Sequential Circuits

Digital System Design

Course Instructor: Ganaraj P S

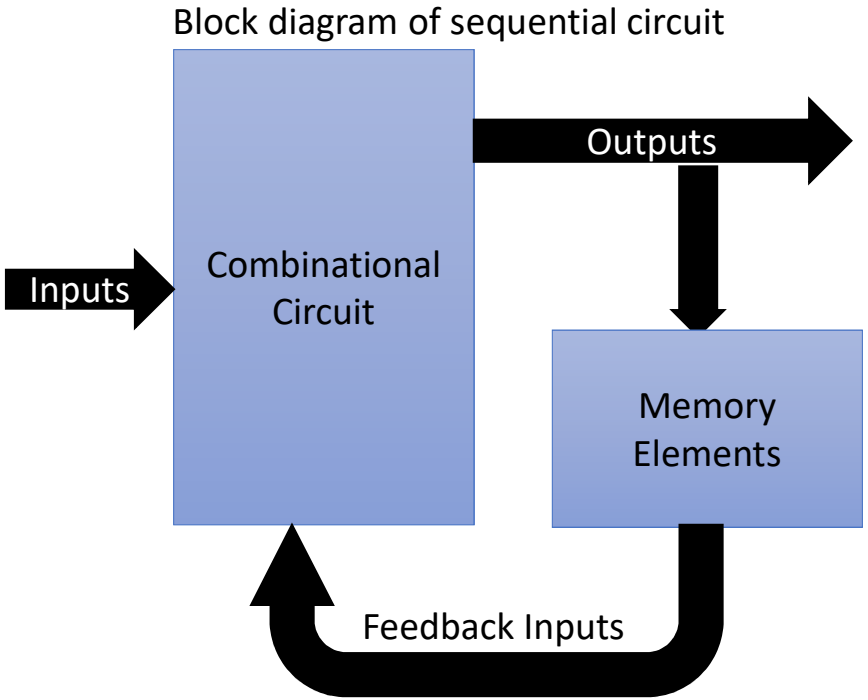
Assistant Professor (I), School of Electronics Engineering

Kalinga Institute of Industrial Engineering, Bhubaneswar

Fundamentals of sequential logic:

- In combinational circuits, output(s) depends only on the combination of present inputs, whereas in case of sequential circuits output(s) depends not only on the present input but also on the previous output(s).

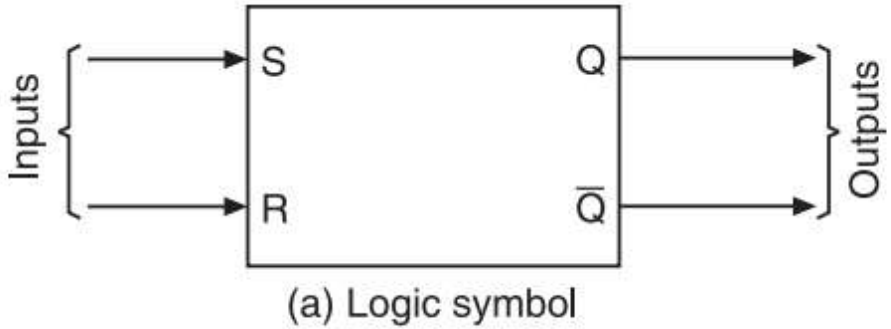
- Sequential circuits contain combinational circuits along with memory (storage) elements.
- The latches and flip-flops are the basic building block for sequential circuits, such as, memories.
- Sequential circuits includes counters, registers, and other sequential control logic



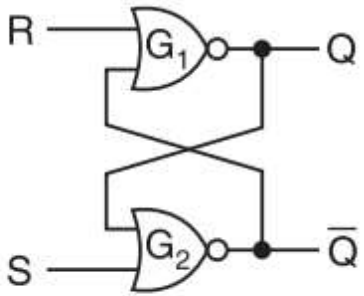
Combinational Circuits	Sequential Circuits
Outputs depend only on present inputs.	Outputs depend on both present inputs and present state.
Feedback path is not present.	Feedback path is present.
Memory elements are not required.	Memory elements are required.
Clock signal is not required.	Clock signal is required.
Easy to design.	Difficult to design.

- **Flip-flops** (also known as **Bistable Multivibrators)** which have two stable states, called SET and RESET; they can retain either of these states indefinitely, making them useful as storage devices.

S-R Latch (Active-High S-R latch):



(a) Logic symbol

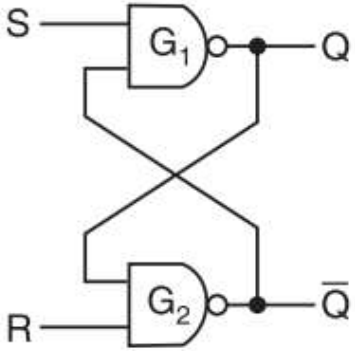


(b) Logic diagram

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	×	Indeterminate (invalid)
1	1	1	×	

(c) Truth table

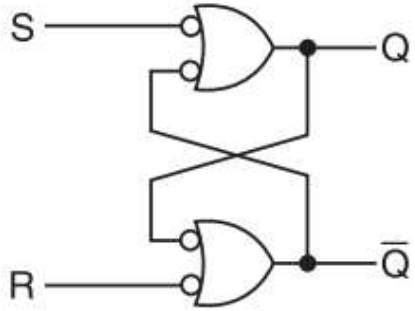
S-R Latch (Active-Low S-R latch):



(a) Using NAND gates

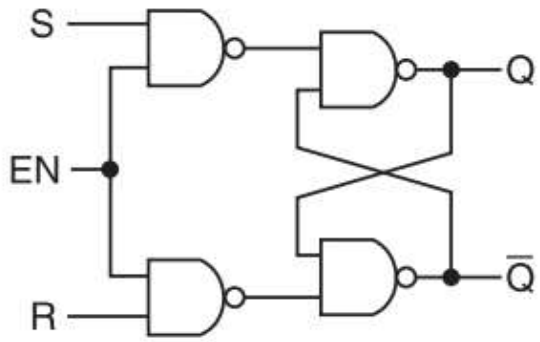
S	R	Q_n	Q_{n+1}	State
0	0	0	×	Indeterminate (invalid)
0	0	1	×	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No Change (NC)
1	1	1	1	

(b) Truth table

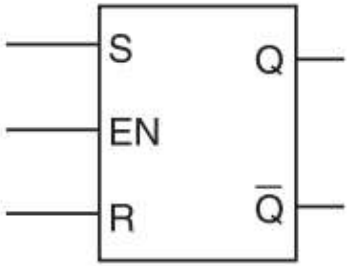


(c) Using OR gates

Gated S-R Latch (Clocked S-R Flip-Flop):



(a) Logic diagram

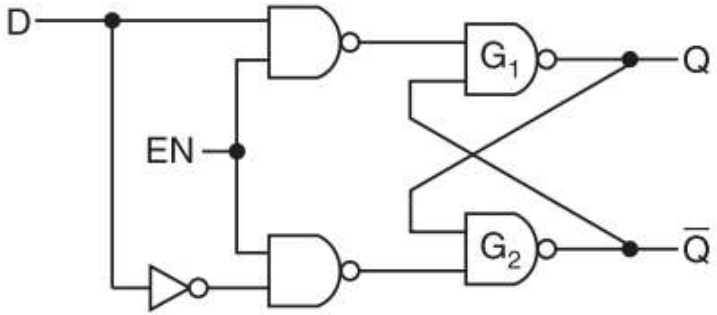


(b) Logic symbol

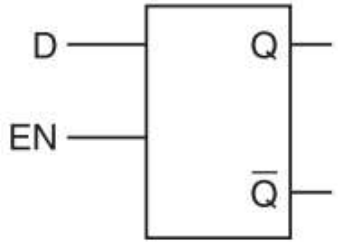
EN	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	×	Indeterminate (invalid)
1	1	1	1	×	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

(c) Truth table

Gated D Latch (Clocked D Flip-Flop):



