Name	Muhammad Asad
Reg. #	2019-EE-383
Marks	4 25-10-20

Experiment # 2

To Understand Laws of Boolean Algebra

Objective:

- Two understand binary arithmetic and binary logic
- Boolean algebra laws

Theory:

Boolean algebra is the basic mathematics needed for the study and design of digital systems. Main ingredients of Algebra are

1. Set of elements
2. Set of operators
3. Number of postulates

In 1904, E. V. Huntington formulated a number of postulates that give us formal definition of Boolean algebra. Boolean algebra is an algebraic structure defined by a set of elements, B, together with two binary operators "+" and "." provided that the following postulates are satisfied.

1. **Closure**
   - with respect to the binary operation OR (+); c=x+y
   - with respect to the binary operation AND (·); c=x.y
2. **Identity**
   - with respect to OR (+) is 0:
     $x + 0 = 0 + x = x$, for $x = 1$ or $x = 0$
   - with respect to AND (·) is 1:
     $x \cdot 1 = 1 \cdot x = x$, for $x = 1$ or $x = 0$
3. **Commutative Law**
   - With respect to OR (+):
     $x + y = y + x$
   - With respect to AND (·):
     $x \cdot y = y \cdot x$

4. **Distributive Law**
   - with respect to the binary operation OR (+):
     $x + (y \cdot z) = (x + y) \cdot (x + z)$ **+** is distributive over **.**
   - with respect to the binary operation AND (·):
     $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ **.** is distributive over **+**

5. **Complement**

For every element x, that belongs to B, there also exists an element x' (complement of x) such that:

- $x + x' = 1$, for $x = 1$ or $x = 0$
- $x \cdot x' = 0$, for $x = 1$ or $x = 0$

6. **Membership**

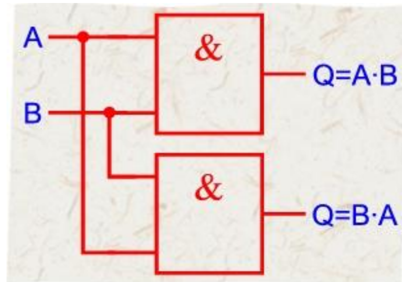
There exists at least two elements, x and y, of the set such that
$x \neq y$ e.g. $0 \neq 1$

LAWS OF BOOLEAN ALGEBRA

Commutative laws $A \cdot B = B \cdot A$ $A \vee B = B \vee A$	**Associative laws** $A \cdot B \cdot C = A \cdot (B \cdot C) = (A \cdot B) \cdot C$ $A \vee B \vee C = A \vee (B \vee C) = (A \vee B) \vee C$
Distributive laws $A \cdot (B \vee C) = (A \cdot B) \vee (A \cdot C)$ $A \vee (B \cdot C) = (A \vee B) \cdot (A \vee C)$	
Absorption $A \vee (A B) = A$ $A \cdot (A \vee B) = A$ $A \cdot (A' \vee B) = A \cdot B$ $A \vee (A' \cdot B) = A \vee B$	**Negation laws (De Morgan)** $\overline{A . B} = A' \vee B'$ $\overline{A \vee B} = A' \cdot B'$

- **Commutative laws**

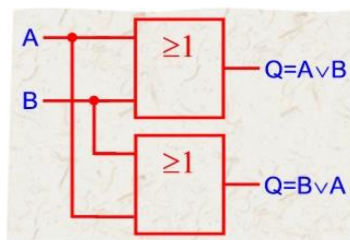
<u>Commutative law for the AND operation</u>



The commutative law for AND operations:

$$A \cdot B = B \cdot A$$

<u>Commutative law for the OR operation</u>

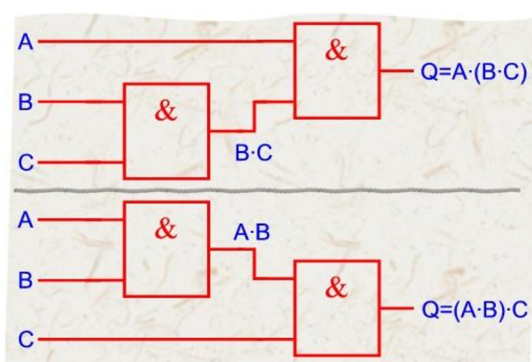


The commutative law for OR operations is:

$$A \vee B = B \vee A$$

- **Associative laws**

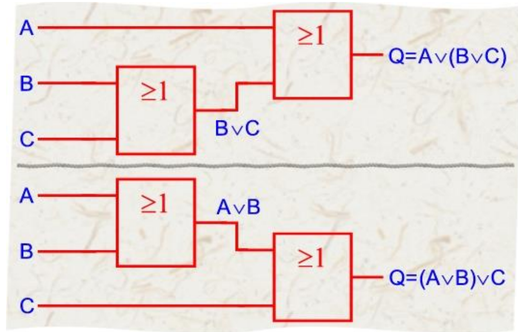
<u>Associative law for the AND operation</u>



