ST.JOSEPH'S COLLEGE OF ENGINEERING
&
ST.JOSEPH'S INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
&
DEPARTMENT OF INFORMATION TECHNOLOGY
Chennai-119

CS 6201: Digital Principles and Systems Design

ASSIGNMENT-I QUESTIONS

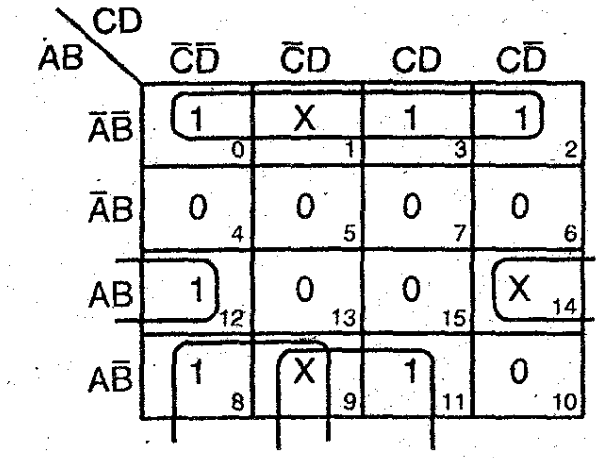
PART A:

1. What is the abbreviation of ASCII and EBCDIC code?
2. Give the different types of binary codes.
3. Define De Morgan's Theorem?
4. What is meant by tabulation method?
5. Define Karnaugh Map? State the limitations of Karnaugh map.
6. Simplify the function using K-Map $F(A, B) = \sum m(0, 2, 3)$
7. Simplify the function using K-Map $F(A, B) = \pi M(0, 1, 3)$
8. How many bits are required to represent the decimal numbers in the range 0 to 999 using straight binary code? Using BCD codes?
9. Show that the excess-3 code is self-complementing.
10. What is meant by weighted and non-weighted code?
11. Write the two properties of Gray code & mention the application of Gray code
12. What is meant by Fast Adder?
13. Define distributive law.
14. Give the canonical product form of $F=x_1'x_2'x_3+x_1'x_2x_3'+x_1x_2'x_3'+x_1'x_2x_3$
15. What is meant by prime implicant?
16. What are Universal Gates? Why are they called so?
17. Define positive logic and negative logic system.
18. Define bit, byte and nibble.
19. Define Combinational circuit?
20. Define SSI and MSI.

PART B:

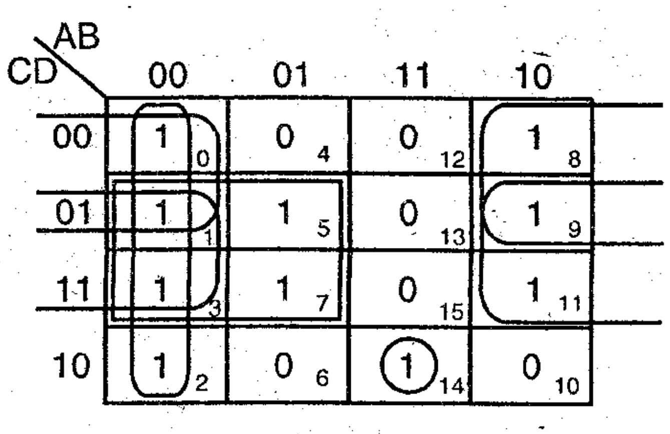
1. a) (i) solve the following using k-map.

1) f (A, B, C, D) = Σm (0, 2, 3, 8, 11, 12) + d (1, 9, 14)



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

2) f (A, B, C, D) = Σm (0, 1, 2, 3, 5, 7, 8, 9, 11,14)



$$Y = \bar{A}\bar{B} + \bar{A}D + \bar{B}D + \bar{B}\bar{C} + ABC\bar{D}$$

ii) Simplify the expression Z =AB + AC + ABC (AB + C). Implement using Minimum number of NAND gates.

Z = AB + AC + ABC (AB + C)

= AB + AC + ABC. AB + ABCC

= AB + AC + ABC + ABC

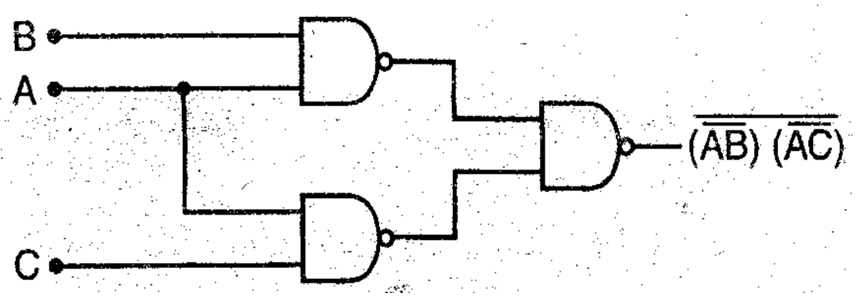
= AB + AC + ABC

=AB (1 + C) + AC

= AB + AC

Implementation using minimum number & NAND gates:

$$\overline{\overline{Z}} = \overline{\overline{AB+AC}} = \overline{(\overline{AB})(\overline{AC})}$$



2. Simplify the function f(A, B, C, D) = (0,1,2,3,5,7,8,10,12,13,15), using tabulation method (Quine–McCluskey method).

First List

	A	B	C	D	
0	0	0	0	0	✓
1	0	0	0	1	✓
2	0	0	1	0	✓
8	1	0	0	0	✓
3	0	0	1	1	✓
5	0	1	0	1	✓
10	1	0	1	0	✓
12	1	1	0	0	✓
7	0	1	1	1	✓
13	1	1	1	1	✓
15	1	1	1	1	✓

Second List

	A	B	C	D	
0,1	0	0	0	–	✓
0,2	0	0	–	0	✓
0,8	–	0	0	0	✓
1,3	0	0	–	1	✓
1,5	0	–	0	1	✓
2,3	0	0	1	–	✓
2,10	–	0	1	0	✓
8,10	1	0	–	0	✓
8,12	1	–	0	0	$A\overline{C}\,\overline{D}$
3,7	0	–	1	1	✓
5,7	0	1	–	1	✓
5,13	–	1	0	1	✓
12,13	1	1	0	–	$AB\overline{C}$
7,15	–	1	1	1	✓
13,15	1	1	–	1	✓

Third List

	A	B	C	D	
0, 1 2 3	0	0	–	–	$\overline{A}\,\overline{B}$
0, 2 1 3	0	0	–	–	
0, 2, 8, 10	–	0	–	0	$\overline{B}\,\overline{D}$
0, 8, 2, 10	–	0	–	0	
1, 3, 5, 7	0	–	–	1	$\overline{A}D$
1, 5, 3, 7	0	–	–	1	
5, 7, 13, 15	–	1	–	1	BD
5, 13, 7, 15	–	1	–	1	

The prime implicants are: $\overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}D + BD + A\overline{C}\,\overline{D} + AB\overline{C}$

3. Simplify the following expressions by Quine–McCluskey method.

$$\overline{A}.B.C+\overline{A}.\overline{B}.D+A.\overline{C}.D+B.\overline{C}.\overline{D}+\overline{A}.B.\overline{C}.D$$

In the first step, we write the expanded version of the given expression. It can be written as follows:

$$\overline{A}.B.C.D+\overline{A}.B.C.\overline{D}+\overline{A}.\overline{B}.C.D+\overline{A}.\overline{B}.\overline{C}.D+A.B.\overline{C}.D+A.\overline{B}.\overline{C}.D+A.B.\overline{C}.\overline{D}$$
$$+\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.\overline{C}.D$$

The formation of groups, the placement of terms in different groups and the first-round matching are shown as follows:

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

A	B	C	D	
0	0	0	1	✓
0	1	0	0	✓
0	0	1	1	✓
0	1	0	1	✓
0	1	1	0	✓
1	0	0	1	✓
1	1	0	0	✓
0	1	1	1	✓
1	1	0	1	✓

A	B	C	D	
0	0	–	1	✓
0	–	0	1	✓
–	0	0	1	✓
0	1	0	–	✓
0	1	–	0	✓
–	1	0	0	✓
0	–	1	1	✓
0	1	–	1	✓
–	1	0	1	✓
0	1	1	–	✓
1	–	0	1	✓
1	1	0	–	✓

The second round of matching begins with the table shown on the previous page. Each term in the first group is compared with every term in the second group. For instance, the first term in the first group 00−1 matches with the second term in the second group 01−1 to yield 0−−1, which is recorded in the table shown below. The process continues until all terms have been compared for a possible match. Since this new table has only one group, the terms contained therein are all prime implicants.

In the present example, the terms in the first and second tables have all found a match. But that is not always the case.

A	B	C	D
0	—	—	1
—	—	0	1
0	1	—	—
—	1	0	—

The next table is what is known as the prime implicant table. The prime implicant table contains all the original terms in different columns and all the prime implicants recorded in different rows as shown below:

0001	0011	0100	0101	0110	0111	1001	1100	1101	
√	√		√		√				0- -1
√			√			√		√	- -01
		√	√	√	√				01- -
		√	√				√	√	-10-

Therefore, the minimized expression = $\overline{A}.D+\overline{C}.D+\overline{A}.B+B.\overline{C}$.

4. a) (i)Simplify the four variable switching functions. Draw the circuit of the minimal expression using only NAND gates.

