Chapter 1-**Introduction**

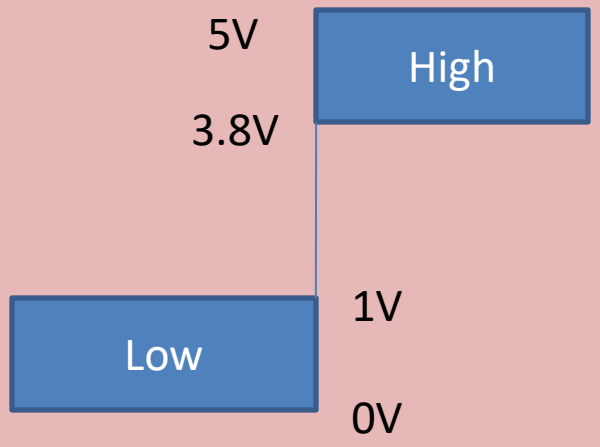
- **Digital system**
- **Analog system**
- **Need of Digital system/advantages of digital systems**
- **Applications of digital systems**

Digital signal and digital system

- Digital signal: The signal which can have any of the two discrete voltage levels. One of these levels is Low level and other is High level.
- Digital system: The system used to process a digital signal is called a digital system.

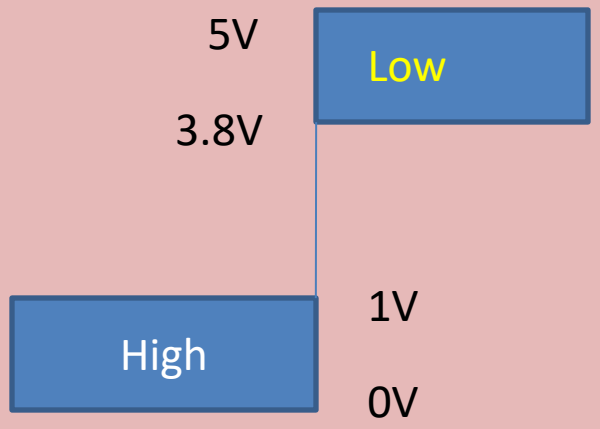
Positive Logic

- In digital system,if lower voltage level is used to represent Low and higher voltage level is used to represent High,it is called positive Logic.



Negative Logic

- In digital system, if lower voltage level is used to represent High and higher voltage level is used to represent Low ,it is called Negative Logic.



Need of digital systems/Advantages of digital systems.

- The devices used in digital systems operates in ON or OFF state.
- There are only few basic operations in digital systems.
- Digital techniques use boolean algebra-easy to understand .
- A large no of ICs are available for performing various operations.

- Effect of ageing ,fluctuations, temperature and noise is small in digital circuits.
- Digital systems have storage capability.
- Display of data and information is convenient.
- Latest electronic systems are digital in nature.
- Designing and development of digital systems is simple due to availability of various logic families.

Applications of Digital systems.

- Computers.
- Medical Electronics.
- Instrumentation.
- Communication e.g. Mobile Cell Phones, Internet.
- Consumer products e.g. watches,calculators,smart TV
- Information sytems
- Industrial Control systems
- Banking and finance
- Scientific Instruments.
- Office machines,homes ,cars,education etc.

Chapter 2

NUMBER SYSTEMS

Number systems

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

Decimal Number system

Base/Radix – 10

Ten symbols-0,1,2,3,4,5,6,7,8,9.

Binary Number system

- Base/radix – 2(two)
- Symbols- 0,1.
- One binary digit is called a bit.e.g. o
- Nibble: A combination of four bits.e.g.0011
- Byte: a combination of eight bits. e.g.10110011
- MSB(Most Significant bit)- the leftmost bit of the binary number
- LSB(Least Significant Bit)-The rightmost bit of the binary number

Octal Number system

- Base/radix-8
- Symbols-0,1,2,3,4,5,6,7.

Hexadecimal number system

- Base/Radix- 16
- Symbols : -

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

CHAPTER3

CODES

Codes.

- Straight Binary code
- BCD Code
- Grey Code
- Excess-3 code

Straight Binary code

- uses natural binary form. It is a weighted code. Each position has a weight-1,2,4,8,16 and so on.

BCD code:

- Decimal digits 0 through 9 are represented by their binary equivalent 4 bit code.
- E.g.23 in decimal is equivalent to 0010 0011 in BCD.
- Also known as 8421 code.-8,4,2,1 are the weights of the four bits of the code.

BCD code:

Binary 0 0 0 0	BCD 0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 0
0 0 1 1	0 0 1 1
0 1 0 0	0 1 0 0
0 1 0 1	0 1 0 1
0 1 1 0	0 1 1 0
0 1 1 1	0 1 1 1
1 0 0 0	1 0 0 0
1 0 0 1	1 0 0 1

Gray code

- Gray Code is a non weighted Code.
- The two consecutive codes differ by one bit. So also called reflected code.