The associative law for AND operations is:

$$A \cdot B \cdot C = A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

Associative law for the OR operation

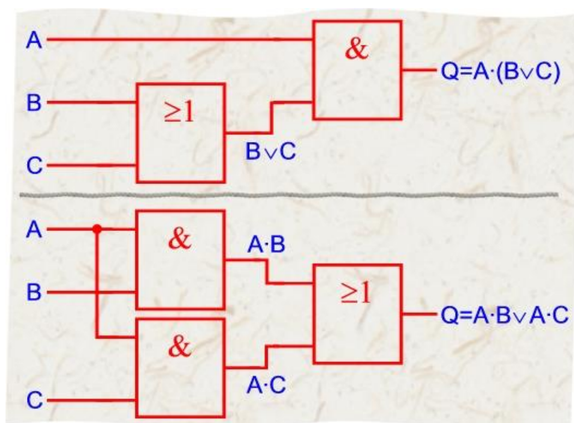


The associative law for OR operations is:

$$A \vee B \vee C = A \vee (B \vee C) = (A \vee B) \vee C$$

- **Distributive laws**

First distributive law

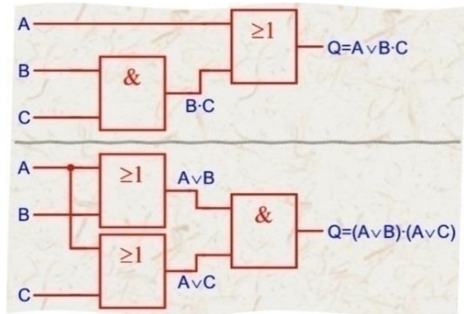


The first distributive law is:

$$A \cdot (B \vee C) = (A \cdot B) \vee (A \cdot C)$$

Note on the sequence of operations:
Similar to ordinary algebra where multiplication and division take precedence over addition and subtraction, here AND operations have priority over OR operations.

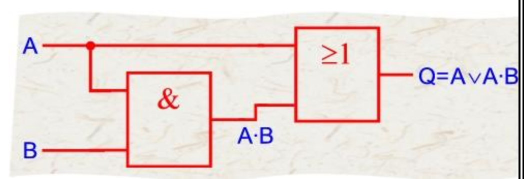
2nd distributive law



The 2nd distributive law is: $A \vee (B \cdot C) = (A \vee B) \cdot (A \vee C)$

- **Absorption**

In the following exercises, we will specifically investigate AND/OR/NOT sequences with two input variables that can be simplified.



Procedure and observation:

Exercise: *Experiment set-up Commutative law, AND*

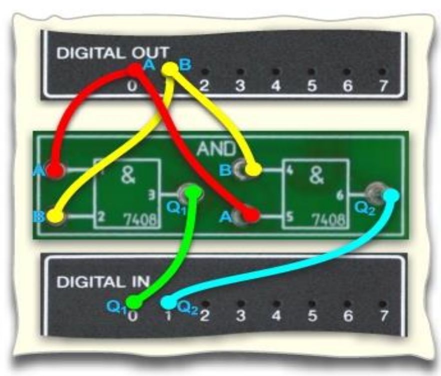


Fig.1 : Experiment set-up Commutative law, AND

Table 1

Q_1	Q_0	I_1	I_0
B	**A**	**Q_2**	**Q_1**
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

↑ B·A ↑ A·B

Exercise: *Experiment set-up Commutative law, OR*

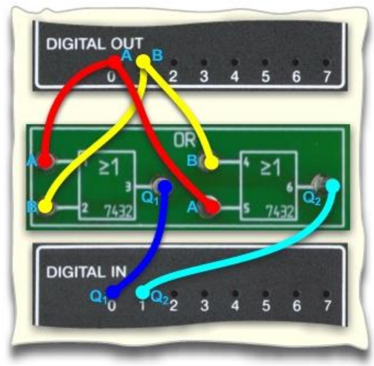


Fig.2 : Experiment set-up Commutative law, OR

Table 2

Q_1	Q_0	I_1	I_0
B	**A**	**Q_2**	**Q_1**
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1
		↑ B ∨ A	↑ A ∨ B

