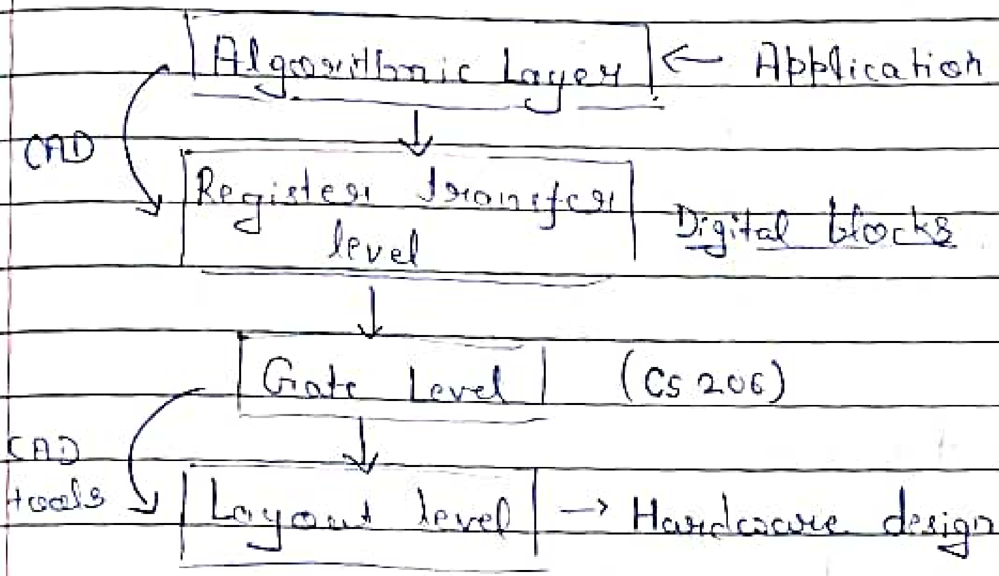
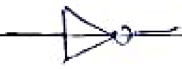
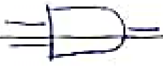
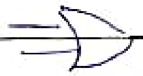
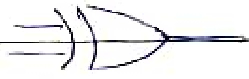
(Brown Vranesic "VHDL")

Digital Systems

(Morris Mano "Digital Logic")



Name	NOT gate	AND gate	OR gate	XOR gate
Expression	$\bar{A}$	AB	$A+B$	$A\bar{B} + \bar{A}B$
Symbol				

A	$\bar{A}$
0	1
1	0

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR (Y) = $\overline{XOR}$ $(AB + \bar{A}\bar{B})$

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

$= A \odot B$

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

Basic Axioms :-

1) $0+0=0$

2) $0+1=1$

3) $1+0=1$

4) $1+1=1$

5) $0.0=0$

6) $0.1=0$

7) $1.1=1$

~~8)~~

Boolean Laws :-

1) Commutative Law

$A+B=B+A$

$AB=BA$

2) Associative Law

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

$A(BC)=(AB)C$

3) Distributive Law:

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

Rules :-

1) $A+0=A$

2) $A+1=1$

3) $A\cdot 0=0$

4) $A\cdot 1=A$

5) $\cdot\, A+A=A$

6) $A+\bar{A}=1$

7) $A\bar{A}=0$

8) $A.A=A$

9) $\cdot\, \bar{\bar{A}}=A$

10) $A+AB=A(1+B)$

$=A$

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

$=A+B$ (absorption

12 $=(A+B)(A+C)=A+BC$

Date:
P. No:

Combining Law:-

1) $A(A+B) = A$

2) $AB + A\bar{B} = A$

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

4) $AC + B\bar{C} = AC + B\bar{C} + AB$ (consensus law)

Consensus Law:

$$F = AC + B\bar{C} + AB$$
$$= AC + B\bar{C} + AB(C+\bar{C})$$
$$= AC + B\bar{C} + ABC + AB\bar{C}$$
$$= AC(1+B) + B\bar{C}(1+A)$$
$$= AC + B\bar{C}$$
$$F = (A+B)(\bar{A}+C)$$

Demorgan's Law:-

1) $\overline{AB} = \bar{A} + \bar{B}$

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

Intel Quartus ~~VB1~~ v.3.1 (CAD tool)

Date:
P. No:

Tutorial - 1

1. Convert from decimal to binary:
a) 0.188
b) 410

2. Hexadecimal to decimal from 257.17B

3. Hexadecimal to binary of 3ACF7

4. Octal to decimal of a) 531 b) 320.127

5. Decimal to octal of 316.32

Ans. ① -

Subtract using Complement Method :-

Q.1→ Subtract $(1000)_2 - (110)_2$ using 2's complement metho

Q.2- Subtract $(13250)_{10} - (72532)_{10}$ Using 9's complement method

Q.3- Subtract $(3250)_{10} - (72352)_{10}$ using 10's complement metho

Soln ①.
- If there's a carry bit, drop the bit & the resut is a +ve number.

- If no carry bit, take 2's complement of the result which is negative.

Date:
P. No:

Code Conversion and BCD addition / subtraction :-

Q. 1- Convert $(10110001 1)_{BCD}$ to decimal.

Q. 2 Convert $(379)_{10}$ to BCD.

Q. 3- Convert Gray code $(11011)_{gray}$ to binary.

Q. 4- Convert binary $(10110)_{binary}$ to Gray code.

Q. 5- BCD addition of $(01100110)_{BCD}$ and $(01010011)_{BCD}$

Q. 6- Add following using BCD addition method : $(956)_{10}$ and $(452)_{10}$.

Q. 7. Subtract following using BCD subtraction method : $(17)_{10}$ and $(12)_{10}$.

Ans Q1- .

Minterm: In a f^n of 'n' variables, each term appears at least once.

SoP : $ABC \pm AC$ (Non-standard)

Product of Sum (PoS): $(A+B+C)\cdot\underset{\hookrightarrow \text{non-standar}}{(A+C)}$

Maxterm

Canonical/Standard SoP/PoS :-

e.g.① Convert :-

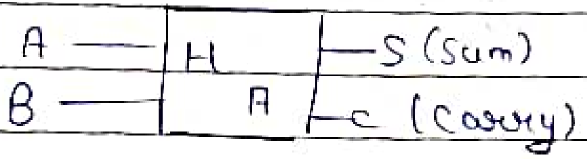
$$X = \bar{A}\bar{B} + ABC$$
$$= \bar{A}\bar{B}(C+\bar{C}) + ABC$$
$$= \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + ABC$$ (all are minterms)

e.g.② Convert :-

$$X = (\bar{A}+\bar{B})(A+B+C)$$
$$= (\bar{A}+\bar{B}+C\bar{C})(A+B+C)$$
$$= (\bar{A}+\bar{B}+C)(\bar{A}+\bar{B}+\bar{C})(A+B+C)$$ (all are Maxterm

Combinational Circuits :-

1) Half - Adder (HA). → (1-bit adder).
→ Addition of 2 boolean variables

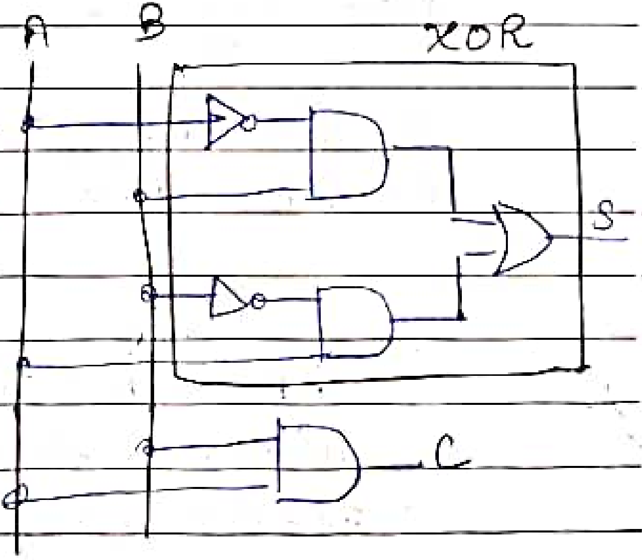


Truth Table :-

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

$S = \bar{A}B + A\bar{B} = A \oplus B$

$C = AB$



NAND logic : $S = \bar{A}B + A\bar{B}$

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

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

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

$= (A+B)\,\overline{AB}$

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

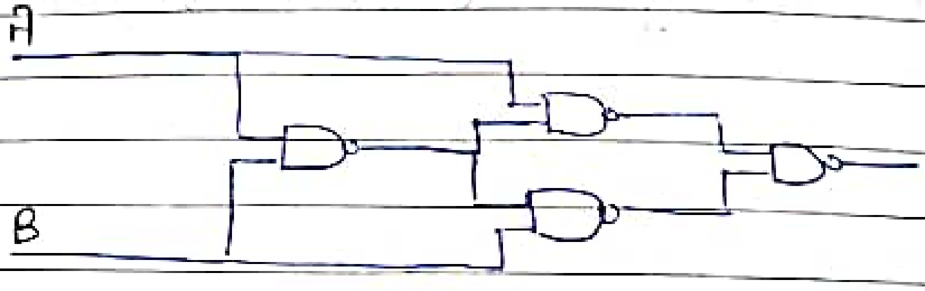
$\therefore S = \overline{\overline{(A.\overline{AB})}\cdot\overline{(B.\overline{AB})}}$

$C = \overline{\overline{AB}}$

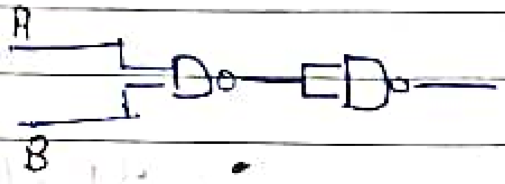
$\overline{\bar{x}\,\bar{y}} = x + y$

$\overline{xy} = \bar{x} + \bar{y}$

S:



C :



NOR logic :-

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

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

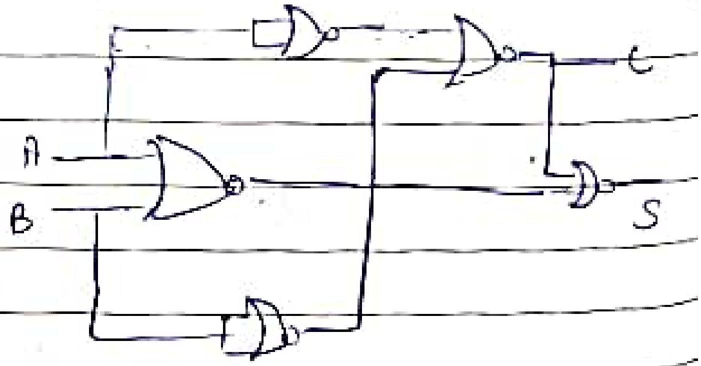
$S = A\bar{B} + A.\bar{A} + \bar{A}B + B\bar{B}$

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

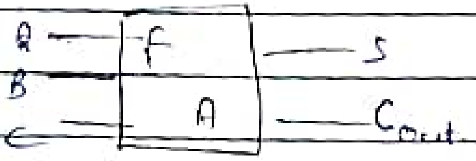
$= (A+B)(\bar{A}+\bar{B})$

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

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



② - Full adder :-



	A	B	C	S	C_{out}
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

SOP :-

$$S = \sum_m (1,2,4,7) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$C_{out} = \sum_m (3,5,6,7) =$$

$$S = \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$
$$= \bar{A}(B \oplus C) + A(B \odot C)$$
$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C})$$
$$\boxed{S = A \oplus B \oplus C}$$

$$\left[\bar{A}x + A\bar{x} = A \oplus x \quad x = B \oplus C\right]$$

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

Date:
P. No:

∴ $\boxed{C = AB + BC + CA}$

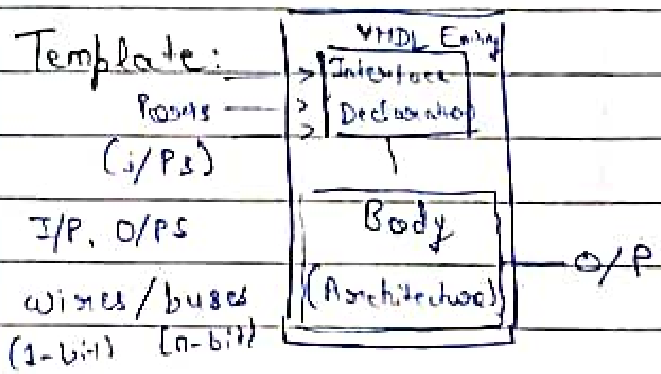
VHDL Fundamentals

VHDL : VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit.

Other languages: Verilog, AHDL, etc.

Template:



Ports : Primary I/P, O/P.

Keywords :-

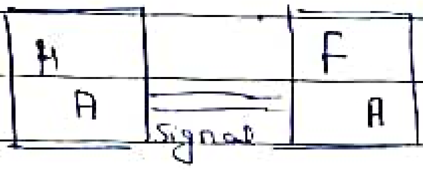
→ IS, IN, OUT, PORT MAP, END, START, INOUT, ENTITY, ARCHITECTURE

→ These are case insensitive

Descriptions:-

1) Signal names: Can be innumerated through commas(,)

2) Mode: IN- specifying that signal is an input
OUT- specifying signal is an output
INOUT- specifying signal is both i/p & o/p.
Buffer-



3) Type: Built-in or user defined signal type.
example :

bit: can have logical value 0 or 1.
std-logic: can have boolean values (1-bit info).
std-logic vector: can have vector of bit values (n downto 0).

{ 3 downto 0
4-bit }

e.g.

ENTITY AND ,

e.g. ENTITY AND2 IS
    Port(x : IN std-logic;
          y : IN std-logic;
          F : OUT std-logic);
end AND2;

architchure name

Architecture ~~AD~~ (AND2) of AND2 IS

entity name

Begin
    F <= x AND y ;
end AND2.

Libraries:-

Library IEEE;
USE IEEE ,std-logic-1164.All;
USE IEEE numeric-std.All;
USE IEEE std-logic-arith.All;

Date:
P. No:

Tutorial - 2

1. Express $F = A + B'C$ in canonical forms SoP and PoS.

2. $\pi[B + C(AB + AC)']$, no. of NAND gates required to design the logic.

3. Is $(A \odot B) \odot B = A \oplus (B \odot C) + (A \oplus A)$

4. $(A' + B) \cdot (A + B) = ?$

5. $F(A, B, C, D) = \pi(0, 1, 5, 7, 8, 9, 15)$
   $f(A, B, C, D) = \Sigma(2, 3, 4, 6, 11, 12)$
   $d(A, B, C, D) = (10, 13, 14)$

Ans. (1) $F = A + B'C$

$= A(BC + \overline{BC}) + AB'C + A'B'C$

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

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

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

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

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

$=$

$(A + C + \bar{C}$

$f = (F')' = (A' \cdot (B + C'))' = (A'B + A'C')'$

$(A + B' + C) \cdot (A + B' + C')(\quad)(\,) = (A + B') \cdot (A + C)$

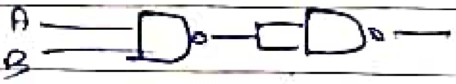
Date:
P. No:

2. $A[B + C(A\cdot(B+C))']$

$A[B + C(A' + B'C')]$

$A[B + A'C] + B$

AB

∴ 

4. $(A'+B)\cdot(A+B) = B$

5. a)

		$\bar C\bar D$	$\bar C D$	CD	$C\bar D$
		00	01	11	10
$\bar A\bar B$	00	1_0	1_1	0_3	0_2
$\bar A B$	01	0_4	1_5	1_7	0_6
AB	11	0_{12}	$\cancel{X}_{13}$	1_{15}	$\cancel{X}_{14}$
$A\bar B$	10	1_8	1_9	0_{11}	$\cancel{X}_{10}$

$\cancel{BD} + \bar B\bar C$

$(B\cancel{+\bar D})\cancel{+B+C}$

essential -> 2

n-essential

B.

6. $F = \cancel{A'C'}\cdot A + B + \cancel{A'}\cancel{B'}\overset{C}{\cancel{B}} + A'B'\cancel{C'}D + A'B'C'\cancel{D'}E$

$= A + B + C + \cancel{A'}D + \cancel{A'}\cancel{B'}\cancel{C'}E$

$= A + B + C + D + E$

$A'B'C'D + A'B'C'\cancel{D'}E$

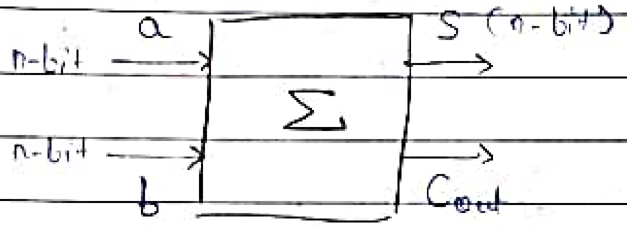
$A'B'C'(D + \cancel{D'}E)$

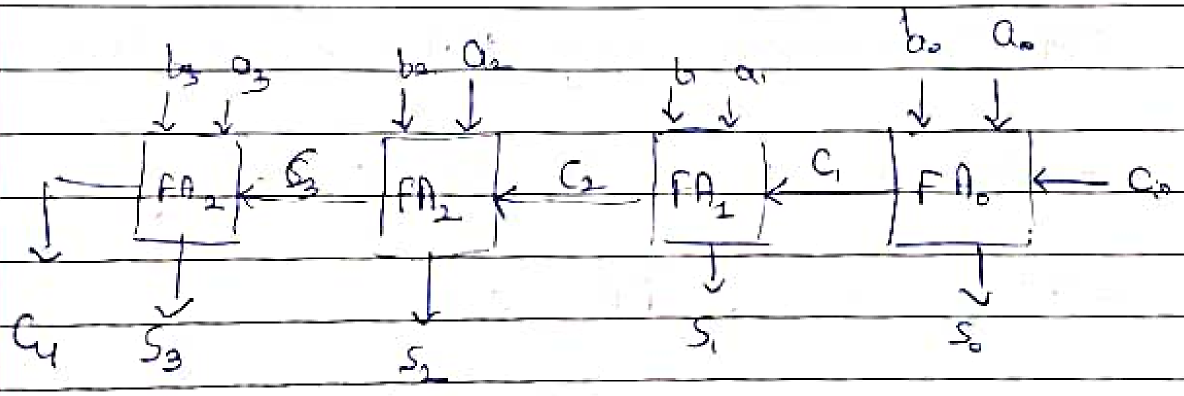
Parallel Adders

→ n-bit addition

→ variable size i/p binary string

→ unrestricted bit length





Parallel adder
(4-bit ripple carry adder)
(RCA)

$\boxed{S_3 S_2 S_1 S_0} \rightarrow S$

1) Area

2) Delay (high delay → low performance)
For higher order bits, the delay is large.
(Carry bit has to reach all the way to Cout)

Date:
P. No:

Delay of RCA :-

$T = (n-1) T_c + T_s$

T_c: delay of carry being rippled through the previous stages

T_s: Delay of producing the final stage sum

Area of RCA: $n \times A_{FA}$
(can be found using no. of gates → expressed using universal logic)

1 inverter → 1 nmos
1 cmos

CMOS → Complementary Metal Oxide Semiconducto

$n \times A_{FA} = (n-1) A_{FA} + A_{HA}$ (if $C_{in} = 0$)

e.g. $C_{in} = 0 \rightarrow FA_0 \rightarrow HA_0$

```
Entity FA is
   port (a, b, Cin: IN std_logic;
            S, C : OUT std_logic);
end FA;

Architecture FA behaviour of FA is

Beign
      S <= (a XOR b) XOR Cin;
      C <= (a AND b) OR (Cin AND (a XOR b));
End behaviour;
```

```
→ Library IEEE;
  Using IEEE.std-logic 1164.all;

Entity FOURBIT IS
    PORT (a,b : IN std-logic-vector (3 downto 0);
              Cin : IN std-logic;
               S : OUT std-logic-vector(3 downto 0);
             Cout : OUT std-logic);
End FOURBIT;

Architecture FOUR-struc of FOURBIT IS
      Signal c : std-logic-vector (4 downto 1);

   Component FA
         port (a,b,~~c~~, Cin : IN std-logic;
                  S, C : OUT std-logic);
   end component;

                      keyword
   Begin
       FA(0) : FA portmap (a(0), b(0), Cin, S(0), C(1));

       FA(1) : FA portmap (a(1), b(1), C(1), S(1), C(2));

       FA(2) : FA portmap (a(2), b(2), C(2), S(2), C(3));

       FA(3) : FA portmap (a(3), b(3), C(3), S(3), C(4));

   Cout = C(4);

END FOUR-struc;
```

(Schematic capture) → design entry modes

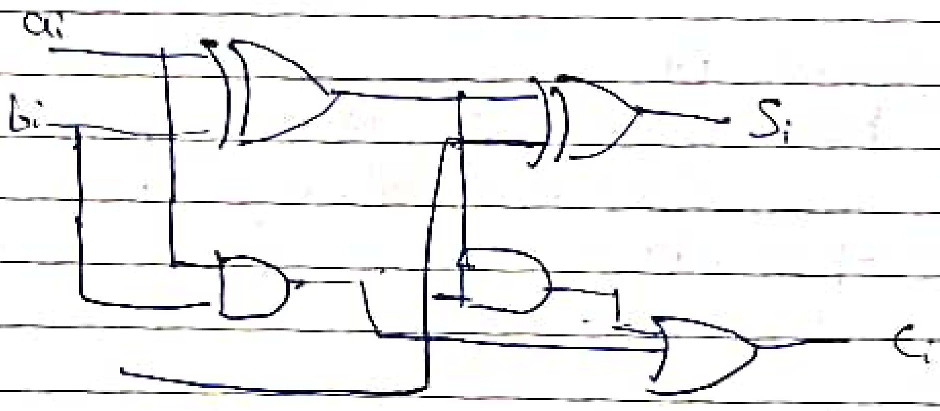
Adders
(contd.)

$C_{i+1} = a_i b_i + (a_i \oplus b_i) \cdot C_i$

$S_i = (a_i \oplus b_i) \oplus C_i$

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

where: $G_i = a_i b_i$ } Carry generate
$P_i = a_i \oplus b_i$ } Carry propagate.



$S_0 = \cancel{G}.\ P_0 \oplus C_0$

$C_1 = G_0 + P_0 C_0$. —(1)

$S_1 = P_1 \oplus C_1$

$C_2 = C_{1+1} = G_1 + P_1 C_1$

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

$= G_1 + P_1 G_0 + P_0 P_1 C_0$ —(2)

$S_2 = P_2 \oplus C_2$

$C_3 = C_{2+1}$

$= G_2 + P_2 C_2$

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

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_0 P_1 C_0$ —(3)

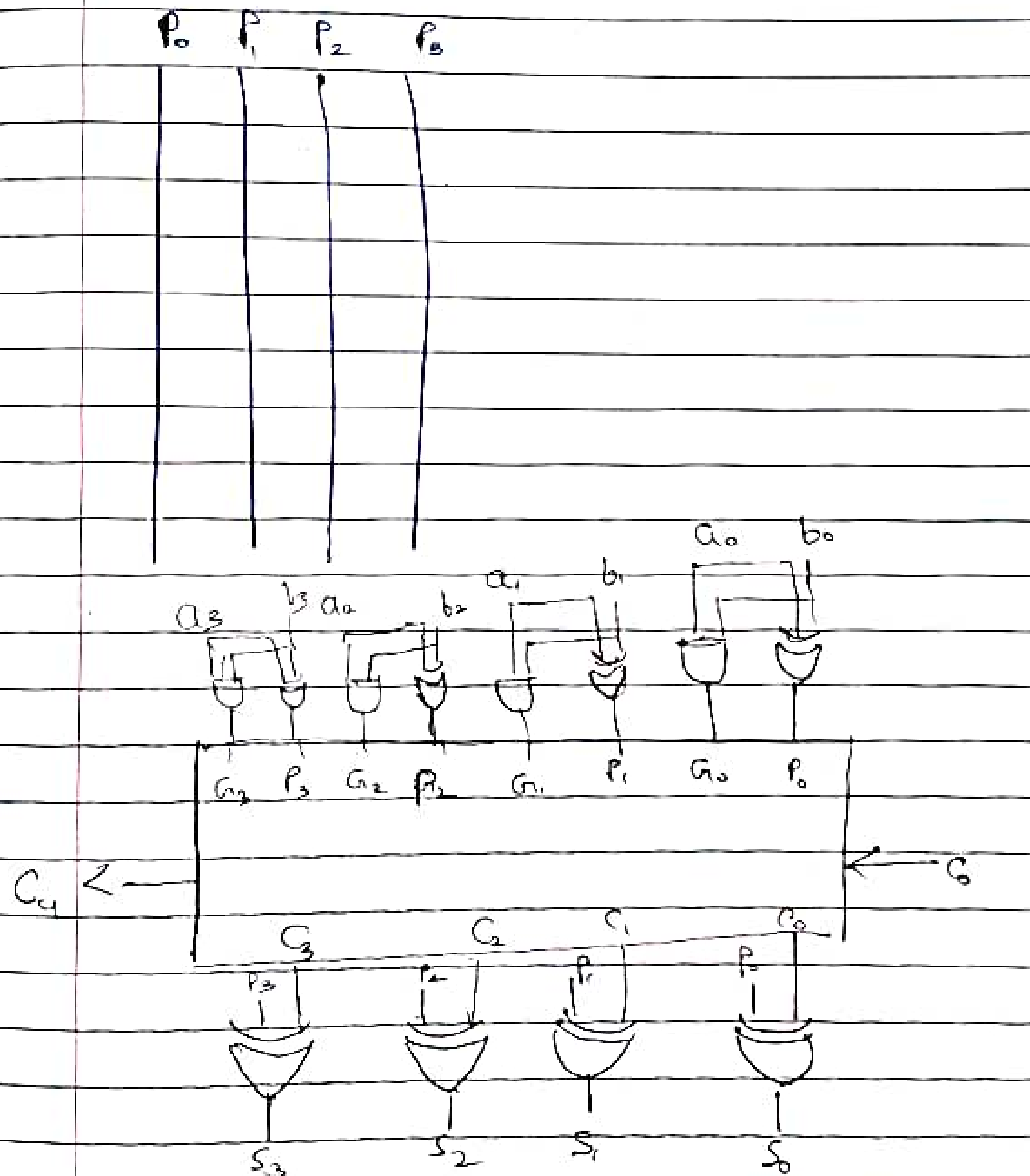
$S_3 = P_3 \oplus C_3$

$C_4 = C_{3+1}$

$= G_3 + P_3 C_3$

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

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$
$+ P_3 P_2 P_1 P_0 C_0$ —(4)



BCD Adder

- Adds 2 binary digits
- Each binary digit is represented as 4-bit number
- Produces sum between 0–9

- e.g. 526 is represented as

$$\underbrace{\overset{5}{0101}\quad\overset{2}{0010}\quad\overset{6}{0110}}_{}$$

010100100110

Case 1) Sum equals to 9 or less with carry 0 :-

6	0 1 1 0
+ 3	0 0 1 1
9 ←	1 0 0 1

Case 2) Sum greater than 9, with carry 1 :-

7	0 1 1 1
+ 9	1 0 0 0
carry ← ①	0 0 0 0

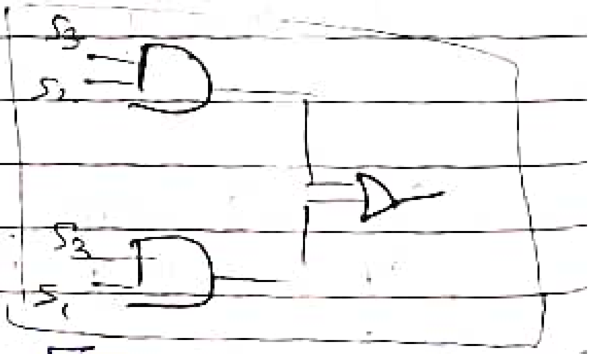
↓

sum the invalid O/P with '6'.

	0 0 0 0
	0 1 1 0
Carry → ①	0 1 1 0

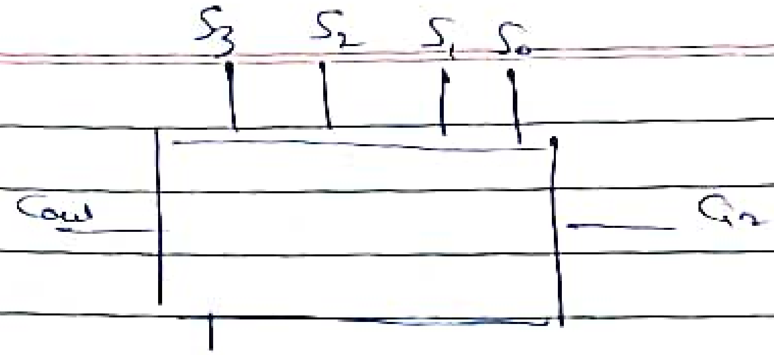
$$\begin{array}{rr} & 6 \\ + & 8 \\ \hline \end{array} \qquad \begin{array}{rrrrrrrr} & & & & 0 & 1 & 1 & 0 \\ & & & & 1 & 0 & 0 & 0 \\ \hline & & & & 1 & 1 & 1 & 0 \\ & & & & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

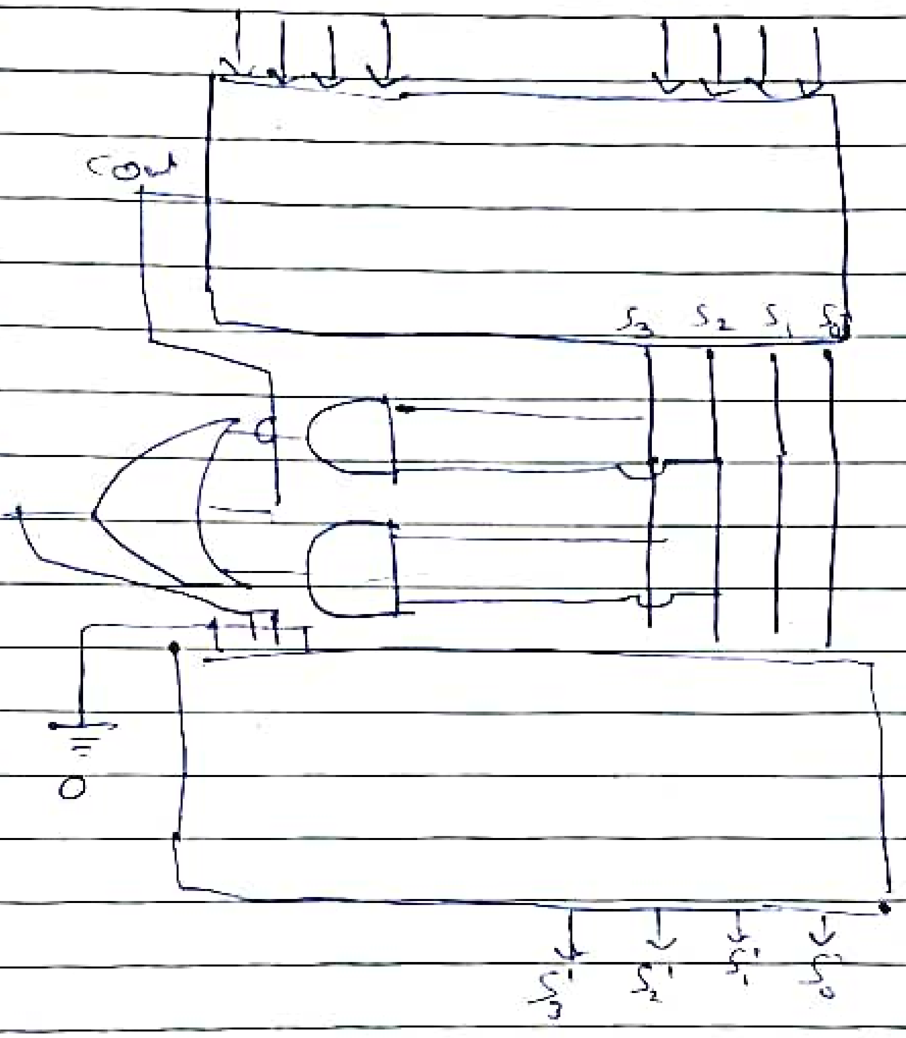
	S_3	S_2	S_1	S_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
Invalid	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1



Invalid BCD detector circuit.

Invalid → $S_3 S_2 + S_3 S_1$


S_3
S_2
S_1
S_0
Cout
Cin


Cout
S_3
S_2
S_1
S_0
0
S_3'
S_2'
S_1'
S_0'

Date:
P. No:

Tutorial-4

1.a) - $f = bc + bcd + ac' + ab$

$= bc + ac' + ab$

$= a(b+c) + bc$

$= a\overline{bc} + bc$

$= \overline{\overline{}}$

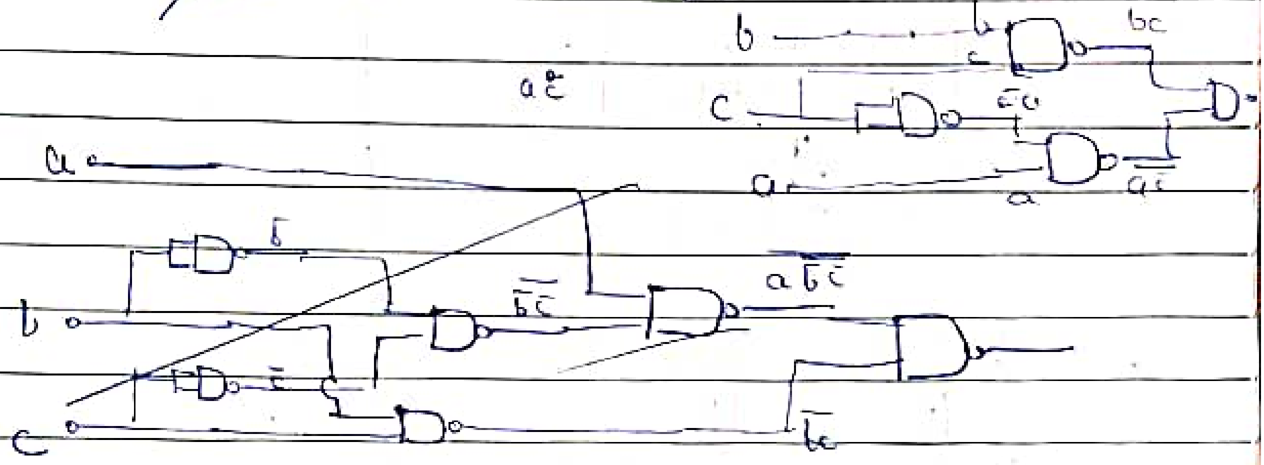
$\cancel{a\overline{bc} \cdot \overline{bc}}$

$bc + ac' + ab$

$bc + ac'$

$\overline{\overline{bc} * \overline{ac'}}$

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



$bc + \overset{ab}{cc'} + ac'$

~~$bc + bb'$~~

$b(c+a) + c'(c+a)$

$(c+a)(c'+b)$

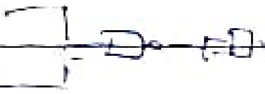
$cc' + bc + ac' + ab$

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

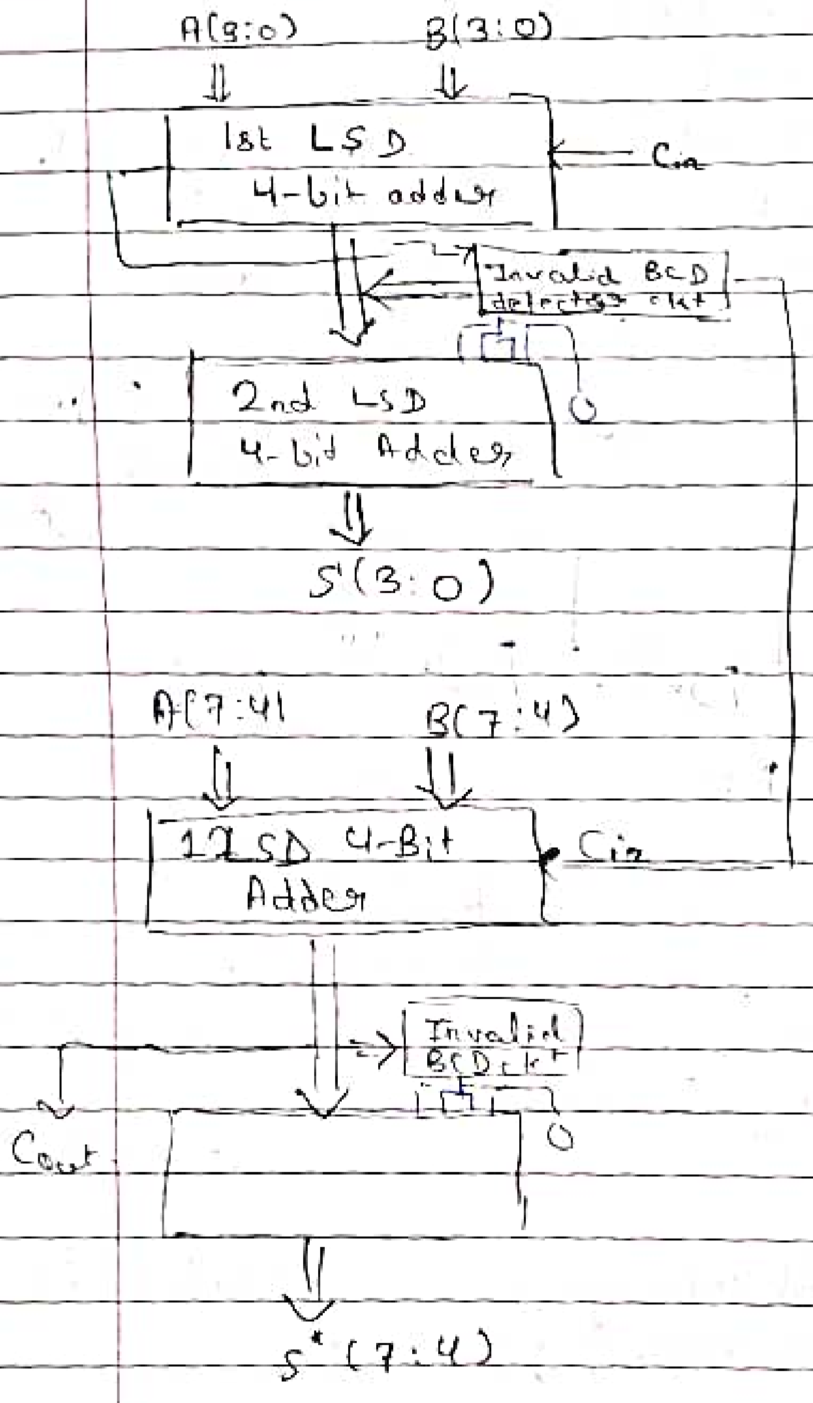
$\overline{\bar{c}\bar{a}} \cdot \bar{c} + \overline{\bar{c}\bar{a}} \cdot b$

b) ->

```
library IEEE;
 Use  IEEE. std_logic_1164.all;

entity circuit is
PORT (a, b, c, d : IN std_logic;
        f : OUT std_logic );
circuit;

architecture circuit of circuit is
begin
   f <= (c OR a) AND ((NOT c) OR b);
end circuit;
```

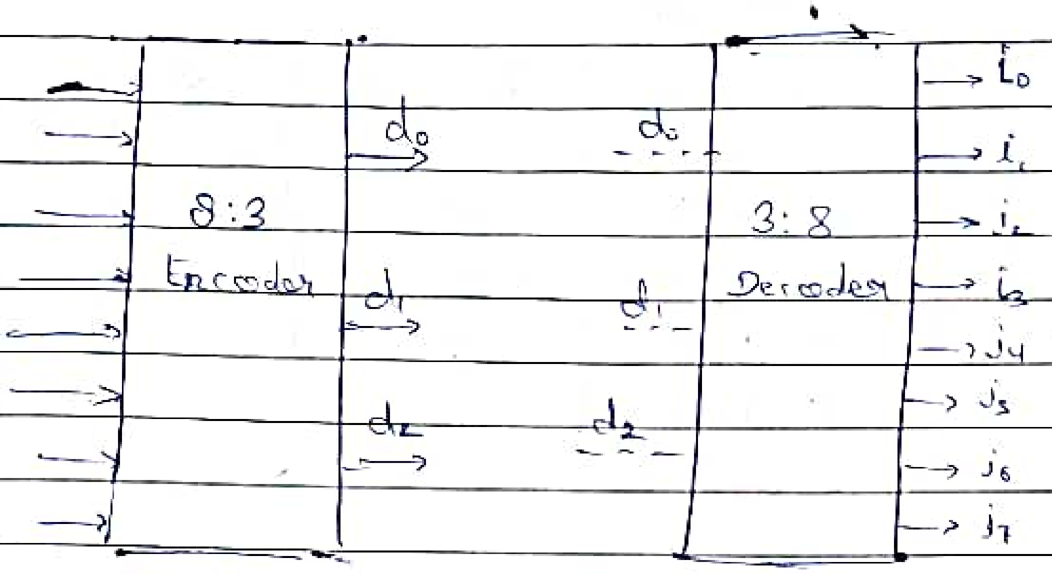


2-digit BCD adder:-

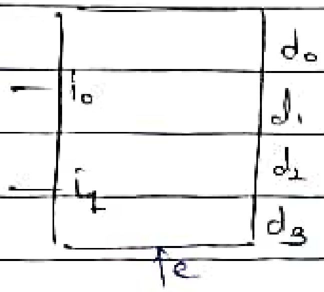


Components: IC 74LS 283 (4x)

Digital Transmission



2:4 Decoder:-



Decoder with enabler: Decoder is responsible for converting an I/P binary no into a high O/P line.

e	i1	i0	d0	d1	d2	d3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

2bit I/P 4-bit I/P.

