SEMBODAI RUKMANI VARATHARAJAN ENGINEERING COLLEGE
SEMBODAI – 614809

(Approved By AICTE, New Delhi – Affiliated To ANNA UNIVERSITY::Chennai)

EC6612 -VLSI DESIGN LABORATORY MANUAL
(REGULATION-2013)

LAB MANUAL

DEPARTMENT OF ECE

NAME: _____

REGISTER NUMBER:_____

YEAR/SEM.: _____

ACADEMIC YEAR: 2015 - 2016

EC6612 -VLSI DESIGN LABORATORY MANUAL (REGULATION-2013)

Prepared By,

VENKATASUBRAMANIAN.R

AP/ECE/SRVEC

Approved By,

Mr.SUNDAR.G

HOD/ECE/SRVEC

SEMBODAI RUKMANI VARATAHARAJAN ENGINEERING COLLEGE SEMBODAI - 614809

EC6612 -VLSI DESIGN LABORATORY MANUAL (REGULATION-2013)

AS PER ANNA UNIVERSITY SYLLABUS

LIST OF EXPERIMENTS

LIST OF EXPERIMENTS FPGA BASED EXPERIMENTS

1. HDL based design entry and simulation of simple counters, state machines, adders (min 8 bit) and multipliers (4 bit min).

2. Synthesis, P&R and post P&R simulation of the components simulated in (I)above. Critical paths and static timing analysis results to be identified. Identify and verify possible conditions under which the blocks will fail to work correctly.

3. Hardware fusing and testing of each of the blocks simulated in (I). Use of either chip scope feature (Xilinx) or the signal tap feature (Altera) is a must. Invoke the PLL and demonstrate the use of the PLL module for clock generation in FPGAs.

IC DESIGN EXPERIMENTS: (BASED ON CADENCE / MENTOR GRAPHICS / EQUIVALENT)

4. Design and simulation of a simple 5 transistor differential amplifier. Measure gain, ICMR, and CMRR

5. Layout generation, parasitic extraction and re simulation of the circuit designed (I)

6. Synthesis and Standard cell based design of an circuits simulated in 1(I) above. Identification of critical paths, power consumption.

7. For expt (c) above, P&R, power and clock routing, and post P&R simulation.

8. Analysis of results of static timing analysis.

TABLE OF CONTENT

Design entry and simulation of combinational logic circuits

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To study about the simulation tools available in Xilinx project navigator using Verilog tools.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name.
3	Select Verilog module.
4	Type your Verilog coding.
5	Check for syntax.
6	Select the new source of test bench waveform
7	Choose behavioral simulation and simulate it by Xilinx ISE simulator.
8	Verify the output.

SEMBODAI RUKMANI VARATHARAJAN ENGINEERING COLLEGE
SEMBODAI - 614820
KNOWLEDGE IS POWER

Logic gates

AND GATE

```
module gl(a,b,c);
input a;
input b;
output c;
and(c,a,b);
end module
```

OR GATE

```
module gl(a,b,c);
input a;
input b;
output c;
or(c,a,b);
end module
```

XOR GATE

```
module gl(a,b,c);
input a;
input b;
output c;
xor (c,a,b);
end module
```

NAND GATE

```
module gl(a,b,c);
input a;
input b;
output c;
nand(c,a,b);
end module
```

NOR GATE

```
module gl(a,b,c);
input a;
input b;
output c
nor(c,a,b);
end module
```

HALF ADDER

```
module half adder(a,b,c,s);
input a;
input b;
output c;
output s;
xor(s,a,b);
and(c,~a,b);
end module
```

HALF SUBTRACTOR

```
module half sub(a,b,c,s);
input a;
input b;
output c;
output s;
xor(s,a,b);
and(c,~a,b);
end module
```

ENCODER

```
module Encd2to4(i0, i1, i2, i3, out0,out1);
input i0,i1, i2, i3;
output out0, out1;
reg out0,out1;
always@(i0,i1,i2,i3)
case({i0,i1,i2,i3})
4'b1000:{out0,out1}=2'b00;
4'b0100:{out0,out1}=2'b01;
4'b0010:{out0,out1}=2'b10;
4'b0001:{out0,out1}=2'11;
default:
$display("Invalid");
endcase
endmodule
```

DECODER

```
// Module Name: Decd2to4
module Decd2to4(i0, i1, out0, out1, out2,
out3); input i0, i1;
output out0, out1, out2,
out3; reg
out0,out1,out2,out3;
always@(i0,i1)
case({i0,i1})
2'b00:
{out0,out1,out2,out3}=4'b1000;
2'b01:
{out0,out1,out2,out3}=4'b0100;
2'b10:
{out0,out1,out2,out3}=4'b0010;
2'b11:
{out0,out1,out2,out3}=4'b0001;
default:
$display("Invalid");
endcase
endmodule
```

MULTIPLEXER

// Module Name: Mux4to1

```
module Mux4to1(i0, i1, i2, i3, s0, s1,
out); input i0, i1, i2, i3, s0, s1;
output
out; wire
s1n,s0n;
wire
y0,y1,y2,y3;
not (s1n,s1);
not (s0n,s0);
and (y0,i0,s1n,s0n);
and (y1,i1,s1n,s0);
and
(y2,i2,s1,s0n);
and
(y3,i3,s1,s0);
or
(out,y0,y1,y2,y3);
endmodule
```

DEMULTIPLEXER

```
// Module Name: Dux1to4
module Dux1to4(in, s0, s1, out0, out1, out2, out3);
input in, s0, s1;
output out0, out1,
out2,out3; wire s0n,s1n;
not(s0n,s0);
not(s1n,s1);
and (out0,in,s1n,s0n);
and (out1,in,s1n,s0);
and (out2,in,s1,s0n);
and (out3,in,s1,s0);
endmodule
```

8 BIT ADDER

```
module adder(a,b, s,c);
 input [7:0] a,b;
output [7:0] s,c;
 assign {c,s} = a + b;
endmodule
```

MULTIPLIER

```
module multi(a,b,c);
input [3:0] a,b;
output [7:0] c;
assign c = a * b;
endmodule
```

RESULT

Thus the program for study of simulation using tools and the output also verified successfully.

Design Entry and simulation of sequential logic circuits

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To study about the simulation tools available in Xilinx project navigator using Verilog tools.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name
3	Select Verilog module.
4	Type your Verilog coding
5	Check for syntax
6	Select (view RTL schematic) from the synthesis-XST menu
7	Verify the logic circuit and equivalent parameters.

PRBS GENERATORS

```
module prbs(a,clk,clr);
output [3:0] a;
input clk,clr; reg [3:0]
tmp;
always @(posedge clk or posedge clr)
begin
if(clr)
begin
tmp = 4'b1111;
 end
else
begin
tmp = { tmp[0]^tmp[1],tmp[3],tmp[2],tmp[1]};
end
end
assign a=tmp;
endmodule
```

ACCUMULATOR

```
module acc(indata, clk,clr, outdata);
 input [3:0] indata;
input clk,clr;
output [3:0] outdata;
reg [3:0] outdata;
always@(posedge clk or posedge clr)
begin
if(clr)
outdata <= 4'd0;
else
outdata <= indata;
end
endmodule
```

2- BIT COUNTER

// Module Name: Count2Bit

```
module Count2Bit(Clock, Clear, out);
input Clock, Clear;
output [1:0] out;
 reg [1:0]out;
always@(posedge Clock, negedge Clear)
if((~Clear) || (out>=4))
out=2'b00;
else
out=out+1;
endmodule
```

RESULT

Thus the program for study of simulation using tools and the output also verified successfully.

Study of Synthesis Tools

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To study about synthesis tools available in Xilinx project navigator.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name
3	Select Verilog module.
4	Type your Verilog coding
5	Check for syntax
6	Select (view RTL schematic) from the synthesis-XST menu
7	Verify the logic circuit and equivalent parameters.

THEORY

Now that you have created the source files, verified the design behaviour with simulation, and added constraints, you are ready to synthesize and implement the design.

Implementing the Design

1. Select the **counter** source file in the Sources in Project window.
2. In the Processes for Source window, click the "+" sign next to

Implement Design. The Translate, Map, and Place & Route processes are displayed. Expand those processes as well by clicking on the "+" sign. You can see that there are many sub-processes and options that can be run during design implementation.

3. Double-click the top level **Implement Design** process.ISE determines the current state of your design and runs the processes needed to pull your design through implementation. In this case, ISE runs the Translate, Map and PAR processes. Your design is now pulled through to a placed-and routed state. This feature is called the "pull through model."

4. After the processes have finished running, notice the status markers in the Processes for Source window. You should see green checkmarks next to several of the processes, indicating that they ran successfully. If there are any yellow exclamation points, check the warnings in the Console tab or the Warnings tab within the Transcript window. If a red X appears next to a process, you must locate and fix the error before you can continue.

