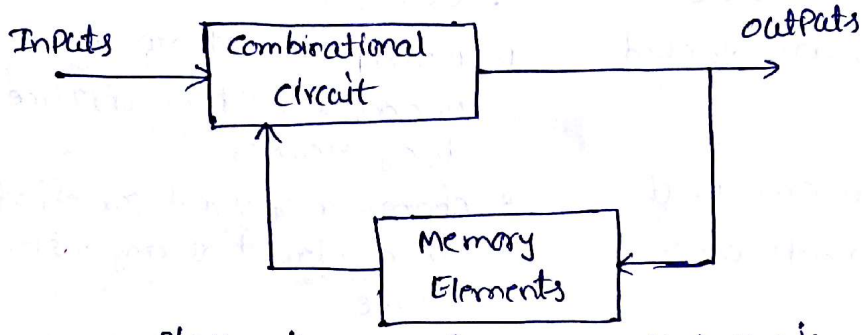
UNIT-V

SEQUENTIAL CIRCUITS-I

Analysis and design of combinational digital circuits is only a part of digital systems. The other major aspect of digital system is analysis and design of sequential circuits.



Block diagram of a sequential circuit.

In sequential circuits the output generated is not only dependent on present i/p conditions but also dependent upon past history of these i/p's.

The past history is provided by feedback from the o/p back to the i/p by using memory elements. The information stored in the memory element at any given time defines the present state of the sequential circuit.

Comparison between combinational & sequential circuits:—

combinational circuits	sequential circuits
1. o/p variables at all times dependent on the combination of i/p variables	1. o/p variables also dependent upon the past history of i/p variables.
2. memory unit is not required	2. Memory unit is required to store past history of the system.
3. Faster in speed. delay is only due to gates.	3. slower than combinational circuits.
4. Easy to design	4. Comparatively harder to design.
5. Parallel adder is a combinational circuit	5. Serial adder is a sequential circuit.

Classification of sequential circuits:—

The sequential circuits may be classified as synchronous sequential and Asynchronous sequential circuits depending on the timing of their signal. the sequential circuits which are controlled by a clock are called "synchronous sequential circuits". These circuits will be active only when clock signal is presented.

the sequential circuits which are not controlled by a clock are called "Asynchronous sequential circuits"

The memory elements used in both circuits are flip-flops which are capable of storing 1-bit binary information.

Comparision between Synchronous & Asynchronous Sequential circuits:—

Synchronous sequential Circuit	Asynchronous sequential Circuit
1. Memory elements are clocked flip-flops	1. memory elements are either unclocked flip flops (or) Time delay elements
2. change in i/p signal can effect memory elements upon activation of clock signal	2. change in i/p signal can effect memory element at any instant of Time
3. The max operating speed of clock depends on time delays involved	3. Because of absence of the clock, Asynchronous circuits can operate faster than synchronous circuits.
4. Easier to design	4. more difficult to design.

Latches and Flip-FLOPS:—

These are bistable elements and basic building blocks of most sequential circuits.

The main difference b/w latches and flip-flops is in the method used for changing their state.

Flip-Flop is a sequential device that normally samples its i/p's and changes its o/p's only at times determined by clocking signal.

Latch is a sequential device that checks all of its i/p's continuously and changes its o/p's accordingly at any time independent of clocking signal

Many times enable signal is provided with the latch. when enable signal is active o/p changes occur as i/p changes. But when enable signal is not activated i/p changes do not affect output.

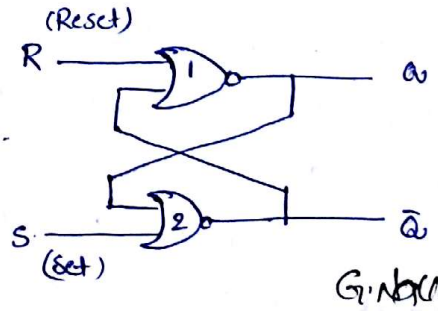
S-R latch:—

The simplest type of latch is the set-Reset (SR) latch. It can be constructed from either two NAND gates (or) two NOR gates.

S-R latch using NOR gates:—

SR latch is designed by using two NOR gates. The two NOR gates are cross coupled so that the o/p of NOR gate '1' is connected to one of the i/p of NOR gate 2 & vice-versa.

The latch has Q & $\bar{Q}$ as 2 o/p's. set & Reset as two i/p's.

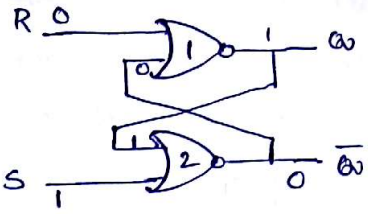


G. NAVEEN
NEC, EEE Dept

W.K.T logic '1' at any i/p of NOR gate forces its o/p to a logic '0'.

Operation of this circuit for various i/p/o/p possibilities:—

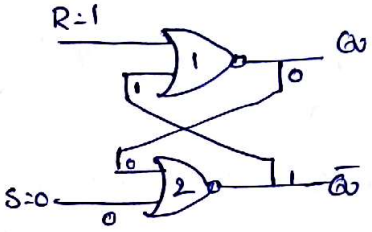
Case 1:— S=1 & R=0



S input of the NOR gate 2 is at logic 1, hence its o/p $\bar{Q}$ is at logic 0.

Both i/p's to NOR gate 1 are now at logic '0' so that its o/p Q is at logic '1'

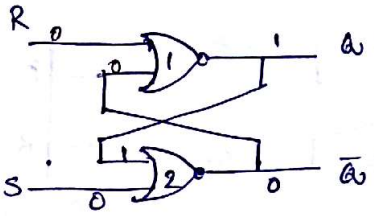
Case 2:— S=0 & R=1



R i/p of the NOR gate '1' is at logic '1'. hence its o/p Q is at logic 0.

Both i/p's to NOR gate 2 are now at logic '0'. so $\bar{Q}$ = 1

Case 3:— S=0 & R=0

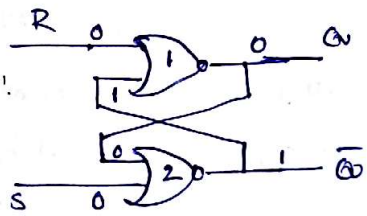


⇒ Initially Q=1 & $\bar{Q}$=0:—

with $\bar{Q}$=0, both i/p's to NOR gate '1' are at logic '0'. i.e Q=1.

when Q=1, input of NOR gate 2 is at logic 1. hence $\bar{Q}$=0. This shows that when S & R both are low the o/p state does not change.

⇒ Initially Q=0 & $\bar{Q}$=1:—



with $\bar{Q}$=1, one i/p of NOR gate 1 is at logic '1'. hence Q=0.

when Q=0, both i/p's to NOR gate 2 is at logic '0' hence $\bar{Q}$=1. In this case also there is no change in the o/p state.

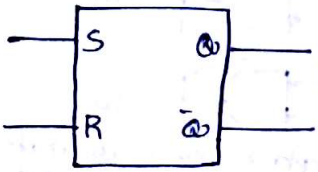
Case 4:— S=1 & R=1

When R & S both are at logic 1, they force the outputs of both NOR gates to the low state. (Q=0 & $\bar{Q}$=0). so this state is called Indeterminate or Prohibited state. and represent this condition in the truth table as an "asterisk" X'

this condition also violates the basic definition of a latch that requires Q to be the complement of $\bar{Q}$.

when 1's are applied to both the inputs then that condition must be avoided.

Symbol for SR latch:—



from the truth table we can summarize the operation of 'SR' latch as follows.

→ with both i/p's are low, o/p does not change this is called "inactive state' (NC)

→ R is low & S is high, the 'Q' o/p of latch is set (1)

→ R is high & S is low, the 'Q' o/p of latch is reset (0).

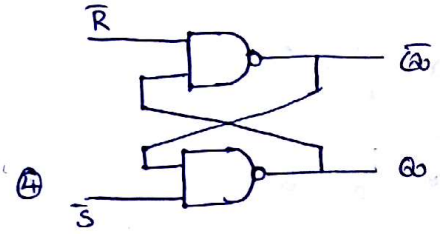
→ when both i/p's are high, o/p is unpredictable. This is called "Indeterminate condition".

Truth table:-

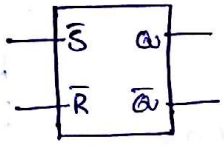
S	R	Q_n	Q_{n+1}	State
0	0	0	0	No change (NC)
0	0	1	1	
0	ϕ	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	Indeterminate
1	1	1	X	

$\bar{S}\bar{R}$ Latch:- (using NAND gates)

Fig shows Equivalent SR latch using two NAND gates.



④



Logic symbol.

$\bar{S}$	$\bar{R}$	Q_n	Q_{n+1}	State.
0	0	0	X	Indeterminate
0	0	1	X	
0	1	0	1	set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No change (NC)
1	1	1	1	

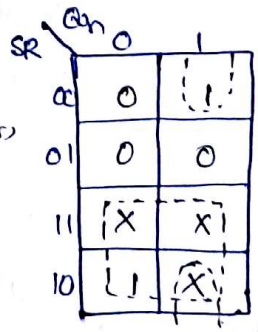
The Truth table for this latch is different from that for the NOR gate latch.

Here R & S i/p's are applied in complemented form.

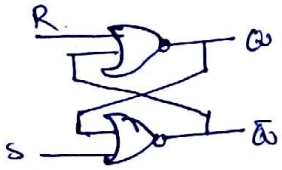
Low on any i/p to a NAND gate forces its o/p high. Thus the low on the $\bar{S}$ i/p will set the $\bar{R}\bar{S}$ latch. i.e $Q=1$, & $\bar{Q}=0$. A low on the $\bar{R}$ i/p will reset it. i.e $Q=0$ & $\bar{Q}=1$. If both $\bar{R}$ & $\bar{S}$ are high, the latch remain in it's previous state. when both are low the O/p is invalid.

Looking at Truth table for SR latch and simplifying Q_{n+1} function by K-Map we get the characteristic equation for SR latch as

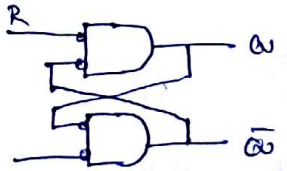
$$Q_{n+1} = S + \bar{R}Q_n$$



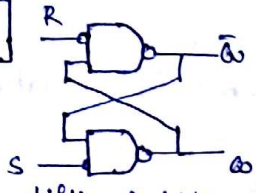
The functional behaviour of a latch (or) flip-flops can be described by the characteristic equation.



① Basic CKT



② Replacing NOR gates by bubbled AND gates



③ shifting bubble.

G. NAVEEN
NEC, EEE DEPT

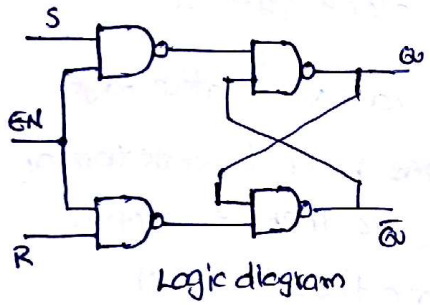
Gated Latches (Clocked-Flip-Flops):-

The Gated S-R Latch:-

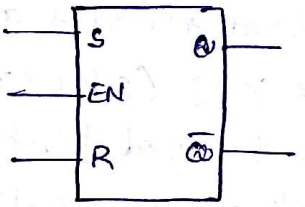
In the latches, the o/p can change state any time the i/p conditions are changed. So they are called "asynchronous latches". A gated S-R latch requires an ENABLE (EN) i/p. It's S & R i/p's will 'control' the state of the flip-flop only when the 'ENABLE is HIGH'.

When the ENABLE is "Low", the i/p's become 'ineffective' and no change of state can take place. The ENABLE i/p may be a clock. So a gated S-R latch is also called a "clocked S-R latch' (or) "synchronous S-R latch". Since this type of flip-flop responds to the changes in i/p's only as long as the "clock is HIGH"; these types of flip-flops are also called "level triggered flip-flops". The sequential circuits controlled by a clock are called "Synchronous sequential circuits"

The logic diagram, logic symbol & Truth table for S-R latch are shown in fig.



Logic diagram



Logic symbol

EN	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Indeterminate (Invalid)
1	1	1	1	X	
0	X	X	0	0	No change (NC)
0	X	X	1	1	

Truth Table.

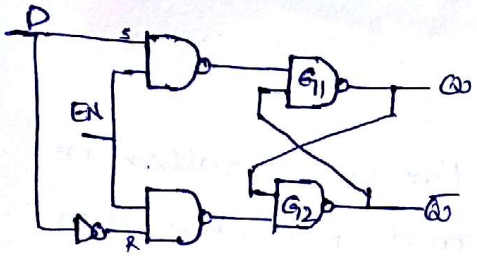
The Gated D-Latch:-

In many applications, it is not necessary to have separate S & R i/p's to a latch. If the i/p combinations S=R=0 & S=R=1, are never needed, the S & R are always the complement of each other. So we can construct a latch with a single i/p(S) and obtain the 'R' i/p by inverting it. This single i/p is labelled D (for data) and the device is called a "D-Latch". So another type of gated latch is the 'gated D-latch'. It has only one i/p in addition to EN. When D=1, we have S=1 and R=0, causing the latch to SET when ENABLED. When D=0, we have S=0 and R=1 causing the latch to RESET when ENABLED

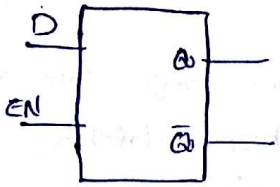
When EN is Low, the latch is ineffective, and any change in the value of D i/p does not affect the o/p at all.

When EN is HIGH, a Low D i/p makes Q Low i.e resets the flip-flop and a HIGH D i/p makes Q HIGH i.e sets the flip-flop.

the logic diagram, logic Symbol & Truth table of a gated D-Latch as shown in fig.



logic diagram.



Logic symbol.

EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	set
1	1	1	1	
0	X	0	0	No change (NC)
0	X	1	1	

Edge-Triggered Flip-Flops:- (Dynamic Triggering):-

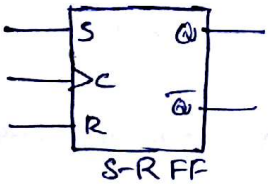
Digital systems can operate either synchronously (or) Asynchronously. In asynchronous systems, the o/p's of logic circuits can change state any time, when 1 (or more i/p changes. An asynchronous system is difficult to design and trouble shoot. In synchronous systems, the exact time at which any o/p can change states are determined by a signal commonly called the "clock". the flip-flops using the clock signal are called the "clocked flip-flops".

