NAME: DHRUVIN CHAWDA

DIVISION/BATCH: B/3

SUBJECT : DIGITAL ELECTRONICS

EXPERIMENT 2

LOGIC SIMPLIFICATION, IMPLEMENTATION USING BASIC GATES.

AIM :

To simplify the Logic and implement the same using Universal gates.

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

Apparatus / Equipment :

Breadboard, IC 7402(NOR Gate), IC 7400(NAND Gate), Power Supply.

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.

NAND Gate

Logic NAND gate is a basic logic gate, with two or more inputs and one output. The output of an NAND gate is Low (or 0) only if all the inputs of the gate are HIGH. The output for all the other cases of the inputs is High (or 1). The logic symbol and the truth table of a NAND gate is shown below.

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

$$y = \sim(A.B)$$

It is read as “Y EQUALS A NAND B”.

NOR Gate

The NOR Gate is used to preform logical ‘NOR’ operation. NOR Gate also contains two or more inputs and one output. The output of an OR gate is Low (or 0) if either of the inputs are HIGH. The output is High (or 1) when all the inputs are LOW. The logic symbol and the truth table of an OR gate is shown below.

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

$$y = \sim(A+B)$$

It is read as “Y EQUALS A NOR B”.

EXOR Gate

XOR gate is a digital logic gate that gives a true output when the number of true inputs is odd. So for a two input XOR gate, a true output results if one, and only one, of the inputs to the gate is true. If both inputs are false or both are true, a false output results.

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

$$y = A \oplus B$$

It is read as “Y EQUALS A XOR B”.

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.** $(A + B)(A + C) = A + BC$

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:

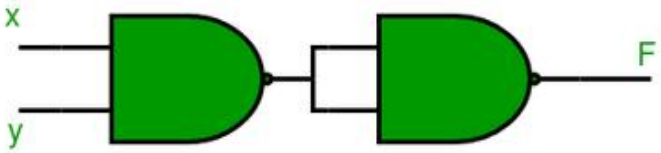
The complement of a sum of variables is equal to the product of the complements of the variables.

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

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

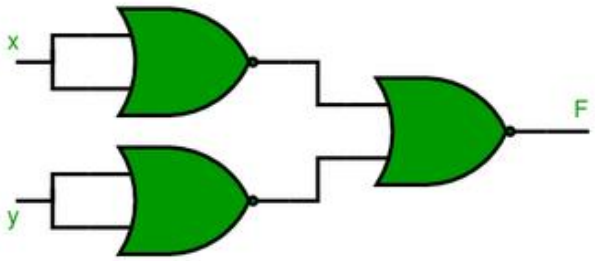
Making an AND gate using NAND



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

gates:

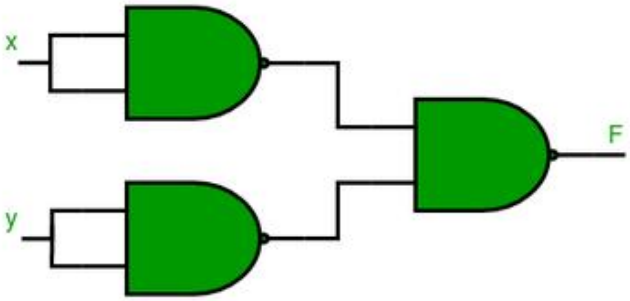
Making an AND gate using NOR



x	y	F
0	0	0
1	0	0
1	1	1
0	1	0

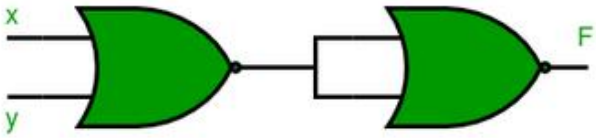
gates:

Making an OR gate using NAND gates:



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Making an OR gate using NOR gates:



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Procedure :

The simplified logic expression can be implemented using universal gates on a breadboard using 2 NAND and NOR gates each. We first connect the inputs of A and D to a nor gate and the output of that nor gate will go as a dual input for another NOR gate, this is how we realize A+D using NOR gates. The output from the above implementation should go as an input in a NAND gate along with the input from C and that output must go as a dual input for another NAND gate, this is how we realize AND using NAND gates. The output from that will be the final required output y.

