Logic Design I – Laboratory 04

NAND – NOR - XOR

#	Student ID	Student Name	Grade (10)	Instructor signature
1				
2				
3				

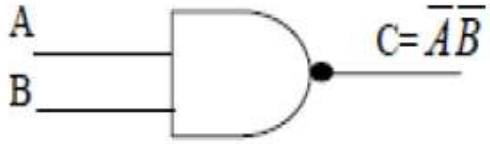
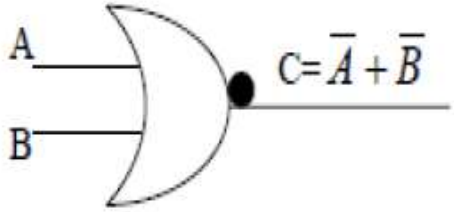
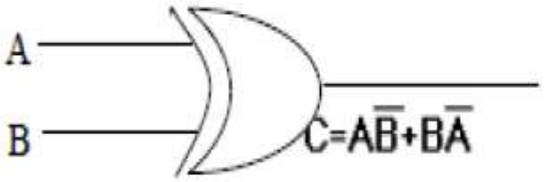
Delivery Date	

Objective

To find the basic NAND & NOR & XOR gates concept and study on multiple inputs and propagation delay.

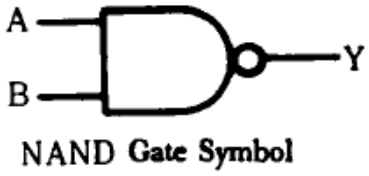
Theory Overview

The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as NAND, NOR, XOR gates, Fig. below shows the circuit symbol, Boolean function, and truth table.

GATE	SYMBOL	INPUTS		OUTPUT
		A	B	C
NAND IC 7400		0	0	1
		0	1	1
		1	0	1
		1	1	0
NOR IC 7402		0	0	1
		0	1	0
		1	0	0
		1	1	0
EX-OR IC 7486		0	0	0
		0	1	1
		1	0	1
		1	1	0

4.1 NAND gate

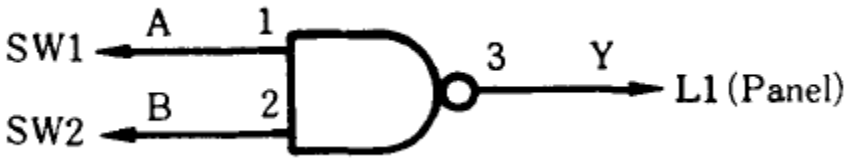
The NAND gate is the same gate as a AND gate and a NOT gate is connected in series, so that the property of a NAND gate is equivalent to that of a AND gate and a NOT gate.



$\overline{A \cdot B} = Y$

Boolean expression

1) Insert a 2-input NAND gate [TTL 7400] into the logic lab breadboard.
2) Construct the circuit shown in Fig



3) Set data switches as shown in following table 4.1. Complete the NAND gate truth table.

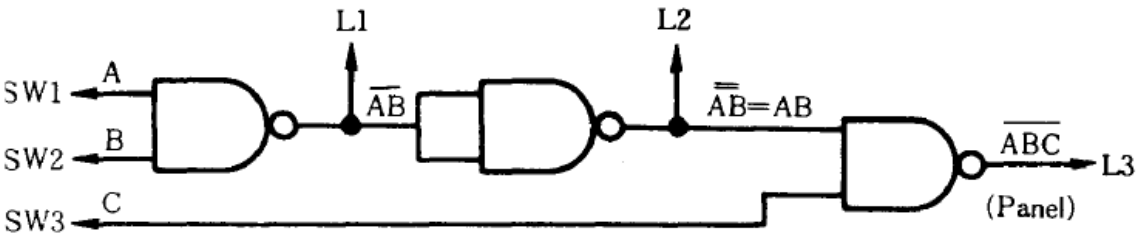
NAND Gate as Inverter

4) Wire as shown in Fig

5) Complete the NAND gate truth table 4.2

Multiple Input NAND Gate:

6) Construct the 3-input NAND gate using 2-input NAND gates.



