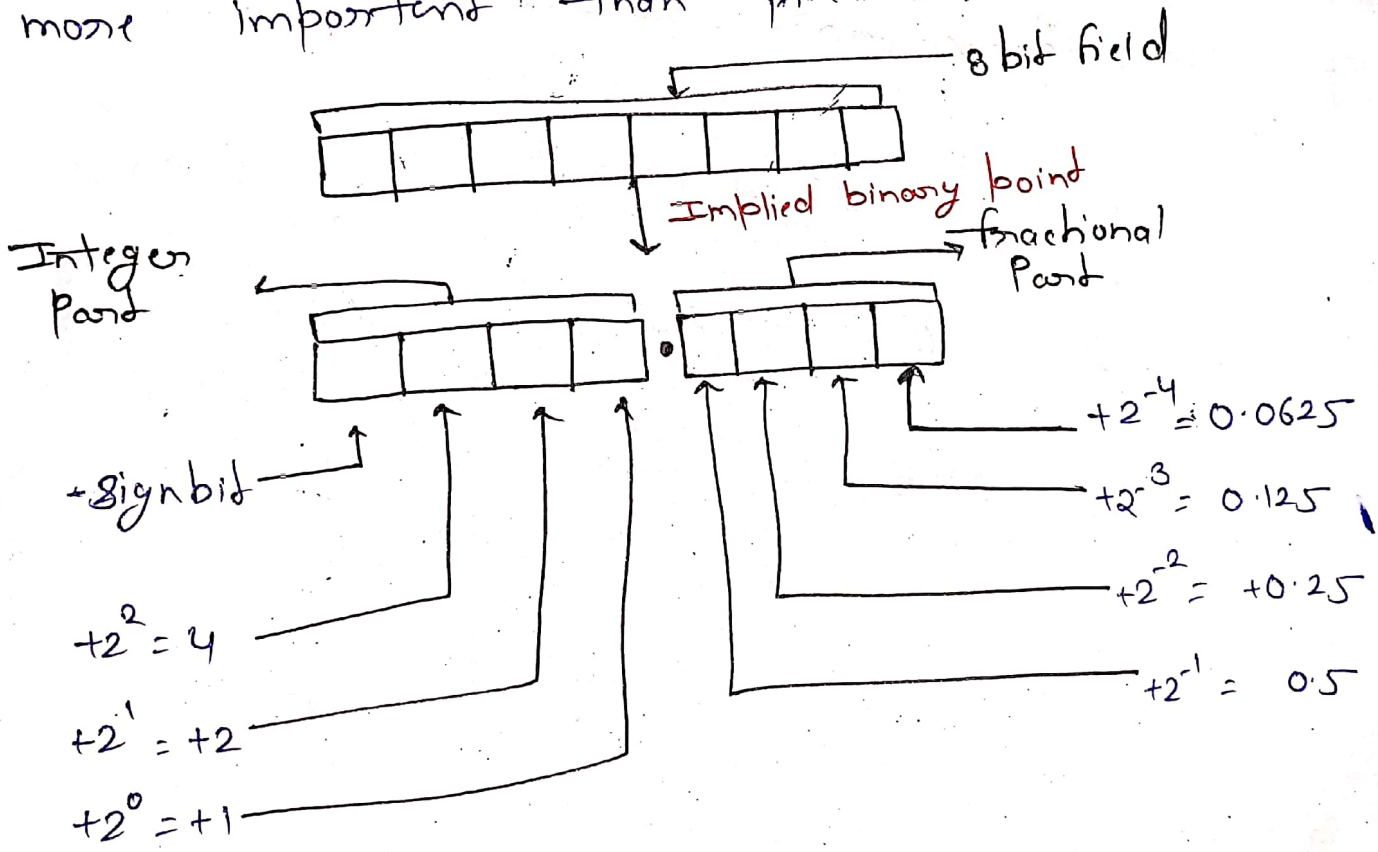
DETAILED LECTURE NOTES

fixed Point Representation

The Real number article describe the fixed point representation of real numbers. The use of the fixed point Representation that is used widely in the signal processing system or in the game applications. where performance is more important than precision.



The shifting point process above is the key to understand fixed point number representation.

To represent a real number in computer, a fixed point is defines by fixing the binary point to be at some position

for representing we need two parameters

→ width of the number representation

→ binary point position within the number.

The representation of the real number

0 0 0 1 0 . 1 1 0

So the representation of the number

$$1 \times 2^{1} + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 2 + 0.5 + 0.25$$

$$= 2.75$$

DETAILED LECTURE NOTES

Codes: It is the symbolic representation of discrete information. It may be represend in the form of number. The symbol used in the binary digit 0 and 1 arranged accordingly.

A code is used to enable an operator to feed data into the computer. The computer convert these data into binary code and after computation transform the data into original formad.

These are classified into five groups

1. Weighted binary code
2. Non weighted binary code
3. Error detecting code
4. Error correcting code
5. Alphanumeric code

1. Weighted Binary Code it obey their positional weighting principal Each position of a number represent a specific weight. In this code, the bits are multiplied by the weight indicated, the sum of these weighted bit give the equalent decimal digit. It is a method to represent binary no to its decimal equalent number.

Decimal	8421	5421	2421
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	1000	1011
6	0110	1001	1100
7	0111	1010	1101
8	1000	1011	1110
9	1001	1100	1111

BCD or 8421 Code

It use the binary number system to specify the decimal number 0 to 9. It has four bits. The weight are assigned according to the position occupied by these digits.

2421 Code

A decimal number is represented in 4 bit form and total weight of 4 bit = 2+4+2+1 = 9. It represents the decimal number upto 9. It is a self complementing code

Reflective Code A code is said to be reflective when the code 9 is the complement of 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4. The 2421, 5211 and Excess 3 codes one reflective codes.

DETAILED LECTURE NOTES

2. Non weighted code These are not positionaly weighted it is not assigned with a fixed value.

Excess-3 code →
It represent a decimal number in binary form as a number greater than 3. It is obtained by adding 3 to the decimal number.

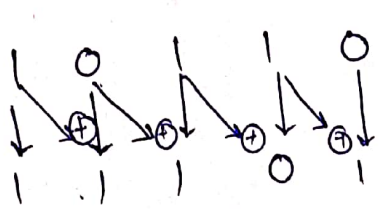
Example
Decimal No. 6 4 3
Add 3 to each bit +3 +3 +3
Sum → 1 9 7 6

Gray Code → It belongs to a class of codes called minimum change code, in which only one bit in the code group change when moving from one to next bit. It is not suitable for airthmetic operation but it used in analog to digital converter.

Conversion of binary number to gray code
→ first bit of the MSB of the gray code is same as the first bit of binary number.
→ Second bit of the gray code is equals the x-or of the first and second bit of binary no.
→ third gray code is equal the x-or of the second and third bit

Example

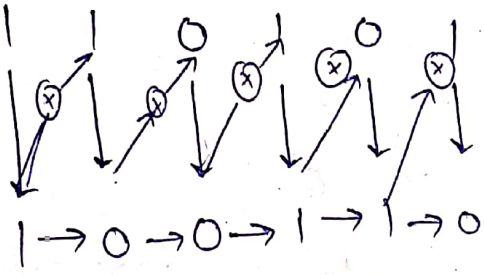
$[10110]_2$ → Gray code



Conversion of gray to binary code

→ the binary bit is same as the first Gray code bit
→ if the second gray bit is 0, the second binary bit is the same as first binary, if the second gray bit is 1, the second binary bit is the inverse of the first binary bit

Example



$[1\,1\,0\,1\,0\,1]_G \rightarrow [1\,0\,0\,1\,1\,0]_B$

3. Error Detecting Code for detecting the error

the error detecting code works. It is used for parity check, in which an extra parity bit is included with the binary msg to make the total no of 1's is either odd or even. The process performed by two types

→ Even Parity method the total no of 1 in the code group must be an even no including the parity bit
→ Odd parity method the total no of 1 must be an odd number, including the parity bit

It can check only the single error not double error. It is used for detecting error in transmission.

4. Error Correcting code

Hamming code R. W. hamming developed a system provide a mathematical way one or more parity bit to a data character in order to detect and correct error. The Hamming distance blw two codes is defined as the number of bit changed from one code word to another.

Example

$C_i = 1\ 0\ 0\ 0\ 1\ 1\ 1$

$C_j = 0\ 0\ 0\ 1\ 0\ 1\ 1$

So the hamming distance

$$d_{ij} = 3$$

5. Alphanumeric codes

It must be capable of handling non-numerical information. The codes that represent numbers, alphabetic letter and special symbol are called alphanumeric code.

It contain set of neressary data includes like (i) 26 lower case letter (ii) 26 upper case letters

(iii) 10 numeric digit
(iv) 25 special character

ASCII Code It is called American standards code for information interchange, used in microcomputers by manufacturers. It represent a character with 7 bits, which can be use as one byte with one bit undefined.

