

VLSI Design Laboratory

Record

B.E (ECE) – FULL TIME

VI SEMESTER

(For the Academic Year 2021-22)



**DEPARTMENT OF ELECTRONICS & COMMUNICATION
ENGINEERING**

**SRI CHANDRASEKHARENDRASARASWATHI VISWA
MAHAVIDYALAYA**

(Deemed to be University u/s 3 of UGC Act, 1956)

(Accredited with 'A' Grade by NAAC)

Enathur, Kanchipuram – 631561

BONAFIDE CERTIFICATE

Certified to be the Bonafide Record of the work done by

..... (Name)..... (Reg.No.)

..... (Semester) (Branch) in the VLSI Design

Laboratory during the year 2021-2022.

Place:

Date:

.....

Faculty in-charge
Dr. G. Senthil Kumar

.....

Head of the department
Prof. V. Swaminathan

Submitted for the Practical Examination held on

Register No:

Internal Examiner

External examiner

LIST OF EXPERIMENTS

- (a) Study of IC design flow using EDA tools of different vendors
- (b) Introduction to JTAG

FPGA Based Experiments:

1. HDL based **design entry, Test bench creation and simulation** of BCD counters, PRBS generators, Comparators (min 4-bit) / Both multiplier / Carry select adder.
2. **Synthesis, Placement and Routing (P&R) and post P&R simulation** of the components simulated in (Expt. No. 1) above
3. **Critical paths and static timing analysis** results to be identified. **Identify and verify** possible conditions under which the **blocks will fail** to work correctly.
4. **Hardware fusing and testing** of each of the blocks simulated in (Expt. 1). Use of either chipscope feature (Xilinx) or the signal tap feature (Altera) is a must.
5. **Invoke the PLL and demonstrate** the use of the PLL module **for clock generation** in FPGAs.

IC Design Experiments:

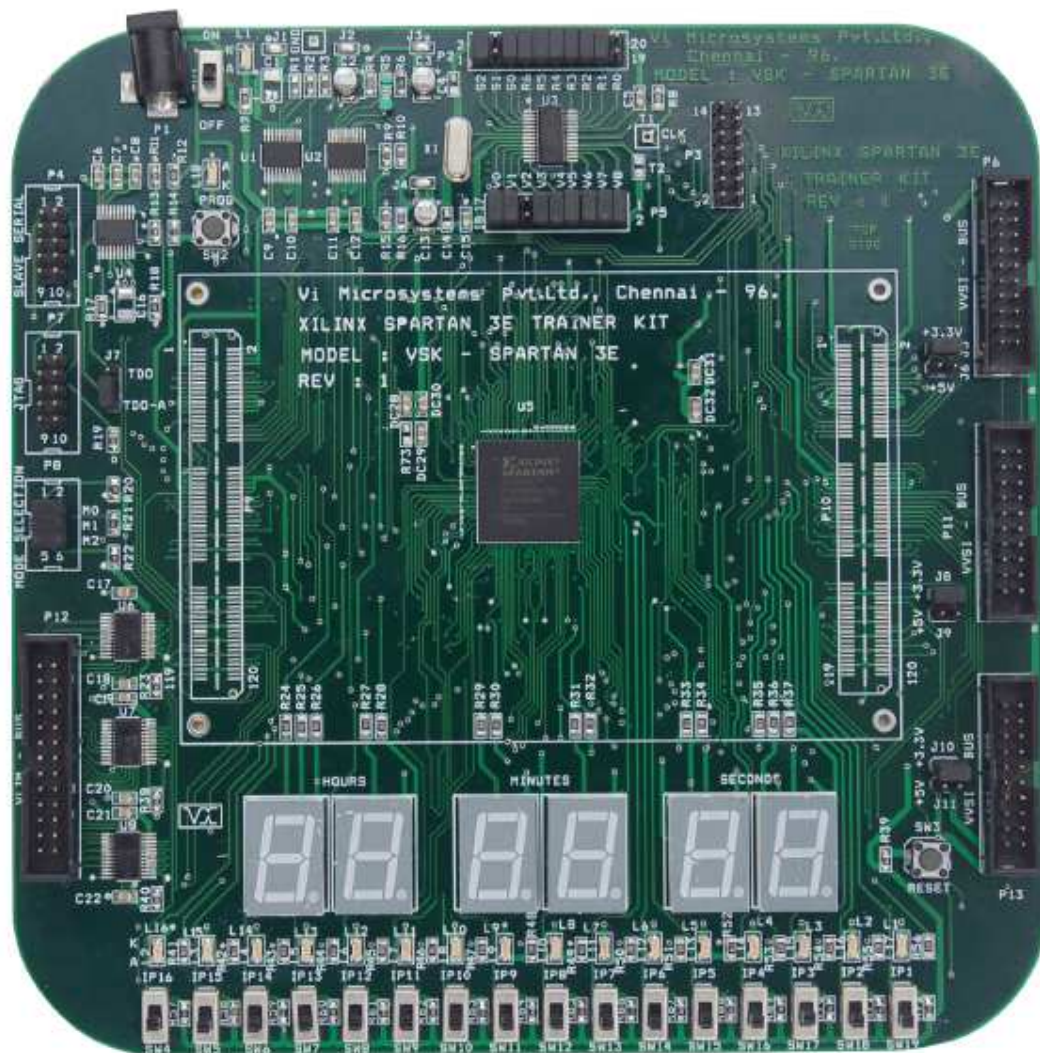
6. **Design and PSPICE simulation** of
 - (a) Simple 5 transistor differential amplifier. Measure gain, BW, output impedance, ICMR, and CMRR.
 - (b) Ring Oscillator
7. **Layout generation, DRC and LVS Checking, Parasitic Extraction and Resimulation** of CMOS Inverter.
8. **Synthesis and Standard cell-based design** of a circuit simulated in (Expt. 6-b) above - Synthesis principles, Logical Effort, Interpreting Scripts, Constraints and Library preparation and generation, Boolean Optimization, Optimization for Area, Power.
9. For Expt. 6-b above, Floor Planning, Placement and Routing (P&R), Power and Clock Routing, and post P&R simulation
10. Static Timing analyses procedures and constraints. Critical path considerations.
11. DFT - Scan chain insertion / Clock Tree Synthesis / Stick diagrams

Index

EX.NO	NAMEOF THE EXPERIMENTS	PAGE NO	SIGN
	CYCLE- I Tools- XILINX ISE 14.1		
1	Design Entry and Simulation of Combinational Circuits		
2	Place and Route and Post Place & Route Simulation		
3	Design and FPGA Implementation of Combinational Circuits		
4	Design and FPGA Implementation of Sequential Circuits		
5	Analysis of Area, Power and Delay For Sequential Circuits		
6	Invoke PLL to generate Real Time Clock		
	CYCLE-II Tools – LT-SPICE and ELECTRIC VLSI System Design EDA		
7	Ring Oscillator		
8	Differential Amplifier		
9	CMOS Inverter		
10	Layout CMOS Inverter		
11	CMOS Inverter – Place and Route		
12	Layout CMOS NAND Gate		
	Study Experiments Tools- Synopsys		
13	Static Timing Analysis		
14	DfT-Scan Chain Insertion		

Procedure for simulation and implementation of EDA tool using XILINX SPARTAN 3E TRAINER KIT

The Spartan-3E Trainer Kit is a demonstration platform intended to become familiar with the new features and availability of the Spartan-3E FPGA family. This Kit provides a easy-to-use development and evaluation platform for Spartan-3E FPGA designs.



Slide Switch connections with FPGA - INPUT PIN

SWITCHES	FPGA PINS
SW4	T14
SW5	T12
SW6	T9
SW7	T7
SW8	T2
SW9	G12
SW10	H1
SW11	R3
SW12	N11
SW13	N3
SW14	M13
SW15	M7
SW16	M3
SW17	K4
SW18	J12
SW19	J11

OUTPUT PINS

LEDS	FPGA PINS
L16	R1
L15	R2
L14	K3
L13	T4
L12	T5
L11	R6
L10	T8
L9	R10
L8	N10
L7	P12
L6	N9
L5	N12
L4	P13
L3	R13
L2	T13
L1	P14

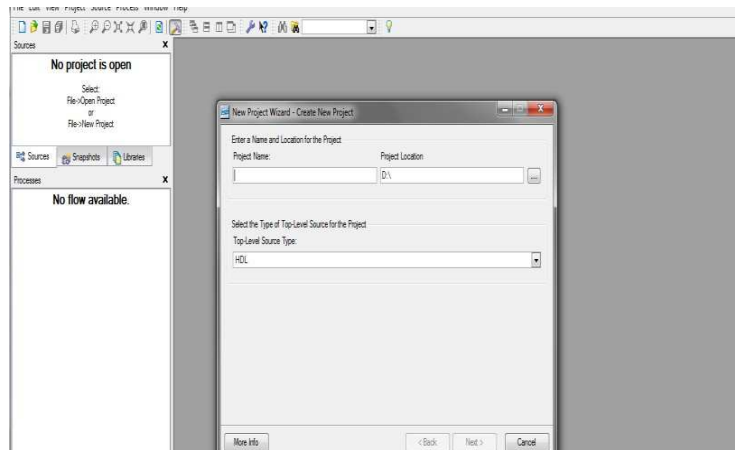
Procedure for simulation and implementation of Xilinx tool and FPGA

STEP1:

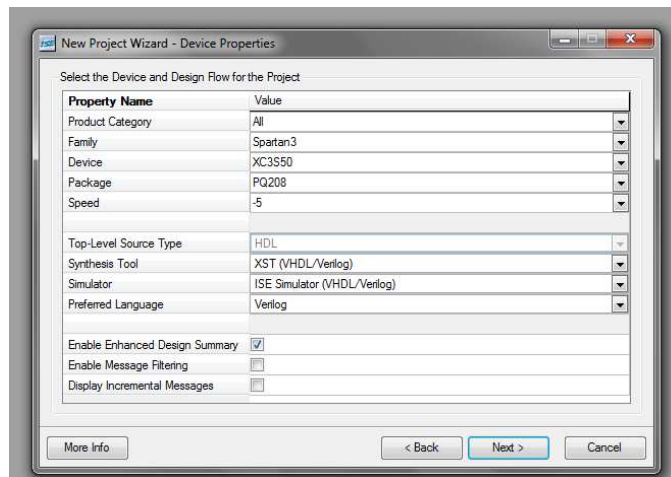
Click Xilinx ISE 9.1

STEP2:

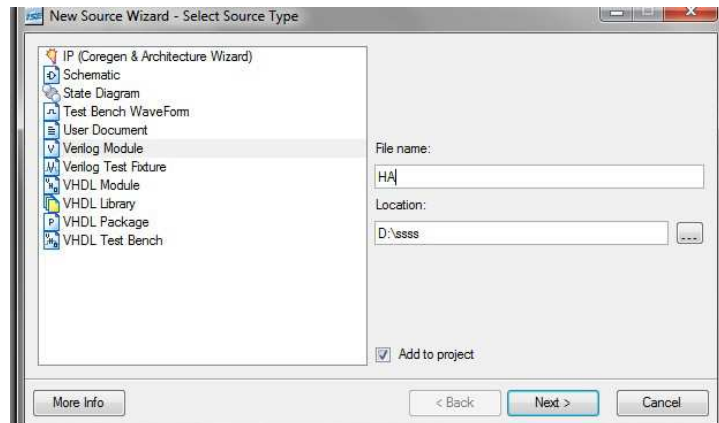
File->New project and type the project name and check the top level source type as HDL



STEP3: Check the device properties and click next

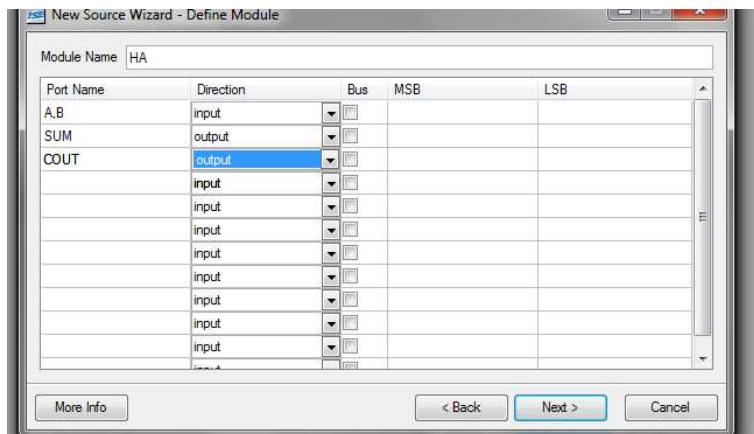


STEP4: Click New Source And Select the Verilog Module and then give the filename



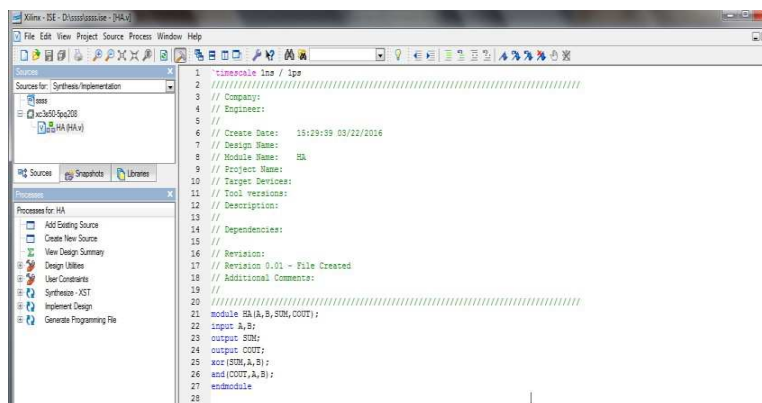
STEP5:

Select the Input, Output port names and click finish.

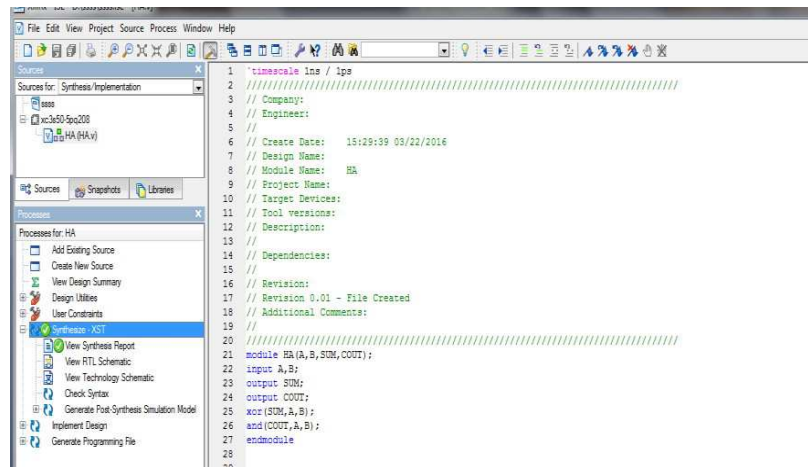


STEP6:

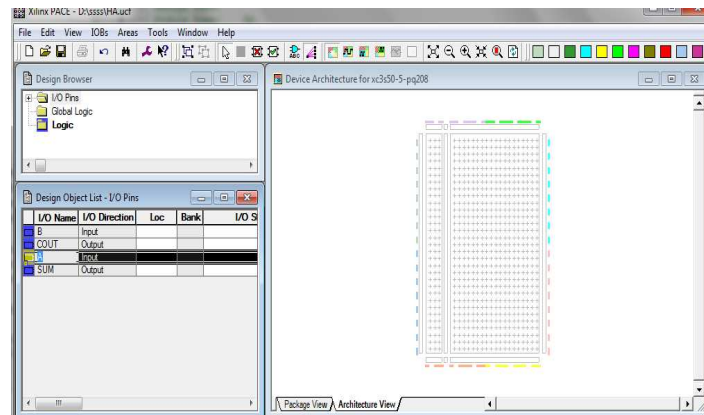
Type the program and save it



STEP7: Check the synthesis XST and check syntax

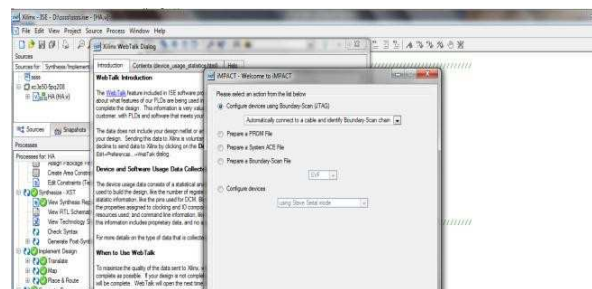


STEP8: Select user constraints-> assign package pins, set port numbers and save it then select IO Bus delimiters as XST default<->->click ok



STEP9:

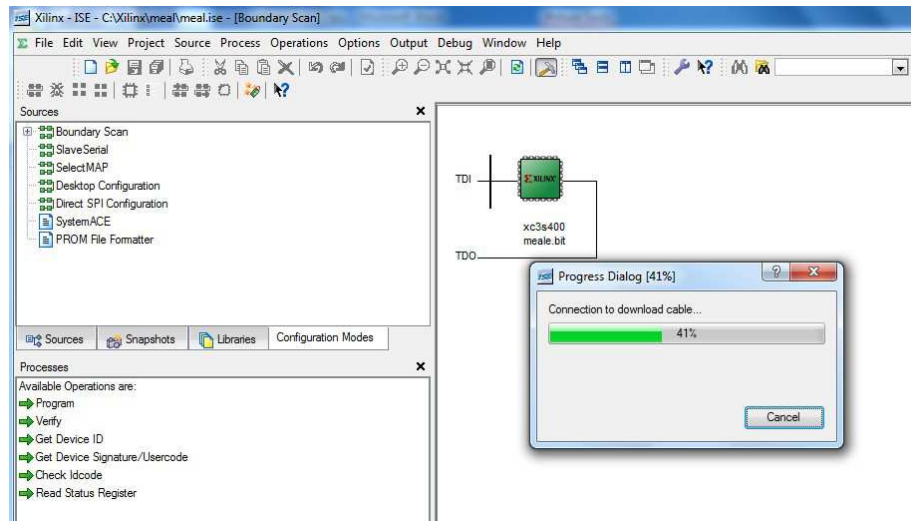
Double click implement design and click generate programming file->configure device (impact)->finish then select bit file



STEP10:

Rightclickonthexc3s400figure->program-

>filenamethenclickfinishandFinallycheckthefunctionalityinhardware



EXP NO: 1
Date:

Design Entry and Simulation of Combinational Circuits

AIM:

To write a Verilog code for the 4-bit Ripple carry adder and 4-bit Comparator and simulate it using Xilinx project navigator.

APPARATUS REQUIRED:

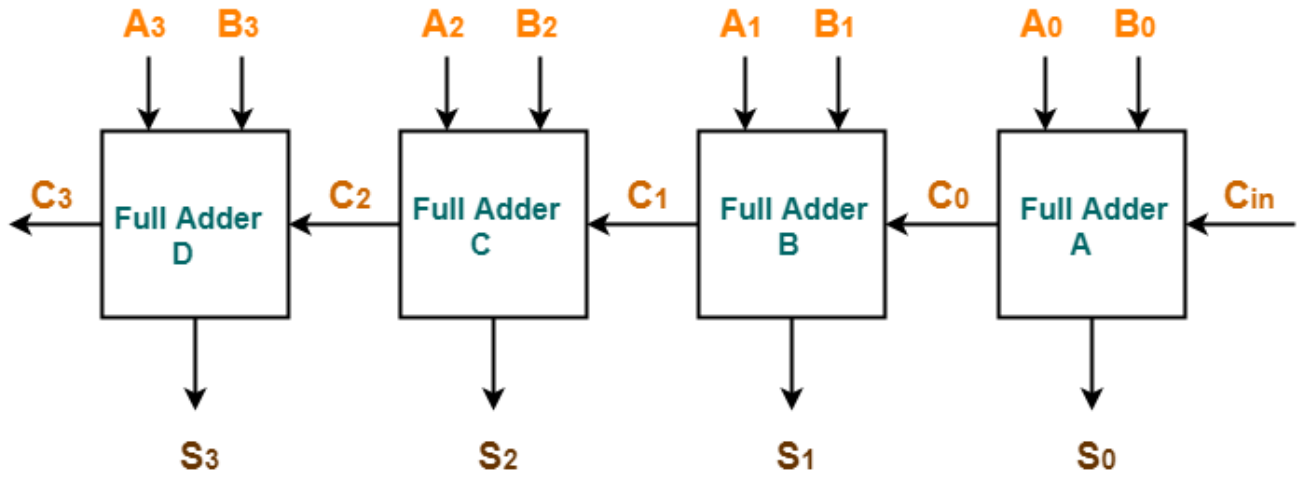
S.No	Name of the equipment/ software	Quantity
1.	PC with Windows	1
2.	Xilinx Project navigator	1

PROCEDURE:

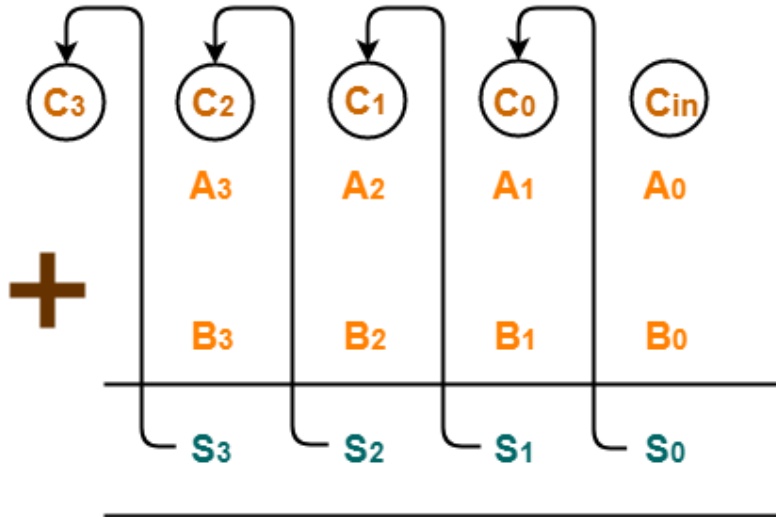
1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the filename → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Click on the symbol of FPGA device and then right click → click on new source.
10. Select the Test Bench Waveform and give the filename → select entity click next and finish.
11. Select the desired parameters for simulating your design. In this case combinational circuit and simulation time click finish.
12. Assign all input signals using just click on graph and save file.
13. From the source process window. Click Behavioral simulation from drop-down menu
14. Select the test bench file (.tbw) and click process button → double click the Simulation Behavioral Model
15. Verify your design in wave window by seeing behavior of output signal with respect to input signal

4-Bit Ripple Carry Adder

Block Diagram:



4-bit Ripple Carry Adder



Adding two 4-bit Numbers

CODING :

```

moduleripple_carry_adder(a, b, cin, sum, cout);
input [03:0] a;
input [03:0] b;
inputcin;
output [03:0] sum;
outputcout;
wire [2:0]c;
fulladd a1(a[0],b[0],cin, sum[0],c[0]);
fulladd a2(a[1],b[1],c[0],sum[1],c[1]);
fulladd a3(a[2],b[2],c[1],sum[2],c[2]);
fulladd a4(a[3],b[3],c[2],sum[3],cout);
endmodule

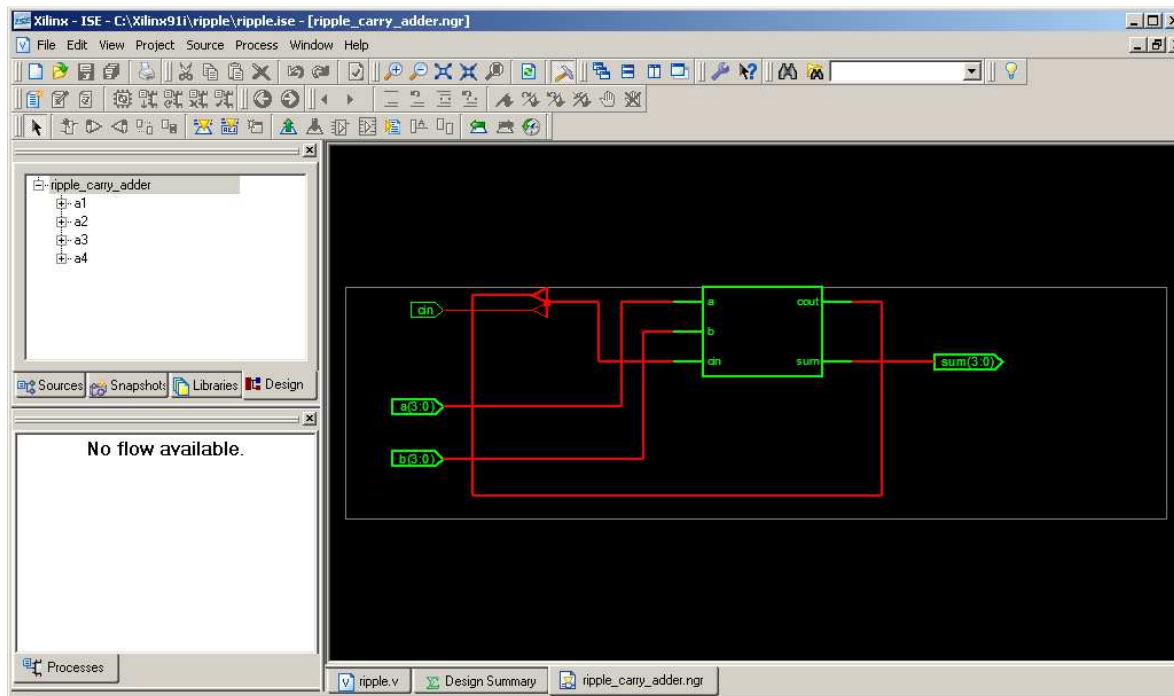
```

```

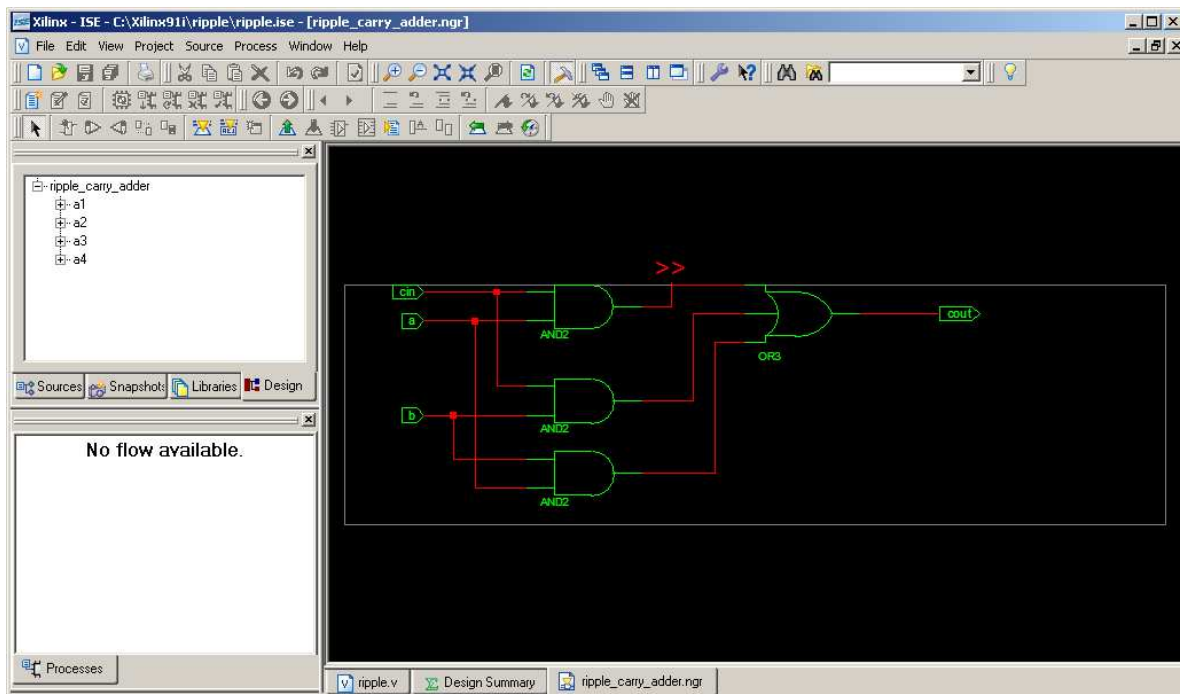
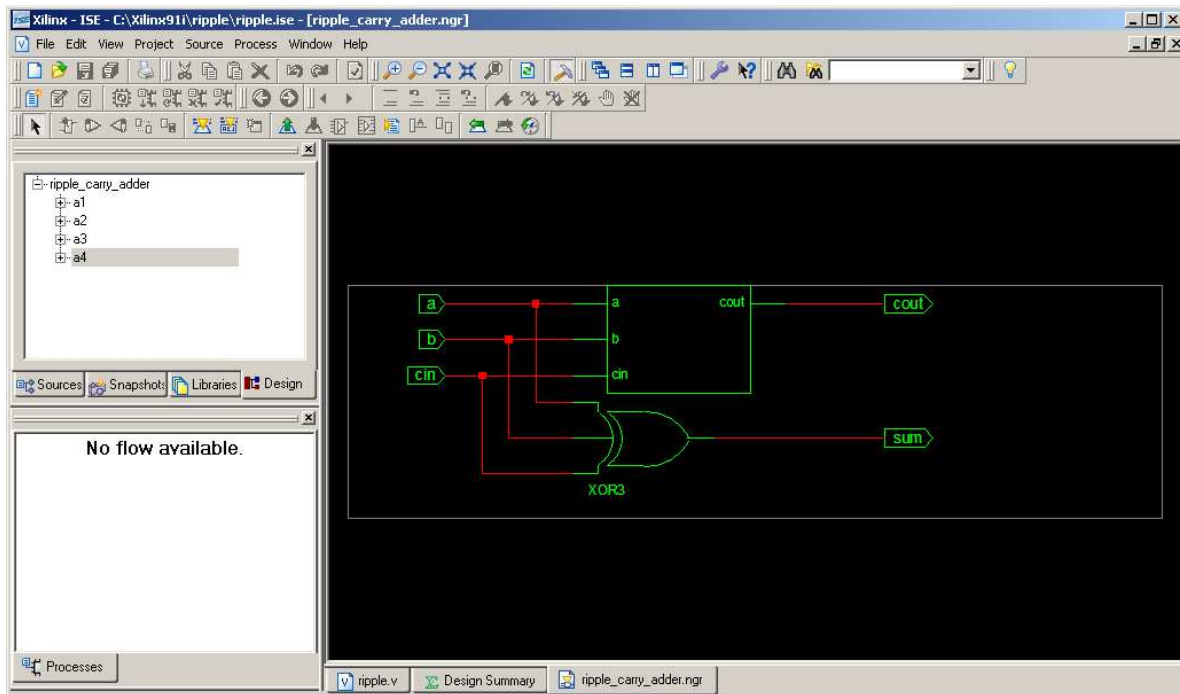
modulefulladd(a,b,cin,sum,cout);
inputa,b,cin;
outputsum,cout;
assign sum=(a^b^cin);
assigncout=((a&b)|(b&cin)|(a&cin));
endmodule

```

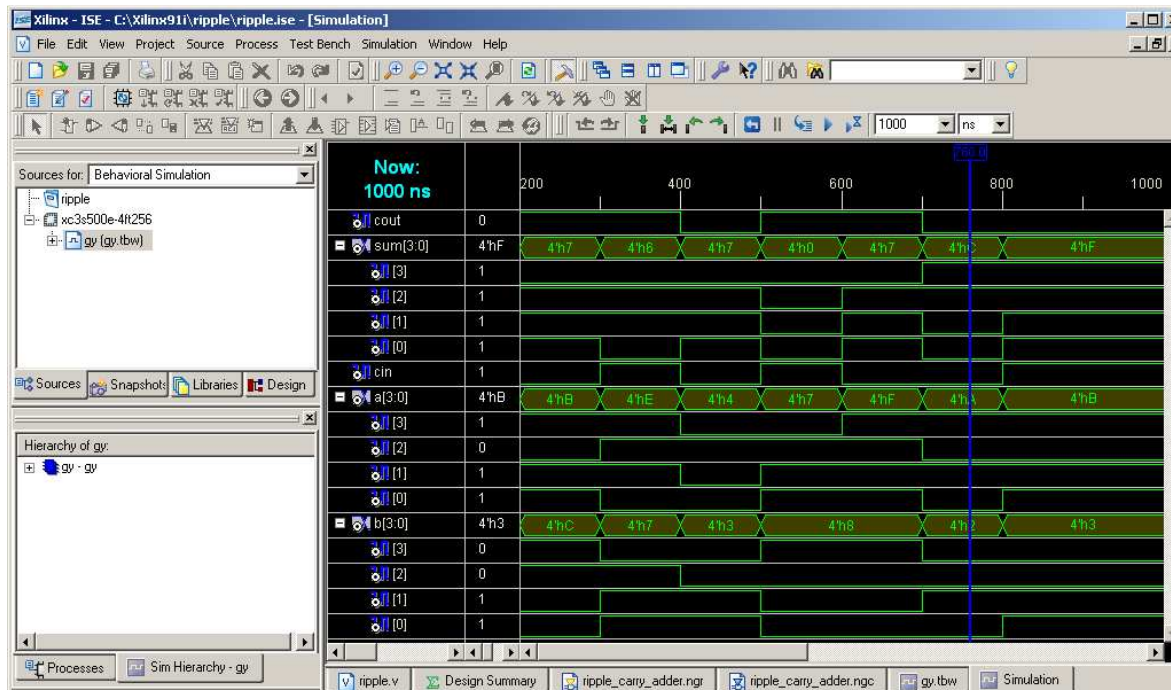
RTL SCHEMATIC:



TECHNOLOGY SCHEMATIC:



SIMULATION OUTPUT:



RESULT:

Thus the Verilog code for 4 bit Ripple Carry Adder is simulated using Xilinx project navigator.

AIM:

To synthesis 4- Bit Comparator and then Place& Route and Post Place & Root using Implementation option available in Xilinx project navigator.

APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	XilinxProject navigator	1

Theory:

- Back annotation is the translation of a routed or fitted design to a timing simulation netlist.
- To define the behavior of the FPGA, a hardware description language (HDL) or a schematic design methods are used. Common HDLs are VHDL and Verilog. Then, using an electronic design automation (EDA) tool, a technology-mapped net list is generated.
- The net list can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA vendor's proprietary place-and-route software.
- The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated is used to (re)configure the FPGA.
- In an attempt to reduce the complexity of designing in HDLs, which have been compared to the equivalent of assembly
- In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process.
- Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results.
- Then, after the synthesis engine has mapped the design to a net list, the net list is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors.
- Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the net list.
- Place & Route, the process of optimization of logic cells for effective utilization of FPGA area and the speed of operation, is used to modify and infer the following:
 1. Re-assignment of Pins
 2. Re-location of Slices
 3. Run time minimization

Procedure:

1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the file name → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Synthesis your design, from the source window select, synthesis/implementation from the window Now double click the Synthesis -XST
10. After Synthesis you assign the Pin Value for your design so, → double click the Assign Package Pins
11. Enter the Pin value for your input and output signals. if you want see your Pin assignment in FPGA zoom in Architecture View or Package View
12. Check the Pins in FPGA. Save file as XST Default click ok and close the window
13. Design Implementation begins with the mapping or fitting of a logical design file to a specific device and is complete when the physical design is successfully routed and a bit stream is generated. Double Click Implementation Design.
14. After finishing the Implementation, you can view the Implementation report.
15. After implementation you see Design Summary, you get the all details about your design. If you want edit the place and route double click View/Edit placed design
16. Check where your IOs are placed in FPGA. And zoom to view how Pins are placed in FPGA. You can see where your pins are placed
17. Just double click View/Edit Routed Design to view interconnection wires and blocks
18. Click the pin to see where its placed in FPGA. And Zoom particular area to see Place and Routing.
19. If required to change the place of the design, click and trace to another slice. View changed place and route of the design
20. Double click Back annotated Pin Location. Once back annotation is completed, constraint file is generated.