The clock signal is distributed to all parts of the system and most of the system outputs can change state only when the clock makes a transition.

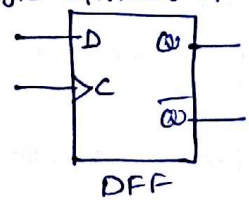
clocked flip-flops may be "positive edge-Triggered" (or) "negative edge-Triggered". positive edge-triggered flip-flops are those in which 'state transitions' take place only at the positive going (0 to 1, (or) Low to High) edge of the clock pulse, and negative edge-triggered flip flops are those in which 'state transitions take place only at the negative going (1 to 0. (or) High to Low) edge of clock signal.

positive-edge triggering is indicated by a 'triangle' at the clock Terminal of flip-flop. the Negative-edge triggering is indicated by a 'triangle' with a 'bubble' at the clock terminal of the clock. thes edge triggered flipflops are sensitive to their i/p's only at the transition of clock. there are 3 types of edge triggered flip-flops: S-R, J-K & D. D & J-K flip-flops are the most widely used ones

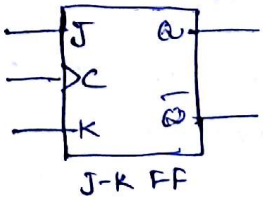
Logic symbols of positive edge-triggered FF's



S-R FF

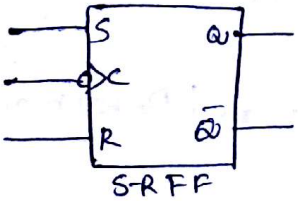


DFF

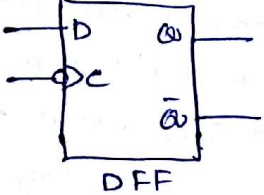


J-K FF

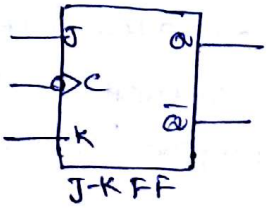
Logic symbols of negative edge-triggered FF's



S-R FF



DFF



J-K FF

G. NAVEEN
NEC, EEE DEPT

The Edge-triggered S-R Flip-Flop's :- (SR Flip-Flop)

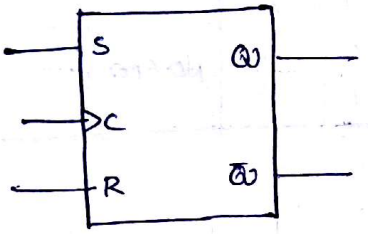
The S&R i/p's of the S-R flip-flop are called the "synchronous control i/p's" because data on these i/p's affect the flip-flop o/p only on the triggering edge of the clock pulse. without a clock pulse the S-R i/p cannot affect the o/p

→ When S is High & R is Low, the Q o/p goes high and flip-flop set (if already in set state remains set)

→ when S is Low & R is high the o/p goes Low and flip-flop is Reset (If already in Reset state, it remains RESET) i.e cleared

→ when both S&R are low, the o/p does not change from its prior state

→ when both S & R are High. an invalid condition exists.

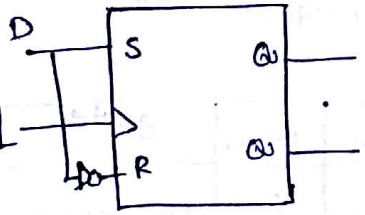


Logic symbol.

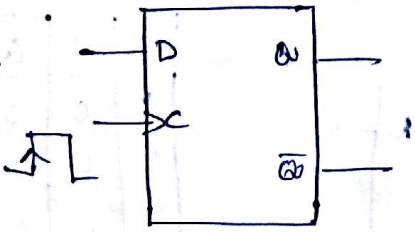
C	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No change (NC)
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	X	Indeterminate
↑	1	1	1	X	
0	X	X	0	0	No change (NC)
0	X	X	1	1	

The Edge-triggered D Flip-Flop :- (D Flip-Flop) :-

It has one i/p terminal. It is obtained from an S-R Flip Flop by putting one inverter b/w the S&R terminals. the i/p is called D(data) i/p. The o/p Q will go to the same state that is present on the D i/p. i.e if D is a '1' and the clock is applied, Q goes to 1 and $\bar{Q}$ to a 0. if D is zero (0). and the clock is applied. Q goes to a 0 & $\bar{Q}$ to a 1



D-FF from the SR FF



Logic symbol

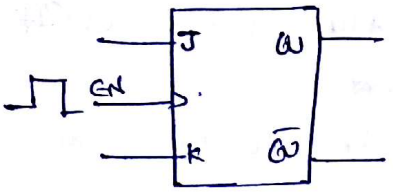
C	D	Q_n	Q_{n+1}	State
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
0	X	0	0	No change (NC)
0	X	1	1	

The edge-triggered J-K Flip-Flop or JK Flip-Flop:-

The JK FF is most widely used.

The functioning of the JK FF is identical to that of the SR FF. except that it has no invalid state - like S-R FF.

Logic symbol & Truth table for positive edge triggered JK FF. are shown in fig



C	J	K	Q_n	Q_{n+1}	state
↑	0	0	0	0	No change
↑	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	1	Toggle
↑	1	1	1	0	
0	X	X	0	0	No change.
0	X	X	1	1	

→ when J=0, K=0, No change of state takeplace. even if a clock pulse is applied

→ when J=0, K=1, the FF resets when the clock pulse is applied

→ when J=1 & K=0 the FF sets

→ when J=1, K=1 the FF toggles. ie goes to the OPPOSITE state when clock pulse is applied.

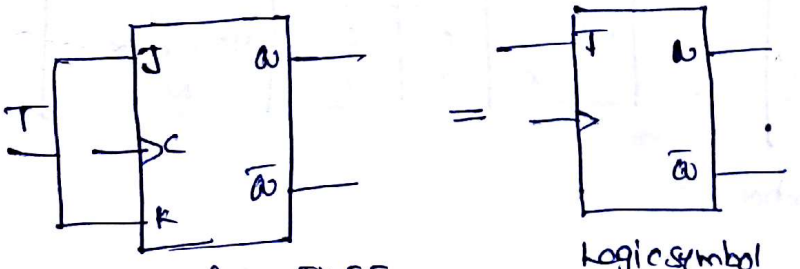
The operation of the 'tve' edge triggering & 've' edge triggering are are and the same except that the change of state takes place at the -ve going edge of clock pulse in -ve triggering. In truth table -ve edge triggering shown's in '↓' (arrow point downwards)

T-Flip-Flop:-

It has also a single control i/p. labelled 'T' for toggle, when T is High, the flip-flop toggles on every new clock pulse. when T is Low, the flip-flop remains in what ever state it was before.

JK FF can be easily convert into T, thus when T=1, we have J=K=1, & the FF toggles. when T=0, we have J=K=0, & there is no change of state.

Logic symbol & Truth Table are shown in Fig



T-FF from JK FF Logic symbol

C	T	Q_n	Q_{n+1}	State
↑	0	0	0	No change (NC)
↑	0	1	1	
↑	1	0	1	Toggle
↑	1	1	0	
0	X	0	0	No change (NC)
0	X	1	1	

G.NAVEEN
NEC, EEE Dept

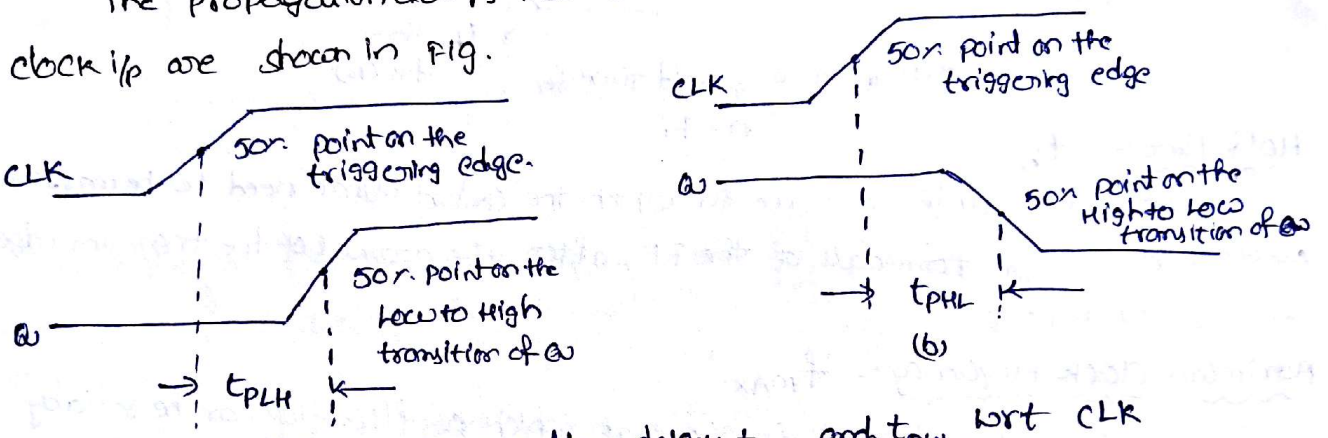
Flip-flop operating characteristics:-

FF specify several important characteristics and timing parameters that must be considered before a flip-flop is used in any circuit application.

Propagation delay Time:-

The time interval b/w the time of application of the triggering edge or a synchronous i/p's and the time at which the o/p actually makes a transition is called the "propagation delay Time". of the flip-flop. It is in the range of afew ns to 1 µs"

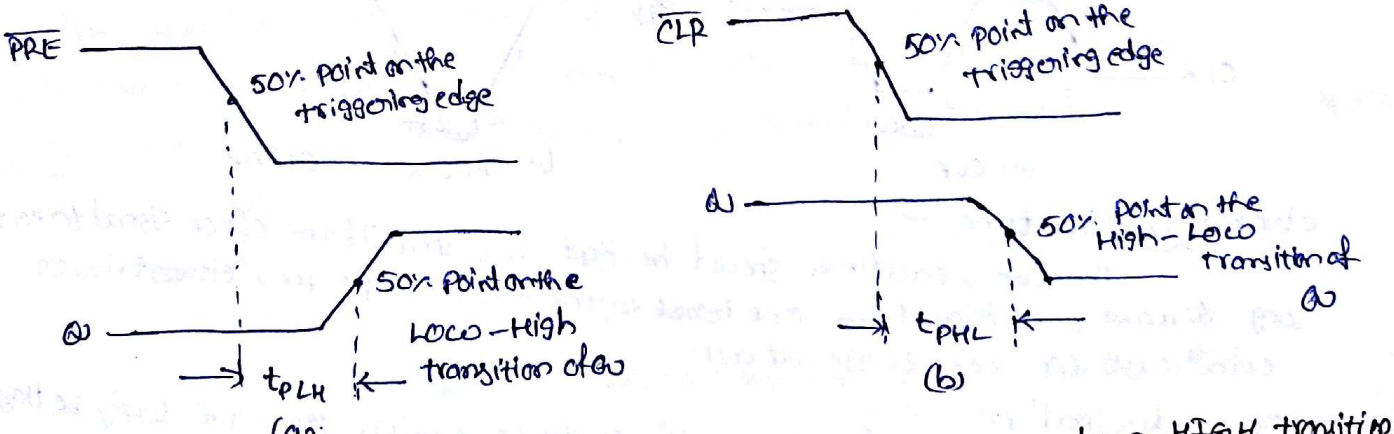
The propagation delays that occur in response to a positive transition on the clock i/p are shown in fig.



propagation delays t_{PLH} and t_{PHL} wrt CLK

1. Propagation delay t_{PLH} measured from the triggering of the clock pulse to the Low to HIGH transition of the o/p. (fig(a))
2. Propagation delay t_{PHL} measured from the triggering of the clock pulse to the HIGH to Low transition of the o/p (fig(b))

the propagation delays that occur in response to signals on a flip-flop's Asynchronous i/p's (PRESET & CLEAR) are shown in fig.



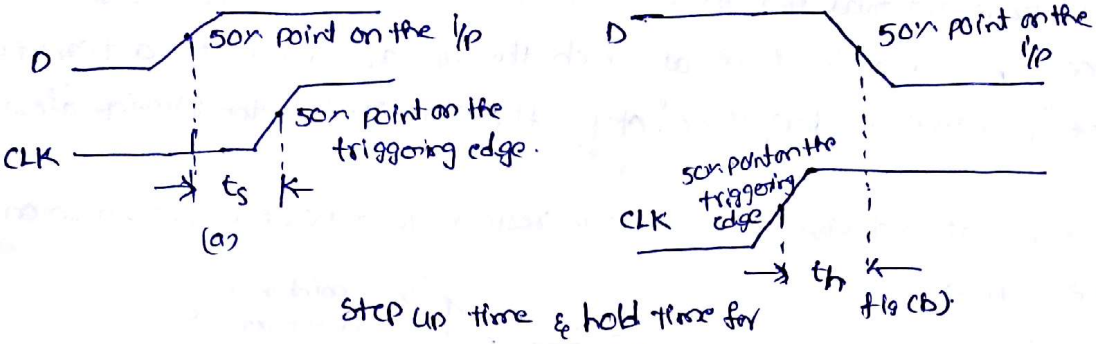
① Propagation delay t_{PLH} measured from PRESET i/p to the Low-HIGH transition of the o/p fig (a). this delay for active 'Low PRESET'

② t_{PHL} measured from the CLEAR i/p to the HIGH to Low transition of the o/p fig(b). This delay for active 'Low CLEAR"

Set up time:- 't_s'

It is the minimum time for which the "control levels" need to be maintained constant on the I/p terminals of the FF, prior to the arrival of the triggering edge of the clock pulse. in order to enable the FF to respond reliably.

Step up time for D FF is shown below fig (a)



Step up time & hold time for D-FF

Hold time:- 't_h'

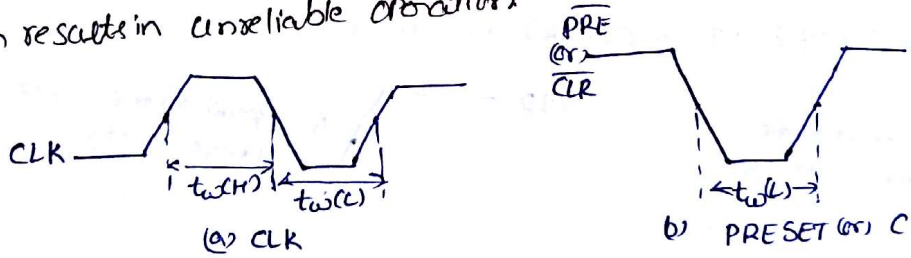
It is the minimum time for which the "control signal" need to be maintained constant at the I/p terminals of the FF, after the arrival of the triggering edge of the clock pulse.

