Number Representation.

Basic Number system types.

Base/Radix.

$N_b = d_{n-1} \ldots . d_2 d_1 d_0 , d_{-1} d_{-2} \ldots d_{-m}$

Number — Most Significant Digit

Decimal to Binary conversion.

For integer Part		For Fractional part.		Final Converted
Continuous division by 2 keeping the track on remainder Go from bottom to top.	+	continuous multiplication by 2 keeping the track of integer part Go from to left to right	=	Binary number merge both (integer & fractional Part) Binary BITS.

Convert given $(42.65625)_{10}$ into BINARY

Integer part 42.

2	42	
2	21	0
2	10	1
2	5	0
2	2	1
2	1	0
	0	1

TOP ↑ Bottom

$(42)_{10} = (101010)_2$

Fractional Part 65625

0.65625 *2 → 1.31250 → 1

0.31250 *2 → 0.6250 → 0

0.6250 *2 → 1.250 → 1

0.250 *2 → 0.5 → 0

0.5 *2 → 1.0 → 1

Left → Right

$.65625 = (.10101)_2$.

Binary To Decimal Conversion.

For integer part of binary number use positive power of two.

For fractional part of binary number use negative power of two.

Add both number to get equivalent decimal number.

Binary to decimal conversion.

eg. Convert the following no into decimal

$(1011.01)_2$

Positive power of 2 — negative power of 2

$= (2^3 \cdot 1) + (2^2 + 0) + (2^1 + 1) + (2^0 + 1) + (2^{-1} + 0) + (2^{-2} + 1)$

$= 11 + 0.25$

$= (11.25)_{10}$.

Decimal to Octal.

eg. $(67.517)_{10}$.

8	67	2
8	8	3
8	1	0
	0	1

0.517	0.136	0.088	0.704
x 8	x 8	x 8	x 8

~~Octal~~

~~6~~

Octal to decimal.

$(670.17)_8$.

$(6 \times 8^2) + (7 \times 8^1) + (0 \times 8^0) + (1 \times 8^{-1}) + (7 \times 8^{-2})$

$= (440.234375)_{10}$.

Number System Table.

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

Octal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
	8
	9

Octal	Hexadecimal
	A
	B
	C
	D
	E
	F.

Binary To Octal.

Eg.

$$(10\ 101\ 111.\ 010\ 011)_2$$

$$(2\ \ 5\ \ 7.\ \ 2\ \ 6)_8.$$

Octal to ~~Hexadecimal~~ Binary

$$(723.17)_8.$$

$$(111\ \ 010\ 011.\ 001\ 111)_2$$

$$(1\ 03.3\ 0)_{16}.$$

$$(0001\ \ 1101\ \ 0011\ \ 1100\ \ 1101)_2$$

$$(723.632)_{16}.$$

~~Octal~~ ~~to~~ Hexadecimal to Binary

Deimal to Binary.

17.

2	17	1
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

10001

Fractional deimal no to Binary:

$(0.65625)_{10} \rightarrow (0.10101)_2$

0.65625	0.31250	0.62500	0.25000
x 2	x 2	x 2	x 2
1.31250	0.62500	1.25000	0.50000
1	0	1	0

0.50000

x 2

1.00000

1

Octal to Binary.

$(736)_8 = (111\ 011\ 110)_2$

Octal	Binary	Octal	Binary
0	000	3	011
1	001	4	100
2	010	5	101

Octal	Binary
6	110
7	111
10	001 000
23	010 011

Binary to Octal.

$(1001110)_2 = (001\ 001\ 110)$

$= (116)_8.$

$(11.11)_2 = (011.110)_2$

$= (3.6)_8.$

Hexadecimal to Decimal Conversion

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

$3\times16^{1} + 10\times16^{0} + 2\times16^{-1} + 15\times16^{-2}$

$= 48 + 10 + \frac{2}{16} + \frac{15}{16^2}$

$= (58.1836)_{10}.$

Hexadecimal to Binary.

$(2F9A)_{16} = (0010\ 1111\ 1001\ 1010)_2$

$= (0010111110011010)_2$

$(1A.B1)_{16} = (0001\ 1010 . 1011\ 0001)_2$.

Binary to Hexadecimal

$(0111111001.1001\ 1001)_2$

0001 1111 0001 . 1001 1001

↓ ↓ ↓ ↓ ↓

1 F 1 9 9

$= (1F1.99)_{16}$.

Decimal to Hexadecimal.

a) Integer part

	Quotient	Remainder
95/16	5	15(F) ↑
5/16	0	5

$(95)_{10} = (5F)_{16}$

b) Fractional part $0.5 \times 16 = 8.0$ ↓

$(0.5)_{10} = (0.8)_{16}$

$\therefore (95.5)_{10} = {}^{8}(5F.8)_{16}$

Hex to Octal.

$(A72E)_{16}$

1010 0111 0010 1110 $)_2$

001

$(1\ 2\ 3\ 4\ 5\ 6)_8$.

Octal to Hex

$(247.36)_8$.

010 100 111 . 011 110 $)_2$

1000

$(A\ 7\ .\ 7\ 8)_{16}$.

weighted code.
eg BCD.
1001.
$2^3 0 + 0 2^0 \rightarrow 8 + 1 \rightarrow (9)$

Non-weighted code.
eg Excess 5, grey.
1001 $\rightarrow$ (9)

weighted codes.
→ The main characteristic of a weighed code is each binary bit is assigned by weight and values depend on the position of the binary bit.

→ The sum of the weights of these binary bits, whose value is 1 is equal to the decimal digit which they represent.

→ A sequence of binary bits which represents a decimal digit is called a "code word"

→ Example of these codes is
BCD, 8421, 6421, 4221, 5211, 3321, etc

Application.
Data manipulation during arithemetic operations for input/output operations in digital circuits Digital Displays like in calculators, digital volt meter etc.

Codes → when ~~Numbers~~ letters, or text represented by group of symbol.

- Weighted Codes
  - Binary Coded Decimal codes (BCD)
- Non-weighted Codes
  - eg. Grey Codes
  - Excess-3 Codes
- Alphanumeric Codes
  - Eg ASCII codes
- Error detecting Codes
  - Eg Parity code
- Error detecting & Correcting codes
  - Hamming codes

Binary coded Decimal (BCD).

Binary codes Decimal (BCD) or 8421 codes or 4 bit code or 1 Nibble code
Each digit is represented by 4 bits of binary code.

eg 3: 0011, 8: 1000.

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

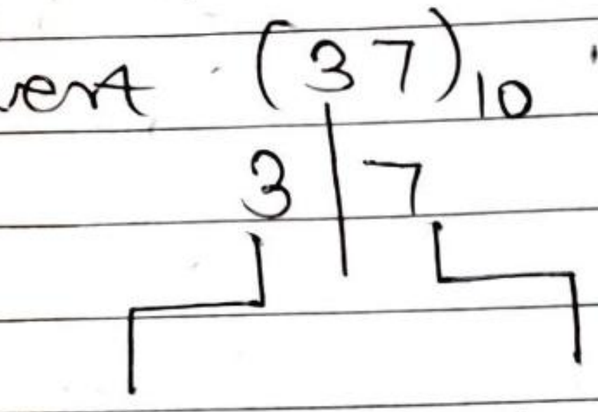
Conversion steps.
Identify nos of digits to be converted in given BCD number.
↓
Split all the digits
↓
Represent each digit with fo bits of binary number.
↓
Combine all the bits.

Examples.

convert $(6)_{10}$ into BCD code.

$(6)_{10} \longrightarrow (0110)_{BCD}$.

convert $(37)_{10}$ into BCD code.



$(37)_{10} = (00110111)_{BCD}$.

BCD to Decimal Conversion.

Step 1 : Start dubbing the group of 4 binary bits together from RHS to LHS (for Integer no).

Step 2 : Represent each group of 4 binary bits into its equivalent decimal number.

Step 3 : If number of binary bits are not multiple of 4 then pad zero to LHS side.

Convert given BCD code into Binary.

$(0001\ 0101)_{BCD}$.

↓

(1 5).

Pad zero LSB

$(00101110)_{BCD}$ → (1 6)

Convert $(17)_{10}$ into BCD.

1 | 7

$(0001\ 0111)_{BCD}$

0001 0111.

Convert $(64.38)_{10}$ into BCD.

6 | 4 . 3 | 8.

↓ ↓ ↓ ↓

$(0110\ 0110 . 0011\ 1000)_{BCD}$.

Non-weighted codes:

→ Non-weighted or Unweighted codes are these codes in which the digit value does not depend upon their position, i.e each digit position within the number is not assigned fixed value.

→ positional weightage is not assigned.

eg. Excess-3 codes., Grey Codes.

Application.

- To perform certain arithmetic operations.
- shift position encodes.
- used for error detecting purpose.

Excess - 3 codes.

- The excess-3 code is a non-weighted & self-complementary BCD code used to represent the decimal numbers
- only weighted code which is self-complementary

Advantages.

- this code plays an important role in arithmetic operation because if resolves deficiences encounted when we use the 8421 BC code for adding two decimal digits whose sum is greater than 9.

Decimal to excess-3 conclusion.

Conversion steps.

Decimal	BCD (8421)	Excess-3 code.
0	0000	0011
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.

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

⇒ Convert $(6)_{10}$ into Excess - 3 code.

$(6)_{10} \longrightarrow (1001)$ excess 3.

$$\begin{array}{r} 0\ 1\ 1\ 0 \\ +\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 1 \end{array}$$

⇒ convert $(35)_{10}$ into Excess 3 code

$(3 \quad 5)$

$$\begin{array}{r} 0\ 0\ 1\ 1 \\ +\ 0\ 0\ 1\ 1 \\ \hline 0\ 1\ 1\ 0 \end{array} \qquad \begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

Convert given $(57)_{10}$ into Excess-3 code.

$(5 \quad 7) \Rightarrow (8 \quad 10)$

$(1000 \quad 1010)$ Excess - 3

Gray codes / reflection.

Gray code is a binary numeral system where two successive values differ only are bit. This code was invented by Frank Gray in 1953.

Why Gray codes?

- Two successive values differ only in 1 bit.
- Binary number is converted into Gray code is reduced switching operation.
- Gray codes are also known as
  ① Unit Distance code.
  ② Minimum Error code.
  ③ Reflection code.

How to construct Gray code.

1 bit Gray code.

	G_0
0 →	0
1 →	1

2 bit.

	G_1	G_0
0	0	0
1	0	1
2	1	1
3	1	0

3 bit.

G_2	G_1	G_0
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

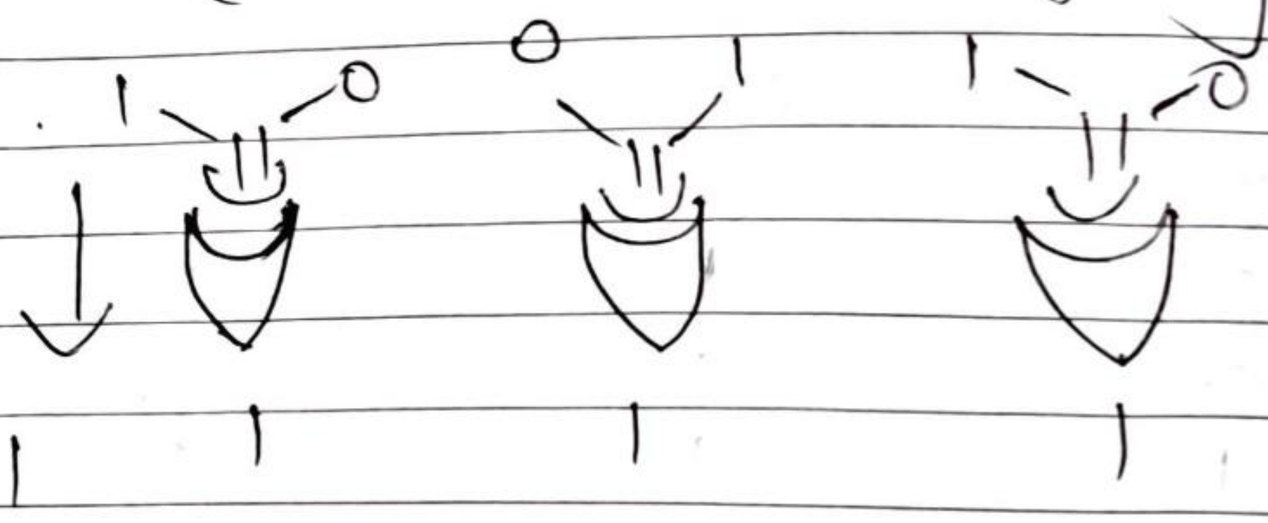
4 bit.

G_3	G_2	G_1	G_0
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

Binary to Gray code Conversion.

Q. Convert (1010) into gray code.

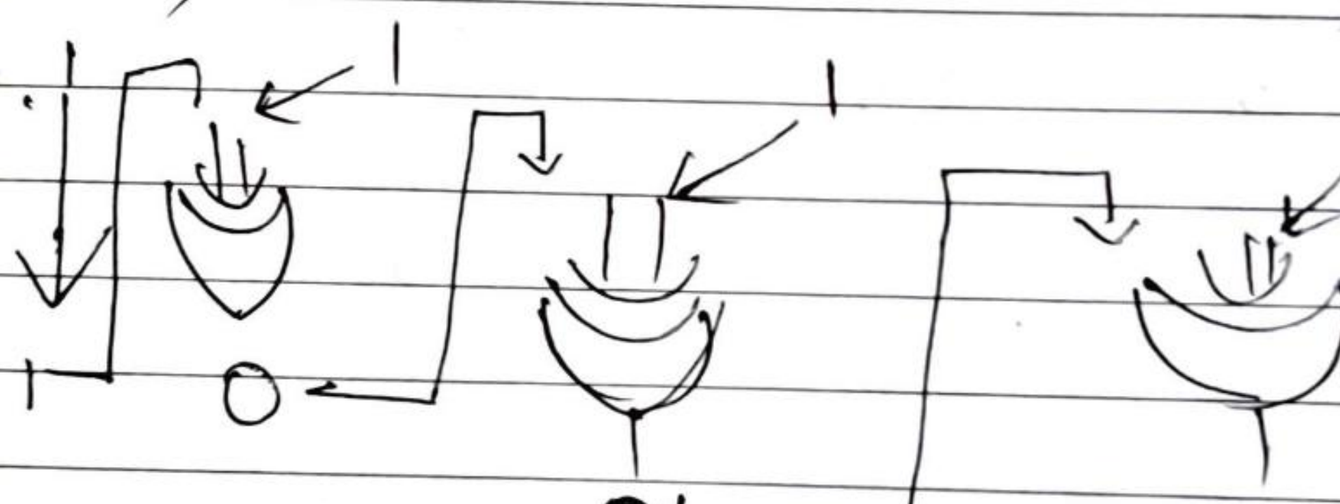
Binary. 1 1 1 1



Gray

convert Given Gray code 1111 into binary

Gray →



binary.

Ans ⟹ (1010).

Convert into Gray code.

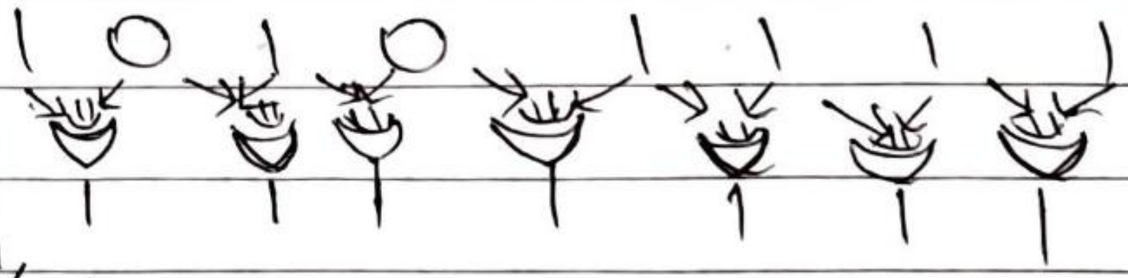
$(34)_{10} \longrightarrow (0100010)_2$.

