

## 7.1 Introduction

A group of flip-flops connected together forms a **register**. A register is used solely for storing and shifting data which is in the form of 1s and/or 0s, entered from an external source. It has no specific sequence of states except in certain very specialized applications. A **counter** is a register capable of counting the number of clock pulses arriving at its clock input. Count represents the number of clock pulses arrived. A specified sequence of states appears as the counter output. This is the main difference between a register and a counter. A specified sequence of states is different for different types of counters.

There are two types of counters, synchronous and asynchronous. In synchronous counter, the common clock input is connected to all of the flip-flops and thus they are clocked simultaneously. In asynchronous counter, commonly called, ripple counters, the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the Q or  $\bar{Q}$  output of the previous flip-flop. Therefore in an asynchronous counter, the flip-flops are not clocked simultaneously. Let us start with asynchronous counters.

## 7.2 Asynchronous / Ripple Up Counters

Fig. 7.1 shows 2-bit asynchronous counter using JK flip-flops. As shown in Fig. 7.1, the clock signal is connected to the clock input of only first stage flip-flop. The clock input of the second stage flip-flop is triggered by the  $Q_A$  output of the first stage. Because of the inherent propagation delay time through a flip-flop, a transition of the input clock pulse and a transition of the  $Q_A$  output of first stage can never occur at exactly the same time. Therefore, the two flip-flops are never simultaneously triggered, which results in asynchronous counter operation.

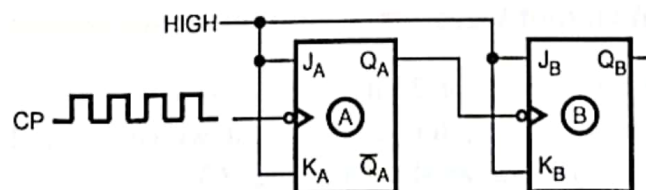


Fig. 7.1 A two-bit asynchronous binary counter

Fig. 7.1 (a) shows the timing diagram for two-bit asynchronous counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock. J and K input of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

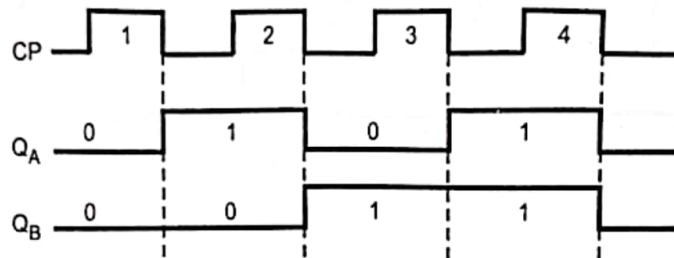


Fig 7.1 (a) Timing diagram for the counter of Fig. 7.1

Example 7.1 : Extend the counter shown in Fig. 7.1 for 3-stages, and draw output waveforms.

Solution :

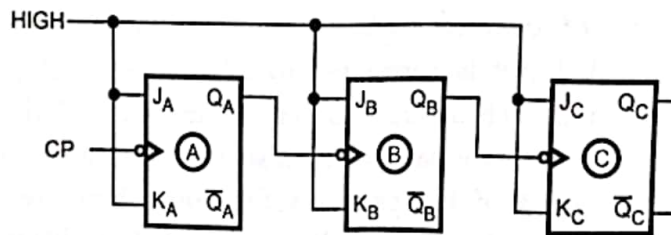


Fig. 7.2 (a) Logic Diagram

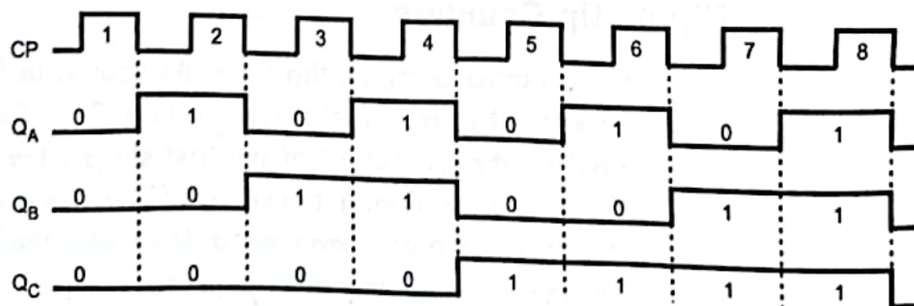


Fig. 7.2 (b) Output waveforms for 3-bit asynchronous counter

In Fig. 7.2 (b), timing diagram for 3-bit asynchronous counter we have not considered the propagation delays of flip-flops, for simplicity. If we consider the propagation delays of flip-flops we get timing diagram as shown in Fig. 7.3.

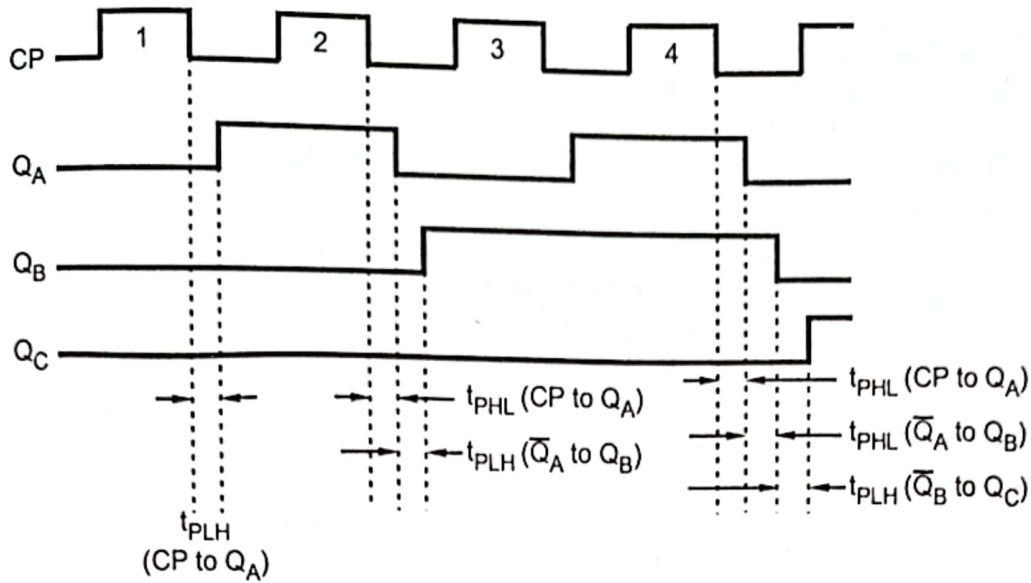


Fig. 7.3 Propagation delays in a ripple clocked binary counter

The timing diagram shows propagation delays. We can see that propagation delay of the first stage is added in the propagation delay of second stage to decide the transition time for third stage. This cumulative delay of an asynchronous counter is a major disadvantage in many applications because it limits the rate at which the counter can be clocked and creates decoding problems.

►►► **Example 7.2 :** Draw the logic diagram for 3-stage asynchronous counter with negative edge triggered flip-flops.

**Solution :** When flip-flops are negatively edge triggered. The Q output of previous stage is connected to the clock input of the next stage. Fig. 7.4 shows 3-stage asynchronous counter with negative edge triggered flip-flops.