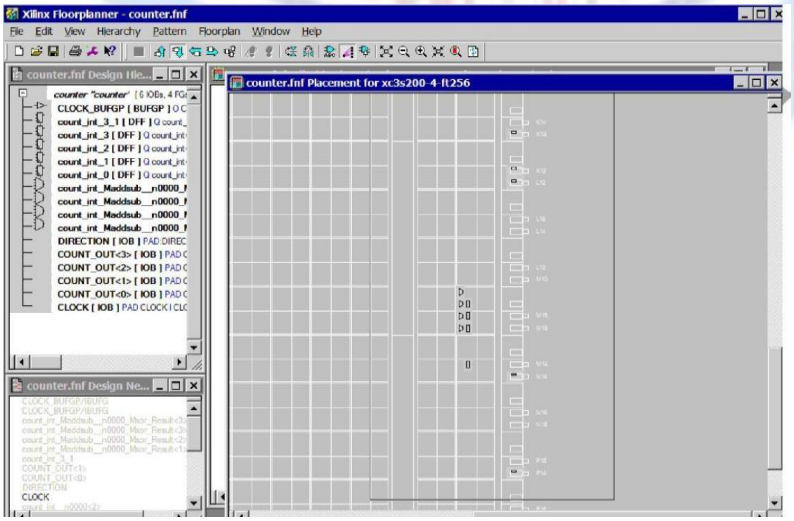


Figure 1*:* Floor planner View - Detailed View

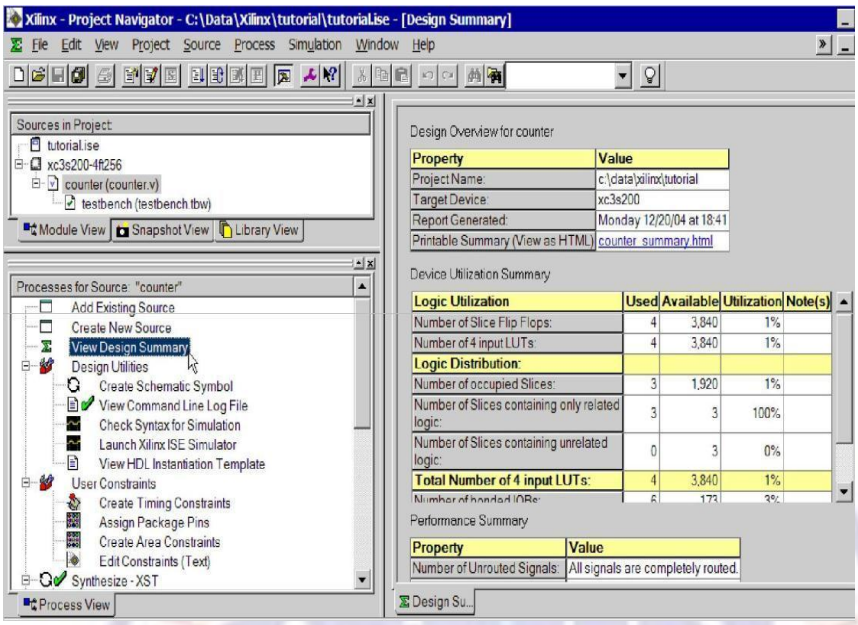


Figure 2: Design Summary View

RESULT

Thus the program to study about synthesis tools available in Xilinx project navigator and the output also verified successfully.

Study of Place and Root-Back Annotation

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To study about place and back annotation in xilinx project navigator using verilog coding.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx ISE – 9.1 navigator.	1

b) Procedure for doing the experiment:

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name.
3	Select Verilog module
4	Type your Verilog coding
5	Check for syntax.
6	Assign package pins and view floor planner diagram of the FPGA.
7	Make necessary changes in the diagram if required.
8	Save the changes, back annotate the changed constraints
9	Verify the UCF that changes are updated
10	Get the length of the sequence as N.

THEORY

After implementation is complete, you can verify your design before downloading it to a device.

Viewing Placement

In this section, you will use the Floor planner to verify your pin outs and placement. Floor planner is also very useful for creating area groups for designs.

1. Click on the **Design Summary** tab at the bottom of the window. If you closed the summary during this tutorial, you can reopen it by double clicking the **View Design Summary** process.

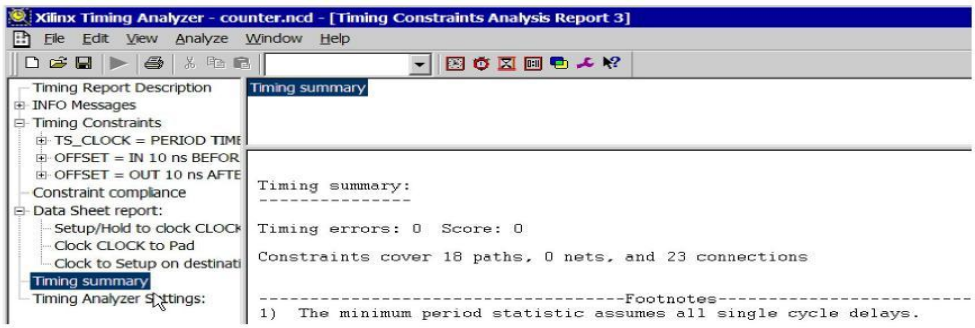


Figure 1: Timing Analyzer - Timing Summary

2.In the Device Utilization Summary section, observe the number of Slice Flip Flops that were used during implementation. You should see 4 flip flops, since you implemented a 4-bit counter.

3. To see other reports, scroll to the bottom of the Design Summary. You can click on a report from here to view it in the ISE Text Editor.

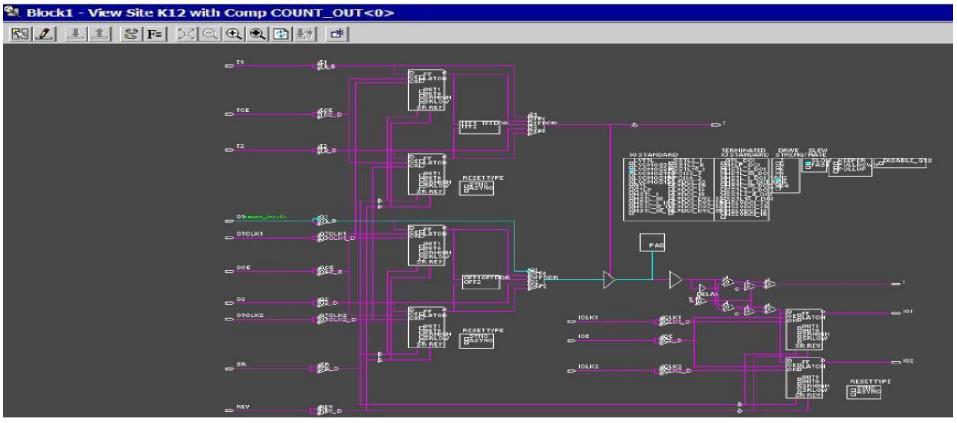


Figure 2: FPGA Editor - Detailed View

Timing Closure

In this section, you will run timing analysis on your design to verify that your timing constraints were met. Timing closure is the process of working on your design to ensure that it meets your necessary timing requirements. ISE provides several tools to assist with timing closure.

1. In the Processes for Source window, under the Place & Route group of processes, expand the **Generate Post-Place & Route Static Timing** group by clicking the "+"sign.

2. Double-click the **Analyse Post-Place & Route Static Timing** process. The Timing Analyser opens.

3. To analyse the design, select **Analyse Against Timing Constraints.** The Analyse with Timing Constraints dialog box opens.

4. Click **OK**. When analysis is complete, the timing report opens.

5. Select **Timing summary** from the Timing Report Description tree in the left window. This displays the summary section of the timing report, where you can see that no timing errors were reported. Close the Timing Analyser without saving.

Viewing the Placed and Routed Design

In this section, you will use the FPGA Editor to view the design. You can view your design on the FPGA device, as well as edit the placement and routing with the FPGA Editor.

1. Double-click the **View/Edit Routed Design (FPGA Editor)** process found in the Place & Route group of processes. Your implemented design opens in the FPGA Editor.

2. Look in the **List** window to examine your design components.

3. Click on the **COUNT_OUT K12** IOB in the List window to select the row. This is one of the outputs in your design.

4. With the COUNT_OUT K12 row selected, select **View _ Zoom Selection**. In the editor window, you can see the COUNT_OUT<0> IOB highlighted in red.

5. Push into (double-click) the red-highlighted **COUNT_OUT K12** IOB. You should see Fig 2.

6. Enlarge the window and zoom in so you can see more detail. This view shows the inside of an FPGA at the lowest viewable level. The blue line

shows the route that is used through the IOB. The red lines show the routes that are available.

7. Verify that the signal goes to the pad as an output.

8. Close the FPGA Editor.

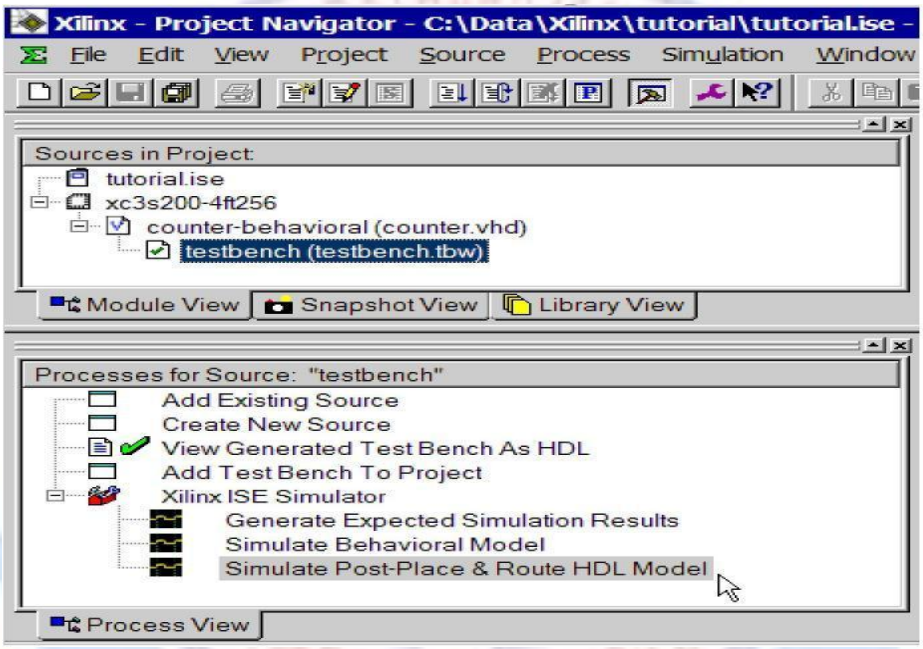


Figure 3: Simulator Processes for Test Bench

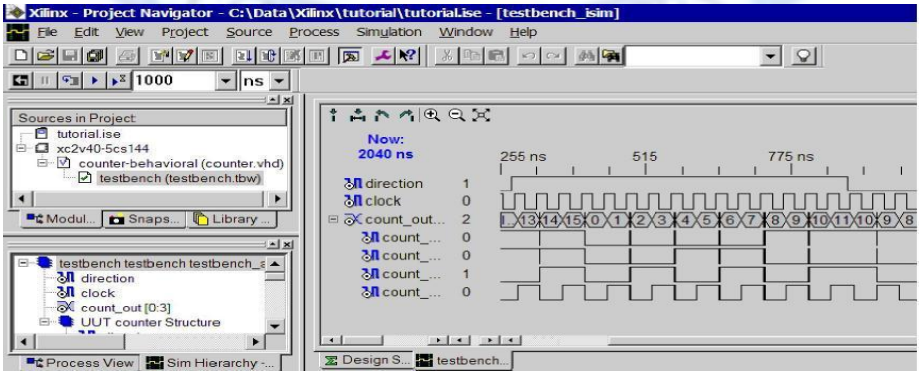


Figure 4: Timing Simulation in ISE Simulator

Timing Simulation (ISE Simulator)

can verify that your design meets your timing requirements by running a timing simulation. You can use the same test bench waveform that was used earlier in the design flow for behavioral simulation. When running timing simulation, the ISE tools create a structural HDL file which includes timing information available after Place and Route is run. The simulator will run on a model that is created based on the design to be downloaded to the FPGA. If you are using ISE Base or Foundation, you can simulate your design with the ISE Simulator. To simulate your design with Modalism, skip to the "Timing Simulation (ModelSim)" section.

