Chapter 4: Part II **Boolean Algebra**

**Hierarchical structuring of a computer system Architecture**

Actually a computer system composed of many layers of software and hardware. But for simplicity we see the computer in Abstract way. This avoids details and makes the user to concern about the outer layer that is easy for the user to communicate with the computer. Generally modern computers consist of multi-layers or levels. The following description shows the different levels.

7. Application level: Language depends on application program

Hide operational details of lower levels

6. High-order languages level: Machine-independent programming language

Program can transfer from one to another comp. easily

5. Assembly level: Assembly language

4. Operating system level: Masks details of hardware from programmer and responsible for program creation, execution, access to I/O and access to a file

3. Machine level: Machine Language

2. Micro programming level: Basic hardware component construction and their interaction

1. Logic gate level: Basic component for microprogramming

0. Device level: Transistors, capacitors & resistors

The digital circuitry in digital computers and other digital systems is designed, and its behavior is analyzed, with the use of mathematical discipline known as Boolean algebra. The name is in honor of an English mathematician George Boole, who proposed the basic principles of this algebra in 1854.

This Boolean algebra becomes very convenient in two areas:

Analysis: It is an economical way of describing the function of digital circuitry.

Design: given a desired function, Boolean algebra can be applied to develop a simplified implementation that function.

As any algebra it make use of logical variables, which will have a value true or false (1 or 0), and logical operations.

A Boolean function is described as follows: F: X Y where X and Y are set of 0’s and 1’s

A Boolean function with n input variables could operate on 2n possible combinations.

**Logic gates(elements)**

In the hierarchical structuring of a computer the logic gate level is above the device level.

These logical elements are primary components of the computer hardware. Given a desired function, Boolean algebra can be applied to develop a simplified implementation of that function using interconnection of logical gates.

A gate is an electronic circuit that produces an output signal that is a simple Boolean operation on its input signals.

Each gate is defined in three ways: graphical symbols, algebraic notation, and truth table. A truth table is a table that shows the output of a logical function for all possible combinations of input values. Each gate has one/two or more inputs and only one-output signals. The signals are either 0 or 1.

F(X,Y)=XY+X

G(X,Y,Z)=XY+XYZ+YZ

H(x,y)=x y = x.y + x.y

There are 3 basic Boolean operators NOT, AND, OR : -, . and +respectively

## **Construction of Logical Circuit From Logical Function and Truth Table**

Any Boolean function can be implemented in electronic form as a network of gates. If you have the truth table you can generate the Boolean function and you can also use the Boolean function to construct the logical circuit. The reverse is also possible.

Generally, before constructing any logical circuit it has to be simplified based on the rules of Boolean equations/theorems in order to minimize the number of required logical gates.

**Rules of the Boolean (equation) operators**

|  |  |  |  |
| --- | --- | --- | --- |
| x. 0 =0 | x+1=1 | | Null |
| x. x =x | x+¯x=1 | | Idempotent |
| x. ‾x =0 | x+¯x=1 | | Inverse |
| x. 1 =x | x +o =x | | Identity |
| x(x +y)=x | x+ x . y =x | | Absorption |
| x .y =y .x | x +y = y +x | | Commutative |
| (x .y) .z =x.(y. z) | (x +y) +z+ x+(y +z) | | Associative |
| A(Ã +B) = AB | A + Ã’B = A + B | | Absorbition |
| x.(y +z) = x. y + x. z | (x +y) (x +z) =x +y z | | distributive |
| x . y = x + y | x +y = x.y | | DeMorgan’s |
| X = x | x =x | | Double Negative |
| A +AB =A | | A(A +B)= A | |