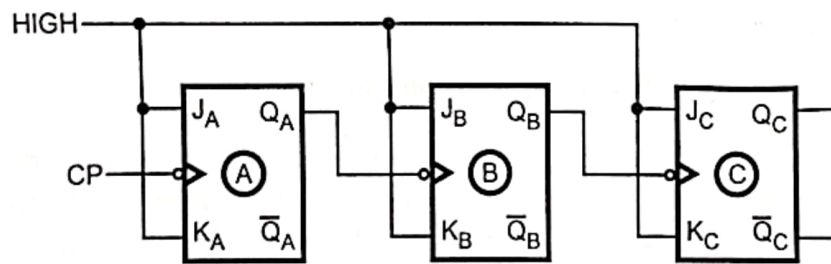


Fig. 7.4 Logic diagram of 3-stage negative edge triggered counter

►►► **Example 7.3 :** A counter has 14 stable states 0000 through 1101. If the input frequency is 50 kHz what will be its output frequency?

**Solution :**

$$\frac{50 \text{ kHz}}{14} = 3.57 \text{ kHz}$$



➔ **Example 7.4 :** The  $t_{pd}$  for each flip-flop is 50 ns , determine the maximum operating frequency for MOD-32 ripple counter.

**Solution :** We know that MOD-32 uses five flip-flops. With  $t_{pd} = 50$  ns, the  $f_{max}$  for ripple counter can be given as,

$$f_{\max(\text{ripple})} = \frac{1}{5 \times 50 \text{ ns}} = 4 \text{ MHz}$$

### 7.3 Asynchronous/Ripple Down Counter

In the last section we have seen that the output of counter is incremented by one for each clock transition. Therefore, we call such counters as up counters. In this section we see the asynchronous/ripple down counter. The down counter will count downward from a maximum count to zero.

The Fig. 7.5 shows the 4-bit asynchronous down counter using JK flip-flops. Here, the clock signal is connected to the clock input of only first flip-flop. This connection is same as asynchronous/ripple up counter. However, the clock input of the remaining flip-flops is triggered by the  $\bar{Q}_A$  output of the previous stage instead of  $Q_A$  output of the previous stage.

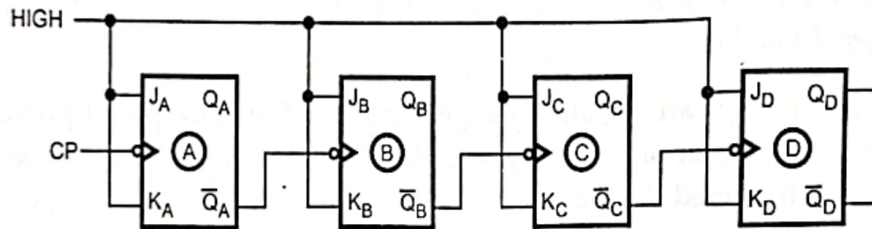


Fig. 7.5 4-bit asynchronous down counter

The Fig. 7.6 shows the timing diagram for 4-bit asynchronous down counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock. Again the J and K inputs of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

Down counters are not as widely used as up counters. They are used in situation where it must be known when a desired number of input pulses has occurred. In these situations the down counter is preset to the desired number and then allowed to count down as the pulses are applied. When the counter reaches the zero state it is detected by a logic gate whose output then indicates that the preset number of pulses has occurred.

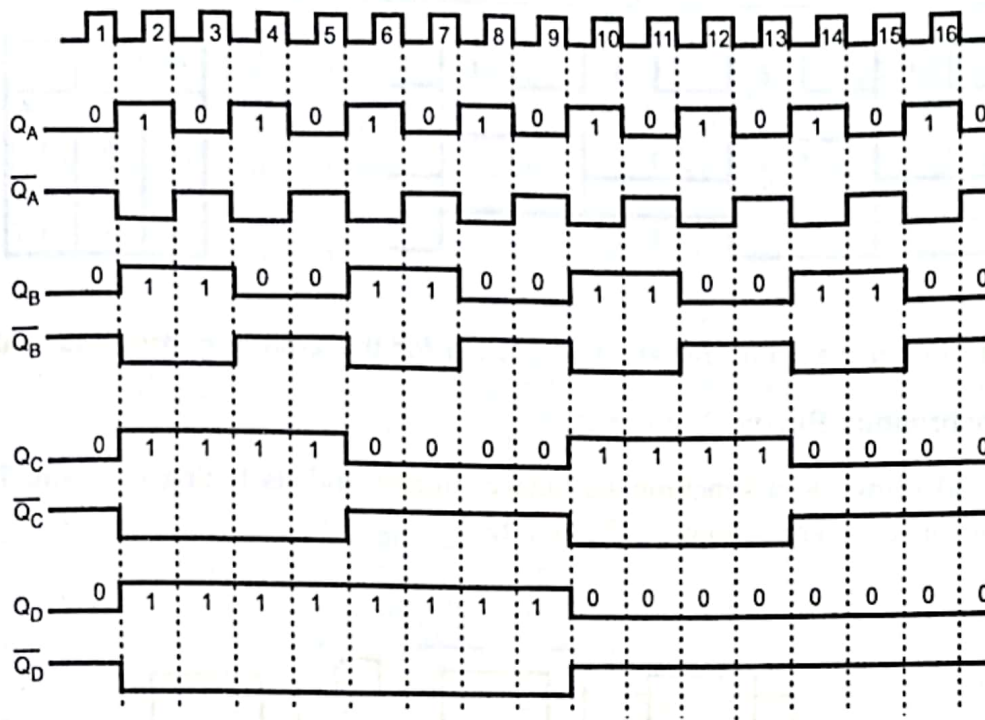


Fig. 7.6 Timing diagram of 4-bit asynchronous down counter

### 7.4 Synchronous Up Counters

When counter is clocked such that each flip-flop in the counter is triggered at the same time, the counter is called as synchronous counter. Fig. 7.7 shows two stage synchronous counter.

Here, clock signal is connected in parallel to clock inputs of both the flip-flops. But the

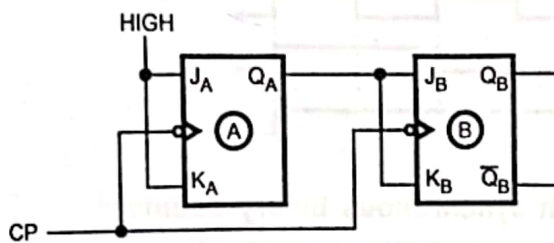


Fig. 7.7 A two-bit synchronous binary counter

$Q_A$  output of first stage is used to drive the J and K inputs of the second stage. Let us see the operation of the circuit. Initially, we assume that the  $Q_A = Q_B = 0$ . When positive edge of the first clock pulse is applied, flip-flop A will toggle because  $J_A = K_A = 1$ , whereas flip-flop B output will remain zero because  $J_B = K_B = 0$ . After first clock pulse  $Q_A = 1$

and  $Q_B = 0$ . At negative going edge of the second clock pulse both flip-flops will toggle because they both have a toggle condition on their J and K inputs ( $J_A = K_A = J_B = K_B = 1$ ). Thus after second clock pulse,  $Q_A = 0$  and  $Q_B = 1$ . At negative going edge of the third clock pulse flip-flop A toggles making  $Q_A = 1$ , but flip-flop B remains set i.e.  $Q_B = 1$ . Finally, at the leading edge of the fourth clock pulse both flip-flops toggle as their JK inputs are at logic 1. This results  $Q_A = Q_B = 0$  and counter recycled back to its original state. The timing details of above operation is shown in Fig. 7.8.

