# EXPERIMENT NO.: - 1

# AIM: To study the basic logic gates which are used in digital circuits

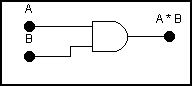
# THEORY:

Any Gate is a logic circuit with one output and one or more inputs. The output signal of any gate occurs only for certain combination of input signals. Different types of gates are used in the digital circuit like AND gate, OR gate, NOR gate, NAND gate, NOT gate are the basic logic gates. Some gates are the combination of above basic gates. Such gates can be prepared by using discrete components like diodes, transistors, and resisters but nowadays, different ICs are used to have different gates. A power supply of +5 V is used to give input. This supply is also used to drive ICs. When power supply to the input is 'ON' we can say that logic level is at '1' and when power supply to the input is 'OFF' the logic level is said to be at '0' level.

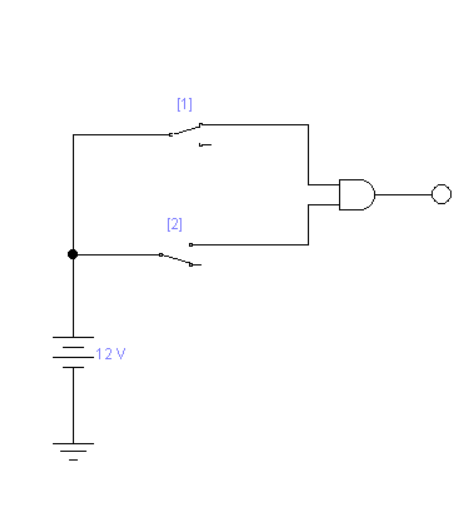
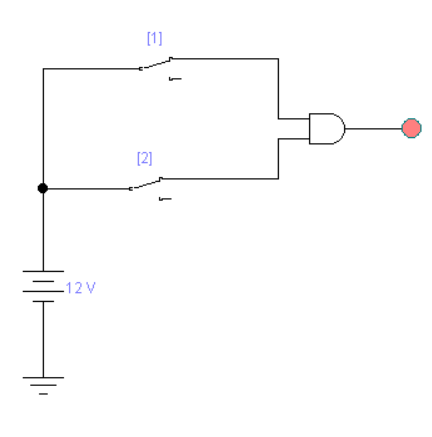
**AND GATE:**

AND gate is a gate which gives output (output at '1' level) only when all inputs are present (i.e. all inputs are at '1' level). Here IC 7408 is used which has 2-inputs AND gate which are four in numbers. Here only one gate is used. The truth table of AND gate is given in the table.

|  |  |  |
| --- | --- | --- |
| **A** | B | OUTPUT |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



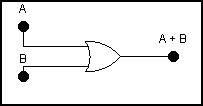
**Practical Figure:**



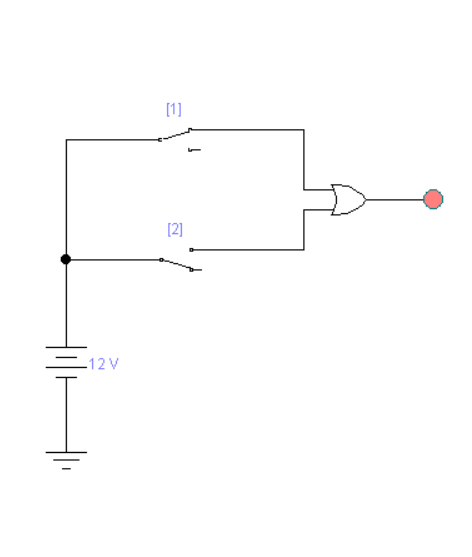
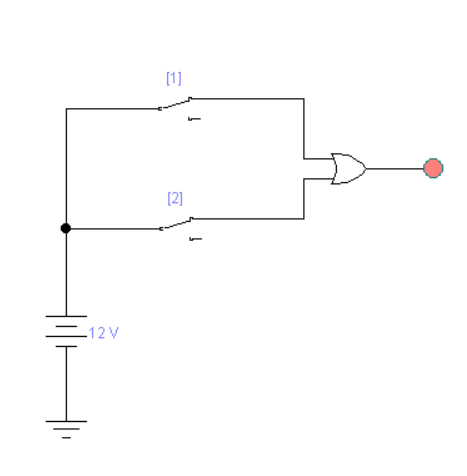
**OR GATE**:

OR gate is a gate which gives output (output at '1' level) when any ONE of the input is present (any ONE out of all input must be at '1' level). Here IC 7432 is used which has two input OR gates which are four in numbers. The truth table of OR gate is given in the table.

|  |  |  |
| --- | --- | --- |
| **A** | B | OUTPUT |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



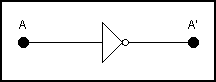
**Practical Figure:**

****

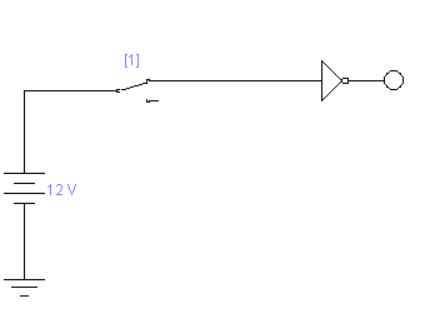
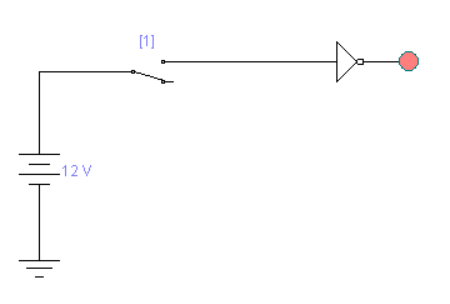
**NOT GATE:**

This is also known as INVERTER gate. This gate has one input and one output. All it does is invert the input signal i.e. if the input is at high level, the output will be at low level and vice versa. ICs 7404 or 7406 can be used to get NOT gates, which are six in numbers. The truth table of NOT gate is given in the table.

|  |  |
| --- | --- |
| A | OUTPUT |
| 0 | 1 |
| 1 | 0 |



**Practical Figure:**

****

**EQUIPMENTS REQUIRED:**

Trainer kit orBread Board, Power Supply and LEDs.

**COMPONENTS REQUIRED:**

**-**ICs **7408, 7432, 7404**.

**-**Hook up wires.

**PROCEDURE:**

1. Pull all switches at ‘0’ position.
2. Switch ON the power supply.
3. Test the truth table of different gates by changing the position of inputs (i.e. ‘1’ level means switch is ON & ‘0’ level means switch is OFF) and check the level of output (if LED glows it is at level ‘1’ and if LED doesn’t glow output is at level ‘0’).

**OBSERVATIONS:**

**We applied the values of truth tables in the circuit of different gates and get the output which shows its working and uses in other circuits.**

**CONCLUSION:**

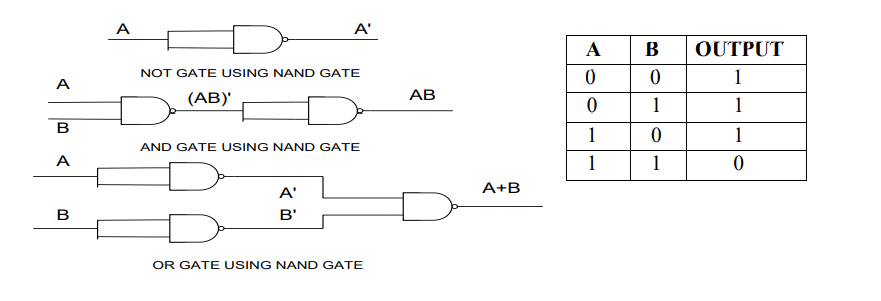
**We have studed the basic logic gates i.e AND gate , OR gate , NOT gate with there working and there truth tables.**

**EXPERIMENT NO.: - 2**

**AIM: To study NAND and NOR gates as Universal Gates.**

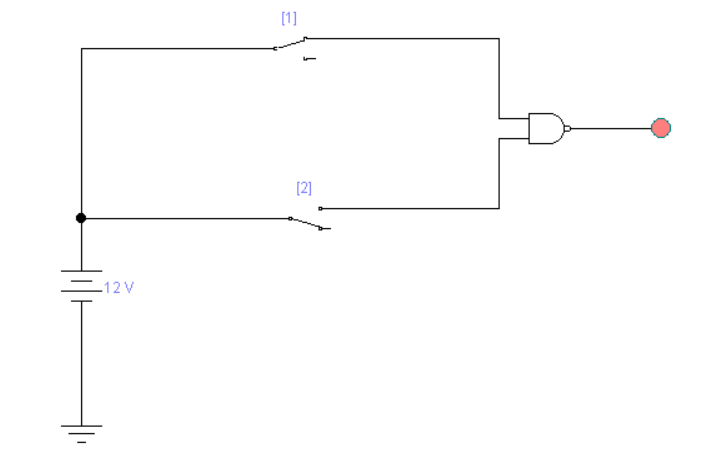
**THEORY:** Any Gate is a logic circuit with one output and one or more inputs. The output signal of any gate occurs only for certain combination of input signals. Different types of gates are used in the digital circuit. AND neither gate, OR gate, NOR gate, NAND gate, NOT gate are the basic logic gates. Some gates are the combination of above basic gates. Such gates can be prepared by using discrete components like diodes, transistors, and resisters but nowadays, different ICs are used to have different gates. A power supply of +5 V is used to give input. This supply is also used to drive ICs. When power supply to the input is 'ON' we can say that logic level is at '1' and when power supply to the input is 'OFF' the logic level is said to be at '0' level.

