CHAPTER – 8

REGISTERS, COUNTERS AND MEMORY UNIT

8.1 Registers

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

8.2 Shift Registers

A register capable of shifting its binary information in one or more direction is called shift register.

Some of the important applications of shift registers are:

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

8.3 Superposition of Registers

8.3.1 Types of Shift Registers

There are four types of shift registers:

i. Serial in serial out (SISO)

ii. Serial in parallel out (SIPO)

iii. Parallel in serial out (PISO)

iv. Parallel in parallel out (PIPO)

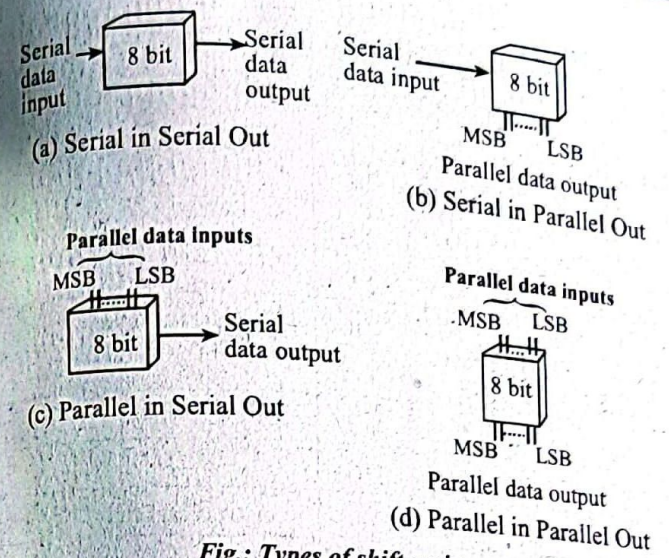


Fig.: Types of shift registers

1. Serial in Serial Out (SISO) Shift Register

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

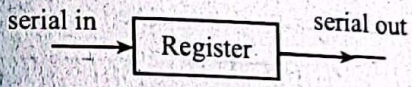


Fig.: Block diagram of right shift register

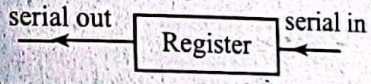


Fig.: Block diagram of left shift register

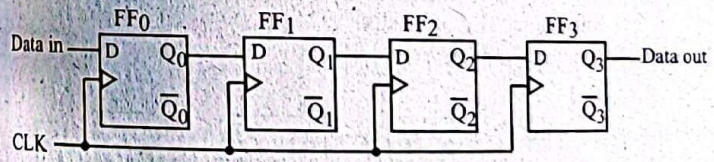


Fig.: Logic diagram of serial in serial out shift register.

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

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

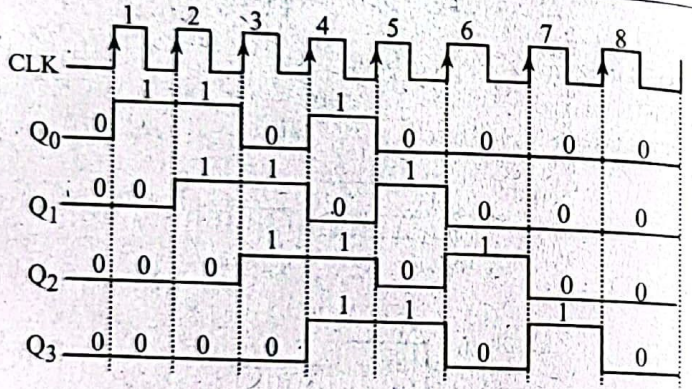


Fig.: Timing diagram for storing 1011

2. Serial in Parallel Out (SIPO) Shift Register

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

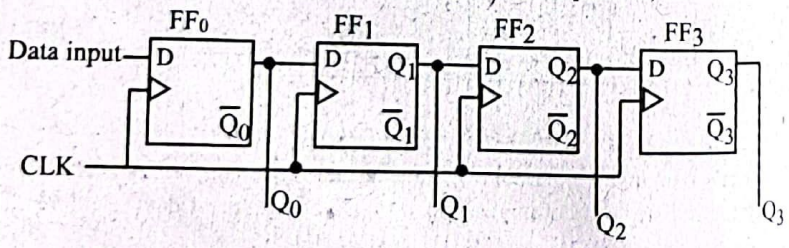


Fig.: Logic diagram of serial in parallel out (SIPO) shift register

Let 1010 is stored from rightmost bit.

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

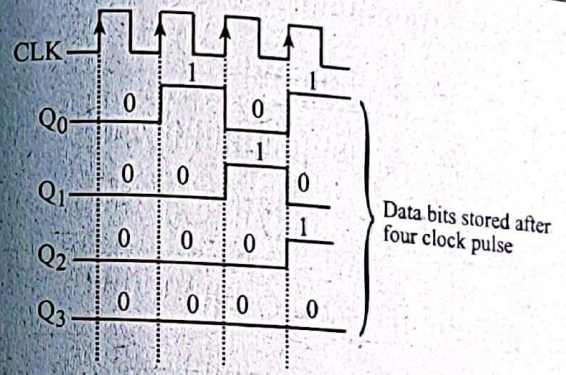


Fig.: Timing diagram for storing 1010 in SIPO shift register

3. Parallel in Parallel Out (PIPO) Shift Register

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

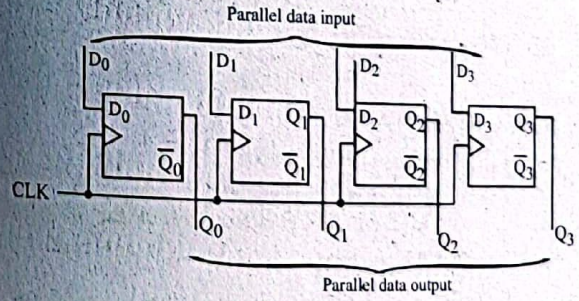


Fig.: Logic diagram of 4-bit parallel in parallel out (PIPO) shift register.