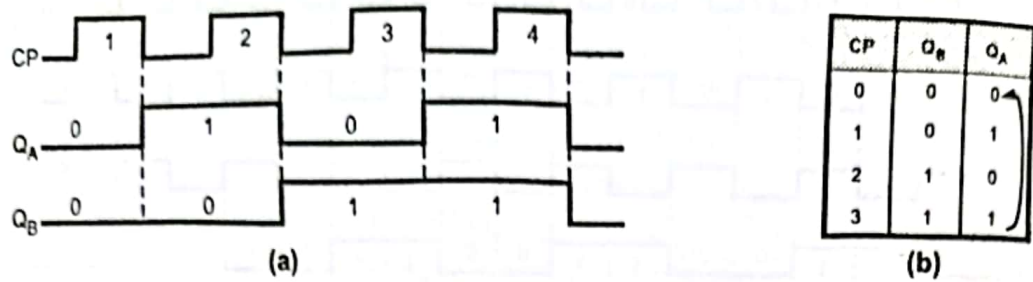


Fig. 7.8 Timing diagram and state sequence for the 2-bit synchronous counter

**A 3-bit Synchronous Binary Counter**

Fig. 7.9 (a) shows 3-bit synchronous binary counter and its timing diagram. The state sequence for this counter is shown in Table 7.1.

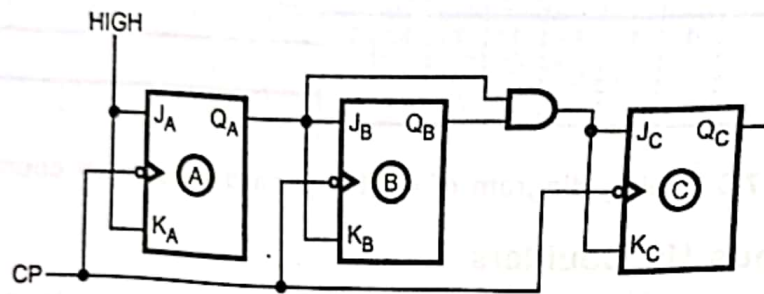


Fig. 7.9 (a) A three-bit synchronous binary counter

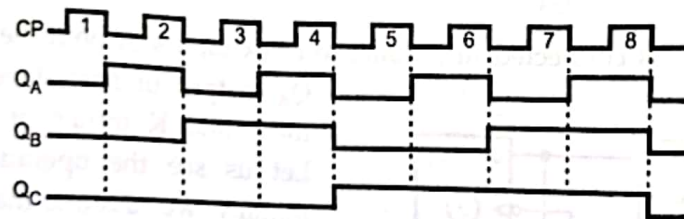


Fig. 7.9 (b) Timing diagram for 3-bit synchronous binary counter

CP	Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>
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

Table 7.1 State sequence for 3-bit binary counter



Looking at Fig. 7.9 (b), we can see that  $Q_A$  changes on each clock pulse as we progress from its original state to its final state and then back to its original state. To produce this operation, flip-flop A is held in the toggle mode by connecting J and K inputs to HIGH. Now let us see what flip-flop B does. Flip-flop B toggles, when  $Q_A$  is 1. When  $Q_A$  is a 0, flip-flop B is in the no-change mode and remains in its present state. Looking at the Table 7.1, we can notice that flip-flop C has to change its state only when  $Q_B$  and  $Q_A$  both are at logic 1. This condition is detected by AND gate and applied to the J and K inputs of flip-flop C. Whenever both  $Q_A$  and  $Q_B$  are HIGH, the output of the AND gate makes the J and K inputs of flip-flop C HIGH, and flip-flop C toggles on the following clock pulse. At all other times, the J and K inputs of flip-flop C are held LOW by the AND gate output, and flip-flop does not change state.

**A Four-Bit Synchronous Binary Counter**

Fig. 7.10 (a) shows logic diagram and timing diagram for 4-bit synchronous binary counter. As counter is implemented with negative edge triggered flip-flops, the transitions occur at the negative edge of the clock pulse. In this circuit, first three flip-flops work same as 3-bit counter discussed previously.

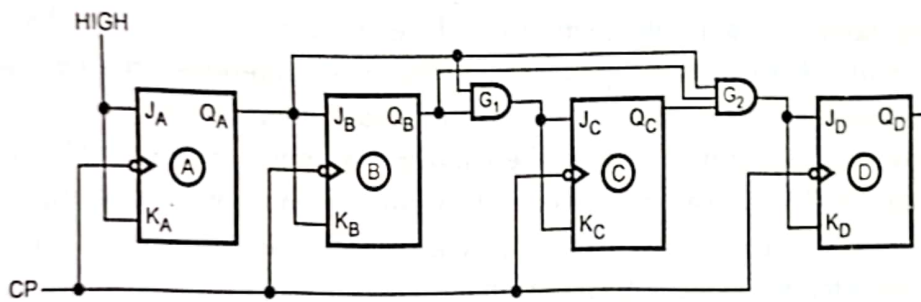


Fig. 7.10 (a)

For the fourth stage, flip-flop has to change the state when  $Q_A = Q_B = Q_C = 1$ . This condition is decoded by 3-input AND gate  $G_2$ . Therefore, when  $Q_A = Q_B = Q_C = 1$ , flip-flop D toggles and for all other times it is in no change condition.

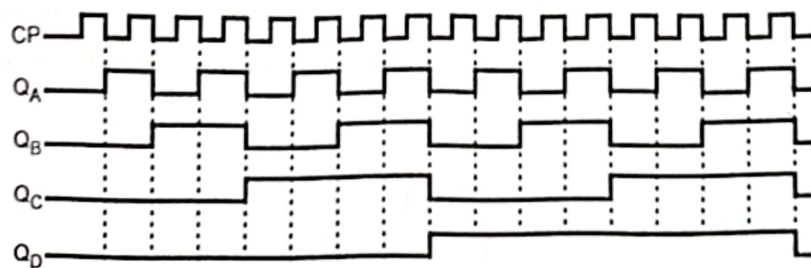


Fig. 7.10 (b) A four-bit synchronous binary counter and timing diagram

►►► **Example 7.5 :** Determine  $f_{max}$  for the 4-bit synchronous counter if  $t_{pd}$  for each flip-flop is 50 ns and  $t_{pd}$  for each AND gate is 20 ns. Compare this with  $f_{max}$  for a MOD - 16 ripple counter.

**Solution :** For a synchronous counter the total delay that must be allowed between input clock pulses is equal to flip-flop  $t_{pd}$  + AND gate  $t_{pd}$ . Thus  $T_{clock} \geq 50 + 20 = 70$  ns, and so the counter has

$$f_{max} = \frac{1}{70 \text{ ns}} = 14.3 \text{ MHz}$$

We know that MOD-16 ripple counter used four flip-flops. With flip-flop  $t_{pd} = 50$  ns, the  $f_{max}$  for ripple counter can be given as

$$f_{max \text{ (ripple)}} = \frac{1}{4 \times 50 \text{ ns}} = 5 \text{ MHz}$$

## 7.5 Synchronous Down and Up/Down Counters

We have seen that a ripple counter could be made to count down by using the inverted output of each flip-flop to drive the next flip-flops in the counter. A parallel/synchronous down counter can be constructed in a similar manner—that is, by using the inverted FF outputs to drive the following JK inputs. For example, the parallel up counter of Fig. 7.10 (a) can be converted to a down counter by connecting the  $\overline{Q}_A$ ,  $\overline{Q}_B$ ,  $\overline{Q}_C$  and  $\overline{Q}_D$  outputs in place of  $Q_A$ ,  $Q_B$ ,  $Q_C$  and  $Q_D$  respectively. The counter will then proceed through the following sequence as input pulses are applied :

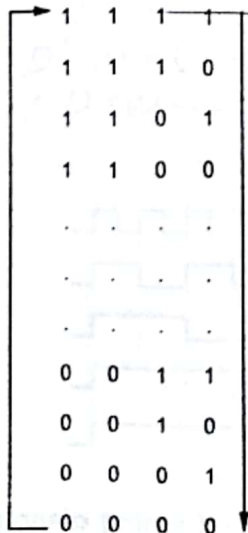


Fig. 7.11

To form a parallel up/down counter the control input (Up/Down) is used to control whether the normal flip-flop outputs or the inverted flip-flop outputs are fed to the J and K inputs of the following flip-flops. The Fig. 7.11 shows 3-bit up/down counter that will count from 000 up to 111 when the Up/Down control input is 1 and from 111 down to 000 when the Up/Down control input is 0.

