UNIT-1

VLSI → Very large scale integration.

VHDL → VHSIC HDL
Very high speed integration ckt hardware description language

VLSI Design Flow :-

Specification (Design Entities)
↓
Design layout / ckt diagram (Component selection)
↓
Finctional / timing selection (Software)
↓
Synthesis (minimization of ckt / optimization)
↓
Placing & Routing (component mounting/wiring interconnection hardware design to mount max. component on a single chip)
↓
Verification (testing)
↓
Chip integration (Physical chip)

VHDL CODE :

```
library ieee;
use ieee.std_logic_1164.all;  --library description

Design entity

A : BIT  <-- Architecture body (main code)
```

code format

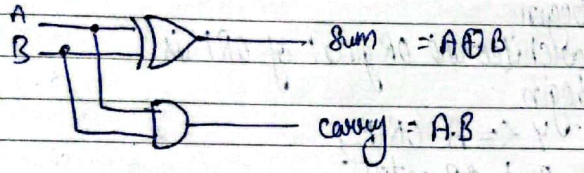
Page No. 11
Date

MODELLING STYLES :-

⇒ Types of architectural code

i) Data flow → logic expression
ii) Structural
iii) ~~Behavioral~~ Behavioral
iv) Mixed

i) Data flow (logic expression)



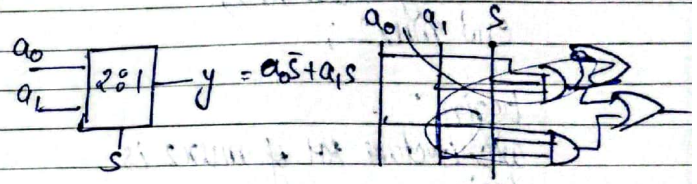
Half adder

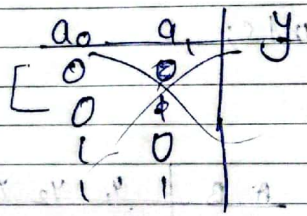
Code :-

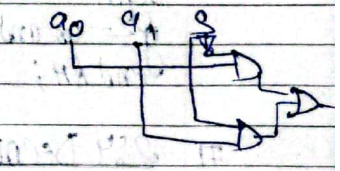
```
library ieee;
use ieee.std_logic_1164.all;

entity HA is
Port ( A : in bit;
       B : in bit;
       S : out bit;
       C : out bit);
end HA
```

~~begin~~
architecture RH of HA is
begin
S <= A xor B;
C <= A and B;
end RH;

~~2:4 Decoder~~
2:1 MUX







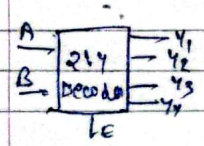
s	a0	a1	y
0	0	x	0
0	1	x	a0
1	x	0	0
1	x	1	a1

8:1 mux
3:8 decoder

Code :

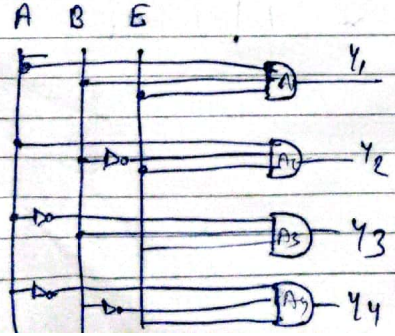
```
library ieee;
use ieee.std_logic_1164.all;

entity mux2 is
port (a0 : in bit;
      a1 : in bit;
      s  : in bit;
      y  : out bit);
end mux2;

begin
architecture RH of mux2 is
begin
y <= a0 and (not s) or a1 and s;
end RH;
```

2:4 DECODER



A	B	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

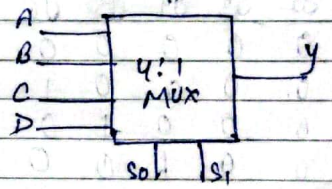
$Y_1 = AB\bar{E}$
$Y_2 = A\bar{B}E$
$Y_3 = \bar{A}BE$
$Y_4 = \bar{A}\bar{B}E$



Code

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2 is
port (a0 : in std_logic;
      E1 , a1 : in std_logic;
      Y0, Y1, Y2, Y3 : out std_logic);
end mux2;

begin
architecture RH of mux2 is
begin
y0 <= E and (not a0) and (not a1);
y1 <= E and (not A) and B;
y2 <= E and A and (not B);
y3 <= E and A and B;
end RH;
```

4:1 MUX



S_0	S_1	A	B	C	D	y
0	0	1	x	x	x	a_0
0	1	x	1	x	x	a_1
1	0	x	x	1	x	a_2
1	1	x	x	x	1	a_3

$y = \bar{S}_0\bar{S}_1A + \bar{S}_0S_1B + S_0\bar{S}_1C + S_0S_1D$

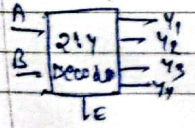
4:1 mux
8:1 mux
3:8 decoder

Page No. 14
Date

Code :

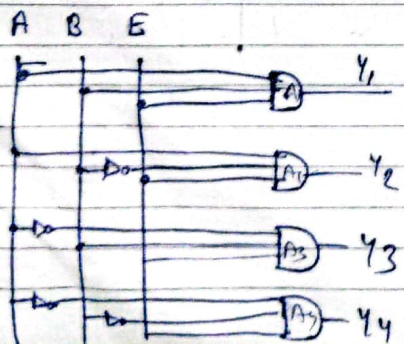
```
library ieee;
use ieee.std_logic_1164.all;

entity mux2 is
port (a0 : in bit;
      a1 : in bit;
      s  : in bit;
      y  : out bit);
end mux2;

begin
architecture RH of mux2 is
begin
y <= a0 and (not s) or a1 and s;
end RH;
```

2:4 DECODER



A	B	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

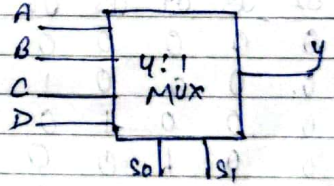
$Y_1 = ABE$
$Y_2 = A\bar{B}E$
$Y_3 = \bar{A}BE$
$Y_4 = \bar{A}\bar{B}E$



Page No. 15
Date

Code
```
library ieee;
use ieee.std_logic_1164.all;

entity mux2 is
port (A: in std_logic;
      E, B: in std_logic;
      Y0, Y1, Y2, Y3 : out std_logic);
end mux2;

architecture RH of mux2 is
begin
y1 <= E and (not A) and (not B);
y3 <= E and (not A) and B;
y2 <= E and A and (not B);
y1 <= E and A and B;
end RH;
```

4:1 MUX



S_0	S_1	A	B	C	D	y
0	0	1	x	x	x	A
0	1	x	1	b	b	B
1	0	x	y	1	x	C
1	1	b	x	x	1	D

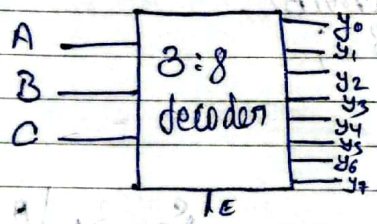
$y = \bar{S}_0\bar{S}_1A + \bar{S}_0S_1B + S_0\bar{S}_1C + S_0S_1D$

~~S0 S1 A B C D~~

Code :

```
library ieee;
use ieee.std_logic_1164.all;
entity mux4 is
port (A,B,C,D,S0,S1 : in bits;
      y : out bit);
end mux4 ii
begin
architecture Muxi of mux4 is
begin
y <= S0 and A and (not S0) and (not S1) or B and (not S0) and S1 or
     C and S0 and (not S1) or D and S0 and S1.
end muxi;
```

3:8 decoder :



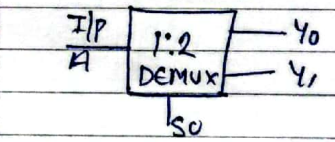
$y_0 = \bar{A}\bar{B}\bar{C}E$ $y_5 = A\bar{B}C\bar{E}$
$y_1 = \bar{A}\bar{B}CE$ $y_6 = A\bar{B}\bar{C}E$
$y_2 = \bar{A}B\bar{C}E$ $y_7 = ABCE$
$y_3 = AB\bar{C}\bar{E}$
$y_4 = ABCE$

E	A	B	C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Code :

```
library ieee;
use ieee.std_logic_1164.all;
entity deco8 is
port (A,B,C,E : in bits;
      y0,y1,y2,y3,y4,y5,y6,y7 : out bits);
end deco8;

begin
architecture decod of deco8 is
begin
y0 <= A and B and C and E
y1 <= E and B and C and (not A)
y2 <= A and (not B) and C and E
y3 <= E and A and (not B) and (not C)
y4 <= E and (not A) and B and C
y5 <= E and (not A) and B and (not C)
y6 <= E and (not A) and (not B) and C
y7 <= E and (not A) and (not B) and (not C)
end decod;
```

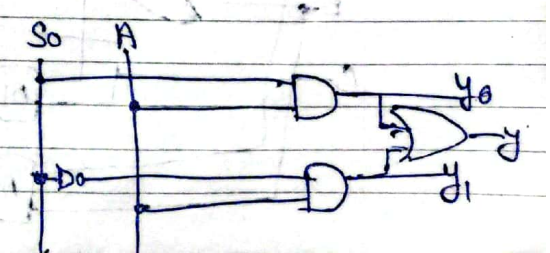
1:2 DeMux :



$S_0 = 0 \Rightarrow y_1$
$S_0 = 1 \Rightarrow y_0$

S_0	A	y_0	y_1	y
0	0	0	0	0
0	1	0	1	1
1	0	0	0	0
1	1	1	0	1

$y_0 = S_0 A$
$y_1 = \bar{S_0} A$
$y = S_0 A + \bar{S_0} A$

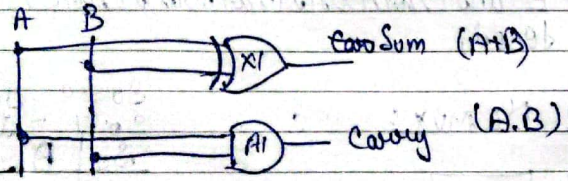


CODE:-

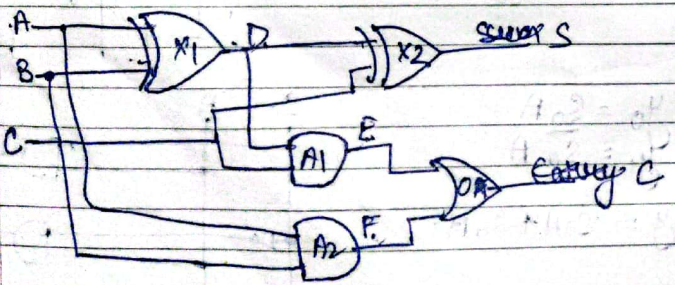
```
library ieee;
use ieee.std_logic_1164.all;
entity demux2 is
port ( s0, A : in bit;
       y0, y1 : out bit);
end demux2;

begin
architecture rH of demux2 is
begin
  y0 <= A and s0
  y1 <= A and (not s0)
end rH;
```

#ii) STRUCTURAL :-

⇒ Half adder :



⇒ Full adder



For half adder code :-

```
library ieee;
use ieee.std_logic_1164.all;
entity HA is
port (A, B, : in bit;
      S, C : out bit);
end HA;
begin
architecture rtl of HA is
component xor2
port ( l,m : in bit;
       n : out bit);
 end component;
component and2
port ( P,Q : in bit;
       r : out bit);
 end component;

