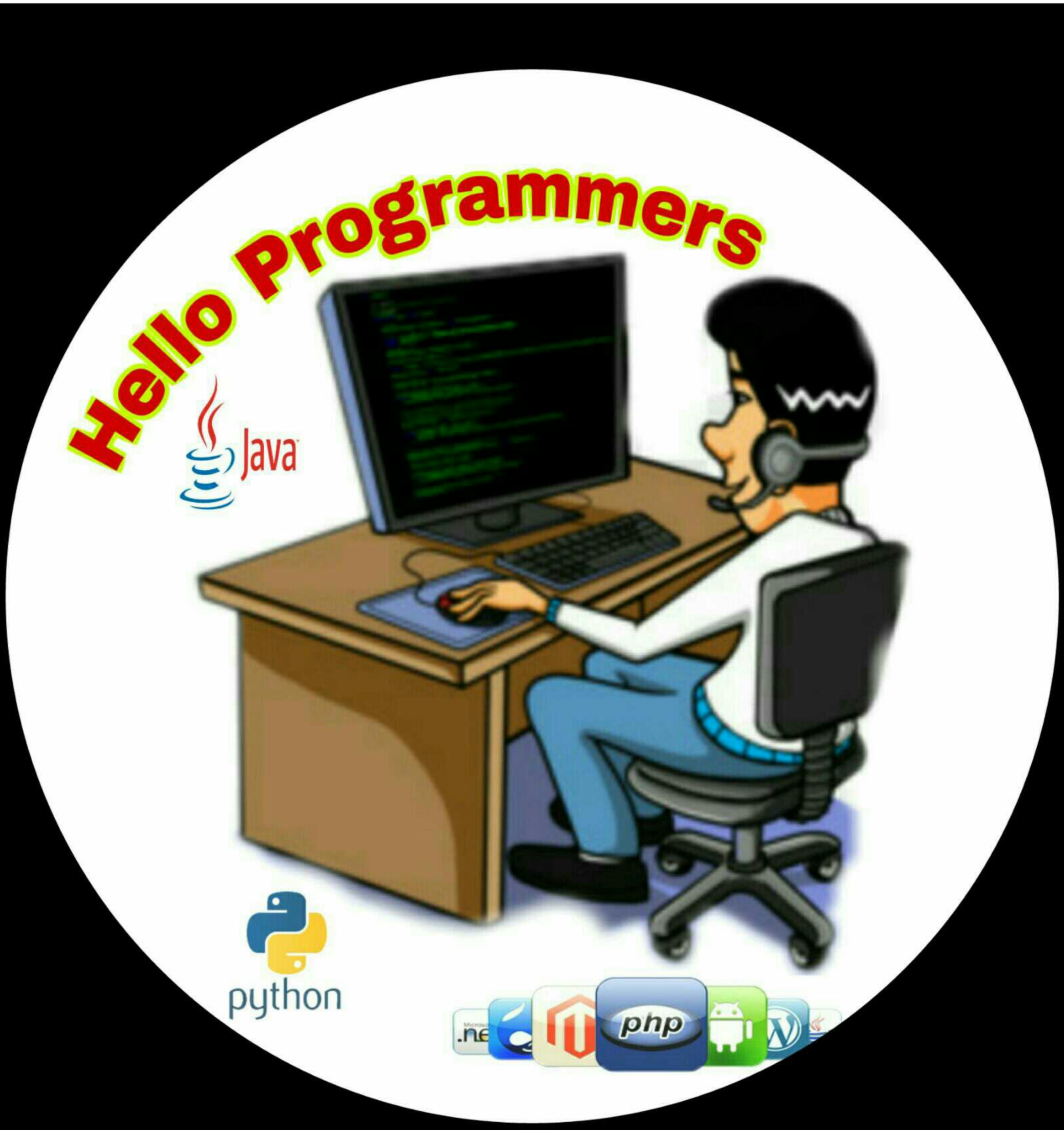


Scanned by TapScanner

Scanned by TapScanner

1

manipulation → skillfull use
simulation → design

Digital Logic

Chapter = 1

BINARY SYSTEMS

1.1 Digital computers and digital systems

Digital computers have advanced many scientific, industrial and commercial sectors. The most striking property of a digital computer is its generality. It can follow a sequence of instructions called a program, that operates on given data.

The general-purpose digital computer is the best known example of digital system. Other examples include digital ~~voltm~~ voltmeters, frequency counters, calculating machines etc.

Characteristic of a digital system is its manipulation of discrete elements of information. Such discrete elements maybe electric impulses, arithmetic operators, an alphabet or any other set of meaningful symbols.

An analog computer performs a direct simulation of a physical system. The term analog means continious signals because "analog computer" has come to mean a computer that manipulates continious variables.

Scanned by TapScanner

* **Analog** → Continious input and output occurs.

* **Digital** → Output may or may not be.

* **Logic** → Technique / Idea to solve problems.

* **Flow chart** → It is the graphical representation of algorithms. (Imp)

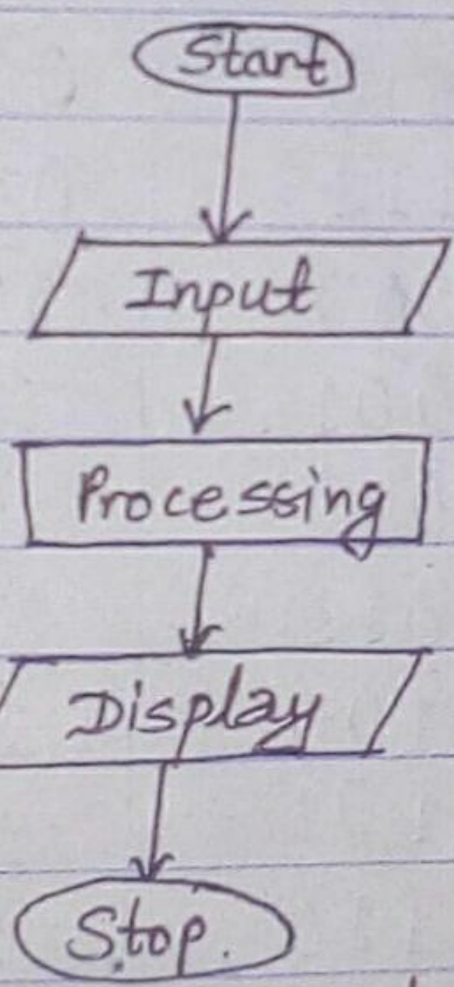
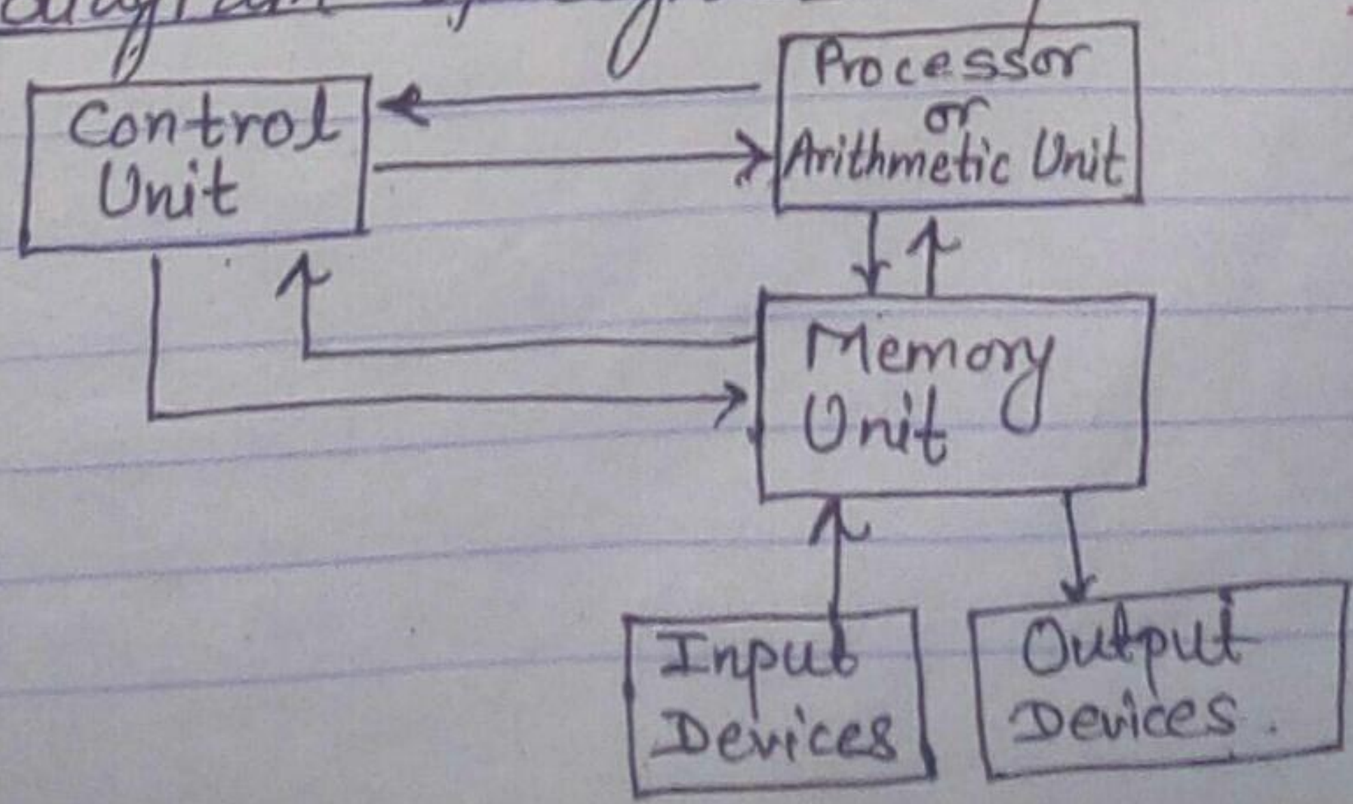


fig. flow chart.

* **Block diagram of digital computer:-** (Imp)



1.2 Binary Numbers :-

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal Base 16.
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Scanned by TapScanner

1.3 Number base conversions:-

i) Binary to decimal
ii) Decimal to binary
iii) Binary to octal
iv) Decimal to hexadecimal etc.

1.4. Octal and hexadecimal numbers

1.5 COMPLEMENTS (Imp)

Complements

(or, 2's complement) r's complement — (r-1)'s complement (or, 1's complement)

i) For binary → 2's and 1's complement
ii) For decimal → 10's and 9's complement.

(a) # The r's complement — n is the number of integer part.

$\boxed{r^n - N}$ for $N \neq 0$.

r is the base — given number (second one)

For e.g.

i) The 10's complement of $(52520)_{10}$ is $10^5 - 52520$
$= 47480$.

ii) The 10's complement of $(0.3267)_{10}$ is $1 - 0.3267$
$= 0.6733$

∵ No integer part, so, $10^n = 10^0 = 1$.

iii) The 10's complement of $(25.639)_{10}$ is $10^2 - 25.639$
$= 74.361$.

iv) The 2's complement of $(101100)_2$ is $(2^6)_{10} - (101100)_2$
$= (1000000 - 101100)_2$
$= 010100$

v) The 2's complement of $(0.0110)_2$ is $(1-0.0110)_2$
$= 0.1010$.

(b) # The (r-1)'s complement :-

$$\boxed{r^n - r^{-m} - N}$$

r = base
n = power
m = ~~deci~~ after decimal part / fraction part.
N = whole given number
OR
Full

For e.g

i) The 9's complement of $(52520)_{10}$ is $(10^5 - 1 - 52520)$
$= 99999 - 52520$
$= 47479$

∵ No fraction part, so $10^{-m} = 10^0 = 1$.

ii) The 9's complement of $(0.3267)_{10}$ is $(1 - 10^{-4} - 0.3267)$
$= 0.9999 - 0.3267$
$= 0.6732$.

∵ No integer part, so $10^n = 10^0 = 1$.

iii) The 9's complement of $(25.639)_{10}$ is $(10^2 - 10^{-3} - 25.639)$
$= 99.999 - 25.639$
$= 74.360$

iv) The 1's complement of $(101100)_2$ is $(2^6 - 1) - (101100)$
$= (111111 - 101100)_2$
$= 010011$

v) The 1's complement of $(0.0110)_2$ is $(1 - 2^{-4})_{10} - (0.0110)_2$
$= (0.1111 - 0.0110)_2 = 0.1001$

Scanned by TapScanner

10's complement → for decimal
2's complement → for binary } r's complement

Subtraction with r's complement:-

For e.g.

1) Using 10's complement, subtract A=72532 and B=3250

Solⁿ.

A=72532
B=03250

10's complement of B = 10^5 − 03250 ($\because r^n - N$)
= 96,750.

Now, adding this with A we get,

```
   72532
 + 96750
 -------
1 69282
```

end carry → 1

Hence discarding end carry ~~10's complement~~ we get 69282 as answer.

2) Subtract using 10's complement: $(3250-72532)_{10}$

Solⁿ

Given,

M=03250
N=72532

10's complement of N = 10^5 − 72532
= 27,468

Now adding this with M we get,

```
  03250
+ 27468
 ------
  30718 (No end carry)
```

No end carry occur so, answer = −(10's complement of 30718)
= −(10^5 − 30718)
= −69282.

Navneet

Date
Page

i) Using 2's complement subtract:

$M = 1010100$

$N = 1000100$

Soln

Given, $M = 1010100$

$N = 1000100$

2's complement of $N = 2^7 - 1000100$ (in binary) then subtract

$= 0111100$

Now, adding this with M.

$$\begin{array}{r} 1010100 \\ +\ 0111100 \\ \hline \text{end carry} \rightarrow \boxed{1}0010000 \end{array}$$

~~Ne~~ Discarding end carry we get answer as 10000

ii) Using 2's complement subtract.

$M = 1000100$

$N = 1010100$

Soln.

Given,

$M = 1000100$

$N = 1010100$

2's complement of $N = 2^7 - 1010100$ (In binary)

$= 0101100$

Now, adding this with M.

$$\begin{array}{r} 0101100 \\ +\ 1000100 \\ \hline 1110000 \end{array}$$ (No end carry)

$\therefore$ Answer $= -$(2's complement of 1110000)

$= -10000$

9's complement → for decimal } (r-1)'s complement
1's complement → for binary

D.# Subtraction with (r-1)'s complement:-

For eg. • For Decimal numbers

i) Use 9's complement to subtract M = 72532 and N = 03250.

Soln.

M = 72532

N = 03250

9's complement of $N = 10^5 - 10^0 - 03250$ ($\because r^n - r^{-m} - N$).

$= 100000 - 1 - 03250$

$= 96749$.

Now, adding this with M.

$$\begin{array}{r} 72532 \\ +\ 96749 \\ \hline \boxed{1}\ 69281 \\ +\ 1 \\ \hline 69282 \end{array}$$

end carry

Answer = 69282.

ii) Use 9's complement to subtract M = 03250 and N = 72532.

Soln 9's complement of $N = 10^5 - 10^0 - 72532$

$= 27467$

Now, adding this with M

$$\begin{array}{r} 03250 \\ +\ 27467 \\ \hline 30717 \end{array}$$ (no end carry).

Answer = −(9's complement of 30717)

$= -69282$ $[\because -(10^5 - 10^0 - 30717)]$

Navneet
Date
Page

Example for binary numbers :-

a) Use 1's complement to subtract $M = 1010100$
$N = 1000100$

Soln

Given, $M = 1010100$
$N = 1000100$

1's complement of $N = 2^7 - 2^0 - 1000100$ (In binary, In binary, then subtract)
$= 0111011$

Now adding this with M.

$$\begin{array}{r} 1010100 \\ +\ 0111011 \\ \hline \boxed{1}\,0001111 \\ +\ 1 \\ \hline 0010000 \end{array}$$

end carry → 1

∴ Answer $= 10000$.

b) Use 1's complement to subtract $M = 1000100$
$N = 1010100$

Soln

Given, $M = 1000100$
$N = 1010100$

1's complement of $N = 2^7 - 2^0 - 1010100$
$= 0101011$

Now adding this with M.

$$\begin{array}{r} 1000100 \\ +\ 0101011 \\ \hline 1101111 \end{array}$$ (no end carry)

Answer $= -$(1's complement of 1101111)
$= -(2^7 - 2^0 - 1101111)$
$= -10000$.

Scanned by TapScanner

1.6 BINARY CODES

Binary codes for the decimal digits.

Decimal digit	(BCD) 8421	Excess 3 code	84-2-1	2421
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	1001	0100	0100
5	0101	1000	1011	1011 (2+2+1)
6	0110	1001	1010	1100
7	1001	1010	1001	1101
8	1000	1011	1000	1110
9	1001	1100	1111 (8+4-3)	1111

Error detection codes / Parity-bit generation.

a (Message)	P(odd)	b (Message)	P (even)
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1

a (Message)	P (odd)	b (Message)	P (even)
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

A parity bit is an extra bit included with a message to make the total number of ones either even or odd.

Reflected / Gray code (Imp)

Decimal no.	Reflected code
0	0000
1	0001
2	0011
3	0010
4	0100
5	0101
6	0111
7	0110
8	1100
9	1101
10	1111
11	1110
12	1000
13	1001
14	1011
15	1010

Scanned by TapScanner

Rough for Reflected or grey code

K - Map

if, a', b', c', d' comes then it is $= 0$.
& if a, b, c, d comes then it is $= 1$.

Note:- Reading pattern should be always in abcd form.

	$c'd'$	$c'd$	cd	cd'
$a'b'$	0000	0001	0011	0010
$a'b$	0100	0101	0111	0110
ab	1100	1101	1111	1110
ab'	1000	1001	1011	1010

Scanned by TapScanner

Navneet
Date
Page

Integrated Circuits:-

An integrated circuit (IC) is a small silicon semiconductor crystal, called a chip, containing electrical components such as transistors, diodes, resistors, capacitors etc. These components are interconnected inside the chip to form an electronic circuit.

Integrated circuits come in two types of packages, the flat package and dual-in-line (DIP) package. The dual-in-line package is the most widely used type because of the low price and easy installation on circuit boards. The envelope of the IC package is made of plastic or ceramic. Most packages have standard sizes, and the number of pins ranges from 8 to 64.

(IC)'s helped in the reduction of size of electronic machines. The cost of (IC)'s is very low, which made them economically low to use. They reduced power consumption and made the computer system more economical to operate.

(IC)'s are used in amplication, voltage comparators etc. The use of (IC)'s reduced the number of external wiring connections because ~~mou~~ many of the connections are internal to the package.

Scanned by TapScanner

②

Chapter-2

Boolean Algebra

Navneet
Date
Page

① Boolean Algebra:-

Boolean Algebra is a mathematical system with a set of elements, a set of operators and a number of unproved postulates.

Postulates:-

Law	Dual pair	Remark
$A+B=B+A$	$A.B=B.A$	commutative law
$A+(B+C)=(A+B)+C$	$A.(B.C)=(AB)C$	Associative law
$A(B+C)=AB+AC$	$A+(BC)=(A+B).(A+C)$	Distributive law.
$A+0=A$	$A.1=A$	Identity
$A+1=1$	$A.0=0$	"
$A+A=A$	$A.A=A$	"
$A+\bar{A}=1$	$A.\bar{A}=0$	"
$A+AB=A$	$A(A+B)=A$	Absorption law
$A+\bar{A}.B=A+B$	$A(A+B)=A$	Theorem
$\bar{\bar{A}}=A$		Involution.

② Duality Theorem:-

According to the duality theorem a Boolean relation can be written to another Boolean relation by

→ changing each 'OR' operation to 'AND' operation.
→ changing each 'AND' operation to 'OR' operation.
→ complementing any 0 or 1 appearing in the expression.

Example: $A(B+C) = A.B+AC$ (Distributive law)

Now, it can be written using duality theorem as

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

Scanned by TapScanner

V.V.I

Demorgan's Theorem:-

It is an important theorem in Boolean algebra. Demorgan suggested two theorems:-

Theorem:-1 The complement of a product is equal to the sum of the complements.

i.e. $\overline{A.B} = \overline{A} + \overline{B}$

Proof

A.B first then $\overline{A.B}$

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

From above table $\overline{A.B} = \overline{A} + \overline{B}$

proved

Theorem:-2 The complement of a sum is equal to the product of the complements.

i.e $\overline{A+B} = \overline{A}.\overline{B}$

Proof

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

From above table $\overline{A+B} = \overline{A.B}$

proved

Scanned by TapScanner

Navneet

Date
Page

Q. Reduce the Boolean expression:-

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

Soln.

Using Demorgan's theorems

$$Y = \overline{\bar{A}\cdot\bar{B}C} \cdot \overline{A\bar{B}\bar{C}} \qquad (\because \overline{A\cdot B} = \bar{A}+\bar{B} \ \& \ \overline{A+B} = \bar{A}\cdot\bar{B})$$

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

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

This is simplified expression.

Q. Simplify the following Boolean expression using Boolean Algebra.

(a) $(A+B)(A+\bar{B})$

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

Soln

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

$$= A\cdot A + A\cdot\bar{B} + A\cdot B + B\cdot\bar{B} \qquad (\because A\cdot A = A, \ A\cdot\bar{A} = 0)$$

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

$$= A(1+\bar{B}+B) \qquad \text{taken common.}$$

$$= A(1+1) \qquad (\because A+\bar{A} = 1)$$

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

$$= A \qquad (\because A\cdot 1 = A)$$

This is simplified Boolean expression.

Scanned by TapScanner

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

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

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

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

$= A(B+C)$

This is simplified form using Boolean expression.

(3) Boolean Functions:-

i) Sum of product form (SOP):- (SOP form)

The SOP form expression contains two or more AND terms which are OR'ed together.

Example:- $AB\bar{C} + \bar{B}C + AB$.

ii) Product of sum form (POS):- (POS form)

The POS form expression contains two or more OR terms which are AND'ed together.

Example:- $(A+B+\bar{C}).(B+C).(A+\bar{B})$

(a) Canonical Form:-

Most of the time, the SOP or POS form expression does not contains all the literals (variables).

Example:- $ABC + B\bar{C}$ here A is missing.

→ If each term in SOP and POS forms contains all the literals than these are known as canonical (standard) SOP and POS expression.

Example:- $A\bar{B} + \bar{A}B + AB$

Navneet

Date
Page

(b) Converting sop expression into canonical sop form:-

Q. Convert the given expression into canonical sop form.

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

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

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

[∵ missing letter are checked and put it coz $A+\bar{A}=1$]

multiplied

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

[∵ Common ABC is written 1 time to make it standard or canonical form.]

This is canonical form.

Q. Convert the following pos expression into canonical pos expression.

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

Soln

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

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

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

[from property of Boolean Algebra]

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

[Taken common]

This is canonical pos expression.

Scanned by TapScanner

(c) Minterms and Maxterms:-

Minterms → These are designed as the product terms which contains all the variables either in normal form or complement form.

Example:-

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

$$000 \quad 001 \quad 011 \quad 110$$

$$= m_0 + m_1 + m_3 + m_6$$

$$Y = \Sigma m(0,1,3,6)$$

Maxterms → These are designed as the sum terms which contains all the variables either in the normal form or complemented form.

Example:-

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

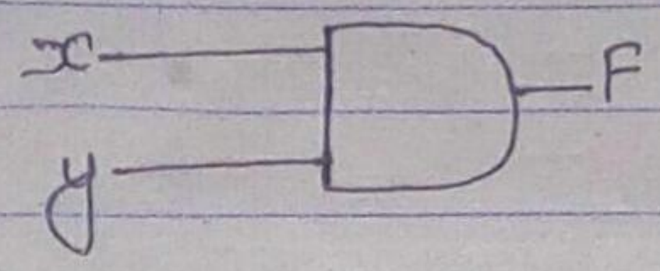
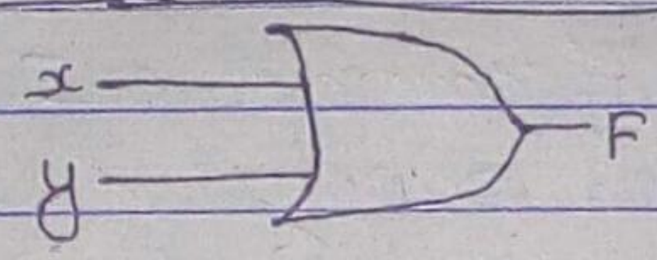
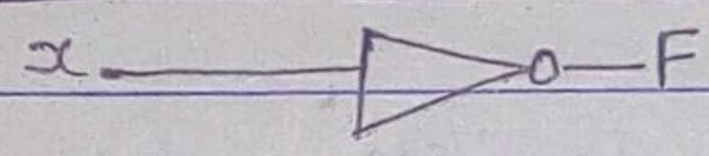
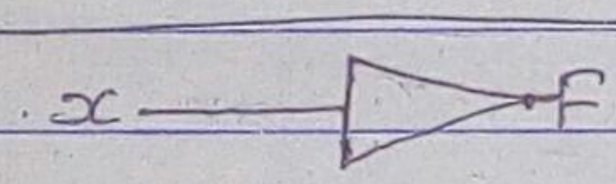
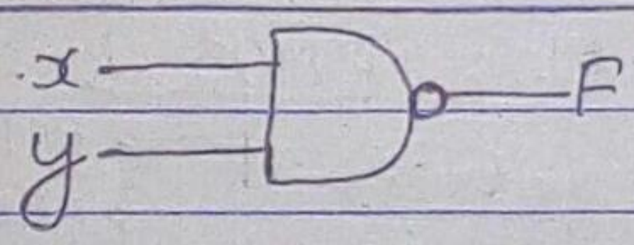
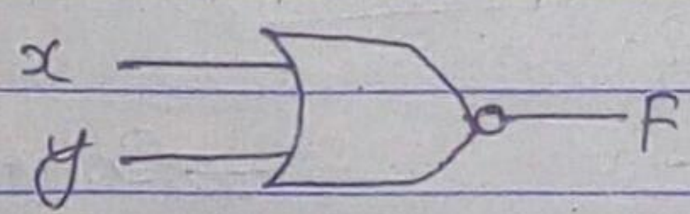
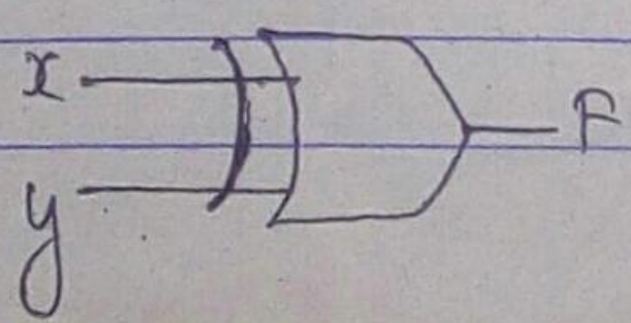
$$001 \quad 011 \quad 110$$

$$= M_1 \cdot M_3 \cdot M_6$$

$$Y = \pi . M(1,3,6)$$

$$\boxed{\text{Minterms} \xleftrightarrow{\text{Complement}} \text{Maxterms}}$$

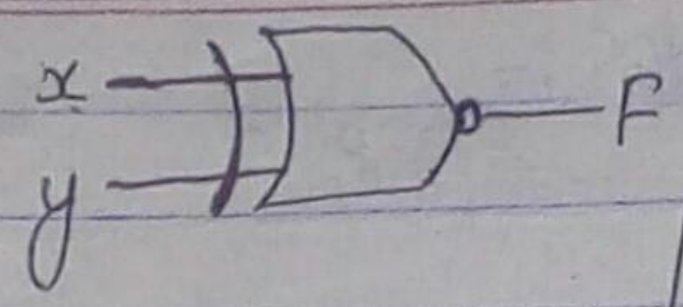
Logic Gates :-

Name	Graphic Symbol	Algebraic function	Truth table
i) AND gate (र)		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
ii) OR gate (या)		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
iii) Inverter OR NOT gate (उल्टा)		$F = x'$	x | F 0 | 1 1 | 0
iv) Buffer (same output)		$F = x$	x | F 0 | 0 1 | 1
v) NAND a) (र) b) (त्यसपछि उल्टा) ⇒ AND को उल्टा		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
vi) NOR (या) (त्यसपछि उल्टा) ⇒ OR को उल्टा		$F = (x+y)'$	x | y | F 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0
vii) Exclusive-OR gate (XOR) (एक मात्र मात्र signal भस true)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

MOSFET → metal-oxide semiconductor field effect transistor.

viii) Exclusive-NOR gate. (X-NOR)

⇒ XOR को उल्टा



$F = xy + x'y'$

$= x \odot y$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Digital logic families:-

A group of compatible IC's with the same logic levels and supply voltages fabricated for performing various logic functions refered to as a logic family.

Digital Logic families

- **Bipolar** (Uses both electrons and holes as charge carriers) {resistors, diodes or transistors are used in bipolar IC's}
  - **Saturated**
    - → Resistor Transistor logic (RTL)
    - → Diode Transistor logic (DTL)
    - → Integrated Injection logic (I^2L)
    - → High threshold logic (HTL)
    - → Transistor transistor logic (TTL)
  - **Un-saturated** (faster than saturated logic families)
    - → Schottkey TTL.
    - → Emitter coupled logic (ECL)
- **Unipolar** (Uses only one type of charge carriers i.e holes or electrons) {MOS devices}
  - → PMOS (P chanel uses MOSFET)
  - → N-MOS (N chanel uses MOSFET)
  - → CMOS (complementary MOSFET)

Both P and N channel are used.

higher voltage → logic 1 → means working condition or working voltage of device

& lower voltage → logic 0 → means ~~device~~ not working voltage of device.

Positive and Negative logic

Positive logic

i) The higher voltage corrosponds to logic "1" is positive logic & The lower voltage corrosponds to logic "0" is also positive logic.

Eg

ii) If we have 5V then it will corrospond to logic "1" and if we have 0V then it will corrospond to logic "0".

Negative logic

~~i) The lower vo~~ (In this we have to reverse the condition)

i) The higher voltage corrosponds to logic "0" and lower voltage corrosponds to logic "1".

Eg.

ii) If we have 5V then it will corrospond to logic "0" and if we have 0V then it will corrospond to logic "1".

Dual form.

It is used to convert positive logic into negative logic or negative logic to positive logic.

Problems related to this

logic 0 → −5V

logic 1 → 0V

Soln

Comparing above we get

0 > −5

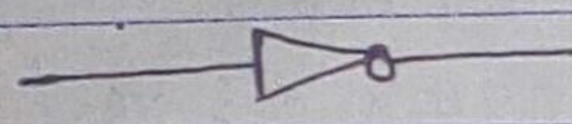
Now, '1' '0' ← It is positive logic

'0' '1' ← It is negative logic.

Universal gates:-

i) NAND
ii) NOR

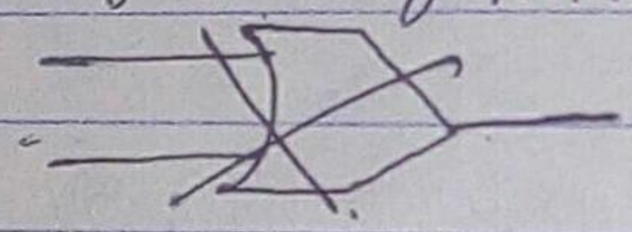
i) NAND gate as universal gate.

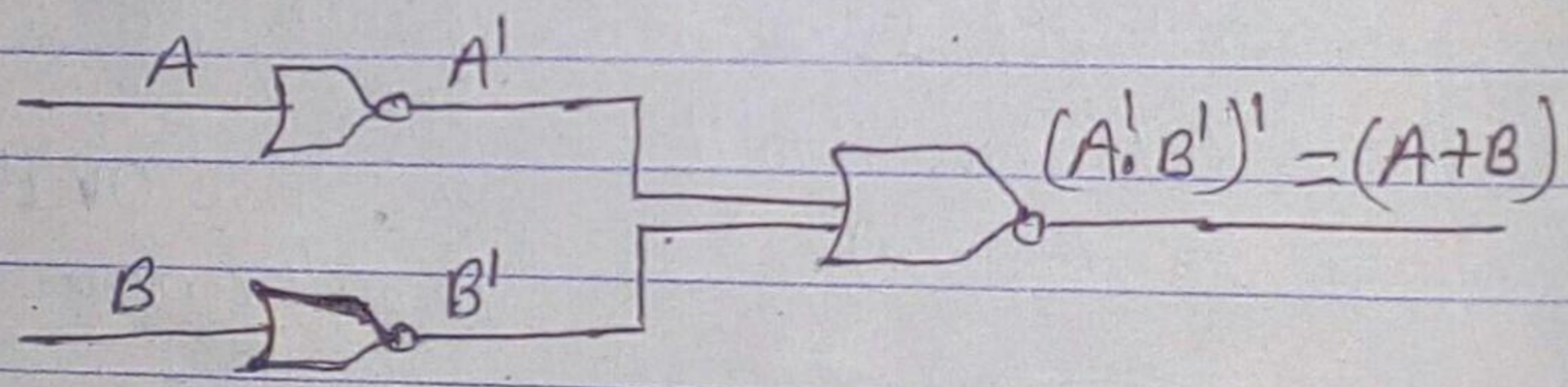
(a). 