A logic 1 on the Up/Down enables AND gates 1 and 2 and disables AND gates 3 and 4. This allows the  $Q_A$  and  $Q_B$  outputs through to the J and K inputs of the next flip-flops so that the counter will count up as pulses are applied. When Up/Down line is logic 0, AND gates 1 and 2 are

disabled and AND gates 3 and 4 are enabled. This allows the  $\overline{Q}_A$  and  $\overline{Q}_B$  outputs through to the J and K inputs of the next flip-flops so that the counter will count down as pulses are applied.



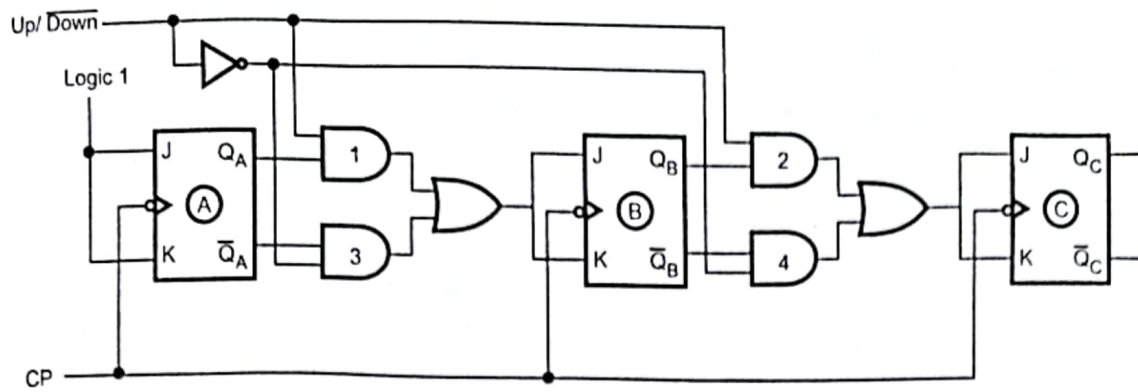


Fig. 7.12 3-bit synchronous/parallel up/down counter

### 7.6 Synchronous Vs Asynchronous Counters

The Table 7.2 shows the comparison between synchronous and asynchronous counters.

Asynchronous Counters	Synchronous Counters
1) In this type of counter flip-flops are connected in such a way that output of first flip-flop drives the clock for the next flip-flop.	1) In this type there is no connection between output of first flip-flop and clock input of the next flip-flop.
2) All the flip-flops are not clocked simultaneously.	2) All the flip-flops are clocked simultaneously.
3) Logic circuit is very simple even for more number of states.	3) Design involves complex logic circuit as number of states increases.
4) Main drawback of these counters is their low speed as the clock is propagated through number of flip-flops before it reaches last flip-flop.	4) As clock is simultaneously given to all flip-flops there is no problem of propagation delay. Hence they are preferred when number of flip-flops increases in the given design.

Table 7.2 Synchronous Vs Asynchronous counters

### 7.7 MOD Counters using Reset Input

Fig. 7.14 shows basic 3-bit ripple counter. In its basic form it is a MOD-8 binary counter which count in sequence from 000 to 111. However, the presence of NAND gate will alter this sequence as follows :

1. The NAND gate output is connected to the asynchronous RESET inputs of each flip-flop. As long as the NAND output is HIGH, it will have no effect on the counter. When it goes LOW, it will reset all the flip-flops so that counter immediately goes to the 000 state.
2. The inputs for the NAND gate are the outputs of the A and C flip-flops, and so the NAND output will go LOW whenever  $Q_A = Q_C = 1$ . This condition will occur when the counter goes from the 100 state to the 101 state (input pulse 5 on waveforms). The LOW at the NAND output will immediately (generally within a few nanoseconds) reset the counter to the 000 state. Once the flip-flops have been reset, the NAND output goes back HIGH, since the  $Q_A = Q_C = 1$  condition no longer exists.

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1

Fig. 7.13

3. The counting sequence is therefore from 000 to 100. Although the counter does go to the 101 state, it remains there for only a few nanoseconds before it recycles to 000. Thus, we can essentially say that the counter counts from 000 to 100 and then recycles to 000. Due to this counter skips 101, 110, and 111 states and it goes through only five different states; thus it is a MOD-5 counter.

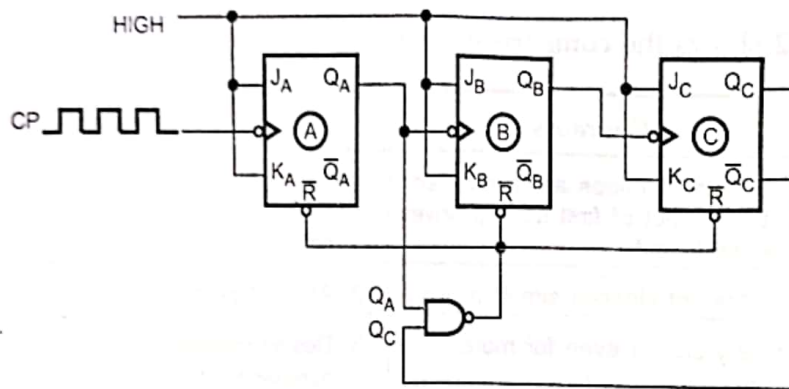


Fig. 7.14 MOD-5 counter using RESET input

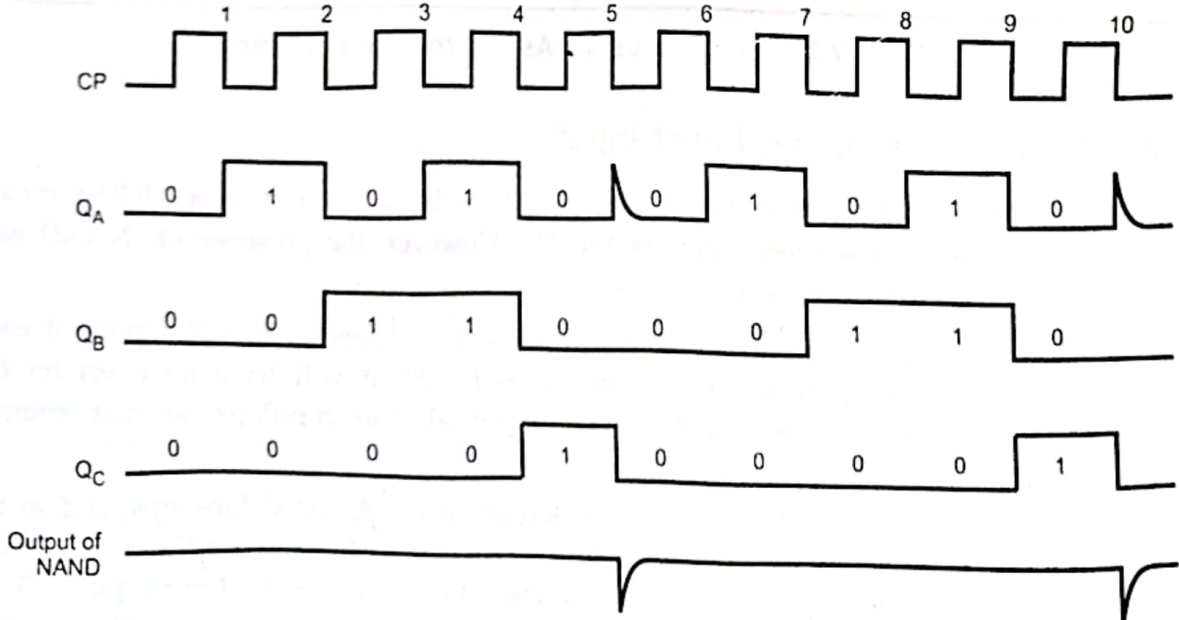


Fig. 7.15 Waveforms for MOD-5 counter

Any desired MOD number can be obtained by changing the NAND gate inputs. The Table 7.3 shows the NAND gate inputs and corresponding MOD-N counter.

