**LAB #11**

Boolean algebra and Logic Simplification

### OBJECTIVE:

###### Introduction to Boolean algebra and Familiarization of Boolean Laws.

1. Simplification of Boolean Logic Equation.

**EQUIPMENT:**

15. IC: 7400LS, 7404LS, 7408LS and 7432LS.

1. Bread board.
2. Connection Wires.
3. Digital Logic Probe.
4. DC supply (0 and +5V).

**THEORY:**

Boolean Algebra is the mathematics which 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. On comparison with the Elementary Algebra where the main operations are Addition and Multiplication, the main operations of Boolean Algebra are AND, OR and NOT. When the number of gates are increased in the circuit, it is difficult to manage the connections, this becomes a challenge. Boolean Algebra provides certain rules to simplify the logic equation in order to reduce the number of gates to the least possible by keeping the circuit overall same functionality. Boolean Algebra was introduced by George Boole in his first book *“The Mathematical Analysis of Logic (1847)”*. The laws of Boolean Algebra are as shown in Table 3.1:

|  |  |  |  |
| --- | --- | --- | --- |
| **Boolean Expression** | **Description** | **Equivalent Switching Circuit** | **Boolean Algebra Law or Rule** |
| *A + 1 = 1* | A in parallel with closed = "CLOSED" | universal parallel circuit | Annulment |
| *A + 0 = A* | A in parallel with open = "A" | universal parallel | Identity |
| *A. 1 = A* | A in series with closed = "A" | universal series circuit | Identity |

**69**

|  |  |  |  |
| --- | --- | --- | --- |
| *A. 0 = 0* | A in series with open = "OPEN" | universal series | Annulment |
| *A + A = A* | A in parallel with A = "A" | idempotent parallel circuit | Idempotent |
| *A. A = A* | A in series with A = "A" | idempotent series circuit | Idempotent |
| *NOT* 𝐴̅ *= A* | NOT A  (double negative) = "A" |  | Double Negation |
| 𝐴̅ + 𝐴 = 1 | A in parallel with NOT A = "CLOSED" | complement parallel circuit | Complement |
| 𝐴. 𝐴̅ = 0 | A in series with NOT A = "OPEN" | complement series circuit | Complement |
| *A+B = B+A* | A in parallel with B = B in parallel with A | absorption parallel circuit | Commutative |
| *A.B = B.A* | A in series with B = B in series with A | absorption series circuit | Commutative |
| ̅𝐴̅̅+̅̅̅̅𝐵̅ = 𝐴̅. 𝐵̅ | invert and replace OR with AND |  | de Morgan’s Theorem |
| ̅𝐴̅̅.̅𝐵̅ = 𝐴̅ + 𝐵̅ | invert and replace AND with OR |  | de Morgan’s Theorem |

TABLE 3.1: Boolean Laws

**70**

**Associative property of addition:**

*A + (B + C) = (A + B) + C*

**Associative property of Multiplication:**

*A ● (B ● C) = (A●B) ●C*

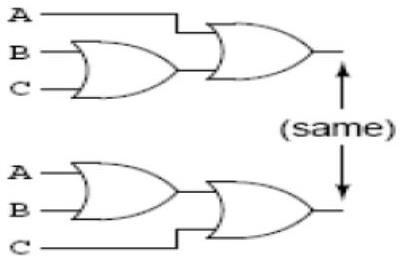


Figure 3.1 Circuit of Associate Property w.r.t. “Addition”

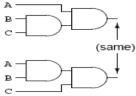
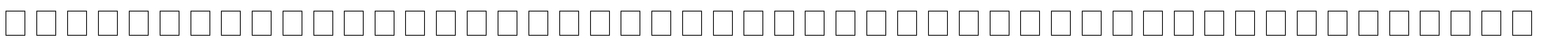
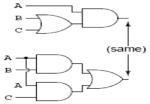


Figure. 3.2: Circuit of Associate Property w.r.t. “Multiplication”



**Distributive property:**

*A ● (B + C) = (A●B) + (A●C)*

Figure. 3.3: Circuit depicting Distributive Proper

##### 71

**Logic Simplification:**

1. Example # 01:

*F = (A+ B). (A+ C)*

*= A.A + A .C+ A .B + B.C*

*= A+ A.C + A.B + B.C*

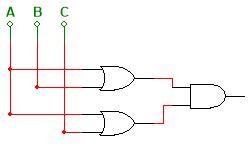
*= A (1 + C) + A.B + B.C*

*= A.1 + A.B + B.C*

*= A +A.B+ B.C*

*= A (1 + B) + B.C*

*= A.1 + B.C*

*= A + B.C*

Distributive Law Idempotent AND Law Distributive Law Identity OR Law Identity AND Law Distributive Law Identity OR Law Identity AND Law

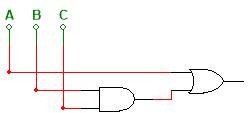
*Is Equal to*

Fig. 3.4: Circuit Diagrams depicting Same Logic after Boolean Simplification – Example 01

1. Example # 02:

*F = A.B + B.C. (B + C)*

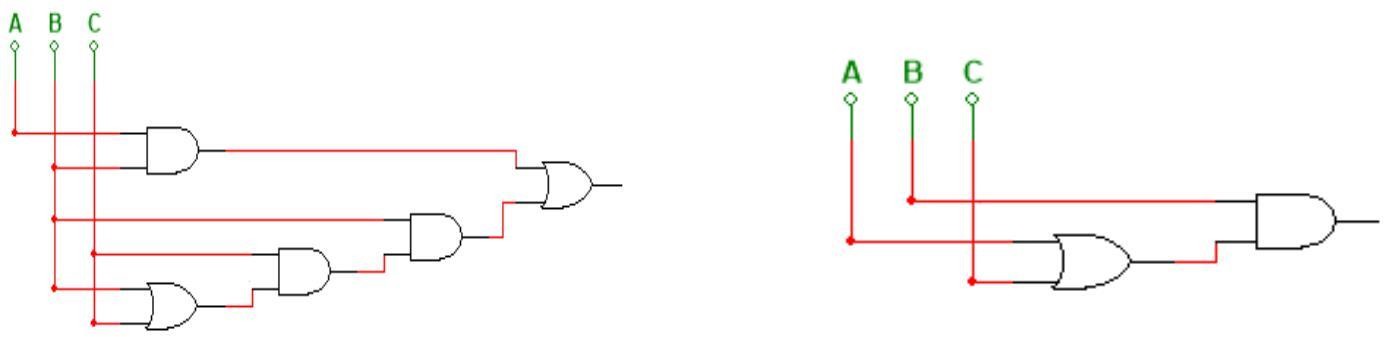
= *A.B + B .C .B + B.C.C* Distributive Law

= *A.B + B .B .C + B .C .C* Commutative Law

*= A.B +B.C + B.C* Idempotent Law

*= A.B +B.C* Idempotent Law

*= B. (A+C)* Distributive Law



*Is Equal to*

Fig. 3.5: Circuit Diagrams depicting Same Logic after Boolean Simplification – Example 02

##### 72

1. Example # 03:

𝐹 = (𝐴̅ )(𝐴 + 𝐵) + (𝐵 + 𝐴)(𝐴 + (𝐵̅)

= ( 𝐴̅ ) 𝐴 + (𝐴̅ ) 𝐵 + (𝐵 + 𝐴) 𝐴 + (𝐵 + 𝐴)𝐵̅ Distributive Law

= ( 𝐴̅ ) 𝐵 + (𝐵 + 𝐴) 𝐴 + (𝐵 + 𝐴) 𝐵̅ Complement Law

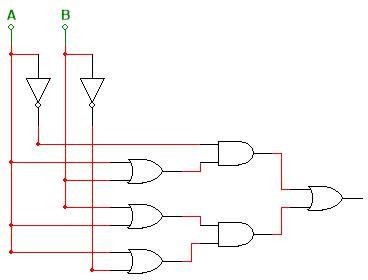
= ( 𝐴̅ ) 𝐵 + 𝐵𝐴 + 𝐴𝐴 + 𝐵𝐵̅+ 𝐴 𝐵̅ Distributive Law

= ( 𝐴̅ ) 𝐵 + 𝐵𝐴 + 𝐴 + A 𝐵̅ Idempotent Law

= 𝐵 (𝐴̅ + 𝐴) + 𝐴 (1 + 𝐵̅ ) Distributive Law

= 𝐵.1 + 𝐴.1 Idempotent & Annulment Law

= 𝐵 + 𝐴 Identity

= 𝐴 + 𝐵 Commutative Law

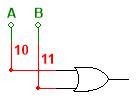
*Is Equal to *

Fig. 3.6: Circuit Diagrams depicting Same Logic after Boolean Simplification – Example 03

##### Morgan’s Theorem:

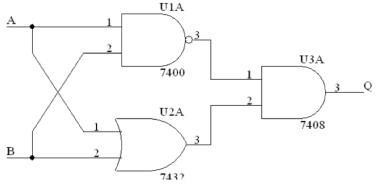
###### De Morgan developed a theorem that allows conversion between logic expressions that has inversions on the output to a different logic expression with the inversions on each of the inputs. This may allow for the simplification of a Boolean Expression by the cancellation of some redundant inversions. There are two Boolean Equations that represent De Morgan's Theorem:

̅𝐴̅̅.̅𝐵̅ = 𝐴̅ + 𝐵̅ Or

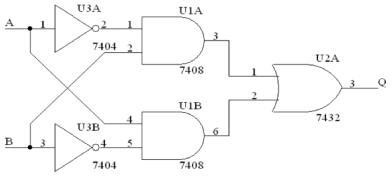
̅𝐴̅̅+̅̅̅̅𝐵̅ = 𝐴̅ . 𝐵̅

**73**

1. **Example (A) – Circuit Diagram:-**



(a)



(b)

Fig. 3.7: Circuit Diagram (a) and Circuit Diagram (b) depicts Same Logic (De – Morgran’s Law)

**PROCEDURE:**

1. At first construct the circuits represented by the original and simplified equations of each example.
2. Construct the Truth Tables of both the original and the simplified equation in each case.
3. Check if the two circuits give the same result.

**74**

**OBSERVATION / RESULTS and DISCUSSION:**

1. Do the original and simplified equation shows same functionality in each case (Example 01, Example 02 and Example 03)? .
2. Do the circuits of Fig. 3.7(a) and Fig. 3.7(b) gives the similar result? .
3. The logical expression for the Fig. 3.7(a) is .
4. Manipulate the equation obtained in observation (3) to reduce the equation that represents the circuit diagram of Fig. 3.7(b)?

**CONCLUSION:**

1. A circuit can easily represent in the form equation
2. It is easy to simplify the circuit using Boolean algebra.
3. After Simplification Hardware Implementation is quite easy.
4. Truth table can easily be converted to Boolean expression and vice-versa.
5. Reduces the complexity.

**EXERCISES:**

1. Simplify the following equation using Boolean Laws. Construct the Truth Tables to verify that the simplified equation gives the same result as that of the original equation.

--

###### 𝐹 = (𝐴 + 𝐶)(𝐴𝐷 + 𝐴 𝐷) + 𝐴𝐶 + 𝐶