$(1100011)_{GC}$.

2	34	
2	17	0
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

2. $(1010\ 1111)_2$

Binary → 1 0 1 0 1 1 1 1



Gray → 1 1 1 1 1 0 0 0

Gray code 11111000.

Convert gray code to binary equivalent

1) $(10001)_2$

Gray code 1 0 0 0 1.

Binary → 1 1 1 1 0.

↳ $(11110)_2$.

2) $(110001)_2$.

Gray code → 110001

Binary → 100001.

↳ $(100001)_2$.

ASCII code.

1. American standard code for Information Interchange.

2. The ASCII code is used to give to each symbol / key from keyboard a unique number called ASCII code.

3. It can be used to convert text into ASCII code and then into binary code.

4. It can be used within your code to identify specific characters in a string or specific keys bring protocol on the keyboard.

ASCII is a complete code which uses 128 different encoding combinations of a group of seven bits $(2 = 128)$.

Error Detecting codes.

Parity code → It is a bit that is included with the binary data to be transmitted to detect the error.

Types

1)

2)

eg1) Binary message bits :- 1011110, transmit with Even parity.

(1) 1 0 1 1 1 1 0.

↓

Parity bit.

Binary message bits :- 1011110, transmit with Odd parity.

(0) 1 0 1 1 1 1 0.

↓

Parity bit.

At receiver's end :-

case 1: Message received correctly.

(1) 1 0 1 1 1 1 0.

case 2: Message received with 1 bit error.

(1) 1 0 1 0 1 1 0. (error)

⇒ Drawback.
If error occurs with multiple of 2 bits it will not be detected.

Q. How Parity bit will help?

→ At Receiver's end.

Parity check.

(1) 1 0 1 0 1 1 0

No. of 1's : 5 (Odd parity)

received message is wrong
Hence proved parity bit is error detecting code.

Q. Generate even parity & odd parity for following binary message bits (1011001

even parity → (0) 1 0 1 1 0 0 1
odd parity → (1) 1 0 1 1 0 0 1.

Binary signed & unsigned number Representation.

1) SIGN magnitude.
2) 1's Complement.
3) 2's Complement.

SIGN magnitude.
- Add 1 bit at MSB.
- As per sign of the number.

+ → 0
− → 1

1's Complement.

Convert 1 to 0 & 0 to 1

eg 0110 → +6

1001 → −6.

2's Complement.

```
  0 1 1 0  ← +6
  1 0 0 1  ← −6
+       1 1
-------------
  1 0 1 0
```

No.	Sign magnitude	1's Complement	2's Complement
3	011	011	011
2	010	010	010
1	001	001	001
0	000	000	000
0	100	111	000
-1	101	110	111
-2	110	101	110
-3	111	100	101.

3 bit → (−4 to +3)

4 bit → (−8 to +7)

5 bit → (−16 to +15)

formula :- 2^n to 2^{n-1}.

Examples.

(Based on addition / subtraction Rules)

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

2)
```
  1 1 1 0
- 0 0 1 0
---------
  1 1 0 0
```

3)
```
  1 1 0 0
+ 1 1 1 1
---------
1 1 0 1 1
```

4)
```
  1 0 0 0 0
- 0 1 1 1
---------
  0 0 0 1
```

Binary subtraction Rules

A	B	Result	Borrow.
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Binary Addition Rules.

A	B	Result	Borrow
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Subtraction using 1's complement:

$(7)_{10} - (5)_{10}$.

$7 \longrightarrow (0111)_2$

$5 \longrightarrow (0101)_2$.

$\hookrightarrow (-5)$

$\hookrightarrow (1010)_2$

1's complement

$$\begin{array}{r} 0111 \\ +\ 1010 \\ \hline \text{Carry}\ (1\,0001)_2 \\ \hline (0010)_2 \end{array}$$

Carry is generated and is positive add back carry.

$(5)_{10} - (7)_{10}$.

$5 \longrightarrow (0101)_2$

$7 \longrightarrow (0111)_2$.

$\hookrightarrow (-7)$

$\hookrightarrow (1000)_2$.

1's complement.

$$\begin{array}{rr} & (0101)_2 \\ + & (1000)_2 \\ \hline \text{carry} & (1101)_2 \\ \hline \end{array}$$

carry not generated ans is in 1's complement form.

carry not generated ans is in 1's complement form.

1's complement of ans.

$(1101)_2 \longrightarrow (0010)_2$.

Rules for subtraction 1's complement.

After addition if carry is generated then add back carry and answer is positive.

After addition if carry is not generated then answer is negative and its in 1's complement form.

Subtraction using 2's complement form

$(5)_{10} - (7)_{10}$.

$5 \rightarrow (0101)_2$

$7 \rightarrow (0111)_2$

$\hookrightarrow (-7)$

$\hookrightarrow (1000)_2$

$$\begin{array}{rr} & 1000 \\ + & 1 \\ \hline & (1001)_2 \\ \hline \end{array}$$

2's complement.

$$(0\ 1\ 0\ 1)_2$$
$$+\ (1\ 0\ 0\ 1)_2$$
$$(1\ 1\ 1\ 0)_2$$

Carry not generated ans in 2's Complementary

Carry not generated ans is in form's complement form.

2's complement of ans.

$(1110)_2 \longrightarrow (0001)_2$

$+ \quad 1$

$(0010)_2$.

$(7)_{10} - (5)_{10}$.

$7 \longrightarrow (0111)_2$.

$5 \longrightarrow (0101)_2$

(-5)

$(1010)_2$

$+ \quad 1$

2's complement. $\longrightarrow (1011)_2$

ADD nos.

$(0111)_2$

$(1011)_2$

Carry $(1\ 0\ 0\ 1\ 0)_2$

Carry is generated ans is positive discard carry.

↳ Discard carry

Rules for subtraction 2's complement

After addition if carry is generated then discard carry and answer is positive.

After addition if carry is not generated then answer is negative and its in 2's complement form.

Exercise problem.

→ $(10)_{10} \rightarrow (3)_{10}$ using 1's and 2's complement.

10 → 01010 → 01010.
3 → 00011 → 11100

1 00110

01010
+ 11101

1 00111

$(00111)_2$.

→ $(3)_{10} \rightarrow (10)_{10}$ using 1's and 2's complement.

3 → (00011)$_2$
10 → (01010)$_2$

00011
+ 10101

11001

1's → $(10101)_2$

BINARY MUITIPLICATION.

1011.011 * 1011.01.

7 X 4 ⟶

```
    0 1 1 1
    0 1 0 0
  ---------
    0 0 0 0
  ---------
  0 0 0 0 0
+ 0 1 1 1 0 0
  ---------
  (0 0 1 1 1 0 0)2
```

BCD Addition.

RULES.

1. SUM < 9, final carry = 0 ⟶ ANSWER IS CORRECT

2. SUM < 9, final carry = 1 ⟶ Tak correction Measure ADD 0110.

3. SUM > 9, final carry = 0 ⟶ Tak correction measure ADD 0110.

Case 1.

$(2)_{10} \rightarrow (6)_{10}$.

2 ⟶ (0010) BCD.
6 ⟶ (0110) BCD.

ANSIS CORRECT

```
  0010   BCD
  0110
 ------   BCD
  1000   BCD.
```

ANSWER < 9.
Carry not generated

BCD Addition case 2.

$(8)_{10} + (9)_{10}$.

$8 \longrightarrow (1000)_{BCD}$.

$9 \longrightarrow (1001)_{BCD}$.

$(1000)_{BCD}$.

$+ (1001)_{BCD}$

$\boxed{1}\ 0001$

BCD.

Answer is Invalid.

Answer < 9 but carry is generated

Take corrective measure ADD 0110.

$(1\ 0\ 0\ 0\ 1)_{BCD}$

$(0\ 1\ 1\ 0)_{BCD}$.

$(0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{BCD}$.

$(1\ \ 7)_{10}$.

BCD. Addition Case 3.

$(3)_{10} - (7)_{10}$.

$3 \longrightarrow (0011)_{BCD}$.