Excess-3 Code

- Another form of BCD code
- The code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit

Excess-3 Code

Binary 0 0 0 0	BCD 0 0 0 0	EXCESS-3 0 0 1 1
0 0 0 1	0 0 0 1	0 1 0 0
0 0 1 0	0 0 1 0	0 1 0 1
0 0 1 1	0 0 1 1	0 1 1 0
0 1 0 0	0 1 0 0	0 1 1 1
0 1 0 1	0 1 0 1	1 0 0 0
0 1 1 0	0 1 1 0	1 0 0 1
0 1 1 1	0 1 1 1	1 0 1 0
1 0 0 0	1 0 0 0	1 0 1 1
1 0 0 1	1 0 0 1	1 1 0 0

Error detecting Codes

- Parity: It is an extra bit added to the binary code to make the number of ones in the code even or odd.
- Parity is used to detect single bit error.

BCD 0 0 0 0	BCD with even parity 0 0 0 0 0	BCD with odd parity 1 0 0 0 0
0 0 0 1	1 0 0 0 1	0 0 0 0 1
0 0 1 0	1 0 0 1 0	0 0 0 1 0
0 0 1 1	0 0 0 1 1	1 0 0 1 1
0 1 0 0	1 0 1 0 0	0 0 1 0 0
0 1 0 1	0 0 1 0 1	1 0 1 0 1
0 1 1 0	0 0 1 1 0	1 0 1 1 0
0 1 1 1	1 0 1 1 1	0 0 1 1 1
1 0 0 0	1 1 0 0 0	0 1 0 0 0
1 0 0 1	0 1 0 0 1	1 1 0 0 1

Chapter 4

Logic Gates

Logic gate

- A **logic gate** is an elementary building block of a digital **circuit**. Most **logic gates** have two or more inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels.

Various Logic gates

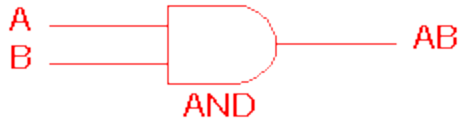
AND, OR, NOT, NAND, NOR, EXOR and EXNOR **gates**.

Truth Tables

- Truth tables help understand the behaviour of logic gates.
- They show how the input(s) of a logic gate relate to its output(s).
- The gate input(s) are shown in the left column(s) of the table with all the different possible input combinations. This is normally done by making the inputs count up in binary.
- The gate output(s) are shown in the right hand side column.

AND Gate: The output is high if all the inputs are high.It has two or more inputs and one output.

- Symbol of AND Gate(2 input)

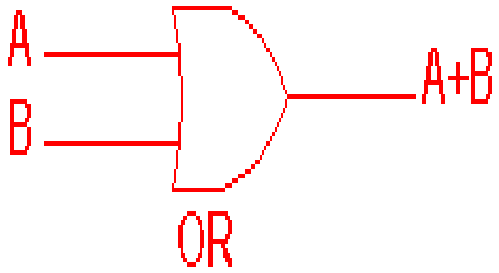


- Truth table of AND gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR- GATE: The output is high if any or all the inputs are high.It has two or more inputs and one output.

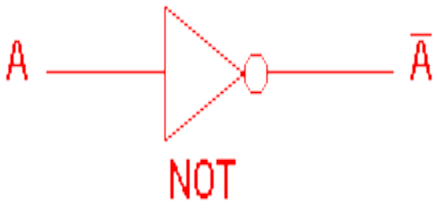
Symbol of OR gate



Truth table of OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

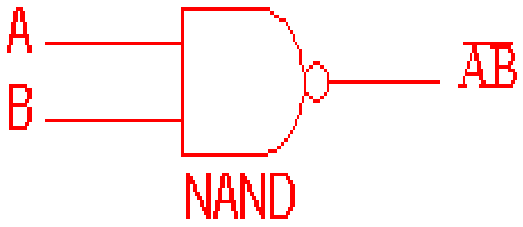
NOT GATE: It has one input and one output.



NOT gate	
A	$\bar{A}$
0	1
1	0

The output is complement of the input.

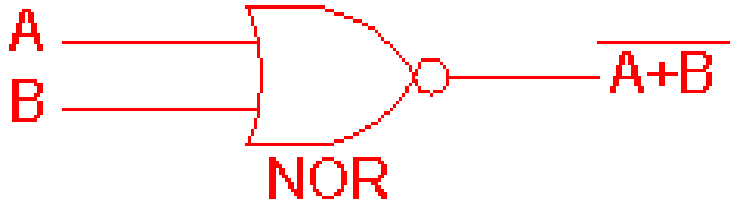
NAND GATE: The output is high if any of the inputs is low.It has two or more inputs and one output.



