Chapter – 6

FLIPFLOPS

6.1 Sequential Circuit

The logic circuits whose outputs at any instant of time depend not only on the present inputs but also on past outputs are called sequential circuits. A sequential circuit consists of a combinational circuit to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information.

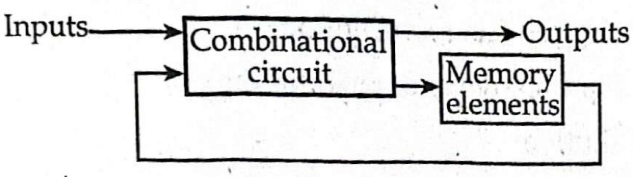


Fig.: Block diagram of a sequential circuit

Sequential circuit is slower in operation than combinational circuit. It may or may not contain clock input.

There are two types of sequential circuits:

i. **Synchronous sequential circuit**

It is a system whose behavior can be defined from the knowledge of its signals at discrete instant of time (i.e., by the clock signal parallelly driven by one clock signal).

ii. **Asynchronous sequential circuit**

It is a system whose behavior depends in the order in which its input signals change and can be affected at any instant of time (one's output is given to clock of another).

6.2 Flipflop

A flipflop is a sequential circuit which is popularly known as a basic digital memory circuit. A flipflop stores 1 bit, therefore, it is called as 1-bit memory cell. It has two stable states: logic 1 and logic 0. It can flip from one state to another and then flop back, so it is called a flipflop and also known as a bistable multivibrator. The outputs of circuit (Q and $\bar{Q}$) will always be complementary. This means that if Q.= 0, then $\bar{Q}$ = 1 and vice-versa. They will never be equal ; Q = $\bar{Q}$ = 0 or 1 is an invalid state. If Q = 1, $\bar{Q}$ = 0, it is

called 1 state or SET state. If $Q = 0, \bar{Q} = 1$, it is called 0 state or RESET state. If the circuit is in reset state, then it continues to be in reset state and if it is in set state, then it continues to be in set state. This characteristic of the circuit illustrates that it can store 1 bit of digital information.

Application of flipflops include:

1. As a memory element
2. In various types of registers
3. In counters/timers
4. As a delay element

The different types of flipflop are explained below:

i. SR flipflop (set reset flipflop)
ii. D flip flop (delay or data flipflop)
iii. JK flipflop
iv. T flipflop (toggle flipflop)
v. Master slave flipflop

6.3 SR Flipflop

SR flipflop has two inputs namely SET (S) and RESET (R) and two outputs Q and $\bar{Q}$. It can be implemented using NOR and NAND gates. The flipflop is often called a *latch* since it will hold, or latch, in either stable state.

NOR-based SR latch

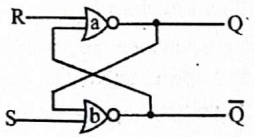


Fig.: NOR latch

Truth Table

S	R	Q	$\bar{Q}$	Remarks
1	0	1	0	Set
0	0	1	0	No change
0	1	0	1	Reset
0	0	0	1	No change
1	1	?	?	Invalid/ Forbidden

Case I: S = 1, R = 0

As S = 1, output of NOR gate b i.e., $\bar{Q} = 0$. This makes both inputs of NOR gate a at 0, therefore Q = 1. Now, both inputs of NOR gate b are 1. Hence, for S = 1, R = 0, Q = 1 and $\bar{Q} = 0$ (Set condition).

Case II: S = 0, R = 1

As R = 1, the output of NOR gate a i.e., Q = 0. This makes both inputs of NOR gate b are 0, therefore $\bar{Q} = 1$. Now, both inputs of NOR gate a are 1. Therefore, for S = 0, R = 1, Q = 0, $\bar{Q} = 1$ (Reset condition).

Case III: S = 0, R = 0

We know that the output of NOR gate a i.e., $Q = \overline{R + \bar{Q}}$

As R = 0, $Q = \overline{0 + \bar{Q}} = \bar{\bar{Q}} = Q$

Again the output of NOR gate b i.e., $\bar{Q} = \overline{S + Q}$

As S = 0, $\bar{Q} = \overline{0 + Q} = \bar{Q}$

Therefore, for S = 0, R = 0, Q and $\bar{Q}$ do not change their state.

Case IV: S = 1, R = 1

If S = 1, R = 1 then the outputs Q and $\bar{Q}$ both are forced to 0. Actually this is indeterminate state which must be avoided (since $Q = \bar{Q}$ which violates the basic requirements of flip–flop i.e., Q and $\bar{Q}$ must be complement of each other).

NAND-based SR latch

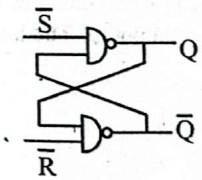


Fig.: NAND latch

Truth Table

$\bar{S}$	$\bar{R}$	Q_{n+1}	Remarks
0	1	1	Set
1	1	1	No change
1	0	0	Reset
1	1	0	No change
0	0	?	Forbidden

Case I: $\bar{S} = 0, \bar{R} = 0$

When any input of NAND gate is 0, the output is forced to 1. Here $\bar{S} = \bar{R} = 0$, therefore, Q and $\bar{Q}$ are forced to be equal to 1 which is indeterminate and must be avoided.

Case II: S = 0, R = 1 (i.e., $\bar{S} = 1, \bar{R} = 0$)

As $\bar{R} = 0$, output of NAND gate b i.e., $\bar{Q} = 1$. This makes both inputs of NAND gate a at 1, therefore, Q= 0. Hence, for S = 0, R = 1, the output Q = 0 and $\bar{Q} = 1$ (Reset condition).

Case III : S = 1, R = 0 (i.e., $\bar{S} = 0, \bar{R} = 1$)

As $\bar{S} = 0$, output of NAND gate a i.e., Q = 1. This makes both inputs of NAND gate b at 1, therefore, $\bar{Q} = 0$. Hence, for S = 1, R = 0, the output Q = 1 and $\bar{Q} = 0$ (Set condition).

Case IV: S = 0, R = 0, (i.e., $\bar{S} = 1, \bar{R} = 1$)

The output of NAND gate a, $Q = \overline{\bar{S}.\bar{Q}} = \overline{1.\bar{Q}} = \bar{\bar{Q}} = Q$

The output of NAND gate b, $\bar{Q} = \overline{\bar{R}.Q} = \overline{1.Q} = \bar{Q}$

Thus, there is no change in the output if $\bar{S} = \bar{R} = 1$.

6.4 Gated Flipflop

1) Gated RS Flipflop

The addition of two AND gates at the R and S inputs will result in a flipflop that can be enabled or disabled. When the ENABLE input is HIGH, information at the R and S will be transmitted directly to the outputs. The latch is said to be enabled. When the ENABLE input is LOW, the AND gate outputs must both be low and changes in neither R nor S will have any effect on the flipflop output Q. The latch is said to be disabled.

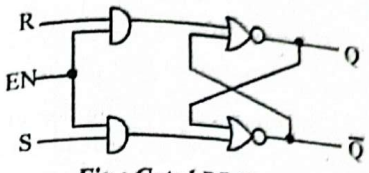


Fig.: Gated RS flipflop (NOR)

Truth Table

EN	S	R	Q_{n+1}
0	×	×	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Invalid

When two inputs signal R and S are applied to the circuit, there is a time delay between these two signals, which may lead to a wrong output result. In order to overcome this problem of unequal propagation delay time of input signals, the gated flipflop is used.

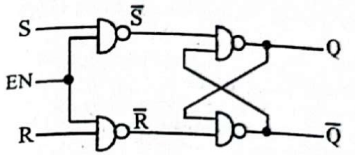


Fig.: Gated RS flipflop (NAND)

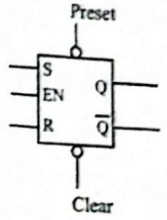
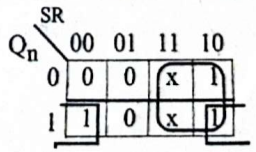


Fig.: Logic symbol

The characteristic table of gated RS flipflop is given below.

Q_n	S	R	Q_{n+1}	Remarks
0	0	0	0	No change
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	×	Invalid
1	0	0	1	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	×	Invalid



$Q_n + 1 = S + Q_n R$

This is the characteristic equation.

2) Gated D Flipflop (Delay or Data Flipflop)

The data flipflop has only one input called data (D) input and two outputs Q and $\bar{Q}$. It can be constructed from SR flipflop by inserting an inverter between S and R and assigning the symbol D to S input. The output Q is same as D input.

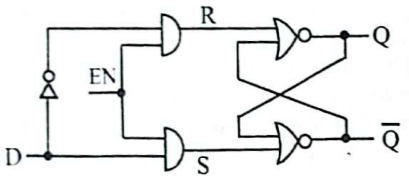


Fig. Logic diagram of a gated D flipflop

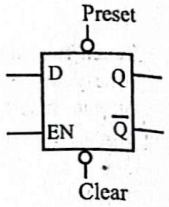


Fig.: Logic symbol

Characteristics table:

Q_n	D	Q_{n+1}	Remarks
0	0	0	same as D
0	1	1	same as D
1	0	0	same as D
1	1	1	same as D

Q_n \ D_o	0	1
0	0	1
1	0	1

$Q_{n+1} = D$

Advantages of D flipflop over RS flipflop:

In some events, both inputs become high in RS flipflop which is undesirable condition. This drawback of RS flipflop is overcome in D flipflop. There is only one input i.e., D which drive the flipflop.

3) Gated JK Flipflop

It is the refinement of RS flipflop as the indeterminate state (or invalid state) of RS is defined as 'Toggle' in JK. A JK flipflop can be constructed using two cross coupled NOR gates and two AND gates as shown in below figure.

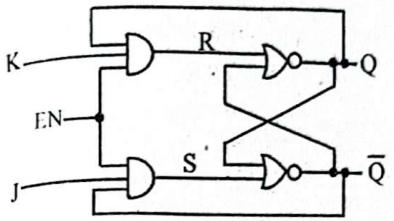


Fig.: JK flip - flop

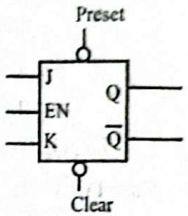


Fig.: Logic Symbol

Case I: J = 0, K = 0

If J = 0, K = 0, the output of two AND gates i.e., R and S are equal to 0. We know that if R = S = 0, the output Q and $\bar{Q}$ i.e., $Q_{n+1} = Q_n$ and $\overline{Q_{n+1}} = \overline{Q_n}$

Case II: J = 0, K = 1

(i) **Provided that Q = 1 and $\bar{Q}$ = 0**

$S = EN.J.\bar{Q} = 1.0.0 = 0$ $\quad R = EN.K.Q = 1.1.1 = 1$

i.e., $Q_{n+1} = 0$

(ii) **Provided that, Q = 0 and $\bar{Q}$ = 1**

$S = EN.J.\bar{Q} = 1.0.1 = 0$ $\quad R = EN.K.Q = 1.1.0 = 0$

i.e., remain in reset state, $Q_{n+1} = 0$

Case II: J = 1, K = 0

(i) **Provided that Q = 1 and $\bar{Q}$ = 0**

$S = EN.J.\bar{Q} = 1.1.0 = 0$ $\quad R = EN.K.Q = 1.0.1 = 0$

i.e., $Q_{n+1} = 1$

(ii) Provided that Q = 0 and $\bar{Q}$ = 1

S = EN.J.$\bar{Q}$ = 1.1.1 = 1

R = EN.K.$\bar{Q}$ = 1.0.0 = 0

i.e., $Q_{n+1} = 1$

Case IV: J = 1, K = 1

(i) Provided that Q = 1 and $\bar{Q}$ = 0

S = EN.J.$\bar{Q}$ = 1.1.0 = 0

R = EN.K.Q = 1.1.1 = 1

i.e., $Q_{n+1} = 0$

(ii) Provided that Q = 0 and $\bar{Q}$ = 1

S = EN.J.$\bar{Q}$ = 1.1.1 = 1

R = EN.K.Q = 1.1.0 = 0

i.e., Qn+1 = 1

The outputs are inverted i.e., toggle from reset to set [Q = 0 to Q = 1]

Characteristics table:

Q_n	J	K	Q_{n+1}	Remarks
0	0	0	0	No change
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	1	Toggle
1	0	0	1	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	0	Toggle

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

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

Gated Toggle Flipflop (T flipflop)

It is made by shorting the J and K input of JK flipflop to single input 1. The output toggles each time for T = 1.

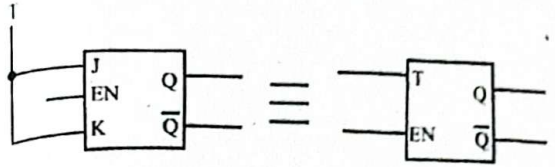


Fig: Logic symbol

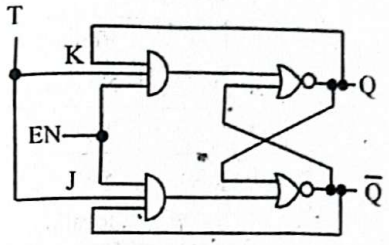


Fig.: Logic diagram of T flipflop

Characteristic table:

Q_n	T	Q_{n+1}	Remarks
0	0	0	No change
0	1	1	Toggle
1	0	1	No change
1	1	0	Toggle

Q_n \ T	0	1
0	0	1
1	1	0

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

$= (Q_n \oplus T)$

6.5 Edge Triggered Flipflop

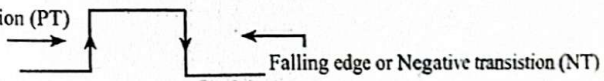


Fig.: clock

Everything is same as gated (or level triggered), only the difference is the clock pulse type which makes the flip–flop enable. Another is the symbol given for edge triggered.

1) Edge Triggered RS Flipflop

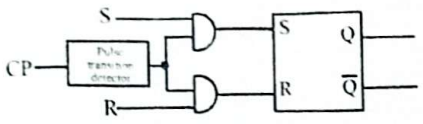


Fig.: Logic diagram of RS flipflop

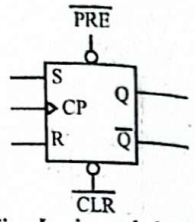


Fig.: Logic symbol of positive edge triggered RS flipflop

Truth Table

CP	S	R	Q_{n+1}	Remarks
↑	0	0	Q_n	No change
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	?	Invalid

Similarly, for negative edge triggered RS flipflop, the logic symbol is

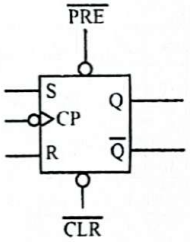


Fig.: Logic symbol for negative edge triggered RS flipflop

2) Edge Triggered D Flipflop

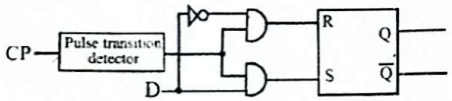


Fig.: Logic diagram of D flipflop

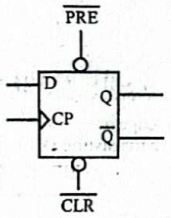


Fig.: Logic symbol of positive edge triggered D flipflop

Truth Table

CP	D	Q_{n+1}
0	×	Q_n
↑	0	0
↑	1	1

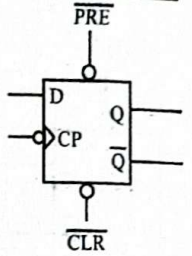


Fig.: Logic symbol of negative edge triggered D flipflop

3) Edge Triggered JK Flipflop

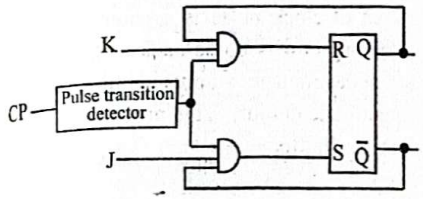


Fig.: Logic diagram of JK flipflop

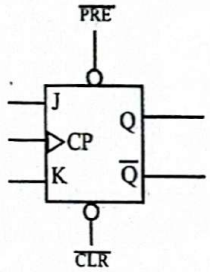


Fig.: Logic symbol of positive edge triggered JK flipflop

CP	J	K	Q_{n+1}
↑	0	0	Q_n (Last state)
↑	1	0	0 (Reset)
↑	0	1	1 (Set)
↑	1	1	$\overline{Q}$ (Toggle)

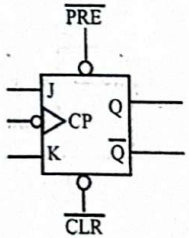


Fig.: Logic symbol of negative edge triggered JK flipflop

Asynchronous Inputs (Direct Inputs)

PRESET and CLEAR are called asynchronous inputs because they activate the flipflop independent of clock. It may be active low or high

If $\overline{CLR}$ is LOW, then the output is reset. If $\overline{PRE}$ is LOW, then the output is set

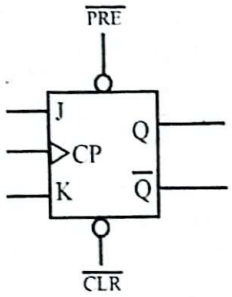


Fig.: Asynchronous inputs in JK flipflop

Race Around Condition

In JK flipflop, if the duration of clock pulse is greater than delay between input and output, the output may come back as the input to the flipflop and may result indeterminate outputs within the same clock pulse (instead of a single state output). This problem is known as "race around condition" in JK flip flop.

4) **Master Slave Flipflop**

Master slave flipflop is required to remove the race around condition. Master slave flipflop consists of two flipflops; one serve as master and another as slave. Master is positive edge triggered and slave is negative edge triggered.

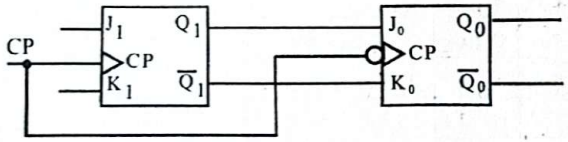


Fig.: Logic symbol of master slave JK flipflop

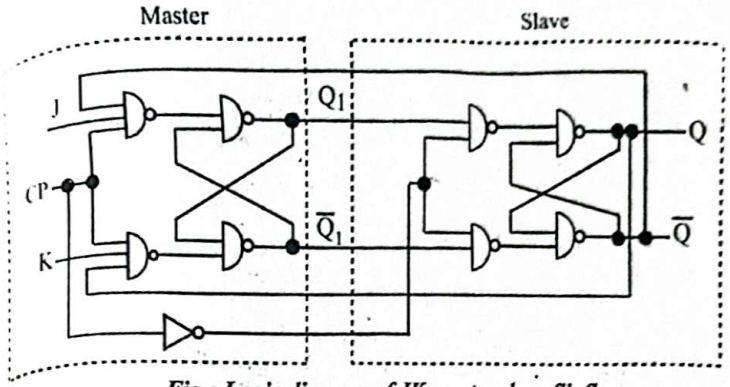


Fig.: Logic diagram of JK master slave flipflop

To begin with, the master is positive-edge triggered and the slave is negative-edge triggered. Therefore, the master responds to its J and K inputs before the slave. If J=1 and K=0, the master sets. The high Q output of the master drives the J input of the slave, so on negative edge of the clock (this makes slave positive-edge triggered), the slave sets copying the action of the master. If J = 0, K = 1, the master resets when it is positive-edge triggered.

The high $\bar{Q}$ output of the master goes to the K input of the slave. So, on negative level of the clock, slave resets. Again, the slave has copied the master. If J = 1, K = 1, first master toggles its output and later slave does the same. If J = K = 0, the flip-flop is disabled and Q remains unchanged.

6.6 Various Representation of Flipflops

Various Representation of Flipflops

i. Characteristic equation of FF
ii. FF as finite state machine
iii. FF excitation table

1) **Characteristic equation of FF**

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

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

JK : $Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$

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

2) FF as finite state machine

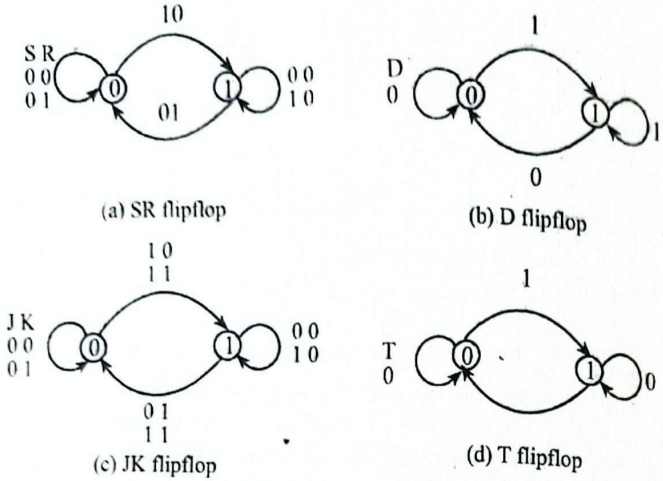


Fig: State transition diagram

3) Flipflop excitation table

It is very important in analysis of problem. It is just a reverse of truth table. Here, based on present output (Q_n) and next output (Q_{n+1}) the input of flipflop is generated.

Excitation table of flipflops

Transition		RS flipflop		JK flipflop		D flipflop	T flipflop
$Q_n \rightarrow Q_{n+1}$		S	R	J	K	D	T
0	0	0	×	0	×	0	0
0	1	1	0	1	×	1	1
1	0	0	1	×	1	0	1
1	1	×	0	×	0	1	0

EXAMPLE:

1. Convert SR flipflop to JK flipflop.

⇒ **Step 1:**

First we generate the truth table for JK flip–flop i.e., note $Q_n \rightarrow Q_{n+1}$ transition for a given combination of JK input.

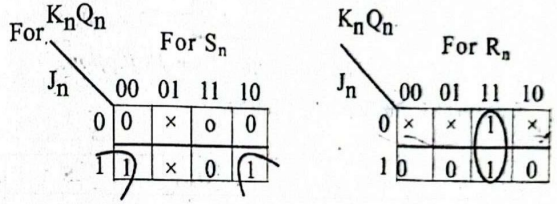
Step 2:

Next on the same table for $Q_n \rightarrow Q_{n+1}$ transition identify the excitation table for SR flip–flop such tables are known as synthesis table.

J_n	K_n	Q_n	Q_{n+1}	S_n	R_n
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	×	0
1	1	0	1	1	0
1	1	1	0	0	1

Step 3:

Write SR inputs as a function JK inputs and present state Q_n. It is done through K-map.



$S_n = J_n \overline{Q}_n$ $\quad R_n = K_n Q_n$

Step 4:

Design the circuit using SR flipflop and input JK.

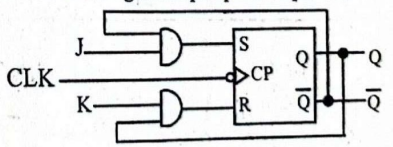


Fig.: JK flipflop from SR flipflop

2. **Convert JK flipflop to SR flipflop.**

⇒

S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	×	1
1	0	0	1	1	×
1	0	1	1	×	0

for, J

S \ RQn	00	01	11	10
0	0	×	×	0
1	1	×	×	×

J = S

for, K

S \ RQn	00	01	11	10
0	×	0	1	×
1	×	0	×	×

K = R

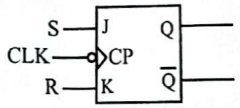


Fig.: SR flipflop from JK flipflop

3. **Convert SR flipflop to D flipflop.**

⇒

D	Q_n	Q_{n+1}	S	R
0	0	0	0	×
0	1	0	0	1
1	0	1	1	0
1	1	1	×	0

For R,

D \ Q_n	0	1
0	×	1
1	0	0

R = $\overline{D}$

For S,

D \ Q_n	0	1
0	0	0
1	1	×

S = D

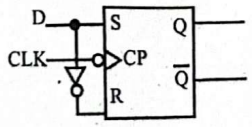


Fig.: SR flipflop to D flipflop

6.7 Analysis of a Sequential Circuit

A sequential logic circuit contains flipflops as memory elements and also contains logic gates as combinational circuits. Analysis of a circuit helps to explain its performance.

EXAMPLE:

1. **Consider the sequential circuit as shown below. It has only CLK and output 'X' is generated from flipflop outputs as shown. Analyse the circuit.**

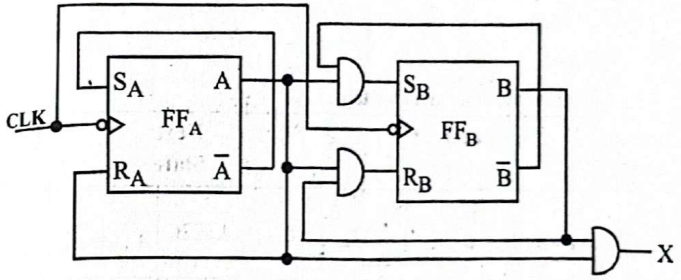


Fig.: A sequential logic circuit for analysis purpose

Step 1:

Note the flipflop input relations from the circuit.

$S_A = \overline{A}_n$, $R_A = A_n$ for FF_A

$S_B = A_n \overline{B}_n$, $R_B = A_n B_n$ for FF_B

Step 2:

Use the characteristic equation of SR flipflop to represent the next output i.e., $Q_{n+1} = S + Q_n \overline{R}$.