To run the integrated simulation processes:

1. Select the test bench waveform in the Sources in Project window. You can see the ISE Simulator processes in the Processes for Source window.

2. Double-click the Simulate Post-Place & Route Model process. This process generates a timing-annotated net list from the implemented and simulates it. The resulting simulation is displayed in the Waveform Viewer. These results look different than those you saw in the behavioural simulation earlier in this tutorial. These results show timing delays.

3. To see your simulation results, zoom in on the transitions and view the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.

4. Zoom in again to see the timing delay between a rising clock edge and output transition.

5. Click the Measure Marker button and then click near the 300 ns mark. Drag the second marker to the point where the output becomes stable to see the time delay between the clock edge and the transition.

6. Close the waveform view window. You have completed timing simulation of your design using the ISE Simulator. Skip past the ModelSim section below, and proceed to the "Creating Configuration Data" section.

RESULT

Thus the program for perform the place and root-back annotation was studied and the output also verified successfully.

CMOS Inverter

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To perform the functional verification of the CMOS Inverter through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	S-Edit using cadance Tool.	1

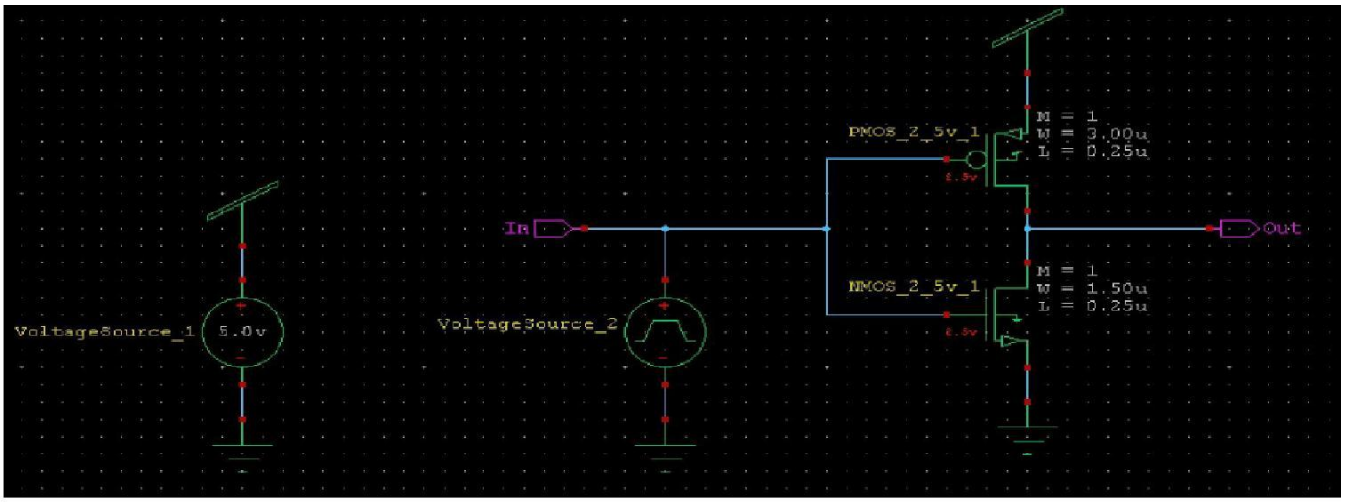
b) Procedure for doing the experiment

S.No	Details of the step
1	Draw the schematic of CMOS Inverter using S-edit.
2	Perform Transient Analysis of the CMOS Inverter.
3	Obtain the output waveform from W-edit
4	Obtain the spice code using T-edit.

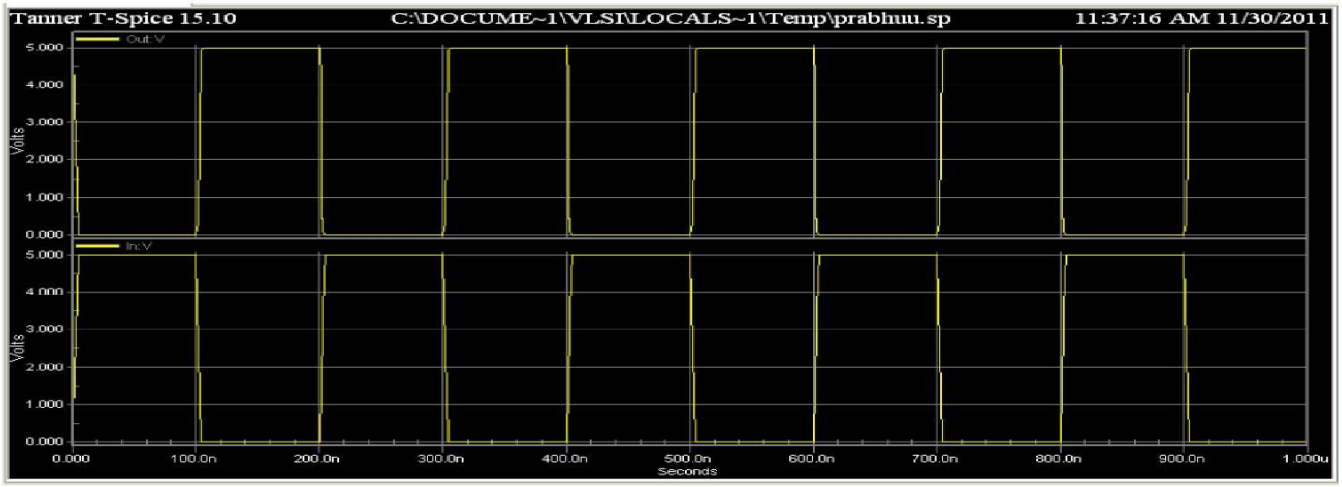
THEORY

Inverter consists of nMOS and pMOS transistor in series connected between VDD and GND. The gate of the two transistors are shorted and connected to the input. When the input to the inverter A = 0, Nmos transistor is OFF and pMOS transistor is ON. The output is pull-up to VDD. When the input A = 1, nMOS transistor is ON and pMOS transistor is OFF. The Output is Pull-down to GND.

SCHEMATIC DIAGRAM



SIMULATED WAVEFORM:



RESULT

Thus the functional verification of the CMOS Inverter through schematic entry, and the output also verified successfully.

Universal Gates

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To perform the functional verification of the universal gate through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

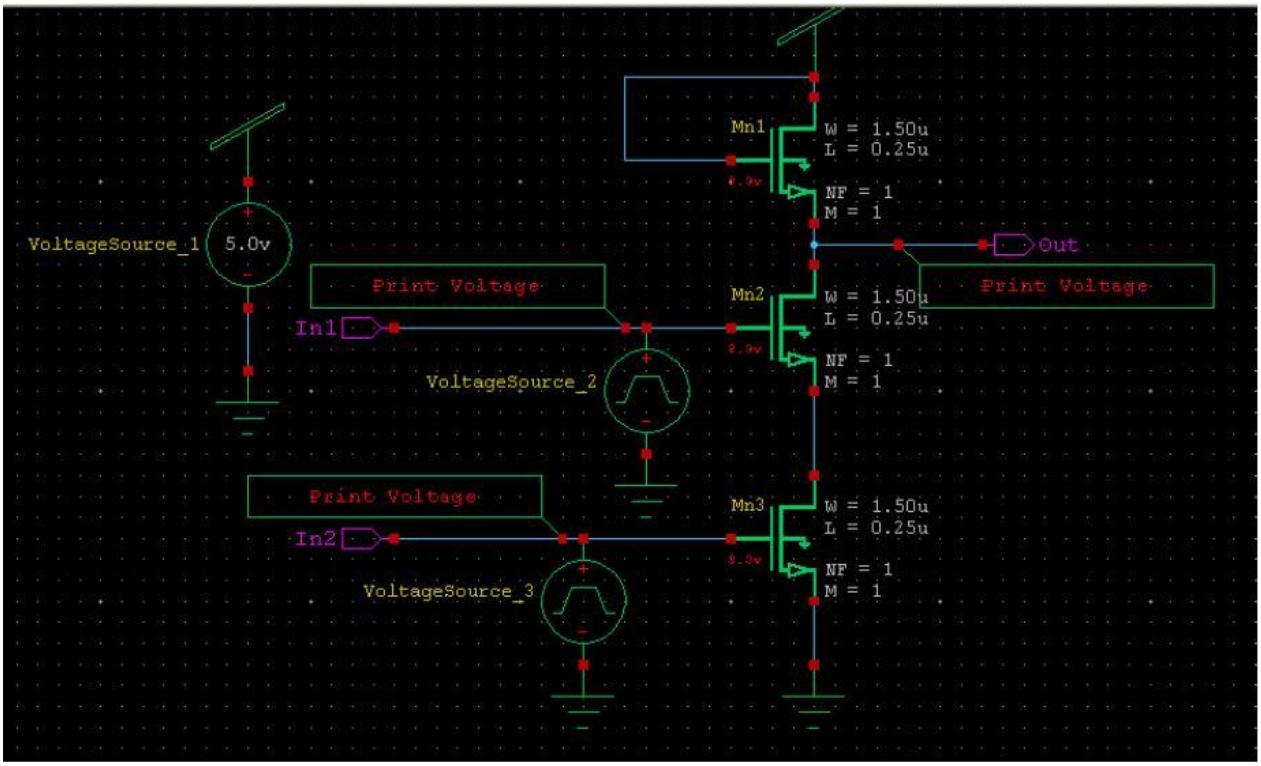
a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	S-Edit using CadanceTool.	1

