

Unit IV

Sequential Logic Design

Comparison between combinational and sequential circuits

Combinational circuit	Sequential circuit
<ol style="list-style-type: none"> 1. In combinational circuits, the o/p variables at any instant of time are dependent only on the present input variables 2. Memory unit is not required in combinational circuit 3. These circuits are faster because the delay between the i/p and o/p is due to propagation delay of gates only 4. Easy to design 5. Examples: Multiplexer, Demultiplexer, encoders, decoders, etc 	<ol style="list-style-type: none"> 1. In sequential circuits the output variables at any instant of time are dependent not only on the present input variables, but also on the present state 2. Memory unit is required to store the past history of the input variables 3. Sequential circuits are slower than combinational circuits 4. Comparatively hard to design 5. Examples: Flip flops, registers, counters, state machine, etc.

Combinational logic refers to circuits whose output is strictly dependent on the present value of the inputs. As soon as inputs are changed, the information about the previous inputs is lost, that is, combinational logic circuits have no memory. Although every digital system is likely to have combinational circuits, most systems encountered in practice also include memory elements, which require that the system be described in terms of sequential logic. Circuits whose output depends not only on the present input value but also the past input value are known as **sequential logic circuits**. The mathematical model of a sequential circuit is usually referred to as a **sequential machine**.

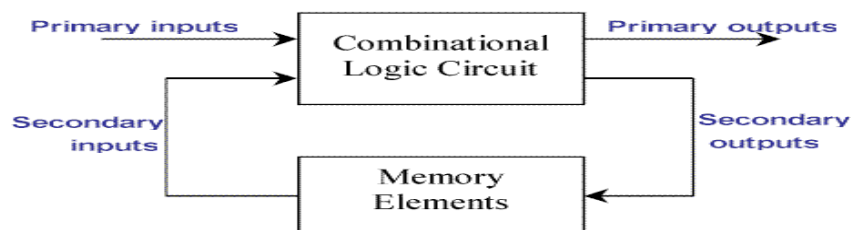
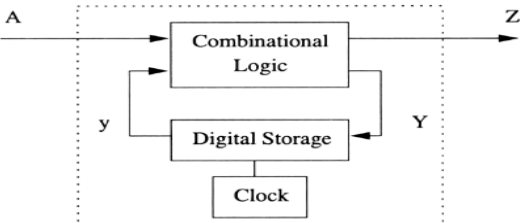
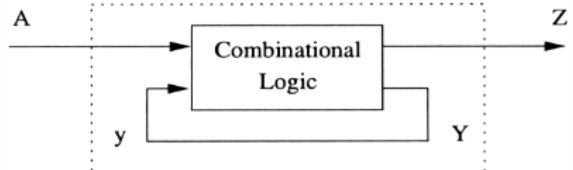


Fig: Sequential circuits

Classification of sequential circuits: Sequential circuits may be classified as two types.

1. Synchronous sequential circuits
2. Asynchronous sequential circuits

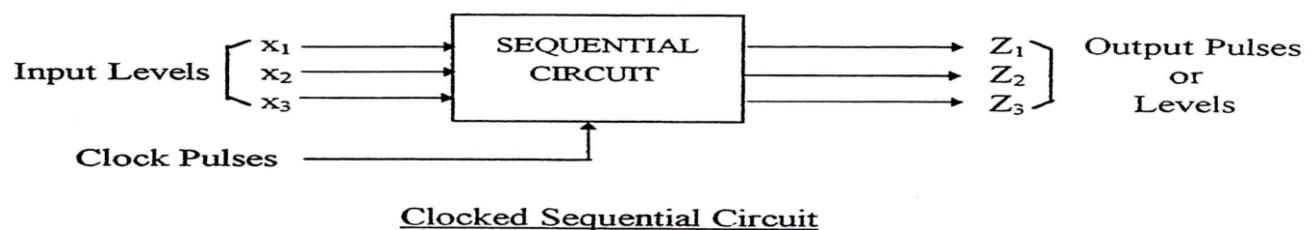
Synchronous Sequential Circuit	Asynchronous Sequential Circuit
<ul style="list-style-type: none"> It is easy to design. 	<ul style="list-style-type: none"> It is difficult to design.
<ul style="list-style-type: none"> A clocked flip flop acts as memory element. 	<ul style="list-style-type: none"> An unlocked flip flop or time delay is used as memory element.
<ul style="list-style-type: none"> They are slower as clock is involved. 	<ul style="list-style-type: none"> They are comparatively faster as no clock is used here.
<ul style="list-style-type: none"> The states of memory element is affected only at active edge of clock, if input is changed. 	<ul style="list-style-type: none"> The states of memory element will change any time as soon as input is changed.
	

Level Mode and Pulse Mode Synchronous Sequential Circuits:

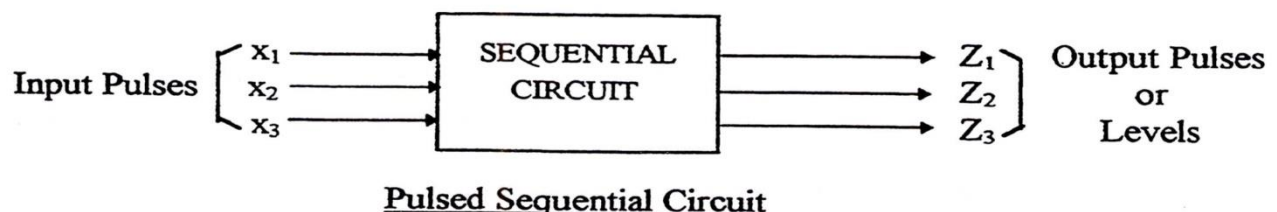
In synchronous circuits the input are pulses (or levels and pulses) with certain restrictions on pulse width and circuit propagation delay. Therefore synchronous circuits can be divided into clocked sequential circuits and unclocked or pulsed sequential circuits.

In a **clocked sequential circuit** which has flip-flops or, in some instances, gated latches, for its memory elements there is a (synchronizing) periodic clock connected to the clock inputs of all the memory elements of the circuit, to synchronize all internal changes of state.

Hence the operation of the entire circuit is controlled and synchronized by the periodic pulses of the clock.



On the other hand in an **unclocked or pulsed sequential circuit**, such a clock is not present. Pulse mode circuits require two consecutive transitions between 0 and 1 - that is a 0-pulse or a 1 pulse to alter the circuit's state. A pulse -mode circuit is designed to respond to pulses of certain duration; the constant signals between the pulses are "null" or "spacer" signals, which do not affect the circuit's behavior



From the above block diagrams we can note the following:

- 1) **Pulse outputs:** For pulsed sequential circuits these occur only for the duration of the respective input pulse and in some cases for duration considerably less. For clocked sequential circuits these outputs occur for the duration of the clock pulse.

2) Level outputs: These change state at the start of the respective input or clock pulse and remain in that state until the next state of output is required.

A requirement of synchronous sequential circuits is that the duration of the activating pulse or clock pulse should be sufficiently low in value that the pulse (or clock) has disappeared by the time the secondaries (the flip-flops outputs) have taken on their new value; otherwise the circuit will change state again. This means that the storage elements (flip-flops) should be edge-triggered devices (for example: D-type flip-flop, the JK flip-flop and their derivatives).

Level Mode and Pulse Mode Asynchronous Sequential Circuits:

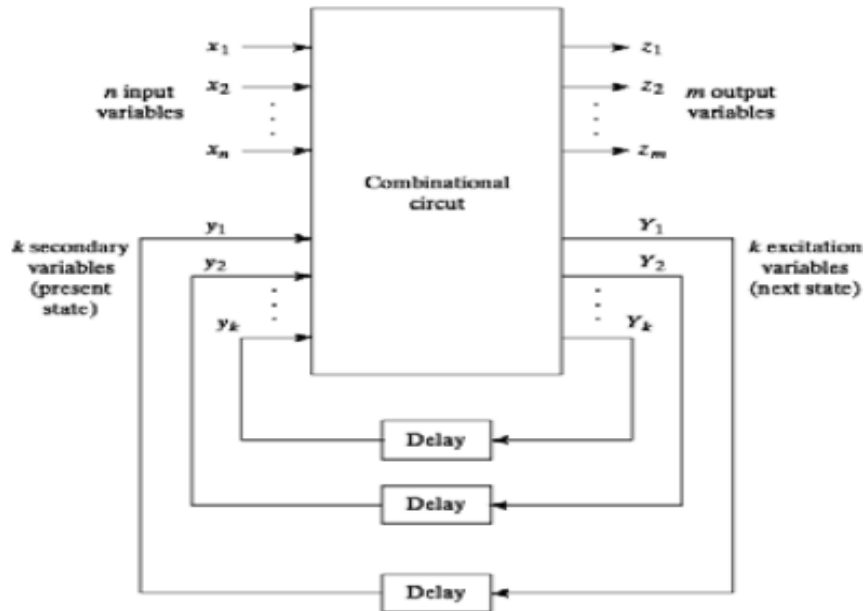


Fig shows a block diagram of an asynchronous sequential circuit. It consists of a combinational circuit and delay elements connected to form the feedback loops. The present state and next state variables in asynchronous sequential circuits called *secondary variables* and *excitation variables* respectively.

When an input variable changes in value, the secondary variables, i.e. $y_1, y_2, y_3, \dots, y_k$ do not change instantaneously. The combinational circuit generates k excitation variables which give the next state of the circuit. The excitation variables are propagated through delay elements to become the new present state for the secondary variables, i.e. $y_1, y_2, y_3, \dots, y_k$. In steady state condition excitation and secondary variables are the same i.e. stable ($y_i = Y_i$), but during transition they are different.

The input variables are considered as; there are two types of asynchronous circuits: fundamental mode circuits and pulse mode circuits.

Level/Fundamental Mode Circuits assumes that:

- The input variables change only when the circuit is stable
- Only one input variable can change at a given time
- Inputs are levels and not pulses

A pulse mode circuit assumes that:

- The input variables are pulses instead of levels
- The width of the pulses is long enough for the circuit to respond to the input
- The pulse width must not be so long that is still present after the new state is reached.

Latches and flip-flops:

Latches	Flip Flops
Latches are building blocks of sequential circuits and these can be built from logic gates	Flip flops are also building blocks of sequential circuits. But, these can be built from the latches.
Latch continuously checks its inputs and changes its output correspondingly.	Flip flop continuously checks its inputs and changes its output correspondingly only at times determined by clocking signal
The latch is sensitive to the duration of the pulse and can send or receive the data when the switch is on	Flipflop is sensitive to a signal change. They can transfer data only at the single instant and data cannot be changed until next signal change. Flip flops are used as a register.
It is based on the enable function input	It works on the basis of clock pulses
It is a level triggered, it means that the output of the present state and input of the next state depends on the level that is binary input 1 or 0.	It is an edge triggered, it means that the output and the next state input changes when there is a change in clock pulse whether it may be a +ve or -ve clock pulse.

Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted. In other words, when they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

A flip-flop (FF), known more formally as a bistable multivibrator, has two stable states. It can remain in either of the states indefinitely. Its state can be changed by applying the proper triggering signal. It is also called a binary or one-bit memory.

There are basically four main types of latches and flip-flops: S-R, D, J-K, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations. In this chapter, we will look at the operations of the various latches and flip-flops. The flip-flops has two outputs, labeled Q and Q'. The Q output is the normal output of the flip flop and Q' is the inverted output.

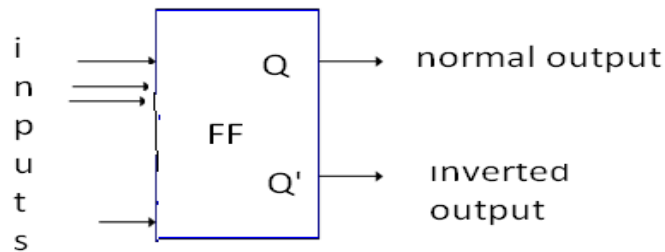


Figure: General flip-flop symbol

A latch may be an active-HIGH input latch or an active-LOW input latch. Active-HIGH means that the SET and RESET inputs are normally resting in the LOW state and one of them will be pulsed HIGH whenever we want to change latch outputs. Active-LOW means that the SET and RESET inputs are normally resting in the HIGH state and one of them will be pulsed LOW whenever we want to change latch outputs.

S-R latch:

The latch has two outputs Q(HIGH(1) or LOW(0)) and Q'(complement of Q) and two inputs labelled S and R.

It can be constructed by:

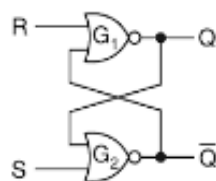
- using two NOR gates, an active-HIGH S-R latch and
- using two NAND gates, an active-LOW S-R latch

NOR gate S-R latch (active-high S-R latch):

When the circuit is switched on the latch may enter into any state. If $Q=1$, then $Q'=0$, which is called SET state (i.e SET input is made HIGH). If $Q=0$, then $Q'=1$, which is called RESET state (i.e RESET input is made HIGH). Whether the latch is in SET state or RESET state, it will continue to remain in the same state, as long as the power is not switched off. If both the inputs S and R are made LOW, there is no change in the state of the latch (i.e) same state in which it was, prior to the application of inputs. If both are HIGH, the output is unpredictable.



(a) Logic symbol



(b) Logic diagram

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Indeterminate (invalid)
1	1	1	x	

(c) Truth table

The analysis of the operation of the active-HIGH NOR latch can be summarized as follows.

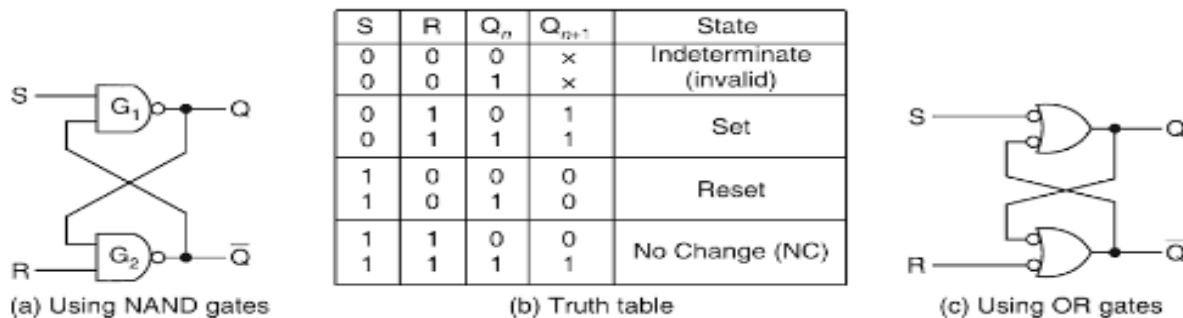
1. SET=0, RESET=0: This is normal resting state of the NOR latch and it has no effect on the output state. Q and Q' will remain in whatever state they were prior to the occurrence of this input condition.
2. SET=1, RESET=0: This will always set Q=1, where it will remain even after SET returns to 0
3. SET=0, RESET=1: This will always reset Q=0, where it will remain even after RESET returns to 0
4. SET=1, RESET=1: This condition tries to SET and RESET the latch at the same time, and it produces Q=Q'=0. If the inputs are returned to zero simultaneously, the resulting output state is erratic and unpredictable. This input condition should not be used.

The SET and RESET inputs are normally in the LOW state and one of them will be pulsed HIGH, whenever we want to change the latch outputs.

NAND gate S-R latch (active-low S-R latch):

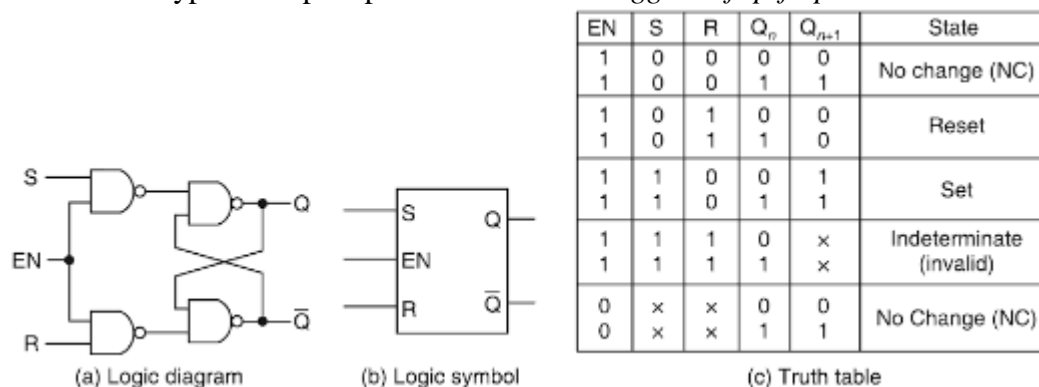
NAND latch is the fundamental building block in constructing a flip-flop. It has the property of holding on to any previous output, as long as it is not disturbed.

