

## MODULE – 5

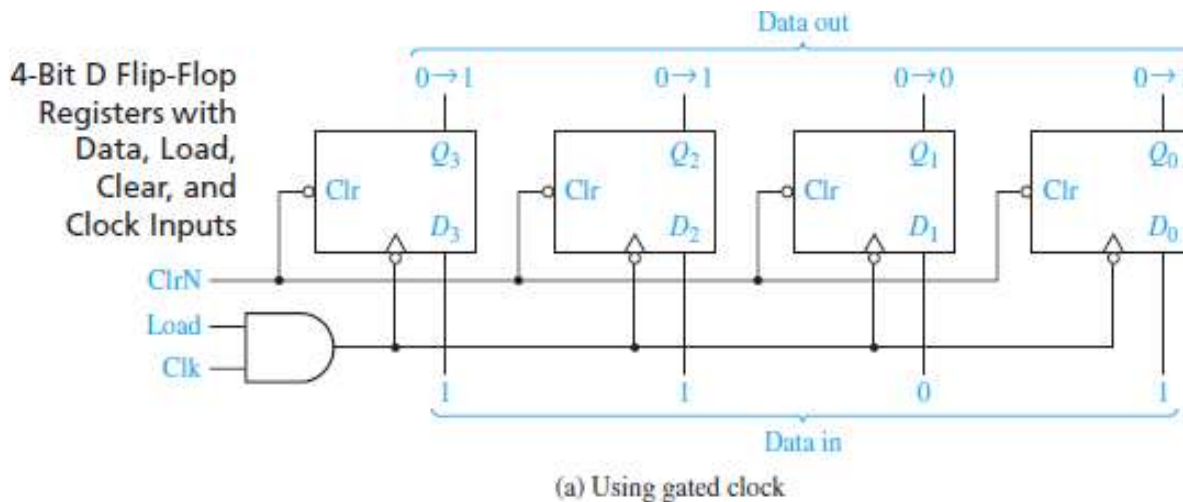
### **REGISTERS AND COUNTERS**

A *register* consists of a group of flip-flops with a common clock input. Registers are commonly used to store and shift binary data.

*Counters* are another simple type of sequential circuits. A counter is usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received.

#### **REGISTERS AND REGISTER TRANSFERS:**

Several D flip-flops may be grouped together with a common clock to form a register (SEE THE FOLLOWING Figure). Since each flip-flop can store one bit of information, this register can store four bits of information. This register has a load signal that is ANDed with the clock.



When  $Load = 0$ , the register is not clocked, and it holds its present value.  $Load = 1$ , the clock signal ( $Clk$ ) is transmitted to the flip-flop clock inputs and the data applied to the D inputs will be loaded into the flip-flops on the falling edge of the clock.

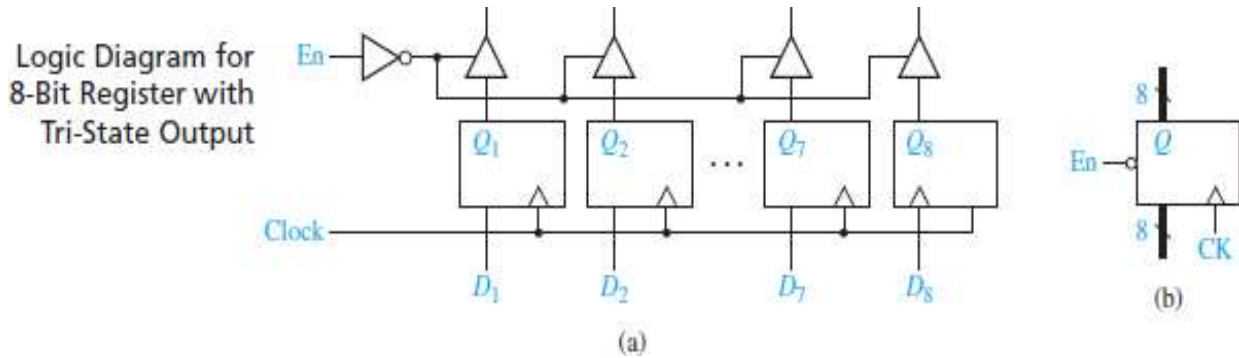
For example, if the Q outputs are 0000 ( $Q_3 = Q_2 = Q_1 = Q_0 = 0$ ) and the data inputs are 1101 ( $D_3 = 1, D_2 = 1, D_1 = 0$  and  $D_0 = 1$ ), after the falling edge of clock, Q will change from 0000 to 1101 as indicated in the above Figure (The notation  $0 \rightarrow 1$  at the flip-flop outputs indicates a change from 0 to 1).

The flip-flops in the register have asynchronous clear inputs that are connected to a common clear signal,  $ClrN$ . The bubble at the clear inputs indicates that a logic 0 is required to clear the flip-flops.  $ClrN$  is normally 1, and if it is changed momentarily to 0, the Q outputs of all four flip-flops will become 0.

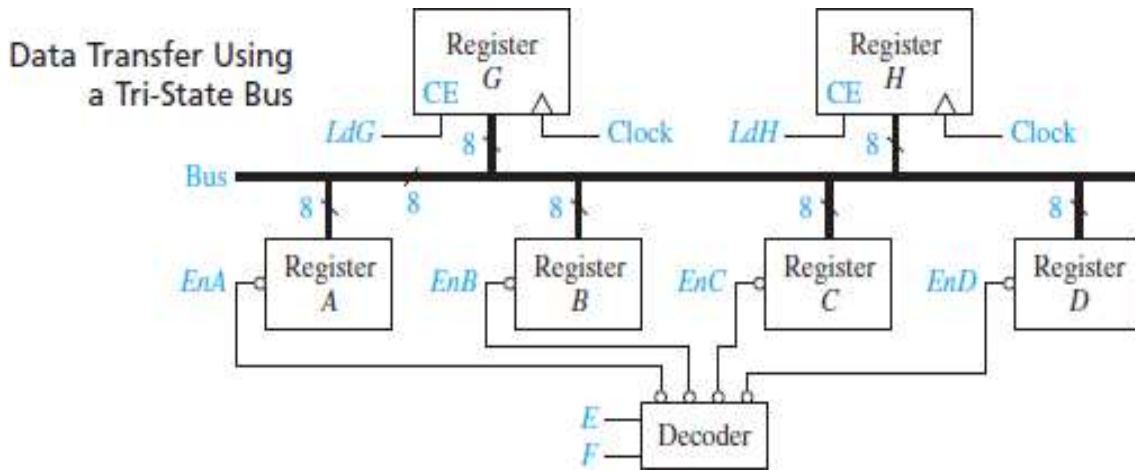
Gating the clock with another signal can cause timing problems. If flip-flops with clock enable are available, the register can be designed as shown in the following Figure (b).



The following Figure (a) shows an integrated circuit register that contains eight D flip-flops with tri-state buffers at the flip-flop outputs. These buffers are enabled when  $En = 0$ . A symbol for this 8-bit register is shown in Figure (b).



The following Figure (c) shows how data can be transferred from one of four 8-bit registers into one of two other registers. Registers A, B, C, and D are of the type shown in above Figure.



The outputs from these registers are all connected in parallel to a common tri-state bus. Registers G and H are 8-bit D type registers (PIPO). The flip-flop inputs of registers G and H are also connected to the bus. When  $EnA = 0$ , the tri-state outputs of register A are enabled onto the bus. If  $LdG = 1$ , these signals on the bus are loaded into register G after the rising clock edge (or into register H if  $LdH = 1$ ). Similarly, the data in register B, C, or D is transferred to G (or H) when  $EnB, EnC$ , or  $EnD$  is 0, respectively and  $LdG = 1$  (or  $LdH = 1$ ). If  $LdG = LdH = 1$ , both G and H will be loaded from the bus. The four enable signals may be generated by a decoder. The operation can be summarized as follows:

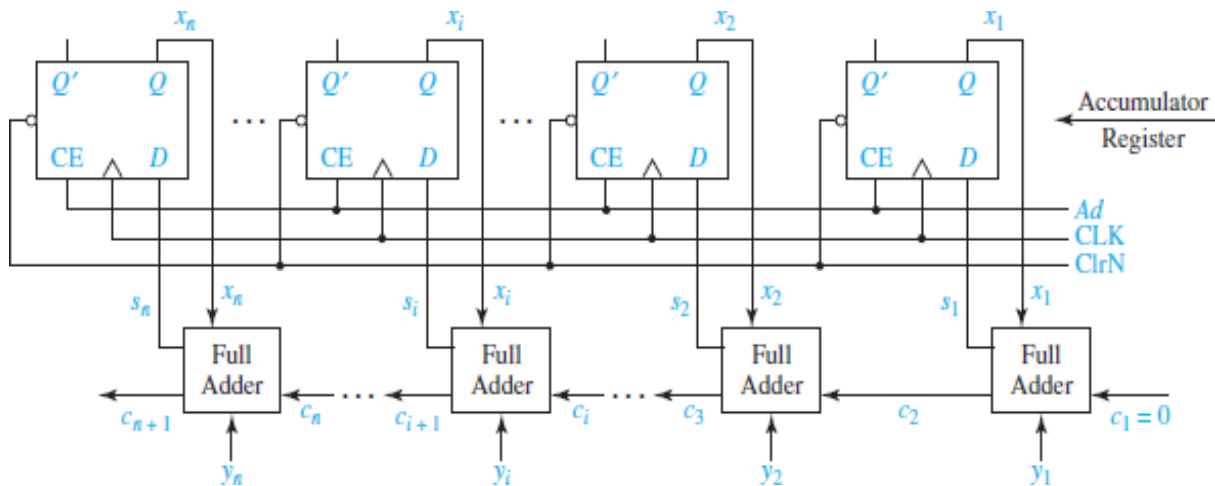
- |  |   |
|--|---|
| If $EF = 00$ , A is stored in G (or H) | If $EF = 01$ , B is stored in G (or H). |
| If $EF = 10$ , C is stored in G (or H) | If $EF = 11$ , D is stored in G (or H). |

Note that 8 bits of data are transferred in parallel from register A, B, C, or D to register G or H. As an alternative to using a bus with tri-state logic, eight 4-to-1 multiplexers could be used, but this would lead to a more complex circuit.

## Parallel Adder with Accumulator:

In computer circuits, it is frequently desirable to store one number in a register of flip-flops (called an accumulator) and add a second number to it, leaving the result stored in the accumulator.

One way to build a parallel adder with an accumulator is to add a register to the adder as shown in the following Figure.

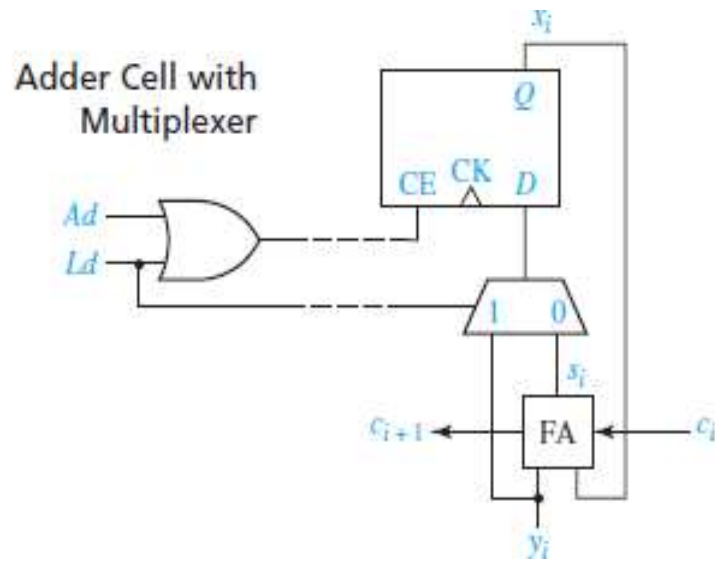


Suppose that the number  $X = x_n \dots x_2 x_1$  is stored in the accumulator. Then, the number  $Y = y_n \dots y_2 y_1$  is applied to the full adder inputs, and after the carry has propagated through the adders, the sum of  $X$  and  $Y$  appears at the adder outputs. An add signal ( $Ad$ ) is used to load the adder outputs into the accumulator flip-flops on the rising clock edge. If  $s_i = 1$ , the next state of flip-flop  $x_i$  will be 1. If  $s_i = 0$ , the next state of flip-flop  $x_i$  will be 0. Thus,  $x_i^+ = s_i$ , and if  $Ad = 1$ , the number  $X$  in the accumulator is replaced with the sum of  $X$  and  $Y$ , following the rising edge of the clock.

Observe that the adder with accumulator is an iterative structure that consists of a number of identical cells. Each cell contains a full adder and an associated accumulator flip-flop. Cell  $i$ , which has inputs  $c_i$  and  $y_i$  and outputs  $c_i = 1$  and  $x_i$ , is referred to as a typical cell.

Before addition can take place, the accumulator must be loaded with  $X$ . This can be accomplished in several ways. The easiest way is to first clear the accumulator using the asynchronous clear inputs on the flip-flops, and then put the  $X$  data on the  $Y$  inputs to the adder and add to the accumulator in the normal way. Alternatively, we could add multiplexers at the accumulator inputs so that we could select either the  $Y$  input data or the adder output to load into the accumulator. This would eliminate the extra step of clearing the accumulator but would add to the hardware complexity.

The following Figure shows a typical cell of the adder where the accumulator flip-flop can either be loaded directly from  $y_i$  or from the sum output ( $s_i$ ). When  $Ld = 1$  the multiplexer selects  $y_i$ , and  $y_i$  is loaded into the accumulator flip-flop ( $x_i$ ) on the rising clock edge. When  $Ad = 1$  and  $Ld = 0$ , the adder output ( $s_i$ ) is loaded into  $x_i$ . The  $Ad$  and  $Ld$  signals are ORed together to enable the clock when either addition or loading occurs. When  $Ad = Ld = 0$ , the clock is disabled and the accumulator outputs do not change.



**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 a shift signal is applied. Bits shifted out one end of the register may be lost, or if the shift register is of cyclic type, bits shifted out one end are shifted back in the other end.

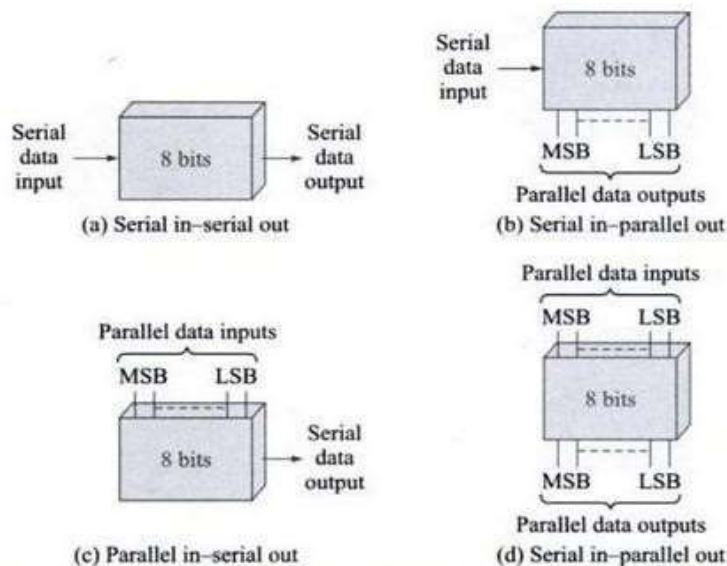
There are two ways to shift data into a register – serial or parallel; and there are two ways to shift the data out of the register – serial or parallel. This leads to the construction of four basic types of registers, as shown in the following Figure. All of these configurations are commercially available as TTL MSI/LSI circuits.

