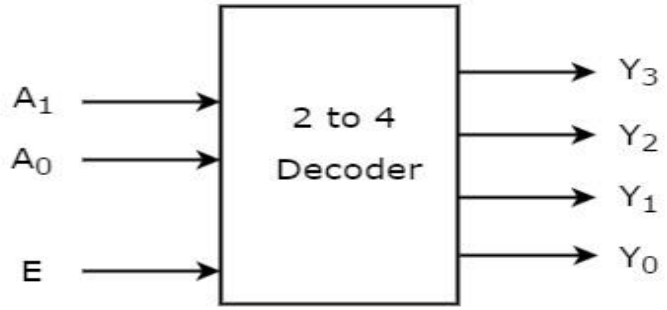
Experiment 5

AIM: Implementing 3–8-line Decoder.

Theory: - Decoder is a combinational circuit that has 'n' input lines and maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the **min terms** of 'n' input variables lines, when it is enabled.

2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A_1 & A_0 and four outputs Y_3, Y_2, Y_1 & Y_0. The **block diagram** of 2 to 4 decoder is shown in the following figure.



One of these four outputs will be '1' for each combination of inputs when enable, E is '1'. The **Truth table** of 2 to 4 decoder is shown below.

Enable	Inputs		Outputs			
E	A1	A0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

From Truth table, we can write the **Boolean functions** for each output as

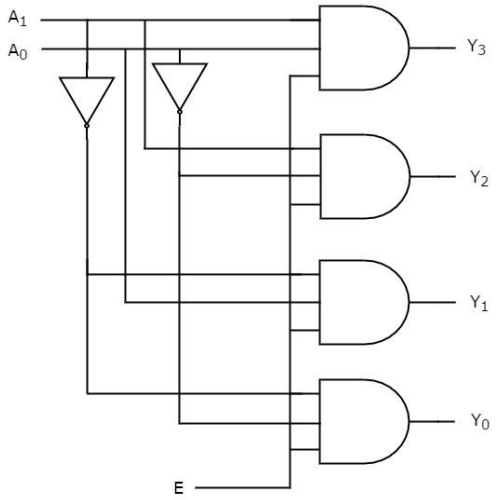
Y0=EA'1.A'0

Y1=EA'1.A0

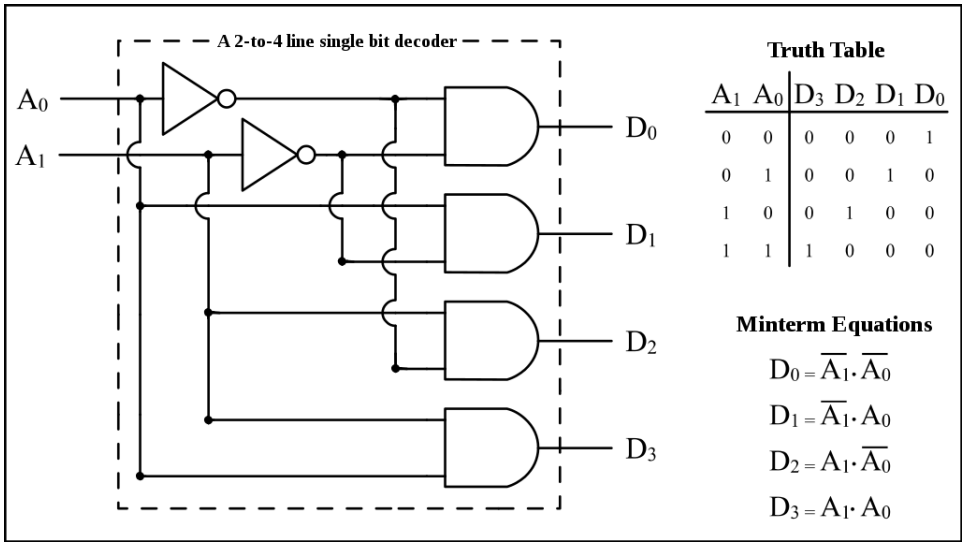
Y2=EA1.A'0

Y3=EA1.A0

Each output is having one product term. So, there are four product terms in total. We can implement these four product terms by using four AND gates having three inputs each & two inverters. The **circuit diagram** of 2 to 4 decoder is shown in the following figure.



By neglecting Enable bit we can design 2-4 Decoder as shown below-



Therefore, the outputs of 2 to 4 decoder are nothing but the **min terms** of two input variables A_1 & A_0, when enable, E is equal to one. If enable, E is zero, then all the outputs of decoder will be equal to zero.

Similarly, 3 to 8 decoder produces eight min terms of three input variables A_2, A_1 & A_0 and 4 to 16 decoder produces sixteen min terms of four input variables A_3, A_2, A_1 & A_0.

3 to 8 Decoder

Similarly 3 to 8 Decoder has three inputs A_2, A_1 & A_0 and eight outputs, Y_7 to Y_0.

We can find the number of lower order decoders required for implementing higher order decoder using following ,

The **Truth table** of 3 to 8 decoder is shown below.