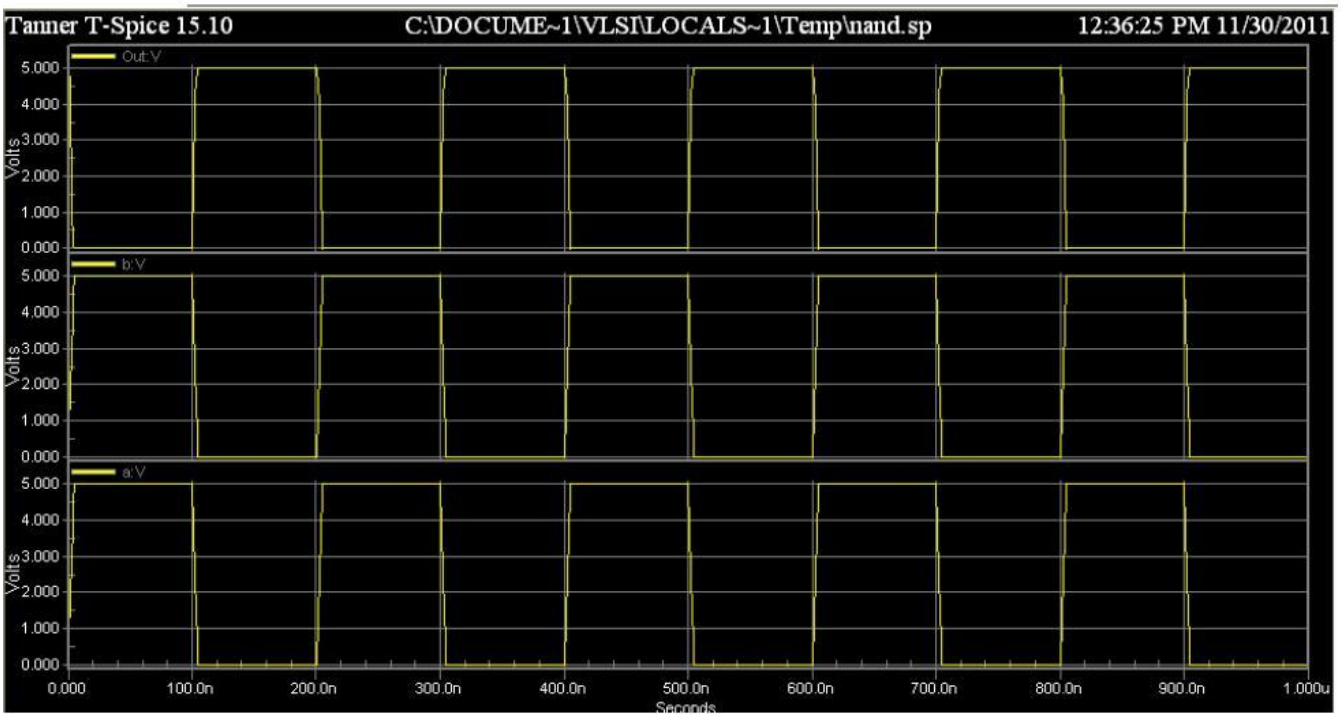
b) Procedure for doing the experiment

S.No	Details of the step
1	Draw the schematic of CMOS Inverter using S-edit.
2	Perform Transient Analysis of the CMOS Inverter.
3	Obtain the output waveform from W-edit.
4	Obtain the spice code using T-edit

NAND GATE

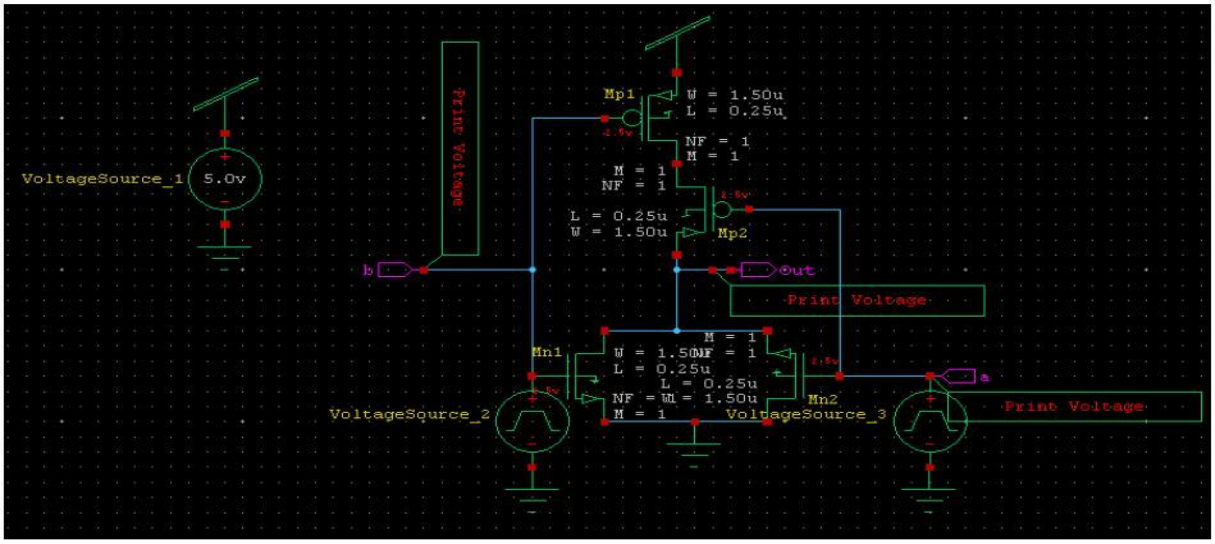


SIMULATED WAVEFORM:

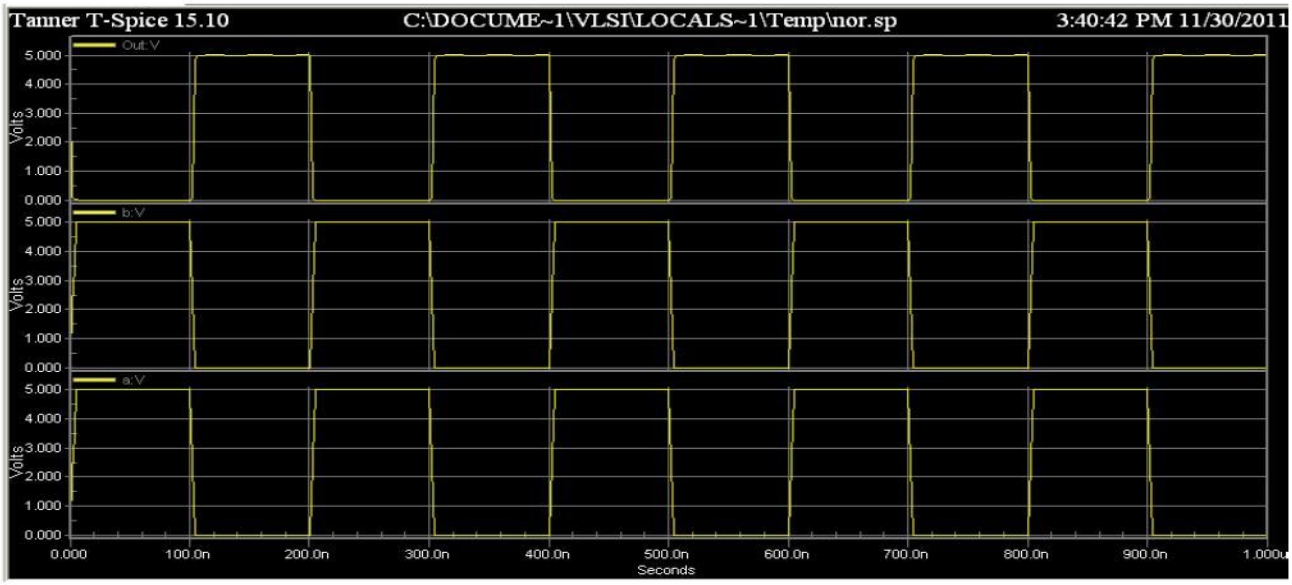


NOR GATE

SCHEMATIC DIAGRAM



SIMULATED WAVEFORM:



RESULT

Thus the functional verification of the NAND& NOR Gate through schematic entry, and the output also verified successfully

Differential Amplifier

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To calculate the gain, bandwidth and CMRR of a differential amplifier through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

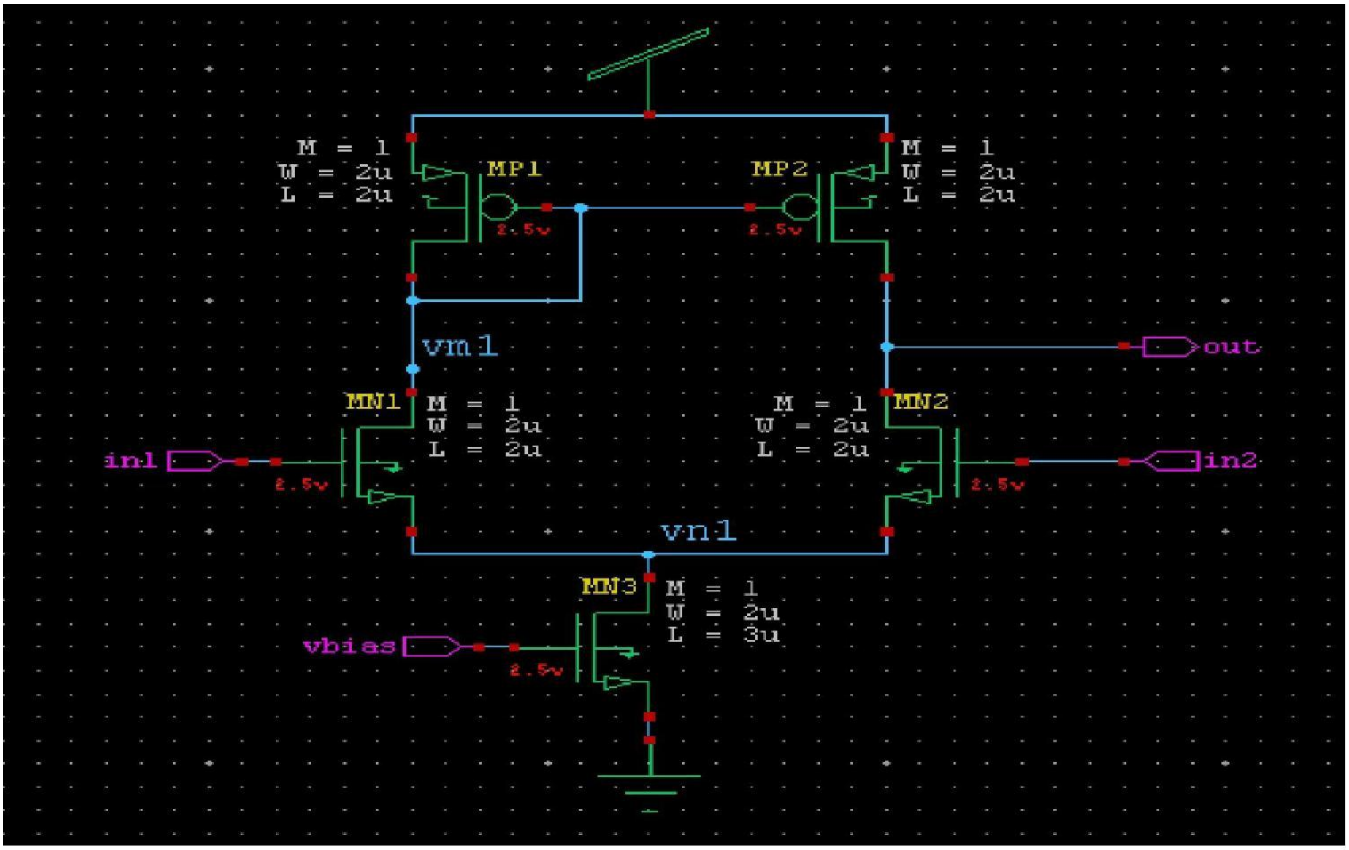
a) Facilities required to do the experiment