Examples: Serial in – serial out (SISO): 54/74LS91, 8-bits

Serial in – parallel out (SIPO): 54/74164, 8-bits

Parallel in – serial out (PISO): 54/74165, 8-bits

Parallel in – parallel out (PIPO): 54/74194, 4-bits & 54/74168, 8-bits.



Types of Shift Registers

The following Figure (a) illustrates a 4-bit right-shift register with serial input and output constructed from D flip-flops. When  $Shift = 1$ , the clock is enabled and shifting occurs on the rising clock edge. When  $Shift = 0$ , no shifting occurs and the data in the register is unchanged. The serial input ( $SI$ ) is loaded into the first flip-flop ( $Q_3$ ) by the rising edge of the clock. At the same time, the output of the first flip-flop is loaded into the second flip-flop, the output of the second flip-flop is loaded into the third flip-flop, and the output of the third flip-flop is loaded into the last flip-flop. Because of the propagation delay of the flip-flops, the output value loaded into each flip-flop is the value before the rising clock edge.

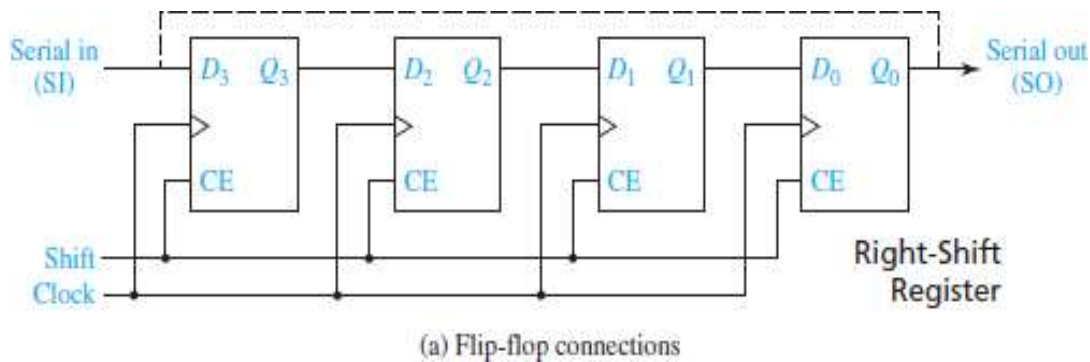
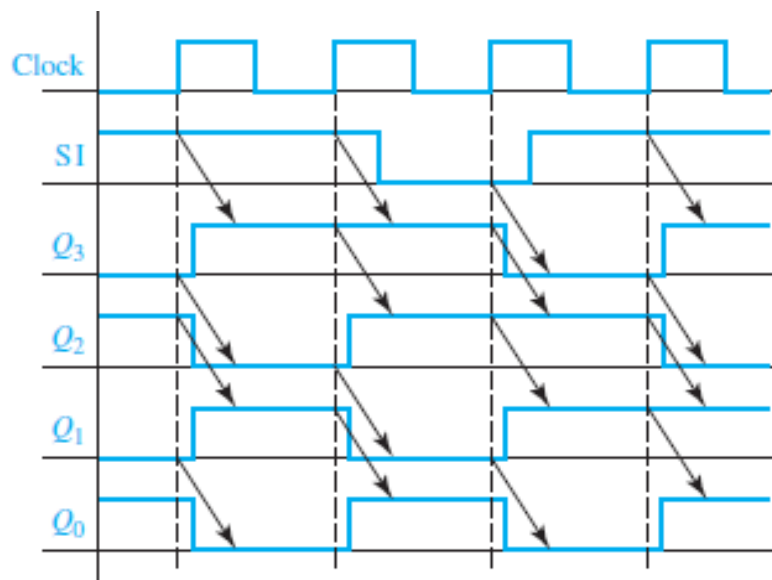


Figure (b) illustrates the timing when the shift register initially contains 0101 and the serial input sequence is 1, 1, 0, 1. The sequence of shift register states is 0101, 1010, 1101, 0110, 1011.

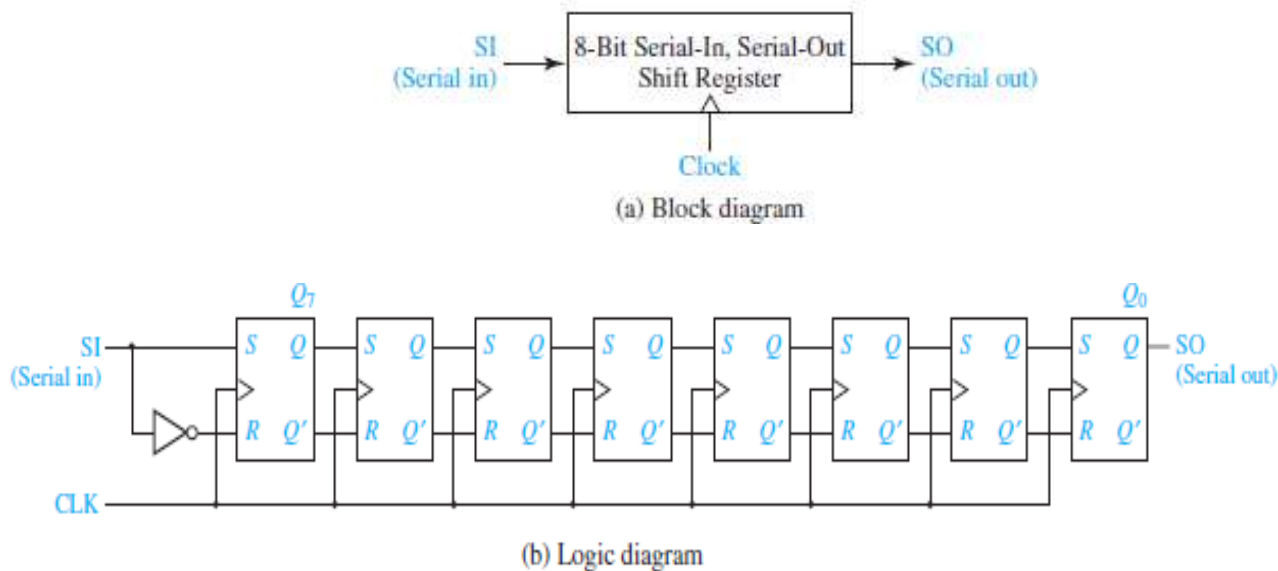


(b) Timing diagram

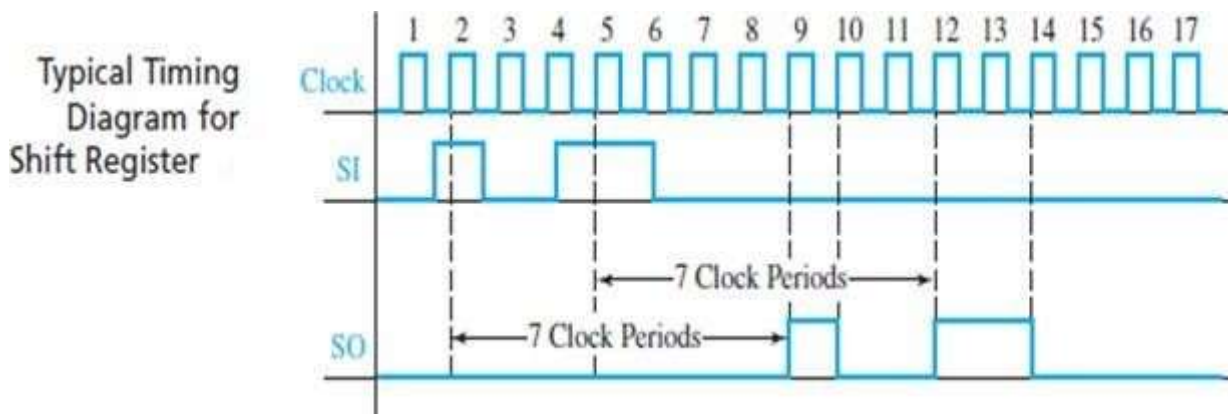
If we connect the serial output to the serial input, as shown by the dashed line, the resulting cyclic shift register performs an end-around shift. If the initial contents of the register is 0111, after one clock cycle the contents is 1011. After a second pulse, the state is 1101, then 1110, and the fourth pulse returns the register to the initial 0111 state.

Shift registers with 4, 8, or more flip-flops are available in integrated circuit form. The following Figure illustrates an 8-bit serial-in, serial-out shift register. Serial in means that data is shifted into the first flip-flop one bit at a time, and the flip-flops cannot be loaded in parallel. Serial out means that data can only be read out of the last flip-flop and the outputs from the other flip-flops are not connected to terminals of the integrated circuit.

### 8-Bit Serial-in, Serial-out Shift Register



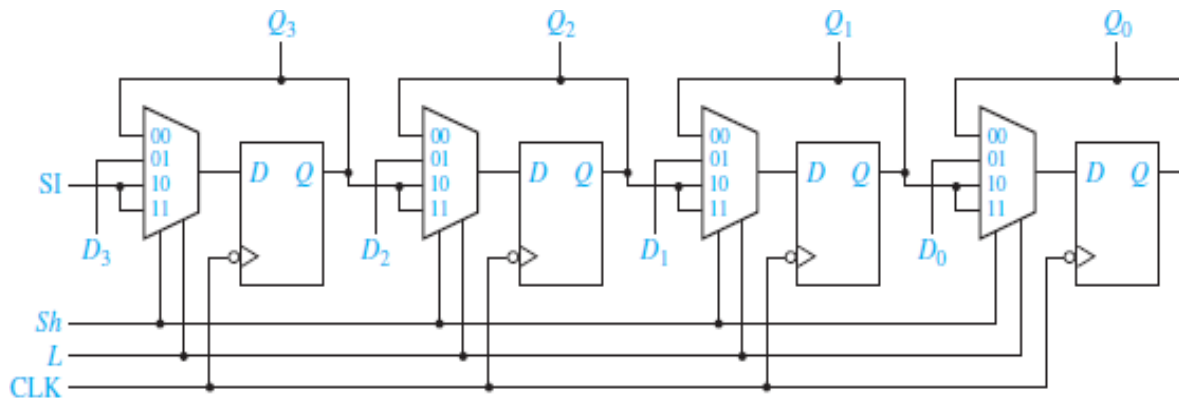
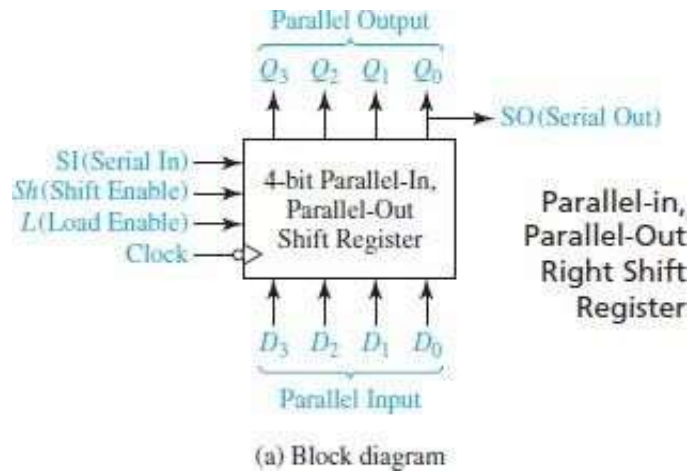
The inputs to the first flip-flop are  $S = SI$  and  $R = SI'$ . Thus, if  $SI = 1$ , a 1 is shifted into the register when it is clocked, and if  $SI = 0$ , a 0 is shifted in. The following Figure shows a typical timing diagram.



The following Figure (a) shows a 4-bit parallel-in, parallel-out shift register. Parallel-in implies that 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. The shift register has two control inputs, shift enable ( $Sh$ ) and load enable ( $L$ ). If  $Sh = 1$  (and  $L = 1$  or  $L = 0$ ), clocking the register causes the serial input ( $SI$ ) to be shifted into the first flip-flop, while the data in flip-flops  $Q_3, Q_2$ , and  $Q_1$  are shifted right. If  $Sh = 0$  and  $L = 1$ , clocking the shift register will



cause the four data inputs (D3, D2, D1, D0) to be loaded in parallel into the flip-flops. If  $Sh = L = 0$ , clocking the register causes no change of state.



The following Table summarizes the operation of this shift register. All state changes occur immediately following the falling edge of the clock.

Shift Register Operation						
Inputs		Next State				Action
$Sh$ (Shift)	$L$ (Load)	$Q_3^+$	$Q_2^+$	$Q_1^+$	$Q_0^+$	
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

The shift register can be implemented using MUXes and D flip-flops, as shown in the above Figure (b). For the first flip-flop, when  $Sh = L = 0$ , the flip-flop  $Q_3$  output is selected by the MUX, so  $Q_3^+ = Q_3$  and no state change occurs. When  $Sh = 0$  and  $L = 1$ , the data input  $D_3$  is selected and loaded into the flip-flop. When  $Sh = 1$  and  $L = 0$  or  $1$ ,  $SI$  is selected and loaded into the flip-flop. The second MUX selects  $Q_2$ ,  $D_2$ , or  $Q_3$ , etc. The next-state equations for the flip-flops are



