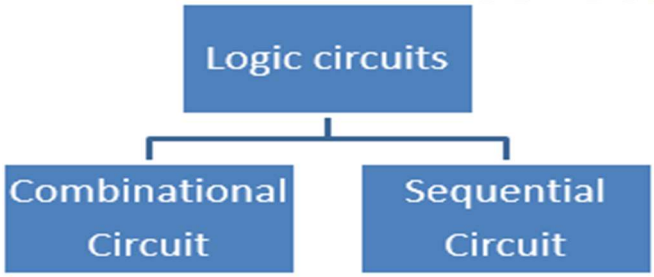
CHAPTER 5
LOGIC CIRCUITS

5.1 Preamble

In digital system, logic circuits generally fall into two categories, both combinational and sequential circuits are the most widely used circuits. These are two broad categories of circuits defined in digital electronics where one type of circuit is independent of time and the other is dependent on time.



Following are the important differences between combinational and sequential circuit:

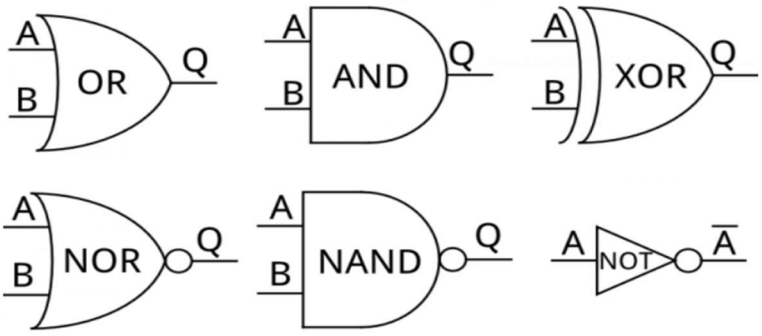
Basis for Comparison	Combinational Circuits	Sequential Circuits
Basic	The output is discovered by the present state of the inputs.	Both the present input and past state output are used to identify the output.
Storage capability	Does not store data.	Can store a small amount of data.
Application	Used in adders, Subtractor.	Used in flip-flop and latches.
Clock	Circuits do not rely on the clock.	Clock is utilized for performing triggering functions.
Feedback	No requirement of the feedback.	Feedback is required.

5.2 Combinatorial Logic

Combinatorial logic refers to a digital logic function made of primitive logic gates (AND, OR, NOT, etc.) in which all outputs of the function are directly related to the current combination of values on its inputs. Any changes to the signals being applied to the inputs will immediately propagate through the gates until their effects appear at the outputs.

5.2.1 Basic Combinational Circuits

- AND gate - output is 1 if BOTH inputs are 1.
- OR gate - output is 1 if AT LEAST one input is 1.
- XOR gate - output is 1 if ONLY one input is 1.
- NAND gate - output is 1 if AT LEAST one input is 0.
- NOR gate - output is 1 if BOTH inputs are 0.
- There is a sixth element in digital logic, the inverter (sometimes called a NOT gate). Inverters are not truly gates, as they do not make any decisions. The output of an inverter is a 1 if the input is a 0, and vise versa.



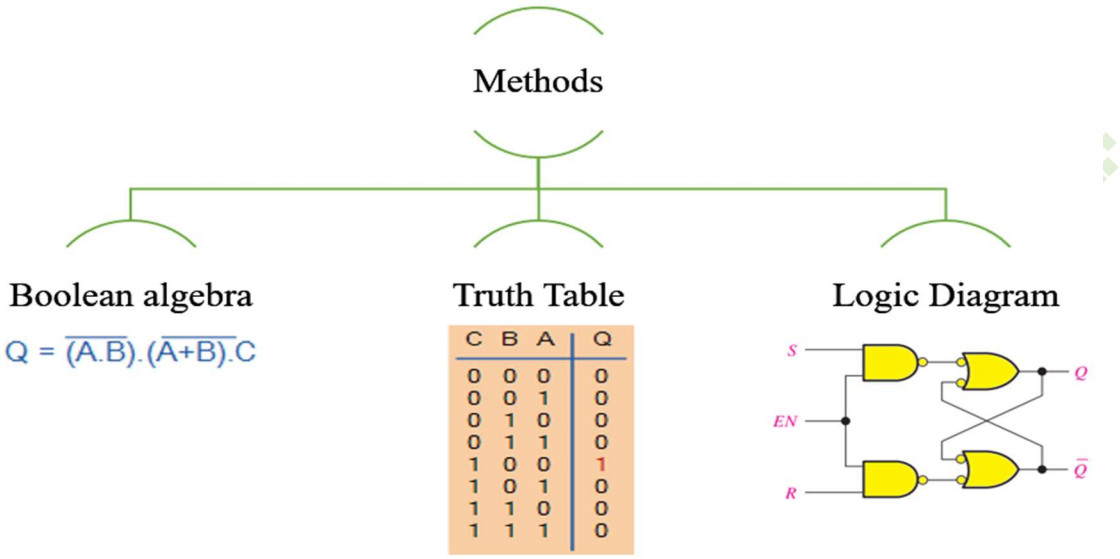
The six basic combinational circuits are built of logic gates

In the above figure, there are two input devices that are standard, we will occasionally see devices with more than two inputs. They will, however, only have one output.

Digital logic circuits are usually represented using these six symbols; inputs are on the left and outputs are to the right. While inputs can be connected together, outputs should never be connected to one another, only to other inputs. One output may be connected to multiple inputs, however.

5.2.2 Specifying the Function of a Combinational Logic Circuit

The three main methods of specifying the function of a combinational logic circuit are:



5.3 Half and Full Adders

An adder is a digital circuit that performs the addition of numbers. In many computers and other kinds of processors, adders are used in the arithmetic logic units or ALU. They are also used in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

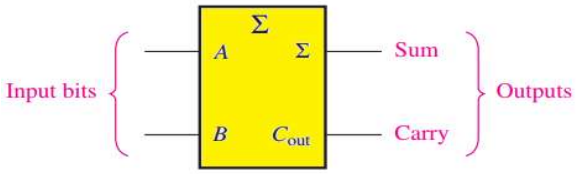
5.3.1 The Half-Adder

The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs—a sum bit and a carry bit. From the operation of the half-adder, as stated in the table below, expressions can be derived for the sum and the output carry as functions of the inputs.

Half-adder truth table.

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

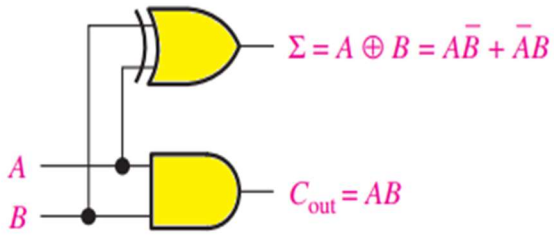
Σ = sum
C_{out} = output carry
A and B = input variables (operands)



Logic symbol for a half-adder

$C_{out=AB}$

$\sum = A \oplus B$



Half-adder logic diagram

Inputs and the sum output is generated with an exclusive-OR gate, as shown in the figure below. Remember that the exclusive-OR can be implemented with AND gates, an OR gate, and inverters.

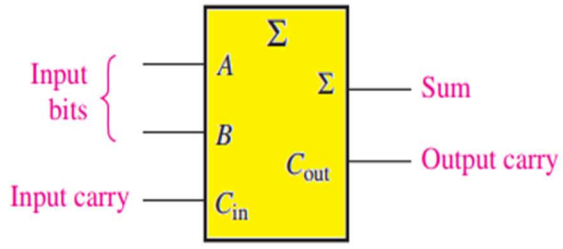
5.3.2 The Full-Adder

The full-adder accepts two input bits and an input carry and generates a sum output and output carry. The basic difference between a full-adder and a half-adder is that the full-adder accepts an input carry. A logic symbol for a full-adder is shown in the figure below, and the truth table in the table below shows the operation of a full-adder.

Full-adder truth table.

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = input carry, sometimes designated as CI
C_{out} = output carry, sometimes designated as CO
Σ = sum
A and B = input variables (operands)



Logic symbol for a full-adder

1. When the three input bits applied are zero the outputs both sum and carry-out are zero.

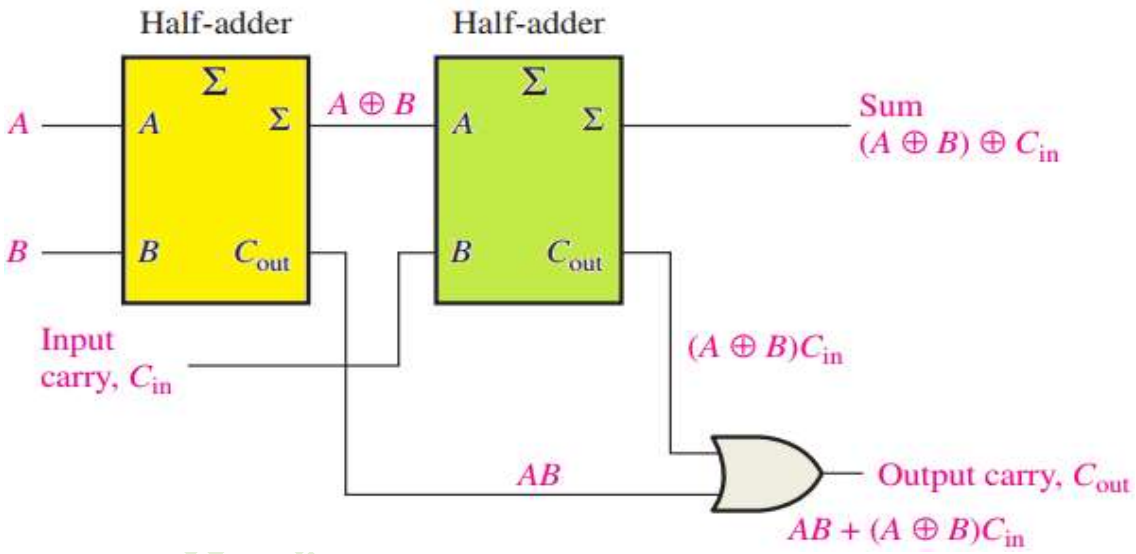
2. The sum output is 1 when only one of the input bit applied is 1 or when all the bits applied are 1.

3. The carry-out can be 1 if two or three inputs applied are 1.

Based on the above table the outputs can be realized in the form of the equation. These equations describe the outputs for any of the combinations.

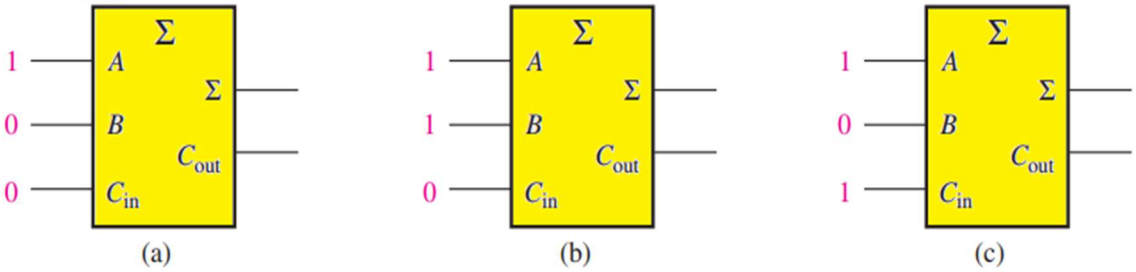
$$\sum = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B)C_{in}$$



Full-adder implemented using half adder

Example: For each of the three full-adders in the figure below, determine the outputs for the inputs shown.



Sol.

(a) The input bits are $A = 1$, $B = 0$, and $C_{in} = 0$.

$1 + 0 + 0 = 1$ with no carry, Therefore, $\sum = \mathbf{1}$ and $C_{out} = \mathbf{0}$.

(b) The input bits are $A = 1$, $B = 1$, and $C_{in} = 0$.

$1 + 1 + 0 = 0$ with a carry of 1, Therefore, $\sum = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

(c) The input bits are $A = 1$, $B = 0$, and $C_{in} = 1$.

$1 + 0 + 1 = 0$ with a carry of 1, Therefore, $\sum = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

H.W.:

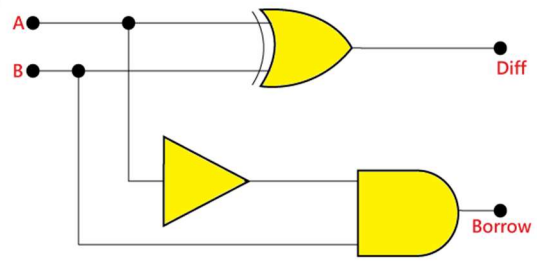
1. Implement Full-adder circuit block diagram using half-adder.
2. What are the full-adder outputs for A = 1, B = 1, and C_{in} = 1?

5.4 Half and Full Subtractor

Subtractor Circuit is a combinational logic circuit that performs subtraction on binary numbers. As the digits involved in Binary Notation are 0 and 1, subtraction of '0' from a '0' or '1' does not change the result. '1' subtracted from '1' results in '0'. Subtracting '1' from '0' requires borrowing.

5.4.1 The Half- Subtractor

The half subtractor is also a building block for subtracting two binary numbers. It has two inputs and two outputs. This circuit is used to subtract two single-bit binary numbers A and B. The 'diff' and 'borrow' are two output states of the half subtractor.



Half- Subtractor Circuit

Truth Table

Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

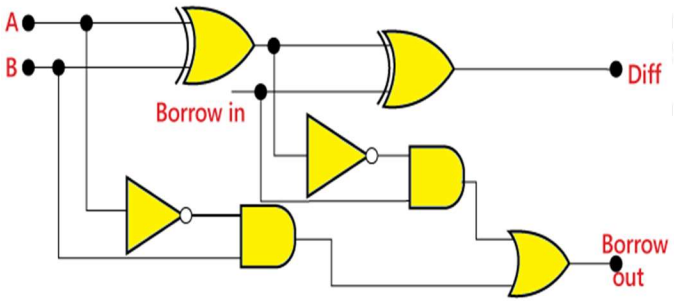
The SOP form of the **Diff** and **Borrow** is as follows:

$$Diff = \bar{A}B + A\bar{B}$$

$$Borrow = \bar{A}B$$

5.4.2 The Full-Subtractor

The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.



Full- Subtractor Circuit

Truth Table

Inputs			Outputs	
A	B	$Borrow_{in}$	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The actual logic circuit of the full subtractor is shown in the above diagram. The full subtractor circuit construction can also be represented in a Boolean expression.

$$Diff = A \oplus B \oplus B_{in}$$

$$Borrow = \overline{A}\,B_{in} + \overline{A}\,B + BB_{in}$$

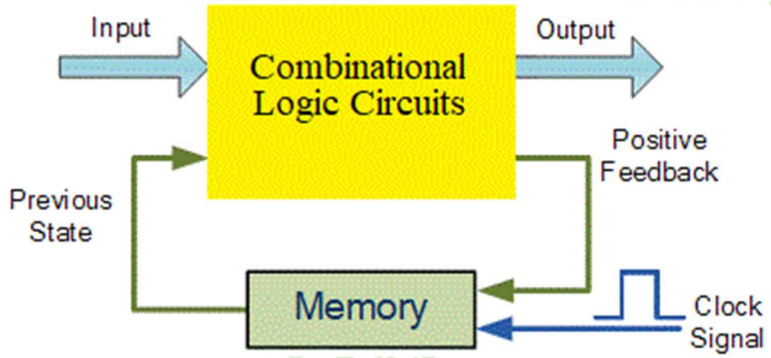
H.W.:

1. Design a full- subtractor logical circuit block diagram using half- subtractor.
2. Design an adder/subtractor circuit using full-adders and gates.

5.5 Sequential Logic Circuits

A sequential circuit is the assimilation of a combinational logic circuit and a storage element (to store binary data). With the applied inputs to the combinational logic, the circuit outputs are derived. These sequential circuits deliver the output based on both the current and previously stored input variables. As sequential circuits work along with the combinational circuit, there are two types of combinational logic inputs where those are:

1. External inputs where these are not monitored by the circuit.
2. Internal inputs are the function of the previous state output.