Maximum clock Frequency:- 'f_{MAX}'

It is the highest frequency at which a flip-flop can be reliably triggered. the f_{MAX} limit will vary from one FF to another

Pulse widths:-

For clock signal, the minimum High time $t_w(H)$ & the minimum LOW time $t_w(L)$ are specified for asynchronous i/p's. ie PRESET & CLEAR. the minimum active state time is specified. Failure to meet these minimum time requirements can results in unreliable operation



clock transition times:-

the rise & fall times should be kept very short. If the clock signal takes too long to make transitions from one level to the other. the FF may either trigger erratically (or) not trigger at all.

Power dissipation:-

The total power consumption of device is equal. Product of supply voltage V_{cc} & current I_{cc}. $P = V_{cc} \cdot I_{cc}$ (mW)

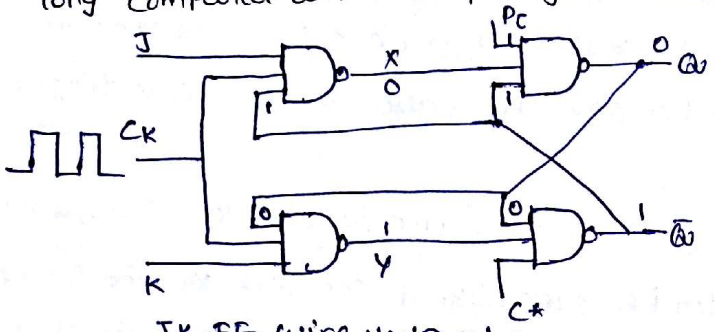
If a digital system has N FF's & if each FF dissipates P mW of power, the total power $P_{TOT} = N \cdot V_{cc} \cdot I_{cc} = (NP)$ mW

G. NAVEEN
NEC, EEE-Dept

Race Around Condition :—

Consider the excitations J=K=1. If the width of the clock pulse t_p is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0, 0 to 1 ... and at the end of the clock pulse, its state will be uncertain. This phenomenon is called "Race Around condition".

The o/p Q & $\bar{Q}$ will change on their own if the clock pulse width t_p is too long compared with the propagation delay τ of each NAND gate.



JK-FF using NAND gates

Time	X	Y	Q	$\bar{Q}$	
Initial $t<0$	1	1	0	1	Initial state assumed
$t>0$ $t=\tau$	0	1	0	1	Pulse is present for $t>\tau$
$t=2\tau$	0	1	1	1	
$t=3\tau$	0	0	1	0	
$t=4\tau$	1	0	1	1	
$t=5\tau$	0	0	0	1	
$t=6\tau$	0	1	1	1	
$t=7\tau$	0	0	1	0	
$t=8\tau$	1	0	1	1	

Flow of signals in race around condition.

From table Q & $\bar{Q}$ keep on changing with time if the clock pulse width t_p is greater than 'τ'.

Assuming that the clock pulse occurs at t=0, & $t_p \gg \tau$, the following expressions hold before and during t_p in the JK FF. Note that $f(t-\tau)$ is $f(t)$ delayed by τ.

During No pulse $X(t) = 1$, $Y(t) = 1$ → No change in Q & $\bar{Q}$

During the pulse of width t_p with J=K=1

$$X(t) = \overline{J \cdot C_K \cdot \bar{Q}(t-\tau)} = \overline{\bar{Q}(t-\tau)}, \quad Q(t) = \overline{X(t-\tau)\bar{Q}(t-\tau)}$$

$$Y(t) = \overline{K \cdot C_K \cdot Q(t-\tau)} = \overline{Q(t-\tau)}, \quad \bar{Q}(t) = \overline{Y(t-\tau)\cdot Q(t-\tau)}$$

From the table we observe that the change of state takes at least 2τ. If the flip-flop is initially at $Q\bar{Q}=01$, then it is easily seen that the transition will follow the sequence of logic levels for each τ as given as

$$Q\bar{Q}: 01 \to 01 \to 11 \to 10 \to 11 \to 01 \cdots$$

If $Q\bar{Q}=10$, the transitions path is

$$Q\bar{Q} = 10 \to 10 \to 11 \to 01 \to 11 \to 10 \cdots$$

We conclude that the state of 'FF' keeps on complementing it-self for every 2τ. The clock pulse width should be such as to allow only one change to complement the state and not too long to allow many changes resulting in uncertainty about the final state. This is a stringent requirement which cannot be in practice. This problem is eliminated using 'master-slave Flip Flop" or "edge triggered flip-flop".

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

Before development of edge-triggered flip-flops there is a timing problem with "no hold" requirements. these are often handled by a class of flip-flops called the "master-slave flip-flops".

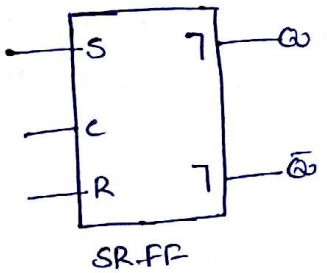
The master-slave flip-flops was developed to make the synchronous operation "more predictable". i-e to avoid the problems of logic race in clock flip-flops. This improvement is achieved by introducing a known "time delay" (equal to the width of one clock pulse) b/w the time that the 'FF' responds to a clock pulse and the time the response appears at its o/p. A master-slave FF is also called a "pulse-triggered flip-flop".

These FF's actually contains two flip-flops "a master Flip Flop" & a "slave FF". The "control i/p's" are applied to the master FF & maintained constant for the set-up time 't_s'. prior to the application of the clock pulse. on the rising edge of the clock-pulse, the levels on the control i/p's are used to determine the o/p of the master.

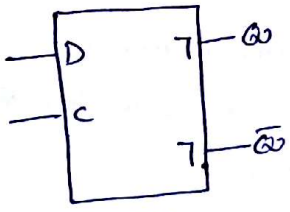
on the falling edge of the clock pulse, the state of master is transferred to the slave.

These M-S FF's function very much like "negative edge triggered FF's except for one major disadvantage. the control i/p's must be held stable while CLK is HIGH. otherwise an unpredictable operation may occur. This problem with the master-slave FF is over come with an improved M-S version called the "MS with data lock-out".

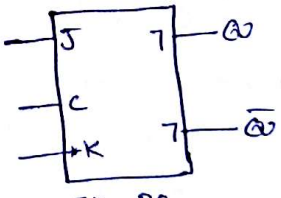
There are 3 basic types of MS FF's – S-R, D & J-K.



SR-FF

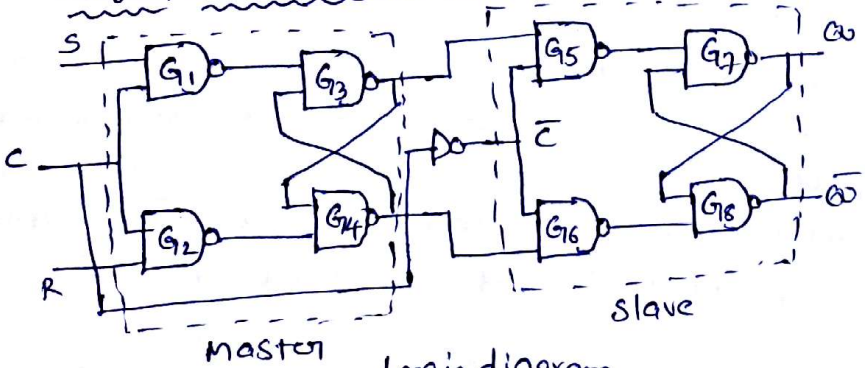


D-FF



JK-FF

Master-slave S-R Flip-Flop :-



Logic diagram

Inputs			O/P	Comments
S	R	CLK	Q	
0	0	⊓	Q_0	No change
0	1	⊓	0	RESET
1	0	⊓	1	SET
1	1	⊓	?	Invalid

G. NAVEEN
NEC, EEE Dept

The Truth table operation is the same as that for the edge-triggered S-R FF except for the way it is clocked.

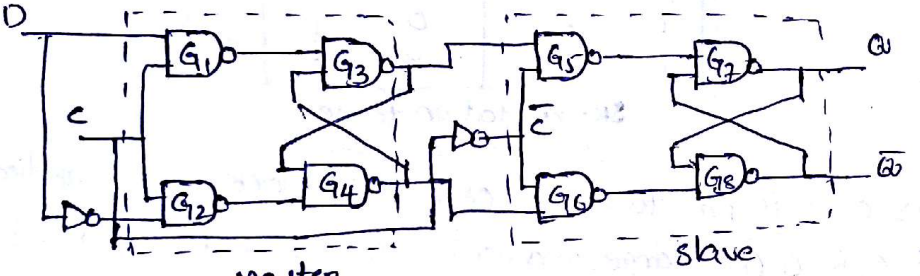
The external control i/p's S & R are applied to the master section. The master is basically a gated S-R latch and responds to the external S-R i/p's applied to it at the positive-going edge of the clock signal.

The slave section is the same as the master section except that it is clocked on the inverted clock pulse and thus responds to its control inputs at the negative going edge of the clock pulse.

Master-slave D Flip-Flop:—

The operation of the master-slave D-FF is the same as that of the negative edge-triggered 'D' FF except for the way it is triggered.

The D i/p is transferred to the master at the positive-going edge of the clock pulse and the same is copied by the slave and appears at the Q o/p of the slave at the negative-going edge of the clock pulse. The logic diagram & Truth table of master-slave D-flip-flop is shown below.

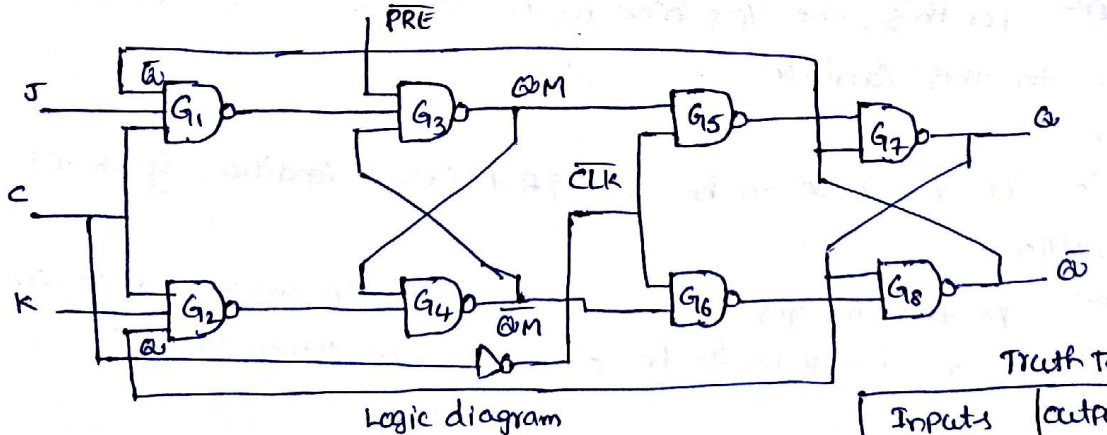


Logic diagram

Inputs		o/p	Comments
D	CLK	Q	
0	⊓	0	RESET
1	⊓	1	SET

Truth table.

Master-slave J-K Flip-Flop:—



Logic diagram

Truth Table

Inputs			output	comments
J	K	C	Q	
0	0	⊓	Q_0	No change
0	1	⊓	0	RESET
1	0	⊓	1	SET
1	1	⊓	$\bar{Q}_0$	Toggle

The operation is the same as that of a negative edge-triggered JK Flip-Flop except for the way in which it is triggered. The logic diagram of the master-slave J-K Flip Flop is similar to that of the master-slave SR flip-flop.

The difference is that the Q o/p is connected back to the i/p of G_2 & the $\bar{Q}$ o/p is connected

back to the input of G_1 and the external i/p's are designated as J & K. the M-S J-K flip flop is a J-K flip-flop followed by an S-R Flip-flop. i.e the master flip-flop is a J-K flip-flop and slave flip-flop is an S-R flip-flop. The problem of logic race is eliminated because while the clock drives the master, the inverted clock drives the slave.

Flip-Flop Excitation Tables:—

The design of sequential circuits we need excitation tables of flip-flops. the excitation table of a flip-flop can be obtained from its truth table. It indicates the i/p's required to be applied to the flip flop to take it from the "present state" (PS) to the "next state" (NS)

S-R flip-flop:—

The truth table and excitation table of an S-R Flip Flop are shown in table.

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

NC
Reset
Set
Invalid

S-R Truth table

PS	NS	Required i/p's	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR - Excitation table.

0→0 transition:—

If PS of FF is '0' & it has to remain '0' when a clock pulse is applied, the i/p can be either S=0, R=0 (No change condition) or, S=0, R=1 (reset condition) Thus 'S' has to be '0' R can be either 0 (or) 1 so SR = 0X for transition

0→1 transition:—

For this, the i/p's have to be S=1, & R=0 (set condition). so SR = 10 for this condition.

1→0 transition:—

For this, the i/p's have to be S=0 & R=1 (reset condition) so SR=01 for this transition.

1→1 transition:—

For this, the i/p's can be either S=0, R=0 (No change condition) (or) S=1, R=0 (set condition) Thus R has to be '0' but S can be either 0 (or) 1. so SR = X0 for this transition.

JK Flip-Flop:—

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

NC
Reset
Set
Invalid

Truth Table.

JK Excitation Table:

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

G. NAVEEN
NEC, EEE Dept

0→0 transition:-

For this transition, the i/p's either J=0, K=0 (No change condition) (or) J=0, K=1 (reset condition) thus J has to be 0 but K can be either 0 or 1. so JK = 0X for this transition.

0→1 transition:-

For this, the i/p's either J=1, K=0 (set condition) (or) J=1, K=1 (toggle condition) Thus J has to be 1 but K can be either 0 (or) 1. So JK = 1X for this transition.

1→0 transition:-

For this, the i/p's either J=0, K=1 (Reset condition) (or) J=1, K=1 (toggle condition) Thus J has to be 0 (or) 1 but K can be 1. so JK = X1 for this transition

1→1 transition:-

For this, the i/p's either J=0, K=0 (No change condition) (or) J=1, K=0 (set condition). thus J has to be 0 (or) 1 but K can be 0. So JK = X0 for this transition.

D Flip-Flop:-

D	Q_{n+1}
0	0
1	1

Truth Table

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

Excitation Table

set
reset
toggle

for a D Flip-Flop, the next state is always equal to the D i/p and it is independent of the present state. ∴ D must be 0 if Q_{n+1} has to be 0, & 1 if Q_{n+1} has to be 1 regardless of the value of Q_n.

