Wachemo University

Department of Electrical and Computer Engineering

Digital Logic Design(DLD)

Chapter 2 Number system, operations and codes

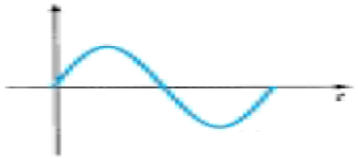
Pre. Getahun.T

Dec.2023

Binary Systems

DIGITAL SYSTEMS

- Digital means electronic technology that generates, stores, and processes data in terms of two states: positive and non-positive.
- Positive is expressed or represented by the number 1 and non-positive by the number 0.



A „**digital system**" is a data technology that uses discrete (discontinuous) values represented by high and low states known as bits.

- By contrast, non-digital (or analog) systems use a continuous range of values to represent information.
- Although digital representations are discrete, the information represented can be either discrete, such as numbers, letters or icons, or continuous, such as sounds, images, and other measurements of continuous systems.

Cont….

- A „**digital system**“ is a data technology that uses discrete (discontinuous) values represented by high and low states known as bits.
  - By contrast, non-digital (or analog) systems use a continuous range of values to represent information.
  - Although digital representations are discrete, the information represented can be either discrete, such as numbers, letters or icons, or continuous, such as sounds, images, and other measurements of continuous systems.

Binary digital logic level and digital waveform

- Binary describes a numbering scheme in which there are only two possible values for each digit: 0 and 1. The term also refers to any digital encoding/decoding system in which there are exactly two possible states.
- In digital data memory, storage, processing, and communications, the 0 and 1 values are sometimes called "low" and "high," respectively.

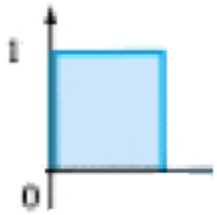


Figure-1 Digital waveform

Number system

- **Numeric systems:** The numeric system we use daily is the decimal system, but this system is not convenient for machines since the information is handled codified in the shape of on or off bits; this way of codifying takes us to the necessity of knowing the positional calculation which will allow us to express a number in any base where we need it.
- A base of a number system or radix defines the range of values that a digit may have.
- In the binary system or base 2, there can be only two values for each digit of a number, either a "0" or a "1".
- The Octal Numbering System is a binary number divided up into groups of only 3 bits, with each group or set of bits having a distinct value of between 000 (0) and 111 (7).
- Octal numbers therefore have a range of just “8” digits, (0, 1, 2, 3, 4, 5, 6, 7) making them a Base-8 numbering system.

Cont….

- The “Hexadecimal” or simply “Hex” numbering system uses the Base of 16 system and is a popular choice for representing long binary values because their format is quite compact and much easier to understand compared to the long binary strings of 1‟s and 0‟s.
- In the hexadecimal system, we allow 16 values for each digit of a number: "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", and "F". Where “A” stands for 10, “B” for 11 and so on.

