Electronics Engineering

Digital Circuits

Comprehensive Theory

with Solved Examples and Practice Questions

MADE EASY Publications

MADE EASY Publications

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016
E-mail: infomep@madeeasy.in
Contact: 011-45124612, 0-9958995830, 8860378007
Visit us at: www.madeeasypublications.org

Digital Circuits

First Edition: 2015
Second Edition: 2016
Third Edition: 2017
Fourth Edition: 2018

Contents

Digital Circuits

■■■

CHAPTER 01

Basics of Digital Circuits

Introduction

Electronic systems are of two types:

(*i*) Analog systems (*ii*) Digital systems

Analog systems are those systems in which voltage and current variations are continuous through the given range and they can take any value within the given specified range, whereas a digital system is one in which the voltage level assumes finite number of distinct values. In all modern digital circuits there are just two discrete voltage level.

Digital circuits are often called switching circuits, because the voltage levels in a digital circuit are assumed to be switched from one value to another instantaneously. Digital circuits are also called logic circuits, because every digital circuit obeys a certain set of logical rules.

Digital systems are extensively used in control systems, communication and measurement, computation and data processing, digital audio and video equipments, etc.

Advantages of Digital Systems

Digital systems have number of advantages over analog systems which are summarized below:

1. Ease of Design

The digital circuits having two voltage levels, OFF and ON or LOW and HIGH, are easier to design in comparison with analog circuits in which signals have numerical significance ; so their design is more complicated.

2. Greater Accuracy and Precision

Digital systems are more accurate and precise than analog systems because they can be easily expanded to handle more digits by adding more switching circuits.

3. Information Storage is Easy

There are different types of semiconductor memories having large capacity, which can store digital data.

4. Digital Systems are More Versatile

It is easy to design digital systems whose operation is controlled by a set of stored instructions called program. However in analog systems, the available options for programming is limited.

5. Digital Systems are Less Affected by Noise

The effect of noise in analog system is more. Since in analog systems the exact values of voltages are important. In digital system noise is not critical because only the range of values is important.

6. Digital Systems are More Reliable

As compared to analog systems, digital systems are more reliable.

Limitations of Digital System

(*i*) The real world is mainly analog.

(*ii*) Human does not understand the digital data.

1.1. Digital Number Systems

Many number systems are used in digital technology. A number system is simply a way to count. The most commonly used number systems are:

- Decimal number system
- Octal number system
- Binary number system
- Hexadecimal number system

A number system with base or radix '*r*' will have *r* number of different digits from $0 \rightarrow (r-1)$ thus, number system is represented by $(N)_b$

where, N = Number ; b = Base or radix

In general a number with an integer part of '*n*' bits and a fraction part of '*m*' bits can be written as

$$(N)_b = \underbrace{b_{n-1}\, b_{n-2} \cdots b_1 b_0}_{\text{Integer part}} \; \underset{\text{Radix point}}{\overset{\uparrow}{\cdot}} \; \underbrace{b_{-1}\, b_{-2} \cdots b_{-m}}_{\text{Fraction part}}$$

1.1.1 Decimal Number System

- This system has **'base 10'**.
- It has 10 distinct symbols (0, 1, 2, 3, 4, 5, 6, 7, 8 and 9).
- This is a positional value system in which the value of a digit depends on its position.

  $\Rightarrow$ Let we have $(453)_{10}$ is a decimal number

  then,

4 5 3

$3 \times 10^0 = 3$

$5 \times 10^1 = 50$

$4 \times 10^2 = 400$

Finally we get, $(453)_{10}$

$\therefore$ We can say "3" is the least significant digit(LSD) and "4" is the most significant digit(MSD).

***Example* - 1.1** A particular number system having base *B* is given as $(\sqrt{41})_B = 5_{10}$. The value of *'B'* is

(a) 5 (b) 6

(c) 7 (d) 8

Solution : (b)

Squaring both side, $(\sqrt{41})^2 = (5)^2$

$$(41)_B = (25)_{10}$$

$$(4B + 1)_{10} = (25)_{10}$$

$\Rightarrow$ $B = 6$

1.1.2 Binary Number System

- It has base '2' i.e. it has two base numbers 0 and 1 and these base numbers are called "Bits".
- In this number system, group of "Four bits" is known as "Nibble" and group of "Eight bits" is known as "Byte".

i.e. 4 bits = 1 Nibble; 8 bits = 1 Byte

Binary to Decimal Conversion

A binary number is converted to decimal equivalent simply by summing together the weights of various positions in the binary number which contains **'1'**.

