

ST.ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai)

(An ISO 9001: 2015 Certified Institution)

ANGUCHETTYPALAYAM, PANRUTI – 607 106.



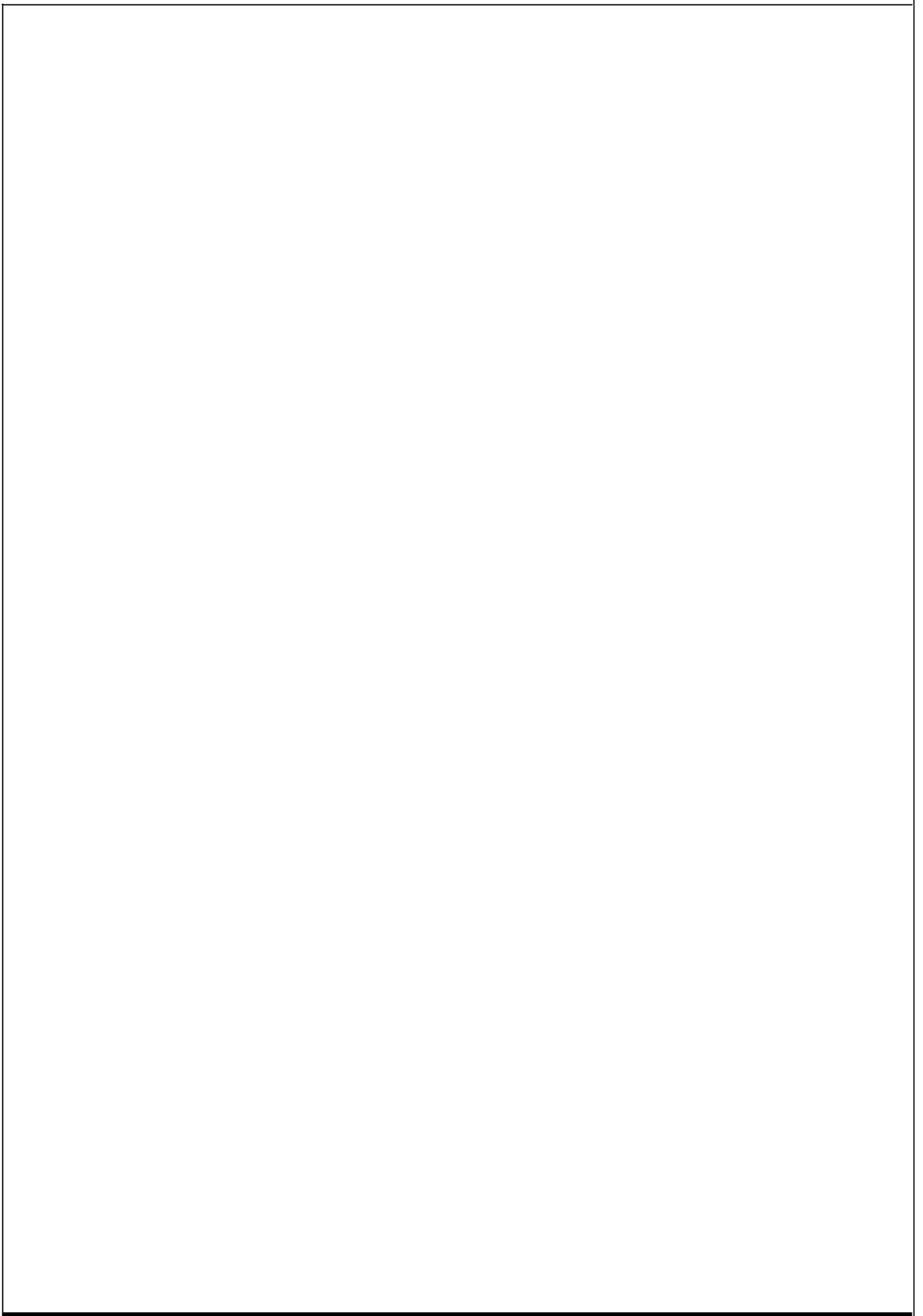
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

SUBJECT CODE/NAME: EC 8661/VLSI DESIGN LABORATORY

YEAR/SEM: III/VI

AS PER ANNA UNIVERSITY, CHENNAI REGULATION 2017



OBJECTIVES:**The student should be made:**

- To learn Hardware Descriptive Language(Verilog/VHDL)
- To learn the fundamental principles of VLSI circuit design in digital and analog domain
- To familiarize fusing of logical modules on FPGAs
- To provide hands on design experience with professional design (EDA) platforms

LIST OF EXPERIMENTS:**Part I: Digital System Design using HDL & FPGA (24 Periods)**

1. Design an Adder (Min 8 Bit) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
2. Design a Multiplier (4 Bit Min) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
3. Design an ALU using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
4. Design a Universal Shift Register using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
5. Design Finite State Machine (Moore/Mealy) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
6. Design Memories using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA

Compare pre synthesis and post synthesis simulation for experiments 1 to 6.

Requirements: Xilinx ISE/Altera Quartus/ equivalent EDA Tools along with Xilinx/Altera/equivalent FPGA Boards

Part-II Digital Circuit Design (24 Periods)

7. Design and simulate a CMOS inverter using digital flow
 8. Design and simulate a CMOS Basic Gates & Flip-Flops
 9. Design and simulate a 4-bit synchronous counter using a Flip-Flops
- Manual/Automatic Layout Generation and Post Layout Extraction for experiments 7 to 9

Part-III Analog Circuit Design (12 Periods)

10. Design and Simulate a CMOS Inverting Amplifier.
11. Design and Simulate basic Common Source, Common Gate and Common Drain Amplifiers.
Analyze the input impedance, output impedance, gain and bandwidth for experiments 10 and 11 by performing Schematic Simulations.
12. Design and simulate simple 5 transistor differential amplifier. Analyze Gain, Bandwidth and CMRR by performing Schematic Simulations.

Requirements: Cadence/Synopsis/ Mentor Graphics/Tanner/equivalent EDA Tools

TOTAL: 60 PERIODS

LAB PLAN

LIST OF EXPERIMENTS

CYCLE - I

1. Design an Adder (Min 8 Bit) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
2. Design a Multiplier (4 Bit Min) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
3. Design an ALU using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
4. Design a Universal Shift Register using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
5. Design Finite State Machine (Moore/Mealy) using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA
6. Design Memories using HDL. Simulate it using Xilinx/Altera Software and implement by Xilinx/Altera FPGA

CYCLE - II

7. Design and simulate a CMOS inverter using digital flow
8. Design and simulate a CMOS Basic Gates & Flip-Flops
9. Design and simulate a 4-bit synchronous counter using a Flip-Flops
10. Design and Simulate a CMOS Inverting Amplifier.
11. Design and Simulate basic Common Source, Common Gate and Common Drain Amplifiers.
12. Design and simulate simple 5 transistor differential amplifier. Analyze Gain, Bandwidth and CMRR by performing Schematic Simulations.

Requirements: Cadence/Synopsis/ Mentor Graphics/Tanner/equivalent EDA Tools

XILINX

TOOLS

EXP.NO:	STUDY OF SIMULATION TOOLS
DATE:	

AIM:

To study simulation tools using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

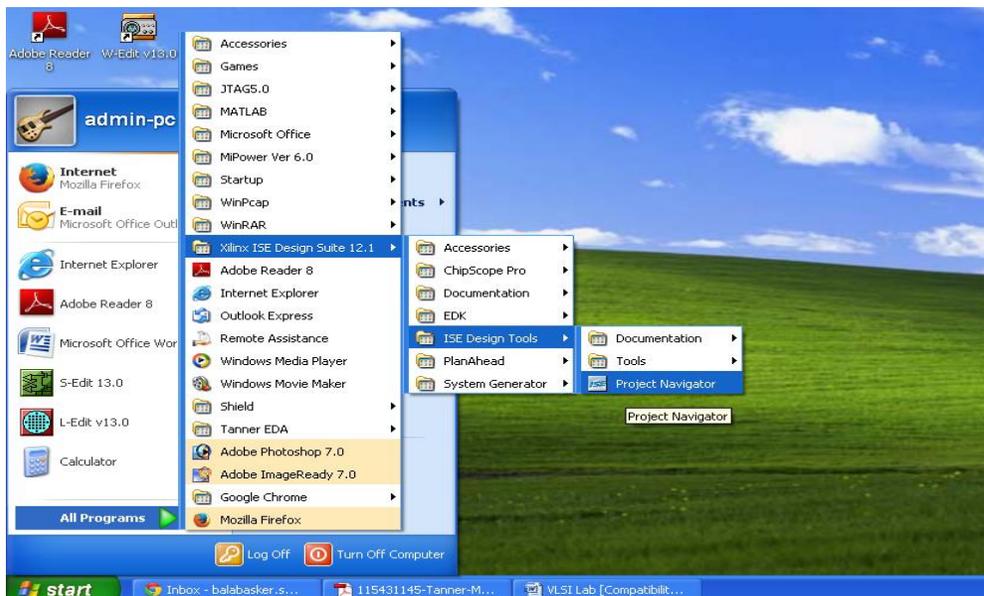
PROCEDURE:

1. Now start the Xilinx ISE Design Suite 12.1
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next and click Finish.
5. Click new source.
6. Select verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally the report is shown click finish.
9. Type the program save and click synthesis.
10. To see the output wave form change the source from implementation to simulation and click simulator behavior model in ISim simulator.
11. Give values to the input variables and then click run
12. In wave window, click run icon and you can see corresponding output.

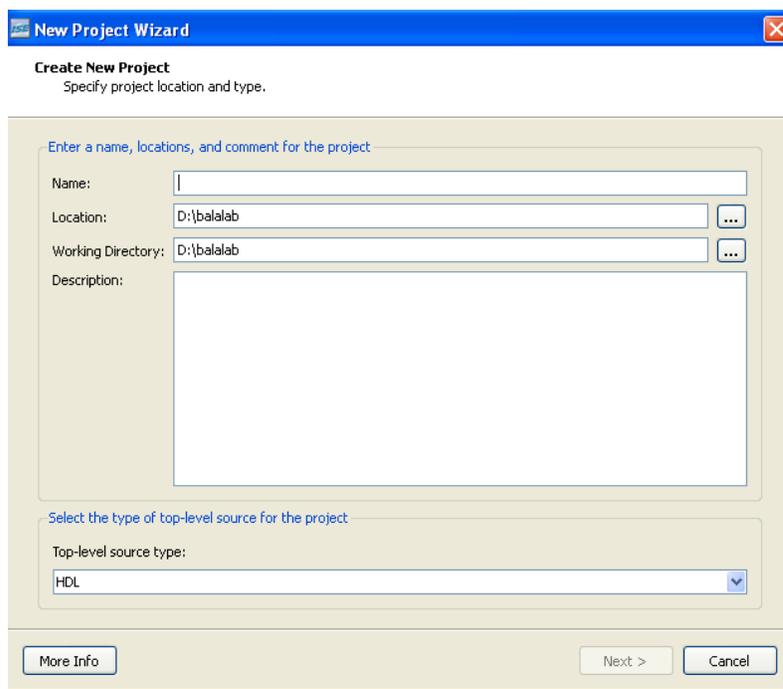
Steps to use Xilinx tool:

Start the Xilinx Project Navigator by using the desktop shortcut or by using the

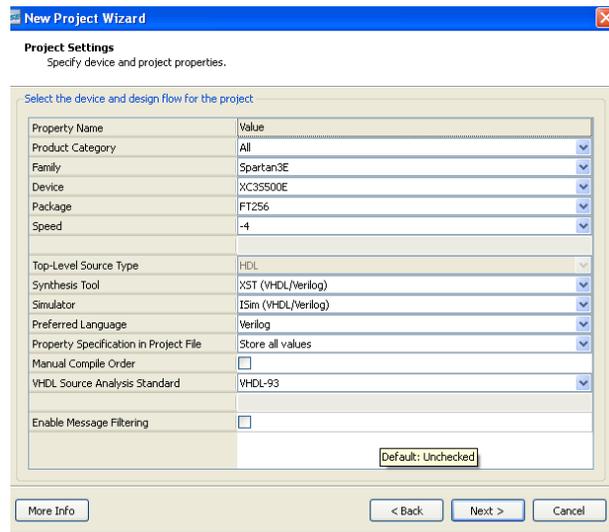
- Start → Programs → Xilinx ISE → Project Navigator.



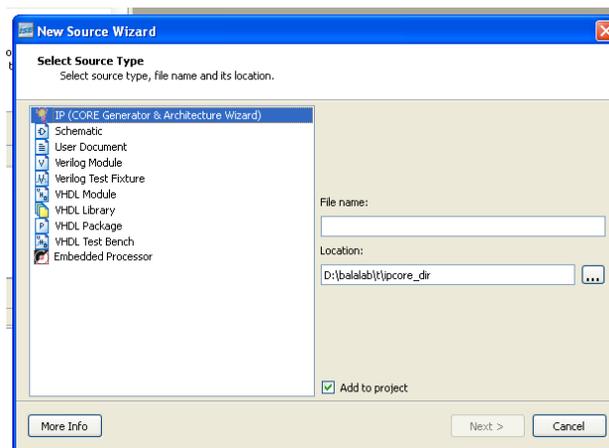
- In the Project Navigator window go to FILE → New project → Click on new source → verilog module and give the name inverter.v → Define ports → Finish



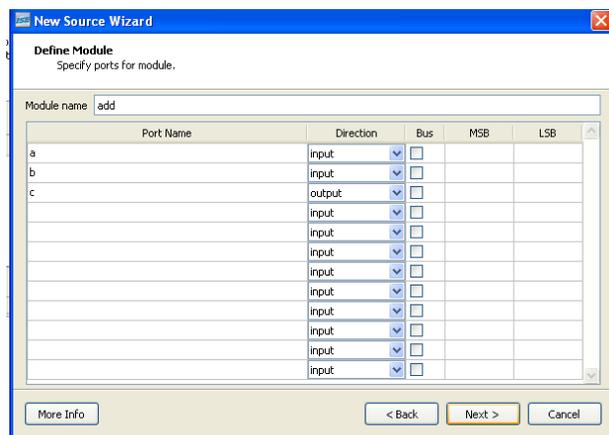
- Select devices → General purpose → Spartan 3 → ISE simulator → verilog



➤ In the create new source window select source type as verilog module give file name



➤ assign inputs and outputs → click next → finish → yes → next → next → finish



➤ Double click on source file → complete the verilog code for inverter

- Check syntax, and remove errors if present
- Simulate the design using ISE Simulator Highlight **inverter.v** file in the Sources in Project window. To run the Behavioral Simulation, Click on the symbol of FPGA device and then right click → Click on new source → Click on verilog text fixture → Give file name with _tb → finish
- Generate test bench file after initial begin assign value for inputs → Click on simulate behavioral model → see the output.

RESULT:

EXP.NO:	STUDY OF SYNTHESIZE TOOLS
DATE:	

AIM:

To study synthesise tools using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:

Synthesis is an automatic method of converting a higher level abstraction to a lower level abstraction. The synthesis tool convert **Register Transfer Level (RTL)** description to **gate level netlists**. These gate level netlists consist of interconnected gate level macro cells. These gate level netlists currently can be optimized for **area, speed** etc., The analyzed design is synthesized to a library of components, typically gates, latches, or flipflops. Hierarchical designs are synthesized in bottom up fashion, that is lower level components are synthesized before higher level components. Once the design is synthesized we have a gate level netlist. This gate level netlist can be simulated. Delay for the individual components are available as part of the description of the component libraries. Timing accurate simulation is not possible at this point because the actual timing characteristics is determined by the physical placement of the design within the FPGA chip. However, the functional simulation that is possible at this point is quite a bit more accurate than simulation based on user specified delays. After run the synthesise in process window then full adder model is converted to **netlist file**.

To convert the RTL to gates, three steps typically occur:

* The RTL description is translated to an unoptimized boolean description usually consisting of primitive gates such as AND and OR gates, flip-flop, and latches. This is a functionally correct but completely unoptimized description.

* Boolean optimization algorithms are executed on this boolean equivalent description to produce an optimized boolean equivalent description.

* This optimized boolean equivalent description is mapped to actual logic gate by making use of a technology library of the target process.

PROCEDURE:

1. Now start the Xilinx ISE Design Suite 12.1
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next and click Finish.
5. Click new source.
6. Select verilog module and type file name and click next.

7. Assign input and output port and click next.
8. Finally the report is shown click finish.
9. Type the program save and click synthesis.
10. Go to synthesis → View RTL schematic

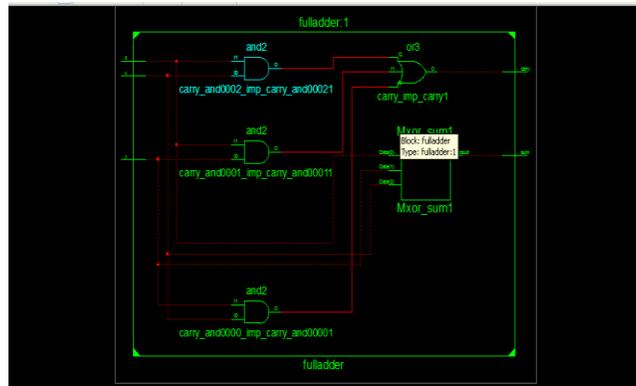
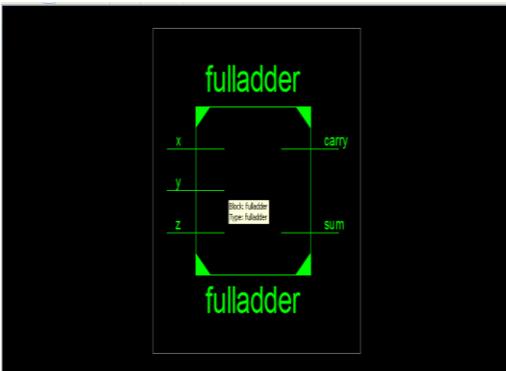
PROGRAM:

```

module full_adder (a,b,c,sum,carry);
output sum,carry ;
input a,b,c ;
assign sum = a ^ b ^ c;
assign carry = (a&b) | (b&c) | (c&a);
endmodule

```

RTL SCHEMATIC:



RESULT:

EXP.NO:	PLACE AND ROUTE AND BACK ANNOTATION FOR FPGAS
DATE:	

AIM:

To study place and route and back annotation for FPGAs synthesize tools using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:

To map this Full adder design onto the FPGA. The primitive hardware elements that are available in Xilinx xc3s500e chip, namely lookup tables and positive-edge-triggered flip-flops are organized as a **two dimensional array of CLBs**. The net list from synthesize is composed of **gates, latches, and flip-flops**. It is necessary to assign CLB to net list primitives. This is the process of **mapping** a design. For example gates will be assigned to look-up tables. This process effectively translates the gate level netlist produce by the synthesize compiler into a netlist of FPGA primitive hardware components. Each elements of this new netlist corresponds to a hardware primitive in the FPGA Chip. The mapped design produces identifies the set of FPGA hardware primitives and their interconnection. The next step is to assign each of the components in the netlist to a equivalent physical primitives on the FPGA chip. Once this assignment or placement is made the interconnection between the components in the netlist must be made within the chip. This will require routing signals through the switch matrix and other inter connect resources available on FPGA Chip. This Place and route layout was generated from Xilinx ISE **Floor planner**. After place and route the design can be simulated to validate the design. At this point timing is more accurate because the propagation delays along routed signals and through CLBs can be more accurately estimated. This is particularly important for designs that are operating under tight timing tolerance.

To convert the RTL to gates, three steps typically occur:

* The RTL description is translated to an unoptimized boolean description usually consisting of primitive gates such as AND and OR gates, flip-flop, and latches. This is a functionally correct but completely unoptimized description.

* Boolean optimization algorithms are executed on this boolean equivalent description to produce an optimized boolean equivalent description.

* This optimized boolean equivalent description is mapped to actual logic gate by making use of a technology library of the target process.

PROCEDURE:

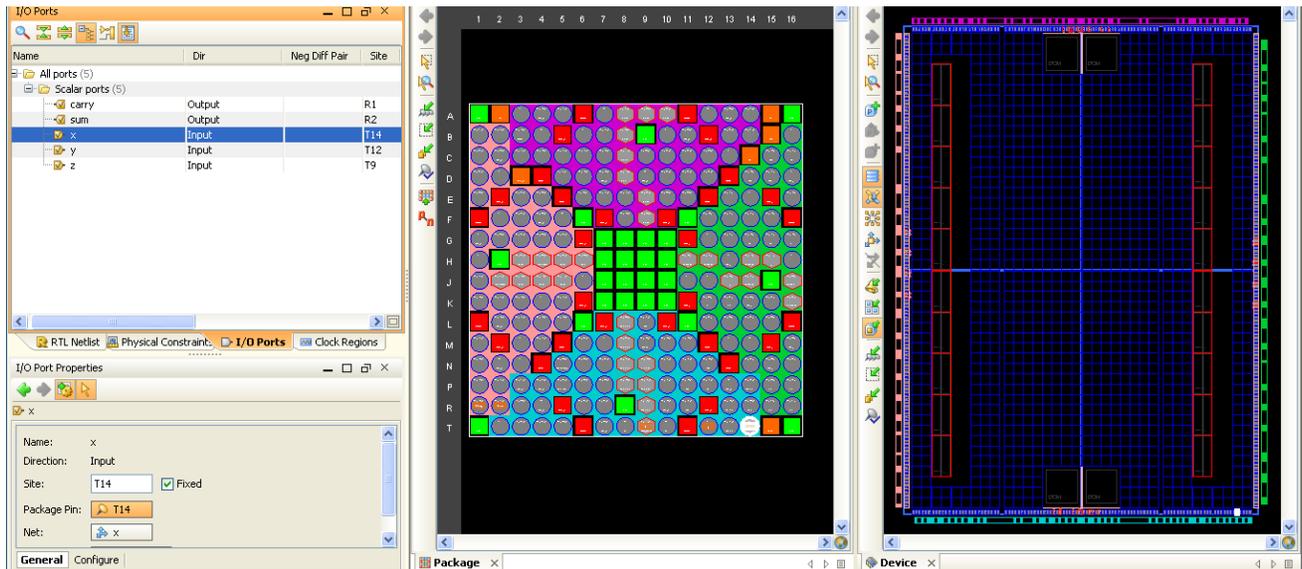
1. Now start the Xilinx ISE Design Suite 12.1
2. Go to file and click new project
3. Enter the project name and click next
4. Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next and

click Finish.

5. Click new source.
6. Select verilog module and type file name and click next.
7. Assign input and output port and click next.
8. Finally the report is shown click finish.
9. Types the program saves and clicks synthesis.
10. Choose Implementation → user constraints → I/O pin planning (plan ahead) pre- synthesis, type the input /output port.

PROGRAM:

```
module full_adder (a,b,c,sum,carry);  
output sum,carry ;  
input a,b,c ;  
assign sum = a ^ b ^ c;  
assign carry = (a&b) | (b&c) | (c&a);  
endmodule
```



RESULT:

EXP.NO:	STUDY OF FPGA BOARD AND ON-BOARD LED'S AND SWITCHES
DATE:	

AIM:

To Study Field Programmable Gate Array (FPGA) board and to test the on-board LEDs and Switches using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:**DIP SWITCHES:**

When in the UP or ON position, a switch connects the FPGA pin to V_{cc} , a logic High. When DOWN or in the OFF position, the switch connects the FPGA pin to ground, a logic low. The switches typically exhibit about 2ms of mechanical bounce and there is no active debouncing circuitry, although such circuitry could easily be added to the FPGA design programmed on the board.

KEY SWITCHES:

The key switches can provide pulse input to the FPGA. The switches connect to an associated FPGA pin. Pressing a key generates logic High on the associated FPGA pin. There is no active debouncing circuitry on the key switches.

LEDS:

Test LEDs are provided for mapping output of FPGA or tracking particular stage in the design. A series current limiting resistor of 270 ohm is associated with every LED. To turn on an individual LED, drive the associated FPGA control signal High.

PROCEDURE:

- 1.Create a new project & create a new Verilog file.
- 2.Type the program for testing LEDs and Switches and Save it
- 3.Synthesize the program and view the RTL Model.
- 4.Create test bench waveform and simulate it.
- 5.Download the program using the procedure given below into the FPGA.
- 6.Now test the physical working of the switches and the LED's on – board.

DOWNLOADING PROCEDURE:

1. Select “Synthesis/Implementation” in the source window.
2. Select the created module (*.v file) in the source window.
3. Select “user constraint” in the process window, double click “edit constraint” to create user constraint file (*.UCF).
4. Type the net list to define the I/O pins and save it.
5. Double click “implement design” in the process window.
6. Double click “Generate programming “file and select the respective created bit file (*.bit)
7. Double click “configure device (iMPACT)”. In the impact window that appears, select ‘configure device using boundary scan’. Click finish
8. Right click on the created Xilinx model & select ‘program’, give OK on the displayed window.

PROGRAM:

```
module leds(a, b);  
    input [0:15]a;  
    output [0:15]b;  
    reg [0:15]b;  
always@ (a)begin  
    b=~a;  
end  
endmodule
```

RESULT:

EXP.NO:	SIMULATION OF BASIC LOGIC GATES
DATE:	

AIM:

To write a verilog program for basic logic gates to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:

AND GATE:

An AND gate is a digital logic gate with two or more inputs and one output that performs logical conjunction. The output of an AND gate is true only when all of the inputs are true. If one or more of an AND gate's inputs are false, then the output of the AND gate is false.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
		A	B	
$Y = a \& b$				

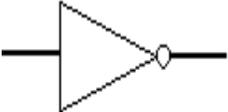
OR GATE:

An OR gate is a digital logic gate with two or more inputs and one output that performs logical disjunction. The output of an OR gate is true when one or more of its inputs are true. If all of an OR gate's inputs are false, then the output of the OR gate is false.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
		A	B	
$Y = a b$				

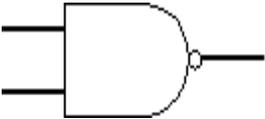
NOT GATE:

A NOT gate, often called an inverter, is a nice digital logic gate to start with because it has only a single input with simple behavior. A NOT gate performs logical negation on its input. In other words, if the input is true, then the output will be false. Similarly, a false input results in a true output.

EQUATION	LOGIC SYMBOL	TRUTH TABLE	
$Y = \sim a$		A	

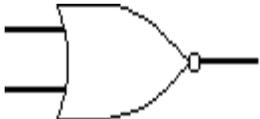
NAND GATE:

A NAND gate (sometimes referred to by its extended name, Negated AND gate) is a digital logic gate with two or more inputs and one output with behavior that is the opposite of an AND gate. The output of a NAND gate is true when one or more, but not all, of its inputs are false. If all of a NAND gate's inputs are true, then the output of the NAND gate is false.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
$Y = \sim(a \& b)$		A	B	Y

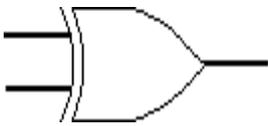
NOR GATE:

A NOR gate (sometimes referred to by its extended name, Negated OR gate) is a digital logic gate with two or more inputs and one output with behavior that is the opposite of an OR gate. The output of a NOR gate is **true** all of its inputs are **false**. If one or more of a NOR gate's inputs are **true**, then the output of the NOR gate is **false**.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
$Y = \sim(a b)$		A	B	Y

EX-OR GATE (XOR):

An XOR gate (sometimes referred to by its extended name, Exclusive OR gate) is a digital logic gate with two or more inputs and one output that performs exclusive disjunction. The output of an XOR gate is true only when exactly one of its inputs is true. If both of an XOR gate's inputs are false, or if both of its inputs are true, then the output of the XOR gate is false.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
		A	B	Y
$Y = a \wedge b$				

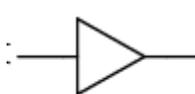
EX-NOR GATE (XNOR):

An XNOR gate (sometimes referred to by its extended name, Exclusive NOR gate) is a digital logic gate with two or more inputs and one output that performs logical equality. The output of an XNOR gate is true when all of its inputs are true or when all of its inputs are false. If some of its inputs are true and others are false, then the output of the XNOR gate is false.

EQUATION	LOGIC SYMBOL	TRUTH TABLE		
		A	B	Y
$Y = \sim(a \wedge b)$				

BUFFER GATE:

A buffer has only a single input and a single output with behavior that is the opposite of an NOT gate. It simply passes its input, unchanged, to its output. In a boolean logic simulator, a buffer is mainly used to increase propagation delay. In a real-world circuit, a buffer can be used to amplify a signal if its current is too weak.

EQUATION	LOGIC SYMBOL	TRUTH TABLE	
		A	Y
$Y = A$			

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using ISIM Simulator.

PROGRAM:**BASIC GATES:**

DATAFLOW	BEHAVIORAL	STRUCTURAL
NOT GATE		
<pre> module notgate(a,y); input a; output y; assign y= ~a; endmodule </pre>	<pre> module notgate (a,y); input a; output y; reg y; always @(a) begin y=~a; end endmodule </pre>	<pre> module notgate (a,y) input a; output y; inv a1(y,a); endmodule </pre>
OR GATE		
<pre> module orgate(a,b,y); input a,b; output y; assign y=(a b); endmodule </pre>	<pre> module orgate(a,b,y); input a,b; output y; reg y; always @(a,b) begin y=a b; end endmodule </pre>	<pre> module orgate(a,b,y); input a,b; output y; or a1(y,a,b); endmodule </pre>
AND GATE		
<pre> module andgate(a,b,y); input a,b; output y; assign y=(a&b); endmodule </pre>	<pre> module andgate(a,b,y); input a,b; output y; reg y; always @(a,b) begin y=a&b; end endmodule </pre>	<pre> module andgate(a,b,y); input a,b; output y; and a1(y,a,b); endmodule </pre>

NOR GATE

```
module norgate(a,b,y);  
input a,b;  
output y;  
assign y=~(a|b);  
endmodule
```

```
module norgate(a,b,y);  
input a,b;  
output y;  
reg y;  
always @(a,b)  
begin  
y=~(a|b);  
end  
endmodule
```

```
module norgate(a,b,y);  
input a,b;  
output y;  
nor a1(y,a,b);  
endmodule
```

NAND Gate

```
module nandgate(a,b,y);  
input a,b;  
output y;  
assign y=~(a&b);  
endmodule
```

```
module nandgate(a,b,y);  
input a,b;  
output y;  
reg y;  
always @(a,b)  
begin  
y=~(a&b);  
end  
endmodule
```

```
module nandgate(a,b,y);  
input a,b;  
output y;  
nand a1(y,a,b);  
endmodule
```

XOR Gate

```
module xorgate(a,b,y);  
input a,b;  
output y;  
assign y=(a^b);  
endmodule
```

```
module xorgate(a,b,y);  
input a,b;  
output y;  
reg y;  
always @(a,b)  
begin  
y=a^b;  
end  
endmodule
```

```
module xorgate(a,b,y);  
input a,b;  
output y;  
xor a1(y,a,b);  
endmodule
```

XNOR GATE

```
module xnorgate(a,b,y);  
input a,b;  
output y;  
assign y=~(a^b);  
endmodule
```

```
module xnorgate(a,b,y);  
input a,b;  
output y;  
reg y;  
always @(a,b)
```

```
module xnorgate(a,b,y);  
input a,b;  
output y;  
xnor a1(y,a,b);  
endmodule
```

	<pre>begin y=~(a^b); end endmodule</pre>	
--	--	--

BUFFER GATE

<pre>module bufgate (a,y); input a; output y; assign y= a; endmodule</pre>	<pre>module bufgate (a,y); input a; output y; reg y; always @(a) begin y=a; end endmodule</pre>	<pre>module bufgate (a,y); input a; output y; buf(y,a); endmodule</pre>
--	---	---

BASIC GATES (ALL GATES IN ONE PROGRAM)

<pre>module gates(a,b,c,d,e,f,g,h,i,j); input a,b; output c,d,e,f,g,h,i,j; assign c=a&b; assign d=a b; assign e=~a; assign f=a; assign g=~(a b); assign h=a^b; assign i=~(a^b); assign j=~(a&b); endmodule</pre>	<pre>module gates(a,b,c,d,e,f,g,h,i,j); input a,b; output c,d,e,f,g,h,i,j; reg c,d,e,f,g,h,i,j; always @(a,b) begin c=a&b; d=a b; e=~a; f=a; g=~(a b); h=a^b; i=~(a^b); j=~(a&b); end endmodule</pre>	<pre>module gates(a,b,c,d,e,f,g,h,i,j); input a,b; output c,d,e,f,g,h,i,j; and (c,a,b); or(d,a,b); not(e,a); buf(f,a); nor(g,a,b); xor(h,a,b); xnor(i,a,b); nand(j,a,b); endmodule</pre>
--	---	--

SIMULATION REPORT

RESULT:

EXP.NO:	SIMULATION OF ADDER
DATE:	

AIM:

To write a Verilog program for Adder and to synthesize, simulate it using Xilinx software tool.

TOOLS REQUIRED:

Software :

1. Xilinx ISE Design Suite 12.1

THEORY :

When you add large numbers carefully together the addition is done digit by digit. In the illustration, two 8 –digit binary numbers are being added. The top row contains the first number and the second row the other. Working from the right-hand side, there can be no 'carry' to add to the sum of the first two digits, so a half adder is sufficient. But for the second and subsequent pairs of digits, full adders must be use (any carry' is indicated by a f below the adder). The output will be an 8-bit and if the carry is formed that will be shown in cout output value.

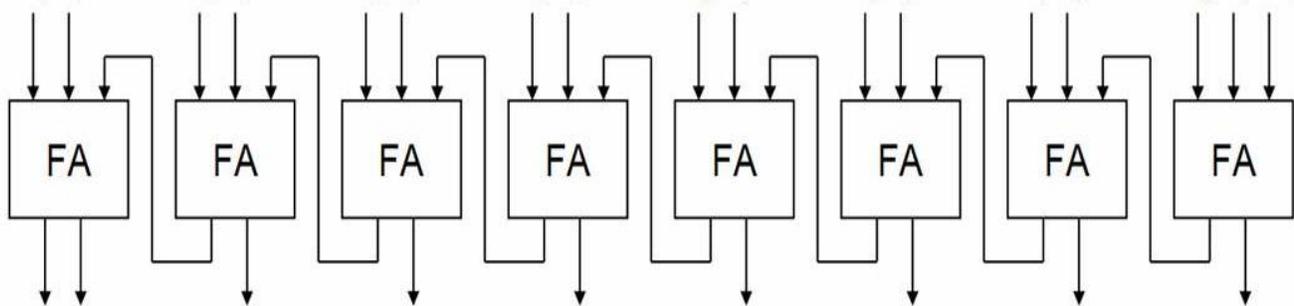
PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double
4. Clicking on the synthesis in the process window.
5. Perform the functional simulation using Xilinx ISE simulator.
6. The output can be observed by using ISIM Simulator.

PROGRAM:

8-BIT RIPPLE CARRY ADDER:

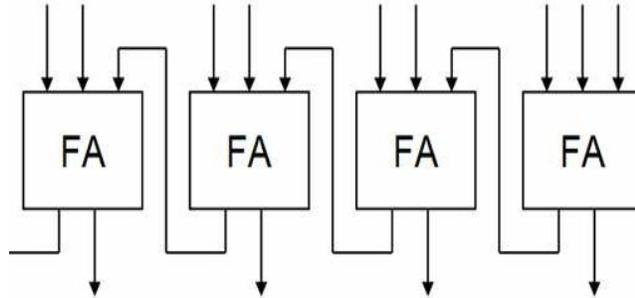


```
module half_adder(a,b,sum,carry);
input a,b;
output sum,carry;
xor(sum,a,b);
and(carry,a,b);
endmodule

module full_adder(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
wire w1,w2,w3;
half_adder ha1(a,b,w1,w2);
half_adder ha2(c,w1,sum,w3);
or(carry,w2,w3);
endmodule

module rippleadder(a, b, cin, sum, cout);
input [7:0] a,b;
input cin;
output [7:0]sum;
output cout;
wire[6:0] c;
full_adder a1(a[0],b[0],cin,sum[0],c[0]);
full_adder a2(a[1],b[1],c[0],sum[1],c[1]);
full_adder a3(a[2],b[2],c[1],sum[2],c[2]);
full_adder a4(a[3],b[3],c[2],sum[3],c[3]);
full_adder a5(a[4],b[4],c[3],sum[4],c[4]);
full_adder a6(a[5],b[5],c[4],sum[5],c[5]);
full_adder a7(a[6],b[6],c[5],sum[6],c[6]);
full_adder a8(a[7],b[7],c[6],sum[7],cout);
endmodule
```

4-BIT RIPPLE CARRY ADDER



```
module half_adder(a,b,sum,carry);  
input a,b;  
output sum,carry;  
xor(sum,a,b);  
and(carry,a,b);  
endmodule
```

```
module full_adder(a,b,c,sum,carry);  
input a,b,c;  
output sum,carry;  
wire w1,w2,w3;  
half_adder ha1(a,b,w1,w2);  
half_adder ha2(c,w1,sum,w3);  
or(carry,w2,w3);  
endmodule
```

```
module rippleadder (a, b, cin, sum, cout);  
input [3:0]a,b;  
input cin;  
output [3:0]sum;  
output cout;  
wire[2:0] c;  
full_adder a1(a[0],b[0],cin,sum[0],c[0]);  
full_adder a2(a[1],b[1],c[0],sum[1],c[1]);  
full_adder a3(a[2],b[2],c[1],sum[2],c[2]);  
full_adder a4(a[3],b[3],c[2],sum[3],cout);  
endmodule
```

SIMULATION REPORT:

RESULT:

EXP.NO:	SIMULATION OF MULTIPLIER
DATE:	

AIM:

To write a Verilog program for 4-bit multiplier and to synthesize, simulate it using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

THEORY:

Multiplication of two elements in the polynomial basis can be accomplished in the normal way of multiplication, but there are a number of ways to speed up multiplication, especially in hardware. In this type the multiplication can be done parallel counter and it generates carry. The multiplication is independent of the carry so we can perform N number of multiplications independent of carry.

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double
4. Clicking on the synthesis in the process window.
5. Perform the functional simulation using Xilinx ISE simulator.
6. The output can be observed by using ISIM Simulator.