Enable	Inputs	Outputs

E	A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

From Truth table, we can write the **Boolean functions** for each output as

Y0=E A’2.A’1.A’0

Y1=E A’2A’1.A0

Y2=E A’2A1.A’0

Y3=E A’2A1.A0

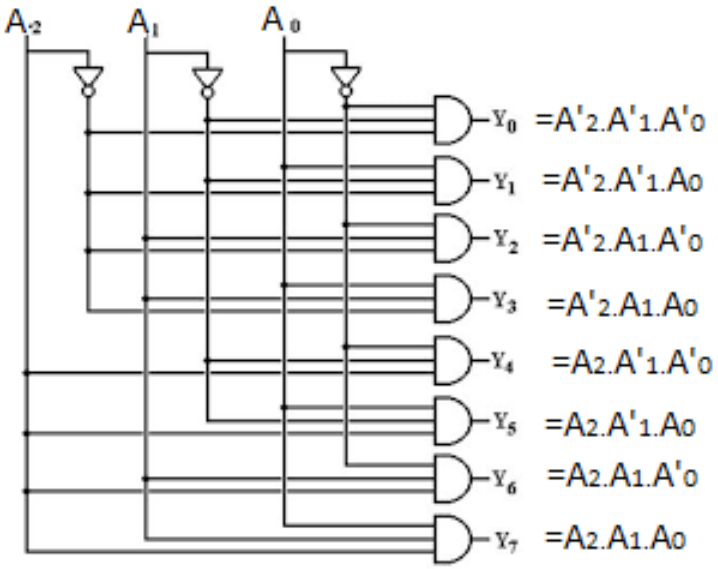
Y4=E A2A’1.A’0

Y5=E A2A’1.A0

Y6=E A2A1.A’0

Y7=E A2A1.A0

By ignoring Enable bit we can design 3-8 line Decoder as shown in Diagram below-



RESULT: Implementation of 3–8-line Decoder done successfully.

EXPERIMENT 6

AIM: Design 4*1 and 8*1 Multiplexers using COA simulator.

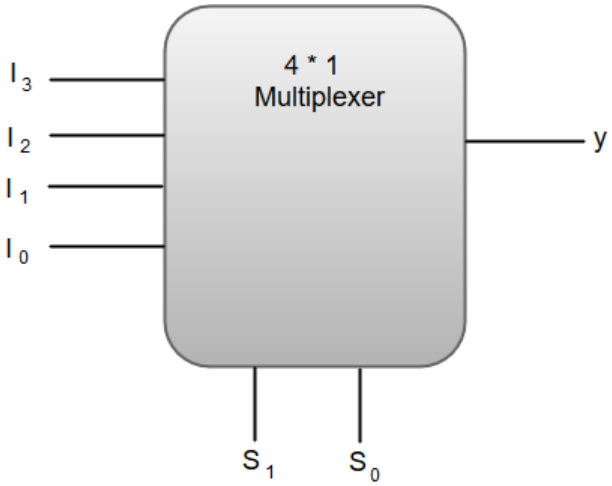
Theory:

Multiplexers:

A Multiplexer (MUX) can be described as a combinational circuit that receives binary information from one of the 2^n input data lines and directs it to a single output line.
The selection of a particular input data line for the output is decided on the basis of selection lines.
The multiplexer is often called as data selector since it selects only one of many data inputs.

Note: A 2^n-to-1 multiplexer has 2^n input data lines and n input selection lines whose bit combinations determine which input data are selected for the output.

Block diagram of 4*1 Multiplexer



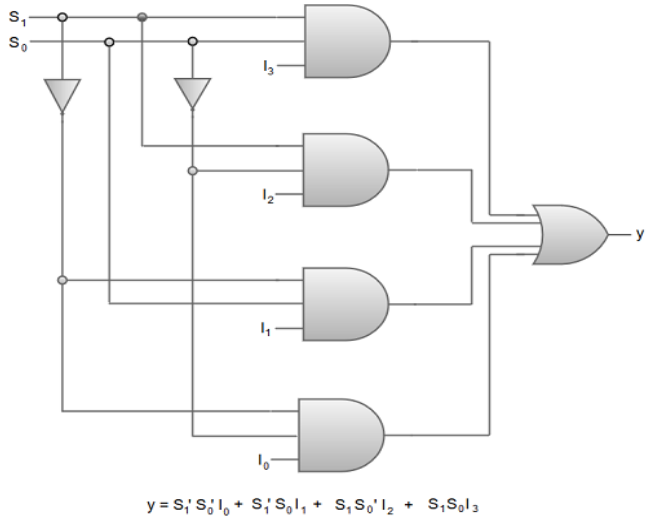
Out of these four input data lines, a particular input data line will be connected to the output based on the combination of inputs present at these two selection lines.

Truth Table of 4*1 Multiplexer

S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

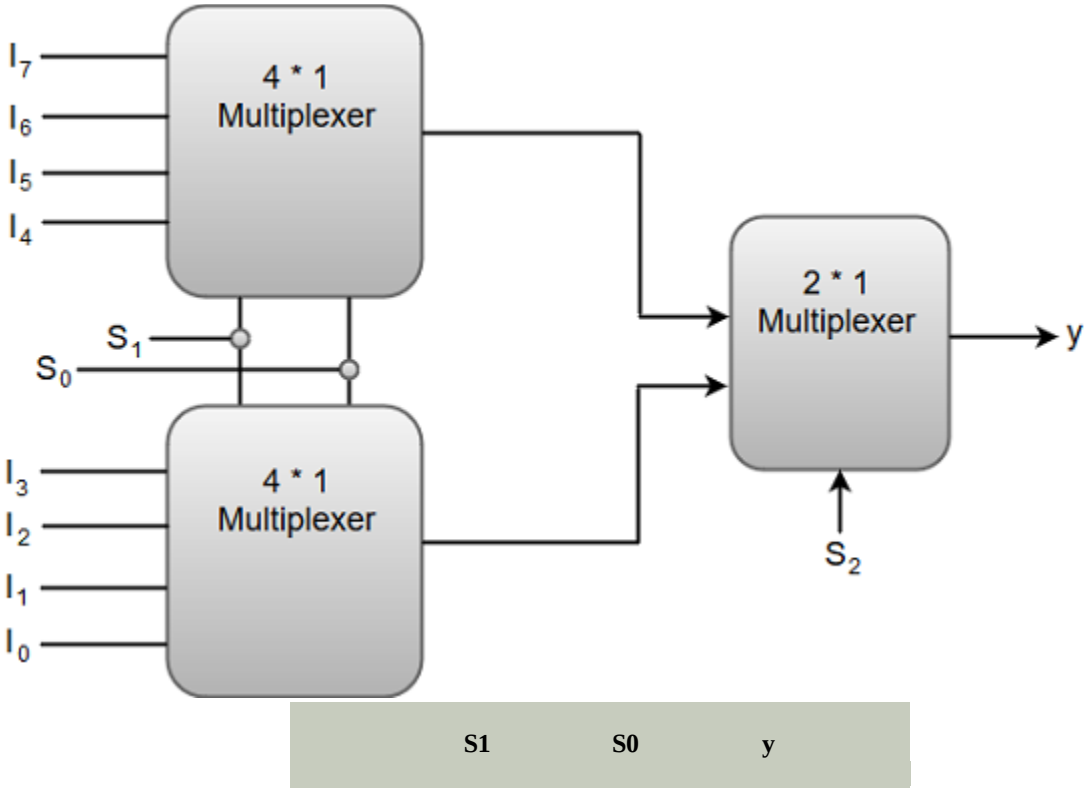
From the function table, we can write the Boolean function for the output (y) as:

y = S1'S0'I0 + S1' S0'I1 + S1S0'I2 + S1S0I3



Digital Circuit Diagram of 4*1 Multiplexer.

We can also implement higher order multiplexers using lower order multiplexers. For instance, let us implement an **8 *1 multiplexer** using two 4*1 multiplexers and a 2*1 multiplexer.
The two 4*1 multiplexers are required in the first stage to get the eight input data lines.
A 2*1 multiplexer is required in the second stage to converge the outputs generated at first stage into a single output.
The following image shows the block diagram of an 8*1 multiplexer designed using two 4*1 multiplexers and a single 2*1 multiplexer.



0	0	0	10
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	17

Function Table of 8*1 Multiplexer.

RESULT: Implementation of multiplexers is done successfully.

Experiment 7

AIM: Verify the excitation tables of various SR and D FLIP-FLOPS.

THEORY: Logic circuits for digital systems are either combinational or sequential. The output of combinational circuits depends only on the current inputs. In contrast, sequential circuit depends not only on the current value of the input but also upon the internal state of the circuit. Basic building blocks (memory elements) of a sequential circuit are the flip-flops (FFs). A flip-flop is a device which stores a single *bit* (binary digit) of data; one of its two states represents a "one" and the other represents a "zero". Such data storage can be used for storage of *state*, and such a circuit is described as sequential logic

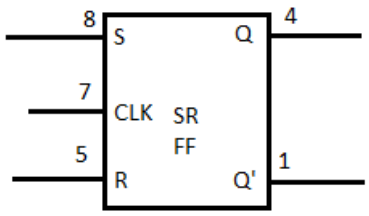
There are four Flip-Flops as follows-

- SR Flip-Flop
- D Flip-Flop
- JK Flip-Flop
- T Flip-Flop

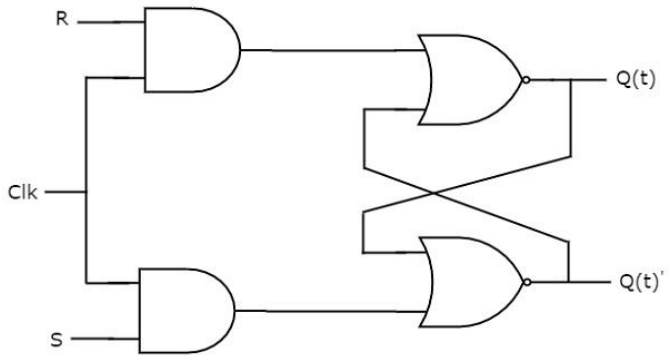
SR Flip-Flop

SR flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, SR latch operates with enable signal.

Block Diagram of SR Flip Flop & Pin Diagram:



The **circuit diagram** of SR flip-flop is shown in the following figure.



This circuit has two inputs S & R and two outputs Qt & Qt'. The operation of SR flipflop is similar to SR Latch. But, this flip-flop affects the outputs only when positive transition of the clock signal is applied instead of active enable.

The following table shows the **state table** of SR flip-flop.

S	R	Qt+1
0	0	Qt
0	1	0
1	0	1
1	1	-

Here, Qt & Qt+1 are present state & next state respectively. So, SR flip-flop can be used for one of these three functions such as Hold, Reset & Set based on the input conditions, when positive transition of clock signal is applied. The following table shows the **characteristic table** of SR flip-flop.

Present Inputs		Present State	Next State
S	**R**	**Qt**	**Qt+1**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1,.
1	0	1	1
1	1	0	x
1	1	1	x

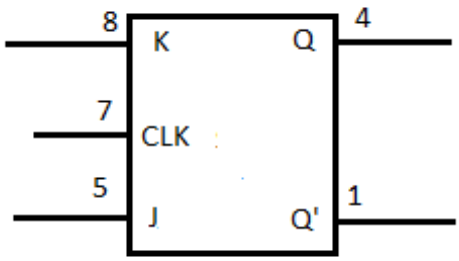
The **simplified expression** for next state Qt+1 is

$$Q(t+1)=S+R'Q(t)Q(t+1)=S+R'Q(t)$$

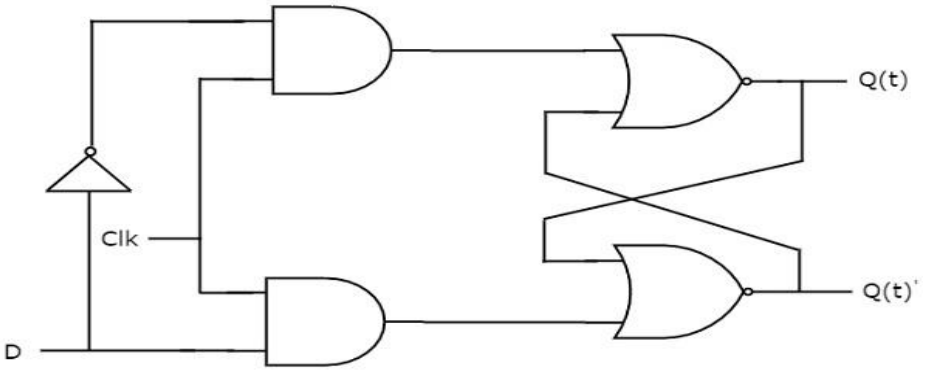
D Flip-Flop

D flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, D latch operates with enable signal. That means, the output of D flip-flop is insensitive to the changes in the input, D except for active transition of the clock signal.