Convert from Decimal to Any Base

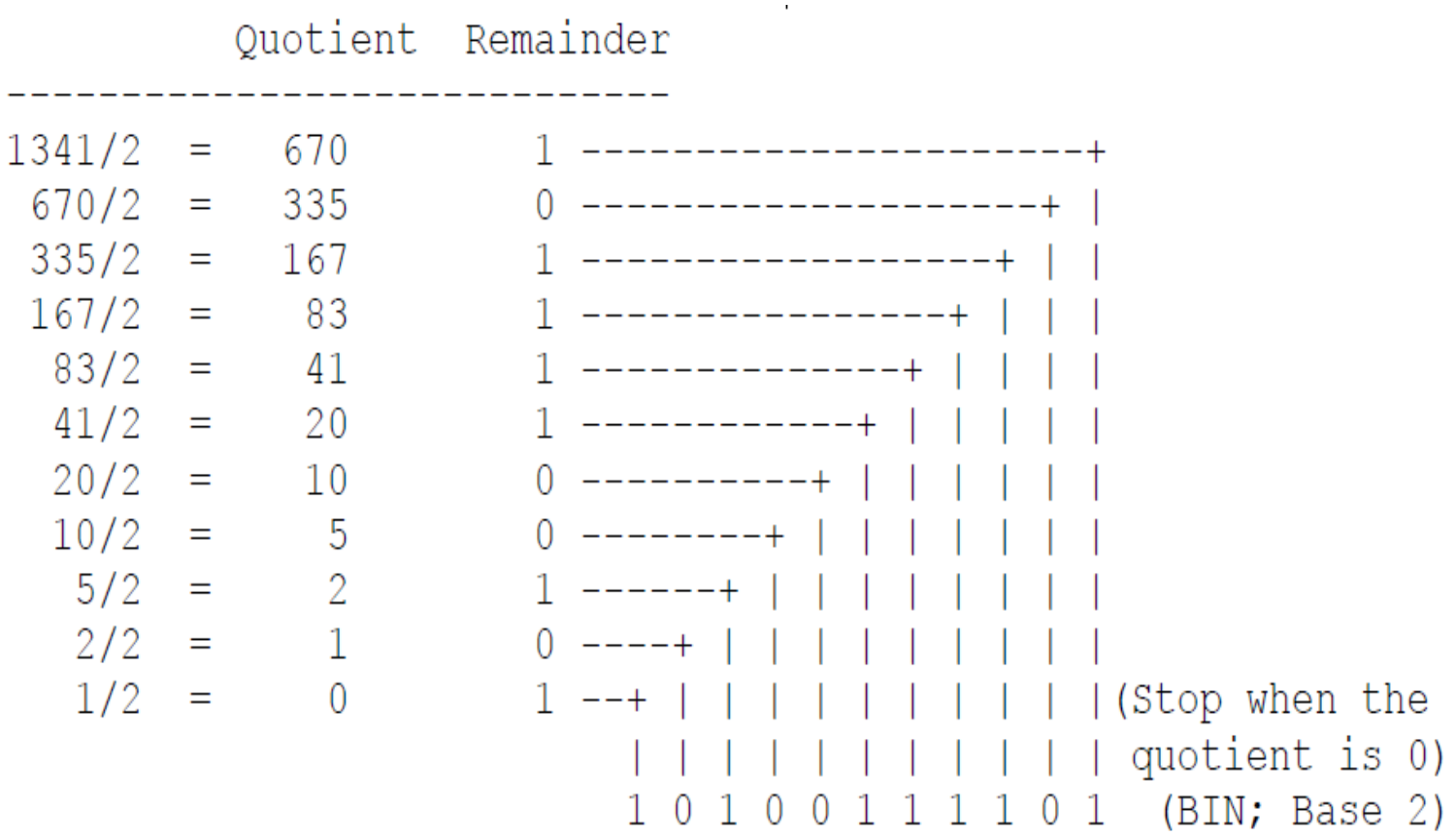
- There are many methods or techniques which can be used to convert code from one format to another. We'll demonstrate here the following:
- As an example, let's start with a decimal number 1234 and convert it to decimal notation (base 10)

```
Quotient  Remainder

     ----------------------------
     1234/10 =   123        4 --------+
      123/10 =    12        3 ------+ |
       12/10 =     1        2 ----+ | |
        1/10 =     0        1 --+ | | |(Stop when the quotient is 0)
                                | | | |
                                1 2 3 4   (Base 10)
```

Cont…

- Let's express a decimal number 1341 in binary notation. Note that the desired base is 2, so we repeatedly divide the given decimal number by **2.**



Cont…

- Let's express the same decimal number 1341 in octal notation.

```
            Quotient  Remainder
  ----------------------------
  1341/8  =   167         5 --------+
   167/8  =    20         7 ------+ |
    20/8  =     2         4 ----+ | |
     2/8  =     0         2 --+ | | |   (Stop when the quotient is 0)
                              | | | |
                              2 4 7 5   (OCT; Base 8)
```

- Let's express the same decimal number 1341 in hexadecimal notation.

```
            Quotient  Remainder
  ----------------------------
  1341/16 =    83        13 ------+
    83/16 =     5         3 ----+ |
     5/16 =     0         5 --+ | |   (Stop when the quotient is 0)
                              | | |
                              5 3 D   (HEX; Base 16)
```