For FF_A,

$A_{n+1} = S_A + A_n \overline{R}_A$

$= \overline{A}_n + A_n \overline{A}_n$

$= \overline{A}_n$

For FF_B,

$B_{n+1} = S_B + B_n \overline{R}_B$

$= A_n \overline{B}_n + B_n \overline{A_n B_n}$

$= A_n B_n + B_n [\bar{A}_n + B_n]$

$= A_n B_n + \bar{A}_n B_n + B_n \bar{B}_n$

$= A_n \bar{B}_n + \bar{A}_n B_n$

$= A_n \oplus B_n$

Step 3:

Write the relation for output.

$X = A_n B_n$

Step 4:

Generate the state analysis table.

Present State		Present Input				Next State		Present Output
B_n	A_n	$S_B = A_n \bar{B}_n$	$R_B = A_n B_n$	$S_A = \bar{A}_n$	$R_A = A_n$	$B_{n+1} = A_n \oplus B_n$	$A_{n+1} = \bar{A}_n$	$X = A_n B_n$
0	0	0	0	1	0	0	1	0
0	1	1	0	0	1	1	0	0
1	0	0	0	1	0	1	1	0
1	1	0	1	0	1	0	0	1
0	0	0	0	1	0	0	1	0
0	1	....	....	....	....	....	....	Repeat

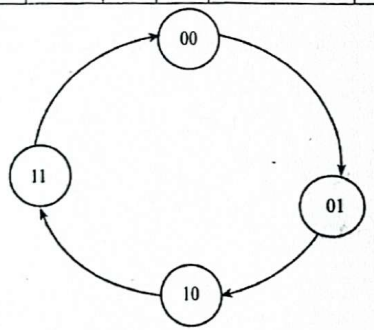


Fig.: State transition diagram of the sequential circuit

6.8 Switch Contact Bounce Circuits

Contact bounce (also called a chatter) as a common problem with mechanical switches and relays. When the contacts strikes together, their momentum and elasticity act together to cause bounce. The result is a rapidly pulsed electric volt instead of a clean transition from 0 V to full voltage. The effect is usually an important in power circuits but causes problems in digital logical circuits that respond fast enough to misinterpret the ON, OFF pulses as a data stream..

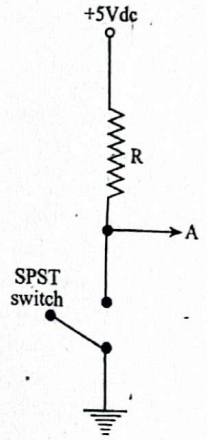


Fig.: A single-pole-single-throw (SPST) switch for illustrating contact bounce

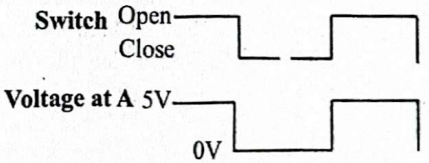


Fig.: Ideal voltage at A

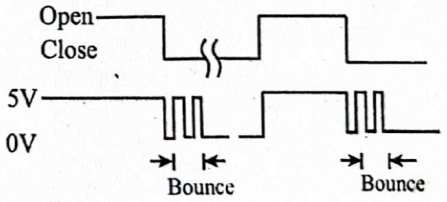


Fig.: Voltage at A showing contact bounce

When the switch is open, the voltage at point A is +5 V dc. When the switch is closed, the voltage at A must be 0 V ideally. But there occurs bounce which results the voltage level as shown in figure. This bounce problem can be eliminated with a simple RS latch debounce circuit.

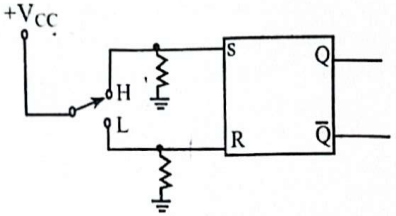


Fig.: Switch contact bounce eliminator

When the switch is moved to position H, R = 0 and S = 1. Bouncing occurs at the S input due to the switch. The flipflop sees this as a series of high and low inputs, settling with a high level. The flipflop will immediately be set with Q = 1 at the first high level on S. When the switch bounces, losing contact, the input signals are R = S = 0, therefore, the flipflop remains set (Q =1). When the switch regains contact, R = 0 and S = 1; this causes an attempt to again set the flipflop. But since the flipflop is already set, no changes occur at Q. The result is that the flipflop responds to the first, and only to the first, high level at its S input, resulting in a "clean" low-to-high signal at its output (Q).

When the switch is moved to position L, S = 0 and R =1. Bouncing occurs at the R input due to the switch. Again, the flipflop sees this as a series of high and low inputs. It simply responds to the first high level, and ignores all following transitions. The result is a "clean" high-to-low signal at the flipflop output.

MORE WORKED OUT EXAMPLES:

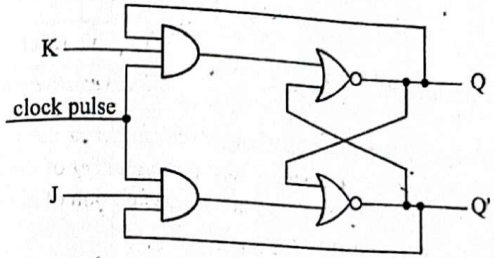
1. Explain about JK flip–flop along with their truth table and characteristic equation. **[2067 Ashadh]**

⇒ The JK flip –flop is versatile and widely used type of flip–flop. The J and K designations for inputs.

Operation:

The output Q is ANDed with K and clock pulse inputs so that the flip–flop is cleared during a clock pulse only if Q was previously 1. Similarly, output Q' is ANDed with J and clock pulse inputs so that the flip–flop is set with clock pulse only if Q' was previously 1.

When both J and K are 1, the clock pulse is transmitted through one AND gate only the one whose input is connected to the flip–flop output. Which is presently equal to 1 upon application of a clock pulse and the flip-flop is cleared. If Q = 1, the output of the lower AND gate becomes 1 and the flip–flop is set. In either use, the output state of flip–flop is complemented.



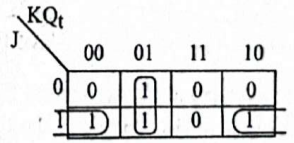
Characteristic table:

J	K	Q_t	Q_{t+1}
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

Q_t = output before apply the input

Q_{t+1} = Output after applying the input

Using K –map



∴ Characteristics equation = $\bar{K} Q_t + J\bar{Q}_t$

2. Differentiate between edge triggered and pulse triggered flip-flops. Explain about master slave flip–flop. [2064 Falgun]

⇒

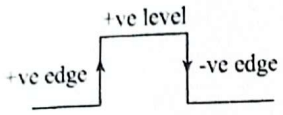


Fig.: Positive pulse

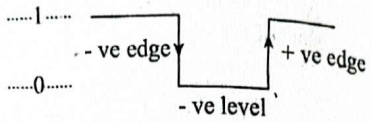


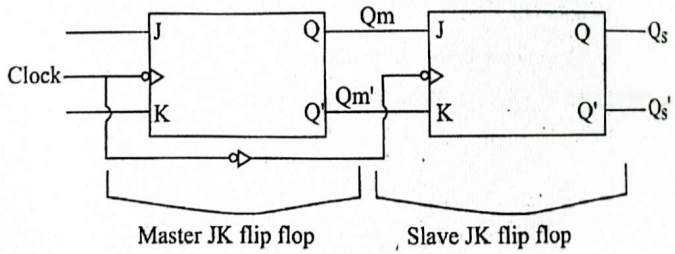
Fig.:Negative pulse

An edge triggered flip–flop changes state either at the positive edge *(rising edge)* or at the negative edge *(falling edge)* of the clock pulse and is sensitive to its input only at this transmission of the clock.

Examples. S-R, D, J-K, T flip flops.

A pulse triggered flip–flop changes state either at the positive pulse *(positive level of pulse)* or at the negative pulse *(negative level of pulse)* of the applied clock pulse.

A master salve flip–flop is constructed from two separate flip–flops one circuit serves as a master and other as a slave and the overall circuit is referred to as the master slave flip-flop. For example, let we take master slave JK flip-flop. This is the type of flip-flop which is composed of two sections master and slave. The master section is basically gated latch and slave is also the same except that it is clocked on inverted clock pulse and is controlled by output of master section rather than by external JK input.



Hence a flip–flop composed of two internal flip–flops one to receive the inputs *(the master)* and one to drive the outputs *(the slave)*. The master flip–flop receives its information during the leading edge of a clock pulse, and the slave or output flip–flop receives its information from the master during the failing edge of the pulse.

3. Derive characteristic equation of a JK flip flop. How do you make it a toggle flip flop? Draw the input and output wave form of JK flip flop. [2071 Chaitra]

⇒ **Characteristics table for JK flip flop is**

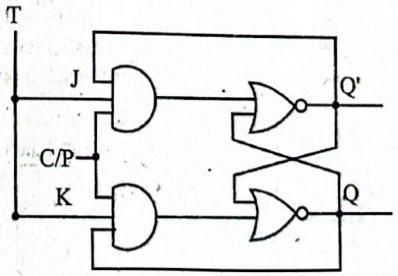
Q_n	J	K	Q_{n+1}	Remarks
0	0	0	0	No change
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	1	Toggle
1	0	0	1	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	0	Toggle

For characteristic equation,

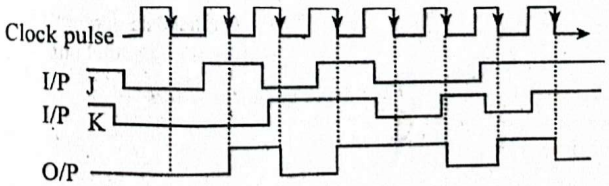
Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$\therefore \quad Q_{n+1} = Q'_nJ + Q_nK'$

Toggle flip-flop is made by shorting the input J and K terminals of JK flip-flop.



For negative edge trigerring,



Chapter – 7

REGISTERS

7.1 Introduction

A register is a group of binary cells (flipflops) suitable for holding information. It is a group of flipflops (FF) sensitive to pulse transition. A register is a digital circuit with two basic functions: data storage and data movement. A register consists of one or more flipflops used to store and shift data. An 'n' bit register consists of a group of 'n' flipflops capable of storing 'n' bits of binary information. A register capable of shifting its binary information in one or more direction is called shift register.

7.2 Types of Shift Registers

There are four types of shift registers:

i. Serial in serial out (SISO)

ii. Serial in Parallel out (SIPO)

iii. Parallel in Serial Out (PISO)

iv. Parallel in Parallel Out (PIPO)

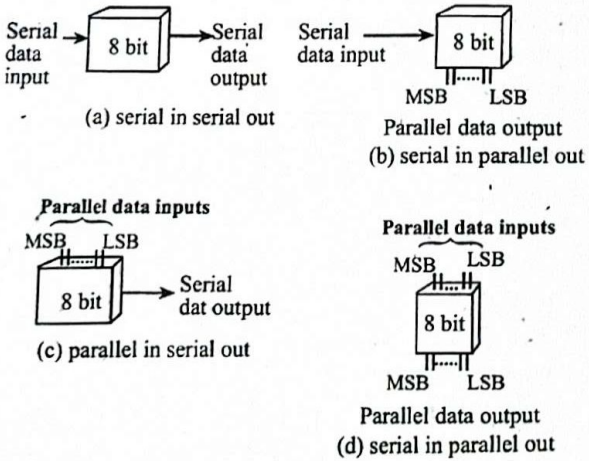


Fig.: Types of shift registers

i) **Serial in Serial Out (SISO) Shift Register**

SISO shift registers accepts data serially i.e., one bit at a time and output taken from it is also in serial form.

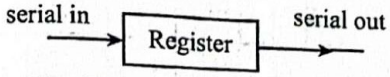


Fig.: Block diagram of right shift register

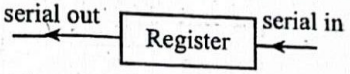


Fig.: Block diagram of left shift register

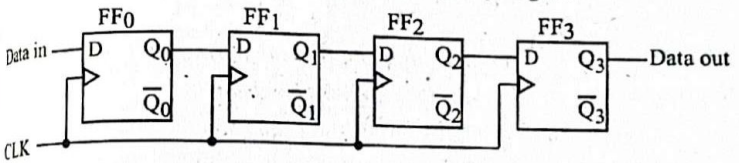


Fig.: Logic diagram of serial in serial out shift register (right shift register).

Let's explain its operation by storing and retrieving "1011". Let the initially the register content is clear i.e., 0000. The rightmost data is entered first.

Clock	Register content				
	Q_0	Q_1	Q_2	Q_3	
Initially	0	0	0	0	
CLK 1	1	0	0	0	
CLK 2	1	1	0	0	
CLK 3	0	1	1	0	
CLK 4	1	0	1	1	All data stored after 4th clock
CLK 5	0	1	0	1	
CLK 6	0	0	1	0	
CLK 7	0	0	0	1	
CLK 8	0	0	0	0	Register clear after 8th clock

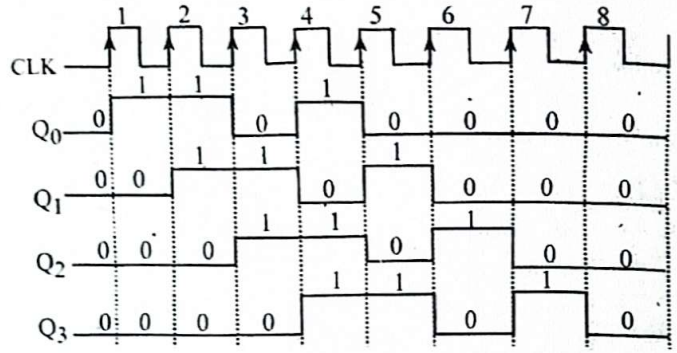


Fig.: Timing diagram for storing 1011

2) Serial in Parallel Out (SIPO) Shift Register

Data are entered serially but output is taken parallelly. Once the data are stored, each bit appears on its respective output line, all at a time.

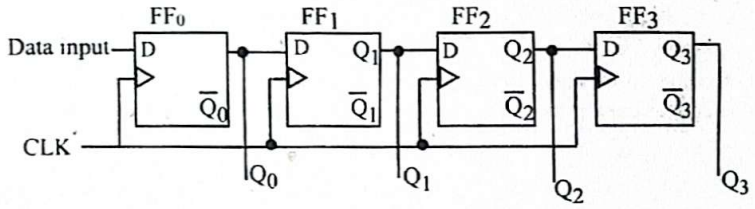


Fig.: Logic diagram of serial in parallel out (SIPO) shift register.

Let 1010 is stored from rightmost bit.

CLK	Register content				Data is completely stored after 4th clock. Output is available all at a time parallelly.
	Q_0	Q_1	Q_2	Q_3	
Initially	0	0	0	0	
CLK 1	0	0	0	0	
CLK 2	1	0	0	0	
CLK 3	0	1	0	1	
CLK 4	1	0	1	0	

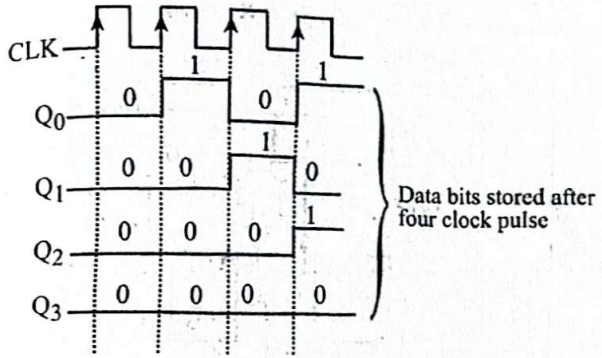


Fig.: Timing diagram for storing 1010 in SIPO shift register.

3) Parallel in Parallel Out (PIPO) Shift Register

Data are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis. Output is available simultaneously.

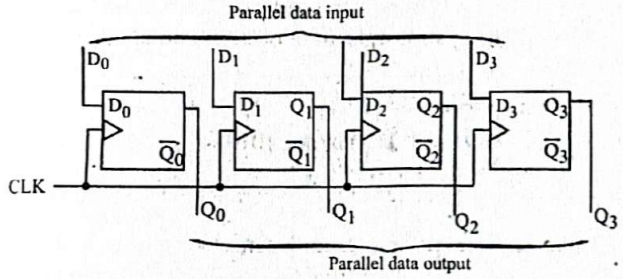


Fig.: Logic diagram of 4-bit parallel in parallel out (PIPO) shift register.

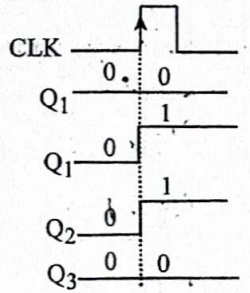


Fig.: Timing diagram for storing 0110 in PIPO shift register

4) Parallel in Serial Out (PISO) Shift Register

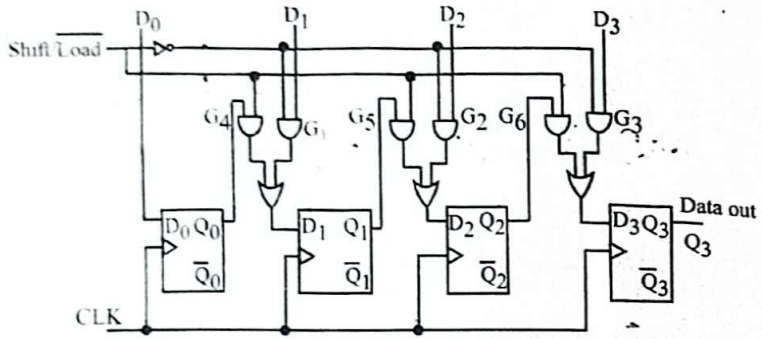


Fig.: Logic diagram of parallel in serial out (PISO) shift register.

Data are entered parallelly but taken in a serial mode. When shift/$\overline{\text{Load}}$ is LOW, G_1, G_2, G_3 gates are enabled so it allows data D_1, D_2, D_3 to be applied at D input of flipflops. When shift $\overline{\text{Load}}$ is HIGH; G_4, G_5, G_6 gates are enabled, allowing bits to shift right from one stage to another on each clock pulse. Output is taken at Q_3 serially.

Let's consider input data as

$D_0 = 1$, $D_1 = 0$, $D_2 = 1$, $D_3 = 0$ (i.e., 1010)

On clock pulse 1, shift $\overline{\text{Load}}$ is LOW, so parallel data ($D_0\ D_1\ D_2\ D_3 = 1010$) are loaded into register making Q_3 at 0. On clock pulse 2, shift/$\overline{\text{Load}}$ is HIGH, so data is shifted i.e., the 1 from Q_2 is right shifted onto Q_3 and so on.

	Register content				
CLK	Q_0	Q_1	Q_2	Q_3	
1	↓	0	1	0	Data is entered parallelly
2	0	1	0	1	
3	0	0	1	0	
4	0	0	0	1	
5	0	0	0	0	

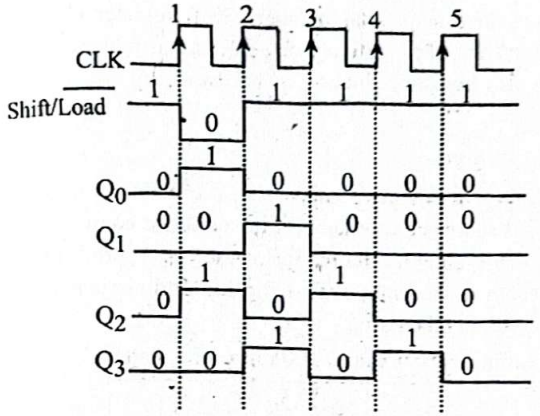


Fig.: Timing diagram for storing 1010 in PISO shift register

7.3 Bidirectional Shift Register (Universal Register)

Data can be shifted either right or left in the bidirectional shift register.

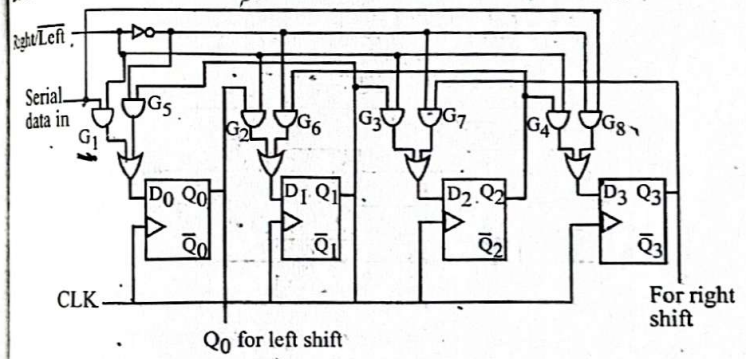


Fig.: Logic diagram of bidirectional shift register.

When Right/$\overline{\text{Left}}$ = 1; G_1, G_2, G_3, G_4 are enabled and right shift of data occurs. When Right/$\overline{\text{Left}}$ = 0; G_5, G_6, G_7, G_8 are enabled and left shift of data occurs.

7.4 Application of Shift Register

Shift registers are used in almost every sphere of a digital logic system. Shift register can be used to count number of pulses entering into a system as ring counter or switched-tail counter. As ring counter, it can generate various

control signals in a sequential manner. Shift register can also generate a prescribed sequence of repetitively or detect a particular sequence from data input. It can also help in reduction of hardware by converting parallel data feed to serial one.

1) Ring Counter

It is a type of counter composed of a circular shift register. It is used to count sequence of operation in sequence control of stepper motor, etc. In ring counter, the output of the last flipflop of shift register is connected to the input of first flipflop and circulates a single one (or zero) bit around the ring.

E.g., Initial register values 100 represents pattern.

CLK	Q_0	Q_1	Q_2	
0	1	0	0	Initially
1	0	1	0	
2	0	0	1	
3	1	0	0	

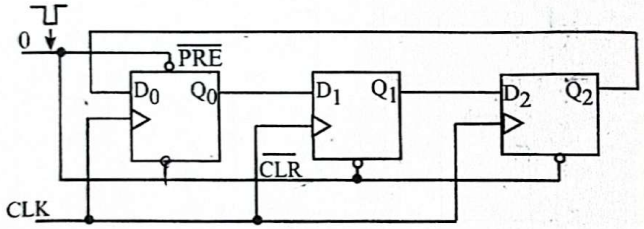


Fig.: Logic diagram of a ring counter.

Initially $\overline{PRE} = 0$, $\overline{CLR} = 0$ should be given i.e., $100 = Q_0 Q_1 Q_2$ and $\overline{PRE} = 1$, $\overline{CLR} = 1$ is given. Now at every clock pulse, the 1 is shifted to next stage.

Drawbacks:

The requirement of initialization is a disadvantage of ring counter.

2) Johnson Counter (Switched-Tail Counter)

This counter eliminates the limitation of ring counter. In Johnson counter, complement output of last flipflop is connected to input of first flipflop.

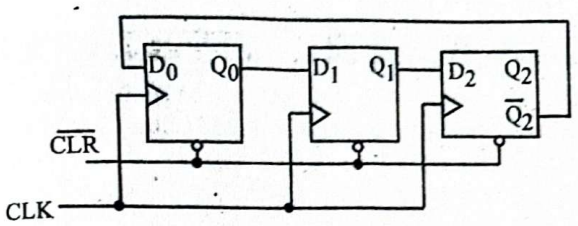


Fig.: Logic diagram of Johnson counter.

Initially $\overline{CLR} = 0$, is given to reset to 000 state.

CLK	Q_0	Q_1	Q_2
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
6	0	0	0

Hence, important applications of shift registers are:

- For temporary data storage
- For serial adder
- To produce time delay
- To convert serial data to parallel data
- To convert parallel data to serial data
- As ring counter
- As Johnson counter

MORE WORKED OUT EXAMPLES:

1. **Four bit data 1010 is entered into serial in parallel out shift register. Draw the circuit diagram and output waveforms after 1, 2, 3, 4 clock pulses. [2064 Jestha]**

⇒

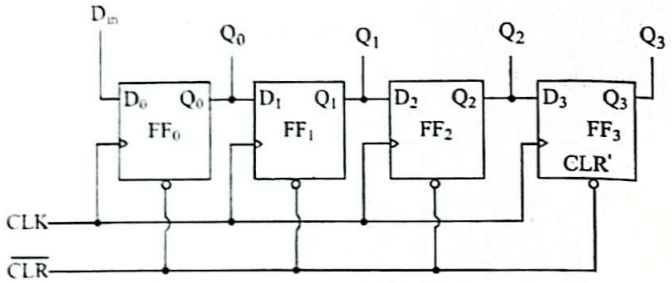
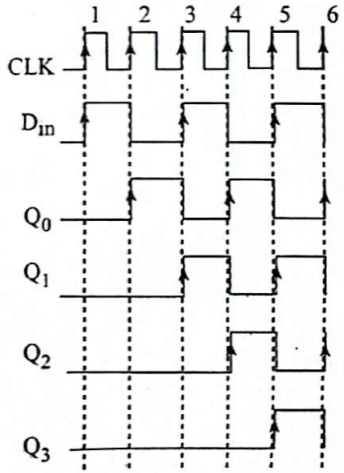


Fig.: 4-bit serial in parallel out shift register

Now, timing diagram for above circuit is



Chapter – 8

COUNTERS

8.1 Introduction

The combination of flipflops that performs the counting operation are known as counter. A counter is probably one of the most useful and versatile subsystems in a digital system. It is a sequential circuit that passes through predefined number of states.

