Exp. no. 01

Aim:- To design and simulate the layout of NMOS inverter.

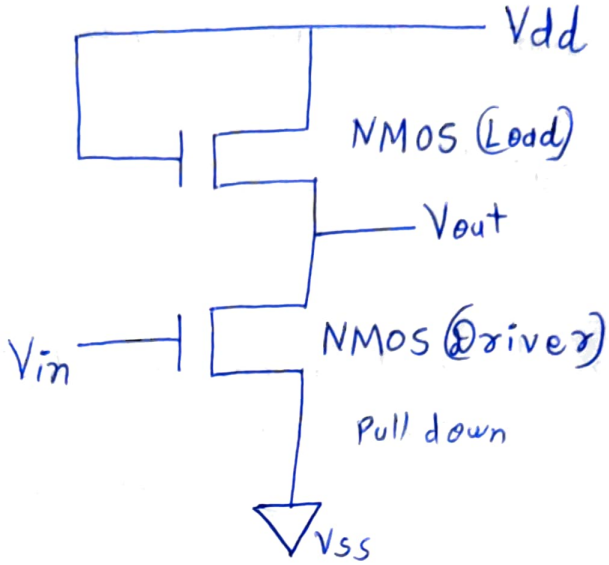
Software required :- Microwind

Procedure :-

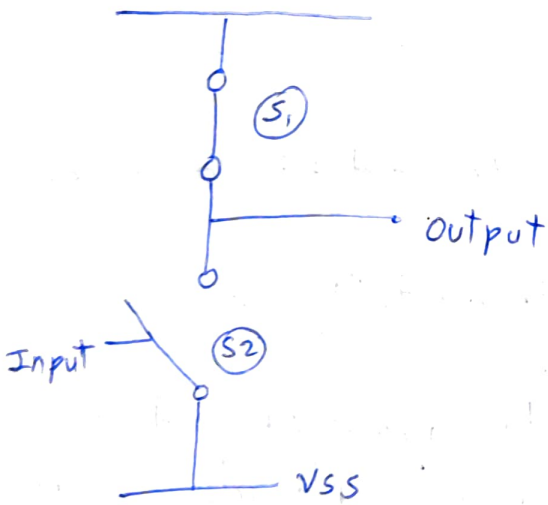
1. Select a new foundary by clicking on file, select foundry.
2. Fix the left corner to be $1\lambda = 0.06\mu m$.
3. Select the metal 1 layer from the ~~plo~~ plette to create metal layer of width 4λ.
4. Select the p+ diffusion intersecting metal layer down of width 4λ.
5. Draw a metal layer intersectiong the p+ diffusion.
6. Place the polysilicon layer for the gate on P+ diffusion as well as on N+ diffusion & connect both gate poly silicon.
7. Draw the metal layer at the other end of the n+ diffusion layer and assign a ground to it.
8. Give the required contacts such as metal to n+ diffusion, P+ diffusion to metal.
9. Assign the Vdd supply to the upper metal layer of P+ diffusion.
10. Assign the input clock pulse to the gate of amplitude equal to 0V to 5V.
11. Assign the visible node as the output.
12. Check for any errors using DRC.
13. Save and simulate to get the required output waveform.

Diagram :-

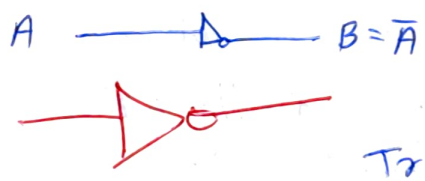
Circuit diagram



Switch Analogy

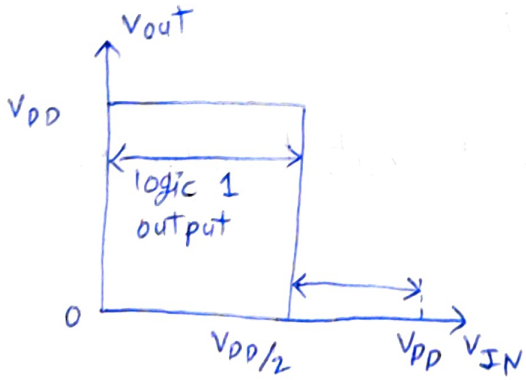


Symbol



Truth Table

A	B
0	1
1	0



VTC

Theory:-

The logic symbol and truth table of ideal inverter is shown in figure.

Using positive logic the Boolean value of logic 1 is represented by V_{DD} & logic 0 is represented by 0.

V_{th} is the inverter threshold voltage, which is $V_{DD}/2$ where V_{DD} is the out put voltage.

So for $0 < V_{in} < V_{th}$ output is equal to logic 0 input and $V_{th} < V_{in} < V_{DD}$ is equal to logic 1 input for inverter.

Case 1

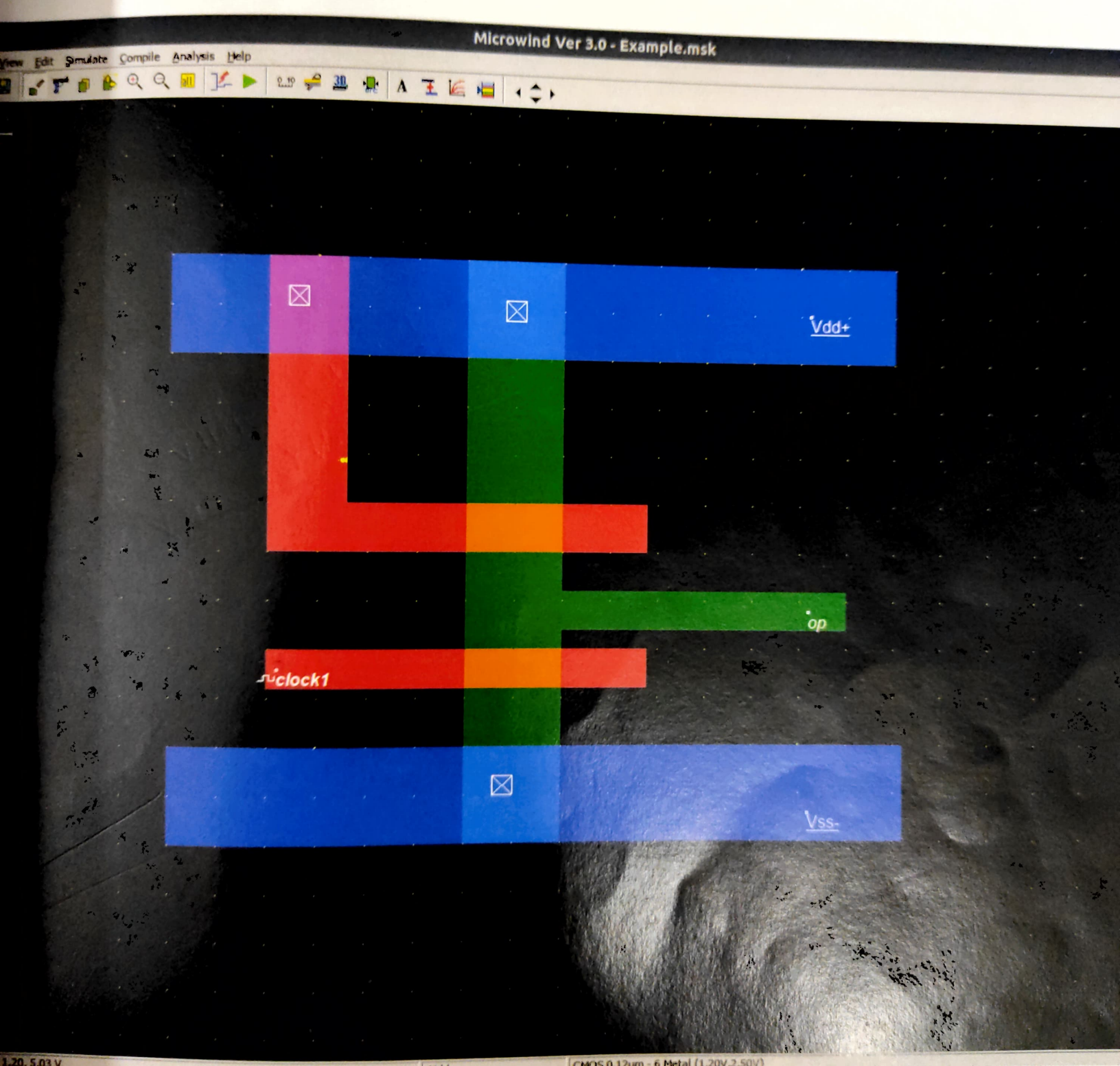
When logic '0' is applied to input, switch 'S_1' stays close & switch 'S_2' is open. Thus whole change flows towards, V_{DD}, thus the charge is pull up with logic at output end is '1'.

Case 2:-

When logic '1' is applied to input, switch 'S_1' stays close and switch 'S_2' is also closed. Thus the whole charge flows towards ground 'V_{SS}' thus the charge is pull down with logic at output end is '0'.

Result:-

Thus we have designed, implemented and simulated the layout of the NMOS inverter using Microwind. The output voltage does not reach 5 volt during logic high, because W/L ratio of pullup & pull down are not appropriatly set. The VTC is also not sharp which is a limitation.


Microwind Ver 3.0 - Example.msk
View Edit Simulate Compile Analysis Help
Vdd+
op
clock1
Vss-
1.20, 5.03 V
CMOS 0.12um - 6 Metal (1.20V,2.50V)

Exp. No. 02

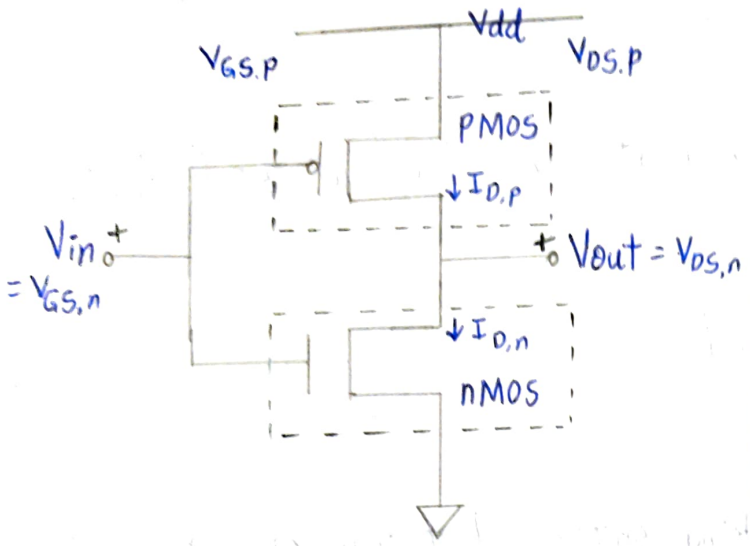
Aim :- To design and simulate the layout of CMOS inverter.

