* Ex-4.1: A combinational circuit has four inputs and one output. The output is equal to 1 when (1) all the inputs are equal to 1 or (2) none of the inputs are equal to 1 or (3) odd number of inputs are equal to 1.

(a) obtain the truth table.

Soln:

Input				output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

(b) Find the simplified output function in sum of products.

Soln:

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

for SOP = $A'B'C' + A'C'D' + B'CD' + A'B'D + BCD + ABD + ABC + ACD$

(c) Find the simplified output function in product of sums.

Soln:

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

for POS = $(A+B+C'+D')(A+B'+C+D)(A+B'+C'+D)(A'+B+C+D')(A'+B+C'+D')(A'+B'+C+D)$

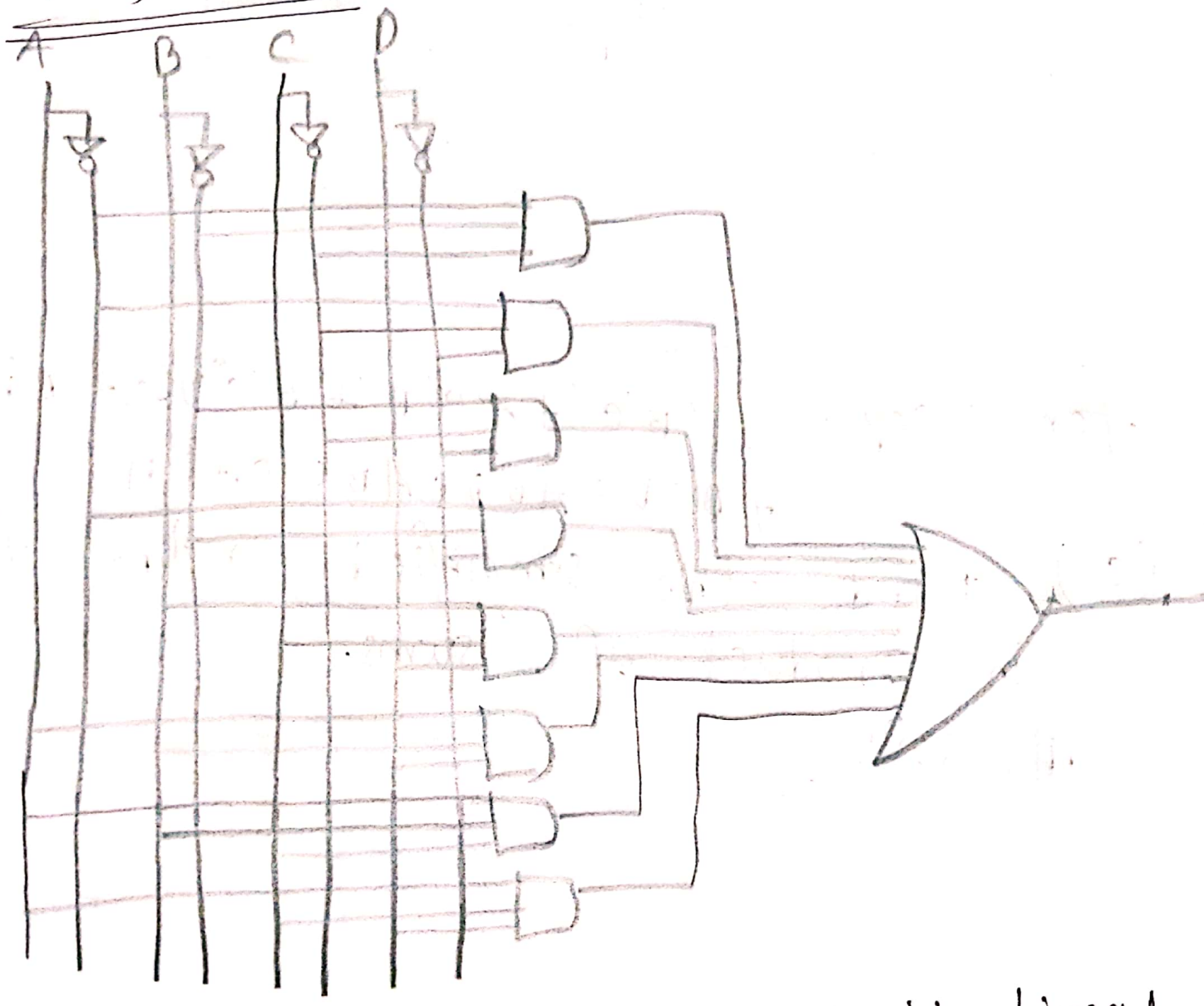
$Y = \Pi(3, 5, 6, 9, 10, 12)$ not simplified.

(d) Draw the circuit diagram.

Soln:

(4)

⇒ for b:



Ex-4.2: Design a combinational circuit that accepts a three-bit number and generates an output binary number equal to the square of the

input number:

Ans Soln:

input			output					
A_3	A_2	A_1	B_6	B_5	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

