# Module-5

## Contents:

- ➢ **Registers and Register Transfers**
- ➢ **Parallel Adder with Accumulator**
- ➢ **Shift Registers**
- ➢ **Design of Binary Counters**
- ➢ **Counters for other sequences**
- ➢ **Counter design using SR and JK Flip-Flop**
- ➢ **Sequential Parity Checker**

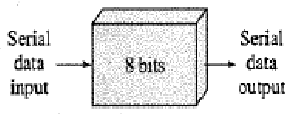
## Registers:

- Register is a group of flip-flops, which is used to store a word.
- Bits in register can be shifted by applying clock signal, hence registers are also called as **Shift Register.**

### Types of Registers:

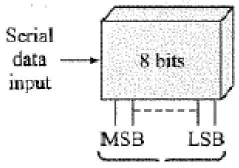
Data can be shifted into a register either serially or in parallel, similarly data can be shifted out of the register either serially or in parallel. Based on serial and parallel transmission, registers are classified into four types

1. Serial in -Serial out Register

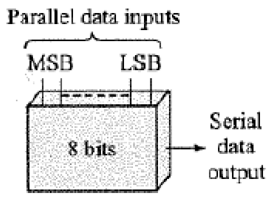


(a) Serial in–serial out

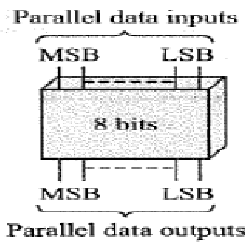
2. Serial in-Parallel out Register



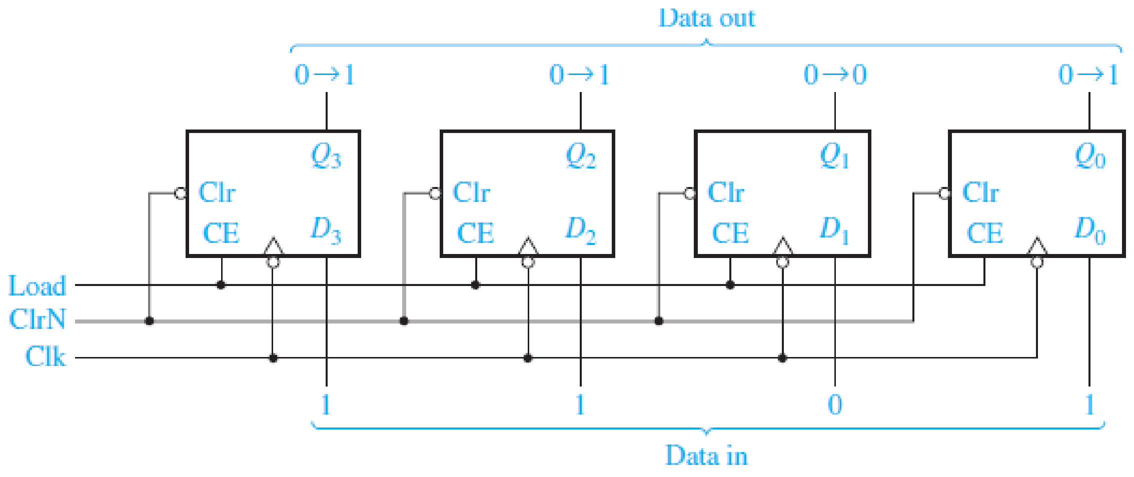
3. Parallel in-Serial out Register



4. Parallel in-Parallel out register

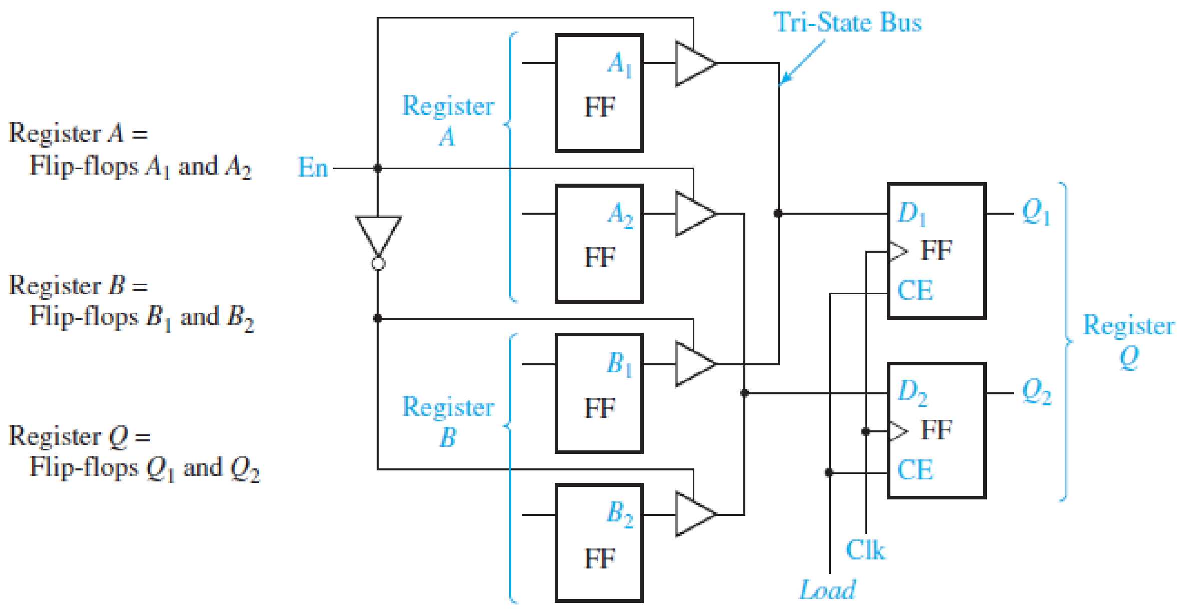


**4 Bit Register:**



Several D Flip-Flops may be grouped together with a common clock to form a register. Because each flip-flop can store one bit of information, this register can store four bits. When **LOAD = 1** and Clock signal is negative, the data applied at D0, D1, D2 and D3 are loaded into flip-flops in parallel. When **ClrN**= 0 all flip-flops will get cleared.

