**Basic AND, OR & NOT LOGIC GATES**

***OBJECTIVE:***

* To investigate the behavior of the OR gate using IC & Transistor.
* To investigate the behavior of the AND gate using IC & Transistor.
* To investigate the behavior of the NOT gate using IC & Transistor.

***THEORY:***

In general logic circuits have one or more inputs and only one output. The circuits respond to various input combinations and a truth table shows this relationship between circuits input combinations and its output. The truth table for a particular circuit explains how the circuit behaves under normal conditions. Familiarization with a logic circuit’s truth table is essential to the technologist or technician before he or she can design with or troubleshoot the circuit.

In this experiment three logic circuits are covered: the OR, AND & NOT gates.

The OR operation can be summarized as follows:

1. When any input is 1, the output is also 1.
2. When all inputs are 0, the output is also 0.

The AND operation can be summarized similarly:

1. When any input is 0, the output is also 0.
2. When any input is 1, the output is also 1.

Finally, the NOT operation is said to be complementary.

1. If the input is 0, the output is 1.
2. If the input is 1, the output is 0.

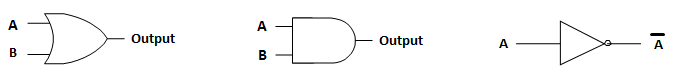
You should recall that the logic levels 0 and 1, have voltage assignments. For TTL circuits a logic 0 can be anywhere from 0 V to +0.8 V, and a logic 1 is in the range of +2.0 V to +5.0.

***EQUIPMENT / REQUIREMENT:***

* 7411 IC AND gate
* 7432 IC OR gate
* 7404 IC NOT gate
* 2 LEDs or Logic Probe
* 6 NPN Transistor
* Breadboard
* 0-5 Volt DC Power Supply

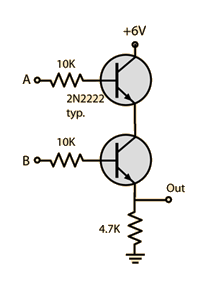
***PROCEDURE:***

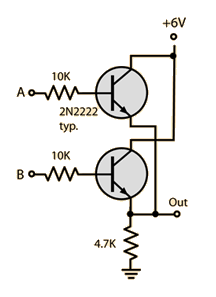
*Figure 1.1* shows logic symbols of AND, OR and NOT gates. *Figure 1.2* shows the layouts of AND gate IC (7411), OR gate IC (7432) and NOT gate IC (7404). The pin configuration is also given in the layouts. Construct the circuits with the help of these layouts. Pin no 7 and pin no 14 of each IC is ground and VCC respectively. Apply different inputs and observe the outputs and then compute the truth *tables 1.1, 1.2 and 1.3*.

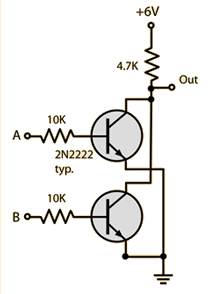
****

***(a) Symbol of OR gate (b) Symbol of AND gate (c) Symbol of NOT gate***

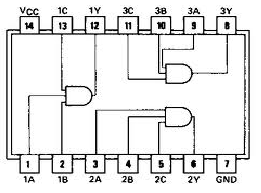
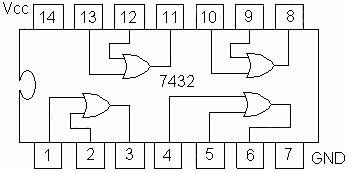
***Fig 1.1***

*****Internal Circuits of AND, OR, NOT Gate using Transistor:***

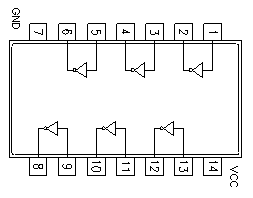




***(a) AND gate (b) NOT gate (c) OR gate***

****

***(a) 7432 quad 2-input OR gate IC (b) 7411 3 input AND gate***

****

***(c) 7404 (Hex Inverter) NOT gate IC***

***Figure 1.2:***

***OBSERVATION TABLE:***

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q=A+B** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **1** |

***Table 1.1***

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q=A\*B** |
| **0** | **0** | **0** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

***Table 1.2***

|  |  |
| --- | --- |
| **A** | **A’** |
| **0** | **1** |
| **1** | **0** |

***Table 1.3***

***QUESTIONS / RESULTS:***

1. The output of an OR gate is **LOW** only when all the input is LOW.
2. The output of an AND gate is LOW whenever any input is LOW.
3. If an OR gate input were accidentally shorted to VCC, the output of the gate would always be **HIGH** no matter what level the other input level might be.
4. A High Voltage (5volt) stood for a high output.

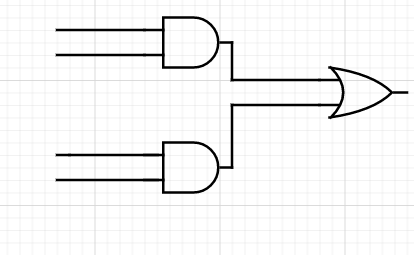
Addition Task to be performed:

1. (AB+CD)
2. (A+B.C+D)

**Task 1:**  AB+CD

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **AB** | **CD** | **AB+CD** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Circuit Diagram:**



A

B

(AB+CD)

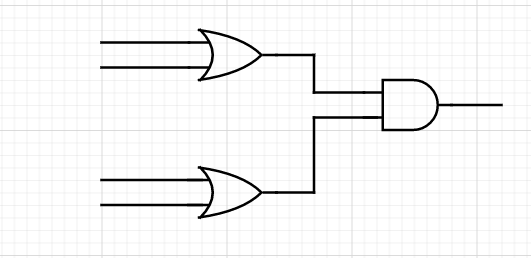
C

D

**Task 2:** A+B.C+D

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **A+B** | **C+D** | **A+B.C+D** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Circuit Diagram:**



A B

(A+B.C+D)

C

D

***CONCLUSION:***

**Logic gates** are the basic building blocks of any digital system. It is an electronic **circuit** having one or more than one input and only one output.

Following are the logic circuits commonly used,

1) **AND**(A HIGH output (1) results only if all the inputs to the AND **gate** are HIGH (1)).

2) **OR**(It gives a high output (1) if one or more of its inputs are high).

3) **NOT**(It works as inverter and invert the given value to 0 or 1).

**NAND & NOR Gate**

***OBJECTIVE:***

* To investigate the behavior of the NOR gate using IC & Transistor.
* To investigate the behavior of the NAND gate using IC & Transistor.

***THEORY:***

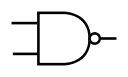
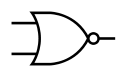
In Experiment 1, you learned the characteristics of three of the fundamental logic gates: the AND, OR, and NOT. You will now be introduced to two of the remaining logic gates: the NAND and NOR. The NAND and NOR gates are nothing more than inverted AND and OR gates, respectively. That is important, but not the most important thing. The fact that a NAND or a NOR can be used to create all other gates is important, because this fact has made them more popular in use than the others.

***EQUIPMENT / REQUIREMENT:***

* 7400 IC
* 7402 IC
* 4 NPN Transistor
* 4 LED or Logic probe
* 0-5 volt DC power supply

***PROCEDURE:***

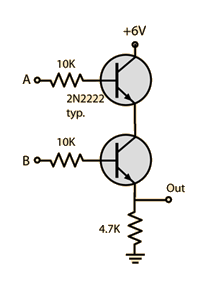
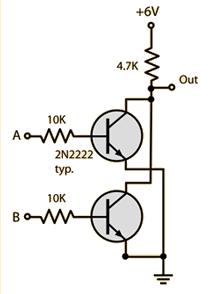
*Figure 2-1* shows logic symbols of NAND & NOR *Figure 2-2* shows the layouts of NAND gate IC (7400) & NOR gate IC (7402). The pin configuration is also given in the layouts. Construct the circuit with the help of these layouts. Pin no. 7 and Pin no. 14 of each IC is Ground and VCC respectively. Apply different inputs on the given input pins and observe the out puts, and then complete the truth *tables 2-1* and *2-2* of these gates.

