

## Lab # 6: Study of Various Flip-Flop Circuits

### Objective:

To construct and study the operations of the following circuits:

- (i) RS and Clocked RS Flip-Flop
- (ii) D Flip-Flop
- (iii) JK and Master-Slave JK Flip-Flop
- (iv) T Flip-Flop

### Overview:

So far you have encountered with *combinatorial logic*, i.e. circuits for which the output depends only on the inputs. In many instances it is desirable to have the next output depending on the current output. A simple example is a *counter*, where the next number to be output is determined by the current number stored. Circuits that remember their current output or state are often called *sequential logic* circuits. Clearly, sequential logic requires the ability to store the current state. In other words, *memory* is required by sequential logic circuits, which can be created with boolean gates. If you arrange the gates correctly, they will remember an input value. This simple concept is the basis of RAM (random access memory) in computers, and also makes it possible to create a wide variety of other useful circuits.

Memory relies on a concept called **feedback**. That is, the output of a gate is fed back into the input. The simplest possible feedback circuit using two inverters is shown below (Fig.1):



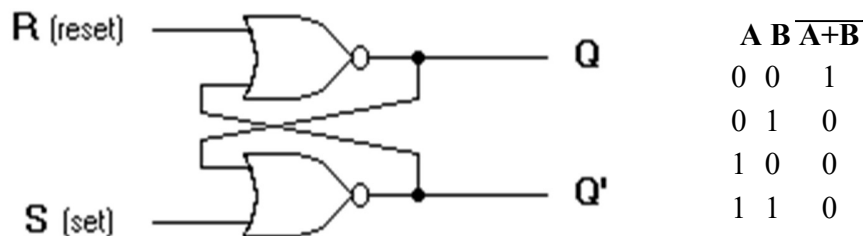
**Fig.1: Simplest realization of feedback circuit**

If you follow the feedback path, you can see that if Q happens to be 1 (or 0), it will always be 1 (or 0). Since it's nice to be able to control the circuits we create, this one doesn't have much use -- but it does let you see how feedback works. It turns out that in "real" sequential circuits, you can actually use this sort of simple inverter feedback approach. The memory elements in these circuits are called *flip-flops*. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Binary information can enter a flip-flop in a variety of ways and gives rise to different types of flip-flops.

## RS Flip-Flop

RS flip-flop is the simplest possible memory element. It can be constructed from two NAND gates or two NOR gates. Let us understand the operation of the RS flip-flop using NOR gates as shown below using the truth table for 'A NOR B' gate. The inputs R and S are referred to as the Reset and Set inputs, respectively. The outputs Q and Q' are complements of each other and are referred to as the normal and complement outputs, respectively. The binary state of the flip-flop is taken to be the value of the normal output. When  $Q=1$  and  $Q'=0$ , it is in the *set state* (or 1-state). When  $Q=0$  and  $Q'=1$ , it is in the *reset/clear state* (or 0-state).

### Circuit Diagram:



- **S=1 and R=0:** The output of the bottom NOR gate is equal to zero,  $Q'=0$ . Hence both inputs to the top NOR gate are equal to 0, thus,  $Q=1$ . Hence, the input combination  $S=1$  and  $R=0$  leads to the flip-flop being **set** to  $Q=1$ .
- **S=0 and R=1:** Similar to the arguments above, the outputs become  $Q=0$  and  $Q'=1$ . We say that the flip-flop is **reset**.
- **S=0 and R=0:** Assume the flip-flop was previously in set ( $S=1$  and  $R=0$ ) condition. Now changing S to 0 results  $Q'$  still at 0 and  $Q=1$ . Similarly, when the flip-flop was previously in a reset state ( $S=0$  and  $R=1$ ), the outputs do not change. Therefore, with inputs  $S=0$  and  $R=0$ , the flip-flop holds its state.
- **S=1 and R=1:** This condition violates the fact that both outputs are complements of each other since each of them tries to go to 0, which is not a stable configuration. It is impossible to predict which output will go to 1 and which will stay at 0. In normal operation this condition must be avoided by making sure that 1's are not applied to both inputs simultaneously, thus making it one of the main disadvantages of RS flip-flop.