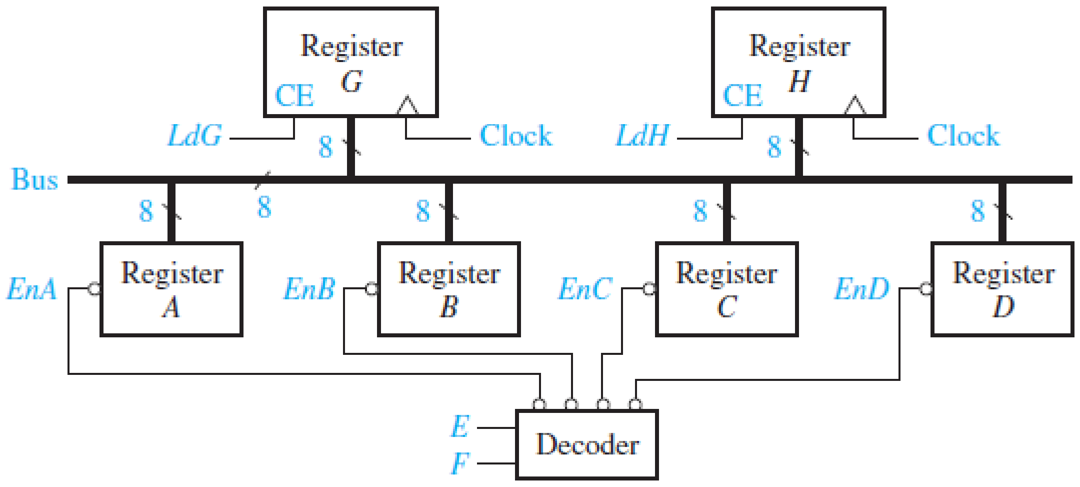
**Data transfer Between Registers:**


Tri-State Bus
Register A = Flip-flops $A_1$ and $A_2$
Register B = Flip-flops $B_1$ and $B_2$
Register Q = Flip-flops $Q_1$ and $Q_2$
En
Register A
Register B
$A_1$ FF
$A_2$ FF
$B_1$ FF
$B_2$ FF
$D_1$ FF CE
$D_2$ FF CE
$Q_1$
$Q_2$
Register Q
Clk
Load

The above figure shows how data can be transferred from the output of one of two registers to a third register using tri-state buffers. If **En=1** and **Load=1** the output of register A is enabled onto the tri-state bus and data in the register A loads into the register Q.

If **En=0** and **Load=1,** the content of register B loads into register Q.

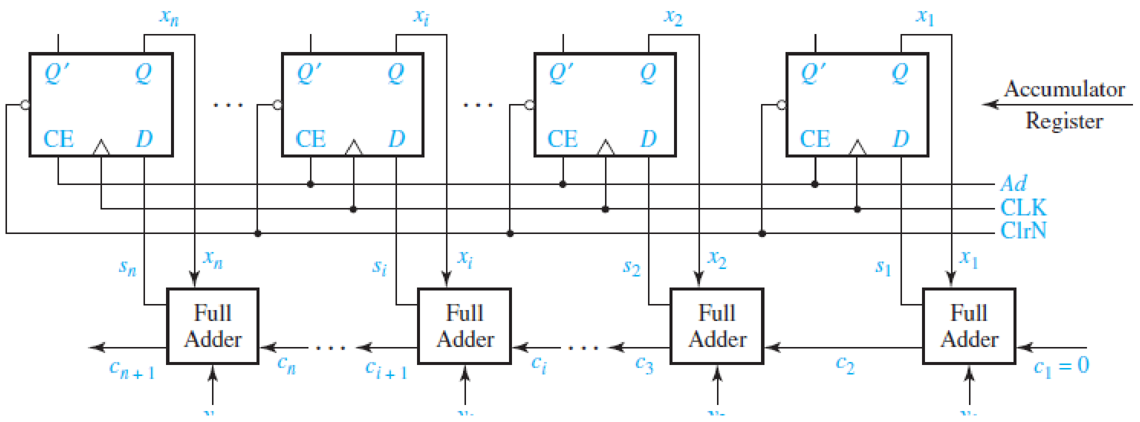
## **Data Transfer using a Tri-State Bus:**



The above figure shows how data can be transferred from one of four 8-bit registers into one of two other registers. The outputs of registers A,B,C,D are connected to Bus, the inputs of registers G and H are connected to Bus. When **EnA=0**, the Register A contents are transferred to Bus, when LdG=1, the data on the bus loads in to the register G at positive edge of clock, if LdH=1, the data on the bus loads into register H, Similarly, the data in registers B,C,D is transferred to either register G or H when **EnB, EnC, EnD is 0** respectively. When both LdG and LdH are 1 the data on the bus will be loaded to both registers. Using decoder any one of registers A,B,C,D can be enabled as shown below.

If $EF = 00$, $A$ is stored in $G$ (or $H$).
If $EF = 01$, $B$ is stored in $G$ (or $H$).
If $EF = 10$, $C$ is stored in $G$ (or $H$).
If $EF = 11$, $D$ is stored in $G$ (or $H$).

**n-bit Parallel Adder with Accumulator:**



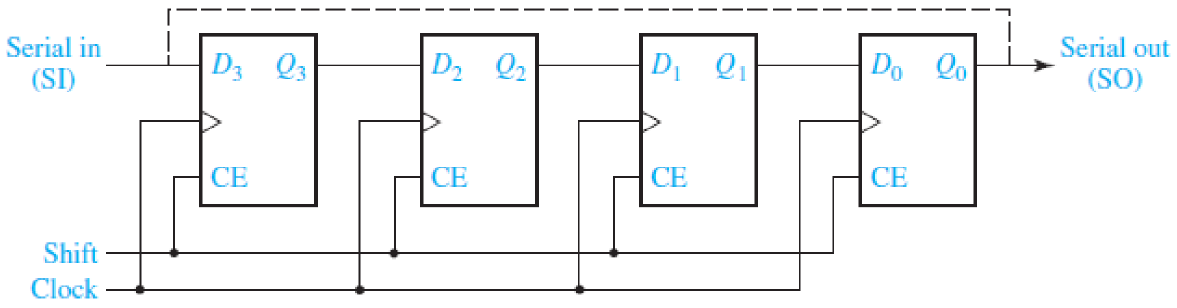
The above circuit can be used to add the number X in accumulator with number Y and store sum in accumulator, initially load the accumulator register with the number **X($X_4$,$X_3$,$X_2$,$X_1$**) and apply the number **Y($Y_4$,$Y_3$,$Y_2$,$Y_1$**) to full adder inputs. After the carry has propagated through the adders, the sum of X and Y appears the Full adder outputs, when **Ad=1, and clk is positive edge,** the outputs of full adders(S4, S3, S2, S1) will be loaded into the accumulator.

Parallel adder with accumulator is iterative structure that consists of a number of identical cells, each cell contains a full adder and an associated accumulator flip-flop. Before addition can take place, the accumulator must be loaded with number X, to do this first clear the accumulator with clrN, and apply the number X on the Y inputs the adder and add to the accumulator in normal way.

## Shift Registers:

A shift register is a register in which binary data can be stored and this data can be shifted to the left or right when shift signal is applied. Bits shifted out one end of the register may be lost.

## 4 –bit SISO register:



### Operation:

Assume initial state of register is 0, hence Q3Q2Q1Q0 = 0000

When bits 1011 are applied serially to register, data in the counter shifted as follows

**At first clock Negative edge:** LSB bit 1 will be shifted to FF-3, $Q_3$ bit will be shifted to $Q_2$, $Q_2$ will be shifted to $Q_1$, $Q_1$ will be shifted to $Q_0$, hence after first Negative edge of clock

**$Q_3Q_2Q_1Q_0$ = 1000**

**At second Clock Negative Edge:** Bit 1 is applied to FF-3, bits in counter will be shifted right side by one position, after second clock negative edge

**$Q_3Q_2Q_1Q_0$ = 1100**

**At third Clock Negative Edge:** Bit 0 is applied to FF-3, bits in counter shifts right side, hence after third clock negative edge

**$Q_3Q_2Q_1Q_0$ = 0110**

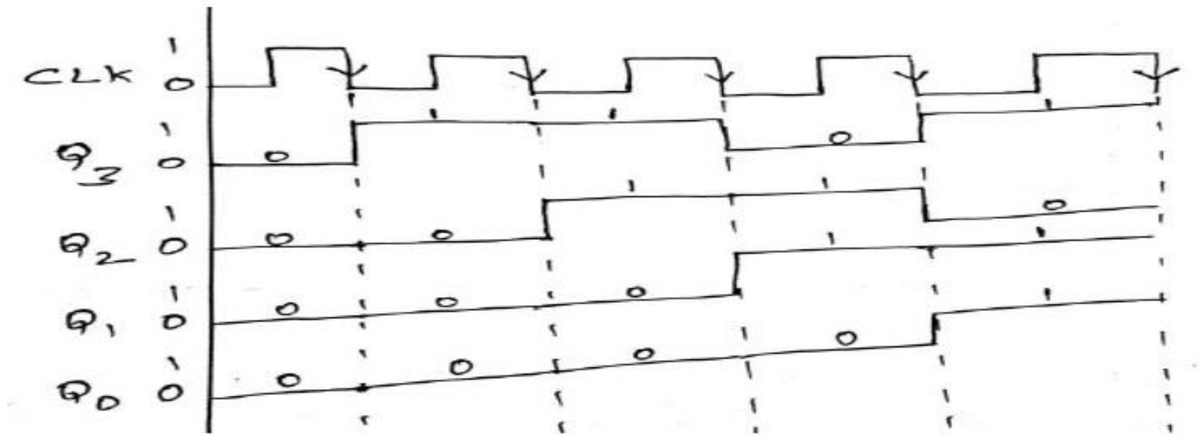
**At Fourth Clock Negative Edge:** Bit 1 is applied to FF-3, bits in counter shifts right side, hence after third clock negative edge

**$Q_3Q_2Q_1Q_0$ = 1011**

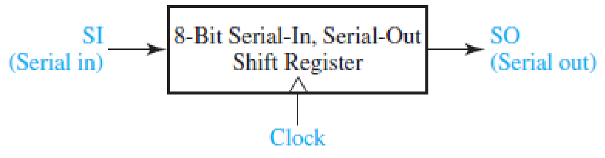
**Hence, after four clock cycles serial data 1011 is loaded in to the register**

| CLK | Serial Data | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 1 | 1 | 0 | 1 | 1 |

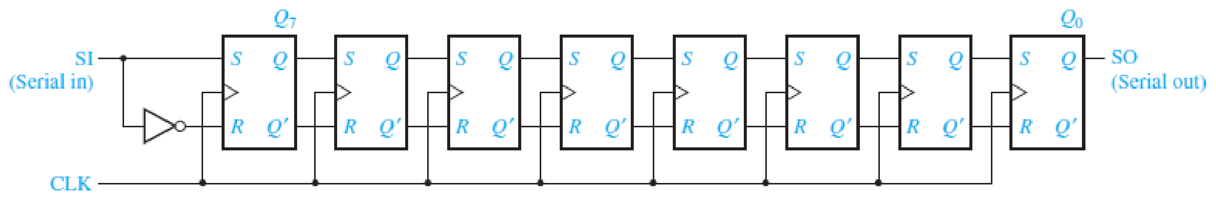
**Timing Waveforms:**



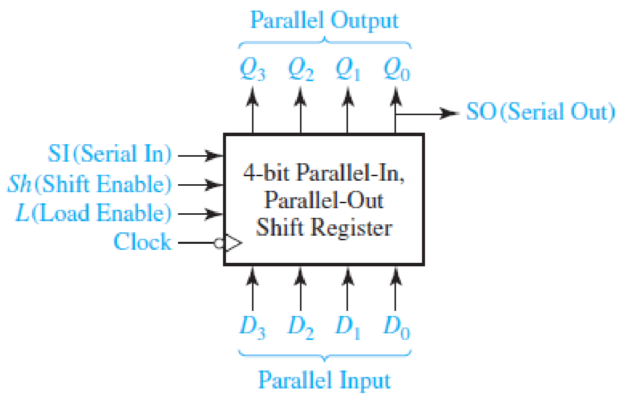
**8-bit serial IN Serial Out Shift register using SR Flip-Flops:**



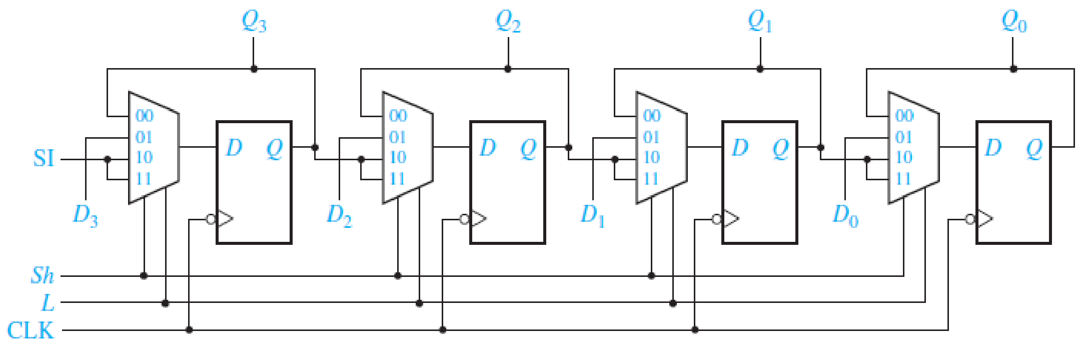
(a) Block diagram



**Parallel-in Parallel-out Right Shift Register:**



(a) Block diagram



(b) Implementation using flip-flops and MUXes

In Parallel-in Parallel-out register, all four bits can be loaded at the same time, and parallel out implies that all bits can be read out at the same time. This register has two control inputs shift enable (Sh) and load enable (L).

Case(i) : Sh=0 and L=0

When Sh=0 and L=0, the outputs of all the flip-flops remain in previous state, hence outputs do not change.

Case(ii) Sh=0 and L=1

When Sh=0 and L=1, the data bits applied at the inputs of flip-flops in parallel will be loaded into the flip-flops simultaneously.

Case(iii) Sh=1 and L=1 or 0

When Sh=1 and L=1 or 0, the data bit applied at the serial input(SI) will be loaded into the first Flip-Flop, while data in the flip-flops Q3, Q2 and Q1 are shifted right.

**Truth Table:**

| Inputs | | Next State | | | | |
|---|---|---|---|---|---|---|
| *Sh* (Shift) | *L* (Load) | $Q_3^+$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | Action |
| 0 | 0 | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | No change |
| 0 | 1 | $D_3$ | $D_2$ | $D_1$ | $D_0$ | Load |
| 1 | X | SI | $Q_3$ | $Q_2$ | $Q_1$ | Right shift |

## Design of Binary Counters:

Binary counters can be designed by connecting the group of flip-flops in series.

**Problem: Design MOD-8 counter using T-Flip Flop**

**Sol) Refer Class Notes**

**Problem: Design MOD-8 Bianary UP – DOWN counter**

**Sol) Refer class Notes**

## Counters for other sequence:

In some applications, the sequence of states of a counter is not in straight order, the below figure shows state graph for such counter. The arrows indicates sequence. If this counter starts in state 000, the next state is 100 and then 111.

$$\begin{tikzcd}
 & 000 \arrow[dr] & \\
011 \arrow[ur] & & 100 \arrow[d] \\
010 \arrow[u] & & 111 \arrow[ll]
\end{tikzcd}$$

**Problem: Design self-correcting MOD-6 counter in which all unused states leads to CBA=000**

**State Table:**

| $C_n$ | $B_n$ | $A_n$ | $C_{n+1}$ | $B_{n+1}$ | $A_{n+1}$ | $J_C$ | $K_C$ | $J_B$ | $K_B$ | $J_A$ | $K_A$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0 | 0 | × | 1 | 0 | × | × | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | × | 1 | × | 1 | 0 | × |
| 1 | 1 | 1 | 0 | 0 | 0 | × | 1 | × | 1 | × | 1 |

| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | × | × | × | × |

$J_C = B_nA_n$

| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | × | × | × | × |
| 1 | 0 | 1 | 1 | 1 |

$K_C = A_n + B_n$

| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | × | × |
| 1 | 0 | 0 | × | × |

$J_B = \overline{C}_nA_n$

| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | × | × | 1 | 0 |
| 1 | × | × | 1 | 1 |

$K_B = A_n + C_n$

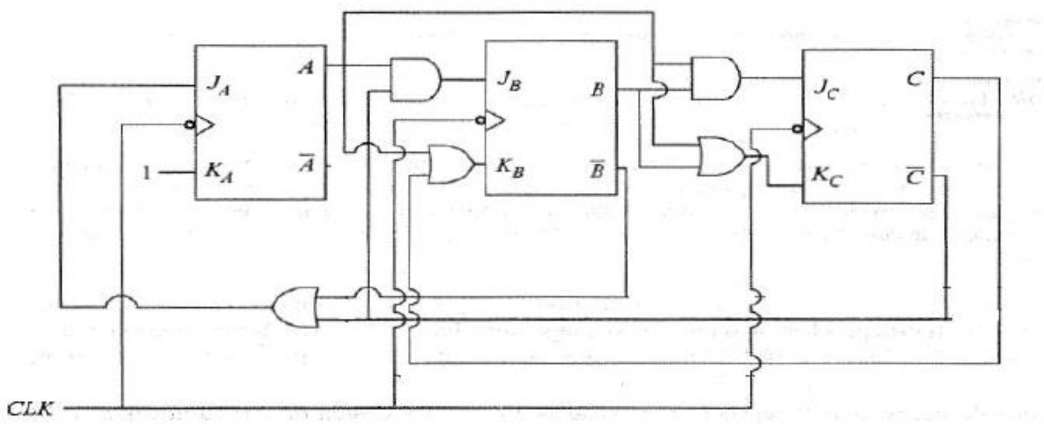
| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | × | × | 1 |
| 1 | 1 | × | × | 0 |

$J_A = \overline{C}_n + \overline{B}_n$

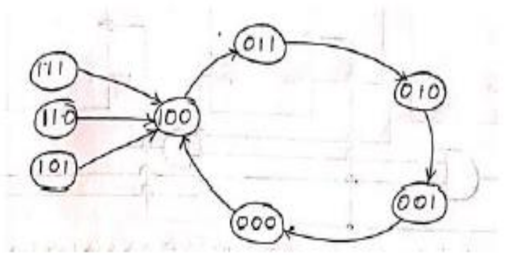
| $C_n \backslash B_nA_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | × | 1 | 1 | × |
| 1 | × | 1 | 1 | × |

