**DEPARTMENT OF ELECTRONICS AND COMMUNICATION**

LABORATORY MANUAL

DIGITAL SYSTEM DESIGN LABORATORY

ATRIA INSTITUTE OF TECHNOLOGY

(Affiliated To Visvesvaraya Technological University, Belgaum)

Anandanagar, Bangalore-24

# DSD LAB MANUAL

3rd SEMESTER ELECTRONICS AND COMMUNICATION

**SUBJECT CODE: 18ECL38**

**2020-21**

**DIGITAL SYSTEM DESIGN LABORATORY**
**(Effective from academic year 2020-2021)**
**SEMESTER-III**

| | | | |
|---|---|---|---|
| **Subject Code** | **18ECL38** | **CIE Marks** | **40** |
| **Number of contact Hours/Week** | **0:2:2** | **SEE Marks** | **60** |
| **Total Number of Contact hours** | **36** | **Exam Hour** | **3** |

**Credits- 02**

**Course Learning Objectives:**

This laboratory course enables students to get practical experience in design, realization and verification of

- DeMorgan's Theorem, SOP, POS forms
- Full/Parallel Adders, Subtractors and Magnitude Comparator
- Multiplexer using logic gates
- Demultiplexers and Decoders
- Flip-Flops, Shift registers and Counters.

**Descriptions (if any):**

- Use discrete components to test and verify the logic gates. TheIC numbers given are suggestive; any equivalent ICs can be used.
- For experiment No. 11 and 12 any open source or licensedsimulation tool may be used.

**Laboratory Experiments:**

| | |
|---|---|
| **1.** | Verify<br>(i) DeMorgan's Theorem for 2 variables.<br>(ii) The sum-of product and product-of-sum expressions using universalgates. |
| **2.** | Design and implement<br>(i) Half Adder & Full Adder using i) basic gates. ii) NAND gates<br>(ii) Half subtractor& Full subtractor using i) basic gates ii) NAND gates |
| **3.** | Design and implement<br>(i) 4-bitParallelAdder/Subtractor using IC 7483.<br>(ii) BCD to Excess-3 code conversion and vice-versa. |
| **4.** | Design and Implementation of<br>(i) 1-bit Comparator<br>(ii) 5-bit Magnitude Comparator using IC 7485. |
| **5.** | Realize<br>(i) Adder & Subtractors using IC 74153.<br>(ii) 4-variable function using IC74151(8:1MUX). |
| **6.** | Realize (i) Adder & Subtractors using IC74139.<br>(ii) Binary to Gray code conversion & vice-versa (74139) |
| **7.** | Realize the following flip-flops using NAND Gates.<br>Master-Slave JK, D & T Flip-Flop. |
| **8.** | Realize the following shift registers using IC7474/7495<br>(i) SISO (ii) SIPO (iii)) PISO(iv) )PIPO (v) Ring (vi) Johnson counter |
| **9.** | Realize (i) Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop (ii) Mod-N Counter using IC7490 / 7476 (iii) Synchronous counter using IC74192 |
| **10.** | Design Pseudo Random Sequence generator using 7495. |
| **11.** | Design Serial Adder with Accumulator and Simulate using Simulation tool. |
| **12.** | Design Binary Multiplier and Simulate using Simulation tool. |

# LIST OF EXPERIMENTS

| EXP. NO | NAME OF THE EXPERIMENT |
|---|---|
| 1 | Verify<br>(a) Demorgan‘s Theorem for 2 variables.<br>(b) The sum-of product and product-of-sum expressions using universal gates. |
| 2 | Design and implement<br>(a) Full Adder using (i) basic logic gates and (ii) NAND gates.<br>(b) Full subtractor using (i) basic logic gates and (ii) NANAD gates. |
| 3 | Design and implement 4-bit Parallel Adder/ Subtractor using IC 7483. |
| 4 | Design and Implementation of 5-bit Magnitude Comparator using IC 7485. |
| 5 | Realize<br>(a) Adder & Subtractor using IC 74153.<br>(b) 3-variable function using IC 74151(8:1MUX). |
| 6 | Realize a Boolean expression using decoder IC74139. |
| 7 | Realize Master-Slave JK, D & T Flip-Flops using NAND Gates. |
| 8 | Realize the following shift registers using IC7474/IC 7495<br>(a) SISO (b) SIPO (c) PISO (d) PIPO (e) Ring and (f) Johnson counter. |
| 9 | Realize (i) Mod-N Asynchronous Counter using IC7490 and<br>( ii) Mod-N Synchronous counter using IC74192 |
| 10 | Design Pseudo Random Sequence generator using 7495. |
|  | Introduction to MULTISIM |
| 11 | Simulate Full- Adder using simulation tool. |
| 12 | Simulate Mod-8 Synchronous UP/DOWN Counter using simulation tool. |

# INTRODUCTORY EXPERIMENT

## VERIFICATION OF TRUTH TABLES OF VARIOUS BASIC GATES

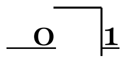
**Aim:** To verify the truth tables of various basic gates.

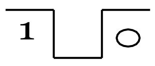
**IC's required:** IC 7408, 7432, 7400, 7402, 7404, 7486, 7411,

**Notes & observations**:

Digital trainer kit provides the following facilities: -

- DC power supplies of + 12V , -12V, +5V (marked red) &
  GND ('0 'level, marked black)
- 4 IC sockets of 16 pin with the corresponding 5V, Vcc & GND knobs.
- 10 No. input (toggle switches ) which can be put ON
  (supplying 5V  0.2 V) OR
  OFF (supplying 0V to 0.8V)
- 10 No. outputs (LED's) where in the low state is indicated by the LED's being 'OFF' & high state represented by the ON (glow) of LED's.
  low state – (0 – 0.4V) at the LED's High
  state- ( 2.7 V– 5V) at the LED's
- Bread board can be used for IC's having more than 16 pins &for an exceptusing more than 4 IC's.
- Mono pulse - 1) Positive (marked red)  gives a pulse of the form 0
  2) Negative (marked block)  gives a pulse of the form 1



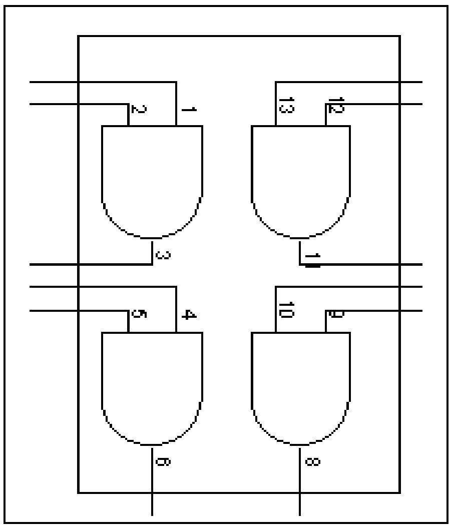


- TTL Clock's - There are 4 or 5 TTL Clocks (square wave GENERATORS OF different frequencies– 1 Hz, 10Hz, 100Hz, 1KHz & 10 KHz with (0V to 5V) peak to peak value.

- Patch chords – connecting wires for making inter connections.

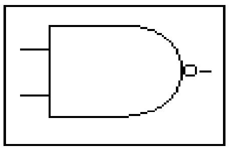
## 1) AND Gate: IC 7408 Quad 2 inputs, 14 pins DIP (Dual in package)

***Pin Diagram***



Pin 14: VCC
Pin 7: GND

***Logic diagram***



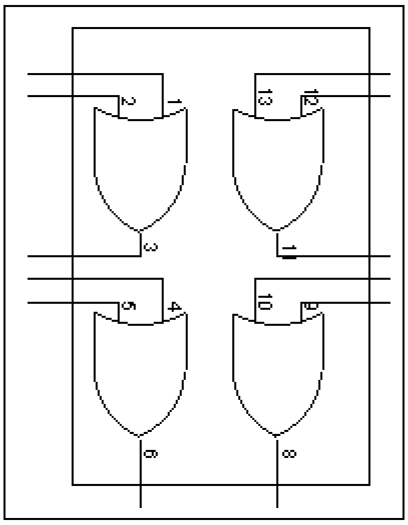
Y= A.B

***Truth Table***

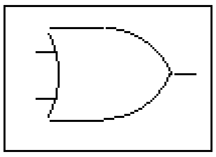
| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 2) OR Gate: IC 7432 Quad 2 I/P, 14 pins DIP (Dual in package) IC

***Pin Diagram***



***Logic Diagram***



Y = A+B

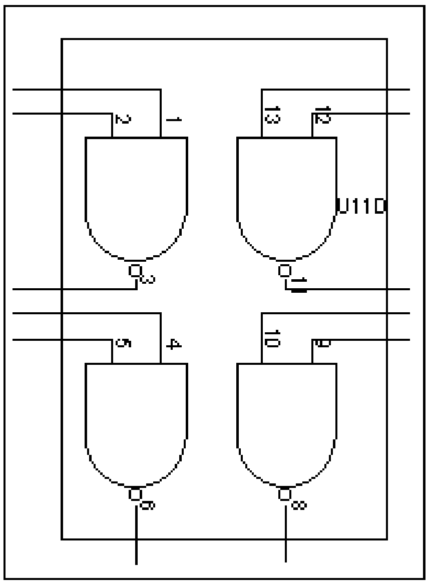
Pin 14: VCC
Pin 7: GND

***Truth table***

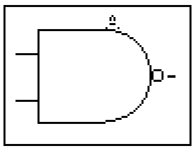
| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## 3) NAND Gate: IC 7400 - Quad 2 I/P, 14 Pins DIP IC

### *Pin Diagram*



### *Logic Diagram*



$Y = \overline{A.B}$

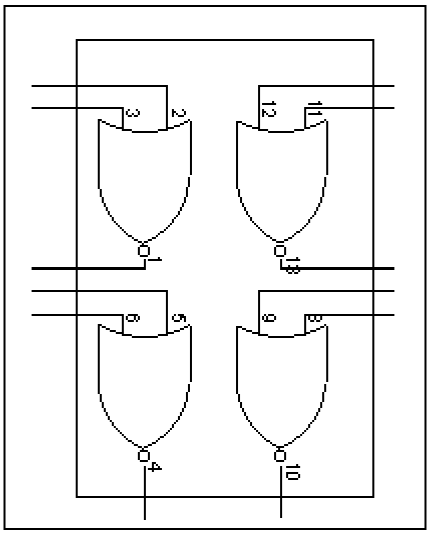
### *Truth Table*

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

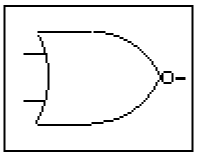
Pin 7: GND
Pin 14: VCC

## 4) NOR Gate: IC 7402 – Quad 2 inputs, 14 pins Dip IC

### *Pin Diagram*



### *Logic Diagram*



Y=A+B

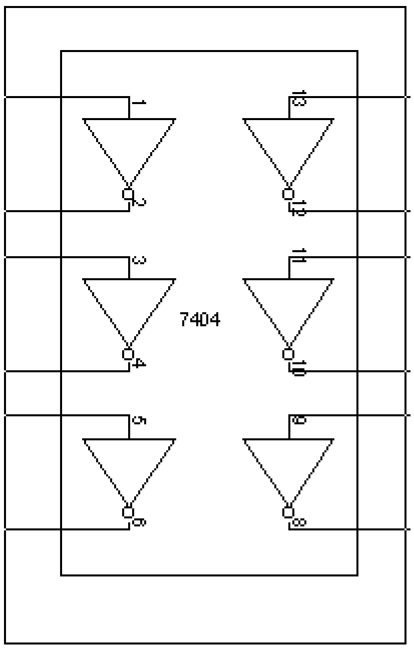
### *Truth Table*

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

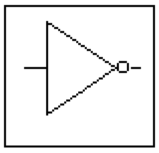
Pin 14:VCC
Pin 7: GND

## 5. NOT Gate: IC 7404 – Hex inverter, 14 pins Dip IC.

**Pin Diagram**



**Logic Diagram**



Y=$\overline{A}$

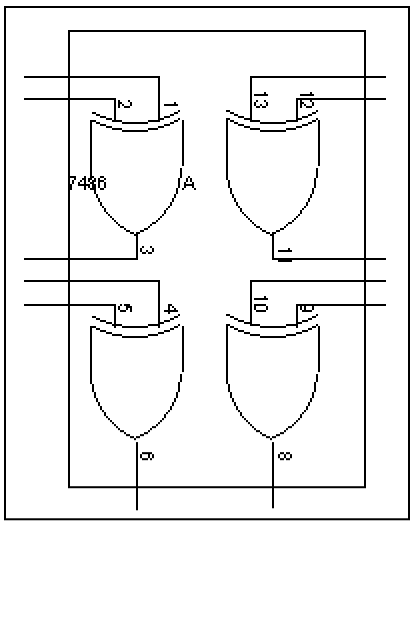
Pin 7: GND
Pin 14: VCC

**Truth Table**

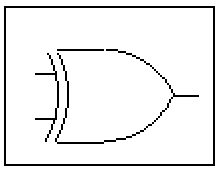
| A | B |
|---|---|
| 0 | 1 |
| 1 | 0 |

## 6) X- OR Gate: IC 7486 – Quad 2-i/p , 14 pins Dip IC.

**Pin Diagram**



**Logic Diagram**



Y = A ⊕ B

Pin 7: GND
Pin 14: VCC

**Truth Table**

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**PROCEDURE:**

1. Fix the IC into the socket provided on the trainer Kit (ZIP), So that the notch on the IC should be at the top end.
2. Make the connections using patch chords, as given in the logic diagram, referring to the pin diagram of the IC's.
3. The Vcc & Gnd pins are connected to +5V & Ground (or) respectively.
4. Give the I/P's (A, B) through the toggle switches & observe the o/p 'Y' by connecting to LED.

**Result:** ..........................................................................................

## Experiment No.1

## DEMORGAN'S THEOREM & SOP and POS

**Aim:** A) To Verification of Demorgan's theorem for 2 variables
B) To Verification of sum of products (SOP) and product of sum (POS) expressions using basic gates and universal gates.

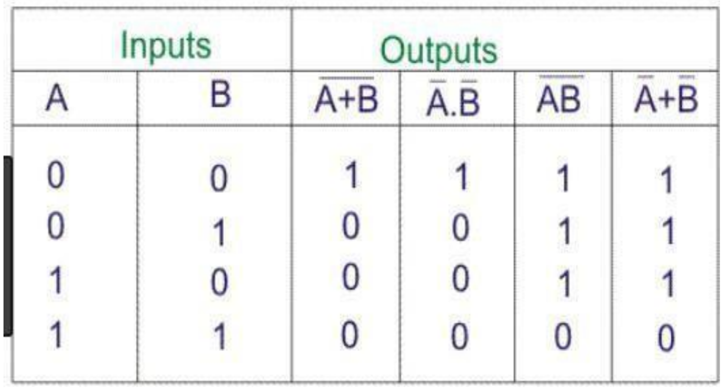
**Components Required:**

| SI.NO | Components & ICs | Numbers |
|---|---|---|
| 1. | Digital IC Trainer Kit | 1 |
| 2. | 7404 | 1 |
| 3. | 7408 | 1 |
| 4. | 7432 | 1 |
| 5. | Patch chords | 20 |

### a) Demorgan's1st theorem Statement:

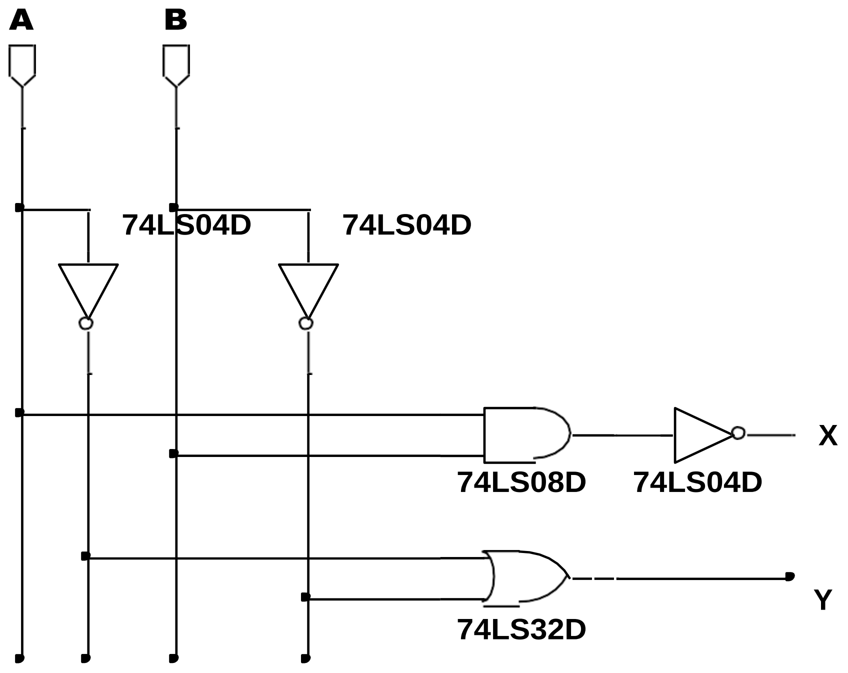
"The compliment of product of 2 or more variables is equal to the sum of compliment of 2 or more variables". Thus according to De-Morgan's laws or De-Morgan's theorem if A and B are the two variables or Boolean numbers. Then accordingly, Eg:

$$\overline{A.B} = \overline{A} + \overline{B}$$



| Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| A | B | $\overline{A+B}$ | $\overline{A}.\overline{B}$ | $\overline{AB}$ | $\overline{A}+\overline{B}$ |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

**Logic Diagram:**



## <u>Demorgan's 2nd Theorem Statement:</u>

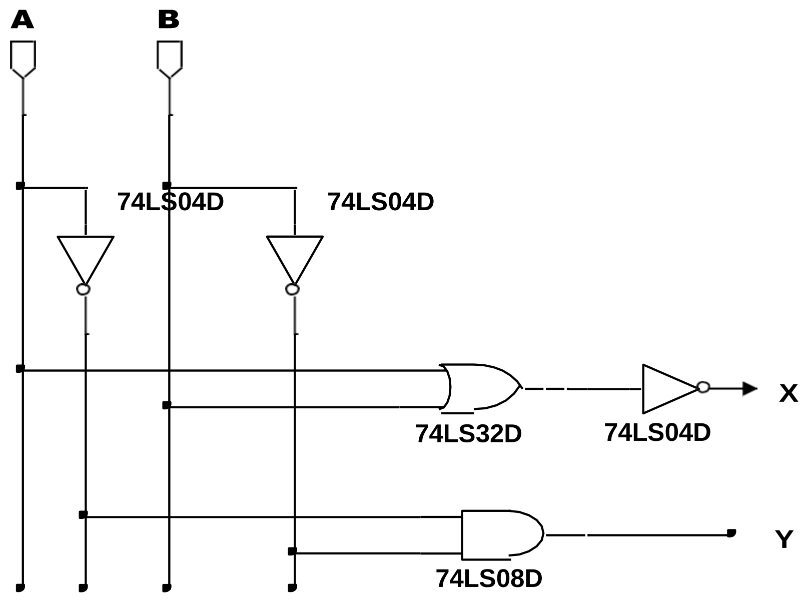
"The compliment of sum of 2 or more variables is equal to the product of compliment of 2 or more variables". Thus according to De Morgan's theorem if A and B are the two variables then, Eg: $\overline{A+B} \equiv \overline{A} \cdot \overline{B}$

**Truth Table:**

**<u>Proof :</u>**

| A | B | $\bar{A}$ | $\bar{B}$ | $\overline{A.B}$ | $\bar{A}+\bar{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |

**Logic Diagram:**



**Procedure:-**

1. Place the IC in the socket of the trainer kit.
2. Make the connections as shown in the circuit diagram.
3. Apply the different combinations of input according to truth table andverify the output.

## b) Verification of SOP and POS expressions

**Components Required:**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1. | 7404 | 1 |
| 2. | 7408 | 1 |
| 3. | 7432 | 1 |
| 4. | 7400 | 2 |
| 5. | 7410 | 1 |
| 6. | 7402 | 2 |
| 7. | 7427 | 1 |
| 8. | Patch chords | 20 |

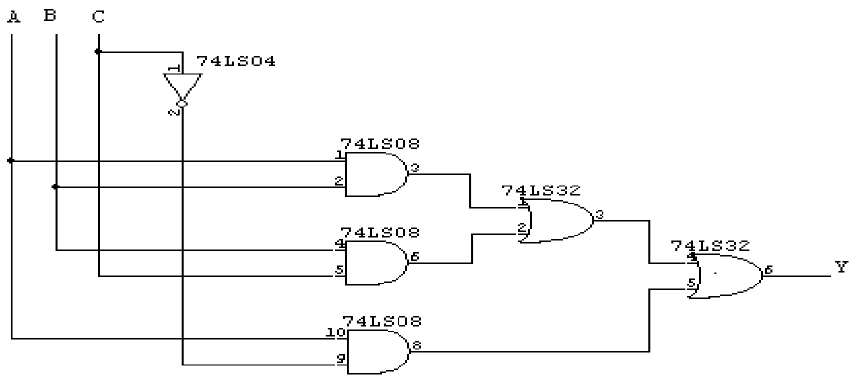
## i) SOP EXPRESSION

**Y = AB + AC□+BC**

**Truth Table:**

| Inputs | | | Output |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Logic Diagram: Realization Using Basic gates**



**Realization Using NAND gates**

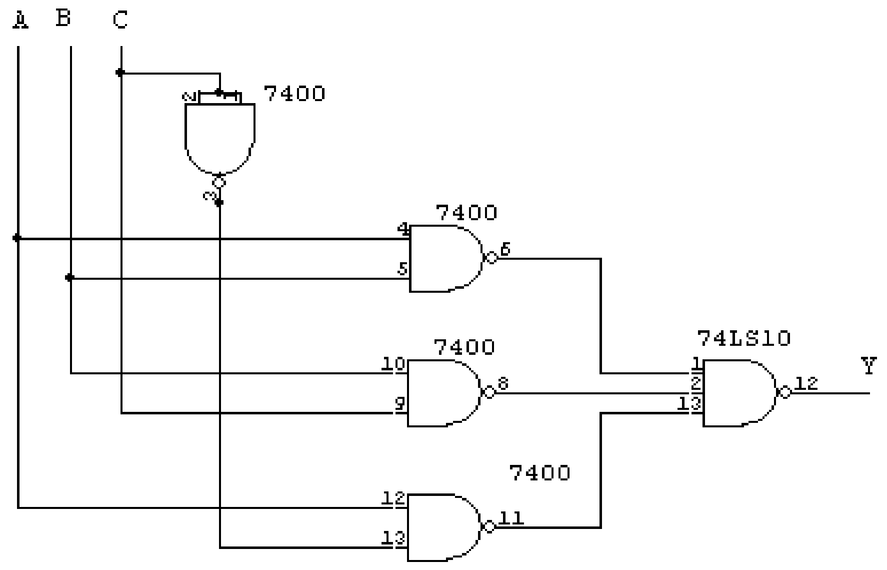
$$Y = AB + A\bar{C} + BC$$

$$Y = \overline{\overline{AB + A\bar{C} + BC}}$$

$$Y = \overline{\overline{AB}\,.\,\overline{A\bar{C}}\,.\,\overline{BC}}$$

(using Demorgan's theorem)

**Logic Diagram:**



**Realization Using NOR gates**

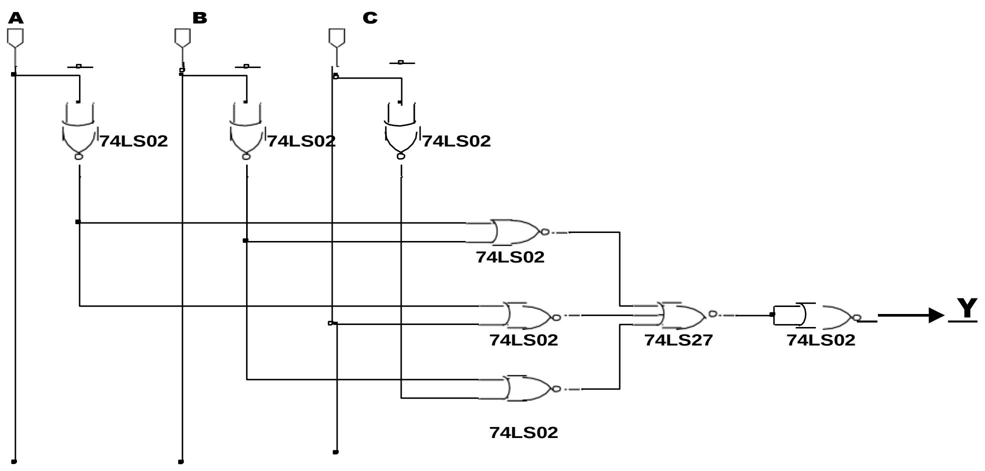
$$Y = AB + A\bar{C} + BC$$

$$Y = \overline{\overline{AB\ +\ A\bar{C} + BC}}$$

$$Y = \overline{\overline{AB}\,.\overline{A\bar{C}}\ .\overline{BC}}$$ (using Demorgan’s theorem)

$$\mathrm{Y} = \overline{(\bar{A}\ +\ \bar{B})} + \overline{(\bar{A}\ +\ C)} + \overline{(\bar{B}\ +\ \bar{C})}$$

