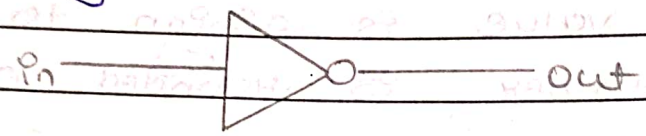
Unit : 2 :-

* Data flow - Modelling :-

1. NOT gate.

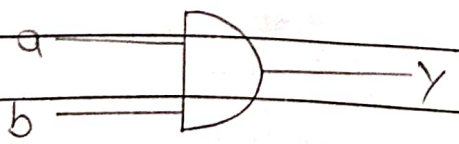


in	out
0	1
1	0

Out = ~~in~~ $\overline{in}$

```
module not_gate (out, in);
output out;
input in;
assign out = ~ in;
endmodule
```

2. And Gate.



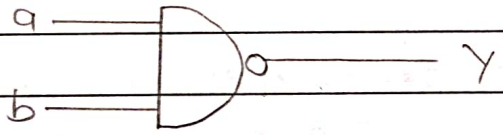
a	b	Y
0	0	0
0	1	0
1	0	0
1	1	1

$y = ab$

a \ b	0	1
0	0	0
1	0	1

```
Module      and-gate (Y, a, b);
output      Y;
input       a, b;
assign      Y = a & b;
end module
```

3. Nand gate



a	b	Y
0	0	1
0	1	1
1	0	1
1	1	0

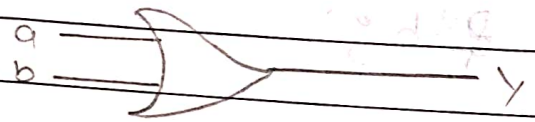
$Y = \overline{ab}$

$= not(a \cdot b)$

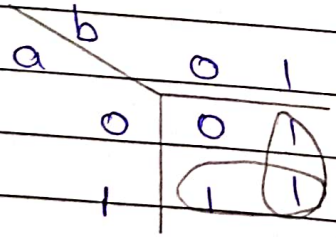
a \ b	0	1
0	1	1
1	1	0

```
module nand_gate (Y, a, b);
output Y;
input a, b;
assign Y = ~ (a & b)
endmodule
```

4. OR gate.



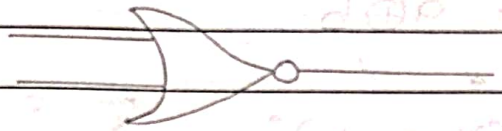
a	b	Y
0	0	0
0	1	1
1	0	1
1	1	1



$Y = a + b$

```
module or_gate (Y, a, b);
output Y;
input a, b;
assign Y = a | b;
endmodule
```

5. NOR gate



a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

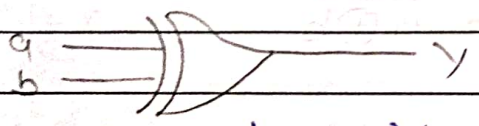
a \ b	0	1
0	1	0
1	0	0

$Y = \overline{a + b}$

$= not(a+b)$

```
module    nor_gate (Y, a, b);
output    Y;
input     a, b;
assign    Y = ~ (a|b);
endmodule
```

6. Xor gate



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

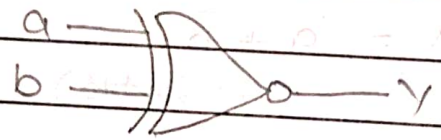
a \ b	0	1
0	0	1
1	1	0

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

$= a \oplus b$

```
module xor-gate (Y, a, b);
output Y;
input a, b;
assign Y = a ^ b;
endmodule
```

7. Xnor gate



a	b	Y
0	0	1
0	1	0
1	0	0
1	1	1

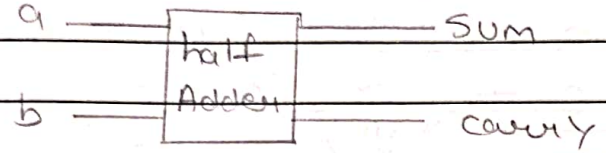
a \ b	0	1
0	1	0
1	0	1

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

$= \overline{a \oplus b}$

```
module xnor-gate (Y, a, b);
output Y;
input a, b;
assign Y = ~(a^b);
endmodule
```

8. half adder



a	b	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

a \ b	0	1
0	0	1
1	1	0

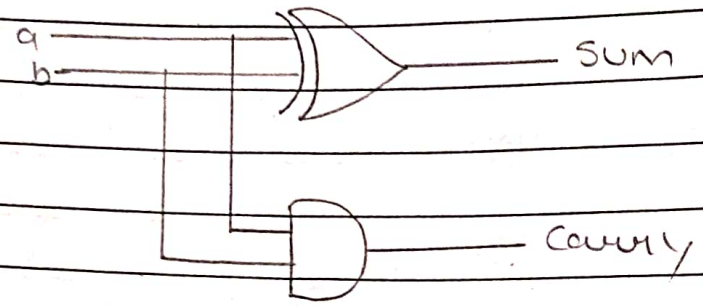
a \ b	0	1
0	0	0
1	0	1

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

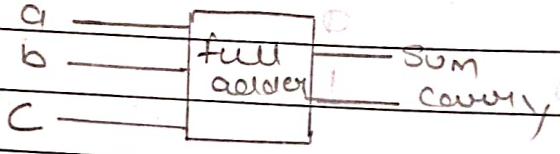
$S = A \oplus B$

$C = A \cdot B$

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



9. 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

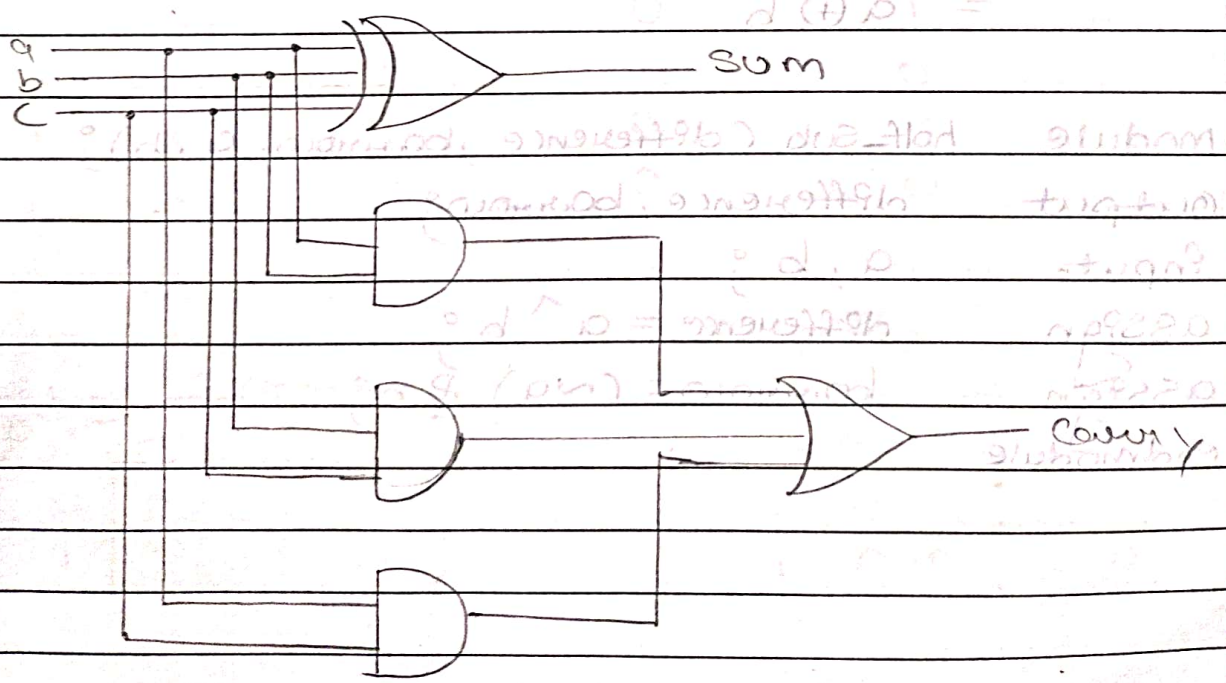
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$Sum = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$
$$= \bar{a}(\bar{b}c + b\bar{c}) + a(\bar{b}\bar{c} + bc)$$
$$= \bar{a}(b \oplus c) + a(\overline{b \oplus c})$$
$$= \bar{a}x + a\bar{x}$$
$$= a \oplus x$$
$$= a \oplus b \oplus c$$

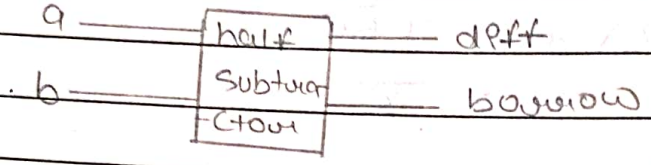
$$Carry = \bar{a}bc + bc + ac + ab$$

```
module full-add (sum, carry, a, b, c);
output  sum, carry;
input   a, b, c;
assign  sum = a^b^c;
assign  carry = (b⊕c) | (a⊕c) | (a&b)
assign  Carry = (b&c) | (a&c) | (a&b);
endmodule
```



3

10. Half Subtractor



a	b	diff	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

a \ b	0	1
0	0	1
1	1	0

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

$= a \oplus b$

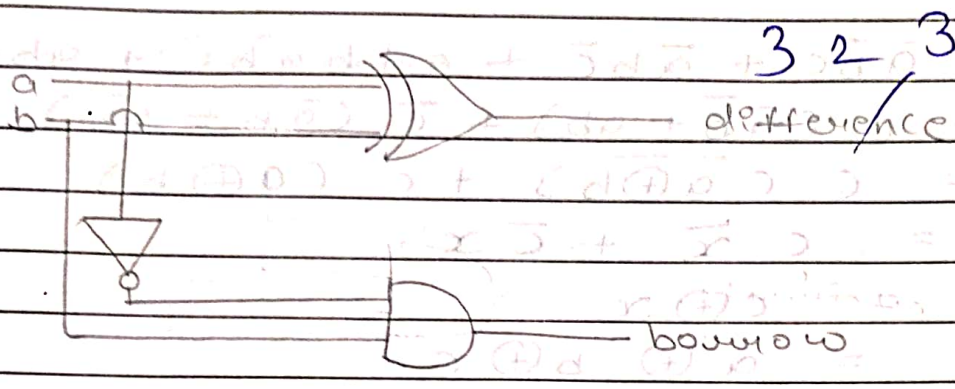
a \ b	0	1
0	0	1
1	0	0

$borrow = \bar{a}b$

```
module half_sub (difference, borrow, a, b);
output difference, borrow;
input a, b;
assign difference = a ^ b;
assign borrow = (~a) & b;
endmodule
```

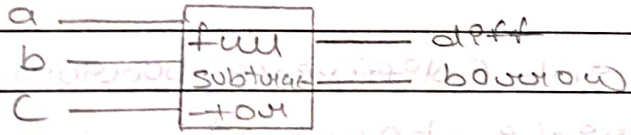
144, 167, 29, 48, 72, 40, 155, 28, 126, 161.

68, 158, 60,



3 2 30

4. Full Subtractor



a	b	c	diff	borrow
0	0	0	0	0
0	0	1	1	1
0	0	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference

a \ bc	00	01	11	10
0	0	1	0	1
1	1	0	1	0

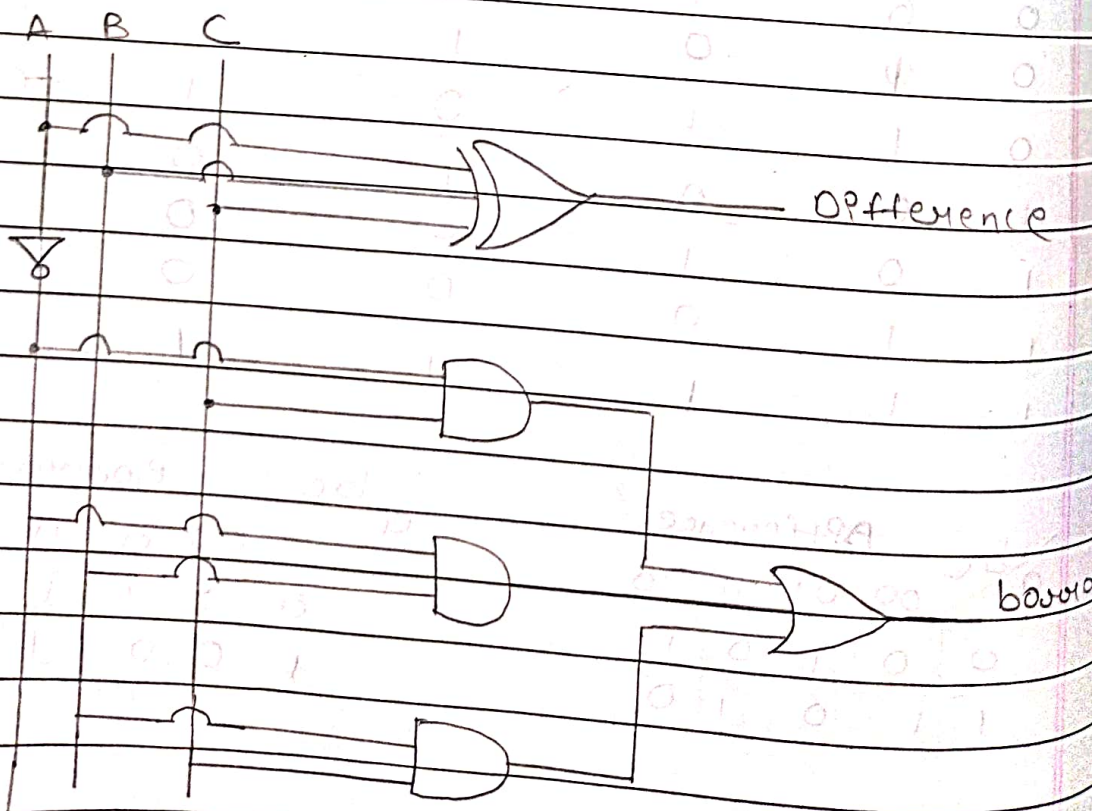
Borrow

a \ bc	00	01	11	10
0	0	1	1	1
1	0	0	1	0

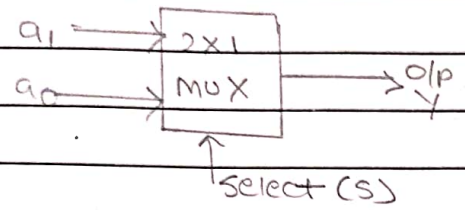
$$D = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$
$$= c(\bar{a}\bar{b} + ab) + \bar{c}(\bar{a}b + a\bar{b})$$
$$= c(\overline{a \oplus b}) + \bar{c}(a \oplus b)$$
$$= c\bar{x} + \bar{c}x$$
$$= c \oplus x$$
$$= a \oplus b \oplus c$$

