EXPERIMENT-1

Realization of Adder & Subtractor

AIM: To realise full adder / subtractor using
(i) Basic gates (ii) Universal gates

SOFTWARE: Circuit verse / Multiverse, Multisim 14, IC 7410
IC Trainer kit, IC 7408, IC 7432, IC 7400, IC 7404

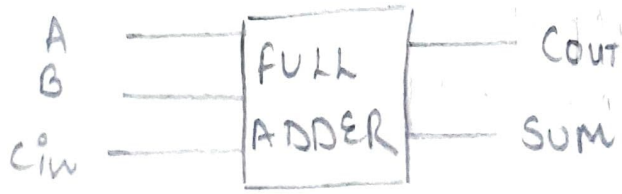
THEORY

Full Adder Theory

Full Adder: It takes 3 inputs & produces 2 outputs
first two inputs are A & B and third one
is Carry → Cin

The outputs are Cout & S [Sum]

A full Adder is designed in such a manner that it can take 3 inputs and creates a byte wide adder and cascade the carry bit from one adder to another.



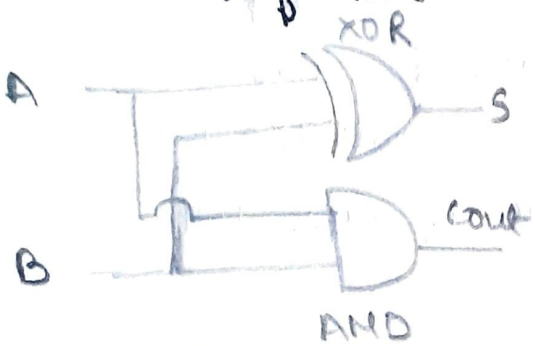
Full subtractor

A full subtractor is a conventional circuit that performs subtractions of two bits, one is minuend and other is subtrahend, taking into amount borrow of previous adjacent lower minuend. The circuit has 3 input & 2 outputs.

ADDERS

- K- map, circuit Diagrams, truth table (Simulations)

(1) Half Adder



circuit

INPUT		OUTPUT	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	1

$A \oplus B$

A \ B	0	1
0	0	1
1	1	0

$A \cdot B$

A \ B	0	1
0		
1		1

Half Adder = $A \oplus B$ for

$A \cdot B$ for carry o/p

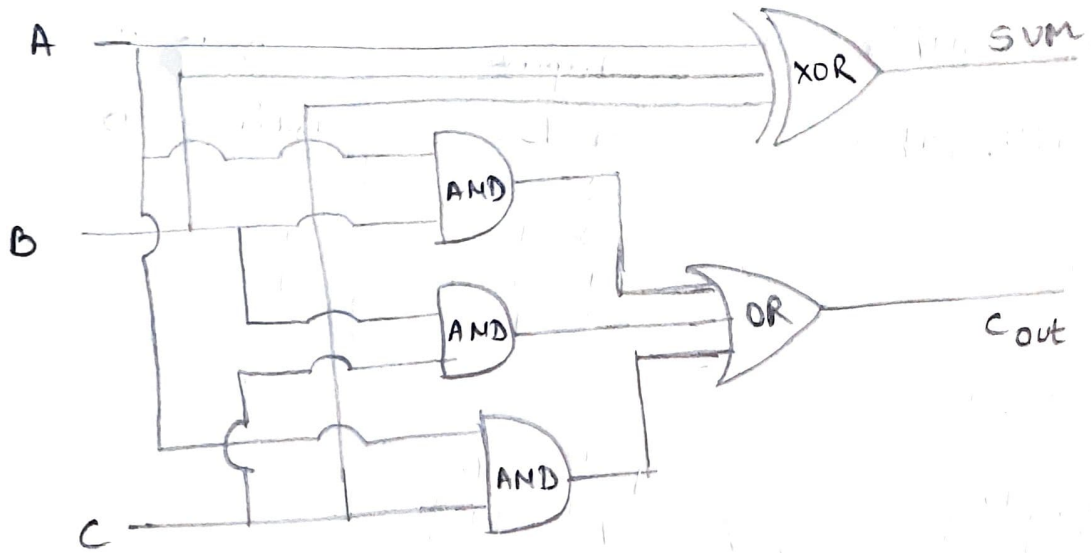
(2) Full Adder

Input			Output	
A	B	C	Sum	Carry
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

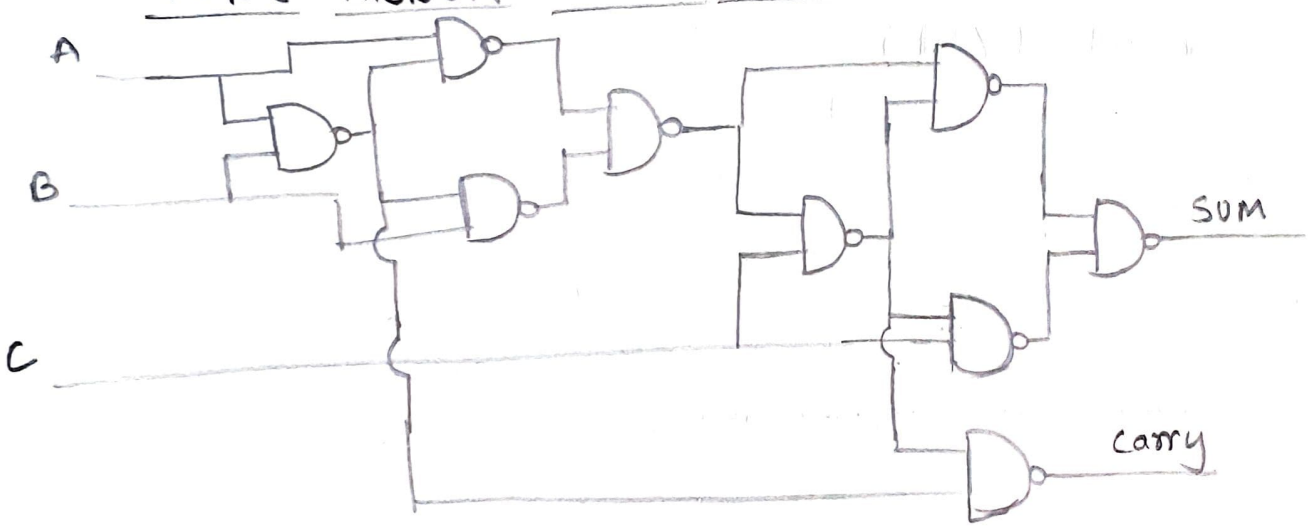
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

A \ B	00	01	11	10
0	0	0	1	
1	0	1	1	1

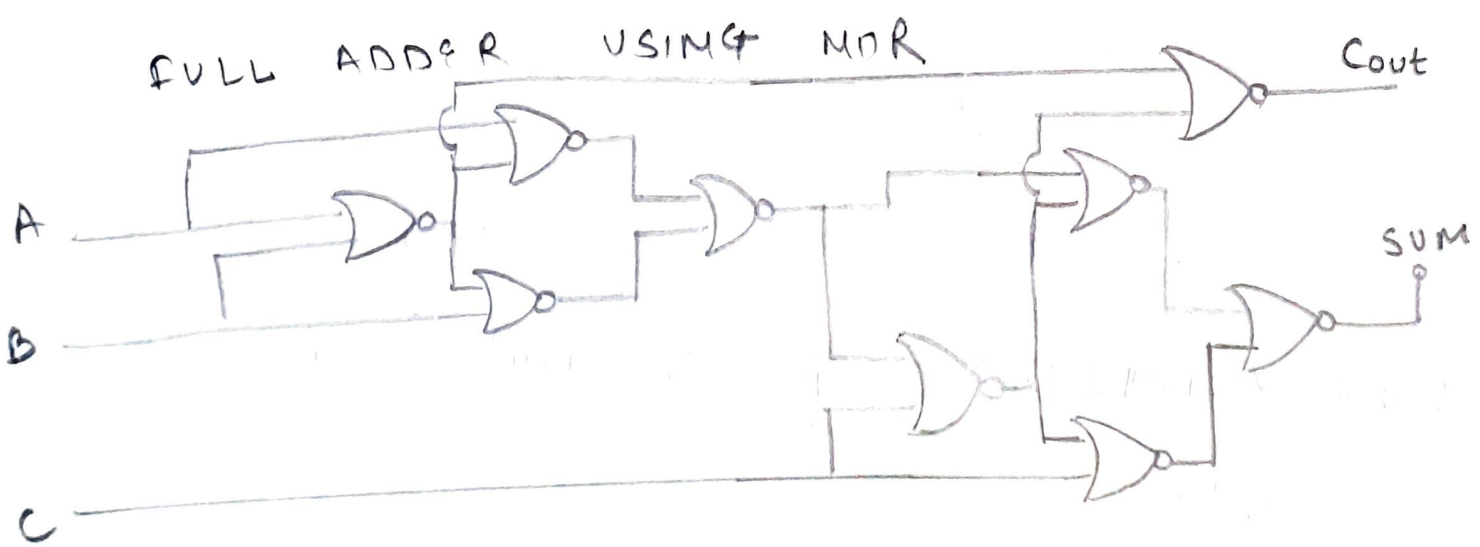
FULL ADDER USING LOGIC GATES



FULL ADDER USING NAND



FULL ADDER USING NOR



SUBTRACTOR

Truth table, karnaugh-Map circuit, Simulator

(1) Half-Subtractor

Truth Table

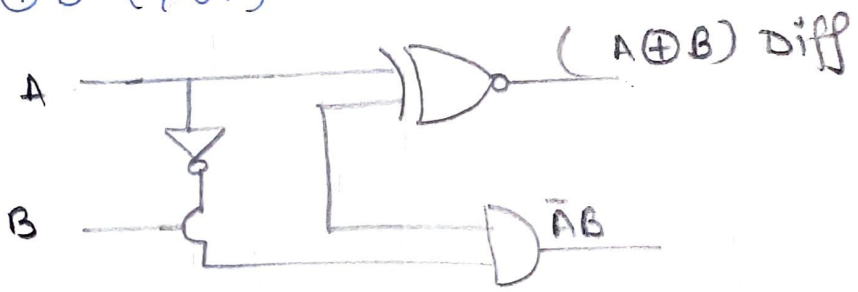
Input		Output	
A	B	Diff	Bor
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference

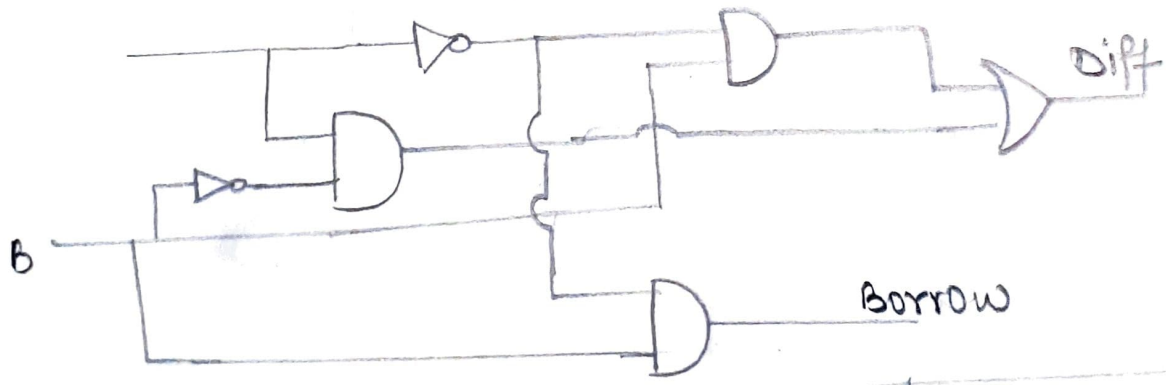
A \ B	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

A \ B	$\bar{B}$	B
$\bar{A}$	0	1
A	0	0

$A \oplus B$ (XOR)



Subtractor using Basic Gate



FULL SUBTRACTOR

truth table

A	B	Bin	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

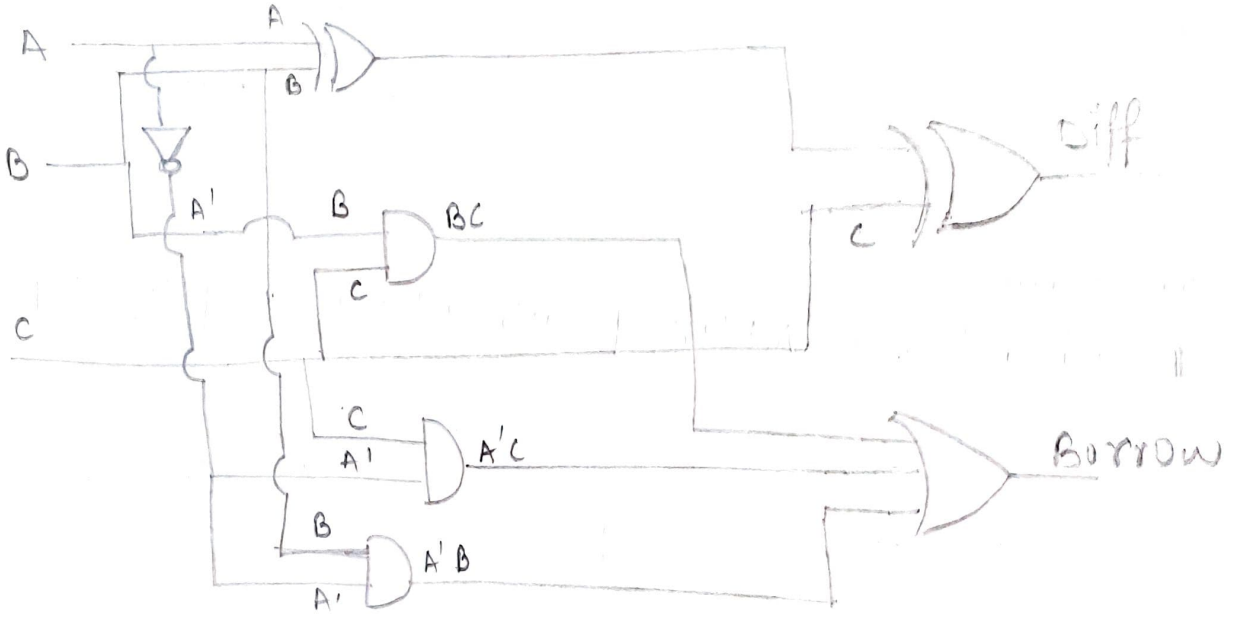
K-map

A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

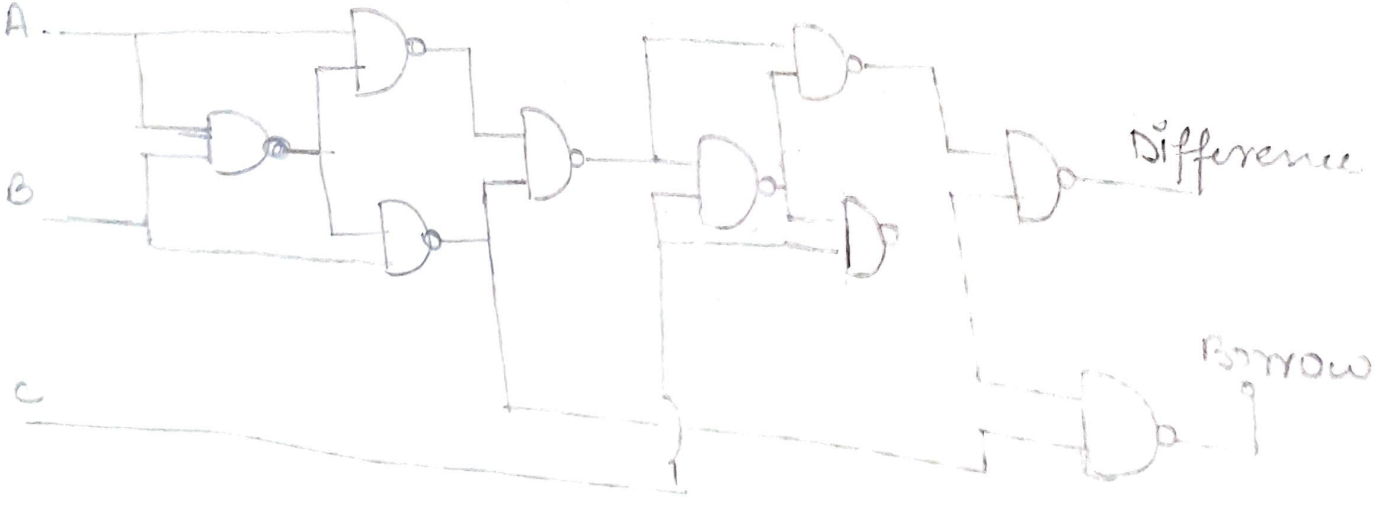
A \ BC				
0	1	1	1	
0	0	1	0	

Diff - $A \oplus B \oplus C$

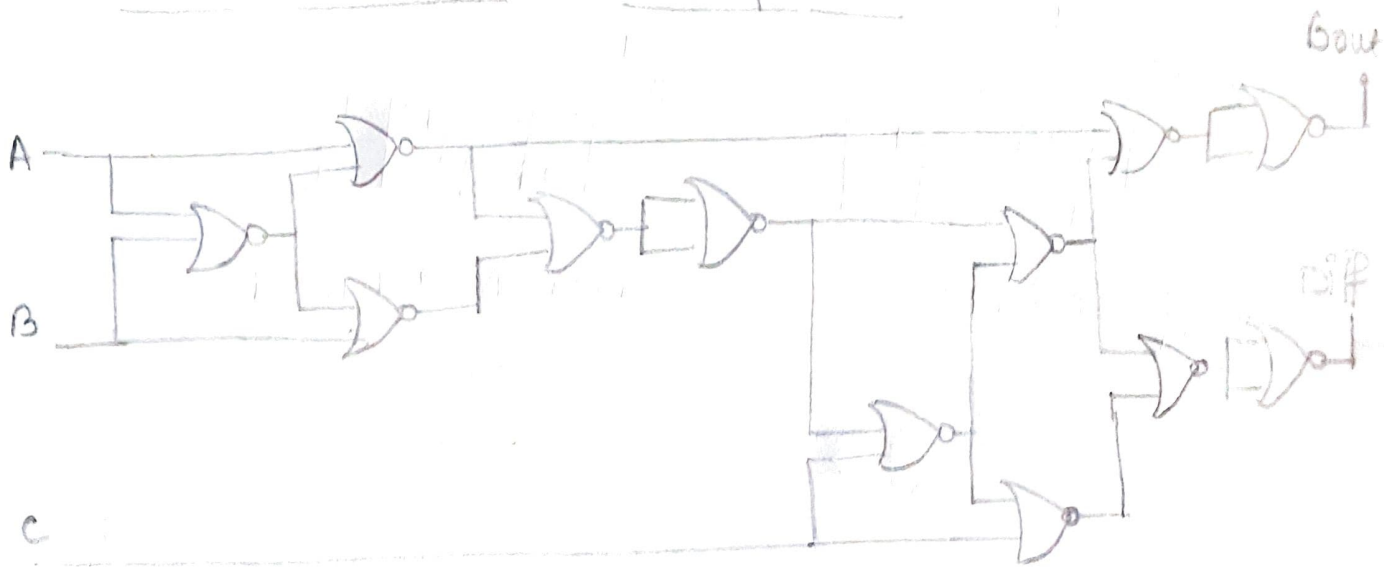
Bor - $A'C + A'B + BC$



FULL SUBTRACTOR USING NAND



FULL SUBTRACTOR USING NOR



Conclusion:

Realization of Adder & Subtractor successfully implemented

SHUBHAM SE-COMP-A

ASSIGNMENT - 2

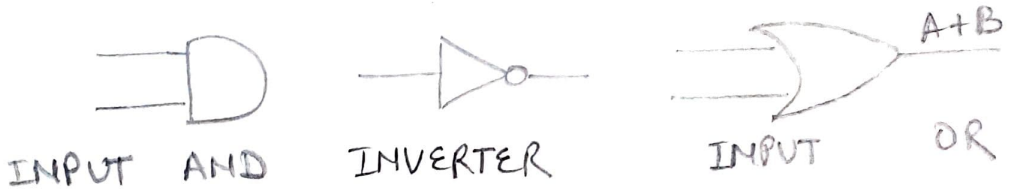
AIM :→ To implement Boolean Expression using logic Gates.

SOFTWARE / HARDWARE / IC-REQUIRED : Multism & IC Trainer Kit, patch chords, IC 7408 (Input And), IC7432 (Input OR) IC 7404 (Input NOR)

THEORY : Implementation of Expression

$A\bar{B}C + [CD + \bar{A}\bar{B}] + B$

LOGIC DIAGRAM FOR THREE BASIC GATES



Any Boolean function can be represented by using a number of logic gates interconnecting them. Logic Gates implementation or logic representation of Boolean function is very simple and easy form.