EBCDIC Code It is extended Binary loded Deimal Information code used code grouping different from ASCII code. It use eight bit for each character and ninth bit for parity.

Hollerith code It is used in the punched card It consist of 80 column and 12 Row. Each column represents an alphanumeric character of 12 bits by punching ware in the particulen Rows.

BCD Codes Example

PAGE NO. .....

BCD Addition The rule for addition of two BCD number is given below.

(1) Add the two number using rules for binary addition

(2) If a four bit sum is equal to or less than 9, than it is a valid BCD number

(3) If a four bit sum is greater than 9 or a carry out of the group is generated, it is the invalid result. Add 6 to the four bit sum in order to skip the six invalid state and return the code to the BCD.

Example

1. Add the following BCD no.

(a) 1001 and 0100

```
  1001
  0100
  ----
  1101   → invalid BCD no      9
  0110   → Add 6              +4
  ----
0001 0011 → Valid BCD no     (13)10
  1    3
```

(b) 00011001 and 00010100

19
+14
$(33)_{10}$

```
 00011001
 00010100
 ---------
 00101101  → Right Group is invalid
     0110  → Add 6
 ---------
 00110011  → Valid BCD number
 └──┘└──┘
  ω    ω
```

Binary Subtraction

(i) $(546)_{10} - (429)_{10}$

```
  546   m        0101 0100 0110
- 429   s        0100 0010 1001   ↓ Subtrahend change into 1's complement
 (117)           1011 1101 0110
```

- Add the 1's complement and minuend

```
         0101   0100   0110
         1011   1101   0110
        ---------------------
Carry ① 0000 ① 0001  ⓪ 1100
  EAC          carry
End around     +1            +1
 carry
        ---------------------
         0001   0001   1101  — Adder 1
```

EAC is added to the first digit and carry generated added to the previous one

As per algorithm

```
   0001   0001    1101
   0000   0000    1010
   ---------------------
   0001   0001  ① 0111    Adder 2
                 → discard the carry in adder 2
   ---------------------
   (1 17)₁₀
```

$(117)_{10}$

BCD Subtraction Algorithm.

1. for BCD subtraction we use two adder
2. Below is the algorithm for Adder 2

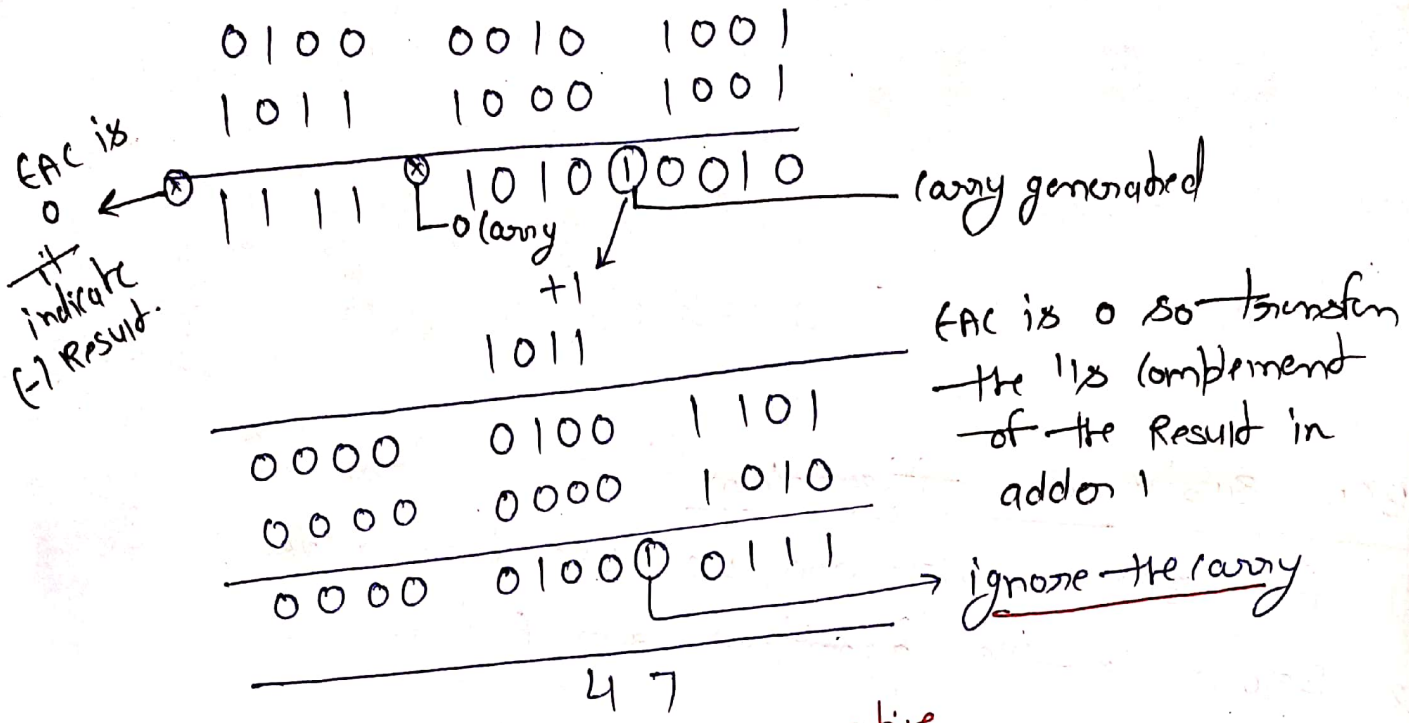
Decade Result.	Sign of total Result. (+) EAC = 1	(-) EAC = 0
$C_n = 1$	Transfer true Result of adder 1 0000 added in adder 2	Transfer 1's complement of Result of adder 1 1010 added in adder 2
$C_n = 0$	1010 added in adder 2	0000 added in adder 2