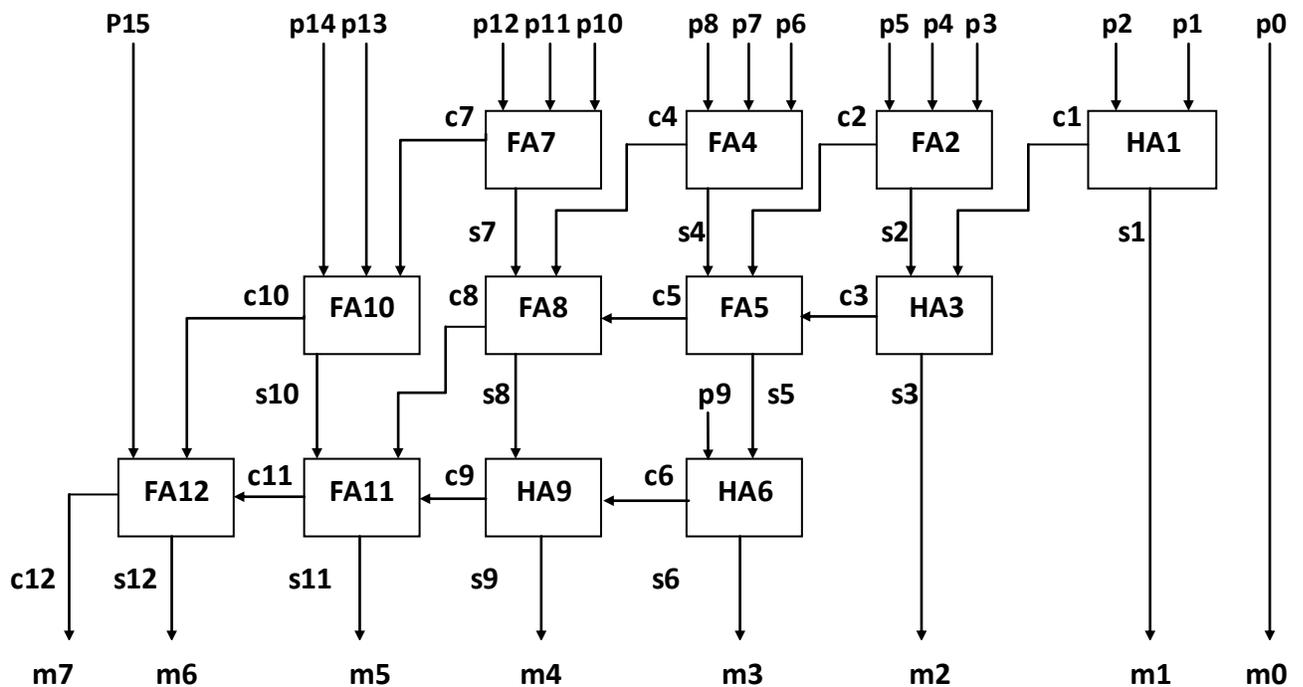
PROGRAM:

4-BIT ARRAY MULTIPLIER:

$$\begin{array}{r}
 \\
 a3 \\
 a2 \\
 a1 \\
 X b3 b1 \\
 \hline
 a0b0 \\
 a3b1 a0b1 \\
 a3b2 a0b2 \\
 a3b3 a0b3
 \end{array}$$

m7 m6 m5 m4 m3 m2 m1 m0

$a0b0=p0$ $a0b1=p2$ $a0b2=p5$ $a0b3=p9$
 $a1b0=p1$ $a1b1=p4$ $a1b2=p8$ $a1b3=p12$
 $a2b0=p3$ $a2b1=p7$ $a2b2=p11$ $a2b3=p14$
 $a3b0=p6$ $a3b1=p10$ $a3b2=p13$ $a3b3=p15$



PROGRAM:

```
module half_adder(a,b,sum,carry);  
input a,b;  
output sum,carry;  
xor(sum,a,b);  
and(carry,a,b);  
endmodule
```

```
module full_adder(a,b,c,sum,carry);  
input a,b,c;  
output sum,carry;  
wire w1,w2,w3;  
half_adder ha1(a,b,w1,w2);  
half_adder ha2(c,w1,sum,w3);  
or(carry,w2,w3);  
endmodule
```

```
module arraymultiplier(m,a,b);  
input [3:0]a,b;  
output [7:0]m;  
wire [15:0]p;  
wire [12:1]s,c;  
and(p[0],a[0],b[0]);  
and(p[1],a[1],b[0]);  
and(p[2],a[0],b[1]);  
and(p[3],a[2],b[0]);  
and(p[4],a[1],b[1]);  
and(p[5],a[0],b[2]);  
and(p[6],a[3],b[0]);  
and(p[7],a[2],b[1]);  
and(p[8],a[1],b[2]);  
and(p[9],a[0],b[3]);  
and(p[10],a[3],b[1]);  
and(p[11],a[2],b[2]);
```

```

and(p[12],a[1],b[3]);
and(p[13],a[3],b[2]);
and(p[14],a[2],b[3]);
and(p[15],a[3],b[3]);
half_adder ha1(s[1],c[1],p[1],p[2]);
full_adder fa2(s[2],c[2],p[4],p[3],p[5]);
half_adder ha3(s[3],c[3],s[2],c[1]);
full_adder fa4(s[4],c[4],p[6],p[7],p[8]);
full_adder fa5(s[5],c[5],s[4],c[2],c[3]);
half_adder ha6(s[6],c[6],s[5],p[9]);
full_adder fa7(s[7],c[7],p[10],p[11],p[12]);
full_adder fa8(s[8],c[8],c[5],c[4],s[7]);
half_adder ha9(s[9],c[9],s[8],c[6]);
full_adder fa10(s[10],c[10],p[14],p[13],c[7]);
full_adder fa11(s[11],c[11],c[9],c[8],s[10]);
full_adder fa12(s[12],c[12],p[15],c[10],c[11]);
buf(m[0],p[0]);
buf(m[1],s[1]);
buf(m[2],s[3]);
buf(m[3],s[6]);
buf(m[4],s[9]);
buf(m[5],s[11]);
buf(m[6],s[12]);
buf(m[7],c[12]);
endmodule

```

4-BIT MULTIPLIER

```

module adder(a,b, out);
input [3:0]a,b;
output [7:0]out;
assign out= a*b;
endmodule

```

SIMULATION REPORT:

RESULT:

EXP.NO:	SIMULATION OF 8-BIT ARITHMETIC AND LOGIC UNIT
DATE:	

AIM:

To write a Verilog program for 8-bit Arithmetic and Logic Unit and to synthesize, simulate it using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

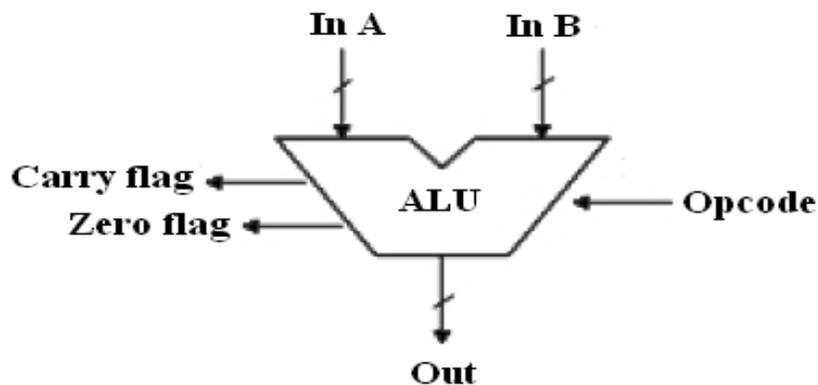
1. Xilinx ISE Design Suite 12.1

THEORY:

8-BIT ARITHMETIC AND LOGIC UNIT:

ALU is the fundamental building block of the processor, which is responsible for carrying out the arithmetic and logic functions. ALU comprises of combinatorial logic that implements arithmetic operations such as Addition, Subtraction and Multiplication, and logic operations such as AND, OR, NOT. The ALU gets operands from the register file or memory.

The block diagram of a typical ALU is shown in Figure below. The ALU reads two input operands In A and In B. The operation to perform on these input operands is selected using the control input Opcode. The ALU performs the selected operation on the input operands In A and In B and produces the output, Out. The ALU also updates different flag signals after performing the selected function. Note that the ALU is purely combinatorial logic and contains no registers or latches.



The **arithmetic functions** are much more complex to implement than the **logic functions**. The performance of the **ALU** depends upon the architecture of each structural components of the **ALU**. In this example only some basic **ALU** functions are implemented. The **ALU** is divided into an arithmetic section and a logical section.

The **Arithmetic Unit** compromises of three functions. They are:

- Addition
- Subtraction
- Multiplication

The **Logical Unit** compromises of five functions. They are:

- Bitwise AND
- Bitwise OR
- Bitwise NAND
- Bitwise NOR
- Bitwise XOR

Flags: **ALU** updates the **conditional flags**, which are used by the processor to perform other operations like condition checking and branching. In this example two flags are implemented. They are:

Zero - If all the bits of the result data are zero then the zero flag is set

- **Carry out** - If the addition of the two operands gives the carry out this flag is set

2. Truth Table

The Figure below shows the truth table of the **8-bit ALU**. The **ALU** works on **8-bit operands**. It supports **8 instructions** which are selected by the **3-bit Opcode**.

S.No.	Opcode	Operation
1	000	Result = Operand1 + Operand2
2	001	Result = Operand1 - Operand2
3	010	Result = Operand1 * Operand2
4	011	Result = Operand1 & Operand2
5	100	Result = Operand1 Operand2
6	101	Result = ~(Operand1 & Operand2)
7	110	Result = ~(Operand1 Operand2)
8	111	Result = Operand1 ^ Operand2

Truth table for **8-bit ALU**.

PROCEDURE:

Software part:

1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using ISIM Simulator

Verilog Module:

```

module ALU1 (
    input [7:0] A,B, // ALU 8-bit Inputs
    input [3:0] ALU_Sel, // ALU Selection
    output [7:0] ALU_Out, // ALU 8-bit Output
    output CarryOut // Carry Out Flag
);
    reg [7:0] ALU_Result;
    wire [8:0] tmp;
    assign ALU_Out = ALU_Result; // ALU out
    assign tmp = {1'b0,A} + {1'b0,B};
    assign CarryOut = tmp[8]; // Carryout flag

```

```

always @(*)
begin
  case(ALU_Sel)
  4'b0000: // Addition
    ALU_Result = A + B ;
  4'b0001: // Subtraction
    ALU_Result = A - B ;
  4'b0010: // Multiplication
    ALU_Result = A * B;
  4'b0100: // Logical shift left
    ALU_Result = A<<1;
  4'b0101: // Logical shift right
    ALU_Result = A>>1;
  4'b0110: // Rotate left
    ALU_Result = {A[6:0],A[7]};
  4'b0111: // Rotate right
    ALU_Result = {A[0],A[7:1]};
  4'b1000: // Logical and
    ALU_Result = A & B;
  4'b1001: // Logical or
    ALU_Result = A | B;
  4'b1010: // Logical xor
    ALU_Result = A ^ B;
  4'b1011: // Logical nor
    ALU_Result = ~(A | B);
  4'b1100: // Logical nand
    ALU_Result = ~(A & B);
  4'b1101: // Logical xnor
    ALU_Result = ~(A ^ B);
  4'b1110: // Greater comparison
    ALU_Result = (A>B)?8'd1:8'd0 ;
  4'b1111: // Equal comparison
    ALU_Result = (A==B)?8'd1:8'd0 ;
  default: ALU_Result = A + B ;
end

```

```
endcase  
end  
endmodule
```

SIMULATION REPORT:

RESULT:

EXP.NO:	SIMULATION OF UNIVERSAL SHIFT REGISTER
DATE:	

AIM:

To write a verilog program for Universal Shift Register and to synthesize, simulate it using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY :

UNIVERSAL SHIFT REGISTER:

A Unidirectional shift register is a register that can capable of transferring data in only one direction. Whereas the register that is capable of transferring data in both left and right direction is called a 'bidirectional shift register.' Now let we have a register which can capable to transfer data in both the shift-right and shift-left, along with the necessary input and output terminals for parallel transfer, then it is called a *shift register* with *parallel load* or 'universal shift register.'

A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift-right.

A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.

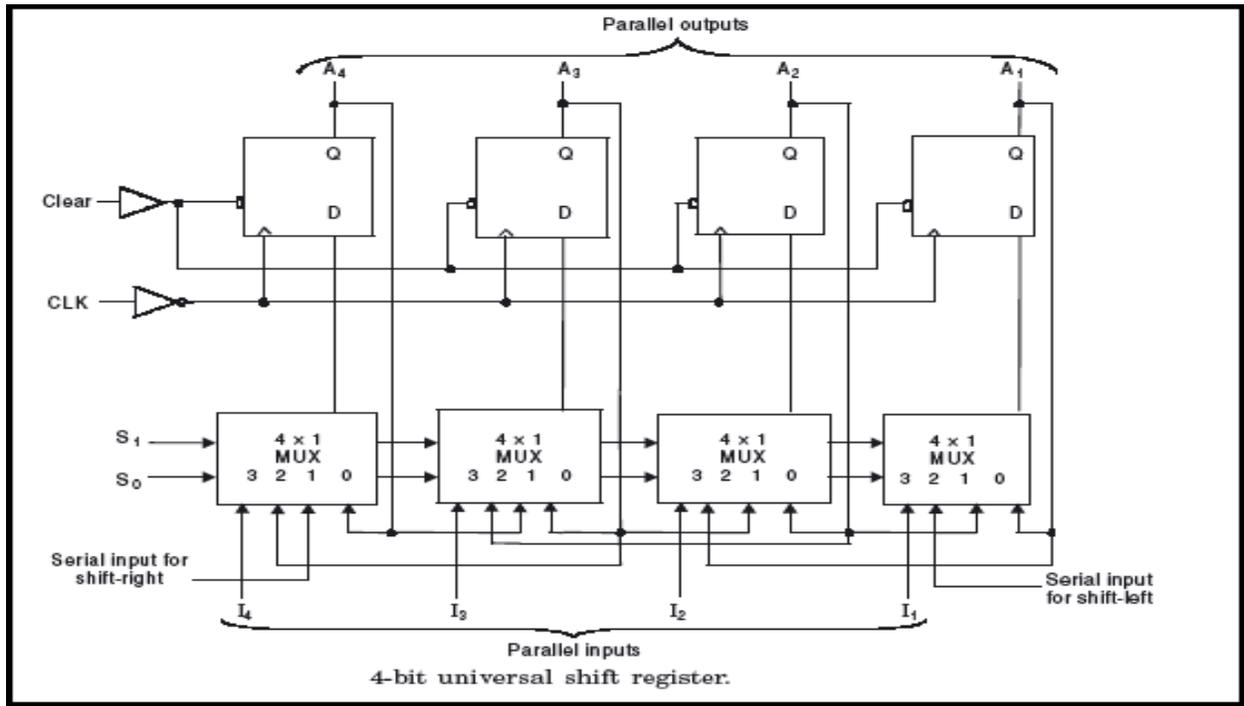
A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.

n parallel output lines.

A *clear* control to clear the register to 0.

A *CLK* input for clock pulses to synchronize all operations.

A control state that leaves the information in the register unchanged even though clock pulses are continuously applied.



PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using ISIM Simulator.

PROGRAM:

S1	S0	REGISTER OPERATION
0	0	No changes
0	1	Shift right
1	0	Shift left
1	1	Parallel load

```
module universal shift(a,s,clk,p);
```

```
input [3:0]a;
```

```
input [1:0]s;
```

```
input clk;
```

```
output reg [3:0]p;
```

```
initial
```

```
p<=4'b0110;
```

```
always@ (posedge clk)
```

```
begin
```

```
case (s)
```

```
2'b00:
```

```
begin
```

```
p[3]<=p[3];
```

```
p[2]<=p[2];
```

```
p[1]<=p[1];
```

```
p[0]<=p[0];
```

```
end
```

```
2'b01:
```

```
begin
```

```
p[3]<=p[0];
```

```
p[2]<=p[3];
```

```
p[1]<=p[2];
```

```
p[0]<=p[1];
```

```
end
```

```
2'b10:
begin
p[0]<=p[3];
p[1]<=p[0];
p[2]<=p[1];
p[3]<=p[2];
end
2'b11:
begin
p[0]<=a[0];
p[1]<=a[1];
p[2]<=a[2];
p[3]<=a[3];
end
endcase
end
endmodule
```

SIMULATION REPORT:

RESULT

EXP.NO:	SIMULATION AND DESIGN OF MEMORIES
DATE:	

AIM:

To write a Verilog program for a 16x4 memory and to synthesize, simulate it using Xilinx software tool.

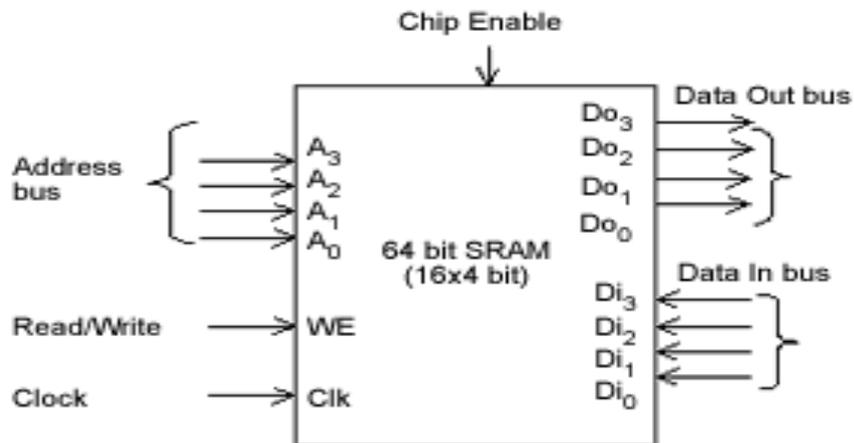
TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:

MOD 10 COUNTERS:



The memory block diagram is shown in above figure. It takes a few assumptions into consideration for easing the operations of the circuit. While data input pin and address pin may have any value depending on the specifications of memory used and your need, clock used in the circuit is active high.

Enable pin triggers the circuit when it is active high, and read operation is performed when read/write pin is high, while write operation is performed when read/write pin is active low.

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using ISIM Simulator.

PROGRAM:

```
module memory_16x4(op,ip,rd_wr,clk,address);
output reg [3:0] op;
input [3:0] ip;
input [3:0] address;
input rd_wr,clk;
reg [3:0] memory[0:15];
always @(posedge clk)
begin
if (rd_wr)
op=memory[address];
else
begin
memory[address]=ip;
end
end
endmodule // memory_16x4
```

Sr. No.	Name of the Pin	Direction	Width	Description
1	Address	Input	4	Input address
2	Ip	input	4	Input data to memory
3	Rd_wr	Input	1	Control signal 1=read from memory 0=write in to memory
4	Clk	Input	1	Clock input
5	op	Output	4	Output read from memory

SIMULATION REPORT:

RESULT:

EXP.NO:	ANALYSIS AND SIMULATION OF STATE MACHINES
DATE:	

AIM:

To analyze, synthesize and simulate state machines using Xilinx simulation tool

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

THEORY:

MEALY MODEL:

Analysis describes what a given circuit will do under certain operating conditions. The behavior of a clocked sequential circuit is determined from the inputs, the outputs, and the state of its flip-flops. The outputs and the next state are both a function of the inputs and the present state. The analysis of a sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states. It is also possible to write Boolean expressions that describe the behavior of the sequential circuit. These expressions must include the necessary time sequence, either directly or indirectly. A logic diagram is recognized as a clocked sequential circuit if it includes flip-flops with clock inputs. The flip-flops may be of any type, and the logic diagram may or may not include combinational logic gates.

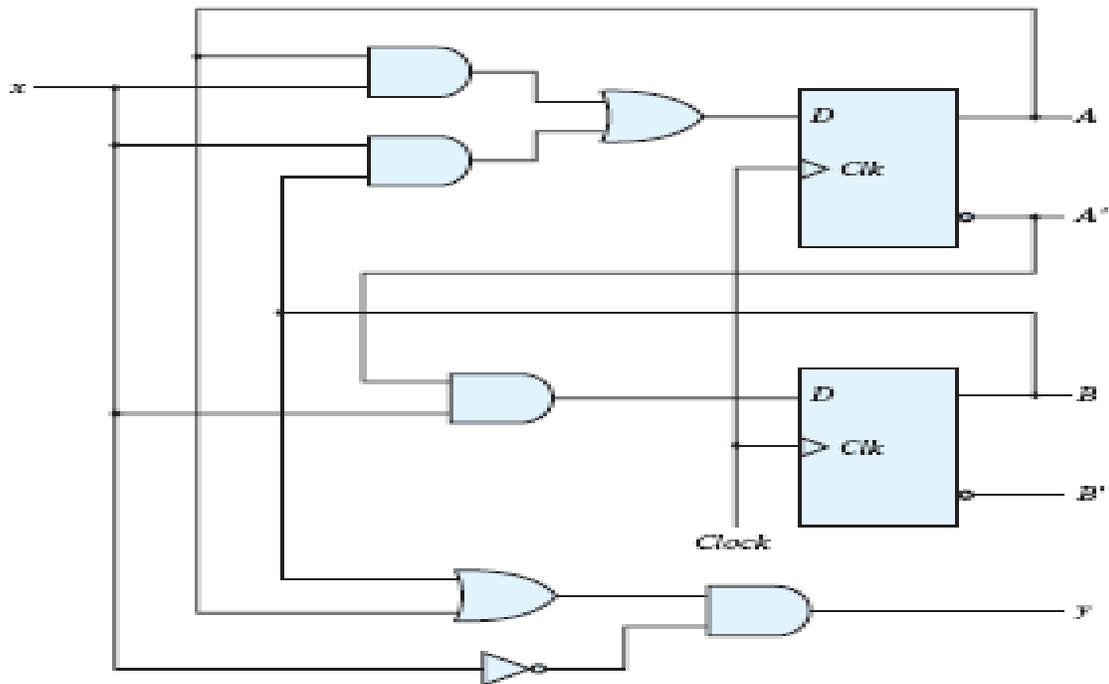
State Equations: The behavior of a clocked sequential circuit can be described algebraically by means of state equations. A *state equation* (also called a *transition equation*) specifies the next state as a function of the present state and inputs. Consider the sequential circuit shown in Fig. 1. It acts as a 0-detector by asserting its output when a 0 is detected in a stream of 1s.

$$A(t + 1) = A(t)x(t) + B(t)x(t) ; B(t + 1) = A'(t)x(t); y(t) = [A(t) + B(t)]x'(t)$$

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation, with $(t + 1)$, denotes the next state of the flip-flop one clock edge later. The right side of the equation is a Boolean expression that specifies the present state and input conditions that make the next state equal to 1. Since all the variables in the Boolean expressions are a function of the present state, we can omit the designation (t) after each variable for convenience and can express the state equations in the more compact form

$$A(t + 1) = Ax + Bx ; B(t + 1) = A'x ; y = Ax' + Bx'$$

LOGIC DIAGRAM:



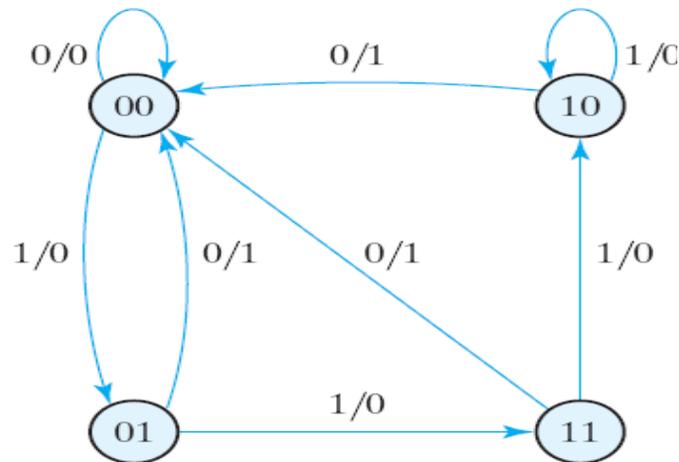
STATE TABLE:

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a *state table* (sometimes called a *transition table*). The table consists of four sections labeled *present state*, *input*, *next state*, and *output*. The present-state section shows the states of flip-flops A and B at any given time t . The input section gives a value of x for each possible present state. The next-state section shows the states of the flip-flops one clock cycle later, at time $t + 1$. The output section gives the value of y at time t for each present state and input condition. State table for the above Circuit diagram

Present State		Input	Next State		Output
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

STATE DIAGRAM:

The information available in a state table can be represented graphically in the form of a state diagram. In this type of diagram, a state is represented by a circle, and the (clock-triggered) transitions between states are indicated by directed lines connecting the circles. The state diagram provides the same information as the state table and is obtained directly from Table. The binary number inside each circle identifies the state of the flip-flops. The directed lines are labeled with two binary numbers separated by a slash. The input value during the present state is labeled first, and the number after the slash gives the output during the present state with the given input. A directed line connecting a circle with itself indicates that no change of state occurs.



PROGRAM:

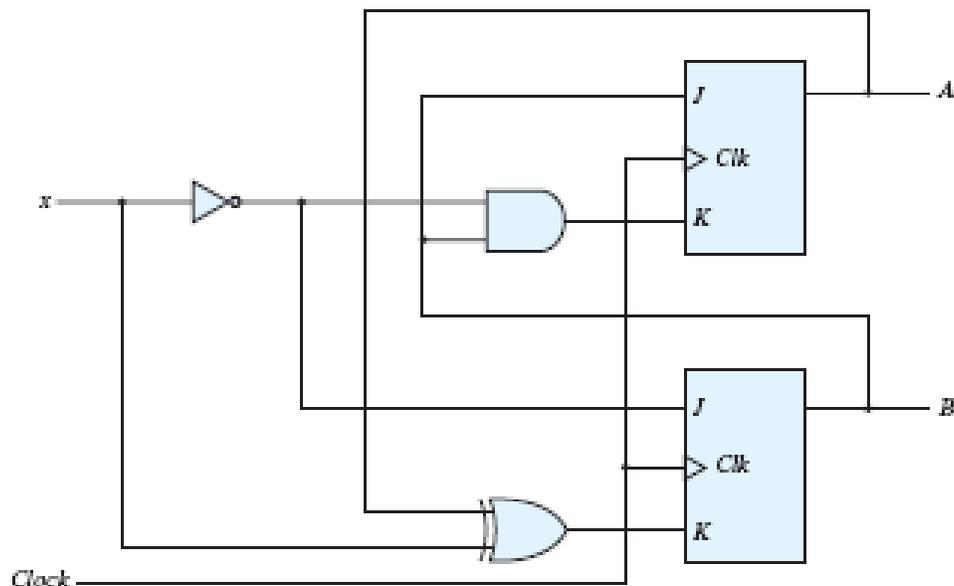
MEALY MODEL:

```
module Mealy_model(y, x, clk, reset);  
input x,clk,reset;  
output reg y;  
reg [1:0] state, next_state;  
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;  
always @ (posedge clk)  
if (reset == 0) state <= S0;  
else state <= next_state;  
always @ (state, x)  
case (state)  
S0: if (x) next_state = S1; else next_state = S0;
```

```
S1: if (x) next_state = S3; else next_state = S0;
S2: if (x) next_state = S2; else next_state = S0;
S3: if (x) next_state = S2; else next_state = S0;
endcase
always @ (state, x)
case (state)
S0: y = 0;
S1, S2, S3: y = ~x;
endcase
endmodule
```

MOORE MODEL:

LOGIC DIAGRAM:



STATE TABLE:

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a *state table* (sometimes called a *transition table*). The table consists of four sections labeled *present state*, *input*, *next state*, and *output*. The present-state section shows the states of flip-flops A and B at any given time t . The input section gives a value of x for each possible present state. The next-state section shows the states of the flip-flops one clock cycle later, at time $t + 1$. The output section gives the value of y at time t for each present state and input condition.

Present State		Input x	Next State		Flip- Flop Inputs			
A	B		A	B	J _A	K _A	J _B	K _B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0

1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

STATE EQUATIONS:

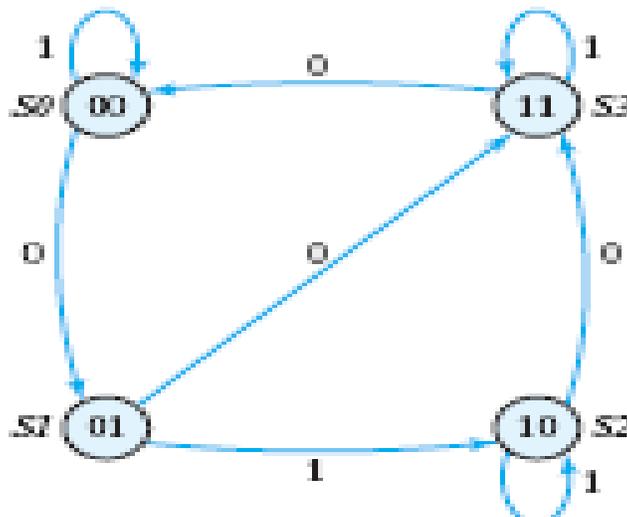
$$A(t+1) = JA' + K'A$$

$$B(t+1) = JB' + K'B$$

$$A(t+1) = BA' + (Bx')'A = A'B + AB' + Ax$$

$$B(t+1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

STATE DIAGRAM:



MOORE MODEL:

```

module Moore_Model(y, x, clk, reset);
input x,clk,reset;
output [1:0]y;
reg [1:0] state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
always @ (posedge clk)
if (reset == 0) state <= S0;
else case (state)
S0: if (x) state <= S0; else state <= S1;
S1: if (x) state <= S2; else state <= S3;
S2: if (x) state <= S2; else state <= S3;
  
```

```
S3: if (x) state <= S3; else state <= S0;
```

```
endcase
```

```
assign y=state;
```

```
endmodule
```

RESULT:

TANNER

TOOLS

Introduction to Tanner Tool

Tanner tool is a Spice Computer Analysis Programmed for Analogue Integrated Circuits. Tanner tool consists of the following Engine Machines:

1. S-EDIT (Schematic Edit)
2. T-EDIT (Simulation Edit)
3. W-EDIT (Waveforms Edit)
4. L-EDIT (Layout Edit)

Using these engine tools, spice program provides facility to the use to design & simulate new ideas in Analogue Integrated Circuits before going to the time consuming & costly process of chip fabrication.

➤ SCHEMATIC EDIT TOOL (S-EDIT)

S-Edit is hierarchy of files, modules & pages. It introduces symbol & schematic modes. S-Edit provides the facility of:

1. Beginning a design.
2. Viewing, drawing & editing of objects.
3. Design connectivity.
4. Properties, net lists & simulation.
5. Instance & browse schematic & symbol mode.

Beginning a design: It explains the design process in detail in terms of file module operation and module.

Browser: Effective schematic design requires a working knowledge of the S-Edit design hierarchy of files & modules. S-Edit design files consist of modules. A module is a functional unit of design such as a transistor, a gate and an amplifier.

Modules contain two components:

- 1) Primitives: Geometrical objects created with drawing tools.
- 2) Instances: References to other modules in file. The instanced module is the original.

S-Edit has two viewing modes:

1. Schematic Mode: to create or view a schematic, we operate in schematic mode.
2. Symbol Mode: it represents symbol of a larger functional unit such as operational amplifier.

T-SPICE PRO CIRCUIT ANALYSIS

An introduction to the integrated components of the T- Spice Pro circuit analysis suite:

Schematic data files (.sdb): describes the circuits to be analyzed in graphical form, for display and editing by **S- Edit" Schematic Editor**.

Simulation input files (.sp): describes the circuits to be analyzed in textual form, for editing and simulation by **T- Spice" Circuit Simulator**.

Simulation output files (.out): containing the numerical results of the circuit analyses, for manipulation and display by **W- Edit" Waveform Viewer**.

➤ CIRCUIT SIMULATOR (T-SPICE)

T- Spice Pro's waveform probing feature integrates S- Edit, T- Spice, and W- Edit to allow individual points in a circuit to be specified and analyzed. A few analysis is described below:

The heart of T-Spice operation is the input file (also known as the circuit description, the net list & the input deck). This is a plain text file that contains the device statement & simulation commands, drawn from the SPICE circuit description language with which T-Spice constructs a model of the circuit to be simulated. Input files can be created and modified with any text editor.

T-Spice is a tool used for simulation of the circuit. It provides the facility of

1. Design Simulation
2. Simulation Commands
3. Device Statements
4. User-Designed External Models
5. Small Signal & Noise Models

T-Spice uses Kirchhoff's Current Law (KCL) to solve circuit problems. To T-Spice, a circuit is a set of devices attached to nodes. The voltage at all nodes represents the circuit state. T-Spice solves for a set of node voltage that satisfied KCL (implying that sum of currents flowing into each node is zero). In order to evaluate whether a set of node voltages is a solution, T-Spice computers and sums all the current flowing out of each device into nodes connected to it (its terminals). The relationship between the voltages at device terminals and the currents through the terminal is determined by the device model for a resistor of resistance R is

$$I=\Delta V/R$$

Where, ΔV represents the voltage difference across the device. A few analyses are discussed below:

DC Operating Point Analysis

DC operating point analysis finds a circuit's steady- state condition, obtained (in principle) after the input voltages have been applied for an infinite amount of time. The .include command causes T- Spice to read in the contents of the model file for the evaluation of NMOS and PMOS transistors.

The technology file assigns values to MOSFET model parameters for both n - and p -type devices. When read by the input file, these parameters are used to evaluate MOSFET model equations, and the results are used to construct internal tables of current and charge values. Values read or interpolated from these tables are used in the computations called for by the simulation. Following each transistor name are the names of its terminals. The required order of terminal names is: drain -gate -source -bulk. Then the model name (NMOS or PMOS in this example), and physical characteristics such as length and width, are specified. The .op command performs a DC operating point calculation and writes the results to the file specified in the Simulate > Start Simulation dialog. The output file lists the DC operating point information for the circuit described by the input file.

DC Transfer Analysis

DC transfer analysis is used to study the voltage or current at one set of points in a circuit as a function of the voltage or current at another set of points. This is done by sweeping the source variables over specified ranges, and recording the output. A list of sources to be swept, and the voltage ranges across which the sweeps are to take place follow the .dc command, indicating transfer analysis. The transfer analysis will be performed as follows: vdd will be set at 5 volts and vin will be swept over its specified range; vdd will then be incremented and vin will be reswept over its range; and so on, until vdd reaches the upper limit of its range. The .dc command ignores the values assigned to the voltage sources vdd and vin in the voltage source statements, but they must still be declared in those statements. The results for nodes in and out are reported by the .print dc command to the specified destination.

Transient Analysis

Transient analysis provides information on how circuit elements vary with time. The basic T- Spice command for transient analysis has three modes. In the default mode, the DC operating point is computed, and T- Spice uses this as the starting point for the transient simulation. The .tran command specifies the characteristics of the transient analysis to be performed.

AC Analysis

AC analysis characterizes the circuit's behavior dependence on small- signal input frequency. It involves

three steps: (1) calculating the DC operating point; (2) linearizing the circuit; and (3) solving the linearized circuit for each frequency. When ac voltage source is to be applied, then vdiff sets the DC voltage difference between nodes the two nodes to -0.0007 volts; its AC magnitude is 1 volt and its AC phase is 180 degrees. The .ac command performs an AC analysis. Following the .ac keyword is information concerning the frequencies to be swept during the analysis. In case, the frequency is to be swept logarithmically, by decades (DEC); 5 data points are to be included per decade is considered to be the standard. The two .print commands write the voltage magnitude (in decibels) and phase (in degrees), respectively, for the node out to the specified file. The .acmodel command writes the small-signal model parameters and operating point voltages and currents for all circuit devices.

Noise Analysis

Real circuits, of course, are never immune from small, random fluctuations in voltage and current levels. In T-Spice, the influence of noise in a circuit can be simulated and reported in conjunction with AC analysis. The purpose of noise analysis is to compute the effect of the noise associated with various circuit devices on an output voltage or voltages as a function of frequency. Noise analysis is performed in conjunction with AC analysis; if the .ac command is missing, then the .noise command is ignored. With the .ac command present, the .noise command causes noise analysis to be performed at the same frequencies. The .noise command takes two arguments: the output at which the effects of noise are to be computed, and the input at which the noise can be considered to be concentrated for the purposes of estimating the equivalent noise spectral density. The print command is used to print results.

➤ WAVEFORM EDIT

The ability to visualize the complex numerical data resulting from VLSI circuit simulation is critical to testing, understanding & improving these circuits. W-Edit is a waveform viewer that provides ease of use, power & speed in a flexible environment designed for graphical data representation. The advantages of W-Edit include:

1. Tight Integration with T-spice, Tanner EDA_s circuit level simulator. W-Edit can chart data generated by T-spice directly, without modification of the output text data files. The data can also be charted dynamically as it is produced during the simulation.
2. Charts can automatically configure for the type of data being presented.
3. A data is treated by W-Edit as a unit called a trace. Multiple traces from different output files can be viewed simultaneously in single or several windows; traces can be copied and moved between charts & windows. Trace arithmetic can be performed on existed tracing to create new ones.

4. Chart views can be panned back & forth and zoomed in & out, including specifying the exact X-Y coordinate range.

5. Properties of axes, traces, rides, charts, text & colors can be customized.

Numerical data is input to W-Edit in the form of plain or binary text files. Header & Comment information supplied by T-Spice is used for automatic chart configuration. Runtime update of results is made possible by linking W-Edit to a running simulation in T-Spice. W-Edit saves data with chart, trace, axis & environment settings in files with the WDB (W-Edit Database).

➤ **LAYOUT(L-EDIT)**

It is a tool that represents the masks that are used to fabricate an integrated circuit. It describes a layout design in terms of files, cells & mask primitives. On the layout level, the component parameters are totally different from schematic level. So it provides the facility to the user to analyze the response of the circuit before forwarding it to the time consuming & costly process of fabrication. There are rules for designing layout diagram of a schematic circuit using which user can compare the output response with the expected one.

L- Edit: An Integrated Circuit Layout Tool

In L- Edit, layers are associated with masks used in the fabrication process. Different layers can be conveniently represented by different colors and patterns. L- Edit describes a layout design in terms of files, cells, instances, and mask primitives. You may load as many files as desired into memory. A file may be composed of any number of cells. A file may be composed of any number of cells. These cells may be hierarchically related, as in a typical design, or they may be independent, as in a library file. Cells may contain any number or combination of mask primitives and instances of other cells.

Cells: The Basic Building Blocks

The basic building block of the integrated circuit design in L- Edit is a cell. Design layout occurs within cells. A cell can:

- ❖ Contain part or all of the entire design.
- ❖ Be referenced in other cells as a sub- cell, or instance.
- ❖ Be made up entirely of instances of other cells.
- ❖ Contain original drawn objects, or primitives.
- ❖ Be made up entirely of primitives or a combination of primitives and instances of other cells.

Hierarchy

L- Edit supports fully hierarchical mask design. Cells may contain instances of other cells. An instance is a reference to a cell; should you edit the instanced cell, the change is reflected in all the instances of that cell. Instances simplify the process of updating a design, and also reduce data storage requirements, because an instance does not need to store all the data within the instanced cell instead, only a reference to the instanced cell is stored, along with information on the position of the instance and on how the instance may be rotated and mirrored.

L- Edit does not use a “separated” hierarchy: instances and primitives may coexist in the same cell at any level in the hierarchy. Design files are self- contained. The pointer to a cell contained in an instance always points to a cell within the same design file. When cells are copied from one file to another, L- Edit automatically copies across any cells that are instanced by the copied cell, to maintain the self- contained nature of the destination file.

