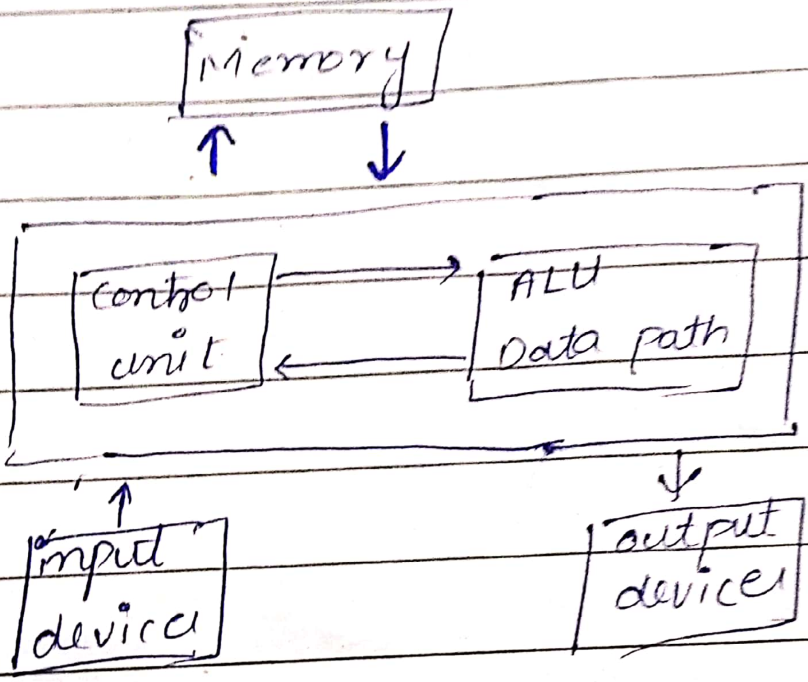
Block diagram of digital computer:



→ Arrows shows flow of data.

→ Different components of computer system are shown in diagram.

→ Memory save data and programs.

* What are different input devices?

* Which are different output devices

Data path:

Arithemetic and logic

control unit : controls everything

e.g.:

flow of program.

Running the system.

Running program fetching code line one by one.

Sample program:

```
integer a, b, c
input a;
input b;
c = a + b
print c;
```

→ if you are a control units-

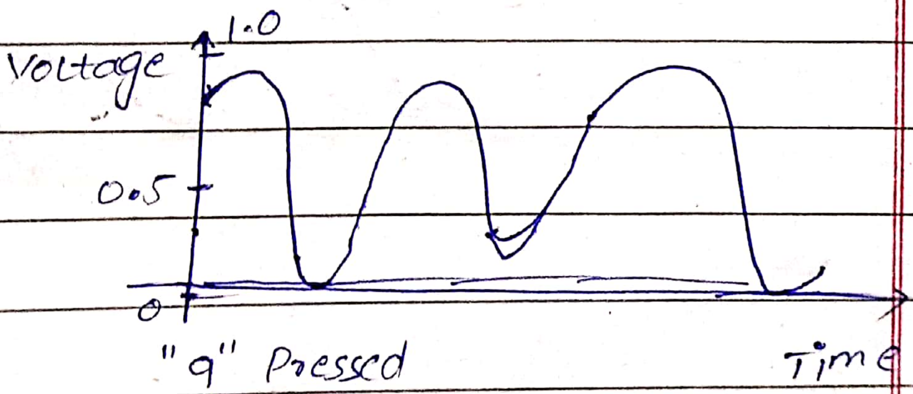
- How control unit running the sample program.
- Data is travelling from one to other component

- Data coming from keyboard.
- Addition in data path.
- storing in memory.
- Display on screen LCD.

How is data travelling?

Data travels in the form of signals.

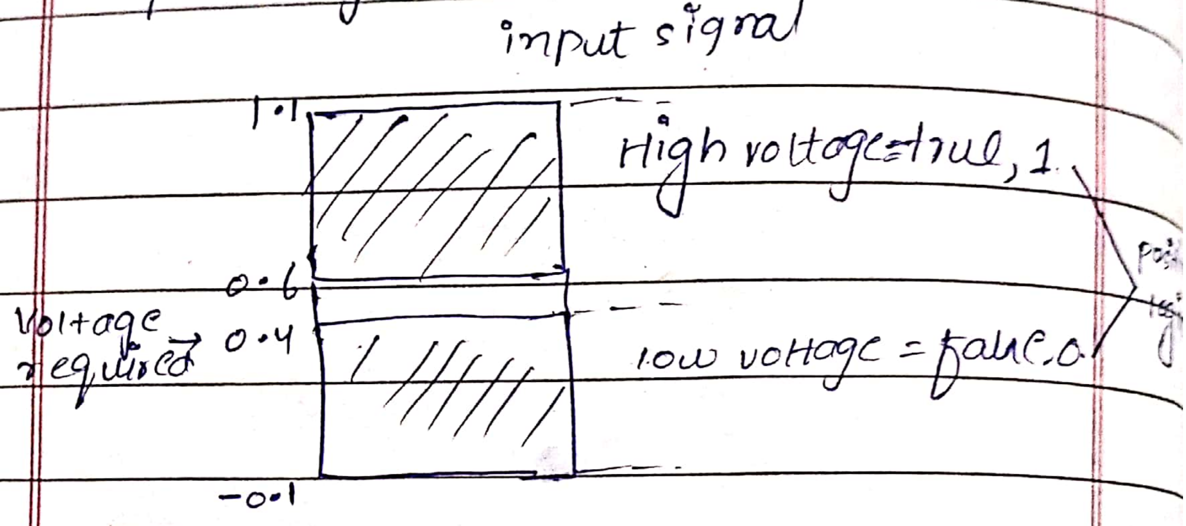
Device 1 $\xrightarrow[\text{Electrical voltage}]{\text{signal}}$ Device 2



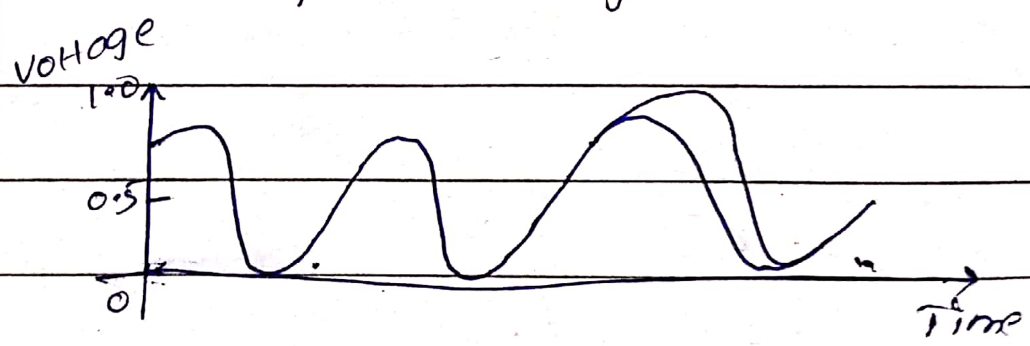
Keyboard → signal → CPU

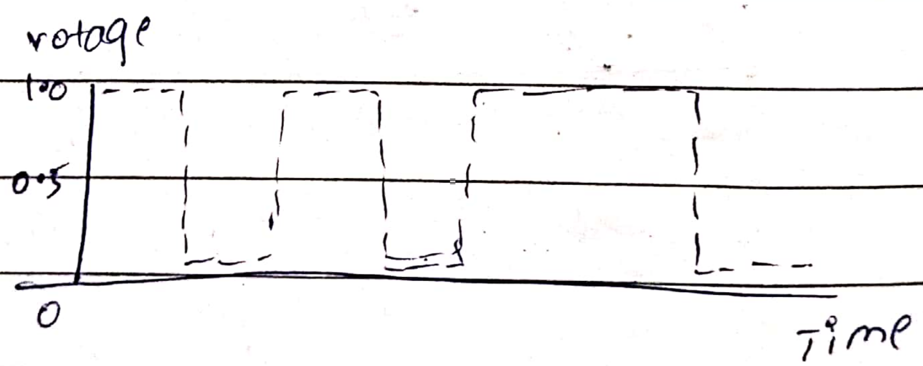
How will we save signal as "9"?

For this we define a range of voltage.



Now voltage has been converted to 0/1 binary number.





if dotted line indicate start of a binary number which

Binary number have we received in above signal ?

Answer : ? 1010110

This is how data is sent/received from/into a device.

This 0/1 is called binary digit or bit.

↳ Binary to decimal conversion :

$(11010)_2 = (\ ?\ )_{10}$

$1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0$

$16+8+0+2+0$

$= (26)_{10}$

As it is -ve power

$(11010.11)_2 = (\ ?\ )_{10}$

$$26 + 1\times2^{-1} + 1\times2^{-2}$$

$$26 + 0.5 + 0.25$$

$$(26.75)_{10}$$

Decimal to binary ?

$(625)_{10} = (?)_2$

2	625		
2	312	—	1
2	156	—	0
2	78	—	0
2	39	—	0
2	19	—	1
2	9	—	1
2	4	—	1
2	2	—	0
	1	—	0

$(100111\ 0001)_2$

M T W T F S   H/W☐-C/W☐   DATE: ___/___/20

→ fraction form:

as it is ← integer   fraction

$(625.6875)_{10} = (?)$

Multiply fraction part by 2:

$.6875 \times 2 = 1.3750$ 1

$.3750 \times 2 = 0.7500$ 0

$.7500 \times 2 = 1.5000$ 1

$.5000 \times 2 = 1.000$ 1

$(0.6875)_{10} = (0.1011)_2$

→ separate both parts

→ convert integer part as it is.

→ convert fraction part as above.

Repeat process until fraction becomes 0.0000 OR sufficient accuracy obtained

$(0.6875)_{10} = (0.1011)_2$

$(625.6875)_{10} = (1001110001.1011)_2$

Types:

We have 4 types of number systems.

1. Decimal (Base - 10)
2. Binary (Base - 2)
3. Octal (Base - 8)
4. Hexadecimal (Base - 16)

We have seen decimal number system representation:

We can represent

Most significant $(724 \cdot 542)_{10}$ least significant

$(7 \times 10^{2} + 2 \times 10^{1} + 4 \times 10^{0} + 5 \times 10^{-1} + 4 \times 10^{-2} + 2 \times 10^{-3}$

9871 9870

Accuracy change (8)871

Digit in base-10 number system are

0, 1, 2, 3, ----, 9

i-e

Digit in base-R

0, 1, 2, ----, r-1

where R is base or radix.

why A, B, C include in hexadecimal?

Ans: To differentiate it from decimal

$(15)_{16} = (?)_{10}$

$1 \times 16^1 + 5 \times 16^0 = ---$

↳ when 1 and 5 are different numb

F

$15 \times 16^0 = ??$

when ~~decimal~~ ~~it~~ is.. F

Scanned with CamScanner

Lecture 2

Data Ranges:

Let me represent 1 bit with one box

$\square$

→ Minimum value it can have $\boxed{0}$

→ Maximum value it can have $\boxed{1}$

Range 0 → 1

i.e 8 digit number in decimal
x x x x x x x x
0 0 0 0 0 0 0 0
9 9 9 9 9 9 9 9

if we have 2 bits

$\square\square$

Min value it can have $\boxed{0}\boxed{0}$

Max value it can have $\boxed{1}\boxed{1}$

All possible value of 2 bits

Range of value 0 → 3

00	0
01	1
10	2
11	3

formula

$0 \rightarrow 2^n - 1$

128 64 16 8 4 2 1

for n = 6 0 → 63

M T W T F S H/W☐-C/W☐ DATE: ___/___/20

Range of fraction:

$$0 \rightarrow \frac{2^n-1}{2^n}$$

Conventions:

Normally we say $10^3 \rightarrow$ kilo

$4kg = 4 \times 10^3 g$
$= 4 \times 1000g$
$= 4000g$

but in dealing with data
$2^{10} = 1024$

2^{10} = K (kilo) | 1kb = 1024byte
2^{20} = Mega (M) | 16M = 16×2^{20} byte
2^{30} = Giga (G)
2^{40} = Tera (T)

Arithemetic operations:

Addition: Decimal addition revision

if Ans >= Base
↓
to carry

765
494
1259

(10+5
10+2

DATE: __/__/20

Binary number:

Ex 1

```
  01100
  10001
  -----
  11101
```

Ex 2:

```
   1 1
  10110
  10111
  ------
 101101
```

2+1

10

Hexadecimal addition:

```
   59F
 + E46
 -----
  1BE5
```

16 | 19
1 – 3

separate 1 and 3

16 | 21
1 – 5

75
46
5

14–9
5

A 10
B 11
C 12
D 13
E 14
F 15

Binary subtraction:

```
   10110
 - 10010
 -------
   00100
```

7051
1923

M T W T F S H/W☐-C/W☐ DATE: __/__/20

Ex 2:

10110 → 22 decimal
− 10011 → 19
00011 → 3

if upper value is smaller than lower than replace values and write "−" minus with it

10011 → 19
− 11110 → 30

10011 < 11110

10110								
10111		5	9	F				
10110 1		F	4	6				
		13	E	5				

19+5

19

15+6=21

16+5

16+3

3E5

octal Multiplication:

$$\begin{array}{r} 762 \\ \times 45 \\ \hline 4672 \\ 3710\times \\ \hline 43772 \end{array}$$

↳ Binary to octal / Hexadecimal
Conversion:

Consider following binary.

(10 1100 0110 1011 1111 0000 01)

↳ 25 digit

Binary → Base n - conversion

For base $= 2^n$, we need minimum n bits to represent any digit of that base

Base $8 = 2^3$

This method is valid for 2^n only.

For octal we represent numb in 3 bits.

Octal	Binary	Binary
0	000	0
1	001	01
2	010	10
3	011	11
4	100	100
5	101	101
6	110	110
7	111	111.

we cannot represent all octal digits in 1 or 2 bits.

consider a decimal number

0000 789.5 26000 → no change

M T W T F S H/W C/W DATE: ___/___/ 20

$(010 \quad 110 \quad 0010110 \quad 10111111000 \quad 0111)_2$

$$\begin{array}{r} 10110 \\ 10010 \\ \hline 00200 \\ \hline \end{array}$$

$$\begin{array}{r} 10011 \\ 11110 \\ \hline \end{array}$$

H.W:

$(673.12)_8 = (?)_2$

$(3A6.C)_{16} = (?)_2$

$(E59C.04B)_{16} = (?)_8$ ~~11110~~

10011

Binary coding decimal 01000

$$\begin{array}{r} 10110 \\ -10010 \\ \hline 00100 \\ \hline \end{array} \qquad \begin{array}{r} 10110 \\ 10010 \\ \hline 00100 \\ \hline \end{array}$$

Lecture # 3

Binary coded decimal:

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

largest no is 9
So we need 4 bits for it.
This conversion is called
binary coded decimal (BCD)
conversion.

→ Decimal to BCD:

$(396)_{10} = (?)_{BCD}$

every digit is taken individually.

$(0011\ 1001\ 0110)_{BCD}$

$(X)_{BCD} \neq (X)_2$

$(396)_{10} = (?)_2$

$(130)_{10} = (110001100)_2 \neq (0011\ 1001\ 0110)_{BCD}$

$$
\begin{array}{r|l}
2 & 396 \\
\hline
2 & 198 - 0 \\
\hline
2 & 99 - 0 \\
\hline
2 & 49 - 1 \\
\hline
2 & 24 - 1 \\
\hline
2 & 12 - 0 \\
\hline
2 & 6 - 0 \\
\hline
2 & 3 - 0 \\
\hline
 & 1 - 1
\end{array}
$$

BCD to decimal:

$(\underbrace{0001}\,\underbrace{1000}\,\underbrace{0101})_{BCD}$

$(1\,8\,5)_{10}$

Binary Number

- Signed Binary number
  - signed magnitude representation
  - 2's complement representation.
- unsigned binary number

① Unsigned numbers:

⇒ Range

← magnitude only →

0 to $2^n - 1$

example: 4 bit no

$0 \rightarrow 2^4 - 1$

Range: 0000

1111

Bit Extension:

e.g making 6 bits number from original 4 bits no:

e.g 976

make it 6 digit

000976

$(9)_{10} = (1001)_2$

Make 8 bit no:

$(9)_{10} = (00001001)_2 = (1001)_2$

② Signed numbers:

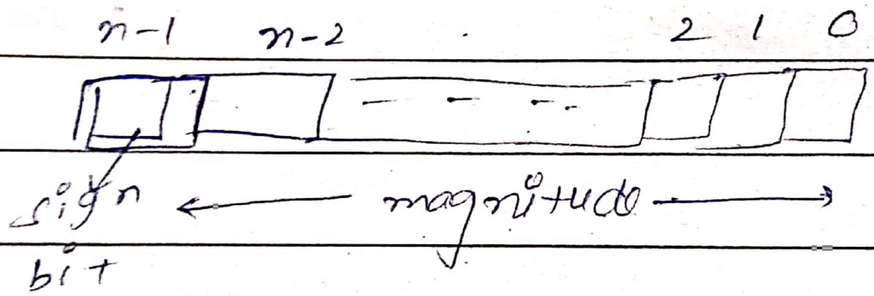
sign Representation

+ → 0

− → 1

we need to represent "+" and "−" sign in binary form 0/1

for n-bit number:



0 → + Total n-1 bits

1 → − to save magnitude

Range:

For example 4 bit no:

- +ve no Range → sign mag
  0 000 +0
  to
  0111 +7
- −ve no Range → 1000 −0
  to
  1111 −7

Bit extension:- 4 bit to 8 bit

$(+9)_{10} = (01001)_2$

$\downarrow$

$(000001001)_2$

$= (+9)_{10}$

$(-9)_{10} = (11001)_2$

$= (100001001)_2$

$= (-9)_{10}$

For n- bit number:-

$-(2^{n-1}-1) \quad +0 \quad +(2^{n-1}-1)$

Note:

The left most bit representa sign.

M T W T F S H/W☐-C/W☐ DATE : ___/___/20

signed decimal	magnitude representation in signed magnitude	
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000
-0	1000	0000
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	11000	1000
	cannot be represented in 4 bits.	

Signed 2's complement representation :

2's complement of N :- (Algorithm)

N = 10100100

1. Start traversing bits from right to left.
2. Leave or copy all data as it is until 1st 1 comes.
3. Copy 1st 1.
4. Flip rest of bits i.e 0101, 1 make 100 make

2's complement of N = 01011100

* positive no's are identical in both representations.

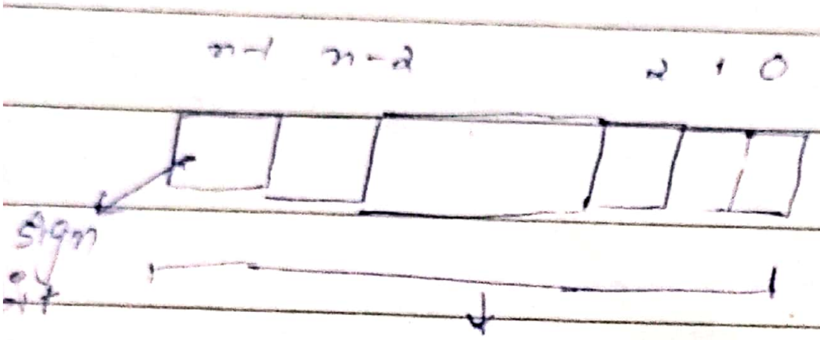
* In both systems left most bit represents sign.

$-X \longrightarrow \boxed{\text{2's complement}} \longrightarrow -(-X) = +X$

$+X \longrightarrow \boxed{\text{2's complement}} \longrightarrow -X$

we this to fill table

signed 2's complement :-



magnitude bits total n bits

available for magnitude

$\boxed{\text{2's complement}}$

~~magnitude~~

Range:

e.g. 4 bit no sig mag

+ve nos Range: $(0000)_{10} \rightarrow (+0)_{10}$

+0

$(0111)_{10} \rightarrow (+7)_{10}$

-ve numb Range:

1000 to 1111

2's compl   2's complement

$-(1000)_2$ $-(0001)_2$

$(-8)_{10}$ $(-1)_{10}$

Range $[-(2^{n-1})$ to $+(2^{n-1}-1)]$

Bit extension:

What is this no in 2's complement representation.

$(0111)_2 = +7$

4 to 8 bit

$(00000111)_2$

Lecture 4

Formula to find 2's complement:

$2^n - N$

N is any binary number

Algorithms of subtraction:

1. Borrows Method
2. Subtraction using 2's complement

$(\pm A) - (+B) = (\pm A) + (-B)$

$(\pm A) - (-B) = (\pm A) + (+B)$

Actual operation performed is addition. Taking 2's complement before that if neccessary.

M T W T F S   H/W ☐ - C/W ☐   DATE : __/__/20

Reason: In our processor we do not have seperate hardware for subtraction. We will use addition hardware.

lesser hardwar

slim proce

Example:

6 - 13 = ?

subtraction using 2's compleme

we need to do

6 + (-13) → Actually additio will be perfomed

step 1:

unsigned representation

6 = 110

13 = 1101

step 2:

signed representation

+6 : 0110

+13 : 01101

Step 3 :

Bit extension

+6 = 00110

+13 = 001101

Step 4 :

Make no of bits same

+6 = 000 110

+13 = 001 101

Step 5 :

A − B = A + (−B)

+6 = 000110

+(−13) = 110011 2's com

111001

We are done here let's check 2's com

−7

M T W T F S   H/W☐-C/W☐   DATE: __/__/20

2nd Example:

$-6-13=?$

we need to do $-6+(-13)$

step 1   unsigned Rep

$6 = 110$

$13 = 1101$

step 2   signed Rep

$+6 = 0110$

$+13 = 01101$

step 3   Bit Extension

$+6 = 00110$

$+13 = 001101$

step 4   Make no of bits same

$+6 = 000\,110$

$+13 = 001\,101$

step 5 $-A-B = -A+(-B)$

* Why we do bit extension?

we have same number of bits -8 available in three numbers A, B, C.

$C = A + B$

For example without bit extension

$-8-8 = -8+(-8)$

discard

$-8 = ①\ 1000$

1000

0000 → wrong ans

↳ with bit extension

$-8 =$ discard ← ① 11000

$-8 = 11000$

①$0000$ → correct ans

$= -16$

* Why we discard last carry?

e.g. $M-N = 13-6 = ?$

M T W T F S H/W☐-C/W☐ DATE: __/__/20

$M = 13 \longrightarrow 1101 \quad 01101$

$N = 6 \quad 0110 \quad 00110$

$M - N \quad 01101$

11010

discard ~~1~~ 00111

2's comp of $N = 2^n - N$

$M - N$

$M +$ 2's comp of N

$= M + (2^n - N$

$= M - N + 2^n$

discard

$000111 + 100000$

$+7$

DATE : ___/___/ 20___

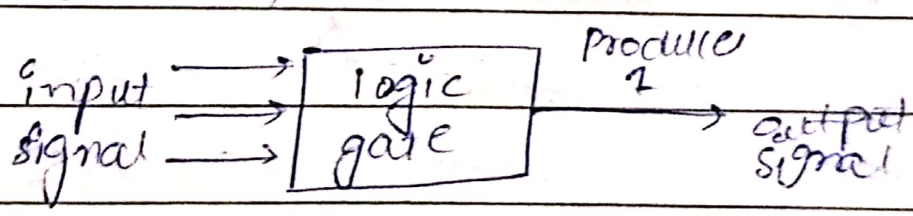
Hardware components that manipulate binary info is called "Digital Circuit."

→ Digital circuits are performing different operations.

→ Basic circuit is called logic Gate ⟶ Base of all circuits

Multiple logic gates make complex circuits.

(one or more)



outputs of gates are applied to inputs of other gates to form digital circuit.

(Electrical signal i.e voltage/current exist througout a digit system in either of two recognizable values

→ Each gate performs a specific logical operation.

Three basic logic operation associated with binary variosble.

- AND
- OR
- NOT

→ AND Gate:

perform AND operation.

operation Representation

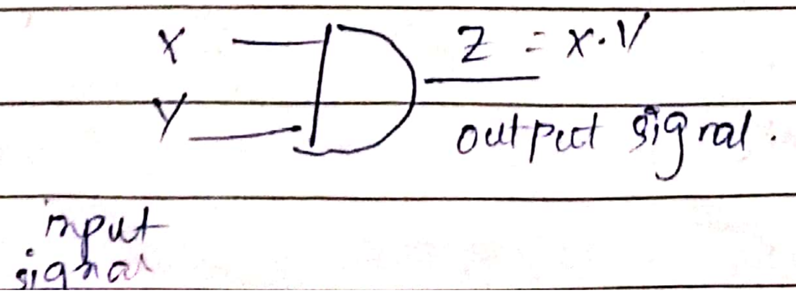
$Z = X \cdot Y$

or

$Z = XY$

Reod as Z is equal to 'X' and 'Y'

Graphical symbol:



Truth Table

input		output
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Truth table: Table showing all possible inputs and their corresponding outputs.

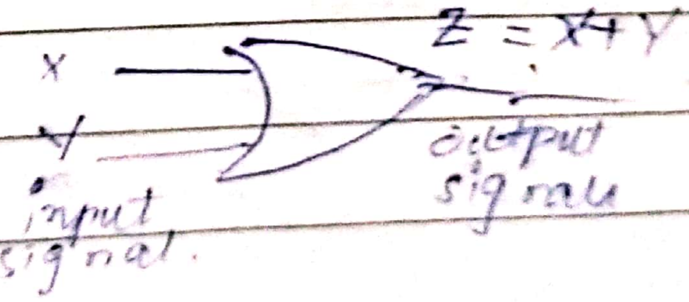
↳ OR Gate:

perform OR operation

$$Z = X + Y$$

Read as Z is equal to X OR Y

graphic symbol:



Truth table:

X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

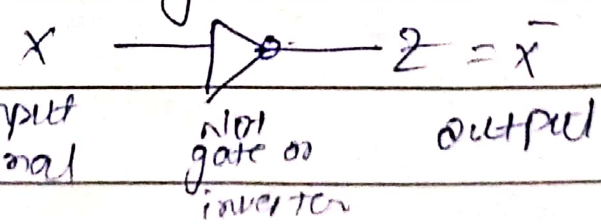
↳ NOT Gate:

- performs complement operation.
- operation: Representation:

$$Z = \bar{X} \text{ or } X'$$

Read as Z is equal to NOT X

- Graphical symbol:

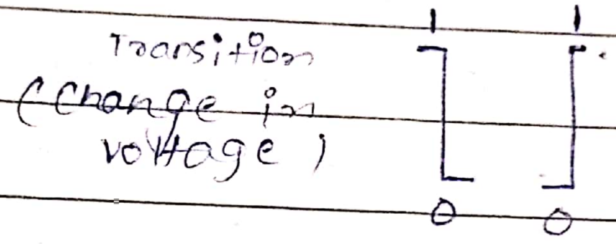


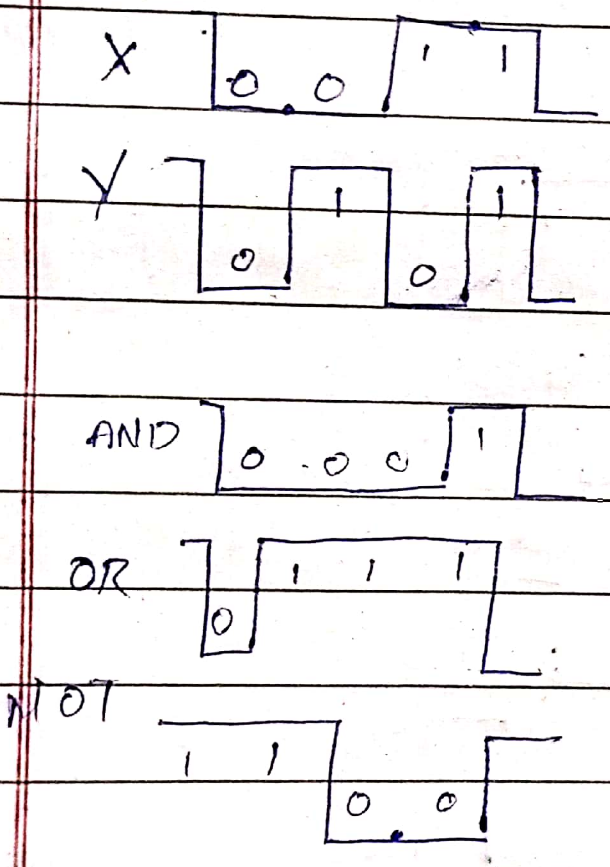
- Truth table:

X	$Z = \bar{X}$
0	1

↳ Timing diagram of AND, OR, NOT:

Timing diagram:



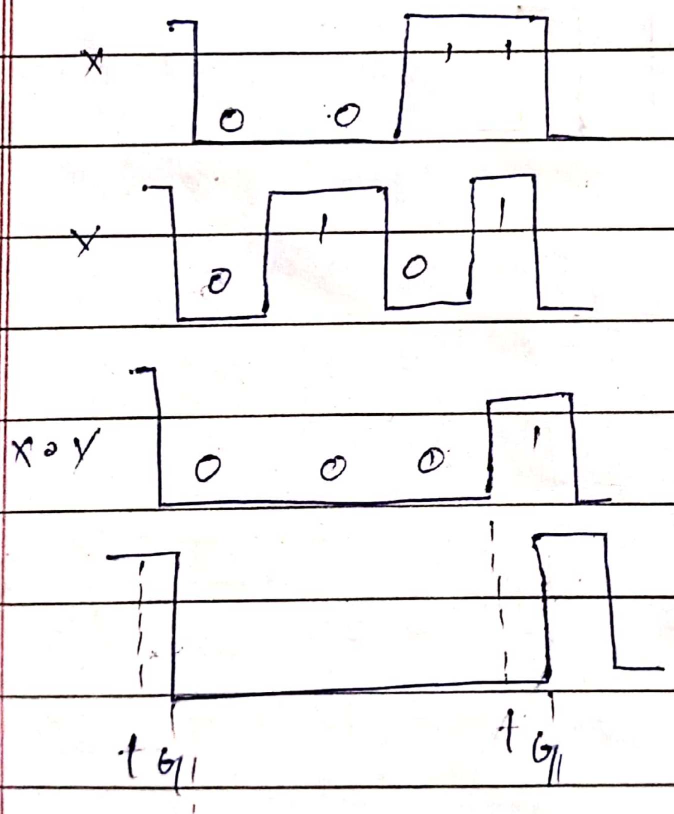


Gate delay:

→ property of a gate.
The length of time it takes for an input change to result in corresponding output change.

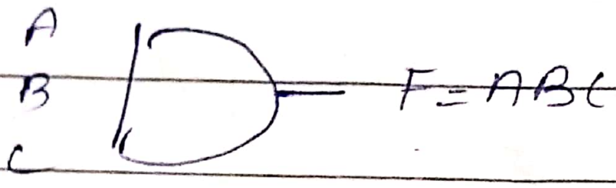
- Denoted by L_G
  depends on (will be different for diff) -
  - * Gate type.
  - * No of inputs
  - * underlying technology.
  - * circuit design of gate.

e.g. :

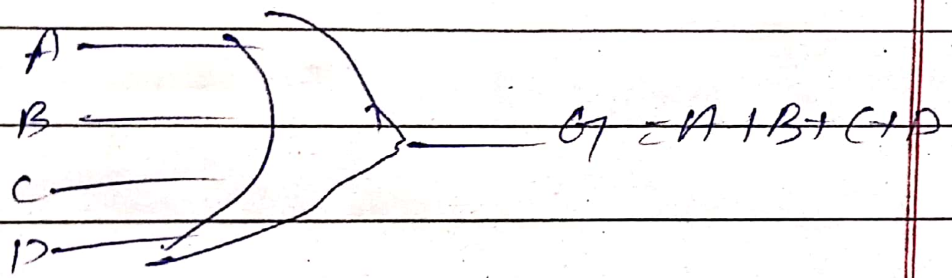


M T W T F S   H/W☐ C/W☐   DATE: ___/___/ 20

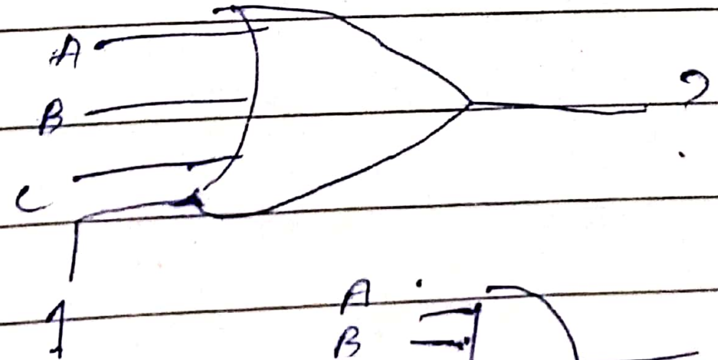
Multiple inputs in gates:

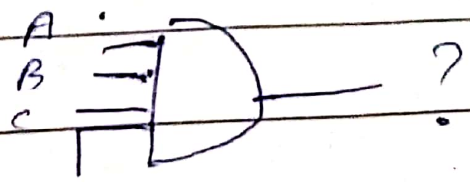


F, will be one (A, B, C) = (1, 1, 1)
and 0 otherwise (for rest of comb)



G will be zero only
when (A, B, C, D) = (0, 0, 0, 0)
for rest of combination
G will be 1





Lecture no 5

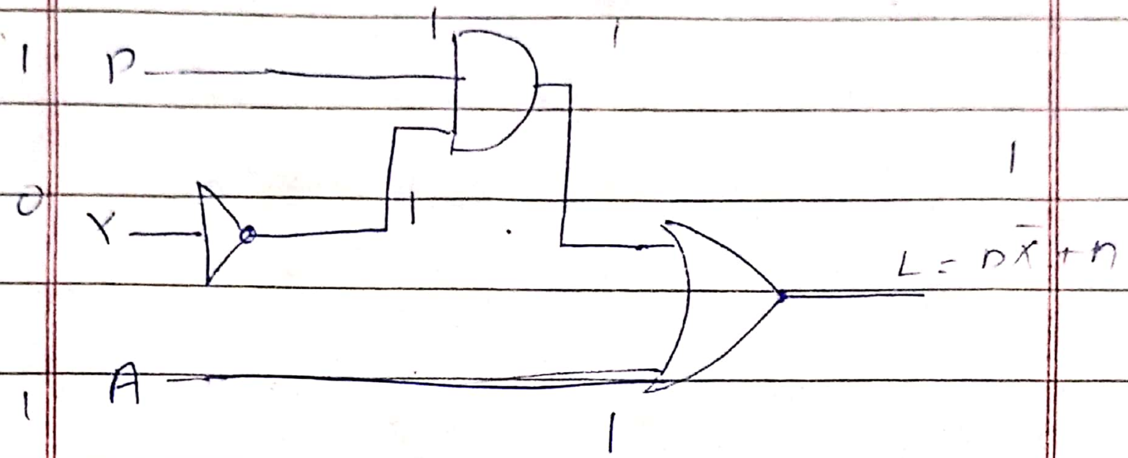
→ for n inputs 2^n Rows in truth table

example:

$L = D\bar{X} + A$

→ First draw circuit.