All the above conditions are summarized in the characteristic table below:

**Characteristic Table:**

R	S	Q	Q'	Comment
0	0	Q	Q'	Hold state
0	1	1	0	Set
1	0	0	1	Reset
1	1	?	?	Indeterminate

***Debounce circuit***

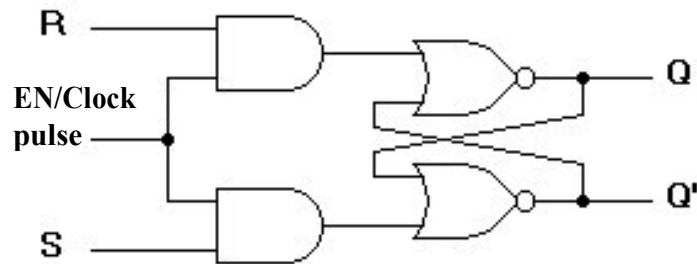
An elementary example using this flip-flop is the debounce circuit. Suppose a piece of electronics is to change state under the action of a mechanical switch. When this switch is moved from position S to R ( $S=0, R=1$ ), the contacts make and break several times at R before settling to good contact. It is desirable that the electronics should respond to the first contact and then remain stable, rather than switching back and forth as the circuit makes and breaks. This is achieved by RS flip-flop which is reset to  $Q=0$  by the first signal  $R=1$  and remains in a fixed state until the switch is moved back to position S, when the signal  $S=1$  sets the flip-flop to  $Q=1$ .

***Gated or Clocked RS Flip-Flop***

It is sometimes desirable in sequential logic circuits to have a bistable RS flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs. By connecting a 2-input AND gate in series with each input terminal of the RS NOR Flip-flop a Gated RS Flip-flop can be created. This extra conditional input is called an "Enable" input and is given the prefix of "EN" as shown below. When the Enable input "EN" = 0, the outputs of the two AND gates are also at logic level 0, (AND Gate principles) regardless of the condition of the two inputs S and R, latching the two outputs Q and Q' into their last known state. When the enable input "EN" = 1, the circuit responds as a normal RS bistable flip-flop with the two AND gates becoming transparent to the Set and Reset signals. This Enable input can also be connected to a clock timing signal adding clock synchronisation to the flip-flop creating what is sometimes called a "Clocked SR Flip-flop".

So a **Gated/Clocked RS Flip-flop** operates as a standard bistable latch but the outputs are only activated when a logic "1" is applied to its EN input and deactivated by a logic "0". The property of this flip-flop is summarized in its characteristic table where  $Q_n$  is the logic state of the previous output and  $Q_{n+1}$  is that of the next output and the clock input being at logic 1 for all the R and S input combinations.

### Circuit Diagram:



### Characteristic Table:

$Q_n$	R	S	$Q_{n+1}$
0	0	0	0 (Hold)
0	1	0	0
0	0	1	1
0	1	1	Indeterminate
1	0	0	1 (Hold)
1	1	0	0
1	0	1	1
1	1	1	Indeterminate

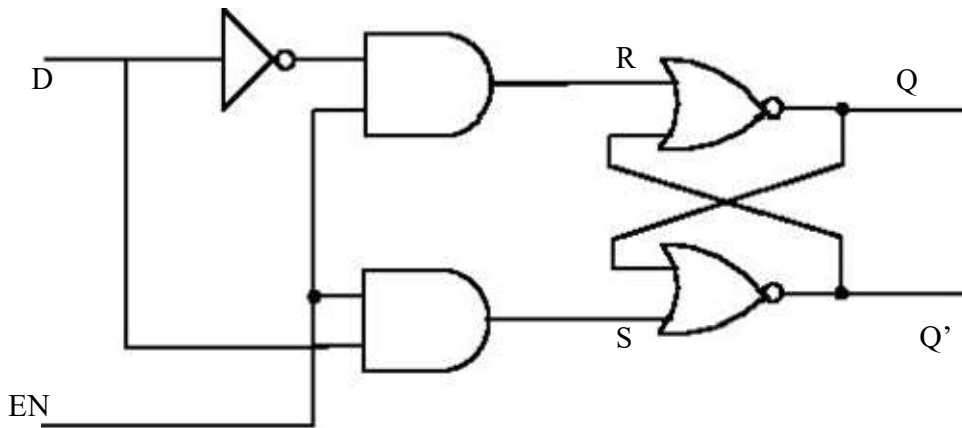
### D FLIP-FLOP

An RS flip-flop is rarely used in actual sequential logic because of its undefined outputs for inputs  $R = S = 1$ . It can be modified to form a more useful circuit called D flip-flop, where D stands for data. The D flip-flop has only a single data input D as shown in the circuit diagram. That data input is connected to the S input of an RS flip-flop, while the inverse of D is connected to the R input. To allow the flip-flop to be in a holding state, a D-flip flop has a second input called Enable, EN. The Enable-input is AND-ed with the D-input.