Block Diagram:



Truth Table

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

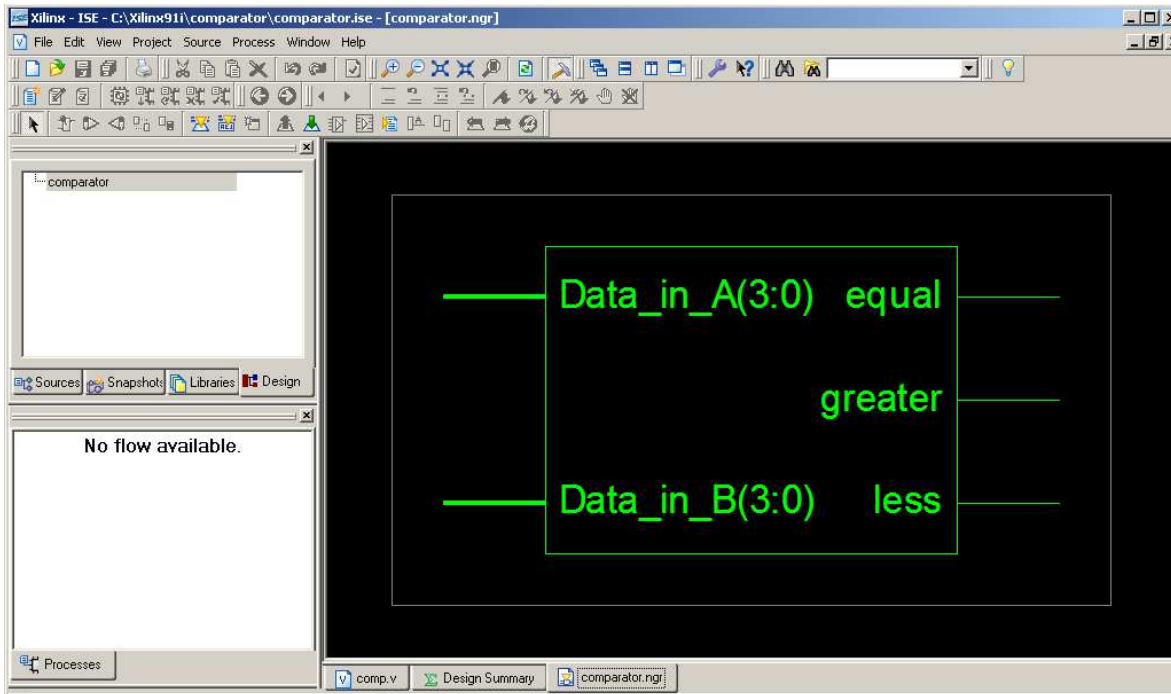
```

//declare the Verilog module - The inputs and output signals.
module comparator(
    Data_in_A, //input A
    Data_in_B, //input B
    less,      //high when A is less than B
    equal,     //high when A is equal to B
    greater    //high when A is greater than B
);

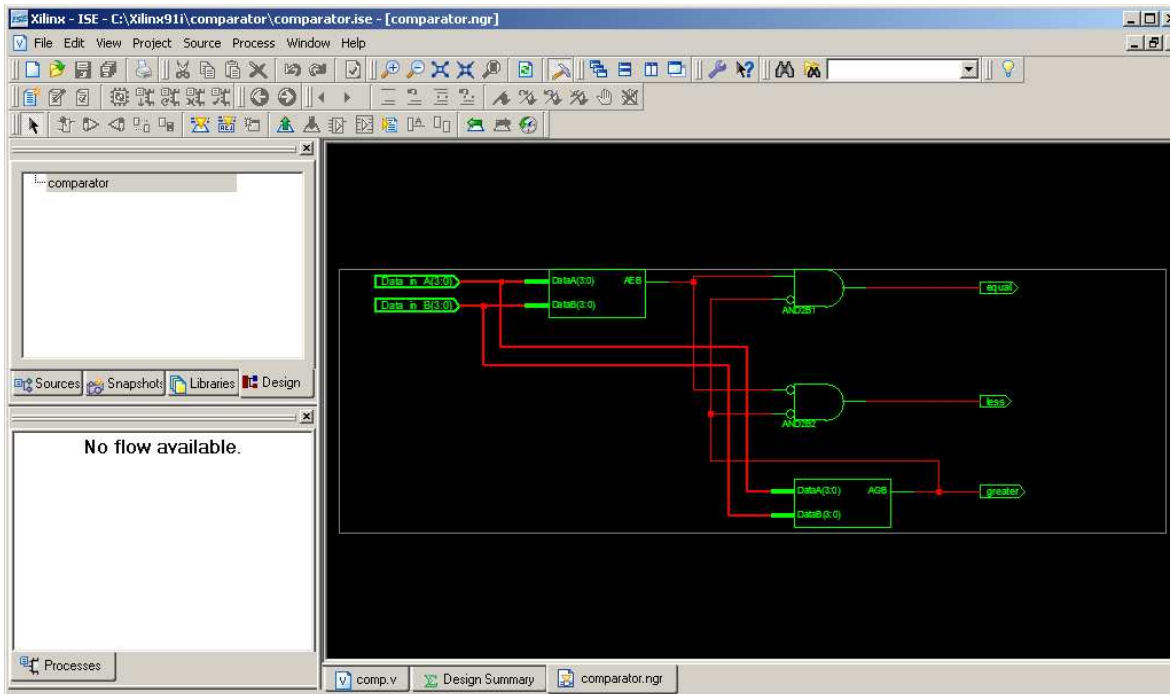
//what are the input ports.
input [3:0] Data_in_A;
input [3:0] Data_in_B;
//What are the output ports.
output less;
output equal;
output greater;
//Internal variables
reg less;
reg equal;
reg greater;

    • //When the inputs and A or B are changed execute this block
      always @(Data_in_A or Data_in_B)
        begin
          if(Data_in_A > Data_in_B) begin //check if A is bigger than B.
            less = 0;
            equal = 0;
            greater = 1; end
          else if(Data_in_A == Data_in_B) begin //Check if A is equal to B
            less = 0;
            equal = 1;
            greater = 0; end
          else begin //Otherwise - check for A less than B.
            less = 1;
            equal = 0;
            greater = 0;
          end
        end
      end
endmodule

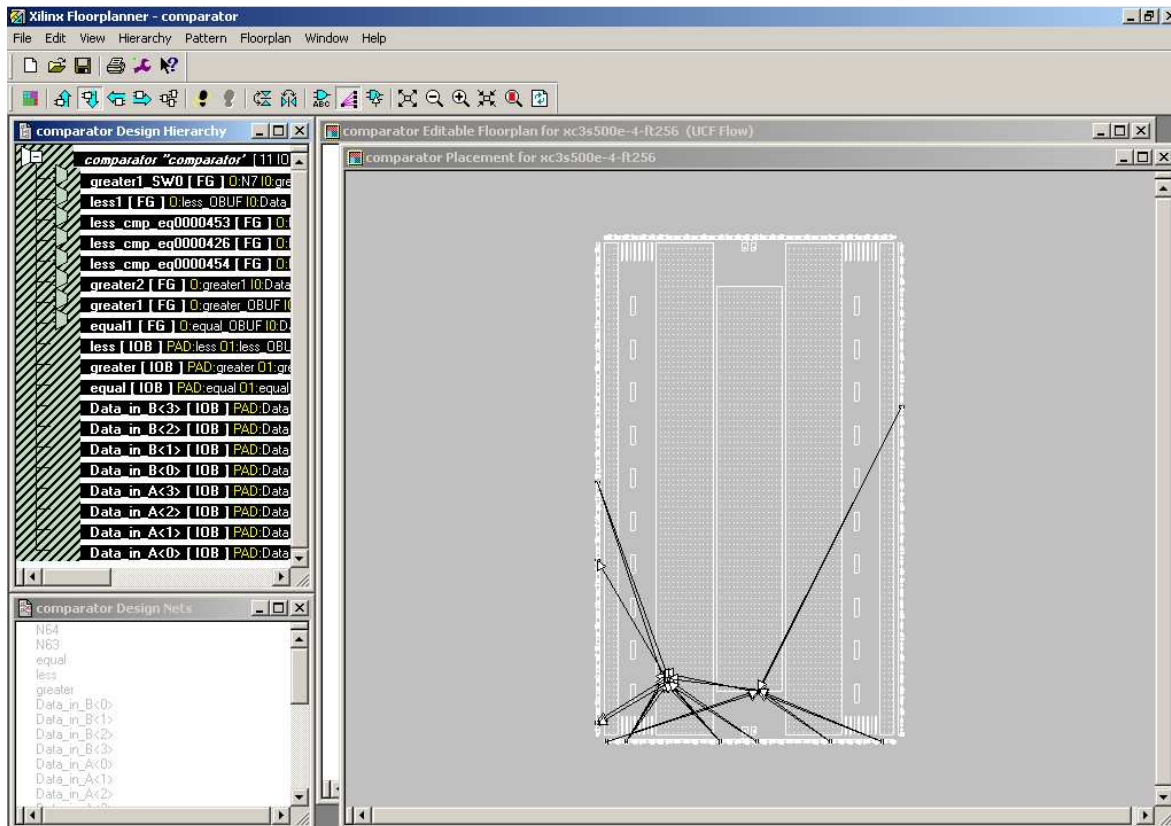
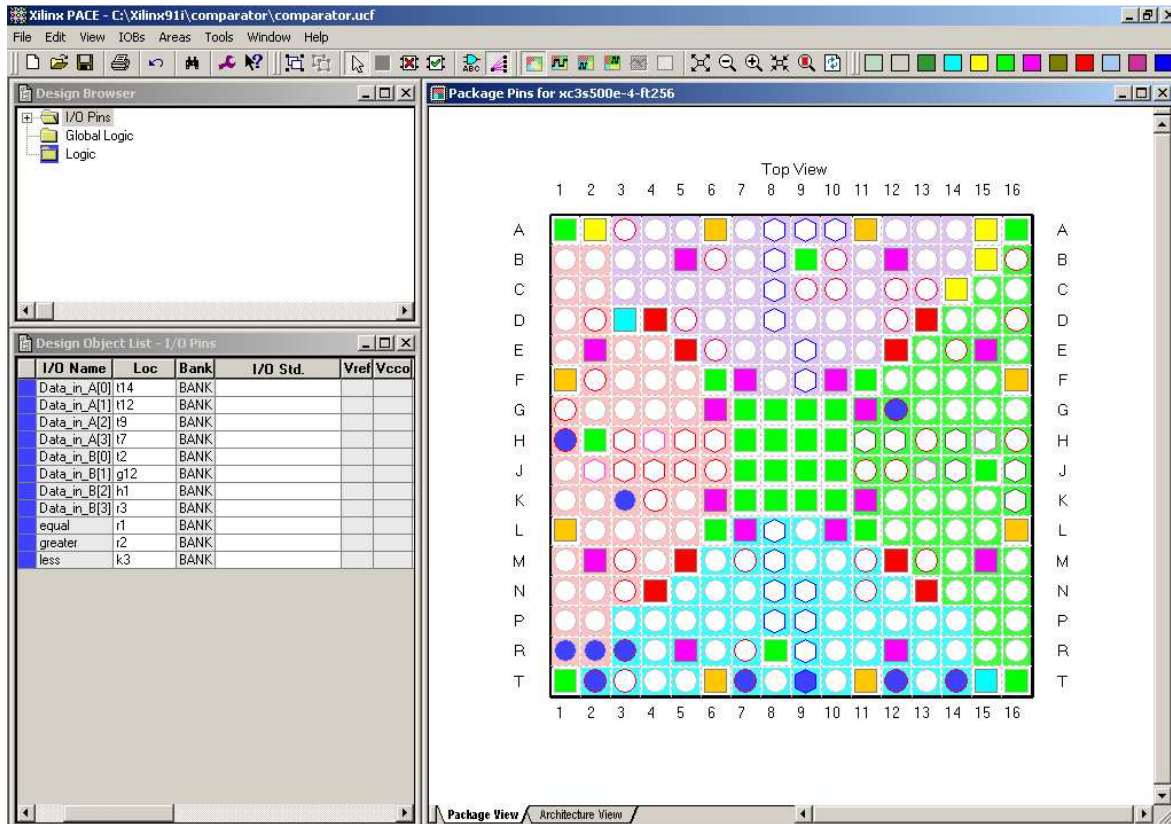
```



TECHNOLOGY SCHEMATIC:



PLACE AND ROUTE:



RESULT:

Thus, the Place and Route and Post Place and Route using Implementation options available in Xilinx project navigator were synthesized for 4-bit Comparator.

EXP NO: 3
Date:

Design and FPGA Implementation of Combinational Circuits

AIM:

To design and implement Booth Multiplier and Carry select Adder in FPGA Spartan 3E Trainer kit using Xilinx project navigator.

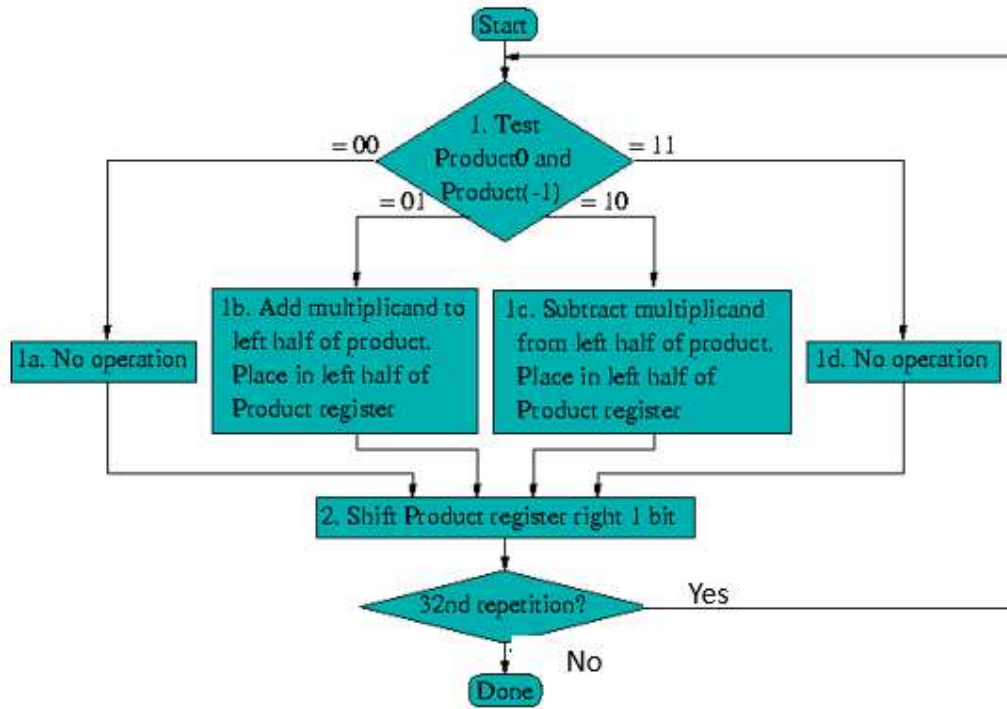
APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	XilinxProject navigator	1

PROCEDURE:

1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the file name → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → Synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Synthesis your design, from the source window select, synthesis/implementation from the window Now double click the Synthesis -XST.
10. After Synthesis, Click on the symbol of FPGA device and Right click and select New Source, Select Implementation Constraints File and type file name and click next.
11. Type the Net list and click save.
12. Implement the design by double clicking Implement design in the process window.
13. Then double click Generate Programming File, Double click Configure Target Device and click OK.
14. Double click Create PROM File in the ISE iMPACT window, Select Storage Target Device as Xilinx Flash PROM and click forward.
15. Add storage Device as xcf01s [2 M] and click forward, Type Output File Name and Location and click OK.
16. Select the corresponding .bit file and click Open, Click No to Add Another Device and Click OK.
17. Double click Generate File.
18. Double click Boundary Scan and Right click on the window and select Initialize Chain, Now Select the corresponding .mcs file and click open.
19. Click OK in the Device Programming Properties window, Download the Program on to the kit by Right clicking on the device icon and select program.
20. Verify the output in the target device.

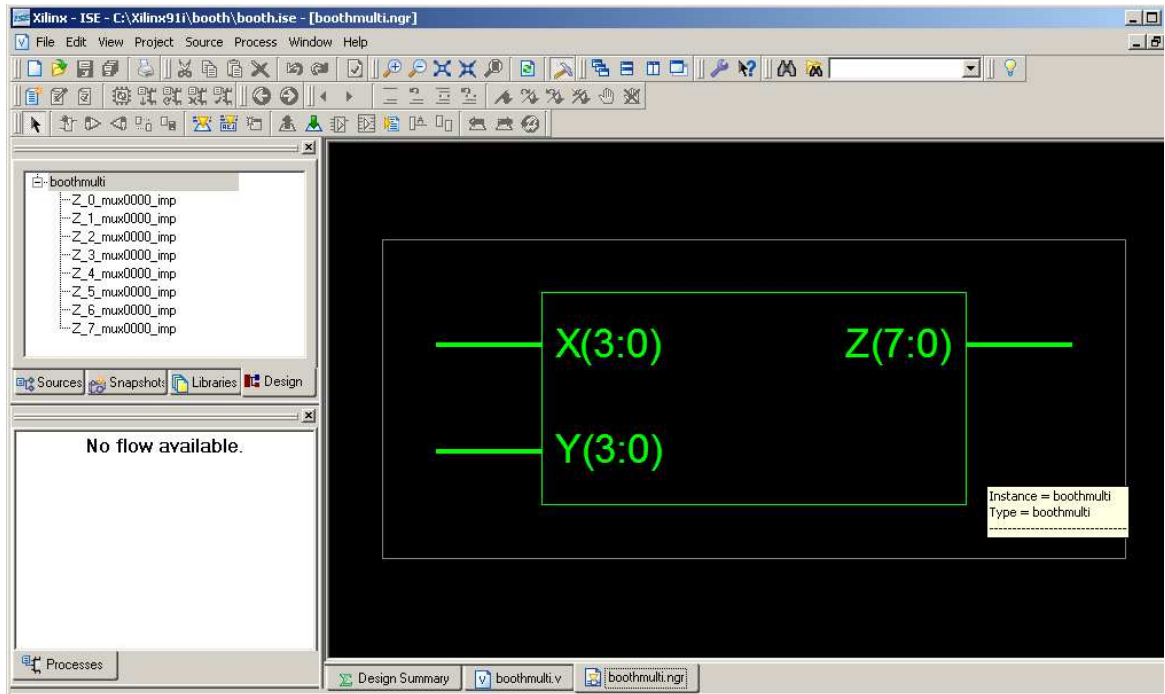
BOOTH ALGORITHM:



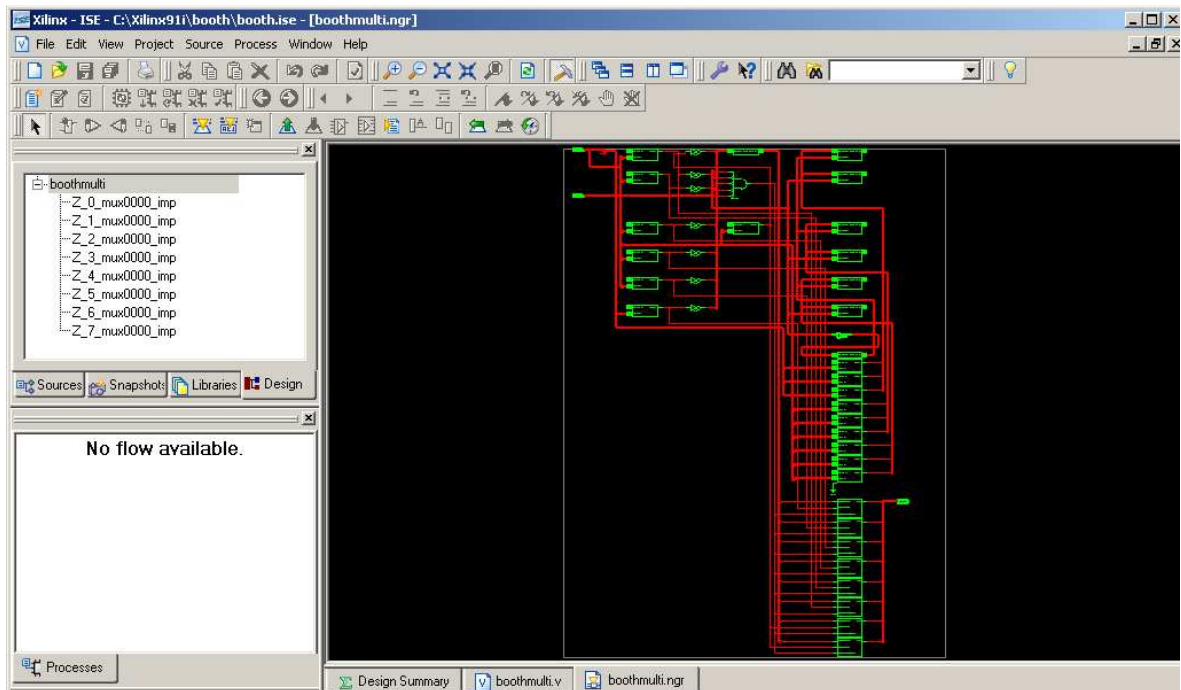
CODING:

```
module boothmulti(X, Y, Z);
input signed [3:0] X, Y;
output signed [7:0] Z;
reg signed [7:0] Z;
reg [1:0] temp;
integer i;
reg E1;
reg [3:0] Y1;
always @ (X, Y)
begin
    Z = 8'd0;
    E1 = 1'd0;
    for (i = 0; i < 4; i = i + 1)
    begin
        temp = {X[i], E1};
        Y1 = - Y;
        case (temp)
        2'd2 : Z [7 : 4] = Z [7 : 4] + Y1;
        2'd1 : Z [7 : 4] = Z [7 : 4] + Y;
        default : begin end
        endcase
        Z = Z >> 1;
        Z[7] = Z[6];
        E1 = X[i];
    end
    if (Y == 4'd8)
    begin
        Z = - Z;
    end
end
endmodule
```

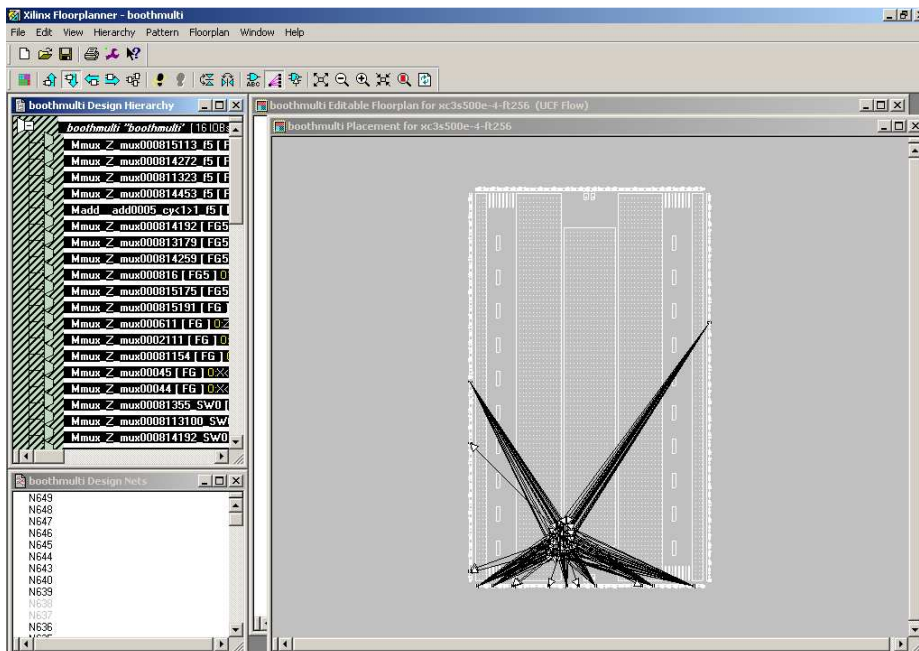
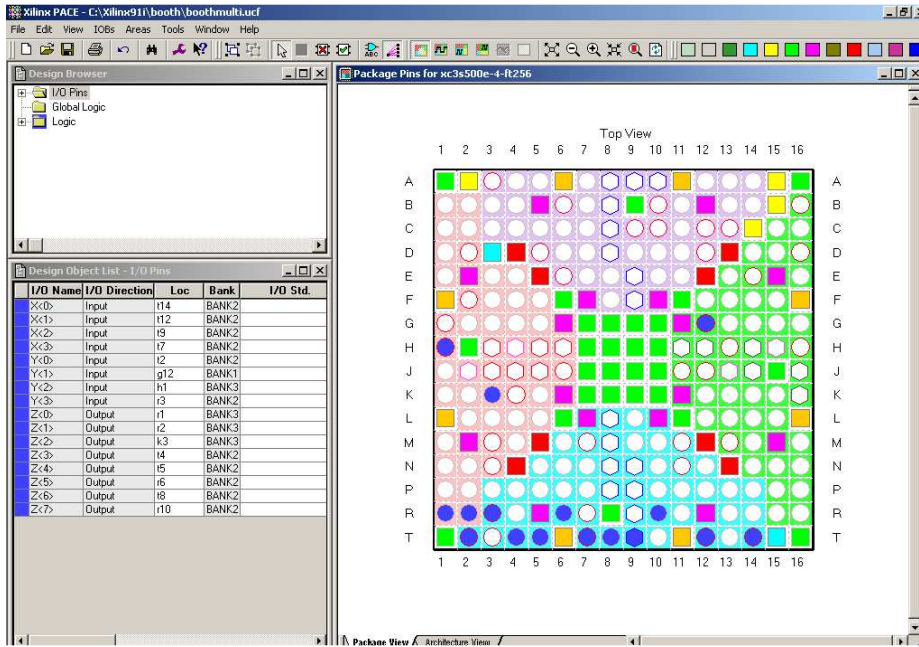
RTL SCHEMATIC:



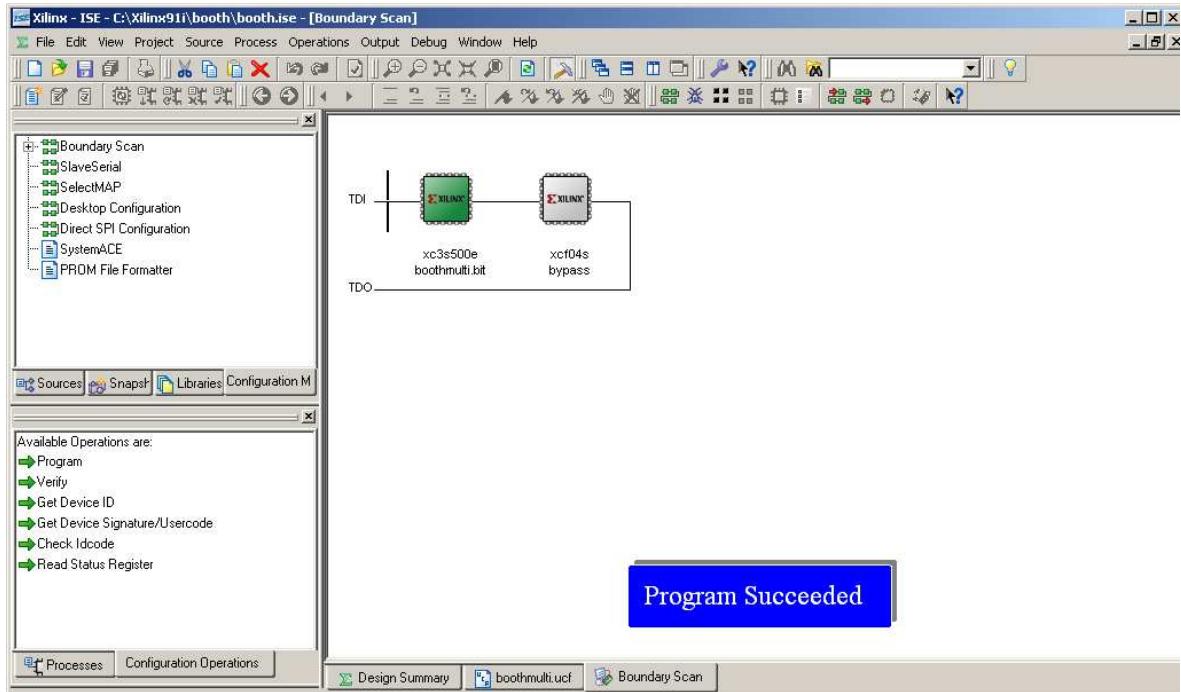
TECHNOLOGY SCHEMATIC:



PLACE AND ROUTE:

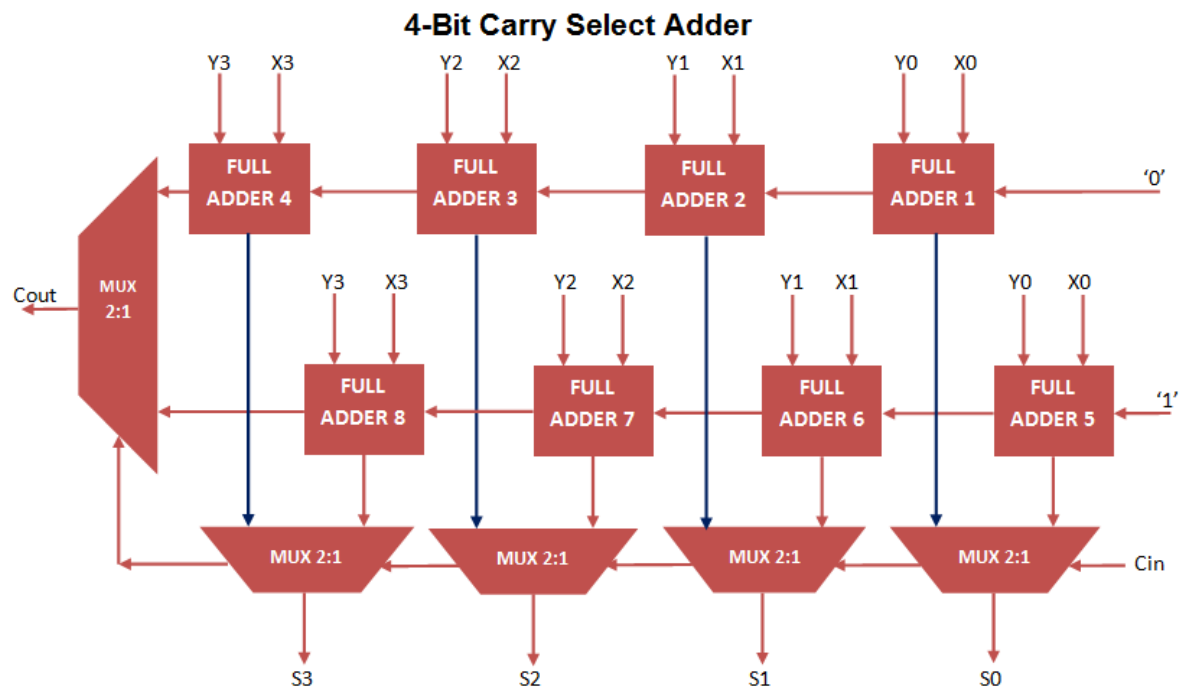


HARDWARE FUSING:



CARRY SELECT ADDER:

BLOCK DIAGRAM:



CODING:

```
module carry_select_adder
    ( input [3:0] A,B,
    input cin,
    output [3:0] S,
    output cout
    );

    wire [3:0] temp0,temp1,carry0,carry1;

    //for carry 0
    fulladder fa00(A[0],B[0],1'b0,temp0[0],carry0[0]);
    fulladder fa01(A[1],B[1],carry0[0],temp0[1],carry0[1]);
    fulladder fa02(A[2],B[2],carry0[1],temp0[2],carry0[2]);
    fulladder fa03(A[3],B[3],carry0[2],temp0[3],carry0[3]);

    //for carry 1
    fulladder fa10(A[0],B[0],1'b1,temp1[0],carry1[0]);
    fulladder fa11(A[1],B[1],carry1[0],temp1[1],carry1[1]);
    fulladder fa12(A[2],B[2],carry1[1],temp1[2],carry1[2]);
    fulladder fa13(A[3],B[3],carry1[2],temp1[3],carry1[3]);

    //mux for carry
    multiplexer2 mux_carry(carry0[3],carry1[3],cin,cout);
    //mux's for sum
    multiplexer2 mux_sum0(temp0[0],temp1[0],cin,S[0]);
    multiplexer2 mux_sum1(temp0[1],temp1[1],cin,S[1]);
    multiplexer2 mux_sum2(temp0[2],temp1[2],cin,S[2]);
    multiplexer2 mux_sum3(temp0[3],temp1[3],cin,S[3]);

    endmodule
```

```
modulefulladder
    ( inputa,b,cin,
    outputsum,carry
    );
```

```
assign sum = a ^ b ^ cin;
assign carry = (a & b) | (cin& b) | (a &cin);
```

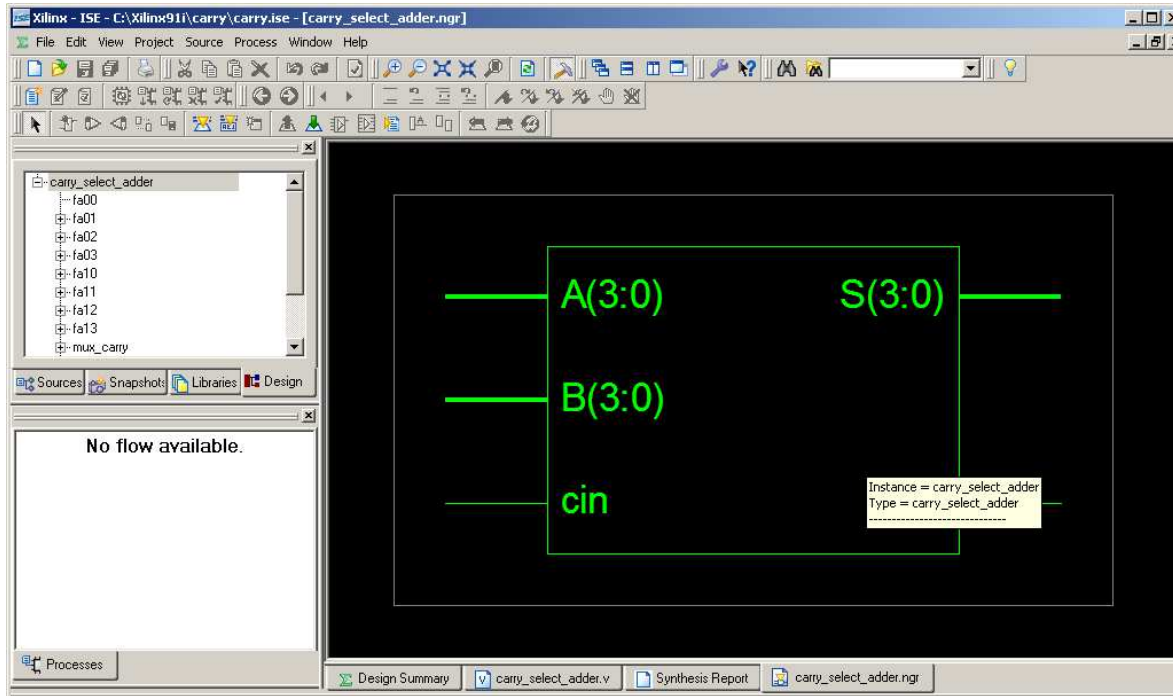
```
endmodule
```

```
module multiplexer2
    ( input i0,i1,sel,
    outputregbitout
    );
```

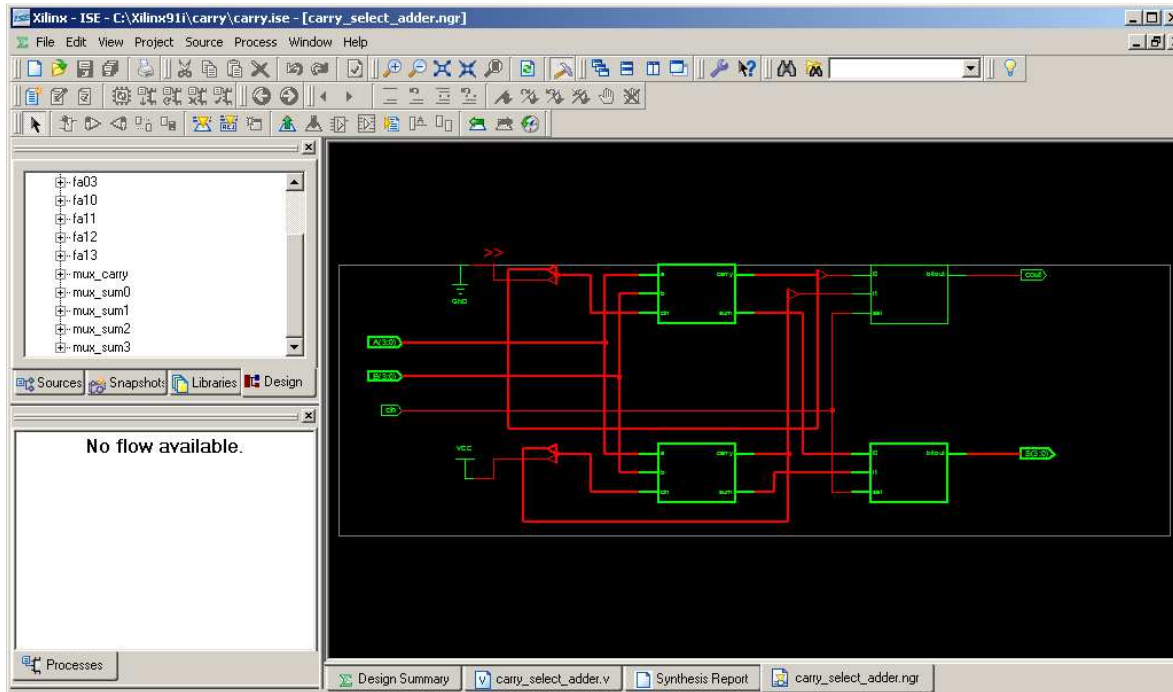
```
always@(i0,i1,sel)
begin
if(sel == 0)
bitout = i0;
else
bitout = i1;
end
```

```
endmodule
```

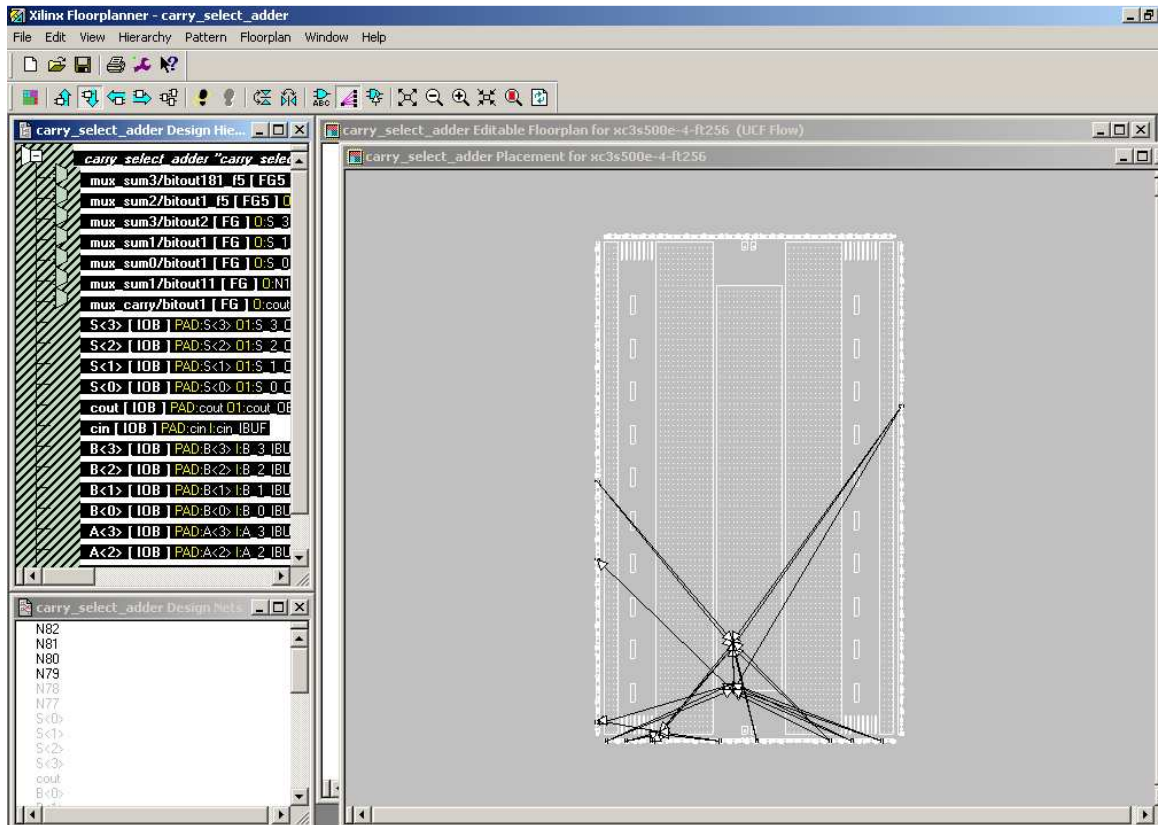
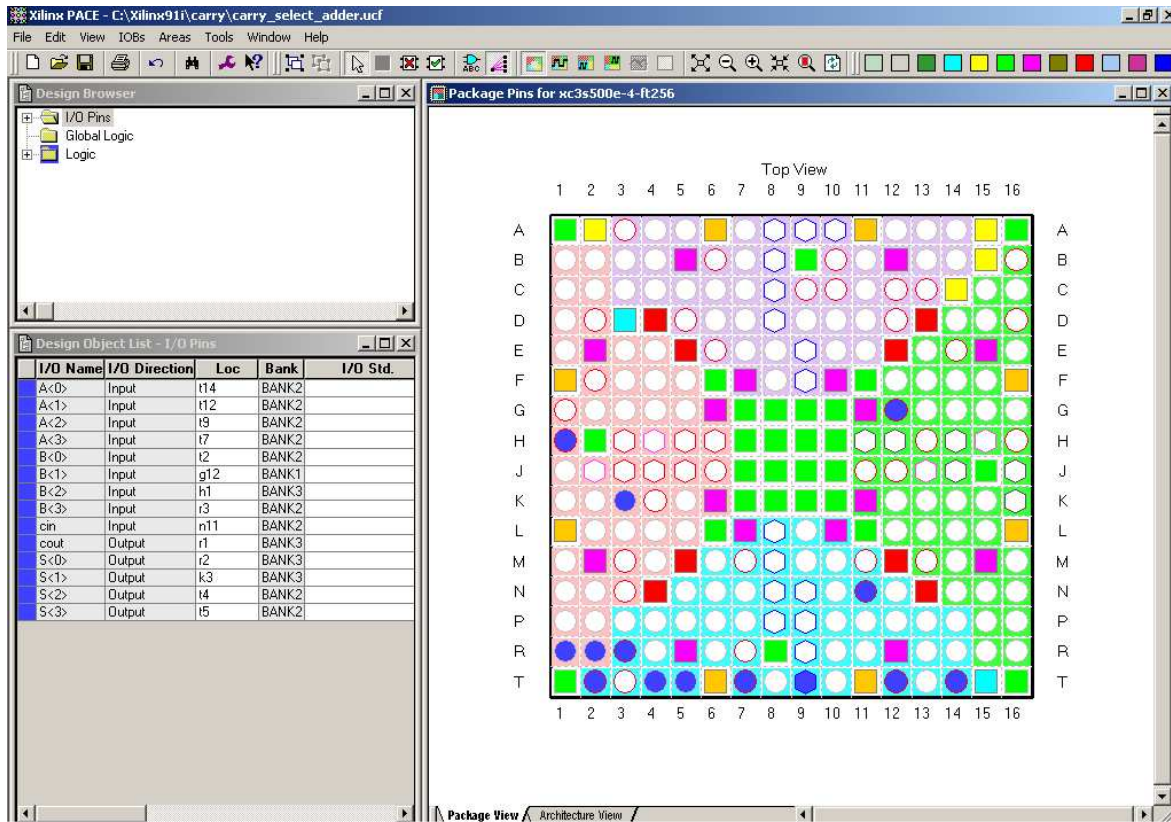

RTL SCHEMATIC:



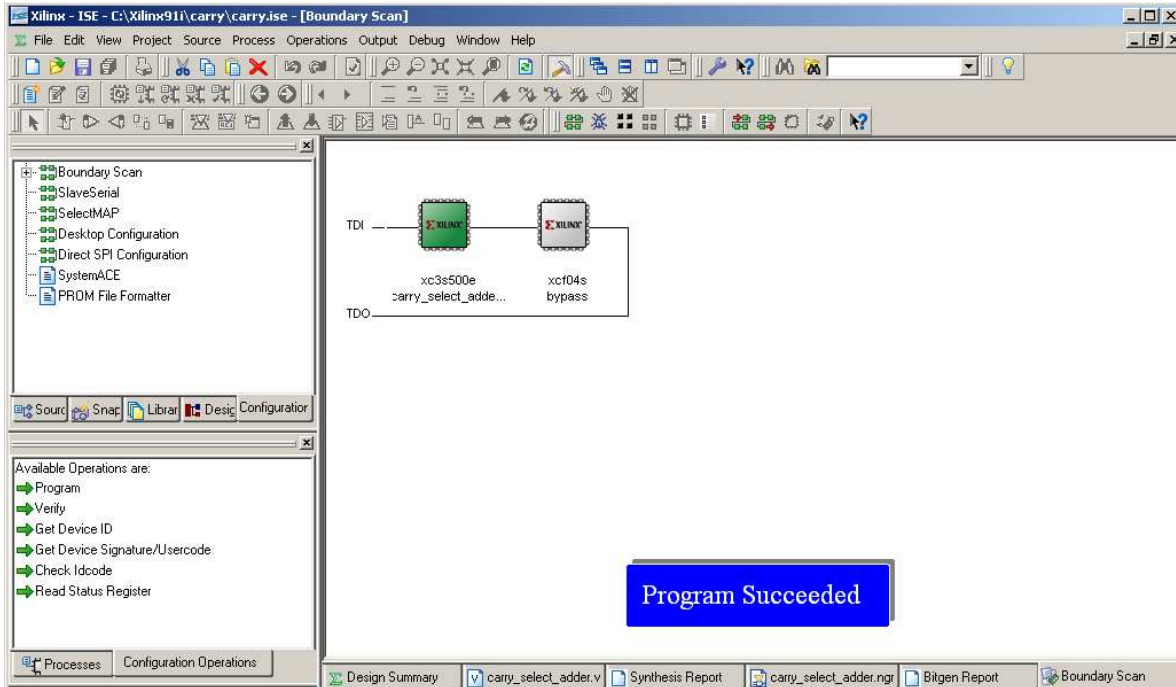
TECHNOLOGY SCHEMATIC:



PLACE AND ROUTE:



HARDWARE FUSING:



RESULT:

Thus, the Hardware fusing and testing of Booth Multiplier and Carry Select Adder were implemented in Spartan 3E FPGA trainer kit using Xilinx project navigator.

EXP NO: 4

Date:

Design and FPGA Implementation of Sequential Circuits

AIM:

To design and implement Counter in FPGA Spartan 3E Trainer kit using Xilinx project navigator.

APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	XilinxProject navigator	1

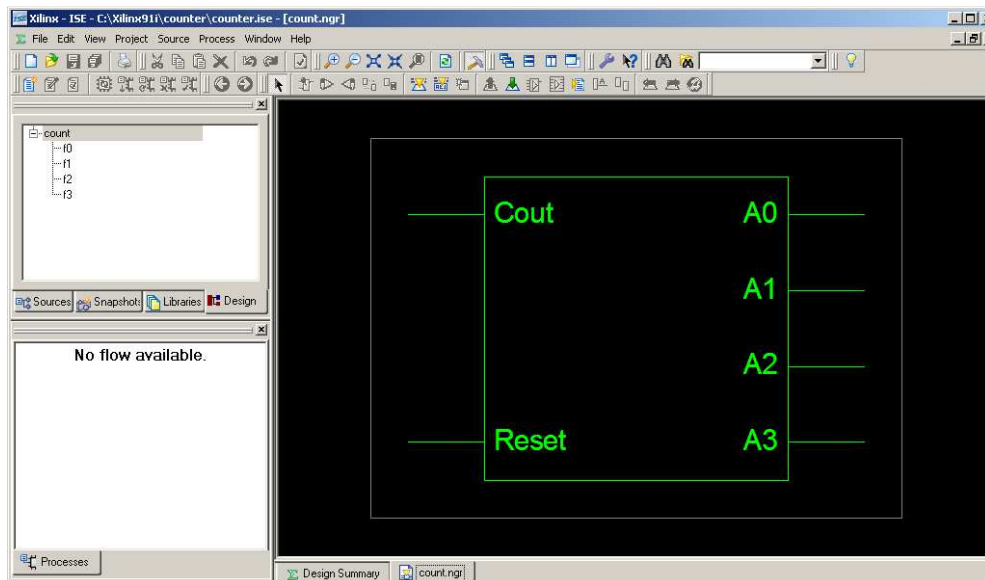
PROCEDURE:

1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the file name → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → Synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Synthesis your design, from the source window select, synthesis/implementation from the window Now double click the Synthesis -XST.
10. After Synthesis, Click on the symbol of FPGA device and Right click and select New Source, Select Implementation Constraints File and type file name and click next.
11. Type the Net list and click save.
12. Implement the design by double clicking Implement design in the process window.
13. Then double click Generate Programming File, Double click Configure Target Device and click OK.
14. Double click Create PROM File in the ISE iMPACT window, Select Storage Target Device as Xilinx Flash PROM and click forward.
15. Add storage Device as xcf01s [2 M] and click forward, Type Output File Name and Location and click OK.
16. Select the corresponding .bit file and click Open, Click No to Add Another Device and Click OK.
17. Double click Generate File.
18. Double click Boundary Scan and Right click on the window and select Initialize Chain, Now Select the corresponding .mcs file and click open.
19. Click OK in the Device Programming Properties window, Download the Program on to the kit by Right clicking on the device icon and select program.
20. Verify the output in the target device.

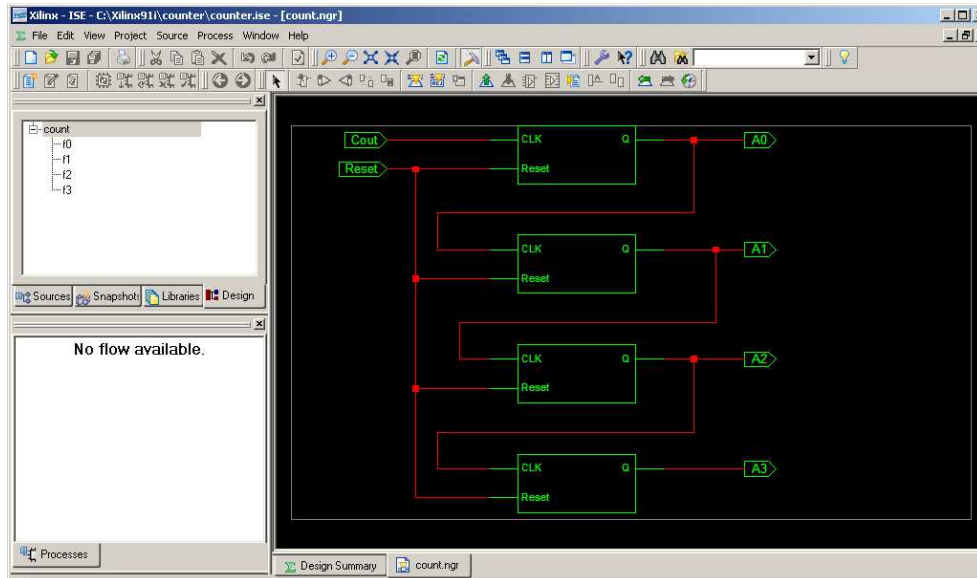
CODING:

```
Module ripple counter (A0, A1, A2, A3, Count, Reset)
Output A0,A1, A2,A3;
Input Count,Reset;
ff f0(A0, Count, Reset);
ff f1(A1, A0, Reset);
ff f2(A2, A1, Reset);
ff f3(A3, A2, Reset);
end module
moduleff
(Q, CLK, Reset);
output Q;
input CLK, Reset;
reg Q;
always @ (negedge CLK or negedge Reset)
if (~Reset)
Q=1'b0;
else Q=(~Q);
endmodule
```

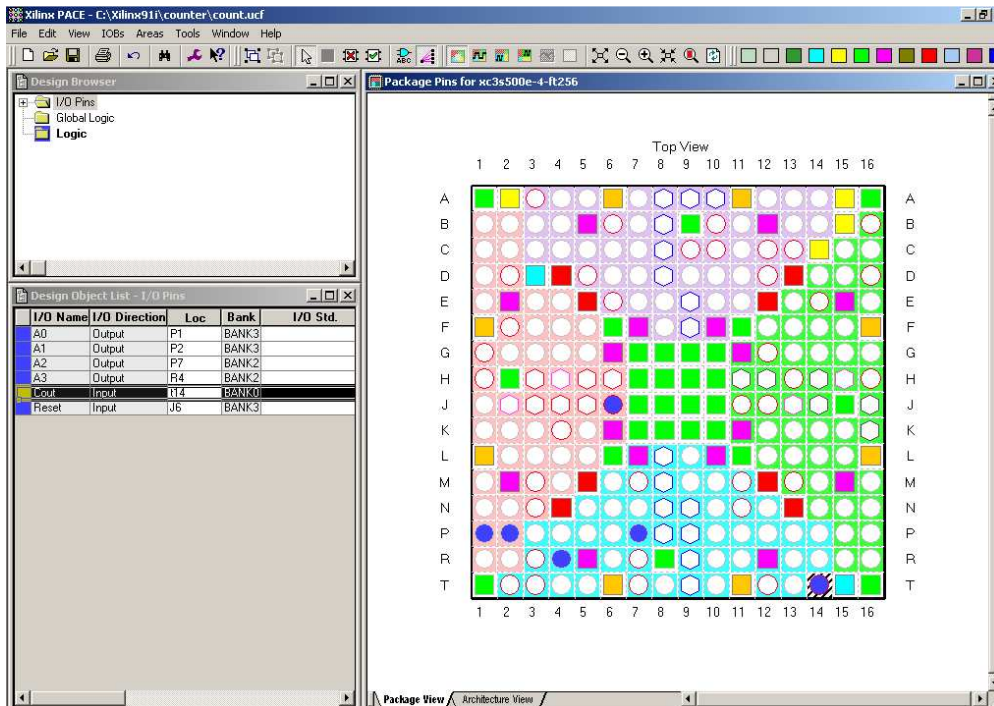
RTL SCHEMATIC:

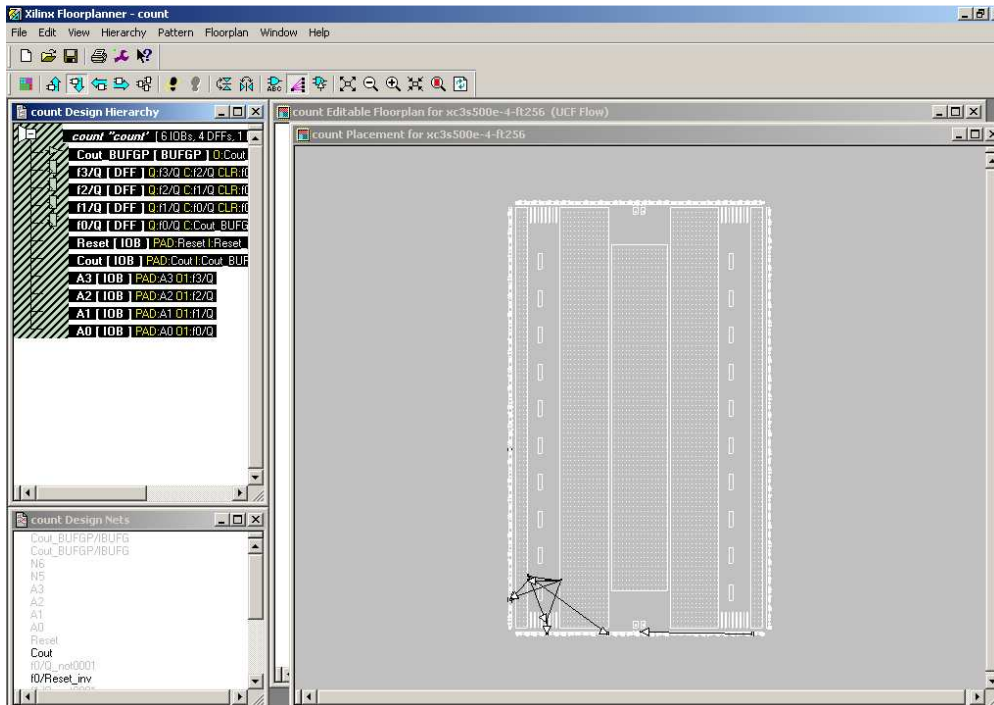


TECHNOLOGY SCHEMATIC:

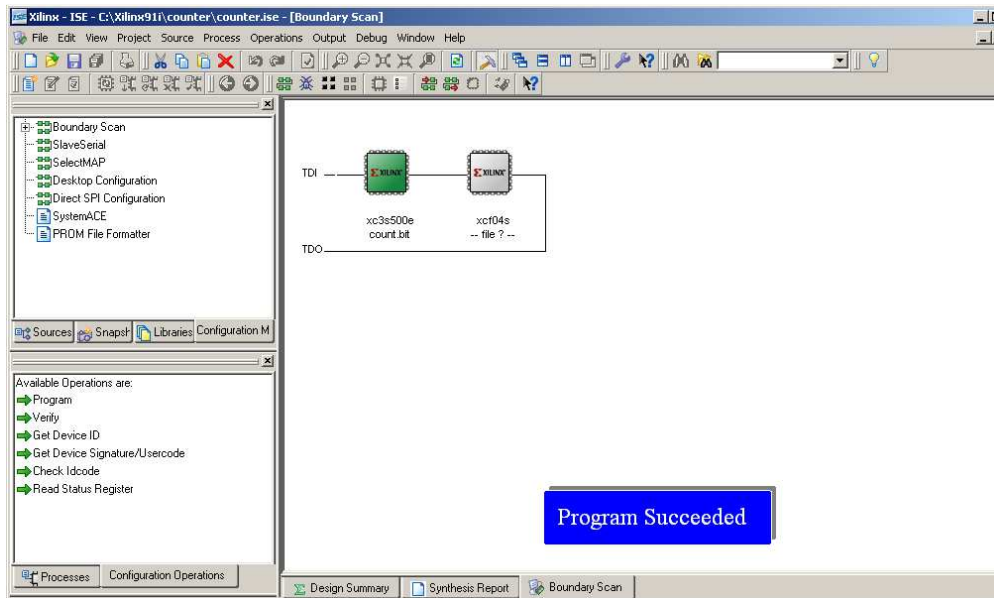


PLACE AND ROUTE:





HARDWARE FUSING



RESULT:

Thus, the Hardware fusing and testing of 4-Bit counter was implemented in Spartan 3E FPGA trainer kit using Xilinx project navigator.

EXP NO: 5
Date:

Analysis of Area, Power and Delay for Sequential Circuits

AIM:

To analyze area, power and delay for Counter and PRBS generator in FPGA Spartan 3E Trainer kit using Xilinx project navigator.

APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	XilinxProject navigator	1

PROCEDURE:

1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the file name → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → Synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Synthesis your design, from the source window select, synthesis/implementation from the window Now double click the Synthesis -XST.
10. After Synthesis, Click on the synthesis report to generate the area and delay summary.
11. Type the Net list and click save.
12. Implement the design by double clicking Implement design in the process window.
13. Then double click Generate Programming File, Double click Configure Target Device and click OK.
14. Double click Create PROM File in the ISE iMPACT window, Select Storage Target Device as Xilinx Flash PROM and click forward.
15. Add storage Device as xcf01s [2 M] and click forward, Type Output File Name and Location and click OK.
16. Select the corresponding .bit file and click Open, Click No to Add another Device and Click OK.
17. Double click Generate File.
18. Double click Boundary Scan and Right click on the window and select Initialize Chain, Now Select the corresponding .mcs file and click open.
19. Click OK in the Device Programming Properties window, Download the Program on to the kit by Right clicking on the device icon and select program.
20. Verify the output in the target device.

RIPPLE COUNTER:

CODING:

```
Module ripple counter (A0, A1, A2, A3, Count, Reset)
Output A0,A1, A2,A3;
Input Count,Reset;
ff f0(A0, Count, Reset);
ff f1(A1, A0, Reset);
ff f2(A2, A1, Reset);
ff f3(A3, A2, Reset);
end module
moduleff
(Q, CLK, Reset);
output Q;
input CLK, Reset;
reg Q;
always @ (negedge CLK or negedge Reset)
if (~Reset)
Q=1'b0;
else Q=~Q;
endmodule
```

ANALYZE REPORT:

Area analysis:

Device utilization summary:

Selected Device: 3s500eft256-4

Number of Slices:	3 out of	4656	0%
Number of Slice Flip Flops:	4 out of	9312	0%
Number of 4 input LUTs:	5 out of	9312	0%
Number of IOs:	6		
Number of bonded IOBs:	6 out of	190	3%
Number of GCLKs:	1 out of	24	4%

Partition Resource Summary:

No Partitions were found in this design.

=====

Power analysis:

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		81
Vccint 1.20V:	26	31
Vccaux 2.50V:	18	45
Vcco25 2.50V:	2	5
Clocks:	0	0
Inputs:	0	0
Logic:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	26	31
Quiescent Vccaux 2.50V:	18	45
Quiescent Vcco25 2.50V:	2	5

Thermal summary:	
Estimated junction temperature:	28C
Ambient temp:	25C
Case temp:	27C
Theta J-A:	31C/W

Delay Analysis:

Timing Summary:

Speed Grade: -4

Minimum period: 2.554ns (Maximum Frequency: 391.543MHz)

Minimum input arrival time before clock: No path found

Maximum output required time after clock: 4.394ns

Maximum combinational path delay: No path found

Timing Detail:

All values displayed in nanoseconds (ns)

=====

Timing constraint: Default period analysis for Clock 'f2/Q'
Clock period: 2.470ns (frequency: 404.858MHz)
Total number of paths / destination ports: 1 / 1

Delay: 2.470ns (Levels of Logic = 1)
Source: f3/Q (FF)
Destination: f3/Q (FF)
Source Clock: f2/Q falling
Destination Clock: f2/Q falling

Data Path: f3/Q to f3/Q

Gate Net

Cell:in->out fanout Delay Delay Logical Name (Net Name)

FDC_1:C->Q	2	0.591	0.447	f3/Q (f3/Q)
INV:I->O	1	0.704	0.420	f3/Q_not00011_INV_0 (f3/Q_not0001)
FDC_1:D		0.308		f3/Q

Total 2.470ns (1.603ns logic, 0.867ns route)
(64.9% logic, 35.1% route)

Timing constraint: Default period analysis for Clock 'f1/Q'
Clock period: 2.554ns (frequency: 391.543MHz)
Total number of paths / destination ports: 1 / 1

Delay: 2.554ns (Levels of Logic = 1)
Source: f2/Q (FF)
Destination: f2/Q (FF)
Source Clock: f1/Q falling
Destination Clock: f1/Q falling

Data Path: f2/Q to f2/Q

Gate Net

Cell:in->out fanout Delay Delay Logical Name (Net Name)

FDC_1:C->Q	3	0.591	0.531	f2/Q (f2/Q)
INV:I->O	1	0.704	0.420	f2/Q_not00011_INV_0 (f2/Q_not0001)
FDC_1:D		0.308		f2/Q

Total 2.554ns (1.603ns logic, 0.951ns route)
(62.8% logic, 37.2% route)

Timing constraint: Default period analysis for Clock 'f0/Q'
Clock period: 2.554ns (frequency: 391.543MHz)
Total number of paths / destination ports: 1 / 1

Delay: 2.554ns (Levels of Logic = 1)
Source: f1/Q (FF)
Destination: f1/Q (FF)
Source Clock: f0/Q falling
Destination Clock: f0/Q falling

Data Path: f1/Q to f1/Q

Gate Net

Cell:in->out fanout Delay Delay Logical Name (Net Name)

FDC_1:C->Q	3	0.591	0.531	f1/Q (f1/Q)
INV:I->O	1	0.704	0.420	f1/Q_not00011_INV_0 (f1/Q_not0001)
FDC_1:D		0.308		f1/Q

Total 2.554ns (1.603ns logic, 0.951ns route)
(62.8% logic, 37.2% route)

Timing constraint: Default period analysis for Clock 'Cout'
Clock period: 2.554ns (frequency: 391.543MHz)
Total number of paths / destination ports: 1 / 1

Delay: 2.554ns (Levels of Logic = 1)
Source: f0/Q (FF)
Destination: f0/Q (FF)
Source Clock: Cout falling
Destination Clock: Cout falling

Data Path: f0/Q to f0/Q

	Gate	Net			
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)	

FDC_1:C->Q	3	0.591	0.531	f0/Q (f0/Q)	
INV:I->O	1	0.704	0.420	f0/Q_not00011_INV_0 (f0/Q_not0001)	
FDC_1:D		0.308		f0/Q	

Total		2.554ns (1.603ns logic, 0.951ns route)			
		(62.8% logic, 37.2% route)			

=====
Timing constraint: Default OFFSET OUT AFTER for Clock 'Cout'
Total number of paths / destination ports: 1 / 1

Offset: 4.394ns (Levels of Logic = 1)
Source: f0/Q (FF)
Destination: A0 (PAD)
Source Clock: Cout falling

Data Path: f0/Q to A0

	Gate	Net			
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)	

FDC_1:C->Q	3	0.591	0.531	f0/Q (f0/Q)	
OBUF:I->O		3.272		A0_OBUF (A0)	

Total		4.394ns (3.863ns logic, 0.531ns route)			
		(87.9% logic, 12.1% route)			

=====
Timing constraint: Default OFFSET OUT AFTER for Clock 'f0/Q'
Total number of paths / destination ports: 1 / 1

Offset: 4.394ns (Levels of Logic = 1)
Source: f1/Q (FF)
Destination: A1 (PAD)
Source Clock: f0/Q falling

Data Path: f1/Q to A1

	Gate	Net			
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)	

FDC_1:C->Q	3	0.591	0.531	f1/Q (f1/Q)	
OBUF:I->O		3.272		A1_OBUF (A1)	

Total		4.394ns (3.863ns logic, 0.531ns route)			
		(87.9% logic, 12.1% route)			

=====

Timing constraint: Default OFFSET OUT AFTER for Clock 'f1/Q'
Total number of paths / destination ports: 1 / 1

Offset: 4.394ns (Levels of Logic = 1)
Source: f2/Q (FF)
Destination: A2 (PAD)
Source Clock: f1/Q falling

Data Path: f2/Q to A2

Cell:in->out Gate Net
fanout Delay Delay Logical Name (Net Name)

FDC_1:C->Q 3 0.591 0.531 f2/Q (f2/Q)
OBUF:I->O 3.272 A2_OBUF (A2)

Total 4.394ns (3.863ns logic, 0.531ns route)
 (87.9% logic, 12.1% route)

=====
Timing constraint: Default OFFSET OUT AFTER for Clock 'f2/Q'
Total number of paths / destination ports: 1 / 1

Offset: 4.310ns (Levels of Logic = 1)
Source: f3/Q (FF)
Destination: A3 (PAD)
Source Clock: f2/Q falling

Data Path: f3/Q to A3

Cell:in->out Gate Net
fanout Delay Delay Logical Name (Net Name)

FDC_1:C->Q 2 0.591 0.447 f3/Q (f3/Q)
OBUF:I->O 3.272 A3_OBUF (A3)

Total 4.310ns (3.863ns logic, 0.447ns route)
 (89.6% logic, 10.4% route)

=====
CPU : 3.91 / 4.03 s | Elapsed : 4.00 / 4.00 s

PRBS GENERATOR:

CODING:

```
module prbs1 (rand, clk, reset);
input clk, reset;
output rand;
wire rand;
reg [3:0] temp;
always @ (posedge reset) begin
temp<= 4'hf;
end
always @ (posedgeclk) begin
if (~reset) begin
temp <= {temp[0]^temp[1],temp[3],temp[2],temp[1]};
end
end
assign rand = temp[0];
endmodule
```

RTL SCHEMATIC:



TECHNOLOGY SCHEMATIC:



REPORT:

Area Analysis:

Device utilization summary:

Selected Device: 3s500eft256-4

Number of Slices: 0 out of 4656 0%

Number of IOs: 3

Number of bonded IOBs: 1 out of 190 0%

Partition Resource Summary:

No Partitions were found in this design.

=====

Power Analysis:

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		81
Vccint 1.20V:	26	31
Vccaux 2.50V:	18	45
Vcco25 2.50V:	2	5
Inputs:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	26	31
Quiescent Vccaux 2.50V:	18	45
Quiescent Vcco25 2.50V:	2	5

Thermal summary:	
Estimated junction temperature:	28C
Ambient temp:	25C
Case temp:	27C
Theta J-A:	31C/W

Delay Analysis:

TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.

FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT GENERATED AFTER PLACE-and-ROUTE.

Clock Information:

No clock signals found in this design

Asynchronous Control Signals Information:

No asynchronous control signals found in this design

Timing Summary:

Speed Grade: -4

Minimum period: No path found

Minimum input arrival time before clock: No path found

Maximum output required time after clock: No path found

Maximum combinational path delay: No path found

Timing Detail:

All values displayed in nanoseconds (ns)

CPU: 3.84 / 4.19 s | Elapsed: 4.00 / 4.00 s

RESULT:

Thus, area, power and delay for Counter and PRBS generator was analyzed in FPGA Spartan 3E Trainer kit using Xilinx project navigator.

EXP NO: 6
Date:

Invoke PLL to generate Real Time Clock

AIM:

To invoke the FPGA Spartan 3E PLL to generate Real time Clock kit using Xilinx project navigator.

APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	XilinxProject navigator	1

PROCEDURE:

1. Start the Xilinx ISE by using Start → Program files → Xilinx ISE → project navigator
2. Click File → New Project
3. Enter the Project Name and select the location then click next
4. Select the Device and other category and click next twice and finish.
5. Click on the symbol of FPGA device and then right click → click on new source.
6. Select the Verilog Module and give the file name → click next and define ports → click next and finish.
7. Writing the Verilog Code in Verilog Editor.
8. Run the Check syntax → Process window → Synthesize → double click check syntax. If any errors found then remove the errors with proper syntax & coding.
9. Synthesis your design, from the source window select, synthesis/implementation from the window Now double click the Synthesis -XST.
10. After Synthesis, Click on the symbol of FPGA device and Right click and select New Source, Select Implementation Constraints File and type file name and click next.
11. Type the Net list and click save.
12. Implement the design by double clicking Implement design in the process window.
13. Then double click Generate Programming File, Double click Configure Target Device and click OK.
14. Double click Create PROM File in the ISE iMPACT window, Select Storage Target Device as Xilinx Flash PROM and click forward.
15. Add storage Device as xcf01s [2 M] and click forward, Type Output File Name and Location and click OK.
16. Select the corresponding .bit file and click Open, Click No to Add Another Device and Click OK.
17. Double click Generate File.
18. Double click Boundary Scan and Right click on the window and select Initialize Chain, Now Select the corresponding .mcs file and click open.
19. Click OK in the Device Programming Properties window, Download the Program on to the kit by Right clicking on the device icon and select program.
20. Verify the output in the target device.