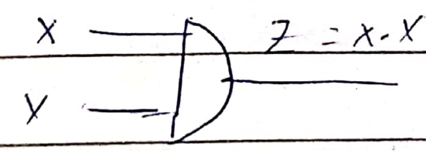
D	X	A	$\bar{X}$	$D\bar{X}$	$L = D\bar{X} + A$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1



variable are contained with logical operations so these circuits are called combinational logic circuits.

→ Draw the circuit of following function.

• Boolean Algebra:



inputs

• Binary variable: X, Y 0/1

• Basic logic operation: AND, OR, NOT

$F = \bar{x}yz + xyz + xz$

M T W T F S H/W☐-C/W☐ DATE: ___/___/20

term → part of expression

$Y = \overbrace{DX}^{\text{term}} + \overbrace{A}^{\text{term}}$ ← boolean expression

→ single variable within term is called literal.

M T W T F S H/W☐-C/W☐ DATE: ___/___/20

→ we use few identities to simplify boolean functions.

Basic identities of Boolean algebra.

1. $x + 0 = x$ 2. $x \cdot 1 = x$

3. $x + 1 = 1$ 4. $x \cdot 0 = 0$

5. $x + x = x$ 6. $x \cdot x = x$

7. $x + \bar{x} = 1$ 8. $x \cdot \bar{x} = 0$

9. $\bar{\bar{x}} = x$

Both of these columns are dual of each other..

Dual of an equation:

AND → OR
OR → AND

① Interchange AND and OR

② Replace 1's by 0's and 0's by 1's.

equation (1, 3, 5, 7) are dual of (2, 4, 6, 8)

→ Duality principle: Boolean equation remains valid if we take the dual of expression on both sides of equal sign. i-e if equation holds true its ~~dual also holds~~ true.

Basic identities of boolean algebra:

* Commutative law:

i-e

10. $X + Y = Y + X$ 11. $XY = YX$

* Associative law:

i-e

12. $X + (Y + Z) = (X + Y) + Z$
13. $X(YZ) = (XY)Z$.

* Distributive law:

14. $X(Y + Z) = XY + XZ$
15. $X + YZ = (X + Y)(X + Z)$

* Demorgan's Law:

16. $\overline{X+Y} = \bar{X} \cdot \bar{Y}$
17. $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

→ proof of distributive law:
start proving from logic expression.

$R.H.S = (X+Y)(X+Z) \Rightarrow X\cdot X + Y\cdot Z + Y\cdot X + Y\cdot Z$

$\because X\cdot X = X$ $\quad X + XZ + XY + YZ \Rightarrow X(1+Z) + XY + YZ$

$\because 1+Z = 1$ $\quad X + XY + YZ \Rightarrow X(1+Y) + YZ \Rightarrow X + YZ$

$= L.H.S$

Demorgan's theorem with n variables-

$$\overline{X_1 + X_2 + \cdots + X_n} = \bar{X}_1 \cdot \bar{X}_2 \cdot \bar{X}_3 \cdots \bar{X}_n$$

$$\overline{X_1 \cdot X_2 \cdots X_n} = \bar{X}_1 + \bar{X}_2 + \bar{X}_3 + \cdots + \bar{X}_n$$

Reduce following function and draw circuit:

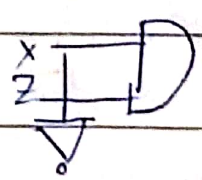
$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$

$= \bar{X}(YZ + Y\bar{Z}) + XZ$

$= \bar{X}Y(Z + \bar{Z}) + XZ$

$= \bar{X}Y(1) + XZ$

$= \bar{X}Y + XZ$



*

HOMEWORK:

Proof

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

→ Consensus theorem:

$XY + \bar{X}\cdot Z + YZ = XY + \bar{X}Z$

Start from longer side

L.H.S

$XY + \bar{X}Z + YZ\cdot 1$ $\quad \because X\cdot 1 = X$

$= XY + \bar{X}Z + YZ(X+\bar{X})$ $\quad \because X+\bar{X} = 1$

$= XY + \bar{X}Z + XYZ + \bar{X}YZ$

$= XY + XYZ + \bar{X}Z + \bar{X}YZ$

$= XY(1+Z) + \bar{X}Z(1+Y)$

$= XY + \bar{X}Z$ $\quad \because 1+Z = 1$

$=$ R.H.S $\quad \because 1+Y = 1$

What is dual of consensus theorem

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

→ complement of a function:

$F_1 = \bar{X} Y \bar{Z} + \bar{X} \bar{Y} Z$

$\bar{F_1} = ?$

$\bar{F_1} = \overline{\bar{X} Y \bar{Z} + \bar{X} \bar{Y} Z}$ By demorgan's law

$= (\overline{\bar{X} Y \bar{Z}}) \cdot (\overline{\bar{X} \cdot \bar{Y} Z})$

$= (\bar{\bar{X}} + \bar{Y} + \bar{\bar{Z}})(\bar{\bar{X}} + \bar{\bar{Y}} + \bar{Z})$

$= (X + \bar{Y} + Z)(X + Y + \bar{Z})$

$F_2 = X(\bar{Y}\bar{Z} + YZ)$ $\bar{F_2} = ?$

$\bar{F_2} =$

→ complementing function using duals:

step 0: Add paranthesis around terms.

Add dot (·) in AND

step 1: Take dual of function

step 2: Complement each literal.

→ [AND → OR
OR → AND
0 → 1
1 → 0

$F_1 = \bar{x} y \bar{z} + \bar{x} \bar{y} z$

$= (\bar{x} \cdot y \cdot \bar{z}) + (\bar{x} \cdot \bar{y} \cdot z)$ step 0

dual $= (\bar{x} + y + \bar{z})(\bar{x} + \bar{y} + z)$ step 1

$\bar{F_1} = (x + \bar{y} + z)(x + y + \bar{z})$

$F_2 = x(\bar{y}\bar{z} + yz)$

- using demorgan's theorem, express the function:

$F = A\bar{B}C + \bar{A}\bar{C} + AB$

(a) with only OR and compliment operations.

(b) with only AND and complement and complement operation.

(a) solution:

There should not be only AND gate in circuit.

$$\bar{\bar{F}} = \overline{A\bar{B}C + \bar{A}\bar{C} + AB}.$$

$$= \overline{(\overline{A\bar{B}C}) \cdot (\overline{\bar{A}\bar{C}}) \cdot (\overline{AB})}$$

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

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

Lecture # 6

↳ Standard forms of Boolean expression:

• A boolean function expressed algebrically can be written in a variety of ways.

2 standard forms are

* sum of products.
* Product of sums.

- standard forms facilitate the simplification procedure for boolean expressions.

Standard forms contain:

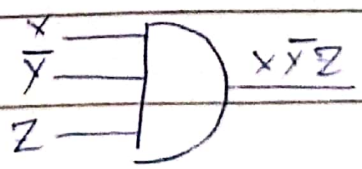
• product term.
• Sum term.

↳ product term:-

"logical product (AND) operation of n literals"

e.g:

$\underbrace{X\bar{Y}Z}_{\text{1 product term}}$



M T W T F S   H/W☐-C/W☐   DATE: ___/___/20

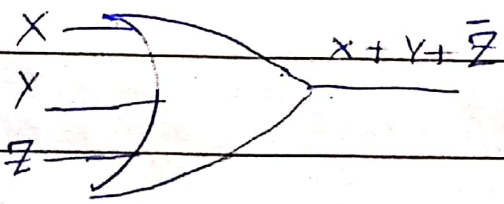
literals can be complemented or uncomplemented.

↳ Sum term:-

"logical sum (OR) operation of n literals"

eg: 1 sum term

$X + Y + \bar{Z}$



Sum of product form:-

output variable ← F = product $term_1$ + product $term_2$ + product $term_3$ + product $term_4$ --- + product $term_n$

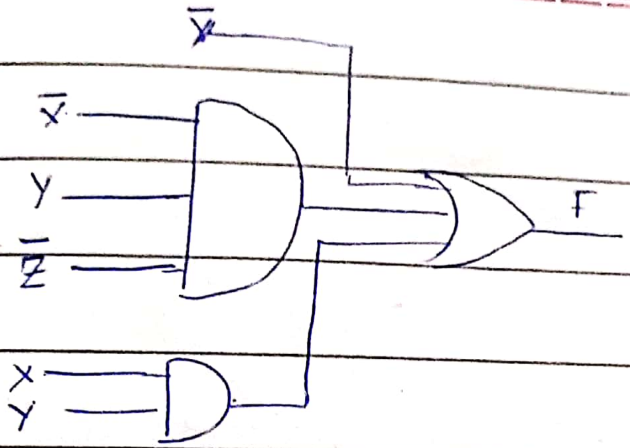
product $term_i$ = product of 1 to n literals.

e.g:

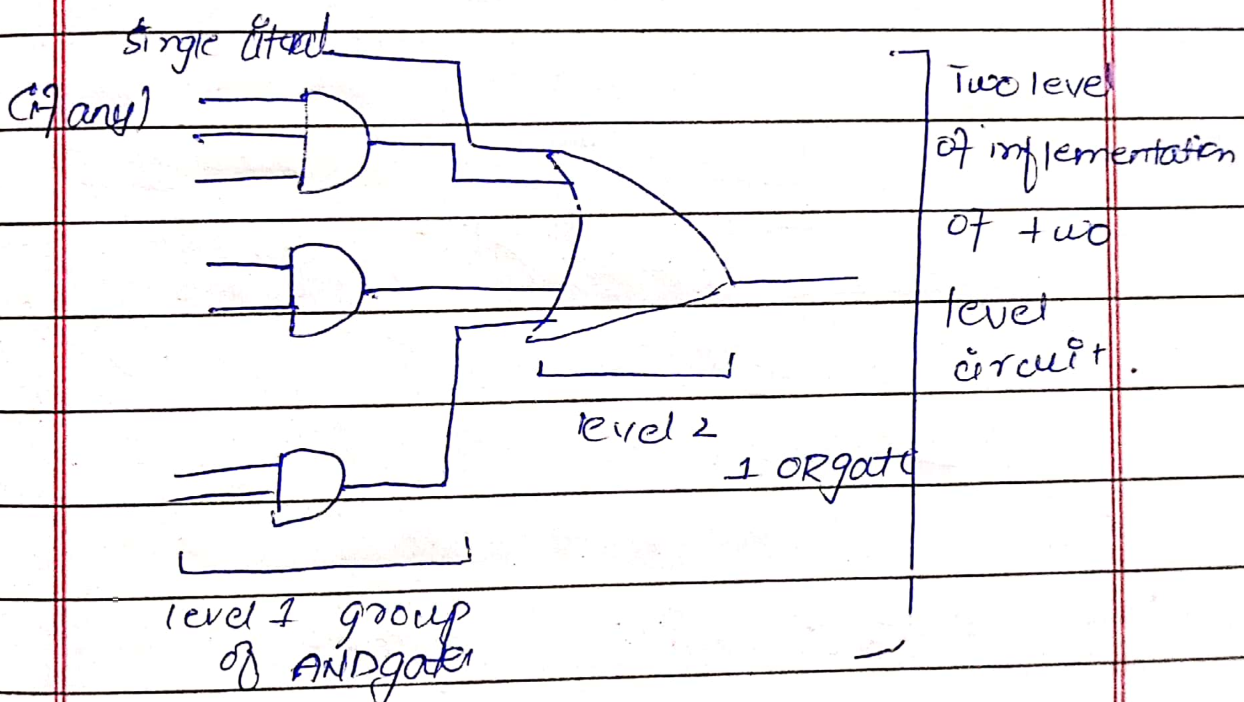
$F = \underbrace{\bar{Y}}_{\text{1 literal}} + \underbrace{\bar{X}Y\bar{Z}}_{\text{3 literals}} + \underbrace{XY}_{\text{2 literals}}$

Total product term = 3

M T W T F S H/W☐-C/W☐ DATE: ___/___/20



Standara form of sum of product circuit.



we assume that input variable are directly available in their complemented or uncomplimented terms. So inverter are not included in diagram.