Exercise: *Associative law*AND

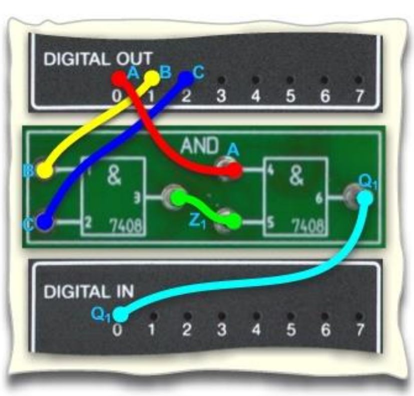


Fig.3a : Experiment set-up - Associative law A·(B·C)

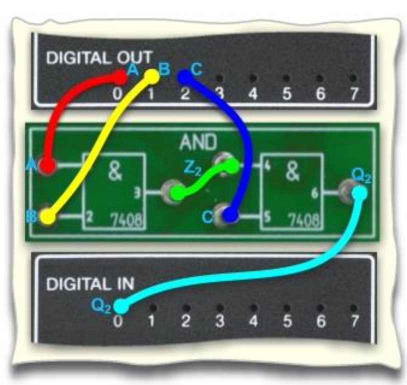


Fig.3b : Experiment set-up - Associative law (A·B)·C

Table 1

			re Fig.1.1	re Fig.1.2
Q_2	Q_1	Q_0	I_0	I_0
C	**B**	**A**	**Q_1**	**Q_2**
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1
			↑ $A \cdot (B \cdot C)$	↑ $(A \cdot B) \cdot C$

Exercise: *Associative law OR*

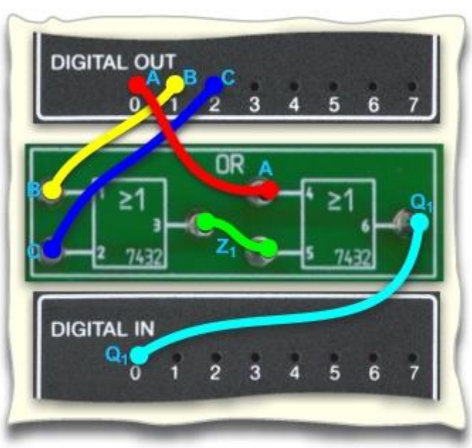


Fig.4a : Experiment set-up - Associative law A∨(B∨C)

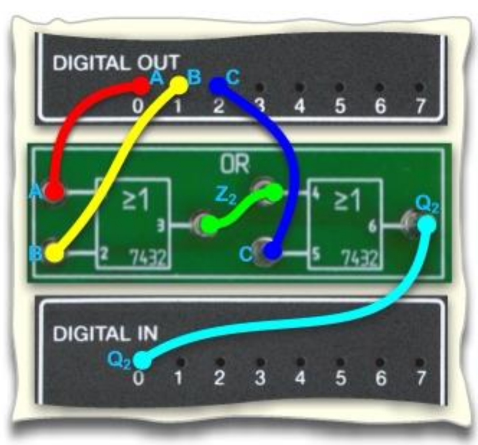


Fig.4b : Experiment set-up - Associative law (A∨B)∨C

Table 2

			re Fig.2.1	re Fig.2.2
Q_2	Q_1	Q_0	I_0	I_0
C	**B**	**A**	**Q_1**	**Q_2**
0	0	0	0	0
0	0	1	1	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
			↑ A∨(B∨C)	↑ (A∨B)∨C

Exercise 1: *1st distributive law*

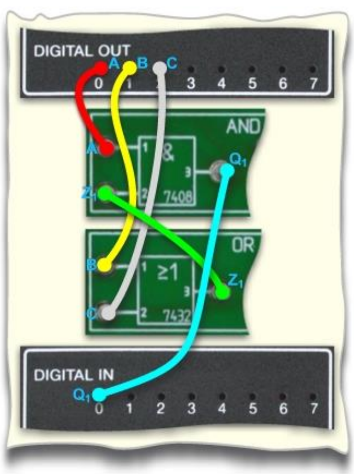


Fig.5a : Experiment set-up - 1st distributive law
$Q_1 = A \cdot (B \lor C)$

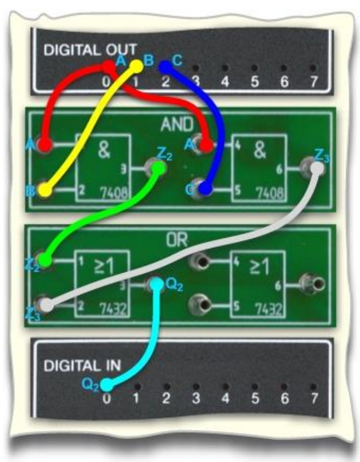


Fig.5b : Experiment set-up, 1st distributive law
$Q_2 = (A \cdot B) \lor (A \cdot C)$