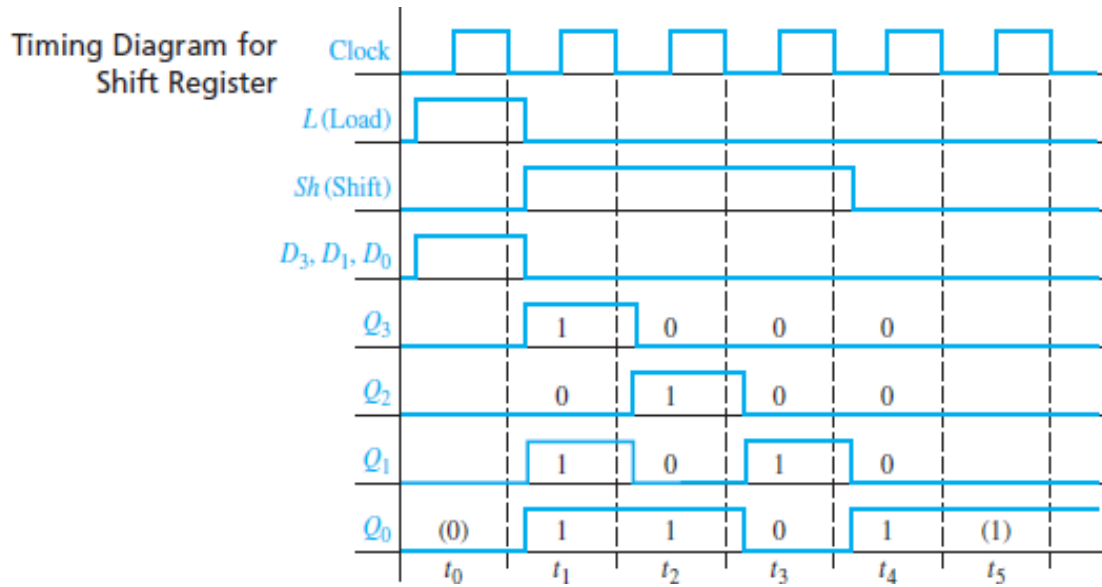
$$Q_3^+ = Sh' \cdot L' \cdot Q_3 + Sh' \cdot L \cdot D_3 + Sh \cdot SI$$

$$Q_2^+ = Sh' \cdot L' \cdot Q_2 + Sh' \cdot L \cdot D_2 + Sh \cdot Q_3$$

$$Q_1^+ = Sh' \cdot L' \cdot Q_1 + Sh' \cdot L \cdot D_1 + Sh \cdot Q_2$$

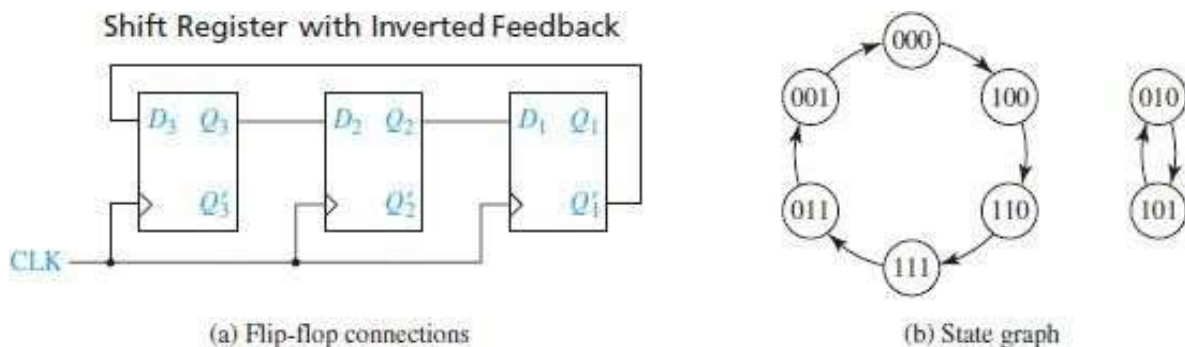
$$Q_0^+ = Sh' \cdot L' \cdot Q_0 + Sh' \cdot L \cdot D_0 + Sh \cdot Q_1$$

A typical application of this register is the conversion of parallel data to serial data. The output from the last flip-flop ( $Q_0$ ) serves as a serial output as well as one of the parallel outputs. The following Figure shows a typical timing diagram.



The first clock pulse loads data into the shift register in parallel. During the next four clock pulses, this data is available at the serial output. Assuming that the register is initially clear ( $Q_3Q_2Q_1Q_0 = 0000$ ), that the serial input is  $SI = 0$  throughout, and that the data inputs  $D_3D_2D_1D_0$  are 1011 during the load time ( $t_0$ ), the resulting waveforms are as shown. Shifting occurs at the end of  $t_1, t_2$ , and  $t_3$ , and the serial output can be read during these clock times. During  $t_4$ ,  $Sh = L = 0$ , so no state change occurs.

The following Figure (a) shows a 3-bit shift register with the  $Q_1$  output from the last flip-flop fed back into the  $D$  input of the first flip-flop.



If the initial state of the register is 000, the initial value of D3 is 1, so after the first clock pulse, the register state is 100. Successive states are shown on the state graph of Figure (b). When the register is in state 001, D3 is 0, and the next register state is 000. Then, successive clock pulses take the register around the loop again. Note that states 010 and 101 are not in the main loop. If the register is in state 010, then a shift pulse takes it to 101 and vice versa; therefore, we have a secondary loop on the stategraph.

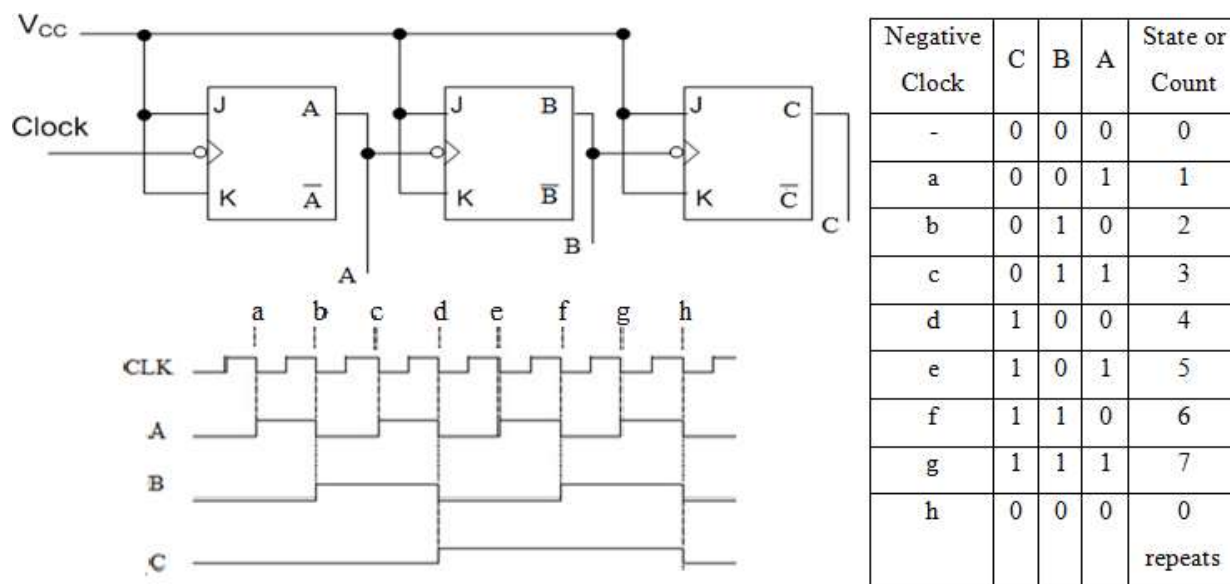
A circuit that goes through a fixed sequence of states is called a *counter*. A shift register with inverted feedback ( $Q1'$  connected to  $D3$ , as shown in above Figure) is often called a *Johnson counter* or *Twisted ring counter*. A shift register with non-inverted feedback (if  $Q1$  connected to  $D3$ , in above Figure) is often called a *Ring counter*.

### **DESIGN OF BINARY COUNTERS:**

A *counter* is a sequential circuit that goes through a prescribed sequence of states up on the application of input pulse. Counters are in two categories –

- *Ripple (Asynchronous) Counter* – consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the clock-pulse input of the next higher-order flip-flop. The flip-flop holding the LSB receives the clock-pulse.
- *Synchronous Counter* – the input pulses/ clock-pulses are applied to all clock-pulse inputs of all the flip-flops simultaneously.

### **Asynchronous Counters:**



**3-Bit Binary Ripple Counter, Waveforms & Truth Table**

The above Figure shows three negative-edge-triggered, JK flip-flops connected in cascade to form a 3-bit ripple counter. The system clock (a square wave), drives flip-flop A. The output of flip-flop A drives B,

and the output of B drives flip-flop C. All the J and K inputs are tied to  $+V_{CC}$ . Hence, flip-flops will toggle with a negative transition at its clock input.

Assume that, the flip-flops are all initially reset to 0 outputs. For every clock NT, flip-flop A will change state. Notice that, the waveform at the output of flip-flop A is one-half the clock frequency.

Since, A acts as clock for B, each time the waveform at A goes low, flip-flop B will toggle. Notice that, the waveform at the output of flip-flop B is one-half the frequency of A and one-fourth the clock frequency.

Since, B acts as clock for C, each time the waveform at B goes low, flip-flop C will toggle. The waveform at the output of flip-flop C is one-half the frequency of B and one-eighth the clock frequency.

**Problem:** What is the clock frequency of a 3-bit ripple counter, if the period of the MSB waveform is  $24 \mu s$ ?

**Solution:** Since there are eight clock cycles in one cycle of MSB, the period of the clock must be  $24/8 = 3 \mu s$ . The clock frequency must be  $1/(3 \times 10^{-6}) = 333 \text{ KHz}$ .

**NOTE:**

1. A binary ripple counter in straight binary sequence will be as shown in the above table. A ripple counter having  $n$  flip-flops will have  $2^n$  output conditions. For example, the three-flip-flop counter has  $2^3 = 8$  output conditions (000 to 111).

**NOTE:**

<i>No. of Flip-Flops</i>	1	2	3	4	5	$n$
<i>No. of States</i>	2	4	8	16	32	$2^n$

2. A three-flip-flop counter is often referred to as modulus-8 (or Mod-8) counter, since it has eight states. The *modulus* of a counter is the total number of states through which the counter can progress.

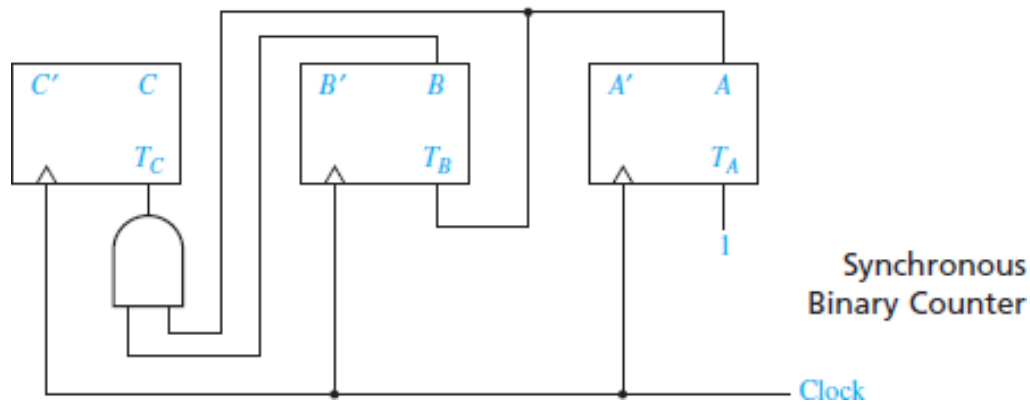
**Problem:** How many flip-flops are required to construct a mod-128 counter? A mod-32 counter? What is the largest decimal number that can be stored in a mod-64 counter?

**Solution:** A mod-128 counter must have seven flip-flops, since  $2^7 = 128$ . Five flip-flops are needed to construct a mod-32 counter. The largest decimal number that can be stored in a mod-64 (six flip-flops) counter is  $111111 = 63$ .

**Synchronous Counters:**

*Synchronous* means the operation of the flip-flops is synchronized by a common clock pulse so that when several flip-flops must change state, the state changes occur simultaneously.

**Synchronous Binary Counters Using T Flip-Flops:** Consider the following Figure, a binary counter using three T flip-flops to count clock pulses.



All the flip-flops change state a short time following the rising edge of the input pulse. The state of the counter is determined by the states of the individual flip-flops; for example, if flip-flop C is in state 0, B in state 1, and A in state 1, the state of the counter is 011.

Initially, assume that all flip-flops are set to the 0 state. When a clock pulse is received, the counter will change to state 001; when a second pulse is received, the state will change to 010, etc. The sequence of flip-flop states is  $CBA = 000, 001, 010, 011, 100, 101, 110, 111, 000, \dots$ . Note that, when the counter reaches state 111, the next pulse resets it to the 000 state, and then the sequence repeats.

First, design the counter by inspection of the counting sequence; then, use a systematic procedure which can be generalized to other types of counters. The problem is to determine the flip-flop inputs— $T_C, T_B,$  and  $T_A$ . From the preceding counting sequence, observe that A changes state every time a clock pulse is received. Because A changes state on every rising clock edge,  $T_A$  must equal 1. Next, observe that B changes state only if  $A = 1$ . Therefore, A is connected to  $T_B$  as shown, so that if  $A = 1$ , B will change state when a rising clock edge occurs. Similarly, C changes state when a rising clock edge occurs only if B and A are both 1. Therefore, an AND gate is connected to  $T_C$  so that C will change state if  $B = 1$  and  $A = 1$  when a rising clock edge occurs.

Now, verify that the circuit of above Figure counts properly by tracing signals through the circuit. Initially,  $CBA = 000$ , so only  $T_A$  is 1 and the state will change to 001 when the first active clock edge arrives. Then,  $T_B = T_A = 1$ , and the state will change to 010 when the second active clock arrives. This process continues until finally when state 111 is reached,  $T_C = T_B = T_A = 1$ , and all flip-flops return to the 0 state.

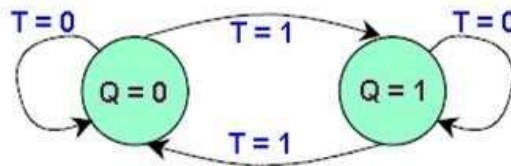
A state table (the following Table) shows the present state of flip-flops C, B, and A (before a clock pulse is received) and the corresponding next state (after the clock pulse is received). For example, if the flip-flops are in state  $CBA = 011$  and a clock pulse is received, the next state will be  $C^+ = B^+ = A^+ = 100$ . Although the clock is not explicit in the table, it is understood to be the input that causes the counter to go

to the next state in sequence. A third column in the table is used to derive the inputs for TC, TB, and TA. Whenever the entries in the A and A<sup>+</sup> columns differ, flip-flop A must change state and TA must be 1. Similarly, if B and B<sup>+</sup> differ, B must change state so TB must be 1. For example, if CBA = 011, C<sup>+</sup>B<sup>+</sup>A<sup>+</sup> = 100, all three flip-flops must change state, so TCTBTA = 111.

State Table for Binary Counter	Present State			Next State			Flip-Flop Inputs		
	C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	T <sub>C</sub>	T <sub>B</sub>	T <sub>A</sub>
	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	1	1	0	0	1	1
	1	1	0	1	1	1	0	0	1
	1	1	1	0	0	0	1	1	1

T	Q	Q <sup>+</sup>
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table



State Diagram

Q → Q <sub>n+1</sub>	T
0	0
0	1
1	0
1	1

Excitation Table

TC, TB, and TA are now derived from the table as functions of C, B, and A. By inspection, TA = 1. The following Figure shows the Karnaugh maps for TC and TB, from which; TC = BA and TB = A. These equations yield the same circuit derived previously.