**NAND GATE:** It is a sequence of series combination of AND gate and NOT gate, known as NAND gate. The output of NAND gate is at ‘0’ level only when all the inputs are ‘1’ level. In rest of all the conditions of inputs the output will be at level ‘1’. IC 7400 can be used to get NAND gate. This IC has two input NAND gates, which are four in numbers. The truth table of NAND gate is given in the table

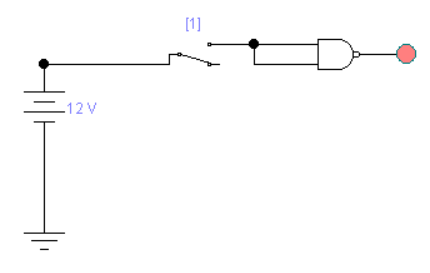


**Practical Figure:**

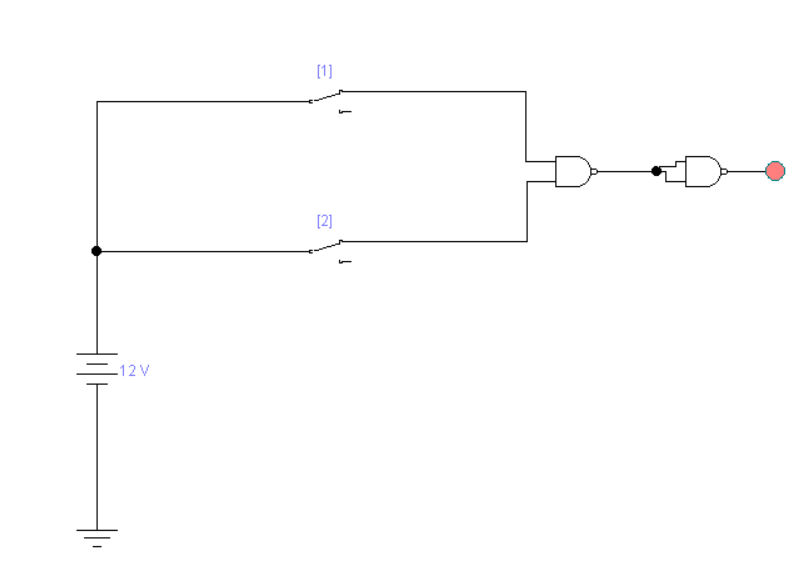
**NAND:**

****

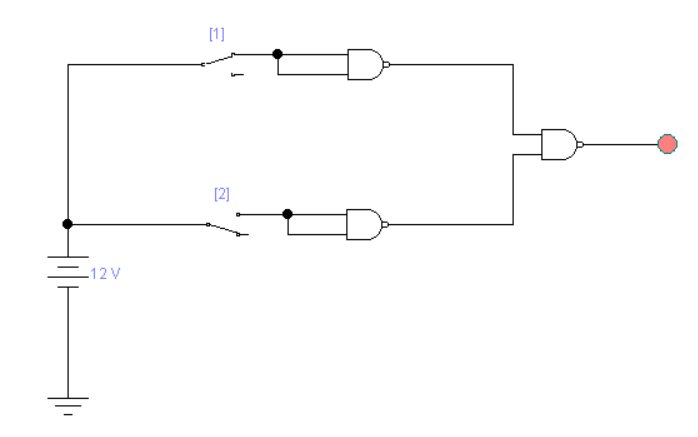
**NOT using NAND:**

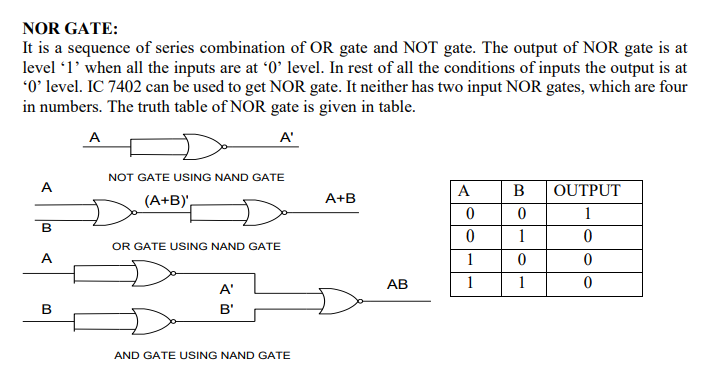
****

**AND using NAND:**

****

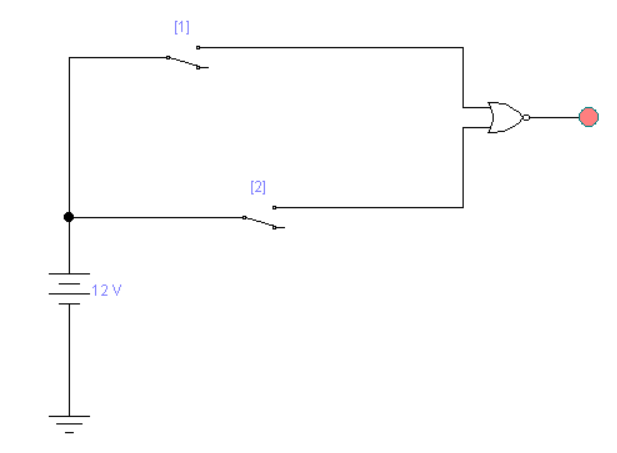
**OR using NAND:**

****

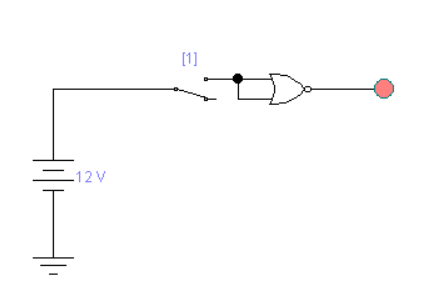
****

**Practical Figure:**

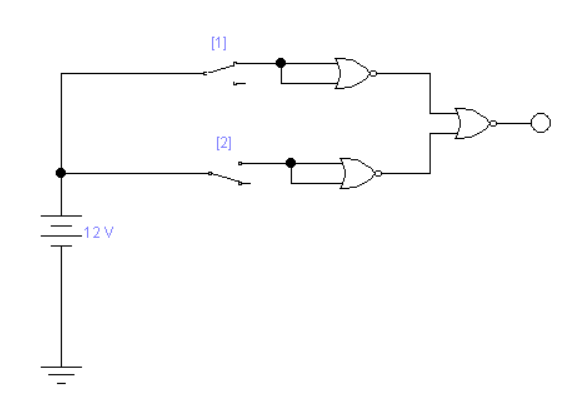
**NOR GATE:**

****

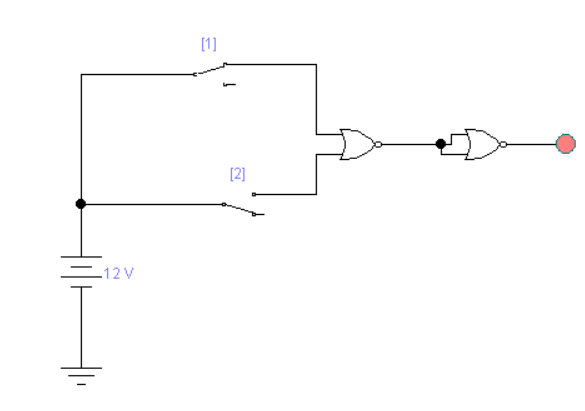
**NOT using NOR:**

****

**AND using NOR:**

****

**OR using NOR:**

****

**EQUIPMENTS REQUIRED:**

Trainer kit or Bread Board, Power Supply and LED’s.

**COMPONENTS REQUIRED:**

-ICs 7400, 7402 -Hook up wires

**PROCEDURE:**

1. Pull all switches at ‘0’ position.

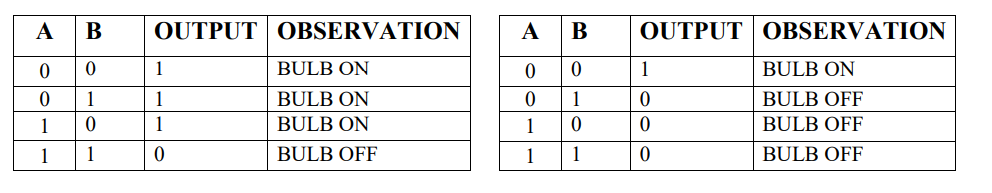
2. Switch ON the power supply.

3. Test the truth table of different gates by changing the position of inputs (i.e. ‘1’ level means

switch is ON & ‘0’ level means switch is OFF) and check the level of output (if LED glows it is at level ‘1’ and if LED doesn’t glow output is at level ‘0’).

**OBSERVATIONS:**

**NAND GATE: NOR GATE:**



**CONCLUSION:**

WE OBSERVE THAT NAND & NOR GATE ARE UNIVERSAL GATE.

WE CAN MAKE OTHER GATE USING THIS GATE LIKE AND GATE OR GATE NOR GATE

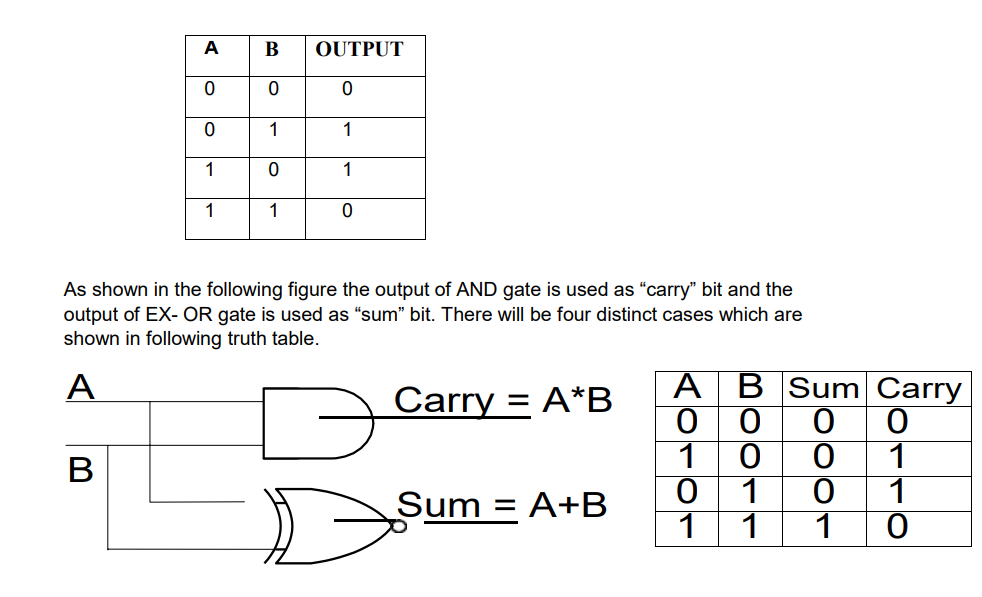
**EXPERIMENT NO.: - 3(A)**

**AIM: To study the arithmetic function of addition by building Half Adder circuit using logic gates.**

**THEORY:** This experiment gives how computer does the function of addition. The addition of two numbers in binary which does no have “carry” is done by half adder circuit and the addition of two numbers in binary which have “carry” is done by full adder circuit.

The half adder adds two-binary digit at a time. Fig. given below shows how to make half adder by using two logic gates namely AND gate and EX-OR gate. The truth table of EXOR gate is as below:

**EX-OR Gate:**

****

**EQUIPMENT**

**REQUIRED:**

-Trainer kit

**COMPONENTS REQUIRED:**

-ICs 7486, 7408. -Hook up wires.

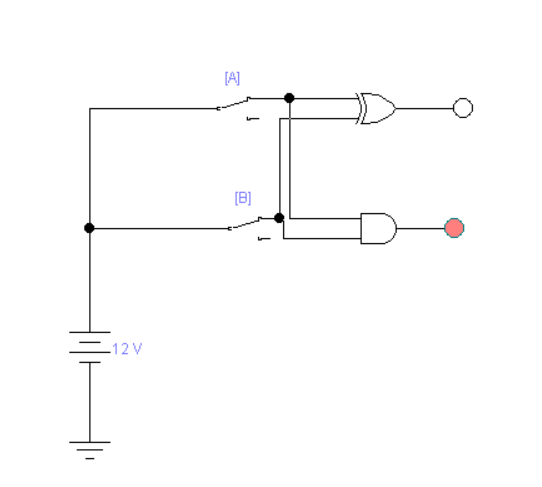
**PROCEDURE:**

1. Connect 5V DC power supply to input terminals.

2. Change the state of A & B (0 & 1) to get four states at output by changing the position of micro switch. Check the truth table of half adder.

( If LED glows it is High state and if it does not glow then it is Low state)

**Practical Figure::**



**OBSERVATIONS:**

The half adder adds two-binary digit at a time. Means we have 2 input at a time we add into addition 2 byte only add in at a time

**CONCLUSION:**

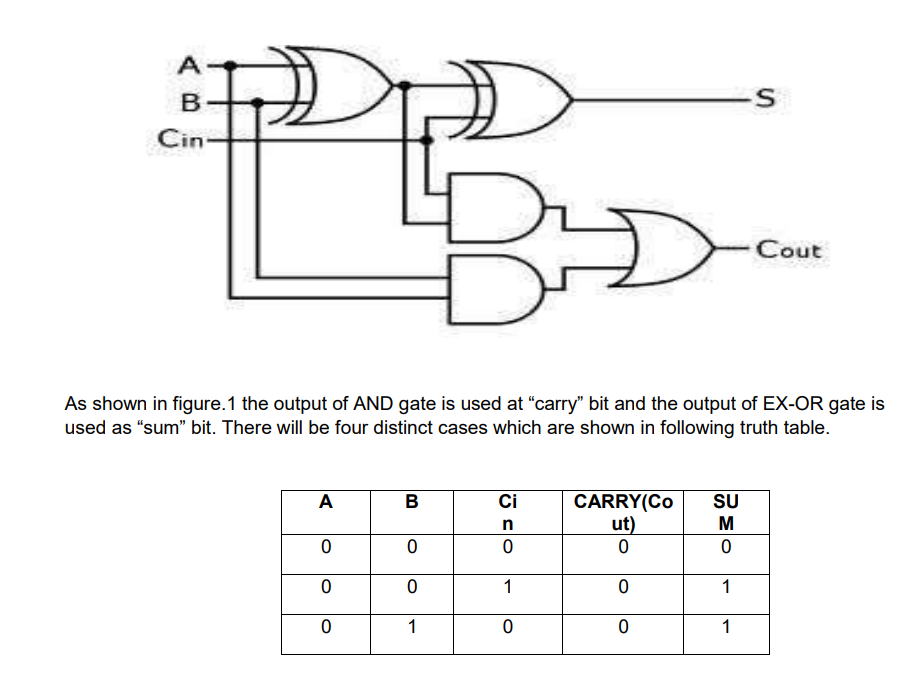
we study the arithmetic function of addition by building Half Adder circuit using logic gates.

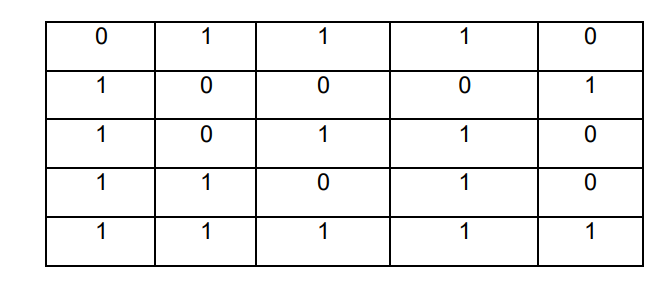
**EXPERIMENT NO.: - 3(B)**

**AIM: To study the arithmetic function of addition by building Full Adder circuit using logic gates.**

**THEORY:**

This experiment gives how computer does the function of addition. The addition of two numbers in binary which does no have “carry” is done by half adder circuit and the addition of two numbers in binary which have “carry” is done by full adder circuit. Sometimes in addition of two binary numbers you may have a carry from one column to the next. So in the next column we have to add three digits. This is not possible by half adder circuit. This is carried by a circuit as shown in fig.1 which consists of two half adder circuits and an ORgate. This is known as full adder circuit.





**EQUIPMENT REQUIRED: -**

-Trainer kit

**COMPONENTS REQUIRED:**

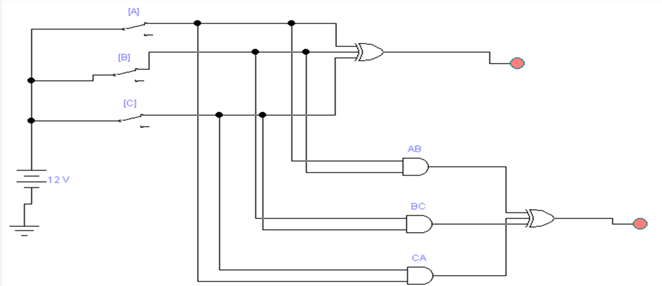
-ICs 7408, 7432, 7486. -Hook up wires.

**PROCEDURE:**

1. Connect 5V DC power supply to input terminals.

2. Change the state of A & B (0 & 1) to get four states at output by changing the position of micro switch. Check the truth table of half adder. ( If LED glows it is High state and if it does not glow then it is Low state)

**Practical Figure::**

****

**OBSERVATIONS:**

The full adder adds two-binary digit at a time. Means we have 3 input at a time we add into addition 3 byte where 2 as input and 1 byte as carry input at a time in addition.

**CONCLUSION:**

we study the arithmetic function of addition by building Full Adder circuit using logic gates.

# EXPERIMENT NO.: - 4(A)

**AIM: To Design Half-subtractor.**

**THEORY:**

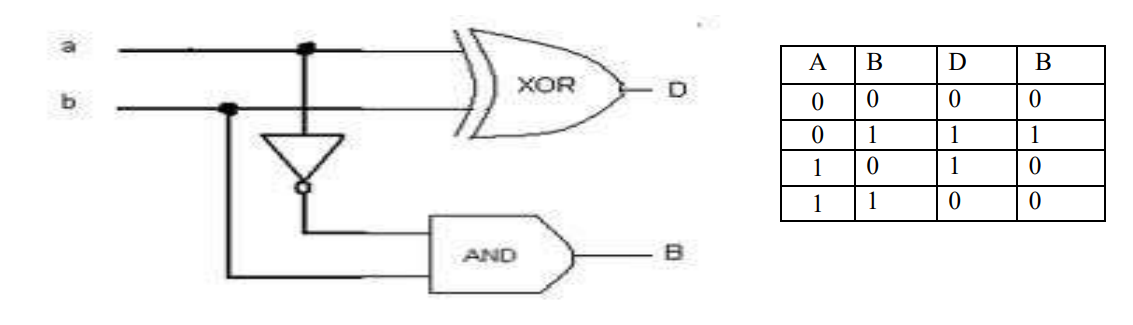
A combination circuit that performs the arithmetic subtraction of two bits is called half subtractor. In figure we assigns two symbols a and b are two input variables and D (difference) and B (borrow) are two output variables. The truth table for half subtractor is shown in figure.

D = a’b + ab’

= a ⊕ b

B = ab’

The half Subtractor consists of an X-OR gate and AND gate.

** EQUIPMENTS REQUIRED:**

-Trainer kit

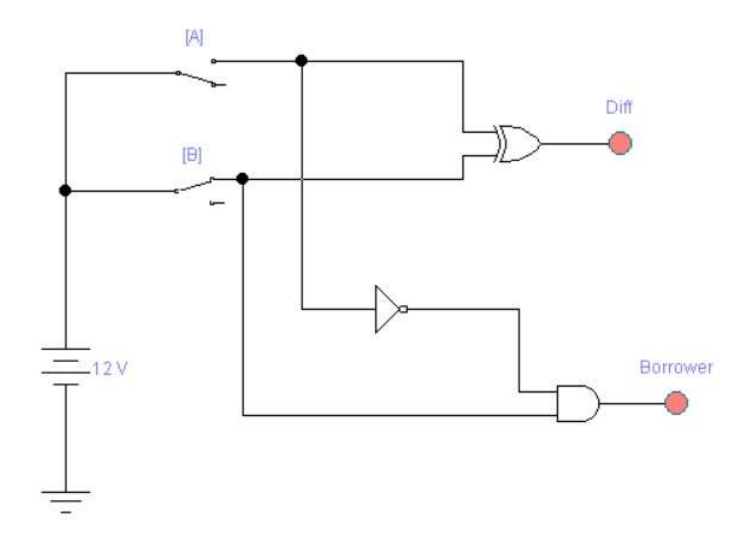
**COMPONENTS REQUIRED:**

* Connecting wires
* IC 7408 (AND gate)
* IC 7486 (X-OR gate)
* IC 7432 (OR gate)

**PROCEDURE:**

1. Join the circuit as per circuit diagram.
2. Verify the truth-table.

**PRACTICAL FIGURE :**

****

**OBSERVATION:**

**Difference :**

**0 1**

**0**

|  |  |
| --- | --- |
|  | **1** |
| **1** |  |

**1**

**D= A’B+AB’**

**Borrow:**

**0 1**

**0**

|  |  |
| --- | --- |
|  | **1** |
|  |  |

**1**

**CARRY=A’B**

**CONCLUSION:**

we study the arithmetic function of Subtraction by building Half Subtractor circuit using logic gates.