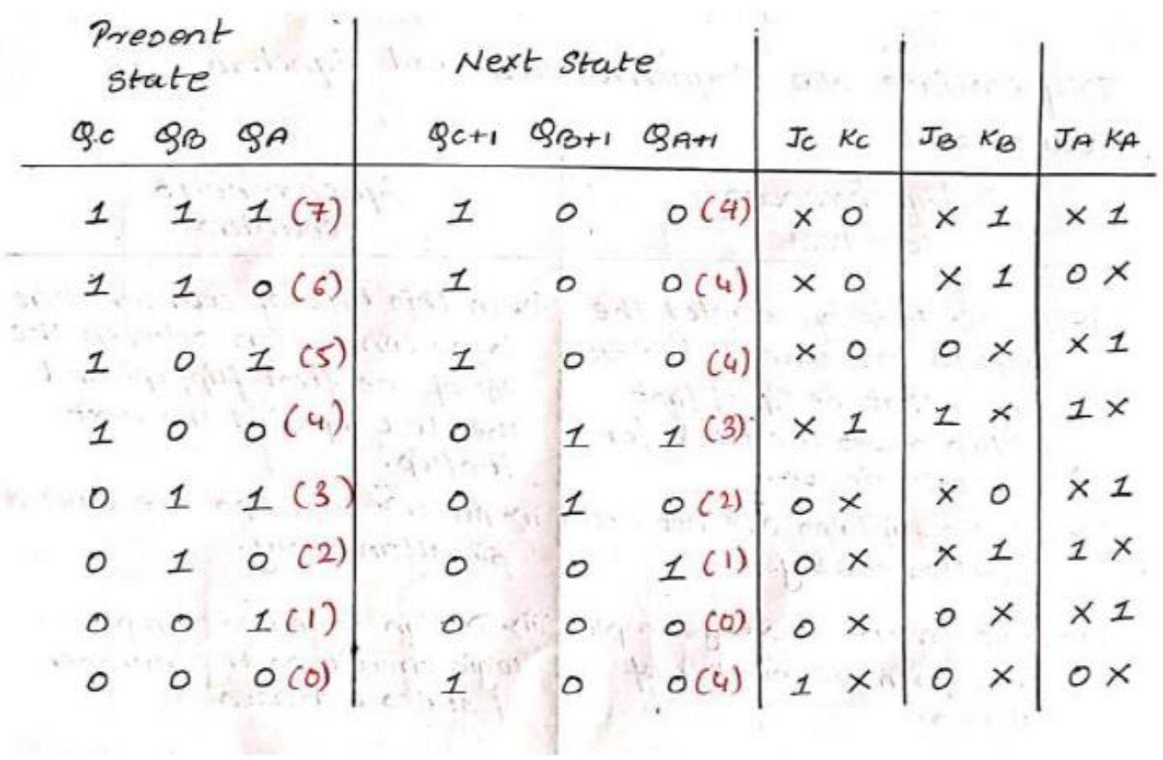
$K_A = 1$

**Mod-6 self-correcting circuit:**



**Problem: Design Self correcting MOD-5 Synchronous Down Counter using JK Flip-flops, Assume 100 as next state for all unused states.**





| Present State | | | Next State | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_C$ | $Q_B$ | $Q_A$ | $Q_{C+1}$ | $Q_{B+1}$ | $Q_{A+1}$ | $J_C$ | $K_C$ | $J_B$ | $K_B$ | $J_A$ | $K_A$ |
| 1 | 1 | 1 (7) | 1 | 0 | 0 (4) | × | 0 | × | 1 | × | 1 |
| 1 | 1 | 0 (6) | 1 | 0 | 0 (4) | × | 0 | × | 1 | 0 | × |
| 1 | 0 | 1 (5) | 1 | 0 | 0 (4) | × | 0 | 0 | × | × | 1 |
| 1 | 0 | 0 (4) | 0 | 1 | 1 (3) | × | 1 | 1 | × | 1 | × |
| 0 | 1 | 1 (3) | 0 | 1 | 0 (2) | 0 | × | × | 0 | × | 1 |
| 0 | 1 | 0 (2) | 0 | 0 | 1 (1) | 0 | × | × | 1 | 1 | × |
| 0 | 0 | 1 (1) | 0 | 0 | 0 (0) | 0 | × | 0 | × | × | 1 |
| 0 | 0 | 0 (0) | 1 | 0 | 0 (4) | 1 | × | 0 | × | 0 | × |

for $J_A$

| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | X | X | 1 |
| 1 | 1 | X | X | 0 |

$J_A = Q_B\overline{Q_C} + \overline{Q_B}Q_C$

for $K_A$

| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | 1 | 1 | X |
| 1 | X | 1 | 1 | X |

$K_A = 1$

for $J_B$

| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | X | X |
| 1 | 1 | 0 | X | X |

$J_B = \overline{Q_A}\,Q_C$

for $K_B$

| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | 0 | 1 |
| 1 | X | X | 1 | 1 |

$K_B = \overline{Q_A} + Q_C$

for $J_C$

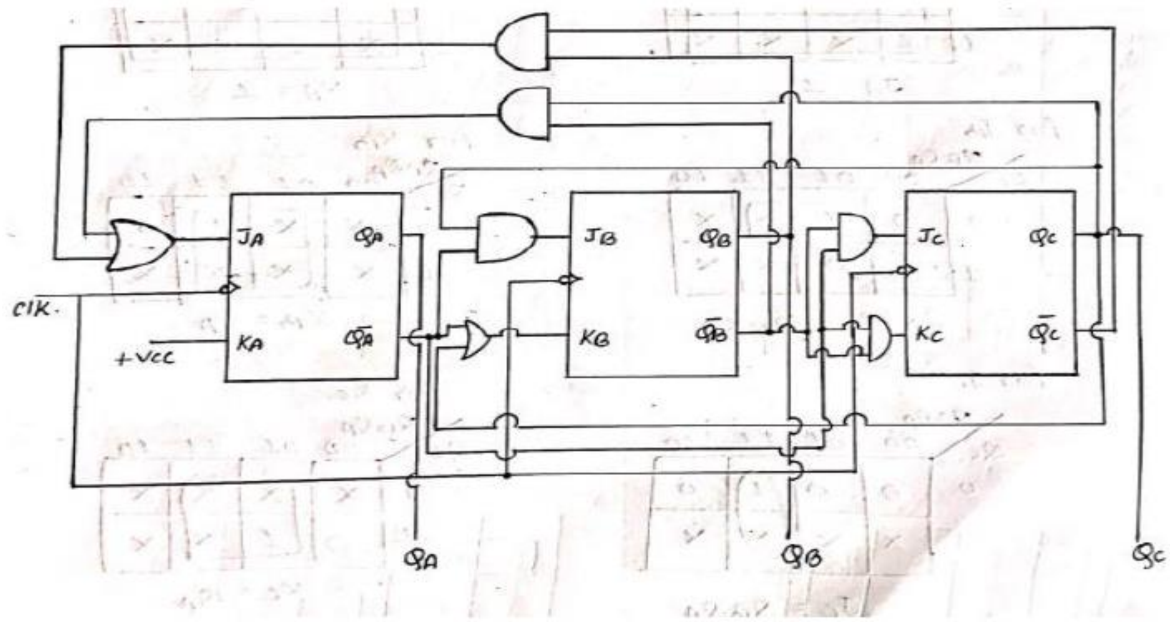
| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | X | X | X | X |

$J_C = \overline{Q_A}\,\overline{Q_B}$

for $K_C$

| $Q_C \backslash Q_BQ_A$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | X | X |
| 1 | 1 | 0 | 0 | 0 |

$K_C = \overline{Q_A}\,\overline{Q_B}$

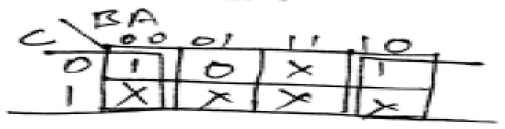


**Define Counter. Design a synchronous counter for the Sequence 0→4→1→2→6→0→4 Using JK Flip-Flop.**

**State table:**

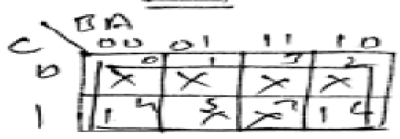
| C B A | C+1 B+1 A+1 | $J_C$ $K_C$ | $J_B$ $K_B$ | $J_A$ $K_A$ |
|---|---|---|---|---|
| 0 0 0 | 1 0 0 | 1 X | 0 X | 0 X |
| 1 0 0 | 0 0 1 | X 1 | 0 X | 1 X |
| 0 0 1 | 0 1 0 | 0 X | 1 X | X 1 |
| 0 1 0 | 1 1 0 | 1 X | X 0 | 0 X |
| 1 1 0 | 0 0 0 | X 1 | X 1 | 0 X |