Example - 1.2 The decimal number represention of 101101.10101 is

Solution:

$$(101101.10101)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 1\times2^{-5}$$

$$= 32 + 0 + 8 + 4 + 0 + 1 + \frac{1}{2} + 0 + \frac{1}{8} + 0 + \frac{1}{32} = (45.65625)_{10}$$

Decimal to Binary Conversion

The integral decimal number is repeatedly divided by '2' and writing the remainders after each division until a quotient '0' is obtained.

Example - 1.3 Convert $(13)_{10}$ to binary.

Solution:

	Quotient	Remainder	
13 ÷ 2	6	1	LSB
6 ÷ 2	3	0	
3 ÷ 2	1	1	
1 ÷ 2	0	1	MSB

$\therefore$ $(13)_{10} \Rightarrow (1101)_2$

To convert Fractional decimal into binary, Multiply the number by '2'. After first multiplication integer digit of the product is the first digit after binary point. Later only fraction part of the first product is multiplied by 2. The integer digit of second multiplication is second digit after binary point, and so on. The multiplication by 2 only on the fraction will continue like this based on conversion accuracy or until fractional part becomes zero.

Example - 1.4 Convert $(0.65625)_{10}$ to an equivalent base-2 number.

Solution:

0.65625 × 2	0.31250 × 2	0.62500 × 2	0.25000 × 2	0.50000 × 2
1.31250	0.62500	1.25000	0.50000	1.00000
↓ 1	↓ 0	↓ 1	↓ 0	↓ 1

Thus, $(0.65625)_{10} = (0.10101)_2$

1.1.3 Octal Number System

- It is very important in digital computer because by using the octal number system, the user can simplify the task of entering or reading computer instructions and thus save time.
- It has a base of '8' and it posses 8 distinct symbols (0,1...7).
- It is a method of grouping binary numbers in group of three bits.

Octal to Decimal Conversion

An octal number can be converted to decimal equivalent by multiplying each octal digit by its positional weightage.

Example - 1.5 Convert $(6327.4051)_8$ into its equivalent decimal number.

Solution:

$$(6327.4051)_8 = 6\times8^3+3\times8^2+2\times8^1+7\times8^0+4\times8^{-1}+0\times8^{-2}+5\times8^{-3}+1\times8^{-4}$$

$$= 3072+192+16+7+\frac{4}{8}+0+\frac{5}{512}+\frac{1}{4096}$$

$$= (3287.5100098)_{10}$$

Thus, $(6327.4051)_8 = (3287.5100098)_{10}$

Decimal to Octal Conversion

- It is similar to decimal to binary conversion.
- For integral decimal, number is repeatedly divided by '8' and for fraction, number is multiplied by '8'.

Example - 1.6 Convert $(3287.5100098)_{10}$ into octal.

Solution:

For integral part:

	Quotient	Remainder
3287 ÷ 8	410	7
410 ÷ 8	51	2
51 ÷ 8	6	3
6 ÷ 8	0	6

$\therefore$ $(3287)_{10} = (6327)_8$

Now for fractional part:

$$0.5100098 \times 8 = 4.0800784 \rightarrow 4$$

$$0.0800784 \times 8 = 0.6406272 \rightarrow 0$$

$$0.6406272 \times 8 = 5.1250176 \rightarrow 5$$

$$0.1250176 \times 8 = 1.0001408 \rightarrow 1$$

$\therefore$ $(0.5100098)_{10} = (0.4051)_8$

Finally, $(3287.5100098)_{10} = (6327.4051)_8$

Octal-to-Binary Conversion

This conversion can be done by converting each octal digit into binary individually.

Example - 1.7 Convert $(472)_8$ into binary

Solution:

$$\therefore \quad (472)_8 = (\underset{\Downarrow}{4}\ \underset{\Downarrow}{7}\ \underset{\Downarrow}{2})_2 = (100\ 111\ 010)_2$$

Binary-to-Octal Conversion

In this conversion the binary bit stream are grouped into groups of three bits starting at the LSB and then each group is converted into its octal equivalent. After decimal point grouping start from left.