Design Rules

Manufacturing constraints can be defined in L- Edit as design rules. Layouts can be checked against these design rules.

Design Features

L- Edit is a full- custom mask editor. Manual layout can be accomplished more quickly because of L Edit’s intuitive user interface. In addition, one can construct special structures to utilize a technology without, worrying about problems caused by automatic transformations. Phototransistors, guard bars, vertical and horizontal bipolar transistors, static structures, and Schottky diodes, for example, are as easy to design in CMOS- Bulk technology as are conventional MOS transistors.

Floor plans

L- Edit is a manual floor planning tool. You have the choice of displaying instances in outline, identified only by name, or as fully fleshed- out mask geometry. When you display your design in outline, you can manipulate the arrangement of the cells in your design quickly and easily to achieve the desired floor plan. One can manipulate instances at any level in the hierarchy, with insides hidden or displayed, using the same graphical move/ select operations or rotation/ mirror commands that you use on primitive mask geometry.

Memory Limits

In L- Edit, one can make your design files as large as one like, given available RAM and disk space.

Hard Copy

L- Edit provides the capability to print hard copy of the design. A multiage option allows very large plots to

be printed to a specific scale on multiple 8 1/2 x 11 inch pages. An L- Edit macro is available to support large- format, high- resolution, color plotting on inkjet plotters.

Variable Grid

L- Edit's grid options support lambda- based design as well as micron- based and mil- based design.

Error Recovery

L- Edit's error- trapping mechanism catches system errors and in most cases provides a means to recover without losing or damaging data.

L- Edit Modules

- ❖ L- Edit TM: a layout editor
- ❖ L- Edit α Extract TM: a layout extractor
- ❖ L- Edit α DRC TM: a design rule checker

L- Edit is a full- featured, high-performance, interactive, graphical mask layout editor. L- Edit generates layouts quickly and easily, supports fully hierarchical designs, and allows an unlimited number of layers, cells, and levels of hierarchy. It includes all major drawing primitives and supports 90°, 45°, and all- angle drawing modes.

L- Edit α Extract creates SPICE- compatible circuit netlists from L- Edit layouts. It can recognize active and passive devices, sub circuits, and the most common device parameters, including resistance, capacitance, device length, width, and area, and device source and drain area.

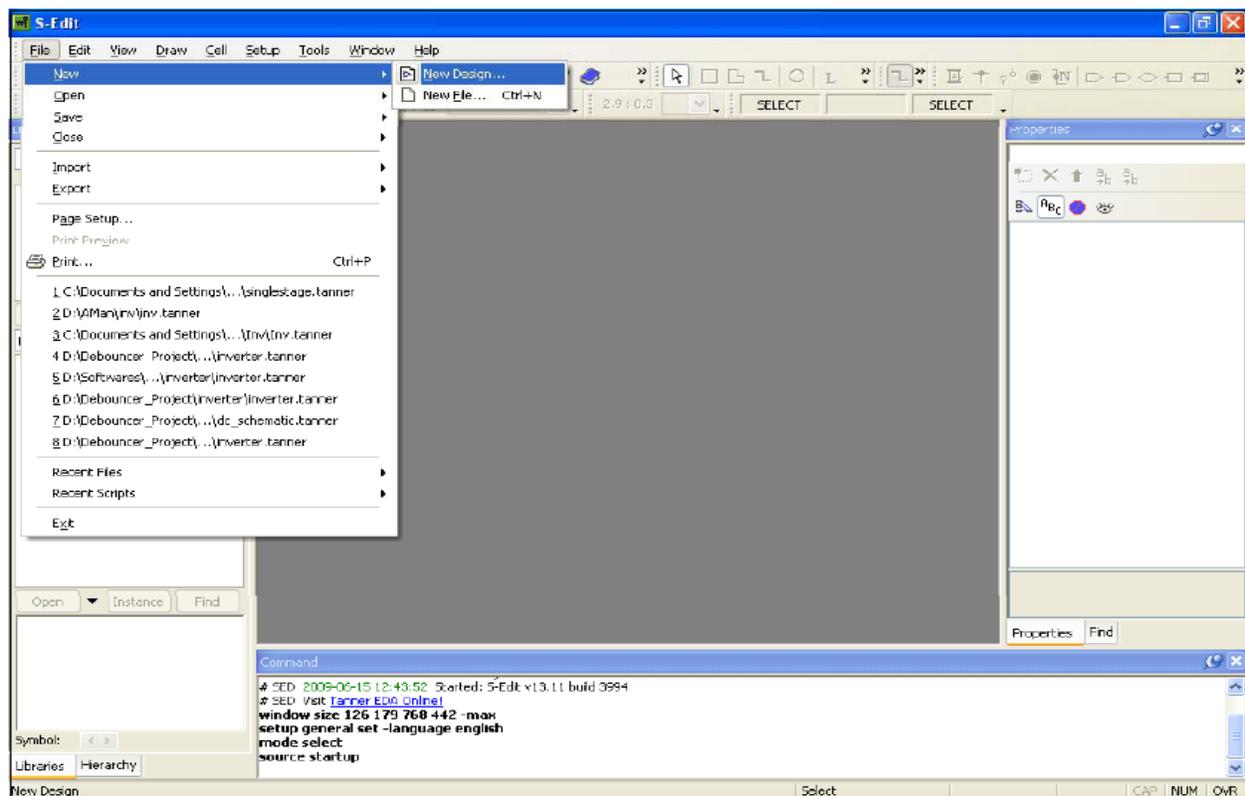
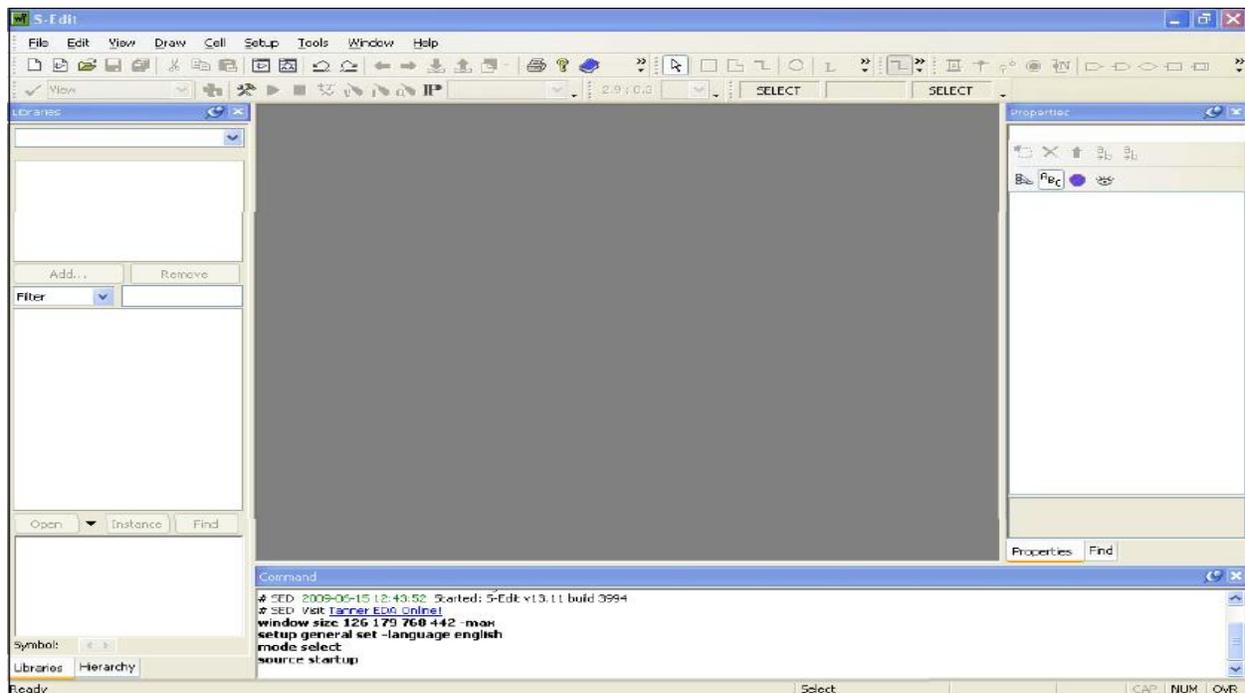
L- Edit α DRC features user- programmable rules and handles minimum width, exact width, minimum space, minimum surround, non- exist, overlap, and extension rules. It can handle full chip and region- only DRC. DRC offers Error Browser and Object Browser functions for quickly and easily cycling through rule- checking errors.

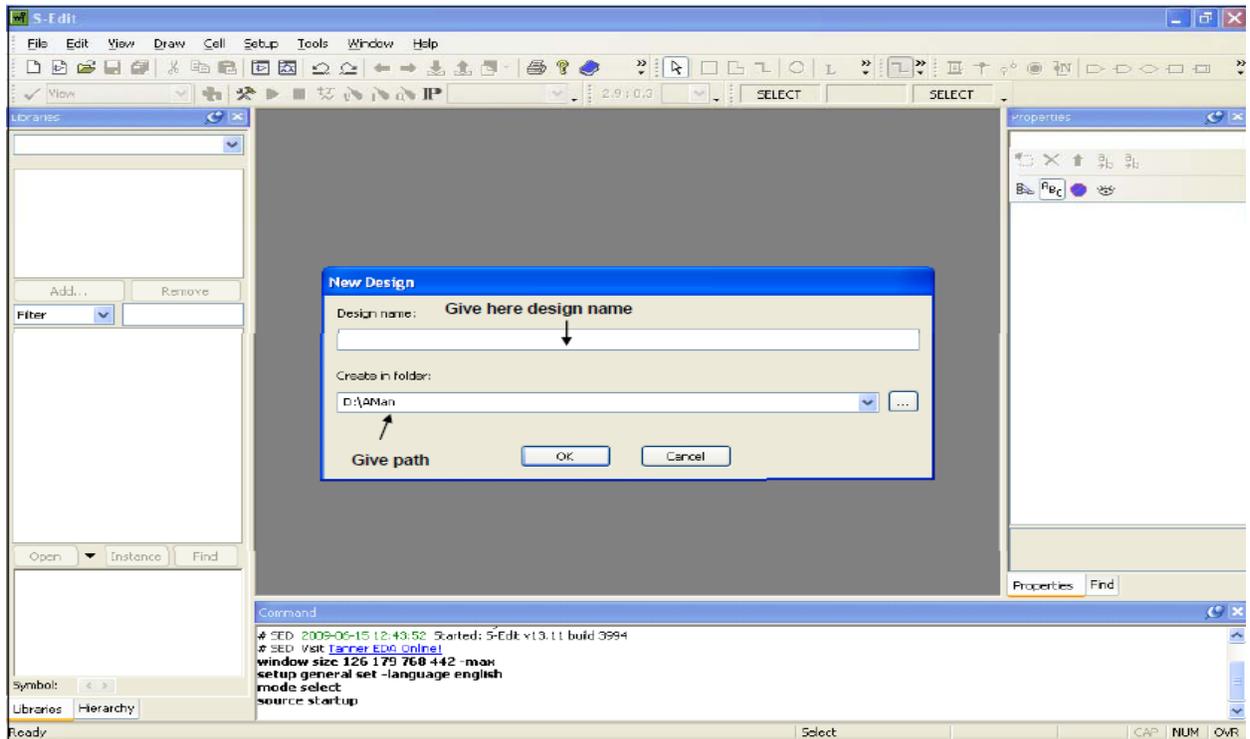
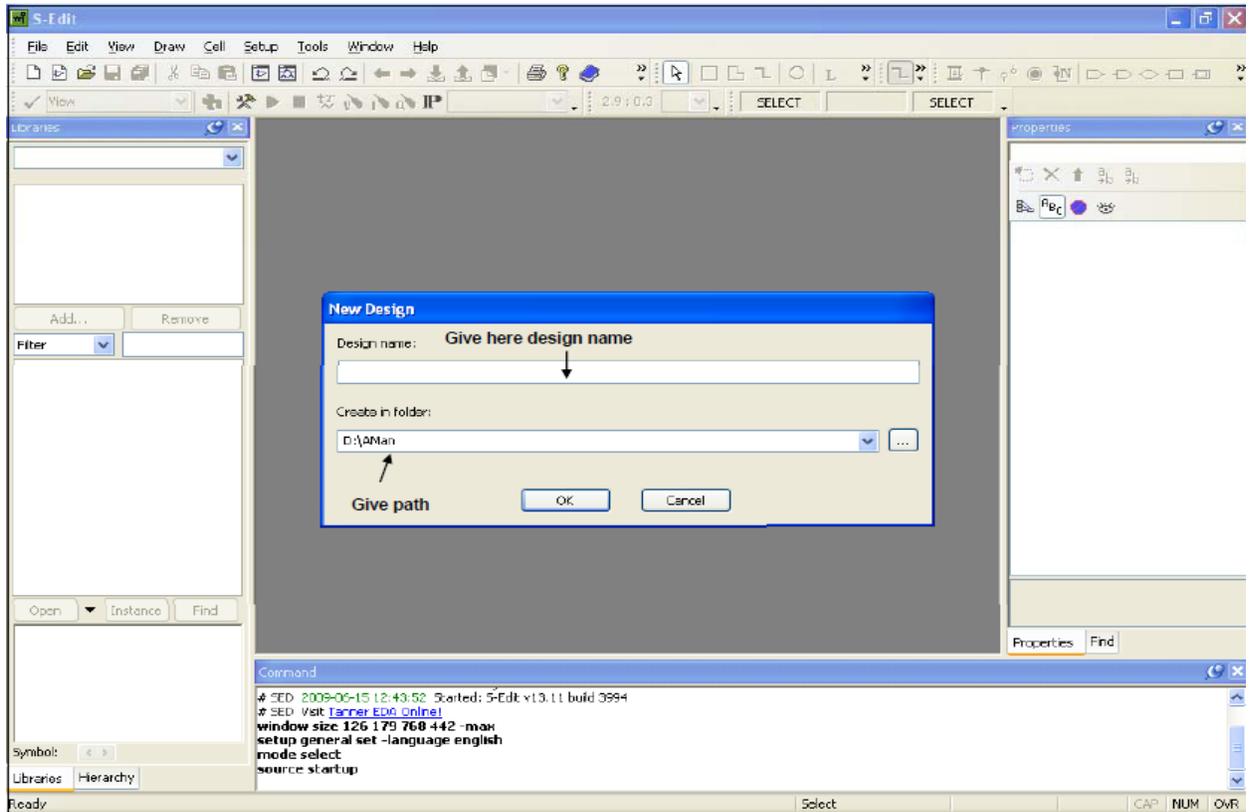
Steps to use Tanner tool:

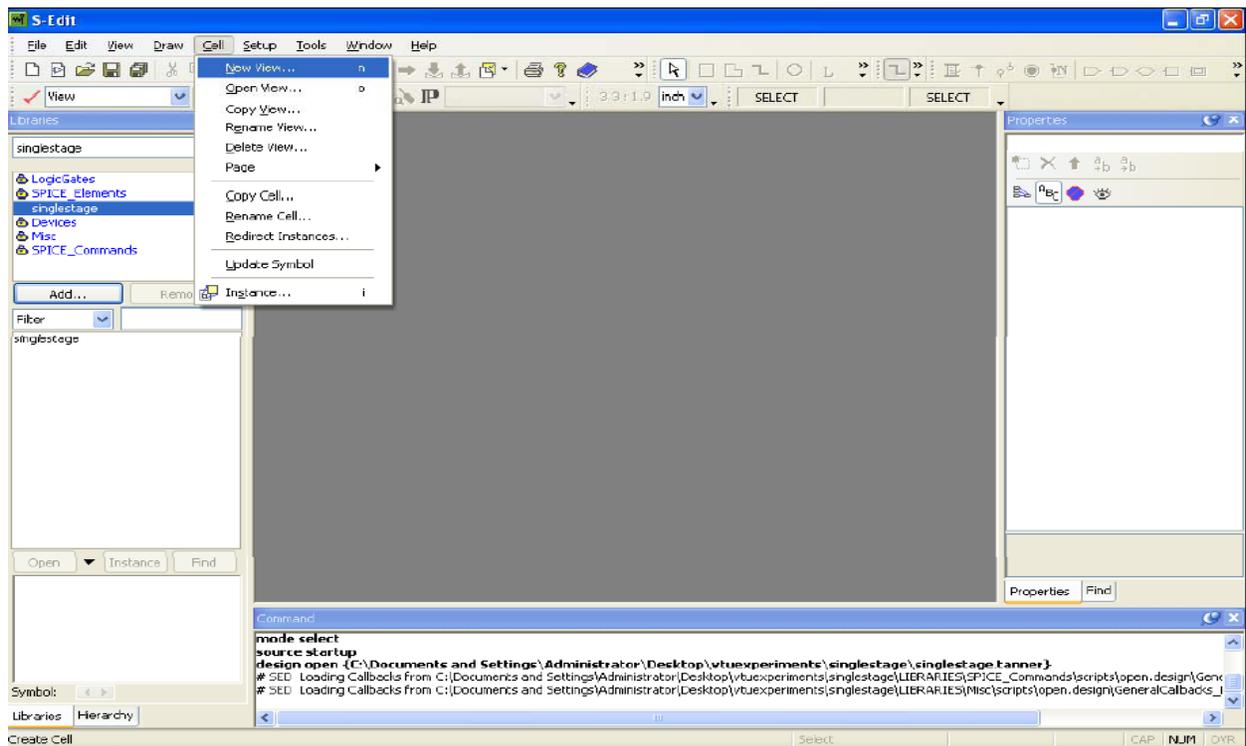
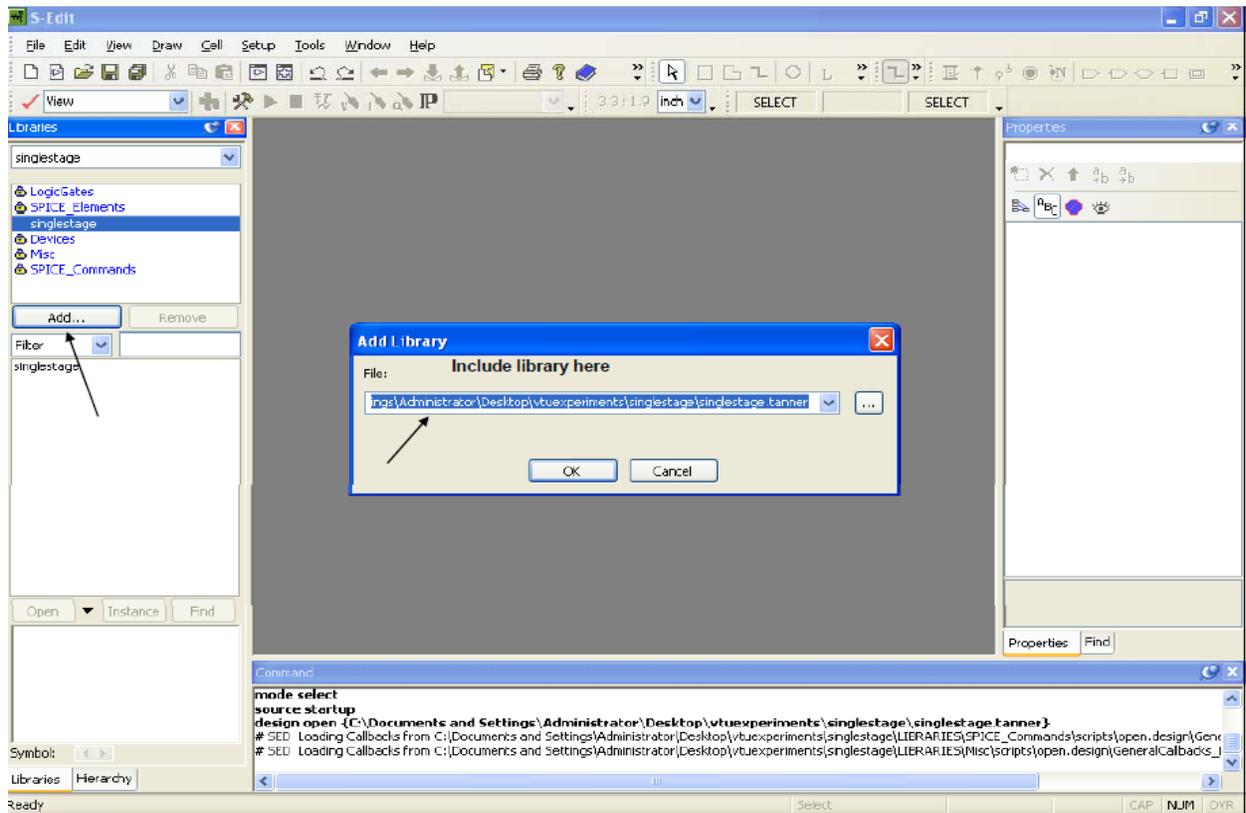
i.SCHEMATIC (S-edit):

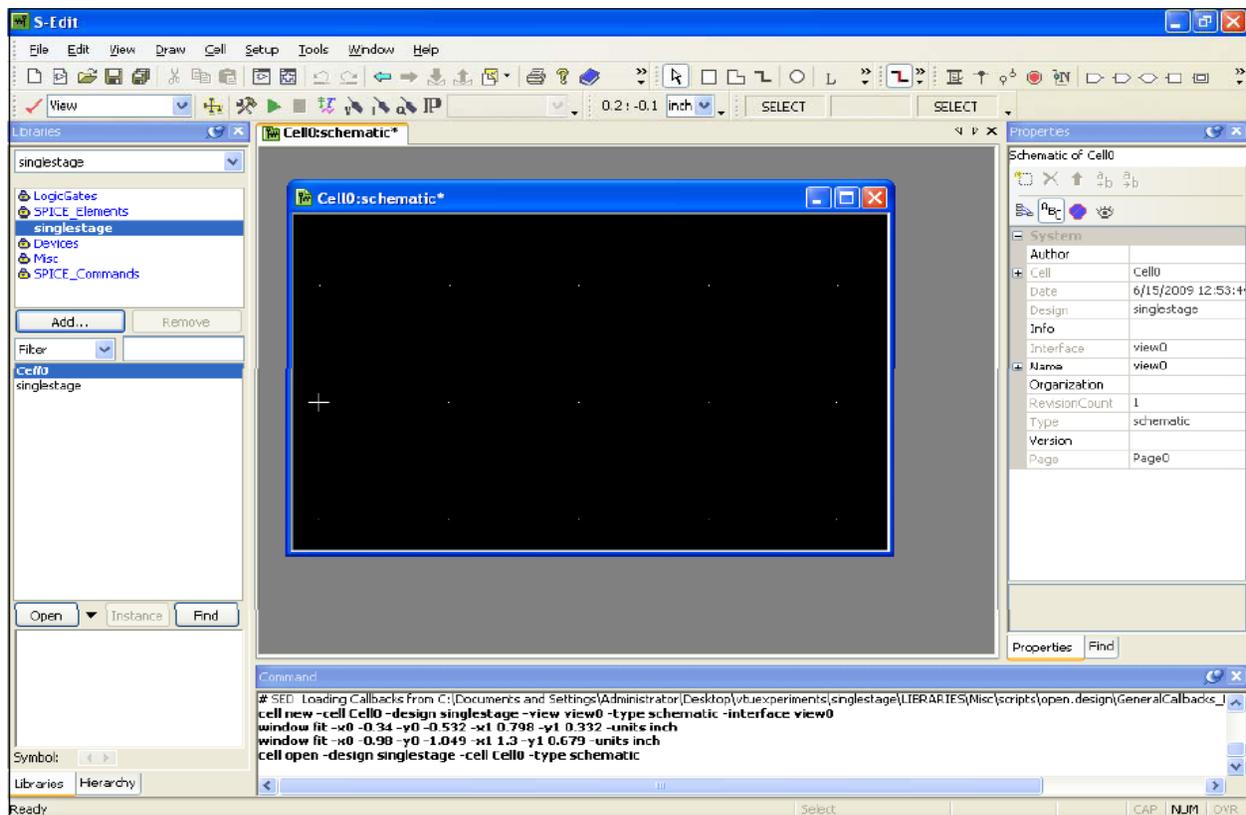
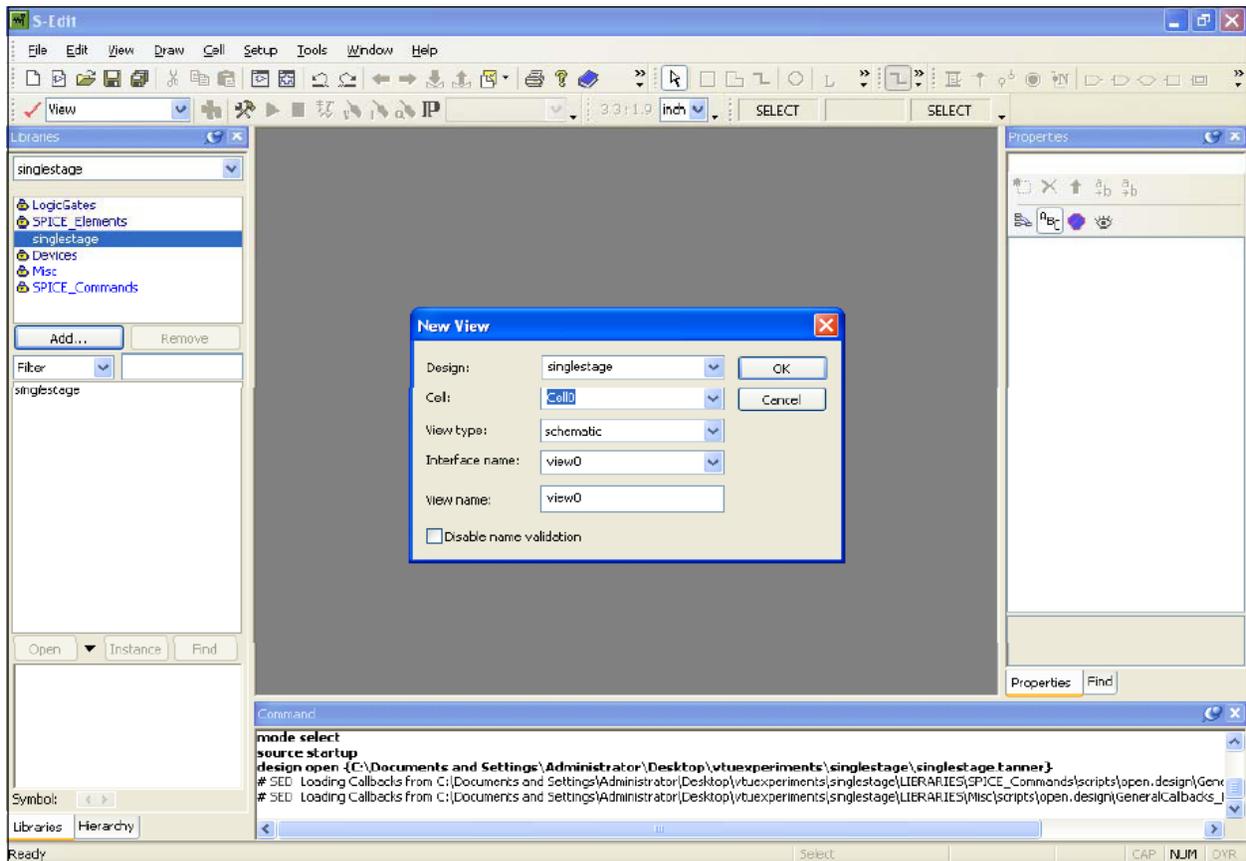
Start the tanner EDA by using the desktop shortcut or by using the

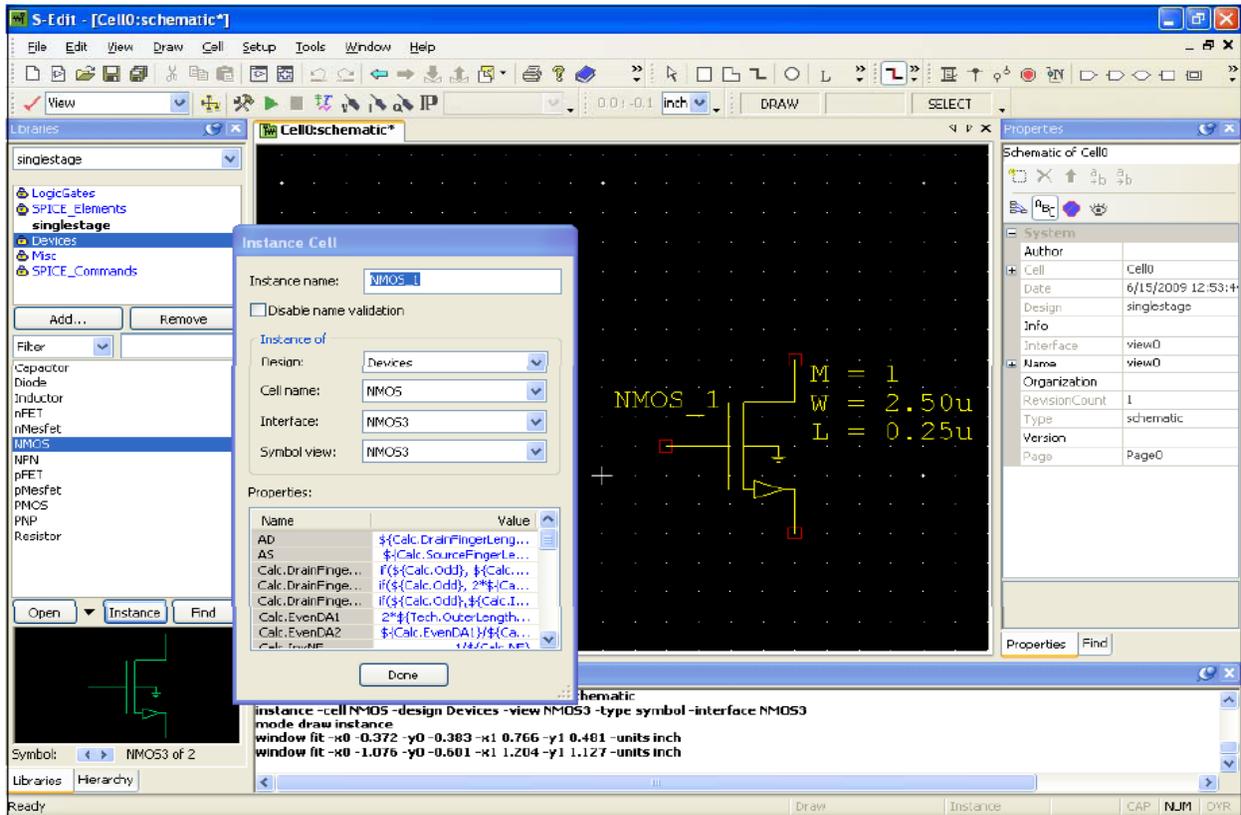
- Start → Programs → tanner EDA → tanner tool v13.0 → S-edit.



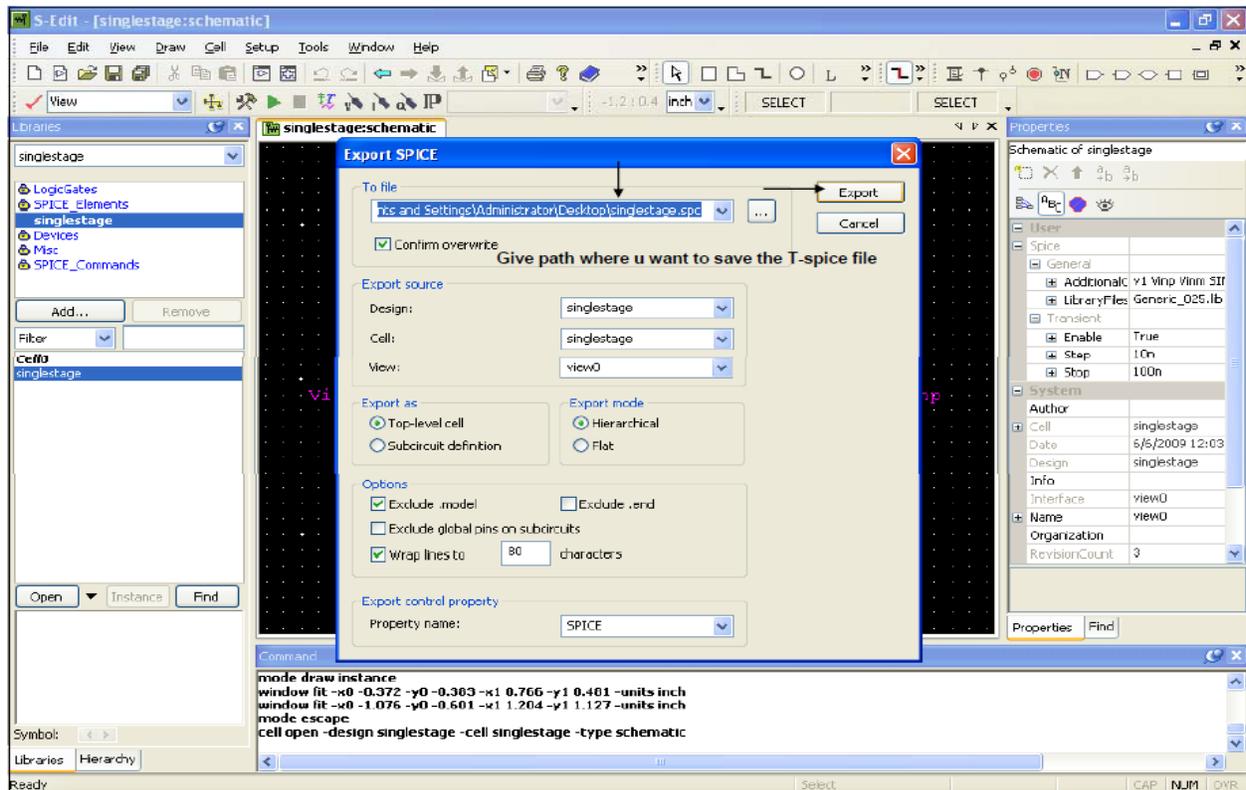
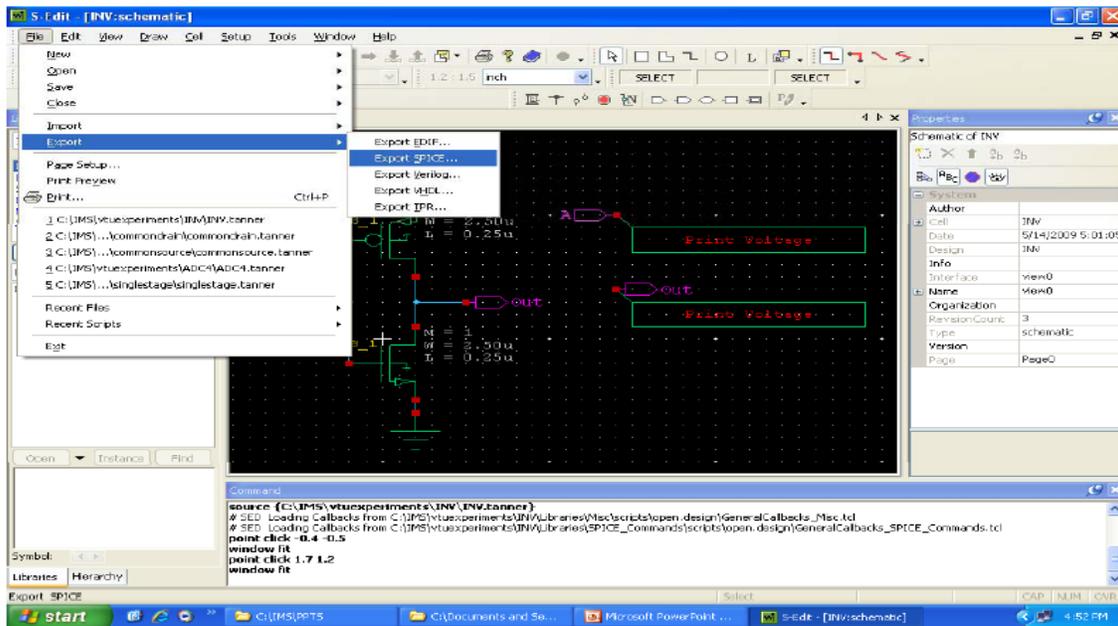