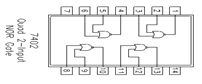
1. ***NAND gate Symbol (b) NOR gate Symbol***

***Fig2-1***

***Internal Circuits of NAND, NOR Gate using Transistor***

**

***(a) NAND gate (b) NOR gate***

***(a) IC configuration NAND 7400 gate (b) IC Configuration NOR 7402 gate***

***Fig 2-2***

***OBSERVATION TABLE:***

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q=(A+B)’** |
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **0** |

***Truth Table 2-1 for NOR gate***

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q=(A\*B)’** |
| **0** | **0** | **1** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

***Truth Table 2-2 for NAND gate***

***QUESTIONS / RESULTS:***

# Why are NAND and NOR gates called universal gates?

**A**. NAND and NOR are called universal gates because all the other gates like

AND,OR,NOT,XOR and NOR can be derived from it.

1. If the 0 and 1 were inputs for a NAND gate, what would be the output?

**A**. The output will be high.

1. If a signal passing through a gate is inhibited by sending a LOW into one of the inputs, and the output is HIGH, the gate is an?

**A**. The gate will be NAND gate.

1. When used with an IC, what does the term "QUAD" indicates?

**A. Quad means** that the **IC has 4** circuits inside it which performs the same function.

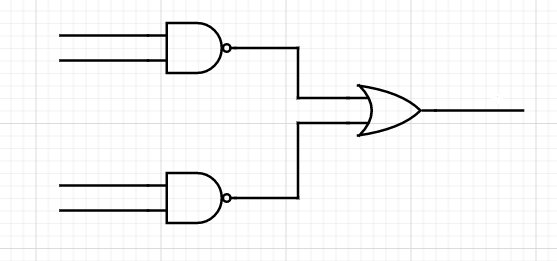
1. A High Voltage (5volt) stood for a **TLL Gate**.

Addition Task to be performed:

1. Implement OR gate using NAND gate

**A.** OR gate can simply be implemented by using two NAND gates.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **AB** | **CD** | **A'B'** | **C'D'** | **(A'B')+( C'D')** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |



**A**

**B**

**C**

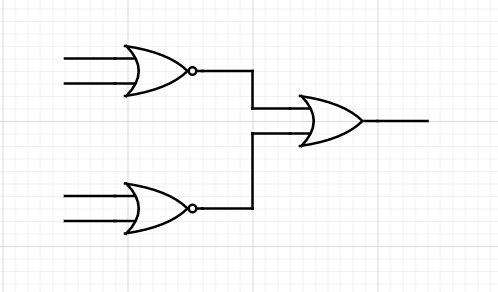
**D**

1. Implement AND, OR gate using NOR gate

**A(1). OR gate** can simply be implemented by using two NOR gates.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **A+B** | **C+D** | **(A+B)'** | **(C+D)'** | **(A+B )' + (C+D) '** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |

**(A+B )' + (C+D) '**



A

B

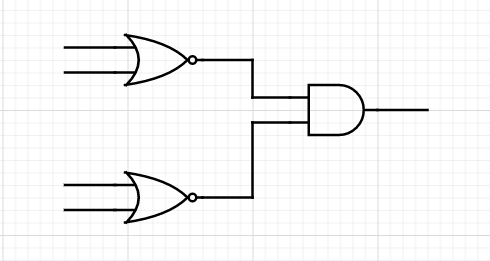
C

D

**A(2). AND gate** can simply be implemented by using two NOR gates.

**(A+B )' . (C+D) '**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **A+B** | **C+D** | **(A+B)'** | **(C+D)'** | **(A+B )' . (C+D) '** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |



***CONCLUSION:***

NAND and NOR are two universal gates, they are called universal gates because other logical gates can be derived from it.

**NAND**(The output of a NAND gate is **true** when one or more, but not all, of its inputs are **false**. If all of a NAND gate's inputs are **true**, then the output of the NAND gate is **false**)

**NOR**(Its output is "true" if both inputs are "false." Otherwise, the output is "false.")

**To Study Exclusive-OR & Exclusive-NOR gate**

***OBJECTIVE:***

* To investigate the behavior of the EX-OR
* To investigate the behavior of the EX-NOR

***THEORY:***

In Experiment 1&2, you learned the characteristics fundamental logic gate. You will now be introduced to two of the remaining logic gates, the EX-OR and EX-NOR. An exclusive OR (XOR) gate is gate with two or more three inputs and one output an output of two inputs XOR assumes a high state if one and only one input assume a high state. This is equivalent to saying that the output is a High if either input X or Y is high exclusively, and low when both are 1 and 0 simultaneously.