(NOT gate).

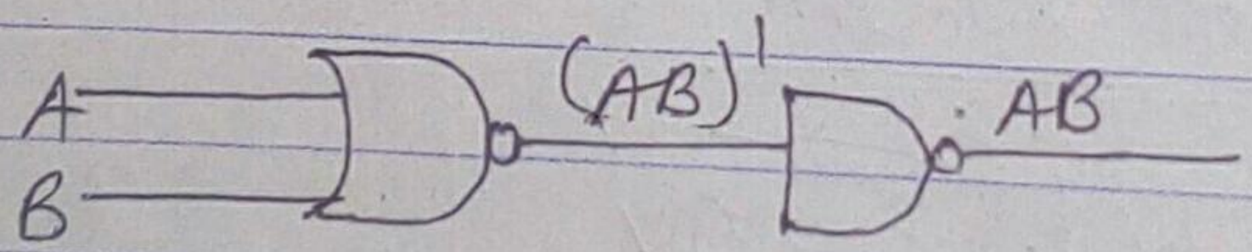
Not gate using NAND gate

(b). OR gate using NAND gate.





OR gate

(c). 

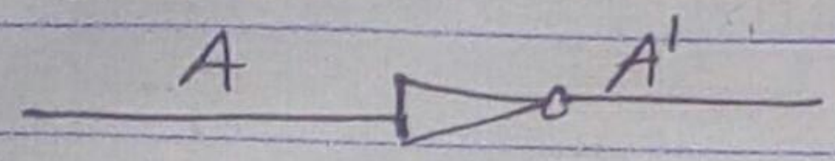
AND gate using NAND gate.

Using NAND gate we can perform basic gates like NOT, AND, OR as above, so NAND gate is called universal gate.

Scanned by TapScanner

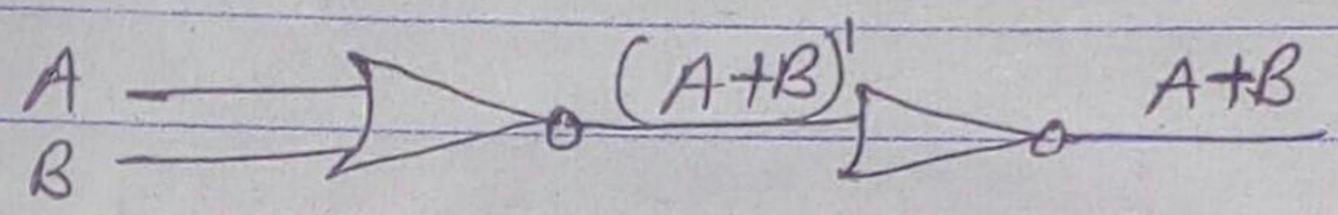
ii) NOR gate as universal gate:-

(a).



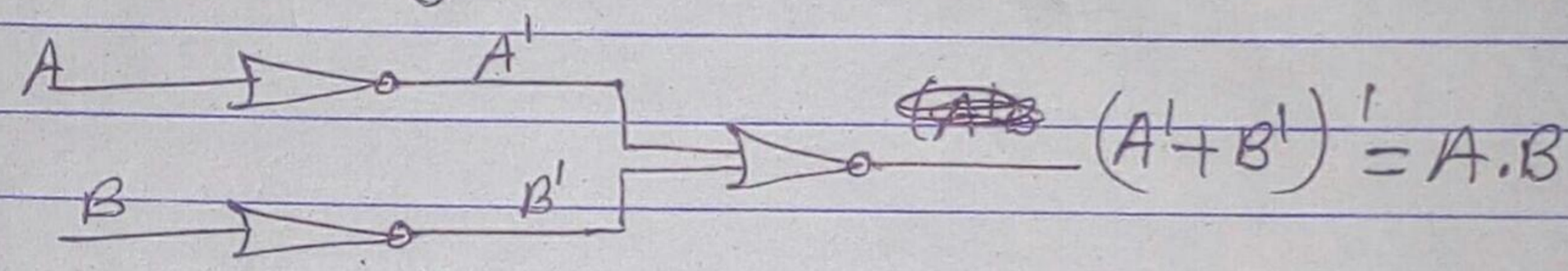
NOT gate using NOR gate.

(b).



AND gate using NOR gate.

(c)



OR gate using NOR gate.

Using NOR gate we can perform basic gates like NOT, AND, OR as above, so, NOR gate is called universal gate.

Principle of duality:-

The duality principle states that, if a theorem or a result can be established for a Boolean algebra then the derived theorem given one by:-

i) Interchange of (+) & (·)

ii) Interchange of elements (0) and (1)

Also holds true and not need to proof seperately.

Chapter = 3

Simplification of Boolean functions:-

Karnaugh Map (K-map):-

The K-map is a graphical representation that provides a systematic method for simplifying the Boolean Expression.

i) Two variable K-map:-

For n variable K-map 2^n cells are required.
Therefore for 2-variable K-map, $2^2 = 4$ cells will be required.

A \ B	B'	B
A'	A'B'	A'B
A	AB'	AB

A \ B		
	0	1
	2	3

← way of filling table.

ii) Three variable K-map:-

A \ BC	B'C'	B'C	BC	BC'
A'	A'B'C'	A'B'C	A'BC	A'BC'
A	AB'C'	AB'C	ABC	ABC'

A \ BC				
	0	1	3	2
	4	5	7	6

iii) Four variable K-map:-

AB \ CD	C'D'	C'D	CD	CD'
A'B'	A'B'C'D'	A'B'C'D	A'B'CD	A'B'CD'
A'B	A'BC'D'	A'BC'D	A'BCD	A'BCD'
AB	ABC'D'	ABC'D	ABCD	ABCD'
AB'	AB'C'D'	AB'C'D	AB'CD	AB'CD'

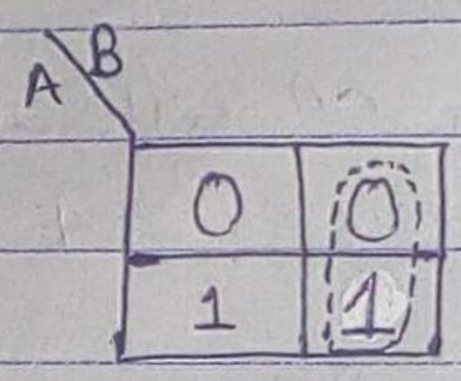
AB \ CD				
	0	1	3	2
	4	5	7	6
	12	13	15	14
	8	9	11	10

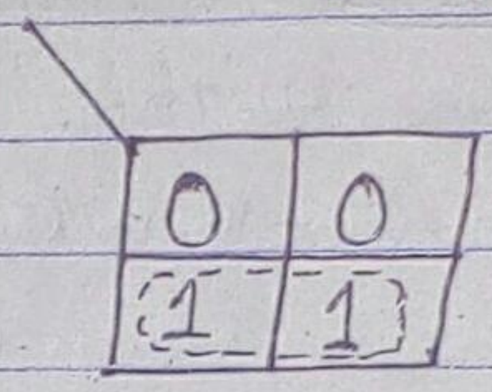
Scanned by TapScanner

Rules for simplification of Boolean functions by using K-map

i) Zero's should not be included in group. Only 1 must be taken while making group.

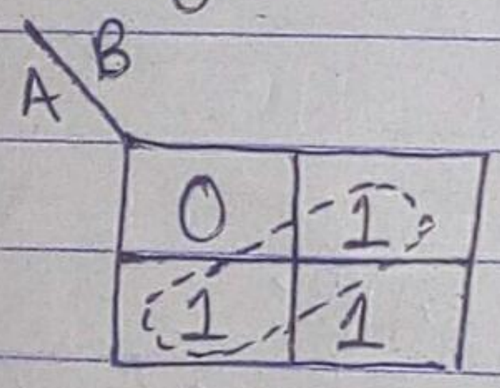
for e.g

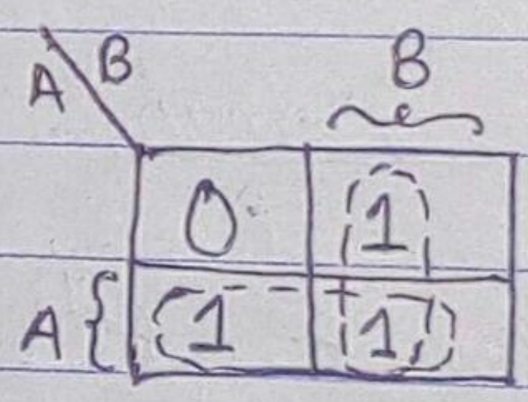




ii) Groups may be horizontal or vertical, but not diagonal.

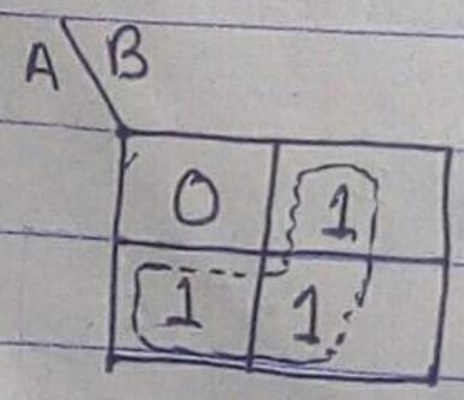
For e.g

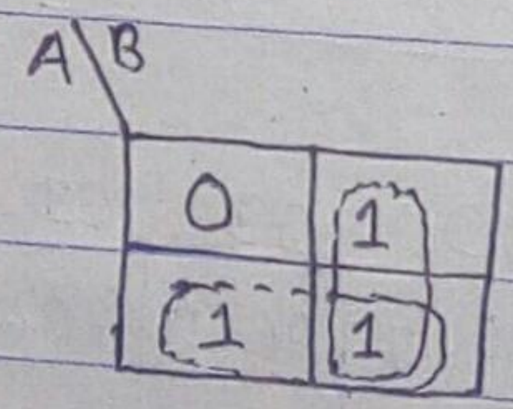




iii) Make groups of 2, 4, 8, 16 etc (i.e. 2^n basis)

For e.g.





iv) Group must be as large as possible.

For e.g

AB \ CD			
0	0	0	1
1	1	1	0
0	1	1	0
0	1	0	1

✗

AB \ CD			
0	0	0	1
1	1	1	0
0	1	1	0
0	1	0	1

✓

v) Each cell containing a one must be part of group if possible.

For e.g.

A \ B	
0	1
1	1

✗

A \ B	
0	1
1	1

✓

vi) Groups may overlap

vii) Groups may wrap/fold around the table.

For e.g.

AB \ CD			
1	0	0	1
0	0	0	0
0	0	0	0
1	0	0	1

OR

AB \ CD			
1	0	0	1
0	0	0	0
1	0	0	1
0	0	0	0

viii) Logical expression must be in canonical form before using K-map.

[canonical form → form containing all variables]

SOP → Σ → minterm → presence denoted by 1 in table in any function.
POS → π → maxterm → presence denoted by 0 in table

Note:-

i) In any 4 variable K-map 1 square represents 1 minterm giving a term of 4 literals.
ii) Two adjcent squares represents a term of 3 literals.
iii) Similarly 4 adjcent square represent the term of 2 literals and 8 adjcent squares represents term of 1 literals and 16 adjcent square represents the function equals to 1.

(Minterms SOP related questions) (Σ)

Q. Using K-map simplify the following Boolean function.

(a) $Z = a'b' + ab'$

Soln Here,

The Boolean Function consist of two variables so using K-map for two variables we have,

a \ b	b'	b
a'	1	
a	1	

From K-map we have b' is common for this group.
So, simplified form is $Z = b'$.

Step for Z i) पुरै row या column हेर्ने
ii) row पुरै लिए column बाट common हेर्ने र column पुरै लिए row बाट हेर्ने,

(b) $Z = a'bc' + abc' + abc + ab'c$

Sol^n

The Boolean function consist of three variables So, using K-map for three variables.

a \ bc	b'c'	b'c	bc	bc'
a'				1
a		1	1	1

bc र bc बाट common term

Simplified form $Z = ac' + bc'$

column

पुरै row मा group

पुरै column तर यसको row a' र a मा केहि हैन त्यसैले केहि लेख्ने.

(c) $F = ab'c + abc' + a'bc + abc + ab'c' + a'b'c'$

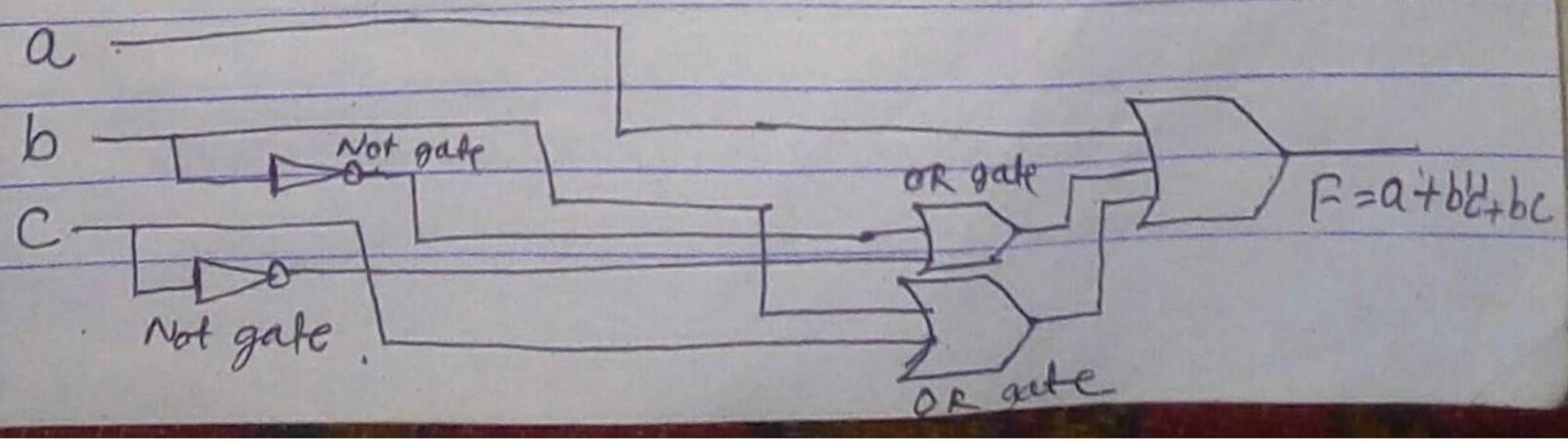
Sol^n

The Boolean function consist of three variables. So, using K-map for three variables.

a \ bc	b'c'	b'c	bc	bc'
a'	1		1	
a	1	1	1	1

Simplified form $Z = a + b'c' + bc$

circuit diagram (only if asked)



$+$ (या) $\rightarrow$ OR gate $\rightarrow$ minterms (SOP)

$(\cdot)$ $\rightarrow$ AND gate $\rightarrow$ maxterms (POS)

(d) $Z = w'x'y'z' + w'x'y'z + w'xyz + w'xyz' + wxyz$

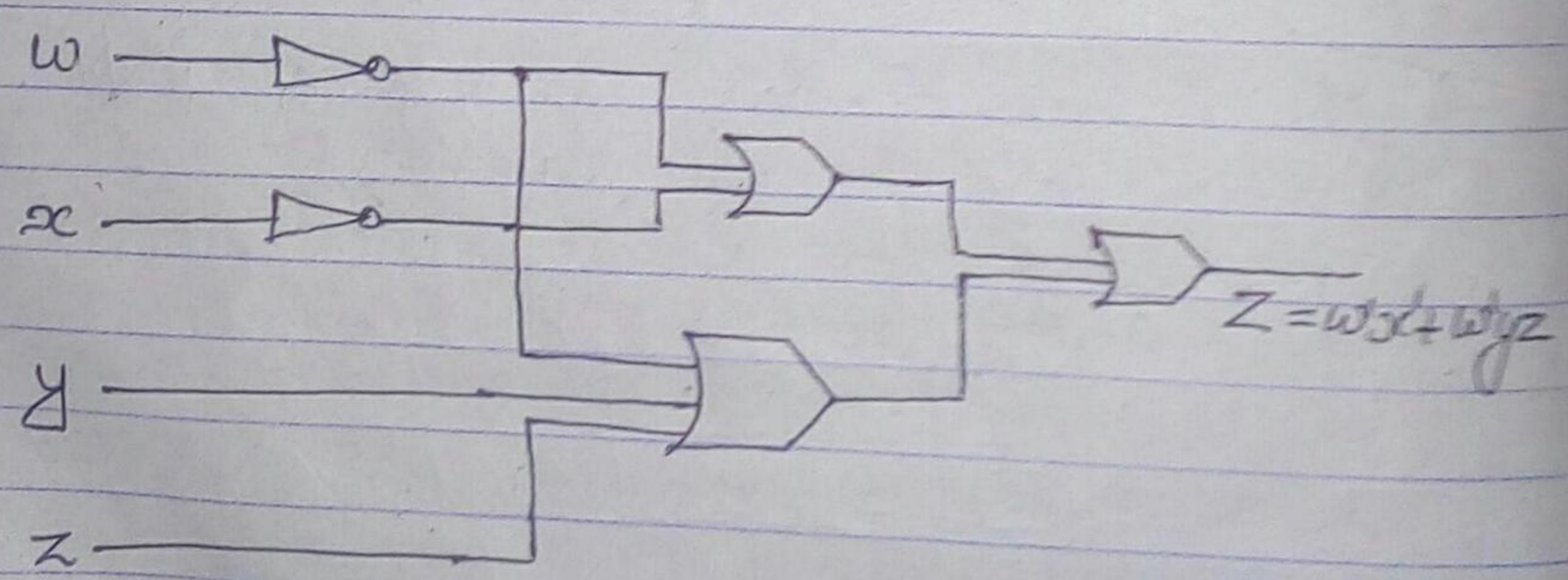
Sol^n

The Boolean function consist of 4 variables. So, using K-map for 4 variables.

$wx \backslash yz$	$y'z'$	$y'z$	yz	yz'
$w'x'$	1	1	1	1
$w'x$			1	
wx				
wx'				

Simplified form $Z = w'x' + w'yz$

Circuit diagram



Date
Page

(*) $F(ABCD) = \Sigma(0,1,2,8,9,10)$

Soln

Given Boolean function contains 4 variables So, using K-map for 4 variables.

AB \ CD	C'D'	C'D	CD	CD'
A'B'	1 (0)	1 (1)	(3)	1 (2)
A'B	(4)	(5)	(7)	(6)
AB	(12)	(13)	(15)	(14)
AB'	1 (8)	1 (9)	(11)	1 (10)

Simplified form $F = A'B'C' + AB'C'$

(*) $F(a,b,c,d) = \Sigma(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)$

Soln

Given Boolean function contains 4 variables. So, using K-map for 4 variables.

ab \ cd	c'd'	c'd	cd	cd'
a'b'	1	1	1	1
a'b	1	1	1	1
ab	1	1	1	1
ab'	1	1	1	1

Simplified form $F = 1$

Have look at point (iii) of Note before 1 page.

Scanned by TapScanner

Navneet

Date
Page

(*) $F(a,b,c,d) = \Sigma(0,2,4,6,8,10,12,14,15)$

Soln

Given Boolean function contains 4 variables. So, using K-map for 4 variables.

ab \ cd	c'd'	c'd	cd	cd'
a'b'	1			1
a'b	1			1
ab	1		1	1
ab'	1			1

common in column c'd' and cd'

Simplified form of $F = d' + abc$

column cd and cd' common term

one fixed row

(*) $F(x,y,z) = \Sigma(2,3,4,6,7)$

Soln

Given Boolean function contains 3 variables. So, using K-map for 3 variables.

x \ yz	y'z'	y'z	yz	yz'
x'			1	1
x	1		1	1

yz र yz' बाट common

Simplified form of $F = y + xz'$

पुरै row

column y'z' र yz' बाट common.

{Maxterms pos related problems (π)} :-

Q. Simplify $F(A,B,C,D) = \pi(4,6,10,12,13,15)$

Solⁿ

Given Boolean function contain 4 variables. So, using K-map for 4 variables.

AB \ CD	CD	CD'	C'D'	C'D
AB				
AB'	0			0
A'B'	0	0	0	
A'B				0

minterms को यहाँ दूसरी ओर

(simplified form पाने pos को term का लेंगे)

cd' र c'd' बाट common

cd र c'd बाट common

Simplified form of F $= (b'+c+d)\cdot(a'+b'+d')\cdot(a+b'+d)$

ab' र a'b' बाट common — पुरै column — पुरै row — पुरै column

Q. $F(a,b,c,d) = \pi(0,1,2,5,6,7)$

Solⁿ

ab \ cd	cd	cd'	c'd'	c'd
ab	0	0		0
ab'		0	0	0
a'b'				
ab'				

Simplified form of F $= (a+b+c)\cdot(a+b'+d')\cdot(a+c'+d)$

minterm

Q. $F(w,x,y,z) = \Sigma(0,4,5,6,8,12,13,15)$.

Soln

wx \ yz	$y'z'$	$y'z$	yz	yz'
$w'x'$	1			
$w'x$	1	1		1
wx	1	1	1	
wx'	1			

$F = y'z' + y'x + wxz + w'xz'$

Don't care condition:-

Sometime we encounter the situation in which some of minterms can't occur. In such situation it doesn't matter that the function produce 0 or 1 for a given minterm. Since the function may be either 0 or 1. We say that we do not care what the function output is to be for this minterm. Hence the minterm that may produce either 0 or 1 for the function are said to be don't care conditions.

Don't care condition is indicated by either 'X' or ϕ on K-map. A don't care square maybe assumed either as 1 square or 0 square as derived while forming groups of square for simplification. Any one such square of some of them may be included or may not be included while forming groups.

Don't care condition मा सकेसम्म ϕ लाई group मा नराखे हटाउने तर 1 नछुट्ने गरि मात्र

Σ_ϕ may also be asked as Σ_x

For eg related to minterms (Σ).

Q.1 Consider the following Boolean function.

$$F(a,b,c) = \Sigma(0,2,6) + \Sigma_\phi(1,3,5)$$

Solⁿ

The given Boolean function contains three variables. So, using K-map for three variables.

a \ bc	bc'	b'c	bc	bc'
a'	1	ϕ	ϕ	1
a		ϕ		1

Simplified form $F(a,b,c) = a' + bc'$

Q.2 $F(w,x,y,z) = \Sigma(0,1,3,5,9,11,15) + \Sigma_\phi(2,13)$

Solⁿ

The given Boolean function contains 4 variables. So, using K-map for four variables.

wx \ yz	y'z'	y'z	yz	yz'
w'x'	1	1	1	ϕ
w'x		1		
wx		ϕ	1	
wx'		1	1	

∴ Simplified form of given Boolean function $F(w,x,y,z) = w'x' + y'z + wz$

Scanned by TapScanner

Navneet

Date
Page

Q.3 $F(w,x,y,z) = \Sigma(1,3,7,11,15) + \Sigma_{\phi}(0,2,5)$.

Soln

Given Boolean function contains 4 variables.
So, using K-map for 4 variables.

wx \ yz	y'z'	y'z	yz	yz'
w'x'	ϕ	1	1	ϕ
w'x		ϕ	1	
wx			1	
wx'			1	

This table is wrong because we have included ϕ which is seperate from others and can be excluded from group.

wx \ yz	y'z'	y'z	yz	yz'
w'x'	ϕ	1	1	ϕ
w'x		ϕ	1	
wx			1	
wx'			1	

$\therefore$ Simplified form of given Boolean function
$F(w,x,y,z) = w'x' + yz$.

Q.4 $F(a,b,c) = \Sigma(0,2,6) + \Sigma_{\phi}(1,3,5)$.

Soln

Given Boolean function contains 3 variables. So, using K-map for 3 variables.

a \ bc	bc'	b'c	bc	bc'
a'	1	ϕ	ϕ	1
a		ϕ		1

$\therefore$ Simplified form of given Boolean function $F(a,b,c)$
$= a' + bc'$.

Scanned by TapScanner

Similarly for maxterms (π):- only change in notation of abc and 1 to 0. ($a \to a'$, $a' \to a$)

यहाँ ϕ हटाकर 0 डालना है.

Q.1 $F(a,b,c) = \pi(0,2,6) + \pi_\phi(1,3,5)$

Sol^n

Given Boolean function contains 3 variables. So using K-map for 3 variables.

a \ bc	bc	bc'	b'c'	b'c
a	0	ϕ	ϕ	0
a'		ϕ		0

$\therefore$ Simplified form of given Boolean function $F(a,b,c) = a + b'c$

Q.2 $F(w,x,y,z) = \pi(1,3,7,11,15) + \pi_\phi(0,2,5)$

Sol^n

Given Boolean function contains 4 variables. So, using K-map for 4 variables.

wx \ yz	yz	yz'	y'z'	y'z
wx	ϕ	0	0	ϕ
wx'		ϕ	0	
w'x'			0	
w'x			0	

$\therefore$ Simplified form of given Boolean function $F(w,x,y,z)$
$= wx + y'z'$

Unit = 4

Combinational Circuits:-

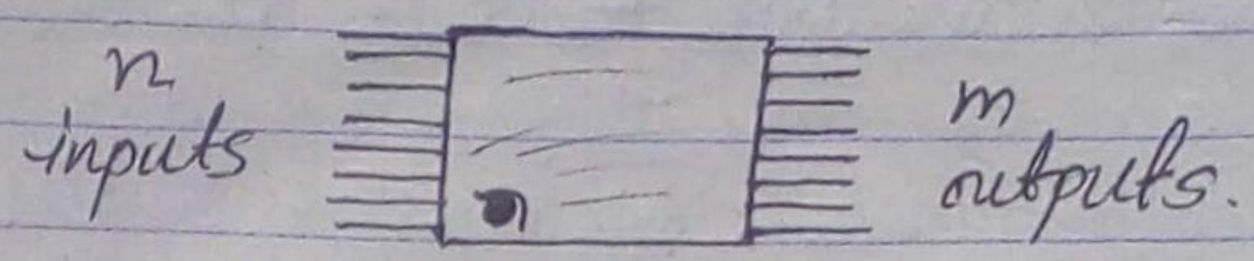


fig. combined circuit

Design procedure of combinational circuit:-

i) Problem should be stated/noticed.
ii) The number of input and output variables are declared.
iii) Both variables are assigned to letters or symbols.
iv) According to the problem statement, truth table is generated.
v) Simplified Boolean expression is obtained (either by K-map or Boolean algebra).
vi) Finally draw the logic circuit diagram according to the simplified function.

Combinational logic circuit:-

It is a type of digital logic which is implemented by Boolean circuits, where the ~~input~~ output is a pure function of the present input only. In other words, sequential logic has memory while combinational logic does not.

Half adder → NAND 1st → carry

NOR 1st and last → carry.

2nd & 3rd Not connected by 1st but connected for all others.

Half subtractor →

NAND → 3rd → Borrow

NOR → 2nd

4th & 5th → carry

Adders

The combinational circuits which are used for performing addition tasks are known as adders. Depending upon the number of bits to be added the adders are classified into two types.

i) Half adder.

ii) Full adder.

i) Half adder:-

The combinational logic circuit which is used for addition of two single bit number is known as half adder.

Truth table for inputs for half adder is as follows:-

A	B	sum (XOR)	carry (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Expression for this is:-

$sum = A \oplus B = AB' + A'B$

$carry = A.B$

This table shows that the sum is equivalent to XOR gate and carry is equivalent to AND gate. So, logical circuit for this can be drawn by using the XOR gate and AND gate as follows:-

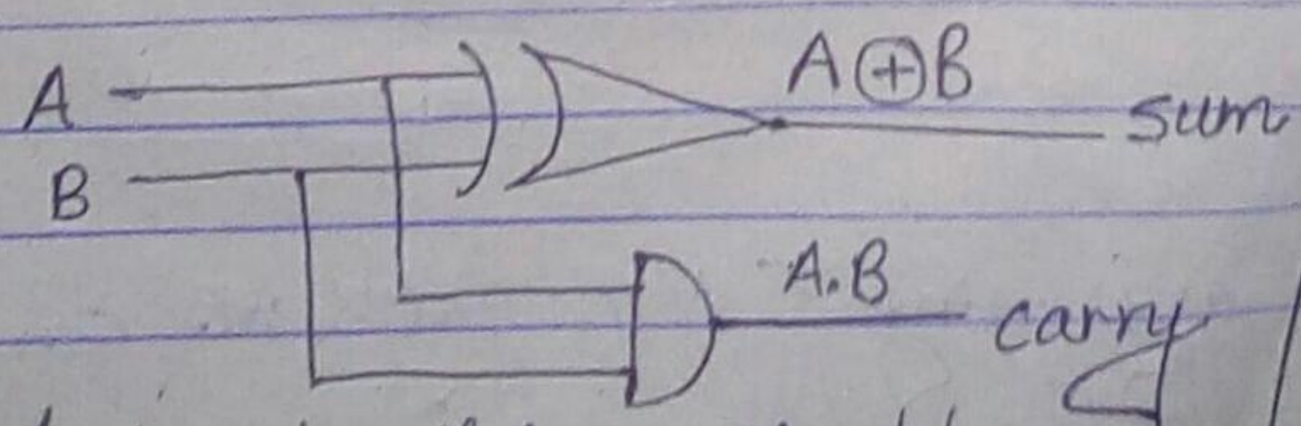
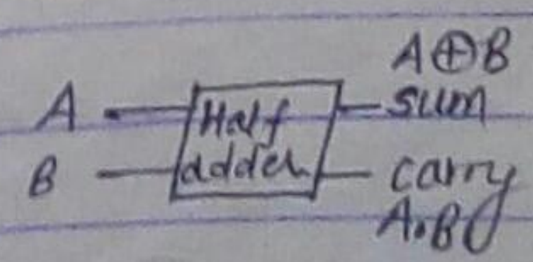
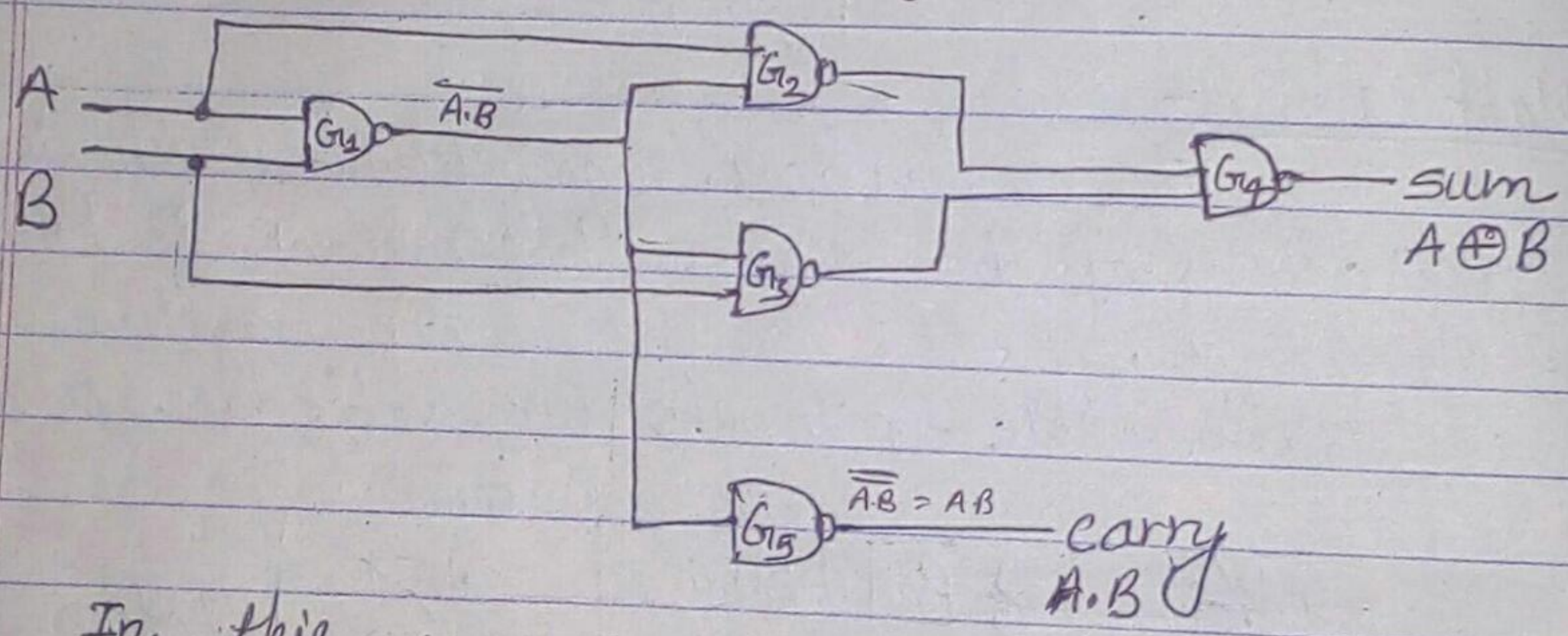


fig. logic circuit for half adder



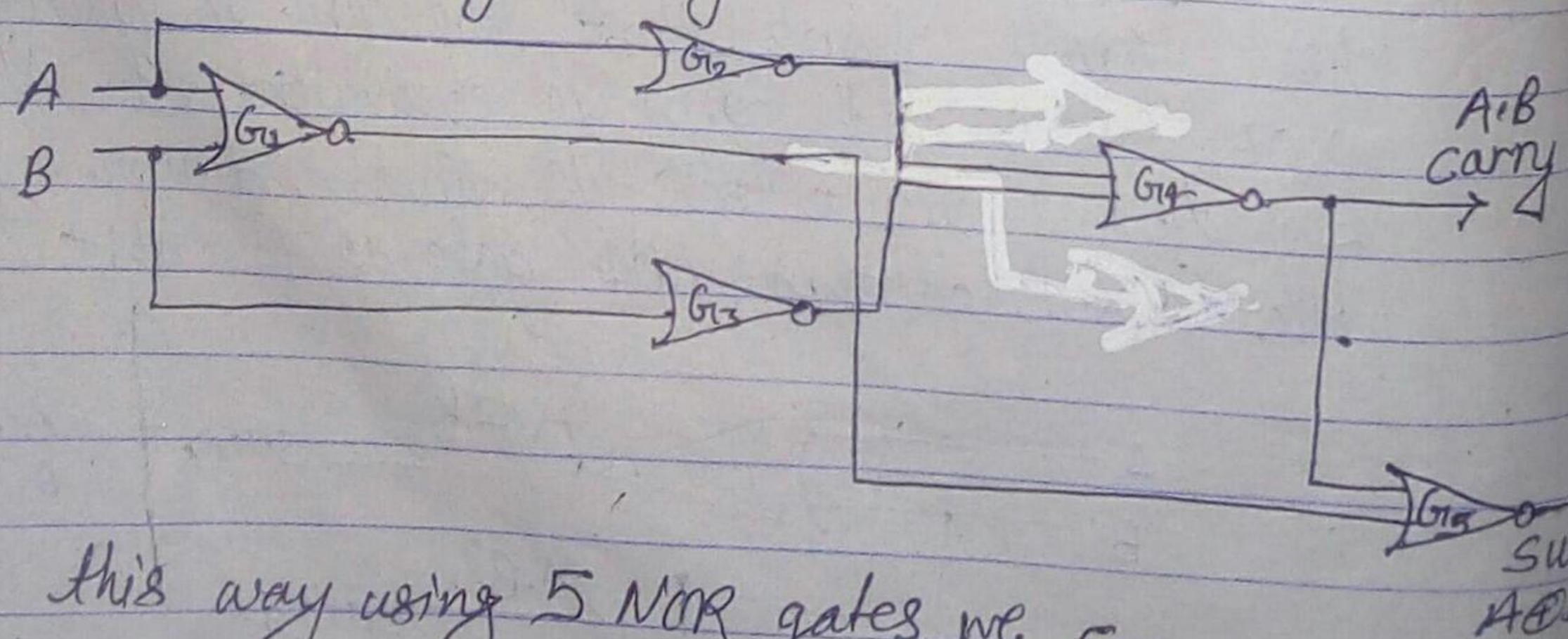
As we know that the NAND and NOR gates are the universal gates and are considered as the expert sets for developing logic circuits.
So, we can design the half adder using both of these gates as follows:-

Half adder using NAND gate:-



In this way using 5 NAND gates we can easily design the half adder circuit.