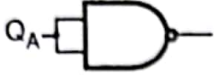
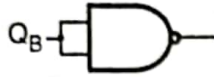
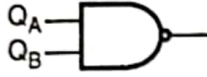

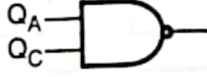
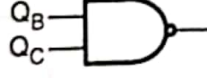

NAND Gate Inputs	Counter
	MOD-1 Counter
	MOD-2 Counter
	MOD-3 Counter
	MOD-4 Counter
	MOD-5 Counter
	MOD-6 Counter
	MOD-7 Counter

Table 7.3 NAND gate inputs for MOD-n counter

➡ **Example 7.6 :** Explain and design asynchronous MOD 10 (decade) counter.

**Solution :** The binary counter has maximum number of states equal to  $2^n$ , where n is the number of flip-flops in the counter. Counters can also be designed to have a number of states in their sequence that is less than  $2^n$ . In decade counters the sequence is truncated upto ten states, 0000 (0 in decimal) through 1001 (9 in decimal). These type of counters are very useful in display applications in which BCD numbers are used.



The truncation in the count sequence is achieved by resetting the counter at particular count instead of going through all of its normal states. In case of BCD decade counter is reset back to the 0000 state after the 1001 state. The resetting of counter is done with the help of reset inputs of each flip-flop. These inputs are activated when desired state is reached. In case of BCD decade counter, reset input is activated using NAND gate when 1010 state is reached. This is illustrated in Fig. 7.16 (a). Fig. 7.16 (b) shows the timing diagram for circuit shown in Fig. 7.16 (a).

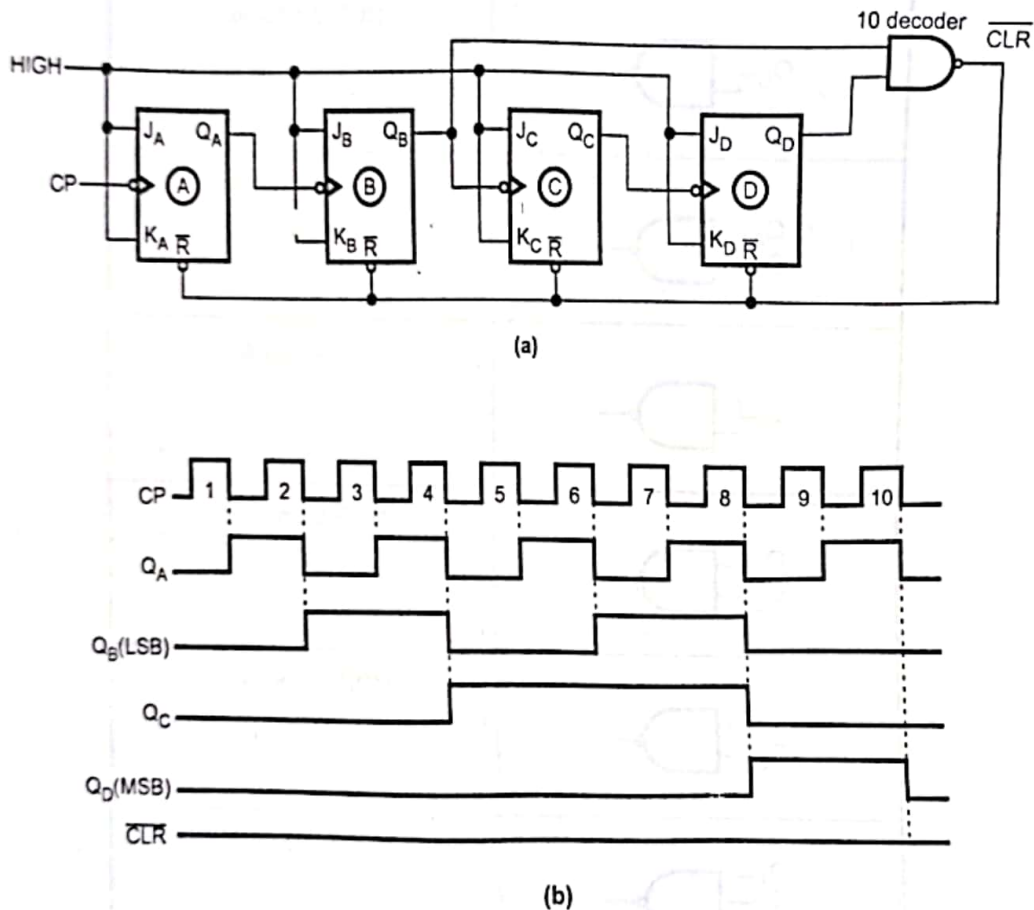


Fig. 7.16 An asynchronously clocked decade counter with asynchronous resetting

## 7.8 Design of Ripple Counters using 7490 and 7493

### 7.8.1 IC 7490 (Decade Binary Counter)

IC 7490 is a decade binary counter. It consists of four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three stage binary counter for which the count cycle length is divide-by-five.

Connection Diagram

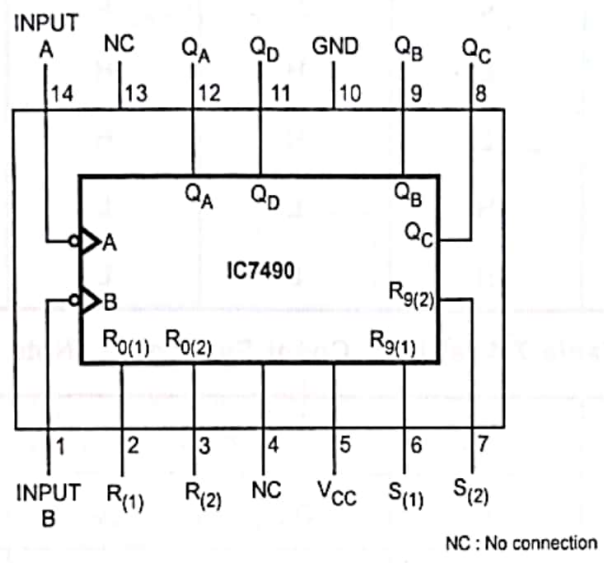


Fig. 7.17 Connection diagram for 7490

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.

1. **BCD Decade (8421) Counter** : The B input must be externally connected to the QA output and A input receives the incoming count and a BCD count sequence is produced.
2. **Symmetrical Bi-quinary Divide-By-Ten Counter** : The QD output must be externally connected to the A input. The input count is then applied to the B input and a divide-by-ten square wave is obtained at output QA.
3. **Divide-By-Two and Divide-By-Five Counter** : No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function (A as the input and QA as the output). The B input is used to obtain binary divide-by-five operation at the QD output.

