Digital System Design

Dr. Oday A.L.A Ridha
University of Baghdad 2019

Syllabus

-Digital system in general;
-Simplification of logic circuits;
-Combinational circuit design;
-Design using programmable logic circuits such as ROM, PLA, PAL and GAL
-Design of synchronous sequential circuit;
-Algorithmic state machine;
-Design of asynchronous sequential circuit;
-Hardware Description Language VHDL .

References

1. M. Morris Mano, Digital design, Prentice/ Hall International, 1984.
2. Fredreck J. Hill and Gerald R. Peterson, Digital logic and microprocessors.
3. Malvino, Digital principles and applications.
4. Tony R. Kuphaldt, Lessons In Electric Circuits, Volume IV "Digital", 2005.
5. Others

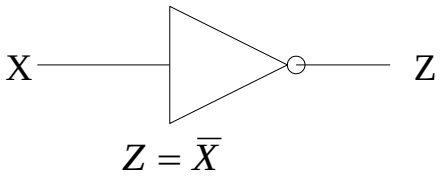
Digital System

Digital design is concerned with the design of digital electronic circuits. The subject is also known by other names such as *logic design, switching circuits, digital logic, and digital systems.* Digital circuits are employed in the design of systems of digital computers, electronic calculators, digital control devices, digital communication equipments, and many other applications that required electronic digital hardware.

Basic elements of the digital systems

The basic elements of any digital system are *logic gates.* These gates are:

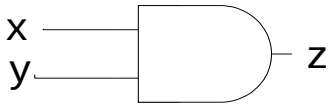
1- **NOT** gate (inverter)



$Z = \overline{X}$

Truth table	
X	Z
0	1
1	0

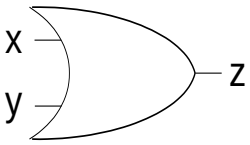
2- **AND** gate (multiplication)



$$Z = X \cdot Y$$

Truth table		
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

3- **OR** gate (sum)

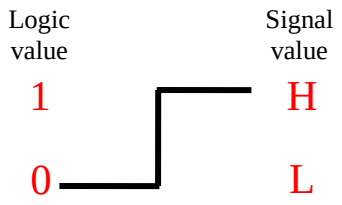


$$Z = X + Y$$

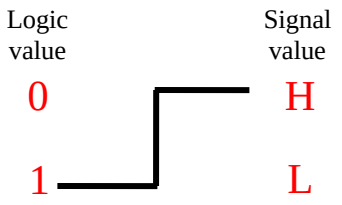
Truth table		
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Positive and negative logic

There are two choices for logic-level assignment. Choosing the high-level ***H*** to represent logic-1 defines a ***positive logic*** system. Choosing the low-level ***L*** to represent logic-1 defines a ***negative logic*** system.



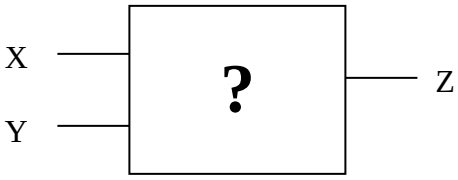
a) Positive logic



b) Negative logic

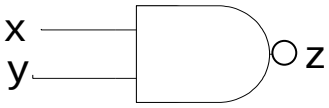
Example:

Truth table		
X	**Y**	**Z**
L	*L*	*H*
L	*H*	*H*
H	*L*	*H*
H	*H*	*L*



a) if we use *positive logic* system

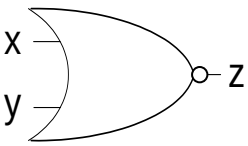
Truth table		
X	**Y**	**Z**
0	*0*	*1*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*



NAND gate

b) if we use *negative logic* system

Truth table		
X	**Y**	**Z**
1	*1*	*0*
1	*0*	*0*
0	*1*	*0*
0	*0*	*1*



NOR gate

Simplification of logic circuits

Logic circuits can be simplified using one or combination of the following methods

1) Boolean algebra
2) Karnaugh mapping
3) Tabular method

Boolean algebra

Like "normal" algebra, Boolean algebra uses alphabetical letters to denote variables. Unlike "normal" algebra, though, Boolean variables are always CAPITAL letters, never lower-case. Because they are allowed to possess only one of two possible values, either 1 or 0, each and every variable has a complement: the opposite of its value. For example, if variable "A" has a value of 0, then the complement of A has a value of 1. Boolean notation uses a bar above the variable character to denote complementation.

Basic Boolean identities:

a) Additive (equivalent to OR logic function)

1) $A+0=A$
2) $A+1=1$
3) $A+\overline{A}=1$
4) $A+A=A$

b) Multiplicative (equivalent to AND logic function)

1) $A\cdot 1=A$
2) $A\cdot 0=0$
3) $A\cdot A=A$
4) $A\cdot \overline{A}=0$

c) Double complement

$\overline{\overline{A}}=A$

Basic Boolean algebraic properties:

1) $A+B=B+A$ *(commutative property of addition)*
2) $A\cdot B=B\cdot A$ *(commutative property of multiplication)*
3) $A+(B+C)=(A+B)+C$ *(associative property of addition)*
4) $A(B\cdot C)=(A\cdot B)C$ *(associative property of multiplication)*