Software required :- Microwind

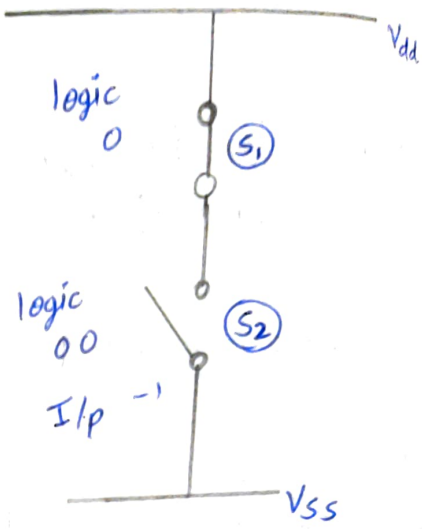
Procedure :-

(1) Select a new foundary by clicking on file, select foundry, CMOS 012.rul.
(2) Fix the left corner to be $1\lambda = 0.06\mu m$.
(3) Select the metal 1 layer from the palette to create metal layer of width 4λ.
(4) Select the p^+ diffusion intersecting metal layer down of width 4λ.
(5) Draw a metal layer intersectiong the p^+ diffusion.
(6) Now draw the n well of area $144\ \lambda^2$ around the p^+ diffusion area so that there will be min.
(7) Now draw the n^+ diffusion below the n well, such that there is at least 6λ seperation.
(8) Now connect the P^+ diffusion layers other end with the n^+ diffusion one end by metal layer and draw the metal layer outside which is output.
(9) Place the polysilicon layer for the gate on P^+ diffusion as well as on N^+ diffusion and connect both gates polysilicon.
(10) Draw the metal layer at the other end of the n^+ diffusion layer and assign a ground to it.
(11) Give the required contacts such as metal to n^+ diffusion, P^+ diffusion to metal.
(12) Assign the Vdd supply to the upper metal layer of P^+ diffusion.
(13) Give Vdd supply to the n well, because it is required not to leave it floating.

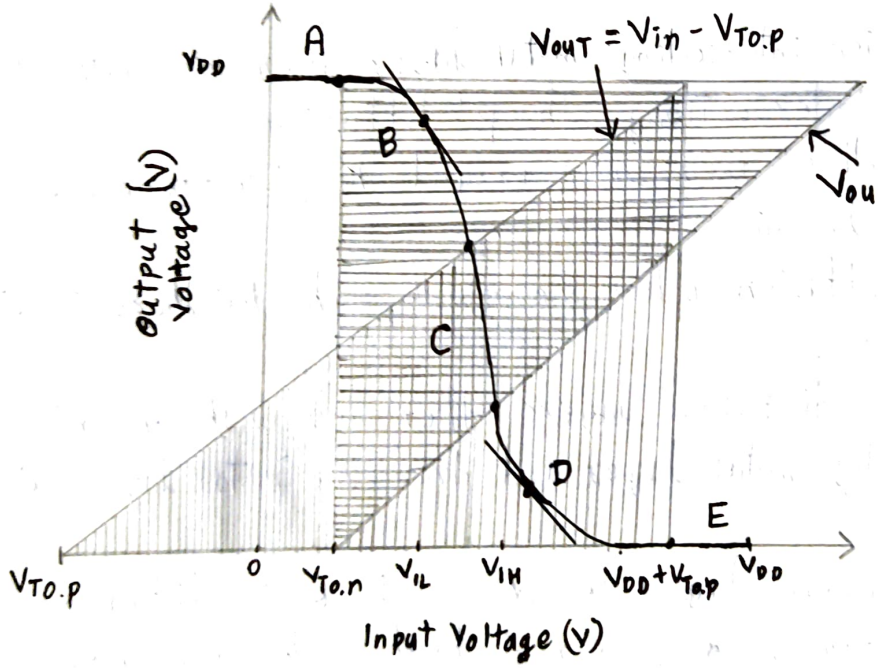
Circuit Diagram



Switch Analog



VTS

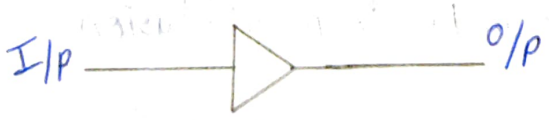


nMOS in satur

pmos in satur

both in saturat

Symbol



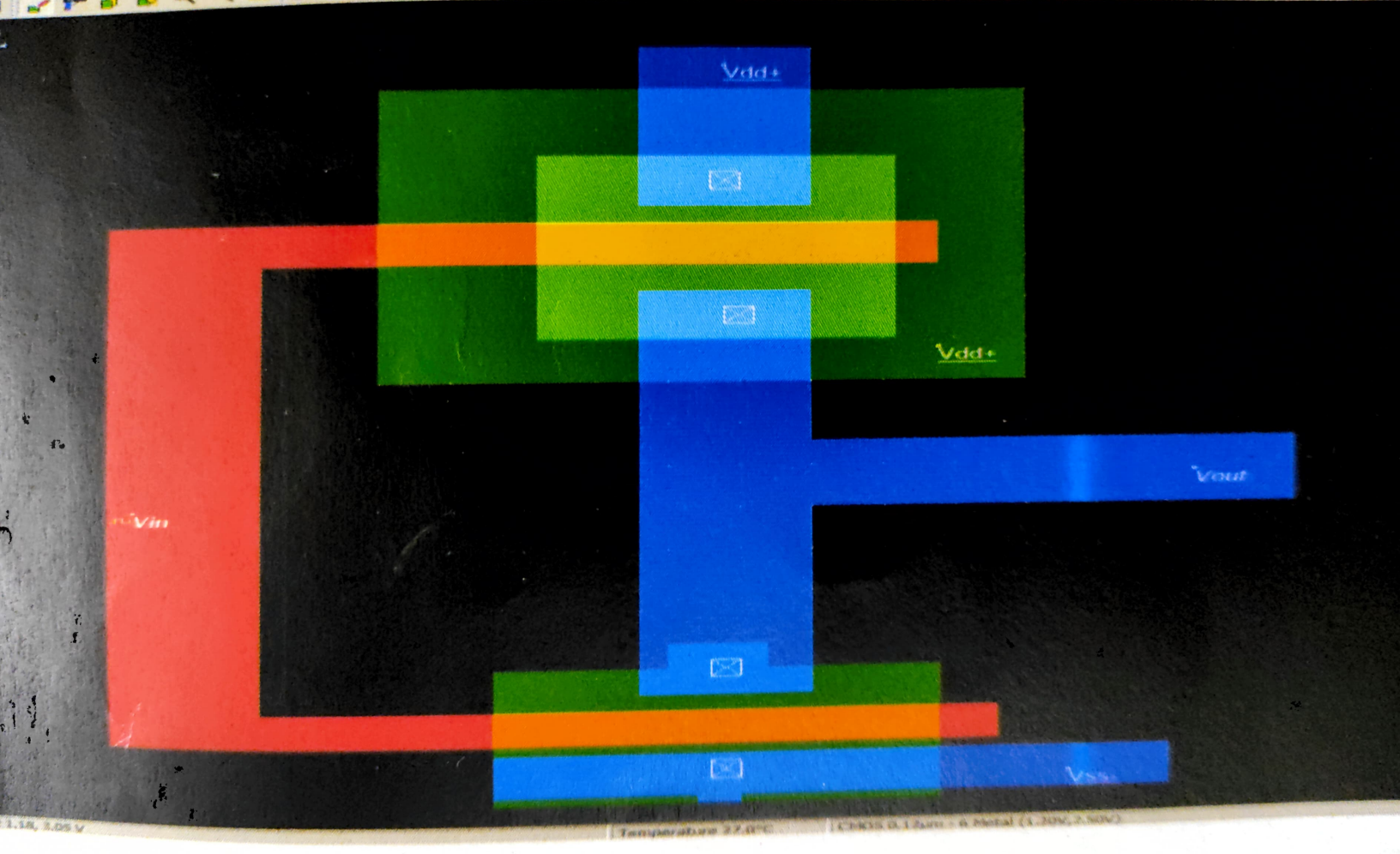
Truth Table

Input	Output
1	0
0	1

(14) Assign the input clock pulse to the gate of amplitude equal to 0v to 5V.
(15) Assign the visible node as the output to the CMOS at both the drains connected.
(16) Check for any errors using DRC.
(17) Save and simulate to get the requied output wave forms.

Theory:-
① When the low input voltage is given to the CMOS inverter, then the PNOS transister is switched ON, where as the NMOS transistor will switch OFF by allowly the flow of electrons throughout the gate terminal and generatly high logic output voltage.
② Similarly, when high input voltage is given to the CMOS inverter, the PMOS transistor is switches OFF where as the NMOS transistor will be switched ON avoidiy as many electrons from attaining the output voltage and generally low logic output.
Thus, direct current supply from the supply voltage (V_{DD}) to output voltage (V_{OUT}) and the load capacitor (C+) can be charged & shows that $V_{OUT} = V_{DD}$

Result:- Thus we have designed, implemented & simulated the layout of the CNOS inverter using microwind. The output voltage does not reach 5volts during logic high because W/L Ratio of Pull up & pull down are not appropriatly set. The VTC is also shown up. as compared to NMOS inverter.