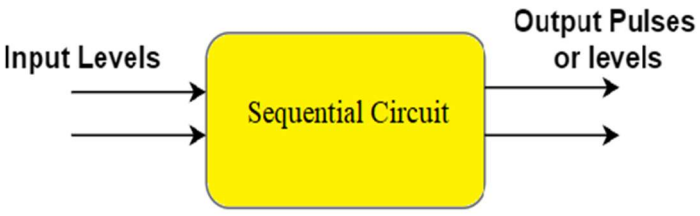


Basic sequential circuit

5.5.1 Types of Sequential Circuits

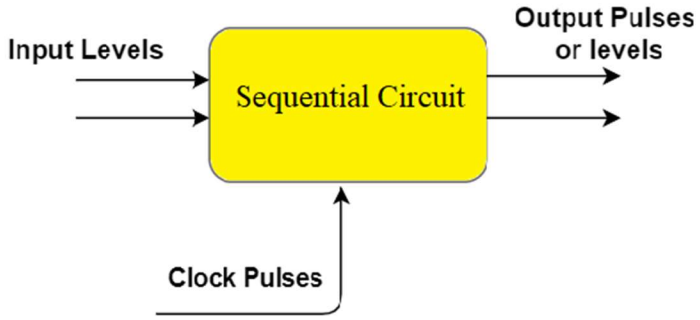
These are of mainly two types and are discussed below:

1. Asynchronous Sequential Circuits: The output can change state at any instant of time when the inputs are changed. The storage element used in an asynchronous sequential circuit called "latches".



Asynchronous sequential circuit

2. Synchronous Sequential Circuits: The output can change state only at a discrete instant of time controlled by a signal called a (clock). The storage element used in a synchronous sequential circuit called "flip-flops".



Synchronous sequential circuit

5.6 Latches

A sequential logic circuit or electronic device used for storing binary information is known as Latches. Latches are bi-stable multi-vibrator; it means that latches have two stable states, LOW and HIGH. A latch is an unclocked flip-flop, or a latch is the basic building block using which clocked flip-flops are constructed. There are various types of latches used in digital circuits which are as follows: (SR Latch, Gated S-R Latch, D latch, Gated D Latch, JK Latch, and T Latch).

5.6.1 The S-R (SET-RESET) Latch

1. Construction of latch by using 2 NOR gates: The logic circuit and the logic symbol for the S-R latch constructed using NOR gates is as shown below:

While constructing a latch using NOR gates, it is compulsory to consider:

Reset input R in normal output Q_n.

Set input S in complemented output $\bar{Q}_n$.

Truth table for the S-R latch constructed using NOR gates is as shown below:

INPUTS			OUTPUTS	REMARKS
R	S	Q_n (Present State)	Q_{n+1} (Next State)	States and Conditions
0	0	X	Q_n	Hold state condition R = S = 0
0	1	X	1	Set state condition R = 0 , S = 1
1	0	X	0	Reset state condition R = 1 , S = 0
1	1	X	Indeterminate	Indeterminate state condition R = S = 1

2. Construction of latch by using 2 NAND gates: The logic circuit and the logic symbol for the S-R latch constructed using NAND gates is as shown below:

While constructing a latch using NAND gates, it is compulsory to consider:

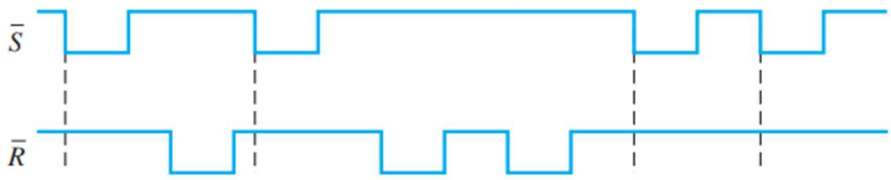
Set input S in normal output Q_n.

Reset input R in complemented output $\bar{Q}_n$.

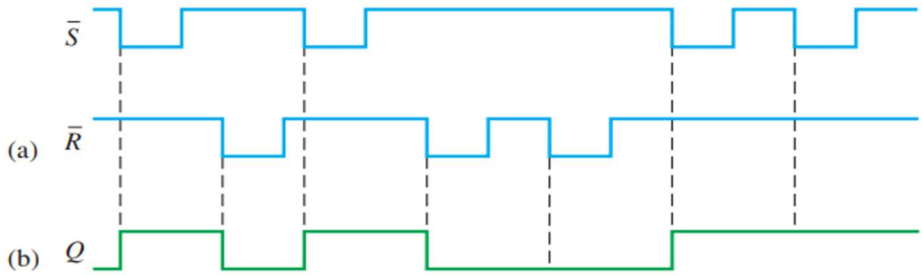
Truth table for the S-R latch constructed using NAND gates is as shown below:

INPUTS			OUTPUTS	REMARKS
S	R	Q_n (Present State)	Q_{n+1} (Next State)	States and Conditions
0	0	X	Indeterminate	Indeterminate state condition S = R = 0
0	1	X	1	Set state condition S = 0 , R = 1
1	0	X	0	Reset state condition S = 1 , R = 0
1	1	X	Q_n	Hold State condition S = R = 1

Example**:** If the $\bar{S}$ and $\bar{R}$ waveforms in figure below are applied to the inputs of the latch, determine the waveform that will be observed on the Q output. Assume that Q is initially LOW.



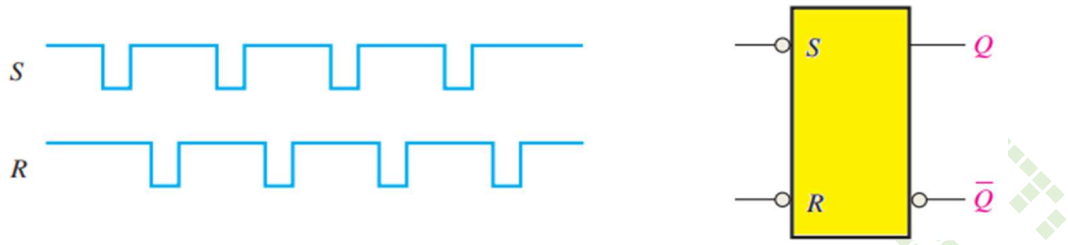
Sol.



H.W.:

1. Determine the Q output of an active-HIGH input S-R latch if the waveforms in figure of previous example are inverted and applied to inputs.

2. If the waveforms in figure below are applied to an active-HIGH S-R latch, draw the resulting *Q* output waveform in relation to the inputs. Assume that Q starts LOW.



5.7 Flip-Flops

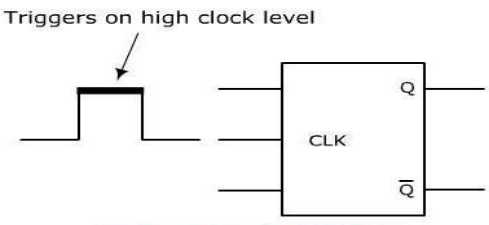
A flip flop is a memory element that is capable of storing one bit of information and has two outputs, and it can maintain a binary state for an unlimited period of time as long as:

- Power is supplied to the circuit.
- Or until it is directed by an input signal to switch states.

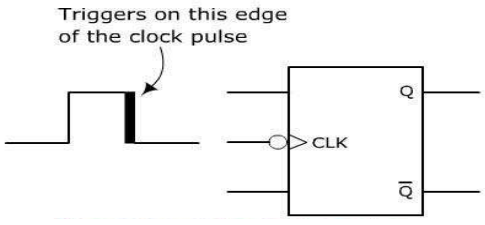
A flip-flop is also called a Bistable Multivibrator because it has two stable states either 0 or 1. There are four basic types of flip-flops: (SR Flip Flop, JK Flip Flop, D Flip Flop, and T Flip Flop).

5.7.1 The Edge-Triggered

Some flip-flops the change on the state takes place when the clock goes from 1 to 0, that is referred to as (trailing edge-triggered) or (negative edge-triggered).

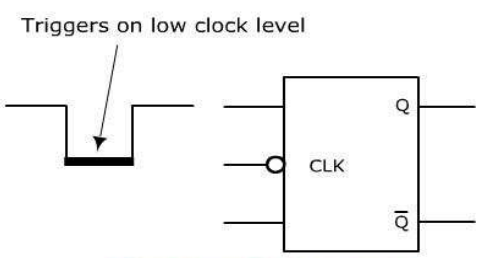


High Level Triggering

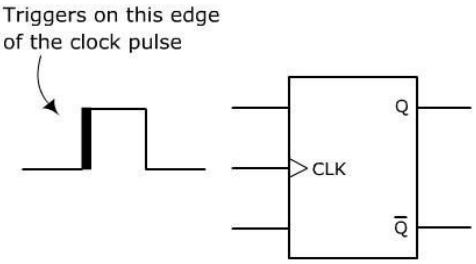


Negative edge-triggered

For others, that changes on the state takes place when the clock goes from 0 to 1, that is referred to as (leading edge-triggered) or (positive edge-triggered).



