10 August 23

22 August 2023 22:13

Positional Weight method [Converting any type to decimal]

$(37)_8 = (\quad)_{10}$

Weight according to the position

$\Rightarrow 3\times8^{1} + 7\times8^{0}$

$= 3\times8 + 7$

$= 24 + 7$

$= \boxed{31}$

$[1A]_{16} = (\quad)_{10}$

$= 1\times16^{1} + 10\times16^{0}$

$= 16 + 10 = \boxed{26}$

$[100]_2 = [\quad]_{10}$

$1\times2^{2} + 0\times2^{1} + 0\times2^{0}$

$= \boxed{4}$

$(23.5)_{10} = (\quad)_2$

2	23
2	11 — 1
2	5 — 1
2	2 — 1
2	1 — 0
	1

$(10111.1)_2$

$0.5 \times 2 = 1.0$

$0 \times 2 = 0.$

For

$(23.6) = (10111.10011)_2$

$0.6 \times 2 = 1.2$

$0.2 \times 2 = 0.4$

$0.4 \times 2 = 0.8$

$0.8 \times 2 = 1.6$

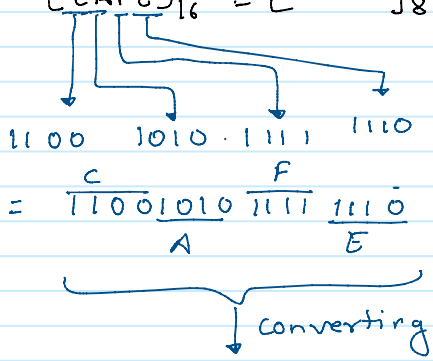
$0.6 \times 2 = 1.2$

For converting from Hexadecimal (16) to Octal (8)

e.g. $[CAFE]_{16} = [\quad]_8$

For converting from Hexadecimal (16) to Octal (8)

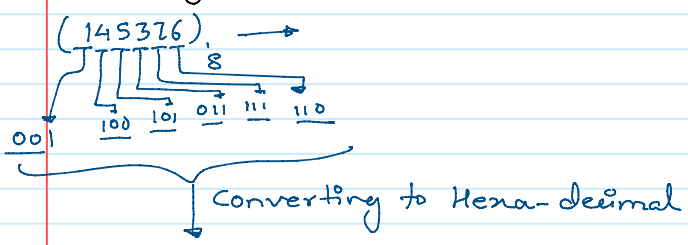
e.g. $[CAFE]_{16} = [\quad]_8$



001 100 101 011 111 110
1 4 5 3 7 6

$= (145376)$

Converting it back to hexadecimal



1100 1010 1111 1110
C A F E

$[CAFE]_{16}$

★ Complement

Q. Find 1's complement for '0110'

Ans:-
$$\begin{array}{r} 1\ 1\ 1\ 1 \\ -\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 0\ 1 \end{array}$$

[complement is applicable only for negetive numbers]

Q. Find 2's complement for '$(0110)_2$'

Ans:- 1's complement of 0110

$$= \begin{array}{r} 1\ 1\ 1\ 1 \\ -\ 0\ 1\ 1\ 0 \\ \hline (1\ 0\ 0\ 1) \\ +\quad 1 \\ \hline 1\ 0\ 1\ 0 \end{array}$$

Q. For a number $(1110)_2$, find 2's complement.

Ans:-

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ -\ 1\ 1\ 1\ 0 \\ \hline 0\ 0\ 0\ 1 \end{array} \longrightarrow 1\text{'s complement}$$

$$\begin{array}{r} 0\ 0\ 0\ 1 \\ +\quad\quad 1 \\ \hline 0\ 0\ 1\ 0 \end{array} \longrightarrow 2\text{'s complement}$$

Q. For a large number like $(1\ 10\ 111\ 00\ 111)_2$ Find 2's complement.

Ans:- First find '1' from right,

1 1 0 1 1 1 0 0 1 1 (1)

So, now just keep '1' from the very right position 1 and take the complement of remaining numbers.

$$\begin{array}{r} \rightarrow\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ -\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\ \hline 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1 \end{array}$$

Q. For $(12345)_{10}$, find 10's complement

Ans:- 9's complement:-

$$\begin{array}{r} 9\ 9\ 9\ 9\ 9 \\ -\ 1\ 2\ 3\ 4\ 5 \\ \hline (8\ 7\ 6\ 5\ 4) \end{array}$$

10's complement

$$= \begin{array}{r} 8\ 7\ 6\ 5\ 4 \\ +\quad\quad 1 \\ \hline 8\ 7\ 6\ 5\ 5 \end{array}$$

If the base $[(\ )_{10}]$ is even, take the previous complement.
If the base is odd take the same number for complement.

Q. For $(12346)_7$ Find 8's complement.

7's complement =

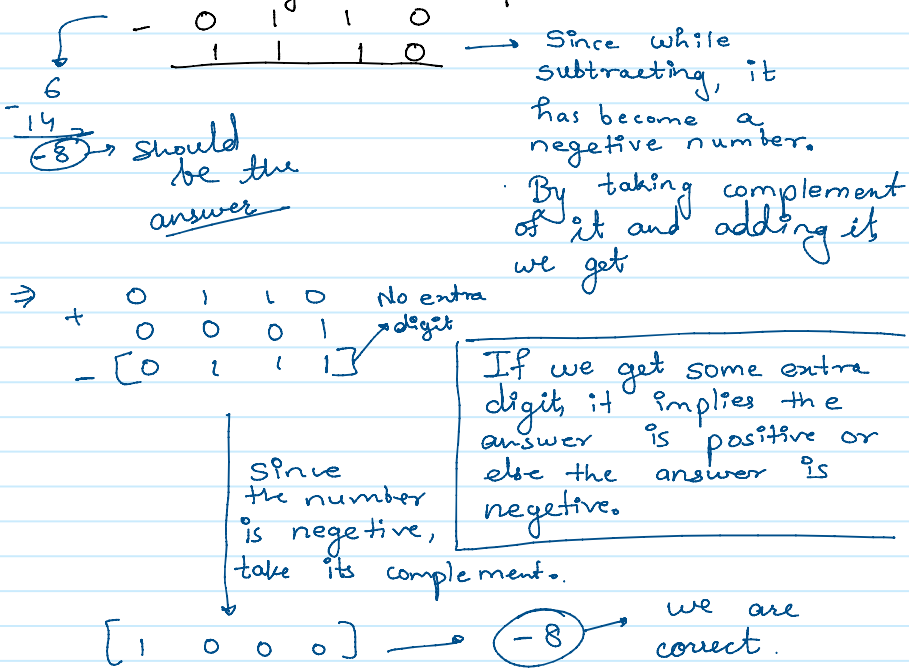
$$\begin{array}{r} 7\ 7\ 7\ 7\ 7 \\ -\ 1\ 2\ 3\ 4\ 6 \\ \hline 6\ 5\ 4\ 3\ 1 \\ +\quad\quad 1 \\ \hline = 6\ 5\ 4\ 3\ 2 \end{array}$$

8's complement

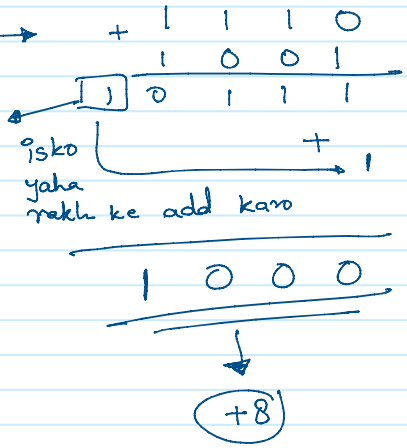
Q. Add

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

Q. Subtract using 1's complement



e.g. 1 1 1 0
- 0 1 1 0



Sign-Bit Representation:-

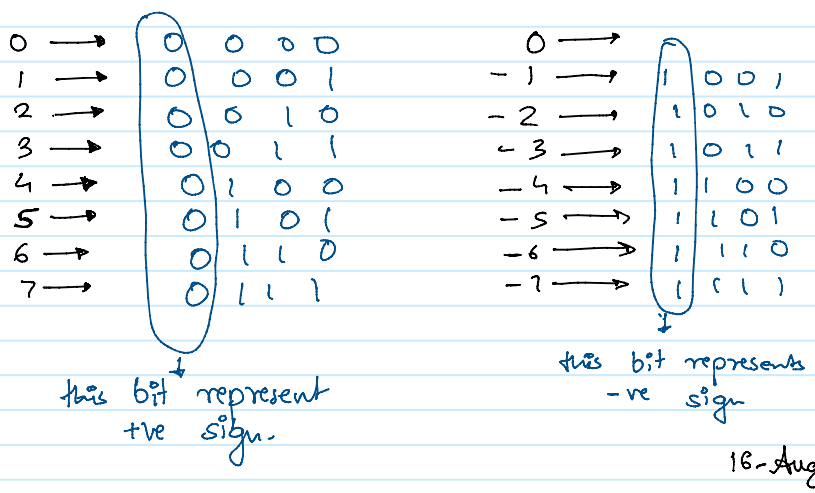
unsigned numbers are positive number, having no space for negetive numbers.

e.g. Express numbers in unsigned but 4-bit format.

0 → 0 0 0 0
1 → 0 0 0 1
⋮
7 → 0 1 1 1

Sign bit is used to represent the sign of a number, '1' bit is reserved in signed bit representation.

e.g.