T-Flip-Flop:-

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Truth Table

PS	NS	Required inputs
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

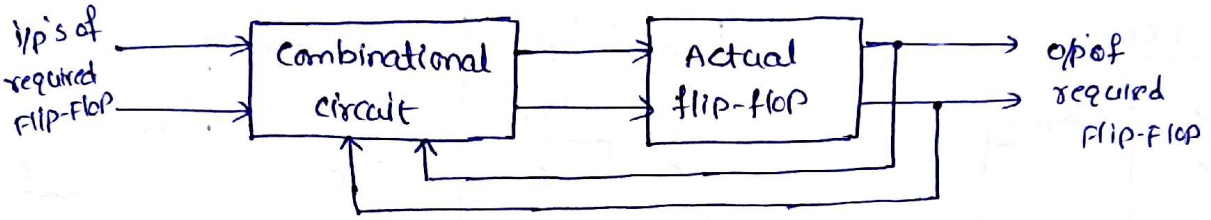
For a T Flip-Flop, when the i/p T=1, the state of the flip-flop is complemented and when T=0, the state of the flip-flop remains unchanged. Thus for 0→0 and 1→1 transitions T must be 0 and for 0→1 & 1→0 transition T must be '1'

Flip-Flop conversions:—

To convert one type of flip-flop into another type, a combinational circuit is designed such that if the i/p of the required Flip-Flop are fed as i/p's to the combinational circuit and the o/p of the combinational circuit is connected to the i/p's of the actual flip-flop, then the o/p of the actual flip-flop is the output of the required flip-flop.

(or)

To convert one type of flip-flop into another type, we have to obtain the expressions for the i/p's of the existing Flip-Flops interms of the inputs of the required flip-flop and the present state variables of the existing flip-flop and implement them as shown in fig.

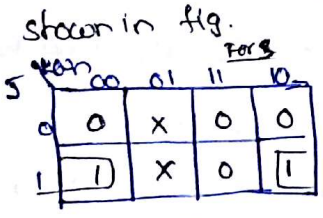


SR Flip-Flop to J-K Flip-Flop:—

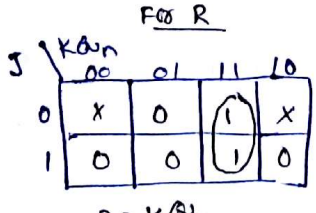
The External i/p's to the already available SR Flip-FLOP will be J & K. S & R are the o/p's of the combirational circuit. which are also the actual i/p's to the SR Flip FLOP. We write a truth table with J, K, Q_n, Q_{n+1}, S & R, where Q_n is present state & Q_{n+1} is the next state obtained when the particular J & K i/p's are applied. i.e Q_n denotes state of 'FF' before the application of the i/p's, & Q_{n+1} refers to the state obtained by the FF after the application of i/p's.

J, K & Q_n can have '8' combirations. for each combination of J, K & Q_n find the corresponding Q_{n+1}. i.e determine to next state (Q_{n+1}) the J-K FF will go from the present state Q_n if the present i/p's J & K are applied.

The conversion table, K maps for S & R, interms of J, K & Q_n & logic diagram are shown in fig.



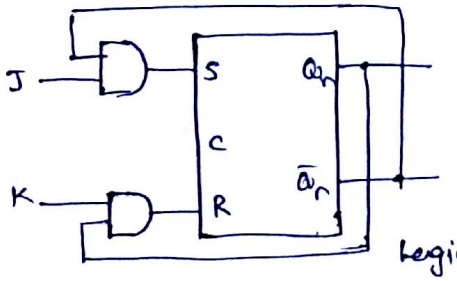
$S = J\bar{Q}_n$



$R = KQ_n$

External I/Ps		Present state	Next state	Flip-Flop I/Ps	
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

from the above table.



Logic diagram

G. NAVEEN
NEC, EEE Dept

JK Flip-Flop to S-R Flip-Flop:—

The External i/p's to the already available JK Flip-Flop will be S & R. J & K are the o/p's of the combinational circuit. which are also the actual i/p to the JK Flip-Flop. So, we have to get the values of J & K interms of S, R & Q_n. by using S, R & Q_n Can make 8 possible Combinations. In SR FF, the combination S=R=1, is not Permitted.

For J

S \ RQ_n	00	01	11	10
0	0	X	X	0
1	1	X	X	X

J = S

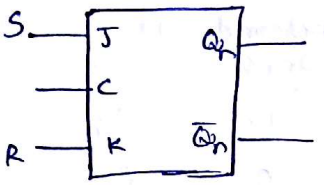
For K

S \ RQ_n	00	01	11	10
0	X	0	1	X
1	X	0	X	X

K = R.

External i/p's		Present state	Next state	Flip-Flop i/ps	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0

Conversion table



Logic diagram

S-R Flipflop to D Flip-Flop:—

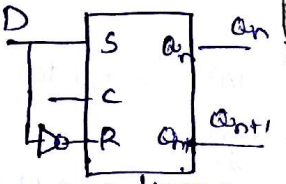
D is the External i/p and the o/p's of the Combinational circuit are the i/p to the available SR Flipflop.

D \ Q_n	0	1
0	0	0
1	1	1

S = D

D \ Q_n	0	1
0	X	1
1	0	0

R = $\bar{D}$



Logic diagram

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

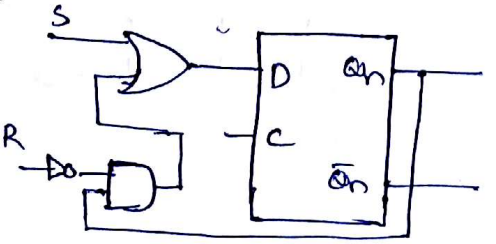
D to S-R Flip Flop:—

S & R are the external i/p's & D is the actual i/p to the existing Flip-Flop. S, R and Q_n make '8' Possible Combinations S=R=1, is invalid condition.

S \ RQ_n	00	01	11	10
0	0	1	0	0
1	1	1	X	X

D = S + $\bar{R}Q_n$.

Logic diagram

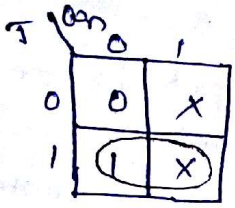


External Inputs		Present state	Next State	Flip-Flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1

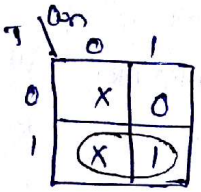
conversion table.

JK to T Flip-Flop:—

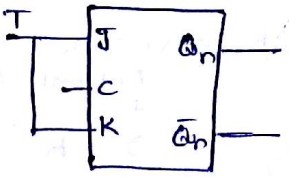
T is the external input & J, K are the actual i/p's to the existing flip-flop. T & Q_n make 4 Combinations.



$J = T$



$K = T$



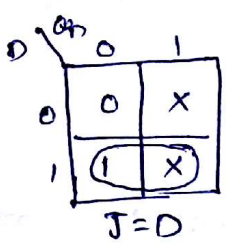
Logic diagram

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

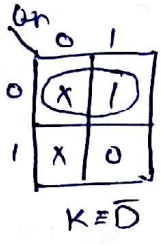
Conversion table:

JK to D Flip-Flop:-

D is the external i/p & J, K are the actual i/p to the existing FF. D & Q_n make 4 combinations.

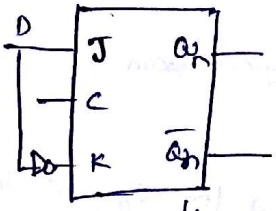


$J = D$



$K = \bar{D}$

K-maps.



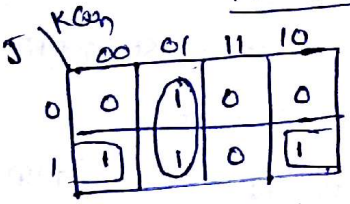
Logic diagram

External Input	Present state	Next state	Flip-Flop Inputs	
D	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

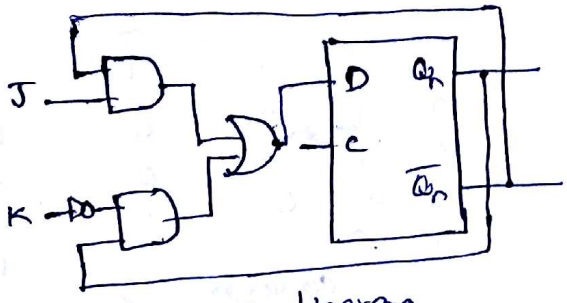
D FF to JK Flip-Flop:-

J, K are the external I/Ps, i.e i/p to combinational circuit and D is the actual input to the existing flip-flop.

K-map for D



$D = J\bar{Q}_n + \bar{K}Q_n$



Logic diagram

External Inputs		Present state	Next state	Flip-Flop Input
J	K	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
1	1	0	1	1
1	1	1	0	0

G. NAVEEN
NEC, EEE Dept

Applications of Flip-Flops:—

Basic applications of Flip-Flop's are Parallel data storage, serial data storage, transfer data, frequency division, counting, Parallel to serial conversion, Serial to Parallel conversion, synchronizing the effect of asynchronous data, detection of an i/p sequence.

Parallel Data storage:—

A group of flip-flops is called a register. To store data of N bits, N-FF's are required. If data is available in Parallel form. i.e all bits are present at a time, these bits may be made available at the D i/p terminal of the FF's. and when a clock pulse is applied to all the FF's simultaneously, these bits will be transferred to the Q o/p's of the FF's and FF's then store the data.

Serial data storage:—

To store a data of N bits available in serial form, N no. of D FF's are connected in cascade. the clock signal is connected to all the FF's. The serial data is applied to the D i/p terminal of the First FF.. each clock pulse transfers the D i/p to its Q o/p's. So, after N clock pulses the register contains the data & then store it..

Transfer of data:—

Data stored in FF's may be transferred out in a serial fashion. i.e bit-by-bit from the o/p of one FF (or) may be transferred out in Parallel form. ie all bits at a time from the Q o/p's of each of the FF's.

Serial to Parallel conversion:—

To convert the data available in serial form into parallel form, the serial data are first entered and stored in a Serial-in Parallel-out shift register and then, since the data is available at the o/p's of the FF's, the data may be taken out parallely. To convert an N-bit serial data to Parallel form, N-FF's are required. N clock pulses are required to enter the data in serial form and one clock pulse is required to shift the data out in Parallel form.

Parallel to serial conversion:—

To convert the data avilable in Parallel form into serial form, the parallel data are first entered in the parallel-in-serial out shift register in parallel form. ie all bits at a time and then data is shifted out of the register serially. i.e bit-by-bit by the application clock pulses. To convert an N-bit parallel data to serial form, N FF's are required. one clock pulse is required to shift the parallel data into the register & N clock pulses are required to shift the data out of the register serially.

Counting:—

A no. of FF's may be connected in a particular fashion to count the pulses electronically. One FF can count upto 2 pulses. 2 FF's can count upto $2^2 = 4$ pulses. In general N FF's can count upto 2^N pulses. In a simple counter, all the flip-flops are connected in toggle mode.

Frequency division:—

FF's may be used to divide the input signal frequency by any number. A single FF may be used to divide the i/p frequency by 2. 2 FF's may be used to divide the i/p frequency by 4. In general N FF's may be used to divide the i/p frequency by 2^N.

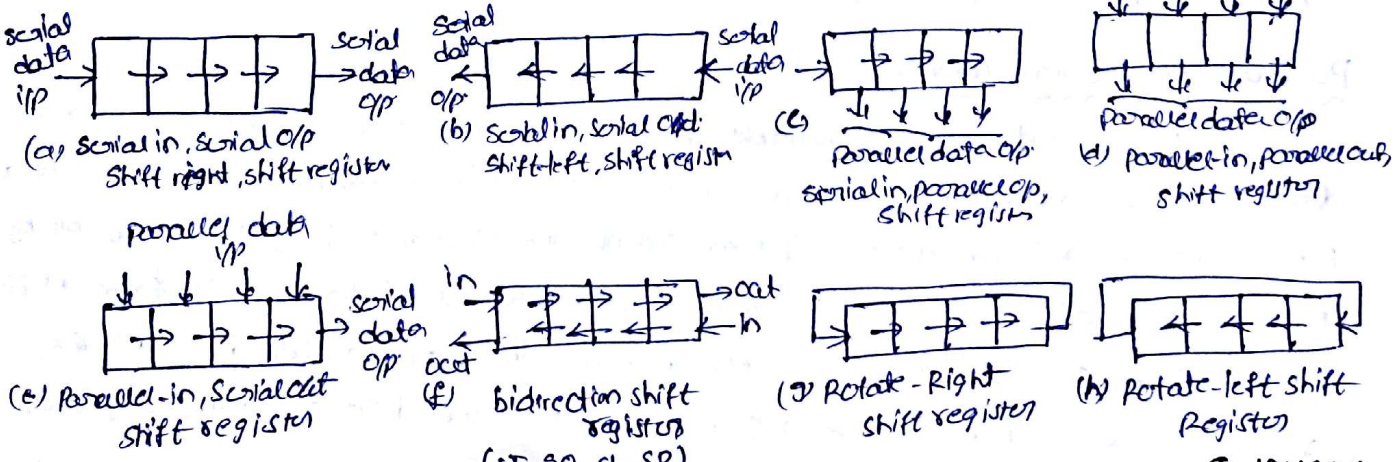
Shift Registers:—

shift registers are a type of logic circuits closely related to counters. They are used basically for the storage and transfer of digital data. The basic difference b/w a shift register and counter is that, a shift register has no specified sequence of states except in certain very specialized applications, whereas a counter has a specified sequence of states.

A shift register is a very important "digital building block". Registers are often used to momentarily store binary information appearing at the o/p of an "encoding matrix". Similarly, shift registers are often used to momentarily store binary data at the o/p of a "decoder".