$\Rightarrow$ for $B_1 = A_1$

$\Rightarrow$ For $B_2 = 0$

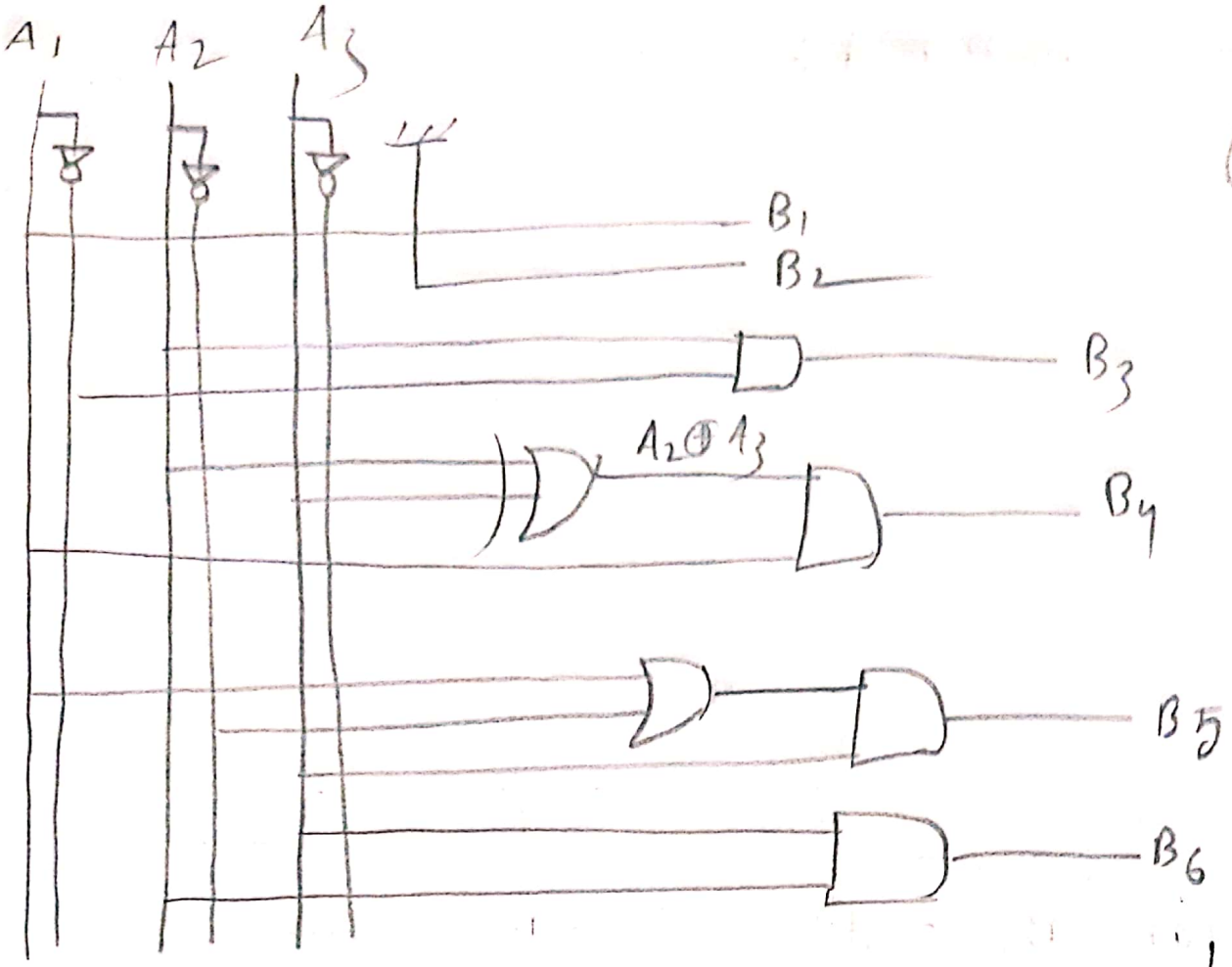
$\Rightarrow$ for $B_3 = A_2 A_1'$

$\Rightarrow$ for $B_4 = A_1(A_2 \oplus A_3)$

$\Rightarrow$ for $B_5 = A_3(A_1 + A_2')$

$\Rightarrow$ for $B_6 = A_3 A_2$.

now we can draw the circuit diagram for these expression.



Ex-4.3: It is necessary to multiply two binary numbers, each two bits long, in order to form their product in binary. Let, the two numbers be represented by a_1, a_0 and b_1, b_0, where subscript 0 denotes the least significant bit.

(a) Determine the number of output lines required.

(5)

Soln: 4 output lines required.

(b) Find the simplified Boolean expressions for each output.

Soln:

⇒ truth table:

a_1	a_0	b_1	b_0	z_3	z_2	z_1	z_0
0	0	0	0				0
0	0	0	1				0
0	0	1	0				0
0	0	1	1				0
0	1	0	0				
0	1	0	1				1
0	1	1	0			1	
0	1	1	1			1	1
1	0	0	0				
1	0	0	1			1	
1	0	1	0		1		
1	0	1	1		1	1	
1	1	0	0				
1	1	0	1			1	1
1	1	1	0		1	1	
1	1	1	1	1			1

⇒ Now, for simplification we use k-map and find:

$Z_3 = a_1 a_0 b_1 b_0$

$Z_2 = a_1 a_0' b_1 + a_1 b_1 b_0'$

$Z_1 = a_1 a_0' b_0 + a_1 b_1' b_0 + a_1' a_0 b_1 + a_0 b_1 b_0'$

$Z_0 = a_0 b_0$

* Ex-9.4: Repeat problem 9.3 to form the sum (instead of product) of the two binary numbers.

Soln:

$Z_2 = a_1 b_1 + a_1 a_0 b_0 + b_1 b_0 + a_0$

$Z_1 = a_1' a_0' b_1 + a_1' a_0 b_1' + a_1' a_0 b_0 + a_1 a_0' b_0 + a_1 a_0 b_1 b_0 + a_1 a_0' b_1' + a_1 a_0 b_1 b_0.$

$Z_0 = a_0 b_0' + a_0' b_0.$

* Ex-9.5: Design a combinational circuit with four input lines that

represent a decimal digit in BCD and four output lines that generate the 9's complement of the input digit.

Soln:

Decimal	BCD input				output				
	A	B	C	D	8 W	4 X	2 Y	1 Z	
0	0	0	0	0	1	0	0	1	9
1	0	0	0	1	1	0	0	0	8
2	0	0	1	0	0	1	1	1	7
3	0	0	1	1	0	1	1	0	6
4	0	1	0	0	0	1	0	1	5
5	0	1	0	1	0	1	0	1	4
6	0	1	1	0	0	0	1	1	3
7	0	1	1	1	0	0	1	0	2
8	1	0	0	0	0	0	0	1	1
9	1	0	0	1	0	0	0	0	0

⟹ Now we simplify each output line using k-map:

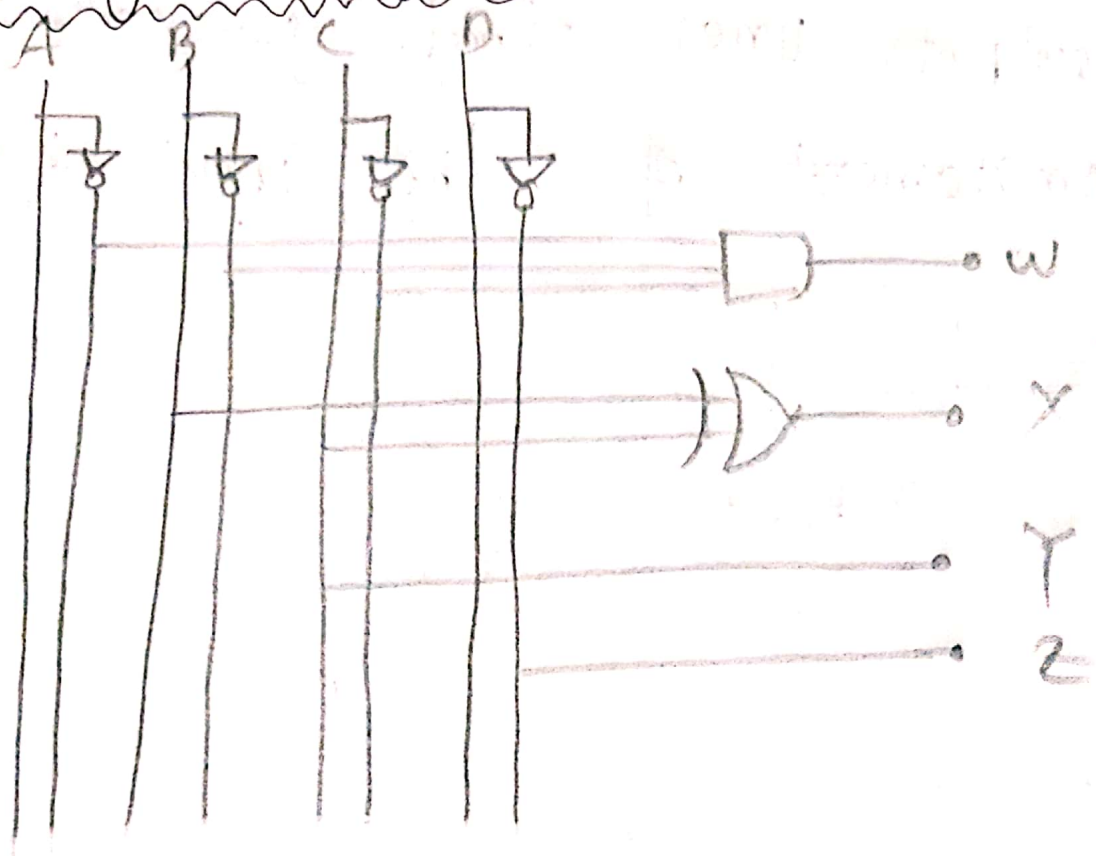
$W = A'B'C'$,

$X = BC' + B'C = B \oplus C$

$Y = C$

$Z = D'$.

⇒ Logic diagram:



* Ex-9.6: Design a combinational circuit whose input is a four-bit number and whose output is the 2's complement of the input number.