(a) Logic diagram

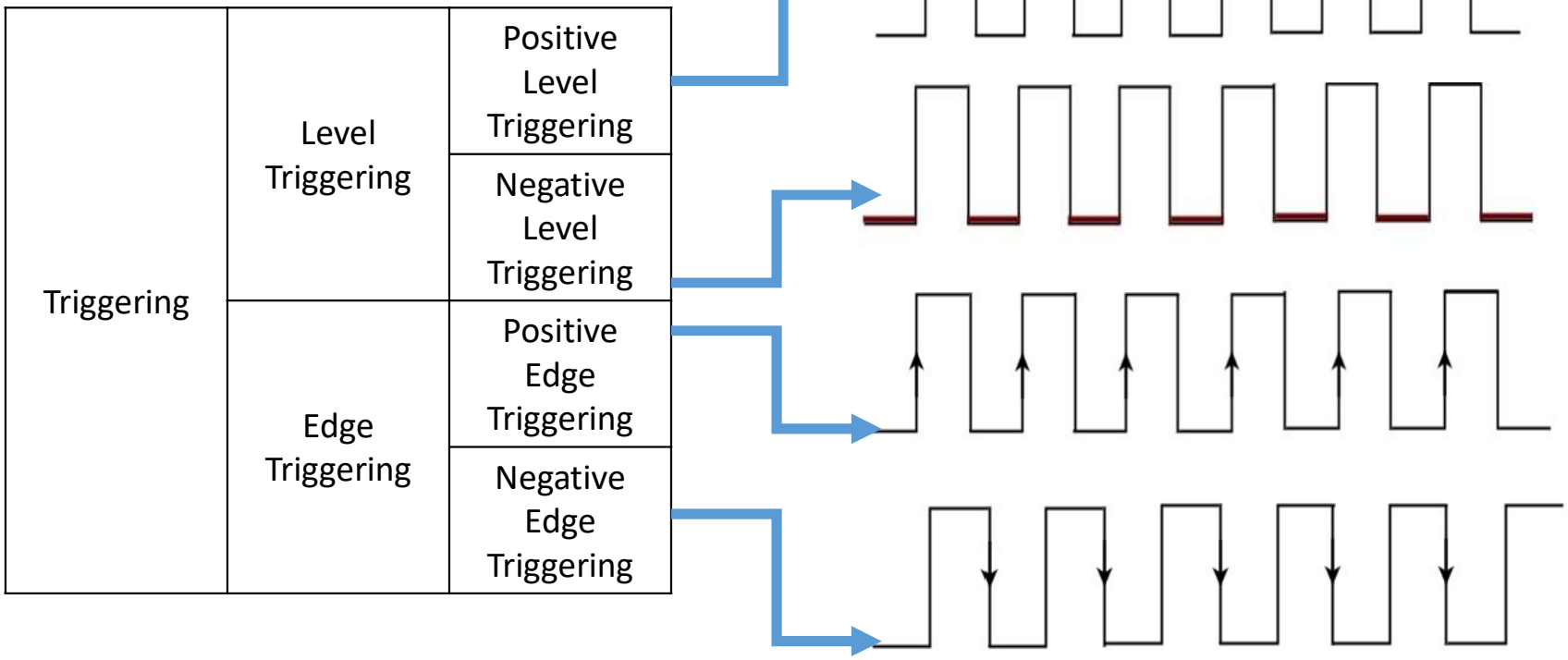


(b) Logic symbol

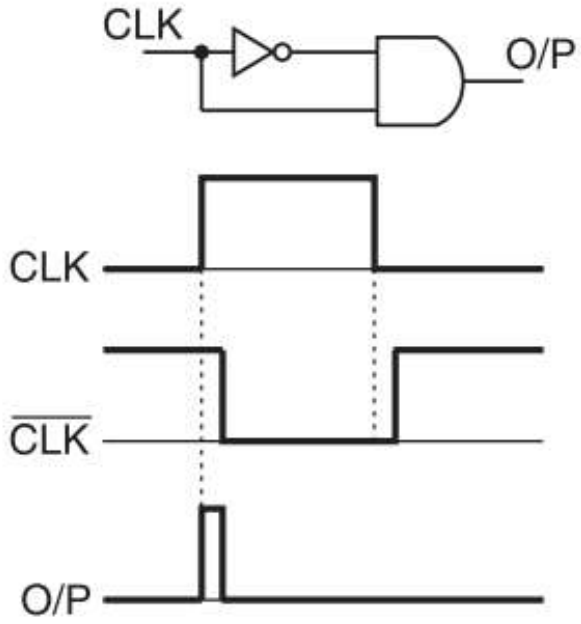
EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	×	0	0	No Change (NC)
0	×	1	1	

(c) Truth table

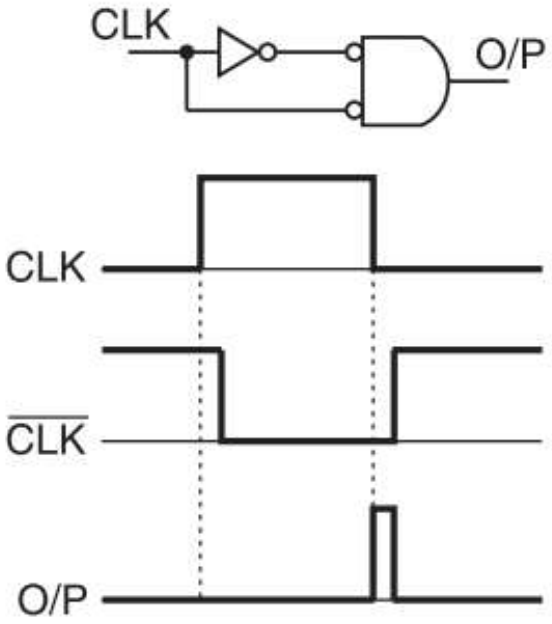
Clock Signal and Triggering



Generation of Narrow Spikes (PTD*):



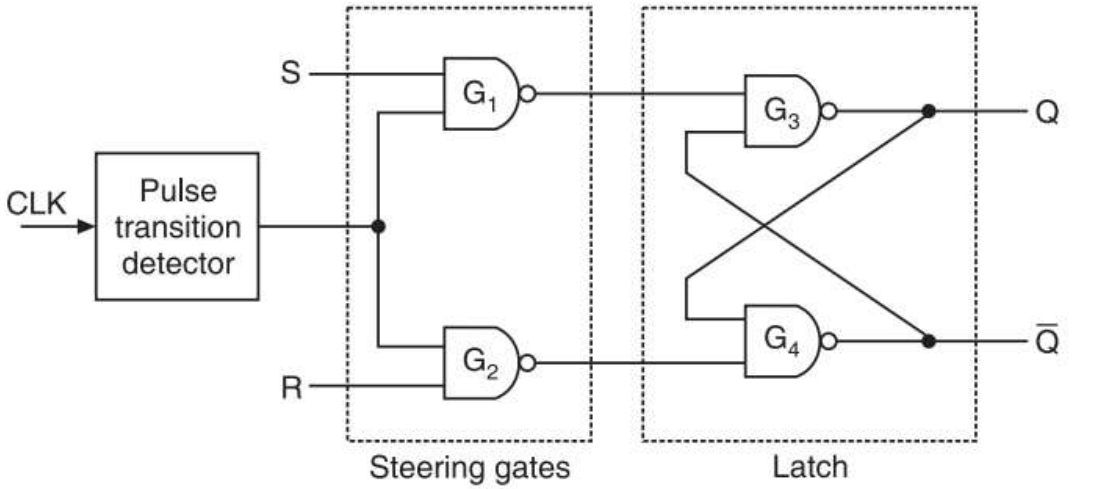
(a) Generation of a narrow spike at positive-going transition of the clock pulse

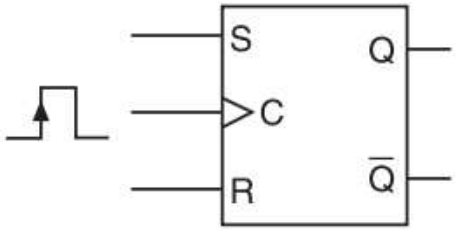


(b) Generation of a narrow spike at negative-going transition of the clock pulse

*** PTD: Pulse Transition Detector**

Edge-Triggered S-R Flip-Flop:



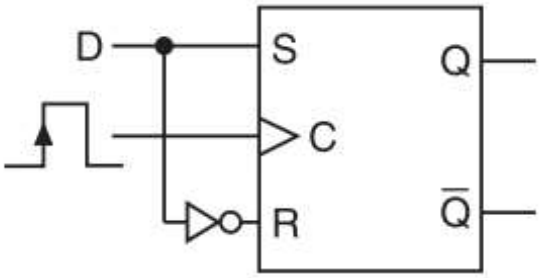


(a) Logic symbol

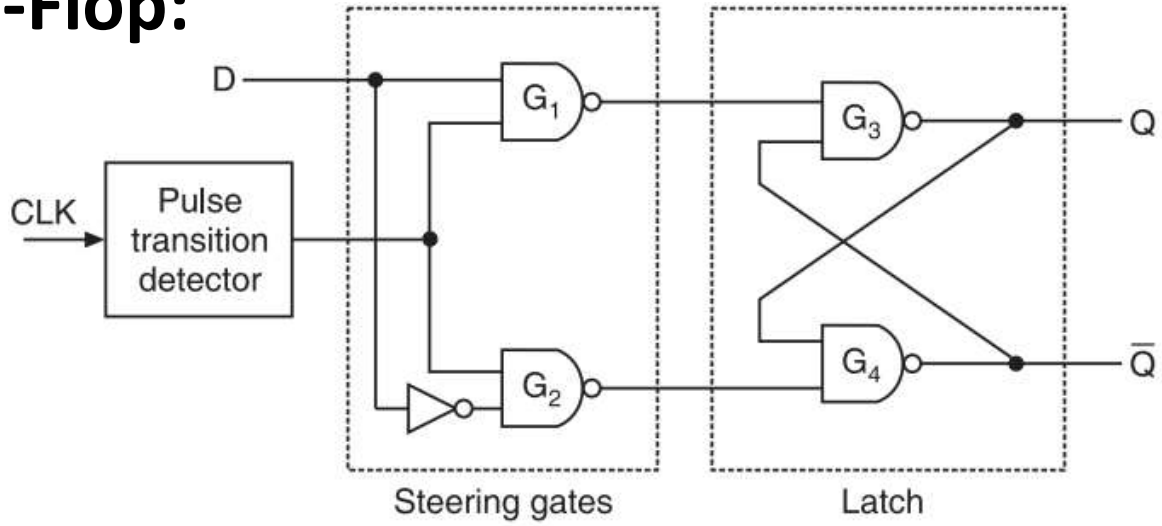
C	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change (NC)
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	×	Indeterminate
↑	1	1	1	×	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

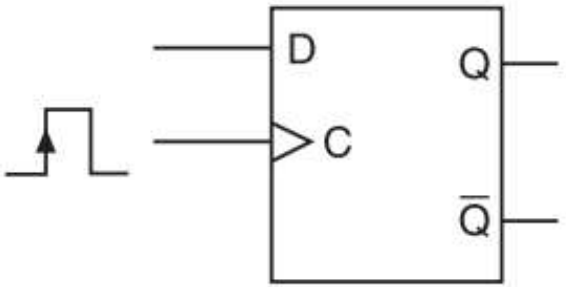
(b) Truth table

Edge-Triggered D Flip-Flop:



(a) D flip-flop from the S-R flip-flop



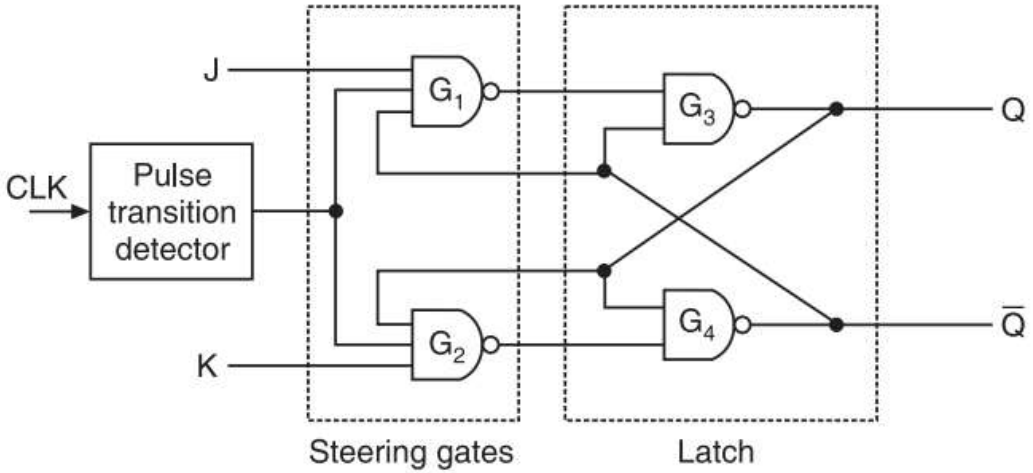


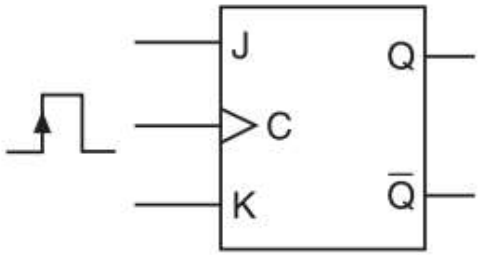
(b) Logic symbol

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

(c) Truth table

Edge-Triggered J-K Flip-Flop:





(a) Logic symbol

C	J	K	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	1	Toggle
↑	1	1	1	0	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

(b) Truth table

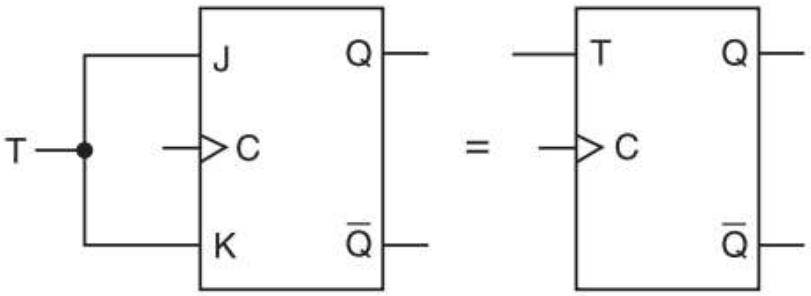
- **Design Code for JK Flip-flop**

```
module JK_FF(output q,qb, input j,k,clk);
reg q=0;
always@(posedge clk)
case({j,k})
    2'b00: q<=q;
    2'b01: q<=0;
    2'b10: q<=1;
    2'b11: q<=~q;
endcase
assign qb=~q;
endmodule
```

Test Bench Code for JK Flip-flop

```
module test_jkff();
reg j,k,clk=0;
always #10 clk=~clk;
JK_FF DUT(q,qb,j,k,clk);
initial begin
j<=0;k<=0;#5;
j<=0;k<=1;#20;
j<=1;k<=0;#20;
j<=1;k<=1;#20;
$finish;
end
endmodule
```

Edge-Triggered T Flip-Flop (Toggle Flip-Flop):

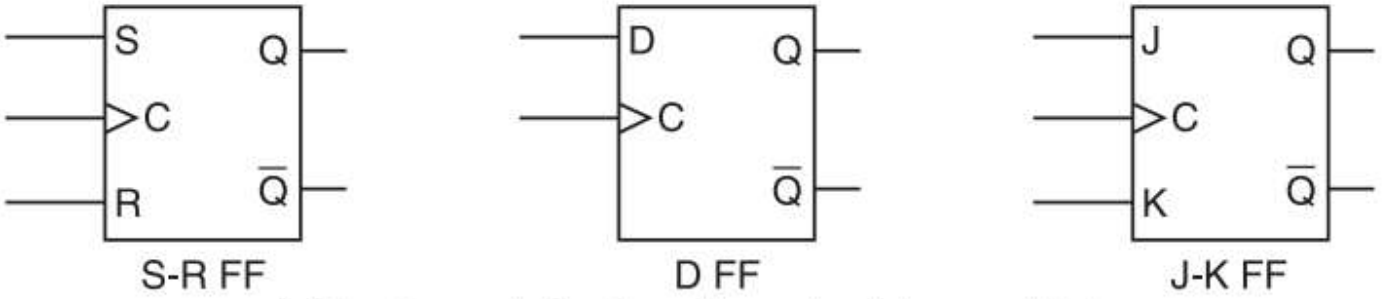


(a) T flip-flop from JK flip-flop (b) Logic symbol

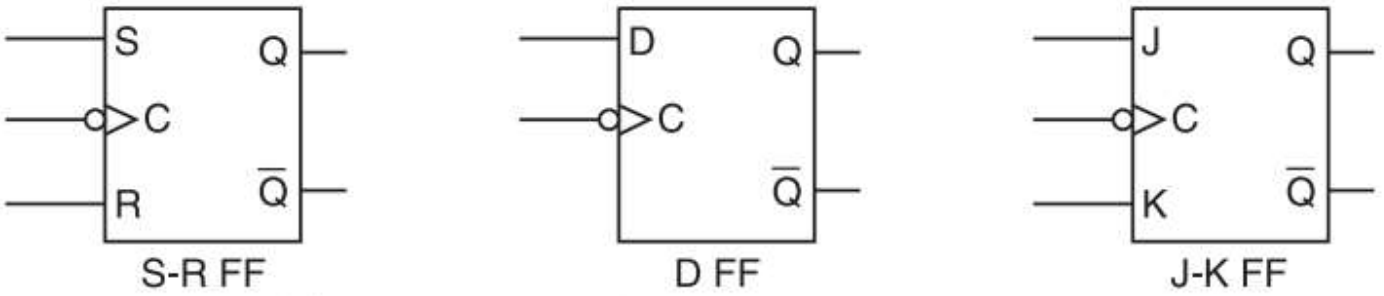
C	T	Q_n	Q_{n+1}	State
↑	0	0	0	No Change (NC)
↑	0	1	1	
↑	1	0	1	Toggle
↑	1	1	0	
0	×	0	0	No Change (NC)
0	×	1	1	

(c) Truth table

Symbols for +ve & -ve Edge-Triggered Flip-Flops:



(a) Logic symbols of positive edge-triggered FFs

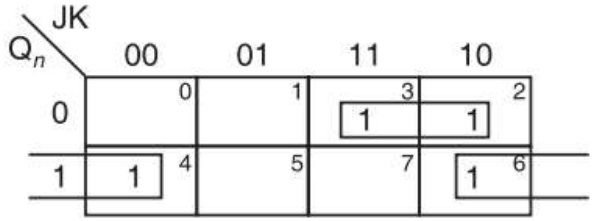


(b) Logic symbols of negative edge-triggered FFs

Characteristic Equations for Flip-Flops:

Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation requirements of JK flip-flop



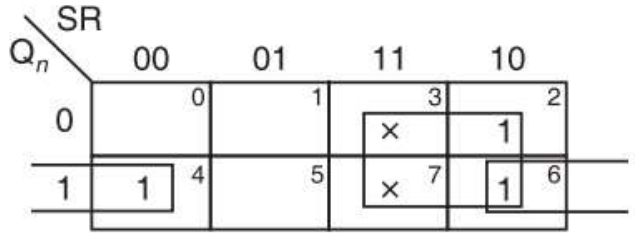
K-map for Q_{n+1}

The characteristic equation of a JK flip-flop is

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

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

Excitation requirements of SR flip-flop



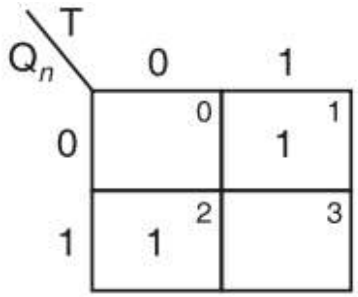
K-map for Q_{n+1} of SR flip-flop

The characteristic equation of SR flip-flop

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

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



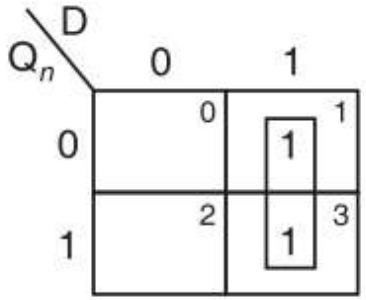
K-map for Q_{n+1} of T flip-flop

The characteristic equation of T flip-flop is

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

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



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

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

Excitation Tables:

During the design process, we usually know the transition from present state to next state and wish to find the flip-flop input conditions that will cause the required transition. For this reason, we need **a table that lists the required inputs for a given change of state.** Such a list is called an **excitation table**.

S-R Flip Flop			
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

J-K Flip Flop			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

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

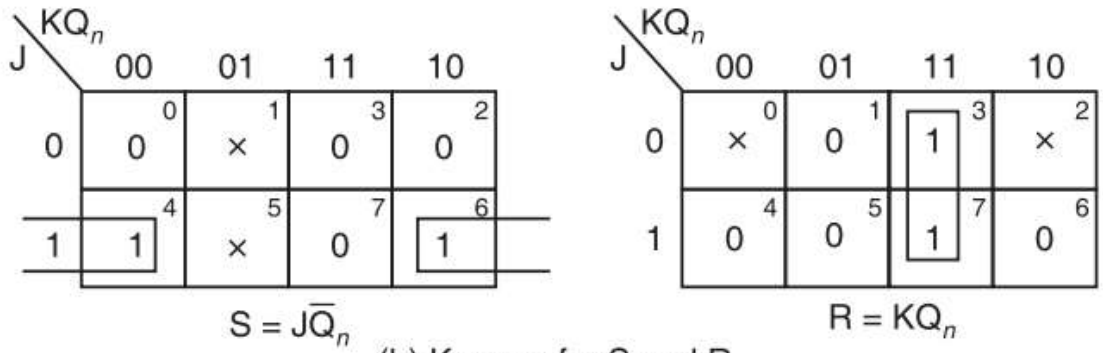
T Flip Flop		
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

CONVERSION OF FLIP-FLOPS:

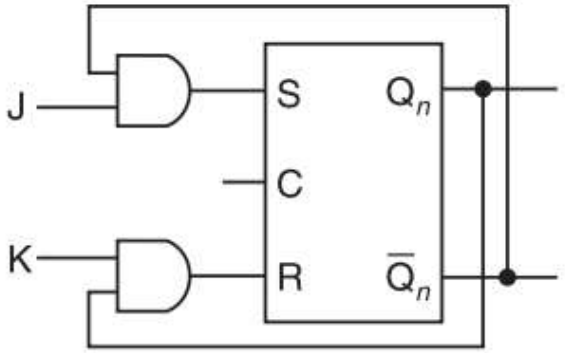
S-R flip-flop to J-K flip-flop:

External Inputs		Present State	Next State	Flip-flop Inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	×	0
1	1	0	1	1	0
1	1	1	0	0	1