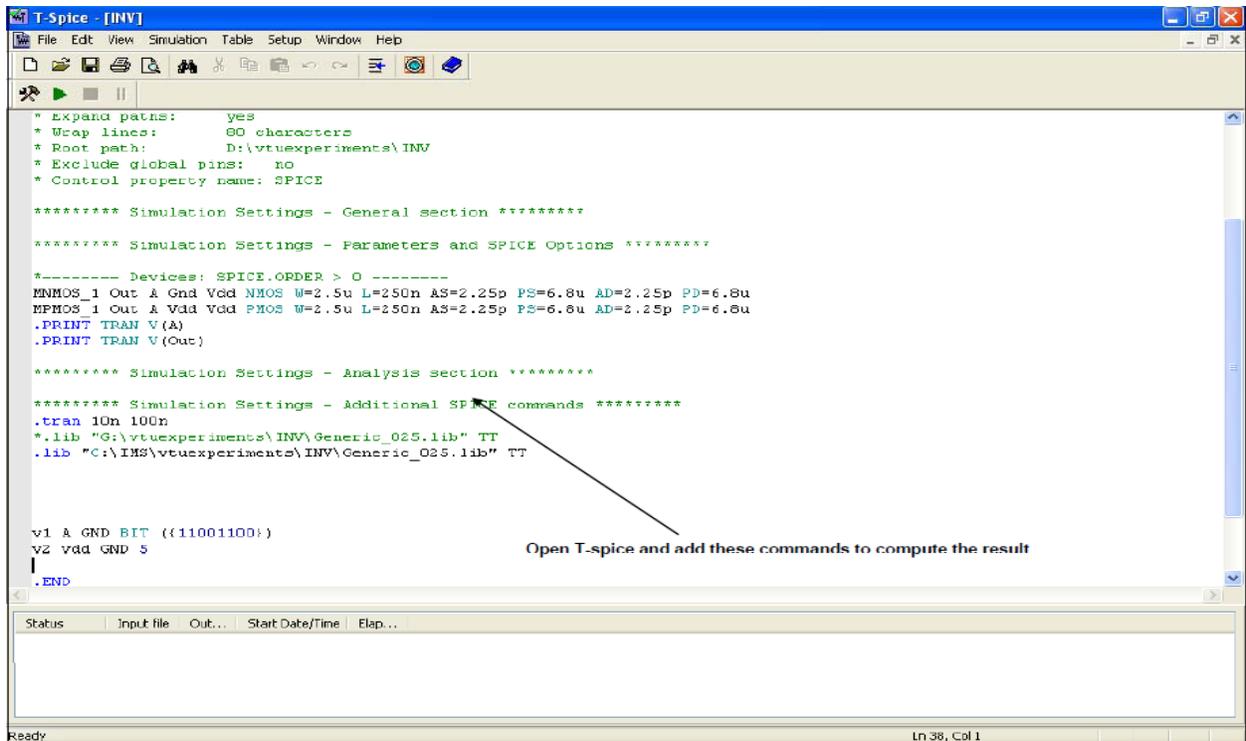
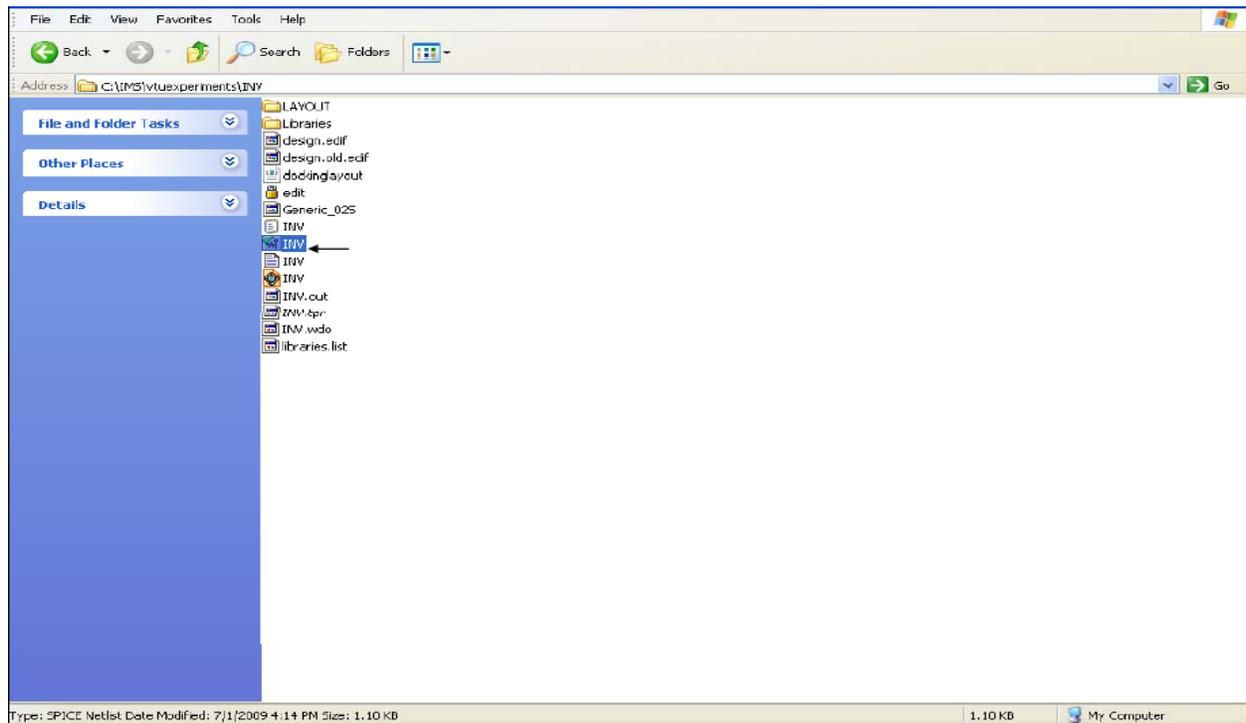






After completion of the design just export T-spice





```

T-Spice - [INV]
File Edit View Simulation Table Setup Window Help
After adding the commands just click on run
* Parameters: yes
* Run simulation: 60 characters
* Root path: D:\vtuexperiments\INV
* Exclude global pins: no
* Control property name: SPICE

***** Simulation Settings - General section *****

***** Simulation Settings - Parameters and SPICE Options *****

*----- Devices: SPICE.ORDER > 0 -----
NMOS_1 Out A Gnd Vdd NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u
PMOS_1 Out A Vdd Vdd PMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u
.PRINT TRAN V(A)
.PRINT TRAN V(Out)

***** Simulation Settings - Analysis section *****

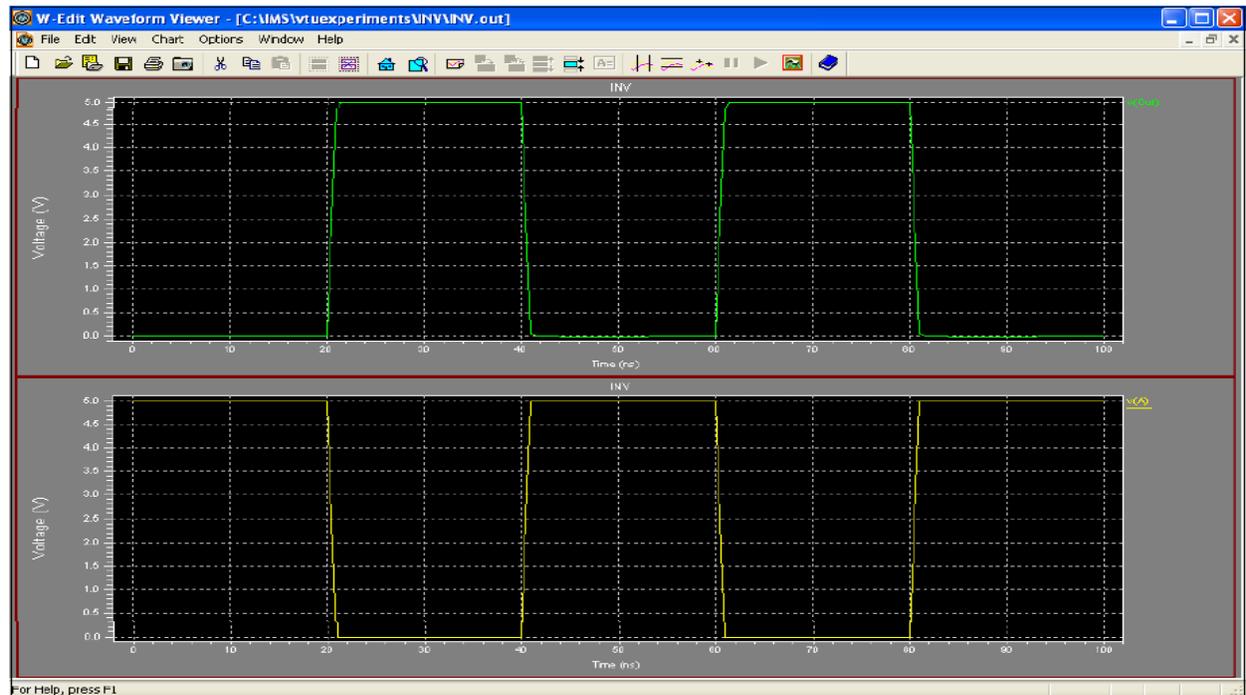
***** Simulation Settings - Additional SPICE commands *****
.tran 10n 100n
*.lib "G:\vtuexperiments\INV\Generic_025.lib" TT
.lib "C:\IMS\vtuexperiments\INV\Generic_025.lib" TT

v1 A GND BIT ((11001100))
v2 vdd GND 5
.END

Status Input file Out... Start Date/Time Elap...

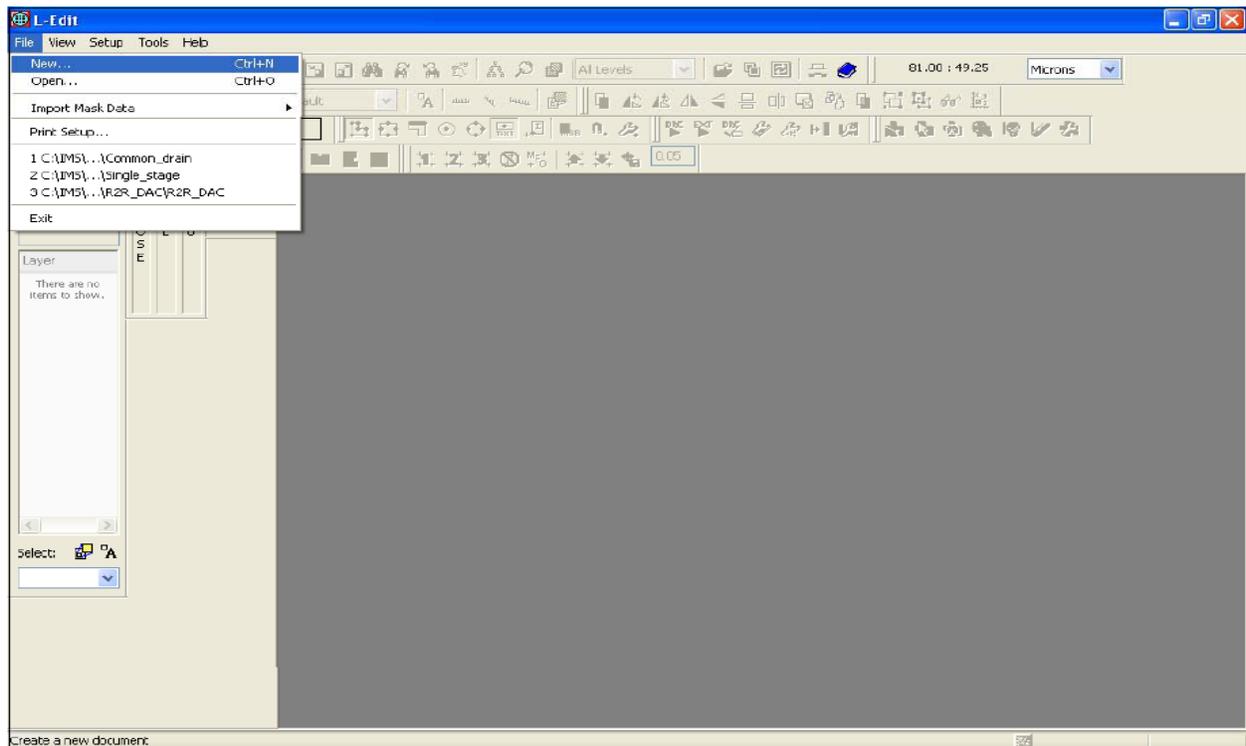
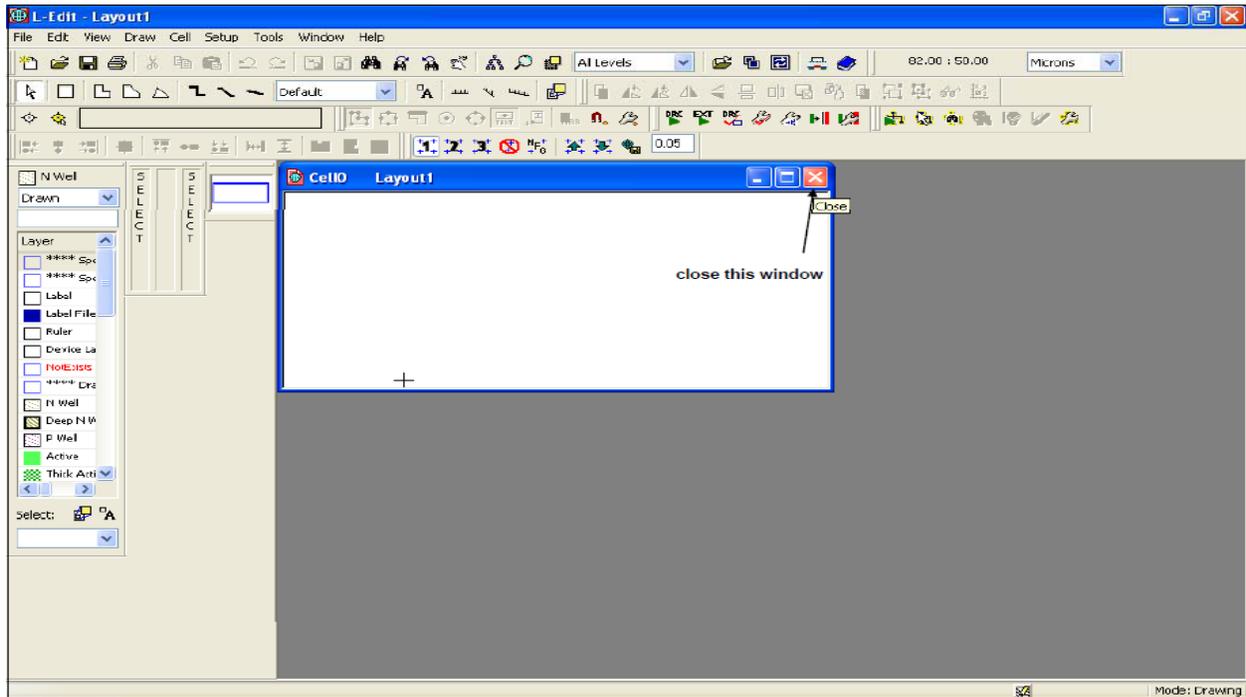
Run a simulation Ln 38, Col 1

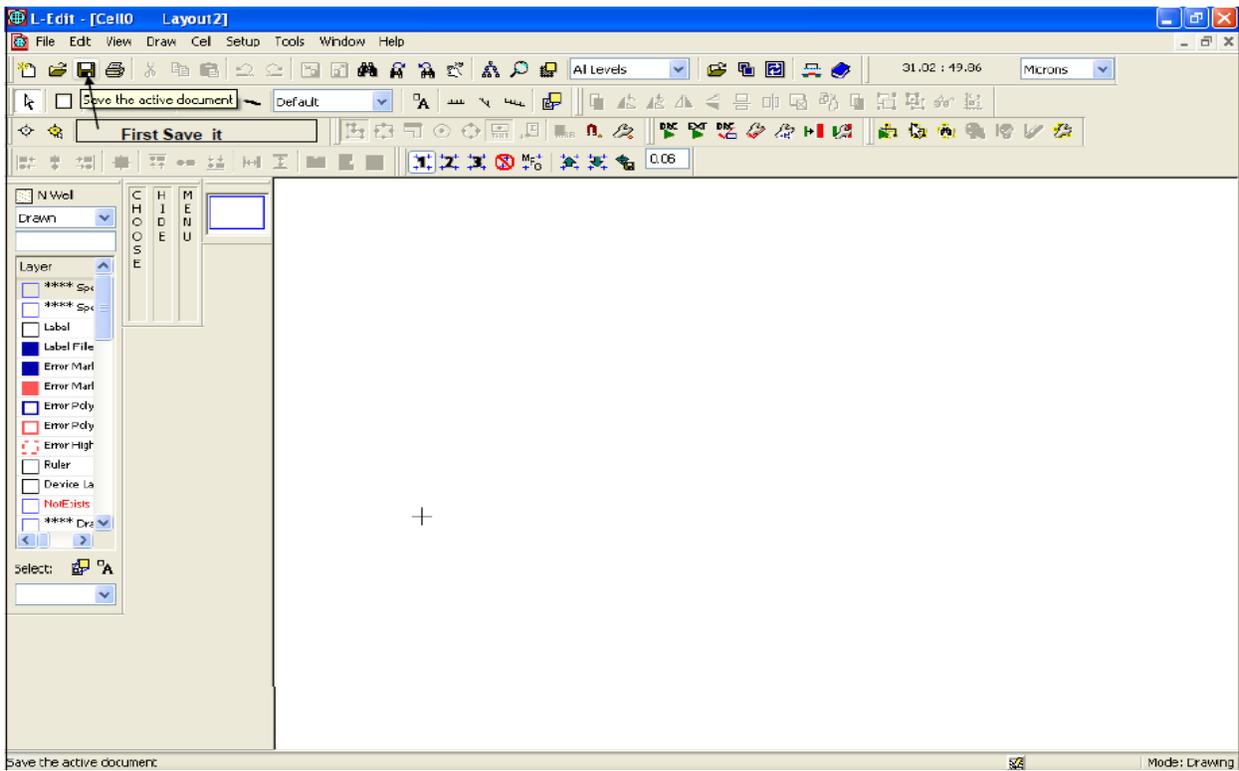
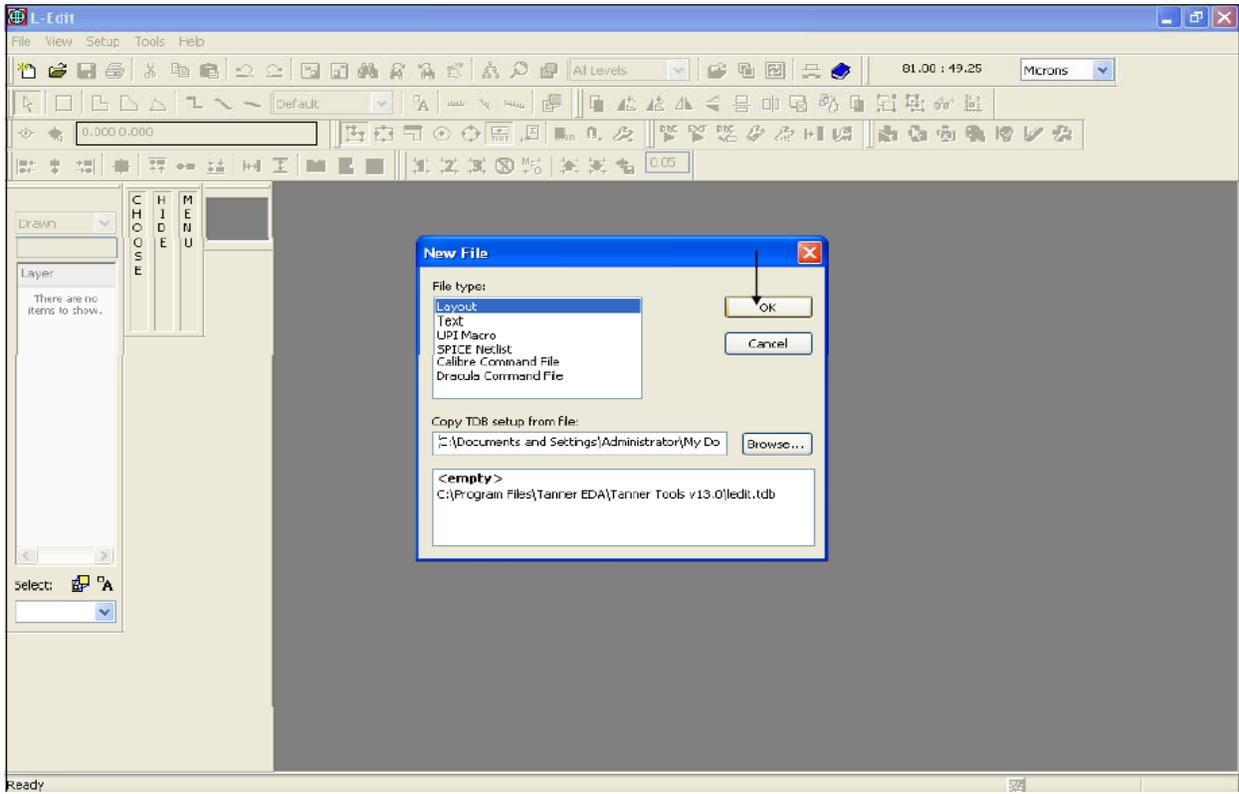
```

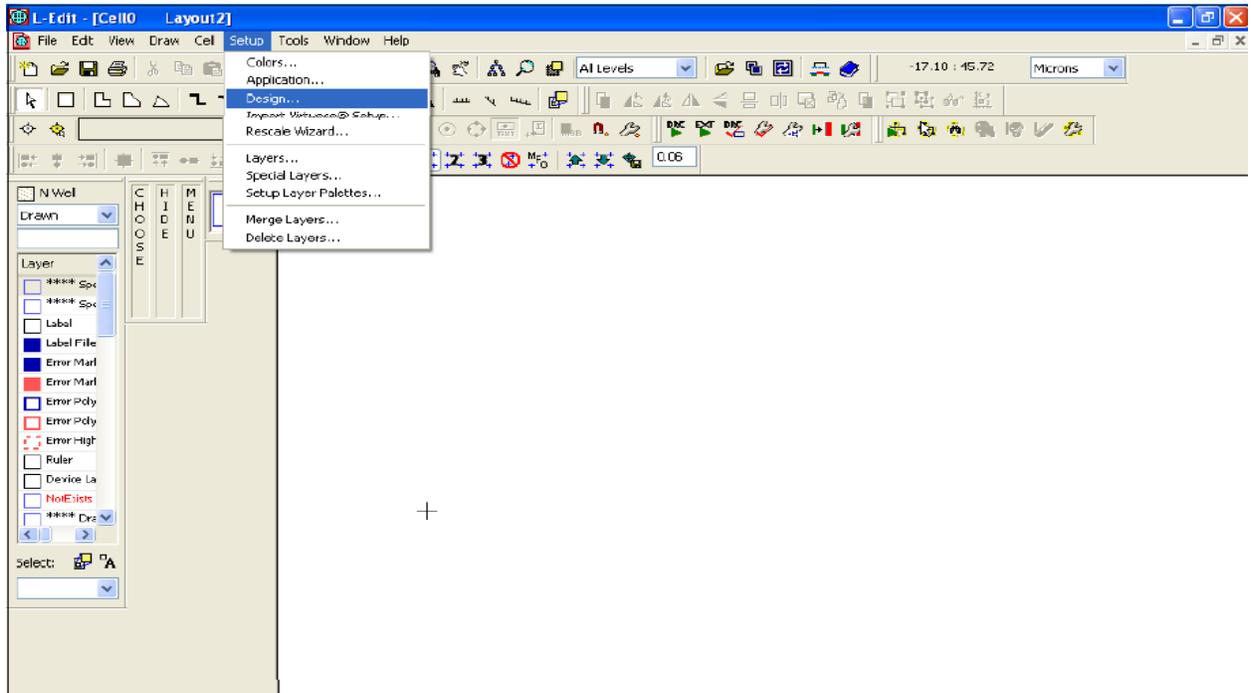
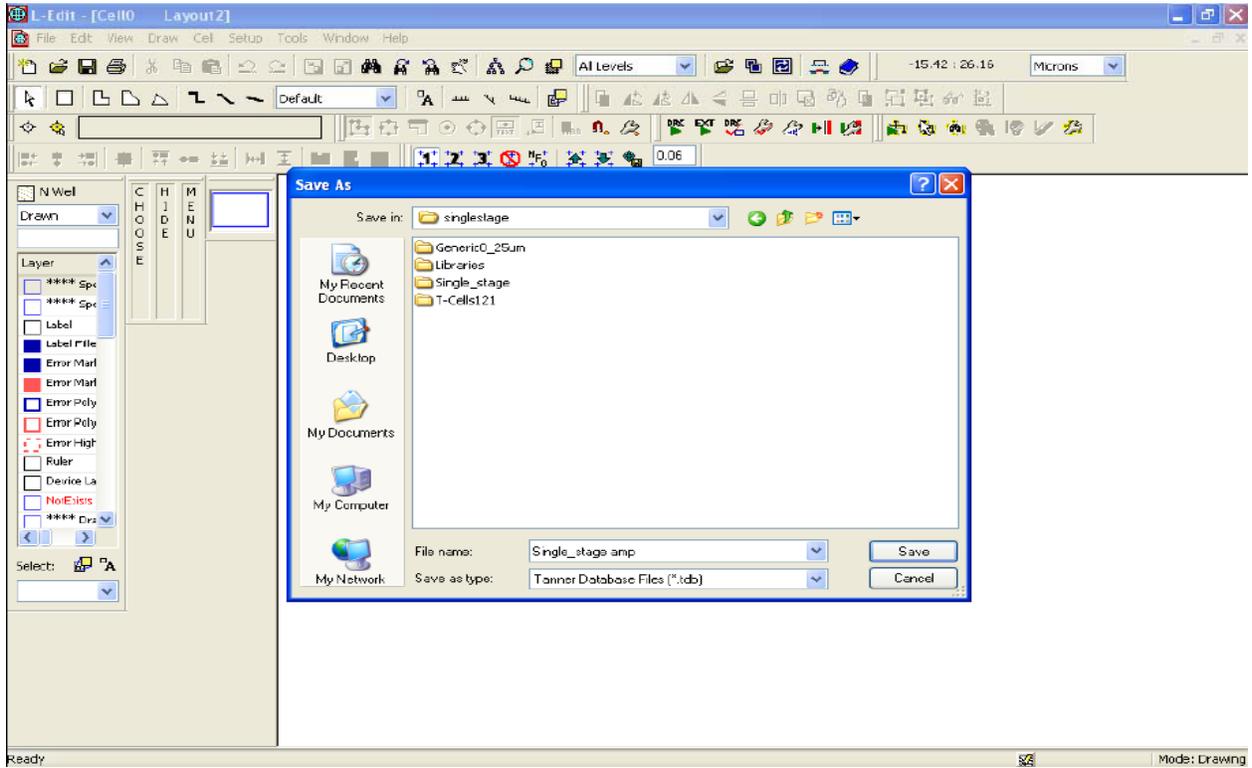


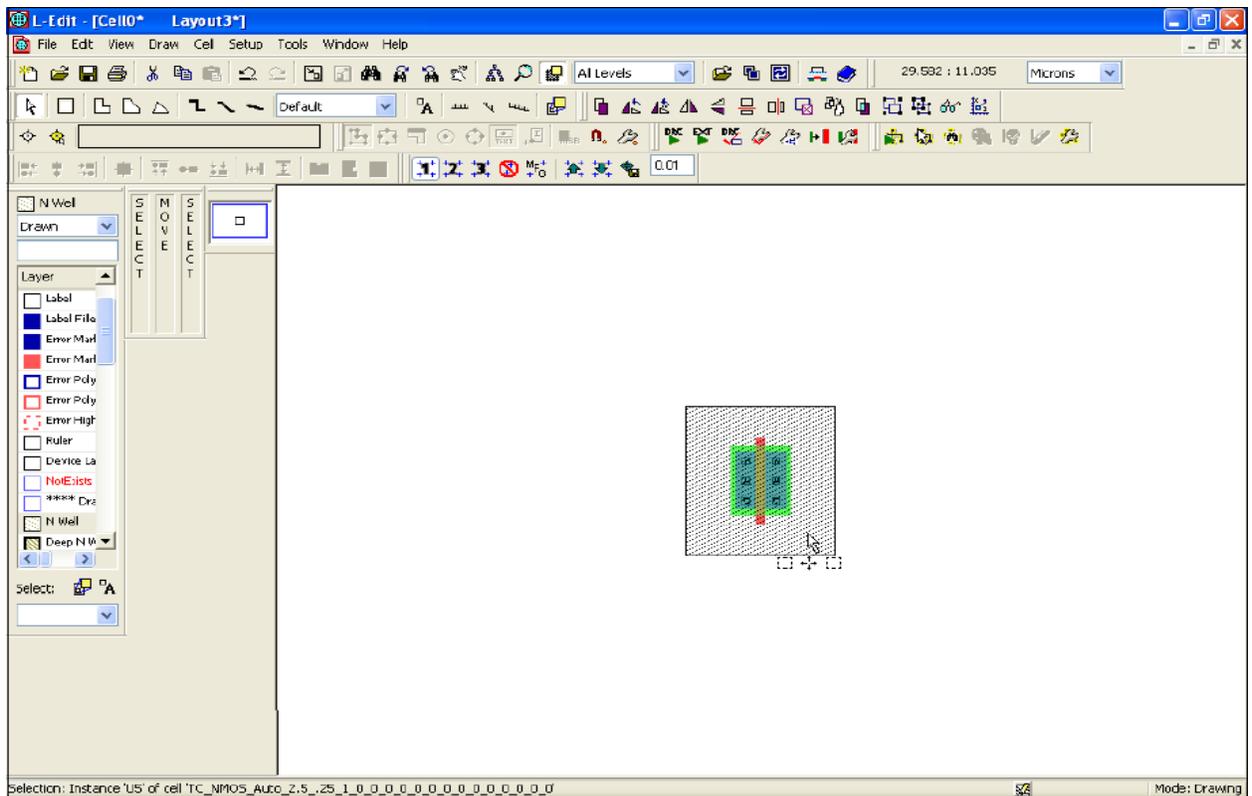
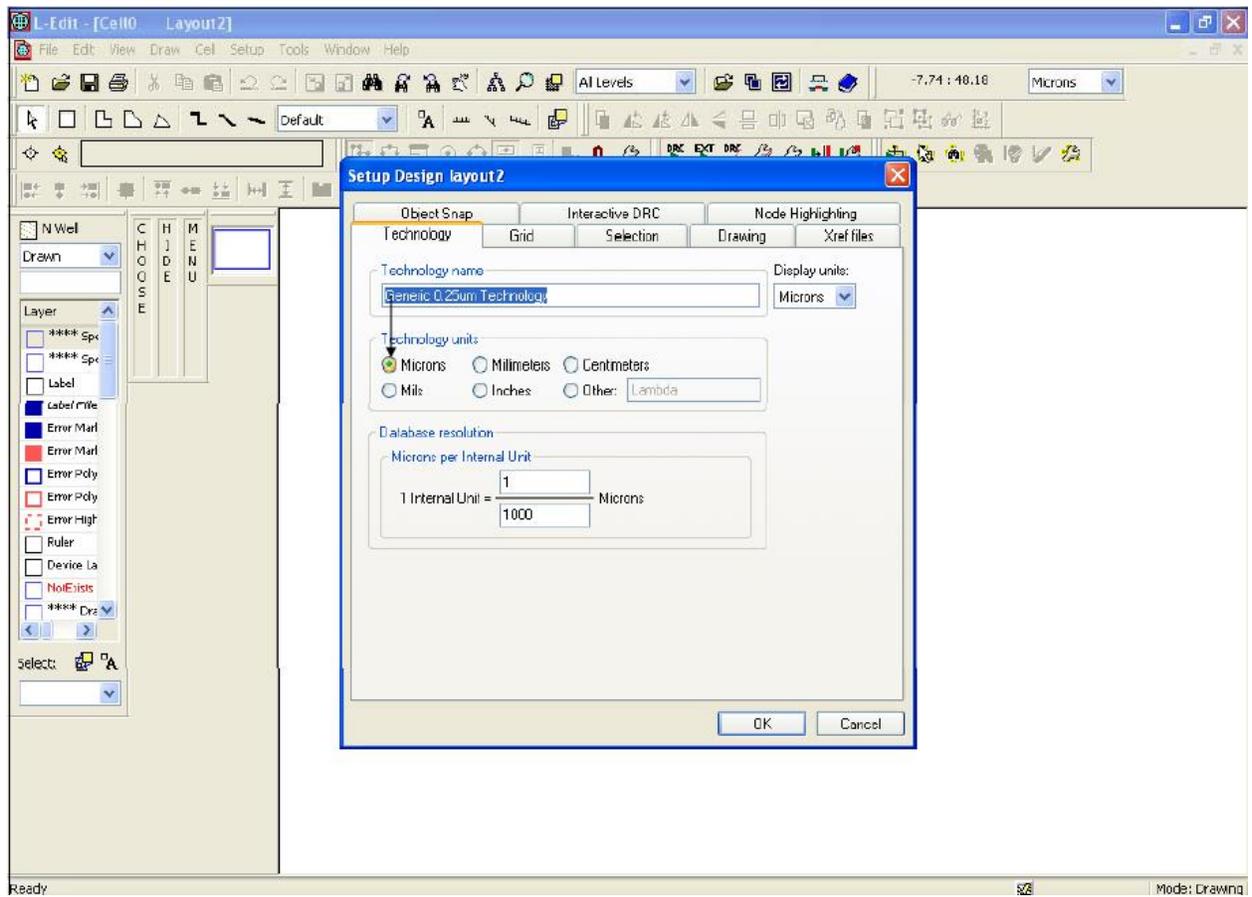
ii) Layout (L-edit):

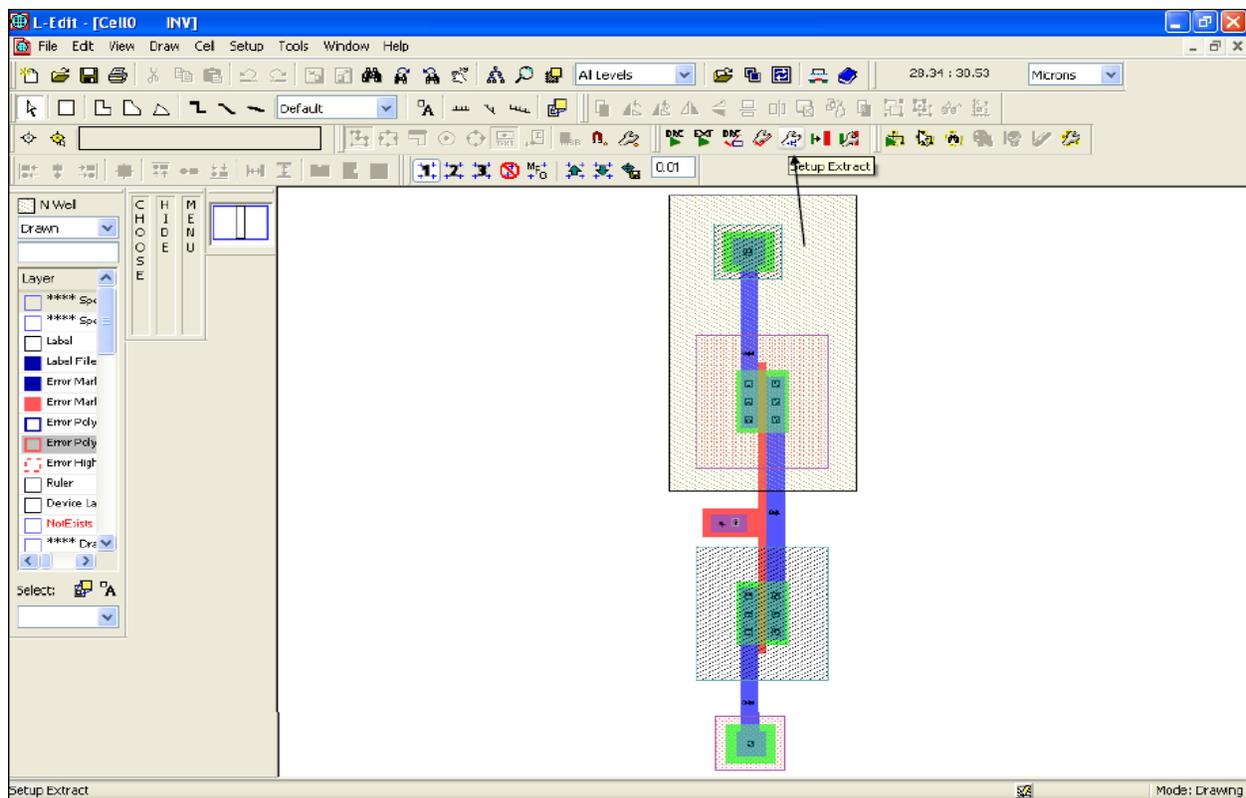
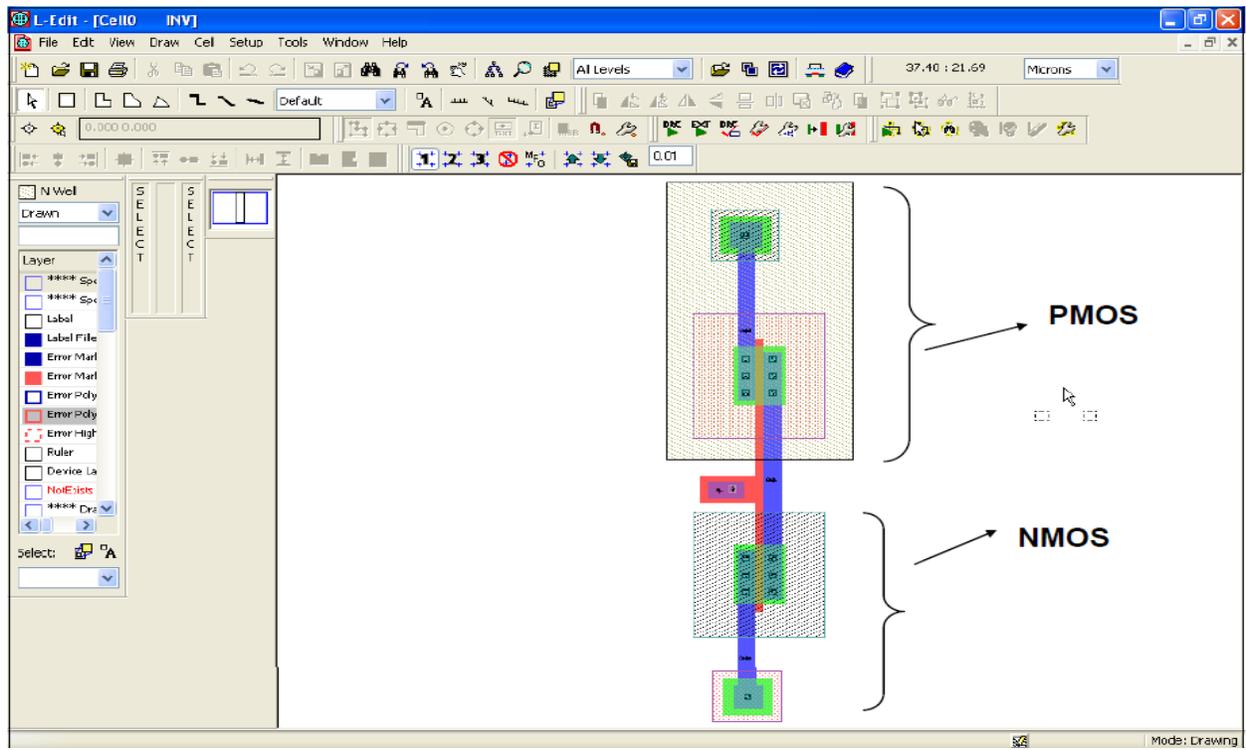
Start → Programs → tanner EDA → tanner tool v13.0 → L-edit