Table 1

Q_2	Q_1	Q_0	I_0	I_0
C	**B**	**A**	**Q_1**	**Q_2**
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

$Q_1 = A \cdot (B \vee C)$

$Q_2 = (A \cdot B) \vee (A \cdot C)$

Exercise : *2nd distributive law*

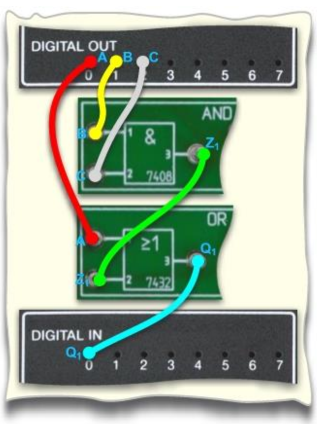


Fig.6a : Experiment set-up,
2nd distributive law
$Q_1 = A \vee (B \cdot C)$

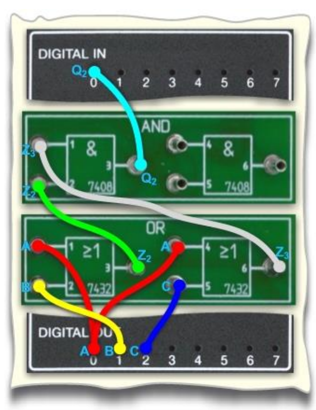


Fig.6b : Experiment set-up,
2nd distributive law
$Q_2 = (A \vee B) \cdot (A \vee C)$

Table 2

Q_2 C	Q_1 B	Q_0 A	I_0 Q_1	I_0 Q_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$Q_1 = A \vee (B \cdot C)$
$Q_2 = (A \vee B) \cdot (A \vee C)$

Exercise: *$Q = A + (A \cdot B)$*

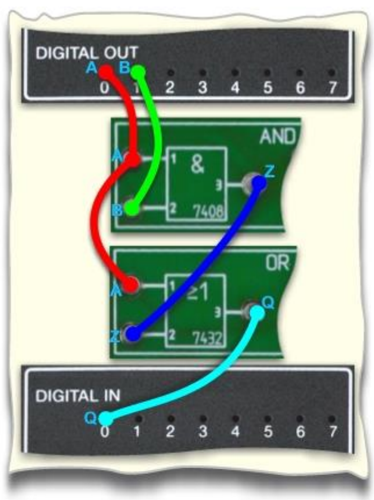


Fig.7 : Experiment set-up - $Q = A \vee (A \cdot B)$

Table 1

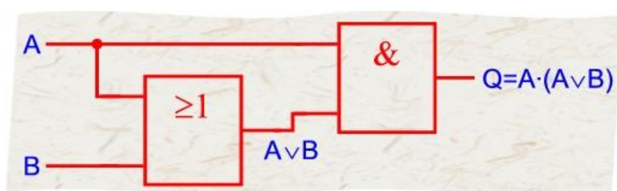
Q_1 Q_0 I_0

B	A	Q
0	0	0
0	1	0
1	0	1
1	1	1

$Q = A \vee (A \cdot B) =$

?

Exercise: **A · (A + B)**



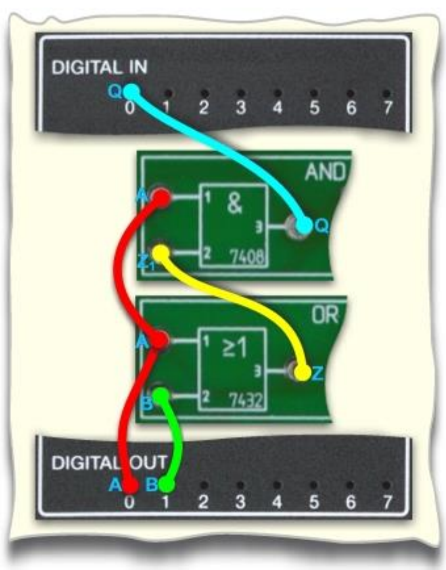


Fig.8 : Experiment set-up - $Q = A \cdot (A \vee B)$

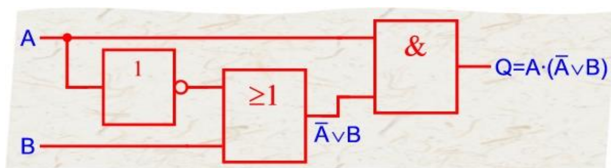
Table 2

Q_1 *Q_0* *I_0*

B	A	Q
0	0	0
0	1	1
1	0	0
1	1	1

$Q = A \cdot (A \vee B) =$?