Block Diagram of SR Flip Flop & Pin Diagram



The **circuit diagram** of D flip-flop is shown in the following figure.



This circuit has single input D and two outputs Qtt & Qtt'. The operation of D flip-flop is similar to D Latch. But, this flip-flop affects the outputs only when positive transition of the clock signal is applied instead of active enable.

The following table shows the **state table** of D flip-flop.

D	
0	0
1	1

Therefore, D flip-flop always Hold the information, which is available on data input, D of earlier positive transition of clock signal. From the above state table, we can directly write the next state equation as

$$Qt+1t+1 = D$$

Next state of D flip-flop is always equal to data input, D for every positive transition of the clock signal. Hence, D flip-flops can be used in registers, **shift registers** and some of the counters.

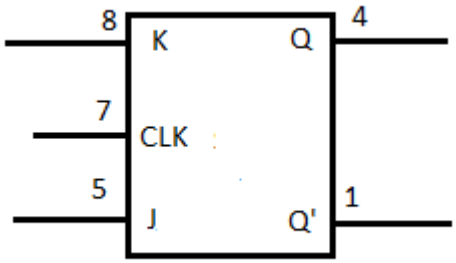
Experiment 8

AIM: Verify the excitation tables of various JK and T FLIP-FLOPS.

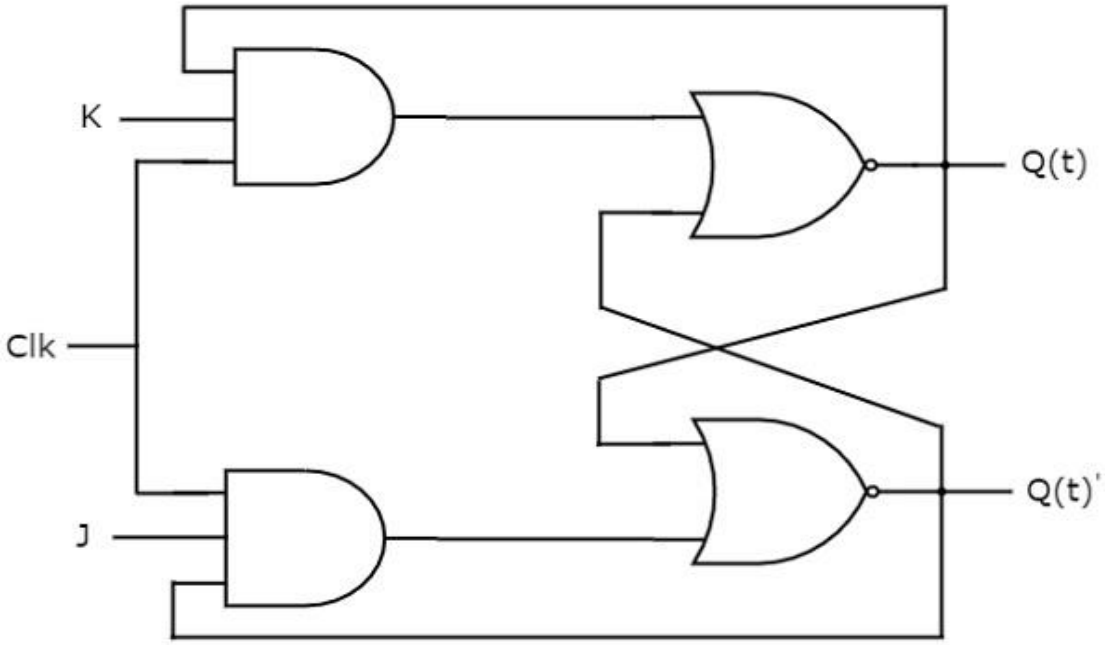
JK Flip-Flop

JK flip-flop is the modified version of SR flip-flop. It operates with only positive clock transitions or negative clock transitions.

Block Diagram of SR Flip Flop & Pin Diagram



The **circuit diagram** of JK flip-flop is shown in the following figure.



This circuit has two inputs J & K and two outputs Qt & Qt'. The operation of JK flip-flop is similar to SR flip-flop. Here, we considered the inputs of SR flip-flop as **S = J Qt'** and **R = KQ**t in order to utilize the modified SR flip-flop for 4 combinations of inputs.

The following table shows the **state table** of JK flip-flop.

J	**K**	**Q**t+1
0	0	Qt
0	1	0
1	0	1

1	1	Qt'

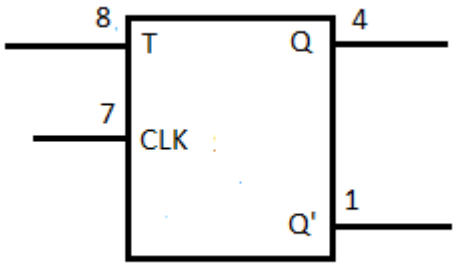
Here, Qt& Qt+1 are present state & next state respectively. So, JK flip-flop can be used for one of these four functions such as Hold, Reset, Set & Complement of present state based on the input conditions, when positive transition of clock signal is applied. The following table shows the **characteristic table** of JK flip-flop.

