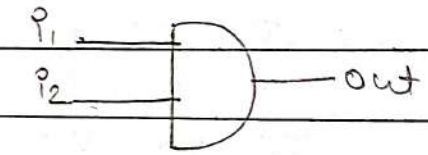
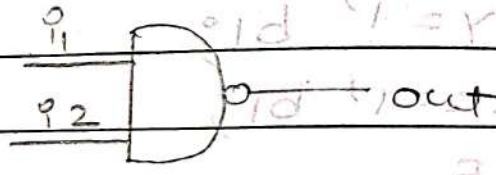
DATE : / /

Unit: 3 :-

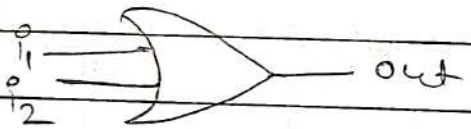
* Gate level modeling :-



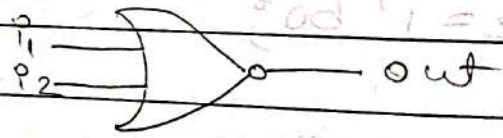
And



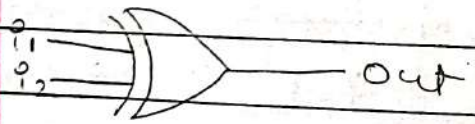
nand



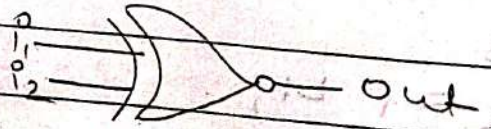
OR



NOR

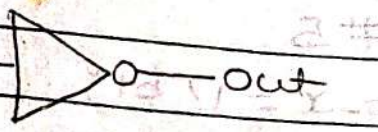


XOR

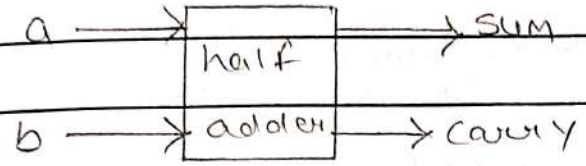


XNOR

and a1 (out, i1, i2);
nand na1 (out, i1, i2);
or o2 (out, i1, i2);
nor no2 (out, i1, i2);
xor x1 (out, i1, i2);
xnor x2 (out, i1, i2);
not n1 (out, in);



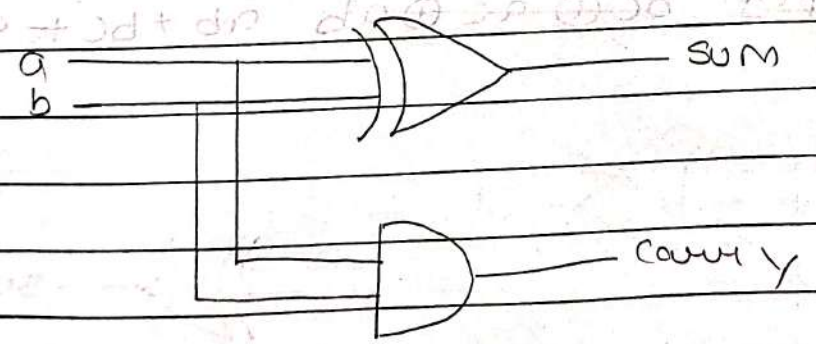
1. Half Adder :-



a	b	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = $a \oplus b$

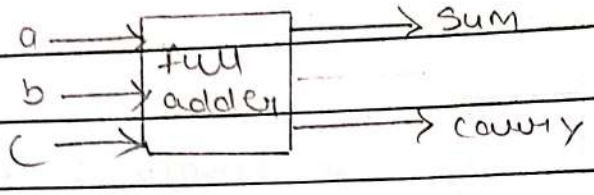
Carry = ab




```
Module    half_adder (a, b, carry, sum);
Output    carry, sum;
Input     a, b;
xor       x1 (sum, a, b);
and       a1 (carry, a, b);
endmodule
```

DATE : / /

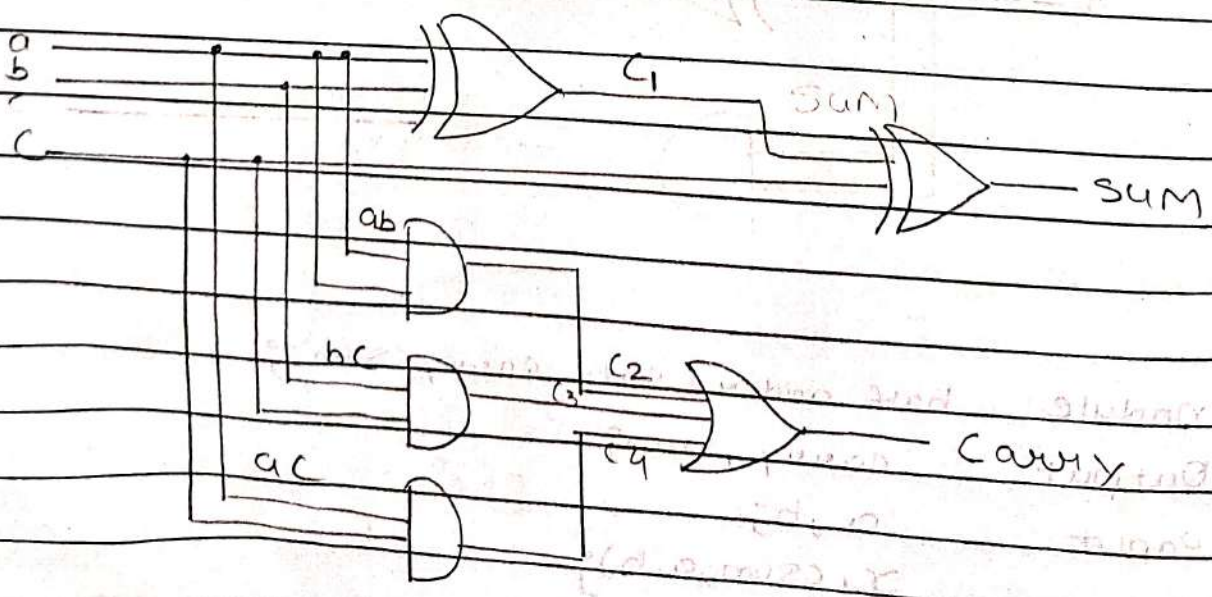
2. Full adder :-



a	b	c	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

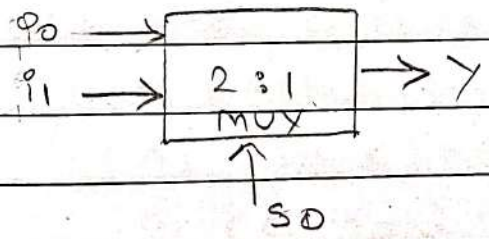
Sum = $a \oplus b \oplus c$ $\quad a \oplus b \oplus c$

Carry = ~~$(b \oplus c)$~~ $bc \oplus ac \oplus ab$ $\quad ab + bc + ac$



```
module full_adder (a, b, c, sum, carry);
output sum, carry;
input a, b, c;
wire c1, c2, c3, c4;
xor x1 (c1, a, b);
xor x2 (sum, c1, cin);
and a1 (c2, a, b);
and a2 (c3, b, c);
and a3 (c4, a, c);
or a4 (carry, c2, c3, c4);
endmodule
```

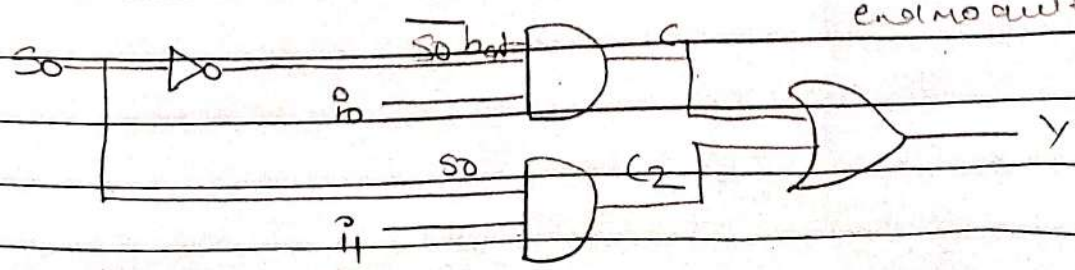
3. 2:1 MUX



I/P	O/P
S0	Y
0	i0
1	i1

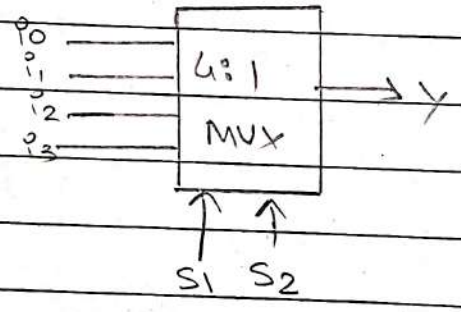
$Y = \bar{S_0} i_0 + S_0 i_1$

```
module mux
module MUX_21 (Y, i0, i1
output Y;
input S0, i1, i0;
wire c1, c2;
not n1 (Sobar, S0);
and a1 (c1, Sobar, i0);
and a2 (c2, S0, i1);
or a3 (Y, c1, c2);
endmodule.
```



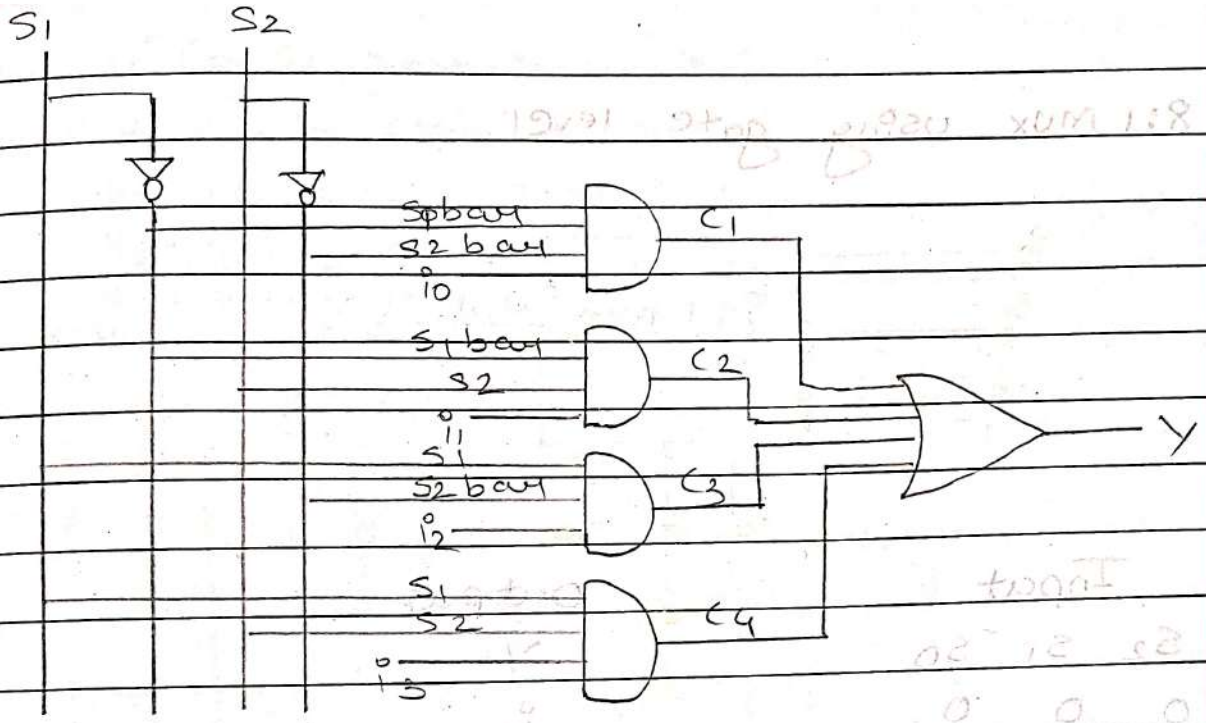
```
module    Mux-21 (i0, i1, s0, Y);
output    Y;
input     i0, i1, s0;
wire      sbar, c1, c2;
not       n1 (sbar, s0);
and       a1 (c1, sbar, i0);
and       a2 (c2, i1, s0);
or        a3 (Y, c1, c2);
endmodule
```