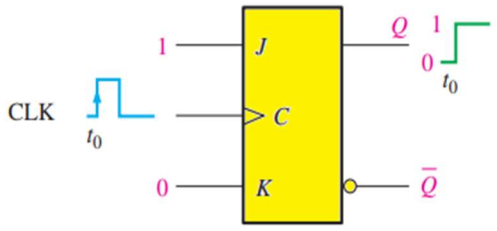
Low Level Triggering



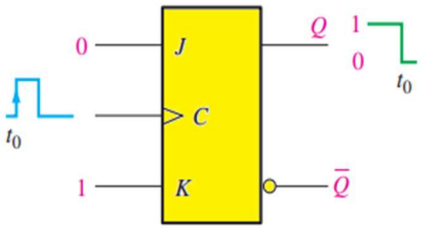
Positive edge-triggered

5.7.2 The JK flip-flop

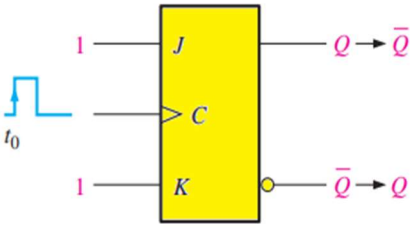
Due to the undefined state in the SR flip-flop, another flip-flop is required in electronics. The JK flip-flop is an improvement on the SR flip-flop where S=R=1 is not a problem. The input condition of J=K=1, gives an output inverting the output state. If J and K data input are different (i.e. high and low) then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. JK Flip-Flops can function as Set or Reset Flip-flops.



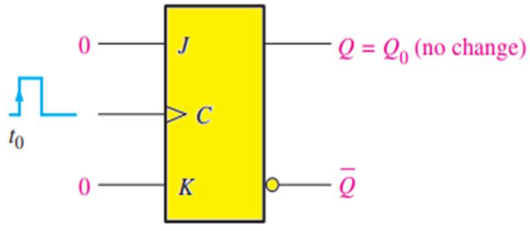
(a) $J = 1$, $K = 0$ flip-flop SETS on positive clock edge. (If already SET, it remains SET.)



(b) $J = 0$, $K = 1$ flip-flop RESETS on positive clock edge. (If already RESET, it remains RESET.)



(c) $J = 1$, $K = 1$ flip-flop changes state (toggle).



(d) $J = 0$, $K = 0$ flip-flop does not change. (If SET, it remains SET; if RESET, it remains RESET.)

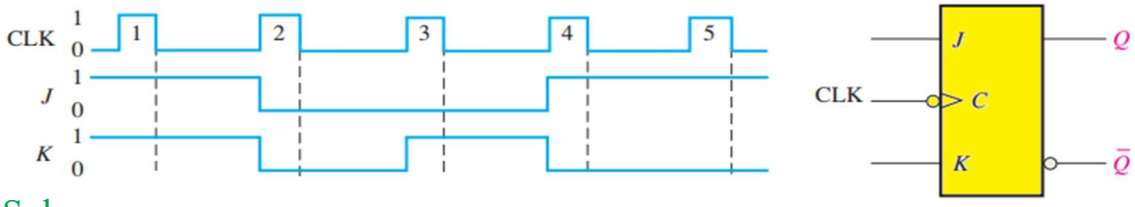
The operation of a positive edge-triggered J-K flip-flop.

Truth table for a positive edge-triggered J-K flip-flop.

Inputs			Outputs		
J	K	CLK	Q	$\overline{Q}$	Comments
0	0	↑	Q_0	$\overline{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Toggle

↑ = clock transition LOW to HIGH
Q_0 = output level prior to clock transition

Example**:** The waveforms in figure below are applied to the J, K, and clock inputs as indicated. Determine the Q output, assuming that the flip-flop is initially RESET.



Sol.

Since this is a negative edge-triggered flip-flop, as indicated by the "bubble" at the clock input, the Q output will change only on the negative-going edge of the clock pulse.

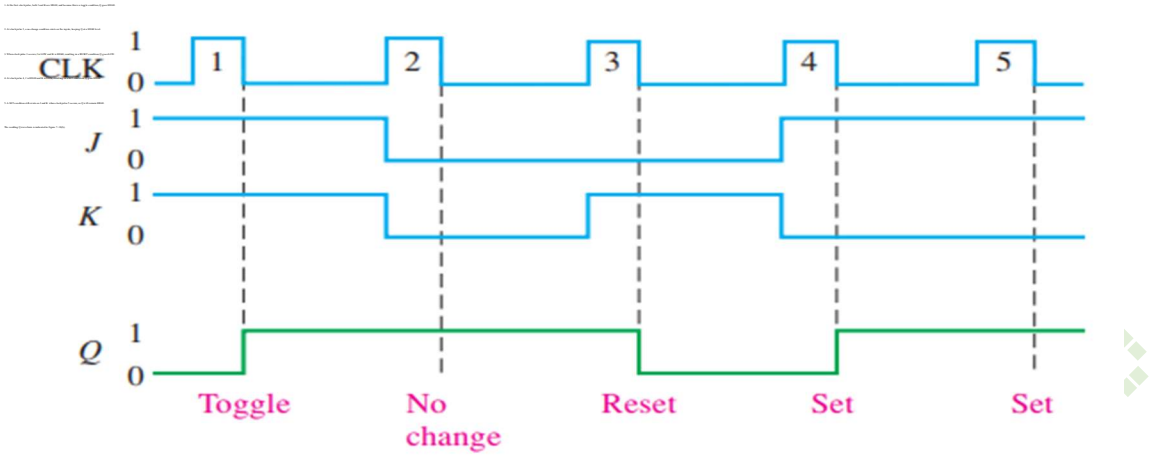
1. At the first clock pulse, both J and K are HIGH; and because this is a toggle condition, Q goes HIGH.

2. At clock pulse 2, a no-change condition exists on the inputs, keeping Q at a HIGH level.

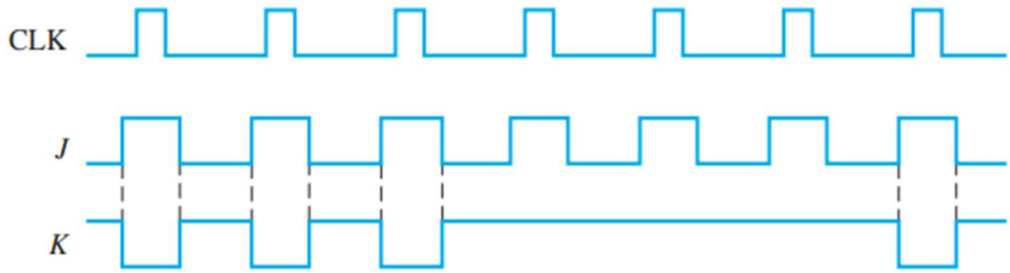
3. When clock pulse 3 occurs, J is LOW and K is HIGH, resulting in a RESET condition; Q goes LOW.

4. At clock pulse 4, J is HIGH and K is LOW, resulting in a SET condition; Q goes HIGH.

5. A SET condition still exists on J and K when clock pulse 5 occurs, so Q will remain HIGH. The resulting Q waveform is indicated in figure below.



H.W.: For a negative edge-triggered J-K flip-flop with the inputs in figure below, develop the Q output waveform relative to the clock. Assume that Q is initially LOW.



5.8 Digital Counters

The counter is a sequential logic circuit implemented using a number of flip-flops that are connected to give the required function. One FF used to count 2 possible states (0 & 1) The FF used to count 4 possible states (00 & 11).

So $N = 2^n$

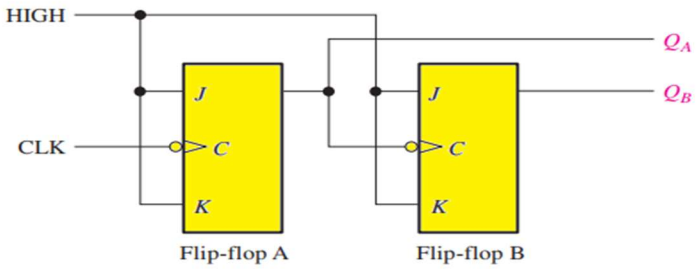
N: Maximum number of counter states.

n: Number of FF used in the counter.