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

```
module  full-sub (difference, borrow, a,b,c
output  difference, borrow;
input   a, b, c;
assign  difference = a^b^c;
assign  borrow =((~a)&c)|((~a)&b)|(b&
endmodule
```



12. 2:1 mux (multiplexer)



I/P	o/p
S	Y
0	a_0
1	a_1

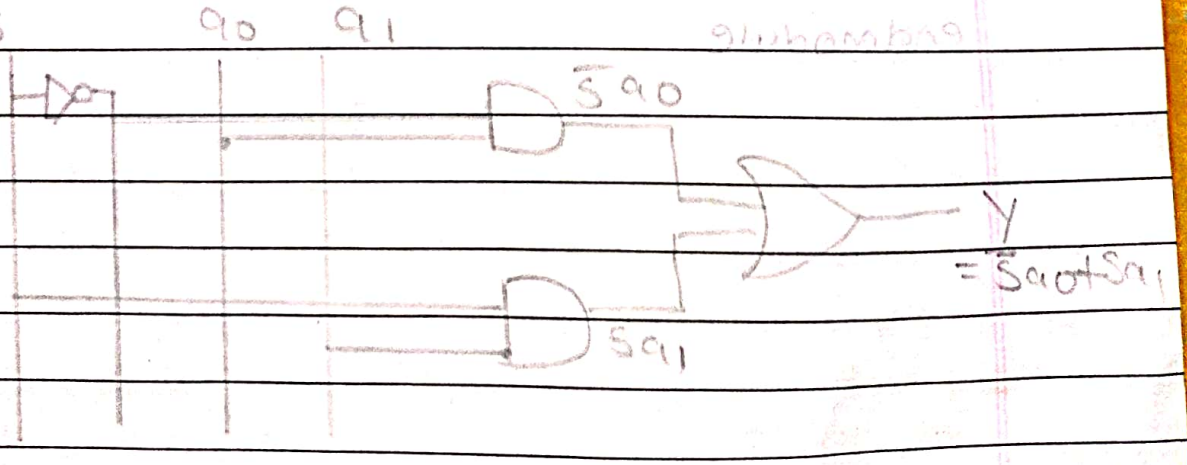
$Y = \overline{S}a_0 + Sa_1$

$Y = a_0$

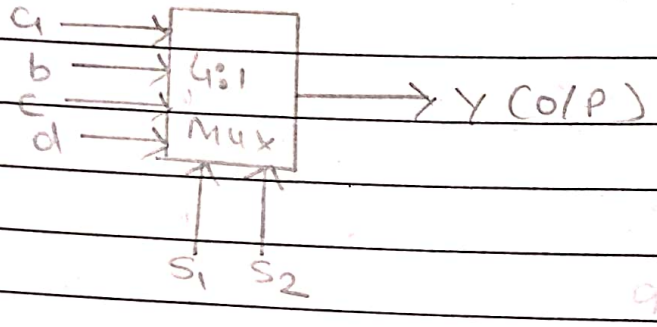
$S = 1$

$Y = a_1$

```
module   MUX21 (Y, S, a0, a1);
output   Y;
input    S, a0, a1;
assign   Y = ((~S) & a0) | (S & a1);
endmodule.
```



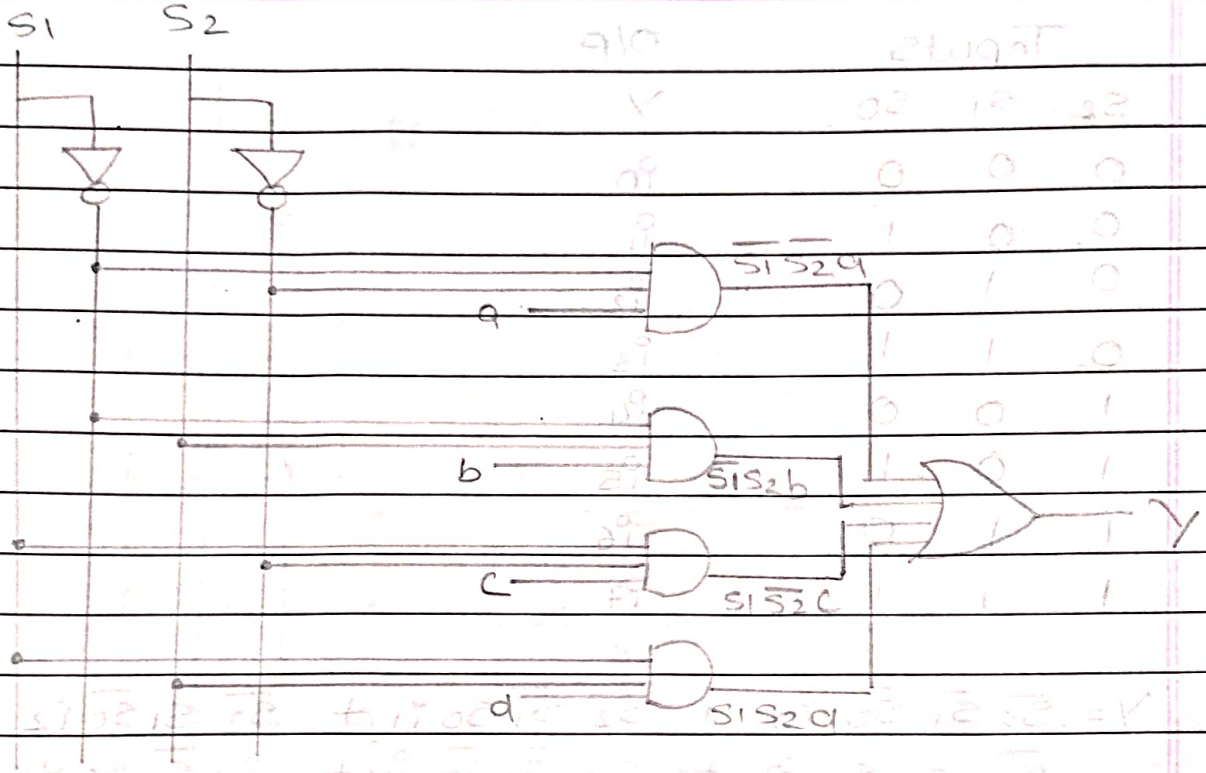
13 4:1 mux (multiplexer)