Is following function in sum of product form??

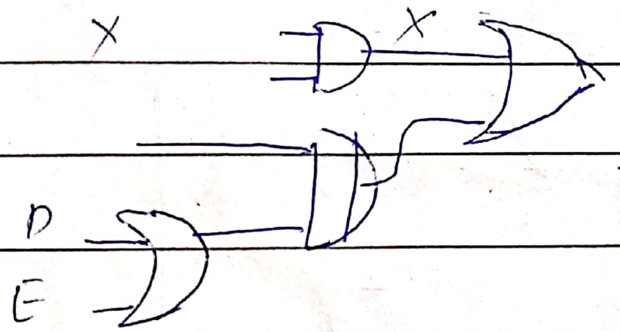
$F = AB' + C(D+E)$

It is not is sum of product form.

It should be a

$F = AB + CD + CE$ ✓

$F = AB + C(D+E)$



Product of sum term:

$F = (\text{sum term}_1) \cdot (\text{sum term}_2) \cdot (\text{sum term}_3) \cdot$
$\text{-----} \cdot (\text{sum term}_n)$

sum term_i = logical sum of 1-to-n literals

For e.g:

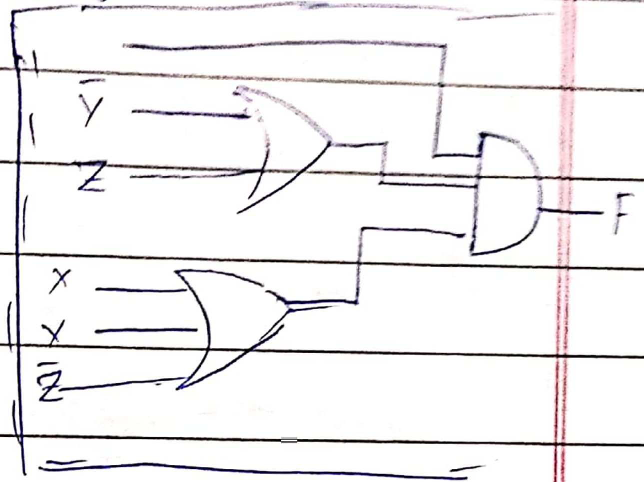
$F = X \cdot (\bar{Y}+Z) \cdot (X+Y+\bar{Z})$ → circuit

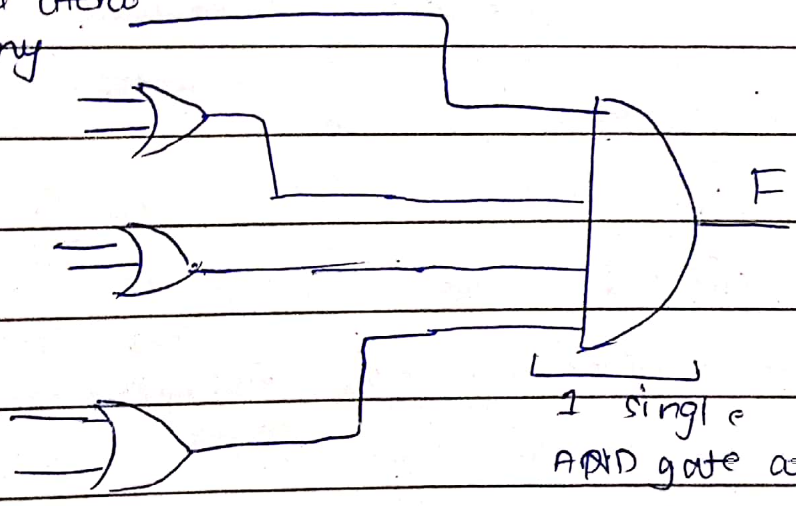
sum term 1 — 1 - literal

sum term 2 — 2 - literals

sum term 3 — 3 literals

Implementation configuration:





Singal literal if any

1 single AND gate at level 2

group of OR gates at level 1

→ single expression results in ⇒ the level of gating structure

Note: if we talk about m_0 it will be 1 other all are 0.

M T W T F S   H/W ☐ C/W ☐   DATE: __/__/20

Minterms and Maxterms:

X, Y, Z   Total variables = 3

Total combination = 2^3 For each binary number there is a related minterm

Basically there are all possible product terms which we can have using 3 variables

X	Y	Z	Product term → these product terms are called minterms	
0	0	0	$\bar{X}\bar{Y}\bar{Z} = 1$ on this combination	m_0
0	0	1	$\bar{X}\bar{Y}Z$ For the corresponding	m_1
0	1	0	$\bar{X}Y\bar{Z}$ input combination ~~all these terms~~	m_2
0	1	1	$\bar{X}YZ$ give output 1	m_3
1	0	0	$X\bar{Y}\bar{Z}$ + each variable	m_4
1	0	1	$X\bar{Y}Z$ included in each product term	m_5
1	1	0	$XY\bar{Z}$ exactly n	m_6
1	1	1	XYZ variable.	m_7

Suppose I want to write a product term in form of XYZ against each combination such that

↳ It includes each variable only once complemented / uncomplemented.

M T W T F S H/W☐-C/W☐ DATE : __/__/20

↳ Result of product term is 1 for that combination.

* Min-term and its properties :-

- A product term in which all the variables appears exactly once either complemented or uncomplemented.
- It represents exactly one combination of binary variable value in truth table.
- It has value one for that combination and "0" for all others.

Analysis of table :-

* For n variable → 2^n distinct minterms.

* For x = 0 we are writing $\bar{x}$ in minterm.

* " x = 1 " " " x "

Same stands for Y and Z.

* symbol for each minterm → m_j

$(j)_{10} = (XYZ)_2$

e.g. $(0) = (000)_2$

$(1) = (001)_2$

$(2) = (010)_2$

Can you write min term for two variables?.

Max terms :- | write sum term for each combination and it ~~contains~~ all variables complemented or uncomplemented.

→ These are called max terms.

x	y	z	Sum terms	Symbol	
0	0	0	$x+y+z$	m_0	* For each combination corresponding
0	0	1	$x+y+\bar{z}$	m_1	maxterms give results "0".
0	1	0	$x+\bar{y}+z$	m_2	* include all variable
0	1	1	$x+\bar{y}+\bar{z}$	m_3	exactly once.
1	0	0	$\bar{x}+y+z$	m_4	In short there are
1	0	1	$\bar{x}+y+\bar{z}$	m_5	all posible sum terms we can
1	1	0	$\bar{x}+\bar{x}+z$	m_6	have with
1	1	1	$\bar{x}+\bar{x}+\bar{z}$	m_7	3 variables.

A sum term that contains all the variables in complemented or uncomplemented form.

n variables → 2^n max terms

if $x=0$ we write x in sum terms

" $x=1$ " " $\bar{x}$ " "

Write possible max term for 2 variables?

Lecture 7

(Relation between)

Property — Minterm and Maxterm

$M_j = \overline{m_j}$

$\overline{M_j} = m_j$

e.g.:

$m_0 = \bar{X}\bar{Y}\bar{Z}$

$\overline{m_0} = \overline{\bar{X}\bar{Y}\bar{Z}}$

$= X + Y + Z$

$= M_0$

step 1: write minterm m_i against each combination

step 2: Select combination or minterm which gives output.

step 3: write function as logical sum of selected minterms.

How to derive a function given with truth table

X	Y	Z	F		$\bar{F}$
0	0	0	1	$\rightarrow \bar{X}\bar{Y}\bar{Z}$	0
0	0	1	0	$\bar{X} Y Z$	1
0	1	0	1		0
0	1	1	0		1
1	0	0	0		1
1	0	1	1	$\rightarrow X\bar{Y}Z$	0
1	1	0	0		1
1	1	1	1	$\rightarrow XYZ$	0

Lecture # 8

What is map?

- A diagram made up of squares.
- Each square representing a minterm

Two - variable map

variable x, y

no of variables $= 2 = n$

no of min terms $= 2^2 = 2^n$

no of cells/squares $= 2^2 = 2^n$

x	y	minterm
0	0	$\bar{x}\bar{x}$
0	1	$\bar{x}\,y$
1	0	$x\,\bar{y}$
1	1	$x\,y$

	0	1
0	$0\,0 = m_0$ $\bar{x}\,\bar{y}$	$m_1 = 0\,1$ $\bar{x}\,y$
1	$1\,0 = m_2$ $x\,\bar{y}$	$m_3 = 1\,1$ $x\,y$

$\longrightarrow$ combination of $x\ y$

information in cells is only for your learning.

$y \backslash x$		
0	0	1
1	2	3

→ no of minterm

How to simplify / optimize boolean equation using K-map.

Given:

1. Truth table. — Required simplified equation
2. $F(A, B) = \Sigma m(\ldots)$ — "
3. Boolean equation — "

Example:

Two variable K-map.

Given:

X	Y	F
0	0	1
0	1	1
1	0	0
1	1	1

step 1:

Draw K-map

step 2:

~~check~~ Enter the function on K-map.

(1) check the minterms in the truth table which are giving value 1

0, 1, 3

(2) write 1 in those cells.

X \ Y	0	1
0	1	1
1		1

→ Now K-map is showing information that output is 1 on minterms

0, 1 and 3

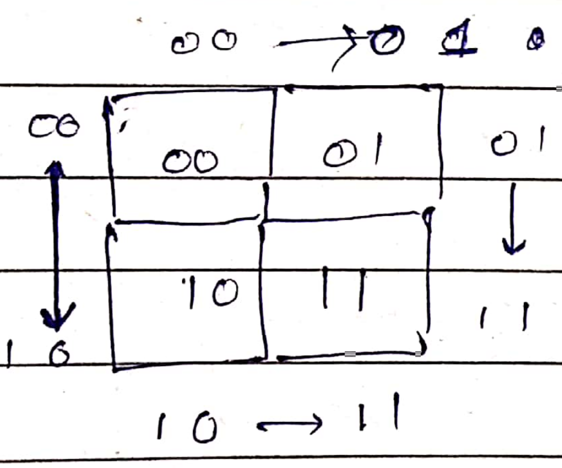
At minterm 2, output

is zero but we don't write in map as we will be working on 1_s only.

Step 3:

Idetify Rectangle of adjacent cells having 1s.

* Adjacent cells combination → which differ in value by only 1 variable. → having adjacent combination



* Total bits = 2
* only 1 should change
↓
Adjacent combination

↓

1 bit changed out of 2 bits

we can define size of rectangles as no of 1s in it.