Half adder using NOR gate:-



In this way using 5 NOR gates we can design half adder circuit.

$x'yz + xy'z + xyz' + xyz$

$= z(x'y + xy) + xy$

$= (x \oplus y)z + xy$

Full adder:—

The combinational circuit which is used for addition of three binary bits or variables is known as full adder. ~~In this~~ For this first two bits are added after that third bit is added to the previous sum and the process of taking carry is taken into account from the previous case. The truth table for full adder for inputs is as follows:—

x	y	z	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$x'y'z$

$x'y'z'$

$x'yz$

$xy'z'$

$xy'z$

xyz'

xyz

By analyzing the truth table we can obtain the boolean function for sum and carry as follows:—

$$\text{Sum} = (x \oplus y) \oplus z = \overset{(3)}{xy'z'} + \overset{(2)}{x'yz'} + \overset{(4)}{xyz} + \overset{(1)}{x'y'z}$$

$$\text{Carry} = (x'y + xy')z + xy$$

$$= (x \oplus y)z + xy$$

This is generated from above table having 15 in carry column using K-map.

Now, the logical circuit can be drawn as follows:-

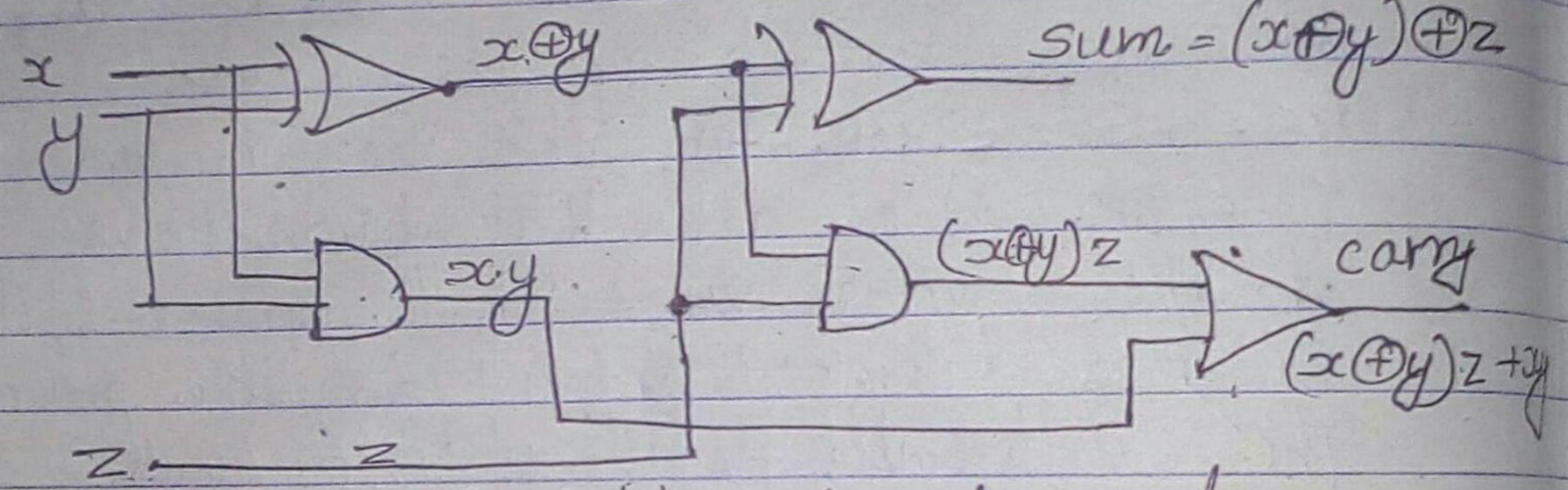


fig. full adder logical circuit

We can draw the full adder logical circuit using NAND gate as follows:-

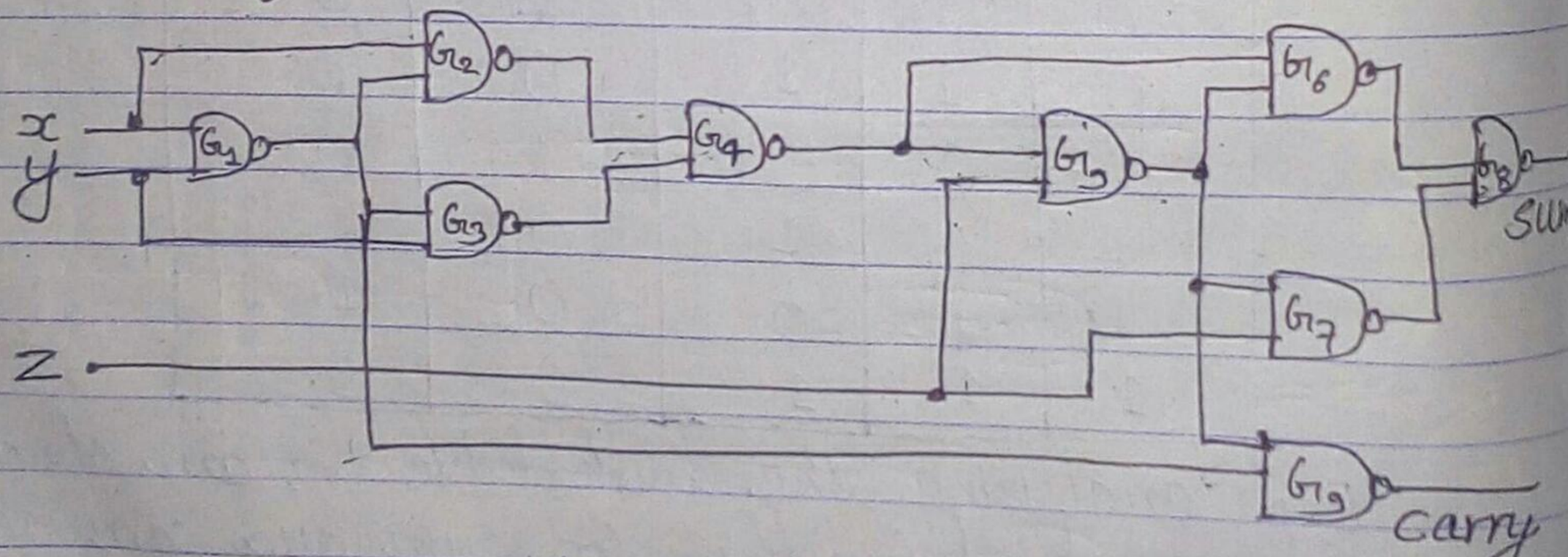


fig. full adder using NAND gate

Navneet
Date
Page

Subtractors:-

The combinational circuit which is used for calculating the difference between two binary numbers, the minuend and the number to be subtracted is called subtractor. Depending upon the number of bits subtractors are also ~~callifi~~ classified into two types:-

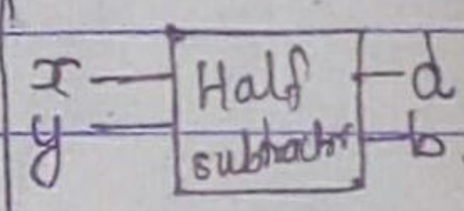
i) Half subtractor
ii) Full subtractor.

i) Half subtractor:-

The subtractor which is used to perform subtraction of two input bits in combinational logic circuit is called half subtractor. The truth table for half subtractor is as follows:-

(XOR)

x	y	difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



This table shows that the ~~sum~~ difference is equivalent to XOR gate which is similar to half adder and the borrow value is equal to $x'y$.

Boolean Expression for this is as follows:-

$$\text{Difference} = x \oplus y = x'y + xy'$$

$$\text{Borrow} = x'.y.$$

Scanned by TapScanner

Now, the logic diagram can be easily drawn for the half subtractor as follows:-

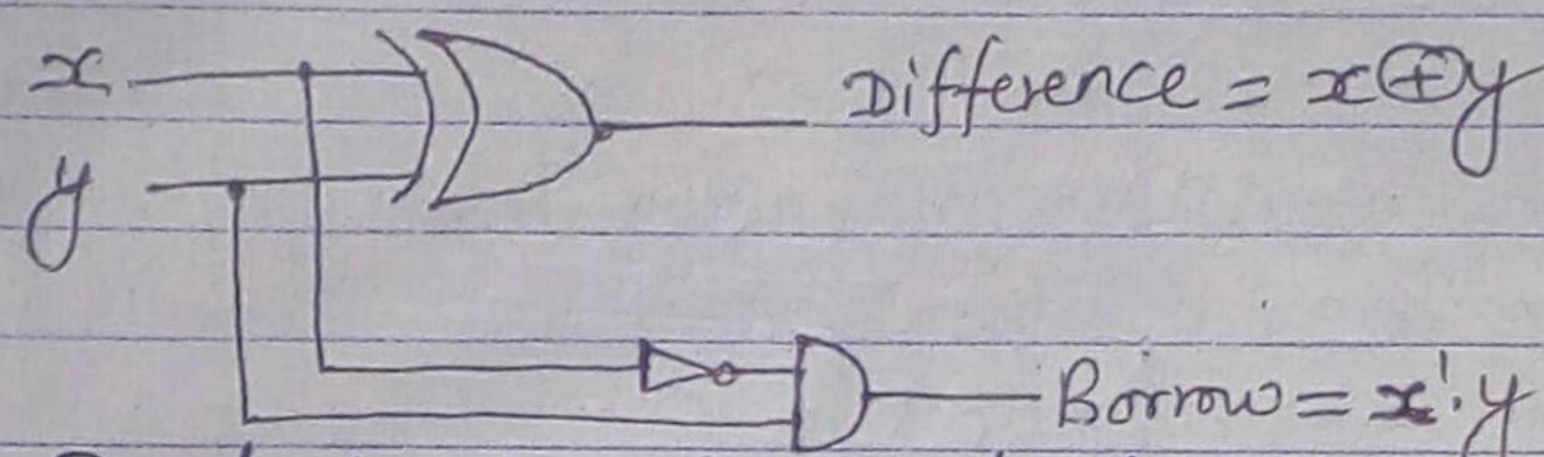


Fig. logic circuit for half subtractor.

As we know that NAND and NOR gates are universal gates and are considered as expert sets for developing logic circuits. So, we also can design the half subtractor using NAND and NOR gates as follows:-

Half subtractor using NAND gates:-

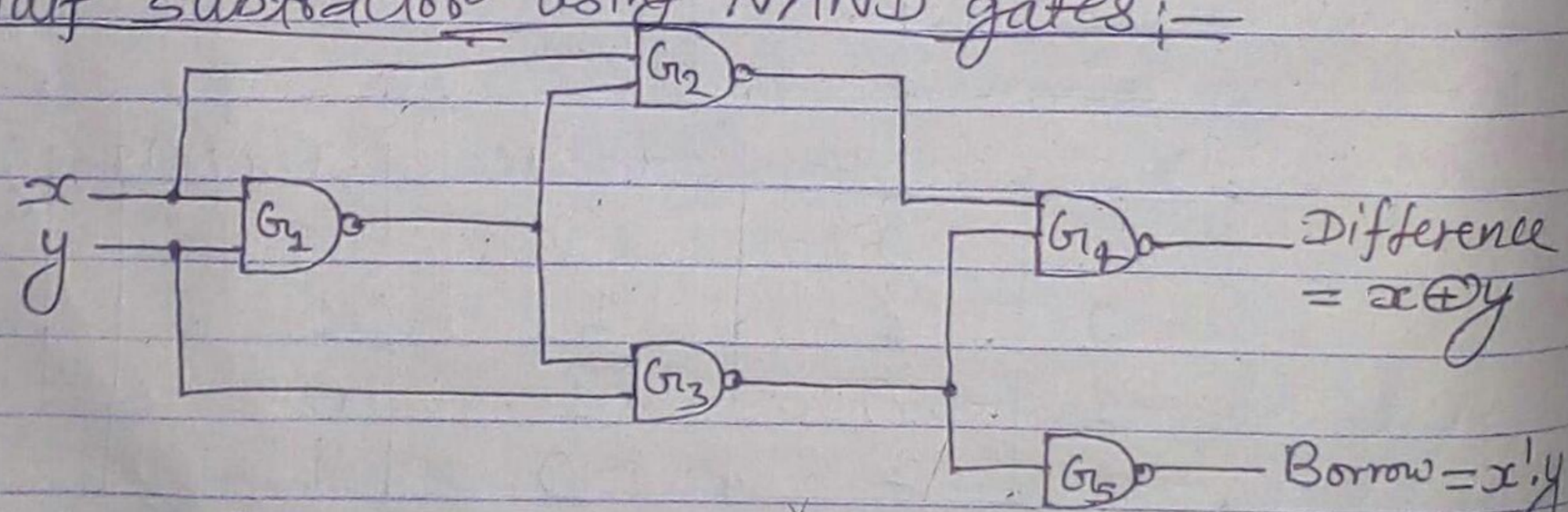


fig. Half subtractor using NAND gates

Half subtractor using NOR gates:-

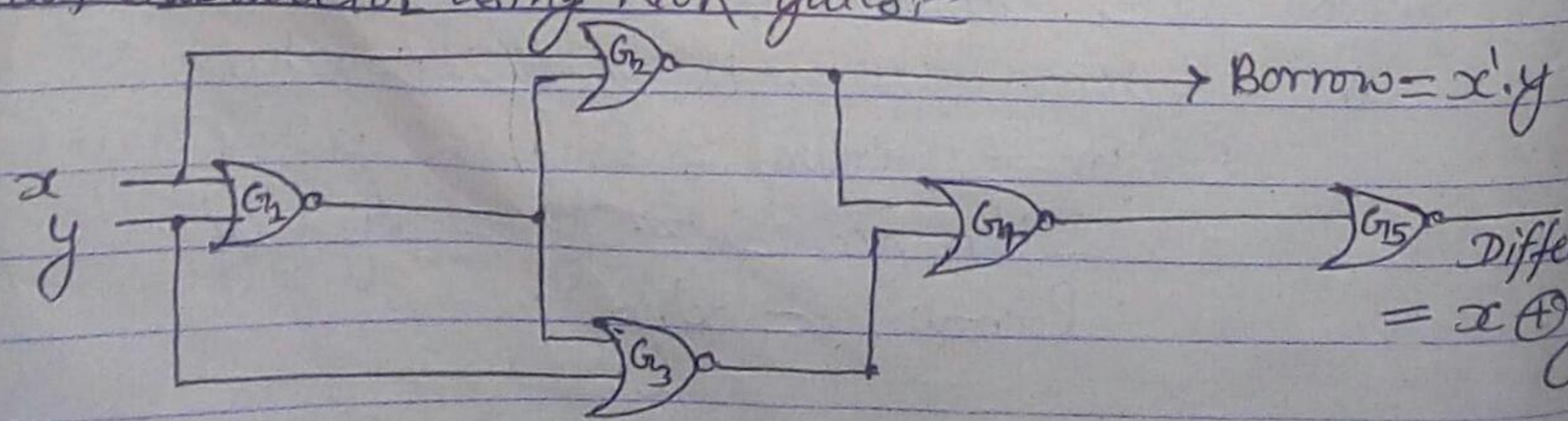


fig. Half subtractor using NOR gates

$x'y'z + x'yz' + x'yz + xyz$
$= x'(y'z + yz') + yz(x' + x)$
$= x'(y \oplus z) + yz$

Full subtractor:-

The subtractor which is used to perform subtraction of ~~two~~ three input bits in combinational logic circuit is called full subtractor. The truth table for full subtractor is as follows:-

x	y	z	difference	borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$F(x,y,z) = \Sigma(1,2,4,7)$ ← difference का नतीजो 1's हरुलाई K-map बाट solve गर्ने

same $= (x \oplus y) \oplus z$ ← $= x'y'z + x'yz' + xy'z' + xyz$ (difference)

$F'(x,y,z) = x'z + x'y + yz$ ← borrow का नतीजा 1's हरुलाई minterms मा लेखी K-map बाट अर्को गरी निकालेमा।

$= x'(y \oplus z) + yz$ (borrow)

Now, the logic diagram can be easily drawn for the full subtractor as follows:-

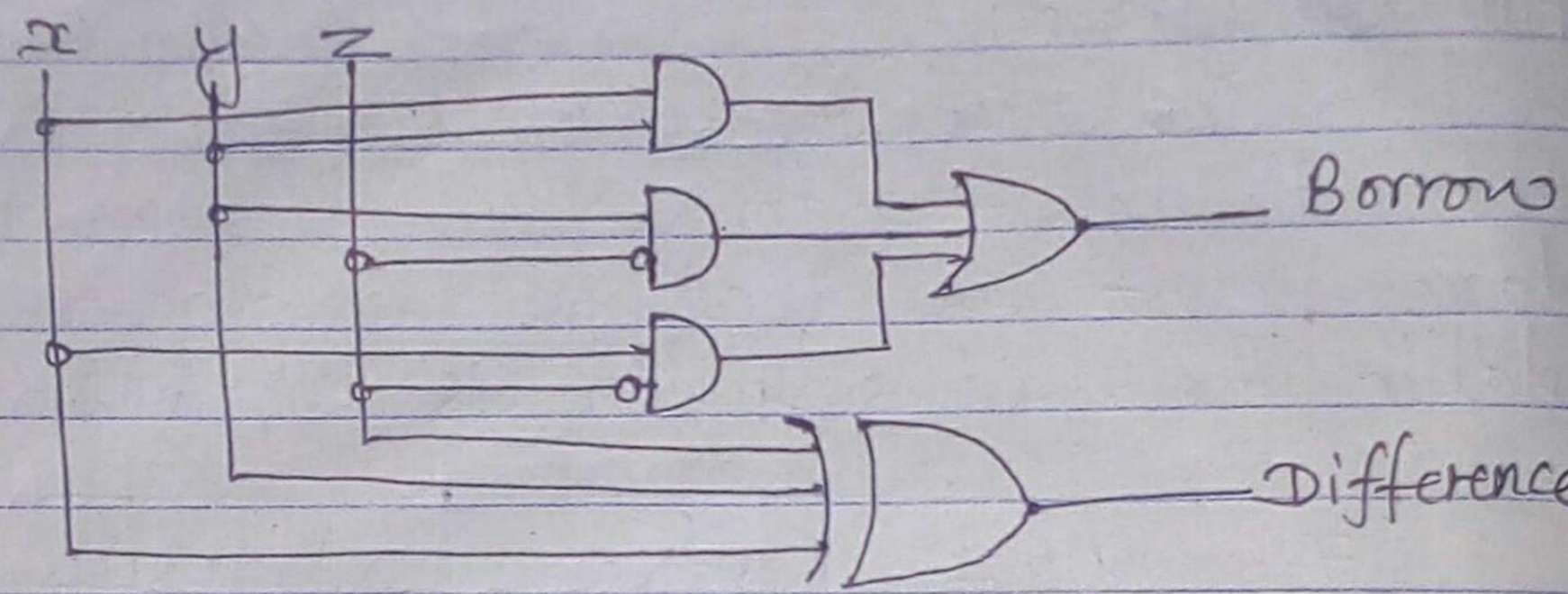


fig. full subtractor circuit.

Code Conversion :-

In different digital systems sometimes the output from one system acts as the input for other system and during that situation we require a converter circuit for converting similar type of input. Hence, any code converter is a circuit that makes the two systems compatible even though they use the different codes.

Here we are considering the converter that convert the BCD codes to excess-3 codes. For this we consider the following table.

Input BCD codes				Output Excess-3 codes.			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Now corresponding to the excess-3 code if we draw the K-map for the values w, x, y & z. Using truth table we can find the simple boolean function as below-

Here using K-map for variable w

from above table analysing 1's in w column

AB \ CD	C'D'	C'D	CD	CD'
A'B'				
A'B		1	1	1
AB	X	X	X	X
AB'	1	1	X	X

$F(A,B,C,D) = \Sigma(5,6,7,8,9)$

9 भन्दा माथि 10 देखि केहि हैन त्यसैले X राखेर group बनाउने Don't care condition मा

$\therefore w = A + BC + BD$

Similarly for x

$$F(A,B,C,D) = \Sigma(1,2,3,4,9)$$

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$		1	1	1
$A'B$	1			
AB	X	X	X	X
AB'		1	X	X

$$\therefore x = B'C + B'D + BC'D'$$

Similarly for y

$$F(A,B,C,D) = \Sigma(0,3,4,7,8)$$

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	1		1	
$A'B$	1		1	
AB	X	X	X	X
AB'	1		X	X

$$\therefore y = C'D' + CD$$

Similarly for z

$$F(A,B,C,D) = \Sigma(0,2,4,6,8)$$

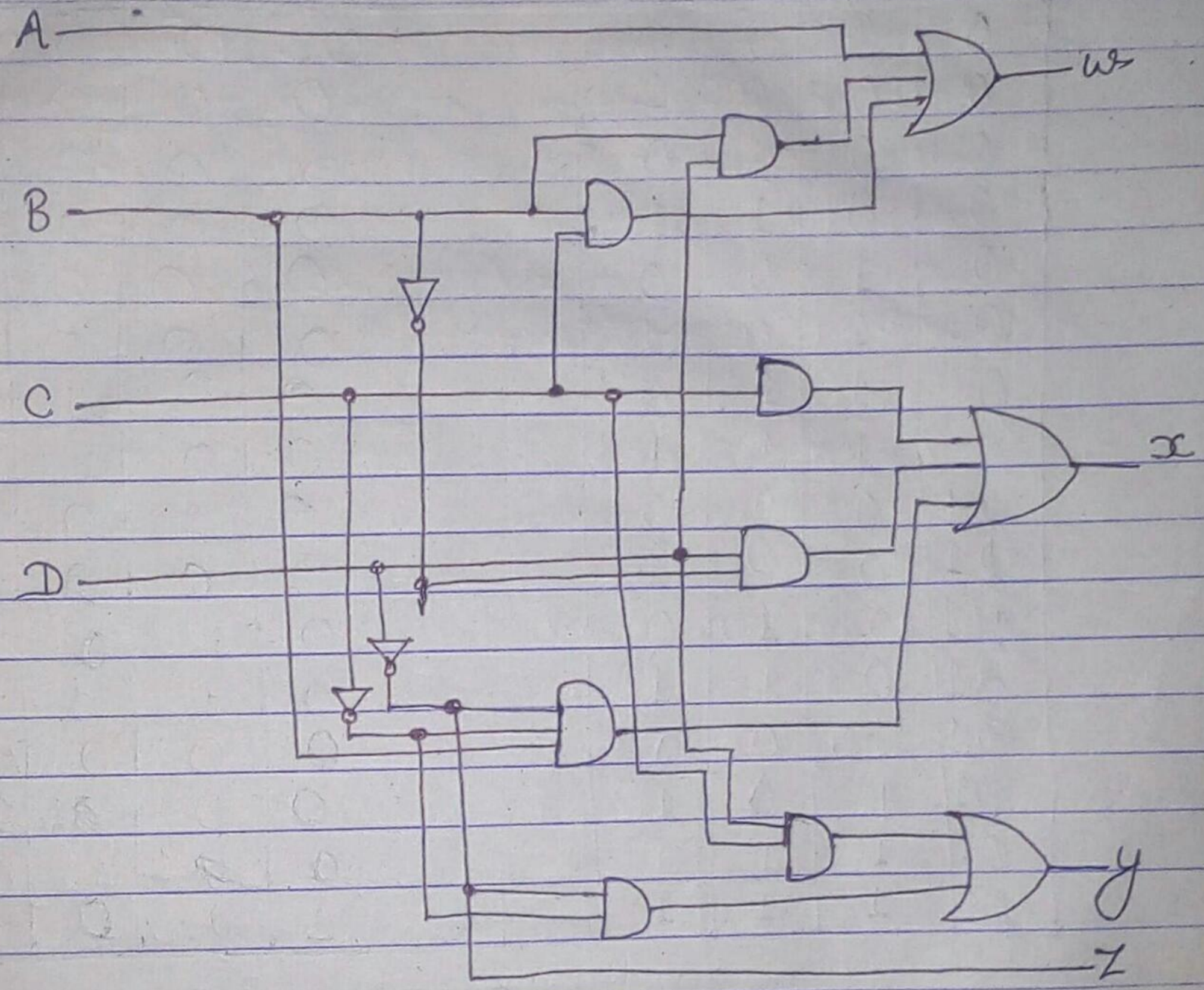
AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	1			1
$A'B$	1			1
AB	X	X	X	X
AB'	1		X	X

$$\therefore z = D'$$

Navneet

Date
Page

Now the circuit diagram can be ~~gi~~ drawn as follows;—



(*) Design the combinational circuit that multiplies 2 bits numbers a_1, a_0 and b_1, b_0 to produce product C_3, C_2, C_1, C_0 as output.

Solⁿ

$2^4 = 16$ combinations will occur. So, constructing following table.

Navneet
Date
Page

Inputs				Outputs.			
a_0	a_1	b_0	b_1	c_0	c_1	c_2	c_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

$\frac{0 \times 0}{0 \quad 0}$

4

6 = 1

~~ALL zeros~~

$C_0 =$

$C_1 = a_0 \times b_0$

$C_2 =$

8 = 8

$C_3 = a_1 \times b$

~~after last line~~

Now, calculating the values of C_0, C_1, C_2 & C_3 on the basis of minterms (analysing 1's) we get.

for $C_0 = \Sigma(15)$

a_0a_1 \ b_0b_1	$b_0'b_1'$	$b_0'b_1$	b_0b_1	b_0b_1'
$a_0'a_1'$				
$a_0'a_1$				
a_0a_1			1	
a_0a_1'				

$C_0 = a_0a_1b_0b_1$

for $C_1 = \Sigma(10, 11, 14)$

a_0a_1 \ b_0b_1	$b_0'b_1'$	$b_0'b_1$	b_0b_1	b_0b_1'
$a_0'a_1'$				
$a_0'a_1$				
a_0a_1			1	
a_0a_1'		1	1	

$C_1 = a_0a_1'b_0 + a_0b_0b_1'$

$= a_0b_0(a_1' + b_1')$

Scanned by TapScanner

$C_2 = \Sigma(6,7,9,11,13,14)$

$a_0a_1 \backslash b_0b_1$	$b_0'b_1'$	$b_0'b_1$	b_0b_1	b_0b_1'
$a_0'a_1'$				
$a_0'a_1$			1	1
a_0a_1		1		1
a_0a_1'		1	1	

$C_3 = \Sigma(5,7,13,15)$

$a_0a_1 \backslash b_0b_1$	$b_0'b_1'$	$b_0'b_1$	b_0b_1	b_0b_1'
$a_0'a_1'$				
$a_0'a_1$		1	1	
a_0a_1		1	1	
a_0a_1'				

$C_2 = a_0a_1'b_1 + a_0b_0'b_1 + a_0'a_1b_0 + a_1b_0b_1'$

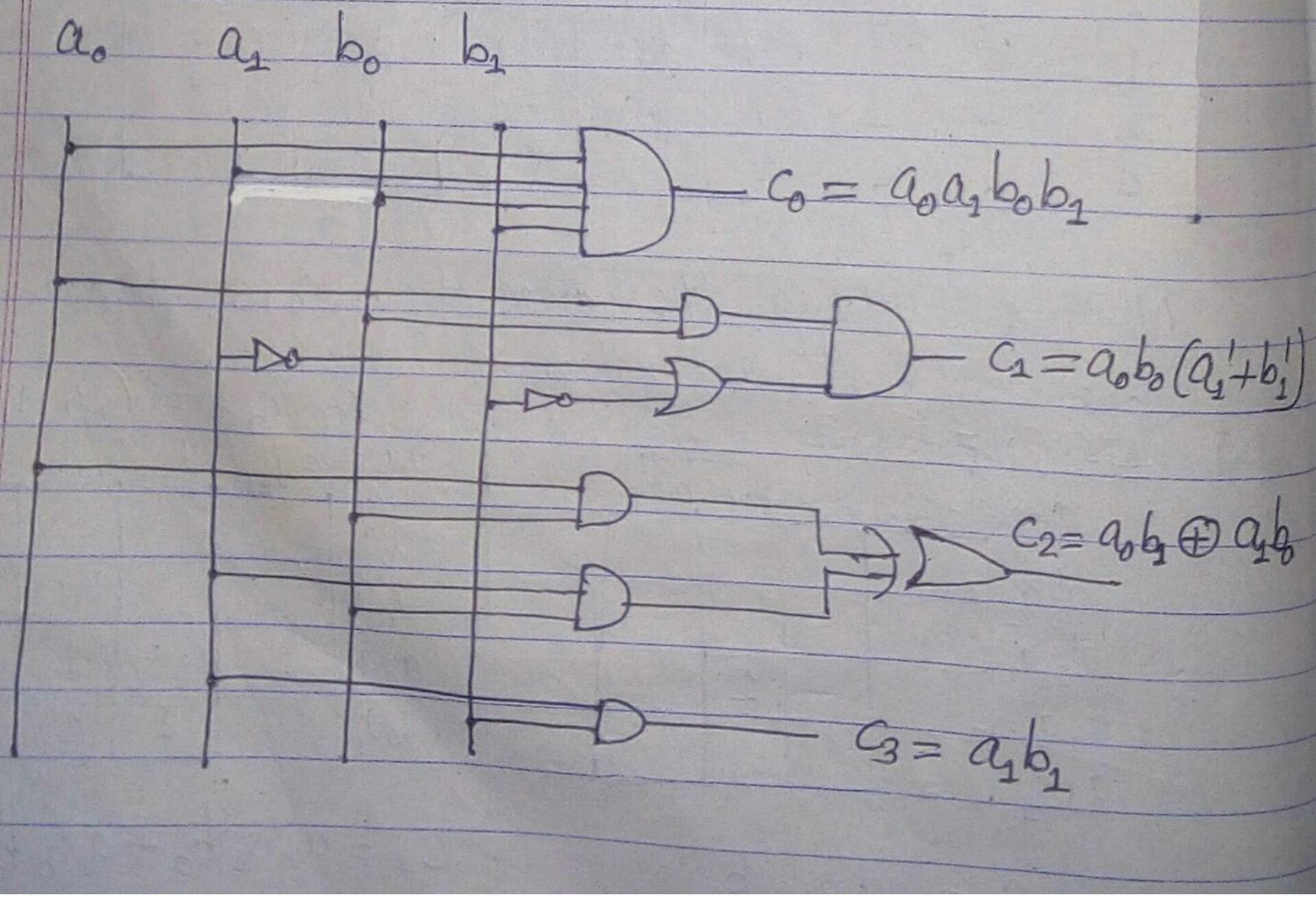
$= a_0b_1(a_1' + b_0') + a_1b_0(a_0' + b_1')$

$= a_0b_1(a_1 \cdot b_0)' + a_1b_0(a_0 \cdot b_1)'$

$= a_0b_1 \oplus a_1b_0$

$C_3 = a_1b_1$

Hence, we can draw the circuit as follows:-



The analysis procedure:-

The analysis procedure comes with the reverse process of design procedure. The analysis can be performed manually by finding the Boolean functions or truth table or by using a computer simulation program. To obtain the output Boolean functions from a logic diagram we proceed as follows:-

Procedures:-

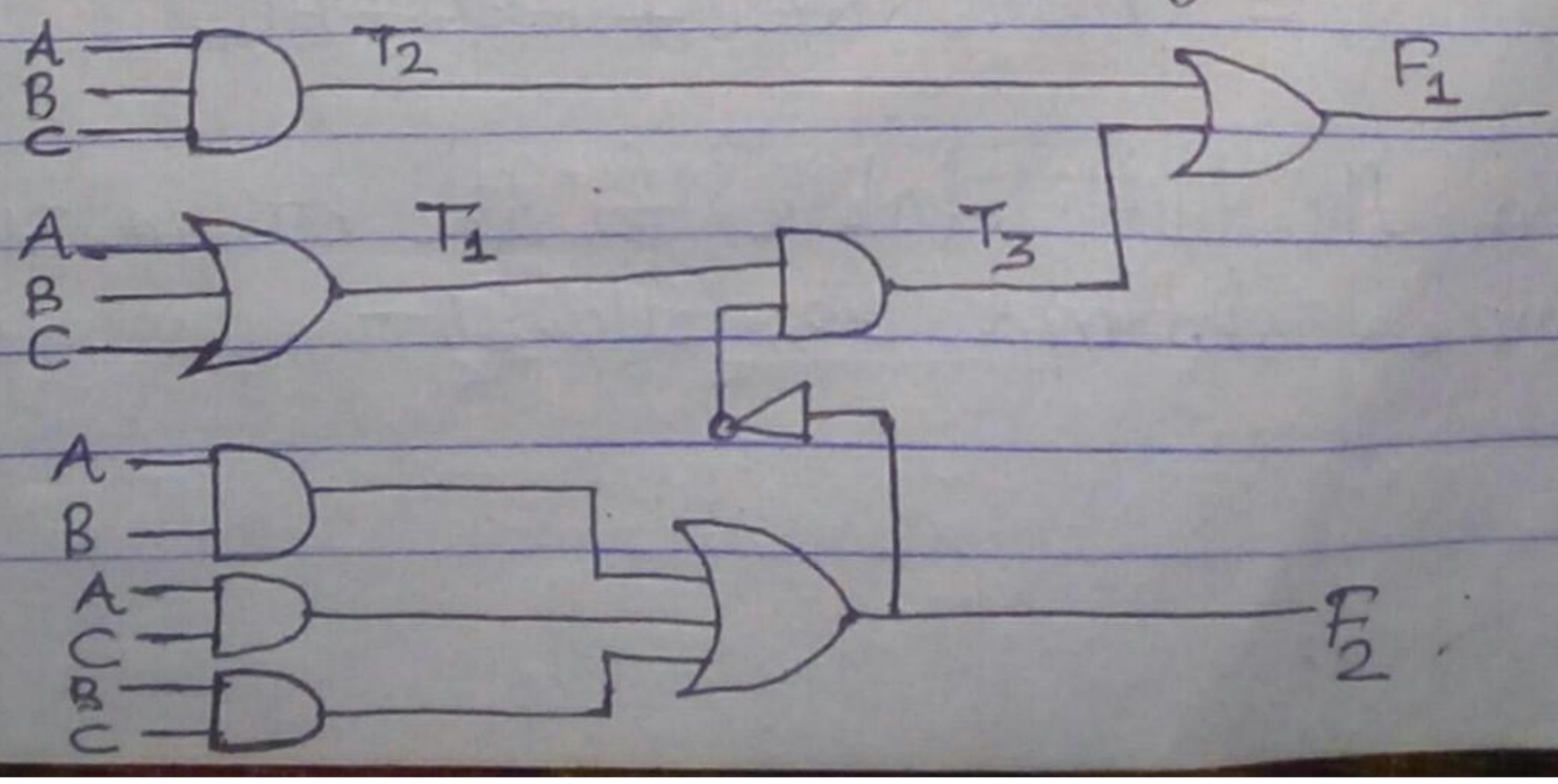
i) Label all gate outputs that are a function of input variables with arbitrary symbols - but with meaningful names. also determine the Boolean functions for each gate output.

ii) Label with other arbitary symbols for those gates which are function of input variables or previously labeled gates and find the Boolean function.

iii) Repeat the process until the output of circuit is obtained.

iv) Finally obtain the Boolean function of the output interms of inputs only.

Consider the following logical Circuit:-



Here, the given circuit has 3 inputs A, B and C and two outputs F_1 and F_2. ~~The Boolean function for those are:-~~ The outputs of gates are the ~~$F_2 = AB + AC + BC$~~ function of input variables and are only F_2, T_1 and T_2. The Boolean function for those are

$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$

Again outputs of the gates which are function of already defined symbols T_3 and F_1 is given by,

$$T_3 = F_2' T_1$$
$$F_1 = T_3 + T_2$$

<u>Now, substituting the values:-</u>

$$F_1 = T_3 + T_2$$

$$\begin{aligned}
\text{or, } F_1 &= F_2' T_1 + ABC \\
&= (AB + AC + BC)' (A + B + C) + ABC \\
&= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\
&= (A' + A'C' + A'B' + B'C')(AB' + BB' + B'C + AC' + BC' + CC') + ABC \\
&= A'B'C + A'BC' + A'B'C + A'BC' + A'B'C + AB'C' + AB'C' + ABC \\
&= A'B'C + A'BC' + AB'C' + ABC
\end{aligned}$$

Now, the truth table for the Boolean function can be obtained as follows:-

Navneet

Inputs			Outputs					
A	B	C	F_2	F_2'	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	1	0	0
1	1	1	1	0	1	1	0	1

In this way we can do the analysis of any circuit and can develop the Boolean function as well as the truth table.

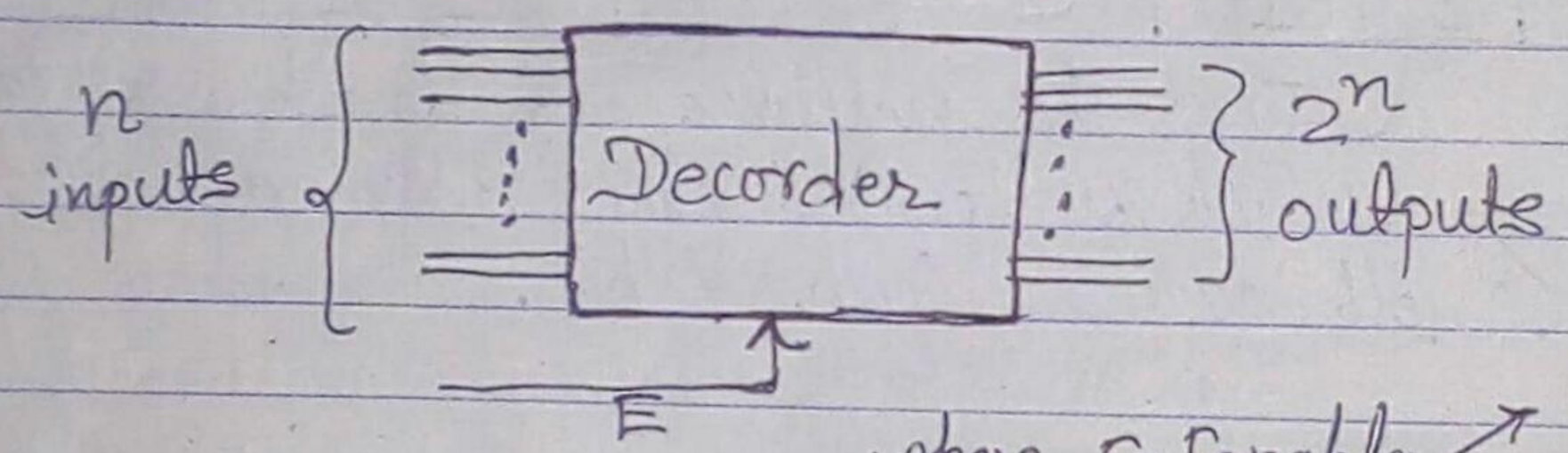
Scanned by TapScanner

Unit = 5

Decoders:-

decode → to discover the meaning of information given in a complicated way.

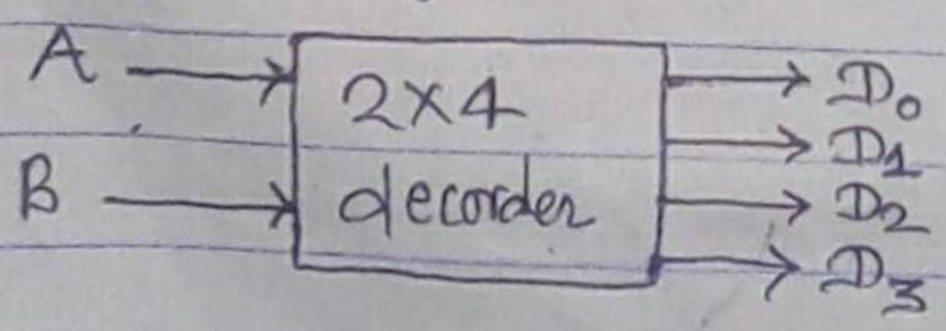
Decoder is a multi-input, multi-output logic circuit which decodes n inputs into 2^n possible outputs. The purpose of this decorder is to generate 2^n or less minterms of n input variables. There is an enabler (E) also which keeps the decorder in ON condition when it is equal to 1 and if enabler is equal to 0 even though we are providing inputs the decoder will provide 0 at the output.



where, E = Enabler
- E = 0 (decorder is disabled)
- E = 1 (decorder is enabled)

2×4 Decorder:-

For 2×4 decorder we have 2 inputs and 4 outputs. Let inputs be denoted by A and B & the outputs be denoted by D_0, D_1, D_2 and D_3. Now the decorder will be as the figure below:-



Scanned by TapScanner

The input output relation for this can be described by using truth table as follows:-

Inputs		Outputs			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Now, analyzing outputs (having 1's) by corrosponding with inputs we get,

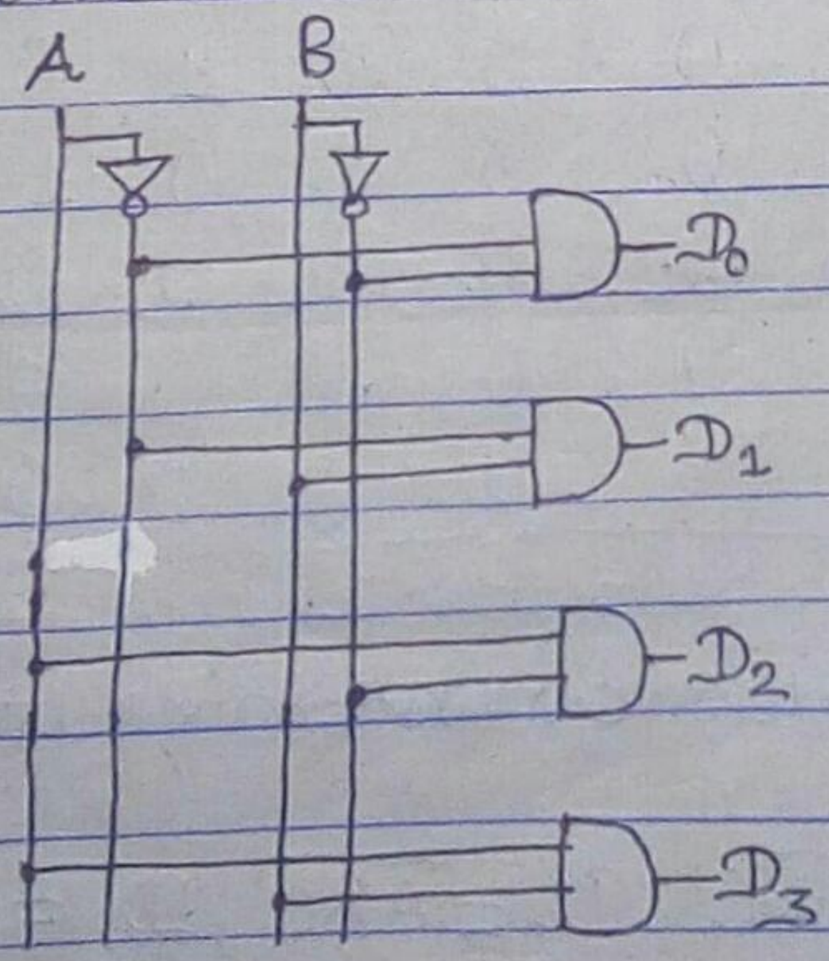
$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

$D_2 = A\bar{B}$

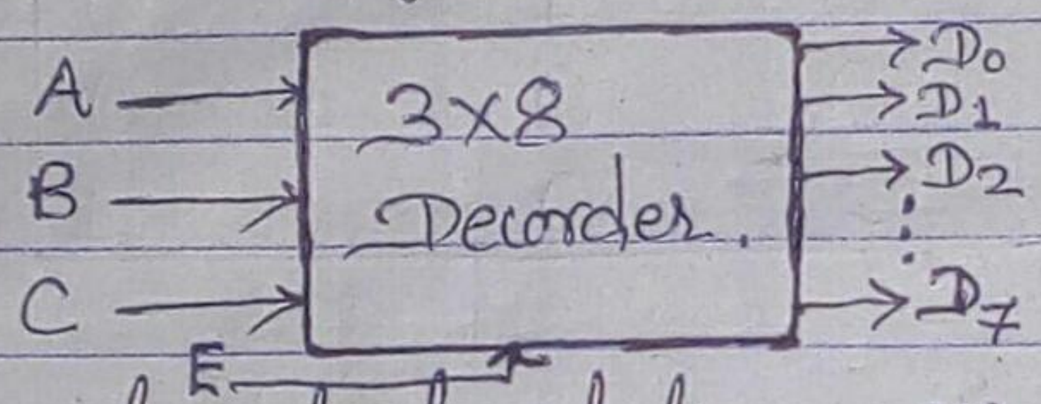
$D_3 = AB$

Now, the logic diagram for these outputs can be drawn as follows:-



3×8 Decorder

For 3×8 decorder we have 3 inputs and 8 outputs. Let these three inputs be denoted by A, B and C & the outputs be denoted by $D_0, D_1, D_2 \ldots D_7$. Now, the decorder will be as the figure below:-



The input output relation can be described by using truth table as follows:-

Inputs			Outputs							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

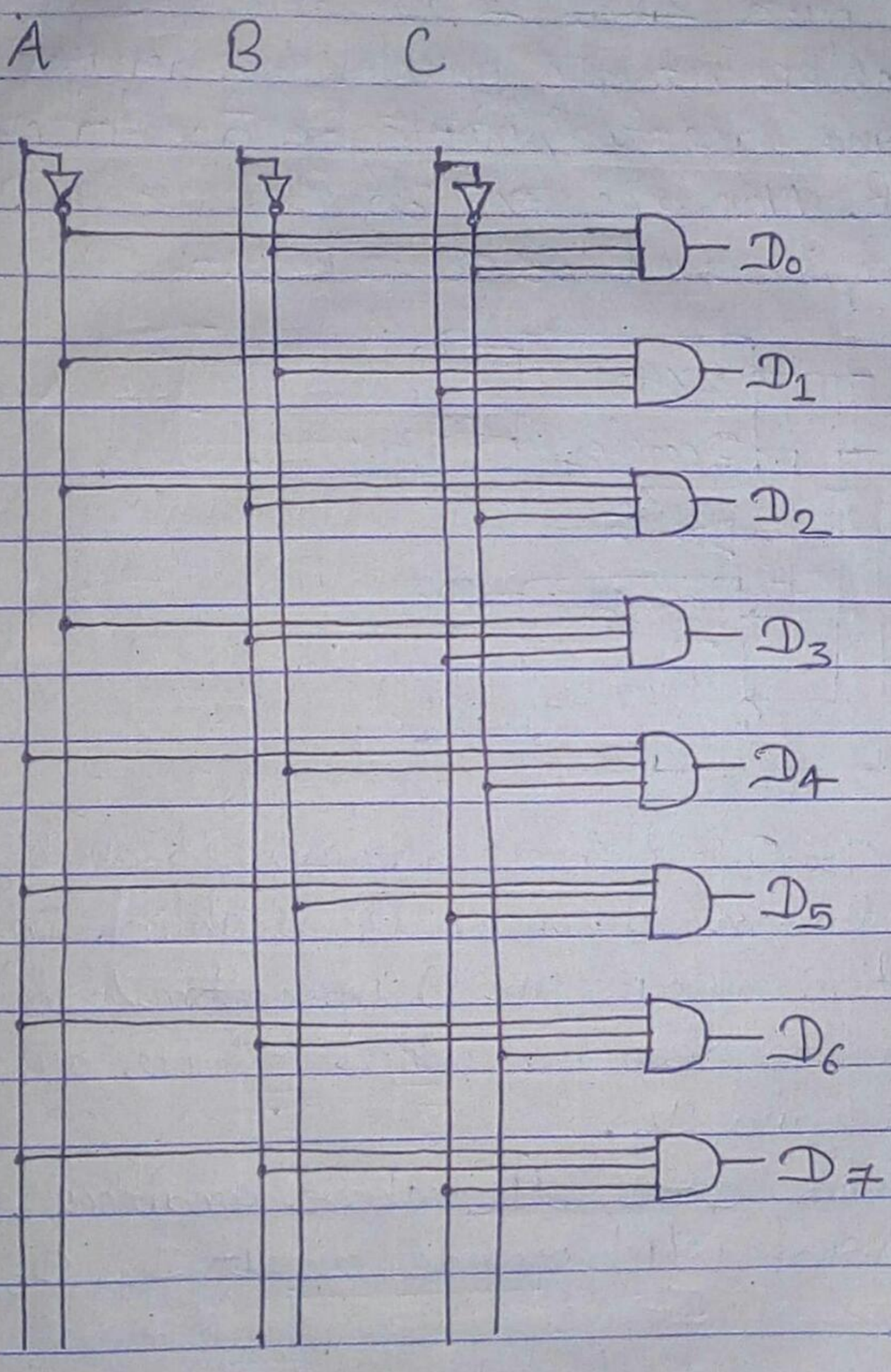
Now, analysing outputs corrosponding to inputs we get,

$D_0 = \bar{A}\bar{B}\bar{C}$, $D_1 = \bar{A}\bar{B}C$, $D_2 = \bar{A}B\bar{C}$, $D_3 = \bar{A}BC$,
$D_4 = A\bar{B}\bar{C}$, $D_5 = A\bar{B}C$, $D_6 = AB\bar{C}$, $D_7 = ABC$

Navneet

Date _____
Page _____

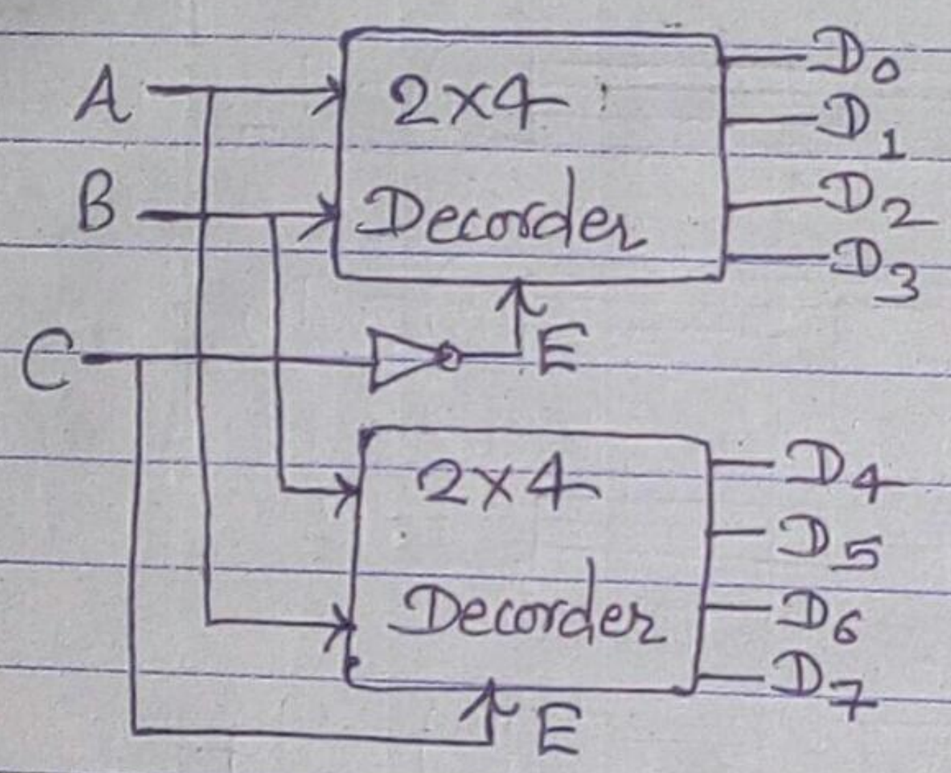
Now, the logical diagram for these outputs can be drawn as follows:-



Note:- For BCD to decimal decorder we use 0-9 as inputs in binary form which contain 4 variables as input. For more information page no. 71 of manual.

Using 2x4 decorder, generate 3x8 decorder.

For generating 3x8 decorder we will need 3 inputs, ~~and~~ 2 as variables and 1 as enabler, Let the enabler be denoted by E and the two other inputs be denoted by A and B. Now, the 3x8 decorder using 2x4 decorder will be as follows:-



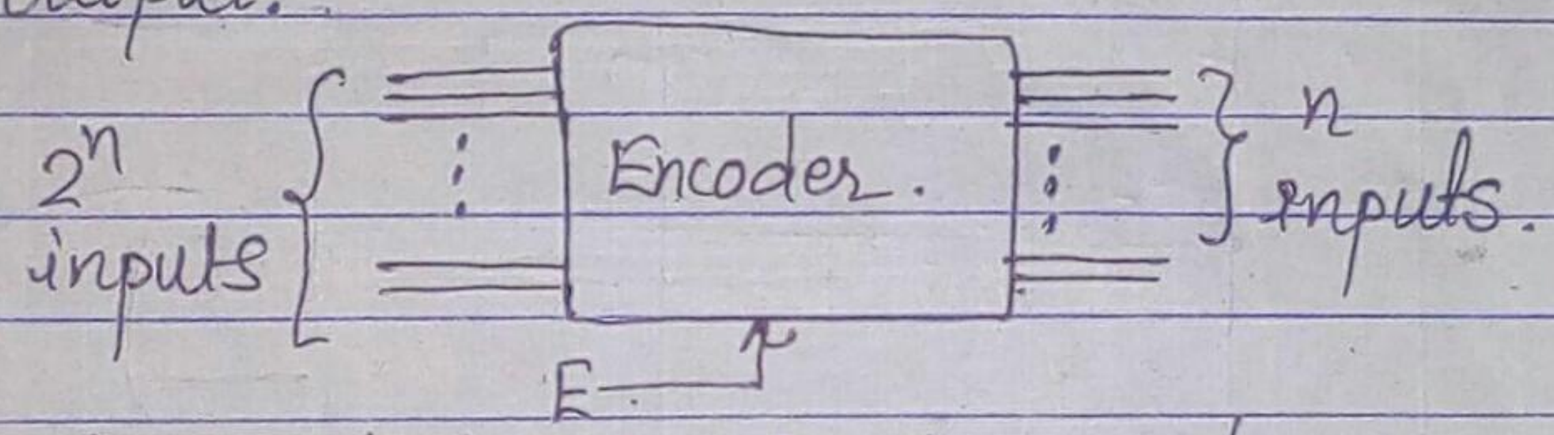
Here, When $C = 0$, the top decorder is enabled and the bottom decorder is disabled. In this condition outputs are 0 in bottom decorder and 1 in top decorder which provide output D_0, D_1, D_2 and D_3.

When $C = 1$, the above conditions are just reversed. which provide output D_4, D_5, D_6 and D_7.

Now we can construct table and ~~diag~~ logic diagram similarly as we did for 3x8 decorder.

Encoders:—

Encoder is a logic circuit which perform the opposite action of decorder. An encoder has a numbers of input lines, only one of which is actived at a time and the output is displayed as a value corrosponding to the actived input. An encoder has 2^n or less input lines and n output lines. So encoder can be defined as the ~~logico~~ combinational logic circuit that converts the logic level "1" data at it's inputs into an equivalent binary code at it's output.



The truth table for the encoder is as follows:-

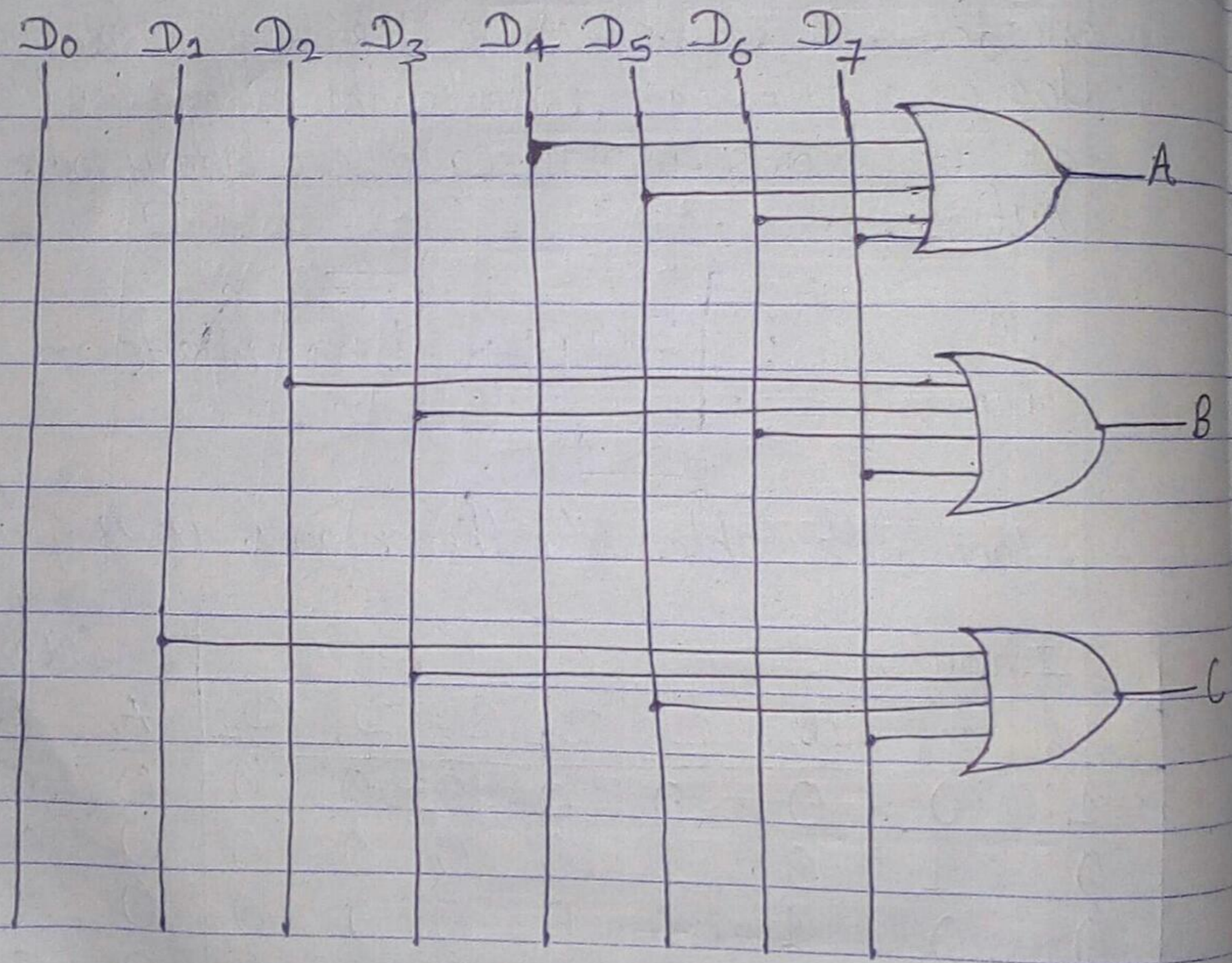
Inputs								Outputs.		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Octal to binary priority encoder

In this for to get output we analyse 1's by ~~using~~ ~~scper~~ column wise

Here, D_0 to D_7 represent octal digits 0 to 7 and ~~X, Y, Z~~ A, B, C represents the binary digits.

Now, the logical diagram can be drawn for this as follows:- A = 4, 5, 6, 7 , B = 2, 3, 6, 7
C = 1, 3, 5, 7.



Disadvantage :- It produces an all 0's output sequence when all input lines are in logic '0' state. This can be overcome by having an additional line to indicate an all 0's input sequence.

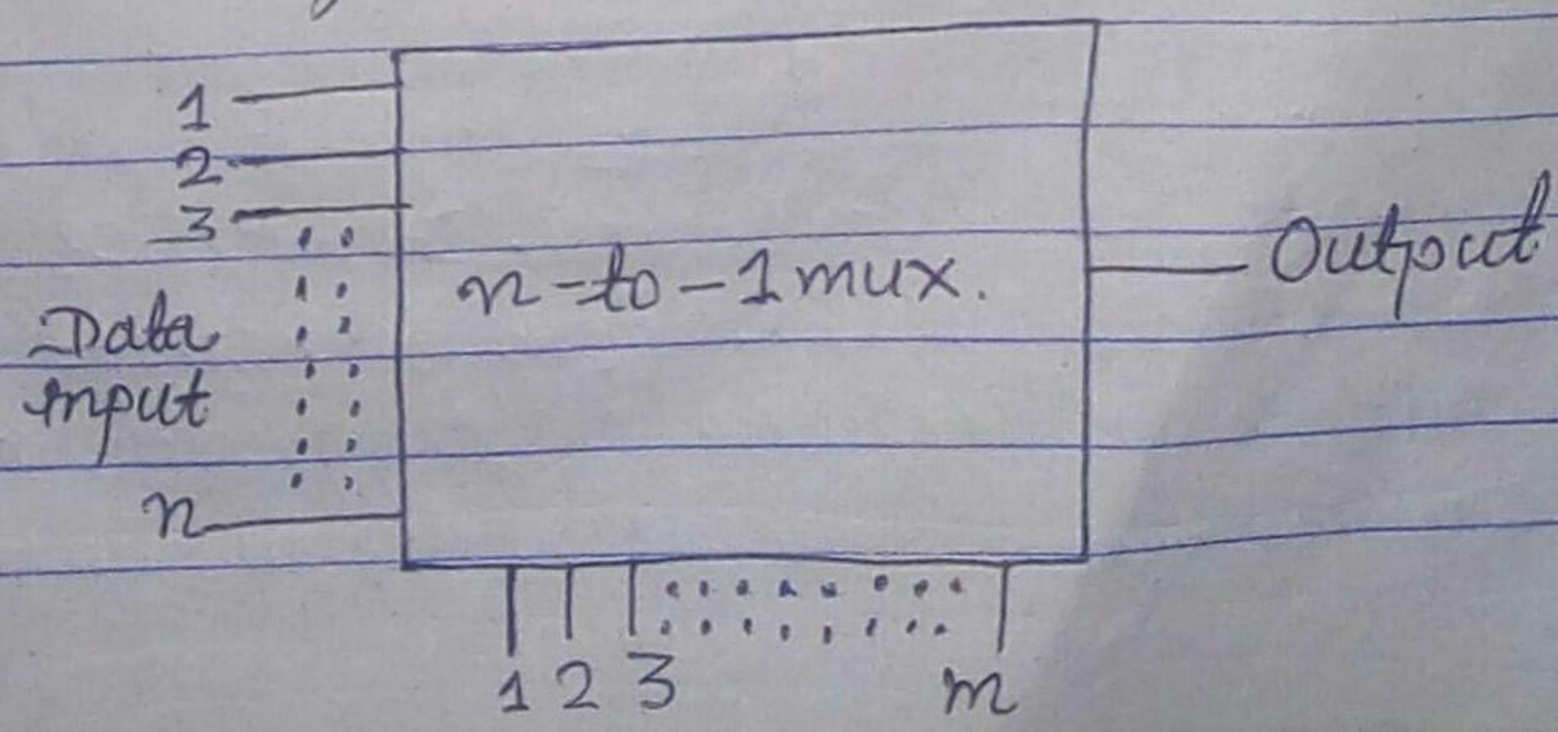
Priority encoder :-

A priority encoder is an practical form of an encoder. The encoders availabe in IC form are all priority encoders.

Multiplexers :-

The combinational logic circuits that accept input from 2^n lines and give the output in a single output line are called multiplexers or also called MUX. It selects binary information frome one of many input lines and directs it to output line. It reduces number of wires, reduces circuit complexity and cost and help in the implementation of various circuits using MUX. It has many inputs but only one output so it is also called many in to one. Normally there are n input line, m selection line and 1 output. We should remember that m control signal can select at most 2^m input signal thus $n \leq 2^m$.

Block diagram

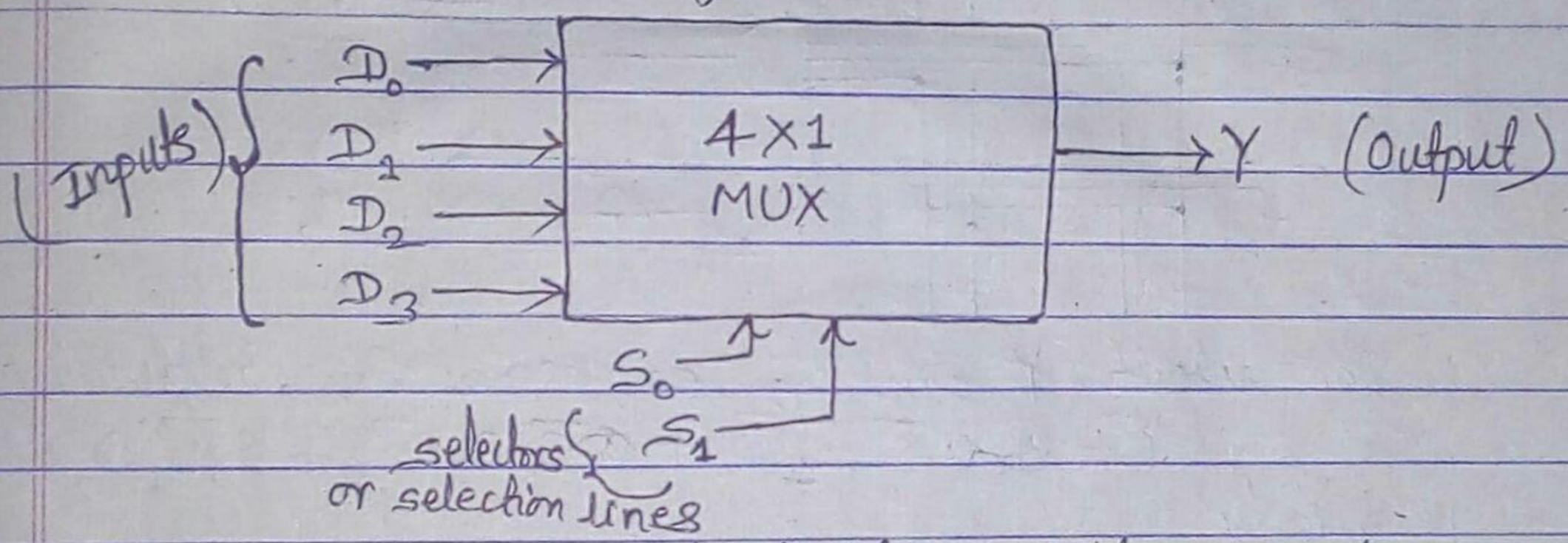


Navneet

2^m 2 selectors needed.
$4 = 2^2 \therefore m = 2$

4 to 1 multiplexers :- (4:1 MUX)

For constructing this multiplexer we ~~weed~~ need 4 data input, let we denote them by D_0, D_1, D_2 and D_3. We get 1 output which be denoted by Y. We need 2 selectors let they be denoted by S_0 and S_1. Now the block diagram for this will be as follows:-

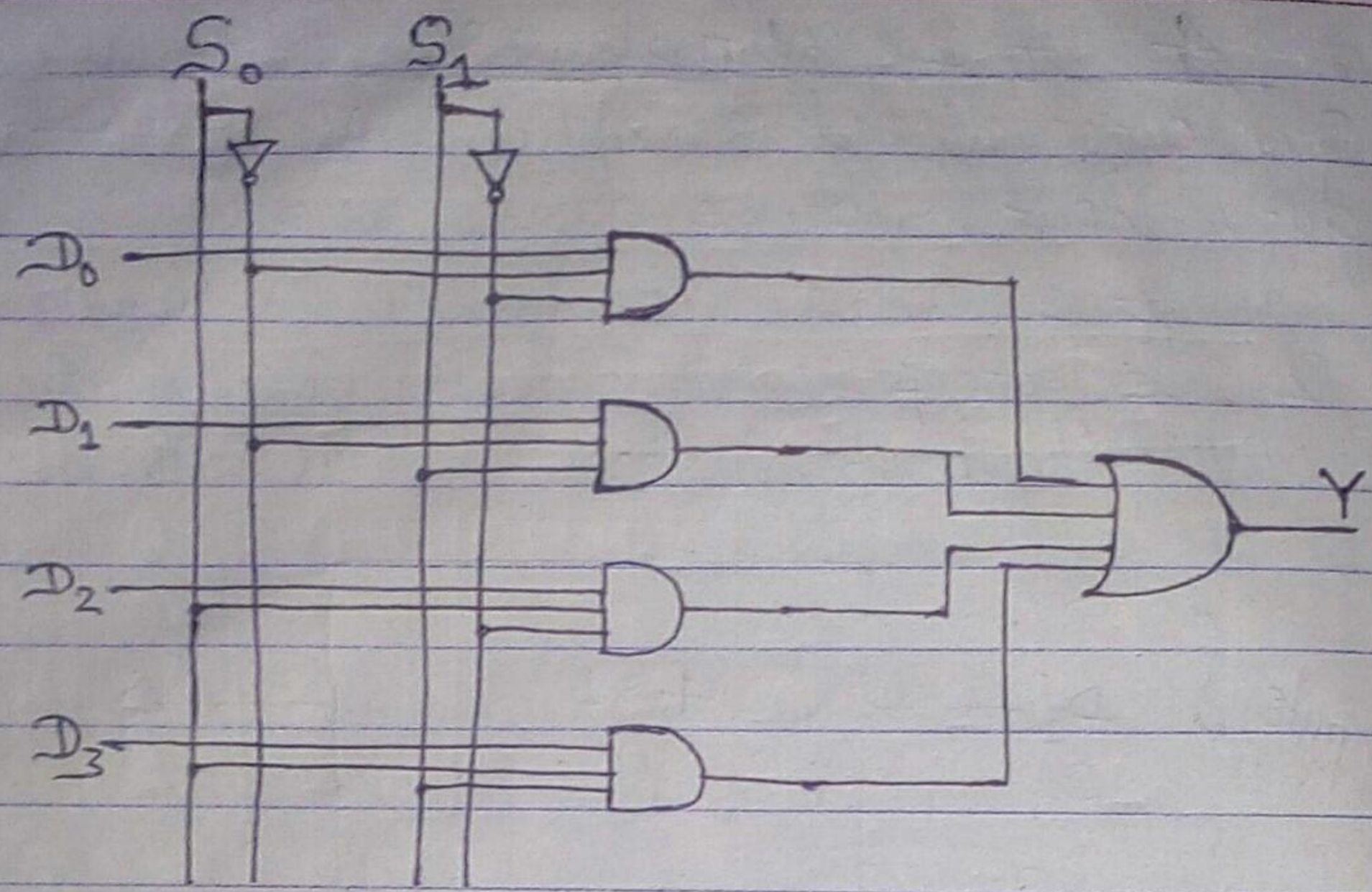


Now we can construct functional table as follows:-

Inputs		Input select	Output
S_0	S_1	Y	
0	0	D_0	$\bar{S}_0 \bar{S}_1 D_0$
0	1	D_1	$\bar{S}_0 S_1 D_1$
1	0	D_2	$S_0 \bar{S}_1 D_2$
1	1	D_3	$S_0 S_1 D_3$

The logical diagram can be drawn for this as follows:-

Scanned by TapScanner



The output Y can be expressed by the functions as

$$Y = \bar{S}_0\bar{S}_1 D_0 + \bar{S}_0 S_1 D_1 + S_0\bar{S}_1 D_2 + S_0 S_1 D_3$$

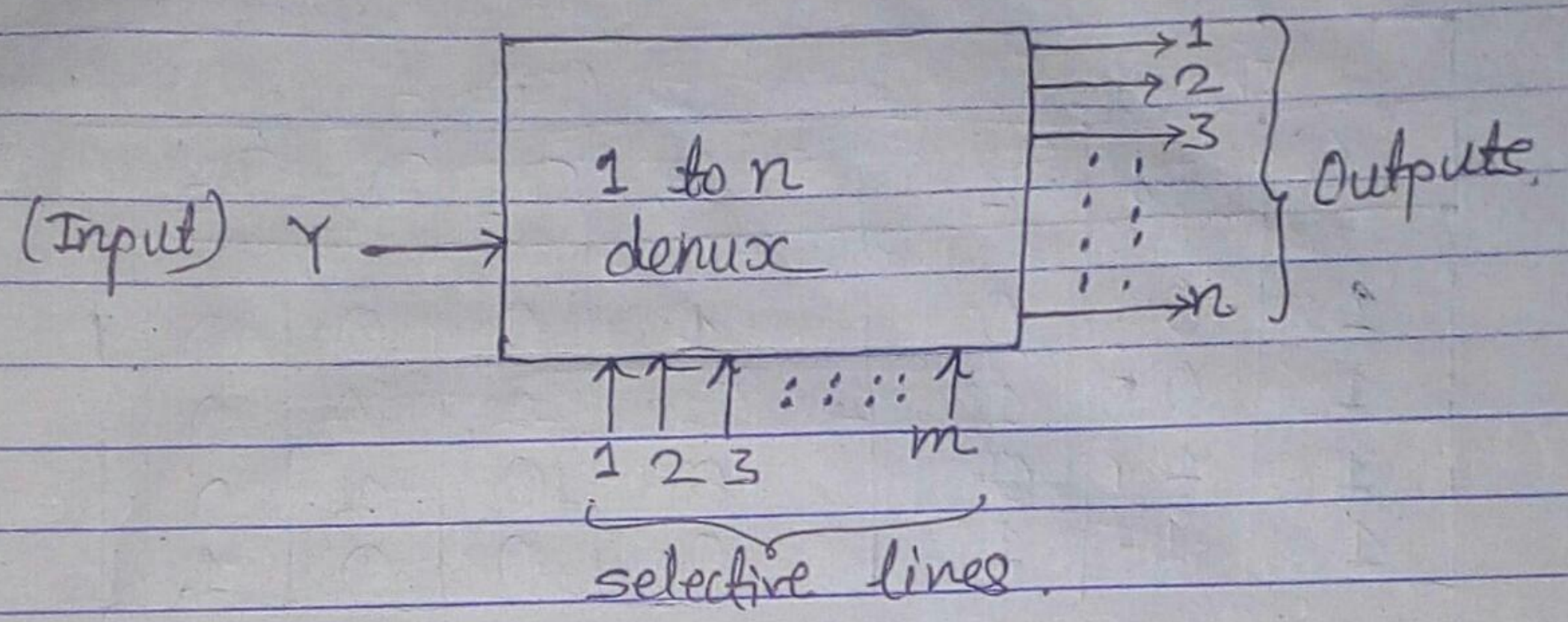
Similarly
In this way we can construct or design for 8:1, 16:1 and 32:1 MUX also.

Demultiplexer (Demux)

The combinational circuit which have only one input but many outputs is called demultiplexer. It works just reverse of the multiplexer. It means 1 into many. It receives information on a single line and transmits this information on one of 2^n possible output line. It is also known as data distributor.

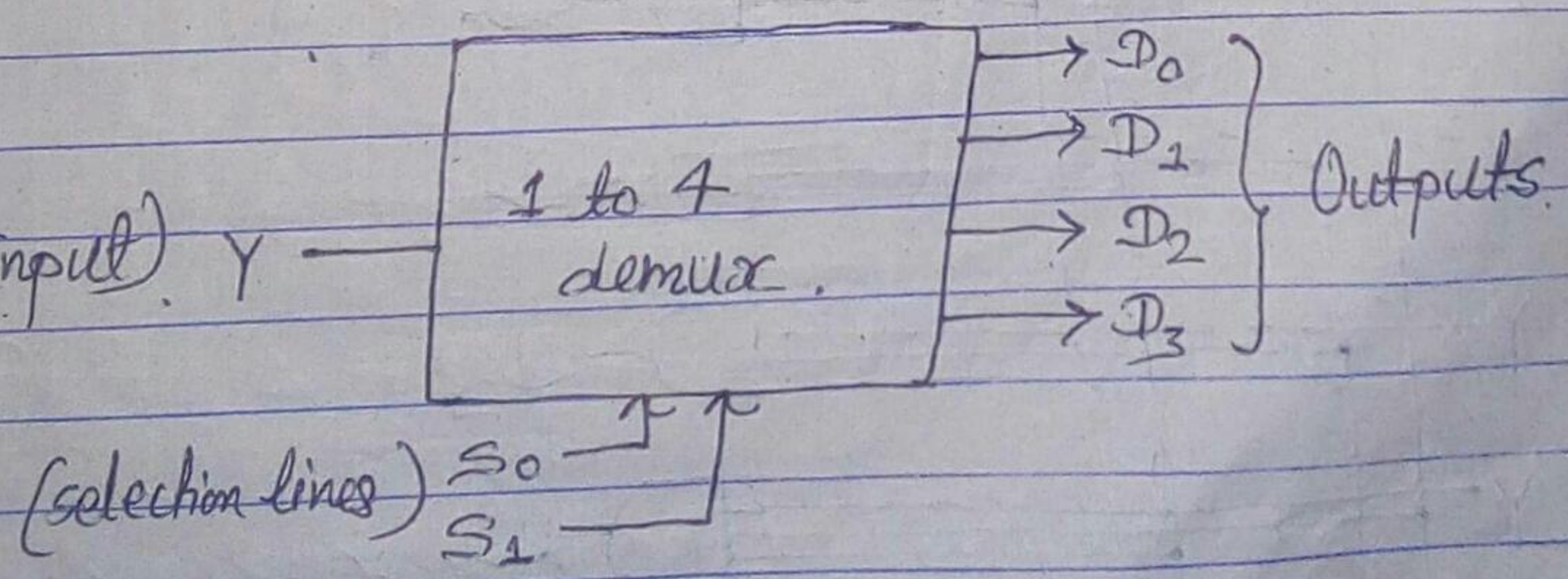
Navneet

Now, the block diagram for this will be as follows:-



1 to 4 demultiplexer (1:4 demux)

For constructing this demultiplexer we need 1 data input let it be Y. Then there will be 4 output let they be D_0, D_1, D_2 and D_3. Here, we know that this will contain 2 selection lines let they be denoted by S_0 and S_1. Now the block diagram for this will be as follows:-



1 to 16 demux

Navneet

Date
Page:

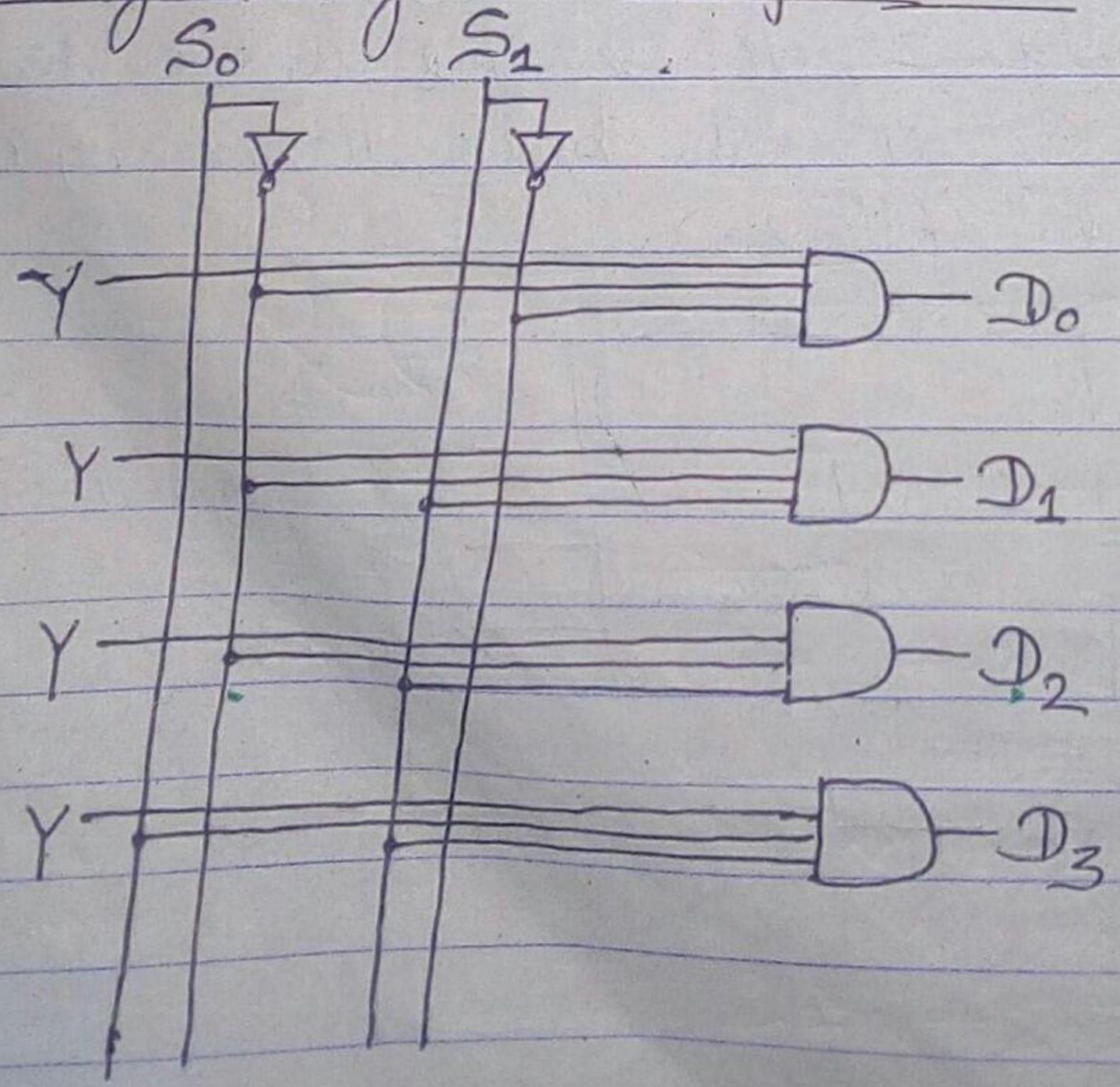
Now we can ~~to~~ know input output relation for this by the help of following truth table.

Inputs			Output			
Y	S_0	S_1	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Boolean Expression for Outputs are

$D_0 = Y\bar{S}_0\bar{S}_1$, $D_1 = Y\bar{S}_0 S_1$, $D_2 = Y S_0 \bar{S}_1$ & $D_3 = Y S_0 S_1$.

The logical diagram is as follows:-



Scanned by TapScanner

BCD Adder :-

A BCD adder is a combinational logic circuit which is used for adding two BCD digits in parallel and generating the sum digit in BCD. This circuit includes the correction logic, Two 4-bit binary adders, 2 AND gates and 1 OR gate.

~~In~~ BCD consists of 4 bit binary number since each input digit does not exceed 9. If it exceeds 9 we add binary 110 to value greater than 9 to get ~~binar~~ BCD sumas in the table below. In the table K is the correction logic which generate 0 upto 15 and 1 after 15. Z_8, Z_4, Z_2, Z_1 and K are binary numbers.

In right side of table there is C which is final correction logic. In the table C, S_8, S_4, S_2, S_1 are BCD sum obtained after ~~co~~ correction. Upto 9 there is no need of correction but after 9 we add 110 binary for correction because output sum can't be greater than $(9+9)+1=19$. 9+9 is for the digits and 1 for the carry.

The logic circuit that detects the required correction can be derived from the table as follows:-

Decimal	Binary numbers					BCD sum				
	K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

1 $\underbrace{\qquad\qquad\qquad\qquad}_{(0-9)}$ Z_8 Z_4 Z_2 Z_1 upto 9

The final condition for correction using K-map ~~we~~ is given by Boolean function

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

The logic circuit of BCD adder for this operation is as follows:-

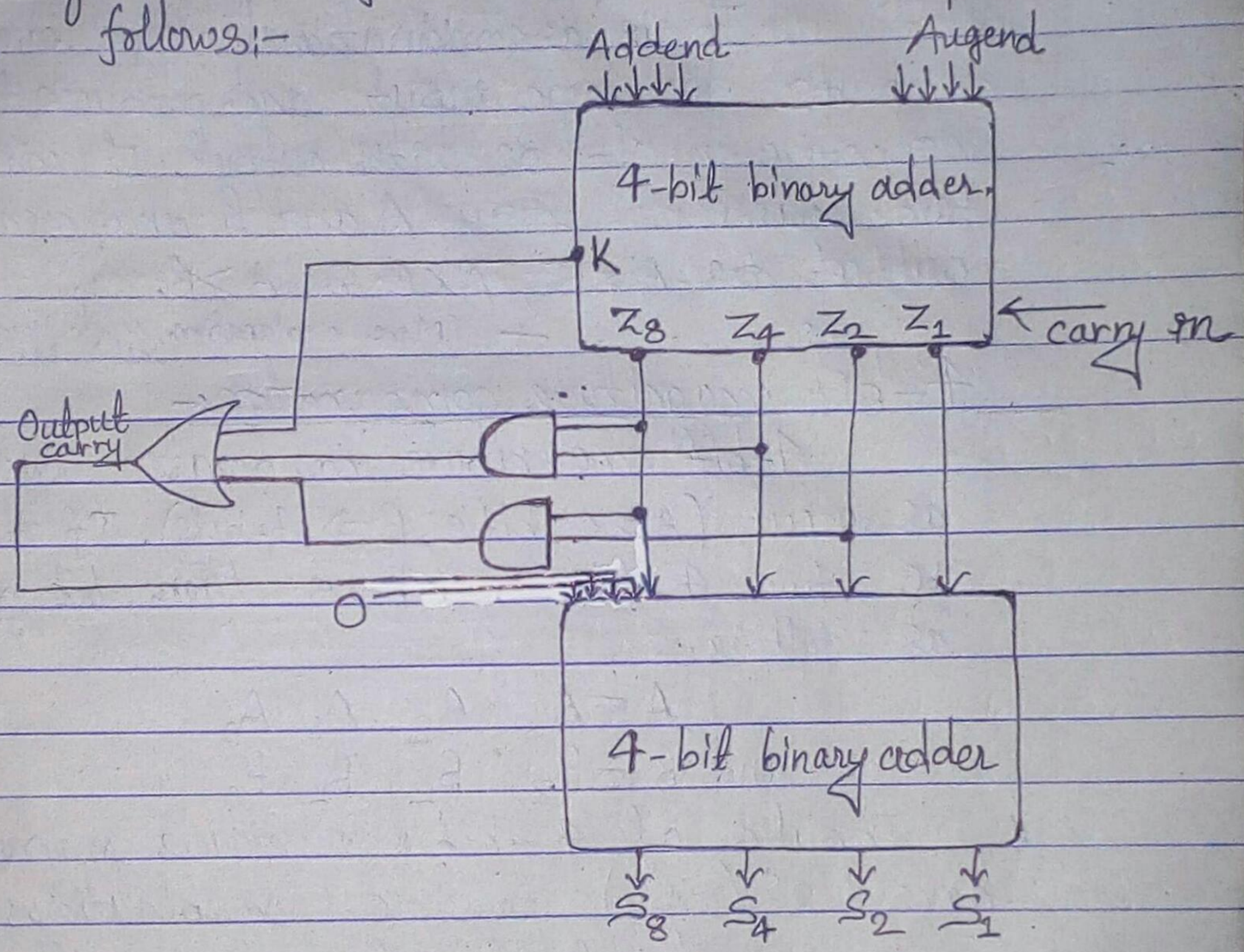


fig. Block diagram ~~of~~ for BCD adder

V.V.I

Magnitude Comparator :-

It is a combinational logic circuit which compares the logic input and produces the respective result as it's output. It compares two numbers say A and B and generate output as $A = B$, $A < B$ or $A > B$.

+ (More explanation with logic diagram and truth table from copied note)

4-bit magnitude comparator :-

4-bit magnitude comparator contain 8-bits as input ($A \Rightarrow$ 4-bits, $B \Rightarrow$ 4-bits). If A and B are two 4-bit numbers then, let denote them as follows :-

$$A = A_3 \; A_2 \; A_1 \; A_0$$

$$\text{and } B = B_3 \; B_2 \; B_1 \; B_0$$

Inputs of 4-bit magnitude comparator have 2^8 (256) possible combinations.

The Boolean function for different conditions $A = B$, $A < B$ and $A > B$ is as follows :-

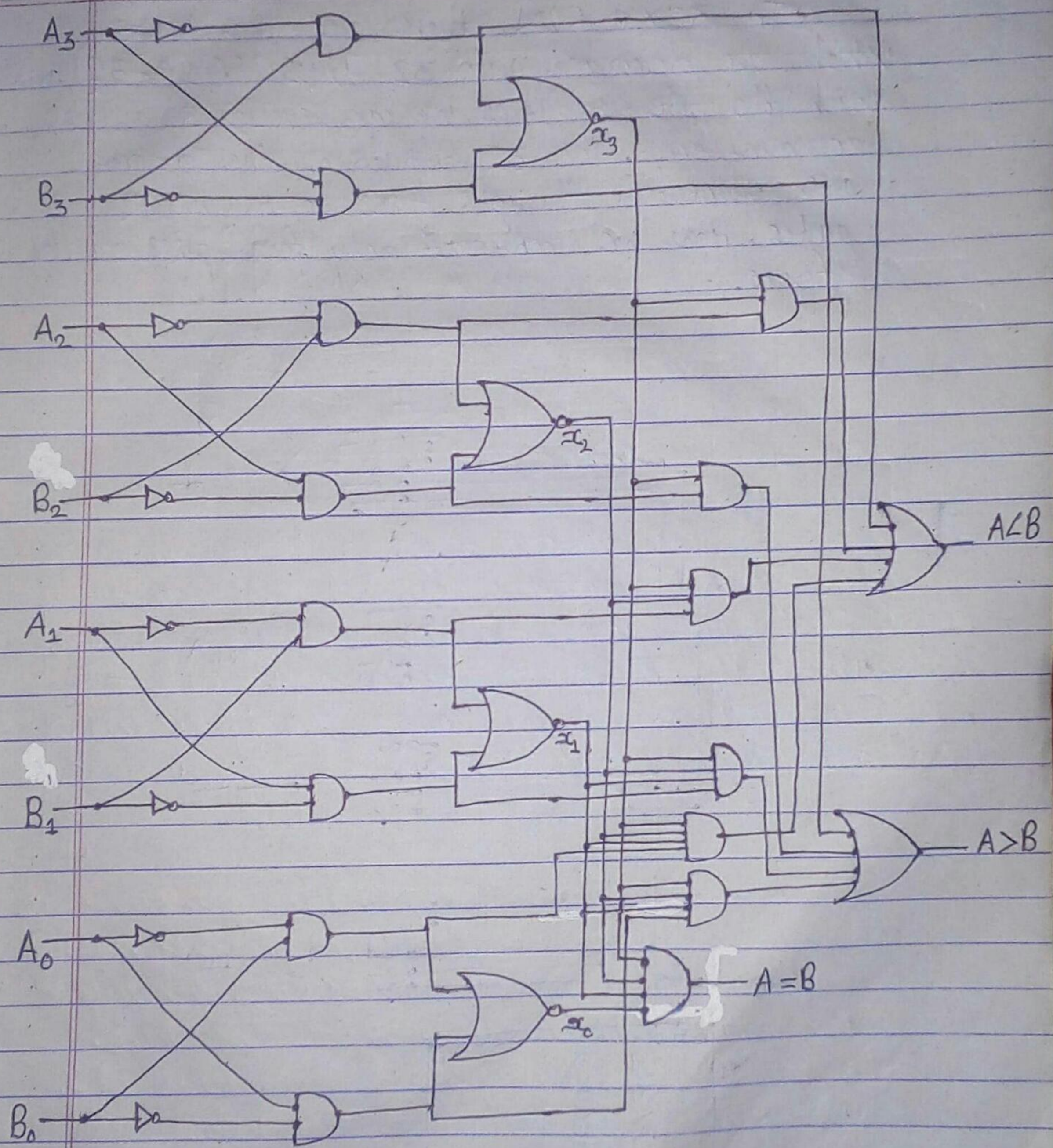
$$(A = B) = x_3 \; x_2 \; x_1 \; x_0$$

$$(A > B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

$$(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

Now, the logical circuit for the 4-bit magnitude comparator is as follows :-

Navneet
Date
Page



Scanned by TapScanner

32×4 ROM

In 32×4 ROM there are five inputs which are decorded into 32 lines ($\because 2^5 = 32$) as with the five variables we can specify 32 address or min-terms. The 32 outputs of the decorder are connected through fuses to each OR gates. Here we represent only four fuses in the figure.

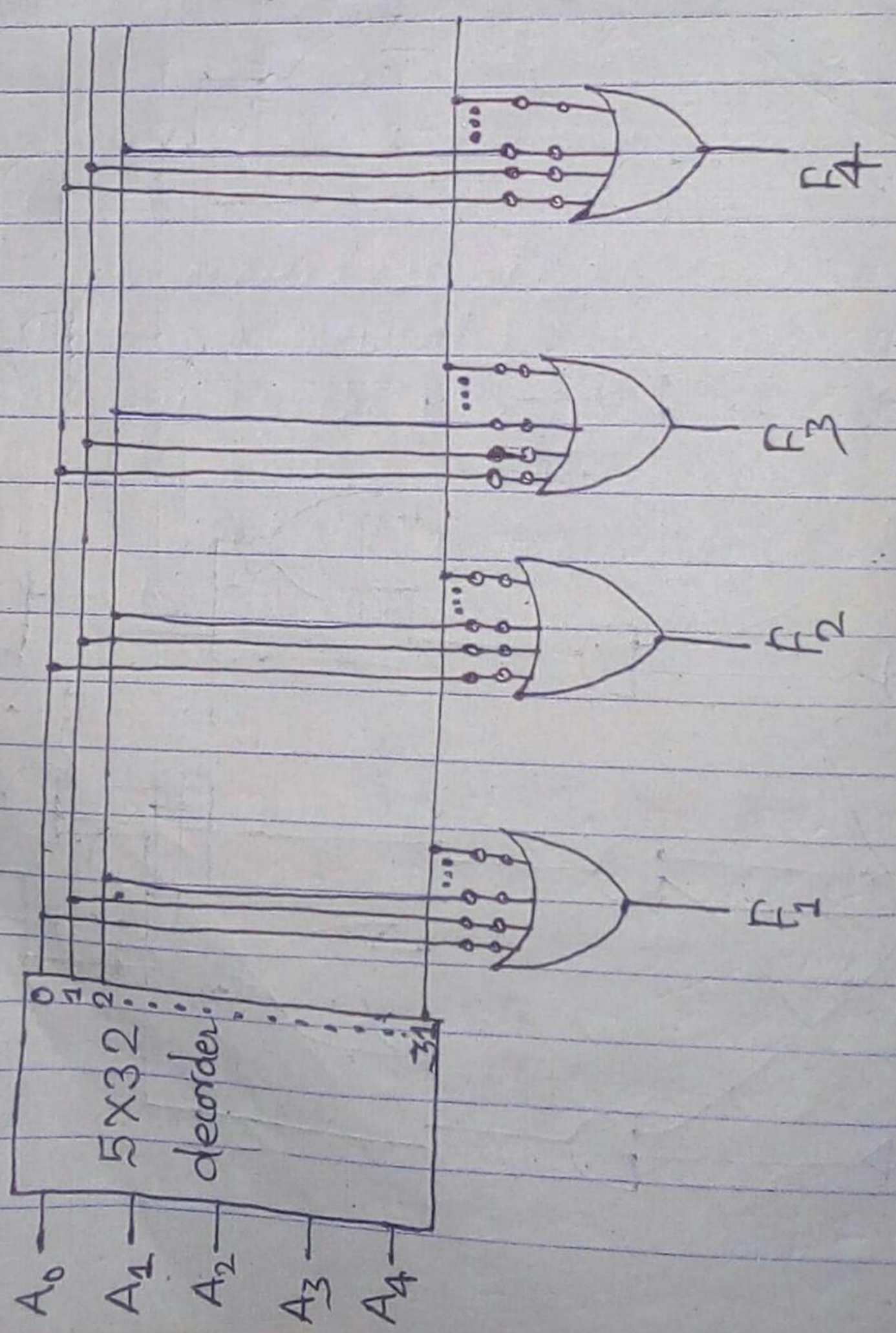


Fig. 32×4 ROM

Navneet
Date
Page

Combinational logic implementation:-

From the diagram of ROM it is clear that each output provides the sum of all minterms of the n input variables.

i) for n input m output combinational circuit we need $2^n \times m$ ROM

ii) The blowing of the fuses is reffered to as programming the ROM.

iii) The information for the required path in ROM is given by the program table.