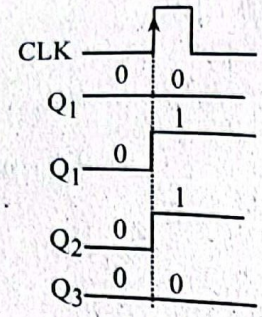


Fig.: Timing diagram for storing 0110 in PIPO shift register

4. Parallel in Serial Out (PISO) Shift Register

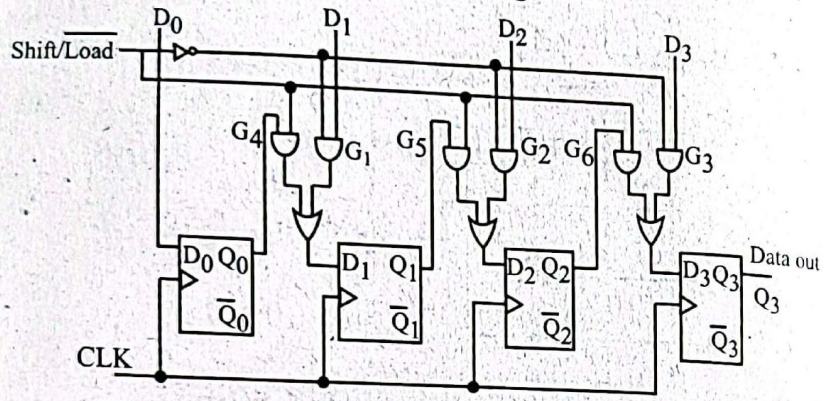


Fig.: Logic diagram of parallel in serial out (PISO) shift register

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

Let's consider input data as

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

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

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

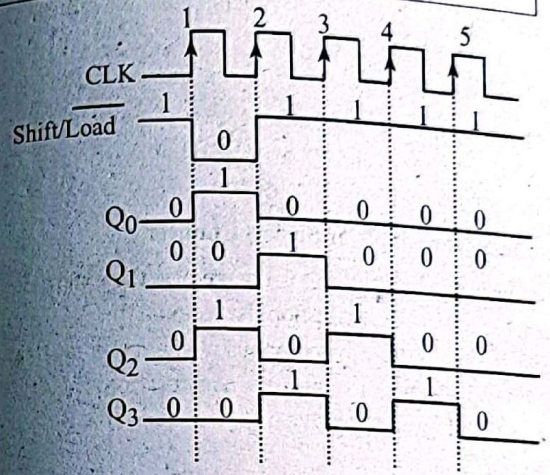


Fig.: Timing diagram for storing 1010 in PISO shift register

8.3.2 Bidirectional Shift Register (Universal Register)

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

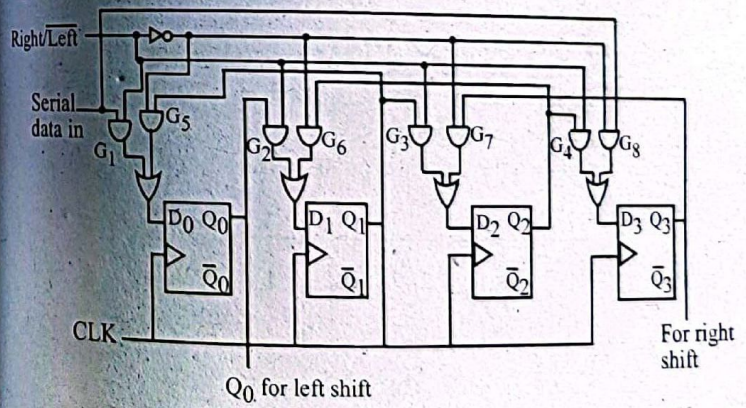


Fig.: Logic diagram of bidirectional shift register.

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

8.4 Counter

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

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

i. Asynchronous (or ripple or serial) counter
ii. Synchronous (or parallel) counter

1. Asynchronous (or Ripple or Serial) Counter

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

2. Synchronous (or Parallel) Counter

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

8.4.1 Asynchronous Counters

1. 3-bit Ripple Up Counter (or MOD-8 Up Counter)

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

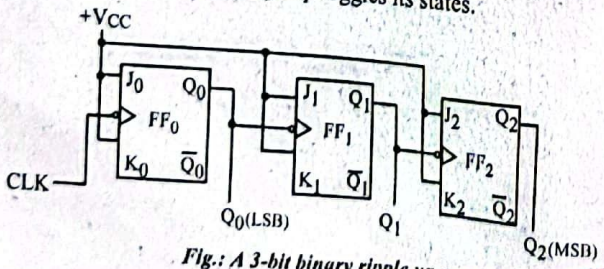


Fig.: A 3-bit binary ripple up counter

Truth table

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

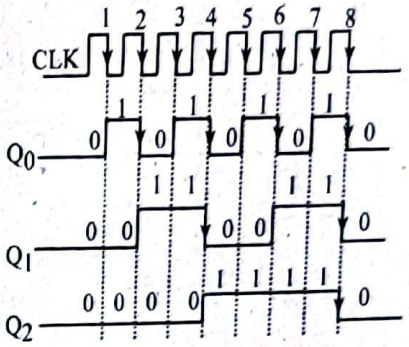


Fig.: Timing diagram

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

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

2. 3-bit Ripple Down Counter (MOD-8 Ripple Down Counter)

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

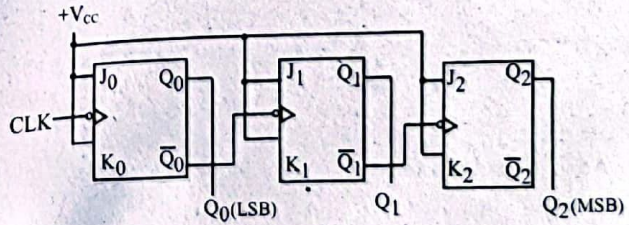


Fig.: 3-bit ripple down counter

Truth table

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

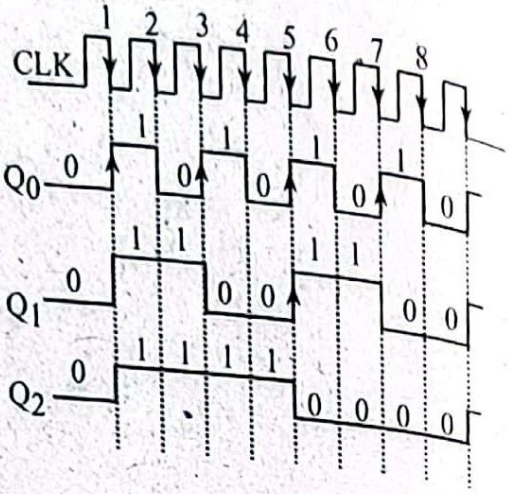


Fig.: Wavefoms

3. 3-bit Ripple Up/Down Counter

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

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

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

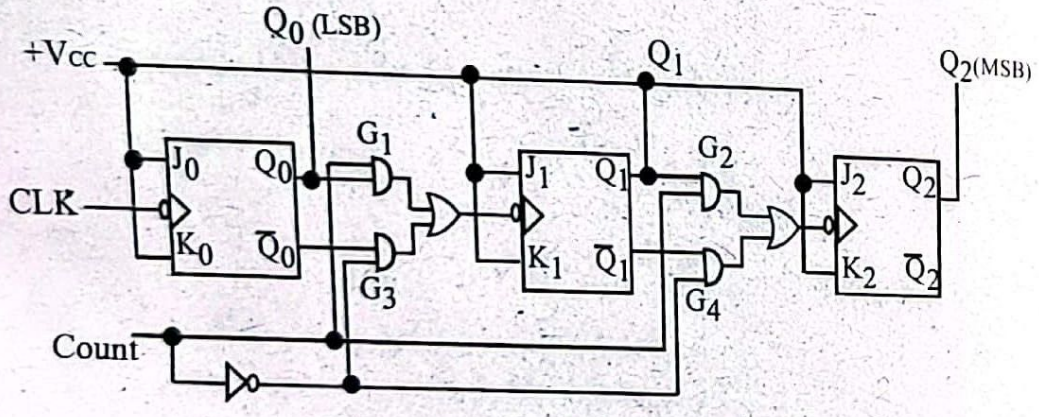


Fig.: 3-bit up/down ripple counter

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

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

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

8.4.2 Synchronous Counter

1. 3-bit Synchronous Up Counter (or MOD-8 Synchronous Up Counter)

