



*Ministry of Higher Education
And Scientific Research
Southern Technical University
Al Qurna Technical Institute
Electrical Power Department*



Lecture notes: Digital Electronics

*Assistant Lecturer: Ehsan Mohsin
Email: e.mahsin@stu.edu.iq*

Week One to Four

Numerical Systems:

Since the ancient time, people used their fingers during their daily counting and calculating process which make the process easy and fast. This process represents the decimal system nowadays which has string of ten digits .The value of each digit depends on the weight of the position in the string. This means the value of the digit varies based on the weight of the position. The weight of the digit increases when its position is located on the left and decreased when its position is located on the right of the number. Furthermore, each number system has a base or radix which reflects the total number of digits in the string. For example the decimal system has ten digits so it has the base or radix of ten.

However, we can summarize the three points on which the value of the digit depends

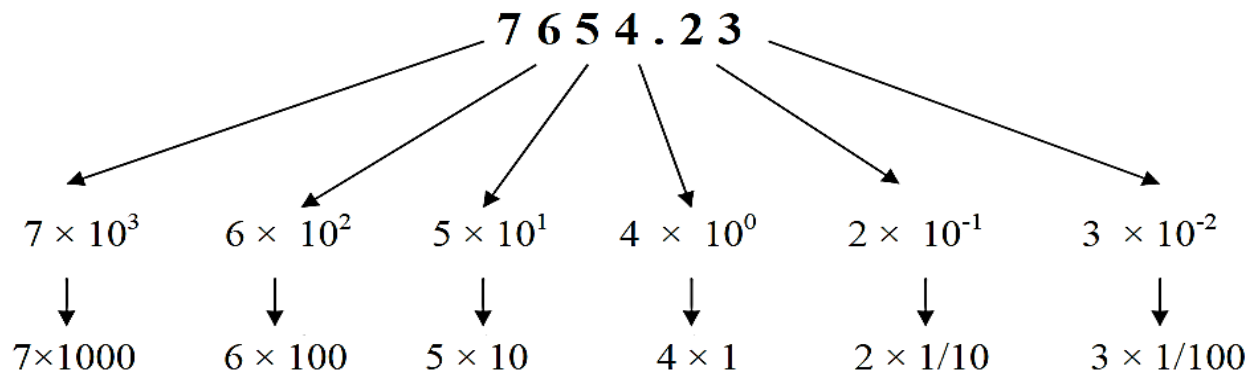
- The digit
- The position of the digit in the number (weight)
- The base of the number system

There are four common number systems include the binary, the octal, the decimal, and the hexadecimal systems. These number systems are useful in many digital systems such as computers, calculators, and microprocessor.

Decimal Number System:

The most common and daily used number system is called the decimal system. All the arithmetical calculations undertaken by human being are executed on the basis of this number system. It consists of a string of ten digits (**0,1,2,3,4,5,6,7,8,9**), that why it called the system of base or radix **10**.The base of each digit is 10 while the weight of the power starts from zero near to the decimal point and increases to the left. If the decimal number has further fraction to the right of the decimal point, then the negative weight will start from -1.

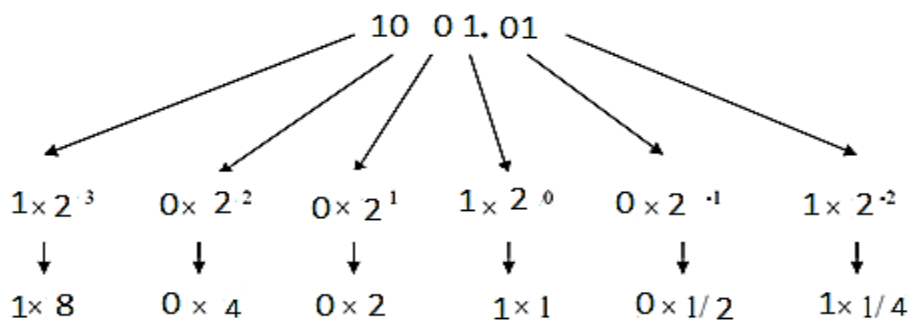
For example we can write the decimal number $(7654.23)_{10}$ as below.



Binary Number System:

The binary system is less complicated than the decimal system because it has only two digits (**0, 1**). So it's a system of base or radix **2**. It is the basic system for mathematics and digital electronics circuitry using the logic gates, the binary system is used internally by almost all modern computers and computer-based devices such as mobile phones where each digit is referred to as a bit.

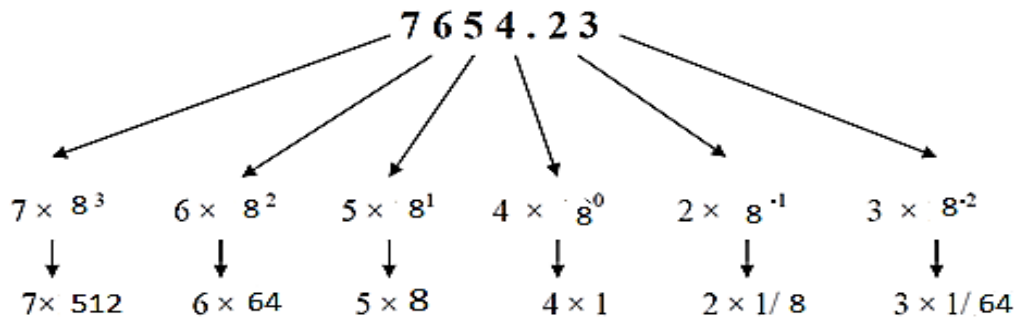
For example we can write the binary number $(1001.01)_2$ as follows.



Octal Number System:

Octal number system has a base or radix 8. Eight different digits namely (**0, 1, 2, 3, 4, 5, 6, 7**) are used to express octal numbers. Conversion of octal numbers to their decimal equivalents can be achieved by using the same way which was followed to convert binary numbers to decimal numbers.

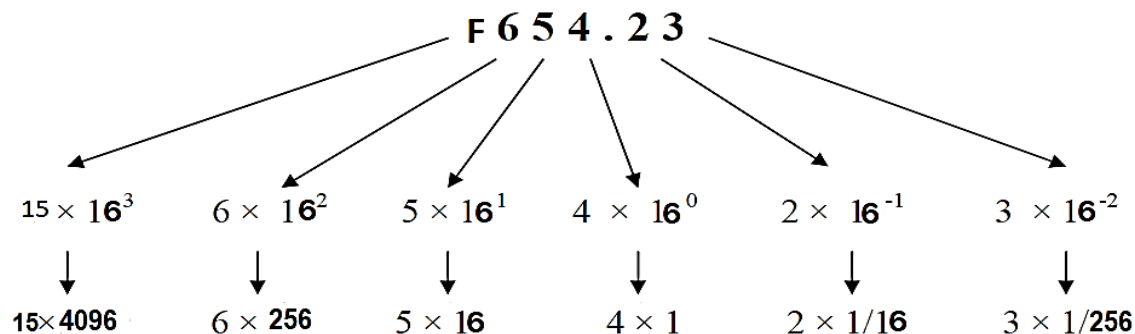
For example we can express the octal number $(7654.23)_8$ as below



Hexadecimal Number System:

It called the system of base or radix 16 because it has 16 digits (**0 to 9, A, B, C, D, E, F**). Where **A, B, C, D, E, and F** represent the decimal numbers **10, 11, 12, 13, 14, and 15** respectively.

For example we can write the hexa-decimal number $(F654.23)_{16}$ as below.

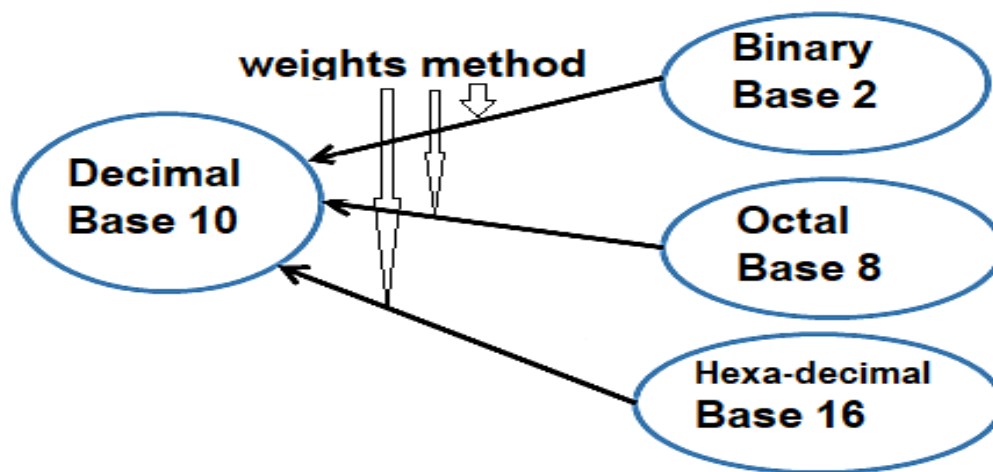


Conversion Among Number Systems:

The conversion among the number systems is a viable operation which the designers of logic system must experience it carefully. In order to easily understand the conversion among these numbers, we can divide it to categories where each category has the same approach of conversion.

Conversion of Non-decimal to Decimal Number System:

The conversion of non-decimal to decimal number depends on the weight of each digit and the base (radix) of the non- decimal number. Thus, each digit will be multiplied by the base and its corresponding weight so that the sum of the multiplication process of the all digits gives the decimal number. The process of evaluating the magnitude of the non-decimal number and convert it to decimal number called the weights method.



Binary –to– Decimal Conversion:

The decimal number of any binary number can be calculated by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.

Example1: Convert the binary number $(1101101)_2$ to decimal.

Solution:-

$$(1101101)_2 = 2^0 \times 1 + 2^1 \times 0 + 2^2 \times 1 + 2^3 \times 1 + 2^4 \times 0 + 2^5 \times 1 + 2^6 \times 1$$

$$1 + 0 + 4 + 8 + 0 + 32 + 64 = (109)_{10}$$

Example2: Convert the binary number $(1101.01)_2$ to decimal.

Solution:-

$$(1101.01)_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 1 \times 1 + 0 \times 2 + 1 \times 4 + 1 \times 8 + 0 \times 1/2 + 1 \times 1/4$$

$$= 1 + 0 + 4 + 8 + 0 + 0.25 = (13.25)_{10}$$

.....
Example3: Convert $(0.1011)_2$ to decimal.