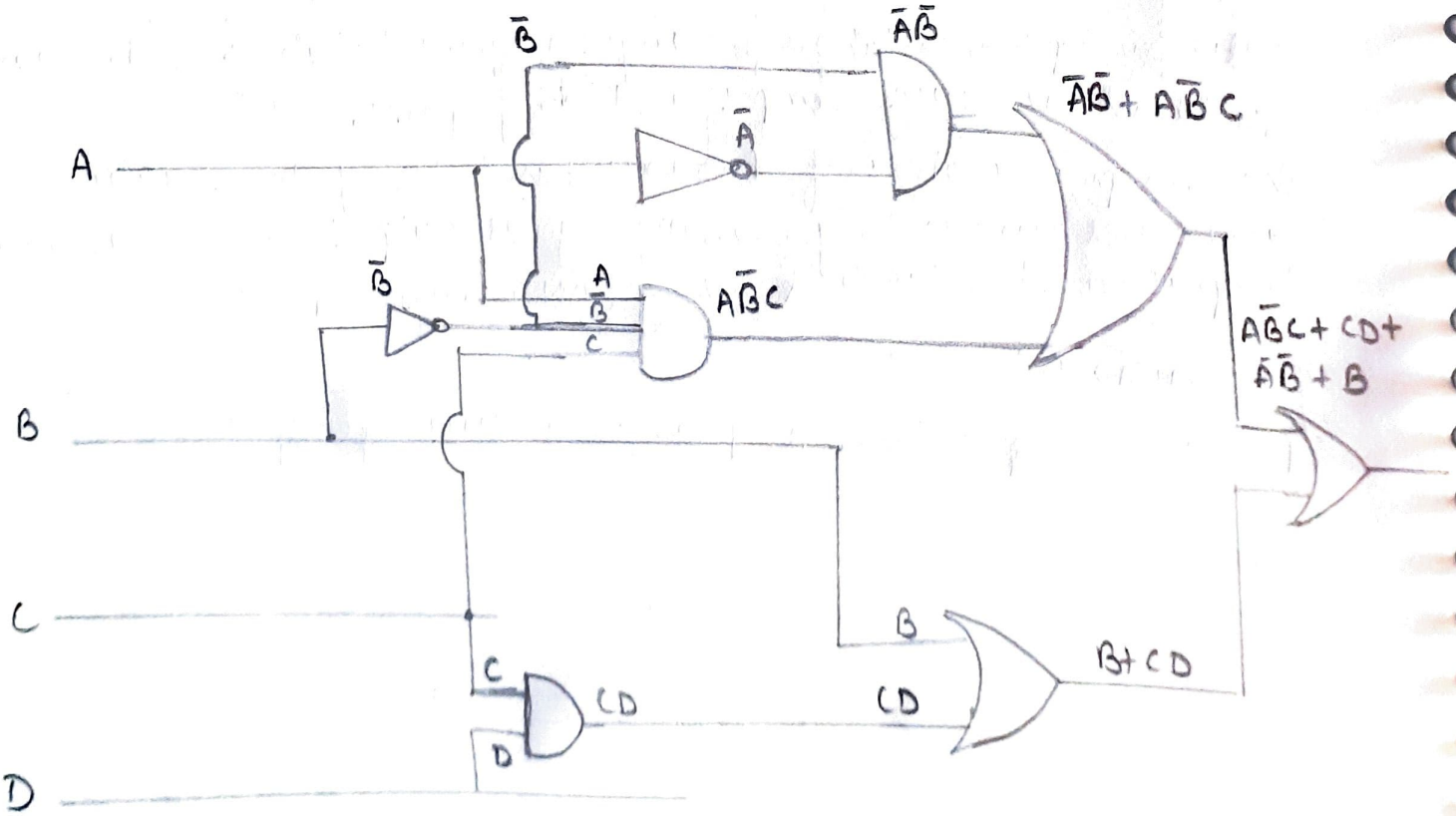
The Implementation of Boolean func^n using logic gates involves connecting one logic gate's output to another Gate's Input and Involved using AND, NOT, OR etc.

Conclusion :-

Boolean expression Implemented successfully using logic Gate.

$A\bar{B}C + [CD + \bar{A}\bar{B}] + B$

A	B	C	D	$\bar{A}$	$\bar{B}$	$\bar{A}\bar{B}$	CD	$A\bar{B}C$	$A\bar{B}C+[CD+\bar{A}\bar{B}]+B$
0	0	0	0	1	1	1	0	0	1
0	0	0	1	1	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1
0	0	1	1	1	1	1	1	0	1
0	1	0	0	1	0	0	0	0	1
0	1	0	1	1	0	0	0	0	1
0	1	1	0	1	0	0	0	0	1
0	1	1	1	1	0	0	1	0	1
1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0
1	0	1	0	0	1	0	0	1	1
1	0	1	1	0	1	0	1	1	1
1	1	0	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	1



EXPERIMENT - 03

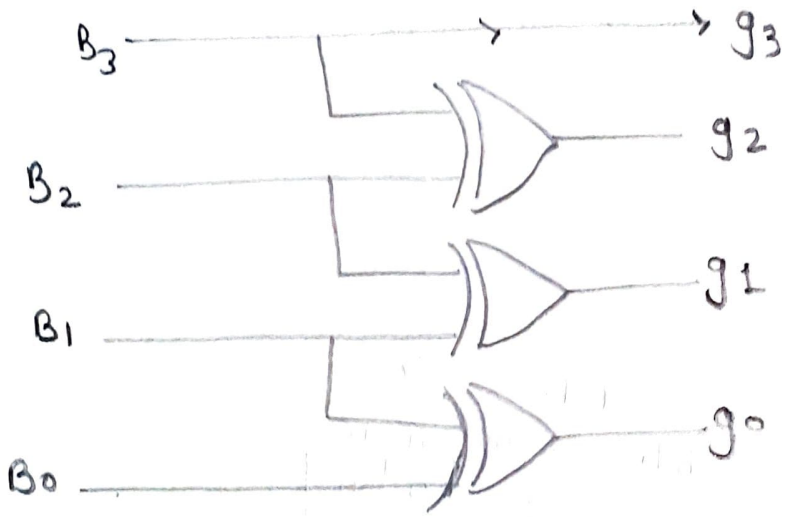
IMPLEMENTATION OF CODE CONVERTOR

AIM: To Design & Implement code convertor Binary to Gray, Gray to Binary, BCD to Excess-3 & Excess 3 to BCD

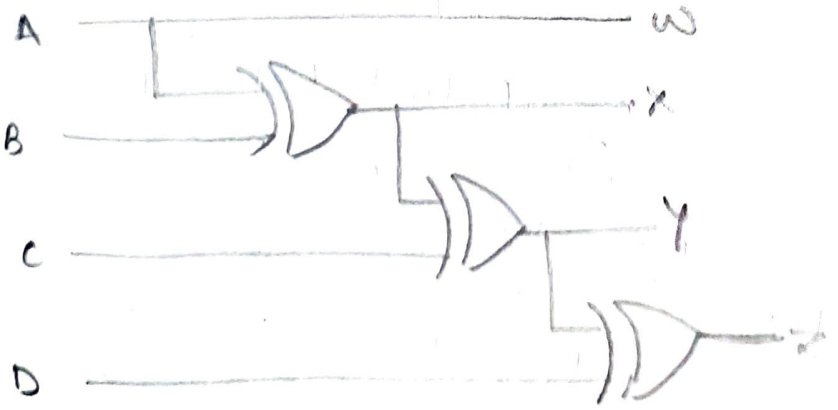
SOFTWARE: Circuit verse | Multisim 14, IC7402, IC7408 IC7432, IC7400, IC7404, IC Trainer kit.

THEORY

Circuit: BINARY TO GREY



Circuit: GREY TO BINARY



BINARY TO GREY

4-Bit Binary code Truth table 4-Bit-Grey-code

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

k-map

G_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

G_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2 = \boxed{B_3 \oplus B_2}$$

G

	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

G_0

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G_0 = B_0 \bar{B}_1 + B_1 \bar{B}_0$$

$$\boxed{G_0 = B_0 \oplus B_1}$$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

G_1

$G_1 = B_2\bar{B}_1 + \bar{B}_2 B$

$\boxed{G_1 = B_2 \oplus B_1}$

GREY TO BINARY

G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

B_3

$\boxed{B_3 = G_3}$

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

B_2

$B_2 = \bar{G}_3 G_2 + G_3 \bar{G}_2$

$\boxed{B_2 = G_3 \oplus G_2}$

B_2

q_3q_2 \ q_1q_0	00	01	11	10
00			1	1
01	1	1		
11			1	1
10	1	1		

B_1

q_3q_2 \ q_1q_0	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$B_1 = \bar{q}_3\bar{q}_2q_1 + \bar{q}_3q_2\bar{q}_1 + q_3q_2q_1 + q_3\bar{q}_2\bar{q}_1$

$\Rightarrow q_3 \oplus q_2 \oplus q_1$

$B_0 = \bar{q}_3\bar{q}_2\bar{q}_1q_0 + \bar{q}_3\bar{q}_2q_1\bar{q}_0 + \bar{q}_3q_2\bar{q}_1\bar{q}_0 + \bar{q}_3q_2q_1q_0 + q_3q_2\bar{q}_1q_0 + q_3q_2q_1\bar{q}_0 + q_3\bar{q}_2\bar{q}_1\bar{q}_0 + q_3\bar{q}_1q_1q_0$

$\boxed{B_0 = q_3 \oplus q_2 \oplus q_1 \oplus q_0}$

BCD to XS-3

BCD

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1
1	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

XS 3

W	X	Y	Z
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

k-map

W

AB \ CD	00	01	11	10
00	0	0	0	0
01		1	1	1
11	X	X	X	X
10	1	1	X	X

$W = A + BD + BC$

X

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$

Y

AB \ CD			
	1	0	1	0
	1	0	1	0
	X	X	X	X
	1	0	X	X

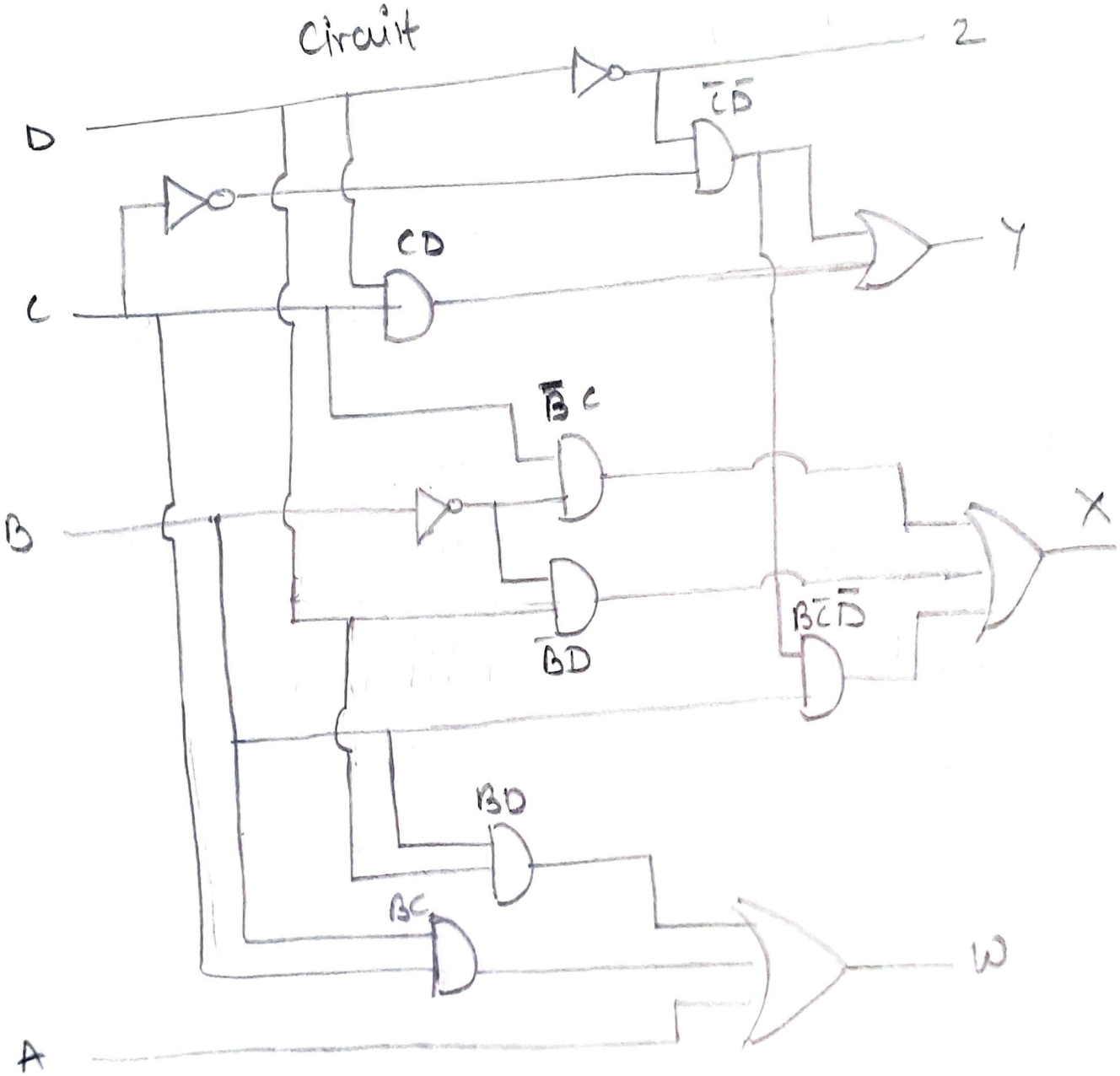
$Y = \bar{C}\bar{D} + CD$

Z

AB \ CD			
	1	0	0	1
	1	0	0	1
	X	X	X	X
	1	0	X	X

$Z = \bar{D}$

Circuit



EXCES-3 TO BCD

E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

K-Map

B_1

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

$$B_3 = E_3E_2 + E_3E_1E_1$$

B_0

E_3E_2 \ E_1E_0				
	X	X		X
			1	
		X	X	X
	1	1		1

$$B_1 = \bar{E}_2\bar{E}_1 + E_2E_1E_0 + \bar{E}_2E_1\bar{E}_0$$

B_1

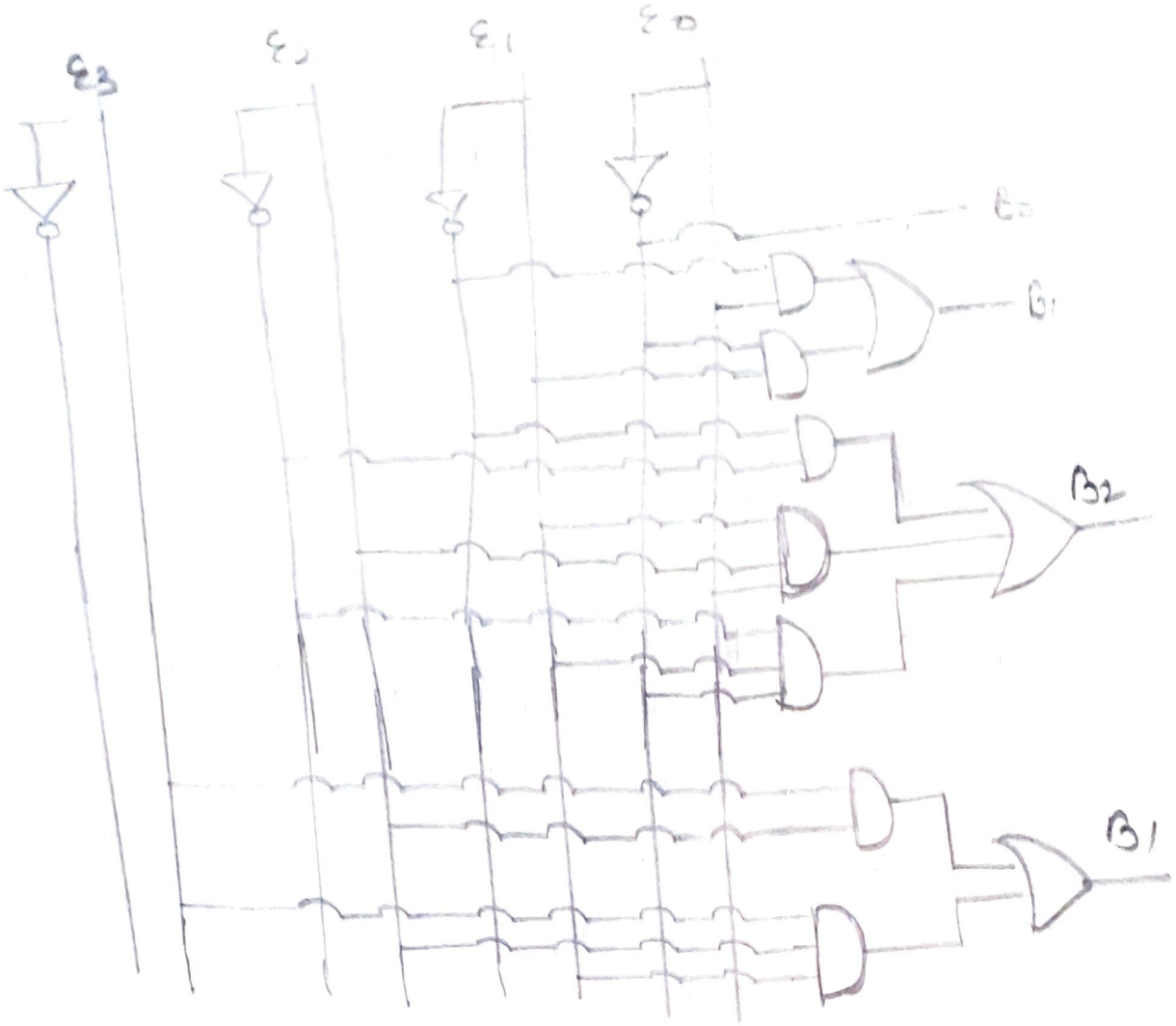
E_3E_2 \ E_1E_0				
	X	X	0	X
	0	1	0	1
	0	X	X	X
	0	1	0	1

$$B_1 = \bar{E}_1E_0 + E_1\bar{E}_0$$

B_0

E_3E_2 \ E_1E_0				
	X	X	0	X
	1	0	0	1
	X	X	X	X
	1	0	0	1

$$B_0 = \bar{E}_0$$



Conclusion

Code converter implementation successfull.

EXPERIMENT - 4

AIM: To implement 4 Bit Binary Adder using IC7483.
To implement 8 Bit Binary Adder using IC7483
To implement 4 Bit BCD Adder using IC7483

Requirment | Software used:
IC Trainer kit, patch chords, IC7408 (AND), IC7432 (OR), IC7483 (4 Bit Binary Adder), Multisim 14.

THEORY

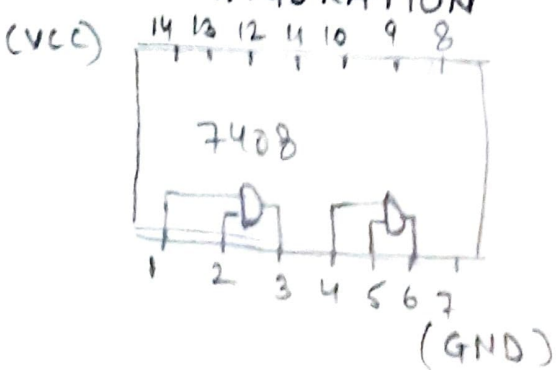
IC 7483 - It is func^n specific or Application specific IC (ASIC)
IC 7483 is a 4 Bit Binary Adder with 2 input of 4 Bits each and 1 (carry in) input. It has one output of 4 Bit & one (carry out) output.