Example (i) $(429)_{10} - (476)_{10} = ?$

429
-476
-47

429 → 0100 0010 1001
476 → 0100 0111 0110 ← 1's complement of subtrahend.
1011 1000 1001

→ add with minuend.



(E)AC = 0 shows the Result is negative

Method-2 Another method in BCD subtraction is addition of the 9's complement of the subtrahend to the minuend.

Q. Subtract 748 from 983 using 9's complement method

Solution 9's complement of 748 = $\begin{array}{r} 999 \\ -748 \\ \hline 251 \end{array}$

Direct method

$\begin{array}{r} 983 \\ -748 \\ \hline 235 \end{array}$

EAC $\begin{array}{r} 983 \\ +251 \\ \hline 234 \end{array}$

(1) → 1 Add End Around carry

$\begin{array}{r} 234 \\ \hline 235 \end{array}$

DETAILED LECTURE NOTES

Basic logic Gates

A logic gate is an electronic circuit which makes logical decision. The most common logic gates are OR, AND, NOT, NAND and NOR gates.

Logic gate have 1 or more input and 1 output. expect for the NOT gate.

OR Gate The OR gate performs logical addition, commonly known as OR function. The OR gate has two or more input and one o/p.

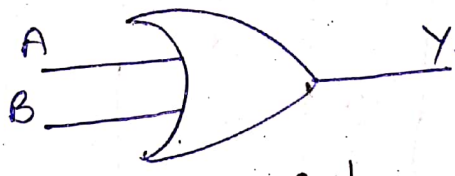
The operation of OR gate is such that a high on the o/p is produced when any one i/p is high.

$$Y = A + B$$

for more than two input

$$Y = A + B + C + D + \cdots$$

logical Symbol:

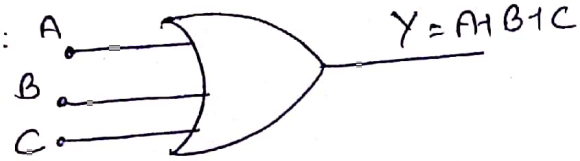


2 i/p OR Gate

TRUTH Table.

Input		output
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

the same idea can be extended for two or more input

for Eg: 

Truth Table:

Input			output
A	B	C	Y = A+B+C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

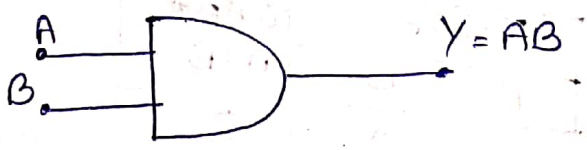
2. AND Gate It performs logical multiplication. Commonly known as AND function. It has two or more input and one output.

$$Y = A \cdot B$$

TRUTH Table:

Input		output
A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

Logical Symbol:



It can also be used as 3 ilp AND gate. The output is high if all the input is high

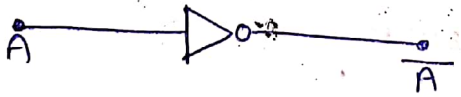
Inputs			output
A	B	C	Y=A·B·C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3. NOT Gate (Inverter)

NOT gate performs the basic logical information called inversion or complementation. The purpose of this gate is to convert one logic level into the opposite

A represents the ilp and Y represent the o/p $Y = \overline{A}$. When the input is high the output is low. and vice versa.