(a) Conversion table



(b) K-maps for S and R

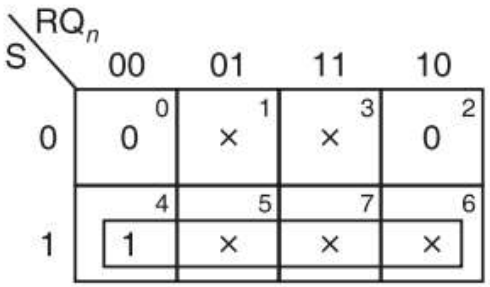


(c) Logic diagram

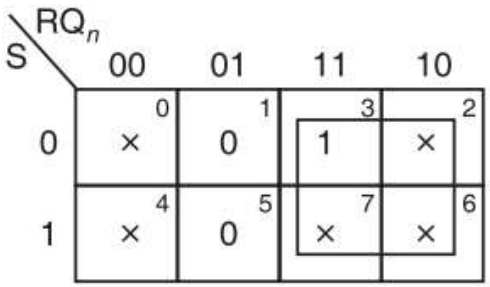
J-K flip-flop to S-R flip-flop:

External Inputs		Present State	Next State	Flip-flop Inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	×	1
1	0	0	1	1	×
1	0	1	1	×	0

(a) Conversion table

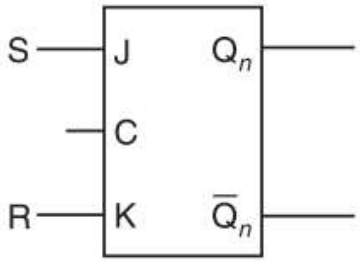


J = S



K = R

(b) K-maps for J and K

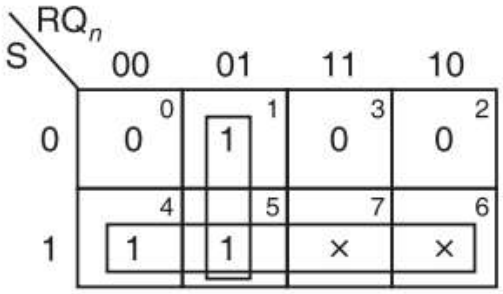


(c) Logic diagram

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

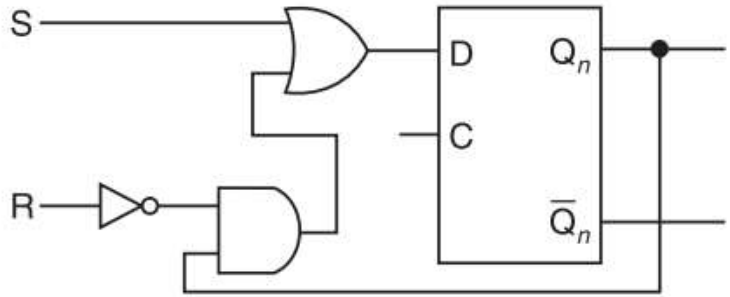
External Inputs		Present State	Next State	Flip-flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1

(a) Conversion table



$D = S + \overline{R}Q_n$

(b) K-map for D



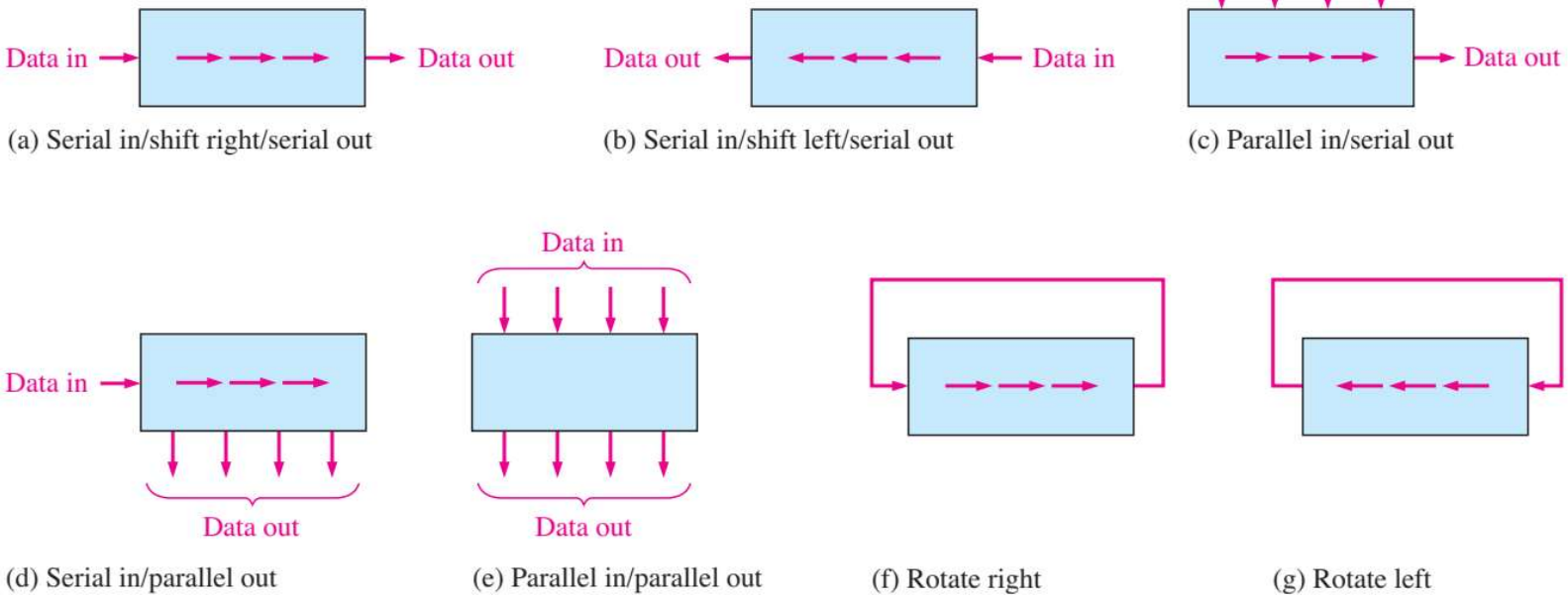
(c) Logic diagram

Shift Registers:

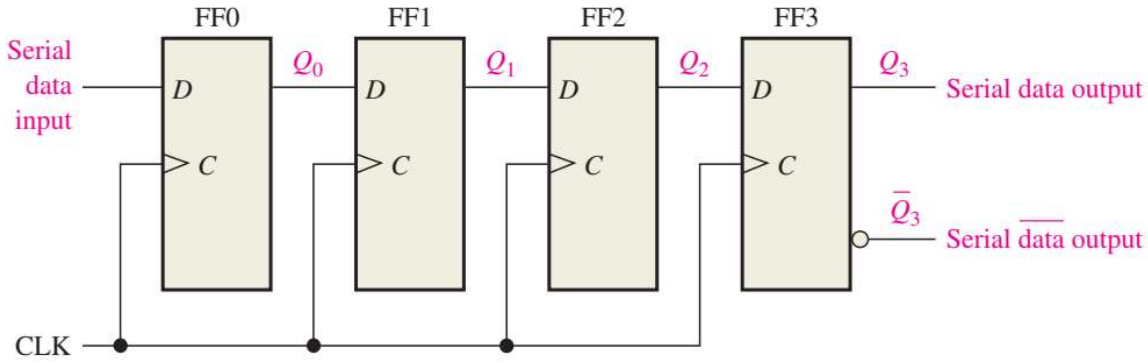
- A register is a group of binary cells (flip-flops) suitable for holding binary information, since each flip-flop is capable of storing 1 bit of information.
- An n-bit register has a group of n flip-flops and is capable of storing any binary information containing n bits.
- In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.
- The flip-flops hold binary information and the gates control when and how new information is transferred into the register.
- A register capable of shifting its binary information either to the right or to the left is called a ***shift register***.

- The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive a common clock pulse that causes the shift from one stage to the next.

Basic Movement of Data in Shift Registers:



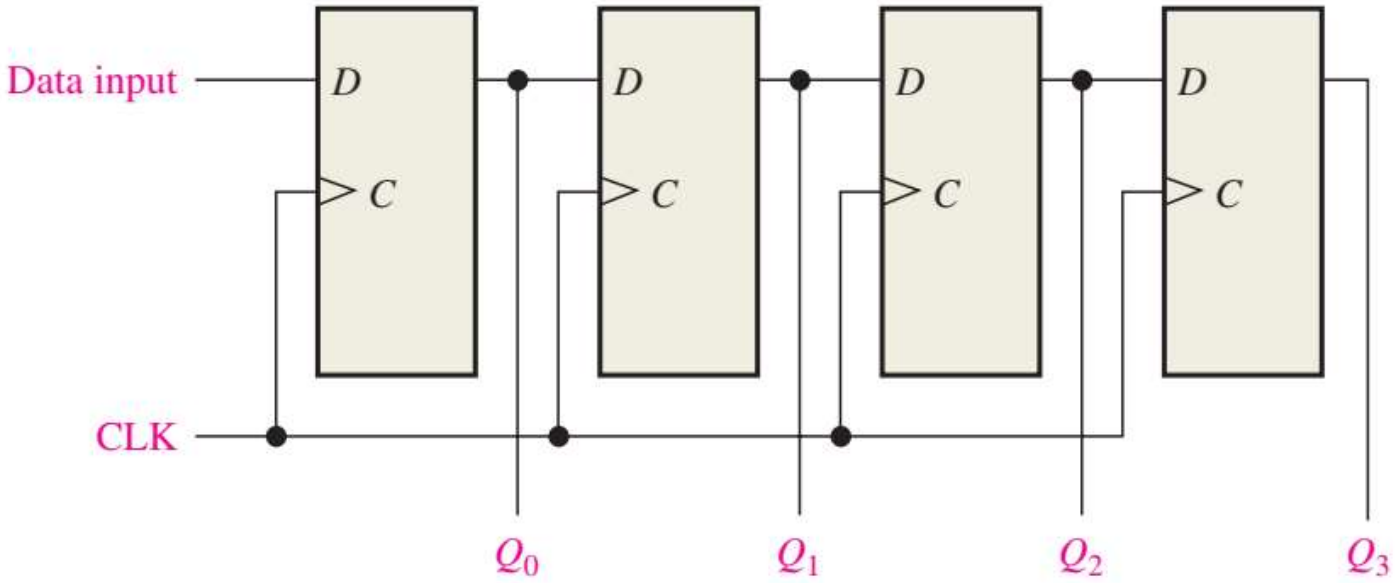
Serial Input – Serial Output (SISO) Shift Register



