SEMESTER

7

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Department of Electronics & Communication Engineering

Name:__________________________________

USN :__________________________________

Manual for VLSI Laboratory (15ECL77)

DEPT. OF ELECTRONICS AND COMMUNICATION ENGINEERING

VLSI Laboratory

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belwadi, SR Patna, Mandya Dist. 571438

Vision of the Institute

'To Gift the Nation with Eminent Engineers and Managers, Capable of Contributing Towards Technological Advancement and Betterment of the Society.'

Mission of the Institute

1. To advance the well-being of society through excellence in teaching, research and service that exploits the rapidly changing technical diversity via a collaborative environment that stimulates faculty, staff and students to reach their highest potential through continuous learning.
2. Instill the foundation of professionalism and provide the tools required to advance the technological world.
3. Inculcate knowledge and ethics, and nurture/foster innovation and team man ship among the graduates/alumnae.
4. Endow eminent education to serve the society with graduates/alumnae possessing ace skills and potential.
5. Sustain highest reception of the institute's alumnae among the stakeholders.

Vision of the Department

'To be recognized by the society at large as offering Value based Quality Education to groom the next generation entrepreneurs, leaders and researchers in the field of Electronics and Communication to meet the challenges at global level.'

Mission of the Department

1. To groom the students with strong foundations of Electronics and Communication Engineering and to facilitate them to pursue higher education and research.
2. To educate and prepare the students to be competent to face the challenges of the industry/society and /or to become successful entrepreneurs.
3. Provide ethical and value-based education by promoting activities addressing the societal needs.
4. Enable students to develop skills to solve complex technological problems of current times and also provide a framework for promoting collaborative and multidisciplinary activities.

Program Educational Objectives

1. To prepare the students to be able to have a successful career in dynamic industry that is global, multi-disciplinary, and evolving;
2. Develop their engineering skills in problem solving, design and innovation as they work individually and/or in multi-disciplinary teams with sense of professional ethics and social responsibility.
3. Communicate effectively and manage resources skillfully as members and leaders of the profession.

Program Specific Outcomes:

1. An ability to apply the basic concepts of engineering science into various areas of Electronics & Communication Engineering.
2. An ability to solve complex Electronics and Communication Engineering problems, using state of the art hardware and software tools, along with analytical skills to arrive at cost effective and efficient solutions.
3. Wisdom of social and environmental awareness along with ethical responsibility to have a successful career and to sustain passion and zeal for real-world applications using optimal resources.

Do’s:

1. Students must bring observation/Manual book along with pen, pencil and eraser etc.
2. Before entering to lab must prepare for viva for which they are going to conduct experiment.
3. Strictly follow the procedures for conduction of experiments.
4. Any breakdown/failure of equipment must be reported to the technical staff immediately.
5. Uniform and ID card are must.
6. Maintain silence inside the laboratory
7. Keep your belongings in designated area.
8. Chairs should be kept under the workbenches when not in use.
9. Sit upright on chairs, keeping feet on the floor.
10. Every student should know the location and operating procedures of all Safety equipment including First Aid Kit and Fire extinguisher.

DONT’S:

1. Don’t eat food, drink beverages or chew gum in the laboratory.
2. Don’t touch any live terminals.
3. Don’t leave the experiment table unattended .

Syllabus

PART A DIGITAL DESIGN

Write Verilog code for following circuits and their Testbench for **verification,** observe the wave waveform **synthesis** the code with technological library with given constraints, do the initial timing verification with gate level simulation.

1. Inverter
2. Buffer
3. Transmission gate
4. Basic- universal gates
5. Realization of basic gates using NAND & NOR
6. Flip flops, T,D,RS,JK,MSJK
7. Adders-Serial and parallel

8.Counters-Asynchronous up and down
Synchronous up /down

PART B (ANALOG DESIGN)

1.Design an **Inverter** with given specifications, and completing the design flow mentioned below:

a. Draw the Schematic and verify the following:

i. DC analysis ii. Transient analysis

b. Draw the layout and verify the DRC

2.Design the following circuits with given specifications and completing the design flow mentioned below:

a. Draw the Schematic and verify the following

i. DC analysis ii.AC analysis iii. Transient analysis.

b. Draw the layout and verify the DRC.

i) Single stage differential amplifier.

(ii) Common Source amplifier

(iii) Common Drain amplifier

3.Design an **OP-AMP** with given specifications, using the differential amplifier common source and common drain amplifier in library and completing the design flow mentioned below:

a. Draw the schematic and verify the following

i. DC analysis ii.AC analysis iii. Transient analysis

b. Draw the layout and verify the DRC

4. Design a **4 bit R-2R ladder DAC** for the given specifications and completing the design flow mentioned below using given OP-AMP in the library

a. Draw the schematic and verify the following

i. DC analysis ii. AC analysis iii. Transient analysis

b. Draw the layout and verify the DRC

c .Verify the theoretical and practical values.

Course Outcomes

Course Name: VLSI Laboratory **Course Code: 15ECL77**

CO's	DESCRIPTION OF THE OUTCOMES
15ECL77.1	**Understand** and **develop** the testbenchcode for combinational circuits and sequential circuits
15ECL77.2	**Simulate** various combinational and sequential logic circuits and **verify** the simulation with truth table through EDA tools
15ECL77.3	**Document** the experimental process and corresponding outcomes.

Scheme of Evaluation

1. Lab manuals will be evaluated for 10 marks (CO1, CO2).
2. Lab record will be evaluated for 10 marks (CO3)
3. Internal Assessment will be conducted for 20marks; Write-up (CO3-5M), