DELD

Unit-3

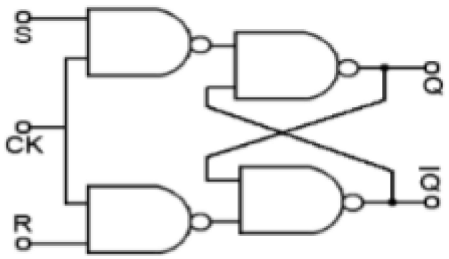
Sequential Logic Design

3.1 Flip-Flop: SR, JK, D, T

- RS Flip Flop
- JK Flip Flop
- D Flip Flop
- T Flip Flop

Logic diagrams and truth tables of the various types of flip-flops are as follows:

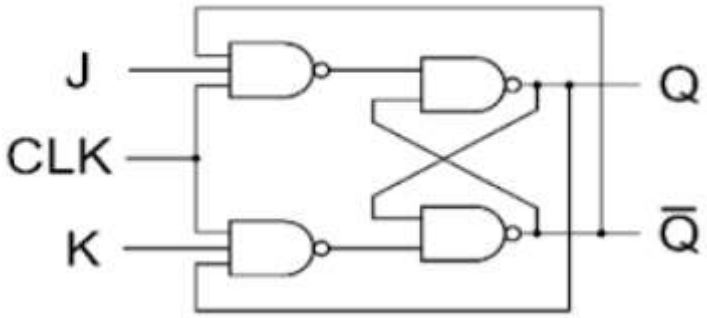
- **S-R Flip Flop :**



TRUTH TABLE

S	R	Q_N	Q_{N+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	-

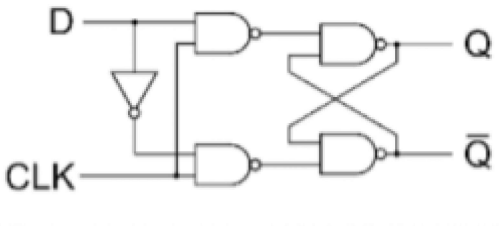
2. J-K Flip Flop:



TRUTH TABLE

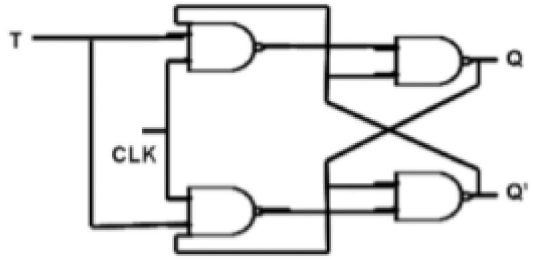
J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

3. D Flip Flop :



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

4. T Flip Flop :



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

3.2 Preset &Clear

As flip flop may come up in random states when power is switched on, it is necessary to use some method where all the flip flops can be set or reset at the same time by a single switch – it is inconvenient to set or reset each flip flop

individually. This is achieved by incorporating PRESET and CLEAR inputs in flip flops which can over-write all other inputs. These I/Ps are **asynchronous** and independent of the clock. While they are present, all other operations are inhibited. They force the output to a certain state irrespective of the I/Ps and clock.

Active High

Active High means that Preset and Clear I/Ps are inactive when low.

Pr = 1, Q = 1, Q' = 0

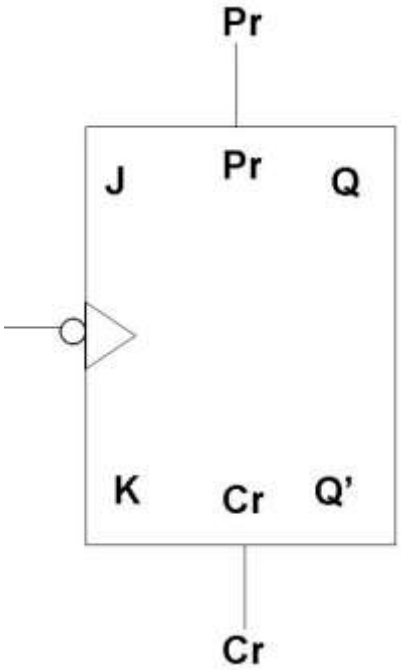
i.e. if the Preset I/P is 1, the output will be forced to **SET** state irrespective of other I/Ps.

Cr = 1, Q = 0, Q' = 1

i.e. if the Clear I/P is 1, the output will be forced to **RESET** state irrespective of other I/Ps.

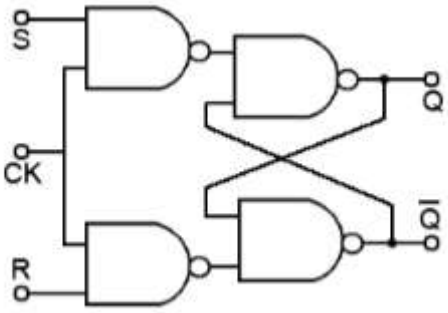
It is not possible to Preset and Clear a flip flop at the same instant of time as Q can't be 0 and 1 at the same time.

i.e. Pr = Cr ≠ 1 at any instant.



3.3 Master-Slave JK Flip Flops

- It is a combination of two JK flip-flops connected in series.
- The first is the **“master”** and the other is a **“slave”**.
- The output from the master is connected to the two inputs of the slave whose output is fed back to inputs of the master.
- In addition to these two flip-flops, the circuit comprises an **inverter**.
- The inverter is connected to clock pulse in such a way that an inverted clock pulse is given to the slave flip-flop.
- In other words, if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and vice versa.



TRUTH TABLE

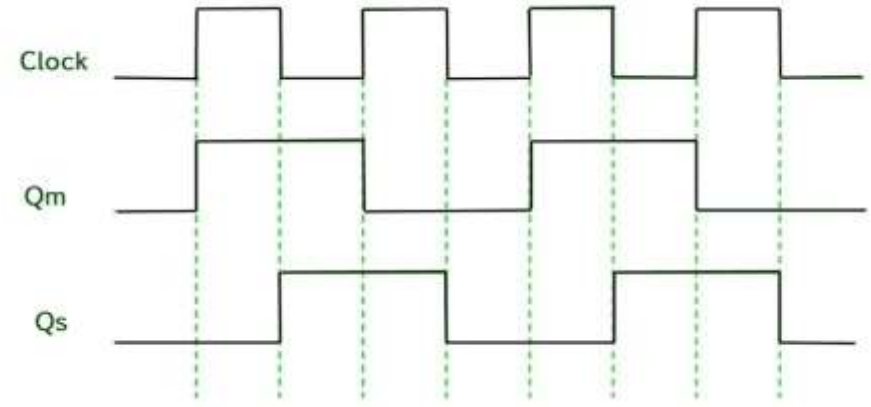
S	R	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

Fig. Master-Slave Flip flop

Working of a master-slave flip flop –

- When the clock pulse goes high, the slave is isolated; J and K inputs can affect the state of the system. The slave flip-flop is isolated when the CP goes low. When the CP goes back to 0, information is transmitted from the master flip-flop to the slave flip-flop, and output is obtained.
- The master flip flop is a positive level triggered and the slave flip flop is negative level triggered, hence the master responds before the slave.
- If J=0 and K=1, Q' = 1 then the master goes to the K input of the slave and the clock forces the slave to reset therefore the slave copies the master.
- If J=1 and K=0, Q = 1 then the master goes to the J input of the slave and the Negative transition of the clock sets the slave and thus copy the master.
- If J=1 and K=1, the master toggles on the positive transition, and the slave toggles on the negative transition of the clock.
- If J=0 and K=0, the flip flop becomes disabled and Q remains unchanged.

Timing Diagram of a Master flip flop –



- When the CP = 1 then the output of master is high and remains high till CP = 0 because the state is stored.
- Now the output of the master becomes low when the clock CP = 1 and remains low until the clock becomes high again.
- Thus toggling takes place for a clock cycle.
- When the CP = 1 then the master is operational but not the slave.
- When the clock is low, the slave becomes operational and remains high until the clock again becomes low.
- Toggling takes place during the whole process since the output changes once in a cycle.
- This makes the Master-Slave J-K flip flop a Synchronous device that passes data with the clock signal.

3.4 Truth Tables and Excitation tables

3.5 Conversion from one type to another type of Flop Flop

Conversion for FlipFlops:-

EXCITATION TABLE:

Q_N	Q_{N+1}	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

i) SR To JK FlipFlop

J	K	Q_N	Q_{N+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Excitation Functions:

$S = JQ_N'$

KQ_n

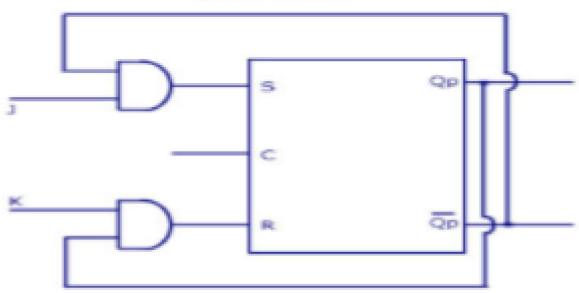
J

0	X	0	0
1	X	0	1

$R = KQ_N$

KQ_n

X	0	1	X
0	0	1	0



ii) Convert SR To D FlipFlop:

D	Q_N	Q_{N+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Excitation Functions:

S = D

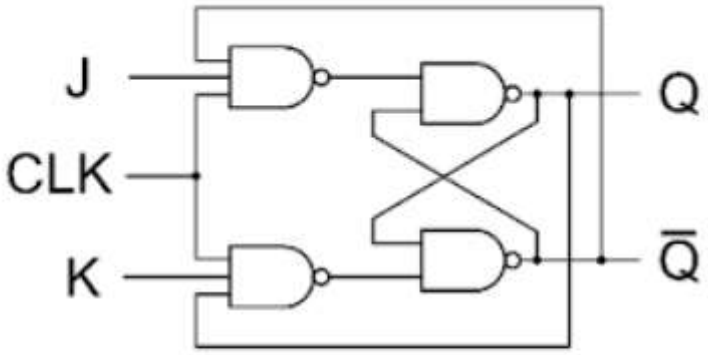
R = D'

S:

D,Q_N		
	0	0
	1	X

R:

D,Q_N		
	X	1
	0	0



TRUTH TABLE

J	K	Q_N	Q_{N+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

3.6 Registers: SISO, SIPO, PISO, PIPO, Shift Registers, Bidirectional Shift Register

- Flip flops are used to store one bit of binary data (1or 0).
- If we need to store multiple bits of data, we use multiple flip flops.
- N flip flops are connected to store n bits of data.
- A **Register** is a device that stores such information. It is a group of flip flops connected in series which is used to store multiple bits of data.
- The information stored in these registers can be transferred with the help of **shift registers**.
- This register is a group of flip flops used to store multiple bits of data.
- The bits stored in these registers can be moved in/out of the registers by applying clock pulses.
- The registers which shift the bits towards left are called “Shift left registers”.

The registers which shift the bits towards the right are called “Shift right registers”.

Shift registers are of 4 types and they are:

- Serial In Serial Out register

- Serial In the parallel Out register
- Parallel In Serial Out register
- Parallel In the parallel Out register

Serial-In Serial-Out Shift Register (SISO) –

- It allows serial input i.e. one bit after another and produces a serial output is known as Serial-In Serial-Out shift register.
- Since it has one output, the data leaves the register one bit at a time in a serial pattern, hence known as Serial-In Serial-Out Shift Register.
- The logic circuit is given underneath.
- The circuit comprises four D flip-flops which are connected serially.
- All these flip-flops are synchronous.

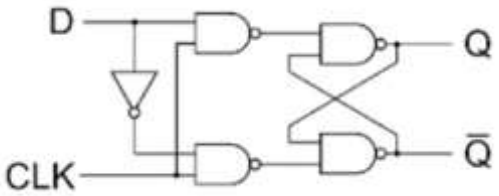


Fig. SISO

Serial-In Parallel-Out shift Register (SIPO) –

- It allows serial input through a single data line and produces a parallel output.
- The logic circuit is given underneath.
- The circuit consists of four D flip-flops which are connected synchronously.
- The clear (CLR) signal is also connected to all the 4 flip flops to RESET them.
- The output of the first flip flop is sent to the input of the next and so on.

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

Fig. SIPO

- They are used in communication lines because the main use of the SIPO register is to convert serial data into parallel data.

Parallel-In Serial-Out Shift Register (PISO) –

- It allows parallel input data and produces a serial output.
- The logic circuit is given underneath.
- The circuit comprises four D flip-flops which are connected synchronously.
- The clock is connected to all the flip flops but the input data is connected to each flip flop individually through a multiplexer.

- The output of the previous flip flop and parallel data input are connected to the input of the MUX and the output of MUX is connected to the next flip flop.

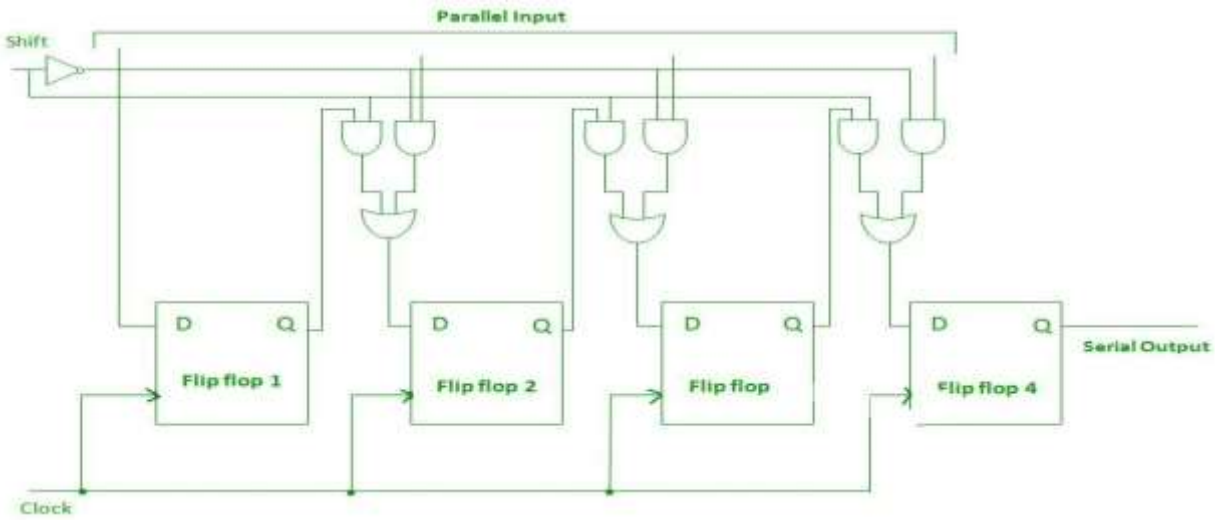


Fig. PISO

- It used to convert parallel data to serial data.

Parallel-In Parallel-Out Shift Register (PIPO) –

- It allows parallel input data and produces a parallel output.
- The logic circuit is given underneath.
- The circuit comprises four D flip-flops which are connected synchronously.
- The clear (CLR) and clock signals are connected to all flip flops.
- In this, there are no interconnections between flip-flops as no serial shifting of the data is required.
- Data is provided separately as input for each flip flop and the output is also collected individually from each flip flop.

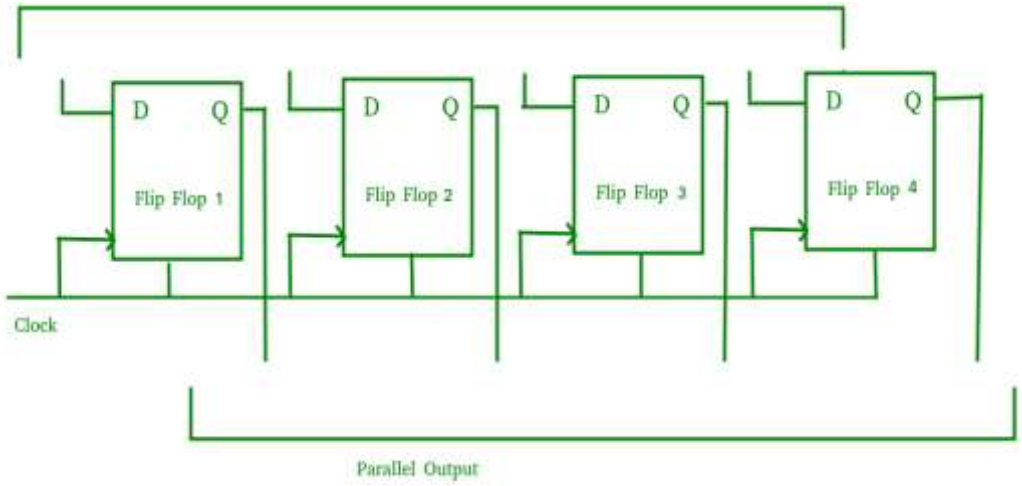


Fig. PIPO

- It is used as a temporary storage device and it acts as a delay element too.

Bidirectional Shift Register –

- If a binary number is shifted towards left by one position then it is equivalent to multiplying the number by 2 and if a binary number towards the right by one position then it is equal to dividing the number by 2.
- To perform the above operations a register is used which can shift the data in either direction.
- Hence, Bidirectional shift registers are capable of shifting the data either right or left depending on the mode selected.
- If the mode =1(high), then the data is shifted towards the right, and if the mode = 0(low), then the data is shifted towards the left direction.
- The logic circuit is given underneath.
- The circuit comprises four D flip-flops which are connected synchronously.
- The input data is connected at two ends of the circuit and depending on the model and gate selected.

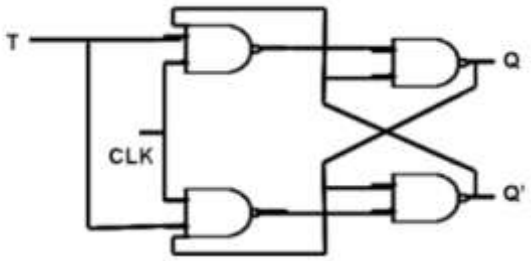


Fig. Bidirectional shift register

3.7 Ring Counter

- It is a shift register counter whose output of the first flip flop is connected to the second and so on and the output of the last flip flop is fed back to the input of the first flip flop, thus named as a ring counter.
- The data pattern in the register circulates as long as clock pulses are applied.
- The circuit below comprises of four D flip-flops which are connected synchronously.
- The data pattern repeats after every four clock pulses shown in the truth table.

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

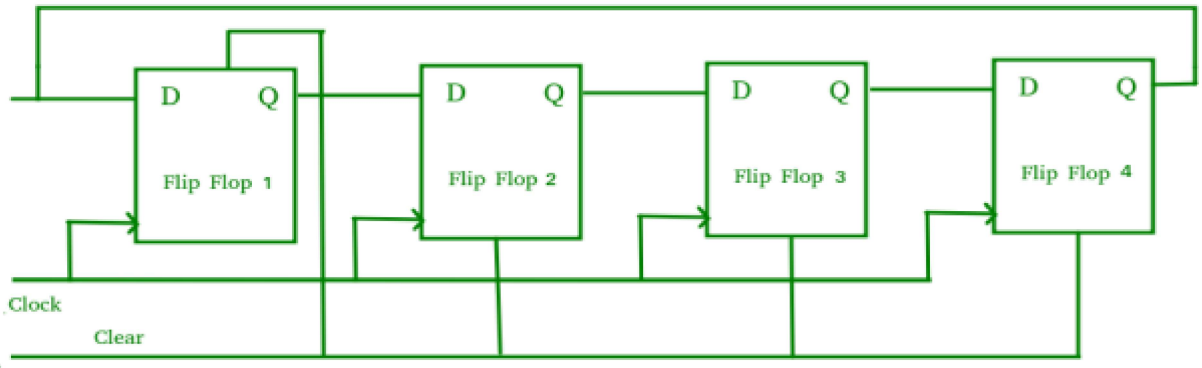


Fig. Ring counter

- It is self-decoding.
- Hence no extra decoding circuit is required to determine which state the counter is in.

3.8 Universal Shift Register Counters: Asynchronous Counter, Synchronous Counter, BCD Counter, Johnson Counter, Modulus of the counter (IC 7490)

Universal Shift Register:

- It is a register that has both the right shift and left shift with parallel load capabilities.
- They are used as memory elements in computers.
- It is capable of shifting in one direction.
- It is capable of shifting in both directions.
- It is a combination of a **bidirectional** and **unidirectional** shift register with parallel load provision.

- **n-bit universal shift register –**

It comprises n flip-flops and n 4×1 multiplexers.

- All the multiplexers share the same select lines (S1 and S0).
- The select inputs select the suitable input for the flip-flops.

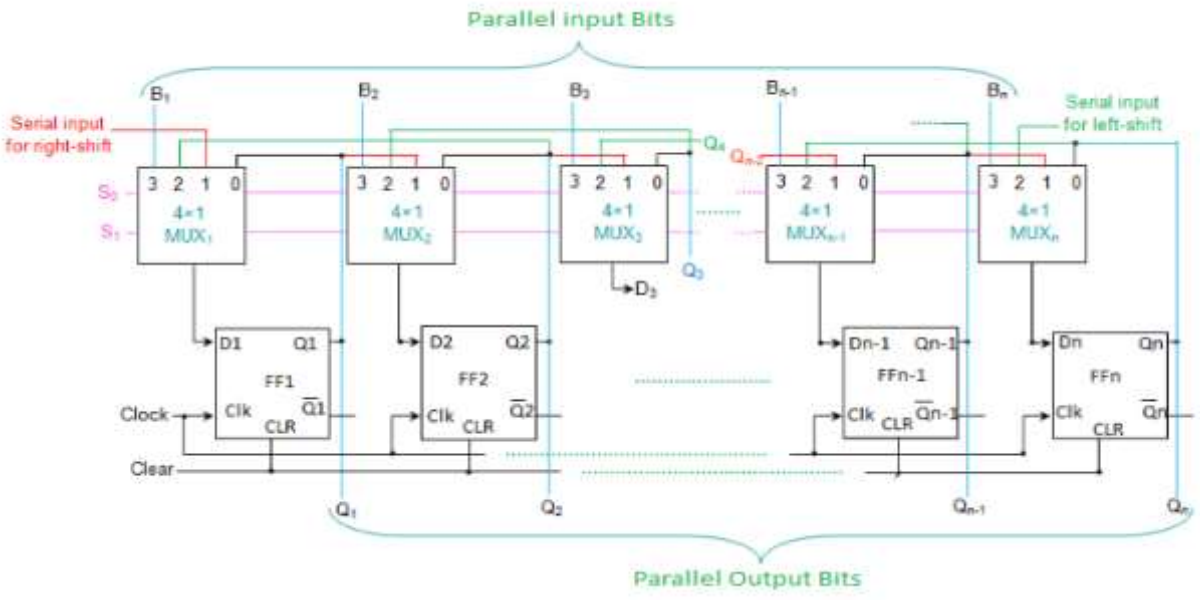


Fig. The N-bit universal shift register

S1	S0	REGISTER OPERATION
0	0	No changes

S1	S0	REGISTER OPERATION
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Asynchronous Counter

- This universal clock is not used and only the first flip flop is driven by the main clock and the clock input of the rest of the following is driven by the output of previous flip flops.

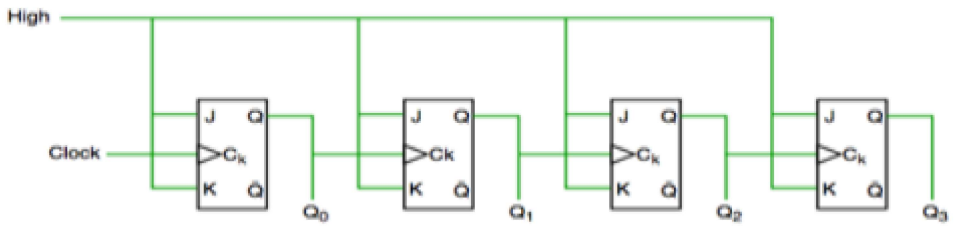


Fig. Asynchronous counter

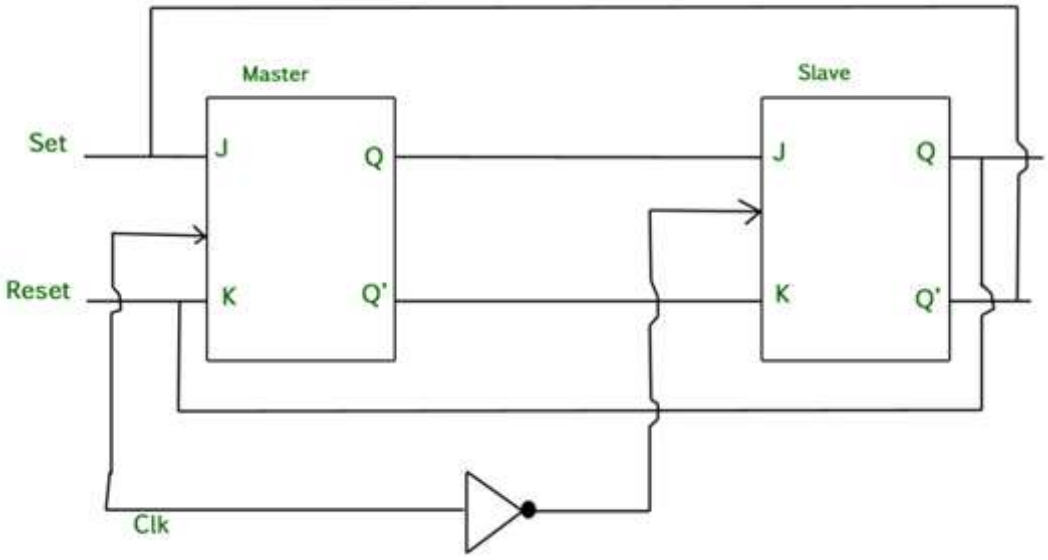


Fig. The timing diagram of Asynchronous counter

- It is seen from the timing diagram that Q0 is changing as soon as the rising edge of the clock pulse is encountered.
- Q1 is changing when the rising edge of Q0 is encountered and so on.
- In this way, ripples are generated through Q0, Q1, Q2, Q3 and therefore it is also called a **RIPPLE counter.**

Synchronous Counter

- It has one global clock which drives each flip flop and hence output changes in parallel.
- The advantage of synchronous counter over the asynchronous counter is that it can operate on a higher frequency and it does not have a cumulative delay.

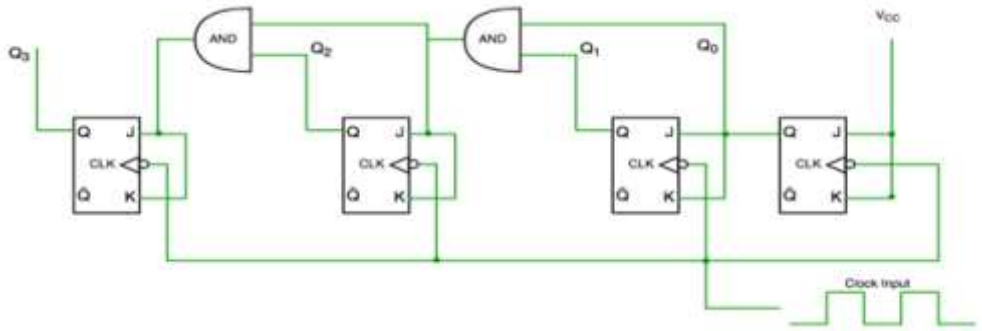


Fig. synchronous counter

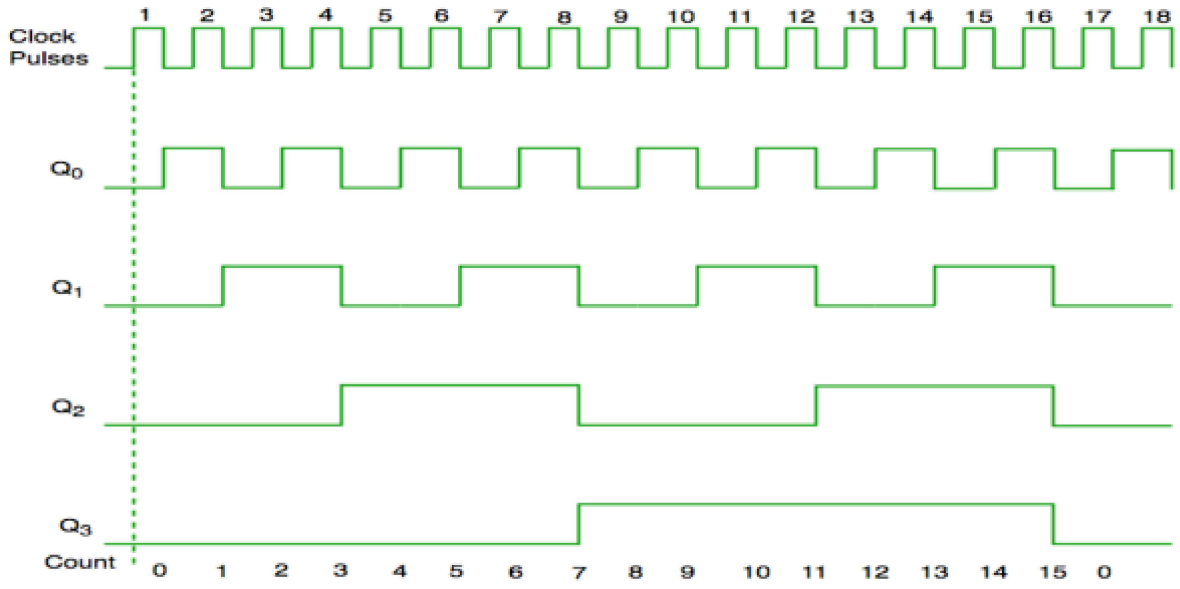
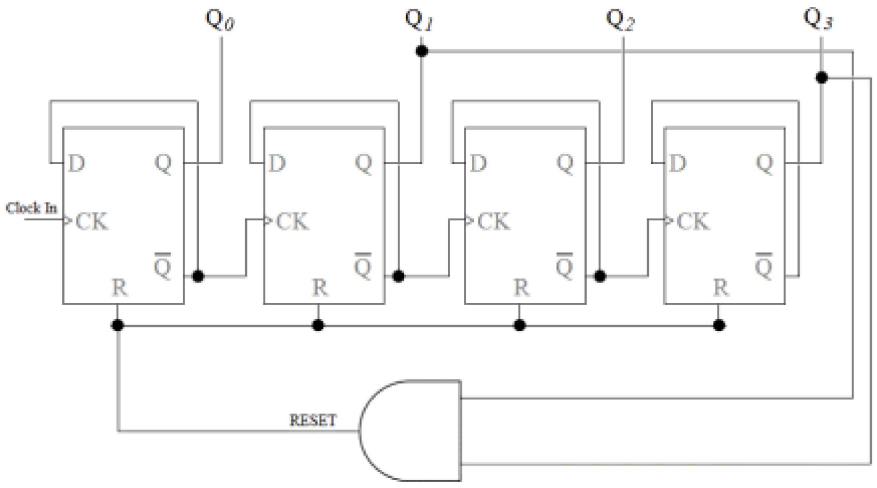