Counters are classified into two broad categories according to the way they are clocked:

i. Asynchronous (or ripple or serial) counter

ii. Synchronous (or parallel) counter.

1. **Asynchronous (or Ripple or Serial) Counter**

In this counter, the first flipflop is clocked by the external clock pulse and then each successive flipflop is clocked by the output of the preceding flip–flop. Asynchronous counter is simple and straightforward in operation and its construction usually requires a minimum of hardware. It does have a speed limitation.

2. **Synchronous (or Parallel) Counter**

In synchronous counter, the clock input is connected to all of the flipflops so that they are clocked simultaneously. An increase in speed of operation can be achieved by use of synchronous counter.

8.2 Asynchronous Counters

3-bit Ripple Up Counter (or MOD-8 Up Counter)

The 3-bit ripple up counter constructed from JK flipflop is shown below. The JK input are tied together to $+V_{CC}$ such that at each negative transition of its clock input, the flipflop toggles its states.

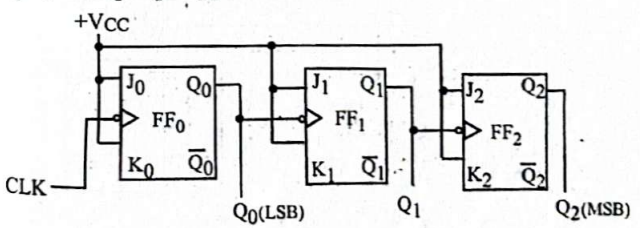


Fig.: A 3-bit binary ripple up counter

Truth table

Clock (NT)	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
6	1	0	1
7	1	1	0
8	1	1	1

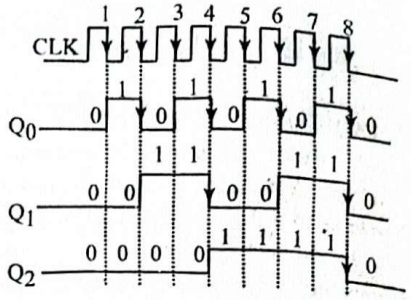


Fig.: Timing diagram

Since it is a 3-bit up counter, it counts from 000 – 111 in binary (i.e., 0 to $2^3 - 1$ in decimal). Modulus of a counter is the total number of states through which the counter can progress. The maximum number of states = $2^n = 2^3 = 8$. For 3-bit counter, 3 flipflops are required. Therefore, 3-bit counter is also known as MOD-8 counter.

The waveforms illustrated above show the action of the counter as the clock runs. Initially, all the flipflops are reset to produce 0 outputs. If we consider Q_0 to be the LSB and Q_2 MSB, we can say the contents of the counter is Q_2 Q_1 Q_0 = 000. Every time there is a clock NT, FF_0 will change state. Since Q_0 acts as the clock for FF_1, each time the waveform at Q_0 goes low, FF_1 will toggle. Since Q_1 acts as the clock for FF_2, each time the waveform at Q_1 goes low, FF_2 will toggle.

3-bit Ripple Down Counter (MOD-8 Ripple Down Counter)

Only first flipflop is given clock pulse from the external source. The remaining flipflop gets clock from the output of preceding flipflop i.e., $\overline{Q}$.

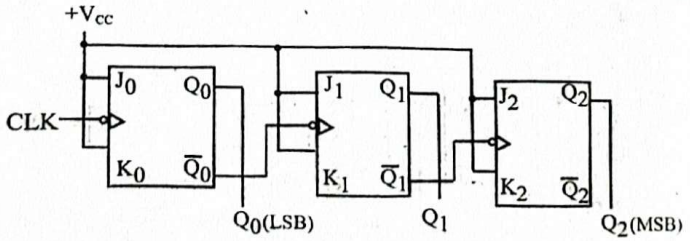


Fig.: 3-bit ripple down counter

Truth table

Clock (NT)	Q_2	Q_1	Q_0	Count
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0

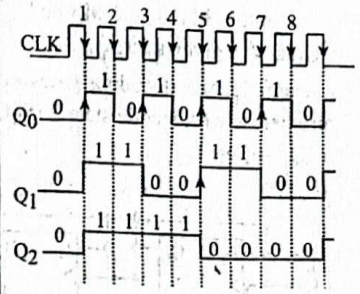


Fig.: Wavefoms

3-bit Ripple Up/Down Counter

It is the combination of the two counter discussed previously. As from above two counter, we observe that for

(a) up-counter, output Q is connected to CLK of successive FF

(b) down-counter, output $\overline{Q}$ is connected to CLK of successive FF

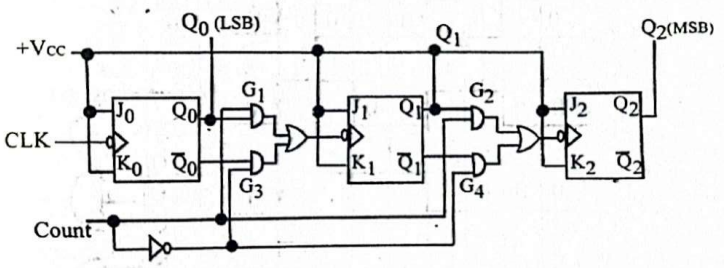


Fig.: 3-bit up/down ripple counter

Here 'Count' controls whether to count up or down.

When Count = 1, gates G_1 & G_2 are enabled, so it performs up count.
When Count = 0, gates G_3 & G_4 are enabled, so it performs down count.

8.3 Decoding Gates

A decoding gate can be connected to the outputs of a counter in such a way that the output of the gate will be high or low only when the counter contents are equal to a given state.

For example, the decoding gates connected to the 3-bit ripple counter in figure below will decode state 7 ($Q_2Q_1Q_0$ =111). The gate will be high when $Q_0 = Q_1 = Q_2 = 1$.

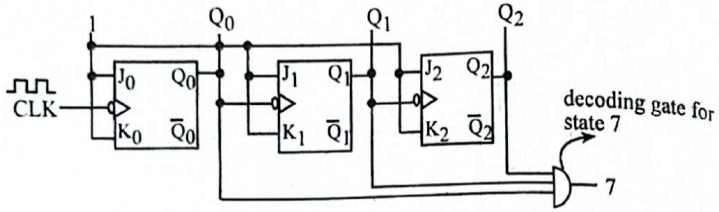


Fig.: Decoding gate for state 7

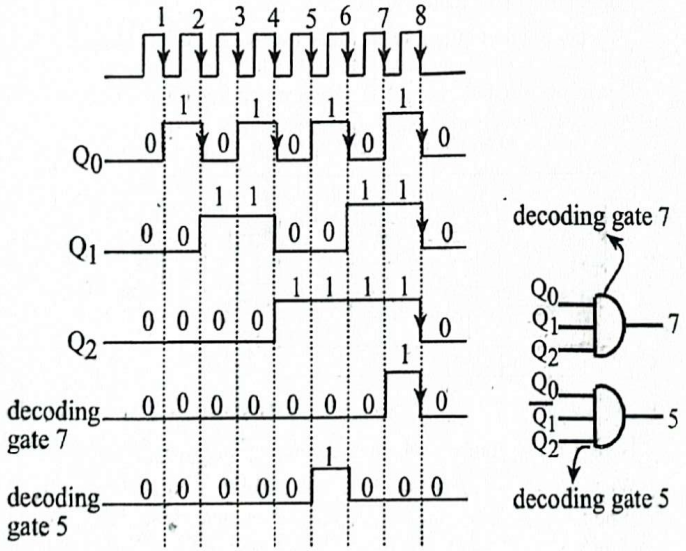


Fig.: Waveforms

8.4 Synchronous Counter (Parallel Counter)

3-bit Synchronous Up Counter (or MOD-8 Synchronous Up Counter)

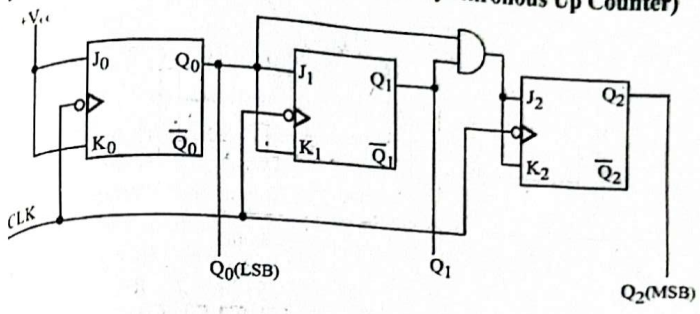


Fig.: MOD-8 synchronous up counter

Truth table

Clock (NT)	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

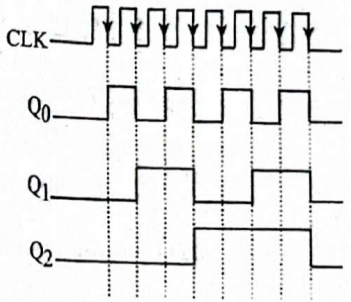


Fig.: Timing diagram

Initially, $Q_0 = Q_1 = Q_2 = 0$. When first clock pulse is applied, Q_0 toggles to 1. J_1 is still 0 for first clock pulse, so $Q_1 = 0$; similarly, $Q_2 = 0$. For second clock pulse, Q_0 toggles to 0, $J_1 = 1$, so Q_1 toggles to 1. Q_2 remains 0 since for second clock pulse, $Q_1 = 0$. For third clock pulse, Q_0 toggles to 1, $J_1 = 0$, so $Q_1 = 1$ and $Q_2 = 0$. For the fourth clock pulse, Q_0 toggles to 0, $J_1 = 1$, so Q_1 toggles to 0, $J_2 = Q_0Q_1 = 1$, so Q_3 toggles to 1, and so on.

3-bit Synchronous Down Counter (MOD-8 Synchronous Down Counter)

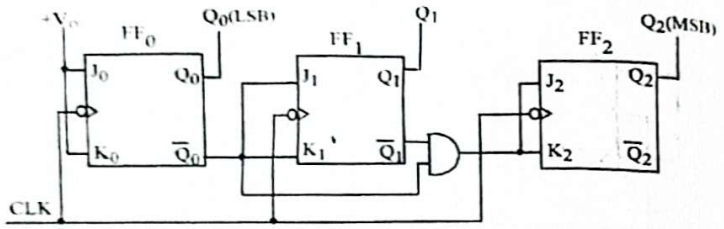


Fig.: 3-bit synchronous down counter

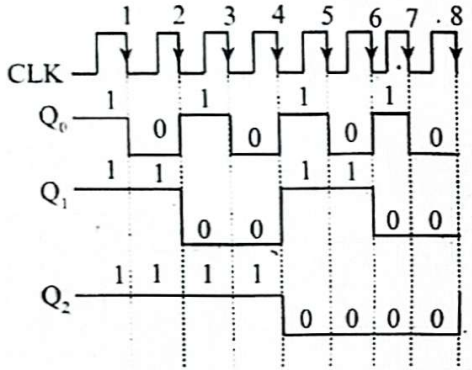


Fig.: Waveforms

3-bit Synchronous Up/Down Counter

As like ripple up/down counter, it is also the combination of up and down synchronous counter.

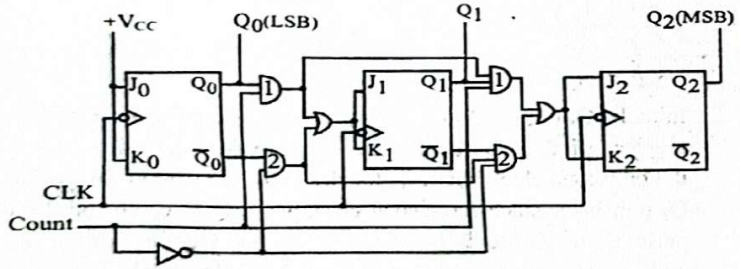


Fig.: 3-bit synchronous up/down counter

When Count = 1, AND gate 1 is ON, so Q_0 is transferred to J_1 and K_1, and so it starts counting up. When Count = 0, AND gate 2 is ON, so $\overline{Q}_0$ is transferred to J_1 and K_1, and so it starts counting down.

8.5 Decade Counter (MOD-10 Counter or BCD Counter)

It is also known as mod-10 counter or BCD counter. It counts from 0 to 9 in decimal. Thus it requires 10 clock pulse for resetting. It counts from 0000 to 1001 and skips rest of the states and this is possible because at 10^{th} clock pulse, it generates its own clear signal from decoding gate and resets to 0000. That is, when the counter counts $Q_3Q_2Q_1Q_0 = 1010$, then the output of NAND gate is low, so it clears all the flipflops.

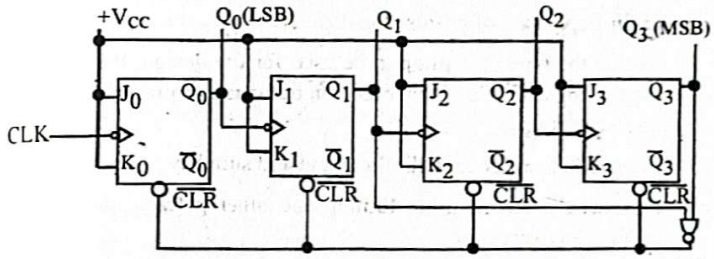


Fig.: Asynchronous decade counter

8.6 Presettable Counters

The presettable counter is one in which the counter starts counting not from zero but from any numbers. The figure below shows the 4-bit resettable counters, it has a special 'Load' terminal and counting begins with P_3, P_2, P_1, P_0 which can be any numbers between 0000 and 1111.

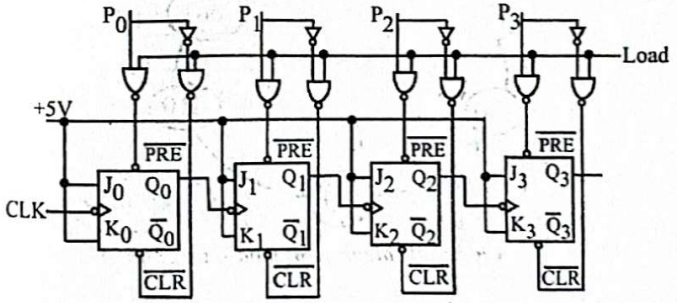


Fig.: 4-bit presettable counter

When Load is LOW, all NAND gates have high outputs, so all PRESET and CLEAR are inactive. Therefore, it counts in normal way. When Load is HIGH, the inputs P_3, P_2, P_1, P_0 and their complements $\overline{P_3}$, $\overline{P_2}$, $\overline{P_1}$, $\overline{P_0}$ passes through NAND gates. This action presets the counter to $P_3P_2P_1P_0$ states in the initial. When Load reverts to LOW again, then the counting starts and successive clock pulse produces the remaining states till to maximum.

8.7 Counter Design as a Synthesis Problem

Following steps are to be followed for designing synchronous counters:

1) Draw the state diagram from the given word description problem.
2) Draw the next state table and find number of flipflops required. No. of flipflops = no. of bits used in counter.
3) Decide the type of flipflop to be used for the design, then determine the flipflop excitation table based on transition of present state (Q_n) to next state (Q_{n+1}).
4) Prepare K-map for each flipflop input and simplify.
5) Connect the circuit using flipflop and other gates according to the minimized expression.

Example:

1. Design a MOD–6 up counter (i.e. 6–state counter)

⇒

Step 1: Draw the state diagram

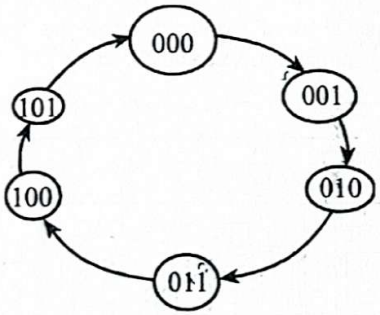


Fig.: State diagram of mod-6 up counter

Step 2 and step 3: Next state table and excitation table

Present State (Q_n)			Next State (Q_{n+1})			FF Excitation					
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	1	0	1	×	0	0	×	1	×
1	0	1	0	0	0	×	1	0	×	×	1

Step 4:

Prepare the K-map for J and K input from transition table. 110 and 111 are unused condition, so are don't care conditions.

For J_2,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	0	1	0
1	×	×	×	×

$J_2 = Q_{1n}Q_{0n}$

For K_2,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	×	×
1	0	1	×	×

$K_2 = Q_{0n}$

For J_1,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	0	1	×	×
1	0	0	×	×

$J_1 = \overline{Q_{2n}}Q_{0n}$

For K_1,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	×	1	0
1	×	×	×	×

$K_1 = Q_{0n}$

For J_0,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	1	×	×	1
1	1	×	×	×

$J_0 = 1$

For K_1,

Q_{2n} \ $Q_{1n}Q_{0n}$	00	01	11	10
0	×	1	1	×
1	×	1	×	×

$K_0 = 1$

Step 5:

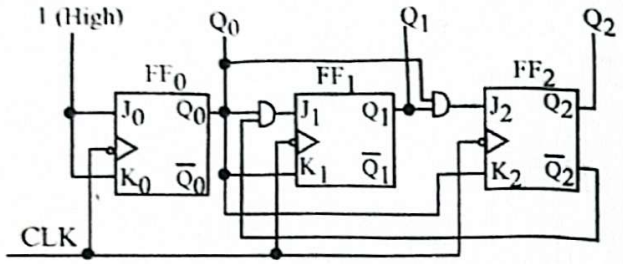


Fig.: Mod-6 synchronous up counter

2. Design a 2-bit up/down counter using T flipflops.

⇒

Step 1: Let Y = 1 (up counter), Y = 0 (down counter)

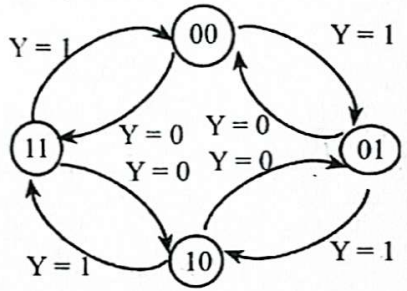


Fig.: State diagram of 2-bit up/down counter

Step 2 and Step 3:

The no. of bits are 2, so no. of flipflops = 2

Present State (Q_n)		Next State (Q_{n+1})			
		Y = 0 (down)		Y = 1 (up)	
Q_{1n}	Q_{0n}	Q_{1n+1}	Q_{0n+1}	Q_{1n+1}	Q_{0n+1}
0	0	1	1	0	1
0	1	0	0	1	0
1	0	0	1	1	1
1	1	1	0	0	0

Present State Q_n		Control Input	Next State (Q_{n+1})		Flipflop Excitation	
Q_{1n}	Q_{0n}	Y	Q_{1n+1}	Q_{0n+1}	T_1	T_0
0	0	0	1	1	1	1
0	0	1	0	1	0	1
0	1	0	0	0	0	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	1	1	1	0	1
1	1	0	1	0	0	1
1	1	1	0	0	1	1

Step 4: K-mapping for T_1 and T_0 (inputs are Q_{1n}, Q_{0n}, Y)

For T_1,

Q_{1n} \ Q_{0n}Y	00	01	11	10
0	1	0	1	0
1	1	0	1	0

$T_1 = \overline{Q}_{01}\overline{Y} + Q_{01}Y$

$= \overline{(Q_{01} \oplus Y)}$

For T_0,

Q_{1n} \ Q_{0n} Y	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$T_0 = 1$

Step 5:

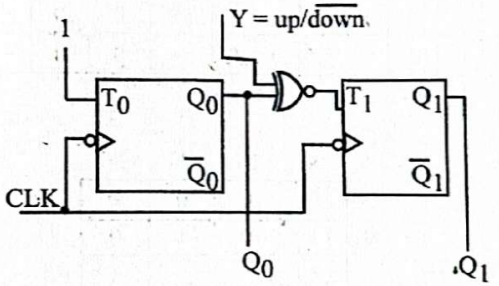
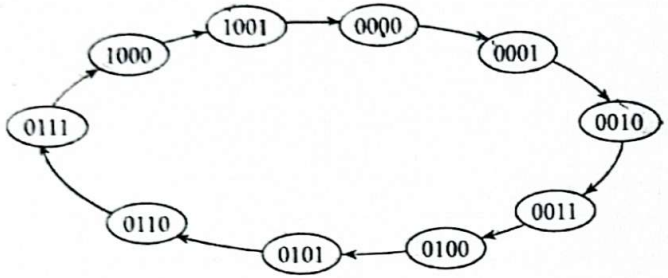


Fig.: 2-bit synchronous up/down counter.

MORE WORKED OUT EXAMPLES:

1. Design the synchronous decade counter and also show the timing diagram. [2069 Chaitra]

⇒



State diagram for synchronous decade counter

Present state				Next state				Flip-flop			
A	B	C	D	P	Q	R	S	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	1	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

For T_A

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

$T_A = AD + BC'D' + BCD$

For T_B

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

$T_B = RD$

For T_C

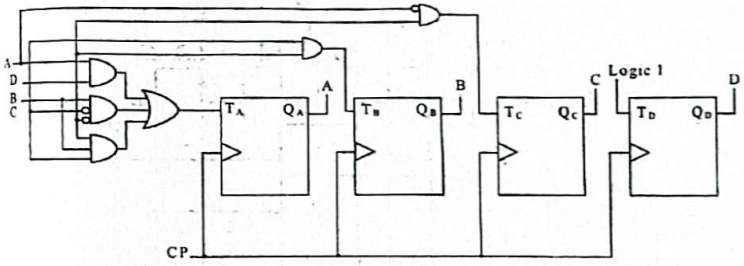
AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	×	×	×	×
10	0	0	×	×

$T_C = A'D$

For T_D

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

$T_D = 1$



2. **Design a MOD–5 binary ripple up counter. What are the advantages of a synchronous counter over a ripple counter? [2066 Bhadra]**

⇒

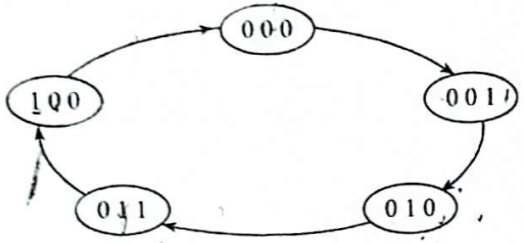


Fig.: State diagram of MOD-5-binary ripple up counter

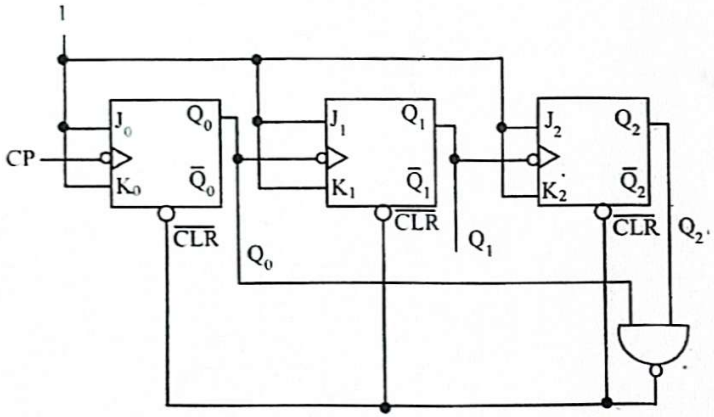


Fig.: Circuit diagram of MOD-5 binary ripple up counter.

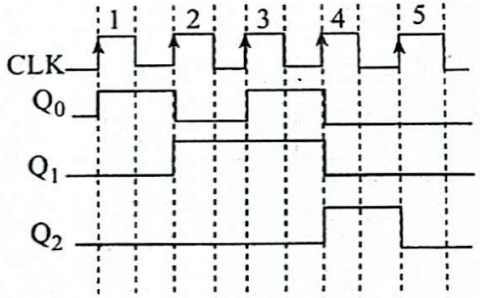


Fig.: Timing diagram

Advantages of a synchronous counter over a ripple counter

1 Synchronous counter is free from glitches.
2 Faster in operation if high frequency is applied.

3. **Design a MOD–5 binary synchronous up counter using the excitation table and draw its timing diagram. [2066 Bhadra]**

⇒

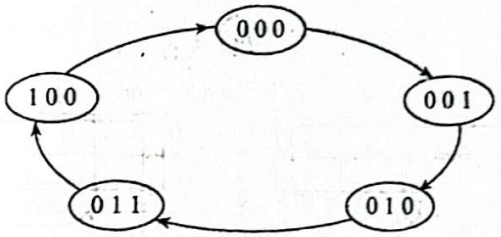


Fig.: Static diagram of Mod-5 binary up counter

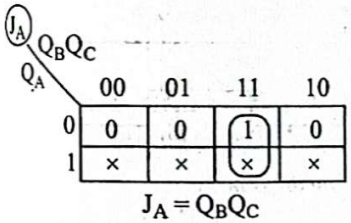
Excitation table of JK flip–flop

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

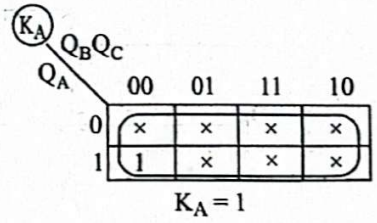
State excitation table:

Present state			Next state			J_A	K_A	J_B	K_B	J_C	K_C
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}						
0	0	0	0	0	1	0	×	×	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	0	0	0	×	1	0	×	0	×

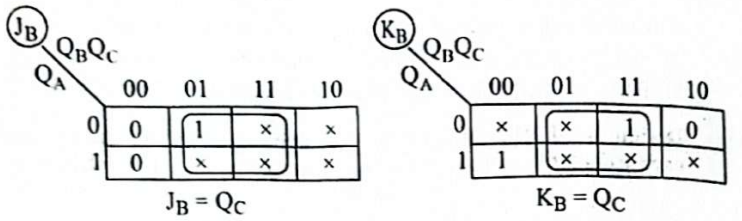
Using **K-maps**

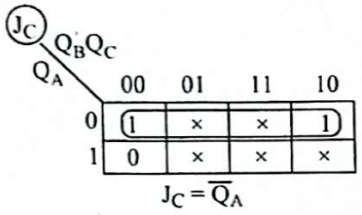


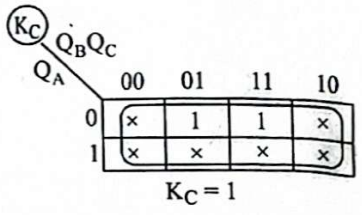
$J_A = Q_BQ_C$



$K_A = 1$







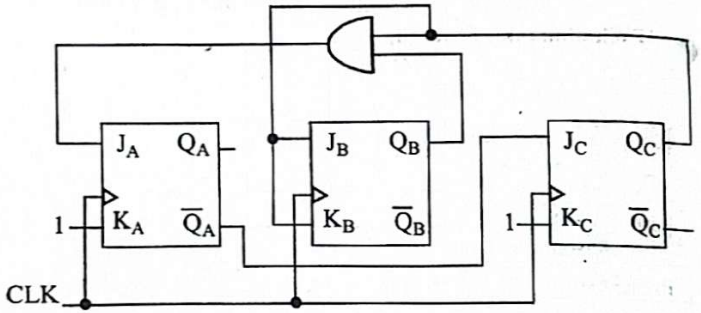


Fig.: 3-bit binary synchronous up counter

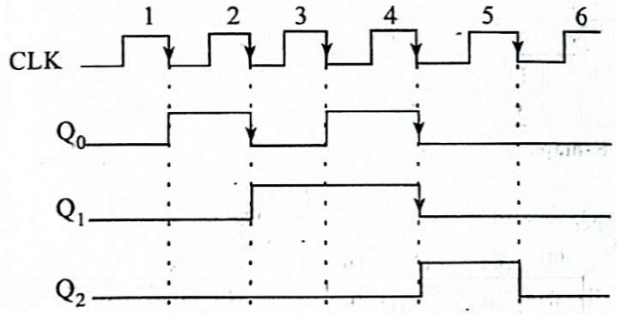


Fig.: Timing diagram

4. **Design a 3-bit down counter, using master–slave JK flipflops and show the timing diagram for 3 clock cycles, assuming that the initial reading of the counter is 0.** **[2065 Shrawan]**

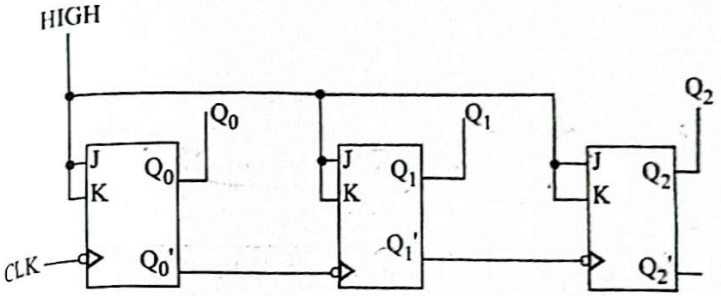


Figure above is the layout of 3-bit down counter. It resembles as the synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 so termed as down counter. For typically design purpose here we use master slave flip-flop, using three JK flip-flops counting three bits binary number on reverse order.

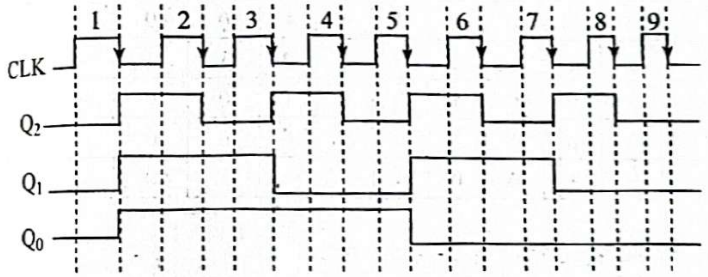
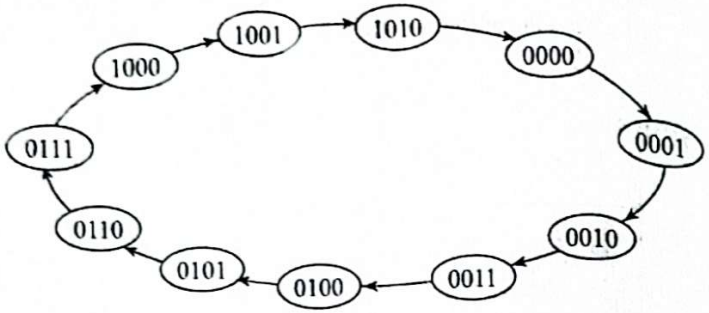


Fig.: Timing diagram

(Assuming the initial reading of counter is 0.)

5. **Design MOD-11 binary counter by showing its state, circuit diagram and output waveforms.** **[2064 Jestha]**

⇒ **State diagram:**

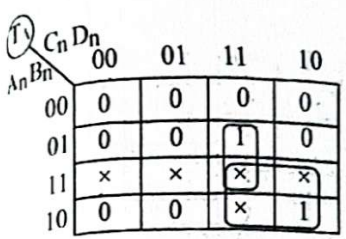


State table:

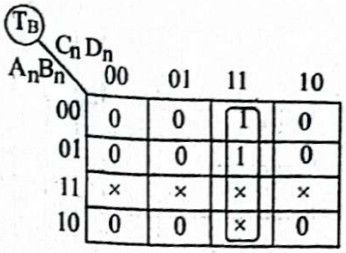
Present State				Next state				T_A	T_B	T_C	T_D
A_n	B_n	C_n	D_n	A_{n+1}	B_{n+1}	C_{n+1}	D_{n+1}				
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	0	0	0	1	0	1	0

Excitation table for T flipflop:

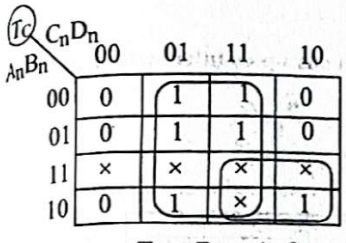
Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0



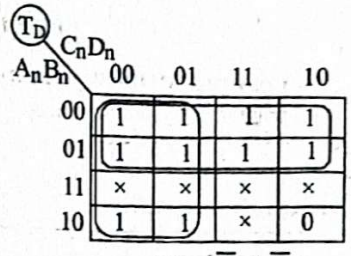
$T_A = B_nC_nD_n + A_nC_n$



$T_B = C_nD_n$



$T_C = D_n + A_nC_n$



$T_D = \overline{C}_n + \overline{A}_n$

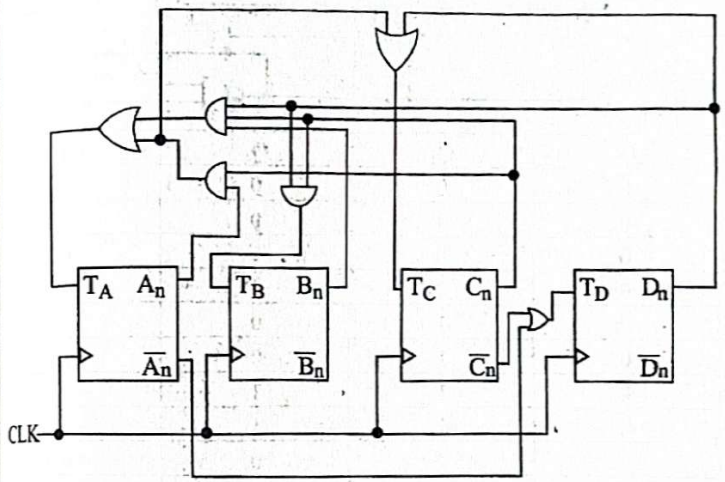


Fig.: MOD 11 counter

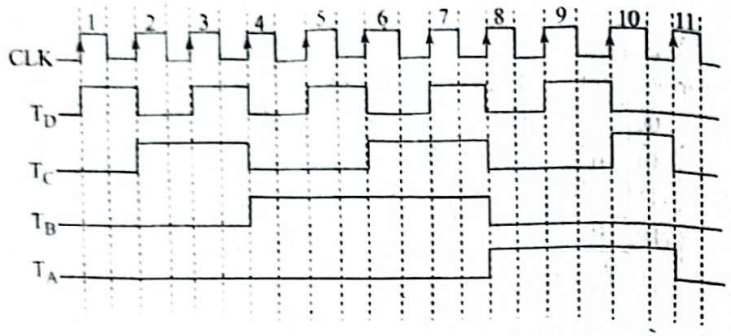
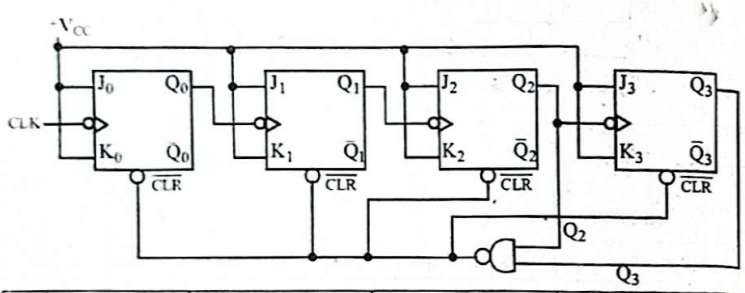


Fig.: Timing diagram

6. **Construct MOD-12 asynchronous up-counter with negative edge triggering system in clock.** **[2072 Kartik]**

⇒



Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
0	0	0	0

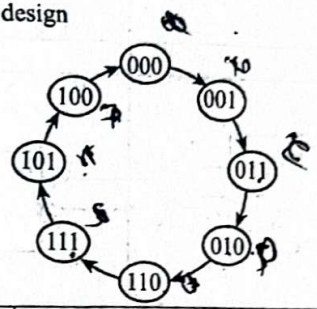
List the advantages and disadvantages of a synchronous counter over asynchronous counter. Design a 3-bit synchronous counter which follow gray code sequence. **[2074 Ashwin]**

Advantage of synchronous counter over asynchronous counter:

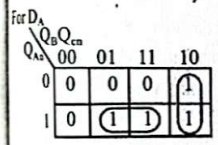
Fast operation

Disadvantages:

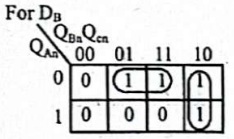
Complex circuit design



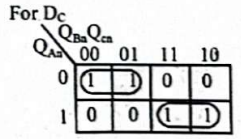
Present State			Next State			Flip flop		
Q_{An}	Q_{Bn}	Q_{Cn}	Q_{n+1}	Q_{Cn+1}	Q_{Cn+1}	D_A	D_B	D_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	1	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	0	1
1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0



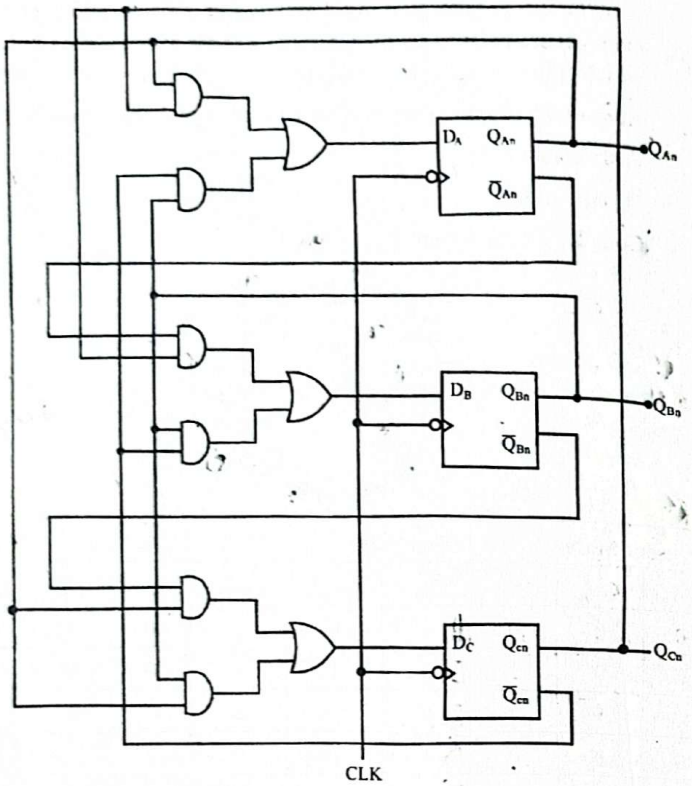
$D_A = Q_{An}Q_{Cn} + Q_{Bn}\overline{Q}_{Cn}$



$D_B = \overline{Q}_{An}Q_{Cn} + Q_{Bn}\overline{Q}_{Cn}$



$D_C = Q_{An}\overline{Q}_{Bn} + Q_{An}Q_{Bn}$



Chapter – 9

SEQUENTIAL MACHINES

9.1 Introduction

Sequential circuits may be classified into following two categories:

- Synchronous sequential circuits
- Asynchronous sequential circuits

1. **Synchronous Sequential Circuits**

   In the synchronous sequential circuits, the contents of memory element can be changed only at the rising or falling edge of clock signals.

2. **Asynchronous Sequential Circuits**

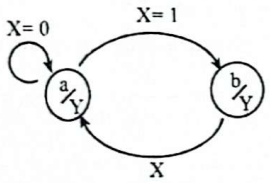
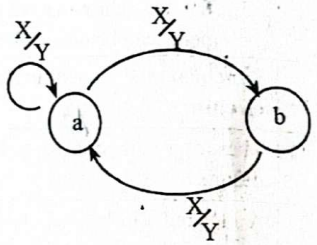
   In the asynchronous sequential circuits, the contents of memory elements can be changed at any instant of time.

	Synchronous sequential circuit		Asynchronous sequential circuit
1.	These circuits are easy to design.	1.	Difficult to design.
2.	A clocked flipflop acts as a memory element.	2.	Unclocked flipflop or time delay element is used as memory element.
3.	These circuits are slower because the delays correspond to those of the memory element.	3.	Faster as the clock is not present.
4.	The status of memory element is affected only at active edge of clock if the input is changed.	4.	The status of memory element will change any time as soon as the input is changed.

9.2 Design of Synchronous Sequential Circuit

There are two distinct models by which asynchronous sequential logic circuit can be designed:

- Moore Model
- Mealy Model

	Moore Model		Mealy Model
i.	The output depends only on present state and not on input.	i.	The output depends on present state as well as input.
ii.	It requires more no. of states and thereby more hardware.	ii.	It requires less no. of states and thereby less hardware to solve any problem.
iii.	Output is generated one clock cycle later than Mealy.	iii.	Output is generated one clock cycle earlier than Moore.
iv.	The output remains stable over entire clock period and changes only when there occurs a state change.  *Fig.: State diagram representation*	iv.	The output does not remain stable over entire clock period and changes over the same state.  *Fig.: State diagram representation*

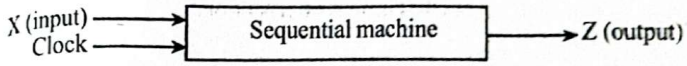
The steps to be followed for designing are:

1) Choose the model to be implemented (Moore or Mealy).
2) Draw the state transition diagram from word description problem.
3) Draw the next state table from the given information.
4) The number of states may be reduced by state reduction method if necessary.
5) Assign the states with binary value.
6) Choose the type of flipflop to be used, determine number of flipflops to be used based on states, and derive the excitation table based on present state (Q_n) to next state (Q_{n+1}) transition and output table.
7) Using K-map, simplify each input of flipflop and circuit output.
8) Draw the logic diagram.

EXAMPLES:

1. **A synchronous machine has one bit serial input 'X'. The output 'Z' of a machine is to be set high when the input contains the message '100'. Draw the state diagram, derive the transition table (state table), excitation table, and design the circuit diagram.**

⇒ **Using Moore Model**



Step 1: We choose Moore model

Step 2: State diagram

Circuit is initialized with state a.

If X = 1, then it moves to state b otherwise (if X = 0), then it remains in same state a.

At state b, if X = 1, it remains in same state b because state b is the state which have detected 1. But if X = 0, the first two pattern is detected so moves to state c.

At state c, if X = 0, then the input stream is 100 and circuit goes to state d with output = 1. But if X = 1, it moves to state b because state b has detected 1.

At state d, if the circuit continues sequence detection job, receiving X = 1, it goes to state b. If X= 0, the circuit goes to initial state a signifying not a single bit has been detected.

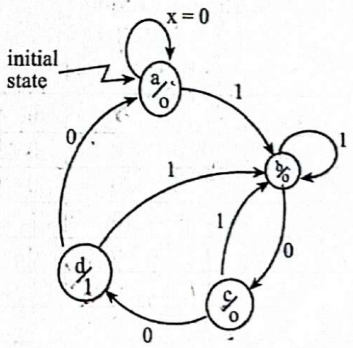


Fig.: State transition diagram of sequence detector.

Step 3: Next state table (transition table)

Previous State	Next State		Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	0
c	d	b	0	0
d	a	b	1	1

Step 4: No reduction required as there is no repetition.

Step 5: Binary assignment

Let a = 00, b = 01, c = 10, d = 11

Step 6: No. of flipflops = 2 because only 4 states are used.

Let's design with two JK- flipflops i.e., J_A K_A and J_B K_B.

Present State		Next State				Output Y		Excitation table							
		X=0		X=1		X=0	X=1	X = 0				X = 1			
B_n	A_n	B_{n+1}	A_{n+1}	B_{n+1}	A_{n+1}			J_B	K_B	J_A	K_A	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	0	0	×	0	×	0	×	1	×
0	1	1	0	0	1	0	0	1	×	×	1	0	×	×	0
1	0	1	1	0	1	0	0	×	0	1	×	×	1	1	×
1	1	0	0	0	1	1	1	×	1	×	1	×	1	×	0

OR

Present State		Present Input	Next State		Output	FF Excitation ($B_n \rightarrow B_{n+1}$) ($A_n \rightarrow A_{n+1}$)			
B_n	A_n	X	B_{n+1}	A_{n+1}	Y	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	×
0	1	0	1	0	0	1	×	×	1
0	1	1	0	1	0	0	×	×	0
1	0	0	1	1	0	×	0	1	×
1	0	1	0	1	0	×	1	1	×
1	1	0	0	0	1	×	1	×	1
1	1	1	0	1	1	×	1	×	0

Step 7: K-mapping for each input of flipflop (i.e., J_B, K_B, J_A, K_A) respective to present state and present input (i.e., B_n, A_n, X). Also, we plot K-map for output Y.

For J_B

B_n \ A_nX	00	01	11	10
0	0	0	0	1
1	×	×	×	×

$J_B = A_n\overline{X}$

For K_B

B_n \ A_nX	00	01	11	10
0	×	×	×	×
1	0	1	1	1

$K_B = X + A_n$

For J_A

B_n \ A_nX	00	01	11	10
0	0	1	×	×
1	1	1	×	×

$J_A = B_n + X$

For K_A

B_n \ A_nX	00	01	11	10
0	×	×	×	1
1	×	×	0	1

$K_A = \overline{X}$

Step 8: Logic diagram

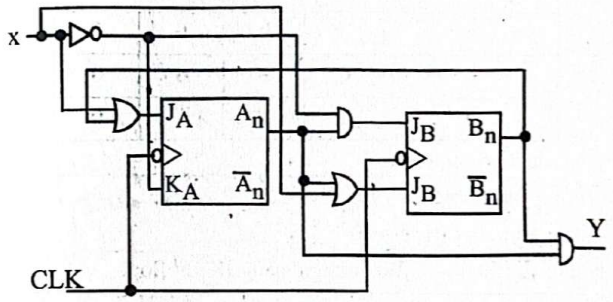


Fig.: Logic diagram for '100' sequence detector.

Using Mealy Model

Step 1: We choose Mealy model

Step 2: State diagram

Circuit is initialized with state a.

If X = 0, then it remains in same state a. If X = 1, then it moves to state b (because first bit is detected.)

At state b, if X = 0, the first two pattern is detected, so moves to state c. But if X = 1, it remains in same state b because state b is the state which have detected 1.

At state c, if X = 0, the input stream is 100, so the output is high and it goes to state a since it has to detect another 1 from starting bit of sequence 100. But if X = 1, it moves to state b because state b has detected 1.

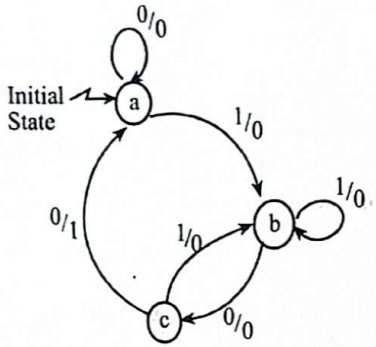


Fig.: State transition diagram

Step 3: Next state table (transition table)

Previous State	Next State		Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	0
c	a	b	1	0

Step 4: No reduction required.

Step 5: Binary assignment

Let a = 00, b = 01, c = 10

Step 6: No. of flipflops = 2. We design using JK flipflop.

Present State		Present Input	Next State		Output	Flipflop Excitation			
B_n	A_n	X	B_{n+1}	A_{n+1}	X	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	×
0	1	0	1	0	0	1	×	×	1
0	1	1	0	1	0	0	×	×	0
1	0	0	0	0	1	×	1	0	×
1	0	1	0	1	0	×	1	1	×

Here, 11 is the unused table, so values corresponding to it are don't care conditions.

Step 7:

For J_A

B_n \ A_n	00	01	11	10
0	0	0	0	1
1	×	×	×	×

$J_B = A_n\overline{X}$

For K_B

B_n \ A_n	00	01	11	10
0	×	×	×	×
1	1	1	×	×

$K_B = 1$

For J_A

B_n \ An x	00	01	11	10
0	0	1	×	×
1	0	1	×	×

$J_A = X$

For Y_1

B_n \ An x	00	01	11	10
0	0	0	0	0
1	0	1	×	×

$Y = B_nX$

For K_A

B_n \ An x	00	01	11	10
0	×	×	0	1
1	×	×	×	×

$K_A = \overline{X}$

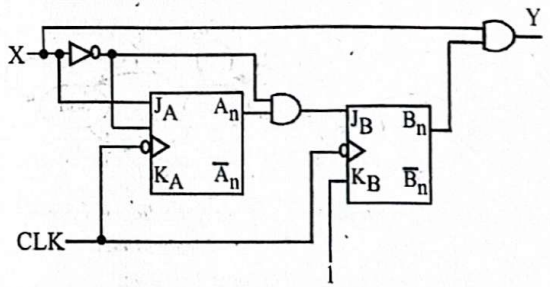


Fig.: Logic diagram of '100' sequence detector

2. **Design a sequential detector that receives binary data stream at its input 'X' and signals when a combination '011' arrives at the input by making its output 'Y' high which otherwise remains low. Consider data is coming from left i.e., the first bit to be identified is 1, second 1, and third 0 from the input sequence.**

⇒ The state diagram is shown below.

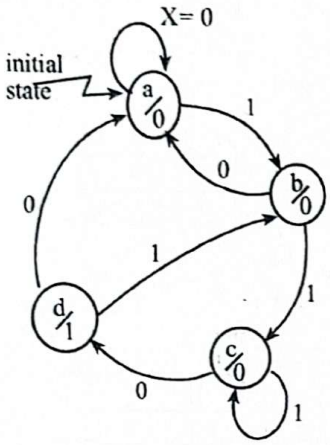


Fig.: State diagram of 011 sequence detector using Moore model

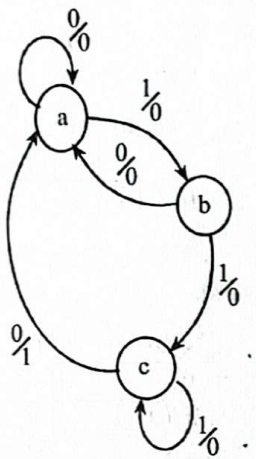


Fig.: State diagram of 011 sequence detector using Mealy model

Note: Students are requested to solve their own.

3. Design a sequential machine which has single output Y such that it is set high if the input sequence 'X' contains odd no. of 1's otherwise low.

⇒ The state diagram is shown below.

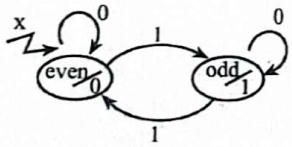


Fig.: State diagram for Moore Model

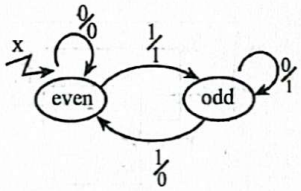


Fig.: State diagram for Mealy Model

Note: Students are requested to solve their own.

4. Consider a machine that has a single input 'X' and the clock, and two outputs A and B. On consecutive rising edge of the clock, the code on A and B changes from 00 to 01 to 10 to 11 and repeats itself when 'X' is ENABLED. If any time 'X' is DISABLED, this machine is supposed to hold its present state. Design the sequential machine.

⇒ The state diagrams using both models are shown below.

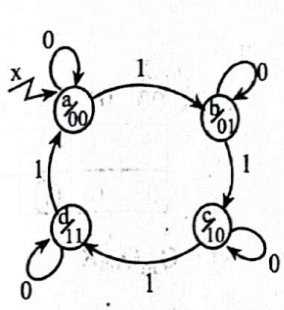


Fig.: State diagram for Moore Model

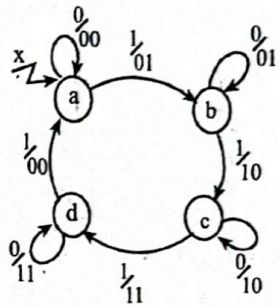


Fig.: State diagram for Mealy Model

Let's design the circuit from Mealy model.

Step 3: State table (transition table)

Present State	Next State		Output			
	X = 0	X = 1	X = 0		X = 1	
			A	B	A	B
a	a	b	0	0	0	1
b	b	c	0	1	1	0
c	c	d	1	0	1	1
d	d	a	1	1	0	0

Step 4: No more reduction required.

Step 5: Binary assignment

Let a = 00, b = 01, c = 10, d = 11

Step 6: Let's design with RS–flipflop, No. of flipflops = 2

Present State		Present input	Next State		Output		FF Excitation			
Q_{1n}	Q_{0n}	X	Q_{1n+1}	Q_{0n+1}	A	B	S_1	R_1	S_0	R_0
0	0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	1	0	×	1	0
0	1	0	0	1	0	1	0	×	×	0
0	1	1	1	0	1	0	1	0	0	1
1	0	0	1	0	1	0	×	0	0	×
1	0	1	1	1	1	1	×	0	1	0
1	1	0	1	1	1	1	×	0	×	0
1	1	1	0	0	0	0	0	1	0	1

Step 7:

For S_1

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	0	1	0
1	×	×	0	×

$S_1 = \overline{Q}_{1n}Q_{0n}X$

For R_1

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	×	0	×
1	0	0	1	0

$R_1 = Q_{1n}Q_{0n}X$

For A(o/p)

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	0	1	0
1	1	1	0	1

$A = Q_{1n}\overline{Q}_{0n} + Q_{1n}\overline{X} + \overline{Q}_{1n}Q_{0n}X$

For S_0

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	1	0	×
1	0	1	0	×

$S_0 = \overline{Q}_{0n}X$

For R_0

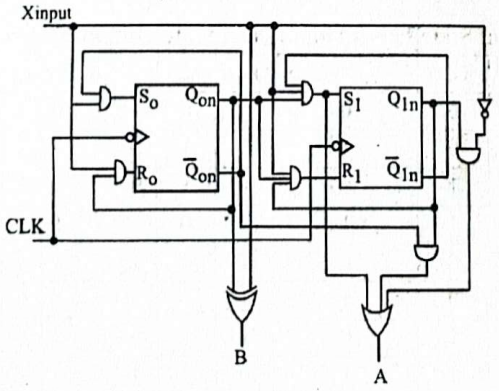
Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	×	0	1	0
1	×	0	1	0

$R_0 = Q_{0n}X$

For B (o/p)

Q_{1n} \ $Q_{0n}X$	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$B = \overline{Q}_{0n}X + Q_{0n}\overline{X}$
$= (Q_{0n} \oplus X)$



5. **A machine has two serial inputs X_1 and X_2 and one output Z. The machine is required to give an output Z = 1 when both X_1 and X_2 contain the message 011.**

⇒

Step 1: Mealy model

Step 2: State diagram

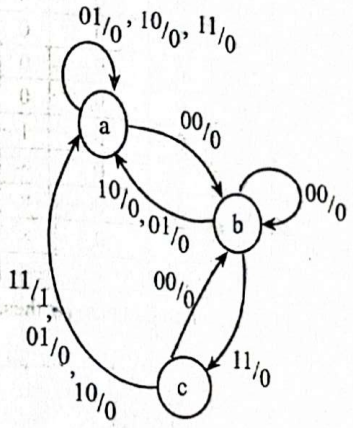


Fig.: State transition diagram for Mealy model

Step 3: Next state table

Present State	Next State				Present Output (Z)			
	X_1X_2 00	X_1X_2 01	X_1X_2 10	X_1X_2 11	X_1X_2 00	X_1X_2 01	X_1X_2 10	X_1X_2 11
a	b	a	a	a	0	0	0	0
b	b	a	a	c	0	0	0	0
c	b	a	a	a	0	0	0	1

Step 5: Binary assignment for states

a = 00, b = 01, c = 10

Step 6: No. of flipflops = 2

Let's design the machine from JK-flipflop.

Let A be o/p of FF J_A K_A &B be o/p of J_B K_B

Present State		Present Input		Next State		Present output	Flipflop Excitation			
B_n	A_n	X_1	X_2	B_{n+1}	A_{n+1}	Z	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	0	×	1	×
0	0	0	1	0	0	0	0	×	0	×
0	0	1	0	0	0	0	0	×	0	×
0	0	1	1	0	0	0	0	×	0	×
0	1	0	0	0	1	0	0	×	×	0
0	1	0	1	0	0	0	0	×	×	1
0	1	1	0	0	0	0	0	×	×	1
0	1	1	1	1	0	0	1	×	×	1
1	0	0	0	0	1	0	×	1	1	×
1	0	0	1	0	0	0	×	1	0	×
1	0	1	0	0	0	0	×	1	0	×
1	0	1	1	0	0	1	×	1	0	×

Here, 1100, 1101 1110, 1111 are unused states, so these are considered as don't care conditions.

Step 7: K-mapping

For J_B

B_nA_n \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	×	×	×	×
10	×	×	×	×

$J_B = A_nX_1X_2$

For K_B

B_nA_n \ X_1X_2	00	01	11	10
00	×	×	×	×
01	×	×	×	×
11	×	×	×	×
10	1	1	1	1

$K_B = 1$

For o/p Z

B_nA_n \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	×	×	×	×
10	0	0	1	0

$Z = BnX_1X_2$

For J_A

B_nA_n \ X_1X_2	00	01	11	10
00	1	0	0	0
01	×	×	×	×
11	×	×	×	×
10	1	0	0	0

$J_A = \overline{X}_1 \cdot \overline{X}_2$

For K_A

B_nA_n \ X_1X_2	00	01	11	10
00	×	×	×	×
01	0	1	1	1
11	×	×	×	×
10	×	×	×	×

$K_A = X_2 + X_1$

Step 8:

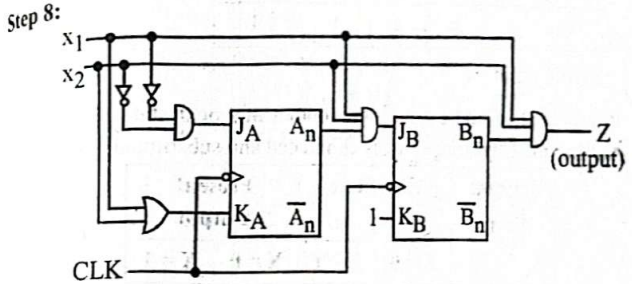


Fig.: Logic circuit diagram

State Reduction Technique

Redundant States

Two states are equivalent if they have same next state and output conditions. If they are equivalent, one of them is redundant i.e. can be eliminated or replaced. The necessary condition for equivalent is "output must be same".

Row elimination method

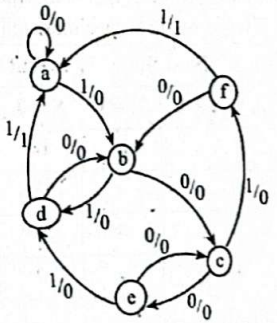


Fig.: State diagram to be reduced

Let's suppose the sequential machine state diagram as follows:

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

a. Original table

In table (a), b and e have same output and next states, so they are equivalent. Therefore e row is eliminated and substituted e by b.

Present State	Next State		Present Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	d	0	0
c	b	f	0	0
d	b	a	0	1
f	b	a	0	1

b. After one row elimination

d and f states also have same output and same next states, so they are also equivalent. Therefore, f row is eliminated and substituted f by d.

Present State	Next State		Presnt Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	d	0	0
c	b	d	0	0
d	b	a	0	1

c. After two row elimination

Here, for X = 0, b → c and c → b, the states c and d are equivalent

Present State	Next State		Present O/P	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	b	d	0	0
d	b	a	0	1

d. Final reduced table after three row elimination

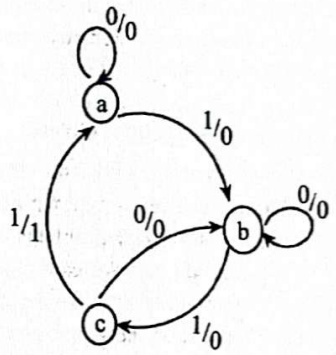


Fig.: Reduced state diagram

9.3 Asynchronous Sequential Circuit Design

Asynchronous sequential circuit, also called "event driven circuit" does not have any clock to trigger change of state.

Example:

AND Gate

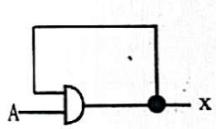


Fig.: AND gate

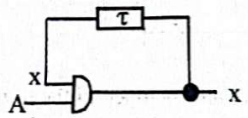


Fig.: AND gate with propagation delay

x \ A	0	1
0	0	0
1	0	1

X = x.A

The two input AND gate with output fed back as one input is shown in **fig.** (a). The circuit can be redrawn shown in **fig.** (b) that includes propagation delay of gate where τ is the finite time after which gate reacts to its input. In the truth table shown above, encircled states indicate the stable condition of the circuit.

For example, if A = 0 and X = 0, then, X = x.A = 0.0 = 0 after time T = τ, x takes the value of X i.e., 0 and because of that output X does not change. Thus, X = 0, A = 0 represents a stable state and is encircled. Similarly x = 0, A = 1 position and x = 1, A = 1 positions are also stable as in each of these cases X = x and no change in output is necessary. But at x = 1, A = 0, X becomes 0 and after time τ, x becomes 0 i.e., we move up by one position in Karnaugh map to x = 0, A = 0 position.

We make an important observation from this discussion which is universally true for asynchronous sequential circuit. For any state, if x = X then the circuit is stable and if x ≠ X it is unstable.

Problems with Asynchronous Sequential Circuit

Asynchronous circuit responds to all the transient values and problems like oscillation, critical race, hazards can cause major problems unless they are addressed at design stage. To explain these problems, we take help of truth table shown below, where the circuit has two external inputs A, B and two outputs X, Y. Both the outputs are fed back to the input side in the form of x and y but with different propagation delays. Thus x, y cannot change simultaneously but with time delays τ_1, and τ_2 respectively and we can write

$x = X(t - \tau_1)$

$y = Y(t - \tau_2)$

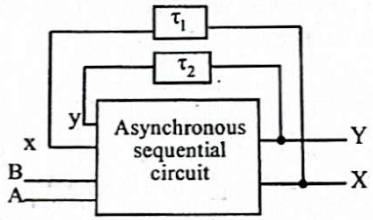


Fig.: Block diagram

xy \ AB	00	01	11	10
00	(00)	11	(00)	01
01	00	(01)	00	(00)
11	(11)	(11)	00	(11)
10	(10)	11	00	(10)

The stable states are encircled in the truth table where xy = XY. The major problems with this truth table are discussed below.

i) Oscillation

Consider the stable state xyAB = 0000, where x = X and y = Y. If input AB changes from 00 to 10, the circuit goes to xyAB = 0010 state and then output XY = 01. This is a transient state because xy ≠ XY. After time τ_2, y takes the value of Y = 1 and the circuit goes to xyAB = 0110 where output XY = 00. This is again a transient state and after another propagation delay of τ_2, the circuit goes to xyAB = 0010. Thus, the circuit oscillates between state 0010 and 0110, and the output Y oscillates between 0 and 1 with a gap τ_2. In asynchronous sequential circuits for any given input, transitions between two unstable states like these are to be avoided to remove oscillation.

ii) Critical race

This occurs when an input tries to modify more than one output. In the truth table shown above, consider the stable state xyAB = 0000. Now, if AB changes to 01, the circuit moves to xyAB = 0001 where XY = 11. Now, depending which of τ_1 and τ_2 is lower, xy moves from 00 to either 01 or 10. If τ_1 is lower, x changes earlier and the circuit goes to xyAB = 1001 which is an unstable state with XY = 11 and xy ≠ XY. The circuit next moves to state xyAB = 1101 which is a stable state and final output XY = 11. If τ_2 is lower, y changes earlier and the circuit goes to xyAB = 0101 which is a stable state and final output XY = 01. Thus, depending on propagation delays in feedback path, the circuit settles at two different states generating two different set of output. Such situation is called critical race condition and is to be avoided in asynchronous sequential circuit.

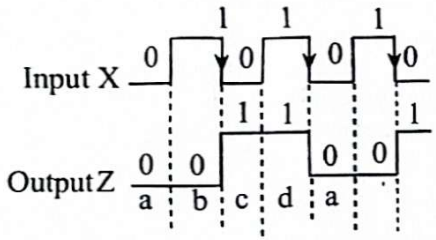
iii) Hazards:

Static and dynamic hazards cause malfunctioning of asynchronous sequential circuit. Situations like Y = A+A or Y = A.A are to be avoided for any input output combination with the help of hazards covers in truth table. In circuit with feedback, even when those hazards are adequately covered there can be another problem called "essential hazard". This occurs when change in input doesn't reach one part of the circuit while from other part one output fed back to the input side becomes available. Essential hazard is avoided by adding delay, may be in the form of additional gates that does not change the logic level in the feedback path.

EXAMPLES

1. **Design frequency divide by 2 counter from asynchronous machine.**

⇒ The waveform for the divide 2 counter is



State diagram:

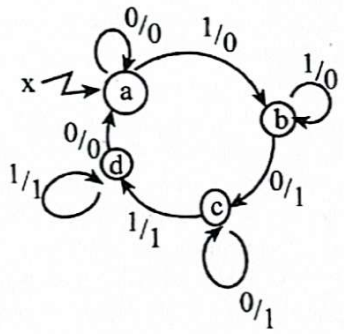


Fig.: State diagram using Mealy model

Primitive flow table (or simply flow table):

Present State	Present Input (X)	Next state	Output
a	0	a	0
a	1	b	0
b	0	c	1
b	1	b	0
c	0	c	1
c	1	d	1
d	0	a	0
d	1	d	1

Binary assignment:

To avoid uncertainty change of secondary variable, we assign binary value so that only one bit changes.

i.e., a = 00, b = 01, c = 11, d = 10

Excitation table:

Present State		Present Input	Next state		Output
y_1	y_2	X	Y_1	Y_2	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	1

For asynchronous machine design 'Next State' is the excitation table. So, plot K-map for Y_1 and Y_2.

For Y_1

y_1 \ y_2X	00	01	11	10
0	0	0	0	1
1	0	1	1	1

$y_1 = y_1x + y_1y_2 + y_2\overline{x}$

For Y_2

y_1 \ y_2X	00	01	11	10
0	0	1	1	1
1	0	0	0	1

$y_2 = \overline{y}_1x + \overline{y}_1y_2 + y_2\overline{x}$

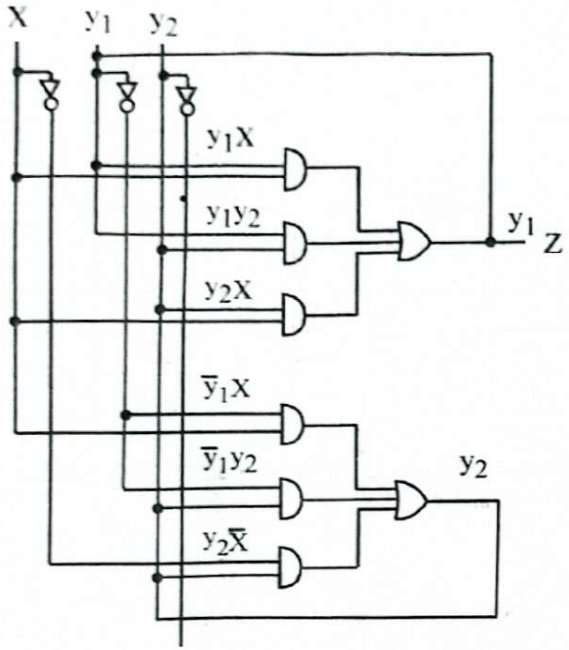
For Z

y_1 \ y_2X	00	01	11	10
0	0	0	0	1
1	0	1	1	1

→ to avoid hazard

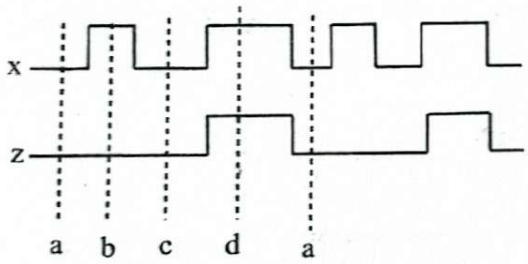
$Z = y_1 \times y_1y_2 + y_2\overline{x} = y_1$

Logic diagram:



2. **Consider a circuit with one input (X) and one output (Z). The output should suppress every alternative pulse on the input starting with the first.**

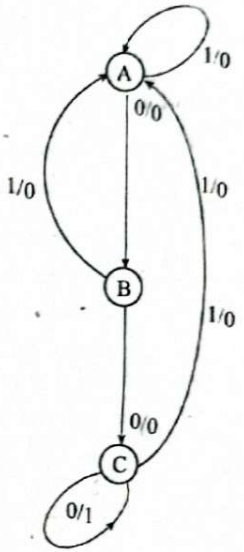
⇒



MORE WORKED OUT EXAMPLES:

1. **Design a sequential machine that detects three consecutive zeroes from an input data stream X by making output Y =1.[2069 Chaitra]**

⇒



Transition table

Present state	Next state		Output	
	At X = 0	At X = 1		
X_1X_2	Y_1	Y_2	At X = 0	At X = 1
A	B	A	0	0
B	C	A	0	0
C	C	A	1	0

Assigning the binary values

A → 00

B → 01

C → 10

Present state		Next state (Y_1Y_2)				Output	
X_1	X_2	At X = 0		At X = 1		At X = 0	At X = 1
0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
1	0	1	0	0	0	1	0

Excitation table of D flip-flop

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Present state		Input	Next State		D_A	D_B
X_1	X_2	X	Y_1	Y_2		
0	0	0	0	1	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	0	0	0
1	1	0	×	×	×	×
1	1	1	×	×	×	×

For D_A

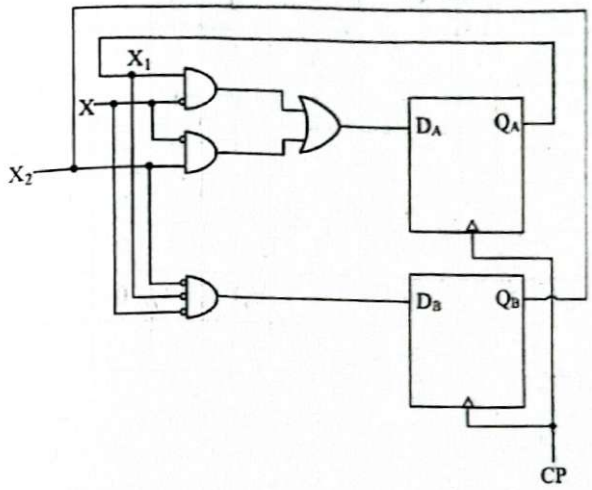
X_1 \ X_2X	00	01	11	10
0	0	0	0	1
1	1	0	×	×

$D_A = X_1X_2' + X_2'X'$

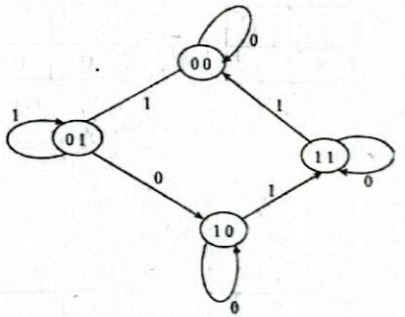
For D_B

X_1 \ X_2X	00	01	11	10
0	1	0	0	0
1	0	0	×	×

$D_B = X_1'X_2'X'$



2. **Design synchronous sequential circuit for the given state diagram using JK flip-flop.** **[2069 Ashadh]**



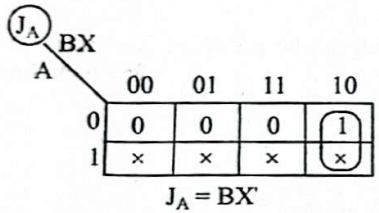
⇒ **Excitation table for JK flip-flop**

Q_t	Q_{t-1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

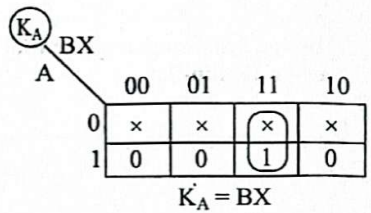
State table:

Present state		Input control X	Next state		J_A	K_A	J_B	K_B
A	B		C	D				
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	1	0	1	×	×	1
0	1	1	0	1	0	×	×	0
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

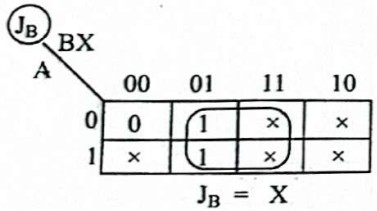
Now, mapping by using K-map



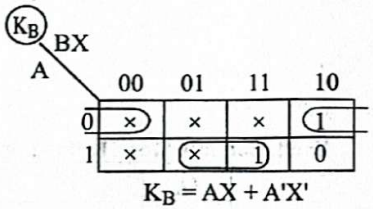
$J_A = BX'$



$K_A = BX$



$J_B = X$



$K_B = AX + A'X'$

$= A \odot X$

Circuit diagram

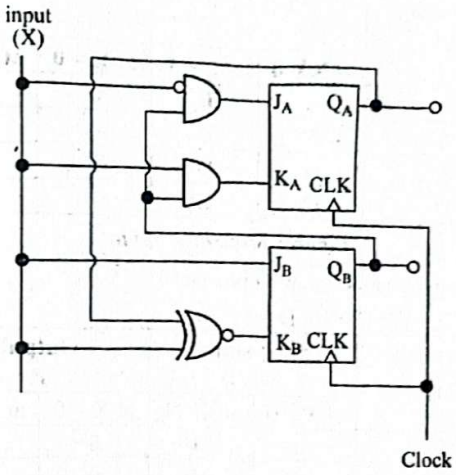


Fig.: Detail Circuit Layout

3. **Design a sequential machine that has one serial input and one output Z. The machine is required to give an output Z = 1 when the input X contains the message 1010. [2068 Chaitra]**

⇒ Here input X = 1010, Output Z = 0001

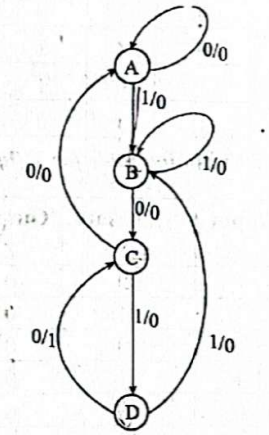


Fig.: State diagram

Present state Y_1Y_2	Next state Y_1Y_2		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
A	A	B	0	0
B	C	B	0	0
C	A	D	0	0
D	C	B	1	0

Figure: Transition table.