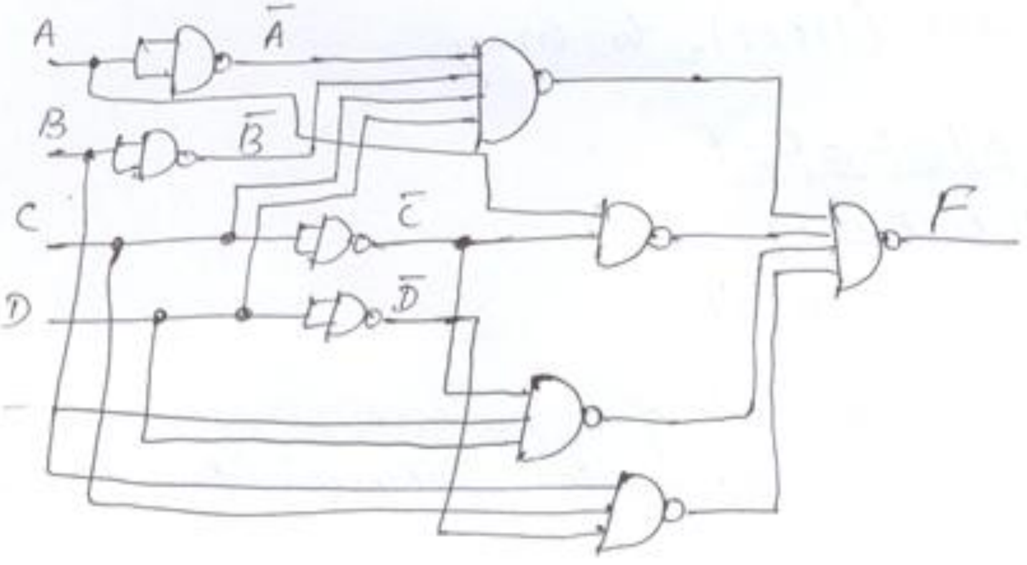
F(A,B,C,D) = ∑m(3,5,6,8,9,12,13,14)

(a) $F(A,B,C,D) = \sum m(3,5,6,8,9,12,13,14)$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	0
$\bar{A}B$	0	1	0	1
AB	1	1	0	1
$A\bar{B}$	1	1	0	0

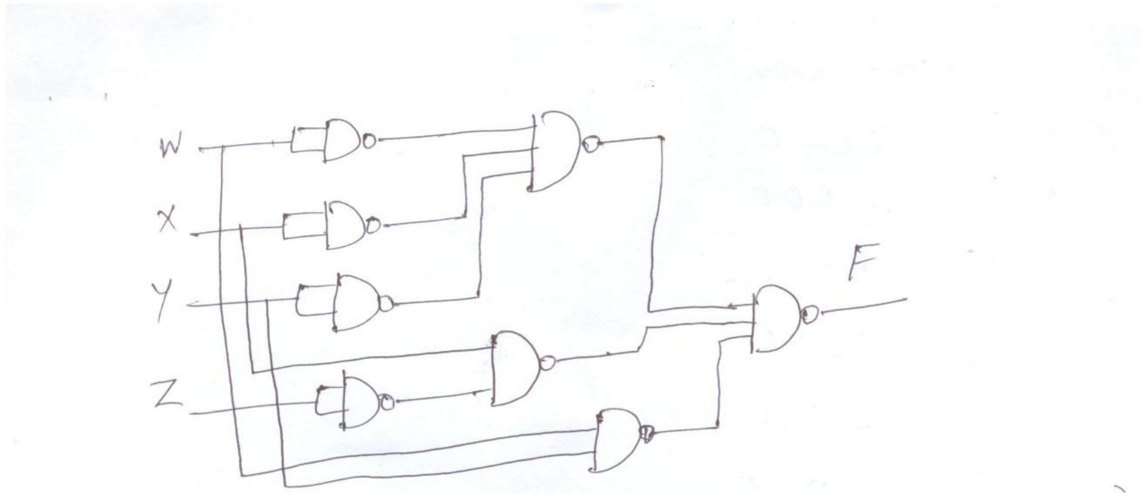
$F(A,B,C,D) = \bar{A}\bar{B}CD + B\bar{C}D + BC\bar{D} + A\bar{C}$

Implementation using NAND Gates



b) (i) Find a minimal sum of products representation for

F(W,X,Y,Z) = ∑m(1,4,6,10,12,14) + d(0,11,13,15) using K-Map. Draw the circuit of the Minimal expression using only NAND gates



$F(W,X,Y,Z) = \sum m(1,4,6,10,12,14) + d(0,11,13,15)$

WX \ YZ	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$	X	1	0	0
$\bar{W}X$	1	0	0	1
WX	1	X	X	1
$W\bar{X}$	0	0	X	1

$F = \bar{W}\bar{X}\bar{Y} + X\bar{Z} + WY$

5. a) Simplify using K-Map

F(A,B,C,D) = ∏M(3,6,8,10,12,13,15)

a) $F(A,B,C,D) = \prod M(3,6,8,10,12,13,15)$

AB \ CD	$C+D$	$C+\bar{D}$	$\bar{C}+\bar{D}$	$\bar{C}+D$
$A+B$	1	1	0	1
$A+\bar{B}$	1	1	1	0
$\bar{A}+\bar{B}$	0	0	0	1
$\bar{A}+B$	0	1	1	0

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

b) i)Perform subtraction on the given unsigned numbers using the 10's complement.

1) 6428 – 3409 **2) 125-1800**

b) i) 1. 6428 − 3409 = 3019

Take 9's complement of subtrahend.

```
  9999
- 3409
  6590
+    1
  6591 → 10's complement.

  6428
+ 6591
[1]3019   Discard carry.
```

Result : 3019

(2) 125 − 1800 = −1675

Take 9's complement of subtrahend.

```
  9999
- 1800
  8199
+    1
  8200 – 10's complement

  8200
+ 0125
  8325  No carry
```

Take 10's complement of result

```
  9999
- 8325
  1674
+    1
  1675
```

Assign negative sign

Result = −1675

ii) Perform subtraction on the given unsigned binary numbers using the 2's complement.

1) 10011 – 10001

2)100010 – 100011

(ii) 1. 10011 − 10001 = 00010

Take 2's complement of 10001

```
  01110
      1
  01111
+ 10011
[1]00010  Discard carry
```

Result = 00010

2. 100010 − 100011 = −000001

Take 2's complement of 100011

```
  011100          011101
+      1        + 100010
  011101          111111  No carry
```

Take 2's complement of result

```
  000000
+      1
  000001
```

Assign negative sign

−000001

6. State the postulates and theorems of Boolean algebra.

<u>Principle of duality</u>:

1. Interchanging the OR and AND operations of the expression. 2. Interchanging the 0 and 1 elements of the expression. 3. Not changing the form of the variables.

<u>Commutative Law</u>

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

(b) $A B = B A$

<u>Associative Law</u>

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

(b) $(A B) C = A (B C)$

<u>Distributive Law</u>

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

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

<u>Identity Law</u>

(a) $A + A = A$

(b) $A A = A$

<u>Negation Law</u>

(a) $(\overline{A}) = \overline{A}$

(b) $(\overline{\overline{A}}) = A$

<u>Redundant Law</u>

(a) $A + A B = A$

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

<u>Postulates:</u>

(a) $0 + A = A$

(b) $1 A = A$

(c) $1 + A = 1$

(d) $0 A = 0$

(e) $\overline{A} + A = 1$

(f) $\overline{A} A = 0$

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

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

<u>De Morgan's Theorem</u>

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

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

7. (i) Reduce AB + (AC)' + AB’C (AB + C)

$AB + (AC)' + AB'C\,(AB + C) = AB + (AC)' + AAB'BC + AB'CC$

$= AB + (AC)' + AB'CC \quad [A.A' = 0]$

$= AB + (AC)' + AB'C \quad [A.A = 1]$

$= AB + A' + C' = AB'C \quad [(AB)' = A' + B']$

$= A' + B + C' + AB'C \quad [A + AB' = A + B]$

$= A' + B'C + B + C' \quad [A + A'B = A + B]$

$= A' + B + C' + B'C$

$= A' + B + C' + B'$

$= A' + C' + 1$

$= 1 \quad [A + 1 = 1]$

ii) Show that $(X + Y' + XY)(X + Y')(X'Y) = 0$

$(X + Y' + XY)(X + Y')(X'Y) = (X + Y' + X)(X + Y')(X' + Y)$

$[A + A'B = A + B]$

$= (X + Y')(X + Y')(X'Y) \quad [A + A = 1]$

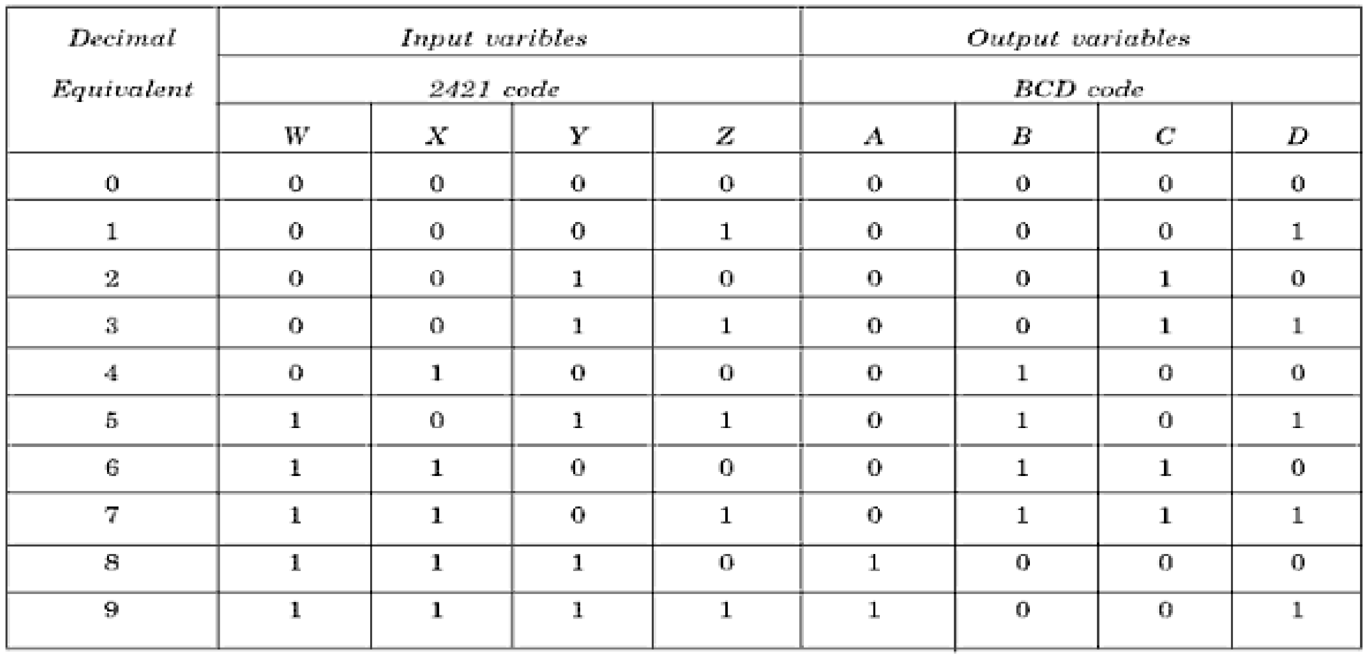
$= (X + Y')(X'Y) \quad [A.A = 1]$

$= X.X' + Y'.X'.Y$

$= 0 \quad [A.A' = 0]$

8. Design a combinational circuit for converting 2421 code to BCD code.

Both the 2421 code and BCD code are 4-bit codes and represent the decimal equivalents 0 to 9. To design the converter circuit for the above, first the truth table is prepared with the input variables W, X, Y, and Z of 2421 code, and the output variables A, B, C, and D. Karnaugh maps to obtain the simplified expressions of the output functions Unused combinations are considered as don’t-care condition.