begin
 x1 : xor2 portmap (A,B,S);
 A1 : and2 portmap (A,B,C);
end rtl;
```

⇒ Full adder code :-

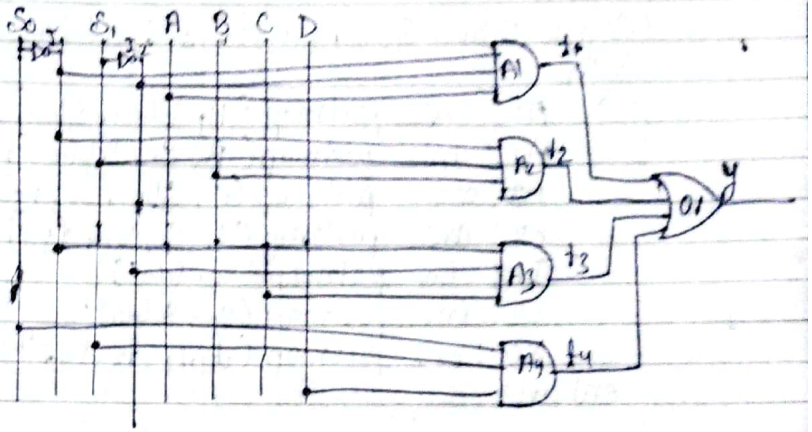
```
library ieee;
use ieee.std_logic_1164.all;
entity FA is
port (A, B, C : in bit;
      S, Ca : out bit);
end FA; D,E,F : in out bit);
```

Full adder Code :-

```
library begin
    architecture sbl of FA is
    component xor2
    port (l, m : in bit;
          n : out bit);
    end component; xor2
component and2
port (P, q : in bit;
      r : out bit);
end component;

component or2
port(x, y : in bit;
     z : out bit);
end component;
signal D,E,F : std_logic;
begin
x1 : xor2 portmap(A,B,D);
x2 : xor2 portmap(D,C,S);
A1 : and2 portmap(A,B,F);
A2 : and2 portmap(C,D,E);
O1 : Or2 portmap(E,F,C);
```


4:1 mux using structural modeling

$y = \bar{S_0}\bar{S_1}A + \bar{S_0}S_1B + S_0\bar{S_1}C + S_0S_1D$



Case

```
library ieee ;
use ieee.std_logic_1164.all;
entity MUX4 is
port (A,B,C,D,S0,S1 : in bit;
      & y : out bit);
end eMUX4;
begin
architecture sbl of mux4 is
component and2
port (P,l,m : in bit;
      n : out bit);
end component;
component or2
port (Q,u,S : in bit;
      t : out bit);
end component
```

```
component inv
port (Se : in bit;
      f : out bit);
end component;

Signal Sobar, S1bar, t1, t2, t3, t4 : std_logic;
begin
    A1 : and2 portmap (A, Sobar, S1bar, t1);
    A2 : and2 portmap (B, S1, Sobar, t2);
    A3 : and2 portmap (C, S1bar, S0, t3);
    A4 : and2 portmap (d, S0, S1, t4);
    I1 : inv portmap (S0, Sobar);
    I2 : inv portmap (S1, S1bar);
    O1 : or2 portmap (t1, t2, t3, t4, Y);
end rtl;
```

2:4 decoder :-

$Y_1 = A\bar{B}E$, $Y_2 = A\bar{B}E$

$Y_3 = \bar{A}BE$ $Y_4 = \bar{A}\bar{B}E$

Code :-

```
library ieee;
use ieee.std_logic_1164.all;
entity dec2 is
port (A, B, E : in bits;
      Y1, Y2, Y3, Y4 : out bit);
end dec;
begin
begin architecture rtl of dec2 is
```



```
component inv
port (c : in bit;
      d : out bit);
end component;

component and2
port (e, f, g : in bit;
      h : out bit);
end component;

begin
    I1 : inv portmap (A
Signal Abar, bbar : std_logic;
begin
    I1 : inv portmap (A, Abar);
    I2 : inv portmap (B, Bbar);
    A1 : and2 portmap (A, B, E, Y1);
    A2 : and2 portmap (A, Bbar, E, Y2);
    A3 : and2 portmap (Abar, B, E, Y3);
    A4 : and2 portmap (Abar, Bbar, E, Y4);
end rtl;
```

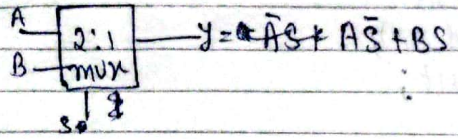
iii) # BEHAVIORAL MODELING :-

Conditions are :

i) if-else :

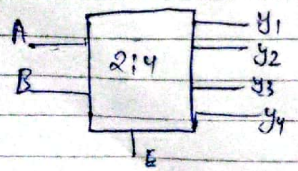
```
if condition then
    statements
else
    statements.
```

2:1 mux :




