Digital system design

Introduction

Signal:- variation of Physical parameters with time.

signals needs to be processed in order to store or reach to the other end user.

Electronic systems-Analogue and digital

Analogue signals vary continuously and can take any value within defined limits.

They can may infinite value within the specified range at any particular instant of time.

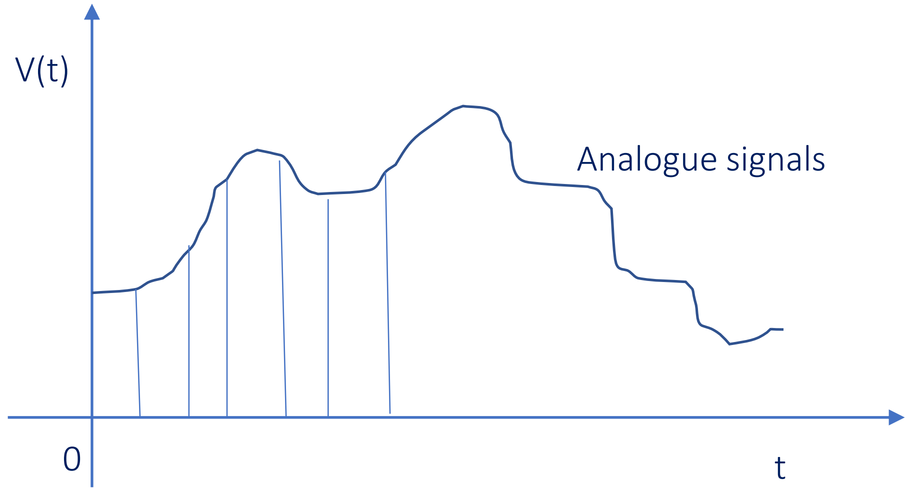
What is digital system?

Digital signals are represented in “0” and “1”.

Why digital systems?

Where are they used?

What to do with analogue signal/systems?



Discrete time signals

Advantages of digital systems

- Easier to design- Switching circuit which has only two voltage levels (High and Low) are easier to design. Exact value is not important, only range in which they fall is important.
- Storage of information is easy- Various semiconductor and magnetic memories with large capacity to store data.
- Accuracy and precision is more-Digital systems can be easily expanded to handle more digits by adding more switching circuits
- More versatile: Same hardware can be used for different applications by just changing the programme
- Less affected by the noise- noise is not critical as long as it is not large enough to prevent us from distinguishing a High from Low.
- Cost effective

Building blocks of digital circuit and design

- Analysis
- Use
- Design

Number systems

- Decimal
- Binary
- Octal
- Hexadecimal

Signed Binary Numbers

Signed-Magnitude System

- Positive numbers (including 0) can be represented as unsigned numbers.
- Negative numbers need notations.
- In ordinary arithmetic, negative number is indicated by “minus” sign and positive number by “plus” sign.
- Computer understand only binary digits because of hardware limitations.
- A bit is placed at left most side of number to represent its sign
  - “0” for positive number
  - “1” for negative number
- This is known as signed-magnitude convention.
  1. String of bits 01001 can be considered as 9 (unsigned) or +9 (signed) because left most bit is 0.
  2. String of bits 11001 represent binary equivalent of 25 (unsigned) and binary equivalent of -9 (signed)
- Positive number always starts with “0”.

Signed-Complement System

- Signed-magnitude convention needs separate handling of sign and magnitude, therefore signed-complement system used to represent negative numbers.
- In this system negative number is represented by its complement.
  - Singed-magnitude system negates a number by changing its sign.
  - Signed-complement system negates a number by taking its complement.
- Signed-complement system can use either 1's or 2's complement of number but 2's is most common.
- Ex. Consider a number 9 represented in eight-bit binary.
  - +9 is represented with a sign-bit 0 in left most position followed by binary equivalent of 9 = 00001001
  - All eight bits have must have value therefore zeros are inserted following sign-bit up to first 1.

Signed-complement system

- Although there is only one way to represent positive number, e.g., +9 (00001001),
- There are three different ways to represent negative number.
- Three different ways to represent -9 with eight-bits are
  - Signed-magnitude representation: 10001001
  - Signed-1’s complement representation: 11110110
  - Signed-2’s complement representation: 11110111
- Using four-bits we can represent 16 binary numbers.
- In signed-magnitude and 1’s complement representation, there are eight positive and eight negative numbers including two zeros.
- In 2’s complement representation, there are eight positive numbers including one zero and eight negative numbers.