Table 7.4 shows function tables and Fig. 7.18 shows logic diagram for IC7490.

Count	Outputs			
	Q <sub>D</sub>	Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H

4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

Table 7.4 (a) BCD Count Sequences (Note 1)

Count	Outputs			
	Q <sub>A</sub>	Q <sub>D</sub>	Q <sub>C</sub>	Q <sub>B</sub>
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

Table 7.4 (b) BCD Bi-Quinary (5-2) (Note 2)



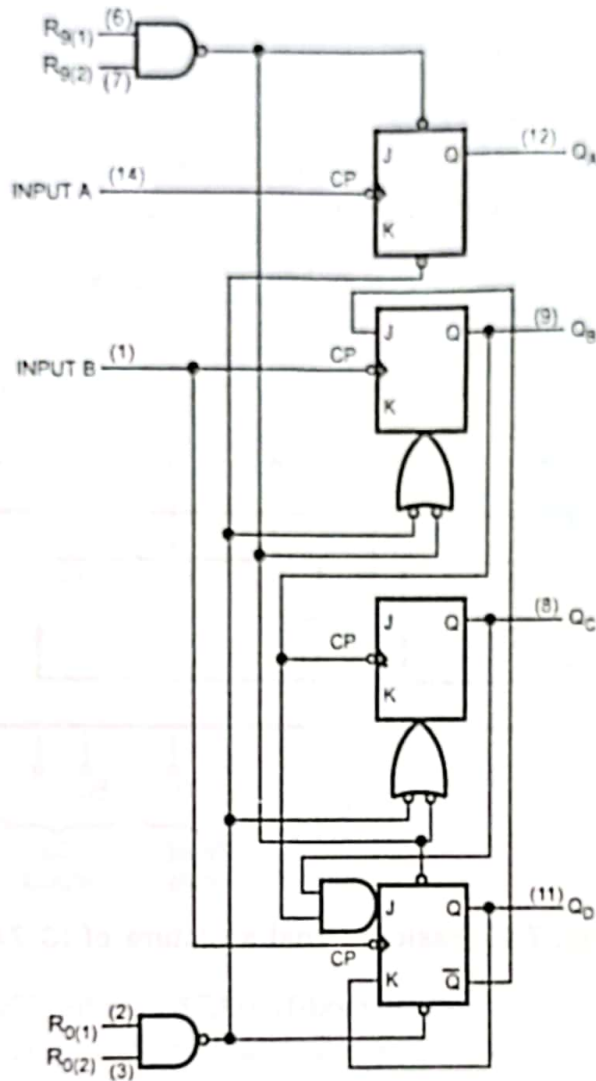


Fig. 7.18 Logic diagram for 7490

Reset Inputs				Outputs			
$R_{0(1)}$	$R_{0(2)}$	$R_{9(1)}$	$R_{9(2)}$	$Q_D$	$Q_C$	$Q_B$	$Q_A$
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

Table 7.4 (c) Reset/Count function table

H : HIGH Level

L : LOW Level

X : Don't Care

Note 1 : Output  $Q_A$  is connected to input B for BCD count.

Note 2 : Output  $Q_D$  is connected to input A for bi-quinary count.

Example 7.7 : Draw basic internal architecture of IC 7490. Design a divide by 20 counter using IC 7490.

Solution : Internal structure of 7490 ripple counter IC is as shown in Fig. 7.19.

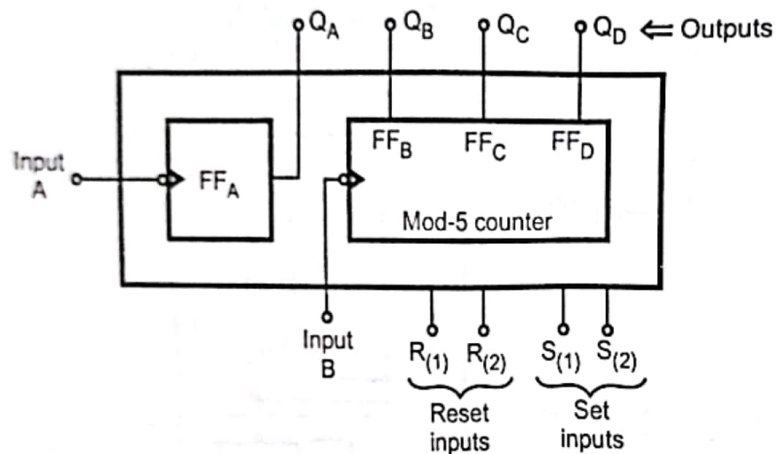


Fig. 7.19 Basic internal structure of IC 7490

We know that, one IC can work as mod-10 (BCD) counter. Therefore we need two ICs. The counter will go through states 0-19 and should be reset on state 20. i.e.

$Q_D$	$Q_C$	$Q_B$	$Q_A$		$Q_D$	$Q_C$	$Q_B$	$Q_A$
0	0	1	0		0	0	0	0
7490 (2)					7490 (1)			

The diagram of divide by 20 counter using IC 7490 is as shown in Fig. 7.20.

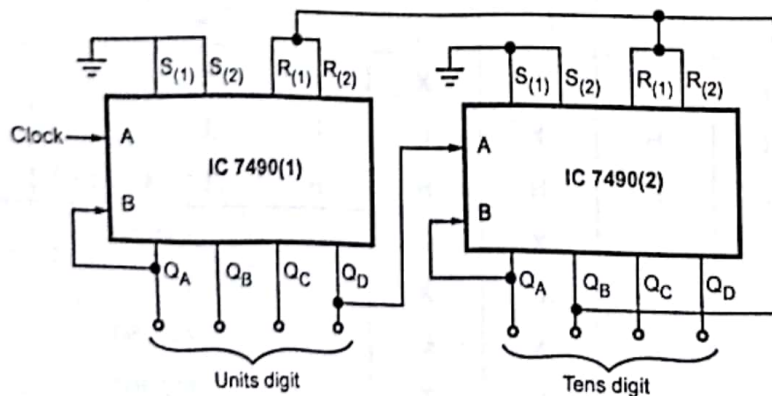


Fig. 7.20 Divide by 20 counter using IC 7490

➔ **Example 7.8 :** Design a divide-by-96 counter using 7490 ICs.

**Solution :** IC 7490 is a decade counter. When two such ICs are cascaded, it becomes a divide by 100 counter. To get a divide-by-96 counter, the counter is reset as soon as it becomes 1001 0110. The diagram is shown in Fig. 7.21.

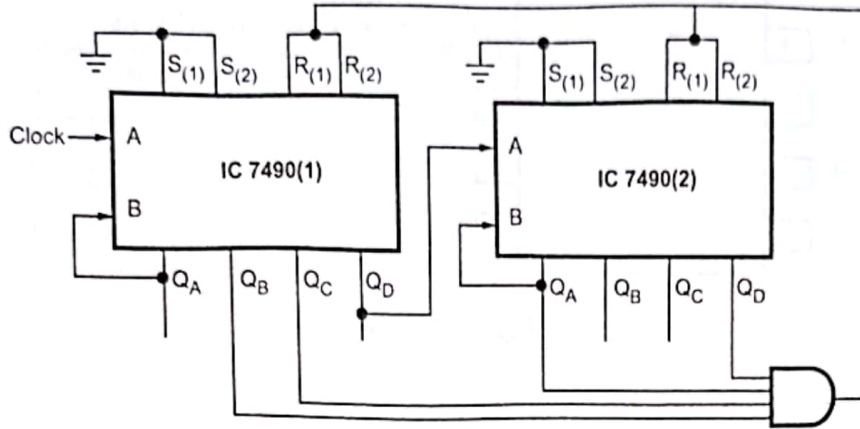


Fig. 7.21 Divide-by-96 counter

### 7.8.2 IC 7492/93 (4-bit Ripple Counters)

The 7492 and 7493 are high speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-six (7492) or divide-by-eight (7493) section which are triggered by a HIGH to LOW transition on the clock inputs. Each section can be used separately or tied together to form divide-by-twelve or divide-by-sixteen counters.