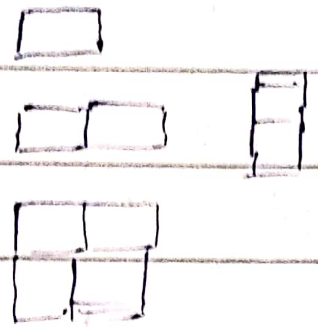
we can have rectangles of size

$2^?$ → no other size allowed

$2^0 = 1$

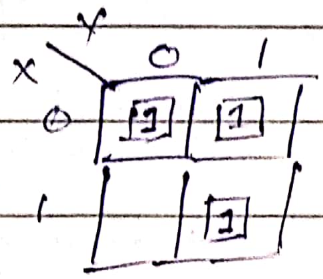
$2^1 = 2$

$2^2 = 4$

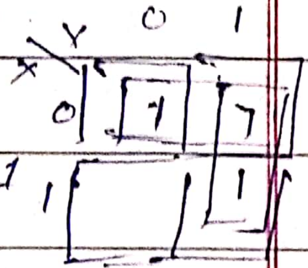


→ either horizontal or vertical.

Now identify all possible rectangles.



← of size 1



our goal is to :-

cover all 1's with least no. of rectangles.

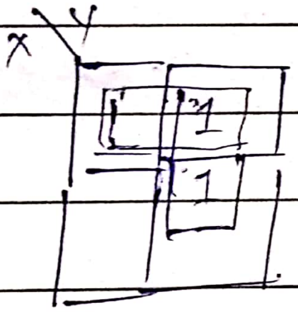
Rectangles should be as large as possible (including maximum no of 1s) step 4 :

Remove extra Rectangles.

is there any rectangle which we can remove and still cover all 1s ??

step 5 :

write simplified expansion against each rectangle

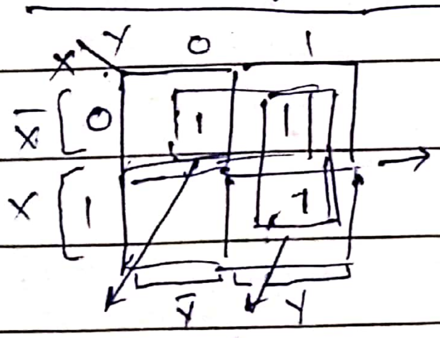


label minterms : while writing minterms we wrote :

$\bar{x}$ for $x = 0$

x " $x = 1$

Similary write in k-map



	$\bar{y}$	y
$\bar{x}$	$\bar{x}\bar{y}$	$\bar{x}y$
x	$x\bar{y}$	xy

$F = \bar{x} + y$

H.W☐ C.W☐ DATE: __/__/20

check the sum of min terms in Rectangle 1.

$= \bar{X}\bar{Y} + \bar{X}Y$

$= \bar{X}(Y + \bar{Y})$

$= \bar{X}$

This is basically what is common between these two cells i.e column in both cells are different
" " same

→ so simplified equation is

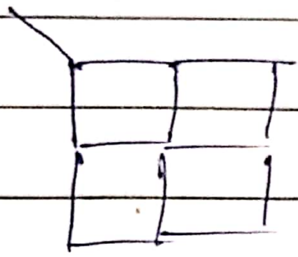
$\bar{X} + Y$

X	Y	F	$\bar{X}$	$\bar{X} + Y$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

Scanned with CamScanner

$G(A, B) = \Sigma m(1, 2)$

notice function variables are A • B



for 2 variable K-map

Rectangle size	no of variable to express it
2	1
1	2

Lecture 9

Three variable maps:

$F(A, B, C) = \Sigma m\,(0,1,2,3,4,5)$

Notice the variables A, B, C

step 1:

Draw K-map

no of variable = 3

" " minterm = 2^3

" " cells = $2^3 = 8$

A \ BC	00 ($\bar{B}$, $\bar{C}$)	01 ($\bar{B}$, C)	11 (B, C)	10 (B, $\bar{C}$)
$\bar{A}$ 0	1 (0)	1 (1)	1 (2)	1 (3)
A 1	1 (4)	1 (5)	(7)	(6)

possible values

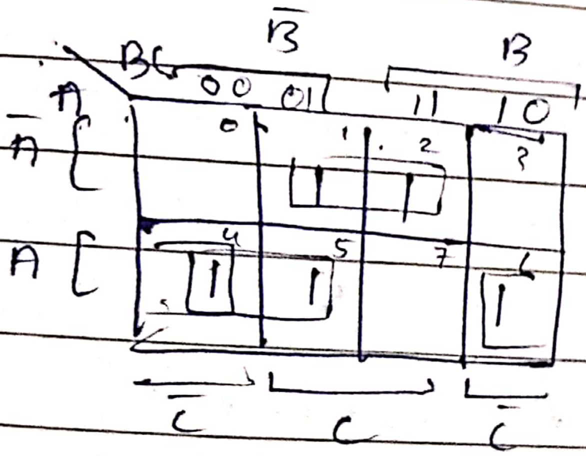
$F = \bar{A}\,(1)(1) + \bar{B}\,(1)(1)$

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

Proof of 1st row:

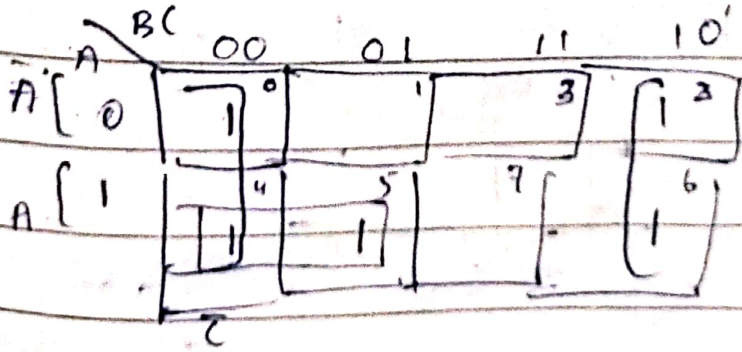
$$= \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + \bar{A}B\bar{C}$$
$$= \bar{A}\bar{B}(C+\bar{C}) + \bar{A}B(C+\bar{C})$$
$$= \bar{A}\bar{B} + \bar{A}B$$
$$= \bar{A}(\bar{B}+B) \Rightarrow \bar{A}$$

→ $H(A,B,C) = \Sigma m(1,3,4,5,6)$



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

→ $G(A,B,C) = \Sigma m(0,2,4,5,6)$



why we swap 3 with 2?

Ans: Because we have to with adjacent cells.

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

$= 1(1)\bar{C}) = \bar{C}$

$A\bar{B}$

$\longrightarrow \bar{C} + A\bar{B}$

$\hookrightarrow$ Four variable map simplification:

$F(A, B, C, D) = \Sigma m\,(0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 13)$

no of variables = 4

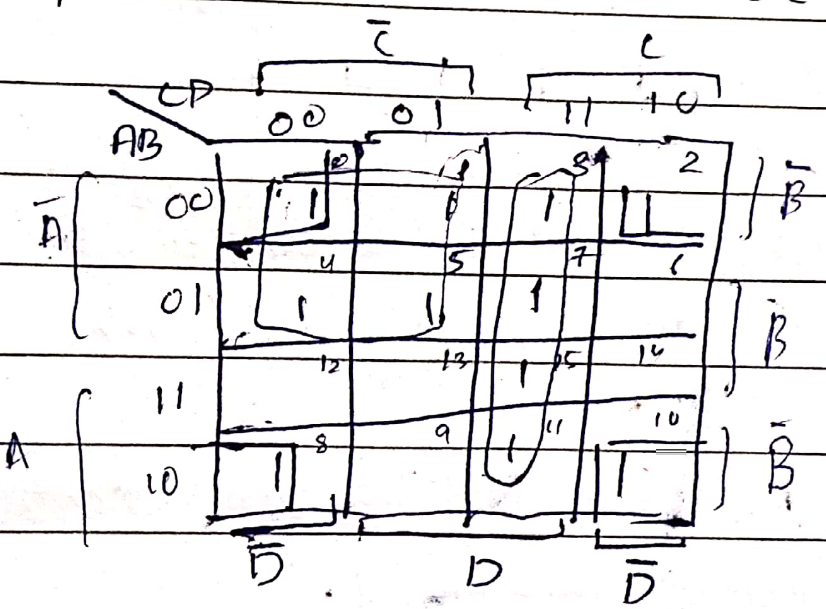
cells $= 2^4 = 16$

AB \ CD	00 ($\bar{C}$)	01 ($\bar{C}$)	11 (C)	10 (C)	
$\bar{A}$ 00	1	1		1	$\bar{B}$
$\bar{A}$ 01	1 ⁴	1 ⁵	⁷	1 ⁶	B
A 11	1 ¹²	1 ¹³	¹⁵	¹⁴	B
A 10	1 ⁸	1 ⁹	¹¹	1 ¹⁰	$\bar{B}$
	$\bar{D}$	D		$\bar{D}$	

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

Making K-map with the help of equation:

$G(A, B, C, D) = \bar{A}\bar{C}\bar{D} + \bar{A}D + \bar{B}C + CD + A\bar{B}\bar{D}$



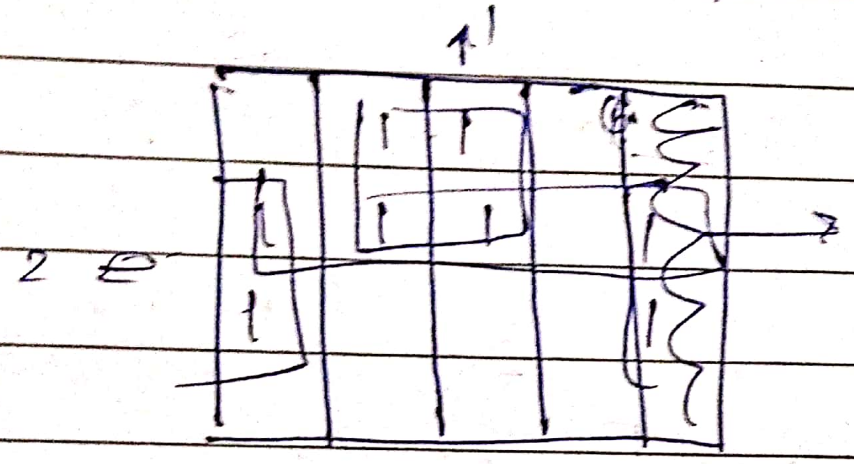
$\bar{A}\bar{C} + CD + \bar{B}\bar{D}$

M T W T F S H/W☐-C/W☐ DATE: __/__/20

Prime Implicants.

