LAB MANUAL# 9

OBJECTIVES:

- To learn and understand how to design a multiple output combinational circuit
- To learn and understand the working of 2-bit binary comparator
- Adder/Subtractor

APPARATUS: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86, 74LS02

THEORY:

Binary comparator is a combinational circuit that compares magnitude of two binary data signals A & B and generates the results of comparison in the form of three output signals A>B, A=B, A<B.

One-bit comparator:

One-bit comparator compares magnitude of two numbers A and B, 4 bit each, and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

$$L = 1 \; if \; A < B$$

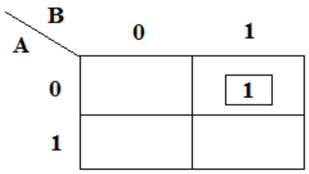
$$E = 1 \; if \; A = B$$

$$G = 1 \; if \; A > B$$

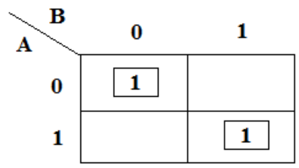
Truth Table:

Inputs		Outputs		
A	**B**	**L**	**E**	**G**
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

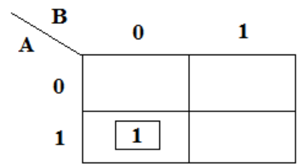
K-Maps for Outputs:



K-Map for Output **L**



K-Map for Output **E**



K-Map for Output **G**

Boolean Expressions of Outputs:

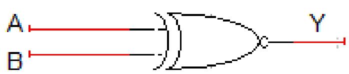
L: $\bar{A}B$

E: $AB + \bar{A}\bar{B}$

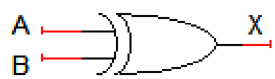
G: $A\bar{B}$

Exclusive-OR & Exclusive-NOR gates:

The figure given below shows the symbol of Exclusive-OR (XOR) and Exclusive-NOR (XNOR) gates.

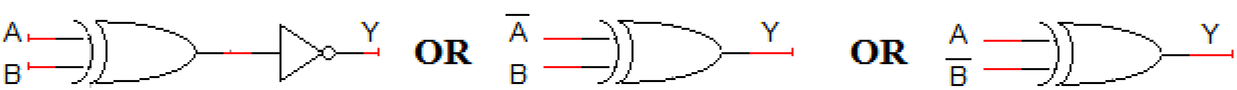


XNOR gate

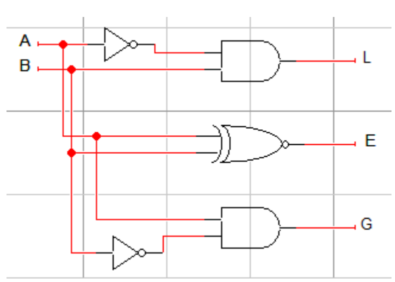


XOR gate

Boolean expression of XNOR gate is $AB + \bar{A}\bar{B}$ and Boolean expression of XOR is $\bar{A}B + A\bar{B}$. Boolean expression of XNOR gate can be implemented using XOR gate as shown in figure below:



Circuit Diagram for one-bit comparator:



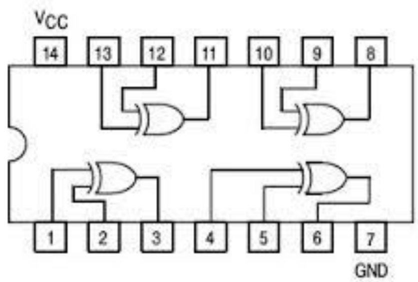
In this experiment 74LS86 IC will be used for implementation of XOR gate function. 74LS86 IC contains four 2-input XOR gates. The function table and connection diagram for this IC are shown below:

Function Table:

Inputs		Output
A	**B**	**Y**
L	L	L
L	H	H
H	L	H
H	H	L

H= Logic High, L= Logic Low

Connection Diagram:



LAB TASK 1:

Design a combinational circuit that compares two 2-bit numbers and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

$$L = 1 \; if \; A < B$$

$$E = 1 \; if \; A = B$$

$$G = 1 \; if \; A > B$$

a) Write truth table

b) Find minimal SOP expressions for the outputs **L**, **E**, and **G** using K-map. Draw separate K-map for each output in the space given below.

c) Fill the following table in order to determine the gate cost for the implementation of binary comparator using SOP expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Adder and subtractor

Half Adder:

Half adder is a logic circuit that performs binary addition of two 1-bit numbers. It generates two outputs namely '**Sum**' and '**Carry**'.

Truth Table:

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Boolean Expressions of Outputs:

$Sum = A \oplus B$

$Carry = AB$

Half Subtractor:

Half subtractor is a logic circuit that performs binary subtraction of two 1-bit numbers. It generates two outputs namely '**Difference**' and '**Borrow**'.

Truth Table:

A	B	Borrow	Difference
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Boolean Expressions of Outputs:

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

Full Adder:

Full adder is a logic circuit that performs binary addition of two 2-bit numbers. It generates two outputs namely '**Sum**' and '**Carry**'.

Truth Table:

A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Boolean Expressions of Outputs:

$$Sum = A \oplus B \oplus C$$
$Carry = AB + BC + AC$ or $Carry = AB + C(A \oplus B)$

Full Subtractor:

Full subtractor is a logic circuit that performs binary subtraction of two 2-bit numbers. It generates two outputs namely '**Difference**' and '**Borrow**'.

Truth Table:

A	B	C	Borrow	Difference
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Boolean Expressions of Outputs:

$Difference = A \oplus B \oplus C$

$Borrow = \bar{A}B + BC + \bar{A}C$ or $Carry = \bar{A}B + C(\bar{A} \oplus B)$

LAB TASK 2:

Implement 4-bit full adder and full subtractor on logic works.

LAB TASK 3:

Implement a 4-bit multiplier using the 4-bit full adder of **LAB TASK 2.**