Data Transmission in shift Registers:—

A No of FF's connected together such that data may be shifted into and shifted out of them is called "shift Register". Data may be shifted into (or) out of the register either in serial form (or) in parallel form. There are '4' basic types of shift registers. In all 4 types data may be rotated left to right. Data may be shifted from left to right or right to left at will. i.e in a bidirectional way. Also data may be shifted in serially (or) in parallel and shifted out serially (or) in parallel



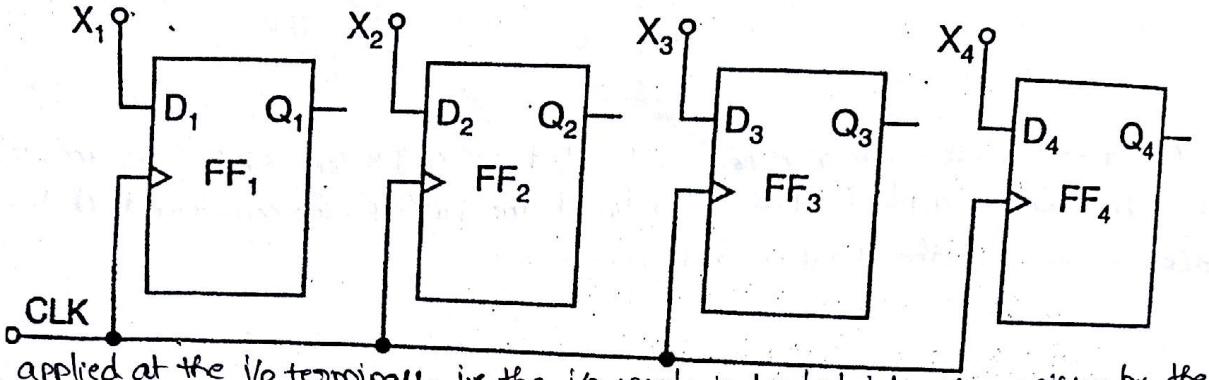
Data Transfer Registers.

G. NAVEEN
NEC, EEE Dept

Buffer Register:

The buffer register is the simplest of registers. It simply stores the binary word. The buffer may be a controlled buffer. most of the buffer registers use 'D'FF's

Fig 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 O/p word becomes the same as the -

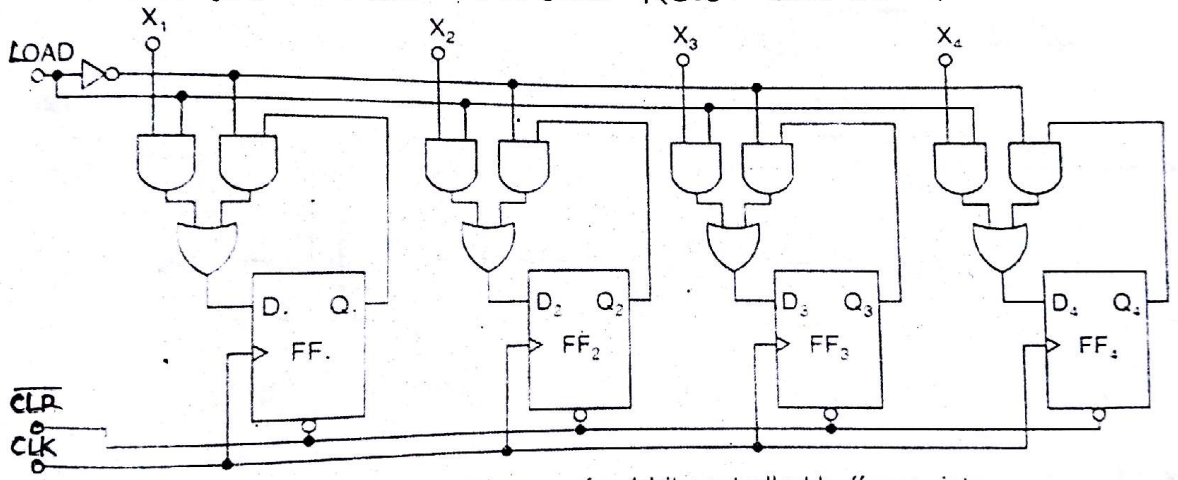


word applied at the I/p terminals. i.e the I/p word is loaded into the register by the application of clock pulse. when the positive clock edge arrives, the stored word becomes.

$Q_4Q_3\ Q_2Q_1 = X_4X_3X_2X_1$ (or) $Q = X$.

Controlled Buffer Register:

If $\overline{CLR}$ goes LOW, all the FF's are RESET and the o/p becomes, Q = 0000



Logic diagram of a 4-bit controlled buffer register.

→ when $\overline{CLR}$ is HIGH, the register is ready for action. LOAD is the control I/p. when LOAD is HIGH, the data bits 'x' can reach the D I/p's of FF's. At the positive-going edge of the next clock pulse, the register is loaded i.e $Q_4Q_3Q_2Q_1 = X_4X_3X_2X_1$, (or) $Q = X$.

→ when LOAD is LOW, the X bits cannot reach the FF's. At the same time, the inverted signal $\overline{LOAD}$ is HIGH. This forces each 'FF' o/p to feed back to its data I/p ∴ data is circulated (or) retained as each clock pulse arrives. In other words, the contents of the register remain unchanged in spite of the clock pulses. Longer buffer registers can be built by adding more FF's.

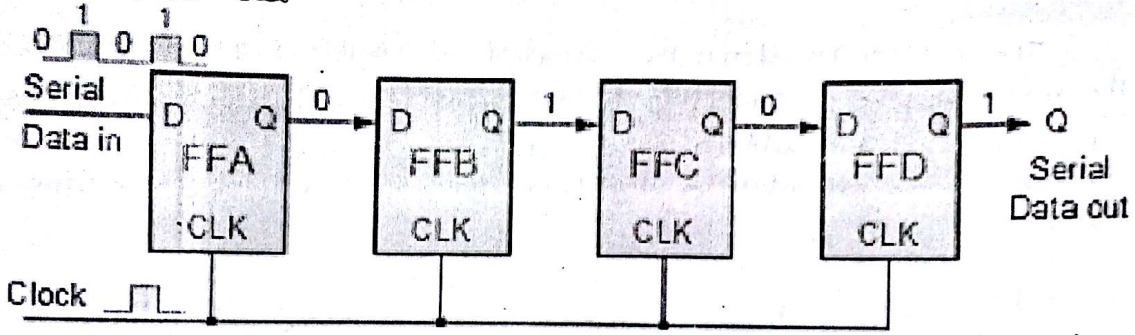
Serial-In, Serial-Out, Shift Register:

This type of shift register accepts data serially. i.e one bit at a time, and also o/p's data serially.

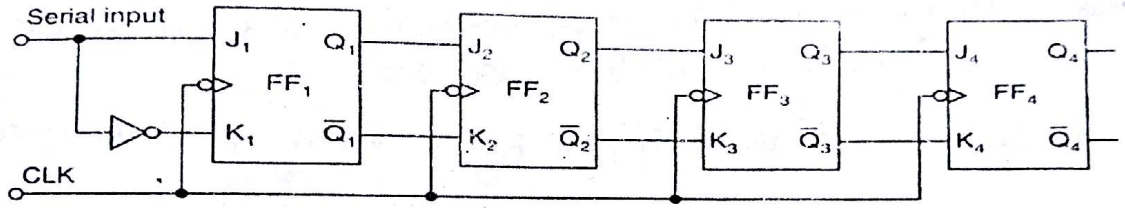
The logic diagram of a 4-bit serial-in, serial-out, shift-right, shift register is shown in fig's. with 4 stages i.e 4 FF's, the register can store upto 4 bits of data. serial data is applied at the D I/p of the first FF. The Q o/p of the first FF is connected to the D input of the second FF ....

When serial data is transfered into a register, each new bit is clocked into the first FF at the positive-going edge of each clock pulse. the bit that was previously

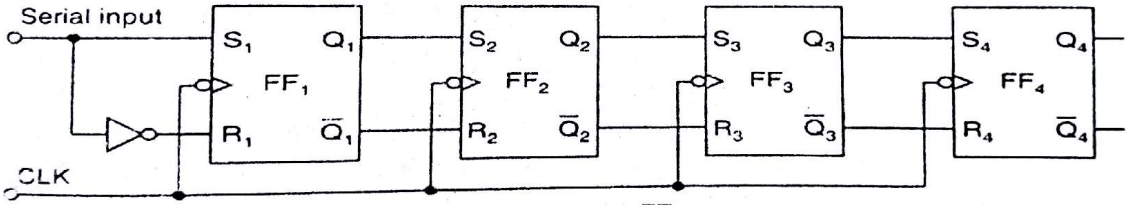
stored by the first FF is transferred to the 2nd, and so on... The bit that was stored by the last FF is shifted out.



A shift register can also be constructed using J-K (or) S-R FF's as shown in below figure. The data is applied at the J(S) i/p of the first FF. The complement of this is fed to K(R) terminal of the first FF. and so on....



(a) Using J-K FFs

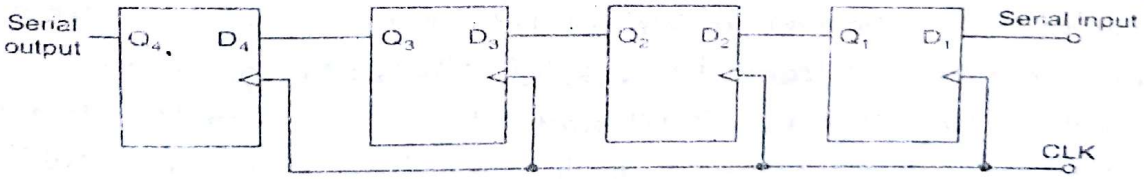


(b) Using S-R FFs

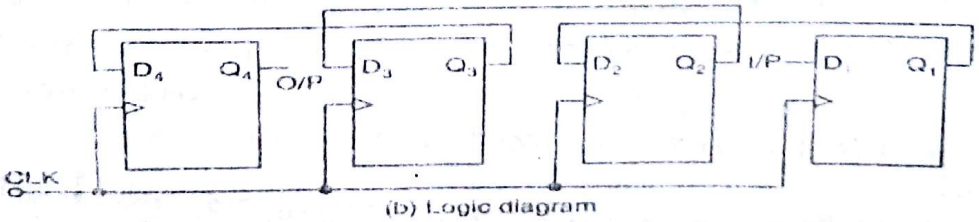
A 4-bit serial-in, serial-out, shift register.

Serial-In, Serial-Out, Shift-Left, Shift Register:

fig shows the logic diagrams of 4-bit serial-in, serial-out, shift-left, shift register.



(a) Logic diagram



(b) Logic diagram

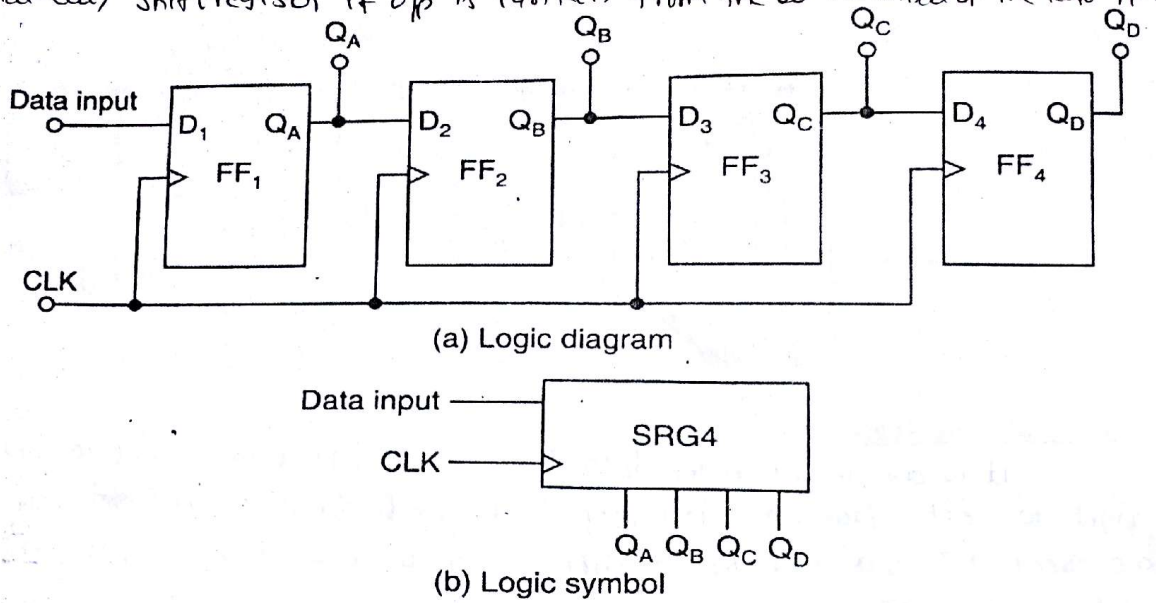
Serial-In, Parallel-Out, Shift Register:

Fig shows the logic diagram & the logic symbol of a 4-bit serial-in, parallel-out, shift register. In this the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form.

Once the data bits are stored, each bit appears on its respective o/p line and all bits are available simultaneously, rather than on a bit by bit basis as

G.NAVEEN

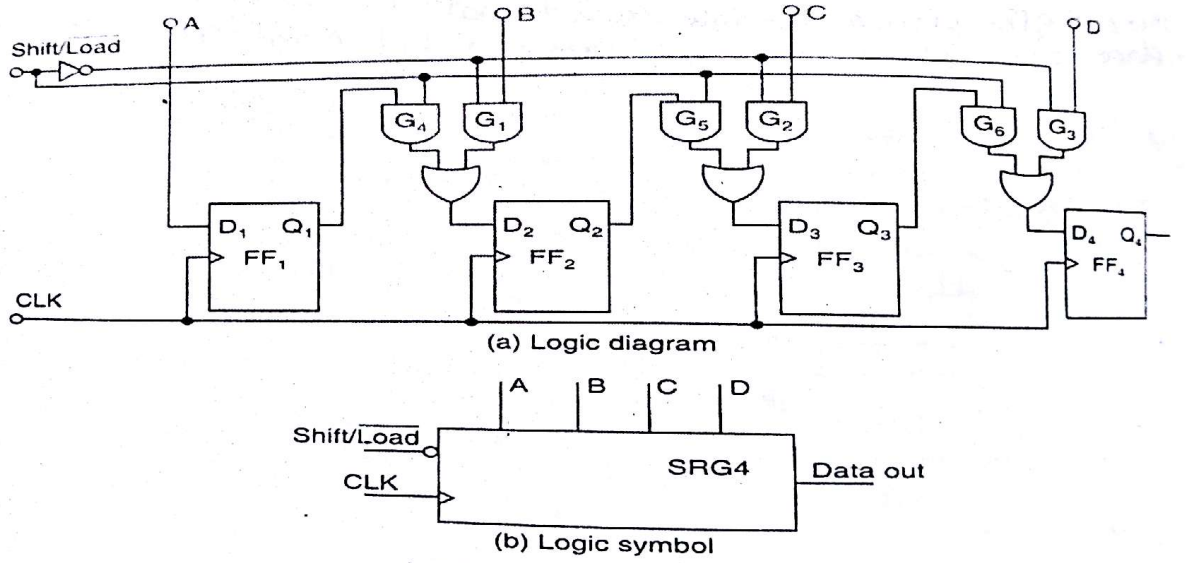
with the serial o/p. The serial-in, parallel-out, shift register can be used as a serial-In, serial-out, shift register if o/p is taken from the Q terminal of the last FF.