4. 4:1 mux



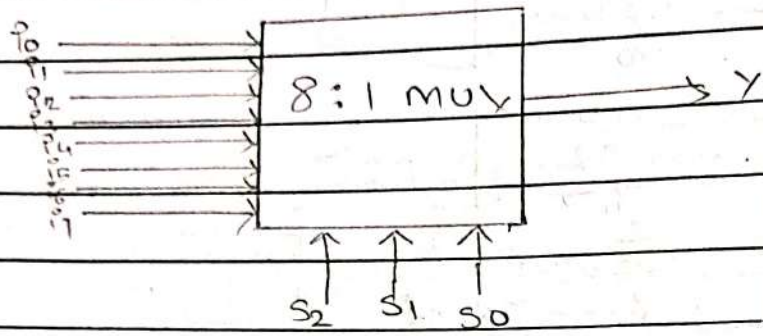
S_1	S_2	Y
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

$$Y = \bar{S_1}\bar{S_2}i_0 + \bar{S_1}S_2i_1 + S_1\bar{S_2}i_2 + S_1S_2i_3$$




```
module    mux-41 (Y, S1, S2, i0, i1, i2, i3);
output    Y;
input     i0, i1, i2, i3, S1, S2;
wire      S2bar, S1bar, C1, C2, C3, C4;
not       n1 (S1bar, S1);
not       n2 (S2bar, S2);
and       a1 (C1, S1bar, S2bar, i0);
and       a2 (C2, S1bar, S2, i1);
and       a3 (C3, S1, S2bar, i2);
and       a4 (C4, S1, S2, i3);
or        o5 (Y, C1, C2, C3, C4);
endmodule
```

5. 8:1 mux using gate level.

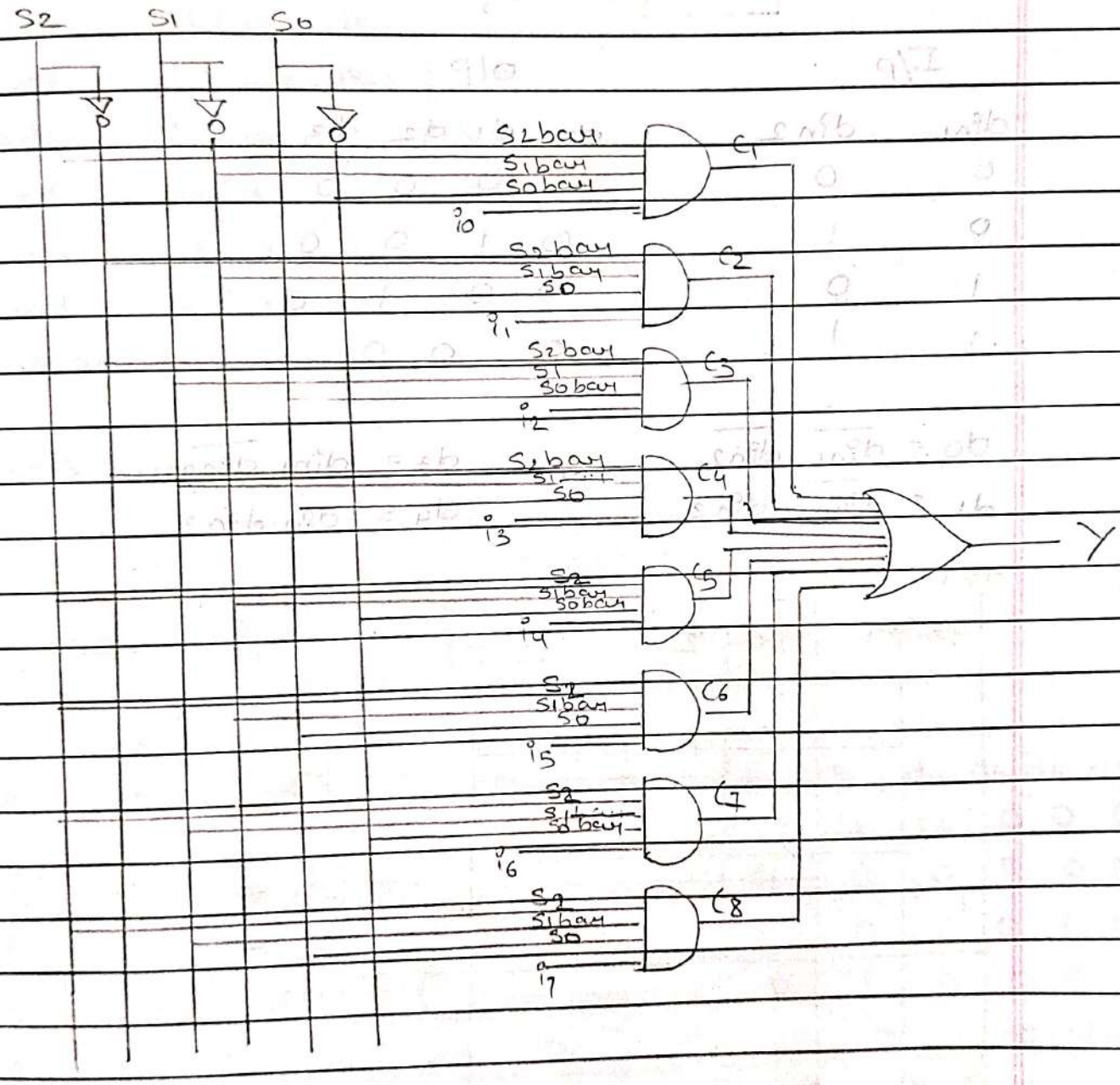


Input			output
S_2	S_1	S_0	Y
0	0	0	i_0
0	0	1	i_1
0	1	0	i_2
0	1	1	i_3
1	0	0	i_4
1	0	1	i_5
1	1	0	i_6
1	1	1	i_7

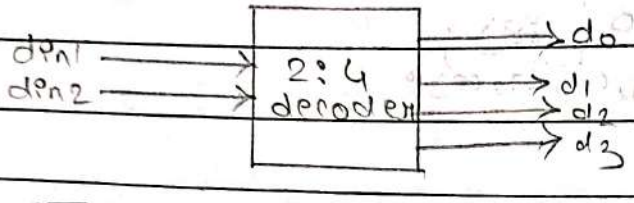
$Y = \bar{S_2}\bar{S_1}\bar{S_0} i_0 + \bar{S_2}\bar{S_1} S_0 i_1 + \bar{S_2} S_1 \bar{S_0} i_2 + \bar{S_2} S_1 S_0 i_3 + S_2 \bar{S_1} \bar{S_0} i_4 + S_2 \bar{S_1} S_0 i_5 + S_2 S_1 \bar{S_0} i_6 + S_2 S_1 S_0 i_7.$

```
module   mux_81 (y, i0,i1,i2,i3,i4,i5,i6,i7, s0,s1,s2);
output   y;
input    i0,i1,i2,i3,i4,i5,i6,i7, s0,s1,s2;
wire     s2bar,c1,c2,c3,c4; s1bar,s0bar; c5,c6,c7,c8;
not      n1 (s2bar,s2);
not      n2 (s1bar,s1);
not      n3 (s0bar,s0);
and      a1 (c1,s2bar,s1bar,s0bar,i0);
and      a2 (c2,s2bar,s1bar,s0,i1);
and      a3 (c3,s2bar,s1,s0bar,i2);
and      a4 (c4,s2bar,s1,s0,
```

```
and  g5 (c5, s2, s1bar, s0bar, i5);
and  g6 (c6, s2, s1bar, s0, i6);
and  g7 (c7, s2, s1, s0bar, i7);
and  g8 (c8, s2, s1, s0, i8);
OR   g9 (Y, c1, c2, c3, c4, c5, c6, c7, c8);
endmodule;
```



6. 2:4 decoder



I/P

d_{in1}	d_{in2}
0	0
0	1
1	0
1	1

O/P

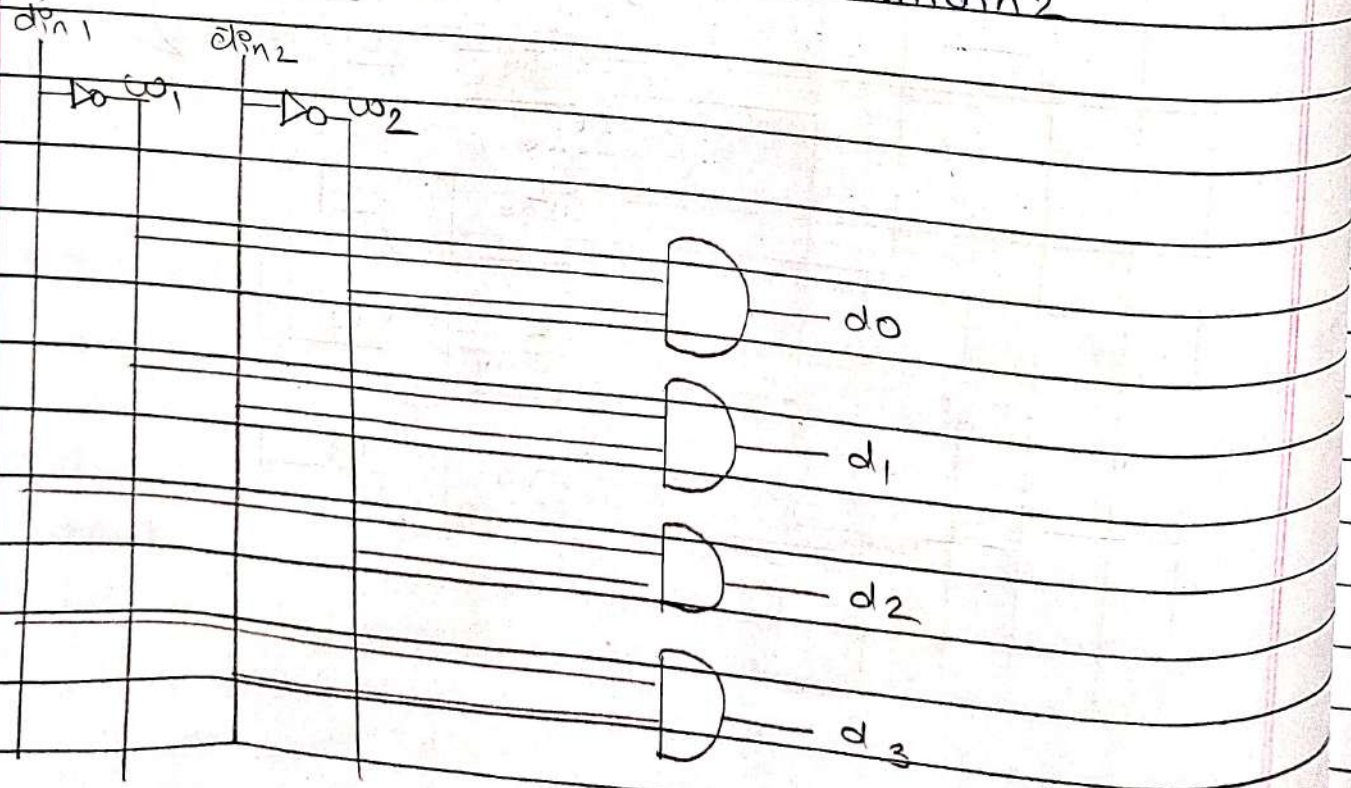
d_0	d_1	d_2	d_3
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$d_0 = \overline{d_{in1}}\ \overline{d_{in2}}$

$d_1 = \overline{d_{in1}}\ d_{in2}$

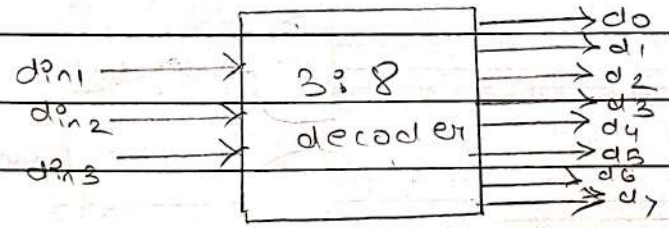
$d_3 = d_{in1}\ \overline{d_{in2}}$

$d_4 = d_{in1} d_{in2}$



6
```
module dec24(d0, d1, d2, d3, din1, din2);
output d0, d1, d2, d3;
input din1, din2;
wire w1, w2;
not n1(w1, din1);
not n2(w2, din2);
and a1(d0, w1, w2);
and a2(d1, w1, din2);
and a3(d2, din1, w2);
and a4(d3, din1, din2);
endmodule
```

7. 3:8 decoder.