Solution:-

$$2^0 \times 0 + 2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 1 + 2^{-4} \times 1 = 0 + 0.5 + 0.125 + 0 + 0.0625 = (0.6875)_{10}$$

We can tabulate the digits of the decimal number and its corresponding binary bits as shown in the table below.

Decimal	0	1	2	3	4	5	6	7	8	9
Binary	000	001	010	011	100	101	110	111	1000	1001

Octal –to– Decimal Conversion:

The same weights methods can be used to convert octal to decimal number where the base of the octal digits is 8 instead of 2 for the binary system. Examples for octal numbers are **$(10.03)_8$** , **$(23.22)_8$** , **$(31.3)_8$** , **$(0.113)_8$** etc.

Example1: Convert the octal number $(2374)_8$ to decimal

Solution:-

$$(2374)_8 = 2 \times 8^3 + 3 \times 8^2 + 7 \times 8^1 + 4 \times 8^0 = 1024 + 192 + 56 + 4 = (1276)_{10}$$

Example2: Convert the octal number to decimal $(203.65)_8$

Solution:-

$$\begin{aligned} (203.65)_8 &= 3 \times 8^0 + 0 \times 8^1 + 2 \times 8^2 + 6 \times 8^{-1} + 5 \times 8^{-2} \\ &= 3 \times 1 + 0 \times 8 + 2 \times 64 + 6 \times 1/8 + 5 \times 1/64 = 3+0+132+0.75+0.078125 = \\ &(135.828125)_{10} \end{aligned}$$

Hexadecimal –to– Decimal Conversion:

The same weights methods can be used to convert Hexadecimal to decimal number where the base of the hexadecimal digits is **16** and the base multiplied by the weight from right to left. Examples for Hexadecimal numbers are **$(10.F3)_{16}$** , **$(AE.2E)_{16}$** , **$(B1.3)_{16}$** , **$(0.FF13)_{16}$** etc.

.....
Example1: convert $(D21.F9)_{16}$ to decimal.

Solution:-

$$\begin{array}{ccccccc}
 D & 2 & 1 & . & F & 9 & \\
 16^2 & 16^1 & 16^0 & . & 16^{-1} & 16^{-2} & \\
 =1 \times 1 + 2 \times 16 + 13 \times 16^2 + 15 \times 16^{-1} + 9 \times (16^{-2}) & & & & & & \\
 = 1 + 32 + 3328 + 0.9375 + 0.03515625 = (3361.972656)_{10}
 \end{array}$$

Example2: convert $(2F1CD)_{16}$ to decimal

Solution:-

$$\begin{array}{ccccccc}
 2 & F & 1 & C & D & & \\
 16^4 & 16^3 & 16^2 & 16^1 & 16^0 & & \\
 =13 \times 1 + 12 \times 16 + 1 \times 256 + 15 \times 4096 + 2 \times 65536 & & & & & & \\
 =13 + 192 + 256 + 61440 + 131072 = (192973)_{10}
 \end{array}$$

Conversion of Decimal to Non-decimal Numbers:

The decimal system is divided into integer and fraction parts before the conversion into non-decimal system. **The integer part is divided while the fraction part is multiplied by the base to which the decimal number is convert (binary, octal, and hexadecimal)** .The result of conversion of the two parts is combined together to obtain the final result of conversion.

I. Conversion of Integer Part

The conversion of decimal integer number **requires repetitive division by the base of the number to which the integer number is required to convert** while the remainder of division process will be listed aside. Next, the result of division will be divided to the second time and keep the remainder too. **This procedure will continue till the obtained division result is zero.** The result of conversion is collected from the successive remainder Colom from bottom upward and will be written from left to right.

II. Conversion of Fraction Part

The conversion of decimal fraction number **requires repetitive multiplication of the decimal fraction by the base to which the decimal fraction is required to convert**. After each multiplication, the integer part is the equivalent fraction number which is obtained by writing the integer digits to the right of the fraction point in the same sequence. **If the fractional part of the product process becomes exactly zero at a certain stage, then the equivalent fraction is finite**, otherwise, the fraction is non-terminating to the desired degree of accuracy.

The following examples explain the conversion procedure of decimal to non-decimal numbers

Example1: Convert $(53)_{10}$ to its equivalent binary number.

Solution:-

Decimal	Integer	Remainder
$53/2=$	26	1
$26/2=$	13	0
$13/2=$	6	1
$6/2=$	3	0
$3/2=$	1	1
$1/2=$	0(Stop)	1(0.5x2)

Hence,

$(53)_{10}$ is equivalent to $(110101)_2$

Example2: Convert $(0.625)_{10}$ to binary

Solution:-

Decimal Numbers to Binary Number Conversion		
Multiplication	Integer	Fraction
$0.625 \times 2 = 1.25$	1	0.25
$0.25 \times 2 = 0.5$	0	0.5
$0.5 \times 2 = 1.0$	1	0(Stop)

Hence, $(0.625)_{10}$ is equivalent to $(0.101)_2$

Example3: Convert $(32.25)_{10}$ to binary

Solution: - Integer Part

Decimal	Integer	Remainder
$32 \div 2 =$	16	0
$16 \div 2 =$	8	0
$8 \div 2 =$	4	0
$4 \div 2 =$	2	0
$2 \div 2 =$	1	0
$1 \div 2 =$	0(Stop)	1(0.5x2)

Thus, $(32)_{10}$ is equivalent to $(100000)_2$

Fraction part

Decimal Numbers to Binary Number Conversion		
Multiplication	Integer	Fraction
$0.25 \times 2 = 0.5$	0	0.5
$0.5 \times 2 = 1.0$	1	0(Stop)

Thus, $(0.25)_{10}$ is equivalent to $(0.01)_2$

Hence, $(32.25)_{10}$ is equivalent to $(100000.01)_2$

Example 4: Convert $(175.23)_{10}$ to its equivalent octal number

Solution: - Integer part

Decimal	Integer	Remainder
$175 \div 8 =$	21	7(.875x8)
$21 \div 8 =$	2	5
$2 \div 8 =$	0 (Stop)	2(0.25x8)

Thus, $(175)_{10} = (257)_8$

Fraction part

Decimal Numbers to Octal Number Conversion		
Multiplication	Integer	Fraction
$0.23 \times 8 = 1.84$	1	0.84
$0.84 \times 8 = 6.72$	6	0.72
$0.72 \times 8 = 5.76$	5	0.76
$0.76 \times 8 = 6.08$	6	0.08
$0.08 \times 8 = 0.64$	0	0.64
$0.64 \times 8 = 5.12$	5	0.12

Thus, $(0.23)_{10} = (0.165605)_8$

Hence, $(175.23)_{10}$ is equivalent to $(257.165605)_8$

Example 5: Convert $(125.34375)_{10}$ to hexadecimal

Solution: - Integer part

Decimal	Integer	Remainder
$125 \div 16 =$	13	$13(0.8125 \times 16)$
$13 \div 16 =$	0	$7(0.4375 \times 16)$

Thus, $(125)_{10} = (7D)_{16}$

Fraction part

Decimal Numbers to Hexa.. Number Conversion		
Multiplication	Integer	Fraction
$0.34375 \times 16 = 5.5$	5	0.5
$0.5 \times 16 = 8.0$	8	0

Thus, $(0.34375)_{10} = (0.58)_{16}$

Hence, $(125.34375)_{10} = (7D.58)_{16}$

❖ Conversion of Binary to Octal and Hexa-decimal Numbers and Vice-versa

Conversion from Binary to Octal Number and Vice versa:

The binary number is divided into 3 bit string when converting it to octal number. The division of integer number starts from the binary fraction point towards the left while the division of fraction number starts from the binary fraction point towards the right. If the binary bits are not a multiple of 3, then we add zeros on the left and right of the integer and fraction part respectively. Finally, each group (3 bit string) is replaced by its corresponding octal digit. The same procedure is considered to convert octal to binary numbers.

Exp1: Convert the $(10011101110)_2$ to octal.

Solution: -

The binary bits are divided to groups of 3 bit string

.....

010 011 101 110

2 3 5 6 **Thus, $(10011101110)_2 = (2356)_8$**

Exp2: Convert $(0.0101111)_2$ to octal.

Solution:-

010 111 100

2 3 4 **Thus, $(0.0101111)_2 = (0.234)_8$**

Exp3: Convert $(1101.11)_2$ to octal.

Solution:-

001 101 . 110

1 5 . 6 **Thus, $(1101.11)_2 = (15.6)_8$**

Exp4: Convert $(35.12)_8$ to binary

Solution:-

3 5 . 1 2

011 101 . 001 010 **Thus, $(35.12)_8 = (11101.00101)_2$**

Conversion of Binary to Hexa-decimal and Vice Versa:

The binary number is divided into 4 bit string when converting it to hexa-decimal number. Each group (4 bit string) is replaced by its corresponding hexa-decimal digit. The same procedure is considered to convert hexadecimal to binary numbers.

Exp1: Convert $(10110011)_2$ to hexadecimal.

Solution:-

The binary digits are divided to groups of 4 bits

1011 0011

B 3 **Thus, $(10110011)_2 = (B3)_{16}$**

Exp2: Convert $(\text{FAC.1D})_{16}$ to its equivalent binary number

Solution:-

F A C . 1 D

1111 1010 1100 . 0001 1101

Thus, $(\text{FAC.1D})_{16} = (111110101100.00011101)_2$

Exp3: Convert $(00101110.1010)_2$ to its hexadecimal equivalent.

Solution:-

0011 1110 . 1010

2 E . A Thus, $(00101110.1010)_2 = (2\text{E.A})_{16}$

Exp4: Convert $(\text{AB.6D})_{16}$ to its binary equivalent

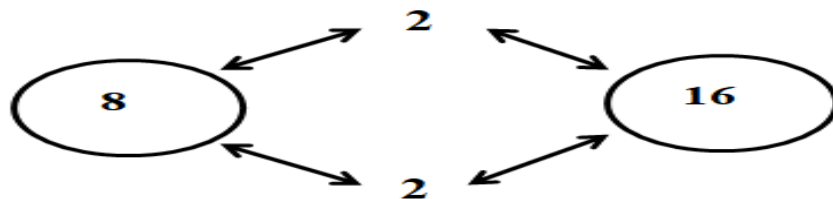
Solution:-

A B . 6 D

1010 1011 . 0110 1101 **Thus, $(\text{AB.6D})_{16} = (10101011.01101101)_2$**

Conversion of Octal to Hexa-decimal and Vice Versa

There is no direct conversion from octal to hexa-decimal number. Therefore, the binary system is a mediator to which octal system is converting before the final conversion to hexadecimal or vice versa. First, the octal number is converted to binary digits, where groups of 3 bits string replace the octal digits. Second, the binary number is converted to 4 bits string and be replaced by its hexa-decimal equivalents digits. The figure below demonstrates how the conversion method is executed between the octal and hexadecimal numbers.