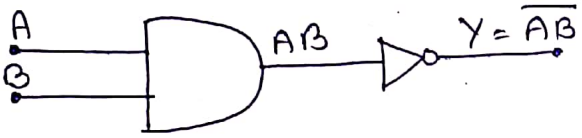
Logic Symbol



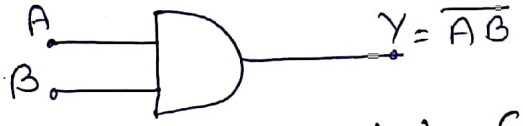
Truth table:

Input A	output $Y = \overline{A}$
0	1
1	0

4. NAND Gate Nand is the combination of NOT-AND gate. It has two or more input and only one output. $Y = \overline{A \cdot B}$. When all i/p are high the o/p is low. If one or both the input are low, the o/p is high.



≡



logic Symbol of NAND Gate

TRUTH Table:

Input A	Input B	output $Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

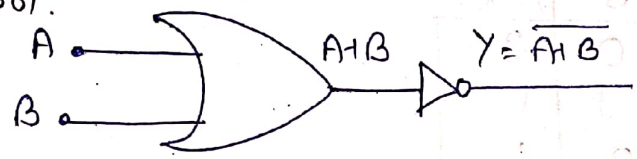
5. NOR Gate NOR is the combination of NOT-OR gate. It has two or more input and one output.

$$Y = \overline{A + B}$$

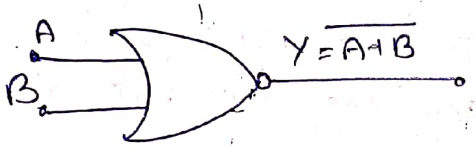
DETAILED LECTURE NOTES

The output is high only when all the inputs are low. If any one or both the input are high, the o/p is low.

Logic Symbol:



≡



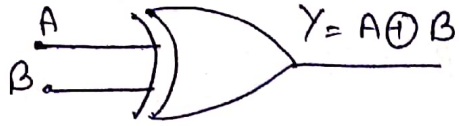
Truth Table

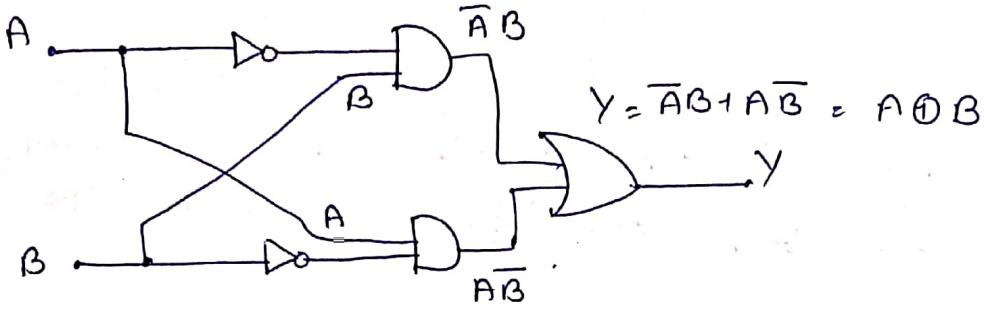
Input A	Input B	output Y = $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

6. Exclusive-OR (Ex-OR) Gate

It is a gate with two or more input and one o/p. The o/p is high if and only one input Assume in the high state. It is to be saying that output is high if either input A or input B is high exclusively. and low when both input are 1 and 0 simultaneously.

Logic Symbol:





TRUTH Table

Inputs A	B	output Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The truth table of X-OR gate shows the output is high when any one but not all, of the input is at 1.

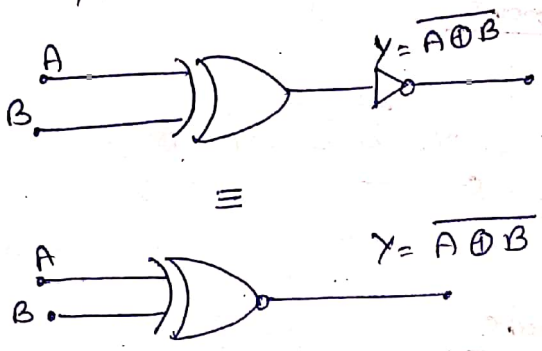
$$Y = \bar{A}B + A\bar{B} = A \oplus B$$

7 Exclusive-NOR (Ex-NOR) Gate

The Ex-NOR gate, is an Ex-OR gate followed by an inverter. An exclusive NOR gate has two or more input and one output. The o/p of two input ex-NOR gate assume high if both the input having the same logic state and its output is low when its input assume diffrend logic state or have an odd no of 1's.