16-August-23

e.g. Here are some 4-bit representations:-

	Sign-bit representation	1's complement form
0 0 0 0	+0	+0
0 0 0 1	+1	+1
0 0 1 0	+2	+2
0 0 1 1	+3	+3
0 1 0 0	+4	+4
0 1 0 1	+5	+5
0 1 1 0	+6	+6
0 1 1 1	+7	+7
1 0 0 0	-0	= -7 (1's complement of 000 is 111)
1 0 0 1	-1	-6
1 0 1 0	-2	-5
1 0 1 1	-3	-4
1 1 0 0	-4	-3
1 1 0 1	-5	-2
1 1 1 0	-6	-1
1 1 1 1	-7	-0

This bit is reserved for sign

Taking complement of these three bit.

	2's complement
0 0 0 0	+0
0 0 0 1	+1
0 0 1 0	+2
0 0 1 1	+3
0 1 0 0	+4
0 1 0 1	+5
0 1 1 0	+6
0 1 1 1	+7
1 0 0 0	-8
1 0 0 1	-7
1 0 1 0	-6
1 0 1 1	-5
1 1 0 0	-4
1 1 0 1	-3
1 1 1 0	-2
1 1 1 1	-1

For 2's complement of,

000

↳ 111 → [1's complement]

+ 1

1000 → [2's complement]

= 8

For 2's complement of a representation, whose unit's place has '1'.

011 → keep the '1' '1' on the right-most end, take the complement of remaining bit.

⇒ For 011, 2's complement

= 011

complement taken ← | → Complement not-taken.

= 011

complement taken ← 011 → Complement not-taken.

10 1

for a 4-bit representation, the formula for finding the range of sign-bit form is,

$-[2^{n-1}-1]$ to $[2^{n-1}-1]$ → this is also applicable for 1's complement form.

for 2's complement form,

$-[2^{n-1}]$ to $[2^{(n-1)}-1]$

e.g. For a 3-bit binary value, Find the range of 1's complement form, and 2's complement

→ for 1's complement,

$-[2^{(3-1)}-1]$ to $[2^{(3-1)}-1]$

$= -[3]$ to $[3]$

for 2's complement,

$-[2^{(3-1)}]$ to $[2^{(3-1)}-1]$

$-[4]$ to $[3]$.

17-August-23

★ Addition through 1's sign-bit form:- [5-bit available]

e.g. $-3 + +6 = +3$

$-3 \rightarrow 11100$

$+6 \rightarrow 00110$

$10001\,0$

Extra digit ⇒ positive number.

$(1)\,0001\,0$

$0\,0\,0\,1\,1$

$+3$

e.g. $3 - 6$

$3 \rightarrow 00011$

$-6 \rightarrow 11001$

11100

No Extra digit = -ve sign

11100

00011

$= -6$

Since the answer is negative, take complement

e.g. -9

$9 = 1001$

0110

$-9 \rightarrow +10110$

e.g. $-9 \rightarrow$ $+10110$

$-9 \rightarrow$ 10110

$+13$ $\quad$ 101100 $\rightarrow$ $\overset{1}{0}11100$

$+1$

01101

$+13$

positive number

$9 = 1001$

0110

The answer is wrong because the range of 5-bit representation is from -15 to 15.

Now, solving it by making it a 6 bit number.

-9 $\quad$ 110110

-9 $\quad$ 110110

$\overset{1}{0}101100$

$+1$

Extra digit $\quad$ $\overset{1}{0}1101$

$-ve$ $\quad$ taking complement

10010

-18

★ Addition through 2's sign bit form [5-bit available]

e.g. $-3 \rightarrow$ 11101

$+6 \rightarrow$ 00110

$+3$ $\quad$ $1\overset{\downarrow}{0}00111$

+ve sign $\quad$ $+3$

Just ignore it $\quad$ Extra digit $\quad$ answer = positive

e.g. $-6 \rightarrow$ $+11010$

$+3 \rightarrow$ 00011

$\overset{1}{1}1101$

$-ve$ sign $\quad$ since, negetive. Take complement

0011

-3

BCD Code [Binary coded decimal]:-

↳ It is a 4-bit representation only

Decimal	
0	0000
1	0001
2	0010

0 →	0 0 0 0
1 →	0 0 0 1
2 →	0 0 1 0
3 →	0 0 1 1
4 →	0 1 0 0
5 →	0 1 0 1
6 →	0 1 1 0
7 →	0 1 1 1
8 →	1 0 0 0
9 →	1 0 0 1

} Valid only. } From 10-15 it is all invalid state.

e.g. For $(12)_{10}$, What is the BCD code?

$= (0001\ \ 0010)_{BCD}$

★ Self- Complementary Code :-

★ BCD addition.

e.g. $\begin{array}{r} 9 \\ +3 \\ \hline 12 \end{array}$ $\quad +\begin{array}{r} 1001 \\ 0011 \\ \hline 1100 \end{array}$ → Valid BCD state, → Valid BCD state, → Invalid BCD state

$\begin{array}{r} 0001 \\ \downarrow \\ 1 \end{array} \ \begin{array}{r} \textcircled{1} \\ +0110 \\ \hline 0010 \\ \downarrow \\ 2 \end{array}$ } we will add +6 in the invalid state to get the code

$(00010010)_{BCD}$ A valid BCD state

$\downarrow \quad \downarrow$

$1 \quad 2 \quad = 12$

e.g. $\begin{array}{r} 28 \\ +32 \\ \hline 60 \end{array}$ $\quad +\begin{array}{r} 0010 \\ 0011 \\ \hline 0101' \\ \hline 0110 \\ \underbrace{\quad}_{6} \end{array} \quad \begin{array}{r} 1000 \\ 0010 \\ \hline 1'0\,10 \\ +\ 0110 \\ \hline 0000 \\ \underbrace{\quad}_{0} \end{array}$ → Valid, → Valid

Valid, valid

$= 60.$

e.g. $\begin{array}{r} 1 \\ 239 \\ +148 \\ \hline 387 \end{array}$ $\quad +\begin{array}{r} 0010 \\ 0001 \\ \hline 0011 \\ \underbrace{\quad}_{3} \end{array} \quad \begin{array}{r} 1\ 1\ 1\ \textcircled{1} \\ 0011 \\ 0100 \\ \hline 1000 \\ \underbrace{\quad}_{8} \end{array} \quad \begin{array}{r} 1001 \\ 1000 \\ \hline 0001 \\ +0110 \\ \hline 0111 \\ \underbrace{\quad} \end{array}$ } → Valid state

Is group se 1 carry forward hua hai, to ham +6 (0110)

3 + 8 = 0 1 1 1 + 0 1 1 0 = 0 1 1 1 (7)

↓ carry forward hua hai, to ham +6 (0110) add karenge.

★ BCD Subtraction:-

e.g. 23 − 12 = 11

0 0 1 0 + 0 0 0 1 = 0 0 1 1

0 0 1 1 + 0 0 1 0 = 0 0 0 1

1 1

for 12 − 3 = 9

Similarly, for binary number system,

[1 − 1 = 0] ← 1 0 → [2 + 0 = 2]

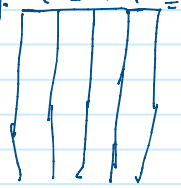
− 1

0 1

★ Self-Complementary code:-

The code, whose sum of values of weight is 9 is called self-complementary code.

e.g. 4 2 2 1 = 4+2+2+1 = 9.



	2	4	2	1		
0→	0	0	0	0		
1	0	0	0	1		
2	0	0	1	0	/	1 0 0 0
3	0	0	1	1	/	1 0 0 1
4	0	1	0	0	/	1 0 1 0
5	0	1	0	1	/	1 0 1 1
6	0	1	1	0	/	1 1 0 0
7	0	1	1	1	/	1 1 0 1
8	1	1	1	0		
9	1	1	1	1		

Lower Precedence Logic is followed.

★ Lower Precedence Logic →

for '2', there are two possible Self-complementary code in this '2421' format

0 0 1 0 / 1 0 0 0

↓

error (1) → MSB is used for lower number

now, '2' complement is 7, whose code is -

0 1 1 1 / 1 1 0 1

↓

Now, 2 complement is 7, whose code is -

0 1 1 1 / 1 1 0 1

↓ complement of 1000

LSB is used representing higher value.

This is not correct. So, for '2' 0010 is more genuine.

6 4 1 -2 Homework.

Is it also a self-complementary Code?

Excess-3 Code:-

{range → 3 - 12}

BCD + 3 = Excess-3 code {This code does not have weight}

{This code is self-complementary code}

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

e.g. $(123)_{10} = (\quad)_{Ex-3}$

$= (0100\ 0101\ 0110)_{Ex_3}$

(1 2 3)

Gray Code:-

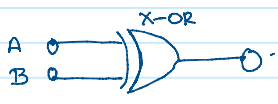
{Mirror Image code} {Cyclic Code}

e.g.

0 0
0 1
1 1
1 0

→

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



For XOR Gate, jaha bhi odd number of 1 hoga waha output 1 hoga.

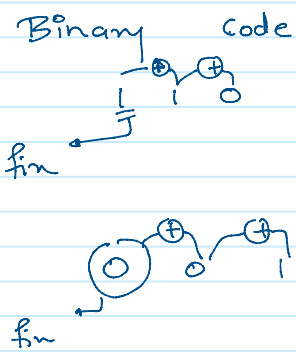
0 0	0
0 1	1
1 0	1
1 1	0

hoga waha output 1 hoga.

1 1 | 0

For Gray code, fix the left-most digit and in the right digit just write the output of XOR Gate.

e.g.



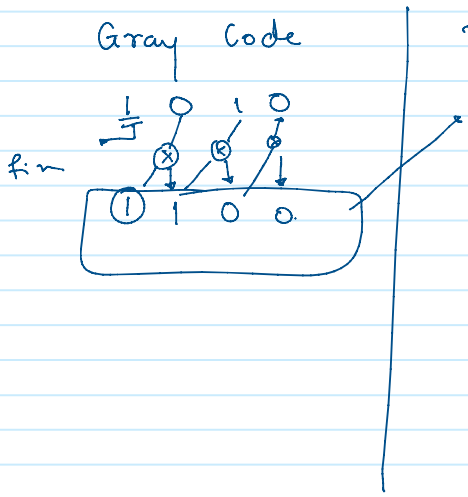
Gray Code.

1 0 1

0 0 1

Conversion of Gray code to Binary,

for this conversion go angle-wise.



Binary Code

0 1 0 0.

Logic Gates:-

AND				OR			
0	0	→	0	0	0	→	0
0	1	→	0	0	1	→	1
1	0	→	0	1	0	→	1
1	1	→	1	1	1	→	1

Digital Language main OR → Addition.
AND → Multiplication.

* Commutative Law:-
  $A+B = B+A$
  $A.B = B.A$

* Associative Law:-
  $A+(B+C) = (A+B)+C$
  $A.(B.C) = (A.B).C$

* Distributive Law:-
  $A.(B+C) = A.B + A.C$

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

* $A+A = A$
  $A+\bar{A} = 1$
  $A+1 = 1$
  $A+0 = A$

* De Morgan's Law:-

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

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

* $\underbrace{1+x+y+xy+xz}_{} = 1$

  as x, y, z may be 0 or 1 in binary.

Transposition Theorem:-

* $[A+B][A+C]$
  $= A.A + A.C + A.B + B.C$
  $= A(1+C+B) + BC$
  $= A + BC$

* $(\underbrace{x+y}_{A} + wz)$
  $= (x+y+w)(x+y+z)$

* $f = A\bar{B} + AB\bar{C} + A\bar{B}\bar{C}D$
  $= A(\bar{B} + B\bar{C}) + A\bar{B}\bar{C}D$
  $= A[(\bar{B}+B)(\bar{B}+\bar{C})] + A\bar{B}\bar{C}D$
  $= A(\bar{B}+\bar{C}) + A\bar{B}\bar{C}D$
  $= A\bar{B} + A\bar{C} + A\bar{B}\bar{C}D$
  $= A\bar{B}(1+\bar{C}D) + A\bar{C}$
  $= A\bar{B} + A\bar{C}$
  $= A(\bar{B}+\bar{C})$
  $= A\bar{B}\bar{C}$

* $f = (x+y+z)(x+y+\bar{z}) + (x+\bar{y}z)$
  $= (x+y+z\bar{z}) + (x+\bar{y})(x+z)$
  $= \underbrace{(x+y)}_{a} + \underbrace{(x+\bar{y})(x+z)}_{b}$ $(a+bc)$ format.
  $= (x+y+x+\bar{y})(x+y+x+z)$
  $= (1)(x+y+z)$
  $= (x+y+z)$

☆ Concensus Theorem :-

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

$$= [AB + \bar{A}C] \checkmark$$

★ $f = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

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

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

XOR Gate: $B \oplus C = (\bar{B}C + B\bar{C})$

XNOR Gate: $B \odot C = (\bar{B}\bar{C} + BC)$

$$= \bar{A}(y) + A(\bar{y})$$

$$= A \cdot \oplus y$$

$$f = A \oplus B \oplus C$$

A	B	f
0	0	0
0	1	1
1	0	1
1	1	1

SOP $= AB + A\bar{B} + \bar{A}B$ [Preferred NAND Gate]

True Logic

POS $= [A+B]$ [Preferred NOR Gate]

False Logic

Q. $f = A + C\left[\overline{AB + B\bar{C}}\right]$

Soln :-

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

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

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

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

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

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

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

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

SOP [True Logic]

	a	b	f
i	0	0	0
ii	0	1	1 → True
iii	1	0	0
iv	1	1	1 → True

for ii,

0 → $\bar{a}$, 1 → b } for SOP,

$\bar{a}b$ & $a\bar{b}$ = 1 → inko literals in product form bolte hai!

from iii | from iv

POS:- [False Logic]

for (i) and (iii);

$(a+b)$ → inko literals in sum form bolte hai

$f_{pos} = (a+b)(\bar{a}+b)$

a=0, b=0 → from (i)

a=1, b=0 → from (iii)

$= a\bar{a} + ab + \bar{a}b + b$

$= b(1+a+\bar{a})$

$= b$

a	b	c	
0	0	0	→ 1
0	0	1	→ 1
0	1	0	→ 1
0	1	1	→ 1
1	0	0	→ 1
1	0	1	→ 1
1	1	0	→ 1
1	1	1	

$f_{sop} = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$

$= \bar{a} + \bar{b} + \bar{c}$

$f_{pos} = \bar{a} + \bar{b} + \bar{c}$

$$f_{sop} = f_{pos}$$

a	b	f_{sop} → minterms
0	0	→ 1
0	1	→ 1
1	0	→ 1
1	1	→ 1

a	b	F_{SOP} → min terms
0	0 →	1
0	1 →	1
1	0 →	1
1	1 →	1

Sum of all the literals in SOP is always 1.

Product of all the literals in POS is always 0.

$F_{pos} = (a+b)\,(b+c)$ — (a+b): OR Gate; (b+c): AND Gate

$F_{SOP} = ab + bc$ — ab: AND Gate; +: OR Gate

AND OR = NAND NAND

OR AND = NOR NOR

a b c	min terms (SOP)	max terms (POS)
0 1 0 →	$\bar{a}b\bar{c}$	$(a+\bar{b}+c)$
1 1 0 →	$ab\bar{c}$	$(\bar{a}+\bar{b}+c)$

$F_{SOP} = [\bar{a}b\bar{c} + ab\bar{c}]$ → SOP

Literals

$F_{pos} = [(a+\bar{b}+c).(\bar{a}+\bar{b}+c)]$

e.g. $F = ab + bc$ → [A SOP expression]

* There are two universal gates
  a) NAND, } Every gate can be designed
  b) NOR } using these

* For SOP Logic, AND-OR Logic gate is preffered
  AND - OR = NAND - NAND

* For POS Logic, OR-AND Logic gate is preffered
  OR-AND = NOR - NOR.

Bubbled OR = NAND
Bubbled AND = NOR

Karnaugh Map [K-Map]

It Gives Gray Code

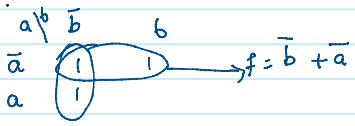
a	b
0	0
0	1
1	0
1	1

a \ b	0	1
0		
1		

a	b	c
0	0	0

a \ bc	00	01	11	10
0	0	1	3	2
1	4	5	7	6

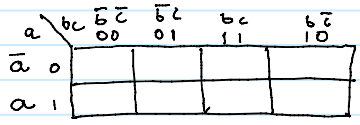
Rules :-



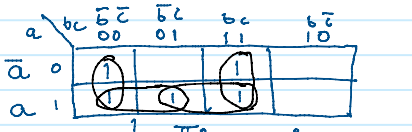
Prefrence :- $2 < 4 < 6 < 8$.

★ $f = \pi(0, 5, 4, 7, 3)$ → π → POS

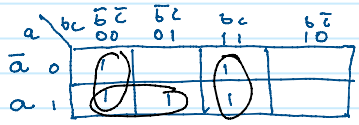
SOP — These values are positions.



a	b	c	
0	0	0	→ 1
0	0	1	
0	1	0	
0	1	1	→ 1
1	0	0	→ 1
1	0	1	→ 1
1	1	0	
1	1	1	→ 1



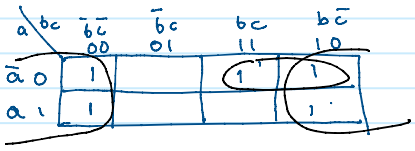
↳ This one is wrong because of 1 pair



$f = a\bar{b} + bc + \bar{b}\bar{c}$

This represents SOP, if Π is present, it represent POS.

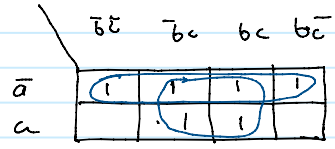
Q. $f = \Sigma(0, 2, 3, 4, 6)$



$f = \bar{c} + \bar{a}b$

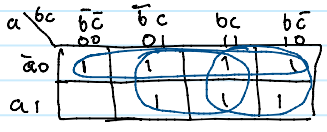
a	b	c	
0	0	0	→ 1
0	0	1	
0	1	0	→ 1
0	1	1	→ 1
1	0	0	→ 1
1	0	1	
1	1	0	→ 1
1	1	1	

Q.



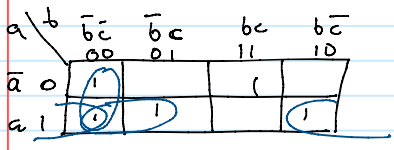
$f = \bar{a} + c$

Q.



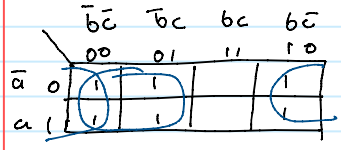
Soln:- $f = \bar{a} + b + c$

Q.



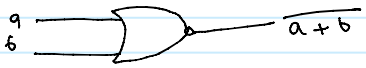
Soln:- $f = a\bar{b} + \bar{b}\bar{c} + a\bar{c} + \bar{a}bc$

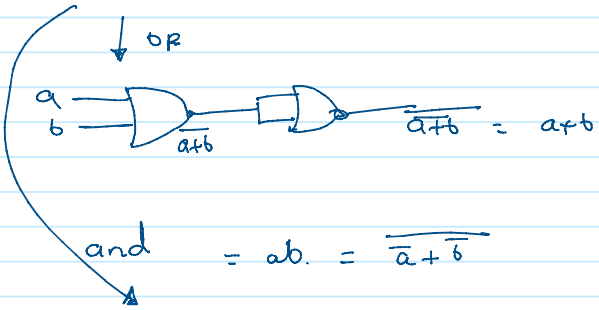
Q.



Soln:- $f = \bar{b} + \bar{c}$

NOR

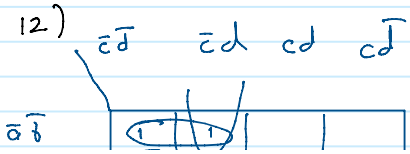


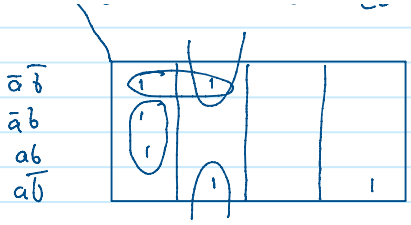


★

ab \ cd	c̄d̄ 00	c̄d 01	cd 11	cd̄ 10
āb̄ 00	0	1	3	2
āb 01	4	5	7	6
ab 11	12	13	15	14
ab̄ 10	8	9	11	10

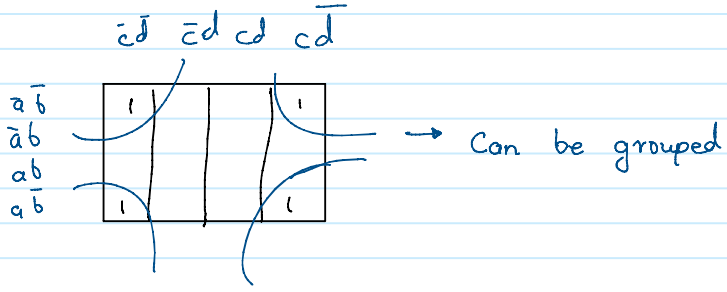
Q. $f = \Sigma(0, 1, 4, 9, 10, 12)$





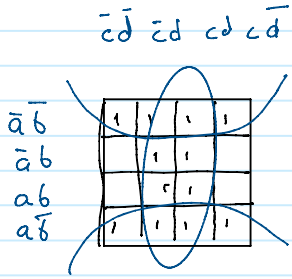
$f = \bar{a}\bar{b}\bar{c} + b\bar{c}\bar{d} + \bar{b}\bar{c}d + a\bar{b}c\bar{d}$

★



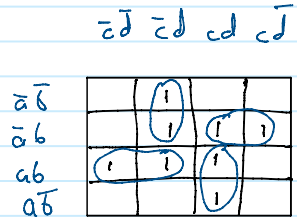
$f = \bar{b}\bar{d}$

Q.



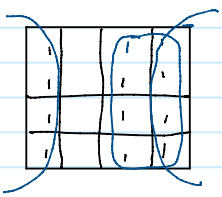
$f = \bar{b} + d$

Q.

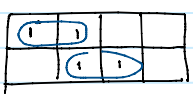


$f = \bar{a}\bar{c}d + \bar{a}bc + acd + ab\bar{c}$.

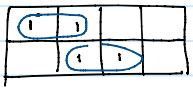
Q.



Q.



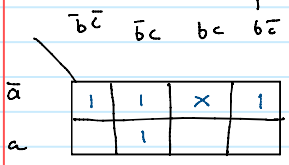
Q.



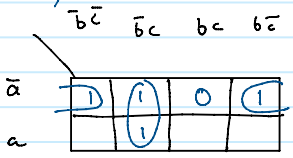
Don't Care :-

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	X → i don't care
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Q.

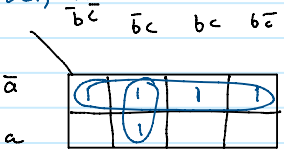


Let, X = 0

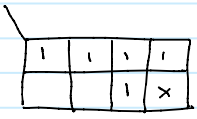


$f = \bar{a}\bar{c} + \bar{b}c$

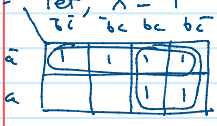
Let, X = 1



$f = \bar{a} + \bar{b}c$

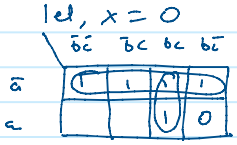


Solr:- let, X = 1



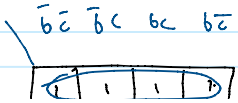
$f = \bar{a} + b$

let, X = 0

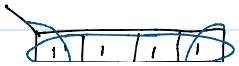


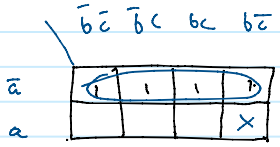
$f = \bar{a} + bc$

Q.

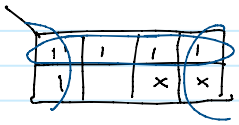


Q.



Q. 

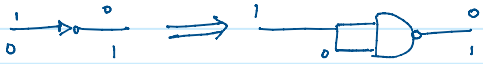
$f = \bar{a}$

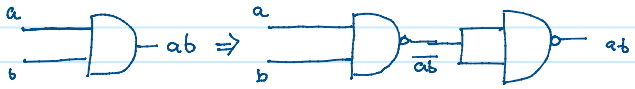
Q. 

$f = \bar{a} + \bar{c}$

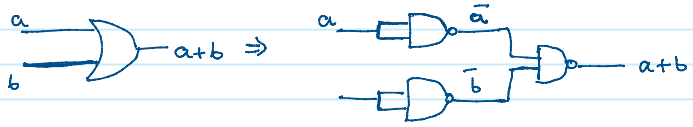
Combinational Circuit :-

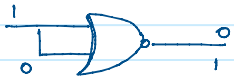
→ Half-Adder :-

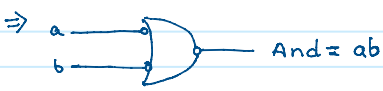




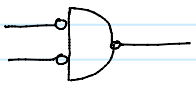
NOR



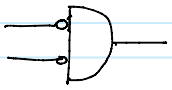


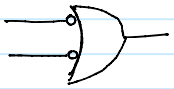


Q.


OR Gate *


AND Gate *


NOR Gate


NAND Gate

XOR Gate [7486] 

OR Gate [7432]

NOR Gate [ ]

XOR Gate

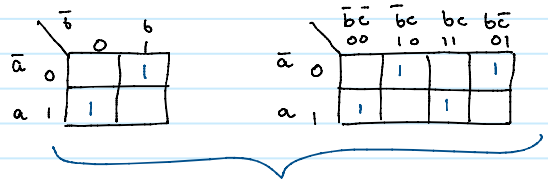
a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

$f = \underbrace{\bar{a}\cdot b + a\bar{b}}$ → SOP

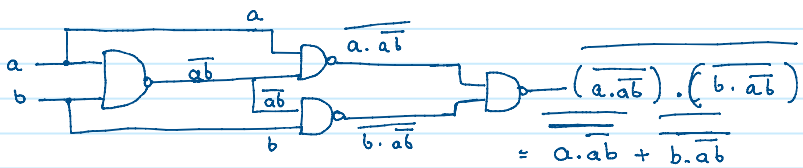
$= a \oplus b$

0	1	1
1	0	1
1	1	0

$= a \oplus b$



XOR Gate $\quad f = a \oplus b \oplus c.$



$= a.\bar{ab} + b.\bar{ab}$

$= (a.\bar{ab}) + (b.\bar{ab})$

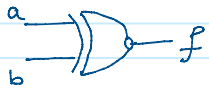
$= a.(\bar{a}+\bar{b}) + b.(\bar{a}+\bar{b})$

$= a\bar{b} + \bar{a}b$

$= a \oplus b.$

XNOR Gate

Co-incidence Gate



a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

$f = \bar{a}\bar{b} + ab$

$= a \odot b$

$\overline{f_{XOR}} = f_{XNOR}$

$\boxed{\overline{a \oplus b} = a \odot b}$

$\overline{\bar{a}\bar{b} + ab} = a \oplus b$

$\overline{\bar{a}b + a\bar{b}} = a \odot b$

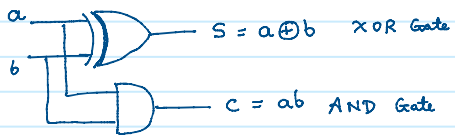
Half-Adder :-

Only applicable for 2-bit operations.

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$Sum = \bar{a}b + a\bar{b} = a \oplus b$

$Carry = ab$



Full-Adder

Applicable for 3 bit.

	a	b	c	sum	carry
	0	0	0	0	0
① →	0	0	1	1	0
② →	0	1	0	1	0
	0	1	1	0	1
④ →	1	0	0	1	0
	1	0	1	0	1
	1	1	0	0	1
⑦ →	1	1	1	1	1

Sum = Σ (1,2,4,7)
Full Adder

Sum = $\bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$

	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	1			1
a		1	1	

$f = a \oplus b \oplus c$

Carry = Σ (3,5,6,7)

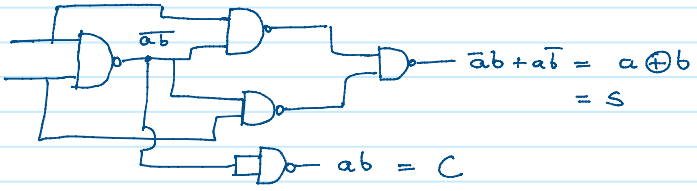
	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$			(1)	
a		(1)	(1)	(1)

Carry = $ab + bc + ca$.

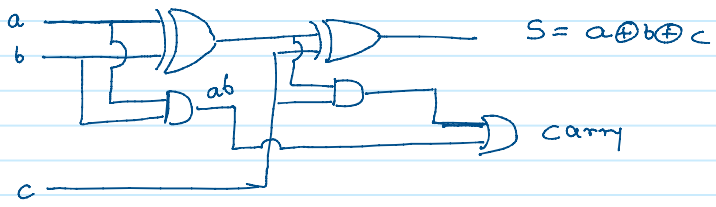
	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$			(1)	
a		(1)	(1)	(1)

★ Carry = $\bar{a}bc + a\bar{b}c + ab$
$= c(\bar{a}b + a\bar{b}) + ab$
$= c(a \oplus b) + ab$.

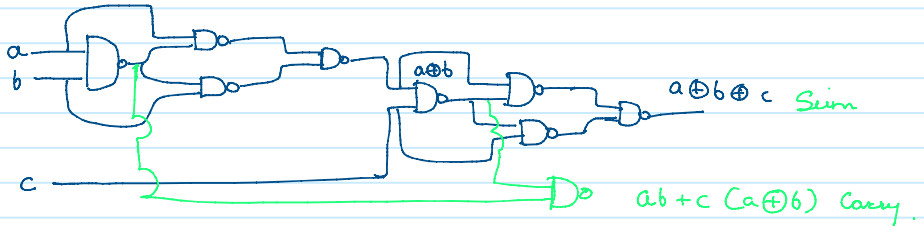
Q. For developing a Half-Adder, how many NAND Gates are required.



Full-Adder Circuit diagram :-



Design a Full Adder using NAND Gate only.

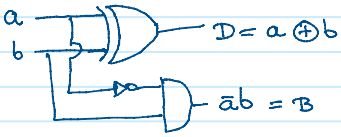


Half- Subtractor

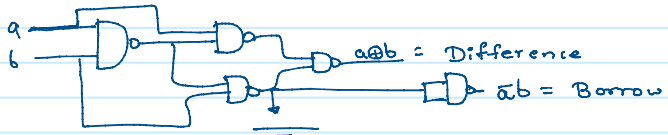
a	b	D [Difference]	B [Borrow]
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

$D = \bar{a}b + a\bar{b} = a \oplus b$

$B = \bar{a}b$



Design a half subtractor using NAND Gate only.



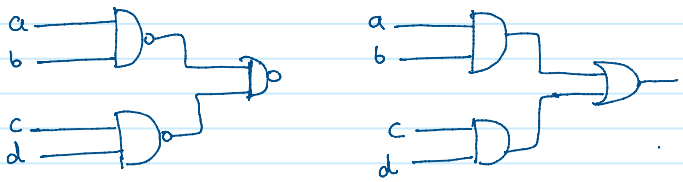
$\overline{\bar{a}b \cdot b}$

$= \overline{(\bar{a}+\bar{b}) \cdot b}$

$= \overline{\bar{a}b + 0}$

$= \overline{\bar{a}b}$

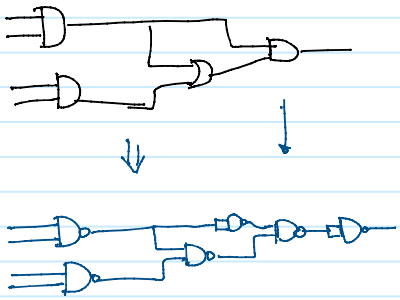
$f = ab + cd \rightarrow$ [AND-OR]



NAND-NAND AND OR

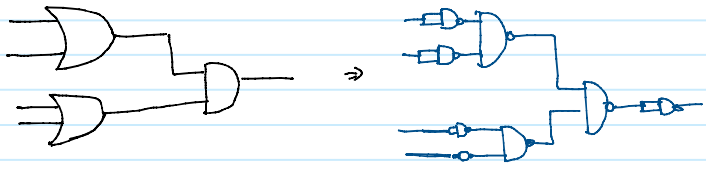
* $F = (a+b)(c+d)$ → OR AND Logic or
NOR NOR logic

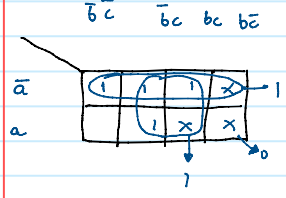
Convert this to NAND Gate.



* $A \oplus A = 0$. $A \odot A = 1$
$A \oplus \bar{A} = 1$ $A \odot \bar{A} = 0$
$A \oplus 0 = A$ $A \odot 0 = \bar{A}$
$A \oplus 1 = \bar{A}$ $A \odot 1 = A$

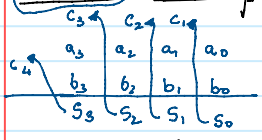
Convert these to NAND Gate.





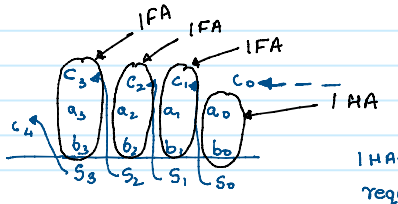
Thoda galat hai ise parallel bulana.

Parallel Binary Adder:-



0 0 1 1
1 0 1 1

[Disadvantage
↳ Time delay]



1 HA + 3 FA required

1 FA = 2 HA + 1 OR Gate

1 HA + 2 HA + 2 HA + 2 HA + 3 OR
= 7 Half adder + 3 OR Gate

→ For 2-n bit, how many Full Adder required?

→ n-full adder

For addition

= (n-1) FA + 1 HA.

= (n-1) (2HA + 1OR) + 1HA

= (n-1)(2HA) + (n-1)(OR) + 1HA

For n=10,

= 18 HA + 9OR + 1HA

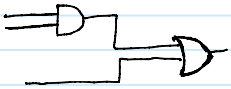
= 19HA + 9OR

Surprise Test :-

Q.1 $f = \sum (0, 1, 2, 3, 4, 5) + d(6)$

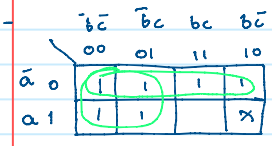
Q2. $f = \Pi (0, 1, 2)$

Q3. $f = ABC + \bar{A}BC + BC + AC$ [form the standard SOP]

Q4. $f =$  [Convert this gate to NAND Gate only]

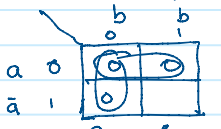
Soln:-

Q1.



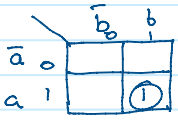
$\Rightarrow f = \bar{a} + \bar{b}$ [Don't care ki yaha koi jarurat nahi]

Q2.



Pos form

$f = ab$



SOP form

$f = ab$

Q3. $f = ABC + \bar{A}BC + BC + AC$

$= BC(A + \bar{A}) + BC + AC$

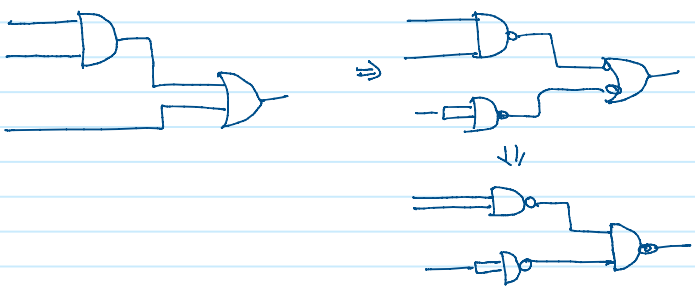
$= \underbrace{BC + BC}_{BC} + AC$

$= BC + AC$

$= C[A + B]$

9861480775 / Prof. P. Bisen Sir ka number.

Q4.



15-09-2023

15-09-2023

Q.
$c_3\ c_2\ c_1$
$a_3\ a_2\ a_1\ a_0$
c_4 $b_3\ b_2\ b_1\ b_0$
$S_3\ S_2\ S_1\ S_0$

For producing sum, it takes 6 ns
For producing carry, it takes 3ns.

For c_1 → we need 3ns
+ For c_2 → we need 3ns
+ For c_3 → we need 3ns
9 ns
$= 3\times(3\text{ns})$

For adding a_3 and b_3 we will need 6ns

Total time required = 9+6 = 15 ns.

In general, if 'n' is number of bit,
we can say
$(n-1)\times$ [time taken by a carry] + [Time taken by sum]

Q. For
c_3 a_3 b_3 + c_2 a_2 b_2 + c_1 a_1 b_1 + 0 a_0 b_0
c_4 S_3 S_2 S_1 S_0

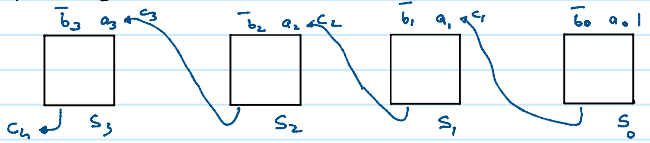
a) How many full adder is required?
→ 4

b) How many HA and Full adder, required
→ 3 Full Adder and 1 Half adder

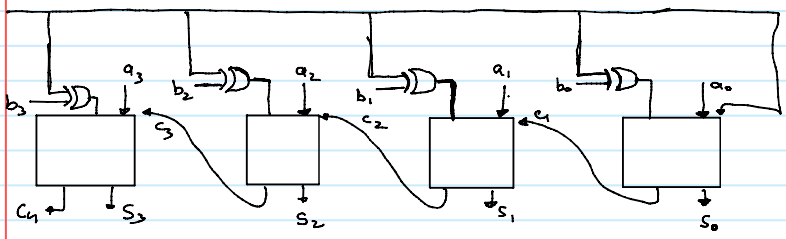
Q. For 13 bit group addition,
a) How many full adder is required - 13 FA
b) How many Full adder and Half adder is required
↳ 12 FA and 1 HA

c) How many Half adder are required
24 HA + 10R Gate

For Subtraction :-



Input



If input is 0, Add
If input is 1, Subtract

$f = x\bar{y} + xw + yw$
$= x\bar{y} + yw$ [consensus]

Carry Look Ahead Addition.

For a 3-bit addition

a	b	c_i	s	c_{i+1}	
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	→

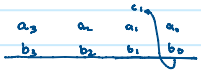
$$c_{i+1} = ab + (a \oplus b).c_i$$

Carry Generator [G] — Carry Propagator [P]

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

$$\Rightarrow C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$
$$= G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

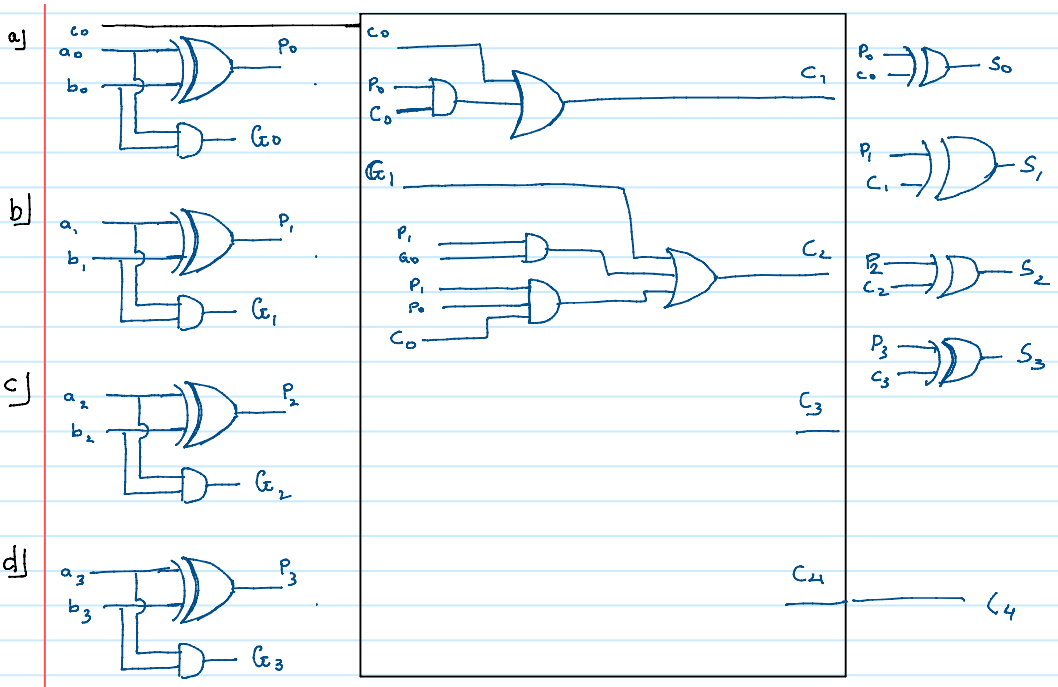


$G_i = a_i b_i$
$P_i = a_i \oplus b_i$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

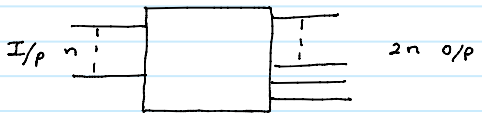
$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$$

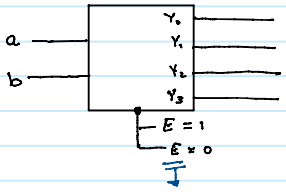
22-09-2023



Time will remain same as all the inputs are given at once

Decoder :-

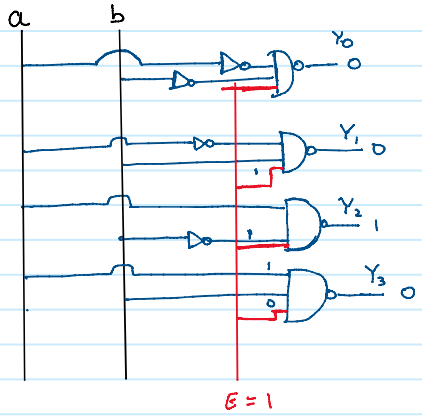




Enabler Pin...

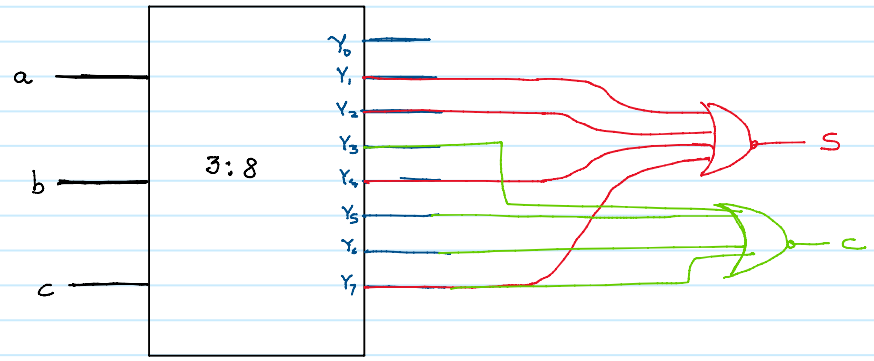
a	b	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

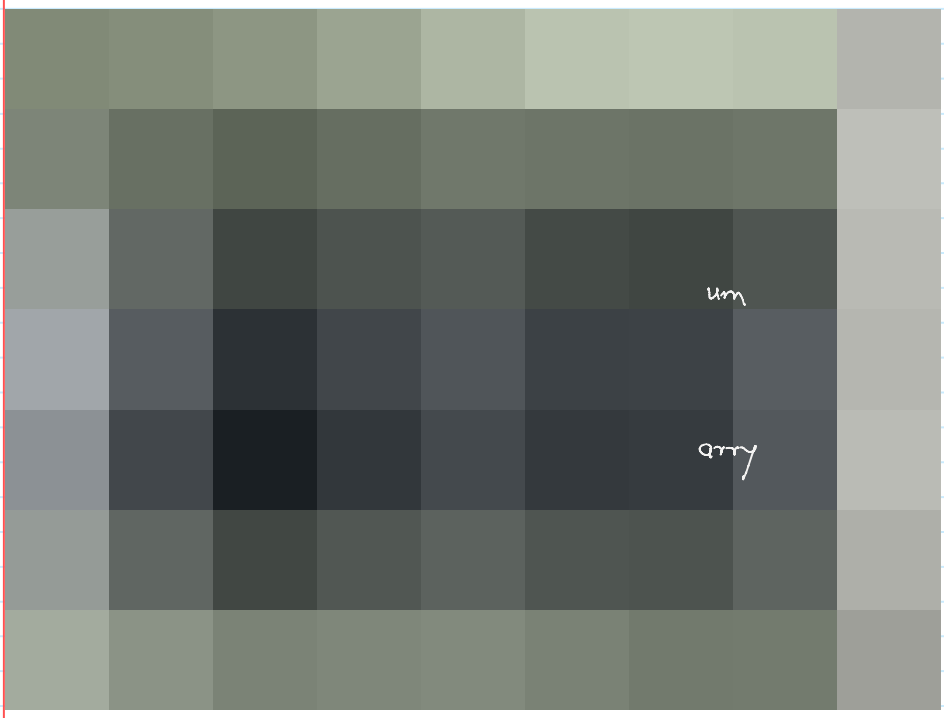
} Active High Logic



Full Adder :-

Full Adder



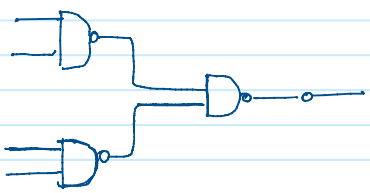


$S = \Sigma\ (1,2,4,7)$ ⟶ SOP of sum
$S = \Sigma\ (3,5,6,7)$ ⟶ SOP of carry,

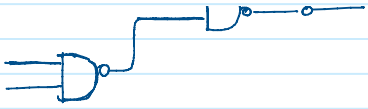
Designing this circuit using AND Gate:-

NAND ke aage NOT Gate laga do!

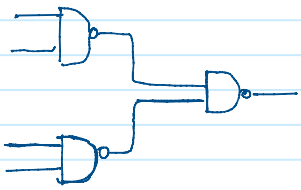
If,



this gives you SOP,



then,



this will give SOP'

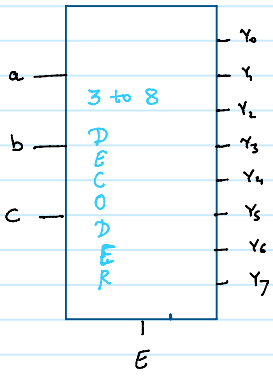
NAND AND - POS

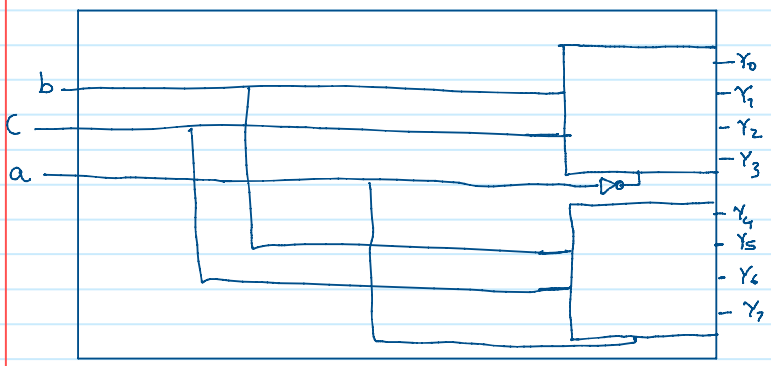
NAND NAND - SOP

AND NOR - POS

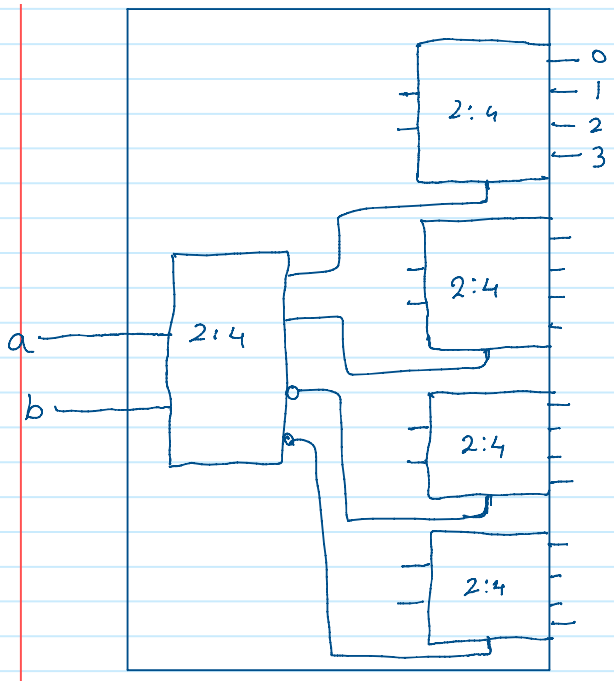
AND OR - SOP.

Q Design 3:8 decoder using 2:4 decoder:-

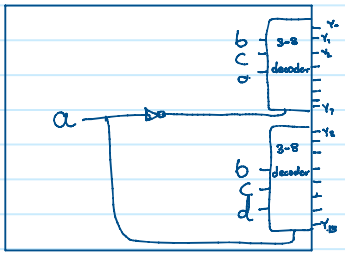




Q. Design 4:16 decoder using 2:4 decoder



Q. Design a 4:16 decoder using 3:8 decoder



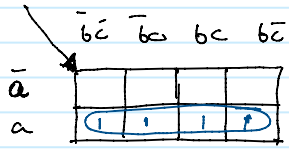
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Binary to Gray Code :-

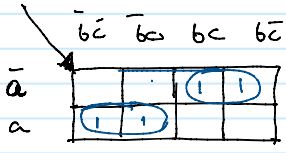
→ 3-bit binary to gray code logic

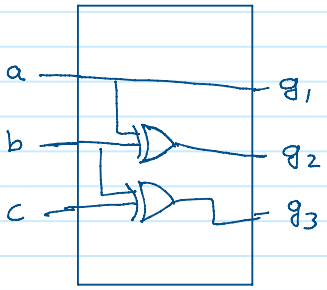
a	b	c	g_1	g_2	g_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

4
5
6
7



$g_1 = a$

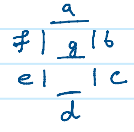




$g_2 = a\bar{b} + \bar{a}b$
$= a \oplus b$

$g_3 = b \oplus c$

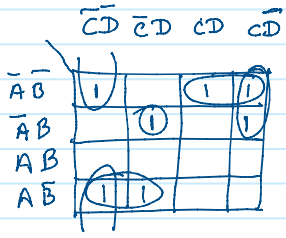
BCD to 7 segment display :-

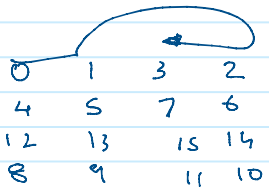


B C D

A B C D	a	b	c	d	e	f	g	
0 0 0 0	1	1	1	1	1	1	0	→ 0
0 0 0 1	0	1	1	0	0	0	0	
0 0 1 0	1	1	0	1	1	0	1	→ 2
0 0 1 1	1	1	1	1	0	0	1	→ 3
0 1 0 0	0	1	1	0	0	1	1	
0 1 0 1	1	0	1	0	1	1	1	
0 1 1 0	1	0	1	1	1	1	1	→ 6
0 1 1 1	1	1	1	0	0	0	0	
1 0 0 0	1	1	1	1	1	1	1	→ 7
1 0 0 1	1	1	1	1	0	1	1	→ 8

For d,



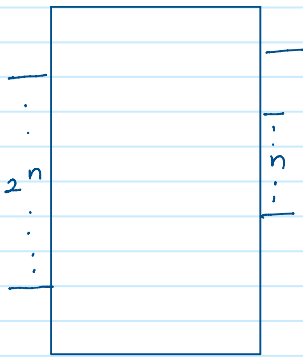


$f = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}C\bar{D} + A\bar{B}\bar{C} + \bar{A}B\bar{C}D$

Encoder :-

↳ Saral bhasha mai ye bhi decoder ki tarah ek convertor hai.

→ 2^n input → kisi bhi form me input ho sakta hai [Octal, Hexadecimal, etc.]
n output → output binary hi hoga.



→ 4:2 Encoder :-

Valid pin
If 0, input is nothing.

A_0	A_1	A_2	A_3	Y_0	Y_1	V
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

If 1, check output

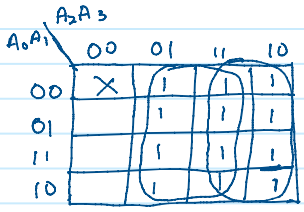
Priority Encoder :-

Highest priority will be considered for output,

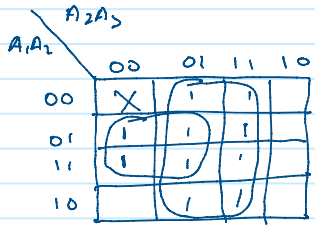
Lower priority bit will not be considered

Y_0,

Kmap

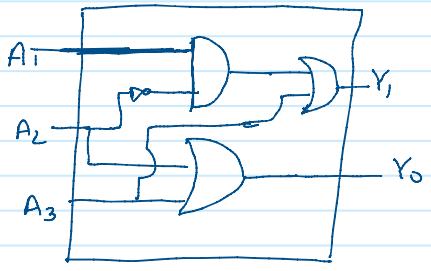


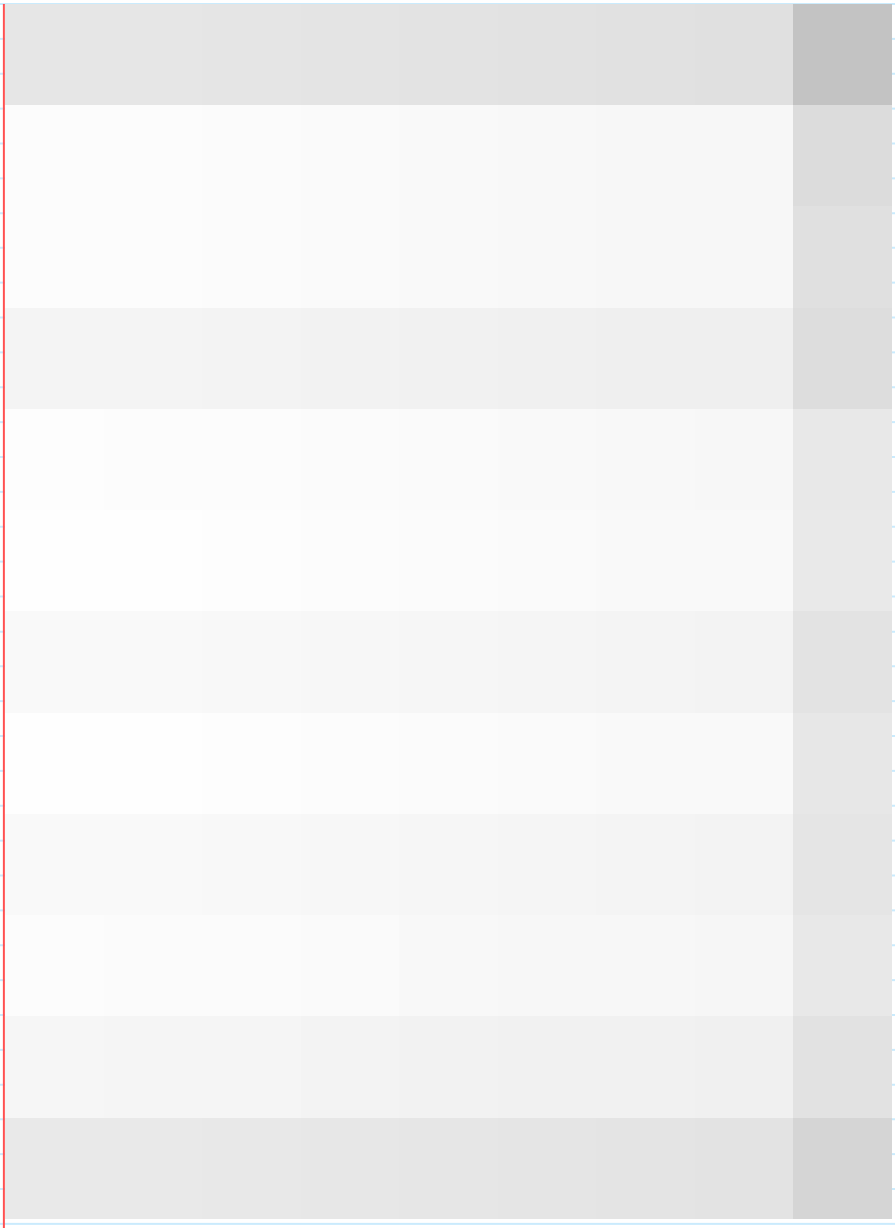
$Y_0 = A_2 + A_3$

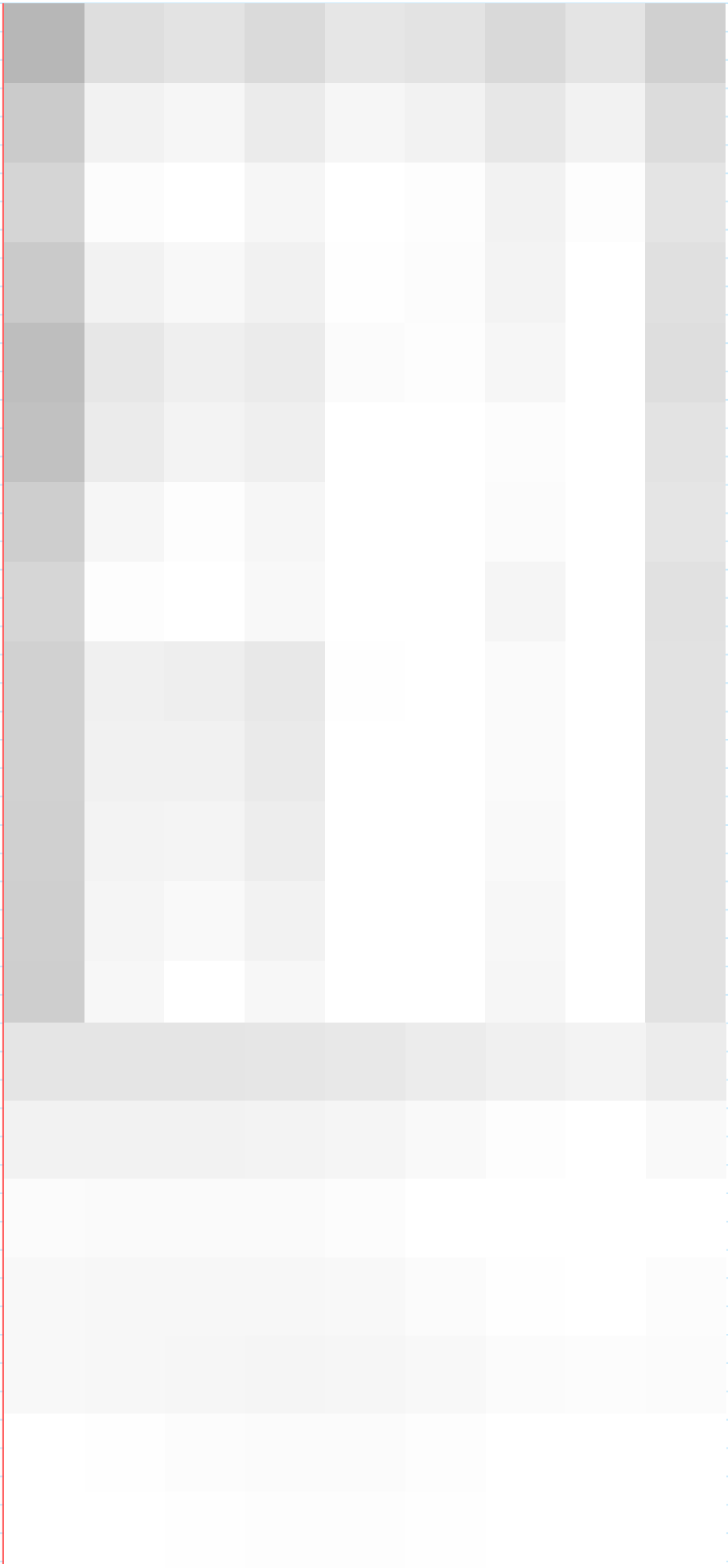


$Y_1 = A_3 + A_1\bar{A}_2$

A_0







5-October-23

Q1.a $-14 - 6$ → $-14 + 6$ ⟷ $+6 - 14$ → $0110 + 0010 = 1000$

$14 \rightarrow 1110$

↕ 2's complement

0010

→ number is negetive.

→ -8 → Answer.

Q1.b $PQ' + PR + QR' = P + QR'$

$= PQ'(R+R') + PR(Q+Q') + QR'(P+P')$

$= PQ'R + PQ'R' + PRQ + PRQ' + PQR' + P'QR'$

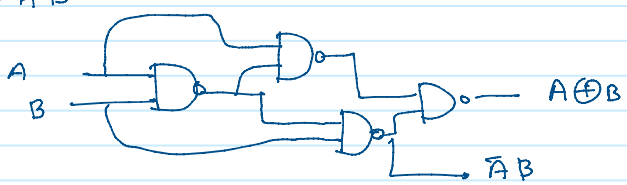
	$\bar{Q}\bar{R}$	$\bar{Q}R$	QR	$Q\bar{R}$
$\bar{P}$				1
P	1	1	1	1

$F = P + Q\bar{R}$

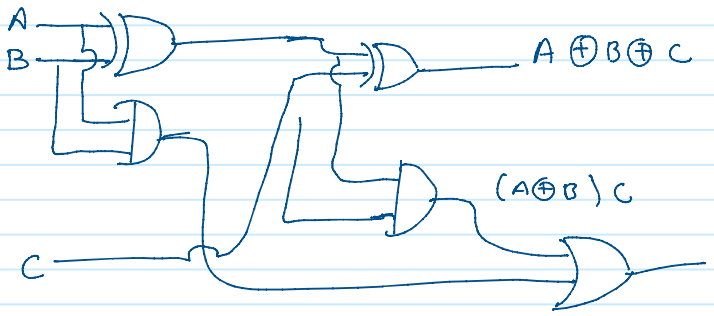
Q1.C Half subtractor using 2 NAND Gate

$D = A \oplus B$

$B = \bar{A}B$



Q1.d



Q1.e. $858 - 749$

$$\begin{array}{r} 1000\ \ 0101\ \ 1000 \\ -\ 0111\ \ 0100\ \ 1001 \\ \hline 0001\ \ 0000\ \ 1111 \\ -\ 0110 \\ \hline 0001\ \ 0000\ \ 1001 \end{array}$$

B

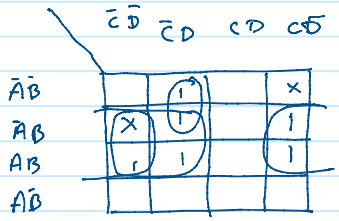
$$1\ 0\ 9$$

$$\begin{array}{r} +\ 0001\ \ 1001 \\ 0001\ \ 0100 \\ \hline 0010\ \ 1101 \\ +\ 0110 \\ \hline 0011\ \ 0011 \end{array}$$

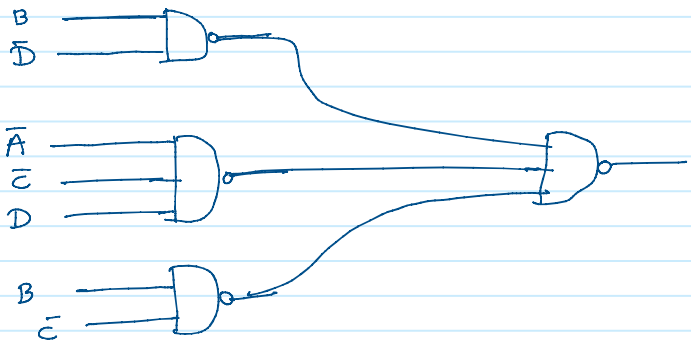
$$3\quad 3$$

$$\begin{array}{r} +19 \\ 14 \\ \hline 33 \end{array}$$

Q2. $\Sigma(0, 1, 2, 9, 11, 15),\ d(3,4)$



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

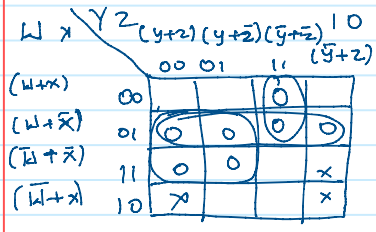


$\Sigma(0, 1, 2, 9, 11, 15)$ $d(8, 10, 14)$

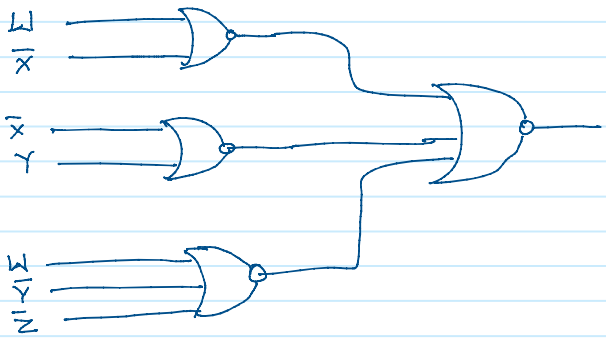
$\Pi(3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 16)$

$\Sigma(0,1,2,9,11,15)$ $d(8,10,14)$

$\pi(3,4,5,6,7,8,10,12,13,14,16)$



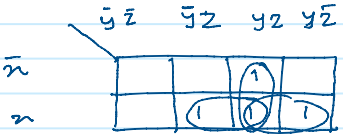
$f = (W+\bar{X}) \cdot (\bar{X}+Y) \cdot (W+\bar{y}+\bar{z})$



3

x	y	z	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	3 (1)	0	0
1	0	0	0	1	1
1	0	1	5 (1)	0	0
1	1	0	6 (1)	0	1
1	1	1	7 (1)	1	0

for A,



$f = xz + yz + xy = A$

$= B$

$= C$

4

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1

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

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

Q4