Exp1: Convert $(44.62)_8$ to hexa-decimal equivalent number

Solution:- 4 4 . 6 2

$\underline{100} \quad \underline{100} \quad . \quad \underline{110} \quad \underline{010}$ Then, $(44.62)_8 = (100100.110010)_2$

$\underline{0010} \quad \underline{0100} \quad . \quad \underline{1100} \quad \underline{1000}$

2 4 . C 8 Then, $(100100.110010)_2 = (24.C8)_{16}$

Hence, $(44.62)_8 = (24.C8)_{16}$

Exp2: Convert $(81.E)_{16}$ to octal equivalent number.

Solution:- 8 1 . E

1000 0001 . 1110 Then, $(81.E)_{16} = (10000001.1110)_2$

010 000 001 . 111 000

2 0 1 . 7 0 Then, $(10000001.1110)_2 = (201.7)_8$

Hence, $(81.E)_{16} = (201.7)_8$

❖ See the given table of conversion from binary to octal and hexadecimal and vice versa

Binary	Octal	Binary	Hexa-decimal
000	0	0000	0
001	1	0001	1
010	2	0010	2
011	3	0011	3
100	4	0100	4
101	5	0101	5
110	6	0110	6
111	7	0111	7
1000	10	1000	8
1001	11	1001	9
1010	12	1010	A
1011	13	1011	B
1100	14	1100	C
1101	15	1101	D
1110	16	1110	E
1111	17	1111	F

Exercises:

1 -Convert $(332.761)_{10}$ to binary number

2-Convert $(54.689)_{10}$ to octal number

3-Convert $(543.321)_{10}$ to hexa-decimal number

4-Convert $(AC.E45)_{16}$ to octal number

5-Convert $(111011.1101)_2$ to decimal number

6- Convert $(665.74)_8$ to decimal number

7-Convert $(111.11)_2$ to $(\dots)_{10}$, $(\dots)_8$, $(\dots)_{16}$

8- Find the value of X for the following

$$(X)_8 = (5543.87)_{10}, \quad (X)_{16} = (1110111.11)_2, \quad (X)_{10} = (F9.26)_{16}$$

Arithmetic Operations of Binary System:

Decimal system is dominant in our daily arithmetic operations .For instance, addition, subtraction, division, and multiplication. Binary system has limited applications because it has only two digits (0, 1).However, the same arithmetic operations can be performed using binary system.

Binary Addition

The simplest way of binary addition is performed between two binary bits which leads to four possibilities as shown in the table below.

Case	A+B	Sum	Carry
1	0+0	0	0
2	0+1	1	0
3	1+0	1	0
4	1+1	0	1

Note: $1+1 = 2$ which is 10 in binary number, where 0 is the sum and 1 is the Carry for the next column.

Exp1: Add the following numbers $(1011.01)_2$, $(11010.1)_2$

Sol:-

$$\begin{array}{r} 01011.01 \\ + 11010.10 \\ \hline 100101.11 \end{array}$$

Exp2: Add the following numbers $(11011.101)_2$, $(1110.11)_2$

Sol:-

Note: $1+1+1=1 \longrightarrow 1$ (Carry)

$$\begin{array}{r} 01110.110 \\ + 11011.101 \\ \hline 10010.011 \end{array}$$

Binary Subtraction

The simplest subtraction method is between two binary bits where there are four possible cases as shown in the table below.

Case	A-B	subtract	Borrow
1	0-0	0	0
2	0-1	1	1
3	1-0	1	0
4	1-1	0	0

Exp1: Subtract the following number $(1011)_2$ from $(1101.1)_2$

Sol:-

$$\begin{array}{r} 1101.1 \\ - 1011.0 \\ \hline 0010.1 \end{array}$$

Exp2: Subtract the following number $(001100)_2$ from $(0011010)_2$

Sol:-

$$\begin{array}{r} 0011010 \\ - 001100 \\ \hline 0001110 \end{array}$$

Binary Multiplication

There are four possible cases of multiplying two binary bits as tabulated below.

Case	AxB	Multiplication
1	0x0	0
2	0x1	0
3	1x0	0
4	1x1	1

Exp1: Multiply the following two numbers $(101)_2$, $(1010)_2$

$$\begin{array}{r}
 1010 \\
 \times 101 \\
 \hline
 1010 \\
 0000 \\
 1010 \\
 \hline
 110010
 \end{array}$$

Binary Division

The division process between two binary bits has four cases as tabulated below

Case	A÷B	Division
1	0÷0	?
2	0÷1	0
3	1÷0	?
4	1÷1	1

Exp1: divide the number $(110)_2$ by $(11)_2$

Sol:-

$$\begin{array}{r}
 10 \\
 11 \overline{) 110} \\
 \underline{-11} \\
 000
 \end{array}$$

Then the answer is $(10)_2$

Exp2: divide the number $(110)_2$ by $(10)_2$

Sol:-
Then the answer is $(11)_2$

$$\begin{array}{r}
 11 \\
 10 \overline{) 110} \\
 \underline{-10} \\
 010 \\
 \underline{-10} \\
 000
 \end{array}$$

Complements of Binary Numbers:

Complement function is commonly used in computers for saving negative numbers and also to replace the subtraction process by repetitive addition method, which enables the electronic circuits to process the addition and subtraction methods.

1st Complement

Finding the 1st complement of binary number is by changing all 1s to 0s and all 0s to 1s. While the 2nd complement is the 1st complement plus 1 on the right.

Exp1: find the 1st complement of the following number.

Sol:-

Given Number	1	0	1	0	1
1st complement	0	1	0	1	0

Exp2: find the 1st and 2nd complement of the following number

Sol:-

$$\begin{array}{r}
 10110010 \\
 01001101 \\
 + \quad \quad 1 \\
 \hline
 01001110
 \end{array}$$

Binary number
1's complement
add 1
2's complement

1st Complement Subtraction

Exp1: subtract the number $(110)_2$ from $(1010)_2$ using the 1st complement.

$$\begin{array}{r}
 \text{المطروح منه} \quad 1010 \\
 \text{المطروح} \quad 110 \text{ —} \\
 \hline
 \text{تكملة مراتب المطروح} \quad 0110 \\
 \text{المتتم لـ 1 للمطروح} \quad 1001 \\
 \hline
 \text{المتتم لـ 1 للمطروح} \quad 1001 \\
 \text{المطروح منه} \quad 1010 \quad + \\
 \hline
 \text{المرتبة الإضافية} \rightarrow 10011 \\
 \hline
 \text{ناتج الطرح} \rightarrow 0100 \quad + 1
 \end{array}$$

1. إكمال مراتب العدد الأقل عددا بالمراتب (المطروح أو المطروح منه)

2. إيجاد المتتم لـ 1 للعدد المطروح.

3. جمع المتتم لـ 1 للمطروح مع المطروح منه.

4. نلاحظ نتيجة الجمع للخطوة 3 وكما يلي:

إذا كان هناك واحد في المرتبة الإضافية فنقوم بجمعه

مع بقية العدد ويكون هو ناتج الطرح الموجب

Exp2: Subtract the number $(10101)_2$ from $(1011)_2$ using the 1st complement

Sol:-

$$\begin{array}{r}
 01011 \\
 10101 \text{ ---} \\
 \hline
 01010 \\
 01011 \text{ +} \\
 \hline
 \cdot 10101 \\
 01010 \text{ ---}
 \end{array}$$

إذا لم يظهر واحد في المرتبة الإضافية نأخذ المتمم ال 1 لناتج الجمع ويكون هو ناتج الطرح (سالِب)

2nd Complement Subtraction

We follow the same procedure of 1st complement in addition to add 1 to the 1st complement before the final addition

Exp1: Subtract the number $(110)_2$ from $(1010)_2$ using the 2nd complement

Sol:-

$$\begin{array}{r}
 1010 \\
 0110 \text{ ---} \\
 1001 \\
 \hline
 1 \text{ +} \\
 1010 \\
 1010 \text{ +} \\
 \hline
 10100 \\
 \underbrace{}_{\uparrow}
 \end{array}$$

1. إكمال مراتب العدد الأقل مراتب .

2. إيجاد المتمم لـ 2 للعدد المطروح .

3. جمع المتمم لـ 2 للعدد المطروح مع المطروح منه

4. نلاحظ نتيجة الجمع للخطوة 3 :

إذا كان هناك واحد في المرتبة الإضافية فنقوم بحذف هذا الواحد

Exp1: Subtract the number $(10101)_2$ from $(1011)_2$ using the 2nd complement

Sol:-

$$\begin{array}{r}
 01011 \\
 10101 \text{ ---} \\
 01010 \\
 \hline
 1 \text{ +} \\
 01011 \\
 01011 \text{ +} \\
 \hline
 10110 \\
 01001 \\
 \hline
 1 \text{ +} \\
 01010
 \end{array}$$

إذا لم يظهر واحد في المرتبة الإضافية فنقوم بأخذ المتمم ال 2 لناتج الجمع ويكون هو ناتج الطرح (سالِب)

Week Five to Six

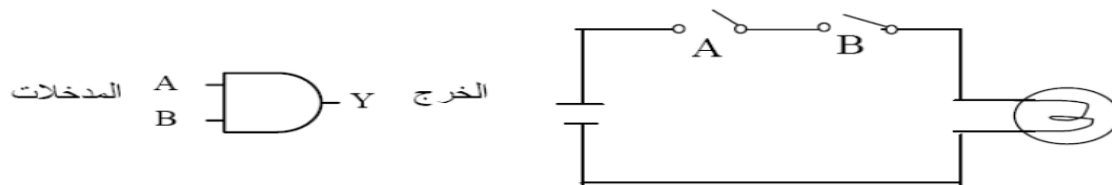
Logic Gates:

Logic gate is the basic unit for building the logic systems. Logic gates utilizes the binary numbering system which includes (0,1) bits to process the functions that why it called the binary logic gates. The logic gates use the high and low voltages which are represented by 1 and 0 respectively.