Arithmetic addition

- Addition of two numbers in signed-magnitude system is similar to ordinary arithmetic.
  - Ex. 25+(-37)
  - Requires comparison of signs and magnitude and then performing either addition or subtraction.
  - Similar procedure is followed in signed-magnitude system.
- Signed-complement system does not need any comparison or subtraction but just addition with sign-bit included.
- Addition of two signed binary numbers with negative numbers.
  - Take 2’s complement of negative number
  - Add two numbers.
  - Discard a carry out of sign-bit position

Arithmetic addition

- Ex.1

+ 6	00000110
+13	00001101
+19	00010011

- Ex.2

- 6	11111010
+13	00001101
+ 7	100000111

1's complement of 6 = 11111001

2's complement of 6 = 11111010

- Ex.3

+ 6	00000110
- 13	11110011
- 7	11111001

1's complement of 13 = 11110010

2's complement of 13 = 11110011

- Ex.4

- 6	11111010
- 13	11110011
- 19	111101101

- Number of bits that hold a number is finite, result that exceed the finite value can not be accommodated.
- To obtain correct answer, the result should have sufficient number of bits to accommodate the sum.

Arithmetic Subtraction

- To determine the value of negative number in signed-2's complement, it is necessary to convert it into positive number.
    - Ex. 11111001 is negative because left most bit is 1.
    - Its 2's complement is 00000111, which is binary equivalent of +7.
    - Therefore, 11111001 is -7.
    - The procedure is same as that of addition.
- Complement of negative number in complement form produces an equivalent positive number.
- Subtraction of two signed binary numbers with negative numbers.
    - Take 2's complement of negative number
    - Add two numbers.
    - Discard a carry out of sign-bit position

Arithmetic Subtraction

- Ex.

```
-   6        11111010
- - 13       11110011
------       --------
+   7
```

- Ex.

```
 11111010
 00001101
---------
100000111
```

1's complement of 6 = 11111001

2's complement of 6 = 11111010

1's complement of 13 = 11110010

2's complement of 13 = 11110011

- Binary numbers in signed-complement system are added and subtracted by the same basic addition and subtraction rules, therefore computer needs one common hardware circuit for both types of arithmetic.