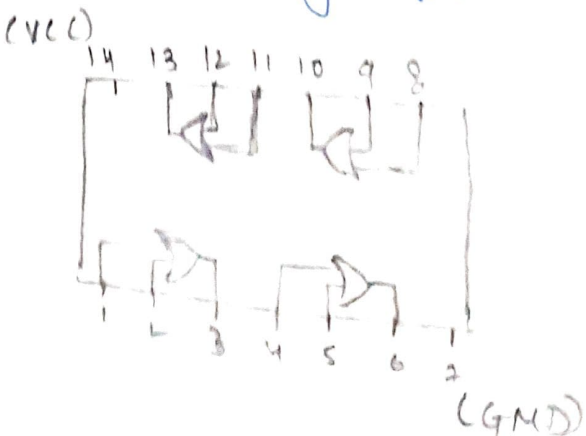
BCD Adder

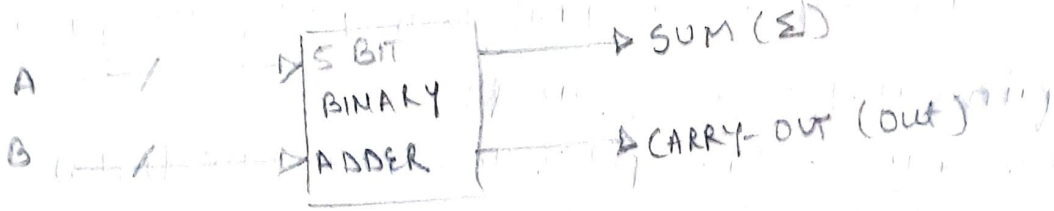
BCD adder is a circuit that Adds two BCD digits and produces a sum digit also in BCD

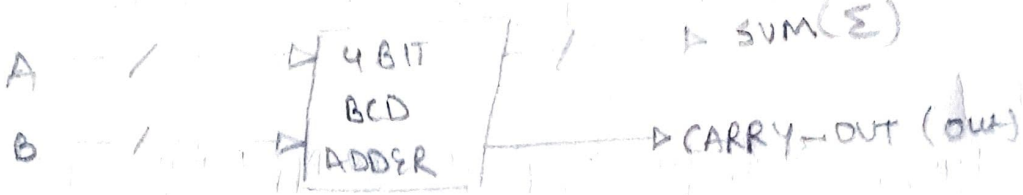
To implement BCD adder we require-
- 4 Bit Binary adder for initial addition
- Logic circuit to detect sum greater than 9
- One more 4 Bit adder to add 6 (0110) in the sum if sum is greater than 9 or carry is 1.

PIN CONFIGURATION







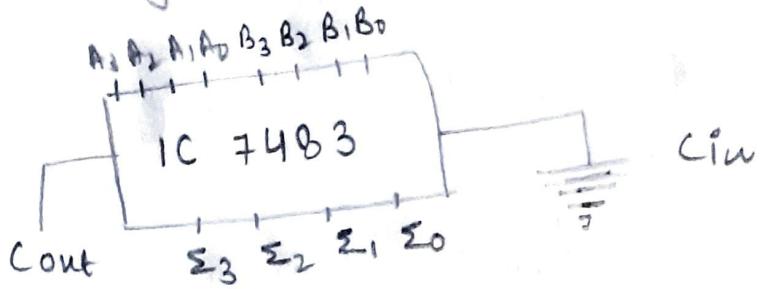


TRUTH TABLE

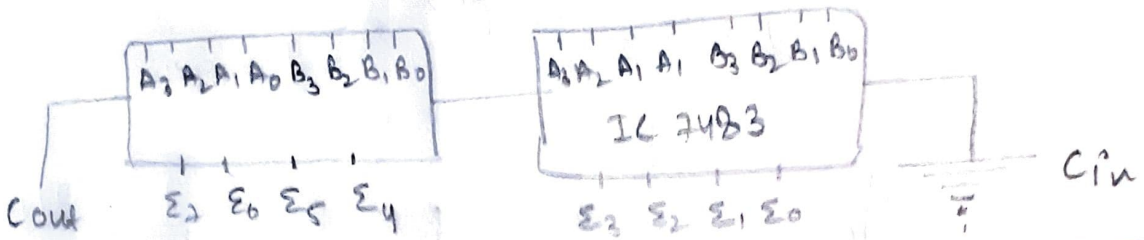
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	C_{out}	Σ_3	Σ_2	Σ_1	Σ_0
0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0	0
1	1	0	0	1	1	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1	0	1	1	1	1
1	1	0	0	0	1	1	0	1	0	0	1	0
1	0	0	0	0	0	0	1	0	1	0	0	1
0	1	0	0	0	1	0	1	0	1	0	0	1

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	C_{out}	Σ_3	Σ_2	Σ_1	Σ_0
1	1	0	1	0	0	1	1	1	0	0	0	0
0	1	0	0	1	0	1	0	0	1	1	1	0
1	1	1	0	1	1	0	0	1	1	0	1	0
1	0	1	1	0	0	1	0	0	1	1	0	1
1	1	1	1	0	1	0	1	1	0	1	0	0
1	0	1	0	1	0	1	1	0	1	1	0	1
0	1	1	0	1	1	1	0	1	0	1	0	0
1	1	1	1	0	0	0	0	1	0	1	1	1

Circuit Diagram



8-BIT BINARY ADDER USING IC7483

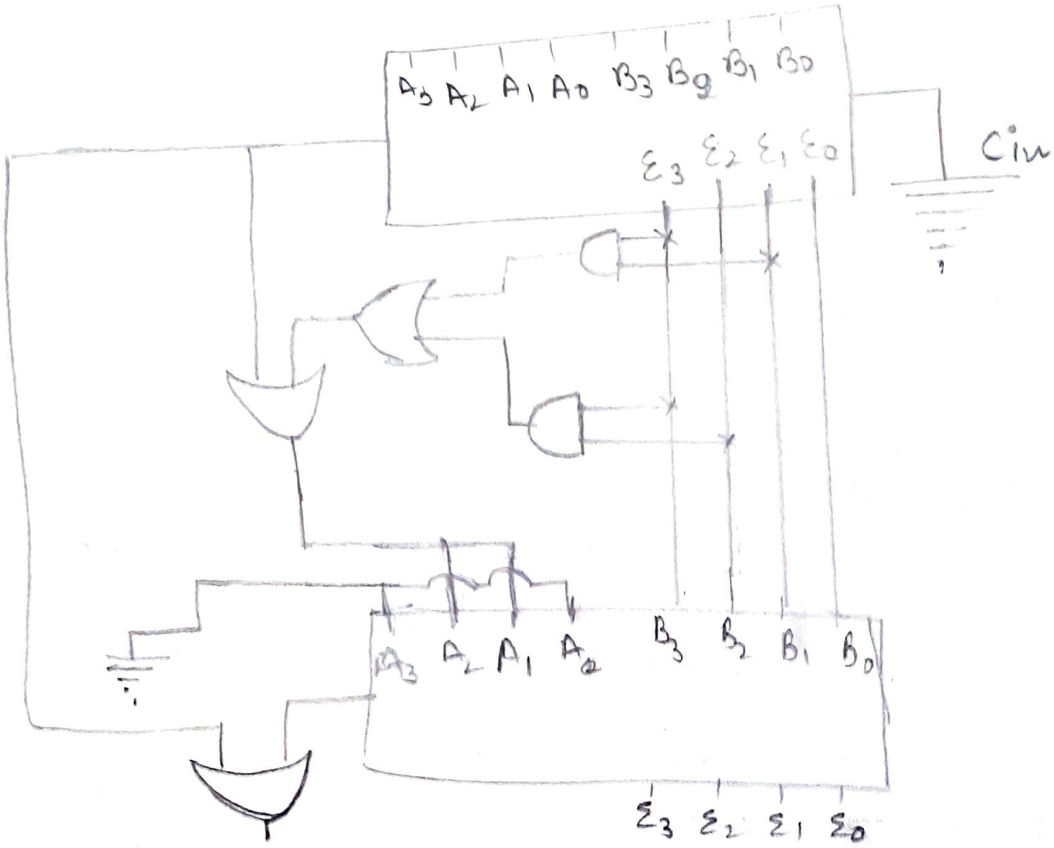


A7	A6	A5	A4	A3	A2	A1	A0	B7	B6	B5	B4	B3	B2	B1	B0	Ci	Σ7	Σ6	Σ5	Σ4	Σ3	Σ2	Σ1	Σ0
0	1	0	1	0	1	1	0	1	1	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0
0	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	1	1	0	1	0	1	1	1	1
0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0	1	0	1
0	0	1	0	1	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1	1	1	0	0	0
0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
1	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1	1	0	1	1	1	1	1	1	1
1	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
0	1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	1	1
1	0	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	0	1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1
1	0	1	1	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1

8-BIT BINARY ADDER USING IC7483

CIRCUIT DIAGRAM

4- BIT BCD ADDER USING IC74B3



4- BIT BCD ADDER Truth Table

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0		C_{out}	Σ_3	Σ_2	Σ_1	Σ_0
0	1	0	0	0	0	1	1	4+3=7	0	0	1	1	1
0	1	0	1	0	0	1	1	5+3=8	0	1	0	0	0
1	0	0	1	1	0	0	1	9+9=18	1	1	0	0	0
0	0	1	0	0	0	1	0	2+2=4	0	0	1	0	0
0	1	0	1	0	1	0	1	5+5=10	1	0	0	0	0
0	0	1	1	0	1	1	0	3+6=9	0	1	0	0	1
1	0	0	0	0	1	1	1	8+7=15	1	0	1	0	1
0	0	1	1	1	0	0	0	3+8=11	1	0	0	0	1

Conclusion

In this lab we learned to build 4-bit Binary adder using IC74B3, 8 Bit Binary Adder using IC 74B3 and 4 Bit BCD Adder using IC74B3

EXPERIMENT - 5.

Multiplexers & Demultiplexers

AIM :- Implement Boolean Expression using MUX74151, 74157, and DMUX 74154.
Conversion of 8:1 MUX using 2:1 MUX16:1, MUX using 4:1.

Hardware | Software | IC Required.

IC 74151, IC74157, IC74154, 74138.

Theory

Multiplexer : It is a device that has multiple input & single line output.
The select lines determine which input is connected to the output. and also to incr the amount of data that can be sent over a network within certain time.
It is also called a data selector.

8:1 MUX : This type of MUX takes 8 inputs and gives single output, also it has 3 select lines

2:1 MUX : This MUX takes 2 input & 1 output is given and it has a 3 sel

Boolean Expression

$F(A,B,C,D) = \Sigma m(7,9,12,15)$ using MUX IC 74151

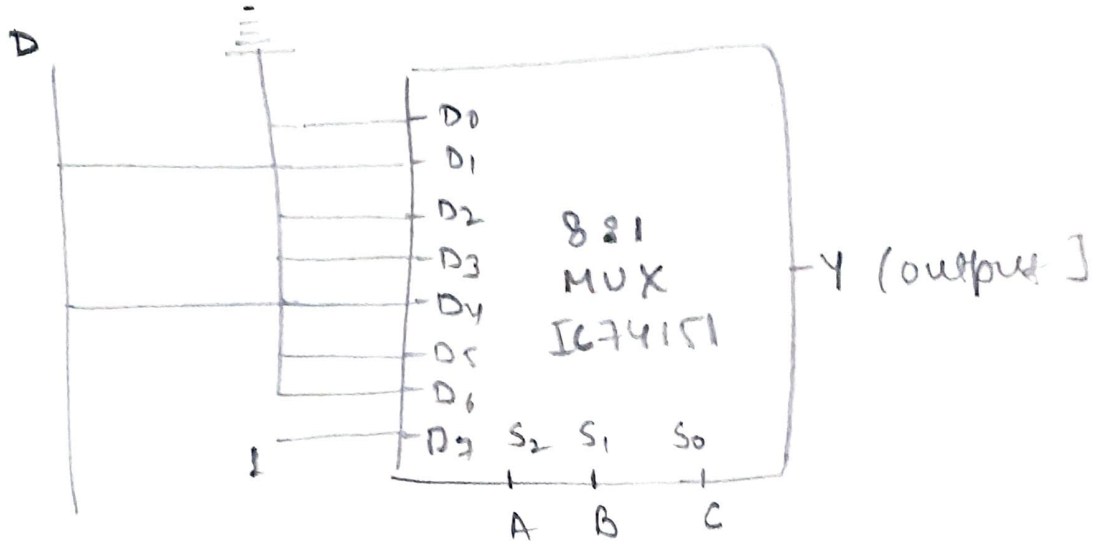
Truth table

Decimal	Binary	output
0	0000	0
1	0001	0
2	0010	0
3	0011	0
4	0100	0
5	0101	0
6	0110	0
7	0111	1
8	1000	0
9	1001	1
10	1010	0
11	1011	0
12	1100	1
13	1101	0
14	1110	0
15	1111	1

$A = B_3$
$B = B_2$
$C = B_1$
$D = B_0$

K-map.

	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$	$B_3B_2B_1$
$\overline{B_0}$	0	0	0	0	0	0	0	1
B_0	0	1	0	0	1	0	0	1
	0	B_0	0	0	B_0	0	0	1
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7

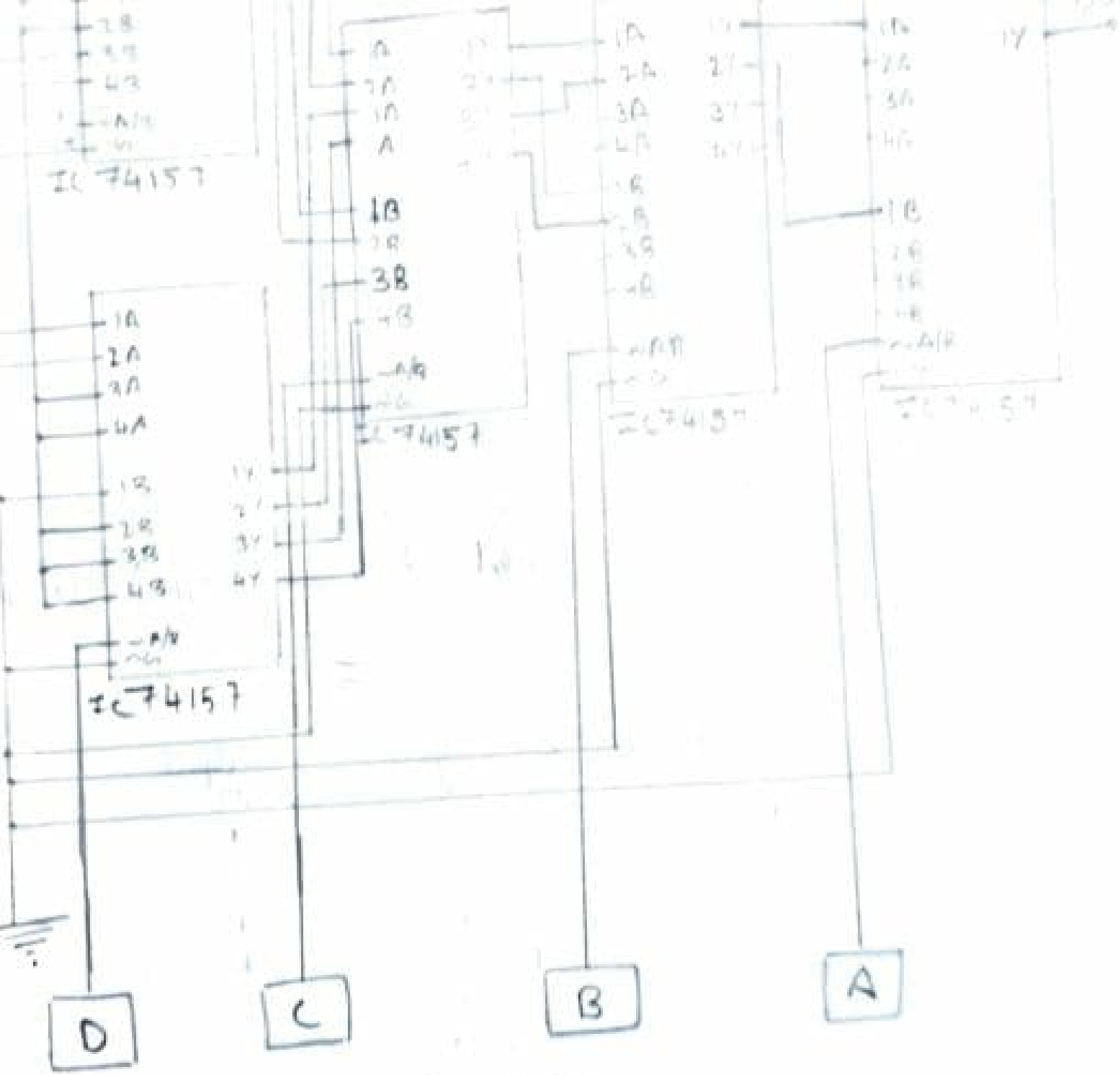


Boolean expression.

$F(A,B,C,D) = \Sigma M(1,3,4,11,12,13,14,5)$ using MUX IC 74157.

Truth table

Decimal	Binary	Output
0	0000	0
1	0001	1
2	0010	0
3	0011	1
4	0100	1
5	0101	0
6	0110	0
7	0111	0
8	1000	0
9	1001	0
10	1010	0
11	1011	1
12	1100	1
13	1101	1
14	1110	1
15	1111	1



Boolean Expression

$$f(A,B,C,D) = \sum m(1, 2, 4, 11, 12, 13, 14, 15)$$

Demultiplexer :-

It is a device that has single input & multiple outputs. The select line determines which input is connected to the output and also to incr the amount of data that can be sent over a network within certain time.

1:16 Demux :-

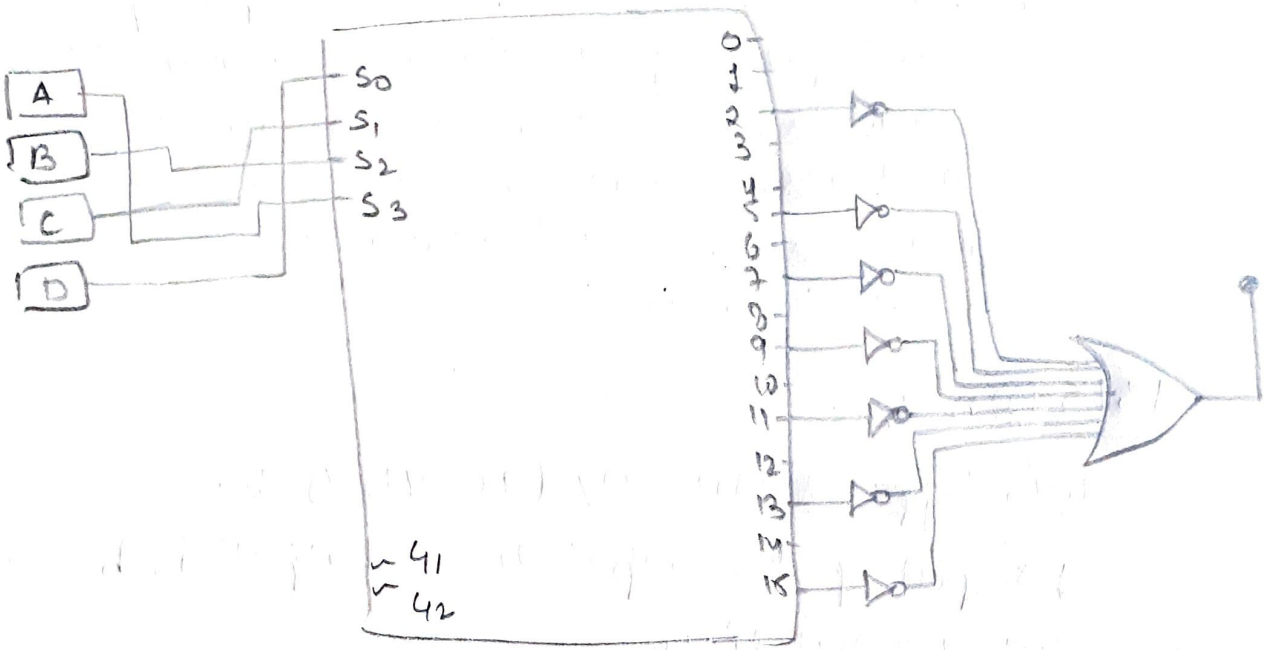
This type of Demux (IC74154) has 1 input & 16 output line. Output is based on 4 select line.

$F(A, B, C, D) = \Sigma m(2, 5, 7, 9, 11, 13, 15)$.