There are three fundamental logic gates which used to build the logic system. These gates called AND gate, OR gate, and NOT gate. However, they are the basic units to design and process multiple functions of the integrated circuits (IC).

1- AND Gate

The logic symbol and electrical circuit imply that the output of AND gate is high or 1 only if both inputs are at high level or 1. So, in order to maintain logic output ($y = 1$) or the lamp is on, both inputs (A and B) must be 1 or both switches (A and B) must be closed (1). otherwise, the output is 0 or the lamp is off

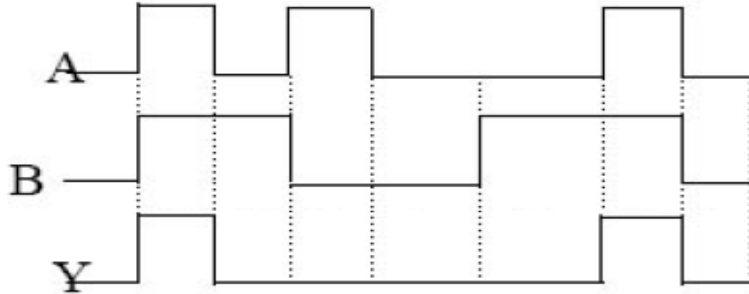


The possible cases of operation depend on the number of logic inputs. For example the logic AND gate of two inputs has four possible cases as shown in the truth table below.

الدخل		الخرج
B	A	Y
0	0	0
0	1	0
1	0	0
1	1	1

الدخل		الخرج
المفتاح		المصباح
B	A	Y
off	off	Off
off	on	Off
on	off	Off
on	on	On

In addition to the logic expression and the truth table .The inputs and output of AND gate can be expressed by pulses or waveforms to express the function of AND gate. The wave forms have on state and off state or 1 and 0 states as shown in the wave forms below



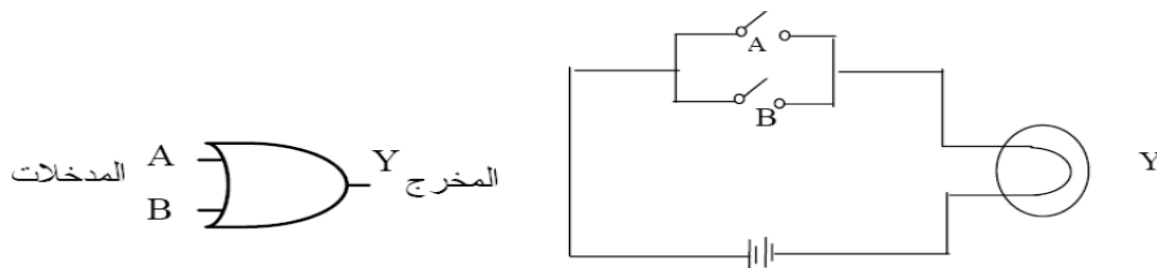
Boolean Algebra:

The Boolean algebra is the symbolic logic that describes how the logic gate works. The Boolean expression is abridging method to explain the process of the logic circuits.

The Boolean algebra for AND gate is $Y=A.B$ or $Y=AB$

2- OR Gate

The logic symbol and electrical circuit explain the function of OR gate. Where the output is at high level (1) when one input is high (1). So, in order to maintain logic output ($y = 1$) or the lamp is on, one input (A or B) must be 1 or one switch (A or B) must be closed (1). Hence, the output is 0 or the lamp is off when both (A&B) are (0).

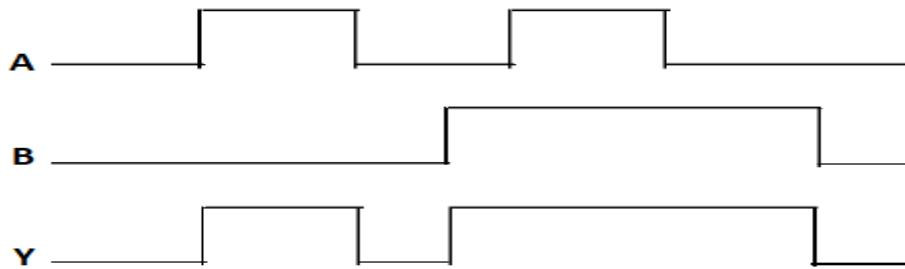


The number of possible cases depends on the number of logic inputs. For example the logic OR gate of two inputs has four possible cases as shown in the truth table below. The Boolean algebra for and gate is $Y=A+B$

الدخل		الخرج
B	A	Y
0	0	0
0	1	1
1	0	1
1	1	1

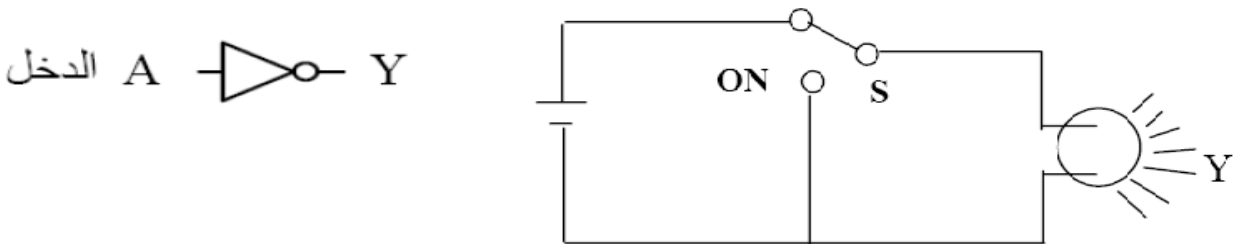
الدخل		الخرج
المفتاح		المصباح
B	A	Y
off	off	Off
off	on	On
on	off	On
on	on	On

We can draw the waveforms of OR gate based on the truth table and the logic expression as shown below



3- NOT Gate

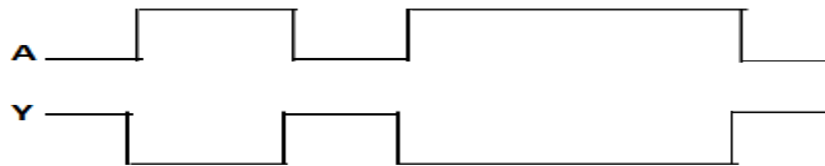
NOT logic gate works as inverter for the signal where the input signal is inverted to obtain reversed output signal. It has one input and one output. The electrical circuit shows the function of NOT gate where the switch invert the input signal so that the lamp is turn from on to off status and vice versa.



The truth table below shows the operation of NOT gate where the output signal is inverted version of the input one.

الدخل	الخرج	الدخل	الخرج
A	Y	المفتاح	المصباح
0	1	A	Y
1	0	Off	on
		on	off

Based on the Boolean expression of NOT gate ($Y=A^{\bar{}}$), the waveforms of this gate can be drawn below

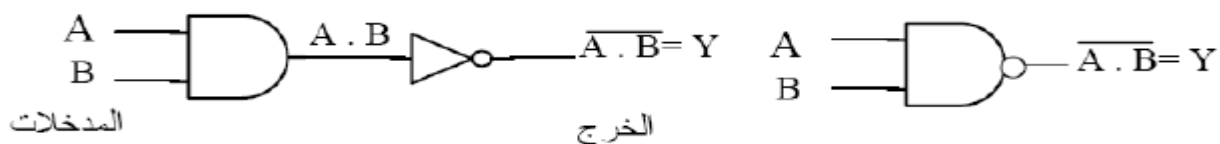


Combination of Logic Gates:

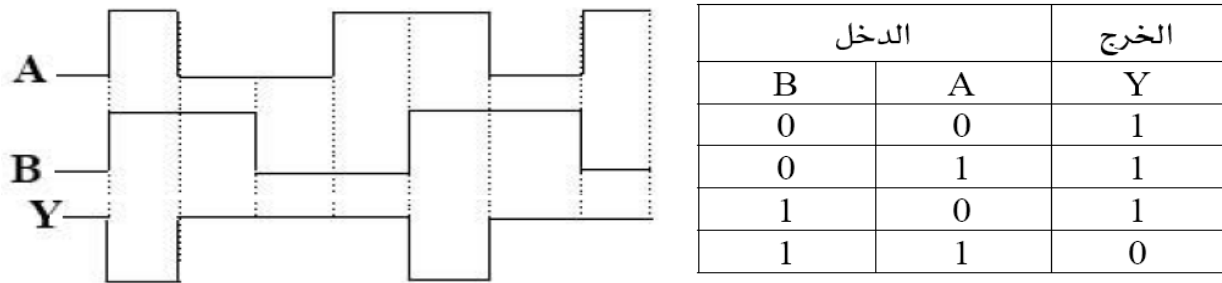
The former three basic gates are not always enough to execute some operations in the logic systems. Therefore, these fundamental gates can combine to form four new types of logic gates such as NOT- AND (NAND), NOT-OR (NOR), Exclusive OR (EXOR), and Exclusive NOR (EXNOR).

4- NAND Gate

The combination of AND-NOT gates forms the NAND gate. The figure below shows the output of AND gate ($A.B$) is multiplied by NOT gate to obtain the inverted ($\overline{A.B}$)

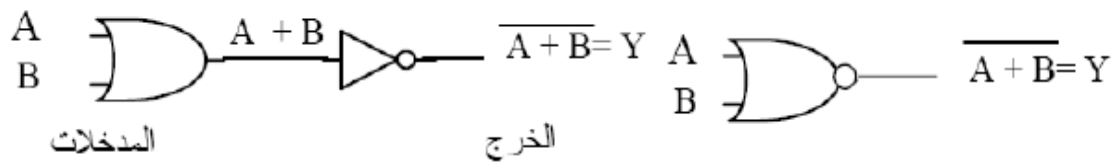


From the structure above we can rewrite the truth table of NAND gate which is the same of AND gate with only inverted output. Therefore, the Boolean expression is inverted by the line above the inputs to be $Y = \overline{A \cdot B}$ while the waveforms of this gate are shown below.