Example - 1.8 Convert $(1011011110.11001010011)_2$ into octal.

Solution:

For left-side of the radix point, we grouped the bits from LSB: $\underset{\downarrow 1}{001}\ \underset{\downarrow 3}{011}\ \underset{\downarrow 3}{011}\ \underset{\downarrow 6}{110}$

Here two 0's at MSB are added to make a complete group of 3 bits.

For right-side of the radix point, we grouped the bits from MSB: $\underset{\text{radix point}}{\bullet}\ \underset{\downarrow 6}{110}\ \underset{\downarrow 2}{010}\ \underset{\downarrow 4}{100}\ \underset{\downarrow 6}{110}$

Here a '0' at LSB is added to make a complete group of 3 bits.

Finally, $(1011011110.11001010011)_2 = (1336.6246)_8$

1.1.4 Hexadecimal Number System

- The base for this system is "16", which requires 16 distinct symbols to represent the numbers.
- It is a method of grouping 4 bits.
- This number system contains numeric digits (0, 1, 2,....9) and alphabets (*A*, *B*, *C*, *D*, *E* and *F*) both, so this is an "ALPHANUMERIC NUMBER SYSTEM".
- Microprocessor deals with instructions and data that use hexadecimal number system for programming purposes.
- To signify a hexadecimal number, a subscript 16 or letter '*H*' is used i.e. $(A7)_{16}$ or $(A7)_H$.

Table-1.1

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

Hexadecimal-to-Decimal Conversion

Example - 1.9 Convert $(3A.2F)_{16}$ into decimal system.

Solution:

$$(3A.2F)_{16} = 3 \times 16^1 + 10 \times 16^0 + 2 \times 16^{-1} + 15 \times 16^{-2}$$

$$= 48 + 10 + \frac{2}{16} + \frac{15}{16^2} = (58.1836)_{10}$$

Decimal-to-Hexadecimal Conversion

Example - 1.10 Convert $(675.625)_{10}$ into Hexadecimal.

Solution:

For Integral Part:

	Quotient	Remainder
$675 \div 16$	42	3
$42 \div 16$	2	10 = A
$2 \div 16$	0	2

$\therefore$ $(675)_{10} = (2A3)_{16}$

For Fractional Part:

$625 \times 16 = 10 = A$

$\therefore$ $(0.625)_{10} = (0.A)_{16}$

Finally, $(675.625)_{10} = (2A3.A)_{16}$

Hexadecimal-to-Binary Conversion

For this conversion replace each hexadecimal digit by its 4 bit binary equivalent.

Example - 1.11 Convert $(2F9A)_{16}$ to Binary System

Solution:

2	F	9	A
↓	↓	↓	↓
0010	1111	1001	1010

$\therefore$ $(2F9A)_{16} = (0010\ 1111\ 1001\ 1010)_2$

Binary-to-Hexadecimal Conversion

For this conversion the binary bit stream is grouped into pairs of four (starting from LSB) and hex number is written for its equivalent binary group.

Example - 1.12 Convert $(10100110101111)_2$ to hexadecimal number system.

Solution:

00 10	10 01	10 10	1111
↓	↓	↓	↓
2	9	A	F

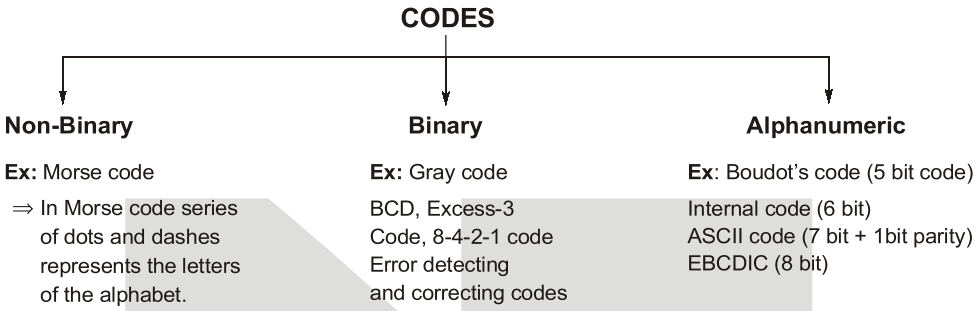
Here two 0's at MSB are added to make a complete group of 4 bits.