4 ff's = 2^4 = 16 states (0000 → 1111)

5 ff's = 2^5 = 32 states

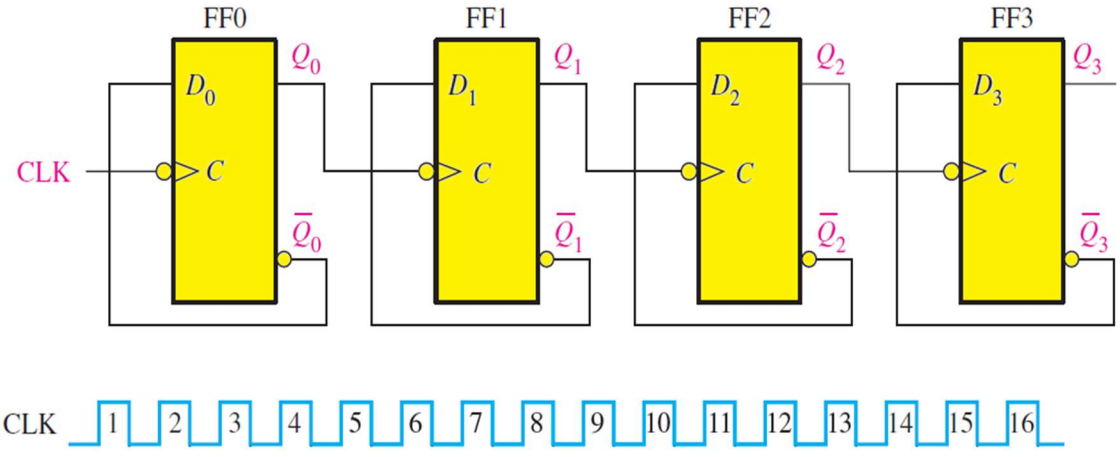
6 ff's = 2^6 = 64 states



5.8.1 Asynchronous Counters

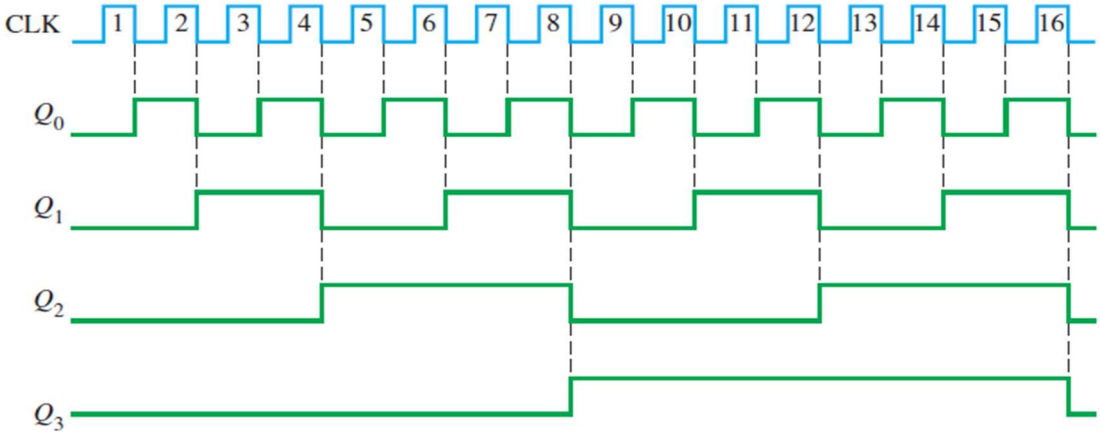
The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time. An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.

Example**:** A 4-bit asynchronous binary counter is shown in the figure below. Each D flip-flop is negative edge-triggered and has a propagation delay for 10 nanoseconds (ns). Develop a timing diagram showing the Q output of each flip-flop, and determine the total propagation delay time from the triggering edge of a clock pulse until a corresponding change can occur in the state of Q_3. Also, determine the maximum clock frequency at which the counter can be operated.



Sol.

The timing diagram with delays omitted is as shown in the figure below.



For the total delay time, the effect of CLK8 or CLK16 must propagate through four flip-flops before Q_3 changes, so

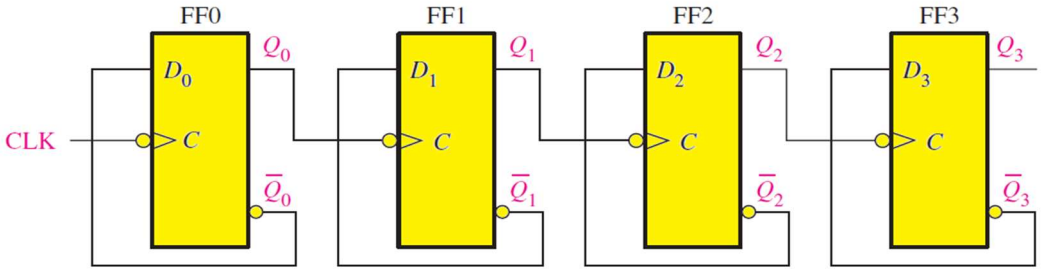
$t_{p(tot)} = 4 * 10 \text{ ns} =$ **40 ns**

The maximum clock frequency is

$f_{max} = 1/ t_{p(tot)} = 1/40 \text{ ns} =$ **25 MHz**

The counter should be operated below this frequency to avoid problems due to the propagation delay.

H.W.: Show the timing diagram if all of the flip-flops in the figure below are positive edge triggered.



5.8.2 Synchronous Counters

A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse. Some mean most be used to control when an FF is to toggle and when it is to remain unaffected by a clock pulse. To design of synchronous counters:

1. State diagram.
2. Next-state table.
3. Flip-Flop transition table.
4. Karnaugh maps and logic expressions for flip-flop inputs.
5. Counter implementation.

Example: Design 2-bit synchronous up counter using J-K flip-flops.

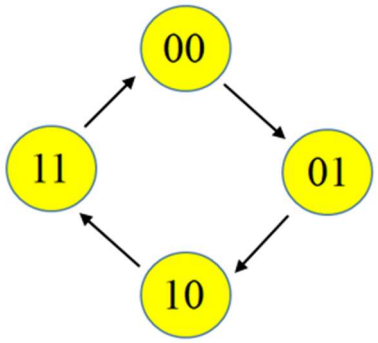
Note: Positive edge triggered.

Sol.

Step 1: A 2-bit up counter requires 2 F.F, the counting sequence are 00-01-10-11. We note that the state diagram count 4 possible states (0 → 3)

Number of states = N = $2^n = 2^2 = 4$

Max. of counts = 4-1 = 3



State diagram for a 2-bit up counter

Step 2: derived the next-state table from the state diagram, is given in the table below:

Present state		Next state	
Q_B	Q_A	Q_B	Q_A
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

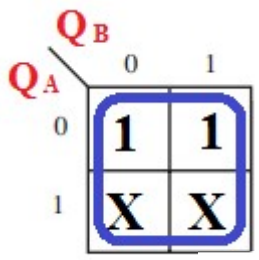
Step 3: The transition table for the J-K flip-flops is repeated in table below.

Transition table for a J-K flip-flop.

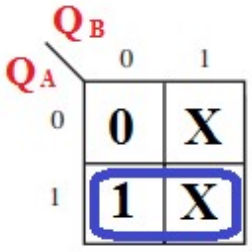
Output Transitions			Flip-Flop Inputs	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Present state		Next state		Input of Flip-Flop			
Q_B	Q_A	Q_B	Q_A	J_B	K_B	J_A	K_A
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

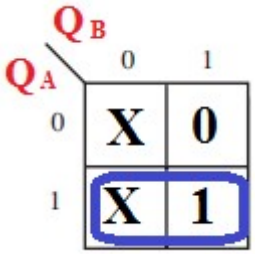
Step 4: Obtain the minimal expression using K-map.