Karnaugh Maps  
for Binary Counter

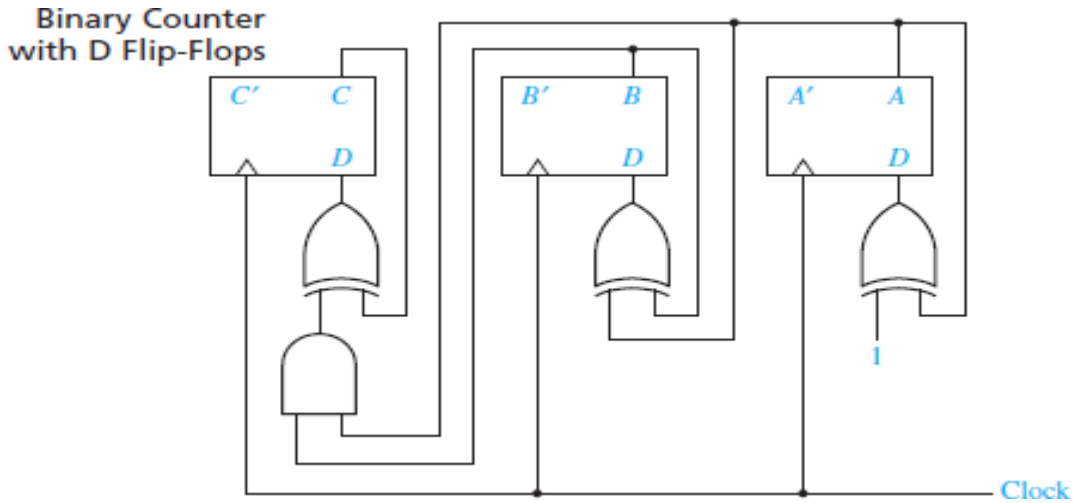
BA	C	
	0	1
00	0	0
01	0	0
11	1	1
10	0	0

$T_C$

BA	C	
	0	1
00	0	0
01	1	1
11	1	1
10	0	0

$T_B$

**Synchronous Binary Counters Using D Flip-Flops:** Next, redesign the binary counter to use D flip-flops instead of T flip flops. The easiest way to do this is to convert each D flip-flop to a T flip-flop by adding an XOR (exclusive-OR) gate, as shown in the following Figure (The rightmost XOR gate can be replaced with an inverter because  $A \oplus 1 = A$ ).

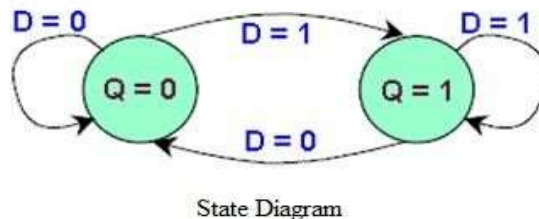


We can also derive the D flip-flop inputs for the binary counter starting with its state table (given below).

Present State			Next State			Flip-Flop Inputs		
C	B	A	C	B	A	D <sub>C</sub>	D <sub>B</sub>	D <sub>A</sub>
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

D	Q	Q <sup>-</sup>
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic Table

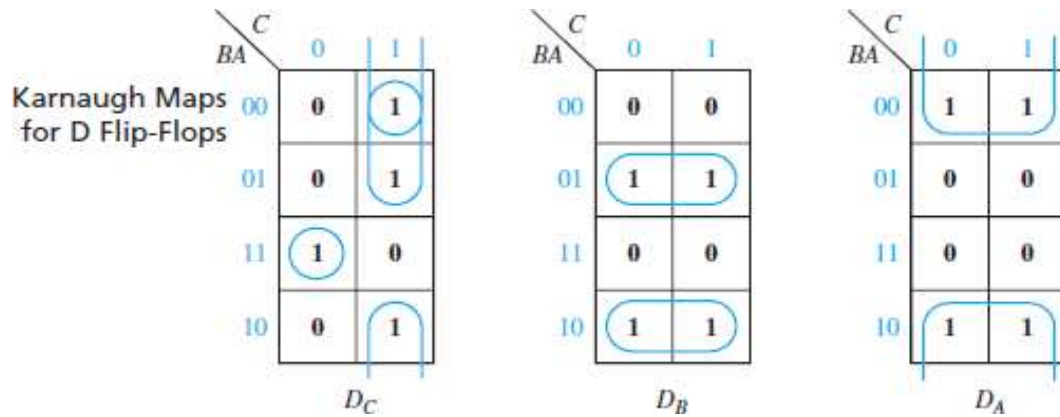


State Diagram

Q	Q <sub>n+1</sub>	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

For a D flip-flop,  $Q^+ = D$ . By inspection of the state table, the maps for  $D_A$ ,  $D_B$  and  $D_C$  are plotted as follows and D input equations are derived. This give the same logic circuit as was obtained by inspection.

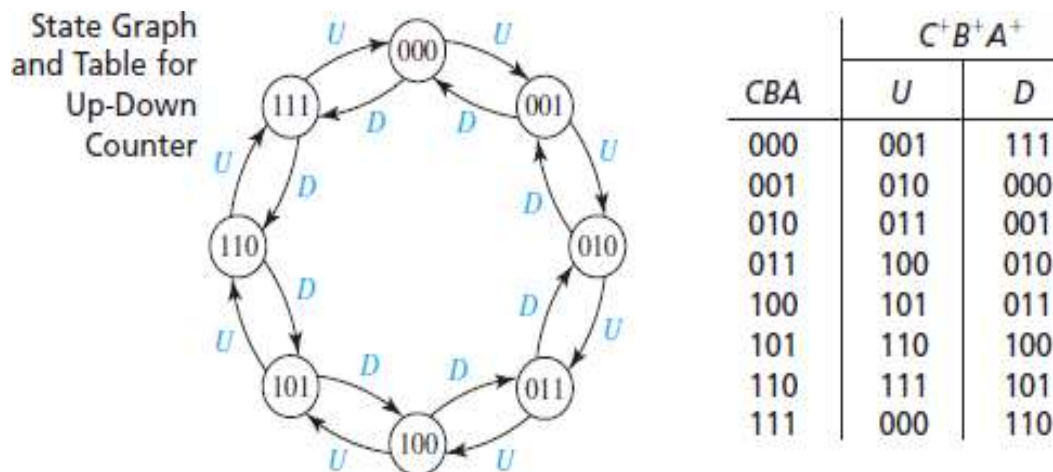


$$D_A = A^+ = A'$$

$$D_B = B^+ = BA' + B'A = B \oplus A$$

$$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$

**Binary Up-Down Counter:** The state graph and table for an up-down counter are shown in the following Figure.



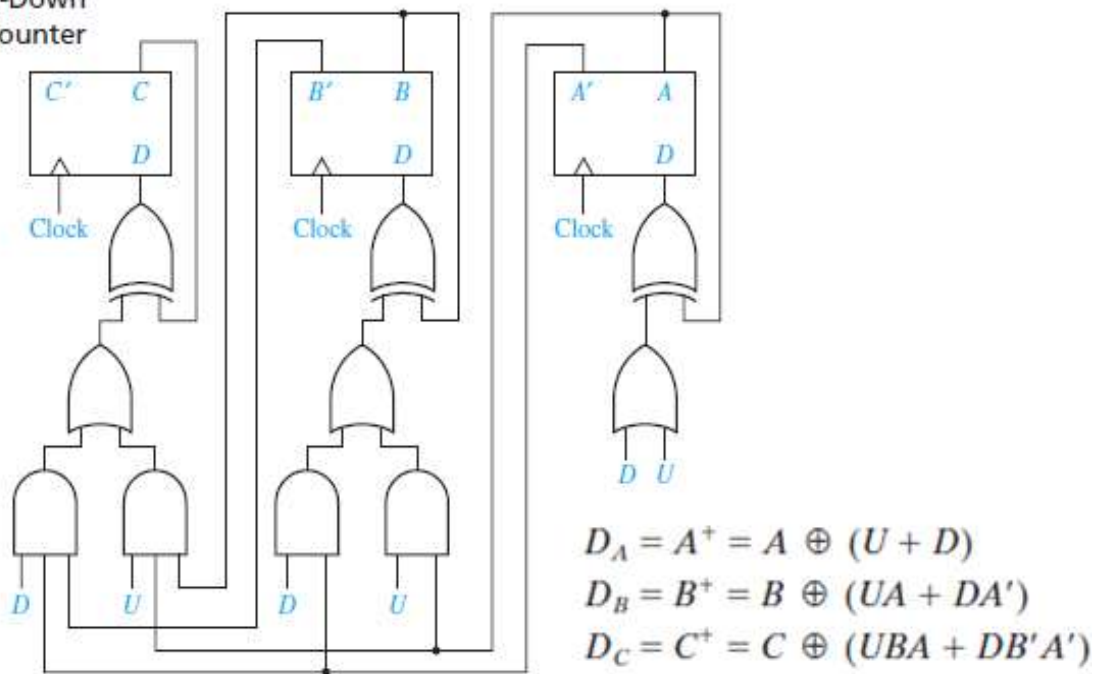
When  $U = 1$ , the counter counts up in the sequence 000, 001, 010, 011, 100, 101, 110, 111, 000 . . . When  $D = 1$ , the counter counts down in the sequence 000, 111, 110, 101, 100, 011, 010, 001, 000 . . . When  $U = D = 0$ , the counter state does not change, and  $U = D = 1$  is not allowed.

By inspection of the table in above Figure, we can verify that these are the correct equations for a down counter. For every row of the table,  $A^+ = A'$ , so A changes state every clock cycle. For those rows where  $A = 0$ ,  $B^+ = B'$ . For those rows where  $B = 0$  and  $A = 0$ ,  $C^+ = C$ .

The up-down counter can be implemented using D flip-flops and gates, as shown in the following Figure. The corresponding logic equations are also given in the following Figure.



### Binary Up-Down Counter



When  $U = 1$  and  $D = 0$ , these equations reduce to equations for a binary up counter.

$$D_A = A^+ = A'$$

$$D_B = B^+ = BA' + B'A = B \oplus A$$

$$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$

When  $U = 0$  and  $D = 1$ , these equations reduce to –

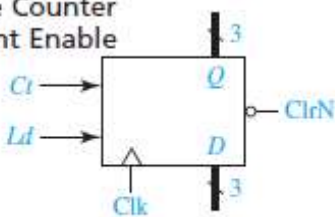
$$D_A = A^+ = A \oplus 1 = A' \quad (A \text{ changes state every clock cycle})$$

$$D_B = B^+ = B \oplus A' \quad (B \text{ changes state when } A = 0)$$

$$D_C = C^+ = C \oplus B'A' \quad (C \text{ changes state when } B = A = 0)$$

**Loadable Counter:** The counter shown in the following Figure (a) has two control signals  $Ld$  (load) and  $Ct$  (count).

Loadable Counter with Count Enable



(a)

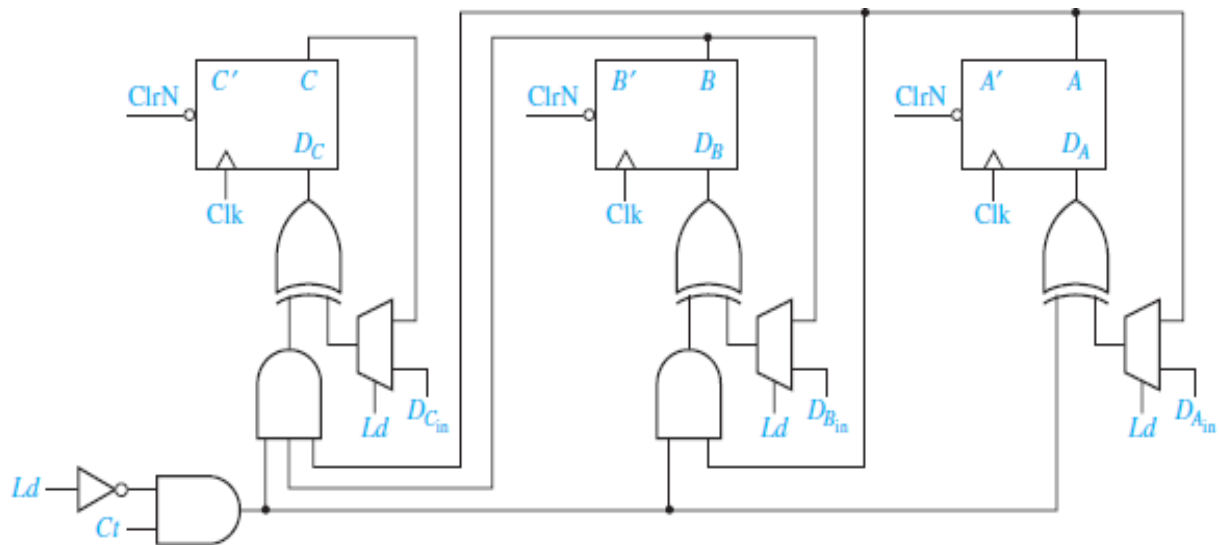
ClrN	Ld	Ct	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	
0	X	X	0	0	0	
1	1	X	D <sub>C</sub>	D <sub>B</sub>	D <sub>A</sub>	(load)
1	0	0	C	B	A	(no change)
1	0	1	Present state + 1			

(b)

When  $Ld = 1$  binary data is loaded into the counter on the rising clock edge, and when  $Ct = 1$ , the counter is incremented on the rising clock edge. When  $Ld = Ct = 0$ , the counter holds its present state. When  $Ld = Ct = 1$ , load overrides count, and data is loaded into the counter. The counter also has an asynchronous

clear input that clears the counter when  $ClrN$  is 0. Figure (b) summarizes the counter operation. All state changes occur on the rising edge of the clock (except for the asynchronous clear).

The following Figure shows how the loadable counter can be implemented using flip-flops, MUXes, and gates.



When  $Ld = 1$ , each MUX selects a  $D_i$  input, and because the output of each AND gate is 0, the output of each XOR gate is  $D_i$ , which gets stored in a flip-flop. When  $Ld = 0$  and  $Ct = 1$ , each MUX selects one of the flip-flop outputs ( $C$ ,  $B$ , or  $A$ ). The circuit then becomes equivalent to a binary counter, and the counter is incremented on the rising clock edge. The next-state equations for the counter of above Figure are –

$$A^+ = D_A = (Ld' \cdot A + Ld \cdot D_{Ain}) \oplus Ld' \cdot Ct$$

$$B^+ = D_B = (Ld' \cdot B + Ld \cdot D_{Bin}) \oplus Ld' \cdot Ct \cdot A$$

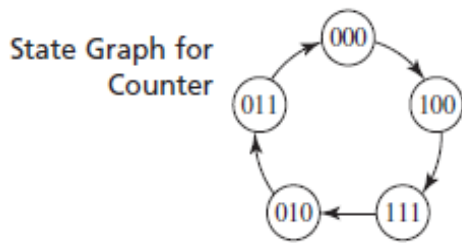
$$C^+ = D_C = (Ld' \cdot C + Ld \cdot D_{Cin}) \oplus Ld' \cdot Ct \cdot B \cdot A$$

When  $Ld = 0$  and  $Ct = 1$ , these equations reduce to  $A^+ = A'$ ,  $B^+ = B \oplus A$ , and  $C^+ = C \oplus BA$ , which are the equations previously derived for a 3-bit counter.

### COUNTER FOR OTHER SEQUENCES:

In some applications, the sequence of states of a counter is not in straight binary order. The following Figure shows the state graph for such a counter. The arrows indicate the state sequence.

If this counter is started in state 000, the first clock pulse will take it to state 100, the next pulse to 111, etc. The clock pulse is implicitly understood to be the input to the circuit and not shown on the graph. The corresponding state table for the counter is given the following Table. Note that the next state is unspecified for the present states 001, 101, and 110.



State Table

C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

We will design the counter specified by the above Table using T flip-flops. We could derive TC, TB, and TA directly from this table, as in the preceding example. However, it is often more convenient to plot next-state maps showing C<sup>+</sup>, B<sup>+</sup>, and A<sup>+</sup> as functions of C, B, and A, and then derive TC, TB, and TA from these maps.

