# **CE102 Digital Logic Design**

Çiğdem Sazak Turgut 2022

Week-2 (Introduction to Digital Logic Design)

Spring Semester, 2021-2022

Download DOC, SLIDE, PPTX



# **PART 1: BINARY SYSTEMS**



# **Binary Systems**

- Analog Vs Digital
- Digital Systems Binarynumbers
- Number base conversions Compliments Binary Systems
  - Octal and Hexadecimal Numbers
- Signed Binary Numbers



### **Analog and Digital**

- Analog information is made up of a continuum of values within a given range.
- At its most basic, digital information can assume only one of two possible values:

```
one/zero,
```

- on/off,
- high/low,
- true/false, etc.
- Digital Information is less susceptible to noise than analog information
- Exact voltage values are not important, only their class (1 or 0)
- The complexity of operations is reduced, thus it is easier to implement them with high accuracy in digital form.

# **Digital Systems**

- Digital;
  - generates stores
  - o processes data



- two states:
  - positive (1) and
  - lacktriangledown non-postitive (0)



# **Digital Systems**

- A "digital system" is a data technology that uses discrete (discontinuous) values represented by high and low states known as bits.
- non-digital (or analog) systems use a continuous range of values to represent information



## **Binary Number System**

#### Binary;

 describes a numbering scheme in which there are only two possible values for each digit: 0 and 1

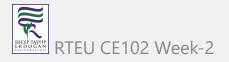
#### Binary Number System

- o a numbering system
- o represents numeric values using 0 and 1
- known as the base-2 number system



## **BINARY NUMBER EXAMPLE**

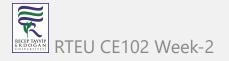
- 10
- 111
- 10101
- 11110



#### **COMPLIMENTS**

- used in digital computers to simplify the subtraction operation and for logical manipulation
- There are 2 types of complements for each base r system
  - (1) The radix complement
  - (2) Diminished radix compliment

Radix compliment: Also referred to as the r"s compliment. Diminished radix compliment: Also referred to as (r-1)"s compliment



#### **OCTAL NUMBERS**

- a binary number is divided up into groups of only 3 bits
  - o 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 and therefore, q is equal to "8"



#### **HEXADECIMAL NUMBERING SYSTEM**

- main disadvantage of binary numbers
  - the binary string equivalent of a large decimal base-10 number can be quite long
  - Working with large digital systems, such as computers, it is common to find binary numbers consisting of 8, 16 and even 32 digits
- Overcome the above problem:
  - to arrange the binary numbers into groups or sets of four bits (4-bits)
  - These groups of 4-bits uses another type of numbering system also commonly used in computer and digital systems called Hexadecimal Numbers
  - uses the Base of 16 system
  - Hexdecimal system format is quite compact and much easier to understand



## **HEXADECIMAL NUMBERING SYSTEM**



#### SIGNED BINARY NUMBERS

- In mathematics,
  - positive numbers (including zero) are represented as unsigned numbers we do
    not put the (+) ve sign in front of them to show that they are positive
    numbers
  - When dealing with negative numbers we do use a (—) 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
  - o there is no provision made to put a plus or even a minus sign to a number
  - $\circ$  digital systems operate with binary numbers that are represented in terms of "  $0\mbox{"s"}$  and "1"s"
- to represent a positive (N) and a negative (-N) binary number we can use the binary numbers with sign



- For signed binary numbers the most significant bit (MSB) is used as the sign
- If the sign bit is "0":
  - the number is positive
- If the sign bit is "1":
  - the number is negative
- The remaining bits are used to represent the magnitude of the binary number in the usual unsigned binary number format.



## **Positive Signed Binary Number**

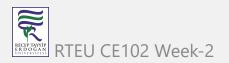
• 8-bit word



## **Negative Signed Binary Number**

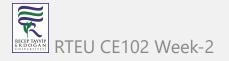
• 8-bit word





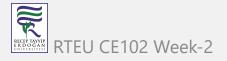
#### **BINARY CODES**

- In the 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 as a code
- digital data is represented, stored and transmitted as group of binary bits
- called BINARYCODE



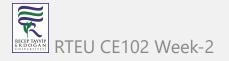
### **Advantages of Binary Code**

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



## **Classification of Binary Codes**

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



## **Weighted Codes**

- 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



## **Non-Weighted Codes**

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



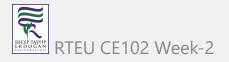
#### **Excess-3 Code**

- also called xs-3 code
- It is non-weighted code used to express decimal numbers
- The Excess-3 code words are derived from the 8421 BCD code words adding (0011)2 or (3)10 to each code word in 8421



