DIGITAL ELECTRONICS

Number System:

A number system is defined as **a system of writing to express numbers**. A number system is a mathematical system with base n, where n represents total numbers present in that system.

Eg: In decimal number system, the number of digits is 10

Radix Point:

Radix point is a base point used to separate the integer part and the fractional part of a number.

Base or radix:

The number of symbols used in a number system is called base or radix of the number system.

Eg: In decimal number system – Base is 10
In Binary number system – Base is 2
In Hexadecimal number system – Base is 16 etc.

Most significant bit (MSB):

The left most bit of a binary number which has the highest place value is called most significant bit.

Least significant bit (LSB):

The right most bit of a binary number which has the least place value is called least significant bit.

Bit:

A binary digit is called a bit.
Example: 0 or 1

Nibble:

A group of four bits is called a Nibble.
Example: 0110, 1110 etc.

Byte:

A group of eight bits is called a byte.
Example: 01101101, 11010011 etc.

Types of number systems

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

Binary number system:

The base of binary number system is **2**. The digits used are 0 and 1.
Example: $11001_{(2)}$, $0101_{(2)}$

Octal number system:

The base of octal number system is **8**.
The digits used are 0, 1, 2, 3, 4, 5, 6 and 7.
Example: $157_{(8)}$, $2312_{(8)}$, $110_{(8)}$

Table representing decimal, Octal and binary number

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

Decimal number system:

The base of decimal number system is **10**.

The digits used are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Example: $0157_{(10)}$, $6587_{(10)}$, $0110_{(10)}$

Hexadecimal number system:

The base of hexadecimal number system is **16**.

The digits used are 0, 1, 2, 3, 4, 5, 6, 7, 8 9, A, B, C, D, E and F.

Example: $0157_{(16)}$, $2312_{(16)}$, $0110_{(16)}$, $ABC5_{(16)}$, $BDEF_{(16)}$

Table representing decimal, hexadecimal and binary number

Decimal Number	Hexadecimal Number	Binary Number
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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Binary Arithmetic

Binary arithmetic includes the basic arithmetic operations of **addition, subtraction, multiplication and division**

Binary Addition: The rules to perform binary addition are

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 with a carry 1

Binary Subtraction: The rules to perform binary subtraction are

0 – 0 = 0
0 – 1 = 1 with a borrow 1
1 – 0 = 1
1 – 1 = 0

Binary Multiplication: The rules to perform binary multiplication are

0 X 0 = 0
0 X 1 = 0
1 X 0 = 0
1 X 1 = 1

Binary Division: The rules to perform binary division are

0 ÷ 0 = not defined
0 ÷ 1 = 0
1 ÷ 0 = not defined
1 ÷ 1 = 1

1's Complement of binary numbers:

The 1's complement of a binary number is obtained by changing each 0 to 1 and 1 to 0.

Eg: The 1's complement of $(101101)_2$ is $(010010)_2$

2'S Complement of binary numbers: The 2'S complement of a binary number is obtained by adding 1 to the 1's complement of the number.

Eg: The 2's complement of $(101101)_2$ is $(010010)_2 + 1 = (010011)_2$

Note: The 1'S complement and the 2'S complement of a binary number help to do the subtraction by the method of addition.

BINARY CODES

Digital Codes or Binary Codes:

A group of binary digits is known as a binary code or a digital code.