Assigning the values on pure binary

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

Present state Y_1Y_2	Next state Y_3Y_4		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
00	00	01	0	0
01	10	01	0	0
10	00	11	0	0
11	10	01	1	0

Fig.: Pure binary assignment

Now, designing by using T flip–flop

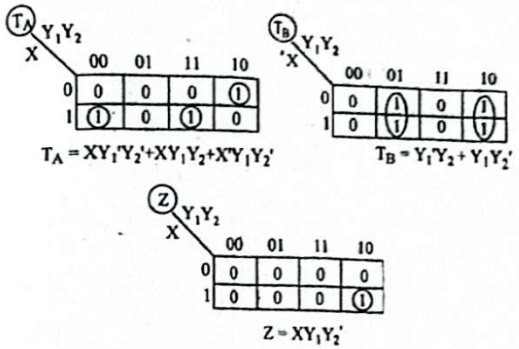
Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

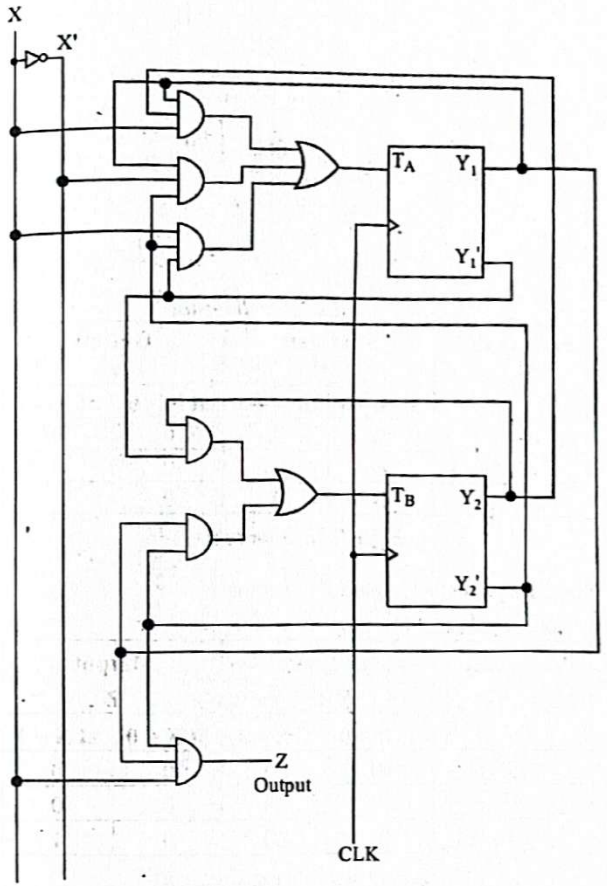
Fig.: Excitation table for T flip-flop

Present State Y_1Y_2	Input X	Next State Y_3Y_4	Output Z	T_A	T_B
00	0	00	0	0	0
00	1	01	0	0	1
01	0	10	0	1	1
01	1	01	0	0	0
10	0	00	0	1	0
10	1	11	0	0	1
11	0	10	1	0	1
11	1	01	0	1	0

Figure: State table

Now, using K–map





4. **A synchronous state machine has one bit serial input X. The output Z of machine is to be set high when the input contains the message 011. Draw the state diagram for this machine and design the circuit. (Use T flip–flop). [2068 Shrawan]**

⇒ Input X = 011, output Z = 001

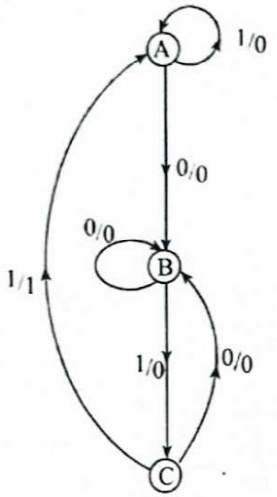


Fig.: State diagram

Present state Y_1Y_2	Next state Y_3Y_4		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
A	B	A	0	0
B	B	C	0	0
C	B	A	0	1

Fig.: Transition table

Now, assigning the values on pure binary

A = 00, B = 01 and C = 10

Present state Y_1Y_2	Next state Y_3Y_4		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
00	01	00	0	0
01	01	10	0	0
10	01	00	0	1

Fig.: Pure binary assignment

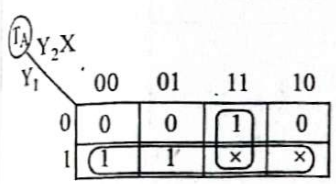
We know, transition table for T flip–flop

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

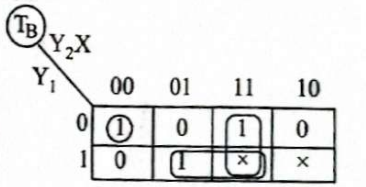
Present State Y_1Y_2	Input X	Next State Y_3Y_4	Output Z	T_A	T_B
00	0	01	0	0	1
00	1	00	0	0	0
01	0	01	0	0	0
01	1	10	0	1	1
10	0	01	0	1	1
10	1	00	1	1	0

Fig.: State table.

Now, using K–map.

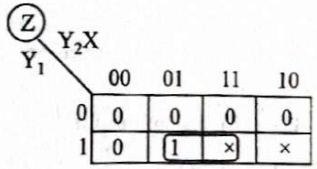


$T_A = Y_1 + Y_2X$



$T_B = Y_2'X' + Y_2X + Y_1Y_2X$

$= Y_2 \odot X + Y_1Y_2X$



$Z = Y_1X$

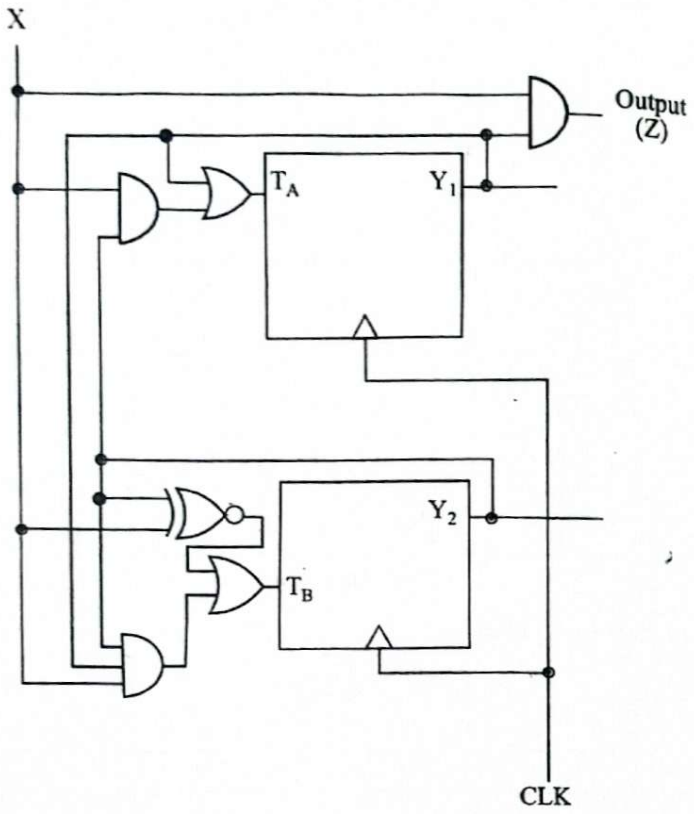


Fig.: Detail circuit diagram

5. Design a synchronous state machine with the following specifications:

a) No. of input: 1 b) No. of output: 2

c) The output of the machine is to be set high when the data in the input 110 in sequence starting from MSB (use SR flip-flop). [2068 Baishakh]

⇒ Input X = 110

Output Z = 001

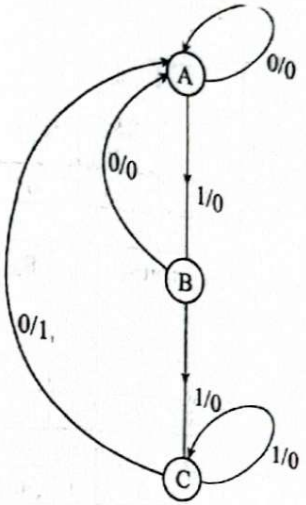


Fig.: State diagram from given parameters

Present state Y_1Y_2	Next state Y_3Y_4		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
A	A	B	0	0
B	A	C	0	0
C	A	C	1	0

Figure: Transition table

Now, assigning the values on pure binary

A = 00; B = 01; C = 10

Present state Y_1Y_2	Next state Y_3Y_4		Output Z	
	at X = 0	at X = 1	at X = 0	at X = 1
00	00	01	0	0
01	00	10	0	0
10	00	10	1	0

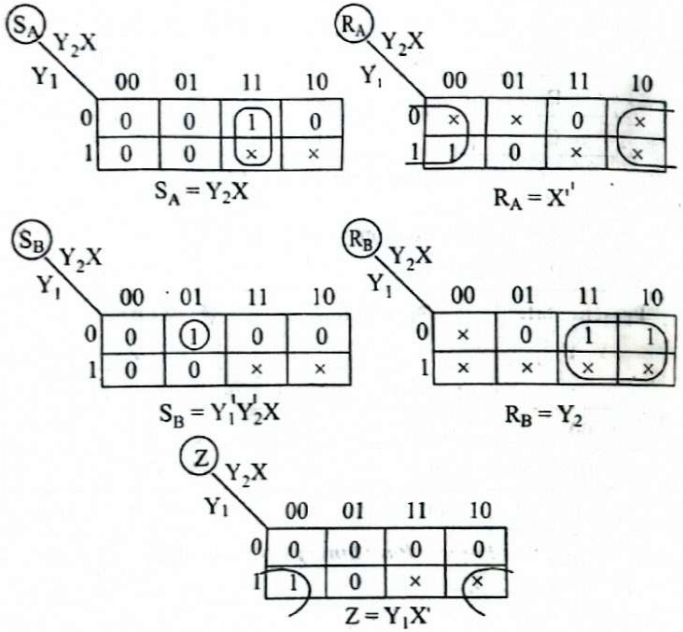
Figure: Pure binary assignment

We know the excitation table for SR flip-flop

Q_t	Q_{t+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

Present state $Y_1 Y_2$	Input control X	Next state $Y_3 Y_4$	Output Z	S_A	R_A	S_B	R_B
00	0	00	0	0	×	0	×
00	1	01	0	0	×	1	0
01	0	00	0	0	×	0	1
01	1	10	0	1	0	0	1
10	0	00	1	0	1	0	×
10	1	10	0	×	0	0	×

Fig.: State table



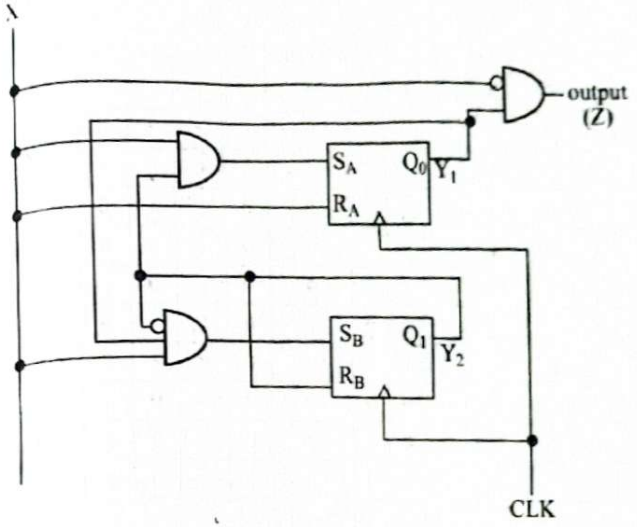
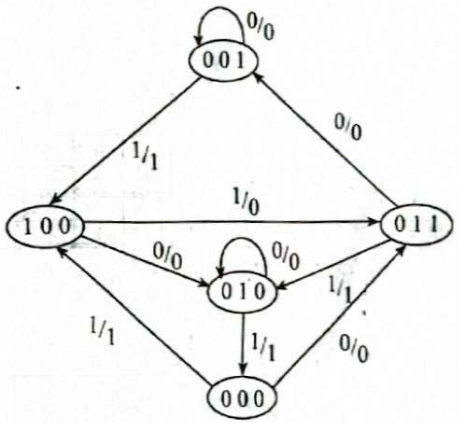


Fig.: Detail logic diagram

6. **A sequential circuit has one input and one output. The state diagram is shown in figure. Design the sequential circuit with RS flip-flops.** **[2067 Ashadh]**



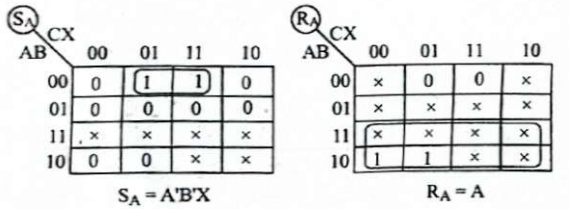
⇒ **State synthesis table:**

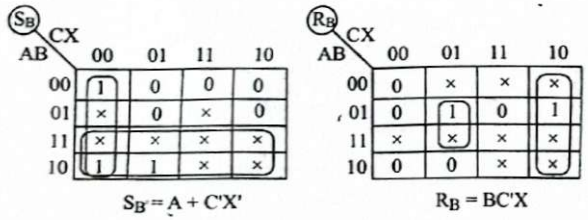
Present State			Input	Next state			Output	S_A	R_A	S_B	R_B	S_C	R_C
A	B	C	X	D	E	F	Y						
0	0	0	0	0	1	1	0	0	×	1	0	1	0
0	0	0	1	1	0	0	1	1	0	0	×	0	×
1	0	0	0	0	1	0	0	0	1	1	0	0	×
1	0	0	1	0	1	1	0	0	1	1	0	1	0
0	1	1	0	0	0	1	0	0	×	0	1	×	0
0	1	1	1	0	1	0	1	0	×	×	0	0	1
0	1	0	0	0	1	0	0	0	×	×	0	0	×
0	1	0	1	0	0	0	1	0	×	0	1	0	×
0	0	1	0	0	0	1	0	0	×	0	×	×	0
0	0	1	1	1	0	0	1	1	0	0	×	0	1

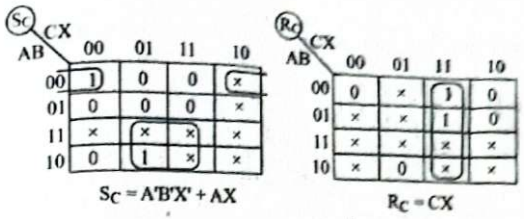
We know, the excitation table for SR flip–flop:

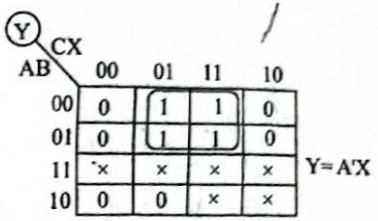
Q	Q_{t+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

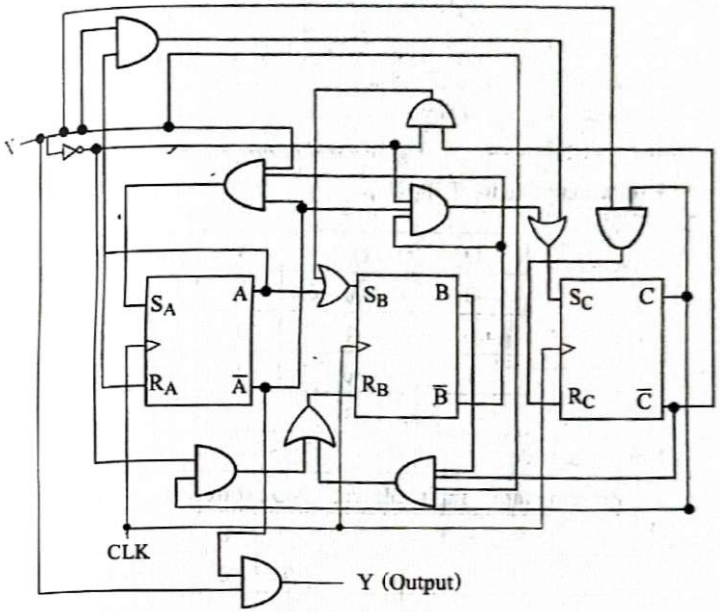
Using K–map





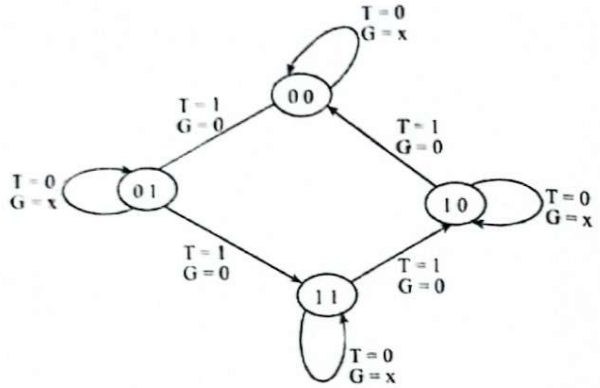






7. **Design a synchronous sequential machine with appropriate number of T flip–flops having A and B two outputs and two control inputs T and G. The System to be designed remains in the same state (i.e., no change) while T = 0 and G = X (don't care). The state machine goes through the state transition from 00 to 10 to 11 to 01 back to 00 and repeats when given control inputs become T = 1 and G = 0.** **[2064 Falgun]**

⇒ Drawing state diagram according to the information given,



Note:

No change T = 0 and G = ×

Next state T = 1 and G = 0

Fig.: State diagram

Excitation table for T flip flop

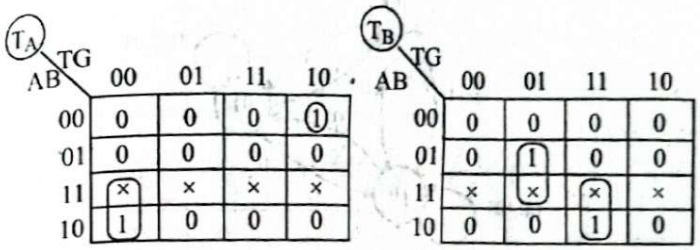
Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Now, state table:

Present state		Input control		Next state		T_A	T_B
A	B	T	G	C	D		
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	1	0	1
1	1	0	0	1	1	0	0

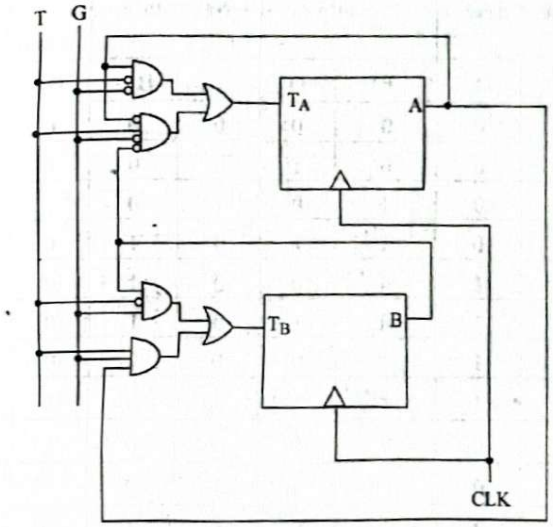
Present state		Input control		Next state		T_A	T_B
A	B	T	G	C	D		
1	1	0	1	1	1	0	0
1	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	1

Now, Using K–map for T_A and T_B



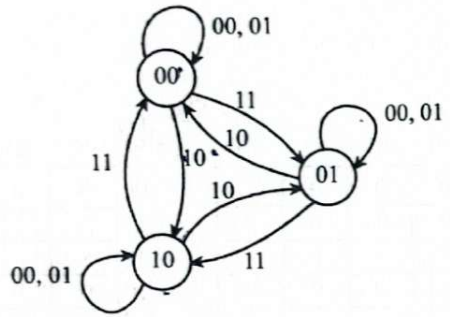
$T_A = AT'G' + AB'TG'$

$T_A = BT'G + ATG$



8. **Design a sequential circuit with two D flip-flops and two inputs, P and Q. If P = 0, the circuit remains in the same state regardless of the value of Q. When P = 1 and Q = 1, the circuit goes through the state transitions from 00 to 01 to 10 back to 00, and repeats. When P = 1 and Q = 0, the circuit goes through the state transitions from 00 to 10 to 01 back to 00, and repeats. The circuit is to be designed by treating the unused state(s) as don't care condition(s).** **[2071 Chaitra]**

⇒



State diagram

Present state		Present i/p		Next state		FF excitation	
A_n	B_n	P	Q	A_{n+1}	B_{n+1}	D_A	D_B
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	0	0	0

For D_A

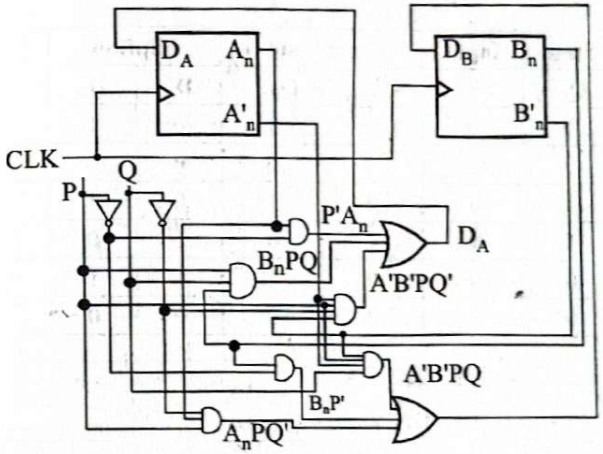
A_nB_n \ PQ	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	X	X	X	X
10	1	1	0	0

$\therefore$ $D_A = A_nP' + B_nPQ + A'_nB'_nPQ'$

For D_B

A_nB_n \ PQ	00	01	11	10
00	0	0	1	0
01	1	1	0	0
11	X	X	X	X
10	0	0	0	1

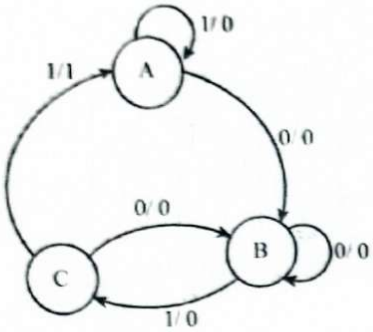
$D_B = B_nP' + A_nPQ' + A'_nB_n'PQ$



9. **Design a synchronous sequential machine such that it gives output Z = 1 if it detects input message 011. Use D flipflop.**

[2073 Chaitra]

⇒

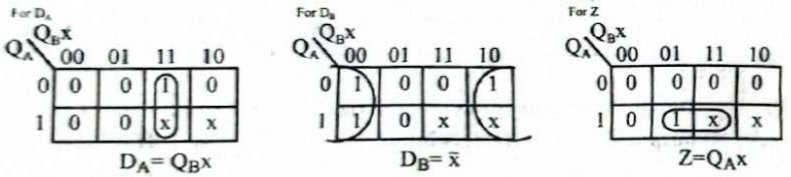


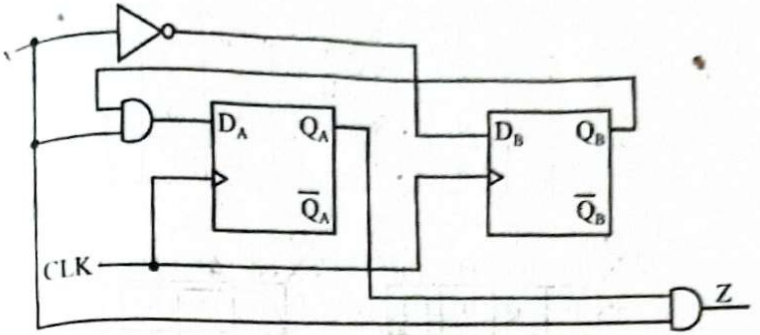
Excitation table for D flipflop

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

Let A = 00, B= 01, C = 10

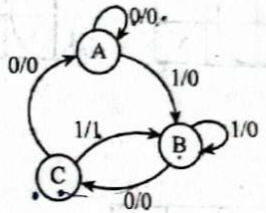
Present State		Input	Next State		Flipflop		Output
Q_{An}	Q_{Bn}	x	Q_{An+1}	Q_{Bn+1}	D_A	D_B	Z
0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	0
0	1	0	0	1	0	1	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	1	0
1	0	1	0	0	0	0	1





10. **A sequential machine has to detect serial input sequence of 101, for which the machine output will be high. The machine contains two JK flipflops, A and B. Assume: single input, X and single output, Y.** **[2073 Shrawan]**

⇒



Excitation table for JK flip flop

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

Let A = 00, B = 01, C = 10

Present State		Input	Next State		Flipflop				Output
Q_{An}	Q_{Bn}	x	Q_{An+1}	Q_{Bn+1}	J_A	K_A	J_B	K_B	Y
0	0	0	0	0	0	×	0	×	0
0	0	1	0	1	0	×	1	×	0

0	1	0	1	0	1	×	×	1	0
0	1	1	0	1	0	×	×	0	0
1	0	0	0	0	×	1	0	×	0
1	0	1	0	1	×	1	1	×	1

For J_A

Q_A \ Q_BX	00	01	11	10
0	0	0	0	1
1	x	x	x	x

$\therefore J_A = Q_B\bar{x}$

For K_A

Q_A \ Q_BX	00	01	11	10
0	x	x	x	x
1	1	1	x	x

$\therefore K_A = 1$

For J_B

Q_A \ Q_BX	00	01	11	10
0	0	1	x	x
1	0	1	x	x

$J_B = x$

For K_B

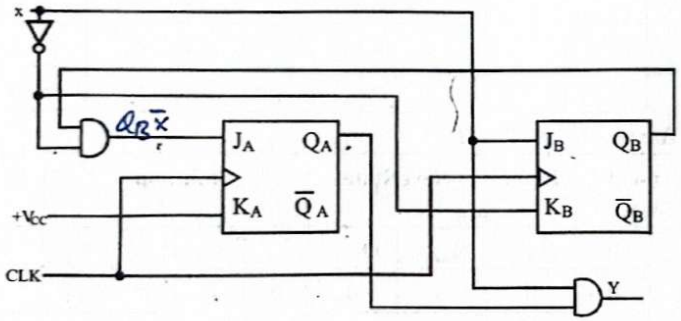
Q_A \ Q_BX	00	01	11	10
0	x	x	0	1
1	x	x	x	x

$K_B = \bar{x}$

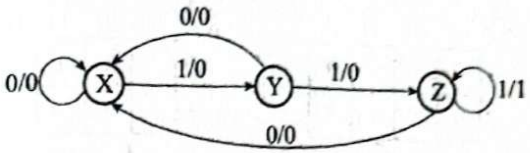
For Y

Q_A \ Q_BX	00	01	11	10
0	0	0	0	0
1	0	1	x	x

$Y = Q_Ax$



11. **Design a synchronous sequenstial machine from the state diagram given below. Use S–R flip–flop.** **[2075 Ashwin]**



⇒ From given state diagram, we will draw the next state table,

Present state	Input	Next state	Output
X	0	X	0
X	1	Y	0
Y	0	X	0
Y	1	Z	0
Z	0	X	0
Z	1	Z	1

Then we will assign X = 00, Y = 01, Z = 10

Let's design with SR-flip flop.