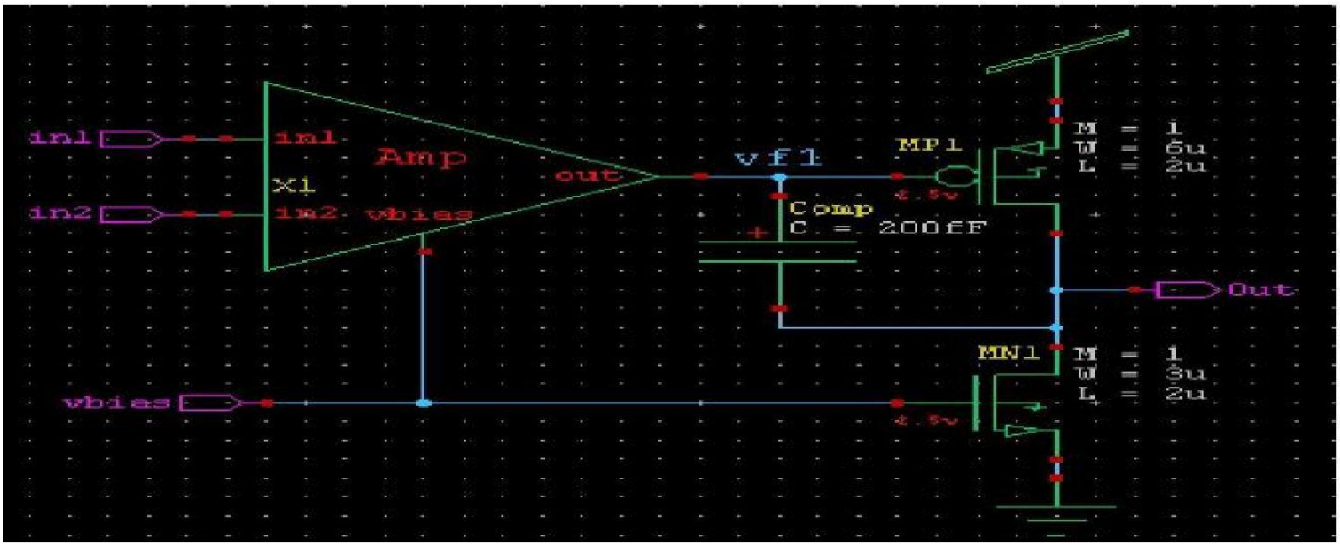
S.No.	SOFTWARE REQUIREMENTS	Quantity
1	S-Edit using CadanceTool.	1

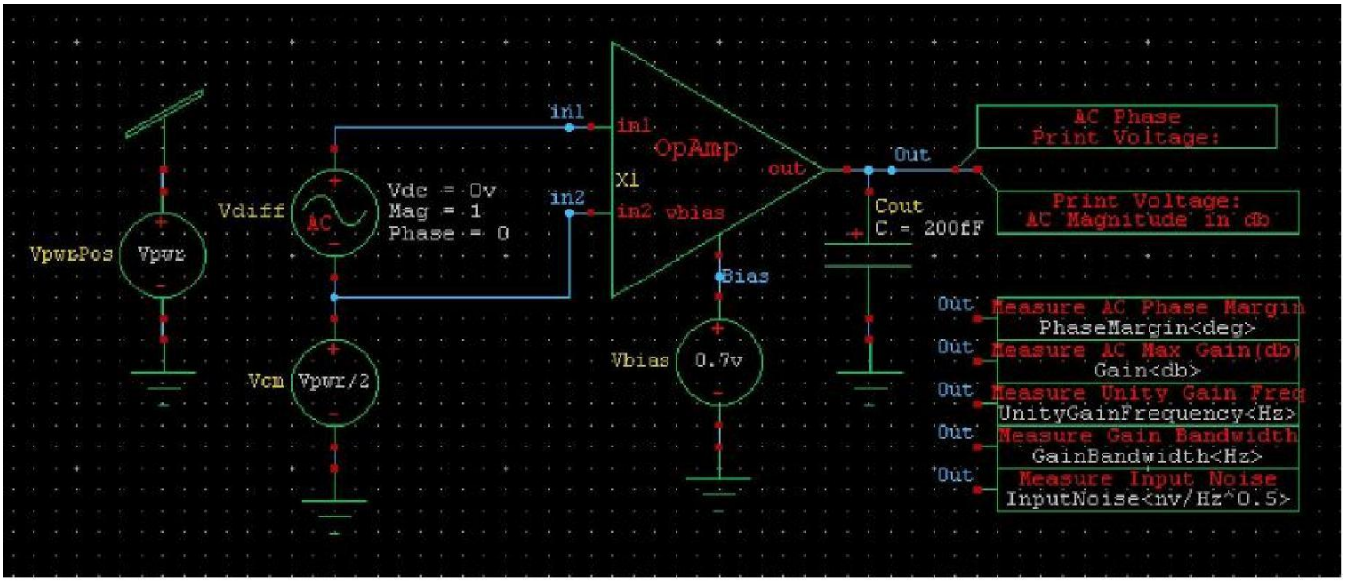
b) Procedure for doing the experiment

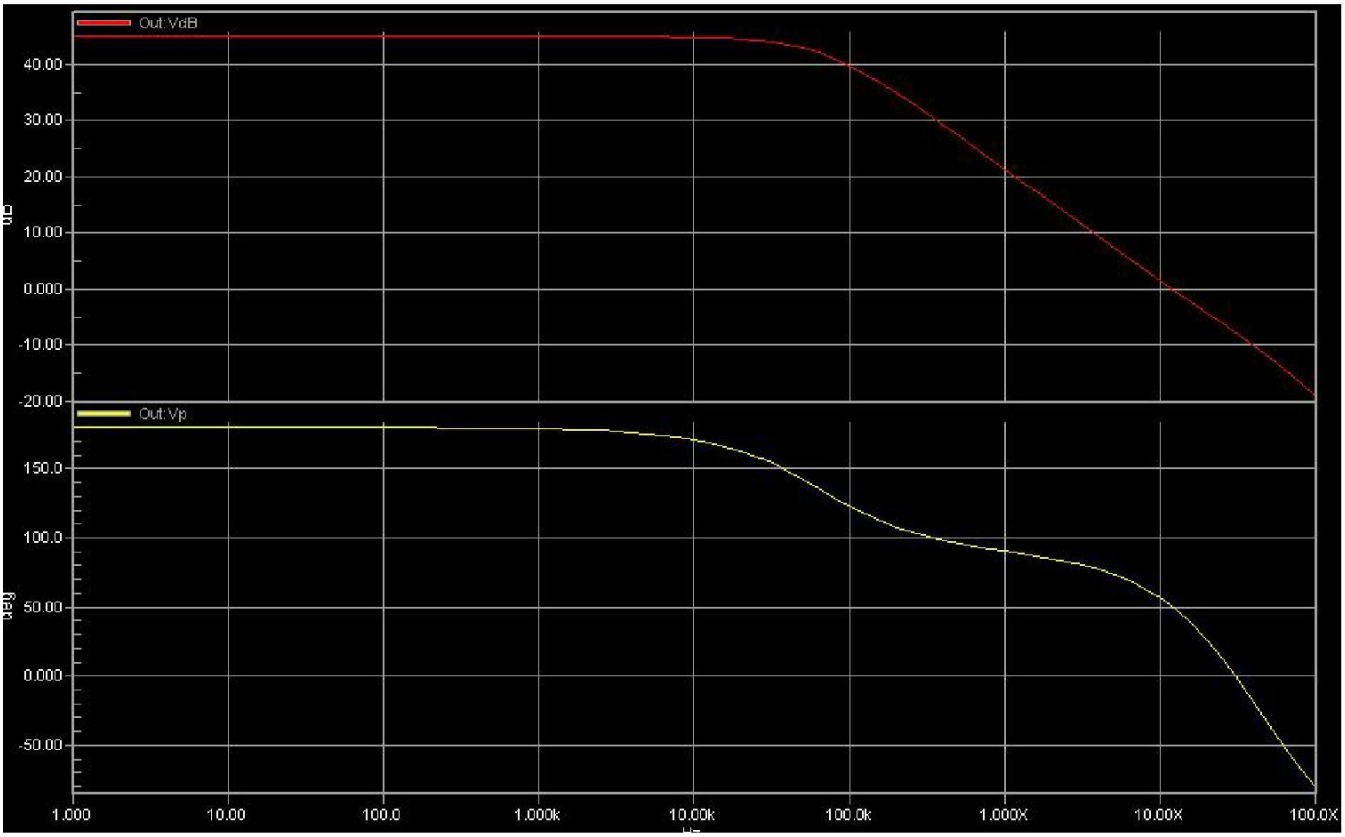
S.No	Details of the step
1	Draw the schematic of differential amplifier using S-edit and generate the symbol.
2	Draw the schematic of differential amplifier circuit using the generated symbol.
3	Perform AC Analysis of the differential amplifier.
4	Obtain the frequency response from W-edit.
5	Obtain the spice code using T-edit.

