Simulation (Switch level code):-

① CMOS inverter:-

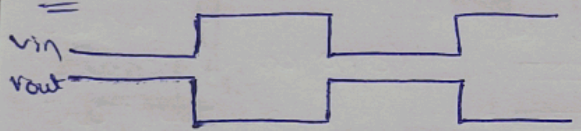
verilog code:-

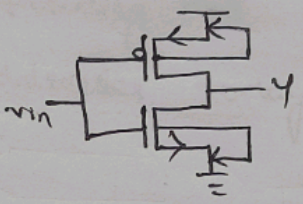
```
module inv(vin,vout);
input vin;
output vout;
supply 0 gnd;
supply 1 pwr;
pmos p1(vout, pwr,vin);
nmos n1( vout, gnd, vin);
endmodule
```

Test Bench:-

```
module inv_txt;
wire vout;
reg vin;
inv i1(vin,vout);
initial
begin
vin=1'b0; #10;
vin=1'b1, #10;
end
endmodule
```

o/p:-





② NAND gate:-

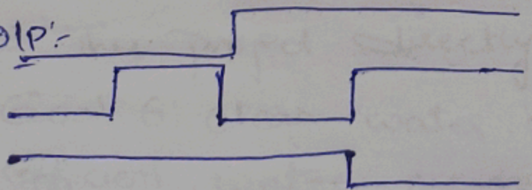
verilog code:-

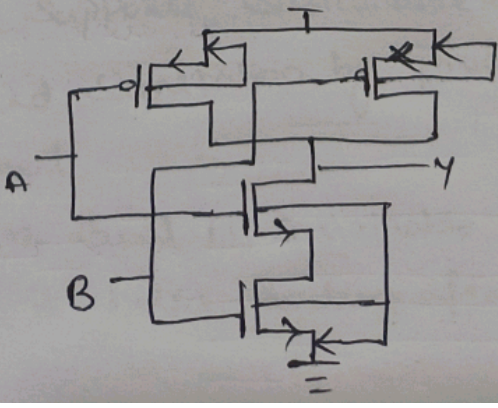
```
'time scale 1ns/1ns
module nand (vin1,vin2, vout);
input vin1, vin2;
output vout;
supply 0 gnd;
supply 1 pwr;
wire contact
pmos p1( vout,pwr,vin1);
pmos p2(vout, pwr, vin2);
nmos n1( vout, contact,vin1);
nmos n2 (vout, gnd ,vin2);
end module
```

TestBench:-

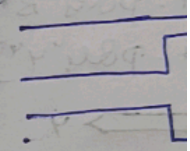
```
'time scale 1ns/1ns
Module nandgate_test;
reg vin1, vin2;
wire vout;
nandgate n1(vout, vin1, vin2);
initial
begin
vin1=1'b0;vin2=1'b0;#10;
vin1=1'b0;vin2=1'b1;#10;
vin1=1'b1;vin2=1'b0;#10;
vin1=1'b1;vin2=1'b1;#10;
end
endmodule
```

o/p:-

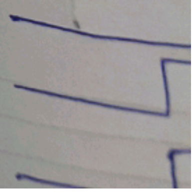




NOR gate:
verilog cod
'time scale
module NO
output vo
input vin
supply 1 p
supply 0
wire con
pmos p1
pmos p2
nmos n1
nmos n2
end modu
o/p:-



XOR gate
verilog co
'time scale
module xo
input vir
output vou
supply 1 p
supply 0
wire con
pmos p1
pmos p2
pmos p3
pmos p4
nmos n1
nmos n2
nmos n3
nmos n4
end mod