- When  $EN=0$ , irrespective of D-input, the  $R = S = 0$  and the **state is held**.
- When  $EN=1$ , the S input of the RS flip-flop equals the D input and R is the inverse of D. Hence, output **Q follows D**, when  $EN=1$ .
- When **EN returns to 0**, the most recent input **D is 'remembered'**.

The circuit operation is summarized in the characteristic table for  $EN=1$ .

**Circuit Diagram:**



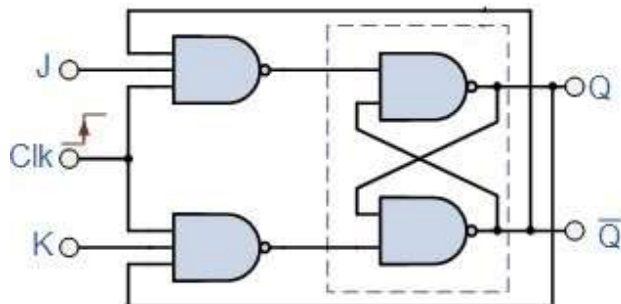
**Characteristic Table:**

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

**JK FLIP-FLOP:**

The JK flip flop (JK means Jack Kilby, a Texas instrument engineer, who invented it) is the most versatile flip-flop, and the most commonly used flip flop. Like the RS flip-flop, it has two data inputs, J and K, and an EN/clock pulse input (CP). Note that in the following circuit diagram NAND gates are used instead of NOR gates. It has no undefined states, however. The fundamental difference of this device is the feedback paths to the AND gates of the input, i.e. Q is AND-ed with K and CP and Q' with J and CP.

**Circuit Diagram:**



**Characteristic Table:**

$Q_n$	J	K	$Q_{n+1}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1(Toggle, $\bar{Q}_n$ )
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0(Toggle, $\bar{Q}_n$ )

The JK flip-flop has the following characteristics:

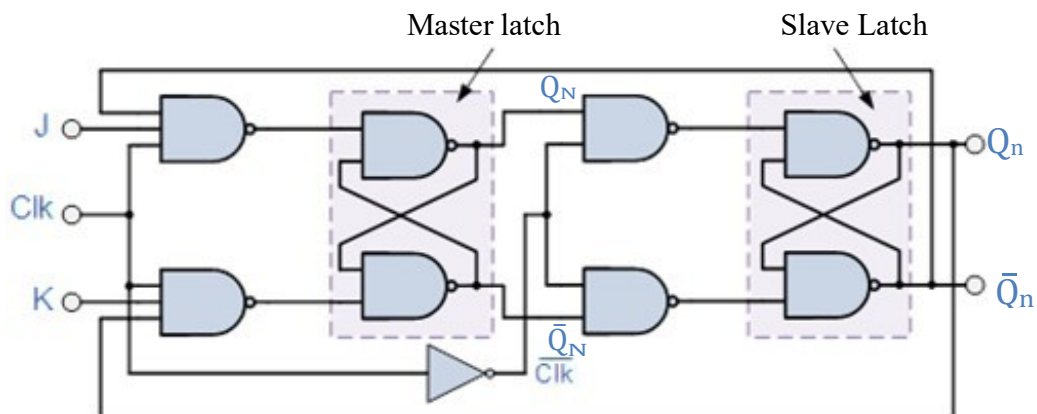
- If one input (J or K) is at logic 0, and the other is at logic 1, then the output is set or reset (by J and K respectively), just like the RS flip-flop.
- If both inputs are 0, then it remains in the same state as it was before the clock pulse occurred; again like the RS flip flop. CP has no effect on the output.
- If both inputs are high, however the flip-flop changes state whenever a clock pulse occurs; i.e., the clock pulse toggles the flip-flop again and again until the CP goes back to 0 as shown in the shaded rows of the characteristic table above. Since this condition is undesirable, it should be eliminated by an improvised form of this flip-flop as discussed in the next section.

### MASTER-SLAVE JK FLIP-FLOP:

Although JK flip-flop is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF", so the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved Master-Slave J-K Flip-Flop was developed. This eliminates all the timing problems by using two SR flip-flops connected together in series, one for the "Master" circuit, which triggers on the leading edge of the clock pulse and the other, the "Slave" circuit, which triggers on the falling edge of the clock pulse.

The master-slave JK flip flop consists of two flip flops arranged so that when the clock pulse enables the first, or master, it disables the second, or slave. When the clock changes state again (i.e., on its falling edge) the output of the master latch is transferred to the slave latch. Again, toggling is accomplished by the connection of the output with the input AND gates.