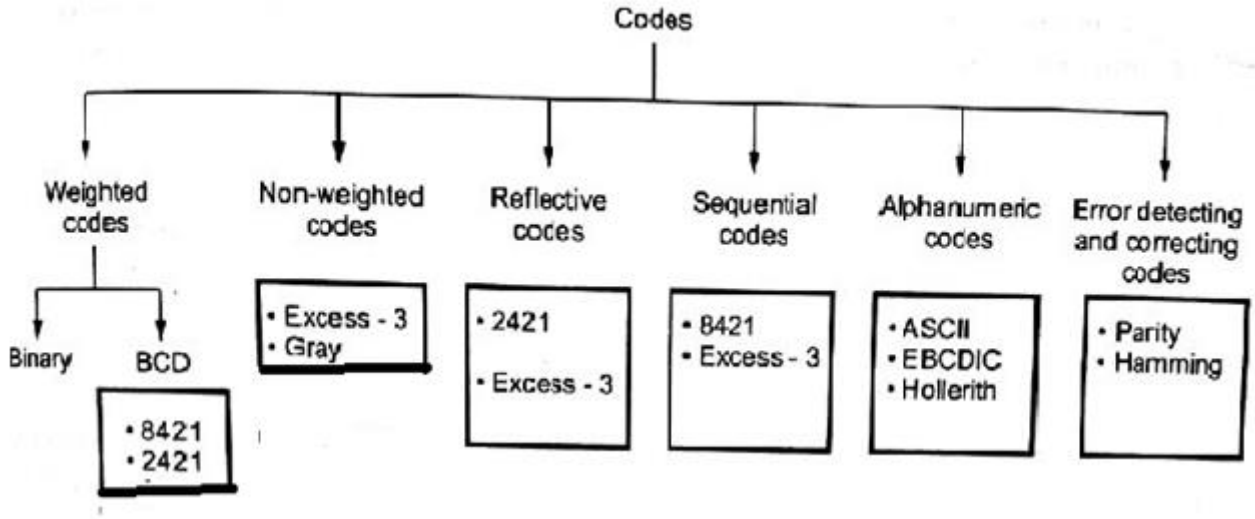
Numeric Code:

Digital code represented by the number is known as a numeric code.

Classification:

The codes are broadly classified into

1. Weighted code
2. Non-weighted code
3. Reflective code
4. Sequential code
5. Error detecting and correcting codes



Weighted codes:

The weighted codes are those where the position of each number represent a specific weight. In these codes each decimal digit is represented by a group of four bits.

In weighted codes, each digit is assigned a specific weight according to its position. For example, in 8421/BCD code, 1001 the weights of 1, 1, 0, 1 (from left to right) are 8, 4, 2 and 1 respectively.

Examples:8421, 2421.

BCD and 2421 Code

Decimal Equivalent	BCD Code				2421 Code			
	D_4	D_3	D_2	D_1	Y_4	Y_3	Y_2	Y_1
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	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

Non-weighted codes:

The non-weighted codes are not positionally weighted. In other words, codes that are not assigned with any weight to each digit position.

Example: Excess-3(XS-3) and Gray Codes.

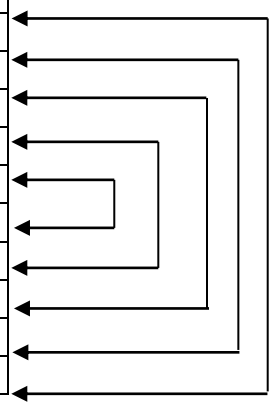
Excess-3 code

Excess-3 codes are Non-weighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary number is obtained using the following steps:

- Find the decimal equivalent of the given binary number.
- Add +3 to each digit of decimal number.

- Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.
- These are following excess-3 codes for decimal digits –

Decimal	BCD 8421	Excess – 3 code BCD + 0011
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100



Gray code:

Gray code is non weighted code, which means there is no specific weight assigned to the bit position.

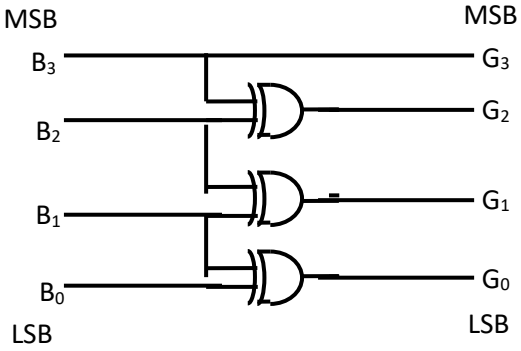
Applications:

- It is used in analog to digital conversion, input / output devices.
- It is used to reduce errors that occur in data transmission.

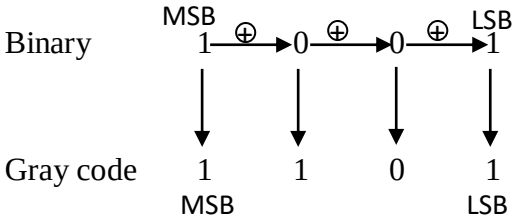
Binary to gray code conversion

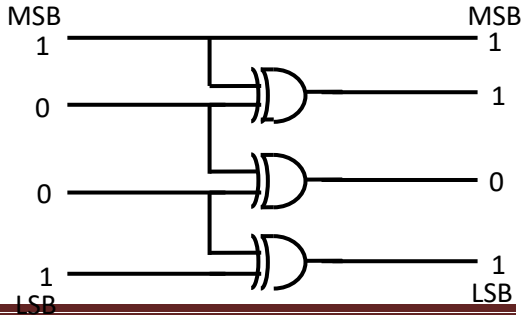
- The MSB of the gray code is the MSB of the binary number.
- Perform XOR operation between the MSB and the second significant bit of the binary number.
- XOR the second and third significant bits of the binary, the result is the third significant bit of the gray code.
- Repeat the process till the end of the LSB of the binary number

.