$Y = \overline{A \oplus B}$

Logic Symbol:



Truth Table:

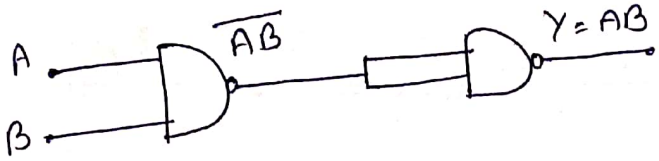
Input A	Input B	output $Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

The important property of Ex-NOR that it can be used for Bit comparison as a bit comparator circuit.

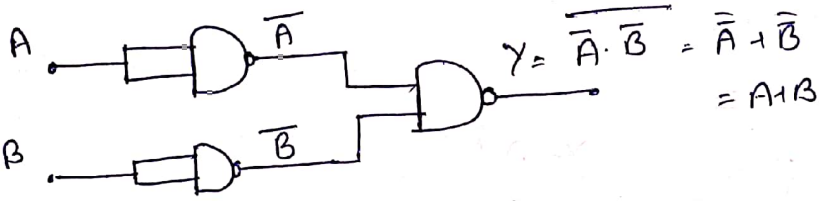
The property of Ex-NOR gate is that it can be used for bit comparison. The o/p of gate is 1 if both the input one similar. It can also be used as an even parity cheker. The o/p of the x-NOR gate is 1 if the number of 1 in its input is even; if the number of 1 is odd the output is 0.