Let us consider the following truth table.

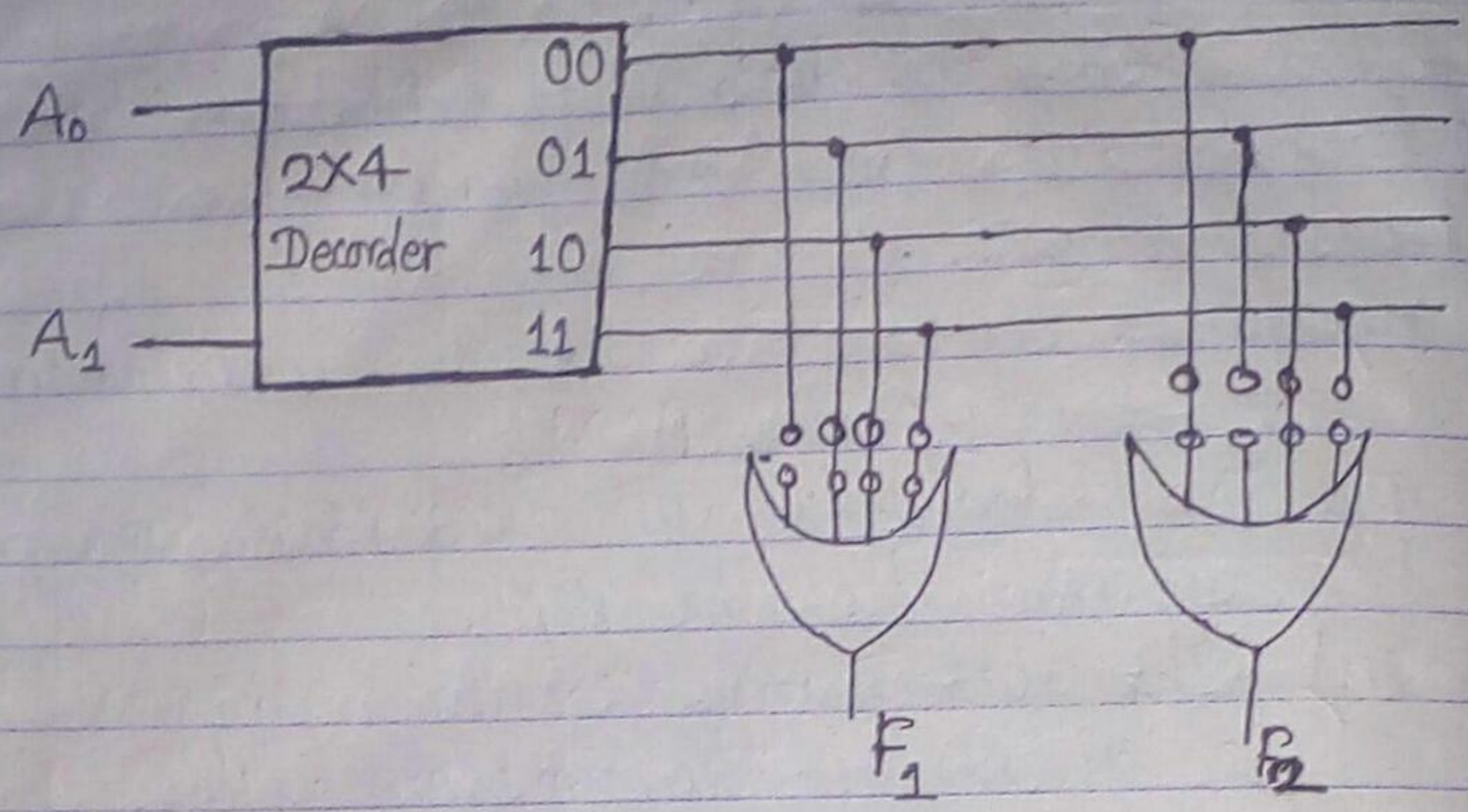
A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

Here we have two inputs and two outputs and the Boolean function for those is as follows:-

$$F_1(A_1, A_0) = \Sigma(1,2,3)$$
$$F_2(A_1, A_0) = \Sigma(0,2)$$

For 4 combination we use 2:4 decorder so input are A_1 and A_0 and output are 00, 01, 10 and 11. The logical diagram for this ROM is as follows:-

Scanned by TapScanner



Types of ROM

The content of the ROM is fabricated by the ROM designer at the time of manufacturing. Depending upon the procedure of storing data and the technique of ~~ear~~ erasing that information the ROM are classified into some categories as follows:-

PROM:-

It is programmable ROM. The user can store programs only once in PROM. The process of making program in PROM is called burning. High voltage (12V) is applied to the fuses to be burnt. The burnt fuses corrospond to 0 and others to 1. This ROM help to achieve the desired relationship between input adresses and stored words.

Navneet

ii) EPROM:-

It is erasable PROM. Informations stored in EPROM can be removed by ultraviolet rays. The UV rays discharge the internal gates due to which informations ~~so~~ stored in it are erased. It is cheaper than PROM because it is re-usable. EPROM have to be removed from computer for re-writing.

iii) EEPROM:-

It is electricity erasable PROM. There are some PROM which are used again by erasing/altering the content present on the chip by the help of electric method. In EEPROM informations can be removed by the electrical signals instead of UV rays. EEPROM chips do not have to be removed from computer for re-writing.

Scanned by TapScanner

Navneet
Date
Page

Programmable Logic Array (PLA) :-

Programmable logic array (PLA) is a LSI component that can be used economically as an alternative to ROM chip where the number of don't care conditions is excessive.

The use of PLA must be considered for combinational circuit that have a large number of inputs and outputs. It is superior to ROM for circuit that have a large number of don't care condition.

Let us taken an example with truth table as below.

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

The simplified function in sum of products are obtained from k-map as below:-

K-map for F_1

A \ BC	$B'C'$	BC'	BC	$B'C$
A'	0	0	0	0
A	1	1	1	0

$F_1 = AB' + AC$

K-map for F_2

A \ BC	$B'C'$	BC'	BC	$B'C$
A'	0	0	1	0
A	0	1	1	0

$F_2 = AC + BC$

Scanned by TapScanner

In this combinational circuits AB', AC and BC, the circuit has three input and two outputs.

As we know that programming the PLA means we specify the paths in its AND-NOT-OR pattern.

A PLA programmer table is as below;-

Product term	Input			Output		
	A	B	C	F_1	F_2	
1	1	0	–	1	–	
2	1	–	1	1	1	
3	–	1	1	–	1	
				T	T	T/C

It consist of three column. The first column lists the product terms numerically. The second column specify the required path between input and AND gates. The third column specify the path between AND gates and OR gates. For each product terms the input are marked with 1, 0 or dash (–). If the variable in the product terms appear in the normal form, the corrosponding variable are marked with 1. If it appear with a complimented then marked with 0 and when it is absent then marked with a dash (–).

The output variable i.e. F_1 and F_2 are marked with 1's for all those product terms that formulate the function. Here, $F_1 = AB' + AC$ so, F_1 is marked with 1's 1 and 2, with dash for product term 3. Each product term that has a 1 in a output column

requires a path for a corrosponding AND gates to the output OR gate. Finally the true (T) output indicates that the link across the output invert remains in place, and a complement (C) specifies that the corrosponding links be broken.

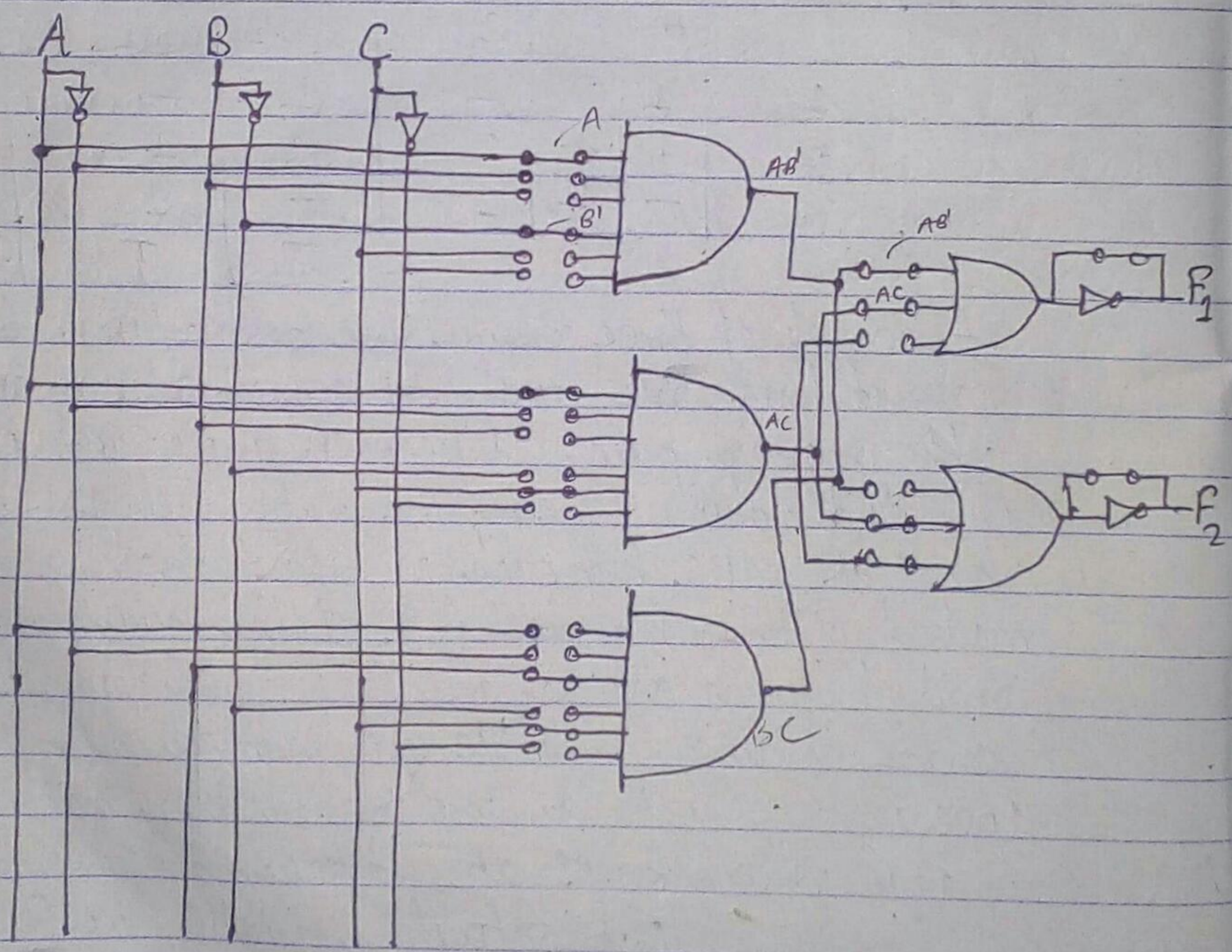


fig. Internal connection of PLA

Differences between ROM and PLA.

ROM	PLA
i> It generates all the minterms as an output of decorder.	i> It does not provide full decoding of the variables.
ii> It uses decorder.	ii> Decorder is replaced by group of AND gates.
iii> Size of ROM is specified by no. of inputs (n) & no. of outputs (m).	iii> Size of PLA is specified by no. of inputs (n), no. of product terms (k), number of outputs (m) & num of sum terms.
iv>. No. of programmed lines $= 2^n * m$	iv> No. of programmed lines $= 2n * k + k * m + m$.

Scanned by TapScanner

Programmable Array Logic (PAL) :-

Programmable array logic (PAL) ~~arti~~ architecture has a programmable AND array at the input and a fixed OR array at the output. The programmable AND array of PAL device is similar to that of PLA device. That is, the number of programmable AND gates is usually smaller than the number required to generate all possible minterms of the given number of input variables. The OR array is fixed and the AND outputs are equally divided between available OR gates. Figure below shows the internal architecture of a PAL device that has two input lines, an array of 4 AND gates at the input and two OR gates at the output.

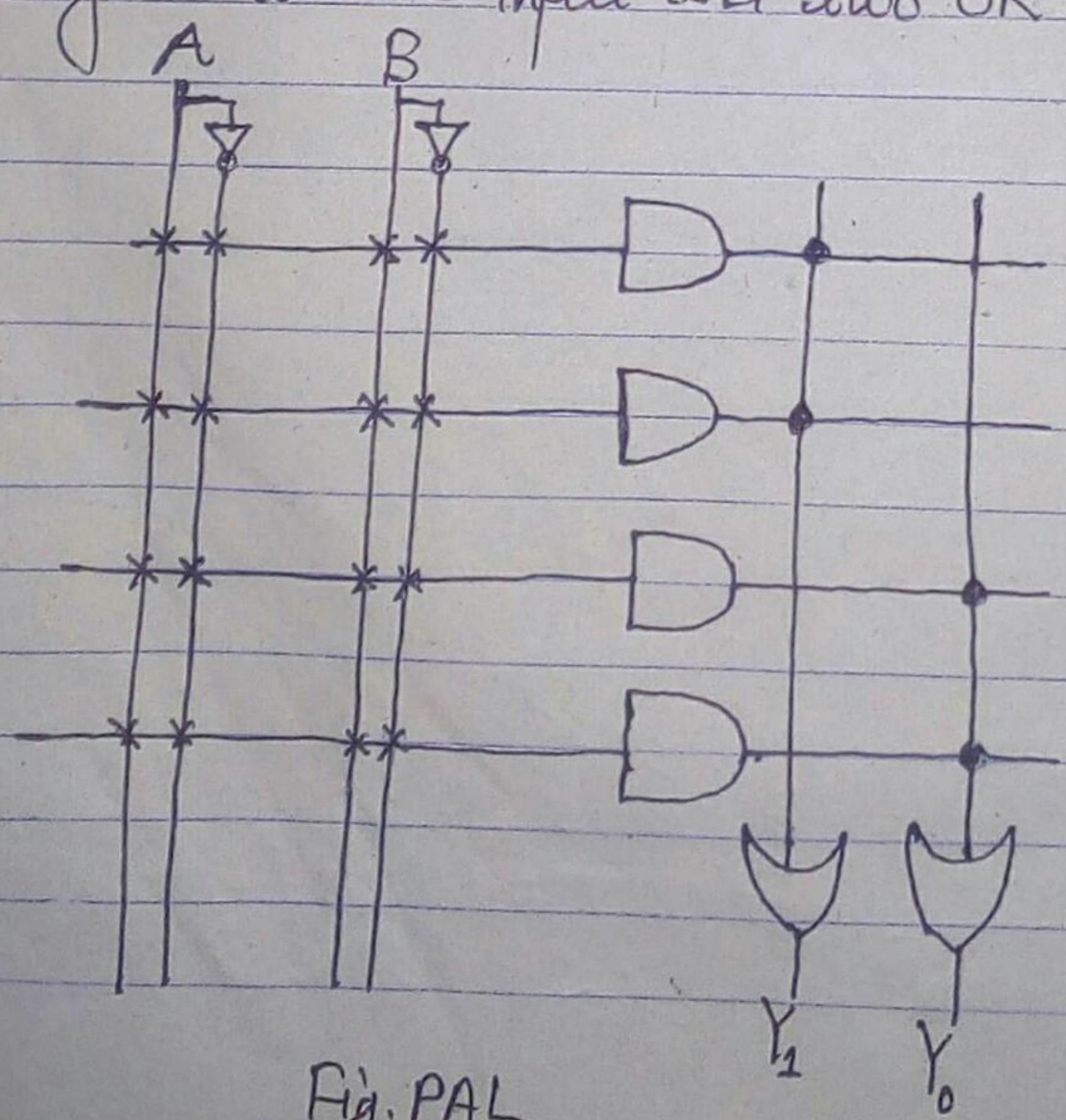


Fig. PAL

Unit-6
Sequential Circuits:-

The combinational circuit with memory element is called sequential circuit. In combinational circuit output depends only upon input but in sequential circuit output depends upon present ~~states and~~ inputs and present state (i.e past output). The information stored in sequential circuit represents present state. The present state and present ~~output w~~ input will define output of next state.

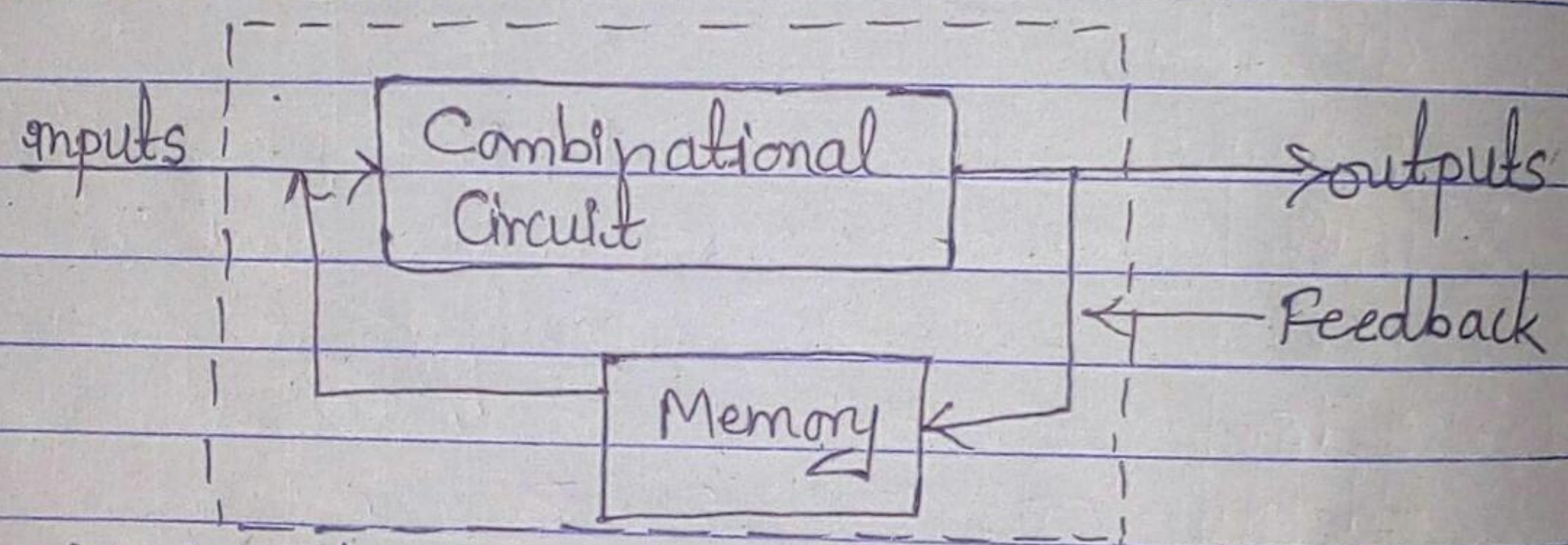


fig. block diagram of sequential circuits.

Types of sequential circuits

i) Synchronus sequential circuits
ii) Asynchronous sequential circuits.

Flip-flops:-

The memory element which is capable of storing one bit of information and is used in clocked sequential circuits is called flip-flop. A flip-flop has two outputs, one for normal value and other for complement value of bit stored in it. A flip flop can store information as

long as power is delivered to the circuit.

A flip flop is also known as bistable multivibrator.

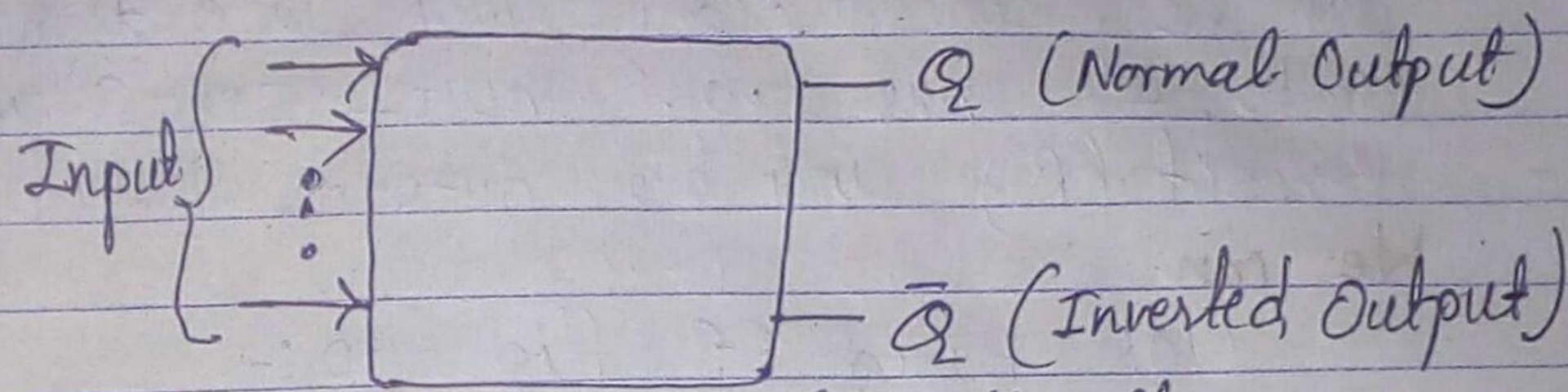


fig. block diagram of a flip flop.

Flip flops are mainly of three types which are as follows:

1) Basic flip flop OR Latch OR (a) RS flipflop:
   - (b) D flip flop.
   - (c) JK flip flop
   - (d) T flip flop
   - (e) JK Master slave flip flop.

2) Clocked flip flop OR Gated flip flop:
   - (a) Clocked RS flip flop.
   - (b) Clocked JK flip flop.

3) Triggered flip flop:- { Clock pulse positive and negative edge triggering }
   - (a) Edge triggered D flip flop

10 → set
01 → reset
00 → illegal/Invalid [NOR] → [NAND]
11 → illegal/Invalid [NAND] → [NOR]

1(a) RS flip flop : (This is latch) -

It is a basic flip flop. We can design other flip flop by using RS flip flop. It can be constructed by NOR gate and NAND gate. It has two input - set (S) and reset (R) and two outputs Q and $\overline{Q}$.

NOR Based RS flip flop :-

NOR 00→1

first set ग्रा input C analyse 1423 inverse

No inverse 1RQ

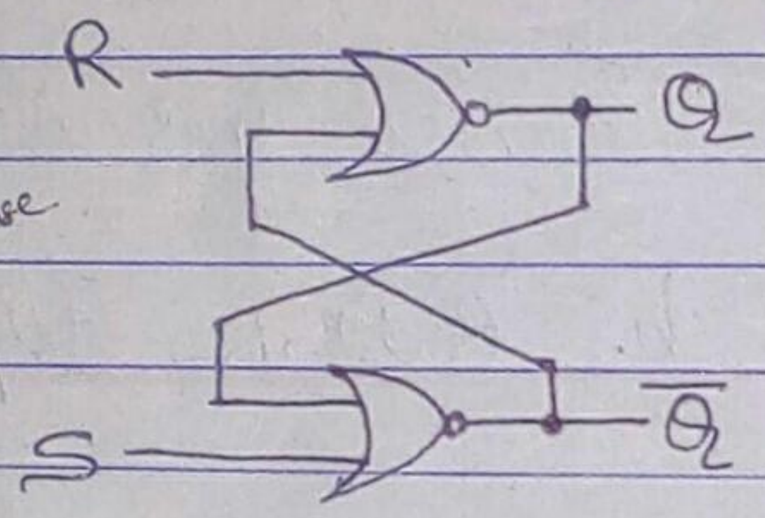


fig. Basic flip-flop circuit using NOR gates.

Truth table.

Inputs		Outputs	
S	R	Q	$\overline{Q}$
0	0	0	1 (reset)
0	1	0	1 (No change)
1	0	1	0 (set)
1	1	0	0 (invalid)

inverse

NAND Based RS flip flop:

NAND 110 1824 inverse

No inverse 4RQ

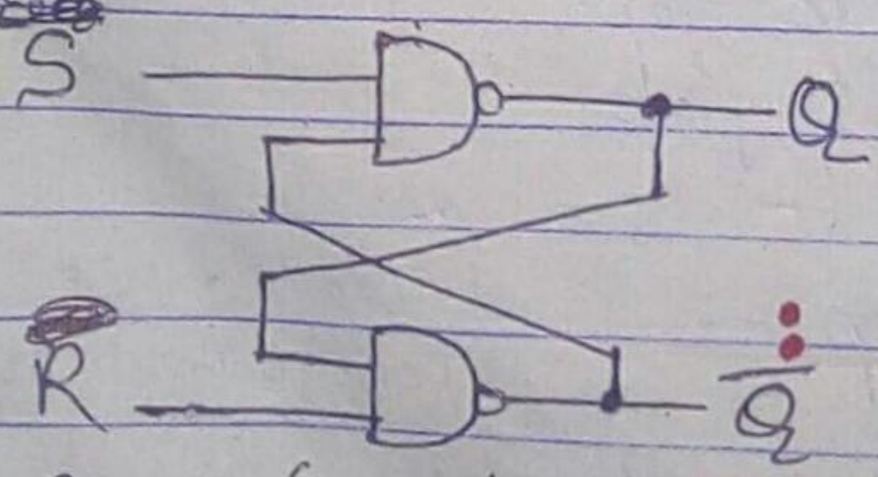


fig. Basic flip-flop circuit using NAND gates

Truth table.

Inputs		Outputs	
S	R	Q	$\overline{Q}$
0	0	1	1 (Invalid)
0	1	1	0 (set)
1	0	0	1 (reset)
1	1	0	1 (no change)

inverse.

Disadvantages

i> It requires two input signal to derive the flip flop.
ii> When both input signal is high, we get illegal condition.

2(a) Clocked RS flip flop :-

We can control the operation of RS flip flop by enabled clock signal. ~~SR f~~ RS flip flop contain the two AND gate at the input of S and R terminal. RS flip flop can be made using ~~bothat~~ both NAND and NOR gates as below:-

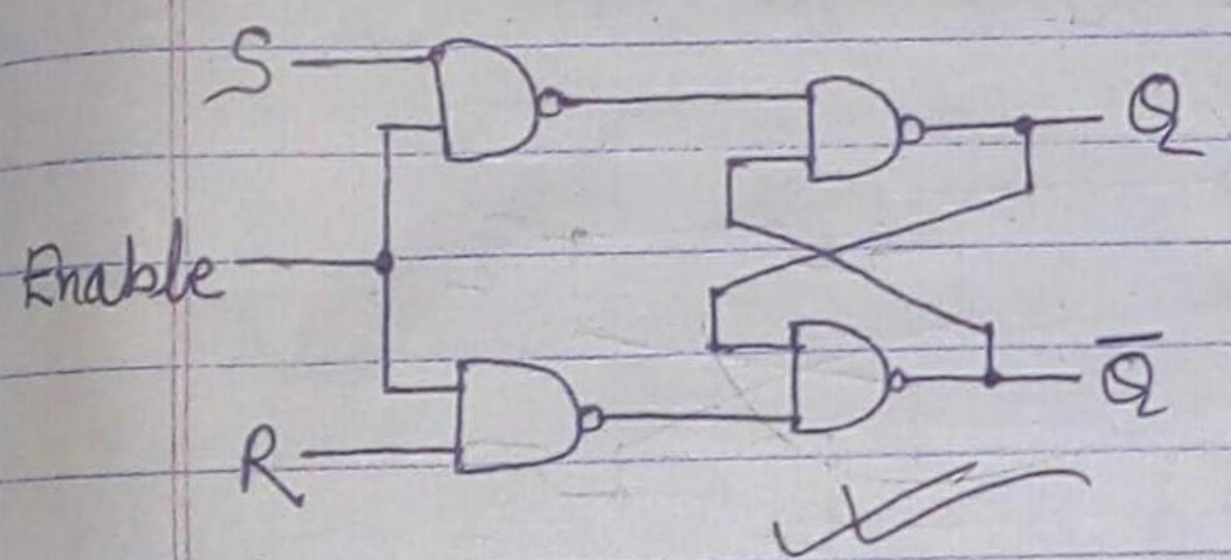


fig. clocked RS flip flop using NAND gate

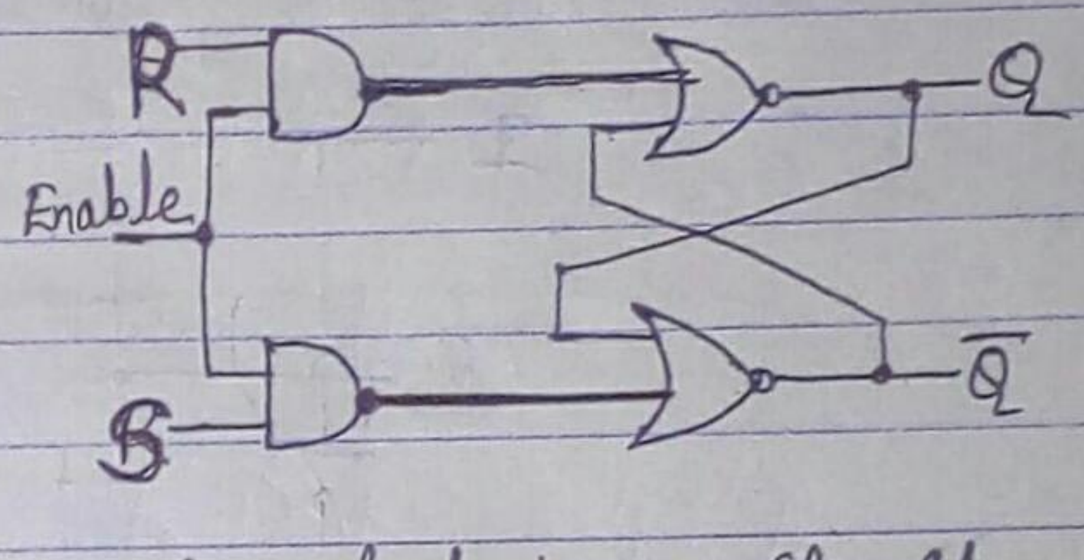


fig. clocked RS flip flop using NOR gate.

Characteristics table

ClK	S	R	Q_{n+1}
0	x	x	Q_n (no change)
1	0	0	Q_n (no change)
1	0	1	0 (reset)
1	1	0	1 (reset)
1	1	1	Illegal.

{S का value हेरु}

Here Q_n is previous output of Q and Q_{n+1} is present output of Q.

This is actived only when clock pulse is 1 or ~~hight~~ high.

Most of the time purpose of flip flop is to store 1 bit of memory and hence either Q or $\overline{Q}$ can be taken in account.

1(b) **D - flip-flop :-**

The D flip-flop is a modification of the clocked RS flip flop. We can design D flip flop using four NAND gates. The inputs R and S are replaced by only single input with true value and complement value and let that single input be D. Then, D flip flop can be designed as follows:-

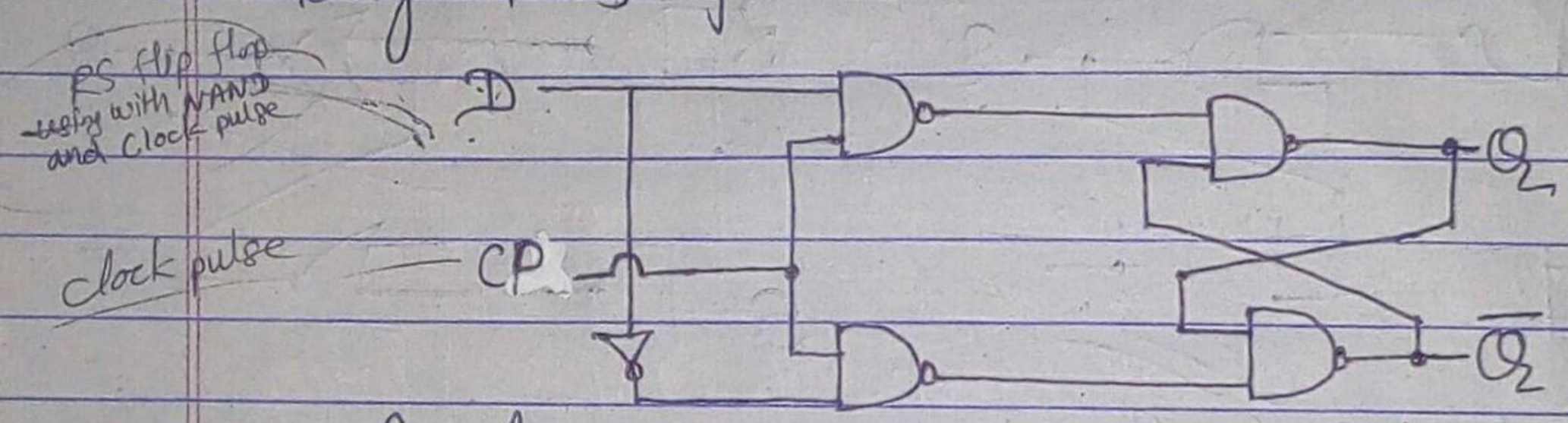
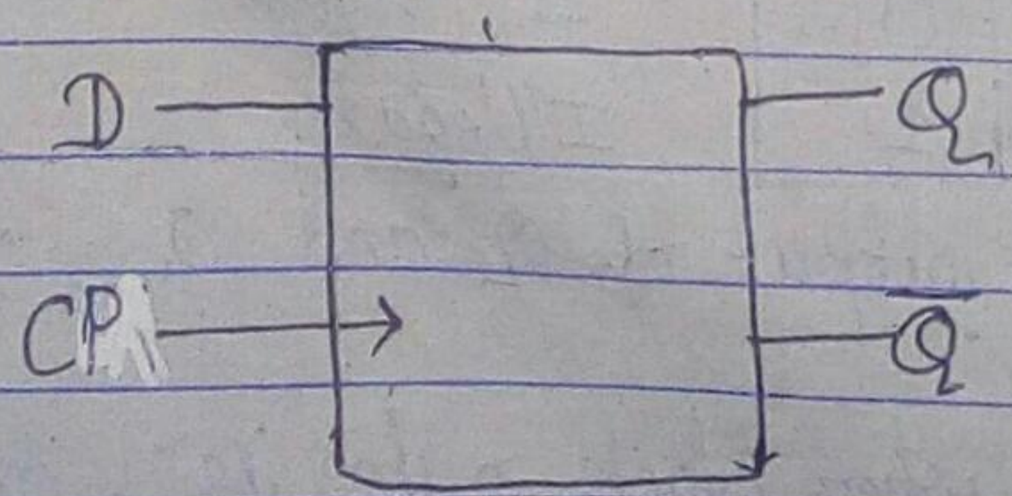


fig. logic diagram of D flip flop.