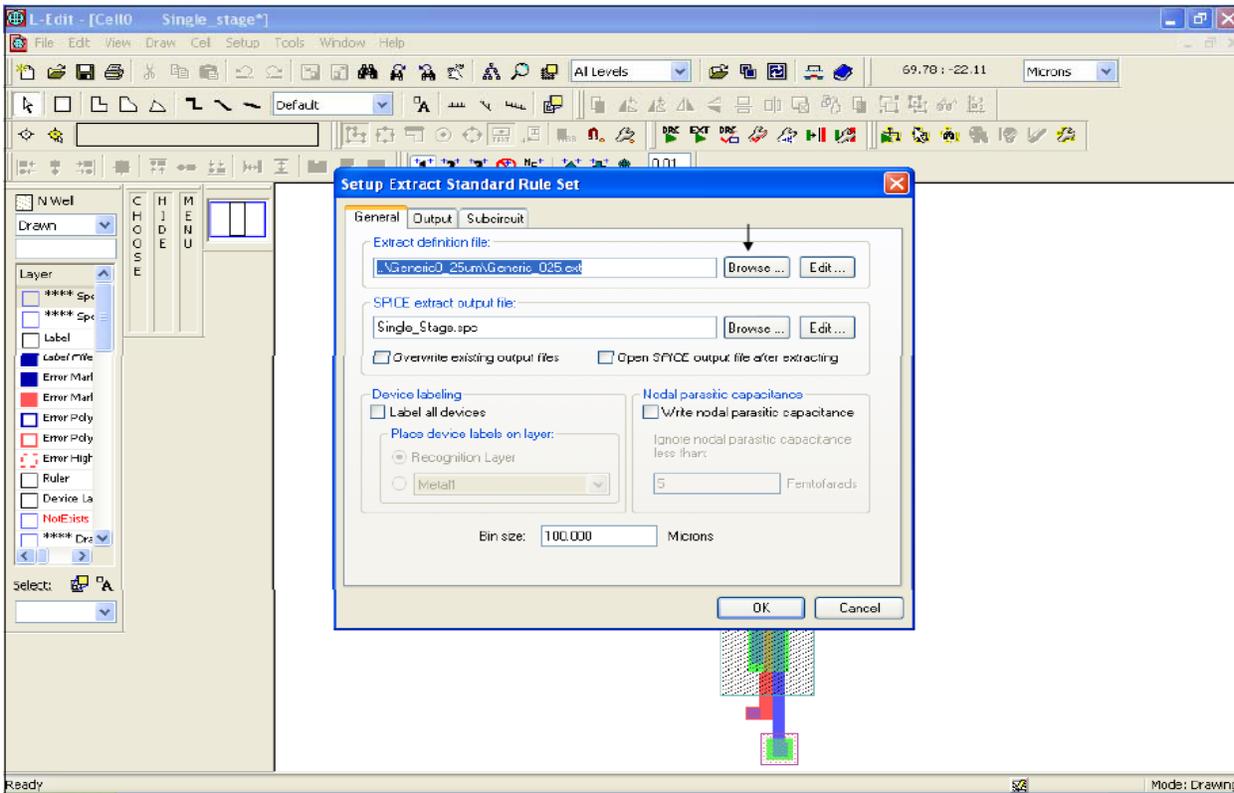
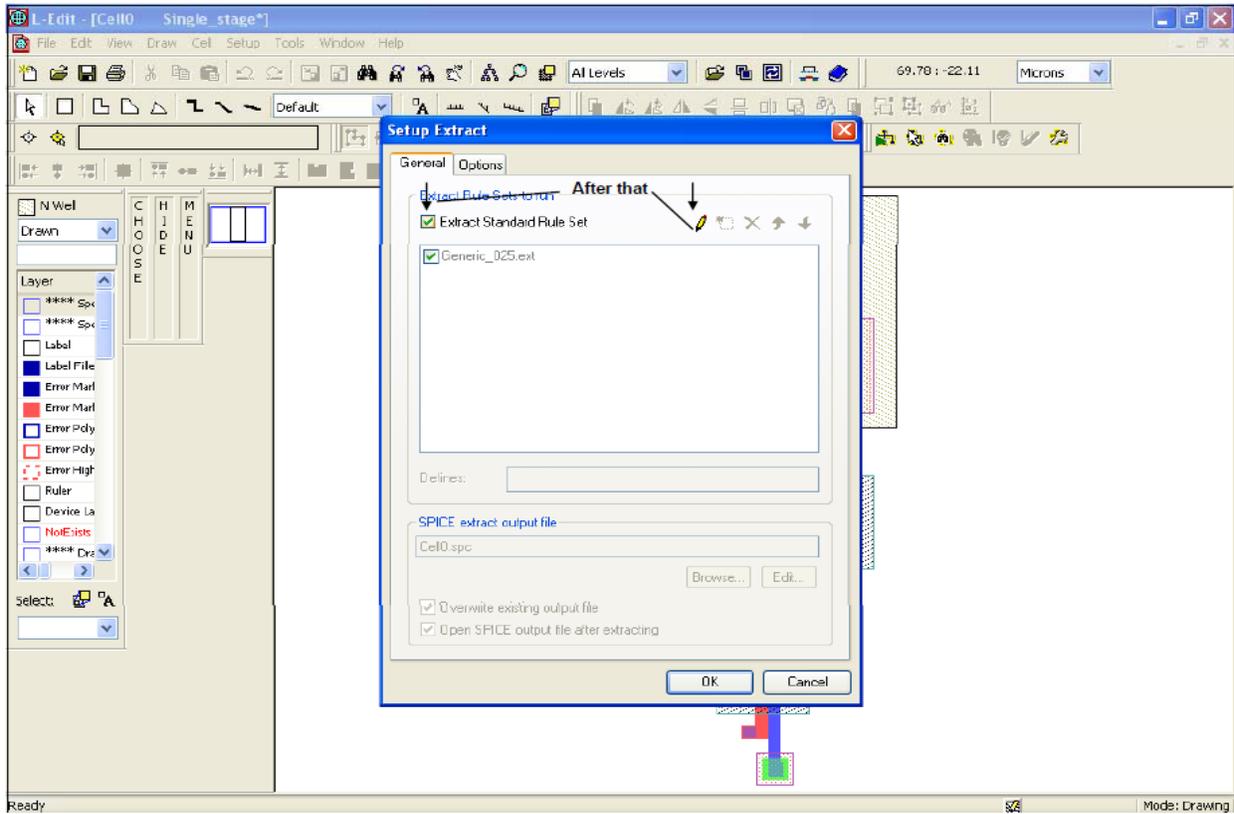


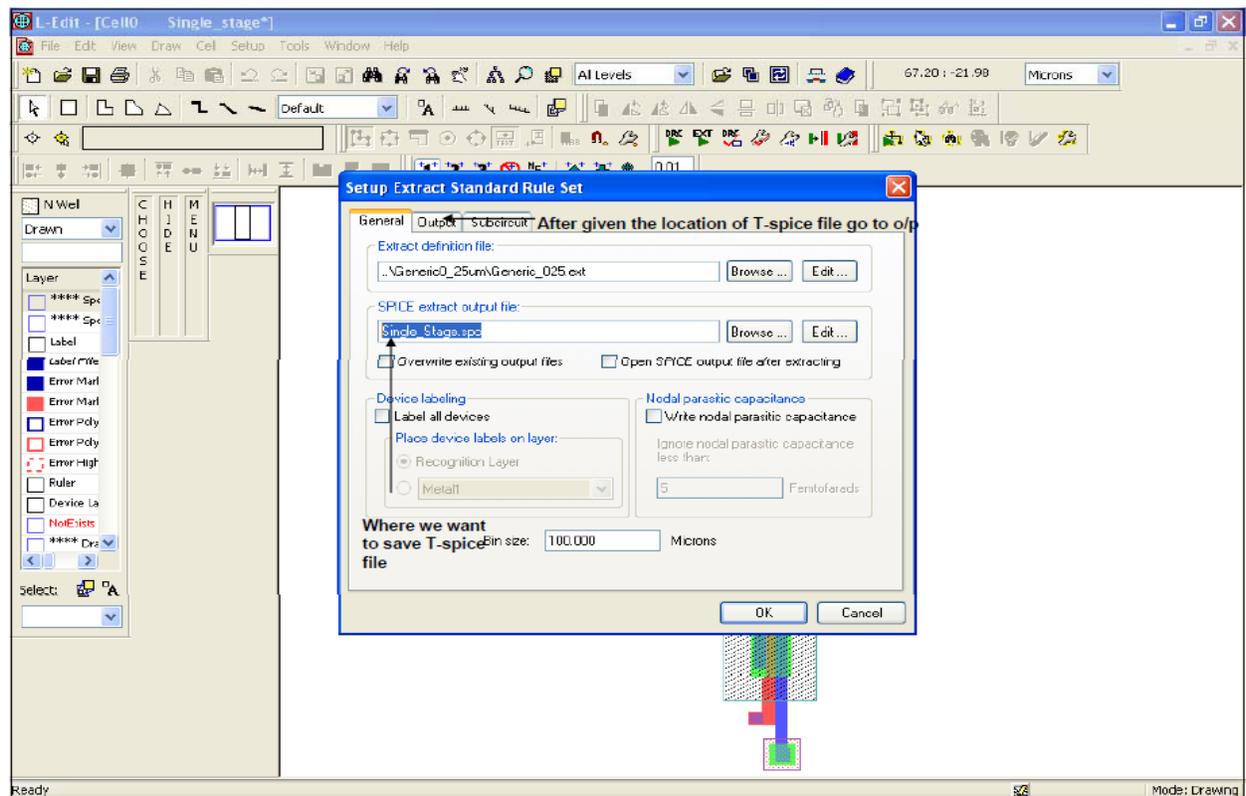
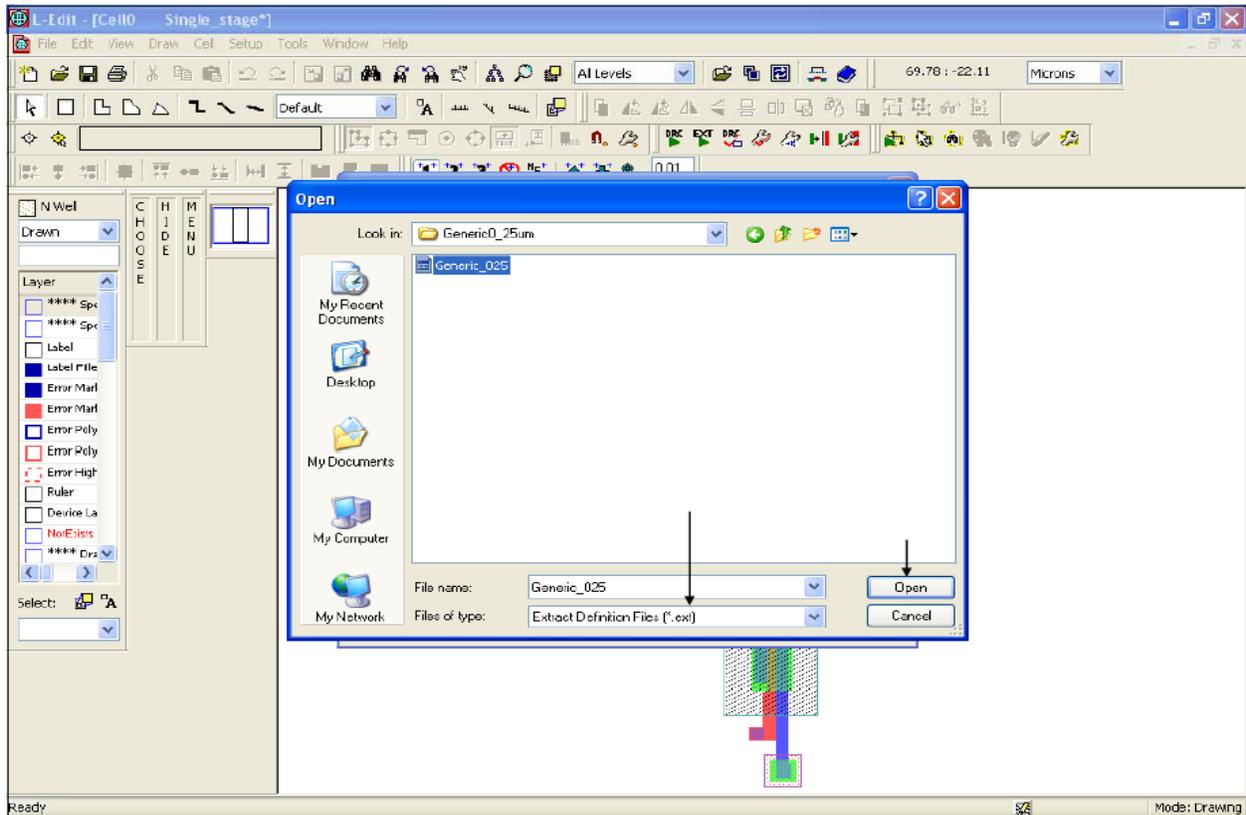


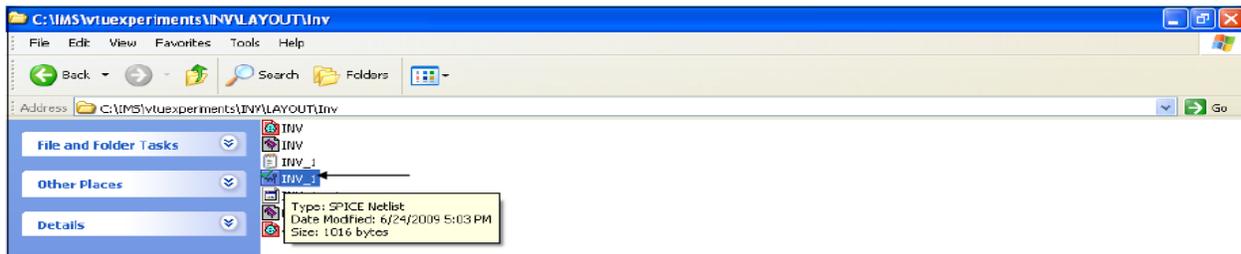
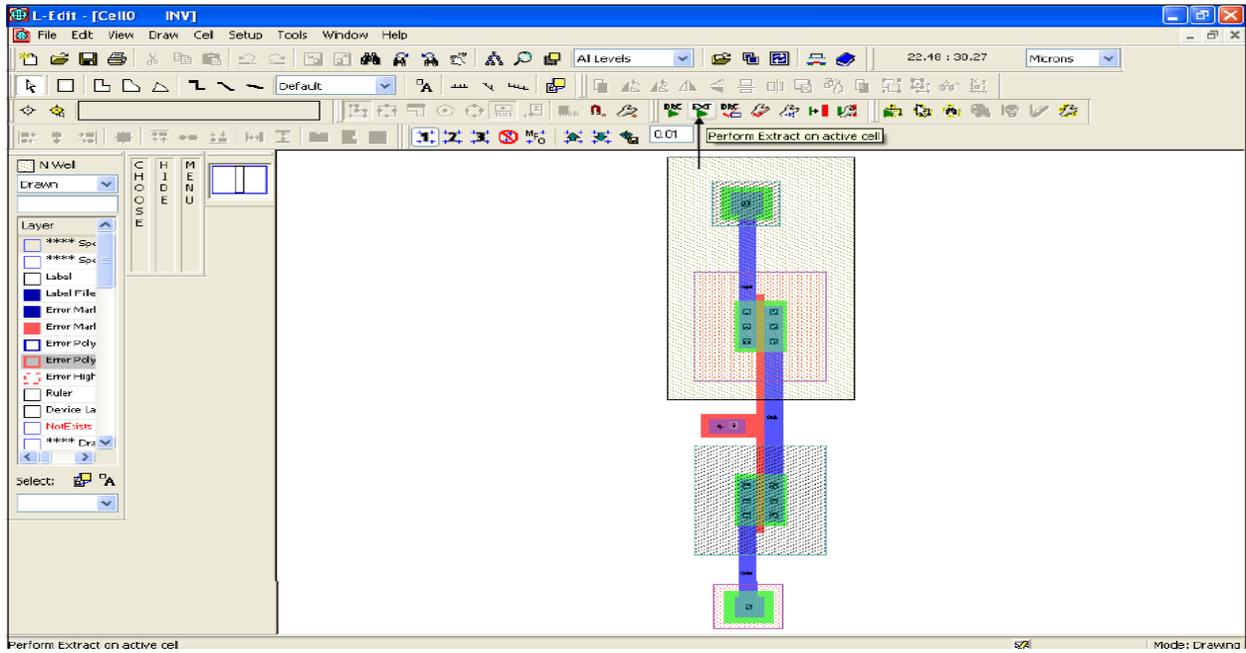
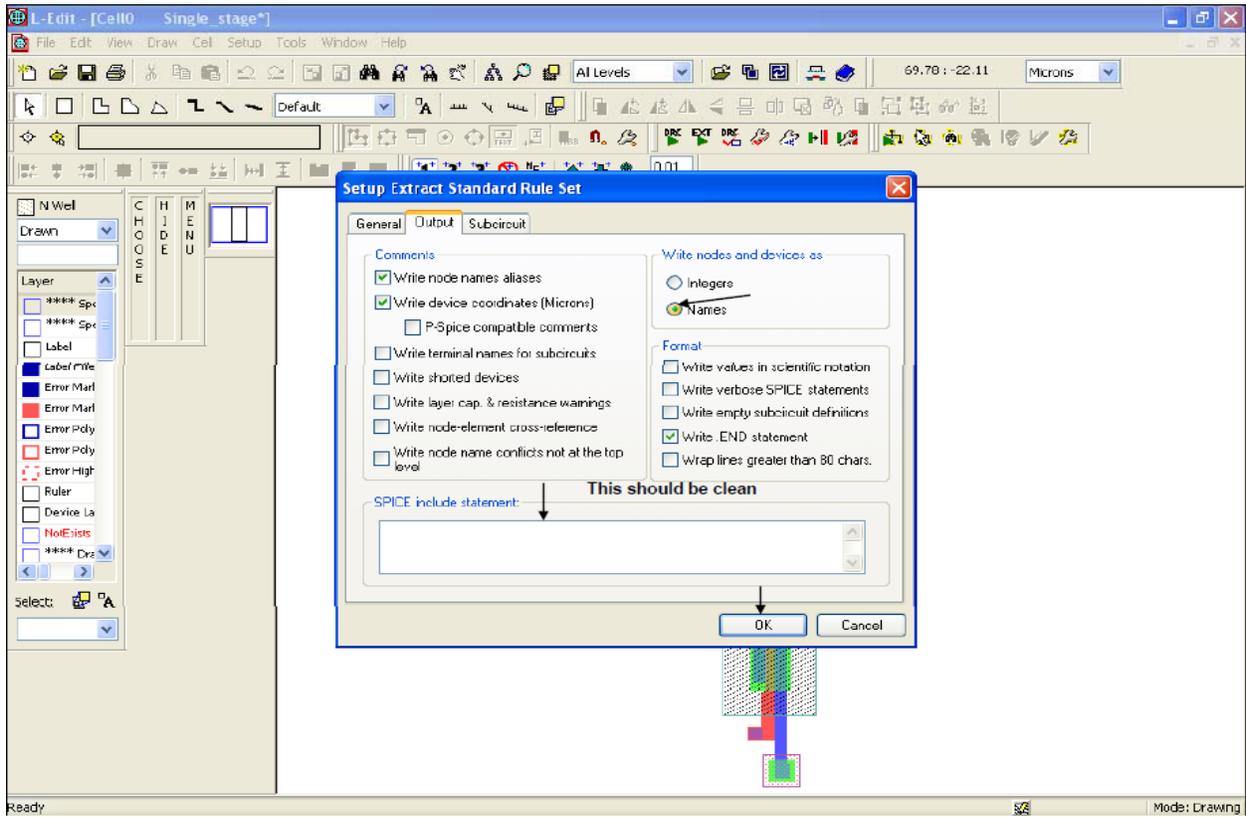


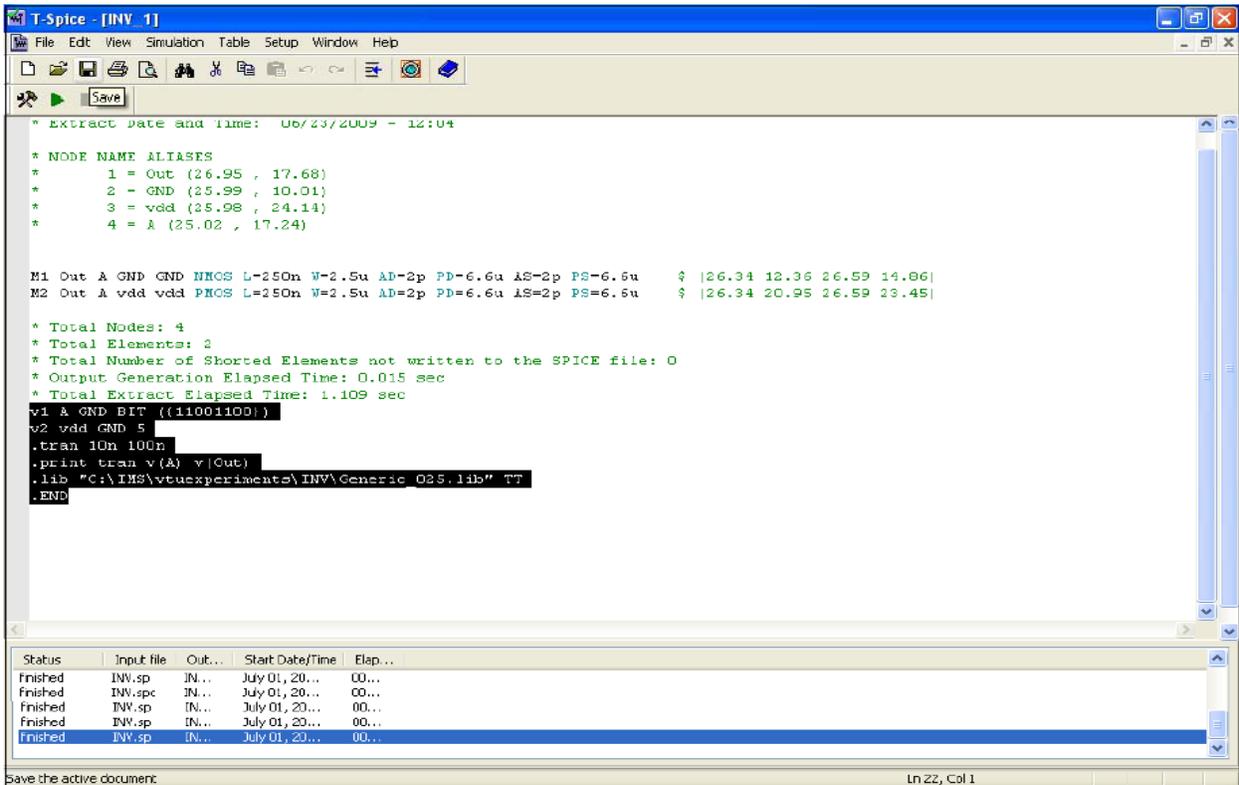
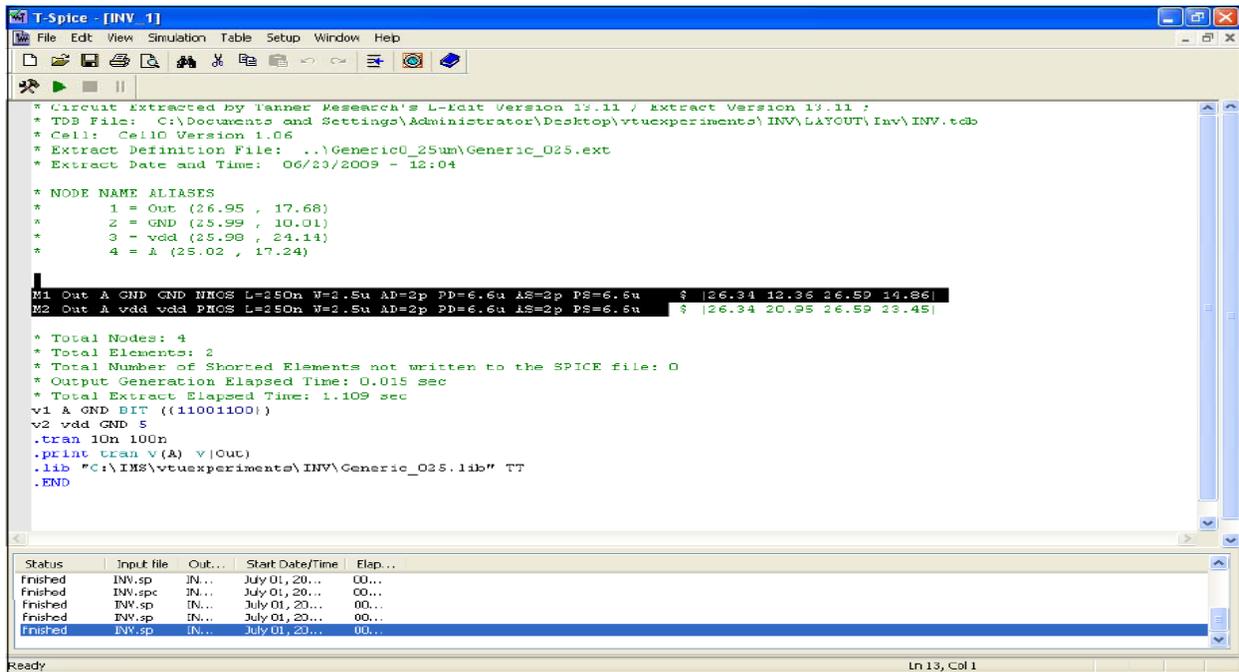












```

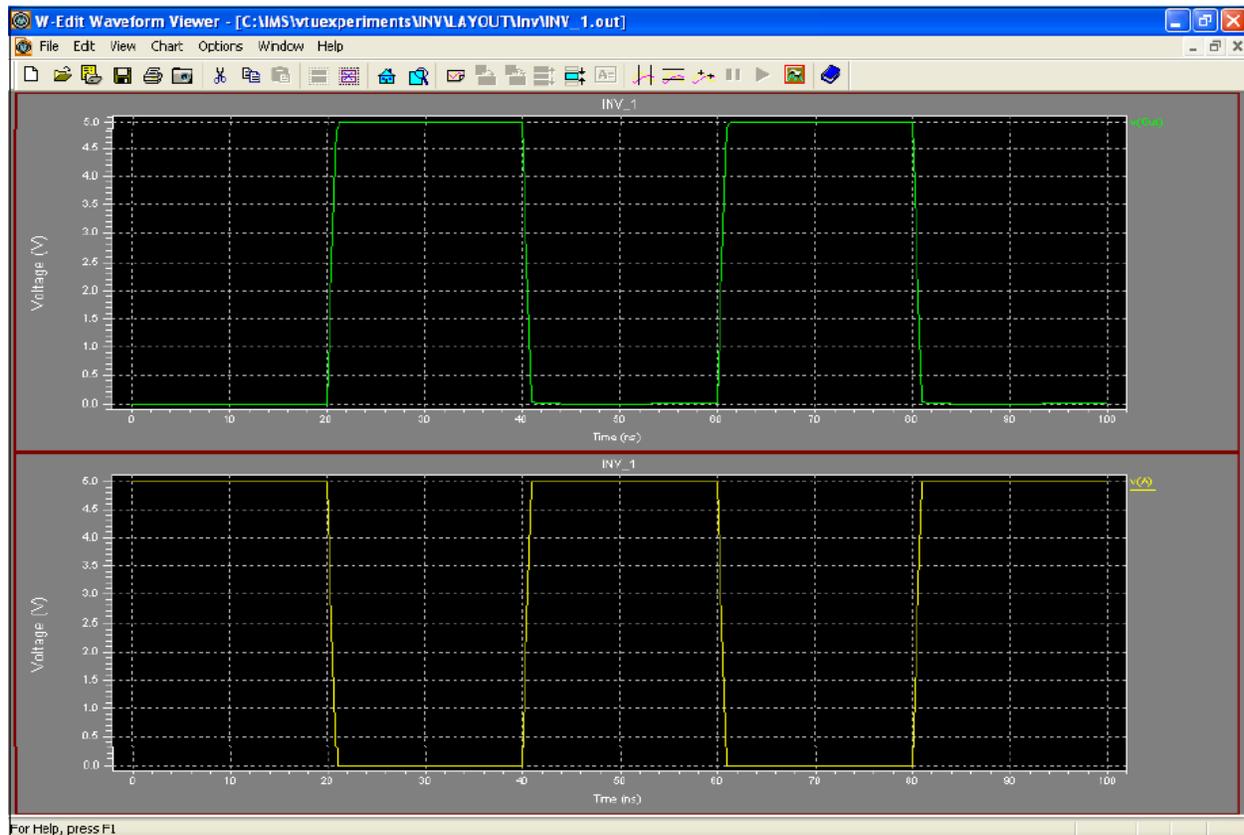
T-Spice - [Single Stage]
File Edit View Simulation Table Setup Window Help
* 4 = Out (70.44 , -37.27)
* Run Simulation VDD (65.74 , -26.79)
* 6 = Vref (65.65 , -52.63)
* 7 = Vinn (61.29 , -44.59)
* 8 = Vinp (70.32 , -44.47)

M1 VDD 3 3 VDD PMOS L=603n W=2.5u AD=1.9975p PD=6.598u AS=2p PS=6.6u $ (61.56 -32.88 62.163 -30.38)
M2 Out 3 VDD PMOS L=603n W=2.5u AD=1.9975p PD=6.598u AS=2p PS=6.6u $ (69.39 -32.87 69.993 -30.37)
M3 2 Vinn 3 GND NMOS L=699n W=2.5u AD=1.9975p PD=6.598u AS=2p PS=6.6u $ (61.56 -41.99 62.259 -39.49)
M4 GND Vref 2 GND NMOS L=400n W=2.88u AD=2.304p PD=7.36u AS=2.304p PS=7.36u $ (66.22 -50.11 66.62 -47.23)
M5 Out Vinp 2 GND NMOS L=699n W=2.5u AD=1.9975p PD=6.598u AS=2p PS=6.6u $ (69.3 -41.87 69.999 -39.37)

* Total Nodes: 8
* Total Elements: 5
* Total Number of Shorted Elements not written to the SPICE file: 0
* Output Generation Elapsed Time: 0.000 sec
* Total Extract Elapsed Time: 1.235 sec
.ac dec 10 1 10g
.tran 10n 100n
.lib "C:\IMS\vtuexperiments\singlestage\Generic_025.lib" TT
v1 Vinp Vinn SIN (0 1m 10) AC 1 180
v2 Vref GND .8
v3 Vdd GND 5
v4 Vinn GND 1.3
.print tran v(Vinn) v(Vinp) v(Vout)
.print ac vdb(Out) vp(Out)
.end

Status Input file Out... Start Date/Time Elap...
Run a simulation Ln 13, Col 5

```



EXP.NO:	DESIGN AND SIMULATION OF CMOS GATES
DATE:	

AIM:

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

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

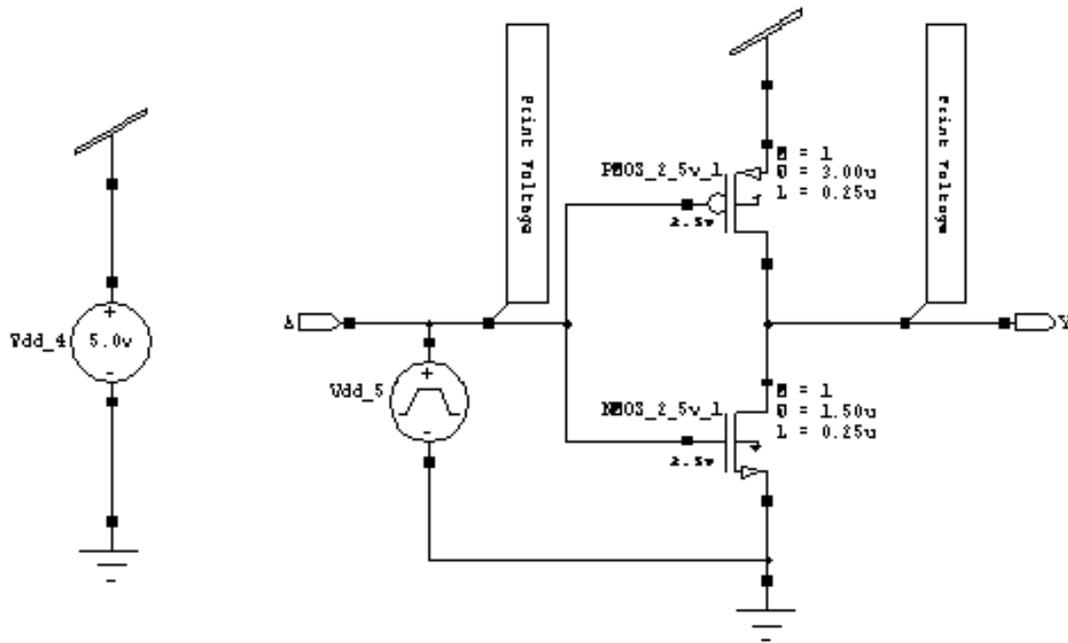
S.NO	DETAILS OF THE STEP
1	Draw the schematic of CMOS Gates using S-edit
2	Perform Transient Analysis of the CMOS Inverter
3	Obtain the output wave form from W-edit
4	Obtain the spice code using T-edit

c) THEORY: (CMOS NOT)

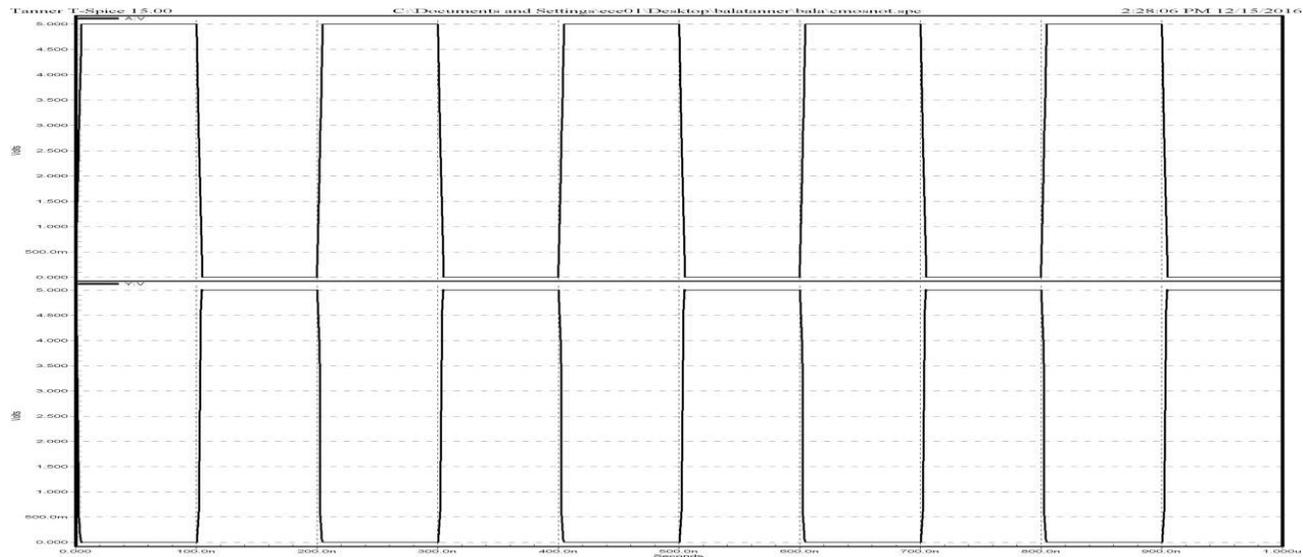
- Inverter consists of nMOS and pMOS transistor in series connected between VDD and GND.
- The gate of the two transistors are shorted and connected to the input. When the input to the inverter $A = 0$, nMOS transistor is OFF and pMOS transistor is ON. The output is pull-up to VDD. When the input $A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

CMOS NOT

SCHEMATIC DIAGRAM:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

*----- Devices With SPICE.ORDER == 0.0 -----

***** Top Level *****

MNMOS_2_5v_1 Y A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$
\$x=4293

+\$y=3300 \$w=414 \$h=600

MPMOS_2_5v_1 Y A Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
+\$x=4293 \$y=4500 \$w=414 \$h=600

*----- Devices With SPICE.ORDER > 0.0 -----

VVdd_4 Vdd Gnd DC 5 \$ \$x=1200 \$y=3800 \$w=400 \$h=600

VVdd_5 A Gnd PULSE(0 5 0 5n 5n 95n 200n) \$ \$x=2800 \$y=3500 \$w=400 \$h=600

.PRINT TRAN V(A) \$ \$x=3250 \$y=4650 \$w=300 \$h=1500 \$r=270

.PRINT TRAN V(Y) \$ \$x=5350 \$y=4650 \$w=300 \$h=1500 \$r=270

***** Simulation Settings - Analysis Section *****

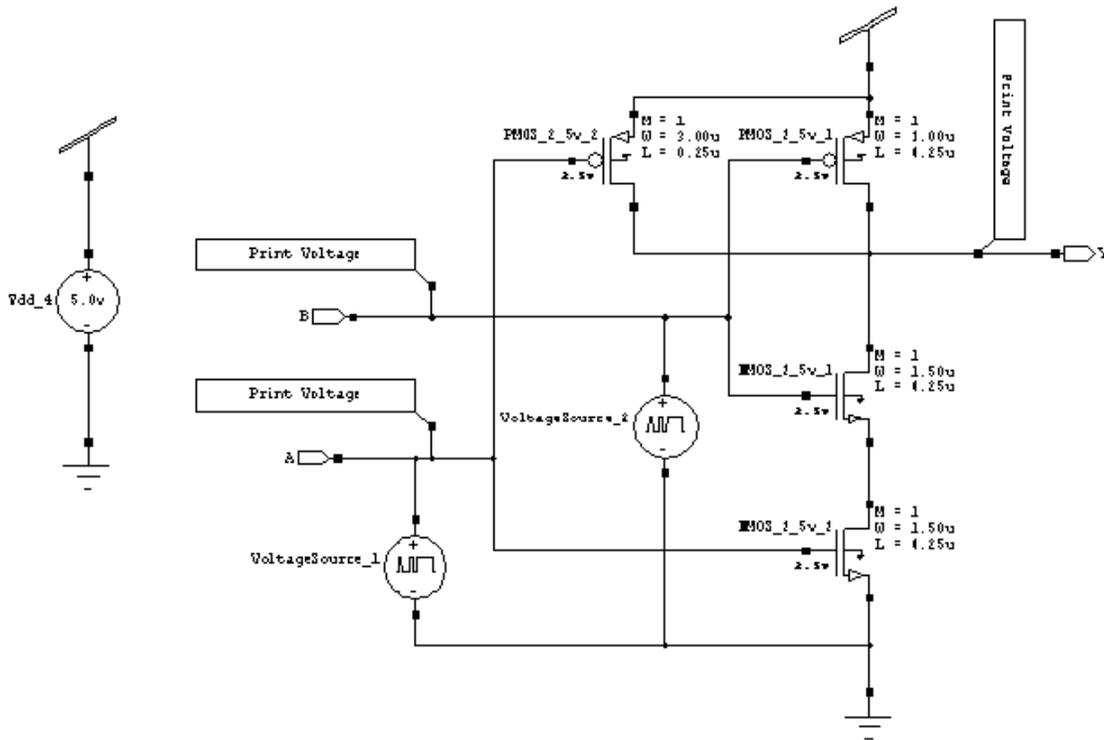
.tran 10ns 1000ns

***** Simulation Settings - Additional SPICE Commands *****

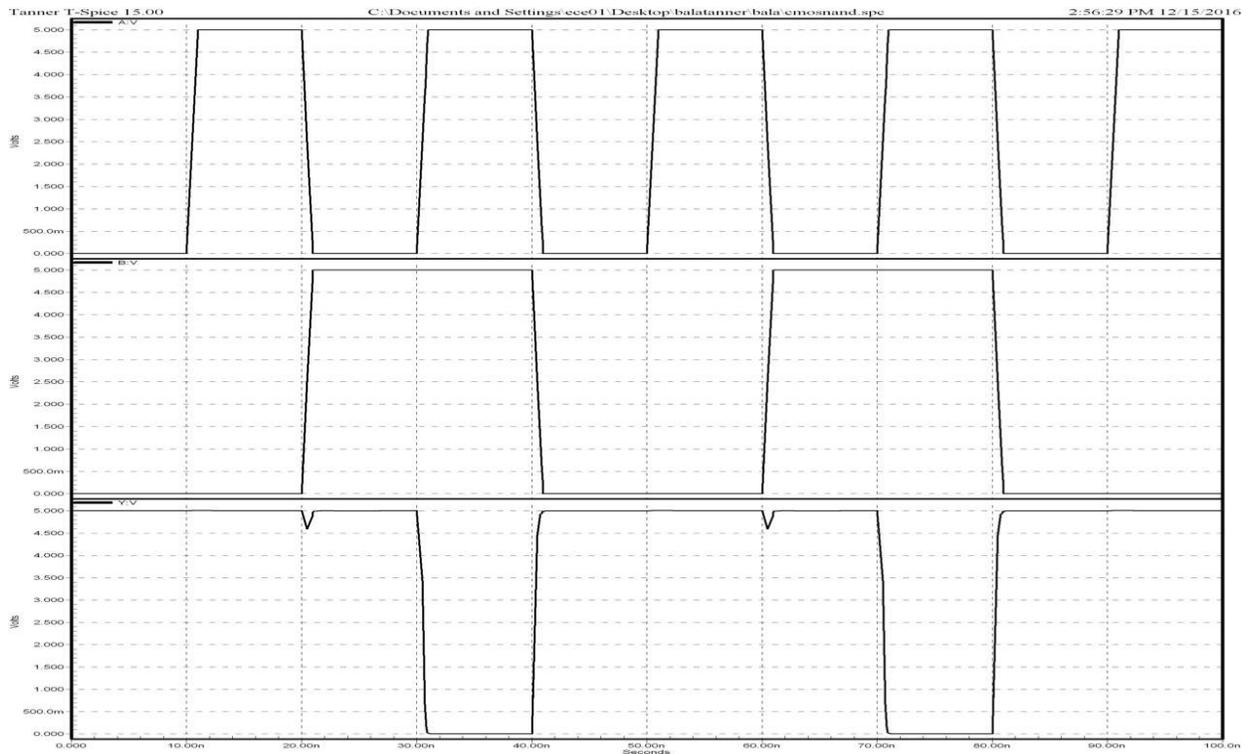
.end

CMOS NAND

SCHEMATIC DIAGRAM:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

*----- Devices With SPICE.ORDER == 0.0 -----

***** Top Level *****

MNMOS_2_5v_1 Y B N_1 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$
\$x=5993