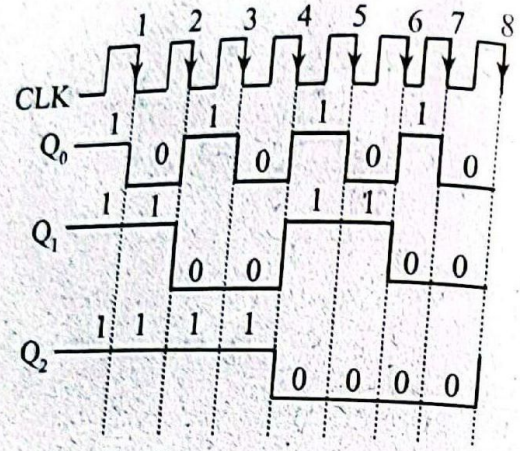


Fig.: Waveforms

3. 3-bit Synchronous Up/Down Counter

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

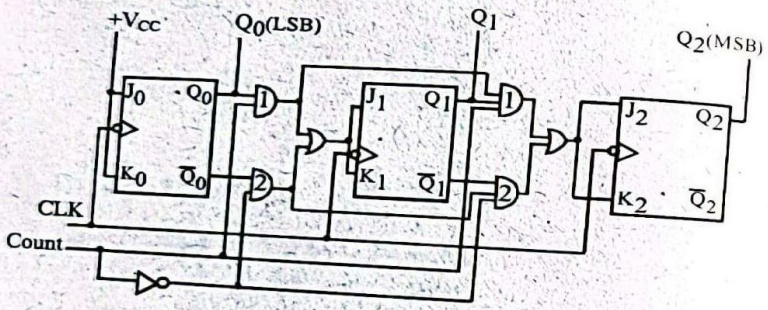


Fig.: 3-bit synchronous up/down counter

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

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

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

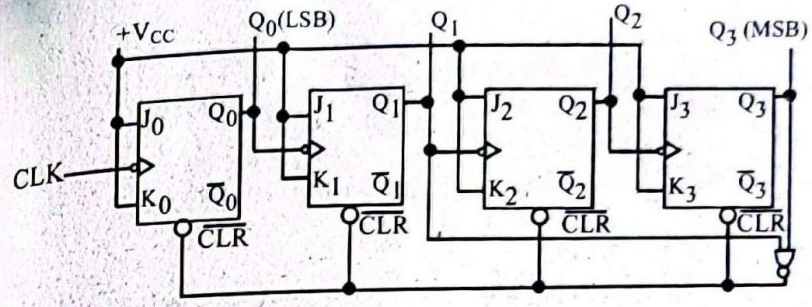


Fig.: Asynchronous decade counter

8.6. Application of Shift Register

Shift registers are used in almost every sphere of a digital logic system. Shift register can be used to count number of pulses entering into a system as ring counter or switched-tail counter. As ring counter, it can generate various control signals in a sequential manner. Shift register can also generate a prescribed sequence of repetitively or detect a particular sequence from data input. It can also help in reduction of hardware by converting parallel data feed to serial one.

1. Ring Counter

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

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

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

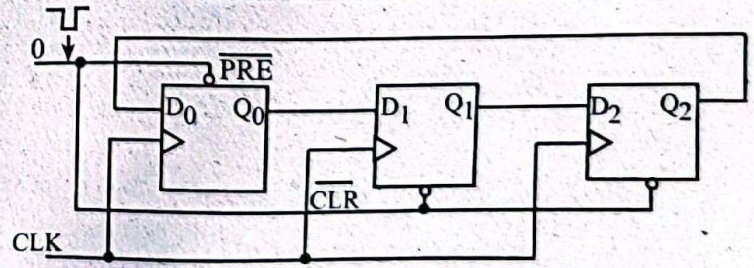


Fig.: Logic diagram of a ring counter

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

Drawbacks:

The requirement of initialization is a disadvantage of ring counter.

2. Johnson Counter (Switched-Tail Counter)

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

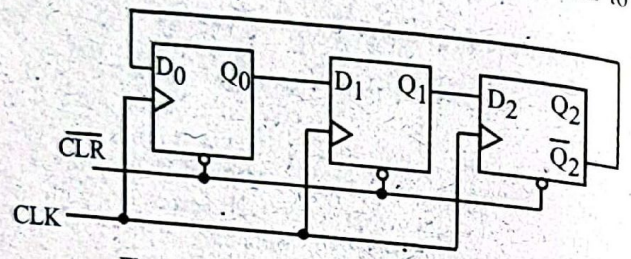


Fig.: Logic diagram of Johnson counter

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

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

8.7 Design of Multiple Input Circuits

Following steps are to be followed for designing synchronous counters:

i. Draw the state diagram from the given word description problem.

ii. Draw the next state table and find number of flipflops required. No. of flipflops = no. of bits used in counter.

iii. Decide the type of flipflop to be used for the design, then determine the flipflop excitation table based on transition of present state (Q_n) to next state (Q_{n+1}).

iv. Prepare K-map for each flipflop input and simplify.

v. Connect the circuit using flipflop and other gates according to the minimized expression.

8.8 Random Access Memory (RAM)

8.8.1 The Memory Unit

A memory unit is a collection of storage registers together with the associated circuits needed to transfer information in and out of the registers. The storage registers in a memory unit are called memory registers.

The component that forms the binary cells of resisters in a memory unit must have certain basic properties, the most important of which are:

1. It must have a reliable two-state property for binary representation.
2. It must be small in size.
3. The cost per bit of storage should be as low as possible.
4. The time of access to a memory register should be reasonably fast.

Examples: Magnetic cores, semiconductor ICS, drums disks etc.

A memory unit stores binary information in groups called words each word being stored in a memory register.

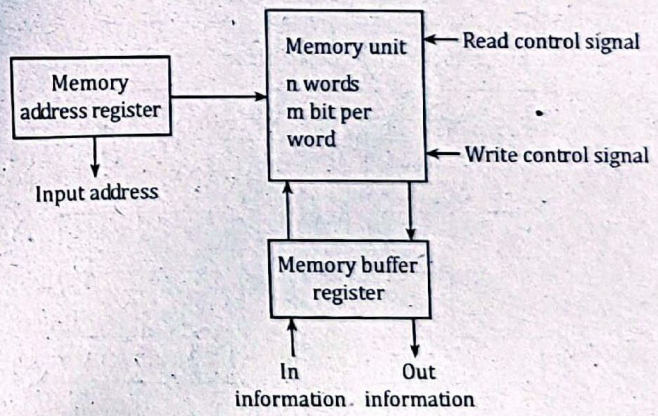


Fig.: Block diagram of a memory unit showing communication with environment

Memory address register:

The memory address register specifies the memory word selected. To communicate with a specific memory work, its location number, or address is transferred to the address resister. An address resister with n bits can specify up to 2^n memory words.