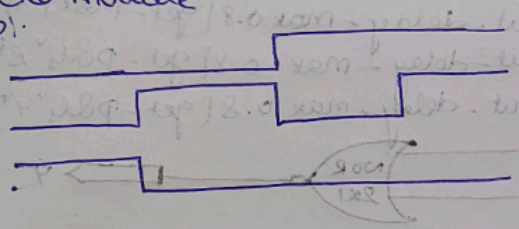


NOR gate:-

code:-

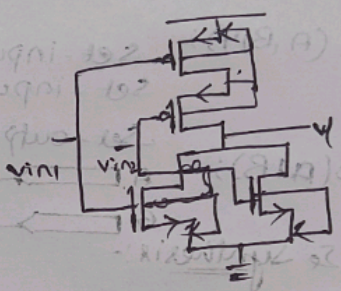
verilog

```
'timescale 1ns/1ns
module NOR (vout, vin1, vin2);
output vout;
input vin1, vin2;
supply1 pwr;
supply0 gnd;
wire Contact;
pmos P1 (Contact, pwr, vin1);
pmos P2 (Contact, vout, vin2);
nmos n1 (vout, gnd, vin1);
nmos n2 (vout, gnd, vin2);
end module
```



Testbench:-

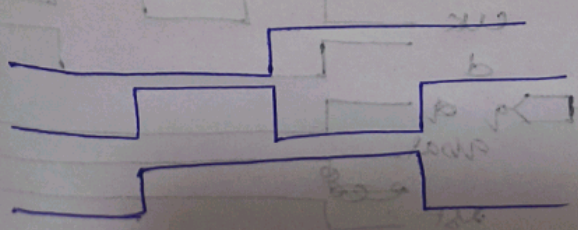
```
'timescale 1ns/1ns
module nor gate(vout, vin1, vin2);
wire vout;
reg vin1, vin2;
norgate n2(vout, vin1, vin2);
initial
begin
vin1 = 1'b0; vin2 = 1'b0; #10;
vin1 = 1'b0; vin2 = 1'b1; #10;
vin1 = 1'b1; vin2 = 1'b0; #10;
vin1 = 1'b1; vin2 = 1'b1; #10;
end
endmodule
```



XOR gate:-

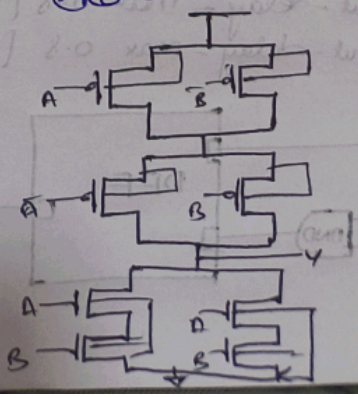
verilog code:-

```
'timescale 1ns/1ns
module XOR(vout, vin1, vin2);
input vin1, vin2;
output vout;
supply1 pwr;
supply0 gnd;
wire Contact1, Contact2, Contact3;
Pmos P1 (Contact1, pwr, vin1);
Pmos P2 (Contact1, pwr, vin2);
Pmos P3 (vout, Contact1, vin1);
Pmos P4 (vout, Contact1, nvin2);
nmos n1 (Contact2, vin1);
nmos n2 (Contact2, gnd, vin2);
nmos n3 (Contact3, vout, vin1);
nmos n4 (Contact3, gnd, nvin2);
end module
```



test bench:-

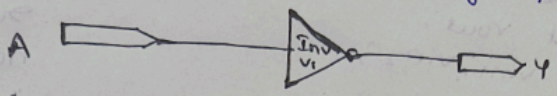
```
'timescale 1ns/1ns;
module XORgate_test;
wire vout;
reg vin1, vin2;
XORgate x1(vout, vin1, vin2);
initial
begin
vin1 = 1'b0; vin2 = 1'b0; #10;
vin1 = 1'b0; vin2 = 1'b1; #10;
vin1 = 1'b1; vin2 = 1'b0; #10;
vin1 = 1'b1; vin2 = 1'b1; #10;
end
endmodule
```



① Inverter:-

```
module inv(A,Y);
input A;
output Y;
assign Y = ~A;
end module
```

