

Numbering system

Objective


- Understand the decimal, binary, octal and hexadecimal number systems.
- Convert from one number system into another.
- Apply arithmetic operations to binary numbers.
- Understand binary codes and alpha numeric codes.

1.1 Types of Numbering systems

A number system defines a set of values to represent quantity. There are many numbering systems used in the real world. Each of these systems has its base (or radix). The base means the number of symbols that are used in the system, and it is indicated by a subscript. The position of each digit in the number indicates the magnitude of the quantity represented and can be assigned as a weight. The weights are increased from right to left, beginning with power 0 of the base. While the fractional number weight start from left to right, starting with power -1, -2, -3...etc.

For example in decimal number:

$$\dots 10^3 10^2 10^1 10^0 \cdot 10^{-1} 10^{-2} 10^{-3} 10^{-4} \dots$$

Decimal point

1.1.1 Decimal systems

Peoples have familiarity with the decimal number system because they use it every day. The decimal number system has a base of 10. The ten symbols (digits) are 0 through 9. The weight of a digit, value, in the number is determined by its position.

Example A1: Find the decimal number of 286 as a sum of the values of each digits?

Solution

$$\begin{aligned} 286 &= (2 \times 10^2) + (8 \times 10^1) + (6 \times 10^0) \\ &= (2 \times 100) + (8 \times 10) + (6 \times 1) \\ &= 200 + 80 + 6 \\ &= (286)_{10} \end{aligned}$$

1.1.2 Binary system

The digital codes and binary number system are fundamental to computers. The system of binary number has a base of 2. It uses two symbols, 1 and 0. Each of the two digits is called a **bit** (binary digit). For example the number (1000110) is accepted in binary system, while the number (1100200) is not accepted because there is the digit 2. The counting in binary system start from 0 ,1 then take two digits 00,01,10,11 and three digit and so on.

In weighting of binary number, the least significant bit (LSB) has the weight $2^0=1$. The weight increases from LSB to most significant bit (MSB) by power of two for each digit. For example:

$$2^{n-1}, \dots, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, \dots, 2^{-n}$$

Binary point

1.1.3 Octal system

The octal number system has a base of 8. The symbols that use **are 0, 1, 2, 3, 4, 5, 6 and 7**. In other words, any number is not accepted in octal system if there is a digit

value more than 7. The counting in octal system starts from 0 and increments by 1 until 7, then the number contains two digits: $7+1=10$, $10+1=11$, $11+1=12$.

1.1.4 Hexadecimal system

The base of this system is 16. It is composed of 16 numeric and alphabetic characters. It consists of digit 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and letters A, B, C, D, E, F. The letters A, B, C, D, E and F are used to represent the digits 10, 11, 12, 13, 14 and 15.

Since the hexadecimal number is very convenient to represent groups of 4 bits, it is the most common system seen today in representing raw data computer data. Each single hexadecimal digit can be represented by four digits in a binary number. This makes conversion between binary and hexadecimal numbers easy, and hexadecimal can be represented with much fewer digits to write large binary numbers. When someone working with computers, it is common to find binary numbers with 8, 16 and may be 32 digits.

Writing a 16 or 32 bit binary number would be caused a quite tedious and error prone. So, by using the hexadecimal, one can write the numbers with fewer digits and much less probability of occurrence of the error.

Note:

For discrimination between numbers with different bases, the coefficients is enclosed in parentheses and write a subscript equal to base that is used.

$(8265)_{10}$, $(1092)_{10}$ → Decimal numbers

$(10110)_2$, $(100.01)_2$ → Binary numbers

$(2751)_8$, $(1763.56)_8$ → Octal numbers

$(A926)_{16}$, $(15B.CF)_{16}$ → Hexadecimal numbers

1.2 Number systems conversion

The systems number can be converted from one system to another system to be able to use the appropriate numbering system that suitable for a particular application, as discussions below.

1.2.1 Decimal number to other systems number conversion

The number may include two parts: integer part and fraction part. Each part in the number is converted from one system to other separately.

To convert a given integer part of decimal number to other number in systems, such as binary, octal, and hexadecimal is done by the following steps:

- (a) Divide the decimal number by r , where r is the base of the other system.
- (b) Write the remainder (one of base symbols) at the rightmost position.
- (c) Repeat the procedure of dividing the quotient by r until the quotient becomes 0 and keep track the remainder at each step of division.
- (d) Concatenate the remainders together starting from the last one to give the equivalent other system number.

Table A1 illustrates the conversion of decimal numbers to their equivalent binary, octal and hexadecimal numbers.

Table A1: The equivalents in numbering system

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Example A2: Convert the number $(72)_{10}$ to binary, octal, hexadecimal numbers?

Solution

The solution is done as following:

Decimal to binary	Decimal to octal	Decimal to hexadecimal
$72 \div 2 = 36 \text{ r } 0$ $36 \div 2 = 18 \text{ r } 0$ $18 \div 2 = 9 \text{ r } 0$ $9 \div 2 = 4 \text{ r } 1$ $4 \div 2 = 2 \text{ r } 0$ $2 \div 2 = 1 \text{ r } 0$ $1 \div 2 = 0 \text{ r } 1$ $(1001000)_2$	$72 \div 8 = 9 \text{ r } 0$ $9 \div 8 = 1 \text{ r } 1$ $1 \div 8 = 0 \text{ r } 1$ $(110)_8$	$72 \div 16 = 4 \text{ r } 8$ $4 \div 16 = 0 \text{ r } 4$ $(48)_{16}$

The resulted number is reading the remainder from down to top and it is writing from left to right.

Example A3: Convert the number $(195)_{10}$ to binary, octal, hexadecimal number?

Solution

$$(195)_{10} = (11000011)_2$$

$$(195)_{10} = (303)_8$$

$$(195)_{10} = (C3)_{16}$$

To convert the decimal fraction to other system number is done by the following steps:

- a) Multiply the decimal fraction by the base of the other system, r .
- b) If the generated integer is non-zero, take the non-zero integer otherwise record 0.
- c) Ignore the non-zero integer and repeat the above steps until the value of the fraction becomes 0.
- d) According to the occurrence, write down the number. The first digit represents the most significant digit.

Example A4: Convert $(0.375)_{10}$ to binary, octal, hexadecimal number fraction?

Solution

To convert the decimal fraction to binary fraction, repeat the multiplication by the base of binary system, the value of $r = 2$, until to the desire number of digit in the fraction part or stop when the fraction part is zero.

Binary fraction
$0.375 \times 2 = 0.75 \quad C \ 0$
$0.75 \times 2 = 1.50 \quad C \ 1$
$0.50 \times 2 = 1.00 \quad C \ 1$
$(0.011)_2$

To octal fraction: $0.375 \times 8 = 3.000$; Carry 3

$$(0.375)_{10} = (0.3)_8$$

To hexadecimal fraction: $0.375 \times 16 = 6.000$

$$(0.375)_{10} = (0.6)_{16}$$

1.2.2 Other systems to decimal system conversion

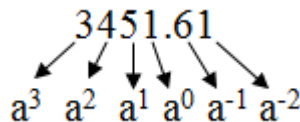
Conversion a number from other numbering systems to decimal number is done by:

- Put the weight of each digit of the decimal number.
- Multiply each digit of the number in the other system by its base power to positional weight.
- Find the sum of multiplying. The positional weight is not to be taken into account where the digit is zero.

Remember the mathematical rule that $n^0 = 1$, or any number raised to the zero power is equal to 1. The value of decimal number is calculated as follow:

$$\text{value} = a_n r^n + a_{n-1} r^{n-1} + \dots + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2}$$

Where r is the base of other numbering system, a is the value of the digit in the other system, and n is the order of digit. For example, the number in octal system $(3451.61)_8$ can found as follow, where $r = 8$ (since the number in octal):



$$\begin{aligned} &\text{The value of } (3451.61)_8 \\ &= a_3 r^3 + a_2 r^2 + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} \\ &= 3 \times 8^3 + 4 \times 8^2 + 5 \times 8^1 + 1 \times 8^0 + 6 \times 8^{-1} + 1 \times 8^{-2} \\ &= 3 \times 512 + 4 \times 64 + 5 \times 8 + 1 \times 1 + 6 \times 0.125 + 1 \times 0.0156 \\ &= 1536 + 256 + 40 + 1 + 0.75 + 0.0156 \\ &= (1833.7656)_{10} \end{aligned}$$

Example A5: Convert the following numbers to decimal number?

a) $(1010.01)_2$

b) $(24)_8$

c) $(1A2)_{16}$

Solution

a) The weight of each digit : $2^3 \ 2^2 \ 2^1 \ 2^0 \ 2^{-1} \ 2^{-2}$

Binary number : 1 0 1 0 . 0 1

$$(1010.01)_2 = 8 + 0 + 2 + 0 + 0 + 0.25$$

$$= (10.25)_{10}$$

b) $(27.3)_8 = 2 \times 8^1 + 7 \times 8^0 + 3 \times (1/8)$

$$= 2 \times 8 + 7 \times 1 + 3 \times 0.125$$

$$= 16 + 7 + 0.375$$

$$= (23.375)_{10}$$

c) $(1A2)_{16} = 1 \times 16^2 + 10 \times 16^1 + 2 \times 16^0$

$$= 1 \times 256 + 10 \times 16 + 2$$

$$= 256 + 160 + 2 = (418)_{10}$$

The binary system is the simplest system, so it can be converted to decimal number by a weighting number. The weight or value of a binary digit increases from right to left by a power of 2. The right most bit is the least

significant bit (LSB) in binary number has a weight of $2^0 = 1$.

The decimal value of any binary number is equal to the sum of weighting of all bits that are 1. For the fraction the weighting begins from $2^{-1} = 0.5$ and decreases from left to right, as shown in below example.

Example A6: Convert $(101101.01)_2$ to decimal number?

Solution

Weight in power of 2:	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
	32	16	8	4	2	1	0.5	0.25
Binary number	: 1	0	1	1	0	1	. 0	1

$$(101101.01)_2 = 32 + 8 + 4 + 1 + 0.25 = (45.25)_{10}$$

We can get the same result by using another way to find the number by summing the weights that corresponding to values=1, as shown in the following table:

Weight	64	32	16	8	4	2	1	1/2	1/4	1/8
Value		1	0	1	1	0	1	0	1	

 ↑ ↑ ↑ ↑ ↑

Example A7: What is the smallest and largest decimal number that can be represented in binary six bits?

Solution

The smallest binary number is of six bits is $(100000)_2$.

The decimal value of this number is $(32)_{10}$.

The largest binary number is of six bits is $(111111)_2$.

The decimal value of this number is $(63)_{10}$.

1.2.3 Binary to octal conversion

The conversions include partition the binary number into 3-bits group, starting at the left digit of fractional separator and replace each group with equivalent octal digit. For the fraction part, the partition is starting from left to right. The weight of group digits are 4, 2, 1 (the weight 1 for LSB digit).

Example A8: Convert $(101101.11)_2$ into octal?

$$101101 = \underbrace{101}_5 \underbrace{101}_5 . \underbrace{110}_6$$

Thus $(101101)_2 = (55.6)_8$.

1.2.4 Binary to hexadecimal conversion.

For the integer part, the conversions include partition the binary number into group, each group contains 4-bits. The partition starts from the left digit of fractional point and replace each group with equivalent hexadecimal symbol. For the fraction part, the partition is starting from left to right. The weight of group digits are 8, 4, 2, 1 (the weight 1 for LSB digit).

Example A9:

Convert the binary number $(10011101101.10101001)_2$ to hexadecimal number?

Solution:

$$\begin{array}{ccccccc}
 & & & & \text{Fractional point} & & \\
 & & & & \downarrow & & \\
 4 & 2 & 1 & 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 \\
 \hline
 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
 \hline
 = & 4 & & E & & D & & A & & & 9 & = (4ED.A9)_{16}
 \end{array}$$

Note: If the last group contains less than 4 digits, we add 0's to the right in integer part, and 0s in left fraction part. For example $(10.101)_2 = (0010.1010)_2 = (2.A)_{16}$

Example A10: Convert to hexadecimal number the binary number $(10110110)_2$?

Solution

Split into groups for 4 digits : 1011 0110
Conversion each group to hexadecimal digit: B 6
The hexadecimal number : $(B6)_{16}$

1.2.5 Octal to binary conversion

The conversion is done by replace each octal digit with equivalent three binary digits. These groups of three bits are concatenated together to form the binary number. The simplest conversion by using the weight 4, 2, and 1 for each octal digit.

Example A11: Convert $(2701.4)_8$ to binary number?

Solution

First, convert each digit alone by convert it to three binary digits and then concatenate them to represent the binary number.

Octal digit	Weight		
	4	2	1
2	0	1	0
7	1	1	1
0	0	0	0
1	0	0	1
4	1	0	0

$$(2701.4)_8 = (010111000001.100)_2$$

1.2.6 Hexadecimal to binary conversion

The conversion of hexadecimal number to its equivalent binary number is done by convert each hexadecimal digit to its equivalent 4 bits binary number. Then the hexadecimal digits are concatenated to constitute the hexadecimal number.

Example A12: Convert $(D26.1A)_{16}$ to binary number?

Solution:

The hexadecimal number: D 2 6 1 A
 ┌───┐ ┌───┐ ┌───┐ ┌───┐ ┌───┐
 1101 0010 0110 0001 1010
 └───┘ └───┘ └───┘ └───┘ └───┘
 ↑ ↑ ↑ ↑ ↑
So, $(D26.1A)_{16} = (110100100110.00011010)_2$

1.2.7 Conversion between hexadecimal and octal

The conversion process includes the following steps:

- Split each digit in the octal to 3 –bit binary.
- Merge all the 3-bit binary numbers.
- Collect them in 4-bit binary group, starting from MSB to LSB.
- Convert each 4-bit group to their hexadecimal digit.

Example A13: Convert $(56)_8$ to hexadecimal number?

Solution

The number $(56)_8$ is an octal number, so each digit will be replaced by three binary digits.

Octal number : 5 6

Binary number: 101 110

Merging the 3-bit binary block, they become 101110.

Grouping them in 4 bit binary form-

The binary number: 0010 1110

The hexadecimal : 2 E

The result of $(55)_8 = (2E)_{16}$.

The conversion from hexadecimal number to octal number is done by reverse the above steps.

Example A14: convert $(E27.9)_{16}$ to octal number?

Solution

Hexadecimal number: E 2 7 . 9

Four bit for each digit: 1110 0010 0111 . 1001

Binary number : 111000100111.1001

Divide into 3 bits groups: 111 000 100 111.100 100

Value of each group : 7 0 4 7 4 4

Octal number : $(7047.44)_8$

Example A15: Convert $(5B1.7)_{16}$ to octal number?

Solution

$$(5B1.7)_{16} = (010110110001.0111)_2$$

$$=(010110110001.0111)_2$$

$$\begin{array}{ccccccc} = & 010 & 110 & 110 & 001 & . & 011 & 100 \\ & \boxed{} & \boxed{} & \boxed{} & \boxed{} & & \boxed{} & \boxed{} \\ & 2 & 6 & 6 & 1 & & 3 & 4 \\ = & (2661.34)_8 \end{array}$$

Example A16: Convert the octal number $(713.5)_8$ to hexadecimal number?

Solution

Octal number : 7 1 3 . 5

3 binary digit : 111 001 011 101

Binary number: $(111001011.101)_2$

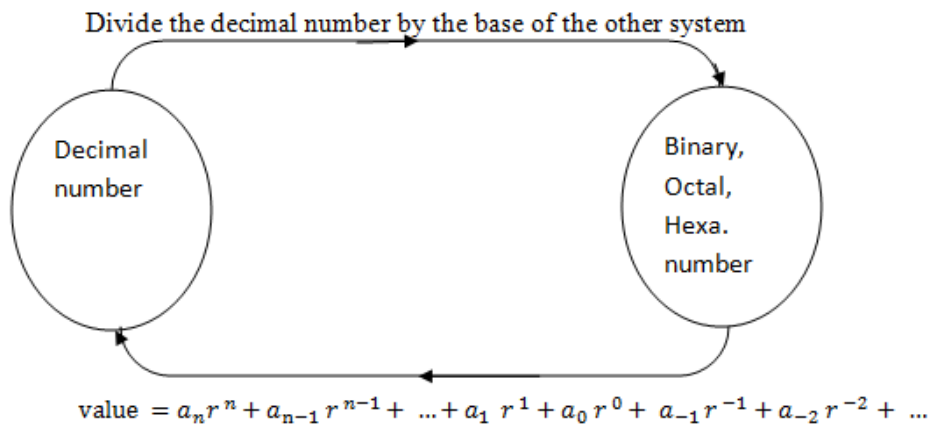
Four bits group : 0001 1100 1011 . 1010

Value of each group: 1 D B A

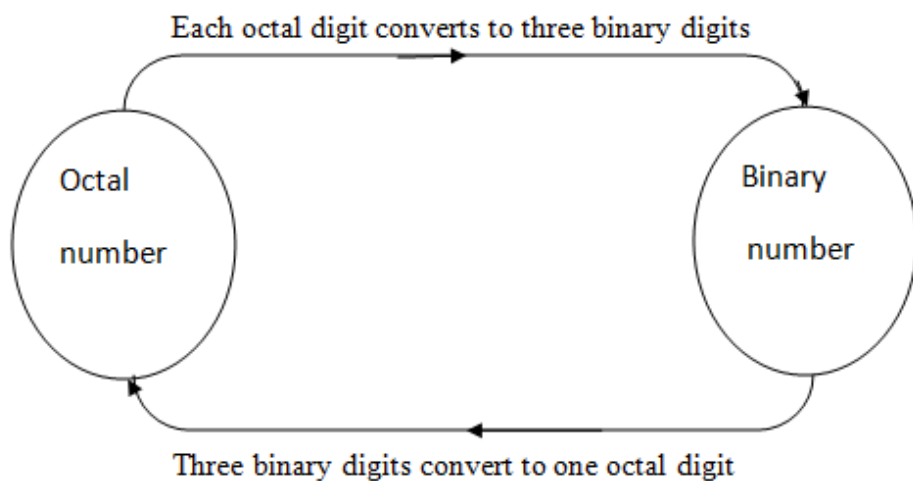
Hexadecimal number: $(1DB.A)_{16}$

The conversions between binary, octal, decimal and hexadecimal system are illustrated in following figure (A1).

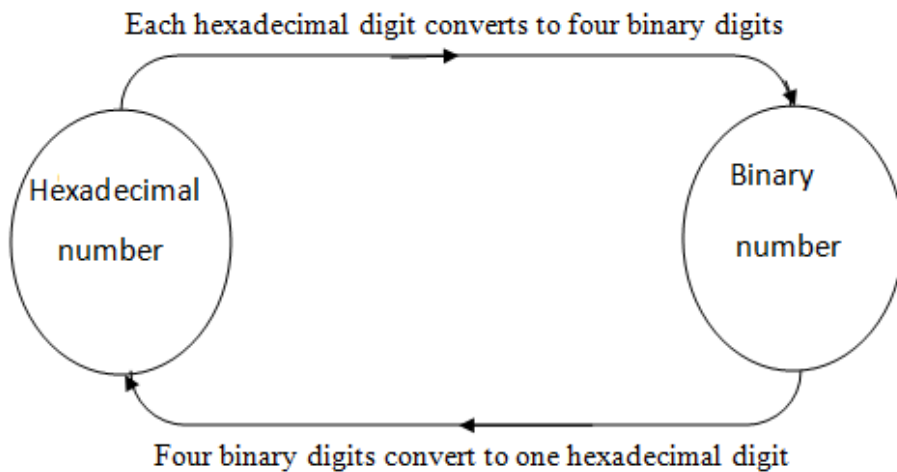
The conversions between numbering systems can be summarized in the figure A1.



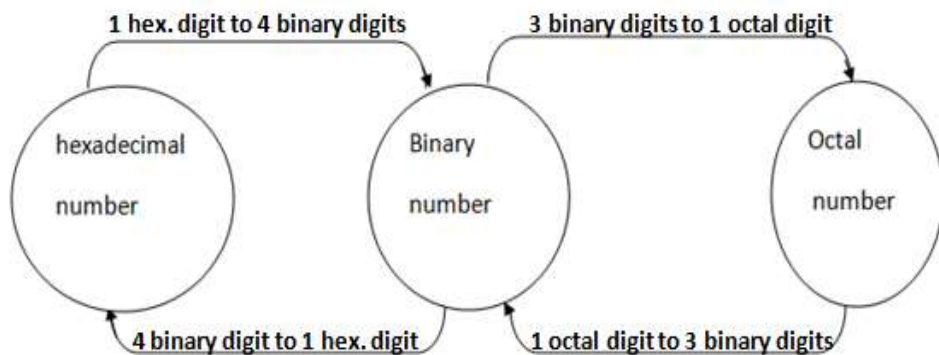
(a) Conversion between decimal system and other systems



(b) Conversion between octal system and binary system



(c) Conversion between hexadecimal system and binary system



(d) Conversion between hexadecimal system and octal system

Figure (A1): Conversions between numbering systems

3. Binary Arithmetic

Arithmetic operations with number in numbering system with base r have the same rules in decimal numbers.

1.3.1 Binary Addition

The binary addition is performed by using the following the binary table.

$$0 + 0 = 0 \rightarrow \text{Sum} = 0, \text{carry} = 0$$

$$0 + 1 = 1 \rightarrow \text{Sum} = 1, \text{carry} = 0$$

$$1 + 0 = 1 \rightarrow \text{Sum} = 1, \text{carry} = 0$$

$$1 + 1 = 0 \rightarrow \text{Sum} = 0, \text{carry} = 1$$

The carry is found in the same way as in decimal operation. Since the 1 is the largest digit in the system of binary. If the sum is greater than 1 need that a digit be considered overflow, therefore is divided by the base.

Example A17: Perform the following addition in binary numbering system?

a) $(111)_2 + (110)_2$

b) $(1010)_2 + (1101)_2$

c) $(11.01)_2 + (101.11)_2$

Solution

11	1	111 1 ← Carry
111	1010	11.01
110	1101	101.11
-----	-----	-----
1001	10111	1001.00

1.3.2 Binary Subtraction

The binary subtraction is performed like a decimal number, except borrow is equal $2 = (10)_2$ rather than $(10)_{10}$ since the borrow value is equal to the base. The results of binary subtraction are:

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$0 - 1 = 1$ with a borrow of 1 is equivalent to

$$2 - 1 = 1$$

Example A18: Perform $(111)_2 - (010)_2$?

Solution


$$\begin{array}{r} 111 - \\ 010 \\ \hline 101 \end{array}$$

Example A19: Perform $(110.01)_2 - (100.10)_2$?

Solution

$$\begin{array}{r} 110.01 \\ 100.10 \\ \hline 001.11 \end{array}$$

need borrow



The second bit from the right need borrow 1 from the next digit to the left making it value 10 (means 2). Since the third digit is 0, so it borrow from the forth digit. So the third column becomes 1. So, the second digit can able to borrow from the third. The second digit become 10 (it means 2 in decimal system).

$$10 - 1 = 1.$$

1.3.3 Binary multiplication

The binary multiplication is easier that decimal number multiplication. The result is either 0 or 1. The binary multiplication table is shown as following:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Example A20: Perform the following?

a) $(1011)_2 \times (110)_2$

b) $(101.01)_2 \times (11.01)_2$

Solution

a)

$$\begin{array}{r} 1011 \text{ x} \\ 110 \\ \hline 0000 \\ 1011 \\ 1011 \\ \hline 1000010 \end{array}$$

$$(1011)_2 \times (110)_2 = (1000010)_2$$

b)

$$\begin{array}{r} 101.01 \text{ x} \\ 11.01 \\ \hline 10101 \\ 00000 \\ 10101 \\ 10101 \\ \hline 10001.0001 \end{array}$$

$$(101.01)_2 \times (11.01)_2 = (1001.0001)_2$$

1.3.4 Binary division

The binary division of two digits is as follows.

$$0 \div 1 = 0$$

$$1 \div 1 = 1$$

As in the decimal system division by zero is meaningless.

Example A21: perform

a) $(1100)_2 \div (11)_2$?

b) $(1001)_2 \div (10)_2$?

$$\begin{array}{r} 100 \\ 11 \overline{) 1100} \\ \underline{11} \\ 0 \end{array}$$

$$\begin{array}{r} 100.1 \\ 10 \overline{) 1001} \\ \underline{10} \\ 0010 \\ \underline{10} \\ 0 \end{array}$$

So, the results of the division are:

a) $(1100)_2 \div (11)_2 = (100)_2$;

b) $(1001)_2 \div (10)_2 = (100.1)_2$

1.4. Complements

Complements are used in digital computer for simplifying the subtraction operations and for logical manipulations. By the complements, the subtraction of

one number from another is done using only addition of positive numbers

There are two types of complements in numbering systems: $(r'-1)$'s complement and r 's complements, which are called true complement and radix-1 complement. In binary system, the complements are 1's complement and 2's complement are used.

1.4.1 r 's complement

For a positive number X with base r and an integer part contains n digits, the r 's complement of X is $r^n - X$ for $X \neq 0$. In other word, the r 's complement can be found keep all significant zeros without changed, subtract the first non-zero digit in least significant from r , and then subtract from $(r-1)$ all other digit in higher significant.

In the decimal system the r 's complement is referred to 10's complement, while in the binary system is referred to 2's complement. There are following examples of r 's complements.

The 10's complement of $(4925)_{10}$ is $10^4 - 4925 = 6075$.

The digits number in the 4925 is four, so $n=4$.

The 10's complement of $(0.273)_{10}$ is $1 - 0.273 = 0.927$

The digits number in the 0.273 is zero, so $n=0$.

Example A22: find the r 's complement of the numbers $(615)_8$, $(1A280)_{16}$, $(100100)_2$?

Solution

$7\ 7\ 8$	$15\ 15\ 15\ 16$	$1\ 1\ 1\ 2$
$6\ 1\ 5 -$	$1\ A\ 2\ 8\ 0 -$	$1\ 0\ 0\ 1\ 0\ 0 -$
<hr style="width: 100%;"/>	<hr style="width: 100%;"/>	<hr style="width: 100%;"/>
$1\ 6\ 3$	$E\ 5\ D\ 8\ 0$	$0\ 1\ 1\ 1\ 0\ 0$

1.4.2 $(r-1)$'s complement

For a base r and positive number X with fraction part of m digits and an integer part of n digits, the $(r-1)$'s complement of X is $r^n - r^m - X$. In other word, the r 's complement can be found by subtracting all other digits

Example A23:

Find the $(r-1)$'s complement for the numbers $(1001010)_2$, $(3982)_{10}$, $(4530)_8$, $(6BE00)_{16}$?

Solution

$$\begin{array}{r}
 1111111 \\
 1001010 - \\
 \hline
 0110101
 \end{array}
 \quad
 \begin{array}{r}
 9999 \\
 3982 - \\
 \hline
 6017
 \end{array}
 \quad
 \begin{array}{r}
 7777 \\
 4530 - \\
 \hline
 3247
 \end{array}
 \quad
 \begin{array}{r}
 1515151515 \\
 6B E 0 0 - \\
 \hline
 94 1FF
 \end{array}$$

Example A24: Find the 9's complement and 8's complements for the number $(6250)_9$?

Solution

The 9's complement can be found using r's complement. It is (2640).

The 8's complement can be found using (r-1)'s complement. It is (2638).

1.4.3 (1)'s complement

In binary system, the r's complement is called 1's complement. To find 1's complement of a binary number, the 1 is change to 0 and 0 change to 1. In other word, each digit of the binary number is subtracted from 1.

Example A25: Find the 1's complement for the numbers $(1010)_2$, $(101101)_2$, $(00011)_2$ and $(110110)_2$?

Solution

The following table includes the 1's complement of each the above number .

Binary number	1's Complement
1010	0101
101101	010010
00011	11100
110110	001001

1.4.4 (1)'s complement subtraction:

To subtract $M-N$, perform the following:

1. Add M number to the 1's complement of the N number.
2. According to the result of step 1, if there is end carry, we add 1 to the digit in least significant (end-around carry). If there is not end carry, obtain the $(r-1)$'s complement of the number getting in step 1 and put a negative sign in front of it.

Example A26: Perform $6 - 3$ using 1's complements method?

Solution

$$\begin{array}{rcl}
 7 & \rightarrow 111 & \text{-----} > 111 + \\
 3 & \rightarrow 011 & \text{1's complement } 100 \\
 \hline
 3 & & 1011 + \\
 & & \text{L} \rightarrow 1 \\
 & & \text{----} \\
 & & 100 \leftarrow \text{result}
 \end{array}$$

If there is no carry after add 1 and the leftmost digit is 1, this indicates that the result is a negative and number will be in its 1's complement form. So we take the complement number to get the final result.

$$\begin{aligned}
 100 & \rightarrow 011 \text{ (after taking 1's complement)} \\
 & = (3)_{10}.
 \end{aligned}$$

Example A27: Perform $8 - 12$ using 1's complements method?

$$\begin{array}{rcl}
 8 & \rightarrow 1000 & \text{-----} > 1000 + \\
 12 & \rightarrow 1100 & \text{1's complement } 0011 \\
 \hline
 -4 & & 1011
 \end{array}$$

Taking the 1's complement to the sum result: 0100

The final result is -4 .

1.4.5 2's Complement

The representation of 2's complement is used because it minimize the complexity in the arithmetic-logic unit hardware in the CPU of the computers. Using the representation of 2's complement, all operations of the arithmetic can be executed using the same hardware whether the number is signed or unsigned. The performing of bit operations are similar; the difference comes from the interpretation of the digits, whether the value is considered to be signed or unsigned value. The 2's complement may obtain in many ways, such as:

1. Add 1 to the digit in the least significant of the 1's complement.

The 2's complement = 1's complement + 1

Example: 10110 \rightarrow 01001 1's complement

$$\begin{array}{r} 1 + \\ \hline 01010 \quad \text{2's complement} \end{array}$$

2. Keep all the least significant positions with values 0's, the first position with value 1 is unchanged, and change the values of the rest positions, 1's to 0's and all 0's to 1.

Example: 1011010 no changing
 ↓ ↓ 2's complement
 0100110

3. Using the form $r^n - N$ for $N \neq 0$, and 0 for $N=0$.

Where n is the number of digits.

Example A28: The 2's complement is $2^6 - (101100)$
 $= (1000000)_2 - (101100)_2 = (010100)_2$

The following table includes some examples about numbers with their 1's and 2's complement.

Binary Number	1's complement	2's complement
1010	0101	0110
0101	1010	1011
1001	0110	0111
0001	1110	1111

1.4.6 2's Complement Subtraction

The subtraction in 2's complement of two positive numbers ($M - N$) with base 2 is done as following:

1. Keep the first number without change.
2. Write the second number in the 2's complement form.
3. Perform the addition of the two numbers.
4. If found a carry, discard it and the sum (remaining part) will be the result, which is positive.
5. If there is no carry, perform the 2's complement of the sum and put negative sign before it, which become negative result.

Example A29: Subtract using 2's complement?

a) $(1000)_2$ from $(1010)_2$

b) $(12)_{10}$ from $(5)_{10}$?

Solution

a) $1010 \text{ -----} > 1010 +$

$1000 \text{ 2's complement } 1000$

$\underline{10010}$

The result is $(0010)_2$.

$$\begin{array}{rcl} \text{b) } 5 & 0101 & \text{-----} > 0101 + \\ 12 & 1100 \text{ 2's complement } 0100 & \\ & & \hline & & 1001 \end{array}$$

Take the 2's complement: - 0111

The result is - 7.

1-5 .1 Signed number

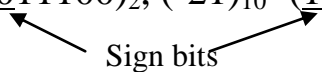
Digital systems have ability to handle the negative and positive numbers. The signed binary number contains the sign and magnitude information. The sign refers to whether the number is negative or positive, it is the leftmost bit in sign number. The 0 value of the sign bit indicates that the number is positive, while 1 is negative. The magnitude represent the number value.

There are three forms of representation the sign binary number:

1. Sign and magnitude: The leftmost bit of the number represents the sign bit. The rest bits represent the magnitude of digits. The magnitude digits of

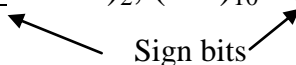
numbers are the same for both negative and positive number. For example, $(+21)_{10} = (\underline{0}10101)_2$, $(-21)_{10} = (\underline{1}10101)_2$.

2. 1's complement: The positive number represent in the same way in sign and magnitude form. While the negative number is founded by performing the 1's complement of the corresponding binary number. For example

$$(+28)_{10} = (\underline{0}11100)_2, (-21)_{10} = (\underline{1}00011)_2.$$


The diagram shows two arrows pointing from the underlined sign bits of the binary representations above to the text "Sign bits". One arrow points from the '0' in $(\underline{0}11100)_2$ to the text, and the other points from the '1' in $(\underline{1}00011)_2$ to the text.

3. 2's complement: The positive number represent in the same way in sign and magnitude form. While the negative number is founded by taking the 2's complement of the corresponding binary number. For example,

$$(+28)_{10} = (\underline{0}11100)_2, (-21)_{10} = (\underline{1}00100)_2.$$


The diagram shows two arrows pointing from the underlined sign bits of the binary representations above to the text "Sign bits". One arrow points from the '0' in $(\underline{0}11100)_2$ to the text, and the other points from the '1' in $(\underline{1}00100)_2$ to the text.

Example A30: Represent using 8 bits binary number the numbers $(+39)_{10}$ and $(-39)_{10}$ using sign and magnitude, 1's complement and 2's complement?

Solution

Number	Sign and magnitude	1's complement	2's complement
+39	<u>0</u> 0100111	<u>0</u> 0100111	<u>0</u> 0100111
-39	<u>1</u> 0100111	<u>1</u> 1011000	<u>1</u> 1011001

The diagram illustrates the conversion of decimal numbers to binary and their complements. For +39, the sign bit is 0, leading to a 1's complement of 0 and a 2's complement of 0. For -39, the sign bit is 1, leading to a 1's complement of 1 and a 2's complement of 1.

1.5.1 Floating point number

The floating point number includes two parts with their sign. The mantissa is the part that represents the magnitude of the number and its value between 0 and 1. The exponent is the part that represents the number of places to be moved. For example, the number 3705100 in the decimal number can be written as 0.37051×10^7 .

1.6 Binary codes

A binary number of n digit may be represented by n binary circuit elements, each having an output signal equivalent to a 0 or 1. Digital systems represent and process not only binary number, but also many other

discrete element of information. Any discrete element of information among group quantities can be represented by a binary code. There are many numbers of binary codes such as: binary coded decimal, excess-3, (8, 4, -2, -1) codes.

1.6.1 Binary coded decimal (BCD)

In digital system, it is possible to represent decimal numbers by encoding each digit in binary form in different codes, one of them is BCD. BCD is a numeric coding system used to encode each digit of a decimal number to a 4-bit binary form with weights 8,4,2,1. There are only ten codes in the BCD codes. The BCD code 0110 is equal to 6 because $0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 = 6$.

Only 10 from 16 patterns are used in BCD. the remaining 6 patterns not occur in logic operation. They treated as don't care condition in the design process. BCD provides a format that is convenient when numerical information is to be displayed on a simple

digit-oriented display. The drawback of BCD is the complexity of the circuit that performs arithmetic operations and wasted 6 possibilities.

Example A31: Convert the following numbers to BCD code: $(136)_{10}$, $(491)_{10}$, $(101101)_2$, $(2A3)_{16}$?

The solution:

$$(136)_{10} = \overbrace{(0001)}^1 \overbrace{0011}^3 \overbrace{0110}^6)_{\text{BCD}}$$

$$(491)_{10} = (0100 \ 1001 \ 0001)_{\text{BCD}}$$

To convert the binary, octal or hexadecimal number to BCD code, we must convert the number to decimal system firstly, and then convert the decimal number to BCD code.

$$(101101)_2 = (45)_{10} = (0100 \ 0101)_{\text{BCD}}$$

$$(2A3)_{16} = (675)_{10} = (0110 \ 0111 \ 0101)_{\text{BCD}}$$

Example A32: convert the BCD code (1001 0101 0000 0100) to decimal number?

BCD code: 1001 0101 0000 0100

Decimal digit: 9 5 0 4

The decimal number = $(9504)_{10}$

1.6.2 Excess-3 code

The excess-3 is unweighted code. Excess-3 code obtains from the corresponding values of BCD after the addition of 3 (adding 3 for each decimal digit). For example, the excess-3 of the digit 8 is 1011. Table A2 includes the excess-3 of the decimal digits.

1.6.3 Other codes

It is possible to use a negative weights for a decimal code such as 8, 4, -2, -1. For example, the code of the digit $(57)_{10}$ is (1011 1001).

Or code with weight 2, 4, 2, 1. For example, the code of digit $(57)_{10}$ is (0101 1101).

There is a difference between conversion to binary number and conversion to binary coding although the result is a series of bits. In binary coding, each digit has its unique representation, and there is a space between the representations of adjacent digits. For example, convert 17 to binary number yields 10001, while to BCD code yields 0001 0011.

The table A2 illustrates the some binary codes for decimal digits.

Table A2: binary codes of decimal system

Decimal digit	BCD code	Excess-3 code	8, 4, -2, -1 code
0	0000	0011	0000
1	0001	0100	0111
2	0010	0101	0110
3	0011	0110	0101
4	0100	0111	0100
5	0101	1000	1011
6	0110	1001	1010
7	0111	1010	1001
8	1000	1011	1000
9	1001	1100	1111

Example A34: convert the number $(193)_{10}$ to

- (a) Binary number.
- (b) BCD code.
- (c) Excess-3 code
- (d) 8, 4, -2, -1 code?

Solution

(a) $(193)_{10} = (11000001)_2$

(b) $(193)_{10} = (0001 \ 1001 \ 0011)$ in BCD code.

(c) $(193)_{10} = (0100 \ 1100 \ 0110)$ in excess-3 code.

(d) $(193)_{10} = (0111 \ 1111 \ 0101)$ in 8,4,-2,-1 code.

Addition in BCD

To add two BCD codes, use the following steps:

1. Add the two BCD numbers like addition in binary system rules.
2. If the fourth digit, MSB, is less or equal to 9, it is a accepted in BCD code.
3. If the fourth digit, MSB, is greater than 9, or a carry out of the fourth digit, it is an unaccepted in BCD code. We add 6 (0110) to the sum of addition operation in order to skip the six invalid codes and return the code to the weight 8,4,2,1. If a carry results when 6 is added, add the carry to the next group.

Example A33: Add the following BCD codes?

- a) (0101) + (0001)
- b) (0011 0101) + (0010 0010)
- c) (1001) + (1000)
- d) (0110 0101) + (0010 0101)
- e) (1000 0111) + (0101 0011)

Solution

$$\begin{array}{rcl}
 \text{a)} & \begin{array}{r} 0101 \\ 0001 + \\ \hline 0110 \end{array} & \begin{array}{l} \leftarrow 5 \\ \leftarrow 1 \\ \hline 6 \end{array} \\
 & \text{Valid in BCD code.} &
 \end{array}$$

$$\begin{array}{rcl}
 \text{b)} & \begin{array}{r} 0011 \ 0101 \\ 0010 \ 0010 + \\ \hline 0101 \ 0111 \end{array} & \begin{array}{l} \leftarrow 35 \\ \leftarrow 22 + \\ \hline 57 \end{array} \\
 & \text{Valid in BCD code} &
 \end{array}$$

c) (1001) + (1000)

$$\begin{array}{rcl}
 & \begin{array}{r} 1001 \\ 1000 + \\ \hline 1 \ 0001 \end{array} & \begin{array}{l} \leftarrow 9 \\ \leftarrow 8 + \\ \hline 17 \end{array} \\
 & \text{Invalid because carry} & \\
 & \begin{array}{r} 0110 + \\ \hline 0001 \ 0111 \end{array} & \text{Add 6} \\
 & \begin{array}{r} 1 \quad 7 \end{array} &
 \end{array}$$

d) (0110 1001) + (0010 0101)

$$\begin{array}{r}
 0110\ 0101 \\
 0010\ 0101 \\
 \hline
 1000\ 1010 \\
 \quad 0110 \\
 \hline
 1001\ 0000 \\
 \quad 9\quad 0
 \end{array}
 \begin{array}{l}
 \leftarrow 65 \\
 \leftarrow 25 + \\
 \text{Right group invalid } (>9) \quad 90 \\
 \text{Add 6 to right group} \\
 \text{Add carry to left group}
 \end{array}$$

e) (1000 0111) + (0101 0011)

$$\begin{array}{r}
 1000\ 0111 \\
 0101\ 0011 + \\
 \hline
 1101\ 1010 \\
 \quad 0110\ 0110 \\
 \hline
 0001\ 0100\ 0000
 \end{array}
 \begin{array}{l}
 87 \\
 53 + \\
 \text{Invalid both group } (>9) \quad 140 \\
 \text{Add 6 to both groups}
 \end{array}$$

1.6.4 Gray code (reflected code)

The gray code is unweighted code, there are no specific weight assigned to the digit position. It is useful code used in digital systems. In this code, the two successive binary values differ in only one bit. The conversion between decimal number and gray code is done by converting the decimal number to binary number, and then the binary number to gray code.

The table (A3) shows the gray code that corresponding to decimal numbers and binary numbers.

Table A3

Decimal	Binary	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

1.6.5 Conversion gray code to binary number

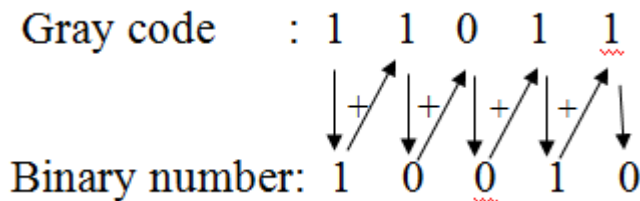
To convert the decimal number to gray code, the following steps are done.

1. The most significant bit (MSB), on the left, is the MSB in binary number. In other word, they are same value.

2. Add the first digit of the binary number to the second digit in code of gray, the carry is ignored. In other word, take XOR operation between them.
3. Generally working from the left to right digit, the n^{th} digit in the binary number is formed from summing the $(n+1)^{\text{th}}$ digit in the binary number with the n^{th} bit in the gray code.

Example A35: Convert the gray code 11011 to binary number?

Solution :



So, the binary number is 10010.

Example A36: Convert the gray code 110001 to binary number?

Solution:

Gray code : 1 1 0 0 0 1

Binary number: 1 0 0 0 0 1

1.6.6 Convert decimal number to gray code

The conversion is down by the following steps:

1. Convert the decimal number to binary number.
2. The MSB digit in code of gray is the same as corresponding digit in the addition of binary number.
3. Starting from left to right (MSB to LSB) , add each adjacent pair of binary digits to get the next gray digits, taking in the account carries are discard.

Example A37: Convert the following binary number 100110 to gray code?

Solution:

Binary number:	1	0	0	1	1	0
	└───┘		└───┘		└───┘	
	↓	+	+	+	+	+
	↓	↓	↓	↓	↓	↓
Gray code:	1	1	0	1	0	1

So the gray code is 110101.

Example A38: Convert the binary number 1010011 to gray code?

Solution

Binary number: 1 0 1 0 0 1 1

Gray code : 1 1 1 1 0 1 0

Example A39: Convert the numbers $(44)_{10}$, $(A.6)_{16}$ to gray codes:

Solution

First, convert the numbers to binary number.

$$(44)_{10} = (101100)_2 ; (A6)_{16} = (10100110)_2$$

Now, convert the binary number to gray code.

1 0 1 1 0 0	1 0 1 0 0 1 1 0
↓ ┐┐┐┐┐	↓ ┐┐┐┐┐┐┐
11 1 0 1 1	11 1 1 0 1 0 1

$$\text{So, } (44)_{10} = (111011)_{\text{gray}}; (A6)_{16} = (11110101)_{\text{gray}}$$

1.6.7 Error detection codes

An error detection code can be used to detect errors during transmission. It is formed by adding additional

bit to the message information. This bit is called parity bit. There are two types of parity:

1. Odd parity: Add bit to the message (0 or 1), so that the total numbers of 1's became odd.
2. Even parity: Add bit to the message (0 or 1), so that the total numbers of 1's became even.

Example A40:

The table A4 illustrates the odd and even parities.

Table A4: Odd parity and even parity

Message	Odd	Even
0000000	1	0
1011101	0	1
1100110	1	0
1111111	0	1

1.6.8 Alphanumeric code

Computers processes both numeric and alphabetic information. Many coding systems have been developed to represent alphanumeric information as a sequences of 0's and 1's. The characters, which are

coded, are alphabets (26 letter), numerals (10 digits) and special characters such as *,\$, +,-, / ... etc.

String of binary digits is used to code each character.

There are two standard codes have been used to ensure uniformity in coding.

1. **ASCII:** It is American Standard Code for Information Interchange. It has 128 characters and each character represents by a 7 bits code. Actually, ASCII can be considered an 8-bit code which MSB for each character is 0.

2. **EBCDIC:** It is Extended Binary Coded Decimal Interchange Code. For each character, 8-bit code are used.

ASCII is represented in the form $A_6, A_5, A_4, A_3, A_2, A_1, A_0$. ASCII is used to code two types of information. The first is the printable character such as digits, alphabets, and special characters. The second is called control characters, which are not printable, use to represent the coded information to control the operation of the digital computer.

Logic Gates

2.1 Introduction

The individual memory cells used in computers are bistable in operation (pendulous between 1 and 0) and capable of storing a single binary bit. Therefore, it is most practical to use the binary number system that uses only two basic symbols 0 and 1 to represent numbers.

Binary logic is used to describe, in mathematical way, the manipulation and processing of binary information. Binary logic consists of binary variables and logic operations. The binary variables are letters of the alphabet such as X, A, D . . . etc. Each variable has two possible values, 0 or 1. The basic logic operations are the three operations OR, AND, and NOT.

2.2 Gates

A **gate** is simply an electronic circuit which operates on one or more input signals to perform the logic operation and produce an output signal. There are many types of gates such as: OR, AND, NOT, NOR, NAND gates. Each gate has its block diagram symbol. The lines connected to each symbol are the inputs on the left and the output on the right of it.

2.3 Truth table

A truth table is a two-dimensional array where there is one column for each input and one column for each output (a circuit may have more than one output). Since we are dealing with binary values, each input can be either 0 or 1. The number of truth table possibilities are 2^n , where n is the number of input variable. For example, if $n=3$ the number of possibilities are $2^3=8$.

The values in the output column are determined from applying the corresponding input values to the functional operator. For example, in the following truth table:

Input		Output
X	Y	F
0	0	0
0	1	1
1	0	0
1	1	1

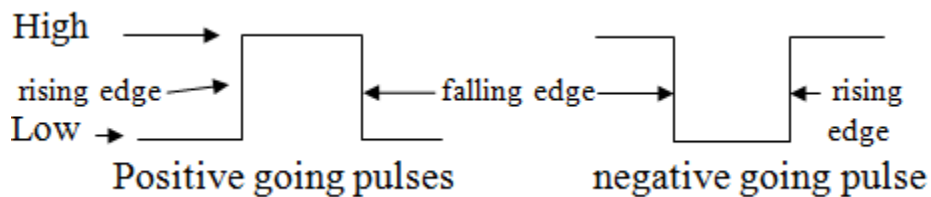
There are 2 input variables, X and Y, and one output variable, F. So there are $2^2=4$ combinations.

The output $F = 1$ when either X and Y are both 1 or $X=0$ and $Y=1$, while the value of F is 0 for the other possibilities.

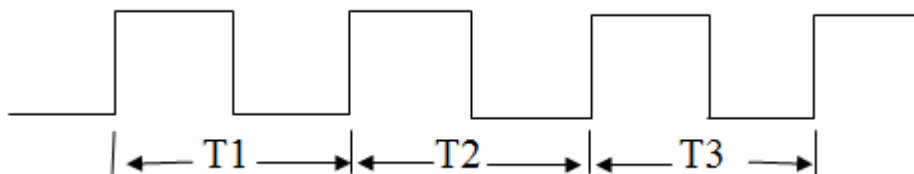
Using truth tables is one method to formally describe the operation of a circuit or function.

2.4 Pulse waveform

Pulses are very important in digital circuits and systems because voltage level are normally changing back and front between High (logic 1) and Low (logic 0) states.



Pulses can be classified as either periodic or non periodic, as shown in fig (2-1).



(a) Periodic waveform



(b) Non- periodic waveform

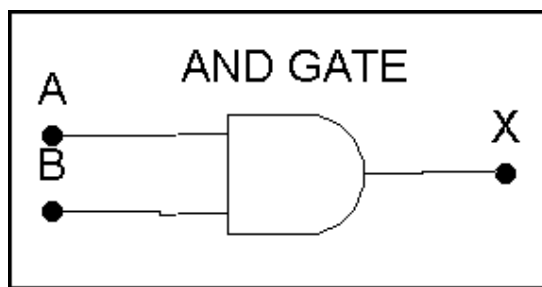
Fig (2-1)

A **timing diagram** is a graphical method of showing the exact output behavior of a logic circuit for every possible of

input condition. It is used to describe the operation of digital devices because its visual characteristics are much easier to understand than explanation using words.

AND, OR, and NOT gates

AND gate: The AND-gate is a device whose output is logic (1) if both of the inputs are logic (1). It performs logical multiplication. It composed of two or more inputs and single output. The logical AND function of two variables is represented either by writing dot between the two variables or by writing the adjacent letters without dot. For example, $X \cdot Y = Z$ or $XY = Z$ is read "X AND Y is equal to Z". This AND gate symbol and its truth table are shown in figure (2-2-a) and (2-2-b) respectively, with two inputs marked A and B and one output marked X.



$$X = A \cdot B$$

(a)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

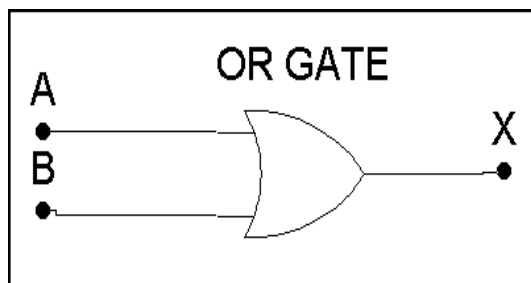
Truth table

(b)

Fig (2-2) AND gate

2.5 OR gate :

It is one of the simplest and most common used. It is a device whose output is logic '1' if either or both of its inputs are '1'. The OR gate composed of two or more inputs and single output. It performs logical addition. The OR operation is represented by a plus sign, +. For example, $X + Y = Z$ is read "X OR Y is equal to Z". The standard logic symbol for the OR gate is shown in fig (2-3-a) and the truth table of combinations for the inputs and outputs for it's is shown in fig (2-3-b).



Block diagram ($X=A+B$)

(a)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

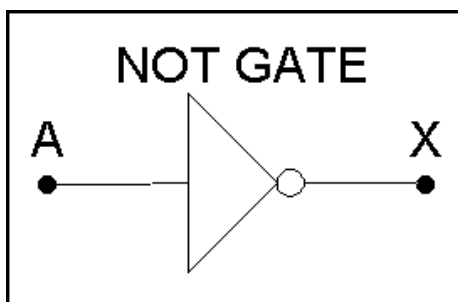
(b)

Fig (2-3): OR gate

2.6 NOT gate (inverter)

The NOT gate (called inverter) perform a basic logic function called inversion or complementation. It has a single input and one output. The purpose of the NOT gate is to change one logic

level to the opposite level. In term of bits, it change '1' at its input to logic '0' at the output , and a '0' to a logic '1'. The NOT operation is represented by a prime or bar. The standard logic symbol for the NOT and its truth table are shown in fig (2-4-a) and fig (2-4-b) respectively.



Block diagram $X = \overline{A}$
(a)

A	$X = \overline{A}$
0	1
1	0

Truth table
(b)

Fig (2-4): NOT gate

From the three basic logic gates (**AND**, **OR**, and **NOT** gates), the most powerful computer circuit can be made. Furthermore, these basic gates are built using transistors, the fundamental building blocks for all digital logic circuits. Transistors are just electronic binary switches that can be turned on or off. The on and off states of transistors are used to represent the two binary values 1 and 0.

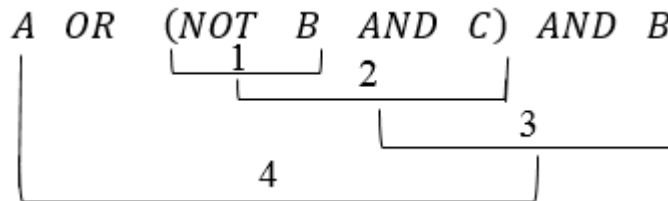
Example: Find the truth table for the following?

A OR (NOT B AND C) AND B?

Solution

Numbering System & Logic Gates

The number of inputs variables is 3, so the number of combinations are $2^3 = 8$. The sequence of perform the operation as follow



The truth table is as following, the NOT is replaced with dash in this table.

A	B	C	B'	$(B' \text{ AND } C)$	$(B' \text{ AND } C) \text{ AND } B$	$A \text{ OR } (B' \text{ AND } C) \text{ AND } B$
0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	1
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	1	0	0	0	1

One can see the last column is equal the value of first column. The block diagram of the circuit that represent the expression is,

figure 2-5:

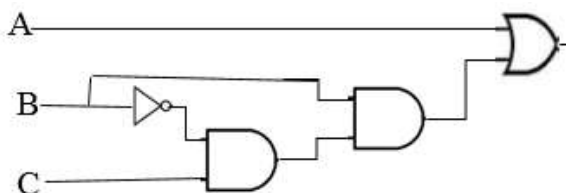


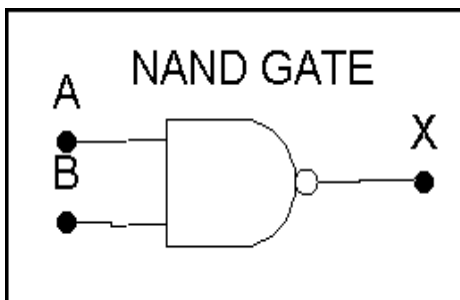
Fig (2-5)

NAND, NOR, X-OR and XNOR gates

2.7 NAND gate: The NAND gate is an inverse (negative) of AND functions. Its output is logic (0) if both inputs are logic (1). It is called NOT-AND gate but is abbreviated NAND gate.

Symbolically the NAND gate

is represented by the AND symbol followed by a small circle indicating an inversion of the output, as shown in fig (2-7). Fig (2-8) shows the equivalent of NAND gate diagram using AND gate and NOT gate.



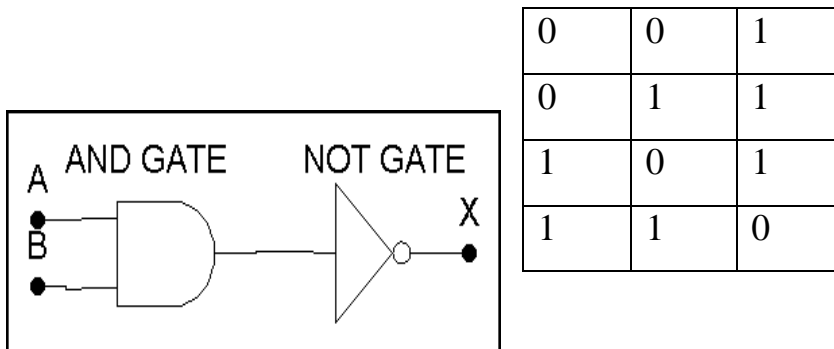
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

$$X = \overline{(A.B)}$$

Truth table

Fig (2-7) NAND gate

A	B	X
---	---	---



$$X = (A.B)$$

(a)

Truth table

(b)

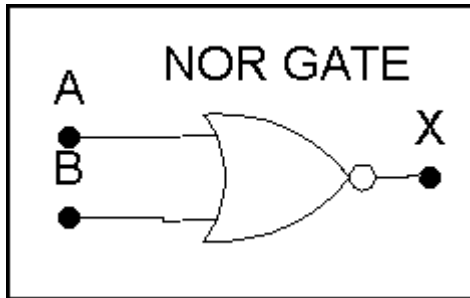
Fig (2-8): Equivalent of NAND gate

2.8 NOR gate:

The NOR gate (NOT–OR) is equivalent of an inverted OR function and will yield a low output (0) if either or both inputs are high (1). Symbolically the NOR gate is represented by the OR gate symbol followed by a bubble to represent the complemented output signal.

The NOR gate is a universal building block of a digital logic because it may be used to implement any logic function.

The NOR gate can be represented by the symbol shown in fig (2-9-a). Its truth table is shown in fig (2-9-b). Fig (2-10) shows the equivalent of NOR gate diagram using OR gate and NOT gate.



Block diagram $X = \overline{A+B}$

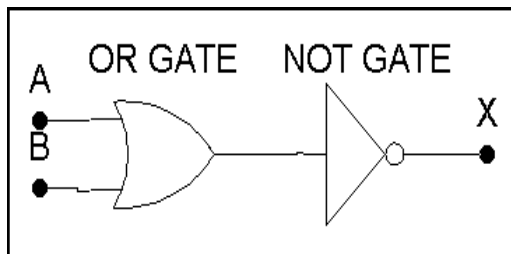
(a)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

(b)

Fig (2-9): NOR gate



Block diagram $X = \overline{A+B}$

(a)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

(b)

Fig (2-10): Equivalent of NOR gate

2.9 Exclusive- OR gate: The Exclusive- OR gate is called the XOR gate. It gives logic (1) at the output when both inputs are

different and logic (0) when both inputs are the same. The XOR of two variables is written by \oplus . The XOR gate symbol and its truth table are shown in fig (2-11).

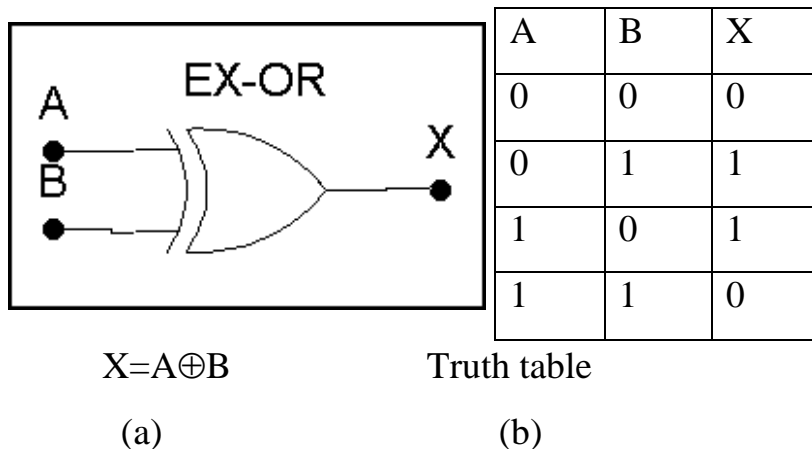
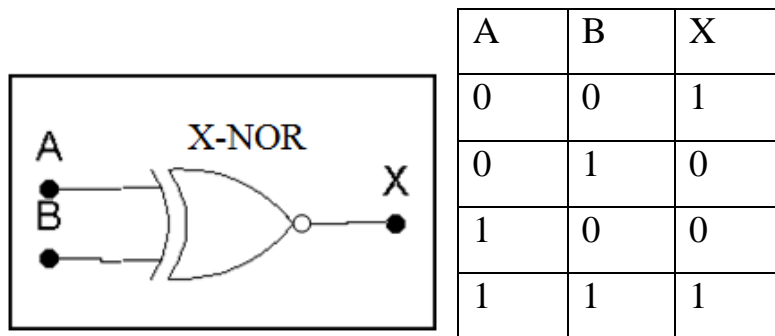


Fig (2-11): XOR gate

2.10 Exclusive NOR gate: Exclusive NOR gate, is called XNOR, is just the inverse of the XOR gate. The XNOR uses the symbol \odot , and it performs the following logic operation :

$$X \odot Y = X \bar{Y} + \bar{X} Y = \overline{(X \oplus Y)}$$

The graphic symbols and truth table of XOR is shown in Fig (2-12). The result is 1 when either both X or Y are 0's or when both are 1's.



$$X = A \odot B$$

(a)

Truth table

(b)

Fig (2-12): XNOR gate

Truth tables clearly show that the exclusive-NOR operation is the complement of the Exclusive-OR. This can also be shown by algebraic manipulation as follows:

$$\begin{aligned}
 \overline{(X \oplus Y)} &= \overline{(X \bar{Y} + \bar{X} Y)} \\
 &= (\overline{X \bar{Y}}) (\overline{\bar{X} Y}) = (\overline{X} + Y) (X + \bar{Y}) \\
 &= (X Y + \bar{X} \bar{Y}) \\
 &= X \odot Y
 \end{aligned}$$

Example: Show using Boolean algebra that $XOR = XNOR$ for three inputs ($A \oplus B \oplus C = A \odot B \odot C$)?

Solution

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

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

2.11 Multiple input gates

Gates are not limited to two input values. There are many variations in the number and types of inputs and outputs allowed for various gates. For example, one can represent the expression $X + Y + Z$ using one OR gate with three inputs. Fig (2-13) illustrate the implementation of this expression using OR gates with two input and its equivalent by using three input OR gate.



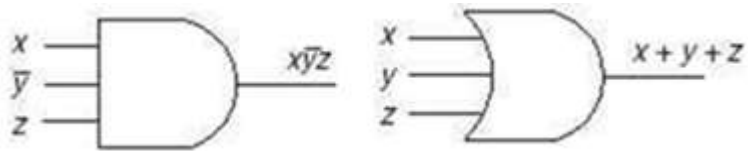


Fig (2-13): Three inputs gates

Multiple inputs of AND gates and multiple input of OR gates in one circuit are illustrates in fig (2-14).

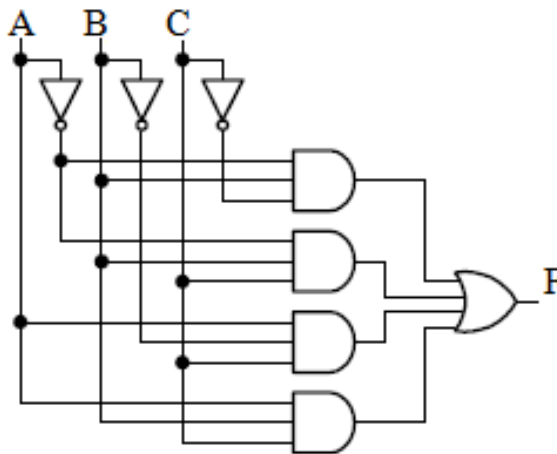


Fig (2-14)