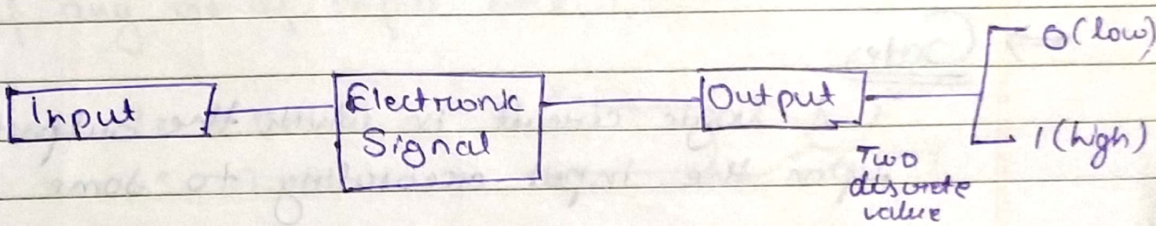
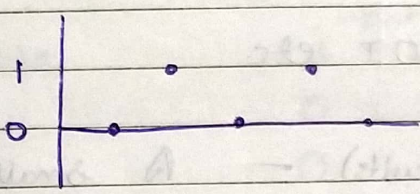
⇒ Digital Electronic - is the branch of electronic which deals with a digital signals to perform various task to meet various requirement.

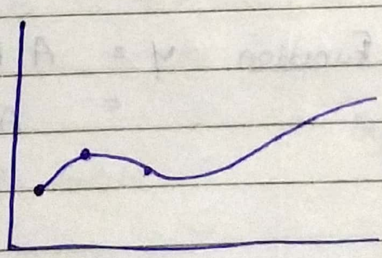
The input signal applied to this circuit is of digital form which is represented in 0 & 1 (Binary lang. format) for example - electronic calculator.





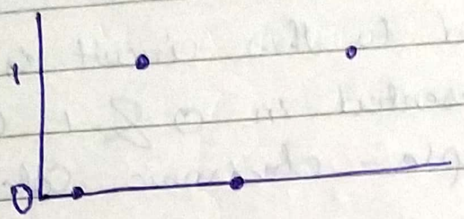
⇒ what is signal?
A signal is a function, that represent the variation of a physical quantity with respect to any parameter.

⇒ Analog Signal - A continuous signal that can have any values in a given range it is also known as continuous signals.



Page No.___ Date _/_/20_

⇒ Digital Signal - Signal with only two discrete values there are usually represented by low and high for 0 & 1



⇒ **Gates**

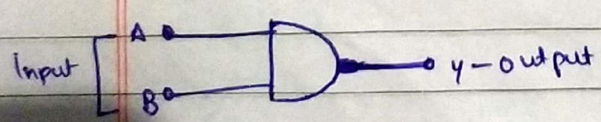
A logic circuit in which the output depends upon the input according to some logic rules.

like - AND, OR, NOT etc

⇒ ICs (Integrated Circuits) - A small semiconductor chip containg several electronic circuit.

⇒ Truth table - A table that gives output for all possible combination of input to a logic circuit.

⇒ AND Operation - A logic circuit whose output is 1 if and only if all the inputs are 1.



Function y = A AND B

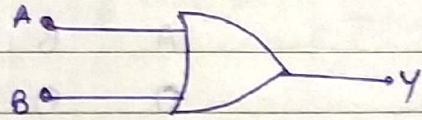
= A · B

Truth table

Inputs A	Inputs B	output Y
0	0	0
0	1	0
1	0	0
1	1	1

⇒ OR operation - A logic circuit whose output is 1 if any one of input are 1.

Function Y = A OR B
  = A + B

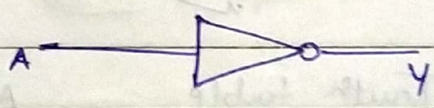


Truth table

Inputs A	Inputs B	output Y
0	0	0
0	1	1
1	0	1
1	1	1

⇒ NOT operation - A logic circuit in which we give one input and get one output and Its also known as inverter. It gives output inverse of input.