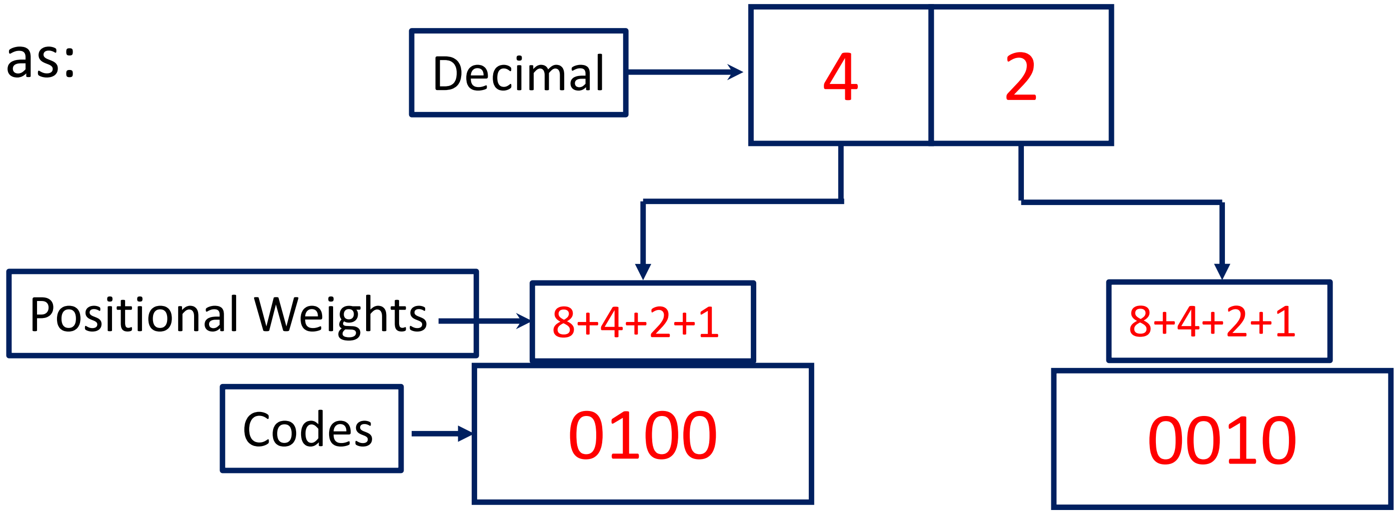
Binary Codes

- Encoding - when numbers, letters or words are represented by a specific group of symbols, and the group of symbols is called as a code.
- Digital system represent and manipulate not only binary numbers but also many other discrete elements of information.
- Any discrete element of information which is distinct among the group of quantities can be represented as **binary code** i.e. pattern of “0”s and “1” s.
- Binary code is represented by the number as well as alphanumeric letter, it merely change the symbol not the meaning of the information.
- A *n*-bit binary code is a group of *n*-bits, with 2^n distinct combinations of 1’s and 0’s.
- Each combination represent one element of the set being coded.
- Set of four elements can be coded with two bits with bit combination: 00,01,10,11.
- Set of eight elements can be coded with three bit, set of sixteen elements can be coded with four bits.

Classification of binary codes

Binary codes are broadly categorized as:

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



8,4,2,1 corresponds to power of two values of each bit

Weighted Codes

- Each bit position is assigned a weighting factor so that each bit can be evaluated by adding weights of all 1's in the coded combinations.
- BCD code has weights of 8,4,2,1 which corresponds to power-of -two values of each bit.
- Ex. 0110 can be interpreted by weight to represent decimal number=

8 x 0+ 4x 1+ 2x 1+ 1x0 = 6

Binary Coded Decimal -*BCD* code

- It is possible to perform arithmetic operations directly on decimal numbers when they are stored in computer in coded form.
- Binary code will have some unassigned bit combinations if the number of elements in that set is not multiple power of 2.
- 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.
- Decimal number in BCD is the same as its equivalent binary only when number is between 0 to 9.
- In the binary, 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	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
BCD	0001	0010	0011	0100	0101	0110	0111	1000	1001

Example:

$(165)_{10}$= $(0001\ 0110\ 0101)_{BCD}$=$(10100101)_2$

- BCD value has 12 bit to encode the characters of decimal value where as, equivalent binary number needs only 8 bits.
  - BCD needs more bits than binary

BCD Addition

- When the binary sum is greater than equal to 1010, the result is invalid.
- Addition of 6=$(0110)_2$, the binary sum converts it to the correct digit and produces carry as required.
- A carry in the most significant bit position of binary sum and a decimal carry differs by 16-10=6.

$$\begin{array}{rrrrrr}
2 & 0010 & 5 & 0101 & 9 & 1001\\
\underline{+3} & +\ \underline{0011} & \underline{+8} & +\ \underline{1000} & \underline{+9} & +\ \underline{1001}\\
5 & 0101 & 13 & 1101 & 18 & 10010\\
 & & & \underline{+\ 0110} & & \underline{+\ 0110}\\
 & & & 10011 & & 11000
\end{array}$$

Addition of two *n*-digit unsigned numbers.

$$\begin{array}{rrrr}
 & 1 & 1 & \\
184 & 0001 & 1000 & 0100\\
\underline{+576} & +\ \underline{0101} & \underline{0111} & \underline{0110}\\
760 & 0111 & 10000 & 1010\\
 & \underline{\hphantom{0111}} & \underline{0110} & \underline{0110}\\
 & 0111 & 0110 & 0000
\end{array}$$

BCD numbers are decimal number and not binary although they use bits in their representation.

Advantages of BCD Codes

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

Disadvantages of BCD Codes

- Addition and subtraction of BCD have different rules, so more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.
- It is not possible to add any two numbers.