To store binary data "0101" the shifting proceeds as,

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	0	0	0	0
1	**0**	0	0	0
2	**1**	**0**	0	0
3	**0**	**1**	**0**	0
4	**1**	**0**	**1**	**0**

Serial In/Parallel Out Shift Registers



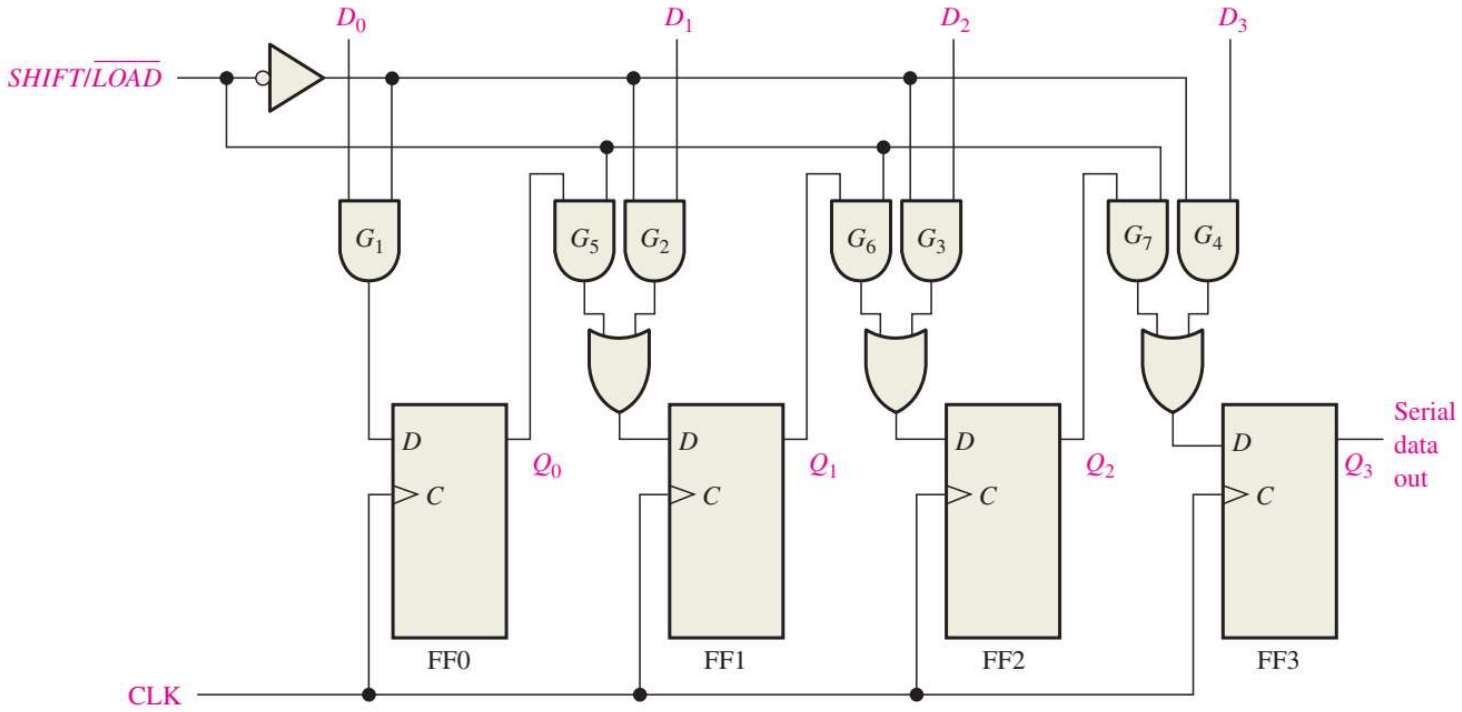
- **D flip-flop Based SISO/SIPO shift register- Behavioral Modelling**

```
module SISO(input Din,clock,reset, output Q0,Q1,Q2,Q3);
//SISO
DFF A(Q0,Din,clock,reset);
DFF B(Q1,Q0,clock,reset);
DFF C(Q2,Q1,clock,reset);
DFF D(Q3,Q2,clock,reset);
endmodule
```

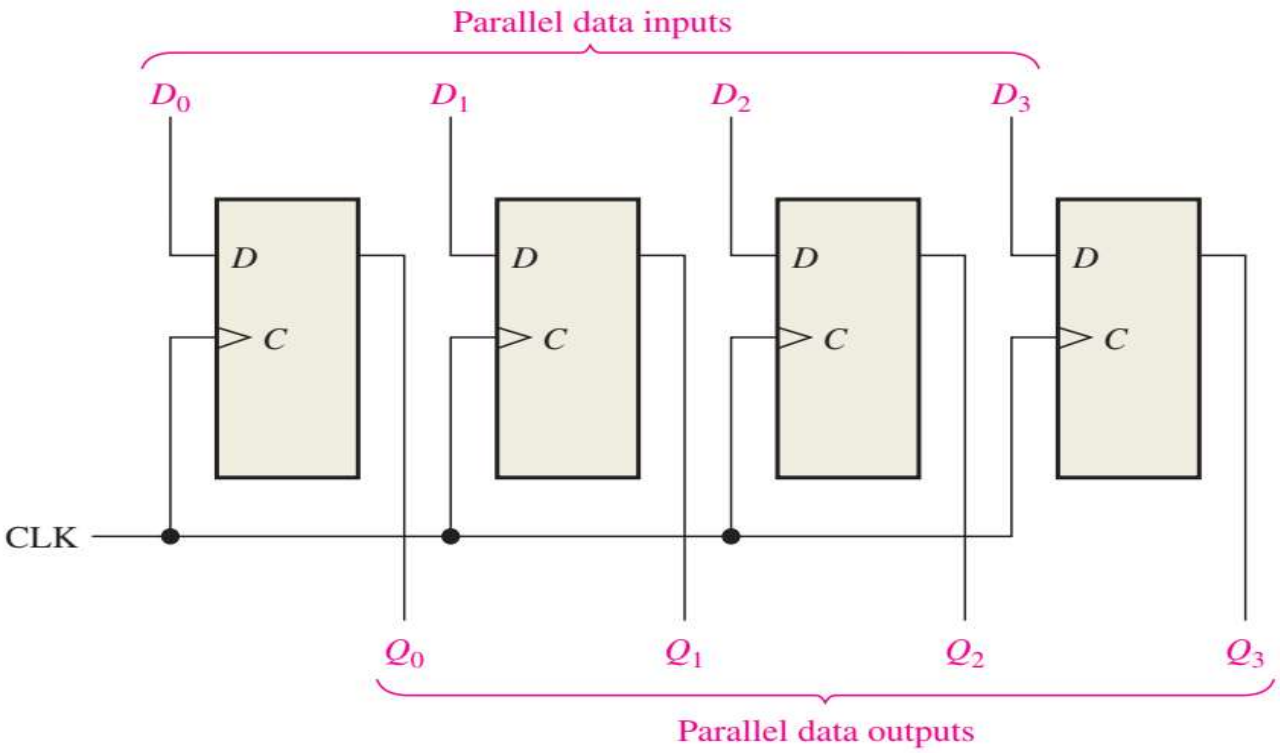
```
//D FLIP FLOP

module DFF ( output reg Q, input D, Clk, rst);

always @ ( posedge Clk, negedge rst)

if (!rst) Q <= 1'b0;

else Q <= D;

Endmodule
```

```
module siso_tb();
reg clk,din;
wire q3;
SISO uut(din,clk,rst,q0,q1,q2,q3);
initial begin clk=1'b0;
forever #5clk=~clk;
end
initial begin
din=1; #10; din=0; #10; din=1; #10; din=0; #50;
$finish;
end
endmodule
```

Parallel In/Serial Out Shift Registers

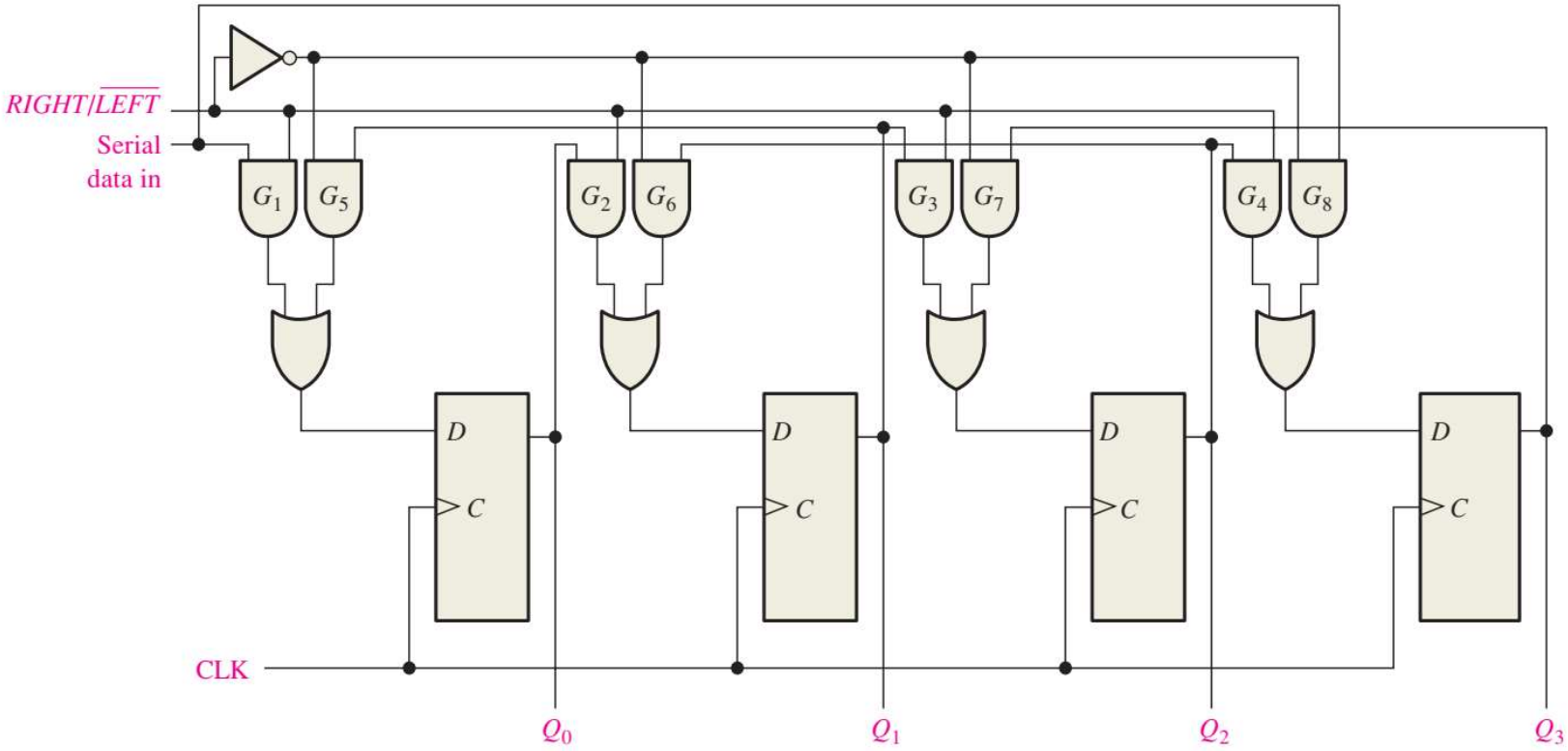


Parallel input – Parallel Output (PIPO) Shift Register



Four – Bit Bidirectional Shift Register

A **bidirectional** shift register is one in which the data can be shifted either left or right. It can be implemented by using gating logic that enables the transfer of a data bit from one stage to the next stage to the right or to the left, depending on the level of a control line.