Vdd
Vdd
Vout
Vin
Vss

Exp. 03 & 04

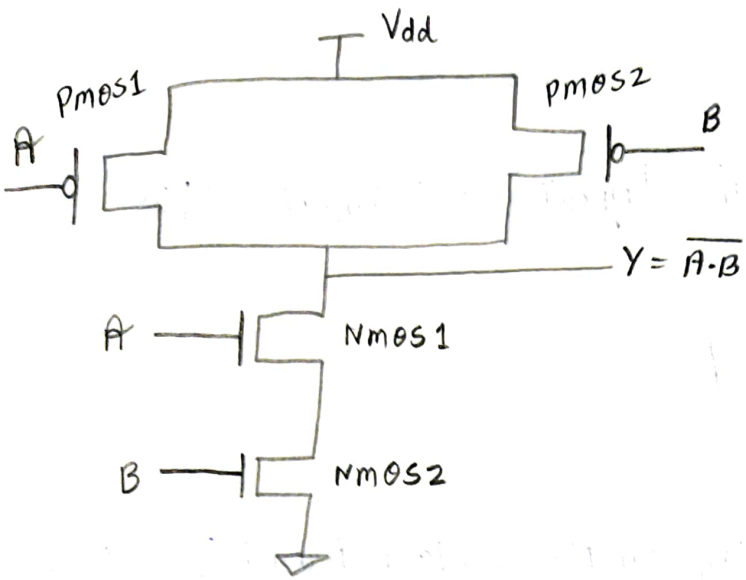
Aim :- To design and simulate the layout of 2 input CMOS NAND and NOR Gate.

Software required- Microwind

Procedure :-

1) Select a new Foundary by clicking on fire, select foundry, CMOS 012.rul.
2) Fix the left corner to be $1\lambda = 0.06\ \mu M$.
3) Select the Metal layer from the Palette to create layer of width 4λ.
4) Select the P+ diffusion intersecting metal layer down of width 4λ such that for NAND gate there will be two parrell P+ diffusions for PMOS or single P+ diffusion sufficient for two PMOS for NOR gate.
5) Draw a metal layer intersecting the p+ diffusion for Vdd supply.
6) Now draw the n well of area $144\lambda^2$ around the P+ diffusion area so that there will be min 6λ saperation P+ between P+ diffusion & n well boundry.
7) Now draw the n+ diffusion below the n well, such that there is at least 6λ saperation: NAND gate take single n+ diffusion for two nmos or two Parrallel diffusion for NOR gate.
8) Now connect the P+ diffusion layer other end with the n+ diffusion are one end by metal layer and draw the metal layer outside which is output.
9) Place the Polysilicon layer for the gate on P+ diffusion as well as on N+ diffusion & connect both gates polysilicon.
10) Draw the metal layer at the other end of the n+ diffusion layer & assign a ground to it.

CIRCUIT DIAGRAM:- NAND Gate

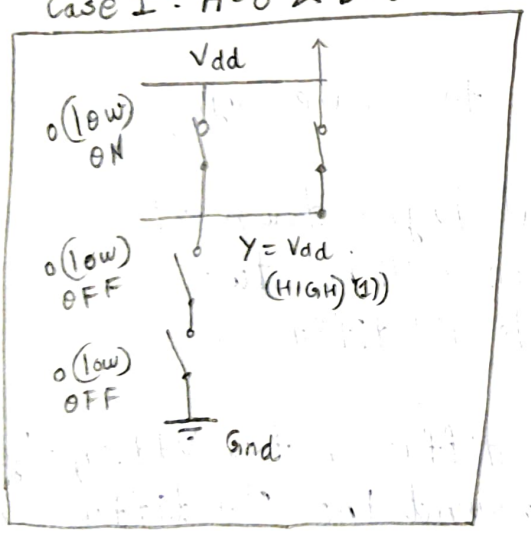


NAND Gate

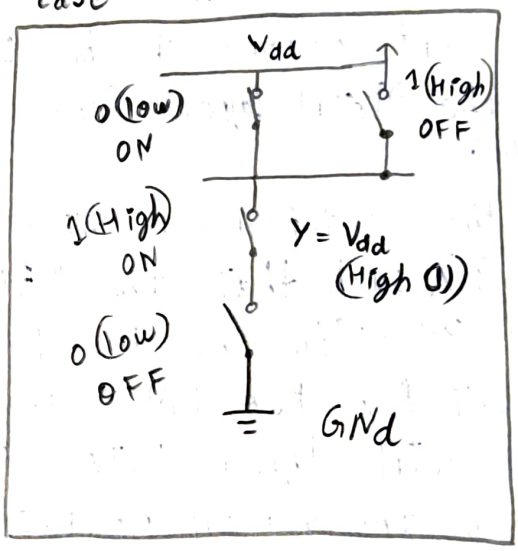
Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

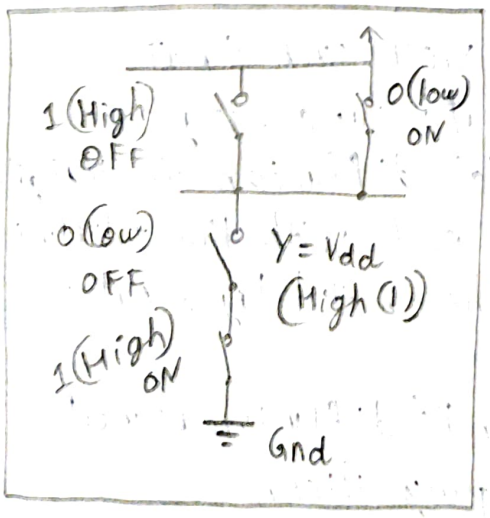
Case I : A = 0 & B = 0



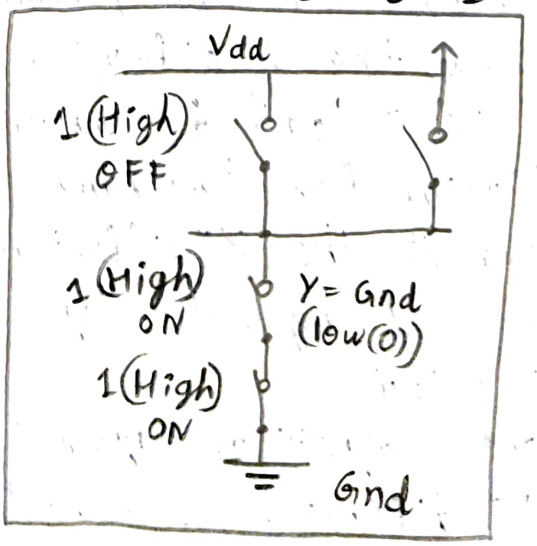
Case II : A = 0 & B = 1



Case III : A = 1 & B = 0



Case IV : A = 1 & B = 1



11] Assign the Vdd supply to the upper metal layer of P+ diffusion.

12] Give Vdd supply to the upper n well, because it is required not to leave it floating.

13] Give the required contacts such as metal ton + diffusion, P+ diffusion to metal.

14] Assign the two input clock pulse to the required gates of the required Moss of amplitude equal to 0V to 5V.

15] Assign the visible node as the output to the output metal drown outside.

16] Check for any error using DRC.

17] Save and simulate to get the required output waveform.

Theory :- NAND Gate

To understand how this circuit will be have like NAND gate, circuit output follow the same as in truth table for different input combinations.

Case 1 :- A = Low & B = low , Y = High

1) Both the Pmos will be ON and both NMOS will be OFF.
2) The output Vout will get two paths through two ON Pmos to get connected with Vdd.
3) There is no Path through which the output line can discharge.
4) The output line will maintain the voltage level at Vdd so high (logic 1)

Case 2 - A = Low & B = High , Y = High

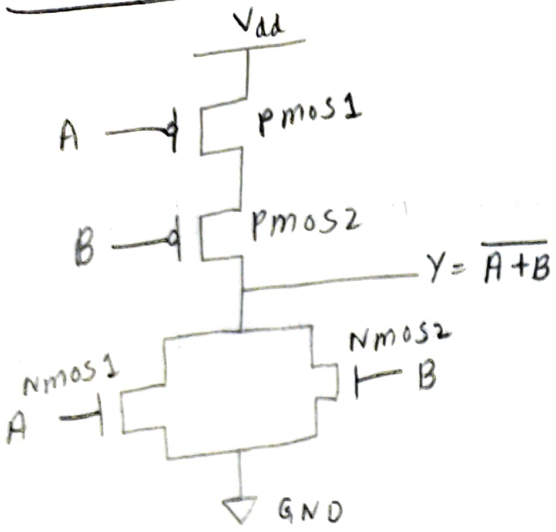
1) A = Low , Pmos 1 = ON, Nmos 1 = OFF
   B = High , Pmos 2 = OFF, Nmos 2 = ON
2) Pmos 1 & Pmos 2 are in Parallel, NMOS 1 & Nmos 2 are in series.

3) The output line will get a path through Pmos2 to get connected with Vdd.
4) The V_{OUT} is maintained at the level of Vdd as Nmos1 is OFF, so High (Logic 1)

Case 3 - A = High & B = low, Y = High
1) A = High, P-mos1 = OFF, Nmos1 = ON
B = Low, Pmos2 = ON, Nmos2 = OFF
2) Pmos1 & Pmos2 are in paralled, Nmos1 & Nmos2 are in sereies.
3) The output line will get a path through Pmos2 to get connected with Vdd.
4) The V_{OUT} is maintained at the level of Vdd as Nmos1 is OFF, so High (Logic 1).

