21CC1F2, Applied Physics, UNIT-V Digital electronics _ Notes

Syllabus

Module 5 [12 hours]

Digital Electronics: Digital and Analog Signals and Systems, Binary Digits, Logic Levels, and Digital Waveforms, Logic Gates: Logical Operators, Logic Gates-Basic Gates (OR, AND, NOT), Other gates (NOR gates and NAND gates), Universal Gates and realization of other gates using universal gates, Half adder and full adder, Boolean Algebra: Rules and laws of Boolean algebra, De-Morgan's Theorems, Numerical, Simulation by verification of truth table of Logic gates.

5.1 Introduction to Digital electronics

The word digital is originated from the way operations are executed by counting digits. In the early days, digital electronics were primarily applied to computer systems only. But now digital electronics has a great impact on almost all technology which are being used in our daily lives. For example, radio, television, automotive electronics, navigation systems, radar, medical instrumentation, embedded systems, military equipment, computers, cell phone and other communication systems. As the size of the devices used in the above applications are reducing and the complexity of the technology is increasing, it is mandatory to learn the basics of digital electronics. This would enable us to become up to date about modern technologies.

5.2 Digital and Analog Signals and Systems

Electronic circuits can be classified into two categories based on their mode of handling the signal. They are namely digital circuits and analog circuits. Digital electronic circuits use quantities having discrete values and analog electronic circuits use quantities having continuous values. Most of the measurable quantities in nature are having continuous values and can be considered analog. Temperature, time, pressure, distance and sound are examples of analog quantities. If we measure the atmospheric temperature, it may start from 25^o C in the early morning and reaches 33^o C around 2 PM, then again starts to decrease during the evening. This variation is continuous and can be considered as analog as shown in the following figure (left).

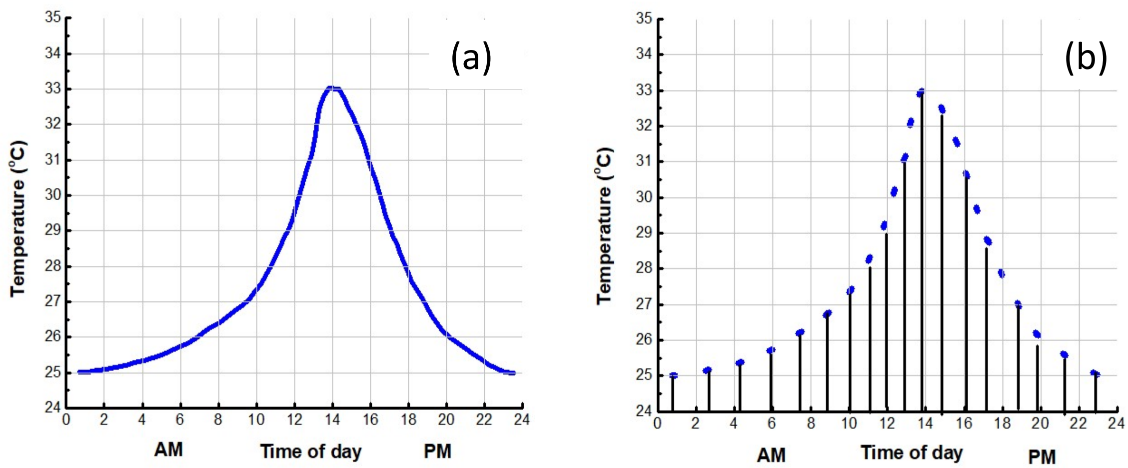


Figure 1:(a) Temperature variation representing the analog quantity. (b) sample valued representation of the same analog quantity. Each dot can be digitized.

Now the same temperature variation can be represented discretely by measuring the temperature at some random timings as shown in the right image. Here it is possible to digitize the values indicated by the dots by converting them as a digital code composed of 1s and 0s. Digital representation has many advantages over analog representation. The digital data can be implemented and sent more effectively and reliably. Also, the data can be saved more orderly and it can be reused with good clarity and precision. The effect of noise on the digital signal is very less compared to the analog representation. Therefore it is possible to make smaller integrated circuits easily, having much more complex functions and utilities.

The amplification of sound during a public gathering system uses analog electronics. Here the amplified sound is audible for a large audience. The analog sound waves are captured by a microphone and it is transferred to a small analog voltage. This is known as the audio signal. This audio signal is given to a linear amplifier and the amplified signal is fed to a speaker which will convert the amplified audio signal into the sound wave. A compact disc (CD) player is a system that utilizes both analog and digital electronic circuits. The stored information in the CD is read by a laser diode-based optical system and it is given to a digital-to-analog converter (DAC). The DAC convert the digital signal into the analog signal.

5.3 Binary Digits

In digital electronics, circuits and systems are working based on two states that are indicated by two voltage levels. One of the voltage levels is HIGH and the other level is LOW. In some applications, the two states are indicated by two current levels or bits and bumps on a CD/DVD etc. In digital systems, information (numbers, symbols, alphabetic characters etc.) is stored in the form of two numbers 1 and 0. It is called a binary number system. The two digits in the binary system, 1 and 0, is called a ***bit*** which is the abbreviation of binary digit. The combination of 1s and 0s are called ***codes***. In the digital circuits, 1 is indicative of HIGH voltage and 0 is indicative of LOW voltage. The usage of HGH=1 and LOW=0 is known as ***positive logic***.

Binary addition

To understand any digital system, it is better to learn binary arithmetics. The binary arithmetics are binary addition, binary subtraction, binary multiplication and binary division. The basic rules for adding binary numbers are given below.

- 0 + 0 = 0   Sum is 0 and carry is 0
- 0 + 1 = 1   Sum is 1 and carry is 0
- 1 + 0 = 1   Sum is 1 and carry is 0
- 1 + 1 = 10   Sum is 0 and carry is 1

Example:

		1	1	1	1		← carry
		1	1	1	0	1	
(+)		1	1	0	1	1	
	1	1	1	0	0	0	

The above example can be understood through the following steps.

- 1 + 1 = 10 = 0   Here carry is 1.
- 1+0+1 = 10 = 0   Here carry is 1
- 1+1+0 = 10+0 = 10 = 0   Here carry is 1
- 1+1+1= 10+1 = 11= 1   Here carry is 1
- 1 +1 +1 = 11
- The final sum is 111000

Binary subtraction

The following are the rules for subtracting binary numbers.

- 0 - 0 = 0
- 1 - 1 = 0
- 1 – 0 = 1
- 10 – 1 = 1   (10-1 has to be read as 0-1 with a borrow of 1)

Example:

		0	10		← borrow
	1	~~1~~	~~0~~	0	
(-)	1	0	1	0	
	0	0	1	0	

The above example can be understood as follows.

- $0 - 0 = 0$
- Now in the case of $0 - 1 = 1$, we need to borrow 1. Then it becomes $10 - 1 = 1$
- For $1 - 0$, since 1 has already been given, it becomes $0 - 0 = 0$
- $1 - 1 = 0$
- Therefore, the result is 0010

Binary Multiplication

The following are the rules for multiplying binary numbers.

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

Binary multiplication is similar to that decimal multiplication. In the first step, we need to form the partial products, and then move all successive partial products left one place followed by adding all the partial products. This is shown below.

Example:

```
x    100
     011
    -----
     100
    100   +
    -----
    1100
```

Binary Division

The binary division is similar to that decimal division. An example is shown below.

The following are the rules for multiplying binary numbers.

- $0 \div 1 = 0$
- $1 \div 1 = 1$
- $0 \div 0 =$ not allowed
- $1 \div 0 =$ not allowed

```
        101
     ______
101 ) 11010
     -101
     ----
        11
      - 00
      -----
       110
      -101
      -----
         1
```

5.4 Logic Levels

The voltages used to indicate a 1 and a 0 are known as logic levels. In a practical digital electronic circuit, a HIGH voltage can be taken as any voltage value between a particular minimum value and a particular maximum value. In a similar manner, a LOW voltage value can be taken as any voltage between a particular minimum and a maximum value. This is illustrated in the below figure. A voltage between V1 and V2 is not acceptable for representing 0 or 1.

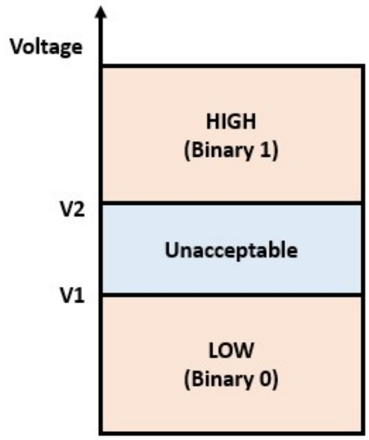
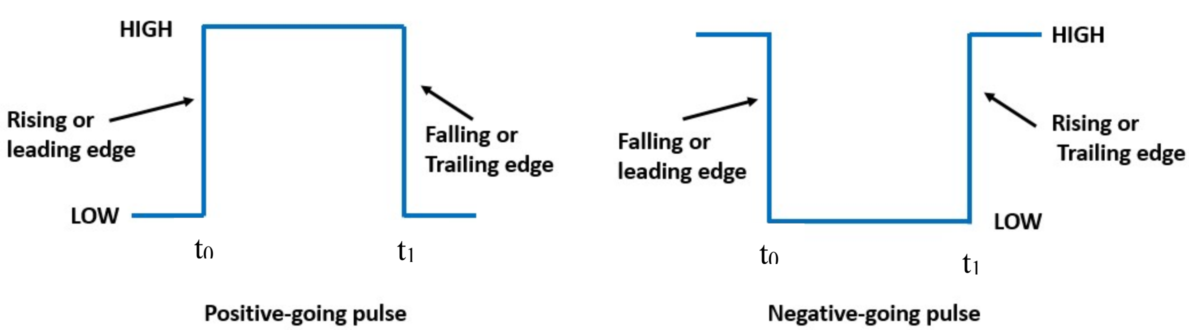


Figure 2: Logic level ranges of voltage for a digital

5.5 Digital Waveforms

When the HIGH and LOW-level voltages are changing back and forth, a digital waveform is created. It can be of two types: 1) positive-going pulse and 2) Negative-going pulse. In the first case, the voltage starts from the LOW level and reaches its HIGH level and comes back to its LOW level. But in the case of a negative-going pulse, the voltage starts from the HIGH level and reaches the LOW level and again goes back to the HIGH level as shown in the following figure.



pulses

A pulse will have two edges. Initially, the leading edge will occur at time t_0 and the trailing edge will occur at time t_1. For a positive-going pulse, the leading edge is a rising edge and the trailing edge is a falling edge. But in the case of a negative-going pulse, it is in the opposite way. In a periodic digital signal, the pulses are repeated at a fixed interval called the ***period (T)***. The rate at which the pulses is repeating is referred to as ***frequency (f)*** of the waveform. The relation between frequency and period can be written as

$$f = \frac{1}{T}$$

A periodic signal with period T is shown in the following figure.