Characteristic table

Q	D	Q (t+1)
0	0	0
0	1	1
1	0	0
1	1	1



Graphical symbol of D flip flop.

1(D) JK flip flop:-

A JK flip-flop is the refinement of RS flip flop to solve the problem of intermediate state (or illegal) when both inputs are 1. In JK flip flop inputs J and K behave like inputs S and R to set and reset the flip-flop. JK flip flop can be made by taking RS flip flop with NOR gates (or NAND gates) having inputs J & K along with clock pulse (CP) as follows:-

(but we are taking NOR gates.)

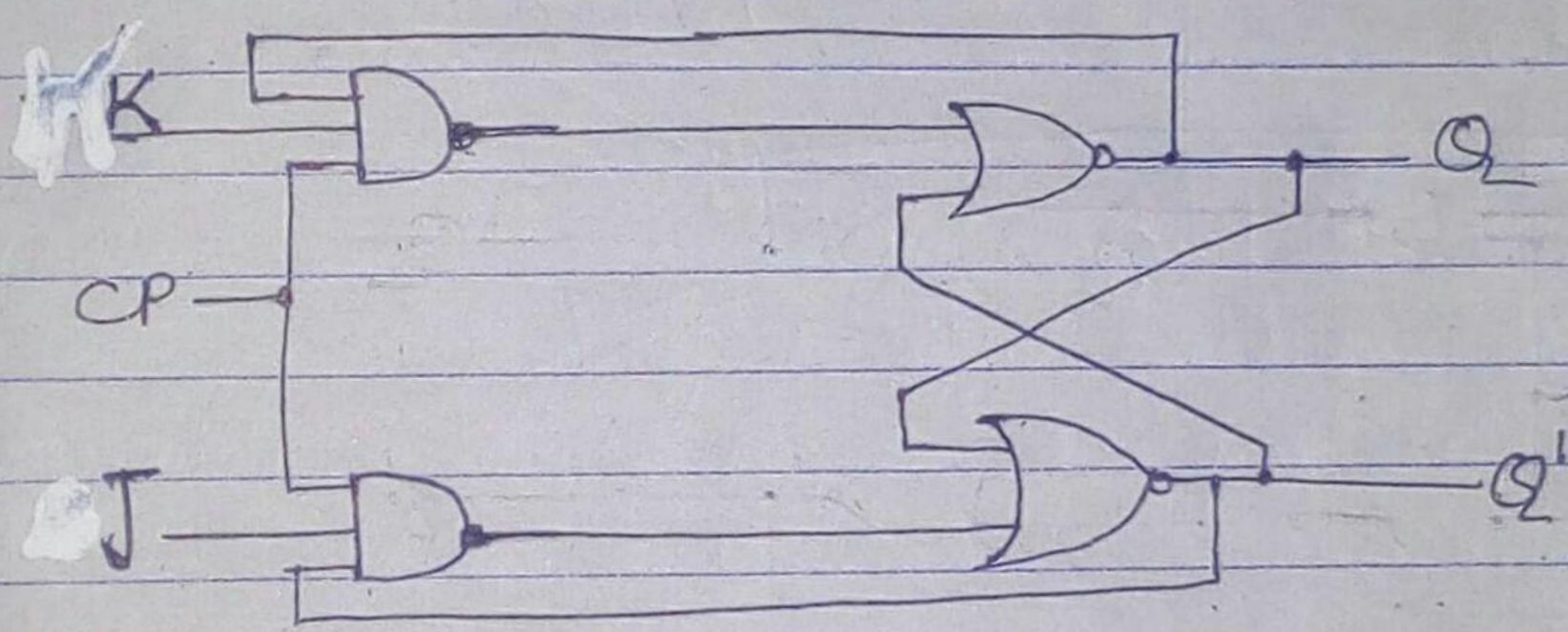
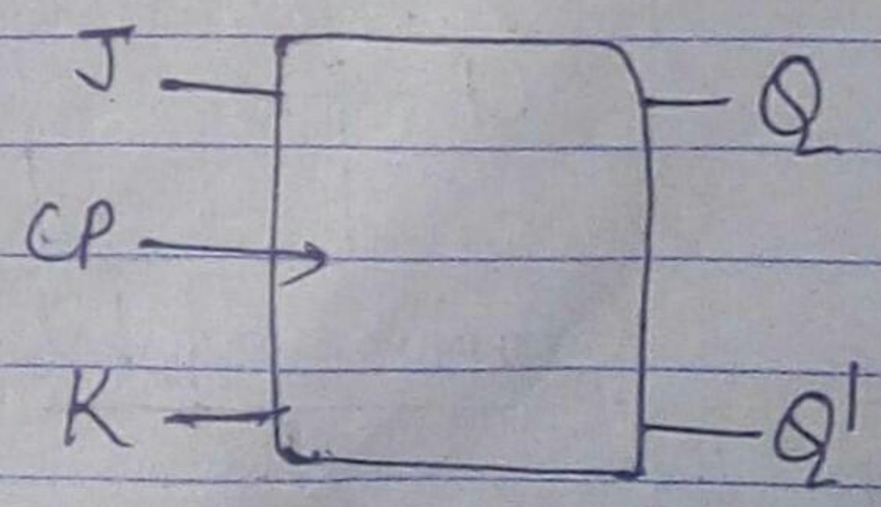


fig. logic diagram for JK flip flop.

Characteristic table

Q	J	K	Q(t+1)
0	0	0	0 → no change
0	0	1	0 → no change
0	1	0	1 → set
0	1	1	1 → set
1	0	0	1 → no change
1	0	1	0 → reset
1	1	0	1 → no change
1	1	1	0 → reset.



Graphic symbol of JK flip flop

1(e) T flip flop :-

T flip flop is ~~the~~ replacement of JK flip flop by only simple input. This flip flop just complement the input (i.e. the output of flip flop is the complement of input value) and hence it is also known as toggle flip flop. The logic diagram can be easily obtained by replacing J & K in the JK flip flop by single input T and is shown below along with characteristic table.

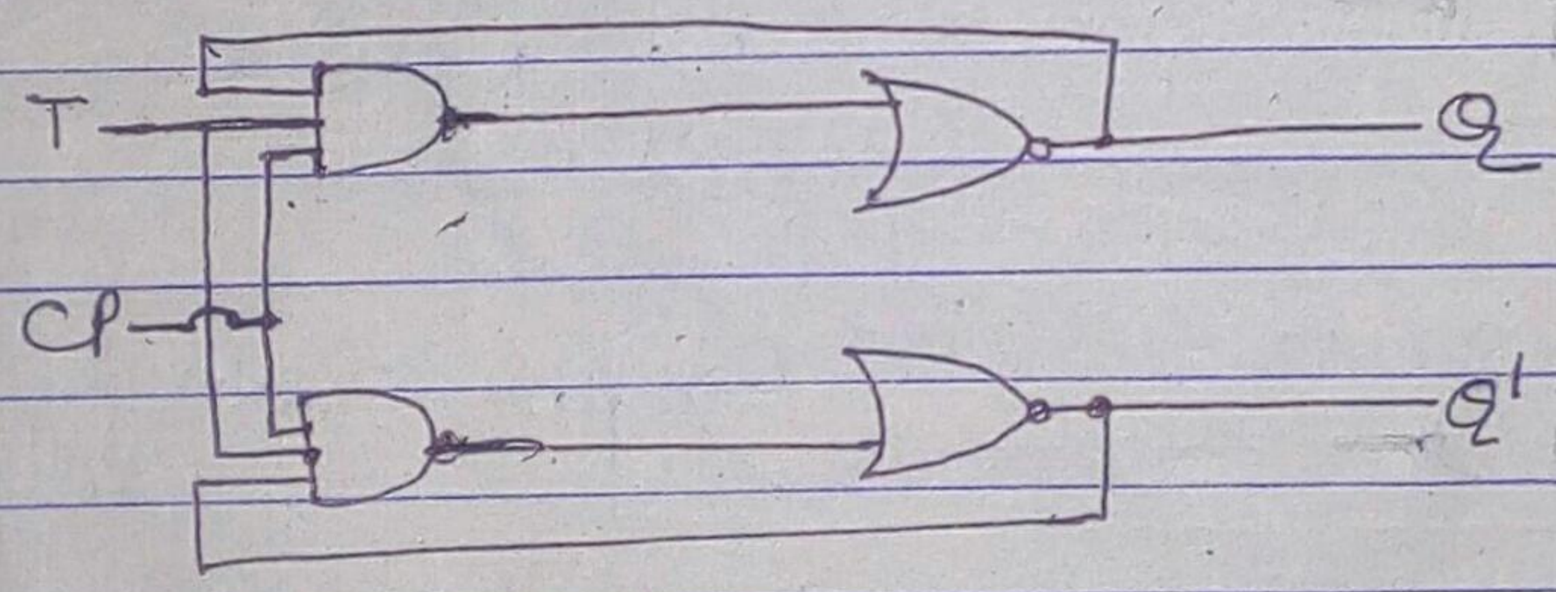
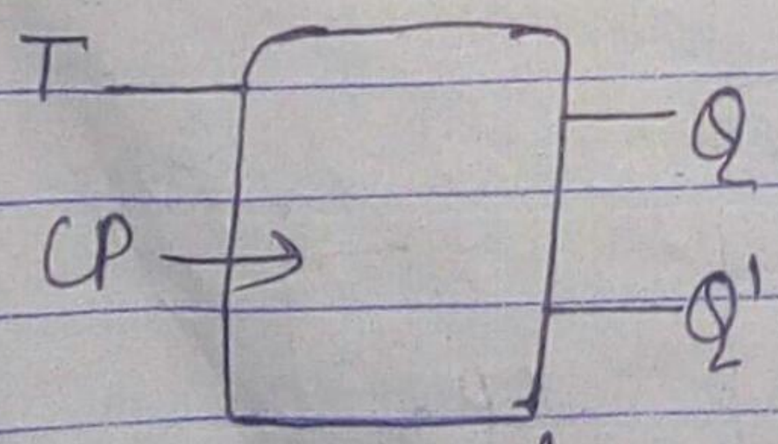


fig. logic diagram for T flip flop.



Graphic symbol.

Characteristic table

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Scanned by TapScanner

(OR master slave OK flip flop)

Navneet
Date
Page

1(e) Master Slave flip flop:- (JK master slave flip flop)

A master slave flip-flop can be constructed using two JK flip-flops as shown in figure ① below. The first flip-flop called the master is driven by the positive clock. The second flip-flop, called the slave, is driven by the negative clock.

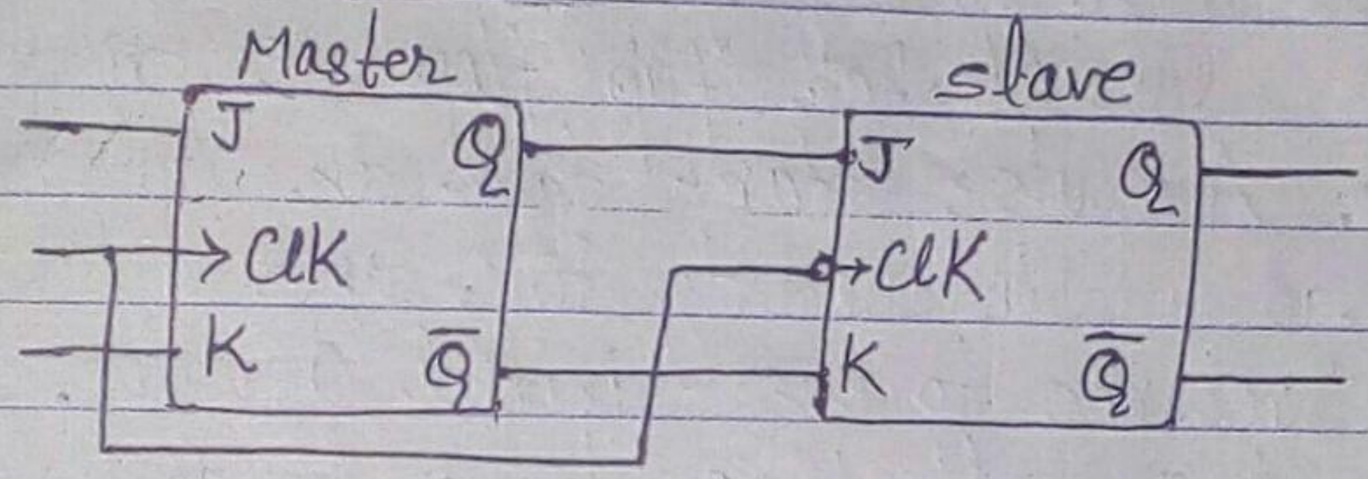


fig ① Master-slave JK flip-flop.

During positive clock master flip-flop gives intermediate output, but the slave does not respond. During negative clock, slave flip-flop activate and it copies the previous output of master flip-flop and produces final output.

A master-slave JK flip-flop can be constructed by using NAND gates as shown in figure ② below:-

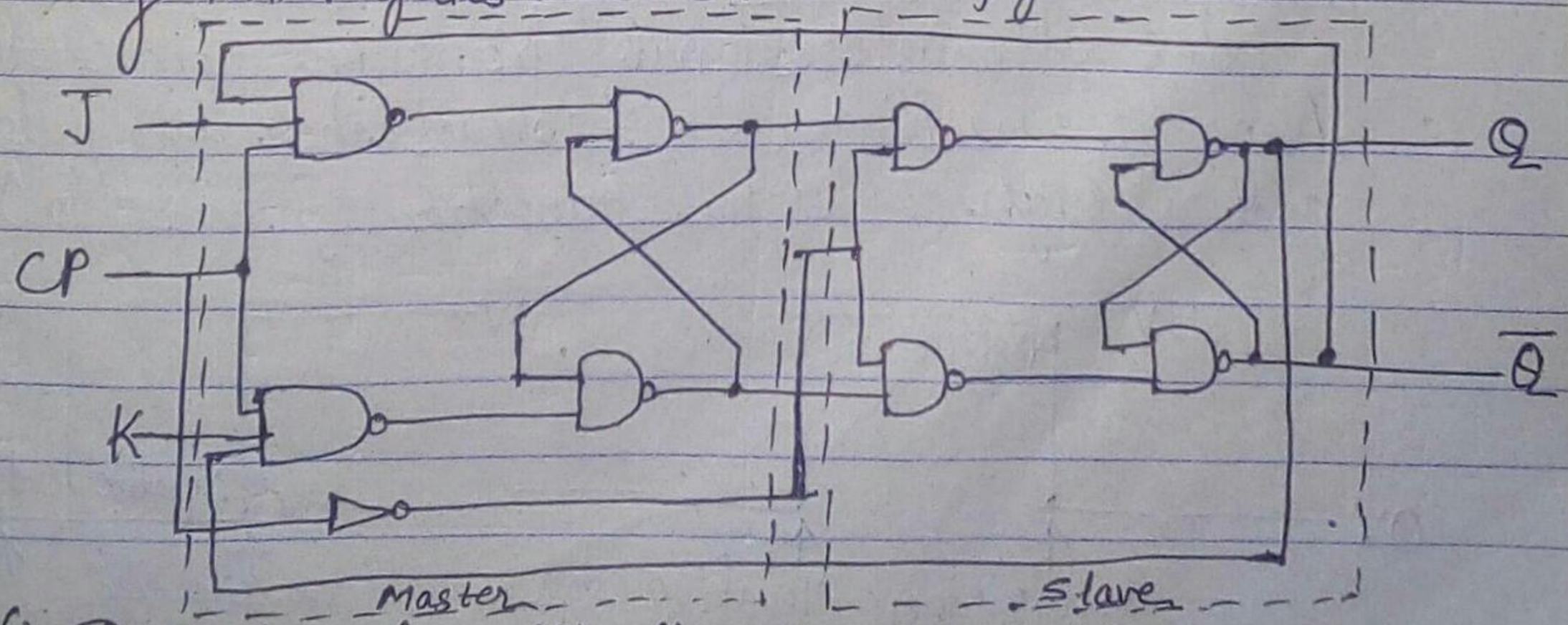


fig ②. Master-slave flip-flop using NAND gates.

I/O depends -> low level triggering.

Navneet

Date
Page

Triggering of flip-flop:-

The momentary change in input signals changes the state of flip flip which is called trigger and the transition that cause the change is called triggering of flip-flop.

Edge triggered flip flop

Clocked flip flop are triggered by pulse. A pulse start from an initial value of 0, go instantaneously to 1 and after a short time returns to its initial 0 value. In triggering by pulse method we have to face feedback timing problem so that the concept of edge triggering or pulse transition occurs rather than the pulse duration. Hence edge triggering is those having capacity to change state either at the positive edge or negative edge of clock pulse and is sensitive to its input only at the transition of clock. It is divided into two types:-

Positive edge trigger → It occurs when the pulse goes through signal transition from 0 to 1.

Negative edge trigger → It occurs when the pulse goes through signal transition from 1 to 0.

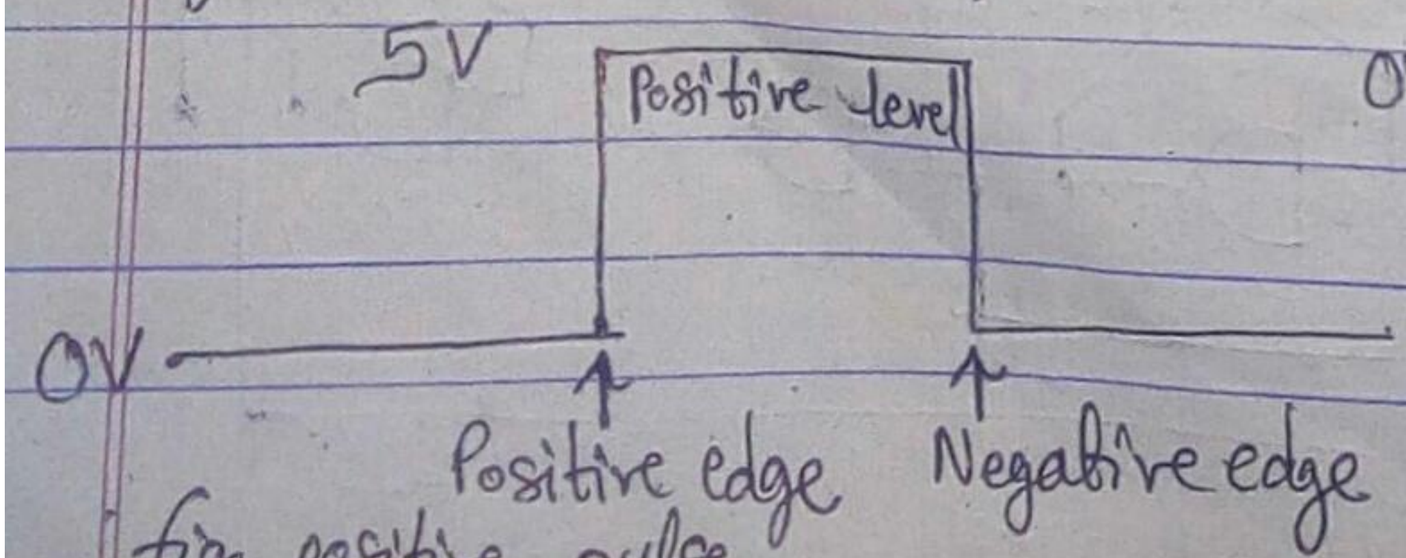


fig. positive pulse

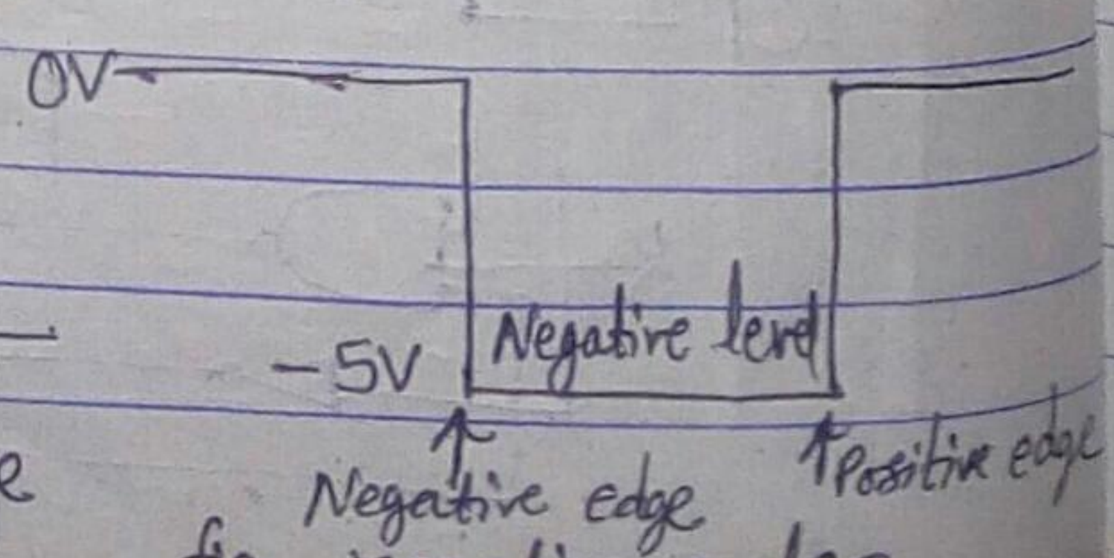


fig. negative pulse

Edge triggered D flip flop :-

Edge triggered D flip flop can be made using three RS latches as follows:-

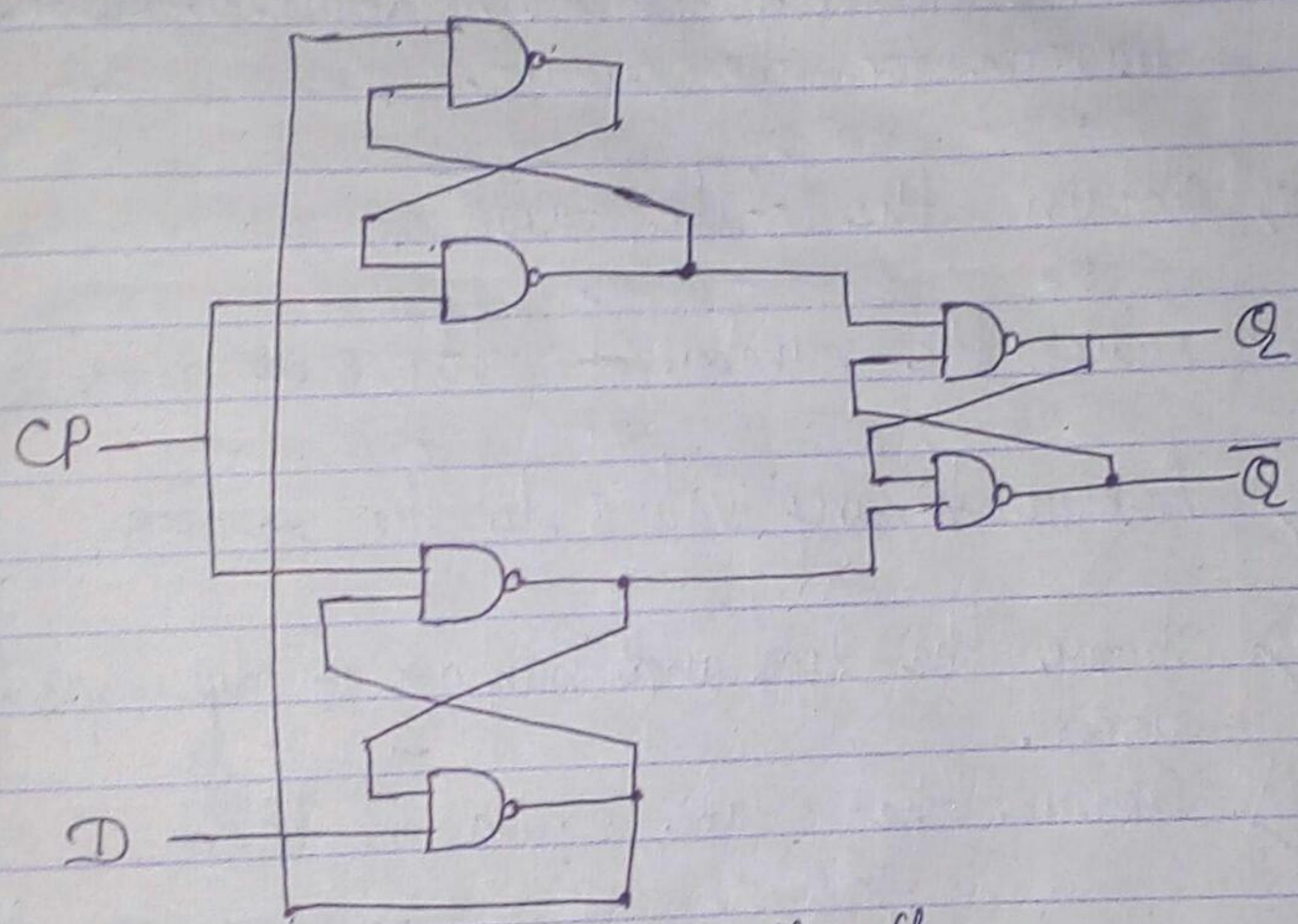


fig. Edge triggered D flip flop.

Truth table			
input		output	
CP	D	Q	$\overline{Q}$
X	0	no change	no change
X	1	no change	no change
1	0	0	1
1	1	1	0

Very Imp

Design Procedure of Sequential Circuits:-

Design procedure of sequential circuits consists of following steps:-

i) From the word description and specifications of the desired operation, derive a state diagram for the circuit.

ii) Obtain the state table.

iii) Reduce the number of states if possible.

iv) Assign binary values to the states.

v) Choose the type and number of flip-flops to be used.

vi) Obtain the state excitation table.

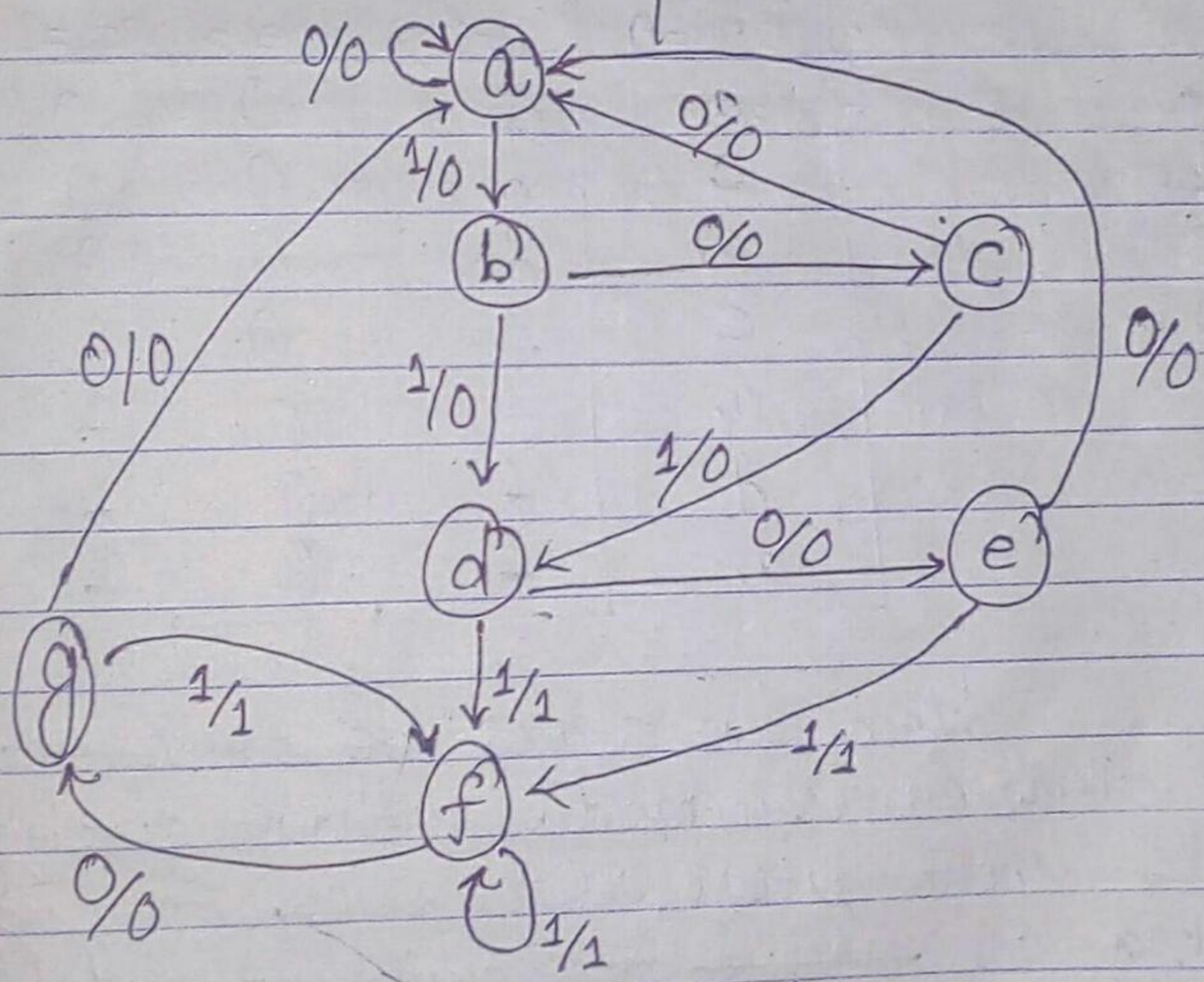
vii) Derive the simplified flip-flop input equations and output equations.

viii) Draw the logic diagram.

Scanned by TapScanner

State reduction diagram/table:-

The process of reducing the no. of flip flops in any sequential circuit is called the state reduction. These processes are concerned with the procedures for reducing the no. of states in the state table keeping input-output unchanged. Consider an example as below for illustration of the state reduction procedure.