2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE- It is NOT-OR gate, If any input is high output is low

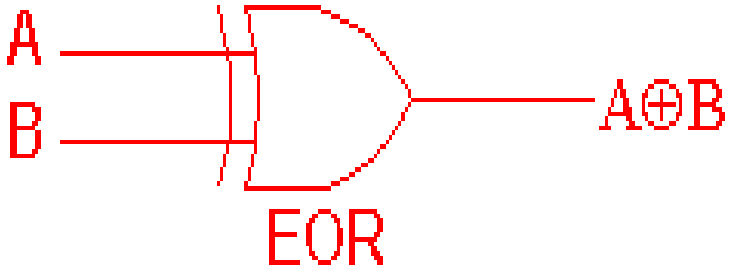
Symbol of 2 input NOR Gate



Truth Table of 2 input NOR Gate

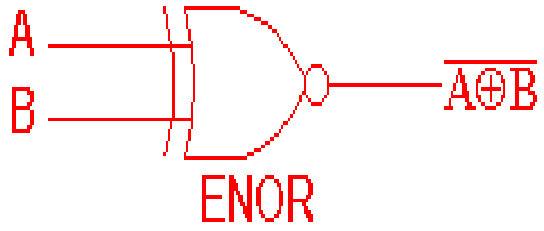
2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

EX-OR GATE



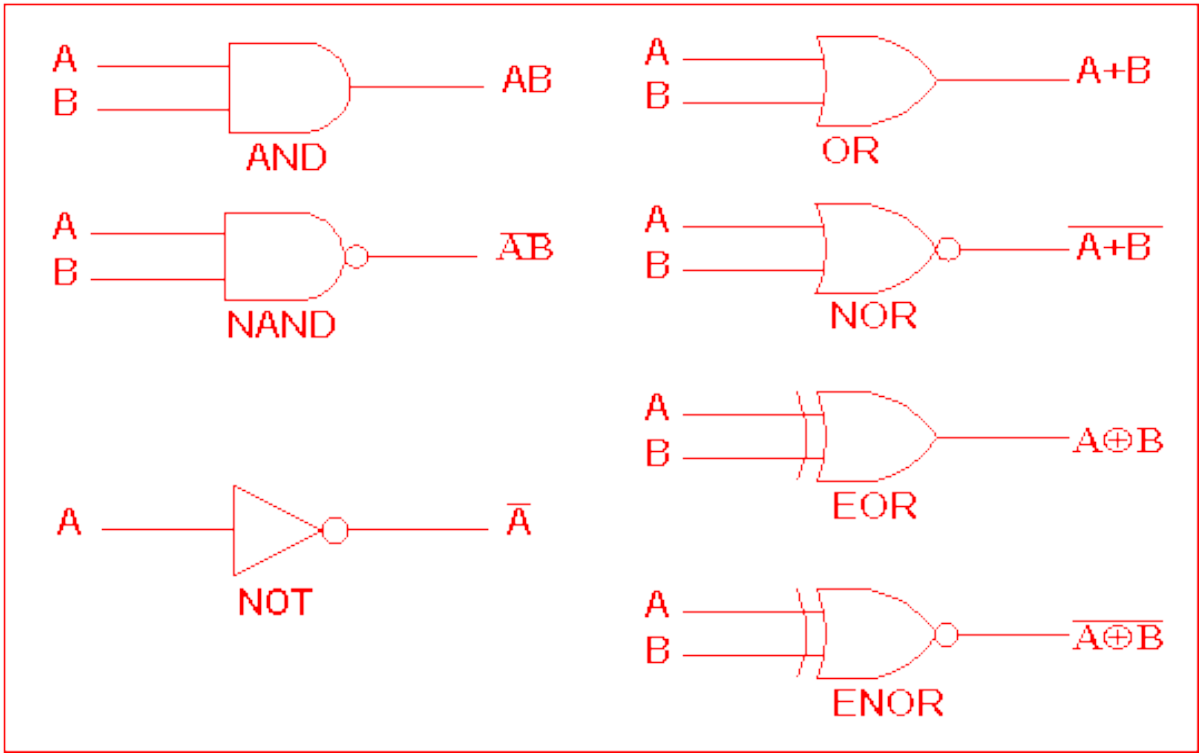
2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

EX-NOR GATE



2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Symbols of all gates

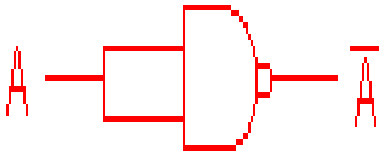


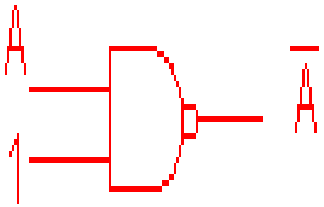
Universal Gates

NAND Gate

NOR Gate

A NAND gate can be used as a NOT gate using either of the following wiring configurations.