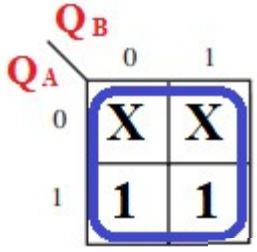
$J_A = 1$



$J_B = Q_A$

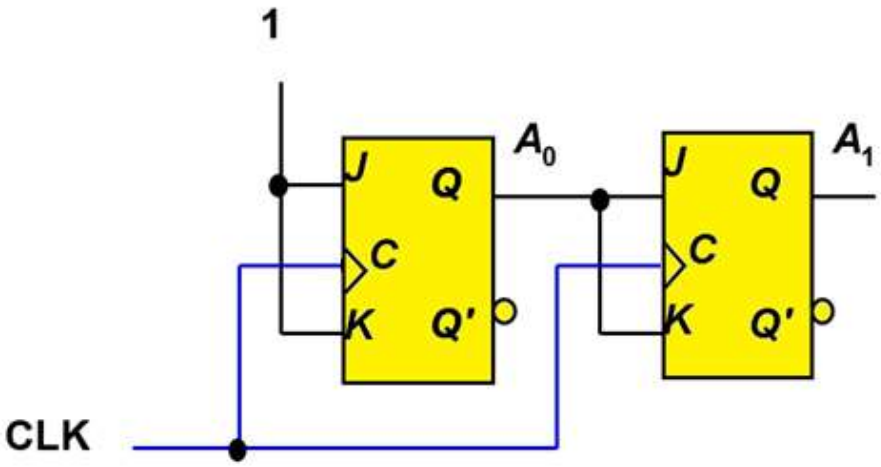


$K_B = Q_A$



$K_A = 1$

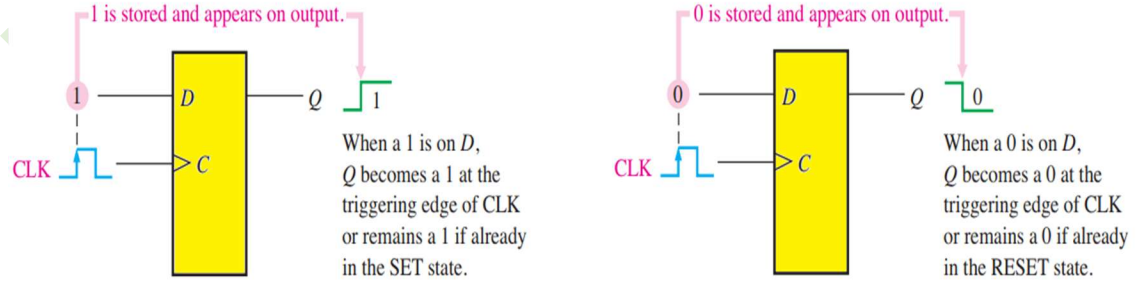
Step 5: The implementation of the counter is shown in the figure below:



5.9 Shift Registers

A register is a storage space for units of memory that are used to transfer data for immediate use by the CPU (Central Processing Unit) for data processing. A register, in general, is used solely for storing and shifting data (1s and 0s) entered into it from an external source and typically possesses no characteristic internal sequence of states, they were used as a form of computer memory.

A shift register is a type of sequential logic circuit using a cascade of flip-flops where the output of one flip-flop is connected to the input of the next. They share a single clock signal, which causes the data stored in the system to shift from one location to the next.

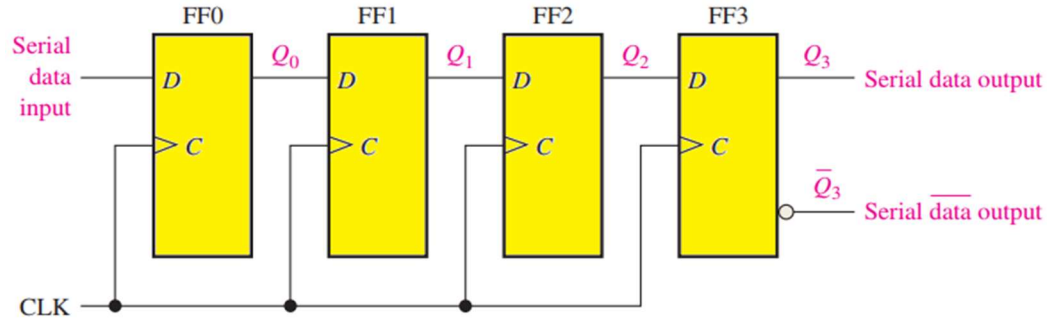


The flip-flop as a storage element.

5.9.1 Serial In/Serial Out (SISO) Shift Registers

- SISO shift register accepts data serially, one bit at a time on a single line.
- It produces the stored information on its output also in serial form.
- Figure below shows a 4-bit device implemented with D flip-flops. With 4 stages, this register can store up to four bits of data.

SISO's logic symbol



SISO's logic diagram

The table below shows the entry of the four bits 1010 into the register in figure above, beginning with the least significant bit. The register is initially clear. The 0 is put onto the data input line, making D = 0 for FF0. When the first clock pulse is applied, FF0 is reset, thus storing the 0.

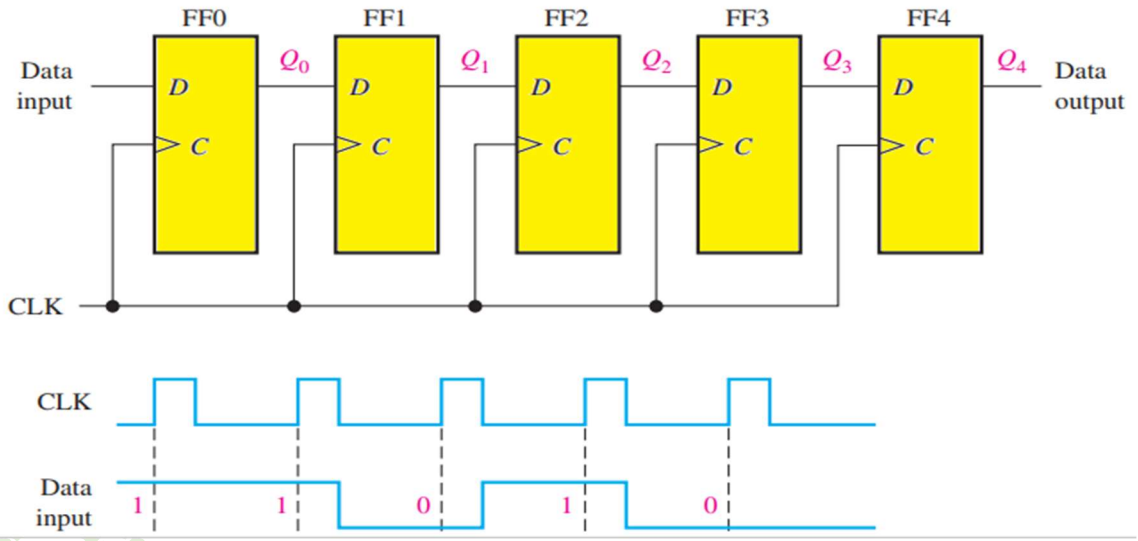
CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	0	0	0	0
1	**0**	0	0	0
2	**1**	**0**	0	0
3	**0**	**1**	**0**	0
4	**1**	**0**	**1**	**0**

If you want to get the data out of the register, the bits must be shifted out serially to the Q_3 output. After CLK4 in the data-entry operation just described, the LSB, 0, appears on the Q_3 output. When clock pulse CLK5 is

applied, the second bit appears on the Q_3 output. Clock pulse CLK6 shifts the third bit to the output, and CLK7 shifts the fourth bit to the output. While the original four bits are being shifted out, more bits can be shifted in. All zeros are shown being shifted in, after CLK8.

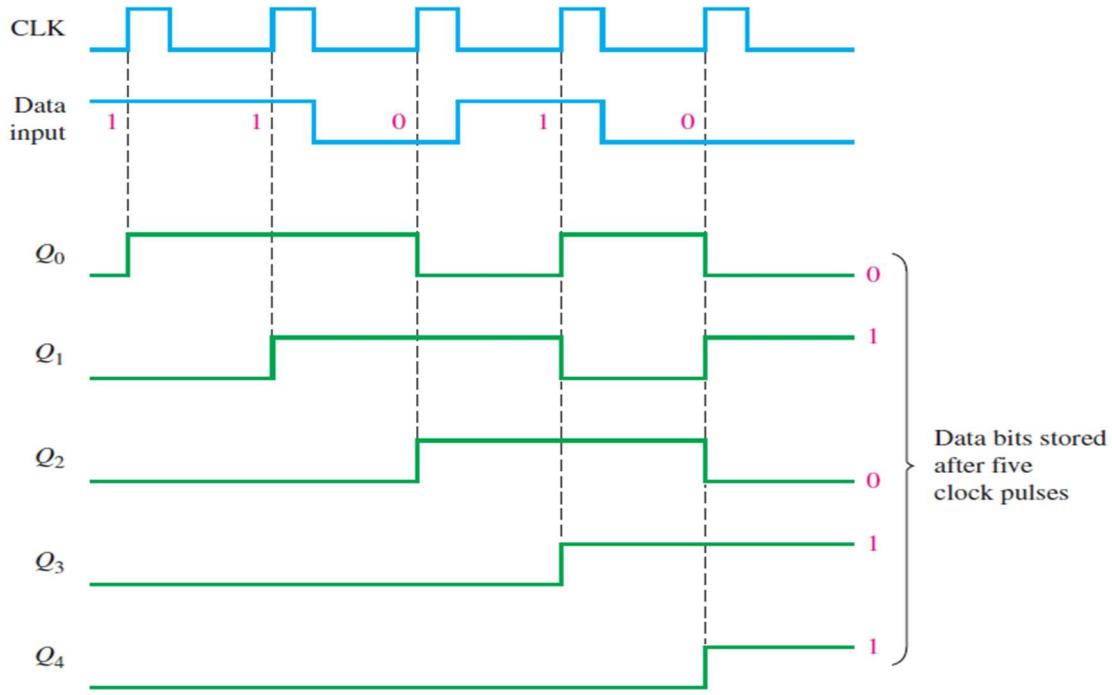
CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	**1**	**0**	**1**	**0**
5	0	**1**	**0**	**1**
6	0	0	**1**	**0**
7	0	0	0	**1**
8	0	0	0	0

Example**:** Show the states of the 5-bit register in the figure below (a) for the specified data input and clock waveforms. Assume that the register is initially cleared (all 0s).



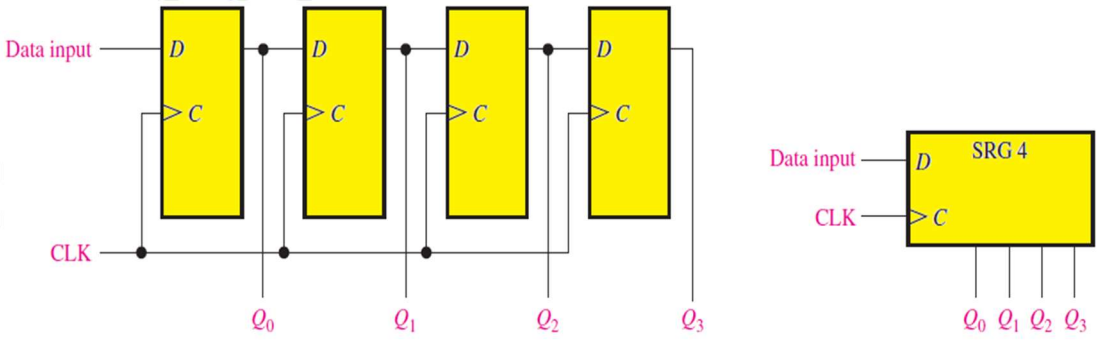
Sol.

The first data bit (1) is entered into the register on the first clock pulse and then shifted from left to right as the remaining bits are entered and shifted. The register contains Q4Q3Q2Q1Q0 = 11010 after five clock pulses.



5.9.2 Serial In/ Parallel Out (SIPO) Shift Registers

- Data bits are entered serially (LSB) into a SIPO shift register.
- The data bits are taken out of the register; in the parallel output register, the output of each stage is available. Once the data are stored.
- The figure below shows a 4-bit SIPO shift register and its logic block symbol.



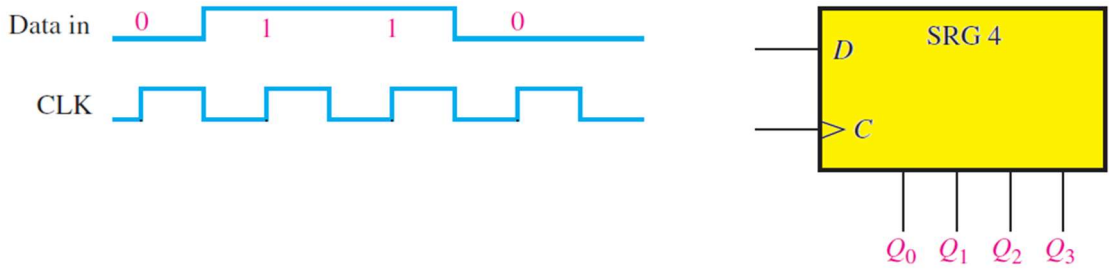
SIPO's logic diagram & logic symbol.

The table below shows the entry of the 4 bits 1011 into the shift register. Beginning with LSB. The register is initially clear.

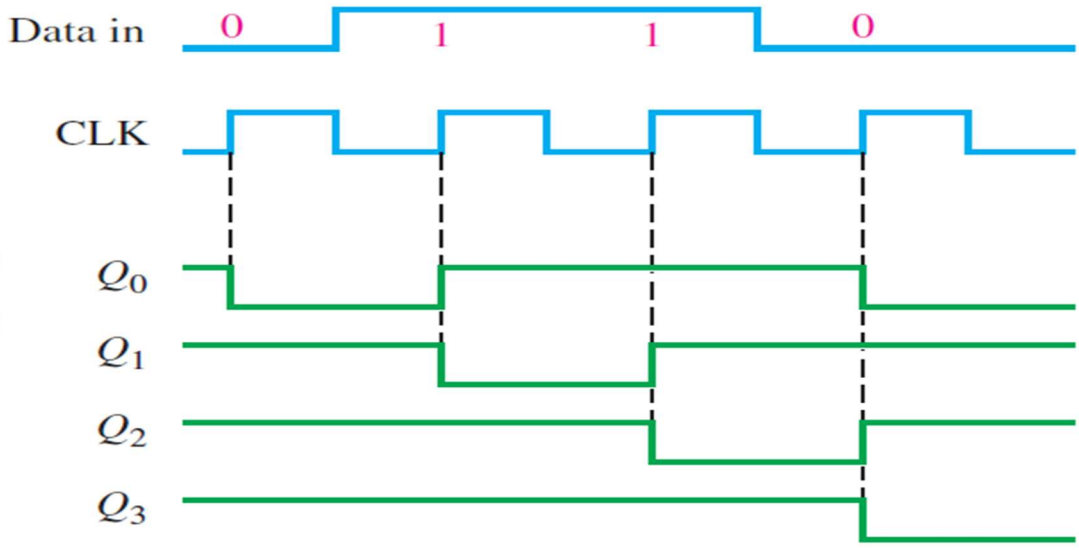
CLK	Q_0	Q_1	Q_2	Q_3
Initial	0	0	0	0
1st CLK	1	0	0	0
2nd CLK	1	1	0	0
3rd CLK	0	1	1	0
4th CLK	1	0	1	1

Note: *In a Serial output shift register, clock pulses are required to read the data, but in the parallel output shift register, clock pulses are not required to read the data.*

Example**:** Show the states of the 4-bit register (SRG 4) for the data input and clock waveforms in the figure below. The register initially contains all 1s.



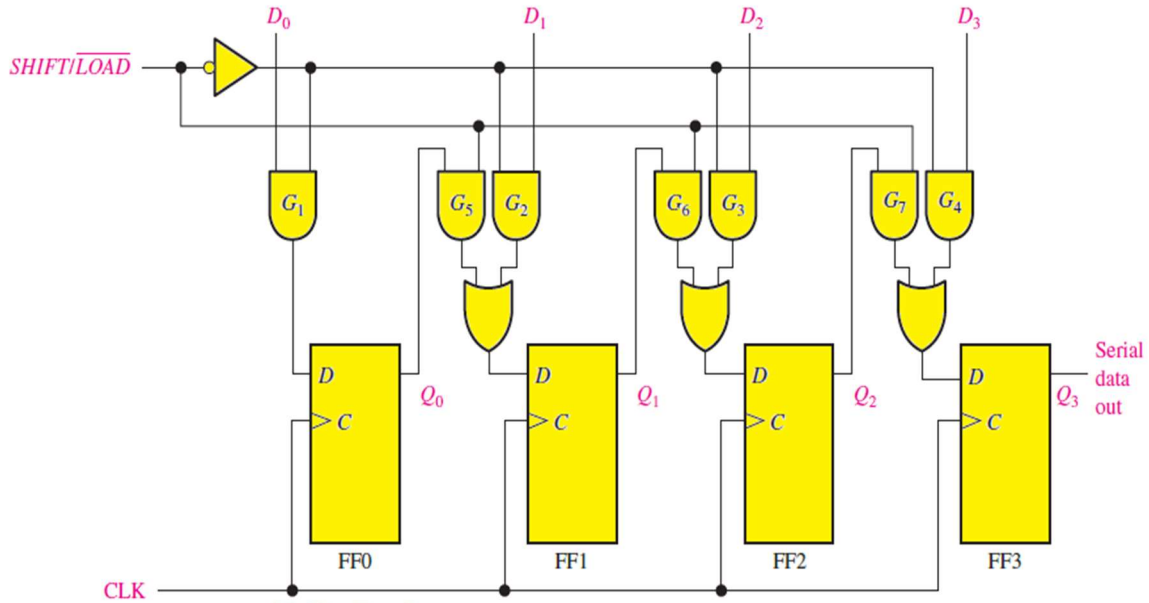
Sol.