Here we consider the input sequence 01010110100 starting from the initial state (a). Each input of 0 or 1 produces an output 0 or 1 and causes the circuit go to the next state. The state, input and output of the sequence from above diagram is as follows:-

State	a	a	b	c	d	e	f	f	g	f	g	a
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

Now, the state table becomes as follows:

	Present state	Next state		Output	
		$x=0$	$x=1$	$x=0$	$x=1$
	a	a	b	0	0
	b	c	d	0	0
	c	a	d	0	0
	d	e	f	0	1
similar	e	a	f	0	1
	f	g	f	0	1
	g	a	f	0	1

When two states are equivalent one of them can be removed without altering the input output relationships. Applying this method in ~~present~~ two equivalent states. e and g present states the reduced table becomes as follows:

	Present state	Next state		Output	
		$x=0$	$x=1$	$x=0$	$x=1$
	a	a	b	0	0
	b	c	d	0	0
	c	a	d	0	0
	d	e	f	0	1
	e	a	f	0	1

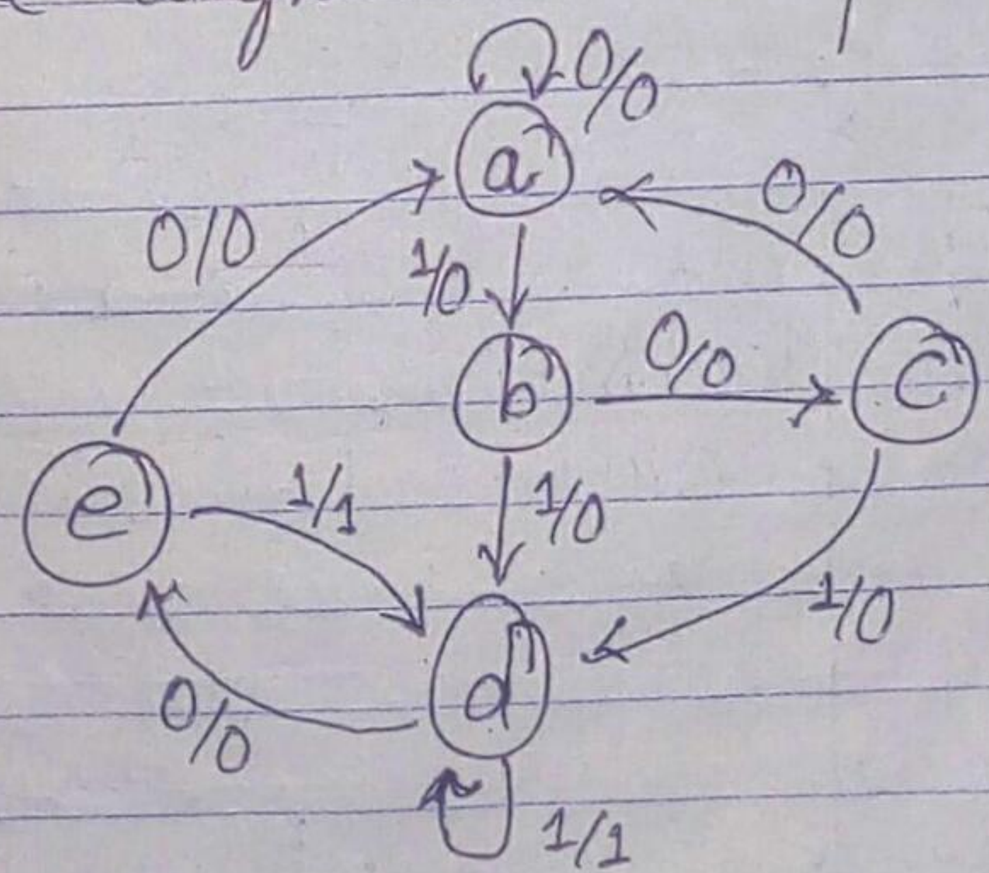
$\because g = e$

f	e	f	0	1

Again present states d and f are equivalent and reducing them we get reduced table as follows:-

Present state	Next state		Output	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d $(\because d=f)$	0	0
d	e	d	0	1
e	d $(\because d=f)$	d	0	1

Now, the diagram and state table can be represented as follows:-



$\because g = e$
$f = d$

state	d	a	b	c	d	e	d	d	e	d	e	a
input	0	1	0	1	1	1	0	1	0	0		
output	0	0	0	0	1	1	0	1	0	0		

Here the states are reduced and circuit consists lesser number of flip flops having similar functionality.

Navneet
Date
Page

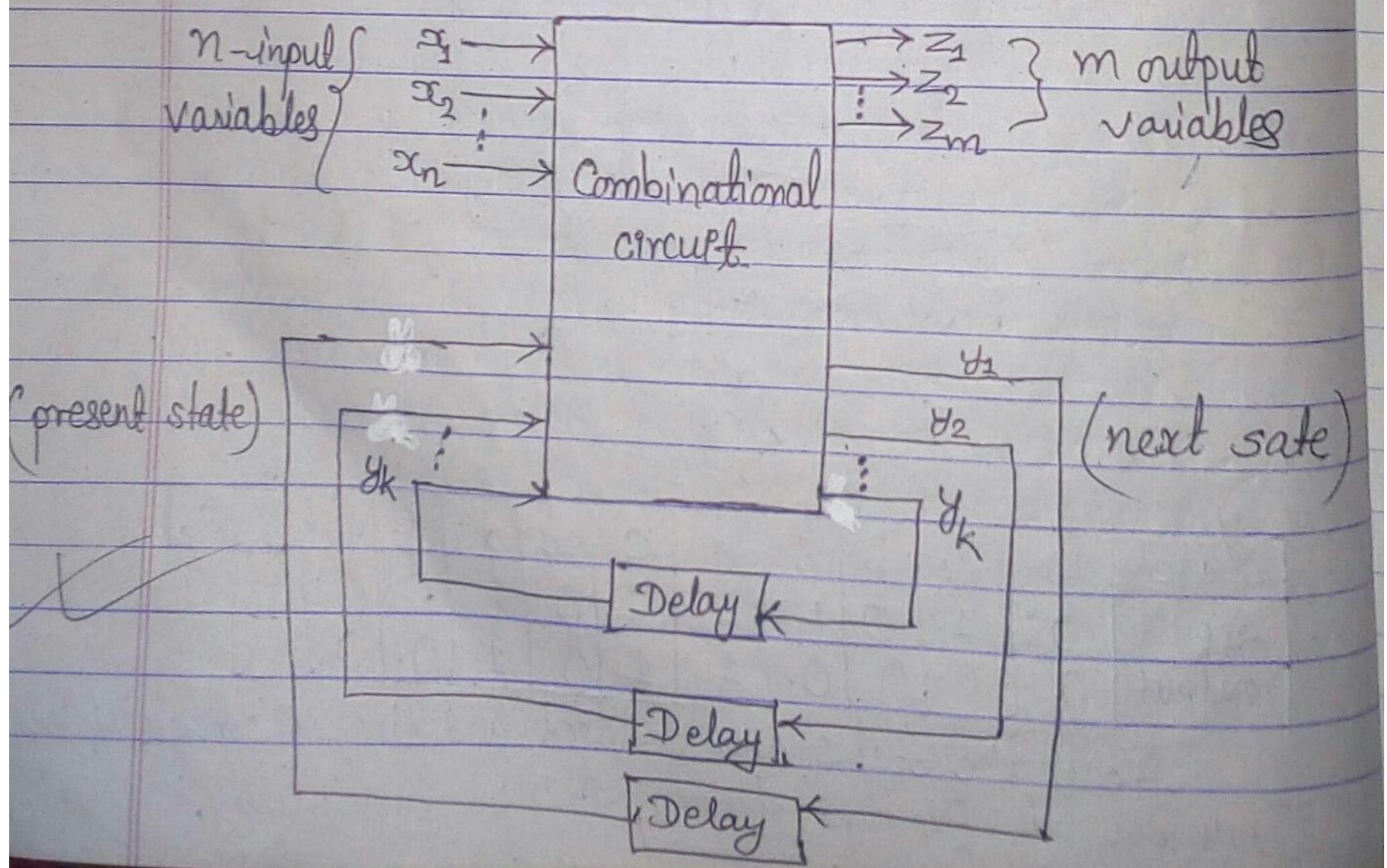
(less imp).

Asynchronous circuits :-

An asynchronous circuit, or self-timed circuit is a sequential digital logic circuit which is not governed by a clock circuit or global clock signal. i.e. the inputs are levels and there are no clock signals.

In short → The circuit is said to be asynchronous if it is not driven by a periodic clock signal to ~~synchronous.~~ synchronize its internal states.

State diagram for asynchronous circuits

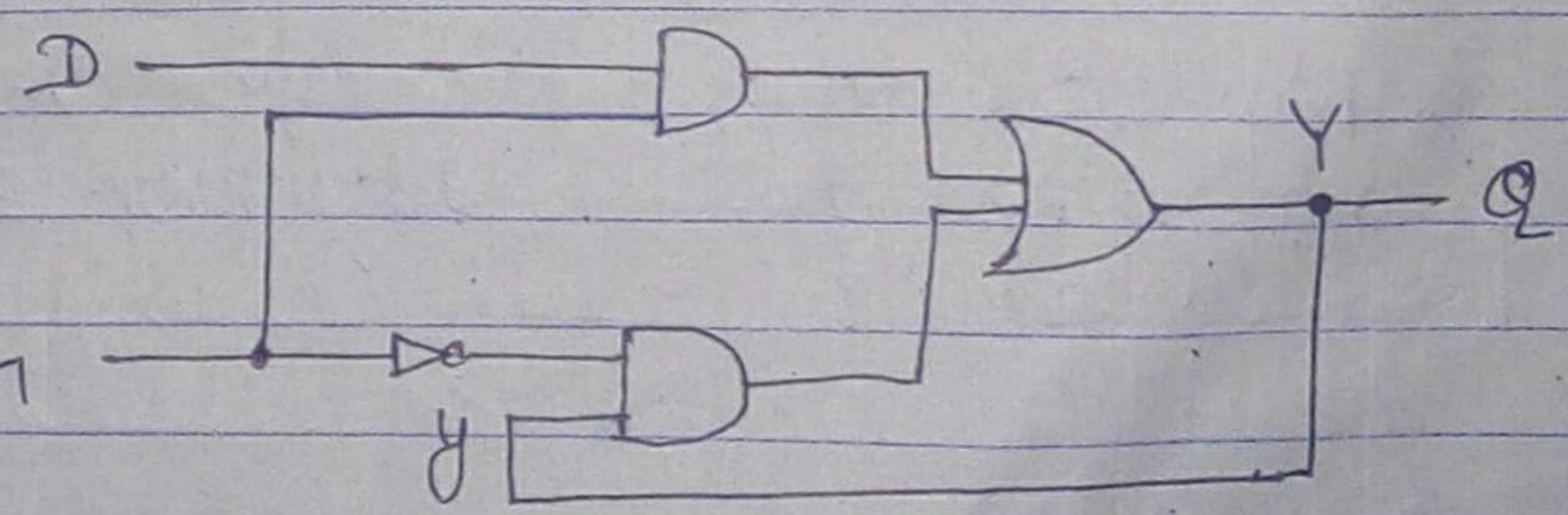


State table for asynchronous circuits:-

State	Input D	Input G	Output	Comments
a	0	1	0	D = Q because G = 1
b	1	1	1	D = Q because G = 1
c	0	0	0	After states a or d
d	1	0	0	after state c
e	1	0	1	after state b or f
f	0	0	1	after state e

	00	01	11	10
a	c,–	a,0	b,–	–,–
b	–,–	a,–	b,1	e,–
c	c,0	a,–	–,–	d,–
d	c,–	–,–	b,–	d,0
e	f,–	–,–	b,–	e,1
f	f,1	a,–	–,–	e,–

State diagram



Circuits with latches:

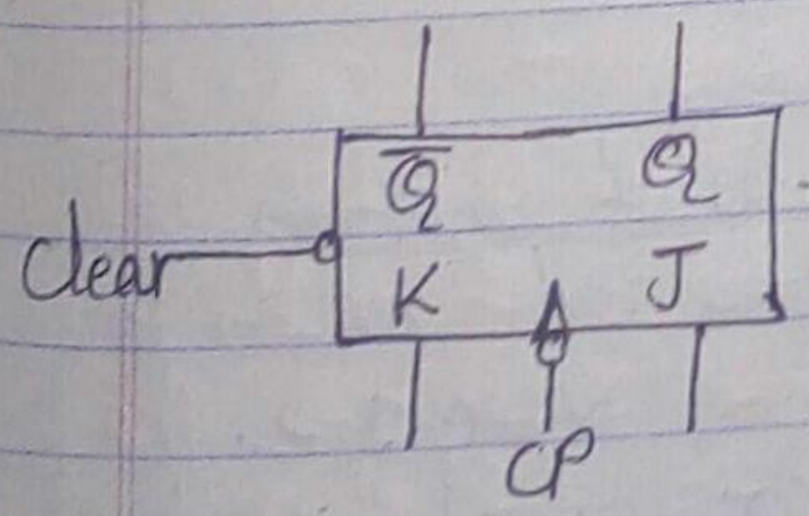
(*) Differences between asynchronous and synchronous sequential circuits.

Synchronous	Asynchronous
i) These circuits are easy to design.	i) These circuits are difficult to design.
ii) A clocked flip flop acts as memory element.	ii) An unclocked flip flop or time delay element is used as memory element.
iii) They are slower.	iii) Faster as clock is not present.
iv) The status of memory element is affected only at the active edge of clock, if input is changed.	iv) The status of memory element will change any time as soon as input is changed.
v) Flip flops are used in synchronous circuits	v) Latches are used in asynchronous circuits.

Direct inputs:

~~Flip~~ Flip-flops available in IC packages sometimes provide special inputs for setting or clearing the flip-flop asynchronously. These inputs are usually called direct present or direct clear. They affect the flip-flop on a positive (or negative) value of the input signal without the need for a clock pulse. These inputs are useful for bringing all flip-flops to an initial state prior to their clocked operation.

Example:- After power is turned on in a digital system, the states of its flip-flops are intermediate. A clear switch clears all the flip-flops to an initial cleared state and a start switch begins the system's clocked operation. The clear switch must clear all flip-flops asynchronously without the need of a pulse.



Block diagram

Function table

Inputs				Outputs	
Clear	Clock	J	K	Q	$\overline{Q}$
0	X	X	X	0	1
1	↓	0	0	No change	
1	↓	0	1	0	1
1	↓	1	0	1	0
1	↓	1	1	Toggle	

Fig. J.K flip-flop with direct clear

Unit = 7

For this chapter read from mobile by bishnu rawal

Registers:

The ~~one~~ 1-bit memory cell is flip flop, to increase the storage capacity, we have to use group of flip-flop which is known as register. The n-bit register consists of 'n' number of flip-flops and is capable of storing "n'-bit" word.

There are different types of registers available in MSI circuits. The mostly used registers are D flip flops and such registers are called shift registers also.

Types of registers

i) Serial in Serial Out (SISO)
ii) Serial in Parallel out (SIPO)
iii) Parallel in Serial Out (PISO)
iv) Parallel in Parallel Out (PIPO).

Shift Register:-

Any register capable of shifting its binary information either to the right or to the left is called the shift register. The configuration of shift register consists of a chain of flip-flops connected in cascade, with the output of one flip Flop connected to the input of next flipflop.

Ripple Counters:- (Asynchronous Counter):-

Ripple counter is one of the asynchronous counter which is not having a similar clock pulse for each flip-flop in that counter.

Asynchronous counters are also known as ripple counters and are those in which the clock pulse is given as input to the ~~flirs~~ first flip flop, the output of the first flip flop is fed as clock input to the second flip flop.

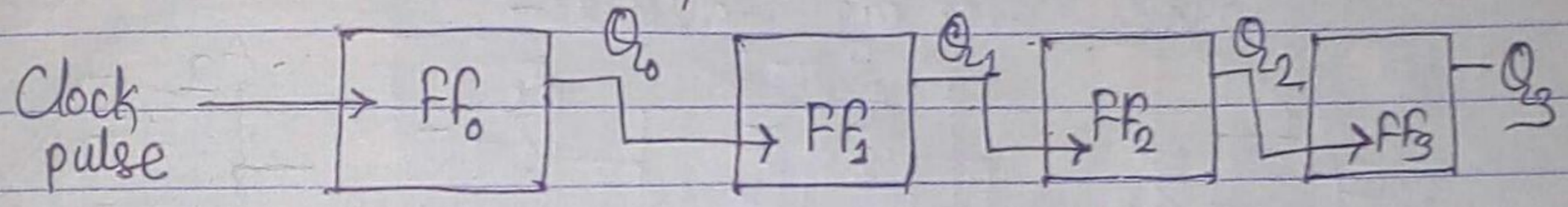


fig. Ripple counter / Asynchronous counter

Synchronous Counter:-

The synchronous counters are those in which the clock input is connected to all the flip flops individually so that they are simultaneously clocked as shown in figure below:

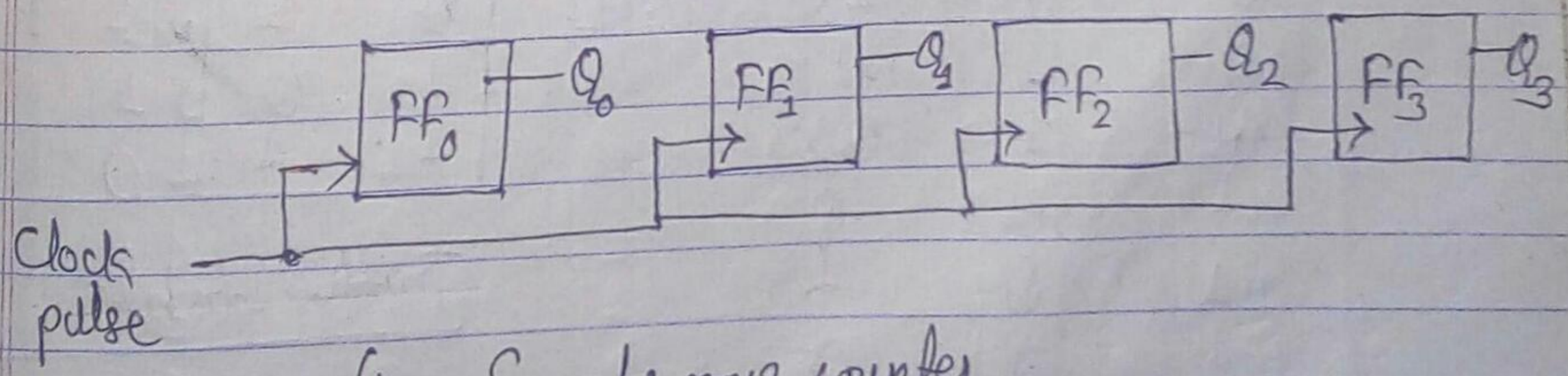


fig. Synchronous counter

Binary ripple counter:— (Binary Ripple Counter)

Binary ripple counter consists of series of connection of complementing flip flops either (T flip flop or JK flip flop) with the output of each flipflop connected to the clock pulse input of the next higher order flip flop. The flip flop holding the least significant bit receive the incoming count pulses. The diagram of binary ripple counter is as follows:-

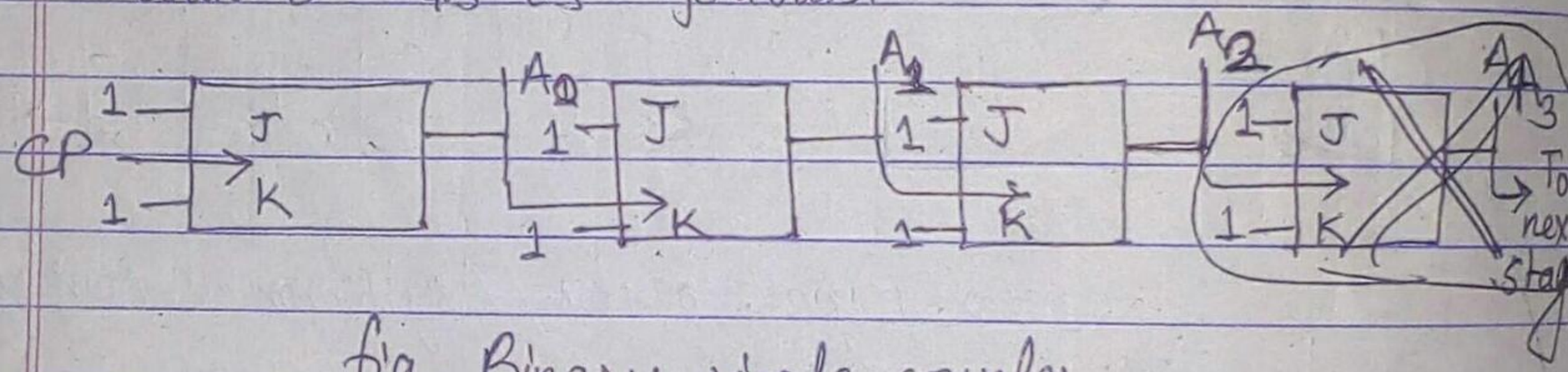


fig. Binary ripple counter

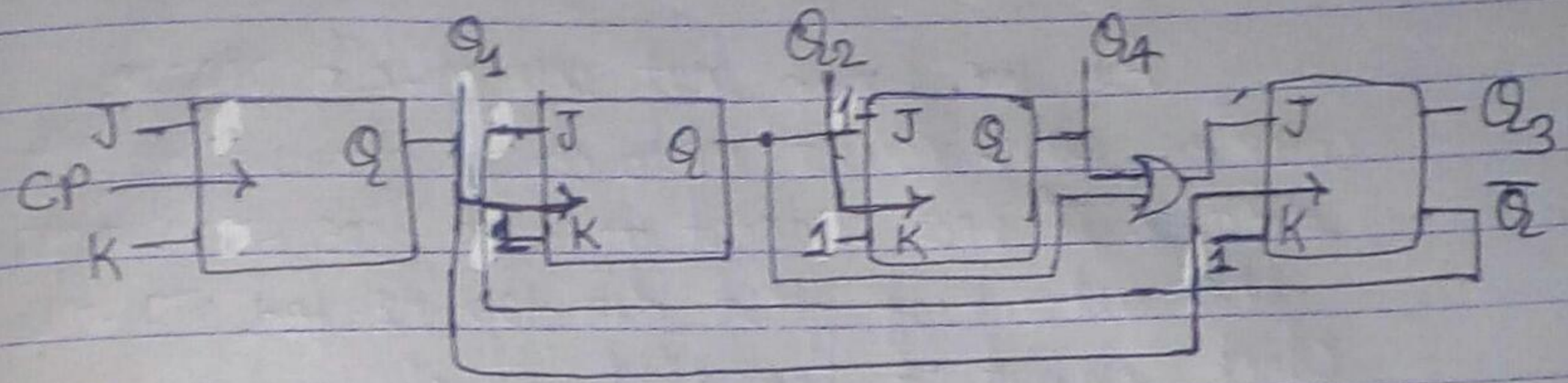
Here all the J-K inputs are equal to 1.

A_2	A_1	A_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

BCD Ripple Counter:- (Decade Counter)



CP	Q_1	Q_2	Q_3	Q_4
Initial	0	0	0	0
1st	0	0	0	1
2nd	0	0	1	0
3rd	0	0	1	1
4th	0	1	0	0
5th	0	1	0	1
6th	0	1	1	0
7th	0	1	1	1
8th	1	0	0	0
9th	1	0	0	1

When CP reaches to 10 we have to reset the flip flop to go back to initial state. By using NAND gate and giving proper input we can reset the flip flop.

Imp

Binary 4-Bit Up/Down Ripple Counter:-

As we know that any n-bit ripple counter can count upto a maximum of 2^n states. If we connect 4 flip flops such that the output of first flip flop is fed to the clock input of the next as shown below, we get the 4-bit ripple counter.

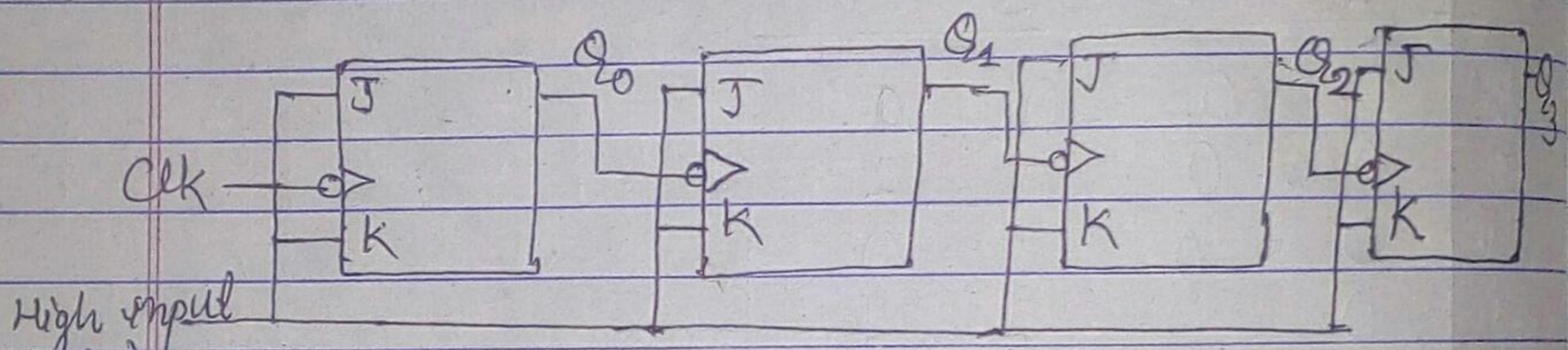


fig. 4-bit ripple counter

This is 4-bit ripple counter and it has $2^4 = 16$ states from 0000 to 1111 and then the counter reset to 0000 as shown in the truth table below:-

Clock Pulse	Q_3	Q_2	Q_1	Q_0	Count
Initial	0	0	0	0	0
First	0	0	0	1	1
Second	0	0	1	0	2
Third	0	0	1	1	3
Fourth	0	1	0	0	4
fifth	0	1	0	1	5
sixth	0	1	1	0	6

clock pulse	Q_3	Q_2	Q_1	Q_0	Count.
seventh	0	1	1	1	7
eighth	1	0	0	0	8
ninth	1	0	0	1	9
tenth	1	0	1	0	10
eleventh	1	0	1	1	11
twelveth	1	1	0	0	12
thirteenth	1	1	0	1	13
fourteenth	1	1	1	0	14
fifteenth.	1	1	1	1	15
Sixteenth	0	0	0	0	Reset

The timing diagram for one cycle for the 4-bit ripple counter is as shown below:

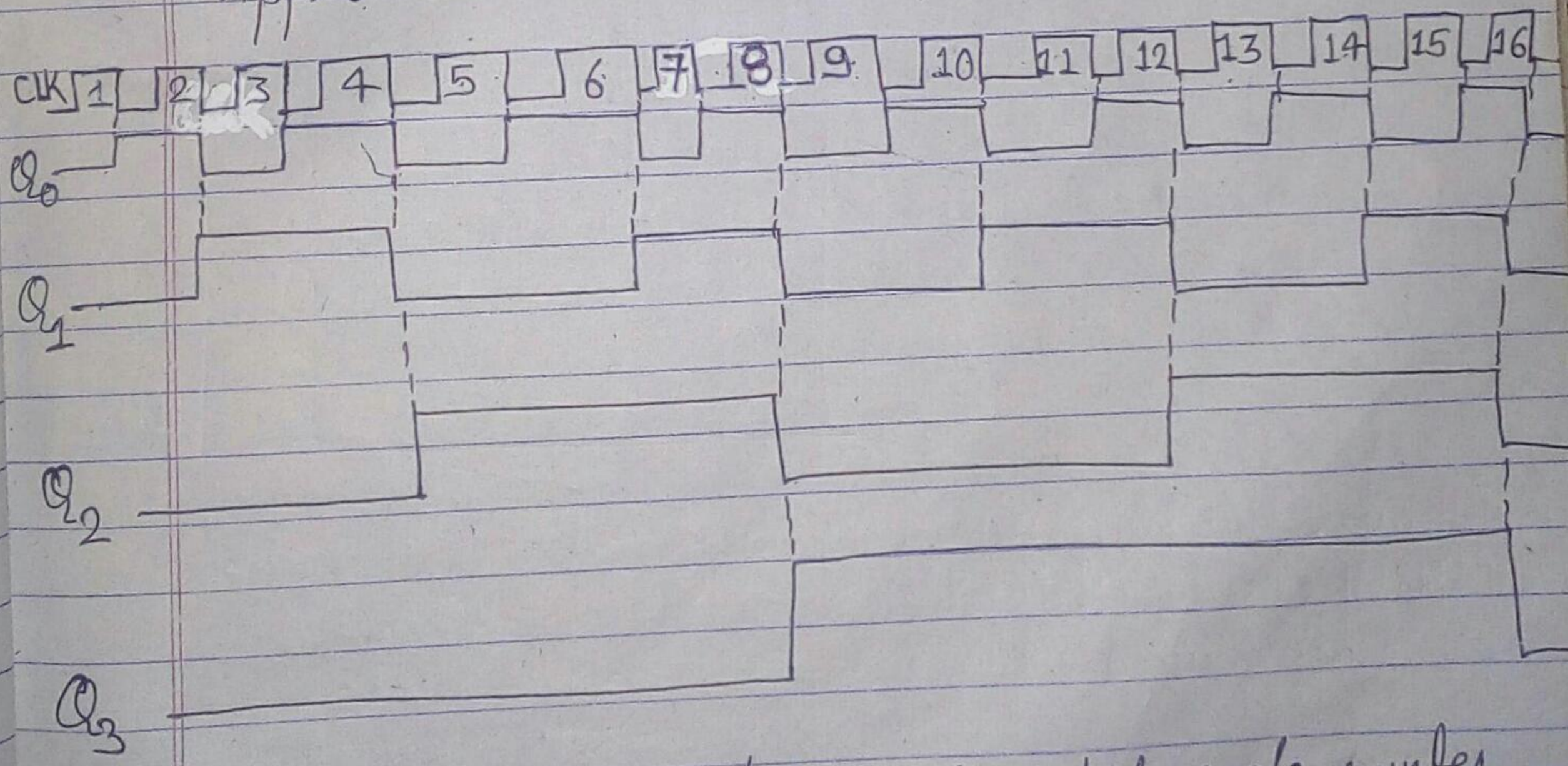
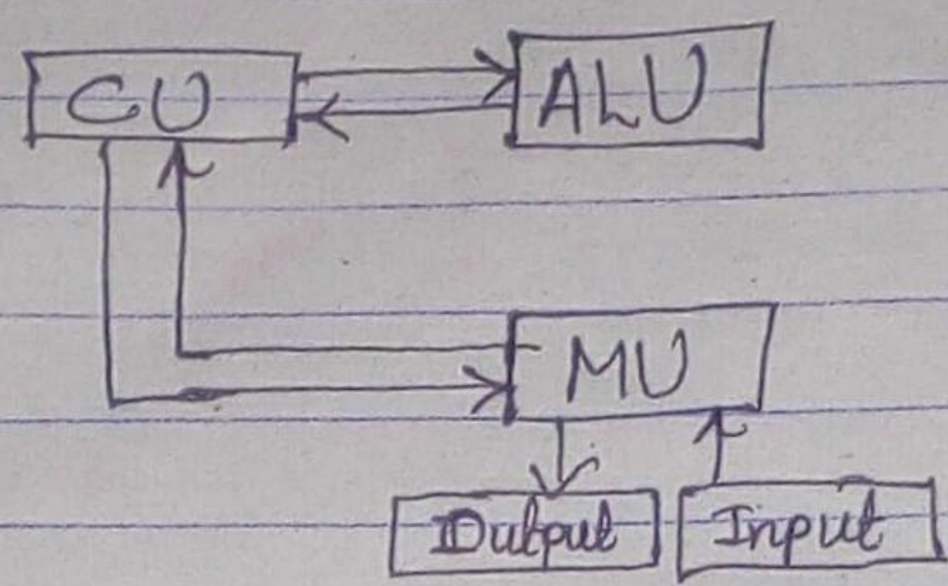


fig. Timing diagram for 4-bit ripple counter.

Additional

Unit - 1

* Block diagram of digital computer.



* Digital and Analog Computers

An analog computer works on continiously changable aspects of physical phenomenon such as mechanical motion and electrical quantities. These computers measure changes in continious physical quantities say current and voltage.

These computers are used to process data generated by ongoing physical processes. A thermometer is an example of an ~~ale~~ analog computer since it measures the change in mercury level continiously.

Digital computers are capable of processing information in discrete form. In digital technology data ~~which~~ are in discrete form represented in binary numbers 0's and 1s.

Binary digits are easily expressed in a digital computer by presence (1) or absence (0) of current or voltage. They are used in industrial, business and scientific applications. They are quite suitable for large data processing.

Navneet

Date
Page

Advantages & Disadvantages of digital computer.

Advantages

i) Digital computers are accurate.
ii) Store huge amount of information.
iii) Less expensive.
iv) More reliable.
v) Easy to manipulate.
vi) Flexibility and Compatibility.
vii) Can do multiple types of tasks.

Disadvantages

i) Digital computers are slow as compared to analog computers.

ii) Use more energy than analog circuits to finish same tasks and produce more heat as well.

iii) Quantization error during analog signal sampling.

Scanned by TapScanner

Navneet

Date
Page

Unit = 2

Boolean Algebra (Definition) →

Boolean Algebra is a mathematical system with a set of elements, a set of operators and a number of unproved postulates.

Integrated Circuits

→ Concept of DIP, SIMM, linear and digital IC's.

DIP → It stands for dual-in-line package. It is an electronic component package with a rectangular housing and two parallel rows of electrical connecting pins. The package may be through-hole mounted to a PCB.

SIMM → It stands for Single-in-line Memory module. It is a type of memory module containing random access memory (RAM) used in computers from the early 1980's to the late 1990's.

Linear IC → It is a solid-state analog device characterized by theoretically infinite number of possible operating states. It operates over a continious range of input levels. Linear IC's are employed in audio amplifiers.

digital IC → digital IC's use logic functions such as AND, OR and NOT. It operates over a finite range of input levels. It consists electronic components such as transistors.

Scanned by TapScanner

Navneet

Date
Page

Characteristics of IC logic families

i) first introduced in 1958
ii) Categorised as digital or linear IC's
iii) usually referred to as a monolithic IC.
iv) Low-cost electronic ~~circuits.~~ components fabricated on a single continuous piece of semiconductor material to perform a high level function.

Scanned by TapScanner