```
library ieee;
use ieee std_logic_1164.all;
entity mux2 is
port (A, B, S: in bit;
      Y: out bit);
end mux;

begin
architecture rtl of mux2 is
begin
    process (A, B, S)
begin
    if S = '0' then
        y <= A;
    else
        y <= B;
end if;
end process;
end rtl;
```

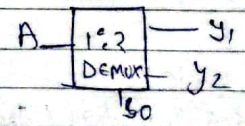
2:4 Decoder :



$Y_1 = ABE$
$Y_2 = A\bar{B}E$
$Y_3 = \bar{A}BE$
$Y_4 = \bar{A}\bar{B}E$


```
library ieee;
use ieee std_logic_1164.all;
entity Deco2 is
port (A, B, E : in bits;
      Y1, Y2, Y3, Y4 : out bit);
end Deco2;

architecture rtl of Deco2 is
begin
    process (A, B, E)
begin
    if E = '1' then
        Y1 <= A and B and E;
        Y2 <= A and (not B);
        Y3 <= (not A) and B;
        Y4 <= (not A) and (not B);
    else
        Y <= "ZZZZ";
end if;
end process
end rtl;
```

1:2 DEMUX :



$y_0 = S_0 A$
$y_1 = \bar{S_0} A$
$y = S_0 A + \bar{S_0} A$

```
library ieee;
use ieee std_logic_1164.all;
entity Dem is
```

port (A, S_0 : in bit;
y_1, y_2 : out bit);
end dem;

begin
~~Process~~ architecture rtl of dem is
process (A, S_0)
begin
~~if S_0 = '0' then~~
~~y_1 <= '0' or A and S_0;~~
~~else~~
y_2
if S_0 = 0 then
y_1 <= A;
else
y_2 <= A;
end if;
end process;
end rtl;

~~1:4 Demux~~ 4:1 MUX:

library ieee;
use ieee.std_logic_1164.all;
entity ~~Dem4~~ mux4 is
port (a, b, c, d : in std_logic;
S : in std_logic_vector (1 down to 0);
Y : out std_logic);
end mux;

architecture rtl of mux4 is;
begin
process (a, b, c, d, S)
begin
if S = "00" then
y <= a;
else if S = "01" then
y <= b;
else if S = "10" then
y <= c;
else if S = "11" then
y <= d;
~~else~~
else y <= d;
end if;
end else
end rtl;

=> CASE Statement:-

=> 4:1 MUX using case statement:-

library ieee;
use ieee.std_logic_1164.all;
entity mux4 is
port (a, b, c, d : in bit;
S : in bit_vector (1 down to 0);
Y : out std_logic);
end mux4;

```
architecture rtl of mux4 is
begin
    process (a, b, c, d, s);
begin
  case s is
    when "00" =>
      "00" => y <= a;
      '01' => y <= b;
      "10" => y <= c;
      "11" => y <= d;
    when others => y <= "Z";
      end case;
          end process
        end rtl;
```

4:1 MUX USING WITH SELECT STATEMENT :

```
library ieee;
entity
```

1:4 DEMUX :

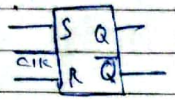
```
architecture rtl of mux1 is
begin
    process (a, b, c, d, s);
begin
    with s select
      y <= a when "00";
      y <= b when "01";
      y <= c when "10";
      y <= d when "11";
when others => y <= "Z";
end with select;
```


MIXED MODELLING :

⇒ Full Adder :

```
library ieee;
use ieee.std_logic_1164.all;
entity FA is
port (A, B, C : in bit;
      S, C : out bit);

architecture rtl of FA is
 component XOR1
    port (l, m : in bit;
          n : out bit);
 end component;

 signal s1 : sin_logic;

begin
   x1 : xor1 portmap (A, B, s1
```

SRFF :-



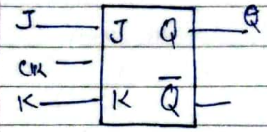
Rst	CLK	S	R	Q	$\bar{Q}$
0	↑	0	0	NC	NC
0	↑	0	1	0	1
0	↑	1	0	1	0
0	↑	1	1	x	x

Code

```
library ieee;
use ieee.std_logic_1164.all;
entity SRFF is
port ( Rst, CLK, S, R, : in bit;
        Q , Qbar: out bit);
end SRFF;

architecture rtl of SRFF is
begin
    Process (clk, rst)
begin
  if rst = '1' then
      Q <= '0' ; Qbar <= '1';
  else if clk and clk'event = '1' then
   if ~~Q<=Q; Qbar<=Qbar; then~~ S=0, R=0 then
      Q<=Q , Qbar<=Qbar;
  else if S=1, R=0 then
      Q<=1 , Qbar<='0';

  else if S=0, R=1 then
      Q=0; Qbar <= '1';
end if;
end if;
end Process;
end rtl;
```

Asynchronous = Rst, Clk

JKFF :-

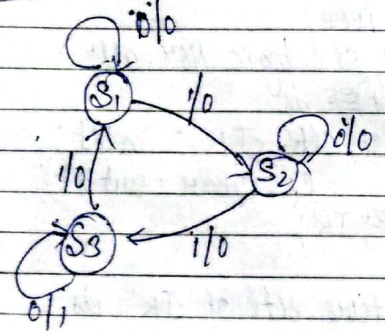


Rst	CLK	S	R	Q	$\bar{Q}$
0	↑	0	0	NC	NC
0	↑	0	1	0	1
0	↑	1	0	1	0
0	↑	1	1	$\bar{Q}$	Q (Toggle)

```
library ieee;
use ieee.std_logic_1164.all;
entity ~~SRFF~~ JK is
port (Rst, clk, J,K : in bit;
        Q , Qbar : out bit);
end ~~SRFF~~ JK;

architecture rtl of JK is
begin
    process (clk, rst)
begin
    if rst = '1' then
        Q <= 0 , Qbar <= 1;
end ~~JK~~ begin.
        if rst = '1' then
            Q <= '0' , Qbar <= 1;
=>  else if clk and clk'event = '1' then,
    if S=0, R=0 then Q <= Q; Qbar <= Qbar;
=>  else if J=1, K=0 then
        Q <= '1', Qbar <= '0';
=>  else if J=0, K=1 then
        Q <= '0'; Qbar <= '1';
=>  else if J='1', K=1 then
        Q <= 'Qbar', Qbar <= 'Q then
    end if;
    end if;
end process;
end rtl;
```

TFF :-

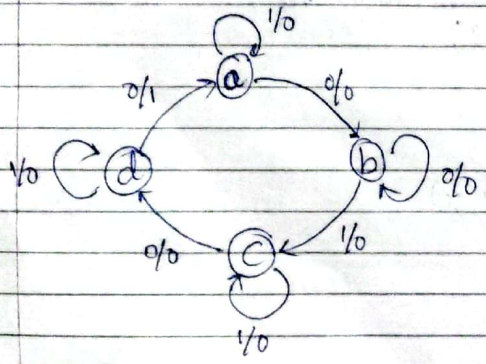
STATE DIAGRAM



Code:

```
library ieee;
use ieee.std_logic_1164.all;
entity ss is
port (x, clk, reset : in bit;
                  y : out bit);
end ss;

architecture rtl of ss is
begin
type state_type is (S1, S2, S3);
 signal ns, ps : state_type;
begin
 process (clk, reset)
begin
    if reset = '1' then
       PS <= S1;
else if (clk and clk event) then,
       PS <= ns;
end if
end process;

P2: process (Ps, x)
begin
case PS is
when S1 =>
  if x=0
  ns <= S1; y <='0';
else
   ns <= S2; y <='0';
end if;
```

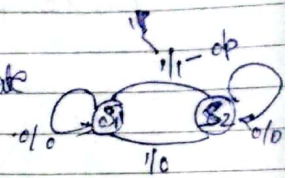
```
when S2 =>
 if x=0
   ns<=S2; y<='0';
 else
   ns<=S3; y<=0;
 end if;

when S3 =>
 if x=0
   ns<=S3;   y<='1';
 else
   ns<=S1;   y<='0';
 end if;
 end if;
 end process;
end sdl;
```




```
Library ieee;
use ieee.std_logic_1164.all;
entity ss is
port
```

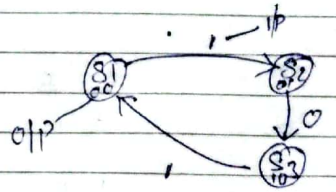
MEALY MACHINE :-

o/p depends on i/p & Present state

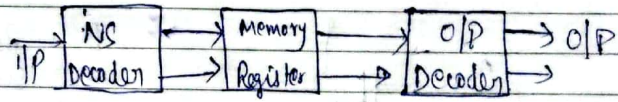


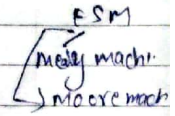
MOORE MACHINE :-

o/p depends on Present state.



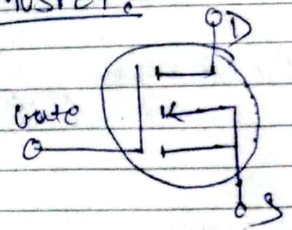
FSM BLOCK DIAGRAM :-



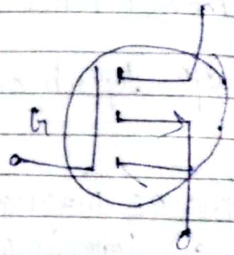


CHAPTER -3

MOSFET :-



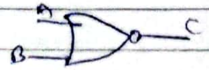
N channel



P-Channel

CHARACTERSTICS

NOR

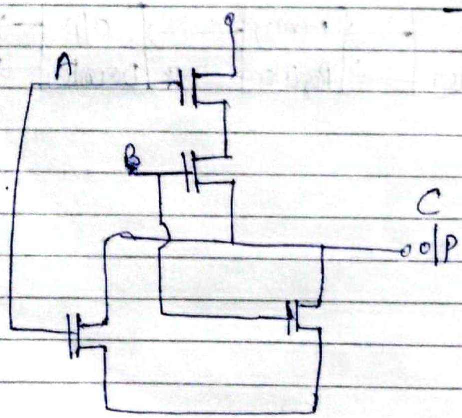


A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

$C = \overline{A+B}$

P switch = Series
N switch = Parallel

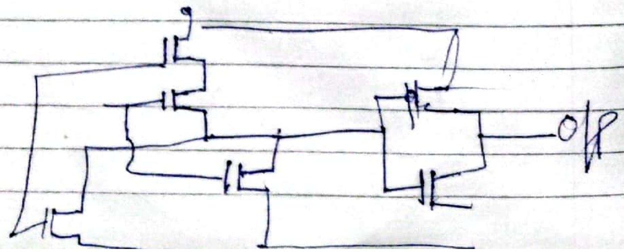
P = Parallel
N = Series



0 = T
1 = 0

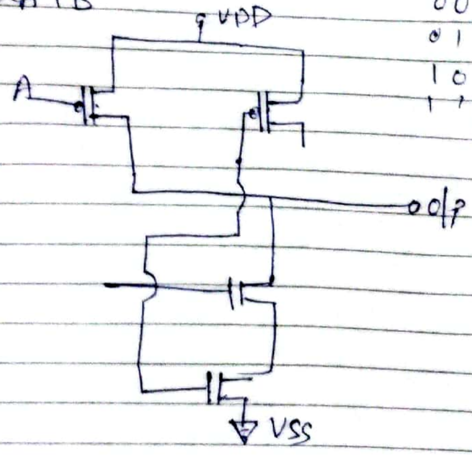
0 = clos
1 = open

OR



De morgan theorem :- NAND GATE

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



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