Chapter 5

LOGIC SIMPLIFICATION

Logic Simplification

- To simplify the logic expressions so as to minimise the number of gates required.
- There are two methods of logic simplification.
    - Using Laws of boolean Algebra
    - Using K-Map

Laws of Boolean Algebra

- Commutative Law
  - A+B=B+A(OR Law)
  - A.B=B.A(AND Law)

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Associative Law

- A+(B+C)= (A+B)+C
- A.(B.C)=(A.B).C

A	B	C	A+(B+C)	(A+B)+C
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Distributive Law

- A(B+C)=AB+AC

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

AND LAWS

- $A.0=0$
- $A.1=A$
- $A.A.=A$
- $A.\overline{A}= 0$

OR LAWS

- $A+0=0$
- $A+1=1$
- $A+A=A$
- $A+\overline{A}= 1$

NOT LAWS

- IF A=0 ,$\overline{A}$= 1
- IF A=1 ,$\overline{A}$= 0

Some other Laws

$$A + \bar{A}B = A + B$$

$$\bar{\bar{A}} = A$$

Demorgan's Theorm

$\overline{A.B} = \overline{A} + \overline{B}$(first Theorm)

$\overline{A+B} = \overline{A}.\overline{B}$(Second Theorm)

Demorgan's FirstTheorm:

- $\overline{A.B}=\overline{A}+\overline{B}$(first Theorm)

A	B	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Demorgan's Theorm

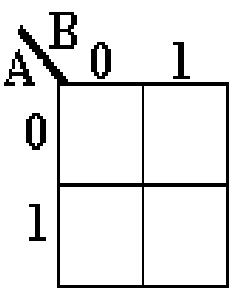
- $\overline{A+B}=\overline{A}.\overline{B}$(first Theorm)

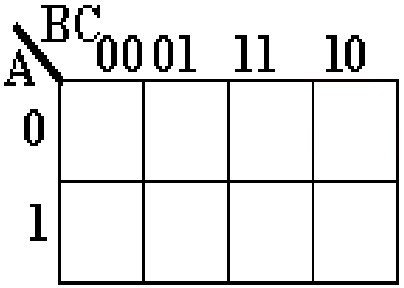
A	B	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

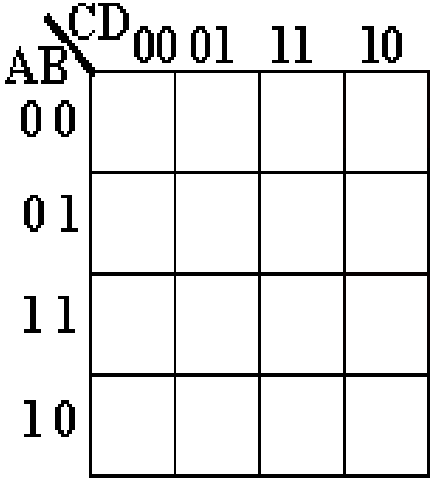
Karnaugh Map

It is a graphical method of logic simplification of logic expressions.

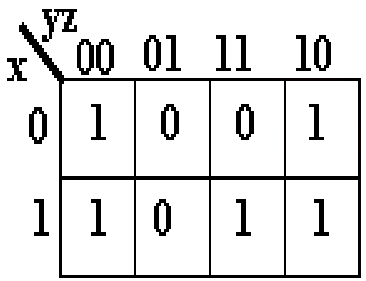
2,3,4 -Variable K-Map







consider the Sum of Product form,
example: f = **x'y'z' + x'yz' + xy'z' + xyz' + xyz**



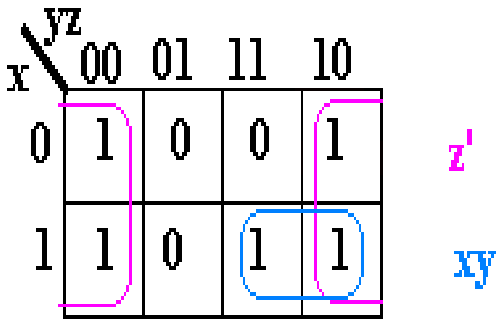
$$f = x'y'z' + x'yz' + xy'z' + xyz' + xyz$$

Designing of k-map

make the largest possible groups of adjacent 1's (remembering that the map wraps on its edges).

Every 1 must be in a group, overlapping groups are ok as are single 1's.
The number of adjacent 1's in any group must be an integeral power of 2, i.e. 1, 2, 4, 8, 16.
Translate each group into a product term by including each variable or it's complement if the variable does not change value over the group

Solution



$$f = x'y'z' + x'yz' + xy'z' + xyz' + xyz$$

$$= xy + z'$$