Soln: At first, we can draw a truth table for visualization.

	Input				Output			
Decimal	A	B	C	D	w	x	y	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	0
3	0	0	1	1	1	1	0	1
4	0	1	0	0	1	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	0	1	1	1
10	1	0	1	0	0	1	1	0
11	1	1	1	1	0	1	0	1
12	1	1	0	0	0	1	0	0
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	0	1	0
15	1	1	1	1	0	0	0	1

$\Rightarrow$ now, we can use k-map Here and find:

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

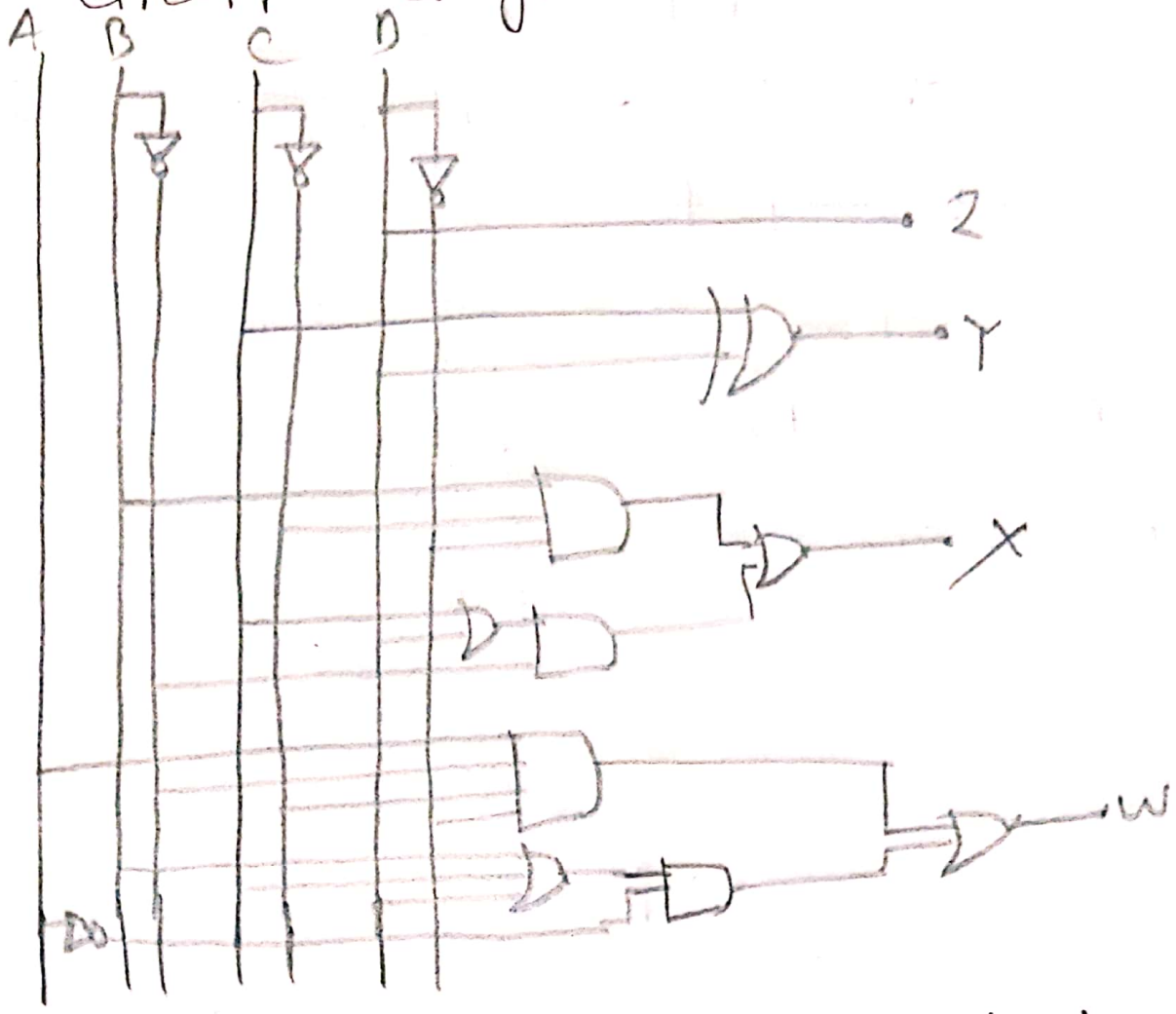
$x = BC'D' + B'D + B'C$

$= BC'D' + B'(C + D)$

$Y = C'D + CD' = C \oplus D$

$Z = D$.

⇒ Circuit diagram:



* E.X-4.7: Design a combinational circuit that multiplies by 5 an input decimal digit represented in BCD. The output is also in BCD. Show that the outputs can be obtained from the input lines without using any logic gates.

	Input				output			BCD			
D	A	B	C	D	T	U	V	W	X	Y	Z
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1	0	1
2	0	0	1	0	0	0	1	0	0	0	0
3	0	0	1	1	0	0	1	0	1	0	1
4	0	1	0	0	0	1	0	0	0	0	0
5	0	1	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	1	1	0	0	0	0
7	0	1	1	1	0	1	1	0	1	0	1
8	1	0	0	0	1	0	0	0	0	0	0
9	1	0	0	1	1	0	0	0	1	0	1

⇒ For simplify, each output we can use k-map and find,

$Z = x = D$,

$V = C$

$U = B$

$T = A$

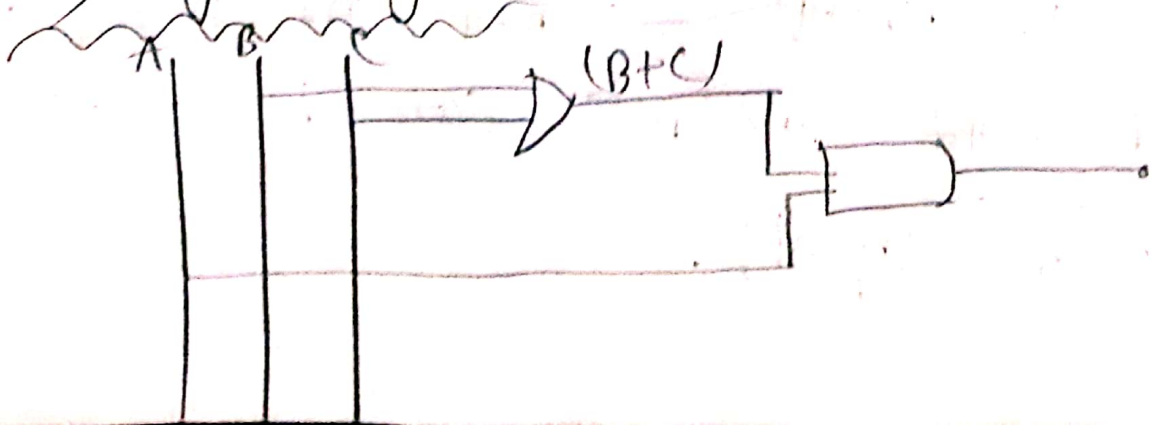
So, it is ~~true~~ that no gate is required and outputs can be realized just with inputs.