Symbolically equivalent circuit is:

6 -T (7410)

12-C (4023)

7) Set data switches as shown below. Verify the output states in table 4.3

Results and data analysis

Table 4.1

Input1 (SW1)	Input2 (SW2)	Output (L1)

Table 4.2

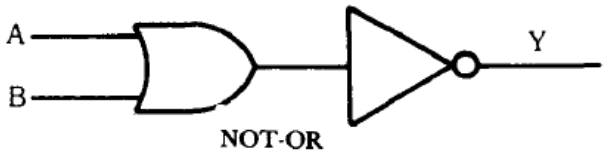
Input1 (SW3)	Output (L2)

Table 4.3

Input			Output		
SW1 A	SW2 B	SW3 C	L1 (AB)'	L2 (AB)''	L3 (ABC)'

4.2 NOR gate

To find a NOR gate characteristics. The NOR gate IS the equivalent of an inverted OR function.



NOT-OR

NOR Gate Symbol

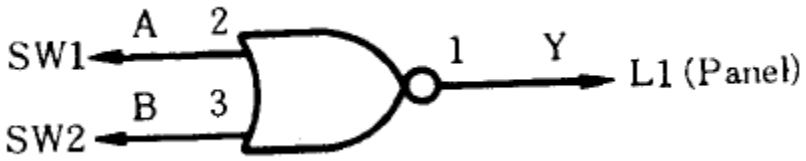
$\overline{A+B}=C$

Boolean expression

2-input NOR Gate

I) Insert a 2-input NOR gate [TTL 7402 #3-T] into the logic lab breadboard.

2) Construct the circuit as shown



3) With data switches as shown in Table 4.4, check the output states.

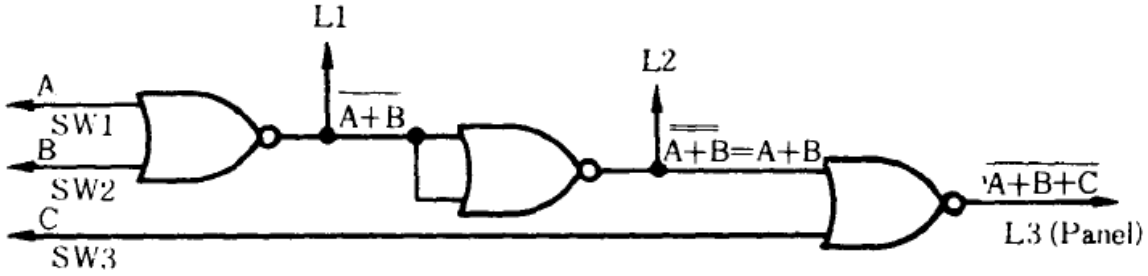
NOR Gate as Inverter

4) The NOR gate can be used as an inverter by connecting all the inputs in one, Wire as shown.

5) Complete the truth table 4.5

Multiple Input NOR gate

6) Construct the 3-input NOR gate using 2-input NOR gates



This circuit can be represented symbolically as

13-T (7427)
13-C (4025)

7) Set data switches as shown in Table 4.6. Record the output states

Results and data analysis

Table 4.4

Input1 (SW1)	Input2 (SW2)	Output (L1)

Table 4.5

Input1 (SW3)	Output (L2)

Table 4.6

Input			Output		
SW1 A	SW2 B	SW3 C	L1 (A+B)'	L2 (A+B)''	L3 (A+B+C)'

4.3 XOR gate

The XOR gate function is to yield a high output state when two inputs are in the opposite state.

XOR Gate Symbol

$$A \oplus B = Y$$

$$(\bar{A} \cdot B) + (A \cdot \bar{B}) = A \oplus B$$

Boolean expression

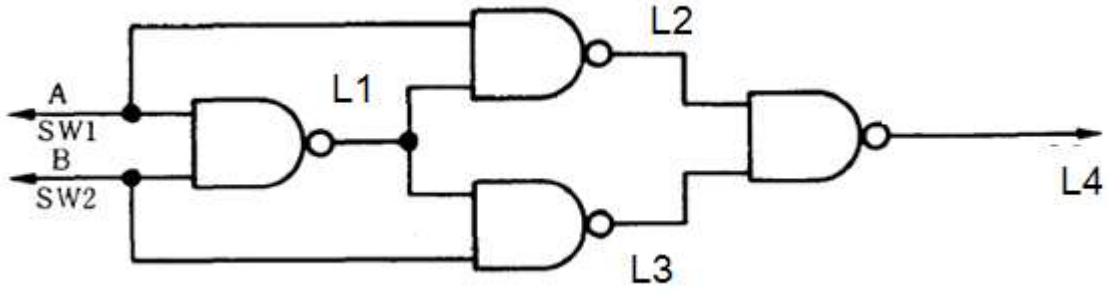
2-input XOR Gate

1) Insert a 2-input XOR gate [TTL 7486] into the logic lab breadboard.
2) Wire as shown in Fig.

