NAME: Dhruvin Chawda

DIVISION/BATCH: A/2

COMPUTER ENGINEERING SEM-3

DIGITAL ELECTRONICS

EXPERIMENT 2

LOGIC SIMPLIFICATION, IMPLEMENTATION USING BASIC GATES.

AIM :

To do Logic Simplification, Implementation using Basic gates.

Y=(A+C)(CD+AC)

Apparatus / Equipment :

Breadboard, IC 7432(OR Gate), IC 7408(AND Gate),

Theory:

A Brief Introduction to Logic Gates

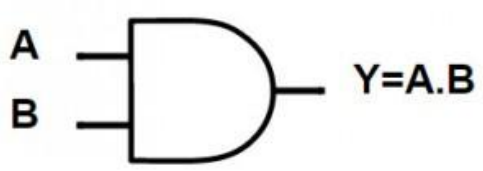
Logic Gates are the basic building blocks of digital electronic circuits. A Logic Gate is a piece of electronic circuit, that can be used to implement Boolean Expressions.

While Laws and Theorems of Boolean Logic are used to manipulate the Boolean Expressions, Logic Gates are used to implement these Boolean Expressions in Digital Electronics.

AND gate, OR gate and NOT gate are the three basic logic gates used in digital electronics. Using these basic logic gates, other Logic Gates like NAND, NOR, Exclusive OR (Ex-OR) and Exclusive NOR (Ex-NOR) are derived.

AND Gate

Logic AND gate is a basic logic gate, with two or more inputs and one output. The output of an AND gate is HIGH only if all the inputs of the gate are HIGH. The output for all the other cases of the inputs is LOW. The logic symbol and the truth table of an AND gate is shown below.



A	B	Y = A AND B
0	0	0

0	1	0
1	0	0
1	1	**1**

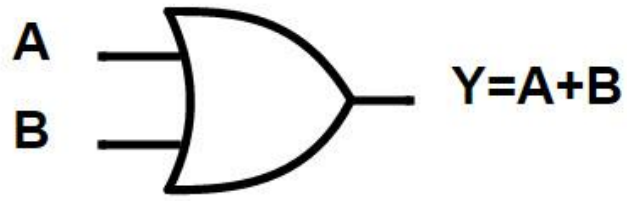
If ‘A’ and ‘B’ are the two inputs of an AND Gate, the output expression is written as:

$$Y = A \cdot B \text{ or } Y = A\ B$$

It is read as “Y EQUALS A AND B”.

OR Gate

The OR Gate is used to preform logical ‘OR’ operation. OR Gate also contains two or more inputs and one output. The output of an OR gate is HIGH if either of the inputs are HIGH. The output is LOW when all the inputs are LOW. The logic symbol and the truth table of an OR gate is shown below.



A	B	Y = A OR B
0	0	0
0	1	**1**
1	0	**1**
1	1	**1**

If ‘A’ and ‘B’ are the two inputs of an OR Gate, the output expression is written as:

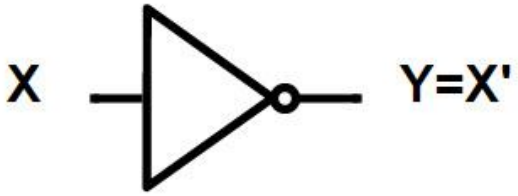
$$Y = A + B$$

It is read as “Y EQUALS A OR B”.

NOT Gate

Logic NOT gate is another basic logic gate with one input and one output. The output of the NOT Gate is always the complement of the input. If the input is HIGH, the output is LOW

and if the input is LOW, the output is HIGH. The logic symbol and the truth table of a NOT gate is shown below.



X	Y = X’
0	1
1	0

NOT Gate is used to produce the complement of a variable in Boolean Algebra. So, it is also called as Complementing or Inverting Circuit.

Rules of Boolean Algebra:

Basic rules of Boolean algebra.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, or C can represent a single variable or a combination of variables.

DeMorgan’s Theorems:

DeMorgan’s first theorem is stated as follows:

The complement of a product of variables is equal to the sum of the complements of the variables.