FPGA Connections to Seven-Segment Display

SEGMENT	FPGA PIN
A	P8
B	P10
C	P9
D	P6
E	P4
F	P5
G	P3
DP	P11

CLOCK SOURCE

Spartan3E FPGA works in different Clock frequencies

Clock Input	FPGA PIN
CLK	A8
RST	J6

Digit Enable Signals

Display	DISP 1	DISP 2	DISP 3	DISP 4	DISP 5	DISP 6
FPGA PIN	P1	P2	P7	R4	R11	N14

PLL OSCILLATOR SETTINGS

For PLL, ICS525-01 or ICS525-02 is used. Select the clock settings as per the PLL in the On board

0 = Shorted

1 = Open

For any other CLK frequency in between 1MHz to 100MHz use the following formula.

$$\text{CLK frequency} = \text{Input frequency} \cdot 2 \frac{(\text{VDW} + 8)}{(\text{RDW} + 2)(\text{OD})}$$

Where,

Reference Divider Word (RDW) = 1 to 127 (0 is not permitted)

VCO Divider Word (VDW) = 4 to 511 (0, 1, 2, 3 are not permitted)

Output Divider (OD) = values below

CODING:

System Clock = 20MHz

Verilog Program

// System clock Frequency 20MHz

// MODE Functions

// 00 HOURS

// 01 Minutes

// 10 Seconds

// 11 Timer ON

```
module timer(clk, rst, mode, set, sl, atoh);
inputclk; // System Clock
inputrst; // Reset(micro switch)
input [1:0] mode; // Mode Selection(switch 1 & switch 2)
input [7:0] set; // Set Value(switch 4 to switch 11)
output [5:0] sl; // Segment Selection
output [7:0] atoh; // Segment Display Control Data
reg [5:0] sl;
reg [7:0] atoh;
reg [26:0] sig2;
reg [19:1] sig3;
reg [7:0] ssdigit1;
reg [7:0] ssdigit2;
reg [7:0] ssdigit3;
reg [7:0] ssdigit4;
reg [7:0] ssdigit5;
reg [7:0] ssdigit6;
reg [3:0] digit1;
reg [3:0] digit2;
reg [3:0] digit3;
reg [3:0] digit4;
reg [3:0] digit5;
reg [3:0] digit6;
always @ (posedgeclk or negedge rst)
begin
if (rst == 1'b0) begin
sig2 = 0;
sig3 = 0;
digit1 = 0;
digit2 = 0;
digit3 = 0;
digit4 = 0;
digit5 = 0;
digit6 = 0;
end
end
```

```
else begin
if (mode == 2'b00) begin // Hours
if (set[7:4] <= 4'b0001) begin
digit1 = set[7:4];
if (set[3:0] <= 4'b1001)
digit2 = set[3:0];
else
digit2 = 0;
end
else if (set[7:4] == 4'b0010) begin
if (set[3:0] <= 4'b0011) begin
digit1 = set[7:4];
digit2 = set[3:0];
end
else begin
digit1 = 0;
digit2 = 0;
end
end
else begin
digit1 = 0;
digit2 = 0;
end
end
else if (mode == 2'b01) begin // Minutes
if (set[7:4] <= 4'b0101) begin
digit3 = set[7:4];
if (set[3:0] <= 4'b1001)
digit4 = set[3:0];
else
digit4 = 0;
end
else begin
digit3 = 0;
digit4 = 0;
end
end
else if (mode == 2'b10) begin // Seconds
if (set[7:4] <= 4'b0101) begin
digit5 = set[7:4];
if (set[3:0] <= 4'b1001)
digit6 = set[3:0];
else
digit6 = 0;
end
else begin
digit5 = 0;
digit6 = 0;
end
end
end
```

```

else begin
sig2 = sig2 + 1;

case (sig2[24:23]) //RTC Function
2'b00 : begin
digit6 = digit6 + 1;
if (digit6 > 4'b1001) begin
digit6 = 4'b0000;
digit5 = digit5 + 1;
if (digit5 > 4'b0101) begin
digit5 = 4'b0000;
digit4 = digit4 + 1;
if (digit4 > 4'b1001) begin
digit4 = 4'b0000;
digit3 = digit3 + 1;
if (digit3 > 4'b0101) begin
digit3 = 4'b0000;
digit2 = digit2 + 1;
if (digit2 > 4'b1001) begin
digit2 = 4'b0000;
digit1 = digit1 + 1;
end
if ((digit1 >= 4'b0010) & (digit2 >= 4'b0100))
begin
digit1 = 4'b0000;
digit2 = 4'b0000;
end
end
end
end
end

sig2[24:23] = 2'b01;
end

2'b11 : begin
if (sig2[22:19] == 4'b1001)
sig2 = 0;
end
default : begin
end
endcase
end

```

Display Settings

```
sig3 = sig3 + 1;
case (sig3[17:15])
3'b000 : begin
sl = 6'b111110;
case (digit1)
4'b0000 : ssdigit1 = 8'b00111111;
4'b0001 : ssdigit1 = 8'b00000110;
4'b0010 : ssdigit1 = 8'b01011011;
default : ssdigit1 = 8'b00000000;
endcase
atoh = ssdigit1;
end
```

```
3'b001 : begin
sl = 6'b111101;
case (digit2)
4'b0000 : ssdigit2 = 8'b00111111;
4'b0001 : ssdigit2 = 8'b00000110;
4'b0010 : ssdigit2 = 8'b01011011;
4'b0011 : ssdigit2 = 8'b01001111;
4'b0100 : ssdigit2 = 8'b01100110;
4'b0101 : ssdigit2 = 8'b01101101;
4'b0110 : ssdigit2 = 8'b01111101;
4'b0111 : ssdigit2 = 8'b00000111;
4'b1000 : ssdigit2 = 8'b01111111;
4'b1001 : ssdigit2 = 8'b01101111;
default : ssdigit2 = 8'b00000000;
endcase
atoh = ssdigit2;
end
```

```
3'b011 : begin
sl = 6'b111011;
case (digit3)
4'b0000 : ssdigit3 = 8'b00111111;
4'b0001 : ssdigit3 = 8'b00000110;
4'b0010 : ssdigit3 = 8'b01011011;
4'b0011 : ssdigit3 = 8'b01001111;
4'b0100 : ssdigit3 = 8'b01100110;
4'b0101 : ssdigit3 = 8'b01101101;
default : ssdigit3 = 8'b00000000
endcase
atoh = ssdigit3;
end
```

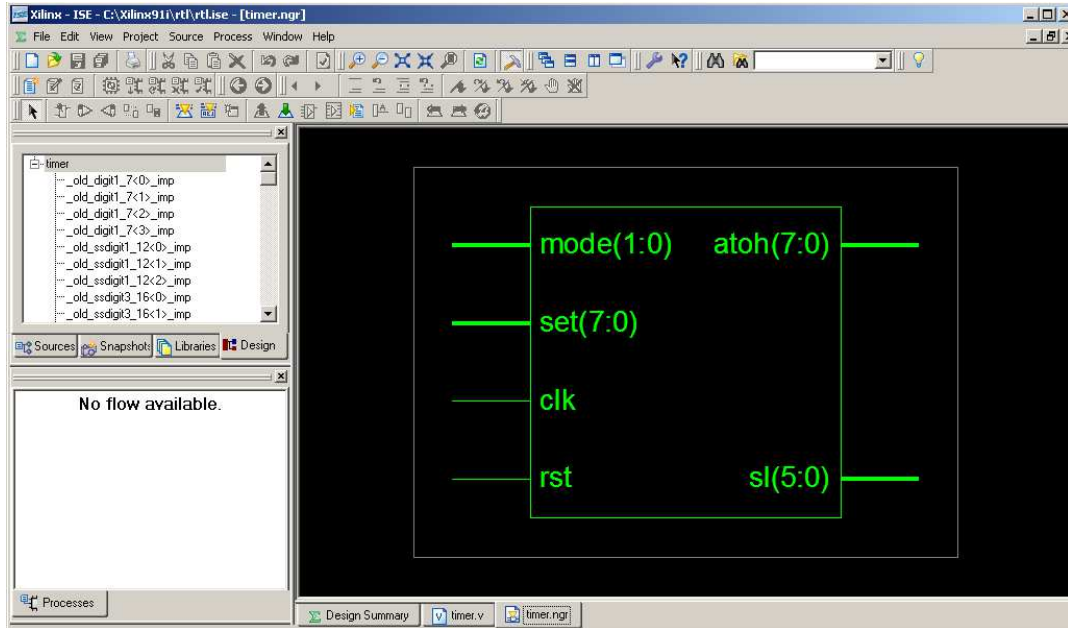
```
3'b100 : begin
sl = 6'b110111;
case (digit4)
4'b0000 : ssdigit4 = 8'b00111111;
4'b0001 : ssdigit4 = 8'b00000110;
4'b0010 : ssdigit4 = 8'b01011011;
```

```
4'b0011 : ssdigit4 = 8'b01001111;
4'b0100 : ssdigit4 = 8'b01100110;
4'b0101 : ssdigit4 = 8'b01101101;
4'b0110 : ssdigit4 = 8'b01111101;
4'b0111 : ssdigit4 = 8'b00000111;
4'b1000 : ssdigit4 = 8'b01111111;
4'b1001 : ssdigit4 = 8'b01101111;
default : ssdigit4 = 8'b00000000;
endcase
atoh = ssdigit4;
end
```

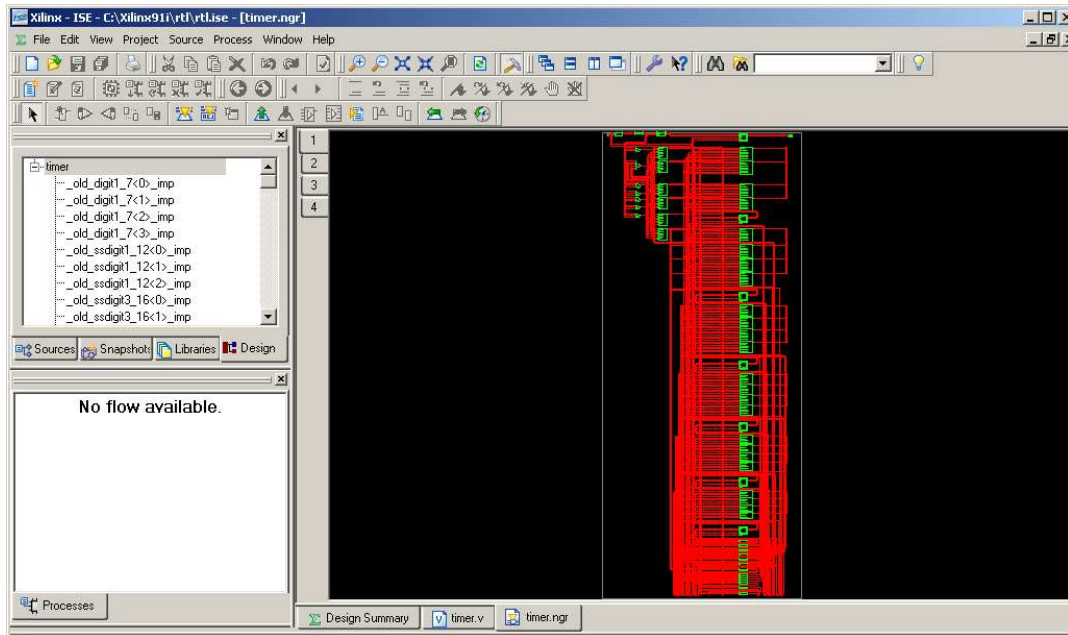
```
3'b110 : begin
sl = 6'b101111;
case (digit5)
4'b0000 : ssdigit5 = 8'b00111111;
4'b0001 : ssdigit5 = 8'b00000110;
4'b0010 : ssdigit5 = 8'b01011011;
4'b0011 : ssdigit5 = 8'b01001111;
4'b0100 : ssdigit5 = 8'b01100110;
4'b0101 : ssdigit5 = 8'b01101101;
default : ssdigit5 = 8'b00000000;
endcase
atoh = ssdigit5;
end
```

```
3'b111 : begin
sl = 6'b011111;
case (digit6)
4'b0000 : ssdigit6 = 8'b00111111;
4'b0001 : ssdigit6 = 8'b00000110;
4'b0010 : ssdigit6 = 8'b01011011;
4'b0011 : ssdigit6 = 8'b01001111;
4'b0100 : ssdigit6 = 8'b01100110;
4'b0101 : ssdigit6 = 8'b01101101;
4'b0110 : ssdigit6 = 8'b01111101;
4'b0111 : ssdigit6 = 8'b00000111;
4'b1000 : ssdigit6 = 8'b01111111;
4'b1001 : ssdigit6 = 8'b01101111;
default : ssdigit6 = 8'b00000000;
endcase
atoh = ssdigit6;
end
endcase
end
end
endmodule
```

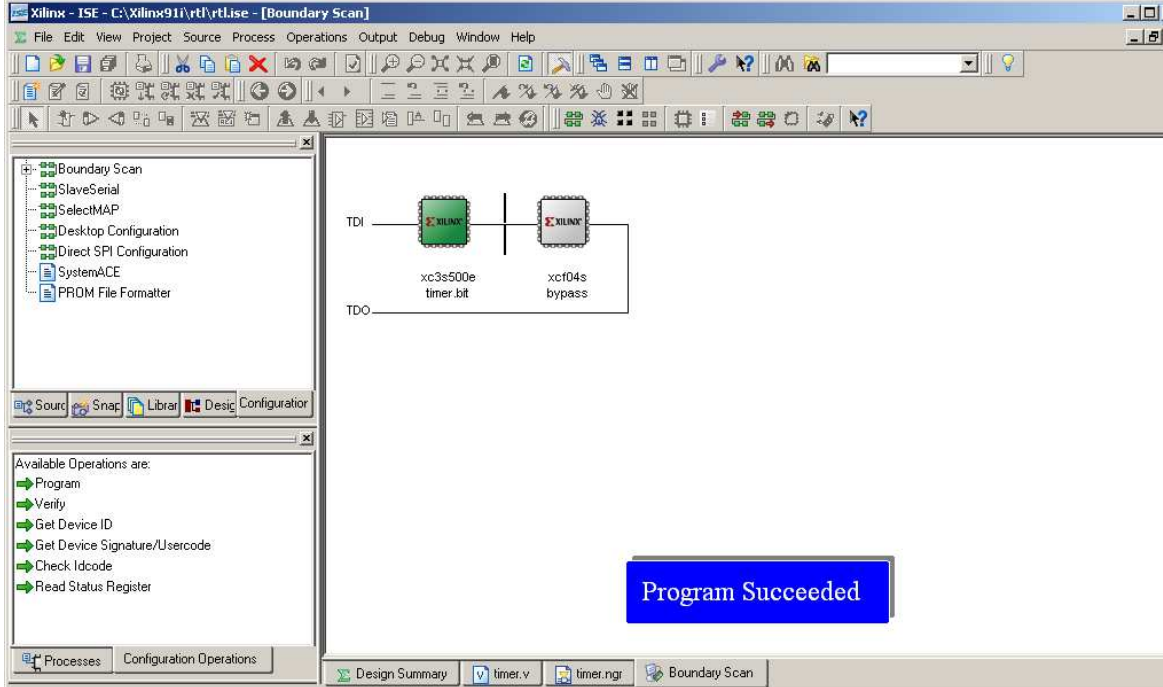
RTL SCHEMATIC



TECHNOLOGY SCHEMATIC



HARDWARE FUSING



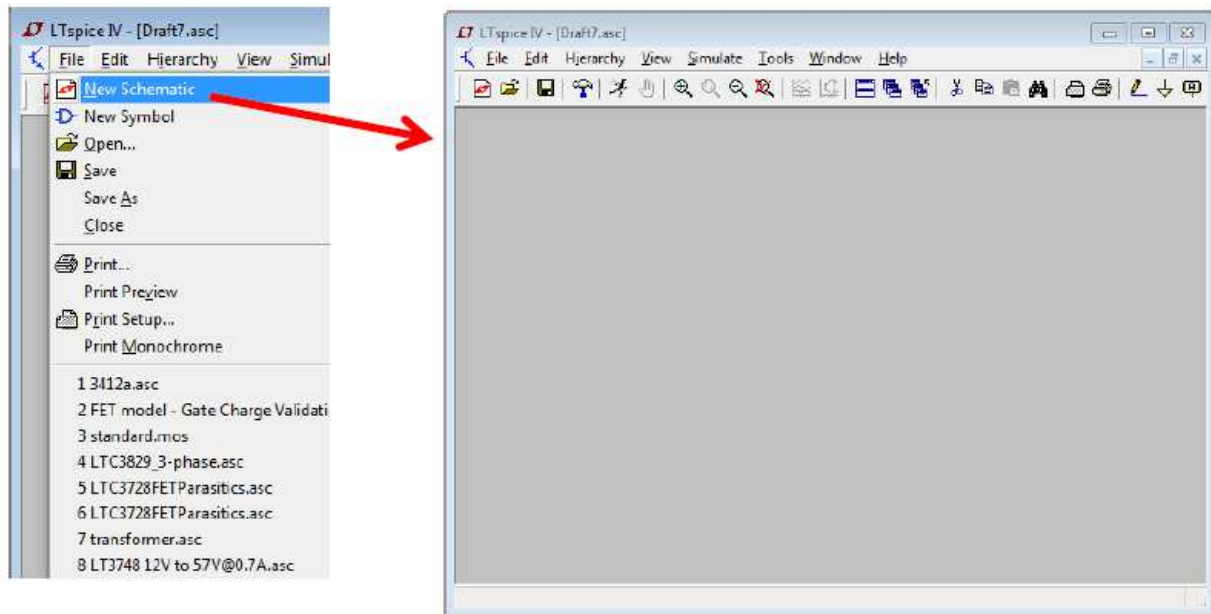
RESULT:

Thus, the FPGA Spartan 3E PLL was invoked to generate Real time Clock in kit using Xilinx project navigator.

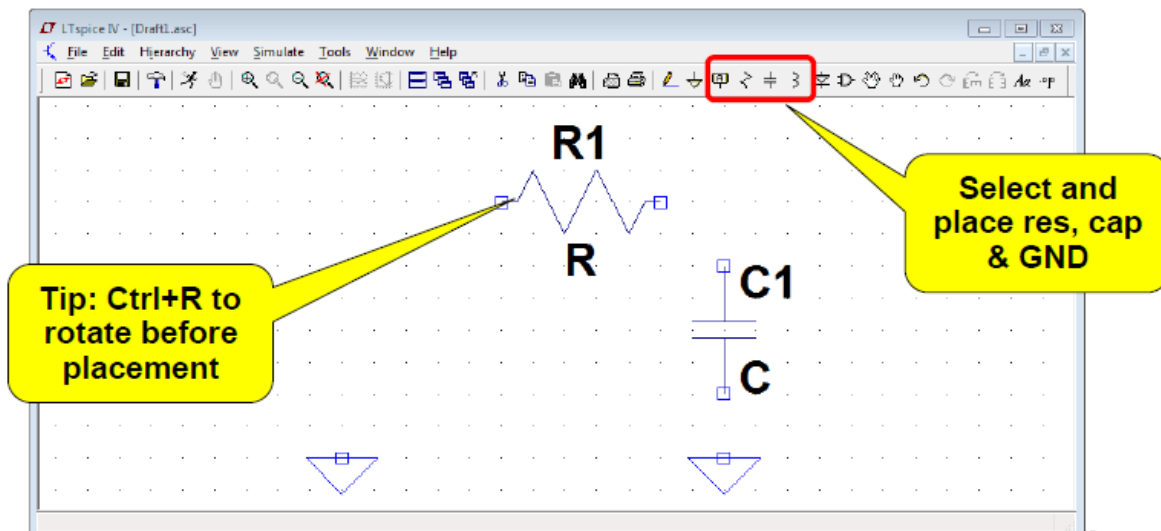
LT SPICE

Procedure for simulation and implementation of EDA tool-LtSPICE

- ❖ Step 1: Open up a blank schematic screen
 - ❖ Select "File" Menu and "New Schematic"

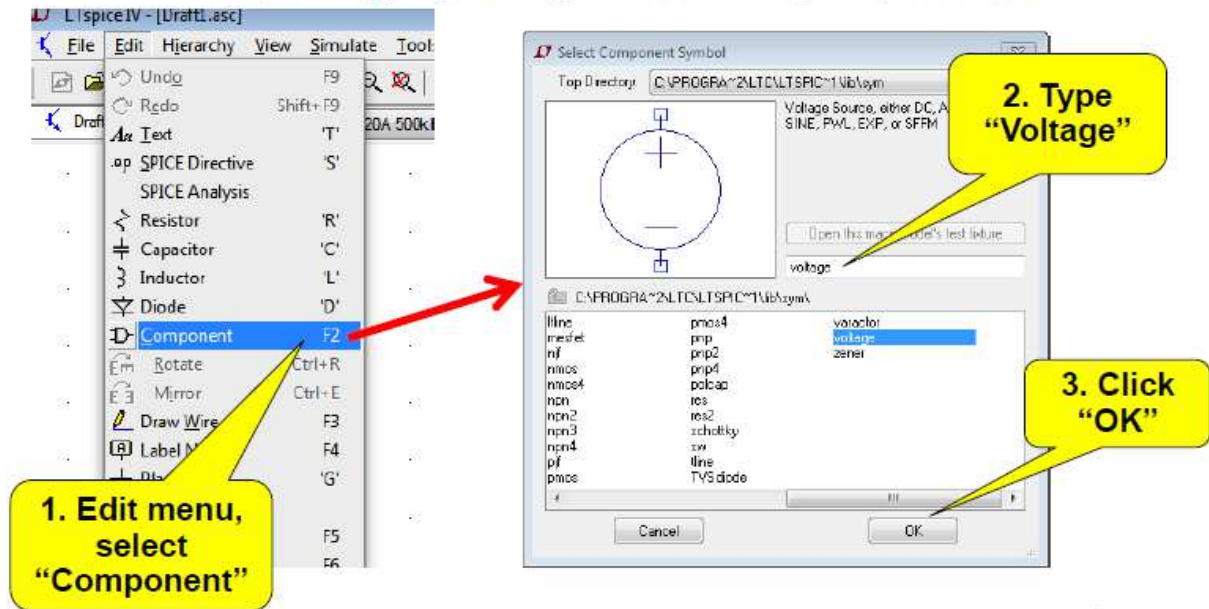


- ❖ Step 2: Add the passives and grounds
 - ❖ Using the toolbar, select Resistor, Capacitor and Ground. Place these symbols on the schematic as shown below. Use Ctrl+R to rotate before placement.

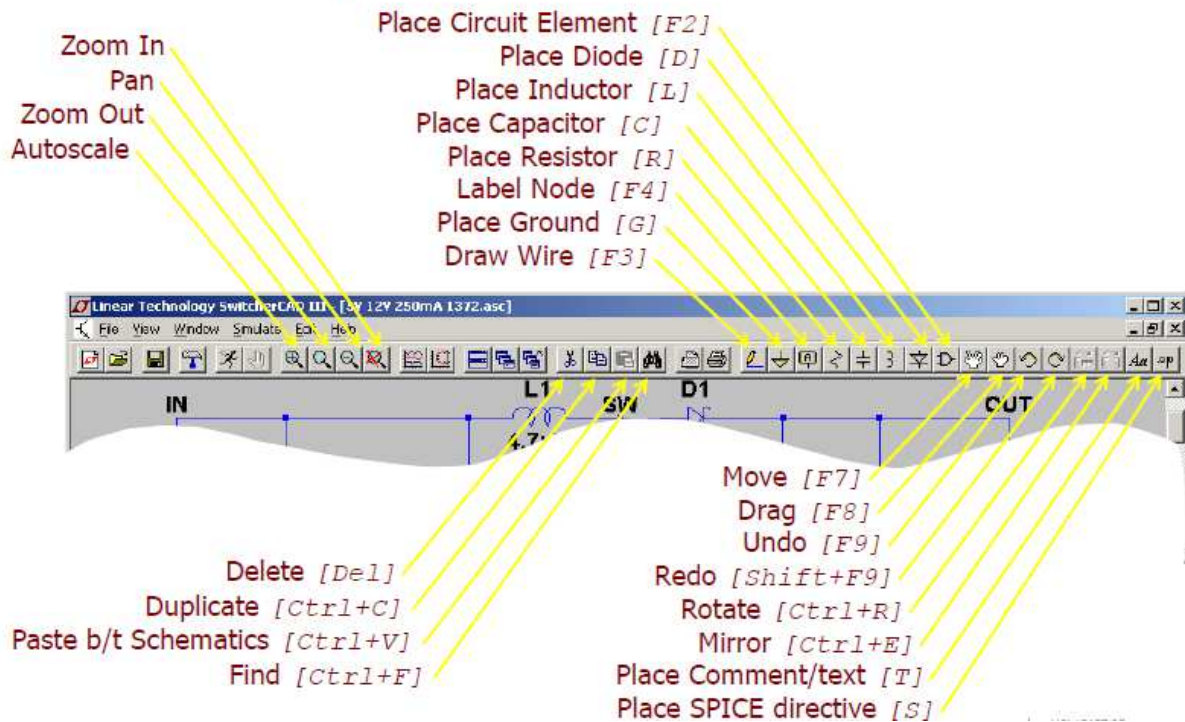


❖ Step 3: Add the voltage source

- ❖ Select "Edit" Menu and "Component". From the component window, start typing "voltage" in the dialog box, and click "OK"

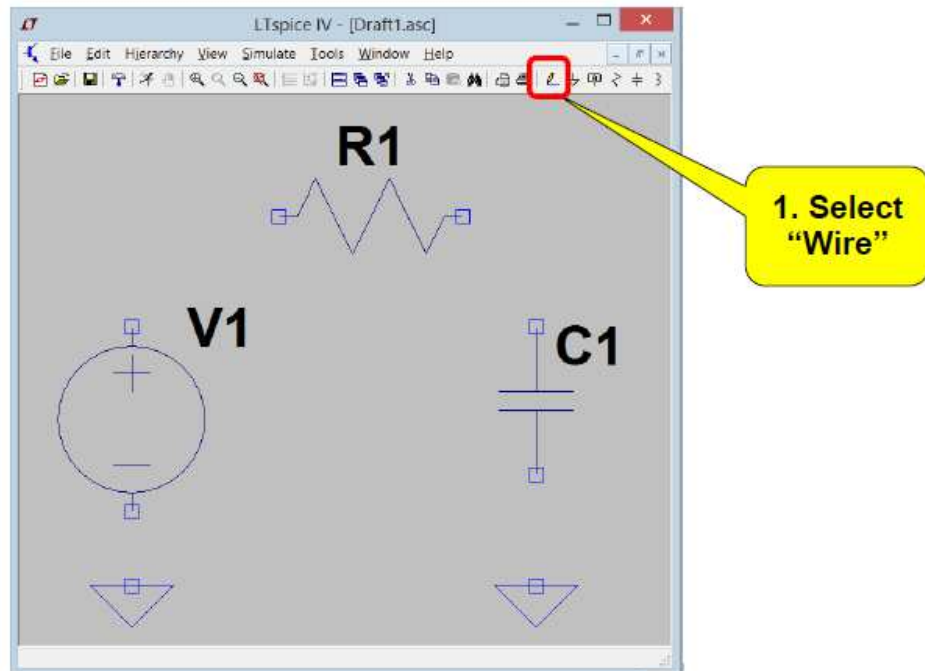


Toolbar and Keyboard Shortcuts

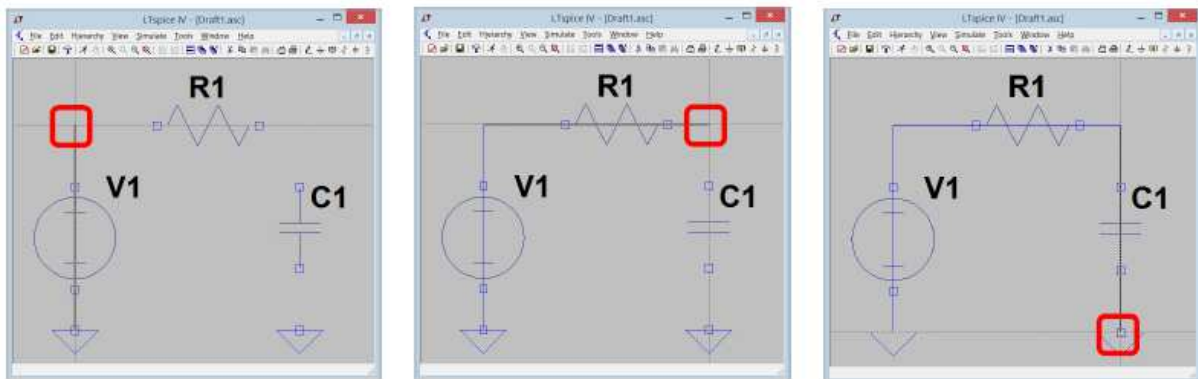


❖ Step 4: Wire up the circuit

- ❖ Using the toolbar, select Wire



❖ Step 4: Wire up the circuit (cont.)



Left-Click ground

"Pull" wire up through the source

Left-Click here  to anchor

"Pull" wire through the resistor

Left-Click here  to anchor

"Pull" wire down through the capacitor

Left-Click here  to anchor & finish

Hint: Press the ESC key at any time to clean up the schematic

❖ Step 5: Add net labels

- ❖ Using the toolbar, select Label Net. Label the input/output nodes as shown below

The screenshot shows the LTspice IV interface with a circuit diagram. A yellow callout box labeled "1. Select 'Label Net'" points to the "Label Net" icon in the toolbar. A second yellow callout box labeled "2. Enter net name" points to the "Net Name" dialog box, which has "IN" entered in the text field. A third yellow callout box labeled "3. Place on wire" points to a wire being placed on the input node of the circuit. The circuit diagram includes a voltage source V1, a resistor R1, and a capacitor C1. The input node is labeled "IN" and the output node is labeled "OUT".

❖ Step 6a: Component values

- ❖ Right-Click on each component symbol to change its value as shown below

The screenshot shows the LTspice IV interface with the circuit diagram. A yellow callout box labeled "Right-click on symbol" points to the resistor symbol R1. Another yellow callout box labeled "Or Right-click on value" points to the "10k" value of the resistor. A red arrow points from the "Right-click on symbol" callout to the "Resistor - R1" dialog box, which shows the "Resistance[Ω]" field set to "10k". Another red arrow points from the "Or Right-click on value" callout to the "Enter new Value for R1" dialog box, which shows the "10k" value entered in the text field.

❖ Step 6b: Source parameters

- ❖ Right-Click on the voltage source and enter the parameters shown below under the "Advanced" tab.

The diagram shows a circuit schematic with a voltage source V1, a resistor R1 (10k), and a capacitor C1 (0.1µF). The source is labeled with parameters: PULSE(0 5 0 1u 10m 20m 3). A red arrow points from the source to the "Voltage Source - V1" dialog box. A yellow callout box says "Click 'Advanced'". Another yellow callout box says "Right-click source".

The "Voltage Source - V1" dialog box shows the "Advanced" tab selected. The "Independent Voltage Source - V1" dialog box shows the "Advanced" tab selected. The "Advanced" tab in the "Independent Voltage Source - V1" dialog box shows the following parameters:

- Functions: PULSE(V1 V2 Tdelay Trise Tfall Ton Period Ncycles)
- DC Value: DC value: []
- Small signal AC analysis(AC): AC Amplitude: [], AC Phase: []
- Parasitic Properties: Series Resistance[Ω]: [], Parallel Capacitance[F]: []

❖ Step 6b: Source parameters

- ❖ Right-Click on the voltage source and enter the parameters shown below under the "Advanced" tab.

The "Independent Voltage Source - V1" dialog box shows the "Advanced" tab selected. The "Functions" section shows the following parameters:

- PULSE(V1 V2 Tdelay Trise Tfall Ton Period Ncycles)
- SINE(Voff Vamp Freq Td Theta Phi Ncycles)
- EXP(V1 V2 Td1 Tau1 Td2 Tau2)
- SFFM(Voff Vamp Fcar MDI Feig)
- PWL(t1 v1 t2 v2...)
- PWL FILE: [] Browse

The "DC Value" section shows the following parameters:

- DC value: []
- Make this information visible on schematic:

The "Small signal AC analysis(AC)" section shows the following parameters:

- AC Amplitude: []
- AC Phase: []
- Make this information visible on schematic:

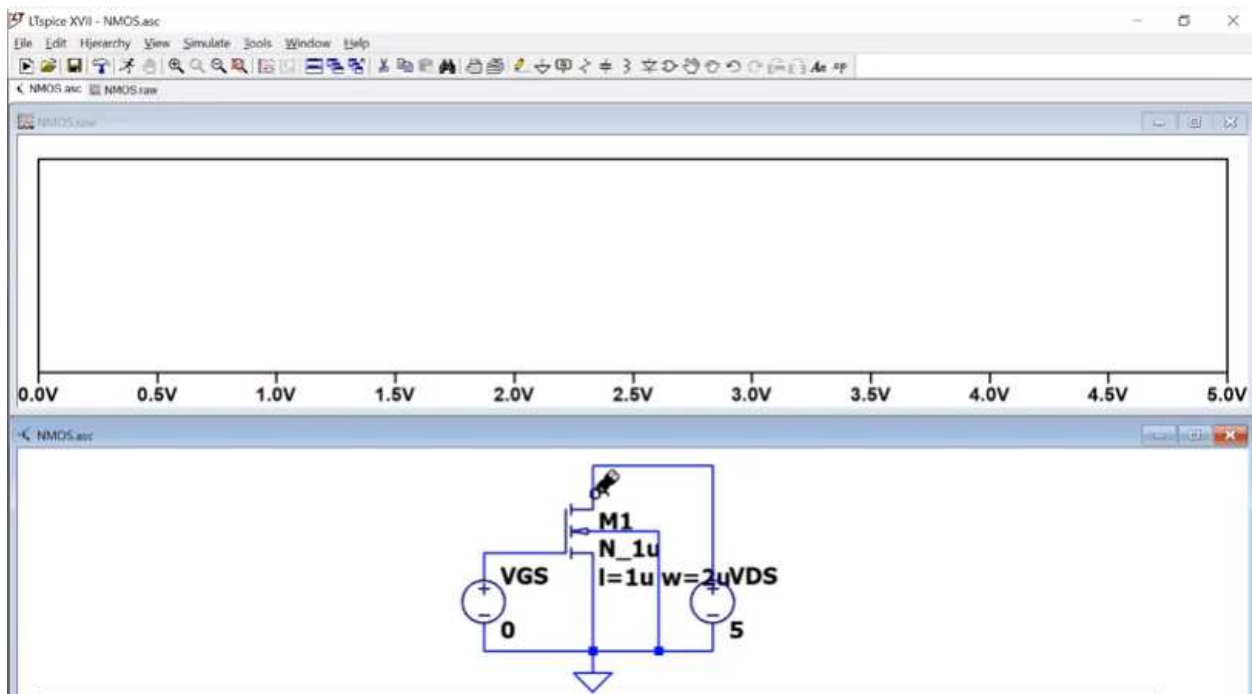
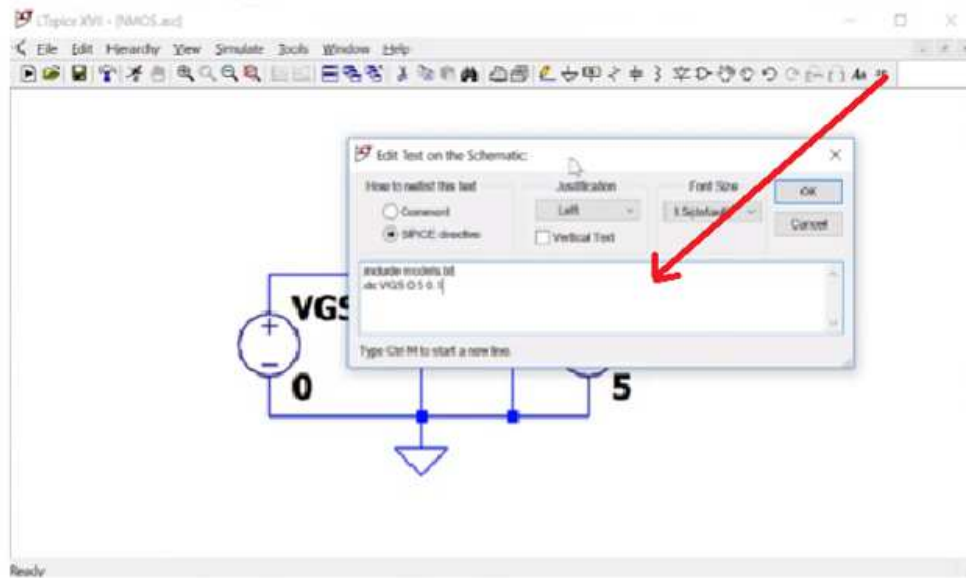
The "Parasitic Properties" section shows the following parameters:

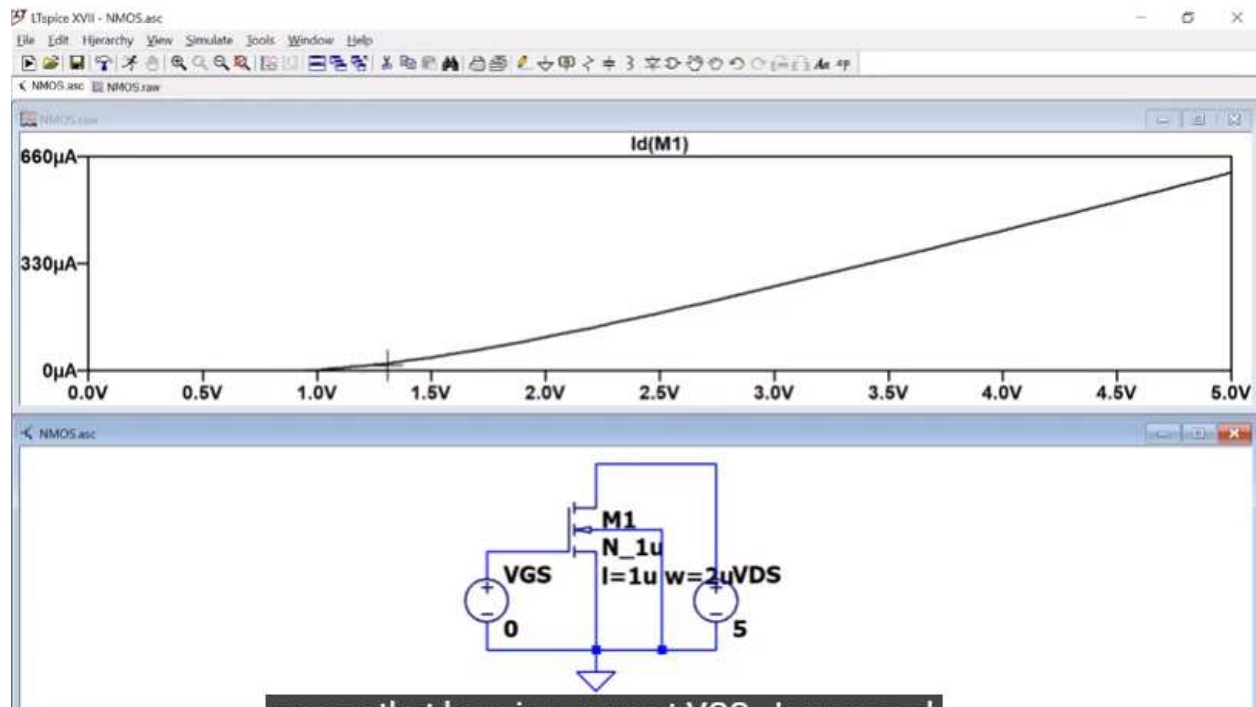
- Series Resistance[Ω]: []
- Parallel Capacitance[F]: []
- Make this information visible on schematic:

The "Additional PwL Points" section shows the following parameters:

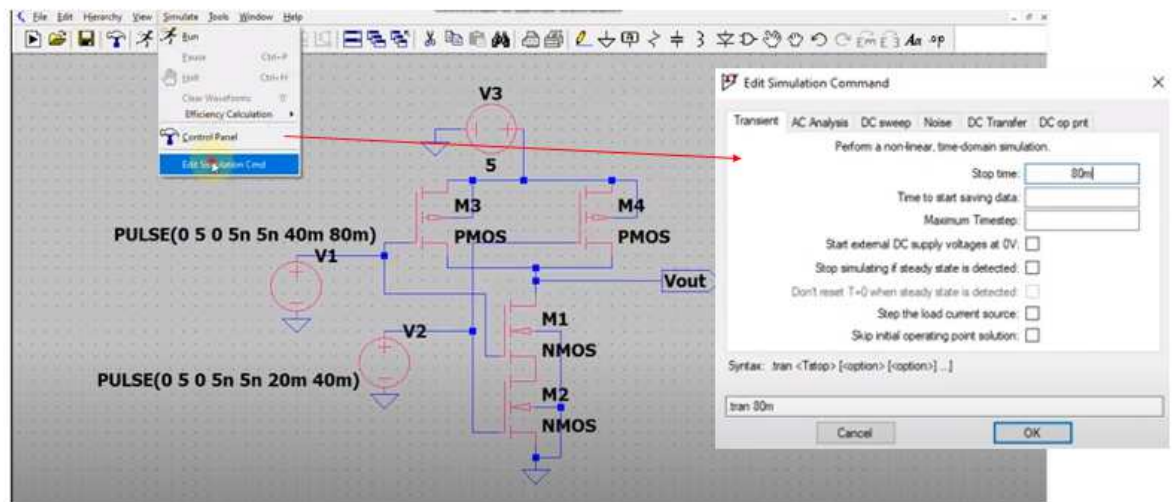
- Additional PwL Points: []
- Make this information visible on schematic:

- DC Sweep

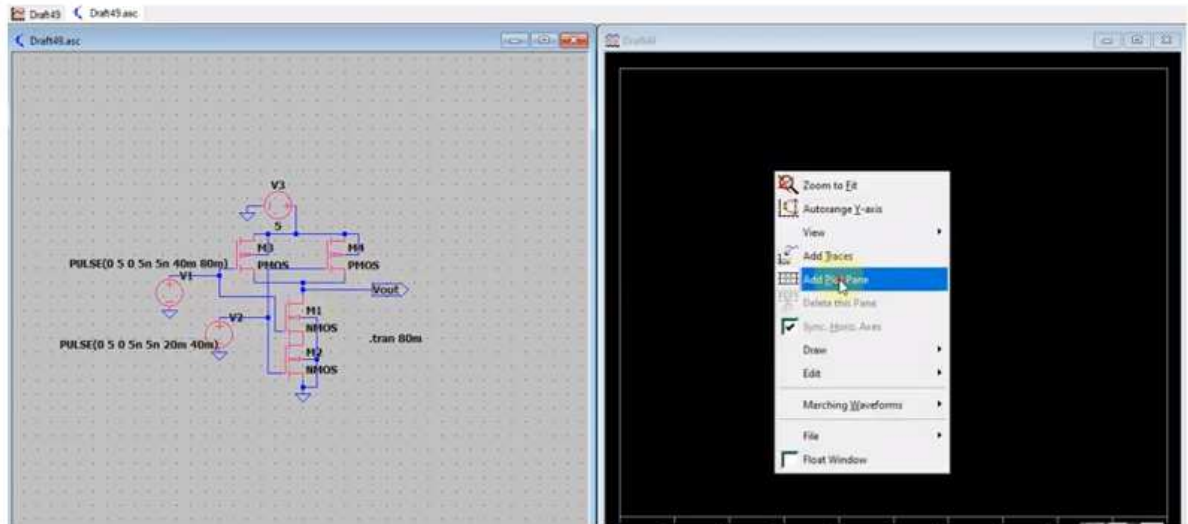




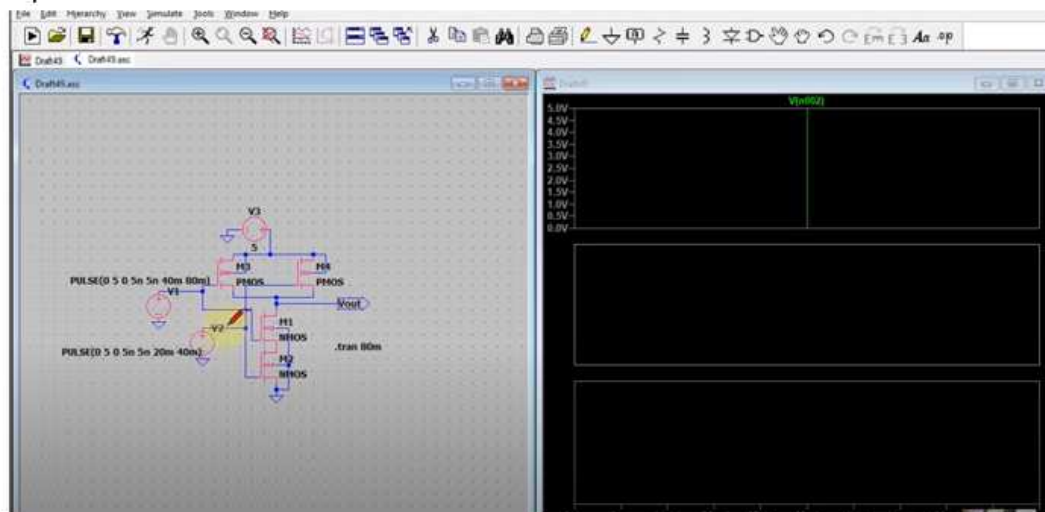
- Run the Simulation for Transient Response



- Add pane to output
- Right click -> add new pane



- Viewing Differential Voltage Waveforms
- **Left-Click** on one node and drag the mouse to another node
- Red voltage probe at the first node
- Black probe on the second



TRANSISTOR MODELS

* Long channel models from CMOS Circuit Design, Layout, and Simulation,

* Level=3 models VDD=5V, see CMOSedu.com

*

.MODEL N_1u NMOS LEVEL = 3

+ TOX = 200E-10	NSUB = 1E17	GAMMA = 0.5
+ PHI = 0.7	VTO = 0.8	DELTA = 3.0
+ UO = 650	ETA = 3.0E-6	THETA = 0.1
+ KP = 120E-6	VMAX = 1E5	KAPPA = 0.3
+ RSH = 0	NFS = 1E12	TPG = 1
+ XJ = 500E-9	LD = 100E-9	
+ CGDO = 200E-12	CGSO = 200E-12	CGBO = 1E-10
+ CJ = 400E-6	PB = 1	MJ = 0.5
+ CJSW = 300E-12	MJSW = 0.5	

*

.MODEL P_1u PMOS LEVEL = 3

+ TOX = 200E-10	NSUB = 1E17	GAMMA = 0.6
+ PHI = 0.7	VTO = -0.9	DELTA = 0.1
+ UO = 250	ETA = 0	THETA = 0.1
+ KP = 40E-6	VMAX = 5E4	KAPPA = 1
+ RSH = 0	NFS = 1E12	TPG = -1
+ XJ = 500E-9	LD = 100E-9	
+ CGDO = 200E-12	CGSO = 200E-12	CGBO = 1E-10
+ CJ = 400E-6	PB = 1	MJ = 0.5
+ CJSW = 300E-12	MJSW = 0.5	

*

*

* Short channel models from CMOS Circuit Design, Layout, and Simulation,

* 50nm BSIM4 models VDD=1V, see CMOSedu.com

*

```
.model N_50n nmos level = 54
+binunit = 1      paramchk= 1      mobmod = 0
+capmod = 2      igcmmod = 1      igbmod = 1      geomod = 0
+diomod = 1      rdsmod = 0      rbodymod= 1      rratemod= 1
+permod = 1      acnqsmmod= 0      trnqsmmod= 0
+tnom  = 27      toxex = 1.4e-009  toxpx = 7e-010  toxmx = 1.4e-009
+epsrox = 3.9    wint  = 5e-009    lint  = 1.2e-008
+ll    = 0      wl    = 0      lln   = 1      wln   = 1
+lw    = 0      ww    = 0      lwn   = 1      wwn   = 1
+lw1   = 0      ww1   = 0      xpart = 0      toxref = 1.4e-009
+vt0   = 0.22   k1    = 0.35   k2    = 0.05   k3    = 0
+k3b   = 0      w0    = 2.5e-006  dvt0  = 2.8    dvt1  = 0.52
+dvt2  = -0.032 dvt0w = 0      dvt1w = 0      dvt2w = 0
+dsub  = 2      minv  = 0.05   voffl  = 0      dvtp0 = 1e-007
+dvtp1 = 0.05   lpe0  = 5.75e-008 lpeb  = 2.3e-010 xj    = 2e-008
+ngate = 5e+020 ndep  = 2.8e+018 nsd   = 1e+020 phin  = 0
+cdsc  = 0.0002 cdscb  = 0      cdsd  = 0      cit   = 0
+voff  = -0.15  nfactor = 1.2   eta0  = 0.15   etab  = 0
+vfb   = -0.55  u0    = 0.032  ua    = 1.6e-010 ub    = 1.1e-017
+uc    = -3e-011 vsat  = 1.1e+005 a0    = 2      ags   = 1e-020
+a1    = 0      a2    = 1      b0    = -1e-020 b1    = 0
+keta  = 0.04   dwg   = 0      dwb   = 0      pclm  = 0.18
+pdibl1 = 0.028 pdibl2 = 0.022 pdiblc = -0.005 drou  = 0.45
+pvag  = 1e-020 delta = 0.01   pscbe1 = 8.14e+008 pscbe2 = 1e-007
+fprout = 0.2   pdits  = 0.2   pditsd = 0.23   pditsl = 2.3e+006
+rsh   = 3      rdsw  = 150   rsw   = 150   rdw   = 150
+rdswmin = 0   rdwmin = 0   rswmin = 0   prwg  = 0
+prwb  = 6.8e-011 wr    = 1      alpha0 = 0.074  alpha1 = 0.005
+beta0 = 30     agidl  = 0.0002 bgidl  = 2.1e+009 cgidl  = 0.0002
+egidl = 0.8
+aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002
+nigbacc = 1    aigbinv = 0.014 bigbinv = 0.004 cigbinv = 0.004
+eigbinv = 1.1 nigbinv = 3    aigc  = 0.017 bigc  = 0.0028
+cigc  = 0.002 aigsd  = 0.017 bigsd  = 0.0028 cigsd  = 0.002
+nigc  = 1     poxedge = 1    pigcd  = 1     ntox  = 1
+xrcrg1 = 12   xrcrg2 = 5
```

```

+cgso = 6.238e-010  cgdo = 6.238e-010  cgbo = 2.56e-011  cgdl = 2.495e-10
+cgsl = 2.495e-10  ckappas = 0.02      ckappad = 0.02      acde = 1
      +moin = 15      noff = 0.9      voffcv = 0.02
+kt1 = -0.21      kt1l = 0.0      kt2 = -0.042      ute = -1.5
+ua1 = 1e-009      ub1 = -3.5e-019  uc1 = 0           prt = 0
      +at = 53000
      +fnoimod = 1      tnoimod = 0
+jss = 0.0001      jsws = 1e-011    jswgs = 1e-010    njs = 1
+ijthsfwd= 0.01    ijthsrev= 0.001    bvs = 10          xjbvs = 1
+jsd = 0.0001      jswd = 1e-011    jswgd = 1e-010    njd = 1
+ijthdfwd= 0.01    ijthdrev= 0.001    bvd = 10          xjbvd = 1
      +pbs = 1          cjs = 0.0005     mjs = 0.5         pbsws = 1
+cjsws = 5e-010    mjsws = 0.33     pbswgs = 1        cjswgs = 5e-010
      +mjswgs = 0.33     pbd = 1          cjd = 0.0005     mjd = 0.5
+pbswd = 1         cjswd = 5e-010   mjswd = 0.33     pbswgd = 1
+cjswgd = 5e-010   mjswgd = 0.33    tpb = 0.005      tcj = 0.001
+tpbsw = 0.005     tcjsw = 0.001    tpbswg = 0.005   tcjswg = 0.001
      +xtis = 3          xtid = 3
+dmcg = 0e-006     dmci = 0e-006    dmdg = 0e-006    dmcgt = 0e-007
      +dwj = 0e-008     xgw = 0e-007     xgl = 0e-008
+rshg = 0.4        gbmin = 1e-010   rbpb = 5          rbpd = 15
      +rbps = 15        rbdb = 15        rbsb = 15        ngcon = 1

```

*

```

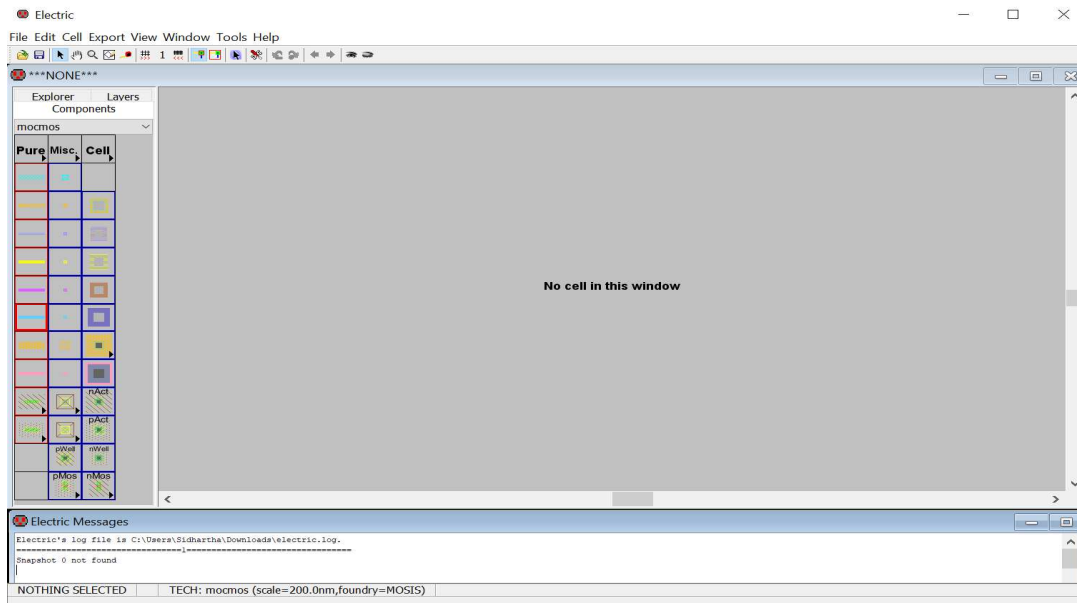
      .model P_50n pmos level = 54
      +binunit = 1      paramchk= 1      mobmod = 0
+capmod = 2          igcmod = 1          igbmod = 1          geomod = 0
+diomod = 1          rdsmod = 0          rbodymod= 1          rgatemod= 1
      +permod = 1      acnqsmod= 0      trnqsmod= 0
+tnom = 27          toxex = 1.4e-009    toxp = 7e-010      toxm = 1.4e-009
      +epsrox = 3.9     wint = 5e-009      lint = 1.2e-008
      +ll = 0           wl = 0             lln = 1            wln = 1
      +lw = 0           ww = 0             lwn = 1            wwn = 1
+lw1 = 0           ww1 = 0             xpart = 0          toxref = 1.4e-009
      +vth0 = -0.22     k1 = 0.39          k2 = 0.05          k3 = 0
+k3b = 0           w0 = 2.5e-006     dvt0 = 3.9          dvt1 = 0.635
      +dvt2 = -0.032     dvt0w = 0          dvt1w = 0          dvt2w = 0
+dsb = 0.7         minv = 0.05        voff1 = 0           dvtp0 = 0.5e-008
+dvtp1 = 0.05      lpe0 = 5.75e-008   lpeb = 2.3e-010    xj = 2e-008
+ngate = 5e+020     ndep = 2.8e+018    nsd = 1e+020        phin = 0
      +cdsc = 0.000258   cdscb = 0          cdsd = 6.1e-008    cit = 0
      +voff = -0.15     nfactor = 2        eta0 = 0.15        etab = 0
+vfb = 0.55        u0 = 0.0095        ua = 1.6e-009       ub = 8e-018
+uc = 4.6e-013     vsat = 90000       a0 = 1.2            ags = 1e-020

```


ELECTRIC VLSI DESIGN EDA TOOL

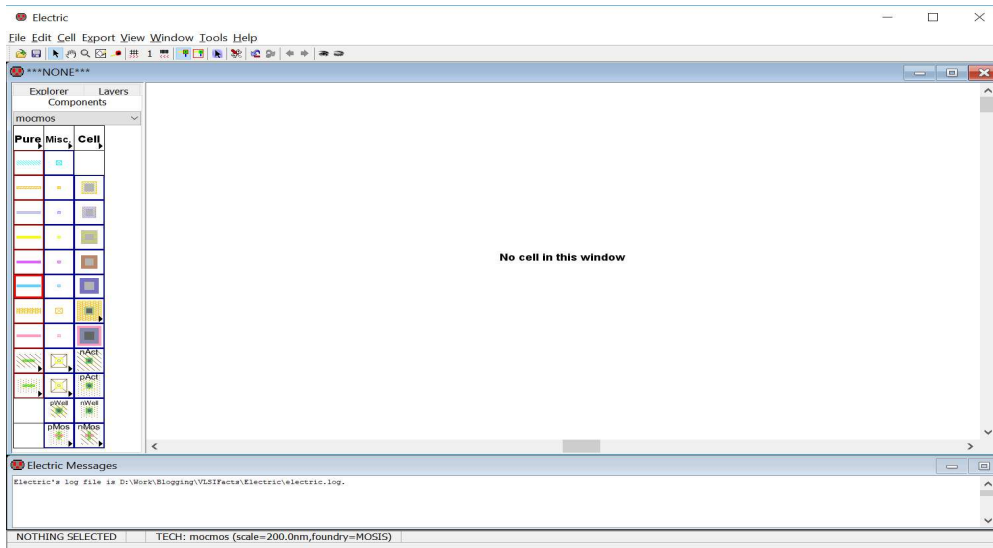
Procedure for simulation and implementation of EDA tool-ELECTRIC

1. Start Electric: The following window will appear.



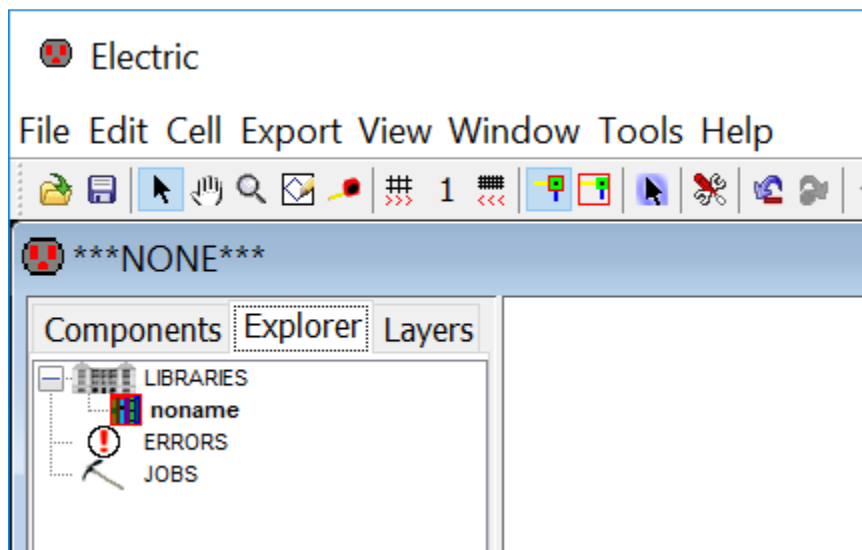
2. Towards the bottom of the window, Electric Messages Window will be found where different messages can be found throughout any design.
3. The background color of the window can change as follows

Window -> Color Schemes -> White Background Colors



4. Let's create a Library

Go to **Explorer** (beside the **Components** view); you will find **LIBRARIES** name as no name



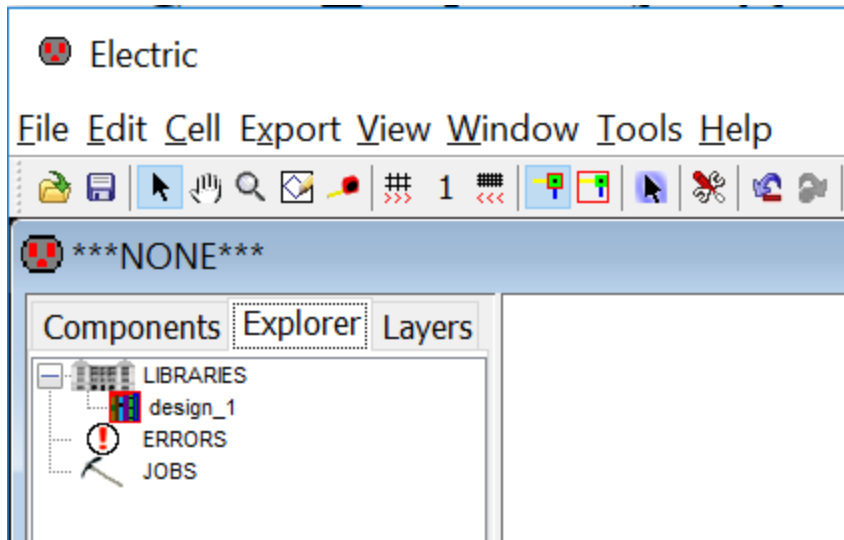
5. File -> Save Library As

Go to the location where design have to be save. (eg: $\$PATH/Electric/Designs$)

Name the design (library name) eg. design_1.jelib

We would create our schematic and layout under this library

Now you will see design_1.jelib under LIBRARIES name in Explorer



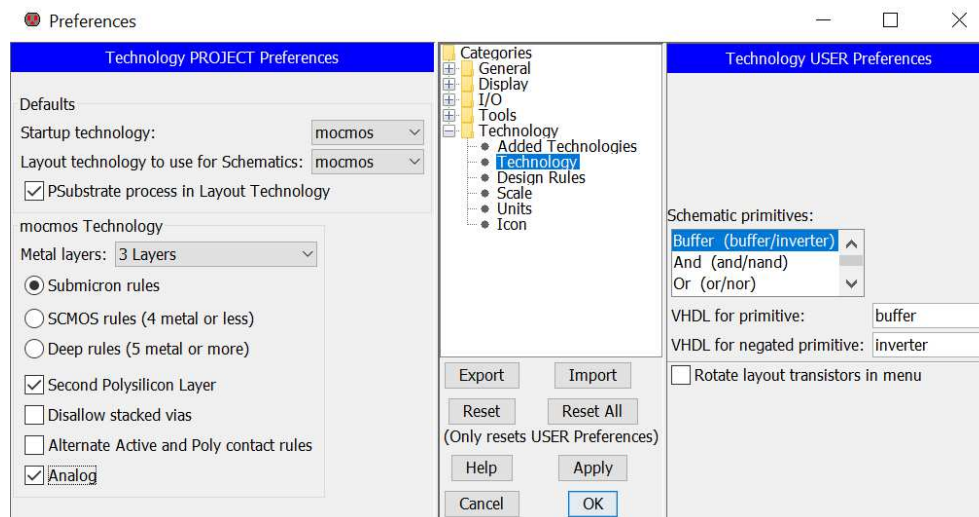
6. Go to Preferences by clicking the following button or executing File → Preferences...
7. Then you have to set the following.

Preferences → Categories → Technology → Technology

mocmos Technology → Metal layers → 3 Layers

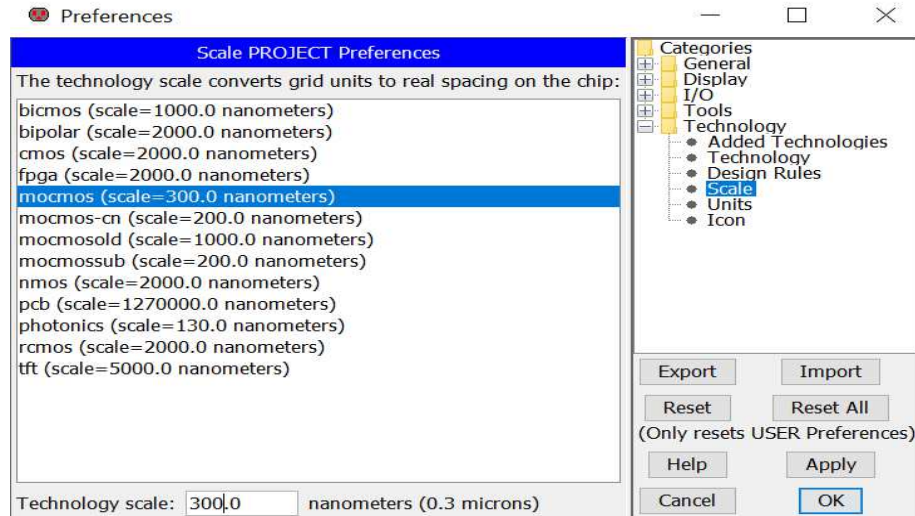
Keep submicron rules and Second Polysilicon Layer checked

Click the Analog checkbox.



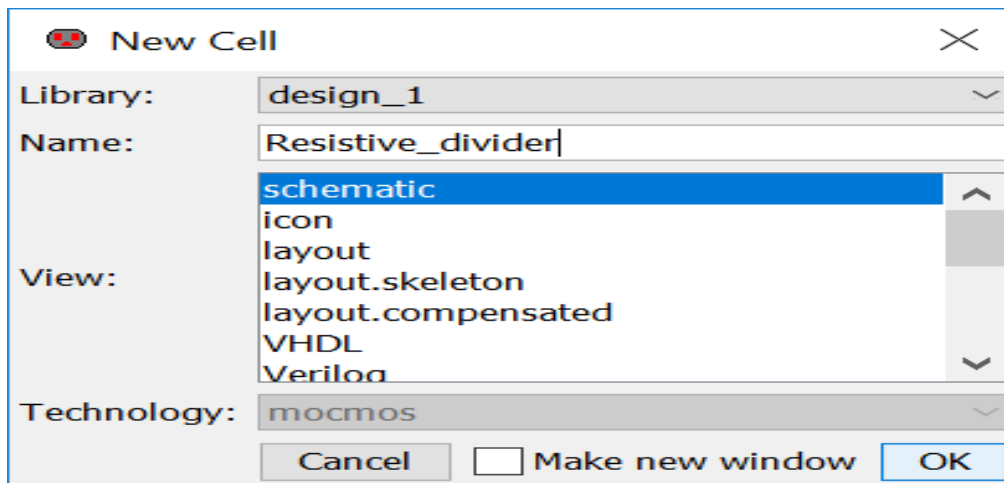
8. To set the scale go to

File → Preferences → Technology → Scale and set mocmos scale to 300 nm



9. Creating a new cell

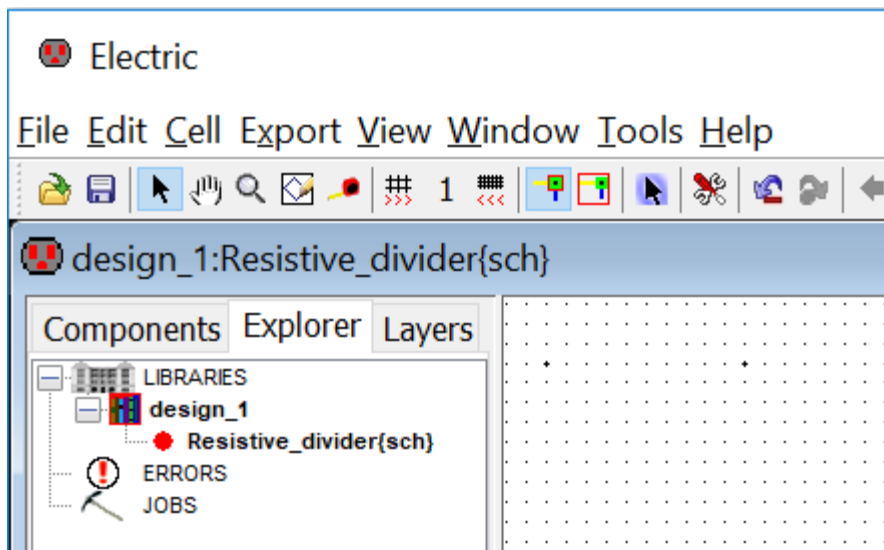
Go to cell → New Cell (or you can press ctrl + N). You will find a window like following.



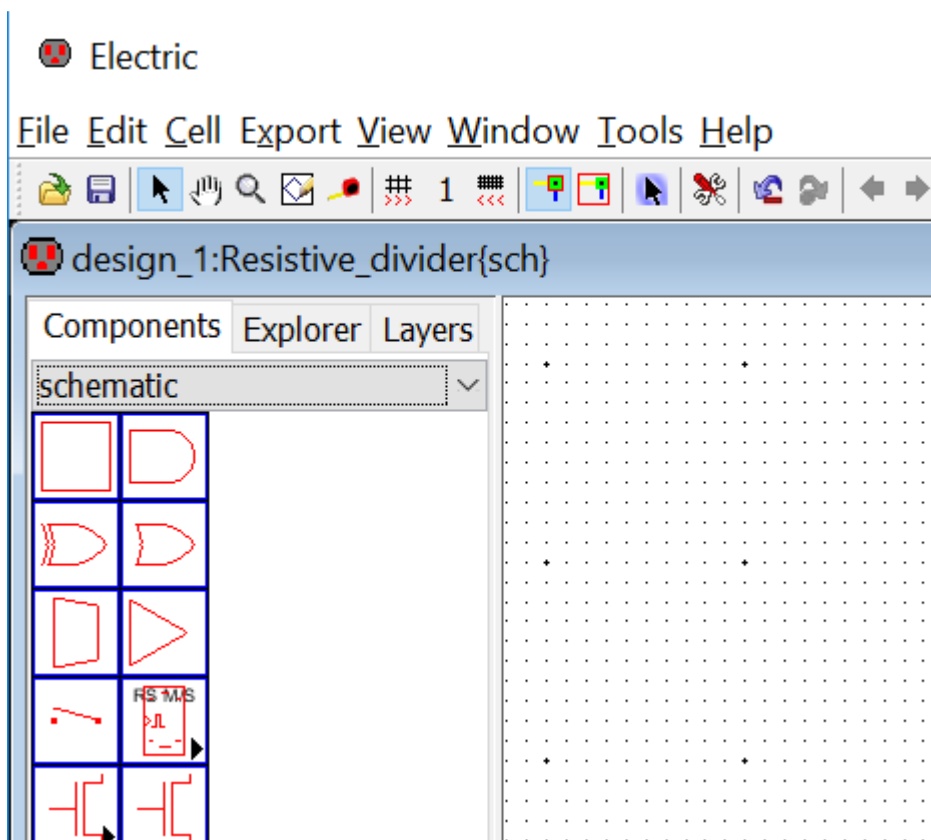
Enter the **name** of the cell {-----} and click the **view** as {schematic}.

Press **ok**.

Now under the **library design_1.jelib** you can find a **schematic cell** named as ----- {sch} with a **red indicator** as follows.



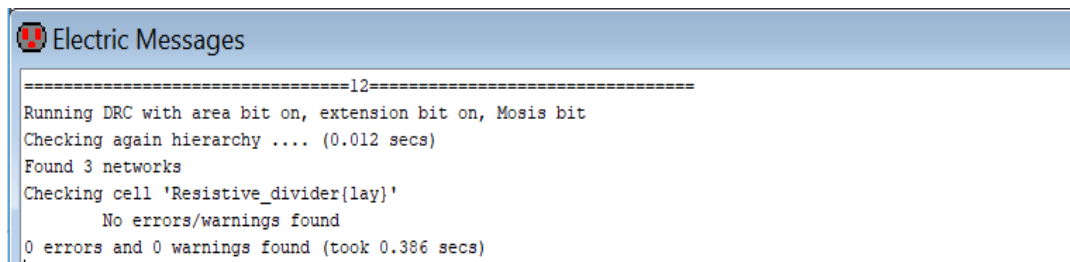
10. Now Press the **Components**. You will find the schematic components unlike the layout components in the startup window.



11. Now we are finished with the setup and ready to fabricate a chip in the C5 process via MOSIS
12. Checking of DRC (Design Rule Check)

To check DRC you can execute Tools -> DRC -> Check Hierarchically or you can press F5.

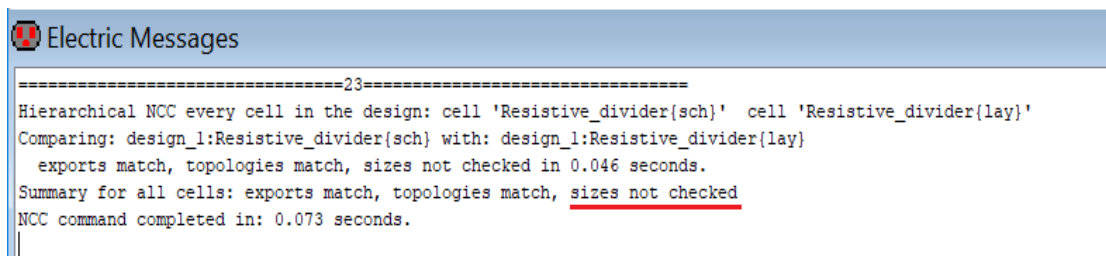
Once DRC is checked, you can see result in the **message window** as follows:



```
Electric Messages
=====12=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.012 secs)
Found 3 networks
Checking cell 'Resistive_divider{lay}'
    No errors/warnings found
0 errors and 0 warnings found (took 0.386 secs)
```

13. Layout vs. Schematic (LVS) in Electric is checked using Network Consistency Checking (NCC)

To check this, execute Tools -> NCC -> Schematic and Layout views of Cell in Current Window. You can run this command being in any design window (schematic / layout).



```
Electric Messages
=====23=====
Hierarchical NCC every cell in the design: cell 'Resistive_divider{sch}' cell 'Resistive_divider{lay}'
Comparing: design_1:Resistive_divider{sch} with: design_1:Resistive_divider{lay}
    exports match, topologies match, sizes not checked in 0.046 seconds.
Summary for all cells: exports match, topologies match, sizes not checked
NCC command completed in: 0.073 seconds.
```

We found a message like **sizes not checked**.

For this we have to take care of the following

Go to File -> Preferences -> Categories -> Tools -> NCC -> Check transistor sizes

Once again execute the NCC, now you will find the following message.

```
Electric Messages
=====24=====
Hierarchical NCC every cell in the design: cell 'Resistive_divider{sch}' cell 'Resistive_divider{lay}'
Comparing: design_1:Resistive_divider{sch} with: design_1:Resistive_divider{lay}
  exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.001 seconds.
```

14. Checking ERC (Well Check)

This process checks the connection of the n-well and p-substrate.

The C5 process used here is an n-well process. The p-type substrate is common to all NMOS devices and should be grounded.

One of the **electrical rule checks** (ERCs) is to verify that the **p-well** (in this case p-substrate) is always connected to **ground**.

Further, in this n-well process, if the design contains only **digital circuits** then the **n-well** should be connected to **V_{DD}**.