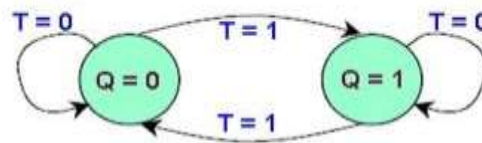
Present State			Next State			Flip-Flop Inputs		
C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	T <sub>C</sub>	T <sub>B</sub>	T <sub>A</sub>
0	0	0	1	0	0			
0	0	1	-	-	-			
0	1	0	0	1	1			
0	1	1	0	0	0			
1	0	0	1	1	1			
1	0	1	-	-	-			
1	1	0	-	-	-			
1	1	1	0	1	0			

From the first row of the table, the CBA = 000; and hence, the C<sup>+</sup>, B<sup>+</sup>, and A<sup>+</sup> columns are filled in with 1, 0, and 0, respectively. From the second row, the CBA = 001; all three columns are filled in with don't-cares. From the third row, the CBA = 010; and the C<sup>+</sup>, B<sup>+</sup>, and A<sup>+</sup> columns are filled with 0, 1, and 1, respectively. The next-state columns can be quickly completed by continuing in this manner.

Next, we will derive the maps for the T inputs from the next-state maps.

T	Q	Q <sup>-</sup>
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table



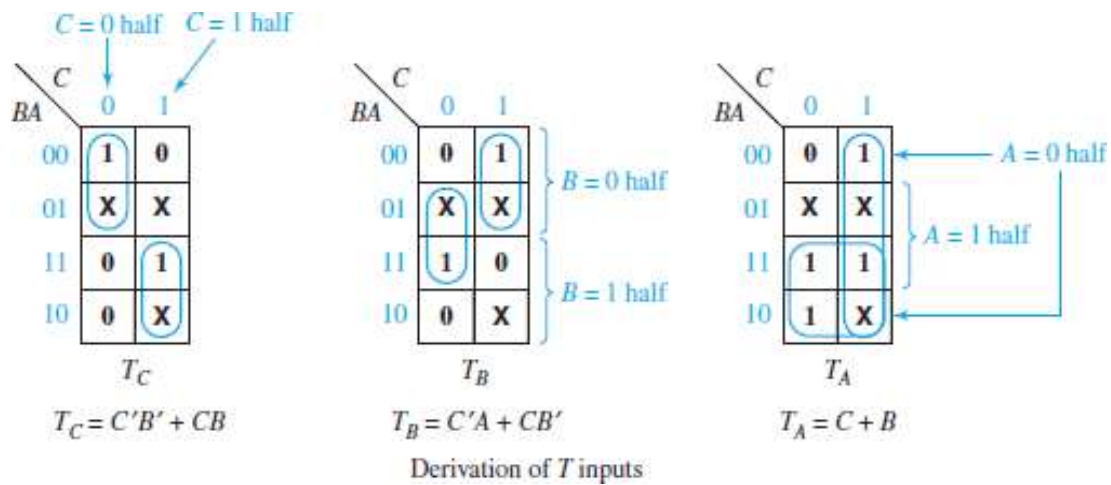
State Diagram

Q → Q <sub>n-1</sub>	T
0	0
0	1
1	0
1	1

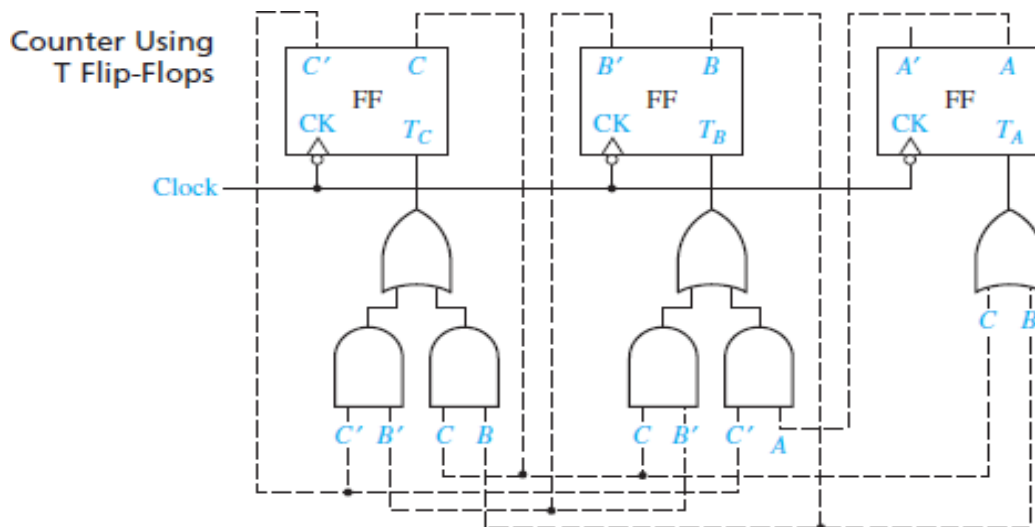
Excitation Table

Present State			Next State			Flip-Flop Inputs		
C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	T <sub>C</sub>	T <sub>B</sub>	T <sub>A</sub>
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	x	x	x
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	1	0	1

Now, draw the K-maps for T<sub>C</sub>, T<sub>B</sub>, and T<sub>A</sub> separately, and derive the expressions.

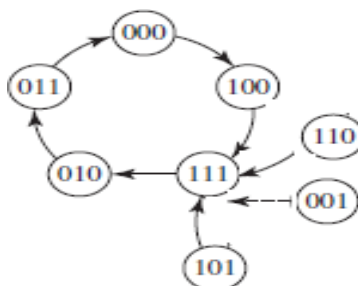


Finally, draw the counter circuit based on the expressions.



**NOTE:** In the above problem, for the design, initially, when the power is switched on, if circuit enters to an invalid state (state 1 or state 5 or state 6), then *lock-in condition* results. The designed counter will not give proper result. For this reason, all of the don't-care states in a counter should be checked to make sure that they eventually lead into the main counting sequence unless a power-up reset is provided.

The solution is *self correcting counter*.



**Counter Design Using D Flip-Flops:** For a D flip-flop,  $Q^+ = D$ , so the D input map is identical with the next-state map.

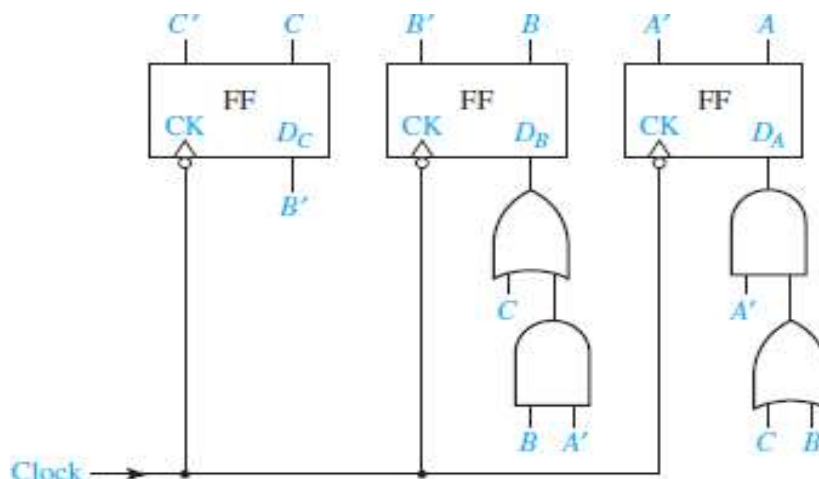
Present State			Next State			Flip-Flop Inputs		
C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	D <sub>C</sub>	D <sub>B</sub>	D <sub>A</sub>
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	x	x	x
0	1	0	0	1	1	0	1	1
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	0	1	0

Draw the K-map; and from the map, we get the following expressions:

$$D_C = C^+ = B' \quad D_B = B^+ = C + BA'$$

$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Corresponding counter circuit is –



## COUNTER DESIGN USING S-R AND J-K FLIP-FLOPS:

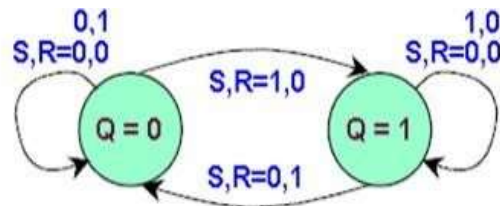
The procedures used to design a counter with S-R OR flip-flops are similar to the procedures discussed. However, instead of deriving an input equation for each D or T flip-flop, the S and R input equations must be derived.

The following Table describes the behavior of S-R flip-flops:

S	R	Q	Q <sup>+</sup>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

inputs not allowed

Characteristic Table

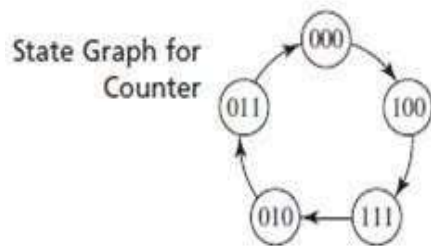


State Diagram

Q	Q <sup>+</sup>	S	R	Q	Q <sup>+</sup>	S	R
0	0	0	0	0	0	0	X
0	1	0	1	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	1	1	1	1
1	1	0	0	-	-	-	-
1	1	1	0	-	-	-	-

Excitation Table

Next, we will redesign the counter for following Figure using S-R flip-flops.



State Graph for Counter

C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	S <sub>C</sub>	R <sub>C</sub>	S <sub>B</sub>	R <sub>B</sub>	S <sub>A</sub>	R <sub>A</sub>
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs.

BA	0	1
00		
01	X	X
11	X	1
10	X	X

$R_C = A$

BA	0	1
00	1	X
01	X	X
11		
10		X

$S_C = B'$

BA	0	1
00	X	
01	X	X
11	1	
10		X

$R_B = C'A$

BA	0	1
00		1
01	X	X
11		X
10	X	X

$S_B = C$

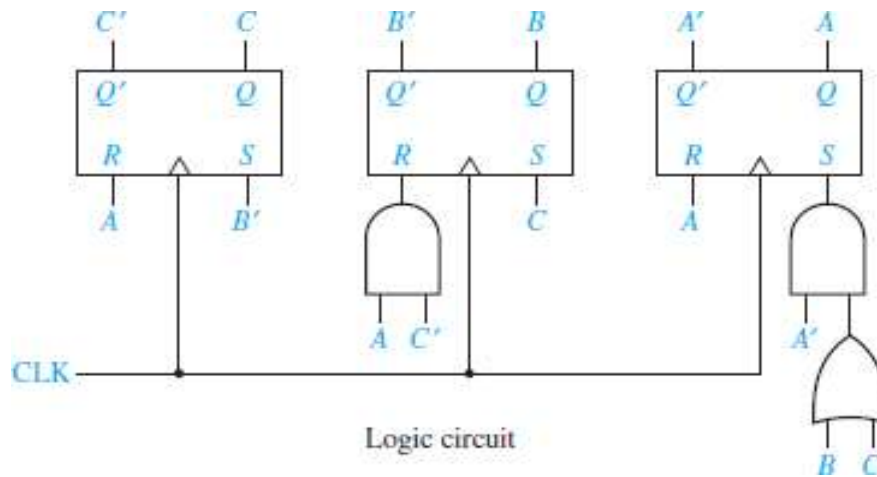
BA	0	1
00	X	
01	X	X
11	1	1
10		X

$R_A = A$

BA	0	1
00		1
01	X	X
11		
10	1	X

$S_A = CA' + BA'$   
 $= A'(C + B)$

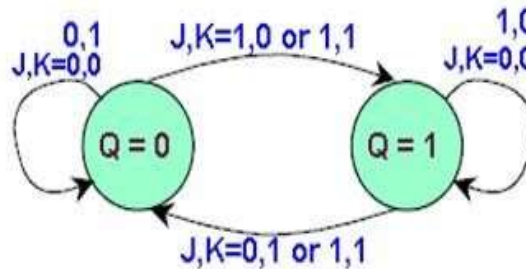
S-R flip-flop equations



The procedure used to design a counter with J-K flip-flops is very similar to that used for S-R flip-flops. The J-K flip-flop is similar to the S-R flip-flop except that J and K can be 1 simultaneously, in which case the flip-flop changes state.

J	K	Q	Q <sup>+</sup>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

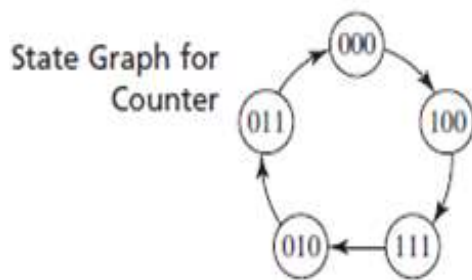
Characteristic Table



Q	Q <sup>+</sup>	J	K	Q	Q <sup>+</sup>	J	K
0	0	0	0	0	0	0	X
0	0	0	1	0	1	1	X
0	1	1	0	1	0	X	1
0	1	1	1	1	1	X	0
1	0	0	1	1	0	X	1
1	1	0	0	1	1	X	0
1	1	1	0	1	1	X	0

Excitation Table

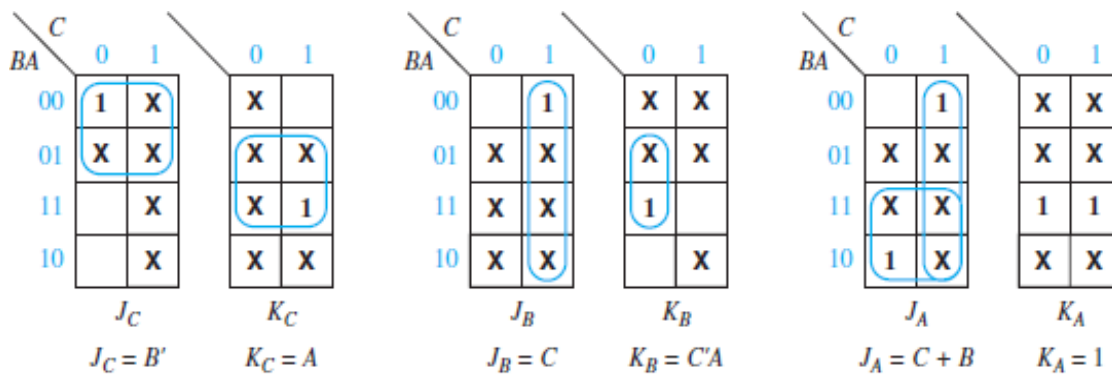
Now, we will redesign the counter for following Figure using JK flip-flops.



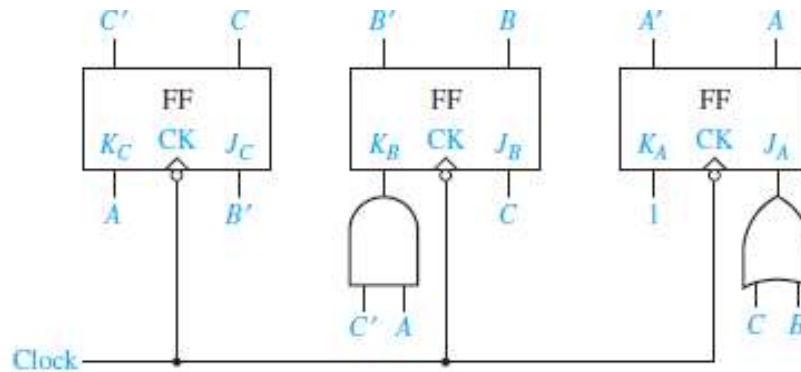
C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>	J <sub>C</sub>	K <sub>C</sub>	J <sub>B</sub>	K <sub>B</sub>	J <sub>A</sub>	K <sub>A</sub>
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs; and finally draw the counter circuit using J-K flip-flops.





