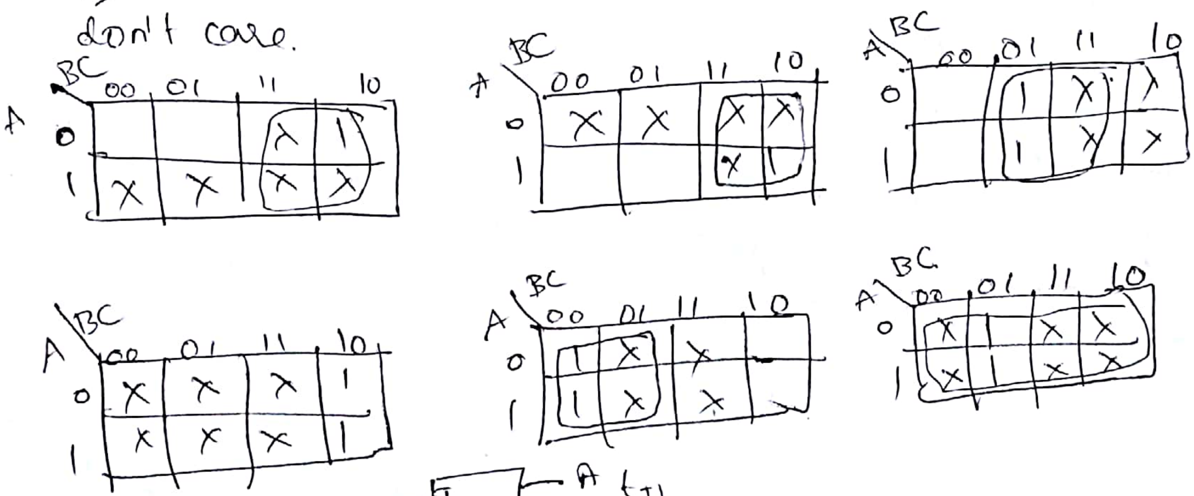
present state			next state		
A_t	B_t	C_t	A_{t+1}	B_{t+1}	C_{t+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

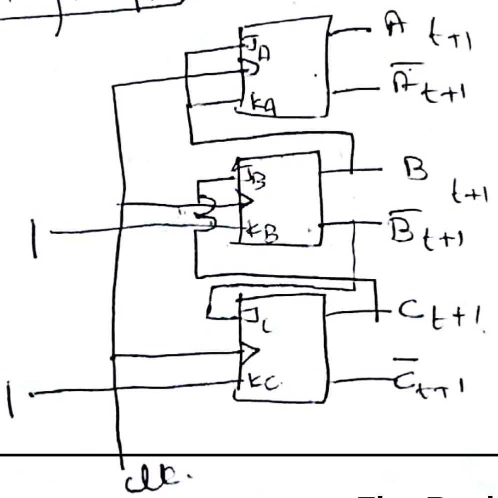
J_A	K_A	J_B	K_B	J_C	K_C
0	X	0	X	1	X
0	X	1	X	X	1
1	X	X	1	0	X
X	0	0	X	1	X
X	0	1	X	X	1
X	1	X	1	0	X

JK excitation table

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

3, 7 are to be considered as don't care.





Relay to Mode conversion and vice versa:

→ A synchronous sequential circuit is a system whose behaviour can be defined from the knowledge of its signals at discrete instants of time

Secondary variables

→ The delay elements are used to provide a short term memory for the sequential circuit. The P.S & N.S variables in asynchronous sequential circuits are called secondary variables

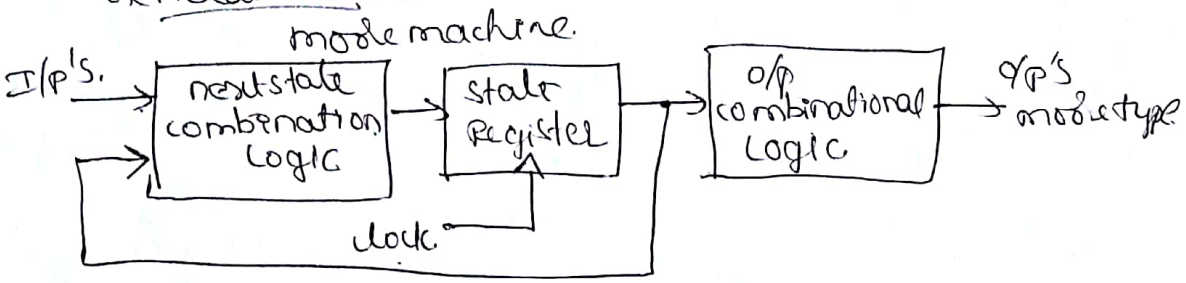
Present state and Next state

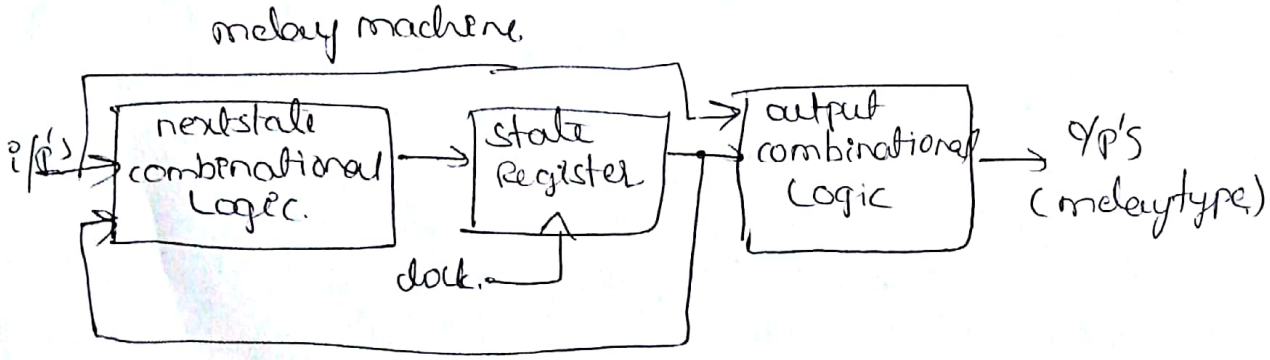
→ The information stored in the memory elements at any given time defines the P.S of sequential ckt

→ The P.S and the external i/p's determine the O/ps and N.S of the sequential circuits.

General model, classification and Design

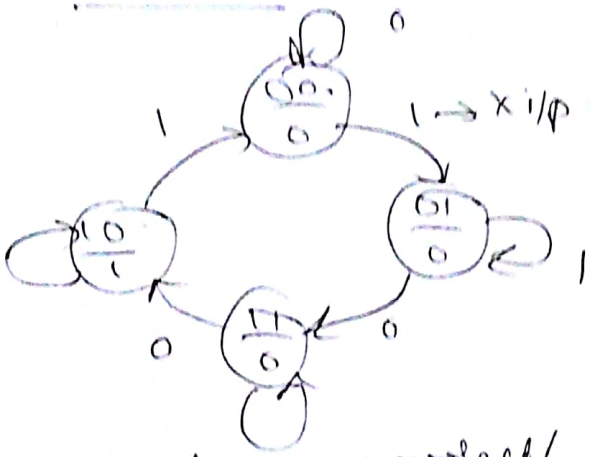
General model:

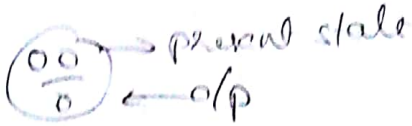




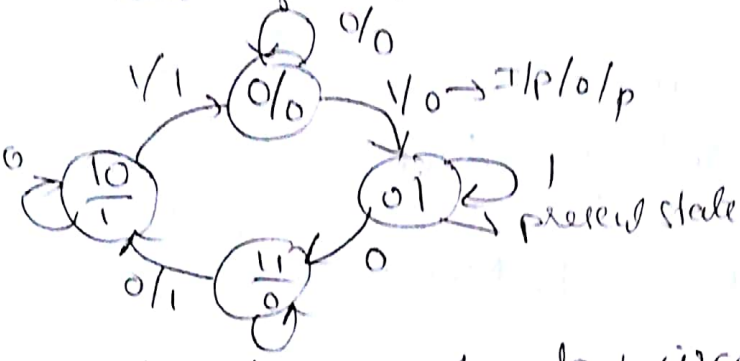
mode and delay design.

State transition diagram of mealy and moore sequence detector model





State diagram of moore ckt

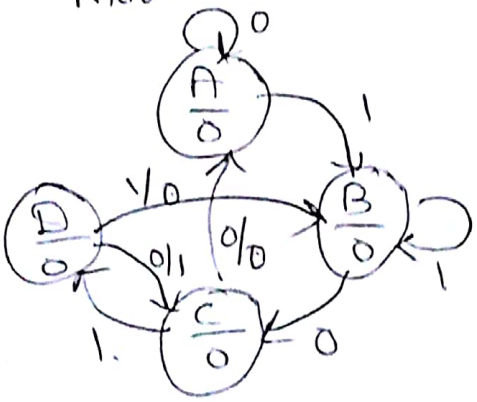


State diagram of mealy circuit.

Problem:

Here we see an example to draw the state transition diagram of a sequence detector circuit that detects '1010' from i/p data stream using moore model, mealy model

Ans. moore model.



mealy model