Essential Prime Implicants

non essential prime Implicants

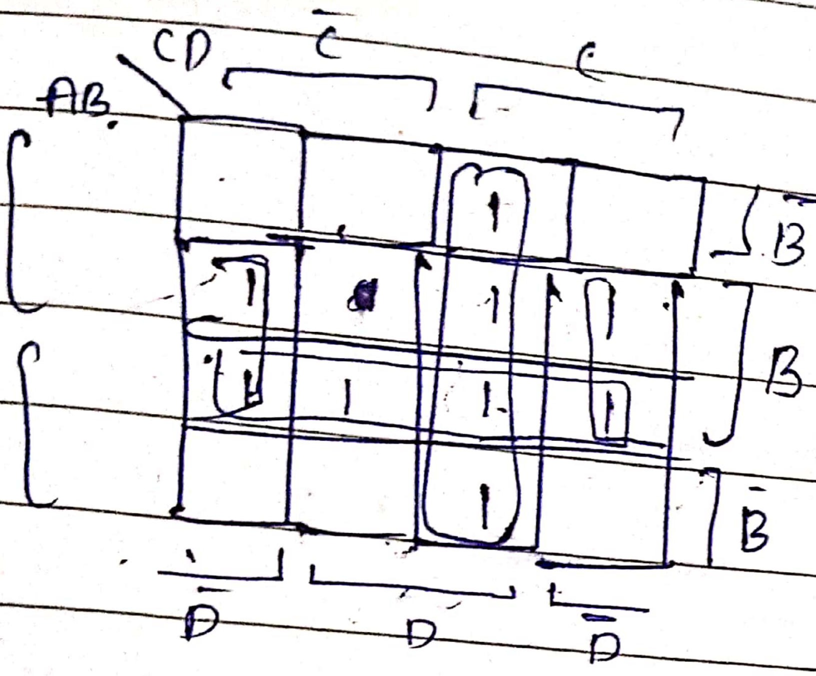


All these 3 rectangles are called prime implicants..

Rectangle. ③. is non-essential prime implicants.

Rectangle ① and ② are called essential prime implicants.

Product of sum optimization:



H.W :

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

Write in product of sum terms.

DATE: __/__/20

Lecture 10

Don't care conditions:-

Suppose we have to design a circuit for a bulb with following requirements.

- Bulb should ON on input 0, 2, 4 and 6.
- Bulb should OFF on input 3 and 5.
- For rest of combinations we don't care about results.

Outputs are said to be unspecified for input combinations 1 and 7.

- Functions that have unspecified outputs for some input combinations are called incompletely specified functions.

M T W T F S H/W C/W DATE / /20

→ unspecified minterm of function are called don't care. conditions are marked as X.

• Don't care minterms cannot be marked as 1 as they

Scanned with CamScanner

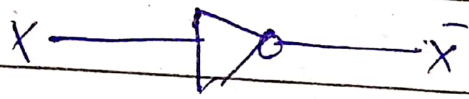
Other gate types:

Buffer:

$F = X$

X	F
0	0
1	1

used to amplify an electrical signal. if we add negation indicator / Bubble at the output it gives inverter symbol.

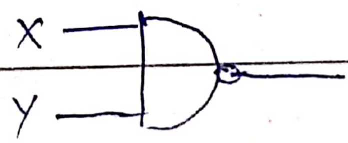


NAND: Not AND

X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

Algebric equation: $F = \overline{X \cdot Y}$

symbol:



Universal gate:

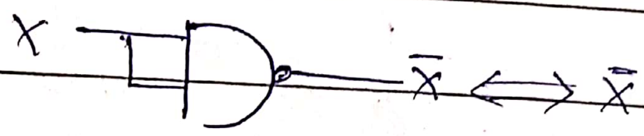
A gate type that alone can be used to implement all boolean functions is called universal gate.

① Not logical operation:

Equation of NAND: $F = \overline{X \cdot Y}$

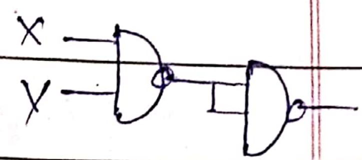
if we put $Y = X$

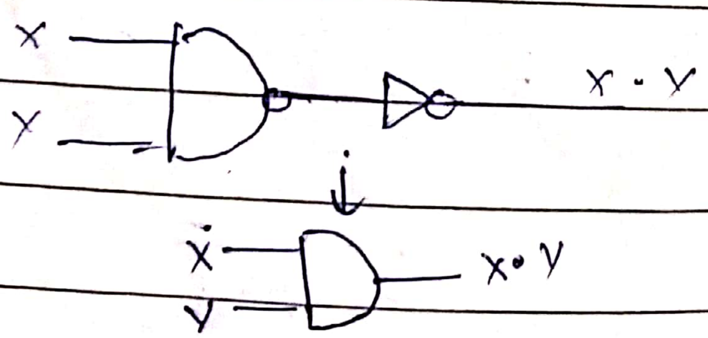
$F = \overline{X \cdot X} = \bar{X}$ complement operation



② AND logical operation:

$F = \overline{X \cdot Y}$





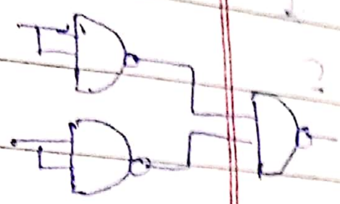
(3) OR logical operation:

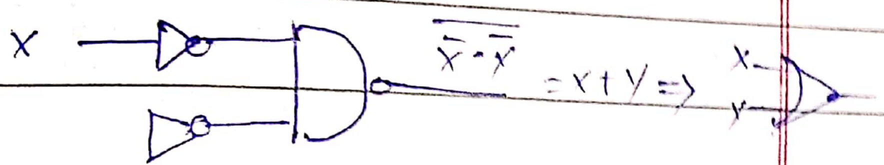
OR

Required $F = X + Y$

$= \overline{\overline{X + Y}}$

$= \overline{\bar{X} \cdot \bar{Y}}$

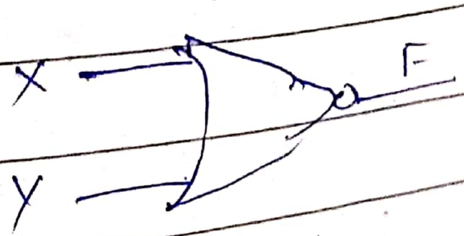




Since we can perform NOT, NAND and OR logical operation using NAND gate so it is a universal gate.

NOR gate: (NOT OR)

Algebric equation: $\overline{X+Y}$



X Y	F
0 0	1
0 1	0
1 0	0
1 1	0

M-W/S

Can you prove that NOR is a universal gate??

⇒ XOR gate (Exclusive OR):

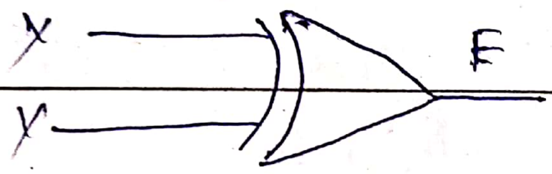
Algebric eq: $F = X\bar{Y} + \bar{X}Y$

$= X \oplus Y$

↓

XOR operation

Graphic symbol:



* Give 1 output only for ODD nos of 1 input.
* if both variables are different, only then $X\bar{Y}$ and $\bar{X}Y$ will give result = 1

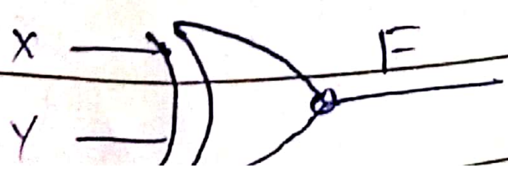
Available in two input only.

XNOR gate: (Exclusive NOR)

$F = XY + \bar{X}\bar{Y}$

$= \overline{X \oplus Y}$

X Y	F
0 0	1
0 1	0
1 0	0
1 1	1



M T W T F S   H/W C/W   DATE / / 20

XOR . XNOR are used

→ To reduce circuit complexity.

→ " " " cost .

→ " " time required for to propagate through a circuit .

HOW :

Can you prove that XNOR and XOR are complement of each other.

XOR

0 0	0
0 1	1
1 0	1
1 1	0

XOR

M T W T F S   H/W ☐ -C/W ☐   DATE: __/__/__

→ identities apply to XOR operation:

$X \oplus 0 = X$ $\quad$ $X \oplus 1 = \bar{X}$

$X \oplus X = 0$ $\quad$ $X \oplus \bar{X} = 1$

$X \oplus \bar{Y} = \overline{X \oplus Y}$ $\quad$ $\bar{X} \oplus Y = \overline{X \oplus Y}$

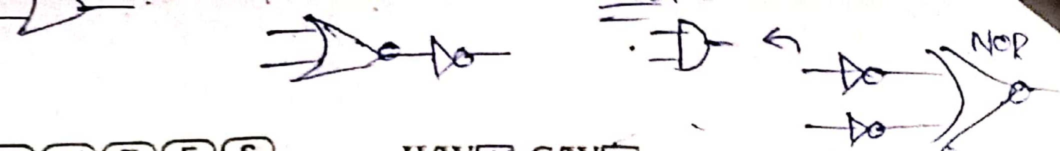
Identities can be proved by replacing $\oplus$ operation by its boolean operation.

$A \oplus B = B \oplus A$ (commutative)

$(A \oplus B) \oplus C = A \oplus (B \oplus C) =$

$A \oplus B \oplus C$

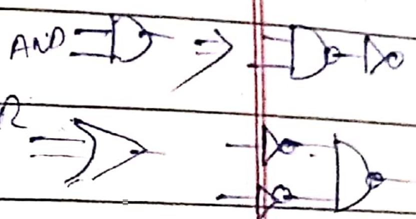
(Associative)



M T W T F S   H/W C/W   DATE: / /20

Note

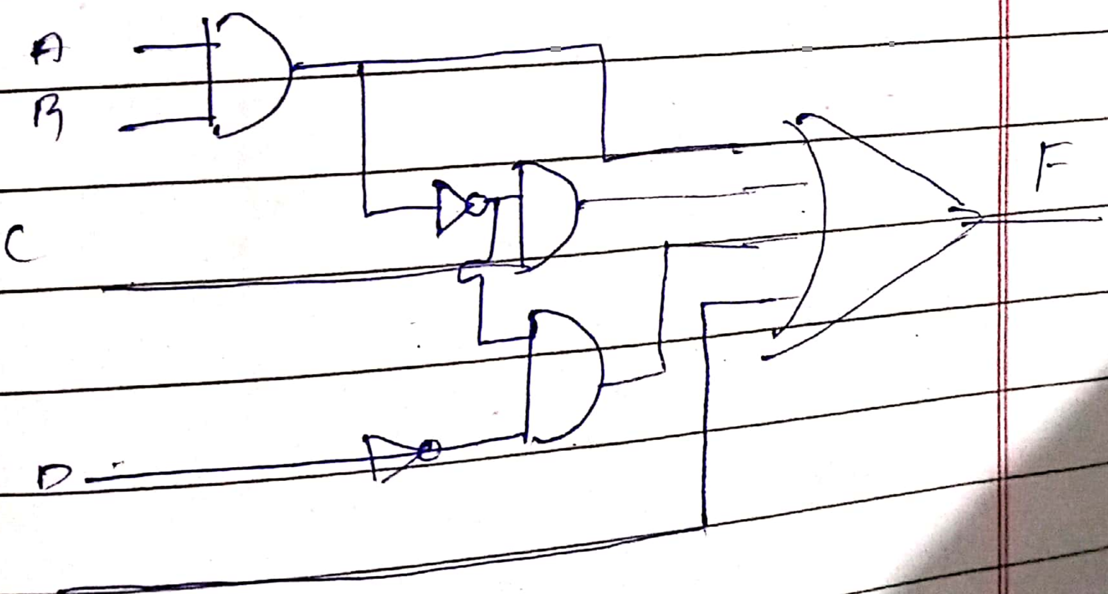
Technology mapping :-



Implementation with NAND / NOR:

Implement following optimized function with NAND gates.