The operation of NAND latch is the reverse of the operation of NOR latch. If 0's are replaced by 1's and 1's are replaced by 0's we get the same truth table as that of the NOR latch

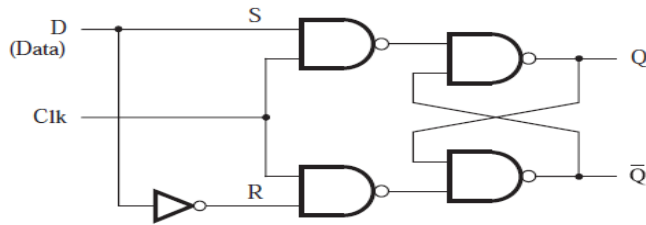


Gated Latches (Clocked Flip-Flops):

The gated S-R latch: The gated S-R latch requires an ENABLE (EN) input. The S and R inputs will control the state of the flip-flop only when the ENABLE is HIGH. The ENABLE input may be clock. This type of flip-flop responds to the changes in inputs only as long as the clock is HIGH, these types of flip-flops are called *level triggered flip-flops*.



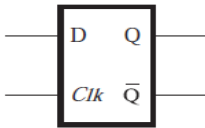
The gated D-latch:



(a) Circuit

Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

(b) Characteristic table



(c) Graphical symbol

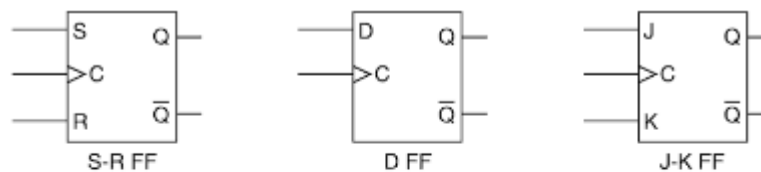
Edge-Triggered flip-flop:

In synchronous systems, the exact times at which any output can change states are determined by a signal commonly called the *clock*. The flip-flops using the clock signal are called the *clocked flip-flops*. Clocked flip-flops may be positive edge-triggered or negative edge-triggered.

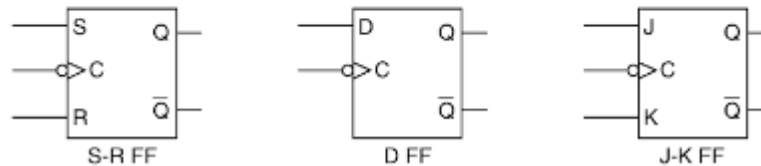
Positive edge-triggered flip-flops are those in which ‘state transitions’ take place only at the positive-going (0 to 1, or LOW to HIGH) edge of the clock pulse and it is indicated by a ‘triangle’.

Negative edge-triggered flip-flops are those in which ‘state transitions’ take place only at the negative-going (1 to 0, or HIGH to LOW) edge of the clock pulse and it is indicated by a ‘triangle’ with a bubble at the clock terminal of the flip-flop.

There are three basic types: S-R, J-K, and D. D and J-K are mostly used. D and J-K are derived from S-R flip-flop.



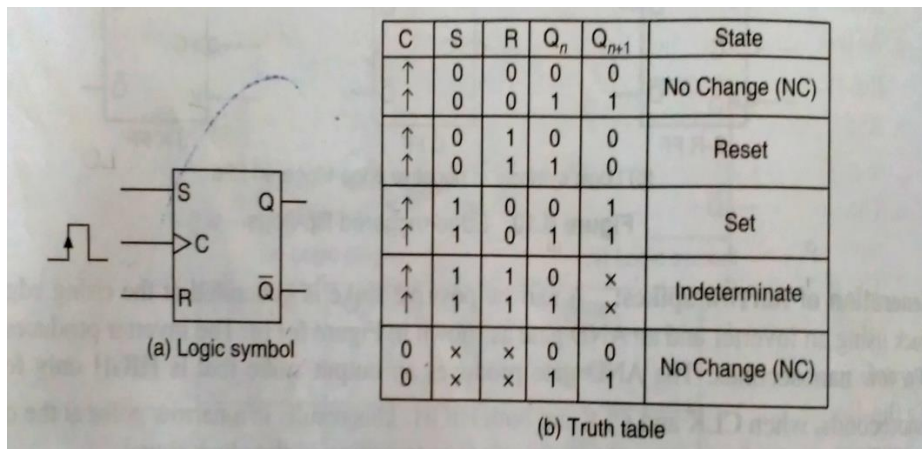
(a) Logic symbols of positive edge-triggered FFs



(b) Logic symbols of negative edge-triggered FFs

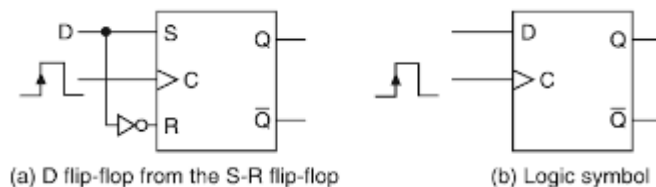
The edge-triggered S-R flip-flop:

The S and R inputs of the S-R flip-flop are called *synchronous control inputs* because data on these inputs affect the flip-flop’s output only on the triggering (positive going) edge of the clock pulse.



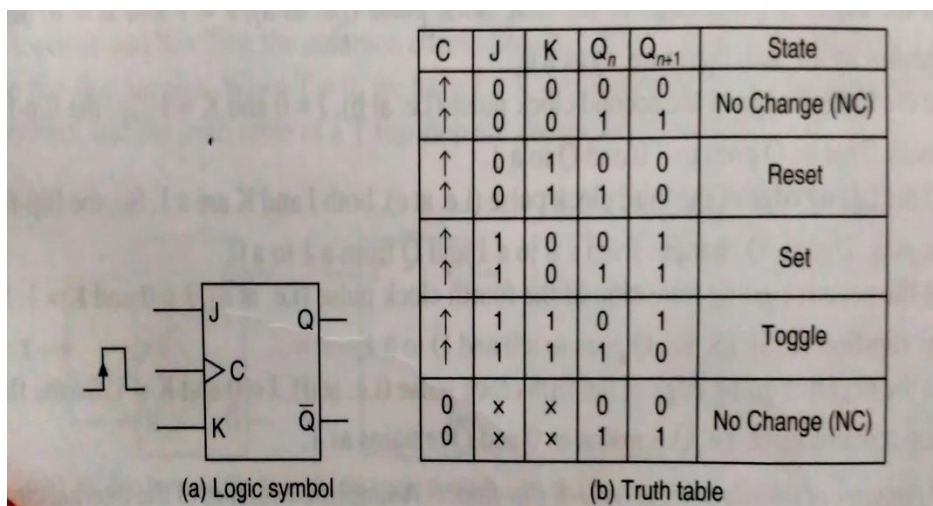
The edge-triggered D flip-flop:

The edge-triggered D flip-flop has only one input terminal. The D flip-flop is obtained from an S-R flip-flop by just putting one inverter between the S and R terminals. D (data) input.



The edge-triggered J-K flip-flop:

The functioning of the J-K flip-flop is identical to that of the S-R flip-flop, except that it has no invalid state. When $J=1$ and $K=1$, the flip-flop toggles, i.e. goes to the opposite state at the positive- going edge of the clock pulse.

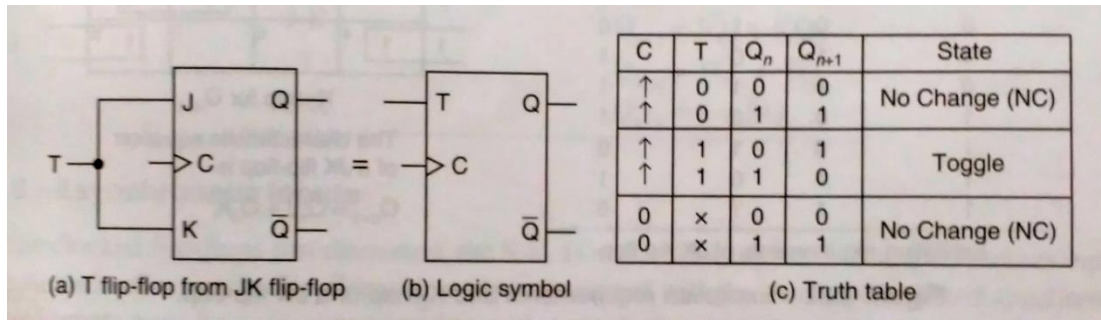


The edge-triggered T flip-flop:

A T flip-flop has a single control input, labeled T for toggle.

When $T=1$, we have $J=K=1$, and the flip-flop toggles.

When $T=0$, we have $J=K=0$, and so there is no change of state.



Triggering and Characteristic Equations of Flip-Flops:

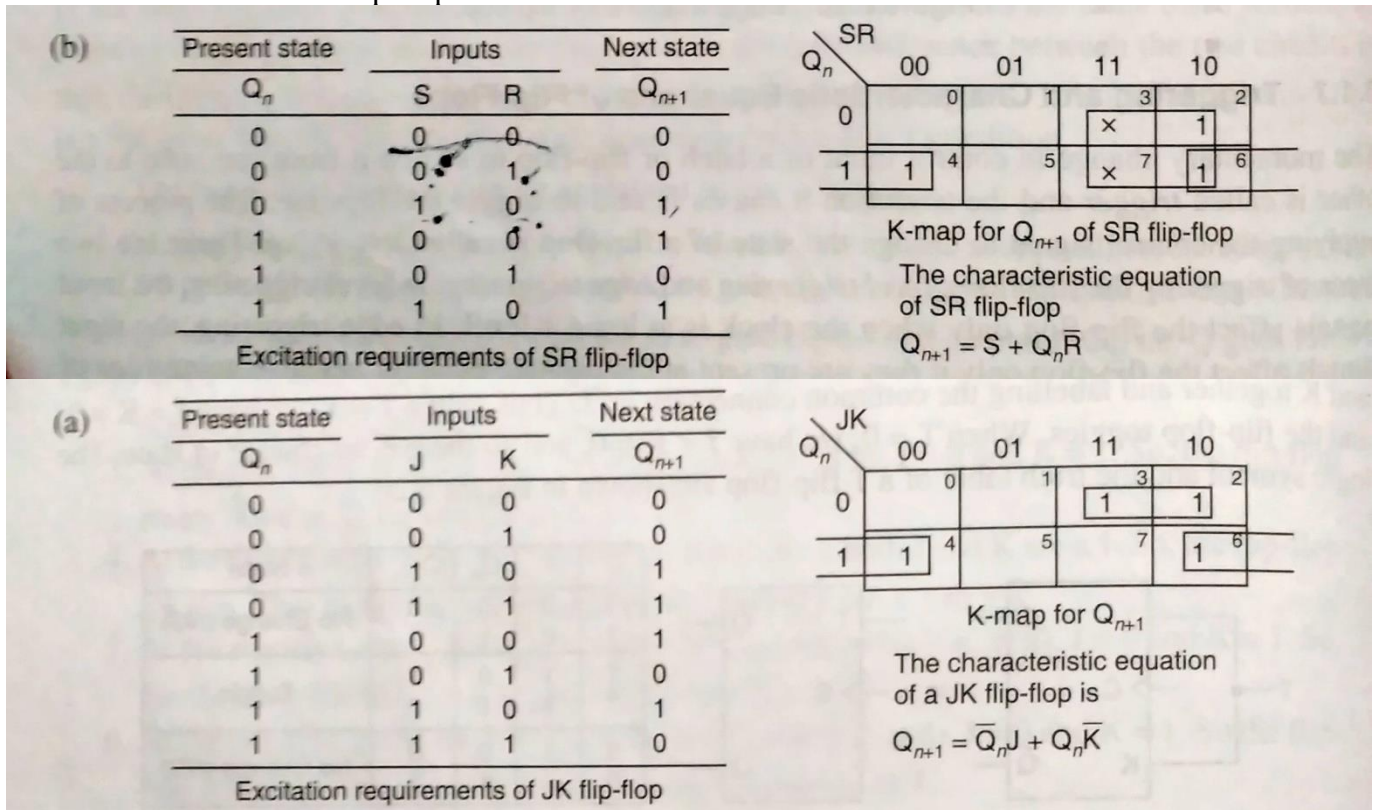
The momentary change in control input of a latch or flip-flop to switch it from one state to the other is called *trigger* and the transition it causes is said to trigger the flip-flop.

The process of applying the control signal to change the state of a flip-flop is called *triggering*.

There are two types of triggering the flip-flops:

- Level triggering: The input signals affect the flip-flop only when the clock is at logic 1 level.
- Edge triggering: The input signals affect the flip-flop only if they are present at the positive going or negative going edge of the clock pulse.

The characteristic table of flip-flops:



Present state	Input	Next state
Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Excitation requirements of D flip-flop

$Q_n \backslash D$	0	1
0	0	1
1	2	3

K-map for Q_{n+1} of D flip-flop

The characteristic equation of D flip-flop is $Q_{n+1} = D$

Present state	Input	Next state
Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation requirements of T flip-flop

$Q_n \backslash T$	0	1
0	0	1
1	1	3

K-map for Q_{n+1} of T flip-flop

The characteristic equation of T flip-flop is

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

Table Characteristic equations of flip-flop

Flip-flop	Characteristic equation
D	$Q_{n+1} = D$
J-K	$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$
T	$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$
S-R	$Q_{n+1} = S + \bar{R}Q_n$

Excitation tables:

SR excitation table:

Table S-R truth table

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

Table 6.4 S-R excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

J-K excitation table:

Table 6.5 J-K truth table

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

Table 6.6 J-K excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

D excitation table:

Table 6.7 D truth table

D	Q_{n+1}
0	0
1	1

Table 6.8 D excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

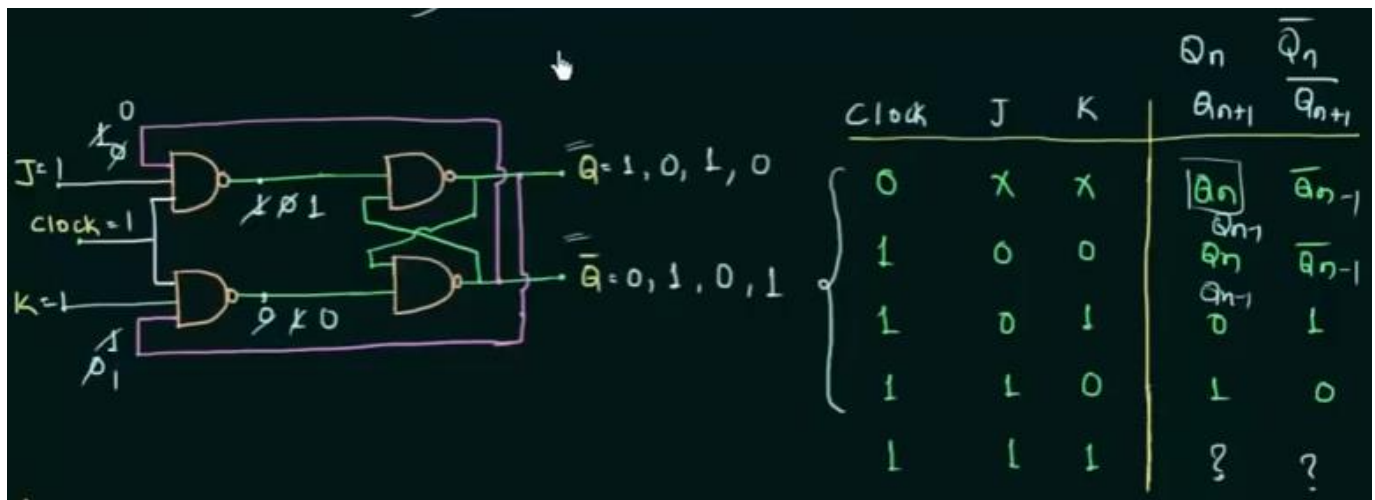
T excitation table:

Table T truth table	
T	Q_{n+1}
0	Q_n
1	\bar{Q}_n

Table T excitation table		
PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Race around Condition:

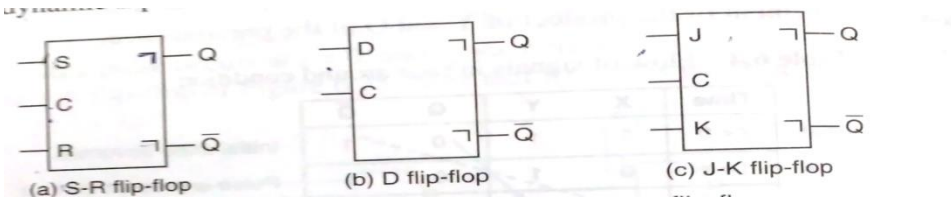
For the J-K flip-flop, consider the assignment of excitations $J=K=1$. If the width of the clock pulse is too long, the state of the flip-flop will keep changing from 0 to 1, 1 to 0, 0 to 1,...so on and at the end of the clock pulse, its state will be uncertain. This phenomenon is called the *race around condition*.