* Ex-4.8: Design a combinational circuit that detects an error in the representation of a decimal digit in BCD. In other words, obtain a logic diagram whose output is logic-1 when the inputs contain an unused combinational in the code:

Soln: For this the error will be detected so,

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

⇒ Logic diagram:



* E.x-4.10: (39) A BCD-to-seven segment decodor is a combinational circuit that accepts a decimal digit in BCD and generates the appropriate outputs for selection of segments in a display indicator used for displaying the decimal digit. The seven outputs of the decoder (a, b, c, d, e, f, g) select the corresponding segments in the display as shown in fig P 4-14(a). The numeric designation chosen to represent the decimal digit is shown in fig. P4-14(b). Design the BCD-to-seven segment decoder circuit

Soln: Truth table:

(10)

	Input				output						
Digits	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

⇒ For simplify each output we use k-map in this table and find.

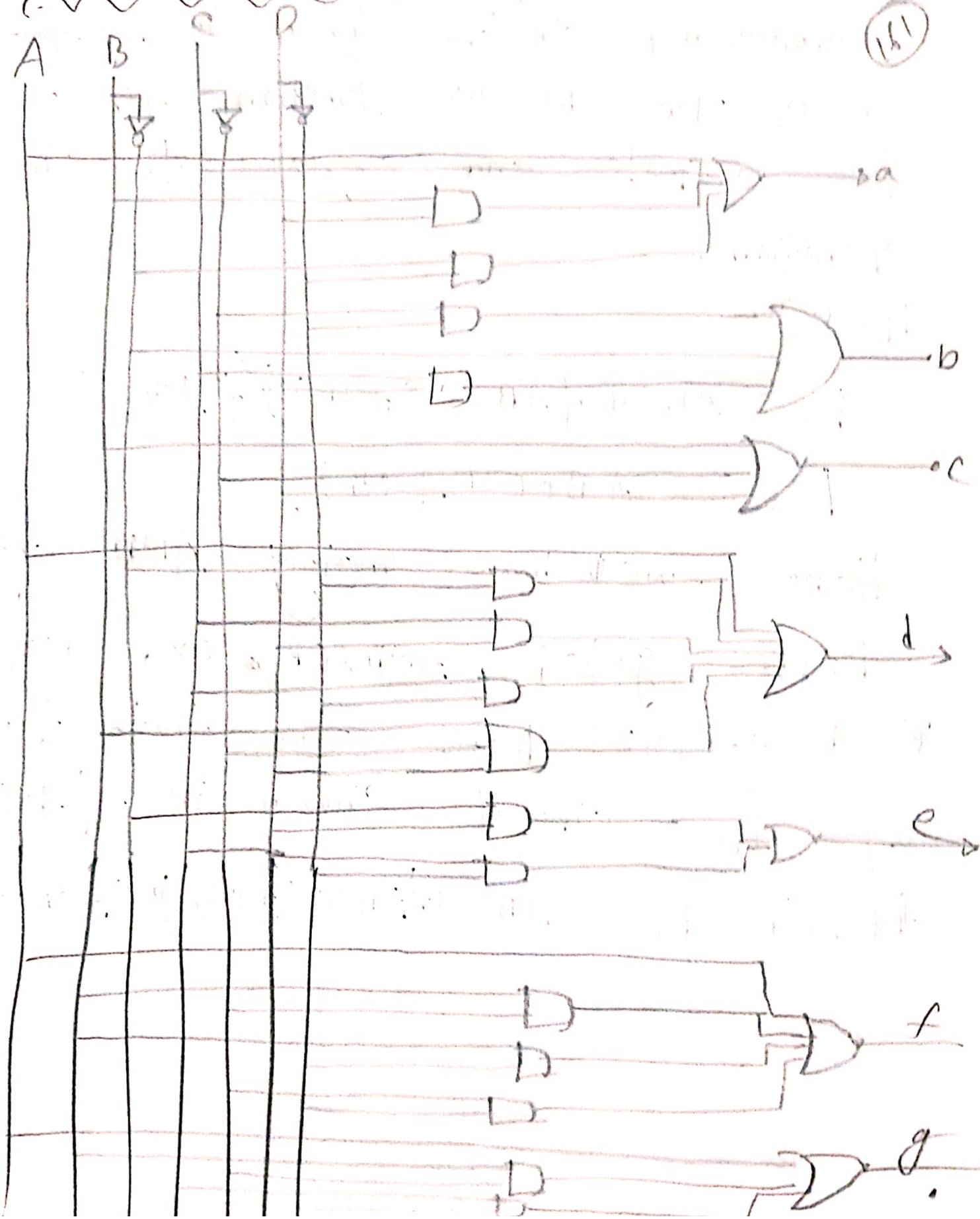
$a = A + C + BD + B'D'$; $b = B' + C'D' + CD$; $c = B + C + D$

$d = A + B'D' + B'C + CD' + BC'D$; $e = B'D + CD'$

$f = A + BC' + BD' + C'D'$; $g = A + BC + B'C + CD$

⇒ now, we can draw circuit diagram.

(151)



* E.x-4.11: Analyze the two-output combinational circuits shown in fig. Pg-15. Obtain the Boolean functions for the two outputs and explain the circuit operation.

Soln:

$F_1 = ABC + \{AB + C(A+B)\}' \{A+B+C\}$

$F_2 = AB + C(A+B)$.

Here, we have ~~mainly AND~~ only Basic gates AND, OR & and NOT

* E.x-4.12: Derive the truth table of the circuit shown in fig. Pg-15

for $F_1 = ABC + \{AB + C(A+B)\}' (A+B+C)$

A	B	C	$AB + C(A+B)$	$A+B+C$	ABC	F1
0	0	0	1	0	0	1
0	0	1	1	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0

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

⇒ for $F_2 = AB + C(A+B)$

A	B	C	A+B	C(A+B)	AB	AB+C(A+B)	F_2
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	1	0	0	0	0
0	1	1	1	1	0	1	1
1	0	0	1	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

* Ex-4.13: Using the block diagram method, convert the logic diagram of fig 4-8 to a NAND implementation.

Soln: Simplify using k-map and find:

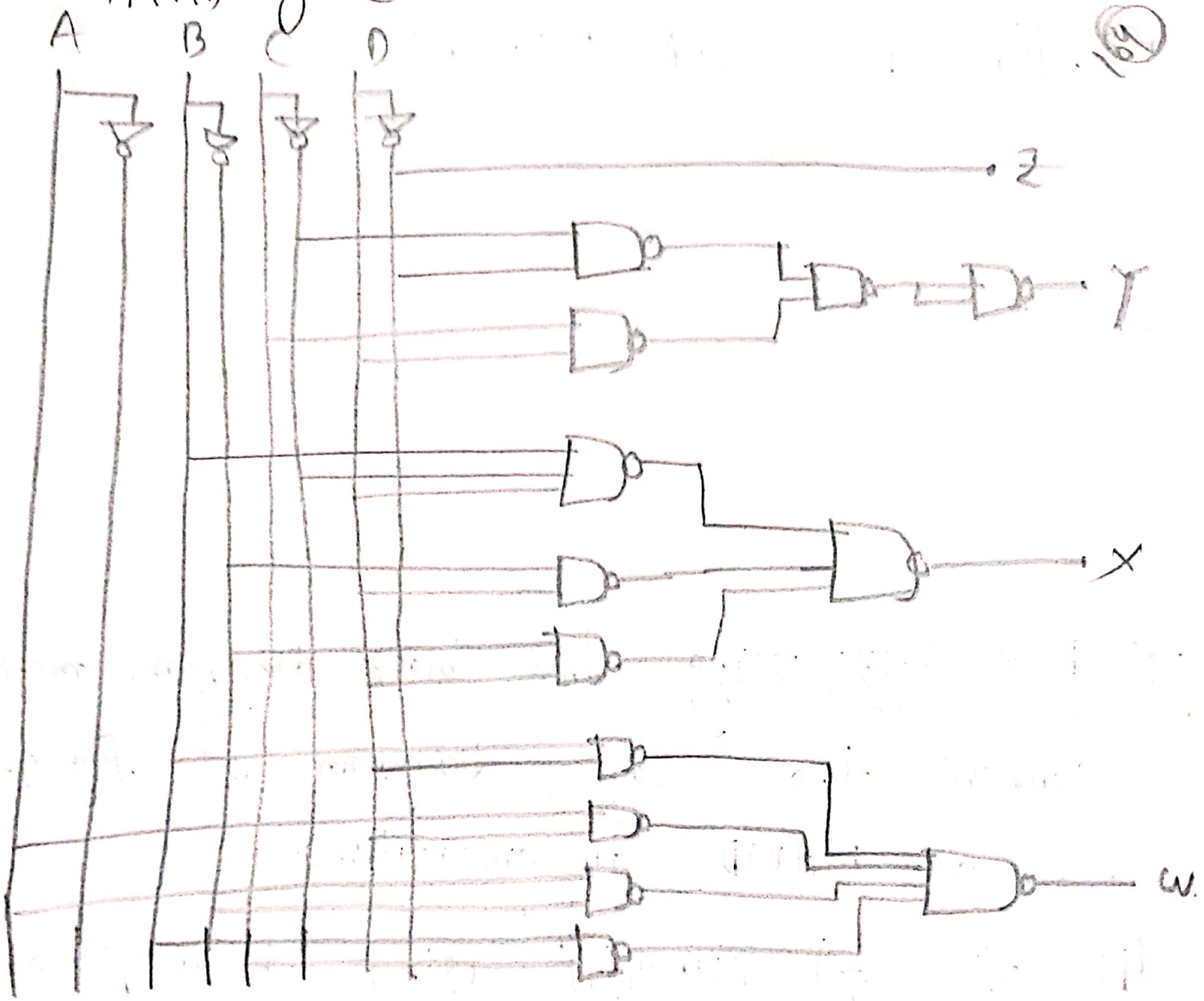
$w = BC'D + B'D + B'C$

$x = (BC'D)' \cdot (B'D)' \cdot (B'C)'$

$y = (C'D) + (CD')'$

$z = D'$

⇒ Now, we can implement it using NAND gate.



* Ex-4.14: Repeat problem 4-15 for NOR implementation.

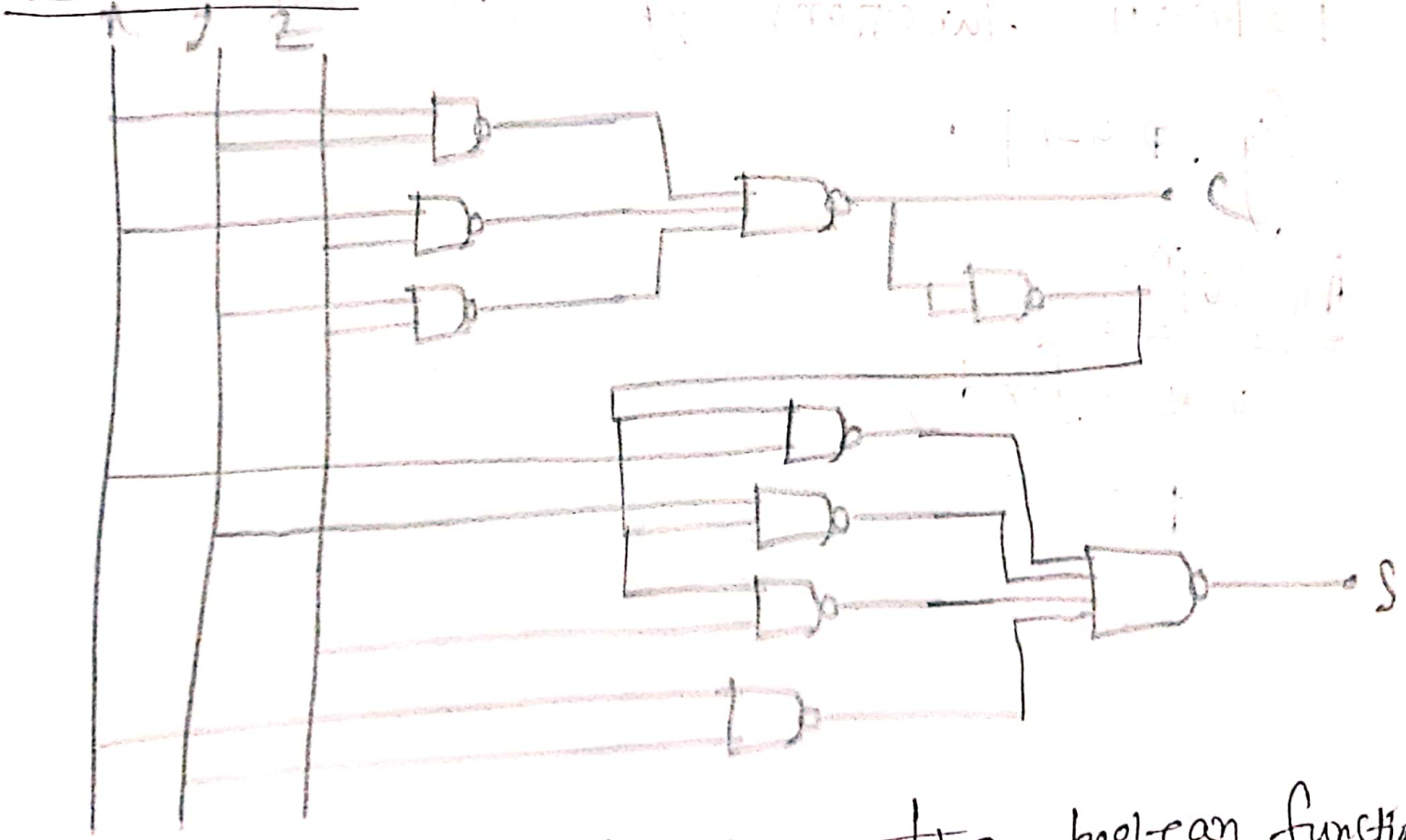
Soln: Full adder circuit.

* Ex-4.15: obtain the NAND logic diagram of a full adder from the Boolean functions:

$$C = xy + xz + yz$$

$$S = C'(x + y + z) + xyz$$

Soln:



* Ex-4.16: Determine the boolean function for the output F of the circuit in fig pg-20. obtain an equivalent

circuit with fewer NOR gates.