A 4-bit serial-in, parallel-out, shift register.

Parallel-In, Serial-Out, Shift Register:

There are 4 data lines A, B, C & D, through which the data is entered into the register in parallel form. The signal Shift/$\overline{LOAD}$ allows the data to be entered in parallel form into the register and (b) the data to shifted out serially from terminal Q_4.



A 4-bit parallel-in, serial-out, shift register.

When Shift/$\overline{LOAD}$ line is HIGH., gates G_1, G_2 & G_3 are disabled, but G_4, G_5, & G_6 are enabled allowing the data bits to shift-right from one stage to the next.

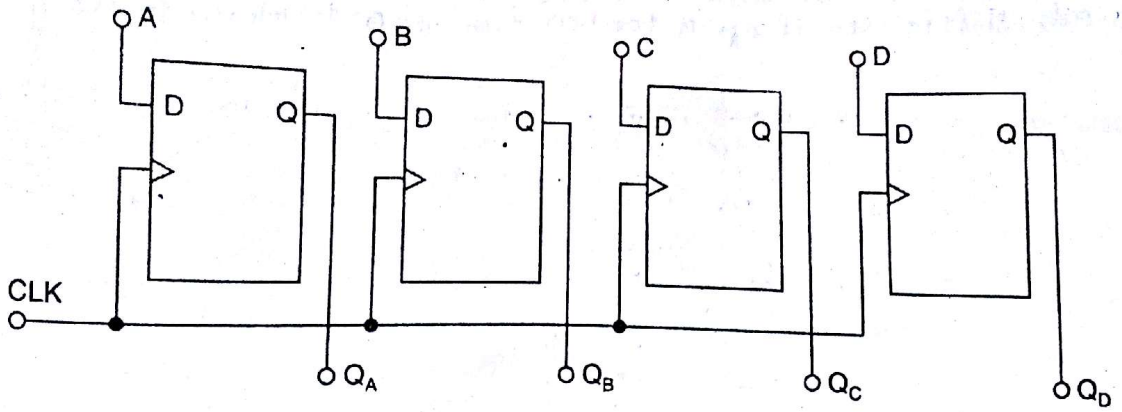
When Shift/$\overline{LOAD}$ line is LOW, gates G_4, G_5, & G_6 are disabled, whereas G_1, G_2 & G_3 are enabled allowing the data i/p to appear at the D i/p's of the respective FF's. When a clock pulse is applied, these data bits are shifted to the Q o/p terminals of the FF's & data is shifted in one step. OR gates allows the parallel data depending on which AND gates are enabled.

Parallel-In, Parallel-Out, Shift Register:

The data is entered into the register in parallel form and also the data is taken out of the register in parallel form.

From Fig. Data is applied to the D i/p terminals of the FF's. When a clock pulse is applied, at the positive-going edge of that pulse, the D i/p are shifted into the Q o/p's of the FF's. The register now stores the data. The stored data is available

Instantaneously for shifting out in parallel form.

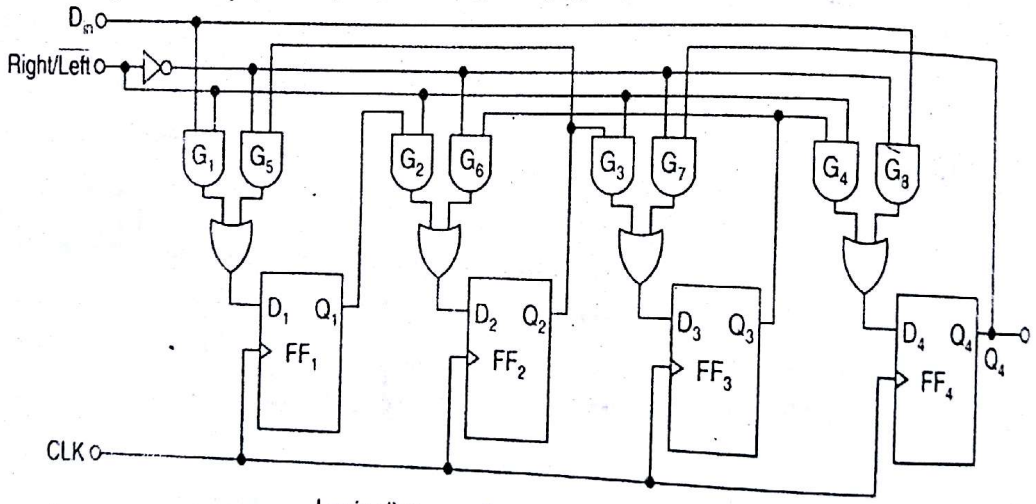


Bidirectional Shift Register:

It is one in which the data bits can be shifted from left to right or from right to left. From Fig Right/$\overline{Left}$ is the mode signal. When Right/$\overline{Left}$ is a '1', the logic circuit works as a shift-right shift register. When Right/$\overline{Left}$ is 0, it works as a shift-left shift register.

A HIGH on the Right/$\overline{Left}$ control i/p enables the AND gates G_1, G_2, G_3 & G_4 & disables the AND gates G_5, G_6, G_7 & G_8. & the state of Q o/p of each FF is passed through the gate to the D input of the following FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the right.

A LOW on the Right/$\overline{Left}$ control i/p enables the AND gates G_5, G_6, G_7 & G_8 and disables the AND gates G_1, G_2, G_3 & G_4 and the Q o/p of each FF is passed to the D i/p of the preceding FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the left. Hence the CKT works as a bidirectional shift register.



Logic diagram of a 4-bit bidirectional shift register.

Universal Shift Registers:

A register capable of shifting in one direction only is a unidirectional shift register. If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register. So universal shift register is a bidirectional register, whose i/p can be either in serial form or in parallel form and whose o/p also can be either in serial form (or) in parallel form.

The most general shift register has the following capabilities.

1. A clear control to clear the register to 0.
2. A clock i/p to synchronize the operations
3. A shift right control to enable the shift-right operation and serial i/p & o/p lines associated with the shift-right.

G.NAVEEN
NEC, EEE Dept

Universal shift register:

A register is capable of shifting in one direction only is a unidirectional shift register. One that can shift both directions is a bidirectional shift register. If the register has both shifts and parallel load capabilities, it is referred to as a universal shift registers. Universal shift register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be in serial form or I parallel form.

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.

A universal shift register can be realized using multiplexers. The below fig shows the logic diagram of a 4-bit universal shift register that has all capabilities. It consists of 4 D flip-flops and four multiplexers. The four multiplexers have two common selection inputs s1 and s0. Input 0 in each multiplexer is selected when S1S0=00, input 1 is selected when S1S0=01 and input 2 is selected when S1S0=10 and input 4 is selected when S1S0=11. The selection inputs control the mode of operation of the register according to the functions entries. When S1S0=0, the present value of the register is applied to the D inputs of flip-flops. The condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs. When S1S0=01, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flop. This causes a shift-right operation, with serial input transferred into flip-flopA4. When S1S0=10, a shift left operation results with the other serial input going into flip-flop A1. Finally when S1S0=11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock cycle

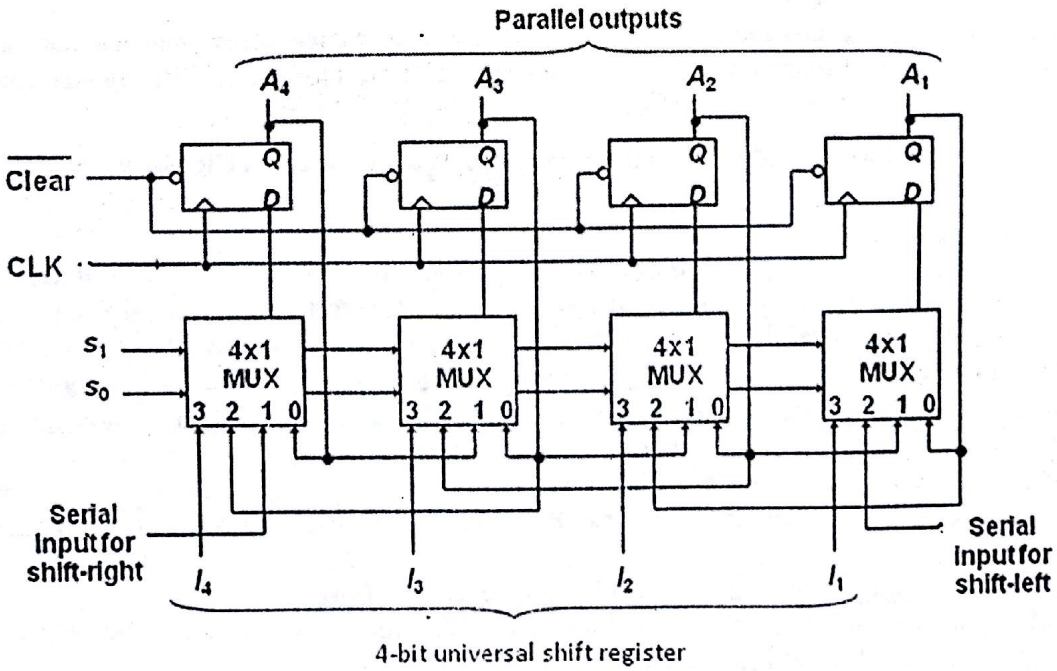


Figure: logic diagram 4-bit universal shift register

Function table for the register:

mode control		
S0	**S1**	**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 in oven 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:

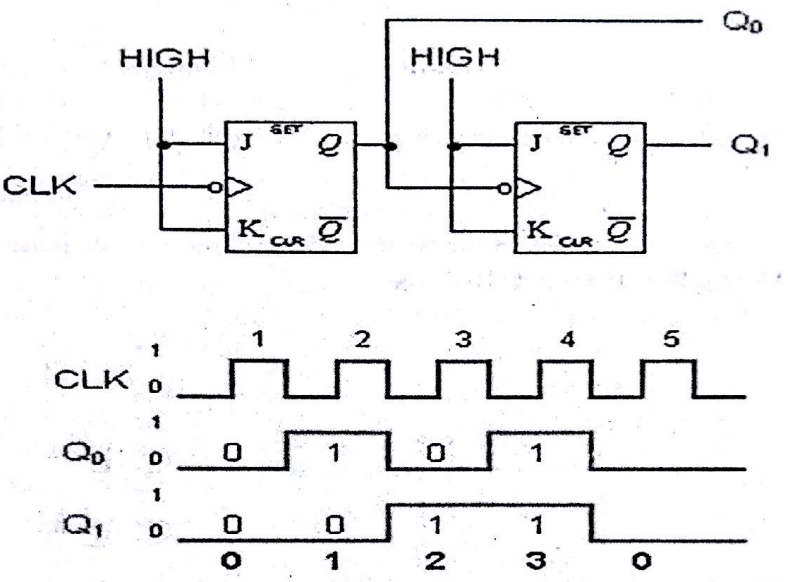
Two bit ripple counter used two flip-flops. There are four possible states from 2 – bit up-counting I.e. 00, 01, 10 and 11.

The counter is initially assumed to be at a state 00 where the outputs of the tow flip-flops

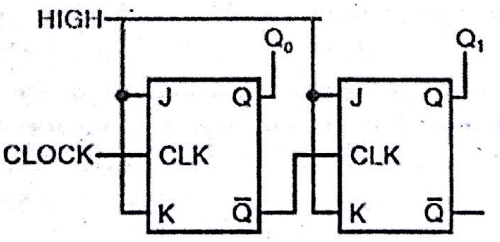
G.NAVEEN
NEC, EEE Dept

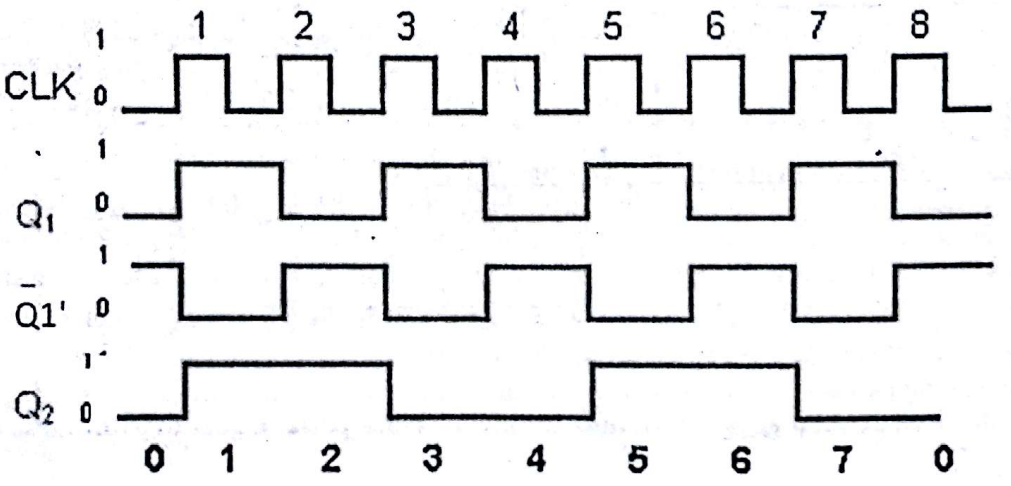
are noted as Q1Q0. Where Q1 forms the MSB and Q0 forms the LSB.