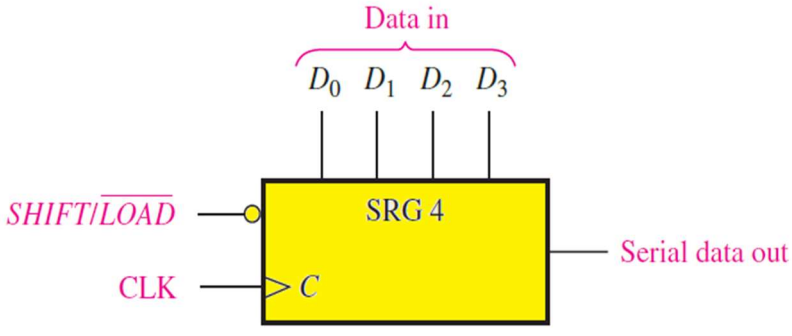
H.W.: If the data input remains 0 after the fourth clock pulse, what is the state of the register after three additional clock pulses?

5.9.3 Parallel In/ Serial Out (PISO) Shift Registers

- The bits are entered simultaneously into their respective stages on parallel lines.
- The serial output is the same as in SISO shift registers, once the data are completely stored in the register.
- The figure below illustrates a 4-bit PISO shift register and a typical logic symbol.



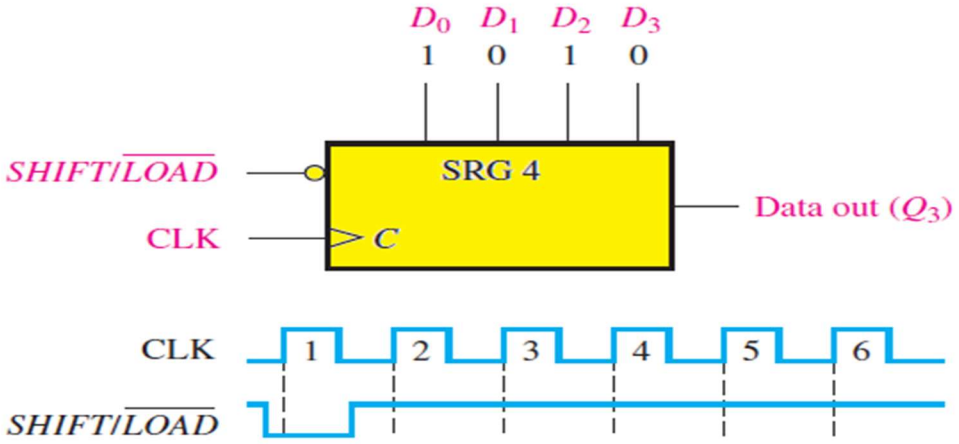
PISO's logic diagram



PISO's logic symbol

There are four data-input lines, D_0, D_1, D_2, and D_3, and a $SHIFT/\overline{LOAD}$ input, which allows four bits of data to load in parallel into the register. When $SHIFT/\overline{LOAD}$ is LOW, gates G_1 through G_4 are enabled, allowing each data bit to be applied to the D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with D = 1 will set, and those with D = 0 will reset, thereby storing all four bits simultaneously. When $SHIFT/\overline{LOAD}$ is HIGH, gates G_1 through G_4 are disabled and gates G_5 through G_7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the $SHIFT/\overline{LOAD}$ input. Notice that FF0 has a single AND to disable the parallel input, D_0. It does not require an AND/OR arrangement because there is no serial data in.

Example**:** Show the data-output waveform for a 4-bit register with the parallel input data and the clock and $SHIFT/\overline{LOAD}$ waveforms given in the figure below. Refer to figure PISO's logic diagram above.



Sol.

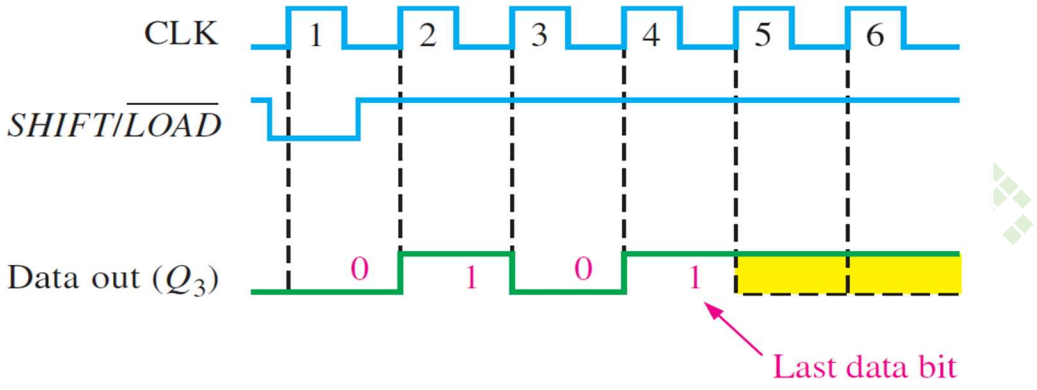
On clock pulse 1, the parallel data ($D_0D_1D_2D_3$ = 1010) are loaded into the register, making Q_3 a 0.

On clock pulse 2, the 1 from Q_2 is shifted onto Q_3.

On clock pulse 3, the 0 is shifted onto Q_3.

On clock pulse 4, the last data bit (1) is shifted onto Q_3.

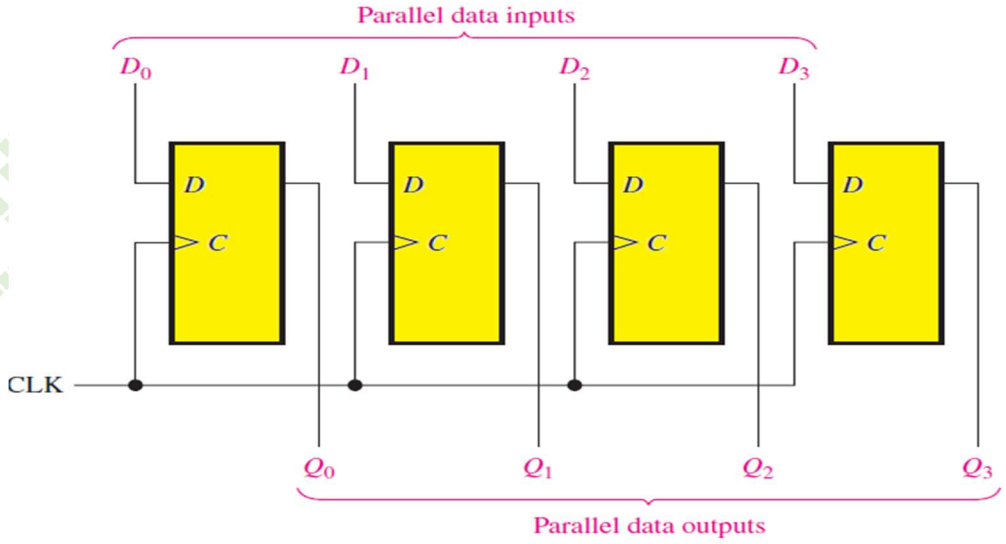
And on clock pulse 5, all data bits have been shifted out, and only 1s remain in the register (assuming the D_0 input remains a 1). As in the figure below.



H.W.: Show the data-output waveform for the clock and $SHIFT/\overline{LOAD}$ inputs shown in the figure of the previous example if the parallel data are D0D1D2D3 = 0101.

5.9.4 Parallel In/ Parallel Out (PIPO) Shift Registers

- There are 4 input D_0, D_1, D_2, and D_3 and 4 output Q_0, Q_1, Q_2 and Q_3.
- When the input bits are applied to the data input, they simultaneously appear on the parallel output on the positive-going edge of the clock pulse.
- The figure below illustrates parallel in/parallel out (PIPO) shift register.



A parallel in/parallel out register

H.W.: In figure below, $D_0 = 1$, $D_1 = 0$, $D_2 = 0$, $D_3 = 1$. What are the data outputs after 3 clock pulse?