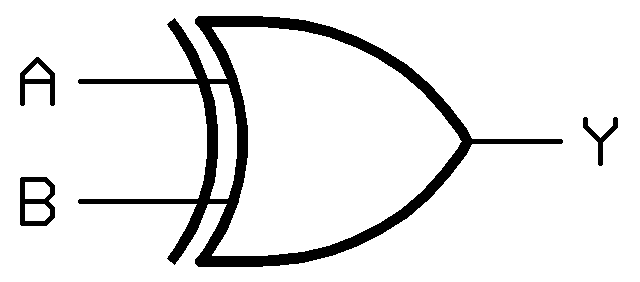
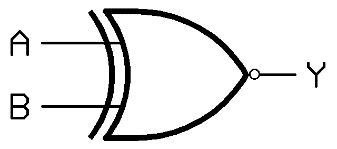
The **Exclusive-NOR Gate** function or Ex-NOR for short, is a digital logic gate that is the reverse or complementary form of the Exclusive-OR function. the output of an Exclusive-NOR gate **ONLY** goes “HIGH” when its two input terminals, A and B are at the “**SAME**” logic level which can be either at a logic level “1″ or at a logic level “0″.

***EQUIPMENT / REQUIREMENT:***

* 7486 IC XOR
* 74266 IC XNOR
* 2 LED or Logic probe
* 0-5 volt DC power supply

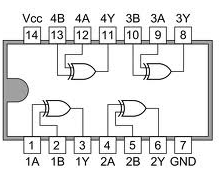
***PROCEDURE:***

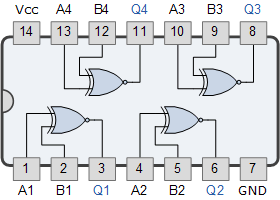
*Figure 2-1* shows logic symbols of XOR & XNOR *Figure 2-2* shows the layouts of XOR gate IC (7486) & XNOR gate IC (74266).The pin configuration is also given in the layouts. Construct the circuit with the help of these layouts. Pin no. 7 and Pin no. 14 of each IC is Ground and VCC respectively, Apply different inputs on the given input pins and observe the out puts, and then complete the truth *tables 3-1* and *3-2* of these gates.

1. ***XOR gate Symbol (b) XNOR gate Symbol***

***Fig3-1***





1. ***IC configuration 7486 (b) IC Configuration 74266***

***Fig3-2***

***OBSERVATION TABLE:***

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

***3-1 Truth Table for XOR gate***

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

***3-2 Truth Table for XNOR gate***

***QUESTIONS / RESULTS***

1. If the 0 and 1 were inputs for a XOR gate, what would be the output?

**A.** The output will be low.

1. If a signal passing through a gate is inhibited by sending a LOW into one of the inputs, and the output is LOW, the gate is an?

**A.** Here the gate is .

1. XNOR provides the inverted output of an XOR.

**A.** Yes

***CONCLUSION:***

**EX-OR(**The Exclusive OR gate high output when the inputs are not at equal logic level. The Exclusive OR operation is widely used in digital circuit. It is also called as XOR**)**

**EX-NOR(**The Exclusive NOR Gate gives 1 at output, when the inputs are identical(both 1’s or both 0’s). It performs the same function as XOR gate followed by a NOT gate. It is also called as XNOR Gate**).**

**BOOLEAN ALGEBRA**

***OBJECTIVE:***

* To verify experimentally some of the Boolean theorems.

***THEORY:***

**Boolean algebra** is the mathematics we use to analyze digital gates and circuits. We can use these “Laws of Boolean” to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required. Boolean algebra is therefore a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions. The combinational logic circuits do not have the ability to memorize their past. The result is that combinational logic circuits have no feedback and any changes to the signals being applied to their inputs will immediately have an effect at the output.