I/P			O/P							
din1	din2	din3	d0	d1	d2	d3	d4	d5	d6	d7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$d_0 = \overline{din_1}\ \overline{din_2}\ \overline{din_3}$

$d_1 = \overline{din_1}\ \overline{din_2}\ din_3$

$d_2 = \overline{din_1}\ din_2\ \overline{din_3}$

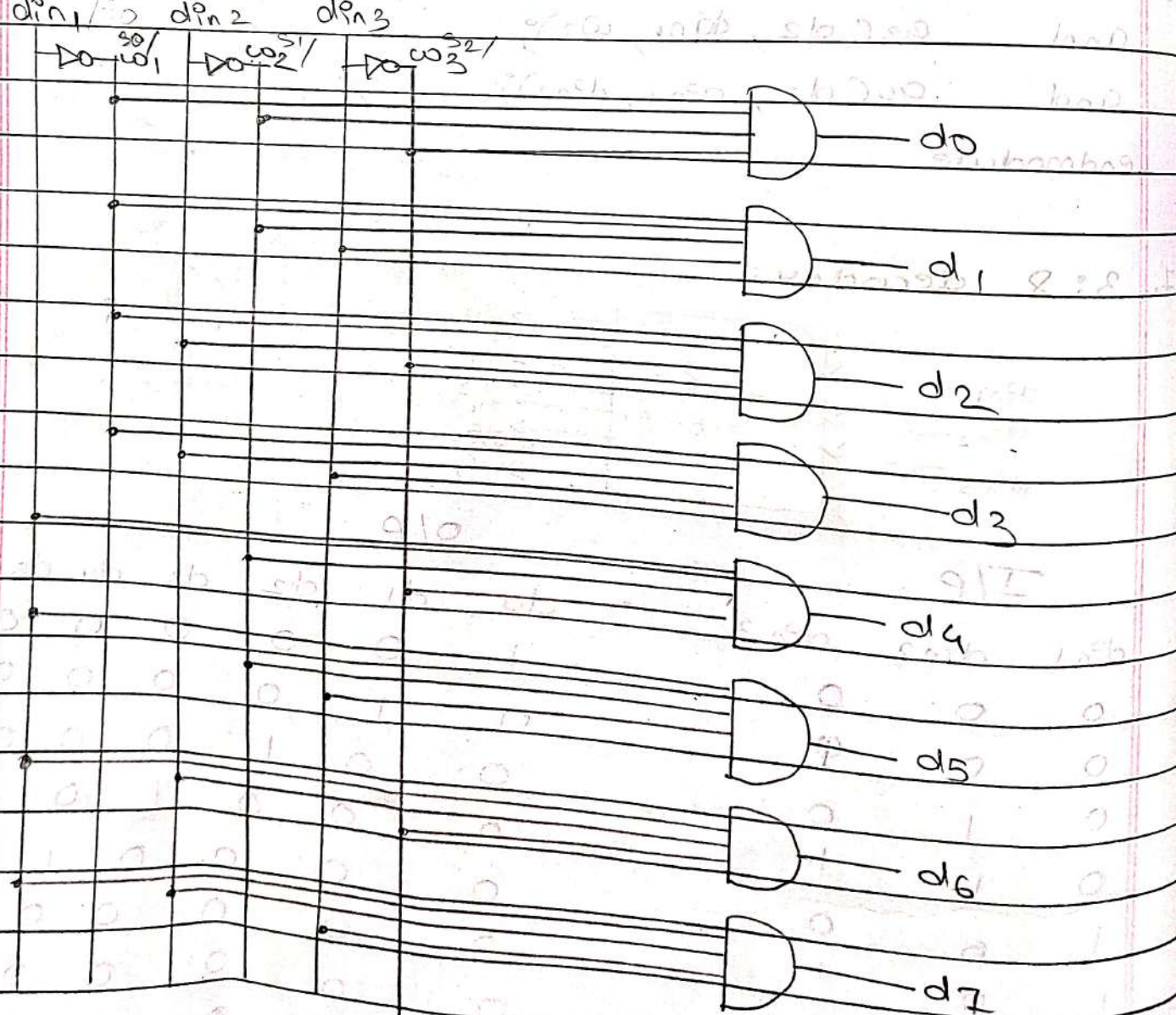
$d_3 = \overline{din_1}\ din_2\ din_3$

$d_4 = din_1\ \overline{din_2}\ \overline{din_3}$

$d_5 = din_1\ \overline{din_2}\ din_3$

$d_6 = din_1\ din_2\ \overline{din_3}$

$d_7 = din_1\ din_2\ din_3$



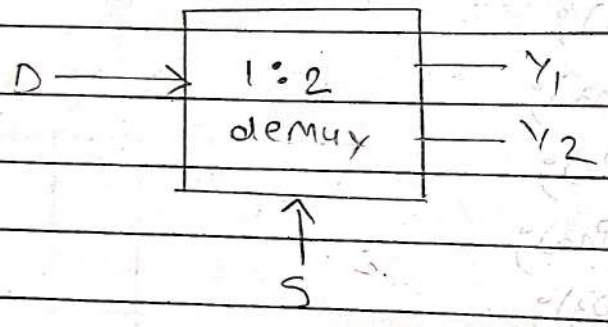
```
module dec-38 (d0, d1, d2, d3, d4, d5, d6, d7, din1, din2, din3);
output d0, d1, d2, d3, d4, d5, d6, d7;
input din1, din2, din3;
wire w1, w2, w3;
not n1 (w1, din1);
not n2 (w2, din2);
not n3 (w3, din3);
and a1 (d0, w1, w2, w3);
and a2 (d1, w1, w2, din3);
and a3 (d2, w1, din2, w3);
and a4 (d3, w1, din2, din3);
and a5 (d4, din1, w2, w3);
and a6 (d5, din1, w2, din3);
and a7 (d6, din1, din2, w3);
and a8 (d7, din, din2, din3);
endmodule
```

Vector code

```
module dec-38 (d, s);
output [7:0] d;
input [2:0] s;
not n1 (w[0], s[0]);
not n2 (w[2], s[1]);
not n3 (w[3], s[3]);
and a1 (d[0], w[0], w[1], w[2]);
and a2 (d[1], w[0], w[1], s[2]);
and a3 (d[2], w[0], s[1], w[2]);
and a4 (d[3], w[0], s[1], s[2]);
```

```
and g5 (d[4], s[0], w[1], w[0]);
and g6 (d[5], s[0], w[1], s[2]);
and g7 (d[6], s[0], s[1], w[2]);
and g8 (d[7], s[0], s[1], s[2]);
endmodule
```

8. 1:2 demux [using Gatelevel]



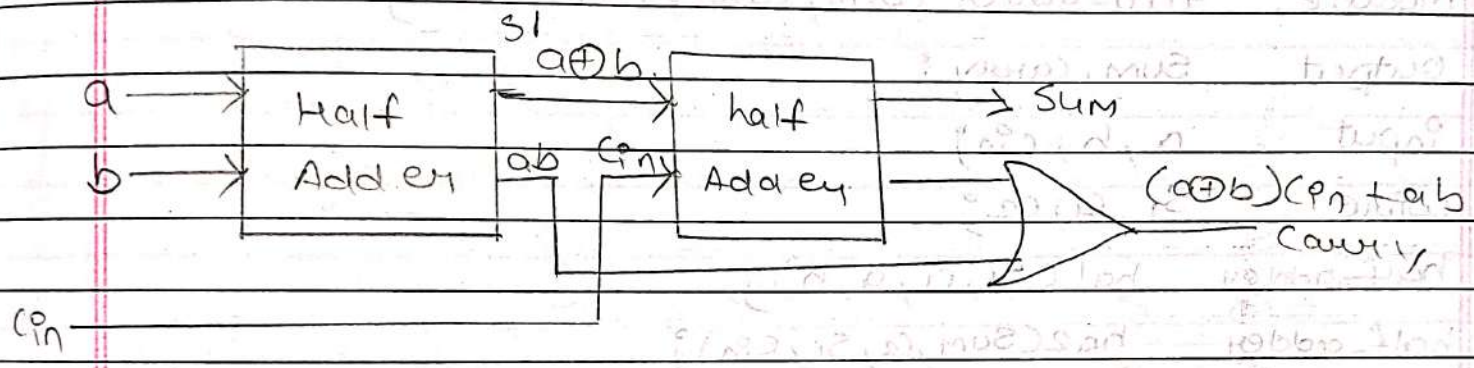
Input		output	
S	D	Y_2	Y_1
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