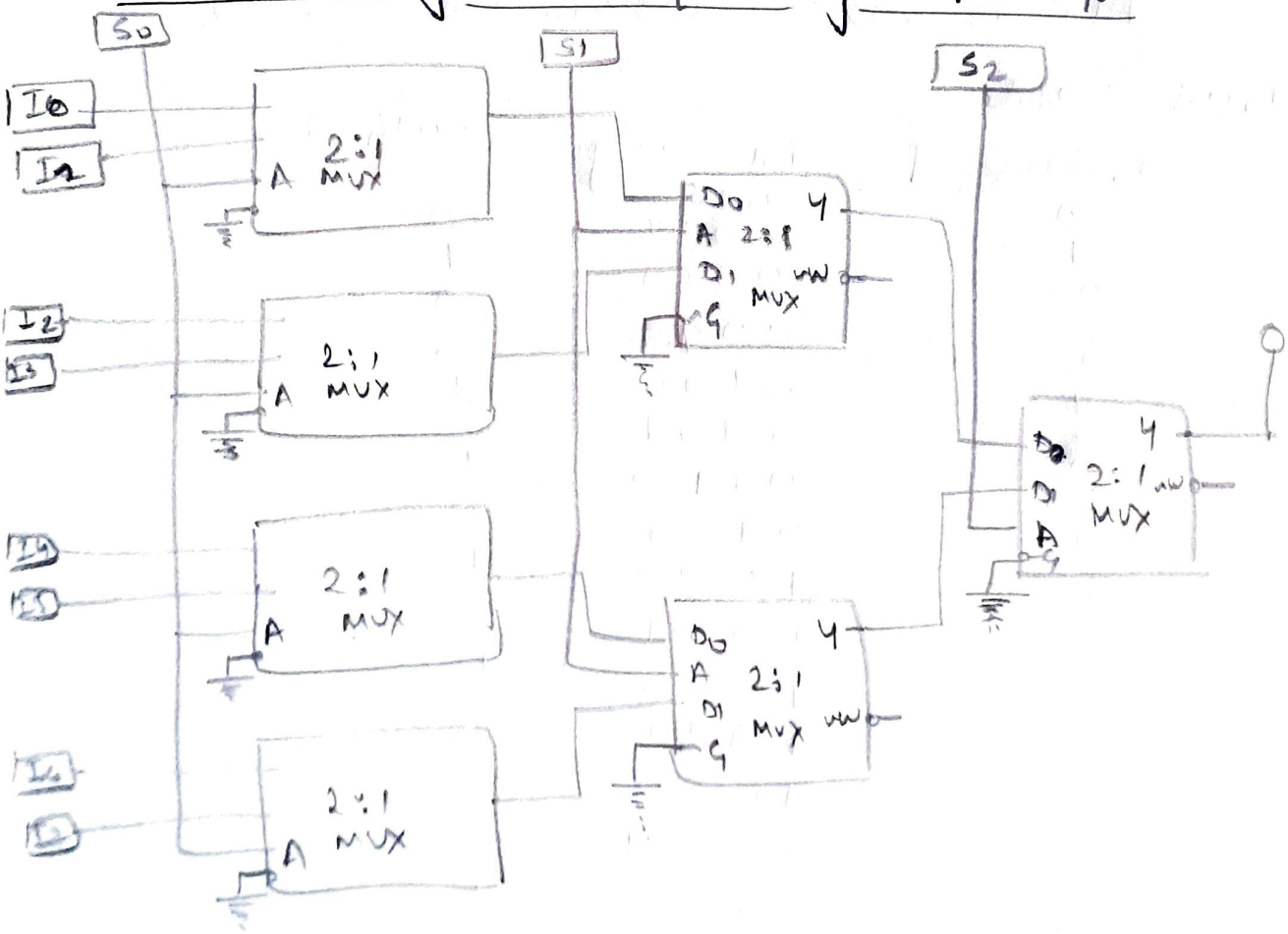
Truth Table

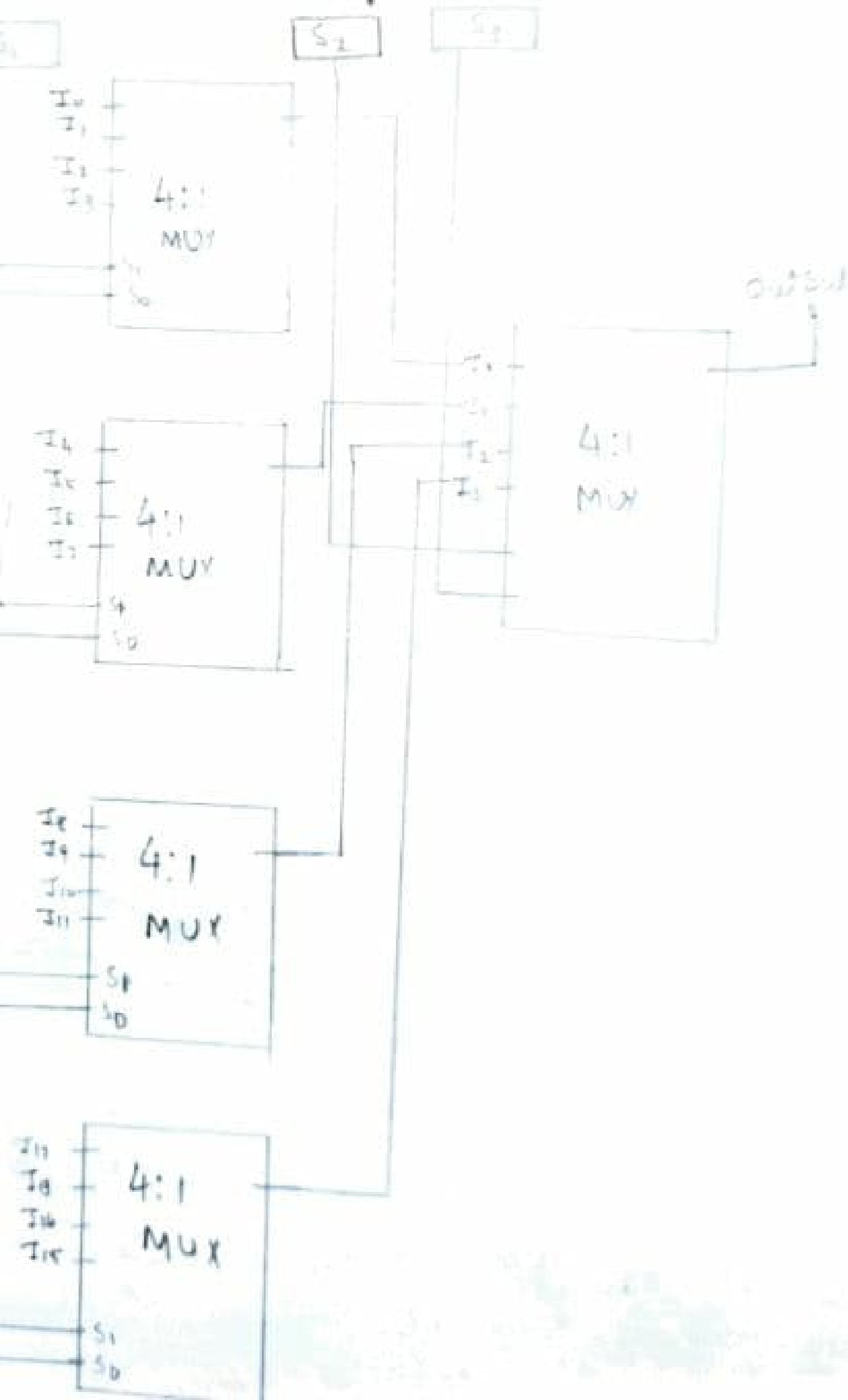
Decimal	Binary	Output
0	0000	0
1	0001	0
2	0010	1
3	0011	0
4	0100	0
5	0101	1
6	0110	0
7	0111	1
8	1000	0
9	1001	1
10	1010	0
11	1011	1
12	1100	0
13	1101	1
14	1110	0
15	1111	0

Circuit Diagram

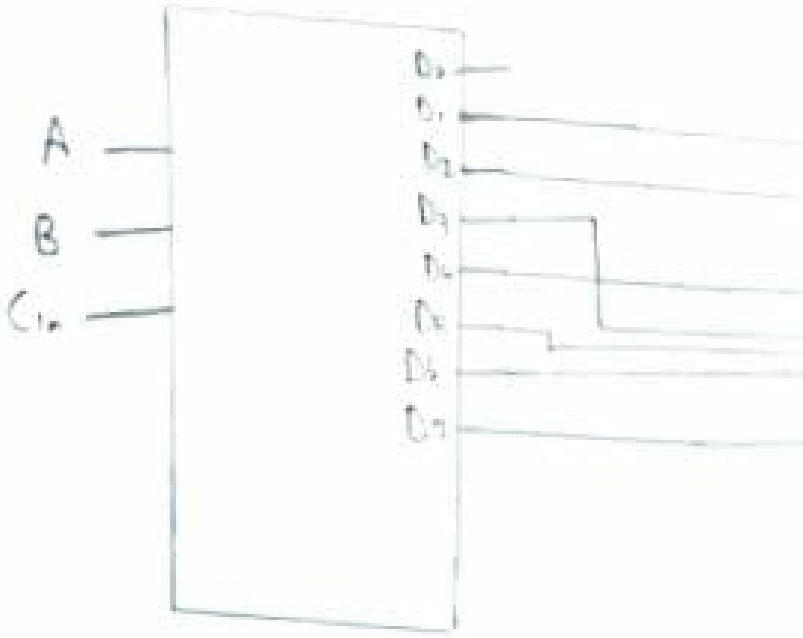


Conversion of 8:1 MUX using 2:1 MUX.




S2
4:1
MUX
4:1
MUX
4:1
MUX
4:1
MUX
4:1
MUX

A	B	C_{in}	Sum	Carry
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



Conclusion

In this lab we implemented Boolea using MUX & DEMUX, Conversion of Full adder, and converting 2:1 MUX and 4:1 MUX to 16:1 MUX

EXPERIMENT - 6

Parity Generator / Checker

AIM :- Design & Implement Parity Generator and checker using EX-OR.

Hardware / Software / Ic required

XOR Gate (IC 7486), X-NOR Gate (IC 4077), Multisim / circuit verse.

Theory

Even Parity Generator

If the data input contains odd no of 1's it will produce logic 1 or high and on even count of 1's it will given low output (0)

TRUTH TABLE

Input			Output
B_2	B_1	B_0	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Circuit diagram

k-map

$P = \bar{B}_2 \bar{B}_1 B_0 + \bar{B}_2 B_1 \bar{B}_0 + B_2 B_1 B_0 + B_2 \bar{B}_1 \bar{B}_0$

$= B_2 (\bar{B}_1 \bar{B}_0 + B_1 B_0) + \bar{B}_2 (\bar{B}_1 B_0 + B_1 \bar{B}_0)$

$= B_2 \oplus B_1 \oplus B_0$

2) Odd Parity generator

If data input has even no. of 1's output will be high(1) otherwise low(0).

TRUTHTABLE

Input			Output
B_2	B_1	B_0	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Circuit (odd Parity generator)

K-Map

B_0 \ B_2B_1	$\bar{B}_2\bar{B}_1$	$\bar{B}_2B_1$	B_2B_1	$B_2\bar{B}_1$
	1	0	1	0
	0	1	0	1

$$P = \bar{B}_2\bar{B}_1\bar{B}_0 + \bar{B}_2B_1B_0 + \bar{B}_2B_1B_0 + B_2\bar{B}_1B_0$$
$$= \bar{B}_2(\bar{B}_1\bar{B}_0 + B_1B_0) + B_2(\bar{B}_1B_0 + B_1\bar{B}_0)$$
$$= (B_2 \oplus B_1) \odot B_0$$

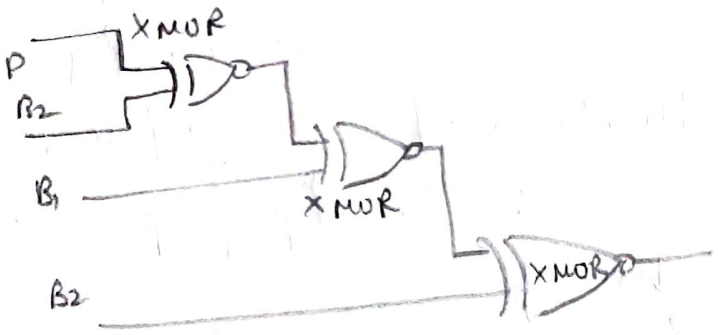
Even Parity checker.

It will produce a logic 1 at its output if data word contains even no. of 1's otherwise it'll give 0 (i.e. of odd ones)

Truth Table

Input				Output
P	B_2	B_1	B_0	E
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Circuit Diagram.



Even Parity Checker

K-Map

B_1B_0 \ PB_2	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

$$\boxed{E = P \text{ XOR } B_2 \text{ XOR } B_1 \text{ XOR } B_0}$$

Conclusion

Successfully designed and implemented Parity Generator & Checker.

EXPERIMENT - 7

Magnitude Comparitor

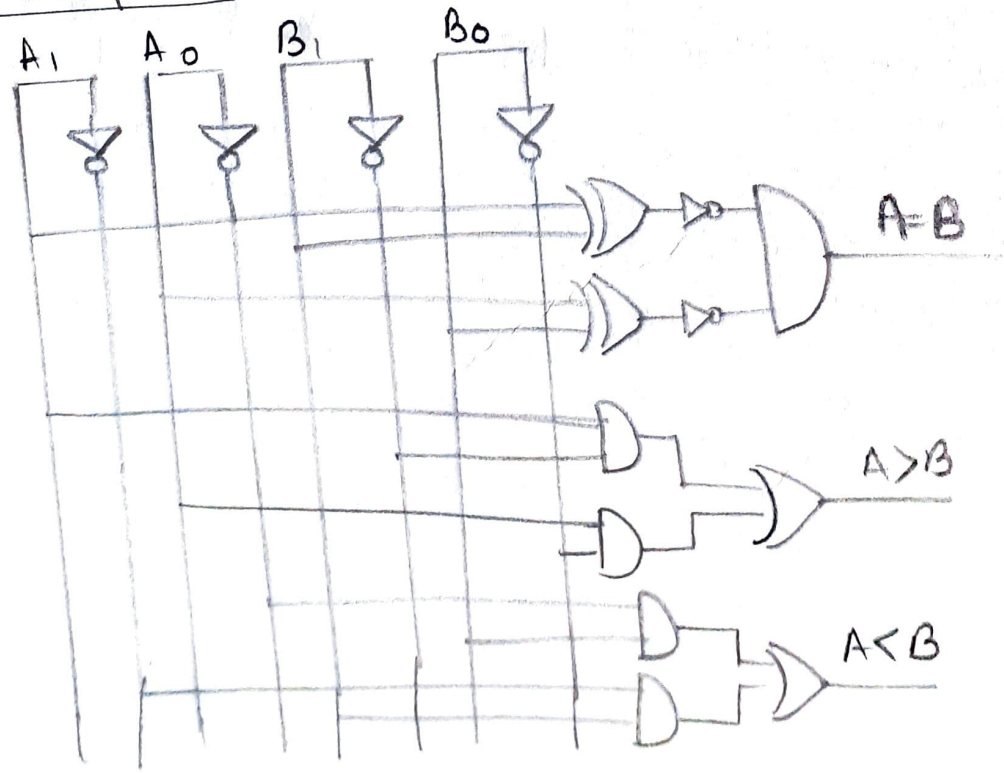
AIM: Design & implement 2, 3, 4 Magnitude comparator using logic gates 4-Bit-Bit magnitude comparator using IC 7485

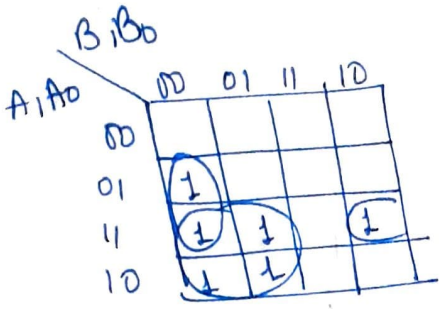
Software used → Circuit verse / Multisim14

THEORY The comparison of two no. is an operator that determine one no. is greater than, less than (or) equal to other no.

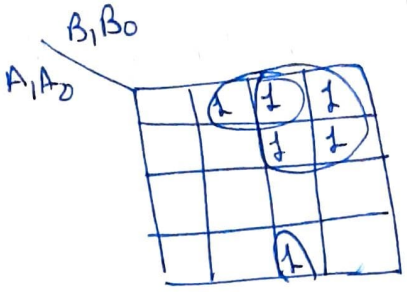
The outcome of the comparator is specified by three Binary value that indicate whether A > B, A = B or A < B

LOGIC DIAGRAM

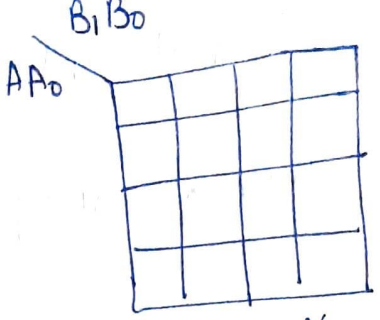




$A > B = A_0\bar{B}_0 B_1 + A_1\bar{B}_1 + A_1 A_0 \bar{B}_0$



$A < B = \bar{A}_1\bar{A}_0 B_0 + \bar{A}_0 B_0 B_1 + \bar{A}_1 B_1$



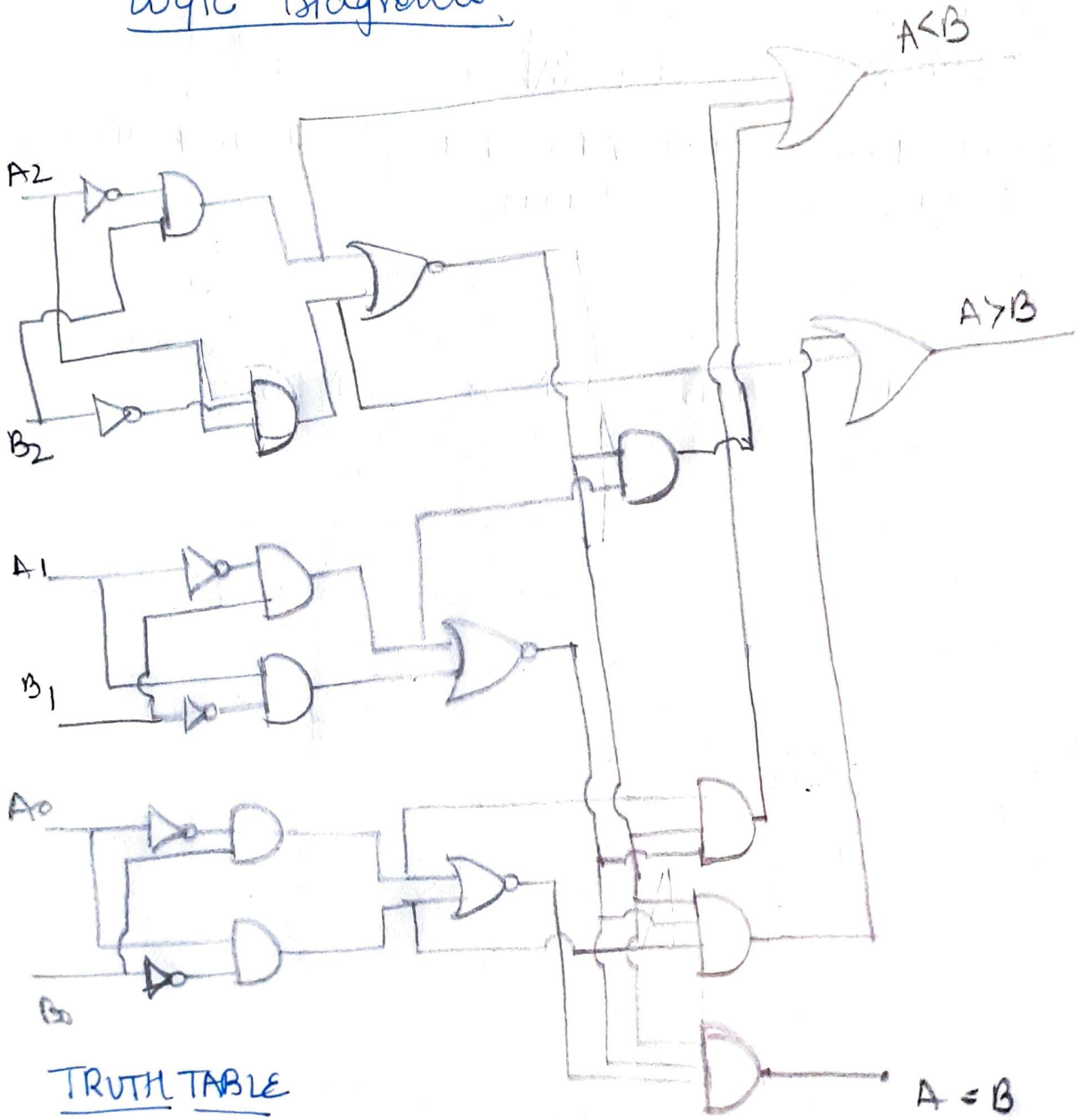
$A = B = (A_0 \oplus B_0)(A_1 \oplus B_1)$