Control signal:

The two control signals applied to the memory unit are called read and write. A write signal specifies a transfer-in function; a read control signal specifies a transfer-out function.

Memory buffer register:

The information transfer to and from register in memory and the external environment is communicated through one common register called the memory buffer register (information register or storage register).

Read and Write Operation

The sequence of operations needed to communicate with the memory unit for the purpose of transferring a word out to the MBR is:

1. Transfer the address bits of the selected word into MAR
2. Activate the read control unit

The sequence of operations needed to store a new word into memory is:

1. Transfer the address bits of the selected word into MAR
2. Transfer the data bits of the word into MBR
3. Activate the write control input

The data bits from MBR are stored in memory register.

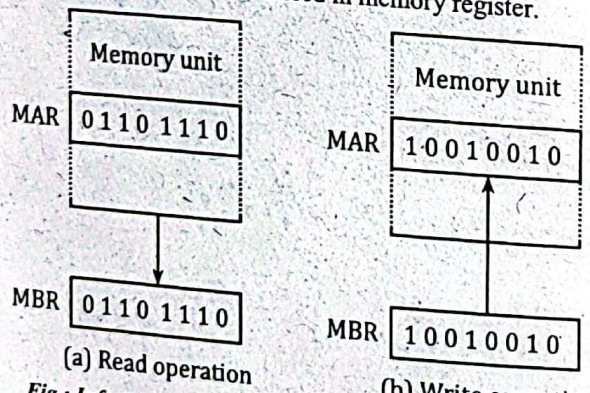
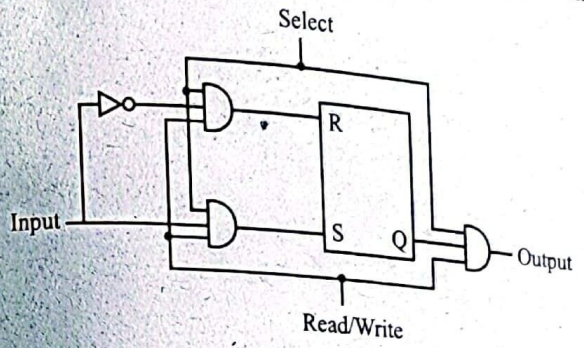


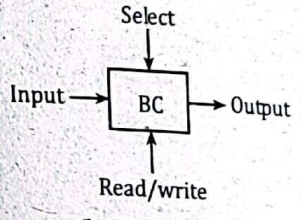
Fig.: Information transfer during read and write operations

8.8.2 Integrated Circuit Memory

The internal construction of a random-access memory of m words with n bits per word consists of m × n binary storage cells and the associated logic for selecting individual words. The binary storage cell is the basic building block of a memory unit.



(a) Logic diagram



(b) Block diagram

Fig.: Memory cell

IC RAMS are constructed internally with cells having a wired-OR capability. Figure shows a IC RAM which consists of 2 words of 3 bits each, for a total of 6 binary cells.

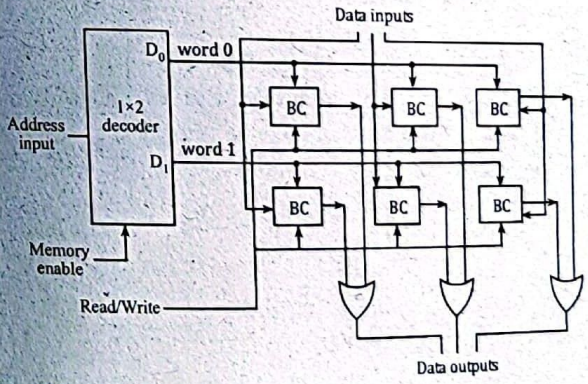


Fig.: Integrated circuit memory

8.9 Memory Decoding

- In a computer system, the physical address space is allocated to various memory chips and other subsystems.
- A microprocessor with n address line has a 2^n address memory space.
- Physical memory space is the actual amount of memory implemented in terms of memory modules (chips).
- Memory decoding places the block of memory in a module into a particular location in the address space.
- Partial address decoding uses only some of the remaining address lines to drive the CS.
- Full address decoding uses all of the remaining address lines to drive the CS signal.

8.10 Error Correction Code

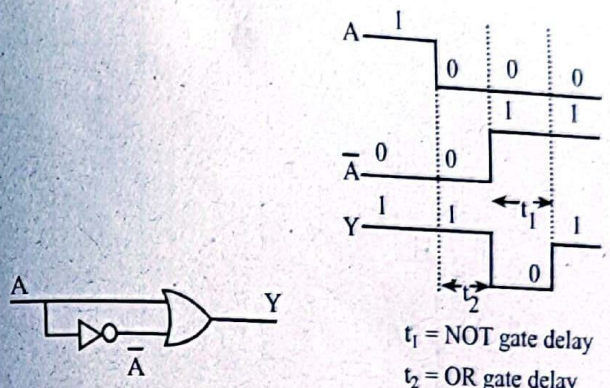
An error correcting code (ECC) or forward error correction (FEC) code is a process of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors [up to the capability of the code being used] were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for re-transmission of the data, a backchannel is not required in, forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks and RAM.

8.11 Output Hazards Races

In practical circuits, it always offer finite propagation delay though very small, in nanosecond order. This gives rise to several hazards and hazard recover are additional terms in an equation that prevents occurring of them.

1. Static-1 Hazard

This type of hazard occurs when $Y = A + \bar{A}$ type of situation appears for a logic circuit for certain other inputs and A makes transition from 1→0. An $A + \bar{A}$ condition should always generate 1 at the output i.e., static 1. But NOT gate output takes finite time to become 1 following 1→0 transition of A. Thus, for the OR gate there are two zeros appearing at its input for the small duration resulting a 0 at this output.



t_1 = NOT gate delay

t_2 = OR gate delay

EXAMPLE:

The K-map is given

A \ BC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y = \bar{A}C + AB$

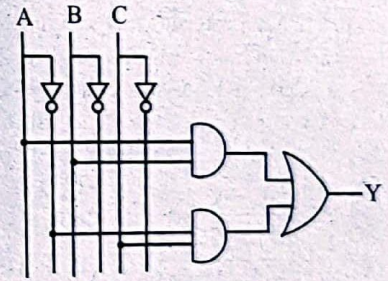


Fig.: Circuit with static-1 hazard

When B = 1, C= 1, and A makes transition from 1→ 0, static–1 hazard occurs. To remove it, additional BC term is added.