***PROCEDURE:***

**Boolean Theorems:** Following Boolean theorems table mount a *7404 IC*, a 7408 IC, and a 7432 IC on the circuit board. Connect *VCC* to *+5 V* and GND to power ground on each IC. To verify each theorem, connect the circuit for that theorem. Monitor the output with logic probe and also with LED.

***EQUIPMENT / REQUIREMENT:***

* 7404IC
* 7408 IC
* 7432 IC
* LED
* 0-5 VOLT DC Power Supply.

***BOOLEAN THEORAM:***

|  |  |
| --- | --- |
| ***Idem potency*** | X + X = X          X • X = X |
| ***Redundancy Law*** | *X+X.Y* |
| ***Double negation*** | = X |
| ***Commutative Law*** | X + Y = Y+X |
| ***Associative laws*** | X + (Y + Z) = (X + Y) + Z |
| ***Distributive laws*** | X (Y + Z) = XY + XZ |
| ***Absorption*** | X + XY = X |
| ***De Morgan's theorem*** | (X1 + X2 + X3.....) = X1 • X2 • X3.......  (X1 • X2 • X3.....) = X1 + X2 + X3....... |

***RULES OF BOOLEAN ALGEBRA***

1. X+0=X 7. X.X=X
2. X+1=1 8. X.X= 0
3. X.0=0 9. X.X=X
4. X.1=1 10. X+XB=A
5. X+X=X 11. X+XB=X+B
6. X+X= X 12. (X+B)(X+C)=X+BC

Verify circuit by using IC’s on breadboard.

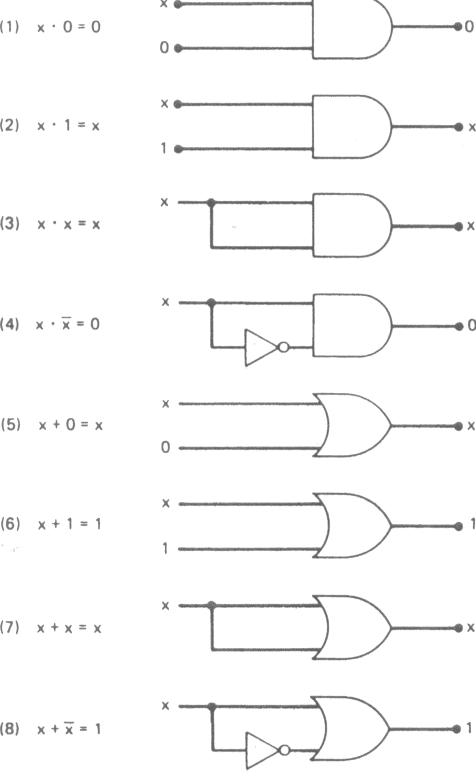
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **X** | **X+0** | **X+1=1** | **X.0=0** | **X.1=1** | **X+X=X** | **X+X'= X** | **X.X= X** | **X.X'= 0** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **0** |

***Table 4.1***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **x** | **y** | **Z** | **X + (Y + Z)** | **X + XY** | *X+X.Y* | **x (x + y)** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** |

***Table 4.2***

***CIRCUIT IMPLEMENTATION:***

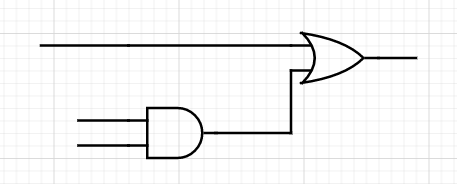


Task#1: Construct Circuit for Redundancy Law

**Answer:**

Equation for redundancy law is,

*X+X.Y*



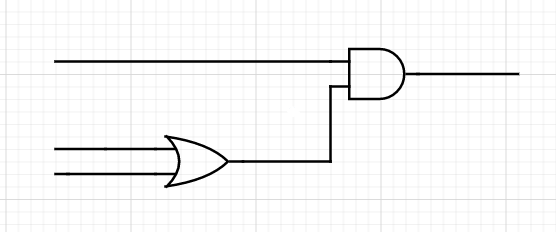
X (X+X.Y)

X (X.Y)

Y

Task#2: Draw logic diagram for x (x + y) and also find out the output x (x + y) =?