Counters:

- A counter is a set of Flip-Flops whose states change in response to the pulse applied at the input to the counter.

- The flip-flops are interconnected in such a way that their combined state at any time is the binary equivalent of the total no. of pulses that have been occurred up to that time.
- Thus the counter is used to count the pulses.

Counters Classifications:

a) Asynchronous counters (also known as, Ripple counters/serial counters):

The flip-flop within the counters are not made to change the states at exactly the same time because the flip-flops are not triggered simultaneously. The clock doesn't directly control the time at which every stage changes its states. It uses T- flip-flops to perform the counting function. The actual hardware involved is th J-K flip-flop connected in toggle mode (J=K=1). It is to note that a D-flip-flop can also be used.

(b) Synchronous counters:

These are clocked in such a way that each flip-flop in the counter is triggered at the same time. This is accomplished by connecting the clock line to each stage of the counter they are faster than asynchronous counter as the propagation delay is less.

Points to be noted :

1. Each count of counter is called state of the counter.
2. The modulus of the counter is equal to the total no. of distinct states (counts) including the zero that the counter can store. In other words, the no. of input pulses that causes the counter to RESET to its initial count is called the modulus of the counter.

   e.g.: 2-bit counter ➔ $2^2 = 4$ ➔ (MOD-4) ➔ (Divide – by – 4)

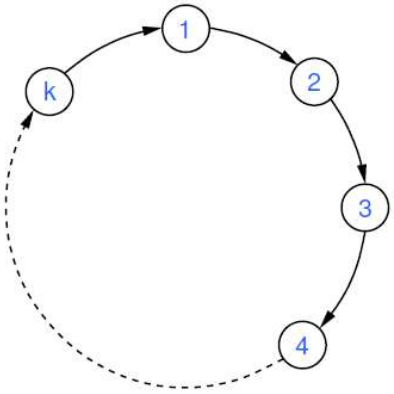
3. An n-bit counter will have n flip-flops and 2^n states and divides the input frequency by 2^n. Hence it is called as "divide-by-2^n" counter.
4. A counter may have shortened modulus. This type of counter does not utilize all the possible states. Some of the states are not utilized, i.e. remains invalid.

e.g.: MOD-6 counter ➔ Minimum no. of flip-flops required = 3 (as $2^3 = 8$)

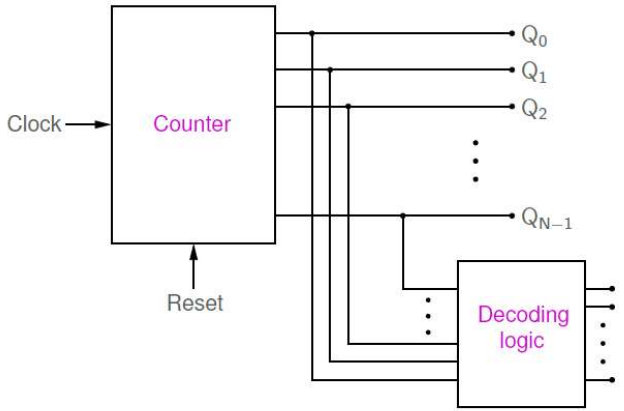
MOD-10 counter ➔ Minimum no. of flip-flops required = 4 (as $2^4 = 10$)

5. The no. of flip-flops required to construct the “MOD-N” counter equals to the smallest integer “n” for which $N \leq 2^n$
6. In an asynchronous counter invalid states are bypassed by providing suitable feedback, whereas, in synchronous counter the invalid states are taken care by treating the excitation as don’t care conditions.
7. The least significant bit (LSB) is the bit that changes most often. In ripple counters the LSB is the Q output of the flip-flop to which external CLOCK Pulse is applied.
8. The final state of the counter sequence is called the terminal count.

Counters



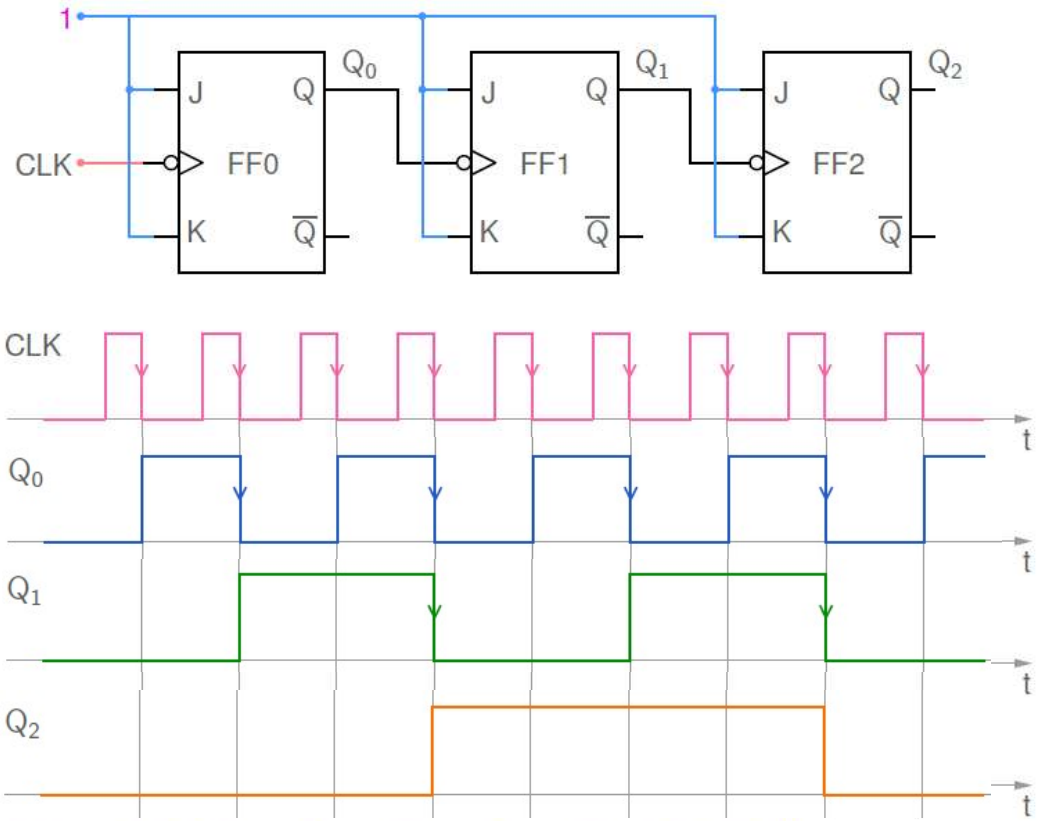
State transition diagram



General configuration

* A counter with k states is called a modulo-k (mod-k) counter.
* A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
* If there are N flip-flops in a counter, there are 2^N possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states.
  $\rightarrow$ N flip-flops can be used to make a mod-k counter with $k \leq 2^N$.
* Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.

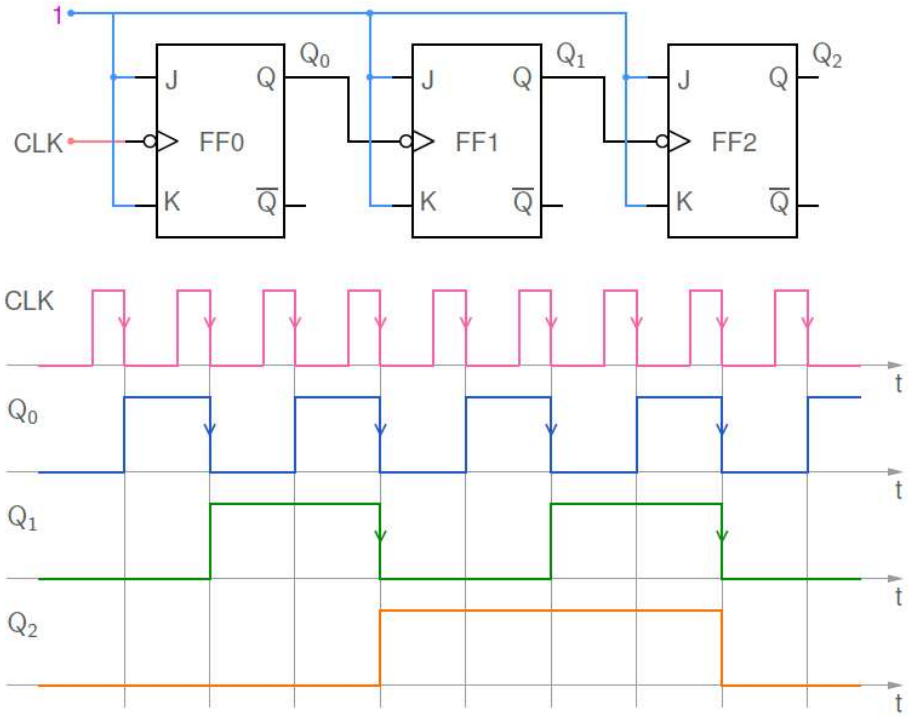
Binary Ripple counter(Asynchronous Counter)



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

repeats

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.