No. of flip-flops = 2

Present state		Input	Next state		Output	Excitation table			
B_n	A_n	C	B_{n+1}	A_{n+1}	D	S_B	R_B	S_A	R_A
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	0
0	1	0	0	0	0	0	×	0	1
0	1	1	1	0	0	1	0	0	1
1	0	0	0	0	0	0	1	0	×
1	0	1	1	0	1	×	0	0	×

For S_B

B_n \ A_nC	00	01	11	10
0	0	0	1	0
1	0	×	×	×

$S_B = A_nC$

For R_B

B_n \ A_nC	00	01	11	10
0	×	×	0	×
1	1	0	×	×

$R_B = \bar{C}$

For S_A

B_n \ A_nC	00	01	11	10
0	0	1	0	0
1	0	0	×	×

$S_A = \bar{B}_n\bar{A}_nC$

For R_A

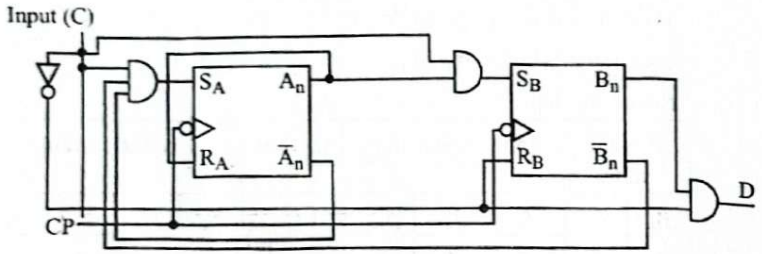
B_n \ A_nC	00	01	11	10
0	×	0	1	1
1	×	×	×	×

$R_A = A_n$

For output (D)

B_n \ A_nC	00	01	11	10
0	0	0	0	0
1	0	1	×	×

$D = B_nC$



Chapter – 10

DIGITAL INTEGRATED CIRCUITS

10.1 Introduction

A digital IC is constructed by an interconnection of resistors, transistors, and perhaps small capacitors, all of which have been formed on the surface of a silicon wafer. The entire circuit resides on a tiny piece of semiconductor material called a chip.

Different types of IC families are listed below:

1. Resist transistor logic (RTL)
2. Diode Transistor logic (DTC)
3. Integrated Injection logic (I^2L)
4. Transistor Transistor logic (TTL)
5. Schottky TTL
6. NMOS
7. CMOS

Characteristics of digital ICs include:

- **Propagation delay (speed of operation)**

  The output of logic gate does not change its state instantaneously in response to the change in state of its input. There is a time delay between these two events, which is called as propagation delay.

- **Power dissipation**

  As a result of applied voltage and current flowing through the logical ICs, some power will be dissipated in it, in the form of heat.

- **Current and voltage parameters**

  These parameters include $V_{IL,\,max}$, $V_{IH,\,min}$, $V_{OH,\,min}$, $V_{OL,\,max}$, I_{IL}, I_{IH}, I_{OH}, I_{OL}.

- **Noise immunity**

  It is the measure of how much stray noise voltage the device can handle without giving any error at the output level.

- **Fan-in and Fan-out**

  Fan-in is defined as the number of inputs a gate has. Fan-out is defined as the maximum number of inputs of the same IC family that a gate can drive without falling outside the specified output voltage limits.

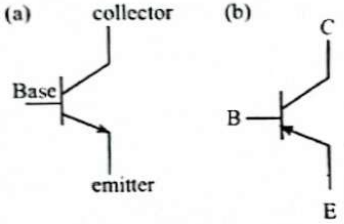
- **Operating temperature**

  The temperature range acceptable for the consumer and industrial applications is 0°C to 70°C and that for the military applications is −55°C to 125°C.

10.2 Switching Circuits

1) **Bipolar Junction Transistor (BJT)**

Bipolar junction transistor is available in two polarities.



Input (B)	npn	pnp
0V	OFF	ON
5V	ON	OFF

2) **Metal Oxide Semiconductor Field Effect Transistor (MOSFET)**

There are two polarities of MOSFETs: n-channel MOSFET (NMOS) and p-channel MOSFET (PMOS). They operate either in enhancement mode or depletion mode. The switch is activated by applying a voltage between gate and source.

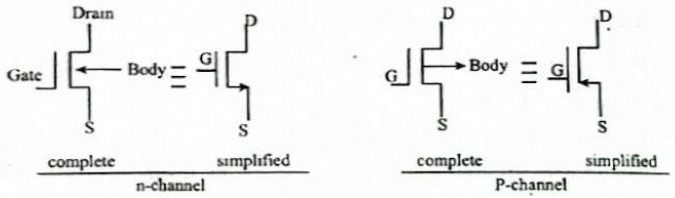


Fig.: Enhancement MOSFET

3) **Complementary Metal Oxide Semiconductor (CMOS) Field Effect Transistor**

ICs constructed entirely from n-channel MOSFET are called NMOS ICs. ICs constructed entirely from p-channel MOSFET are called PMOS ICs. ICs constructed using both PMOS and NMOS are called CMOS ICs.

10.3 External Drive for TTL Loads

Some of the ways to drive a TTL load are explained as follows:

1. **Switch drive**

Figure below shows the preferred method for driving a TTL input from a switch. With the switch open, the input is pulled up to +5 V. When the switch is closed, the input is pulled down to ground.

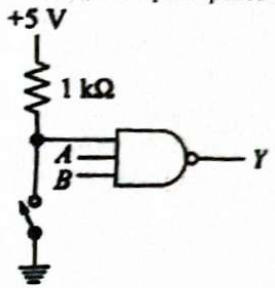


Fig.: Switch drive for TTL input

2. **Size of pull-up resistor**

The smaller the pull-up resistance, the larger the current drain. On the other hand, too large a pull-up resistance causes the time constant (RC) to be larger which in turn causes speed problems. Pull-up resistances between 1 and 10 KΩ are typical.

3. **Transistor drive**

In this case, we use a transistor switch to control the state of the TTL input. When V_i is low, the transistor is off and is equivalent to open switch. Then, the TTL input is pulled up to +5 V through a resistance of 1 KΩ. When V_i is high, the transistor is on and is equivalent to a closed switch. In this case, it easily sinks the 1.6 mA of input current.

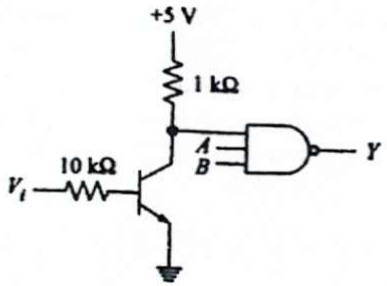


Fig.: Transistor drive for TTL input.

4. **Operational amplifier drive**
5. **Comparator drive**

10.4 74XX Series TTL

Characteristic of standard TTL are:

i) **High voltage range:** Normal supply voltage (V_{CC}) for both 54 and 74 series is 5V. 54 series operate reliably over the range of 4.5V to 5.5V but 74 series operates reliably over the range of 4.75V to 5.25V.

ii) **Temperature range:** 54 series operates properly in ambient temperature ranging from –55°C to +125°C while 74 series operates only in the range from 0°C to 70°C.

iii) **Power dissipation:** A standard TTL gate takes an average power of 10 miliwatts.

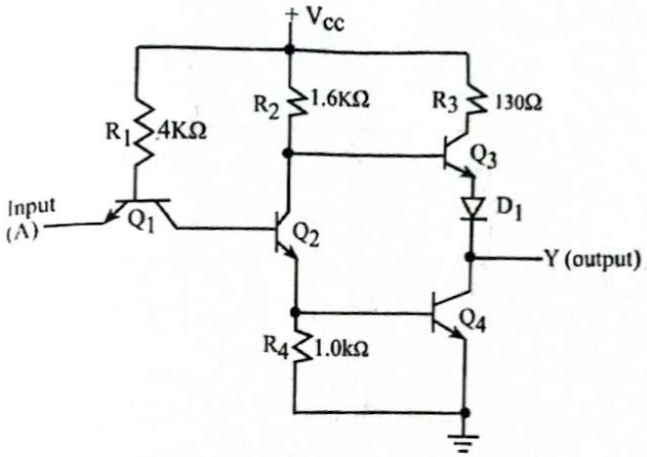
iv) **Propagation delay:** It is approximately 10 nanosecond.

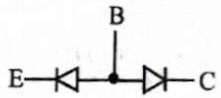
v) **Fan-out**: For standard TTL, fan-out is 10.

vi) **Noise margin:** The noise margin for the TTL family is 0.4V.

1. TTL Inverter (NOT Gate)

Totem-pole arrangement: Here, the combination of Q_3 and Q_4 transistors forms the output circuit often referred to as totem-pole arrangement.





npn transistor (Q_1)

When the input is HIGH, the base-emitter junction of Q_1 is reversed biased and the base-collector junction is forward biased. This condition permits current through R_1 and the base-collector junction of Q_1 into the base of Q_2, thus driving Q_2 into saturation (Q_2 'ON'). As a result, Q_4 is turned ON by Q_2, and its collector voltage, which is the output, is at near ground potential. Therefore, the output is LOW for HIGH input. At the same time, the collector of Q_2 is at LOW (ground) which keeps Q_3 OFF.

When the input is LOW, the base-emitter junction is forward biased and the base-collector junction is reversed biased. There is current through R_1 and the base-emitter junction of Q_1 to the LOW input. A LOW input provides path to ground for current. There is no current on the base of Q_2, so it is OFF. Thus, collector of Q_2 is HIGH, causing transistor Q_3 turned ON, meanwhile transistor Q_4 is turned OFF. Hence, the output is HIGH. Diode 'D_1' provides an additional V_{BE} (0.7 V) equivalent drop in series with base-emitter junction of Q_3 to ensure it is turned OFF when Q_2 is ON.

2. TTL NAND Gate

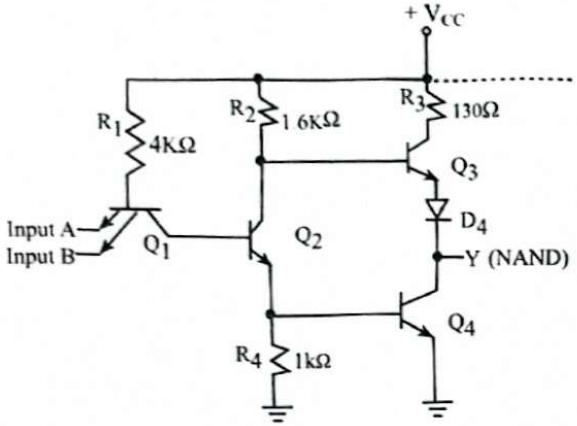


Fig.: A TTL NAND gate circuit

1. A LOW input on either input A or input B or both A and B forward biases the base-emitter junction of Q_1 and reverse biases the base-collector junction of Q_1 i.e., transistor Q_1 is saturated. This causes Q_2 to turn OFF, thereby causing Q_4 to turn OFF. Meanwhile, Q_3 is turned ON and the output is HIGH.
2. A HIGH on both input reverse biases the base-emitter junction of Q_1 and forward biases the base collector junction of Q_1. This causes Q_2 to turn ON which in turn makes transistor Q_4 ON and the output is LOW.

Inputs		State			Output
A	B	Q_2	Q_3	Q_4	Y
0	0	OFF	ON	OFF	1
0	1	OFF	ON	OFF	1
1	0	OFF	ON	OFF	1
1	1	ON	OFF	ON	0

3. TTL NOR Gate

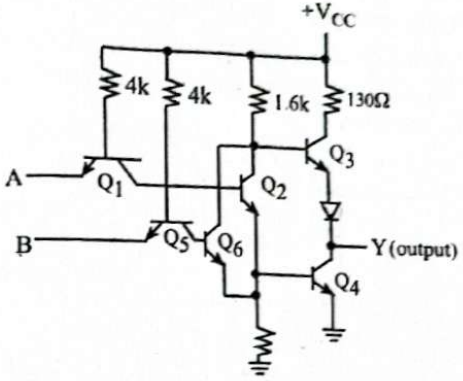


Fig.: TTL NOR gate circuit

Here, Q_5 and Q_6 have been added to basic NAND gate design. Since Q_6 and Q_2 are in parallel we get OR function which is followed by inversion to get NOR function.

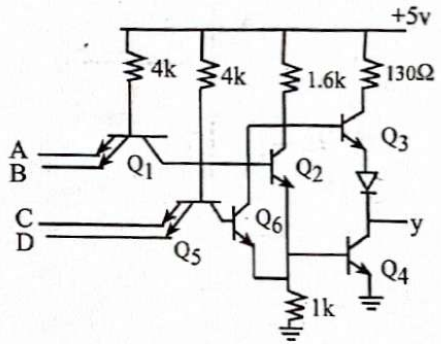
When A and B are LOW, the bases of Q_1 and Q_5 are pulled LOW, thus cuts off Q_2 and Q_6. Then, Q_3 is turned ON and Q_4 is OFF. Thus, output is HIGH.

When A or B is HIGH or both are HIGH, Q_1 and Q_5 are cut off respectively forcing Q_2 or Q_6 or both turned ON. This causes Q_4 to ON and pulling the output voltage to LOW.

4. TTL AND and OR Gate

To produce the AND function, another inverting stage is inserted to basic NAND. Similarly, to produce OR function, another inverting stage is inserted to basic NOR.

5. AND-OR-INVERT gate:



Here, Q_1 & Q_5 acts as 2 input AND gate, Q_2 & Q_6 produce ORing of output of Q_1 and Q_5, and Q_4 performs inversion.

6. **Open-Collector Gates**

Instead of totem-pole output, some TTL have an open collector output. For this kind of gate, an external pull-up resistor is required for proper operation.

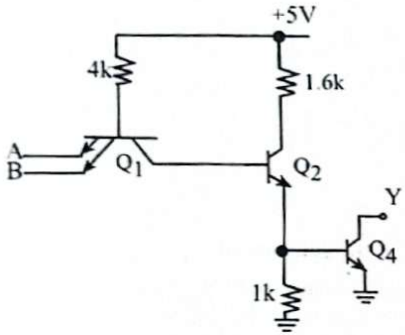
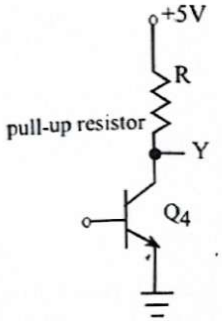


Fig (a) Open-collector NAND TTL



(b)pull-up resistor

Advantages:

It has the advantages of combining three devices without using a final OR gate (or AND gate). This is done through direct connecting of the '3' outputs to the lower end of common pull–up resistor.

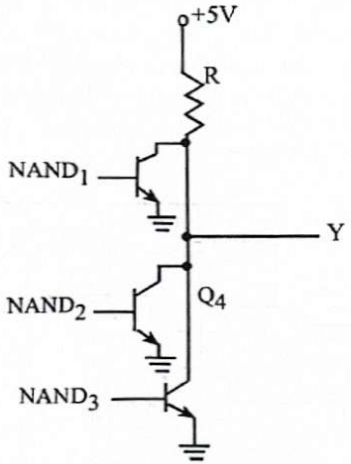


Fig.: Wired ANDing of three open collector TTL gates.

Disadvantages:

Open collector gate is slow in switching speed because the pull-up resistor are of few kiloohms which results in long time constant when multiplied to stray capacitance ($\tau = RC$). Another disadvantage is increased power dissipation.

7. **Tri-State TTL Gates**

For a standard TTL, while we wire more gates together there is an excessive flow of current that may destroy our TTL device. This led to new device called tri-state TTL.

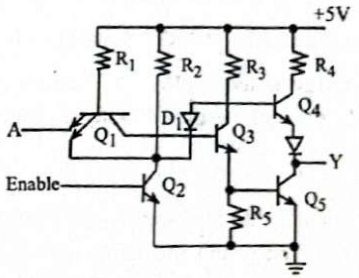
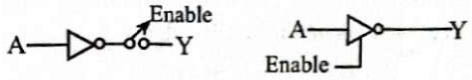


Fig.: Basic tri-state inverter



(a) Equivalent ckt (b) Logic symbol

When the enable input is LOW, Q_2 is OFF and the output circuit operates as a normal totem-pole configuration in which the output state depends on the input state.

When the enable input is HIGH, Q_2 is ON. There is thus low on the ground emitter of Q_1 causing Q_3 and Q_5 to turn OFF and diode D_1 is forward biased, causing Q_4 also to turn OFF. When both totem-pole transistors are OFF, they are effectively open and the output is completely disconnected from the internal circuitry.

10.5 74XX Series CMOS Gates

The **characteristic of CMOS gates include:**

i) **Supply voltage:** The 400 series and 74C series operate with 3V to 15V power supply and 74HC, 74 HCT series operate with 2V to 6V power supply.

ii) **Temperature range:** 74C series operates properly in ambient temperature ranging from –40 C to +85 C and this is adequate for most of the commercial applications. But 54C series can be operated in temperature ranging from –55 C to +125 C.

iii) **Power dissipation:** CMOS logic circuit dissipates 2.5nW power per gate (extremely low power). This is the reason why CMOS is used so widely.

iv) **Propagation delay:** A standard CMOS gate has a propagation delay time approximately 25 ns to 100 ns depending on the operating voltage and other factors.

v) **Fan-out:** For standard CMOS, fan-out is 50.

vi) **Static charge susceptibility:** It is more susceptible to static discharge, so it gets destroyed easily by excessive gate voltage.

1. CMOS Inverter

When HIGH is applied to the input A, the PMOS Q_1 is OFF and the NMOS Q_2 is ON. This results the output to be pulled down to ground voltage causing the output Y to be LOW. When LOW is applied to the input, transistor Q_1 is ON and Q_2 is OFF. This connects the output to $+V_{DD}$ (i.e., +5V) resulting HIGH output.

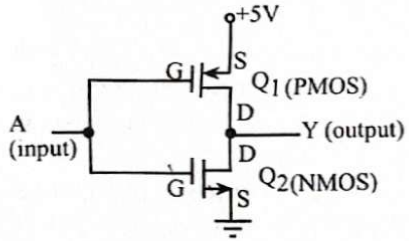


Fig.: CMOS inverter

Input			Output
A	Q_1	Q_2	Y
0	ON	OFF	1
1	OFF	ON	0

2. CMOS AND Gate

Input						Output
A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

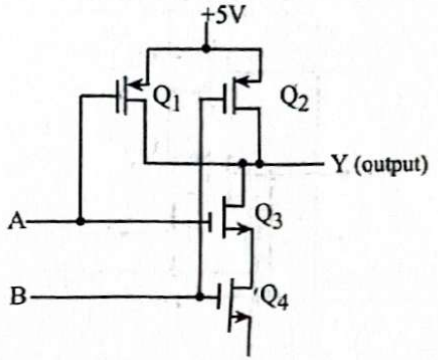


Fig.: CMOS NAND gate

When both inputs are LOW, Q_1 & Q_2 are ON, and Q_3 & Q_4 are OFF. The output is pulled HIGH through the ON resistance of Q_1 & Q_2 in parallel.

When A is LOW, B is HIGH, Q_1 & Q_4 are ON, and Q_2 & Q_3 are OFF. The output is pulled HIGH through the on resistance of Q_1.

When A is HIGH, B is LOW, Q_1 & Q_4 are OFF, and Q_2 & Q_3 are ON. The output is pulled HIGH through the low on resistance of Q_2.

When both inputs are HIGH, Q_1 & Q_2 are cut off, and Q_3 & Q_4 are ON. In this case, the output is pulled low through the on resistance of Q_3 & Q_4 in series to ground.

3. CMOS NOR Gate

Input						Output
A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	0
1	1	OFF	OFF	ON	ON	0

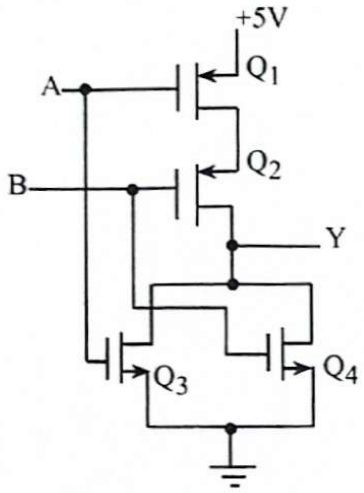


Fig.: CMOS NOR gate

When both inputs are LOW, Q_1 & Q_2 are ON, and Q_3 & Q_4 are OFF. As a result, the output is HIGH through Q_1 and Q_2 in series.

When A is LOW, B is HIGH, Q_1 & Q_4 are ON, and Q_2 & Q_3 are OFF. The output is LOW through Q_4 to ground.

When A is HIGH, B is LOW, Q_1 & Q_4 are OFF, and Q_2 & Q_3 are ON. The output is LOW through Q_3 to ground.

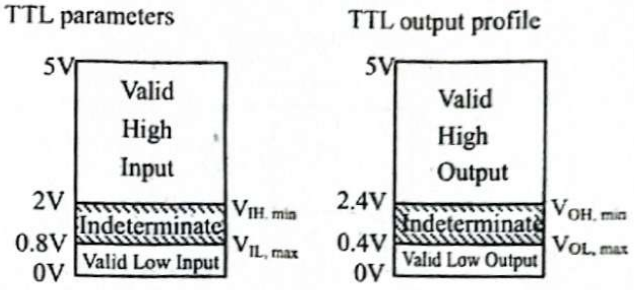
When both inputs are HIGH, Q_1 & Q_2 are OFF, and Q_3 & Q_4 are ON. The output is LOW through Q_3 & Q_4 in parallel.

4. CMOS AND and OR gates

These can be formed by just inserting the inverter to their outputs.

10.6 TTL and CMOS Parameters

1) TTL Parameters



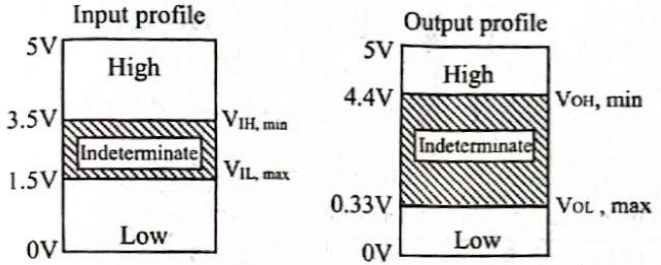
$V_{IL, max} = V_{in}$ maximum which is low = 0.8V

$V_{IH, min} = 2V$

$V_{OL, max} = 0.4$ V

$V_{OH, min} = 2.4$ V

2) CMOS Parameters



10.7 Interface between TTL and CMOS

The different voltage level requirements of TTL and CMOS technology presents problem when these two types of gates exist in the same system. Although CMOS and TTL gates are operating on the same 5V power supply, the output voltage level of TTL is not compatible to input voltage level of CMOS and vice-versa.

Example:

For instance, if the TTL gate outputs a high signal with 3V (outputs HIGH between 2.4V and 5V) is fed to CMOS input, it might not be properly interpreted by CMOS gates input as HIGH (HIGH is between 3.5V to 5V).

This results the high output of TTL to uncertain region of CMOS output and may be interpreted as LOW

1. TTL to CMOS Interface

The pull-up resistor raises the output level of TTL gate to about +5 V, in the high state

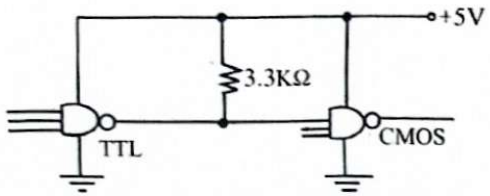


Fig.: TTL drive and CMOS load

2. CMOS to TTL Interface

Buffer increases the current sourcing capacity of CMOS gate.

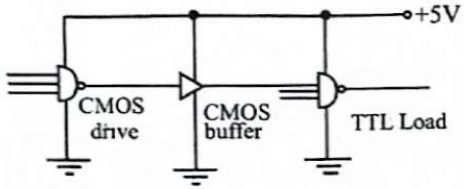


Fig.: CMOS buffer can drive standard TTL load.