**Answer:**

Given,

x.(x+y)

x x.(x+y)

x (x+y)

y

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **(X+Y)** | **X(X+Y)** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** |

Task#3: Construct the circuit for Associative & Absorption theorems and observe the Output.

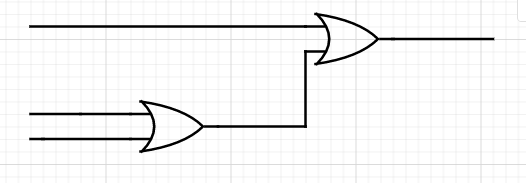
**Answer:**

Equation for Associative theorems is,

X + (Y + Z) = (X + Y) + Z

**FOR LHS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **Y+Z** | **X+(Y+Z) (L.H.S)** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** |

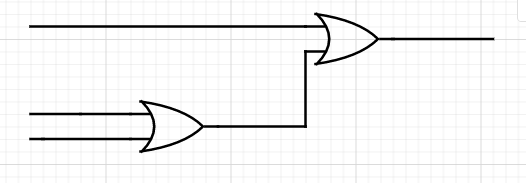
 **X X + (Y+Z)**

**Y (Y+Z)**

**Z**

**FOR RHS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **X+Y** | **(X+Y)+Z (R.H.S)** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** |

 **Z (X+Y) +Z**

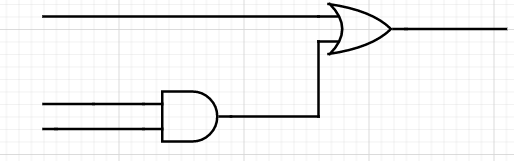
**X (X+Y)**

**Y**

Equation for absorption theorem is,

X + XY = X

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **XY** | **X+XY** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** |



X X + XY

X X.Y

Y

***CONCLUSION:***

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**.

Following are the important rules used in Boolean algebra.

* Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
* Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as B Bar. Thus if B = 0 then B Bar = 1 and B = 1 then B Bar = 0.
* OR-ing of the variables is represented by a plus (+) sign between them. For example, OR-ing of A, B, C is represented as A + B + C.
* Logical AND-ing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

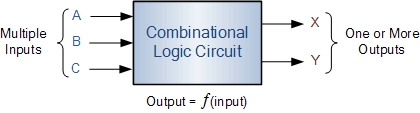
**COMBINATIONAL CIRCUITS**

***OBJECTIVE:***

* Implementation of Combinational Circuits.

***PROCEDURE*:**

**Combination Circuits:** The combinational logic circuits are a type of logic circuits containing only logic gates (AND, OR, XOR, NOT, NAND, NOR) and its output only depends on the current input (do not have memory).



The three main ways of specifying the function of a combinational logic circuit are:

1. **Boolean algebra:**This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that result in a logic “1″ output.
2. **Truth Table:**A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
3. **Logic Diagram:** This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol that implements the logic circuit.

***EQUIPMENT / REQUIREMENT:***

* 7404IC.
* 7408 IC.
* 7432 IC.
* 7400 & 7402 IC
* LED
* 0-5 VOLT DC Power Supply.

***COMBINATIONAL LOGIC GATE SCHEMATIC***

The circuit in the diagram below uses two-input AND gates to provide a three-input AND gate.

A

B

C

This is one method of producing a three-input AND gate

A

B

C

A truth table drawn for both circuits will show the logic function clearly:

|  |  |  |  |
| --- | --- | --- | --- |
| C | B | A | O/P |
| 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 | 0 |
| 1 | 1 | 1 | 1 |

***Table 5.1***

***Task#1:*** Derive the truth table and Boolean expression for the output of the following logic circuits.

P

Q

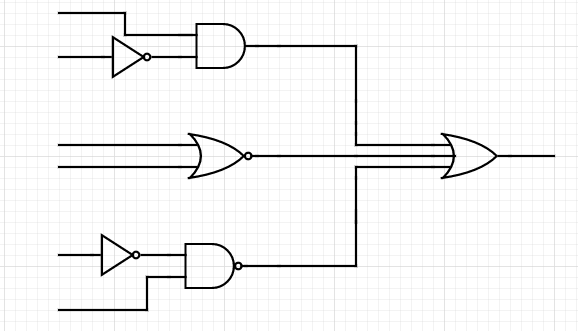
R

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R | Q | P | \_\_\_  P.Q | \_  R | \_\_\_\_\_\_\_  \_\_\_ \_  P.Q + R |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

***Task#2****:* Implement circuit for the following equation

*Y=AB+C+D+DE*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **B’** | **C** | **D** | **D’** | **E** | ***AB’*** | **(C or D)’** | **(D’ and E)’** | **Output** |
| **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** |
| **0** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** |

 A AB’

B

C (C or D)’ AB’ + (C or D)’ + (D’ and E)’

D

D

E (D’ and E)’

***CONCLUSION:***

A [Combinational Circuit](https://www.geeksforgeeks.org/difference-between-combinational-and-sequential-circuit/) consist of logic gates whose outputs at any instant of time are determined directly from the present combination of inputs without regard to previous input.

**HALF ADDER**

***OBJECTIVE:***

* To observe the working of half adder

***THEORY:***

### **Half adder:** *A half adder is a logical circuit that performs an addition operation on two binary digits. The half adder produces a sum and a carry value which are both binary digits. The drawback of this circuit is that in case of a multi bit addition, it cannot cater to carry*.

***EQUATION FOR HALF ADDER:***

Description: S = A \oplus B

Description: C = A \cdot B

***EQUIPMENT / REQUIREMENT:***

* IC 7486
* 7408 IC.
* 7432 IC.
* Breadboard
* LED
* 0-5 VOLT DC Power Supply.

***PROCEDURE:***

Construct the combinational circuit as diagram given *figure 6.1* after constructing both of these circuits, observe the output and complete the truth table.

[](http://en.wikipedia.org/wiki/Image:Half-adder.svg)

***Figure 7.1 half adder circuit***

***OBSERVATION TABLE:***

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **S** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** |

***Table 7.1***

***CONCLUSION:***

**Half adder** is a combinational arithmetic **circuit** that adds two numbers and produces a sum bit (S) and carry bit (C) as the output.

It consists of one **XOR** and one **AND** gate.

**FULL ADDER**

***OBJECTIVE:***

* To observe the working of full adder

***THEORY***:

*Full adder:* A full adder is a logical circuit that performs an addition operation on three binary digits. The full adder produces a ‘sum’ and ‘carry’ value, which are both binary digits. It can be combined with other full adders or work on its own.

***EQUATION FOR FULL ADDER:***

Description: S = (A \oplus B) \oplus C_i

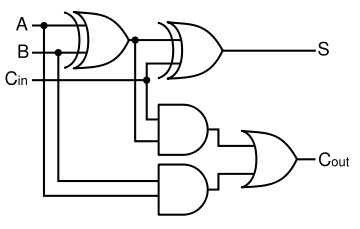
Description: C_o = (A \cdot B) + (C_i \cdot (A \oplus B)) = (A \cdot B) + (B \cdot C_i) + (C_i \cdot A)

***EQUIPMENT / REQUIREMENT:***

* IC 7486
* 7408 IC.
* 7432 IC.
* Breadboard
* LED
* 0-5 VOLT DC Power Supply.

***PROCEDURE:***

Construct the combinational circuit as diagram given figure 7.1. After constructing both of these circuits, observe the output and complete the truth table.

[](http://en.wikipedia.org/wiki/Image:Full-adder.svg)

***Figure 7.1 Full adder circuit***

***OBSERVATION TABLE:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cī | Co | S |
| 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 |

***Table 7.1***

***QUESTIONS / RESULTS:***

1. A full adder can be constructed from **two** half adders by connecting *A* and *B* to the input of **one** half adder
2. Full adder performs the arithmetic addition of **three** input bits.
3. The **binary** adder is one that does not take a carry-in from another adder.