$J_C$



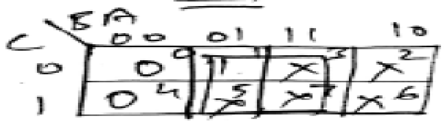
$J_C = \bar{A}$

$K_C$



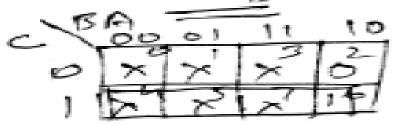
$K_C = 1$

$J_B$



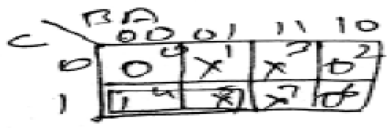
$J_B = A$

$K_B$



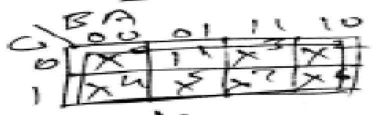
$K_B = C$

$J_A$



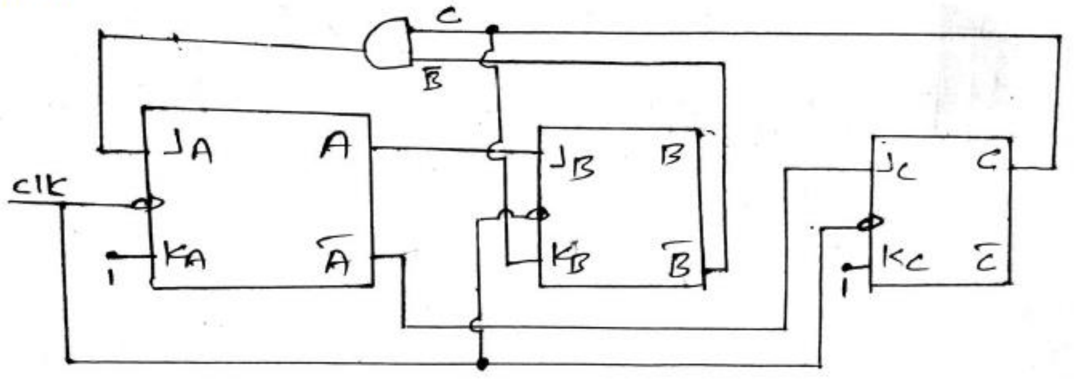
$J_A = \bar{B}C$

$K_A$



$K_A = 1$

**Circuit:**

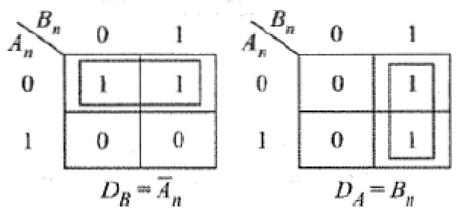


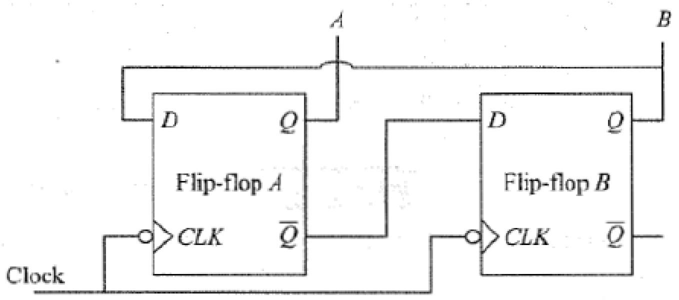
**Problem: Design Modulo-4 irregular counter with following sequence using D-Flip-Flop**

$$\begin{tikzcd}
00 \arrow[r] & 10 \arrow[r] & 11 \arrow[r] & 01 \arrow[lll, bend left]
\end{tikzcd}$$

| $B_n$ | $A_n$ | $B_{n+1}$ | $A_{n+1}$ | $D_B$ | $D_A$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |

Design equations from Karnaugh Map can be derived as shown in Fig. 10.38(a), and corresponding logic circuit is shown in Fig. 10.38(b).