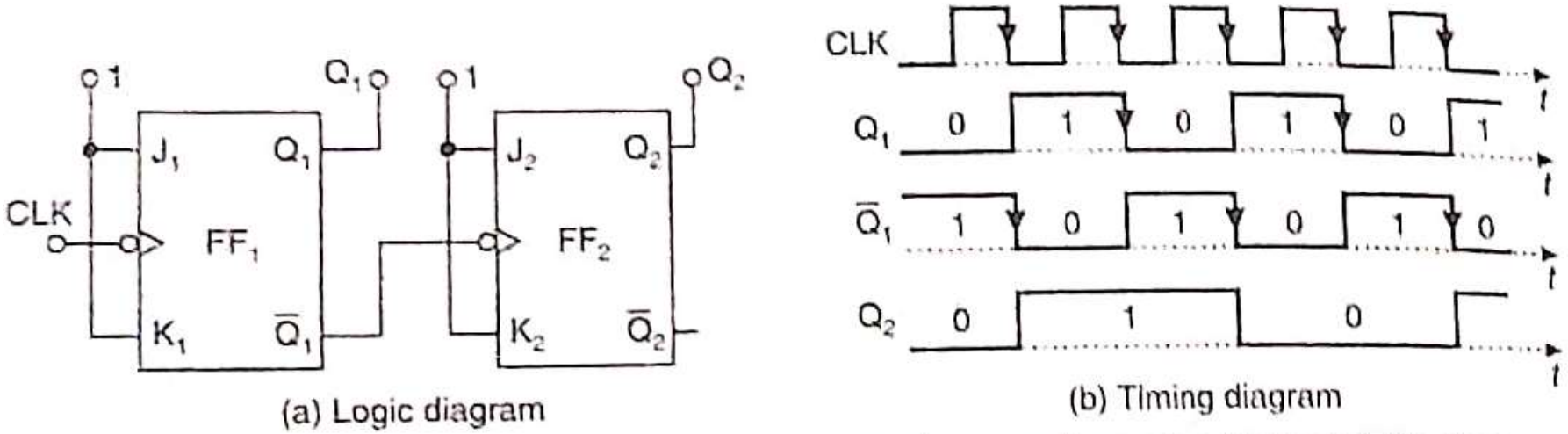
Q_2	Q_1	Q_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	
0	0	0	repeats

* The counter has 8 states, $Q_2 Q_1 Q_0 = 000$, 001, 010, 011, 100, 101, 110, 111.
  $\rightarrow$ it is a mod-8 counter. In particular, it is a *binary, mod-8, up* counter (since it counts *up* from 000 to 111).

* If the clock frequency is f_c, the frequency at the Q_0, Q_1, Q_2 outputs is $f_c/2$, $f_c/4$, $f_c/8$, respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.

* This type of counter is called a "ripple" counter since the clock transitions *ripple* through the flip-flops.

Asynchronous 2-bit down counter using Negative edge triggered Flip-Flops

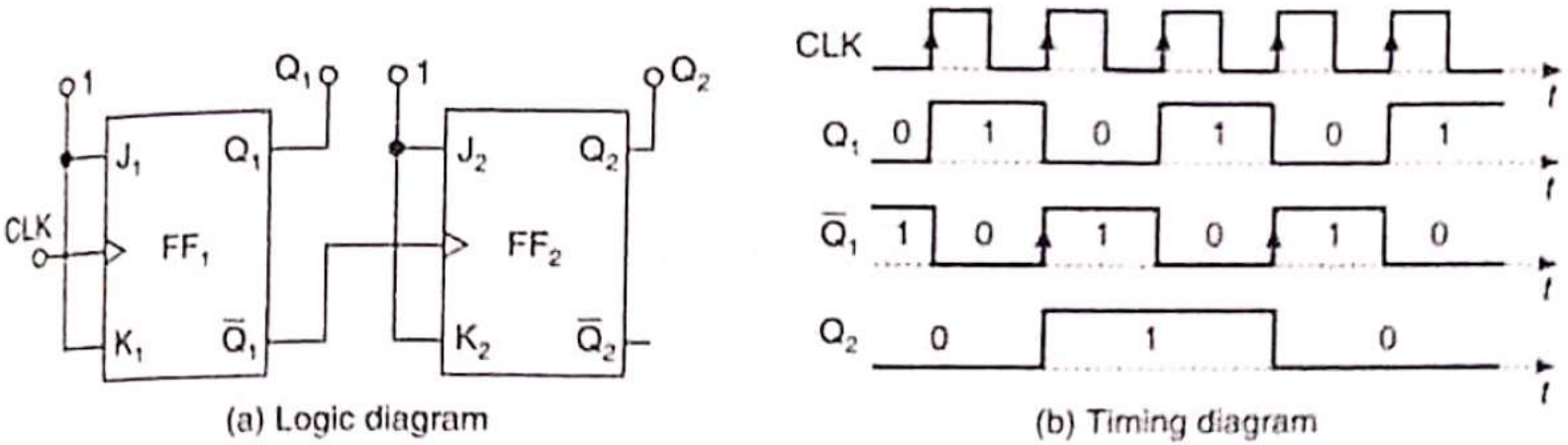


(a) Logic diagram

(b) Timing diagram

Asynchronous 2-bit down-counter using negative edge-triggered flip-flops.

2-bit Ripple Up-Counter Using Positive Edge triggered Flip-Flops

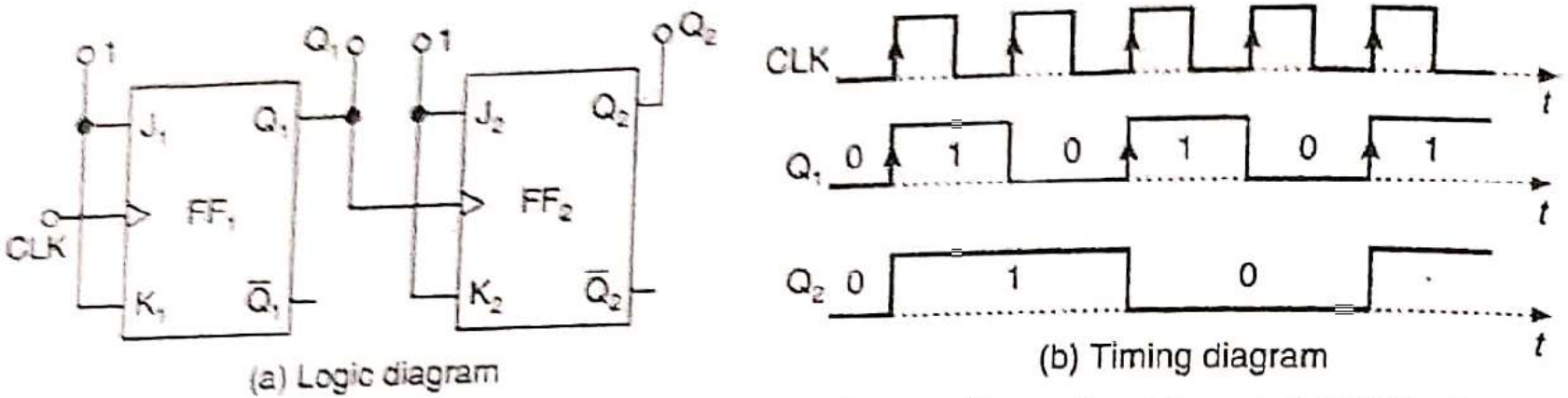


(a) Logic diagram

(b) Timing diagram

Asynchronous 2-bit up-counter using positive-edge triggered J-K flip-flops.