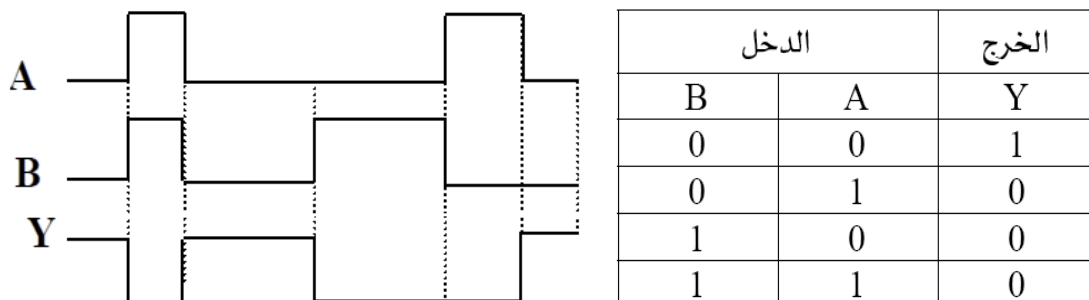


5- NOR Gate

The combination of OR and NOT gates forms NOR gate. The figure below shows the output of OR gate ($A+B$) is multiplied by NOT gate to obtain the inverted ($\overline{A+B}$)

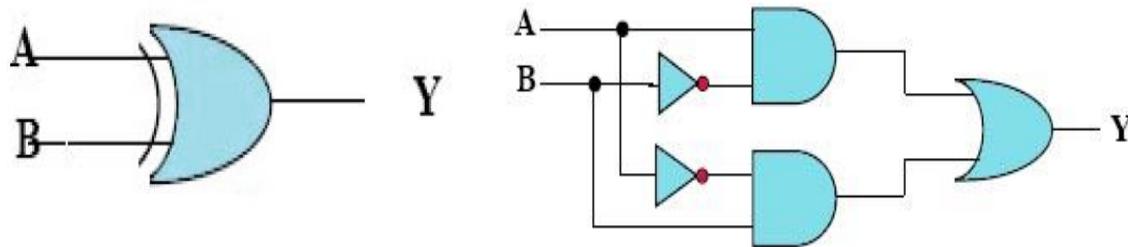


The truth table of NOR gate is the same as of OR gate with inverted output only. Therefore, the Boolean expression is inverted by the line above the inputs ($A \& B$). So the Boolean formula is $Y = \overline{A + B}$



6- XOR Gate

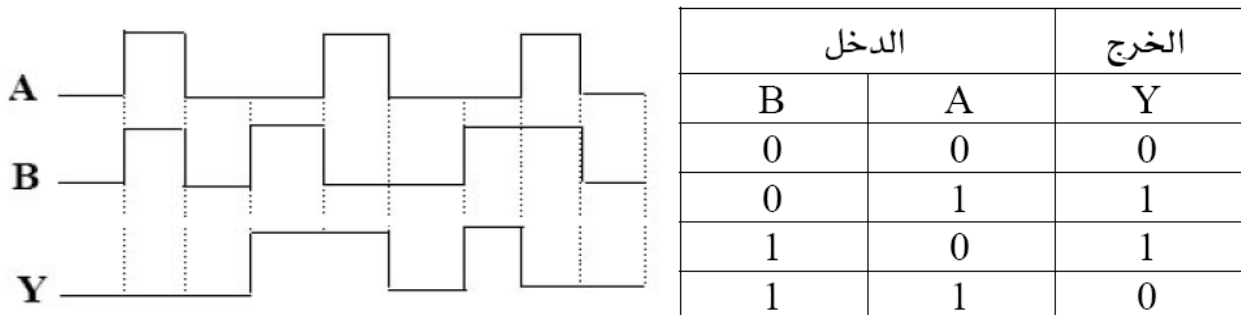
The exclusive OR gate (XOR) called the comparator logic gate too. The output of this gate is 1 only when the inputs are opposite to each other (0 and 1). Otherwise, the output is 0.



The truth table and the Boolean expression show the output is 1 only when the inputs (A&B) have different signal signs. So, based on the Boolean expression of

EXOR gate ($Y = A.B^{\bar{}} + A^{\bar{}}.B = A \oplus B$) and the truth table, we can draw the

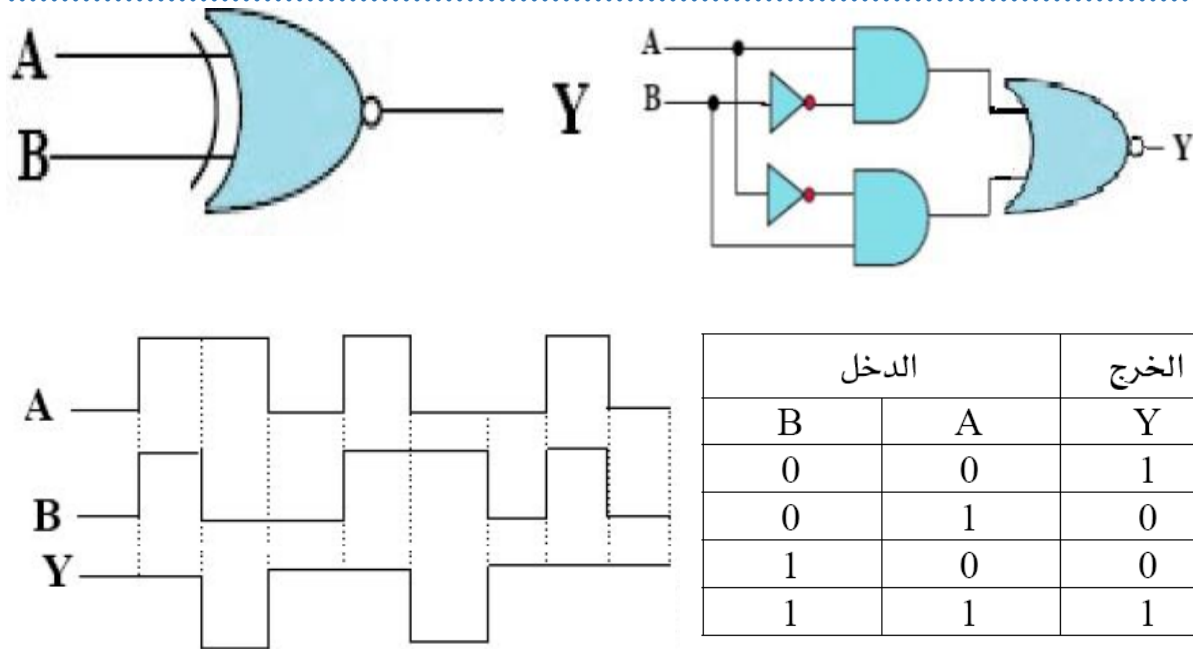
waveforms as below.



7- XNOR Gate

It is also called the exclusive EXOR gate (EXNOR) which is the EXOR gate plus the NOT gate (see the figure). The output of this gate is the opposite of the EXOR gate where the output is 1 only when the inputs (A&B) has the same sign either (0&0) or (1&1). Otherwise the output is 0.

The truth table is based on the Boolean expression as below $Y = \overline{A \oplus B}$



Week Seven to Ten

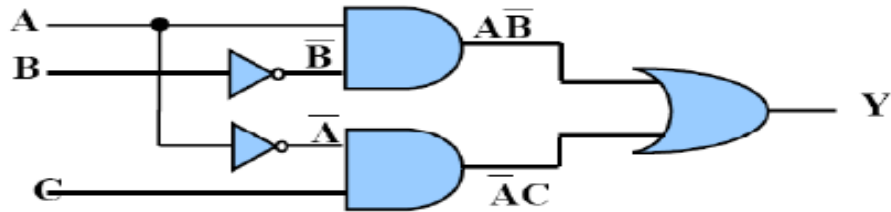
Boolean Algebra:

The Boolean rules dedicated to simplify the design of the logic circuits by simplifying the Boolean logic expression. This section will concentrate on writing the Boolean expression of the complex logic circuits, De Morgan theorem, Boolean rules, writing the truth table of the logic circuits, and writing the logic expressions using the product of sums and sum of products.

The Boolean Expression for a Logic Circuits;

Writing the logic expression for a complex logic circuit starts from the inputs of the logic gates towards the output as expressed by the example below.

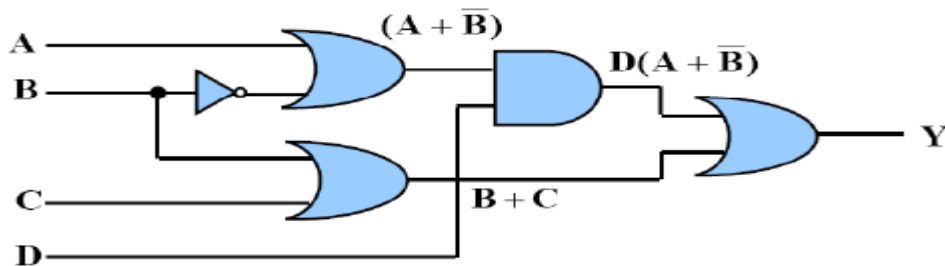
Exp:- write the logic expression for the circuit below



Sol:-

- 1- Write the logic expression of AND gate $A\bar{B}$
- 2- Write the logic expression of second AND gate $\bar{A}C$
- 3- Write the logic expression of OR gate $Y = A\bar{B} + \bar{A}C$

EXP: write the logic expression for the circuit showing below



Sol: -We write the Boolean expression for input logic gates towards the output gate

- 1- Write the logic expression of OR input gate $(A + \bar{B})$
- 2- Write the logic expression of OR input gate $(B + C)$
- 3- Write the logic expression of AND gate $D(A + \bar{B})$
- 4- Write the logic expression of OR output gate $Y = D(A + \bar{B}) + (B + C)$

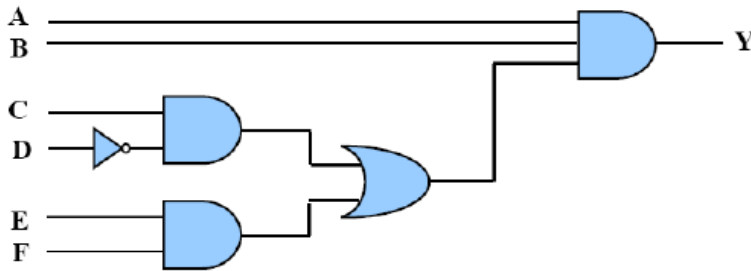
Implementation of a Logic Circuit Using Boolean Expression:

The Boolean expression can be used to implement a logic circuit where each part of the expression represents a logic gate. Therefore, the Boolean expression must be divided into parts of variables.