**Logic Diagram:**



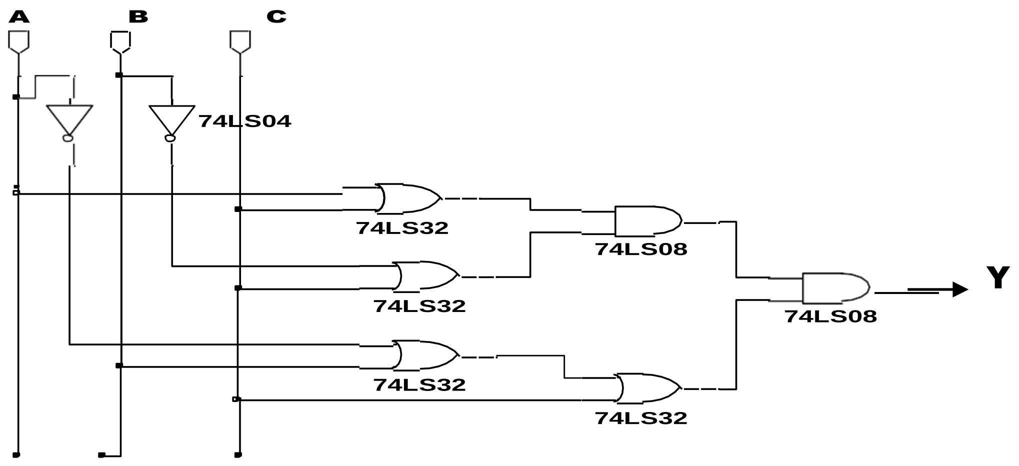
## ii) POS EXPRESSION

$$Y = (A + C)\,(B + C)\,(\bar{A} + B + C)$$

**Truth Table:**

| Inputs | | | Output |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Logic Diagram: Realization Using Basic gates**



**Realization Using NAND gates**

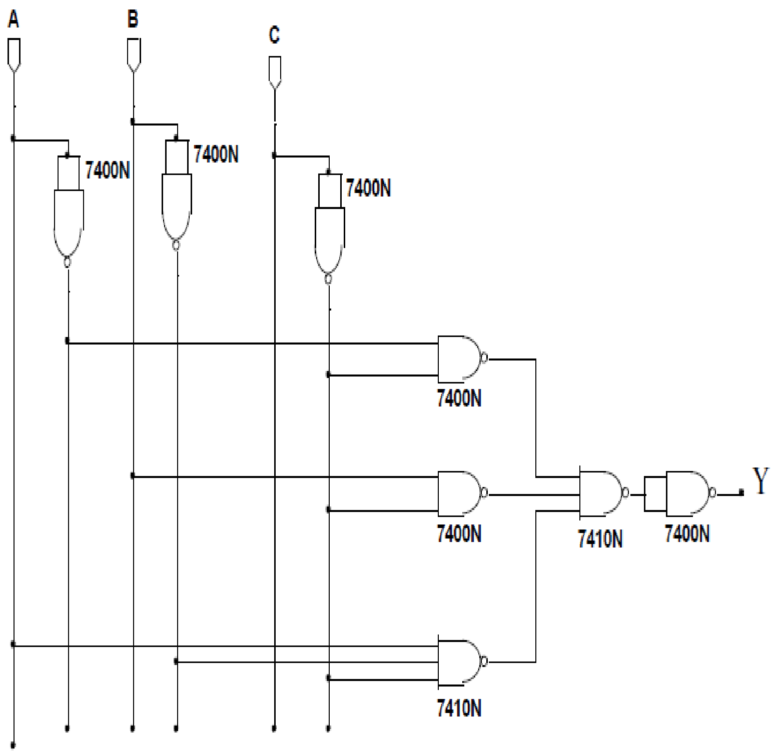
$$Y = (A + C)(\bar{B} + C)(\bar{A} + B + C)$$

$$Y = \overline{\overline{(A + C)(\bar{B} + C)(\bar{A} + B + C)}}$$

$$Y = \overline{\overline{(A + C)} + \overline{(\bar{B} + C)} + \overline{(\bar{A} + B + C)}}$$

$$Y = \overline{(\bar{A}.\bar{C}) + (B.\bar{C}) + (A.\bar{B}.\bar{C})}$$

$$Y = \overline{(\bar{A}.\bar{C})} . \overline{(B.\bar{C})} . \overline{(A.\bar{B}.\bar{C})}$$



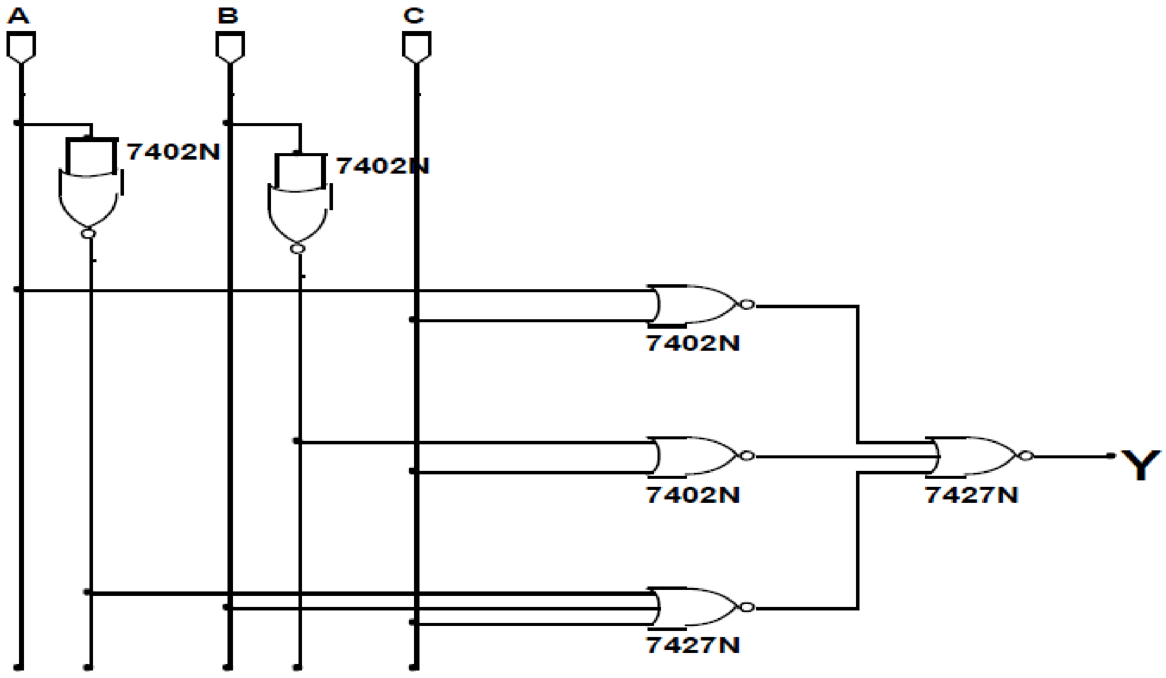
## Realization Using NOR gates

$$Y = (A + C)(\bar{B} + C)(\bar{A} + B + C)$$

$$Y = \overline{\overline{(A + C)(\bar{B} + C)(\bar{A} + B + C)}}$$

$$Y = \overline{\overline{(A + C)} + \overline{(\bar{B} + C)} + \overline{(\bar{A} + B + C)}}$$

**Logic Diagram:**



**Procedure:-**

1. Place the IC in the socket of the trainer kit.
2. Make the connections as shown in the circuit diagram.
3. Apply the different combinations of input according to truth table andverify the output.

**Result:** ...........................................................................................

## Experiment No. 2

## FULL ADDER AND FULL SUBTRACTOR

**Aim:** To Design and implement
(a) Half/Full Adder using (i) basic logic gates and (ii) NAND gates.
(b) Half/Full Subtractor using (i) basic logic gates and (ii) NANAD gates.

**Components Required:**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1. | 7404 | 1 |
| 2. | 7486 | 1 |
| 3. | 7408 | 1 |
| 4. | 7432 | 1 |
| 5. | 7400 | 2 |
| 6. | Patch chords | 20 |

### (a) Half/Full adder

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C. The Boolean functions describing the half adder are:**Sum =** $A \oplus B$ **, Cout = A B**

### I) TO REALIZE HALF ADDER

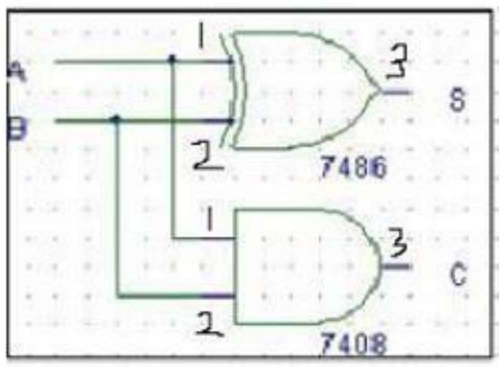
**TRUTH TABLE**

| INPUTS | | OUTPUTS | |
|---|---|---|---|
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**BOOLEAN EXPRESSIONS:**

$S = A \oplus B$
$C = A\,B$

**Circuit diagram:**



## Half adder using NAND gates

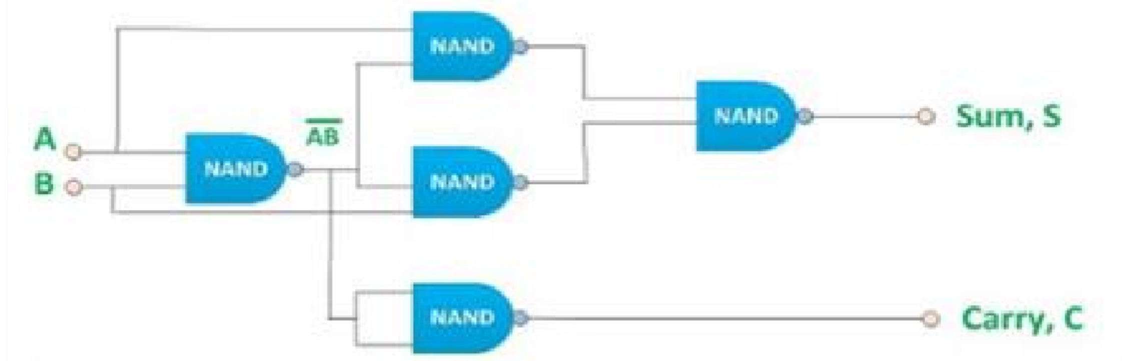
## Realization :

$$\text{SUM, } S = \overline{\overline{A.\overline{AB}}\;.\;\overline{B.\overline{AB}}}$$

$$\text{SUM, } S = A\overline{B} + \overline{A}B$$

$$\text{Carry, } C = AB = \overline{\overline{AB}}$$

## Circuit diagram



**FULL ADDDER:**

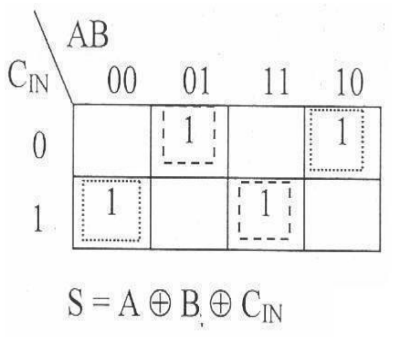
The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, C $_{in}$ , is called a full-adder. The Boolean functions describing the full-adder are:

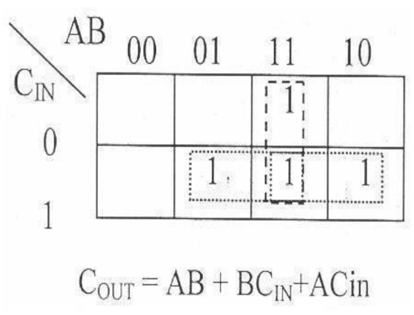
**Sum =** $A \oplus B\ Cin$ **Cout = AB + Cin (**$A \oplus B$**)**

**Truth Table:**

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | $C_{in}$ | Sum (S) | Carry Out (Cout) |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Design:**





**SUM**

$S = A\bar{B}\bar{C}+\bar{A}\bar{B}C+\bar{A}B\bar{C}+AB\ C$

$S = \bar{C}(A\bar{B}+\bar{A}B)\ +C\ (\bar{A}\bar{B}+AB)$

$S = \bar{C}(A\oplus B)\ +C(\overline{A\oplus B})$

$S=A\ \oplus(B\oplus C)$

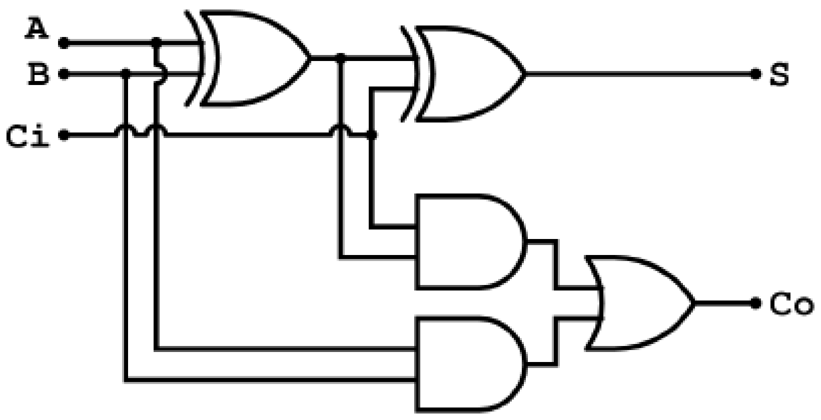
**CARRY**

$Cout=\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC$

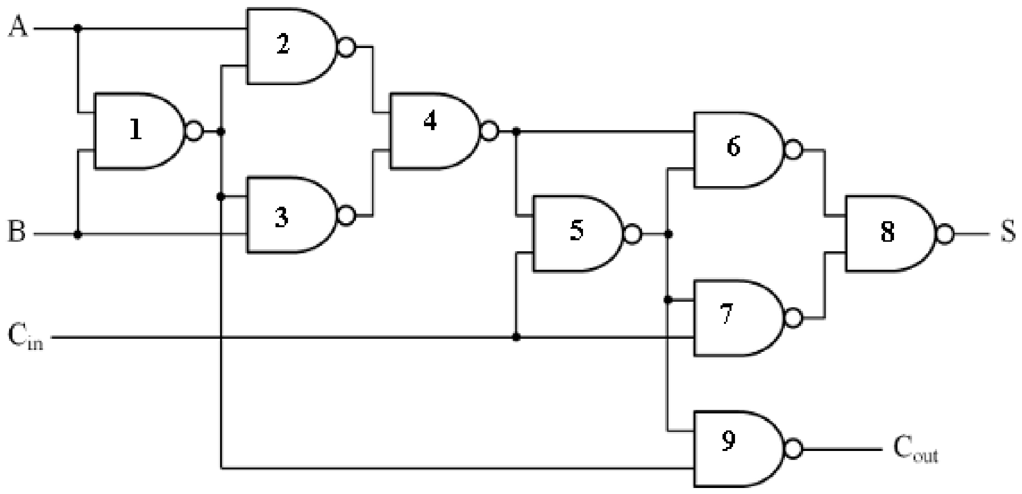
$Cout=AB(C+\bar{C})\ +\ C\ (A\oplus B)$

$Cout=AB+\ C\ (A\oplus B)$

## (i) Full adder using basic gates



## (ii)Full adder using NAND gates



## (b) Half/Full Subtractor:

### III) HALF SUBTRACTOR

**TRUTH TABLE**

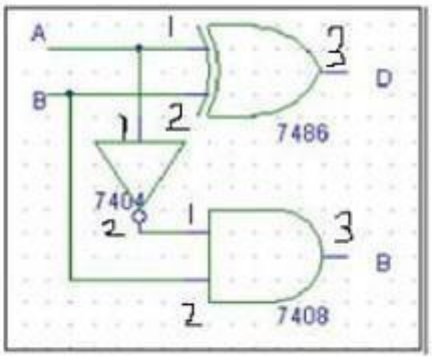
| INPUTS | | OUTPUTS | |
|---|---|---|---|
| A | B | D | Br |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

**BOOLEAN EXPRESSIONS:**

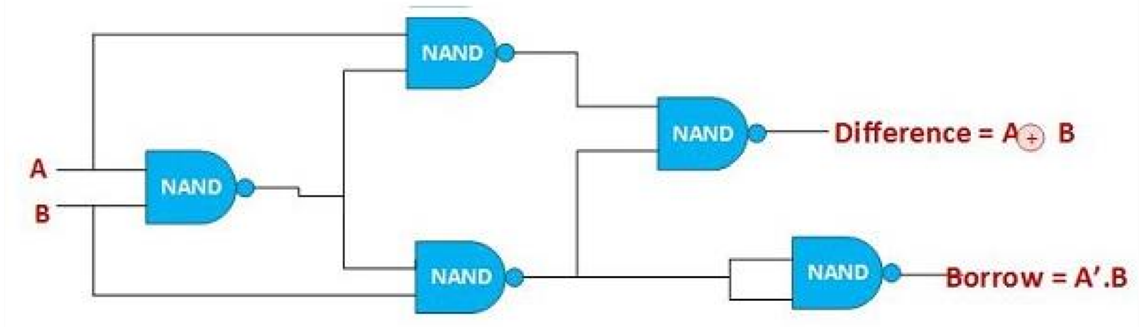
$D = A \oplus B$

$Br = \bar{A} B$

**Circuit diagram:**



## Using NAND gates:

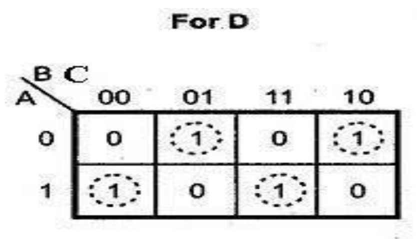


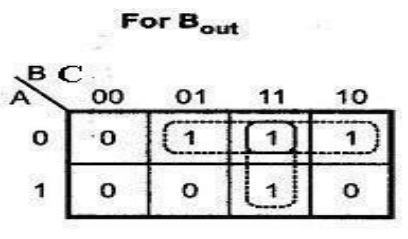
## FULL SUBTRACTOR:

**Truth Table**

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Diff (D) | Borrow (Bout) |
| 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 |

**Design:**





**DIFFERENCE**

$D=A\overline{BC}+A\overline{B}C+\overline{A}B\overline{C}+ABC$

$D=\overline{C}(A\overline{B}+\overline{A}B)+C(\overline{AB}+AB)$

$D=\overline{C}(A\oplus B)+ C(\overline{A\oplus B})$

$D=A\oplus(B\oplus C)$

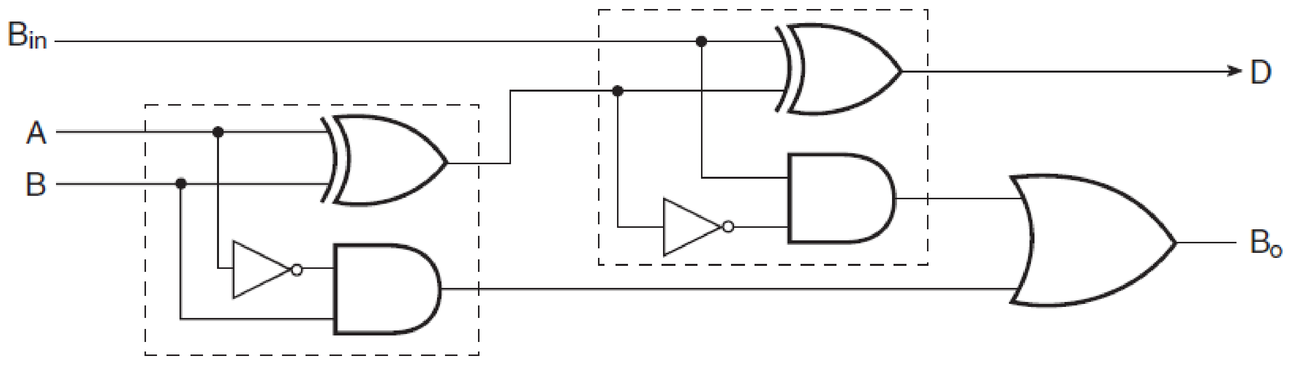
**BORROW**

$Bo=\overline{A}\,\overline{B}C+\overline{A}B\overline{C}+\overline{A}BC+ABC$

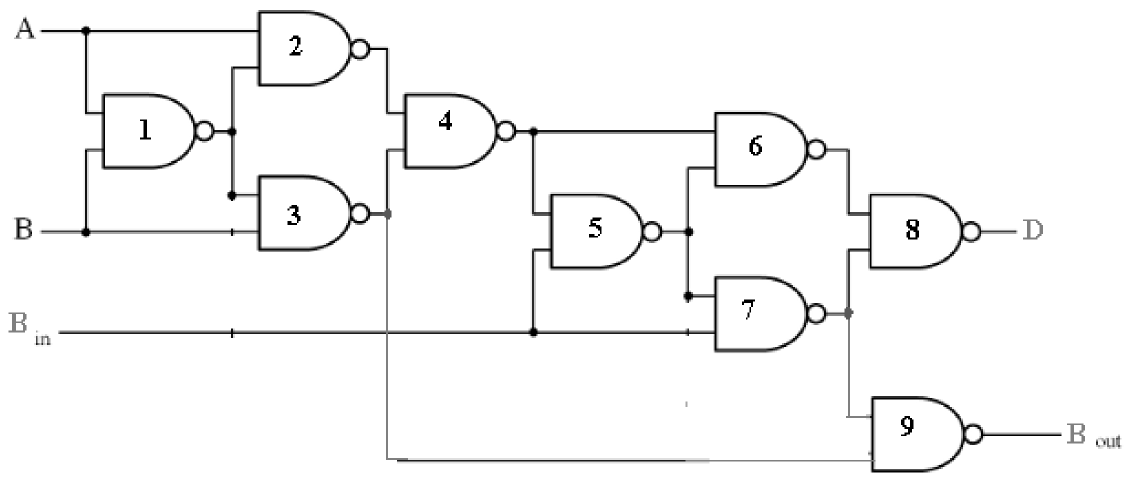
$Bo = \overline{A}B(C+\overline{C})+C(\overline{A\oplus B})$

$Bo=\overline{A}B +C(\overline{A\oplus B})$

(i) **Full subtractor using basic gates**



(ii) **Full subtractor using NAND gates**



**Procedure**

1. Place the IC in the socket of the trainer kit.
2. Make the connections as per the circuit diagram.
3. Switch on $V_{CC}$ and apply various combinations of input according to the truth table.
4. Note down the output readings for full adder and full subtractor sum/difference and the carry/borrow bit for different combinations of inputs.

***Result:*** ……………………………………………………………………………………

## Experiment No: 3

## Design and implement

**(i) 4-bitParallelAdder/Subtractor using IC 7483.**
**(ii) BCD to Excess-3 code conversion and vice-versa.**

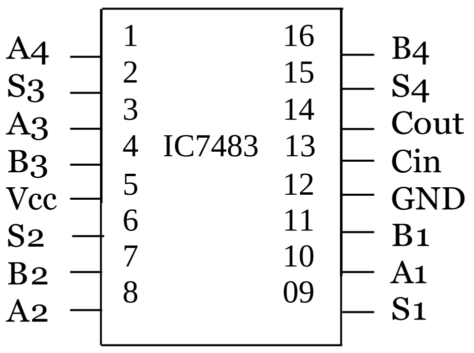
**Aim:** To Study of Parallel Adder/Subtractor/Code Converters using IC 7483.

### Components Required:

| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | 7483 | 1 |
| 2 | 7486 | 1 |
| 3 | Patch chords | 20 |

### (i) 4-bitParallelAdder/Subtractor using IC 7483

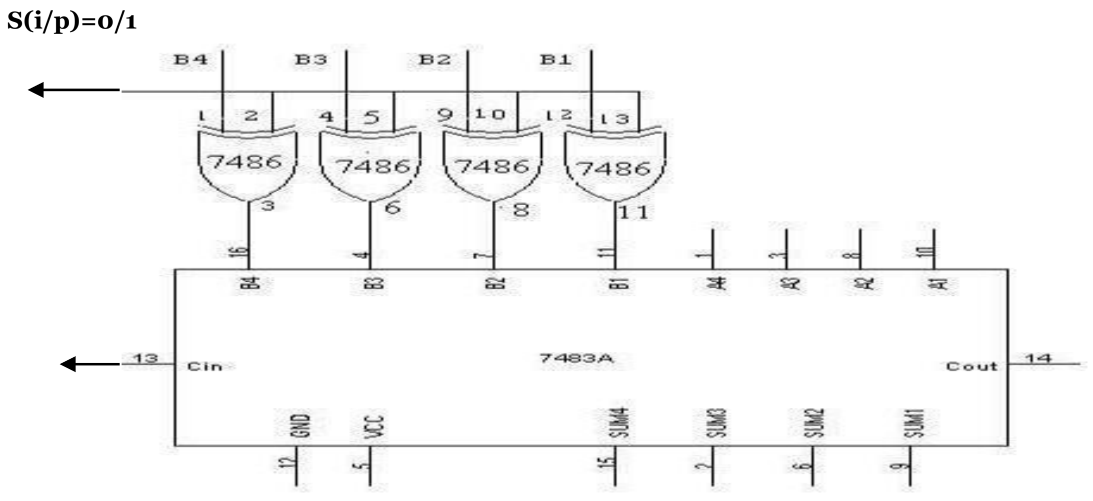
**Pin details of 7483 chip:**



here A1 A2 A3 A4 & B1 B2 B3 B4 & Cin are the I/P's

S1 S2 S3 S4 & Cout are the O/p's

**4-BIT ADDER / SUBTRACTOR OPERATION:**



## 4-BIT ADDER OPERATION:

If control I/P S=0 & Cin =0, addition can be performed.

```
Eg 1: if    A4  A3   A2  A1 =  0 0 1 0  0← Cin
            B4   B3   B2  B1 =1 0 0 0
       ______________     __________________
  Sum = S4   S3   S2   S1 = 1 0 1 0 & Cout =0
```

## 4 – BIT SUBTRACTION USING ONE’S & TWO’S COMPLEMENT METHOD:

### One’s Complement method:

If S=1 & Cout is shorted to Cin, 1’s complement method of subtraction can be performed.

```
 Eg 1: if A4  A3  A2  A1 = 1 0 0 1
            B4 B3 B2 B1 = 0 0 1 1
Diff=   S4   S3   S2   S1 =   0 1 1 0    [Normal  Subtraction]


If        A4 A3 A2     A1   =   1 0 0 1
         B4 B3 B2      B1   =   1 1 0 0     ( 1’S Complement of 0011)

                                1 0 1 0 1   [Normal  Subtraction]
                    └──────►            1
                          ________________
        Diff= S4 S3   S2   S1 = 0 1 1 0
```

If Cout =1, the result is positive

If Cout =0 , the result is negative & is in 1’s complement form.

### 2’s Complement method:

If S=1 & Cin =1 , 2’s complement method of subtraction canbe performed.

```
Eg 1: if A4  A3  A2   A1   = 1 0 0 1
         B4   B3  B2  B1  =   0  0 1 1     [2’s complement of 0011]
       ______________      __________
Diff=   S4   S3   S2 S1 =  1  0 1 1 0
```

If Cout =1, the result is positive, Discard the carry

If Cout =0, the result is nagative & is in 2’s complement form

## (ii) BCD to Excess-3 code conversion and vice-versa

**Aim :** To design & implement the following using 4- bit adder chip.

1) Excess-3 to BCD Code converter.
2) BCD to Excess –3 Code converter.

***Notes & Observations:***

- Excess-3 Code can be obtained by adding binary 3 with the BCD code.
- BCD code is obtained by adding binary 13 with excess –3 code. This is achieved by applying binary 12 as the second no. of addition & another 1 is added by making Cin as logic '1'.
- IC 7483 is a 4-bit Parallel adder chip; where in two 4-bit binary numbers can be addedparallel.

### a) ***Excess –3 to BCD:***

***Truth Table:***

| I/P's Excess-3 | | | | O/P's BCD Code | | | |
|---|---|---|---|---|---|---|---|
| E3 | E2 | E1 | E0 | B3 | B2 | B1 | B0 |
| 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 |

In the truth table of Excess-3 to BCD Code converter, only ten out of sixteen combinations are used, & the other six are taken as don't care conditions.

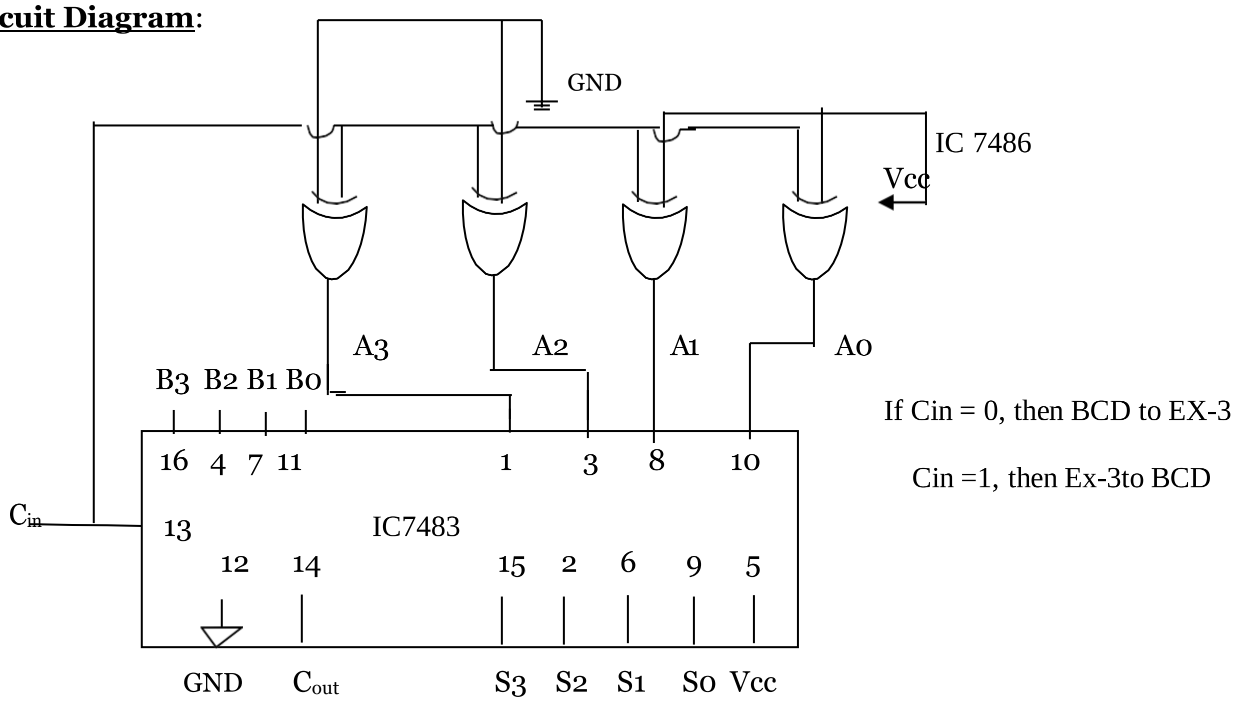
***b) BCD to Excess – 3***

In the truth table of BCD Code to Excess-3 converter, only ten out of sixteen combinations are used, & the other six are taken as don't care conditions

Truth Table

| **I/P's BCD Code** | | | | **O/P's Excess-3 Code** | | | |
|---|---|---|---|---|---|---|---|
| **B3** | **B2** | **B1** | **B0** | **E3** | **E2** | **E1** | **E0** |
| 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 |

**Circuit Diagram**:



If Cin = 0, then BCD to EX-3

Cin =1, then Ex-3to BCD

**Procedure:**

1. Make the connections as shown above.
2. Switch ON the trainer power supply.
3. Give the different I/P's at B3 B2 B1 B0 by means of toggle switches & note the corresponding O/P's on the LED's.
4. When Cin =0 a binary 3 at A3 A2 A1 A0 is added to the augend & hence the O/P will be in Excess-3. This is the conversion of BCD to Excess – 3. Similarly verify for all combinations.
5. When Cin=1, a binary 12 at A3A2 A1 A0 +1 (as Cin) is added to the augend & this is the conversion of Excess-3 to BCD. Similarly verify for all combinations.
6. Note that during Excess-3 to BCD conversion, Cout is set high.

***Result:*** **.......................................................................................**

# Experiment No. 4
## Design and Implementation of
## (i) 1-bit Comparator
## (ii) 5-bit Magnitude Comparator using IC 7485.

**Aim:-**a) To realize a one bit comparator.
b) To Implementation of 5-bit Magnitude Comparator using IC 7485.

**Components Required**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | 7485 | 1 |
| 3 | 7404 | 1 |
| 4 | 7408 | 1 |
| 5 | 7486 | 1 |
| 6 | Patch chords | 20 |

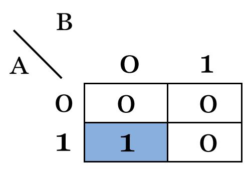
## i) One bit Comparator

**Truth table**

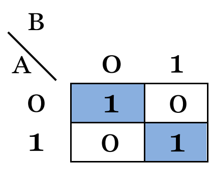
| Inputs | | Outputs | | |
|---|---|---|---|---|
| A | B | A<B | A=B | A>B |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

**Design:**

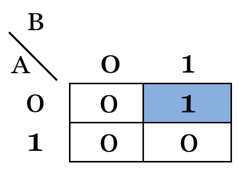
*A>B*



**A=B**



**A<B**



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

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

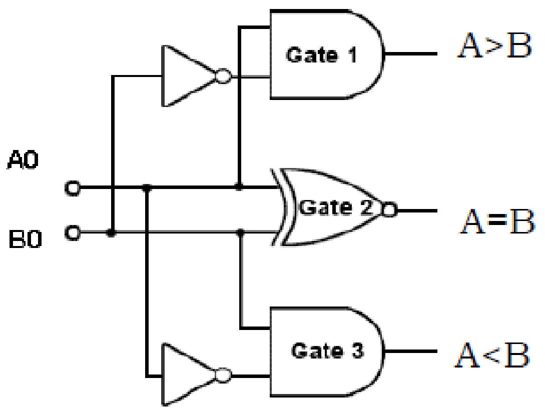
$A<B = A\ B$

—

**Logic diagram:Realization using Logic Gates**

$A>B = A\overline{B}$ $A=B = \bar{A}\bar{B} + AB$ $A<B = \overline{A}B$

**Logic diagram:Realization using Logic Gates**



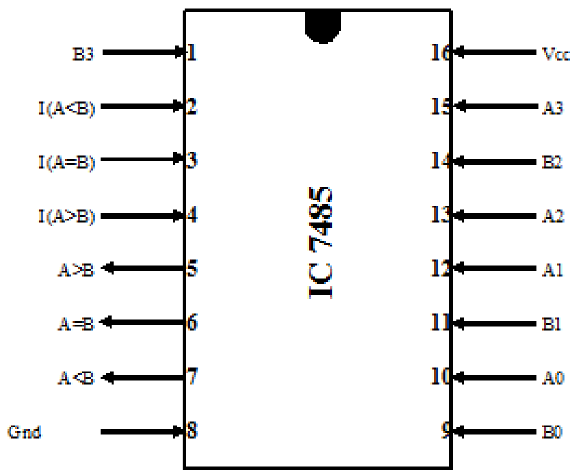
## ii) 5-bit Magnitude Comparator using IC 7485

In simple words -IC 7485 compares 4 bit inputs. If we want 5 bit magnitude comparator then we have to adjust one input line.

Every 7485 IC has three cascading input lines which can used in this case because we are using only one IC to make 7485 comparator. So, as shown in above diagram A>B is grouped with one of the input and A<B is grouped with another group. Now we have 5 bits inputs. Input side is done!

The output side will have EX- NOR gate between two OUTPUTS A>B and A<B. EX-NOR will give output ‘1’ when both the inputs are zero. So, we will consider output of EX-NOR gate as A= B. simple !Thus we have now 5 bit magnitude comparator.

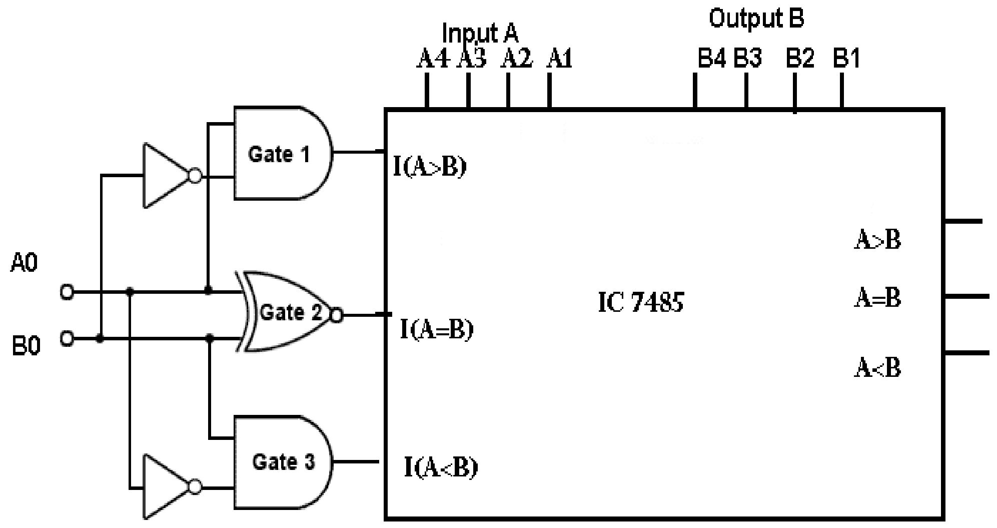
**Pin details of 7485 chip:**



**Truth table for 5-bit Comparator**

| INPUTS | | | | | | | | | | OUTPUTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | | | B | | | | | A>B | A=B | A<B |
| A4 | A3 | A2 | A1 | A0 | B4 | B3 | B2 | B1 | B0 | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**5- bit Magnitude Comparator using IC7485**



**Procedure:**

1. The IC is fixed on the IC zip socket and Vcc & Gnd connections aregiven from 5V Supply.
2. Connections are made as shown in the Logicdiagram.
3. All the inputs are connected to the switches & output to the LEDs.
4. Truth table is verified for different combinations of input.

***Result:*** ........................................................................................

# Experiment No. 5
# MULTIPLEXER

**Aim:** a) To realize Adder & Subtractor using IC 74153.
b) To realize 4- variable function using IC 74151 (8:1 MUX)

**Components Required:**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | 74153 | 1 |
| 2 | 74151 | 1 |
| 3 | 7404 | 1 |
| 4 | Patch chords | 20 |

## MULTIPLEXER & DEMULTIPLEXER

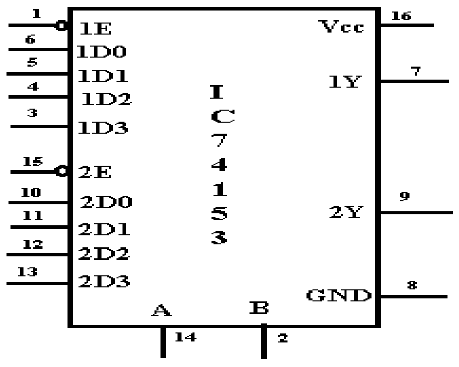
The multiplexer (or data selector) is a logic circuit that gets one out of several inputs to single output. The input selected is controlled by a set of select inputs. For selecting one out of n-inputs, a set of m- select inputs is required, where $2^m = n$.

### DE- MULTIPLEXER:

The Demultiplexer performs the reverse operation of a multiplexer. It accepts single input & distributes it over several outputs. The selected I/P code determines to which output, the data input will be transmitted.

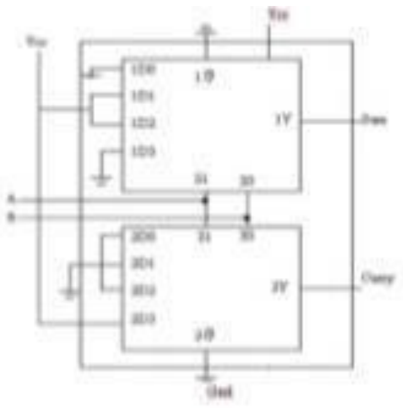
## a) Adder & Subtractor using IC 74153.

### Pin Diagram



i) HALF ADDER USING MUX 74153:

Circuit diagram:

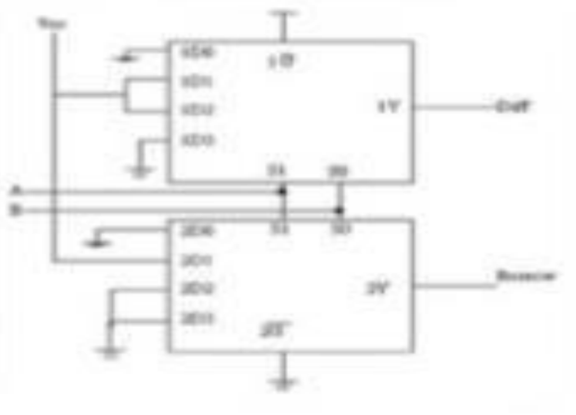


**TRUTH TABLE**

| Input s | | Outputs | |
|---|---|---|---|
| **A** | **B** | **S** | **C** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

ii) HALF SUBTRACTOR USING MUX 74153:

Circuit diagram:



**TRUTH TABLE**

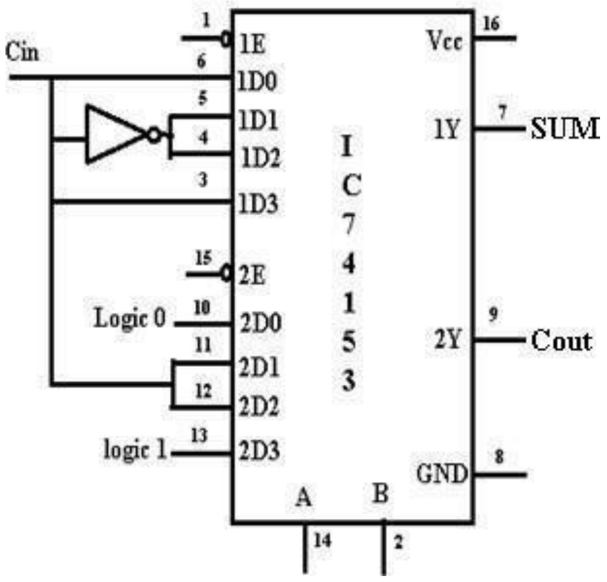
| Inputs | | Outputs | |
|---|---|---|---|
| **A** | **B** | **D** | **Br** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

## Truth Table of Full adder and Full subtractor:

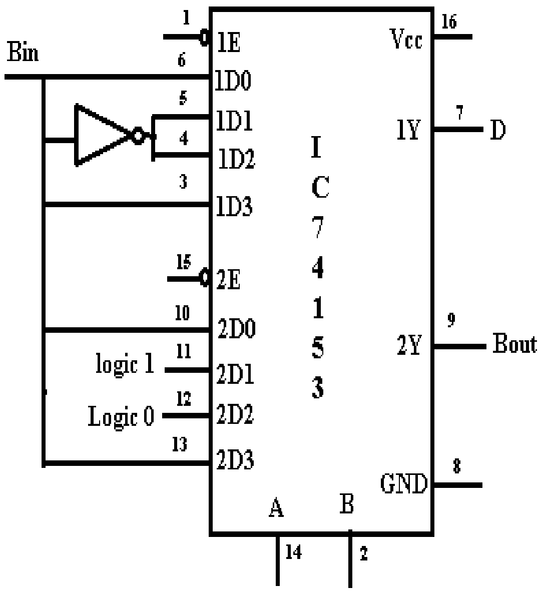
| Inputs | | | Full Adder Outputs | | Full Subtractor Outputs | |
|---|---|---|---|---|---|---|
| A | B | Cin/Bin | SUM | Cout | D | Bout |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## BLOCK DIAGRAM:

Full Adder using 74153

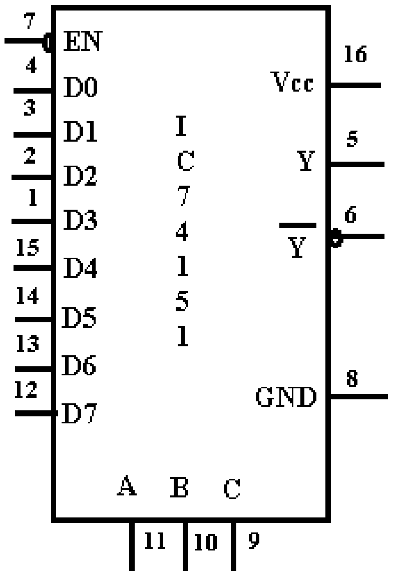


Full Subtractor using 74153



## b) 8:1 MUX

**Pin configuration**



**Truth Table**

| Active low Enable input | Select Inputs | | | Outputs | |
|---|---|---|---|---|---|
| **EN** | **A** | **B** | **C** | **Y** | $\overline{\mathbf{Y}}$ |
| 1 | x | x | x | 0 | 1 |
| 0 | 0 | 0 | 0 | $D_0$ | $\overline{D0}$ |
| 0 | 0 | 0 | 1 | $D_1$ | $\overline{D1}$ |
| 0 | 0 | 1 | 0 | $D_2$ | $\overline{D2}$ |
| 0 | 0 | 1 | 1 | $D_3$ | $\overline{D3}$ |
| 0 | 1 | 0 | 0 | $D_4$ | $\overline{D4}$ |
| 0 | 1 | 0 | 1 | $D_5$ | $\overline{D5}$ |
| 0 | 1 | 1 | 0 | $D_6$ | $\overline{D6}$ |
| 0 | 1 | 1 | 1 | $D_7$ | $\overline{D7}$ |

## 4-VARIABLE FUNCTION USING IC 74151(8:1MUX)

**Types of MUX: 1.) 2:1 MUX 2) 4:1 MUX 3) 8:1 MUX 4) 16:1 MUX 5) 32:1 MUX**

**Ex:** Implement the following Boolean function using 8:1 multiplexer.

$F(A, B, C, D) = \sum m(2, 4, 5, 7, 10, 14)$

**Design Using MSB Bit A:**

| | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | |
|---|---|---|---|---|---|---|---|---|---|
| $\overline{A}$ | 0 | 1 | ② | 3 | ④ | ⑤ | 6 | ⑦ | Row 1 |
| A | 8 | 9 | ⑩ | 11 | 12 | 13 | ⑭ | 15 | Row 2 |
| I/P To MUX | 0 | 0 | 1 | 0 | $\overline{A}$ | $\overline{A}$ | A | $\overline{A}$ | |

*Fig: Design Table*

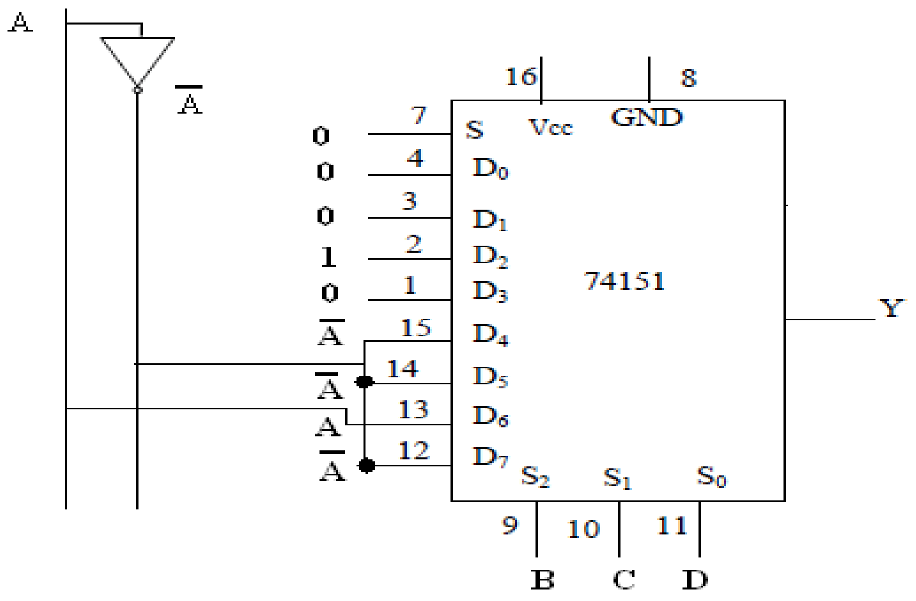
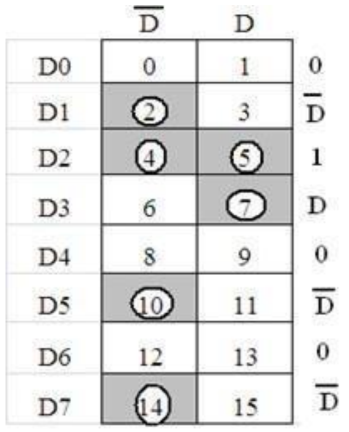
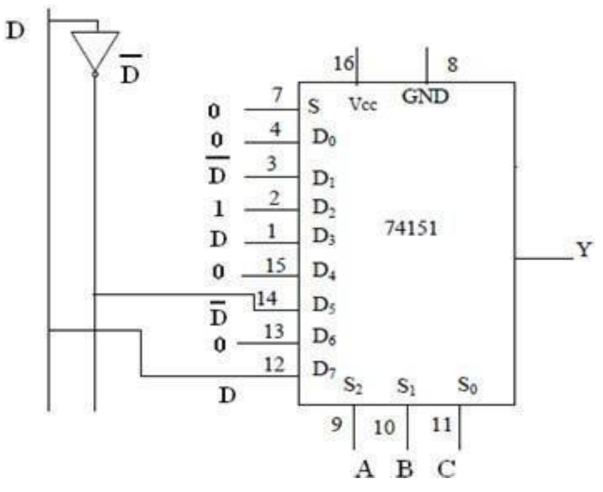


*Fig: Logic Diagram*

## Design Using LSB Bit D:



| | $\overline{D}$ | D | |
|---|---|---|---|
| D0 | 0 | 1 | 0 |
| D1 | 2 | 3 | $\overline{D}$ |
| D2 | 4 | 5 | 1 |
| D3 | 6 | 7 | D |
| D4 | 8 | 9 | 0 |
| D5 | 10 | 11 | $\overline{D}$ |
| D6 | 12 | 13 | 0 |
| D7 | 14 | 15 | $\overline{D}$ |



**PROCEDURE:**

1. The IC is fixed on the IC zip socket and Vcc & Gnd connections aregiven from 5V Supply.
2. Connections are made as shown in the Logicdiagram.
3. All the inputs are connected to the switches & output to the LEDs.
4. Truth table is verified for different combinations of input.

**Result:** ........................................................................................

## Experiment No. 6
## Realize (i) Adder & Subtractors using IC74139.
## (ii) Binary to Gray code conversion & vice-versa (74139)

**Aim:** To Realize Adder/Subtractor/Code Converter using decoder IC74139.

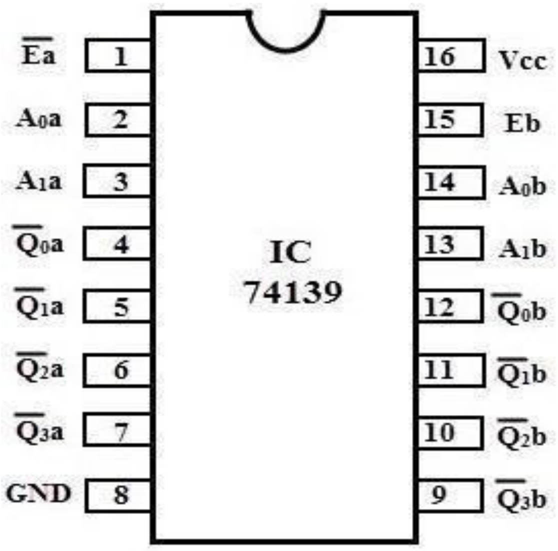
**Components Required**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | IC 74139 | 1 |
| 2 | IC 7400 | 1 |
| 3 | Patch chords | 20 |
| 4 | Trainer Kit | 1 |
| 5 | IC 7404 | 1 |
| 6 | IC 7420 | 2 |

**Theory:**

A decoder is a combinational circuit that connects the binary information from “n” input lines to a maximum of 2n unique output lines. Decoder is also called a min-term generator/maxterm generator. A min-term generator is constructed using AND and NOT gates. The appropriate output is indicated by logic 1 (positive logic). Max-term generator is constructed using NAND gates. The appropriate output is indicated by logic 0 (Negative logic). The IC 74139 accepts two binary inputs and when enable provides 4 individual active low outputs. The device has 2 enable inputs (Two active low).

**Pin Diagram:**

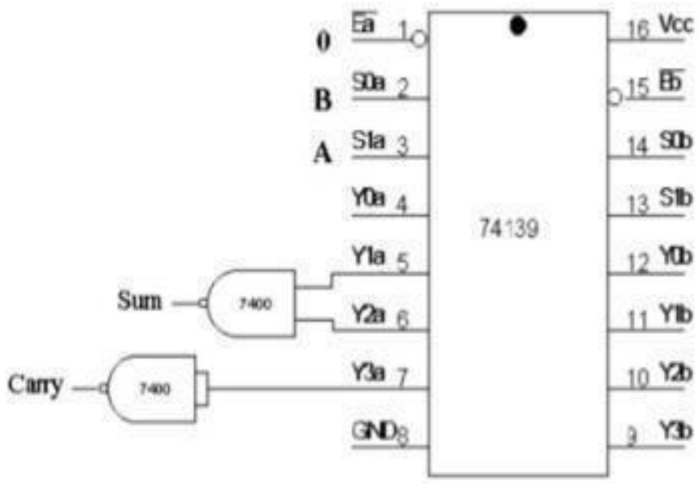


## ADDER & SUBTRACTORS USING IC 74139

### 1:4 DEMUX USING 74139 IC

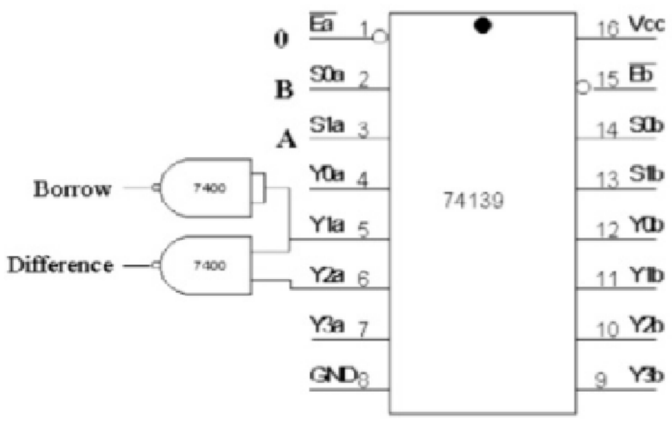
It is a dual 1:4 Decoder ( 2 line to 4 line Decoder) & O/P is inverted I/P.

- **<u>Half Adder using 74139</u>**
- **<u>Minterm of Sum (A,B)=m{1,2}, Minterm of Carry (A,B)=m{3}</u>**



- **<u>Half Subtractor using 74139</u>**

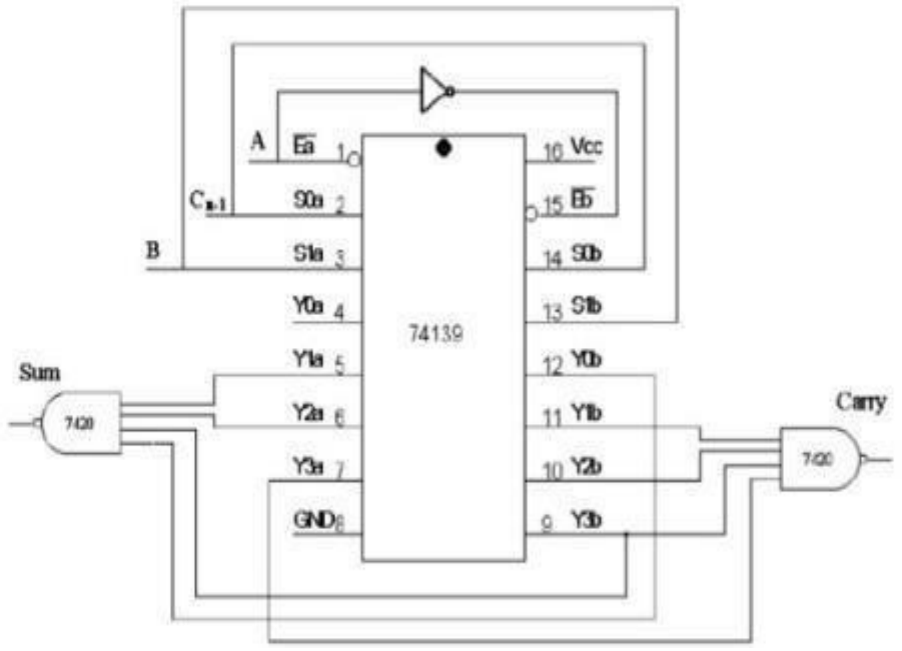
- **<u>Minterm of diff (A,B)=m{1,2}, Minterm of borrow {A,B}=m{1}</u>**



**Truth Tables:**

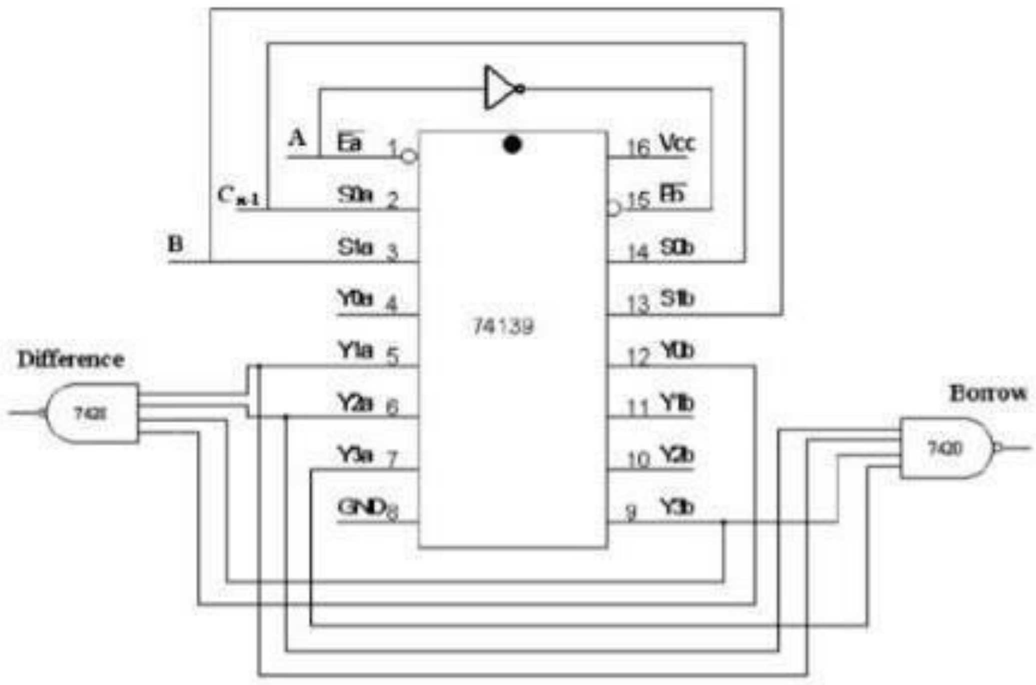
| Inputs | | Half Adder Outputs | | Half Subtractor Outputs | |
|---|---|---|---|---|---|
| A | B | Sum | Carry | Diff | Borrow |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |

**Full Adder Using 74139**

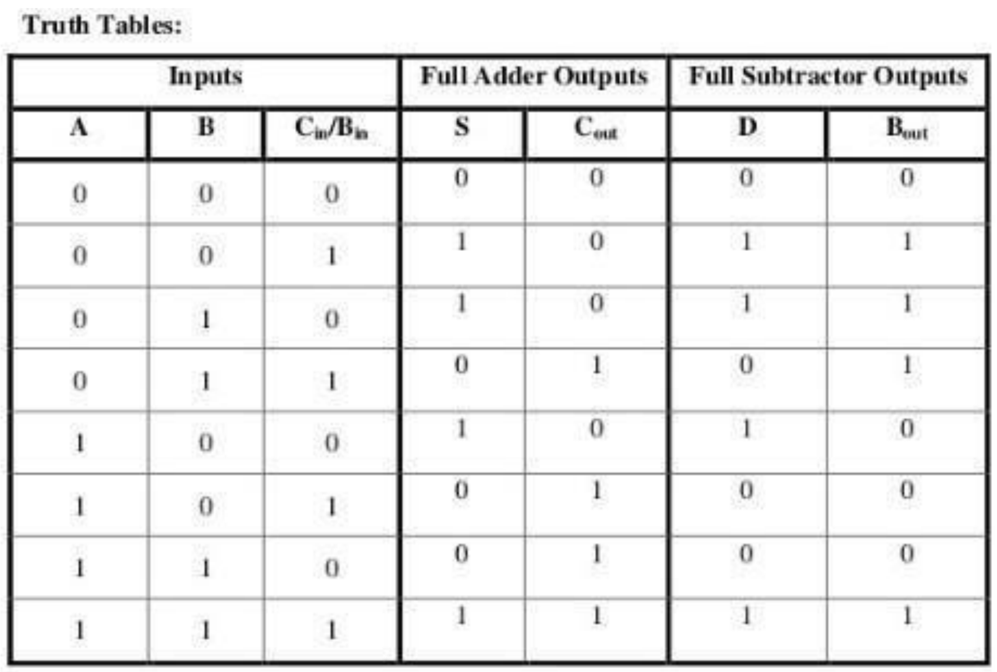


- **Minterm of Sum (A,B,Cin)=m(1,2,4,7)**
- **Minterm of Carry (A,B,Cin)=m(3,5,6,7)**

**Full Subtractor Using 74139**



- **Minterm of diff (A,B,Bin)=m{1,2,4,7}**
- **Minterm of borrow {A,B,Bin}=m{1,2,3,7}**



**Truth Tables:**

| Inputs | | | Full Adder Outputs | | Full Subtractor Outputs | |
|---|---|---|---|---|---|---|
| A | B | $C_{in}/B_{in}$ | S | $C_{out}$ | D | $B_{out}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## (ii) Binary to Gray code conversion & vice-versa (74139)

Realization of 3-bit binary to Gray Code Conversion Using 74139 Logic equations used for converting Binary to Gray code Conversion are

$G2 = B2, \quad G1 = B2 \oplus B1, \quad G0 = B1 \oplus B0$

| Decimal Equivalent | Binary | | | Gray Code | | |
|---|---|---|---|---|---|---|
| | B2 | B1 | B0 | G2 | G1 | G0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 1 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 |
| 5 | 1 | 0 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 | 0 | 0 |

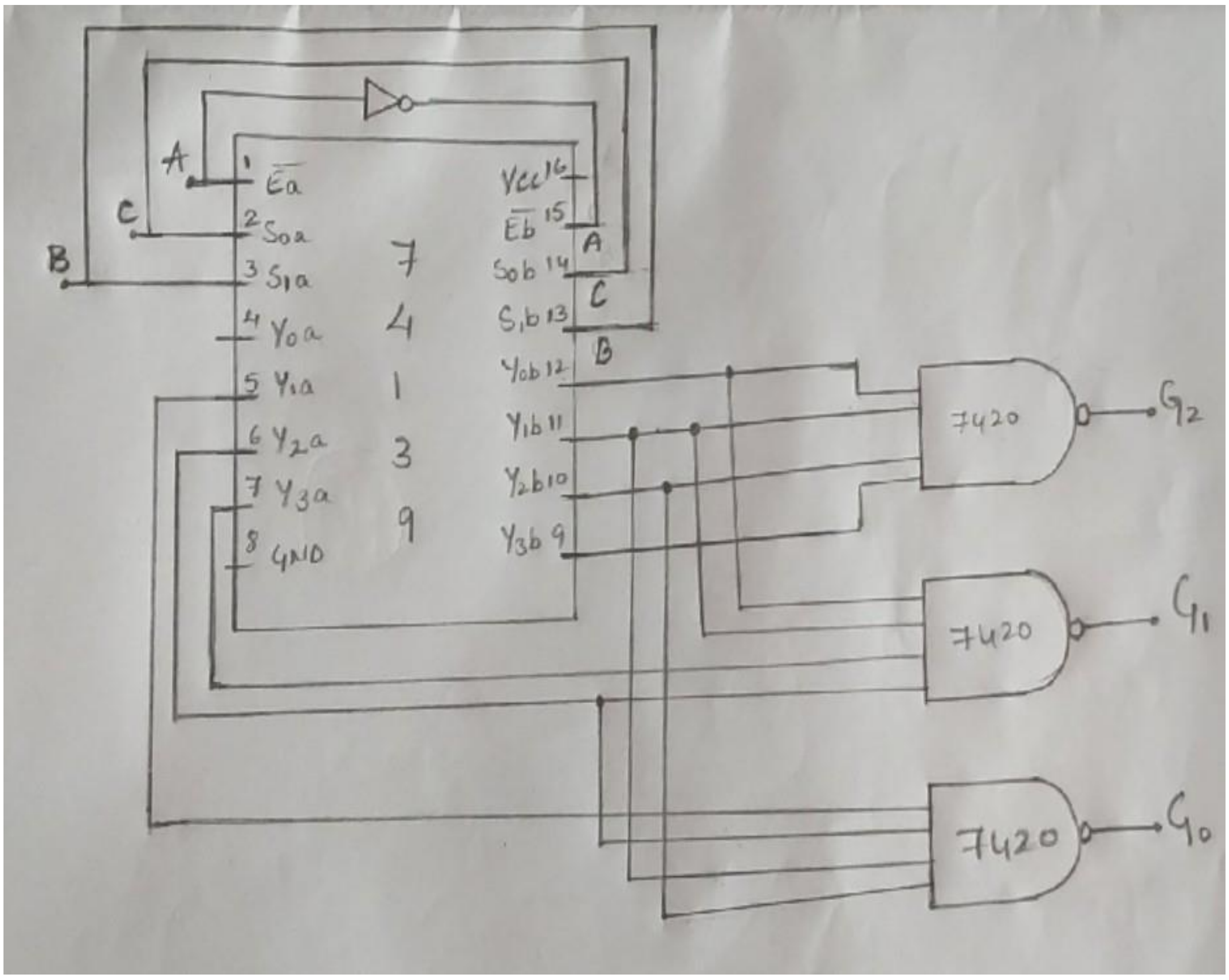
Expressions for G0 G1 & G2 are given by

$G0 = \sum m\ (1,2,5,6) = \{5,6,11,10\}$

$G1 = \sum m\ (2,3,4,5) = \{6,7,12,11\}$

$G2 = \sum m\ (4,5,6,7) = \{12,11,10,9\}$

CIRCUIT:



**Procedure:**

1. The IC is fixed on the IC zip socket and Vcc & Gnd connections are given from 5V Supply.
2. Connections are made as shown in the Logicdiagram.
3. All the inputs are connected to the switches & output to the LEDs.
4. Truth table is verified for different combinations of input.

**Result:** ……………………………………………………………………………………

## Experiment No. 7
## Realize the following flip-flops using NAND Gates.
## Master-Slave JK, D & T Flip-Flop

**Aim:** a) To realize Master-Slave JK, D & T Flip-Flops using NAND Gates.

## Components Required

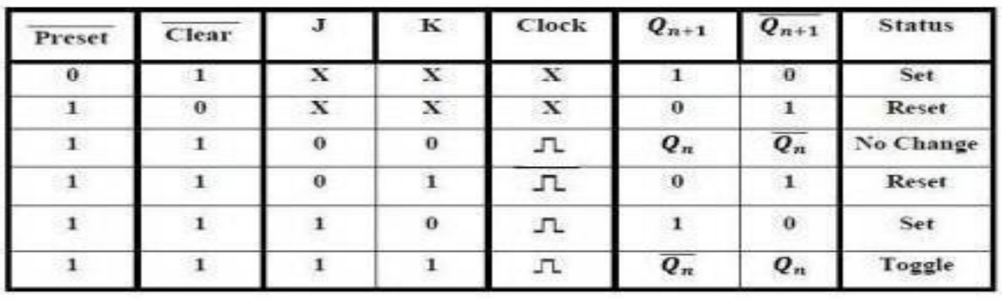
| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | IC 7410 | 2 |
| 2 | IC7400 | 2 |
| 3 | Patch chords | 20 |
| 4 | Trainer Kit | 1 |

**Theory:**

In electronics, a flip-flop or latch is a circuit that has two stable states and can be used to store state information The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. It is the basic storage element in sequential logic. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops can be divided into common types: the SR ("set-reset"), D ("data" or "delay"), T ("toggle"), and JK (Jack Kilby).

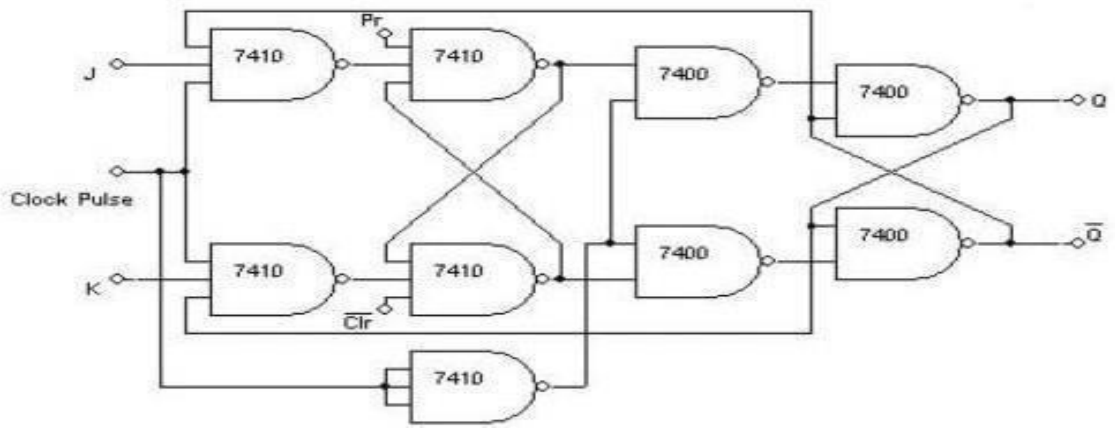
- **Master Slave JK Flip Flop:**

**Truth Table:**



| $\overline{Preset}$ | $\overline{Clear}$ | J | K | Clock | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | Status |
|---|---|---|---|---|---|---|---|
| 0 | 1 | X | X | X | 1 | 0 | Set |
| 1 | 0 | X | X | X | 0 | 1 | Reset |
| 1 | 1 | 0 | 0 | ⎍ | $Q_n$ | $\overline{Q_n}$ | No Change |
| 1 | 1 | 0 | 1 | ⎍ | 0 | 1 | Reset |
| 1 | 1 | 1 | 0 | ⎍ | 1 | 0 | Set |
| 1 | 1 | 1 | 1 | ⎍ | $\overline{Q_n}$ | $Q_n$ | Toggle |

**Circuit diagram:**

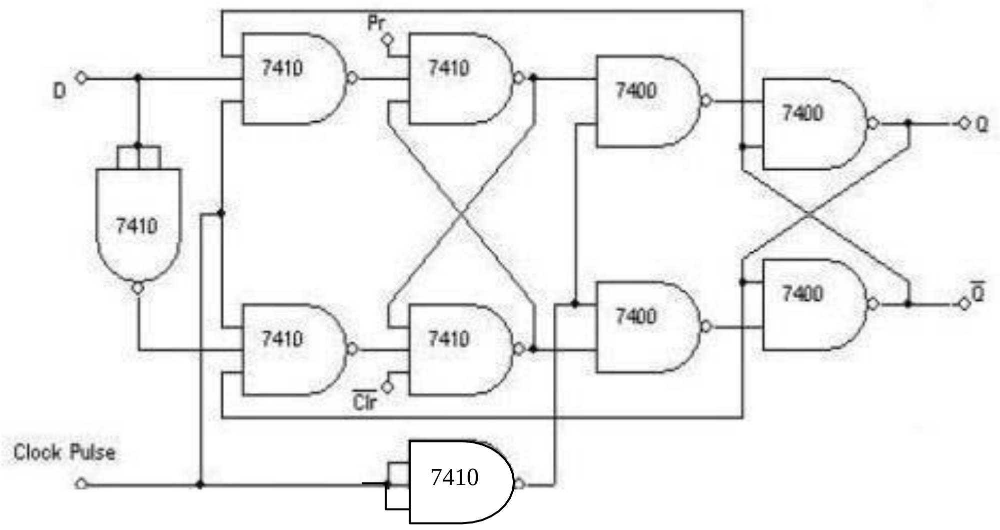


- **D Flip Flop:**

**Truth Table:**

| $\overline{\text{Preset}}$ | $\overline{\text{Clear}}$ | D | Clock | $Q_{n+1}$ | $\overline{Q_{n+1}}$ |
|---|---|---|---|---|---|
| 1 | 1 | 0 | ⎍ | 0 | 1 |
| 1 | 1 | 1 | ⎍ | 1 | 0 |

**Circuit diagram:**

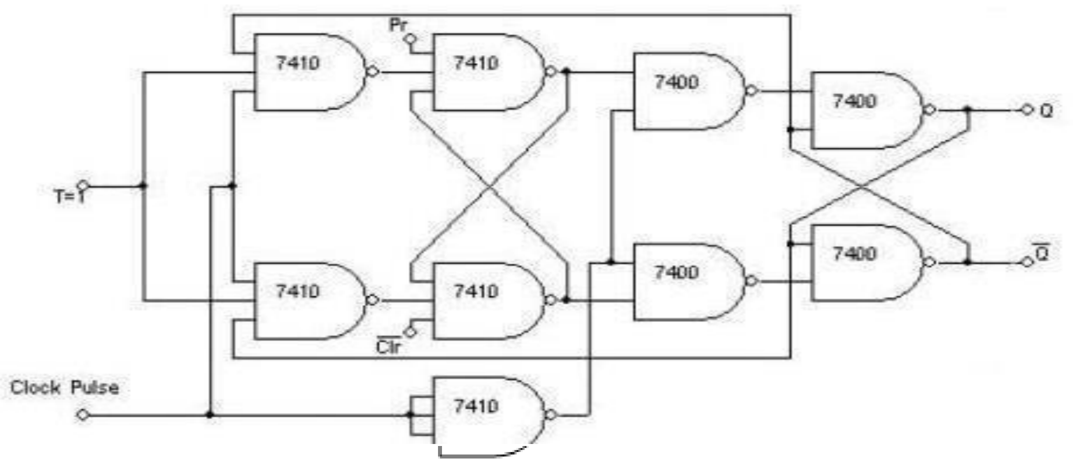


- **T Flip Flop:**

**Truth Table:**

| $\overline{\text{Preset}}$ | $\overline{\text{Clear}}$ | T | Clock | $Q_{n+1}$ | $\overline{Q_{n+1}}$ |
|---|---|---|---|---|---|
| 1 | 1 | 0 | ⎍ | $Q_n$ | $\overline{Q_n}$ |
| 1 | 1 | 1 | ⎍ | $\overline{Q_n}$ | $Q_n$ |

**Circuit diagram:**



**PROCEDURE:**

1. Circuit connections are made as shown in the diagram.
2. For different inputs (J, K, Pr’ and Cr’), note down the outputs Q and Q’ and verify the truth table.
3. To demonstrate the Master-Slave action, connect to clock input.
4. Considering the condition J=K=1, it is clear that output toggles. Hence J and K are connected to High for T FF.
5. Considering the condition K=J’, output will be Q=J=D. Hence to realize a D-FF, D input is connected to J and K=J’ for D FF.
6. The output is noted down and the TTs are verified.