$\therefore$ $(10100110101111)_2 = (29AF)_{16}$

The number systems can also be classified as weighted binary number and unweighted binary number. Where weighted number system is a positional weighted system for example, Binary, Octal, Hexadecimal *BCD*, 2421 etc. The unweighted number systems are non-positional weightage system for example Gray code, Excess-3 code etc.

1.2 Codes

When numbers, letters or words are represented by a special group of symbols, we say that they are being encoded, and the group of symbols is called "CODE".



1.2.1 Binary Coded Decimal Code (BCD)

- In this code, each digit of a decimal number is represented by binary equivalent.
- It is a 4-bit binary code.
- It is also known as "8-4-2-1 code" or simply "BCD Code".
- It is very useful and convenient code for input and output operations in digital circuits.
- Also, it is a "weighted code system".

For example:

$(943)_{decimal} \longrightarrow (.....)_{BCD}$

$\Rightarrow$ 9 4 3

↓ ↓ ↓

1001 0100 0011

$\therefore \quad (943)_{10} = (100101000011)_2$

Advantages of BCD Code

- The main advantage of the BCD code is relative ease of converting to and from decimal.
- Only 4-bit code groups for the decimal digits "0 through 9" need to be remembered.
- This case of conversion is especially important from the hardware standpoint.

  $\Rightarrow$ In 4-bit binary formats, total number of possible representation = $2^4 = 16$

  Then, Valid BCD codes = 10

  Invalid BCD codes = 6

  $\Rightarrow$ In 8-bit binary formats,

  Valid BCD codes = 100

  Invalid BCD codes = 256 – 100 = 156

1.2.2 Excess-3 Code

- It is a 4-bit code.
- It can be derived from BCD code by adding "3" to each coded number.
- It is an "unweighted code".

- It is a "self-complimenting code" i.e. the 1's compliment of an excess-3 number is the excess-3 code for the 9's compliment of corresponding decimal number.
- This code is used in arithmetic circuits because of its property of self complimenting.

Example - 1.13 Convert $(48)_{10}$ into Excess-3 code.

Solution:

4	8
+3	+3
7	11
↓	↓
0111	1011

$\therefore \quad (48)_{10} = (01111011)$

⇓ equivalent 4–bit binary

Example - 1.14 Represent the decimal number 6248 in

(*i*) BCD code (*ii*) Excess-3 code (*iii*) 2421 code

Solution:

(*i*) BCD code

6 2 4 8

0110 – 0010 – 0100 – 1000

(*ii*) Excess-3 = BCD + 3

= 1001 0101 0111 1011

(*iii*) 2421 code

2	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
1	0	1	1	5
1	1	0	0	6
1	1	0	1	7
1	1	1	0	8
1	1	1	1	9

6248 = 1100 0010 0100 1110

Example - 1.15 The state of a 12-bit register is 1000100101111. What is its content if it represents?

(*i*) Three decimal digits in BCD?

(*ii*) Three decimal digits in Excess-3 code?

Solution:

(*i*) In BCD ⇒ 1000 1001 0111 ; Decimal digits = 897

(*ii*) In Excess-3 ⇒ 1000 (8−3=5) 1001 (9−3=6) 0111 (7−3=4); Decimal digits = 564

1.2.3 Gray Code

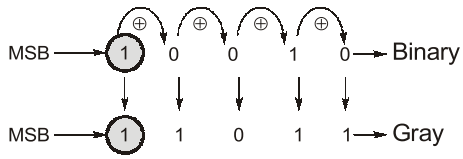
- It is a very useful code also called "minimum change codes" in which only one bit in the code group changes when going from one step to the next.
- It is also known as "Reflected code".
- It is an unweighted code, meaning that the bit positions in the code groups do not have any specific weight assigned to them.
- This code is not well suited for arithmetic operations but it finds application in input/output devices.
- These are used in instrumentation such as shaft encoders to measures angular displacement or in linear encoders for measurement of linear displacement.

Binary-to-Gray Conversion

- 'MSB' in the gray code is same as corresponding digit in binary number.
- Starting from "Left to Right", add each adjacent pair of binary bits to get next gray code bit. (Discard the carry if generated).