The 7492 is 4-bit divide by twelve counter and 7493 is 4-bit binary counter. Each device consists of four master slave flip-flops which are internally connected to provide a divide-by-two section. Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.

#### Connection Diagram

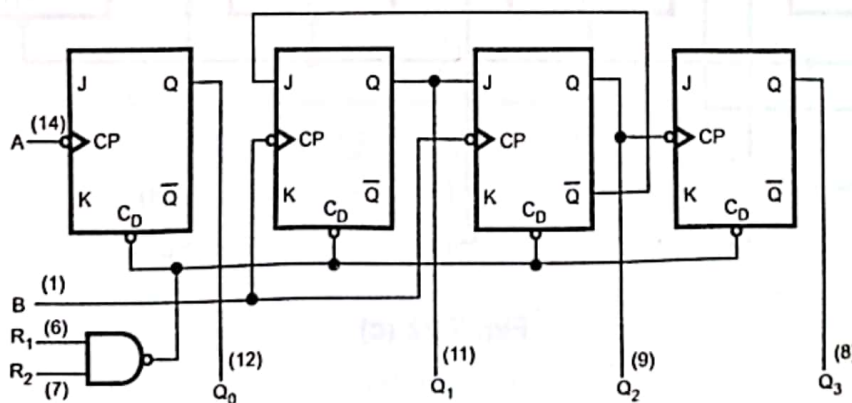


Fig. 7.22 (a)



Pin Diagram

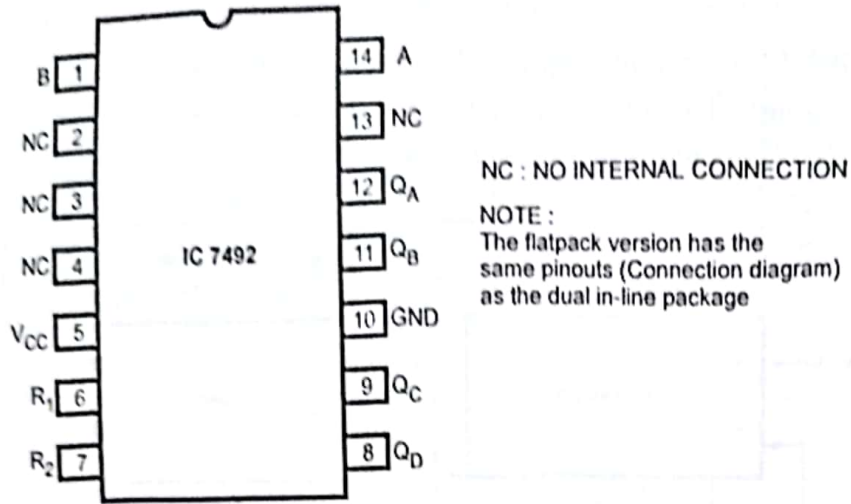


Fig. 7.22 (b)

Modes of 7492

1. Modulo 12, Divide-By-Twelve Counter : The B input must be externally connected to the  $Q_A$  output. The A input receives the incoming count and  $Q_D$  produces a symmetrical divide-by-twelve square wave output.

2. Divide-By-Two and Divide-By-Six Counter : No external inter-connections are required. The first flip-flop is used as a binary element for the divide-by-two function. The B input is used to obtain the divide-by-three operation at the  $Q_B$  and  $Q_C$  outputs and divide by six operation at the  $Q_D$  output.

Connection Diagram

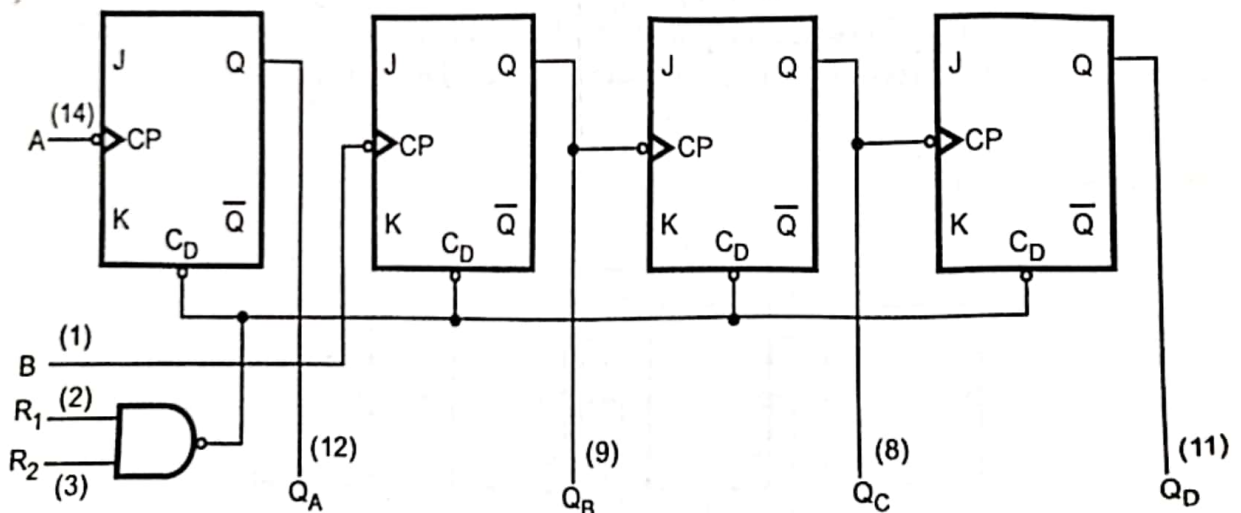


Fig. 7.22 (c)

Pin Diagram

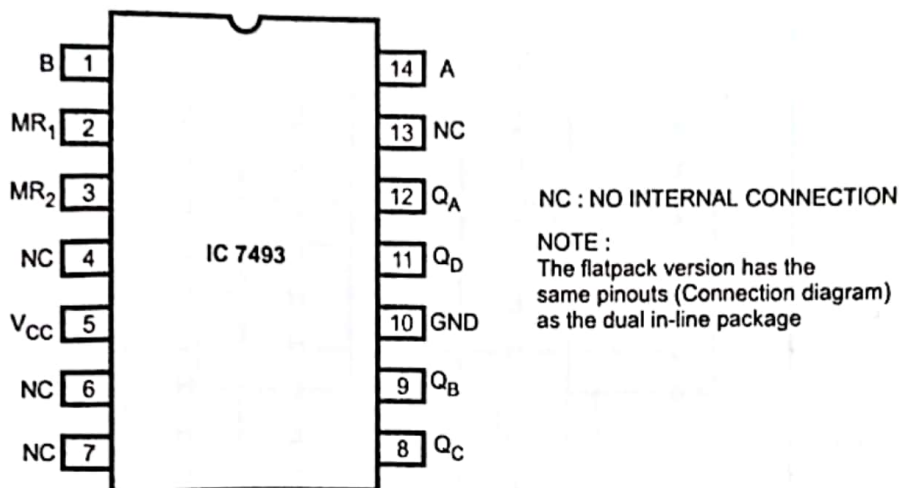


Fig. 7.22 (d)

Modes of 7493

1. 4-Bit Ripple Counter : The output  $Q_A$  must be externally connected to input B. The input count pulses are applied to the input A. Simultaneously divisions of 2, 4, 8 and 16 are performed at the  $Q_A$ ,  $Q_B$ ,  $Q_C$  and  $Q_D$  outputs as shown in the truth table.