**Result:** ……………………………………………………………………………………

## Experiment No. 8
## Shift registers and Counter.

**Aim:** To Realize the following shift registers using IC7474/IC 7495
(a) SISO (b) SIPO (c) PISO (d) PIPO (e) Ring and (f) Johnson counter.

**Components Required:**

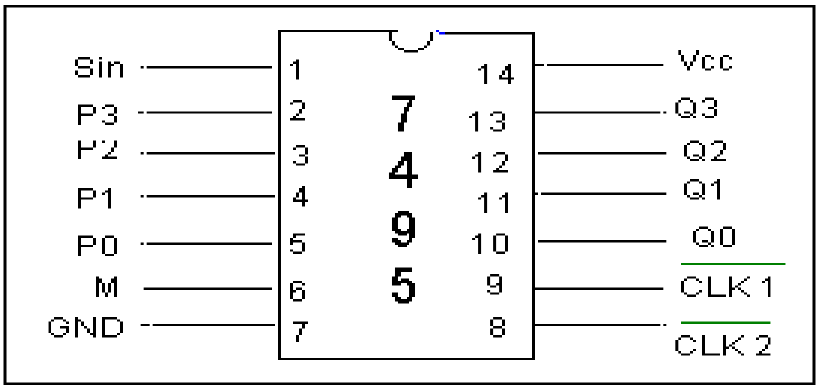
| SI.NO | ICs | Numbers |
|---|---|---|
| 1 | 7495 | 1 |
| 2 | 7404 | 1 |
| 3 | Patch chords | 20 |

**Theory:**

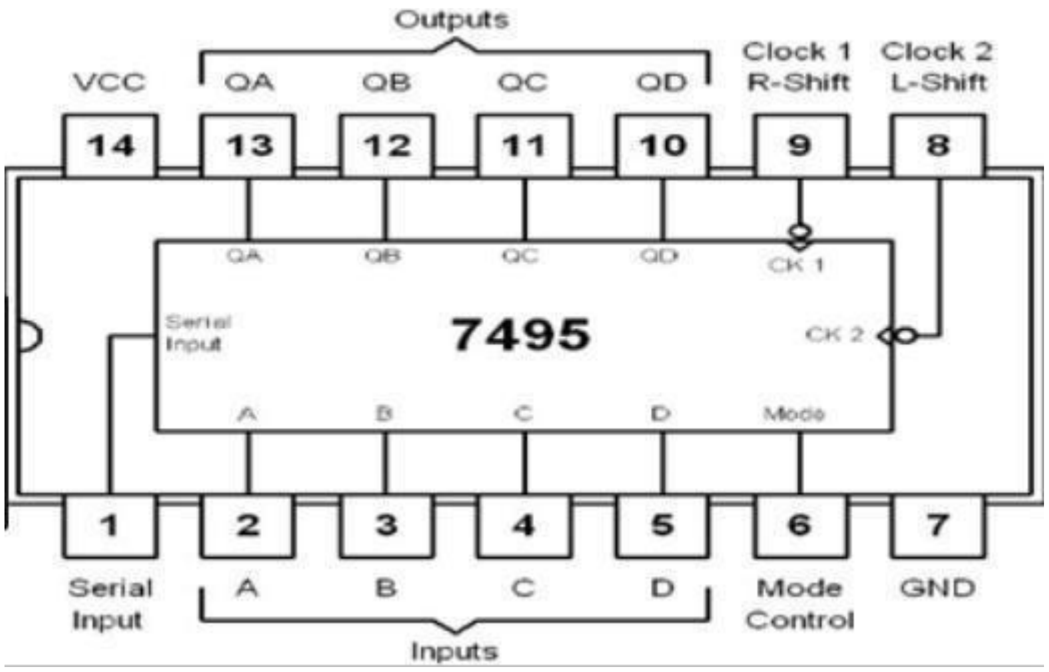
Shift Registers: In digital circuits, a shift register is a cascade of flip flops, sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it, "shifting in" the data present at its input and 'shifting out' the last bit in the array, at each transition of the clock input.More generally, a shift register may be multidimensional, such that its "data in" and stage outputs are themselves bit arrays; this is implemented simply by running several shift registers of the same bit-length in parallel.

Shift registers can have both parallel and serial inputs and outputs. These are often configured as "serial-in, parallel-out" (SIPO) or as "parallel-in, serial-out" (PISO). There are also types that have both serial and parallel input and types with serial and parallel output. There are also "bidirectional" shift registers which allow shifting in both directions: L→R or R→L. The serial input and last output of a shift register can also be connected to create a "circular shift register". One register is PIPO (parallel in parallel out), which is very fast, within single clock pulse, it is giving output.

**Pin configuration of IC 7495:**



## (a) To realize the SISO/SIPO/PISO/PIPO shift registers using IC 7495



**OBSERVATION TABLE: SISO**

| Clock | Time | Sin | O/P's | | | |
|---|---|---|---|---|---|---|
| | | | $Q_3$ | $Q_2$ | $Q_1$ | Serial o/p on $Q_0$ $Q_0$ |
| CP 1 | $t_0$ | 1 | 1 | | | |
| CP 2 | $t_1$ | 0 | 0 | 1 | | |
| CP 3 | $t_2$ | 1 | 1 | 0 | 1 | |
| CP 4 | $t_3$ | 1 | 1 | 1 | 0 | 1 |
| CP 5 | $t_4$ | | | 1 | 1 | 0 |
| CP 6 | $t_5$ | | | | 1 | 1 |
| CP 7 | $t_6$ | | | | | 1 |

**PROCEDURE:**

1. Connections are made as shown in the SISO circuit diagram.
2. Keep the mode control Mode Control = 0
3. The shift register is loaded with 4 bits of data one by one serially.
4. At the end of the 4th clock pulse, the first data 'd0' appears at Q0.
5. Applying another clock pulse will push the second data bit, 'd1' to Q0.
6. Applying yet another clock pulse gives the third data bit, 'd2' at Q0, and so on.

## (b) To realize the SIPO shift registers using IC 7495

**OBSERVATION TABLE: SIPO**

| Clock | Time | Sin | O/P's (Parallel o/p data) | | | |
|---|---|---|---|---|---|---|
| | | | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| CP 1 | $t_0$ | **1** | **1** | | | |
| CP 2 | $t_1$ | **0** | **0** | **1** | | |
| CP 3 | $t_2$ | **1** | **1** | **0** | **1** | |
| CP 4 | $t_3$ | **1** | **1** | **1** | **0** | **1** |

**PROCEDURE:**
1. Connections are made as shown in the SIPO circuit diagram.
2. Keep the Mode Control = 0
3. On applying the first bit of data and then a clock pulse, it can be observed that this data appears at (Q3).
4. Now, applying the second bit of data and a clock pulse, the bit at Q3 shiftsto Q2 and Q3 will be loaded with the new data.
5. This repeats until all 4 data bits are loaded.
6. At the end of the 4th clock pulse, all 4 bits are available at theparallel output pins Q3 through Q0.

## (c) To realize the PIPO shift registers using IC 7495

**OBSERVATION TABLE: PIPO**

| Clock | Time | I/P's | | | | O/P's | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | D3 | D2 | D1 | D0 | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| Clk1 | $t_0$ | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

**PROCEDURE:**
1. Connections are made as shown in the PIPO mode circuit diagram.
2. Mode Control = 1
3. Apply the 4 data bits as input to D3, D2, D1, D0.
4. Apply one clock pulse
5. Note that the 4 bit data at parallel inputs appears at the parallel output pins Q3, Q2, Q1, Q0 respectively.

## (d) To realize the PISO shift registers using IC 7495

### Observation Table: PISO

| Clock | Time | I/P's | | | | O/P's( Serial o/p data) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | D3 | D2 | D1 | D0 | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| Clk1 | $t_0$ | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| Clk2 | $t_1$ | 0 | 0 | 0 | 0 | | 1 | 1 | 0 |
| Clk3 | $t_2$ | 0 | 0 | 0 | 0 | | | 1 | 1 |
| Clk4 | $t_3$ | 0 | 0 | 0 | 0 | | | | 1 |

**PROCEDURE:**
1. Connections are made as shown in the PISO circuit diagram.
2. Mode Control = 1
3. Apply the 4-bit data at the parallel I/P pins D3, D2, D1, and D0 andapply single clock pulse to load the data to all 4 registers.
4. Now make all data bits as 0 and apply clock pulse one by one to get the data bit by bit at the output line.
5. The data applied at the parallel input pins will shift and comes out serially at the output line Q0.

## (e) To realize the Ring Counter using IC 7495

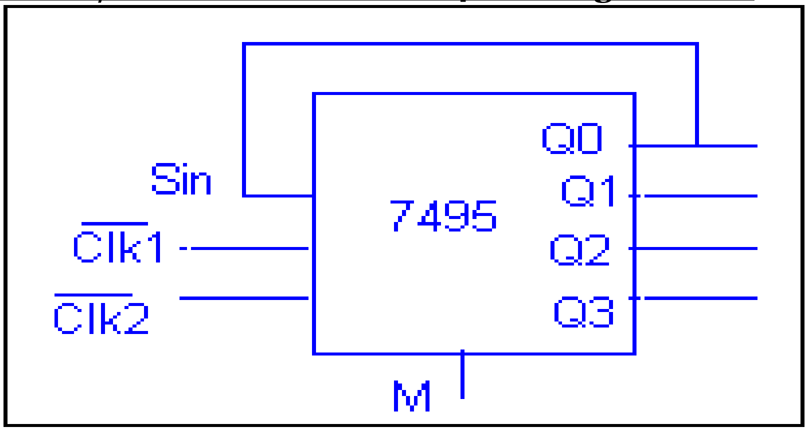
### Theory: Ring and Johnson Counter

Two most important types of shift register counters are Johnson counter and Ring counter. These shift register counters with serial outputs are connected to serial inputs to produce particular pattern of sequences. These shift registers are used as counters because of the specified sequence of states. Ring counter is a basic application of shift registers. It is formed by the feedback of the output to its own input. This counter has N states where N denotes the number of flip-flops in the ring counter.

**Johnson Counter**

Johnson Counter also known as Twisted Ring Counter is another basic application of shift registers with a feedback. Here the feedback is given from the inverted output of the last flip flop to the input of the first flip-flop. Figure below shows a 4-bit Johnson counter. It consists of four flip-flops FF0, FF1, FF2 and FF3. Here the inverted output of the last flip-flop FF3 is given as feedback to the input of the first flip-flop FF0. Here, at first four logic zeros will be passed to the flip-flops. When clock pulses are given "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" outputs will be obtained and the sequence will repeat for the next clock pulses. More outputs than ring counter.

**LOGIC DIAGRAM/TIMING DIAGRAM: 4-bit Ring Counter**



**OBSERVATION TABLE:**

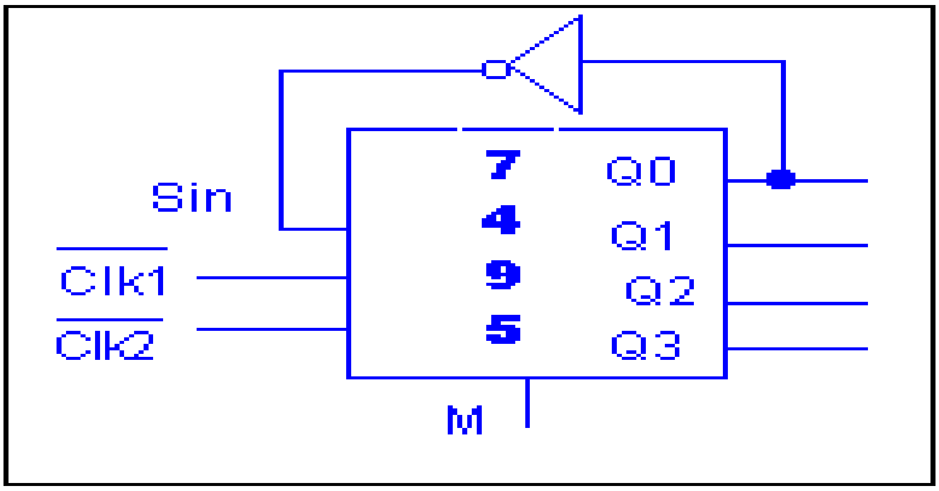
| Clock | Time | O/P's | | | |
|---|---|---|---|---|---|
| | | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| Clk1 | $t_0$ | 1 | 0 | 0 | 0 |
| Clk2 | $t_1$ | 0 | 1 | 0 | 0 |
| Clk3 | $t_2$ | 0 | 0 | 1 | 0 |
| Clk4 | $t_3$ | 0 | 0 | 0 | 1 |
| Clk5 | $t_4$ | 1 | 0 | 0 | 0 |

## 1. Ring Counter: - Procedure

1. Mode control is made 1
2. Parallel i/p's say 1000 at P3 P2 P1 P0 is given.
3. Clk 2 is pulsed once. Now data 1000 comes at o/p lines Q3 Q2 Q1 Q0 respectively.
4. Now mode control is set to 0 & clock 1 is connected to monopulse.
5. Now when the clock pulses are applied, '1'at the o/p circulates in the form of aring.

## (f) To realize the Johnson Counter using IC 7495

### LOGIC DIAGRAM: Johnson Counter



### OBSERVATION TABLE:

| Clock | Time | O/P's | | | |
|---|---|---|---|---|---|
| | | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| Clk1 | $t_0$ | 0 | 0 | 0 | 0 |
| Clk2 | $t_1$ | 1 | 0 | 0 | 0 |
| Clk3 | $t_2$ | 1 | 1 | 0 | 0 |
| Clk4 | $t_3$ | 1 | 1 | 1 | 0 |
| Clk5 | $t_4$ | 1 | 1 | 1 | 1 |
| Clk6 | $t_5$ | 0 | 1 | 1 | 1 |
| Clk7 | $t_6$ | 0 | 0 | 1 | 1 |
| Clk8 | $t_7$ | 0 | 0 | 0 | 1 |
| Clk9 | $t_8$ | 0 | 0 | 0 | 0 |

### PROCEDURE:

1. Connect the circuit as shown below.
2. Mode selection i/p is 0
3. Clk 1 is connected to the monopulse.
4. Observe the count sequence as given in the Truth Table.

**Result:** ..................................................................................................

## Experiment No. 9

**Realize (i) Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop (ii) Mod-N Counter using IC7490 / 7476 (iii) Synchronous counter using IC74192**

**Aim:** To realize Realize the following
(a) Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop
(b) Mod-N Counter using IC7490 / 7476
(c) Synchronous counter using IC74192

**COMPONENTS REQUIRED:**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1. | IC 7490 | 1 |
| 2. | IC74192 | 1 |
| 3. | IC 7476 | 1 |
| 4. | IC 7420 | 1 |
| 5. | IC 7408 | 1 |
| 6. | Patch chords | 20 |
| | Trainer Kit | 1 |

### Theory:

Synchronous Counters can be made from Toggle or D-type flip-flops. Synchronous counters are easier to design than asynchronous counters. are all clocked together at the same time with the same clock signal. Due to this common clock pulse all output states switch or change simultaneously.

### (i) Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop

**Count Table: -**

| CLK | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 |

Note: $Q_{n+1}$ is present state
$Q_n$ is previous state

**TT of JK flip flop**

| J | K | $Q_{n+1}$ | Comment |
|---|---|---|---|
| 0 | 0 | $Q_n$ | N C |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | $\overline{Q_n}$ | Toggle |

## State table and Excitation Table: -

**FF Excitation table**

| Present State | | | Next State | | | Excitation Table | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_1$ | $K_1$ | $J_2$ | $K_2$ | $J_3$ | $K_3$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | x | 1 | x | 0 | x |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | x | x | 0 | 0 | x |
| 0 | 1 | 1 | 1 | 1 | 0 | x | 1 | x | 0 | 1 | x |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | x | x | 1 | x | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | x | 0 | 0 | x | x | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | x | 1 | 0 | x | 0 | x |

**State Table**

| $Q_n$ | $Q_{n+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

**K – Maps:**

J1

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | x | x | 1 |
| 1 | x | x | x | 1 |

J1=Q2

K1

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | x | 1 | 1 | x |
| 1 | x | 0 | x | x |

K1=$\overline{Q3}$

J2

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | x | x |
| 1 | x | 0 | x | x |

J2=$\overline{Q1}$

K2

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | 0 | 0 |
| 1 | X | X | X | 1 |

K2=Q3

J3

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | X | X | X | X |

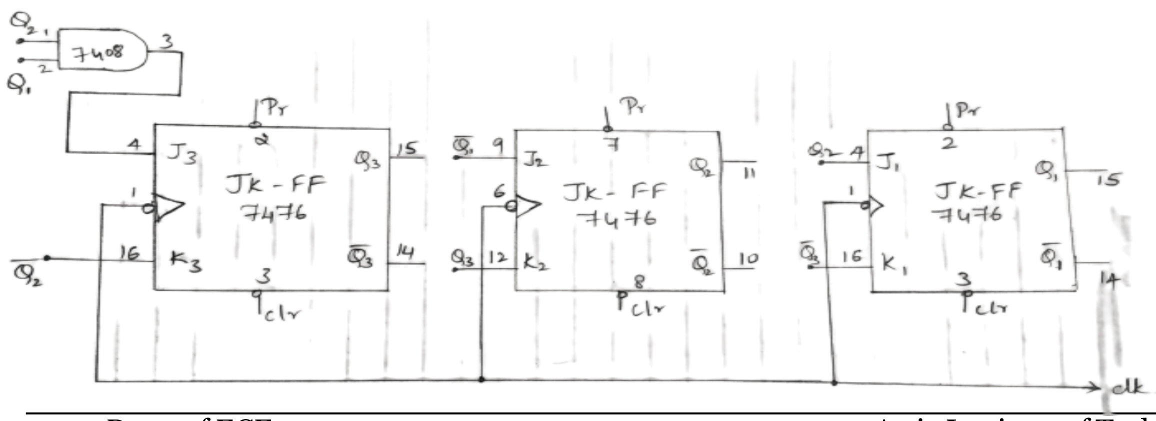
J3=Q2*Q1

K3

| Q3 \ Q2 Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | X | X |
| 1 | X | 1 | X | 0 |

K3=Q2

## Circuit Diagram: -



**Procedure: -**

1. Make connections as shown in logic diagram
2. Provide power supply to trainer kit
3. Observe the count sequence as per the count table
4. Observe the waveforms on CRO by giving 1 KHz TTL clocks as clock input.

ii) Realize Mod-N Counter using IC7490 / 7476

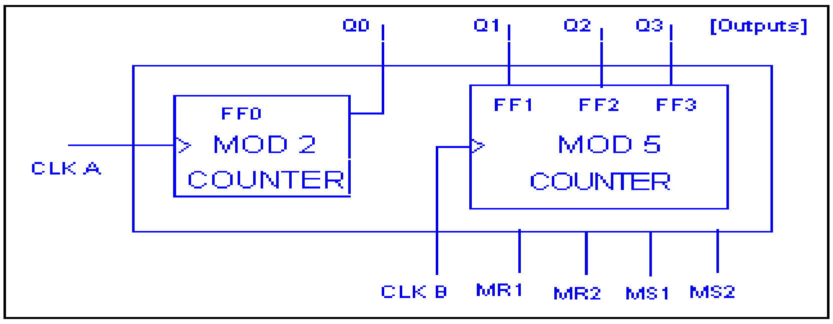
**<u>B) DECADE COUNTER USING 7490 : -</u>**

- IC 7490 is a divide by 10 counter using four Master Slave FFs. It contains one divided by two & four divided by five counters which can be cascaded to give a divide by ten counter.
- MR1 and MR2 are the Reset inputs, which when enabled, clear all the FFs.
- MS1 and MS2 are the Set inputs, which when enabled, make all the flip flops outputs go high.

**LOGIC DIAGRAM OF 7490-RIPPLE COUNTER: -**

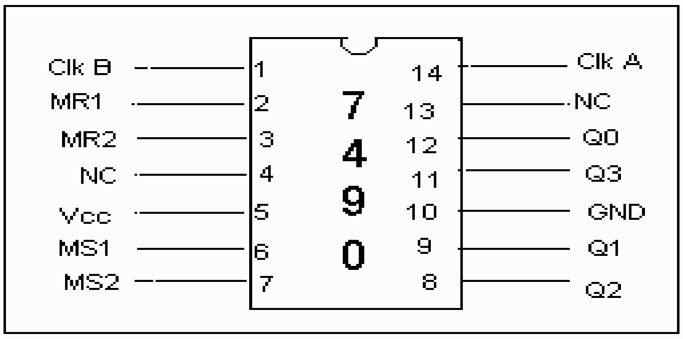
## Pin diagram: -

**Note: 1) MR1&MR2 are RESET inputs, 2) MS1 & MS2 are set inputs**



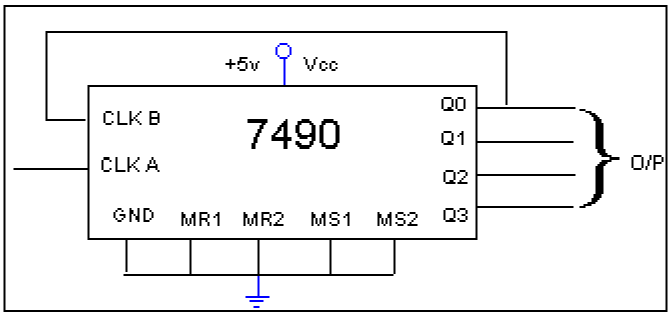
## Mode Selection Table: -

| Reset I / P MR1 MR2 | Set I/P MS1 MS2 | Output $Q_3$ $Q_2$ $Q_1$ $Q_0$ |
|---|---|---|
| 1 1 | 0 X | 0 0 0 0 Reset |
| 1 1 | X 0 | 0 0 0 0 |
| X X | 1 1 | 1 0 0 1 Set to 9 |
| At least one of the I/Ps is 0 | At least one of the I/Ps is 1 | Count mode |



Count Table: -

| CLK | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 |



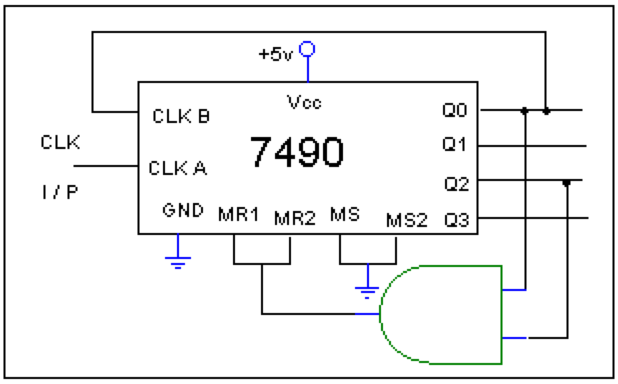
**B)** **MOD – N COUNTER USING 7490 (N=5)**

**Decoding logic**

**Count Table: -**

| CLK | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 |

**Logic Diagram: -**



**O/P waveforms: -**

Similar to above but the counter resets after 5th clock pulse.

## Procedure for (a) and (b) : -

1. To start the count from $Q_3$ $Q_2$ $Q_1$ $Q_0$ = 0 0 0 0, MS1 and MS2 inputs are connected to GND, MR1 and MR2 are connected to GND.
2. The other connections are made as shown in the ckt diagrams.
3. Count sequence is observed on LED's giving monopulses.
4. Output waveforms are observed on the CRO by giving a square wave of 1KHz frequency asthe clock.

**Result: -**

Ripple counter & MOD – N counter are realized using 7490 IC & the count sequence is verified.

iii) Realize Synchronous counter using IC74192

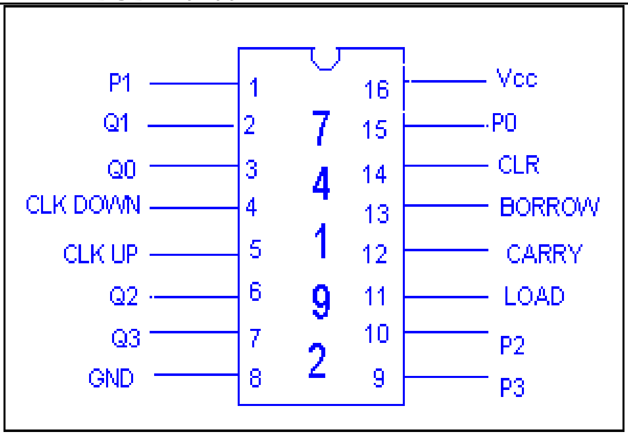
Note: **-74192 is a synchronous 4 – bit decade up / down counter with preset and clear**

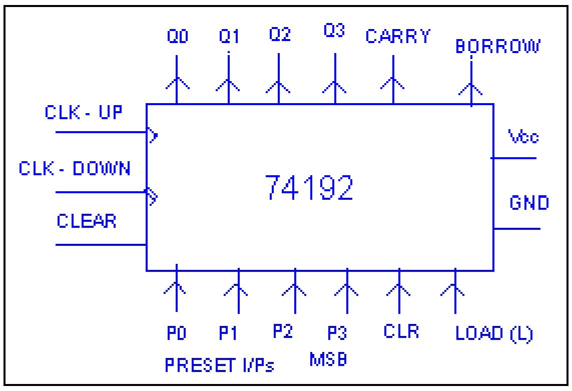
**IC's required: -**

74192, 7404, 7420

## Logic diagram of 74192: -

## Pin details of 74192 IC: -





| Load | Clear | Clk–up | Clk–down | Mode |
|---|---|---|---|---|
| x | 1 | x | x | Preset to zero |
| 1 | 0 | ↑ | 1 | Up – count |
| 1 | 0 | 1 | ↑ | Down count |
| 0 | 0 | x | x | Preset |
| 1 | 0 | 1 | 1 | Stop count |

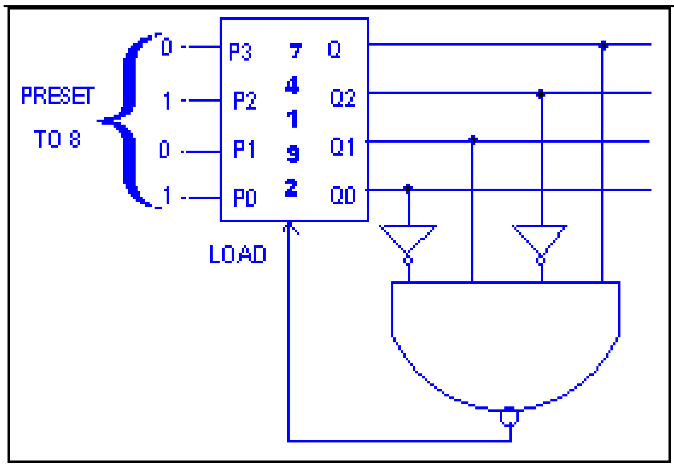
**(i) ÷ 10 UP COUNTER & DOWN COUNTER**

- Clear the counter
- Keep clear = 0, load = 1.
- Clk to monopulse.
- Observe the count sequence at $Q_3$ $Q_2$ $Q_1$ $Q_0$ for many monopulse. This gives up count.
- For down count, clear the counter.
- Keep clear = 0, load = 1.
- CLK to monopulse.
- Observe the count sequence at the o/p at $Q_3$ $Q_2$ $Q_1$ $Q_0$.

## (II) 74192 AS PRESETTABLE UP COUNTER: -

**Circuit diagram: -**

**Clr =0**



**Procedure: -**

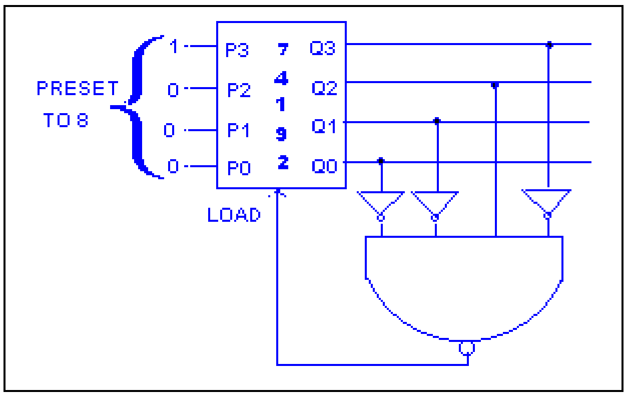
To count from 5 to 9

1. Preset the data input to 5.
2. Arrange for 'count – up' mode.
3. Using the external logic shown, the o/p count changes from 5 to 9, 5 to 9 & so on.

## (III) PRESETTABLE DOWN COUNTER: -

**Circuit diagram: -**

Clr =0



**Procedure: -**

**To count from 8 to 5**

1. Preset data input to 8.
2. Range for 'count – down ' mode.
3. Using the external logic shown, the output count sequence changes as follows: 8, 7, 6, 5, 8, 7, 6, 5 ----etc.

**Result: - -----------------------------------------------------------------**

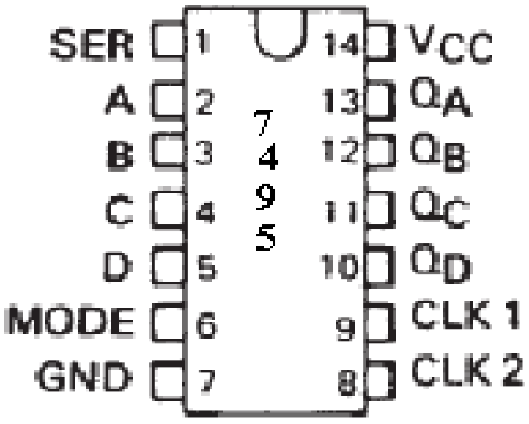
## Experiment No.10
## Pseudo Random Sequence generator

**Aim:** A) To Design Pseudo Random Sequence generator using 7495.

**COMPONENTS REQUIRED:**

| SI.NO | ICs | Numbers |
|---|---|---|
| 1. | IC 7495 | 1 |
| 2. | IC 7486 | 1 |
| 3. | Patch chords | 20 |
| 4. | Trainer Kit | 1 |

**Pin diagram**



**PROCEDURE:**
- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- By Keeping mode=1. Load the input A,B,C,D as in Truth Table 1st Row andgive a clock pulse
- For count mode make mode = 0.
- Verify the Truth Table and observe the outputs.

**DESIGN 1:**
Sequence = 100010011010111
Sequence length S = 15

**Truth Table:**

| Map Value | Clock | $Q_A$ | $Q_B$ | $Q_C$ | $Q_D$ | O/p D |
|---|---|---|---|---|---|---|
| 15 | 1 | 1 | 1 | 1 | 1 | 0 |
| 7 | 2 | 0 | 1 | 1 | 1 | 0 |
| 3 | 3 | 0 | 0 | 1 | 1 | 0 |
| 1 | 4 | 0 | 0 | 0 | 1 | 1 |
| 8 | 5 | 1 | 0 | 0 | 0 | 0 |
| 4 | 6 | 0 | 1 | 0 | 0 | 0 |
| 2 | 7 | 0 | 0 | 1 | 0 | 1 |
| 9 | 8 | 1 | 0 | 0 | 1 | 1 |
| 12 | 9 | 1 | 1 | 0 | 0 | 0 |
| 6 | 10 | 0 | 1 | 1 | 0 | 1 |
| 11 | 11 | 1 | 0 | 1 | 1 | 0 |
| 5 | 12 | 0 | 1 | 0 | 1 | 1 |
| 10 | 13 | 1 | 0 | 1 | 0 | 1 |
| 13 | 14 | 1 | 1 | 0 | 1 | 1 |
| 14 | 15 | 1 | 1 | 1 | 0 | 1 |
|  |  |  | 1 | 1 | 1 |  |
|  |  |  |  | 1 | 1 |  |
|  |  |  |  |  | 1 |  |

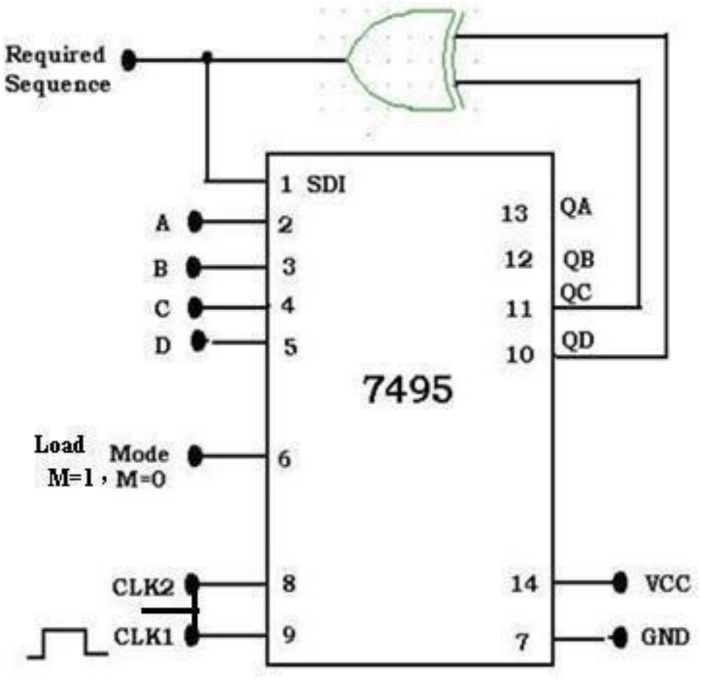
**Karnaugh Map:**

FOR D

| QAQB \ QCQD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 0 | 1 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |

$D = QC \oplus QD$

**Circuit diagram:**



## DESIGN 2:

Sequence = 1001011
Sequence length S = 7

### Truth Table:

| Map Value | Clock | QA | QB | QC | QD | O/P D |
|---|---|---|---|---|---|---|
| 14 | 1 | 1 | 1 | 1 | 0 | 0 |
| 7 | 2 | 0 | 1 | 1 | 1 | 0 |
| 3 | 3 | 0 | 0 | 1 | 1 | 1 |
| 9 | 4 | 1 | 0 | 0 | 1 | 0 |
| 4 | 5 | 0 | 1 | 0 | 0 | 1 |
| 10 | 6 | 1 | 0 | 1 | 0 | 1 |
| 13 | 7 | 1 | 1 | 0 | 1 | 1 |

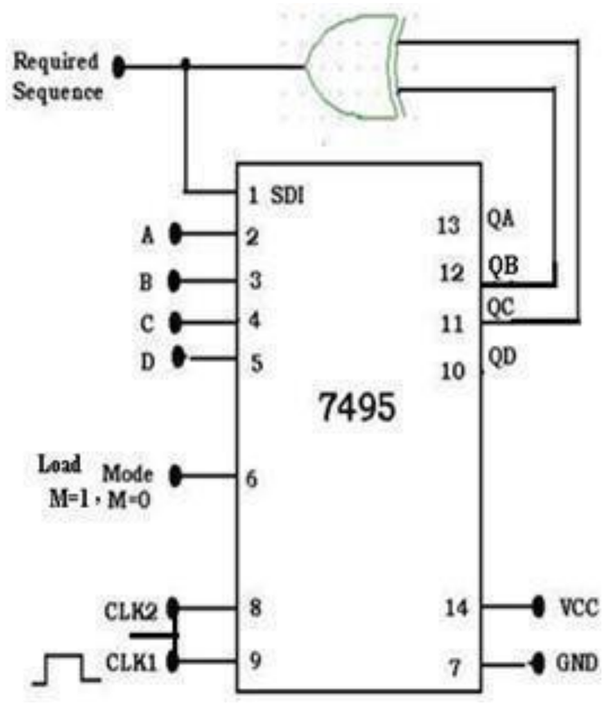
### Karnaugh Map:

FOR D

| QAQB \ QCQD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | X | 1 | X |
| 01 | 1 | X | 0 | X |
| 11 | X | 1 | X | 0 |
| 10 | X | 0 | X | 1 |

D= QB ⊕ QC

### Circuit diagram:



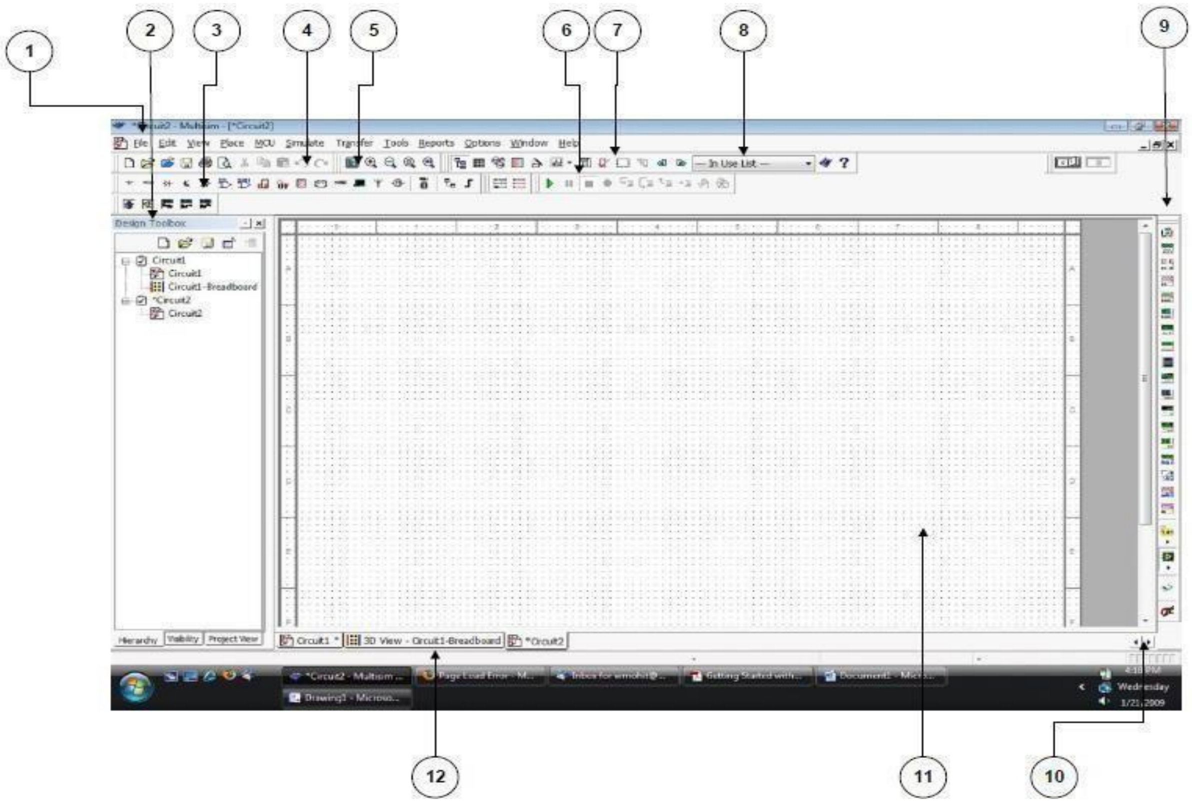
**Result:** ..........................................................................................................

## Experiment No.11

## AIM: Serial Adder with Accumulator using simulation tool

## INTRODUCTION TO MULTISIM

Multisim is the schematic capture and simulation application of National Instruments Circuit Design Suite, a suite of EDA (Electronic Design Automation) tools. It is similar to PSpice, but it is more easy to use in practical sense and has lots of features to make circuit drawing/simulating, a really simple task. Here is window of multisim, as it appears first time when you start the software.

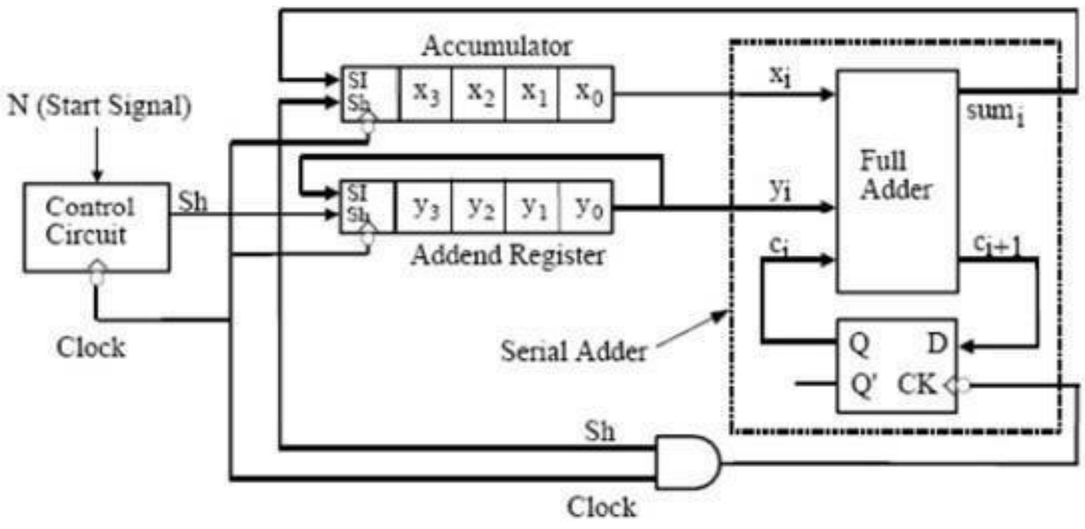


1. The Menu Bar is where you find commands for all functions.
2. The Design Toolbox lets you navigate through the different types of files in a project (schematics, PCBs, reports), view a schematic's hierarchy and show or hide different layers.
3. The Component toolbar contains buttons that let you select components from the Multisim databases for placement inyour schematic.
4. The Standard toolbar contains buttons for commonly-performed functions such as Save, Print, Cut, and Paste.
5. The View toolbar contains buttons for modifying the way the screen is displayed.
6. The Simulation toolbar contains buttons for starting, stopping, and other simulation functions.
7. The Main toolbar contains buttons for common Multisim functions.
8. The In Use List contains a list of all components used in the design.

9. The Instruments toolbar contains buttons for each instrument.
10. Scroll Left –right is to ensure ease in handling larger designs.
11. The Circuit Window (or workspace) is where you build your circuit.
12. Active tab indicates the current active circuit window.

Theory:

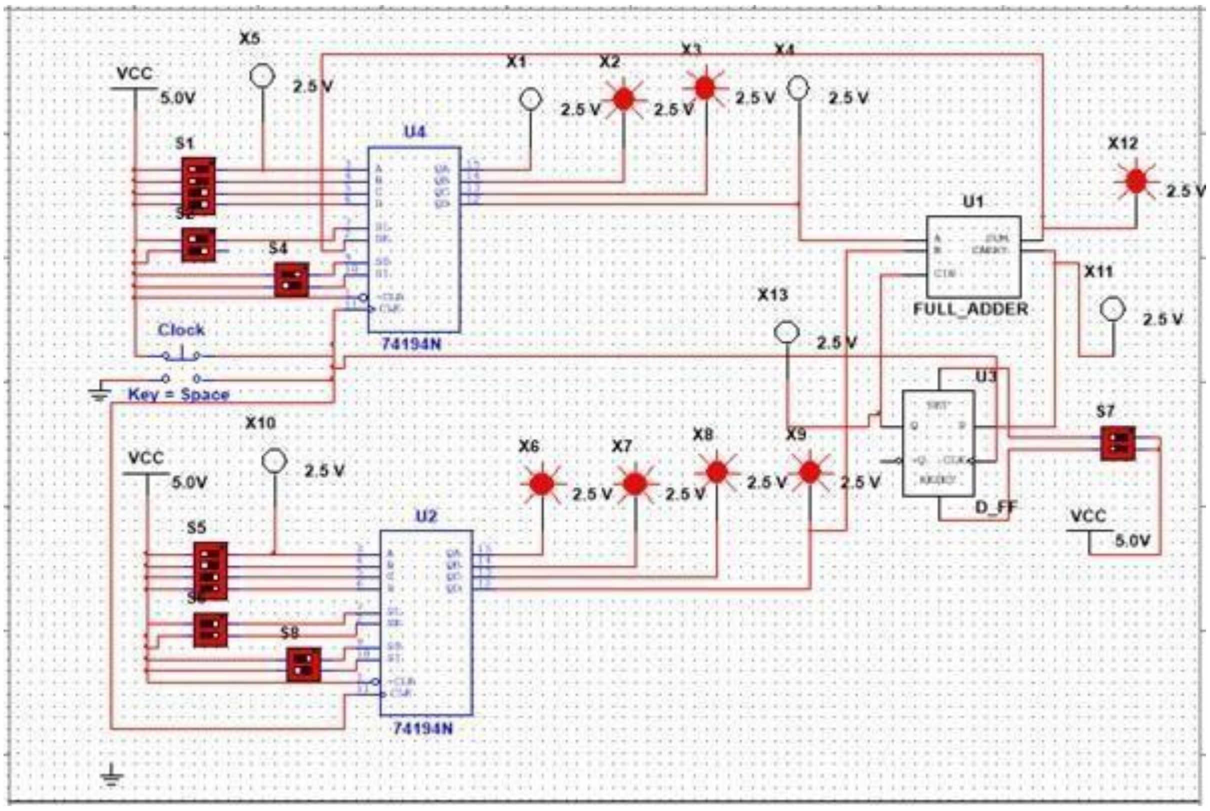
Adders are very critical components in digital systems because of the fact they are extensively used in fundamental digital operations such as multiplication, division, and subtraction. A serial adder with accumulator is an important digital electronic component used extensively in addition operations. Serial binary addition is a binary operation that is achieved using a flip-flop and a full adder. The carry-out signal from the full adder is fed to the flip-flop on each clock cycle. The serial adder with accumulator shown in figure is a special type of adder that uses two shift registers, a D flip-flop, a full adder, and a control circuit to achieve serial binary addition.



The full adder is used to perform bit by bit addition and D-Flip flop is used to store the carry output generated after addition. This carry is used to carry input for the next addition. Initially the D Flip flop is cleared and addition starts with the least significant bits of both register. After each clock pulse data within the right shift registers are shifted right 1-bit and we get from next digit and carry of precious addition as new inputs for the full adder. Consider two 4bit numbers X=0101 and Y= 0111. The operation at each instant is given below.

| | X | Y | $c_i$ | $sum_i$ | $c_{i+1}$ |
|---|---|---|---|---|---|
| $t_0$ | 0101 | 0111 | 0 | 0 | 1 |
| $t_1$ | 0010 | 1011 | 1 | 0 | 1 |
| $t_2$ | 0001 | 1101 | 1 | 1 | 1 |
| $t_3$ | 1000 | 1110 | 1 | 1 | 0 |
| $t_4$ | 1100 | 0111 | 0 | (1) | (0) |

**Circuit Diagram:**



**Procedure:**

1. Connect the circuit as shown in figure using simulation tool.
2. As a example A=1000, B=0000 (A+B).
3. Clear D-flipflop ,by making Set=0,Reset=1 . Apply one clock pulses, than make Set=0,Reset=0.
4. To Shift the register: Make SR= 1 and SL=0 for both the shift register.
5. To load the data : Make S0=1 and S1=1 for both the shift register.
6. Apply one clock pulses.
7. Data will be loaded at output of shift register(Indicators).
8. For shifting the data : Change S1=0 for both shift register and apply 4 clock pulses.
9. Check for the second shift register until you get back the second number (B).
10. Check the sum at the output of first shift register .
11. Carry out at the Full adder output..
12. If any carry is generated it will be stored in the D FF and get added at the next clock pulse.

**Result:** ..........................................................................................

# Experiment No: 12
## Binary Multiplier using simulation tool

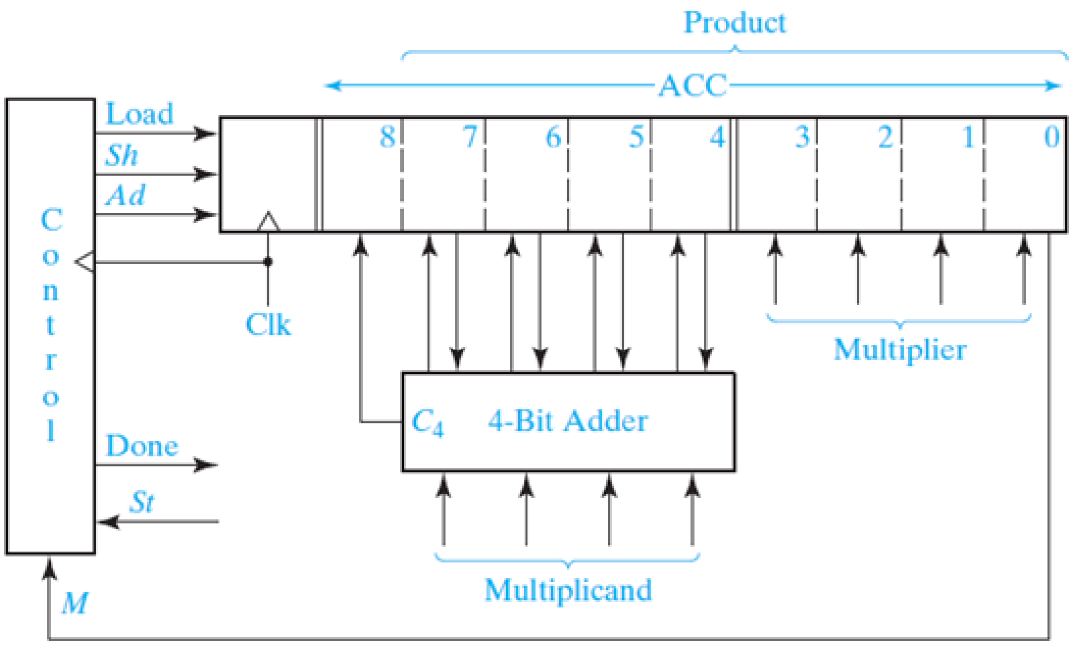
**Aim:**Design Binary Multiplier and Simulate using Simulation tool.

**Theory:**

Binary multiplication requires only shifting and adding. The following example shows how each partial product is added in as soon as it is formed. This eliminates the need for adding more than two binary numbers at a time.

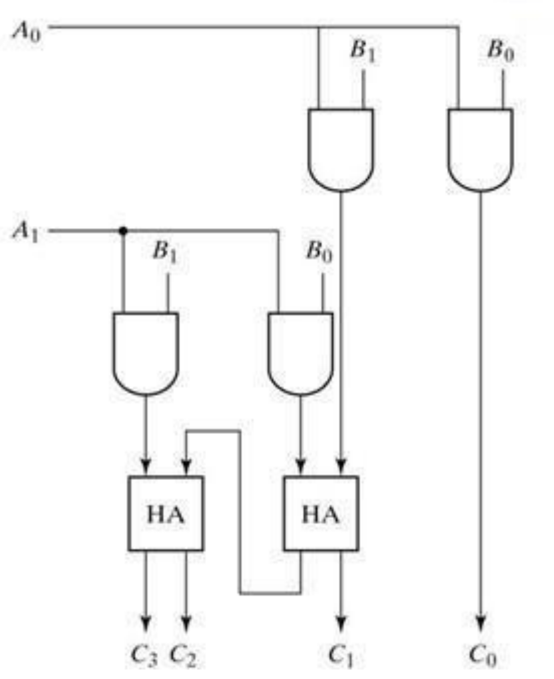
```
Multiplicand ——→      1101      (13)
Multiplier   ——→      1011      (11)
                      1101
                     1101
Partial   }          100111
Products  }          0000
                     100111
                    1101
Product      ——→    10001111    (143)
```

The multiplication of two 4-bit numbers requires a 4-bit multiplicand register, a 4-bit multiplier register, and an 8-bit register for the product. The product register serves as an accumulator to accumulate the sum of the partial products. Instead of shifting the multiplicand left each time before it is added, as was done in the previous example, it is more convenient to shift the product register to the right each time.



**Circuit Diagram:**

$$\begin{array}{cccc} & & B_1 & B_0 \\ & & A_1 & A_0 \\ \hline & & A_0B_1 & A_0B_0 \\ & A_1B_1 & A_1B_0 & \\ \hline C_3 & C_2 & C_1 & C_0 \end{array}$$



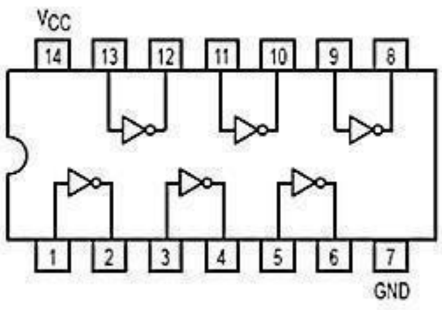
**Procedure:**

1. Connect the circuit as shown in figure using simulation tool.
2. Load the2 bit inputs A(S1) and B(S4) through switches
3. View the output at P3P2P1P0

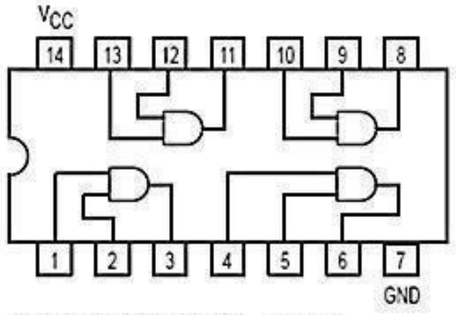
**Result:** ..........................................................................................

## IC PIN DETAILS

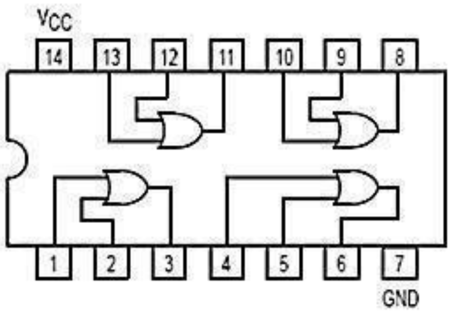
**Inverter (NOT Gate) - 7404LS**



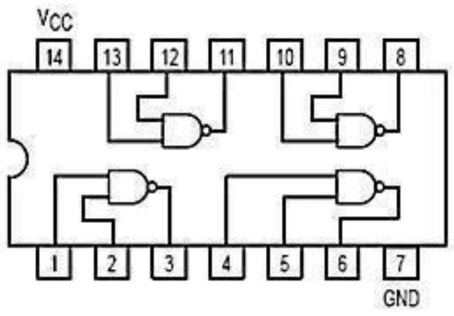
**2-Input AND Gate - 7408LS**



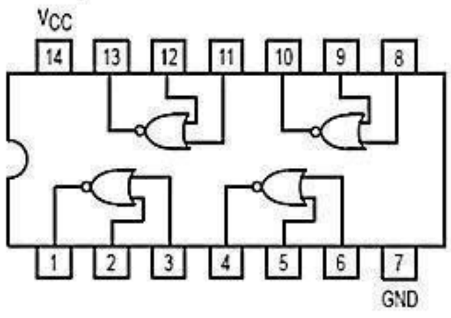
**2-Input OR Gate - 7432LS**



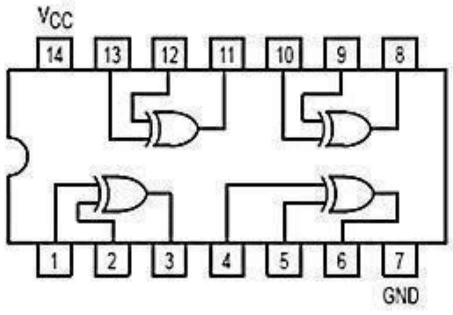
**2-Input NAND Gate - 7400LS**



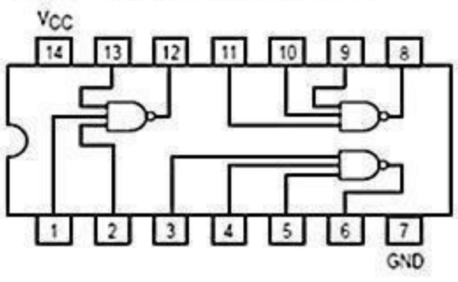
**2-Input NOR Gate - 7402LS**



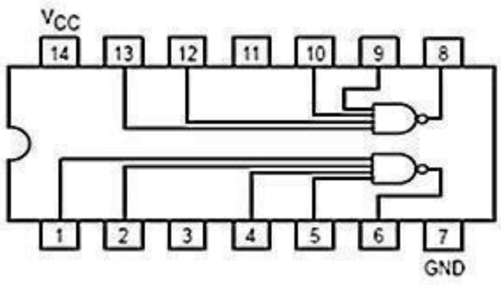
**2-Input EX-OR Gate - 7486LS**

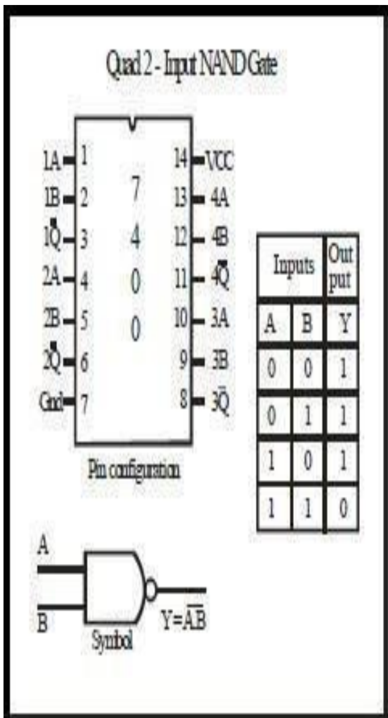


**3-Input NAND Gate - 7410LS**

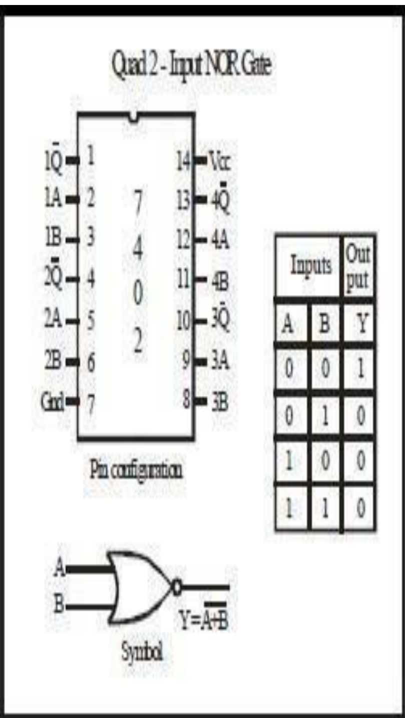


**Dual 4-Input NAND Gate - 7420LS**

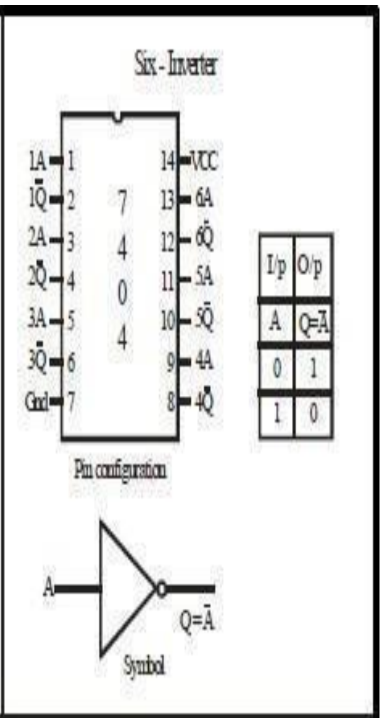




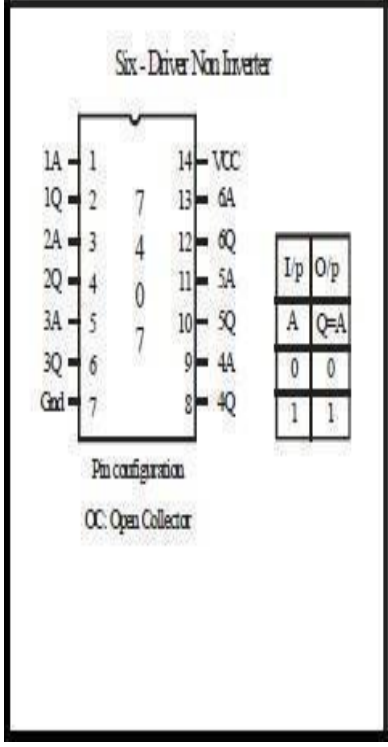
| Inputs | | Out put |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



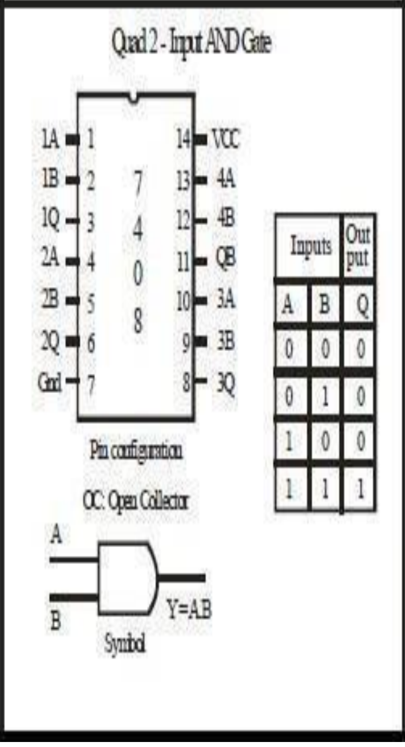
| Inputs | | Out put |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



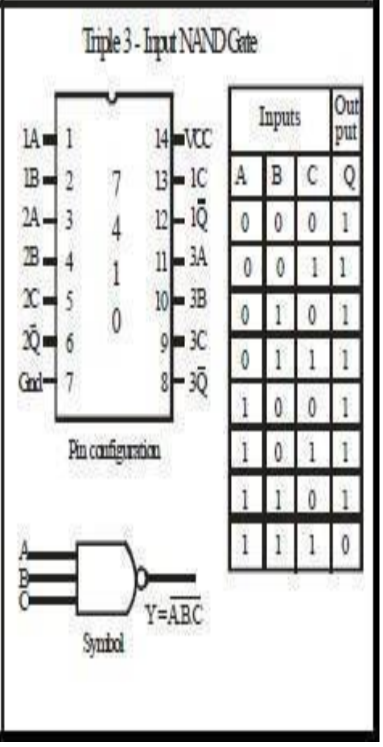
| I/p | O/p |
|---|---|
| A | Q=$\bar{A}$ |
| 0 | 1 |
| 1 | 0 |



| I/p | O/p |
|---|---|
| A | Q=A |
| 0 | 0 |
| 1 | 1 |

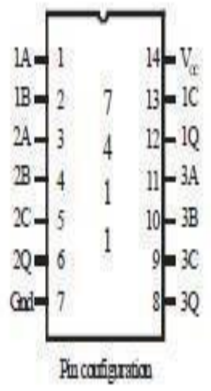


| Inputs | | Out put |
|---|---|---|
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



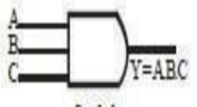
| Inputs | | | Out put |
|---|---|---|---|
| A | B | C | Q |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

7411 Triple 3 - Input AND Gate



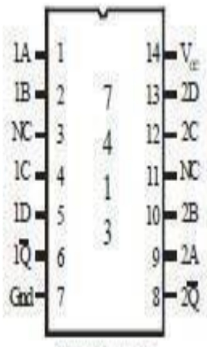
Pin configuration

| Inputs | | | Out put |
|---|---|---|---|
| A | B | C | Y |
| X | X | 0 | 0 |
| X | 0 | X | 0 |
| 0 | X | X | 0 |
| 1 | 1 | 1 | 1 |



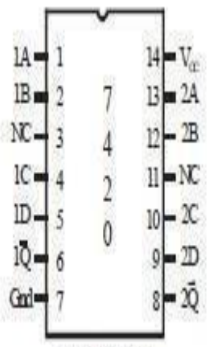
Symbol

7413 Dual 4-Input NAND Schmitt Trigger



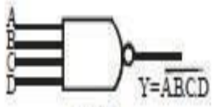
Pin configuration

7420 Dual 4 - Input NAND Gate



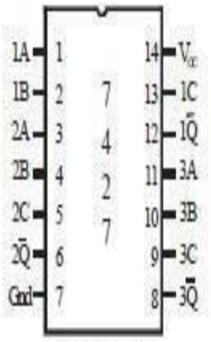
Pin configuration

| Inputs | | | | Out put |
|---|---|---|---|---|
| A | B | C | D | Y |
| X | X | X | 0 | 1 |
| X | X | 0 | X | 1 |
| X | 0 | X | X | 1 |
| 0 | X | X | X | 1 |
| 1 | 1 | 1 | 1 | 0 |



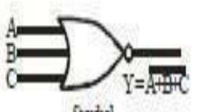
Symbol

7427 Triple 3 - Input NOR Gate



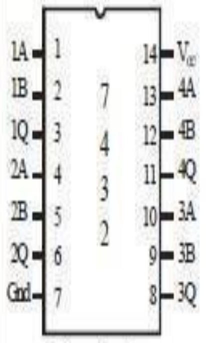
Pin configuration

| Inputs | | | Out put |
|---|---|---|---|
| A | B | C | Q |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



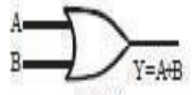
Symbol

7432 Quad 2 - Input NOR Gate



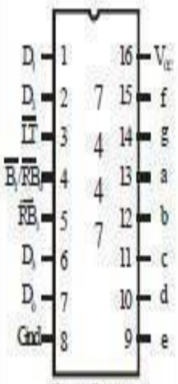
Pin configuration

| Inputs | | Out put |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

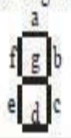


Symbol

7447(OC) BCD to 7 - Segment Decoder

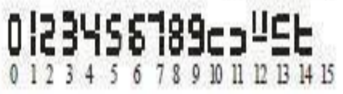


Pin configuration

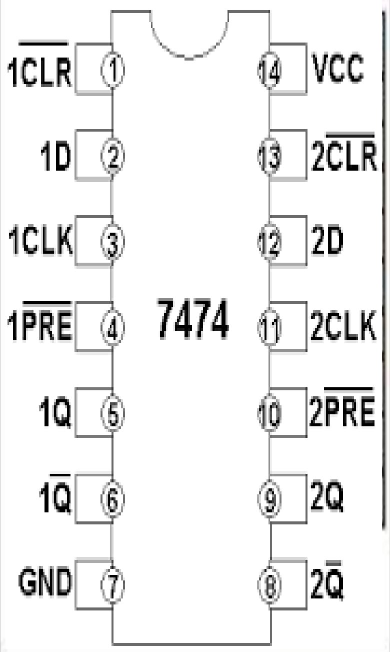


| Inputs | | | | | | I/p | O/p |
|---|---|---|---|---|---|---|---|
| $D_3$ | $D_2$ | $D_1$ | $D_0$ | $\overline{LT}$ | $\overline{RBI}$ | $\overline{BI}/\overline{RBO}$ | Q |
| x | x | x | x | 0 | x | 1 | 8 |
| x | x | x | x | x | x | 0 | - |
| 0 | 0 | 0 | 0 | 1 | x | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | x | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | x | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 | x | 1 | 15 |

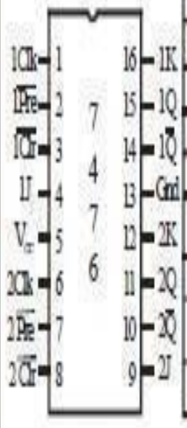
OC: Open Collector



(TOP VIEW)
W PACKAGE



7476 Dual J-K Flip flop with Preset & Clear

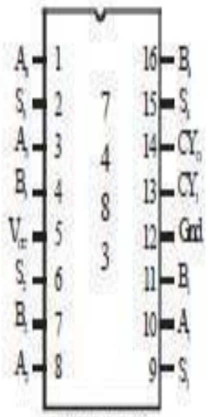


Pin configuration

| Inputs | | | | | Output | |
|---|---|---|---|---|---|---|
| $\overline{Pre}$ | $\overline{Clr}$ | CLK | J | K | Q | $\overline{Q}$ |
| 0 | 1 | x | x | x | 1 | 0 |
| 1 | 0 | x | x | x | 0 | 1 |
| 0 | 0 | x | x | x | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | NO CHANGE | |
| 1 | 1 | ⎍ | 0 | 1 | 0 | 1 |
| 1 | 1 | ⎍ | 1 | 0 | 1 | 0 |
| 1 | 1 | ⎍ | 1 | 0 | TOGGLE | |