Realization of other logic gate using NAND gate

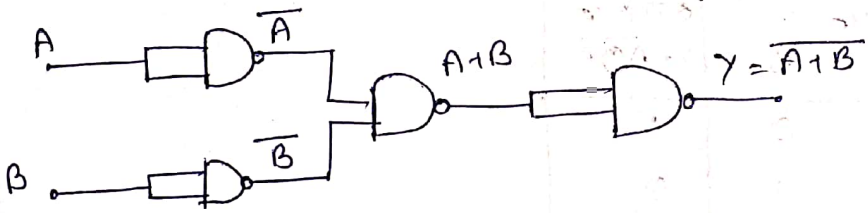
NOT gate



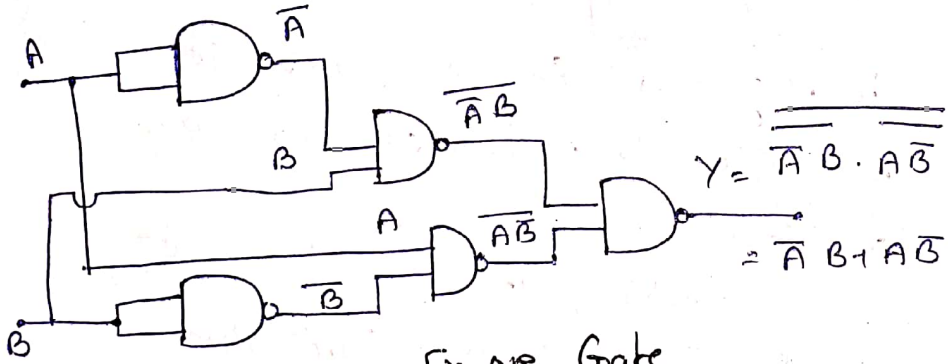
AND Gate



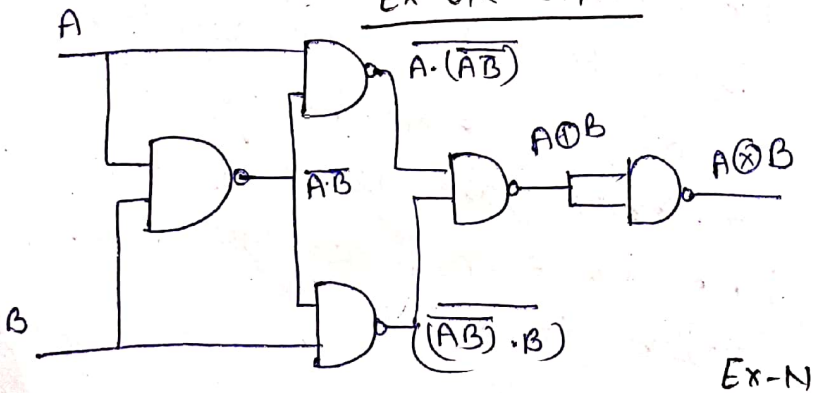
OR Gate



NOR Gate



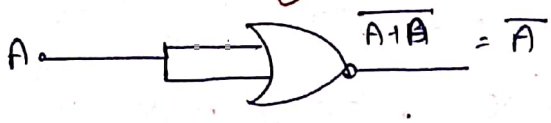
Ex-OR Gate



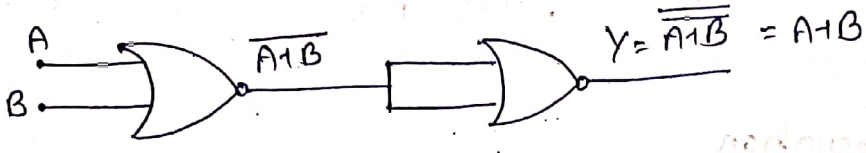
Ex-NOR Gate

DETAILED LECTURE NOTES

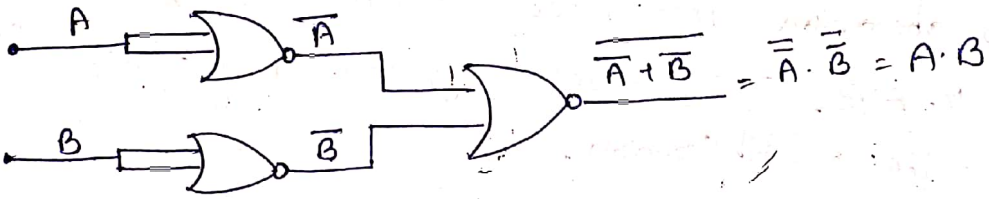
Realisation of other logic Gate using NOR Gate



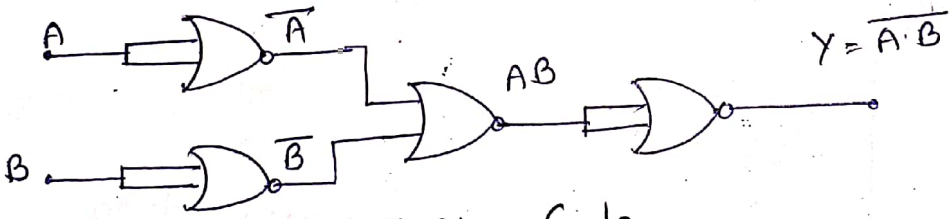
NOT Gate



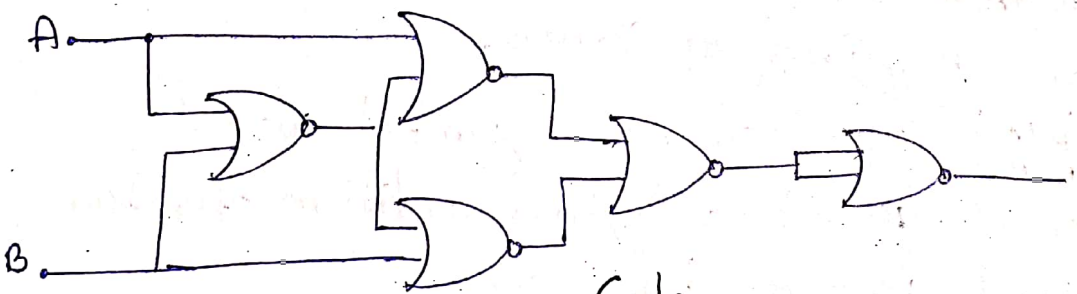
OR Gate



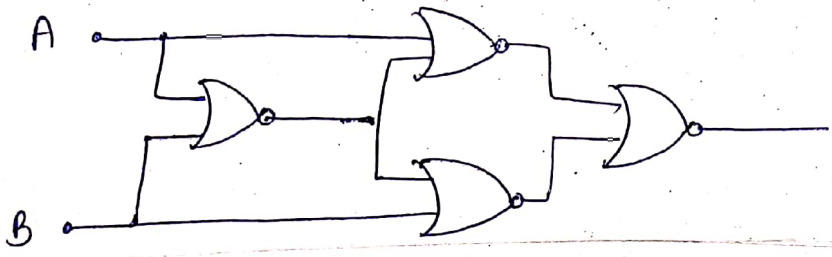
AND Gate



NAND Gate



Exclusive OR X-OR Gate



X-NOR Gate

Boolean Algebra

It can be used for simplify the design of logic gates. The switching function can be explained with Boolean equation, truth table and logical diagram.