Signed Decimal Numbers

- It is customary to represent plus with four 0's and minus with BCD equivalent of nine, which is 1001.
- Signed complement system can either 9's or 10's but 10's is most often used.
- The procedure for signed-10's complement system is similar to the 2's complement system.
- Ex. Addition of (+375) + (-240)

  - Ex.1

```
+  375          0  375        10's complement of 240 =760
-  240         +9  760
               ---------
                0  135        (0000000100110101)BCD
```

$$\begin{array}{r} +\ 375 \\ \underline{-\ \ 240} \end{array} \qquad \begin{array}{r} 0\ \ 375 \\ \underline{+9\ \ 760} \\ 0\ \ 135 \end{array} \qquad (0000000100110101)_{BCD}$$

- Decimal numbers inside the computer, including sign-digit must be in BCD.
- Many computers have special hardware to perform arithmetic calculations directly with decimal numbers in BCD.

Non-Weighted Codes

In this type of binary codes, the positional weights are not assigned.

Ex.:- Excess-3 code and Gray code.

Excess-3 code

- The Excess-3 code is also called as XS-3 code or self complementing code.
- It is non-weighted code i.e. there is no relation between position of a digit and its weight
- It is also used to express decimal numbers.
- Such code has property that, 9’s complement of the decimal number is obtained directly by changing 1’s to 0’s and 0’s to 1’s i.e. complementing each bit in the pattern.
- ex. 4 and 5 are 9’s complement and in XS-3 code 0111 and 1000.
- XS-3 code has been used in old computers because of its self complementing property.
- No number is expressed as 0000, which helps checking error, if any.
- The Excess-3 code are derived from the 8421 BCD code by adding 0011 to each code in 8421.

Excess-3 code

Encoding decimal number into XS-3 code

- Add 3 to each decimal digit
- In case, addition of 3 produces carry, then do not carry it to the next higher place, keep it as it is in its position.
- Convert each digit into its four bit binary equivalent.
- In case of carry, convert both the digits at the same time.

Ex. 1. 5

i) Add 3: 5+3 = 8

ii) Convert to binary:1000

$(5)_{10} = 1000$

Ex. 2. 6

i) Add 3: 6+3 = 9

ii) Convert to binary:1001

$(6)_{10} = 1001$

Ex. 3. 9

i) Add 3: 9+3 = 12

ii) Convert to binary:1100

$(9)_{10} = 1100$

Ex. 4. 89

i) Add 3: 8+3 = 11 and 9+3 = 12

ii) Convert to binary:11 = 1011 and 12 = 1100

$(89)_{10} = (1011\ 1100)$

Ex. 5. 27

i) Add 3: 2+3 = 5 and 7+3=10

ii) Convert to binary:5=0101 and 10 = 1010

$(27)_{10} = (0101\ 1010)$

Excess-3 code-Addition

- Add XS-3 numbers by adding 4-bits in each column starting from LSD.
- If there is no carry, subtract 0011 from the sum.
- If there is carry, add 0011 to the sum because when there is carry invalid states are skipped and result is in normal binary.

$$\begin{array}{r r}
5 & 1000 \\
\underline{+3} & +\,\underline{0110} \\
5 & 1110 \\
 & \underline{-\ 0011} \\
 & 1011
\end{array}$$

$$\begin{array}{r r r}
37 & 0110 & 1010 \\
\underline{+28} & +\,\underline{0101} & \underline{1011} \\
65 & 1100 & 0101 \\
 & \underline{-\ 0011} & \underline{+0011} \\
 & 1001 & 1000
\end{array}$$

XS-3 code for 5= 5+3=8
XS-3 code for 3= 3+3=6
XS-3 code for 7= 7+3=10
XS-3 code for 8= 8+3=11
XS-3 code for 2= 2+3=5

Excess-3 code-Subtraction

- Subtraction is performed by 10’s complement method
- Take 10’s complement of the number to be subtracted
- Add two numbers.
- Follow the steps for addition.

EX.1

$$\begin{array}{r} 5 \\ \underline{-3} \\ 2 \end{array} \qquad \begin{array}{r} 5 \\ \underline{+7} \\ \textcircled{1}2 \end{array} \qquad \begin{array}{r} 1000 \\ +\ \underline{1010} \\ \textcircled{1}0010 \\ \underline{+\ 0011} \\ 0101 \end{array}$$

10’s complement of 3= 7

2’s complement of 0110 (3) in XS-3 code= 1010

EX.2

$$\begin{array}{r} 687 \\ \underline{-348} \\ 339 \end{array} \qquad \begin{array}{r} 687 \\ \underline{+652} \\ \textcircled{1}339 \end{array}$$

687 in XS-3 code

2’s complement of 348 in XS-3 code

$$\begin{array}{rrr} 1001 & 1100 & 1010 \\ \underline{+1001} & \underline{1000} & \underline{0101} \\ \textcircled{0}0011 & 0011 & 1111 \\ \underline{+0011} & \underline{+0011} & \underline{-0011} \\ 0110 & 0110 & 1100 \end{array}$$

339 in XS-3 code

Gray Code

- It is non-weighted code, means no specific weights assigned to bit position.
- Gray code is used to represent digital data that has been converted from analogue data.
- Advantage of Gary code over straight binary number sequence is that, only one bit in code group changes in going from one number to next, therefore also, called as Unit Distance Code or reflected code.
- Gray code cannot be used for arithmetic operation.
- Gray code is used in applications where normal sequence of binary numbers may produce an error during transition from one number to next.
- Ex. Change of binary number from 0111 to 1000 may produce immediate enormous number 1001, if rightmost bit takes more time to change than rest of the three.

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

Alphanumeric codes

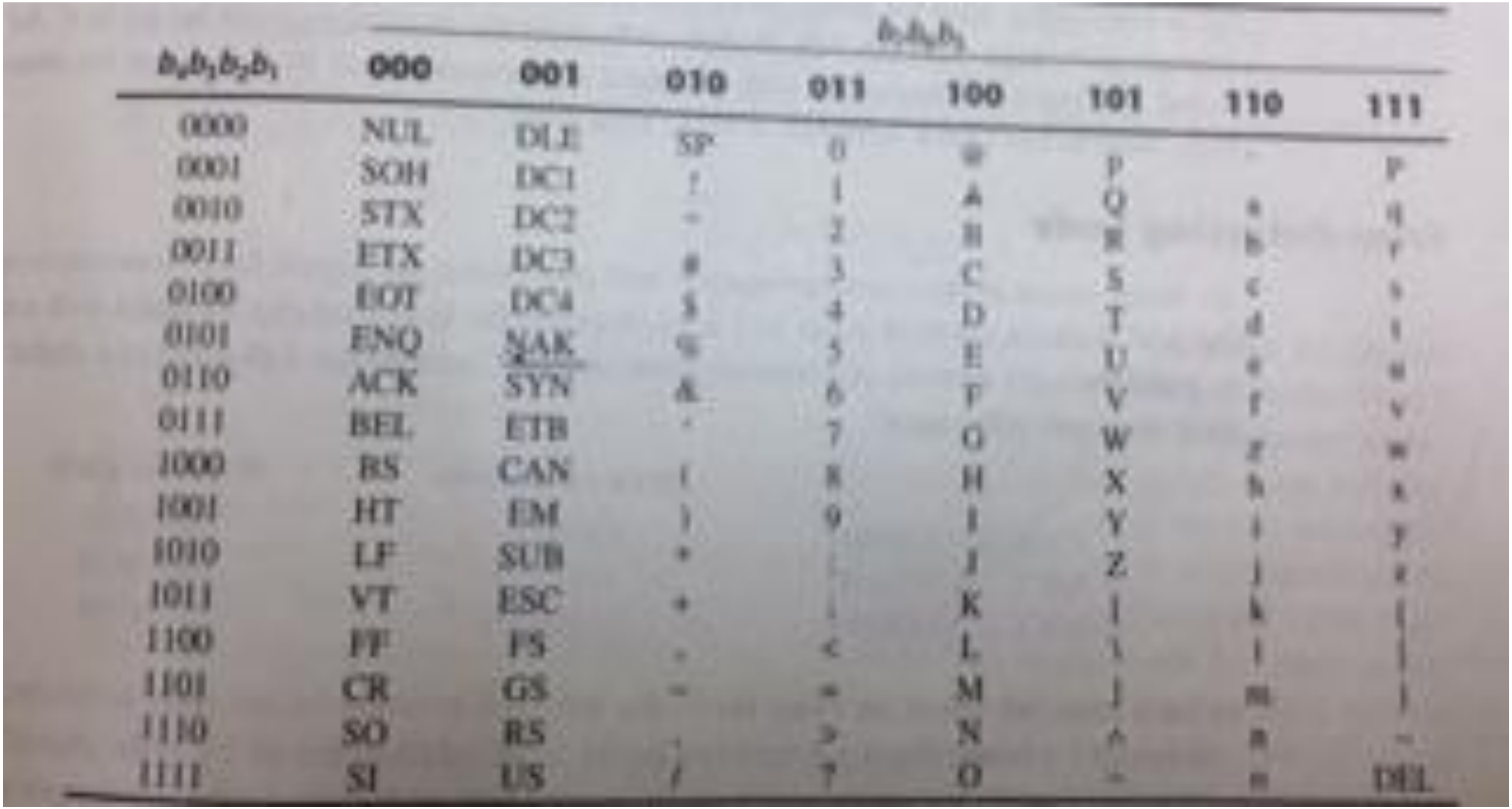
- A binary digit or bit can represent only two symbols as it has only two states '0' or '1'. But this is not enough for communication between two computers because there we need many more symbols for communication.
- These symbols are required to represent 26 alphabets with capital and small letters, numbers from 0 to 9, punctuation marks and other symbols.
- An alphanumeric character is set of elements that include 10 decimal digits, 26 letters of alphabets and number of special characters.
- Such set contains between 36 and 64 elements if only capital letters are included and between 64 and 128 if both uppercase and lowercase letters are included.
- First case needs binary code of six bits and second needs binary code of seven bits.