* Structural level Modelling:-

1. Full adder using halfadder

$Sum = a \oplus b \oplus c_{in}$

$Carry = ab + bc_{in} + ac_{in}$



$Carry = (a \oplus b)\, c_{in} + ab$

$= ab + (\bar{a}b + a\bar{b})\, c_{in}$

$= b(a + c_{in}\bar{a}) + c_{in}\, a\bar{b}$

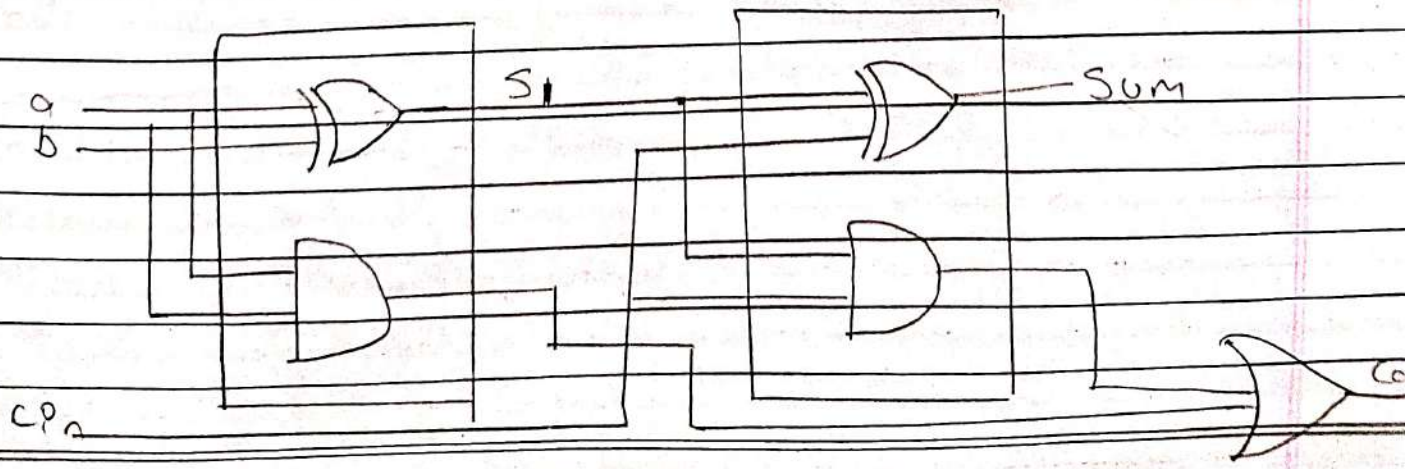
$x + \bar{x}y = x + y$

$= b(a + c_{in}) + c_{in}\, a\bar{b}$

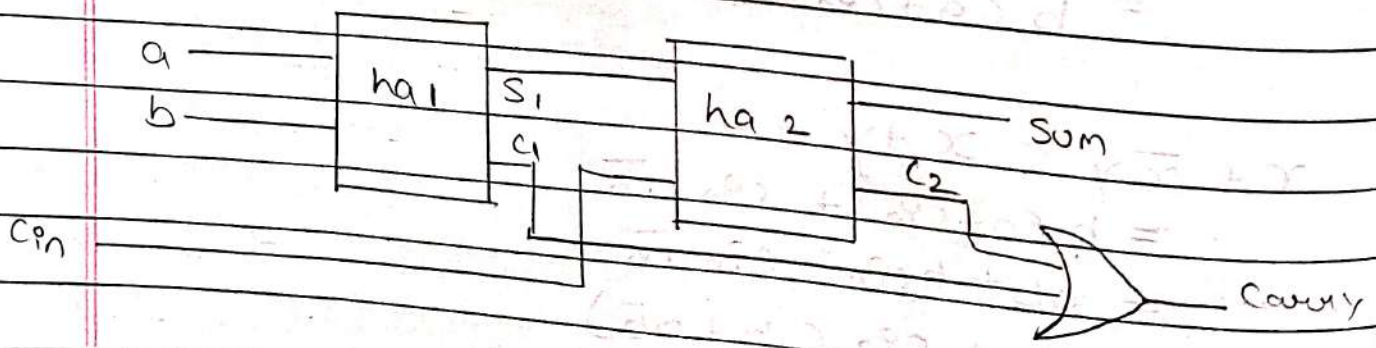
$= ab + bc_{in} + c_{in}a\bar{b}$

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

$= ab + c_{in}a + c_{in}b$

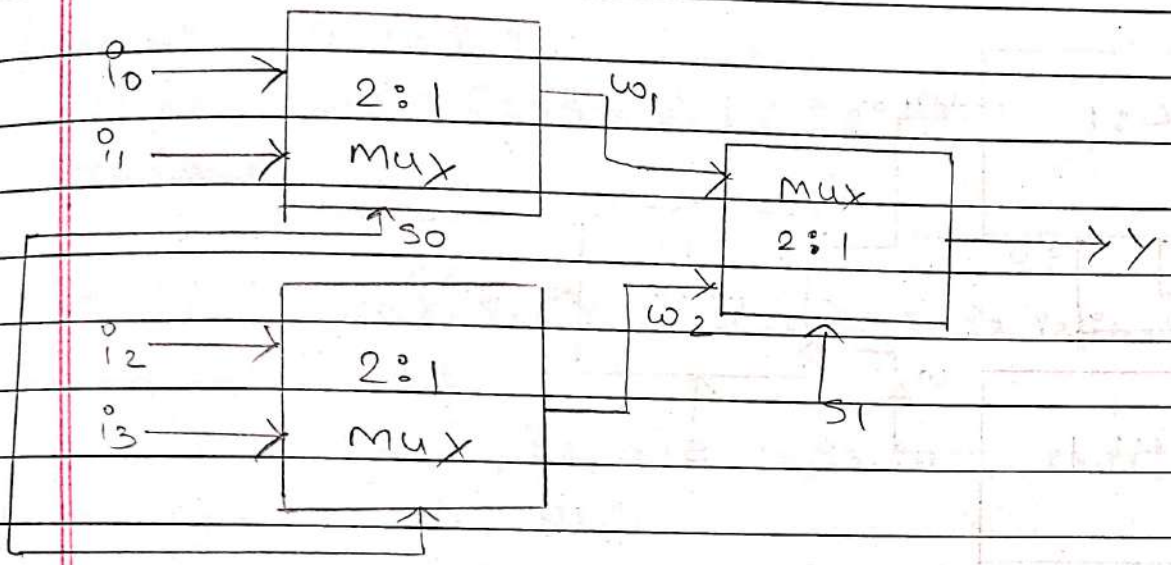


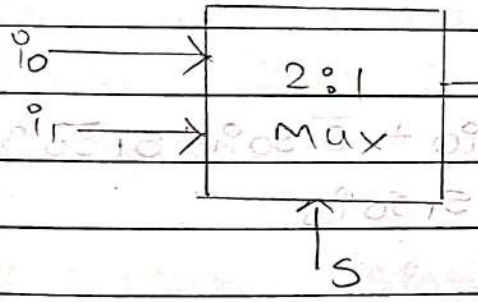
```
module half_adder (sum, carry, a, b);
output sum, carry;
input a, b;
assign sum = a^b;
assign carry = a & b;
endmodule

module full_adder (sum, carry, a, b, cin);
output sum, carry;
input a, b, cin;
wire s1, c1, c2;
half_adder ha1 (s1, c1, a, b);
half_adder ha2 (sum, c2, s1, cin);
or (carry, c1, c2);
endmodule
```



PAGE NO.:
DATE : / /

2. 4:1 mux using 2:1 mux



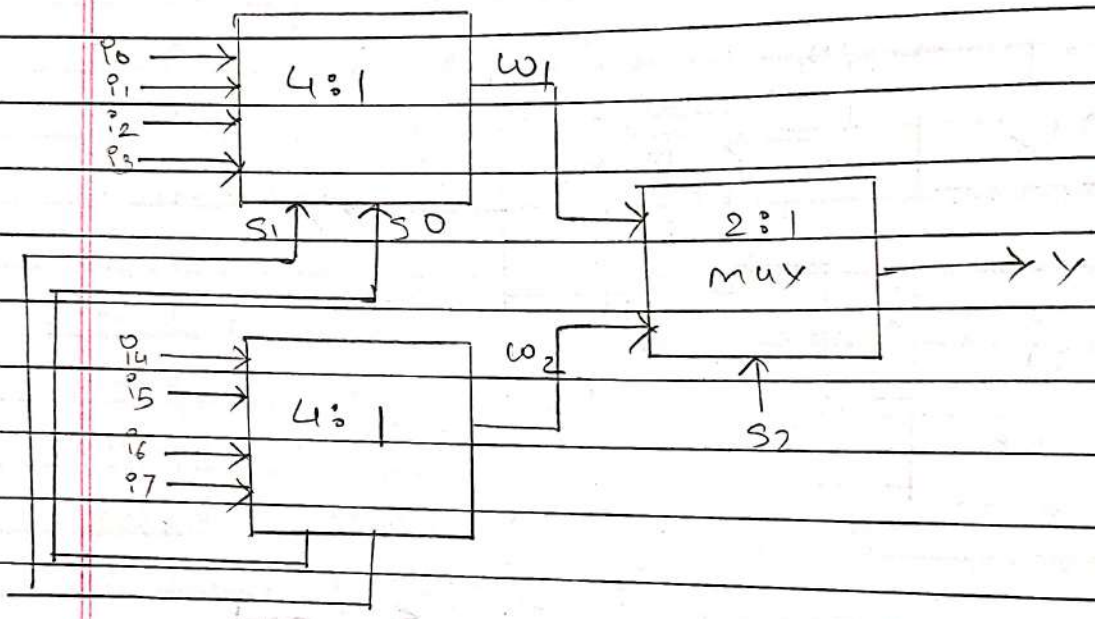


S	out
0	i0
1	i1

$out = i_0 \bar{S} + i_1 S$

```
Module    mux-21 (out, S, i0, i1);
output    out;
input     S, i0, i1;
assign    out = ((~S) & i0) | (S & i1);
endmodule

module    mux-41 (Y, S0, S1, i0, i1, i2, i3);
output    Y;
input     S0, S1, i0, i1, i2, i3;
wire      w1, w2;
mux-21    M1 (w1, S0, i0, i1);
mux-21    M2 (w2, S0, i2, i3);
mux-21    M3 (Y, S1, w1, w2);
endmodule
```

3. 8:1 mux using 4:1 mux and 2:1 mux



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

$Y = \bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$

```
module mux_4l (Y, S1, S0, i0, i1, i2, i3);
output Y;
input S1, S0, i0, i1, i2, i3;
assign Y=((~S1)&(~S0)&i0) | ((~S1)&S0&i1) | (S1&(~S0)&
       | (S1&S0&i3);
endmodule
```

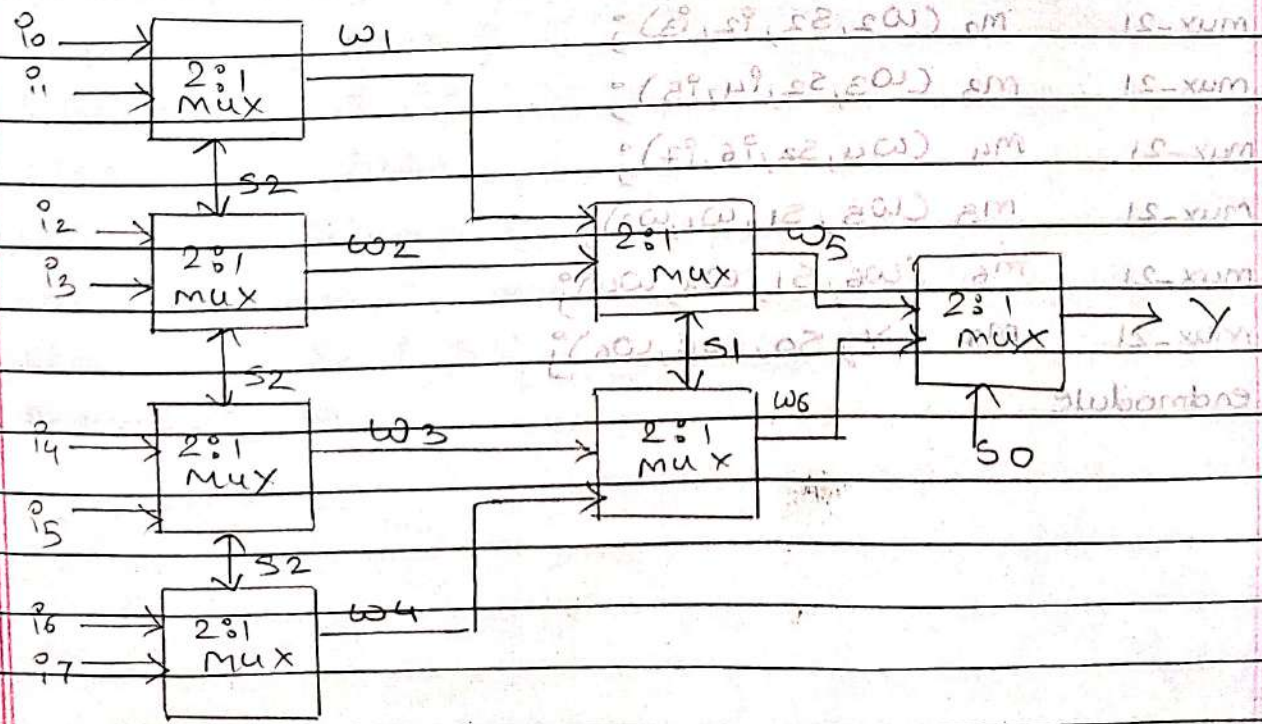
S	out
0	I_0
1	I_1

$out = \bar{S}I_0 + SI_1$

```
module mux-21 (out, s, i0, i1);
output out;
input s, i0, i1;
assign out = ((~s & i0) | (s & i1));
endmodule
```

```
Module mux-81 (Y, S0, S1, S2, i0, i1, i2, i3, i4, i5, i6, i7);
output Y;
input S0, S1, S2, i0, i1, i2, i3, i4, i5, i6, i7;
wire w1, w2;
mux-41 M1 (w1, i0, i1, i2, i3, S1, S0);
mux-41 M2 (w2, i4, i5, i6, i7, S1, S0);
mux-21 M3 (Y, S2, w1, w2);
endmodule
```

4. 8:1 mux using 2:1 mux.