Fig. Timing diagram of the synchronous counter

BCD Counter

This BCD counter uses d-type flip-flops, and this particular design is a 4-bit BCD counter with an AND gate. BCD counters usually count up to ten, also otherwise known as MOD 10. Since a 4-bit counter counts from binary 0 0 0 0 to binary 1 1 1 1, which is up to 16, we need a way to stop the count after ten, and we achieve this using an AND gate to initiate a reset. The 4-bit binary pattern for decimal ten is 1 0 1 0, and this is the only time when we see this pattern, therefore we can use the 1's and feed them into an AND gate, which

should produce a HIGH output, which can then reset all the flip-flops simultaneously, thereby stopping the count.



Understanding MSB and LSB

Decimal	8	4	2	1
Outputs	Q_3	Q_2	Q_1	Q_0
Position	MSB			LSB
Binary	1	0	1	0

The first thing to identify is the most significant bit (MSB), and least significant bit (LSB) concerning the outputs Q_0, Q_1, Q_2, Q_3. If you do not get this part right, then you will trip-up when it comes to connecting the AND gate.

The way I remember it is that this counter always starts with the smallest number, which appears first at Q_0, therefore it has to be the LSB end.

As you can see from the table above, the outputs Q_3 and Q_1 contain the 1's and therefore we connect these outputs to the AND gate.

Johnson Counter –
It is a shift register counter whose output of the first flip flop is connected to the second and so on and then an inverted output of the last flip flop is sent back to the first flip flop.

- They are also called as twisted ring counters.
- The circuit below consists of four D flip-flops which are connected synchronously.
- An n-stage Johnson counter provides a count sequence of 2^n different states, thus also known as a mod-2n counter.
- Since the circuit has four flip flops, the data pattern will repeat every eight clock pulses which are shown in the truth table below:

Q_N	Q_{N+1}	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

J	K	Q_N	Q_{N+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Fig. Johnson counter

- The main advantage is that it only needs n number of flip-flops to circulate a given data to generate a sequence.

Modulus of the counter (IC 7490)

- It counts ten different states and then reset to its initial states.
- A simple decade counter will count from 0 to 9 but the decade counters which can go through any ten states between 0 to 15(for 4 bit counter) can also be made.

Truth table is as follows:

Clock pulse	Q3	Q2	Q1	Q0

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

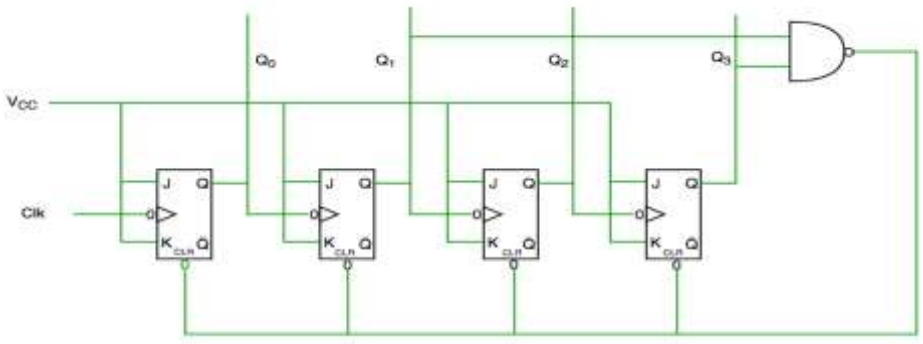


Fig. Decade counter

In the above circuit diagram we used nand gate for Q3 and Q1 and sending this to clear input line as the binary representation of 10 is—

1010

And Q3 and Q1 are 1 here, if we give NAND of these two bits then counter clears at 10 and again starts from the beginning.

3.9 Synchronous Sequential Circuit Design: Models- Moore and Mealy, State diagram and State Table, Design Procedure

Mealy Machine

A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

It can be described by a 6 tuple $(Q, \sum, O, \delta, X, q_0)$ where −

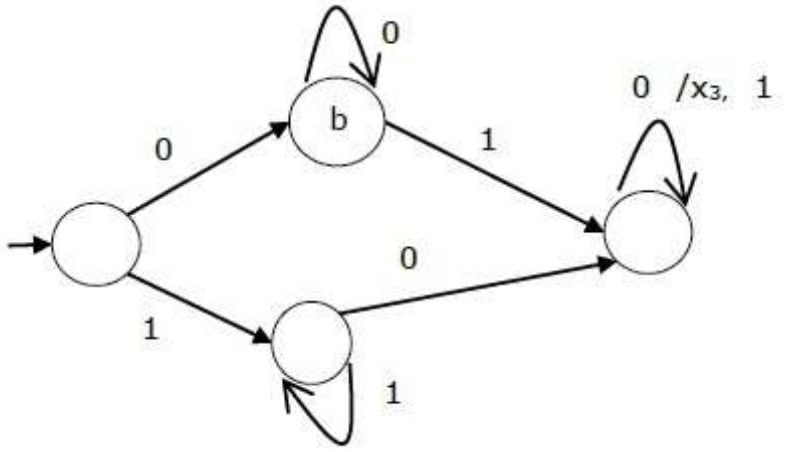
- **Q** is a finite set of states.
- $\sum$ is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where $\delta: Q \times \sum \rightarrow Q$
- **X** is the output transition function where $X: Q \times \sum \rightarrow O$
- $\mathbf{q_0}$ is the initial state from where any input is processed ($q_0 \in Q$).

The state table of a Mealy Machine is shown below −

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1

d	d	x_3	d	x_2

The state diagram of the above Mealy Machine is −



Moore Machine

Moore machine is an FSM whose outputs depend on only the present state.