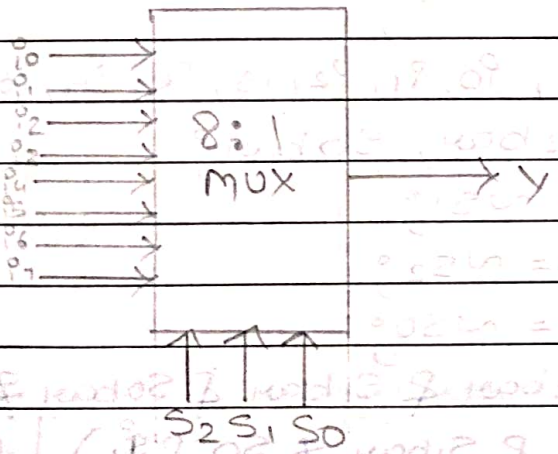
S_1	S_2	Y
0	0	a
0	1	b
1	0	c
1	1	d

$Y = \overline{S_1}\,\overline{S_2}a + \overline{S_1}S_2b + S_1\overline{S_2}c + S_1S_2d$

```
module  mux41 (Y, S1, S2, a, b, c, d);
output  Y;
input   a, b, c, d, S1, S2;
assign  Y = ((~S1) & (~S2) & a) | ((~S1) & S2 & b) |
            (S1 & (~S2) & c) | (S1 & S2 & a);
endmodule
```



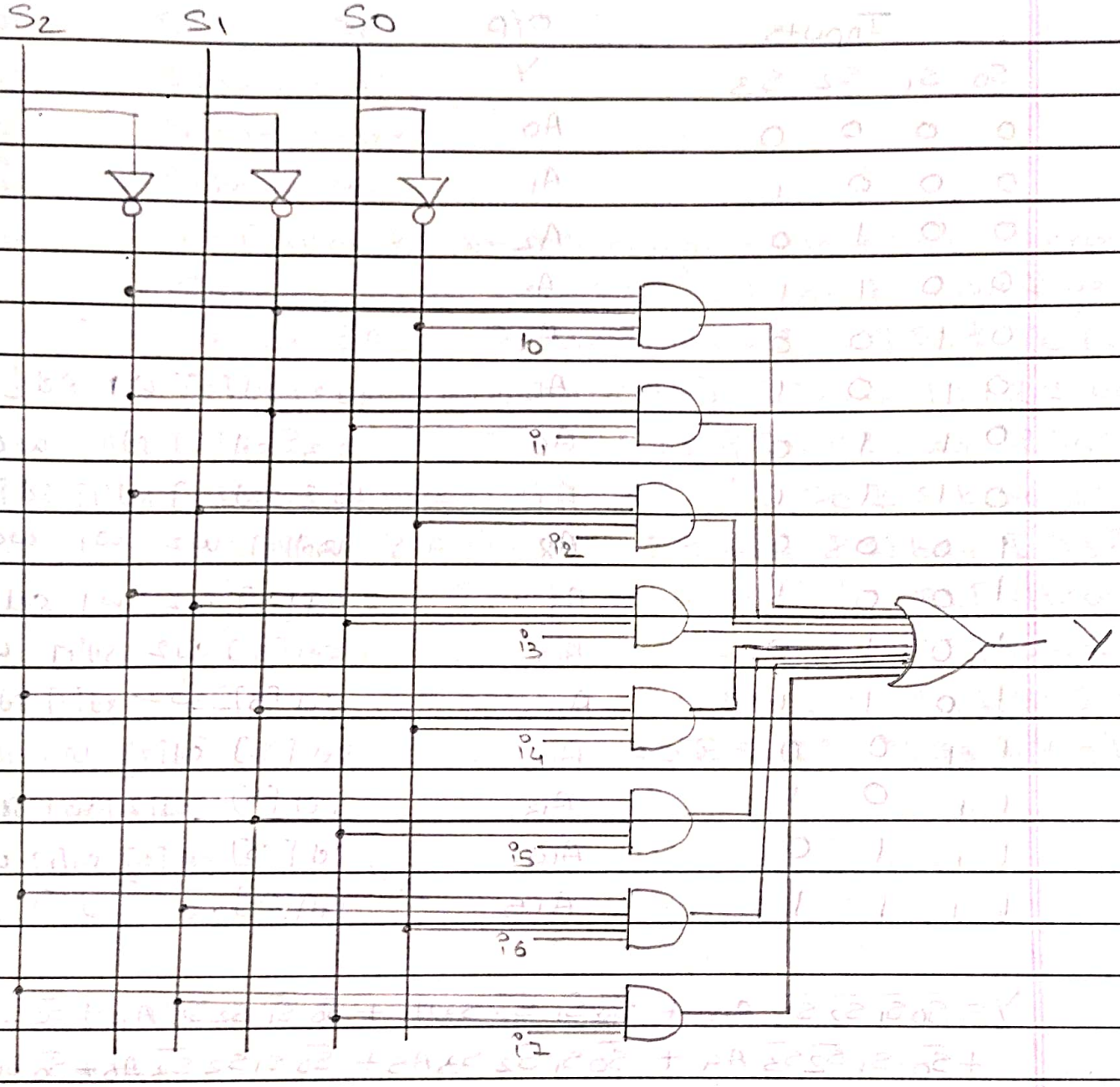
14. 8:1 mux (multiplexer)



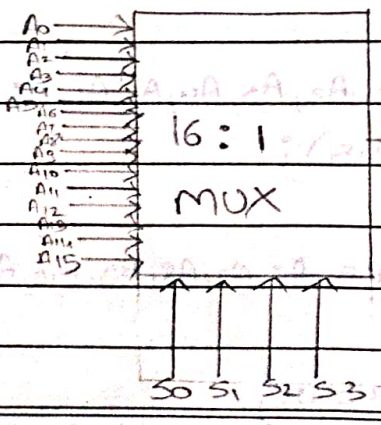
Inputs			O/P
S_2	S_1	S_0	Y
0	0	0	i_0
0	0	1	i_1
0	1	0	i_2
0	1	1	i_3
1	0	0	i_4
1	0	1	i_5
1	1	0	i_6
1	1	1	i_7

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

```
module mux81 (Y, S0, S1, S2, i0, i1, i2, i3, i4, i5, i6, i7);
output Y;
input S2, S1, S0, i0, i1, i2, i3, i4, i5, i6, i7;
wire S1bar, S2bar, S0bar;
assign S1bar = ~S1;
assign S2bar = ~S2;
assign S0bar = ~S0;
assign Y = (S2bar & S1bar & S0bar & i0) | (S2bar & S1bar & S0 & i1) | (S2bar & S1 & S0bar & i2) | (S2bar & S1 & S0 & i3) | (S2 & S1bar & S0bar & i4) | (S2 & S1bar & S0 & i5) | (S2 & S1 & S0bar & i6) | (S2 & S1 & S0 & i7);
endmodule
```



15 16:1 Mux (multiplexer)



Inputs				O/P
S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1
0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

w3 sel[2] w1 sel[0] is

w3 sel[2] sel[1] w0

w3 sel[2] sel[1] sel[0]

sel[3] w2 w1 w0

sel[3] w2 w1 sel[0]

sel[3] w2 sel[1] w0

sel[3] w2 sel[1] sel[0]

sel[3] sel[2] w1 w0

sel[3] sel[2] w1 sel[0]