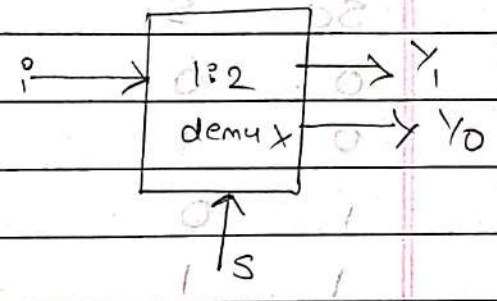
S_0	Y
0	i_0
1	i_1

$Y = \bar{S_0}\, i_0 + S_0 i_1$

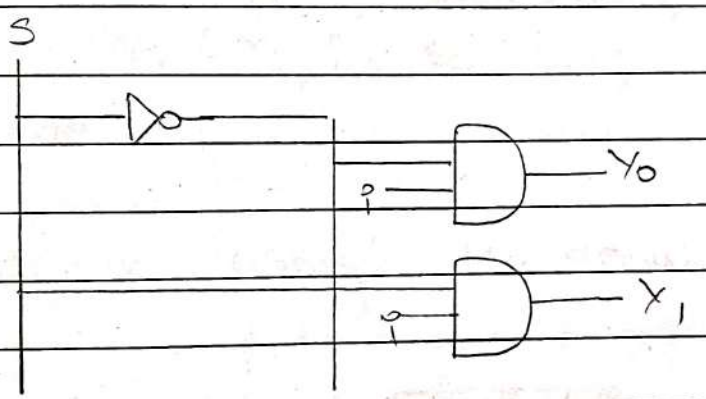
```
module mux-21 (Y, S0, i0, i1);
output Y;
input S0, i0, i1;
assign Y = ((~S0) & i0) | (S0 & i1));
endmodule

module mux-81 (Y, S0, S1, S2, i0, i1, i2, i3, i4, i5, i6, i7);
output Y;
input S0, S1, S2, i0, i1, i2, i3, i4, i5, i6, i7;
wire w1, w2, w3, w4, w5, w6;
mux-21 m1 (w1, S2, i0, i1);
mux-21 m2 (w2, S2, i2, i3);
mux-21 m3 (w3, S2, i4, i5);
mux-21 m4 (w4, S2, i6, i7);
mux-21 m5 (w5, S1, w1, w2);
mux-21 m6 (w6, S1, w3, w4);
mux-21 m7 (Y, S0, w5, w6);
endmodule.
```

5. 1:2 demux using gate level :-

I/P	O/P	
S	Y_1	Y_0
0	0	i
1	i	0



$Y_0 = \bar{S}i$, $Y_1 = Si$




```
module demux-12 (Y0, Y1, S, i);
output Y0, Y1;
input S, i;
wire Sbar;
not (S.bar, S);
and (Y0, i, Sbar);
and (Y1, i, S);
endmodule
```

6. 1:4 demux using gate level:

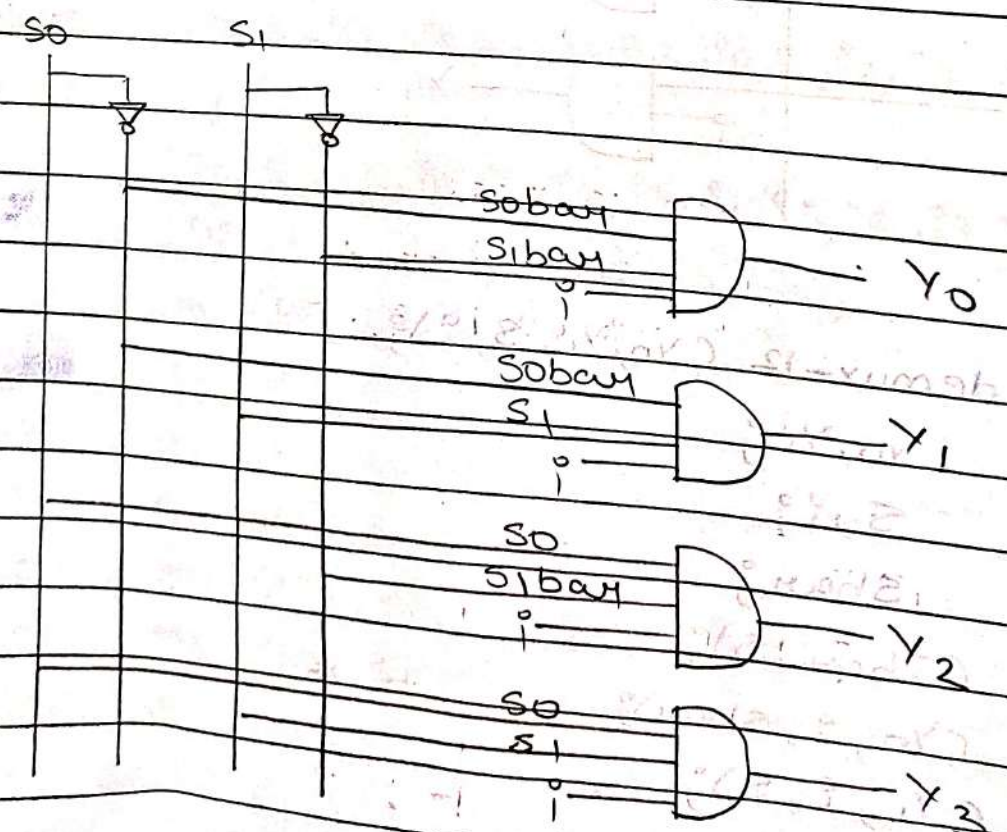
S_0	S_1	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$Y_3 = S_0 S_1$

$Y_2 = S_0 \overline{S_1}$

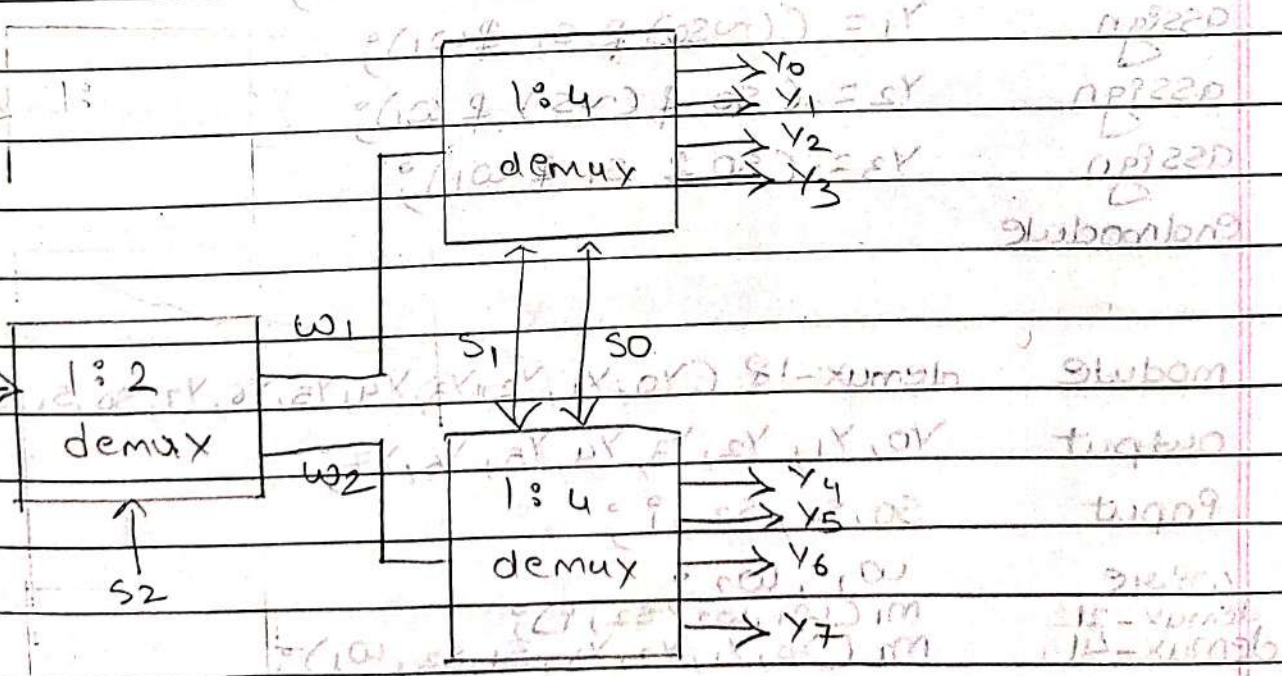
$Y_1 = \overline{S_0} S_1$

$Y_0 = \overline{S_0}\,\overline{S_1}$



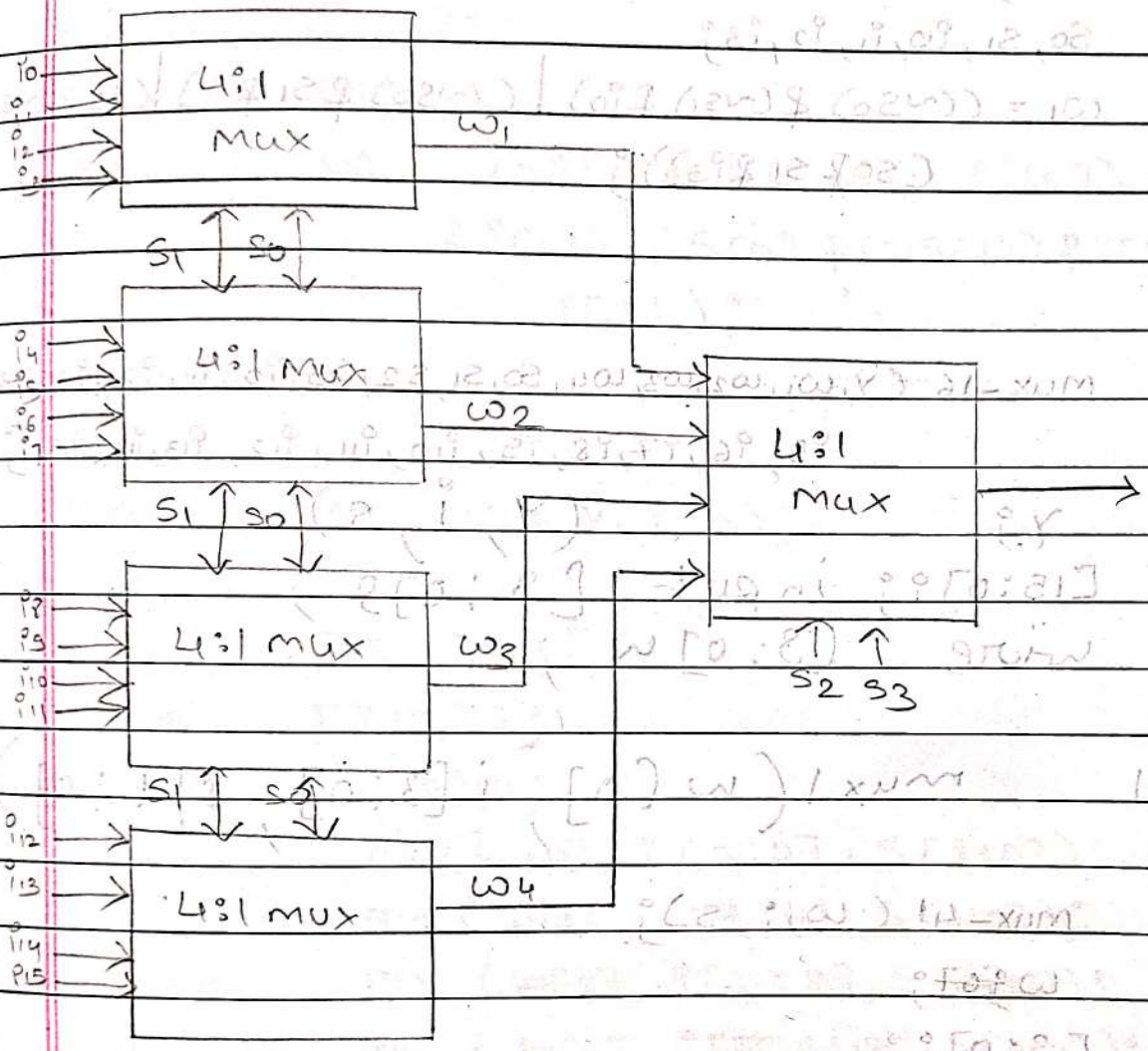
```
module    demux_14 (Y0, Y1, Y2, Y3, S0, S1, i);
output    Y0, Y1, Y2, Y3;
input     S0, S1, i;
wire      S0bar, S1bar;
not       n1 (S0bar, S0);
not       n2 (S1bar, S1);
and       a1 (Y0, i, S0bar, S1bar);
and       a2 (Y1, i, S0bar, S1);
and       a3 (Y2, i, S0, S1bar);
and       a4 (Y3, i, S0, S1);
endmodule
```

