Lecture Four
Complements in Digital Computers

1. Introduction

The basic building blocks of a computer are called **logical gates**. Gates are basic circuits that have at least one (and usually more) **input** and exactly one output. Input and output values are the logical values **true** and **false**. In computer architecture it is common to use 0 for false and 1 for true.

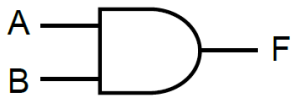
Gates have no memory. The value of the output depends only on the current value of the inputs. A useful way of describing the relationship between the inputs of gates and their output is the truth table. In a truth table, the value of each output is tabulated for every possible combination of the input values. We usually consider three basic kinds of gates, **AND**-gates, **OR**-gates, and **NOT**-gates (or **Inverters**).

2. Basic Gates

1) AND Gate

The AND operation is represented by a dot(**.**) or by the absence of an operator. **E.g.** A.B=F ,AB=F are all read as A AND B=F. the logical operation AND is interpreted to mean that F=1 if A=1 and B=1 otherwise F=0.

Symbol



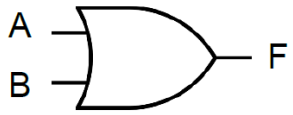
Truth Table

A	B	F = (A.B)
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

2) OR Gate

The OR operation is represented by a (+) sign for example, A+B=F is interpreted as A OR B=F meaning that F=1 if A=1 or B=1 or if both A=1 and B=1. If both A and B are 0, then F=0.

Symbol



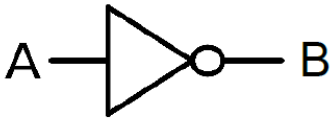
Truth Table

A	B	F = (A+B)
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

3) NOT Gate or INVERTER

The inverter is a little different from AND and OR gates in that it always has exactly one input as well as one output. Whatever logical state is applied to the input, the opposite state will appear at the output. This operation is represented by a bar or a prime. For example, A′=A=B is interpreted as NOT A =B.

Symbol



Truth Table

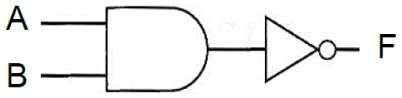
A	B
0	1
1	0

3. Combined Gates

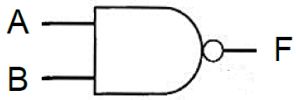
1) NAND Gate

The NAND-gate is an AND-gate with an inverter on the output. This operation is represented by F=$\overline{A.B}$. So instead of drawing several gates like this:

Symbol



We draw a single NAND-gate with a little ring on the output like this:



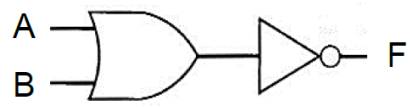
Truth Table

A	B	F = $(\overline{A.B})$
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

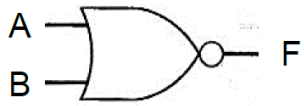
2) NOR Gate

The NOR-gate is an OR-gate with an inverter on the output. This operation is represented by F=$\overline{A + B}$. So instead of drawing several gates like this:

Symbol



We draw a single NOR-gate with a little ring on the output like this:



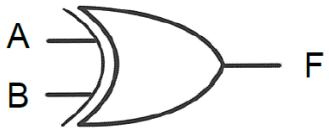
Truth Table

A	B	F = $(\overline{A+B})$
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

3) Exclusive OR Gate(Ex-OR/ XOR): -

The Exclusive-OR-Gate is similar to an OR-gate. It can have an arbitrary number of inputs, and its output value is 1 if exactly one input is 1 (and thus the others 0). Otherwise, the output is 0. This operation is represented by F=A.$\overline{B}$+$\overline{A}$.B = A $\oplus$ B .

Symbol



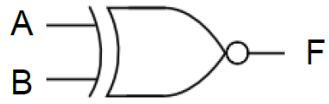
Truth Table

A	B	F = A⊕B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

4) Exclusive NOR Gate(Ex-NOR/ XNOR): -

The Exclusive-NOR-Gate is similar to an NOR-gate. It can have an arbitrary number of inputs, and its output value is 1 if the two inputs are of the same values (1 and 1 or 0 and 0). Otherwise, the output is 0. This operation is represented by $F=A.B+\overline{A}.\overline{B}$ = A⊙B .

Symbol



Truth Table

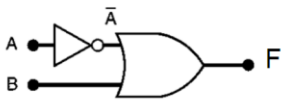
A	B	F = A⊙B
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

4. Examples

Draw the following functions?

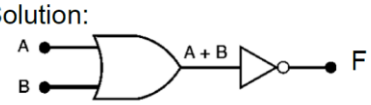
1) F=$\overline{A}$+B

Solution:



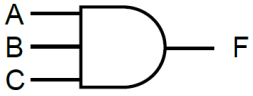
F=$\overline{A+B}$

Solution:



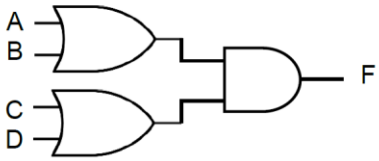
2) F=A.B.C

Solution:



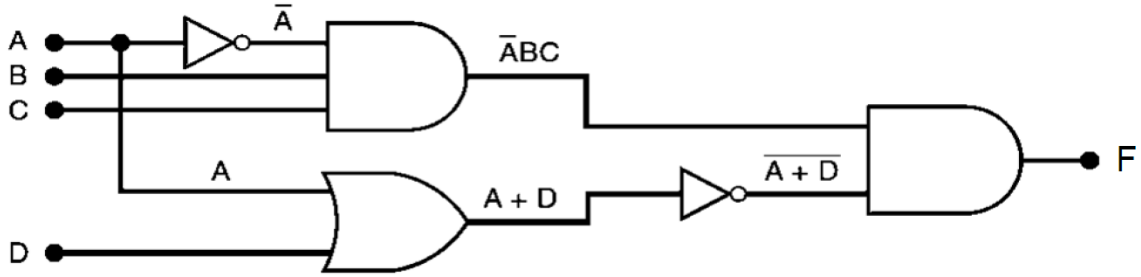
3) A+B.C+D

Solution:



4) $F=\overline{A}BC(\overline{A+D})$

Solution:

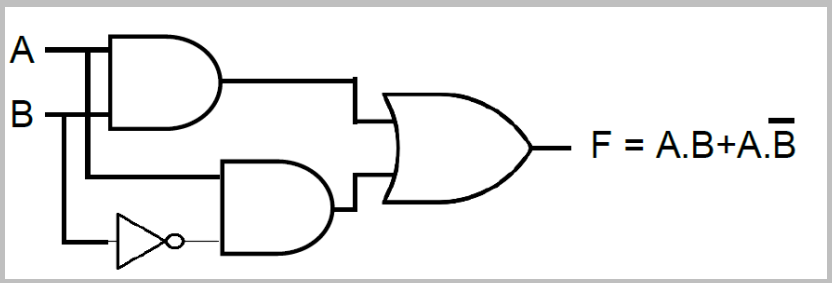


Activity

1) $F = [D + \overline{(A + B)C}] \bullet E$

2) $F = (A + B)(\overline{B} + C)$

3) Find the truth table of F?



4. Boolean Algebra

Boolean Algebra is an algebra, which deals with binary numbers & binary variables. Hence, it is also called as Binary Algebra or logical Algebra. A mathematician, named George Boole had developed this algebra in 1854. The variables used in this algebra are also called as Boolean variables. Boolean Algebra is set of rules, used to simplify the given logic expression without changing its functionality.

4.1 Basic Laws of Boolean Algebra

These laws help simplify logic expressions:

- **Identity Law**
  - $A + 0 = A$
  - $A \cdot 1 = A$
- **Null Law**
  - $A + 1 = 1$
  - $A \cdot 0 = 0$
- **Idempotent Law**
  - $A + A = A$
  - $A \cdot A = A$
- **Complement Law**
  - $A + A' = 1$
  - $A \cdot A' = 0$
- **Commutative Law**
  - $A + B = B + A$
  - $A \cdot B = B \cdot A$
- **Associative Law**
  - $(A + B) + C = A + (B + C)$
  - $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- **Distributive Law**
  - $A \cdot (B + C) = AB + AC$
  - $A + (BC) = (A + B)(A + C)$
- **Involution Law**
  - A''=A
- **DeMorgan's Theorem**

This theorem is useful in finding the **complement of Boolean function**. It states that the complement of logical OR of at least two Boolean variables is equal to the logical AND of each complemented variable.