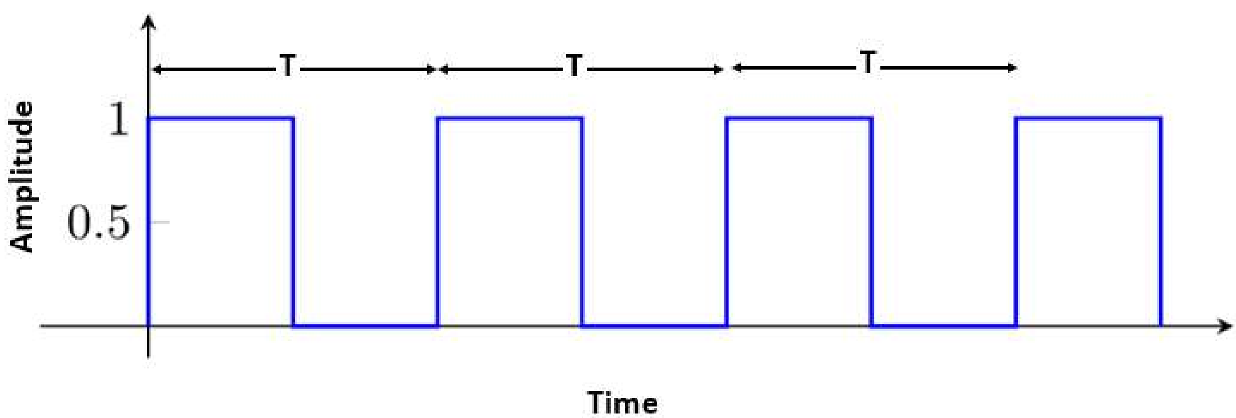


Figure 4: A periodic signal with period

Binary information in the digital systems is handled as waveforms which represent a sequence of bits. When the waveform is having a HIGH voltage, binary 1 is spotted and when the waveform is having a LOW voltage, binary 0 is spotted**.**

5.7 Boolean Algebra

Digital circuits operate based on the binary number system (0 and 1). These circuits use Boolean algebra as a mathematical tool for analysis and design. Boolean algebra is a mathematical framework upon which logic design is based. It is very different from conventional arithmetics and it is based on the logical statements that can be referred to as either true (1) or false (0). In Boolean algebra, the variables and constants are allowed to have only two values- either 1 or 0.

5.8 Logic functions and logic gates

Two types of operations are carried out on binary data. They are arithmetic and logic operations. Addition, subtraction, multiplication and division are the basic arithmetic operations. NOT, AND and OR are the basic logic functions.

Logic gates:

For the manipulation of binary variables and simplifying the logic expressions, Boolean algebra is utilized. Binary data is incorporated in digital devices with the help of certain electronic circuits. They are called logic gates. A logic gate is an electronic circuit that performs a particular logic function. The basic logic gates are NOT, AND and OR which are used for performing the NOT, AND and OR logic functions respectively. For any digital system including computers, the logic gate is the most fundamental building block. They manipulate the input binary signal and gives a particular output signal which is either 1 or 0. Logic gates have only one output and they can have more than one input except in the case of NOT gate. Each logic gates have its symbol as shown in the following figure.

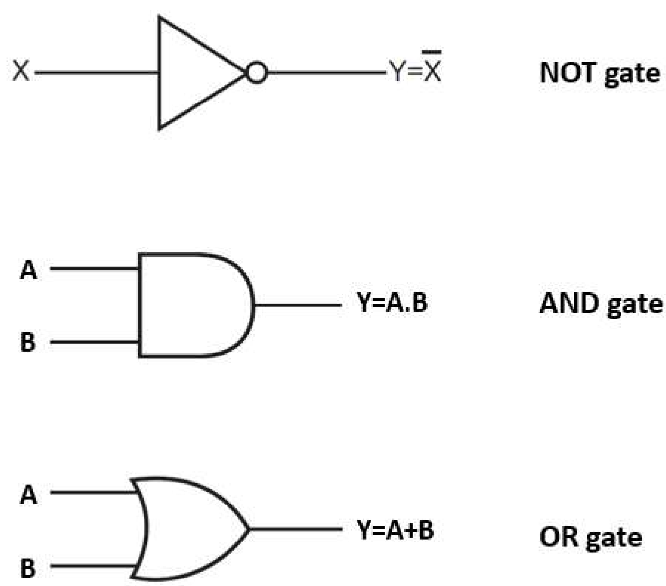


Figure 5: Symbols for NOT, AND and OR gates. Only two inputs are shown for AND and OR gates for simplicity.

Here A, B and X are the logic input (Either 1 or 0). Y is the logic output which is expressed in the form of a Boolean expression. The Boolean expression would connect the output and the input of a digital circuit.

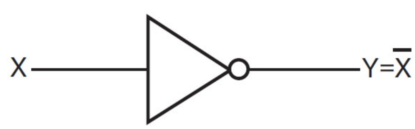
Truth table:

Usually, the input-output relationship of the binary variable for a logic gate is tabulated and is referred to as a truth table. A truth table for a logic gate will show all the possible combinations of input binary variables and the respective outputs. If the number of input binary variables is only one, then the input is either '0' or '1'. Now if the number of inputs is two, the number of possible input combinations are four, i.e. 00, 01, 10 and 11. In a similar manner, if there are three input variables, there can be eight input combinations, i.e. 000, 001, 010, 011, 100, 101, 110 and 111.