Logic simplification (Boolean)/Observation table/Circuit diagram:

Exp 2 & 3

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

* logic Simplification (k maps).

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

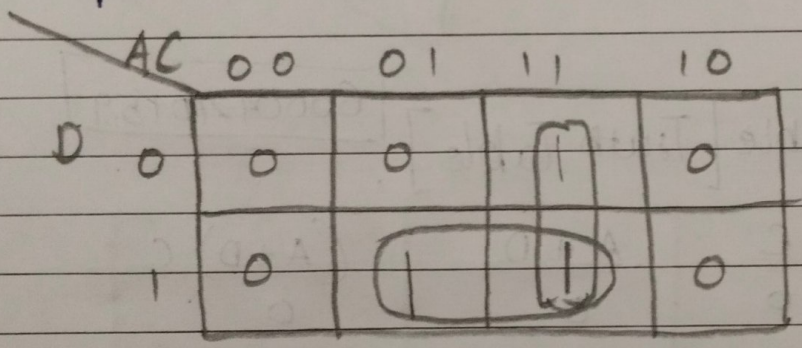
$= ACD + AC + CD + AC$

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

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

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

K-map:



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

60004210159

* Using Boolean laws:

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

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

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

$= AC + CDC$

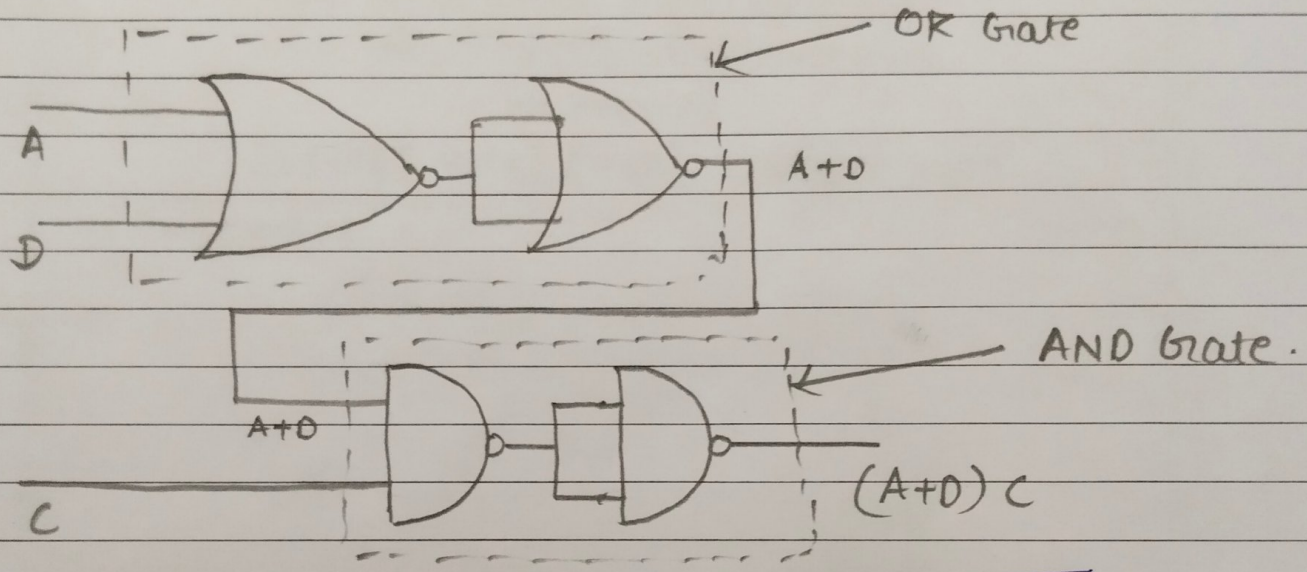
$= AC + CD \quad [\because A \cdot A = A]$

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

Exp 3

* Implementation Using Universal gates [(A+D)C].

Here Implementation is done by using Universal gates [NOR & NAND gates].



* Observation Table [Truth Table]. [600042101599]

A	D	C	$\overline{A+D} = A+D$	$\overline{(A+D)\cdot C} = (A+D)C$
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 Universal gates.