Stated another way, The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

The formula for expressing this theorem for two variables is

$$\overline{XY} = \overline{X} + \overline{Y}$$

DeMorgan’s second theorem is stated as follows:

The complement of a sum of variables is equal to the product of the complements of the variables.

Stated another way, The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

The formula for expressing this theorem for two variables is

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$

Procedure :

Boolean algebra finds its most practical use in the simplification of logic circuits.

If we translate a logic circuit's function into symbolic (Boolean) form, and apply certain algebraic rules to the resulting equation to reduce the number of terms and/or arithmetic operations, the simplified equation may be translated back into circuit form for a logic circuit performing the same function with fewer components.

If equivalent function may be achieved with fewer components, the result will be increased reliability and decreased cost of manufacture.

To this end, there are several rules of Boolean algebra presented in the above section for use in reducing expressions to their simplest forms.

Once, the logic has been simplified, the circuit diagram can be implemented using gates corresponding to the simplified equation obtained.

Logic Diagram/ Truth Table/ K-maps :

Exp 2 & 3

Equation : $(A+C)(CD+AC)$

* logic Simplification (k maps).

$(A+C)(CD+AC)$

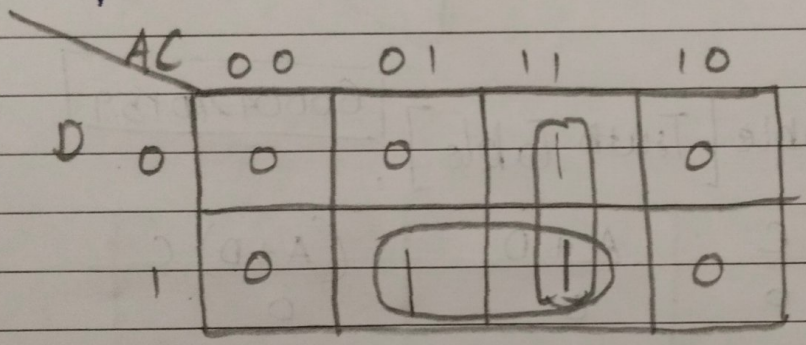
$= ACD + AC + CD + AC$

$= ACD + ACD + AC\bar{D} + ACD + \bar{A}CD$

$= ACD + AC\bar{D} + \bar{A}CD$

$\quad\;\; 111 \quad\; 110 \quad 011$

K-map:



logic simpl. : $ABC + CD = \boxed{C(A+D)}$

60004210159

* Using Boolean laws:

$(A+C)(CD+AC)$

$= (A+C)(A+D)C$

$= (A+CD)C \quad [\because (A+B)(A+C) = A+BC]$

$= AC + CDC$

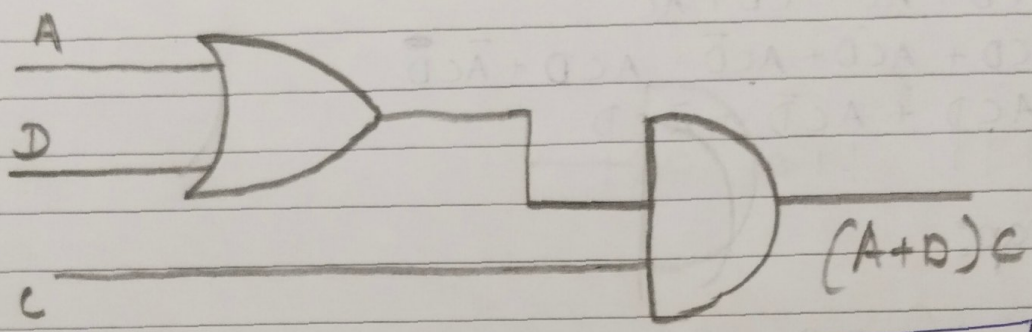
$= AC + CD \quad [\because A \cdot A = A]$

$= \boxed{C(A+D)}$

Exp 2

* Implemenation using Basic gates. $((A+D)C)$.

Here 2 gates are required, one AND gate & one OR gate (for A & D).



* Observation Table [Truth Table]. 600042101 59

A	D	C	A+D	(A+D) C
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Inference / conclusion :

Thus, we have simplified and implemented a logic expression using Basic gates.