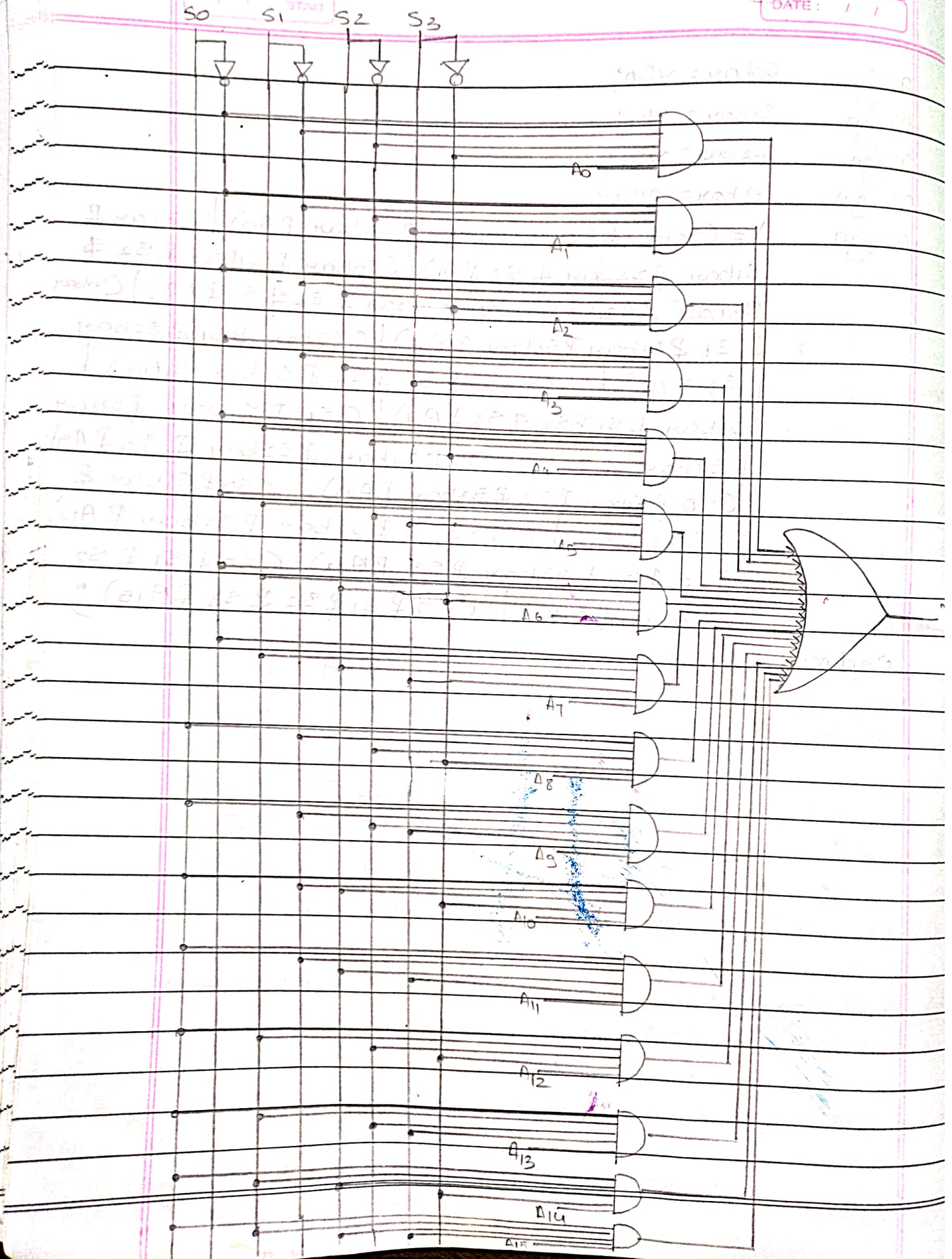
sel[3] sel[2] sel[1] w0

sel[3] sel[2] sel[1] sel[0]

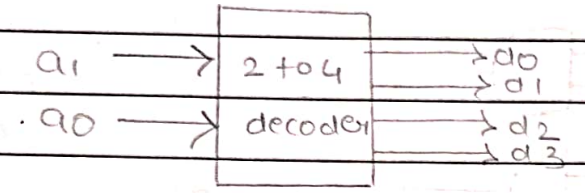
$Y = \bar{S_0}\bar{S_1}\bar{S_2}\bar{S_3}A_0 + \bar{S_0}\bar{S_1}\bar{S_2}S_3A_1 + \bar{S_0}\bar{S_1}S_2\bar{S_3}A_2 + \bar{S_0}\bar{S_1}S_2S_3$
$+ \bar{S_0}S_1\bar{S_2}\bar{S_3}A_4 + \bar{S_0}S_1\bar{S_2}S_3A_5 + \bar{S_0}S_1S_2\bar{S_3}A_6 + \bar{S_0}S_1S_2S_3A$
$+ S_0\bar{S_1}\bar{S_2}\bar{S_3}A_8 + S_0\bar{S_1}\bar{S_2}S_3A_9 + S_0\bar{S_1}S_2\bar{S_3}A_{10} + S_0\bar{S_1}S_2S_3A$
$+ S_0S_1\bar{S_2}\bar{S_3}A_{12} + S_0S_1\bar{S_2}S_3A_{13} + S_0S_1S_2\bar{S_3}A_{14} + S_0S_1S_2S_3A$

```
module MUX16 (Y,S0,S1,S2,S3,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,
              A11,A12,A13,A14,A15);
Output  Y;
Input   S0,S1,S2,S3,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,
        A14,A15;
wire    S0bar,S1bar,S2bar,S3bar;
```

```
assign   S0bar = ~S0;
assign   S1bar = ~S1;
assign   S2bar = ~S2;
assign   S3bar = ~S3;
assign   Y = (S0bar & S1bar & S2bar & S3bar & A0) | (S0bar &
         S1bar & S2bar & S3 & A1) | (S0bar & S1bar & S2 &
         S3bar & A2) | (S0bar & S1bar & S2 & S3 & A3) | (S0bar
         & S1 & S2bar & S3bar & A4) | (S0bar & S1 & S2bar
         & S3 & A5) | (S0bar & S1 & S2 & S3bar & A6) |
         (S0bar & S1 & S2 & S3 & A7) | (S0 & S1bar & S2bar
         & S3bar & A8) | (S0 & S1bar & S2bar & S3 & A9)
         | (S0 & S1bar & S2 & S3bar & A10) | (S0 & S1bar &
         S2 & S3 & A11) | (S0 & S1 & S2bar & S3bar & A12)
         | (S0 & S1 & S2bar & S3 & A13) | (S0 & S1 & S2
         & S3bar & A14) | (S0 & S1 & S2 & S3 & A15);
endmodule
```


S0
S1
S2
S3
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15

16. 2 to 4 decoder



I/P		O/P			
a_1	a_0	d_3	d_2	d_1	d_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
X	X	0	0	0	0