SCHEMATIC DIAGRAM









RESULT

Thus the functional verification of the Differential Amplifier through schematic entry, and the output also verified successfully

Layout Of CMOS Inverter

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To draw the layout of CMOS Inverter using L-Edit and extract the SPICE code.

FACILITIES REQUIRED AND PROCEDURE

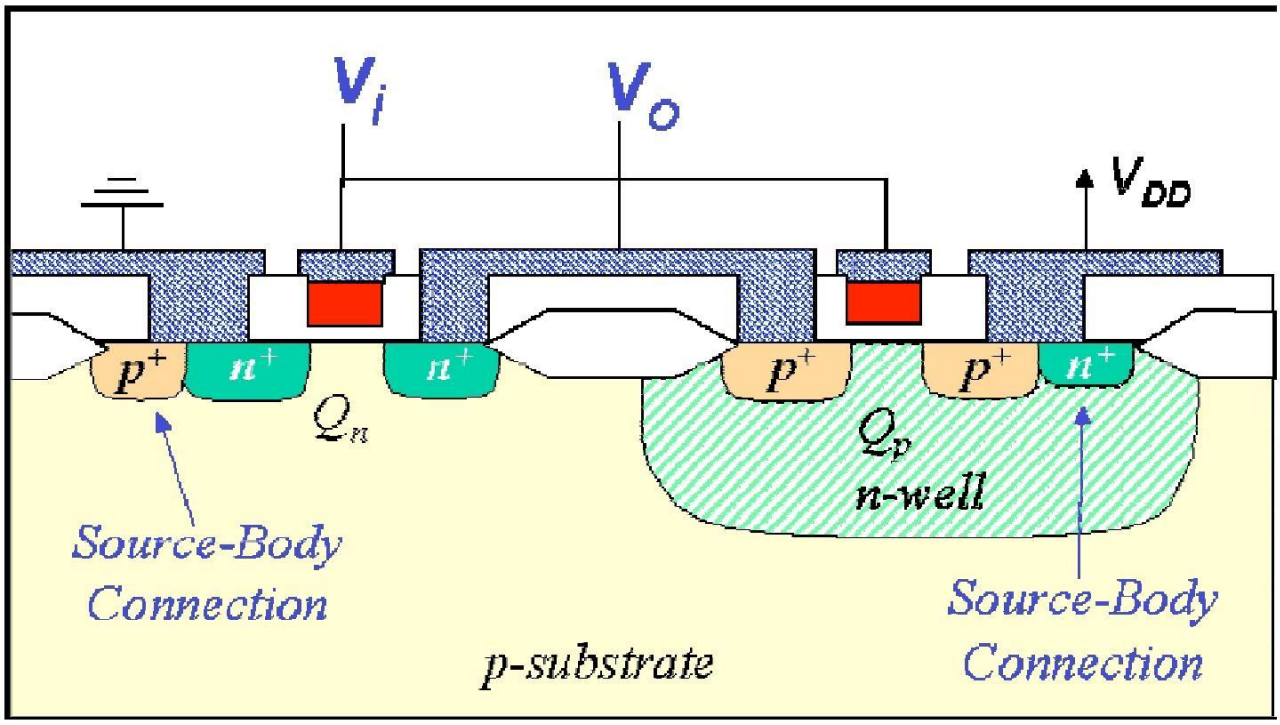
a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	L-Edit using CadanceTool.	1

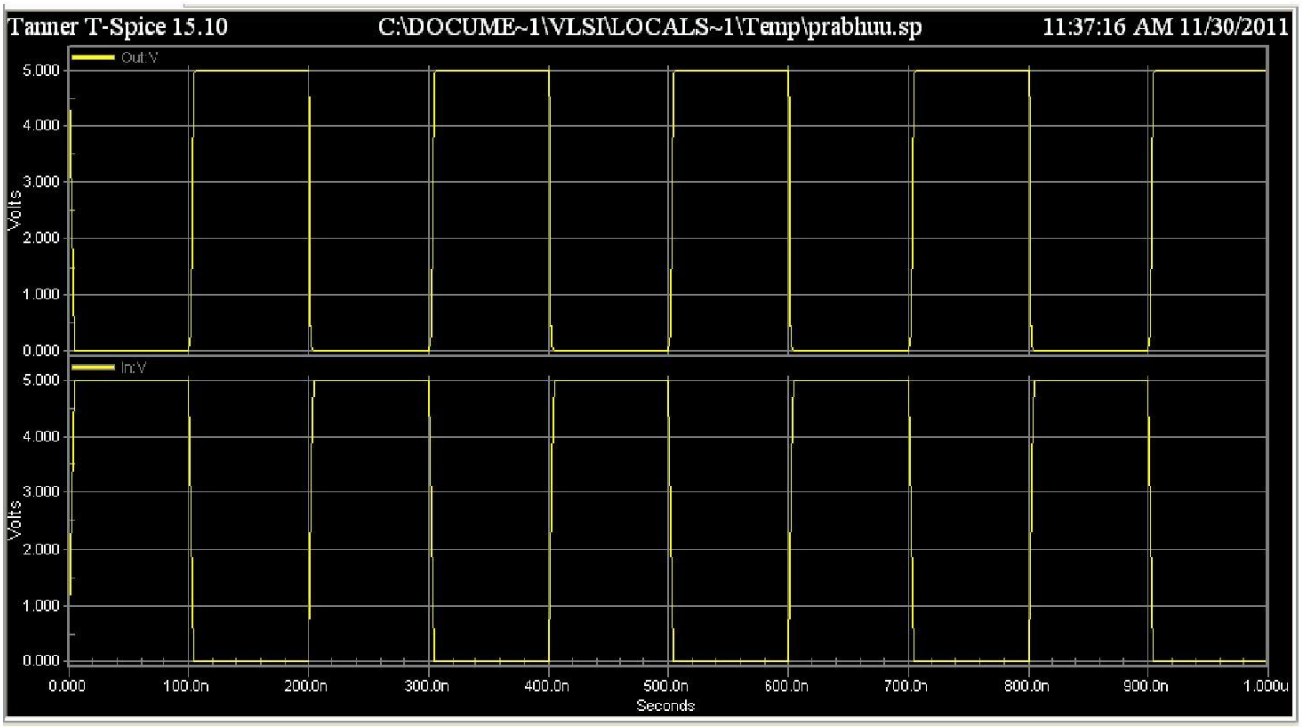
b) Procedure for doing the experiment

S.No	Details of the step
1	Draw the CMOS Inverter layout by obeying the Lamda Rules using Ledit.
2	Poly - 2λ ii. Active contact - 2 λ iii. Active Contact – Metal - 1 λ iv. Active Contact – Active region - 2 λ v. Active Region – Pselect - 3 λ vi. Pselect – nWell - 3λ
3	Check DRC to verify whether any region violate the lamda rule
4	Setup the extraction and extract the spice code using T-spice.

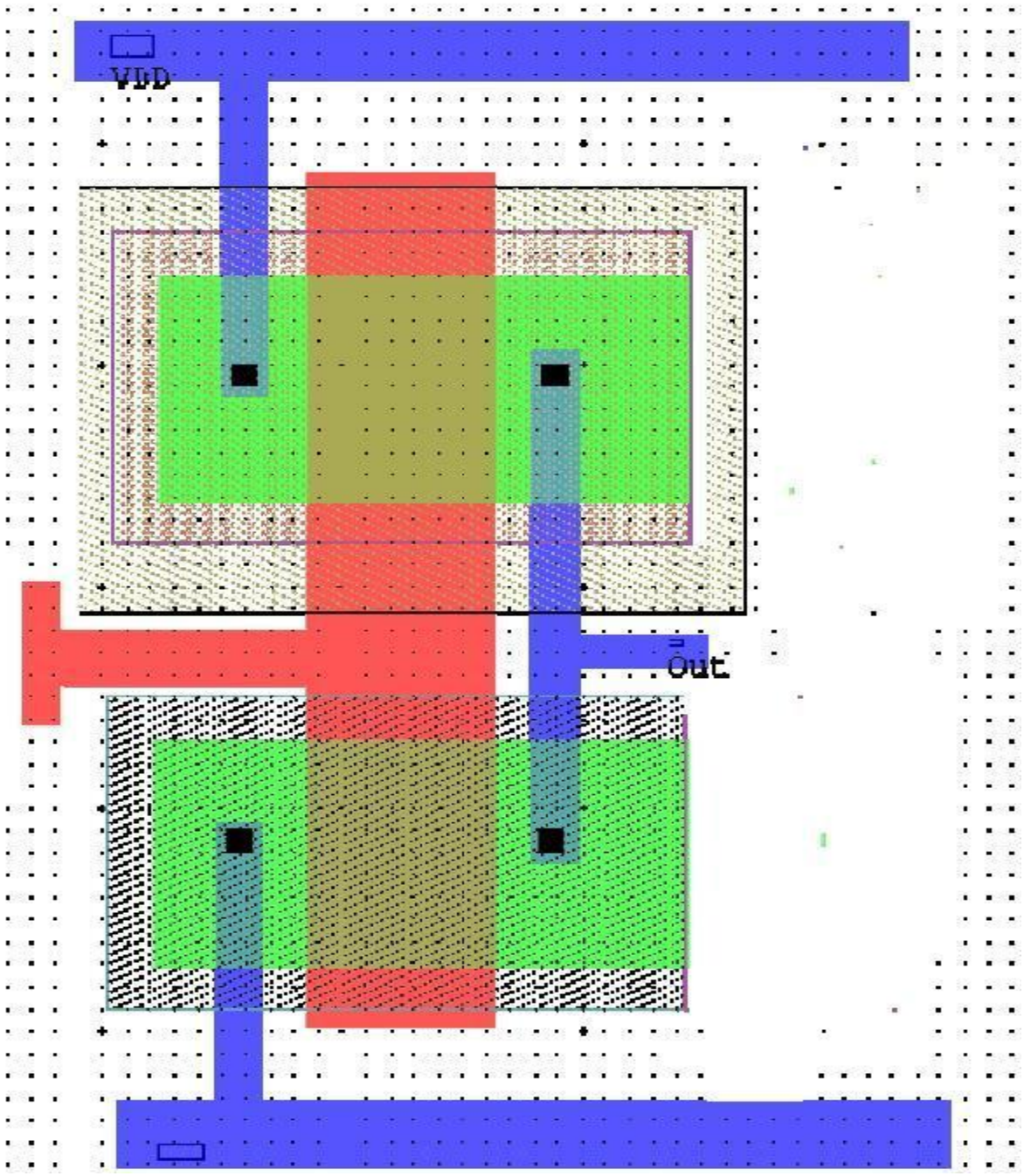
CMOS INVERTER



SIMULATED WAVEFORM:



LAYOUT DIAGRAM:



RESULT

Thus the layout of CMOS Inverter using L-Edit and extract the SPICE code, and the output also verified successfully.