7. 1:8 demux using 1:4 demux and 1:2 demux.



```
module demux-12 (w1, w2, s2, i);
output w1, w2;
input s2, i;
assign w1 = ((~s2) & i);
assign w2 = s2 & i;
endmodule

module demux-14 (y0, y1, y2, y3, y4, y5, y6, y7, s0, s1, w1, w2
output

module demux-14 (y0, y1, y2, y3, s0, s1, w1);
output y0, y1, y2, y3;
input s0, s1, w1;
assign y0 = ((~s0) & (~s1) & w1);
assign y1 = ((~s0) & s1 & w1);
assign y2 = (s0 & (~s1) & w1);
assign y3 = (s0 & s1 & w1);
endmodule

module demux-18 (y0, y1, y2, y3, y4, y5, y6, y7, s0, s1, s2, i);
output y0, y1, y2, y3, y4, y5, y6, y7;
Input s0, s1, s2, i;
wire w1, w2;
demux-212 m1 (w1, w2, s2, i);
demux-414 m2 (y0, y1, y2, y3, s1, s2, w1);
demux-414 m3 (y4, y5, y6, y7, s1, s2, w2);
endmodule
demux
```

8. 16:1 mux using 4:1 mux



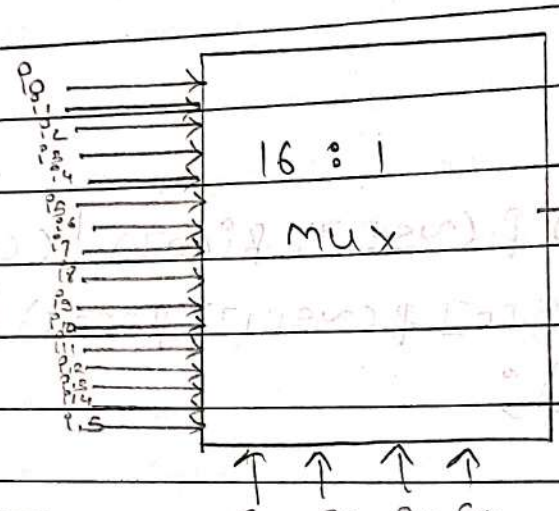
S0	S1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \overline{S_0}\,\overline{S_1} I_0 + \overline{S_0} S_1 I_1 + S_0 \overline{S_1} I_2 + S_0 S_1 I_3$$

```
module mux-41 (w1, s0, s1, i0, i1, i2, i3);
output w1;
input s0, s1, i0, i1, i2, i3;
assign w1 = ((~s0) & (~s1) & i0) | ((~s0) & s1 & i1) | (s0 &
            (s0 & s1 & i3));
endmodule

module mux-16 (y, w1, w2, w3, w4, s0, s1, s2, s3, i0, i1, i2, i3,
               i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15);
output y;                    (y, i, s)
input [15:0] i; input [3:0] s;
assign wire [3:0] w;

mux41  mux1(w[0], i[3:0], s[1:0]

module mux-41 (w, i, s);
output w;
input [3:0] i;
input [1:0] s;
assign w = i[3:0], s[1:0], ((~s[0]) & (~s[
                                  i[0]
endmodule
```

```
module mux-41 (w, i, s);
output w;
input [3:0] i;
input [1:0] s;
assign w = ((~s[0]) & (~s[1]) & i[0]) | ((~s[0])
        & i[1]) | (s[0] & (~s[1]) & i[2]) | (s[0] &
        i[3]);
endmodule

module mux-16 (y, i, s);
output y;
input [15:0] i;
input [3:0] s;
wire [3:0] w;
mux-41 m1 (w[0], i[3:0], s[1:0]);
mux-41 m2 (w[1], i[7:4], s[1:0]);
mux-41 m3 (w[2], i[11:8], s[1:0]);
mux-41 m4 (w[3], i[15:12], s[1:0]);
mux-41 m5 (y, w[3:0], s[3:2]);
endmodule
```

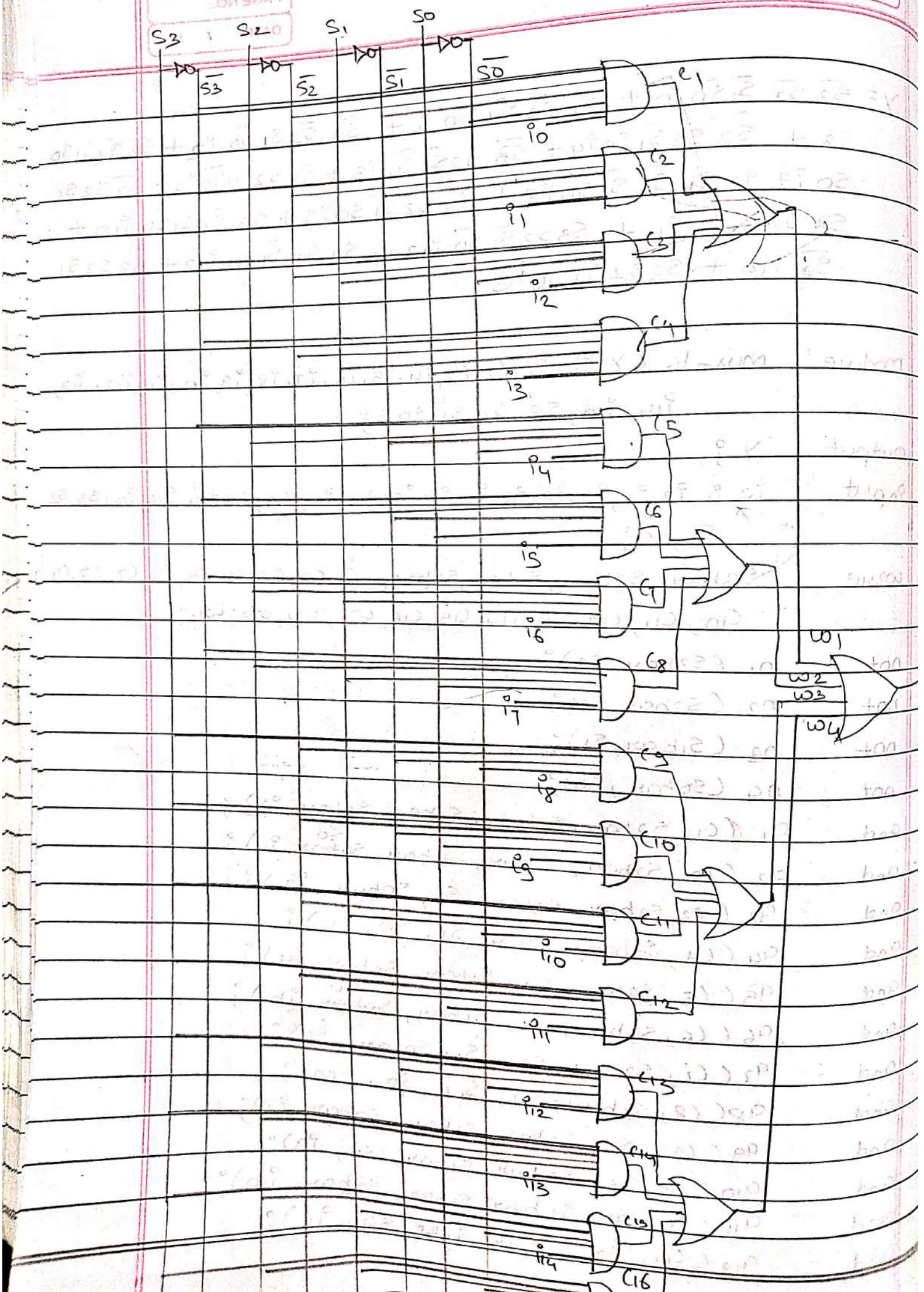
9. 16:1 mux using gate level.



Input				output
S_3	S_2	S_1	S_0	Y
0	0	0	0	i_0
0	0	0	1	i_1
0	0	1	0	i_2
0	0	1	1	i_3
0	1	0	0	i_4
0	1	0	1	i_5
0	1	1	0	i_6
0	1	1	1	i_7
1	0	0	0	i_8
1	0	0	1	i_9
1	0	1	0	i_{10}
1	0	1	1	i_{11}
1	1	0	0	i_{12}
1	1	0	1	i_{13}
1	1	1	0	i_{14}
1	1	1	1	

$Y = \bar{S_3}\bar{S_2}\bar{S_1}\bar{S_0}i_0 + \bar{S_3}\bar{S_2}\bar{S_1}S_0 i_1 + \bar{S_3}\bar{S_2}S_1\bar{S_0}i_2 + \bar{S_3}\bar{S_2}S_1S_0 i_3 + \bar{S_3}S_2\bar{S_1}\bar{S_0}i_4 + \bar{S_3}S_2\bar{S_1}S_0 i_5 + \bar{S_3}S_2S_1\bar{S_0}i_6 + \bar{S_3}S_2S_1S_0 i_7 + S_3\bar{S_2}\bar{S_1}\bar{S_0}i_8 + S_3\bar{S_2}\bar{S_1}S_0 i_9 + S_3\bar{S_2}S_1\bar{S_0}i_{10} + S_3\bar{S_2}S_1S_0 i_{11} + S_3S_2\bar{S_1}\bar{S_0}i_{12} + S_3S_2\bar{S_1}S_0 i_{13} + S_3S_2S_1\bar{S_0}i_{14} + S_3S_2S_1S_0 i_{15}$

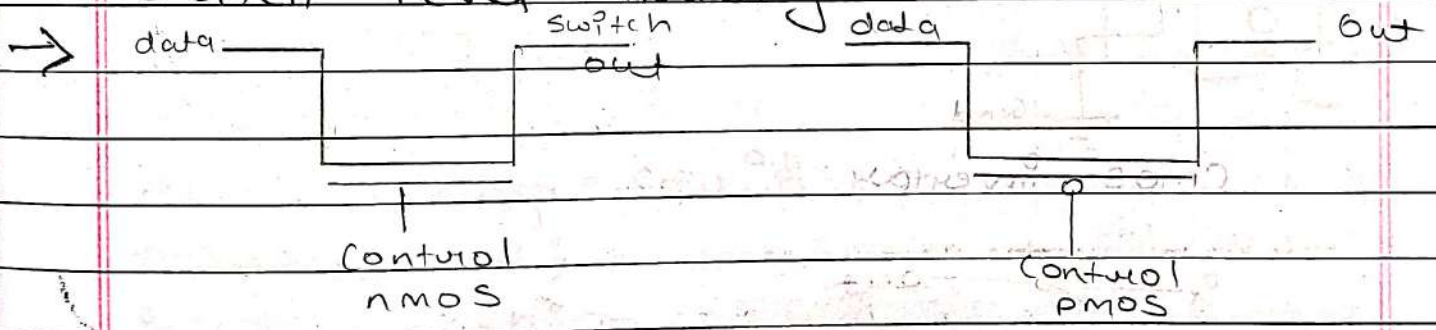
```
module  MUX-16 (Y, i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13,
                i14, i15, S3, S2, S1, S0);
output  Y;
input   i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15, S3, S2
        S1, S0;
wire    S3bar, S2bar, S1bar, S0bar, C1, C2, C3, C4, C5, C6, C7, C8, C9
        C10, C11, C12, C13, C14, C15, C16, W1, W2, W3, W4;
not     n1 (S3bar, S3);
not     n2 (S2bar, S2);
not     n3 (S1bar, S1);
not     n4 (S0bar, S0);
and     a1 (C1, S3bar, S2bar, S1bar, S0bar, i0);
and     a2 (C2, S3bar, S2bar, S1bar, S0, i1);
and     a3 (C3, S3bar, S2bar, S1, S0bar, i2);
and     a4 (C4, S3bar, S2bar, S1, S0, i3);
and     a5 (C5, S3bar, S2, S1bar, S0bar, i4);
and     a6 (C6, S3bar, S2, S1bar, S0, i5);
and     a7 (C7, S3bar, S2, S1, S0bar, i6);
and     a8 (C8, S3bar, S2, S1, S0, i7);
and     a9 (C9, S3, S2bar, S1bar, S0bar, i8);
and     a10 (C10, S3, S2bar, S1bar, S0, i9);
and     a11 (C11, S3, S2bar, S1, S0bar, i10);
and     a12 (C12, S3, S2bar, S1, S0, i11);
```


S3
S2
S1
S0
$\overline{S3}$
$\overline{S2}$
$\overline{S1}$
$\overline{S0}$
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
I14
W1
W2
W3
W4

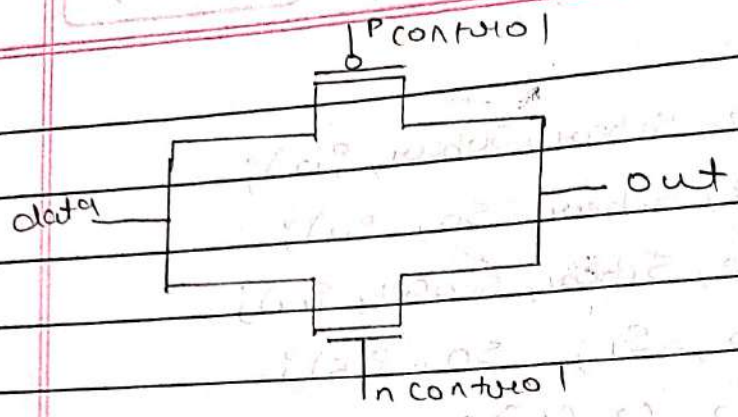
and g13 (c13, s3, s2, s1bar, s0bar, i12);
and g14 (c14, s3, s2, s1bar, s0, i13);
and g15 (c15, s3, s2, s1bar, s0bar, i14);
and g16 (c16, s3, s2, s1, s0, i15);
or g17 (w1, c1, c2, c3, c4);
or g18 (w2, c5, c6, c7, c8);
or g19 (w3, c9, c10, c11, c12);
or g20 (w4, c13, c14, c15, c16);
or g21 (Y, w1, w2, w3, w4);
endmodule.