### Circuit Diagram:



**Characteristic Table:**

CP	J	K	$Q_m$	$\bar{Q}_m$	$Q_n$	$\bar{Q}_n$
0→1	0	0	Hold		Hold	
1→0	0	0	Hold		Hold	
0→1	0	1	0	1	Hold	
1→0	0	1	Hold		0	1
0→1	1	0	1	0	Hold	
1→0	1	0	Hold		1	0
0→1	1	1	Toggle		Hold	
1→0	1	1	Hold		Toggle	

**T FLIP-FLOP:**

The T flip-flop is a single input version of the JK flip-flop. The T flip-flop is obtained from the JK type if both inputs are tied together.

**Circuit Diagram:**

Same as Master-Slave JK flip-flop with J=K=1

- The toggle, or T, flip-flop is a bistable device, where the output of the T flip-flop "toggles" with each clock pulse.
- Till CP=0, the output is in hold state (three input AND gate principle).
- When CP=1, for T=0, previous output is memorized by the circuit. When T=1 along with the clock pulse, the output toggles from the previous value as given in the characteristic table below.

**Characteristic Table:**

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

**Circuit components/Equipments:**

1. Resistors (1K $\Omega$ , 5 Nos)
2. ICs [NOR-7402, AND(2-input)-7408, NAND(3-input)-7410, NAND-7400, NOT-7404]
3. A Surface mount dip switch
4. D.C. Power supply (5V)
5. Red/Green LEDs (4 Nos)
6. Connecting wires
7. Breadboard

**Circuit Diagrams:**

Already provided with text.

**Procedure:**

1. Assemble the circuits one after another on your breadboard as per the circuit diagrams. Circuit diagrams given here do not show connections to power supply and LEDs assuming that you are already familiar with it from your previous lab experience.
2. Connect the ICs properly to power supply (pin 14) and ground (pin 7) following the schematics for ICs given above.
3. Using dip switch and resistors, facilitate all possible combinations of inputs from the power supply. Use the switch also to facilitate pulse input to the circuit.
4. Turn on power to your experimental circuit.
5. For each input combination, note the logic state of the normal and complementary outputs as indicated by the LEDs (ON = 1; OFF = 0), and record the results in a table.
6. Compare your results with the characteristic tables.
7. When you are done, turn off the power to your experimental circuit.

**Observations:**

Table For RS FF: \_\_\_\_\_

Table For Gated RS FF: \_\_\_\_\_

Table For D FF: \_\_\_\_\_

Table For JK FF: \_\_\_\_\_

Table For Master-Slave JK FF: \_\_\_\_\_

Table For T FF: \_\_\_\_\_

**Discussions:****Precautions:**

1. Watch out for loose connections.
  2. While changing the input condition keep the dip switch well pressed.
-



## **Lab # 7: Study of Counter Circuits**

**Objectives:** To construct and study the operations of the following circuits:

- (i) A 4-bit binary ripple Up-counter
- (ii) A 4-bit binary ripple Down-counter
- (iii) A Mod-12 counter
- (iv) A Ring counter

### **Overview:**

Binary Counters are one of the applications of sequential logic using flip-flops. A counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, in form of a clock pulse. Counters can be formed by connecting individual flip-flops together. On application of pulses, the flip-flops in the counter undergo a change of state in such a manner that the binary number stored in the flip-flops represents the number of pulses applied at input. When clock pulses are applied to a counter, the counter progresses from one state to another and the final output of the flip-flop in the counter indicates the pulse count. If all the flip-flops are not clocked at the same time, the counter is *asynchronous (or Ripple)* and if they are clocked simultaneously, the counter is *synchronous*. In practice, there are two types of counters:

- up counters, for increment in value
- down counters, for decrement in value

**Frequency Division:** For frequency division, toggle mode flip-flops are used in a chain as a divide by two- counter. One flip-flop will divide the input clock frequency by 2, two flip-flops will divide it by 4 (and so on). One benefit of using toggle flip-flops for frequency division is that the output at any point has an exact 50% duty cycle. The final output clock signal will have a frequency value equal to the input clock frequency divided by the MOD number of the counter. Such circuits are known as "divide-by-n" counters, where "n" is the number of counter stages used.