A \ BC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y = \bar{A}C + AB + BC$

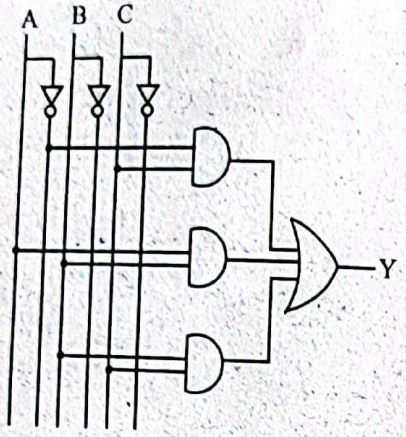
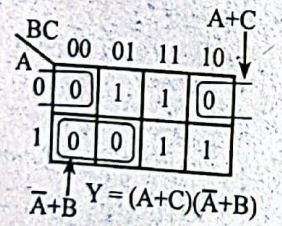


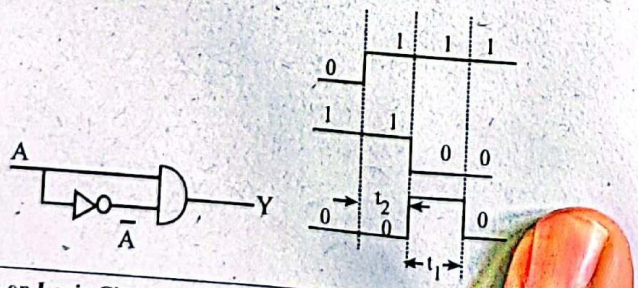
Fig.: Hazard free circuit

When, B = 1, C = 1, and A makes transition from 1 to 0, there is no effect upon output.

2. Static–0 Hazard

This type of hazard occurs when Y = A. $\bar{A}$ kind of situation occurs in a logic circuit for certain combination of other inputs and A makes transition from 0 → 1. A. $\bar{A}$ condition should always generate 0 at the output i.e., static 0. But NOT gate output takes finite time to become 0 following 0 → 1 transition of A. Thus, for final AND gate, there are two 1 appearing at its input for a small duration resulting 1 at its output. Such output may malfunction the sequential circuit.





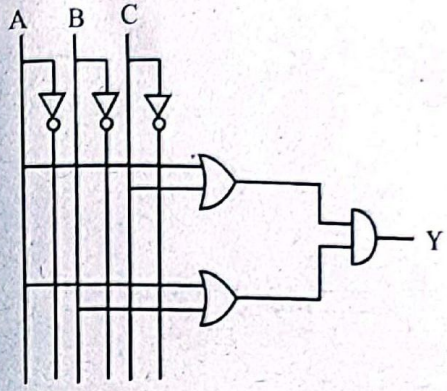
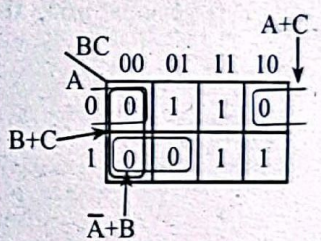


Fig.: Circuit with static–0 hazard

When B = 0, C = 0 and A makes transition from 0 →1, static–0 hazard will occur at output. To prevent this, we add additional term (B + C).



Y = (A+C)($\bar{A}$+B)(B+C)

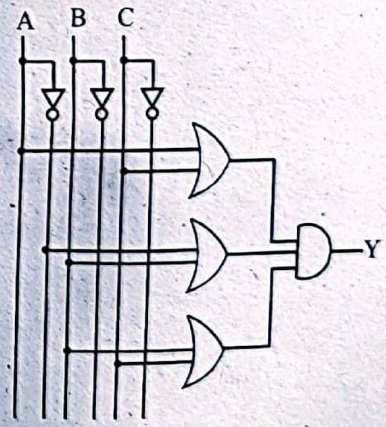


Fig.: Hazard free circuit

When B = 0, C = 0 and C tends 0 → 1, there is no change in output Y = 0

3. Dynamic Hazard

Dynamic hazard occurs when circuit output makes multiple transitions before it settles to a final value while the logic equation asks for only one transition. It occurs when

$Y = A + \bar{A}.A$ **OR**

$Y = (A + \bar{A}).A$

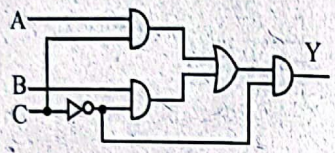


Fig.: For illustrating dynamic hazard

SOLUTION TO IMPORTANT AND EXAM QUESTIONS

1. Design and implement MOD-5 counter using JK flipflop.

[Fall 2020]

Solution:

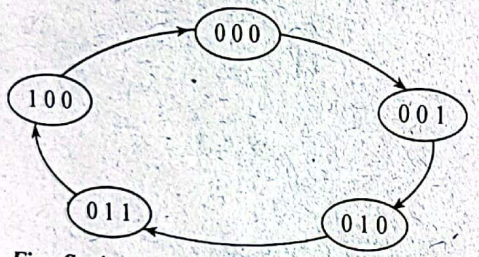


Fig.: Static diagram of Mod-5 binary up counter

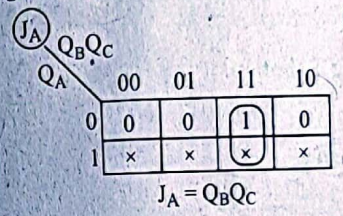
Excitation table of JK flipflop

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

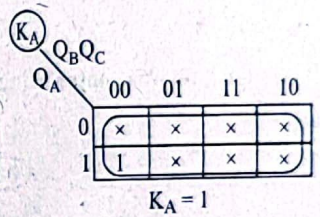
Synthesis Table

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

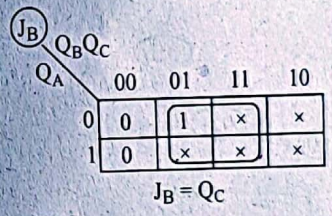
Using K-maps



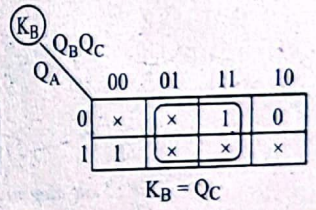
$J_A = Q_BQ_C$



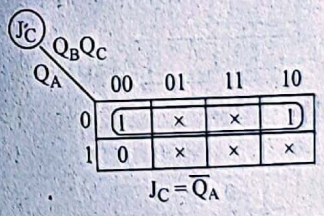
$K_A = 1$



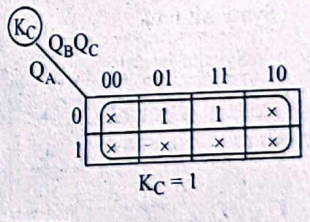
$J_B = Q_C$



$K_B = Q_C$



$J_C = \bar{Q}_A$



$K_C = 1$

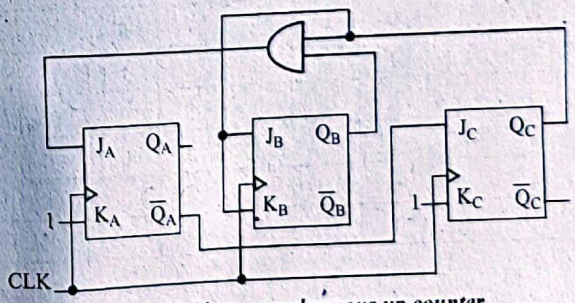


Fig.: 3-bit binary synchronous up counter

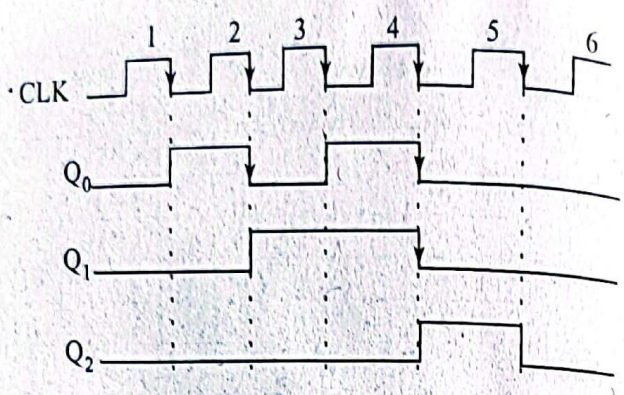


Fig.: Timing diagram

2. Design 3-bit up counter using T flipflop. [Spring 2019]

Solution:

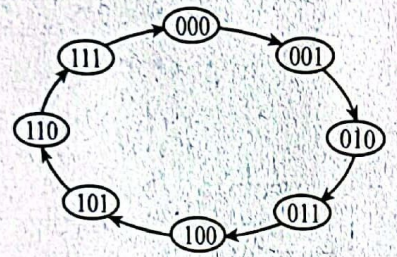


Fig.: State diagram of 3-bit up counter

Synthesis table

Synthesis state			Next state			FF Excitation		
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Using K-map,

For T_2,

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

$\therefore \quad T_2 = Q_{1n}Q_{0n}$

For T_1,

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

$\therefore \quad T_1 = Q_{0n}$

For T_0,

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

$\therefore \quad T_0 = 1$

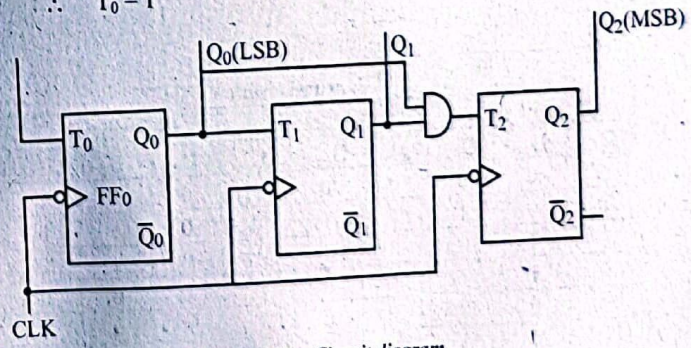


Fig.: Circuit diagram

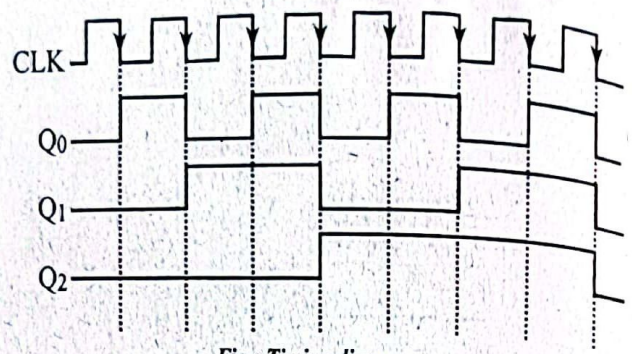


Fig.: Timing diagram

3. Design a synchronous mod-6 counter using clocked D flipflop. [Spring 2018]

Solution:

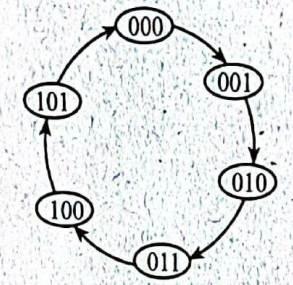


Fig.: State diagram for mod-6 up counter

Excitation table of D flipflop

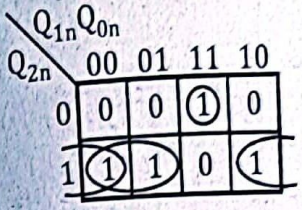
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Synthesis table

Present state			Next state			FF Excitation		
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	D_2	D_1	D_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Using K-map,

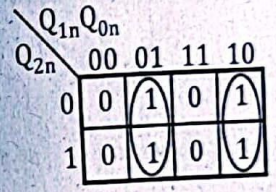
For D_2,



$$\therefore \quad D_2 = \bar{Q}_{2n}Q_{1n}Q_{0n} + Q_{2n}\bar{Q}_{0n} + Q_{2n}\bar{Q}_{1n}$$

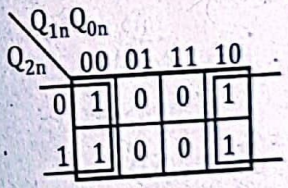
$$= \bar{Q}_{2n}Q_{1n}Q_{0n} + Q_{2n}(\bar{Q}_{0n} + \bar{Q}_{1n})$$

For D_1,



$$\therefore \quad D_1 = \bar{Q}_{1n}Q_{0n} + Q_{1n}\bar{Q}_{0n} = Q_{1n} \oplus Q_{0n}$$