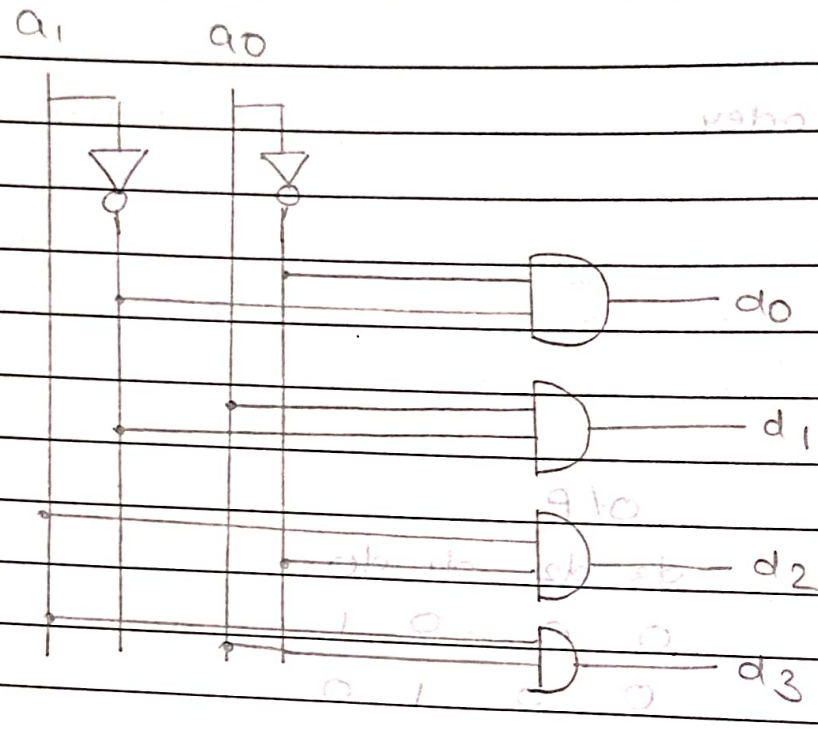
$d_0 = \overline{a_1}\,\overline{a_0}$

$d_1 = \overline{a_1}\,a_0$

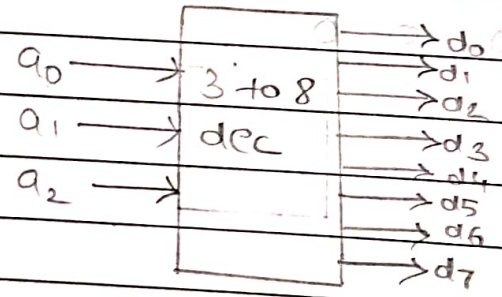
$d_2 = a_1\,\overline{a_0}$

$d_3 = a_1\,a_0$

```
module dec24(d0, d1, d2, d3, a1, a0);
output d0, d1, d2, d3;
input a1, a0;
wire s0, s1;
assign s0 = ~a0;
assign s1 = ~a1;
assign d0 = s0 & s1;
assign d1 = s0 & a0;
assign d2 = a1 & s0;
assign d3 = a1 & a0;
endmodule
```



17. 3 to 8 Decoder



I/P			O/P							
a0	a1	a3	d7	d6	d5	d4	d3	d2	d1	d0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$d_0 = \bar{q_0}\bar{q_1}\bar{q_2}$