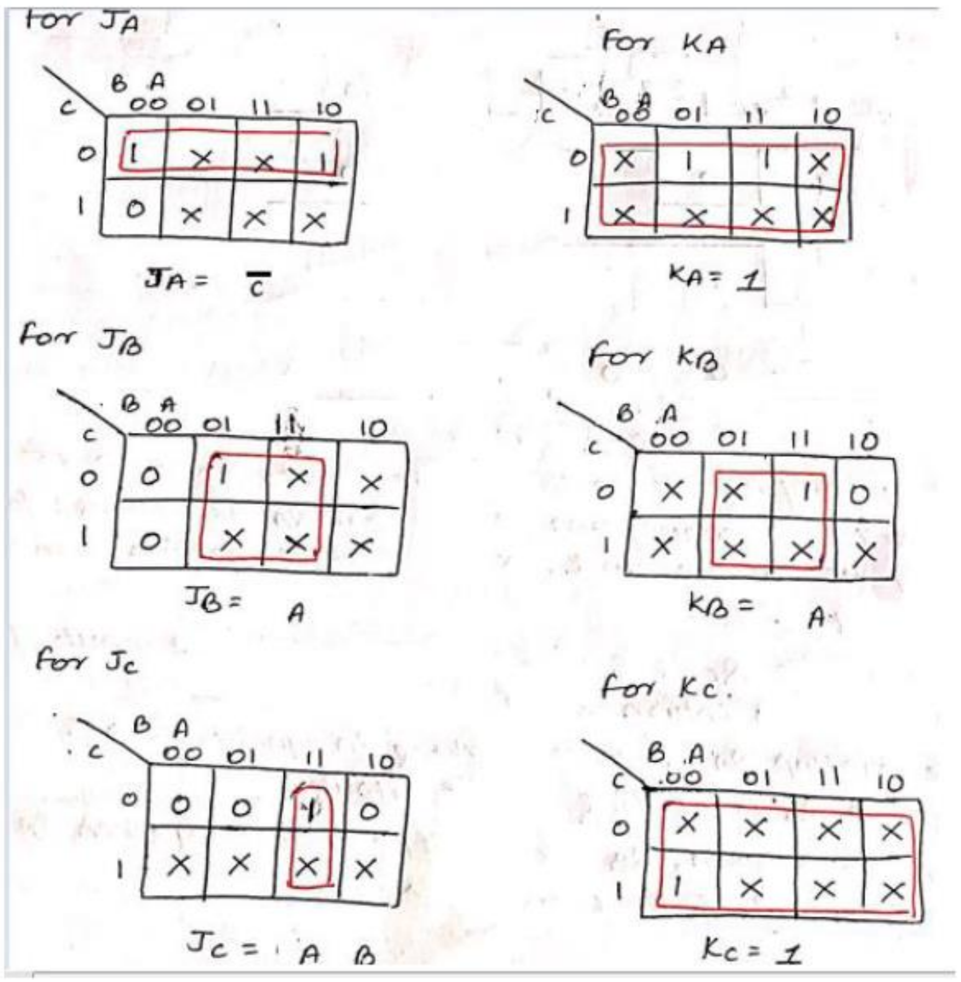


## Counter design using SR and JK Flip-Flop:

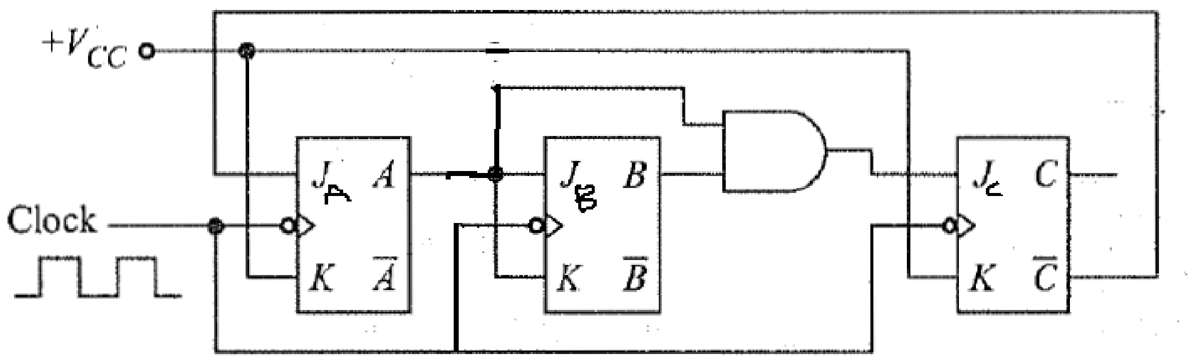
**Problem: Design MOD-5 counter using JK flip-flop**

**State table**

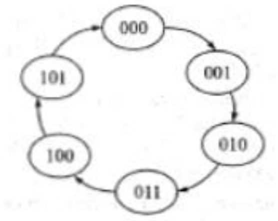
| C B A | C+1 B+1 A+1 | $J_C$ $K_C$ | $J_B$ $K_B$ | $J_A$ $K_B$ |
|---|---|---|---|---|
| 0 0 0 | 0 0 1 | 0 × | 0 × | 1 × |
| 0 0 1 | 0 1 0 | 0 × | 1 × | × 1 |
| 0 1 0 | 0 1 1 | 0 × | × 0 | 1 × |
| 0 1 1 | 1 0 0 | 1 × | × 1 | × 1 |
| 1 0 0 | 0 0 0 | × 1 | 0 × | 0 × |



**MOD-5 Counter Circuit:**

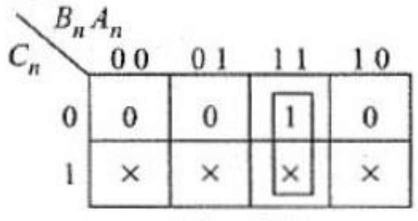


**Designing MOD-6 Counter:**

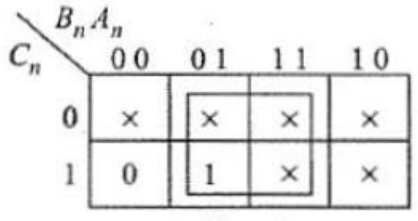


**State table:**

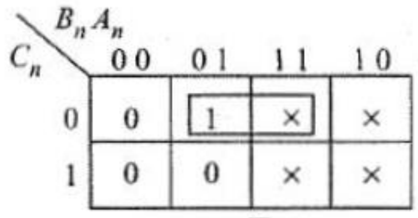
| $C_n$ | $B_n$ | $A_n$ | $C_{n+1}$ | $B_{n+1}$ | $A_{n+1}$ | $J_C$ | $K_C$ | $J_B$ | $K_B$ | $J_A$ | $K_A$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | × | 0 | × | 1 | × |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | × | 1 | × | × | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | × | × | 0 | 1 | × |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | × | × | 1 | × | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | × | 0 | 0 | × | 1 | × |
| 1 | 0 | 1 | 0 | 0 | 0 | × | 1 | 0 | × | × | 1 |



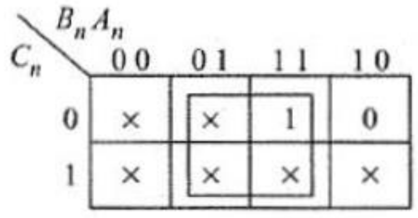
$J_C = B_n A_n$



$K_C = A_n$



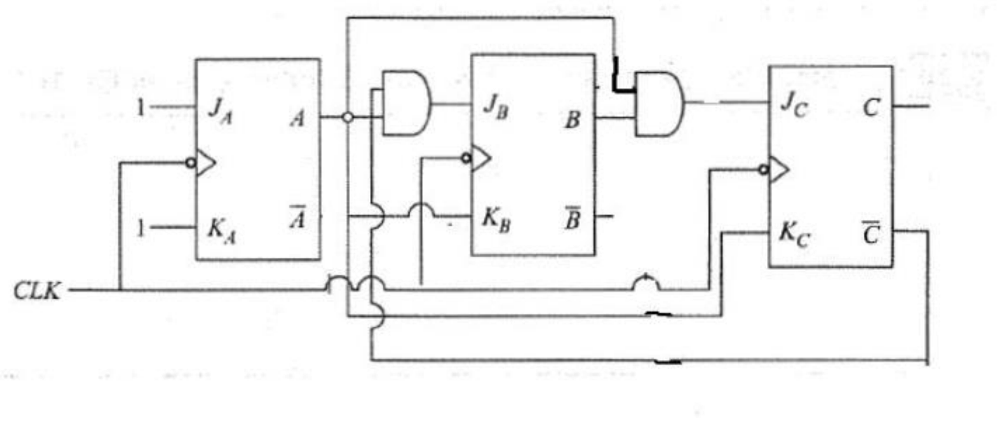
$J_B = \overline{C}_n A_n$



$K_B = A_n$

**$J_A$ =1 and $K_A$=1**

**MOD-6 Circuit-Diagram:**



**<u>Pre-settable counter or Loadable counter:</u>**

**In loadable counter, user can set initial state of the counter, after initial state counte can be made to count either UP or DOWN.**