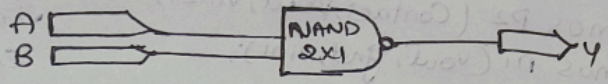
Set_input_delay_max 0.8 [get_ports "A"]
Set_output_delay_max 0.8 [get_ports "Y"]



② NAND:-

```
module NAND(A,B,Y);
input A,B;
output Y;
assign Y = ~(A&B);
end module
```

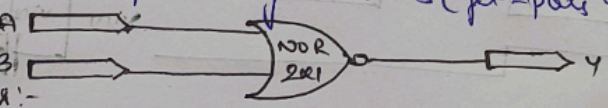
Set_input_delay_max 0.8 [get_ports "A"]
Set_input_delay_max 0.8 [get_ports "B"]
Set_output_delay_max 0.8 [get_ports "Y"]



③ NOR:-

```
module NOR(A,B,Y);
input A,B;
output Y;
assign Y = ~(A|B);
end module
```

Set_input_delay_max 0.8 [get_ports "A"]
Set_input_delay_max 0.8 [get_ports "B"]
Set_output_delay_max 0.8 [get_ports "Y"]



Simulation & Synthesis:-

④ D flip flop:-

Verilog code:-

```
module dff (d,clk,q);
input wire d,clk;
output reg q;
always @(posedge clk)
begin
q <= d;
end
endmodule
```

test bench:-

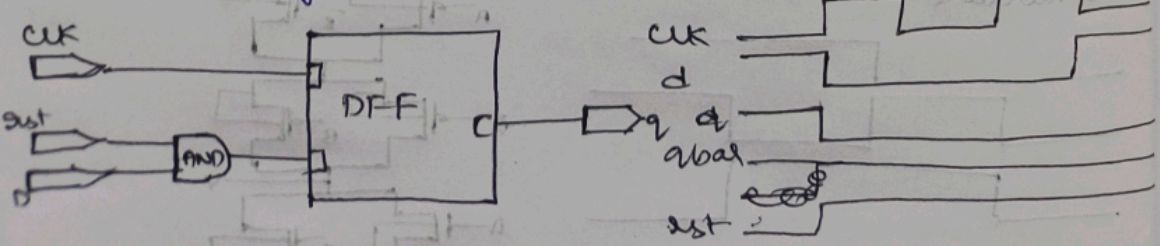
```
module dff_tst;
reg clk,d;
wire q;
dff d1(q,d,clk);
always #10 clk = ~clk;
initial begin
clk = 1; #10;
d = 1'b0; #10;
d = 1'b1; #10;
$finish
end
endmodule
```

TT:-

Rst	clk	D	Q	Qb
0	↑	X	0	1
1	↑	0	0	1
1	↑	1	1	0

Constraints:-

set_input_delay_max 0.8 [get_ports "d"]
set_input_delay_max 0.8 [get_ports "clk"]
set_output_delay_max 0.8 [get_ports "q"]



4-bit pa

Verilog cod

```
module Parall
input [3:0] x,
input c_in;
output [3:0]
output c_out
wire c1,c2,
fulladd stage
fulladd stage
fulladd stage
fulladd stage
end module
module full
input a,b,ci
Output S, co
assign S =
assign Cout =
end module
```