ASCII code

- Standard binary code for alphanumeric characters is the American Standard Code for Information Interchange (ASCII), uses seven bits to code 128 characters.
- Seven bits of code are designated by b1 through b7, with b7 MSB.
- ASCII code contains 94 graphic characters that can be printed and 34 nonprinting characters for various control functions.
- Graphic characters consist of 26 uppercase letter, 26 lowercase letters, 10 numerals and 32 special printable characters such as %,*, and $.
- 34 control characters-ex. ESC-escape, BS-Backspace, HT- horizontal tab etc.
- Control characters: Used for routing the data and arranging printed text into prescribed format.
- Three types of control characters: Format effector, Information Separator and communication-control characters.
- Format effectors: Control layout of printing e.g. backspace (BS), Horizontal Tabulation (HT), Carriage return (CR).

ASCII code

- Three types of control characters: Format effector, Information Separator and communication-control characters.
- Format effectors: Control layout of printing e.g. backspace (BS), Horizontal Tabulation (HT), Carriage return (CR).
- Information Separator: Used to separate data into divisions such as paragraphs and pages. eg. Record separator (RS), file separator (FS).
- Communication-control characters: Used during the transmission of text between remote terminals. eg. Start of text (STX) and end of text (ETX), which are used to frame a text message transmitted through telephone wires.
- ASCII is seven bit code and most computers manipulate eight-bit quantity called as “byte”, therefore ASCII characters often stored one per byte.
- Extra bit is used for other purposes depending on applications.
- Some printer recognizes eight-bit ASCII characters with MSB 0.
- ASCII characters with MSB 1 can be used for symbols such as Greek Alphabets or italic-type font.



	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Error-Detecting codes

- Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted.
- To avoid this, we use error-detecting codes, which are additional data added to a given digital message to help us detect, if an error occurred during transmission of the message.
- To detect error in data communication and processing, an eighth bit sometime added to the ASCII character to indicate its parity.
- A parity bit is an extra bit included in the message to make total number of 1's either even or odd.

	With even parity	with odd parity
• ASCII A =1000001	01000001	11000001
• ASCII T=1010100	11010100	01010100
• ASCII a= 1100101	01100101	11100101

- In general, one of the parity is adopted but even parity is more common.
- Parity bit helps detecting error during transmission of information from one location to another.
- Error detection
  - Generating even parity bit at sending end for each character.
  - Eight-bit character including parity bit is transmitted to the destination.
  - Parity of each character is checked at receiving end.
  - If the parity of received character is not even then at least one bits has changed the value during transmission.
  - This method detects odd combinations of errors in each character that is transmitted.
  - Even combinations of errors goes undetected and additional error-detection code is needed for this.

Error correction

- Request retransmission of message assuming error was random and will not occur again.
- If receiver detects parity error, it sends back ASCII NAK (negative acknowledge) control character consisting of even-parity bit 10010101.
- If no error is detected, receiver sends back an ACK (acknowledge) control character 00000110.
- Sending end will respond to NAK by transmitting the message again until the correct parity is received.
- If error still occurs after number attempts, message can be sent to operator to check for malfunctions in transmission path.