**Example: Simplify the following**

1. **x’[yz + y’z + yz’ + y’z’]**

**= x’[(y+y’)z + (y+y’)z’]**

**= x’[z+z’]**

**= x’**

1. **x’y’z’+x’yz’+xy’z’+xyz’**

**= x’(y’+y)z’ +x(y’+y)z’**

**= (x’+x)(y’+y)z’**

**= z’**

1. Y = AB C D + AB CD + AB C D + ABCD

= BCD ( A + A) + ABCD + ABCD

=BCD + ABCD + ABCD

= CDC (B + AB) + ABCD

**Exercise**: Simplify

**1.**



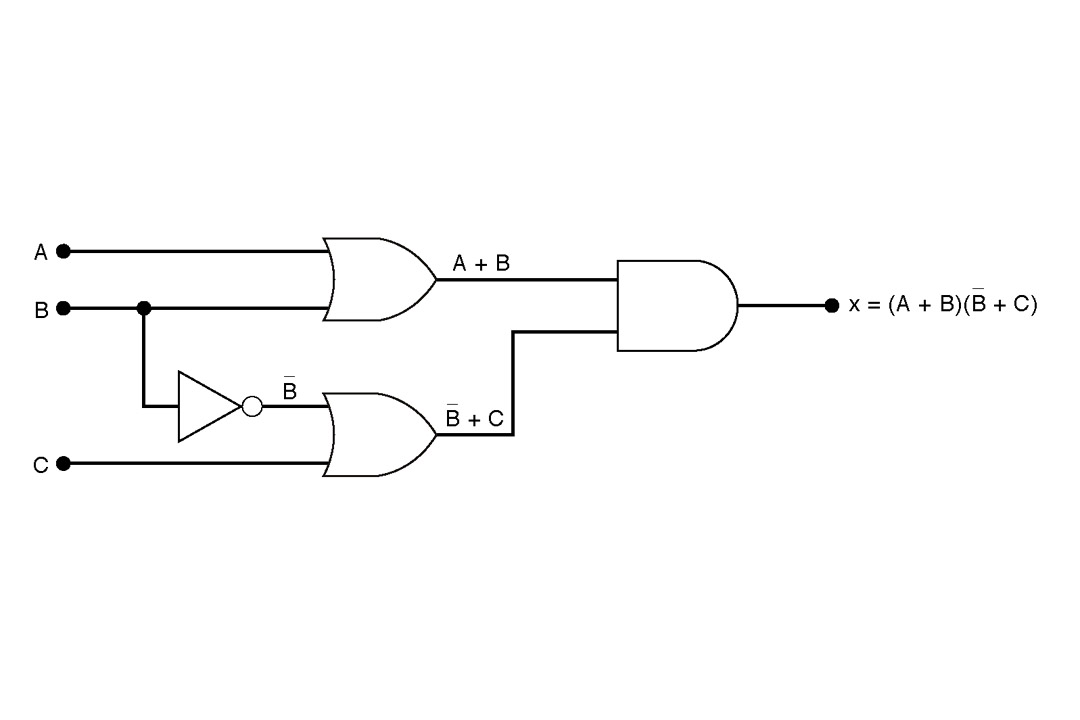
**2.**



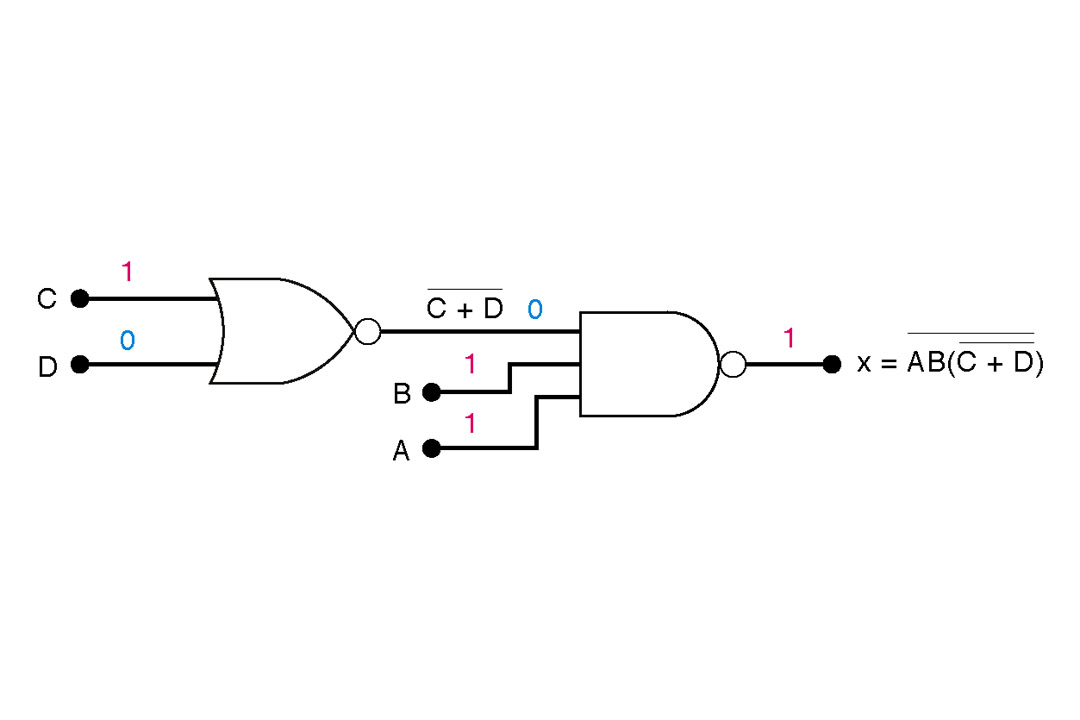
**Examples:**

**1. Draw the circuit diagram to implement the expression**

**X= (A+B)( B‾‾+C)**

****

**2. Implement the logic circuit that has the expression using only NOR and NAND gates**

****

**Exercise**:

**Construct the logic circuit of**

1. F(A,b)= AB +AB
2. F(x,y) =x +y + x y + ( x+ y) x y
3. F(A,B,C,D) = AB + AC + BD + CD
4. L = CBA + CBA + CBA + CBA + CBA