Present Inputs		**Present State**	**Next State**
J	**K**	**Q**t	**Q**t+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

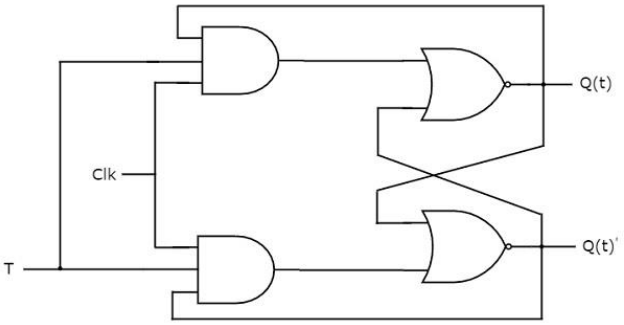
The simplified expression for next state Qt+1 = JQ(t)′+K′Q(t)Q(t+1) =JQ(t)′+K′Q(t)
RESULT: Verification of JK and D FLIP-FLOPS are done successfully.

T Flip-Flop

T flip-flop is the simplified version of JK flip-flop. It is obtained by connecting the same input 'T' to both inputs of JK flip-flop. It operates with only positive clock transitions or negative clock transitions.



The **circuit diagram** of T flip-flop is shown in the following figure.



This circuit has single input T and two outputs Qtt & Qtt'. The operation of T flip-flop is same as that of JK flip-flop. Here, we considered the inputs of JK flip-flop as **J = T** and **K = T** in order to utilize the modified JK flip-flop for 2 combinations of inputs. So, we eliminated the other two combinations of J & K, for which those two values are complement to each other in T flip-flop.

The following table shows the **state table** of T flip-flop.

T	**Q**t+1
0	Qt
1	Qt'

Here, Qt & Qt+1 are present state & next state respectively. So, T flip-flop can be used for one of these two functions such as Hold, & Complement of present state based on the input conditions, when positive transition of clock signal is applied. The following table shows the **characteristic table** of T flip-flop.

Inputs	**Present State**	**Next State**
T	**Q**t	**Q**t+1
0	0	0
0	1	1
1	0	1
1	1	0

From the above characteristic table, we can directly write the **next state equation** as

$$Q(t+1)=T'Q(t)+TQ(t)'Q(t+1)=T'Q(t)+TQ(t)'$$

$$\Rightarrow Q(t+1)=T\oplus Q(t)\Rightarrow Q(t+1)=T\oplus Q(t)$$

The output of T flip-flop always toggles for every positive transition of the clock signal, when input T remains at logic High 11. Hence, T flip-flop can be used in **counters**.

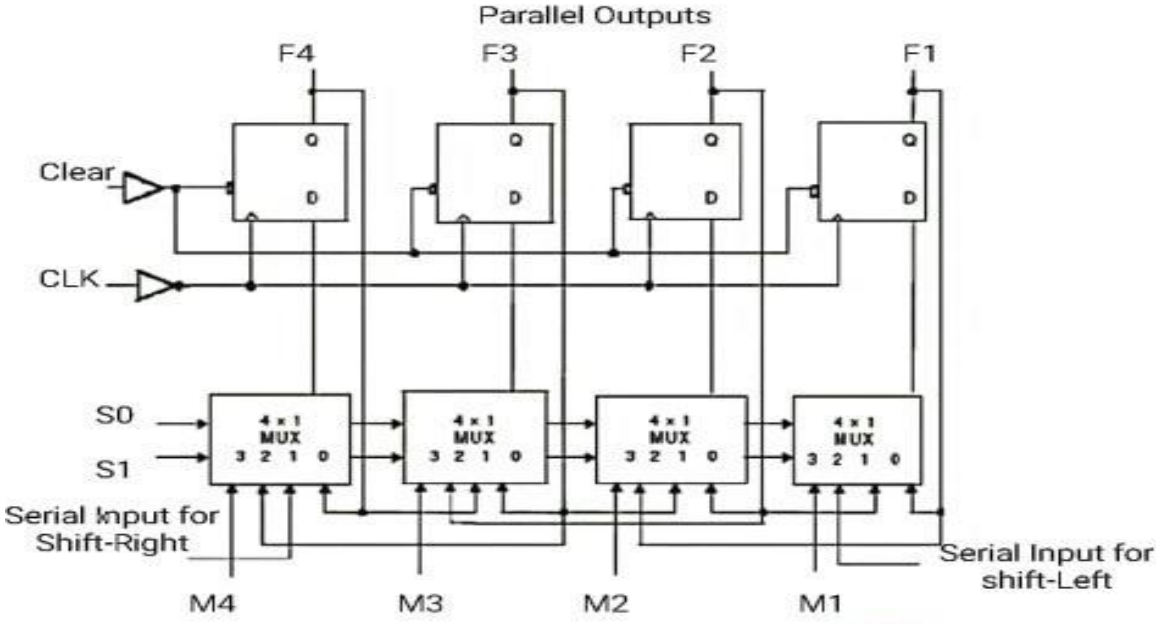
In this chapter, we implemented various flip-flops by providing the cross coupling between NOR gates. Similarly, you can implement these flip-flops by using NAND gates.

RESULT: Verification of SR and T FLIP-FLOPS are done successfully.

Experiment 9

AIM: Design a 4bit Universal Shift Register

THEORY:
The design of a 4-bit universal shift register using multiplexers and flip-flops is shown below.



Parallel Inputs

Universal Shift Register Design

- S0 and S1 are the selected pins that are used to select the mode of operation of this register. It may be shift left operation or shift right operation or parallel mode.
- Pin-0 of first 4×1 Mux is fed to the output pin of the first flip-flop. Observe the connections as shown in the figure.
- Pin-1 of the first 4X1 MUX is connected to serial input for shift right. In this mode, the register shifts the data towards the right.
- Similarly, pin-2 of 4X1 MUX is connected to the serial input for shift-left. In this mode, the universal shift register shifts the data towards the left.
- M1 is the parallel input data given to the pin-3 of the first 4×1 MUX to provide parallel mode operation and stores the data into the register.
- Similarly, remaining individual parallel input data bits are given to the pin-3 of related 4X1MUX to provide parallel loading.
- F1, F2, F3, and F4 are the parallel outputs of Flip-flops, which are associated with the 4×1 MUX.