### **Binary ripple Up-counter:**

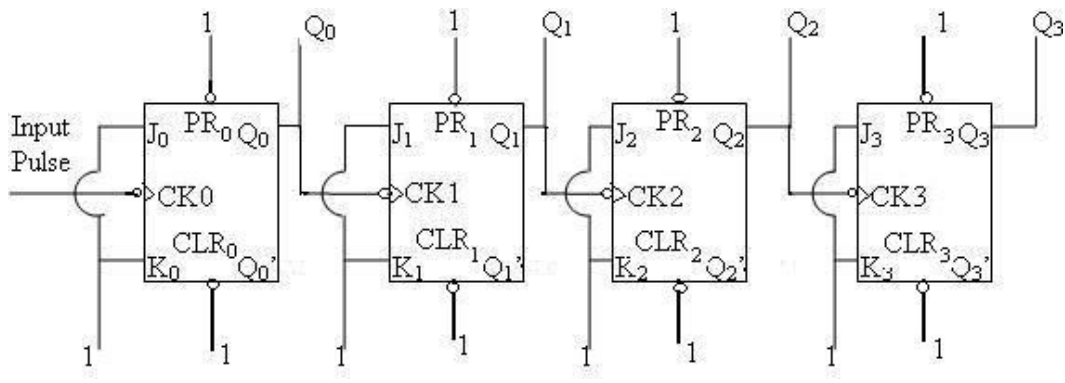
We will consider a basic *4-bit binary up counter*, which belongs to the class of *asynchronous counter* circuits and is commonly known as a *ripple counter*. Since a flip-flop has two states, a counter having n flip-flops will have  $2^n$  states. Hence, in this case the counter will have  $2^4$  or 16 states.

The schematics below shows a 4-bit up-counter implemented with four JK flip-flops. It can be noticed that the normal output of each flip-flop is connected to the clock input of next flip-flop. Please recall that in case of JK flip-flop, with  $J=K=1$ , if an input clock pulse is supplied, the output toggles during the positive or negative (which is the

case here, i.e. transition of pulse from 1 to 0) edge of the pulse. The count held by this counter is read in the reverse order from the order in which the flip-flops are triggered. Thus, output  $Q_3$  is the highest order of the count, while output  $Q_0$  is the lowest order. The binary count held by the counter is then  $Q=Q_3Q_2Q_1Q_0$ .

Let us start from the reset condition of all the flip-flops so that counter reads 0000 (decimal 0). When the trailing or negative end of the first pulse arrives, the first flip-flop gives an output  $Q_0=1$ , which does not affect the second flip-flop and the counter reads 0001. The counting sequence corresponding to each input pulse is summarized in the table below. On supplying 15<sup>th</sup> pulse the counter reads 1111 (decimal 15). The next clock pulse after count 1111 will cause the counter to try to increment to 10000 (decimal 16). However, that 1 bit is not held by any flip-flop and is therefore lost. As a result, the counter actually reverts to 0000, and the count begins again.

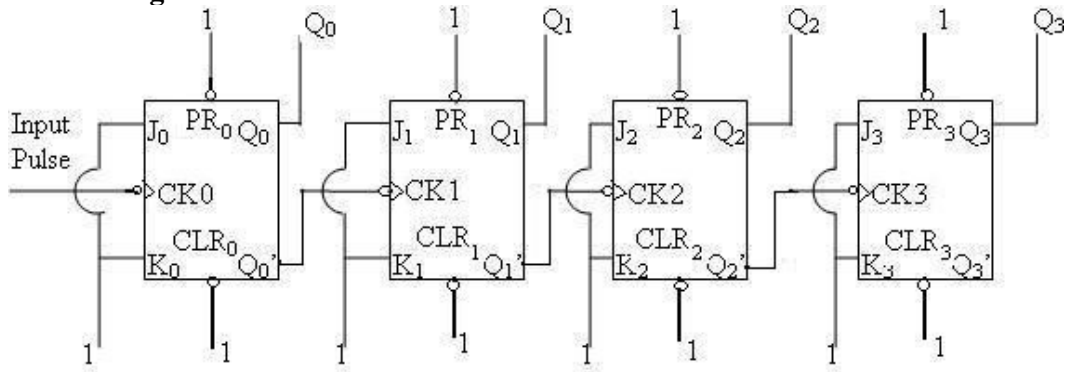
**Circuit Diagram:**



**Binary ripple Down-counter:**

The binary ripple down-counter decreases the count by one each time a pulse occurs at the input. The only difference it has from the up-counter is that the complement output of one flip-flop is connected to the clock input of the subsequent flip-flop. Here the complement output toggles at each negative edge of the clock pulse (1 to 0 transition), which is equivalent to a normal output toggling for positive edge of the clock pulse (0 to 1 transition). The counter starts from 1111 with the first pulse after it is reset and reverts back to 0000 after 15 pulses.

