CHAPTER 1

Basics of Digital Electronics

The Electronics circuits and systems are of two kinds namely analog and digital. Analog circuits are those in which voltages and currents vary continuously through the given range. For example, signal generator, power supplies and electric motors. In digital circuits, the voltage levels assume a finite number of distinct values. These are often called as switching circuits because the voltage levels in digital circuits are assumed from one value to another instantaneously.

1.1 Introduction To Digital Electronics

Digital electronics is a field of electronics involving the study of digital signals and the engineering of devices that use or produce them. This is in contrast to analog electronics and analog signals.

The binary number system was refined by Gottfried Wilhelm Leibniz (published in 1705) and he also established that by using the binary system, the principles of arithmetic and logic could be joined. Digital logic as we know it was the brain-child of George Boole in the mid-19th century. In an 1886 letter, Charles Sanders Peirce described how logical operations could be carried out by electrical switching circuits. Eventually, vacuum tubes replaced relays for logic operations. Lee De Forest's modification, in 1907, of the Fleming valve can be used as an AND gate. Ludwig Wittgenstein introduced a version of the 16-row truth table as proposition 5.101 of *TractatusLogico-Philosophicus* (1921). Walther Bothe, inventor of the coincidence circuit, shared the 1954 Nobel Prize in physics, for the first modern electronic AND gate in 1924.

Digital electronic circuits are usually made from large assemblies of logic gates. Digital describes electronic technology that generates, stores, and processes data in terms of two states: 1 and number 0.

Digital electronics deals with the electronic manipulation of numbers, or with the manipulation of varying quantities by means of numbers. Because it is convenient to do so, today's digital systems deal only with the numbers 'zero' and 'one', because they can be represented easily by 'off and 'on' within a circuit.

Digital stand for digit, digital electronics basically has two possible conditions which are 0 (low logic) and 1 (high logic). These systems use digital signals that are composed of mathematical features to work.

1.2 Need of Digital

Almost all devices, used on a daily basis make use of digital electronics in some capacity. Digital electronics simply refers to any kind of circuit that uses digital signals rather than analogue. It is constructed using circuits calls logic gates, each of which performs a different function. The circuit will make use of different components that are all standard, but that are put together in different combinations to achieve the desired result. Its circuit will also include resistors and diodes, which are used to control current and voltage.

Even many of our household items make use of digital electronics. This could include laptops, televisions, remote controls and other entertainment systems, to kitchen appliances like dishwashers and washing machines. Computers are one of the most complex examples and will make use of numerous, complex circuits. There may be millions of pathways within the circuit, depending on how complex the computer and its functions need to be.

Digital Electronics is very important in today's life because if digital circuits compared to analog circuits are that signals represented digitally can be transmitted without degradation due to noise.Many systems use a maximum of analog and digital electronics to take advantage of each technology. For example: A typical CD player accepts digital data from the CD drive and converts it to an analog signal for amplification.

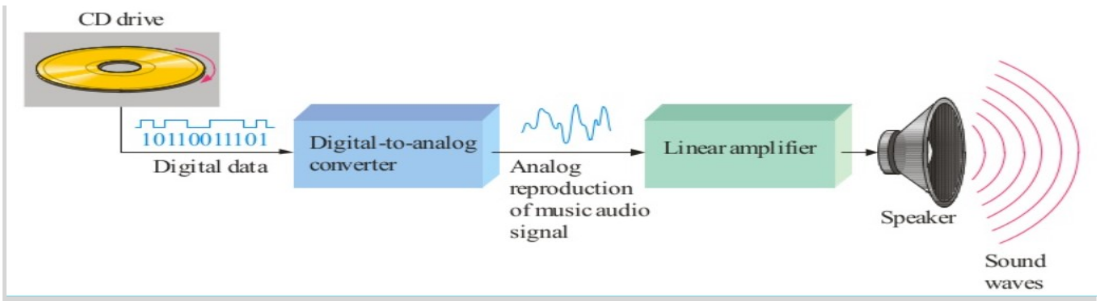

CD drive
10110011101
Digital data
Digital-to-analog converter
Analog reproduction of music audio signal
Linear amplifier
Speaker
Sound waves

Figure 1 How a CD player works?

Digital Electronics is very important in today's life because if digital circuits compared to analog circuits, signals those are represented digitally can be transmitted without degradation due to noise.

For instance, a continuous audio signal transmitted as a sequence of 1s and 0s, can be reconstructed without error, provided the noise picked up in transmission is not enough to prevent identification of the 1s and 0s.

An hour of music can be stored on a compact disc using about 6 billion binary digits. Also in digital system information stored is easier than that of analog system. All real life signals are analog in nature and at first sight; it seems use of analog is much better as compared to digital signal.

But digital signals have various advantages over the analog signal like

- Digital signal is used in communication process to minimize the effect of noise.
- Digital Signals carry more information per unit time as compared to analog signals.
- Quality of digital signal is better over long distance transmission.
- Use of bandwidth is less in case of transmission using digital signals.
- Digital signals can be encrypted.
- Noise removal is easy in digital signals.

1.3 Merits and Demerits of Analog and Digital Signals

Analog signal is a continuous signal as represented in figure 2, in which one time-varying quantity represents another time-based variable. These kind of signals works with physical values and natural phenomena such as earthquake, frequency, volcano, speed of wind, weight, lighting, etc.

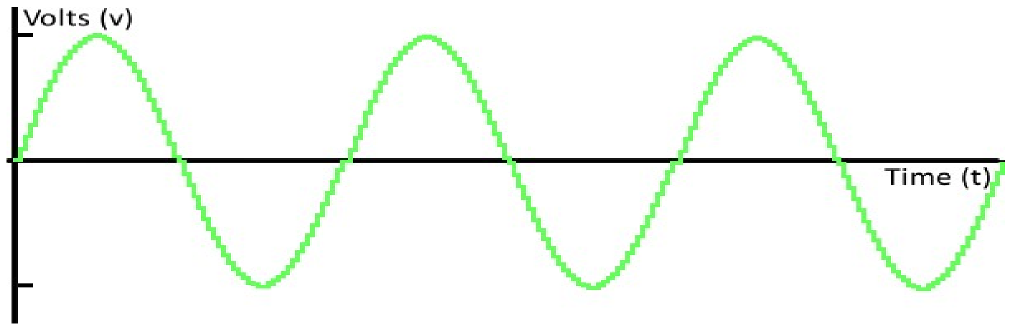


Figure 2 Analog Signal

A digital signal is a signal that is used to represent data as a sequence of separate values at any point in time. It can only take on one of a fixed number of values. This type of signal represents a real number within a constant range of values and is presented in figure 3.

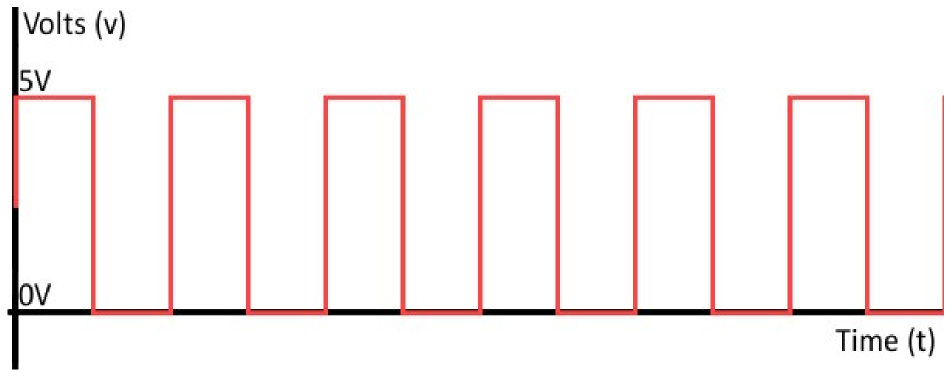


Figure 3 Digital signal

Characteristics of Analog Signal:

- These are time-varying electronic signals.
- Lowest and highest values which is either positive or negative.
- It is often either periodic or non-periodic.
- Analog Signal works on continuous data.
- The accuracy of the analog signal isn't high in comparison to the digital signal.
- It helps you to live natural or physical values.
- Analog signal output form is like Curve, Line, or Graph, so it's going to not be meaningful to all or any.

Advantages of Analog Signals:

Here, are some pros/benefits of Analog Signals.

- It is Easier in processing.
- Analog Signals are best fitted to audio and video transmission.
- It has a coffee cost and is portable.
- It posses higher density.
- Not necessary in Analog Signals to shop for a replacement graphics board.
- It uses less bandwidth than digital sounds.
- It Provide more accurate representation of a sound.
- It is the natural sort of a sound.

- It has less bandwidth.
- Binary digits 0 and 1 represent the optical pulse for storing, processing and transmitting information.

Disadvantages of Analog Signals:

Here are cons/drawback of Analog Signals are as follows.

- Analog tends to possess a lower quality signal than digital.
- The cables are sensitive to external influences.
- Analog wire is expensive and not easily portable.
- In this, it has Low availability of models with digital interfaces.
- Recording analog sound on tape is sort of expensive if the tape is broken.
- It offers limitations in editing.
- Tape is becoming hard to seek out.
- It is quite difficult to synchronize analog sound.
- Quality is definitely lost.
- Data can become corrupted in analog signals.
- Most of sound capturing devices such as phones etc which may become confusing to store a digital signal.
- Digital sounds can cut an analog acoustic wave which suggests that you simply can't get an ideal reproduction of a sound.
- It offers poor multi-user interfaces.

Advantages of Digital Systems

- **Easier Designing**: The Digital systems can be easily designed as they involve digital signals. These signals do not require exact value at a particular time but it consists of range of particular values of voltage. Thus, it comprises of basically two values 0 and 1 i.e. high or low.
- **Noise Immune**: Digital systems are noise immune because digital signal consists of range of particular values. Thus, when noise is introduced in the medium and digital signal and analog signal both passes through it. The analog signal will be affected more because it varies continuous with time so it is difficult to identify that noise has destroyed which value of voltage. While in case of digital system, noise effect the

particular range of the signal thus, it is clear to identify the particular range of filtering is also easy in case of digital signals.

- **Information Storage is Simpler**: The storage of information in digital systems is easy. It can be stored by latching thus; the it can be stored for a long period of time.
- **High Accuracy and Precision**: The digital signal offers high accuracy and precision. This is because the processing of digital signal is done through the switching circuit. While in case of analog signals the processing and its output is highly dependant on circuit components. The accuracy obtained in analog circuits is restricted to 3-4 digits while in case of digital signals the accuracy is far more than the analog circuit.
- **Programmable**: The digital systems are easily programmable but analog system becomes complex when excessive programming of components is done.

Disadvantages of Digital Systems

Everything comes up with pros and cons. Similarly digital systems to hold certain disadvantages which are as follows:-

- Expensive: Digital systems are expensive because it involves switching elements.
- Analog nature of Real World Entities: We need to convert the digital output in analog form because all the real world entities are analog.
- Digital electronics circuits require more energy than the analog circuits which are accomplish to do the same tasks. And so these are producing too much heat and so it increases the complexity of the digital electronics circuits

PRACTICE QUESTIONS:

Q1. Name some advantages of Digital Technology.

Q2. Give examples of analog system.

Q3. What is the difference between Analog and Digital Electronics?

Q4. What are disadvantages of Analog systems?

Q5. What are advantages of Digital systems?

CHAPTER 2

Number System

Number system is an arranged set of numerals known as digits with defined rules for arithmetic operations like multiplication, addition, subtraction, etc. Array of digits makes an arbitrary number. Number has two parts-integer and fraction, separated by a radix point (.), known as decimal.

$$N_b = \underbrace{d_{n-1}\, d_{n-2}\ \ldots .\, d_i\ \ldots d_1\, d_0}_{Integer\ part}\ \underbrace{.}_{radix\ point}\ \underbrace{d_{-1}\, d_{-2}\ \ldots .\, d_{-f}\ \ldots .\, d_{-m}}_{Fraction\ part}$$

$$0\ \le \left(d_i\ or\ d_{-f}\right) \le b-1$$

Where, N_b is a number with base b. Number of digits in integer and fraction part is represented by n and m respectively. d_{n-1} is the Most Significant Digit (MSD) and d_{-m} is the least significant digit (LSD).

Position of each digit in a number is assigned a specific weight or index. Leftmost digit with maximum positional weight is known as MSD. Rightmost bit with least positional weight is termed as LSD. Decimal number system, binary, octal and hexadecimal are commonly used number systems. These number systems are widely used in digital circuits, microprocessor and microcomputers. Therefore, it is useful to study the number systems for understanding, designing and analyzing the digital electronics.

2.1Decimal Number System

Decimal number system is well known to all of us. It is a base 10 number system and arranged set of digits 0,1,2,3….,9. For representing the large numbers various digits can be placed at appropriate positions. Fractional part has weight of negative power of 10 whereas integer part has weight of positive power of 10. Decimal number is the sum of the products of the digits with their respective position weight. For example, 3452.453 is a decimal number which can be written as:

3452.4533$\times10^3$ + 4$\times10^2$+5$\times10^1$+ 2$\times10^0$ + 4$\times$ 10^{-1} +5$\times10^{-2}$+ 2$\times10^{-3}$

2.2 Binary Number System

It is a positional weighted system with radix 2. It has only two symbols 0 and 1; termed as bits. Group of eight binary bits is known as Byte and group of four binary bits is known as nibble. Binary number system is preferred in digital systems because switching circuits have two states which can be represented by two binary digits 0 and 1. Transistors and diodes are two state devices; on-state is represented as 1 and off-state is represented as 0. Counting in binary system is written as sequence of 0 and 1. Four-bit binary counting (0000-1111) with their decimal equivalent is shown in Table1.

Table 1. 4-bit Binary counting and decimal value

Binary Number	Decimal Number	Binary Number	Decimal Number
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

Numbers from different number systems can be converted from one form to another. To differentiate the numbers in different number systems, base is written as a subscript to the number.

2.3 Octal Number System

Binary numbers are consisting of very long array of 0 s and 1s. It is difficult to deal with large array of binary bits. Handling of binary numbers is eased by grouping them into three digits. This group of three digit is termed as Octal with radix 8. Since, the base is $8 = 2^3$, so every three group of 3 binary bits can be written as octal number. In the octal number system, the digits have following eight values 0, 1, 2, 3, 4, 5, 6 and 7. Table2 lists the binary counting.

Table 2. Counting in Octal number system

Octal Number	Binary Number	Decimal Number
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
10	001 000	8
11	001 001	9
12	001 010	10

2.4 Hexadecimal Number System

Similar to octal number system, hexadecimal numbers are also used to represent the binary numbers concisely. Base of this number system is 16. Independent symbols are 0,1,2, 3….9, A, B, C, D, E, F. Binary number is grouped in four bits and referred as hexadecimal numbers. Table3 lists the binary counting with equivalent hexadecimal numbers.

Table 3. Hexadecimal Counting

Hexadecimal Number	Binary Number	Decimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

2.5 Conversions

2.5.1 Base Conversion

Let's think more carefully what a decimal number means. For example, 1234 means that there are four digits and 4 is the right-most no i.e. least significant digit, 1 is the left-most i.e. most significant digit. Then it can also be written as

```
    Original Number:      1     2     3    4

                          |     |     |    |

                 Digit Value: 1000   100    10    1

        Value:         1000 + 200  + 30  + 4  = 1234
```

or simply, 1*1000 + 2*100 + 3*10 + 4*1 = 1234

Thus, each digit has a value: 10^^{0}=1 for the least significant digit, increasing to 10^^{1}=10, 10^^{2}=100, 10^^{3}=1000, and so forth.

Likewise, the least significant digit in a hexadecimal number has a value of 16^^{0}=1 for the least significant digit, increasing to 16^^{1}=16 for the next digit, 16^^{2}=256 for the next, 16^^{3}=4096 for the next, and so forth. Thus, 1234 means that there are four digits; and the total is:

1*4096 + 2*256 + 3*16 + 4*1 = 4660

Table 4 Conversion Table

BIN	OCT	HEX	DEC
0000	00	0	0
0001	01	1	1
0010	02	2	2
0011	03	3	3
0100	04	4	4
0101	05	5	5

0110	06	6	6
0111	07	7	7

1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

2.5.2 Decimal to Binary Conversion

Integer part of decimal number is converted into binary number by continuing divide by 2 and keeping the track of remainders from bottom (leftmost) to top (rightmost). Fractional part is converted by multiplying the binary number with 2 and tracking the integer part. This process is explained with the given example.

Example1: convert the given decimal numbers into binary numbers

$(105.15)_{10} = (?)_2$

Step1: conversion of integer part by division method

$(105)_{10} = (1101001)_2$

2	105	1
2	52	0
2	26	0
2	13	1
2	6	0

2	3	1
2	1	1

Step2: conversion of fractional part by multiplication method

$(0.15)_{10} = (001001)_2$

Given fraction	Product
0.15 ×2	0.30
0.30 ×2	0.60
0.60 ×2	1.20
0.20 ×2	0.40
0.40 ×2	0.80
0.80×2	1.60

Step3: writing the correct order

$(105.15)_{10} = (1101001.001001)_2$

Example2: Convert 152 into binary number

2	152	0
2	76	0
2	38	0
2	19	1
2	9	1
2	4	0
2	2	0
2	1	1

$(152)_{10} = (10011000)_2$

2.5.3 Binary to Decimal conversion

Binary number can be converted into equivalent decimal number system by using the positional weight assigned to each bit. Conversion of $(1100.1011)_2$ into decimal number system is explained.

Example3: $(1100.1011)_2 = (?)_{10}$

$1100.1011 = 1\times2^3 + 1\times2^2+0\times2^1+ 0\times2^0 + 1\times 2^{-1} +0\times2^{-2}+ 1\times2^{-3}+ 1\times2^{-4}$

$= 8+4+0+0+0.5+0+0.125+0.0625$

$= (12.6875)_{10}$

Example4: $(10110.001)_2 = (?)_{10}$

$= (1\times2^4)+(0\times2^3)+(1\times2^2)+(1\times2^1)+(0\times2^0)+(0\times2^{-1})+(0\times2^{-2})+(1\times2^{-3})$

$=(1\times16)+(0\times8)+(1\times4)+(1\times2)+(0\times1)+ (0\times1/2)+(0\times1/4)+(1\times1/8)$

$=16+0+4+2+0+0+0+0.125$

$=\mathbf{(22.125\ )_{10}}$

2.5.4 Conversion of Octal Number to Binary Number

To convert an octal number into binary, each octal digit is represented by three binary bits.

Example 5: Convert $(152)_8$ into binary number

Given octal number	1	5	2
Representation as group of three binary bits	001	101	010

Result **$(152)_8=(001101010)_2$**

Example 6: Convert $(357.52)_8$ into binary number system is explained.

Given octal number	3	5	7	.	5	2
Representation as group of three binary bits	011	101	111	.	101	010

Result $(011101111.101010)_2$

2.5.5 Conversion of Octal Number to Decimal Number System

To convert octal number into equivalent decimal, each octal digit is multiplied by equivalent position weight and product is aggregated. For example, convert $(4057.06)_8$ into decimal number.

Example 7: Convert given octal number into decimal

$(152.25)_8$ $=(1\times8^2)+(5\times8^1)+(2\times8^0)+(2\times8^{-1})+(5\times8^{-2})$

$(152.25)_8$ $=64+40+2+(2\times1/8)+(5\times1/64)$

$(152.25)_8$ $=64+40+2+0.25+0.078125$

$(152.25)_8$ $=(106.328125)_{10}$

Example 8: Convert given octal number into decimal

$(4057.06)_8 = 4\times8^3 + 0\times8^2+5\times8^1+ 7\times8^0 + 0\times 8^{-1} +6\times8^{-2}$

$= 2048+0+40+7+0+0.0937$

$= (2095.0937)_{10}$

2.5.6 Conversion of Decimal Number System to Octal

Mixed decimal number is converted separately. Integer decimal is divided successively by 8 till quotient value is 0. Last remainder is written as most significant digit. Given decimal fraction is multiplied by 8 successively; integer of product from top to bottom is written as the equivalent octal fractional.

Example 9: Convert $(678.93)_{10}$ into octal number system

Step1: conversion of integer part by division method and write remainder from bottom to top.

$(678)_{10} = (1246)_8$

8	678	6
8	84	4
8	10	2
8	1	1

Step2: conversion of fractional part by multiplication method and write integer part of fraction from top to bottom.

$(0.93)_{10}=(0.7341)_8$

Given fraction	Product
0.93 ×8	7.44
0.44×8	3.52
0.52 ×8	4.16
0.16 ×8	1.28

Step3: writing the correct order.

$(678.93)_{10} = (1246.7341)_8$

Example10: Convert $(152)_{10}$ into octal number system

8	152	0
8	19	3
8	2	2
	0	

$(152)_{10} = (230)_8$

2.5.7 Conversion of Binary Number System to Octal

Grouping of binary number is done for the binary to octal conversion. Integer part of binary number is grouped in three bits each starting from right to left. Fractional part of binary number is grouped from left to right. Each group is replaced with its equivalent octal digit.

Example11: Convert $(11011001111100.00110101)_2$ into octal number system

Weight	21 421 421 421 421 . 421 421 421
Groups	11 011 001 111 100 . 001 101 010
Octal	3 3 1 7 4 . 1 52
Result	$(33174.152)_8$

Example12: Convert **$(111110101011.0011)_2$**into octal number system

Firstly, we make pairs of three bits on both sides of the binary point.

Groups	111	110	101	011.	001	100
Octal	7	6	5	3 .	1	4
Result	**$(7653.14)_8$**					

2.5.8 Conversion of Hexadecimal Number to Binary Number

To convert a hexadecimal number into binary, each digit is represented by four binary bits.

Example 13: Conversion of $(957.A2)_{16}$ into binary number system is explained.

Given Hexadecimal number	9	5	7	.	A	2
Representation as group	1001	0101	0111	.	1010	0010

of four binary bits

Result (100101010111.10100010)$_2$

2.5.9 Conversion of Hexadecimal Number to Decimal Number System

To convert hexadecimal number into equivalent decimal, each hexadecimal digit is multiplied by equivalent position weight and product is aggregated. For example, convert (4057.06)$_{16}$ into decimal number.

Example14:$(4057.06)_{16} = 4\times16^3 + 0\times16^2+5\times16^1+ 7\times16^0 + 0\times 16^{-1} +6\times16^{-2}$

$= 16384+0+80+7+0+0.0234$

$= (16471.0234)_{16}$

2.5.10 Conversion of Hexadecimal Number to Octal Number System

Easiest way to convert the hexadecimal number into octal number is first binary conversion and then into octal number as explained in given example.

Example 15: Convert (957.A2)$_{16}$ into Octal number

Given Hexadecimal number	9	5	7	.	A	2
Representation as group of four binary bits	1001	0101	0111	.	1010	0010

Binary Number (100101010111.10100010)$_2$

Group of three bits 100 101 010 111.101 000 100

Octal number 4 5 2 7 . 5 0 4

Result (4527.504)$_8$

2.5.11 Conversion of Decimal Number System to Hexadecimal

Mixed decimal number is converted separately. Integer decimal is divided successively by 16 till quotient value is 0. Last remainder is written as most significant digit. Given decimal fraction is multiplied by 16 successively; integer of product from top to bottom is written as the equivalent octal fractional.

Example 16: Convert (2598.675)$_{10}$ into Hexadecimal number system

Step1: conversion of integer part by division method and write remainder from bottom to top.

$(2598)_{10} = (A26)_{16}$

Successive division	Quotient	Remainder
2598÷16	162	6
162÷16	10	2
10÷16	0	10 is A in Hexadecimal

$\uparrow$

Step2: conversion of fractional part by multiplication method and write integer part of fraction from top to bottom.

$(0.675)_{10} = (0.ACCC)_{16}$

Given fraction	Product
0.675 ×16	10.8
0.800×16	12.8
0.800 ×16	12.8
0.800×16	12.8

$\downarrow$

Step3: writing the correct order.

$(2598.675)_{10} = (A26.ACCC)_{16}$

2.5.12 Conversion of Binary Number System to Hexadecimal

First grouping of binary number is done for the binary to hexadecimal conversion. Integer part of binary number is grouped in four bits starting from right to left. Fractional part of binary number is grouped from left to right. Each group is replaced with its equivalent hexadecimal digit.

Example17: Convert $(011001111101.00110101)_2$ into hexadecimal number system

```
Weight              8421  8421 8421 . 8421  8421
Groups          011001111101 . 00110101
Hexadecimal           6    7D.   35
Result          (67D.35)16
```

2.5.13 Conversion of Octal Number System to Hexadecimal

The easiest way for octal to hexadecimal conversion is to first convert into binary and then binary into hexadecimal.

Example18: Convert $(757.42)_8$ into hexadecimal number

Given Octal number	7	5	7	.	4	2
Representation as group of three binary bits	111	101	111	.	100	010

Binary Number $(111101111.100010)_2$

Group of four bits 0001 1110 1111. 1000 1000

Hexadecimal number 1 E F . 8 8

Result $(1EF.88)_{16}$

2.6 Signed and unsigned number

In decimal number system, + sign before a number indicates it as a positive number and - sign before a number indicates it as a negative number. It is not convenient on computers. So, a proper mechanism is required to represent the negative numbers for binary number systems. There are two methods to represent the negative binary numbers: (i) Sign magnitude presentation (ii) complemented form.

2.6.1 Sign magnitude representation:

In sign magnitude one additional bit is added in front of number. Positive number is represented by 0 sign bit and negative number is represented by 1 as a sign bit. Some of the examples are shown in Table.

(+1100101)2	(01100101)
(+101.001)2	(0101.001)2
(-10010)2	(110010)2
(-110.101)2	(1110.101)2

Example19: Represent the given numbers into sign magnitude format. (+41) ad (-41)

8-bits Binary equivalent of 41 = 00101001

Sign representation of (+41) = 0 **0101001**

$$\begin{tikzcd} 0 \arrow[d] & \mathbf{0101001} \arrow[d] \\ \text{Sign bit} & \text{Magnitude} \end{tikzcd}$$

Sign representation of (-41) = 1 **0101001**

$$\begin{tikzcd} 1 \arrow[d] & \mathbf{0101001} \arrow[d] \\ \text{Sign bit} & \text{Magnitude} \end{tikzcd}$$

2.6.2 Complement form representation

There are two complement form representation.

- Diminished Radix Complement (DRC) or (r-1) – complement. DRC is known as one's-complement in binary number system and 9's complement in decimal number system.
- The 9's complement of 546700 is (999999-546700=453299)
- The 1's complement of 1011000 is 0100111.
- Radix Complement (RXC) or r-complement Binary numbers. RXC is known as two's-complement in binary number system and 10's complement in decimal number system. The r's complement is obtained by adding 1 to the (r-1) 's complement.
- The 10's complement of 546700 is ((999999-546700) +1=453300)
- The 2's complement of 1011000 is 0100111+1=0101000.

One's complement representation: If each 1 of a binary number is replaced by 0 and each 0 by 1, the resultant is the 1's complement of given number. This method is used for signed number representation. The maximum positive numbers represented in 1's complement form is (2^{n-1} -1) and the negative numbers is – (2^{n-1} -1). In one's complemented form, maximum seven positive and seven negative numbers can be represented by four binary bits.

Example19: Represent the given numbers in 1's complement form: +7, -7, +8, -8, +15, -15.

Solution:

Decimal	Signed 1's Complement	Decimal	Signed 1's Complement
+7	0111	-7	1000
+8	01000	-8	10111
+15	01111	-15	10000

Two's Complement representation: It is obtained by adding one in the one's complemented form of the binary number. For example, 2's complement of 0110 is 1010. Since 0110 is $(+6)_{10}$, therefore, 1010 is $(-6)_{10}$ in 2's complemented form. MSB is 1, number is negative whereas if the MSB is 0 the number is positive. The maximum positive numbers represented in 1's complement form is $(2^{n-1} -1)$ and the negative numbers is $-(2^{n-1})$. In two's complement form, maximum seven positive and eight negative numbers can be represented by four binary bits. 2's complement of a 2's complement number is number itself.

Example 20: Find two's complement of given numbers:

Number	1's complement	1's complement+1 = 2's complement
01100100	10011011	10011011+1 = 10011100
10010010	01101101	01101101+1 = 01101110
11011000	00100111	00100111+1 = 00101000
01100111	10011000	10011000+1 = 10011001

Example 21: Represent $(-17)_{10}$ in sign magnitude, one's and two's complemented form.

Solution: $(+17)_{10}$ in sign magnitude is represented as $(010001)_2$.

$(-17)_{10}$ in sign magnitude is represented as $(110001)_2$.

$(-17)_{10}$ in 1's complemented form is $(101110)_2$.

$(-17)_{10}$ in 2's complemented form is $(101111)_2$.

2.7 Binary Arithmetic

Mathematical operations like addition, subtration, multiplication and divison can be performed on binary numbers. Binary numbers have only two symbols, thus calculation is much simpler.

2.7.1 Binary Addition

Binary addition follows some specific rules which are given in Table4 below. Similar to the decimal addition carry is added with next higher order bits.

Table 5: Binary Addition rules

Augend (A)	Addend (B)	Sum (S)	Carry (C)	Output
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	10

Example 22: (i) Add 1011.01 with 1100.10

(ii) Add 01101010, 00001000,10000001, and 11111111.

Solution (i):

$$\begin{array}{cccccccc}
 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & . & \mathbf{0} & \mathbf{1} \\
+ & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & . & \mathbf{1} & \mathbf{0} \\
\hline
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & . & \mathbf{1} & \mathbf{1} \\
\hline
\end{array}$$

Output = 10111.11

Solution (ii):

$$\begin{array}{lcccccccccc}
\text{Carry->} & & & & & & \mathbf{1} & & & & \\
 & & & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \_ \\
 & & \mathbf{1} & & & & & & & & \\
\hline
 & & & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\
 & & & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 & & & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
 & + & & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
\hline
 & \mathbf{1} & & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\
\hline
\end{array}$$

The sum= 111110010

From this example it can be concluded that, even number of one's in column gives 0 sum bit and odd number of one's in column gives 1 as sum bit. Every pair of one's generate a carry.

2.7.2 Binary Subtraction

Rules for binary subtraction are listed in Table5 below.

Table 6: Binary Subtraction rules

Minuened	Subtrahend	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Example 23: Subtract 111.111 from 1010.010

1	0	1	0	.	0	1	0
-	1	1	1	.	1	1	1
0	0	1	0	.	0	1	1

Subtract 110 from 1011

	1	0	1	1
-	0	1	1	0
	0	1	0	1

2.7.3 Binary Multiplication

Binary multiplication is similar to the decimal multiplication with following rules: 0×0 =0, 0×1 =0, 1×0 =0, 1×1=1.

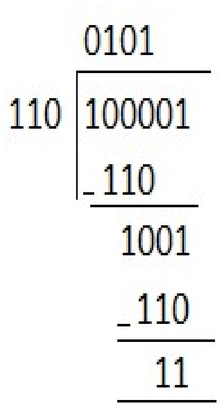
Example24: Multiply 100.1 by 110.1 using binary multiplication

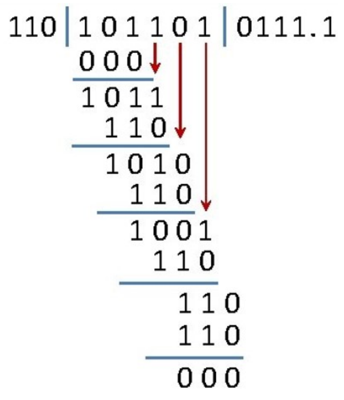
			1	0	0	.	1		Multiplicand
			1	1	0	.	1		Multiplier
			1	0	0		1	i	Partial Products
		0	0	0	0		×	ii	
	1	0	0	1	×		×	iii	
1	0	0	1	×	×		×	iv	
1	1	1	0	1	0		1		Final Product=11101.01

2.7.4 Binary Divison

Binary divison is similar to the decimal divison.

Example25: Divide: (i) 100001 by 110 (ii) 101101 by 110





2.7.5 1's Complement Subtraction

Rules for 1's complement subtraction is listed as:

(1) Write the given numbers in positive binary numbers.
(2) 1's complement the negative number.
(3) Add the numbers.
(4) If carry is generated; end around the carry.
(5) If MSB=1 (carry generated or not); number is negative and in 1's complementd form.
(6) If MSB=0 (carry generated or not); number is positive.

Example 26: Subtract 27.50 from 68.75 using the 12 bit 1's complement arithmetic.

+68.75 = 01000100.1100

+27.50 = 00011011.1000

To perform 1's complement subtraction: 68.75 + (-27.50)

So, Complement the negative number.-27.50 = 11100100.0111 and Perform addition

	6	8	.	7	5		0	1	0	0	0	1	0	0	.	1	1	0	0
-	2	7	.	5	0		1	1	1	0	0	1	0	0	.	0	1	1	1
+	4	1	.	2	5	carry1	0	0	1	0	1	0	0	1	.	0	0	1	0
						End around carry												+	1
						MSB=1, Number is positive	0	0	1	0	1	0	0	1	.	0	1	0	0

Example 27: Subtract 25 from 14 using the 8 bit 1's complement arithmetic.

+14 = 00001110

+25= 00011001

Subtraction: 14+(-25)

-25= 11100110

0	0	0	0	1	1	1	0
1	1	1	0	0	1	1	0
1	1	1	1	0	1	0	0

There is no carry, MSB=1, number is negative and in 1's complemented form. (14-25 = -11)

Complement the result and put negative sign to get the magnitude sign.

2.7.6 Two's Complement Subtraction

Rules for 2's complement subtraction is listed as:

(1) Write the given numbers in positive binary numbers.
(2) 2's complement the negative number.
(3) Add the numbers.
(4) If carry is generated; discard the carry.
(5) If MSB=1 (carry generated or not); number is negative and in 2's complementd form. 2's complement the result to get the magnitude.

(6) If MSB=0 (carry generated or not); number is positive.

Example28: Subtract -75 from 26 using 8-bit 2's complement method.

+75 = 01001011

-75 = 10110101   (by two's complement)

+26= 00011010

Add :   00011010

10110101

110011111

There is no carry, MSB=1, result is negative and in two's complement form. The magnitude is 2's complemented, that is 00110001 =49, Result is -49.

Example29: Subtract 45.75 from 87.5 using the 12 bit 2's complement method.

Solution: 87.50 + (-45.75)

+87.50 = 01010111.1000

-45.75 = 11010010.0100 (in 2's complemented form)

100101001.1100

There is a carry, ignore it. MSB=0, result is positive and in normal binary form. The result is 41.75.

2.8 Binary Codes

- Coding is the process of altering the characteristics of information to make it more suitable for intended application
- Coding schemes depend on Security requirements, Complexity of the medium of transmission, Levels of error tolerated, Need for standardization.
- In binary coding each item of information is assigned a unique combination of 1's and 0's
- Come commonly used binary codes are BCD and Gray codes.

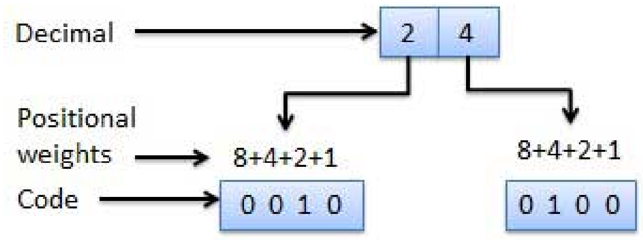
2.8.1 Classification of codes

The codes are broadly categorized into following four categories.

- Weighted Codes
- Non-Weighted Codes
- Alphanumeric Codes
- Error Detecting and Correcting Codes
- Reflective codes
- Sequential codes

Weighted Codes

Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.



Non-Weighted Codes

In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

Alphanumeric Codes

The alphanumeric codes are the codes that represent numbers and alphabetic characters.The following three alphanumeric codes are very commonly used for the data representation.

- American Standard Code for Information Interchange (ASCII).
- Extended Binary Coded Decimal Interchange Code (EBCDIC).
- Five- bit Baudot Code.

ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code. ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.

Error Detection and Correction Code

These codes are used to detect and correct the error bits in the received bit stream. Parity bits, hamming code are the examples of error detection and correction code.

Reflective codes:

A code is reflective when the code is self-complementing. In other words, when the code for 9 is the complement the code for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.

2421BCD, 5421BCD and Excess-3 code are reflective codes

Sequential codes:

In sequential codes, each succeeding 'code is one binary number greater than its preceding code. This property helps in manipulation of data.

8421 BCD and Excess-3 are sequential codes.

2.8.2 BCD (Binary Coded Decimal) Codes

It is a **weighted code** the decimal value of a code is the algebraic sum of the weights of 1s appearing in the number. Let $(A)_{10}$ be a decimal number encoded in the binary form as $a_3a_2a_1a_0$. Then

$$(A)_{10} = w_3a_3 + w_2a_2 + w_1a_1 + w_0a_0$$

where w_3, w_2, w_1 and w_0 are the weights selected for a given code, and a_3, a_2, a_1 and a_0 are either 0s or 1s. The more popularly used codes have the weights as

w_3	w_2	w_1	w_0
8	4	2	1
2	4	2	1
8	4	-2	-1

The decimal numbers in these three codes are

Table 7: BCD codes

Decimal digit	Weights 8 4 2 1	Weights 2 4 2 1	Weights 8 4 -2 -1
0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 1 1 1
2	0 0 1 0	0 0 1 0	0 1 1 0
3	0 0 1 1	0 0 1 1	0 1 0 1
4	0 1 0 0	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1	1 0 1 1
6	0 1 1 0	1 1 0 0	1 0 1 0
7	0 1 1 1	1 1 0 1	1 0 0 1
8	1 0 0 0	1 1 1 0	1 0 0 0
9	1 0 0 1	1 1 1 1	1 1 1 1

In all the cases only ten combinations are utilized to represent the decimal digits. The remaining six combinations are illegal. However, they may be utilized for error detectionpurposes.

Example30: Consider, for example, the representation of the decimal number 16.85 in Binary Coded Decimal code (BCD) with weights 8421.

$(16.85)_{10} = (00010110\ .\ 10000101)$

1 6 8 5

Here, each decimal digit is represented by four binary bits.

2.8.3 Gray Codes

There are many applications in which it is desirable to have a code in which the adjacent codes differ only in one bit. Such codes are called Unit distance Codes. “Gray code” is the most popular example of unit distance code.The gray code is a cyclic code. Gray code cannot be used for arithmetic operation. The 3-bit and 4-bit gray codesare listed in Table

Table 8: Gray codes

Decimal	3-bit Gray	4-bit Gray
0	000	0000
1	001	0001
2	011	0011
3	010	0010
4	110	0110
5	111	0111
6	101	0101
7	100	0100
8	-	1100
9	-	1101
10	-	1111
11	-	1110
12	-	1010
13	-	1011
14	-	1001
15	-	1000

Conversion of binary to gray code:

1. Write the MSB of binary number as MSB of Gray code.
2. To find the next bit of gray code, Exclusive OR the MSB of binary number with successive bit, result will be the next bit of gray code.
3. Repeat the same process till the LSB is found.

Example 31: Convert Binary 1001 into gray code

Binary 1 ex-or 0 ex-or 0 ex-0r 1

Gray 1 1 0 1

Example Convert 1101011 into gray code

Binary = 1101011

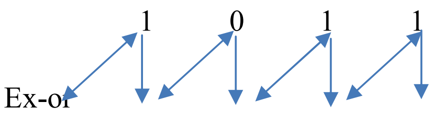
Gray = 1011110

Conversion of gray to binary:

1. Write the MSB of gray number as MSB of binary number.
2. To find the next bit of binary, Exclusive OR the left side bit of binary number with successive gray bit, result will be the next bit of binary number.
3. Repeat the same process till the LSB is found.

Example 32: Convert 1011 into binary number

Gray = 1 0 1 1



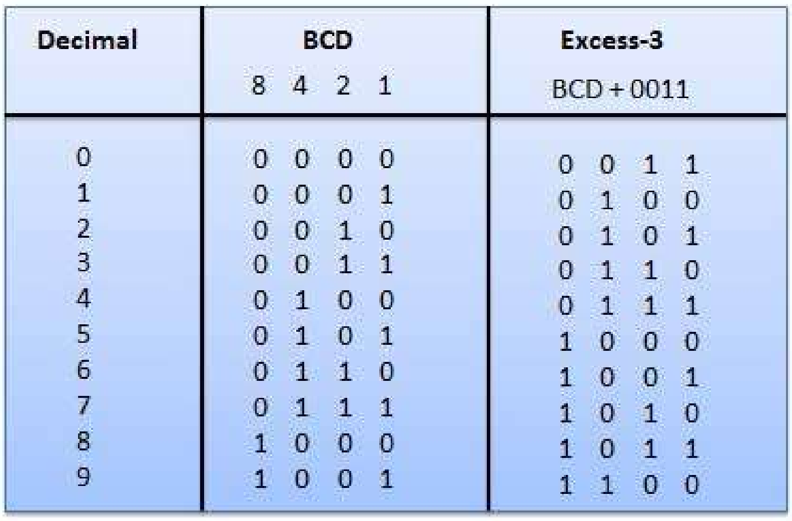
Binary = 1 1 0 1 0

2.8.4 Excess-3 code

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding $(0011)_2$ or $(3)_{10}$ to each code word in 8421. The excess-3 codes are obtained as follows −

$$\begin{tikzcd}\text{Decimal Number} \arrow[r] & \text{8421 BCD} \arrow[r, "\text{Add}", "0011"'] & \text{Excess-3}\end{tikzcd}$$

Table 9: Excess-3 codes



Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Practice Questions

Multiple Choice Questions

1. Which number system is understood easily by thecomputer?

   (a) Binary (b)Decimal (c)Octal (d) Hexadecimal

2. How many symbols are used in the decimal number system? (a)

   2 (b)8 (c)10 (d)16

3. What does $(10)_{16}$ represent in decimal number system?

   (a)10 (b) 0A (c) 16 (d) 15

4. How many bits have to be grouped together to convert the binary number to its corresponding octalnumber?

   (a) 2 (b) 3 (c) 4 (d) 5

5. Which bit represents the sign bit in a signed numbersystem?

   a. Left mostbit
   b. Right mostbit
   c. Left centre
   d. Rightcentre

6. The ones complement of 1010is

   (a) 1100 (b)0101 (c)0111 (d) 1011

7. How many bits are required to cover the numbers from +63 to -63 in one's complementrepresentation?

   (a) 6 (b) 7 (c) 8 (d) 9

Problems

1. Perform the following number systemconversions:

   (a) $10110111_2 = ?_{10}$ (b) $5674_{10} = ?_2$
   (c) $10011100_2 = ?_8$ (d) $2453_8 = ?_2$
   (e) $111100010_2 = ?_{16}$ (f) $68934_{10} = ?_2$
   (g) $10101.001_2 = ?_{10}$ (h) $6FAB7_{16} =$

$?_{10}$

(i) $11101.101_2 = ?_8$ (j) $56238_{16} = ?_2$

2. Convert the following hexadecimal numbers into binary and octalnumbers

(a) 78AD	(b)DA643	(c)EDC8
(d)3245	(e)68912	(f)AF4D

3. Convert the following octal numbers into binary and hexadecimalnumbers

(a) 7643	(b) 2643	(c) 1034
(d) 3245	(e) 6712	(f) 7512

4. Convert the following numbers intobinary:

(a) 1236_{10}	(b) 2349_{10}	(c) 345.275_{10}
(d) 4567_8	(e) 45.65_8	(f) 145.23_8
(g) $ADF5_{16}$	(h) $AD.F3_{16}$	(i) $12.DA_{16}$

5. Whatistherangeofunsigneddecimalvaluesthatcanberepresentedby8bits?
6. Whatistherangeofsigneddecimalvaluesthatcanberepresentedby8bits?
7. Howmanybitsarerequiredtorepresentdecimalvaluesrangingfrom75to-75?
8. Represent each of the following values as a 6-bit signed binary number in one's complement and two's complementforms.

   (a)28 (b)-21 (c)-5 (d)-13

9. Determine the decimal equivalent of two's complement numbers given below:

   (a) 1010101 (b)0111011 (c) 11100010

10. Solve each of the following 5-bit subtraction problems using 1's complement representation.

    a. $00110_2 – 00101_2$
    b. $01100_2 - 01010_2$
    c. $00100_2 - 00101_2$
    d. $01001_2 - 01011_2$
    e. $00011_2 - 01100_2$
    f. $00110_2 - 01001_2$

11. For each of the following problems convert the subtrahend to an 8-bit 2's complement representation and subtract. Leave your answer in binary.
    (a) $01111111_2 - 78_{10}$
    (b) $00110010_2 - 123_{10}$
    (c) $01001001_2 - 111_{10}$
    (d) $00000111_2 - 35_{10}$

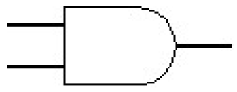
CHAPTER 3

3.1 Introduction to basic Logic Gate

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output and at any given moment, every terminal is in one of the two binary conditions either low (0) or high (1), represented by different voltage levels. The logic state of a terminal change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V). There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and NOR.

3.1.2 The AND Gate

The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true." Otherwise, the output is "false."

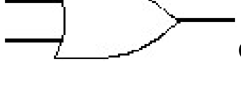


AND gate

Input 1	**Input 2**	**Output**
0	0	0
0	1	0
1	0	0
1	1	1

3.1.2 The OR Gate

The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false."

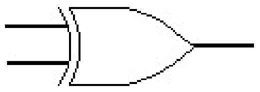


OR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

3.1.3 The XOR Gate

The *XOR* (E*xclusive-OR*) *gate* acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

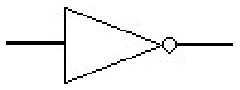


XOR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

3.1.4 The NOT Gate

A logical *inverter*, sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.

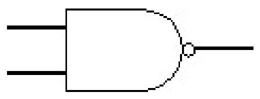


Inverter or NOT gate

Input	Output
1	0
0	1

3.1.5 The NAND Gate

The *NAND gate* operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

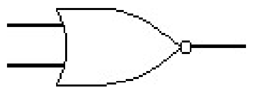


NAND gate

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0

3.1.6 The NOR Gate

The *NOR gate* is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

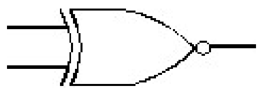


NOR gate

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

3.1.7 The XNOR Gate

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same and “false" if the inputs are different.



XNOR gate

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	1

Using combinations of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital integrated circuits (ICs). As IC technology advances, the required physical volume for each individual logic gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.

3.2 Universal Gate

A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families. In fact, an AND gate is typically implemented as a NAND gate followed by an inverter not the other way around!! Likewise, an OR gate is typically implemented as a NOR gate followed by an inverter not the other way around!!

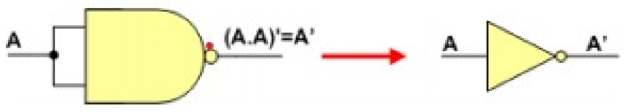
NAND Gate as a Universal Gate:

To prove that any Boolean function can be implemented using only NAND gates, lets draw AND, OR, and NOT operations using only these gates.

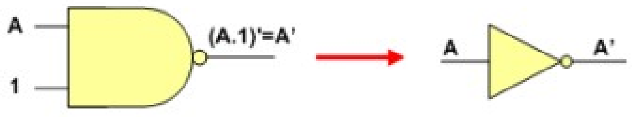
Implementing an Inverter Using only NAND Gate:

The figure shows two ways in which a NAND gate can be used as an inverter (NOT gate).

1. All NAND input pins connect to the input signal A gives an output A'.

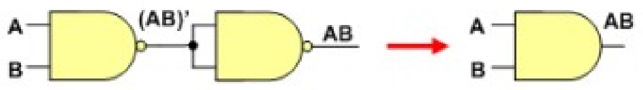


2. One NAND input pin is connected to the input signal A while all other input pins are connected to logic 1. The output will be A'.



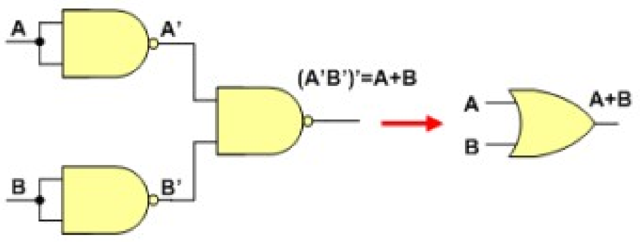
Implementing AND Using only NAND Gates

An AND gate can be replaced by NAND gates as shown in the figure (The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter).



Implementing OR Using only NAND Gates

An OR gate can be replaced by NAND gates as shown in the figure (The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverters.)



Thus, the NAND is a universal gate since it can be implemented the AND, OR and NOT functions.

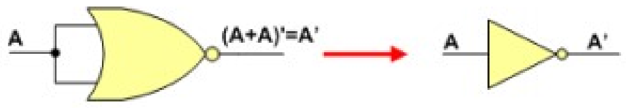
NOR Gate is a Universal Gate:

To prove that any Boolean function can be implemented using only NOR gates, lets show that the AND, OR, and NOT operations can be performed using only these gates.

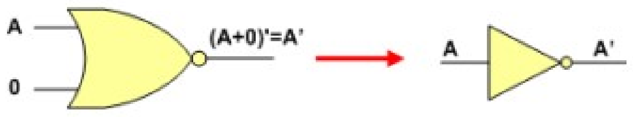
Implementing an Inverter Using only NOR Gate

The figure shows two ways in which a NOR gate can be used as an inverter (NOT gate).

1. All NOR input pins connect to the input signal A gives an output A'.

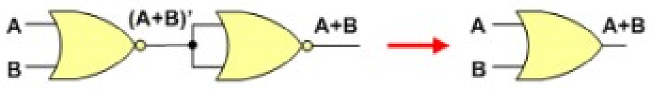


2. One NOR input pin is connected to the input signal A while all other input pins are connected to logic 0. The output will be A’.



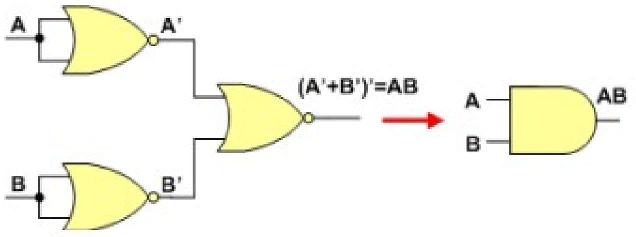
Implementing OR Using only NOR Gates

An OR gate can be replaced by NOR gates as shown in the figure (The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter)



Implementing AND Using only NOR Gates

An AND gate can be replaced by NOR gates as shown in the figure (The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters)

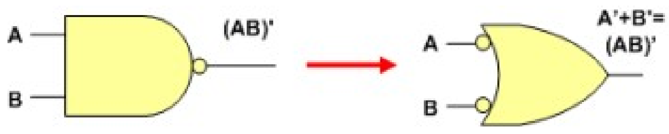


Thus, the NOR gate is a universal gate since it can implement the AND, OR and NOT functions.

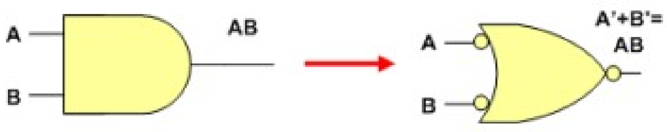
3.3 Equivalent Gates:

The shown figure summarizes important cases of gate equivalence. Note that bubbles indicate a complement operation (inverter).

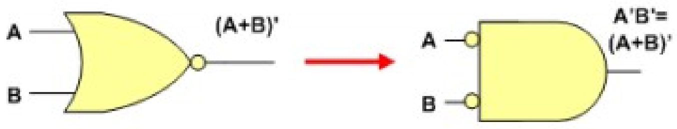
A NAND gate is equivalent to an inverted-input OR gate.



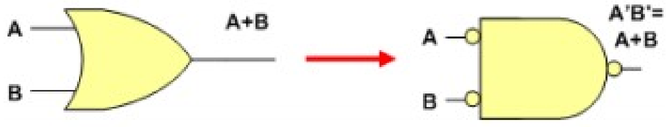
An AND gate is equivalent to an inverted-input NOR gate.



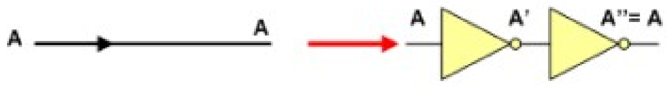
A NOR gate is equivalent to an inverted-input AND gate.



An OR gate is equivalent to an inverted-input NAND gate.



Two NOT gates in series are same as a buffer because they cancel each other as A'' =



3.4 Boolean Algebra

The most obvious way to simplify Boolean expressions is to manipulate them in the same way as normal algebraic expressions are manipulated. With regards to logic relations in digital forms, a set of rules for symbolic manipulation is needed in order to solve for the unknowns.

A set of rules formulated by the English mathematician *George Boole* describe certain propositions whose outcome would be either *true or false*. With regard to digital logic, these rules are used to describe circuits whose state can be either, *1 (true) or 0 (false)*. In order to fully understand this, the relation between the AND gate, OR gate and NOT gate operations should be appreciated. A number of rules can be derived as:

- **X = 0 or X = 1**
- **0 . 0 = 0**
- **1 + 1 = 1**
- **0 + 0 = 0**
- **1 . 1 = 1**
- **1 . 0 = 0 . 1 = 0**
- **1 + 0 = 0 + 1 = 1**

3.4.1 Laws of Boolean Algebra

Table 2 shows the basic Boolean laws. Note that every law has two expressions, (a) and (b). This is known as *duality*. These are obtained by changing every AND(.) to OR(+), every OR(+) to AND(.) and all 1's to 0's and vice-versa.
It has become conventional to drop the **.** (AND symbol) i.e. A**.**B is written as AB.

1 : Commutative Law

(a) $A + B = B + A$

(b) $A B = B A$

2 : Associate Law

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

(b) $(A B) C = A (B C)$

3 : Distributive Law

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

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

4 : Identity Law

(a) $A + A = A$

(b) $A A = A$

5 :

(a) $AB + A\overline{B} = A$

(b) $(A+B)(A+\overline{B}) = A$

6 : Redundancy Law

(a) $A + A\,B = A$

(b) $A\,(A + B) = A$

7 :

(a) $0 + A = A$

(b) $0\,A = 0$

8 :

(a) $1 + A = 1$

(b) $1\,A = A$

9:

(a) $\overline{A} + A = 1$

(b) $\overline{A}\,A = 0$

10:

(a) $A + \overline{A}\,B = A + B$

(b) $A\,(\overline{A} + B) = A\,B$

11: De Morgan's Theorem

(a) $(\overline{A + B}) = \overline{A}\;\overline{B}$

(b) $(\overline{A\,B}) = \overline{A} + \overline{B}$

Table 2: Boolean Laws

Examples

Prove 10: (a) $A + \overline{A}\,B = A + B$

(1)Algebraically:

$$\begin{aligned} A + \overline{A}\,B &= A\,1 + \overline{A}\,B && \text{T7(a)}\\ &= A\,(1 + B) + \overline{A}\,B && \text{T7(c)}\\ &= A + A\,B + \overline{A}\,B && \text{T3(a)}\\ &= A + B\,(A + \overline{A}) && \text{T3(a)}\\ &= A + B && \text{T(8)} \end{aligned}$$

(2) Using the truth table:

A	B	$A + B$	$A\,B$	$A + A\,B$
0	0	**0**	0	**0**
0	1	**1**	1	**1**
1	0	**1**	0	**1**
1	1	**1**	0	**1**

Using the laws given above, complicated expressions can be simplified.

$$Z = (A + \overline{B} + \overline{C})(A + \overline{B}C)$$
$$Z = AA + A\overline{B}C + A\overline{B} + \overline{B}\overline{B}C + A\overline{C} + \overline{B}C\overline{C}$$
$$Z = A(1 + \overline{B}C + \overline{B} + \overline{C}) + \overline{B}C + \overline{B}C\overline{C} \quad \text{from laws T8b and Tb}$$
$$Z = A + \overline{B}C \quad \text{from laws T8a, T8b and T9b}$$

3.5 CANONICAL FORM

A Boolean function can be represented in one of the following methods:

- Truth Table
- Logic Circuit
- SOP expression
- POS expression

SOP (Sum of product expression): It is the sum of all product term of each input combination for which the output is high (logic '1'). Each product term of SOP expression is known as minterm. Minterms are ORed to get the Boolean expression.For a given truth table, minterm and SOP expression can be written as

2-variable Truth Table:

Index	xy	Minterm	F	Symbol
0	00	x' y'	1	m0
1	01	x 'y	1	m1
2	10	x y'	1	m2
3	11	x y	1	m3

3.5.1 SOP representation:

F (x, y) = ∑ (m0, m1, m2, m3) ----------------------------minterm presentation

F (x, y) = x' y' + x 'y + x y'+ x y ---------------------SOP presentation

3-variable Truth Table:

X	Y	Z	Term	F	Symbol
0	0	0	x'y'z'	1	m0
0	0	1	x'y'z	1	m1
0	1	0	x'yz'	1	m2
0	1	1	x'yz	1	m3
1	0	0	xy'z'	1	m4
1	0	1	xy'z	1	m5
1	1	0	xyz'	1	m6
1	1	1	xyz	1	m7

SOP representation:

F (x, y, z) = ∑ (m0, m1, m2, m3, m4, m5, m6, m7)

F (x, y, z) = x'y'z'+ x'y'z +-x'yz'+ x'yz+ xy'z'+ xy'z + xyz'+xyz

Example1. Represent the given truth table in SOP form

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

F = AB'C+ABC

F (x, y, z) = ∑ (m5, m7)

3.5.2 Product of Sum expression(POS):

3.5.2 Product of Sum expression(POS): It is the product of all sum term of each input combination for which the output is low (logic ‘0’). Each sum term of POS expression is known as maxterm (M). Maxterms are ANDed to get the Boolean expression. For a given truth table, maxterm and POS expression can be written as

3-variable Truth Table:

X	Y	Z	Term	F	Symbol
0	0	0	X+Y+Z	0	M0
0	0	1	X+Y+Z’	0	M1
0	1	0	X+Y’+Z	0	M2
0	1	1	X+Y’+Z’	0	M3
1	0	0	X’+Y+Z	0	M4
1	0	1	X’+Y+Z’	0	M5
1	1	0	X’+Y’+Z	0	M6
1	1	1	X’+Y’+Z’	0	M7

POS representation:

F (X,Y,Z) = (X+Y+Z) (X+Y+Z’) (X+Y’+Z) (X+Y’+Z’)(X’+Y+Z)(X’+Y+Z’)(X’+Y’+Z)(X’+Y’+Z’) -------- POS Form

F (X,Y,Z) = π(M0.M1.M2.M3.M4.m5.M6.M7) -------Maxterms

Example2. Represent the given truth table in POS form

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1

1	1	0	0
1	1	1	1

F (X, Y, Z) = π (M1.M4.M6)

F (X, Y, Z) = (X+Y+Z') (X'+Y+Z) (X'+Y'+Z)

Types of SOP and POS form: Canonical and Standard

Standard form: In standard form It is not necessary for each product term (SOP) and sum term (POS) to contain all literals. Therefore, the product terms and sum terms may or may not be the min terms and max terms respectively. Standard form is the reduced Boolean expression.

Standard SOP expression:

F (A, B, C) = A'B+B'C

The given function has three literals, in first term literal C is missing. Similarly, in second term literal A is missing.

Standard POS expression:

F (A, B, C) = (A' +C). (B'+C)

The given function has three literals, in first term literal B is missing. Similarly, in second term literal A is missing.

Canonical form: In this form both the product and sum terms contain all literals. Canonical expressions are simplified into standard form. There can be situations where that it is impossible to simplify the canonical form. Then the canonical and standard forms are similar.

Canonical SOP expression:

F (A, B, C) = A'BC+AB'C

The given function has three literals, both the product terms contain all three literals.

Canonical POS expression:

F (A, B, C) = (A' +B+C). (A'+B'+C)

The given function has three literals, both the sum terms contain all three literals.

Standard forms can be converted into canonical form.

Conversion of SOP into canonical form:

The normal SOP form function can be converted to canonical SOP form by using the Boolean algebraic law, (A + A’ = 1) and by following the below steps.

Step 1: Multiply the non-standard product term with the sum of its missing variable and its complement, which results in 2 product terms

Step 2: Repeat the step 1, until all resulting product terms contain all variables

Step3: Discard the duplicate term (if exist).

By these two steps we can convert the SOP function into canonical SOP function. In this process, for each missing variable in the function, the number of product terms will double.

Example 3: Convert into canonical SOP function F = x y + x z + y z

Sol: F = x y + x z + y z

= x y (z + z’) + x (y + y’) z + (x + x’) y z

= x y z + x y z’ + x y z + x y’ z + x y z + x’ y z

= x y z + x y z’ + x y’ z + x’ y z

The canonical SOP form is F = x y z + x y z’ + x y’ z + x’ y z

Example4: Convert into canonical SOP form

E (X, Y, Z) = Y’ + X’Z’

= Y’(X+X’) (Z+Z’) + X’Z’(Y+Y’)

= (XY’+X’Y’) (Z+Z’) + X’YZ’+ X’Y’Z’

= XY’Z+X’Y’Z+XY’Z’+X’Y’Z’+ X’YZ’+X’Y’Z’

= = XY’Z+X’Y’Z+XY’Z’+X’Y’Z’+ X’YZ’

Conversion of POS form Canonical POS form

We can include all the variables in each product term of the POS form equation, which doesn’t have all the variables by converting into standard POS form. The normal POS form function can be converted to standard POS form by using the Boolean algebraic law, (A . A’ = 0) and by following the below steps.

Step 1: Add each non-standard sum term to the product of its missing variable and its complement, which results in 2 sum terms.

Step 2: Applying Boolean algebraic law, A + BC = (A + B) (A + C)

Step 3: By repeating the step 1, until all resulting sum terms contain all variables.

Step4: Discard the duplicate sum terms (if exists)

By these steps we can convert the POS function into standard POS function.

Example5: Convert into canonical POS form

F = (A’ + B + C) (B’ + C + D’) (A + B’ + C’ + D)

Solution:

In the first term, the variable D or D’ is missing, so we add D.D’ = 1 to it. Then

(A’ + B + C + D.D’) = (A’ + B + C + D). (A’ + B + C + D’)

Similarly, in the second term, the variable A or A’ is missing, so we add A*A’ = 1 to it. Then

(B’ + C + D’ + A*A’) = (A + B’ + C + D’) * (A’ + B’ + C + D’)

The third term is already in the standard form, as it has all the variables. Now the standard POS form equation of the function is

F = (A’ + B + C + D) (A’ + B + C + D’) (A + B’ + C + D’) (A’ + B’ + C + D’) (A + B’ + C’ + D)

Conversion of SOP form to POS form

To convert the SOP form into POS form, first we should change the Σ to Π and then write the numeric indexes of missing variables of the given Boolean function.

Example6:The given SOP function is

F (A, B, C) = ∑ m (0, 2, 3, 5, 7) = A’ B’ C’ + A B’ C’ + A B’ C + ABC’ + ABC

is written in POS form by following these steps.

Step 1: changing the operational sign ∑ to Π

Step 2: writing the missing indexes of the terms, 001 (1), 100 (4) and 110 (6). Now write the sum form for these noted terms.

001 = (A + B + C) 100 = (A + B’ + C’) 110 = (A + B’ + C’)

Writing down the new equation in the form of POS form,

F (A, B, C) = ΠM (1, 4, 6) = (A + B + C) (A + B’ + C’)(A + B’ + C’)

Conversion of POS form to SOP form

To convert the POS form into SOP form, first we should change the Π to Σ and then write the numeric indexes of missing variables of the given Boolean function.

Example7: The POS function F (A, B, C) = ΠM (2, 3, 5) = A B’ C’ + A B’ C + ABC’ is written in SOP form by

Step 1: changing the operational sign to Σ

Step 2: writing the missing indexes of the terms, 000 (0), 001 (1), 100 (4), 110 (6), and 111 (7). Now write the product form for these noted terms.

000 = A’ B’ C’ 001 = A’ B’C 100 = A B’C’

110 = A B C’ 111 = A B C

Writing down the new equation in the form of SOP form,

F = Σ A, B, C (0, 1, 4, 6, 7) = (A’ B’ C’) + (A’ B’ C) + (A B’ C’) + (A BC’) + (A B C)

Example8: Express the Boolean function F = XY + X′Z in a product of maxterm form.

Solution:

First, convert the function (Product terms) into OR (Sum) terms using the distributive law:

F = XY + X′Z = (XY + X’) (XY + Z)

= (X + X’) (Y + X’) (X + Z) (Y + Z)

= (X’+Y) (X+Z) (Y+Z)

The function has three variables: X, Y and Z. Each OR term is missing one variable; therefore

Term1: (X’+ Y + Z. Z’) = (X’+ Y + Z) (X’+ Y + Z’)

Term2: (X+ Y.Y’ + Z) = (X+ Y + Z) (X+ Y’ + Z)

Term3: (XX’+ Y + Z) = (X’+ Y + Z) (X+ Y + Z)

Combining all the terms and removing those that appear more than once

F = (X+Y+Z)(X+Y’+Z)(X’+Y+Z)(X’+Y+Z) = M0M2M4M5 = ΠM (0,2,4,5)

3.6 K-Map

Karnaugh maps (K-maps) are graphical method for the simplification of Boolean functions. K- Maps consist of cells.

Where no. of cells =2n as n being no. of input variables

K-map cells are arranged so that adjacent cells differ in only one bit position (based on Gray code).

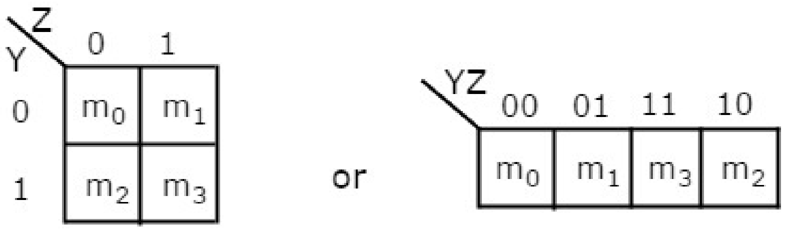
Karnaugh Map Advantages

1. Minimization can be done more systematically
2. Much simpler to find minimum solutions

3. Easier to see what is happening (graphical)
4. Almost always used instead of Boolean minimization.

2-Variable K-Map

The number of cells in 2 variables K-map is four, since the number of variables is two. The following figure shows 2 variables K-Map.

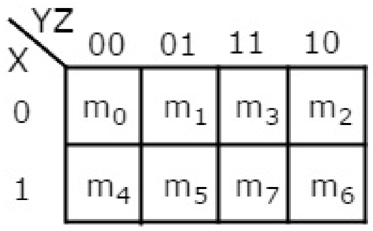


There is only one possibility of grouping 4 adjacent min terms.

The possible combinations of grouping 2 adjacent min terms are {(m0, m1), (m2, m3), (m0, m2) and (m1, m3)}.

3-Variable K-Map

The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows 3 variable K-Map.



There is only one possibility of grouping 8 adjacent min terms.

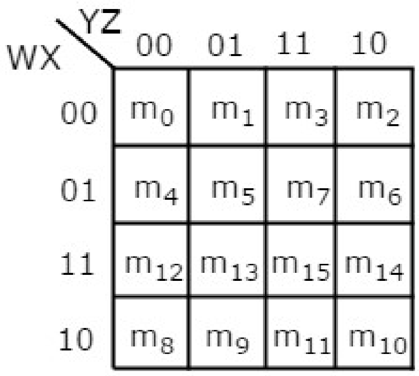
The possible combinations of grouping 4 adjacent min terms are {(m0, m1, m3, m2), (m4, m5, m7, m6), (m0, m1, m4, m5), (m1, m3, m5, m7), (m3, m2, m7, m6) and (m2, m0, m6, m4)}.

The possible combinations of grouping 2 adjacent min terms are {(m0, m1), (m1, m3), (m3, m2), (m2, m0), (m4, m5), (m5, m7), (m7, m6), (m6, m4), (m0, m4), (m1, m5), (m3, m7) and (m2, m6)}.

If x=0, then 3 variable K-map becomes 2 variable K-map.

Four Variable K-Map

The number of cells in 4 variables K-map is sixteen, since the number of variables is four. The following figure shows 4 variables K-Map.



WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

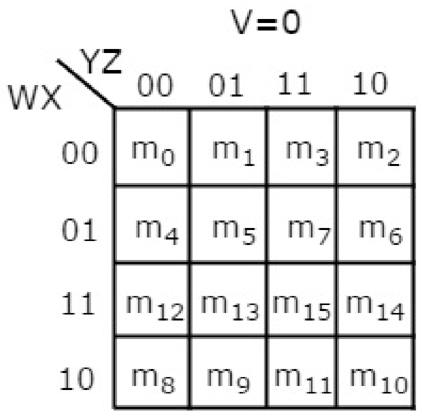
There is only one possibility of grouping 16 adjacent min terms.

Let R1, R2, R3 and R4 represents the min terms of first row, second row, third row and fourth row respectively. Similarly, C1, C2, C3 and C4 represents the min terms of first column, second column, third column and fourth column respectively. The possible combinations of grouping 8 adjacent min terms are {(R1, R2), (R2, R3), (R3, R4), (R4, R1), (C1, C2), (C2, C3), (C3, C4), (C4, C1)}.

If w=0, then 4 variable K-map becomes 3 variable K-map.

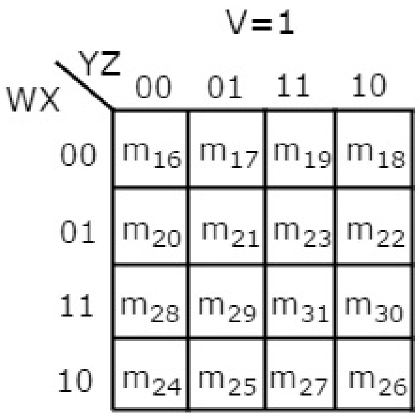
Five Variable K-Map

The number of cells in 5 variable K-map is thirty-two, since the number of variables is 5. The following figure shows 5 variable K-Map.



V=0

WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}



V=1

WX \ YZ	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

There is only one possibility of grouping 32 adjacent min terms.

There are two possibilities of grouping 16 adjacent min terms. i.e., grouping of min terms from m0 to m15 and m16 to m31.

If v=0, then 5 variable K-map becomes 4 variable K-map.

In the above all K-maps, we used exclusively the min terms notation. Similarly, you can use exclusively the Max terms notation.

Minimization of Boolean Functions using K-Maps

If we consider the combination of inputs for which the Boolean function is ‘1’, then we will get the Boolean function, which is in standard sum of products form after simplifying the K-map.

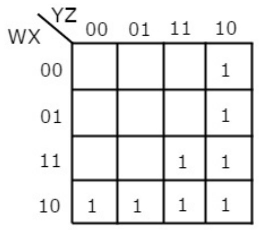
Similarly, if we consider the combination of inputs for which the Boolean function is ‘0’, then we will get the Boolean function, which is in standard product of sums form after simplifying the K-map.

Rules for simplifying K-maps in order to get standard sum of product form.

1. Select the respective K-map based on the number of variables present in the Boolean function.
2. If the Boolean function is given as sum of min terms form, then place the ones at respective min term cells in the K-map. If the Boolean function is given as sum of products form, then place the ones in all possible cells of K-map for which the given product terms are valid.
3. Check for the possibilities of grouping maximum number of adjacent ones. It should be powers of two. Start from highest power of two and upto least power of two. Highest power is equal to the number of variables considered in K-map and least power is zero.
4. Each grouping will give either a literal or one product term. It is known as prime implicant. The prime implicant is said to be essential prime implicant, if at least single ‘1’ is not covered with any other groupings but only that grouping covers.
5. Note down all the prime implicants and essential prime implicants. The simplified Boolean function contains all essential prime implicants and only the required prime implicants.
6. If outputs are not defined for some combination of inputs, then those output values will be represented with don’t care symbol ‘x’. That means, we can consider them as either ‘0’ or ‘1’.
7. If don’t care terms also present, then place don’t care ‘x’ in the respective cells of K-map. Consider only the don’t cares ‘x’ that are helpful for grouping maximum number of adjacent ones. In those cases, treat the don’t care value as ‘1’.

Example 9: simplify the following Boolean function, f= WX’Y’ + WY + W’YZ’ using K-map.

The given Boolean function is in sum of products form. It is having 4 variables W, X, Y & Z. So, we require 4 variable K-map. The 4 variable K-map with ones corresponding to the given product terms is shown in the following figure.



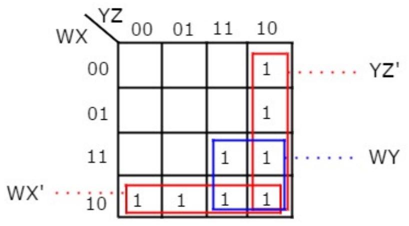
Here, 1s are placed in the following cells of K-map.

The cells, which are common to the intersection of Row 4 and columns 1 & 2 are corresponding to the product term, WX'Y'.

The cells, which are common to the intersection of Rows 3 & 4 and columns 3 & 4 are corresponding to the product term, WY.

The cells, which are common to the intersection of Rows 1 & 2 and column 4 are corresponding to the product term, W'YZ'.

There are no possibilities of grouping either 16 adjacent ones or 8 adjacent ones. There are three possibilities of grouping 4 adjacent ones. After these three groupings, there is no single one left as ungrouped. So, we no need to check for grouping of 2 adjacent ones. The 4 variable K-map with these three groupings is shown in the following figure.



Here, we got three prime implicants WX', WY & YZ'. All these prime implicants are essential because of following reasons.

Two ones (m8 & m9) of fourth row grouping are not covered by any other groupings. Only fourth row grouping covers those two ones.

Single one (m15) of square shape grouping is not covered by any other groupings. Only the square shape grouping covers that one.

Two ones (m2 & m6) of fourth column grouping are not covered by any other groupings. Only fourth column grouping covers those two ones.

Therefore, the simplified Boolean function is

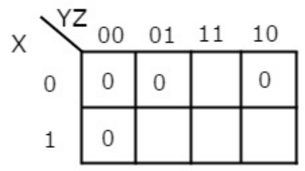
f = WX' + WY + YZ'

Rules for simplifying K-maps in order to get standard product of sum form.

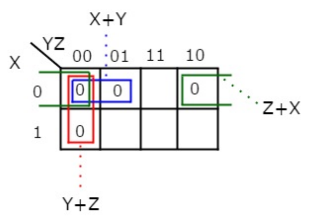
1. Select the respective K-map based on the number of variables present in the Boolean function.
2. If the Boolean function is given as product of Max terms form, then place the zeroes at respective Max term cells in the K-map. If the Boolean function is given as product of sums form, then place the zeroes in all possible cells of K-map for which the given sum terms are valid.
3. Check for the possibilities of grouping maximum number of adjacent zeroes. It should be powers of two. Start from highest power of two and up to least power of two. Highest power is equal to the number of variables considered in K-map and least power is zero.
4. Each grouping will give either a literal or one sum term. It is known as prime implicant. The prime implicant is said to be essential prime implicant, if at least single '0' is not covered with any other groupings but only that grouping covers.
5. Note down all the prime implicants and essential prime implicants. The simplified Boolean function contains all essential prime implicants and only the required prime implicants.
6. If don't care terms also present, then place don't care 'x' in the respective cells of K-map. Consider only the don't cares 'x' that are helpful for grouping maximum number of adjacent zeroes. In those cases, treat the don't care value as '0'.

Example10. Simplify the following Boolean function, f(X, Y,Z)=∏M(0,1,2,4)

The given Boolean function is in product of Max terms form. It is having 3 variables X, Y & Z. So, we require 3 variable K-map. The given Max terms are M0, M1, M2 & M4. The 3 variable K-map with zeroes corresponding to the given Max terms is shown in the following figure.



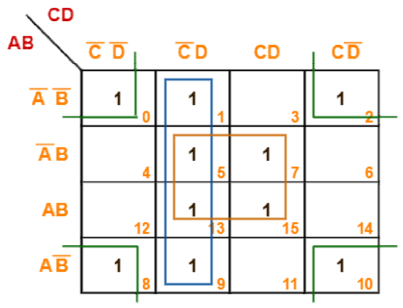
There are no possibilities of grouping either 8 adjacent zeroes or 4 adjacent zeroes. There are three possibilities of grouping 2 adjacent zeroes. After these three groupings, there is no single zero left as ungrouped. The 3 variable K-map with these three groupings is shown in the following figure.



Here, we got three prime implicants X + Y, Y + Z & Z + X. All these prime implicants are essential because one zero in each grouping is not covered by any other groupings except with their individual groupings.

Example11: Minimize the following Boolean function-F (A, B, C, D) = Σ m (0, 1, 2, 5, 7, 8, 9, 10, 13, 15)

Solution-

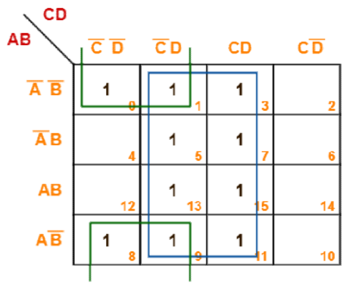


F(A, B, C, D)

= (A’B + AB)(C’D + CD) + (A’B’ + A’B + AB + AB’)C’D + (A’B’ + AB’)(C’D’ + CD’)

= BD + C’D + B’D’

Example12: Minimize the following Boolean function-F(A, B, C, D) = Σm(0, 1, 3, 5, 7, 8, 9, 11, 13, 15)



Now,F(A, B, C, D)

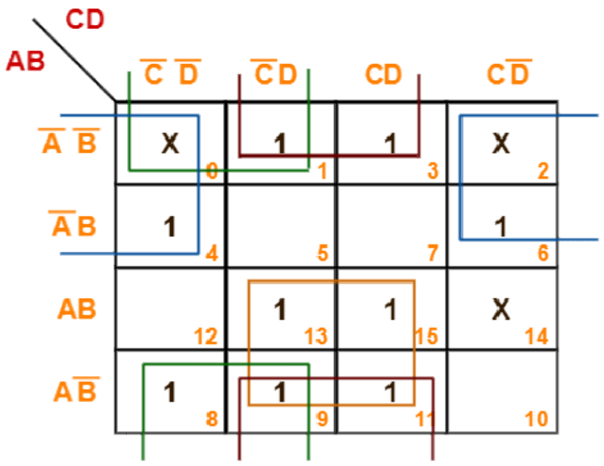
= (A’B’ + A’B + AB + AB’)(C’D + CD) + (A’B’ + AB’)(C’D’ + C’D)

= D + B’C’

Thus, minimized Boolean expression is-F(A, B, C, D) = B’C’ + D

Example13: Minimize the following Boolean function-F(A, B, C, D) = Σm(1, 3, 4, 6, 8, 9, 11, 13, 15) + Σd(0, 2, 14)

Solution-



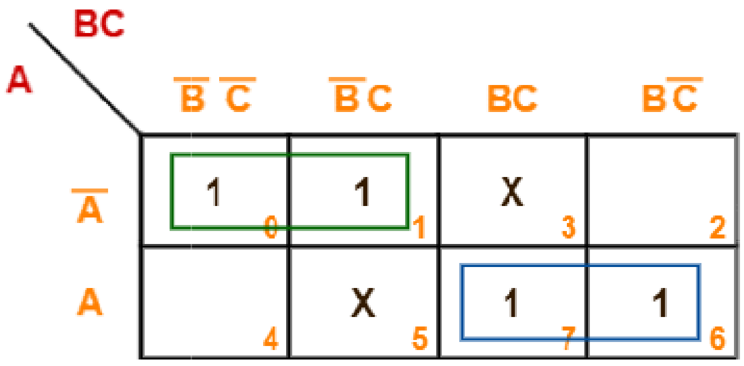
Now,F(A, B, C, D)

= (AB + AB’)(C’D + CD) + (A’B’ + AB’)(C’D + CD) + (A’B’ + AB’)(C’D’ + C’D) + (A’B’ + A’B)(C’D’ + CD’)

= AD + B’D + B’C’ + A’D’

Example14: Minimize the given Boolean function-F(A, B, C) = Σm(0, 1, 6, 7) + Σd(3, 5)

Solution-



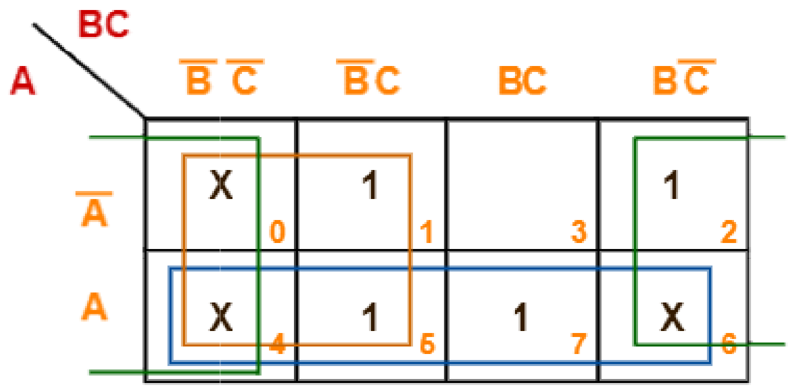
Now,F(A, B, C)= A'(B’C’ + B’C) + A(BC + BC’)

= A’B’ + AB

NOTE- It may be noted that there is no need of considering the quad group.This is because even if we consider that group, we will have to consider the other two duets.

Example15: Minimize the given function-F(A, B, C) = Σm(1, 2, 5, 7) + Σd(0, 4, 6)

Solution-

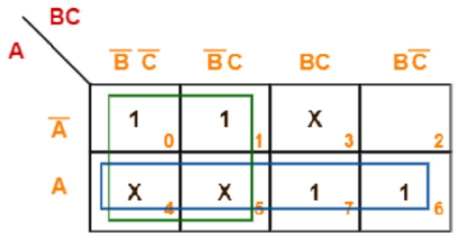


Now,F(A, B, C)= (A + A’)(B’C’ + B’C) + A(B’C’ + B’C + BC + BC’) + (A + A’)(B’C’ + BC’)

= B’ + A + C’

Example16: Minimize the Boolean function-F(A, B, C) = Σm(0, 1, 6, 7) + Σd(3, 4, 5)

Solution-

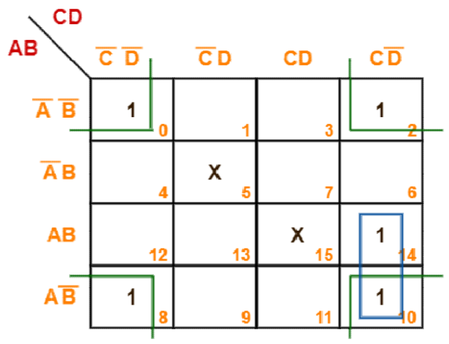


Now,F(A, B, C)= (A + A’)(B’C’ + B’C) + A(B’C’ + B’C + BC + BC’)

= B’ + A

Example17: Minimize the following Boolean function-F(A, B, C, D) = Σm(0, 2, 8, 10, 14) + Σd(5, 15)

Solution-

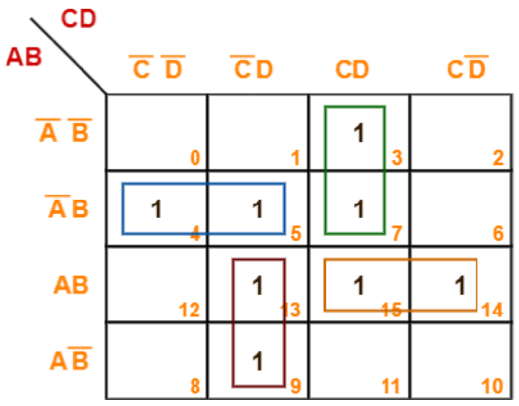


Now,F(A, B, C, D)= (AB + AB’)CD’ + (A’B’ + AB’)(C’D’ + CD’)

= ACD’ + B’D’

Example18: Minimize the function-F(A, B, C, D) = Σm(3, 4, 5, 7, 9, 13, 14, 15)

Solution-



Now,

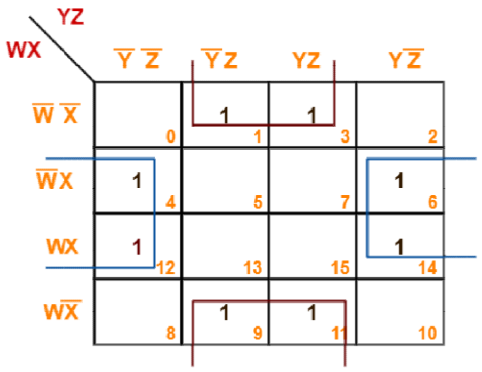
F(A, B, C, D)= A’B(C’D’ + C’D) + (A’B’ + A’B)(CD) + (AB + AB’)(C’D) + AB(CD + CD’)

= A’BC’ + A’CD + AC’D + ABC

It is important to note that we are not considering the quad group because we have to consider the duets anyhow.

Example19 :F(W, X, Y, Z) = Σm(1, 3, 4, 6, 9, 11, 12, 14). This function is independent ________ number of variables. Fill in the blank.

Solution-



F (W, X, Y, Z)= (W’X + WX) (Y’Z’ + YZ’) + (W’X’ + WX’) (Y’Z + YZ)

= XZ’ + X’Z

= X □ Z

Thus, minimized Boolean expression is-

F(W, X, Y, Z) = X □ Z

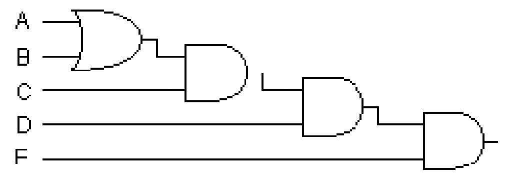
Clearly, the given Boolean function depends on only two variables X and Z.

Hence, it is independent of other two variables W and Y.

Question

1 Determine the values of A, B, C, and D that make the sum term $\overline{A}+B+\overline{C}+D$ equal to zero.

2 Derive the Boolean expression for the logic circuit shown below



3 For the SOP expression $A\overline{B}C+\overline{A}BC+AB\overline{C}$, how many 1s are in the truth table's output column?

4 How many gates would be required to implement the following Boolean expression before simplification? XY + X(X + Z) + Y(X + Z)

5 Determine the values of A, B, C, and D that make the product term $\overline{A}B\overline{C}D$ equal to 1.

6 What is the primary motivation for using Boolean algebra to simplify logic expressions?

7 How many gates would be required to implement the following Boolean expression after simplification? XY + X(X + Z) + Y(X + Z)

8 Consider the following Boolean function-

F(W, X, Y, Z) = Σm(1, 3, 4, 6, 9, 11, 12, 14)

This function is independent ________ number of variables. Fill in the blank.

9 Use Karnaugh maps to find the minimum-cost SOP and POS expressions for the function $f(x_1,...,x_4) = x_1x_3x_4 + x_3x_4 + x_1x_2x_4 + x_1x_2x_3x_4$ assuming that there are also don't-cares defined as D = d(9, 12, 14).

10 F(A,B,C,D)=π(3,5,7,8,10,11,12,13)

11 $F(A,B,C)=\pi(0,3,6,7)$

12 Simplify the given 3-variable Boolean equation by using k-map.

$$F = X'YZ + X'Y'Z + XYZ' + X'Y'Z' + XYZ + XY'Z'$$

13. Simplify the given 4-variable Boolean equation by using k-map. $F(W, X, Y, Z) = (1, 5, 12, 13)$