# EXPERIMENT NO.: - 4(B)

**AIM: To design Full-subtractor.**

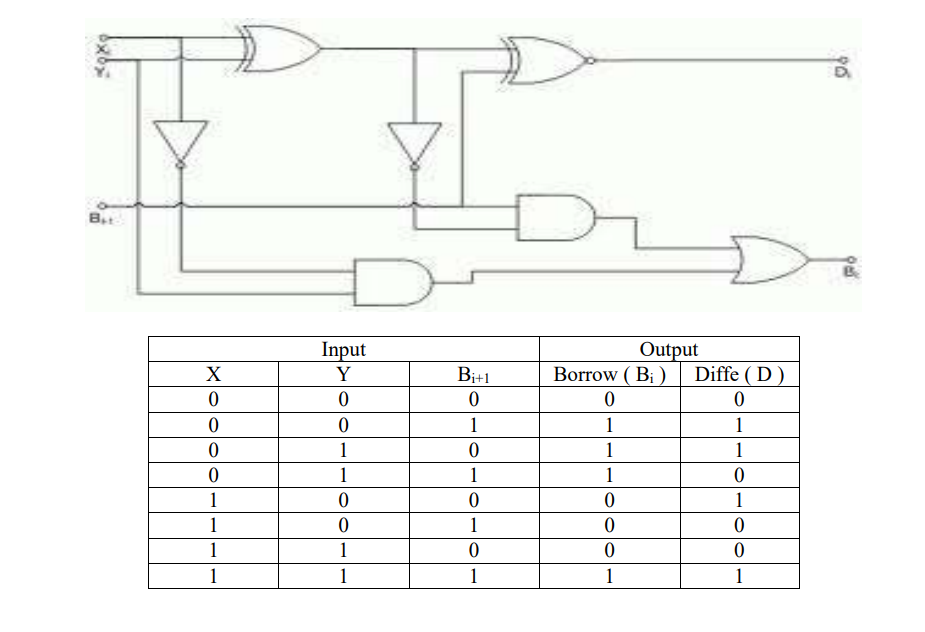
**THEORY:**

A combination circuit that performs the arithmetic subtraction of three bits is called full subtractor. In figure we assigns three symbols X, Y and Z are three input variables and D (difference) and B(borrow) are two output variables. The truth table for full substractor is shown in figure.

D = x ⊕ y ⊕ z

B = x’y + x’z + yz

The full subtractor consists of two half-subtractor and one OR gate.

****

**DIFFERENCE**

**0**

**00 01 11 10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 |  | 1 |
| 1 |  | 1 |  |

1

**SUM** = A’B’C+A’BC’+AB’C’+ABC

= A’B’C+ABC+A’BC’+AB’C’

= C(A’B’+AB)+C’(A’B+AB’)

= C(A EXLCUSIVE NOR B)+C’(A EXLCUSIVE OR B) (A EXLCUSIVE OR B=X)

= CX’+CX

= (C EXLCUSIVE OR X)

= (A EXLCUSIVE OR B EXLCUSIVE OR C)

**BORROW**

**00 01 11 10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 1 | 1 |
|  |  | 1 |  |

**0**

**1**

**BORROW** = BC+A’(B’C+BC)+A’(BC+BC’)

= BC+A’C(B’+B)+A’B(C+C’)

= BC+A’B+A’C

**EQUIPMENTS REQUIRED:**

-Trainer kit COMPONENTS

**REQUIRED:**

• Connecting wires

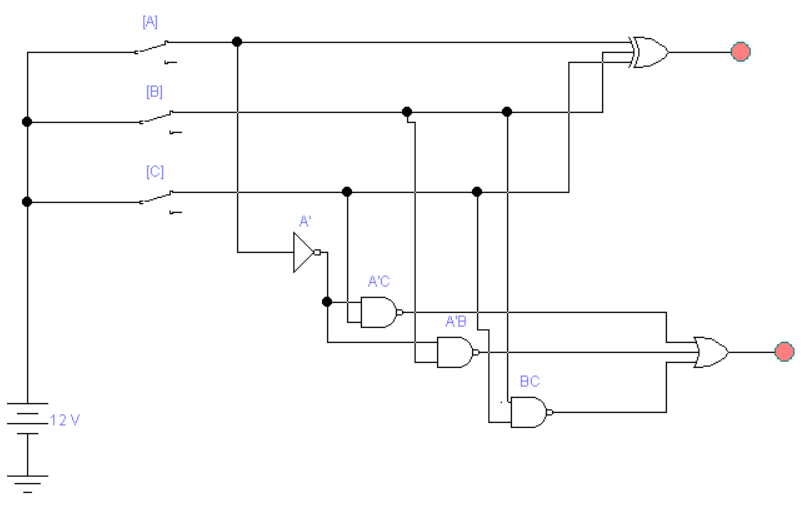
• IC 7408 (AND gate)

• IC 7486 (X-OR gate)

• IC 7432 (OR gate)

**PROCEDURE:** 1.Join the circuit as per circuit diagram. 2.Verify the truth table for full-subtractor.

**Practical Figure:**

****

**CONCLUSION:**

we study the arithmetic function of Subtraction by building Full Subtractor circuit using logic gates.

# EXPERIMENT NO.: - 5

**AIM *:* To study the operation of multiplexer circuit.**

**THEORY:**

A multiplexer is a MSI logic circuit capable of selecting single input bit from a number of different inputs and routing the selected bit to a single output. The bit selected is determined by the appropriate input address lines. For instant a multiplexer having three data select lines A, B, C is capable of selecting one of the eight possible input bits (i.e. D0 to D7). Here IC 74151 is used which is eight bit multiplexer. The connection diagram and truth table are shown in respective figures.

|  |
| --- |
|  |

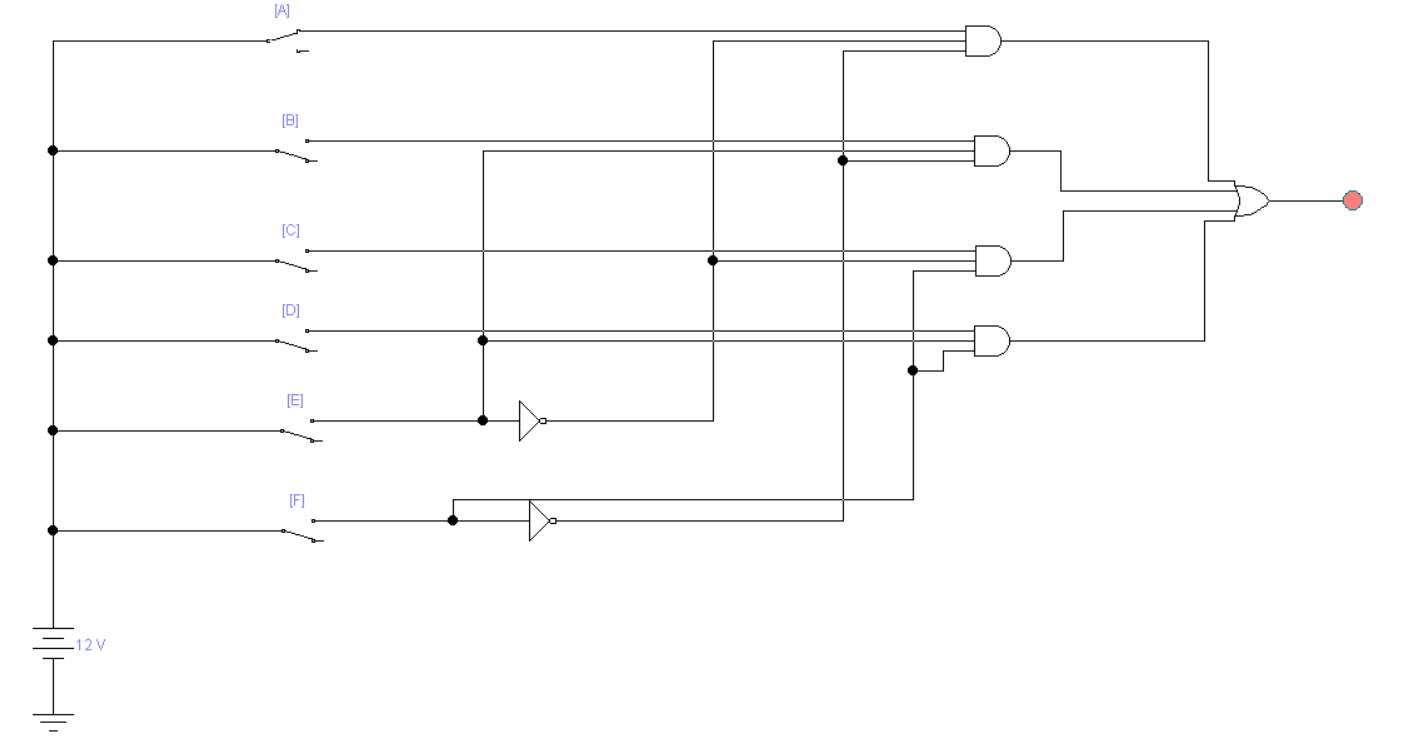


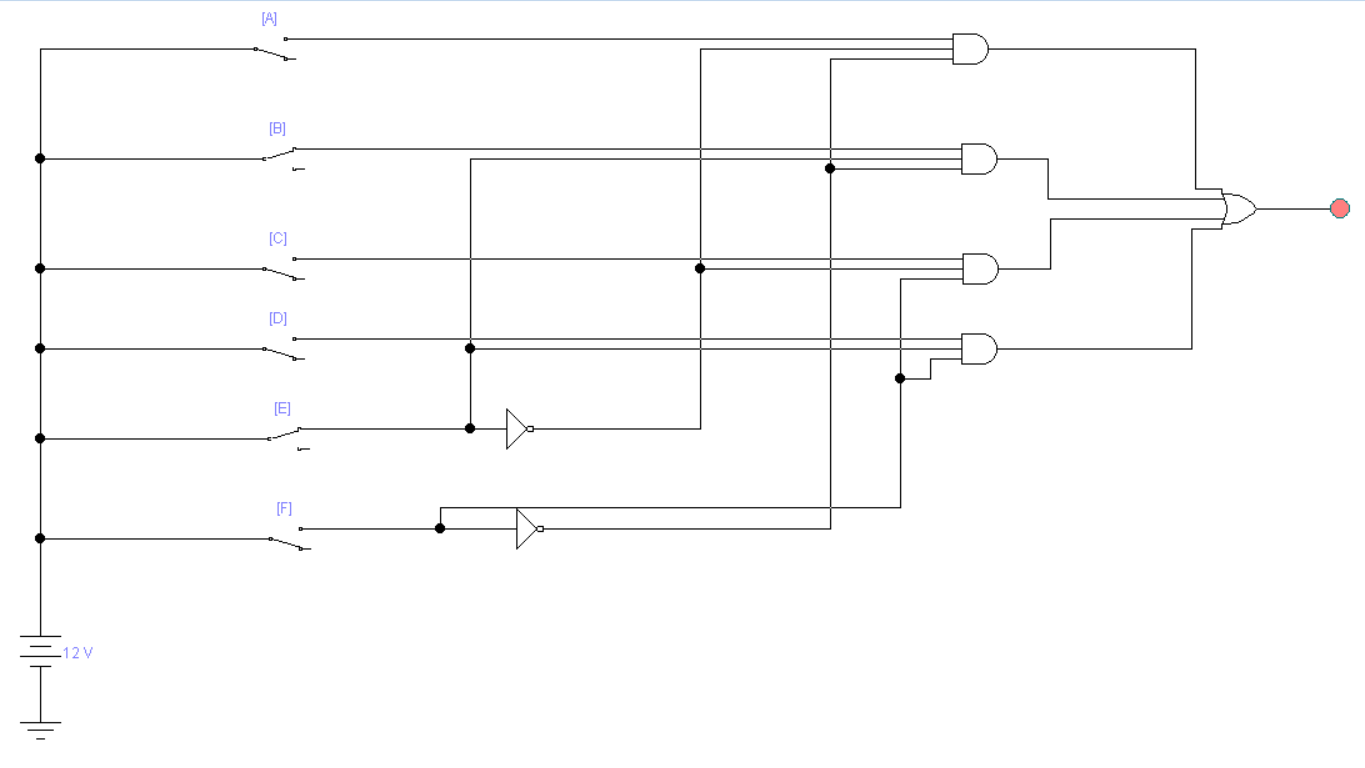
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INPUTS** | | | | **OUTPUTS** |  |
| **SELECT** | | | **STROBE**  **G’** | **Y** | **W** |
| **C** | **B** | **A** |
| X | X | X | H | L | H |
| L | L | L | L | D0 | D0’ |
| L | L | H | L | D1 | D1’ |
| L | H | L | L | D2 | D2’ |
| L | H | H | L | D3 | D3’ |
| H | L | L | L | D4 | D4’ |
| H | L | H | L | D5 | D5’ |
| H | H | L | L | D6 | D6’ |
| H | H | H | L | D7 | D7’ |

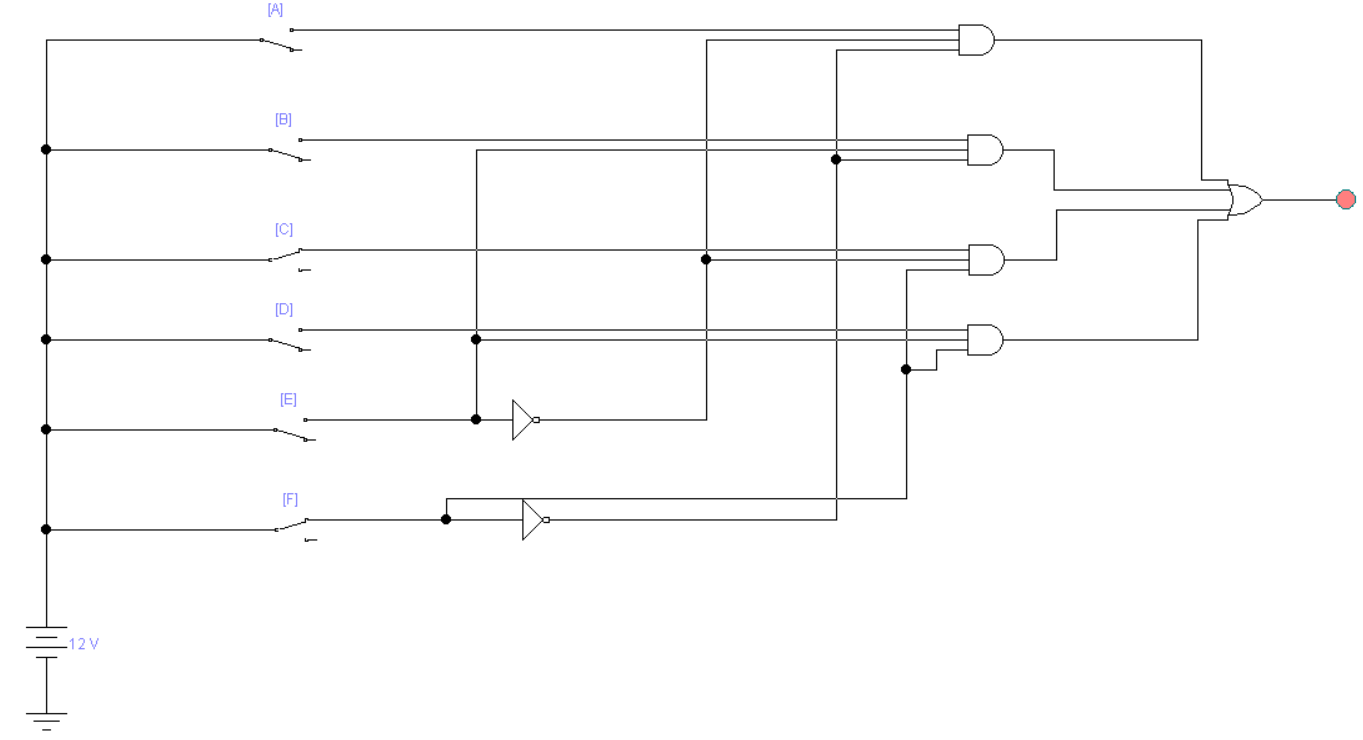
8-Multiplexer circuit

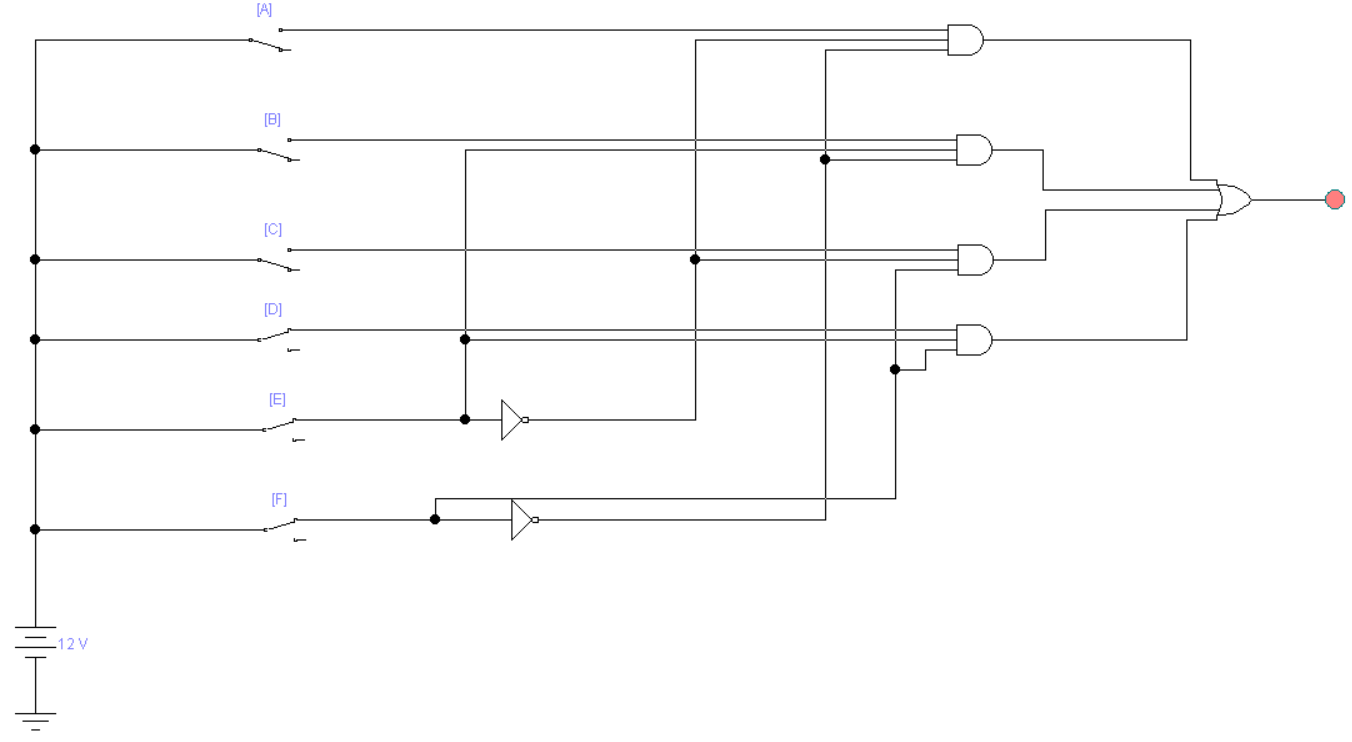
**Practical figure:**

4-Multiplexer circuit

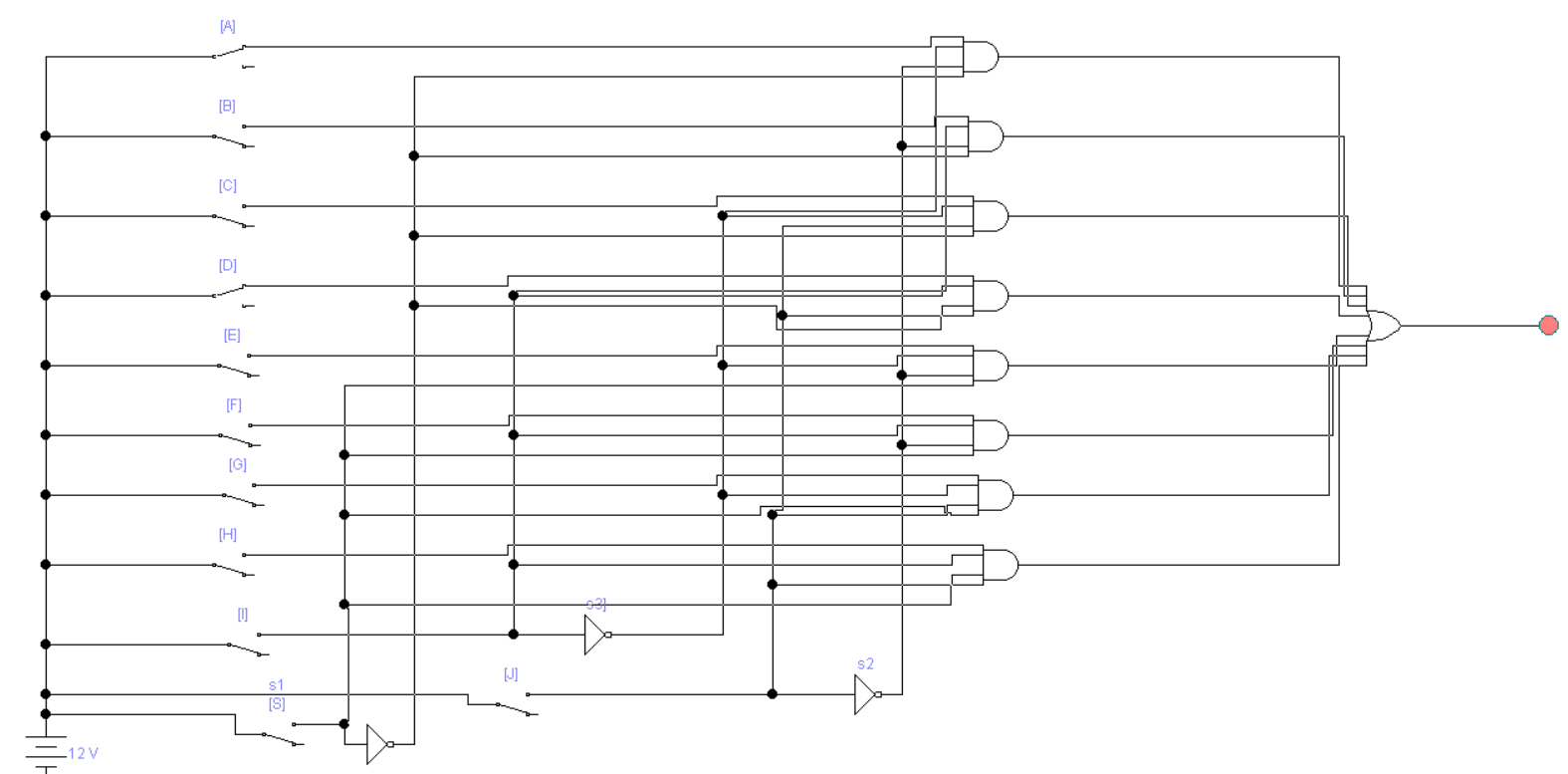








**For 3 Selection Lines**

****

**EQUIPMENT REQUIRED:**

-Trainer kit

**COMPONENTS REQUIRED:**

-ICs 74151.

-Hook up wires

PROCEDURE

1. SwitchON the power supply.
2. Set inputs (D0 to D7) at high level or at low level as desired.
3. Set STR to low level.
4. Set three data select lines A, B, C at any value as per truth table ( say A = Low, B= High , C= Low i.e. 010 i.e. decimal 2 as per column no. 4)
5. Multiplexer will select D2 input at Y output terminal.
6. Check the truth table for different value of data select lines A, B, C mentioned in the truth table and checks the selected input Y appears at output.
7. If the level of strobe is changed to high level at any point, Y output will be only at low level irrespective of any position of A, B, C.

**OBSERVATIONS:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INPUTS** | | | | **OUTPUTS** |  |
| **SELECT** | | | **STROBE**  **G’** | **Y** | **W** |
| **C** | **B** | **A** |
| X | X | X | H | L | H |
| L | L | L | L | D0 | D0’ |
| L | L | H | L | D1 | D1’ |
| L | H | L | L | D2 | D2’ |
| L | H | H | L | D3 | D3’ |
| H | L | L | L | D4 | D4’ |
| H | L | H | L | D5 | D5’ |
| H | H | L | L | D6 | D6’ |
| H | H | H | L | D7 | D7’ |

**CONCLUSION:**

we studied about the multiplexer with two and three selection Lines.

# EXPERIMENT NO.: - 6

**AIM: To study the operation of de-multiplexer circuit.**

**THEORY:**

A Demultiplexer is a MSI logic circuit capable of routing data from a single source to one of a number of possible destinations the data bits are applied at the enable inputs and they appear at an output specified by the address inputs A0, A1,A2.

|  |
| --- |
| **Untitled4** |

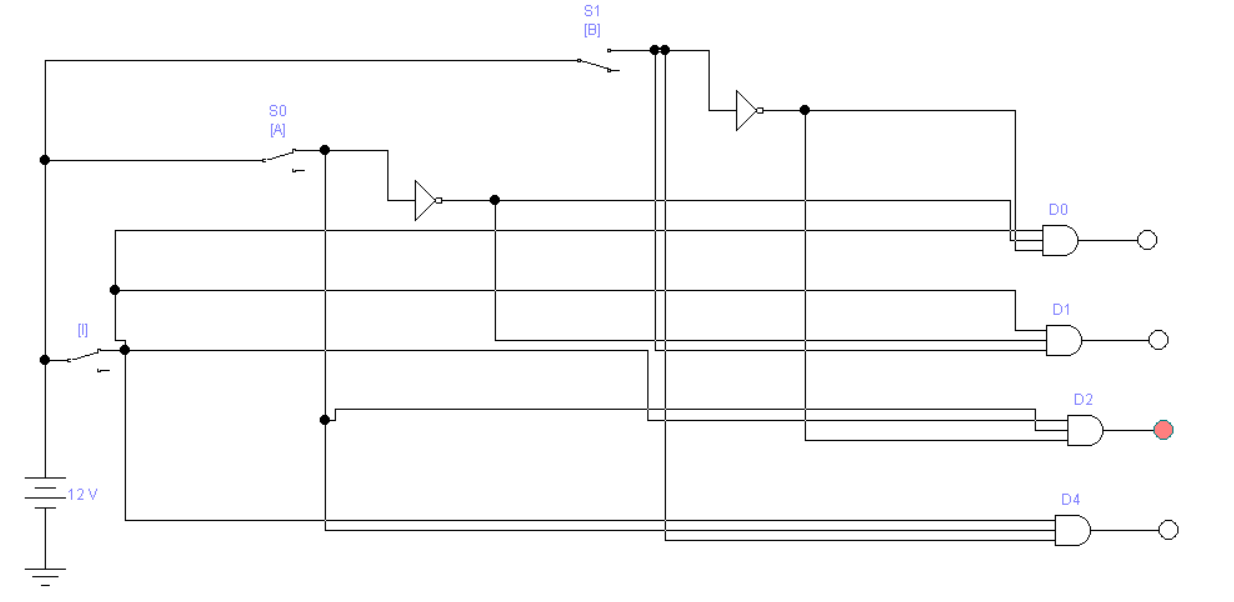
Here IC 74138 is used which decodes one of eight lines based upon the conditions at the three binary select inputs and the three enable inputs. The connection diagram and function table are shown below.