* Write a verilog code for CMOS inverter using switch level modeling.

→

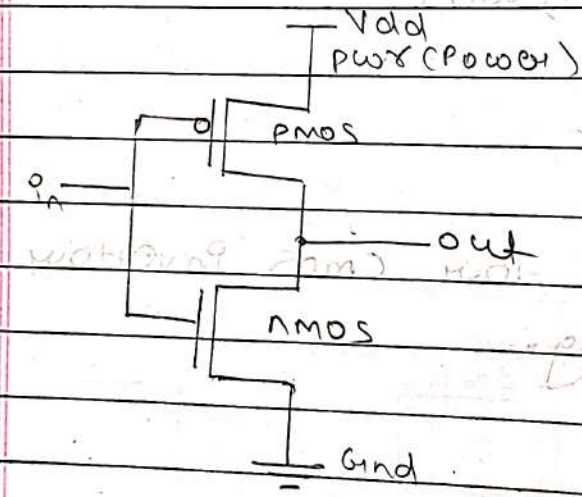


nmos n1
nmos (out, data, control);
nmos (out, data, control);
pmos p1 (out, data, control);
pmos (out, data, control);

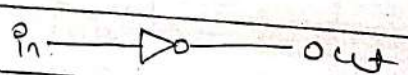


Cmos switch, nmos and pmos in parallel.

cmos c1 (out, data, ncontrol, pcontrol);
cmos out (out, data, ncontrol, pcontrol);



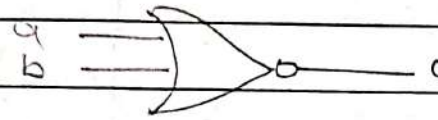
Cmos Inverter




```
module My_not (out, in);
output out;
input in;
// declare power, gnd
supply1 pwr;
supply0 gnd;
nmos (out, gnd, in);
pmos (out, pwr, in);
endmodule
```

* Write a verilog code for CMOS 2-input NOR gate using switch level modeling.

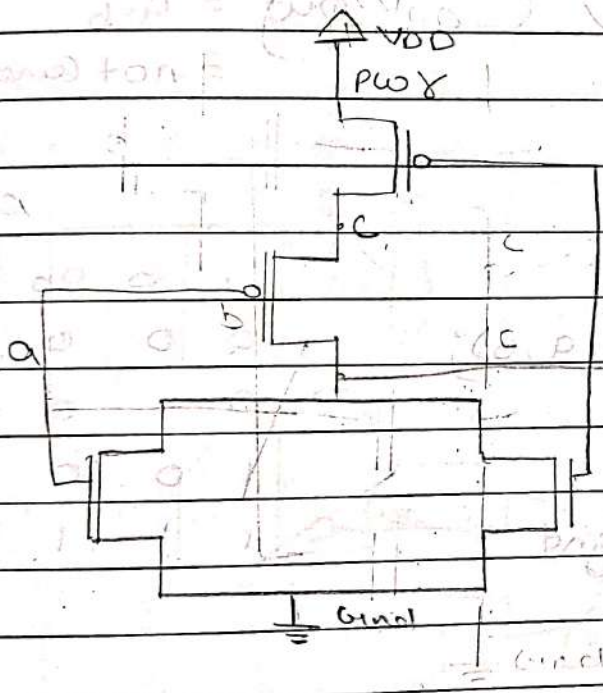
→



$out = \overline{a+b}$

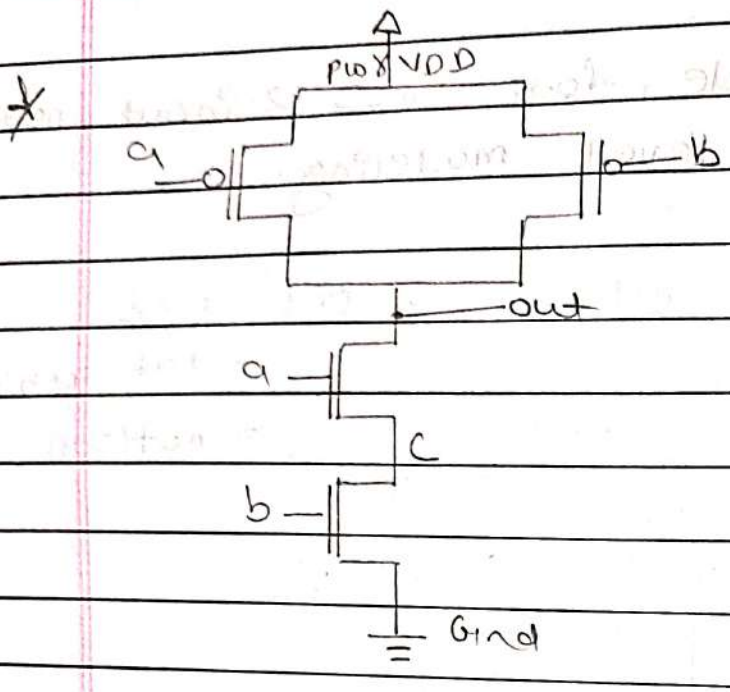
$= not(a+b)$

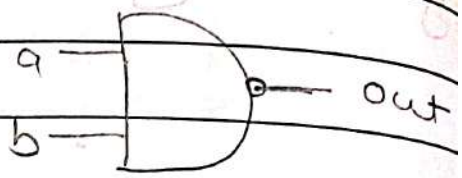
$= not(or)$



a	b	a+b	out
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1

```
module norgate (out, a, b);
output out;
input a, b;
// declare power, gnd
supply1 pwr;
supply0 gnd;
wire c;
nmos (out, gnd, a);
nmos (out, gnd, b);
pmos (c, pwr, b);
pmos (out, c, a);
endmodule
```





$out = \overline{a \cdot b}$

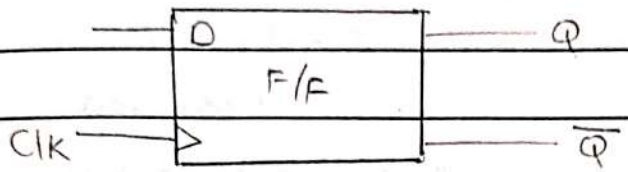
= not (and)

a	b	ab	o
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

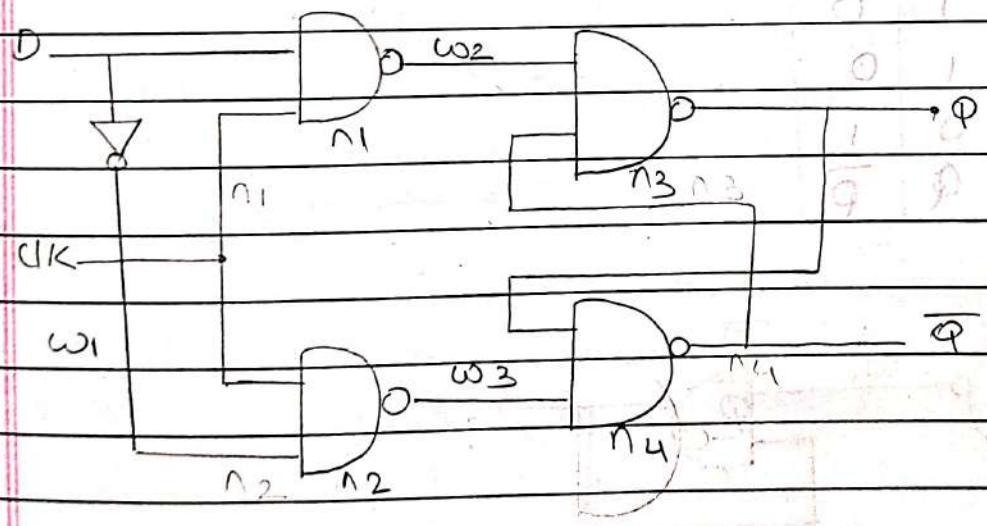
```
module nandgate (out, a, b);
output out;
input a, b;
  // declare power, gnd
supply1 pwr;
supply0 gnd;
wire c; out, c, a
nmos (c, gnd, b);
nmos (out, c, a);
pmos (out, pwr, b);
pmos (out, pwr, b);
endmodule
```

nMOS (c, gnd, b
nMOS (out, c, a
PMOS (pwr
PMOS (out, pwr, a

* D flipflop using nand gate:-



Clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0



Module d-flipflop (q, $\overline{q}$, d, clk);

output q, $\overline{q}$;

input d, clk;

wire w1, w2, w3;

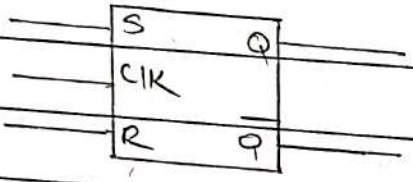
not n1 (w1, d);

nand q1 (w2, d, clk);

nand q2 (w3, d, clk);

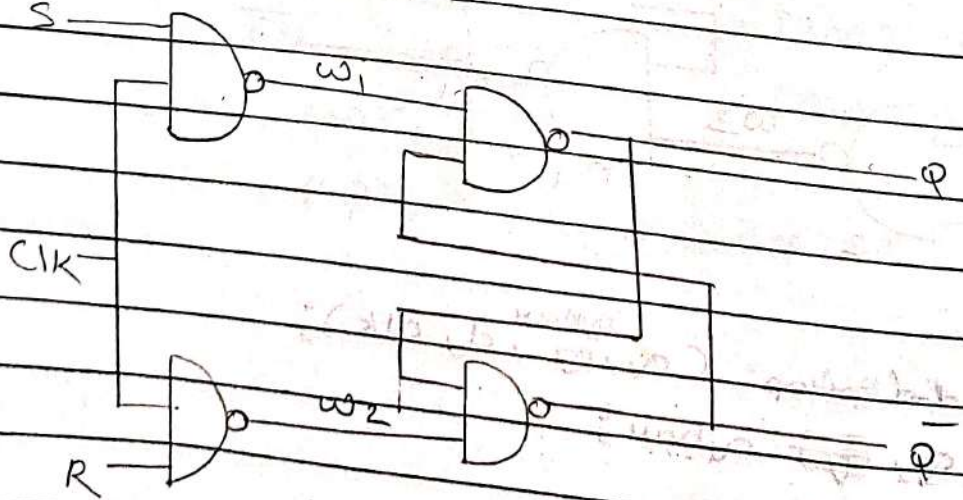
nand g3(q, w2, clk);
nand g4(qbar, w3, q);
endmodule