EXP: Draw the logic circuit for the expression $Y = AB(C\bar{D} + EF)$

Sol:- We must draw the input logic gates based on their variables (CD , EF, AB)

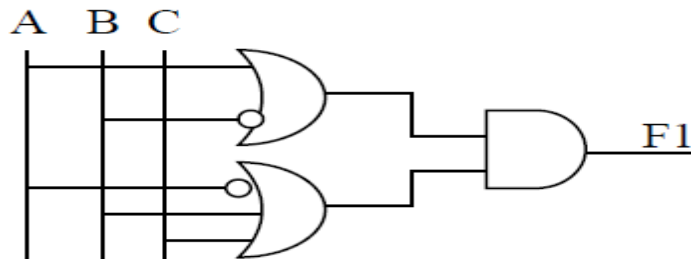
- 1- Draw NOT gate for \overline{D}
- 2- Draw AND gate for $C\overline{D}$
- 3- Draw AND gate for EF
- 4- Draw OR gate for $(C\overline{D} + EF)$
- 5- Draw AND gate for $Y = AB(C\overline{D} + EF)$



EXP: Draw the logic circuit for the expression $F1 = (A+B)(A+B+C)$

Sol:- We must draw the input logic gates based on their variables (A+B , A+B+C)

- 1- Draw OR gate for $(A+B)$
- 2- Draw OR gate for $(A+B+C)$
- 3- Draw AND gate for $(A+B)(A+B+C)$



Implementation of a Logic Circuit Via a Truth Table:

This section teaches us how to represent a logic circuit via a corresponding truth table instead of a Boolean expression. The truth table is firstly converted to Boolean expression, and then the expression is converted to a logic circuit as follows

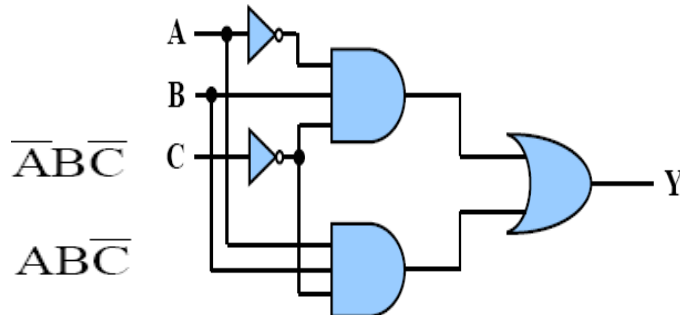
- 1- Point out the output whose value is (1) from the truth table

- 2- Point out the corresponding input variables of the output and its values(0 ,1)
- 3- Write the Boolean expression for outputs of (1)
- 4- Write the full Boolean expression and convert it to a logic circuit

EXP: Write the logic circuit for the following truth table

Sol:-

المدخلات			الخرج
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

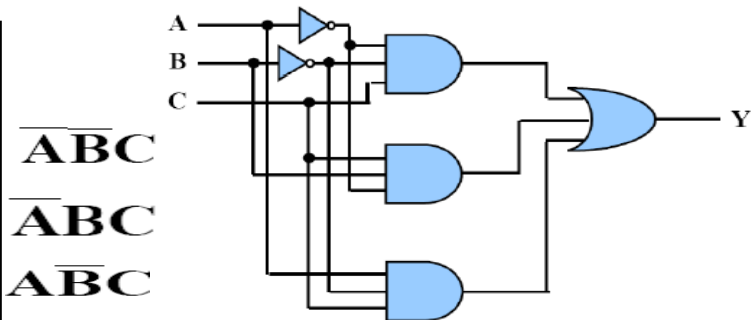


Then the Boolean expression is $Y = \overline{A}BC\overline{C} + ABC\overline{C}$

EXP: Write the logic circuit for the following truth table

Sol:-

المدخلات			الخرج
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Then the Boolean expression is $Y = \overline{A}BC + \overline{A}BC + A\overline{B}C$

Converting a Boolean Expression to a Truth Table:

The truth table has number of iterations for the inputs and its corresponding output. Writing a truth table relies on the number of input variables (A,B,C) and its values

(0 or 1).So, firstly the input variables must be filled in the table and then only the given input variables will have output of (1).otherwise $Y = \overline{A}B\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

EXP: Write the truth table of the following expression

Sol:-

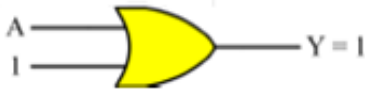

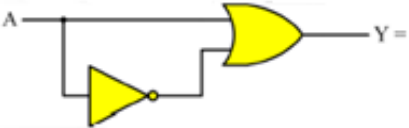
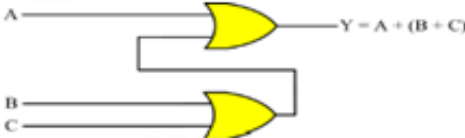
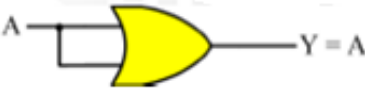
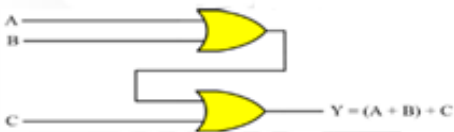
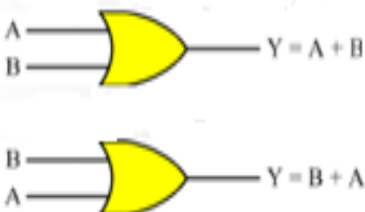
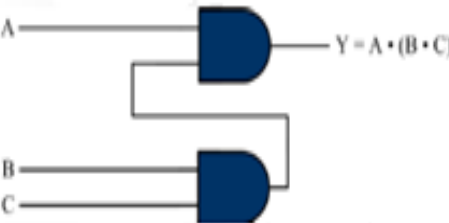
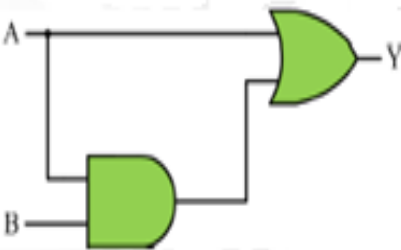
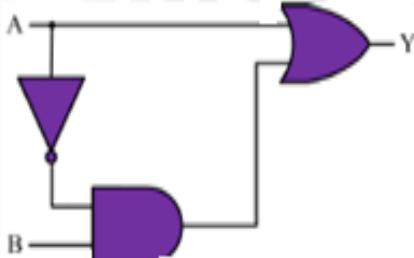
Fill in the table with input variables and then point out the given variables (A, B, C) and write 1 for its corresponding output (y).

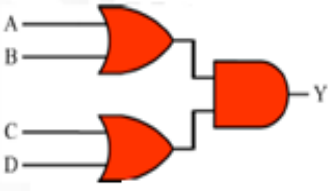
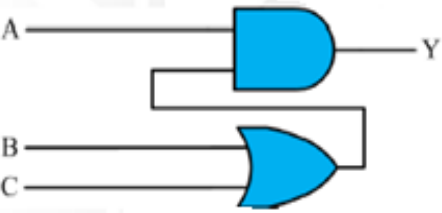
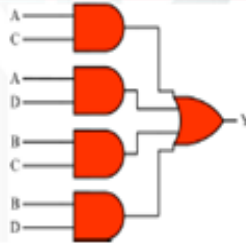
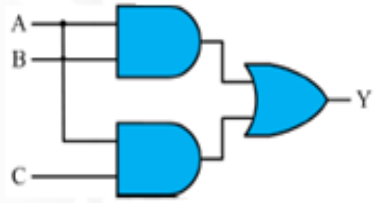
$$\overline{A}B\overline{C} = 000, \overline{A}B\overline{C} = 010, A\overline{B}\overline{C} = 110, ABC = 111$$

المدخلات			الخرج
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

قواعد الجبر البولياني: Rules of Boolean Algebra:

1	<p>T1: $A \cdot 0 = 0$</p>	4	<p>T4: $A \cdot \overline{A} = 0$</p>
2	<p>T2: $A \cdot 1 = A$</p>	5	<p>T3: $A \cdot A = A$</p>
3	<p>T4: $A \cdot \overline{A} = 0$</p>	6	<p>T5: $A + 0 = A$</p>

<p>7</p>	<p>T6: $A + 1 = 1$</p> 	<p>12</p>	<p>T10: $A \cdot B = B \cdot A$ Commutative law</p> 
<p>8</p>	<p>T8: $A + \bar{A} = 1$</p> 	<p>13</p>	<p>T11: $A + (B \cdot C) = (A + B) \cdot C = A + B + C$ Associative law</p> 
<p>9</p>	<p>T7: $A + A = A$</p> 	<p>14</p>	<p>T11: $A \cdot (B + C) = (A \cdot B) + C = A + B + C$ Associative law</p> 
<p>10</p>	<p>T9: $A + B = B + A$ Commutative law</p> 	<p>15</p>	<p>T12: $A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$ Associative law</p> 
<p>11</p>	<p>T14: $A + A \cdot B = A$</p> $A + A \cdot B = A \cdot (1 + B) = A \cdot (1) = A$ 	<p>16</p>	<p>T15: $A + \bar{A} \cdot B = A + B$</p> $A + \bar{A} \cdot B = (A + B) \cdot (A + \bar{A})$ 

17	<p>T13b: $(A+B) \cdot (C+D) = A \cdot C + A \cdot D + B \cdot C + B \cdot D$</p> 	19	<p>T13a: $A \cdot (B+C) = (A \cdot B) + A \cdot C$</p> <p>Distributive law</p> 
18	<p>T13b: $(A+B) \cdot (C+D) = A \cdot C + A \cdot D + B \cdot C + B \cdot D$</p> 	20	<p>T13a: $A \cdot (B+C) = (A \cdot B) + A \cdot C$</p> <p>Distributive law</p> 

However the basic rules of Boolean algebra can be tabulated as follows

<p>1. $A + 0 = A$</p> <p>3. $A \cdot 0 = 0$</p> <p>5. $A + A = A$</p> <p>7. $A \cdot A = A$</p> <p>9. $\overline{\overline{A}} = A$</p>	<p>2. $A + 1 = 1$</p> <p>4. $A \cdot 1 = A$</p> <p>6. $A + \overline{A} = 1$</p> <p>8. $A \cdot \overline{A} = 0$</p> <p>10. $A + AB = A$</p>
--	--

De Morgan's Theorems:

It is a significant part of Boolean algebra that simplifies the Boolean expression of two or more variables so that the logic circuits simply simplified.

. It converts the standard AND gate to OR gate and vice versa as shown below.