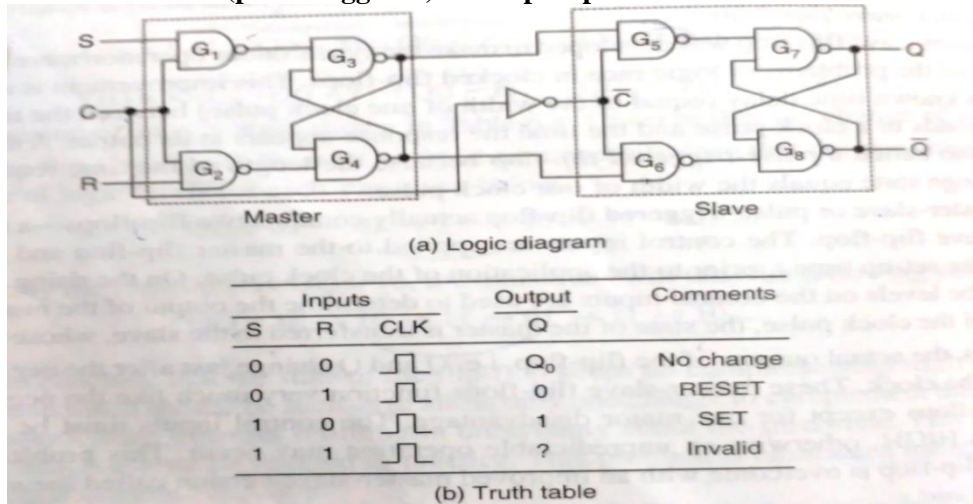
Master-Slave (Pulse-Triggered) Flip-Flops:

The master-slave flip-flop was developed to avoid the problems of logic race in clocked flip-flop. This flip-flop is also called a *pulse-triggered flip-flop* because the length of the time required for its output to change state equals the width of one clock pulse.

The master slave contains two flip-flops-a master flip-flop and slave flip-flop. On the rising edge of the clock pulse, the levels on the control inputs are used to determine the output of the master. On the falling edge of the clock pulse, the state of the master is transferred to the slave, whose outputs are Q and \bar{Q} . There are three basic types of master-slave flip-flops- S-R, D, and J-K. The key to identify master-slave flip-flop by its logic symbol is the postponed output symbol \neg at the outputs.

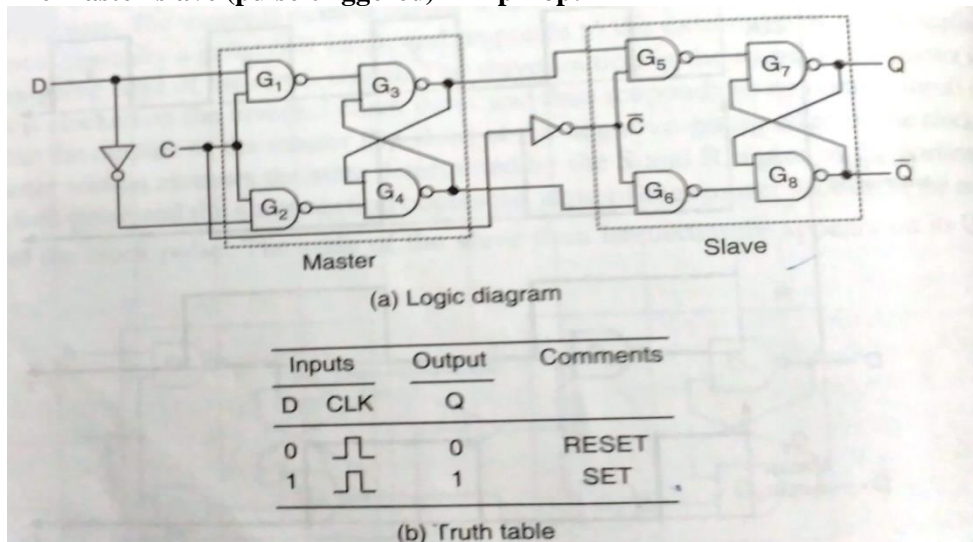


The master-slave (pulse-triggered) S-R flip-flop:

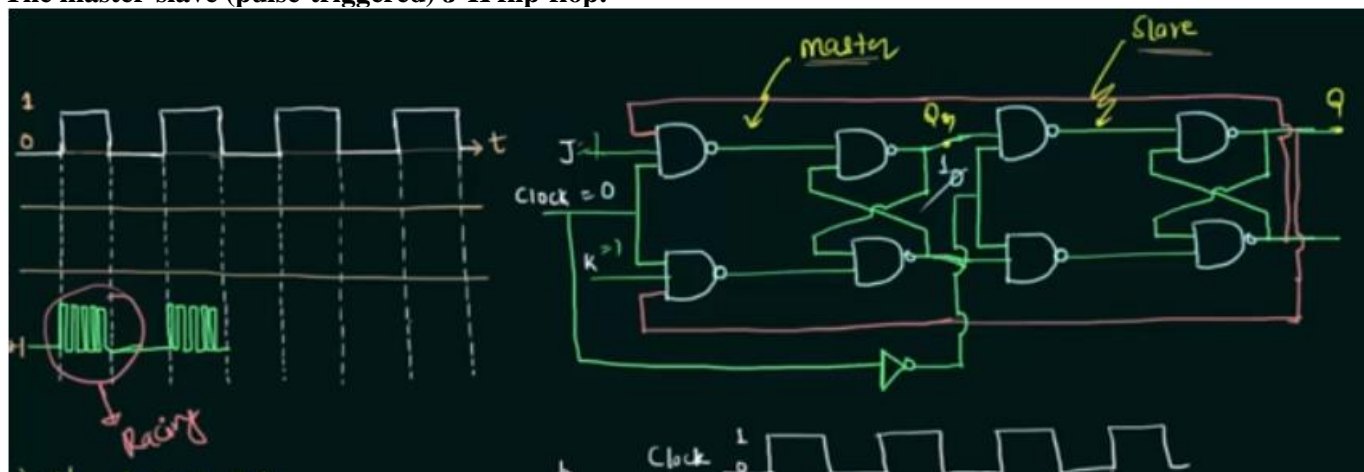


When clock input is 1, master flip-flop is activated and when clock is 0, slave flip flop is activated.

The master-slave (pulse-triggered) D flip-flop:

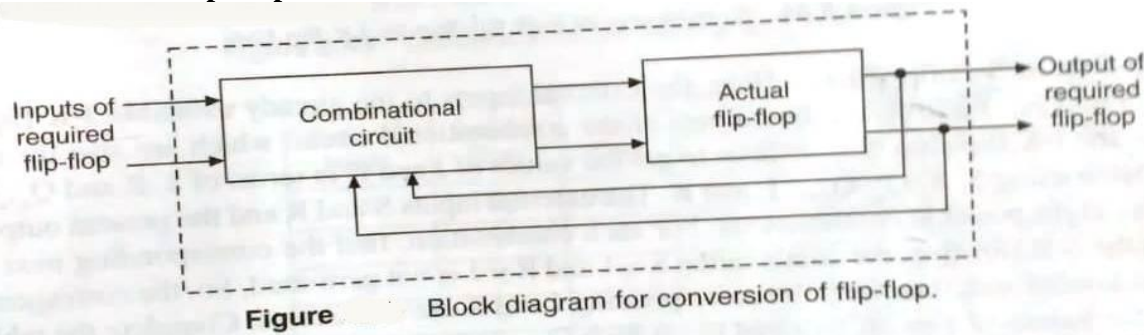


The master-slave (pulse-triggered) J-K flip-flop:

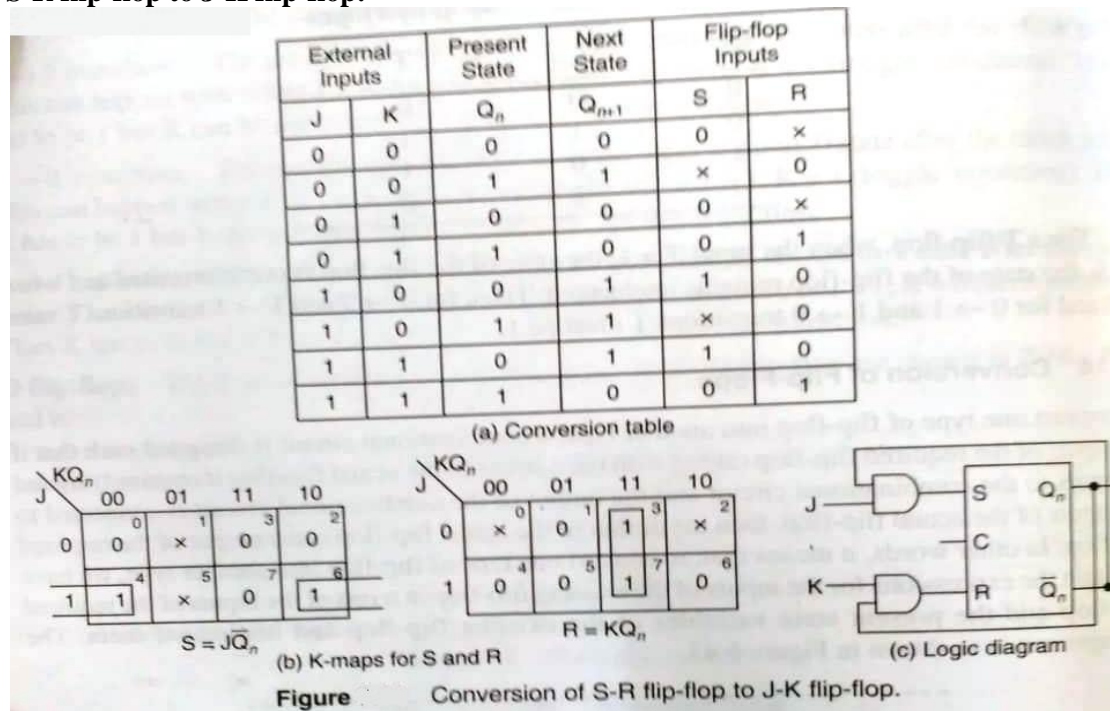


i.e it will not toggle before times in one clock cycle

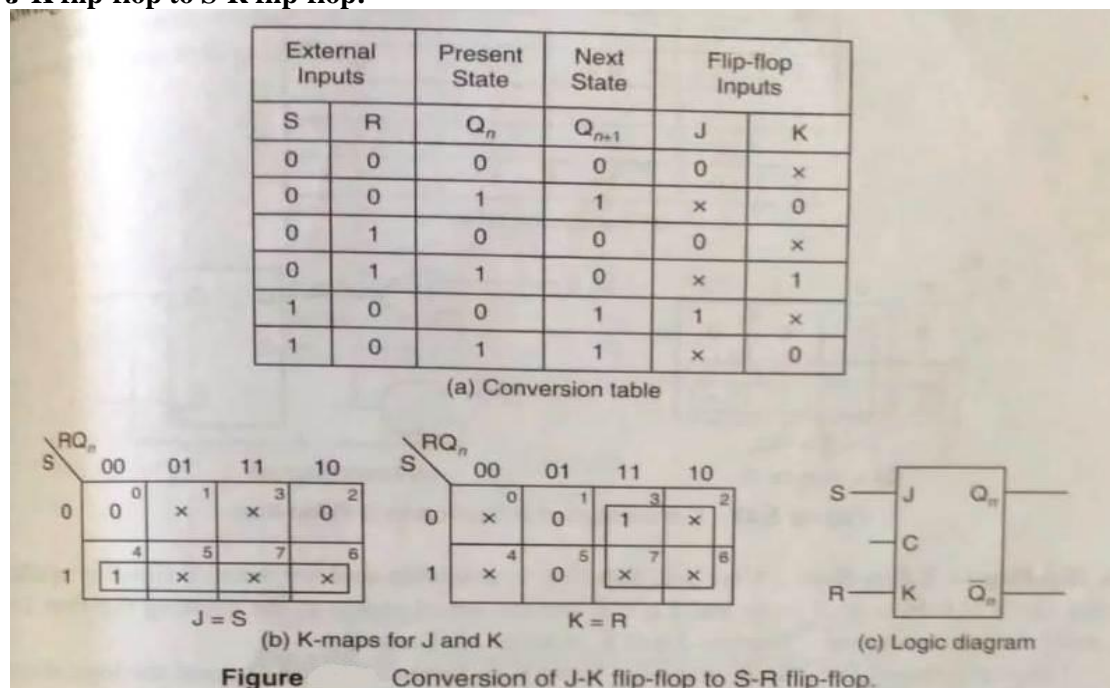
Conversion of Flip-Flops:



S-R flip-flop to J-K flip-flop:



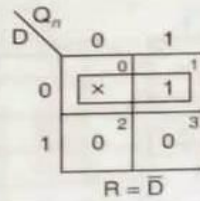
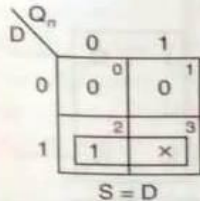
J-K flip-flop to S-R flip-flop:



S-R flip-flop to D flip-flop:

External Inputs	Present State	Next State	Flip-flop Inputs	
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

(a) Conversion table



(b) K-maps for S and R

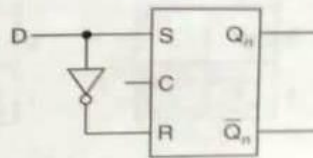
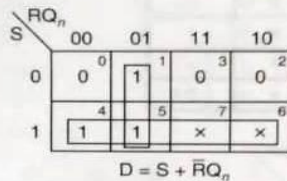


Figure Conversion of S-R flip-flop to D flip-flop.

D flip-flop to S-R flip-flop:

External Inputs		Present State	Next State	Flip-flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1

(a) Conversion table



(b) K-map for D

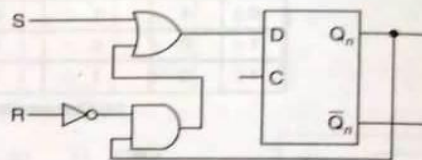
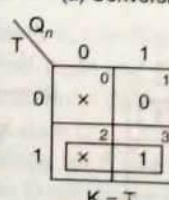
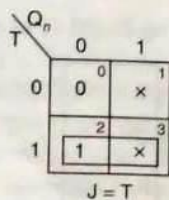


Figure Conversion of D flip-flop to S-R flip-flop.