$7 \longrightarrow (0111)_{BCD}$.

$$\begin{array}{r} (0011)_{BCD} \\ + (0111)_{BCD} \\ \hline (1010)_{BCD} \end{array}$$

Answer is Invalid.

Carry not generated but ans > 9.

Take corrective measure Add 0110.

$$\begin{array}{r} (1010) \\ + (0110)_{BCD} \\ \hline (0001\ 0000)_{BCD} \end{array}$$

$(1\ 0)_{10}$

Examples.

$(57)_{10} + (26)_{10}$.

$(77)_{10} + (77)_{10}$.

$$\begin{array}{r} 83 \\ \hline \end{array}$$

$(57)_{10}$ 0101 0111

$(26)_{10}$ 0010 0110

$$\begin{array}{r} 0111\ \ 1101 \\ + \quad\ \ 0110 \\ \hline \end{array}$$

1000 0011

8 3

$(77)_{10} + (77)_{10}$

77 → 0111 0111

+ 0111 0111

1110 1110

+

0110 0110

0001 0101 0100

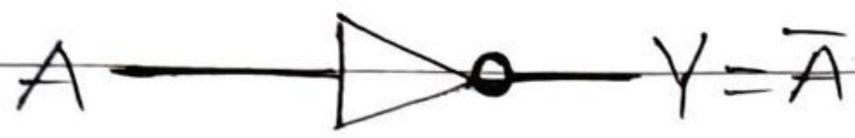
↓ ↓ ↓

1 5 4

154.

Basic Logic Gates.

NOT Gate.



NOT Gate.

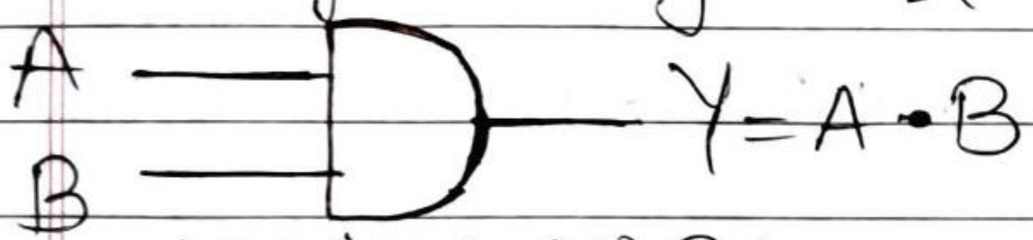
Truth Table.

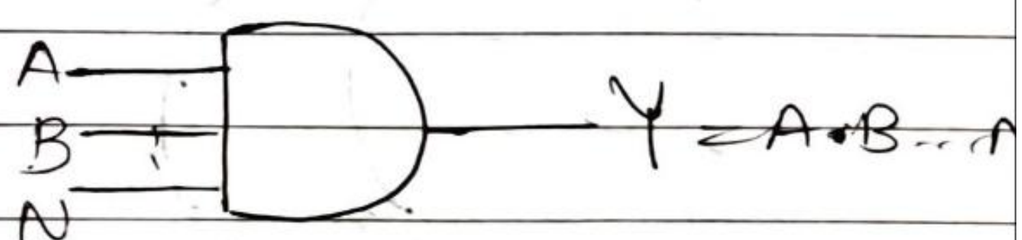
A	Y
0	1
1	0.

Logic symbol.

AND Gate.

logical symbol.





logical AND.

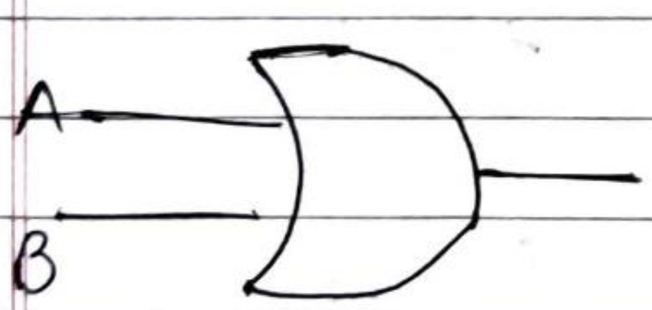
Truth table.

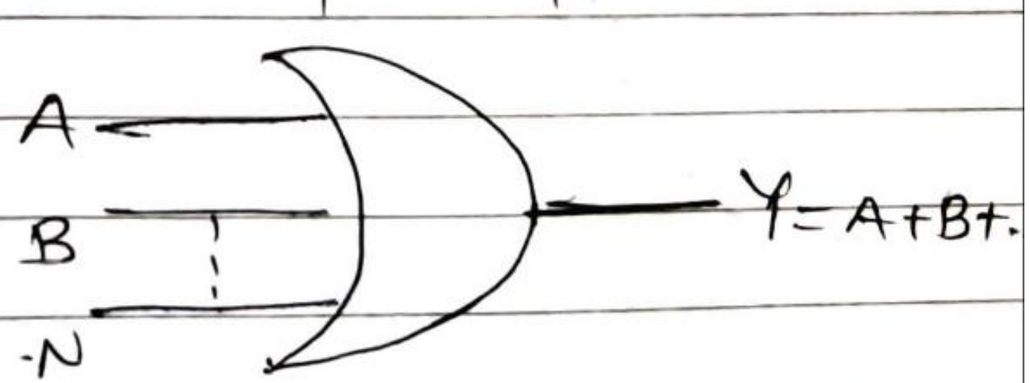
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth table.

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

OR Gate.



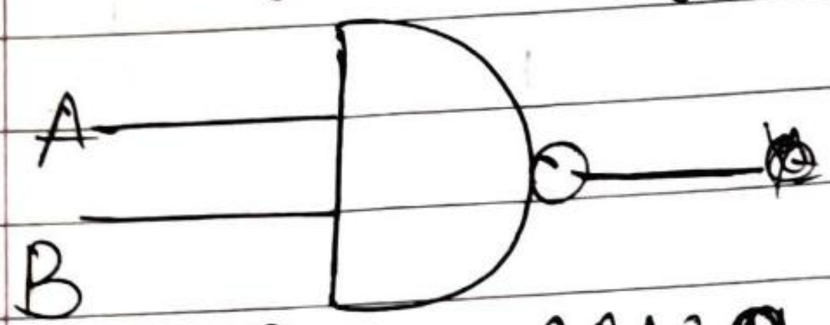


logical symbol.

Universal Gates.

NAND Gate.

logical symbol.

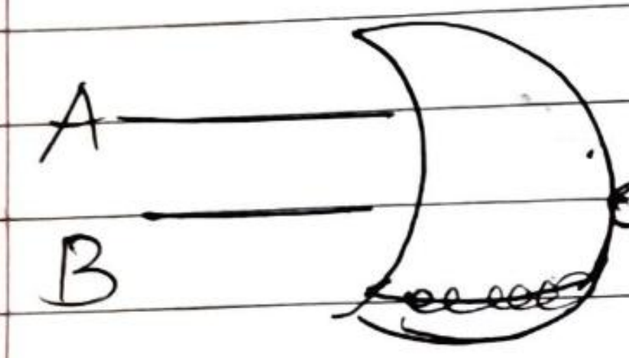


Logical NAND

Truth table.