J-K flip-flop input equations



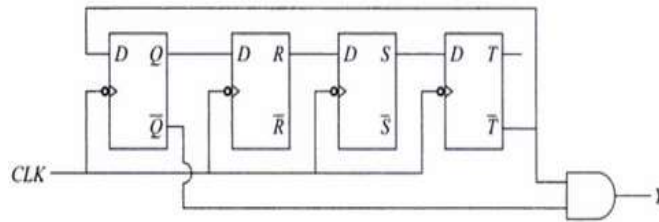
Logic circuit (omitting the feedback lines)

**Homework:**

1] Construct a 4-bit Johnson counter using

(a) D flip-flops

(b) J-K flip-flops.



Clock	Serial in = $\bar{T}$	Q	R	S	T	$Y = \bar{Q}\bar{T}$
0	1	0	<u>0</u>	<u>0</u>	<u>0</u>	1
1	1	1	<u>0</u>	<u>0</u>	<u>0</u>	0
2	1	1	<u>1</u>	<u>0</u>	<u>0</u>	0
3	1	1	<u>1</u>	<u>1</u>	<u>0</u>	0
4	0	1	<u>1</u>	<u>1</u>	<u>1</u>	0
5	0	0	<u>1</u>	<u>1</u>	<u>1</u>	0
6	0	0	<u>0</u>	<u>1</u>	<u>1</u>	0
7	0	0	<u>0</u>	<u>0</u>	<u>1</u>	0
8	1	0	<u>0</u>	<u>0</u>	<u>0</u>	1
9	1	1	<u>0</u>	<u>0</u>	<u>0</u>	0
						repeats

2] Design a 3-bit counter which counts in the sequence:

001, 011, 010, 110, 111, 101, 100, (repeat) 001, . . .

(a) Use D flip-flops

(b) Use T flip-flops

(c) Use S-R flip-flops

(d) Use J-K flip-flops.

3] Design a 3-bit counter which counts in the sequence:

001, 100, 101, 111, 110, 010, 011, 001, . . .

(a) Use D flip-flops

(b) Use J-K flip-flops

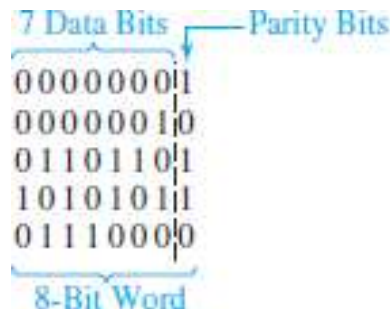
(c) Use T flip-flops

(d) Use S-R flip-flops

(e) What will happen if the counter of (a) is started in state 000?

### A SEQUENTIAL PARITY CHECKER:

When binary data is transmitted or stored, an extra bit (called a *parity bit*) is frequently added for purposes of error detection. For example, if data is being transmitted in groups of 7 bits, an eighth bit can be added to each group of 7 bits to make the total number of 1's in each block of 8 bits an odd number. When the total number of 1 bits in the block (including the parity bit) is odd, we say that the *parity is odd*. Alternately, the parity bit could be chosen such that the total number of 1's in the block is even, in which case we would have *even parity*. Some examples of 8-bit words with odd parity are –

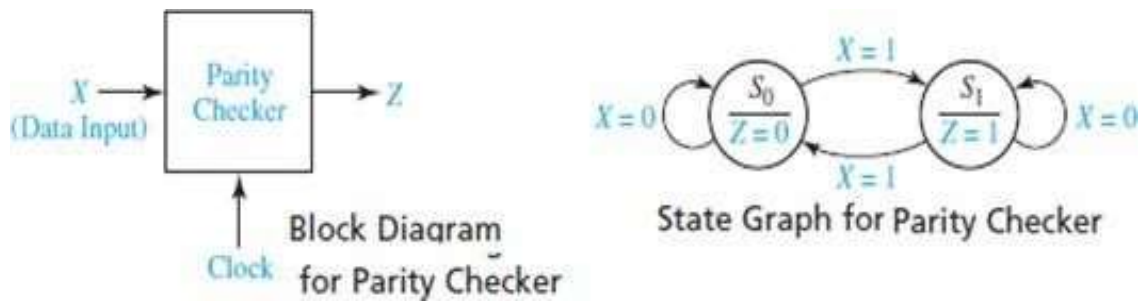


If any single bit in the 8-bit word is changed from 0 to 1 or from 1 to 0, the parity is no longer odd. Thus, if any single bit error occurs in transmission of a word with odd parity, the presence of this error can be detected because the number of 1 bits in the word has been changed from odd to even.

As a simple example of a sequential circuit which has one input in addition to the clock, we will design a parity checker for serial data. (Serial implies that the data enters the circuit sequentially, one bit at a time.) This circuit has the form shown in the following Figure.

When a sequence of 0's and 1's is applied to the X input, the output of the circuit should be  $Z = 1$ , if the total number of 1 inputs received is odd; that is, the output should be 1 if the input parity is odd. Thus, if data which originally had odd parity is transmitted to the circuit, a final output of  $Z = 0$  indicates that an error in transmission has occurred.

The value of X is read at the time of the active clock edge. The X input must be synchronized with the clock so that it assumes its next value before the next active clock edge. The clock input is necessary in order to distinguish consecutive 0's or consecutive 1's on the X input.



The sequential circuit must *remember* whether the total number of 1 inputs received is even or odd; therefore, only two states are required. We will designate these states as  $S_0$  and  $S_1$ , corresponding respectively to an even number of 1's received and an odd number of 1's received. We will start the circuit in state  $S_0$  because initially zero 1's have been received, and zero is an even number.

As indicated in state graph (above Figure), if the circuit is in state  $S_0$  (even number of 1's received) and  $X = 0$  is received, the circuit must stay in  $S_0$  because the number of 1's received is still even. However, if  $X = 1$  is received, the circuit goes to state  $S_1$  because the number of 1's received is then odd.

Similarly, if the circuit is in state  $S_1$  (odd number of 1's received) a 0 input causes no state change, but a 1 causes a change to  $S_0$  because the number of 1's received is then even.

The output Z should be 1 whenever the circuit is in state  $S_1$  (odd number of 1's received). The output is listed below the state on the state graph.

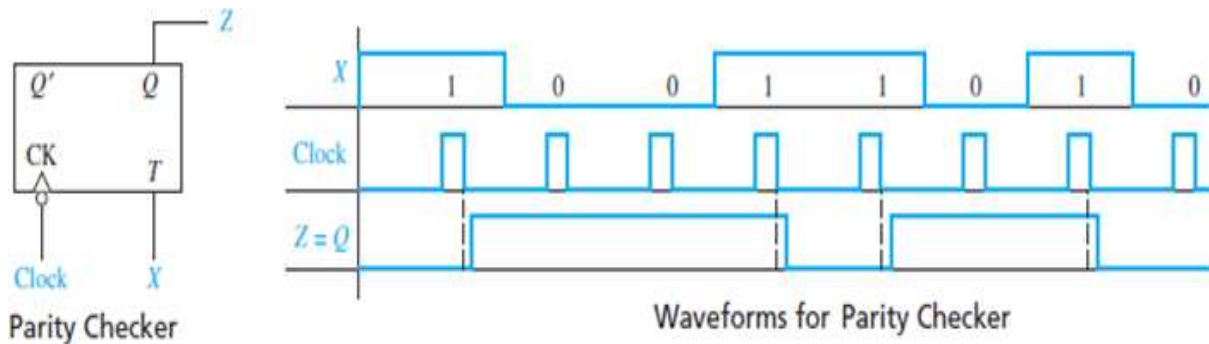
The following Table (a) gives the same information as the state graph in tabular form. The table shows that if the present state is  $S_0$ , the output is  $Z = 0$ , and if the input is  $X = 1$ , the next state will be  $S_1$ .

Present State	(a) Next State		Present Output	Q	$Q^+$		T		Z
	X = 0	X = 1			X = 0	X = 1	X = 0	X = 1	
$S_0$	$S_0$	$S_1$	0	0	0	1	0	1	0
$S_1$	$S_1$	$S_0$	1	1	1	0	0	1	1

State Table for Parity Checker

Since only two states are required, a single flip-flop (Q) will suffice. We will let  $Q = 0$  correspond to state  $S_0$  and  $Q = 1$  corresponds to  $S_1$ . We can then set up a table which shows the next state of flip-flop Q as a function of the present state and X. If we use a T flip-flop, T must be 1 whenever Q and  $Q^+$  differ. From the above Table (b), the T input must be 1 whenever  $X = 1$ . The following Figure shows the resulting circuit.

The following Figure also shows the output waveform for the circuit. When  $X = 1$ , the flip-flop changes state after the falling edge of the clock. Note that the final value of  $Z$  is 0 because an even number of 1's was received. If the number of 1's received had been odd, the final value of  $Z$  would be 1. In this case, it would be necessary to reset the flip-flop to the proper initial state ( $Q = 0$ ) before checking the parity of another input sequence.



### STATE TABLES AND GRAPHS:

The *state tables* and *graphs* provides a more systematic approach which is useful for the analysis of larger circuits and which leads to a general synthesis procedure for sequential circuits. The *state table* specifies the next state and output of a sequential circuit in terms of its present state and input.

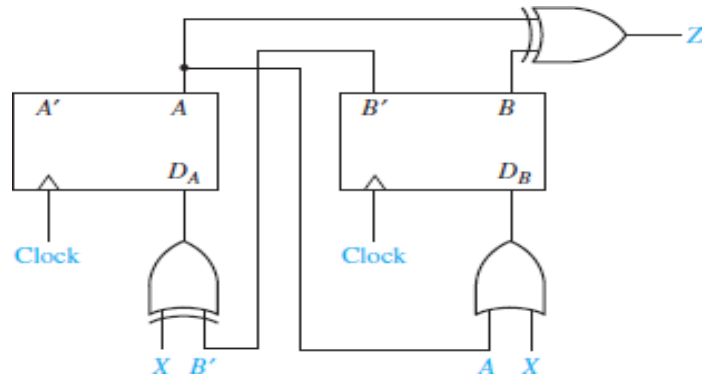
The following method can be used to construct the state table:

1. Determine the flip-flop input equations and the output equations from the circuit.
2. Derive the next-state equation for each flip-flop from its input equations, using one of the following relations:

D flip-flop	$Q^+ = D$
D-CE flip-flop	$Q^+ = D \cdot CE + Q \cdot CE'$
T flip-flop	$Q^+ = T \oplus Q$
S-R flip-flop	$Q^+ = S + R'Q$
J-K flip-flop	$Q^+ = JQ' + K'Q$

3. Plot a next-state map for each flip-flop.
4. Combine these maps to form the state table. Such a state table, which gives the next state of the flip-flops as a function of their present state and the circuit inputs, is frequently referred to as a *transition table*.

**Example:** Derive the state table for the circuit of the following Figure (Moore Model):



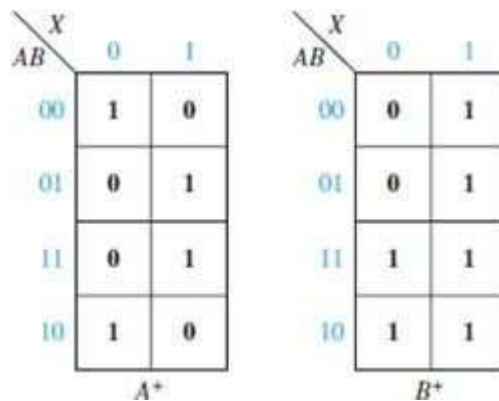
1. The flip-flop input equations and output equation are

$$D_A = X \oplus B' \qquad D_B = X + A \qquad Z = A \oplus B$$

2. The next-state equations for the flip-flops are

$$A^+ = X \oplus B' \qquad B^+ = X + A$$

3. The corresponding maps are

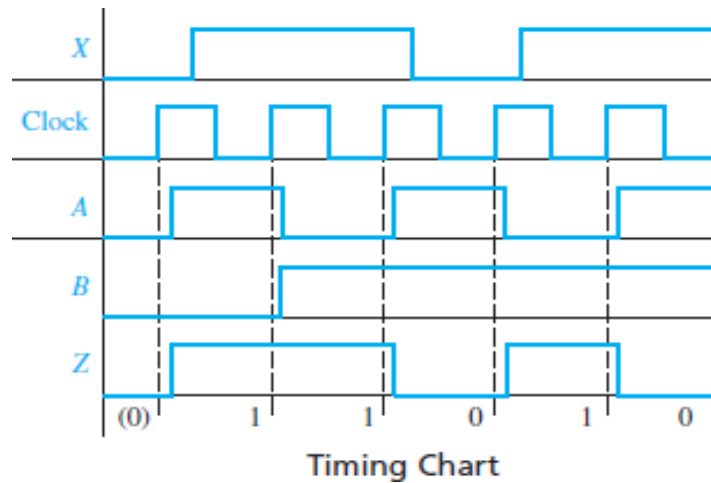


AB	(a) $A^+B^+$		Z	Present State	(b) Next State		Present Output (Z)
	X = 0	X = 1			X = 0	X = 1	
00	10	01	0	$S_0$	$S_3$	$S_1$	0
01	00	11	1	$S_1$	$S_0$	$S_2$	1
11	01	11	0	$S_2$	$S_1$	$S_2$	0
10	11	01	1	$S_3$	$S_2$	$S_1$	1

State Tables

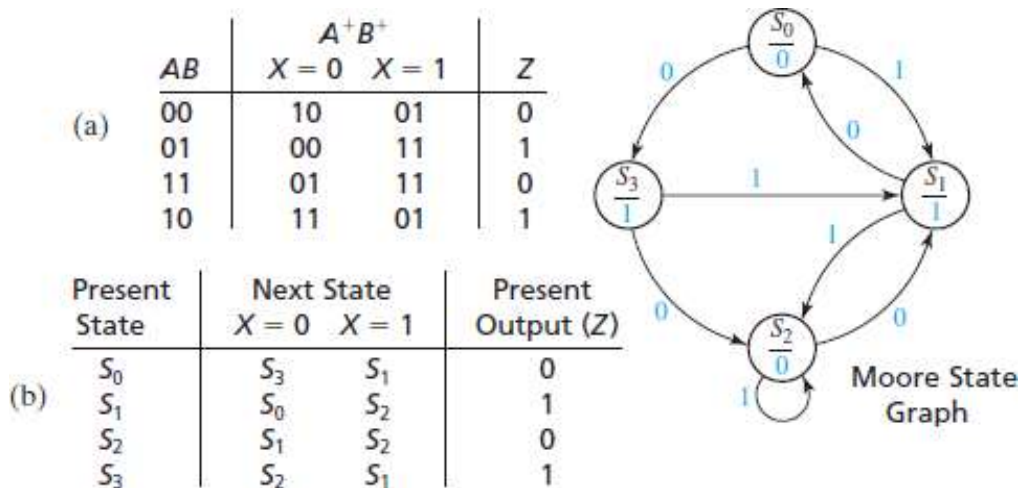
4. Combining these maps yields the transition table (Table (a)), which gives the next state of both flip-flops ( $A^+B^+$ ) as a function of the present state and input. The output function Z is then added to the table. In this example, the output depends only on the present state of the flip-flops and not on the input, so only a single output column is required.