Cont…

- The conversion of the fractional part can be obtained by successively multiplying the fraction with the base. If we iterate this process on the remaining fraction, then we will obtain successive significant digit. This methods form the basis of the multiplication methods of converting fractions between bases
- Example. Convert the decimal number 3315 to hexadecimal notation. What about the hexadecimal equivalent of the decimal number 3315.3?

```
           Quotient  Remainder
-----------------------------
3315/16 =    207         3 ------+
 207/16 =     12        15 ----+ |
  12/16 =      0        12 --+ | |   (Stop when the quotient is 0)
                             | | |
                             C F 3   (HEX; Base 16)
```

```
                                                (HEX; Base 16)
             Product  Integer Part    0.4 C C C ...
-------------------------------       | | | |
0.3*16   =    4.8         4          ----+ | | | | |
0.8*16   =   12.8        12          ------+ | | | |
0.8*16   =   12.8        12          --------+ | | |
0.8*16   =   12.8        12          ----------+ | |
           :                                     ----------------------+
           :
Thus, 3315.3 (DEC) --> CF3.4CCC... (HEX)
```

Convert From Any Base to Decimal

- Example-1. Convert 234.14 expressed in an octal notation to decimal.

$$2*8^2 + 3*8^1 + 4*8^0 + 1*8^{-1} + 4*8^{-2} = 2*64 + 3*8 + 4*1 + 1/8 + 4/64 = 156.1875$$

- Example-2. Convert the hexadecimal number 4B3 to decimal notation. What about the decimal equivalent of the hexadecimal number 4B3.3?

```
Original Number:      4     B     3  .  3
                      |     |     |     |
How Many Tokens:      4    11     3     3
Digit/Token Value:  256    16     1     0.0625
Value:             1024  +176  + 3   + 0.1875  = 1203.1875
```

- **Example**. Convert 234.14 expressed in an octal notation to decimal

```
Original Number:     2    3    4  .  1        4
                     |    |    |     |        |
How Many Tokens:     2    3    4     1        4
Digit/Token Value:  64    8    1     0.125    0.015625
Value:           128 + 24  + 4   + 0.125 + 0.0625    = 156.1875
```

- To convert a binary number to octal, we partition the base 2 number into groups of three starting from the radix point, and pad the outermost groups with 0's as needed to form triples. Then, we convert each triple to the octal equivalent.
- For conversion from base 2 to base 16, we use groups of four.

Consider converting 10110_2 to base 8:

$10110_2 = 010_2\ \ 110_2 = 2_8\ \ 6_8 = 26_8$

Notice that the leftmost two bits are padded with a 0 on the left in order to create a full triplet.

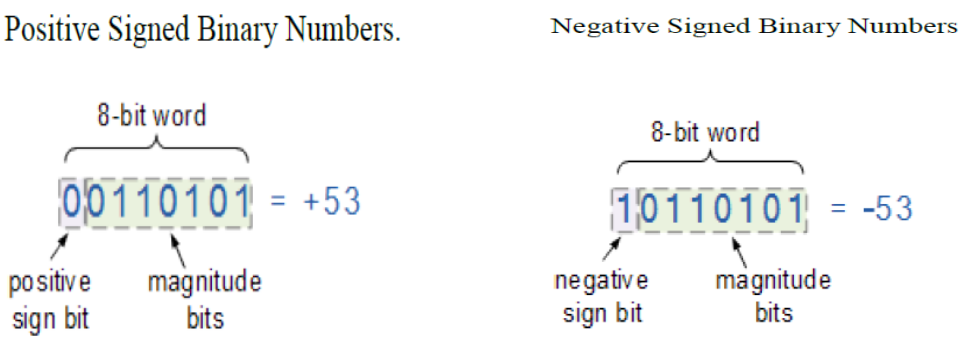
Now consider converting 10110110_2 to base 16:

$10110110_2 = 1011_2\ \ 0110_2 = B_{16}\ \ 6_{16} = B6_{16}$

(Note that 'B' is a base 16 digit corresponding to 11_{10}. B is not a variable.)

SIGNED BINARY NUMBERS

- Positive numbers (including zero) are represented as unsigned numbers in mathematics. We do not put the +ve sign in front of them to show that they are positive numbers.
- However, when dealing with negative numbers we do use a -ve sign in front of the number to show that the number is negative in value and different from a positive unsigned value, and the same is true with signed binary numbers.
- However, in digital circuits, there is no provision made to put a plus or even a minus sign to a number, since digital systems operate with binary numbers that are represented in terms of "0"s" and "1"s".
- So to represent a positive (N) and a negative (-N) binary number we can use the binary numbers with signs. For signed binary numbers the most significant bit (MSB) is used as the sign. If the sign bit is "0", this means the number is positive.
- If the sign bit is "1", then the number is negative. The remaining bits represent the binary number's magnitude in the usual unsigned binary number format.