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

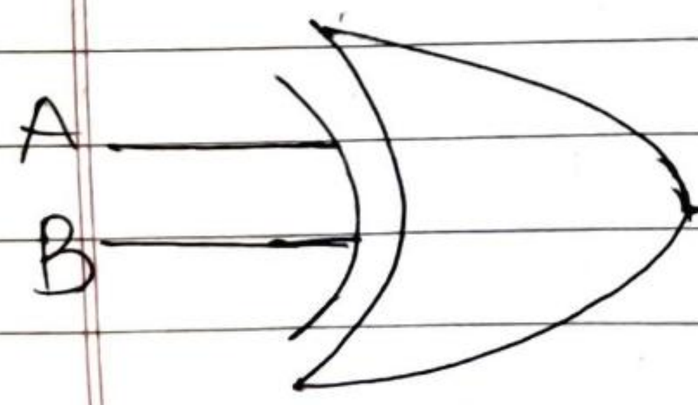
$$Y = \overline{A \cdot B}$$



$$Y = \overline{A + B}$$ ... NOR

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

EX-OR.



logic Symbol

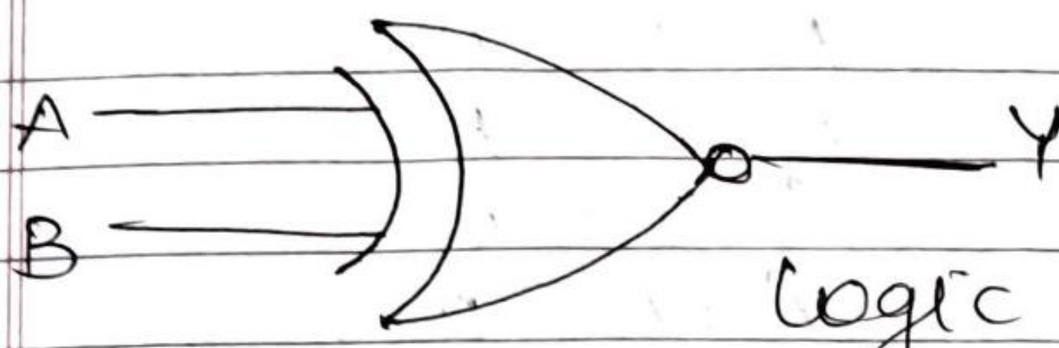
$$Y = \bar{A}B + A\bar{B}$$

Truth table.

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

Ex-NOR Gate.

Logic symbol.



Logic symbol.

Truth Table.

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

Boolean expression for logic gates

AND	$A \cdot B$
OR	$A + B$
NOT	$\overline{A}$
NAND	$\overline{A \cdot B}$
NOR	$\overline{A + B}$
EXOR	$(A \cdot \overline{B}) + (\overline{A} \cdot B)$ or $A \oplus B$
Ex-NOR	$(A \cdot B) + (\overline{A} \cdot \overline{B})$ or $\overline{A \oplus B}$

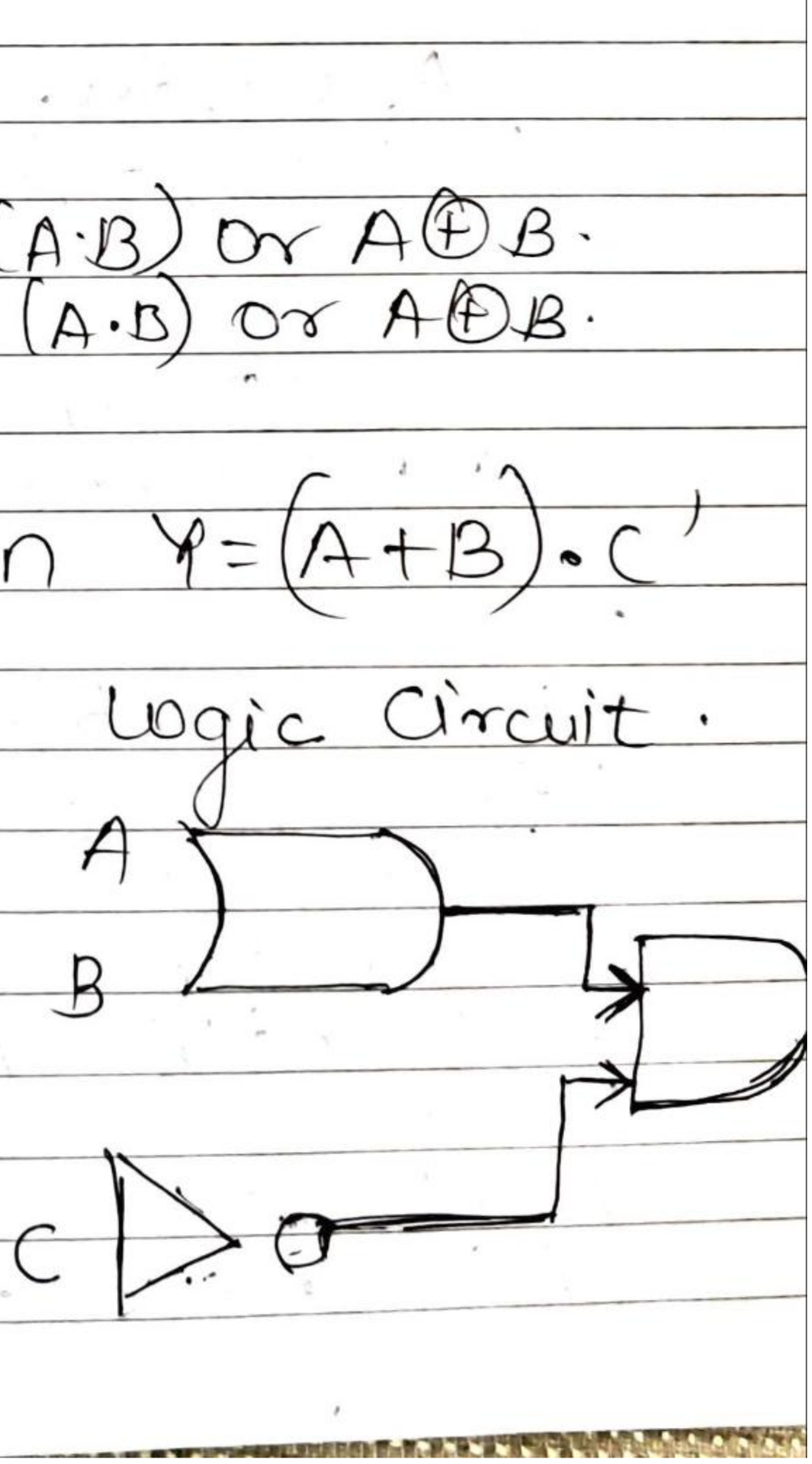
Cycle.

Boolean expression $Y = (A + B) \cdot C'$

Truth table.

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

Logic Circuit.



Boolean laws.

AND law.

$A \cdot A = A$

$A \cdot 0 = 0$

$A \cdot 1 = 1$

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

OR law.

$A + A = A$.

$A + 0 = A$.

$A + 1 = 1$

$A + \bar{A} = 1$.

NOT law.

$\bar{1} = 0$.

$\bar{0} = 1$.

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

commulative law.

$A \cdot B = B \cdot A$

$A + B = B + A$.

Associative law.

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

Distributive law.

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

Other laws.

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

$A + A'B = A + B$.

$A(A' + B) = AB$.

$AB + AB' = A$

$(A + B)(A + B') = A$.

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

$A + AB = A$

$(A + B)(A' + C) = AC + A'B$.

D. E Morgan's theorem.

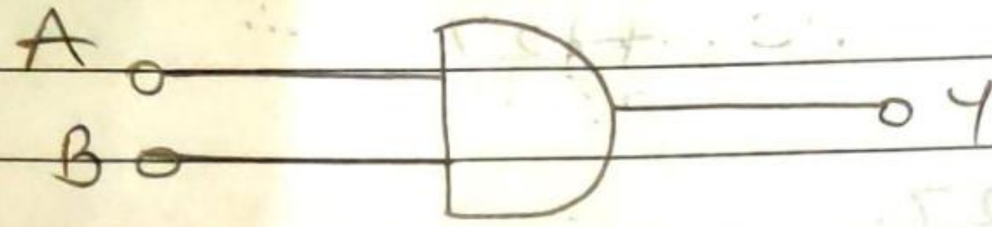
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