- For the negative edge of the first clock pulse, output of the first flip-flop FF1 toggles its state. Thus Q1 remains at 0 and Q0 toggles to 1 and the counter state are now read as 01.
- During the next negative edge of the input clock pulse FF1 toggles and Q0 = 0. The output Q0 being a clock signal for the second flip-flop FF2 and the present transition acts as a negative edge for FF2 thus toggles its state Q1 = 1. The counter state is now read as 10.
- For the next negative edge of the input clock to FF1 output Q0 toggles to 1. But this transition from 0 to 1 being a positive edge for FF2 output Q1 remains at 1. The counter state is now read as 11.
- For the next negative edge of the input clock, Q0 toggles to 0. This transition from 1 to 0 acts as a negative edge clock for FF2 and its output Q1 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, 00,11,10,01,00,11 .....,etc. 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 Q1' is applied to the clock input of FF2, toggles Ff2 and, therefore, Q2 goes from a 0 to a 1.so, after one clock pulse Q2=1 and Q1=1, I.e., the state of the counter is 11.
- At the negative-going edge of the second clock pulse, Q1 changes from a 1 to a 0 and Q1' from a 0 to a 1.
- This positive-going signal at Q1' does not affect FF2 and, therefore, Q2 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 Q1, goes from a 0 to a 1 and Q1' from 1 to 0. This negative going signal at Q1' toggles FF2 and, so, Q2 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 Q1, goes from a 1 to a 0 and Q1' from 0 to 1. . This positive going signal at Q1' does not affect FF2 and, so, Q2 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:

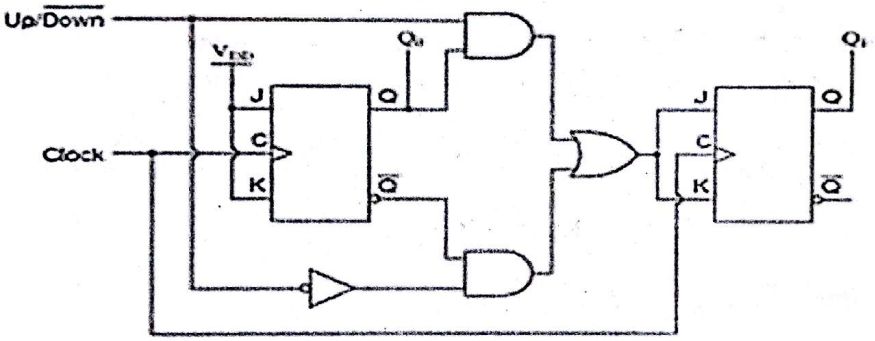


Figure: asynchronous 2-bit ripple up-down counter using negative edge triggered flip flop

- 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, Q1 is transmitted to clock of FF2 and when M=0 for down counting, Q1' 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= (Q1.Up)+(Q1'. Down)=Q1m+Q1'M'

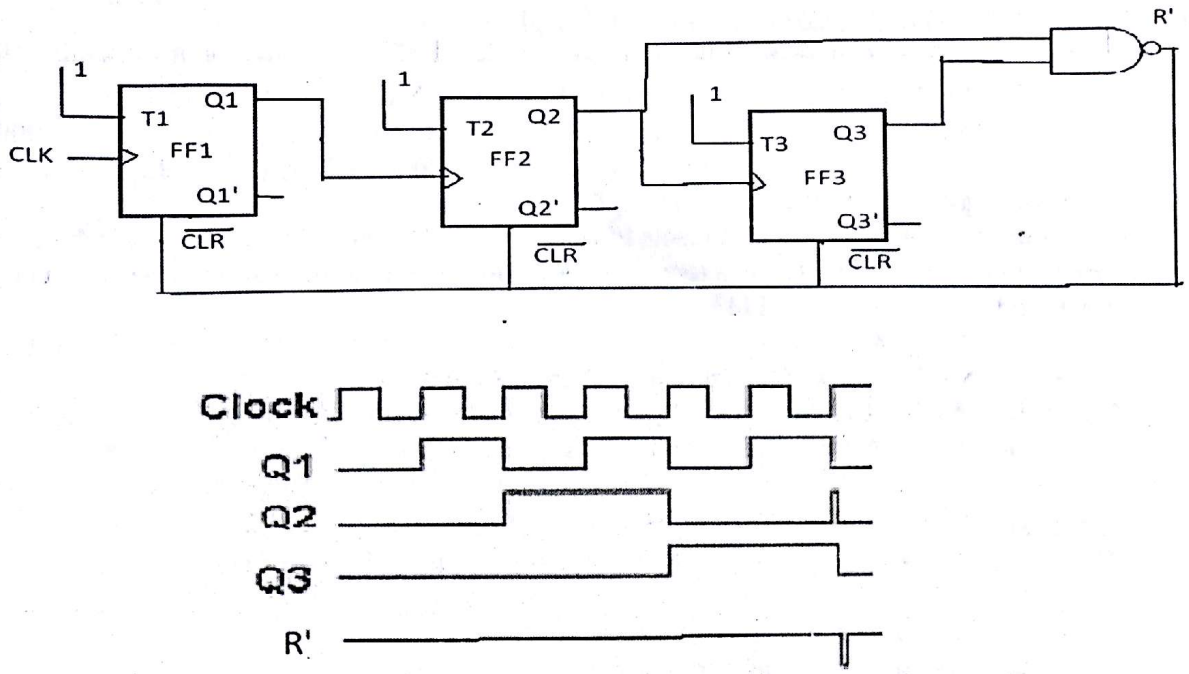
Design of Asynchronous counters:

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 –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 110and 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.

G. NAVEEN
NEC, EEE DEPT



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 Q1that decides the feedback into be provided. From the truth table, R=Q3Q2. 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=for111

Therefore,

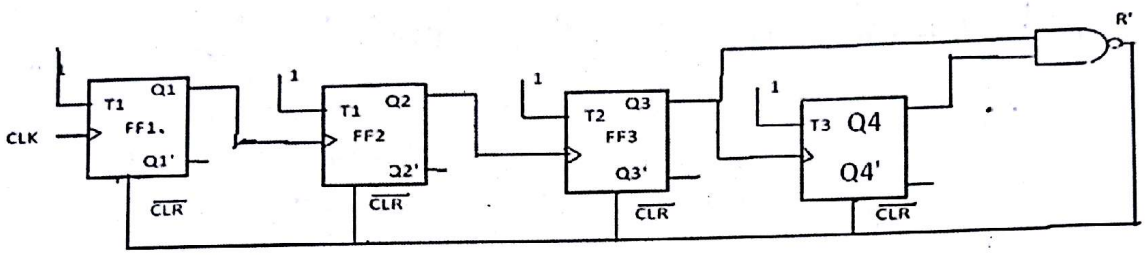
R=Q3Q2Q1‘+Q3Q2Q1=Q3Q2

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

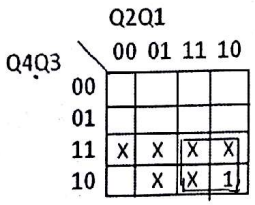
After pulses	States Q3	Q2	Q1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	↓
	0	0	0	0
7	0	0	0	0

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

A mod-10 counter is a decade counter. It 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 state, 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=Q4Q2. So, feedback is provided from second and fourth FFs. For active –HIGH reset, Q4Q2 is applied to the clear terminal. For active-LOW reset $Q4Q2$ is connected CLR isof all Flip=flops.

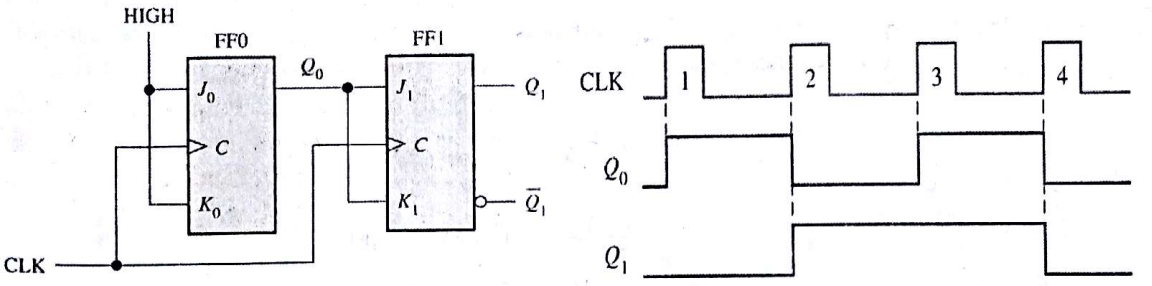


After pulses	Count Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

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 is shown in below fig.

GNAVEEN
NEC, EEEDept



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

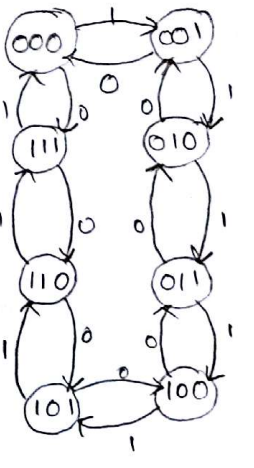
Step3: select the type of flip flop and draw the excitation table: JK flip-flops are selected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in fig.

JK FF excitation Table

PS NS J K
0 0 0 x
0 1 1 x
1 0 x 1
1 1 x 0

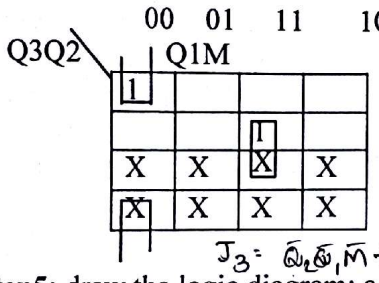
PS			mode	NS			required excitations					
Q3	Q2	Q1	M	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	1	1	1	1	x	1	x	1	x
0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	0	0	0	x	0	x	x	1
0	0	1	1	0	1	0	0	x	1	x	x	1
0	1	0	0	0	0	1	0	x	x	1	1	x
0	1	0	1	0	1	1	0	x	x	0	1	x
0	1	1	0	0	1	0	0	x	x	0	x	1
0	1	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	1	1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

Excitation table.



state diagram

Step4: obtain the minimal expressions: From the excitation table we can conclude that J1=1 and K1=1, because all the entries for J1and 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.

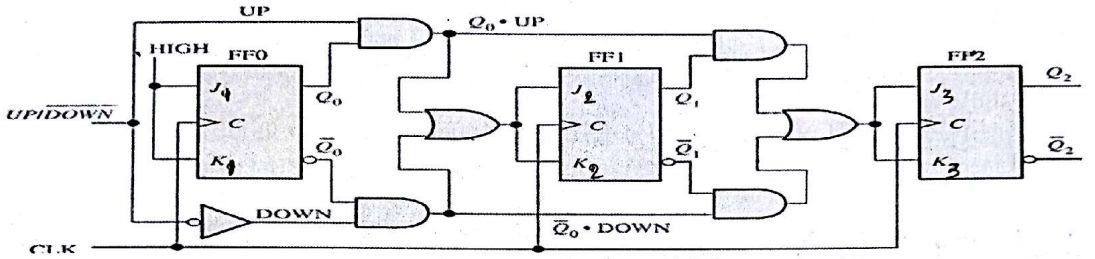


$K_3 = \bar{Q}_2\bar{Q}_1\bar{M} + Q_2Q_1M$

$J_2 = \bar{Q}_1\bar{M} + Q_1M$

$K_2 = \bar{Q}_1\bar{M} + Q_1M$

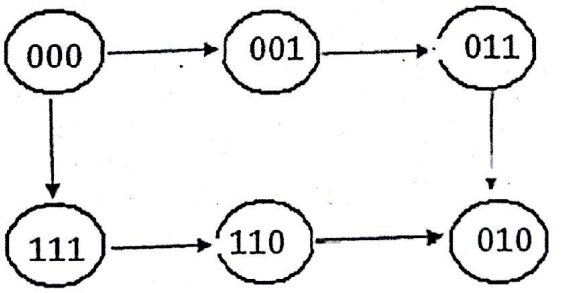
Step5: draw the logic diagram: a logic diagram using those minimal expressions can be drawn as shown in fig.



Design of a synchronous modulo-6 gray code 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=3FFs ($N \le 2^n$, i.e., $6 \le 2^3$). 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.

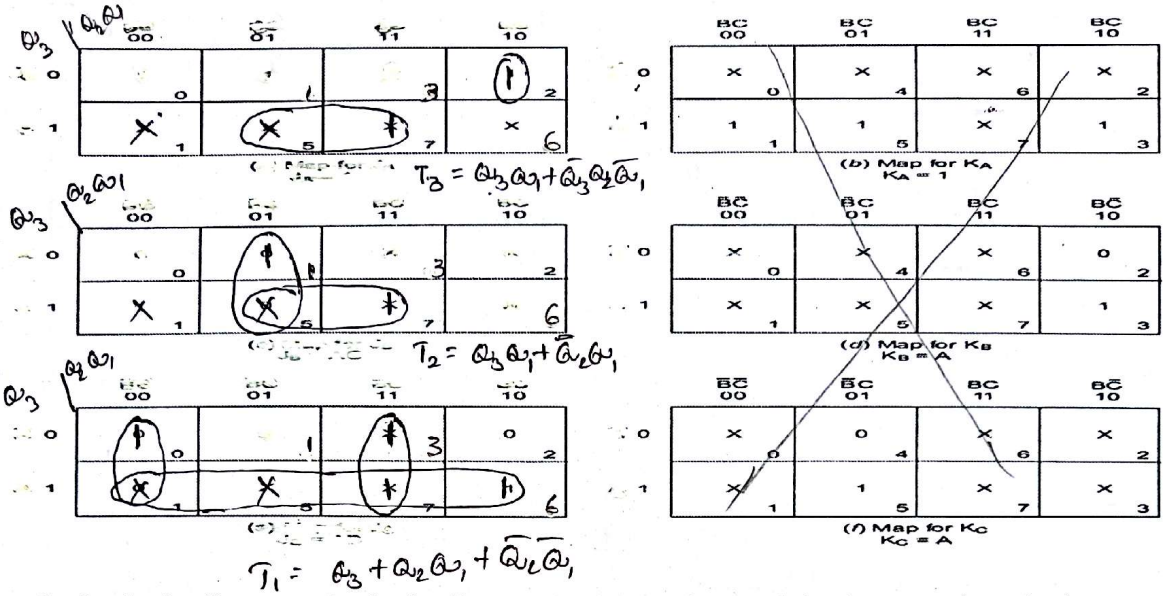
PS			NS			required excitations		
Q3	Q2	Q1	Q3	Q2	Q1	T3	T2	T1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T-FF Excitation

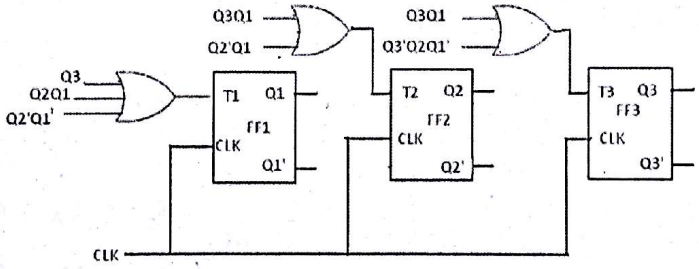
PS	NS	T
0	0	0
0	1	1
1	0	1
1	1	0

G.NAVEEN
NEC, EEE DEPT

Step4: The minimal expressions: the K-maps for excitations of FFs T3,T2,and T1 in terms of outputs of FFs Q3,Q2, and Q1, their minimization and the minimal expressions for excitations obtained from them are shown if fig



Step5: the logic diagram: the logic diagram based on those minimal expressions is drawn 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=4FFs($N \leq 2^n$,, 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 isapplied to all FFs.