Truth Table

A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

3-Bit Magnitude Comparator

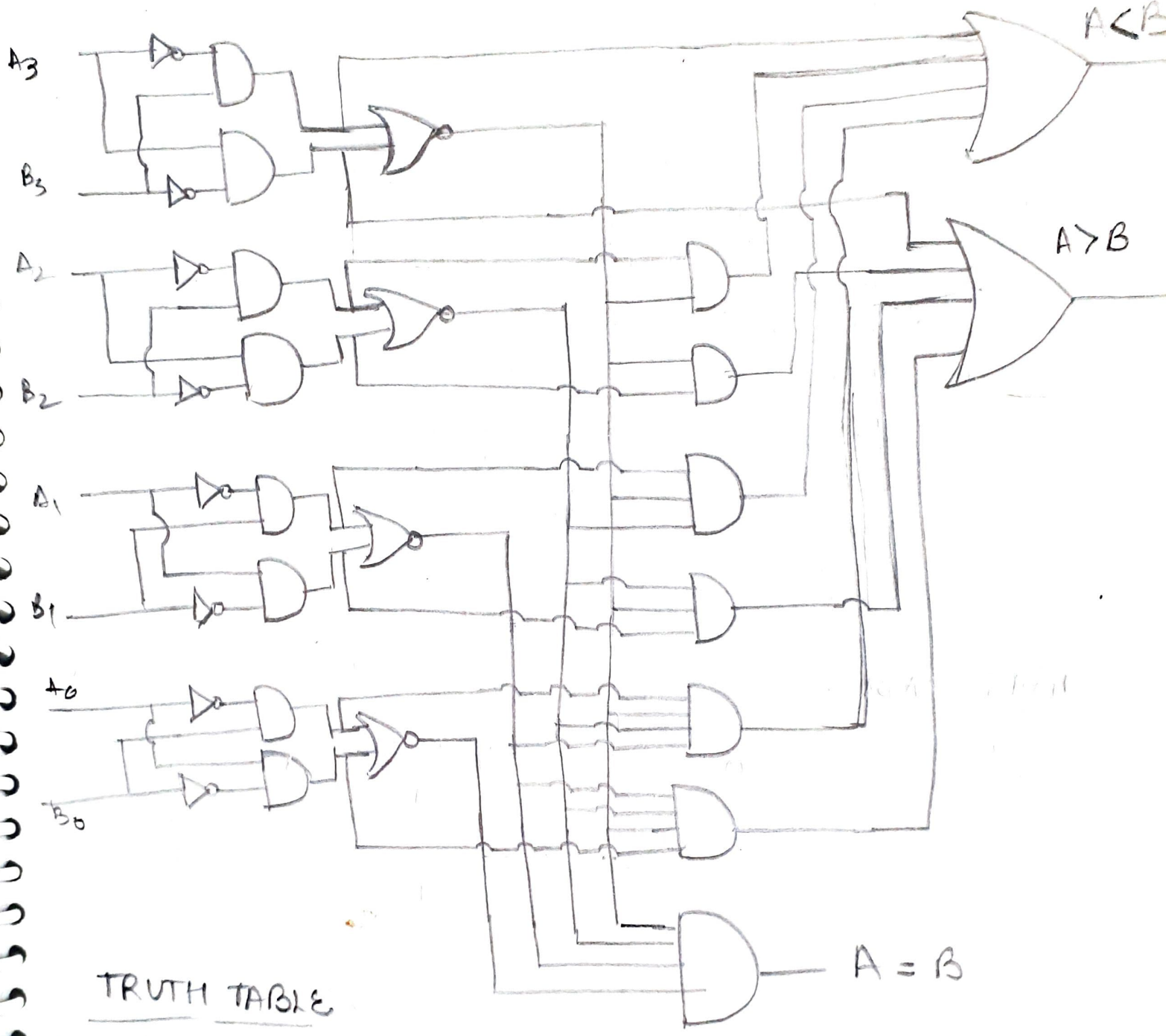
Logic Diagram



TRUTH TABLE

A_2	A_1	A_0	B_2	B_1	B_0	A<B	A = B	A>B
0	0	0	0	0	0	0	1	0
0	1	0	0	1	1	1	0	0
1	0	1	1	0	0	0	0	1
1	1	1	1	0	1	0	0	1
1	1	1	0	1	0	0	0	1
1	0	0	1	0	0	0	1	0
0	0	0	0	1	0	1	0	0
1	0	1	1	1	1	1	0	0

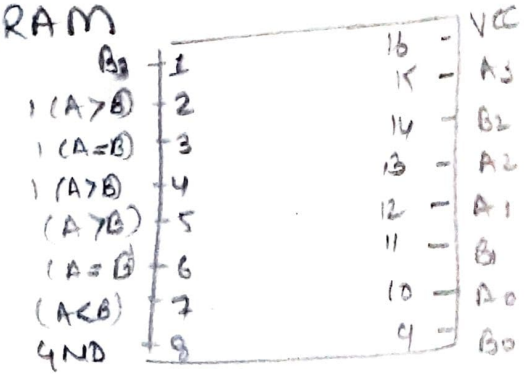
4-BIT Magnitude Comparator



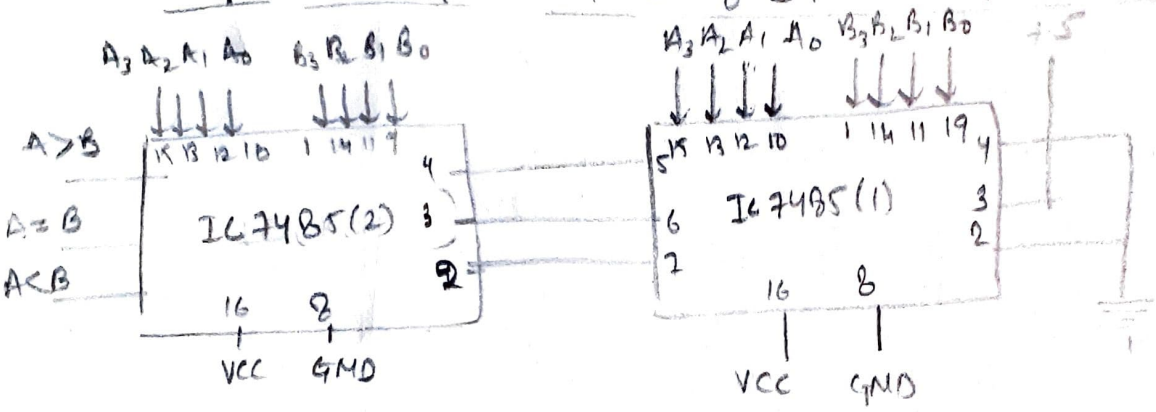
TRUTH TABLE

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	A<B	A = B	A>B
0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	1	0	1	1	0	0
0	1	1	1	1	0	0	0	1	0	0
1	1	0	0	1	0	1	0	1	0	0
1	1	1	1	0	1	0	0	0	0	1
1	1	0	1	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	0	1	1
0	0	1	0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0	1	0	0
0	0	0	1	1	0	0	1	0	1	0

PIN DIAGRAM



LOGIC DIAGRAM FOR 8 BIT - MAGNITUDE COMPARATOR



TRUTH TABLE

A	B	A>B	A=B	A<B
0000	0000	0	1	0
0001	0000	1	0	0
0000	0001	0	0	1
0001	0001	0	0	1
0101	0000	1	0	0
1111	1111	0	1	0

CONCLUSION

Thus the design & Implentation of Magnitude comparator were done.

EXPERIMENT - 8

FLIP - FLOP CONVESIONS

AIM: Implementation of conversion of
i) SR to JK
ii) JK to SR
iii) JK to T
iv) JK to D
v) SR to D
vi) D to SR
vii) D to JK

Software used Circuit verse | multisim

Theory flip-flop

flip-flop is a sequential circuit driven by clock input either by positive edge or by -ve edge. It is Binary storage device capable of storing one of information.

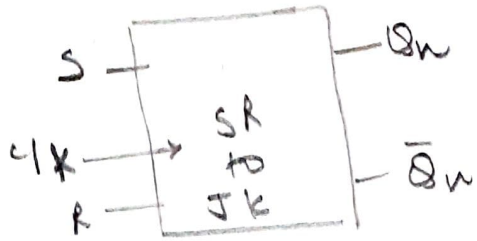
The flip-flop are of various types. Their characteristics can be easily seen from their truth table.

PRESENT & CLEAR

When the power is turn on, the state of flip flop is uncertain. It may come to set (Q=1) or reset (Q=0) state. In many application, it is necessary to initially set or reset the flip-flop. Such initial state of flip-flop can be accomplished by using the direct or asynchronous inputs of flip-flops. These inputs are Preset ($\bar{P}$) and clear ($\bar{C}$). They can be applied anytime b/w clock pulses and are not in synchronism with clock.

SR to JK flip-flop

Logic Diagram | SR to JK Flip-Flop |



Truth Table

J	K	Q_p	Q_n	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0 } Toggle
1	1	1	0	0	1

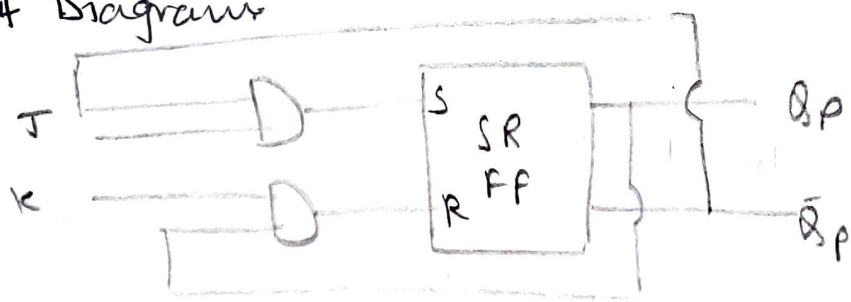
J \ KQp	00	01	11	10
0	0	X	0	0
1	1	X	0	1

$\boxed{S = J\bar{Q}_p}$

J \ KQp	00	01	11	10
0	X	0	1	X
1	0	0	1	0

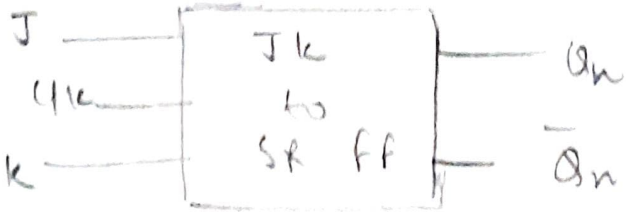
$\boxed{R = KQ_p}$

Circuit Diagram



JK – to SR flip flop

Logic Diagram



Truth Table

S	R	Q_p	Q_n	J	k
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	x	1
1	0	0	1	1	x
1	0	1	1	x	0
1	1	0	x	x	x
1	1	1	x	x	x

K-map

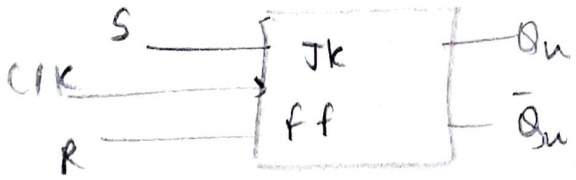
S \ RQP	00	01	11	10
0	0	x	x	0
1	1	x	x	x

$\boxed{J = S}$

S \ pQR	00	01	11	10
0	0	x	1	x
1	x	0	x	x

$\boxed{k = R}$

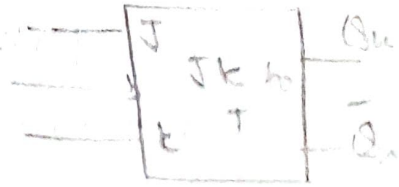
Circuit Diagram



Jk to T

Truth Table

I	Q_p	Q_n	J	k
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1



K-map

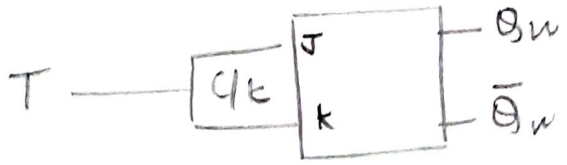
T \ Q_p	0	1
0	0	x
1	1	x

$\boxed{J = T}$

T \ Q_p	0	1
0	x	0
1	x	1

$\boxed{K = T}$

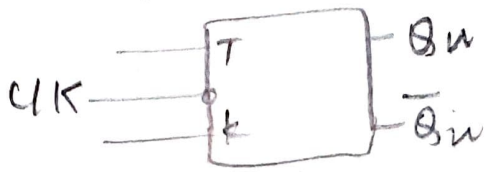
Circuit Diagram



JK to T flip-flop

JK to D flip-flop.

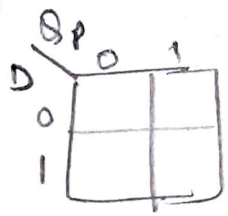
Logic Diagram



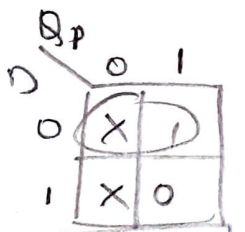
Truth table.

D	Q_P	Q_n	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

K-map.



$\boxed{J = D}$

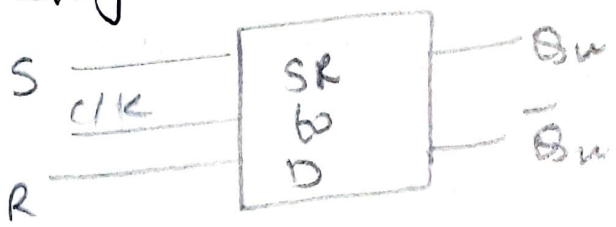


$\boxed{K = \bar{D}}$

Circuit - Diagram

SR to D.

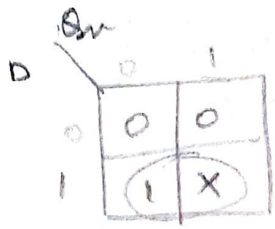
Logic Diagram:



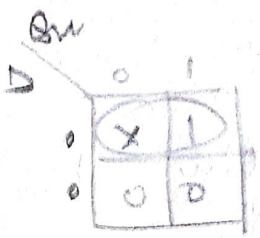
Truth table.

D	Qp	Qn	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

K-map.

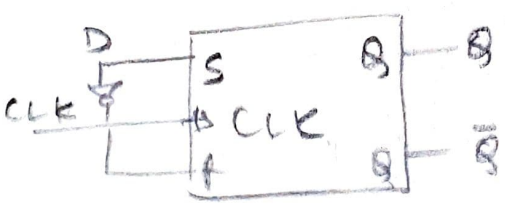


S = D



$R = \overline{D}$

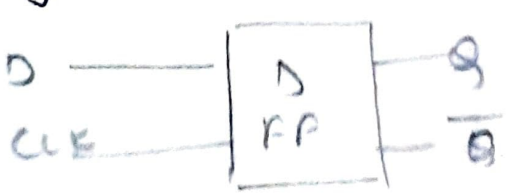
Circuit Diagram



SR to D

D to SR Flip-flop

Logic Diagram



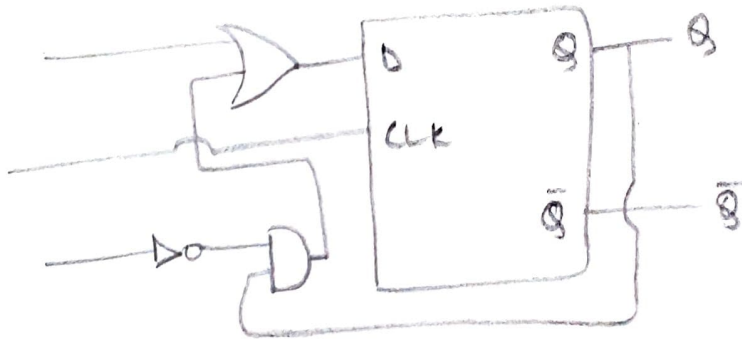
Truth table

S	R	Q_p	Q_n	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	Invalid		X
1	1	Invalid		X

K-map

S \ RQ_p	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$D = S + \bar{R}Q_p$



D to SR

D to JK

Logic Diagram

Truth table

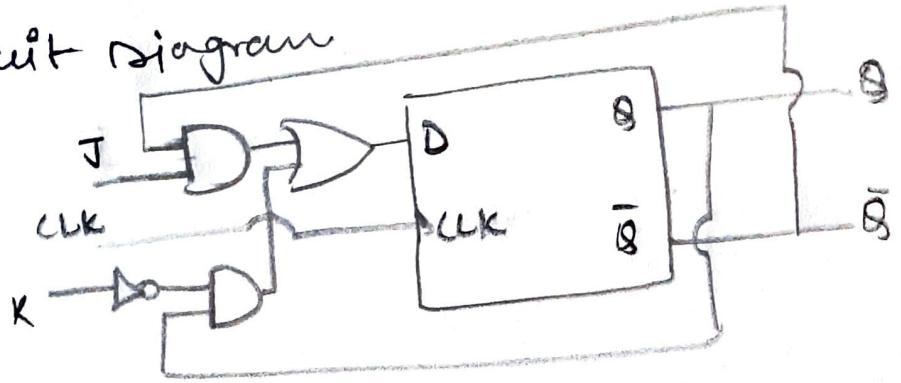
J	K	Q_p	Q_n	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

K-map

J \ KQn	00	01	11	10
0		1		
1	1	1		1

$D = J\overline{Q_n} + \overline{K}\,Q_n$

Circuit diagram



Conclusion

In this lab we learned and implement conversion SR to JK, JK to SR, JK to T, JK to D SR to D, D to SR and D to JK.

EXPERIMENT - 9

Asynchronous counter

AIM: Implementation of 2, 3, 4 Asynchronous up/down counter.

SOFTWARE: M | Multisim | Circuit verse

Requirement - IC trainer kit, Patch chords, IC7646

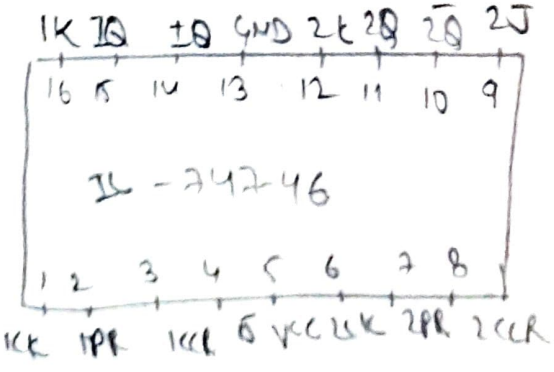
THEORY ASYNCHRONOUS COUNTERS

In this counter, flip-flops are connected in such a way that output of first clock drives clock for the next flip-flop.

All the flip-flop are not clocked simultaneously logic circuits is very simple even for more number of states.

Main drawback of the counter is their low speed as the clock is propagated through no. of flip-flops before it reaches that flip-flop

PIN - CONFIGURATION

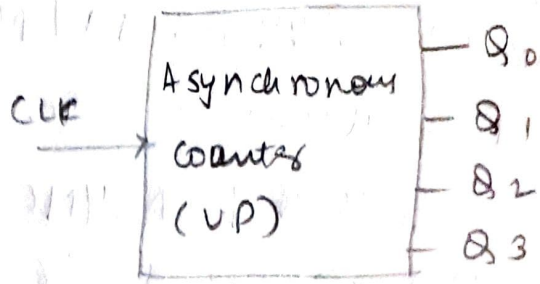


4-Bit - Asynchronous up-counter

TRUTH TABLE For All 4

PS				NS				J	K
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J	K
0	0	0	0	0	0	0	1	1	1
0	0	0	1	0	0	1	0	1	1
0	0	1	0	0	0	1	1	1	1
0	0	1	1	0	1	0	0	1	1
0	1	0	0	0	1	0	1	1	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	1	1
0	1	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	1	1	1
1	0	0	1	1	0	1	0	1	1
1	0	1	0	1	0	1	1	1	1
1	0	1	1	1	1	0	0	1	1
1	1	0	0	1	1	0	1	1	1
1	1	0	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1

LOGIC DIAGRAM

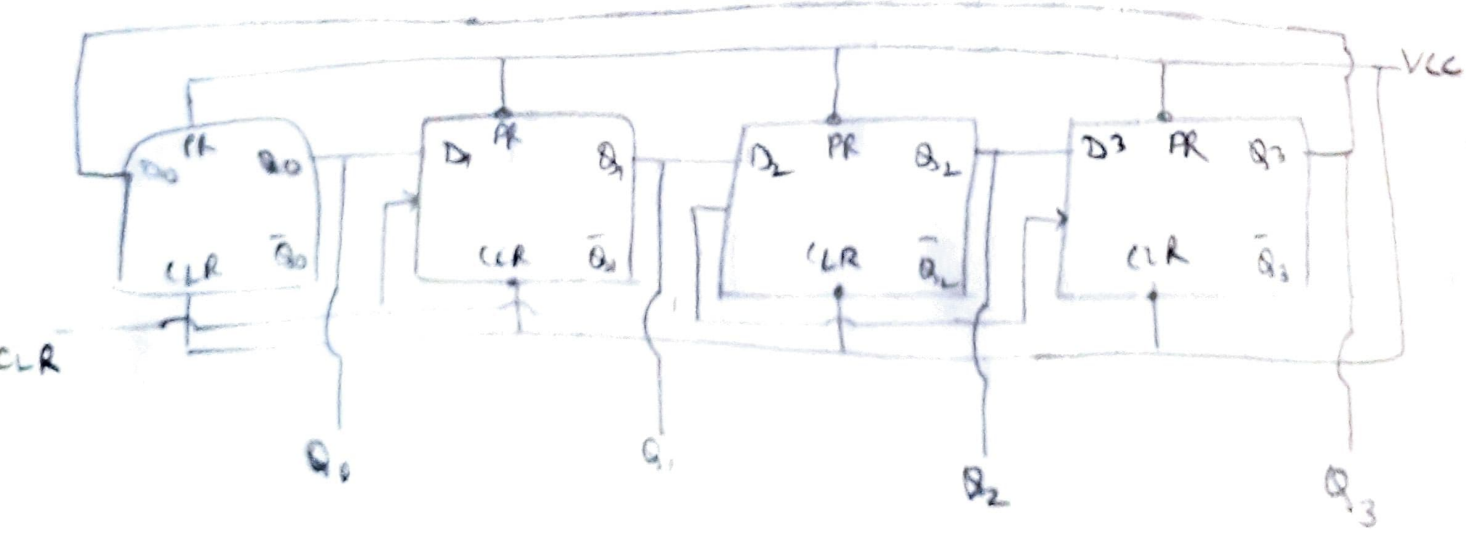


K-MAP

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

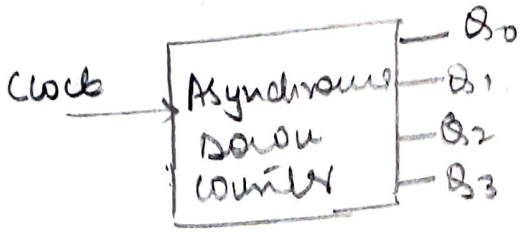
J = 1, K = 1 for All JK flip-flop

CIRCUIT DIAGRAM



4-Bit Asynchronous Down Counter.