* SR Flipflop using nand gate:-



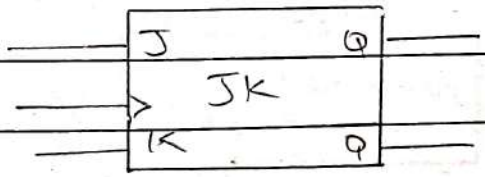
Module SR_ff(q,qbar,S,R,clk);

CLK	S	R	Q	$\bar{Q}$
+ve	0	0	1	D
+ve	0	1	1	0
+ve	1	0	0	1
+ve	1	1	Q	$\bar{Q}$

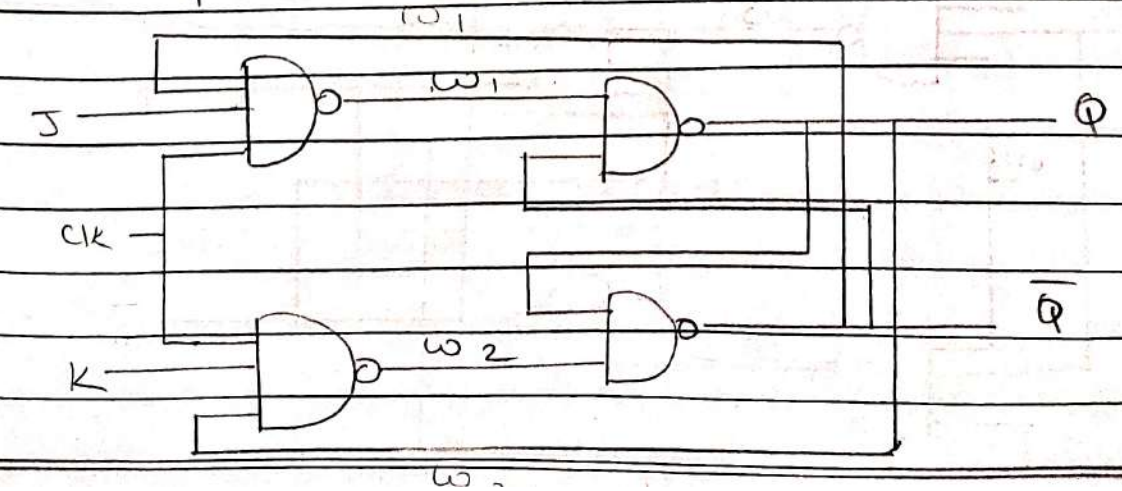


```
module sr-ff (q, qbar, s, r, clk);
output q, qbar;
input s, r, clk;
wire w1, w2;
nand n1 (w1, s, clk);
nand n2 (w2, r, clk);
nand n3 (q, w1, qbar);
nand n4 (qbar, w2, q);
endmodule
```

* JK Flipflop using nand gate:-

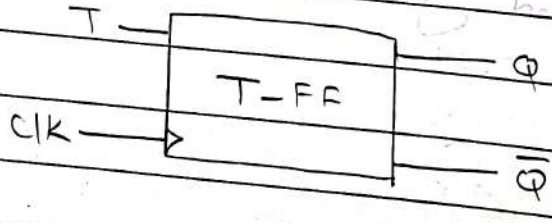


clk	J	K	Q	$\overline{Q}$
+ve	0	0	Q	$\overline{Q}$
	0	1	0	1
	1	0	1	0
	1	1	$\overline{Q}$	Q

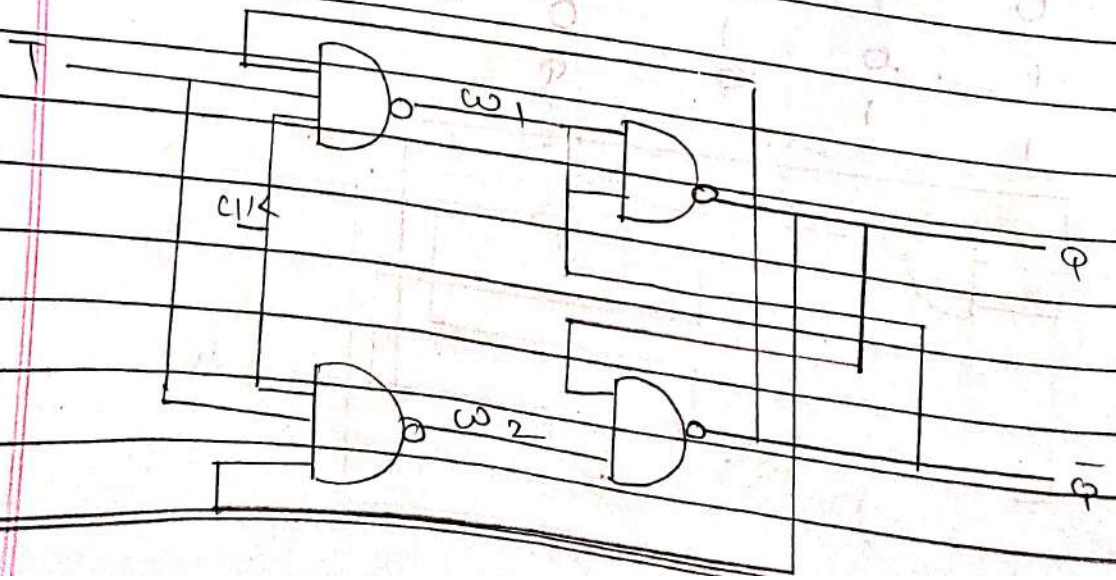


```
module jk_ff (q, qbar, j, k, clk);
output q, qbar;
input j, k, clk;
wire w1, w2;
nand n1 (w1, j, clk, qbar);
nand n2 (w2, k, q, clk);
nand n3 (q, w1, qbar, clk);
nand n4 (qbar, w2, q);
endmodule
```

* TFF using nand gate :-

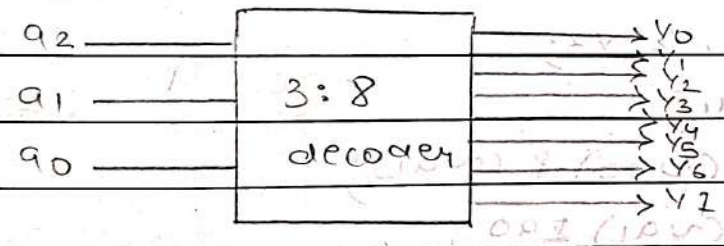


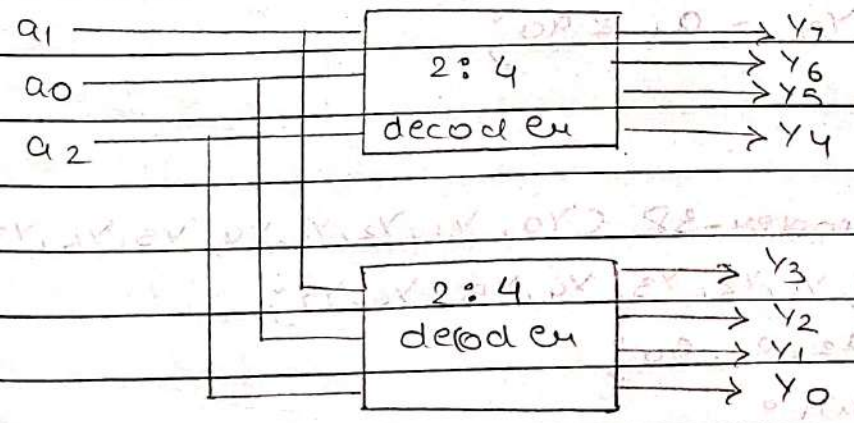
T	Q	$\overline{Q}$
0	0	1
1	0	1
0	1	0
1	1	0



```
module t_ff (q, qbar, t, clk);
output q, qbar;
input t, clk;
wire w1, w2;
nand n1 (w1, t, qbar, clk);
nand n2 (w2, t, q, clk);
nand n3 (q, w1, qbar);
nand n4 (qbar, w2, q);
endmodule
```

* 3:8 decoder using 2:4 decoder.





2:4 decoder

I/P		O/P			
a_1	a_0	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

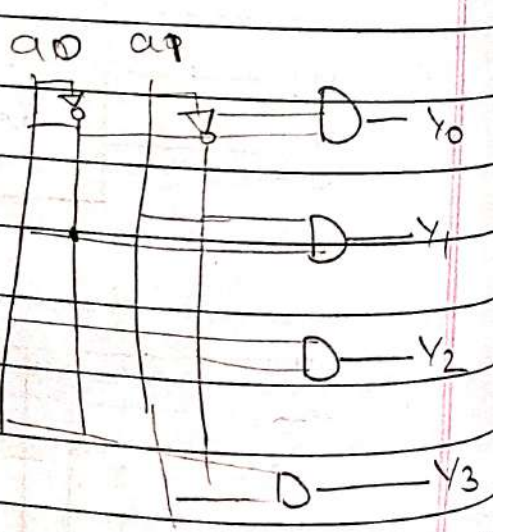
$Y_0 = \overline{a_1}\,\overline{a_0}$

$Y_1 = \overline{a_1}\,a_0$

$Y_2 = a_1\,\overline{a_0}$

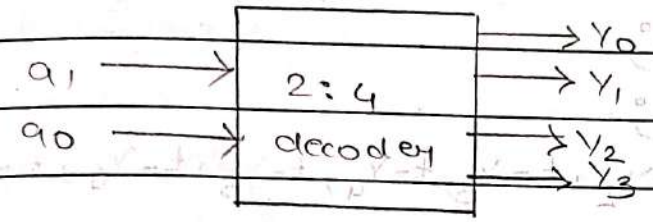
$Y_3 = a_1\,a_0$

```
module decoder_24 (Y0, Y1, Y2, Y3, a0, a1);
output Y0, Y1, Y2, Y3;
input a0, a1;
assign Y0 = (~a0) & (~a1);
assign Y1 = (~a1) & a0;
assign Y2 = a1 & (~a0);
assign Y3 = a1 & a0;
endmodule
```




```
module decoder_38 (Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, a2, a1, a0);
output Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;
input a2, a1, a0;
wire w1;
not (w1, a2);
decoder_24 d1 (Y7, Y6, Y5, Y4, a1, a0);
decoder_24 d2 (Y0, Y1, Y2, Y3, a1, a0);
endmodule
```

* Testbench for 2:4 decoder




```
module decoder24_tb;
reg t_a0, t_a1;
wire t_y0, t_y1, t_y2, t_y3;
decoder24 my_gate (.y0(t_y0), .y1(t_y1), .y2(t_y2),
                   .y3(t_y3), .a0(t_a0), .a1(t_a1));
initial
begin
t_a1 = 1'b0;
t_a0 = 1'b0;
#5
t_a1 = 1'b0;
t_a0 = 1'b1;
#5
t_a1 = 1'b1;
t_a0 = 1'b0;
#5
t_a1 = 1'b1;
t_a0 = 1'b1;
end
endmodule
```

* testbench for 3:8 decoder

```
module decoder38-tb;
reg t-a0, t-a1, t-a2;
wire t-y0, t-y1, t-y2, t-y3, t-y4, t-y5, t-y6, t-y7;

decoder38 my-gate (.y0(t-y0), .y1(t-y1), .y2(t-y2),
                   (t-y3), .y4(t-y4), .y5(t-y5), .y6(
                   .y7(t-y7));

initial
begin
t-a2 = 1'b0;
t-a1 = 1'b0;
t-a0 = 1'b0;
#5
t-a2 = 1'b0;
t-a1 = 1'b0;
t-a0 = 1'b1;
#5
t-a2 = 1'b0;
t-a1 = 1'b1;
t-a0 = 1'b0;
#5
t-a2 = 1'b0;
t-a1 = 1'b1;
t-a0 = 1'b1;
#5
```

```
t_q2 = 1'b0;
t_q1 = 1'b0;
t_q0 = 1'b0;
#5
t_q2 = 1'b0;
t_q1 = 1'b0;
t_q0 = 1'b1;
#5
t_q2 = 1'b1;
t_q1 = 1'b0;
t_q0 = 1'b0;
#5
t_q2 = 1'b1;
t_q1 = 1'b0;
t_q0 = 1'b1;
#5
t_q2 = 1'b1;
t_q1 = 1'b1;
t_q0 = 1'b0;
#5
end
endmodule
```

-X-