***Example* - 1.16** Convert $(10010)_2$ to gray code.

Solution:



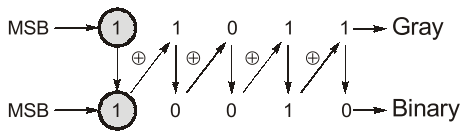
$\therefore$ $(10010)_2 = (11011)_{Gray}$

Gray-to-Binary Conversion

- "MSB" of Binary is same as that of gray code .
- Add each binary bit to the gray code bit of the next adjacent position (discard the carry if generated), to get next bit of the binary number.

***Example* - 1.17** Convert $(11011)_{Gray}$ to Binary code.

Solution:



$\therefore$ $(11011)_{Gray} = (10010)_2$

Various Binary Codes

Table-1.2 *Various binary codes*

Decimal Number	Binary				BCD				Excess-3				Gray			
	B_3	B_2	B_1	B_0	D	C	B	A	E_3	E_2	E_1	E_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	0	0	1	0	0	1	0	1	0	0	1	1
3	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	0
4	0	1	0	0	0	1	0	0	0	1	1	1	0	1	1	0
5	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	1
6	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	1
7	0	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	0
9	1	0	0	1	1	0	0	1	1	1	0	0	1	1	0	1
10	1	0	1	0									1	1	1	1
11	1	0	1	1									1	1	1	0
12	1	1	0	0									1	0	1	0
13	1	1	0	1									1	0	1	1
14	1	1	1	0									1	0	0	1
15	1	1	1	1									1	0	0	0

1.3 Arithmetic Operations

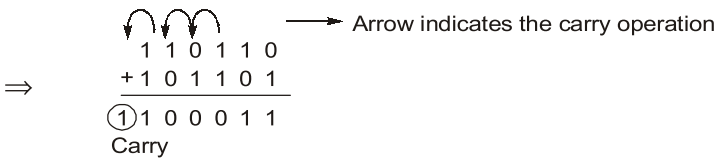
We are all familiar with the arithmetic operations like addition, subtraction, multiplication and division using decimal numbers. Such operations can also be performed on digital numbers.

1.3.1 Binary Addition

$$0 + 0 = 0; \quad 0 + 1 = 1$$
$$1 + 0 = 1; \quad 1 + 1 = 10$$

For example:

Add the binary numbers 110110 and 101101



1.3.2 Binary Subtraction

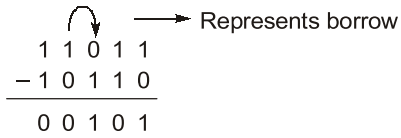
$$0 - 0 = 0; \quad 10 - 1 = 1 \text{ (Borrow)}$$
$$1 - 0 = 1; \quad 1 - 1 = 0$$

While subtracting a large number from a smaller number, we can subtract the smaller from the larger and change the sign.

For example:

Subtract two binary numbers 11011 and 10110.

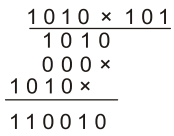
$\Rightarrow$



1.3.3 Binary Multiplication

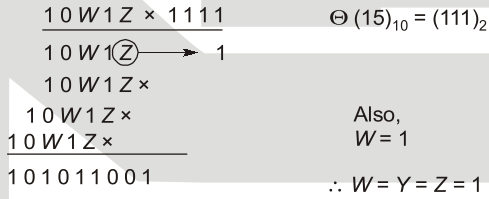
Multiply two binary numbers 1010 and 101

$\Rightarrow$



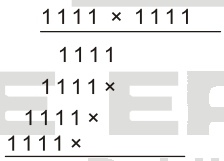
Example - 1.18 If $(10W1Z)_2 \times (15)_{10} = (Y01011001)_2$ then find the value of W, Y, Z.

Solution:



Example - 1.19 $(1111)_2 \times (1111)_2 = ?$

Solution:



1.3.4 Octal Addition

$0 + 0 = 0$

$0 + 2 = 2$

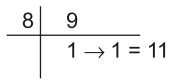
$1 + 6 = 7$

$1 + 7 = 0$ with carry = 1

Whenever the generated number is greater than 7 then, after decimal addition it should be converted into octal.

For example:

$7 + 2 = 9$



$7 + 17 = 26$