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

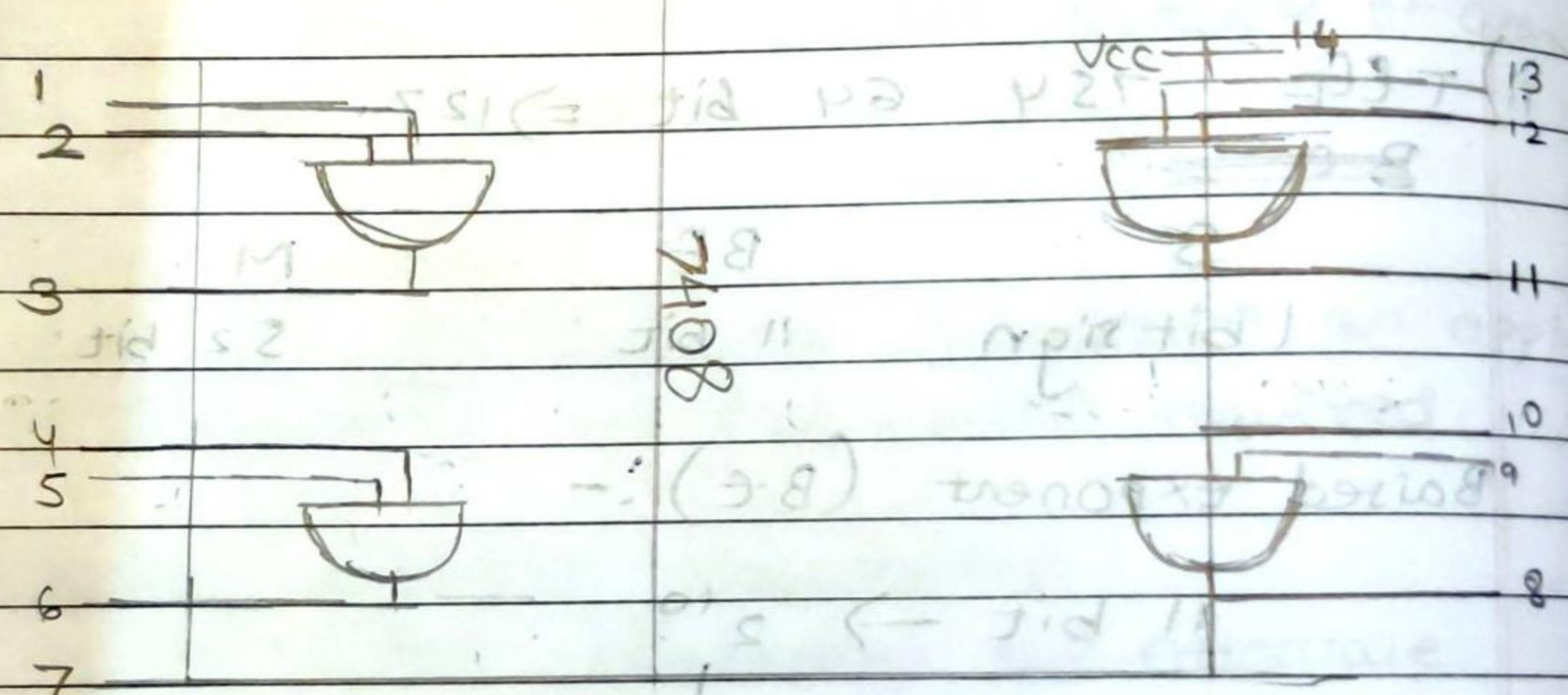
A	B	$\overline{A}$	$\overline{B}$	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$	$\overline{A + B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1	1	1	1
0	1	1	0	1	1	0	0
1	0	0	1	1	1	0	0
1	1	0	0	0	0	0	0

PIN DIAGRAM.

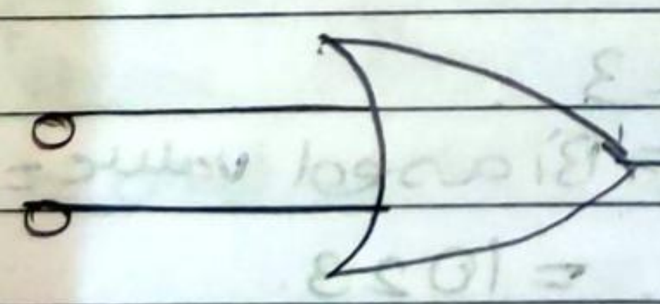
1) AND GATE



INTERNAL PIN schematic of I.C

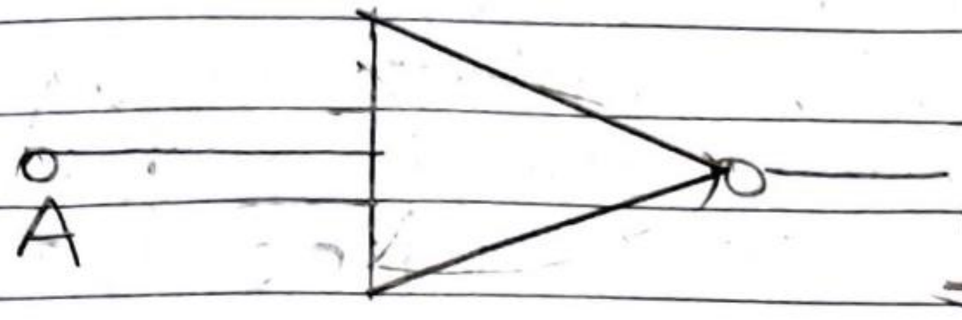


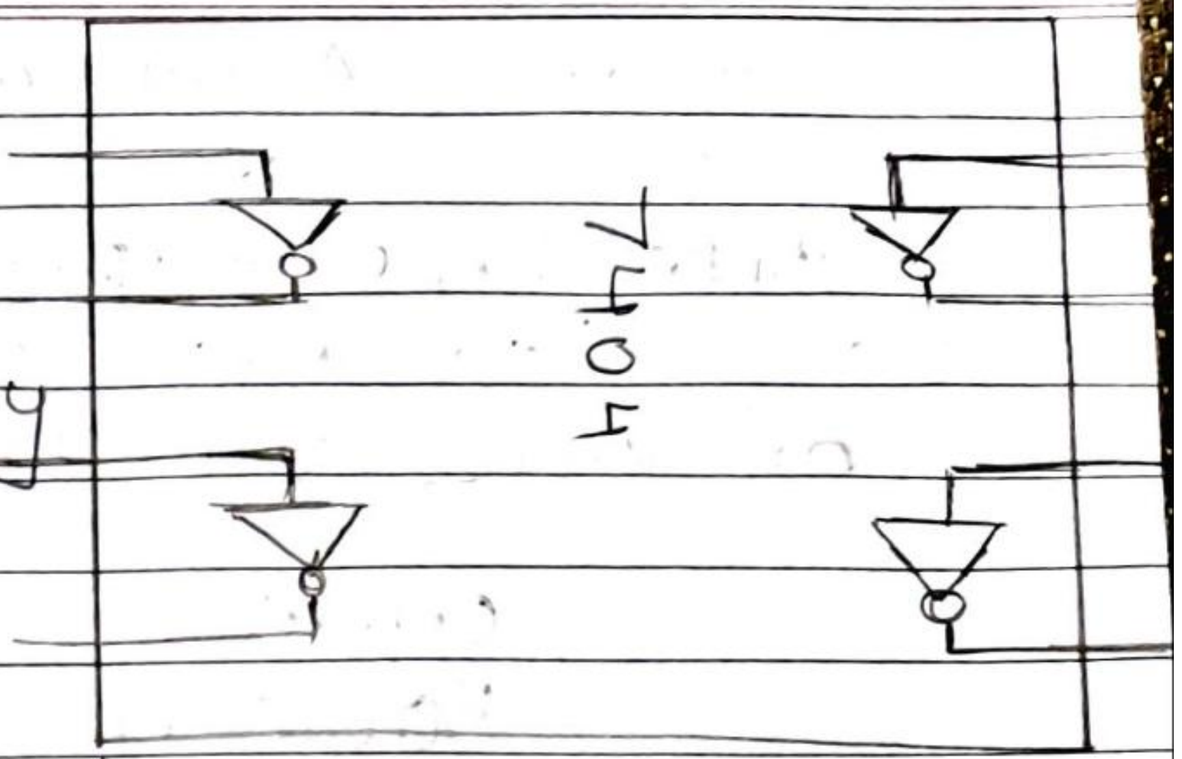
2) OR GATE



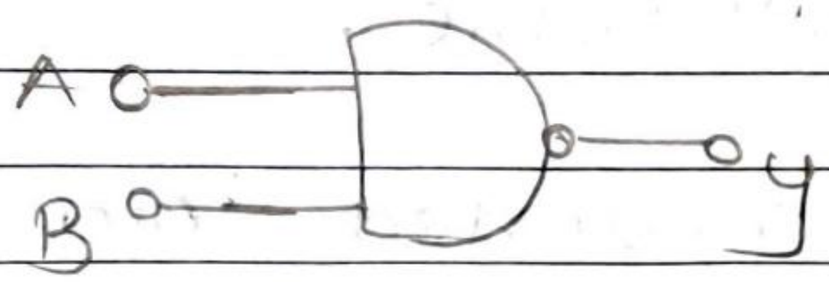
INTERNAL PIN schematic of PIN.