Soln: From the above figure we find the function;

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

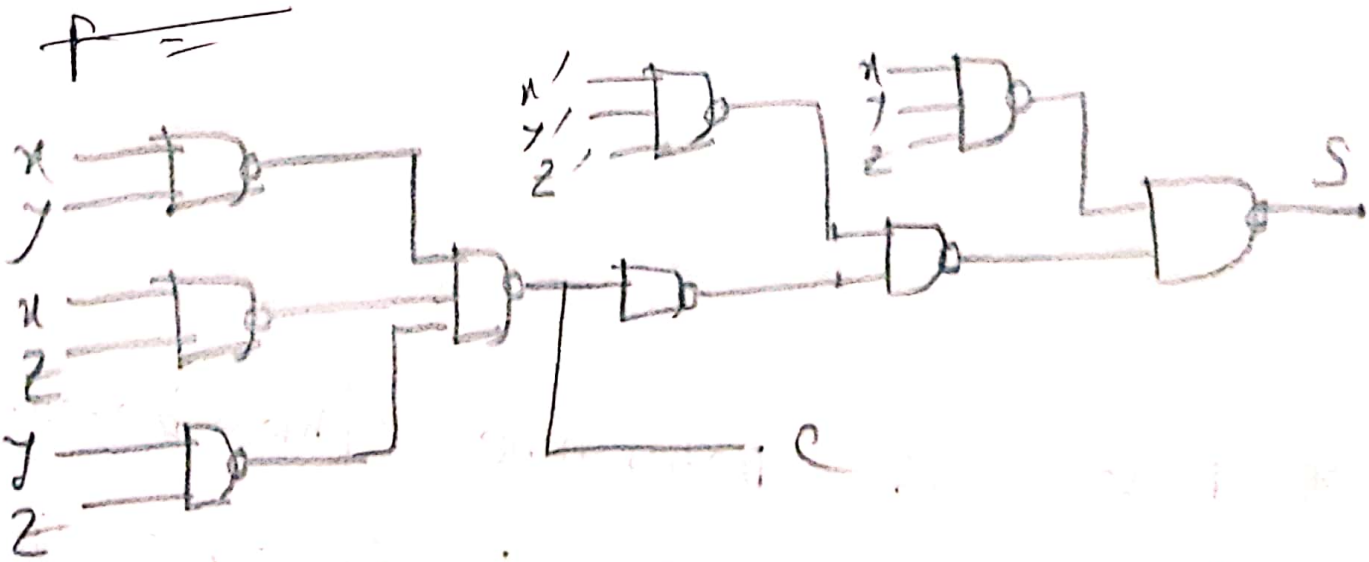
(6)

* E.x-4.17: Determine the output Boolean functions of the circuits in fig. #4-21.

Soln:

we find,

F =



* Ex-4.18: obtain the truth table for the circuits in fig. Pg-21.

Soln: from the circuit we find.

$$F = ABC' + A'B + B'$$

⇒ truth table:

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

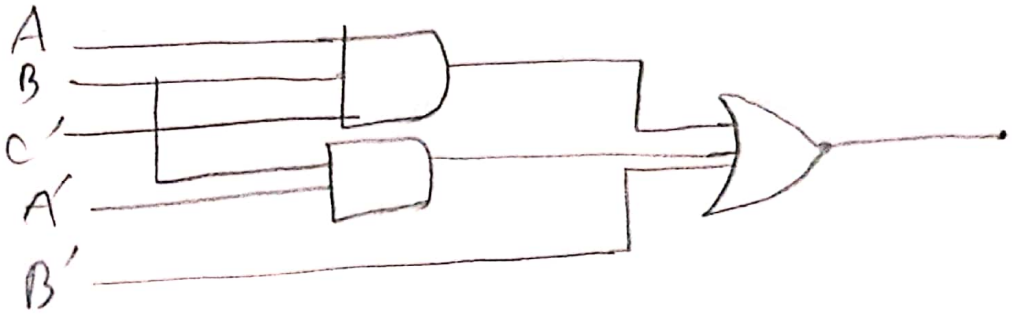
* Ex-4.19: Obtain the equivalent AND-OR logic diagram of Fig. Pg-21(a).

Soln: Full adder, F1 is the sum and F2 is the carry.

* E.X-4.20: obtain the equivalent AND-OR logic diagram of fig P4-4(b)

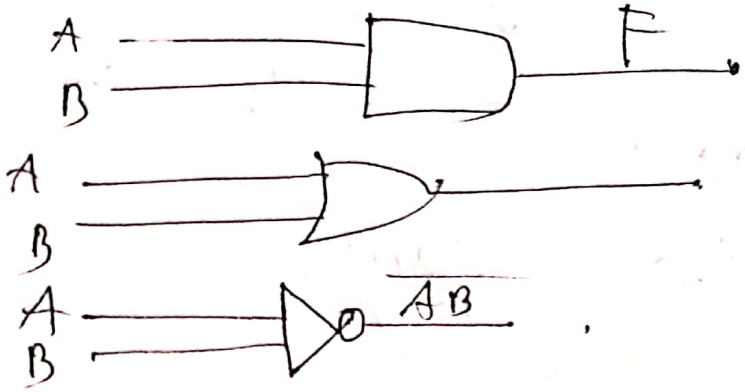
we find the final expression from (4.18):

$$F = ABC' + A'B + B'$$



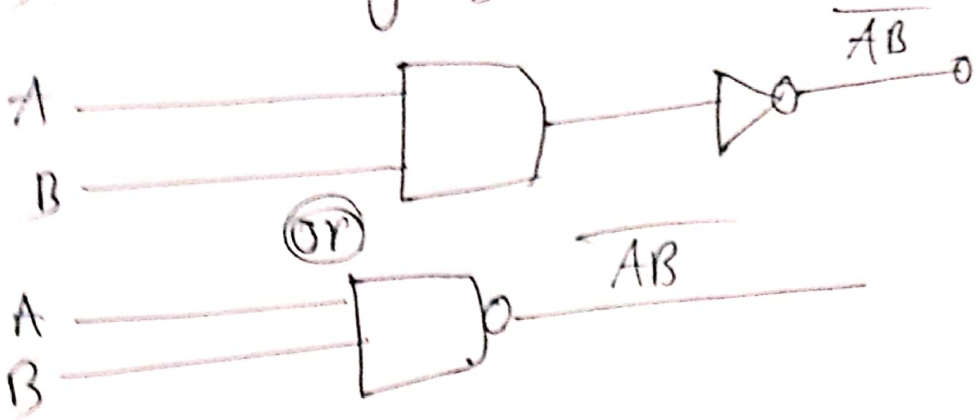
* E.X-4.21: obtain the logic diagram of a two-input equivalence function using.

a) AND, OR & NOT Gates:



(b) NOR gates.

(c) NAND gates:



* Ex-4.22: Show that the circuit in fig. 4-21(b) is an Ex-OR.

Soln:

$$F = ((((BP(A+B)')P(A+P(A+B)')'+C)'$$
$$+((B+(A+B)')+(A+P(A+B)')'+C)'+(BP(A+P(B+$$
$$,')')')'+(((BP(A+B)')'+(A+P(A+B)')'+C)'$$
$$+C)'')'.$$

Now, we can simplify it and find Ex-OR operation.

* Ex-4.23: Show that $A \odot B \odot C \odot D = \Sigma(0,3,5,6,9,10,12,15)$.

We can prove it using truth table:

A	B	C	D	output
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The sum of minterms expression for the rows identified is:

$$F = A'B'C'D' + A'B'CD' + A'BC'D + A'BCD + AB'C'D + AB'CD + ABC'D + ABCD.$$

using the shorthand notation for minterms we can write this as:

$$F = \Sigma(0, 3, 5, 6, 9, 10, 12, 15).$$

(17)

* Ex:- 4.24: Design a combinational circuit that converts a four bit reflected code number (Table - 1.9) to a four-bit binary number. Implement the circuit with Ex-OR gates.