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

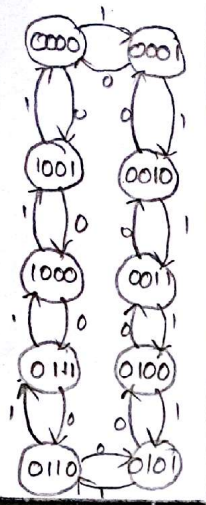
Step3: types of flip-flops and excitation table: T flip-flops are selected and the excitation table of the modulo-10 up down counter using T flip-flops is drawn as shown in fig.

The remaining minterms are don't cares($\sum$d(20,21,22,23,24,25,26,37,28,29,30,31)) 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	T1
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	0	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

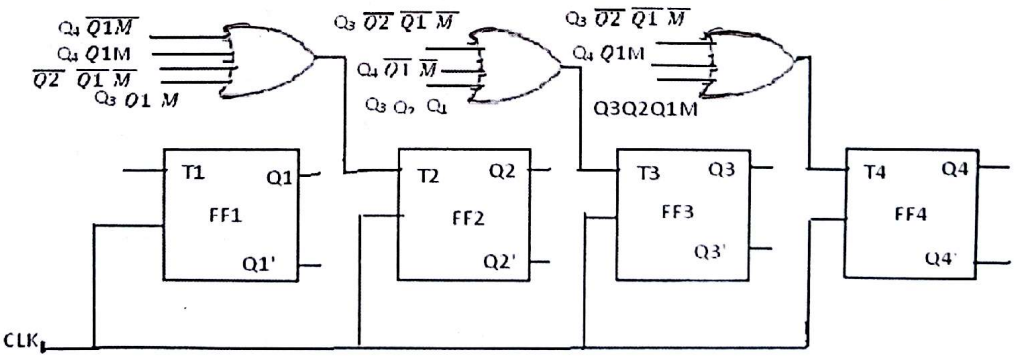
Step4: The minimal expression: since there are 4 state variables and a mode signal, we require 5 variable kmaps. 20 conditions of Q4Q3Q2Q1M are valid and the remaining 12 combinations are invalid. So the entries for excitations corresponding to those invalid combinations are don't cares. Minimizing K-maps for T2 we get

$$T2 = Q4\overline{Q1}\,\overline{M} + \overline{Q4}\,Q1M + Q2\overline{Q1}\,\overline{M} + Q3\overline{Q1}\,\overline{M}$$

$T_3 = Q_4\bar{Q}_1\bar{M} + Q_2Q_1M + Q_3\bar{Q}_2\bar{Q}_1\bar{M}$

$T_4 = Q_4Q_1M + \bar{Q}_3\bar{Q}_2\bar{Q}_1M + Q_3Q_2Q_1M$

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



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.

G. NAVEEN
NEC, EEE DEPT

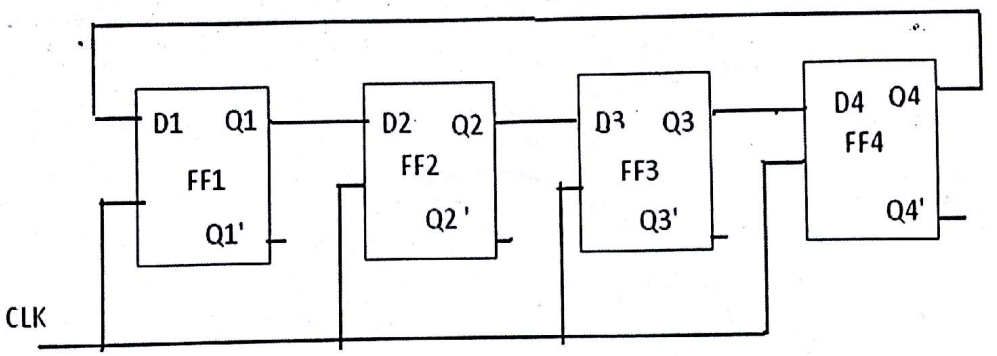


FIGURE: logic diagram of 4-bit ring counter using D flip-flops

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., Q1=1, Q2=0,Q3=0,Q4=0. After each clock pulse, the contents of the register are shifted to the right by one bit and Q4 is shifted back to Q1. 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, where as 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:

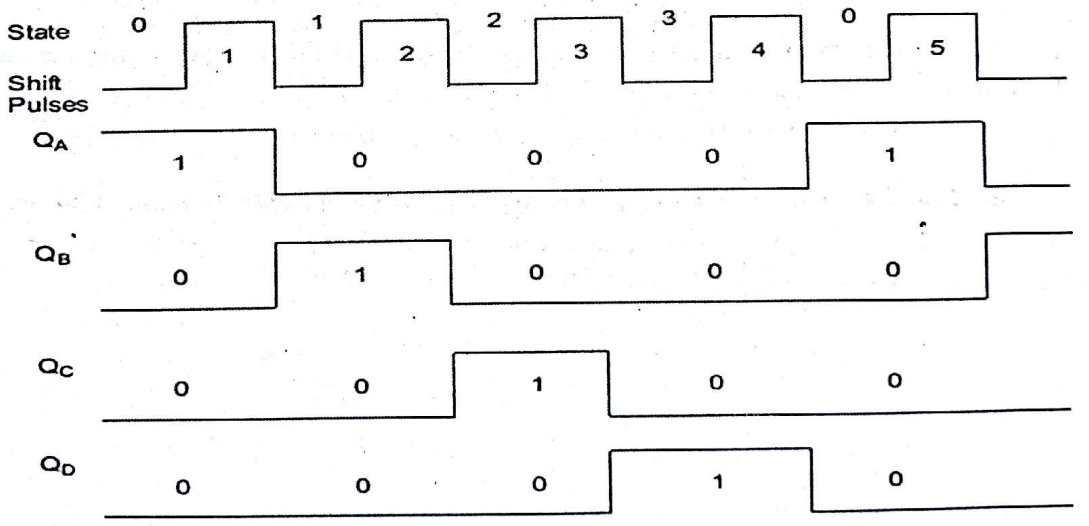
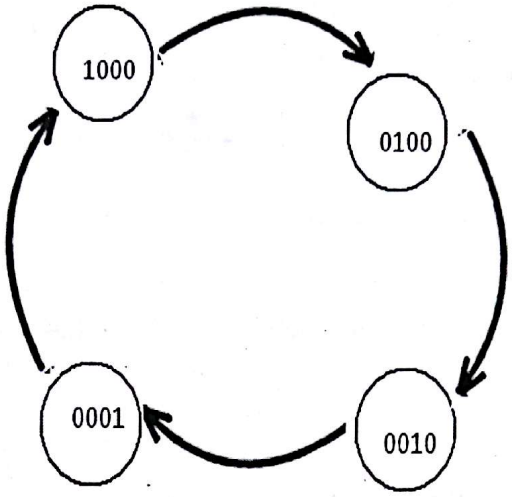


Figure: state diagram



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 Q2to Q3, Q3 to Q4 and the level of Q4'to Q1 and the sequences given in fig.

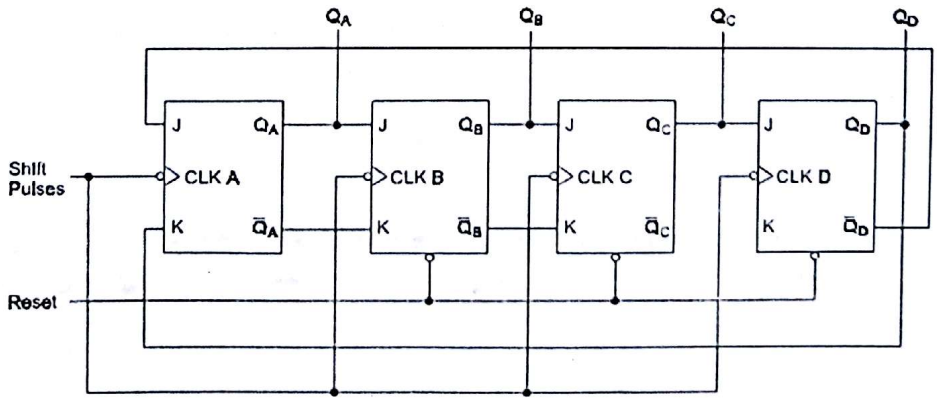


Figure: Johnson counter with JK flip-flops

G.NAVEEN
NEC, EEE Dept

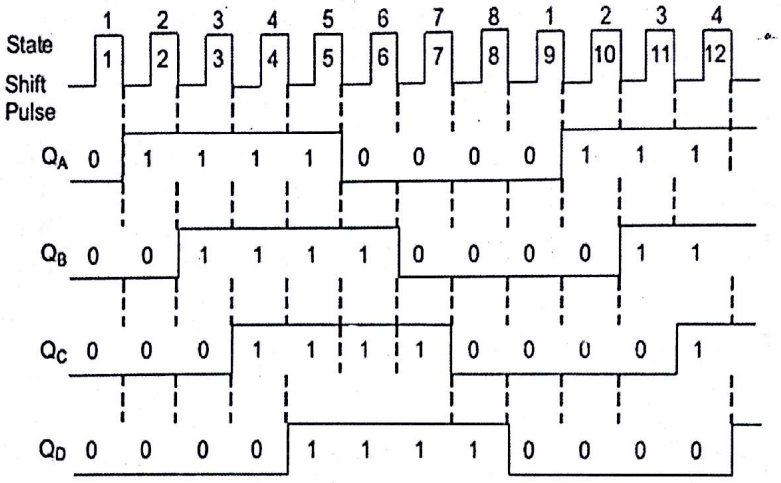
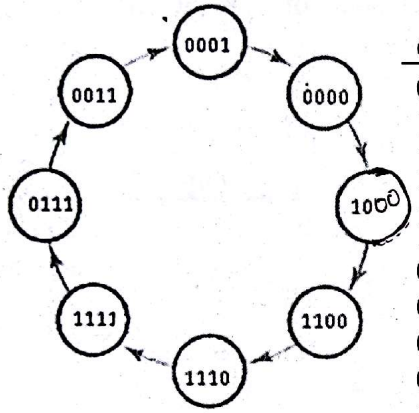


Figure: timing diagram

State diagram:



Q1	Q2	Q3	Q4	after clock pulse
0	0	0	0	0
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9

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. Meelay 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 fast that the functional behavior of these circuits can be represented using a finite number of states.

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.

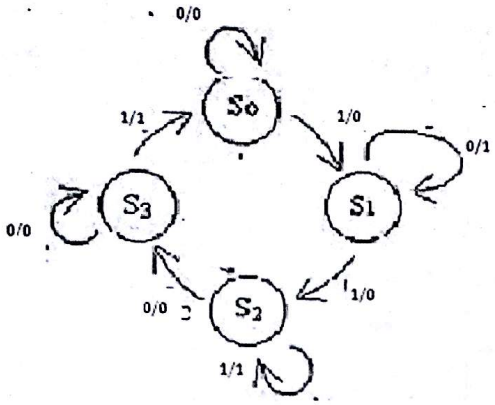


Fig :a) state diagram (meelay circuit)

	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: 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.

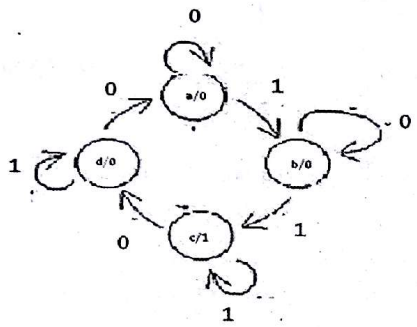


Fig: a) state diagram (moore circuit)

	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 X1and X2 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 Zi at time tiis a function of the inputs X1(ti) and X2(ti) at that time ti-1 and of carry which had been generated at ti-

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

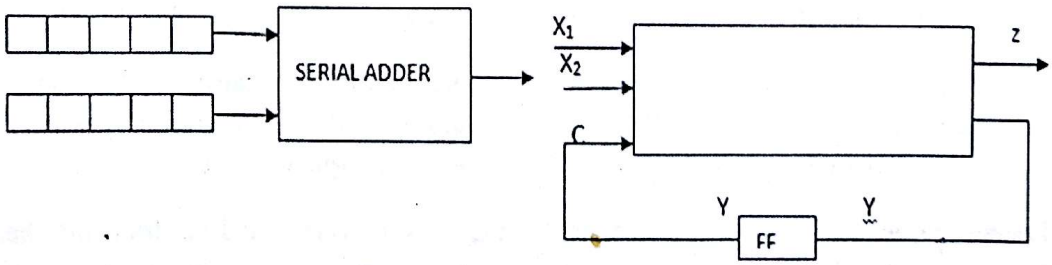


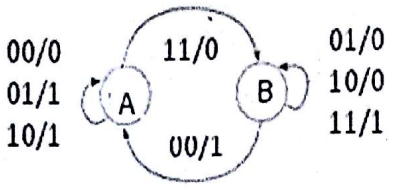
Figure: block diagram of serial binary adder

Step2 and 3: state diagram and state table: let a designate the state of the serial adder at ti if a carry 0 was generated at ti-1, and let b designate the state of the serial adder at ti if carry 1 was

G.NAVEEN
NEC, EEE DEPT

generated at ti-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	1	1
	0	1	0	1
A	A,0	B,0	B,1	B,0
B	A,1	B,0	B,0	B,1

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 X1=0 and X2=1 gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs X1=1 and X2=0 gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs X1=1 and X2=1 gives sum, 1 and carry 0. So the machine remains in state B and outputs a 1. Inputs X1=0 and X2=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.

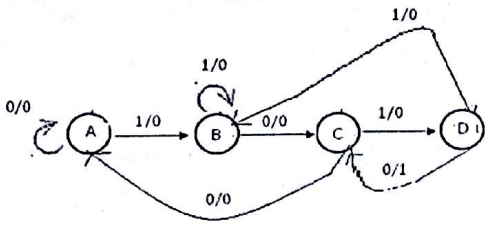
PS	I/P		NS	I/P-FF	O/P
y	x1	x2	Y	D	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Sequence detector:

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 t1, 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 o r 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.

	NS(Y1Y2)		O/P(z)	
PS(y1y2	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 - FFS		
PS		**I/P**	**NS**				**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

G. NAVEEN
EEE DEPT
NEC