Function Y = NOT A
  = $\bar{A}$



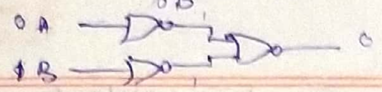
Truth table

Input A	output Y
0	1
1	0

NOT using NOR

OR

AND

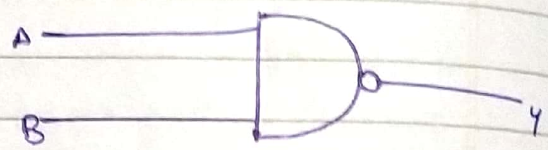


⇒ $\underbrace{\text{NAND}}$ operation – A logic gate whose output
NOT AND is zero if ~~any~~ and only if all its input is logic one.

Function $Y = A \text{ NAND } B$

$= A \text{ NOT AND } B$

$= \overline{AB}$

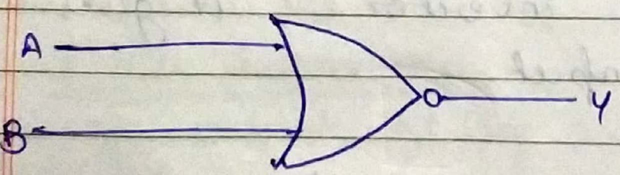


Truth table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

⇒ $\underbrace{\text{NOR}}$ operation – A logic gate whose output is
NOT OR zero if any of input is one.

A logic gate whose output is one if and only if all its input are logic is zero.



Function $Y = A \text{ NOR } B$

$Y = A \text{ NOT OR } B$

$= \overline{A+B}$

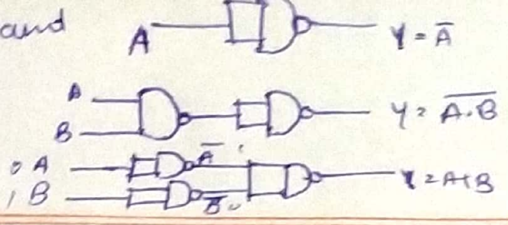
Truth table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NOT using Nand

AND " "

OR " "



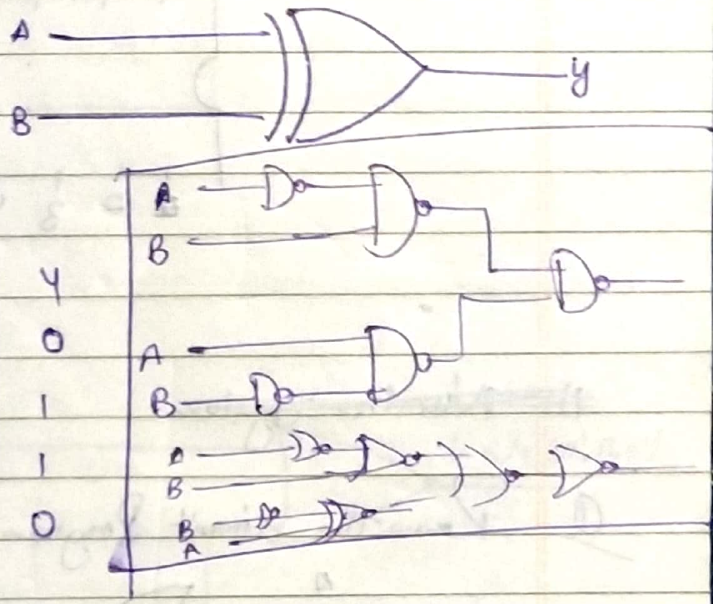
⇒ Exclusive OR - Gate / 'XOR' Gate

A two input gates whose output is logic zero if both the input are equal and one when the inputs are unequal.

function. $Y = A \text{ XOR } B$

$A \oplus B$

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



Truth table

A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

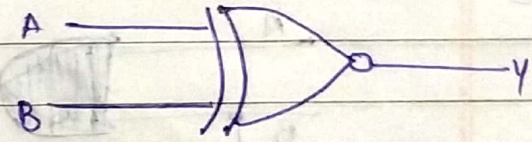
⇒ Exclusive NOR Gate - / X-NOR

A two input gate whose output is logic one when both the input are same.

function $Y = A \text{ XNOR } B$

$= \overline{A \oplus B}$

$= A \odot B$

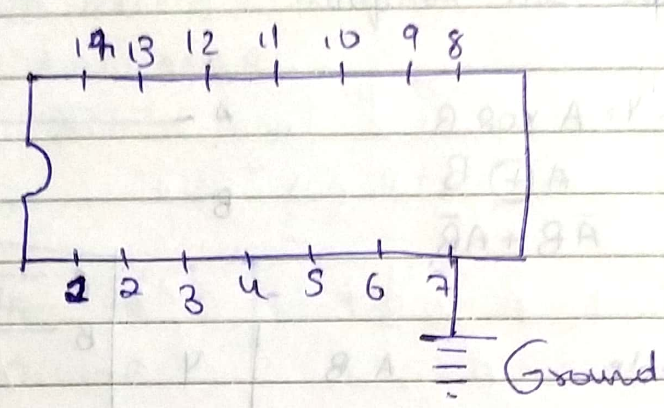


Truth table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

* Integrated Circuit (IC)

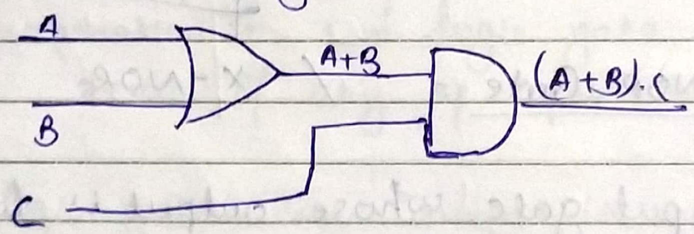
IC or Integrated circuit is a small semiconductor chip containing several electronic circuits.



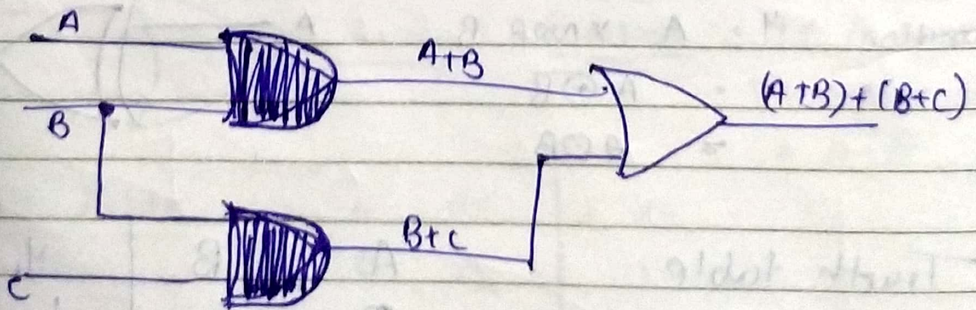
NOT	7404
AND	7408
OR	7432
NAND	7400
NOR	7402
Ex OR	7486

~~# Number System~~

Q Draw the circuit Program of (A+B).C



Q (A•B) + (B•C)



Boolean Algebric Theoram

$A + 0 = A$

$A \cdot 1 = A$

$A + 1 = 1$

$A \cdot 0 = 0$

$A + A = A$

$A \cdot A = A$

$A + \overline{A} = 1$

$A \cdot \overline{A} = 0$

$A \cdot (B + C) = AB + AC$

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

$A + AB = A$

$A(A + B) = A$

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

$A(\overline{A} + B) = AB$

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

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

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

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

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

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

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

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

Number System

⇒ Binary to Decimal $(\times 2^n)$

Ex $(1111)_2 \longrightarrow (\ )_{10}$

$1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$

$8+4+2+1 = (15)_{10}$ Ans

⇒ Decimal to Binary

Ex $(13)_{10} \longrightarrow (\ )_2$

$\frac{13}{2}$	6	1
$\frac{6}{2}$	3	0
$\frac{3}{2}$	1	1
$\frac{1}{2}$	0	1

$(1101)_2$

⇒ $(0.65625)_{10} \longrightarrow ()_2$

0.65625 ×2	1.31250 ×2	0.62500 ×2	.25000 ×2	0.500 ×2
1.31250	0.62500	1.25000	.0.50000	1.00
1	0	1	0	1

$(0.65625)_{10} \longrightarrow (10101)_2$

⇒ Signed Binary number → Decimal.

$(101100)_2 \longrightarrow (-12)_{10}$

1 = −
0 = +

$(01000)_2 \longrightarrow (+8)_{10}$

⇒ One's Complement

0100911001 → 1011000110

⇒ Two Complement

i) 01001110 → Complement + 1 (2)

01001110 →

10110001
+ 1

Ans

→ Binary Addition

		Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
    1 0 1 1
+   1 1 0 0
-----------
  1 0 1 1 1
```

$\underbrace{1}_{\text{carry}}0111$

```
   (1)  (1)
    0 1 0 1
+   1 1 1 1
-----------
  1 0 1 0 0
```

$\underbrace{1}_{\text{carry}}0100$

→ Binary Subtraction

		Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

```
    1 0 1 1
-   0 1 1 0
-----------
    0 1 0 1
```

⇒ Binary Multiplication

```
        1 0 0 1
        1 0 0 1
      ---------
        1 0 0 1
      0 0 0 0 x
    1 0 0 1 x x
  1 0 0 1 x x x
  -------------
  1 1 1 0 1 0 1
```

⇒ Binary Division