***CONCLUSION:***

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

**DECODER& ENCODER**

***OBJECTIVE:***

* To observe the working of decoder
* To observe the working of Encoder

***THEORY:***

**Decoder**

The process of taking some type of code and determining what it represents in terms of a

Recognizable number or character is called decoding. A decoder is a combinational logic circuit that performs the decoding function, and produces an output that indicates the meaning of the input code.   
**Encoder**

An encoder is a combinational logic circuit that generate n output lines from 2n (or less) inputs. It has the reverse function of the decoder.

***EQUIPMENT / REQUIREMENT:***

* ICs – 7408, 7432 and 7404
* Breadboard
* LED
* DC Power Supply.

***PROCEDURE:***

1. Construct a circuit as shown in *Fig. (1),* set data switches as shown in the two to four line

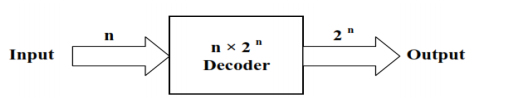
Decoder output table. Record the output indications of L1 to L4.

2. Construct the circuit as shown in *Fig. (3),* set data switches as shown in the four to two line encoder truth table. Record the output indications of L1 & L2.

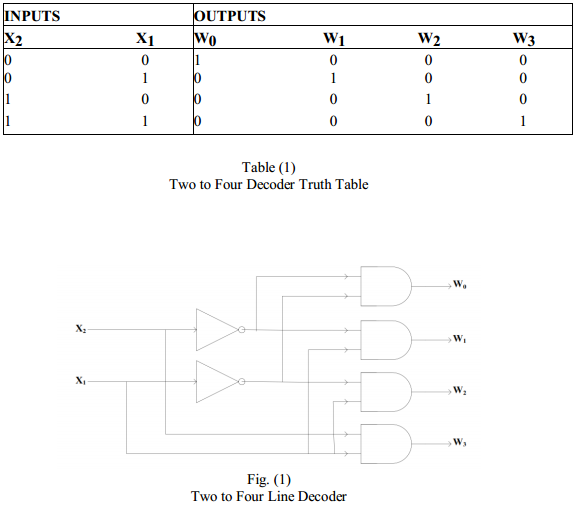
**Decoder**

The decoder is an important part of the system which selects the cells to be read from and write into. This particular circuit is called a decoder matrix, or simply a decoder, and has a

Characteristic that for each of the possible 2n binary input number which can be taken by the n input cells, the matrix will have a unique one of its 2n output lines selected.

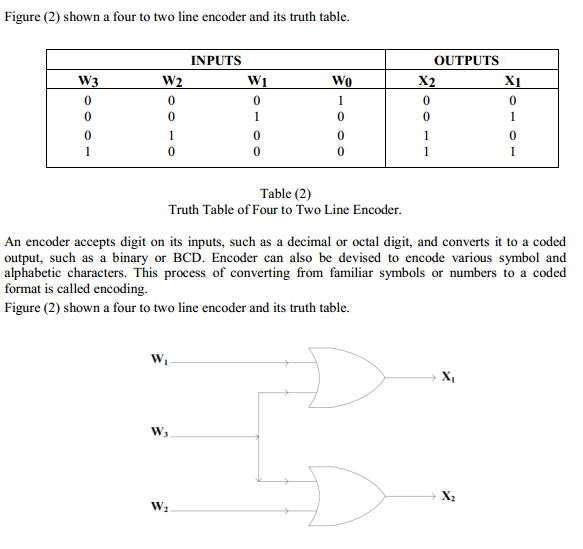


The decoder is called *n* to m where *m < 2n* for example two to four line decoder, *Fig. (1)* Shows a two to four line decoder and its truth table.



**Encoder**

An encoder accepts digit on its inputs, such as a decimal or octal digit, and converts it to a coded output, such as a binary or BCD. Encoder can also be devised to encode various symbol and alphabetic characters. This process of converting from familiar symbols or numbers to a coded format is called encoding.



***CONCLUSION:***

The **decoder** is an electronic device that is used to convert digital signal to an analogue signal. It allows single input line and produces multiple output lines.

It is a combinational circuit that converts n lines of input into 2n lines of output.

An **encoder** is an electronic device used to convert an analogue signal to a digital signal such as a BCD code. It has a number of input lines, but only one of the inputs is activated at a given time and produces an N-bit output code that depends on the activated input