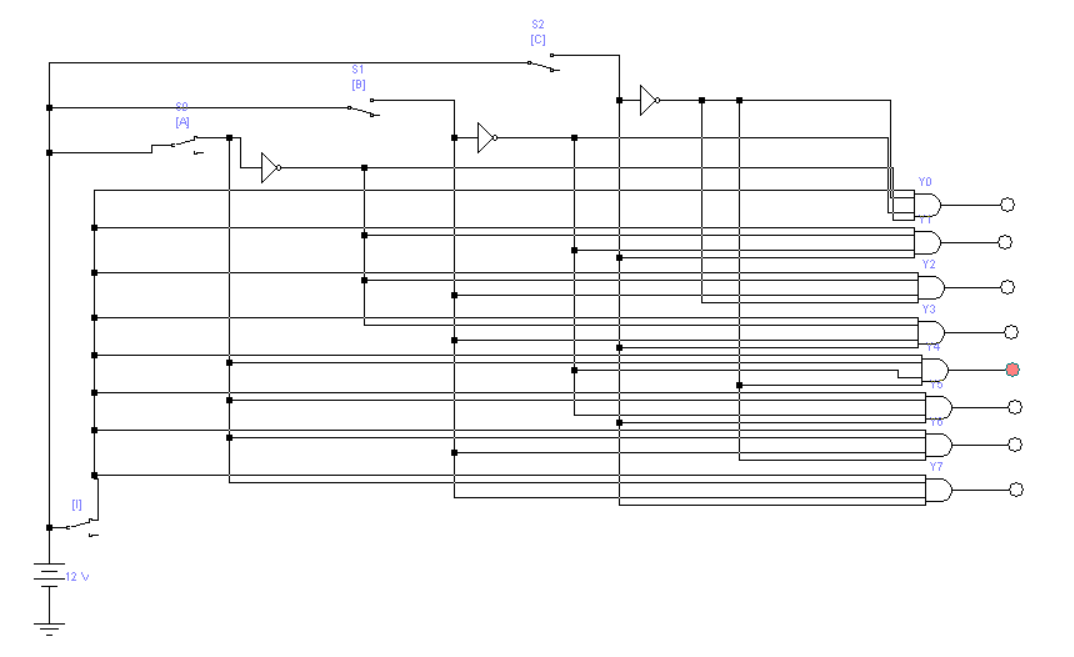
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INPUTS | | | | | OUTPUTS | | | | | | | |
| ENABLE | | SELECT | | |
| G1 | G2 | C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | L | H | H | H | H | H | H | H |
| H | L | L | L | H | H | L | H | H | H | H | H | H |
| H | L | L | H | L | H | H | L | H | H | H | H | H |
| H | L | L | H | H | H | H | H | L | H | H | H | H |
| H | L | H | L | L | H | H | H | H | L | H | H | H |
| H | L | H | L | H | H | H | H | H | H | L | H | H |
| H | L | H | H | L | H | H | H | H | H | H | L | H |
| H | L | H | H | H | H | H | H | H | H | H | H | L |

Practical **Figure:**

1. **For 2 Selection Lines**



1. **For 3 Selection Lines**



**EQUIPMENT REQUIRED:**

-Trainer kit

**COMPONENTS REQUIRED:**

**-**ICs 74138.

**-**Hook up wires.

**PROCEDURE:**

1. SwitchON the power supply.
2. Set G1, G2A and G2B at low level.
3. Set inputs C, B, A at low level.
4. All output must be at high level ( i.e. all LED should glow - condition no.2)
5. Set G1, G2A and G2B at high level & inputs C, B, A as it is i.e. at low level.
6. All output must be at high level (i.e. all LED should glow – condition no.1)
7. Set G1 at high level and G2A and G2B at low level.
8. Set C, B, A at any different levels and check the outputs as per truth table, for example, if you adjust C, B, A to 0 1 0 , the output Y2 only will be low, others will be at high level – condition no.5.

**OBSERVATION:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INPUTS | | | | | OUTPUTS | | | | | | | |
| ENABLE | | SELECT | | |
| G1 | G2 | C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | L | H | H | H | H | H | H | H |
| H | L | L | L | H | H | L | H | H | H | H | H | H |
| H | L | L | H | L | H | H | L | H | H | H | H | H |
| H | L | L | H | H | H | H | H | L | H | H | H | H |
| H | L | H | L | L | H | H | H | H | L | H | H | H |
| H | L | H | L | H | H | H | H | H | H | L | H | H |
| H | L | H | H | L | H | H | H | H | H | H | L | H |
| H | L | H | H | H | H | H | H | H | H | H | H | L |

**CONCLUSION:**

we studied about the de-multiplexer with two and three selection Lines.