Cont…

One's Complement Representation

- The one's complement operation is trivial to perform: convert all of the 1's in the number to 0's, and all of the 0's to 1's. We can observe from the table that in the one's complement representation, the leftmost bit is 0 for positive numbers and 1 for negative numbers, as it is for the signed magnitude representation. This negation, changing 1's to 0's and changing 0's to 1's is known as complementing the bits.
- Consider representing $(+12)_{10}$ and $(-12)_{10}$ in an eight-bit format, now using the one's complement representation:

$$(+12)10 = (00001100)2$$

$$(-12)10 = (11110011)2$$

Two's Complement Representation

- The two's complement is formed in a way similar to forming the one's complement: complement all of the bits in the number, but then add 1, and if that addition results in a carry-out from the most significant bit of the number, discard the carry-out.

Cont…

- Consider again representing $(+12)_{10}$ and $(-12)_{10}$ in an eight-bit format, this time using the two's complement representation. Starting with (+12)10 =(00001100)2, complement, or negate the number, producing (11110011)2.

Now add 1, producing $(11110100)_2$, and thus $(-12)_{10} = (11110100)_2$:

$(+12)_{10} = (00001100)_2$

$(-12)_{10} = (11110100)_2$

Binary Codes

- In coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called a code.
- The digital data is represented, stored and transmitted as a group of binary bits. This group is also called binary code.
- The binary code is represented by a number as well as an alphanumeric letter.

The advantages of Binary Code

- are suitable for computer applications and digital communications.
- Binary codes make the analysis and designing of digital circuits Since only 0 & 1 are being used, implementation becomes easy.

Classification of binary codes

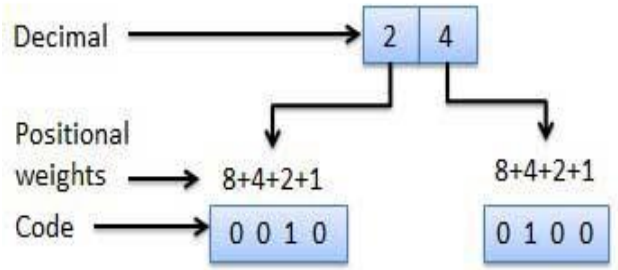
The codes are broadly categorized into the following four categories.

- Weighted Codes
- Non-Weighted Codes
- Binary Coded Decimal Code
- Alphanumeric Codes
- Error Detecting Codes
- Error Correcting Codes

Cont…

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.

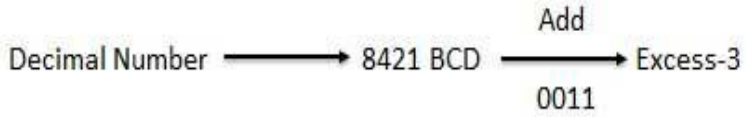


Non-Weighted Codes

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

Excess-3 code

- The Excess-3 code is also called as XS-3 code. It is a 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



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

Gray Code

- It is a non-weighted code and it is not an arithmetic code. That means there are no specific weights assigned to the bit position.
- It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in Fig below.
- As only one-bit changes at a time, the gray code is called a unit distance code. The gray code is cyclic and cannot be used for arithmetic operations.

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Binary Coded Decimal (BCD) code

- In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code.
- In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Advantages of BCD Codes

- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

cont…

- Example − convert (11101)2 to BCD.

Step 1 − Convert to Decimal

Binary Number − 11101_2

Calculating Decimal Equivalent −

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	11101_2	$(16 + 8 + 4 + 0 + 1)_{10}$
Step 3	11101_2	29_{10}

Binary Number − 11101_2 = Decimal Number − 29_{10}

Step 2 − Convert to BCD

Decimal Number − 29_{10}

Calculating BCD Equivalent. Convert each digit into groups of four binary digits equivalent.

Step	Decimal Number	Conversion
Step 1	29_{10}	$0010_2\ 1001_2$
Step 2	29_{10}	00101001_{BCD}

Result

$(11101)_2 = (00101001)_{BCD}$

Logic Gates

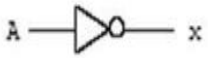
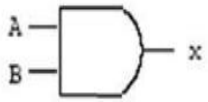
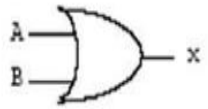
Formal logic: In formal logic, a statement (proposition) is a declarative sentence that is either true (1) or false (0).

- It is easier to communicate with computers using formal logic.