J-K flip-flop to T flip-flop:

External Input	Present State	Next State	Flip-flop Inputs	
T	Q_n	Q_{n+1}	J	K
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1

(a) Conversion table



(b) K-maps for J and K

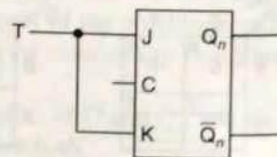
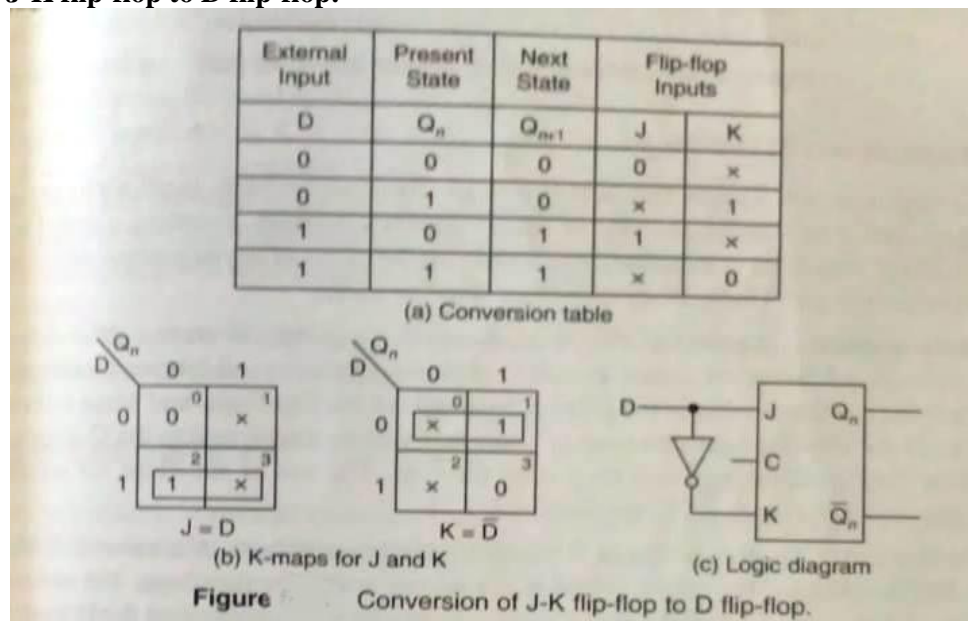
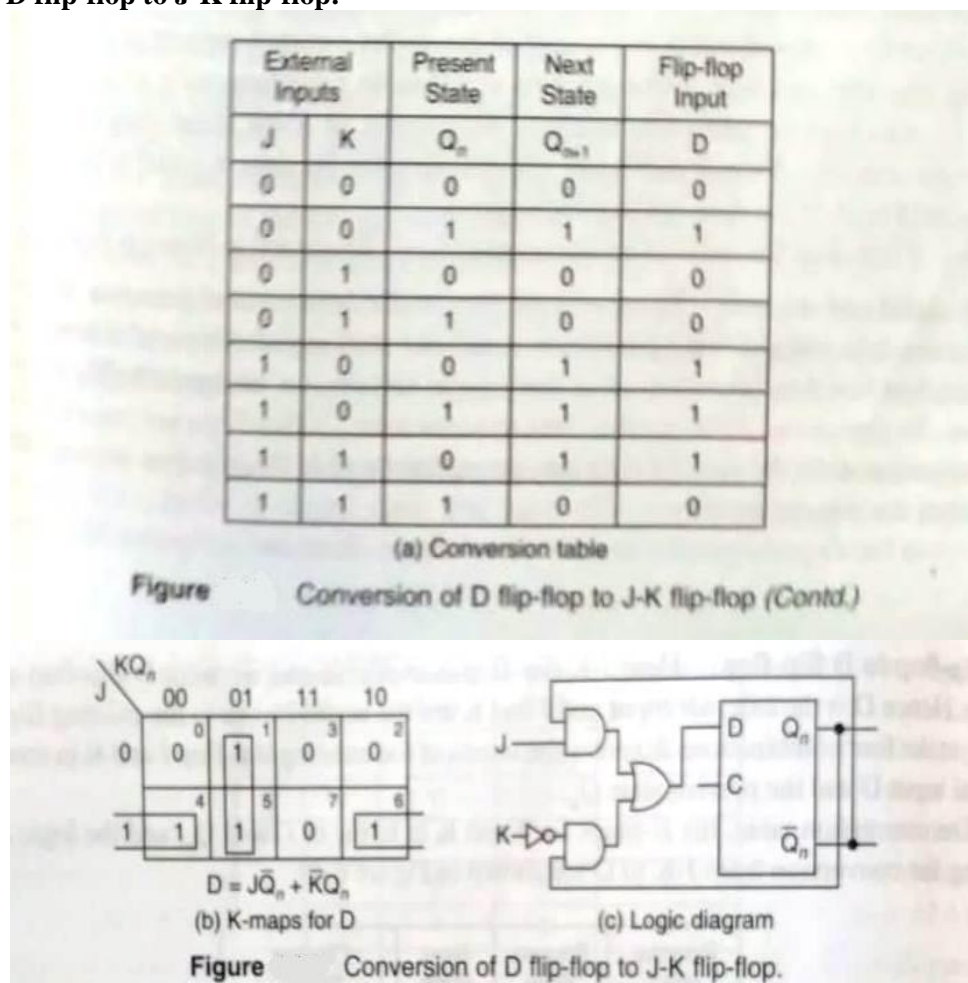


Figure Conversion of J-K flip-flop to T flip-flop.

J-K flip-flop to D flip-flop:



D flip-flop to J-K flip-flop:



Registers:

Flip-flop is a 1 bit memory cell which can store either 0 or 1.

To increase the storage capacity we have to use group of flip-flops which is known as register.

The n bit register consists of n number of flip-flops and is capable of storing n bit word.

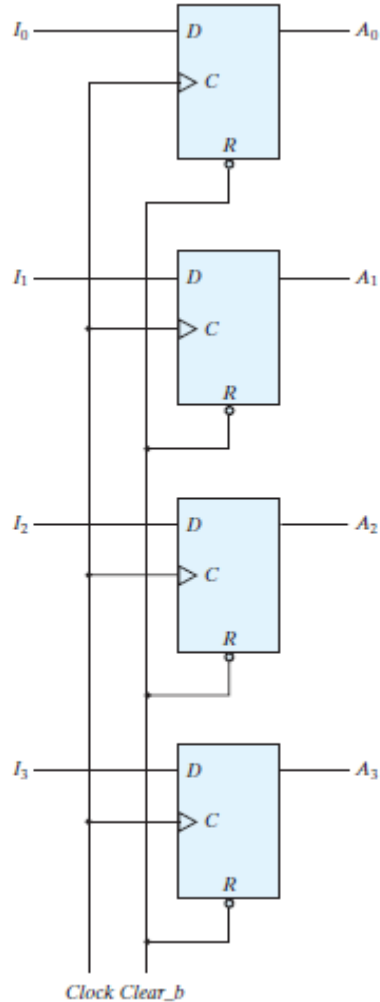


Fig 1: 4-Bit Register

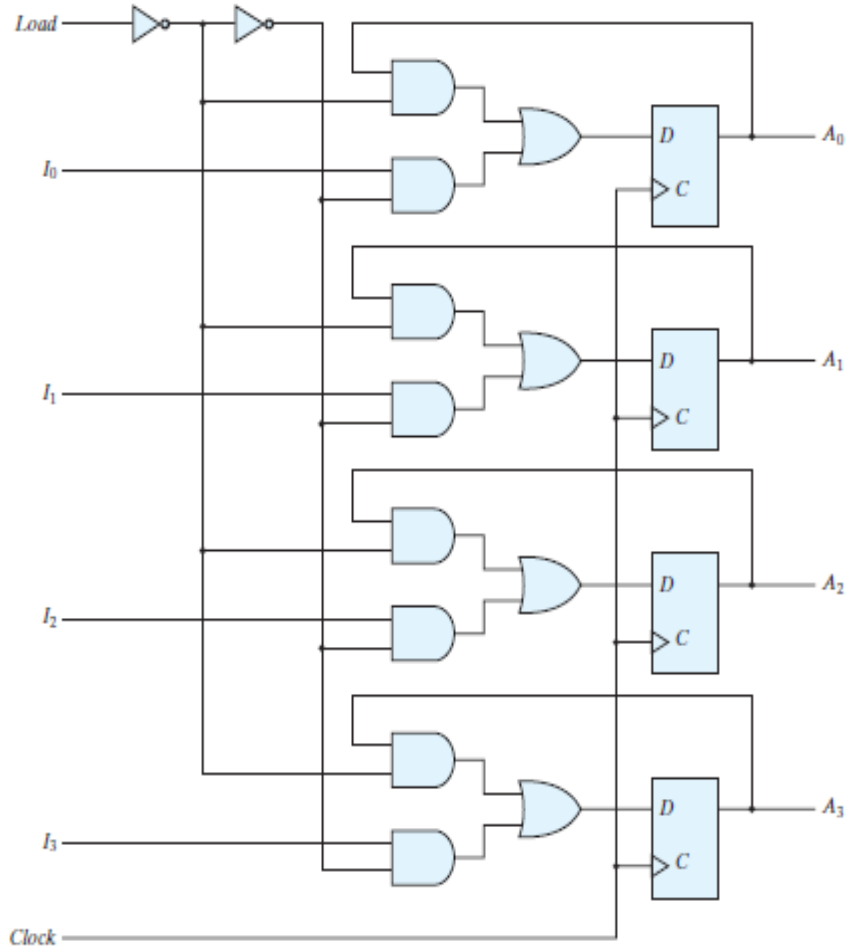


Fig 2: 4-Bit Register with Parallel Load

Fig1 shows a register constructed with 4 D-type flip-flops. The common clock input triggers all flip-flops on the positive edge of each pulse, and the binary data available at the four inputs are transferred into the 4-bit register. The outputs can be sampled at any time to obtain the binary information stored in the register. The input Clear_b goes to the active-low R (reset) input of all four flip-flops. When this input goes to 0, all flip-flops are reset asynchronously.

In Fig2, when the load input is 1, the data at the four external inputs are transferred into the register with the next positive edge of the clock. When the load input is 0, the outputs of the flip-flops are connected to their respective inputs.

Shift registers:

Shift registers are used for the storage and transfer of digital data. A shift register are used to momentarily store binary data at the output of decoder. Operations performed by shift register are complementation, multiplication and division.

Buffer register:

The buffer register is the simple set of registers. It is simply stores the binary word. The buffer may be controlled buffer. Most of the buffer registers used D Flip-flops.

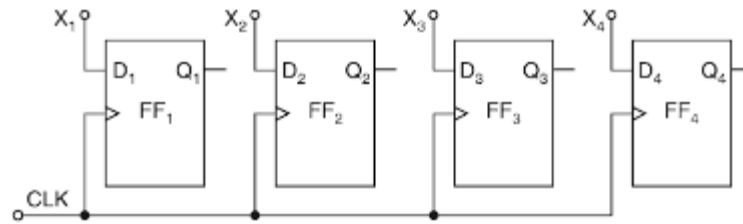
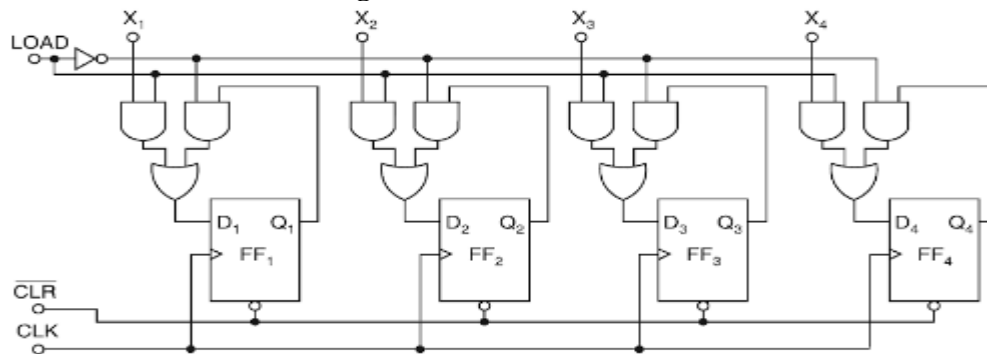


Figure: logic diagram of 4-bit buffer register

The figure shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals. On the application of clock pulse, the output word becomes the same as the word applied at the input terminals. i.e., the input word is loaded into the register by the application of clock pulse.

When the positive clock edge arrives, the stored word becomes: $Q_4Q_3Q_2Q_1 = X_4X_3X_2X_1$ or $Q = X$

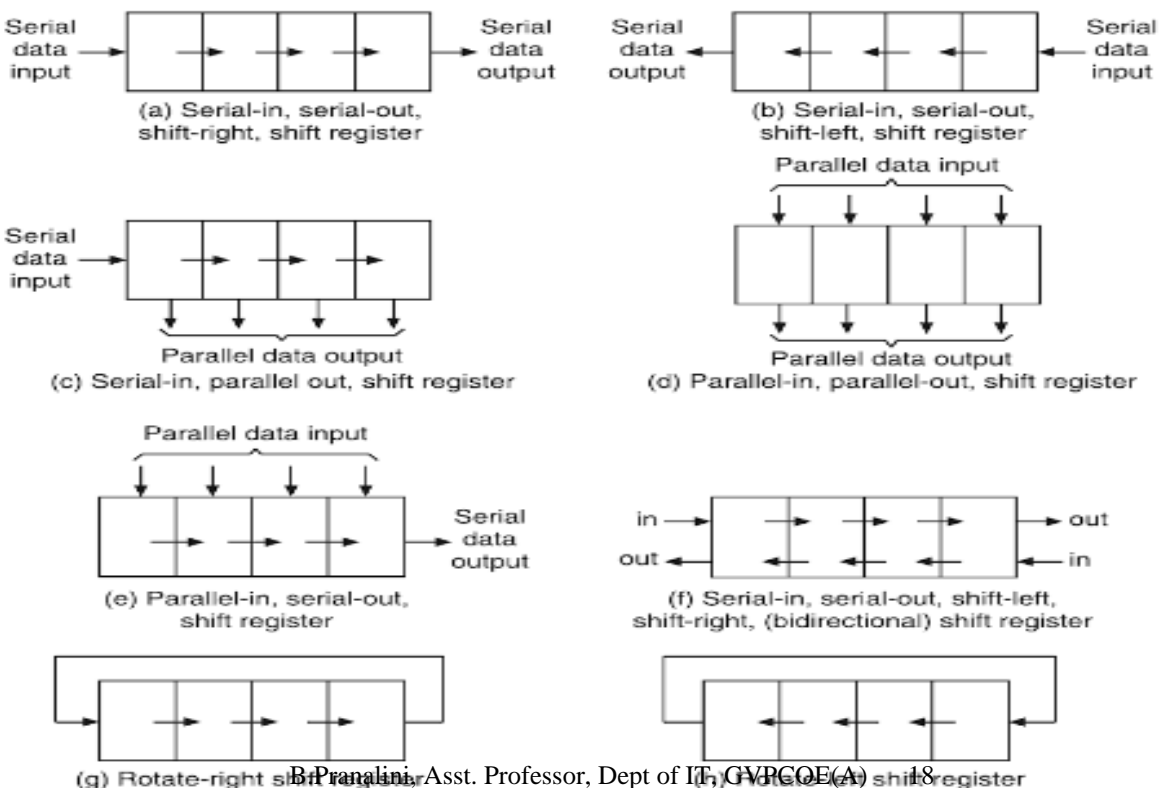
Controlled buffer register:



- If \overline{CLR} goes LOW, all the FFs are RESET and the output becomes, $Q = 0000$.
- When \overline{CLR} is HIGH, the register is ready for action. LOAD is the control input. When LOAD is HIGH, the data bits X can reach the D inputs of FF's. $Q_4Q_3Q_2Q_1 = X_4X_3X_2X_1$ or $Q = X$
- When LOAD is low, the X bits cannot reach the FF's. If \overline{LOAD} is HIGH, this forces each flip-flop output to feed back to its data input.

Data transmission in shift registers:

A no. of FFs connected together such that data may be shifted into and shifted out of them is called shift registers



Serial-in, Serial-out, Shift Register:

Shift register accepts data serially, i.e one bit at a time and also outputs data serially.

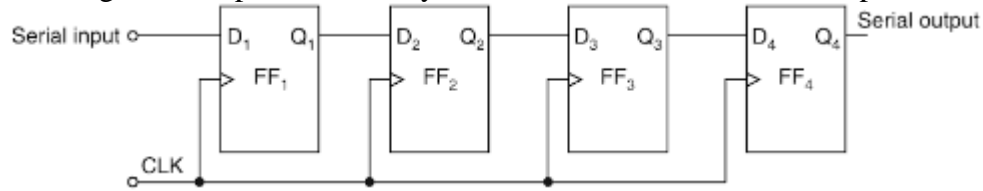
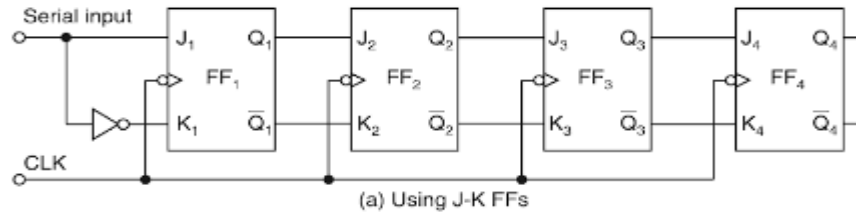
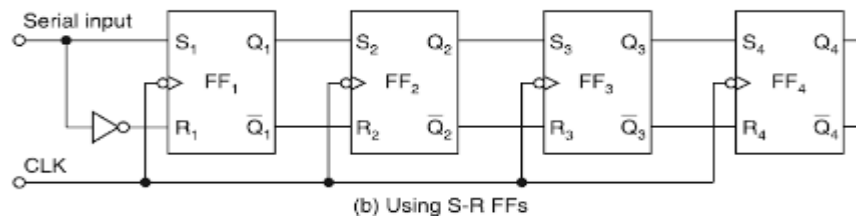


Fig: 4-bit serial-in, serial-out, shift-right, shift register

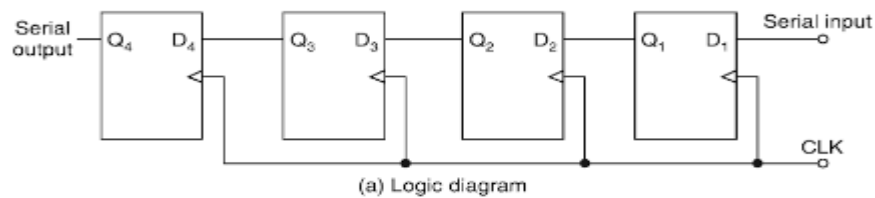


(a) Using J-K FFs

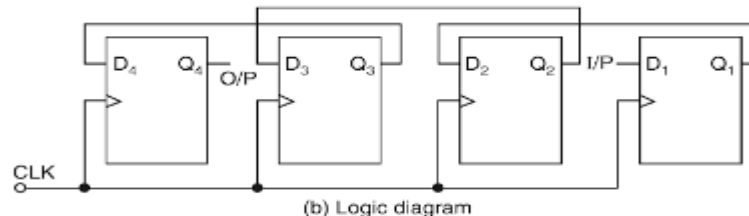


(b) Using S-R FFs

Fig: A 4-bit serial-in, serial-out, shift register



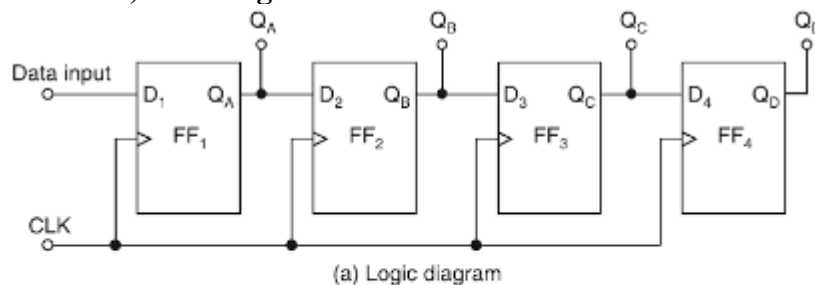
(a) Logic diagram



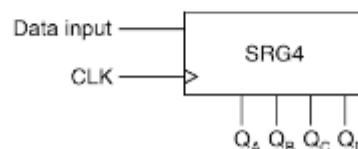
(b) Logic diagram

Fig: 4-bit serial-in, serial-out, shift-left, shift register

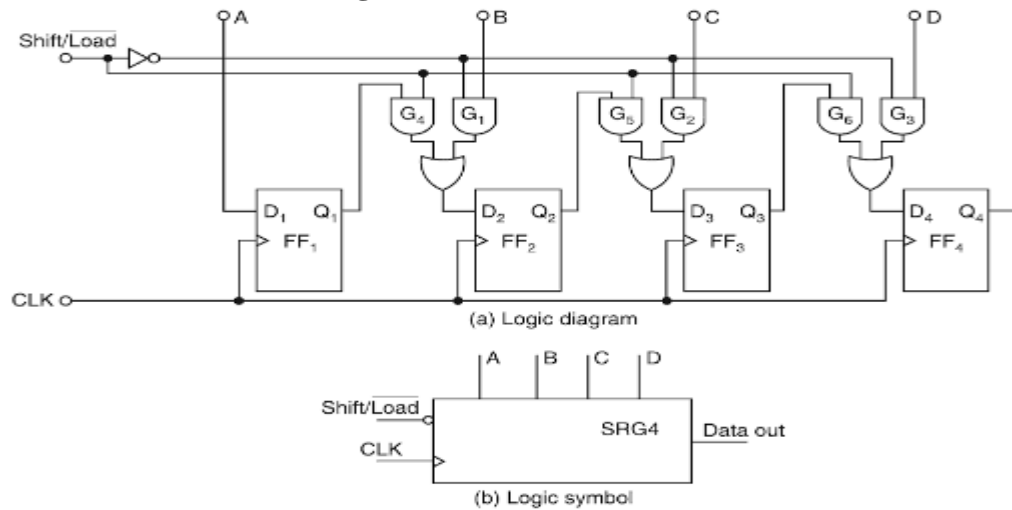
Serial-in, Parallel-out, Shift Register:



(a) Logic diagram



Parallel-in, Serial-out, Shift Register:



Parallel-in, Parallel-out, Shift Register:

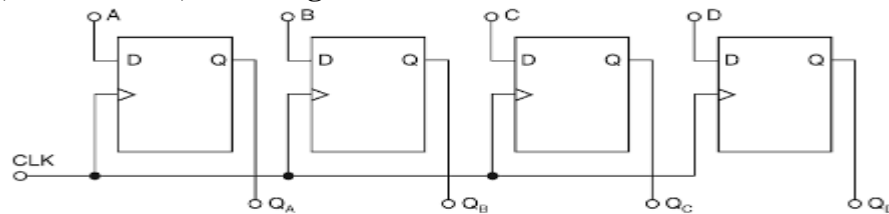
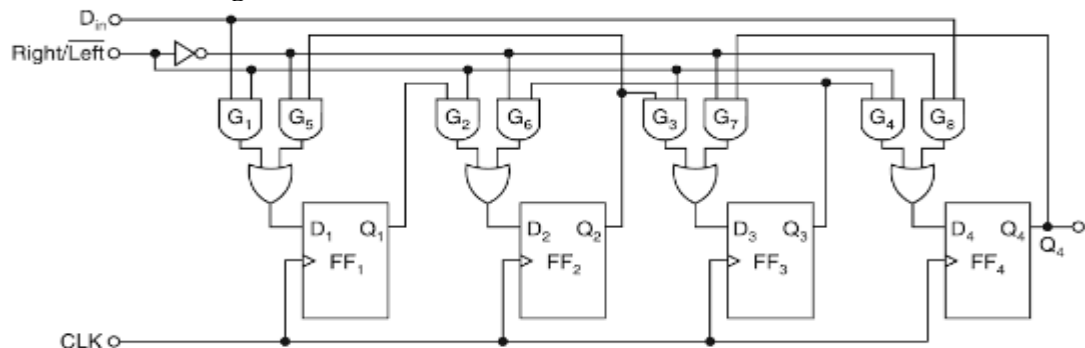


Fig: Logic diagram of a 4-bit parallel-in, parallel-out, shift register

Bidirectional shift register:



Universal Shift Registers:

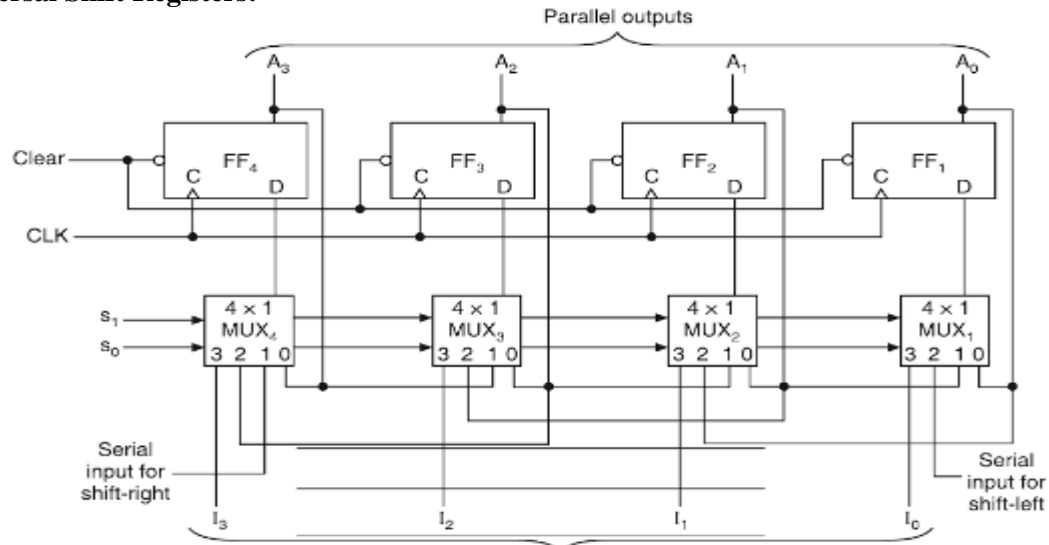


Figure 4-bit universal shift register.

The most general shift register has the following capabilities.

1. A clear control to clear the register to 0
2. A clock input to synchronize the operations
3. A shift-right control to enable the shift-right operation and serial input and output lines associated with the shift-right
4. A shift-left control to enable the shift-left operation and serial input and output lines associated with the shift-left
5. A parallel loads control to enable a parallel transfer and the n input lines associated with the parallel transfer
6. N parallel output lines
7. A control state that leaves the information in the register unchanged in the presence of the clock.

Table Function table for the register of Figure

Mode control		
S_1	S_0	Register operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Counters:

Counter is a device which stores (and sometimes displays) the number of times particular event or process has occurred, often in relationship to a clock signal. A Digital counter is a set of flip flops whose state change in response to pulses applied at the input to the counter. Counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters

In electronics counters can be implemented quite easily using register-type circuits such as the flip-flops and a wide variety of classifications exist:

- Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops
- Synchronous counter – all state bits change under control of a single clock
- Decade counter – counts through ten states per stage
- Up/down counter – counts both up and down, under command of a control input
- Ring counter – formed by a shift register with feedback connection in a ring
- Johnson counter – a *twisted* ring counter
- Cascaded counter
- Modulus counter.

Each is useful for different applications. Usually, counter circuits are digital in nature, and count in natural binary. Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters.

Occasionally there are advantages to using a counting sequence other than the natural binary sequence such as the binary coded decimal counter, a linear feed-back shift register counter, or a gray-code counter.

Counters are useful for digital clocks and timers and interval timers, VCR clocks, etc.

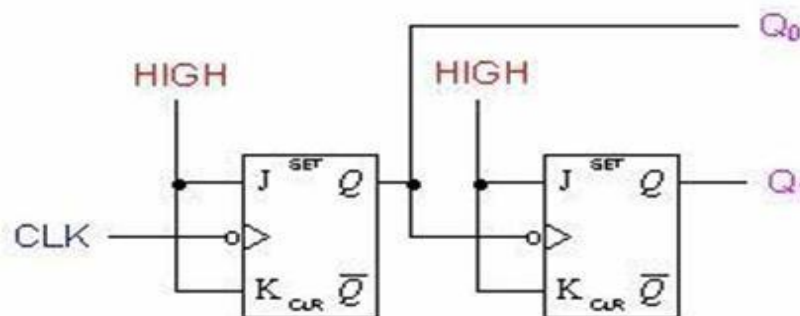
Asynchronous counters:

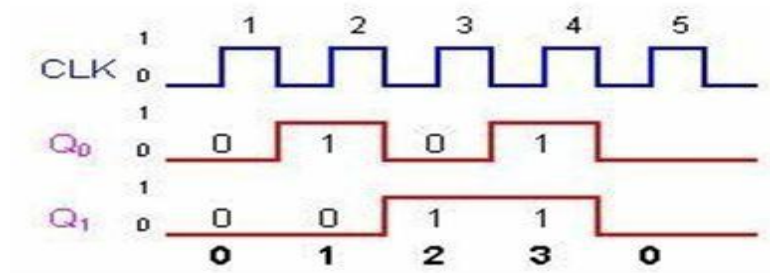
An asynchronous (ripple) counter is a single JK-type flip-flop, with its J (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), one will get another 1 bit counter that counts half as fast. Putting them together yields a two-bit counter:

Two-bit ripple up-counter using negative edge triggered flip flop:

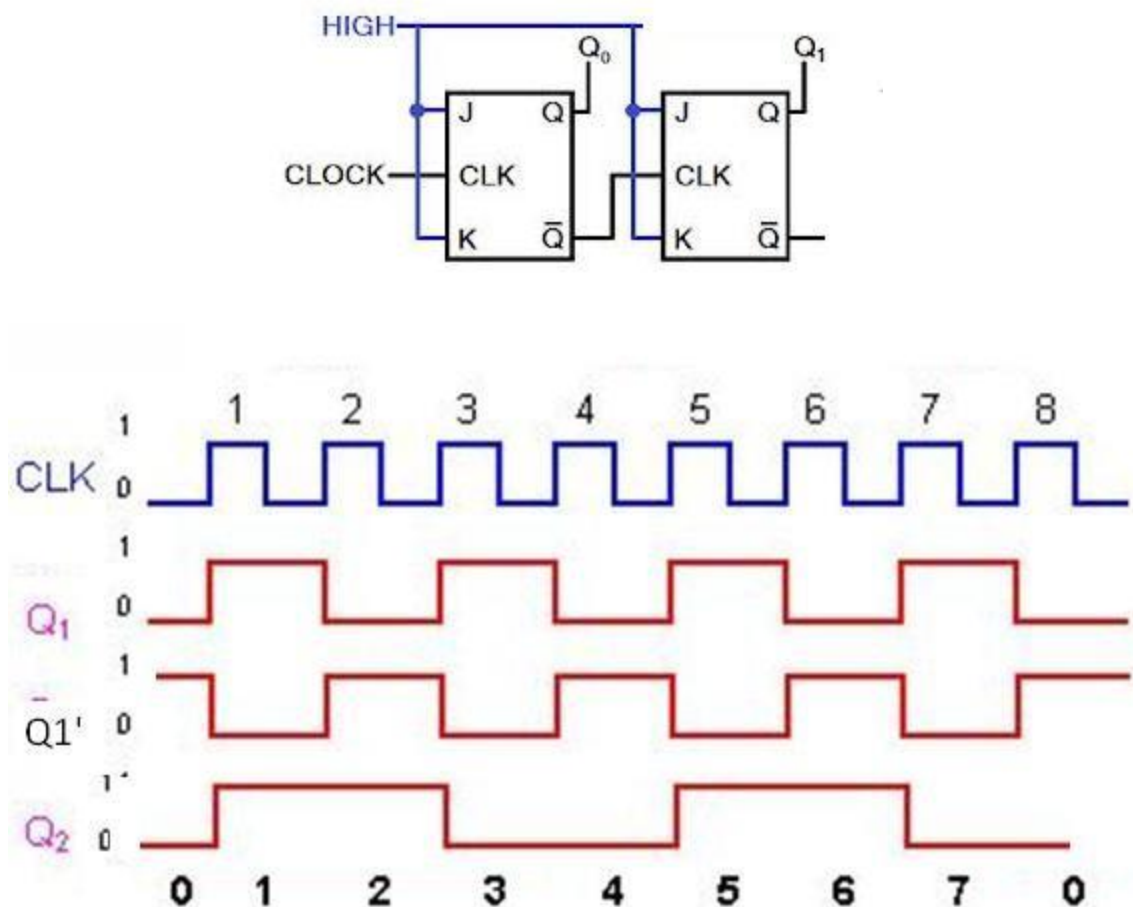
Two bit ripple counter used two flip-flops. There are four possible states from 2 – bit up-counting I.e. 00, 01, 10 and 11.

- The counter is initially assumed to be at a state 00 where the outputs of the two flip-flops are noted as Q_1Q_0 . Where Q_1 forms the MSB and Q_0 forms the LSB.
- For the negative edge of the first clock pulse, output of the first flip-flop FF_1 toggles its state. Thus Q_1 remains at 0 and Q_0 toggles to 1 and the counter state are now read as 01.
- During the next negative edge of the input clock pulse FF_1 toggles and $Q_0 = 0$. The output Q_0 being a clock signal for the second flip-flop FF_2 and the present transition acts as a negative edge for FF_2 thus toggles its state $Q_1 = 1$. The counter state is now read as 10.
- For the next negative edge of the input clock to FF_1 output Q_0 toggles to 1. But this transition from 0 to 1 being a positive edge for FF_2 output Q_1 remains at 1. The counter state is now read as 11.
- For the next negative edge of the input clock, Q_0 toggles to 0. This transition from 1 to 0 acts as a negative edge clock for FF_2 and its output Q_1 toggles to 0. Thus the starting state 00 is attained. Figure shown below





Two-bit ripple down-counter using negative edge triggered flip flop:



A 2-bit down-counter counts in the order 0,3,2,1,0,1.....,i.e. 00,11,10,01,00,11etc. the above fig. shows ripple down counter, using negative edge triggered J-K FFs and its timing diagram.

- For down counting, Q1' of FF1 is connected to the clock of Ff2. Let initially all the FF1 toggles, so, Q1 goes from a 0 to a 1 and Q1' goes from a 1 to a 0.

- The negative-going signal at Q_1' is applied to the clock input of FF2, toggles FF2 and, therefore, Q_2 goes from a 0 to a 1. so, after one clock pulse $Q_2=1$ and $Q_1=1$, I.e., the state of the counter is 11.
- At the negative-going edge of the second clock pulse, Q_1 changes from a 1 to a 0 and Q_1' from a 0 to a 1.
- This positive-going signal at Q_1' does not affect FF2 and, therefore, Q_2 remains at a 1. Hence, the state of the counter after second clock pulse is 10
- At the negative going edge of the third clock pulse, FF1 toggles. So Q_1 , goes from a 0 to a 1 and Q_1' from 1 to 0. This negative going signal at Q_1' toggles FF2 and, so, Q_2 changes from 1 to 0, hence, the state of the counter after the third clock pulse is 01.
- At the negative going edge of the fourth clock pulse, FF1 toggles. So Q_1 , goes from a 1 to a 0 and Q_1' from 0 to 1. . This positive going signal at Q_1' does not affect FF2 and, so, Q_2 remains at 0, hence, the state of the counter after the fourth clock pulse is 00.