For Well Check execute Tools -> ERC -> Check Wells or press W (as we have bounded this key to Well Check).

```
Electric Messages
=====38=====
Checking Wells and Substrates in 'design_1:Resistive_divider{lay}' ...
  Geometry collection found 1 well pieces, took 0.003 secs
  Geometry analysis used 4 threads and took 0.003 secs
NetValues propagation took 0.0 secs
Checking short circuits in 2 well contacts
  Additional analysis took 0.005 secs
FOUND 2 WELL ERRORS (took 0.012 secs)
ERC Well Check found 2 errors, 0 warnings!
Type > and < to step through errors, or open the ERRORS view in the explorer
=====39=====
ERC Well Check error 1 of 2: N-Well contact 'resnwell@0' not connected to ground
```

The reason is as below:

In Digital Design all the N-Wells to be connected to V_{DD} and all the P-Wells to be connected to Ground.

Here we can see For N-Well, **Must connect to Power** is checked.

But this Resistive_divider is not a digital design. Here the N-Well is used as a resister which is an **anlog design**.

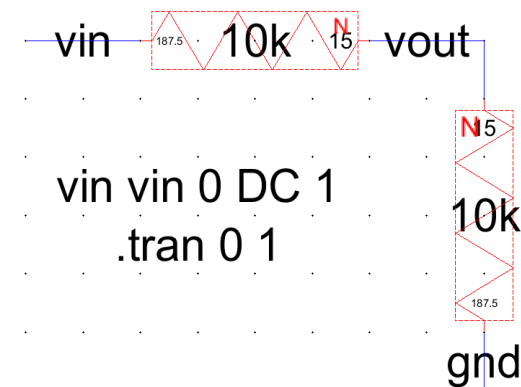
So **uncheck** “Must connect to Power” under “For N-Well”. You will find zero Well Check error.

15. Schematic Simulation

Now we would **simulate** the resistive divider circuit which has been built, and would observe the output voltage w.r.t. a particular input voltage.

For this we need to write a **SPICE code** which would give the description of the input voltage and would indicate the type of simulation we want to perform.

Writing SPICE Code



Go to the **Components menu**. Click on the arrowhead in the **Misc box** to add **SPICE code** to the schematic as seen in the figure.

Place the SPICE code in the **schematic** and use Ctrl+I to edit its properties.

Ensure, in the SPICE code property box, that the **Multi-line Text box is checked**.

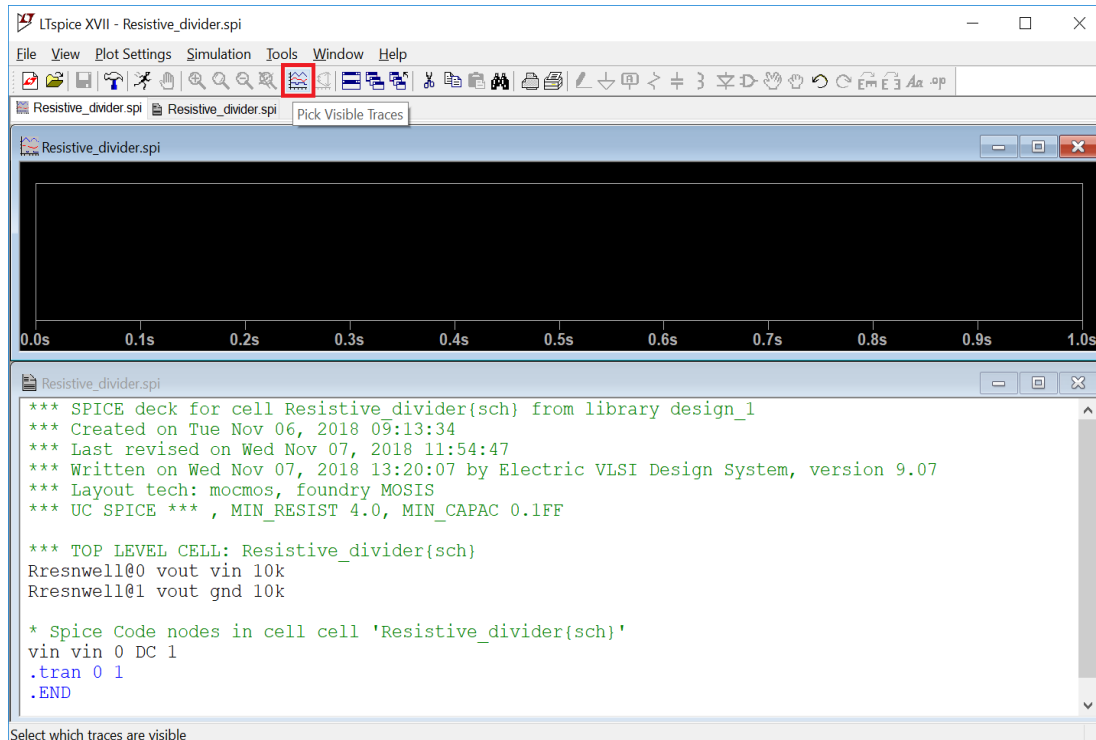
Add the code shown in the figure for specifying a **SPICE transient analysis** and an **input voltage source**. The code indicates an input voltage of 1 V DC is applied to the circuit. The analysis would be a transient one for 1 second.

Press F5 to check the schematic.

16. Simulation of the Schematic

Go to Tools -> Simulation (Spice) -> Write Spice Deck.

The following LTspice window will open.



17. Resistive Divider Layout

Open the layout view of the Resistive_divider cell and then copy/paste (Ctrl+C/Ctrl+V) an **additional resistor**.

Running a **DRC** (pressing F5) on the above layout results in the following **error**.

By pressing > we see that there is **too little space between the N-wells**.

Move the Nodes apart until the layout passes the DRCs. Of-course the error will not appear if you have initially placed both the resistors apart enough, which would **satisfy the MOSIS rule** for space between N-wells.

Electric

File Edit Cell Export View Window Tools Help

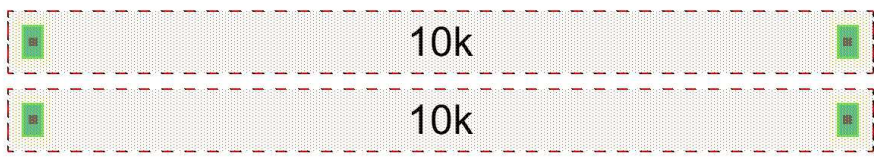
design_1:Resistive_divider(lay)

Components Explorer Layers

LIBRARIES

- design_1
 - Resistive_divider
 - Resistive_dividerSc
 - Resistive_divider1

ERRORS JOBS



10k

10k

Electric Messages

Checking cell 'Resistive_divider(lay)'

FOUND 2 ERRORS

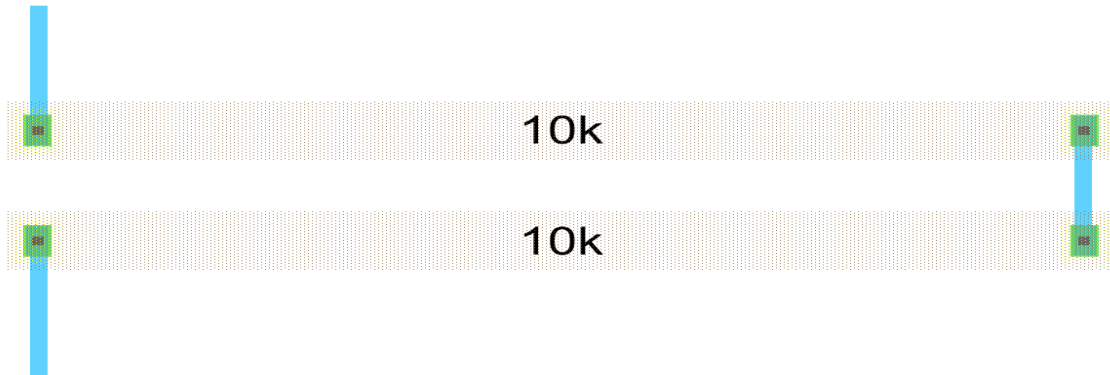
2 errors and 0 warnings found (took 0.027 secs)

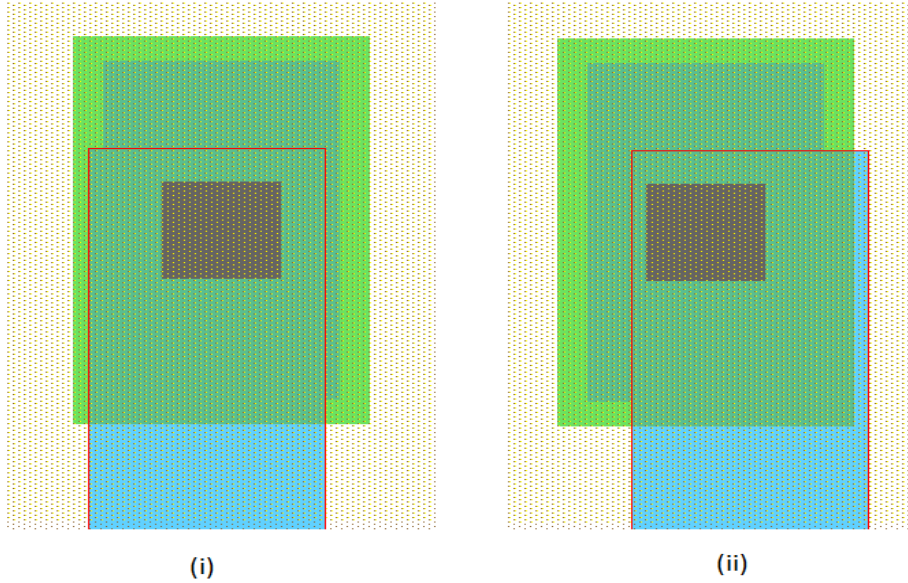
Layout DRC (full) found 2 errors, 0 warnings!

Type > and < to step through errors, or open the ERRORS view in the explorer

Layout DRC (full) error 1 of 2: Notch (layer 'N-Well'): node N-Well-Resistor['resw1180'] LESS (BY 2) THAN 6 TO node N-Well-Resistor['resw1181'] [Rule '1.3 Mosis']

NOTHING SELECTED SIZE: 208.5 x 34 TECH: mocmos (scale=300.0nm,foundry=MOSIS) (115, 23)





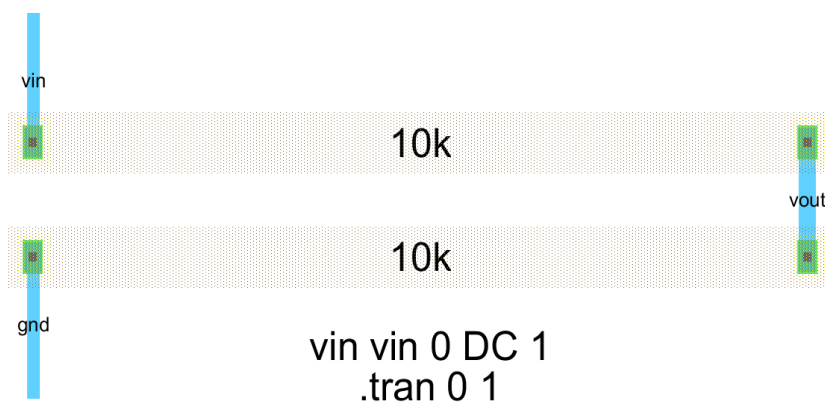
Run DRC to check the design is free of error or not.

This layout cell should match the schematic cell. Verify this by running the NCC (aka LVS check).

The following figure shows the Electric Messages for **DRC of layout** and **NCC of both layout and schematic (LVS)**.

18. Layout Simulation

The following figure shows the visible spice code.

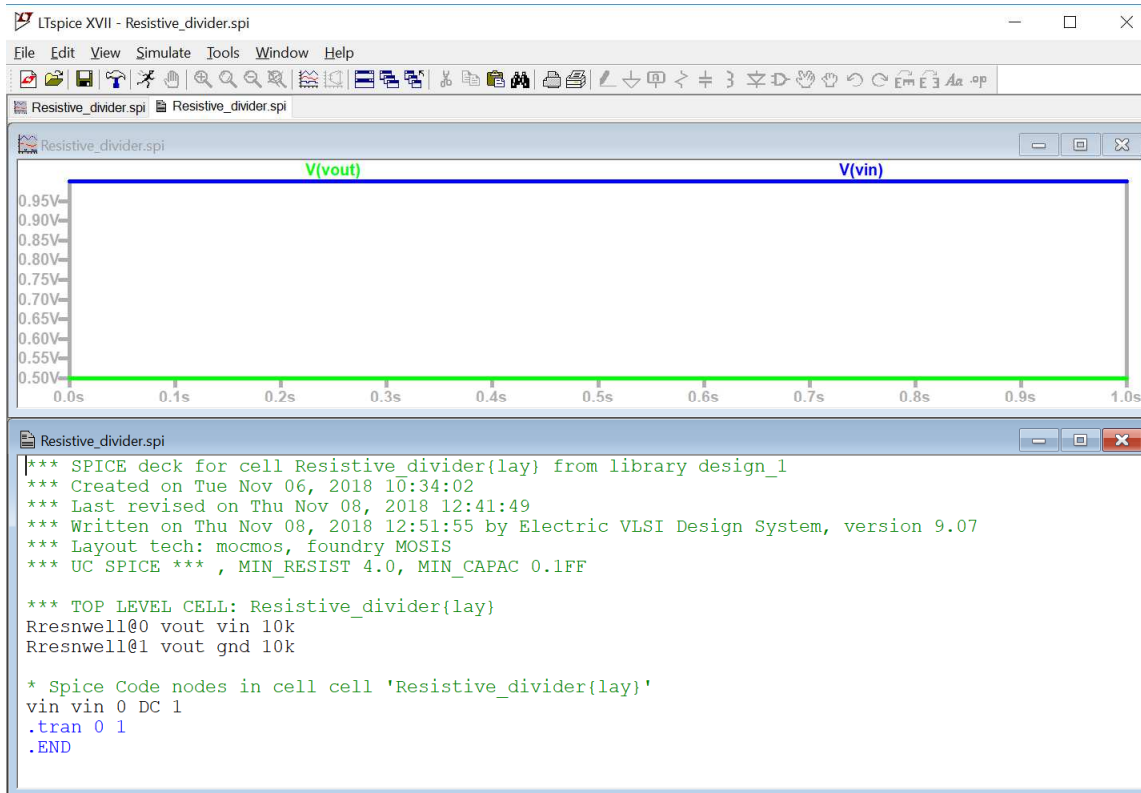


Run a DRC, NCC, and a Well Check to ensure that there aren't any errors.

This cell can be simulated following the same steps used for simulating the schematic view above.

Simulate this cell using LTspice now.

The following figure shows the simulation output from LTspice for the Resistive_divider layout.



*** BSIM3 models for AMI Semiconductor's C5 process**

*

* Don't forget the .options scale=300nm if using drawn lengths

* and the MOSIS SUBM design rules

*

* $2 < L_{drawn} < 500$ $10 < W_{drawn} < 10000$ $V_{dd} = 5V$

* Note minimum L is 0.6 um while minimum W is 3 um

* Change to level=49 when using HSPICE or SmartSpice

```
.MODEL NMOS NMOS (          LEVEL = 8
+VERSION = 3.1      TNOM  = 27      TOX  = 1.39E-8
+XJ   = 1.5E-7      NCH   = 1.7E17   VTH0 = 0.6696061
+K1   = 0.8351612   K2    = -0.0839158   K3   = 23.1023856
+K3B  = -7.6841108   W0    = 1E-8      NLX  = 1E-9
+DVT0W = 0          DVT1W = 0          DVT2W = 0
+DVT0  = 2.9047241   DVT1  = 0.4302695   DVT2  = -0.134857
+U0   = 458.439679   UA    = 1E-13      UB   = 1.485499E-18
+UC   = 1.629939E-11 VSAT  = 1.643993E5   A0   = 0.6103537
+AGS  = 0.1194608   B0    = 2.674756E-6   B1   = 5E-6
+KETA  = -2.640681E-3 A1    = 8.219585E-5   A2   = 0.3564792
+RDSW  = 1.387108E3  PRWG  = 0.0299916   PRWB  = 0.0363981
+WR   = 1           WINT  = 2.472348E-7   LINT  = 3.597605E-8
+XL   = 0           XW    = 0           DWG   = -1.287163E-8
+DWB  = 5.306586E-8 VOFF  = 0           NFACTOR = 0.8365585
+CIT  = 0           CDSC  = 2.4E-4      CDSCD = 0

+CDSCB = 0          ETA0  = 0.0246738   ETAB  = -1.406123E-3
+DSUB  = 0.2543458  PCLM  = 2.5945188   PDIBLC1 = -0.4282336
```

+PDIBLC2 = 2.311743E-3 PDIBLCB = -0.0272914 DROUT = 0.7283566
+PSCBE1 = 5.598623E8 PSCBE2 = 5.461645E-5 PVAG = 0
+DELTA = 0.01 RSH = 81.8 MOBMOD = 1
+PRT = 8.621 UTE = -1 KT1 = -0.2501
+KT1L = -2.58E-9 KT2 = 0 UA1 = 5.4E-10
+UB1 = -4.8E-19 UC1 = -7.5E-11 AT = 1E5
+WL = 0 WLN = 1 WW = 0
+WWN = 1 WWL = 0 LL = 0
+LLN = 1 LW = 0 LWN = 1
+LWL = 0 CAPMOD = 2 XPART = 0.5
+CGDO = 2E-10 CGSO = 2E-10 CGBO = 1E-9
+CJ = 4.197772E-4 PB = 0.99 MJ = 0.4515044
+CJSW = 3.242724E-10 PBSW = 0.1 MJSW = 0.1153991
+CJSWG = 1.64E-10 PBSWG = 0.1 MJSWG = 0.1153991
+CF = 0 PVTH0 = 0.0585501 PRDSW = 133.285505
+PK2 = -0.0299638 WKETA = -0.0248758 LKETA = 1.173187E-3
+AF = 1 KF = 0)

*

.MODEL PMOS PMOS (LEVEL = 8
+VERSION = 3.1 TNOM = 27 TOX = 1.39E-8
+XJ = 1.5E-7 NCH = 1.7E17 VTH0 = -0.9214347
K1 = 0.5553722 K2 = 8.763328E-3 K3 = 6.3063558
+K3B = -0.6487362 W0 = 1.280703E-8 NLX = 2.593997E-8
+DVT0W = 0 DVT1W = 0 DVT2W = 0
+DVT0 = 2.5131165 DVT1 = 0.5480536 DVT2 = -0.1186489

+U0 = 212.0166131 UA = 2.807115E-9 UB = 1E-21
+UC = -5.82128E-11 VSAT = 1.713601E5 A0 = 0.8430019

+AGS = 0.1328608 B0 = 7.117912E-7 B1 = 5E-6
KETA = -3.674859E-3 A1 = 4.77502E-5 A2 = 0.3
+RDSW = 2.837206E3 PRWG = -0.0363908 PRWB = -1.016722E-5
+WR = 1 WINT = 2.838038E-7 LINT = 5.528807E-8
+XL = 0 XW = 0 DWG = -1.606385E-8
+DWB = 2.266386E-8 VOFF = -0.0558512 NFACTOR = 0.9342488
+CIT = 0 CDSC = 2.4E-4 CDSCD = 0
+CDSCB = 0 ETA0 = 0.3251882 ETAB = -0.0580325
DSUB = 1 PCLM = 2.2409567 PDIBLC1 = 0.0411445
+PDIBLC2 = 3.355575E-3 PDIBLCB = -0.0551797 DROUT = 0.2036901
+PSCBE1 = 6.44809E9 PSCBE2 = 6.300848E-10 PVAG = 0
+DELTA = 0.01 RSH = 101.6 MOBMOD = 1
+PRT = 59.494 UTE = -1 KT1 = -0.2942
+KT1L = 1.68E-9 KT2 = 0 UA1 = 4.5E-9
+UB1 = -6.3E-18 UC1 = -1E-10 AT = 1E3
+WL = 0 WLN = 1 WW = 0
+WWN = 1 WWL = 0 LL = 0
+LLN = 1 LW = 0 LWN = 1
+LWL = 0 CAPMOD = 2 XPART = 0.5
+CGDO = 2.9E-10 CGSO = 2.9E-10 CGBO = 1E-9
+CJ = 7.235528E-4 PB = 0.9527355 MJ = 0.4955293
+CJSW = 2.692786E-10 PBSW = 0.99 MJSW = 0.2958392
+CJSWG = 6.4E-11 PBSWG = 0.99 MJSWG = 0.2958392
+CF = 0 PVTH0 = 5.98016E-3 PRDSW = 14.8598424
+PK2 = 3.73981E-3 WKETA = 5.292165E-3 LKETA = -4.205905E-3
AF = 1 KF = 0)

EXP NO: 7

Date:

Ring Oscillator

AIM:

To design, analyze and simulate the ring oscillator using LT-SPICE.

APPARATUS REQUIRED:

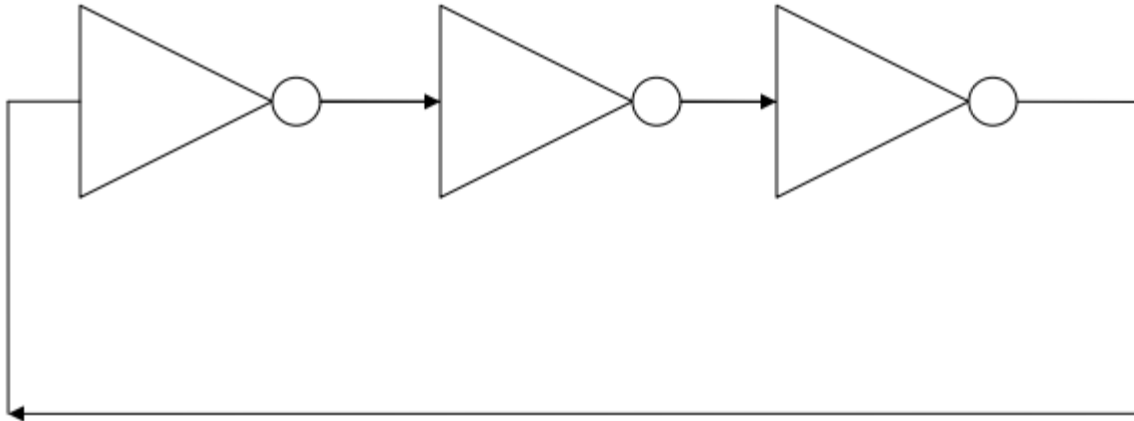
S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	LT-SPLICE	1

PROCEDURE:

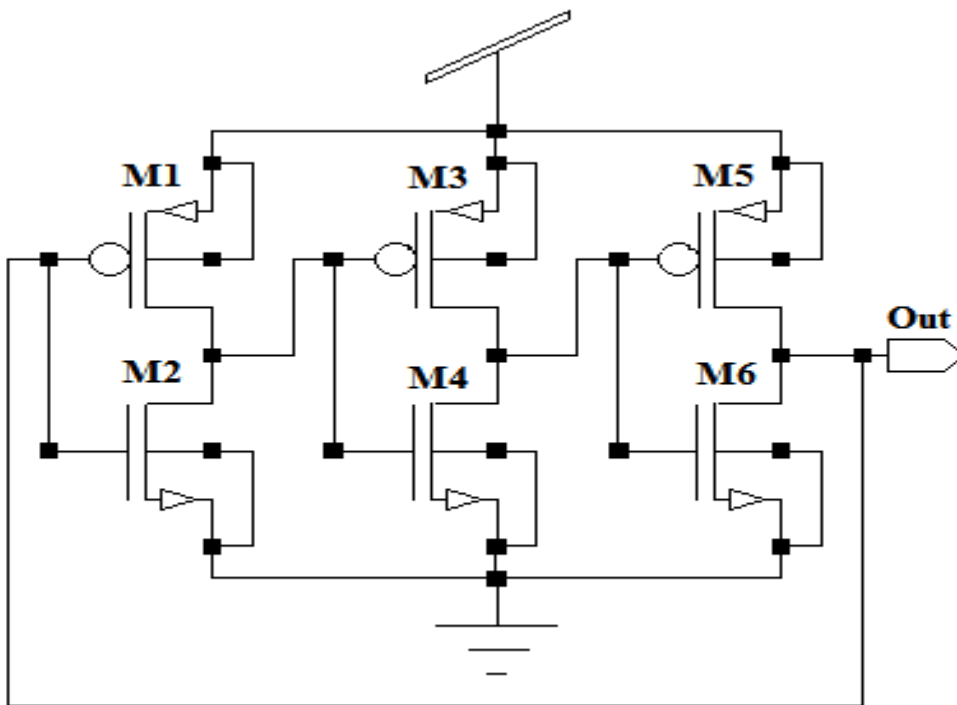
1. Start “LTSpice XVII” (or earlier version)
2. Start a new Project under the File -> New Schematic
3. Make sure , files are saved in a convenient directory. The root directory (C:\) or Desktop are probably not good choices. I would suggest creating a directory “C:\Circuits” and saving your work there
4. Double click on LTspice XVII item →Select the file menu → double click save button
5. Click on component button →click type nmos→ click on nmos4 item → select ok
→click left in screen → press cntl+E for required number of nmos
6. Follow step 5 for selecting pmos4 device.
7. Click on wire button and give connection in circuit diagram
8. Click the ground button and place in screen and give connection using wire.
9. Click voltage in component list and place in screen. Give connection using wire.
10. Click on spice directive button and type “.include level 3 and 54.txt” then click ok
11. Click File menu → select save as and select desktop item outline item and select the file name and type the name of file then give ok.
12. Click on run button.
13. Obtain the Transient analysis of Ring oscillator

Circuit Diagram:

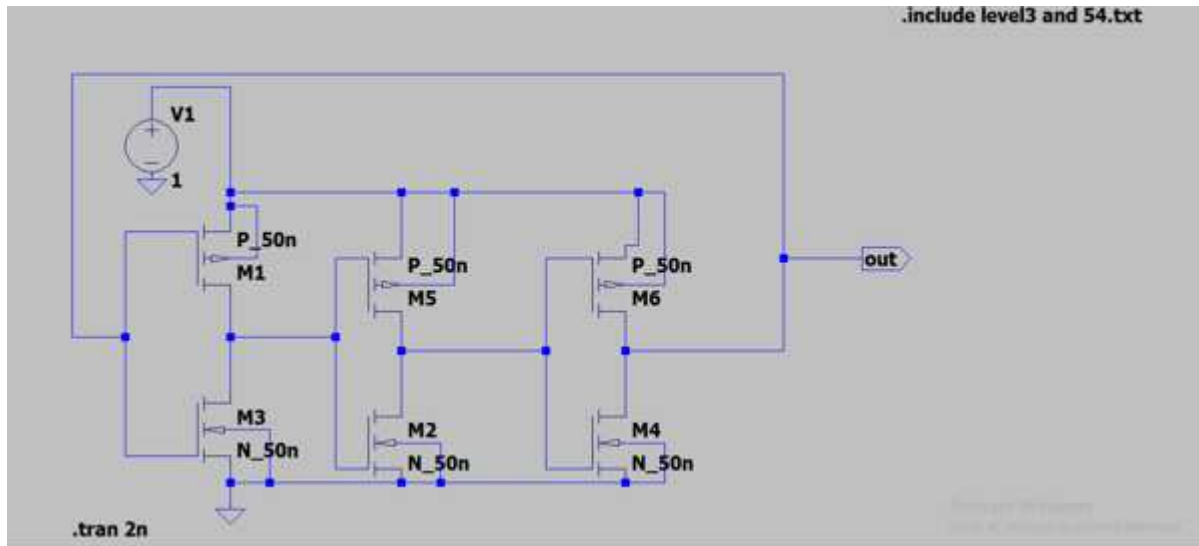
3 stage ring oscillator



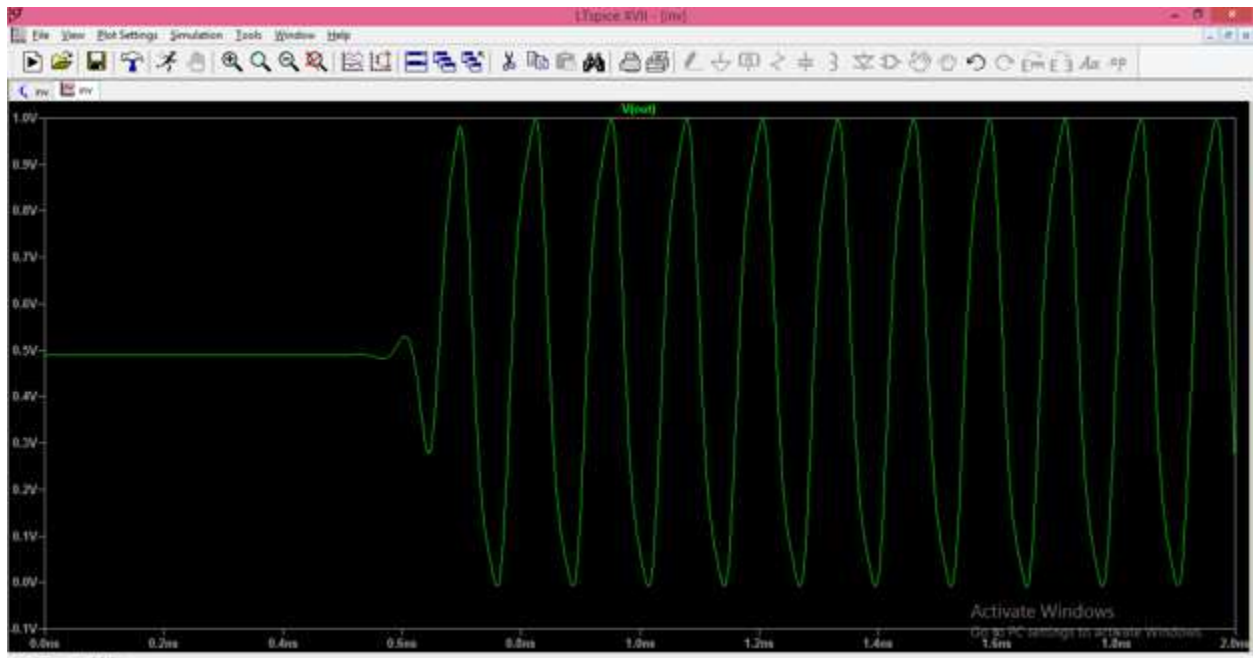
3 stage ring oscillator using CMOS



SCHEMATIC:



SIMULATION OUTPUT:



Calculation:

The frequency of oscillation formula for ring oscillator is

$$f = \frac{1}{2nT}$$

Here T = time delay for single inverter

n = number of inverters in the oscillator

RESULT:

Thus the Ring oscillator is simulated using LT-SPICE.

EXP NO: 8
Date:

Differential Amplifier

AIM:

To design, analyze and simulate the Differential Amplifier using LT-SPICE.

APPARATUS REQUIRED:

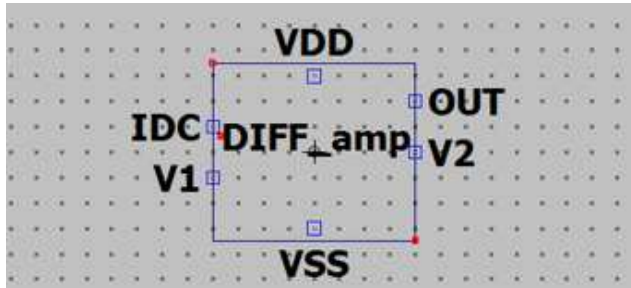
S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	LT-SPLICE	1

PROCEDURE:

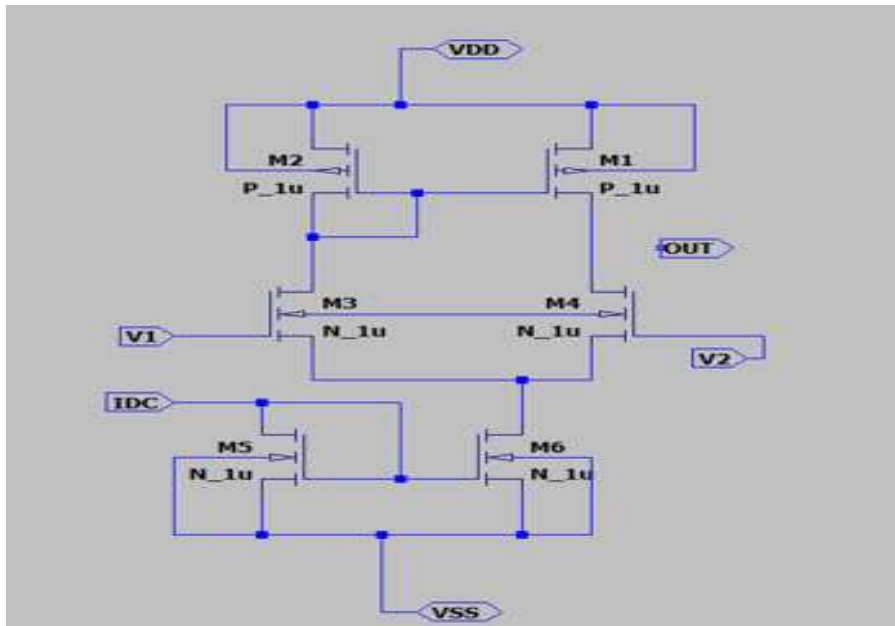
1. Start “LTSpice XVII” (or earlier version)
2. Start a new Project under the File -> New Schematic
3. Make sure , files are saved in a convenient directory. The root directory (C:\) or Desktop are probably not good choices. I would suggest creating a directory “C:\Circuits” and saving your work there
4. Double click on LTspice XVII item →Select the file menu → double click save button
5. Click on component button →click type nmos→ click on nmos4 item → select ok
→click left in screen → press cntl+E for required number of nmos
6. Follow step 5 for selecting pmos4 device.
7. Connect the Circuit as schematic.
8. Click on spice directive button and type “.include BISM4_models.txt” then click ok
9. Click File menu → select save as and select desktop item outline item and select the file name and type the name of file then give ok.
10. .To Obtain the AC analysis, Click on spice directive button and type “.ac oct 20 7 150” then click ok
11. To Obtain the DC analysis, Click on spice directive button and type “.dc V1 -5 5 1m” then click ok
12. To Obtain the Transient analysis,Click on spice directive button and type “.tran 4m” then click ok

SCHEMATIC:

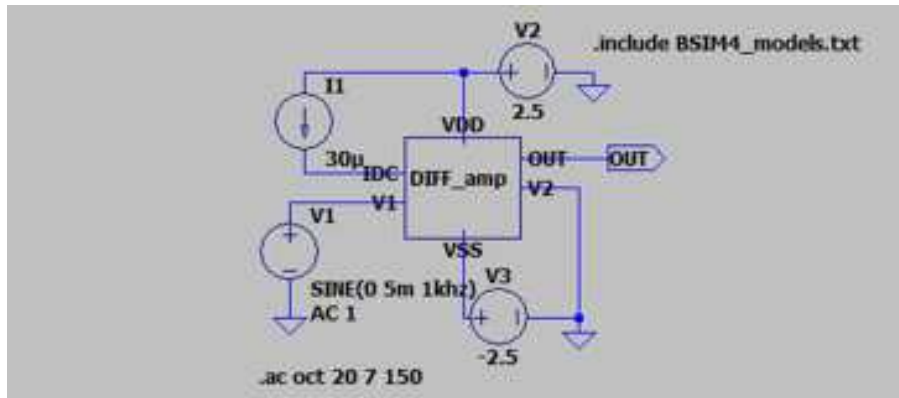
Symbol:



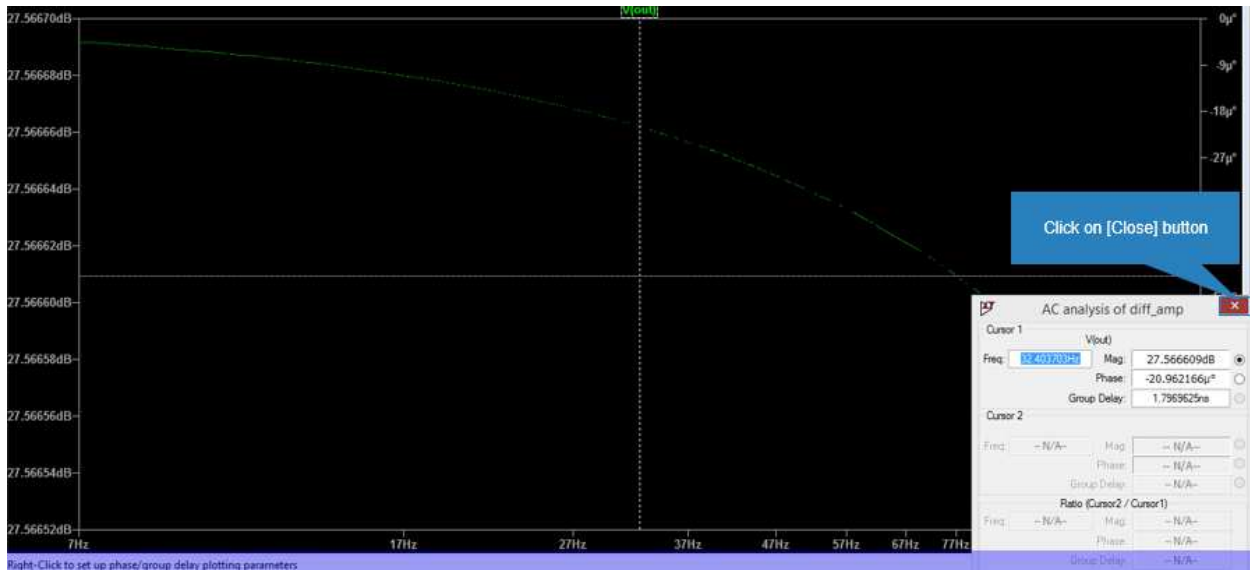
Circuit:



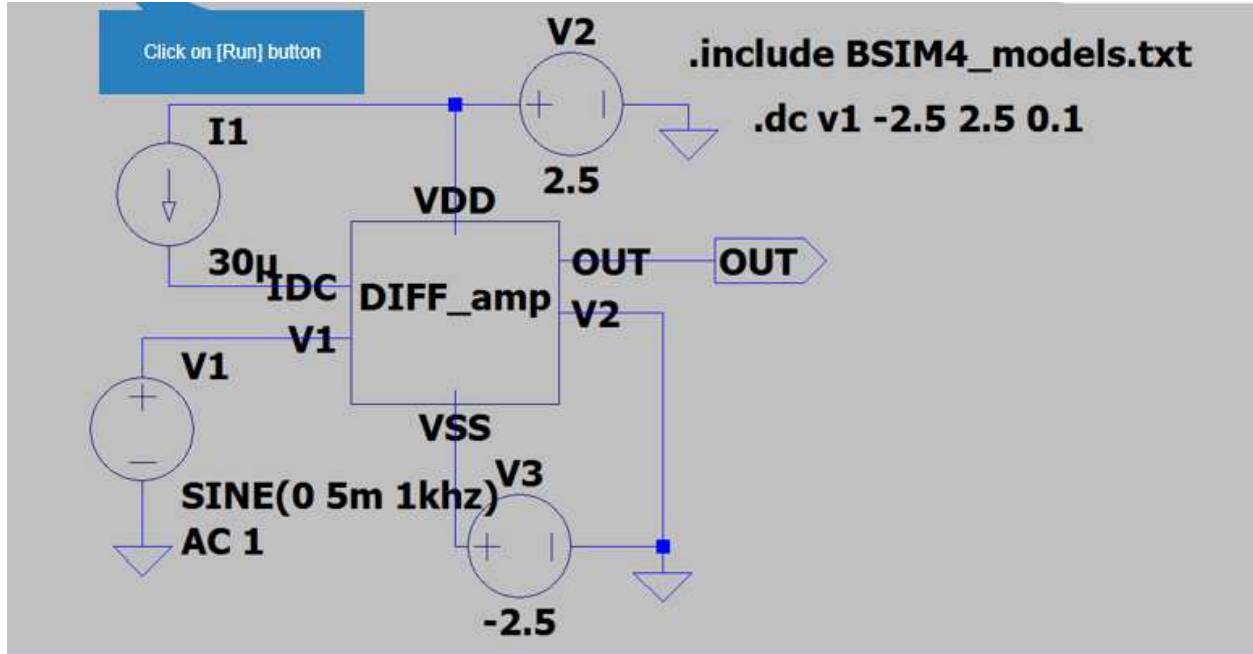
AC ANALYSIS:



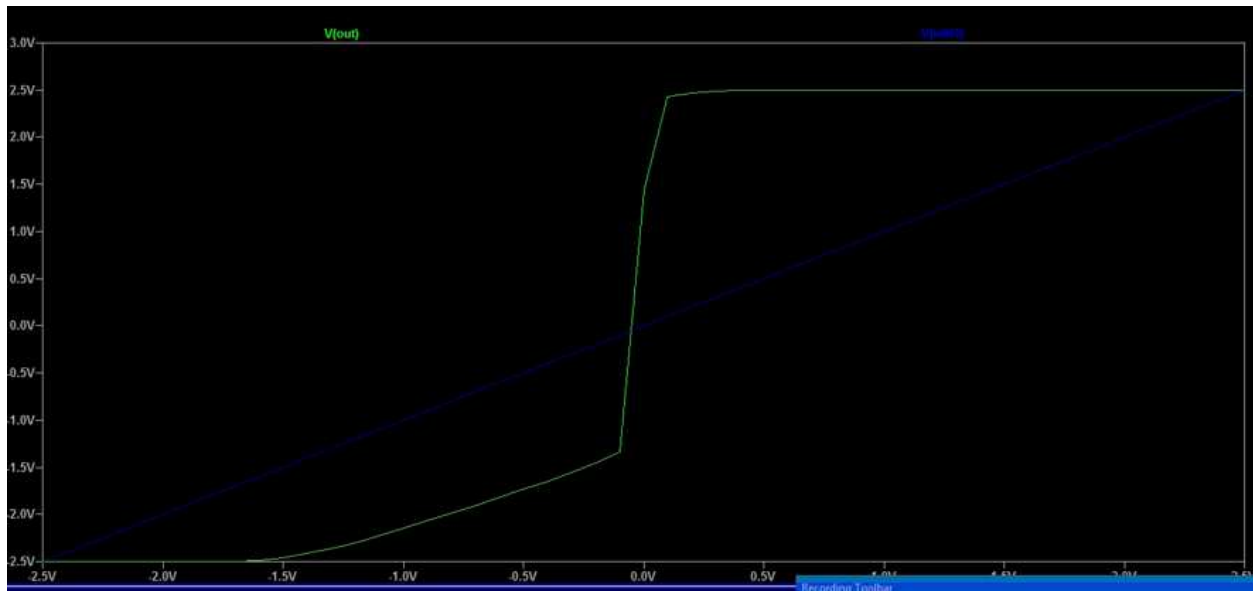
SIMULATION OUTPUT:



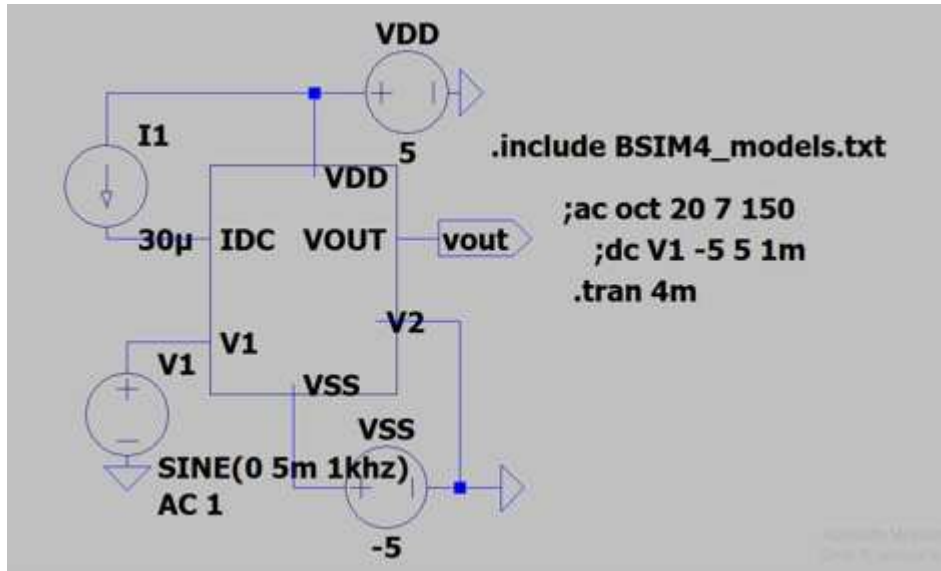
DC ANALYSIS



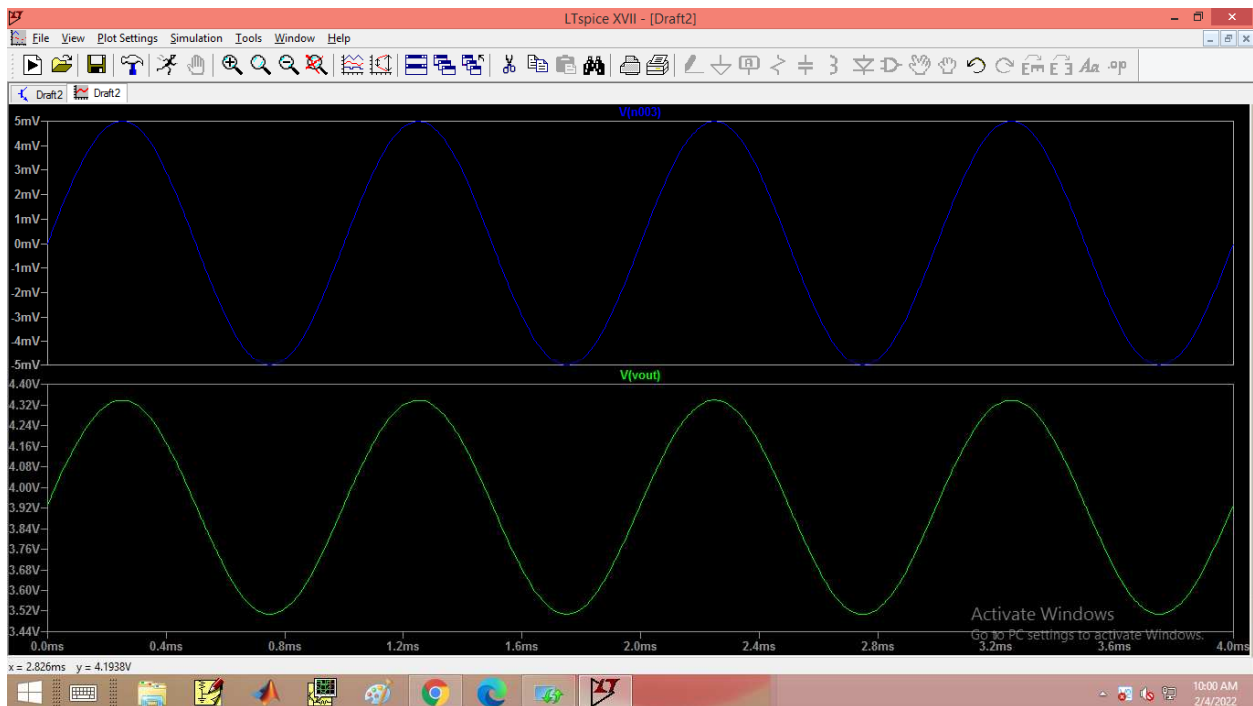
SIMULATION OUTPUT:



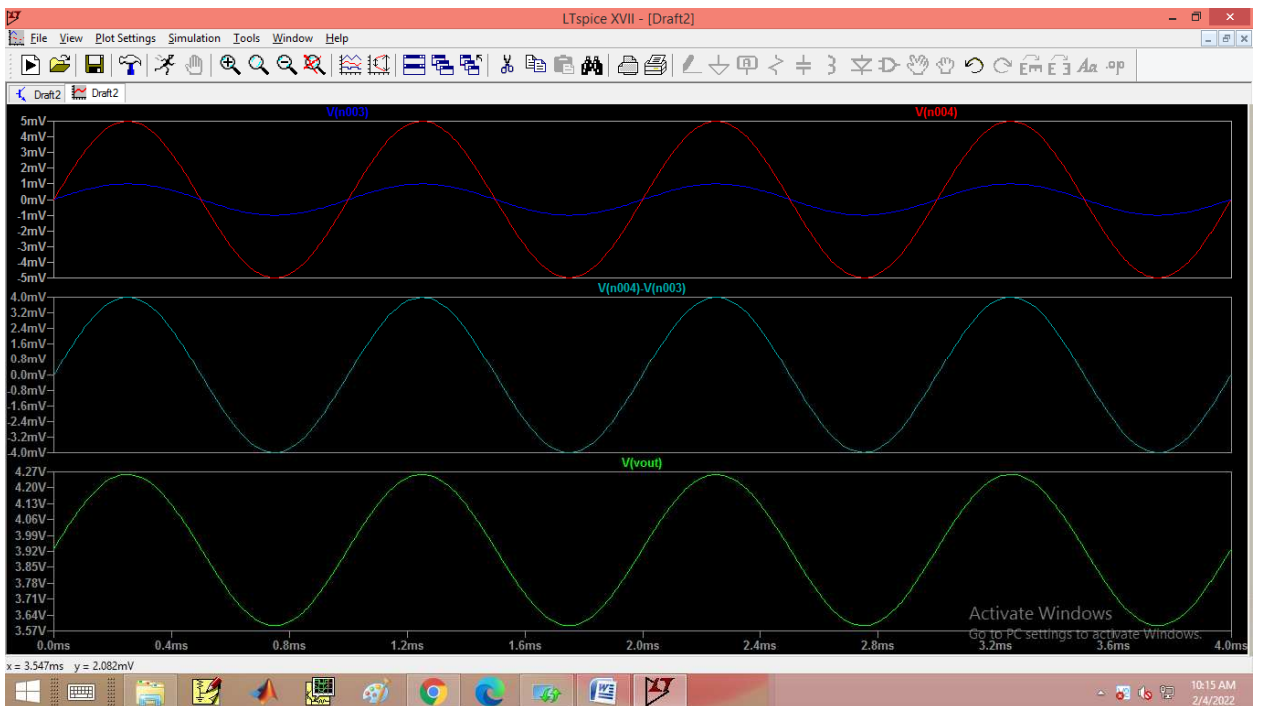
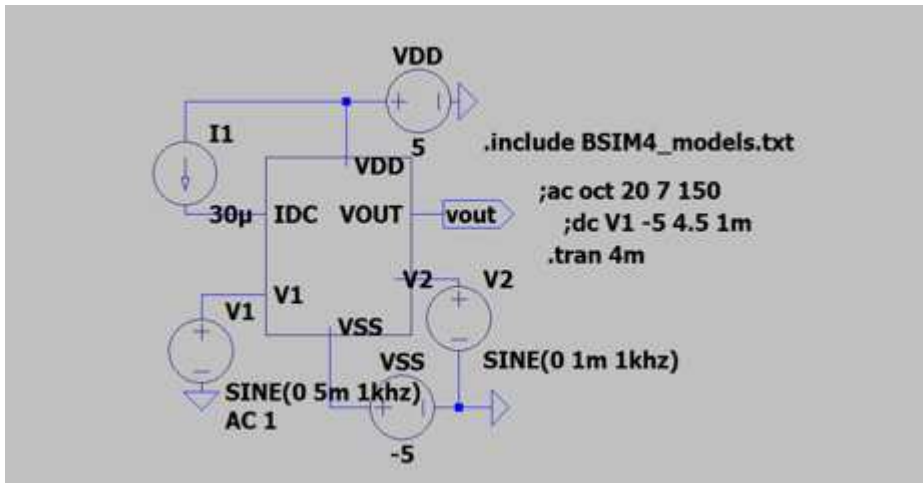
Transient Analysis:



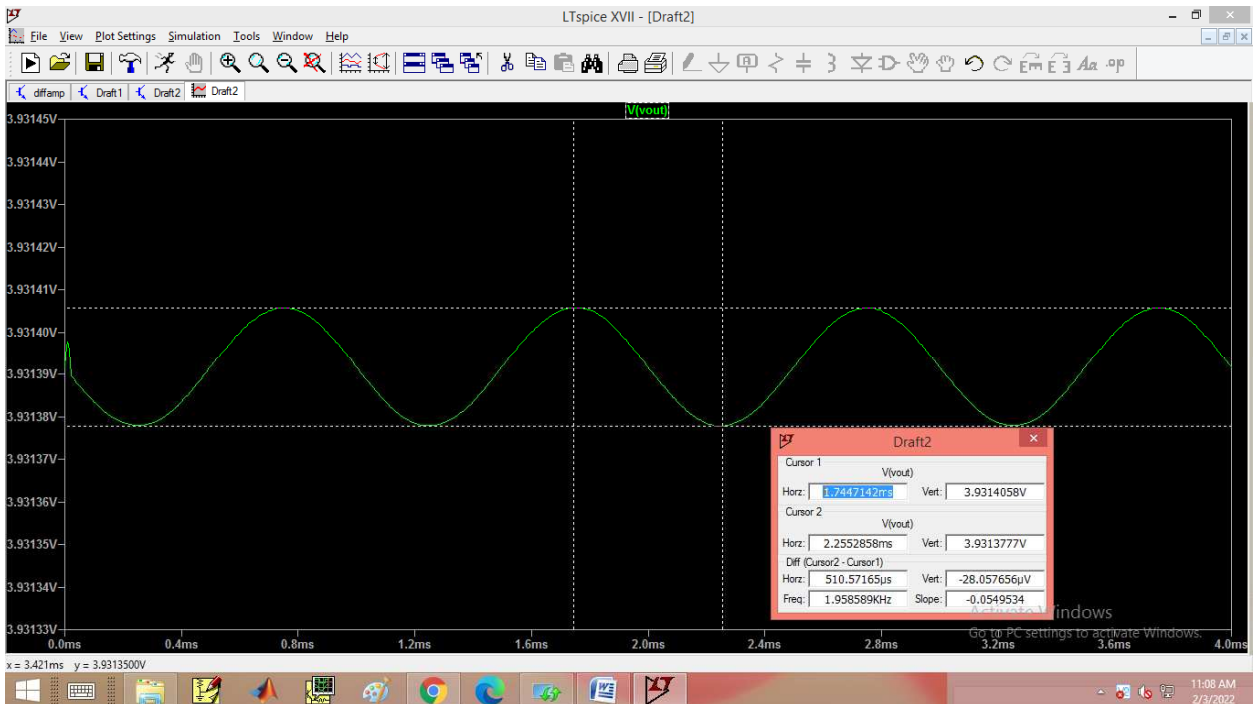
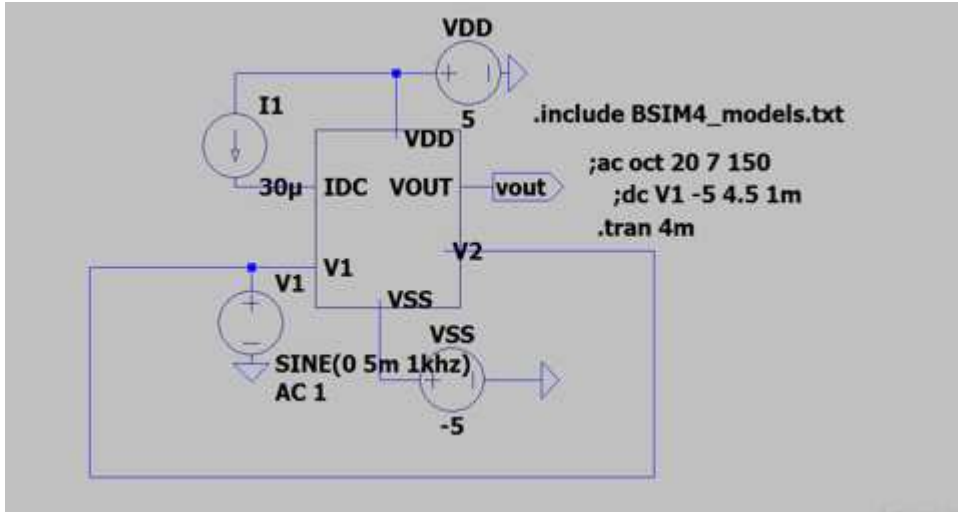
SIMULATION OUTPUT:



Differential Mode Gain:



Common Mode Gain:



Calculation:

$$CMRR = \frac{A_{DM}}{A_{CM}}$$

$$CMRR \text{ in dB} = 20 \log \frac{A_{DM}}{A_{CM}}$$

A_{DM} = Differential Voltage Gain

$$A_{DM} = \frac{V_{out}}{V1 - V2}$$

V1= 5mv V2= 1mv Vout =

A_{CM} = Common mode Voltage Gain

$$A_{CM} = \frac{V_{out}}{(V1 + V2)/2}$$

V1= 5mv V2= 5mv Vout =

RESULT:

Thus the Differential Amplifier is simulated and CMRR is determined using LT-SPICE.

EXP NO: 9

Date:

CMOS Inverter

AIM:

To design a CMOS inverter using the Schematic entry tool - Electric and verify its functioning.

APPARATUS REQUIRED:

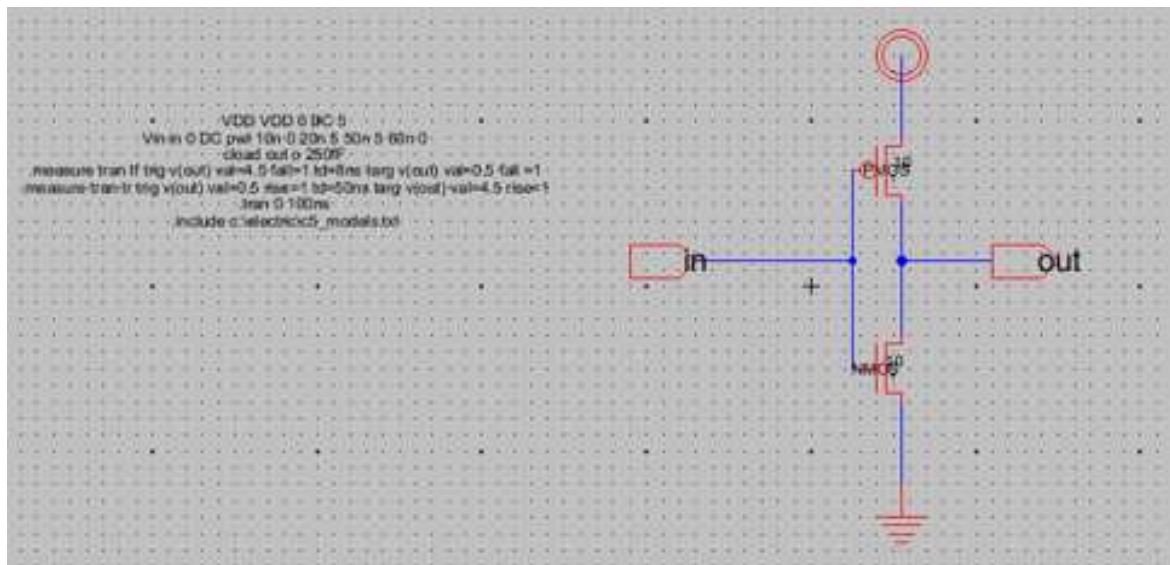
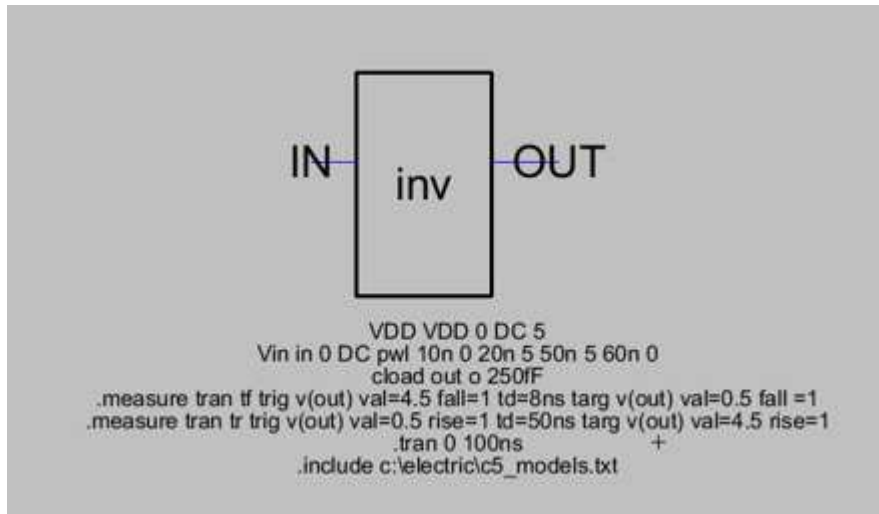
S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	ELECTRIC-EDA Tool	1

PROCEDURE:

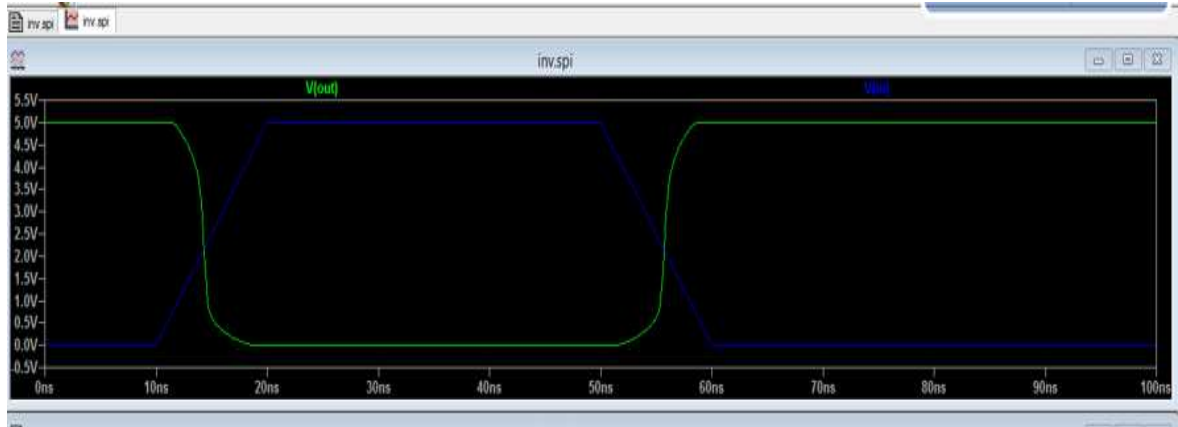
1. Start Electric VLSI system Design tool.
2. Start a new Project under the File -> New Schematic
3. Make sure, files are saved in a convenient directory. Save as filename.jelib under LIBRARIES name in Explorer
4. Go to Preferences by clicking the following button or executing File -> Preferences Then set the following.
 - a. Preferences -> Categories -> Technology -> Technology
5. Go to cell -> New Cell → Schematic
6. Go to **Components**. The schematic components will appear unlike the layout components in the startup window.
7. Connect the Circuit as shown in Figure
8. To check DRC ,execute Tools -> DRC -> Check Hierarchically
9. To check NCC, execute Tools -> NCC -> Schematic and Layout views of Cell in Current Window.
10. **For Well Check** execute Tools -> ERC -> Check Wells
11. Go to the **Components menu**. Click on the arrowhead in the **Misc box** to add **SPICE code** to the schematic . Place the SPICE code in the **schematic**
12. Go to Tools -> Simulation (Spice) -> Write Spice Deck.
13. Obtain the output waveform of CMOS Inverter.

SCHEMATIC

Symbol:



SIMULATION OUTPUT:



RESULT

Thus the design & simulation of a CMOS inverter has been carried out using schematic of Electric EDA Tools.

EXP NO: 10
Date:

Layout CMOS Inverter

AIM:

To draw the layout of CMOS Inverter using Electric EDA tool and extract the SPICE code.

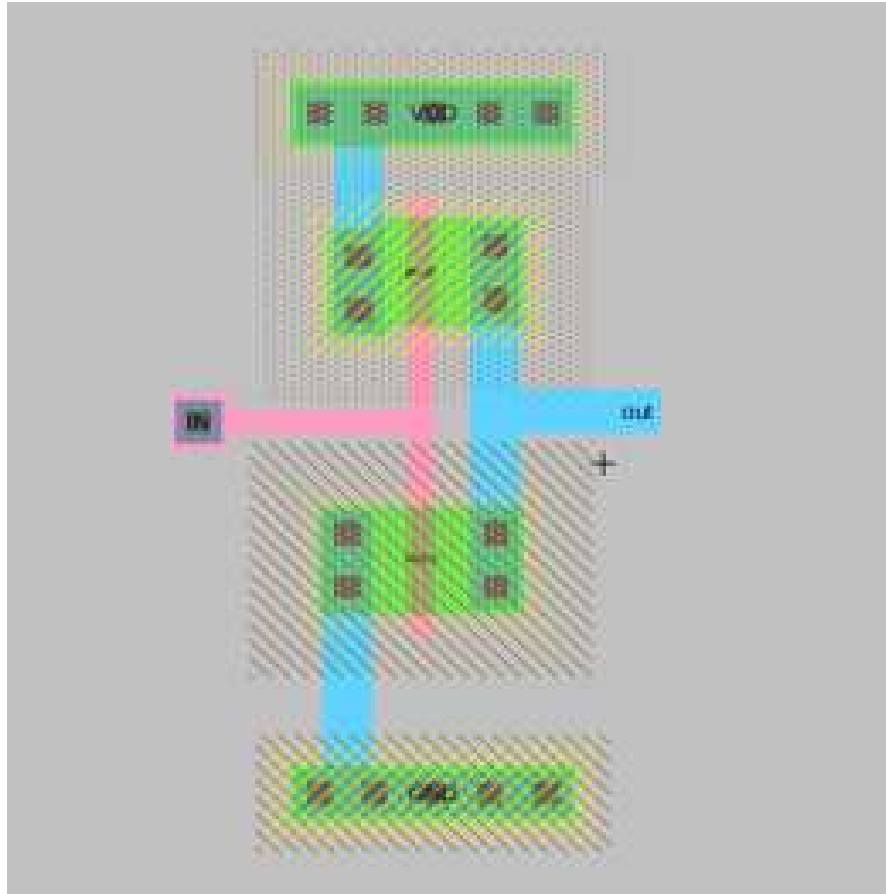
APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	ELECTRIC-EDA Tool	1

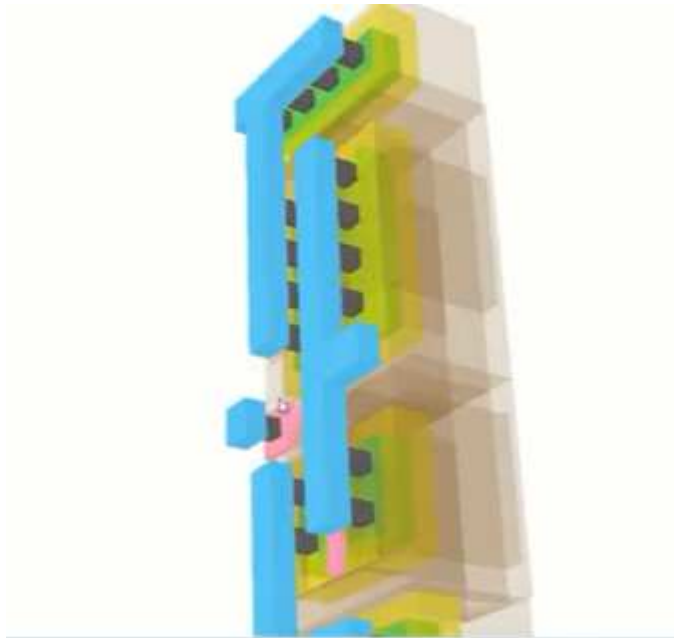
PROCEDURE:

1. Start Electric VLSI system Design tool.
2. Start a new Project under the File -> New Schematic
3. Make sure, files are saved in a convenient directory. Save as filename.jelib under LIBRARIES name in Explorer
4. Go to Preferences by clicking the following button or executing File -> Preferences Then set the following.
 - a. Preferences -> Categories -> Technology -> Technology
5. Go to cell -> New Cell → Schematic
6. Go to **Layers**. The Layer components will appear unlike the schematic components in the components window.
7. Connect the Layout as shown in Figure
8. To check DRC ,execute Tools -> DRC -> Check Hierarchically
9. To check NCC, execute Tools -> NCC -> Schematic and Layout views of Cell in Current Window.
10. **For Well Check** execute Tools -> ERC -> Check Wells
11. Go to the **Components menu**. Click on the arrowhead in the **Misc box** to add **SPICE code** to the schematic . Place the SPICE code in the **layout**
12. Go to Tools -> Simulation (Spice) -> Write Spice Deck.
13. Obtain the output waveform of CMOS Inverter.

LAYOUT:



3D VIEW:



SPICE Code:

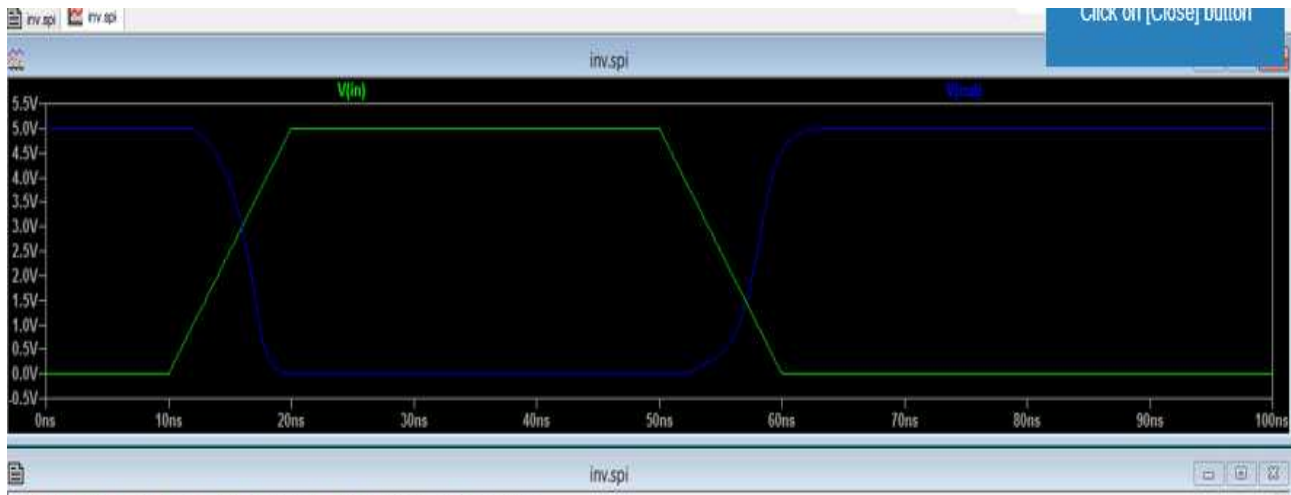
```
*** SUBCIRCUIT inverter1__inv FROM CELL inverter1:inv(lay)
.SUBCKT inverter1__inv gnd IN out vdd
Mnmos@0 gnd IN out gnd NMOS L=0.4U W=2U AS=3.2P AD=8.2P PS=7.2U PD=19.2U
Mpmos@0 vdd IN out vdd PMOS L=0.4U W=2U AS=3.2P AD=8.84P PS=7.2U PD=19.6U

* Spice Code nodes in cell cell 'inverter1:inv(lay)'
vdd vdd 0 DC 5
vin in 0 DC pw1 10n 0 20n 5 50n 5 60n 0
load out 0 250fF
.measure tran tf trig v(out) val=4.5 fall=1 td=8ns targ v(out) val=0.5 fall=1
.measure tran tr trig v(out) val=0.5 rise=1 td=50ns targ v(out) val=4.5 fall=1
.tran 0 100ns
.include c:\electric\c5_models.txt
.ENDS inverter1__inv

*** TOP LEVEL CELL: inv_sym(lay)
Xinv@0 gnd IN OUT vdd inverter1__inv

* Spice Code nodes in cell cell 'inv_sym(lay)'
vdd vdd 0 DC 5
vin in 0 DC pw1 10n 0 20n 5 50n 5 60n 0
load out 0 250fF
.measure tran tf trig v(out) val=4.5 fall=1 td=8ns targ v(out) val=0.5 fall=1
.measure tran tr trig v(out) val=0.5 rise=1 td=50ns targ v(out) val=4.5 fall=1
.tran 0 100ns
.include c:\electric\c5_models.txt
.END
```

SIMULATION OUTPUT:



RESULT:

Thus the layout of CMOS Inverter was verified through Electric EDA tool

EXP NO: 11
Date:

CMOS Inverter – Place and Route

AIM:

To design placement and routing, and post placement and routing parameters and observe logical effort for CMOS Inverter using Electric EDA tools.