```
           1 1 0 1
      ------------
 1001 ) 1 1 1 0 1 0 1
        1 0 0 1
        -------
        0 1 0 1 1
          1 0 0 1
          -------
            1 0 0 1
            1 0 0 1
            -------
                  0
```

~~⇒ Complement Arithmetic (Add / Sub in 2's comp rep)~~

~~48 - 23~~

~~2's complement repr of +48 =~~

Octal number system.

The no system with base (or radix) 8 is known as octal number system. In this system ~~digit~~ symbol 0 1 2 3 4 5 6 7 are used to represent no.

⇒ Octal ⟶ decimal

$(6327.4051)_8 \longrightarrow (\quad)_{10}$

$6\times8^3 + 3\times8^2 + 2\times8^1 + 7\times8^0 + 4\times8^{-1} + 0\times8^{-2} +$
$5\times8^{-3} + 1\times8^{-4}$

$3072 + 192 + 16 + 7 + \frac{1}{2} + 0 + \frac{5}{512} + \frac{1}{4096}$

$(3287.51001)_{10}$

$\Rightarrow$ Decimal to Octal

$(247)_{10} \longrightarrow (\quad)_8$

$\frac{247}{8}$	30	7
$\frac{30}{8}$	3	6
$\frac{3}{8}$	0	3

$(367)_8$

$(0.6875)_{10} \longrightarrow (\quad)_8$

$0.6875 \times 8 = 5.5 \qquad 0.5 \times 8 = 4.0$

$(0.54)_8$ Ans

Red Wings
Page No.___ Date __/__/202

Octal to Binary Conversion

Octal number can be converted into equivalent binary number by replace each octal digit by its 3 bit equivalent binary. In octal number we can take 0 to 7 decimal

$(736)_8 \longrightarrow (\ )_2$

8 4 2 1
code

$(0111\ 011\ 110)_2$

$(\ )_2 \rightarrow (\ )_8$

$(001\ 001\ 110)_2 \longrightarrow (\ )_8$

$(116)_8$

$(0.10100110)_2 \longrightarrow (\ )_8$

101 001 100

$(0.514)_8$

011 001 110 001 . 000 101 111 001

$(3161.0571)_8$

Octal Arithmatic

$$\begin{array}{r} 23 \\ +\ 67 \\ \hline (112)_8 \end{array} \qquad \begin{array}{r} =\ 010011 \\ =\ +\ 110111 \\ \hline 1001010 \end{array}$$

(1) (1)

carry

Hexadecimal N S

$\Rightarrow ( )_{16} \rightarrow ( )_{10}$

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

$(3A.2F)_{16} \rightarrow ( )_{10}$

$3\times16^{1} + 10\times16^{0} + 2\times16^{-1} + 15\times16^{-2}$
$= (58.1836)_{10}$

$\Rightarrow ( )_{10} \rightarrow ( )_{16}$
$(95.5)_{16} \rightarrow ( )_{10}$

95 / 16	5	15 (F)	
5 / 16	0	5	

.5
× 16
8.0

16
32
48
64
80
96

$(5F.8)_{16}$

$( )_{16} \rightarrow ( )_{2}$ 4 bit
$(2F9A)_{16}$
$(0010\ \ 1111\ \ 1001\ \ 1010)_{2}$

$( )_{2} \rightarrow ( )_{16}$
$(0010\,1001\,1010\,1111)_{2}$
2 9 A F
$(29AF)_{16}$

Page No. ___ Date _/_/202_

$( )_{16} \rightarrow ( )_8$

$(A72E)_{16} \rightarrow (1010\ 0111\ 0010\ 1110)_2$

$(001\,010\,011\,100\,101\,110)_2 \rightarrow ( )_8$

$(123456)_8$

2's Complement Arithmetic

7 → 0111

-5 → 1101 ← find 2's complement

0010
+1
0011

7 → 0111
-5 → + 1011
1 0010
Discard

2 → 0 101
-7 → 1 111

000
+1
001

0101
+1001
1110

The final carry is 0

∴ answer is negative and is in 2's comp form

2's comp of 1110 = 0010

∴ $(-2)_{10}$

Codes

- weighted Codes
  eg Binary code
  8421 code
  2421 etc
- Non weighted
  eg XS-3
  gray code
- Reflective
  (self complentary)
  eg 2421 code
  XS-3
- Sequently code
  eg 8421
  XS 3
- Alpnumeric code
  eg ASCII
- Error Detecting & Correcting codes
  eg Hamming code

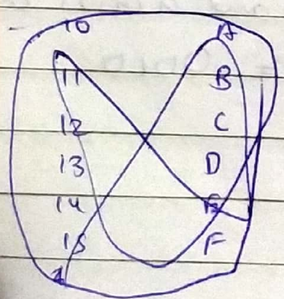
ASCII — American standard code for Information Interchange

→ BCD (Binary coded Decimal) Code — 8421

Decimal	BCD 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal → BCD
$(17)_{10} \rightarrow (0001\ 0111)$ BCD

Compare Binary & BCD
$(10)_{10} \rightarrow$ Binary → BCD