It is a combinational ckt. that converts binary info from "n" i/p lines to "2^n" o/p lines.

@e=0, the decoder ckt is disabled OFF
@e=1, the decoder ckt works normally

Priority Encoder:-

Priority ←

4:2 P.E.

D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

e.g.:- $D_3 > D_2 > D_1 > D_0$

$A_0 = D_3 \bar{D}_2 D_1 + D_3$

$A_1 = \bar{D}_3 D_2 + D_3$

Date:
P. No:

Tutorial-6

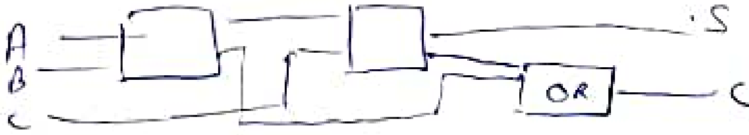
1. Structural VHDL for full adder:-

```
library IEEE;
use IEEE.std_logic_1164.all;

entity or_2 is
   PORT(a,b: IN std_logic;
            x : OUT std_logic);
end or_2;

architecture or_2 of or_2 is
  begin
        x ⇐ a OR b;
end or_2;
```

''

"

```
entity half_adder is
  PORT(x,y :IN std_logic;
          s,c : OUT std_logic);

architecture half_adder of half_adder is

     component or_2
          PORT (a,b : IN std_logic;
                  x : OUT std_logic);
     end component;
```



Date:
P. No:

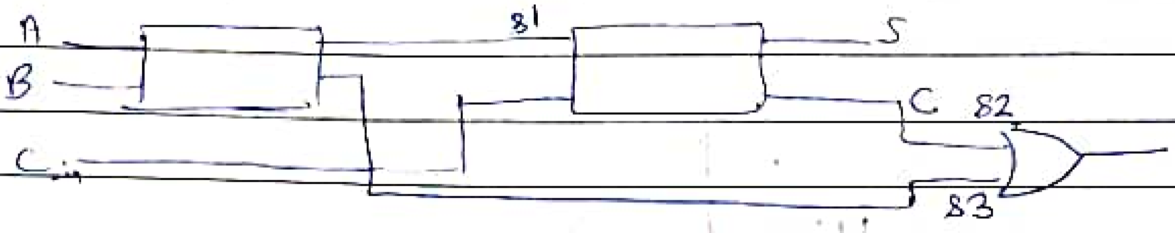
```
        begin
              s <= x XOR y;
              C <= x AND y;
end half_adder;

"
"

entity full-adder is
      PORT (A, B, C_in : IN std_logic;
               S, C  : OUT std_logic)
end full_adder;
signal s1, s2, s3 : std_logic;
architecture full- adder of full-adder is

     component half_adder is
           PORT(x,y : IN std_logic;
                 s,c : OUT std_logic);
     end component;
→    component or_2 is ( . . . . .          end component;
     begin
        FA1: half adder port map (A, B, s1, s3);
        FA2: half_adder port map (s1, C_in, S, s2);
        or_c : or_2 port_map (s2, s3, C);

end full_adder;
```

~~$A \oplus B + (AB)$~~ $\oplus$

$AB + C(A \oplus B)$

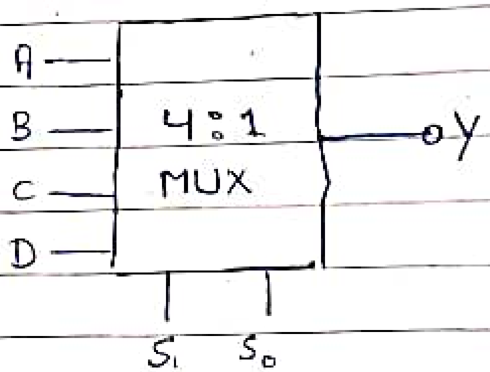
Date:

P. No:



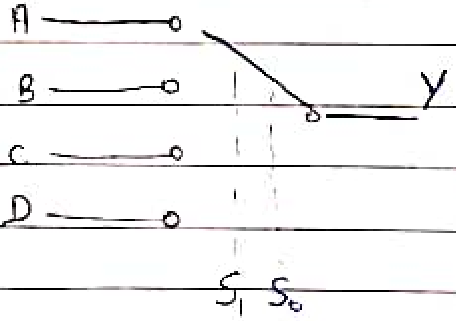
Date:
P. No:

MULTIPLEXER



Def. A digital ckt. with 2^n inputs & one output line ($2^n : 1$)

n: # select lines/control lines.



4 input Analog MUX (switch)

Encoder: $2^n \to n$

Decoder: $n \to 2^n$

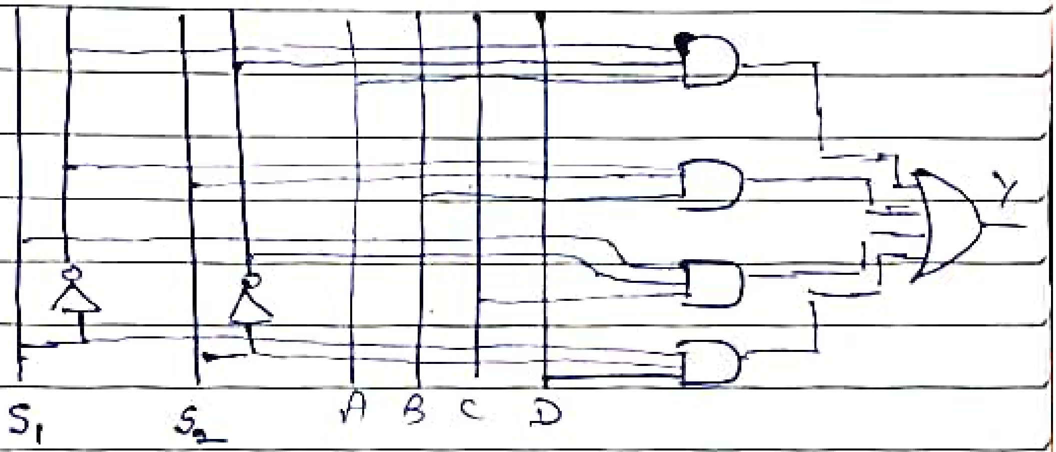
MUX: $2^n \to 1$

Date:
P. No:

Truth Table :-

S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

$\longrightarrow$ $Y = S_1'S_0'A + S_1'S_0B + S_1S_0'C + S_1S_0D$



S_2	S_1	S_0	Y
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	F
1	1	0	G
1	1	1	H

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0A + \bar{S}_2\bar{S}_1S_0B + \bar{S}_2S_1\bar{S}_0C + \bar{S}_2S_1S_0D$$
$$+ \bar{S}_2\bar{S}_1\bar{S}_0E + S_2\bar{S}_1S_1F + S_2S_1\bar{S}_0G + S_2S_1S_0H$$

$y = x_1 \times x_2 + m_1 \times m_2 + n_1 \times n_2 + z_1 \times z_2$

Multiplier (1M)   Adder (1A)

Tri-State Buffers (TSB)

A TSB has one i/p, one o/p and one control line (e).

e	x	f
0	0	z
0	1	z
1	0	0
1	1	1

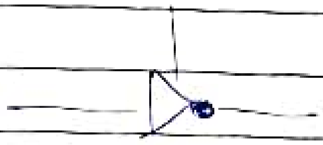
Active-high non-inverted TSB

"f" can have 3 o/p states 'z', '0', '1'

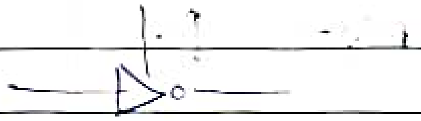
z → high-temperature (open-ckt)

If $e = 0$, $y = z$
else $y = x$

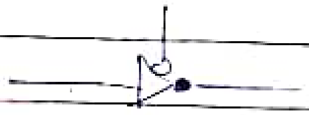
Symbols :-



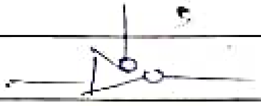
a) Active high non-inverted TSB



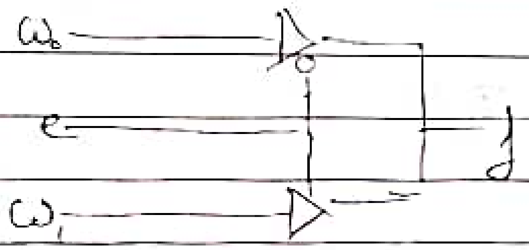
b) Active - high inverted TSB



c) Active low non-inverted TSB



d) Active low inverted TSB



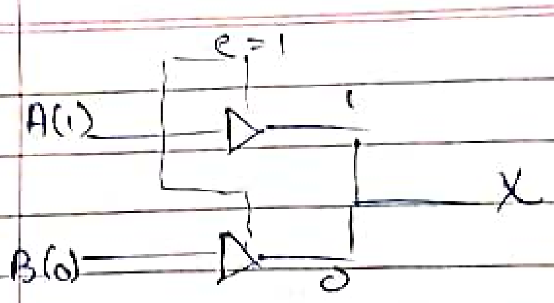
0, Z → Z

~~0~~1, Z → 1

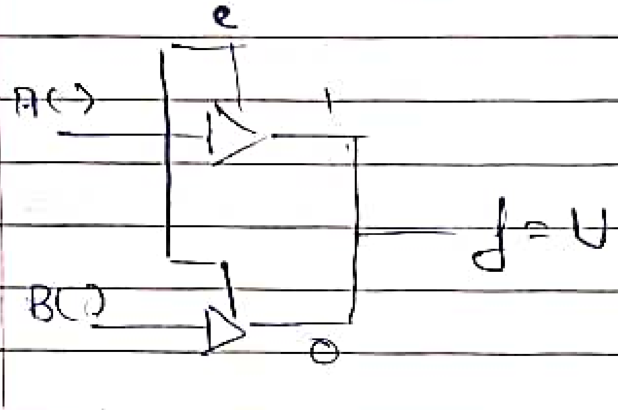
'0', '1', 'Z', 'X' → 'X': don't care

'X': conflict of output

'U': uninitialized i/p value.