$$F = AB + (\overline{AB})C + (\overline{A}B)\bar{D} + E$$

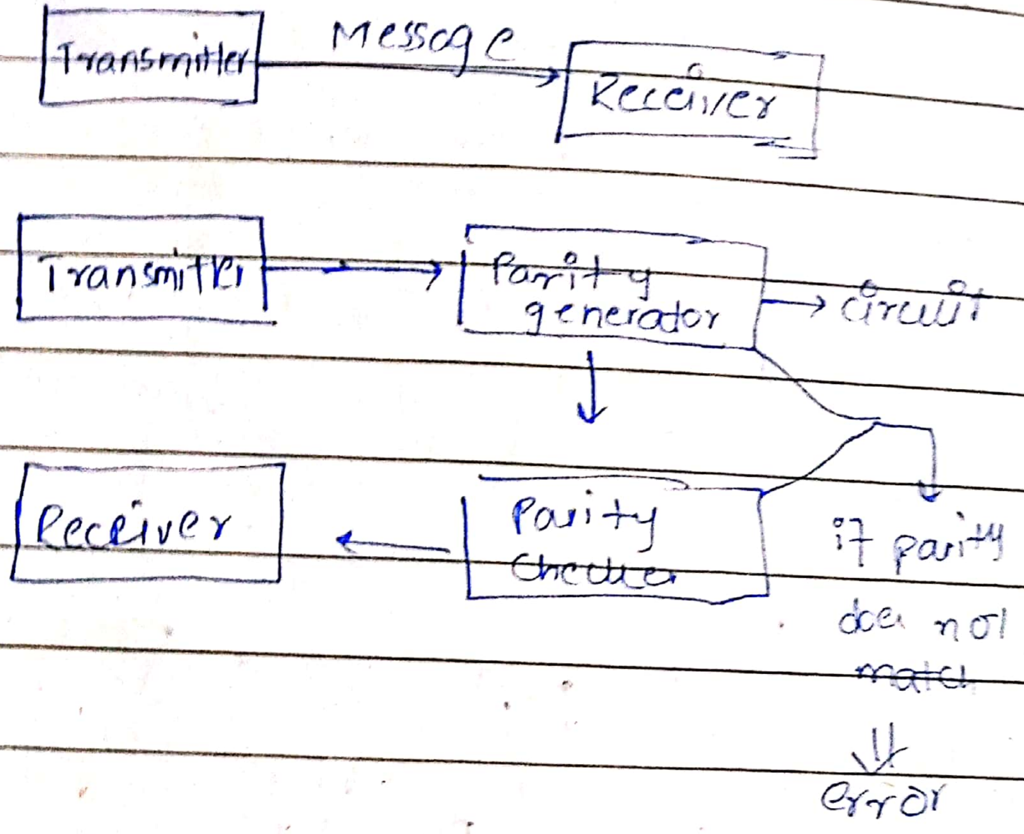


H.W 8

Implement them in NOT and NOR

Lecture # 11

→ Parity Generation and checking:



Lets design circuit of parity generator:

Requirement:

We have to transmit 3 bit message with 4th bit as parity.

First make truth table

DATE: __/__/20

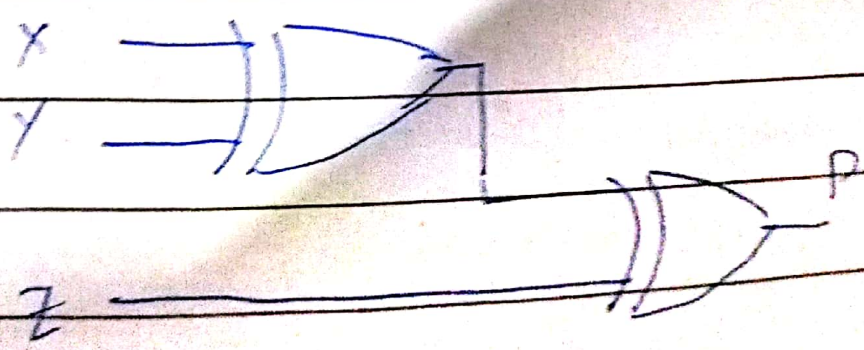
2^3 8
2^4 16

Error parity generator :

Three bit message

X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$F = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$$



$$P = X \oplus Y \oplus Z$$

→ Parity checker:

Require if parity is odd → error (1)
else → 0

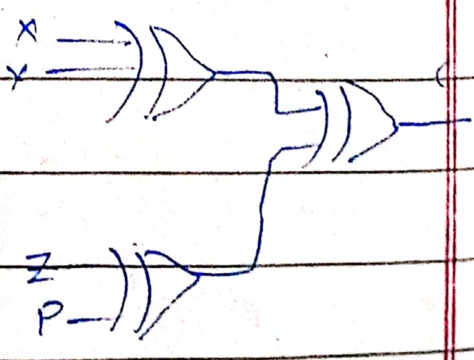
x	y	z	p	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

C = 1 → error

C = 0 → non error

0	1		1
1		1	
	1		1
1		1	

$C = X \oplus Y \oplus Z \oplus P$



→ Combinational logic design Procedure :

given a problem statement we have to design a circuit

example:

design of A B C D - to excess 3 code converter.

step 1 :

Specification / Requirements :-

given a decimal digit (in binary coded decimal) produce excess - 3 code for that digit

e.g : decimal digit 5

5 + 3 = 8 = 1000

↓

Excess 3 of 5

Excess 3 code of decimal digit = Binary combination corresponding to decimal digit plus 3.

Step-2: Formulation:

Derive Truth table or initial equation that define relationship between circuits.

* we have to handle all 2^n combination of a related circuit.

Decimal digit	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
	1	0	1	0	x	x	x	x
	1	0	1	1	x	x	x	x
	1	1	0	0	x	x	x	x
	1	1	0	1	x	x	x	x
	1	1	1	0	x	x	x	x
	1	1	1	1	x	x	x	x

For w:

AB \ CD	00	01	11	10
00	0			
01		1	1	1
11	X	X	X	X
10	1	1	X	X

For x:

AB \ CD	00	10	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

For y:

AB \ CD	00		11	
	1		1	
	1		1	
	X	X	X	X
	1		X	X

4 bit equality comparator:-

$a_3 a_2 a_1 a_0 = b_3 b_2 b_1 b_0$

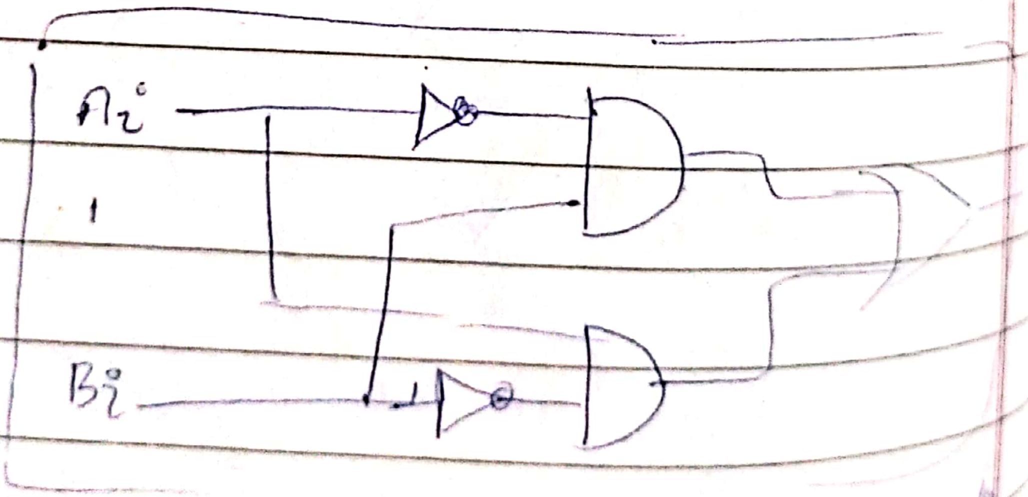
$0000 = 0000 \Rightarrow 1$

$0110 \neq 0100 \Rightarrow 0$

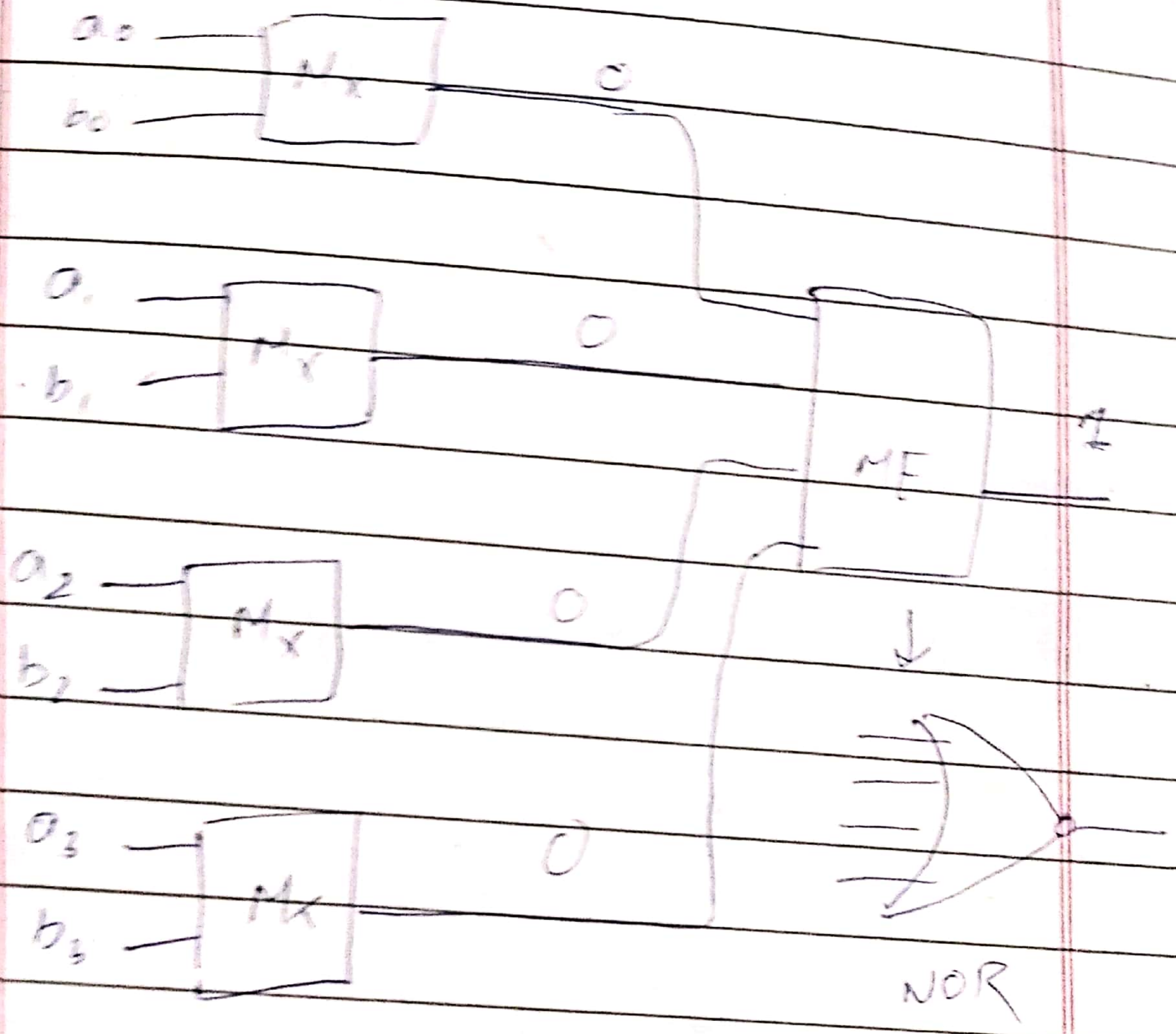
Hierarchical design (divide & conquer)

A_i	B_i	N_i
0	0	0
0	1	1
1	0	1
1	1	0

$\overline{A_i} B_i + A_i \overline{B_i}$



1 bit equality comparator



$$\overline{N_1 + N_2 + N_3 + N_4}$$

H.W.:

99 input odd function