1010 (0001 0000)



Shift 3 Add   2421 code

Excess 3

5 → 0101 $\xrightarrow{+3}$ 0101
0011
1000

		XS-3
0	0001	0000
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

24

0010 0100
0011 0011
0101 0111
5 7 XS code for 24

Gray code (cyclic code)

Binary → GC

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

1 0 1 1
1 0 1
1 1 1 0 Gray code

GC → Binary

1 1 1 0
1 0 1
1 0 1 1 Binary code

⇒ **Parity** The simplest technique for detecting errors is to add an extra bit known as parity bit to each word being transmitted

MSB							LSB
P	d_6	d_5	d_4	d_3	d_2	d_1	d_0

The parity of 8 bit transmitted word can be either even or odd parity

→ Even parity means the number of 1's in the given word including the parity bit should be even (2, 4, 6 – – ∞)
odd – (1, 3, 5, – – –)

i) Even ⇒ (2, 4, 6) है तो — 0
(1, 3, 5) — 1

ii) odd (1, 3, 5) — 0
(2, 4, 6) — 1

eg

P							
0	1	0	1	0	0	1	0	1

Parity 0 1's even
So Parity odd type

Hamming code
- Data bits – 4
- Parity bits – 3

7	6	5	4	3	2	1

Position for parity bit 2^n =
$2^0 = 1$
$2^1 = 2$
$2^2 = 4$

D_7	D_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1

Red Wings Page No.___ Date _/_/202_

⇒ Selection P_1 → consider bits 1, 3, 5, 7, 9, 11, 13, 15

P_2 → ——— 2, 3, 6, 7, 10, 11, 14, 15

P_4 → ——— 4, 5, 6, 7, 12, 13, 14, 15

P_8 ——— → 8, 9, 10, 11, 12, 13, 14, 15

Ex: 1011011 with the even parity check code correct or not.

Sol:

1	0	1	1	0	1	1
D_7	D_6	D_5	P_4	D_3	P_2	P_1

P_1 D_3 D_5 D_7

1 0 1 1 There is error exists $P_1 = 1$

P_2 D_3 D_6 D_7

1 0 0 1 No error $P_2 = 0$

P_4 D_5 D_6 D_7

1 1 0 1 There is error exist $P_4 = 1$

~~there 3 1's and parity for 3~~ even and P_4

P_4 & P_1 are not equal to zero so recieved code is wrong.

Correcting error:

error word ⇒

P_4	P_2	P_1
1	0	1

decimal value = $(5)_{10}$

this show that 5^{th} bit is in error

So we write the correct word by simply inverting the 5 bit

1	0	0	1	0	1	1

Ex 1110101 Even

Sol

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	0	1	0	1

Step1 check bit 4, 5, 6, 7

$P_4 D_5 D_6 D_7$ → 0 1 1 1 → odd parity

There error exist $P_4 = 1$

Step2 check bit 2, 3, 6, 7

P_2, D_3, D_6, D_7 — 0 1 1 1 → odd parity

There error in exist $P_2 = 1$

Step3 check bit 1, 3, 5, 7

P_1 D_3 D_5 D_7 — 1 1 1 1 → even parity

There is no error $P_1 = 0$

Error word $E = \boxed{P_4 | P_2 | P_1}$ → $\boxed{1 | 1 | 0}$

$E = (6)_{10}$

Hence bit 6 of the transmitted word is in error

1	1	1	0	1	0	1

↑ Incorrect bit

Correct Code — Invert the incorrect bit

1	0	1	0	1	0	1

Page No. ___ Date _/_/202

Ex 0 0 1 1 1 0 0 1 0 1 1 0 1 odd

Sol

0	0	1	1	1	0	0	1	0	1	1	0	1
D_{13}	D_{12}	D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1

Check Check bit 8, 9, 10, 11, 12, 13

P_8 D_9 D_{10} D_{11} D_{12} D_{13} → 0 1 1 1 0 0 odd parity

There is not error $P_8 = 0$

Check check bit 4, 5, 6, 7, 12, 13, 14, 15

P_4 D_5 D_6 D_7 D_{12} D_{13} → 1 0 1 0 0 0 even parity

There is error $P_4 = 1$

Check Check bit 2, 3, 6, 7, 10, 11, 14, 15

P_2, D_3 D_6 D_7 D_{10} D_{11} 0 1 1 0 1 1 even parity

There is error $P_2 = 1$

Check bit 1, 3, 5, 7, 9, 11, 13, 15

P_1 D_3 D_5 D_7 D_9 D_{11} D_{13} — 1 1 0 0 1 1 0 even parity

Error Exist $P_1 = 1$

P_8 P_4 P_2 P_1

Error word E = | 0 | 1 | 1 | 1 |

$E = (7)_{10}$

In transmitted word 7^{th} bit is in error

Inverted 7^{th} bit

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