Soln:

⇒ Truth table:

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1

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

(12)

⟹ For simplify: we use k-map and find:

$W = A$; $X = A'B + AB' = A \oplus B$;

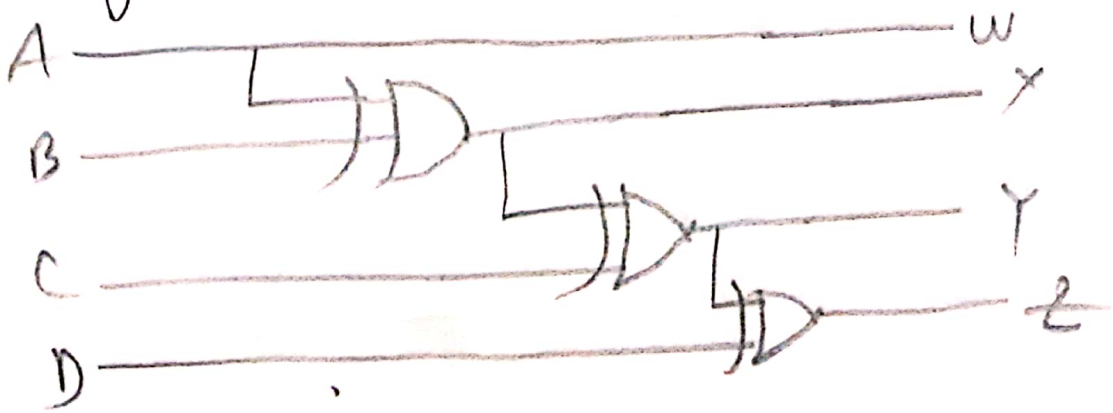
$Y = A'B'C + A'BC' + AB'C' = A \oplus B \oplus C$

$= X \oplus C$

$Z = A \oplus B \oplus C \oplus D$

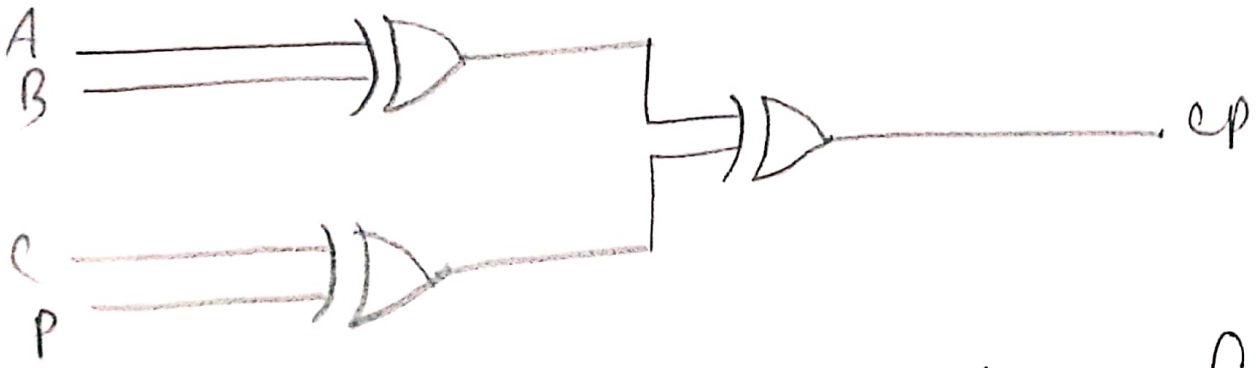
$= Y \oplus C$.

⟹ Logic Diagram:



* Ex-4.25: Design a combinational circuit to check for even parity of four bits. A logic-1 output is required when the four bits do not constitute an even parity.

Soln:



* Ex-4.26: Implement the four Boolean functions listed using three half-adder circuits (Fig. 4-26).

Soln:

$$D = A \oplus B \oplus C$$
$$E = A'BC + AB'C$$
$$F = ABC' + (A' + B')C$$
$$G = ABC.$$

* Ex-4.27: Design a combinational circuit which accepts a two-bit number and equal to the cube of the input number.

Soln: The possible ~~output~~ inputs for a two-bit binary number are 00, 01, 10, and 11. The cube of each of these numbers is as follows

00 cubed = 0000
01 cubed = 0001
10 cubed = 1000
11 cubed = 1011

⇒ truth table:

A	B	output
0	0	0000
0	1	0001
1	0	1000
1	1	1011

now we can draw the circuit diagram.

* Ex-4.28: Implement a combinational circuit which convert four bit excess-3 code to four bit BCD code.

Soln: To get BCD code $S_3 S_2 S_1 S_0$ from this excess-3 code we will subtract 3 from it.

we will carry out this subtraction using twos complement method.

$A_3 A_2 A_1 A_0 - (3)_{10} = A_3 A_2 A_1 +$ 1's complement of $3_{10} + 1$

$= A_3 A_2 A_1 A_0 + (1100)_2 + (1)_2$

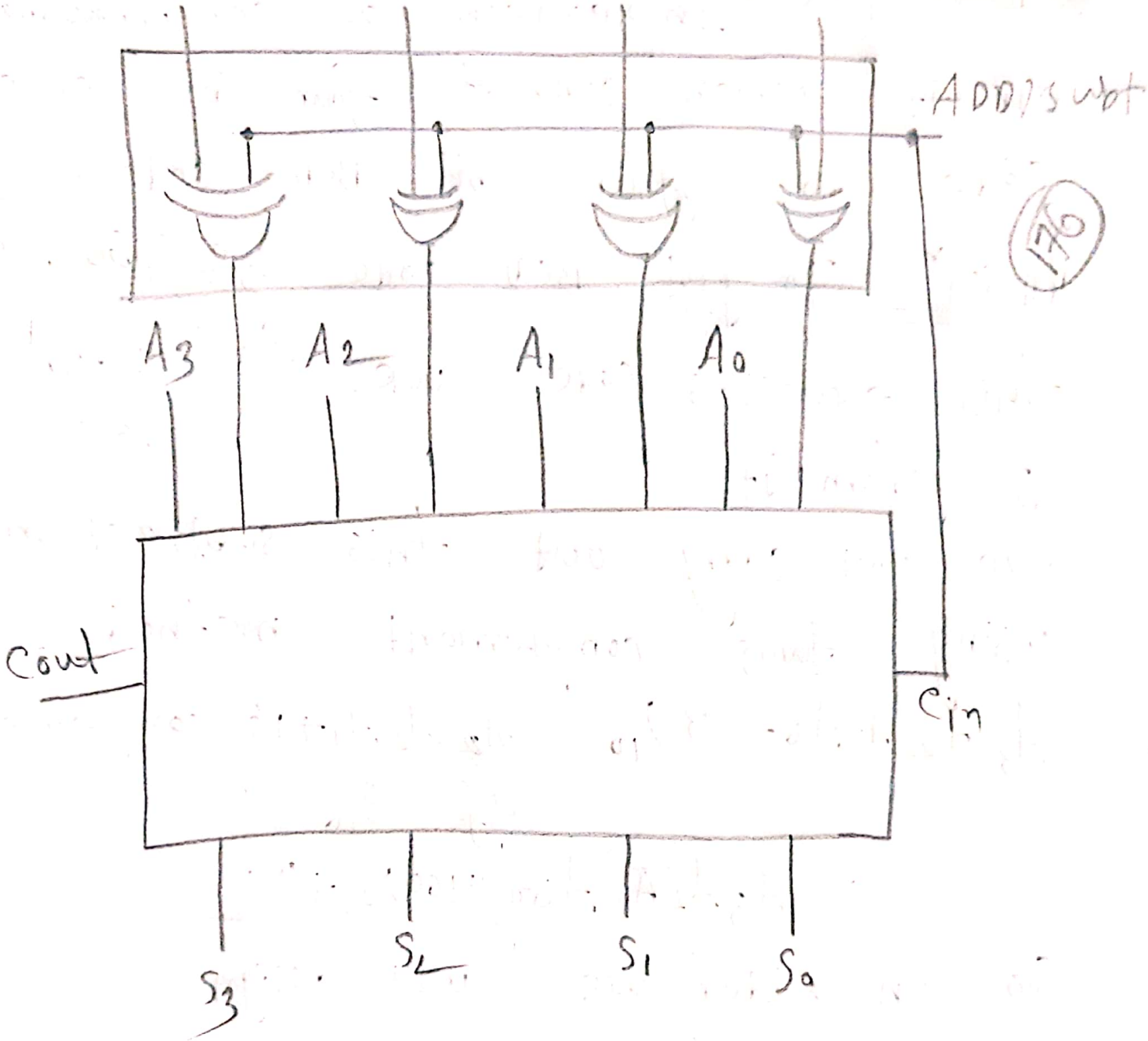
So in adder we will assign

$B_3 B_2 B_1 B_0 = (1100)_2$

And $C_{in} = (1)_2$

And the output $S_3 S_2 S_1 S_0$ will give BCD code.