Example:

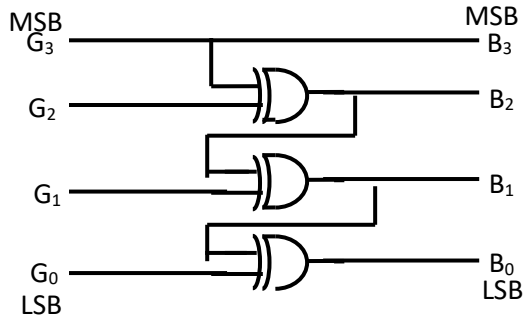




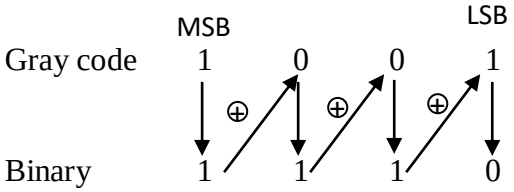
Gray code to binary conversion

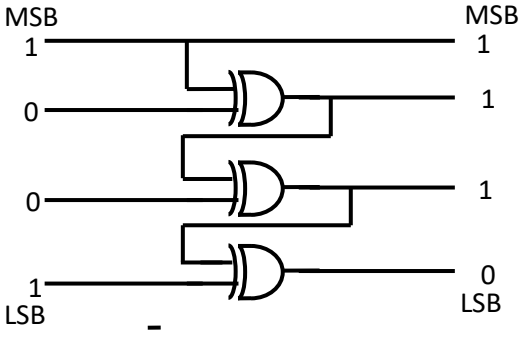
- The MSB of the binary is same as the MSB of gray code.
- Perform the XOR operation between the MSB of the binary and the second significant bit of gray code the result is the second significant bit of binary.
- Perform the XOR operation between the second significant bit of binary and the third significant bit of gray code the result is the third significant bit of binary.
- Repeat the process until all the gray code bits are XOR.

Circuit diagram of gray code to binary conversion



Example:



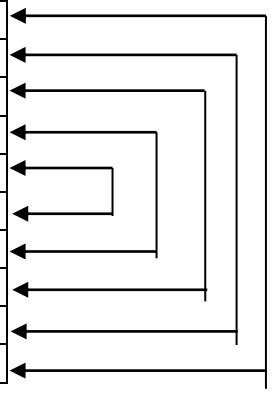


Reflective codes:

A code is said to be reflective when the code for 9 is the complement for the code 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.

Eg: 2421 code and Excess-3 code

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



Sequential codes:

A code is said to be sequential when each succeeding code is binary number greater than its preceding code.

Eg: 8421 code and Excess-3 code

Parity codes:

The parity code is used for the purpose of detecting errors during the transmission of binary information. The parity code is a bit that is included with the binary data to be transmitted.

The inclusion of a parity bit will make the number of 1's either odd or even. Based on the number of 1's in the transmitted data, the parity code is of two types.

- Even parity code
- Odd parity code

Even parity code:

If the total number of 1 bits in the word including parity bit is Even, then such a parity code is said to be **Even parity code**.

Odd parity code:

If the total number of 1 bits in the word including parity bit is odd, then such a parity code is said to be **odd parity code.**

The following table shows the even and odd parity bits for 4 bit data word

Data	Even parity	Odd parity
0 0 0 0	0	0
0 0 0 1	1	0
0 0 1 0	1	0
0 0 1 1	0	1
0 1 0 0	1	0
0 1 0 1	0	1
0 1 1 0	0	1
0 1 1 1	1	0
1 0 0 0	1	0
1 0 0 1	0	1

The simple parity will not detect two errors within the same word. To detect and correct error in the word Hamming code is used.

Hamming code:

Hamming code is a **set of error-correction codes** that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is technique developed by R.W. Hamming for error correction.