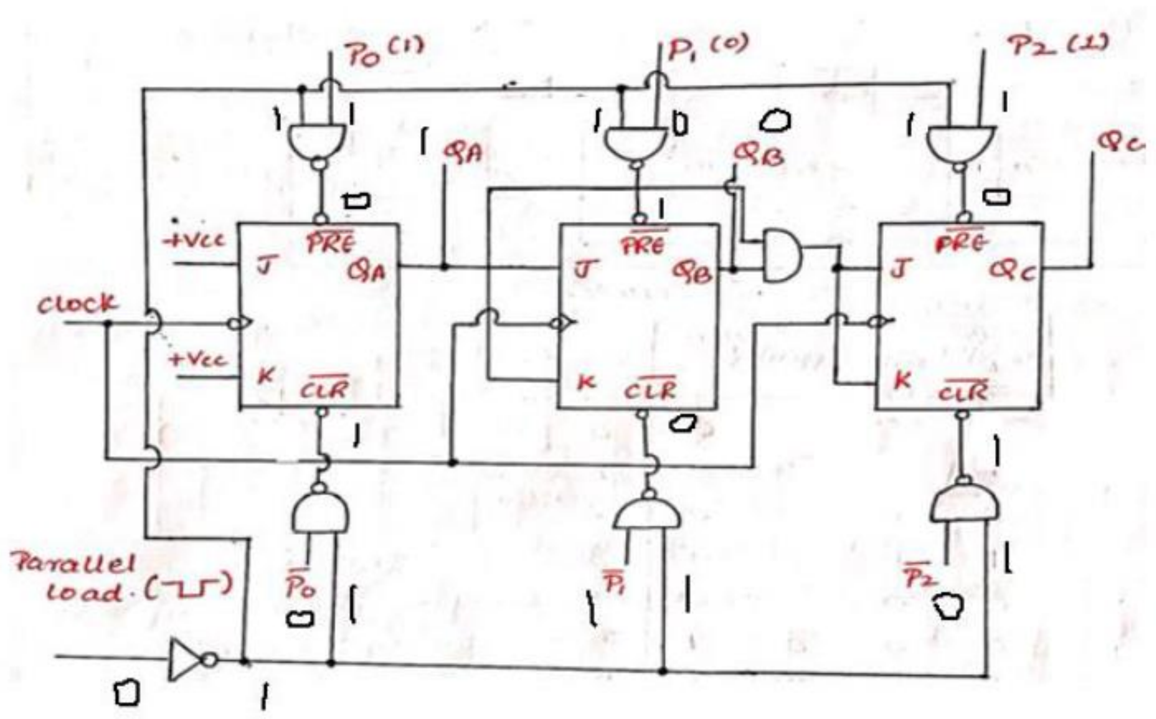
**MOD-8 Presettable counter with initial state =101 (5):**

Figure below shows, MOD-8 counter, to set the initial state 101 (5), apply 101 to parallel inputs $P_2P_1P_0$ =101, and apply parallel load =0, hence inversion of parallel load 1 will be applied to one of inputs of all NAND gates, at flip-flop inputs for NAND gate are 11 hence its output 0 is applied to PRESET of FF-A, hence output of a FF-A is 1, similarly at FF-B PRESET=1 and CLR=0, hence output of FF-B is 0, and at FF-C PRESET=0 and CLR=1, hence FF-C output is 1, now state of counter $Q_CQ_BQ_A$=101, after this state if **Parallel Load is made 1,** hence all

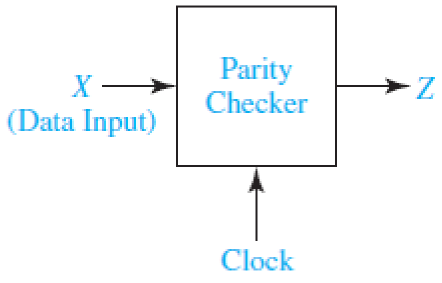
PRESET and RESET inputs are deactivated, since counter is designed as UP-counter, it increments the count by 1 on every negative clock cycle.

**Truth Table:**

| PL | P2 P1 P0 | $Q_C$ $Q_B$ $Q_A$ |
|---|---|---|
| 0 | 1 0 1 | 1 0 1 |
| 1 | × × × | 1 1 0 |
| 1 | × × × | 1 1 1 |
| 1 | × × × | 0 0 0 |
| 1 | × × × | 0 0 1 |
| 1 | × × × | 0 1 0 |
| 1 | × × × | 0 1 1 |
| 1 | × × × | 1 0 0 |



**Sequential Parity Checker:**

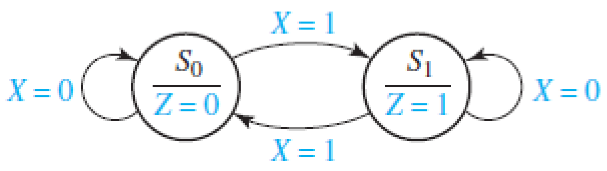


A sequential parity checker is used to check the parity of the binary data applied at input X, the data is applied serially (bit by bit) at the input X, parity checker produce the output 1 if the total number of 1's applied at the input is ODD, it produce the output 1 if total number of 1'ss applied at the input is EVEN.

**State Graph:**

The output of SPC is either 0 or 1, hence it has two states, lets designate these two states as **$S_0$ and $S_1$.** As shown in the graph below, assume circuit is in state **S0** and **X=0** is received the circuit must stay in State **S0,** if X=1 is received, the circuit goes to state S1 because number of 1's received is **odd.**

Similarly if circuit is in state S1, a 0 input causes no change, but a input 1 causes a change to S0 because number of 1's received is even.

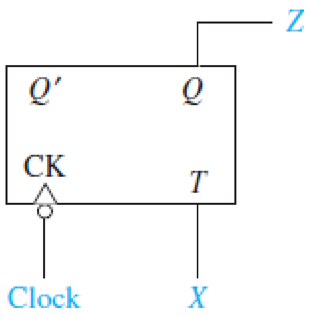


**State Table:**

(a)

| Present State | Next State $X = 0$ | Next State $X = 1$ | Present Output |
|---|---|---|---|
| $S_0$ | $S_0$ | $S_1$ | 0 |
| $S_1$ | $S_1$ | $S_0$ | 1 |

**Implementation of SPC using T-Flip-Flop**



| $Q$ | $Q^+$ $X = 0$ | $Q^+$ $X = 1$ | $T$ $X = 0$ | $T$ $X = 1$ | $Z$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |

**Transition Table**

## Important Questions

**1. Briefly explain how data can be transferred between registers.**

**2. Explain the operational principle of n-bit parallel adder with accumulator.**

**3. Explain  4-bit PIPO right shift register.**

**4. Design Self- correcting MOD-5 Synchronous Down Counter using JK Flip-flops, Assume 100 as next state for all unused states.**

5. **Design Synchronous counter using JK flip-flop for sequence 0→4→1→2→6→0→4**

**6. Design MOD-5 synchronous counter using JK Flip-Flop.**

**7. With state graph, state table, transition table explain the operation of sequential parity checker.**

**8. Design MOD-8 UP-DOWN counter.**