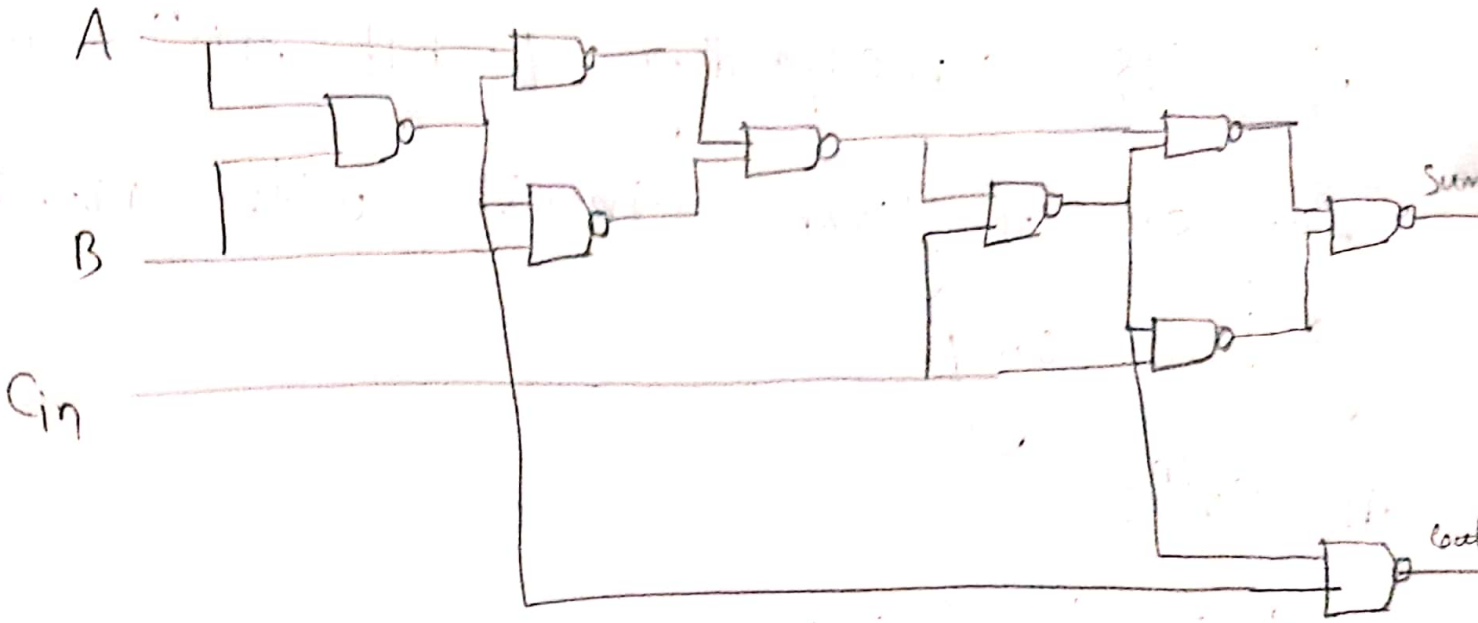
The other way to do this is by using following 4 bit Adder / Subtractor circuit.



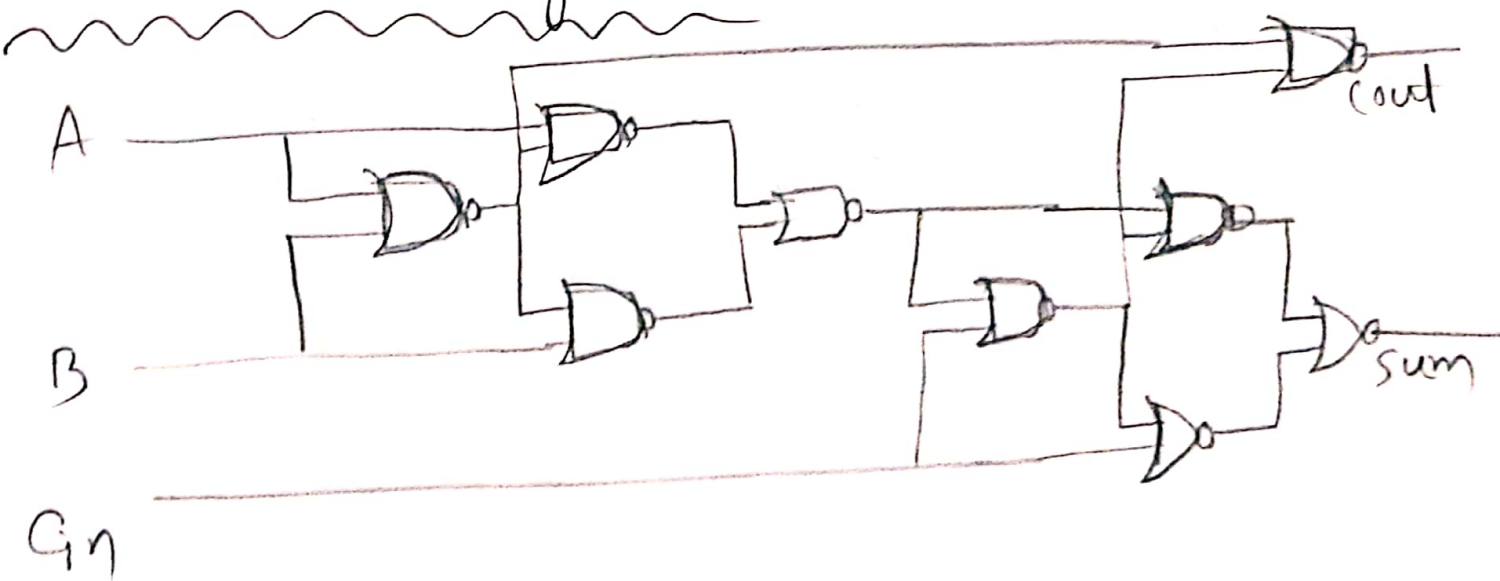
* Ex-9.29: Implement full adder with the help of
(a) NAND gates
(b) NOR gates.

Soln:

⇒ with NAND gates:



⇒ with NOR gates:



* Ex-4.30: Design a combinational circuit which convert BCD number to its corresponding 7 bit ASCII bit. Implement your circuit with NAND gates only.

Soln:

⇒ Truth table

BCD	ASCII
0000	0110000
0001	0110001
0010	0110010
0011	0110011
0100	0110100
0101	0110101
0110	0110110
0111	0110111
1000	0111000
1001	0111001

now, we can implement circuit diagra using NAND gate.

* Ex-4.31: What are universal gates? Why are they called so?

Soln: Universal gates are logic gates that can be used to implement any other type of logic gate. The two most commonly used universal gates are the NAND gate and the NOR gate.

NAND gates and NOR gates are called universal gates because they can be used to implement any other type of logic gate, including AND, OR, NOT and XOR gates. This is because both NAND and NOR gates can be used to perform the functions of all other logic gates.