Two-bit ripple up-down counter using negative edge triggered flip flop:

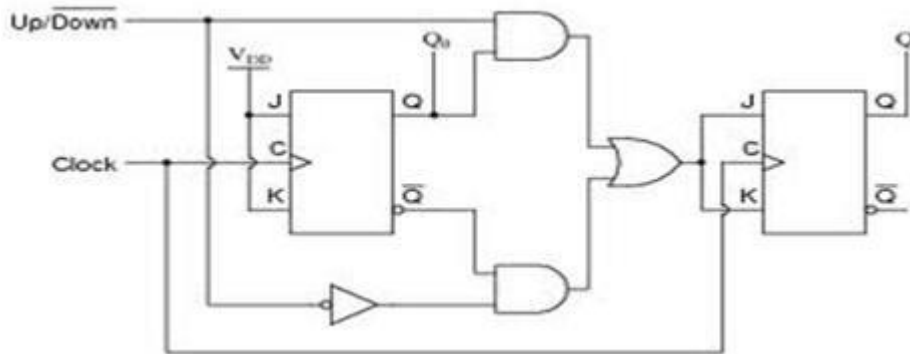


Figure: asynchronous 2-bit ripple up-down counter using negative edge triggered flip flop:

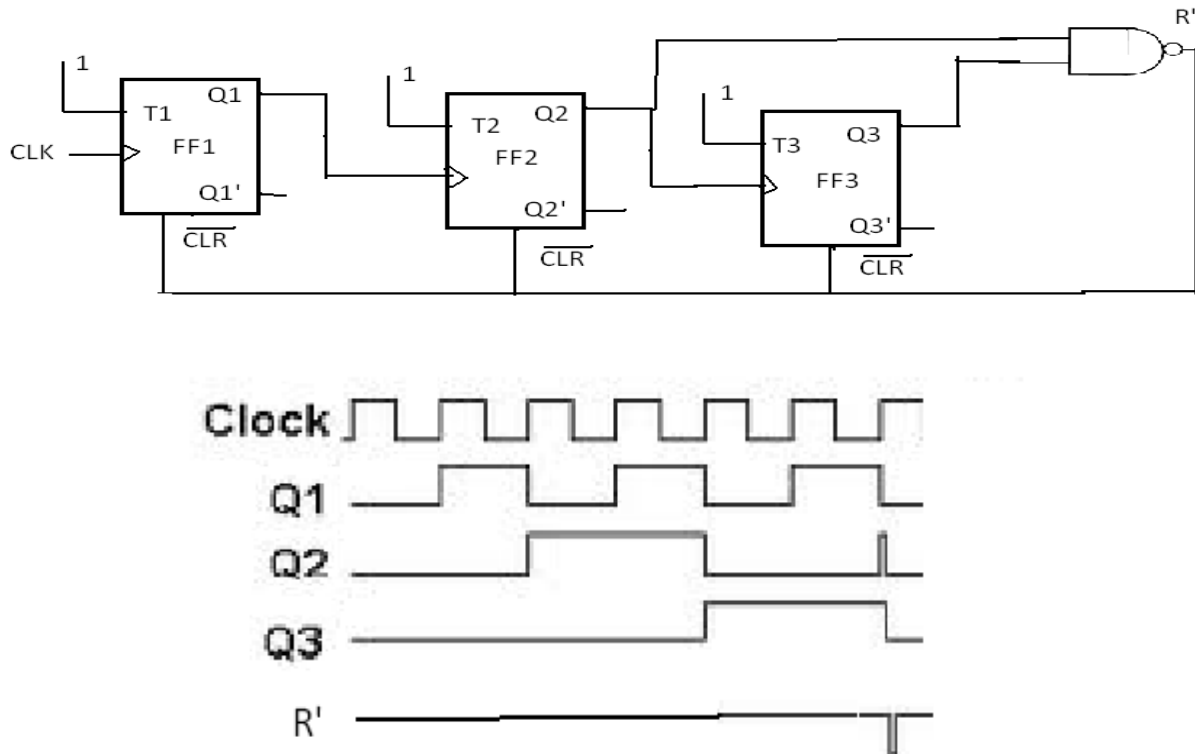
- As the name indicates an up-down counter is a counter which can count both in upward and downward directions. An up-down counter is also called a forward/backward counter or a bidirectional counter. So, a control signal or a mode signal M is required to choose the direction of count. When $M=1$ for up counting, Q_1 is transmitted to clock of FF2 and when $M=0$ for down counting, Q_1' is transmitted to clock of FF2. This is achieved by using two AND gates and one OR gates. The external clock signal is applied to FF1.
- Clock signal to FF2 = $(Q_1 \cdot \text{Up}) + (Q_1' \cdot \text{Down}) = Q_1m + Q_1'M'$

Design of Asynchronous counters:

To design a asynchronous counter, first we write the sequence, then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R and R' using K-Map or any other method. Provide a feedback such that R and R' resets all the FF's after the desired count

Design of a Mod-6 asynchronous counter using T FFs:

A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided. It is a divide-by-6 counter, in the sense that it divides the input clock frequency by 6. It requires three FFs, because the smallest value of n satisfying the condition $N \leq 2^n$ is $n=3$; three FFs can have 8 possible states, out of which only six are utilized and the remaining two states 110 and 111, are invalid. If initially the counter is in 000 state, then after the sixth clock pulse, it goes to 001, after the second clock pulse, it goes to 010, and so on.



After sixth clock pulse it goes to 000. For the design, write the truth table with present state outputs Q3, Q2 and Q1 as the variables, and reset R as the output and obtain an expression for R in terms of Q3, Q2, and Q1 that decides the feedback into be provided. From the truth table, $R = Q_3Q_2$. For active-low Reset, R' is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.

$R=0$ for 000 to 101, $R=1$ for 110, and $R=X$ for 111

Therefore,

$$R = Q_3Q_2Q_1' + Q_3Q_2Q_1 = Q_3Q_2$$

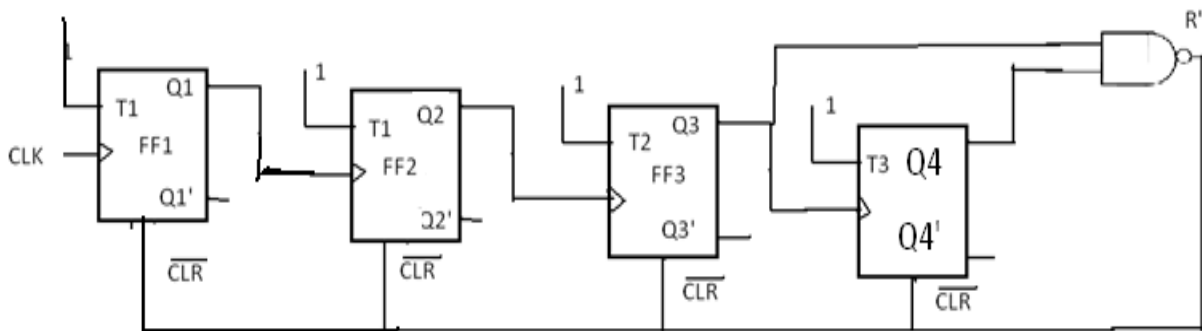
The logic diagram and timing diagram of Mod-6 counter is shown in the above fig.

The truth table is as shown in below.

After pulses	States			
	Q3	Q2	Q1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
	0	0	0	0
7	0	0	0	0

Design of a mod-10 asynchronous counter using T-flip-flops:

A mod-10 counter is a decade counter. It is also called a BCD counter or a divide-by-10 counter. It requires four flip-flops (condition $10 \leq 2^n$ is $n=4$). So, there are 16 possible states, out of which ten are valid and remaining six are invalid. The counter has ten stable states, 0000 through 1001, i.e., it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001. When the tenth clock pulse is applied, the counter goes to state 1010 temporarily, but because of the feedback provided, it resets to initial state 0000. So, there will be a glitch in the waveform of Q2. The state 1010 is a temporary state for which the reset signal $R=1$, $R=0$ for 0000 to 1001, and $R=C$ for 1011 to 1111.



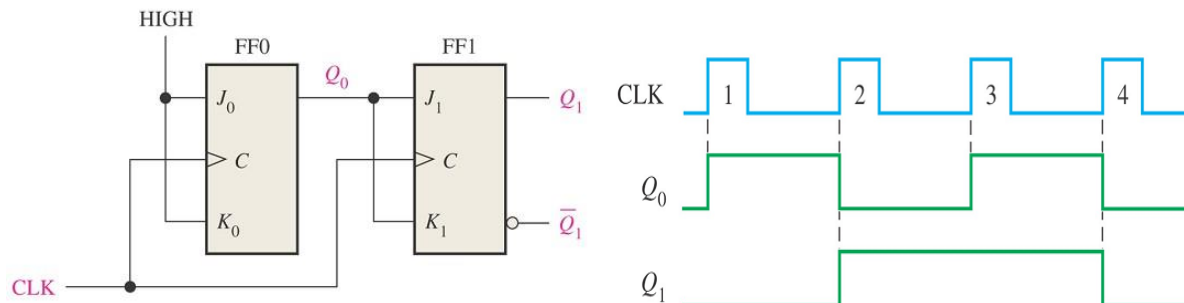
The count table and the K-Map for reset are shown in fig. from the K-Map $R=Q_4Q_2$. So, feedback is provided from second and fourth FFs. For active-HIGH reset, Q_4Q_2 is applied to the clear terminal. For active-LOW reset $\overline{Q_4Q_2}$ is connected to \overline{CLR} of all Flip-flops.

		Q2Q1			
		00	01	11	10
Q4Q3	00				
	01				
	11	X	X	X	X
	10		X	X	1

After pulses	Count			
	Q4	Q3	Q2	Q1
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	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	0	1	0	1
10	0	0	0	0

Synchronous counters:

Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. If the clock frequency is very high, the asynchronous counter may skip some of the states. This problem is overcome in synchronous counters or parallel counters. Synchronous counters are counters in which all the flip-flops are triggered simultaneously by the clock pulses. Synchronous counters have a common clock pulse applied simultaneously to all flip-flops. **A 2-Bit Synchronous Binary Counter**



Binary Counter:

The design of a synchronous binary counter is so simple that there is no need to go through a sequential logic design process. In a synchronous binary counter, the flip-flop in the least significant position is complemented with every pulse. A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1.

For example, if the present state of a four-bit counter is $A_3A_2A_1A_0 = 0011$, the next count is 0100. A_0 is always complemented. A_1 is complemented because the present state of $A_0 = 1$. A_2 is complemented because the present state of $A_1A_0 = 11$. However, A_3 is not complemented, because the present state of $A_2A_1A_0 = 011$, which does not give an all 1's condition.

Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flops and gates. The regular pattern can be seen from the four-bit counter depicted in Fig.. The C inputs of all flip-flops are connected to a common clock. The counter is enabled by Count_enable. If the enable input is 0, all J and K inputs are equal to 0 and the clock does not change the state of the counter. The first stage, A_0 , has its J and K equal to 1 if the counter is enabled. The other J and K inputs are equal to 1 if all previous least significant stages are equal to 1 and the count is enabled.

The chain of AND gates generates the required logic for the J and K inputs in each stage. The counter can be extended to any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flop outputs are 1.

[Note that the flip-flops trigger on the positive edge of the clock. The polarity of the clock is not essential here, but it is with the ripple counter. The synchronous counter can be triggered with either the positive or negative edge of the clock.]

the negative clock edge. The complementing flip-flops in a binary counter can be of either the JK type, the T type, or the D type with XOR gates. The equivalency of the three types is indicated in previous fig.

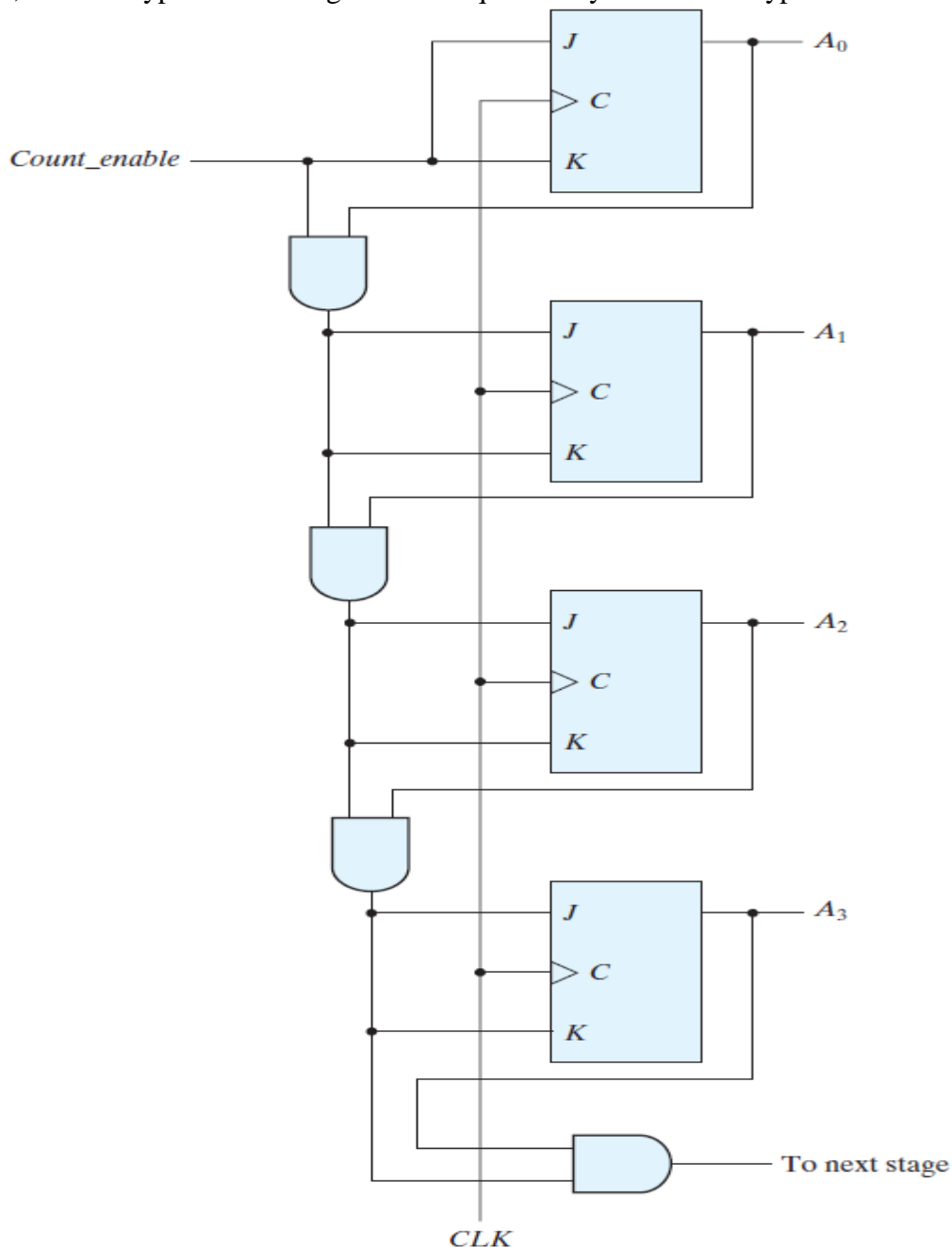


Fig: 4-bit binary counter

Up-Down Binary Counter:

A synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 and back to 1111 to repeat the count. It is possible to design a countdown counter in the usual manner, but the result is predictable by inspection of the downward binary count. The bit in the least significant position is complemented with each pulse. A bit in any other position is complemented if all lower significant bits are equal to 0.