Decimal Equivalent	*Input varibles*				*Output variables*			
	2421 code				*BCD code*			
	W	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	0	1	0	1
6	1	1	0	0	0	1	1	0
7	1	1	0	1	0	1	1	1
8	1	1	1	0	1	0	0	0
9	1	1	1	1	1	0	0	1

	Y′Z′	Y′Z	YZ	YZ′
W′X′				
W′X		X	X	X
WX			1	1
WX′	X	X		X

	Y′Z′	Y′Z	YZ	YZ′
W′X′				
W′X	1	X	X	X
WX	1	1		
WX′	X	X	1	X

	Y′Z′	Y′Z	YZ	YZ′
W′X′			1	1
W′X		X	X	X
WX	1	1		
WX′	X	X		X

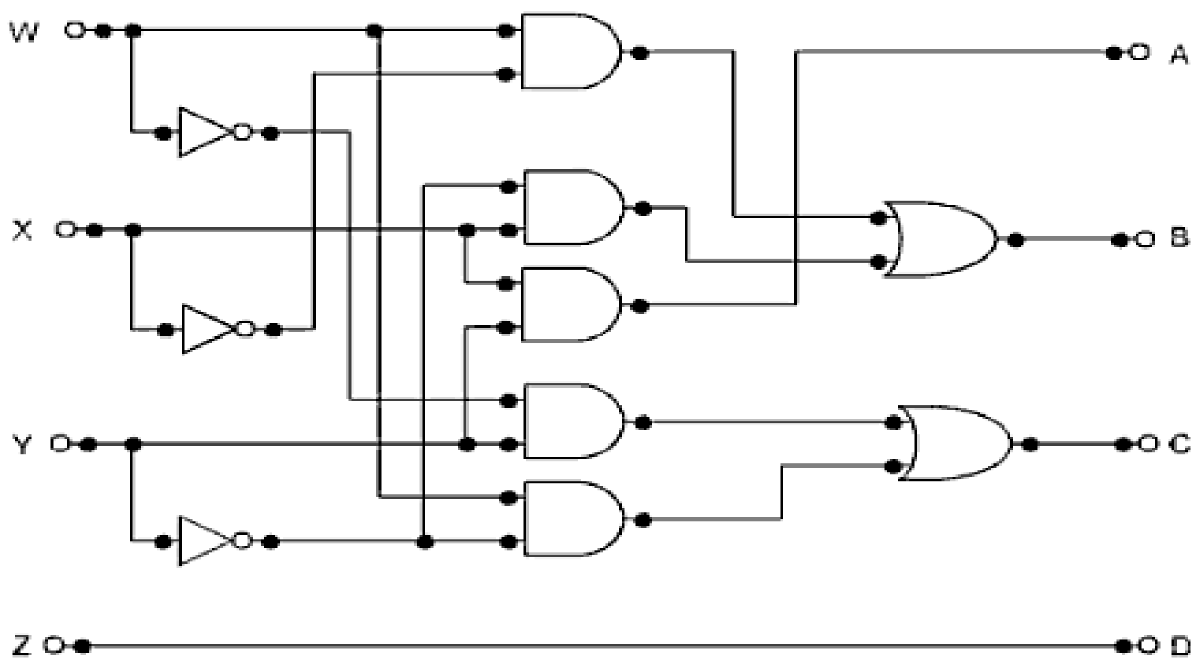
	Y′Z′	Y′Z	YZ	YZ′
W′X′		1	1	
W′X		X	X	X
WX		1	1	
WX′	X	X	1	X

The Boolean expressions for the output functions are

A = XY B = XY′+WX′

C = W′Y + WY′ D = Z.

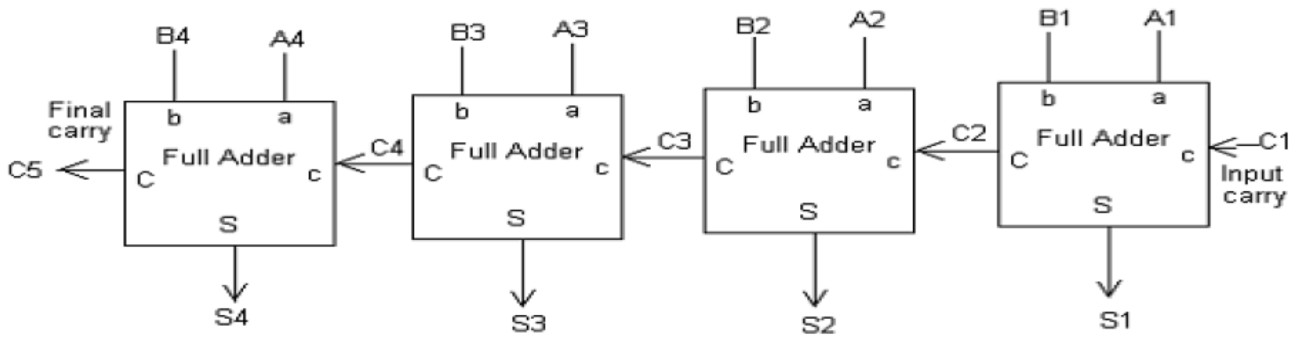
LOGIC DIAGRAM:



9.Design a Binary Adder and Subtractor.

Binary Adder:

In this adder we need n full adders for n bit adder. In this adder we use the n full adders in cascaded from to implement the ripple carry adder. This type of adder is also called carry propagation adder. The circuit for 4-bit parallel adder is as follow:



For example:

To add A= 1011 and B= 0011

Subscript i:	3 2 1 0
Input carry:	0 1 1 0 Ci
Augend:	1 0 1 1 Ai
Addend:	0 0 1 1 Bi

Sum:	1 1 1 0 Si
Output carry:	0 0 1 1 Ci+1

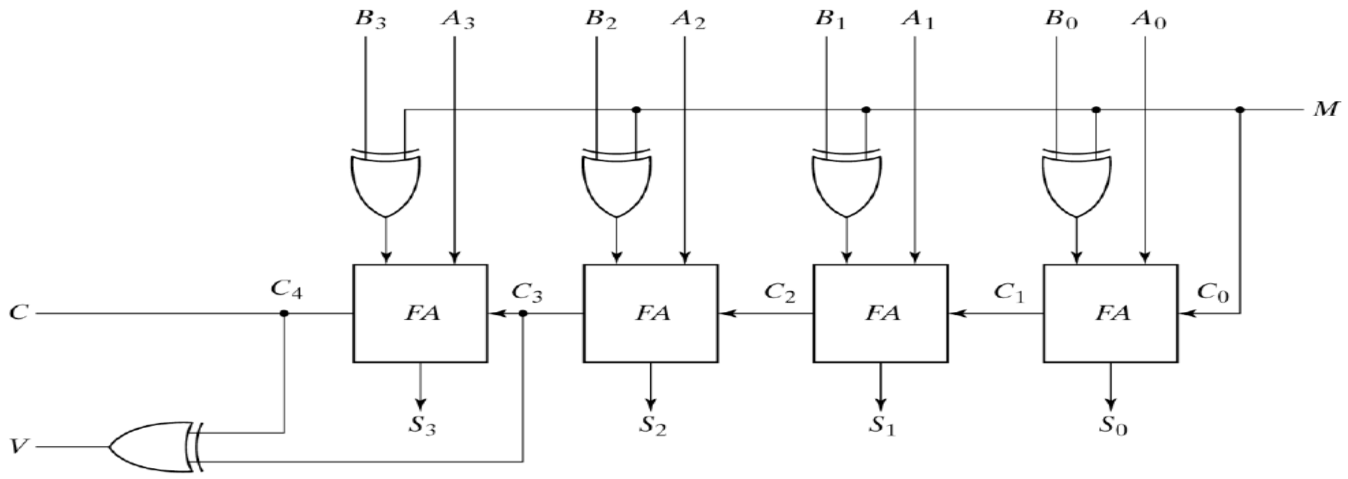
Binary Subtractor:

The subtraction A – B can be done by taking the 2‟s complement of B and adding it to A because A- B = A + (-B).It means if we use the inverters to make 1‟s complement of B (connecting each Bi to an inverter) and then add 1 to the least significant bit (by setting carry C0 to 1) of binary adder, then we can make a binary subtractor.

Binary Adder Subtractor:

• The addition and subtraction can be combined into one circuit with one common binary adder (see next slide).

• The mode M controls the operation. When M=0 the circuit is an adder when M=1 the circuit is subtractor. It can be done by using exclusive-OR for each Bi and M. Note that 1 ⊕ x = x‟ and 0 ⊕ x = x.

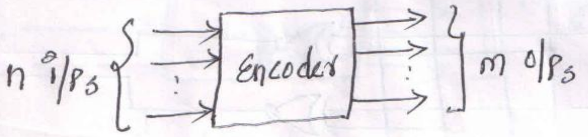


10. Design a combinational circuit for ENCODER and DECODER.

Encoder:-

↳ An encoder is a digital circuit that performs the inverse operation of a decoder.

↳ An encoder is a combinational logic circuit that converts an active input signal into a coded output signal.



$\boxed{2^n : n}$

Octal to binary Encoder (8:3)

it accepts eight inputs and produces 3-bit ouput code corresponding to the activated input.

Truth Table:-

Inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0
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

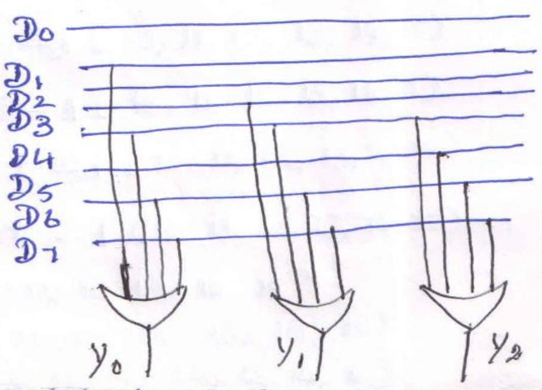
Boolean expression

$Y_0 = D_1 + D_3 + D_5 + D_7$

$Y_1 = D_2 + D_3 + D_6 + D_7$

$Y_2 = D_4 + D_5 + D_6 + D_7$

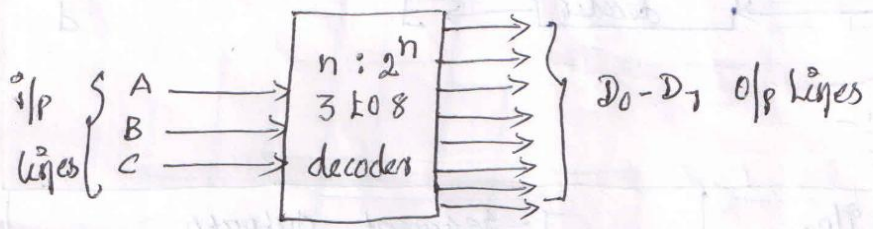
Logic diagram:-



Decoder:-

↳ A decoder is a logic circuit that converts an n-bit binary input code (data) into 2^n output lines, such that each output line will be activated for only one of the possible combinations of inputs.

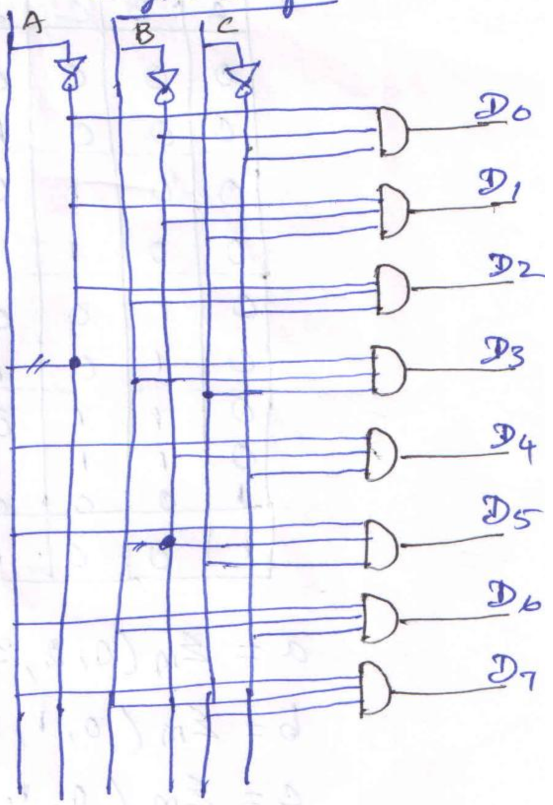
3 to 8 Decoder:-



Truth Table:-

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

Logic diagram



Boolean Expression:-

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

$D_1 = \bar{A}\bar{B}C$

$D_2 = \bar{A}B\bar{C}$

$D_3 = \bar{A}BC$

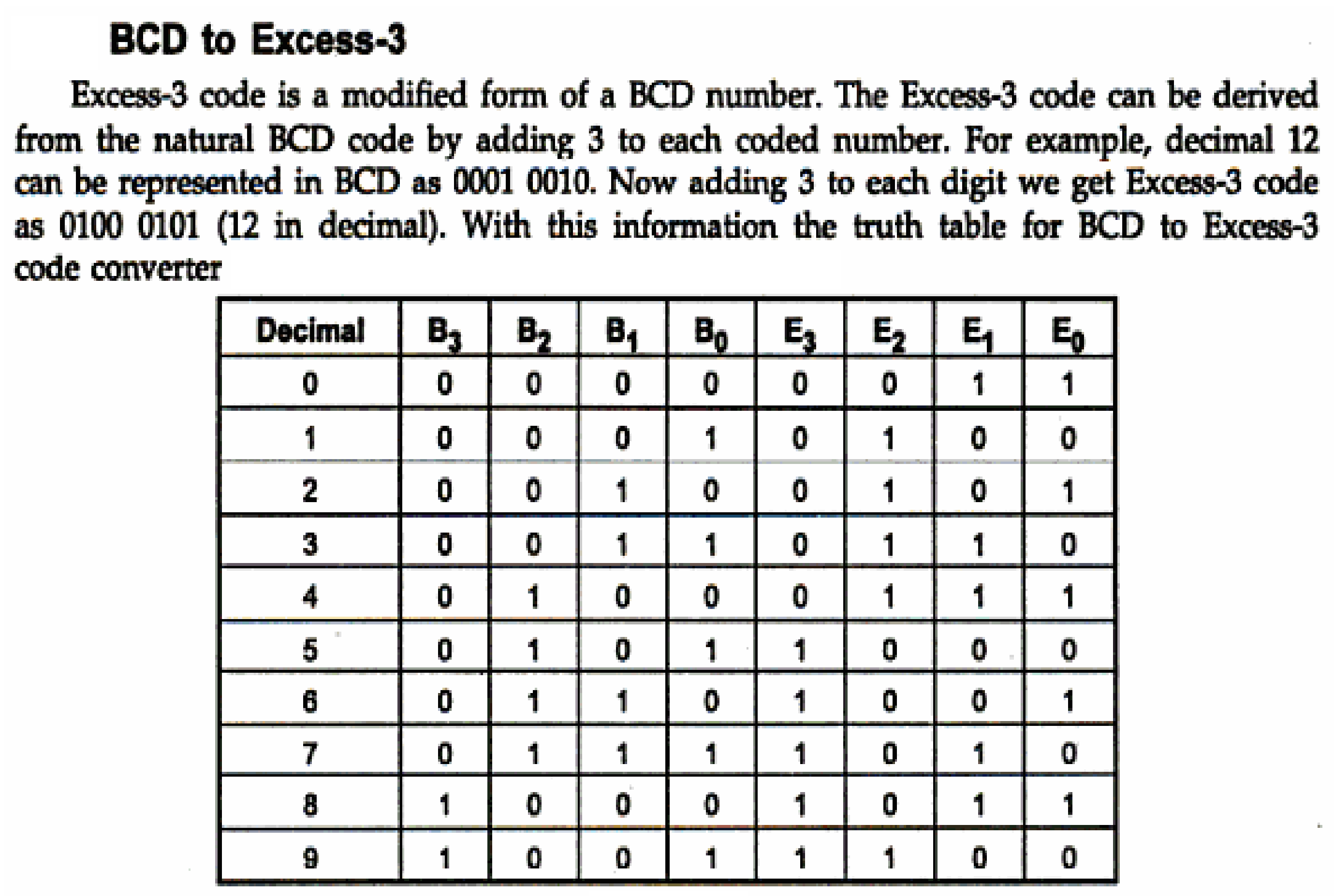
$D_4 = A\bar{B}\bar{C}$

$D_5 = A\bar{B}C$

$D_6 = AB\bar{C}$

$D_7 = ABC$

11. Design a combinational circuit that converts a decimal digit from BCD to Excess 3 code.

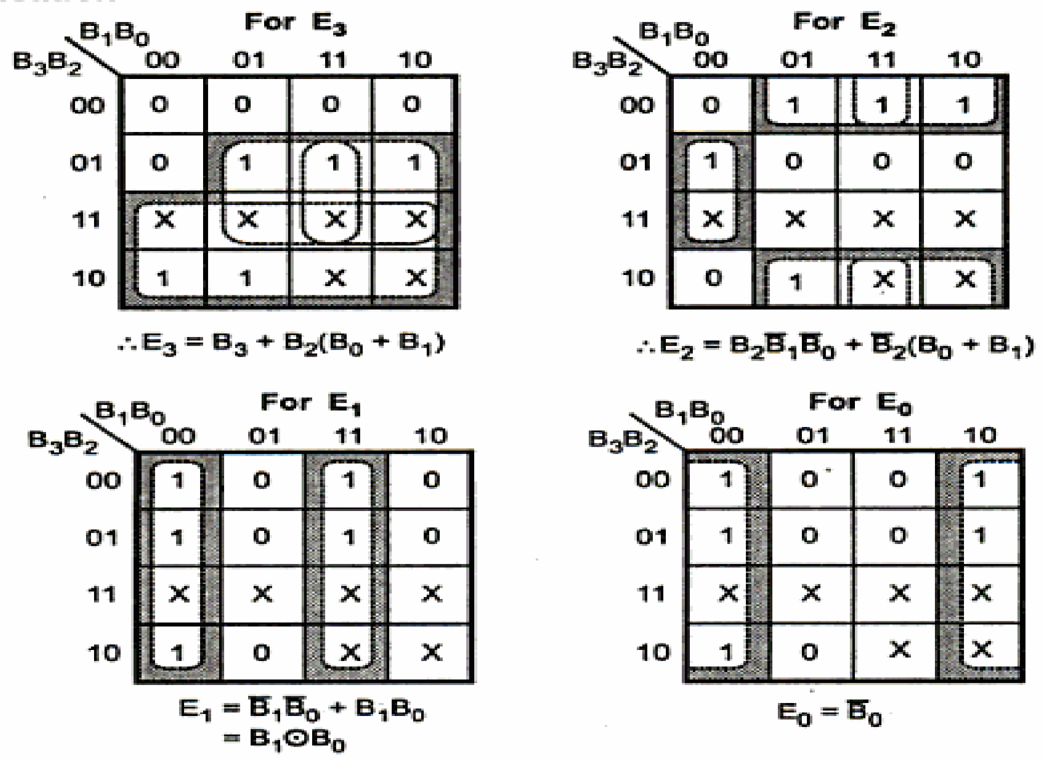


BCD to Excess-3

Excess-3 code is a modified form of a BCD number. The Excess-3 code can be derived from the natural BCD code by adding 3 to each coded number. For example, decimal 12 can be represented in BCD as 0001 0010. Now adding 3 to each digit we get Excess-3 code as 0100 0101 (12 in decimal). With this information the truth table for BCD to Excess-3 code converter

Decimal	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

K-map simplification



For E_3

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$\therefore E_3 = B_3 + B_2(B_0 + B_1)$

For E_2

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$\therefore E_2 = B_2\overline{B}_1\overline{B}_0 + \overline{B}_2(B_0 + B_1)$

For E_1

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

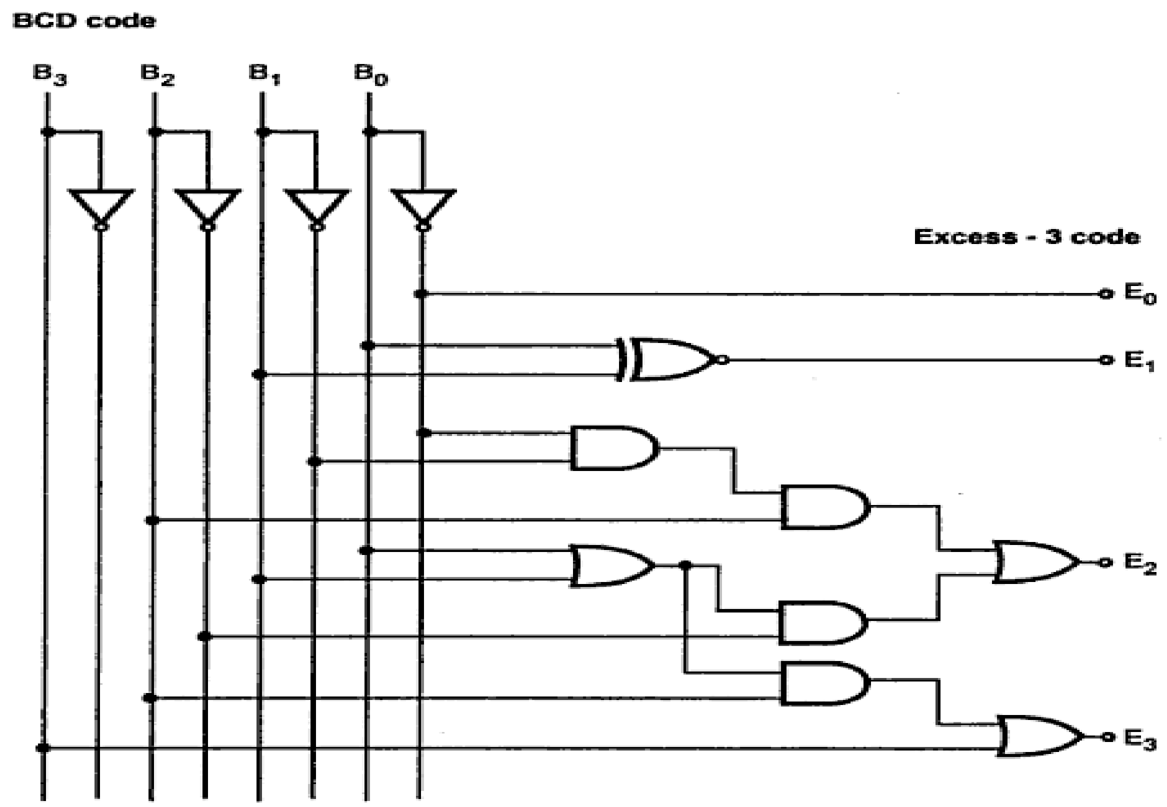
$E_1 = \overline{B}_1\overline{B}_0 + B_1B_0$
$= B_1 \odot B_0$

For E_0

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_0 = \overline{B}_0$

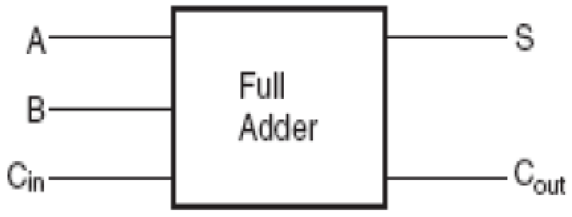
Logic diagram



12.Design a combinational circuit of Full adder& Subtractor.

A full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a SUM and a CARRY output. Such a building block becomes a necessity when it comes to adding binary numbers with a large number of bits. The full adder circuit overcomes the limitation of the half-adder, which can be used to add two bits only. Let us recall the procedure for adding larger binary numbers. We begin with the addition of LSBs of the two numbers. We record the sum under the LSB column and take the carry, if any, forward to the next higher column bits. As a result, when we add the next adjacent higher column bits, we would be required to add three bits if there were a carry from the previous addition. We have a similar situation for the other higher column bits. Also until we reach the MSB. A full adder is therefore essential for the hardware implementation of an adder circuit capable of adding larger binary numbers. A half-adder can be used for addition of LSBs only.

<u>SYMBOL:</u>



TRUTH TABLE:

A	B	C_{in}	SUM (S)	C_{out}
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

Figure shows the truth table of a full adder circuit showing all possible input combinations and corresponding outputs. In order to arrive at the logic circuit for hardware implementation of a full adder, we will firstly write the Boolean expressions for the two output variables, that is, the SUM and CARRY outputs, in terms of input variables. These expressions are then simplified by using any of the simplification techniques described in the previous chapter. The Boolean expressions for the two output variables are given in Equation below for the SUM output (S) and in above Equation for the CARRY output (Cout):

$$S = \overline{A}.\overline{B}.C_{in} + \overline{A}.B.\overline{C}_{in} + A.\overline{B}.\overline{C}_{in} + A.B.C_{in}$$

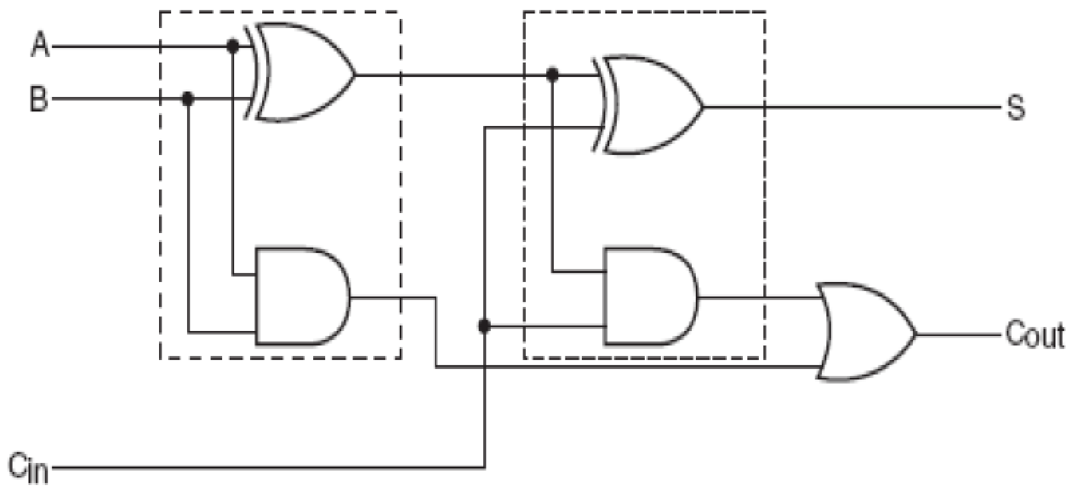
$$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C}_{in} + A.B.C_{in}$$

The next step is to simplify the two expressions.

$$S = \overline{C}_{in}.(\overline{A}.B + A.\overline{B}) + C_{in}.(\overline{\overline{A}.B + A.\overline{B}})$$

$$C_{out} = A.B + C_{in}.(\overline{A}.B + A.\overline{B})$$

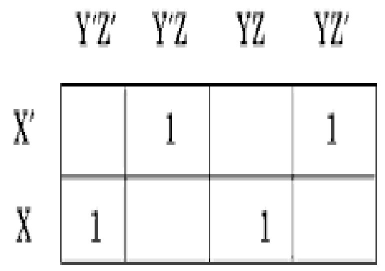
LOGIC DIAGRAM:

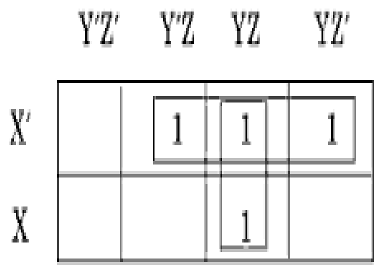


FULL SUBTARCTOR:

A combinational circuit of full-subtractor performs the operation of subtraction of three bits—the minuend, subtrahend, and borrow generated from the subtraction operation of previous significant digits and produces the outputs difference and borrow. Let us designate the input variables minuend as X, subtrahend as Y, and previous borrow as Z, and outputs difference as D and borrow as B. Eight different input combinations are possible for three input variables. The truth table is shown in Figure 5.10(*a*) according to its functions.

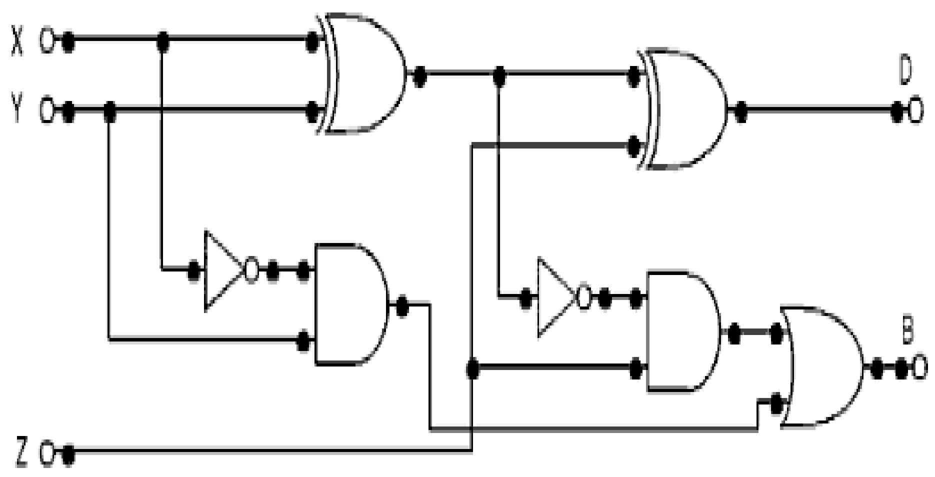
Input variables			*Outputs*	
X	*Y*	*Z*	*D*	*B*
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1





$$D = X'Y'Z + X'YZ' + XY'Z' + XYZ$$
$$= X'(Y'Z + YZ') + X(Y'Z' + YZ)$$
$$= X'(Y \oplus Z) + X(Y \oplus Z)'$$
$$= X \oplus Y \oplus Z$$
$$B = X'Z + X'Y + YZ = X'Y + Z(X' + Y)$$
$$= X'Y + Z(X'Y + X'Y' + XY + X'Y)$$
$$= X'Y + Z(X'Y + X'Y' + XY)$$
$$= X'Y + X'YZ + Z(X'Y' + XY)$$
$$= X'Y + Z(X \oplus Y)'$$

LOGIC DIAGRAM:



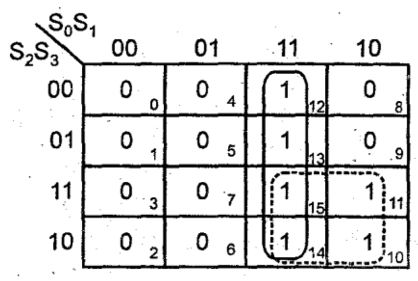
13. Design a combinational circuit to perform BCD addition.

A BCD adder adds two BCD digits and produces a BCD digit If the sum of BCD digits is less than equal to 9 then it is valid BCD form If sum is greater than 9 then 6 (six) i.e. (0110)2 is added in sum to make it valid

Let us 4 bit binary adder A0 A1 A2 A3 and B0 B1 B2 B3 produces sum S0 S1 S2 S3. A combinational circuit is there to check sum is 9 or more than 9 so that six can be added to it. For combination circuit, truth-table is as:

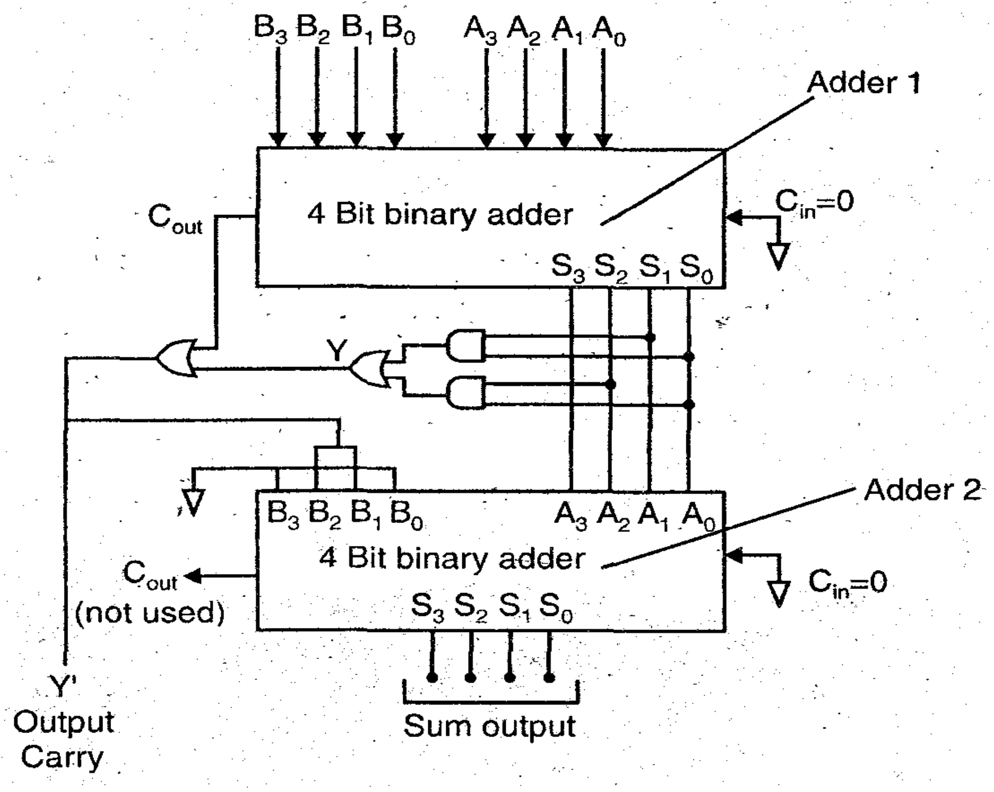
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

K-map



$$Y = S_0S_2 + S_0S_1$$

BCD circuit diagram:

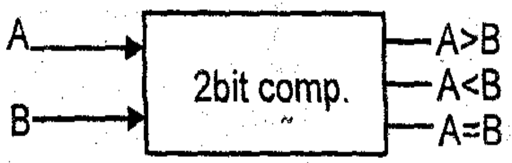


We will get correct BCD sum from output of Adder-2. Which will be in BCD form.

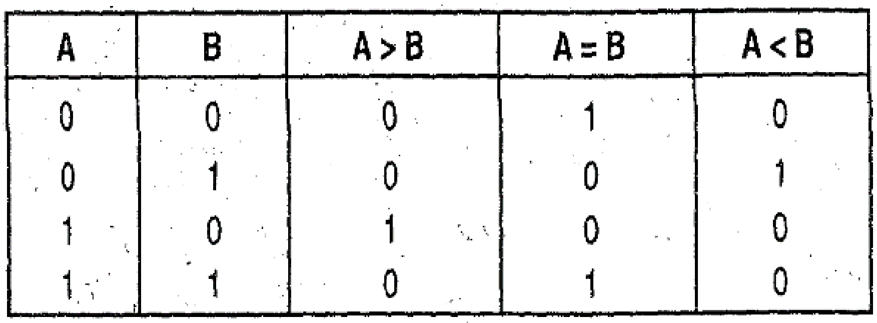
14. (i) what are Magnitude comparators" Explain the design of magnitude comparators with the help of a suitable example

A comparator is a logic circuit used to compare magnitude to two binary numbers or more The result will be, either bits will be equal greater or lesser

SYMBOL:

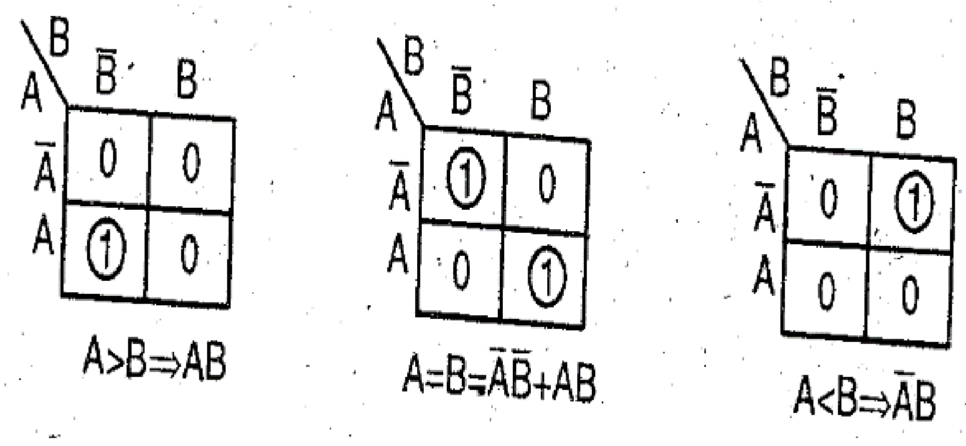


TRUTH TABLE:

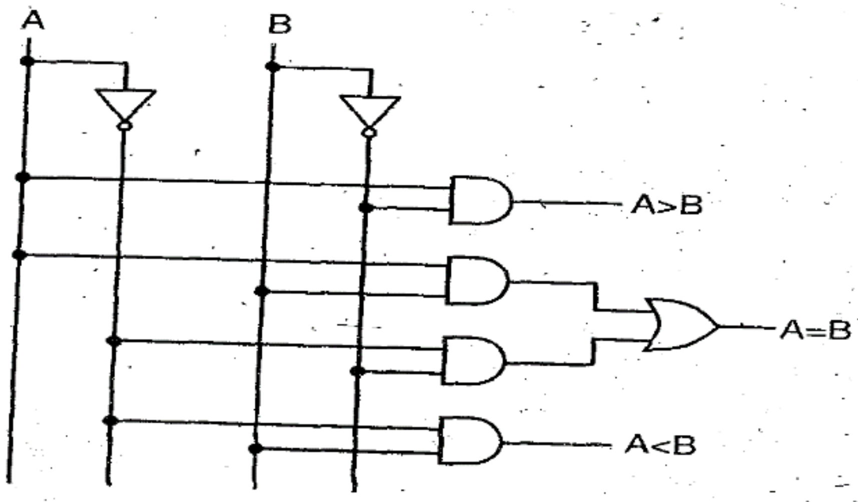


A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

K-MAP:



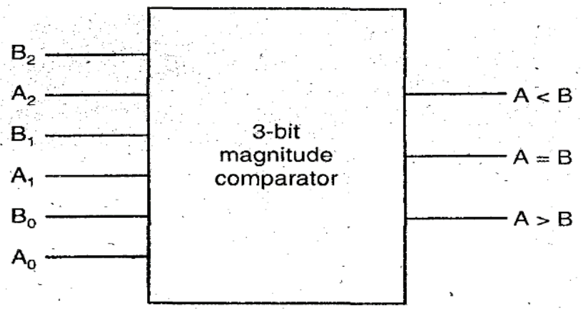
LOGIC DIAGRAM:



Similarly, 7485 is TTL 4 bit magnitude comparator. These inputs can be extended to compare more than 4 bits.

14. (ii) Write note on 3 bit binary magnitude comparator.

SYMBOL:



Where, A = A2 A1 A0 having three bits and

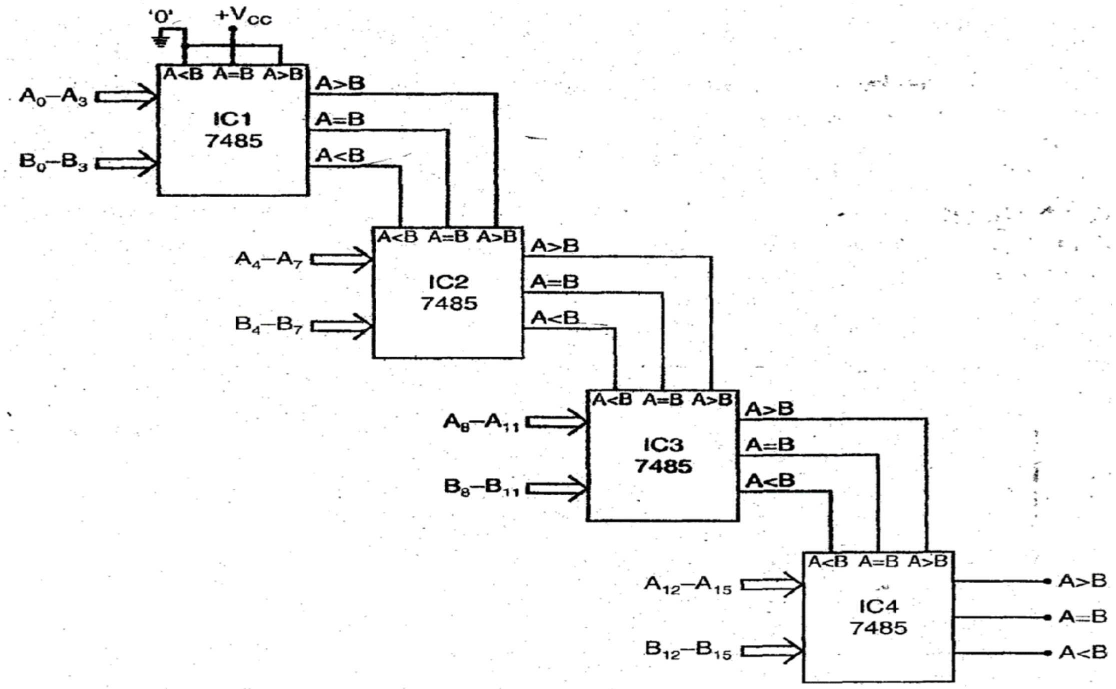
B = B2 B1 B0 also having three bits.

A2 A1 A0 are compared with another three bits i.e. B2 B1 B0

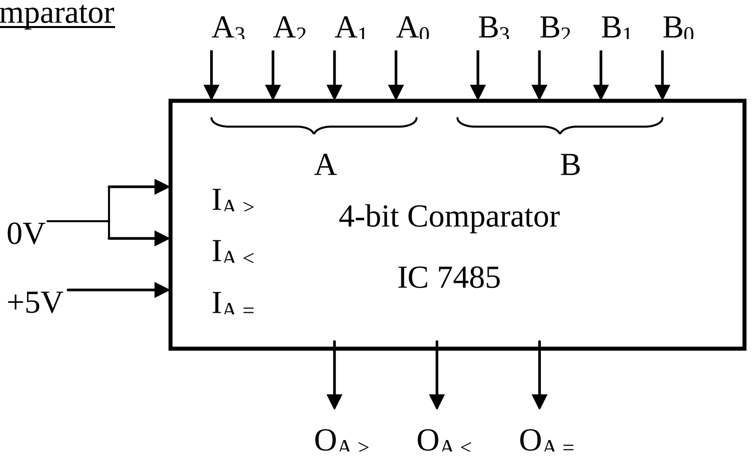
Total combinations are possible. As 2^6 = 64. Thus, six variable k-maps are required to solve for A > B, A = B and A < B.

14. (iii) Construct 16-bit comparator using 4-bit comparator as a building block.

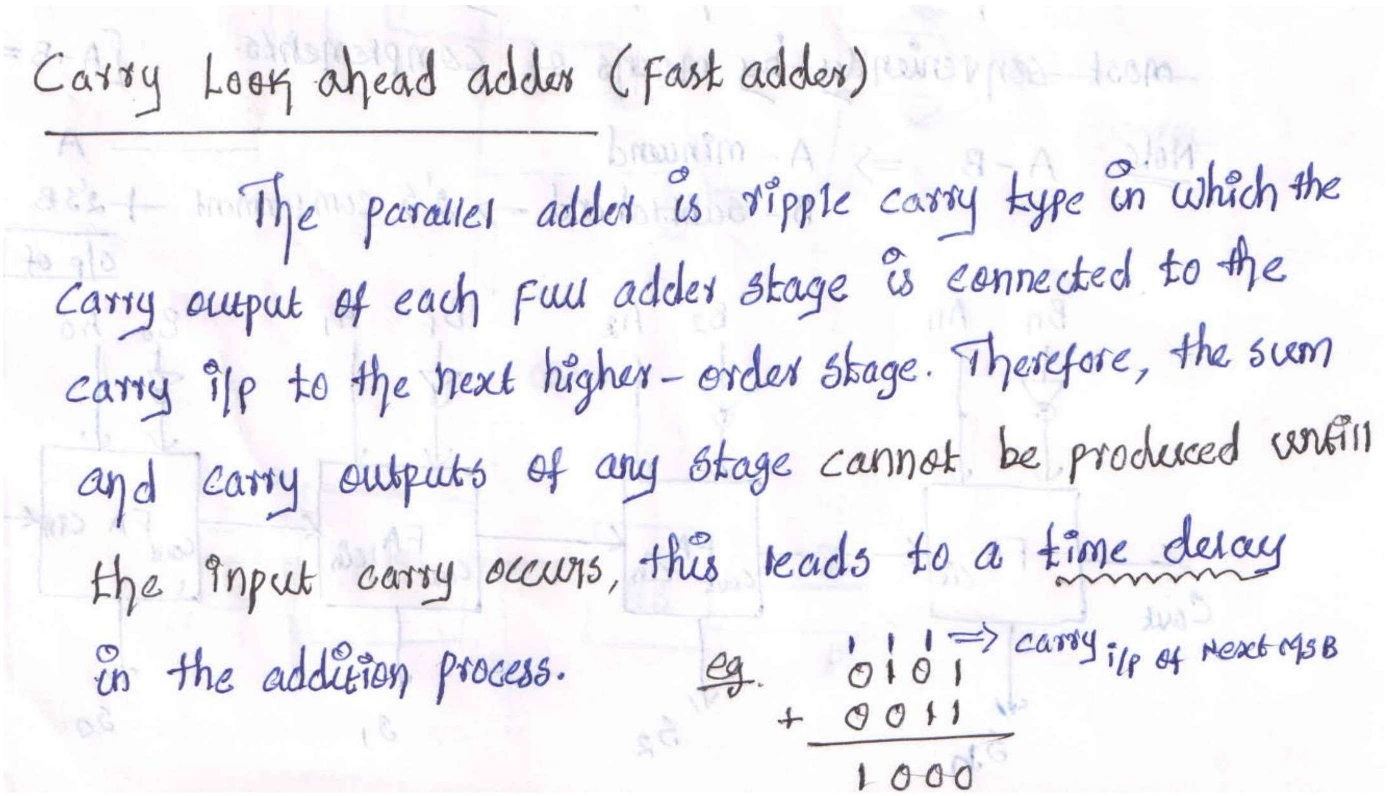
4 bit comparator IC in 7485. It is used for 16 bit comparator. Thus, 4 IC'S are used.



Logic diagram of a 4-bit comparator



15. Discuss the need and working principle of Carry Look ahead adder.



Carry Look ahead adder (Fast adder)

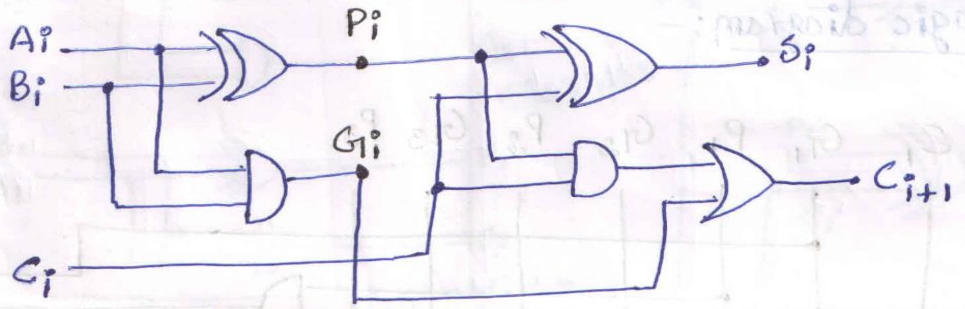
The parallel adder is ripple carry type in which the carry output of each full adder stage is connected to the carry i/p to the next higher-order stage. Therefore, the sum and carry outputs of any stage cannot be produced untill the input carry occurs, this leads to a time delay in the addition process.

eg.

$$\begin{array}{rcccc} & 1 & 1 & 1 & \Rightarrow \text{carry i/p of Next MSB} \\ & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 0 \end{array}$$

↳ A Method of increasing the speed of this process by eliminating inter stage carry delay is called look ahead carry-addition.

↳ it uses 2 functions such as
① carry generate, G_i
② carry propagate, P_i



⇓

$P_i = A_i \oplus B_i$ ⇒ carry propagate } 1st stage
$G_i = A_i B_i$ ⇒ carry generate }

The o/p sum and carry can be expressed as,

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

Boolean function:-

$i=0$, $C_{0+1} = G_0 + P_0 C_0$
$\therefore C_1 = G_0 + P_0 C_0 \longrightarrow$ stage ①

$i=1$, $C_{1+1} = G_1 + P_1 C_1$
$\therefore C_2 = G_1 + P_1 C_1$
$= G_1 + P_1 [G_0 + P_0 C_0]$
$= G_1 + P_1 G_0 + P_1 P_0 C_0 \longrightarrow$ ②

$i = 2,$ $C_{2+1} = G_2 + P_2 C_2$

$C_3 = G_2 + P_2 [G_1 + P_1 C_1]$

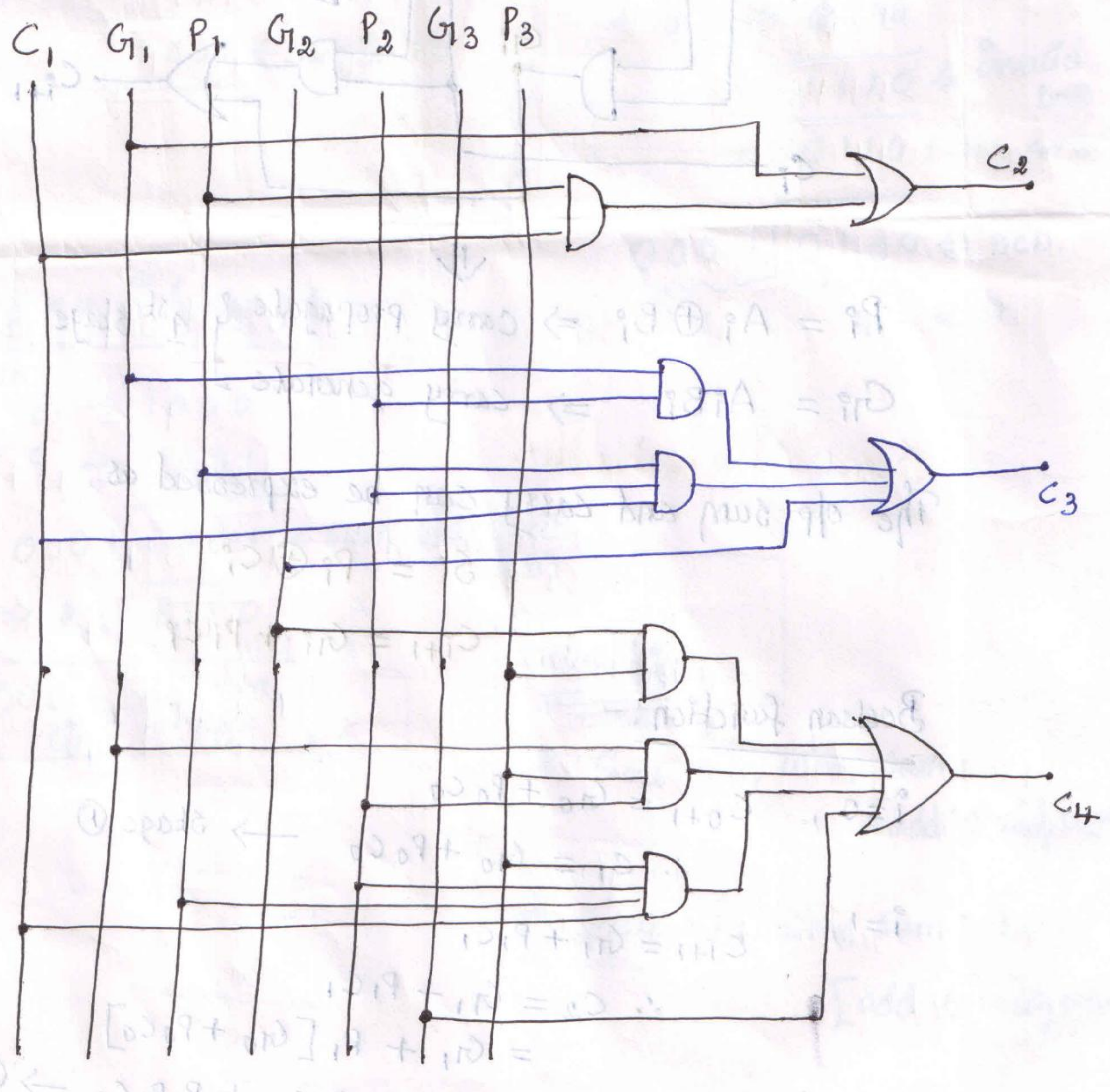
$\therefore C_3 = G_2 + G_1 P_2 + ~~G_1~~ P_1 P_2 C_1 \rightarrow$ ③

$i = 3,$ $C_{3+1} = G_3 + P_3 C_3$

$C_4 = G_3 + P_3 [G_2 + G_1 P_2 + P_1 P_2 C_1]$

$= G_3 + G_2 P_3 + G_1 P_3 P_2 + P_3 P_2 P_1 C_1 \rightarrow$ ④

Logic diagram:-



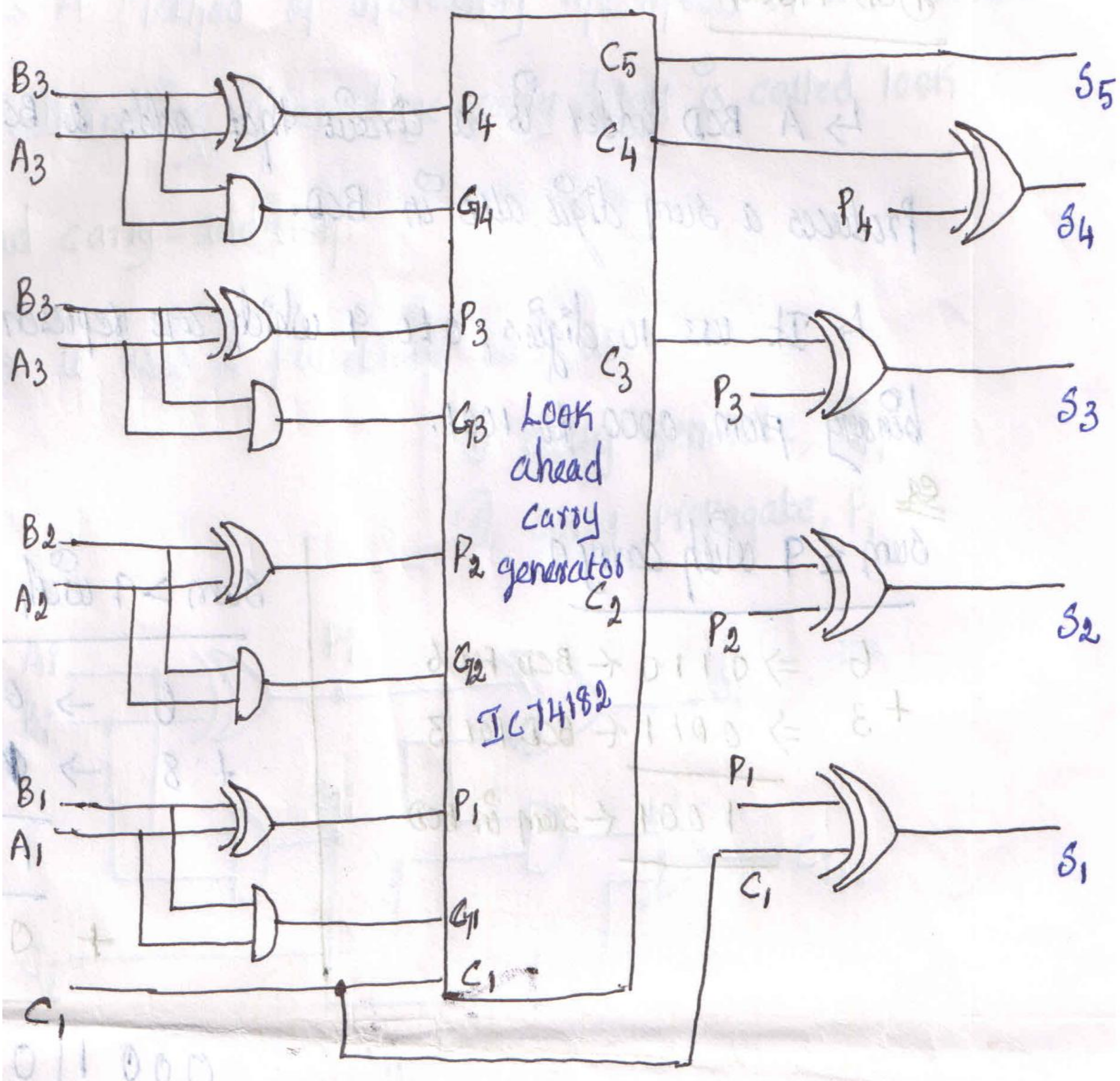


Fig. 4-bit parallel adder with look ahead carry generator