$d_1 = \bar{q_0}\bar{q_1}q_2$

$d_2 = \bar{q_0}q_1\bar{q_2}$

$d_3 = \bar{q_0}q_1q_2$

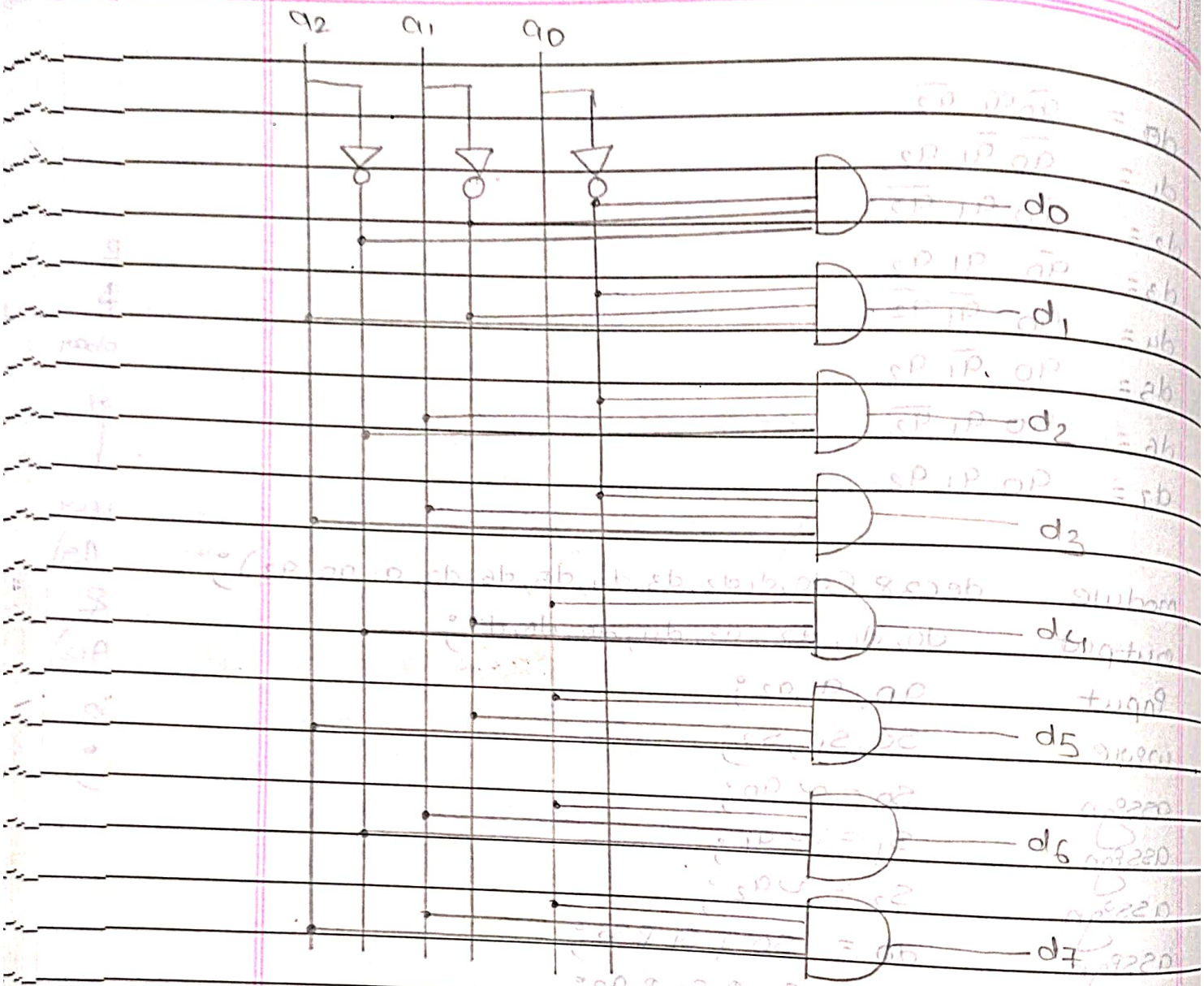
$d_4 = q_0\bar{q_1}\bar{q_2}$

$d_5 = q_0\bar{q_1}q_2$

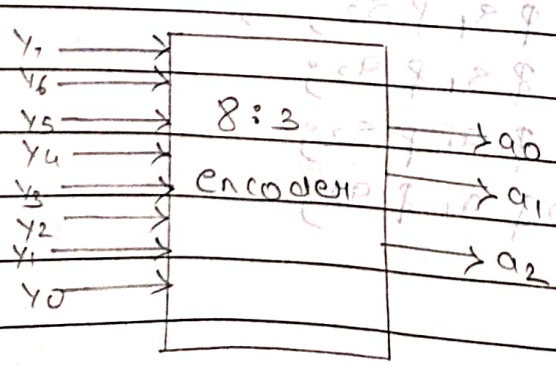
$d_6 = q_0q_1\bar{q_2}$

$d_7 = q_0q_1q_2$

```
module    dec38 (d0,d1,d2,d3,d4,d5,d6,d7,q1,q0,q3);
output    d0, d1, d2, d3, d4, d5, d6, d7;
input     q0, q1, q2;
wire      s0, s1, s2;
assign    s0 = ~q0;
assign    s1 = ~q1;
assign    s2 = ~q2;
assign    d0 = s0 & s1 & s2;
assign    d1 = s0 & s1 & q2;
assign    d2 = s0 & q1 & s2;
assign    d3 = s0 & q1 & q2;
assign    d4 = q0 & s1 & s2;
assign    d5 = q0 & s1 & q2;
assign    d6 = q0 & q1 & s2;
assign    d7 = q0 & q1 & q2;
endmodule
```



18. 8:3 encoder



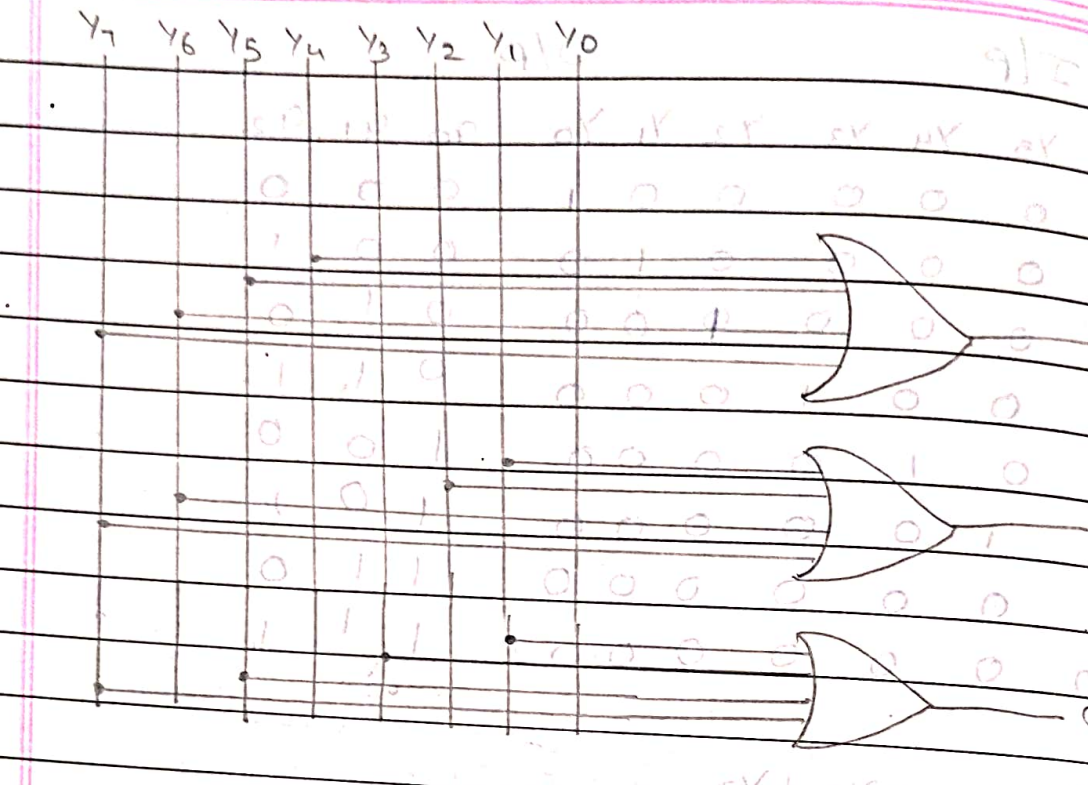
I/P								O/P		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	q_0	q_1	q_2
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$q_2 = Y_1 + Y_3 + Y_5 + Y_7$

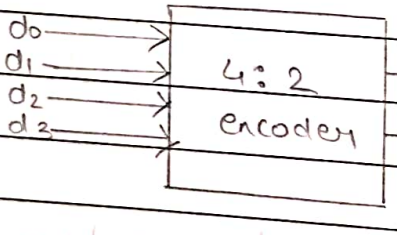
$q_1 = Y_2 + Y_3 + Y_6 + Y_7$

$q_0 = Y_4 + Y_5 + Y_6 + Y_7$

```
module    enc83 (q,Y);
output    [2:0] q;
input     [7:0] Y;
assign    q[2] = Y[1] | Y[3] | Y[5] | Y[7];
assign    q[1] = Y[2] | Y[3] | Y[6] | Y[7];
assign    q[0] = Y[4] | Y[5] | Y[6] | Y[7];
endmodule
```



19. 4:2 encoder

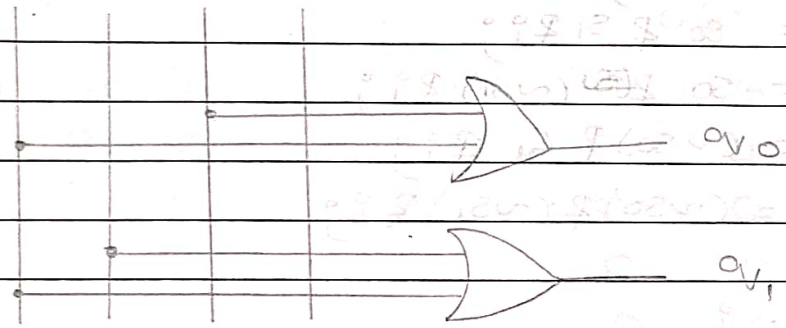


I/P				O/P	
d_3	d_2	d_1	d_0	q_1	q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$q_0 = d_1 + d_3$

$q_1 = d_2 + d_3$

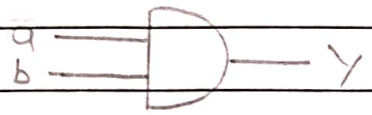
```
module enc42 (qv1, qv0, d3, d2, d1, d0);
output qv1, qv0;
input d3, d2, d1, d0;
assign qv0 = d1 | d3;
assign qv1 = d2 | d3;
endmodule
```