2. 3-Bit Ripple Counter : The input count pulses are applied to input A. Simultaneous frequency divisions of 2, 4 and 8 are available at the  $Q_B$ ,  $Q_C$  and  $Q_D$  outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple through counter.

Table 7.5 shows the truth tables for 7492 and 7493.

7492 and 7493

Reset Inputs		Outputs			
$R_1$	$R_2$	$Q_A$	$Q_B$	$Q_C$	$Q_D$
H	H	L	L	L	L
L	H	Count			
H	L	Count			
L	L	Count			

MODE SELECTION

H : High Voltage Level

L : Low Voltage Level

X : Don't Care

Table 7.5 (a)

7492

Count	Output			
	Q <sub>A</sub>	Q <sub>B</sub>	Q <sub>C</sub>	Q <sub>D</sub>
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	L	L	H
7	H	L	L	H
8	L	H	L	H
9	H	H	L	H
10	L	L	H	H
11	H	L	H	H

Note : Output Q<sub>A</sub> is connected to input B.

Table 7.5 (b) Truth table

7493

Count	Output			
	Q <sub>A</sub>	Q <sub>B</sub>	Q <sub>C</sub>	Q <sub>D</sub>
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

Note : Output Q<sub>A</sub> is connected to input B.

Table 7.5 (c) Truth table



➔ **Example 7.9 :** Draw basic internal architecture of IC 7492 and design divide by 9 counter using IC 7492.

**Solution :** Internal structure of 7492 is as shown in Fig. 7.23.

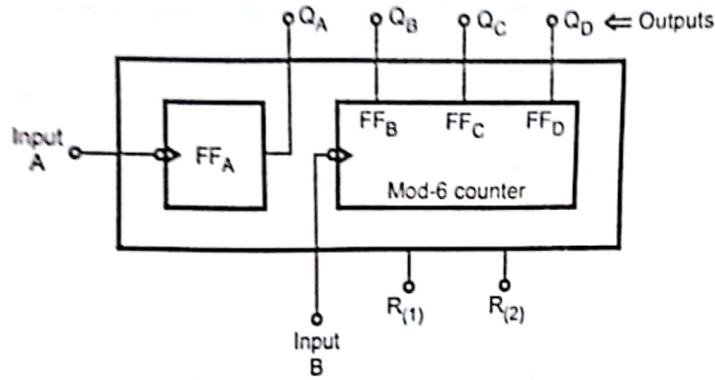


Fig. 7.23

The circuit diagram for divide-by-9 counter is as shown in Fig. 7.24.

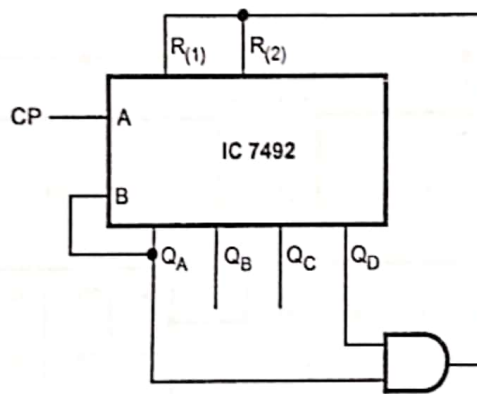


Fig. 7.24 Divide-by-9 counter

➔ **Example 7.10 :** Design a divide-by-128 counter using 7493 ICs.

**Solution :** Since  $128 = 16 \times 8$ , a divide-by-16 counter followed by a divide-by-8 counter will become a divide-by-128 counter. IC 7493 is a 4-bit binary counter (i.e. mod-16 or divide-by-16), therefore, two IC packages will be required.

The circuit diagram is as shown in the Fig. 7.25.

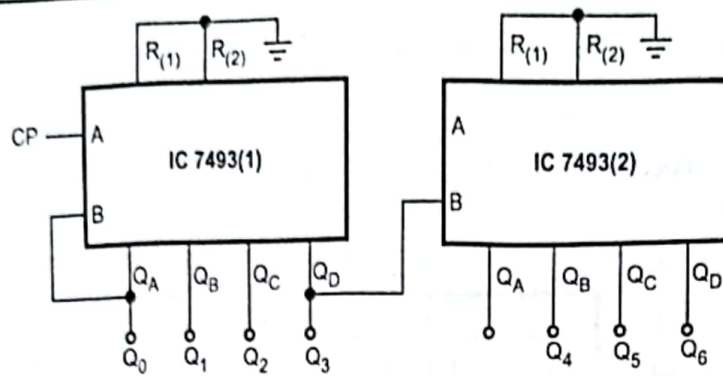


Fig. 7.25 Divide-by-128 counter

➡ **Example 7.11 :** Design a divide-by-78 counter using 7493 and 7492 (as divide-by-6) counter ICs.

**Solution :** Since  $78 = 13 \times 6$ , we have to use 7493 as mod-13 and 7492 as mod-6 counters. For the mod-13 counter  $Q_D$ ,  $Q_C$  and  $Q_A$  outputs of 7493 are ANDed and used to clear the count when the count reaches 1101. For the mod-6 counter, clock is applied to B input of 7492.

The circuit diagram is as shown in the Fig. 7.26.

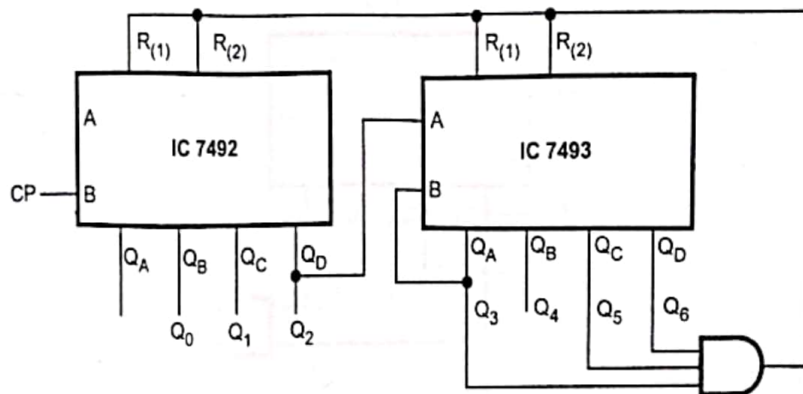


Fig. 7.26 Divide-by-78 counter

➡ **Example 7.12 :** Design Modulo 120 counter using ICs 7490 and 7492.

**Solution :** The Modulo 120 counter can be designed by cascading modulo 12 and modulo 10 counters as shown in the Fig. 7.27.

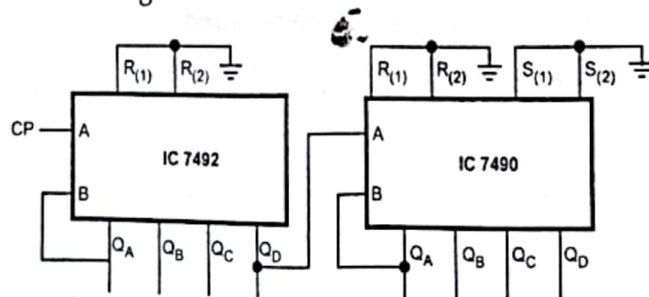


Fig. 7.27 Modulo 120 counter