5) $A(B+C) = A \cdot B + A \cdot C$ *(distributive property)*

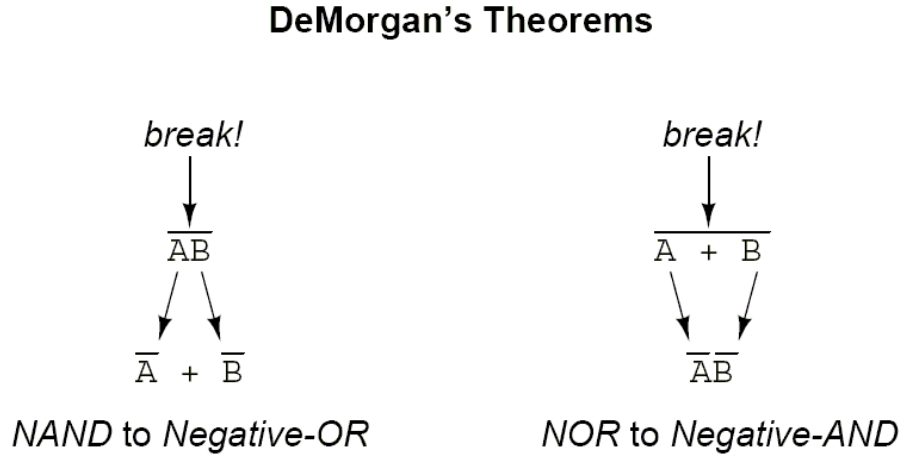
Boolean rules are

1) $A + AB = A$
2) $A + \overline{A}B = A + B$
3) $(A + B)(A + C) = A + BC$

DeMorgan's Theorems

A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra.

DeMorgan's theorem may be thought of in terms of breaking a long bar symbol. When a long bar is broken, the operation directly underneath the break changes from addition to multiplication, or vice versa, and the broken bar pieces remain over the individual variables.



Boolean rules for simplification

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.

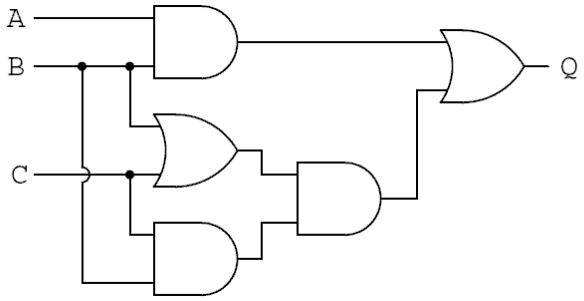
Steps for simplifying logic circuits

Step 1: Convert the logic circuit to a Boolean expression. Label each gate output with a Boolean sub-expression corresponding to the gates' input signals, until a final expression is reached at the last gate;

Step 2 : simplify Boolean expression, obtained from first step, using basic rules and properties of the Boolean algebra;

Step 3: convert simplified Boolean expression to a logic circuit. Evaluate the expression using standard order of operations: multiplication before addition, and operations within parentheses before anything else.

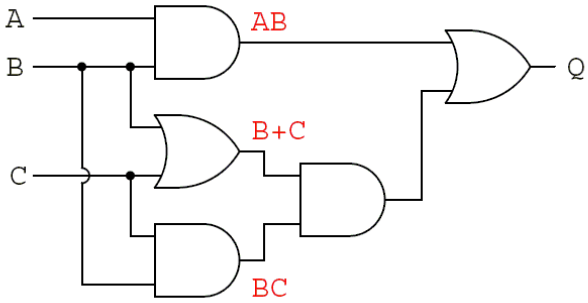
Example: simplify the following logic circuit



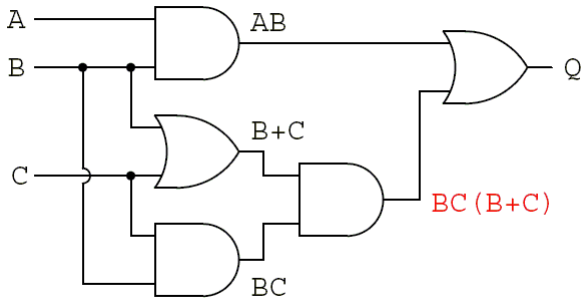
Solution:

Step 1:

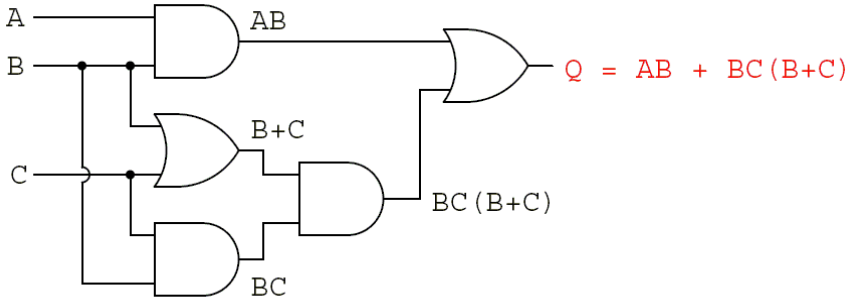
a)



b)



c)



Step 2:

AB + BC(B + C)

↓ Distributing terms

AB + BBC + BCC

↓ Applying identity **AA = A** to 2nd and 3rd terms

AB + BC + BC

↓ Applying identity **A + A = A** to 2nd and 3rd terms

AB + BC

↓ Factoring **B** out of terms

B(A + C)

Step 3: