→ Digital system processing unit is the system that converts ~~along~~ analog signals to digital signals.

DIGITAL SYSTEM: refers to elements such as hardware, software and networks and their uses. Designed to store, process and communicate information in digital form.

DESIGN: Plans or technique.

→ Gates and integrated circuits (chips) are present in digital system.

* DECIMAL:

→ We are accousted to so called decimal number systems.

10 digits: 0 to 9.

Every digit position has a weight which is power of 10.

Base or Radix: 10.

→ The value associated with a digit is ~~de~~ dependent on its position.

→ The value of number is weighted sum of its digits.

Eg: $2357. = 2\times10^3 + 3\times10^2 + 5\times10^1 + 7\times10^0$

→ Decimalpoint allows -ve and +ve powers of 10.

Eg: $526.47 = 5\times10^2 + 2\times10^1 + 6\times10^0 + 4\times10^{-1} + 7\times10^{-2}$

MSD LSD.

MSD - most significant digit

LSD - least significant digit

* BINARY:

2 digits - 0 and 1.

Every digit position has a weight that is power of 2.

Base or Radix: 2

Binary digits are called BITS. (Binary digits)

Eg: $110 = 1\times2^2 + 1\times2^1 + 0\times2^0 = (6)_{10}$

$101.01 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$

$1101.11 = (13.75)_{10}$

MSB LSB

* OCTAL:

→ A compact way to represent binary numbers.

→ Group of 3 binary digits are represented as octal.

octal digits : 0 to 7.

Base & Radix: 8.

Eg: $(123)_8 = 1\times8^2 + 2\times8^1 + 3\times8^0 = 64+16+3 = (83)_{10}$

MSD ↑ LSD ↑

* HEXADECIMAL:

→ A compact way to represent binary numbers.

→ Group of 4 binary digits are represented as hexadecimal.

Digits: 0 to 9, A to F.

Base & Radix: 16.

Eg: $(A3C) = 10\times16^2 + 3\times16^1 + 12\times16^0$

MSD ↑ LSD ↑

* GENERALISATION OF RADIX-BASED NUMBER SYSTEM.

Radix (r): no. of distinct digits

- Assume that the digits are $(0, 1, 2 - - - (r-1))$.
- Every digit position has a weights that is some power of r (say r^k).

$K \geq 0$, for integer parts

$K < 0$, for fractional parts

A(n+m) - digit number representation:

$D = d_{n-1} d_{n-2} - - - - d_1 d_0 . d_1 d_2 - - - - - d_{-m}$.

* Any radix-r to Decimal conversion:

Multiplying each radix digit by its positional weights and adding the resulting product.

Eg: Binary to decimal:

$(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (11)_{10}$.

Octal to decimal:

$(127)_8 = 1\times8^2 + 2\times8^1 + 7\times8^0 = (87)_{10}$.

Binary to decimal:

$(10101)_2 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = (21)_{10}$.

Hexadecimal to decimal:

$(9AC7)_{16} = 9\times16^3 + 10\times16^2 + 12\times16^1 + 7\times16^0 = (39623)_{10}$

Any radix to decimal:

$(235)_6 = 2\times6^2 + 3\times6^1 + 5\times6^0 = (95)_{10}$.

* Decimal to any radix:

- For integer part, repeatedly divide number by r and accumulate the remainder. Remainders are arranged in reverse order.
- For fractional part, repeatedly multiply by r and accumulate and discard the integer part. The digits are arranged in the order they are generated.

Eg: Decimal to Binary:

$(41)_{10} = (101001)_2$

2	41
2	20 - 1
2	10 - 0
2	5 - 0
2	2 - 1
	1 - 0

$(0.75)_{10} = 0.75\times2$

$= 1.50\times2$ (carry 1) $= 0.5\times2 = 1.0$ (carry 1).

i.e, $(0.75)_{10} = (0.11)_2$

Decimal to octal:

$(153)_{10}$

8	153
8	19 - 1
	2 - 3

$= (231)_8$.

* BINARY TO OCTAL:

Divide binary numbers into 3 groups starting from LSB and moving towards MSB.

Eg: $(101011110)_2 = (536)_8$. (LSB, MSB)

For fractional part, grouping of 3 bits is made starting from Binary point and moving towards right

$(0.101011011)_2 = (0.533)_8$.

(Binary point)

* BINARY TO HEXADECIMAL:

Divide Binary numbers into 4 groups from LSB and moving towards MSB.

Eg: $(1\,0101\,1010\,1101)_2 = (15AD)_{16}$.

(LSB ↑ on the last group; MSB ↓ on the first group; ← moving towards MSB)

For fractional parts, grouping of 4bits is made starting from Binary point and moving towards right.

Eg: $(0.1010\,1101\,1)_2 = (0.AD8)_{16}$.

(Binary point ↑; → moving towards right)

* OCTAL TO BINARY:

Replace each octal digit by 3bit equivalent binary.

Eg: $(7\ 3\ 5)_8 = (111011101)_2$

(7 → 111, 3 → 011, 5 → 101)

For fractional part also replace each ~~ee~~ octal digit by 3bit equivalent binary.

Eg: $(753.24)_8 = (111\,101\,011.010\,100)_2$.

* HEXADECIMAL TO BINARY:

Replace each hexadecimal digit by 4bit equivalent binary.

Eg: $(AC9.87)_{16} = (101011001001.10000111)_2$

(A → 1010, C → 1100, 9 → 1001, 8 → 1000, 7 → 0111)

* OCTAL TO HEXADECIMAL:

① Octal → Binary.

② Binary → Hexadecimal.

Eg: $(734)_8 = (1\,1101\,100)_2 = (1DC)_{16}$.

* HEXADECIMAL TO OCTAL:

$(9AB8.73)_{16} = (100\,1101\,0101\,1000.0111\,0011)_2$

$= (115270.346)_8$.

* DETERMINE THE VALUE BASE:

$(16)_{10} = (100)_b \Rightarrow (16)_{10} = (b^2 \times 1 + 0 \times b^1 + 0 \times b^0)_{10}$

$b^2 = 16 \Rightarrow b = 4$.

* REPRESENTATION OF NEGATIVE NUMBERS:

(sign magnitude numbers (or) signed numbers).

unsigned numbers → 1, 2, 3

signed numbers → -15, +3, +1, -5

unsigned representation: $(0101)_2 = (5)_{10}$.

signed representation: $(0101)_2 = (+5)_{10}$ (MSB)

$(1101)_2 = (-5)_{10}$ (MSB)

MSB → 0 → +Ve signed number.

MSB → 1 → -ve signed number.

Eg: 00001001 = $(+9)_{10}$ and 10001001 = $(-9)_{10}$

MSB (+ve), magnitude. MSB (-ve)

$(+8)_{10}$ = 01000

It cannot be taken in a 4bit representation.

$(00001000)_2$ = $(+8)_{10}$.

8 bit representation

→ For signed representation, we have the range equals to,

$-(2^{n-1}-1)$ to $+(2^{n-1}-1)$. where, n → no. of bits

Eg: n = 4, $-(2^{4-1}-1)$ to $+(2^{4-1}-1)$ = -7 to 7.

* signed numbers require more hardware.

→ For unsigned representation, we have the range equals to,

0 to (2^n-1), where n → no. of bits.

Eg: n = 4, 0 to (2^4-1) = 0 to 15

→ Two ways for represented signed numbers:

① sign magnitude form.

→ requires more hardware to design a circuit.

→ to reduce this, we can use complements form.

② Complements form.

i) Radix (r) representation (or) r's complements form.

ii) Diminished radix representation (or) (r-1)'s complement form.

ii) DIMINISHED RADIX COMPLEMENT: ((r-1)'s complement)

$(r-1)$'s complement $= (r^n - r^{-m} - N)$

where, r → base, N → +ve number, n → integer parts and m → fractional parts

Eg: $(7)_{10}$

9's complement $= (10^1 - 10^0 - 7) = 10 - 7 - 1 = (2)$

(or)

$9 - 7 = 2$

$(73)_{10}$

9's complement $= (10^2 - 10^0 - 73) = 100 - 1 - 73 = 26.$

(or)

$99 - 73 = 26.$

i) RADIX REPRESENTATION: (r's complement)

r's complements $= r^n - N$

where, r → base, N → +ve number, n → integer parts ~~and~~

(or)

r's complement = (r-1)'s complement + 1 (LSD).

Eg: $(19)_{10} = 10^2 - 12 = 100 - 12 = 88.$

* 1 should be added to the least significant value

Eg: $(26)_8 = 51 + 1 = 52.$

$(1011.101)_2$

2's complement $= (0100.010) + 1 = 0100.011.$

(1's complement can be obtained by replacing 1 by 0 & vice versa)

* ARITHEMATIC OPERATIONS:

(addition) addend ↑ augent ↑

octal addition: $(533.44)_8$ to $(471.62)_8$.

Carry → 1 1

5 3 3 . 4 4

4 7 1 . 6 2

1 2 5 . 2 6

8 | 10

1 - 2

Binary addition Rules:

Addend	+ Augend.	LSB (sum)	MSB (carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$2\overline{)2}$
$1-0$

(Substraction)

Binary substraction:

① Direct substraction.

② Complement substraction.

Binary substraction rules:

minuend (A)	- substrahend (B)	difference.	Borrow.
0	0	0	0
* 0	1	1	1 *
1	0	1	0
1	1	0	0

Substract $(27)_{10}$ ~~from~~ - $(21)_{10}$ using binary.

$(27)_{10}$ = 1 1 0 1 1

$(21)_{10}$ = 1 0 1 0 1

0 0 1 1 0

r's complement substraction:

r = 10.

10's complement

A = $(53)_{10}$ = minuend , B = $(27)_{10}$ = subtrahend

10's complement of 27 = 9's complement +1 = 72+1 = $(73)_{10}$

A = 53

10's complement of B = 73

126

end carry (neglected)

∴ The result is 26.

* If minuend > subtrahend, end carry should be neglected.
  minuend < subtrahend, no end carry is obtained.

* In r's complement substraction (A>B), end carry should be discarded.
  In (r-1)'s complement substraction (A<B), end carry should be added inplace of LSB. (or) added to LSB.

* Bits → Binary digits.

  1100 → 4 bits as one group.
  nibble.

  10011100 → Bytes.
  group of 8 bits

  16 bits → word.

* Weighted code:

① Binary.

  Eg: 0 and 1.

② BCD (Binary coded decimal weighted)

  Eg: 8421, 2421, 5211, 4221.

5211 code:

0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
1011 = 5
1100 = 6
1101 = 7
1110 = 8
1111 = 9

excess 3 or XS3 = 0011

XS-3 and gray codes are weighted codes.

2421 } reflected codes.
5211 }

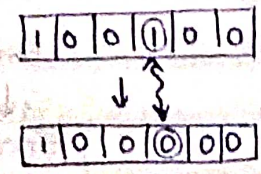
Decimal	BCD(8) Binary.	Gray code
0	0000	0000
1	0001	0001
2.	0010	0011
3	0011	0010
4.	0100	0110
5	0101	0111
6	0110	0101
7.	0111	0100
8.	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13.	1101	1011
14.	1110	1001
15.	1111	1000

Gray → 1 0 1 0
↓
Binary → 1 1 0 0.

Gray → 1 1 0 1
↓
Binary → 1 0 0 1.

→ Graycodes are also called cyclic & unit distance.

* SINGLE BIT ERROR:



parity bit & extra bit

even parity → ② (1's) even no.

Eg: 1000000

1001001 ③ odd no. (add 1 for getting even)

10011 ② add no. then add '0'.

1011011 ⑤ odd no.

(to get even parity we should add 1)

* Adding '1' & '0' are called parity generator.
* The circuit which checks is the parity check.

Eg: 101 ② even parity

3 bit data			Message with even parity		Message with odd parity	
A	B	C	Message	Parity	Message	Parity
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

→ Errors detecting and correcting code technique is developed by R.M.Hamming.

⇒ Length of Hamming code = no.of information bit + no.of parity bit

$2^P \geq n + P + 1$, where $n \rightarrow$ no.of bits in datastring.
$P \rightarrow$ parity bits.

Eg: $P = 2, n = 4$

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

$4 \geq 7$ ⇒ not satisfied

Let, $P = 3, n = 4$

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

$8 \geq 8$ ⇒ satisfied

⇒ 7 bit Hamming code

$n = 4, P = 3$

total length = 4+3 = 7.

4 are informating bits and 3 are parity bits

2^4 2^3 2^2 2^1 2^0.

P_{16} P_8 P_4 P_2 P_1 > parity bits

D_7 D_6 D_5 D_3 → Data bits

Eg: 1101

$n = 4$ (information bits).

$P = 2$

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

$4 \geq 7$ (not satisfied).

then $P = 3$

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

$8 \geq 8$ (satisfied)

↓

no. of parity bits is '3'.

Bit Designation.	D_7	D_6	D_5	D_4	D_3	D_2	D_1
Bit location.	7	6	5	4	3	2	1
Binary locations.	111	110	101	110	011	010	001
Data Bits (D_B).	1	1	0	—	1	—	—
Parity Bits (P_B).	—	—	—	0	—	1	0

Q) Determine the single error correcting code (Hamming code) for the information code 10111 for the odd parity bit.

10111

Data bits (information bits) = 5, $P = ?$, Length of data = 5+4 = 9

$P = 3$,

$2^3 \geq 5 + 3 + 1$

$8 \geq 9$ (not satisfied)

then $P = 4$,

$2^4 \geq 5 + 4 + 1$

$16 \geq 10$ (satisfied)

Bit Designation	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1
Bit location	9	8	7	6	5	4	3	2	1
Binary location no.	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bits	1	–	0	1	1	–	1	–	–
Parity bits	0	0	–	–	–	1	–	1	0

Total Hamming code = 100111110

$P_1 \rightarrow 3,5,7,9 \rightarrow 1,1,0,1 \Rightarrow P_1 = 0$

$P_2 \rightarrow 3,6,7 \rightarrow 1,1,0 \Rightarrow P_2 = 1$

$P_4 \rightarrow 5,6,7 \rightarrow 1,1,0 \Rightarrow P_4 = 1$

$P_8 \rightarrow 9 \Rightarrow 1 \Rightarrow P_8 = 0$.

Q) Determine which bit, if any error in the even parity Hamming code character 1100111. Decode the message.

length of the Hamming code = 7, $2^P \geq n+P+1$, $P=3, n=4$

Bit Designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Bit location no.	0111	0110	0101	0100	0011	0010	0001
Received code	1	1	0	0	1	1	(1)
correct hamming code	1	1	0	0	1	1	0

parity $P_1 = 1,3,5,7 = 1101 = 1$ (LSB)

$P_2 = 2,3,6,7 = 1111 = 0$ $\quad (001)_2 = (1)_{10}$

$P_4 = 4,5,6,7 = 0011 = 0$ (MSB)

∴ error is in first bit.

Q) Determine which bit, if any error in the odd parity Hamming code character 1001001. Decode the message.

$2^P \geq n+P+1$ $(P=3, n=4)$

Bit Designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Bit location no.	111	110	101	100	011	010	001
Received code	1	0	(0)	1	0	0	1

parity $P_1 = .1,3,5,7. = 1001 = 1$ (LSB)

$P_2 = .2,3,6,7 = .0001 = 0.$ $\uparrow$ $(101)_2 = (5)_{10}$.

$P_3 = 4,5,6,7 = 1001 = 1$ (MSB) ∴ error is in 5th bit.

∴ correct hamming code = 1011001.

Q) Encode the information character 01101110101 according to 1sbit even Hamming code.

$n = 11$, $P = ?$

$2^P \geq n+P+1 \Rightarrow 2^P \geq 11+P+1.$

let $P = 4$, $16 \geq 16.$

Length of Hamming code $= 11+4 = 15$

Bit designation	D_{15}	D_{14}	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
Bit location	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bit location no.	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Binary bits	0	1	1	0	1	1	1	–	0	1	0	–	1	–	–
parity bits	–	–	–	–	–	–	–	1	–	–	–	1	–	1	0

$P_1 \rightarrow .0,3,5,7,9,11,13,15 = .1001110 = 0.$

$P_2 \rightarrow 3,6,7,10,11,14,15 = .1101110 = 1$

$P_4 \rightarrow .5,6,7,12,13,14,15 = 0100110 = 1$

$P_8 \rightarrow .9,10,11,12,13,14,15 = .1110110. = 1$

∴ Hamming code = 011011110101110

~~Received even parity Hamming code as 011011001011010~~

* BOOLEAN ALGEBRA:

→ Axioms (postulates) of Boolean Algebra:

Set of logical expressions that we accepts without proof and upon which we can build a set of useful theorems.

→ NOT operator is called unary operator.

	AND operation	OR operation	NOT operation
axiom1 (postulate1)	$0 \cdot 0 = 0$	$0+0=0$	$\overline{1} = 0$ (or) $1'=0$
axiom2 (postulate2)	$0 \cdot 1 = 0$	$0+1=1$	$\overline{0} = 1$ (or) $0'=1$
axiom3 (postulate3)	$1 \cdot 0 = 0$	$1+0=1$	
axiom4 (postulate4)	$1 \cdot 1 = 1$	$1+1=1$	

* LAWS OF BOOLEAN ALGEBRA:

① complementation law: (or inversion law) (or unary operator law).

a) $\overline{0} = 1$

b) $\overline{1} = 0$

c) If $A=0$ then $\overline{A}=1$

d) If $A=1$ then $\overline{A}=0$.

e) $\overline{\overline{A}} = A$. (Double inversion or Double complementation $(\overline{\overline{A}})$).

② AND law:

a) $A \cdot 0 = 0$ (Null law).

b) $A \cdot 1 = A$. (Identity law).

c) $A \cdot A = A$

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

③ OR law:

a) $A+0 = A$ (Null law).

b) $A+1 = 1$

c) $A+A = A$.

d) $A+\overline{A} = 1$

④ Commutative law:

a) $A+B = B+A$.

b) $AB = BA$

proof:

A	B	A+B	B+A	AB	BA
0	0	0	0	0	0
0	1	1	1	0	0
1	0	1	1	0	0
1	1	1	1	1	1

⑤ Associative law:

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

b) $(AB)C = A(BC)$.

⑥ Distributive law:

a) $A(B+C) = AB + AC$.

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

* ⑦ Redundent literal rule:

a) $A + \bar{A}B = A+B$.

($A + \bar{A}B$ → ③ literals).

proof:

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

(∵ according to distributive law).

∵ WKT, $A+\bar{A} = 1$. according to OR law.

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

∵ $1.(A+B) = A+B$ according to AND law.

∴ $A+\bar{A}B = A+B$.

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

proof:

$A(\bar{A}+B) = A\bar{A} + AB$ (∵ according to distributive law)

$A(\bar{A}+B) = 0 + AB$ (∵ according to AND law)

∴ $A(\bar{A}+B) = AB$ (∵ according to OR law).

⑧ Idempotence law:

a) $A \cdot A = A$

b) $A + A = A$

⑨ Absorption law:

a) $A + AB = A$

proof:

$A + AB = A(1+B)$

$= A \cdot 1$ ($\because 1+B = 1$ according to OR law)

$\therefore A + AB = A$ ($\because$ according to AND law)

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

* THEOREMS:

consensus theorem: (or included factor theorem)

Theorem1: $AB + \bar{A}C + BC = AB + \bar{A}C$

proof:

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

$= AB + \bar{A}C + ABC + \bar{A}BC \cdot = AB(1+C) + \bar{A}C(1+B)$

($\because 1+C=1$ and $1+B=1$)

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

→ This theorem can be extended to any number of literals.

Eg: $AB + \bar{A}C + BCD = AB + \bar{A}C$

LHS $= AB + \bar{A}C + BCD = AB + \bar{A}C + BCA + BCB(0)$

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

$= AB + \bar{A}C + BC(1+D) - BC$

($\because 1+D=1$)

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

$\therefore$ LHS = RHS.

Demorgen's theorem:

① $\overline{A+B} = \bar{A} \cdot \bar{B}$

② $\overline{AB} = \bar{A} + \bar{B}$

Duality theorem:

According to duality theorem, 0 becomes 1

1 becomes 0

AND becomes OR

OR becomes AND

NOTE: Only ~~variables~~ operations complement but not variables.

① $ABCD + ABD$, reduce the Boolean expression.

$f = ABCD + ABD$ (①②③④ → 4 literals) $= ABD(C+1)$

$= ABD$ (① ② ③ → 3 literals) $(\because 1+C=1)$.

② $f = AB\bar{C} + A\bar{B}C + ABC + ABCD$

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

$= AB(\bar{C}+D) + AC$. $(\because \bar{C}+CD = \bar{C}+D$ and $\bar{B}+B=1)$

$= AB\bar{C} + ABD + AC$.

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

$= AC + AB + ABD = AC + AB(1+D)$

$= AC + AB$ $(\because 1+D=1)$

$= A(C+B)$.

③ $A + \bar{A}B + A\bar{B}$, reduce this Boolean function to min. no. of literals.

① ②② ④ → 4 literals.

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

$A(1+\bar{B}) + \bar{A}B \Rightarrow A + \bar{A}B$.

① ② → 2 literals.

$= A+B$

④ $f = \overline{(\bar{A}B)} + (\bar{A}) + AB$, reduce this expression into min. no. of literals.

$f = \overline{\bar{A}B} \cdot \bar{\bar{A}} \cdot \overline{A\bar{B}}$

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

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

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

$= 0$

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

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

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

$A \cdot A = A$.

$A \cdot \bar{A} = 0$.

⑤ $f = AB + \bar{A}C + A\bar{B}C(AB+C)$, reduce this Boolean expression into min. no. of literals.

$$f = AB + \bar{A}C + A\bar{B}C \cdot AB + A\bar{B}C \cdot C.$$

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

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

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

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

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

$$= A + \bar{A} + B + \bar{C} = 1 + B + \bar{C}$$

$$= 1.$$

$\because B \cdot \bar{B} = 0$
$A \cdot A = A$
$C \cdot C = C$
$A + \bar{A}B = A + B$
$B + \bar{B}C = B + C$
$\bar{A}C = \bar{A} + \bar{C}$
$1 + A = 1$
$1 + B + \bar{C} = 1$

⑥ Determine Dual of a given Boolean function.

$f = A\bar{B} + \bar{A} \cdot \bar{B}$
(AND, OR, AND)

$f = (A+B) \cdot (\bar{A} + \bar{B})$
(OR, AND, OR)

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

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

$f = AB + \bar{A}\bar{B}$ its dual of f as $A\bar{B} + \bar{A}B$

⑦ $f = A\bar{B} + \bar{A}B$, determine the dual of this function.

$f = A\bar{B} + \bar{A}B.$
(AND, OR, AND.)

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

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

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

$B \cdot \bar{B} = 0$

$A \cdot A = A$

$C \cdot C = C$

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

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

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

$1 + A = 1$

$1 + B + \bar{C} = 1$

* COMPLEMENT OF A FUNCTION:

① Take dual.

② determine complement : from step ① complement each literal.

→ $f = AB + \bar{A}\bar{B}$, determine complement of f.

① $f = (A+B)(\bar{A}+\bar{B})$.

② $\bar{f} = (\bar{A}+\bar{B})(A+B)$

$f = \overline{AB + \bar{A}\bar{B}} = \overline{AB} \cdot \overline{\bar{A}\bar{B}}$

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

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

LOGIC GATES:

* BINARY LOGIC:

→ Two distinct values in binary logic → 0 and 1

* BASIC LOGIC GATES:

① NOT Gate.

→ A single input A and an output $\bar{A}$.

A	$\bar{A}$
0	1
1	0.

② AND Gate.

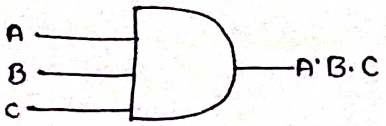
→ For two inputs, the output will be 1 if both inputs are 1, it will be 0.

→ Denoted as A.B

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

3 i/p AND gate.

2^3 possible combinations.



A	B	C	ABC
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
0	1	1	0
1	0	1	0
1	1	0	0
1	1	1	1

③ OR Gate.

→ For two inputs, the output will be 1 if atleast one of the input is 1 and will be 0 otherwise.

→ Denoted by A+B

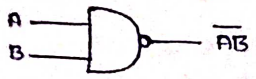
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

* UNIVERSAL GATES:

① NAND Gate.

→ For two inputs, the output will be 1 if atleast one of the inputs are at 0, will be 0 otherwise.

→ Denoted by $(AB)'$.



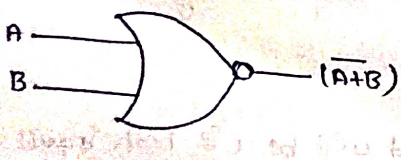
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NOTE: NAND gate is equivalent to bubble i/p OR gate.

② NOR Gate.

→ For two inputs, the output will be 1 if both the inputs are at 0, will be 0 otherwise.

→ Denoted by $(A+B)'$

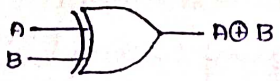


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

* Exclusive OR (EXOR) Gate:

→ For two inputs, the output will be 1 if odd no. of inputs are at 1, will be 0 otherwise.

→ Denoted by $A \oplus B$



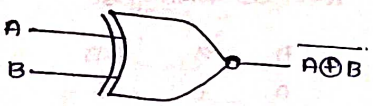
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

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

* Exclusive NOR (EXNOR) Gate: (Equivalence Gate).

→ For two inputs, the output will be 1 if even no. of inputs are at 1, will be 0 otherwise.

→ Denoted by $(A \oplus B)'$



A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

$\overline{A \oplus B} = A \odot B = \bar{A}\bar{B} + AB$

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

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

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

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

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

$= \bar{A}\bar{B} + AB = A \odot B = \overline{A \oplus B} = (A \oplus B)'$

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

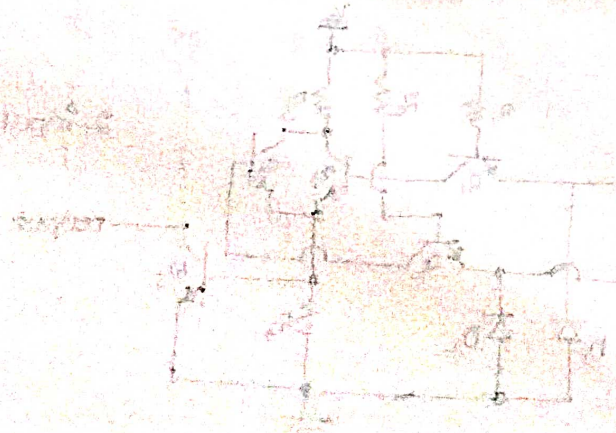
$\overline{AB} = \bar{A} + \bar{B}$

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

$\bar{\bar{B}} = B$

$B \cdot \bar{B} = 0$

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



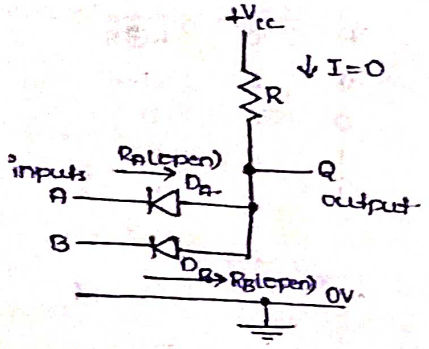
* How to construct these Gates?

- Various logic families exist:
  - Diode transistor logic (DTL)
  - Transistor logic gate (TTL)
  - Emitter coupled logic (ECL)
  - Complementary Metal Oxide Semiconductor (CMOS) logic

- CMOS is almost universally used today.

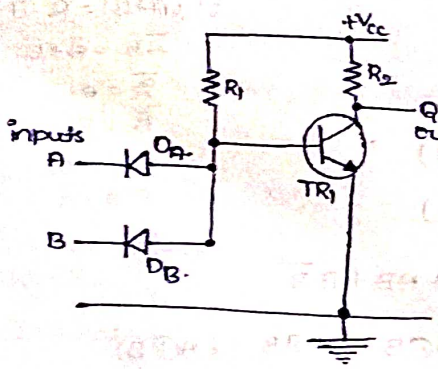
① DIODE TRANSISTOR LOGIC:

→ uses semiconductor diodes and bipolar transistors, along with resistances.



2-input AND gate

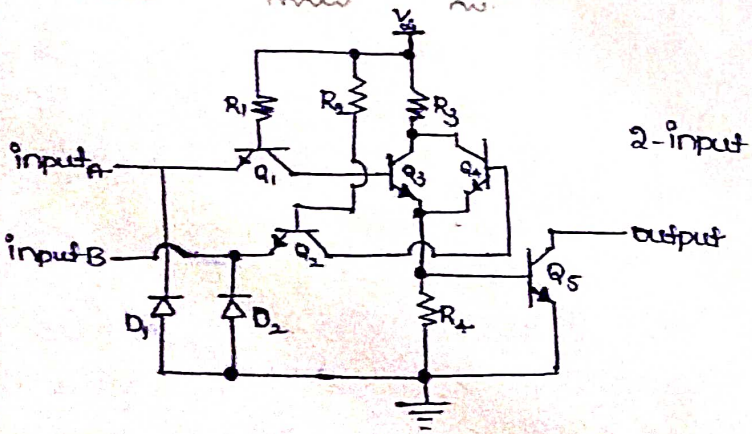
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1



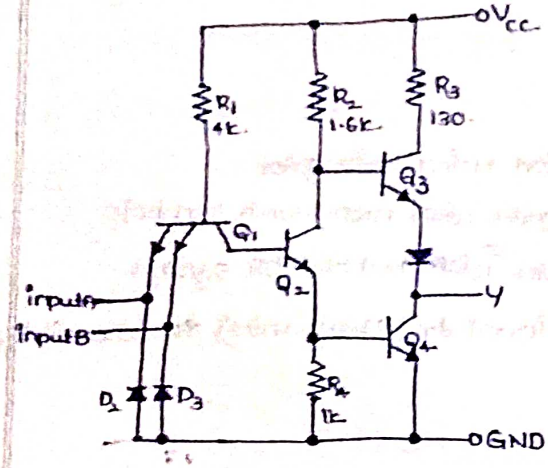
2-input NAND gate

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

② TRANSISTOR TRANSISTOR LOGIC:



2-input NOR gate.

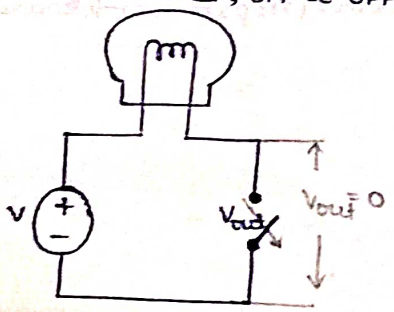


2-input NAND gate.

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

* BASIC CONCEPTS OF SWITCH BASED CIRCUITS:

They rely on the operation of tiny switches, which can be in 1 of 2 states - open or closed, ON or OFF, voltage or no voltage etc.

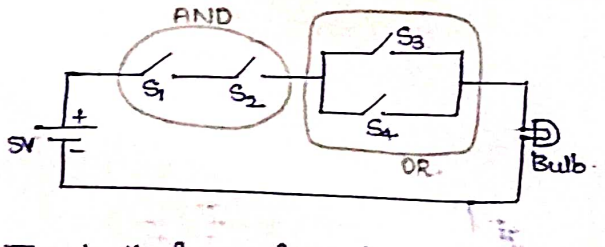


Switch open:
→ No current flows.
→ Light is OFF.

Switch closed:
→ Current flows.
→ Light is ON.

Examples:

⇒ A control circuit for an electric bulb.



The bulb is switched on if the switches S1 and S2 are closed and S3 and S4 is also closed, otherwise the bulb will not be switched ON

i-e,	S_1	S_2	S_3	S_4
	1	1	1	0
	1	1	0	1
	1	1	1	1

* NAND Realization:

1. Implements given function using basic gates.
2. Converts AND to NAND gates with AND inverts symbol.
3. Converts OR to NAND gates with bubbled OR symbol.
4. Where ever received an inverter (Step 2 and 3) followed by inverter use another inverter.

* NOR Realization:

1. Implements given function using basic gates.
2. Converts AND to NOR gates with bubbled inputs AND symbol.
3. Converts OR to NOR gates with OR followed by inverts symbol.
4. Where ever received an inverter (Step 2 and 3) followed by inverter use another inverter.