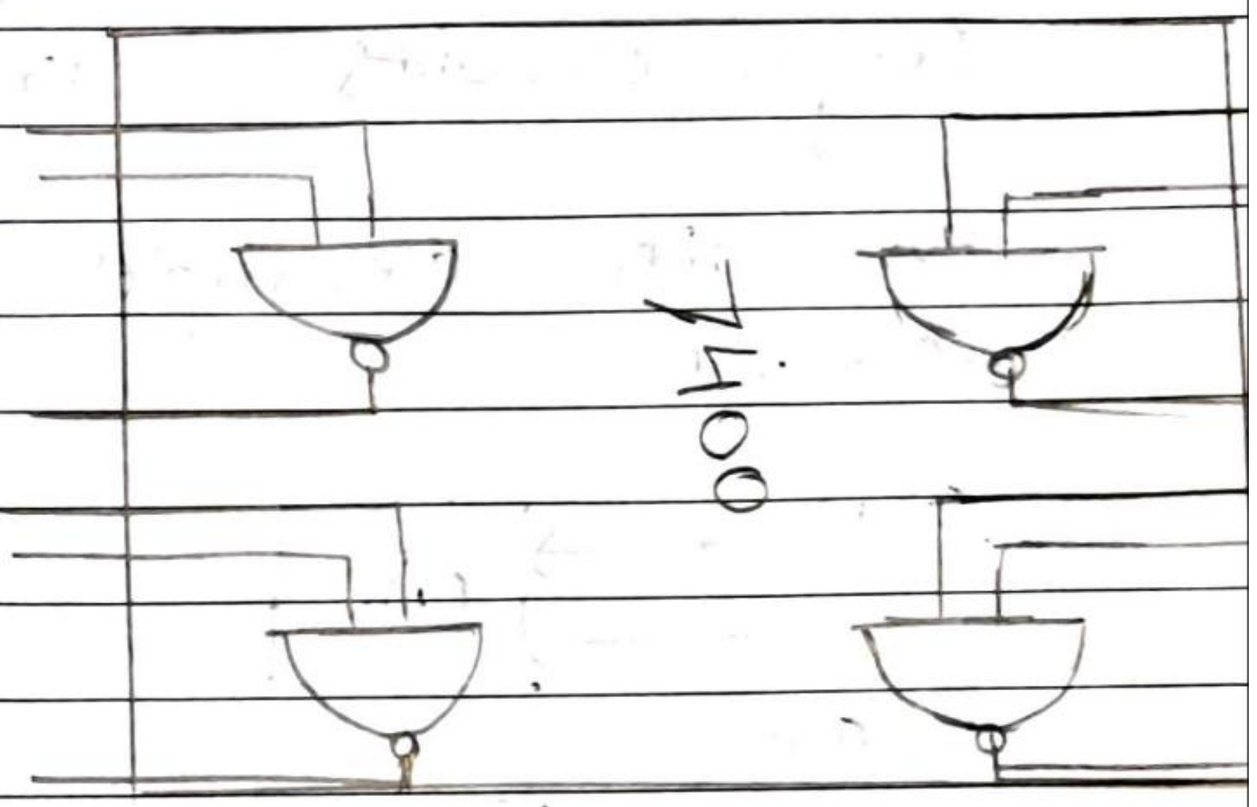
3) NOT



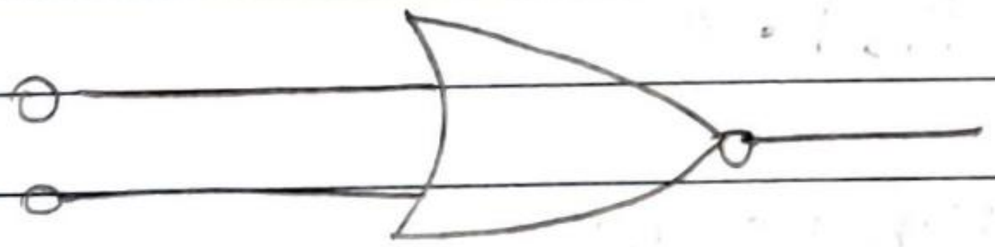


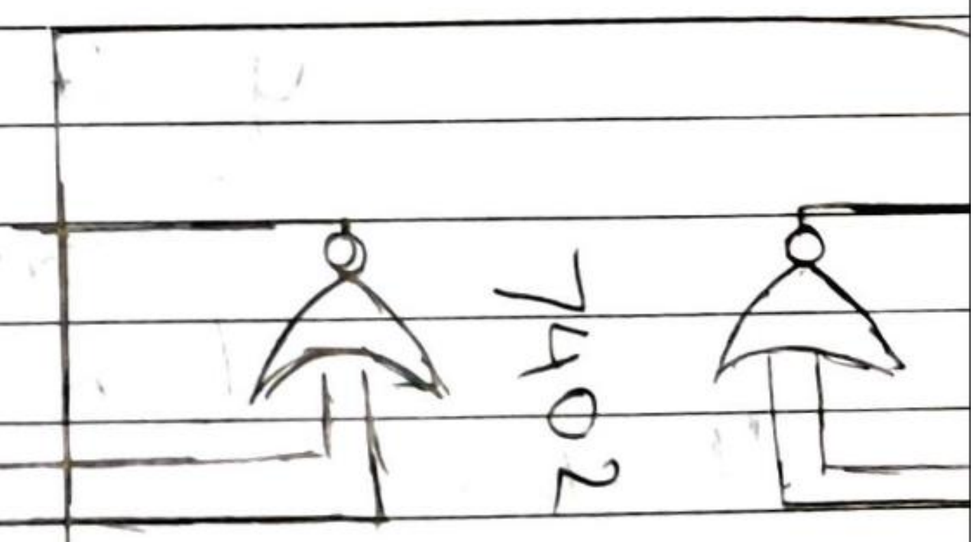
4) NAND





5) NOR





Boolean Algebra.

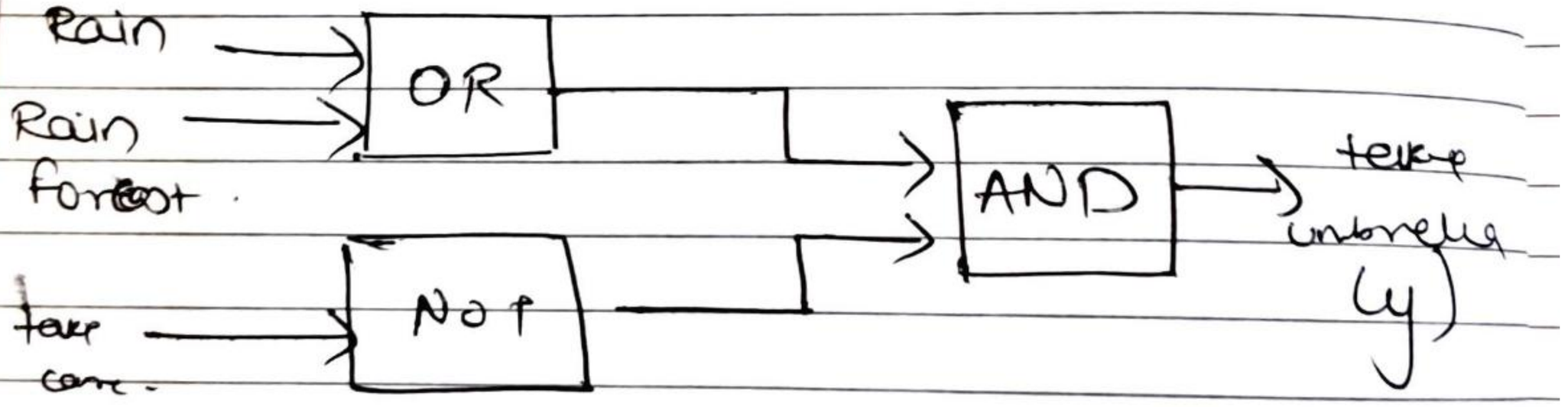
→ It is a set of rules used to simplify given logic expression, without changing function.