NOT gate

The NOT gate converts the input logic level to the opposite logic level. For example, if the input signal is HIGH (1), then the NOT gate output will be LOW (0). Similarly, if the input is LOW(0), then the output will be HIGH (1). In the case of NOT gate, the output is just opposite to the input signal as shown in the following truth table. Or in other words, the output signal is always the complement of the input signal. If the input variable is represented as X, then the logic output is represented as $\overline{X}$. This implies that if X is the input value that we are giving to a NOT gate, then its output value Y is given by

$$Y = \bar{X}$$



X	Y
0	1
1	0

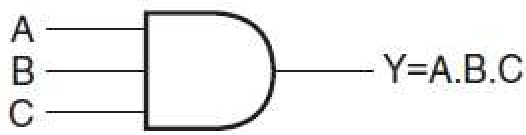
Figure 6: NOT gate symbol and the truth

AND gate

An AND gate is a logic circuit which will have two or more input signals and only one output. The AND operation between two logic variables A and B can be written as

$$Y = A.B$$

It should be reads as Y equals A AND B. The output of an AND gate will be HIGH (1) only when all of its inputs are in the HIGH (1) state. For all other input combinations, the output logic will be LOW. This shows that the output of the AND gate is '1' only when all of its inputs are '1'. In all other cases, the output is '0'. The following figures shows the logic symbol and truth table of a three-input AND gate.



A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figure 7: The logic symbol and truth table of a three-input AND

For a three-point AND gate, the Boolean expression is $Y = A.B.C.$

For a four-point AND gate, it can be written as $Y = A.B.C.D$

OR gate

The OR operation between two logic variables (let it be A and B) is expressed as

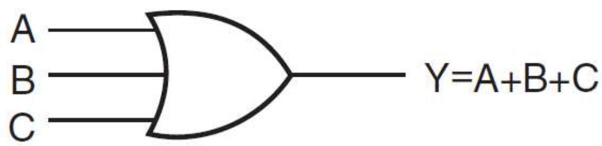
$$Y = A+B$$

It should be reads as Y is equal to A OR B and not as A plus B. As in the case of the AND gate, an OR gate is also a logic circuit which can take two or more input values and will give only one output value. The output of an OR gate is 0 only when all of its inputs are 0. For all other input combinations, the output will be 1.

For a three-point OR gate, the Boolean expression can be written as $Y = A + B + C.$

For a four-point OR gate, the expressin modifies to $Y = A + B + C + D$

The circuit symbol and the truth table of a three-input OR gate is shown in the following figure.



A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

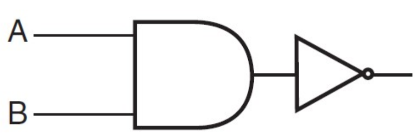
Figure 8: The logic symbol and truth table of a three-input OR gate.

5.9 Other logic gates – NAND, NOR, EXCLUSIVE OR and EXCLUSIVE - NOR gates

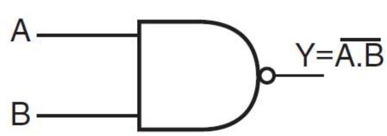
The NOT, AND and the OR gates are considered to be the basic logic gates. They can be used for constructing other gates such as the NAND gate, NOR gate, the EXCLUSIVE OR gate and the EXCLUSIVE-NOR gate.

NAND gate

NAND gate is derived from the AND and the NOT gate. It is the short form for NOT AND. A NAND gate can be constructed by feeding the output of a AND gate to the input of a NOT gate. The circuit construction, gate symbol and a two-input truth table for the NAND gate are shown in the following figure.



(a)



(b)

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(c)

Figure 9: (a) The circuit construction, (b) gate symbol and (c) a two-input truth table for the NAND gate

The NAND gate truth table can be constructed by complementing the output of an AND gate. When all the inputs are '1', the output of a NAND gate is '0'. For all other combination of input signals, the output will be '1'. In Boolean algebra, the expression for the logical NAND gate operation is

$$Y = \overline{A.B}$$

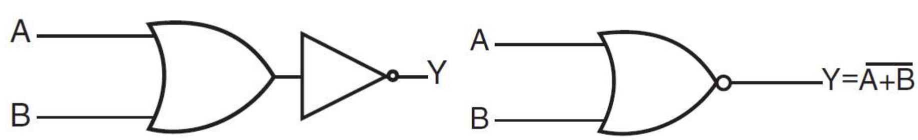
For a three-input NAND gate, the expression will become as follows

$$Y = \overline{A.B.C}$$

Since all the basic logic gates (NOT, AND, OR) can be constructed by using the NAND gates alone, it is also called the universal gate. This implies that the combination of NAND gates can be used to obtain all other basic logic gate operations such as NOT, AND and OR.

<u>NOR gate</u>

NOR gate is derived from the OR and the NOT gate. It is the short form for NOT OR. The NOR gate can be constructed by feeding the output of an OR gate to the input of a NOT gate. The circuit construction, gate symbol and a two-input truth table for the NOR gate are shown in the following figure.



(a) **(a)**

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(c)

Figure 10: (a) The circuit construction, (b) gate symbol and (c) a two-input truth table for

The NOR gate truth table can be constructed by complementing the output of an OR gate. If all the inputs are '0' for the NOR gate, then the corresponding output will be 1. For all other possible input combinations, the output will be '0'. In Boolean algebra, the expression for the logical NOR gate operation is

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

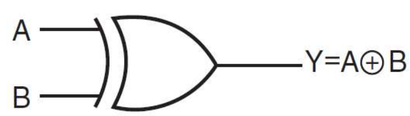
For a three-input NOR gate, the expression will become as follows

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

As in the case of the NAND gate, the NOR gate can also be utilized as a universal gate. This implies that the combination of NOR gates can be used to obtain all other basic logic gate operations such as NOT, AND and OR.

EXCLUSIVE OR gate

The EXCLUSIVE-OR gate is usually abbreviated as the EX-OR gate. The output value of an EX-OR gate will be '1' when the input values are unlike and it will be '0' when the input values are like. This implies that the output value will be '1' for the input combinations A=0, B=1 and A=1, B=0. The gate symbol and a two-input truth table for the EX-OR gate are shown in the following figure.