Case 4 - A = High & B = High, Y = Low.
1) A = High, Pmos1 = OFF, Nmos1 = ON.
B = High, Pmos2 = OFF, Nmos2 = ON.
2) Both Pmos are OFF, so V_{OUT} will not find any path to get connected with Vdd.
3) Both Nmos are ON is series, so it will create a path from V_{OUT} to GNd.
4) Since, Path to ground is estabilised, V_{OUT} will be discharged so low. (Logic 0)

CIRCUIT Diagram : NOR gate

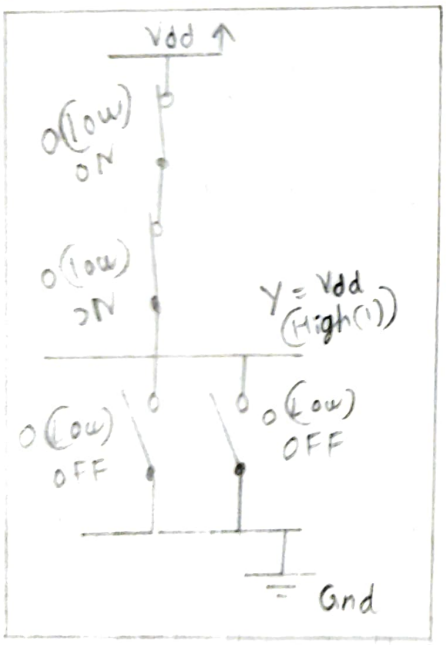


NOR GATE

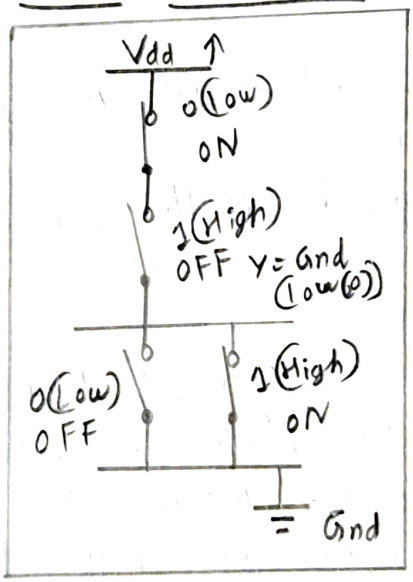
TRUTH TABLE :-

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

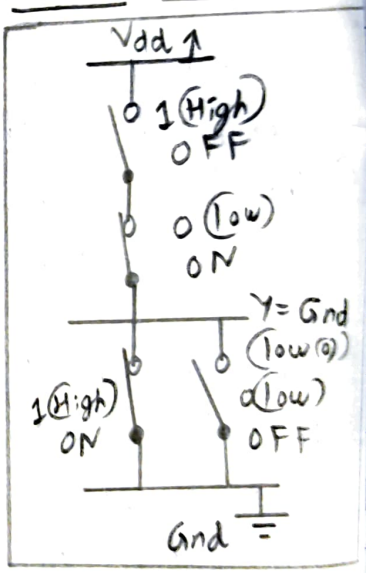
Case I : A = 0 & B = 0



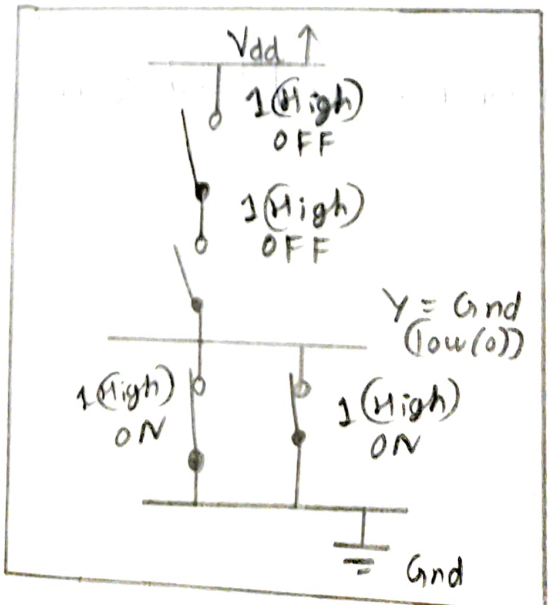
Case II : A = 0 & B = 1



Case III : A = 1 & B = 0



Case IV : A = 1 & B = 1



NOR GATE.

Case 1- A = low & B = low, Y - High

1) As A & B both are low both PMOS will be 'ON' and both NMOS will be 'OFF'.
2) Both PMOS get connected to Vdd & As both NMOS are OFF, V_{OUT} will not be able to find a path to GND to get discharged.
3) This twins V_{OUT} to be maintained as a level of Vdd; So logic High (logic 1).

Case 2- A = low & B = High, Y = High.

1) A- low - PMOS1 = ON, NMOS 1 = OFF
   B - High - PMOS2 = OFF, NMOS2 = ON
2) As PMOS2 is OFF the Vdd will not be able to find Path to pass charge towards V_{OUT}.
3) This twins V_{OUT} to be maintained as a level of Vdd; so logic High (logic 1).

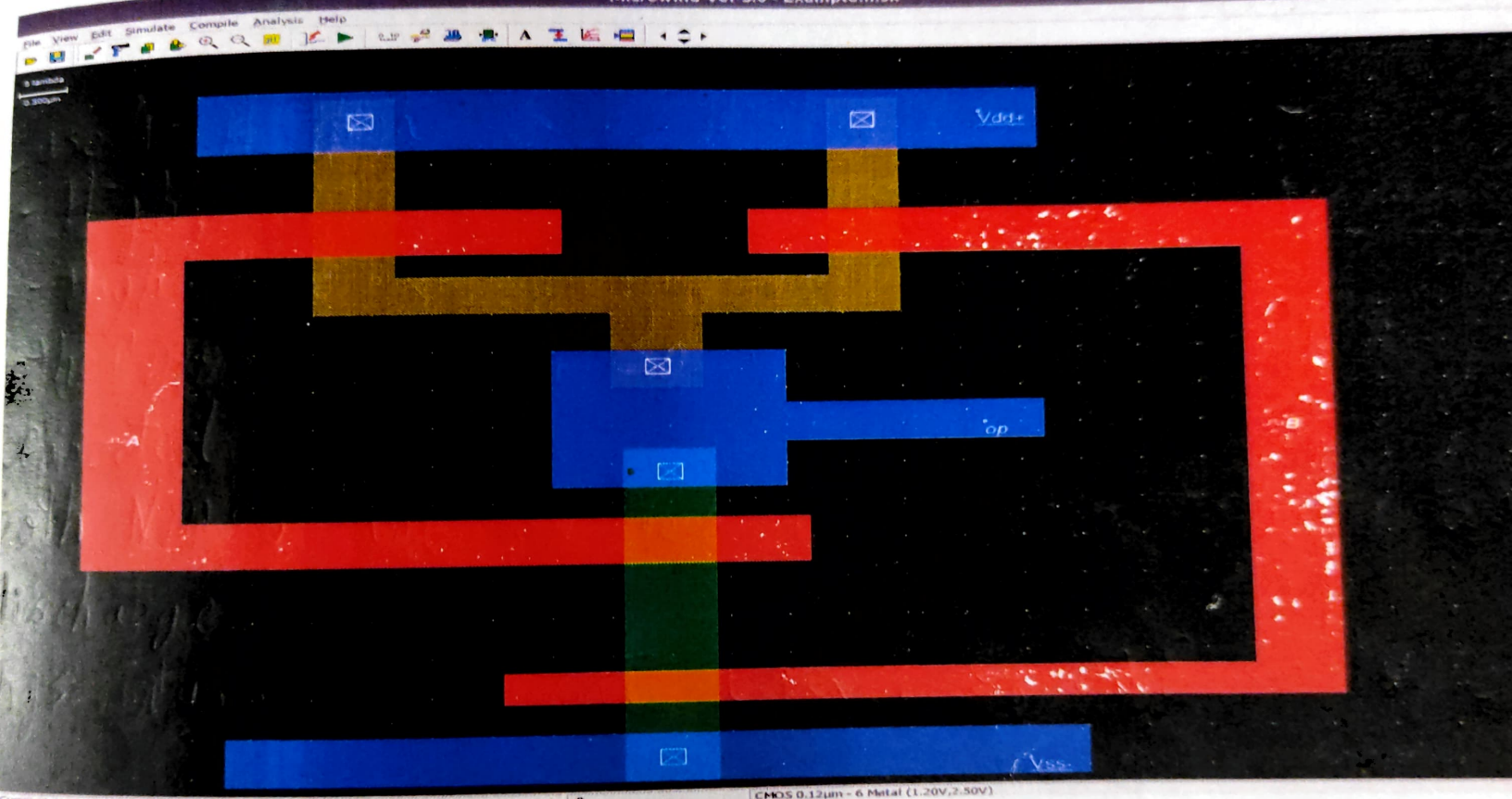
Case -3 - A = High & B - low, Y = low.