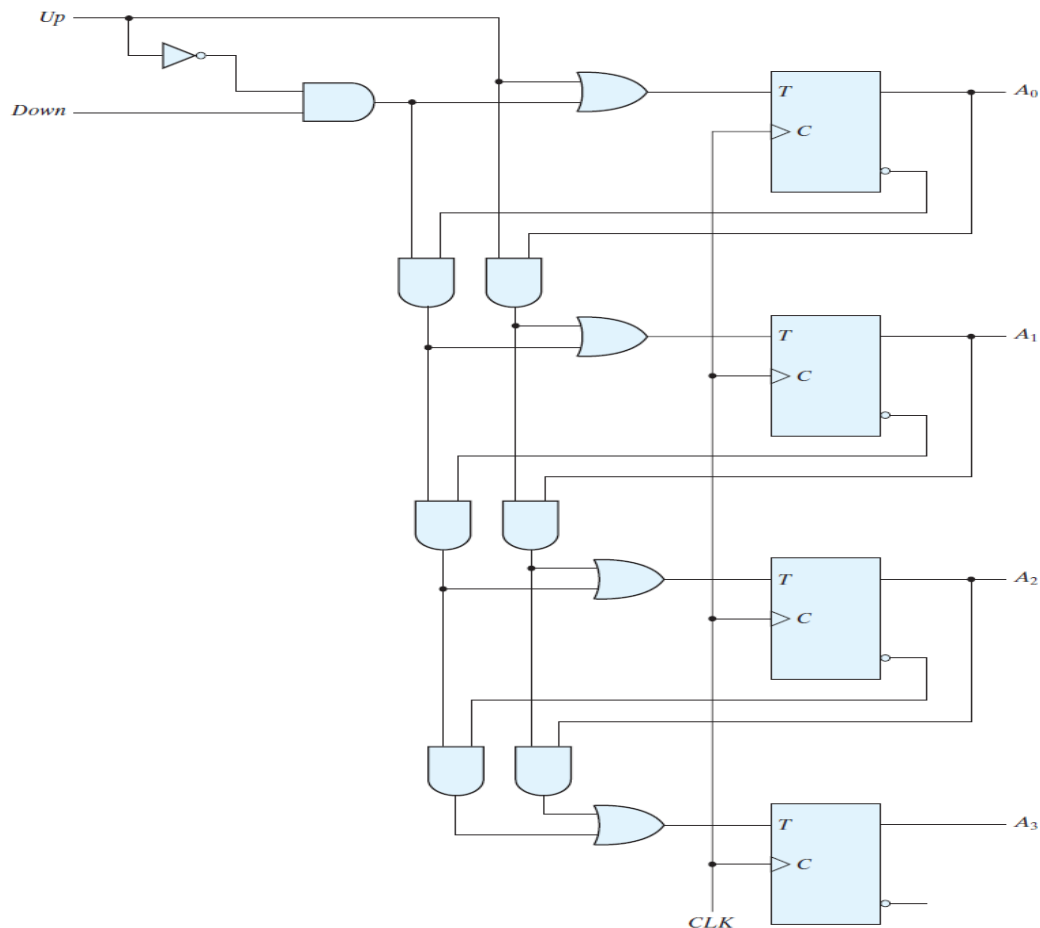
For example, the next state after the present state of 0100 is 0011. The least significant bit is always complemented. The second significant bit is complemented because the first bit is 0. The third significant bit is complemented because the first two bits are equal to 0.

But the fourth bit does not change, because not all lower significant bits are equal to 0.

A countdown binary counter can be constructed as shown in Fig., except that the inputs to the AND gates must come from the complemented outputs, instead of the normal outputs, of the previous flip-flops. The two operations can be combined in one circuit to form a counter capable of counting either up or down.

The circuit of an up-down binary counter using T flip-flops is shown in Fig. It has an up control input and a down control input. When the up input is 1, the circuit counts up, since the T inputs receive their signals from the values of the previous normal outputs of the flip-flops. When the down input is 1 and the up input is 0, the circuit counts down, since the complemented outputs of the previous flip-flops are applied to the T inputs.

When the up and down inputs are both 0, the circuit does not change state and remains in the same count. When the up and down inputs are both 1, the circuit counts up. This set of conditions ensures that only one operation is performed at any given time. [Note that the up input has priority over the down input.]



4-Bit up-Down Binary Counter

Design of synchronous counters:

For a systematic design of synchronous counters. The following procedure is used.

Step 1:State Diagram: draw the state diagram showing all the possible states state diagram which also be called nth transition diagrams, is a graphical means of depicting the sequence of states through which the counter progresses.

Step2: number of flip-flops: based on the description of the problem, determine the required number n of the flip-flops- the smallest value of n is such that the number of states $N \leq 2^n$ --- and the desired counting sequence.

Step3: choice of flip-flops excitation table: select the type of flip-flop to be used and write the excitation table. An excitation table is a table that lists the present state (ps) , the next state(ns) and required excitations.

Step4: minimal expressions for excitations: obtain the minimal expressions for the excitations of the FF using K-maps drawn for the excitation of the flip-flops in terms of the present states and inputs.

Step5: logic diagram: draw a logic diagram based on the minimal expressions

Design of a synchronous 3-bit up-down counter using JK flip-flops:

Step1: determine the number of flip-flops required. A 3-bit counter requires three FFs. It has 8 states (000,001,010,011,101,110,111) and all the states are valid. Hence no don't cares. For selecting up and down modes, a control or mode signal M is required. When the mode signal M=1 and counts down when M=0. The clock signal is applied to all the FFs simultaneously.

Step2: draw the state diagrams: the state diagram of the 3-bit up-down counter is drawn as

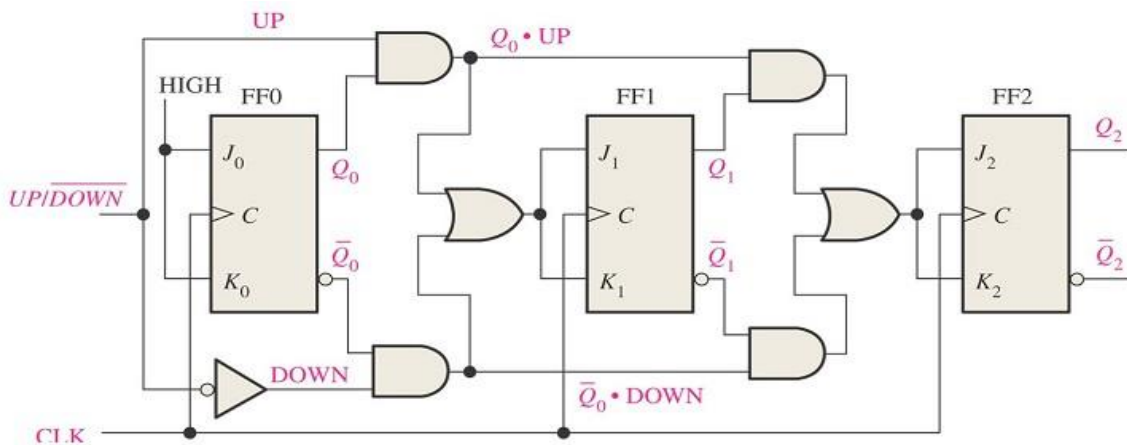
Step3: select the type of flip flop and draw the excitation table: JK flip-flops are selected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in fig.

PS			mode	NS			required excitations					
Q3	Q2	Q1	M	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	1	1	1	1	x	1	x	1	x
0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	0	0	0	x	0	x	x	1
0	0	1	1	0	1	0	0	x	1	x	x	1
0	1	0	0	0	0	1	0	x	x	1	1	x
0	1	0	1	0	1	1	0	x	x	0	1	x
0	1	1	0	0	1	0	0	x	x	0	x	1
0	1	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	1	1	x	1	1	x	1	x
1	0	0	1	1	0	1	x	0	0	x	1	x
1	0	1	0	1	0	0	x	0	0	x	x	1
1	0	1	1	1	1	0	x	0	1	x	x	1
1	1	0	0	1	0	1	x	0	x	1	1	x
1	1	0	1	1	1	1	x	0	x	0	1	x
1	1	1	0	1	1	0	x	0	x	0	x	1
1	1	1	1	0	0	0	x	1	x	1	x	1

Step4: obtain the minimal expressions: From the excitation table we can conclude that J1=1 and K1=1, because all the entries for J1 and K1 are either X or 1. The K-maps for J3, K3, J2 and K2 based on the excitation table and the minimal expression obtained from them are shown in fig.

	00	01	11	10
Q3Q2	Q1M			
1				
		1		
X	X	X	X	X
X	X	X	X	X

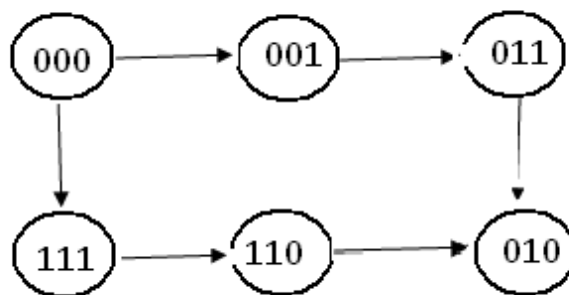
Step5: draw the logic diagram: a logic diagram using those minimal expressions can be drawn as shown in fig.



Design of a synchronous modulo-6 gray cod counter:

Step 1: the number of flip-flops: we know that the counting sequence for a modulo-6 gray code counter is 000, 001, 011, 010, 110, and 111. It requires $n=3$ FFs ($N \leq 2^n$, i.e., $6 \leq 2^3$). 3 FFs can have 8 states. So the remaining two states 101 and 100 are invalid. The entries for excitation corresponding to invalid states are don't cares.

Step2: the state diagram: the state diagram of the mod-6 gray code converter is drawn as shown in fig.



Step3: type of flip-flop and the excitation table: T flip-flops are selected and the excitation table of the mod-6 gray code counter using T-flip-flops is written as shown in fig.

Design of a synchronous BCD Up-Down counter using FFs:

Step1: the number of flip-flops: a BCD counter is a mod-10 counter has 10 states (0000 through 1001) and so it requires $n=4\text{FFs}(N \leq 2^n, \text{ i.e., } 10 \leq 2^4)$. 4 FFS can have 16 states. So out of 16 states, six states (1010 through 1111) are invalid. For selecting up and down mode, a control or mode signal M is required. , it counts up when $M=1$ and counts down when $M=0$. The clock signal is applied to all FFs.

Step2: the state diagram: The state diagram of the mod-10 up-down counter is drawn as shown in fig.

Step3: types of flip-flops and excitation table: T flip-flops are selected and the excitation table of the modulo-10 up down counter using T flip-flops is drawn as shown in fig.

The remaining minterms are don't cares ($\sum d(20,21,22,23,24,25,26,27,28,29,30,31)$) from the excitation table we can see that $T1=1$ and the expression for $T4, T3, T2$ are as follows.

$$T4 = \sum m(0,15,16,19) + d(20,21,22,23,24,25,26,27,28,29,30,31)$$

$$T3 = \sum m(7,15,16,8) + d(20,21,22,23,24,25,26,27,28,29,30,31)$$

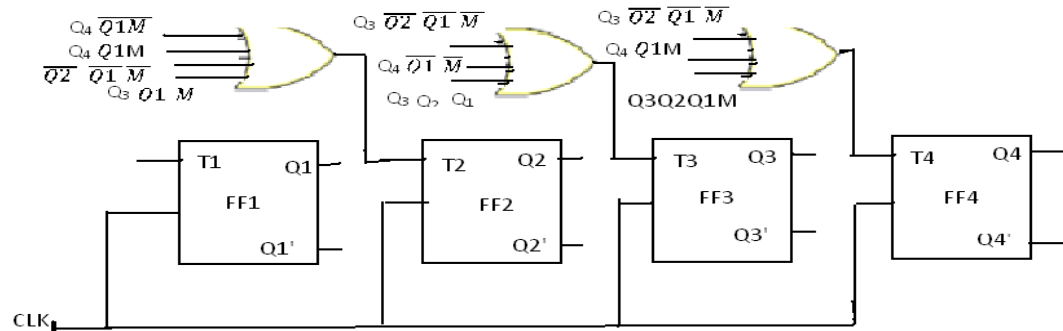
$$T2 = \sum m(3,4,7,8,11,12,15,16) + d(20,21,22,23,24,25,26,27,28,29,30,31)$$

PS				mode	NS				required excitations			
Q4	Q3	Q2	Q1		M	Q4	Q3	Q2	Q1	T4	T3	T2
0	0	0	0	0	1	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	0	1	0	0	1	1
0	0	1	0	1	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1	1	0	0	0	1
0	1	1	1	0	0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	0	0	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	1	0	0	1

Step4: The minimal expression: since there are 4 state variables and a mode signal, we require 5 variable kmaps. 20 conditions of $Q_4Q_3Q_2Q_1M$ are valid and the remaining 12 combinations are invalid. So the entries for excitations corresponding to those invalid combinations are don't cares. Minimizing K-maps for T2 we get

$$T_2 = Q_4Q_1'M + Q_4'Q_1M + Q_2Q_1'M' + Q_3Q_1'M'$$

Step5: the logic diagram: the logic diagram based on the above equation is shown in fig.



State diagram: the state diagram or state graph is a pictorial representation of the relationships between the present state, the input, the next state, and the output of a sequential circuit. The state diagram is a pictorial representation of the behavior of a sequential circuit.

The state represented by a circle also called the node or vertex and the transition between states is indicated by directed lines connecting circle. a directed line connecting a circle with itself indicates that the next state is the same as the present state. The binary number inside each circle identifies the state represented by the circle. The direct lines are labeled with two binary numbers separated by a symbol. The input value is applied during the present state is labeled after the symbol.

Shift register counters:

One of the applications of shift register is that they can be arranged to form several types of counters. The most widely used shift register counter is ring counter as well as the twisted ring counter.

Ring counter: this is the simplest shift register counter. The basic ring counter using D flip-flops is shown in fig. the realization of this counter using JK FFs. The Q output of each stage is connected to the D flip-flop connected back to the ring counter.

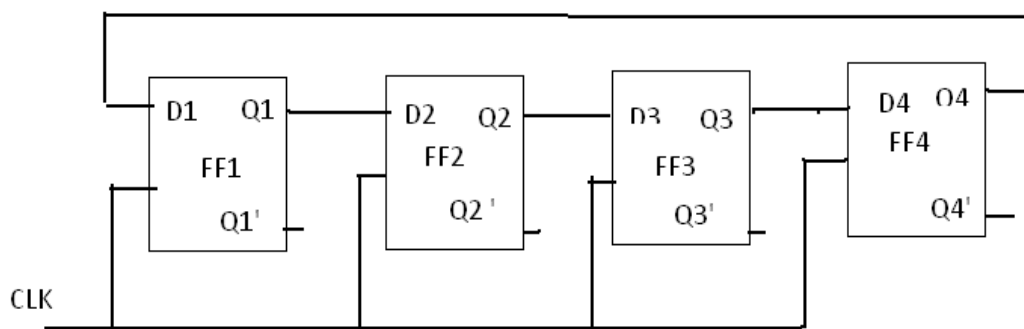


FIGURE: logic diagram of 4-bit ring counter using D flip-flops

Only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially the first FF is present to a 1. So, the initial state is 1000, i.e., $Q_1=1, Q_2=0, Q_3=0, Q_4=0$. After each clock pulse, the contents of the register are shifted to the right by one bit and Q_4 is shifted back to Q_1 . The sequence repeats after four clock pulses. The number of distinct states in the ring counter, i.e., the mod of the ring counter is equal to number of FFs used in the counter. An n-bit ring counter can count only n bits, whereas n-bit ripple counter can count 2^n bits. So, the ring counter is uneconomical compared to a ripple counter but has advantage of requiring no decoder, since we can read the count by simply noting which FF is set. Since it is entirely a synchronous operation and requires no gates external FFs, it has the further advantage of being very fast.

Timing diagram:

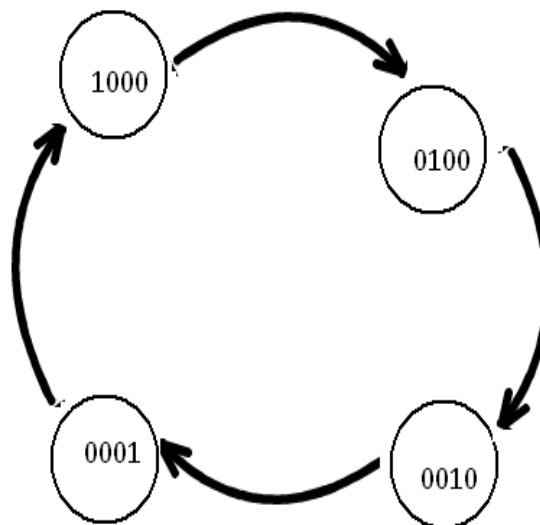
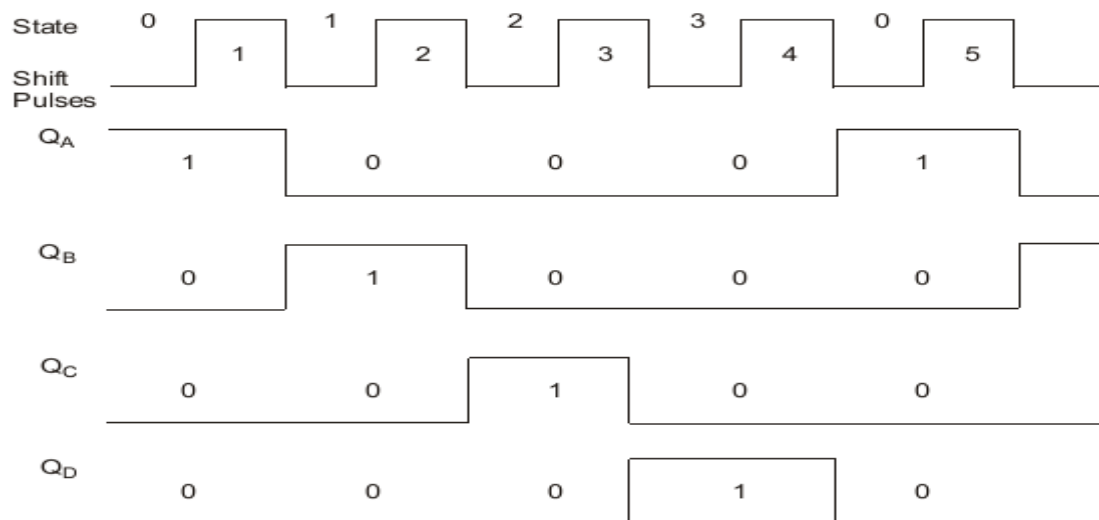


Figure: state diagram

Twisted Ring counter (Johnson counter):

This counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. the Q output of each is connected to the D input of the next stage, but the Q' output of the last stage is connected to the D input of the first stage, therefore, the name twisted ring counter. This feedback arrangement produces a unique sequence of states.