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

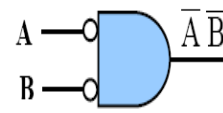
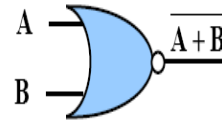
نظرية دي مورجان الأولى:

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

نظرية دي مورجان الثانية:

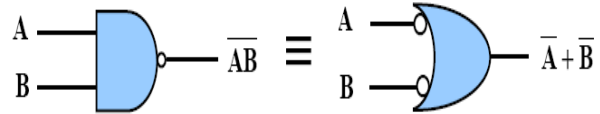
The first theorem converts the OR+NOT gates (NOR) to AND gate with NOT labels on its input variables as shown in the logical symbol and the table below.

المدخلات		الخرج	
A	B	$\overline{A+B}$	$\overline{A \cdot B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0



The second theorem converts the AND+NOT gate (NAND) to OR gate with NOT label on its input variables as shown in the logical symbol and the table below.

المدخلات		الخرج	
A	B	$\overline{A \cdot B}$	$\overline{A+B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



EXP: Apply DE Morgan's theorem to simplify the following expression

$$Y = \overline{(A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C})}$$

Sol:-

$$\begin{aligned} Y &= \overline{(A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C})} \\ &= \overline{(A + \overline{B} + \overline{C})} + \overline{(\overline{A} + B + \overline{C})} \\ &= \overline{A} \overline{\overline{B}} \overline{\overline{C}} + \overline{\overline{A}} \overline{B} \overline{\overline{C}} = \overline{A} B C + A \overline{B} \overline{C} \end{aligned}$$

Simplification of Boolean Expression Using Boolean algebra:

Boolean algebra rules are used to simplify the Boolean expressions. Hence the logic circuits are simplified as well.

Exp: simplify the Boolean expression using the Boolean rules $Y = AB + A(A+C) + B(A+C)$

Sol:- $Y = AB + AA + AC + AB + BC$

$$Y = AB + A + AC + AB + BC$$

(Rule 7) $A = AA$

$$AB + AB = AB \leftarrow \text{(Rule 5) } A + A = A$$

$$Y = AB + A + AC + BC$$

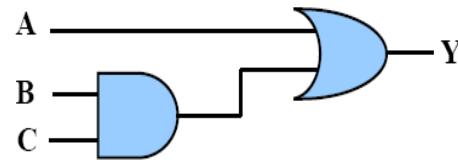
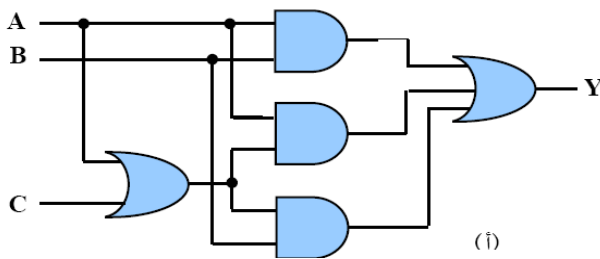
$$Y = A(B + 1 + C) + BC$$

(Rule 2) $A + 1 = 1$

$$Y = A(1) + BC$$

(Rule 4) $A(1) = A$

$$Y = A + BC$$



EXP:- Simplify the following Boolean expression and draw the logic circuit

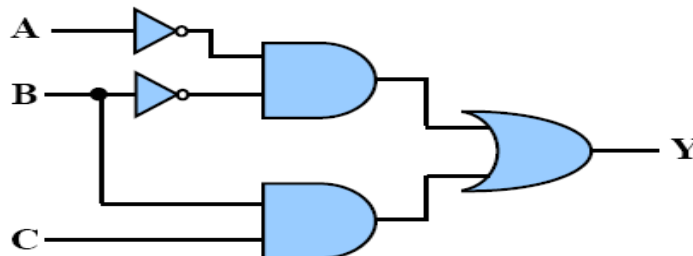
$$Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + ABC$$

Sol:-

$$\begin{aligned} Y &= (\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C) + (\overline{A}BC + ABC) \\ &= \overline{A}\overline{B}(\overline{C} + C) + BC(\overline{A} + A) \end{aligned}$$

$$Y = \overline{A}\overline{B} \cdot 1 + BC \cdot 1 \quad \text{(Rule 6)}$$

$$Y = \overline{A}\overline{B} + BC \quad \text{(Rule 4)}$$



EXP:- Simplify the following expression using Boolean algebra rules

$$\begin{aligned}
 y &= \overline{AB} + \overline{AC} + \overline{A}BC \\
 &= (\overline{AB})(\overline{AC}) + \overline{A}BC \\
 &= (\overline{A+B})(\overline{A+C}) + \overline{A}BC \\
 &= \overline{AA} + \overline{AC} + \overline{AB} + \overline{BC} + \overline{ABC} \\
 &= \overline{A} + \overline{AC} + \overline{AB} + \overline{BC} && \text{(Rule 7) } A\overline{A}=A\overline{A} \quad \text{(Rule 10) } A\overline{B} + A\overline{B}C = A\overline{B} \\
 &= \overline{A} + \overline{AB} + \overline{BC} && \text{(Rule 10) } A\overline{A} + A\overline{C} = A\overline{C} \\
 &= \overline{A} + \overline{BC} && \text{(Rule 10) } A\overline{A} + A\overline{B} = A\overline{B}
 \end{aligned}$$

EXP: Simplify the following expression using Boolean algebra rules.

Sol:-

$$\begin{aligned}
 y &= AB + A(B + C) + B(B + C) \\
 &= AB + AB + AC + BB + BC && \text{(Rule 7) } BB=B \quad \text{(Rule5) } AB + AB = AB \\
 &= AB + AC + B + BC \\
 &= AB + AC + B && \text{(Rule 10) } B+AB= B \\
 &= B + AC
 \end{aligned}$$

STANDARD FORMS OF BOOLEAN EXPRESSIONS

All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the **sum-of-products (SOP)** form or the **product-of sums (POS)** form. Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

The Sum-of-Products (SOP): مجموع الحدود المضروبة او جمع المضروبات

When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products (SOP).

مجموع المضروبات عبارة عن متغيرات مضروبه مع بعضها عن طريق بوابة And ويتم جمع هذه الحدود عن طريق بوابة OR للحصول على (SOP) وكمثال على ذلك

$$AB + ABC$$

$$ABC + CDE + BCD$$

$$AB + BCD + AC$$

The Product-of-Sums (POS) Form: ضرب الحدود المجموعة او ضرب المجموعات

A sum term was defined before as a term consisting of the sum (Boolean addition) of literals (variables or their complements). When two or more sum terms are multiplied, the resulting expression is a product-of-sums (POS).

عبارة عن متغيرات يتم جمعها عن طريق بوابة OR ضرب الحدود المجموعة والحدود المجموعة And للحصول على ضرب المجموعات او (POS). وكمثال على ذلك ويتم ضربها عن طريق بوابة

- 1- $(\bar{A} + B)(A + \bar{B} + C)$
- 2- $(A + B + C)(C + D + E)(B + C + D)$
- 3- $(A + \bar{B})(A + \bar{B} + C)(A + C)$

Converting Standard (SOP) to Standard (POS):

The binary values of any standard SOP form do not appear in the equivalent standard expression of POS. Furthermore, the binary values which are not represented in the SOP appear as POS in the equivalent standard expression. Consequently, the following procedure must be followed to convert SOP to POS.

- 1- Point out the binary numbers of the products in the SOP
- 2- point out the unavailable binary numbers
- 3- Write the binary numbers of 2 as POS
- 4- Follow the same procedure to convert POS to SOP

Exp1: convert the following standard SOP expression to POS

$$Y = A^-B^-C + A^-BC + AB^-C^- + ABC^- + ABC$$

Sol:- put 0 for A^-, B^-, C^- put 1 for A, B, C

So that

$$y = 001 + 011 + 100 + 110 + 111$$

So the unavailable binary numbers in SOP form are (000 , 010, 101)

So the POS form is

$$Y = (0 + 0 + 0)(0 + 1 + 0)(1 + 0 + 1)$$

$$\text{or } Y = (A + B + C)(A + B^- + C)(A^- + B + C^-)$$

Exp2: Convert the following standard SOP expression to POS

$$Y = A^-B^-C^- + A^-B^-C + AB^-C + ABC^-$$

Sol:- put(0) for A^- , B^- , C^- ; put(1) for A , B , C

So, that

$$y = 000 + 001 + 101 + 110$$

So the POS form is

$$Y = (0 + 1 + 0)(0 + 1 + 1)(1 + 0 + 0)(1 + 1 + 1)$$

$$\text{or } Y = (A + B^- + C)(A + B^- + C^-)(A^- + B + C)(A^- + B^- + C^-)$$

Converting Standard (POS) to Standard (SOP):

We follow the same procedure when converting POS to SOP. The only difference is putting 0 for A, B, and C and putting 1 for A^- , B^- , and C^- .

Exp1: Convert the following standard POS to SOP

$$Y = (A + B + C)(A + B + C^-)(A + B^- + C^-)(A^- + B + C^-)(A^- + B^- + C)$$

Sol:- put(0) for A , B , C ; put(1) for A^- , B^- , C^-

So, that

$$y = (0 + 0 + 0)(0 + 0 + 1)(0 + 1 + 1)(1 + 0 + 1)(1 + 1 + 0)$$

So the SOP form is

$$Y = (010) + (100) + (111)$$

$$\text{or } Y = (A^-BC^-) + (AB^-C^-) + (ABC)$$

Exp2:- Convert the following standard POS to SOP

$$Y = (A+B^-+C)(A+B^-+C^-)(A^-+B^-+C^-)(A^-+B^-+C^-)$$

Sol:-

put(0) for A, B, C; put(1) for A⁻, B⁻, C⁻

So, that

$$y = (0+1+0)(0+1+1)(1+0+1)(1+1+1)$$

So the SOP form is

$$Y = (000) + (001) + (100) + (110)$$

$$\text{or } Y = (A^-B^-C^-) + (A^-B^-C^-) + (AB^-C^-) + (AB^-C^-)$$

Karnaugh Maps خرائط كارنوف

Karnaugh map is a graphical representation of the logic system. It is an array of arranged number of cells which depend on the number of input variables. Karnaugh map can be drawn either from the truth table or from the standard POS or SOP expressions.