**Circuit diagram:**



**Characteristic Tables:**

**UP COUNTER**

**DOWN COUNTER**

Input pulse	Binary count				Decimal count	Input pulse	Binary count				Decimal count
	Q <sub>3</sub> 2 <sup>3</sup>	Q <sub>2</sub> 2 <sup>2</sup>	Q <sub>1</sub> 2 <sup>1</sup>	Q <sub>0</sub> 2 <sup>0</sup>			Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	
						0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	15
1	0	0	0	1	1	2	1	1	1	0	14
2	0	0	1	0	2	3	1	1	0	1	13
3	0	0	1	1	3	4	1	1	0	0	12
4	0	1	0	0	4	5	1	0	1	1	11
5	0	1	0	1	5	6	1	0	1	0	10
6	0	1	1	0	6	7	1	0	0	1	9
7	0	1	1	1	7	8	1	0	0	0	8
8	1	0	0	0	8	9	0	1	1	1	7
9	1	0	0	1	9	10	0	1	1	0	6
10	1	0	1	0	10	11	0	1	0	1	5
11	1	0	1	1	11	12	0	1	0	0	4
12	1	1	0	0	12	13	0	0	1	1	3
13	1	1	0	1	13	14	0	0	1	0	2
14	1	1	1	0	14	15	0	0	0	1	1
15	1	1	1	1	15	16	0	0	0	0	0 (RESET)
16	0	0	0	0	0 (RESET)						

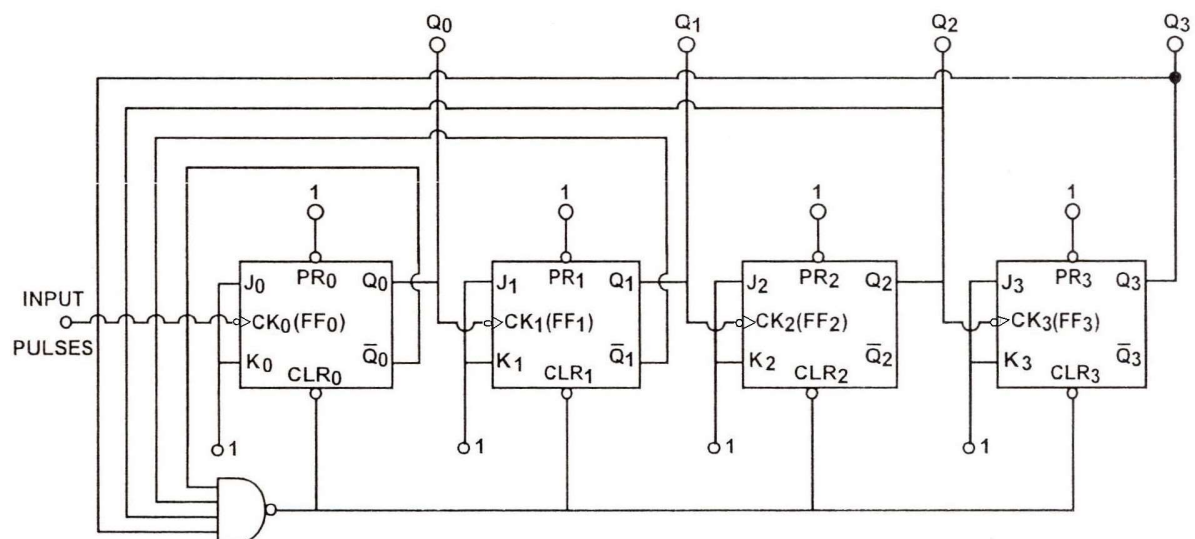
**Modulus-12 Counters:**

The modulus of a counter is the number of discrete states a counter can take up. A counter with n no. of flip flops will have 2<sup>n</sup> number of possible states. So counters with modulus, for example, 2, 4, 8, 16, can be built up using 1, 2, 3, 4 flip flops. It is quite

often desirable to construct a counter having a modulus of 5, 9 or 12 etc. To design counters of modulus-12 (say), one has to use a modulus 16 counter and to arrange the circuit in such a way that it skips some of its natural states restricting it to 12. The simplest way of doing this is the direct clearing method, where a gate circuit is used to clear all the flip flops as the desired count is reached. Thus, for a modulus N counter, the number n of flip-flops should be such that n is the smallest number for which  $2^n > N$  and then to skip the surplus states with some rearrangements of the circuit.

The circuit diagram for a Mod-12 counter is shown below. It is obvious that a mod-12 counter will require 4 flip-flops which when connected as a counter, will provide 16 states. This counter counts 0, 1, 2, ..., 15 and then it resets to 0. For a mod-12 counter, one may skip state 12 and return to state 0 from state 11 and the cycle should continue this way. For this an additional combinational logic circuit, i.e. a 4-input NAND gate is required, whose output is connected to clear terminal of all the flip flops. This will feed a reset pulse to the counter during state 12 (1100) and immediately after state 11 (1011). The flip-flops are reset and the counter starts counting again.

### Circuit diagram:



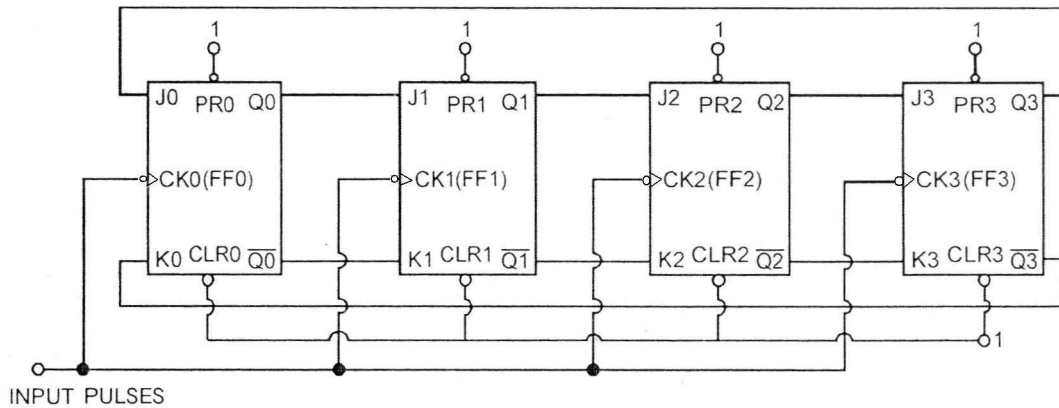
### Ring Counter:

Ring counters provide a sequence of equally spaced timing pulses and hence find considerable application in logic circuits which require such pulses for setting in motion a series of operations in a predetermined sequence at precise time intervals. A ring counter consists of an array of coupled flip-flops and the last flip-flop is coupled back to the first as shown in the circuit diagram. If one of the flip-flops is in the SET (or 1) state and the

others are in the RESET (or 0 state) and then applying clock pulses, the logic 1 will advance by one flip-flop around the ring for each pulse. The sequence of operation of the ring counter is summarized in the characteristic table. The logic 1 will return to the original flip-flop after exactly 4 clock pulses (shown in shades) for a 4-bit ring counter.

Ring counter is extremely fast but it is uneconomical in the number of flip-flops (A simple mod-8 counter requires 4 flip-flops whereas a mod-8 ring counter needs 8!!). This is overcome by a modified circuit known as a Johnson counter or switchtail ring counter or twisted counter, where the outputs of the last flip-flop are crossed over and then fed back to the first flip-flop. That is the normal and complement output of the last flip-flop are connected to the K and J inputs of the first flip-flop respectively.

**Circuit diagram:**



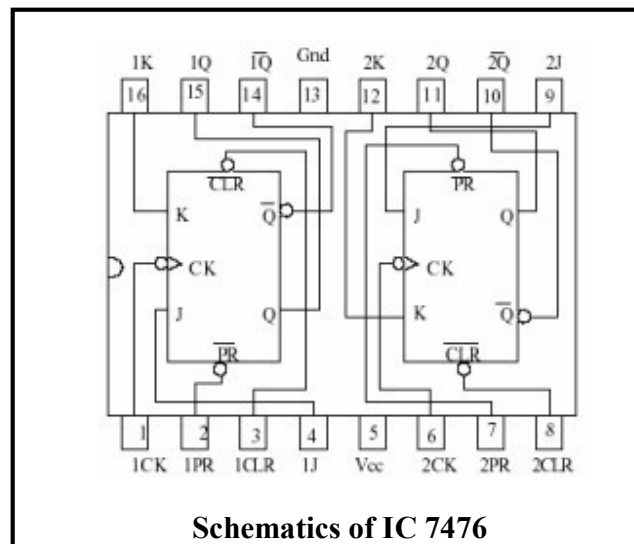
**Characteristic Table:**

No. of Input pulses	Binary states			
	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	0	0	1
5	0	0	1	0
6	0	1	0	0
7	1	0	0	0
8	0	0	0	1

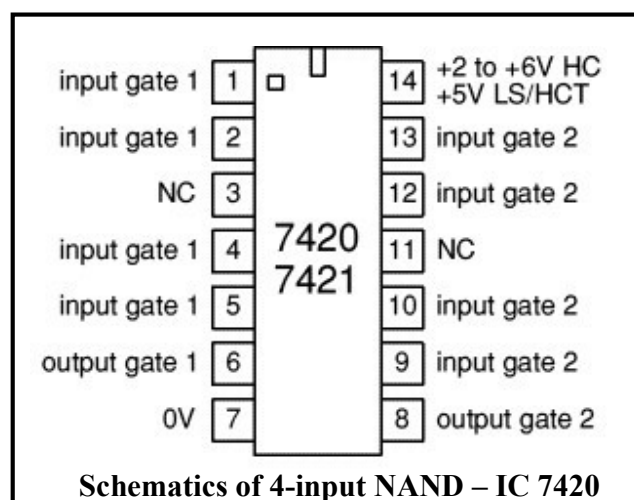
**Circuit components/Equipments:**

1. Resistors (1K $\Omega$ , 5 Nos)
2. ICs [JK FF-7476, 2 Nos; OR-7432,1 No; 2-input AND- 7408, 2 Nos; 4-input NAND -7420, 1 No; NOT-7404, 1 No]
3. A Surface mount dip switch
4. D.C. Power supply (5V)
5. Function Generator
6. Oscilloscope
7. Red/Green LEDs (4 Nos)
8. Connecting wires
9. Breadboard

**Circuit Diagrams:** Already provided with text.



**Schematics of IC 7476**



**Schematics of 4-input NAND – IC 7420**

### Procedure:

1. Assemble the circuits one after another on your breadboard as per the circuit diagrams. Circuit diagrams given here do not show connections to power supply and LEDs assuming that you are already familiar with it from your previous lab experience. Here, all the LEDs are connected to the normal output of each flip flop. You will also use the oscilloscope to compare the timing diagrams of each of the output terminal and the input.
2. Connect the ICs properly to power supply and ground following the schematics for ICs given above.
3. Using dip switch and resistors, facilitate the required inputs from the power supply to the J, K, Pr and Cr terminals of the IC.
4. Use the function generator to facilitate clock pulse input to the circuit.
5. Turn on power to your experimental circuit.
6. **Reset the circuit before applying pulse. For ring counter preset the first flip-flop to give 1 at its normal output before applying pulse. After verifying it try other combinations.**
7. Set the function generator in “Pulse” mode by pressing the “Function” button. Set the frequency at a very low value ( $\sim 1$  Hz, amplitude  $\sim 5$  V) so that you can notice the logic states of the normal outputs indicated by the LEDs (ON = 1; OFF = 0).
8. The logic states of the J, K inputs must not be allowed to change when clock is high.
9. Record the normal output states of all the flip flops in a table for every pulse applied. Determine characteristic table for each operation.
10. **Feed the input and each of the output of the up-counter to oscilloscope.** Save the waveforms (Timing diagram) and compare their frequencies.
11. When you are done, turn off the power to your experimental circuit.

### Observations:

Table for ripple Up-counter: \_\_\_\_\_ Timing Diagram for up-counter: \_\_\_\_\_

Table for ripple Down-counter: \_\_\_\_\_ Table for Mod-12 counter: \_\_\_\_\_





Table for ring counter : \_\_\_\_\_

### Discussions:

### Precautions:

1. Watch out for loose connections.
2. The logic states of the J, K inputs must not be allowed to change when clock is high.


**Appendix:** IC 7476 datasheet

Inputs					Outputs	
PR	CLR	CLK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
H	H		L	L	$Q_0$	$\bar{Q}_0$
H	H		H	L	H	L
H	H		L	H	L	H
H	H		H	H	Toggle	

H = High Logic Level

L = Low Logic Level

X = Either Low or High Logic Level

 = Positive pulse data. The J and K inputs must be held constant while the clock is high. Data is transferred to the outputs on the falling edge of the clock pulse.

$Q_0$  = The output logic level before the indicated input conditions were established.

Toggle = Each output changes to the complement of its previous level on each complete active high level clock pulse.

**Note 1:** This configuration is nonstable; that is, it will not persist when the preset and/or clear inputs return to their inactive (high) level.