Boolean expression can be defined by

→ A constant is a boolean expression

→ A variable is a Boolean expression

A is the variable, combination of variable such as $\bar{A}B + A\bar{B}+C$, but A-B is not a boolean expression

Boolean logic Operation

1. Logical AND operation. for two boolean variable A, B, given as $Y = A \cdot B$. The common symbol for this operation is the multiplication.

Input A	Input B	Output $Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

2. Logical OR operation logical operation between two variable A and B given as $Y = A + B$.

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

The common symbol used for this logical addition operation is the plus sign.

DETAILED LECTURE NOTES

3. Logical Complementation The logical inverse operation convert the logic 1 to logic 0. The method is called the NOT operation. the complement of A is represented by $\bar{A}$.

Basic laws of Boolean Algebra

Logic expression can be expressed and minimized by using the rules, law and thorem of boolean algebra.

1. Boolean Addition: The method involve variable having value of either binary 1 or / or 0. It given as belwoow,

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=1$

2. Boolean multiplication: The method is pen as follows

$0\cdot0=0$
$0\cdot1=0$
$1\cdot0=0$
$1\cdot1=1$

3. Properties of Boolean Algebra

The boolean algebra use diffrent property for minimising the logic expression

Commutative Property $\longrightarrow$ $A+B=B+A$
$A\cdot B=B\cdot A$

Associative Property $A+(B+C)=(A+B)+C$
$A\cdot(B\cdot C)=(A\cdot B)\cdot C$

Distributive Property (i) $A + BC = (A+B)(A+C)$

Proof:

$A + BC = A \cdot 1 + BC$ $(\because A \cdot 1 = A)$

$= A(1+B) + BC$ $(\because 1+B = 1)$

$= A \cdot 1 + AB + BC$ $(\because A(B+C) = AB + BC)$

$= A \cdot (1+C) + AB + BC$ $(\because 1+C = 1)$

$= A \cdot 1 + A \cdot C + AB + BC$

$= A \cdot A + AC + AB + BC$ $(\because A \cdot A = A)$

$= A(A+C) + B(A+C)$

$= (A+B)(A+C)$

(ii) $A \cdot (B+C) = A \cdot B + A \cdot C$

Absorption laws (i) $A + AB = A$

Proof

$A + AB = A \cdot 1 + AB$

$= A(1+B)$

$= A \cdot 1 = A$

$\boxed{A + AB = A}$

(i) $A \cdot (A+B) = A$

$A \cdot (A+B) = A \cdot A + A \cdot B$

$= A + AB$

$= A(1+B)$

$= A$

$\boxed{A \cdot (A+B) = A}$

(iii)

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

$(A + \bar{A})(A+B)$ $[\because A + BC = (A+B)(A+C)]$

$= 1 \cdot (A+B)$

(iv) $\boxed{A \cdot (\bar{A} + B) = AB}$

$= A \cdot \bar{A} + AB$

$= AB$

DETAILED LECTURE NOTES

Consensus laws

(i) $\boxed{AB + \bar{A}C + BC = AB + \bar{A}C}$

Proof $AB + \bar{A}C + BC = AB + \bar{A}C + BC \cdot 1$

$= AB + \bar{A}C + BC(A + \bar{A}) \quad (\because A + \bar{A} = 1)$

$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB(1 + C) + \bar{A}C(1 + B) \quad (1 + B = 1 = 1 + C)$

$= AB + \bar{A}C$

(ii) $\boxed{(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)}$

$(A + B)(\bar{A} + C)(B + C + 0)$

$= (A + B)(\bar{A} + C)(B + C + A\bar{A})$

$= (A + B)(\bar{A} + C)(B + C + A)(B + C + \bar{A})$

$[\because A + BC = (A + B)(A + C)]$

$= (A + B)(A + B + C)(\bar{A} + C)(\bar{A} + B + B)$

$= (A + B)(\bar{A} + C) \quad [\because A(A + B) = A]$

other laws of boolean Algebra

Boolean laws

(i) (a) $A + 0 = A$ (b) $A \cdot 1 = A$

(ii) (a) $A + 1 = 1$ (b) $A \cdot 0 = 0$

(iii) (a) $A + A = A$ (b) $A \cdot A = A$ Idempotency

(iv) (a) $A + \overline{A} = 1$ (b) $A \cdot \overline{A} = 0$ full set, null set

$\overline{\overline{A}} = A$ Double Inversion or involution