For D_0,



$$\therefore \quad D_0 = \bar{Q}_{0n}$$

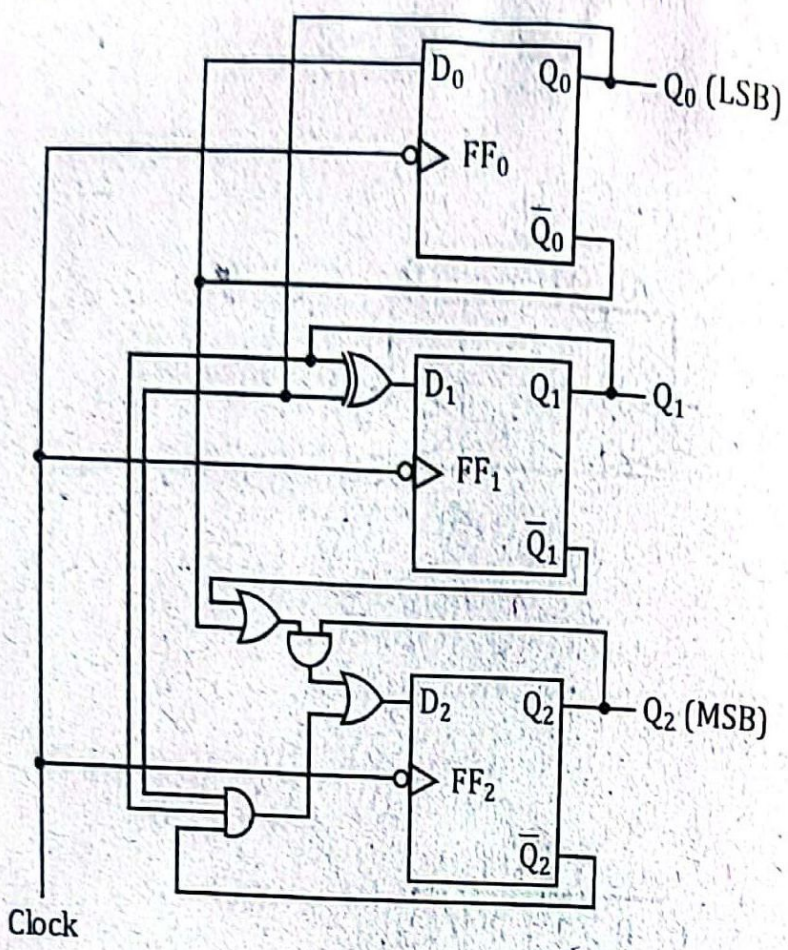


Fig.: Circuit diagram of MOD-6 up counter using clocked D flipflop

4. Design a 3-bit synchronous DOWN counter using T flipflop. [Spring 2018]

Solution:

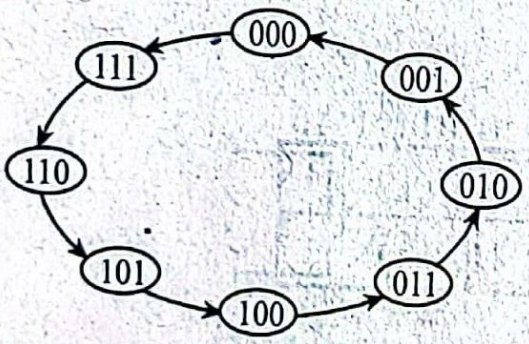


Fig.: State diagram for 3-bit DOWN counter

Excitation table of T flipflop

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

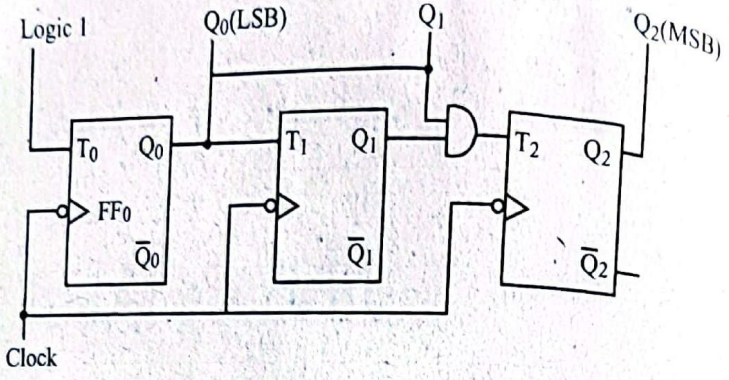
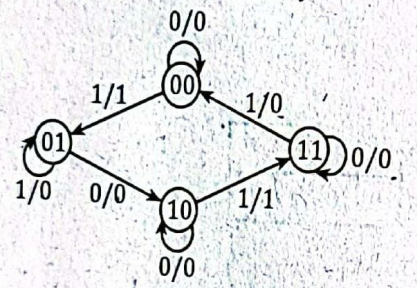


Fig.: Circuit diagram of 3-bit synchronous down counter

5. **Design a sequential circuit corresponding to the given state diagram using SR flipflop for the following state diagram.**

[Fall 2018]



Solution:

Excitation table of SR flipflop

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

Synthesis table

Present state		Present input (X)	Next state		Output (Y)	FF Excitation			
Q_{1n}	Q_{0n}		Q_{1n+1}	Q_{0n+1}		S_1	R_1	S_0	R_0
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	1	0	×	1	0
0	1	0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	×	×	0
1	0	0	1	0	0	×	0	0	×
1	0	1	1	1	1	×	0	1	0
1	1	0	1	1	0	×	0	×	0
1	1	1	0	0	0	0	1	0	1

Using K-map,

For S_1,

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

$\therefore \quad S_1 = Q_{1n}\bar{X}$

For R_1,

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

$\therefore \quad R_1 = Q_{1n}X$

For S_0,

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

$\therefore \quad S_0 = \bar{Q}_{1n}X$

For R_0,

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

$\therefore \quad R_0 = Q_{0n}Q_{1n}X + \bar{Q}_{0n}\bar{X}$

For output Y,

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

$\therefore \quad Y = \bar{Q}_{1n}X$

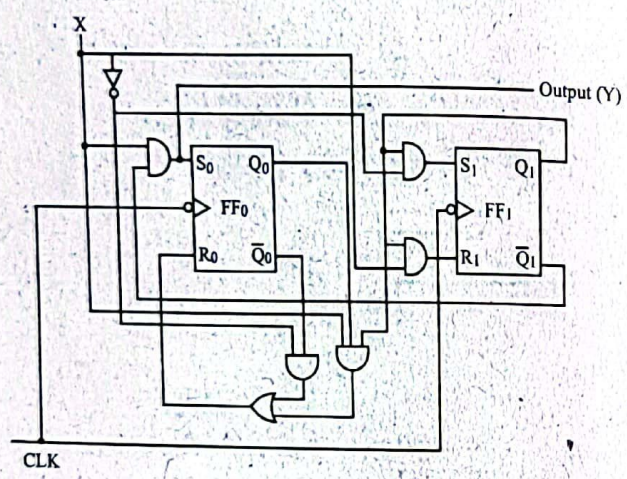


Fig.: Circuit diagram of sequential circuit using SR flipflop

6. Design a synchronous binary 3-bit up counter using RS flipflop. [Spring 2017]

Solution:

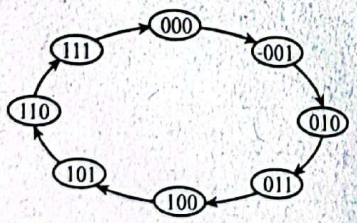


Fig.: State diagram of binary 3-bit up counter

Excitation table of R-S flipflop

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

Synthesis table

Present state			Next state			FF Excitation					
Q_{2n}	Q_{1n}	Q_{0n}	Q_{0n+1}	Q_{1n+1}	Q_{0n+1}	S_2	R_2	S_1	R_1	S_0	R_0
0	0	0	0	0	1	0	×	0	×	1	0
0	0	1	0	1	0	0	×	1	0	0	1
0	1	0	0	1	1	0	×	×	0	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	×	0	0	×	1	0
1	0	1	1	1	0	×	0	1	0	0	1
1	1	0	1	1	1	×	0	×	0	1	0
1	1	1	0	0	0	0	1	0	1	0	1