LOGIC DIAGRAM



TRUTH TABLE

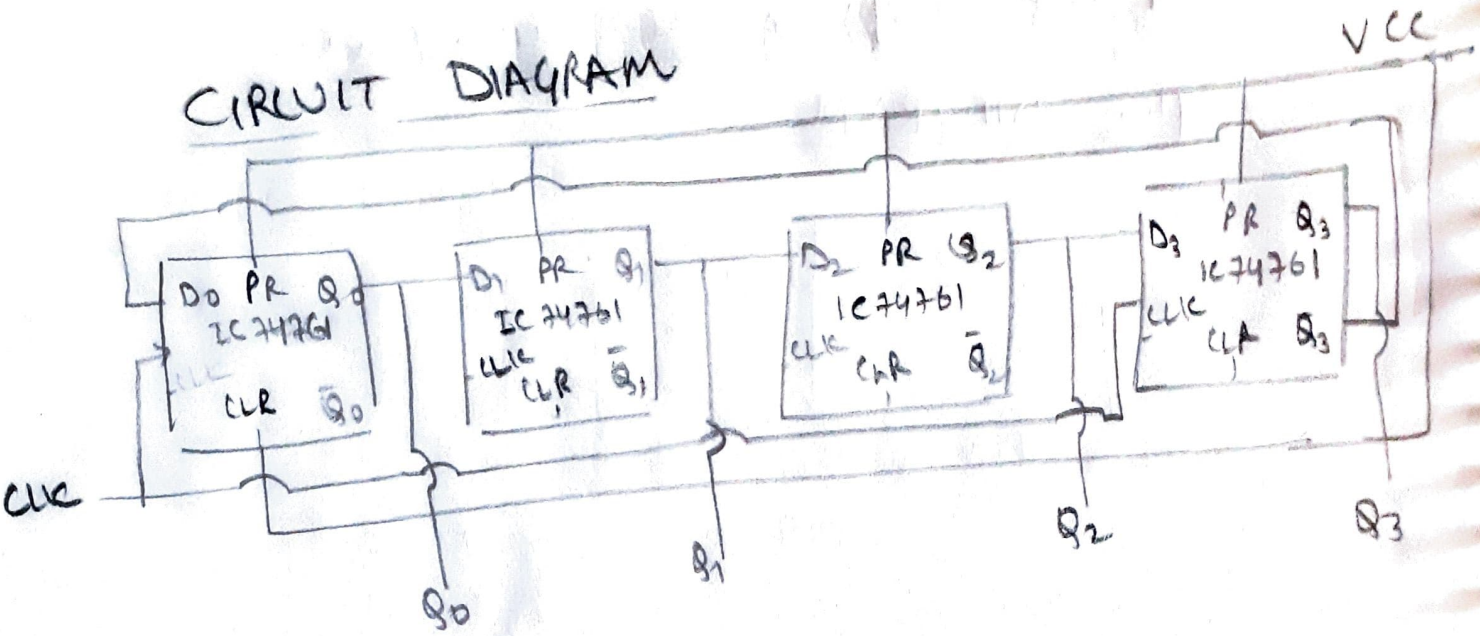
PS				NS				for All 4	
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J	K
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	1	1	1
0	0	1	1	0	0	1	0	1	1
0	1	0	0	0	0	1	1	1	1
0	1	0	1	0	1	0	0	1	1
0	1	1	0	0	1	0	1	1	1
0	1	1	1	0	1	1	0	1	1
1	0	0	0	0	1	1	1	1	1
1	0	0	1	1	0	0	0	1	1
1	0	1	0	1	0	0	1	1	1
1	0	1	1	1	0	1	0	1	1
1	1	0	0	1	0	1	0	1	1
1	1	0	1	1	1	0	0	1	1
1	1	1	0	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1

K-MAP

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

J = 1
K = 1 } for all JK-FF

CIRCUIT DIAGRAM



CONCLUSION

In this LAB we implemented 2, 3, 4 Asynchronous up/down counter using JK-flip-flop.

EXPERIMENT - 10

SYNCHRONOUS COUNTERS

AIM – Synchronous counter (up & Down) Implementation of 2, 3 & 4 bit using J-K flip-flop.

REQUIREMENT - IC Trainer kit, Patch chords, JK flip flop (IC7476)

SOFTWARE - Multisim | Circuit Verse

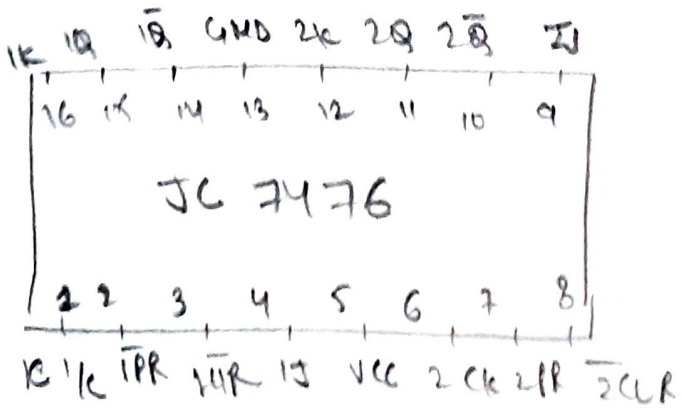
THEORY flip-flop

flip-flop is a sequential circuit driven by clock input either by positive edge. It is a binary storage device capable of storing one of information. The flip-flop are of various types. Their characteristic can be easily seen from their truth table.

PRESENT & CLEAR

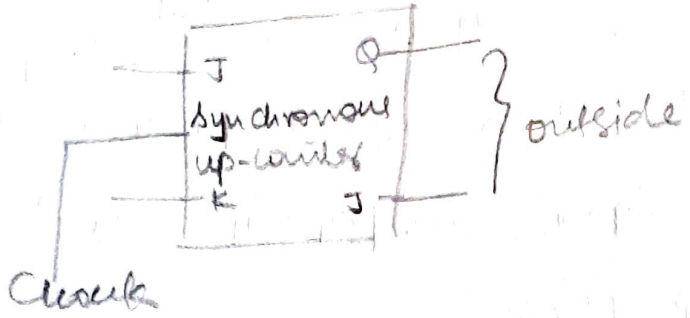
When Power is turned on the state of flip-flop is uncertain. It may come to set (Q=1) or reset (Q=0) state. In many applications it is necessary to initially set or reset the flip-flop these input are preset($\bar{P}$) and clear ($\bar{C}$). They can be applied at anytime b/w clock pulses and are not in synchronism with clock.

PIN CONFIGURATION



2, 3, 4 Bit Synchronous up-counter.

LOGIC DIAGRAM



Exitation table.

Previous State	Next-State	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

TRUTH TABLE

PS				NS											
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0	1	1	X	0	X	0	X	0	X
0	0	0	1	0	0	1	0	X	1	1	X	0	X	0	X
0	0	1	0	0	0	1	1	1	X	X	0	0	X	0	X
0	0	1	1	0	1	0	0	X	1	X	1	1	X	0	X
0	1	0	0	0	1	0	1	1	X	0	X	X	0	0	X
0	1	0	1	0	1	1	0	X	1	1	X	X	0	0	X
0	1	1	0	0	1	1	1	1	X	X	0	X	0	0	X
0	1	1	1	1	0	0	0	X	1	X	1	X	1	1	X
1	0	0	0	1	0	0	1	1	X	0	X	0	X	X	0
1	0	0	1	1	0	1	0	X	1	1	X	0	X	X	0
1	0	1	0	1	0	1	1	1	X	X	0	0	X	X	0
1	0	1	1	1	1	0	0	X	1	X	1	1	X	X	0
1	1	0	0	1	1	0	1	1	X	0	X	X	0	X	0
1	1	0	1	1	1	1	0	X	1	1	X	X	0	X	0
1	1	1	0	1	1	1	1	1	X	X	0	X	0	X	0
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

K-MAP

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	1	x	x	1
01	1	x	x	1
11	1	x	x	1
10	1	x	x	1

$J_0 = 1$

	00	01	11	10
00	x	1	1	x
01	x	1	1	x
11	x	1	1	x
10	x	1	1	x

$K_0 = 1$

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	0	1	x	x
01	0	1	x	x
11	0	1	x	x
10	0	1	x	x

$J_1 = Q_0$

Q_3Q_2 \ Q_0Q_1	00	01	11	10
00	x	x	1	x
01	x	x	1	x
11	x	x	1	x
10	x	x	1	x

$K_1 = Q_0$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	x	x	x	x
11	x	x	x	x
10	0	0	1	0

$J_2 = Q_1Q_2$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	x	x	x	x
01	0	0	1	0
11	0	0	1	0
10	x	x	x	x

$K_2 = Q_1Q_0$

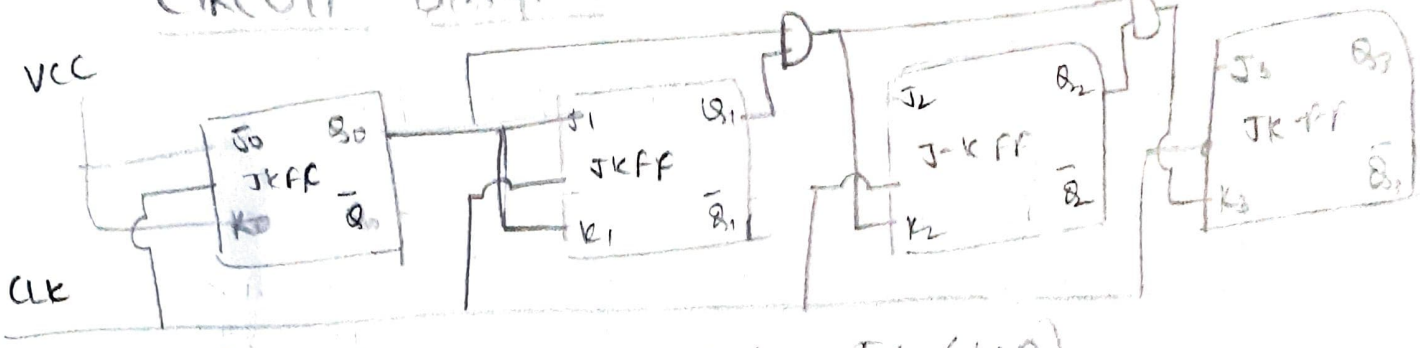
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	x	x	x	x
10	x	x	x	x

$J_3 = Q_2Q_1Q_0$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	0	0	1	0
10	0	0	0	0

$K_3 = Q_2Q_1Q_0$

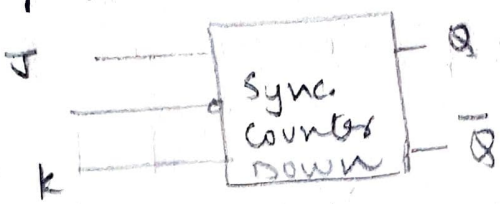
CIRCUIT DIAGRAM



four bit synchronous counter (up)

2, 3, 4 Bit Synchronous counter (Down)

LOGIC DIAGRAM.



EXITATION TABLE.

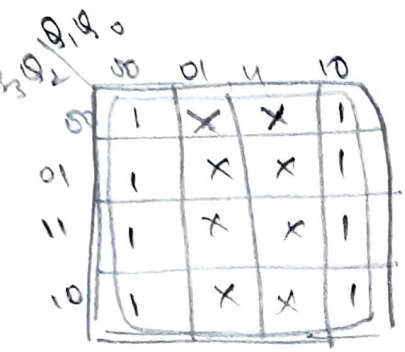
PS	NS	J	k
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Truth Table

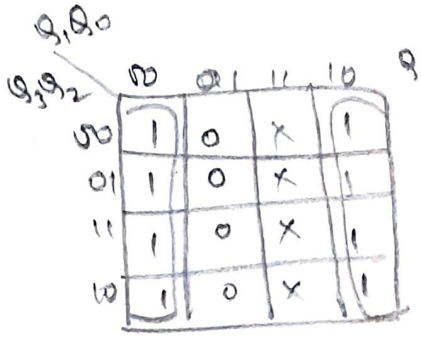
P S	N S	J_0	k_0	J_1	k_1	J_2	k_2	J_3	k_3
0 0 0 0	1 1 1 1	1	x	1	x	1	x	1	x
0 0 0 1	0 0 0 0	x	1	0	x	0	x	0	x
0 0 1 0	0 0 0 1	1	x	x	1	0	x	0	x
0 0 1 1	0 0 1 0	x	1	x	0	0	x	0	x
0 1 0 0	0 0 1 1	1	x	1	x	x	1	0	x
0 1 0 1	0 1 0 0	x	1	0	x	x	0	0	x
0 1 1 0	0 1 0 1	1	x	x	1	x	0	0	x
0 1 1 1	0 1 1 0	x	1	x	0	x	0	0	x
1 0 0 0	0 1 1 1	1	x	1	x	1	x	x	1
1 0 0 1	1 0 0 0	x	1	0	x	0	x	x	0
1 0 1 0	1 0 0 1	1	x	x	1	0	x	x	0
1 0 1 1	1 0 1 0	x	1	x	0	0	x	x	0
1 1 0 0	1 0 1 1	1	x	1	x	x	1	x	0

1	1	0	1	1	1	0	0	X	1	0	X	X	0	X	0
1	1	1	0	1	1	0	1	1	X	X	1	X	0	X	0
1	1	1	1	1	1	1	0	X	1	X	0	X	0	X	0

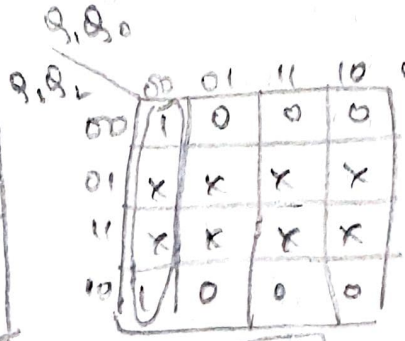
K-MAP



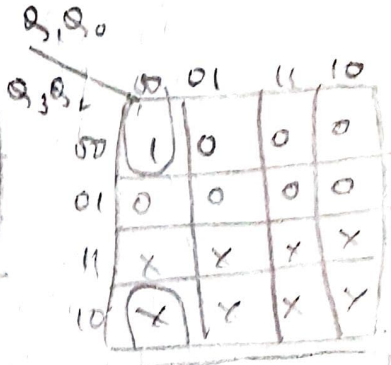
$J_0 = 1$
$K_0 = 1$



$J_1 = \bar{Q}_0$
$K_1 = Q_0$

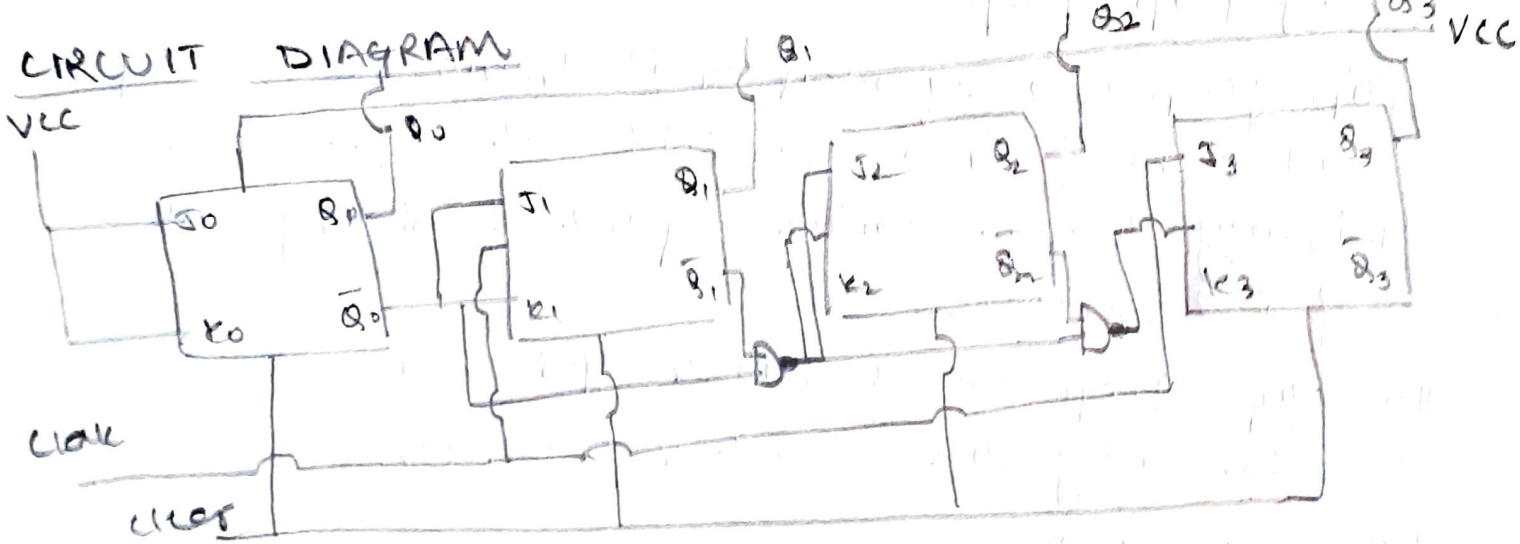


$J_2 = \bar{Q}_1 Q_0$
$K_2 = \bar{Q}\bar{Q}_0$



$J_3 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0$
$K_3 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0$

CIRCUIT DIAGRAM



CONCLUSION

In the Lab we learnt and build UP & DOWN Synchronous counters (2, 3, 4 bit)

EXPERIMENT - 11

MOD COUNTERS

AIM: Realization of MOD-N Counter using Decode Counter IC-7490 and to implement mode counter 4, 5. using 7476

THEORY:

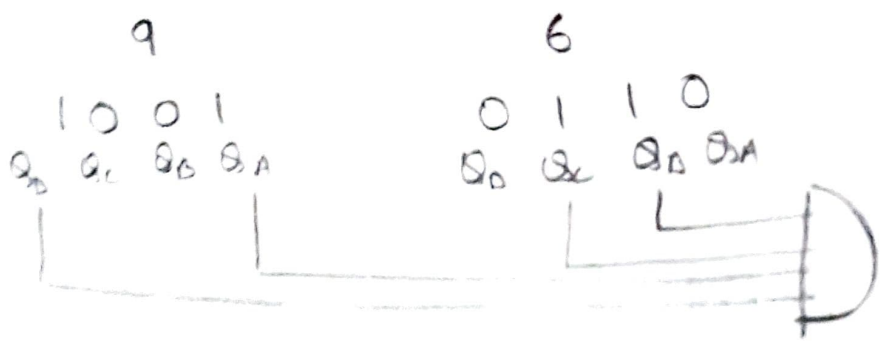
Decode Counter (IC 7490):- It is a simple counter which can count from 0-9. As it is a 4-Bit Binary decode counter, it has four output ports. Q_A, Q_B, Q_C, Q_D. When the count Reaches to Binary output reset to 0 (0000), Every time another pulse starts at pin no. 9. This IC 7490 has inbuilt divide by 2 and divide 5 counter in it.