(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(b)

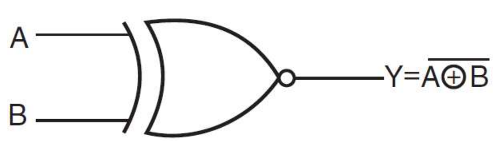
Figure 11: (a) gate symbol and (b) a two-input truth table for the NOR

In Boolean algebra, the expression for the logical EX-OR gate operation is

$$Y = A \oplus B = \bar{A}.B + A.\bar{B}$$

EXCLUSIVE-NOR gate

EXCLUSIVE-NOR is written in short as EX-NOR. It is the NOT of EX-OR. This is achieved by giving the output of an EX-OR gate to the input of a NOT gate. This means that the EX-NOR logic gate can be realized by complementing the output values of an EX-OR gate. The EX-NOR gate truth table can be constructed from the EX-OR gate truth table by complementing the output values. The gate symbol and a two-input truth table for the EX-OR gate are shown in the following figure.



(a)

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(b)

Figure 12: (a) gate symbol and (b) a two-input truth table for the EX-NOR

In Boolean algebra, the expression for the logical EX-NOR gate operation is

$$Y = \overline{A \oplus B} = A.B + \bar{A}.\bar{B}$$

5.9 NAND and NOR gate as universal gates-Realization of other gates

Any logic circuit that are representing a Boolean expression can be realized by utilizing the basic logic gates such as OR, AND and NOT. We can also realize any logic circuit by using simply NAND or NOR gates. That is, any Boolean expression can be realized by using either NAND gates alone or using NOR gates alone. This can be achieved by combining more than one NAND gate or NOR gate. Due to this reason, NAND and NOR gates are referred to as universal gates. Apart from constructing the basic gates, a combination of NAND gates can be used for constructing NOR gate and a combination of NOR can be used for constructing NAND gate also. The following figure shows how we can utilize two-input NAND gates for realizing other gates such as NOT gate, AND gate and OR gate.

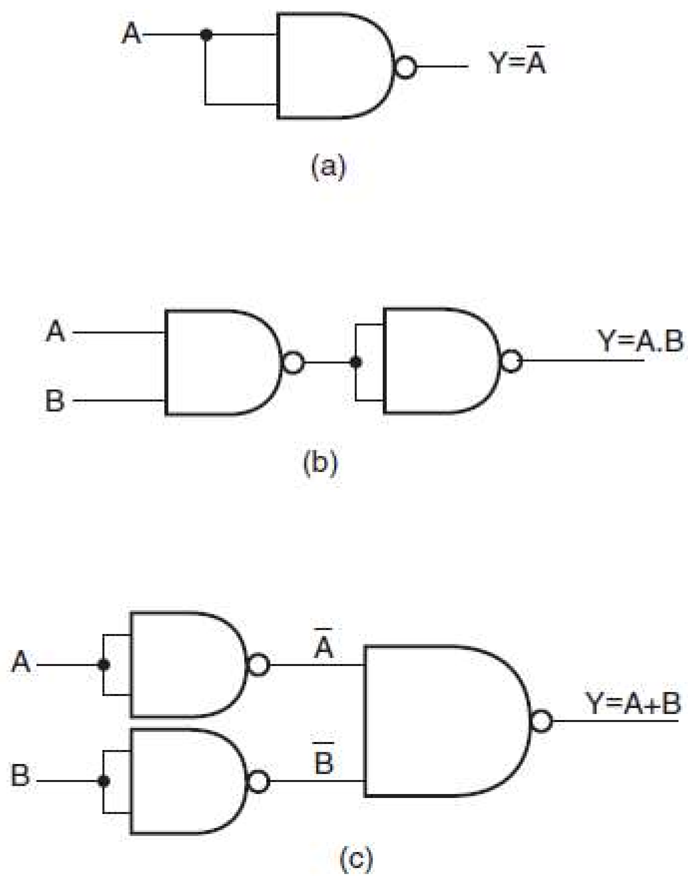


Figure 13: Construction of (a) NOT gate (b) AND gate (c) OR gate using NAND gates

Similarly, the basic logic gates can be constructed by using NOR gates alone. This is illustrated in the following figure.

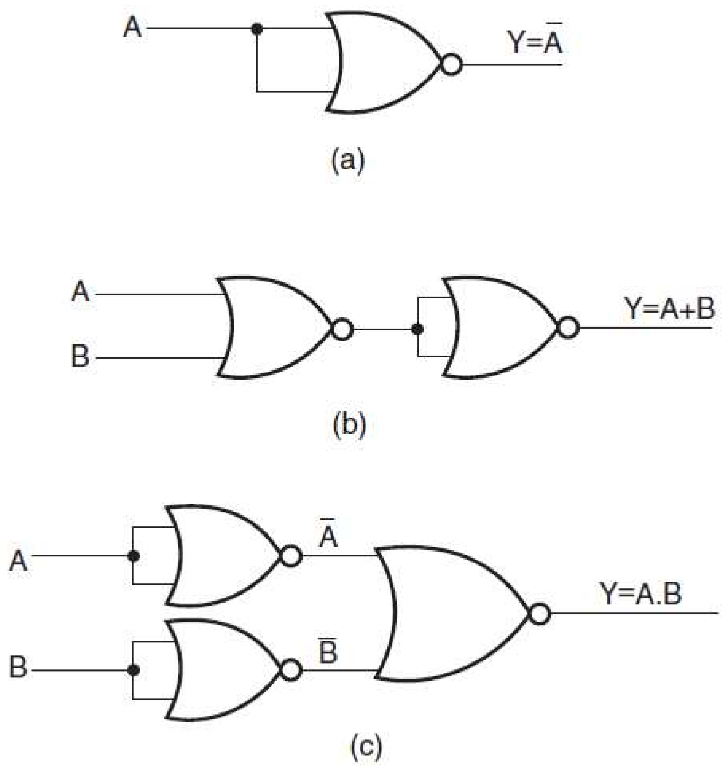


Figure 14: Construction of (a) NOT gate (b) AND gate (c) OR gate using NOR gates alone.