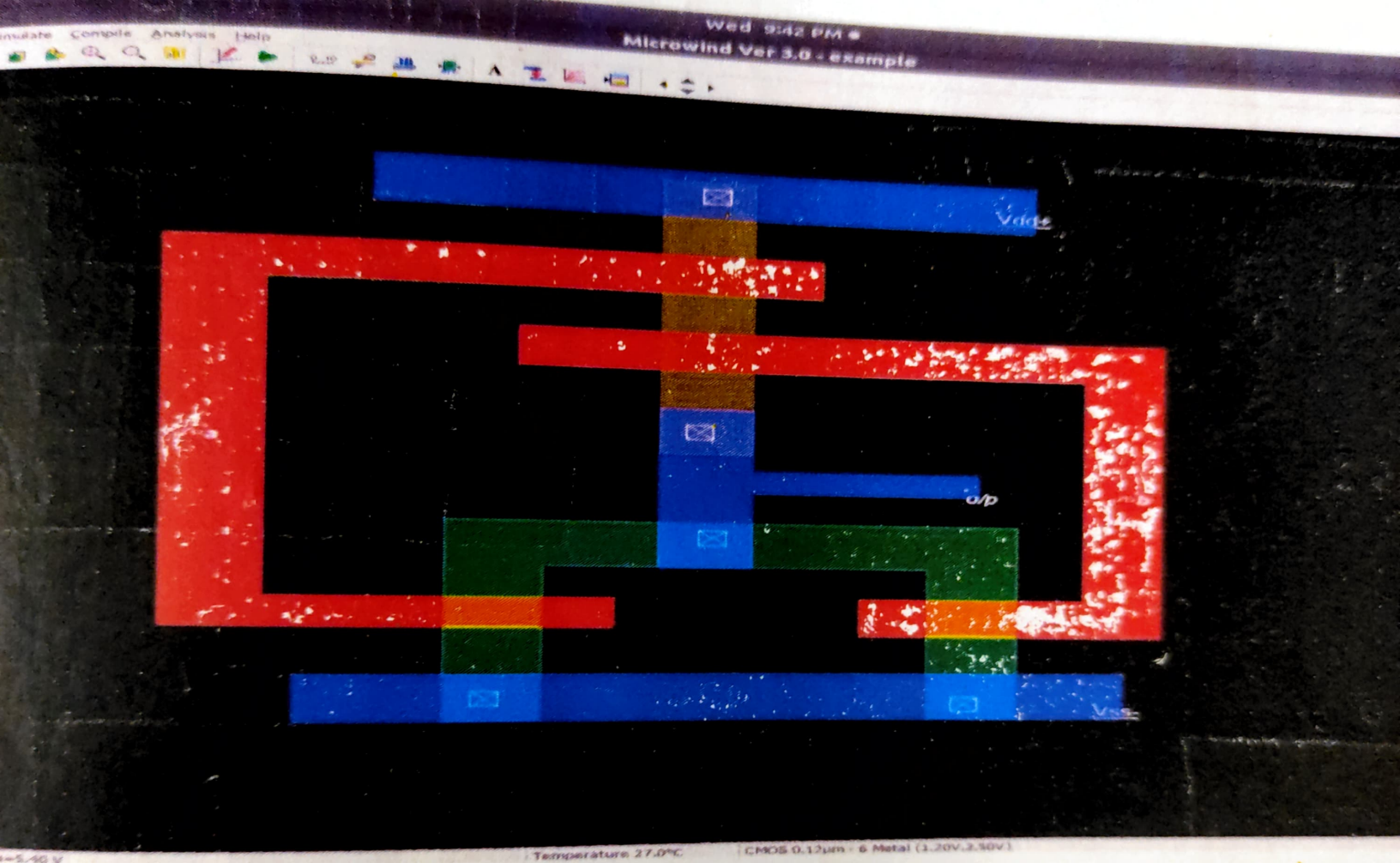
1) As A - High = PMOS 1 = OFF, NMOS 1 = ON
   B - low = PMOS2 = OFF, NMOS2 = ON
2) PMOS1 is OFF Vdd doesnot have a path to transfer charge further.
3) NMOS1 is ON thus V_{OUT} has a Path toward GND.
4) Thus turn V_{OUT} at a level of GND so logic low (logic 0).

Case 4 - A = High & B = High, Y = low.

1) A - High, PMOS 1 = OFF, NMOS 1 = ON.
   B - High, PMOS 2 = OFF, NMOS 2 = ON.
2) Both PMOS are OFF so V_{OUT} will not find any path to get connected with V_{dd}.
3) Both NMOS are ON. V_{OUT} finds then path to GND to get discharge.
4) This turns V_{OUT} at a level of GND, so logic low (logic 0).

Result:- Thus in this way we have studied, designed, implemented & simulated the layout of NAND & NOR gate using CMOS.


File View Edit Simulate Compile Analysis Help
Vdd+
op
A
B
Vss-
Cursor at time 19.98ns , A=1.14 V
CMOS 0.12µm - 6 Metal (1.20V,2.50V)


Wed 9:42 PM
Microwind Ver 3.0 - example
Vdde
o/p
Vss
Temperature 27.0°C
CMOS 0.12µm - 6 Metal (1.20V,2.50V)

Exp no.-05

Aim:- To design and simulate the layout of the Boolean function $F = \overline{A(B+C)}$ using CMOS.

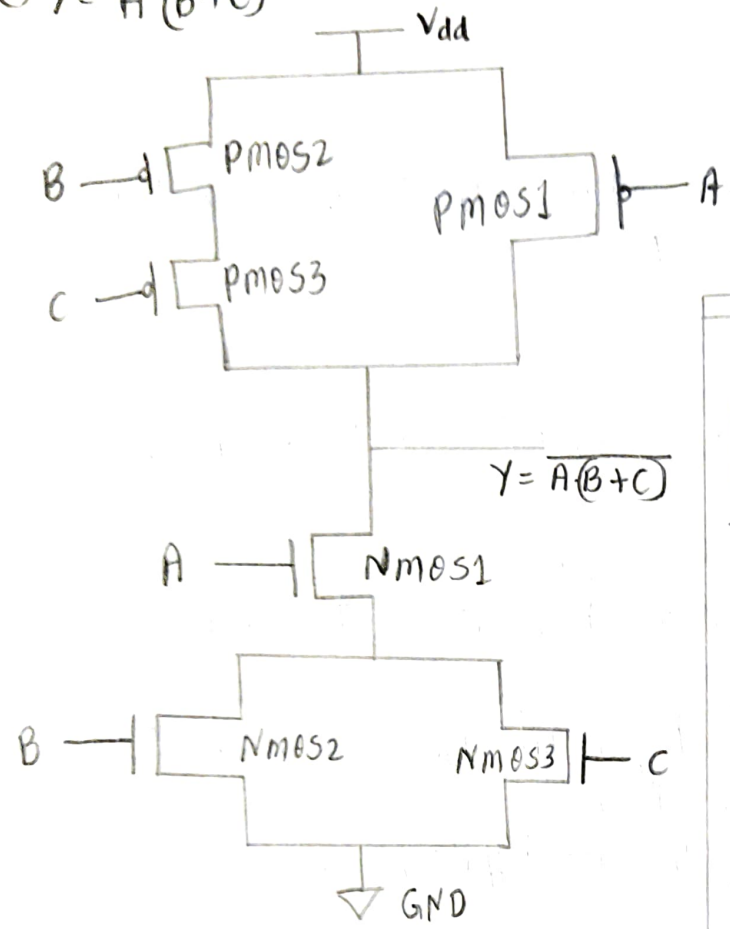
Software Required:- Microwind.

Procedure:-

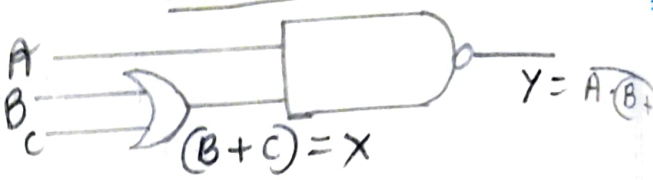
1. Select a new foundary by clicking on file, select foundary, CMOS 012 rul.
2. Fix the left corner to be $1\lambda = 0.06\mu m$.
3. Select metal 1 layer from the palette to create metal layer of width 4λ.
4. Select the P+ diffusion intersecting metal layer down of width 4λ such that to make the required PMOS.
5. Draw a metal layer intersecting the P+ diffusion which is for Vdd supply.
6. Draw n well of area $144\lambda^2$ around the p+ diffusion area so that there will be minimum 6λ seperation between it diffusion and nwell boundary.
7. Now draw n+ diffusion below the n well, such that there is atleast 6λ ~~sep~~ seperation for the required NMOS's.
8. Now connect the P+ diffusion layers other end with the n+ diffusion one end by metal layer and draw the metal layer outside which is output.
9. Place the polysilicon layer for the gate on P+ diffusion of the required PMOS as well as on N+ diffusion for the required NMOS's and connect gates polysilicon as per requirement.
10. Draw the metal layer at the other end of the n+ diffusion layer and assign a ground to it.
11. Give required contacts such as metal to ~~not~~ n+ diffusion, P+ diffusion to metal.

CIRCUIT DIAGRAM

(i) $Y = \overline{A(B+C)}$



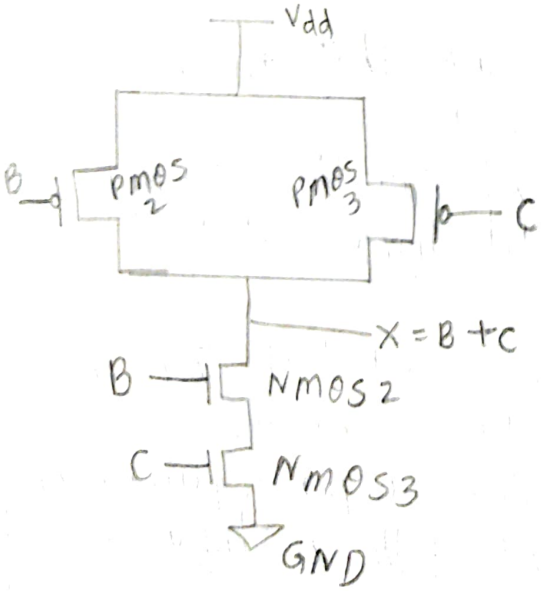
LOGIC GATE



TRUTH TABLE: (i) FOR NAND GATE

A	B	C	(B+C) = X	$Y = \overline{A \cdot (B+C)}$
0	0	0	0	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

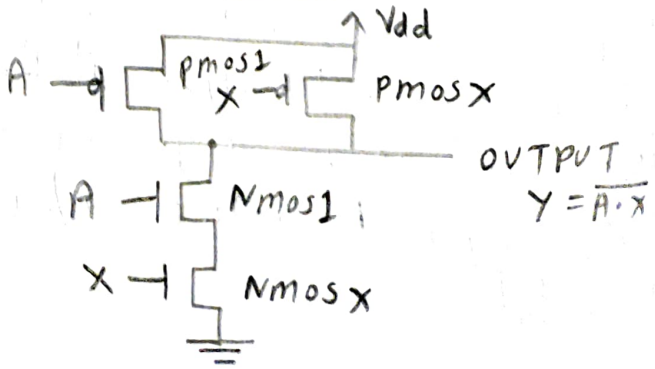
(ii) OR Gate



(i) FOR OR GATE

B	C	X = (B+C)
0	0	0
0	1	1
1	0	1
1	1	1

(ii) NAND GATE :-



(12) Assign the Vdd supply to the upper metal layer of P+ diffusion.
(13) Give Vdd supply to the nwell, because it is required not to leave it floating.
(14) Assign the input clock pulses to the required gate of the repuired MOS; of amplitude equal to 0V to 5V.
(15) Assign the visible note as the output to the output metal drown outside.
(16) Check for any error using DRC.
(17) Save and simulate to get the requited output waveform.

Theory:-

⇒ As we could see in logic gate diagram and truth table that with the use of OR gate giving inputs 'B' and 'C' we have got output of OR gate as 'X' and then 'A' and 'X' are given as inputs in NAND gate to get 'Y' as output.

(A) OR GATE

Case I : B-low & C-low -- X-low

⇒ As 'B' and 'C' both are low thus, both the PMOS will be OFF and both NMOS will be ON, In this case V_{OUT} will not be able to find any path to get connected between V_{OUT} to GND. Thus, V_{OUT} will be discharged, so low [logic 0].

Case II:- B-low & C-High ....X-high.

⇒ B-low: NMOS2 OFF, PMOS2-ON, C-High: NMOS3-ON, Pmos3-OFF Although PMOS3 is OFF, output will still get a path through PMOS2 to get connected with Vdd. Nmos2 and NMOS3 are in series, as NMOS2 is OFF so output will not be able to find path to GND to get dischared. So, High [logic 1].

Case III :- B - High & C - low ..... X - High

⇒ B - High : NMOS2 - ON, PMOS 2 - OFF, C - low : NMOS3 - OFF, PMOS3 - ON, the explaination is similar to Case II, output level will be High [logic 1]

Case IV :- B - high & c - high ..... X - High

⇒ As 'B' and 'C' both are high; thus both the PMOS will be ON and both the NMOS will be OFF. In this case V_{OUT} gets the path to connect with Vdd, And as both NMOS are OFF. V_{OUT} ~~does~~ doesnot get path to connect with GND to get discharged, so High [logic 1].

(5) NAND GATE.

Case I : A - low & X - low .. Y - High.

⇒ As 'A' and 'X' both are low; both the PMOS will be ON and both the NMOS will be OFF. So output (V_{OUT}) will get two paths though two ON PMOS, to get connected with Vdd. Thus output will be charged to Vdd level. The output line will not get any path to the GND and both the NMOS are OFF. So there is no path through which the output line can discharge. The output line will ~~=~~ maintain voltage level at Vdd. So High (logic 1).

Case II : A - low & X - High Y - High

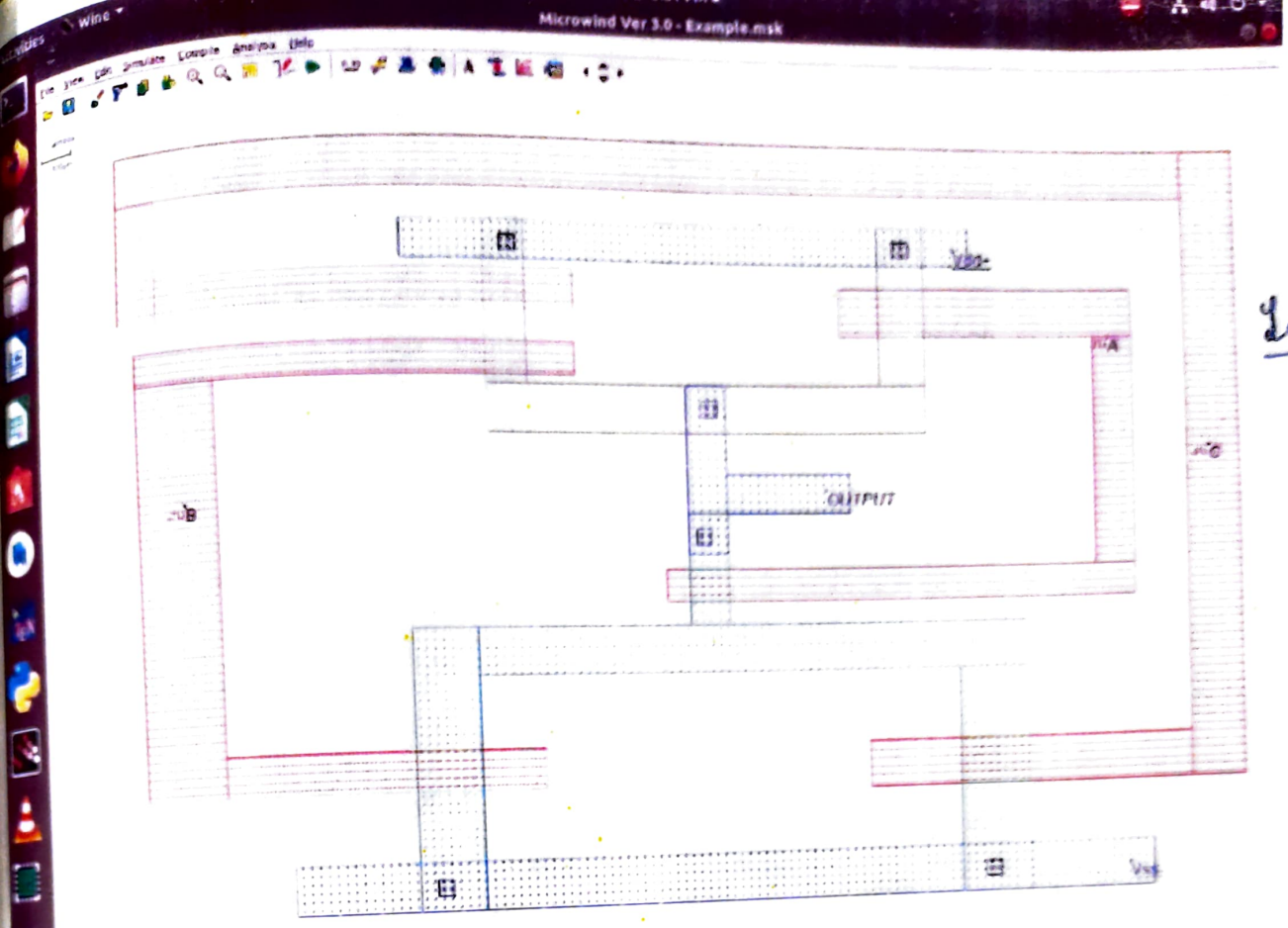
⇒ A-low : PMOS 1 - ON, NMOS 1 - OFF ; B - High : PMOS X - OFF, NMOSx ON. PMOS 1 and PMOS X are in parelled. Though PMOS X OFF, still output line will get a path through PMOS1 to get connected with Vdd. NMOS1 & NMOSx are in series as NMOS1 is OFF, so V_{OUT} will not be able to find a path to GND to get discharged. This in turn results

the V_{OUT} to be maintained as a level of V_{dd}. So High [logic 1]

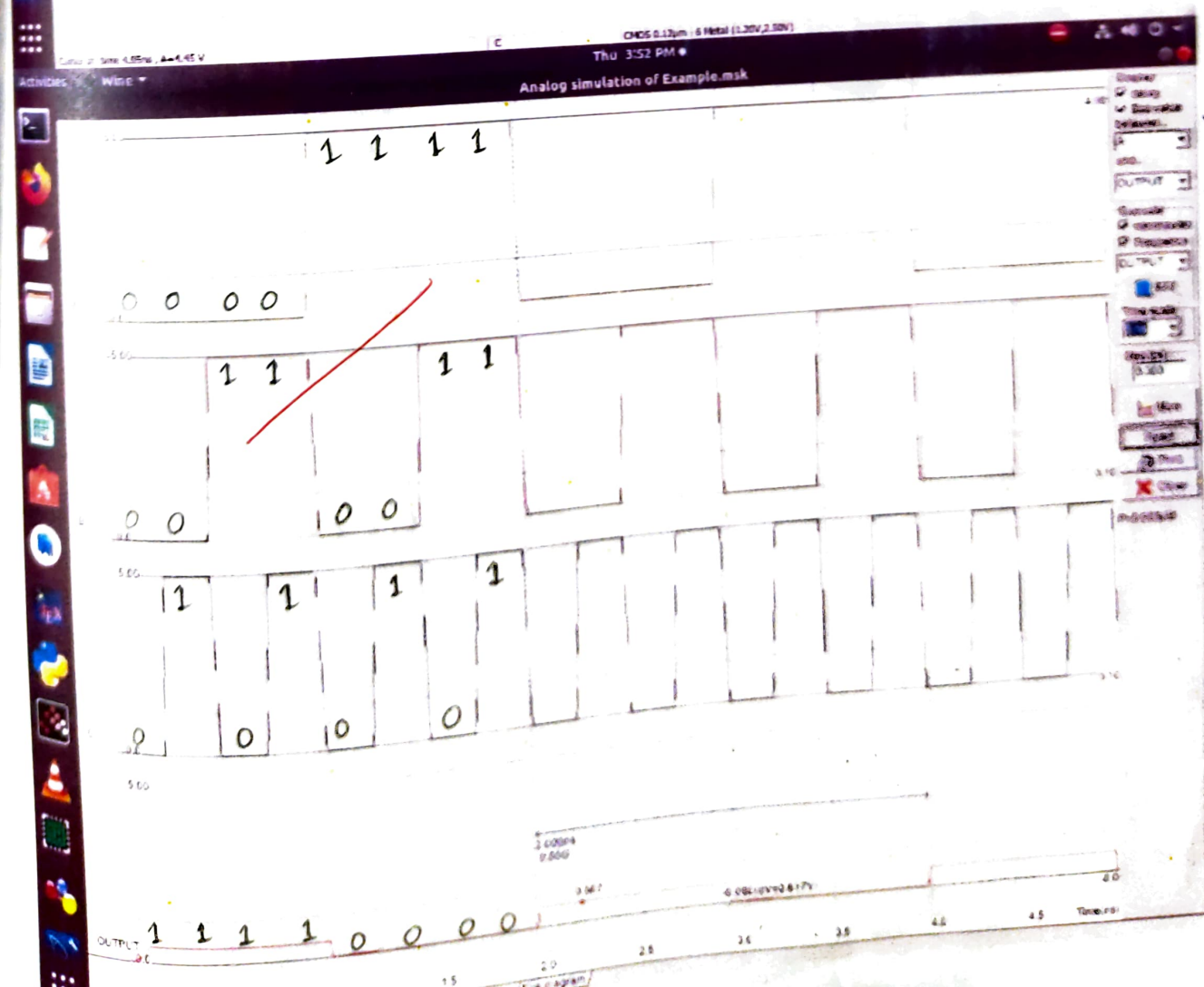
Case III :- A - High & X - low ..... Y - High.
A-High: PMOS1 - OFF, NMOS1 - ON; X-low : PMOS X - ON, NMOS X - OFF
The explaintain is similar to case II, V_{OUT} level will be high [logic 1]

Case IV :- A - High & X - High ... Y - low
A - High : PMOS 1 - OFF, NMOS 1 - ON; X - High : PMOS X - OFF, NMOS X - OFF. In this case both the PNOS are OFF, So V_{OUT} will not find any path to get connected with V_{dd}. As both NMOS are ON, the series connection of NMOS will create a path from V_{OUT} to GND. Since the path to ground is established V_{OUT} will be discharged. So low [logic 0].

Result:- Thus, in this way we have studied, designed, ~~implemented~~ implemented and simulated the layout of the Boolean expression using CMOS.



Layout



Voltage vs Time

Exp - 06

Aim:- To write a verilog program for basic logic gates to synthesize and simulate using xilinx.

Software :- Xilinx ISE Design Suite

Algorithm:-

1) Start the program.
2) Declare the input and output variables.
3) Declare the output as register data type.
4) Use PROCEDURAL construct statement for verilog code.
5) Write the functionality of the gates.
6) Terminal the program.

Theory - AND GATE in verilog is expresed as y = a & b
OR GATE in verilog is expressed as y = a | b
NOT GATE in verilog is expressed as y = ~a
NAND GATE in verilog is expressed as y = ~(a & b)
NOR GATE in verilog is expressed as y = ~(a | b)

Program:-

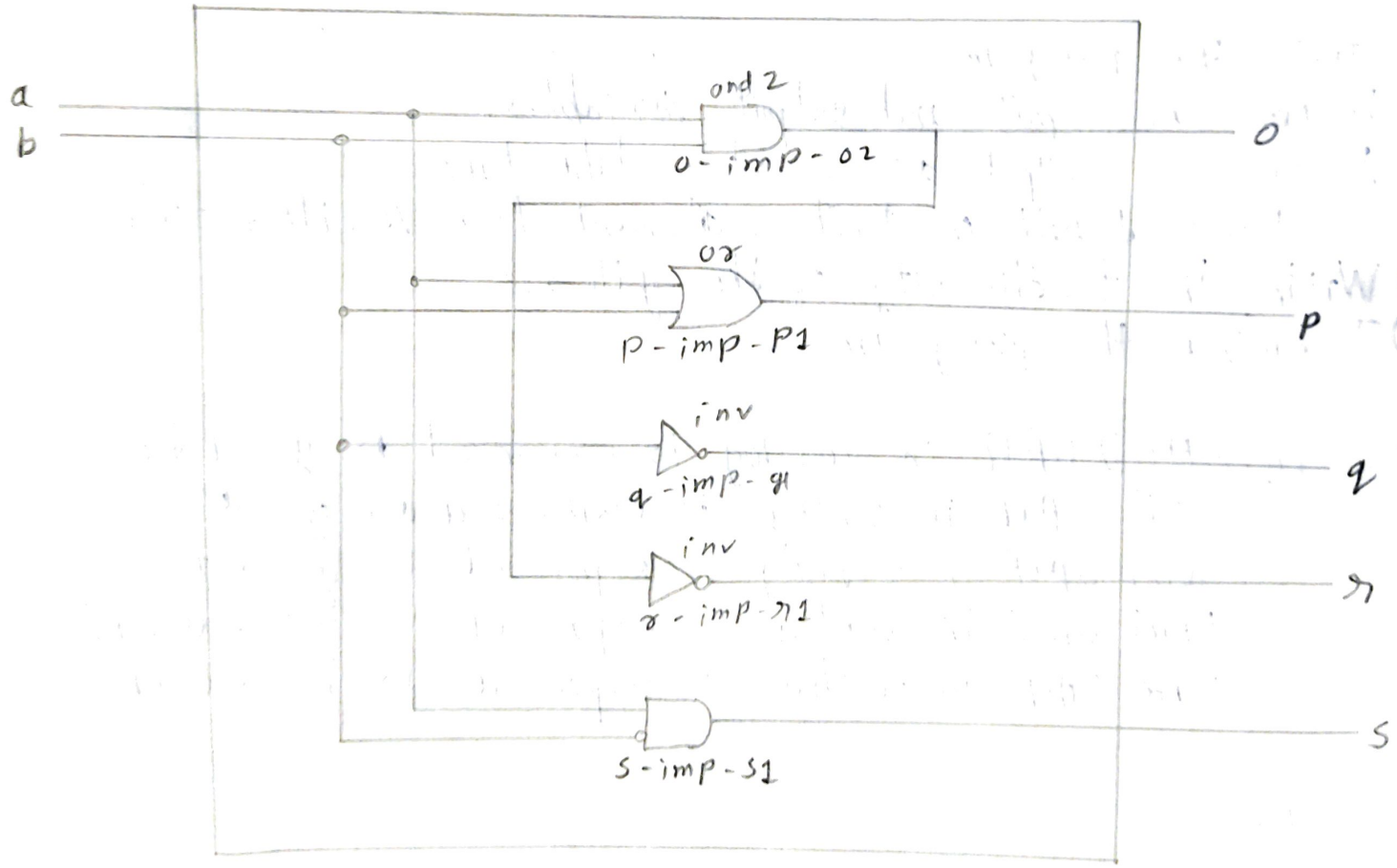
```
Module logic gates (a, b, o, p, q, r, s);
input a, b;
output o, p, q, r, s;
and (o, a, b);
or (p, a, b);
not (q, a);
```

PROGRAM :-

```
Module allgate (
  input a,b,
  output o,p,q,r,s
);
assign o = a & b; // and gate
assign p = a|b; // or gate
assign q = ~a ; // not gate
assign r = ~(a&b); // nand gate
assign s = ~a ^b; // not gate
end module
```

Diagram:-

All logic gates



```
nand (u, a, b);
nor (3, a, b);
end module
```

Procedure :-

1. Click on the xilinx ISE Design Suite or xilinx project Navigator icon on the ~~dest~~ desktop.
2. Write the verilog code by chosing HDL as top level, source Module.
3. Check Syntax, view RTL Schematic and note the device utilixation Summary by double clicking on the Synthesis in the process window.
4. Perform the functional Simulation using xilinx ISE Simulator.
5. The output waveform can be Observed in Model Sine.

Result:- Thus the verilog program for basic logic gates were written synthesized and simulated using Xilinx.

Exp. - 07

Aim:- To write a verilog program for full adder to synthesize and simulate using Xilinx software tool.

Tool Required:- Xilinx ISE Design Suite software.

Algorithm:- ① Start the program.
② Declare the input and output variables
③ Declare the output as register data type.
④ Use PROCEDURAL construct statement for verilog code.
⑤ Termintе the program.

Theory:-

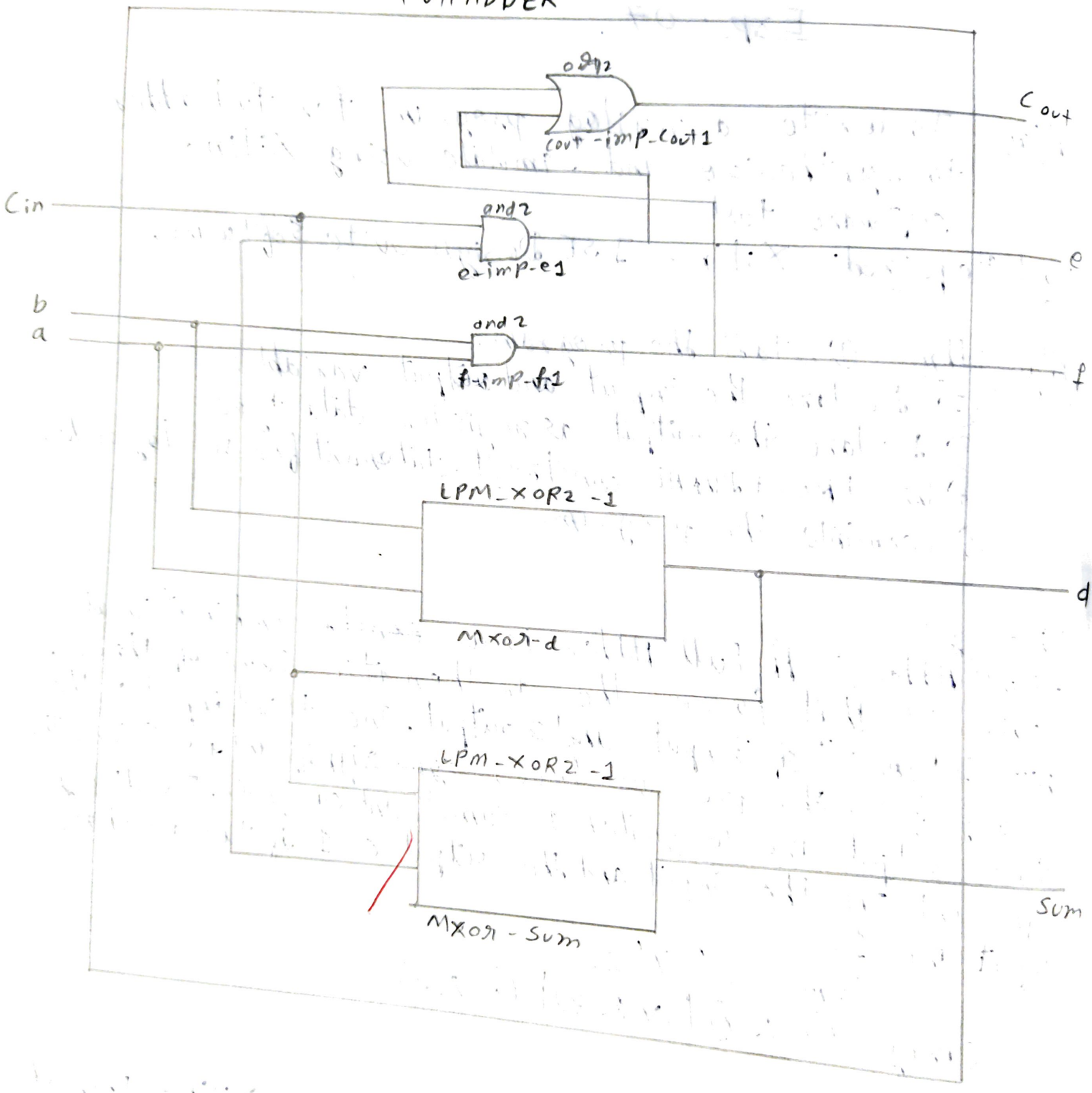
· Full Adder.: A Full Adder is a ~~consta~~ combinational circuit that foceses the arithematic sums of three bits. It consist of 3 input and 2 output. The third input is the carry from the previous lower ~~Syn~~ significant position. The two outputs are designated as sum (S) and carry (C). The dinary present in the input and the output C=1 if two are three input are 1.

$$Sum = x \hat{} y \hat{} z$$

$$Carry = (x \& y) | (y \& z) | (x \& z)$$

Procedure:-
① Click on the Xilinx ISE Design suite or Xilinx project Navigator icon on the desktop.
② Write the verilog code by choosing HDL as top level source module.

FULL ADDER


or2p2
Cout
cout_imp_cout1
Cin
and2
e_imp_e1
e
b
a
and2
f_imp_f1
f
LPM_XOR2_1
MXOR_d
d
LPM_XOR2_1
MXOR_Sum
Sum

③ Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
④ Perform the functional simulation using Xilinx ISE simulator.
⑤ Output can be observed by using model sim.

Program:-

Verilog code For Full Adder:-

```
module FULLADDER (
input a,
input b,
input Cin,
output d,
output e,
output f,
output sum,
output cout,
);
assign d = a^b;
assign e = d & cin;
assign f = a & b;
assign Sum = d ^ cin;
assign Cout = e | f;
end module.
```

Result:- Thus, verilog program for fulladder was written, synthesized & simulated using Xilinx tool.

Exp.- 08

Aim:- To write a verilog program for multiplexer to synthesize and simulate using Xilinx software tools.

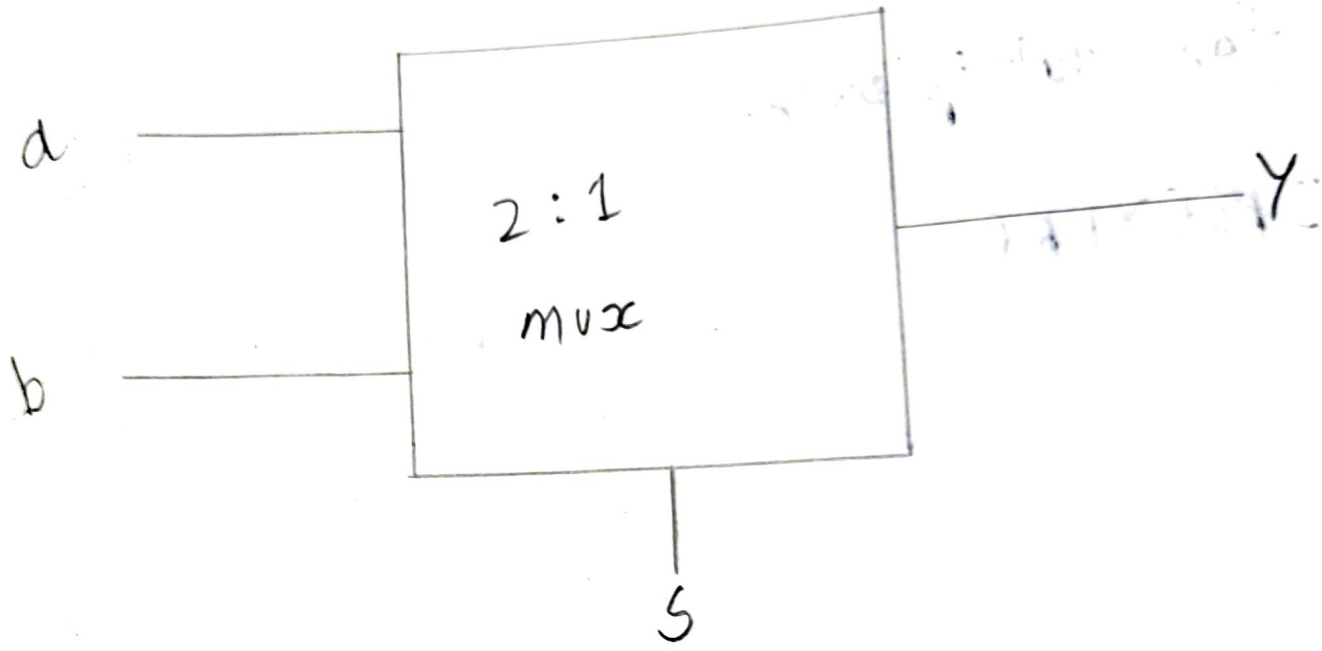
Tools required:- Xilinx ISE Design Suite 12.1 software

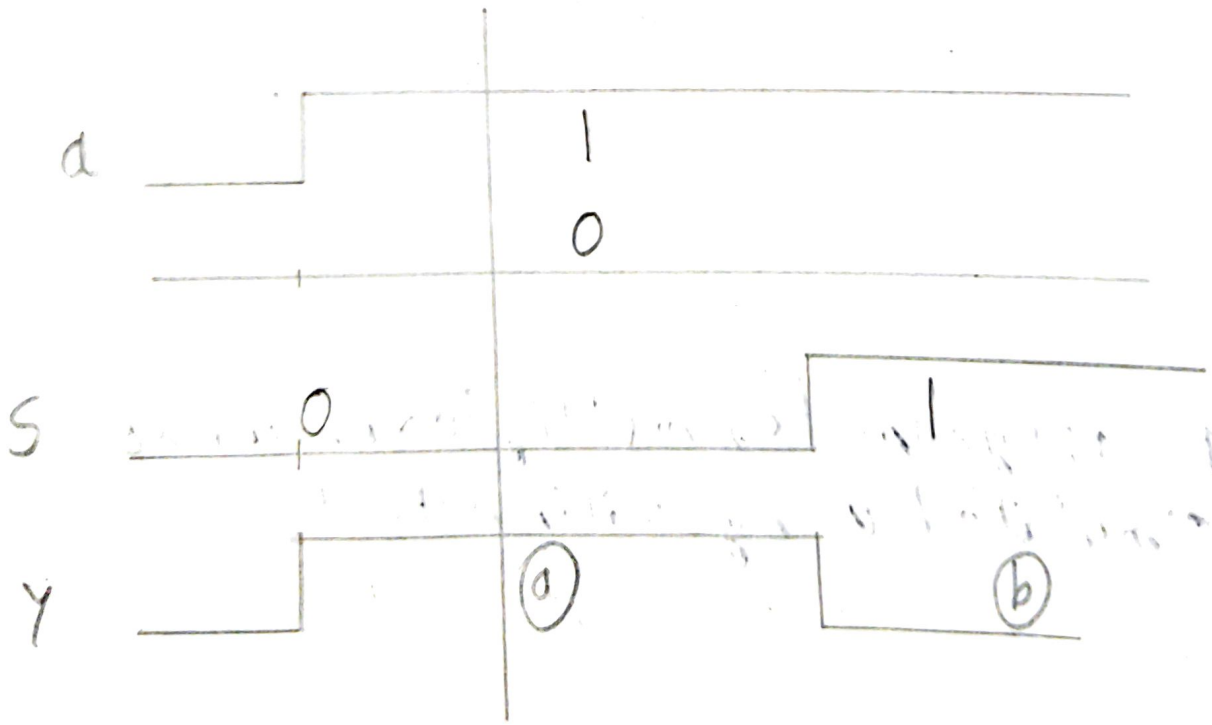
Theory:-

Multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it is to a single output line. The set of selection of a particular line is controlled by selection line. Normally there are 2^{nd} input lines and n selection lines whose bit combinations determine which input is selected. The 4:1 Mux has four inputs I_0, I, I_2, I_3 and select lines S_0 and S_1 the select line S_0 & S_1 are decoded to select a particular AND gate, the output of the AND gates are applied to a single OR gate that provides the oneline output Y.

Procedure:-

1. Click on the Xilinx ISE Design Suite or Xilinx Project navigator icon on the desktop.
2. Write Verilog code by chosing HDL as top level source module.
3. Check Syntax, view RTL schematic ~~in the process window~~ and note device utilization summary by double clicking on the Synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed using model simulation.


a
b
2 : 1
mux
Y
S


a
1
0
S
0
1
Y
a
b

Program:-

- Verilog code for Multiplexer:-

```
module MULTIPLEXER(
 input A,
 input B,
 input C,
 output D,
 output E,
 output Y
 );
 assign  D = A & ~C;
 assign  E = B & C;
 assign  Y = D + E;

 end module.
```

Result:- Thus, the verilog program for multiplexer we written, synthesized and simulated using Xilinx tool.

Verilog program for basic logic gates to synthesize & simulate

```
module demo1(
   input a,b,
   output o,p,q,r,s
   );
assign o= a&b;//and gate
assign p= a|b; //or gate
assign q= ~a;//not gate
assign r= ~(a&b);//nand gate
assign s= ~a&b;//not gate
endmodule
```

Verilog program for full adder to synthesize and simulate

```
module fulladder(
   input a,
   input b,
   input cin,
   output d,
   output e,
   output f,
   output sum,
   output cout
   );
assign d = a ^ b;
assign e = d & cin;
assign f = a & b;
assign sum = d^cin;
assign cout = e|f;
endmodule
```

VERILOG CODE for Multiplexer to synthesize

```
module multiplexer(
   input A,
   input B,
   input C,
   output D,
   output E,
   output Y
   );
assign D = A&~C;
assign E = B&C;
assign F = D+E;

endmodule
```