A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where −

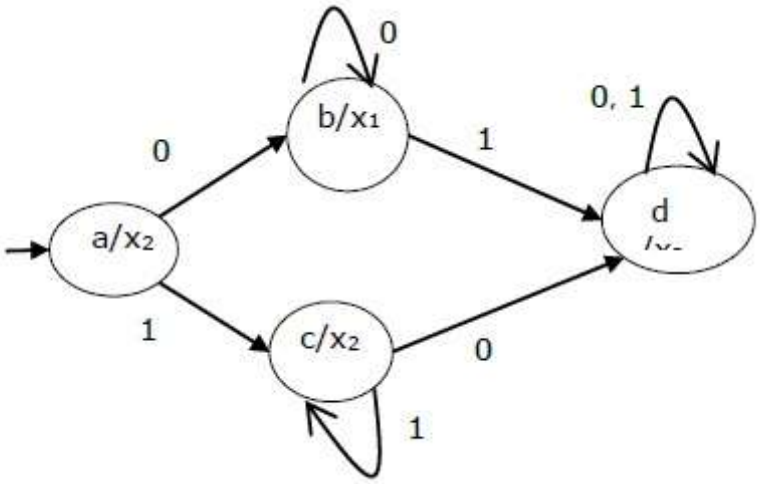
- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where δ: Q ×∑→ Q
- **X** is the output transition function where X: Q → O
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).

The state table of a Moore Machine is shown below −

Present state	**Next State**		**Output**
	Input = 0	**Input = 1**	
→ a	b	c	x_2

b	b	d	x_1
c	c	d	x_2
d	d	d	x_3

The state diagram of the above Moore Machine is −



Mealy Machine vs. Moore Machine

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

Mealy Machine	**Moore Machine**
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.

Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

3.10 Sequence Generator, and a detector

Sequence Generator:

The sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways in which these circuits can be designed including those which are based on multiplexers and flip-flops. Here in this article we deal with the designing of sequence generator using D flip-flops.

As an example, let us consider that we intend to design a circuit which moves through the states 0-1-3-2 before repeating the same pattern. The steps involved during this process are as follows.

- At first, we need to determine the number of flip-flops which would be required to achieve our objective. In our example, there are 4 states which are identical to the states of a 2-bit counter except the order in which they transit. From this, we can guess the requirement of flip-flops to be 2 in order to achieve our objective.
- The state transition table is shown by the first four columns of Table I in which the first two columns indicate the present states while the next two columns indicate the corresponding next states. For instance, first state in our example is 0 = “00” which leads to the next state 1 = “01”.
- Now this state transition table is to be extended so as to include the excitation table of the flip-flop with which we desire to design our circuit. In our case, it is nothing but D flip-flop due to which we have the fifth and the sixth columns of the table representing the excitation table of D flip-flop.

For example, look at the orange shaded row in Table I in which the present and the next states 1 and 0 (respectively) result in D_1 to be 0. The same row also shows the case wherein

$S = JQ_N'$

KQ$_N$

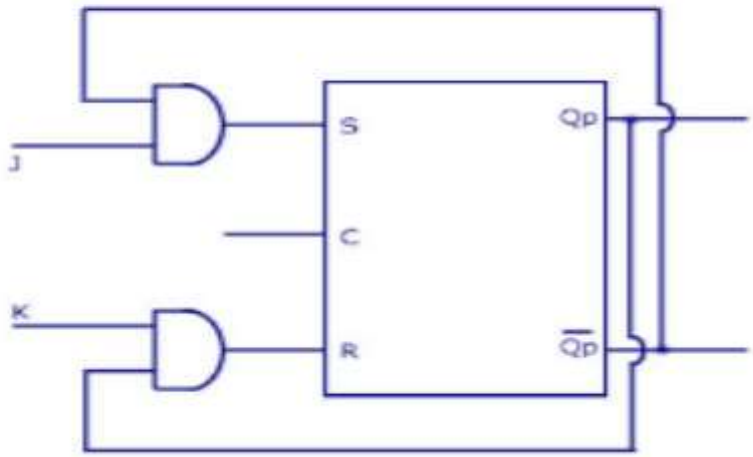
J

0	X	0	0
1	X	0	1

$R = KQ_N$

KQ$_N$

X	0	1	X
0	0	1	0

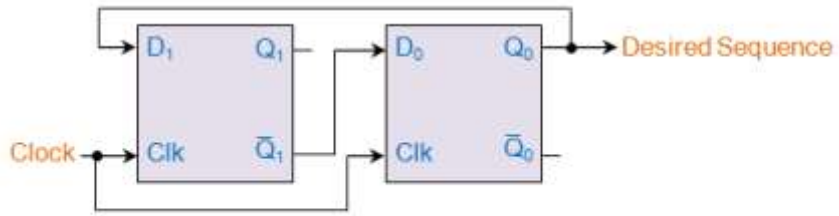


Present States		Next States		Inputs of D flip-flops	
Q_1	Q_0	Q_1^+	Q_0^+	D_1	D_0
0	0	0	1	0	1
0	1	1	1	1	1
1	1	1	0	1	0
1	0	0	0	0	0

Now its time to derive the Boolean expressions for D_1 and D_0. This can be done using any kind of simplification technique including K-map. However as our example is quite simple, we can just use the Boolean laws to solve for D_1 and D_0. Thus

D	Q_N	Q_{N+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Having known the inputs to either of the D flip-flops, now we can design our **sequence generator** as shown in this figure.



Sequence Generator Designed Using D Flip-Flops

Sequence detector

- It is a sequential state machine that takes an input string of bits and generates an output 1 whenever the target sequence has been detected.
- In a Mealy machine, output depends on the present state and the external input (x).
- Hence the output is written outside the states, along with inputs.
- Sequence detector is of two types:
  - Overlapping
  - Non-Overlapping
- In an overlapping sequence detector, the last bit of a sequence becomes the first bit of the next one.
- In a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next one.
- Examples:

  For non-overlapping case

Input :0110101011001

Output:0000100010000

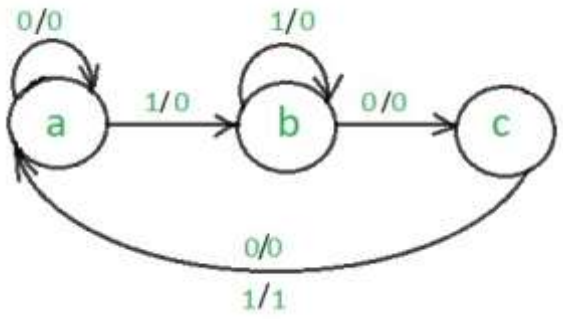
For overlapping case

Input :0110101011001

Output:0000101010000

The steps to design a non-overlapping 101 Mealy sequence detector are:

Step 1: Development of the state diagram –
The state diagram of a Mealy machine for a 101 sequence detector is:



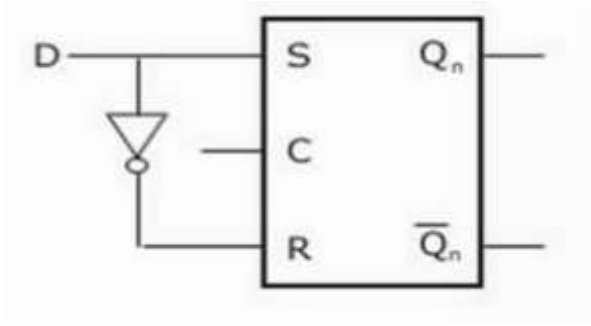
Step 2: Assignment of the code–
Rule 1: States having the same next states for a given input condition should have adjacent assignments.
Rule 2:States that are the next states to a single state must be given adjacent assignments.
Rule 1 given preference over Rule 2.

S:

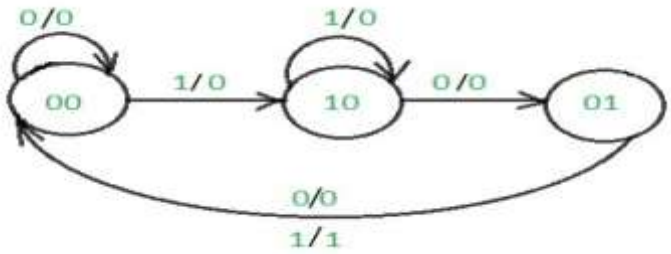
D,Q_N		
	0	0
	1	X

R:

D,Q_N		
	X	1
	0	0



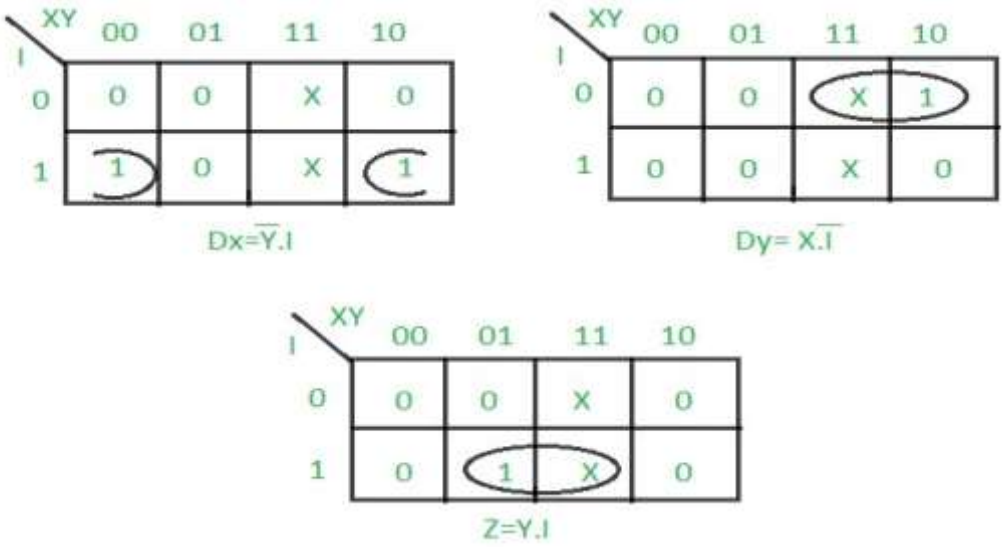
The state diagram after the code assignment is:



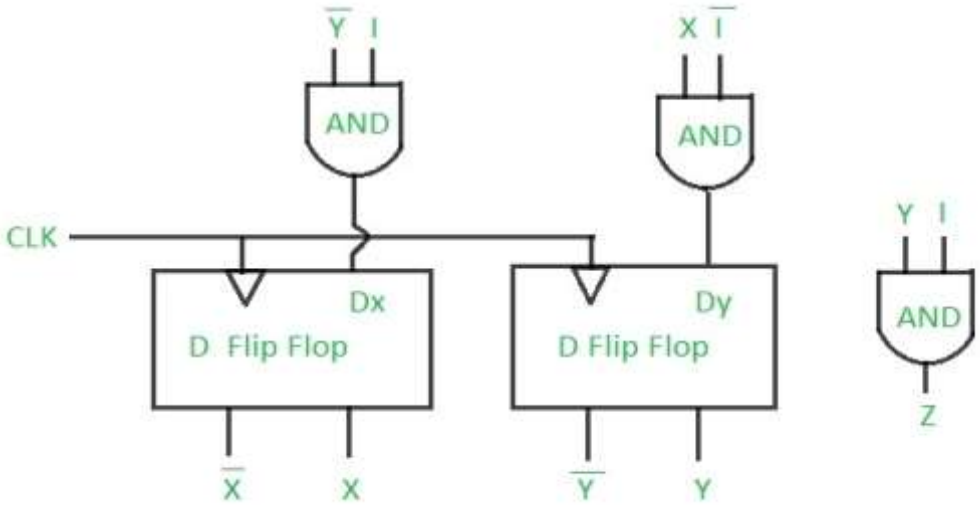
Step 3: Making Present State/Next State table –

Present States		i/p	Next States		Flip Flop Excitations		O/P
X	Y		X'	Y'	Dx	Dy	
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	1	1	0	1	0	0
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

Step 4: Draw K-maps for Dx, Dy, and output (Z) –



Step 5: Final implementation of the circuit –



This is the desired circuit for a Mealy 101 non-overlapping sequence detector.

Reference Books:

1. John Yarbrough, ―Digital Logic Applications and Design, Cengage Learning, ISBN – 13: 978-81-315-0058-3

2. D. Leach, Malvino, Saha, ―Digital Principles and Applications‖, Tata McGraw Hill, ISBN – 13:978-0-07-014170-4.

3. Anil Maini, ―Digital Electronics: Principles and Integrated Circuits‖, Wiley India Ltd, ISBN:978-81-265-1466-3.

4. Norman B & Bradley, ―Digital Logic Design Principles, Wiley India Ltd, ISBN:978-81-265-1258