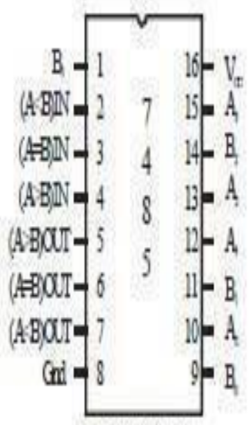
Pre = Preset
Clr = Clear

7483 4 Bit Full Adder



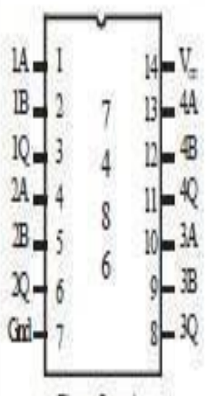
Pin configuration

7485 4 Bit Comparator



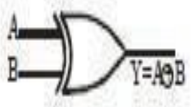
Pin configuration

7486 Quad 2 - Input EX - OR Gate



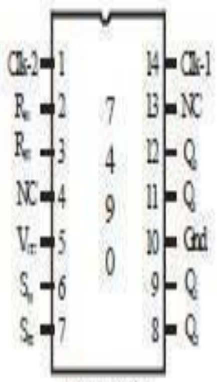
Pin configuration

| Inputs | | Out put |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

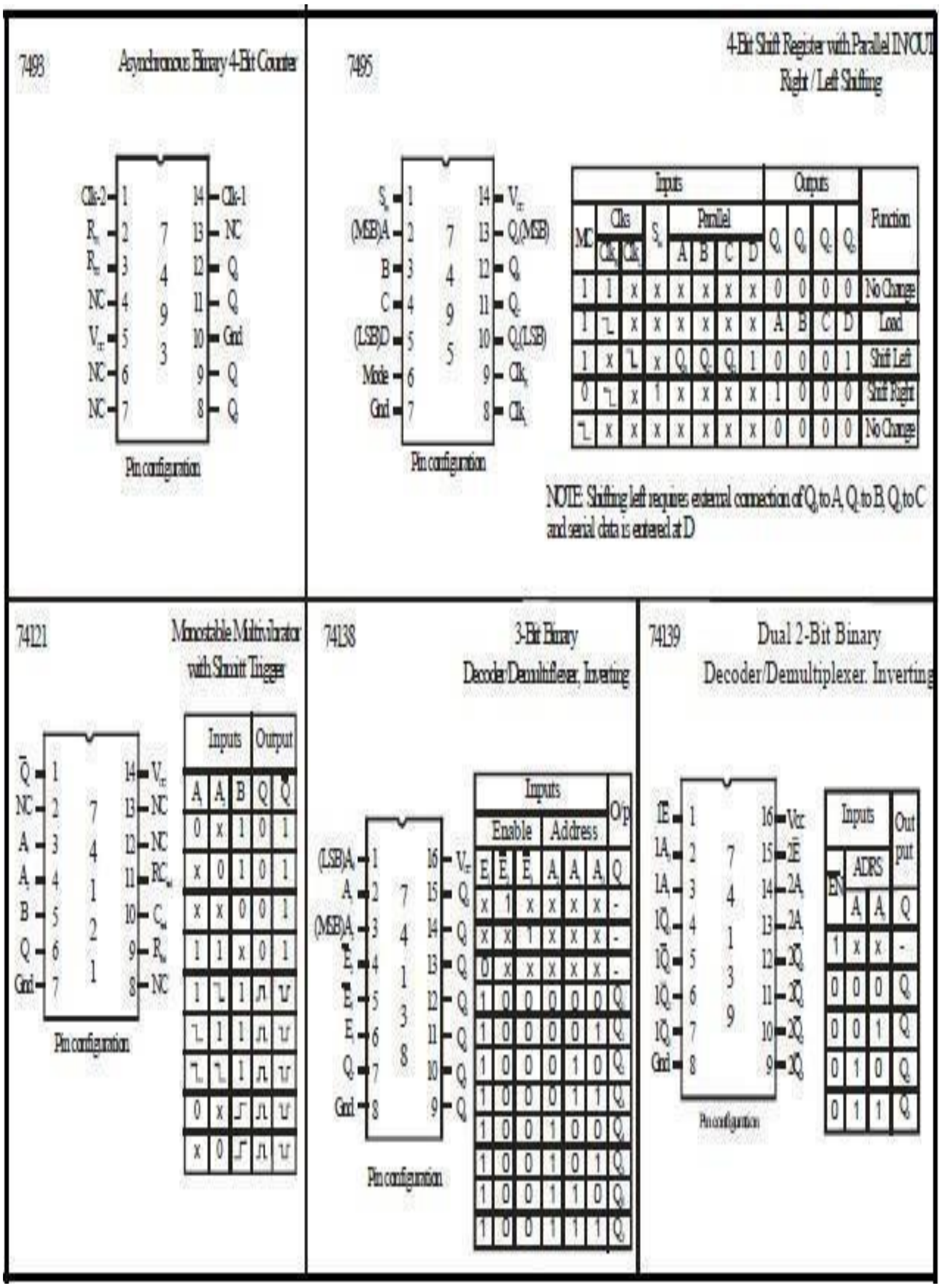


Symbol

7490 Asynchronous Decade up Counter



Pin configuration



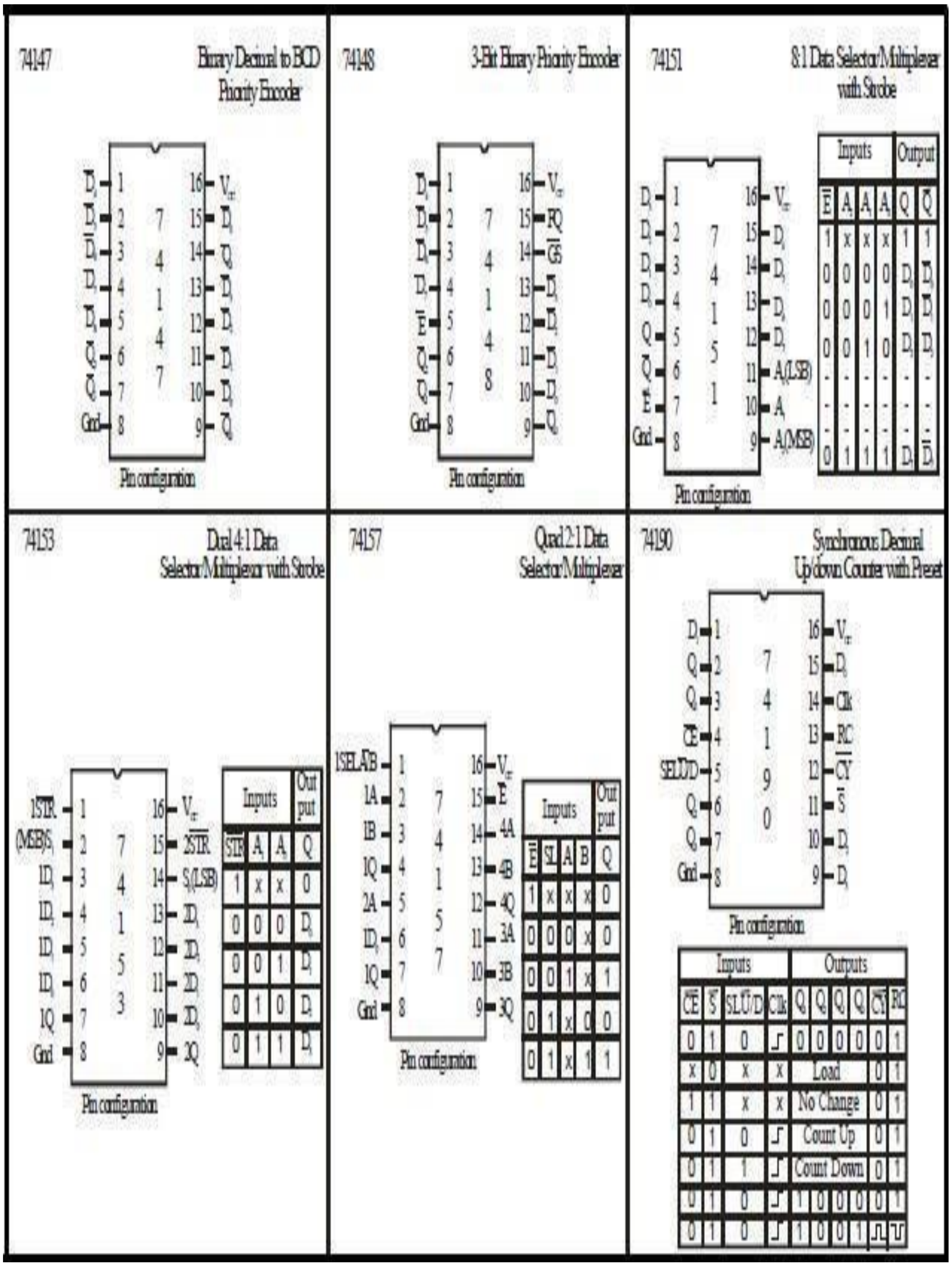
| Inputs | | | | | | | | Outputs | | | | Function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MC | Clks | | $S_{in}$ | Parallel | | | | $Q_A$ | $Q_B$ | $Q_C$ | $Q_D$ | |
| | $Clk_1$ | $Clk_2$ | | A | B | C | D | | | | | |
| 1 | 1 | x | x | x | x | x | x | 0 | 0 | 0 | 0 | No Change |
| 1 | ↓ | x | x | x | x | x | x | A | B | C | D | Load |
| 1 | x | ↓ | x | $Q_B$ | $Q_C$ | $Q_D$ | 1 | 0 | 0 | 0 | 1 | Shift Left |
| 0 | ↓ | x | 1 | x | x | x | x | 1 | 0 | 0 | 0 | Shift Right |
| ↓ | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 | No Change |

NOTE: Shifting left requires external connection of $Q_B$ to A, $Q_C$ to B, $Q_D$ to C and serial data is entered at D

| Inputs | | | Output | |
|---|---|---|---|---|
| $A_1$ | $A_2$ | B | Q | Q̄ |
| 0 | x | 1 | 0 | 1 |
| x | 0 | 1 | 0 | 1 |
| x | x | 0 | 0 | 1 |
| 1 | 1 | x | 0 | 1 |
| 1 | ↓ | 1 | ⊓ | ⊔ |
| ↓ | 1 | 1 | ⊓ | ⊔ |
| ↓ | ↓ | 1 | ⊓ | ⊔ |
| 0 | x | ↑ | ⊓ | ⊔ |
| x | 0 | ↑ | ⊓ | ⊔ |

| Inputs | | | | | | O/p |
|---|---|---|---|---|---|---|
| Enable | | | Address | | | |
| $E_3$ | $\bar{E}_2$ | $\bar{E}_1$ | $A_2$ | $A_1$ | $A_0$ | Q |
| x | 1 | x | x | x | x | - |
| x | x | 1 | x | x | x | - |
| 0 | x | x | x | x | x | - |
| 1 | 0 | 0 | 0 | 0 | 0 | $Q_0$ |
| 1 | 0 | 0 | 0 | 0 | 1 | $Q_1$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $Q_2$ |
| 1 | 0 | 0 | 0 | 1 | 1 | $Q_3$ |
| 1 | 0 | 0 | 1 | 0 | 0 | $Q_4$ |
| 1 | 0 | 0 | 1 | 0 | 1 | $Q_5$ |
| 1 | 0 | 0 | 1 | 1 | 0 | $Q_6$ |
| 1 | 0 | 0 | 1 | 1 | 1 | $Q_7$ |

| Inputs | | | Output |
|---|---|---|---|
| $\bar{EN}$ | ADRS | | |
| | $A_1$ | $A_0$ | Q |
| 1 | x | x | - |
| 0 | 0 | 0 | $Q_0$ |
| 0 | 0 | 1 | $Q_1$ |
| 0 | 1 | 0 | $Q_2$ |
| 0 | 1 | 1 | $Q_3$ |

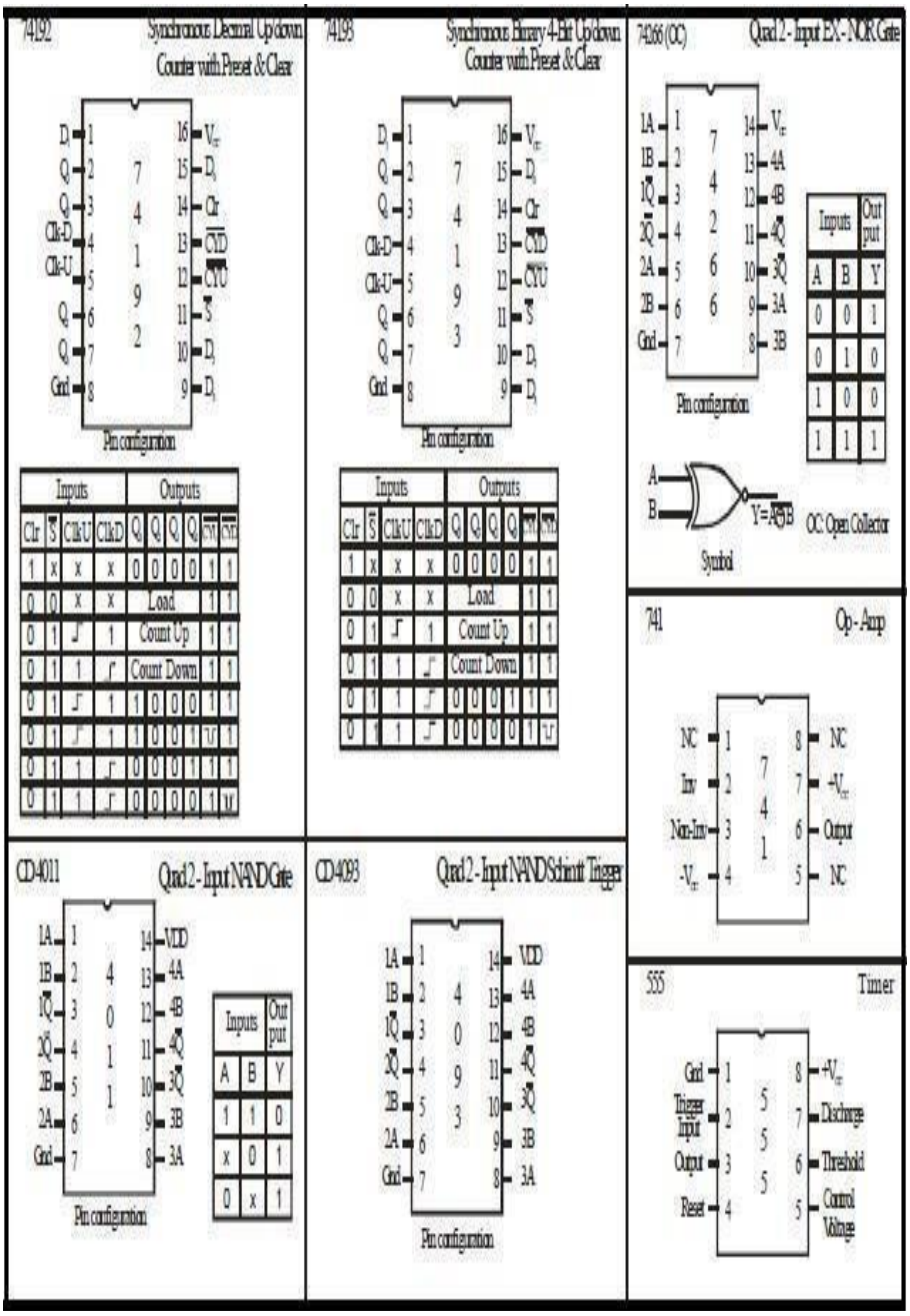

74147
Binary Decimal to BCD Priority Encoder
Pin configuration
74148
3-Bit Binary Priority Encoder
Pin configuration
74151
8:1 Data Selector/Multiplexer with Strobe
Pin configuration
74153
Dual 4:1 Data Selector/Multiplexer with Strobe
Pin configuration
74157
Quad 2:1 Data Selector/Multiplexer
Pin configuration
74190
Synchronous Decimal Up/down Counter with Preset
Pin configuration

| Inputs | | | | Output | |
|---|---|---|---|---|---|
| $\overline{E}$ | $A_2$ | $A_1$ | $A_0$ | Q | $\overline{Q}$ |
| 1 | x | x | x | 1 | 1 |
| 0 | 0 | 0 | 0 | $D_0$ | $\overline{D}_0$ |
| 0 | 0 | 0 | 1 | $D_1$ | $\overline{D}_1$ |
| 0 | 0 | 1 | 0 | $D_2$ | $\overline{D}_2$ |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| 0 | 1 | 1 | 1 | $D_7$ | $\overline{D}_7$ |

| Inputs | | | Output |
|---|---|---|---|
| $\overline{STR}$ | $A_1$ | $A_0$ | Q |
| 1 | x | x | 0 |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |

| Inputs | | | | Output |
|---|---|---|---|---|
| $\overline{E}$ | SL | A | B | Q |
| 1 | x | x | x | 0 |
| 0 | 0 | 0 | x | 0 |
| 0 | 0 | 1 | x | 1 |
| 0 | 1 | x | 0 | 0 |
| 0 | 1 | x | 1 | 1 |

| Inputs | | | | Outputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\overline{CE}$ | $\overline{S}$ | SL $\overline{U}$/D | Clk | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $\overline{CY}$ | RC |
| 0 | 1 | 0 | ⎍ | 0 | 0 | 0 | 0 | 0 | 1 |
| x | 0 | x | x | Load | | | | 0 | 1 |
| 1 | 1 | x | x | No Change | | | | 0 | 1 |
| 0 | 1 | 0 | ⎍ | Count Up | | | | 0 | 1 |
| 0 | 1 | 1 | ⎍ | Count Down | | | | 0 | 1 |
| 0 | 1 | 0 | ⎍ | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | ⎍ | 1 | 0 | 0 | 1 | ⎍ | ⊔ |



74192

| Inputs | | | | Outputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Clr | $\overline{S}$ | ClkU | ClkD | Q | Q | Q | Q | $\overline{CYU}$ | $\overline{CYD}$ |
| 1 | x | x | x | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | x | x | Load | | | | 1 | 1 |
| 0 | 1 | ↑ | 1 | Count Up | | | | 1 | 1 |
| 0 | 1 | 1 | ↑ | Count Down | | | | 1 | 1 |
| 0 | 1 | ↑ | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | ↑ | 1 | 1 | 0 | 0 | 1 | ⊔ | 1 |
| 0 | 1 | 1 | ↑ | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | ↑ | 0 | 0 | 0 | 0 | 1 | ⊔ |

74193

| Inputs | | | | Outputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Clr | $\overline{S}$ | ClkU | ClkD | Q | Q | Q | Q | $\overline{CYU}$ | $\overline{CYD}$ |
| 1 | x | x | x | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | x | x | Load | | | | 1 | 1 |
| 0 | 1 | ↑ | 1 | Count Up | | | | 1 | 1 |
| 0 | 1 | 1 | ↑ | Count Down | | | | 1 | 1 |
| 0 | 1 | 1 | ↑ | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | ↑ | 0 | 0 | 0 | 0 | 1 | ⊔ |

74266 (OC)

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

CD4011

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 1 | 1 | 0 |
| x | 0 | 1 |
| 0 | x | 1 |

## DE LAB VIVA QUESTIONS & ANSWERS

1. State the method used to simplify the Boolean equation Karnaugh map, quine Mcluskey method; petric's method, decimal method and variable entered mapping technique.

2. What is meant by duality in Boolean algebra?
One expression can be obtained from other by replacing every 0 with 1, every 1 with 0, every (+) with (.) and every (.) with (+ ). Such pair of expressions is called dual expression and this characteristic is called principle of duality

3. State the Demorgan's theorem
    a. complement of product is equal to the products.$\overline{AB}=\overline{A}+\overline{B}$
    b. The complement of a sum is equal to the sum of the products.

4. What are Minterms and Maxterms?
A product term containing all the k variables (literals) of the function is either complemented or uncomplemented form is called a Minterm.
A sum term containing the entire k variable (literals) of the function in either complemented or uncomplemented form is called a Maxterm.

5. What are the basic logic operations?
Logic AND, logic OR, and logic complementation.

6. What are the combinational logic circuits?
Combinational logic circuits are digital circuits, whose out put at any time depends on the combination of present input values at that time.

7. What is the difference between full adder and half adder?
Full adder is ckt where the carry out from adjacent bit LSB position is also added.
Half adder is one – bit adder ckt where the carry out from the adjacent least bit position is neglected.

8. What is basic drawback of n- bit parallel adder? How is it overcome?
In an n-bit parallel adder, the o/p (Count $S_3$, $S_2$, $S_1$, $S_0$ ) is available only after the carry is propagated through each pf the address. The delay is n x tp where tp propagation delays of each adder. This is reduced by using Look ahead carry adder, where the carry required at each stage is made available simultaneously.

9. Explain how an adder can be converted in to Sub tractor?
Consider the subtraction, A-B which can be written as A+ (-B).here
(-B) is represented in 2's complemented form and adder with A, if the difference is `-ve 'it will be in the 2's complemented form.

10. What are the weighed binary codes? Give examples.
Weighed binary codes are those, which obey their positional weighting Principles, the bits are multiplied by the weights indicated and the sum of these Weighted bits give the equivalent decimal value e.g. 8421, 5421, 2421.

10. What are non – weighted codes? Give examples.
Non –weighted codes are codes that are not positionally weighted.
E .g:- excess -3codes and gray code.

11. Which is the number system used in digital system?
Binary number system is widely used in digital system, because only two voltage levels are sufficient to represent the two numbers 0 and 1 of binary system.

12. What is encoding a code?

    When numbers letters or words are represented by a special group of symbols, this is called encoding and the group of symbols is called a code.

13. What is gray code or unit distance code?

    Gray code is one which only one bit in the code changes when moving from one step to next; it is non-weighted code.

14. What are the applications of the gray code?

    Gray code finds applications in input/output devices and in some types of analog to digital converters and in communications.

15. What is a MUX?

    Multiplex means many into one output. A multiplexer is a circuit with many inputs but only one output. By applying control signal we can drive any input to the output.

16. Why multiplexers are called data –selectors?
    Because the output data depends on the input data bit that is selected by the control signals.

17. What is Demultiplexers?
    Demultiplexer means one into many. A demultiplexer is a logic ckt with one input and many outputs. By applying control signal, and drive the input to one of the output lines.

18. What is 74153 IC?
    74153 IC is a dual 4:1 MUX.

19. What is difference between decoder and demultiplexers?
    A decoder is similar to the demultiplexers, with one exception – there is no data input i.e. the only input are the controlbits.

20. What is comparator?

    A comparator is a combinational circuit designed primarily to compare the relative magnitude of two binary numbers.

21. What is a sequential circuit?
    Sequential circuit is one whose o/p at any instant upon the present i/p's as well as past o/p.

22. What is D flip flop?

    The D(delay flip flop has only one input called delay (D) inputs and two out puts Q and Q ,where the next state follows the delay input.

23. Write down the next state table of SK flip flop.

| J | K | Q | $Q^{-1}$ |
|---|---|---|---|
| 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 |

24. What are the differences b/w a FF and latch?
Latch from a class of flip flops, where the timing of the o/p changes is not controlled even through sometimes a special control signal called or clock is used. Flip flops are storage device, where the o/p where the o/p changes are controlled by the changes by the changes in the controlled signal called a trigger.

25. What is characteristic equation of a flip flop?
The algebraic description of the next state table of a flip-flop is called the characteristic equation of the flip flop.

26. What are synchronous and asynchronous counter?
Asynchronous /ripple counter are those where each flip flop is triggered by the output from the previous flip-flop, which limits its speed of operation.
Synchronous counters are those where all the flip flops are triggered /locked simultaneously.

27. What does the modulus number of a counter signify?

i. Modulus number signifies the total number of state it sequences through in each complete cycle
ii. MOD-number $=2^n$, where n=no. of flip flop.

28. What is frequency division or frequency scaling w.r.t counters?
The output frequency of I the flip flop in n – bit counter is divided by $2^I$, & at the MS bit flip flop, it is divided by $2^n$, therefore we say the counter is a frequency divider.

29. Categorize the following IC's into asynchronous & synchronous counter IC's (7490, 74192, 7493, 74193)

iii. asynchronous counter IC's (7490, 7493)
iv. Synchronous counter IC's (74192, 74193).

30. What is Decade counter?
A counter has 10 different state (0-9) is called a decade counter.

31. What are a register and a shift register?
A register is a group of flip flops suitable for storing binary information & the data can be moved in and out of the FF's only. A shift register is a register where the data can also be moved between the flip flops of the register.

32. What are the different types of shift register?

(i) Serial in serial out
(ii) Serial in parallel out
(iii) parallel in serial out
(iv) parallel in parallel out

33. What are unidirectional and bi-directional shift registers?
A register, which can shift data in only one directional, is called a unidirectional shift register.A register, which is capable of shifting of shifting data both to the right and left, is called a shift register with parallel register.

34. What is universal shift register/shift resister with parallel load?
If the register has shift and parallel load capabilities, then it is called a shift register with parallel load or universal shift register.

35. What is a main difference b/w a register & a counter?
A register has no specific sequence of state except in certain specialized applications, but a counter has to have a defined sequence.

36. What are shift register counter?

Shift register counter are circuit which use feedback, where the o/p of the last FF in the shift register is connected back to the first flip flop in different manners, so as to obtain a specific sequence of states.

37. What is ring counter?

In a ring counter, the true o/p 'Q' of the last FF of a shift register is back to the serial i/p of the first FF.

38. What is Johnson counter /twisted ring counter /shift counter?

In a Johnson counter, the inverted o/p Q of the last FF in a shift register is connected back to the serial i/p of the first FF.

## VISION

To become a pioneer in developing competent professionals with societal and ethical values through transformational learning and interdisciplinary research in the field of Electronics and Communication Engineering.

## MISSION

The department of Electronics and Communication is committed to:

**M1**: Offer quality technical education through experiential learning to produce competent engineering professionals.

**M2**: Encourage a culture of innovation and multidisciplinary research in collaboration with industries/universities.

**M3**: Develop interpersonal, intrapersonal, entrepreneurial and communication skills among students to enhance their employability.

**M4**: Create a congenial environment for the faculty and students to achieve their desired goals and to serve society by upholding ethical values.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

Upon completion of the program, graduates will be able to:
**PEO1**: Tackle complex engineering problems with the sound knowledge of basic science and mathematics.

**PEO2:** Utilize their knowledge and skills to develop solutions in multi-disciplinary environments through collaborative research.

**PEO 3:** Inculcate effective communication skills, teamwork and leadership for a successful career in industry and academia.

**PEO4:** Exhibit professional ethics and social awareness in their professional career and engage in lifelong learning.

## PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:
**PO1.** Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2**. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3**. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4.** Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5.** Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6.** The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7.** Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8.** Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9.** Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10.** Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11.** Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12.** Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1**

Apply the knowledge of leading-edge hardware and software tools to solve problems in the area of Embedded Systems, VLSI and IoT.

**PSO2**

Apply the concepts of Signal and Image Processing to solve problems in communication systems.