3) Complete the following truth table. 4.7

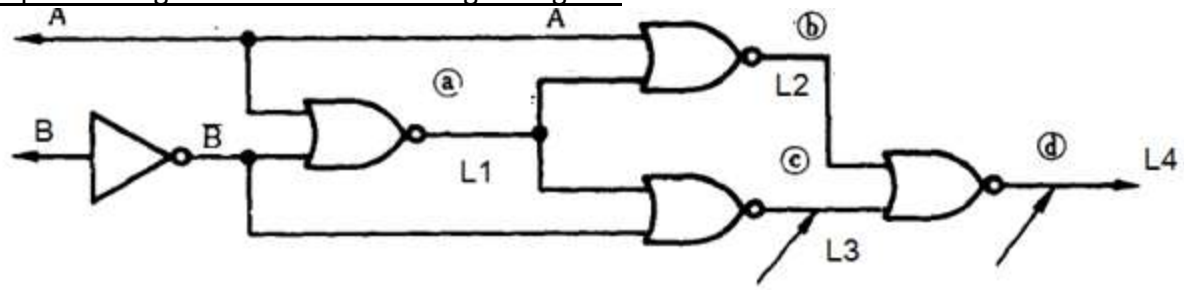
Implementing the XOR Function using NAND gates

1) Insert a Quad 2-input NAND gate [7400] into the logic lab breadboard.
2) Construct the circuit as shown in Fig.



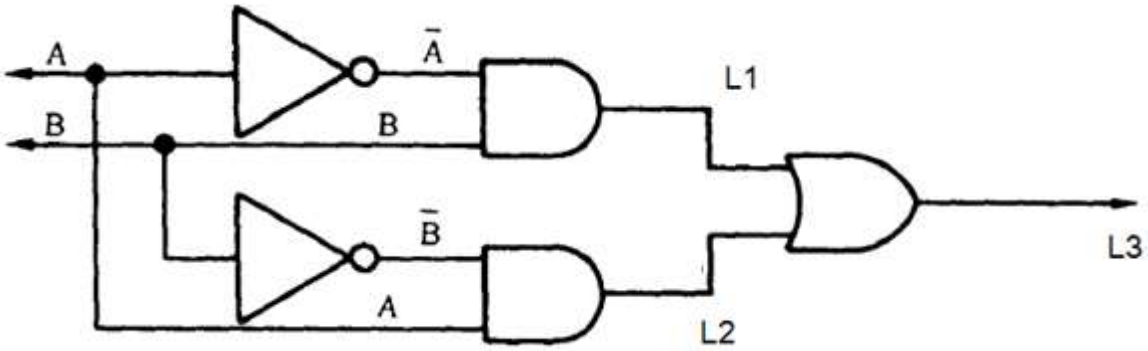
3) Complete the following truth table 4.8.
4) write theL1, L2, L3 Expressions inside the table

Implementing the XOR Function using NOR gates



1) Construct the circuit as shown in Fig.
2) Complete the following truth table 4.9.
3) write theL1, L2, L3 Expressions inside the table

Implementing the XOR Function using AND, OR, NOT gates



1) Construct the circuit as shown in Fig.
2) Complete the following truth table 4.10.
3) write theL1, L2, L3 Expressions inside the table

Results and data analysis

Table 4.7

Input1 (SW1)	Input2 (SW2)	Output (L1)

Table 4.8

Input		Output			
SW1 A	SW2 B	L1 ……	L2 ……	L3 ……	L4 ……

Table 4.9

Input		Output			
SW1 A	SW2 B	L1	L2	L3	L4 .....

Table 4.10

Input		Output			
SW1 A	SW2 B	L1	L2	L3	L4 .....

Questions and Conclusions

1. Simplify the following Boolean expressions to a minimum number of literals:
   (b)* $(x + y)(x + y')$

2. Draw logic diagrams of the circuits that implement the original and simplified expressions in
(b)* $(x + y)(x + y')$