Using K-map,

For S_2,

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

$\therefore \quad S_2 = \bar{Q}_{2n}Q_{1n}Q_{0n}$

For R_2,

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

$R_2 = Q_{2n}Q_{1n}Q_{0n}$

For S_1,

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

$\therefore \quad S_1 = \bar{Q}_{1n}Q_{0n}$

For R_1,

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

$\therefore \quad R_1 = Q_{0n}Q_{1n}$

For S_0,

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

$\therefore \quad S_0 = \bar{Q}_{0n}$

For R_0,

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

$\therefore \quad R_0 = Q_{0n}$

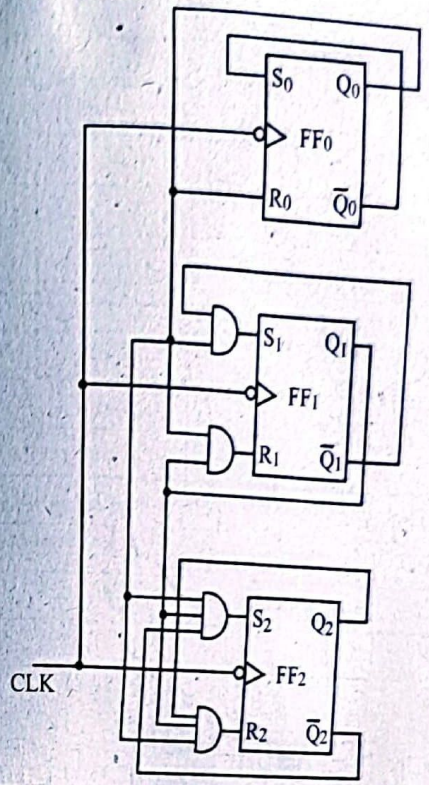
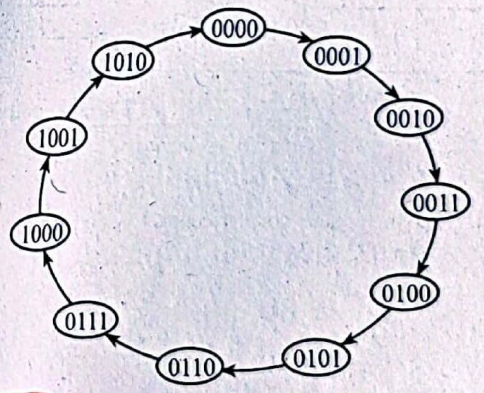


Fig.: Synchronous binary 3-bit up counter using RS flipflop

7. Design a MOD-11 asynchronous counter using JK flipflop and showing with its working, counting sequence and timing diagram. [Spring 2016]

Solution:



...: State diagram of MOD 11 up counter

Truth table

Clock	Q_3	Q_2	Q_1	Q_0	Count
Initially, 0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	10
11	0	0	0	0	0

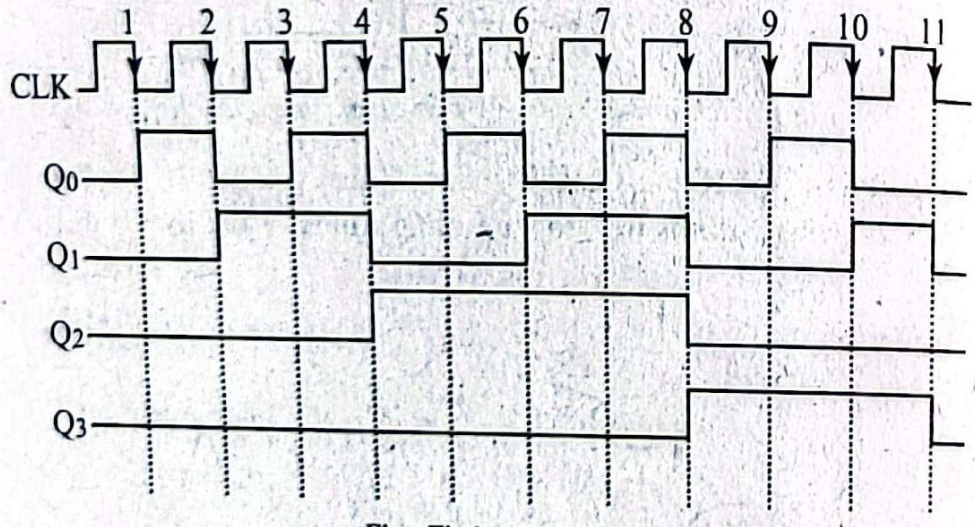


Fig.: Timing diagram

MOD 11 asynchronous counter counts form 0000 to 1010 and resets to 0000 when $Q_3Q_2Q_1Q_0 = 1011$.

Synthesis table

Present state			Next state			FF Excitation		
Q_{2n}	Q_{1n}	Q_{0n}	Q_{2n+1}	Q_{1n+1}	Q_{0n+1}	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	1	0	0	0	1	0
1	1	1	0	0	0	1	1	1

Using k-map,

For T_2,

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

$\therefore \quad T_2 = Q_{2n}Q_{1n}Q_{0n} + \bar{Q}_{2n}Q_{1n}\bar{Q}_{0n}$

$= Q_{1n}(Q_{2n}Q_{0n} + \bar{Q}_{2n}Q_{0n})$

$= Q_{1n}(Q_{0n} \odot Q_{2n})$

For T_1,

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

$\therefore \quad T_1 = \bar{Q}_{1n}Q_{0n} + Q_{2n}Q_{0n}$

$= Q_{0n}(\bar{Q}_{1n} + Q_{2n})$

For T_0,

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

$\therefore \quad T_0 = \bar{Q}_{2n}\bar{Q}_{0n} + Q_{2n}Q_{0n}$

$= Q_{0n} \odot Q_{2n}$

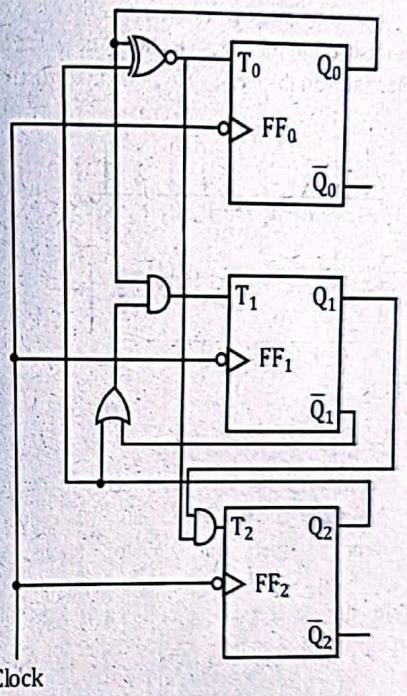


Fig.: Detailed circuit diagram of counter using T flipflop