Constraint:

```
set_input
set_input
set_input
set_outp
set_outp
```

4- bit parallel adder :-

Verilog Code :-

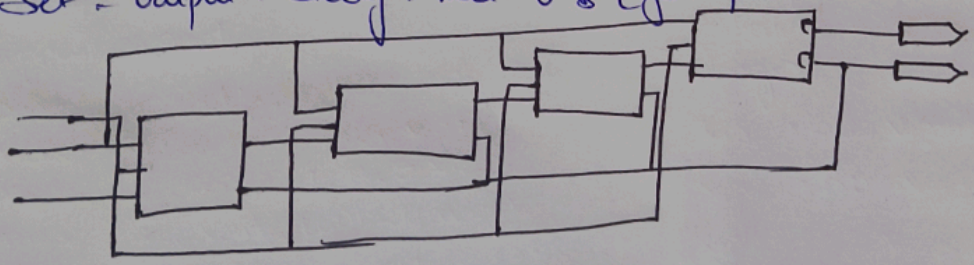
```
module parallel_adder (x,y,c_in,sum,c_out);
input [3:0] x,y;
input c_in;
output [3:0] sum;
output c_out;
wire c1,c2,c3;
fulladd stage 0(x[0],y[0],cin,sum[0],c1);
fulladd stage 1(x[1],y[1],c1,sum[1],c2);
fulladd stage 2(x[2],y[2],c2,sum[2],c3);
fulladd stage 3(x[3],y[3],c3,sum[3],cout);
end module
module fulladd (a,b,cin,S,cout);
input a,b,cin;
Output S,cout;
assign S = a^b^cin;
assign cout = (a&b)|(b&cin)|(cin&cin);
end module
```

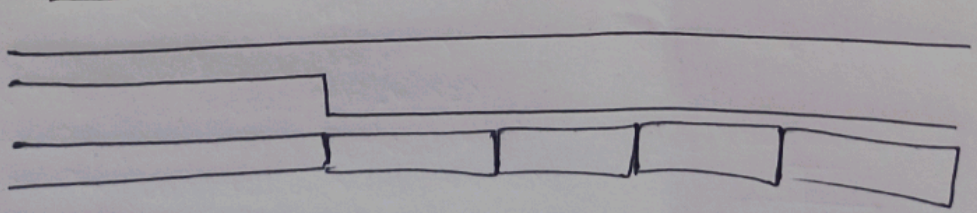
Testbench :-

```
module padder_tb();
reg [3:0] x,y;
reg cin;
wire [3:0] sum;
wire c_out;
parall_add DUT (x,y,cin,sum,
                cout);
initial begin
x=4'b0000; y=4'b0101; cin=1b0
                      #10;
x=4'b1010; y=4'b1101; #40;
end
endmodule
```

Constraint :-

set_input_delay -max 0.8 [get_ports "x"]
set_input_delay -max 0.8 [get_ports "y"]
set_input_delay -max 0.8 [get_ports "cin"]
set_output_delay -max 0.8 [get_ports "sum"]
set_output_delay -max 0.8 [get_ports "c_out"]





14

T-flip-flop:-

verilog code:-

```
module tff_m (t,clk,q);
input wire t,clk;
output q;
always @(posedge clk)
begin
q = ~t;
end
endmodule
```

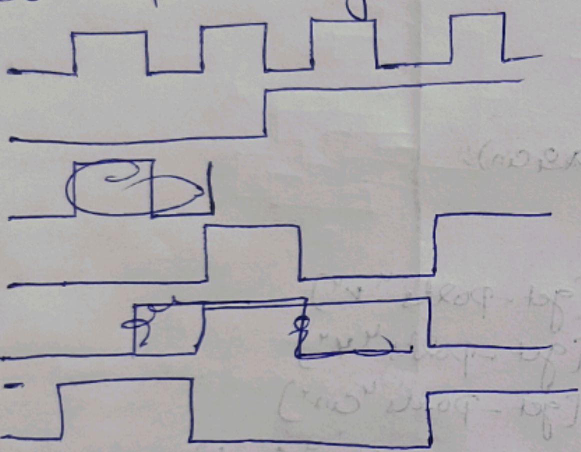
Test bench:-

```
module tff_test;
reg clk,t;
wire q;
tff t1 (clk,t,q);
always #10 clk = ~clk;
initial begin
t=1'b0;#10;
t=1'b1;#10;
$finish
end
end module
```

Constraints:-

```
set_input_delay -max 0.8 [get_ports "t"]
set_input_delay -max 0.8 [get_ports "clk"]
set_input_delay -max 0.8 [get_ports "q"]
set_output_delay -max 0.8 [get_ports "q"]
```