X → 1 and 0 gives conflict output.



5-valued logic : 1, 0, X, Z, U

Parity Generator

Function → code word.

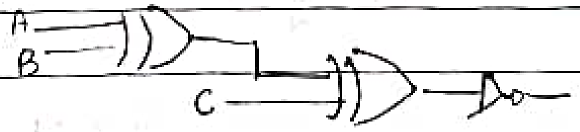
1 data word

A	B	C	O/P (P_o)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Concept: Adding a parity bit to the data word, such that the code-word has odd no. of 1s.

$$P(A,B,C) = \Sigma(0,3,5,6)$$

$$P = A'B'C' + A'BC + AB'C + ABC'$$
$$= C(A'B + AB') + C'(A'B' + AB)$$
$$= C \oplus A \oplus B$$



Single event upset (SEU) → We send an extra-bit to realize if bit flip occured during transmission.

Detects but does not rectify. Can't detect multiple bit flips.

Odd Parity Checker:-

	A	B	C	P	PEC
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

Concept: Checks the 4-bit code word, to check for odd number of 1's.

If odd # 1's are present, then PEC = 0
else PEC = 1.

$PEC = \sum_m (0, 3, 5, 6, 9, 10, 12, 15)$

Date:
P. No:

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1_0	0_1	1_3	0_2
$\bar{A}B$	0_4	1_5	0_7	1_6
AB	1_{12}	0_{13}	1_{15}	0_{14}
$A\bar{B}$	0_8	1_9	0_{11}	1_{10}

PEC : $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D$
$+ A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABCD$

$\rightarrow \bar{A}(\bar{B}\bar{C}\bar{D} + \bar{B}CD + B\bar{C}D + BC\bar{D}) + A(\bar{B}\bar{C}D + \bar{B}C\bar{D}$
$+ B\bar{C}\bar{D} + BCD)$

$\rightarrow \bar{A}(B \oplus C \oplus D) + A(\overline{B \oplus C \oplus D})$
$\rightarrow \overline{A \oplus B \oplus C \oplus D}$ (D = P)
$\rightarrow \overline{A \oplus B \oplus C \oplus P}$

	A	B	C	D	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

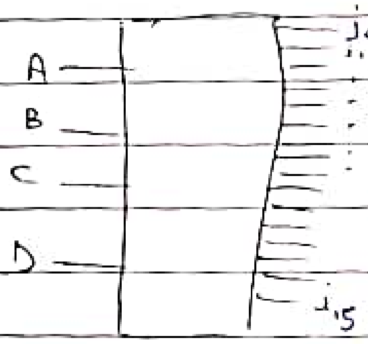
$E3 = \sum m(5,6,7,8,9)$
$E_2 = \sum m(1,2,3,4,9)$
$E_1 = \sum m(0,3,4,7,8)$
$E_0 = \sum m(0,2,4,6,8)$

BCD to XS-3 converted

0101

Date:

P. No:



$E_3 =$ ~~$i_4 + i_5 + i_6 + i_7 + i_8$~~ $i_5 + i_6 + i_7 + i_8 + i_9$

$E_2 = i_q + i_2 + i_3 + i_4 + i_9$

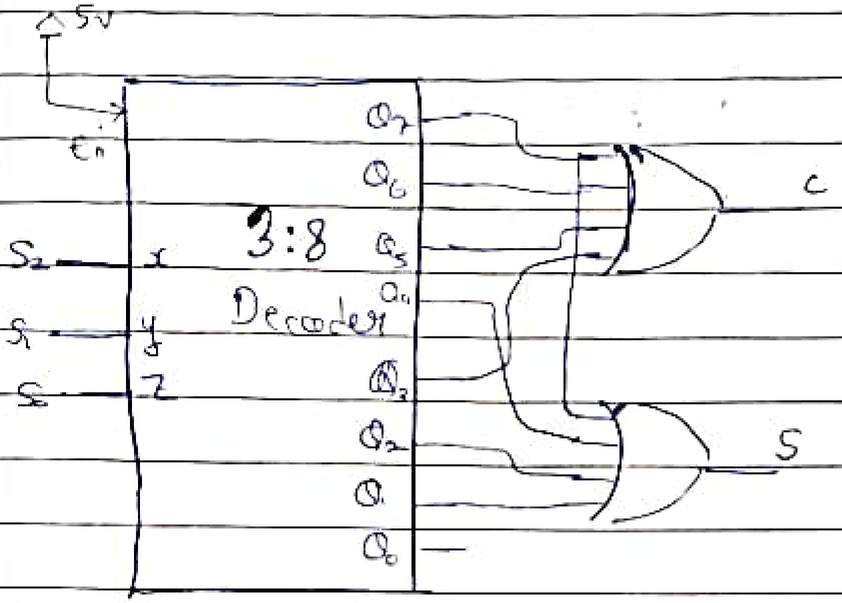
$E_1 = i_0 + i_3 + i_4 + i_7 + i_8$

$E_0 = i_0 + i_2 + i_4 + i_6 + i_8$

Date:
P. No:

Digital Design

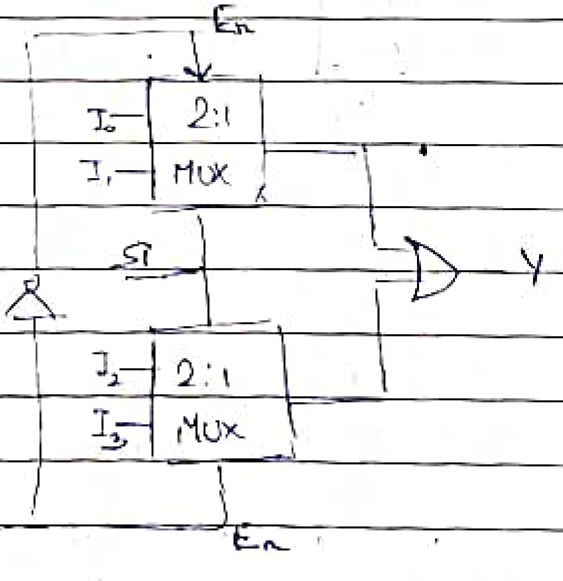
Implementing Adder Using Decoder :-



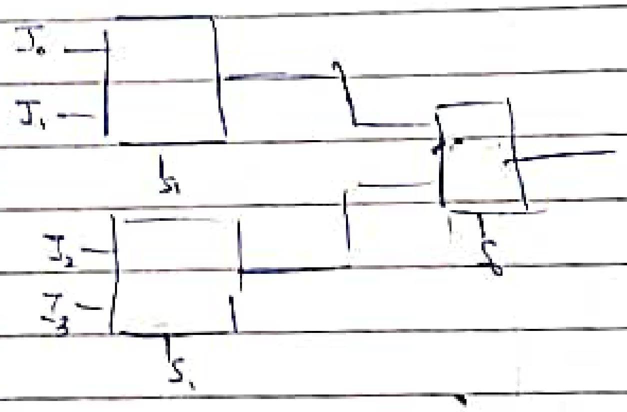
$C = \sum m(3,5,6,7)$
$S = \sum m(1,2,4,7)$