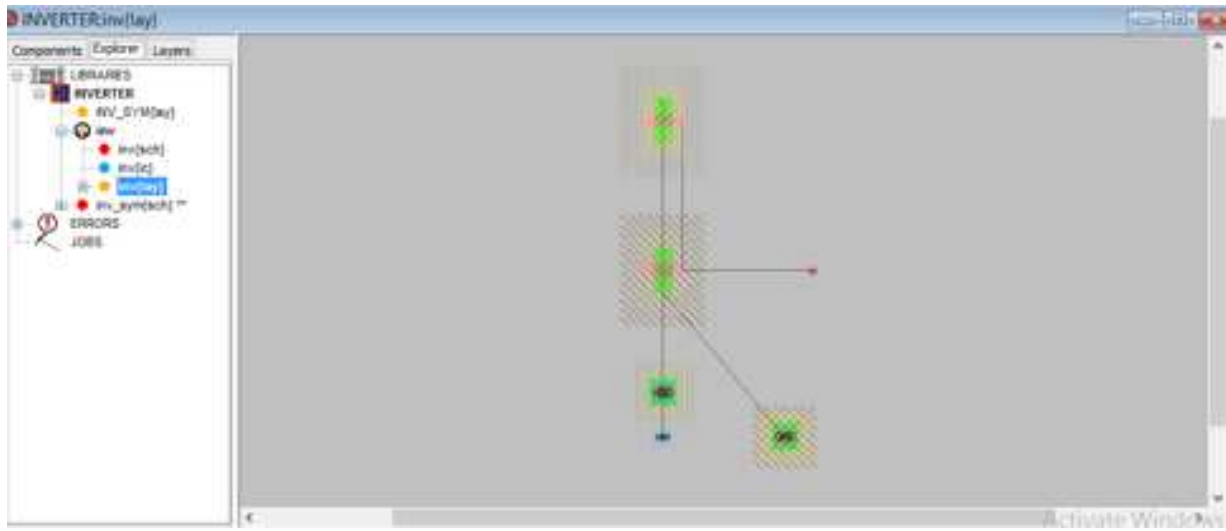
APPARATUS REQUIRED:

S.No	Name of the equipment/ software	Quantity
1.	PC with Windows	1
2.	ELECTRIC-EDA Tool	1

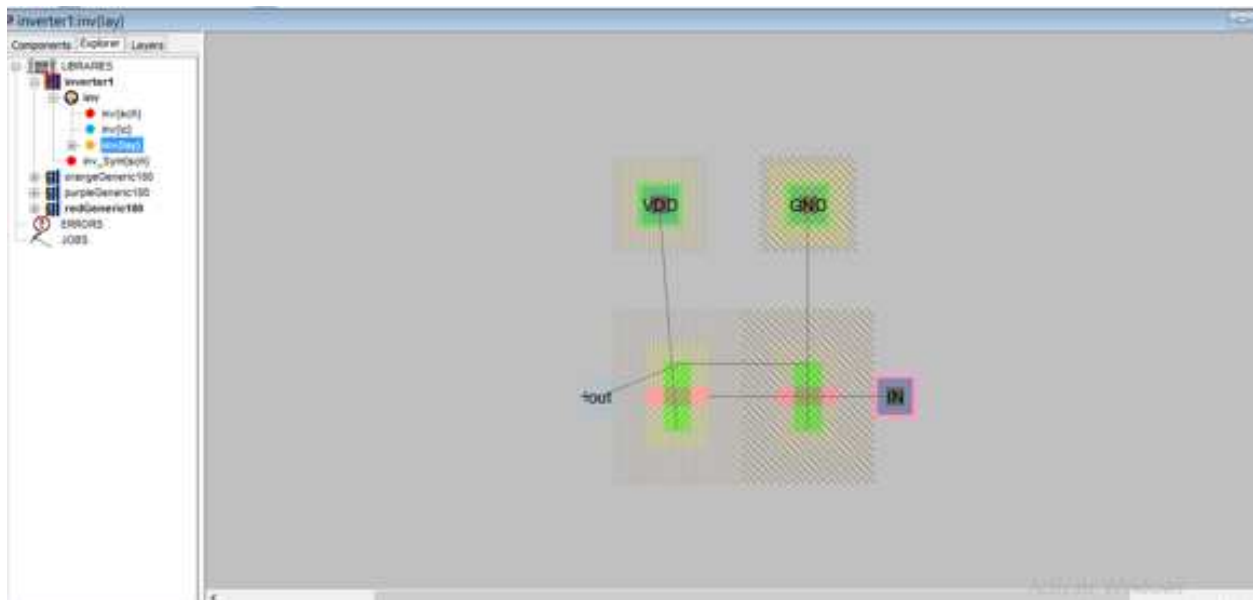
PROCEDURE:

1. Start Electric VLSI system Design tool.
2. Start a new Project under the File -> New Schematic
3. Make sure, files are saved in a convenient directory. Save as filename.jelib under LIBRARIES name in Explorer
4. Go to Preferences by clicking the following button or executing File -> Preferences Then set the following.
 - a. Preferences -> Categories -> Technology -> Technology
5. Go to cell -> New Cell -> Schematic
6. Go to **Components**. The schematic components will appear unlike the layout components in the startup window.
7. Connect the Circuit as shown in Figure
8. To check DRC ,execute Tools -> DRC -> Check Hierarchically
9. To check NCC, execute Tools -> NCC -> Schematic and Layout views of Cell in Current Window.
10. **For Well Check** execute Tools -> ERC -> Check Wells
11. Go to the **Components menu**. Click on the arrowhead in the **Misc box** to add **SPICE code** to the schematic . Place the SPICE code in the **schematic**
12. Go to Tools -> Simulation (Spice) -> Write Spice Deck.
13. Go to Tools -> Placement -> FloorPlanning to obtain place and route of Inerter
14. Go to Tools -> Logic effort -> Logic Effort libraries to obtain logic effort of Inverter.

Floor Planning:



Place and Route:

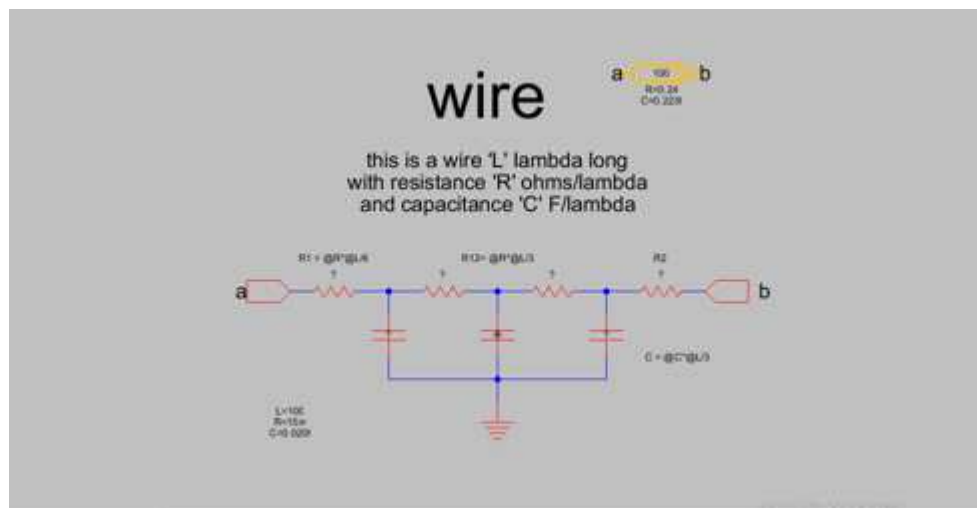


Report:

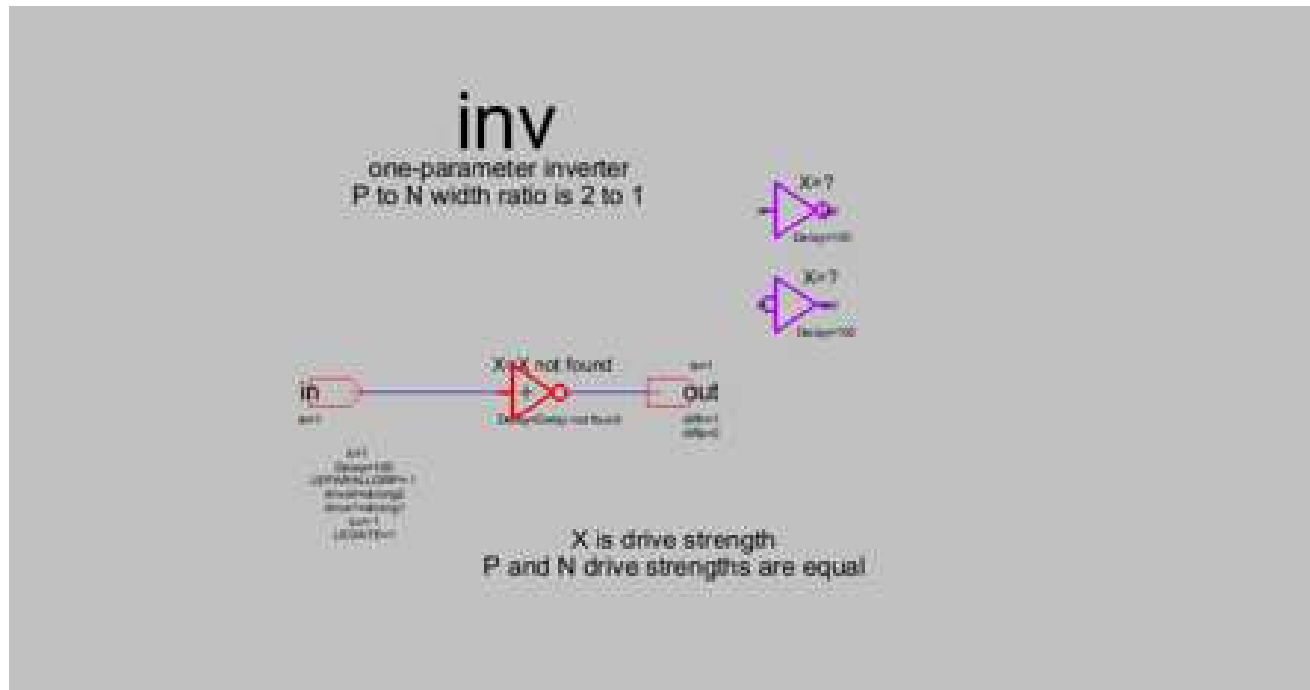
```
File Edit Cell Export View Window Tools Help
Electric Messages
-----
Electric's log file is C:\Users\ECE_CASDI\Desktop\electric.log.
-----
Library C:\Users\ECE_CASDI\Documents\inverter1.sp1ib read, took 0.015 secs
Checking library 'inverter1' for repair... library checked
No errors found
-----
Delay estimation for each network is a ratio of the following:
Numerator is the sum of these layers on the network:
  Transistor widths
  Diffusion half-perimeter, weighted by 0.7
  Poly silicon half-perimeter, weighted by 0.25
  Metal half-perimeter, weighted by 0.25
Denominator is the width of transistors on the network
Separate results are computed for N and P transistors, as well as their sum
-----
Network 'in' ratio computation:
N Gate width = 10.0, P Gate width = 10.0
Layer 1m half-perimeter is 0 x 0.25 = 0
Numerator is the sum of these factors (20)
N denominator = 10.0, ratio = 2
P denominator = 10.0, ratio = 2
N+P Denominator = 20.0, ratio = 1
-----
```

Activate Windows

RC MODEL:



LOGIC EFFORT:



RESULT:

Thus the post placement and routing parameters and logical effort of CMOS Inverter are observed using Electric EDA tools.

EXP NO: 12
Date:

Layout CMOS NAND Gate

AIM:

To draw the layout of CMOS NAND using Electric EDA tool and extract the SPICE code.

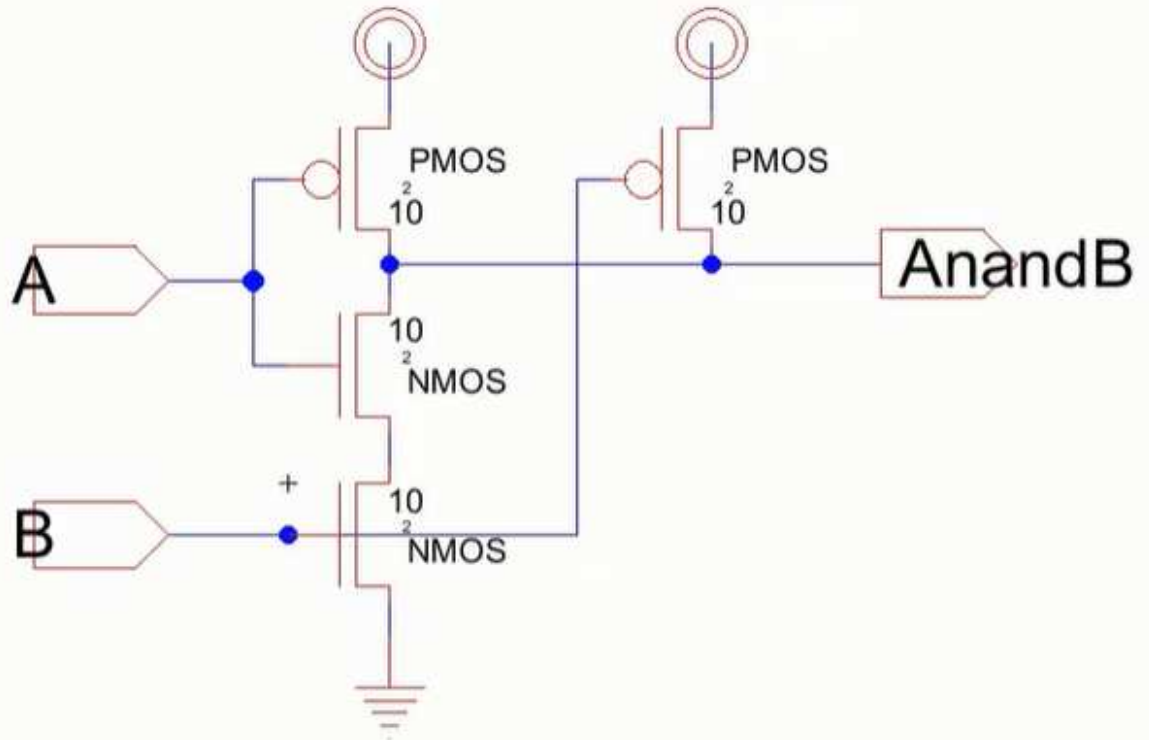
APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	ELECTRIC-EDA Tool	1

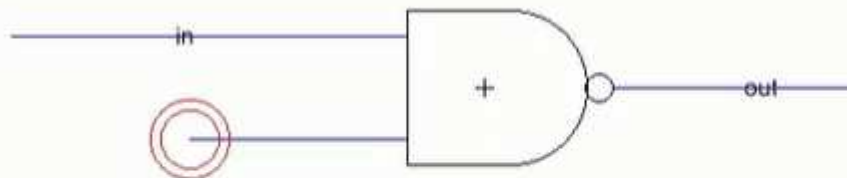
PROCEDURE:

1. Start Electric VLSI system Design tool.
2. Start a new Project under the File -> New Schematic
3. Make sure, files are saved in a convenient directory. Save as filename.jelib under LIBRARIES name in Explorer
4. Go to Preferences by clicking the following button or executing File -> Preferences Then set the following.
 - a. Preferences -> Categories -> Technology -> Technology
5. Go to cell -> New Cell → Schematic
6. Go to **Layers**. The Layer components will appear unlike the schematic components in the components window.
7. Connect the Layout as shown in Figure
8. To check DRC ,execute Tools -> DRC -> Check Hierarchically
9. To check NCC, execute Tools -> NCC -> Schematic and Layout views of Cell in Current Window.
10. **For Well Check** execute Tools -> ERC -> Check Wells
11. Go to the **Components menu**. Click on the arrowhead in the **Misc box** to add **SPICE code** to the schematic . Place the SPICE code in the **layout**
12. Go to Tools -> Simulation (Spice) -> Write Spice Deck.
13. Obtain the output waveform of CMOS NAND.

SCHEMATIC:

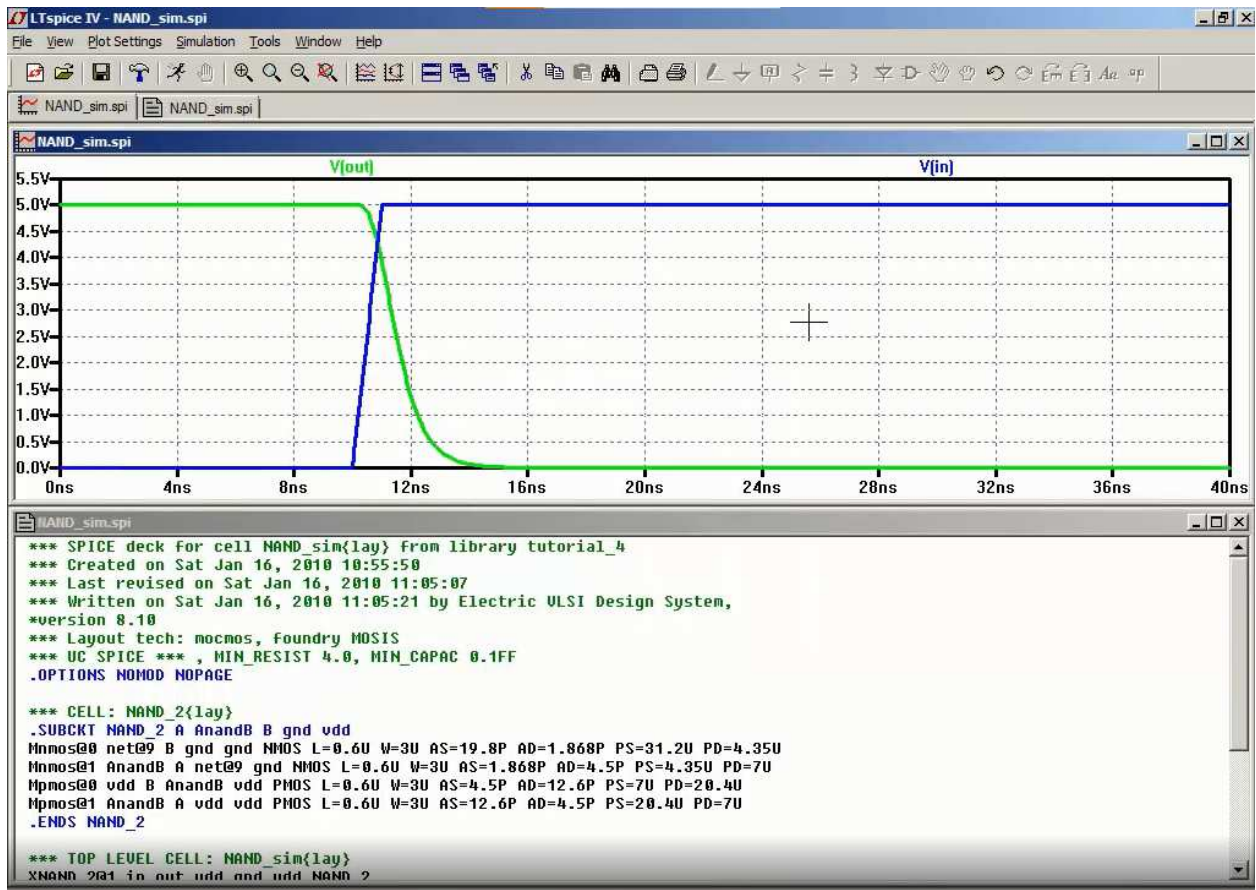


SYMBOL:



```
vdd vdd 0 dc 5
vin in 0 dc pulse 0 5 10n 1n
cload out 0 250fF
.tran 0 40n
.include C:\Electric\C5_models.txt
```


Simulation Output:



RESULT:

Thus the layout of CMOS NAND was verified through Electric EDA tool

EXP NO: 13
Date:

Static Timing Analysis

AIM:

To study the given circuit and perform static timing analysis using Synopsys - PrimeTime STA tools.

APPARATUS REQUIRED:

S.No	Name of the equipment/ software	Quantity
1.	PC with Windows	1
2.	Synopsys -PrimeTime STA tool	1

PROCEDURE:

Invoke PrimeTime STA tool

To invoke PrimeTime, choose either options

pt_shell(command mode)

primitime&(GUI mode)

Command mode is preferred because:

- The command mode helps you to keep a record of what you have done.
- The command mode runs more efficiently than GUI mode.
- The command mode helps you to lookup the manual/reference quickly.

In spite of the above advantages, command mode sometimes is not as good as GUI mode in terms of debugging the schematic problem.

Start Operating PrimeTime STA tool

STA Environment Setting for TSMC 0.13um Technology:

- Set search path (*If it has not been set up yet*)

```
setsearch_path "/home/raid2_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db"
```

- Set link library

```
setlink_path "* typical.dbfast.dbslow.db"
```

```
~/Desktop/synopsys_example/DesignCompiler/dc
~/Desktop/synopsys_example/DesignCompiler/dc$ pt_shell

PrimeTime (R)
Version J-2014.12-SP2 for RHEL64 -- Feb 26, 2015
Copyright (c) 1988-2015 Synopsys, Inc.
ALL RIGHTS RESERVED

This program is proprietary and confidential information of Synopsys, Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

pt_shell> set link
link_allow_design_mismatch link_library
link_create_black_boxes link_path
link_force_case link_path_per_instance
pt_shell> set link_library {* re
ref/ results/
pt_shell> set link_library {* ref/models/saed
saed90nm_typ_ht.db saed90nm_typ_ht.lib
pt_shell> set link_library {* ref/models/saed90nm_typ_ht.db }
```

Read Gate level Netlist Files and Link design:

1. Type these lines to read in CIC .18 library and your gate level netlist.

```
read_verilog ./Counter_syn.v
```

2. Link all designs

```
link_design Counter
```

Note, to check the search path and include library, if the error message occurred after step 2.

Read Timing and RC information:

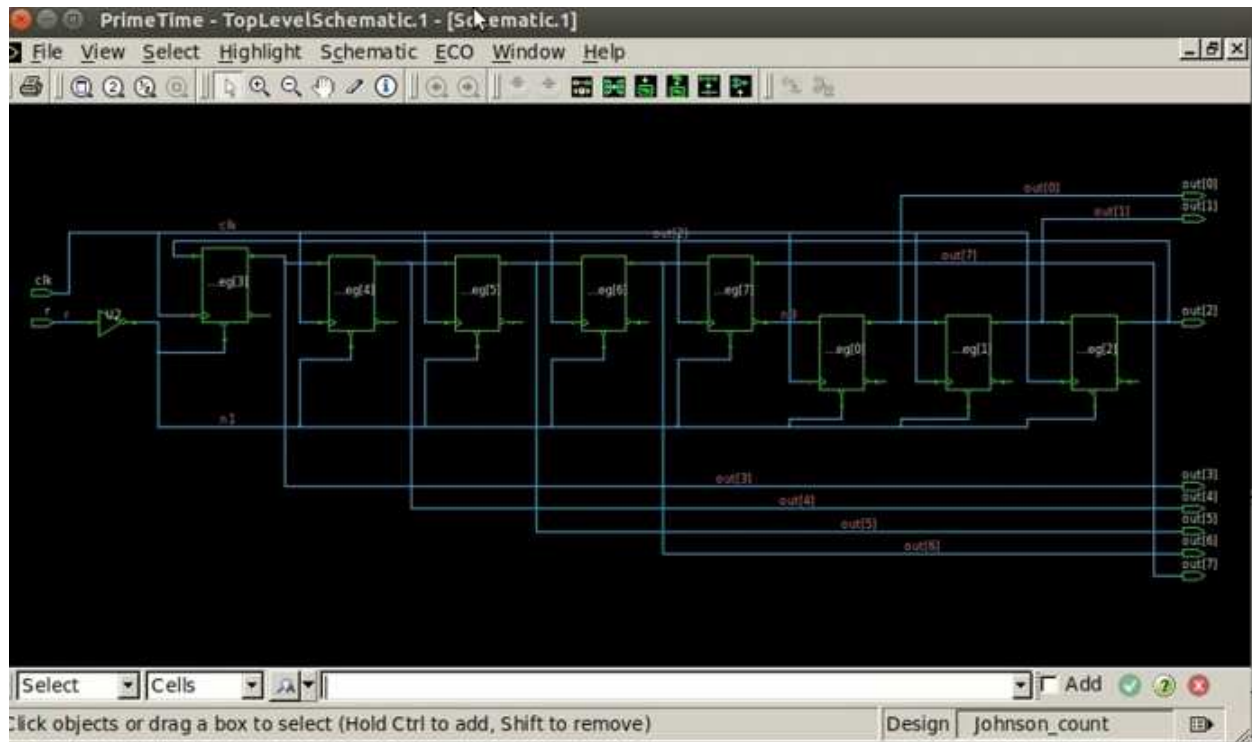
Reads leaf cell and net timing and RC information from a file in SPEF Format and uses that information to annotate the current design.

```
read_parasitics counter.spf
```

Note: The file can be get during synthesis with “write_parasitics” comment.

Set Operating Conditions:

```
set_operating_conditions typical -library typical
```



Set Design Constraints:

This step tells the Dft Compiler how many scan chains are needed. specify the names of scan related pins (scan_enable, scan_in, scan_out).

1. Specify the clock name, period, and clock characteristic


```
create_clock -period 10 -waveform {0 5} [get_portsclk]
setdesign_clock [get_clockclk]
set_clock_uncertainty 0.5 $design_clock
set_clock_latency -min 1.5 $design_clock
set_clock_latency -max 2.5 $design_clock
set_clock_transition -min 0.25 $design_clock
set_clock_transition -max 0.30 $design_clock
set_propagated_clock $design_clock
```

2. Set wire load model


```
set_wire_load_model -name "ForQA" -library "typical"
```

3. Set wire load mode


```
set_wire_load_mode top
```

4. Report


```
report_design
report_reference
```

```

~/Desktop/synopsys_example/DesignCompiler/dc
clock clock (rise edge)          70.000    70.000
clock network delay (ideal)     0.000    70.000
out_reg[7]/CLK (DFFARX1)        0.000    70.000 r
out_reg[7]/QN (DFFARX1)        0.224    70.224 r
U1/Z (AOBUFX2)                  0.129    70.353 r
out_reg[0]/D (DFFARX1)         0.032    70.385 r
data arrival time                70.385

clock clock (rise edge)          190.000   190.000
clock network delay (ideal)     0.000   190.000
clock reconvergence pessimism   0.000   190.000
out_reg[0]/CLK (DFFARX1)       190.000 r
library setup time              -0.161   189.839
data required time              189.839
-----
data required time              189.839
data arrival time               -70.385
-----
slack (MET)                      119.454

pt_shell> write_changes -format text -output eco
1
pt_shell> pt_shell>

```

Timing analysis and report possible problems:

This step checks your scan specification for consistency. Please type the following commands to set the input/output delay:

```

set_input_delay 1.5 [get_ports inputA] -clock $design_clock
set_input_delay 1.5 [get_ports inputB] -clock $design_clock
set_input_delay 1.5 [get_ports instruction] -clock $design_clock
set_input_delay 1.5 [get_ports reset] -clock $design_clock
set_output_delay 1.5 [get_ports alu_out] -clock $design_clock

```

And then check the timing:

```

check_timing
set_true_delay_prove_true_backtrack_limit 20000
report_timing -true
report_bottleneck

```

RESULT:

Thus the static timing analysis of the given circuit has been studied.

EXP NO: 14
Date:

DfT – Scan Chain Insertion

AIM:

To study the given circuit and perform DfT-Scan chain insertion using Synopsys - TetraMaxtools.

APPARATUS REQUIRED:

S.No	Nameofthe equipment/ software	Quantity
1.	PC with Windows	1
2.	Synopsys - TetraMax tool	1

PROCEDURE:

Invoke DftCompiler

Dft Compiler is actually embedded in the Design Compiler.

To invoke Dft Compiler, choose either options.

dc_shell(command mode)

dv&(GUI mode)

Command mode is preferred because:

- a. Command mode helps you to keep a record of what you have done.
- b. Command mode runs more efficiently than GUI mode.
- c. Command mode helps you to lookup the manual/reference quickly.

In spite of the above advantages, command mode sometimes is not as good as GUI mode in terms of debugging the schematic problem.

NOTE: maybe occurrence of some error message like “Error: current design notdefined.” just ignore it for now.

STEP 1: Read Input Files

1. Please check there is no error message when starting the “dc_shell”. If there are errors in the windows, please check the .synopsys_dc.setup. Type either one of these lines to read your gate level netlist (The circuit after synthesis).

```
read_verilogfilename.v  
read_filefilename.v -format Verilog
```

2. Set the working design to you top design. In this case, set ALU as the working design.
current_design ALU

3. Resolve the design references and check if there is any errors.
Link check_design

4. Set the design constraints and check if the designs have any violations. The constraints.tcl is based on the constraints that you used in the synthesis lab.

```
sourceconstraints.tcl  
report_constraint -all_violators
```

5. To obtain a timing/area/power report of your original design, type (where ALU is your top design)

```
report_area>filename.area_rpt  
report_timing>filename.timing_rpt  
report_power>filename.power_rpt
```

STEP 2: Select scan style

Define the default scan style for the insert_dft command if a scanstyle is not specified with the set_scan_style command. This variable must identify one of the following supported scan styles: multiplexed_flip_flop, clocked_scan, lssd, aux_clock_lssd, combinational, or none. You can skip this step because the default is multiplexed_flip_flop.

```
settest_default_scan_style multiplexed_flip_flop
```

STEP 3: Set ATE configuration and create test protocol

The timing of the test clock is based on the test_default_period, test_default_delay, test_default_strobe, and test_default_strobe_width variables.

```
settest_default_delay 0  
settest_default_bidir_delay 0  
settest_default_strobe 40  
settest_default_period 100
```

To create a test protocol for a non-scan design, you can just type

```
create_test_protocol -infer_asynch -infer_clock
```

When -infer_asynch is specified, create_test_protocol infers asynchronous set and reset signals in the design, and places them at offstate during scan shifting. When -infer_clock is specified, create_test_protocol infers test clock pins from the design, and places them during scan shifting.

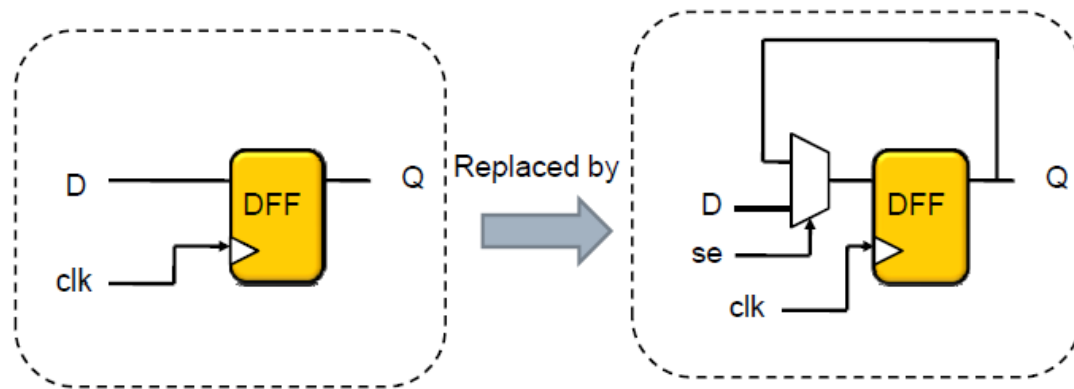
STEP 4: Pre-scan Check

Check if there is any design constraint violations before scan insertion.

report_constraint -all_violators

Perform pre-scan test design rule checking.

dft_drc



STEP 5: Scan specification

This step tells the Dft Compiler how many scan chains needed. This allows to specify the names of scan related pins (scan_enable, scan_in,scan_out).

set_scan_configuration -chain_count 1

STEP 6: Scan preview

This step checks your scan specification for consistency. Please type

preview_dft

STEP 7: scan chain synthesis

Stitch your scan cells into a chain. And do some more optimizations.

insert_dft

STEP8: Post-scan check

Check if there is any design constraint violations after scan insertion.

report_constraint -all_violators

Perform post-scan test design rule checking.

dft_drc

```
Resetting current test mode
Beginning Mapping Optimizations
-----
In mode: Internal_scan...
Design has scan chains in this mode
Design is scan routed
Post-DFT DRC enabled

Information: Starting test design rule checking. [TRST-222]
Loading test protocol
...basic checks...
...basic sequential cell checks...
...checking vector rules...
...checking clock rules...
...checking scan chain rules...
...checking scan compression rules...
...checking X-state rules...
...checking tristate rules...
...extracting scan details...
...saving simulation value info...
-----
Begin Modeling violations...
```

```
Pattern Summary Report
-----
#internal patterns                                0
-----

Uncollapsed Stuck Fault Summary Report
-----
fault class          code    #faults
-----
Detected              DT      2164
Possibly detected     PT        0
Undetectable          UD       90
ATPG untestable       AU      150
Not detected           ND        0
-----
total faults                2404
test coverage                93.52%
-----
Information: The test coverage above may be inferior
than the real test coverage with customized
protocol and test simulation library.
```


STEP 9: Reports

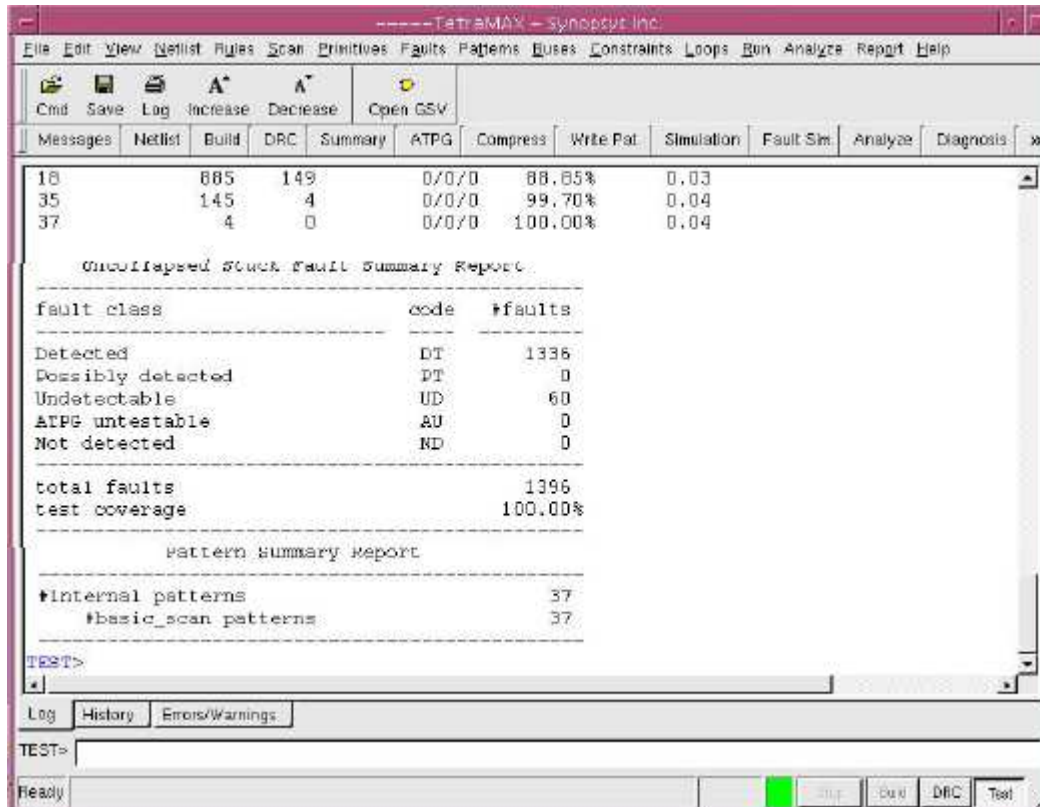
Report the scan cells and the scan paths

```
report_scan_path -view existing -chain all >ALU_syn_dft.scan_path
```

```
report_scan_path -view existing -cell all >ALU_syn_dft.scan_cell
```

To obtain a timing/area report of your scan_inserted design, type

```
report_timing>ALU_syn_dft.timing_rpt
```



The screenshot shows the TetraMAX software interface. The main window displays a 'Collapsed Stack Fault Summary Report' with the following data:

Messages	Netlist	Build	DRC	Summary	ATPG	Compress	Write Pat.	Simulation	Fault Sim.	Analyze	Diagnosis
18	885	149		0/0/0	88.85%	0.03					
35	145	4		0/0/0	99.70%	0.04					
37	4	0		0/0/0	100.00%	0.04					

Collapsed Stack Fault Summary Report

fault class	code	#faults
Detected	DT	1336
Possibly detected	PT	0
Undetectable	UD	60
ATPG untestable	AU	0
Not detected	ND	0

total faults 1396
test coverage 100.00%

Pattern Summary Report

#internal patterns	37
#basic_scan patterns	37

TEST>

Log History Errors/Warnings

Ready

Buttons: Stop, Run, DfC, Test

RESULT:

Thus the DfT-Scan chain insertion of the given circuit has been studied.