+\$y=3200 \$w=414 \$h=600

MNMOS_2_5v_2 N_1 A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$
+\$x=5993 \$y=2200 \$w=414 \$h=600

MPMOS_2_5v_1 Y B Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
+\$x=5993 \$y=4700 \$w=414 \$h=600

MPMOS_2_5v_2 Y A Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
+\$x=4493 \$y=4700 \$w=414 \$h=600

*----- Devices With SPICE.ORDER > 0.0 -----

VVdd_4 Vdd Gnd DC 5 \$ \$x=1200 \$y=3800 \$w=400 \$h=600

VVoltageSource_1 A Gnd BIT({0101}) \$ \$x=3300 \$y=2100 \$w=400 \$h=600

VVoltageSource_2 B Gnd BIT({0011}) \$ \$x=4900 \$y=3000 \$w=400 \$h=600

.PRINT TRAN V(A) \$ \$x=2650 \$y=3150 \$w=1500 \$h=300 \$r=180

.PRINT TRAN V(B) \$ \$x=2650 \$y=4050 \$w=1500 \$h=300 \$r=180

.PRINT TRAN V(Y) \$ \$x=7050 \$y=4850 \$w=300 \$h=1500 \$r=270

***** Simulation Settings - Analysis Section *****

.tran 10ns 100ns

***** Simulation Settings - Additional SPICE Commands *****

.end

NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

*----- Devices With SPICE.ORDER == 0.0 -----

***** Top Level *****

MNMOS_1 Y A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=4493
+\$y=2300 \$w=414 \$h=600

MNMOS_2 Y B Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5993
+\$y=2200 \$w=414 \$h=600

MPMOS_1 Y B N_3 Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=5993
+\$y=3600 \$w=414 \$h=600

MPMOS_2 N_3 A Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=5993
+\$y=4500 \$w=414 \$h=600

*----- Devices With SPICE.ORDER > 0.0 -----

VVdd_4 Vdd Gnd DC 5 \$ \$x=700 \$y=3800 \$w=400 \$h=600

Vv1 A Gnd BIT({0101}) \$ \$x=2800 \$y=2100 \$w=400 \$h=600

VV2 B Gnd BIT({0011}) \$ \$x=3500 \$y=2100 \$w=400 \$h=600

.PRINT TRAN V(A) \$ \$x=2150 \$y=3150 \$w=1500 \$h=300 \$r=180

.PRINT TRAN V(B) \$ \$x=2750 \$y=3850 \$w=1500 \$h=300 \$r=180

.PRINT TRAN V(Y) \$ \$x=7050 \$y=3650 \$w=300 \$h=1500 \$r=270

***** Simulation Settings - Analysis Section *****

.tran 10ns 100ns

***** Simulation Settings - Additional SPICE Commands *****

.end

RESULT

EXP.NO:	DESIGN AND SIMULATION OF A CMOS INVERTER USING DIGITAL FLOW
DATE:	

AIM:

To perform the functional verification of a CMOS inverter circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

S.NO	DETAILS OF THE STEP
1	Draw the schematic of CMOS inverter using S-edit
2	Perform Transient Analysis of CMOS inverter circuit
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

c) THEORY:

The term CMOS stands for “Complementary Metal Oxide Semiconductor”. CMOS technology is one of the most popular technology in the computer chip design industry and broadly used today to form integrated circuits in numerous and varied applications. Today’s computer memories, CPUs and cell phones make use of this technology due to several key advantages. This technology makes use of both P channel and N channel semiconductor devices.

One of the most popular MOSFET technologies available today is the Complementary MOS or CMOS technology. This is the dominant semiconductor technology for microprocessors, microcontroller chips, memories like RAM, ROM, EEPROM and application specific integrated circuits (ASICs)

SCHEMATIC ENTRY:

NETLIST:

- * SPICE export by: SEDIT 13.12
- * Export time: Wed Dec 11 11:43:07 2019
- * Design: adm705-1
- * Cell: Cell3
- * View: view0
- * Export as: top-level cell
- * Export mode: hierarchical
- * Exclude .model: no
- * Exclude .end: no
- * Expand paths: yes
- * Wrap lines: no
- * Root path: C:\Documents and Settings\Administrator\Desktop\adm705-1
- * Exclude global pins: no
- * Control property name: SPICE

***** Simulation Settings - General section *****

.lib "C:\Documents and Settings\Administrator\My Documents\Tanner EDA\Tanner Tools
v13.1\Libraries\Models\Generic_025.lib" TT

***** Simulation Settings - Parameters and SPICE Options *****

*----- Devices: SPICE.ORDER > 0 -----

MNMOS_1 Out N_2 Gnd N_1 NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u

MPMOS_1 Out N_2 Vdd N_3 PMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u

VVoltageSource_1 Vdd Gnd DC 5

VVoltageSource_2 N_2 Gnd PULSE(0 5 0 5n 5n 95n 200n)

.PRINT TRAN V(Out)

```
.PRINT TRAN V(N_2)
```

```
***** Simulation Settings - Analysis section *****
```

```
.tran 350ns 500ns
```

```
.dc lin source VVoltageSource_1 0 5 0.5
```

```
.print dc v(MNMOS_1,Gnd)
```

```
***** Simulation Settings - Additional SPICE commands *****
```

```
.end
```

EXP.NO:	DESIGN AND SIMULATION OF 4-BIT COUNTER
DATE:	

AIM:

To perform the functional verification of the 4-bit counter circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required for doing the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

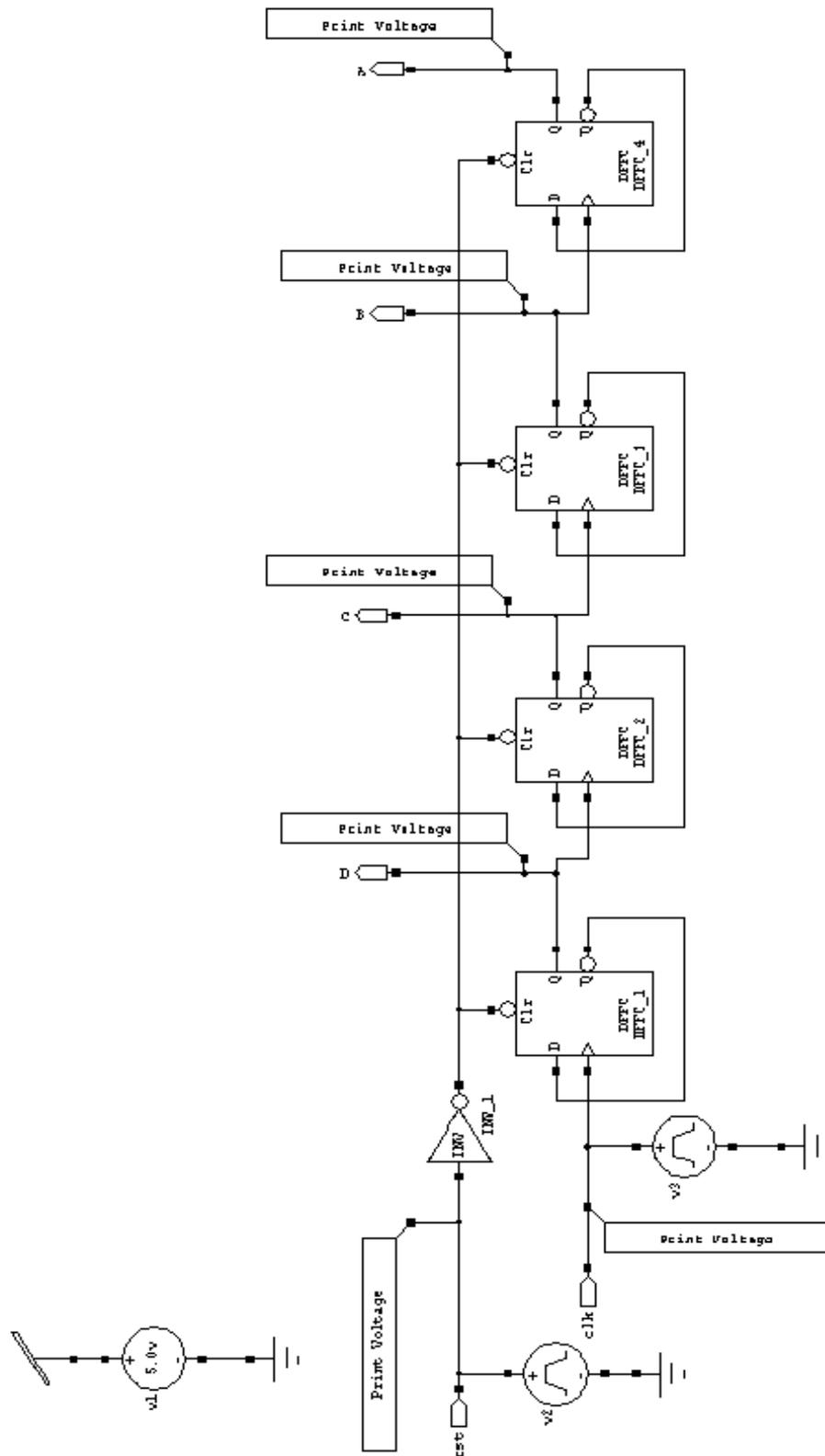
S.NO	DETAILS OF THE STEP
1	Draw the schematic of 4-bit counter using S-edit
2	Perform Transient Analysis of the 4-bit counter
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

c) THEORY: (COUNTER)

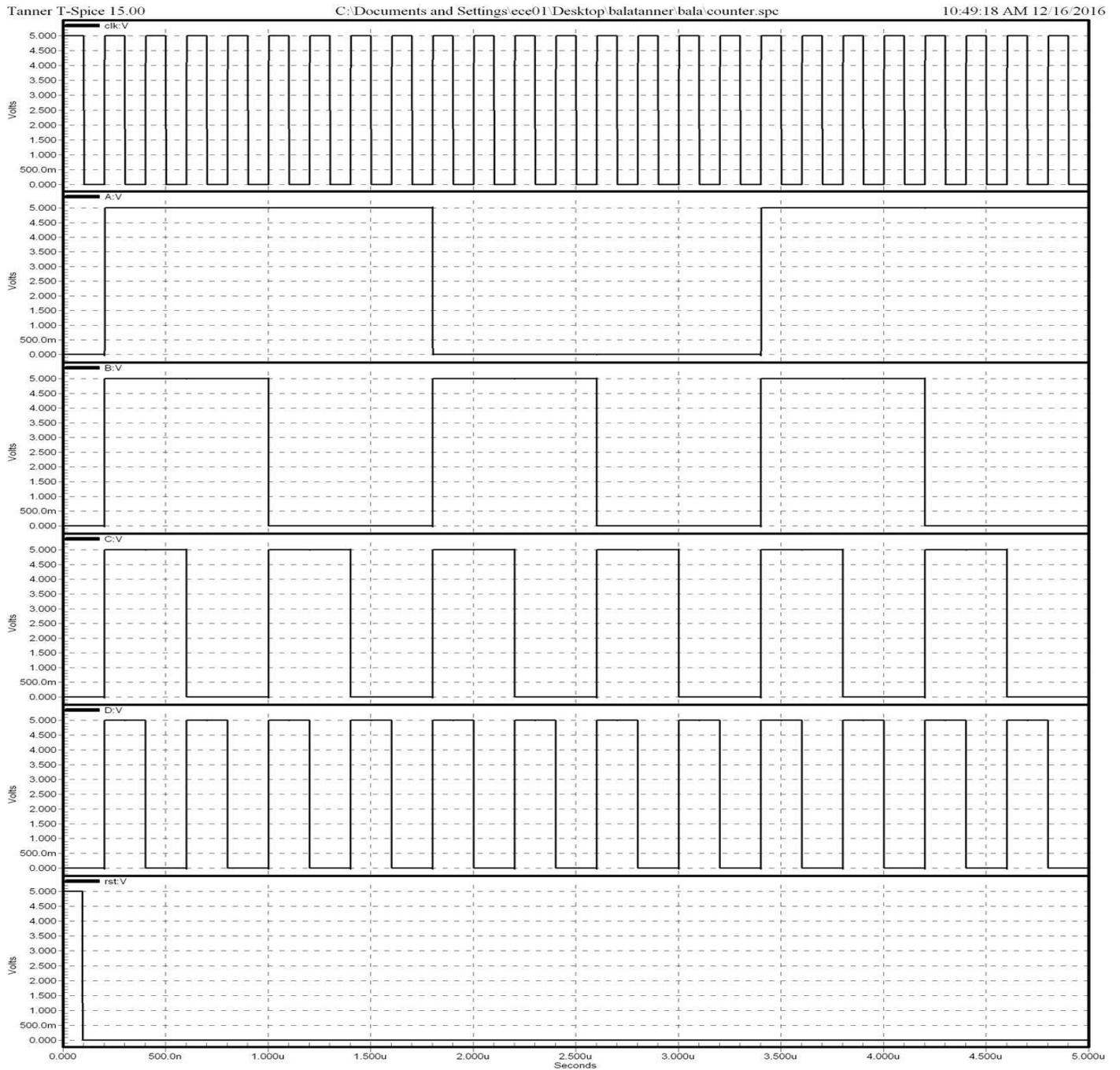
A counter that can change state in either direction, under the control of an up or down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value. When the selector is in the down state, the counter decrements the count. Likewise the counter counts in both the directions continuously until attaining the end of the count. The count is initiated by the positive clock pulse. The counter counts from 0000 to 1111 for up count and 1111 to 0000 for down count.

4-BIT COUNTER

SCHEMATIC DIAGRAM:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

***** Subcircuits *****

.subckt DFFC Clk Clr Data Q QB Gnd Vdd

*----- Devices With SPICE.ORDER < 0.0 -----

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

*----- Devices With SPICE.ORDER == 0.0 -----

MM1p CB Clk Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
\$x=1293

+\$y=1800 \$w=414 \$h=600

MM2n CB Clk Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1293

+\$y=1000 \$w=414 \$h=600

MM3p C CB Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=2693

+\$y=1800 \$w=414 \$h=600

MM4n C CB Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=2693

\$y=1000

+\$w=414 \$h=600

MM5p 3 Data Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=3993

+\$y=5600 \$w=414 \$h=600

MM6p 4 C 3 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=3993

+\$y=4700 \$w=414 \$h=600

MM7n 4 CB 5 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

\$y=3900

+\$w=414 \$h=600

MM8n 5 Data Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

+\$y=3100 \$w=414 \$h=600

MM9p 6 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=5693

+\$y=5600 \$w=414 \$h=600

MM10p 4 CB 6 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=6107

+\$y=4700 \$w=414 \$h=600 \$m

MM11n 4 C 7 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3900

+\$w=414 \$h=600

MM12n 7 10 8 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3100

+\$w=414 \$h=600
 MM13n 8 Clr Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693
 +\$y=2300 \$w=414 \$h=600
 MM14p 9 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=7293
 +\$y=5600 \$w=414 \$h=600
 MM15p 4 CB 9 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=7293
 +\$y=4700 \$w=414 \$h=600
 MM16p 10 4 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=7193
 +\$y=3400 \$w=414 \$h=600
 MM17n 10 4 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=7193
 +\$y=2600 \$w=414 \$h=600
 MM18p 11 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=1393
 +\$y=5200 \$w=414 \$h=600
 MM19p 12 CB 11 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=1393
 +\$y=4400 \$w=414 \$h=600
 MM20An 12 Clr 15 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1393
 +\$y=3600 \$w=414 \$h=600
 MM20n 15 C 13 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1393
 +\$y=2800 \$w=414 \$h=600
 MM21n 13 10 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1393
 +\$y=2000 \$w=414 \$h=600
 MM22p 17 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=3093
 +\$y=5200 \$w=414 \$h=600
 MM23p 12 C 17 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=3093
 +\$y=4400 \$w=414 \$h=600
 MM24n 12 CB 17 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3093
 +\$y=3600 \$w=414 \$h=600
 MM25Ap Q 17 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=5293
 +\$y=4400 \$w=414 \$h=600
 MM25p 17 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=4493
 +\$y=5200 \$w=414 \$h=600
 MM26An Q 17 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5293
 +\$y=3600 \$w=414 \$h=600
 MM26n 17 Clr 14 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=4493
 +\$y=2800 \$w=414 \$h=600
 MM27n 14 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=4493
 +\$y=2000 \$w=414 \$h=600
 MM28Ap QB 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=8193
 +\$y=4300 \$w=414 \$h=600
 MM28p 16 12 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
 \$x=6693
 +\$y=4400 \$w=414 \$h=600
 MM29An QB 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$
 \$x=8193
 +\$y=3500 \$w=414 \$h=600
 MM29n 16 12 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=6693
 +\$y=3600 \$w=414 \$h=600

```

.ends
.subckt INV A Out Gnd Vdd
*----- Devices With SPICE.ORDER < 0.0 -----
* Design: Generic_250nm_LogicGates / Cell: INV / View: Main / Page:
* Designed by: Tanner EDA Library Development Team
* Organization: Tanner EDA - Tanner Research, Inc.
* Info: Inverter
* Date: 5/30/2008 4:06:39 PM
* Revision: 13 $ $x=7600 $y=600 $w=3600 $h=1200
*----- Devices With SPICE.ORDER == 0.0 -----
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $x=4593
+$y=2600 $w=414 $h=600
MM2p Out A Vdd Vdd PMOS25 W=3u L=250n M=2 AS=1.125p PS=3.75u AD=1.95p PD=7.3u $
+$x=4593 $y=3600 $w=414 $h=600
.ends

*----- Devices With SPICE.ORDER == 0.0 -----
***** Top Level *****
XDFFC_1 clk N_1 N_2 D N_2 Gnd Vdd DFFC $ $x=3100 $y=3400 $w=800 $h=1000
XDFFC_2 D N_1 N_4 C N_4 Gnd Vdd DFFC $ $x=4900 $y=3400 $w=800 $h=1000
XDFFC_3 C N_1 N_5 B N_5 Gnd Vdd DFFC $ $x=6700 $y=3400 $w=800 $h=1000
XDFFC_4 B N_1 N_6 A N_6 Gnd Vdd DFFC $ $x=8700 $y=3400 $w=800 $h=1000
XINV_1 rst N_1 Gnd Vdd INV $ $x=2300 $y=4100 $w=600 $h=400

*----- Devices With SPICE.ORDER > 0.0 -----
Vv1 Vdd Gnd DC 5 $ $x=800 $y=6000 $w=400 $h=600
Vv2 rst Gnd PULSE(0 5 0 1n 1n 95n 10u) $ $x=700 $y=3500 $w=400 $h=600
Vv3 clk Gnd PULSE(0 5 0 5n 5n 95n 200n) $ $x=2200 $y=2700 $w=400 $h=600
.PRINT TRAN V(A) $ $x=9550 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(B) $ $x=7950 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(C) $ $x=5950 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(clk) $ $x=1650 $y=2550 $w=300 $h=1500 $r=90
.PRINT TRAN V(D) $ $x=4250 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(rst) $ $x=950 $y=4550 $w=1500 $h=300 $r=180
***** Simulation Settings - Analysis Section *****
.tran 1u 5u
***** Simulation Settings - Additional SPICE Commands *****
.end

```

RESULT:

EXP.NO:	LAYOUT CMOS INVERTOR
DATE:	

AIM:

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

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

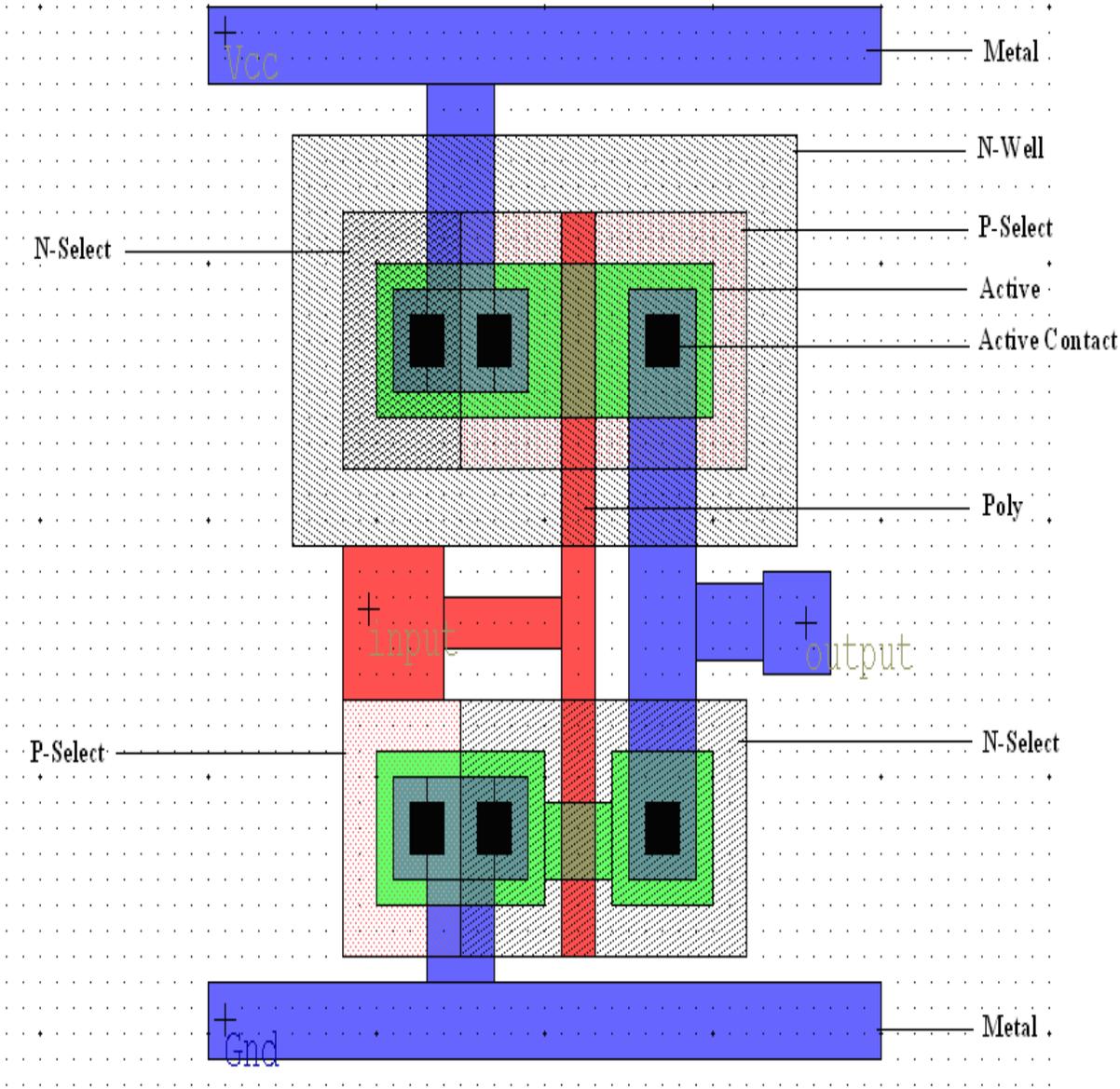
S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	L-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

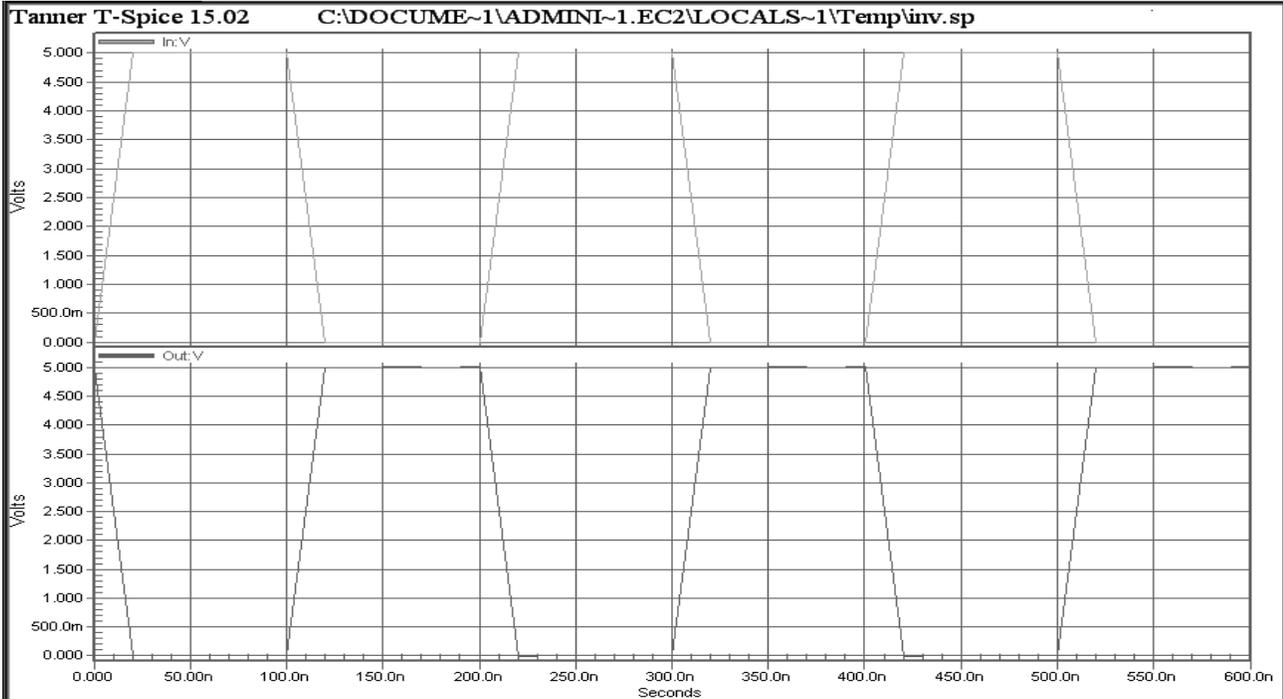
S.NO	DETAILS OF THE STEP
1	Draw the CMOS Inverter layout by obeying the Lamda Rules using L-edit.
2	i.Poly- 2λ ii.Activecontact- 2λ iii.ActiveContact–Metal- 1λ iv.ActiveContact–Activeregion- 2λ v.ActiveRegion–Pselect- 3λ vi.Pselect–nWell- 3λ
3	Check DRC to verify whether any region violate the Lamda rule
4	Setup the extraction and extract the spice code using T-spice.

CMOS INVERTER:

LAYOUT DAIGRAM:



OUTPUT WAVEFORM:



RESULT:

EXP.NO:	AUTOMATIC LAYOUT GENERATION
DATE:	

AIM:

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

FACILITIES REQUIRED AND PROCEDURE

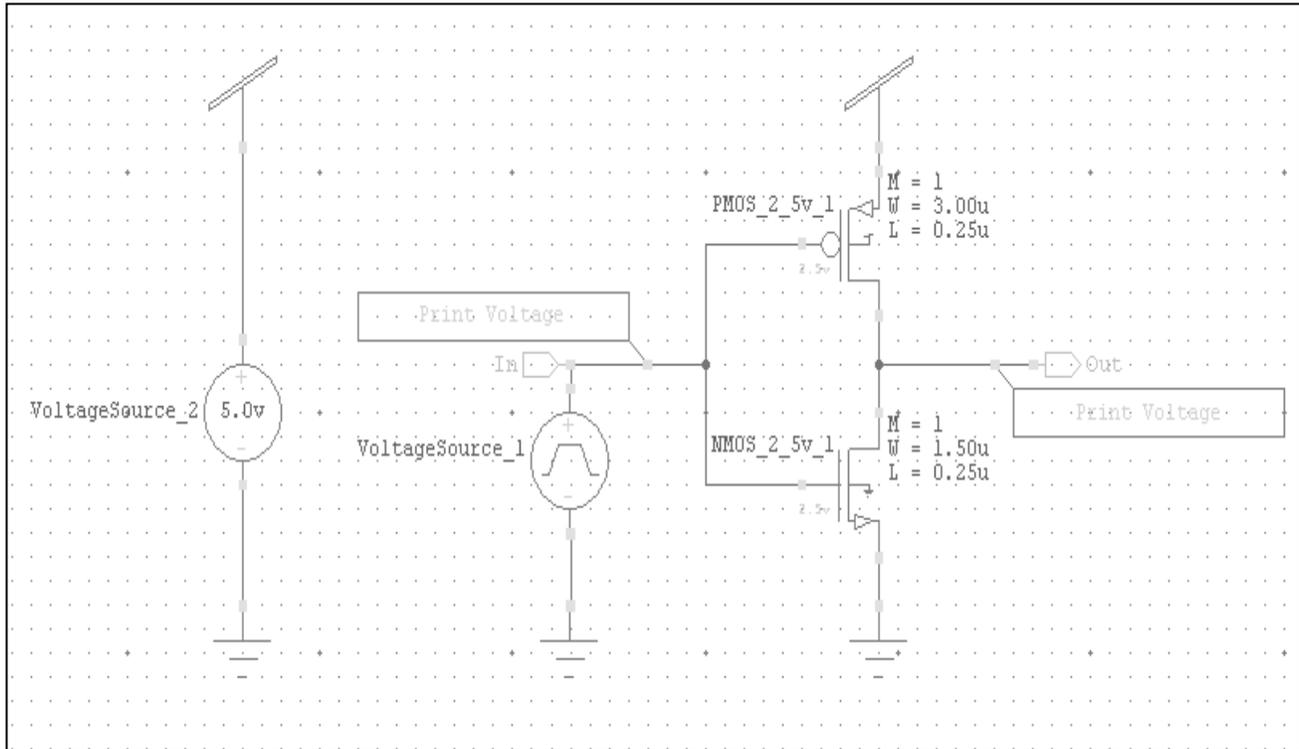
a) Facilities required to do the experiment

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

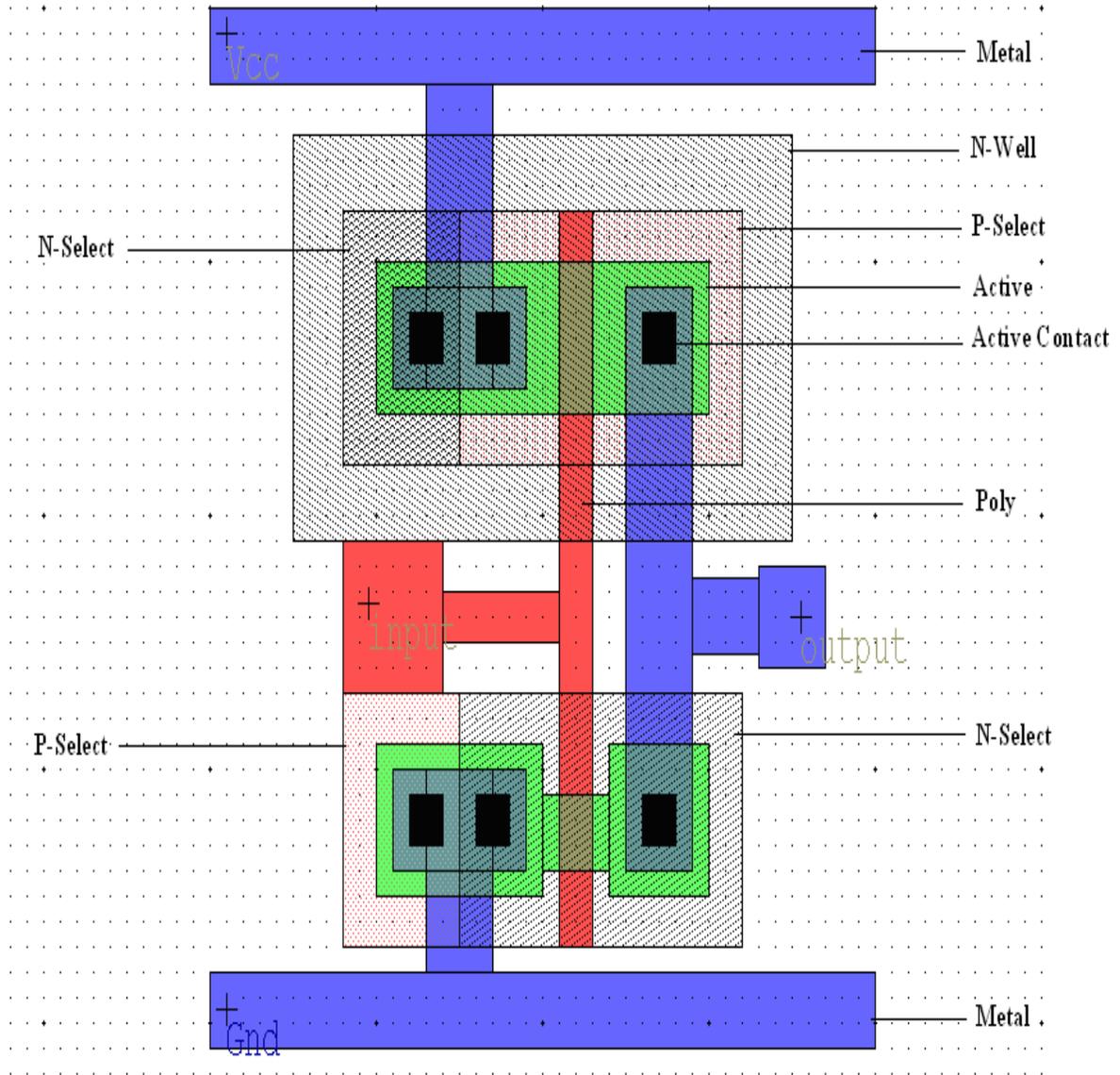
b) Procedure for doing the experiment

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

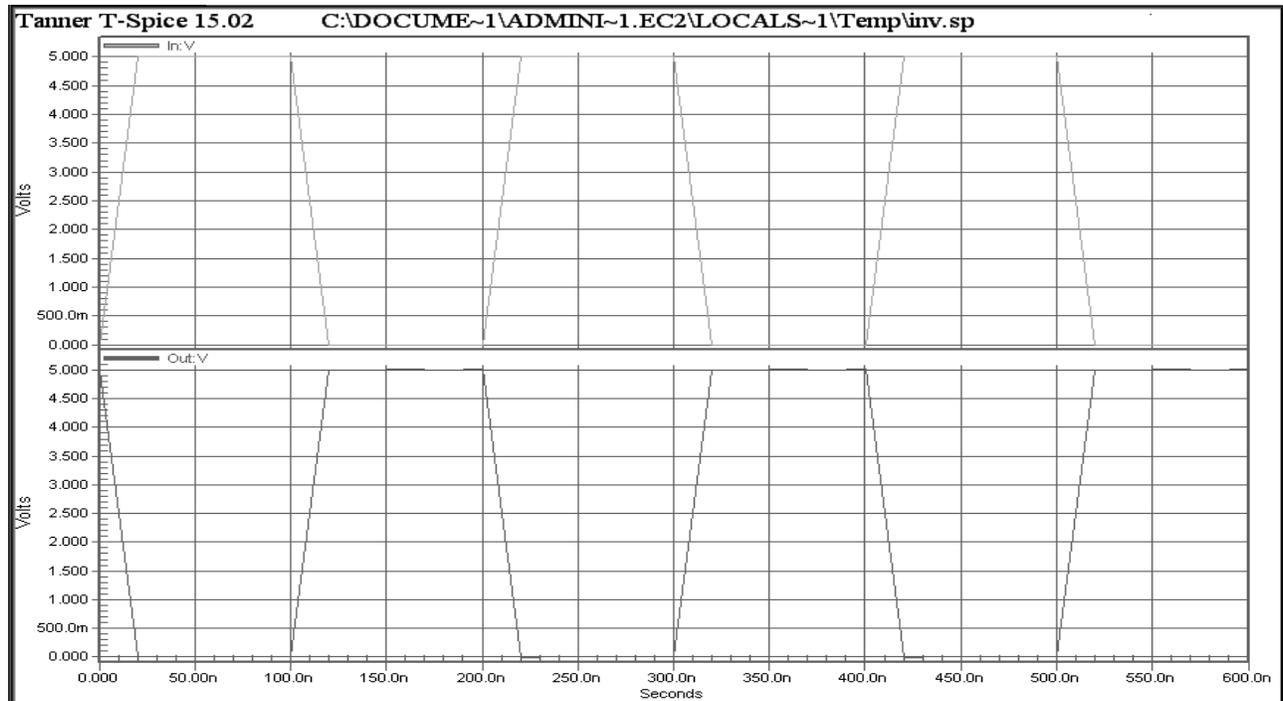
SCHEMATIC DIAGRAM:



LAYOUT GENERATION:



SIMULATED WAVEFORM:



RESULT:

EXP.NO:	DESIGN AND SIMULATE A FLIP-FLOP	
DATE:		

AIM:

To perform the functional verification of a D-flip flop through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

S.NO	DETAILS OF THE STEP
1	Draw the schematic of D-flipflop using S-edit
2	Perform Transient Analysis of D-flipflop
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

c) THEORY: (D-FLIP FLOP)

The D flip-flop tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter. The result may be clocked.

NETLIST:

Circuit Extracted by Tanner Research's L-Edit Version 13.00 / Extract Version 13.00 ;

* TDB File: Layout1

* Cell: Core Version 1.01

* Extract Definition File: lights.ext

* Extract Date and Time: 12/12/2019 - 09:47

.include lights.md

* NODE NAME ALIASES

* 1 = U1/NAND2C_5/Out2 (78 , 54)

* 2 = U1/NAND2C_4/Out2 (44 , 54)

* 3 = U1/NAND2C_3/Out2 (10 , 54)

* 6 = q (100 , 12.5)

* 6 = U1/NAND2C_3/Out1 (2 , 52)

* 6 = U1/NAND2C_4/A (20 , 70)

* 7 = Vdd (-101 , 4)

* 7 = U1/NAND2C_1/Vdd (-51 , 86)

* 7 = U1/NAND2C_2/Vdd (-17 , 86)

* 7 = U1/NAND2C_3/Vdd (-17 , 86)

* 7 = U1/NAND2C_4/Vdd (51 , 86)

* 7 = U1/NAND2C_5/Vdd (85 , 86)

* 8 = d (-101 , 106.5)

* 8 = U1/NAND2C_1/A (-82 , 70)

* 8 = U1/NAND2C_5/A (54 , 70)

* 8 = U1/NAND2C_5/B (62 , 63)

* 9 = U1/NAND2C_1/Out1 (-66 , 52)

* 9 = U1/NAND2C_3/A (-14 , 70)

* 10 = U1/NAND2C_2/Out1 (-32 , 52)

* 10 = U1/NAND2C_4/B (28 , 63)

- * 11 = U1/NAND2C_2/B (-40 , 63)
- * 11 = U1/NAND2C_5/Out1 (70 , 52)
- * 12 = clk (-101 , 4.5)
- * 12 = U1/NAND2C_1/B (-74 , 63)
- * 12 = U1/NAND2C_2/A (-48 , 70)
- * 13 = U1/NAND2C_2/Out2 (-24 , 54)
- * 14 = U1/NAND2C_1/Out2 (-58 , 54)
- * 17 = Gnd (92 , 4)
- * 17 = U1/NAND2C_1/Gnd (-51 , 28)
- * 17 = U1/NAND2C_2/Gnd (-17 , 28)
- * 17 = U1/NAND2C_3/Gnd (-17 , 28)
- * 17 = U1/NAND2C_4/Gnd (51 , 28)
- * 17 = U1/NAND2C_5/Gnd (85 , 28)
- * 18 = qbar (100 , 98.5)
- * 18 = U1/NAND2C_3/B (-6 , 63)
- * 18 = U1/NAND2C_4/Out1 (36 , 52)

M1 Vdd d U1/NAND2C_2/B Vdd PMOS L=2u W=28u AD=84p PD=34u AS=84p PS=34u

M2 U1/NAND2C_2/B d Vdd Vdd PMOS L=2u W=28u AD=84p PD=34u AS=144p PS=68u

M3 U1/NAND2C_5/Out2 U1/NAND2C_2/B Vdd Vdd PMOS L=2u W=28u AD=148p PD=68u AS=84p PS=34u

M4 Vdd U1/NAND2C_2/Out1 qbar Vdd PMOS L=2u W=28u AD=84p PD=34u AS=84p PS=34u

M5 qbar q Vdd Vdd PMOS L=2u W=28u AD=84p PD=34u AS=144p PS=68u

M6 U1/NAND2C_4/Out2 qbar Vdd Vdd PMOS L=2u W=28u AD=148p PD=68u AS=84p PS=34u

M7 Vdd qbar q Vdd PMOS L=2u W=28u AD=84p PD=34u AS=84p PS=34u

M8 U1/NAND2C_3/Out2 q Vdd Vdd PMOS L=2u W=28u AD=148p PD=68u AS=84p PS=34u

M9 Gnd d 5 Gnd NMOS L=2u W=28u AD=122p PD=47u AS=28p PS=30u

M10 5 d U1/NAND2C_2/B Gnd NMOS L=2u W=28u AD=28p PD=30u AS=148p PS=68u

M11 U1/NAND2C_5/Out2 U1/NAND2C_2/B Gnd Gnd NMOS L=2u W=28u AD=148p PD=68u AS=122p PS=47u

M12 Gnd U1/NAND2C_2/Out1 4 Gnd NMOS L=2u W=28u AD=122p PD=47u AS=28p PS=30u

M13 4 q qbar Gnd NMOS L=2u W=28u AD=28p PD=30u AS=148p PS=68u

M14 U1/NAND2C_4/Out2 qbar Gnd Gnd NMOS L=2u W=28u AD=148p PD=68u AS=122p PS=47u

M15 U1/NAND2C_3/Out2 q Gnd Gnd NMOS L=2u W=28u AD=148p PD=68u AS=122p PS=47u

M16 q U1/NAND2C_1/Out1 Vdd Vdd PMOS L=2u W=28u AD=84p PD=34u AS=144p PS=68u

M17 Vdd U1/NAND2C_2/B U1/NAND2C_2/Out1 Vdd PMOS L=2u W=28u AD=84p PD=34u AS=84p PS=34u

M18 U1/NAND2C_2/Out1 clk Vdd Vdd PMOS L=2u W=28u AD=84p PD=34u AS=144p PS=68u

M19 U1/NAND2C_2/Out2 U1/NAND2C_2/Out1 Vdd Vdd PMOS L=2u W=28u AD=148p PD=68u AS=84p PS=34u

M20 Vdd clk U1/NAND2C_1/Out1 Vdd PMOS L=2u W=28u AD=84p PD=34u AS=84p PS=34u

M21 U1/NAND2C_1/Out1 d Vdd Vdd PMOS L=2u W=28u AD=84p PD=34u AS=144p PS=68u

M22 U1/NAND2C_1/Out2 U1/NAND2C_1/Out1 Vdd Vdd PMOS L=2u W=28u AD=148p PD=68u AS=84p PS=34u

M23 Gnd qbar 19 Gnd NMOS L=2u W=28u AD=122p PD=47u AS=28p PS=30u

M24 19 U1/NAND2C_1/Out1 q Gnd NMOS L=2u W=28u AD=28p PD=30u AS=148p PS=68u

M25 Gnd U1/NAND2C_2/B 16 Gnd NMOS L=2u W=28u AD=122p PD=47u AS=28p PS=30u

M26 16 clk U1/NAND2C_2/Out1 Gnd NMOS L=2u W=28u AD=28p PD=30u AS=148p PS=68u

M27 U1/NAND2C_2/Out2 U1/NAND2C_2/Out1 Gnd Gnd NMOS L=2u W=28u AD=148p PD=68u AS=122p PS=47u

M28 Gnd clk 15 Gnd NMOS L=2u W=28u AD=122p PD=47u AS=28p PS=30u

M29 15 d U1/NAND2C_1/Out1 Gnd NMOS L=2u W=28u AD=28p PD=30u AS=148p PS=68u

M30 U1/NAND2C_1/Out2 U1/NAND2C_1/Out1 Gnd Gnd NMOS L=2u W=28u AD=148p PD=68u AS=122p PS=47u

* Total Nodes: 19

* Total Elements: 30

* Total Number of Shorted Elements not written to the SPICE file: 10

* Output Generation Elapsed Time: 0.000 sec

* Total Extract Elapsed Time: 1.875 sec

.END

RESULT:

EXP.NO:	DESIGN AND SIMULATE A CMOS INVERTING AMPLIFIER	
DATE:		

AIM:

To perform the functional verification of a CMOS inverting amplifier circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

S.NO	DETAILS OF THE STEP
1	Draw the schematic of CMOS inverting amplifier using S-edit
2	Perform Transient Analysis of CMOS inverting amplifier circuit
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

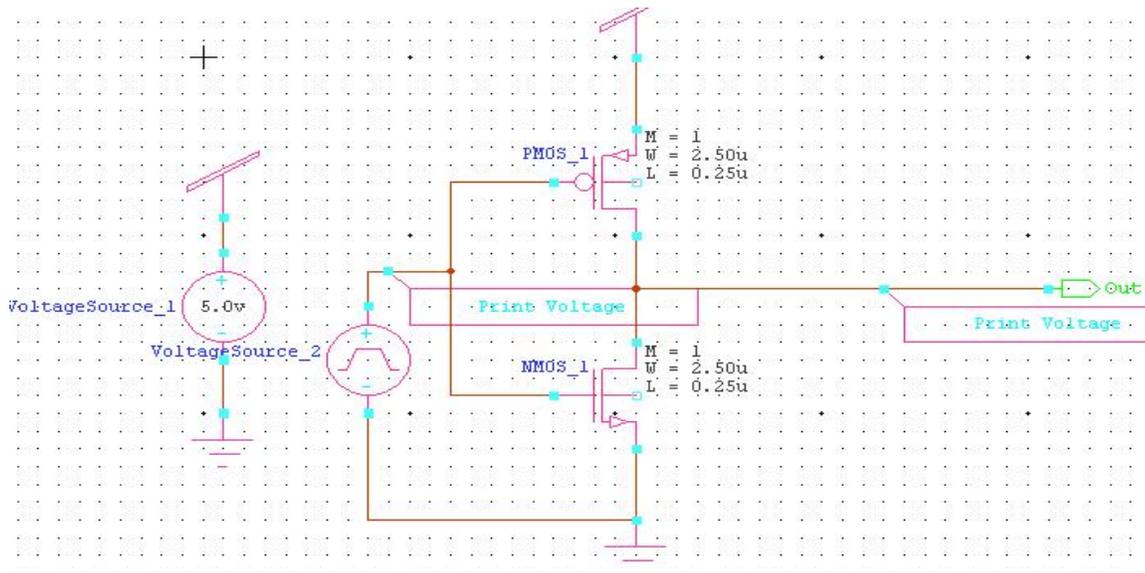
c) THEORY:

CMOS Inverter consists of nMOS and pMOS transistor in series connected between VDD and GND. The gate of the two transistors are shorted and connected to the input.

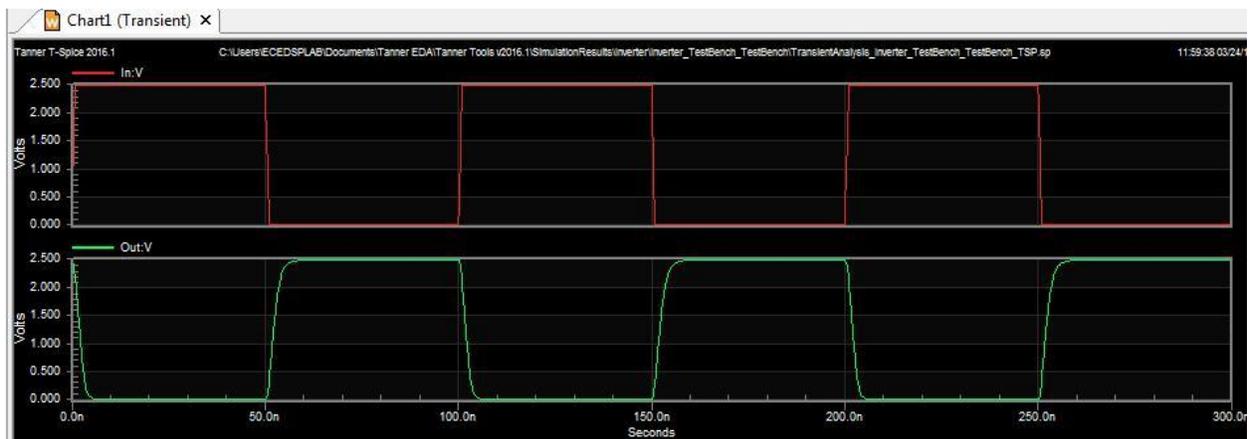
When the input to the inverter $A = 0$, nMOS transistor is OFF and pMOS transistor is ON. The output is pull-up to VDD.

When the input $A = 1$, nMOS transistor is ON and pMOS transistor is OFF. The Output is Pull-down to GND.

SCHEMATIC ENTRY:



OUTPUT WAVEFORM:



NETLIST:

SPICE export by: SEDIT 13.12

* Export time: Wed Dec 11 11:43:07 2019

* Design: adm705-1

* Cell: Cell3

* View: view0

* Export as: top-level cell

* Export mode: hierarchical

* Exclude .model: no

* Exclude .end: no

* Expand paths: yes

* Wrap lines: no

* Root path: C:\Documents and Settings\Administrator\Desktop\adm705-1

* Exclude global pins: no

* Control property name: SPICE

***** Simulation Settings - General section *****

.lib "C:\Documents and Settings\Administrator\My Documents\Tanner EDA\Tanner Tools
v13.1\Libraries\Models\Generic_025.lib" TT

***** Simulation Settings - Parameters and SPICE Options *****

*----- Devices: SPICE.ORDER > 0 -----

MNMOS_1 Out N_2 Gnd N_1 NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u

MPMOS_1 Out N_2 Vdd N_3 PMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u

VVoltageSource_1 Vdd Gnd DC 5

VVoltageSource_2 N_2 Gnd PULSE(0 5 0 5n 5n 95n 200n)

.PRINT TRAN V(Out)

.PRINT TRAN V(N_2)

***** Simulation Settings - Analysis section *****

```
.tran 350ns 500ns
```

```
.dc lin source VVoltageSource_1 0 5 0.5
```

```
.print dc v(MNMOS_1,Gnd)
```

```
***** Simulation Settings - Additional SPICE commands *****
```

```
.end
```

RESULT:

EXP.NO:	DESIGN AND SIMULATION OF COMMON SOURCE AMPLIFIER	
DATE:		

AIM:

To perform the Design and Simulation of Common Source Amplifier circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

S.NO	DETAILS OF THE STEP
1	Draw the schematic of Common Source Amplifier using S-edit
2	Perform Transient Analysis of Common Source Amplifier circuit
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

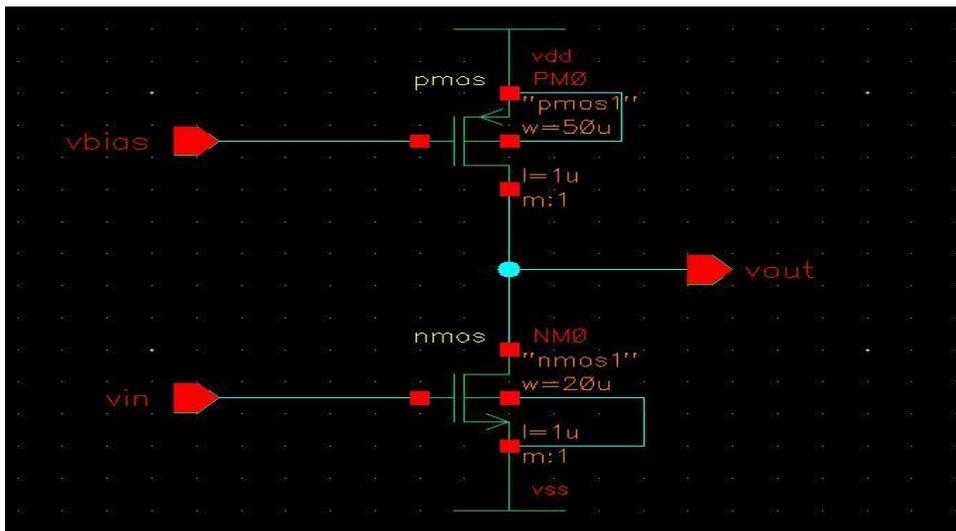
c) THEORY:

In electronics, a **common-source** amplifier is one of three basic single-stage field-effect transistor (FET) amplifier topologies, typically used as a voltage or transconductance amplifier. The easiest way to tell if a FET is common source, common drain, or common gate is to examine where the signal enters and leaves. The remaining terminal is what is known as "common". In this example, the signal enters the gate, and exits the drain. The only terminal remaining is the source. This is a common-source FET circuit. The analogous bipolar junction transistor circuit is the common-emitter amplifier.

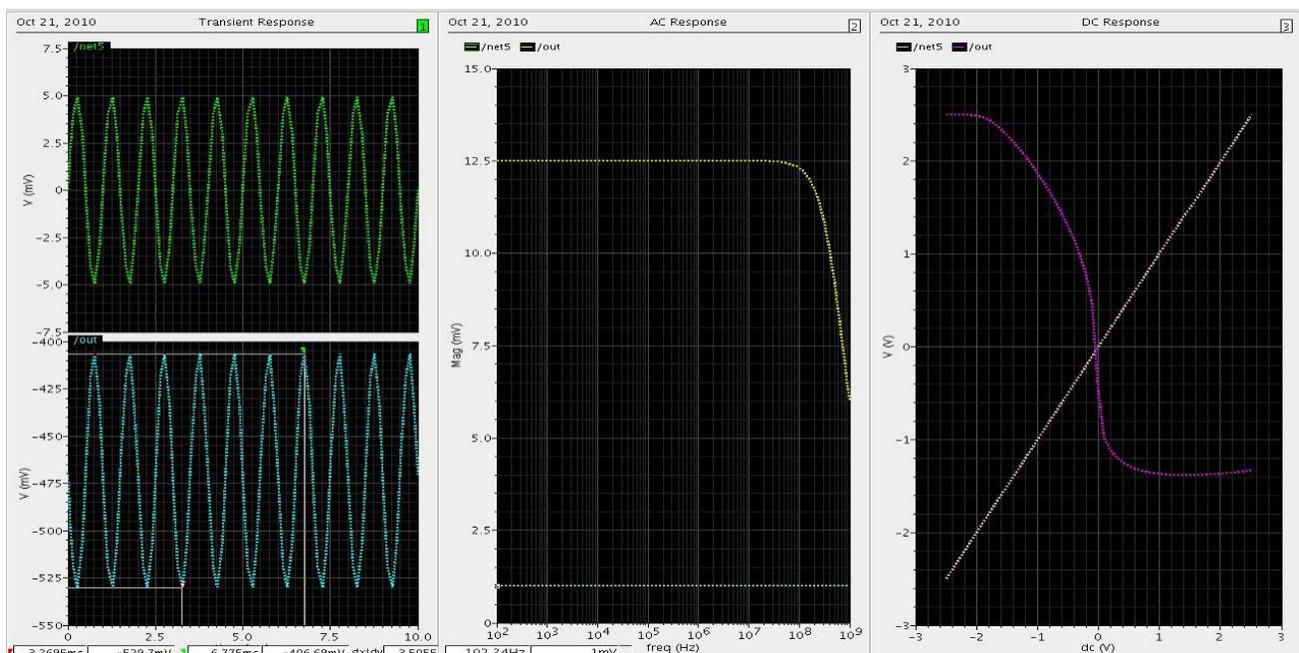
The common-source (CS) amplifier may be viewed as a transconductance amplifier or as a voltage amplifier. (See classification of amplifiers).

As a transconductance amplifier, the input voltage is seen as modulating the current going to the load. As a voltage amplifier, input voltage modulates the amount of current flowing through the FET, changing the voltage across the output resistance according to Ohm's law. However, the FET device's output resistance typically is not high enough for a reasonable transconductance amplifier (ideally infinite), nor low enough for a decent voltage amplifier (ideally zero). Another major drawback is the amplifier's limited high-frequency response.

SCHEMATIC ENTRY:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

***** Subcircuits *****

.subckt DFFC Clk Clr Data Q QB Gnd Vdd

*----- Devices With SPICE.ORDER < 0.0 -----

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

*----- Devices With SPICE.ORDER == 0.0 -----

MM1p CB Clk Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
\$x=1293

+\$y=1800 \$w=414 \$h=600

MM2n CB Clk Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1293

+\$y=1000 \$w=414 \$h=600

MM3p C CB Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=2693

+\$y=1800 \$w=414 \$h=600

MM4n C CB Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=2693

\$y=1000

+\$w=414 \$h=600

MM5p 3 Data Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=3993

+\$y=5600 \$w=414 \$h=600

MM6p 4 C 3 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=3993

+\$y=4700 \$w=414 \$h=600

MM7n 4 CB 5 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

\$y=3900

+\$w=414 \$h=600

MM8n 5 Data Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

+\$y=3100 \$w=414 \$h=600

MM9p 6 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=5693

+\$y=5600 \$w=414 \$h=600

MM10p 4 CB 6 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=6107

+\$y=4700 \$w=414 \$h=600 \$m

MM11n 4 C 7 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3900

+\$w=414 \$h=600

MM12n 7 10 8 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3100

+\$w=414 \$h=600

MM13n 8 Clr Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

+\$y=2300 \$w=414 \$h=600

```

MM14p 9 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=5600 $W=414 $H=600
MM15p 4 CB 9 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=4700 $W=414 $H=600
MM16p 10 4 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7193
+$Y=3400 $W=414 $H=600
MM17n 10 4 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=7193
+$Y=2600 $W=414 $H=600
MM18p 11 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=5200 $W=414 $H=600
MM19p 12 CB 11 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=4400 $W=414 $H=600
MM20An 12 Clr 15 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=1393
+$Y=3600 $W=414 $H=600
MM20n 15 C 13 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2800 $W=414 $H=600
MM21n 13 10 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2000 $W=414 $H=600
MM22p 17 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=5200 $W=414 $H=600
MM23p 12 C 17 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=4400 $W=414 $H=600
MM24n 12 CB 17 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=3093
+$Y=3600 $W=414 $H=600
MM25Ap Q 17 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=5293
+$Y=4400 $W=414 $H=600
MM25p 17 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=4493
+$Y=5200 $W=414 $H=600
MM26An Q 17 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=5293
+$Y=3600 $W=414 $H=600
MM26n 17 Clr 14 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2800 $W=414 $H=600
MM27n 14 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2000 $W=414 $H=600
MM28Ap QB 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=8193
+$Y=4300 $W=414 $H=600
MM28p 16 12 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=6693
+$Y=4400 $W=414 $H=600
MM29An QB 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=8193
+$Y=3500 $W=414 $H=600
MM29n 16 12 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=6693
+$Y=3600 $W=414 $H=600
.ends

```

```

.subckt INV A Out Gnd Vdd
*----- Devices With SPICE.ORDER < 0.0 -----
* Design: Generic_250nm_LogicGates / Cell: INV / View: Main / Page:
* Designed by: Tanner EDA Library Development Team
* Organization: Tanner EDA - Tanner Research, Inc.
* Info: Inverter
* Date: 5/30/2008 4:06:39 PM
* Revision: 13 $ $x=7600 $y=600 $w=3600 $h=1200
*----- Devices With SPICE.ORDER == 0.0 -----
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $x=4593
+$y=2600 $w=414 $h=600
MM2p Out A Vdd Vdd PMOS25 W=3u L=250n M=2 AS=1.125p PS=3.75u AD=1.95p PD=7.3u $
+$x=4593 $y=3600 $w=414 $h=600
.ends

*----- Devices With SPICE.ORDER == 0.0 -----
***** Top Level *****
XDFFC_1 clk N_1 N_2 D N_2 Gnd Vdd DFFC $ $x=3100 $y=3400 $w=800 $h=1000
XDFFC_2 D N_1 N_4 C N_4 Gnd Vdd DFFC $ $x=4900 $y=3400 $w=800 $h=1000
XDFFC_3 C N_1 N_5 B N_5 Gnd Vdd DFFC $ $x=6700 $y=3400 $w=800 $h=1000
XDFFC_4 B N_1 N_6 A N_6 Gnd Vdd DFFC $ $x=8700 $y=3400 $w=800 $h=1000
XINV_1 rst N_1 Gnd Vdd INV $ $x=2300 $y=4100 $w=600 $h=400

*----- Devices With SPICE.ORDER > 0.0 -----
Vv1 Vdd Gnd DC 5 $ $x=800 $y=6000 $w=400 $h=600
Vv2 rst Gnd PULSE(0 5 0 1n 1n 95n 10u) $ $x=700 $y=3500 $w=400 $h=600
Vv3 clk Gnd PULSE(0 5 0 5n 5n 95n 200n) $ $x=2200 $y=2700 $w=400 $h=600
.PRINT TRAN V(A) $ $x=9550 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(B) $ $x=7950 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(C) $ $x=5950 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(clk) $ $x=1650 $y=2550 $w=300 $h=1500 $r=90
.PRINT TRAN V(D) $ $x=4250 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(rst) $ $x=950 $y=4550 $w=1500 $h=300 $r=180
***** Simulation Settings - Analysis Section *****
.tran 1u 5u
***** Simulation Settings - Additional SPICE Commands *****
.end

```

RESULT:

EXP.NO:	DESIGN AND SIMULATION OF COMMON DRAIN AMPLIFIER	
DATE:		

AIM:

To perform the Design and Simulation of Common drain amplifier circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

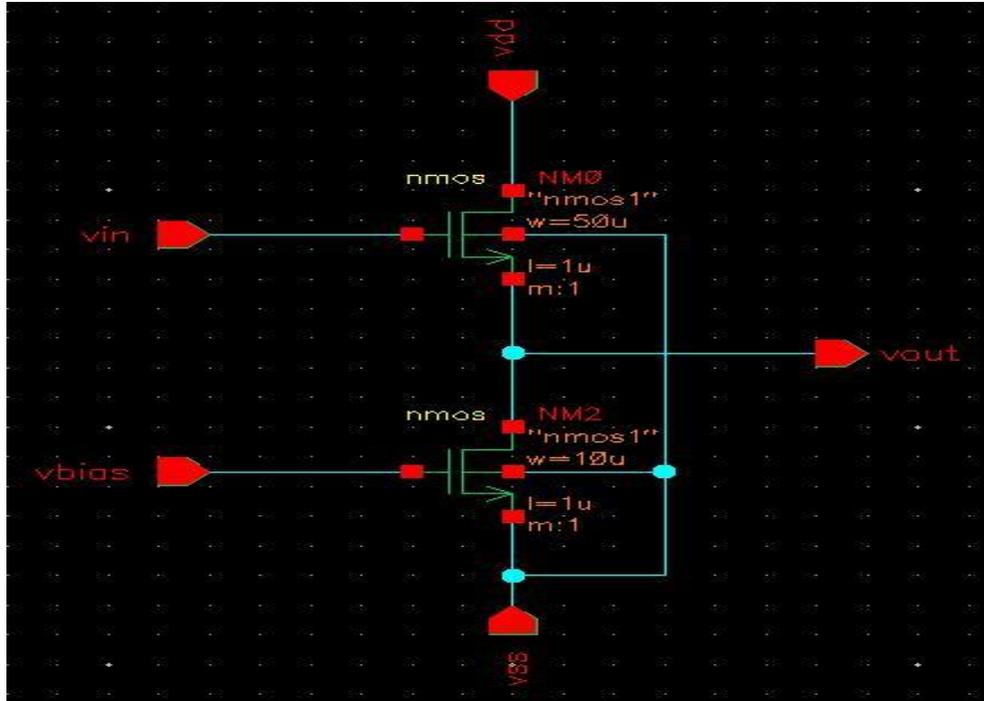
b) Procedure for doing the experiment

S.NO	DETAILS OF THE STEP
1	Draw the schematic of Common drain amplifier using S-edit
2	Perform Transient Analysis of Common drain amplifier circuit
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

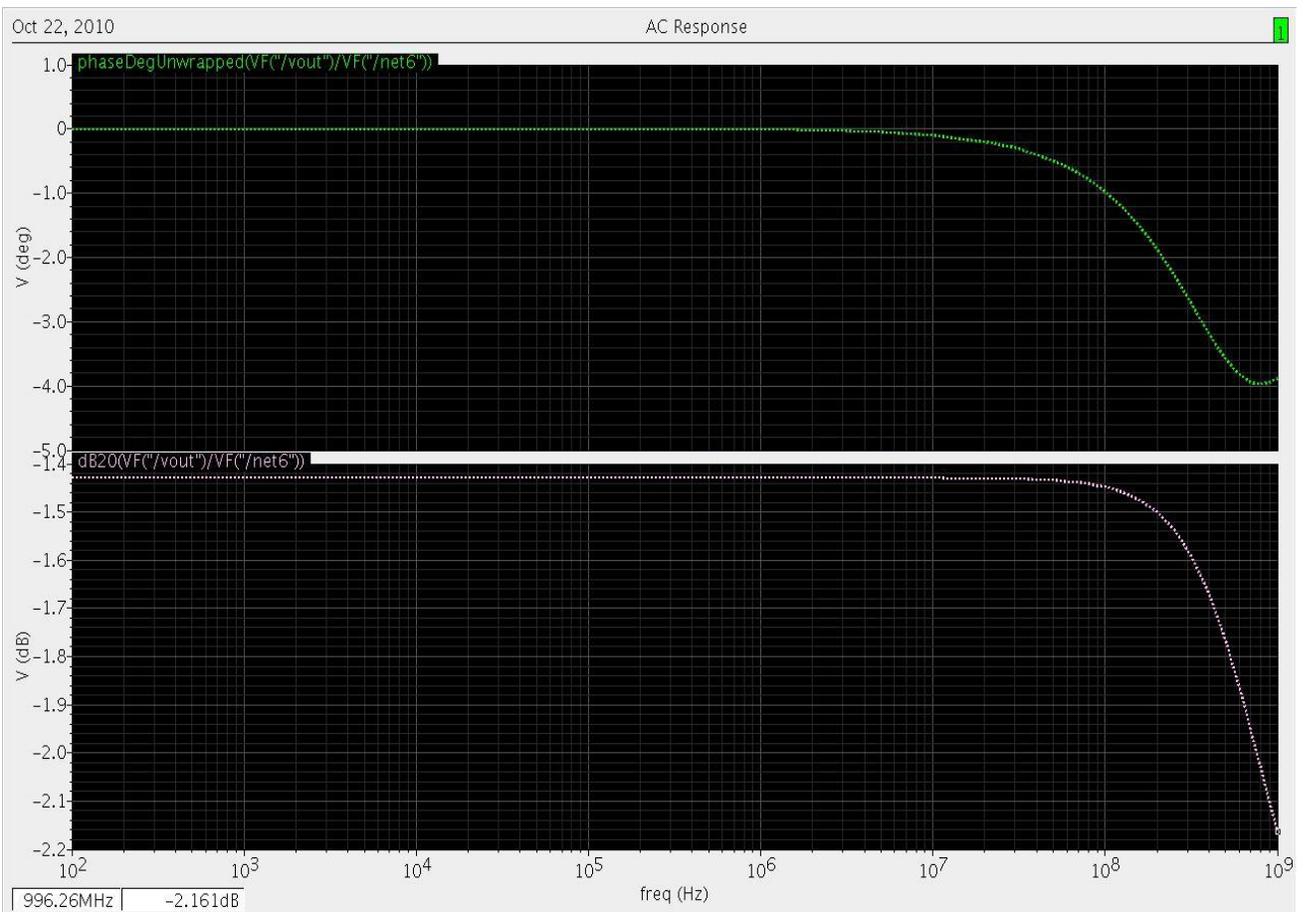
c) THEORY:

Common drain amplifier is a source follower or buffer amplifier circuit using a MOSFET. The output is simply equal to the input minus about 2.2V. The advantage of this circuit is that the MOSFET can provide current and power gain; the MOSFET draws no current from the input. It provides low output impedance to any circuit using the output of the follower, meaning that the output will not drop under load. Its output impedance is not as low as that of an emitter follower using a bipolar transistor (as you can verify by connecting a resistor from the output to -15V), but it has the advantage that the input impedance is infinite. The MOSFET is in saturation, so the current across it is determined by the gate-source voltage. Since a current source keeps the current constant, the gate-source voltage is also constant.

SCHEMATIC ENTRY:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

***** Subcircuits *****

.subckt DFFC Clk Clr Data Q QB Gnd Vdd

*----- Devices With SPICE.ORDER < 0.0 -----

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

*----- Devices With SPICE.ORDER == 0.0 -----

MM1p CB Clk Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
\$x=1293

+\$y=1800 \$w=414 \$h=600

MM2n CB Clk Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1293

+\$y=1000 \$w=414 \$h=600

MM3p C CB Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=2693

+\$y=1800 \$w=414 \$h=600

MM4n C CB Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=2693

\$y=1000

+\$w=414 \$h=600

MM5p 3 Data Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=3993

+\$y=5600 \$w=414 \$h=600

MM6p 4 C 3 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=3993

+\$y=4700 \$w=414 \$h=600

MM7n 4 CB 5 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

\$y=3900

+\$w=414 \$h=600

MM8n 5 Data Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

+\$y=3100 \$w=414 \$h=600

MM9p 6 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=5693

+\$y=5600 \$w=414 \$h=600

MM10p 4 CB 6 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=6107

+\$y=4700 \$w=414 \$h=600 \$m

MM11n 4 C 7 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3900

+\$w=414 \$h=600

MM12n 7 10 8 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3100

+\$w=414 \$h=600

MM13n 8 Clr Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

+\$y=2300 \$w=414 \$h=600

```

MM14p 9 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=5600 $W=414 $H=600
MM15p 4 CB 9 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=4700 $W=414 $H=600
MM16p 10 4 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7193
+$Y=3400 $W=414 $H=600
MM17n 10 4 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=7193
+$Y=2600 $W=414 $H=600
MM18p 11 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=5200 $W=414 $H=600
MM19p 12 CB 11 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=4400 $W=414 $H=600
MM20An 12 Clr 15 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=1393
+$Y=3600 $W=414 $H=600
MM20n 15 C 13 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2800 $W=414 $H=600
MM21n 13 10 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2000 $W=414 $H=600
MM22p 17 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=5200 $W=414 $H=600
MM23p 12 C 17 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=4400 $W=414 $H=600
MM24n 12 CB 17 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=3093
+$Y=3600 $W=414 $H=600
MM25Ap Q 17 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=5293
+$Y=4400 $W=414 $H=600
MM25p 17 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=4493
+$Y=5200 $W=414 $H=600
MM26An Q 17 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=5293
+$Y=3600 $W=414 $H=600
MM26n 17 Clr 14 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2800 $W=414 $H=600
MM27n 14 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2000 $W=414 $H=600
MM28Ap QB 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=8193
+$Y=4300 $W=414 $H=600
MM28p 16 12 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=6693
+$Y=4400 $W=414 $H=600
MM29An QB 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=8193
+$Y=3500 $W=414 $H=600
MM29n 16 12 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=6693
+$Y=3600 $W=414 $H=600
.ends

```

```

.subckt INV A Out Gnd Vdd
*----- Devices With SPICE.ORDER < 0.0 -----
* Design: Generic_250nm_LogicGates / Cell: INV / View: Main / Page:
* Designed by: Tanner EDA Library Development Team
* Organization: Tanner EDA - Tanner Research, Inc.
* Info: Inverter
* Date: 5/30/2008 4:06:39 PM
* Revision: 13 $ $x=7600 $y=600 $w=3600 $h=1200
*----- Devices With SPICE.ORDER == 0.0 -----
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $x=4593
+$y=2600 $w=414 $h=600
MM2p Out A Vdd Vdd PMOS25 W=3u L=250n M=2 AS=1.125p PS=3.75u AD=1.95p PD=7.3u $
+$x=4593 $y=3600 $w=414 $h=600
.ends

*----- Devices With SPICE.ORDER == 0.0 -----
***** Top Level *****
XDFFC_1 clk N_1 N_2 D N_2 Gnd Vdd DFFC $ $x=3100 $y=3400 $w=800 $h=1000
XDFFC_2 D N_1 N_4 C N_4 Gnd Vdd DFFC $ $x=4900 $y=3400 $w=800 $h=1000
XDFFC_3 C N_1 N_5 B N_5 Gnd Vdd DFFC $ $x=6700 $y=3400 $w=800 $h=1000
XDFFC_4 B N_1 N_6 A N_6 Gnd Vdd DFFC $ $x=8700 $y=3400 $w=800 $h=1000
XINV_1 rst N_1 Gnd Vdd INV $ $x=2300 $y=4100 $w=600 $h=400

*----- Devices With SPICE.ORDER > 0.0 -----
Vv1 Vdd Gnd DC 5 $ $x=800 $y=6000 $w=400 $h=600
Vv2 rst Gnd PULSE(0 5 0 1n 1n 95n 10u) $ $x=700 $y=3500 $w=400 $h=600
Vv3 clk Gnd PULSE(0 5 0 5n 5n 95n 200n) $ $x=2200 $y=2700 $w=400 $h=600
.PRINT TRAN V(A) $ $x=9550 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(B) $ $x=7950 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(C) $ $x=5950 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(clk) $ $x=1650 $y=2550 $w=300 $h=1500 $r=90
.PRINT TRAN V(D) $ $x=4250 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(rst) $ $x=950 $y=4550 $w=1500 $h=300 $r=180
***** Simulation Settings - Analysis Section *****
.tran 1u 5u
***** Simulation Settings - Additional SPICE Commands *****
.end

```

RESULT:

EXP.NO:	DESIGN AND SIMULATION OF COMMON GATE AMPLIFIER	
DATE:		

AIM:

To perform the Design and Simulation of Common drain amplifier circuit through schematic entry.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required for doing the experiment