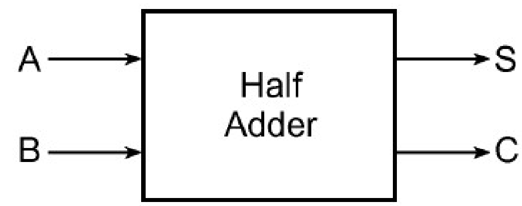
5.10 Half adder and full adder

When logic gates (such as NOT, AND, OR, NAND and NOR) are connected in a specified combination of input variables to carry out a particular logic function, the resulting circuit is referred to as a combinational circuit. The output value of a combinational circuit at a particular time depends only on the combination of inputs at that point of time. It will not be depending on the earlier state of the inputs. Also, there is no data storage involved in a combinational circuit. Computers and other digital systems handle and process a huge amount of binary data. The most commonly used logic operations are binary addition and binary subtraction.

Half-adder: A half-adder is an arithmetic circuit block that can be utilized for adding two binary digits (bits). A half-adder circuit will have two input values (A and B) and two output values (SUM and CARRY). The input values A and B represent the two bits that are to be added. Among the two output values, one is referred to as the SUM (S) and the other is referred to as the CARRY (C). The following figure shows the truth table containing all the possible input combinations and the respective SUM and CARRY values. The logic symbol for the half-adder is shown in figure 15(b).

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a)



(b)

Figure 15: (a) Truth table and (b) Logic symbol for half-

The Boolean expression for the SUM (S) and CARRY (C) for a half-adder is expressed as

$$S = \bar{A}.B + A.\bar{B}$$

$$C = A.B$$

From these expressions, it is clear that the Boolean expression for the SUM is similar to that of the expression for EX-OR gate and the Boolean expression for the CARRY is similar to that of an AND gate. Therefore the above Boolean expressions for the half-adder can be realized by using the combination of EX-OR gate and AND gate as shown below.

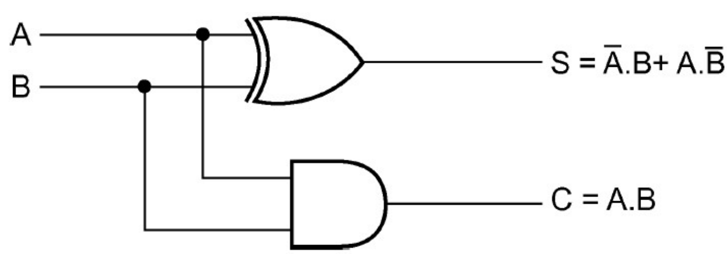


Figure 16: Realizing half-aader circuit by using EX-OR and AND

Full-adder: A full-adder is an arithmetic circuit block that can be utilized for adding three binary digits (bits). The full-adder circuit has the advantage over the half-adder circuit in that it can add three bits. A full-adder is very important for adding larger binary numbers. Full-adder will have two input bits (referred to as A &B) along with an input CARRY (C_{in}) and produces a SUM (S) and an output CARRY (C_{out}). The following figure shows the truth table consisting of all the possible combinations of the input values and the corresponding SUM and output CARRY. The logic symbol for the full-adder is shown in figure 17(b).

A	B	C_{in}	SUM (S)	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a)

(b)

Figure 17: (a) Truth table and (b) Logic symbol for full-

The full-adder will add the two input bits and the input CARRY. The sum of the two bits A and B is an EX-OR logic A⊕B. Therefore adding the input CARRY (C_{in}) to A and B has to be done by EX-ORing it with A⊕B. Therefore the SUM of a full-adder can be written as

$$\text{SUM:}\quad S = A \oplus B \oplus C_{in}$$

This logic can be realized by using two 2-input EX-OR gates as shown below.

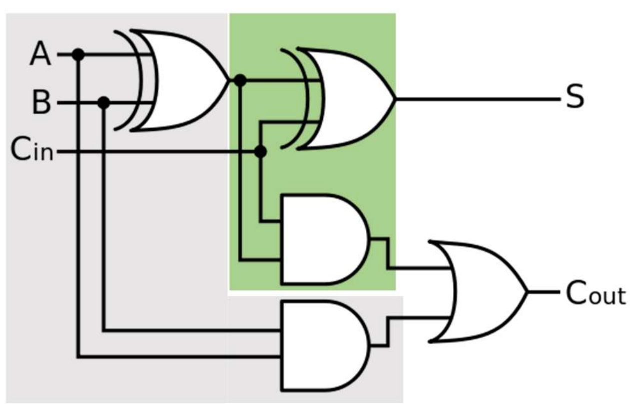


Figure 18: Logic circuit for a full-

From the truth table, it is clear that the output CARRY is 1 when both the inputs of the first EX-OR gate is 1 or when both the inputs of the second EX-OR gate is 1. That means the output CARRY of the full-adder can be obtained as follows: Firstly the input A will be ANDed with input B, after this A⊕B will be ANDed with C_{in}. Secondly, the two ANDed outputs are ORed to get the C_{out} as shown in the above figure. The C_{out} of a full-adder can be witten as

$$C_{out} = A.B + (A \oplus B).C_{in}$$

From the above figure, we can say that there are two half-adder circuits in a full-adder which are highlighted in grey and green colours.