Selecting the number of parity bits:

The number of parity bits to be chosen depends on word length.

Let n be the number of information or data bits, then the number of parity bits P is determined from the following formula,

$$2^P \geq n + P + 1$$

Example:

If 4-bit information is to be transmitted, then n=4. The number of parity bits is determined by the trial and error method

Let P=2, we get,

$$2^2 \geq 4 + 2 + 1$$

The above equation implies 4 not greater than or equal to 7. So let's choose another value of P=3.

$$2^3 \geq 4 + 3 + 1$$

Now, the equation satisfies the condition. So number of parity bits, P=3.

The bit designation is

D7	D6	D5	**P3**	D1	**P2**	**P1**

Problem:

Data bits 1011 must be transmitted. Construct the even parity, seven bit Hamming code for this data

D7	D6	D5	**P3**	D1	**P2**	**P1**
1	0	1		1		

P1 checks for even parity of bit positions 1,3,5 and 7. For even parity **P1 must be 1**

D4	D3	D2	**P3**	D1	**P2**	**P1**
1	0	1		1		**1**

P2 checks for even parity of bit positions 2,3,6 and 7. For even parity **P2 must be 0**

D4	D3	D2	**P3**	D1	**P2**	**P1**
1	0	1		1	**0**	**1**

P3 checks for even parity of bit positions 4,5,6 and 7. For even parity **P3 must be 0**

D4	D3	D2	**P3**	D1	**P2**	**P1**
1	0	1	**0**	1	**0**	**1**

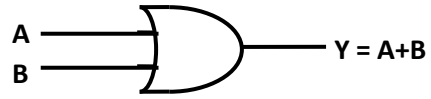
LOGIC GATES

Logic gate is an electronic circuit that performs a Boolean logical operation. A logic gate has one or more inputs but only one output.

- The basic gates are AND gate, OR gate and NOT gate.
- The universal gates are NAND gate and NOR gate.
- The special gates are XOR and XNOR.

OR gate:

OR gate is a basic gate which has two or more input but only one output. The output is high if any one of the input is high.



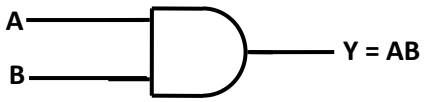
Logic symbol

Input		Output
A	B	**Y = A+B**
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

AND gate:

AND gate is a basic gate which has two or more input but only one output. The output is high if all the inputs are high.



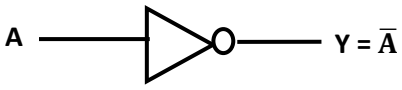
Logic symbol

Input		Output
A	B	**Y = AB**
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

NOT gate:

It is a basic gate which complements the input signal value. It is also called as an inverter. It is a logic gate with only one input and one output.



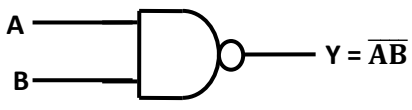
Logic symbol

Input	Output
A	**Y = $\overline{A}$**
0	1
1	0

Truth table

NAND gate:

NAND gate is a universal gate which performs complement of AND logic. NAND gate has two or more input but only one output. The output is high only when any of the input is low.



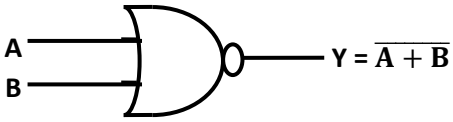
Logic symbol

Inputs		Output
A	B	**Y = $\overline{AB}$**
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

NOR gate:

NOR gate is a universal gate which performs complement of OR logic. NOR gate has two or more input but only one output. The output is high if and only if all the inputs are low.



Logic symbol

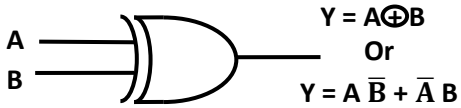
Inputs		Output
A	B	Y = $\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

Exclusive OR (XOR) gate:

XOR gate is a special gate which has two or more input but only one output. The output is high only when odd numbers of input are high.

XOR gate is also called as **Inequality detector**.



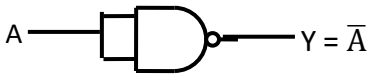
Logic symbol

Inputs		Output
A	B	Y = A $\overline{B}$ + $\overline{A}$ B
0	0	0
0	1	1
1	0	1
1	1	0

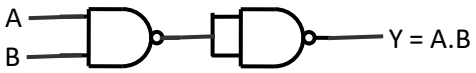
Truth table

Realization of NOT, AND, OR and XOR gates using NAND gate.