Clk	Qn	T	Qn+1	Qn
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1



4-bit Synchronous Counter:-

verilog code:-

```
module Counter(clk,reset,count);
input wire clk,reset;
output reg[3:0] count;
always @(posedge clk)
if (reset
   count <= 4'b0000;
else
   count <= count + 4'b0001;
end module
```

Testbench:-

```
module counter_test;
reg clk,reset;
wire [3:0] count;
initial
clk = 1'b0;
always #5 clk = ~clk;
s_count term (clk,reset,count);
initial begin
reset = 1'b0;
#15 reset = 1'b1;
#15 reset = 1'b0;
$finish
end
end module
```

set_clk_ lo
set_clk_ tra
set_clk_ unc
set_input_
set_output

Simulation:-
(instead of vis
→ nclaunch
multiple s
Create cds.
→ Don't i
→ OK →
→ A NC
nc la
=) un title
→ nc lau
maximize
→ inv.N
→ inv.th
→ It s
→ Click
→ Now
Sen
→ ge

set_clk_transition -rise 0.1 [get_ports "clk"]
set_clk_transition -fall 0.1 [get_ports "clk"]
set_clk_uncertainity 1.0 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "reset"] clock [get_clock "clk"]
set_output_delay -max 1.0 [get_ports "count"] clock [get_clock "clk"]

Simulation:-

(instead of virtuoso)

⇒ nclaunch -new.

multiple step

Create cds.lib file → file name cds.lib (default) → Save

⇒ Don't include any libraries (verilog design).

⇒ OK → OK.

⇒ A NC launch dialogue box appears now give nc launch > gedit

⇒ Untitled doc → type code → save it

⇒ nclaunch > gedit → type testbench → save it

Maximize nclaunch.

⇒ inv.v → Tools (2) option check ellab block

⇒ inv.tb → Tools (2) compile check ellab, if ellab go down gedit inver

⇒ It shows nclaunch >.

⇒ Click inv_test & elaborate (3) option again invelab (3) elaborate.

⇒ Now click Snapshots of inv_test → launch Simulation (4) option

Send to waveform window

⇒ graph - new

gedit

Save 2 files

↳ inv.v

↳ inv.sdc

csh

source /home/install/

genus

genus -f run.tcl.

DATE : ..... PAGE NO. ..... EXP. NO. .....

Experiment - 10.

Draw the layout and perform physical verification using DRC, ERC and LVS. Extract RC and back-annotate the same and verify the design

Aim: To draw the layout of CMOS Inverter and perform physical verification

Procedure:

* Create New folder
  - ↳ open in terminal - type csh
* Enter & type - cp home / install /cshrc
* Cadence Environment - virtuoso &
* Draw schematic
* Launch
  - ↳ Layout XL
    - ↳ create new - Automatic
* New file, Type } No change
* Two windows will open
  - ↳ select connectivity
    - ↳ Generate → All form source
* Instance ✓
  Ip pin ✓
  PR boundary ✓ } no change. → ok
* Shape
  - ↳ Rectangle - Ok
* Shift F
  right click on pmos & nmos one by one

DATE : PAGE NO. EXP. NO.

Press q

* Select parameter
* Body type - se Integrated - ok
* Select nwell
  Press R → Rectangle (cover pmos)
* Select place
  ↳ pin placement

Pin placement window will open → select V_{dd}

* Create → Select schematic,
  ↳ Select HRail (power Rail)

Vdd Rail will be create.

* Repeat for nmos → select V_{ss}

Press p

* In metal & metal case Connection.
  Right click → Via down to → polysilicon
* Select Assura
  ↳ Technology file

filter → press ..

home / install / foundary / analog / 45nm / assura - tech.lib.

press ok

Select assura
  ↳ Run DRC (Design rule check)

* Give run ~~on~~ name - any name.
* Select Technology - gpdk045 - a_v - apply ok

Select Assura
  ↳ Run LVS.

Give run name - (any name)

Apply ok

Violation report will be displayed