Exercise 3 **A · (A’ + B)**



Experiment set-up

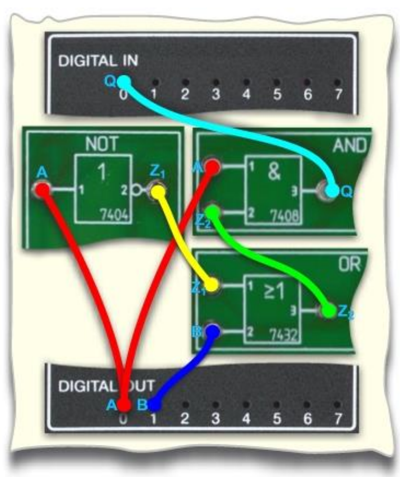


Fig.9: Experiment set-up - Q = A · (A’ + B)

Table 3

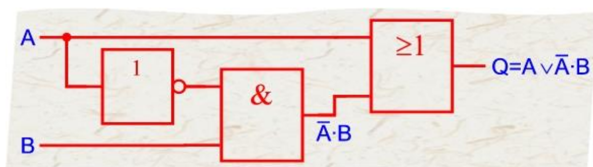
Q_1 *Q_0* *I_0*

B	**A**	**Q**
0	0	0
0	1	0
1	0	0
1	1	1

$Q = A \cdot (A' \vee B) = \boxed{}$

? 0

Exercise 4 **A +(A' · B)**



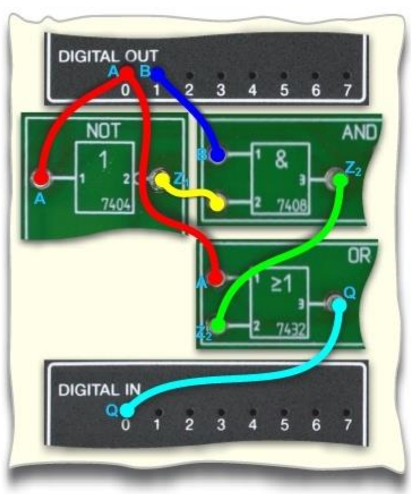


Fig.10: Experiment set-up - Q = A ∨ (A' · B)

Table 4

Q_1 *Q_0* *I_0*

B	A	Q
0	0	0
0	1	1
1	0	1
1	1	1

Q = A ∨ (A’ · B) =	

?

Lab Exercises:

- ***Q2.1*** *Write the commutative law in your own words. Is this law the same as the commutative law for normal multiplication?*

  Commutative law, in mathematics, either of two laws relating to number operations of addition and multiplication, stated symbolically: a + b = b + a and ab = ba. From these laws it follows that any finite sum or product is unaltered by reordering its terms or factors.
  Yes this law the same as the commutative law for normal multiplication.

- ***Q2.2*** *Write the associative law in your own words. Is this law the same as the associative law for normal multiplication & normal addition?*
  In mathematics, either of two laws relating to number operations of addition and multiplication, stated symbolically: a + (b + c) = (a + b) + c, and a(bc) = (ab)c; that is, the terms or factors may be associated in any way desired.

  Yes, *this law the same as the associative law for normal multiplication & normal addition.*
  It is repesented as
  A) A+(B+C)=(A+B)+C=A+B+C
  B) A(B.C)=(A.B).C=A.B.C

- ***Q2.3*** *Write the distributive law in your own words. Is this law the same as the distributive law in normal algebra?*
-

To “distribute” means to divide something or give a share or part of something. According to the distributive property, multiplying the sum of two or more addends by a number will give the same result as multiplying each addend individually by the number and then adding the products together.

Yes this law same as the distributive law in normal algebra.

- ***Q2.4*** *Why we simplify the logical expressions in Boolean algebra?*

Boolean algebra is used to simplify Boolean expressions which represent combinational logic circuits. It reduces the original expression to an equivalent expression that has fewer terms which means that less logic gates are needed to implement the combinational logic circuit. Boolean algebra makes our logic easy and solve it easily other than simple logical expression.
We use logic gates in Boolean algebra to solve the logical expression because it simplfy the logic .