The logic diagram of a 4-bit Johnson counter using D FF is shown in fig. the realization of the same using J-K FFs is shown in fig.. The state diagram and the sequence table are shown in figure. The timing diagram of a Johnson counter is shown in figure.

Let initially all the FFs be reset, i.e., the state of the counter be 0000. After each clock pulse, the level of Q1 is shifted to Q2, the level of Q2 to Q3, Q3 to Q4 and the level of Q4' to Q1 and the sequences given in fig.

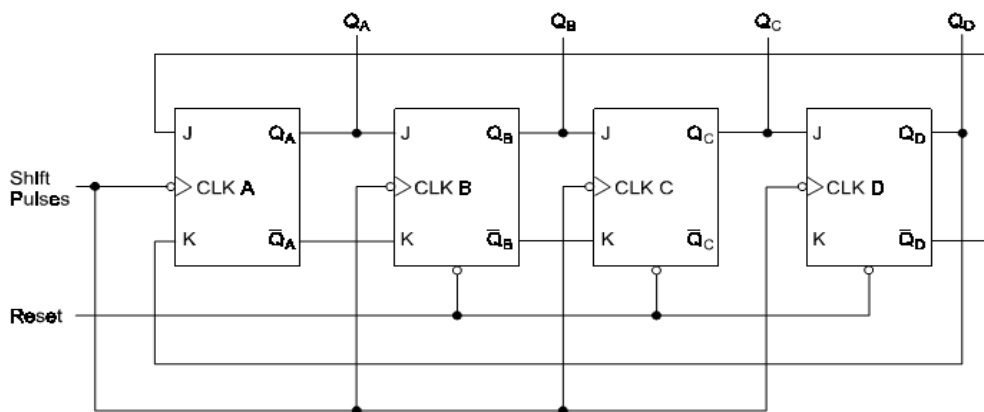


Figure: Johnson counter with JK flip-flops

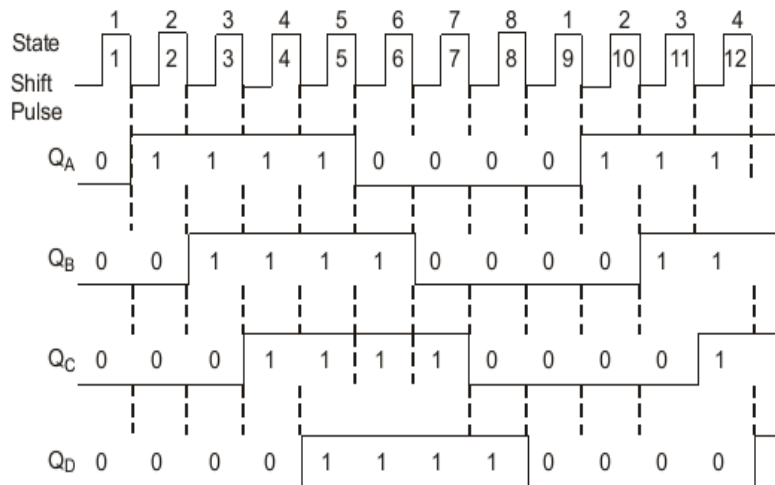
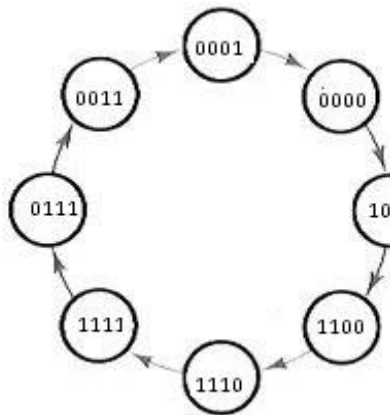


Figure: timing diagram

State diagram:



Q1	Q2	Q3	Q4
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0
1	0	0	0

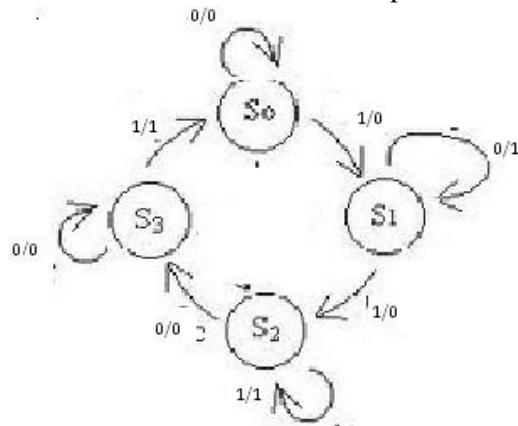
Excitation table

Synthesis of sequential circuits:

The synchronous or clocked sequential circuits are represented by two models.

1. Moore circuit: in this model, the output depends only on the present state of the flip-flops
2. Mealy circuit: in this model, the output depends on both present state of the flip-flop. And the inputs.

Sequential circuits are also called finite state machines (FSMs). This name is due to the fact that the functional behavior of these circuits can be represented using a finite number of states



NS,O/P		
INPUT X		
PS	X=0	X=1
a	a,0	b,0
b	b,1	c,0
c	d,0	c,1
d	d,0	a,1

Fig :a) state diagram (mealy circuit)

fig: b) state table

In case of moore circuit ,the directed lines are labeled with only one binary number representing the input that causes the state transition. The output is indicated with in the circle below the present state, because the output depends only on the present state and not on the input.

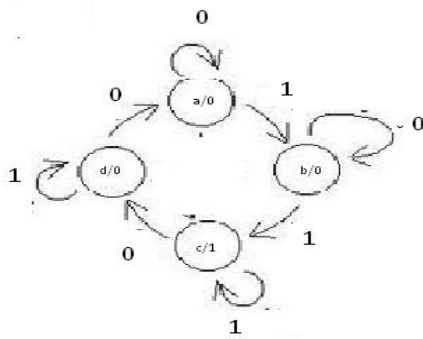


Fig: a) state diagram (moore circuit)

NS			
INPUT X			
PS	X=0	X=1	O/P
a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	0

fig:b) state table

Serial binary adder:

Step1: word statement of the problem: the block diagram of a serial binary adder is shown in fig. it is a synchronous circuit with two input terminals designated X_1 and X_2 which carry the two binary numbers to be added and one output terminal Z which represents the sum. The inputs and outputs consist of fixed-length sequences 0s and 1s. the output of the serial Z_i at time t_i is a function of the inputs $X_1(t_i)$ and $X_2(t_i)$ at that time t_{i-1} and of carry which had been generated at t_{i-1} . The carry which represent the past history of the serial adder may be a 0 or 1. The circuit has two states. If one state indicates that carry from the previous addition is a 0, the other state indicates that the carry from the previous addition is a 1

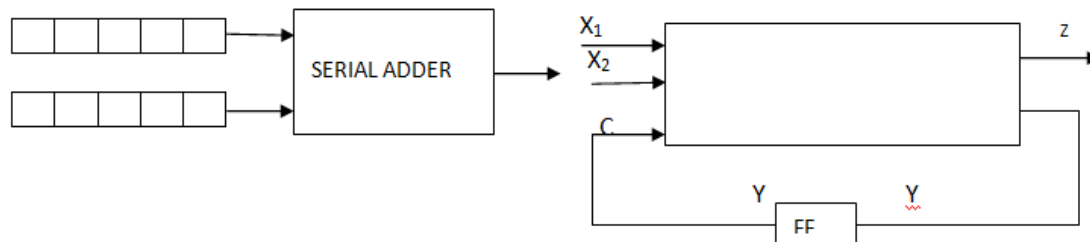
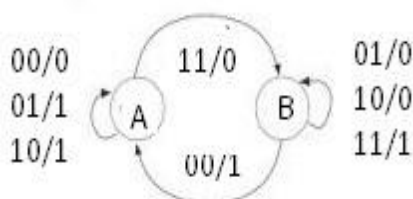


Figure: block diagram of serial binary adder

Step2 and 3: state diagram and state table: let a designate the state of the serial adder at t_i if a carry 0 was generated at t_{i-1} , and let b designate the state of the serial adder at t_i if carry 1 was generated at t_{i-1} . the state of the adder at that time when the present inputs are applied is referred to as the present state(PS) and the state to which the adder goes as a result of the new carry value is referred to as next state(NS).

The behavior of serial adder may be described by the state diagram and state table.



PS	NS ,O/P	
	X1	X2
	0	0
	0	1
A	A,0	B,0
B	A,1	B,0

Figures: serial adder state diagram and state table

If the machine is in state B, i.e., carry from the previous addition is a 1, inputs $X_1=0$ and $X_2=1$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=0$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=1$ gives sum, 1 and carry 0. So the machine remains in state B and outputs a 1. Inputs $X_1=0$ and $X_2=0$ gives sum, 1 and carry 0. So the machine goes to state A and outputs a 1. The state table also gives the same information.

Setp4: reduced standard from state table: the machine is already in this form. So no need to do anything

Step5: state assignment and transition and output table:

The states, A=0 and B=1 have already been assigned. So, the transition and output table is as shown.

PS	NS				O/P			
	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1
0	0	0	0	1	0	1	1	1
1	0	1	1	1	1	0	0	1

STEP6: choose type of FF and excitation table: to write table, select the memory element the excitation table is as shown in fig.

PS	I/P		NS	I/P-FF	O/P
y	x1	x2	Y	D	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Step 7. K-maps and minimal expressions: Obtain the minimal expressions for D and z in terms of the state variable y and inputs x_1 , and x_2 by using K-maps as shown in Figure 6.162.

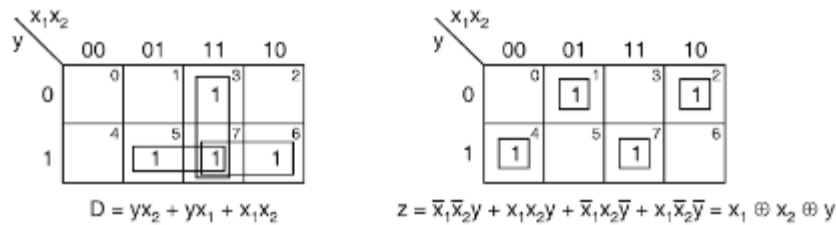


Figure 6.162 K-maps for the serial adder.

Step 8. Implementation: Implement the circuit using those minimal expressions as shown in Figure 6.163.

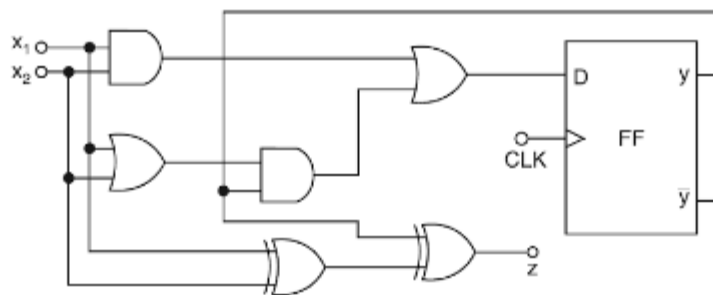


Figure 6.163 Logic diagram of the serial binary adder.

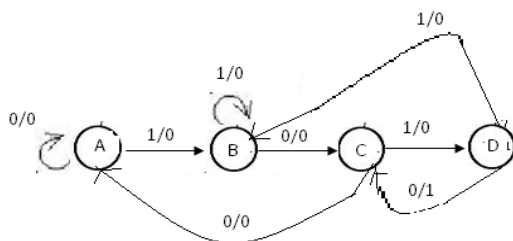
Sequence detector:

Mealy type circuit:

Step1: word statement of the problem: a sequence detector is a sequential machine which produces an output 1 every time the desired sequence is detected and an output 0 at all other times

Suppose we want to design a sequence detector to detect the sequence 1010 and say that overlapping is permitted i.e., for example, if the input sequence is 01101010 the corresponding output sequence is 00000101.

Step2 and 3: state diagram and state table: the state diagram and the state table of the sequence detector. At the time t_1 , the machine is assumed to be in the initial state designed arbitrarily as A. while in this state, the machine can receive first bit input, either a 0 or a 1. If the input bit is 0, the machine does not start the detection process because the first bit in the desired sequence is a 1. If the input bit is a 1 the detection process starts.



PS	NS,Z	
	X=0	X=1
A	A,0	B,0
B	C,0	B,0
C	A,0	D,0
D	C,1	B,0

Figure: state diagram and state table of sequence detector

So, the machine goes to state B and outputs a 0. While in state B, the machinery may receive 0 or 1 bit. If the bit is 0, the machine goes to the next state, say state c, because the previous two bits are 10 which are a part of the valid sequence, and outputs 0.. if the bit is a 1, the two bits become 11 and this not a part of the valid sequence

Step4: reduced standard form state table: the machine is already in this form. So no need to do anything.

Step5: state assignment and transition and output table: there are four states therefore two states variables are required. Two state variables can have a maximum of four states, so, all states are utilized and thus there are no invalid states. Hence, there are no don't cares. Let a=00, B=01, C=10 and D=11 be the state assignment.

PS(y1y2)	NS(Y1Y2)				O/P(z)	
	X=0		X=1		X=0	X=1
A= 0 0	0	0	0	1	0	0
B=0 1	1	0	0	1	0	0
C=1 0	0	0	1	1	0	0
D=1 1	1	1	0	1	1	0

Step6: choose type of flip-flops and form the excitation table: select the D flip-flops as memory elements and draw the excitation table.

					INPUTS -		
PS		I/P	NS		FFS		O/P
y1	Y2	X	Y1	Y2	D1	D2	Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0

Step7: K-maps and minimal functions: based on the contents of the excitation table , draw the k-map and simplify them to obtain the minimal expressions for D1 and D2 in terms of y1, y2 and x as shown in fig. The expression for z (z=y1,y2) can be obtained directly from table

Step8: implementation: the logic diagram based on these minimal expressions

Moore Type Circuit

For the design of Moore type of circuit, the same steps are to be followed. The state diagram and the state table of a Moore type sequence detector to detect the sequence 1010 are shown in Figure 6.168. In the Moore type state diagram the outputs are written inside the circle below the state name. The state diagram is drawn in the normal way. The machine will be in state D if the last three bits are 101. If the next bit is a 0, the last four bits will be 1010 which is a valid sequence. So the machine outputs a 1, but to utilize overlapping it cannot go to state C because the output at C is 0. Instead it goes to a new state E where output is equal to 1. While at E, if the next bit is a 0 the last five bits become 10100. So the machine goes to state A to restart the detection. If the next bit is a 1, the last five bits become 10101. So to utilize the last three bits, i.e. 101 it goes to state D. Once, the Moore type state diagram and state table are drawn, the Moore circuit can be designed following the normal procedure.

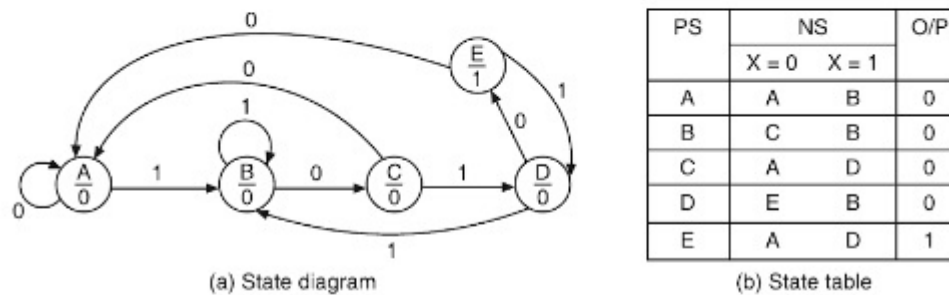


Figure 6.168 Moore circuit.