Designing 4:1 MUX :-

S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

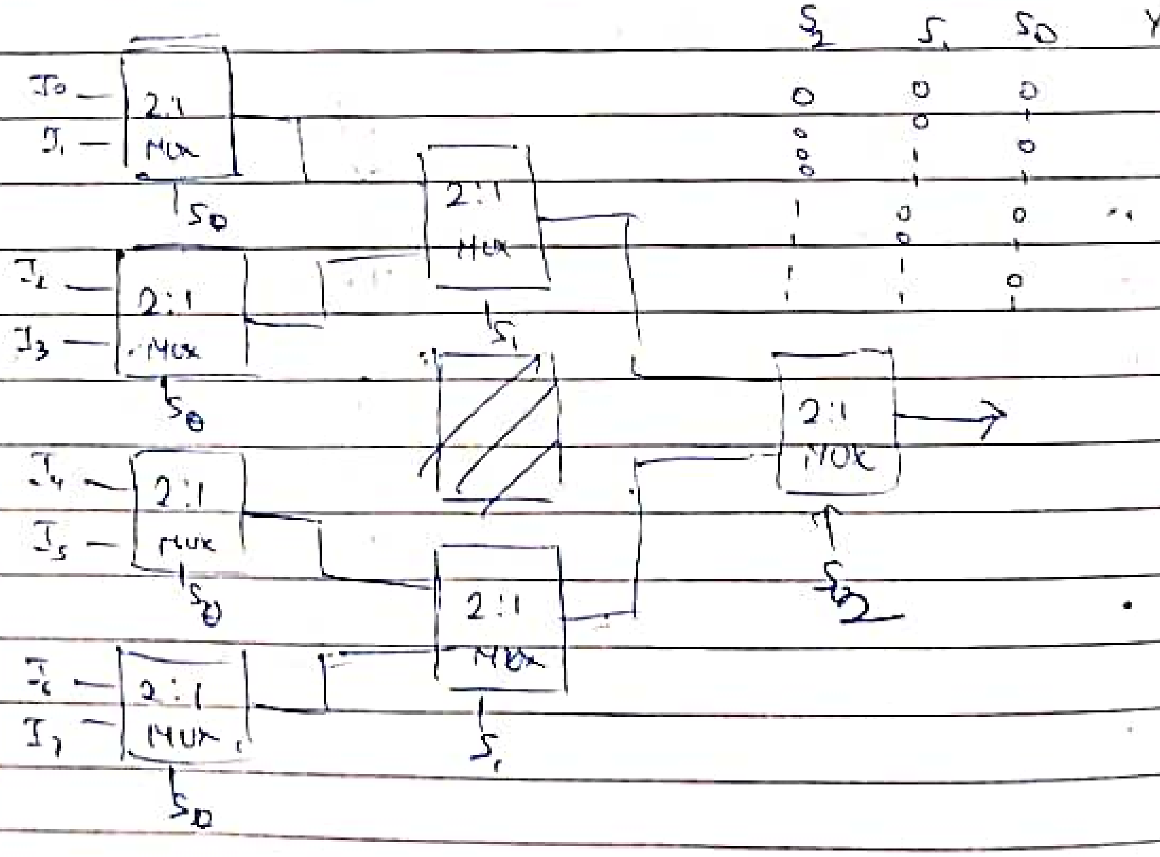


Designing 4:1 MUX using 2:1 MUX only:-

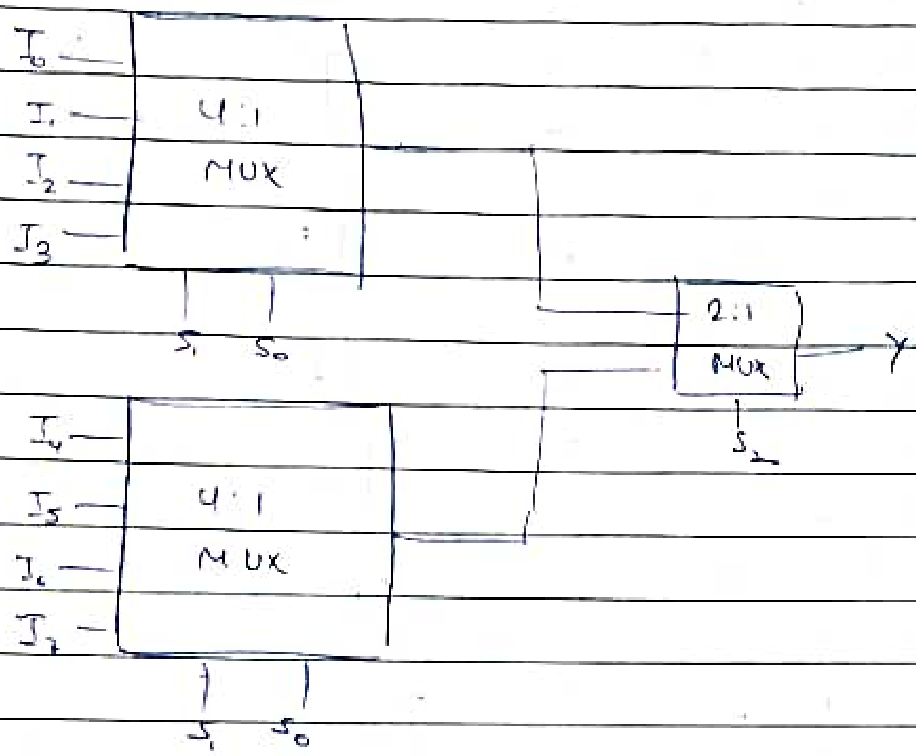


Design 32:1 MUX using 16:1 MUX & OR gate:-

Cascading of MUXES:-



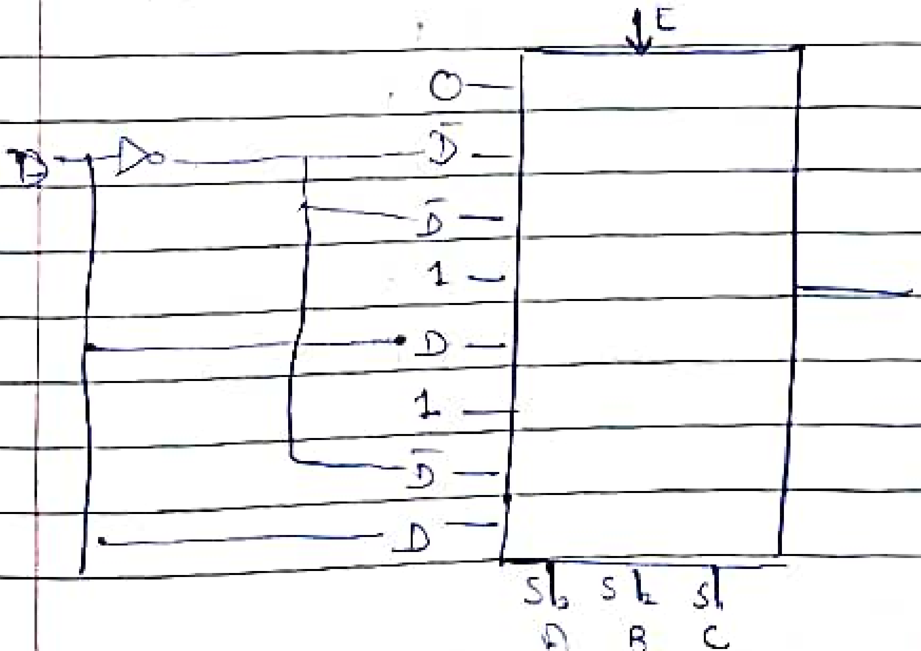
Design 8:1 MUX using 4:1 MUX and 2:1 MUX



Ques: $f(A, B, C, D) = \sum_m (2, 4, 6, 7, 9, 10, 11, 12, 15)$

Implement the boolean function using 8:1 MUX.

→



S_3	S_2	S_1	S_0	Y	
0	0	0	0	0	0
0	0	0	1	0	
0	0	1	0	1	$\bar{D}$
0	0	1	1	0	
0	1	0	0	1	$\bar{D}$
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	D
1	0	0	1	1	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	1	$\bar{D}$
1	1	0	1	0	
1	1	1	0	0	D
1	1	1	1	1	

000

001

011

010

100

101

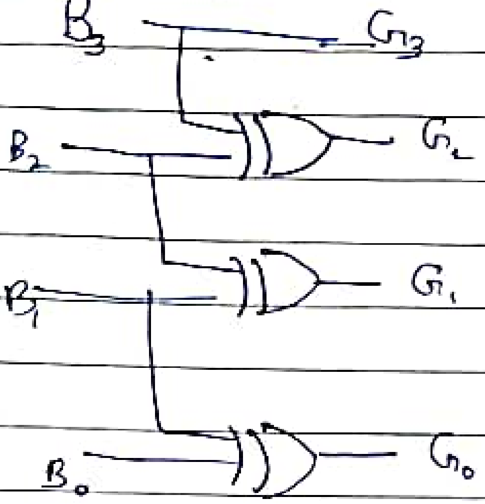
111

110

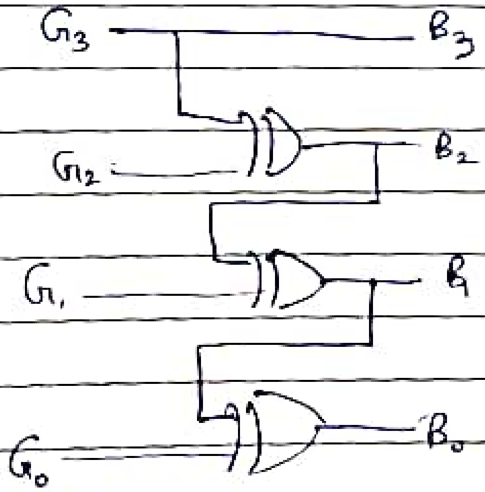
Date:
P. No:

Tutorial - 8

(1). Binary to Gray :-



(2). Gray to binary :-



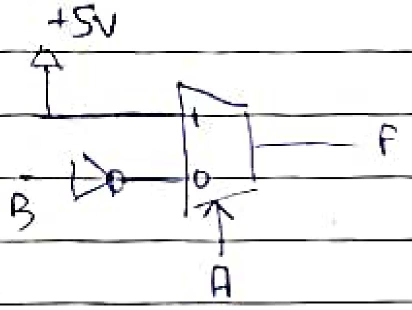
Implementation of Boolean functions

1) Implementing NOR gate using MUX :-

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

⇒

A	F
0	B'
1	0



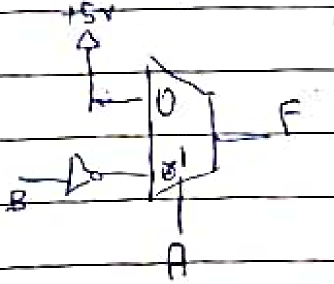
2) Implementing NAND gate using MUX :-

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

⇒

A	F
0	1
1	B'

⇓



Date:
P. No:

3/. Implementing INVERTER using MUX :-

Shannon's expansion →

$f = x f_x + x' f_{x'}$,

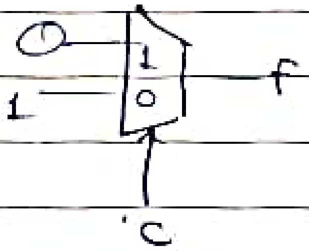
where f is any function
f_x and $f_{x'}$ are the positive and negative shannon co-factors.

+ve cofactor is evaluated @ $x=1$
neg. cofactor is evaluated @ $x=0$

→ $f = c'$

$x = \cancel{0}\ \cancel{1}$	$x = 0$
$\cancel{f_x} = c'$	$f_x = c$

$\Rightarrow f = 1c' + 0c$ (or) $0c + 1c'$



$A X + A'x$

$AB + A'B = B$

→ $F = AB + AC + BC$

Use Shannon's expansion w.r.t. A.

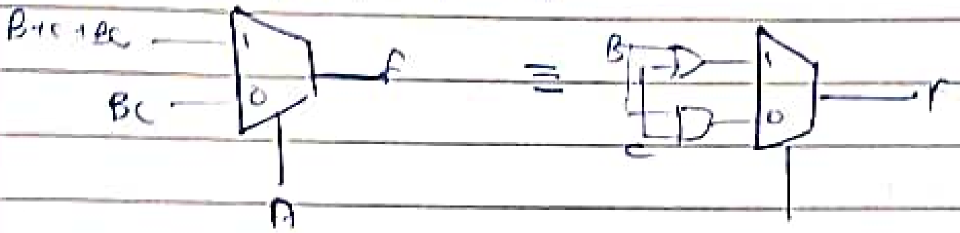
$F = A(B+C) + BC$

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

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

$= A(1B + 1C + 1BC) + A'(0B + 0C + 1BC)$

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



Sol^n - $F = A F_A + A' F_{A'}$

$F = AG + A'H$ $\begin{cases} G = B+C \\ H = BC \end{cases}$

$G = B G_B + B' G_{B'}$

$= B(1+C) + B'(0+C)$

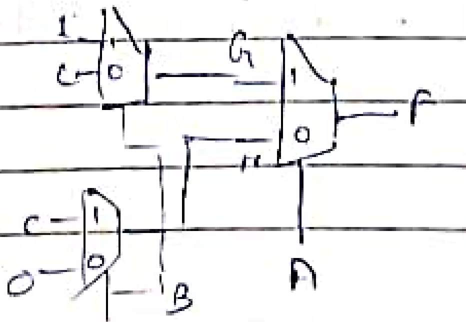
$= B(1) + B'(C)$

$H = BC$

$H = B H_B + B' H_{B'}$

$= B(1.C) + B'(0.C)$

$= B(C) + B'(0)$



Date:
P. No:

Digital Comparators:-

1-Bit Comparator:-

A_i	B_i	E_i	L_i	G_i
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

→ $L_i = \bar{A}_i B_i$

→ $G_i = A_i \bar{B}_i$

→ $E_i = \bar{A}_i \bar{B}_i + A_i B_i$

$= \underbrace{A_i \odot B_i}_{x\bar{n}}$

Let; $A = A_3 A_2 A_1 A_0$

$B = B_3 B_2 B_1 B_0$

→ $E = (A_3 \odot B_3)(A_2 \odot B_1)(A_1 \odot B_1)(A_0 \odot B_0)$

→ $L = \bar{A}_3 B_3 + (A_3 \odot B_3)(\bar{A}_2 B_2) + (A_3 \odot B_3)(A_2 \odot B_2)(\bar{A}$

$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(\bar{A}_0 B_0)$

→ $G = A_3 \bar{B}_3 + (A_3 \odot B_3)(A_2 \bar{B}_2) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \bar{A}_1)$

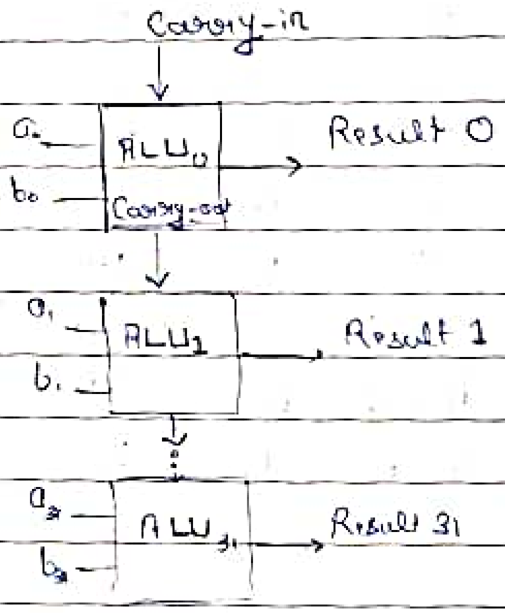
$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \bar{B}_0)$

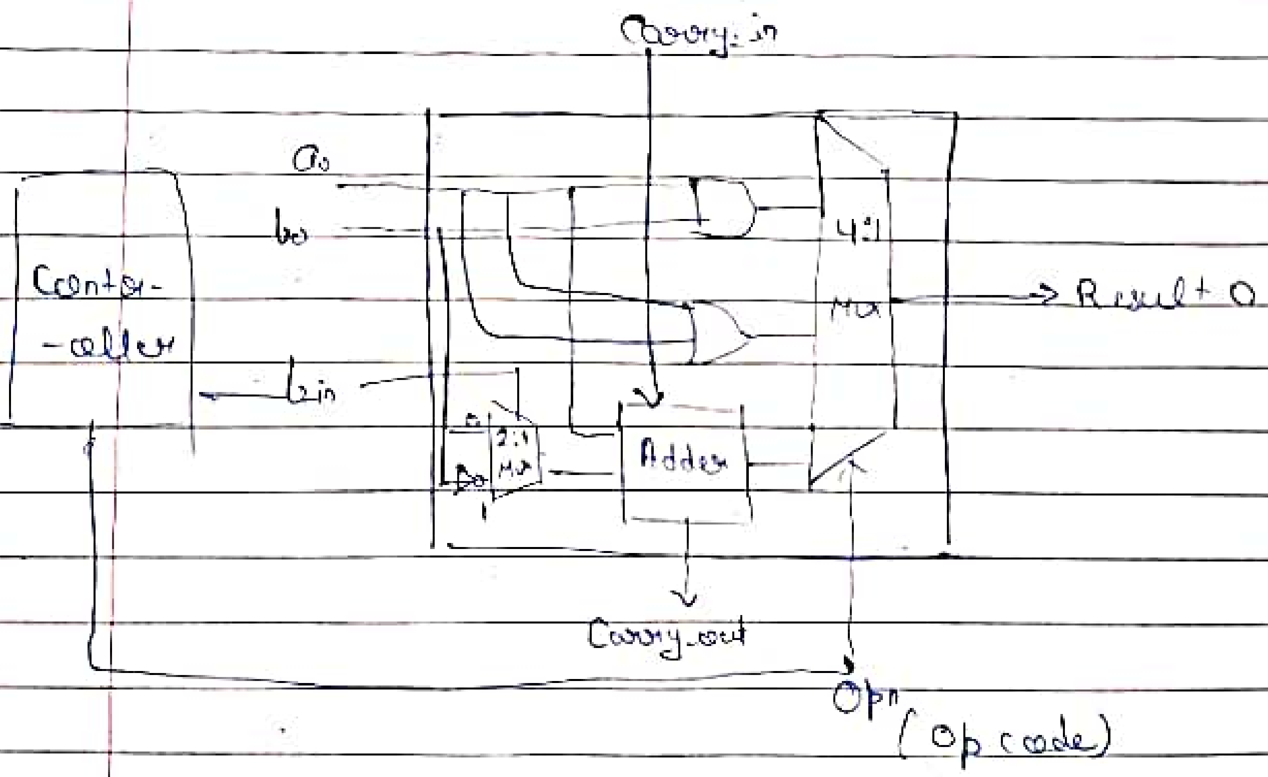
Date:

P. No:

A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$A > B$	$A < B$	$A = B$
$A_3 > B_3$	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	H

Arithmetic Logic Unit (ALU):-





Controller + ALU = Datapath

Crontrol Lines :- (Signals)

b_inv	opn	
0	00	AND
0	01	OR
0	10	ADD
1	10	SUBTRACT

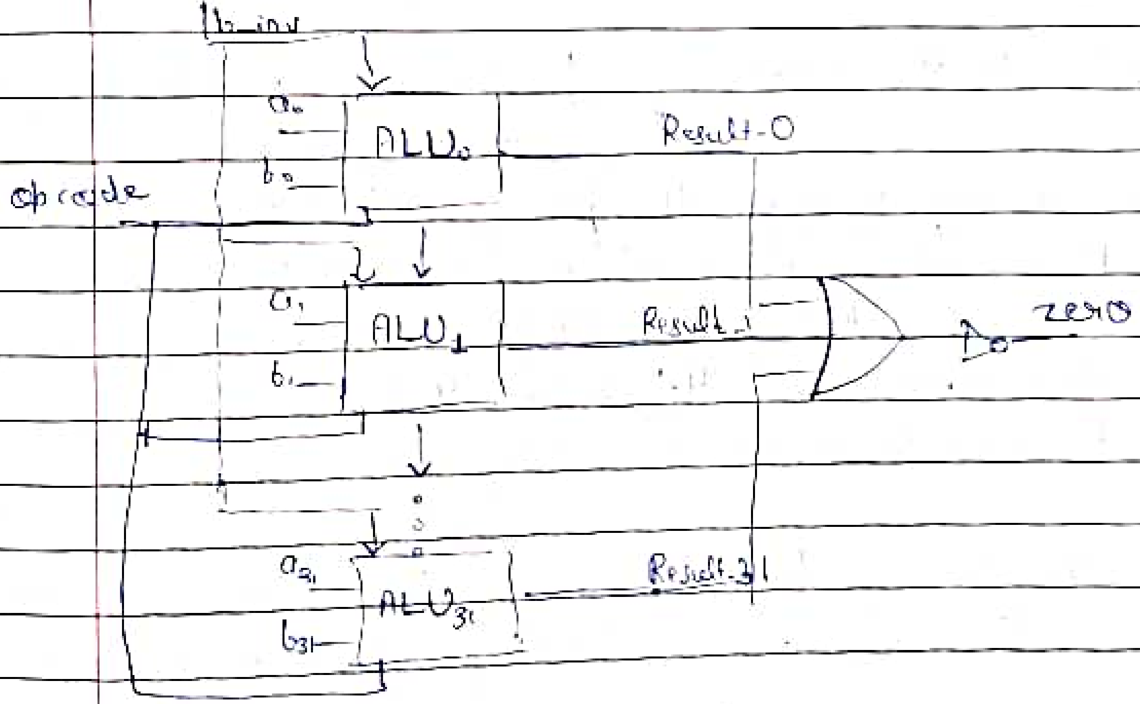
Test of Equality :-

if A - B = 0

then zero(flag) = 1

else

zero = 0



Date:
P. No:

Hamming Code :-

① ② 3 ④ 5 6 7

P_1 P_2 D_1 P_3 D_2 D_3 D_4

4-bit data word

7-bit code word

3-bit parity

Rules for Hamming Code Generation

Step 1: Mark all bits that are power of 2 as parity bits. (1, 2, 4, ...)

Step 2: Mark all the remaining positions for data encoding

Step 3: Each parity calculates the parity of some bits.

Position 1: Check 1 bit, Skip 1 bit, Check 1 bit, Skip 1-bit, ...

~~Step 4:~~ Position 2: Check 2-bit, Skip 1-bit, ...

Position 3: Check 4-bits, Skip 4 bits, ...

Position 8: Chek 8bits, Skip 8 bits, ..

Step 4: Set the parity bit to '1' if there are odd no. of '1' (because we're checking for even parity.)

Date:
P. No:

Data word :-

→ 1 0 0 1 1 0 1 0

1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	0	0	1	0	1	0	1	0
P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5	D_6	D_7	D_8

Pos 1 → 1, 3, 5, 7, 9, 11, . . .
Pos 2 → 2, 3, 6, 7, 10, 11, . . .
Pos 4 → 4, 5, 6, 7, 12, 13, 14, 15 . . .

Transmitted Code Word → 0 1 1 1 0 0 1 0 1 $\boxed{0}$ 1 0

Received Code Word → 0 1 1 1 0 0 1 0 1 $\boxed{1}$ 1 0

① Check for even parity @ pos's : 8, 9, 10, 11, 12
→ fails
→ $x_1 = 1$

②. Check for even parity @ posⁿs : 4, 5, 6, 7, 12
→ passes
→ $x_2 = 0$

③. Check for even " " " : 2, 3, 6, 7, 10, 11
→ fails
→ $x_3 = 1$

Date:
P. No:

→ Check for even parity @ $pos^n s$: 1,3,5,7,9,11

→ passes

→ $x_4 = 0$

$x_1 \quad x_2 \quad x_3 \quad x_4$

$$\underbrace{1 \quad 0 \quad 1 \quad 0}_{pos^n \, 10}$$