Boolean variable: Takes only two values – either true (1) or false (0). They are used as basic units of formal logic.

- Boolean algebra: Deals with binary variables and logic operations operating on those variables.

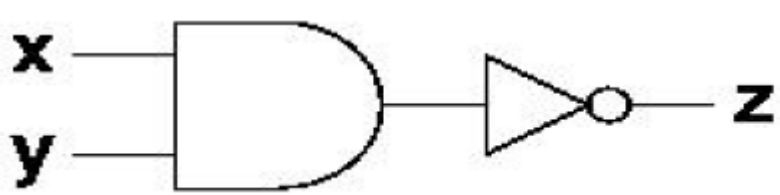
Logic diagram: Composed of graphic symbols for logic gates. A simple circuit sketch that represents inputs and outputs of Boolean functions.

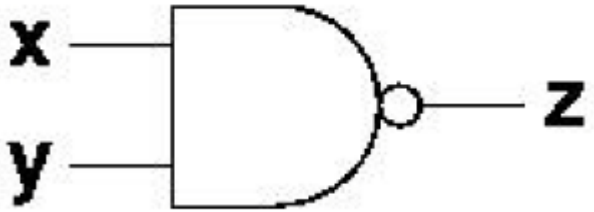
Name	Graphic symbol	Algebraic function	Truth table	
Inverter		x = A'	A | x 0 | 1 1 | 0	
AND		x = AB	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	True if both are true.
OR		x = A + B	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	True if either one is true.

Cont…

- **The *NAND-gate***

The *nand*-gate is an *and*-gate with an inverter on the output.





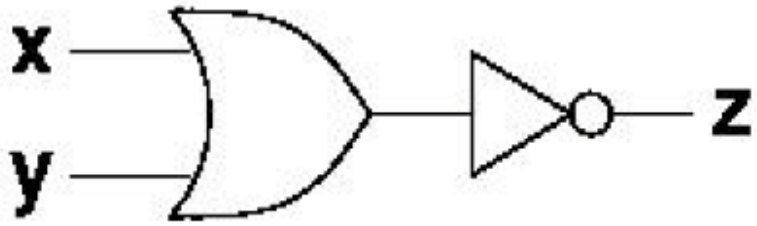
x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

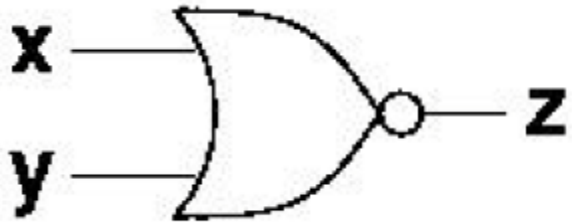
The truth table clearly shows that the NAND operation is the complement of the AND

Cont…

- **The *NOR -gate***

The *nor*-gate is an *or*-gate with an inverter on the output.



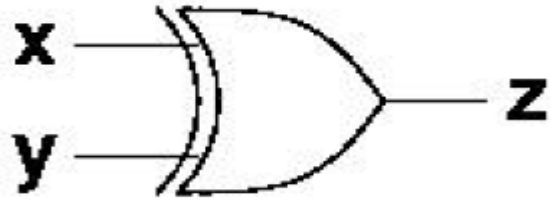


The truth table for the *nor*-gate is like the one for the *or*-gate, except that all output values have been inverted:

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

Cont…

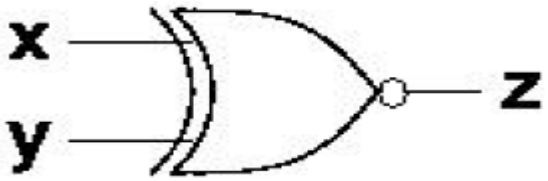
- **The *exclusive-or-gate***
  - The exclusive-or-gate is similar to an or-gate. It can have an arbitrary number of inputs, and its output value is 1 if and only if exactly one input is 1 (and thus the others 0). Otherwise, the output is 0. We draw an exclusive-or-gate like this:



x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

The *exclusive-Nor-gate*

- The *exclusive-Nor-gate is similar to an N-or-gate. It can have an arbitrary number of inputs, and its output value is 1 if and only if the two inputs are of the same values (1 and 1 or 0 and 0). Otherwise, the output is 0. We draw an exclusive Nor gate like this:*



x	y	z
0	0	1
0	1	0
1	0	0
1	1	1