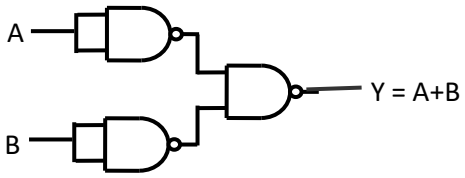
The NAND gate as a NOT gate



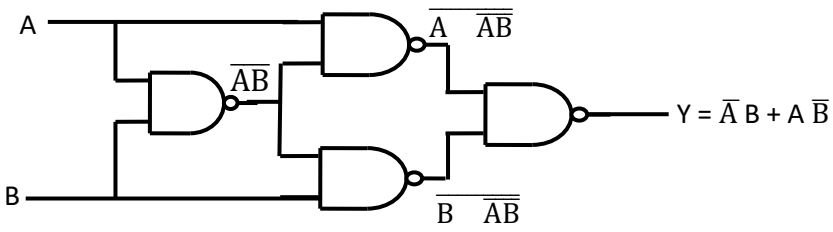
The NAND gate as an AND gate



The NAND gate as an OR gate

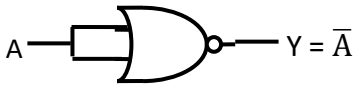


The NAND gate as an XOR gate

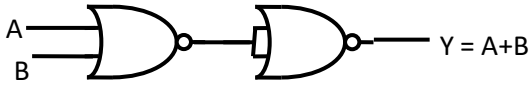


Realization of NOT, AND, OR and XNOR gates using NOR gate.

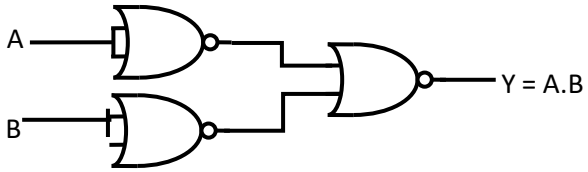
The NOR gate as a NOT gate



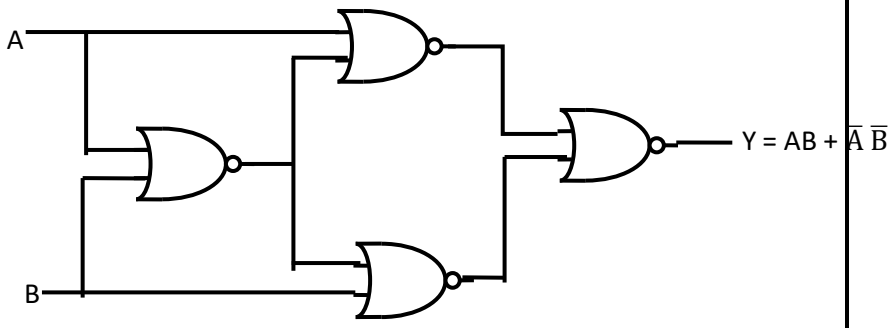
The NOR gate as a OR gate



The NOR gate as a AND gate



The NOR gate as a XNOR gate



Simplification of Boolean expressions

Product term: The logical product of Boolean variables, complemented or uncomplemented form is a product term.
Example: A $\overline{B}$ C, A B C, A B $\overline{C}$, A B, $\overline{A}$ $\overline{B}$ etc

Sum term: The logical sum of Boolean variables, complemented or uncomplemented form is a sum term.
Example: (A + $\overline{B}$ + C), (A + B), ($\overline{A}$ + $\overline{B}$ + C)

Sum of products (SOP): The logical sum of two or more logical product terms is known as sum of products.
Example: A $\overline{B}$ + $\overline{C}$ A + A B

Product of sum (POS): The logical product of two or more logical sum terms is known as product of sum.
Example: (A + $\overline{B}$) (B + C) (A + B)

Canonical SOP (or) Standard SOP expression: The Boolean expression containing all the input variables in each of the product term either in complemented or uncomplemented form is known as canonical SOP expression.

Canonical POS (or) Standard POS expression: The Boolean expression containing all the input variables in each of the sum term either in complemented or uncomplemented form is known as canonical POS expression.