Universal Shift Register Working:

- From the above figure, selected pins the mode of operation of the universal shift register. Serial input shifts the data towards the right and left and stores the data within the register.
- Clear pin and CLK pin are connected to the flip-flop.
- M0, M1, M2, M3 are the parallel inputs while F0, F1, F2, F3 are the parallel outputs of flip-flops
- When the input pin is active HIGH, then the universal shift register loads / retrieve the data in parallel. In this case, the input pin is directly connected to 4×1 MUX
- When the input pin (mode) is active LOW, then the universal shift register shifts the data. In this case, the input pin is connected to 4×1 MUX via NOT gate.
- When the input pin (mode) is connected to GND (Ground), then the universal shift register acts as a Bi-directional shift register.

- To perform the shift-right operation, the input pin is fed to the 1st AND gate of the 1st flip-flop via serial input for shit-right.
- To perform the shift-left operation, the input pin is fed to the 8th AND gate of the last flip-flop via input M.
- If the selected pins S0= 0 and S1 = 0, then this register doesn't operate in any mode. That means it will be in a Locked state or no change state even though the clock pulses are applied.
- If the selected pins S0 = 0 and S1 = 1, then this register transfers or shifts the data to left and stores the data.
- If the selected pins S0 = 1 and S1 = 0, then this register shifts the data to right and hence performs the shift-right operation.
- If the selected pins S0 = 1 and S1 = 1, then this register loads the data in parallel. Hence it performs the parallel loading operation and stores the data.

S0	**S1**	**Mode of Operation**
0	0	Locked state (No change)
0	1	Shift-right
1	0	Shift-left
1	1	Parallel Loading

From the above table, we can observe that this register operates in all modes with serial/parallel inputs using 4×1 multiplexers and flip-flops.

Advantages

The **advantages of a universal shift register** include the following.

- This register can perform 3 operations such as shift-left, shift-right, and parallel loading.
- Stores the data temporarily with in the register.
- It can perform serial to parallel, parallel to serial, parallel to parallel and serial to serial operations.
- It can perform input-output operations in both the modes serial and parallel.
- A Combination of the unidirectional shift register and bidirectional shift register gives the universe shift register.
- This register acts as an interface between one device to another device to transfer the data.

Applications

The **applications of a universal shift register** include the following.

- Used in micro-controllers for I/O expansion
- Used as a serial-to-serial converter
- Used as a parallel-to-parallel data converter
- Used as a serial-to-parallel data converter.
- Used in serial – to – serial data transfer
- Used in parallel data transfer.
- Used as a memory element in digital electronics like computers.
- Used in time delay applications
- Used as frequency counters, binary counters, and Digital clocks
- Used in data manipulation applications.

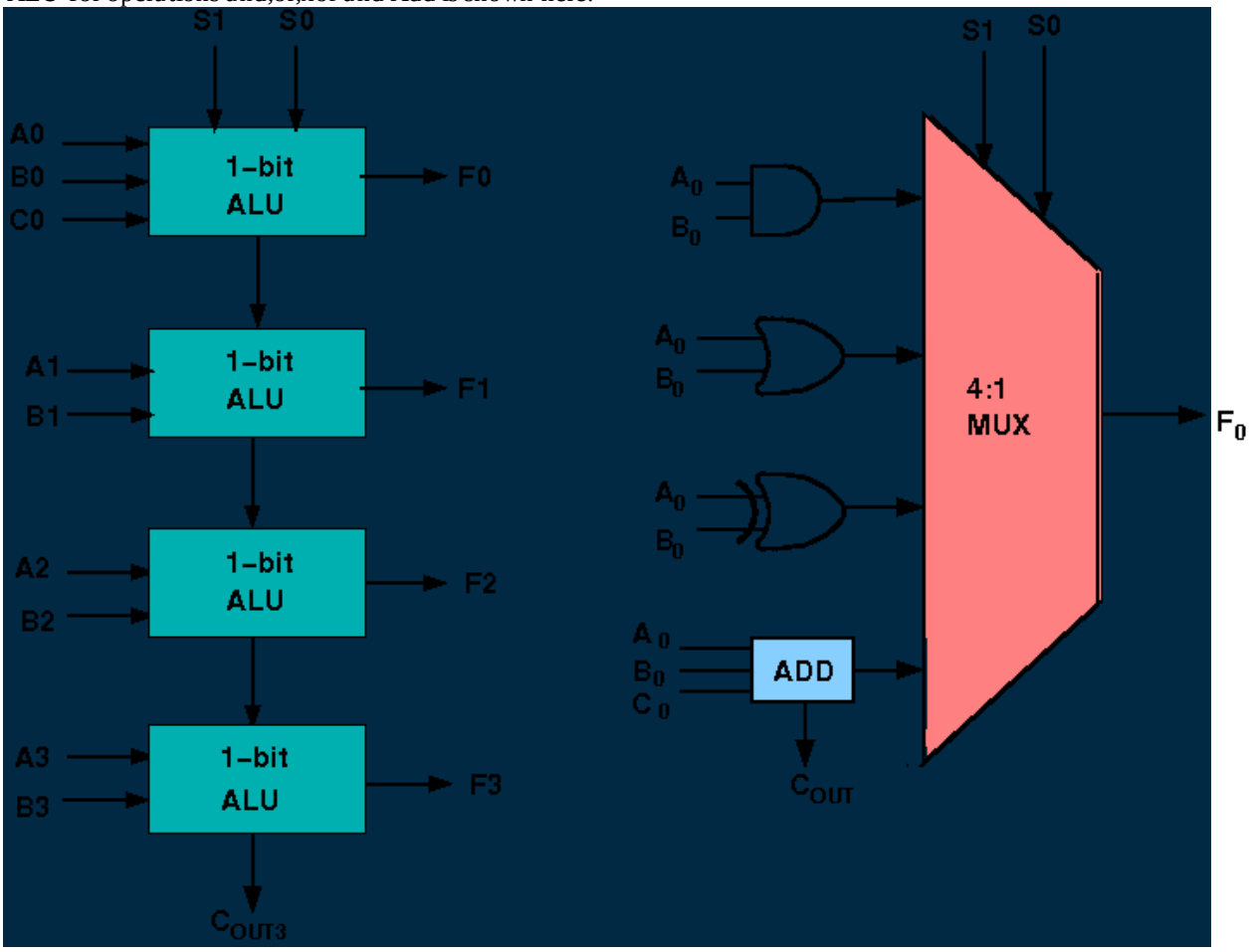
RESULT: Implementation of 4-bit Universal Shift Register is done successfully.

Experiment 10

AIM: Design of a 4-bit ARITHMETIC LOGIC UNIT.

THEORY:

Design of ALU:
ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction, division, multiplication and logical operations like and, or, xor, nand, nor etc. A simple block diagram of a 4-bit ALU for operations and,or,xor and Add is shown here:



The 4-bit ALU block is combined using 4 1-bit ALU block

Design Issues:

The circuit functionality of a 1-bit ALU is shown here, depending upon the control signal S_1 and S_0 th e circuit operates as follows:
for Control signal **S_1 = 0,** S_0 = 0, the output is **A And B**,
for Control signal **S_1 = 0,** S_0 = 1, the output is **A Or B**,

for Control signal S_1 **= 1,** S_0 = 0, the output is **A Xor B**,
for Control signal S_1 **= 1,** S_0 = 1, the output is **A Add B**.

The truth table for 16-bit ALU with capabilities similar to 74181 is shown here:

MODE SELECT				F_N FOR ACTIVE HIGH OPERANDS	
INPUTS				LOGIC	ARITHMETIC (NOTE 2)
S3	S2	S1	S0	(M = H)	(M = L) **(C_n=L)**
L	L	L	L	A'	A
L	L	L	H	A'+B'	A+B
L	L	H	L	A'B	A+B'
L	L	H	H	Logic 0	minus 1
L	H	L	L	(AB)'	A plus AB'
L	H	L	H	B'	(A + B) plus AB'
L	H	H	L	A ⊕ B	A minus B minus 1
L	H	H	H	AB'	AB minus 1
H	L	L	L	A'+B	A plus AB
H	L	L	H	(A ⊕ B)'	A plus B
H	L	H	L	B	(A + B') plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	Logic 1	A plus A (Note 1)
H	H	L	H	A+B'	(A + B) plus A

H	H	H	L	A+B	(A + B') plus A
H	H	H	H	A	A minus 1

Required functionality of ALU (inputs and outputs are active high)

The L denotes the logic low, and H denotes logic high.

RESULT: Implementation of ALU is done successfully.

Experiment 11

AIM: Design and implement half and full subtractor

THEORY:

Subtractor circuits take two binary numbers as input and subtract one binary number input from the other binary number input. Similar to adders, it gives out two outputs, difference and borrow (carry-in the case of Adder). There are two types of subtractors.

1. Half Subtractor
2. Full Subtractor

1) Half Subtractor

The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, A (minuend) and B (subtrahend) and two outputs Difference and Borrow. The logic symbol and truth table are shown below.

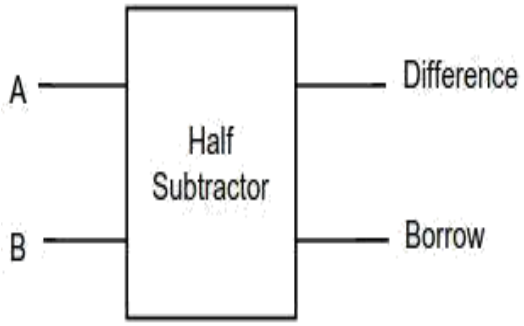


Figure-1: Logic Symbol of Half subtractor

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Figure-2: Truth Table of Half subtractor

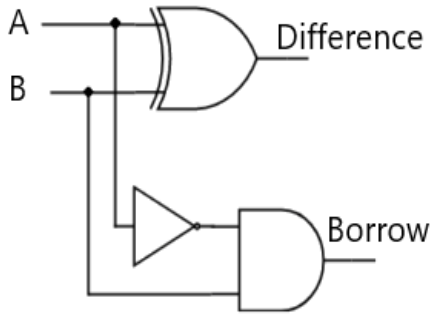


Figure-3: Circuit Diagram of Half subtractor

From the above truth table we can find the boolean expression.

$$\textbf{Difference} = A \oplus B$$
$$\textbf{Borrow} = A'\,B$$

From the equation we can draw the half-subtractor circuit as shown in the figure 3.

2) Full Subtractor

A full subtractor is a combinational circuit that performs subtraction involving three bits, namely A (minuend), B (subtrahend), and Bin (borrow-in) . It accepts three inputs: A (minuend), B (subtrahend) and a Bin (borrow bit) and it produces two outputs: D (difference) and Bout (borrow out). The logic symbol and truth table are shown below.*

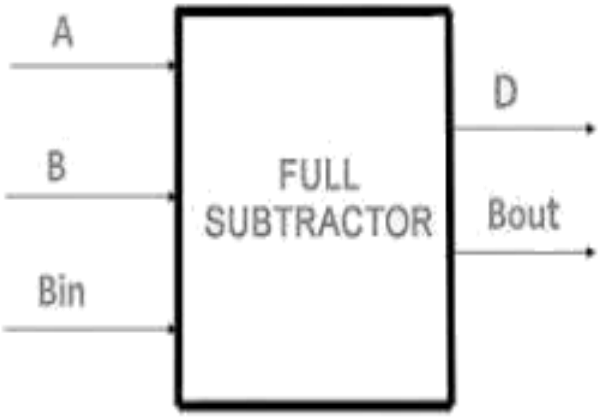


Figure-4: Logic Symbol of Full subtractor

A	B	B_{in}	D	B_{out}
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

Figure-5: Truth Table of Full subtractor

From the above truth table we can find the boolean expression.

$$D = A \oplus B \oplus Bin$$
$$Bout = A' Bin + A' B + B Bin$$

From the equation we can draw the Full-subtractor circuit as shown in the figure 6.

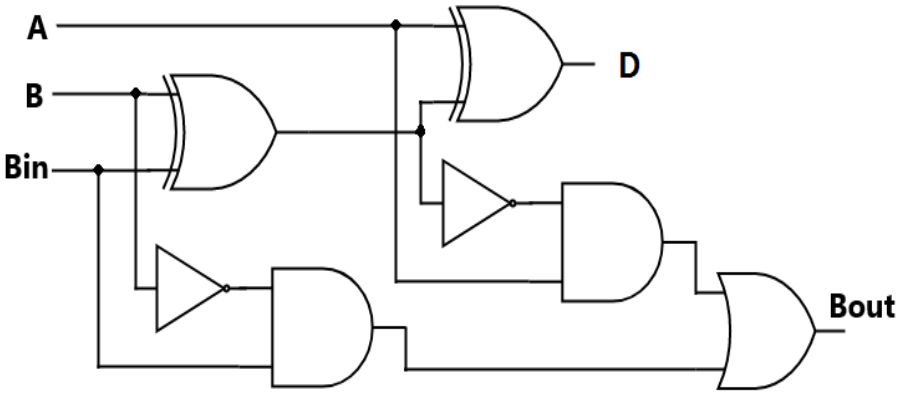


Figure-6: Circuit Diagram of Full subtractor

RESULT: Implementation of half and full subtractor is done successfully.

Experiment 12

AIM: Design and implement Binary to BCD and BCD to Binary converter.

THEORY:

Binary to BCD code converter

In BCD code, 0 to 9 numbers represent the equivalent binary numbers. For the numbers above 10, LSB of a decimal number is represented by its equivalent binary number and MSB of a decimal number is also represented by their equivalent binary numbers.

For example, the BCD code of 12 is represented as

The BCD code for 12 is 10010

The following truth table shows the conversion between the binary code input and the BCD code output. As you see from the table, the 4-bit binary number is converted into 5-bit BCD code. Decimal code is added in the table to understand the equivalence of Binary and BCD code.

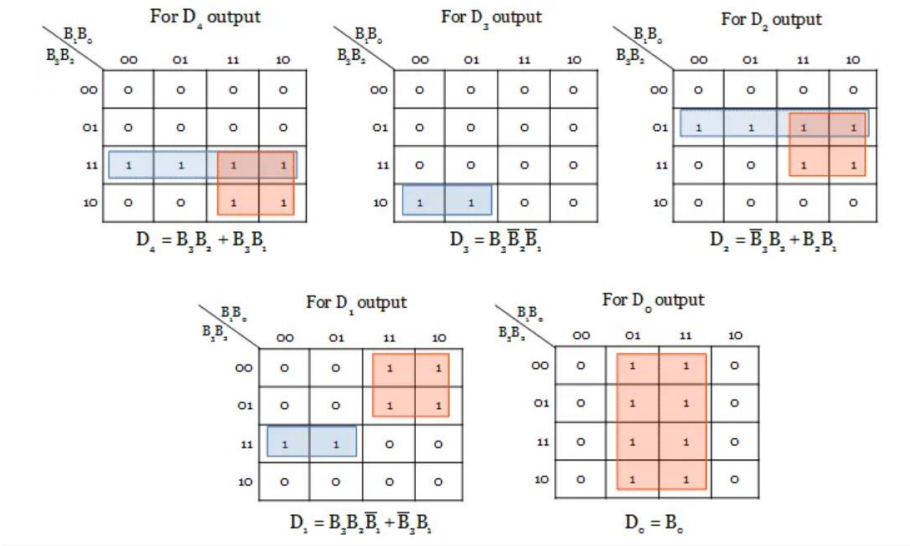
Decimal Number	Binary code (Input)				BCD code (Output)				
	B_3	B_2	B_1	B_0	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	0	1
12	1	1	0	0	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

The converter has 5 outputs D_0, D_1, D_2, D_3 and D_4. From the truth table, the minterms can be obtained for each output.

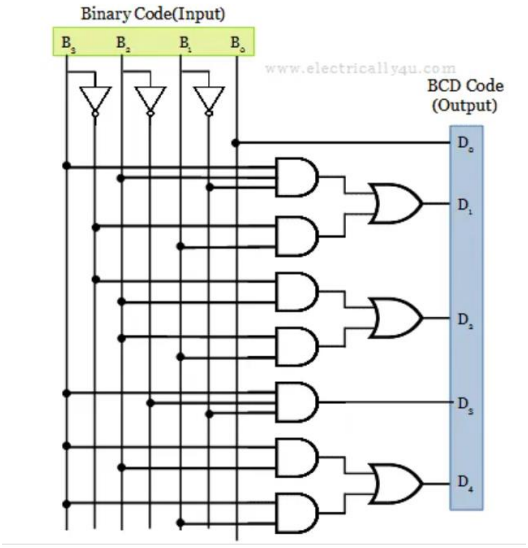
$D_4 = \sum m(10, 11, 12, 13, 14, 15)$, $D_3 = \sum m(8, 9)$, $D_2 = \sum m(4, 5, 6, 7, 14, 15)$, $D_1 = \sum m(2, 3, 6, 7, 12, 13)$, $D_0 = \sum m(1, 3, 5, 7, 9, 11, 13, 15)$

The minterms are plotted in the karnaugh map and the simplified boolean expressions are obtained.

Minimize a boolean function using K-map



The digital logic circuit for Binary to BCD code converter is designed from the simplified output expressions obtained from karnaugh map.



RESULT: Design and implementation of Binary to BCD is done successfully.