→ It is used when ~~changing~~ number of variables are less (like 1, 2, 3)

cycle
Boolean
Expression.

Logic circuits ⟷ Truth table.

Real life example converted into logic circuit.



$$Y = (A + B) \cdot C'$$

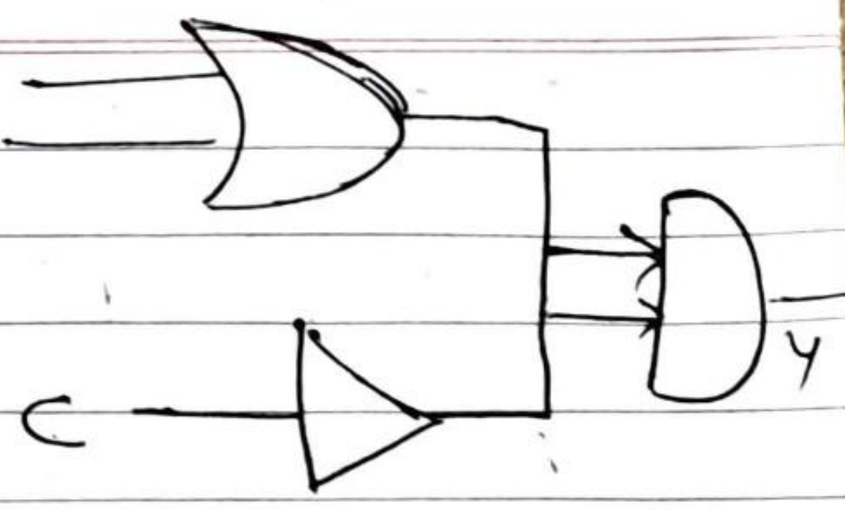
$Y = (A+B) \cdot C'$

Truth table

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

Truth table

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



Reduction for an expression.

And 1 And 2 And 3

$y = AB + BC + ABC$.

OR Gate 1.

And 1 And 2

$y = A'B + BC$

OR Gate 1

Introduction to Reduction techniques.

1. using Boolean law.
2. K-Maps.
3. Quine Mc-Cluskey.

Boolean laws.

AND Law	OR Law	NOT Law	Commutative law.
$A \cdot A = A$	$A + A = A$	$\overline{1} = 0$	$A \cdot B = B \cdot A$
$A \cdot 0 = 0$	$A + 0 = A$	$\overline{0} = 1$	$A + B = B + A$
$A \cdot 1 = A$	$A + 1 = 1$	$\overline{\overline{A}} = A$	
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$		

Distributive law.

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

$A + A'B = A+B$.

$A(A'+B) = AB$.

$AB + AB' = A$.

$A + AB = A$.

Associative law.

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

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

$(A+B)(A+B') = A$

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

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

DE Morgan theorem.

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

1	2	3	4	5	6	7	8
A	B	$\overline{A}$	$\overline{B}$	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1	1	1	1
0	1	1	0	1	1	0	0
1	0	0	1	1	1	0	0
1	1	0	0	0	0	0	0

Reduce given expression.

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

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

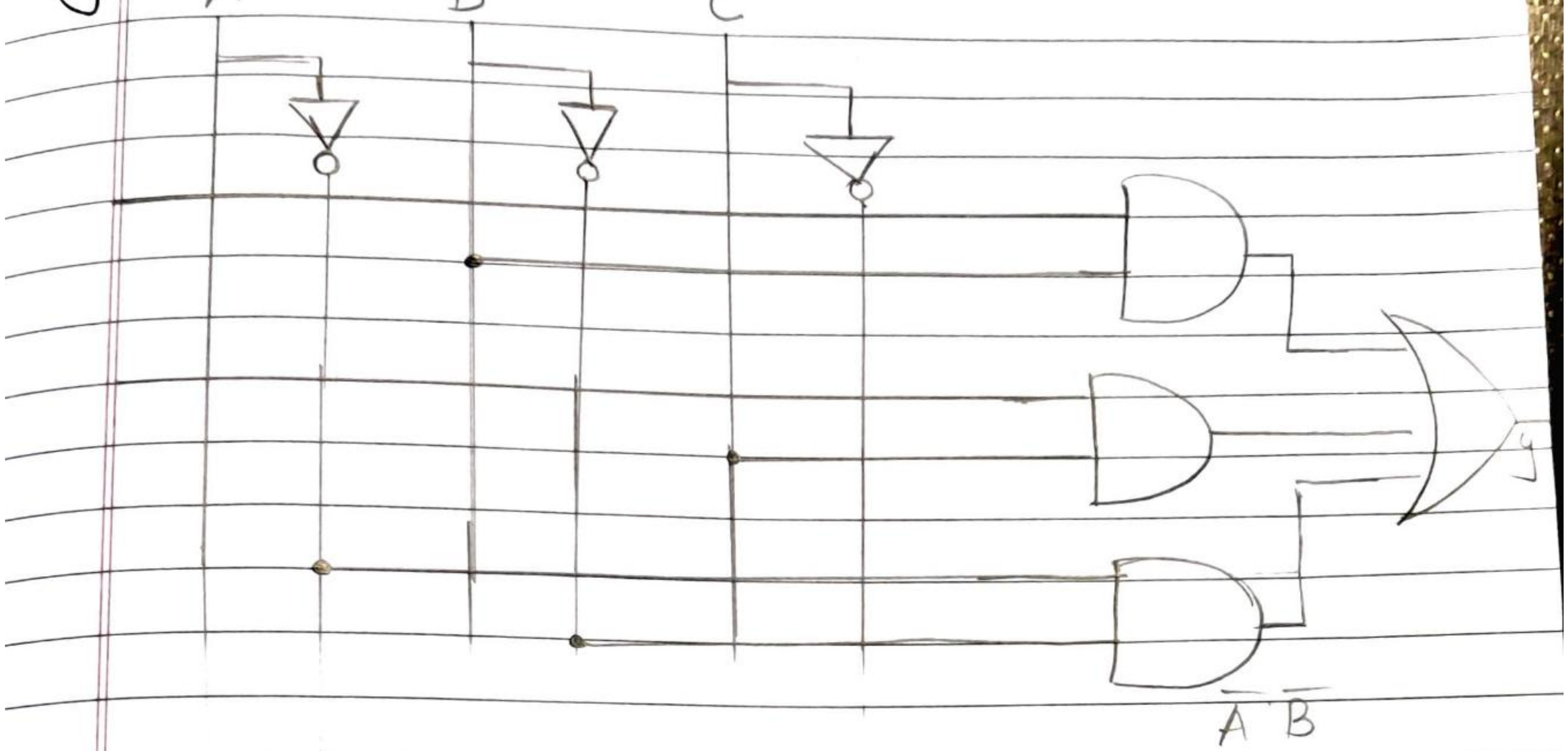
$= AC + \bar{A}B + AB\bar{C}$

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

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

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

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



Simple problem solving.

$$A\left[B + C\left[\overline{AB} + \overline{AC}\right]\right]$$

$$= A\left[B + C\left[\overline{\overline{AB}} \cdot \overline{AC}\right]\right]$$

~~$= A\left[B + C\left(\overline{A} + \overline{B}\right)\right]$~~

$$= A\left[B + C\left[\bar{A} + B\right] \cdot \left[\bar{A} + \bar{C}\right]\right]$$

$$= A\left[B + C\left[\bar{A}\cdot\bar{A} + \bar{A}\cdot\bar{C} + B\cdot A + B\,\bar{C}\right]\right]$$

$$= A\left[B + \bar{A}C + \bar{A}\cdot\bar{C}\cdot C + B\cdot AC + B\,\bar{C}\cdot C\right]$$

$$= A\left[B + \bar{A}C + 0 + \bar{B}\cdot\bar{A}C + 0\right]$$

$$= AB + 0 + 0 + 0$$

$$= AB.$$