Using the above Table (a), we can construct the timing chart for some given input sequence and specified initial state.



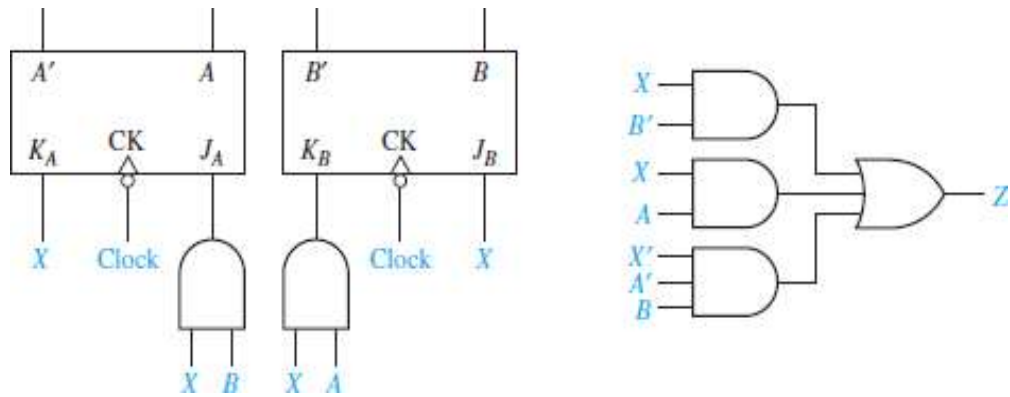
Initially  $AB = 00$  and  $X = 0$ , so  $Z = 0$  and  $A+B^+ = 10$ . This means that after the rising clock edge, the flip-flop state will be  $AB = 10$ . Then, with  $AB = 10$ , the output is  $Z = 1$ . The next input is  $X = 1$ , so  $A+B^+ = 01$  and the state will change after the next rising clock edge. Continuing in this manner, we can complete the timing chart.

If we are not interested in the individual flip-flop states, we can replace each combination of flip-flop states with a single symbol which represents the state of the circuit. Replacing 00 with  $S_0$ , 01 with  $S_1$ , 11 with  $S_2$ , and 10 with  $S_3$  in Table (a) yields Table (b). The Z column is labeled Present Output because it is the output associated with the Present State.



The state graph of (given in above Figure) represents Table (b). Each node of the graph represents a state of the circuit, and the corresponding output is placed in the circle below the state symbol. The arc joining two nodes is labeled with the value of X which will cause a state change between these nodes. Thus, if the circuit is in state  $S_0$  and  $X = 1$ , a clock edge will cause a transition to state  $S_1$ .

**Example:** Derive the state table for the circuit of the following Figure (Mealy Model):



The next-state and output equations are –

$$A^+ = J_A A' + K_A A = XBA' + X'A$$

$$B^+ = J_B B' + K_B B = XB' + (AX)'B = XB' + X'B + A'B$$

$$Z = X'A'B + XB' + XA$$

The next-state and output maps (following Figure) combine to form the transition table (shown in Table (a)). Given values for A, B, and X, the current value of the output is determined from the Z column of this table, and the states of the flip-flops after the active clock edge are determined from the A<sup>+</sup>B<sup>+</sup> columns.

	X		
		0	1
AB			
00		0	0
01		0	1
11		1	0
10		1	0
		A <sup>+</sup>	

	X		
		0	1
AB			
00		0	1
01		1	1
11		1	0
10		0	1
		B <sup>+</sup>	

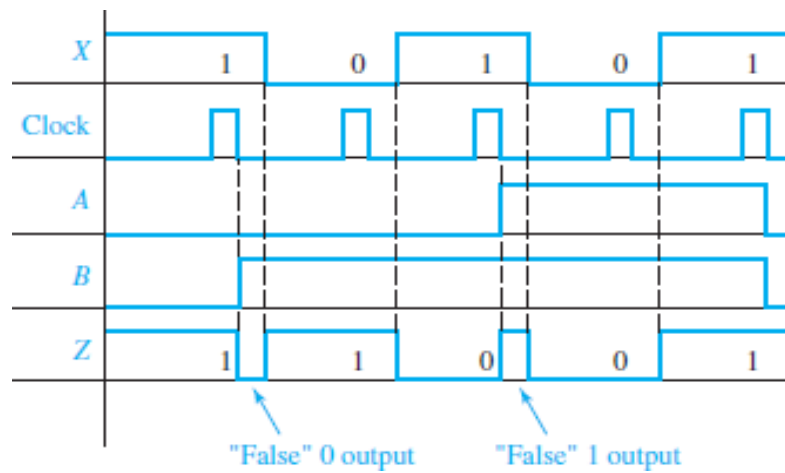
	X		
		0	1
AB			
00		0	1
01		1	0
11		0	1
10		0	1
		Z	

We can construct the timing chart of (given in the following Figure) using Table (a) (given below).

(a)				
	A <sup>+</sup> B <sup>+</sup>		Z	
	X = 0	1	X = 0	1
AB				
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)				
	Next State		Present Output	
	X = 0	1	X = 0	1
Present State				
S <sub>0</sub>	S <sub>0</sub>	S <sub>1</sub>	0	1
S <sub>1</sub>	S <sub>1</sub>	S <sub>2</sub>	1	0
S <sub>2</sub>	S <sub>2</sub>	S <sub>0</sub>	0	1
S <sub>3</sub>	S <sub>3</sub>	S <sub>1</sub>	0	1





Initially with  $A = B = 0$  and  $X = 1$ , the table shows that  $Z = 1$  and  $A^+B^+ = 01$ . Therefore, after the falling clock edge, the state of flip-flop B will change to 1, as indicated in above Figure. Now, from the 01 row of the table, if X is still 1, the output will be 0 until the input is changed to  $X = 0$ . Then, the output is  $Z = 1$ , and the next falling clock edge produces no state change. Finish stepping through the state table in this manner and verify that A, B, and Z are as given in the above Figure.

If we let  $AB = 00$  correspond to circuit state  $S_0$ , 01 to  $S_1$ , 11 to  $S_2$ , and 10 to  $S_3$ , we can construct the state table as given in Table (b) and the state graph of the following Figure.

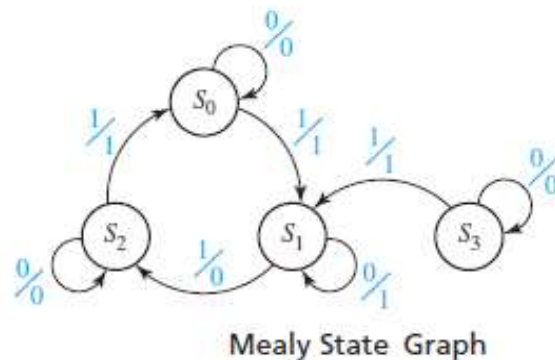
Mealy State Tables

(a)

AB	$A^+B^+$		Z	
	X = 0	1	X = 0	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)

Present State	Next State		Present Output	
	X = 0	1	X = 0	1
$S_0$	$S_0$	$S_1$	0	1
$S_1$	$S_1$	$S_2$	1	0
$S_2$	$S_2$	$S_0$	0	1
$S_3$	$S_3$	$S_1$	0	1

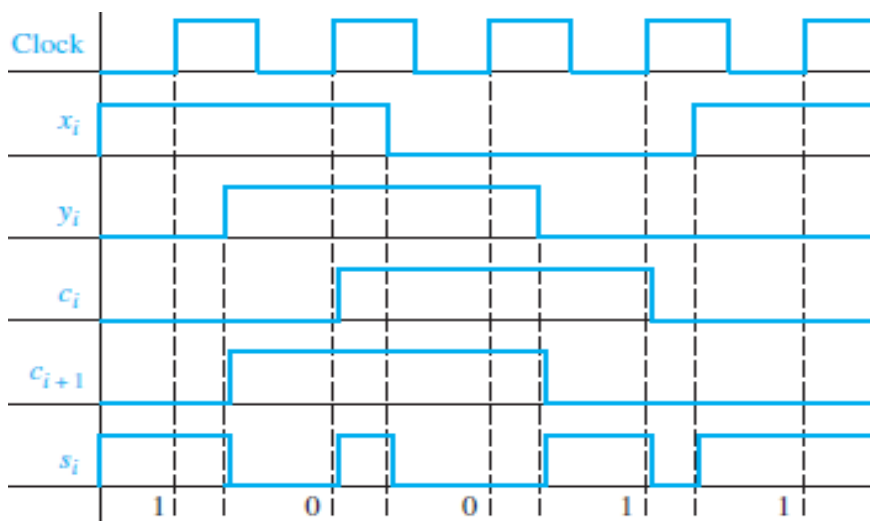
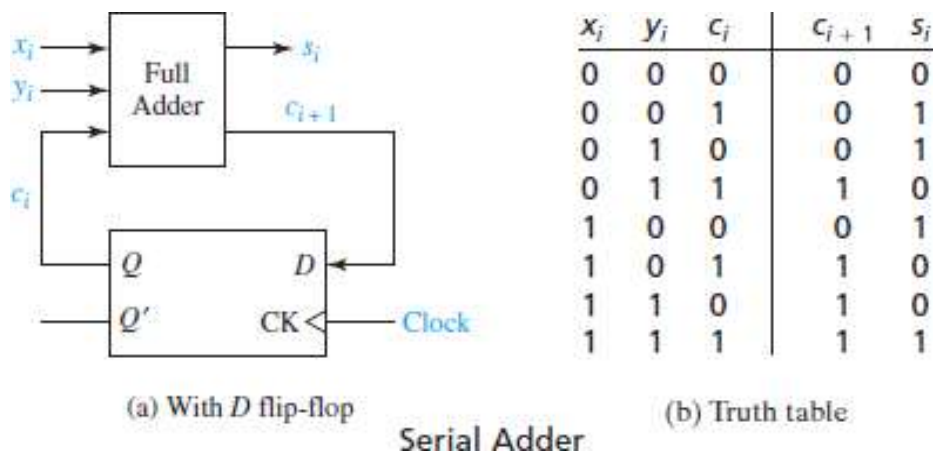


In Table (b), the Present Output column gives the output associated with the present state and present input. Thus, if the present state is  $S_0$  and the input changes from 0 to 1, the output will immediately change from 0 to 1. However, the state will not change to the next state ( $S_1$ ) until after the clock pulse. For the above Figure (Mealy State Graph), the labels on the arrows between states are of the form X/Z, where the symbol before the slash is the input and the symbol after the slash is the corresponding output.

Thus, in state S0 an input of 0 gives an output of 0, and an input of 1 gives an output of 1. For any given input sequence, we can easily trace out the state and output sequences on the state graph. For the input sequence  $X = 10101$ , verify that the corresponding output sequence is 11001. This agrees with timing chart; if the false outputs are ignored. Note that the false outputs do not show on the state graph because the inputs are read at the active clock edge, and no provision is made for extra input changes between active edges.

**Operation of Serial Adder:**

Consider a serial adder [following Figure (a)] that adds two n-bit binary numbers  $X = x_{n-1} \dots x_1x_0$  and  $Y = y_{n-1} \dots y_1y_0$ .



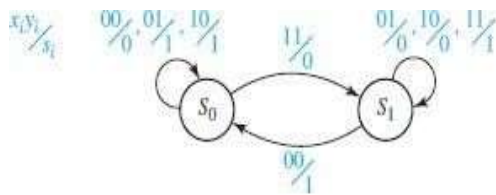
**Timing Diagram for Serial Adder**

The binary numbers are fed in serially, one pair of bits at a time, and the sum is read out serially, one bit at a time. First,  $x_0$  and  $y_0$  are fed in; a sum digit  $s_0$  is generated, and the carry  $c_1$  is stored. At the next clock time,  $x_1$  and  $y_1$  are fed in and added to  $c_1$  to give the next sum digit  $s_1$  and the new carry  $c^2$ , which is

stored. This process continues until all bits have been added. A full adder is used to add the  $x_i$ ,  $y_i$ , and  $c_i$  bits to form  $c_{i+1}$  and  $s_i$ . A D flip-flop is used to store the carry ( $c_{i+1}$ ) on the rising edge of the clock. The  $x_i$  and  $y_i$  inputs must be synchronized with the clock.

The above Figure also shows a timing diagram for the serial adder. In this example, we add 10011 + 00110 to give a sum of 11001 and a final carry of 0. Initially the carry flip-flop must be cleared so that  $c_0 = 0$ . We start by adding the least-significant (rightmost) bits in each word. Adding 1 + 0 + 0 gives  $s_0 = 1$  and  $c_1 = 0$ , which is stored in the flip-flop at the rising clock edge. Since  $y_1$  is 1, adding 1 + 1 + 0 gives  $s_1 = 0$  and  $c_2 = 1$ , which is stored in the flip-flop on the rising clock edge. This process continues until the addition is completed. Reading the sum output just before the rising edge of the clock gives the correct result.

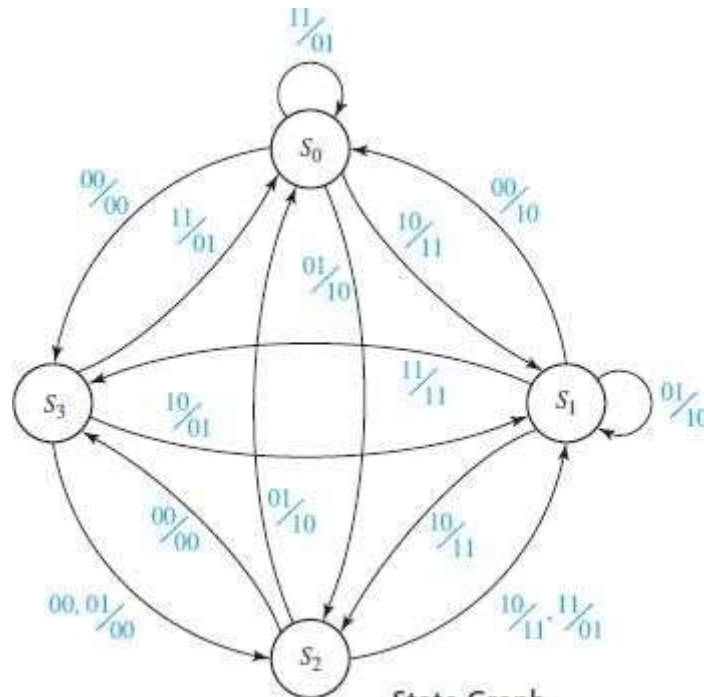
Using the truth table for the full adder (Table given with above Figure), we can construct a state graph (following Figure) for the serial adder.