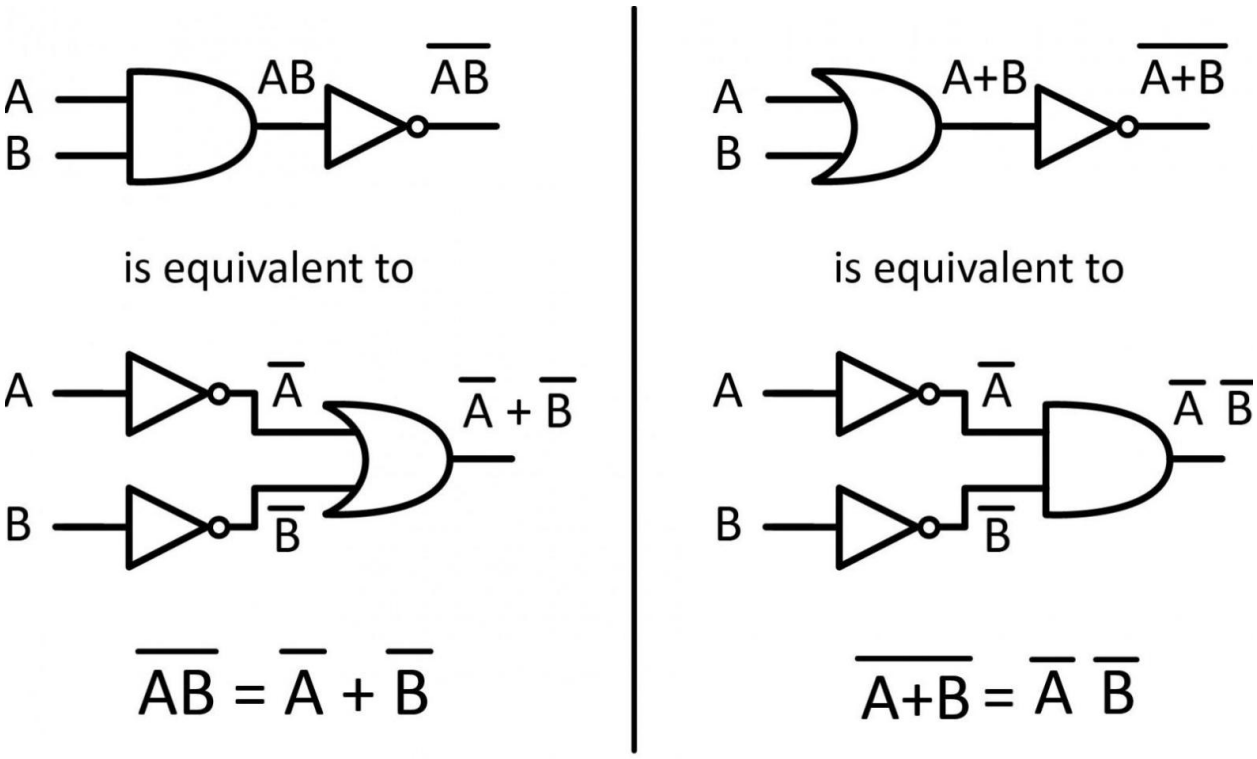
DeMorgan's theorem with 2 Boolean variables x and y can be represented as

$$(x + y)' = x'.y'$$

The dual of the above Boolean function is

$$(x.y)' = x' + y'$$

Therefore, the complement of logical AND of two Boolean variables is equal to the logical OR of each complemented variable. Similarly, we can apply DeMorgan's theorem for more than 2 Boolean variables also.



De Morgan's Laws

4.3 Simplification of Boolean Expression

When a Boolean expression is implemented with logic gates, each literal in the function is designated as input to the gate. The literal may be a primed or unprimed variable. ***Minimization of the number of literals and the number of terms leads to less complex circuits as well as less number of gates***, which should be a designer's aim.

There are **several** methods to minimize the Boolean function. In this lecture, **simplification** or minimization of complex algebraic expressions will be shown with the ***help of theorems of Boolean algebra***.

Example: *Simplify the Boolean function F=AB+ BC + B′C.*
Solution.

F = AB + BC + B′C
= AB + C (B + B′)
= AB + C

Example: *Simplify the Boolean function F= A + A′B.*

Solution.

F = A+ A′B
= (A + A′) (A + B)
= A + B

Example *: Simplify the Boolean function F= A′B′C + A′BC + AB′.*

Solution. F = A′B′C + A′BC + AB′
= A′C (B′+B) + AB′
= A′C + AB′

Activity

Use Boolean algebra to simplify the following Boolean functions:

1. F = AB + (AC)′ + AB′C(AB + C) (Ans: F=1)

2. F = ((XY′ + XYZ)′ + X(Y + XY′))′ (Ans: F=0)

3. F = XYZ + XY′Z + XYZ′ (Ans: F=X(Y+Z))