5.7 Boolean Algebra

Digital circuits operate based on the binary number system (0 and 1). These circuits use Boolean algebra as a mathematical tool for analysis and design. Boolean algebra is a mathematical framework upon which logic design is based. It is very different from conventional arithmetics and it is based on the logical statements that can be referred to as either true (1) or false (0). In boolean algebra, the variables and constants are allowed to have only two values- either 1 or 0. Boolean algebra consists of certain symbols and a set of rules for manipulating these symbols. The difference between conventional algebra and Boolean algebra is given below.

conventional algebra	Boolean algebra
letter symbols might be assigned to have any number of values including infinity	It can take either 1 or 0
The values of a variable can have numerical significance	Values of a variable have a logical significance
Multiplication and addition are represented as '.' and '+' respectively	'.' referrers AND operation '+' referrers OR operation
A+B is read as A plus B	A+B is read as A OR B
AB is read as A into B	AB is read as A AND B

Variables, Complements and Literals in Boolean Expressions

Variables:

In a Boolean expression, variables are indicated with different symbols. They can take only two values. Either '0' or '1'. Consider the following Boolean expressions.

$$\bar{A}.B + A.\bar{B} + A.\mathrm{B} + \mathrm{A.B.C}$$

$$P.Q + P.\bar{Q} + P.Q.R.S$$

In the first Boolean expression, the three variables are A, B and C. In the second expression, the variables are P, Q, R and S.

Complement

The complement of a variable is obtained by taking the inverse of that variable and it is represented by putting a bar over the variable. Complement is not considered a separate variable.

Literals

The total number of times the variables and their complements appearing in a Boolean expression is called a *literal*. In the first expression, there are nine literals and in the second expression, there are eight literals.

Complement and dual of a Boolean expression

The complement of a given Boolean expression can be obtained by following the steps given below.

- In the given expression, complement each literal
- In the next step, change all '.' to '+' and all '+' to '.'
- Convert all 0s to 1s and all 1s to 0s.

Example: Boolean expression $\bar{A}.B + A.\bar{B}$

Complement $(A + \bar{B}).(\bar{A} + \mathrm{B})$

The dual of a Boolean expression is obtained by

- Leaving all literals unchanged.
- Changing all '.' to '+' and all '+' to '.'
- Changing all 0s to 1s and all 1s to 0s.

Example: Boolean expression $\bar{A}.B + A.\bar{B}$

Dual $(\bar{A} + B).(A + \bar{B})$

Boolean addition and multiplication

In Boolean algebra, addition is equivalent to the OR operation and multiplication is equivalent to the AND operation. The basic rules are

Boolean addition	Boolean multiplication
❖ 0+0=0	❖ 0.0=0
❖ 0+1=1	❖ 0.1=0
❖ 1+0=1	❖ 1.0=0
❖ 1+1=1	❖ 1.1=1

Laws of Boolean algebra

The following are the basic laws of Boolean algebra

1) Commutative law
2) Associative law and
3) Distributive law.

Commutative law

The commutative law of addition for two variables is expressed as

$$A + B = B + A$$

The commutative law of multiplication for two variables is expressed as

$$AB = BA$$

Associative law

The associative law of addition for three variables is expressed as

$$A + (B+C) = (A+B) + C$$

The associative law of multiplication for three variables is expressed as

$$A(BC) = (AB)C$$

Distributive law

The distributive law for three variables is expressed as

$$A(B+C) = AB + AC$$

Rules of Boolean algebra

The following are the basic rules in Boolean algebra that will help in manipulating and simplifying Boolean expressions,

- A+0=A
- A+1=1
- A.0=0
- A.1=A
- A+A=A
- A+$\bar{A}$=1
- A.A=A
- A.$\bar{A}$=0
- $\bar{\bar{A}}$=A
- A+AB=A
- A+$\bar{A}$B=A+B
- (A+B)(A+C)= A+AC+BA+BC

DeMorgan's Theorems

DeMorgan's first theorem can be stated as follows.

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

For two variables, this can be expressed as $\overline{AB} = \bar{A}+\bar{B}$

DeMorgan's second theorem can be stated as follows.

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

For two variables, this can be expressed as $\overline{A + B} = \bar{A}\,\bar{B}$

NUMERALS

1) Add the following binary numbers
    a) 11+11
    b) 100+10
    c) 111+11
    d) 110+100
    e) 10011+1111101

ANSWERS

a) $\begin{array}{r} 11 \\ \underline{+1} \\ 110 \end{array}$

b) $\begin{array}{r} 100 \\ \underline{+\ \ 10} \\ 110 \end{array}$

c) $\begin{array}{r} 111 \\ \underline{+\ \ \ 11} \\ 1010 \end{array}$

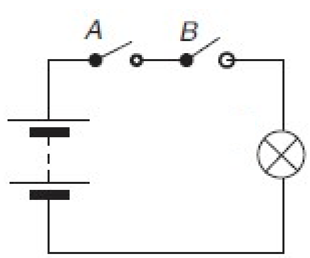
d) $\begin{array}{r} 110 \\ \underline{+\ 100} \\ 1010 \end{array}$

e) $\begin{array}{r} 10011 \\ \underline{+\ 1111101} \\ 10010000 \end{array}$

2) Subtract the following binary numbers
    a) 111100-11110
    b) 1100100-110010
    c) 11001-1001

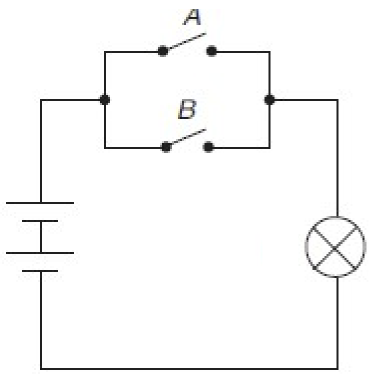
3) Represent the AND function by switch analogy.