Construction of Karnaugh Map:

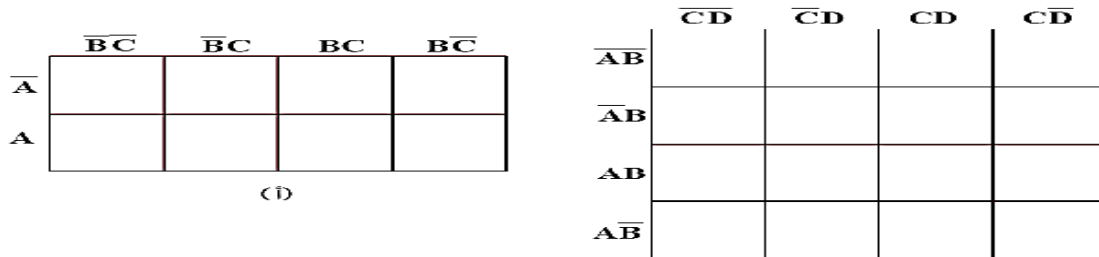
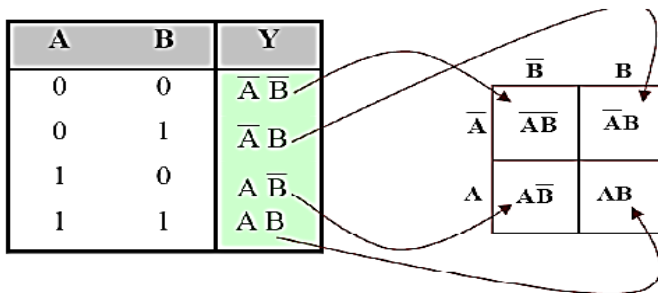
The number of cells or squares of karnaugh map relies on the input variables of the truth table or the standard expressions (POS and SOP).

Karnaugh Map for Two, Three, and Four variables;

The number of squares is 2^n , where n is the number of input variables of the truth

table or standard expression. So, Karnough map of two inputs variables is 2^2 which means it has 4 squares, while Karnough map of three inputs variables has 8 squares and Karnough map of four inputs variables has 16 squares etc.

The following figures show the Karnough Maps for two, three, and four variables. The truth table below has two variables (A, B) and their complements are (\bar{A} , \bar{B}). Each cell represent one of the four possible input of the truth table while the input labels are located outside the cells and its applied on both the columns and arrows of the cells. For example the variable \bar{A} is applied on the arrow of upper cells and the variable A is applied on the arrow of lower cells. While the variable \bar{B} is applied on the left column and the variable B is applied on the right column. So the right upper cell represented as $\bar{A}\bar{B}$



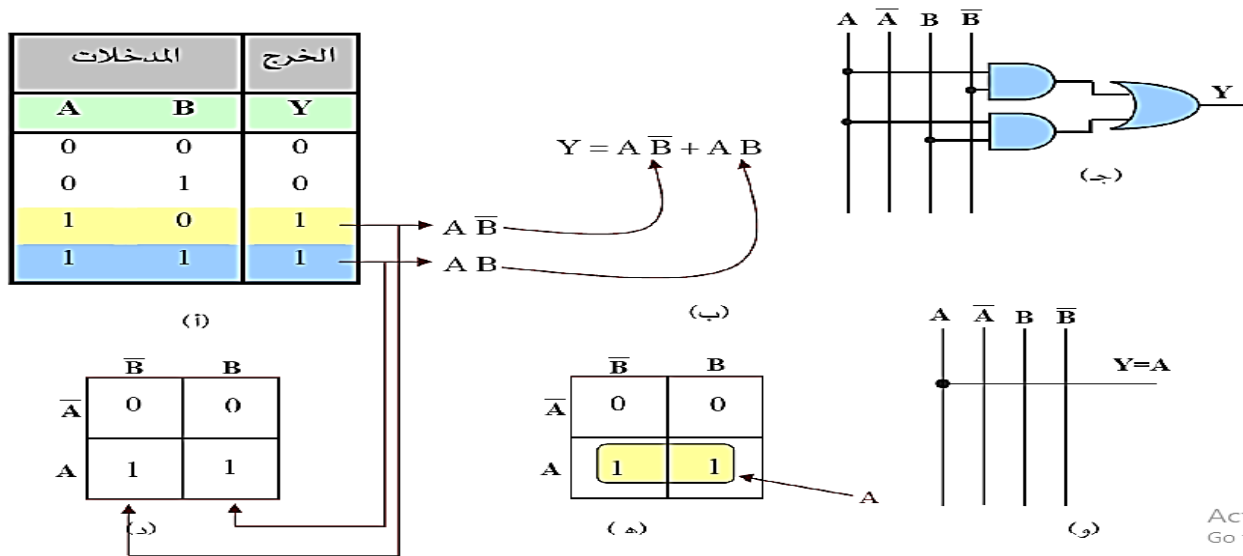
Karnough Map (SOP) Minimization:

The following example reveals how to use the map to simplify the Boolean expressions of SOP standard form.

Exp1:- use Karnaugh map to simplify the Boolean expression and draw the logic circuit.

Sol:-

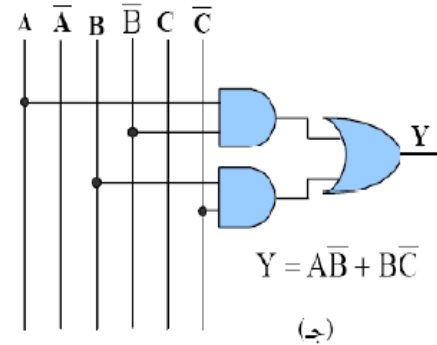
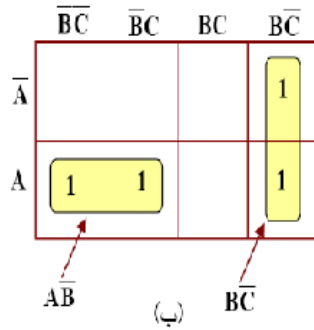
- 1- Write the SOP Boolean expression from the truth table
- 2- Write the Boolean expression on Karnaugh map of two variables
- 3- Write the output of 1 and 0 of the truth table in the corresponding cell of the karnaugh map.
- 4- The lower left cell ($AB^{\bar{}}$) and the lower right cell (AB) of the map will have (1) while ($A^{\bar{}}B$, $A^{\bar{}}B^{\bar{}}$) will have (0) based on the truth table.
- 5- Collect the adjacent cells of (1) as a group in the map
- 6- Point out the input labels for each group of cells
- 7- Delete the variable which has complement in the concluded expression such as ($B, B^{\bar{}}$) so $Y=AB^{\bar{}}+AB \rightarrow Y=A(B^{\bar{}}+B)=A(1)=A$



Exp2: use the truth table below to write the Boolean expression and use Karnaugh map to simplify the expression and draw the logic circuit

Sol:- $Y = A^{\bar{}}BC^{\bar{}} + AB^{\bar{}}C^{\bar{}} + AB^{\bar{}}C + ABC^{\bar{}}$

المدخلات			الخرج
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



تصميم الدائرة المنطقية باستخدام خريطة كارنوف.

EXP3:- Write the **SOP** Boolean expression from the truth table and use karnough map to simplify the expression and draw the logic circuit

المدخلات				الخرج
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Sol:- 1-write the boolean expression from the truth table

$$Y = \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + ABCD$$

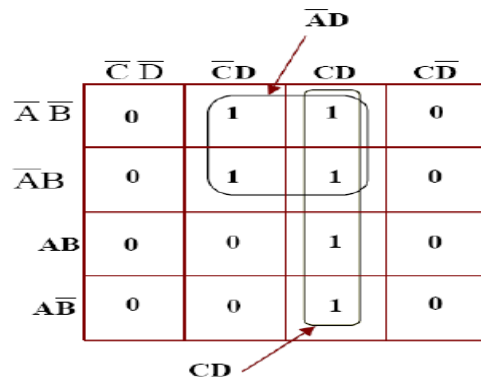
2- draw karnough map of four variables

3-collect the adjacent cells to groups of cells

4-delete the variables of complements

Such as B,B- C,C- , A,A-

5-write the output as $Y = A\overline{D} + CD$



Karnough Map (POS) Minimization:

The same procedure will be used to simplify the Boolean expression by Karnough map. The only difference between SOP and POS simplification is that we write the output of the truth table of 0 instead of 1 also we put 0 in the corresponding cells in the map.

Exp:- Use the truth table to write the POS expression and use karnough map to simplify it.

Sol:-1- write the boolean POS expression from the truth table

$$Y = (A + B + C + D)(A + B + C + \bar{D})(\bar{A} + B + C + D)(A + \bar{B} + C + \bar{D}) \\ (A + \bar{B} + \bar{C} + \bar{D})(\bar{A} + B + C + D)(\bar{A} + \bar{B} + C + D)$$

2- draw karnough map of four variables

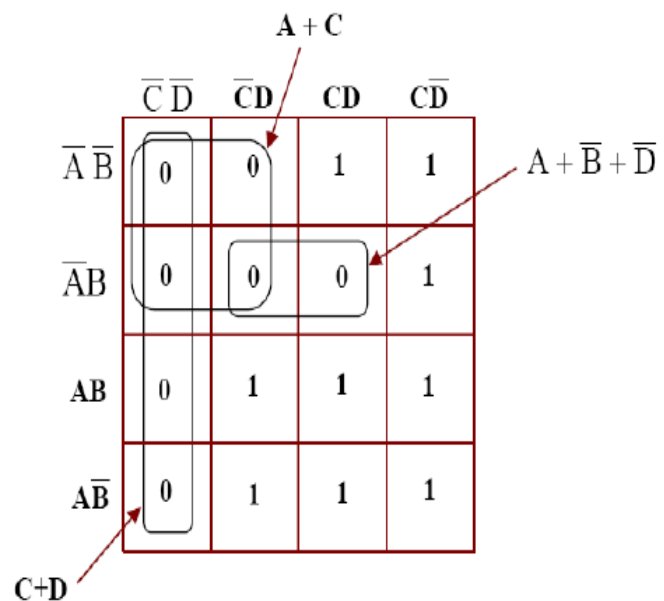
3- put 0 for the corresponding cells in the map

4- Collect the adjacent cells to groups

5- Delete the variables of complements such as D-,D ,B-,B ,C-,C , A-,A

6- Write the POS output of all the groups to be $Y=(A+C)(C+D)(A+\bar{B}+\bar{D})$

المدخلات				الخرج
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



References:

1-Digital Principles &Application

2-Digital computer fundamentals (thematic bartee))

3-Introduction to Digital computer ((louis mashelsky))

4-Modern Digital electronics (R.P.Jain)

5-((مالفينو :تأليف)) وتطبيقاته الرقمي الالكترونيك-5.