S.No.	SOFTWARE REQUIREMENTS	QUANTITY
1	S-Edit, W-Edit, T-Edit using Tanner Tool.	1

b) Procedure for doing the experiment

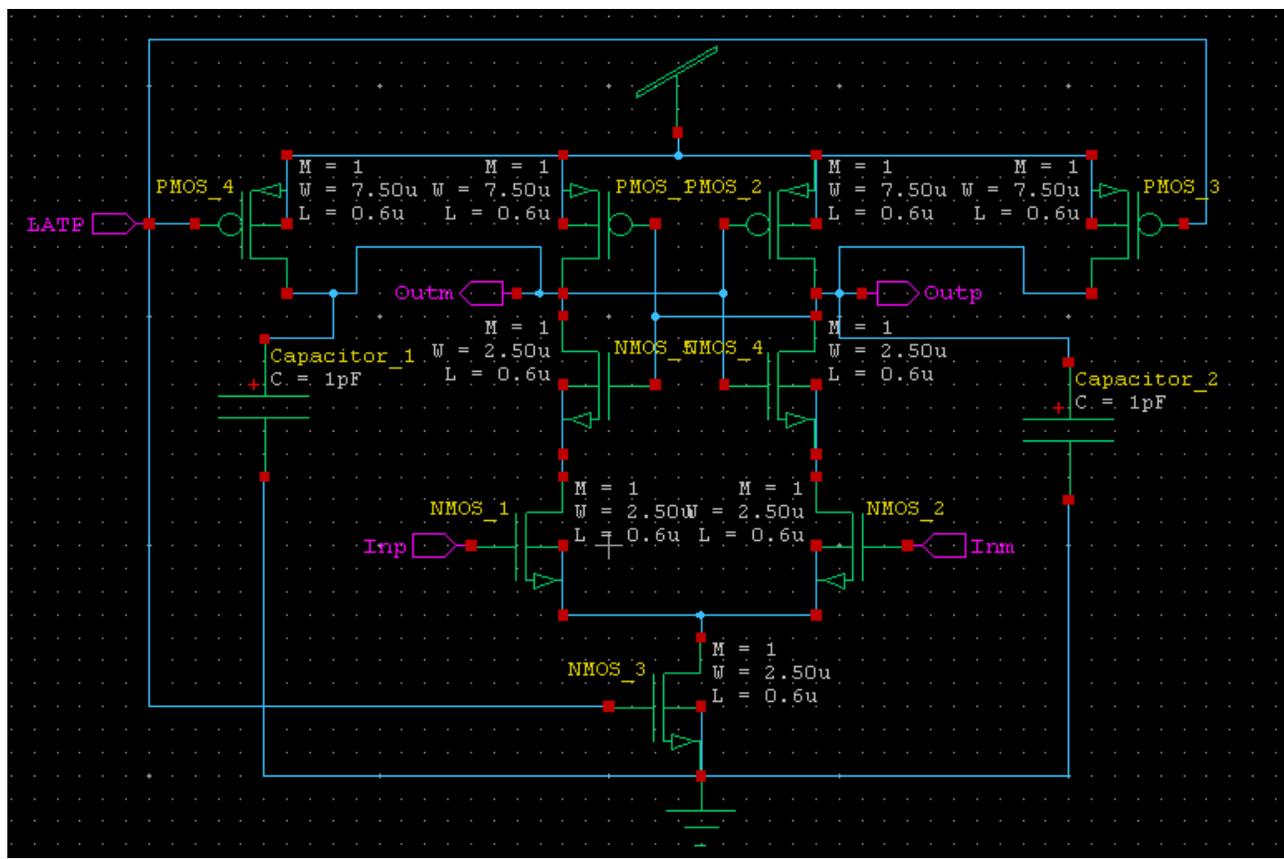
S.NO	DETAILS OF THE STEP
1	Draw the schematic of Common drain amplifier using S-edit
2	Perform Transient Analysis of Common drain amplifier circuit
3	Obtain the spice code using T-edit
4	Obtain the output wave form from W-edit

c) THEORY:

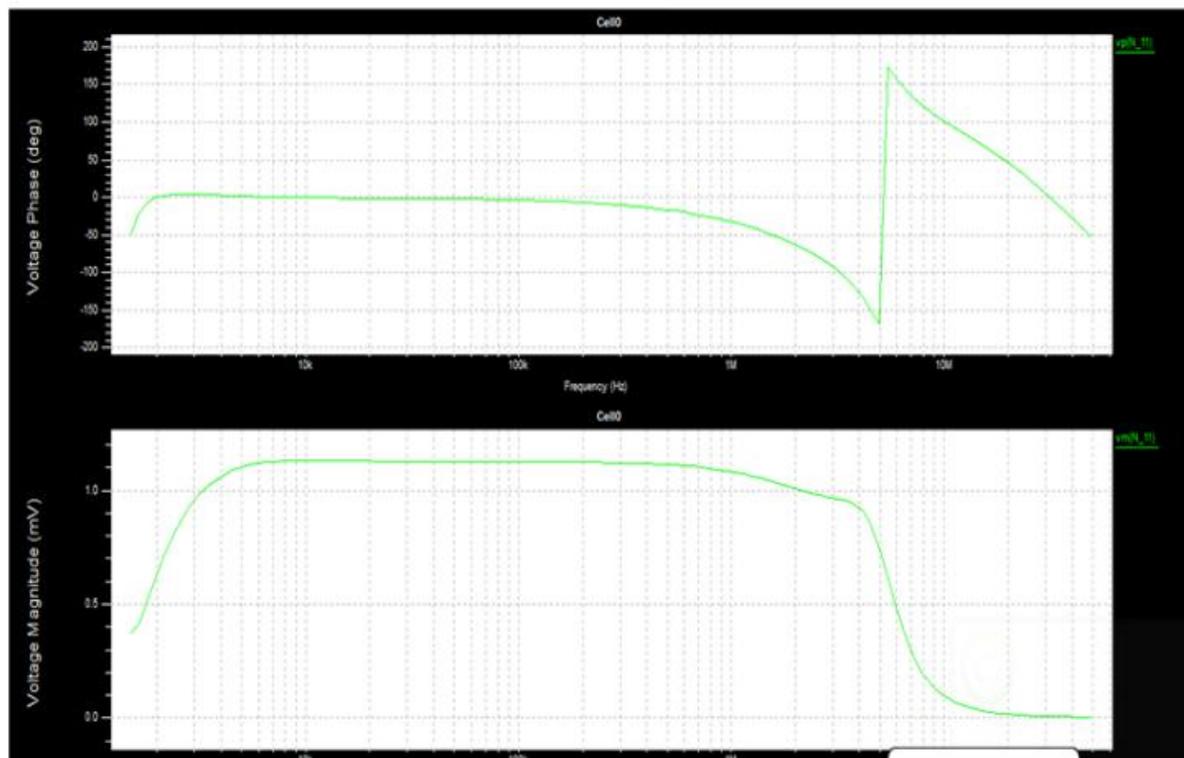
In common source amplifier and source follower circuits, the input signal are applied to the gate of a MOSFET. It is also possible to apply the input signal to the source terminal by keeping common gate terminal. This type of amplifier is called as common gate amplifier.

Figure below shows the CG amplifier in which the input signal is sensed at the source terminal and the output is produced at the drain terminal. The gate terminal is connected to V_B i.e. dc potential which will maintain the proper operating conditions.

SCHEMATIC ENTRY:



OUTPUT WAVEFORM:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

***** Subcircuits *****

.subckt DFFC Clk Clr Data Q QB Gnd Vdd

*----- Devices With SPICE.ORDER < 0.0 -----

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

* Design: Generic_250nm_LogicGates / Cell: DFFC / View: Main / Page:

* Designed by: Tanner EDA Library Development Team

* Organization: Tanner EDA - Tanner Research, Inc.

* Info: D Flip-Flop with Clear

* Date: 10/15/2008 12:04:43 PM

* Revision: 144 \$ \$x=7600 \$y=600 \$w=3600 \$h=1200

*----- Devices With SPICE.ORDER == 0.0 -----

MM1p CB Clk Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$
\$x=1293

+\$y=1800 \$w=414 \$h=600

MM2n CB Clk Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=1293

+\$y=1000 \$w=414 \$h=600

MM3p C CB Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=2693

+\$y=1800 \$w=414 \$h=600

MM4n C CB Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=2693

\$y=1000

+\$w=414 \$h=600

MM5p 3 Data Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=3993

+\$y=5600 \$w=414 \$h=600

MM6p 4 C 3 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$ \$x=3993

+\$y=4700 \$w=414 \$h=600

MM7n 4 CB 5 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

\$y=3900

+\$w=414 \$h=600

MM8n 5 Data Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=3993

+\$y=3100 \$w=414 \$h=600

MM9p 6 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=5693

+\$y=5600 \$w=414 \$h=600

MM10p 4 CB 6 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u \$

\$x=6107

+\$y=4700 \$w=414 \$h=600 \$m

MM11n 4 C 7 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3900

+\$w=414 \$h=600

MM12n 7 10 8 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

\$y=3100

+\$w=414 \$h=600

MM13n 8 Clr Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u \$ \$x=5693

+\$y=2300 \$w=414 \$h=600

```

MM14p 9 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=5600 $W=414 $H=600
MM15p 4 CB 9 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7293
+$Y=4700 $W=414 $H=600
MM16p 10 4 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=7193
+$Y=3400 $W=414 $H=600
MM17n 10 4 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=7193
+$Y=2600 $W=414 $H=600
MM18p 11 10 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=5200 $W=414 $H=600
MM19p 12 CB 11 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=1393
+$Y=4400 $W=414 $H=600
MM20An 12 Clr 15 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=1393
+$Y=3600 $W=414 $H=600
MM20n 15 C 13 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2800 $W=414 $H=600
MM21n 13 10 Gnd 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=1393
+$Y=2000 $W=414 $H=600
MM22p 17 Clr Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=5200 $W=414 $H=600
MM23p 12 C 17 Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=3093
+$Y=4400 $W=414 $H=600
MM24n 12 CB 17 0 NMOS25 W=1.50u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=3093
+$Y=3600 $W=414 $H=600
MM25Ap Q 17 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=5293
+$Y=4400 $W=414 $H=600
MM25p 17 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=4493
+$Y=5200 $W=414 $H=600
MM26An Q 17 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=5293
+$Y=3600 $W=414 $H=600
MM26n 17 Clr 14 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2800 $W=414 $H=600
MM27n 14 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=4493
+$Y=2000 $W=414 $H=600
MM28Ap QB 16 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=8193
+$Y=4300 $W=414 $H=600
MM28p 16 12 Vdd Vdd PMOS25 W=3.00u L=250n AS=1.95p PS=7.3u AD=1.95p PD=7.3u $
$X=6693
+$Y=4400 $W=414 $H=600
MM29An QB 16 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $
$X=8193
+$Y=3500 $W=414 $H=600
MM29n 16 12 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $X=6693
+$Y=3600 $W=414 $H=600
.ends

```

```

.subckt INV A Out Gnd Vdd
*----- Devices With SPICE.ORDER < 0.0 -----
* Design: Generic_250nm_LogicGates / Cell: INV / View: Main / Page:
* Designed by: Tanner EDA Library Development Team
* Organization: Tanner EDA - Tanner Research, Inc.
* Info: Inverter
* Date: 5/30/2008 4:06:39 PM
* Revision: 13 $ $x=7600 $y=600 $w=3600 $h=1200
*----- Devices With SPICE.ORDER == 0.0 -----
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u $ $x=4593
+$y=2600 $w=414 $h=600
MM2p Out A Vdd Vdd PMOS25 W=3u L=250n M=2 AS=1.125p PS=3.75u AD=1.95p PD=7.3u $
+$x=4593 $y=3600 $w=414 $h=600
.ends

*----- Devices With SPICE.ORDER == 0.0 -----
***** Top Level *****
XDFFC_1 clk N_1 N_2 D N_2 Gnd Vdd DFFC $ $x=3100 $y=3400 $w=800 $h=1000
XDFFC_2 D N_1 N_4 C N_4 Gnd Vdd DFFC $ $x=4900 $y=3400 $w=800 $h=1000
XDFFC_3 C N_1 N_5 B N_5 Gnd Vdd DFFC $ $x=6700 $y=3400 $w=800 $h=1000
XDFFC_4 B N_1 N_6 A N_6 Gnd Vdd DFFC $ $x=8700 $y=3400 $w=800 $h=1000
XINV_1 rst N_1 Gnd Vdd INV $ $x=2300 $y=4100 $w=600 $h=400

*----- Devices With SPICE.ORDER > 0.0 -----
Vv1 Vdd Gnd DC 5 $ $x=800 $y=6000 $w=400 $h=600
Vv2 rst Gnd PULSE(0 5 0 1n 1n 95n 10u) $ $x=700 $y=3500 $w=400 $h=600
Vv3 clk Gnd PULSE(0 5 0 5n 5n 95n 200n) $ $x=2200 $y=2700 $w=400 $h=600
.PRINT TRAN V(A) $ $x=9550 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(B) $ $x=7950 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(C) $ $x=5950 $y=4550 $w=300 $h=1500 $r=270
.PRINT TRAN V(clk) $ $x=1650 $y=2550 $w=300 $h=1500 $r=90
.PRINT TRAN V(D) $ $x=4250 $y=4450 $w=300 $h=1500 $r=270
.PRINT TRAN V(rst) $ $x=950 $y=4550 $w=1500 $h=300 $r=180
***** Simulation Settings - Analysis Section *****
.tran 1u 5u
***** Simulation Settings - Additional SPICE Commands *****
.end

```

RESULT:

EXP.NO:	DESIGN AND SIMULATION OF DIFFERENTIAL AMPLIFIER	
DATE:		

AIM:

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

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required for doing the experiment

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

b) Procedure for doing the experiment

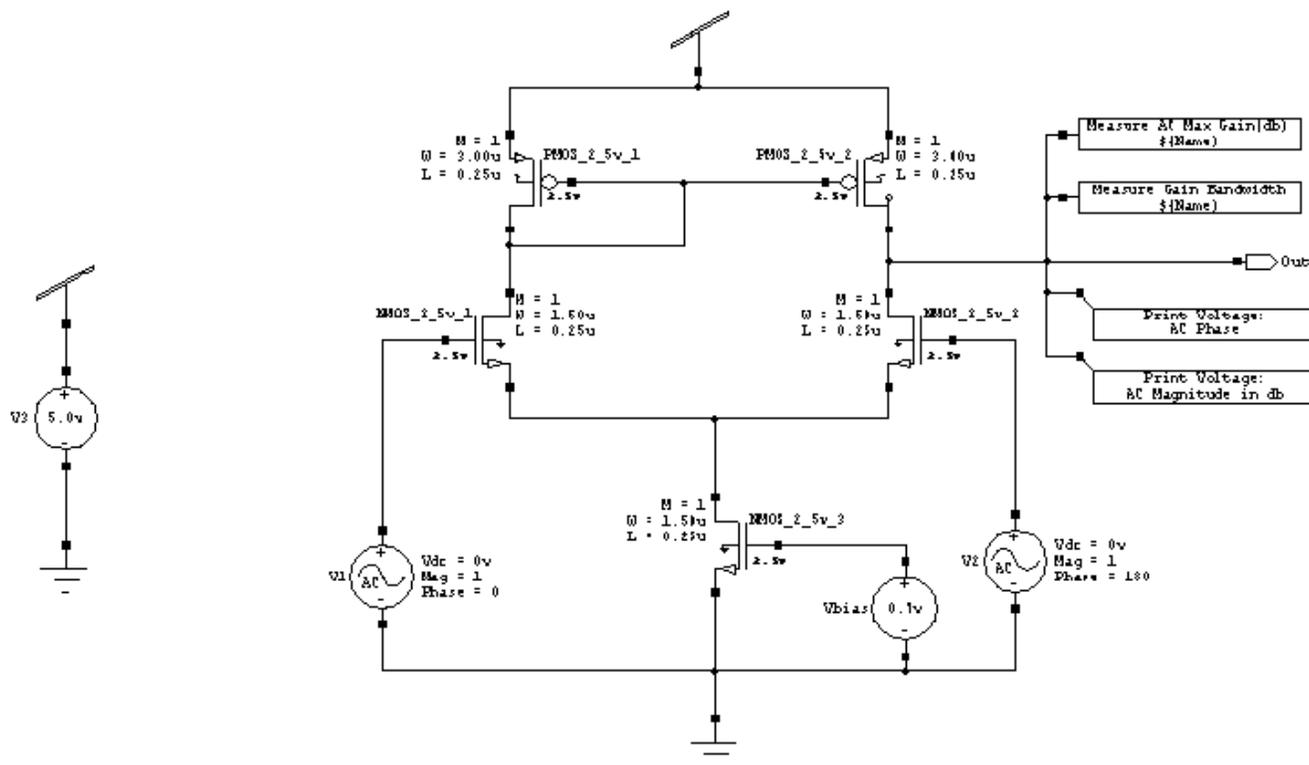
S.No	Details of the step
1	Draw the schematic of differential amplifier using S-edit and generate the Symbol.
2	Draw the schematic of differential amplifier circuit using the generated Symbol.
3	Perform AC Analysis of the differential amplifier.
4	Obtain the frequency response from W-edit.
5	Obtain the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

PROCEDURE:

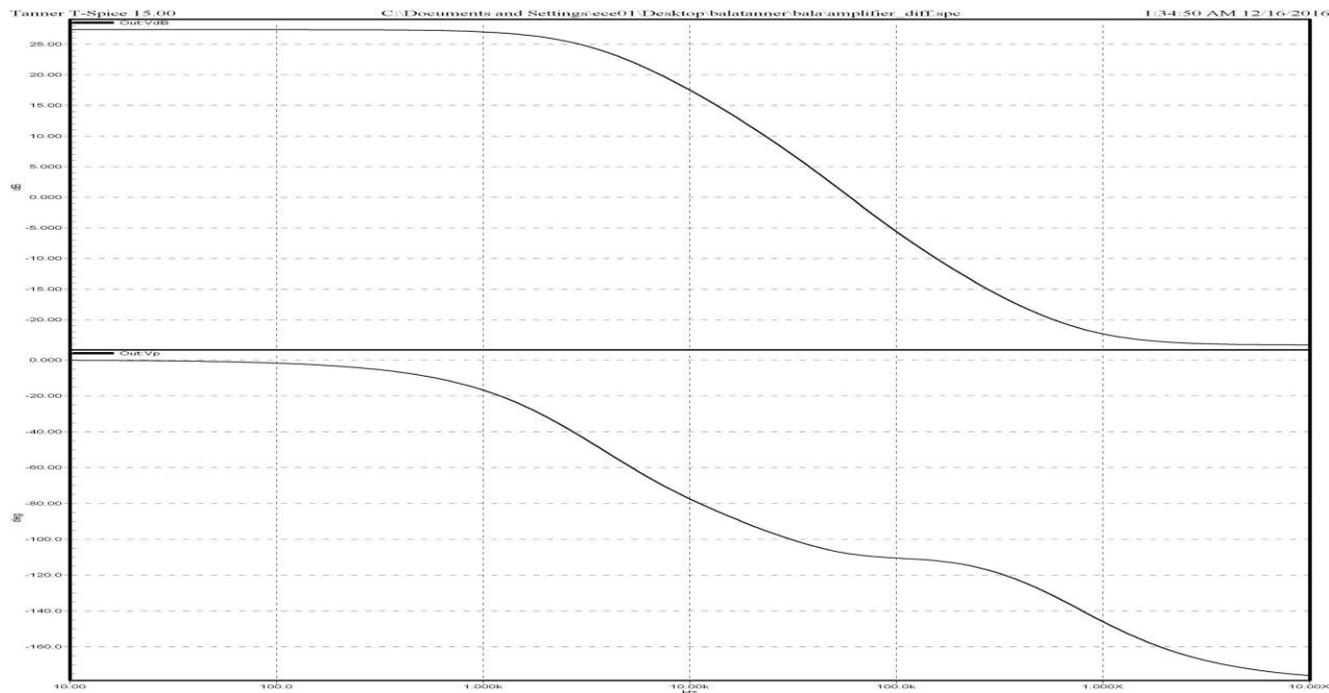
- Enter the schematic of differential amplifier using S-Edit.
- Perform AC Analysis of the differential amplifier.
- Go to ‘setup’ in that select ‘spice simulation’. Choose ‘ac analysis’ and give the following values.
- Set ‘Start frequency =10’, ‘Stop frequency=10meg’, ‘No. of frequency=25’, ‘Sweep type = dec’. Click on ‘general’ type and give path to Generic_250nm.lib. Then Click OK.
- RUN Simulation to get output.
- Obtain the frequency response from W-Edit.
- Obtain the spice code using T-Edit.

SCHEMATIC DIAGRAM:

DIFFERENTIAL MODE:



DIFFERENTIAL MODE OUTPUT:



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

*----- Devices With SPICE.ORDER == 0.0 -----

***** Top Level *****

MNMOS_2_5v_1 N_1 N_3 N_4 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f
PD=4.3u

+\$ \$x=3793 \$y=4300 \$w=414 \$h=600

MNMOS_2_5v_2 Out N_5 N_4 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u

+\$ \$x=6607 \$y=4300 \$w=414 \$h=600 \$m

MNMOS_2_5v_3 N_4 N_6 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f
PD=4.3u

+\$ \$x=5507 \$y=3000 \$w=414 \$h=600 \$m

MPMOS_2_5v_1 N_1 N_1 Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p
PD=7.3u

+\$ \$x=4207 \$y=5300 \$w=414 \$h=600 \$m

MPMOS_2_5v_2 Out N_1 Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p
PD=7.3u

+\$ \$x=6193 \$y=5300 \$w=414 \$h=600

*----- Devices With SPICE.ORDER > 0.0 -----

VV3 Vdd Gnd DC 5 \$ \$x=1200 \$y=3800 \$w=400 \$h=600

VVbias N_6 Gnd DC 700m \$ \$x=6500 \$y=2600 \$w=400 \$h=600

VV1 N_3 Gnd DC 0 AC 1 0 \$ \$x=3200 \$y=2800 \$w=400 \$h=600

VV2 N_5 Gnd DC 0 AC 1 180 \$ \$x=7200 \$y=2900 \$w=400 \$h=600

.PRINT AC Vdb(Out) \$ \$x=8350 \$y=4050 \$w=1500 \$h=300

.PRINT AC Vp(Out) \$ \$x=8350 \$y=4450 \$w=1500 \$h=300

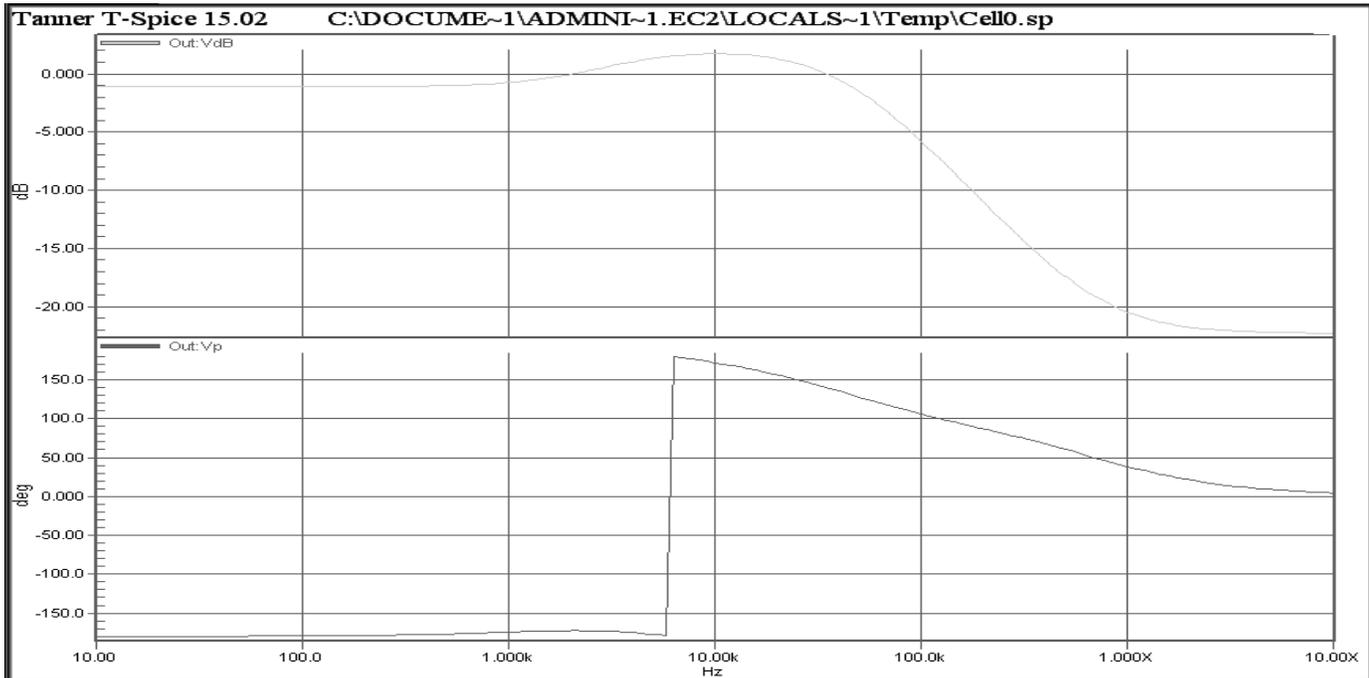
.MEASURE AC AC_Measure_Gain_1 MAX vdb(Out) ON \$ \$x=8250 \$y=5600 \$w=1500 \$h=200

.MEASURE AC AC_Measure_GainBandwidthProduct_1_Gain MAX vdb(Out) OFF

.MEASURE AC AC_Measure_GainBandwidthProduct_1_UGFreq WHEN Vdb(Out)=0 OFF

.MEASURE AC AC_Measure_GainBandwidthProduct_1

PARAM='AC_Measure_GainBandwidthProduct_1_Gain*AC_Measure_GainBandwidthProduct_1_
UGFreq' +ON \$ \$x=8250 \$y=5200 \$w=1500 \$h=200



NETLIST:

***** Simulation Settings - General Section *****

.lib "C:\Documents and Settings\ece01\My Documents\Tanner EDA\Tanner Tools
v15.0\Process\Generic_250nm\Generic_250nm_Tech\Generic_250nm.lib" TT

*----- Devices With SPICE.ORDER == 0.0 -----

***** Top Level *****

MNMOS_2_5v_1 N_1 N_2 N_3 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f
PD=4.3u

+\$ \$x=3793 \$y=4300 \$w=414 \$h=600

MNMOS_2_5v_2 Out N_2 N_3 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f PD=4.3u

+\$ \$x=6607 \$y=4300 \$w=414 \$h=600 \$m

MNMOS_2_5v_3 N_3 N_5 Gnd 0 NMOS25 W=1.5u L=250n AS=975f PS=4.3u AD=975f
PD=4.3u

+\$ \$x=5507 \$y=3000 \$w=414 \$h=600 \$m

MPMOS_2_5v_1 N_1 N_1 Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p
PD=7.3u

+\$ \$x=4207 \$y=5300 \$w=414 \$h=600 \$m

MPMOS_2_5v_2 Out N_1 Vdd Vdd PMOS25 W=3u L=250n AS=1.95p PS=7.3u AD=1.95p
 PD=7.3u

+\$ \$x=6193 \$y=5300 \$w=414 \$h=600

*----- Devices With SPICE.ORDER > 0.0 -----

VV3 Vdd Gnd DC 5 \$ \$x=1200 \$y=3800 \$w=400 \$h=600

VVbias N_5 Gnd DC 700m \$ \$x=6500 \$y=2600 \$w=400 \$h=600

VV1 N_2 Gnd DC 0 AC 1 0 \$ \$x=3200 \$y=2800 \$w=400 \$h=600

.PRINT AC Vdb(Out) \$ \$x=8350 \$y=4050 \$w=1500 \$h=300

.PRINT AC Vp(Out) \$ \$x=8350 \$y=4450 \$w=1500 \$h=300

.MEASURE AC AC_Measure_Gain_1 MAX vdb(Out) ON \$ \$x=8250 \$y=5600 \$w=1500 \$h=200

.MEASURE AC AC_Measure_GainBandwidthProduct_1_Gain MAX vdb(Out) OFF

.MEASURE AC AC_Measure_GainBandwidthProduct_1_UGFreq WHEN Vdb(Out)=0 OFF

.MEASURE AC AC_Measure_GainBandwidthProduct_1

PARAM='AC_Measure_GainBandwidthProduct_1_Gain*AC_Measure_GainBandwidthProduct_1_UGFreq'

+ON \$ \$x=8250 \$y=5200 \$w=1500 \$h=200

***** Simulation Settings - Analysis Section *****

.ac dec 25 10 10X

***** Simulation Settings - Additional SPICE Commands *****

.end

MEASUREMENT RESULT SUMMARY:

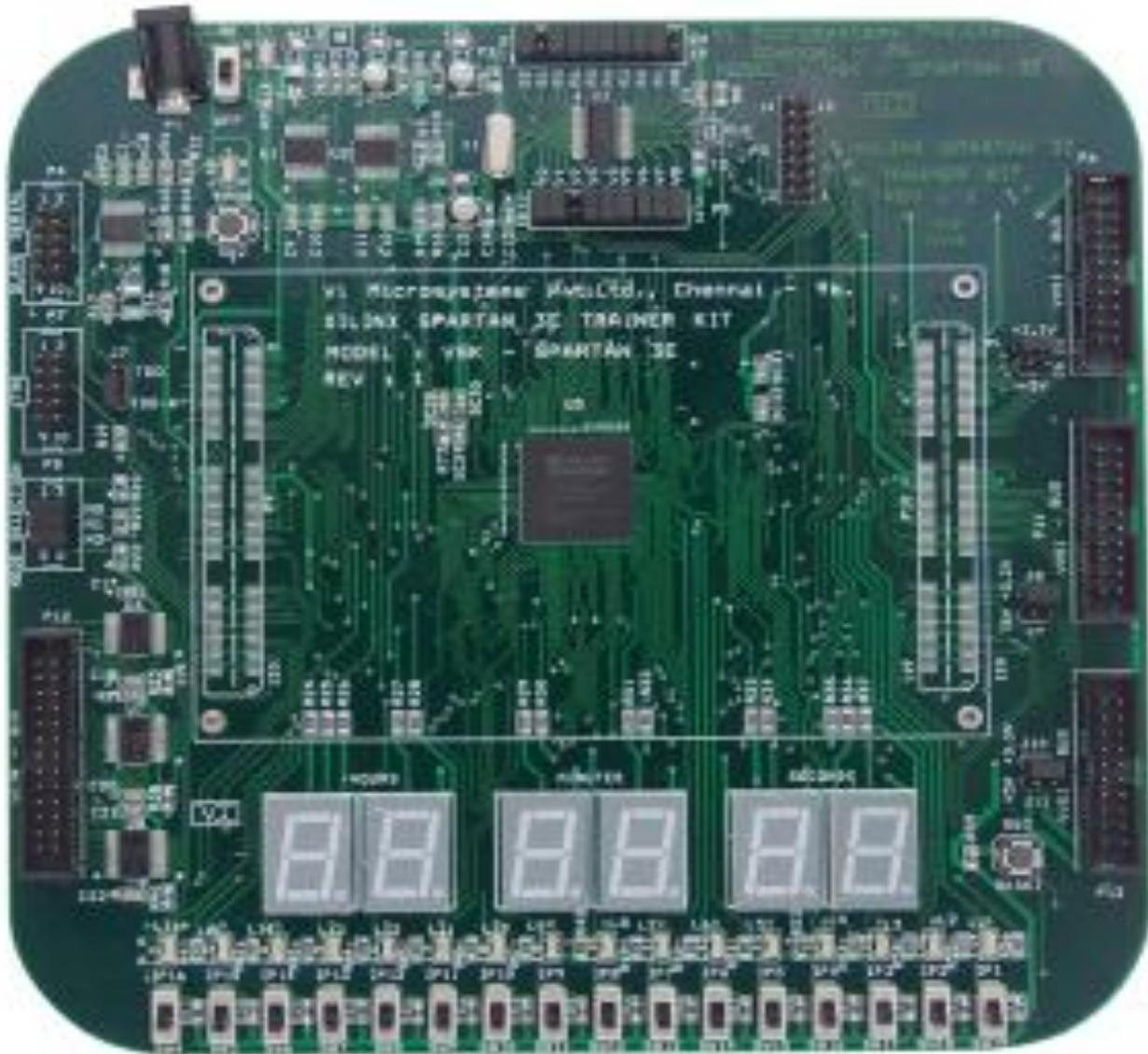
DIFFERENTIAL AMPLIFIER		
	COMMOM MODE	DIFFERNTIAL MODE
AC_Measure_Gain	Ac=	Ad=
AC_Measure_GainBandwidthProduct		

$$\text{CMRR} = A_d / A_c$$

=

RESULT

APPENDIX (FPGA PIN DETAILS)



CLOCK SOURCE:\

CLOCK INPUT	FPGA PIN
CLK	A8

APPENDIX (FPGA PIN DETAILS)

Slide Switch connections with FPGA

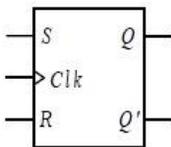
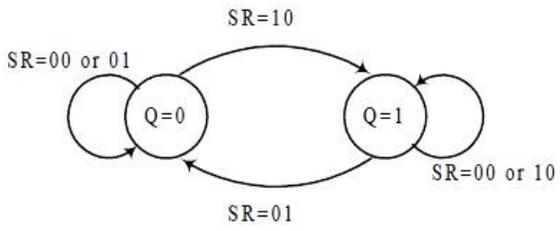
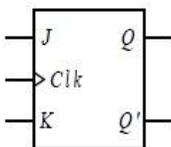
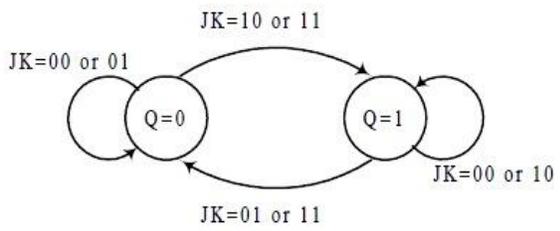
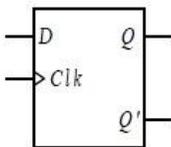
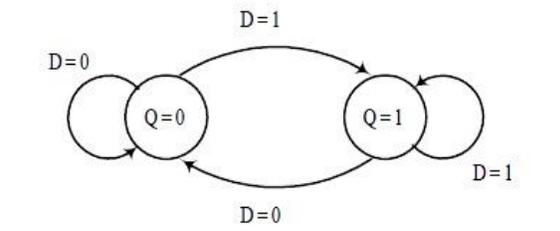
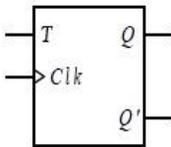
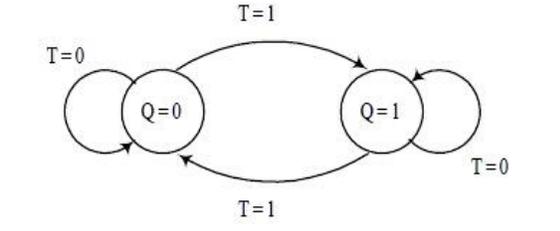
SWITCHES	FPGA PINS
SW4	T14
SW5	T12
SW6	T9
SW7	T7
SW8	T2
SW9	G12
SW10	H1
SW11	R3
SW12	N11
SW13	N3
SW14	M13

LED connections with FPGA

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

Type of operation	Operator symbol	Description	Number of operands
Arithmetic	+	addition	2
	-	subtraction	2
	*	multiplication	2
	/	division	2
	%	modulus	2
	**	exponentiation	2
Shift	>>	logical right shift	2
	<<	logical left shift	2
	>>>	arithmetic right shift	2
	<<<	logical left shift	2
Relational	>	greater than	2
	<	less than	2
	>=	greater than or equal to	2
	<=	less than or equal to	2
Equality	==	equality	2
	!=	inequality	2
	===	case equality	2
	!==	case inequality	2
Bitwise	~	bitwise negation	1
	&	bitwise and	2
		bitwise or	2
	^	bitwise xor	2
Reduction	&	reduction and	1
		reduction or	1
	^	reduction xor	1
Logical	!	logical negation	1
	&&	logical and	2
		logical or	2
Concatenation	{ }	concatenation	any
	{ { } }	replication	any
Conditional	? :	conditional	3

Operator	Precedence
! ~ + - (unary)	highest
**	
* / %	
+ - (binary)	
>> << >>> <<<	
< <= > >=	
== != === !==	
&	
^	
&&	
?:	lowest

Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table																																																								
SR 	<table border="1"> <thead> <tr> <th>S</th> <th>R</th> <th>Q</th> <th>Q_{next}</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>×</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>×</td></tr> </tbody> </table>	S	R	Q	Q _{next}	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	1	1	1	0	×	1	1	1	×	 <p>SR=00 or 01</p> <p>SR=10</p> <p>SR=01</p> <p>SR=00 or 10</p> $Q_{next} = S + R'Q$ $SR = 0$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_{next}</th> <th>S</th> <th>R</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>×</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>×</td><td>0</td></tr> </tbody> </table>	Q	Q _{next}	S	R	0	0	0	×	0	1	1	0	1	0	0	1	1	1	×	0
S	R	Q	Q _{next}																																																								
0	0	0	0																																																								
0	0	1	1																																																								
0	1	0	0																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	0	1	1																																																								
1	1	0	×																																																								
1	1	1	×																																																								
Q	Q _{next}	S	R																																																								
0	0	0	×																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	1	×	0																																																								
JK 	<table border="1"> <thead> <tr> <th>J</th> <th>K</th> <th>Q</th> <th>Q_{next}</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	J	K	Q	Q _{next}	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	 <p>JK=00 or 01</p> <p>JK=10 or 11</p> <p>JK=01 or 11</p> <p>JK=00 or 10</p> $Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J'+J) + JQ'(K'+K)$ $= K'Q + JQ'$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_{next}</th> <th>J</th> <th>K</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>×</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>×</td></tr> <tr><td>1</td><td>0</td><td>×</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>×</td><td>0</td></tr> </tbody> </table>	Q	Q _{next}	J	K	0	0	0	×	0	1	1	×	1	0	×	1	1	1	×	0
J	K	Q	Q _{next}																																																								
0	0	0	0																																																								
0	0	1	1																																																								
0	1	0	0																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	0	1	1																																																								
1	1	0	1																																																								
1	1	1	0																																																								
Q	Q _{next}	J	K																																																								
0	0	0	×																																																								
0	1	1	×																																																								
1	0	×	1																																																								
1	1	×	0																																																								
D 	<table border="1"> <thead> <tr> <th>D</th> <th>Q</th> <th>Q_{next}</th> </tr> </thead> <tbody> <tr><td>0</td><td>×</td><td>0</td></tr> <tr><td>1</td><td>×</td><td>1</td></tr> </tbody> </table>	D	Q	Q _{next}	0	×	0	1	×	1	 <p>D=0</p> <p>D=1</p> <p>D=0</p> <p>D=1</p> $Q_{next} = D$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_{next}</th> <th>D</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	Q	Q _{next}	D	0	0	0	0	1	1	1	0	0	1	1	1																																
D	Q	Q _{next}																																																									
0	×	0																																																									
1	×	1																																																									
Q	Q _{next}	D																																																									
0	0	0																																																									
0	1	1																																																									
1	0	0																																																									
1	1	1																																																									
T 	<table border="1"> <thead> <tr> <th>T</th> <th>Q</th> <th>Q_{next}</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	T	Q	Q _{next}	0	0	0	0	1	1	1	0	1	1	1	0	 <p>T=0</p> <p>T=1</p> <p>T=0</p> <p>T=1</p> $Q_{next} = TQ' + T'Q = T \oplus Q$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_{next}</th> <th>T</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	Q	Q _{next}	T	0	0	0	0	1	1	1	0	1	1	1	0																										
T	Q	Q _{next}																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									
Q	Q _{next}	T																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									