# EXPERIMENT NO.: - 7 A

**AIM*:***To study Binary to Gray code converter.

**THEORY:**

Gray Code system is a binary number system in which every successive pair of numbers differs in only one bit. It is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next. For example, the states of a system may change from 3(011) to 4(100) as- 011 — 001 — 101 — 100. Therefore, there is a high chance of a wrong state being read while the system changes from the initial state to the final state. This could have serious consequences for the machine using the information. The Gray code eliminates this problem since only one bit changes its value during any transition between two numbers.

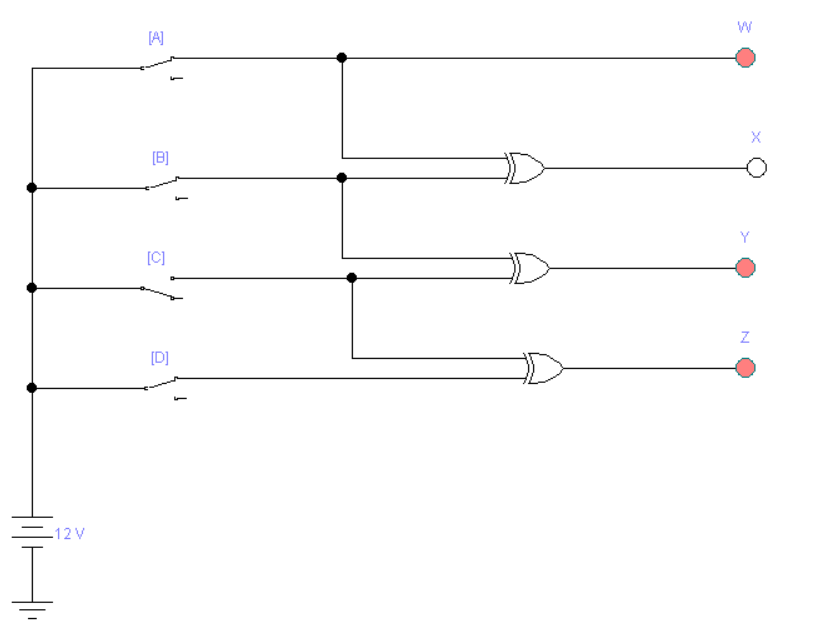
**TRUTH TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Binary | | | | Gray Code | | | |
| A | B | C | D | W | X | Y | Z |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR W**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 |   **W=A** | **K-MAP FOR X**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 |   **X=A’B+AB’** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR Y**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 |   **Y=BC’+B’C** | **K-MAP FOR Z**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 1 |   **Z=C’D+CD’** |

**Practical Figure :**



**CONCLUSION:**

We studied about binary to grey code converter.

# EXPERIMENT NO.: - 7 B

**AIM*:* To study Gray code to Binary converter.**

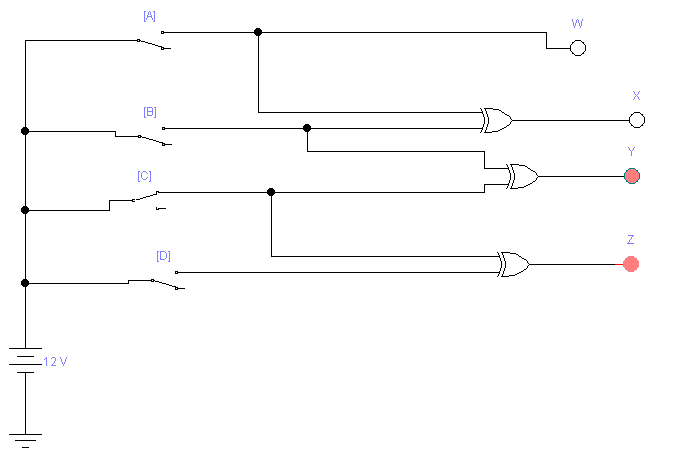
**TRUTH TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Gray Code | | | | Binary | | | |
| W | X | Y | Z | A | B | C | D |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR W**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 |   **A=W** | **K-MAP FOR X**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 |   **X=A’B+AB’** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR Y**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 |   **Y=A XOR BXORC** | **K-MAP FOR Z**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 0 | | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 0 |   **Z=A XOR B XOR C XOR D** |

**CIRCUIT:**

****

**CONCLUSION:**

We have studied Gray code to Binary converter

# EXPERIMENT NO.: - 8A

**AIM*:***To study BCD to Excess-3 converter.

**THEORY:**

Excess-3 binary code is a unweighted self-complementary BCD code. Self-Complementary property means that the 1’s complement of an excess-3 number is the excess-3 code of the 9’s complement of the corresponding decimal number. This property is useful since a decimal number can be nines’ complemented (for subtraction) as easily as a binary number can be ones’ complemented; just by inverting all bits.

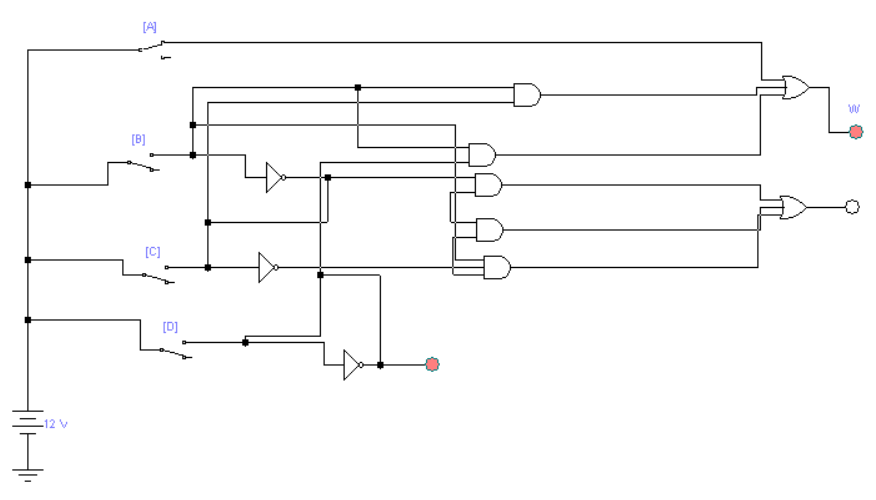
**TRUTH TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BCD | | | | Excess-3 | | | |
| A | B | C | D | W | X | Y | Z |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | X | X | X | X |
| 1 | 0 | 1 | 1 | X | X | X | X |
| 1 | 1 | 0 | 0 | X | X | X | X |
| 1 | 1 | 0 | 1 | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR W**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | | X | X | X | X | | 1 | 1 | X | X |   **W= A+B(C+D)** | **K-MAP FOR X**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | X | X | X | X | | 0 | 1 | X | X |   **X= B’(C+D)+B(C+D)’** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR Y**   |  |  |  |  | | --- | --- | --- | --- | | 1 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | | X | X | X | X | | 1 | 0 | X | X |   **Y= CD+(C+D)’** | **K-MAP FOR Z**   |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | | X | X | X | X | | 1 | 1 | X | X |   **Z= D’** |

**CIRCUIT:**



**CONCLUSION:**

We studied about BCD to Excess-3 Code Converter.

# EXPERIMENT NO.: - 8 B

**AIM*:***To study Excess-3 to BCD converter.

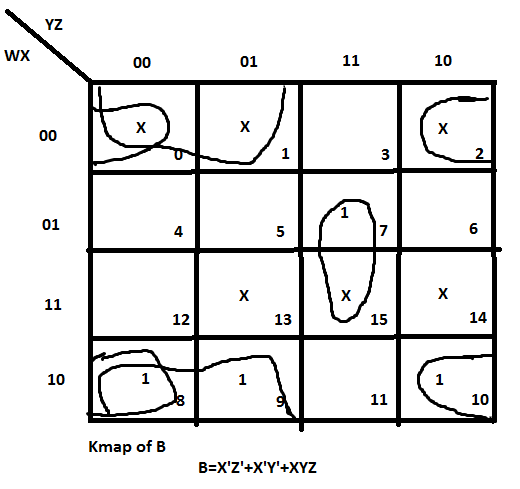
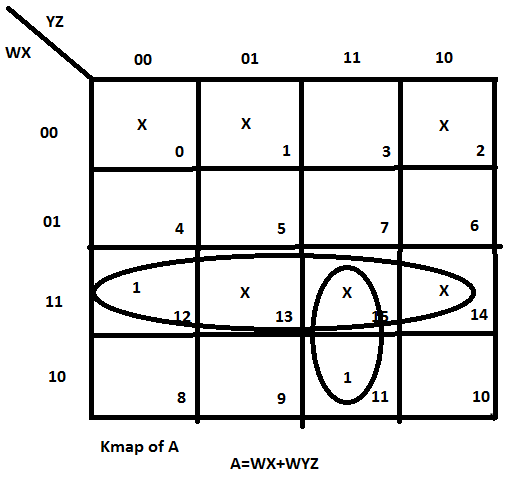
**THEORY:**

Excess-3 binary code is an unweighted self-complementary BCD code. Self-Complementary property means that the 1’s complement of an excess-3 number is the excess-3 code of the 9’s complement of the corresponding decimal number. This property is useful since a decimal number can be nines’ complemented (for subtraction) as easily as a binary number can be ones’ complemented; just by inverting all bits.