Design Of a 10 Bit Number Controlled Oscillator

Ex.No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To perform the functional verification of the design of a 10 bit number controlled oscillator through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

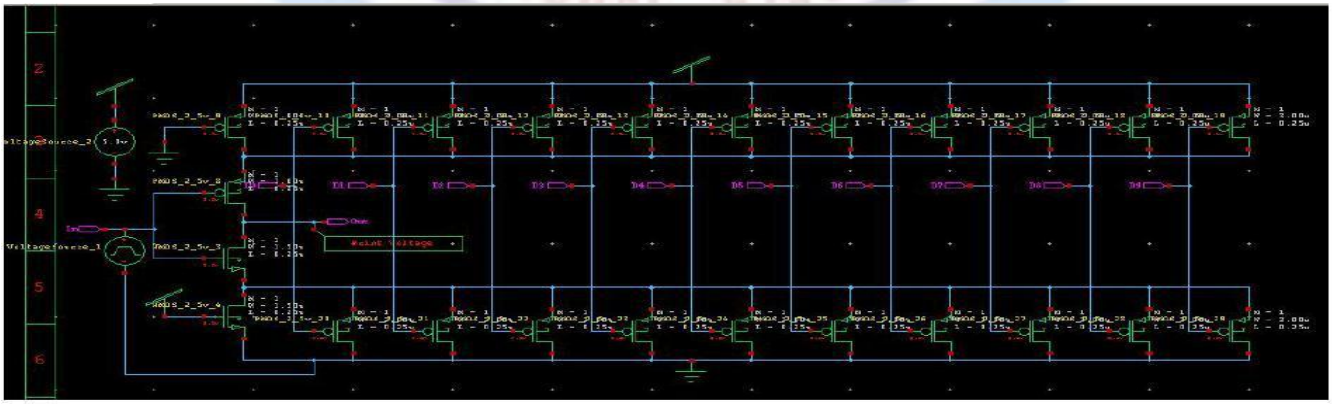
a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	S-Edit using CadanceTool	1

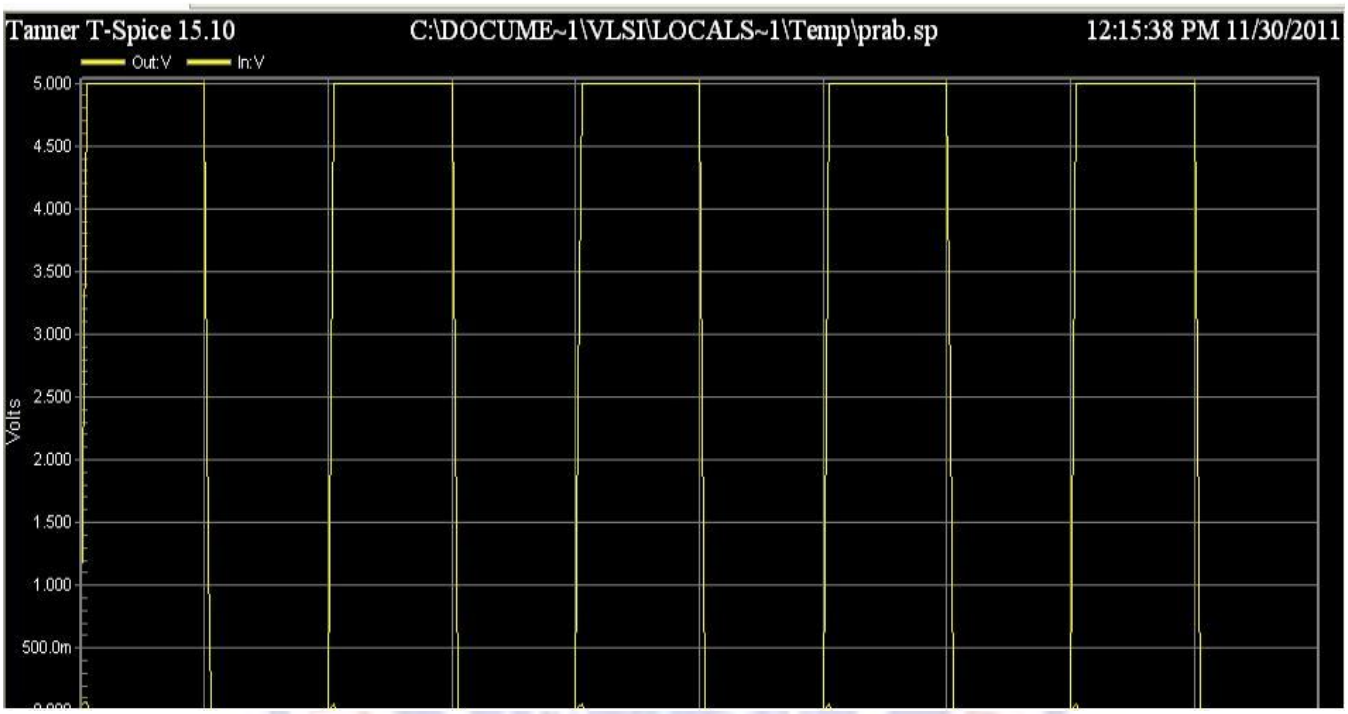
b) Procedure for doing the experiment

S.No	Details of the step
1	Draw the schematic of CMOS Inverter using S-edit.
2	Perform Transient Analysis of the CMOS Inverter
3	Obtain the output waveform from W-edit
4	Obtain the spice code using T-edit.

SCHEMATIC DIAGRAM



SIMULATED WAVEFORM:



RESULT

Thus the the functional verification of the design of a 10 bit number controlled oscillator through schematic entry.

Automatic Layout Generation

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To generate the Layout from the schematic using the Tanner tool and verify the layout using simulation.

FACILITIES REQUIRED AND PROCEDURE

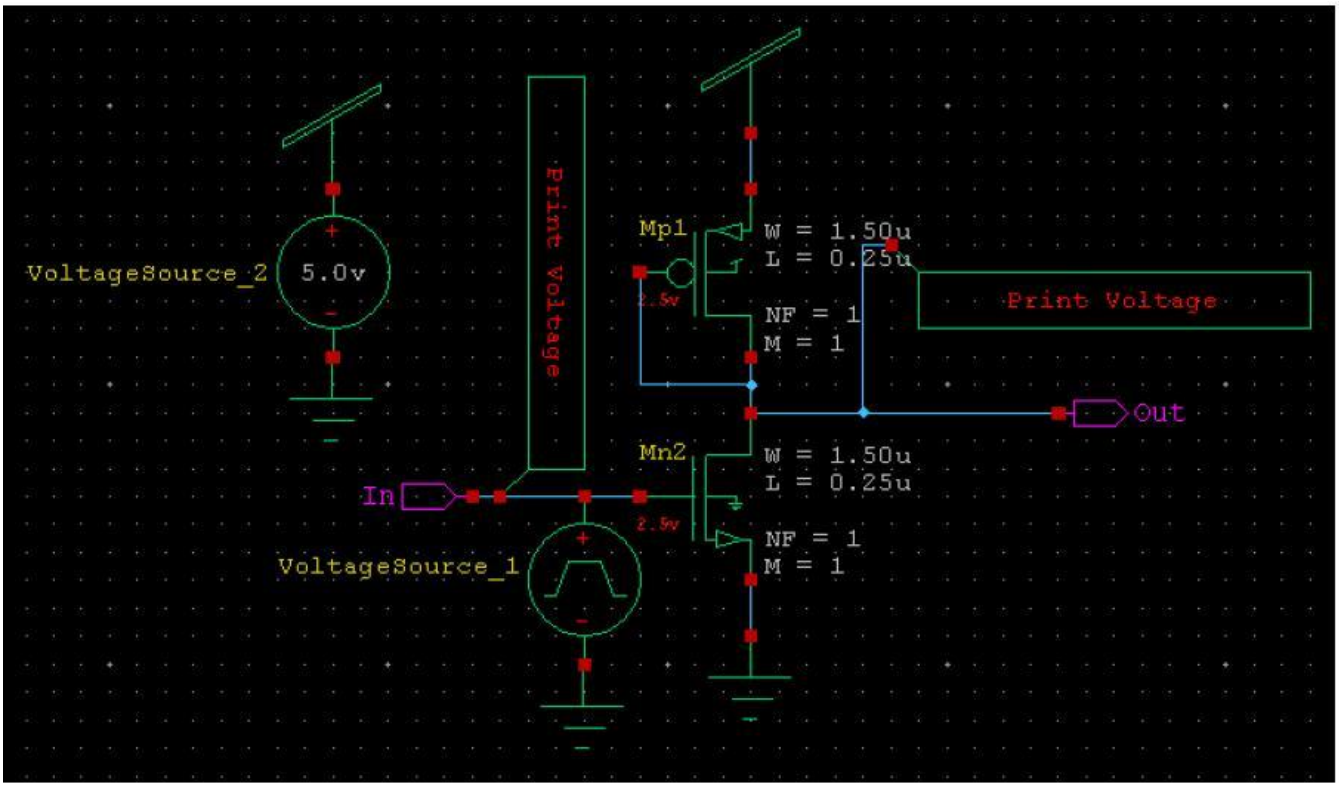
a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	S-Edit, L-Edit using Tanner Tool	1