2-bit Ripple Down-Counter Using Positive Edge triggered Flip-Flops

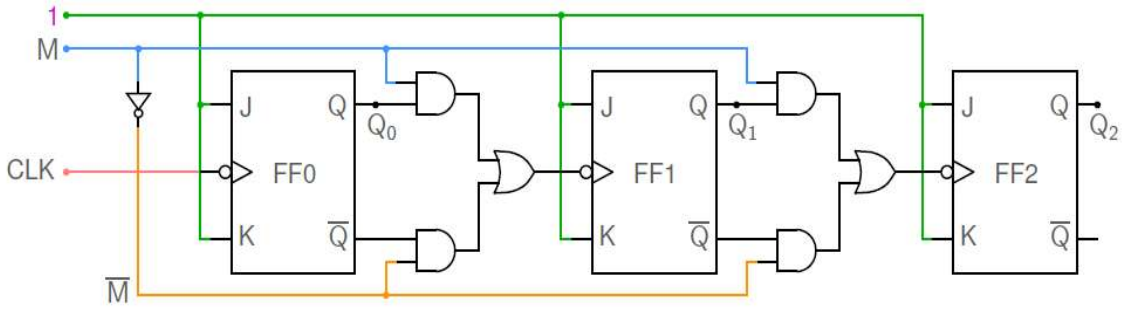


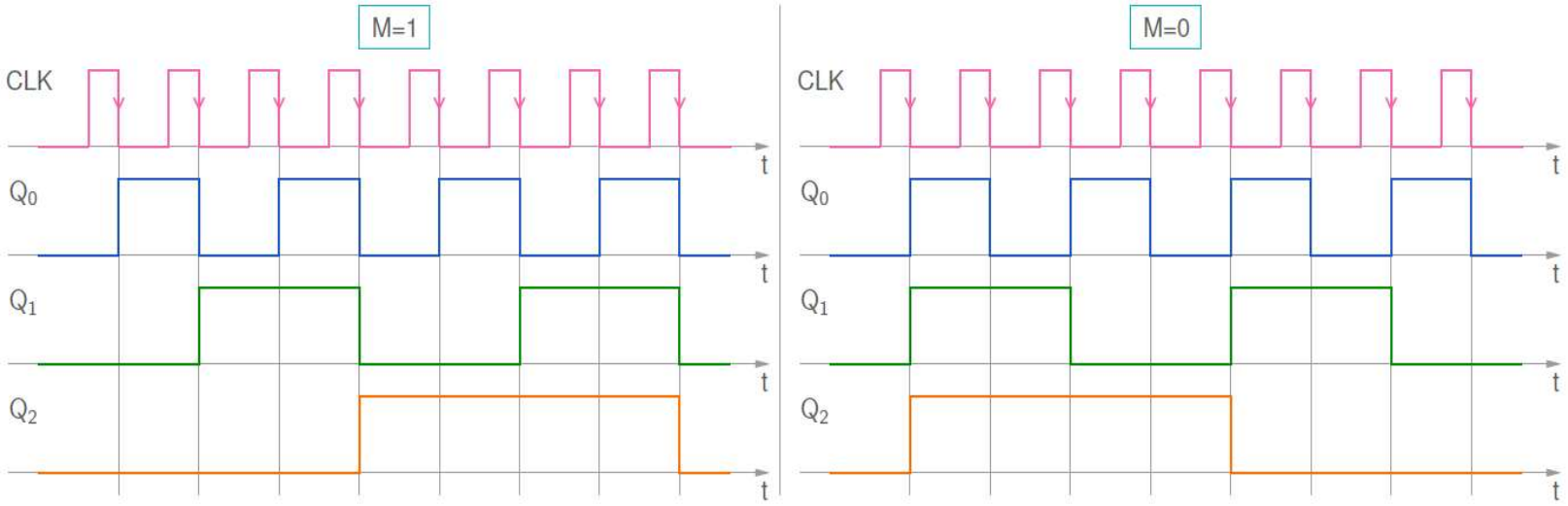
(a) Logic diagram

(b) Timing diagram

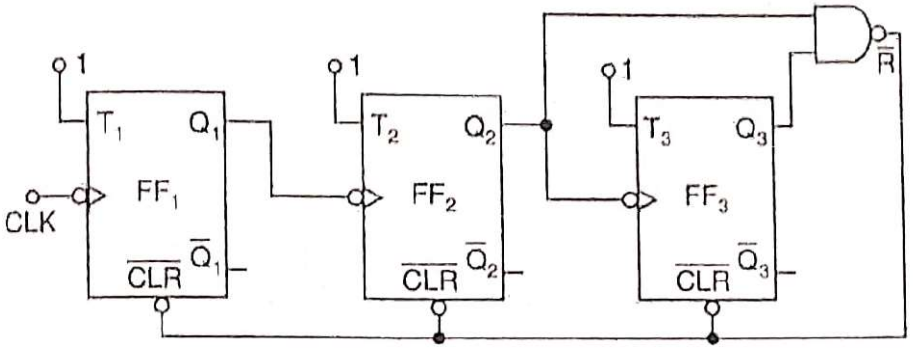
Asynchronous 2-bit down-counter using positive edge-triggered J-K flip-flops.

3-bit Up-Down binary ripple counter

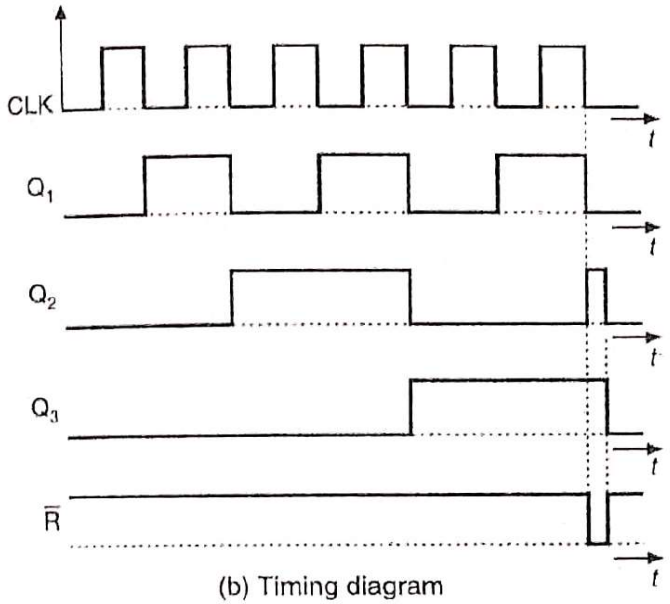




Design of a Mod-6 Asynchronous Counter using T- FFs



(a) Logic diagram

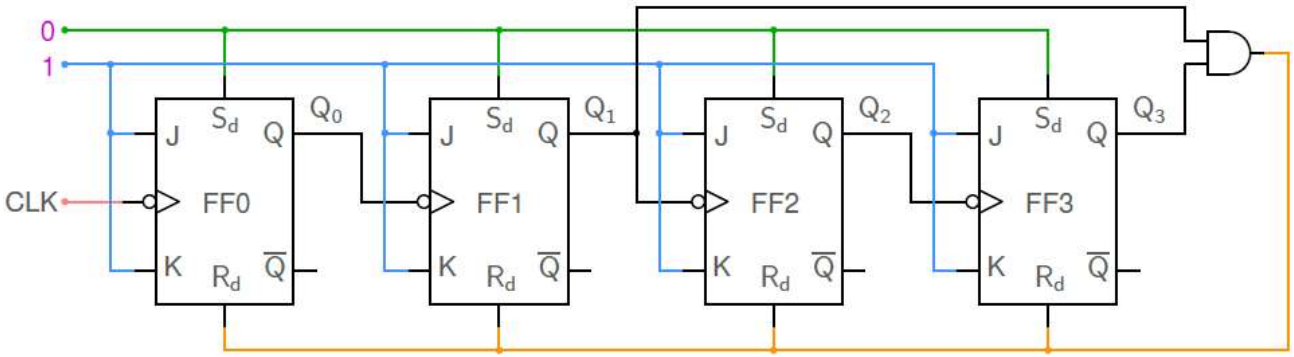


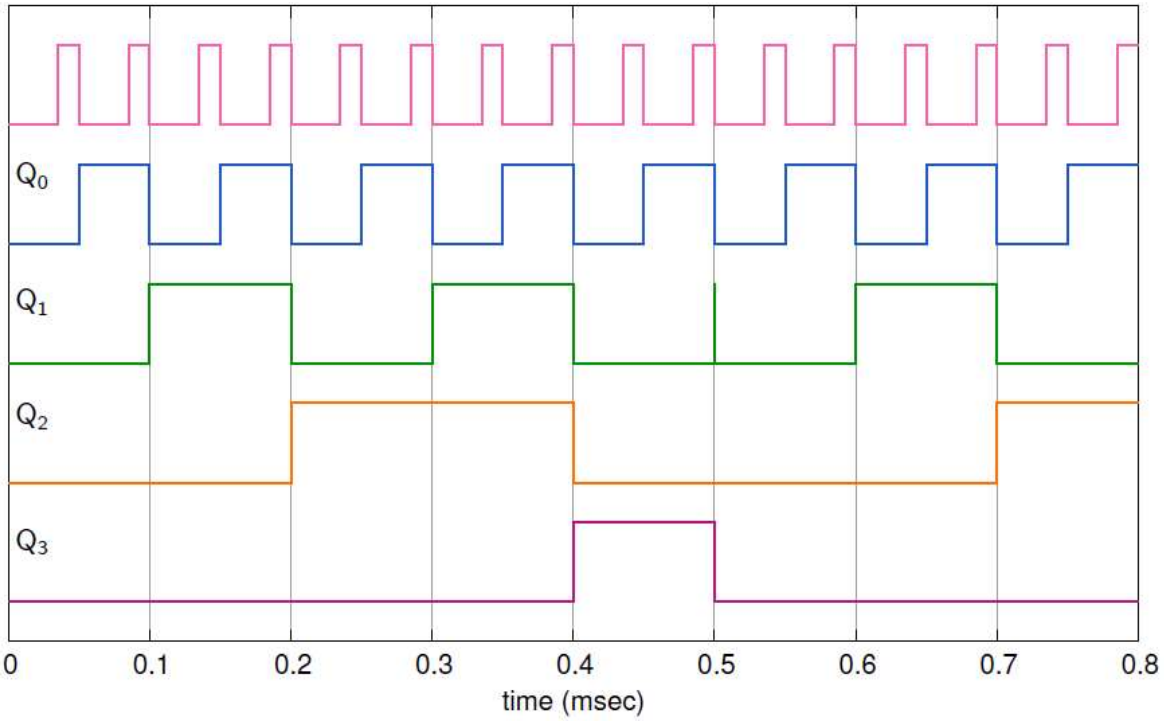
(b) Timing diagram

After pulses	State Q_3	Q_2	Q_1	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	1	0

(c) Table for R

Decade counter using direct inputs

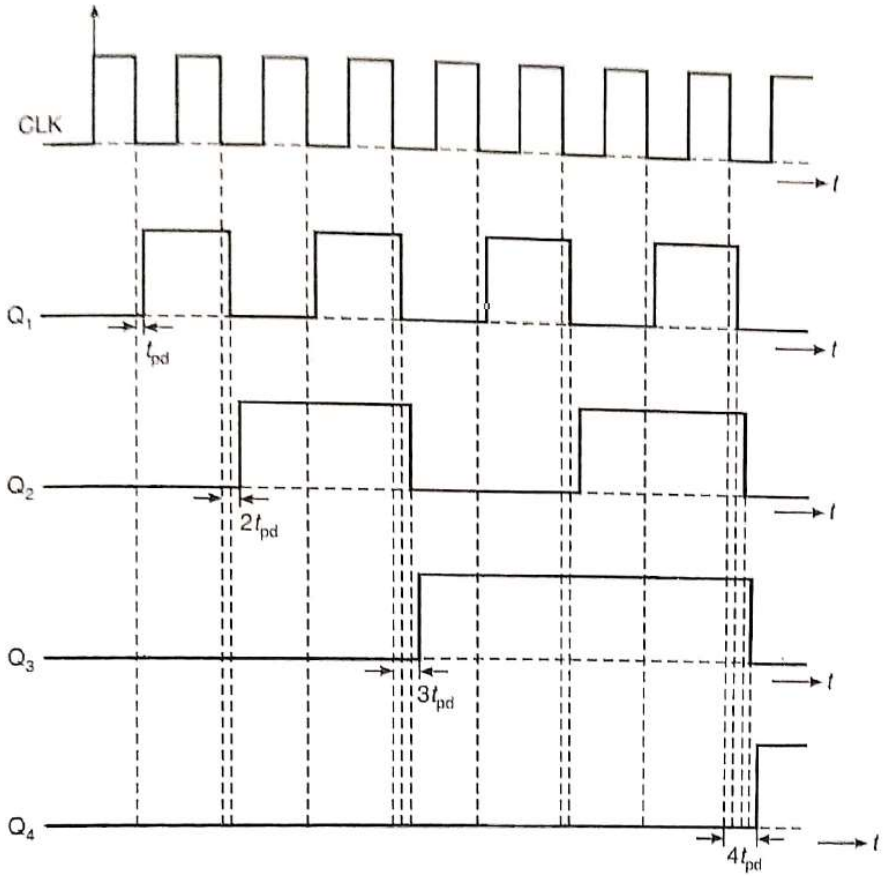




Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

repeats

Effects of Propagation Delay in Ripple counters



Timing diagram considering propagation delay (no skipping of states).