MOD counters:

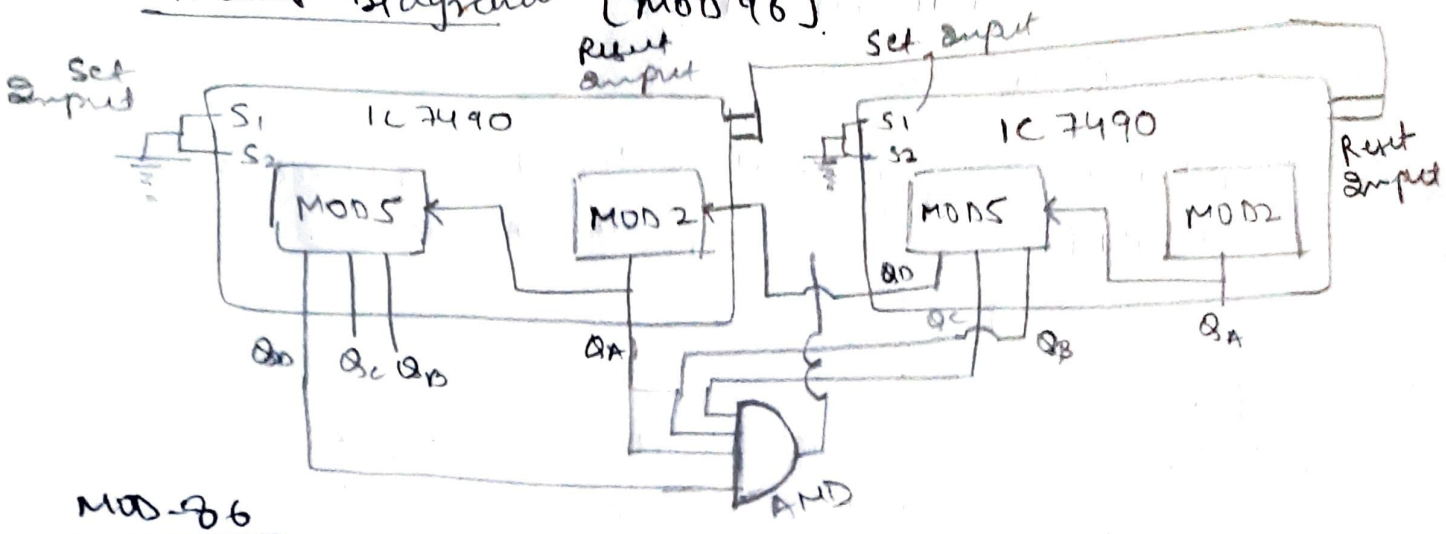
MOD counters are definitely based on no of state that the counter will sequence through before returning back to its original value

MOD-96

State Required 0, 1, 2, 3 94, 95...,

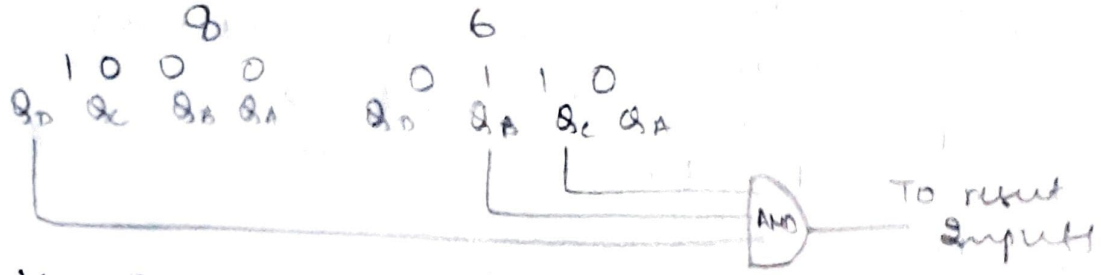


Circuit Diagram [MOD 96]

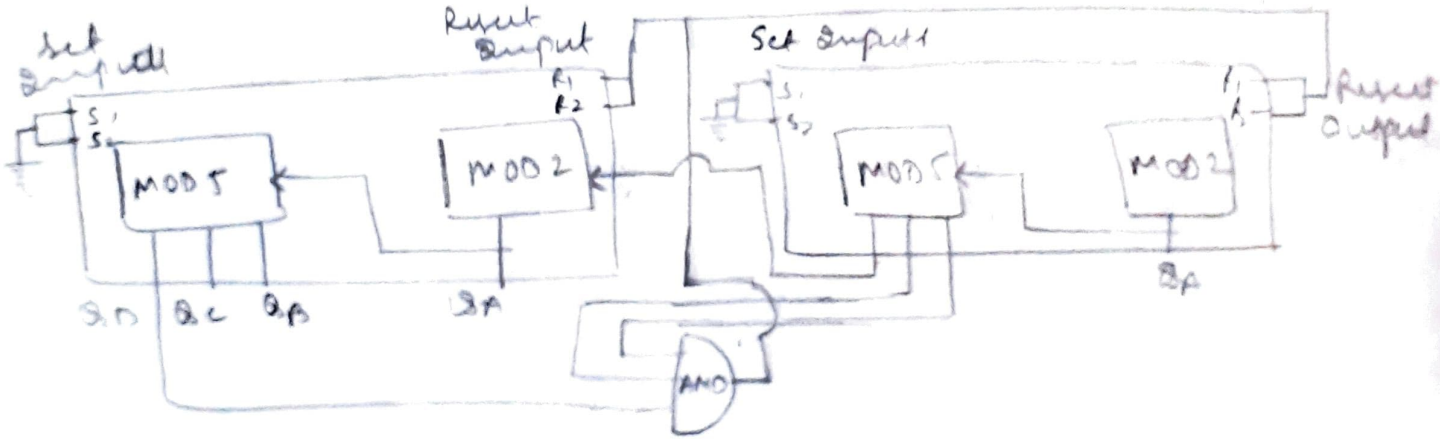


MOD-86

State Required - 0, 1, 2, ... 84, 85



Circuit Diagram (MOD 86)



MOD-5 counter
using IC 7476

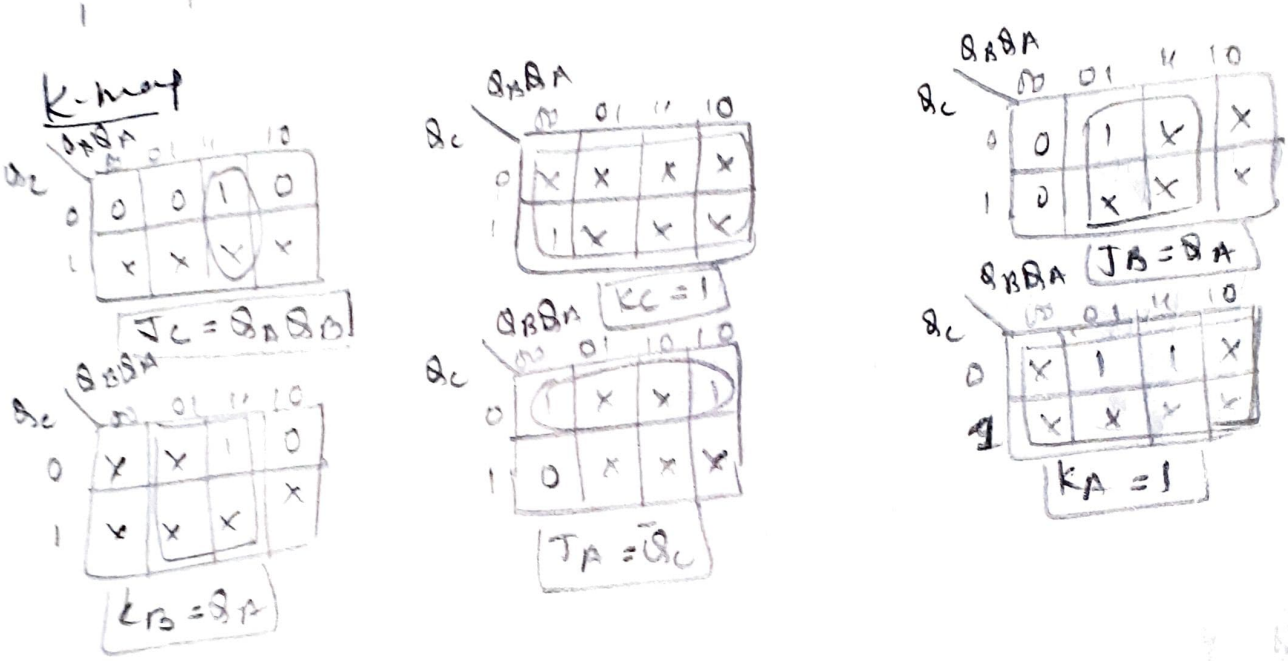
Excitation table for JK flip-flop.

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

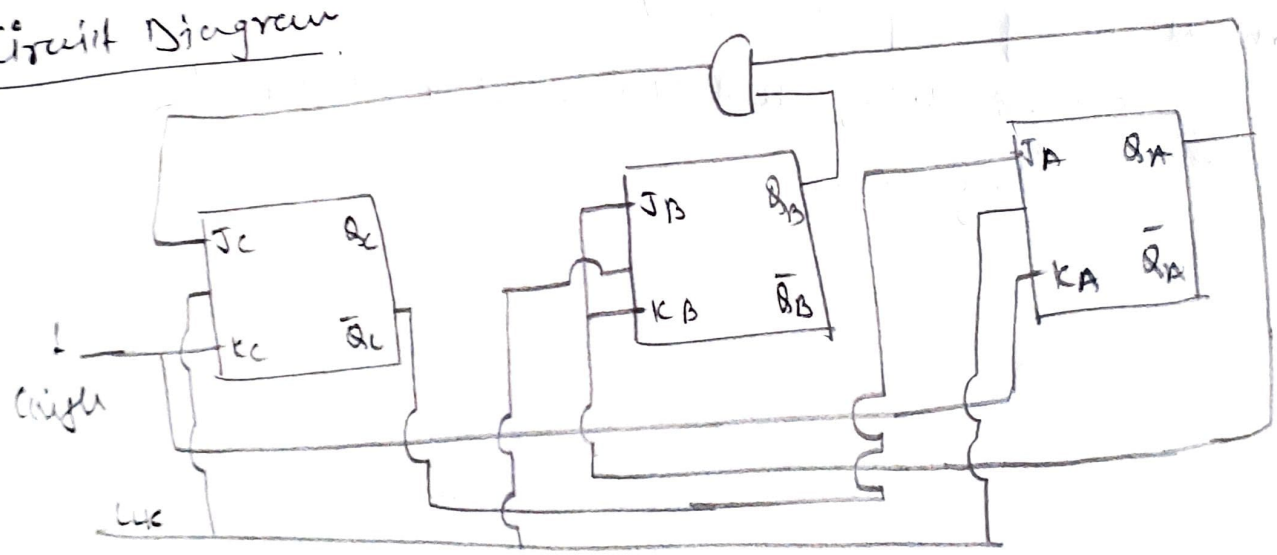
Truth table

Present State			Next State			Flip-flop Input					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	0	x
1	0	0	0	0	0	x	1	0	x	x	x
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

K-map



Circuit Diagram



MOD-4 Counter

Q_B	Q_A	Q_{B+1}	Q_{A+1}	J_B	K_B	J_A	K_A
0	0	0	1	0	x	1	x
0	1	1	0	1	x	x	1
1	0	1	1	x	0	1	x
1	1	0	0	x	1	x	1

K-Map

Q_B \ Q_A	0	1
0	0	1
1	x	x

$J_B = Q_A$

Q_B \ Q_A	0	1
0	x	x
1	0	1

$K_B = Q_A$

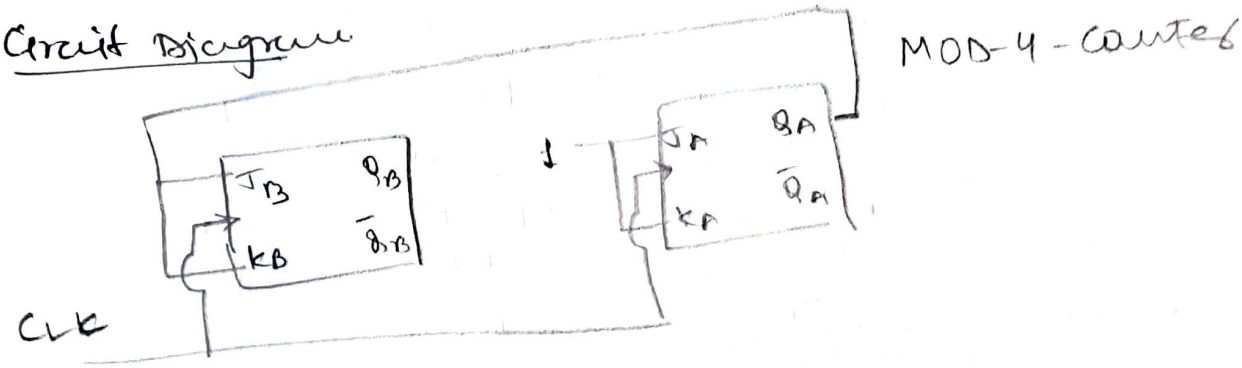
Q_B \ Q_A	0	1
0	1	x
1	1	x

$J_A = 1$

Q_B \ Q_A	0	1
0	x	1
1	x	1

$K_A = 1$

Circuit Diagram



Conclusion

Successfully implemented Mod 86, Mod 96 counter using IC7490 and Mod-4 Mod8 using IC 7476.

EXPERIMENT – 12

RING & JOHNSON'S COUNTER

AIM:

1) 4 Bit Ring counter using D flip-flop (IC 7474)

2) 4 Bit Johnson's counter using DFF

SOFTWARE Requirement

IC trainer kit, patch chords

IC7474, multisim / circuit maker

THEORY

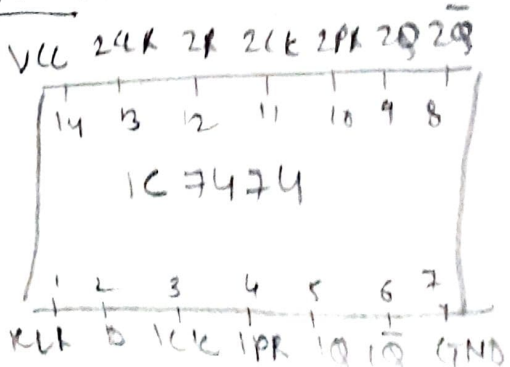
Ring counter

Ring counter is used in those applications in which several operations are to be controlled in a sequential manner. for Ex. in resistance welding the operation called squeeze, hold, weld and off are to be programed sequentially. we can use ring counter to initiate those operations

Johnson's Counter

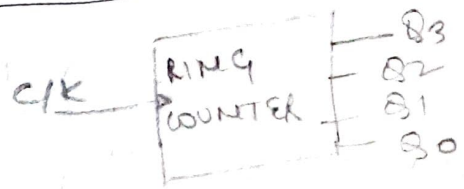
A Johnson counter is a modified Ring counter where the inverted output of the last flip-flop is connected to the input of first. The register cycles through a sequence of bit patterns. The mod. of the Johnson's counter is 2n if n-flip flop are used.

PIN configuration



RING COUNTER

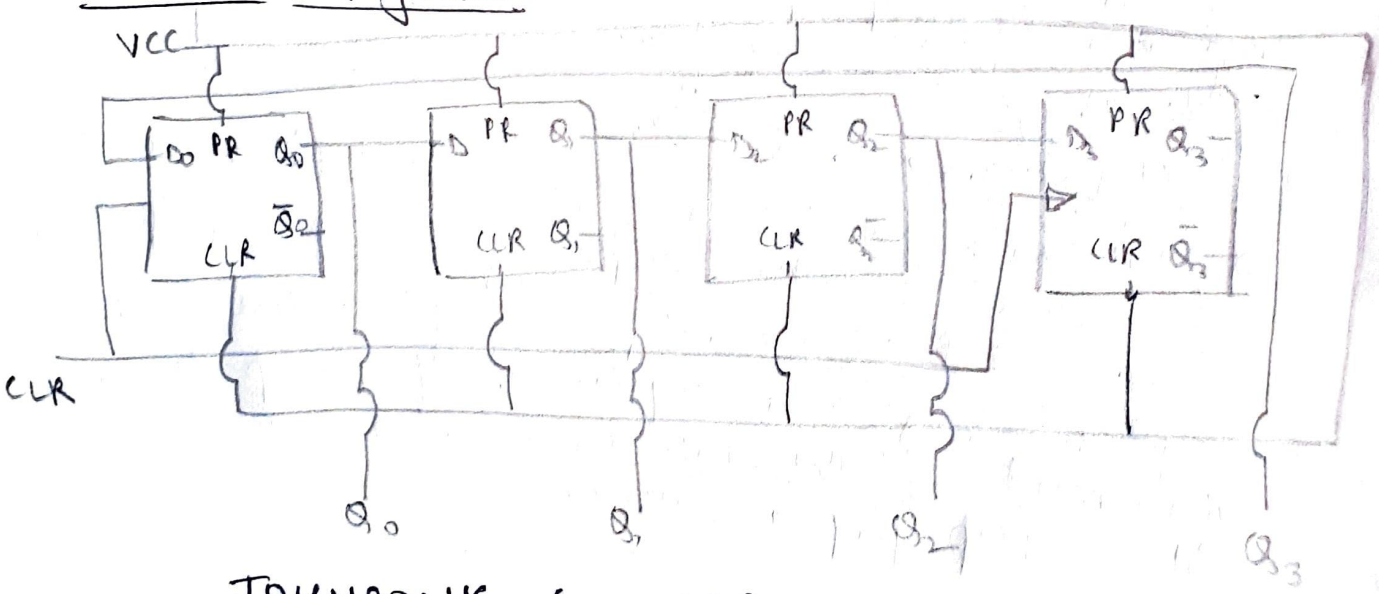
LOGIC DIAGRAM



Truth Table

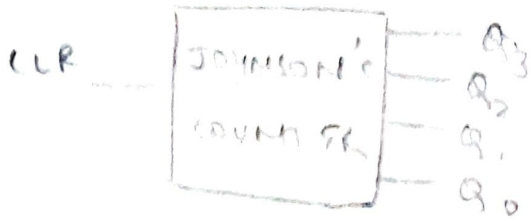
Q_3	Q_2	Q_1	Q_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Circuit Diagram



JOHNSON'S COUNTER

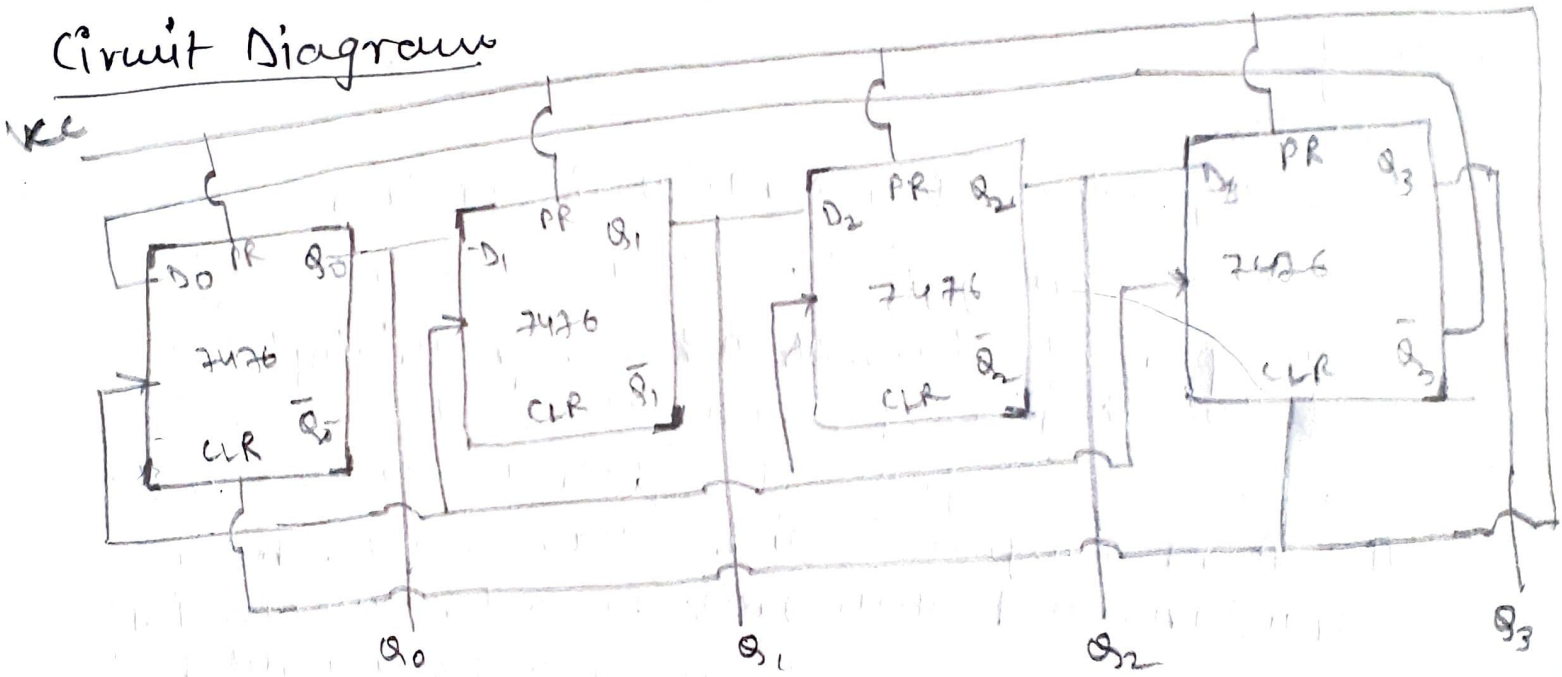
LOGIC DIAGRAM



Truth table

Q_3	Q_2	Q_1	Q_0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

Circuit Diagram



Conclusion

In this lab, we learnt to implement Ring counter and Johnson's Counter Ring IC7476 (D-FF)

EXPERIMENT - 13

SEQUENCE GENERATOR

AIM:- Design and implement sequence generator using JK flip-flop.