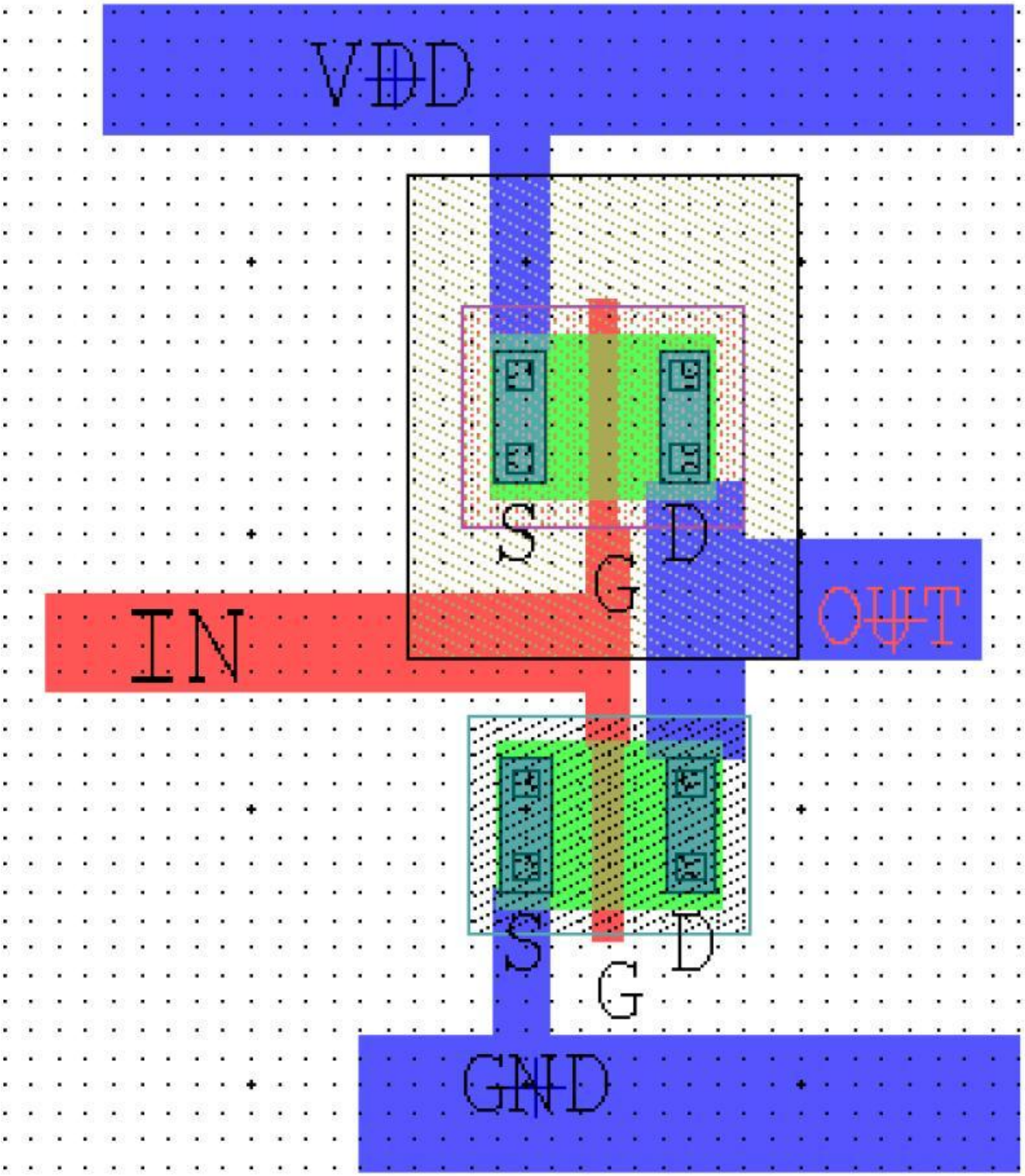
b) Procedure for doing the experiment

S.No	Details of the step
1	Draw the schematic using S Edit and verify the output in W Edit.
2	Extract the schematic and store it in another location
3	Open the L Edit, open the design in Ring VCO
4	Create the new cell
5	Open the schematic file (.sdl) using the SDL Navigator
6	Do the necessary connections as per the design.
7	Name the ports and check the design for the DRC Rules
8	Locate the Destination file in the setup Extract window and extract the layout.
9	Include the Library and the print voltage statements in the net list which is obtained.
10	Verify the layout design using W Edit.

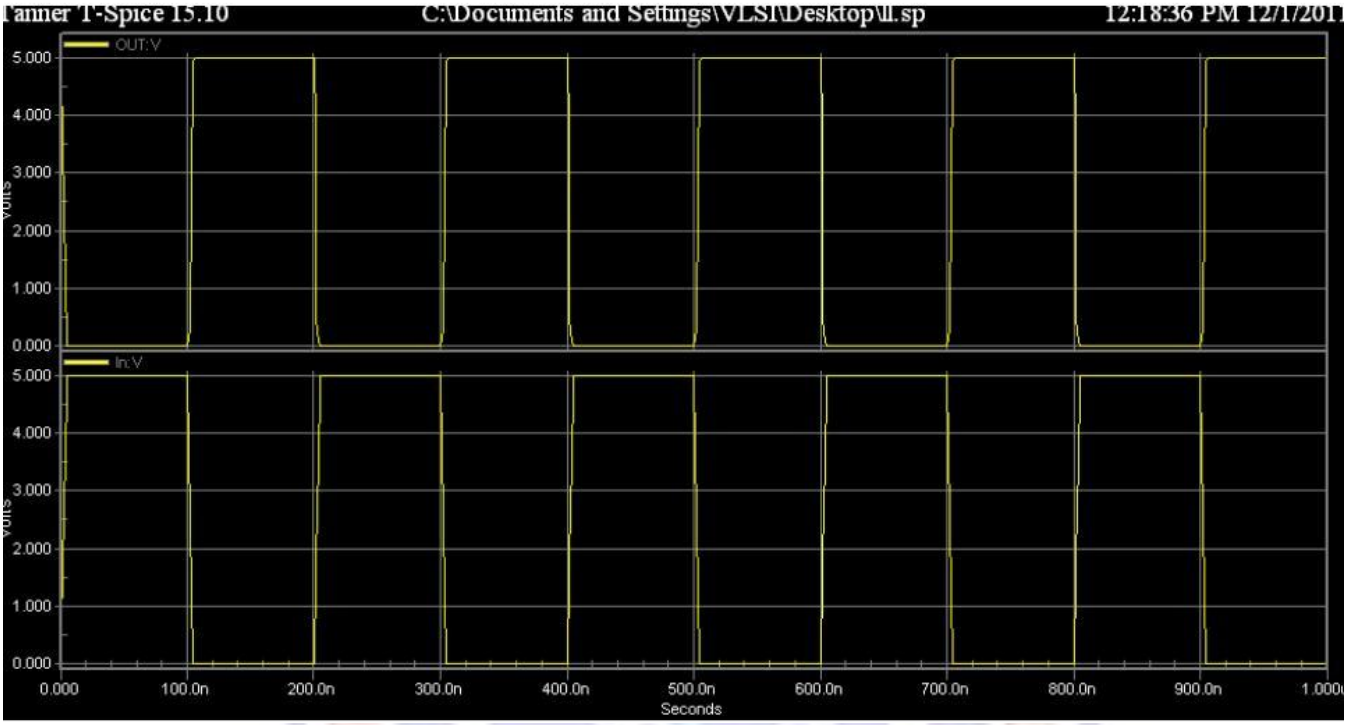
SCHEMATIC DIAGRAM



LAYOUT:



SIMULATED WAVEFORM:



RESULT

Thus the the Layout from the schematic using the Cadance tool and verify the layout using simulation and the output also verified successfully.

Implementation Of Flip-Flops

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To implement Flip-flops using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name.
3	Select Verilog module.
4	Type your verilog coding.
5	Check for syntax.
6	Select the new source of test bench waveform
7	Choose behavioral simulation and simulate it by Xilinx ISE simulator.
8	Verify the output.

D Flip-Flop

```
//    Module Name:   DFF

module DFF(Clock, Reset, d, q);
 input Clock;

input Reset;
 input d;
output q;

reg q;

always@(posedge Clock or negedge Reset)
 if (~Reset) q=1'b0;

else
q=d;

endmodule
```

T Flip-Flop

```
// Module Name:   TFF

module TFF(Clock, Reset, t, q);
 input Clock;

input Reset;

 input t;

output q;

reg q;

always@(posedge Clock , negedge Reset)

if(~Reset) q=0;

else
 if (t) q=~q;
else q=q;
endmodule
```

JK Flip-Flop

```
// Module Name:   JKFF

  module JKFF(Clock, Reset, j, k, q);
   input Clock;
  input Reset;
   input j;
   input k;
   output q;
  reg q;
  always@(posedge Clock, negedge Reset)
  if(~Reset)
  q=0;
  else
   begin
  case({j,k})
  2'b00: q=q;
  2'b01: q=0;
  2'b10: q=1;
  2'b11: q=~q;
  endcase
  end
  endmodule
```

RESULT

Thus the flip-flops program was implemented using tools and the output also verified successfully.

Implementation Of Counters

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To implement Counters using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name.
3	Select Verilog module.
4	Type your verilog coding.
5	Check for syntax.
6	Select the new source of test bench waveform
7	Choose behavioral simulation and simulate it by xilinx ISE simulator.
8	Verify the output.

2- Bit Counter

```
// Module Name: Count2Bit module
Count2Bit(Clock, Clear, out);
input Clock;
input Clear;
output [1:0] out;
reg [1:0]out;
always@(posedge Clock, negedge Clear)
if((~Clear) || (out>=4))
out=2'b00;
else
out=out+1;
endmodule
```

RESULT

Thus the counters program was implemented using tools and the output also verified successfully.

Implementation Of Registers

Ex No: **Date:**

OBJECTIVE OF THE EXPERIMENT

To implement Registers using Verilog HDL.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	Quantity
1	Xilinx Project navigator – ISE 8.1	1

b) Procedure for doing the experiment

S.No	Details of the step
1	Double click the project navigator and select the option File-New project.
2	Give the project name.
3	Select Verilog module.
4	Type your Verilog coding.
5	Check for syntax.
6	Select the new source of test bench waveform
7	Choose behavioral simulation and simulate it by Xilinx ISE simulator.
8	Verify the output.

2 – Bit Register

```
// Module Name:   Reg2Bit

    module Reg2Bit(Clock, Clear, in, out);

    input Clock;

    input Clear; input [0:1] in;

    output [0:1] out;

    reg [0:1] out;

    always@(posedge Clock, negedge Clear) if(~Clear)

    out=2'b00;

    else
    out=in;

    endmodule
```

RESULT

Thus the Registers program was implemented using tools and the output also verified successfully.