10.8 Difference between TTL and CMOS

The differences between TTL and CMOS are tabulated below:

	TTL		CMOS
1)	TTL circuits utilize BJTs (Bipolar Junction Transistors).	1)	CMOS circuits utilize FETs (Field Effect Transistors).
2)	Less density of logic function can be fabricated in a single chip.	2)	It allows much higher density of logic function in a single chip compared to TTL.
3)	TTL chips are less susceptible to static discharge compared to CMOS.	3)	More susceptible to static discharge compared to TTL.
4)	Consumes more power compared to CMOS.	4)	Consumes less power.

MORE WORKED OUT EXAMPLES:

1. **What do you mean by totem-pole output?**

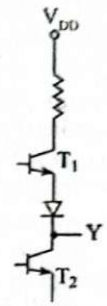


Figure: Tottem-pole output

The totem pole output means the transistor T_1 sits a top of T_2 so as to gsive low output impedence. The low output impedance implies a short time constant RC so that the output can change quickly from one state to another.

2. **Explain the use of BJT as a switch.**

⇒ To explain the switching characteristics of BJT let us take an npn transistor of common emitter circuit.

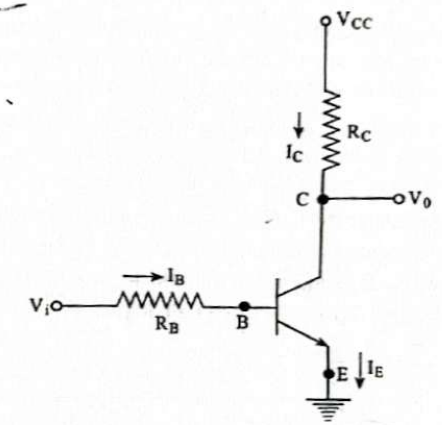


Fig.: Common emitter BJT

The current marked I_C flows through resister R_C and the collector of the transistor current I_B flows through resistor R_B and the base of the transistor. The emitter is connected to ground and $I_E = I_B + I_C$. The supply voltage is between V_{CC} and ground. The input is between V_i and ground and the output is between V_0 and ground.

Case I

If V_{BE} is than 0.6V ≈ 0V, the transistor is cut off with $I_B = 0$. The collector to emitters circuit when behaves like an open circuit with $I_C = 0$ and drop R_C is 0 and $V_0 = V_{CC}$.

Case II

If V_{BE} is from 0.8V to $V_{CC} \approx V_{CC}$, the transistor is biased with collector to emitter voltage $V_{CE} = 0$ this leads $I_C = V_{CC}/R_C$ So, $V_0 = 0V$

3. **What do you understand by the term 'common bus'? Explain the construction of common bus using Tristate buffer with a clear diagram. What issues should be considered when designing a bus with tristate buffer?**

⇒ In normal logic circuits, there are two states of output, either 'LOW' or 'HIGH'. But in complex systems like microcomputers and microprocessors, a number of gate outputs may be required to be connected to common line which is referred to as a "bus" which, inturn, may be required to drive a number of gate inputs. When a number of gate output are connected to the bus, some difficulties are encountered these are.

i Totem-pole outputs cannot be connected together due to very large current drain from the supply and consequent heating of IC's which may get damaged.

ii Open collector outputs can be connected together with a common collector resistor connected externally. This causes the problems of loading and operation speed.

To overcome these difficulties, special circuit have been developed in which there is one more output, referred to as the third state or high impedance (Hi–Z) state. In addition to LOW and HIGH states, such circuit are called Tristate or three state logic.

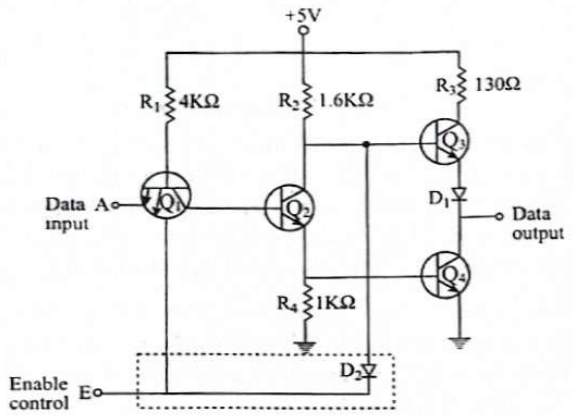


Fig.: Tristate inverter (circuit diagram)

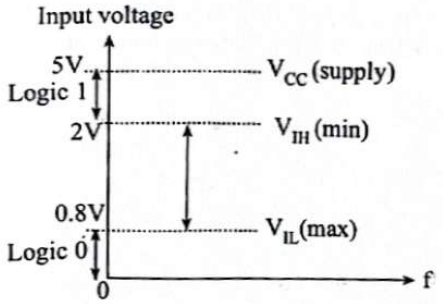
The tristate operation is achieved by modifying the basic totem–pole circuit. Figure above shows the circuit for a Tristate inverter where the portion enclosed in a dotted rectangle has been added to a basic circuit. The circuit has two inputs: A is the normal logic input and E is enable input capable of producing Hi–Z state. When E = 0 the circuit goes into Hi–Z state regardless of the state of logic input A. The LOW E forward biases the emitter –base junction of Q_1 and shunt the resistor R_1 current away from Q_2 and that Q_2 turns off, which turns Q_2 off. The LOW at E also forward biases D_2 to shunt current away from base of Q_3 so that Q_3 also turns off with both totem–pole transistors in the off state, the output terminal is essentially open circuit.

When E = 1, the circuit operates at normal inverter because HIGH input at E has no effect on transistor Q_1 and D_2. In this enabled condition output is simply inverse of logic inputs.

4. **Discuss the following TTL parameters:**

i. Worst-case input voltages

⇒ Worst case input voltage:



Ideally logic 0 is the input voltage level of 0V and logic 1 is the input voltage level of +5V. But, practically the exact matching of voltage level is not obtained. Thus, the maximum input voltage which is considered as logic 0 is V_{IL}(max) and the minimum input voltage which is considered as logic 1 is V_{IH}(min) as shown in graph. These are the worst case input voltage levels.

Chapter – 11

APPLICATIONS

In this chapter, we will try to tie together many of the fundamental ideas presented previously by considering some of the more common digital circuit design encountered in industry.

11.1 Multiplexing Displays

The decimal outputs of digital instruments such as digital voltmeters and frequency counters are often displayed using seven-segment indicators. Such indicators are constructed by using a fluorescent bar, a liquid crystal bar, or a LED bar for each segment. LED-type indicators are convenient because they are directly compatible with TTL circuits, do not require the higher voltages used with fluorescents, and are generally brighter than liquid crystals. On the other hand, LEDs do generally require more power than either of the other two types, and *multiplexing* is a technique used to reduce indicator power requirements.

Basically, multiplexing is accomplished by applying current to each display digit in short, repeated pulses rather than continuously. If the pulse repetition rate is sufficiently high, our eye will perceive a steady illumination without any flicker. When DIGIT is high, the transistor (switch) is off, and the indicator current is zero. When DIGIT is low, the transistor is on, and a number is displayed.

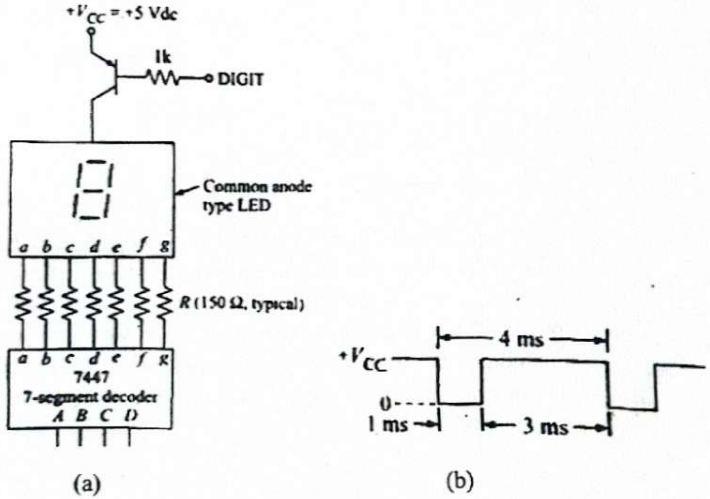


Fig.: (a) Multiplexed display (b) DIGIT waveform

11.2 Frequency Counter

A frequency counter is a digital instrument that can be used to measure the frequency of any periodic waveform. The fundamental concepts involved are illustrated in the block diagram shown below. A GATE ENABLE signal that has a known period t is generated with a clock oscillator and a divider circuit and is applied to leg of an AND gate. The unknown signal is applied to the other leg of the AND gate and output of AND gate acts as the clock for the counter. The counter will advance one count for each transition of the unknown signal, and at the end of the known time period, the contents of the counter will be equal to the number of periods of the unknown signal that have occurred during t. In other words, the counter contents will be proportional to the frequency of unknown signal.

For instance, suppose that the gate signal is exactly 1s and unknown input signal is a 750-Hz square wave. At the end of 1 s, the counter will have counted up to 750, which is exactly the frequency of the input signal.

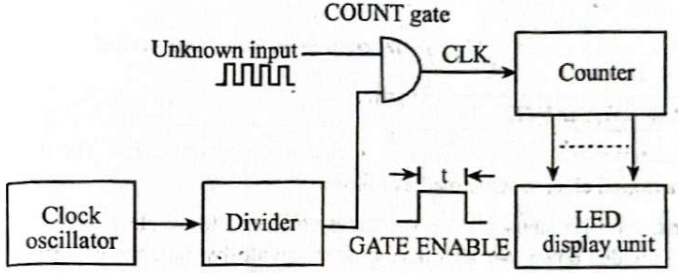


Fig.: Basic frequency counter

11.3 Time Measurement

With only slight modifications, the frequency counter we have studied just now can be converted into an instrument for measuring time. The logic block diagram shown below illustrates the fundamental ideas used to construct an instrument that can be used to measure the period of any periodic waveform. The unknown voltage is passed through a conditioning amplifier to produce a periodic waveform that is compatible with TTL circuits and is then applied to a JK flip-flop. The output of this flip-flop is used as the ENABLE-gate signal, since it is high for a time t that is exactly equal to the time period of the unknown input voltage. The oscillator and divider provides a series of pulses that are passed through the count gate and serve as the clock for the counter. The contents of the counter and display unit will then be proportional to the time period of the unknown input signal.

For instance, if the unknown input signal is a 5-kHz sine wave and the clock pulses from the divider are 0.1 μs in width and are spaced every 1.0 μs, the counter and display will read 200. Clearly, this means 200 μs, since 200 of these 0.1-μs pulses will pass through the COUNT gate during the 200 μs that ENABLE-gate signal is high. Naturally, the counter and the display have an accuracy of plus or minus one count.

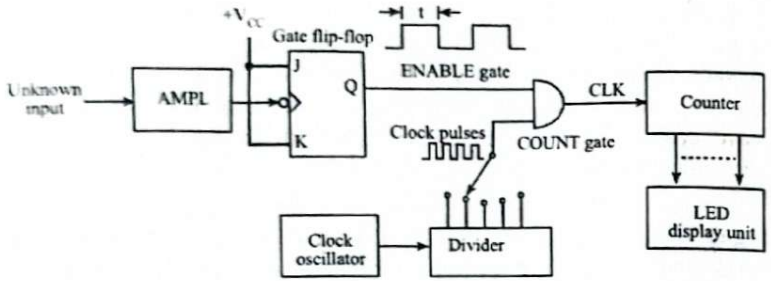


Fig.: Instrument to measure time period

11.4 Digital Clock

A common example of a counter application is digital clock. The operation of a digital clock is explained as follows.

First, a 60 Hz sinusoidal ac voltage is converted to 60 Hz pulse waveform and divided down to 1 Hz pulse wave by divide–by–60 counter formed by a divide–by–10 counter followed by divide–by–6 counter. Then second divide–by–60 counter produces value for 'seconds' which counts from 00 to 59 and then recycle to 00 at each second. The third divide–by–60 counter changes state once each minute and counts from 00 to 59 minutes. The last counter changes state once on each 60 minutes (once each hour). Thus it is a divide–by–12 counter, it will count from 1 to 12 states that can be decoded to provide signals to display the correct hour. The values of counter are displayed in seven segment display unit using BCD to seven segment decoder.

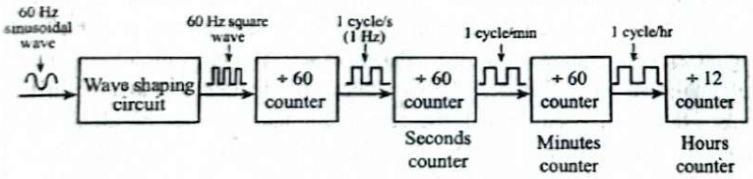


Fig.: Simplified block diagram of diagram of a digital clock

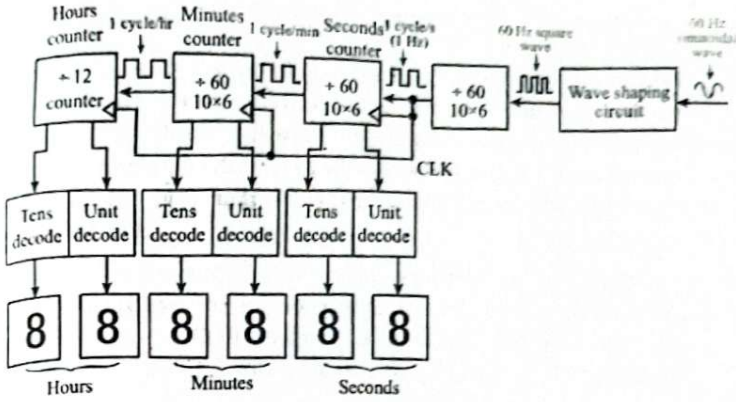
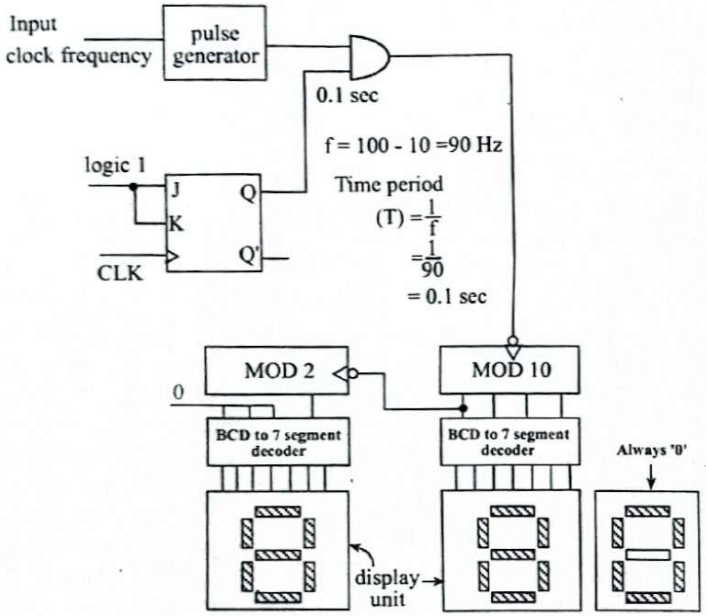


Fig.: Detailed diagram of a digital clock

MORE WORKED OUT EXAMPLES:

1. **Describe how will you construct a frequency counter (assume square wave of 5V) of a wide range, 10 Hz to 100 Hz. You need not provide the circuit details, but must specifically mention what type of arrangement you will make to read the data and display it appropriately. Also, discuss any error that your circuit might possess. [2065 Shrawan]**

⇒ For designing the frequency counter, which counts the frequencies from 10 Hz to 100 Hz, several components are used such as pulse generator, different synchronous counters, BCD to seven segment display and flip-flops. The basic layout is drawn below:



In circuit above the input clock frequency is generated to a clock pulses by pulse generator and ANDed with 0.1 sec clock pulses. Three seven segment displays are taken and rightmost display is taken to always '0' and respective onward displays are driven by MOD 10 and MOD 2 synchronous counters.

Limitation: This circuit only counts the signals which are only a multiple of 10. The other frequency signals, this circuit can't works.

2. **A digital system requires 3 Hz clock, but the clock coming to this digital system is 6 Hz. How do you solve such problem by using sequential logic? Explain with necessary diagram. [2064 Jestha]**

⇒

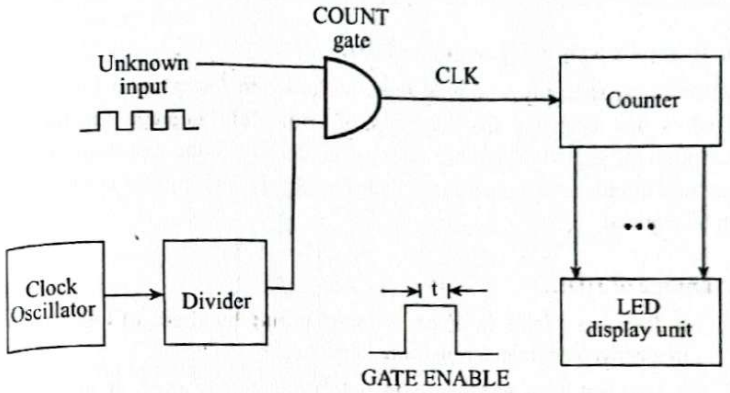


Figure above shows the digital system that counts the frequency which of 3 Hz clock signals as the input to a counter. Initially applied 6 Hz clock is fed to clock oscillator and according to system requirement this clock signal is divided into half by divider circuit. Now the 3 Hz clock frequency signal of periodic nature is applied to the counting purpose.

APPENDIX

HARDWARE DESCRIPTION LANGUAGE

Hardware Description Language(HDL)

In electronics, specially in digital logic a Hardware Description Language (HDL) is any language from a class of computer language for formal description of digital electronic circuits. It describes the behavior of an electronic circuit or system from which the physical circuit or system can then be attained.

Advantage of HDL:

- It describes a large complex design requiring hundreds of logic gates in a convenient manner, in a smaller space.
- It uses software test-bench to detect functional error, if any, and correct it (called simulation).
- Finally, get hardware implementation details (called synthesis).

HDL types:

Currently there are two widely used HDLs.

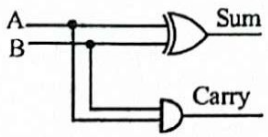
1. Verilog HDL
2. VHDL (Very high speed integrated circuit HDL)

Verilog is the first introduced in 1985 and is considered to be simpler than VHDL and more popular. However both of this two share lot of common features and it is not difficult to switch from one to the other. Here, we will be discussing about Verilog only.

Examples using Gate Modeling or Structural Model:

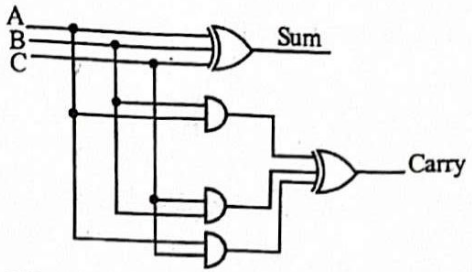
1) Half adder

```
module half_adder (A,B, sum, carry);
```




```
input A, B;
output sum, carry;
xor sum_1(sum, A, B);        // sum = A xor B
and carry_1(carry, A, B);    // carry = A and B .
endmodule
```

2) Full adder




```
module full_adder(A, B, C, sum, carry);
input A, B, C;
output sum, carry;
wire and1, and2, and3;        // wire shows gate level inter connection
  and U_and1(and1, A, B);
  and U_and2(and2, B, C);
  and U_and3(and3, A, C);
  or U_or(carry, and1, and2, and3);    // carry
  xor U_sum(sum, A, B, C);             //sum
endmodule
```

3) Full subtractor

```
module full_subtractor (A, B, C, difference, borrow);
input A, B, C;
output difference, borrow;
wire inv_A, and1, and2, and3;
      not i1 (inv_A, A);
      and U_and1(and1, inv_A, B);
      and U_and2(and2, inv_A, C);
      and U_and3(and3, B, C);
      xor U_difference (difference, A, B, C);
      or U_borrow (borrow, and1, and2, and3);
endmodule
```

4) **2:1 MUX**

```
module 2_1MUX (D0, D1, sel, y);
input D0, D1, sel,
output y;
wire inv_sel, and1, and2;
        not i1(inv_sel, sel);
        and gate1(and1, D0, inv_sel);
        and gate2(and2, D1, sel);
endmodule
```

5) **2 to 4 decoder**

```
module 2_4decoder (A, B, D0, D1, D2, D3);
input A, B;
output D0, D1, D2, D3;
wire x, y;
  not i1 (x, A);
  not i2 (y, B);
  and and1(D0, x, y);
  and and2(D1, x, B);
  and and3(D2, A, y);
  and and4(D3, A, B);
endmodule
```

6) **4 to 2 encoder**

```
module encoder4_2(D0, D1, D2, D3, y);
input D0, D1, D2, D3;
output [1,0]y;        //y in array form i.e. two output y[0] and y[1]
  or output1(y[0], D1, D3);    // y[0] = D1 +D3
  or output2(y[1], D2, D3);    // y[1] = D2 + D3
endmodule
```

Examples using Data Flow Model:

1) **2 to 1 MUX**

```
module 2_1MUX(y, D0, D1, S0);
input D0, D1, S0;
output y;
  assign y = (~ S0& D0) | (D1& S0);
endmodule
```

OR

```
module 2_1MUX (y, D0, D1);
input D0, D1, S0;
```



```
output y;
  assign y = S0? D1:D0   // if S0 = 1, y = D1; if S0 = 0, y = D0
endmodule
```

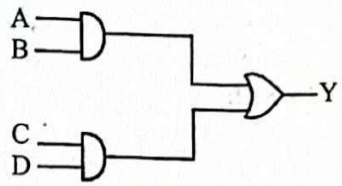
2) **4 to 1 MUX**

```
module 4_1MUX(A, B, D0, D1, D2, D3, y);
input A, B, D0, D1, D2, D3;
output y;
  assign y= A? ((B? D3:D2): (B? D1:D0));
endmodule
```

/* nested condition for assign, if A =1, condition (B? D3:D2) is evaluated, then if B = 1, Y = D3 and B = 0, Y = D2. Similarly, other combination of A and B evaluated and Y is assigned from D2 to D0. */

Examples using Behavioral Model:

1) **AND – OR gate**




```
module figure (A, B, C, D, Y);
input A, B, C, D;
output Y;
reg Y;
always @ (A or B or C or D)
if ((A = = 1) && (B = = 1))
  Y = 1;
else if ((C = = 1) && (D = = 1))
  Y = 1;
else
  Y = 0;
endmodule
```

2) **4 to 1 MUX**

```
module 4_1MUX (D0, D1, D2, D3, sel, y);
input D0, D1, D2, D3;
input [1,0]sel;    //select as array sel[0] and sel[1]
output y;
reg y;
```

```
always @ (D0 or D1 or D2 or D3 or sel)
case (sel)
0: y = D0;   // sel [0] = 0 and sel [1] = 0
1: y = D1;   // sel [0] = 0 and sel [1] = 1
2: y = D2;   // sel [0] = 1 and sel [1] = 0
3: y = D3;   // sel [0] = 1 and sel [1] = 1
endcase
endmodule
```

MORE WORKED OUT EXAMPLES:

1. Implement 1:4 DEMUX using VHDL **[2069 Ashadh]**

⇒ *HDL for 1 to 4 DEMUX*

```
module 1to4DMUX (A, S0, S1, D0, D1, D2, D3);
input A, S0, S1;
output D0, D1, D2, D3;
  assign D0 = A & ~S0 & ~S1;
  assign D1 = A & ~ S0 & S1;
  assign D2 = A & S0 & ~S1;
  assign D3 = A & S0 & S1;
endmodule
```

2. Design full adder circuit using HDL. **[2069 Ashadh]**

⇒ *HDL for full adder*

```
module fulladder (A, B, C Sm, Cr);
input A, B, C;
output Sm, Cr;
  assign Sm = A^B^C;
  assign Cr = (A & B) | (B & C) | (C & A);
endmodule
```

3. Design 2 to 4 line decoder circuit using HDL. **[2068 Baishakh]**

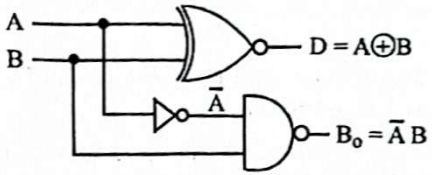
⇒ *HDL for 2 to 4 decoder*

```
module 2to4decoder (A,B,D0,D1,D2,D3);
input A, B;
output D0, D1, D2, D3;
  assign D0 = ~A & ~B;
  assign D1 = ~A&B;
  assign D2 = A&~B;
```



```
  assign D3 = A&B;
endmodule
```

4. Design half-subtractor using HDL. **[2075 Ashwin]**




```
Module half–subtractor (A, B, D, Bo);
input A, B;
output D, Bo;
wire inv_A;
x or difference_1 (D, A, B);
not not_A (inv_A, A);
and borrow_1 (Bo, inv_A, B);
endmodule
```