PAGE NO.:
DATE : / /

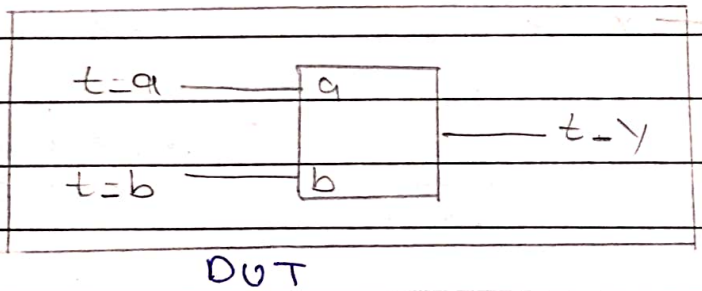
* Testbench code :-

1. and gate



a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

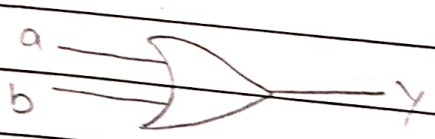
```
module and-gate (y, a, b);
output y;
input a, b;
assign y = a & b;
endmodule
```




```
module and-gate-tb;
wire t-y;
reg t-a, t-b;
and-gate my-gate (.y(t-y), .a(t-a), .b(t-b));
initial
```

```
begin
t_a = 1'b0;
t_b = 1'b0;
#5
t_a = 1'b1;
t_b = 1'b0;
#5
t_a = 1'b0;
t_b = 1'b1;
#5
t_a = 1'b1;
t_b = 1'b1;
end
endmodule
```

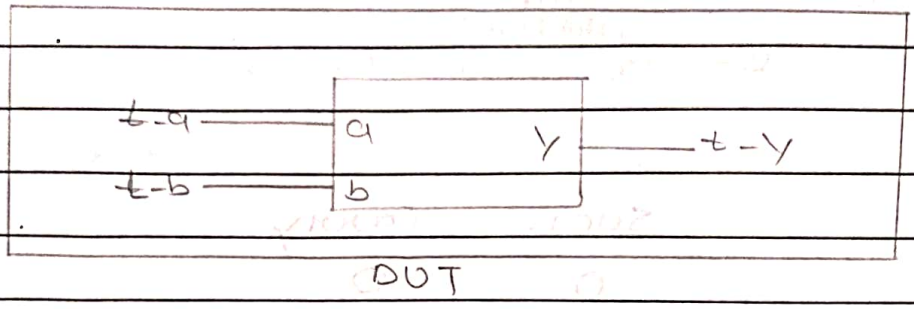
2. OR Gate



a	b	Y
0	0	0
0	1	1
1	0	1
1	1	1

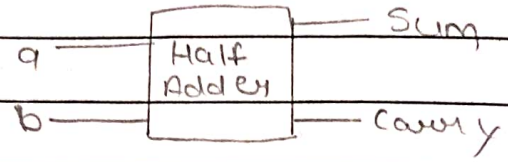
```
Module OR_gate (Y, a, b);
output Y;
input a, b;
```

assign y = a | b;
endmodule



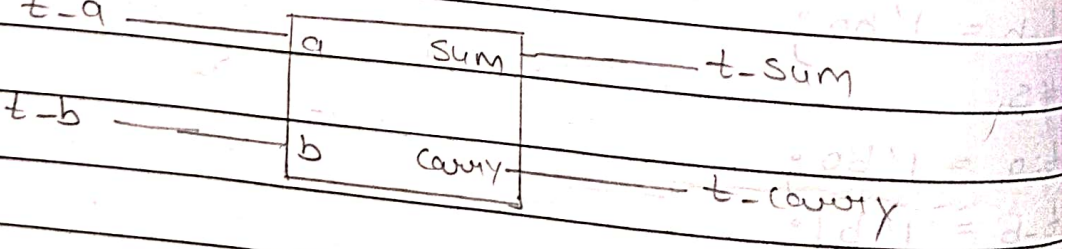
module Or-gate-tb;
wire t-Y;
reg t-a, t-b;
Or-gate my-gate (.Y(t-Y), .a(t-a), .b(t-b));
initial
begin
t-a = 1'b0;
t-b = 1'b0;
#5
t-a = 1'b1;
t-b = 1'b0;
#5
t-a = 1'b0;
t-b = 1'b1;
#5
t-a = 1'b1;
t-b = 1'b1;
end
endmodule.

3. Half adder



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

```
Module half-adder (sum, carry, a, b);
output sum, carry;
input a, b;
assign sum = a^b;
assign carry = a & b;
endmodule
```



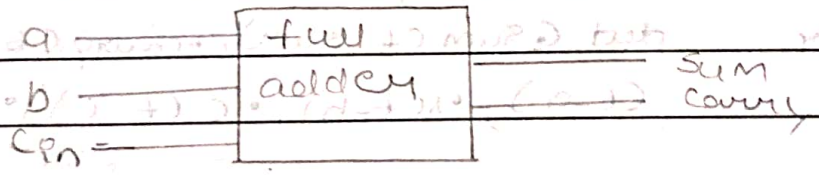

```
Module half-adder-tb;
wire t-sum, t-carry;
reg t-a, t-b;
half_adder my-gate (.sum(t-sum), .carry(t-
                    .a(t-a), .b(t-b));
Initial
```

full adder.

PAGE NO.:
DATE : / /

```
begin
t_a = 1'b0;
t_b = 1'b0;
#5
t_a = 1'b1;
t_b = 1'b0;
#5
t_a = 1'b0;
t_b = 1'b1;
#5
t_a = 1'b0;
t_b = 1'b1;
end
endmodule.
```

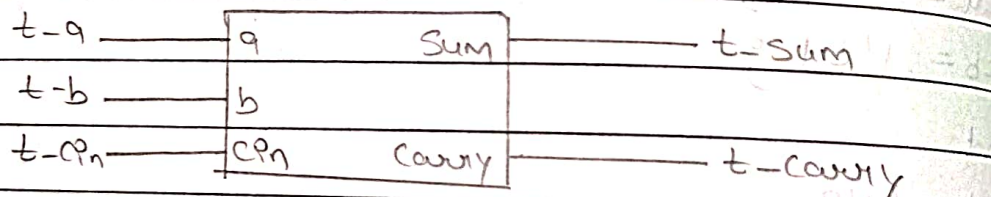
4 Full Adder



a	b	Cin	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

```
module      full-adder (sum, carry, a, b, cin);
output      sum, carry;
input       a, b, cin;
assign      sum = a^b^c;
assign      carry = (b&c) | (a&c) | (a&b);
endmodule
```




```
module      full-adder_tb;
wire        t-sum, t-carry;
reg         t-a, t-b, t-c;
full-adder  dut (.sum(t-sum), .carry(t-carry),
                 .a(t-a), .b(t-b), .c(t-c));
initial
begin
t-a = 1'b0;
t-b = 1'b0;
t-cin = 1'b0;
#5
t-a = 1'b0;
t-b = 1'b0;
```

```
t_cin = 1'b1;
#5
t_a = 1'b0;
t_b = 1'b1;
t_cin = 1'b0;
#5
t_a = 1'b0;
t_b = 1'b1;
t_cin = 1'b1;
#5
t_a = 1'b1;
t_b = 1'b0;
t_cin = 1'b0;
#5
t_a = 1'b1;
t_b = 1'b0;
t_cin = 1'b1;
#5
t_a = 1'b1;
t_b = 1'b1;
t_cin = 1'b0;
#5
t_a = 1'b1;
t_b = 1'b1;
t_cin = 1'b1;
end
endmodule
```