ANSWER



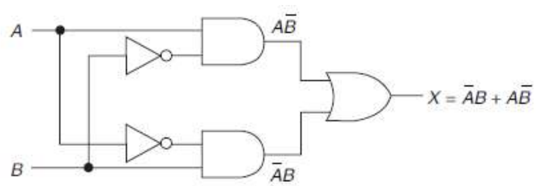
4) Represent the OR function by switch analogy.

ANSWER



5) Draw the combination of AND, OR, and NAND gates to provide the XOR function.

ANSWER



6) Proove that $A\,(\bar{A} + C)\,(\bar{A}B + \bar{C}) = 0$

ANSWER

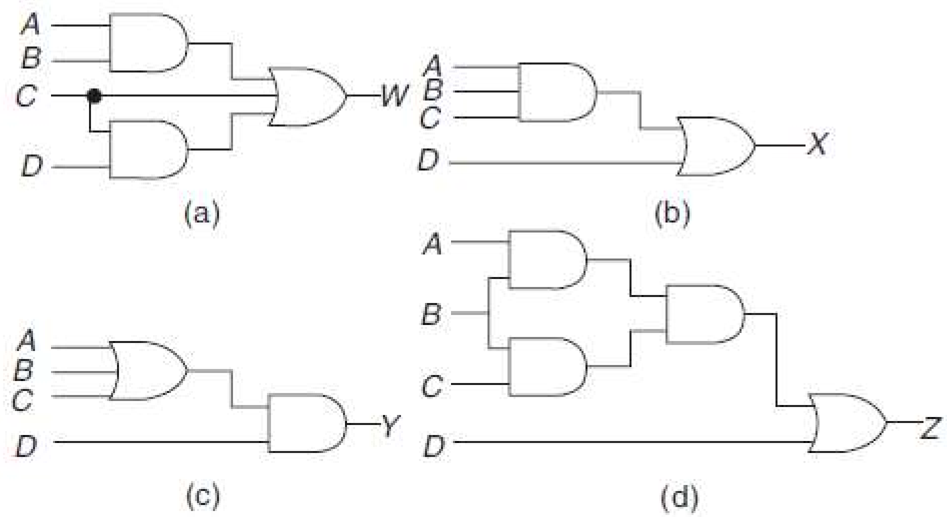
$$
\begin{aligned}
A(\bar{A} + C)\,(\bar{A}B + \bar{C}) &= (A\bar{A} + AC)\,(\bar{A}B + \bar{C}) \\
&= (0 + AC)\,(\bar{A}B + \bar{C}) \\
&= AC\bar{A}B + AC\bar{C} \\
&= A\bar{A}BC + AC\bar{C} \\
&= 0 \cdot BC + A \cdot 0 \\
&= 0
\end{aligned}
$$

7) Verify DeMorgan's law for the Boolean function $\overline{A + B} = \bar{A} \cdot \bar{B}$ through truth table

ANSWER

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

8) Write the Boolean equation for each of the logic circuits shown below

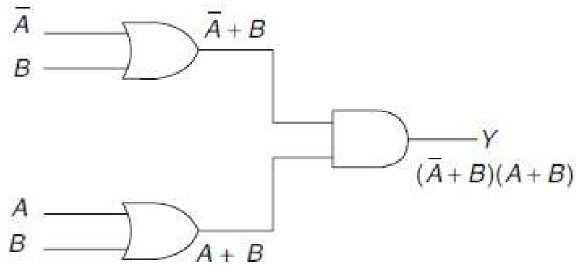


ANSWER

(a) $W = (AB + C) + CD$
(b) $X = ABC + D$
(c) $Y = (A + B + C)D$
(d) $Z = ((AB)(BC)) + D$

9) Show the logic circuit for the Boolean equation $Y = (\overline{A} + B)(A + B)$ .Simplify the circuit as much as possible using algebra.

ANSWER



Simplifying yields $Y = \bar{A}A + \bar{A}B + BA + BB$

$$Y = \bar{A}B + AB + B$$

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

Since Y = B, we don't need a logic circuit. All we need is a wire connecting the input B to the output Y.

10) Find the following binary arithmetic
    a) 1 0 1 1 1 × 1 1 0
    b) 1 0 0 0 1 × 101
    c) 11010÷101
    d) 10010011÷1011

ANSWERS

a)

$$\begin{array}{r} 1\,0\,1\,1\,1 \\ \times\,1\,1\,0 \\ \hline 0\,0\,0\,0\,0 \\ 1\,0\,1\,1\,1 \\ 1\,0\,1\,1\,1 \\ \hline 1\,0\,0\,0\,1\,0\,1\,0 \end{array}$$

b)

```
     1 0 0 0 1
     × 1 0 1
  ----------
     1 0 0 0 1
   0 0 0 0 0
 1 0 0 0 1
  ----------
 1 0 1 0 1 0 1
  ----------
```

c)

```
       101      ⟶ Quotient
101 ) 11010
      -101
      ----
        11
       -00
      ----
        110
       -101
      ----
          1 ⟶ Reminder.
```

d)

```
        00001101 ⟶ Quotient
1011 ) 10010011
       -1011
       --------
        001110
          1011
       --------
          001111
            1011
       --------
             100 ⟶ Reminder.
```