The excess-3 codes are obtained as follows

Example : Decimal  $\Longrightarrow 8421_{BCD} \Longrightarrow$  Excess-3



### **Gray Code**

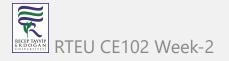
- It is the non-weighted code and it is not arithmetic codes
- Application of Gray code
  - Gray code is popularly used in the shaft position encoders
  - A shaft position encoder produces a code word which represents the angular position of the shaft



### Binary Coded Decimal (BCD) Code

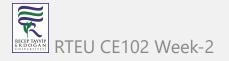
- In this code each decimal digit is represented by a 4-bit binary number
- BCD is a way to express each of the decimal digits with a binary code
- In the BCD, with four bits we can represent sixteen numbers (0000 to 1111)

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

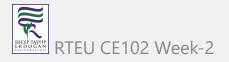


#### **Alphanumeric Codes**

- Abinary digit or bit can represent only two symbols as it has only two states '0' or
   '1'
- But this is not enough for communication between two computers because there we need many more symbols for communication.
- These symbols are required to represent 26 alphabets with capital and small letters, numbers from 0 to 9, punctuation marks and other symbols
- The alphanumeric codes are the codes that represent numbers and alphabetic characters
- Mostly such codes also represent other characters such as symbol and various instructions necessary for conveying information



- The following three alphanumeric codes are very commonly used for the data representation.
  - American Standard Code for Information Interchange (ASCII)
  - Extended Binary Coded Decimal Interchange Code (EBCDIC)
  - Five bit Baudot Code



#### **Number Base Conversions**

- Binary to BCD Conversion
- BCD to Binary Conversion
- BCD to Excess-3
- Excess-3 to BCD



## **Binary to BCD Conversion**

- Step-1: Convert the binary number to decimal
- Step-2: Convert decimal number to BCD



### **Step-1: Binary to Decimal Conversion**

Convert to **Decimal** Equivalent

**Example**: Convert  $(11101)_2$  to BCD

$$= (11101)_2$$
 $= ((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$ 
 $= (16 + 8 + 4 + 0 + 1)_{10}$ 
 $= 29_{10}$ 
 $\downarrow$ 
 $(11101)_2 = 29_{10}$ 



### Step-2: Decimal to BCD Conversion

Convert to **BCD** Equivalent

**Example**: Convert  $(11101)_2$  to BCD

Convert each digit into groups of four binary digits equivalent

$$= (11101)_2 = 29_{10}$$
 $= 29_{10}$ 
 $= 0010_21001_2$ 
 $= (00101001)_{BCD}$ 
 $\downarrow$ 
 $(11101)_2 = (00101001)_{BCD}$ 



#### **BCD** to Decimal Conversion

- Calculating Decimal Equivalent
  - Convert each four digit into a group and get decimal equivalent for each group

```
egin{aligned} &= (00101001)_{BCD} \ &= 0010_2 1001_2 \ &= 2_{10} 9_{10} \ &= 29_{10} \ \downarrow \ &= (00101001)_{BCD} = 29_{10} \end{aligned}
```



- ullet Calculating Binary Equivalent of  $29_{10}$ 
  - Used long division method for decimal to binary conversion



#### **BCD** to Excess-3 Conversion

Step 1: Convert BCD to decimal

**Step 2**: Add  $(3)_{10}$  to this decimal number

Step 3:Convert into binary to get excess-3 code



#### **BCD to Excess-3 Conversion**

Example – convert  $(1001)_{BCD}$  to Excess-3

- = Step-1:Convert to Decimal  $\rightarrow (1001)_{BCD} = 9_{10}$
- = Step-2:Add 3 to decimal  $\to 9_{10} + 3_{10} = 12_{10}$
- = Step-3:Convert to Excess-3  $\rightarrow$  12<sub>10</sub> = (1100)<sub>2</sub>

+

$$(1001)_{BCD} = (1100)_{XS-3}$$



#### **Excess-3 to BCD Conversion**

 $\bullet$  Subtract  $(0011)_2$  from each 4 bit of  $\,$  excess-3  $\,$  digit to obtain the corresponding BCD code



#### **Excess-3 to BCD Conversion**

Example: Convert  $(10011010)_{XS-3}$  to BCD.

```
Given XS-3 number = 1 0 0 1 1 0 1 0

Subtract (0011)_2 = 0 0 1 1 0 0 1 1

BCD = 0 1 1 0 0 1 1 1
```

Result

$$(10011010)_{XS-3} = (01100111)_{BCD}$$



# **PART 2: BINARY ARITHMETIC SYSTEMS**



# References



$$End-Of-Week-2-Module$$