Hardware / Software / IC Requirement

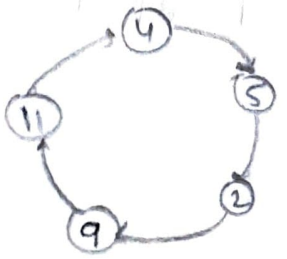
MS JK flip-flop, IC7408, IC7432, IC4077

Theory

Sequence Generator

A digital logic circuit whose purpose is to produce a prescribed sequence of outputs. Each output will be one of a no of symbols of Binary or q-ary logic levels. The sequence may be of indefinite length or of predetermined fixed length. A Binary counter is special type of sequence generator.

State Diagram



(4-5-2-9-11-4)

Exitation table

Present	Next	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State Table

Present				Next				flip-flop							
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
0	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x
0	0	1	0	1	0	0	1	1	x	0	x	x	1	1	x
0	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	0	1	0	0	x	x	1	1	x	x	1
0	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x
0	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x
1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
1	0	0	1	1	0	1	1	0	0	0	x	x	x	x	x
1	0	1	0	x	x	x	x	x	x	x	x	x	1	x	1
1	0	1	1	0	1	0	0	1	1	1	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x

J_3

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	x	x	x	1
01	0	0	x	x
11	x	x	x	x
10	x	x	x	x

$\boxed{J_3 = Q_1}$

K_3

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	x	x	x	x
10	x	0	1	x

$\boxed{K_3 = Q_1}$

J_2

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	x	x	x	0
01	x	x	x	x
11	x	x	x	x
10	x	0	1	x

$\boxed{J_2 = Q_1Q_0}$

K_2

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	x	x	x	x
01	0	1	x	x
11	x	x	x	x
10	x	x	x	x

$\boxed{K_2 = Q_0}$

J_1

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	X	X	X	X
01	0	1	X	X
11	X	X	X	X
10	X	1	X	X

$\boxed{J_1 = Q_0}$

K_1

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	X	X	X	1
01	X	X	X	X
11	X	X	X	X
10	X	X	1	X

$\boxed{K_1 = 1}$

J_0

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	X	X	X	1
01	1	X	X	X
11	X	X	X	X
10	X	X	X	X

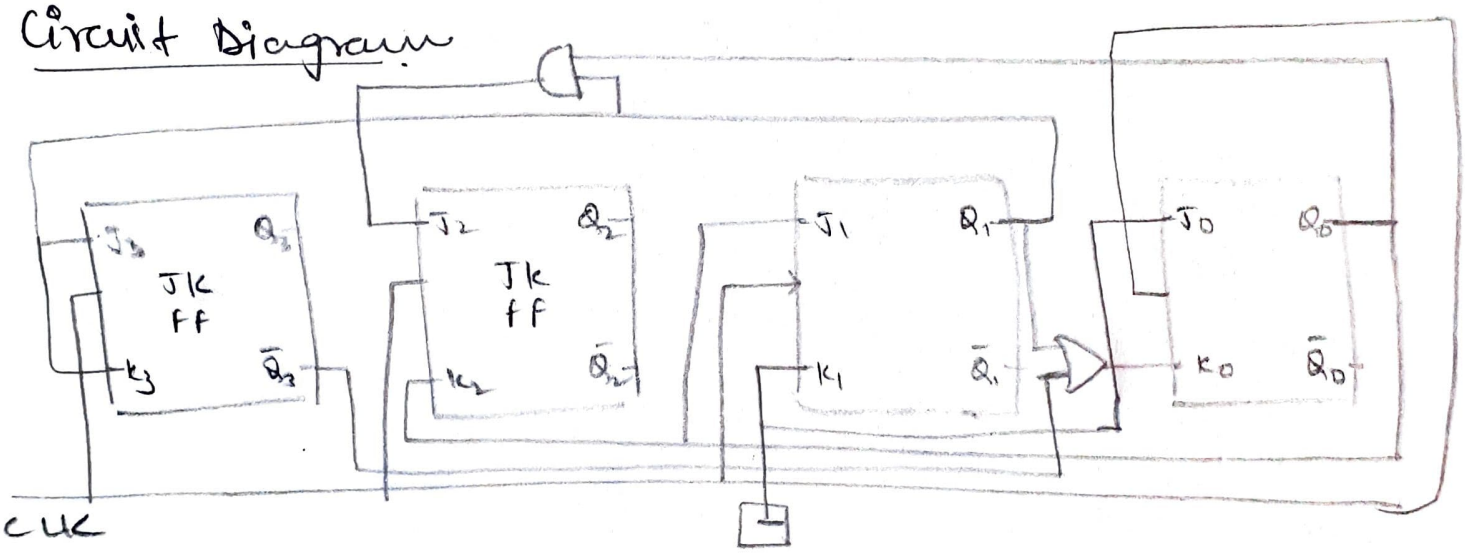
$\boxed{J_0 = 1}$

K_3

Q_2Q_3 \ Q_1Q_0	00	01	11	10
00	X	X	X	X
01	X	1	X	X
11	X	X	X	X
10	X	0	1	X

$\boxed{K_3 = \bar{Q}_3 + Q_1}$

Circuit Diagram



Conclusion

Successfully designed & implemented sequence generator for sequence (4-5-2-9-11)

EXPERIMENT - 14

SEQUENCE DETECTOR

AIM:- Design & Implement J-K flip-flop using MS-JK flip-flop.

Hardware | Software | IC required

MSJK-ff, Not Gate (IC7404) OR Gate (IC7432) AND Gate (IC7408) - Multisim

THEORY

Sequence Detector - A Sequence detector is a sequential state machine which takes an input string of bits and generating an output 1 whenever the target sequence has been detected. In a mealy machine, output depends upon the present state & the external input

To detect Sequence 110

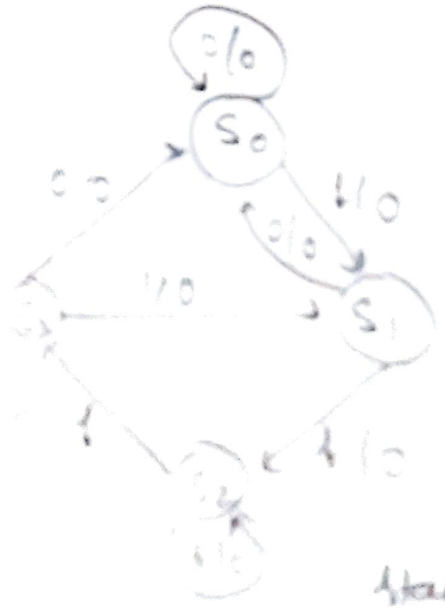
State Diagram

$S_0 = 0$

$S_1 = 1$

$S_2 = 11$

$S_3 = 110$



State Diagram

Input X = 0 1 1 1 0 1 1 0 1 0 1 0 0

Output Y = 0 0 0 0 1 0 0 1 0 0 0 0 1

Q_A	Q_A^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State Table

Q_A	Q_B	Input x	Q_{A+1}	Q_{B+1}	Flip flop J_A	K_A	Flip flop J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	0	0	0	X	X	1
0	1	1	1	0	1	X	X	1
1	0	0	1	1	X	0	1	X
1	0	1	1	0	X	0	0	X
1	1	0	0	0	0	X	X	1
1	1	1	0	1	0	X	X	0

J_A

$Q_A \backslash Q_B x$	00	01	11	10
0	0	0	1	0
1	x	x	0	0

$J_A = \overline{Q_A} Q_B x$

K_A

$Q_A \backslash Q_B x$	00	01	11	10
0	X	X	X	X
1	0	0	X	X

$K_A = 0$

$Q_A \backslash Q_B x$	00	01	11	10
0	0	1	X	X
1	1	0	X	X

$J_B = \overline{Q_A} x + Q_A \overline{x}$

$J_B = Q_A \oplus x$

$Q_A \backslash Q_B x$	00	01	11	10
0	X	X	1	1
1	X	X	0	1

K_B

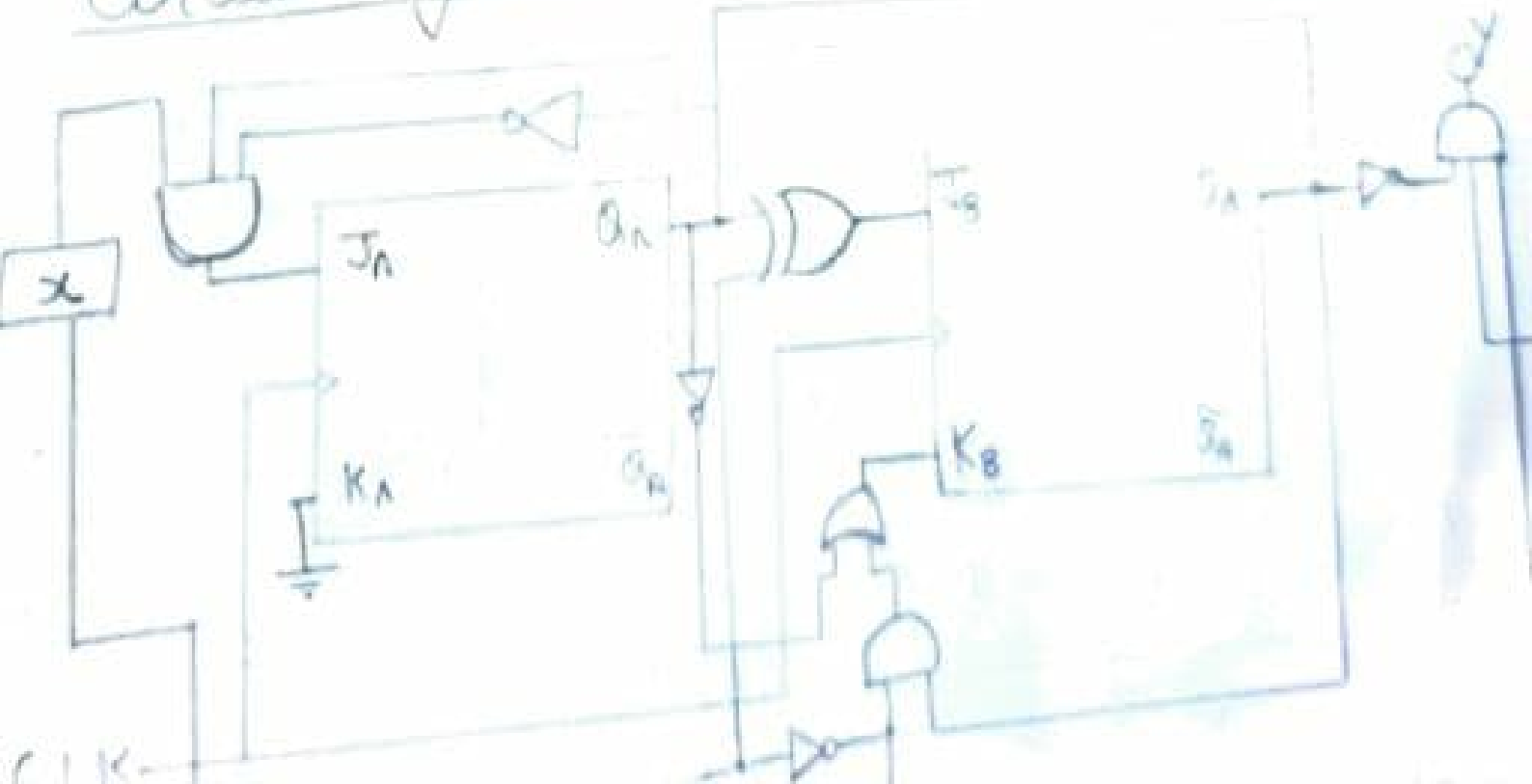
$K_B = \overline{Q_A} + Q_B \overline{x}$

$Q_A \backslash Q_B x$	00	01	11	10
0	0	0	0	0
1	1	0	0	0

Output Y

$Y = Q_A \overline{Q_B}$

Circuit Diagram



EXPERIMENT - 15

ASM

AIM: Implement 2 Bit up | Down counter using MUX controller method when x = 0 then count up and x = 1 then count Down.

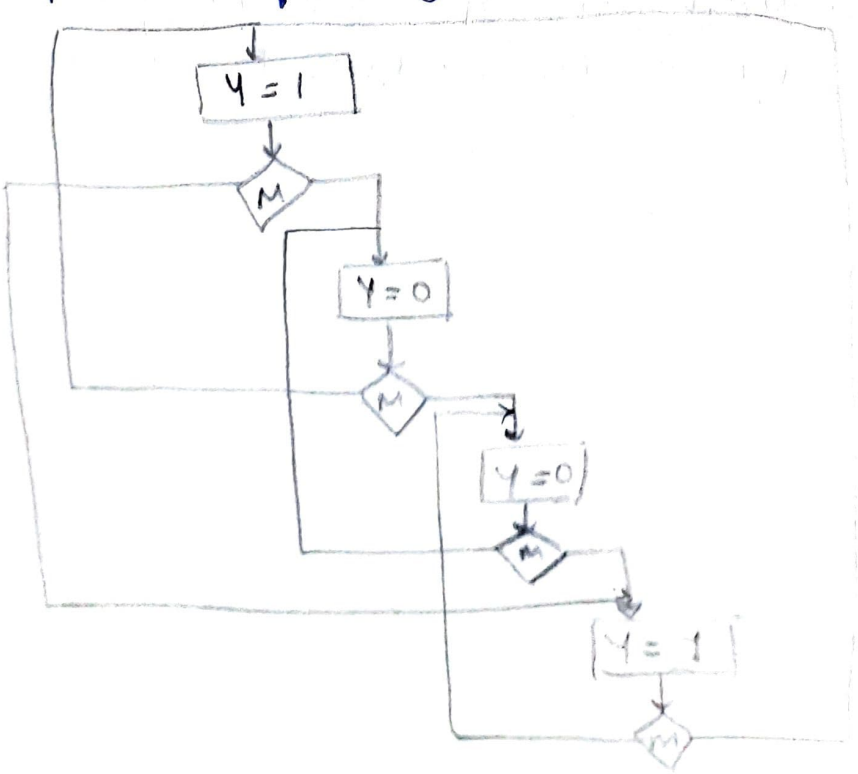
Hardware | Software | IC Required:

4:1 MUX, D-flip-flop, NOT Gate (IC7404)

Theory

ASM Chart:

The algorithmic state machine (ASM) method is a method for describing finite state machine originally developed by thomas osborne & christopher clare at Hewlett-Packard in the 1970s. It is used to represent diagram of digital integrated circuits. The ASM Diagram is like a state Diagram but more structured and thus easier to understand. An ASM chart is a method of describing the sequential operation of a digital system.



ASM Chart
2-Bit up-Down-counter

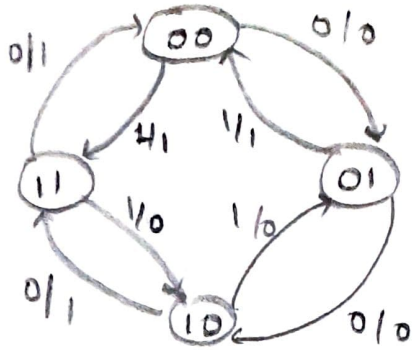
State Diagram

for two Bit up/Down counter

M=0 up counting

M=0 Down counting

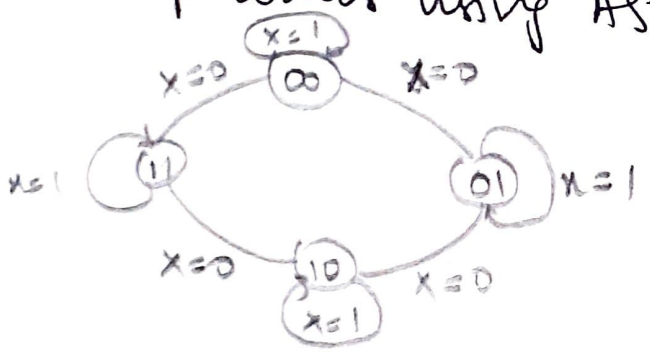
So Basically counting up when M=0 & counting down when M=1



EXCITATION TABLE (For D-flip flop)

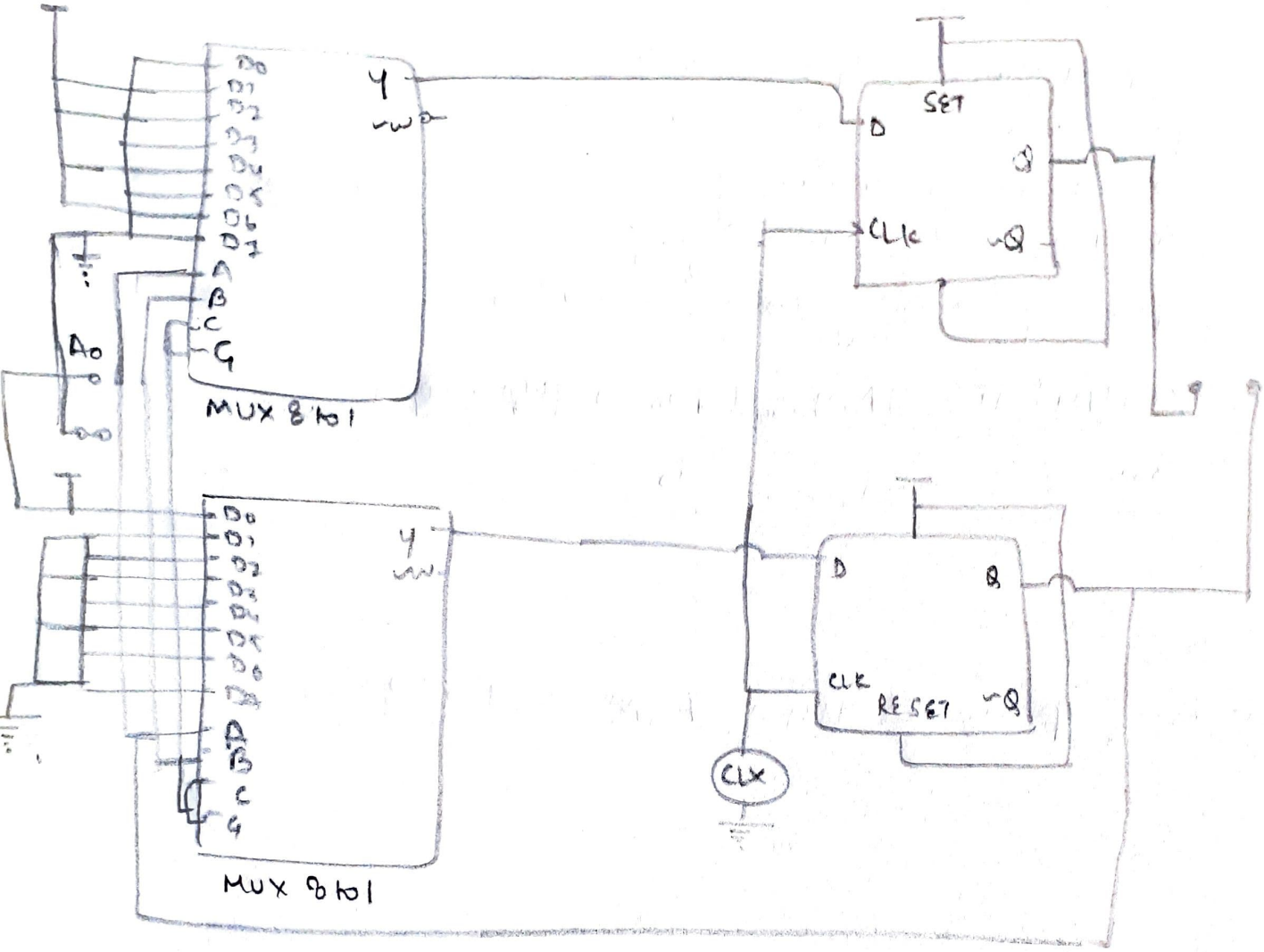
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

2-Bit up counter using ASM with MUX



State Table

Present S_1	Present S_0	Next S_1'	Next S_0'	Input x	MUX1	MUX2
0	0	0	1	0	$D_0=0$	$D_0=\bar{x}$
0	0	0	0	1		
0	1	1	0	0	$D_1=\bar{x}$	$D_1=x$
0	1	0	1	1		
1	0	1	1	0	$D_2=1$	$D_2=\bar{x}$
1	0	1	0	1		
1	1	0	0	0	$D_3=x$	$D_3=x$
1	1	1	1	1		



Conclusion:

Successfully implement of designed ALU with MUX controlled method.