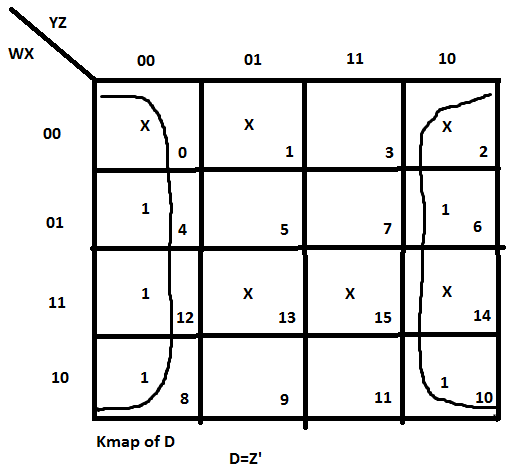
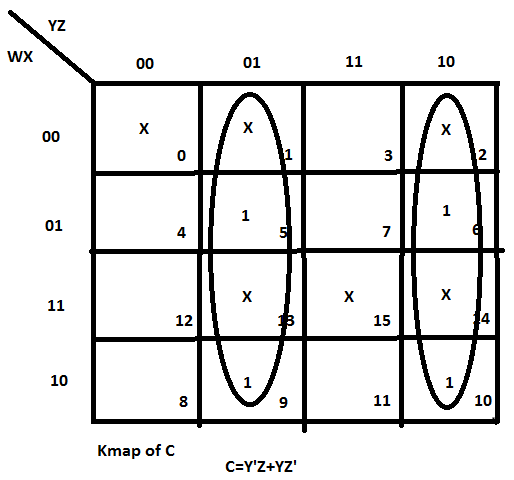
**TRUTH TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Excess-3 | | | | BCD | | | |
| W | X | Y | Z | A | B | C | D |
| 0 | 0 | 0 | 0 | X | X | X | X |
| 0 | 0 | 0 | 1 | X | X | X | X |
| 0 | 0 | 1 | 0 | X | X | X | X |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X |

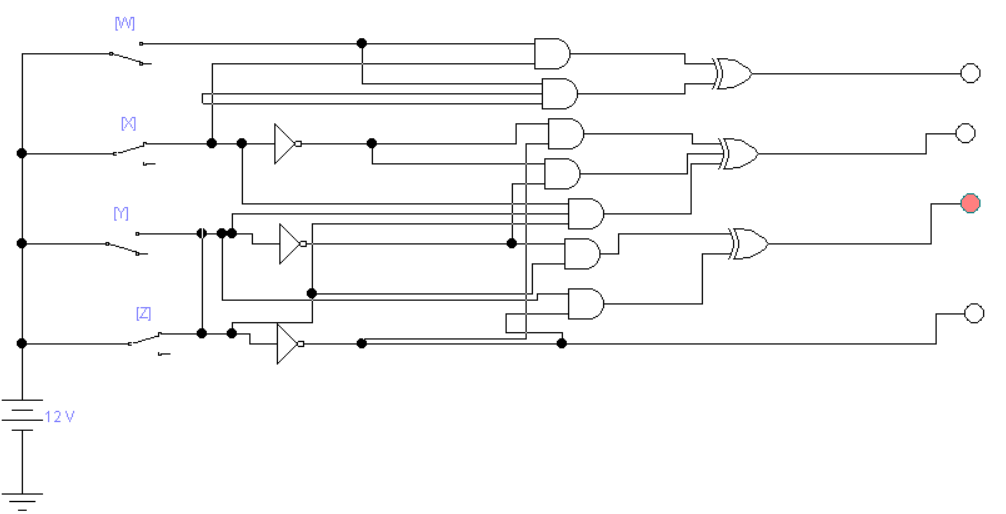
|  |  |
| --- | --- |
| **K-MAP FOR A** | **K-MAP FOR B** |



|  |  |
| --- | --- |
| **K-MAP FOR C** | **K-MAP FOR D** |



**CIRCUIT:**



**CONCLUSION:**

We studied about Excess-3 to BCD code Converter

# EXPERIMENT NO.: - 9

**AIM*:*** To study Binary to BCD code.

**TRUTH TABLE**

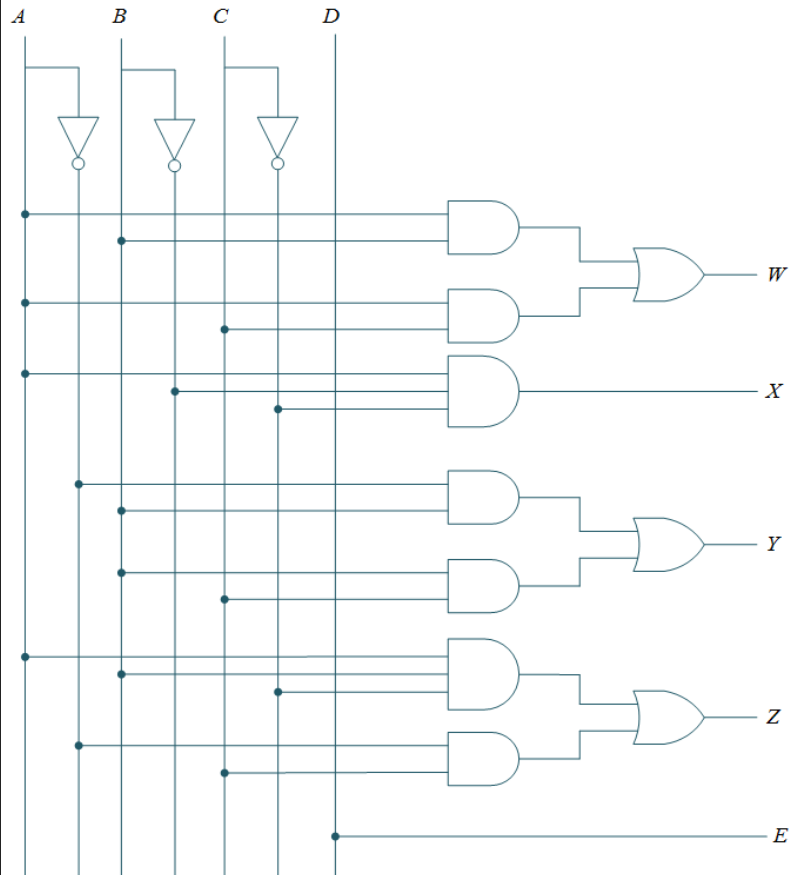
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Binary | | | | BCD | | | | |
|  | W | X | Y | Z | A | B | C | D | E |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 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 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR A**   |  |  |  |  | | --- | --- | --- | --- | | **0** | **0** | **0** | **0** | | **0** | **0** | **0** | **0** | | **1** | **1** | **1** | **1** | | **0** | **0** | **1** | **1** |   **A= WX+XY** | **K-MAP FOR B**   |  |  |  |  | | --- | --- | --- | --- | | **0** | **0** | **0** | **0** | | **0** | **0** | **0** | **0** | | **0** | **0** | **0** | **0** | | **1** | **1** | **0** | **0** |   **B= WX’Y’** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR C**   |  |  |  |  | | --- | --- | --- | --- | | **0** | **0** | **0** | **0** | | **1** | **1** | **1** | **1** | | **0** | **0** | **1** | **1** | | **0** | **0** | **0** | **0** |   **C= W’X+XY** | **K-MAP FOR D**   |  |  |  |  | | --- | --- | --- | --- | | **0** | **0** | **1** | **1** | | **0** | **0** | **1** | **1** | | **1** | **1** | **0** | **0** | | **0** | **0** | **0** | **0** |   **D= WXY’+W’Y** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K-MAP FOR E**   |  |  |  |  | | --- | --- | --- | --- | | **0** | **1** | **1** | **0** | | **0** | **1** | **1** | **0** | | **0** | **1** | **1** | **0** | | **0** | **1** | **1** | **0** |   **E= Z** |

**CIRCUIT:**



**CONCLUSION:**

We are studied Binary code to BCD converter And use this.

# EXPERIMENT NO.: - 10

**AIM:** To study about different Flip flop circuit.

**R-S Flip Flop:**

A multivibrator is a regenerative circuit with two active devices, designed so that one device conducts while the cuts off. Multivibrator can store binary numbers. So, it can perform essential functions like counting of pulsed, synchronizing arithmetic operations etc. Such type of circuit is known as Flip-Flop circuits.



The R-S Flip-Flop is another name of bi-stable multivibrator, one whose output is low or high, 0 or 1. This output can be changed to other state only with the help of an external input called ‘TRIGGER’. Until the external input is applied, the original state of output remains unchanged indefinitely fig. no. 1 shows the circuit diagram of R-S flip-flop, which of two NOT gates and two NAND gates. This flip-flop has two inputs namely R and S (R=reset & S=set) and has outputs namely Q and Q’, where Q’ is always the complement of Q. The truth-table of this R-S flip-flop is as under.



|  |  |  |
| --- | --- | --- |
| **R** | **S** | **Q** |
| 0 | 0 | Last state |
| 0 | 1 | 1(Set) |
| 1 | 0 | 0(Reset) |
| 1 | 1 | 1 Forbidden |

The first input condition in the truth-table is R=0 & S=0. Since 0 input has no effect on its output, the flip-flop simply remains on its previous state i.e. Q remains unchanged.

The second input condition of the truth-table is R=0 & S=1 forces the output to switch over to 1 i.e. at high level. Thus input at 1 level is said that the flip-flop has SET i.e. Q=1(naturally Q’=0).

The third input condition of the truth-table is R=1 & S=0 forces the output to switch over to 0 i.e. at low level. Thus input at 0 level is said that the flip-flop has RESET i.e. Q=0(naturally Q’=1).

The fourth input condition of the truth-table is R=1 & S=1 forces the output of both NAND gates to switch over to 0 i.e. at low level. In other words Q=Q’+0 at the same time. This is nothing but violation of the basic definition of flip-flop that Q must complement of Q’. This state is known as forbidden state. Generally it is agreed upon never to impose this input condition.

**D Flip Flop:**

A multivibrator is a regenerative circuit with two active devices, designed so that one device conducts while the cuts off. Multivibrator can store binary numbers. So, it can perform essential functions like counting of pulsed, synchronizing arithmetic operations etc. Such type of circuit is known as Flip-Flop circuits.



The R-S Flip-Flop has two data input R & S. To store a high bit you need a high S. To store a low bit you need a high R. Generation of two signals to drive a Flip-Flop is a disadvantage in many applications. Further more the forbidden condition of both R=S=1 may occur inadvertently. This had led to change R-S Flip-Flop i.e. a Flip-Flop that needs only a single data input. This Flip-Flop is shown in figure. The truth-table of D Flip-Flop is as under:

|  |  |  |
| --- | --- | --- |
| **CLK** | **D** | **Qn+1** |
| 0 | X | On(Last state) |
| 1 | 0 | 0(Reset) |
| 1 | 1 | 1(Set) |

This kind of Flip-Flop prevents the value of D from reaching the Q output until a clock occurs.

In general D Flip-Flop is a bi-stable circuit whose D input is transferred to the output only after clock pulse is received.

**EQUIPMENT REQUIRED:**

* Trainer kit

**COMPONENTS REQUIRED:**

* IC 7432
* ICs 7400, 7432, 7408
* Hook up wires.

**PROCEDURE:**

1. Connect circuit as per the ckt. Diagram and give 5V DC power supply to input terminals.
2. Switch ON the power supply.
3. Change the state of flip flop as per truth-table and note down the status of output at every stage as per truth-table.

**Conclusion:**

We learn here about two different Flip Flop like R-S Flip Flop and D Flip Flop.

In SR  Flip-Flop, both NAND inputs must normally be logic 1 level. Thus, the logic level of the Q and outputs will become relative. While both R and S inputs become logic 0 level at the same period, it is forbidden. In this state, both Q and outputs will become logic 1 level.

The **D Flip Flop** is by far the most important of the clocked flip-flops as it ensures that ensures that inputs S and R are never equal to one at the same time. The D-type flip flop are constructed from a gated SR flip-flop with an inverter added between the S and the R inputs to allow for a single D (Data) input.