State Graph for Serial Adder

Present State	Next State				Present Output ( $Z_1Z_2$ )			
	$X_1X_2 = 00$	01	10	11	$X_1X_2 = 00$	01	10	11
$S_0$	$S_3$	$S_2$	$S_1$	$S_0$	00	10	11	01
$S_1$	$S_0$	$S_1$	$S_2$	$S_3$	10	10	11	11
$S_2$	$S_3$	$S_0$	$S_1$	$S_1$	00	10	11	01
$S_3$	$S_2$	$S_2$	$S_1$	$S_0$	00	00	01	01

A State Table with Multiple Inputs and Outputs



State Graph

The serial adder is a Mealy machine with inputs  $x_i$  and  $y_i$  and output  $s_i$ . The two states represent a carry ( $c_i$ ) of 0 and 1, respectively. From the table,  $c_i$  is the present state of the sequential circuit, and  $c_{i+1}$  is the

next state. If we start in  $S_0$  (no carry), and  $x_i y_i = 11$ , the output is  $s_i = 0$  and the next state is  $S_1$ . This is indicated by the arrow going from state  $S_0$  to  $S_1$ .

The above Table is the state table for a Mealy sequential circuit with two inputs and two outputs. The above Figure also shows the corresponding state graph. The notation 00, 01/00 on the arc from  $S_3$  to  $S_2$  means if  $X_1 = X_2 = 0$  or  $X_1 = 0$  and  $X_2 = 1$ , then  $Z_1 = 0$  and  $Z_2 = 0$ .

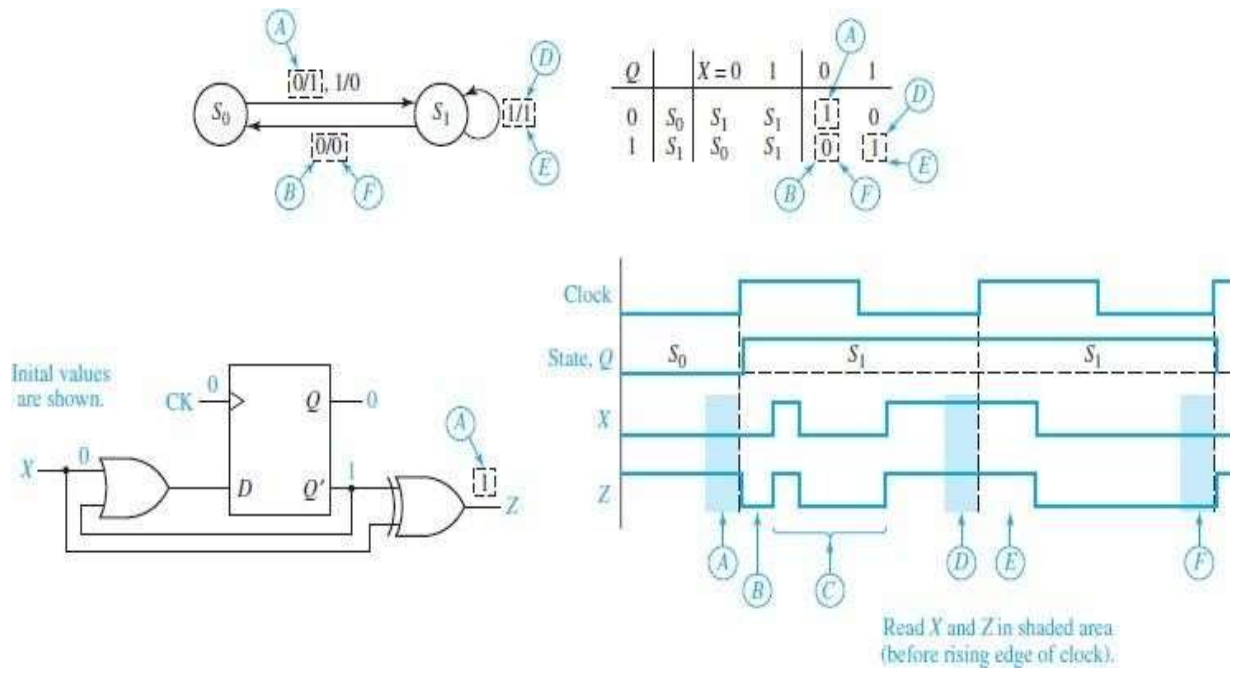
### **Construction and Interpretation of Timing Charts:**

Several important points concerning the construction and interpretation of timing charts are summarized as follows:

1. When constructing timing charts, note that a state change can only occur after the rising (or falling) edge of the clock, depending on the type of flip-flop used.
2. The input will normally be stable immediately before and after the active clock edge.
3. For a Moore circuit, the output can change only when the state changes, but for a Mealy circuit, the output can change when the input changes as well as when the state changes. A false output may occur between the time the state changes and the time the input is changed to its new value. (In other words, if the state has changed to its next value, but the old input is still present, the output may be temporarily incorrect).
4. False outputs are difficult to determine from the state graph, so use either signal tracing through the circuit or use the state table when constructing timing charts for Mealy circuits.
5. When using a Mealy state table for constructing timing charts, the procedure is as follows:
  - a) For the first input, read the present output and plot it.
  - b) Read the next state and plot it (following the active edge of the clock pulse).
  - c) Go to the row in the table which corresponds to the next state and read the output under the old input column and plot it. (This may be a false output).
  - d) Change to the next input and repeat steps (a), (b), and (c). (Note: If you are just trying to read the correct output sequence from the table, step (c) is naturally omitted).
6. For Mealy circuits, the best time to read the output is just before the active edge of the clock, because the output should always be correct at that time.

The example in the following Figure shows a state graph, a state table, a circuit that implements the table, and a timing chart.

When the state is  $S_0$  and the input is  $X = 0$ , the output from the state graph, state table, circuit, and timing chart is  $Z = 1$  (labeled A on the figure). Note that this output occurs before the rising edge of the clock. In a Mealy circuit, the output is a function of the present state and input; therefore, the output should be read just before the clock edge that causes the state to change.



As you continue to study this example, each time the input X changes, trace the changes on the state graph, the state table, the circuit, and the timing chart. Because the input X was 0 before the first rising edge of the clock, the state changes to S<sub>1</sub> after the first rising edge of the clock. Because of the state change, the output also changes (B on the timing chart), but because the input has not yet changed to its new value, the output value may not be correct. We refer to this as a *false output* or *glitch*. If the input changes several times before it assumes its correct value, the output may also change several times (C). The input must assume its correct value before the rising edge of the clock, and the output should be read at this time (D). After the rising clock edge, the state stays the same and the output stays the same for this particular example. In general, the state may change after a rising edge of the clock, and the state change may result in an output change. Again, the output value may be wrong because the input still has the old value (E). When the input is changed to its new value, the output changes to its new value (F), and this value should be read before the next rising clock edge. If we look at the input and output just before each rising edge of the clock, we find the following sequences:

$$X = 0 \ 1 \ 0$$

$$Z = 1 \ 1 \ 0$$

You should be able to verify the sequence for Z using the state graph, using the state table, and using the circuit diagram.

## References

Books:

1. Charles H Roth and Larry L Kinney, Analog and Digital Electronics, Cengage Learning, 2019
2. Donald P Leach, Albert Paul Malvino & Goutam Saha, Digital Principles and Applications, 8th Edition, Tata McGraw Hill, 2015.

## Web Resources

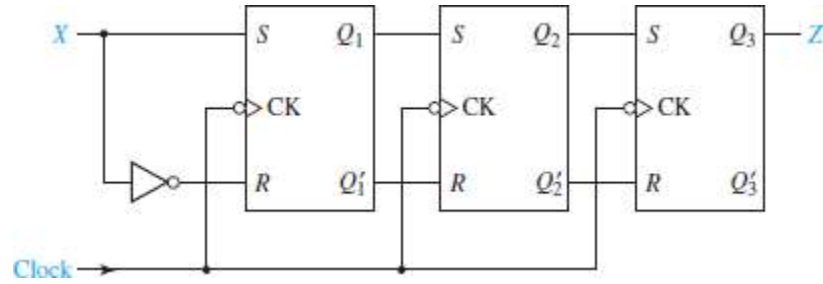
1. [https://www.tutorialspoint.com/computer\\_logical\\_organization/digital\\_registers.htm#:~:text=Flip%2Dflop%20is%20a%201,is%20known%20as%20a%20Register.](https://www.tutorialspoint.com/computer_logical_organization/digital_registers.htm#:~:text=Flip%2Dflop%20is%20a%201,is%20known%20as%20a%20Register.)
2. <https://www.geeksforgeeks.org/shift-registers-in-digital-logic/>
3. <https://learnabout-electronics.org/Digital/dig57.php>
4. [https://www.tutorialspoint.com/computer\\_logical\\_organization/digital\\_counters.htm](https://www.tutorialspoint.com/computer_logical_organization/digital_counters.htm)
5. <https://www.geeksforgeeks.org/counters-in-digital-logic/>
6. <https://www.javatpoint.com/counters-in-digital-electronics>
7. <https://www.coursehero.com/file/p25c4rs/Parallel-Adder-with-Accumulator-Spring-2012-ECE-301-Digital-Electronics-47-In/>
8. [http://www.ee.surrey.ac.uk/Projects/CAL/seqswitching/state\\_diagrams\\_and\\_state\\_tables.htm](http://www.ee.surrey.ac.uk/Projects/CAL/seqswitching/state_diagrams_and_state_tables.htm)

## Video Resources

1. <https://www.youtube.com/watch?v=5IG0IFetGPc>
2. <https://www.youtube.com/watch?v=5BAuSuT6CsE>
3. <https://www.youtube.com/watch?v=mTg3F1Jautk>
4. <https://www.youtube.com/watch?v=O5orrznBEI0>
5. [https://www.youtube.com/watch?v=5\\_jaAI7qlxg](https://www.youtube.com/watch?v=5_jaAI7qlxg)
6. <https://www.youtube.com/watch?v=ppIwIi3NPso&t=221s>
7. <https://www.youtube.com/watch?v=BKFAQSdjey4>
8. <https://www.youtube.com/watch?v=To4WoZsAbCc>
9. [https://www.youtube.com/watch?v=y0sp5Vqf\\_aQ](https://www.youtube.com/watch?v=y0sp5Vqf_aQ)
10. <https://www.youtube.com/watch?v=iJmeYLRadsA>
11. <https://www.youtube.com/watch?v=0nOrsPTcK6Q>
12. <https://www.youtube.com/watch?v=PplA-E7pjFY>
13. <https://www.youtube.com/watch?v=SIV2gnA1mIE>

## Question Bank

1] Construct a state graph for the shift register shown. ( $X$  is the input, and  $Z$  is the output). Is this a Mealy or Moore machine?



2] Below is a state transition table with the outputs missing. The output should be  $Z = X'B' + XB$ .

ABC	$A+B+C$	
	$X = 0$	$X = 1$
000	011	010
001	000	100
010	100	100
011	010	000
100	100	001

(a) Is this a Mealy machine or Moore machine?

(b) Fill in the outputs on the state transition table.

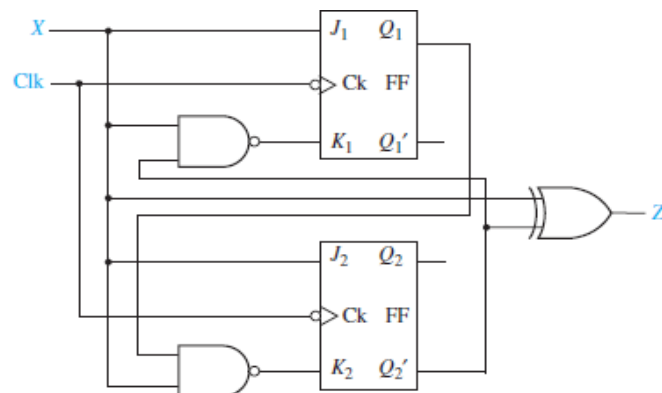
(c) Give the state graph.

(d) For an input sequence of  $X = 10101$ , give a timing diagram for the clock,  $X$ ,  $A$ ,  $B$ ,  $C$ , and  $Z$ . State changes occur on the rising clock edge. What is the correct output sequence for  $Z$ ? Change  $X$  between rising and falling clock edges so that we can see false outputs, and indicate any false outputs on the diagram.

3] (a) Construct a state table and graph for the circuit shown.

(b) Construct a timing chart for the circuit for an input sequence  $X = 10111$ . (Assume that initially  $Q_1 = Q_2 = 0$  and that  $X$  changes midway between the rising and falling clock edges.)

(c) List the output values produced by the input sequence.





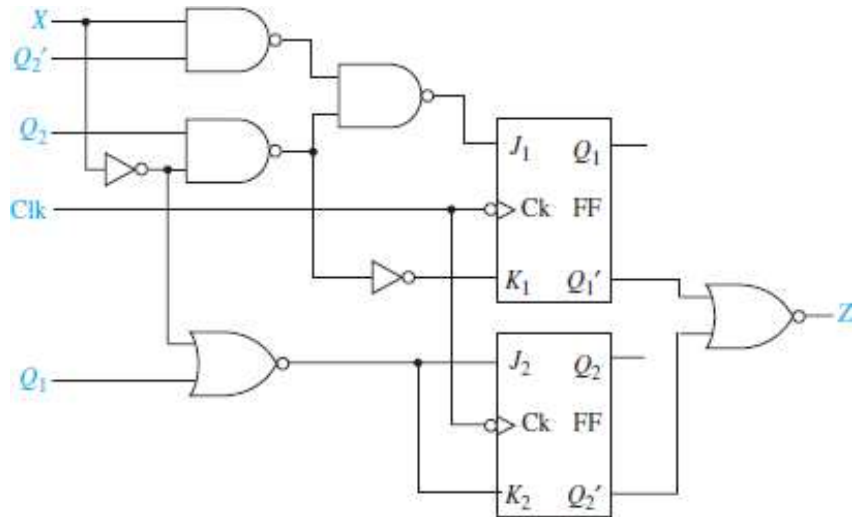
4] Consider the circuit shown.

(a) Construct a state table and graph for the following circuit. Is the circuit a Mealy or Moore circuit?

Does the circuit have any unused states? Assume 00 is the initial state.

(b) Draw a timing diagram for the input sequence  $X = 01100$ .

(c) What is the output sequence for the input sequence?



## University Questions

- 1) a. What is shift register? Explain the working of 8 bit SISO shift register using SR flip flop. (06 Marks)
- b. With the help of state graph, state and transition tables and timing diagram explain\* sequential parity checker. (06 Marks)
- c. Design a random counter using T flip flops whose transition graph is shown in Fig.Q.9(c). (08 Marks)

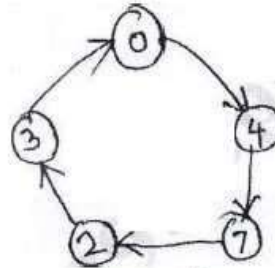


Fig.Q.9(c)

### OR

- 2) a. What is register? Explain how 4 bit register with data, load, clear and clock input is constructed using D flip flops. (06 Marks)
  - b. With a block diagram explain the working of n-bit parallel adder with accumulator. (06 Marks)
  - c. Differentiate between Moore and Melay machines. Analyze following Moore sequential circuit for an input sequence of  $X = 01101$  and draw the timing diagram. (08 Marks)
- 
- 3) a. With neat diagram, explain 4 bit parallel adder with accumulator. (08 Marks)
  - b. With diagram explain 4 bit SISO register. (08 Marks)
  - c. Write a note on Johnson tail counter. (04 Marks)
- 
- 4) a. Design Mod 5 counter using JK flip flops. (10 Marks)
  - b. Explain 4 bit PIPO shift register with block diagram. (06 Marks)
  - c. Write a note on ring counter. (04 Marks)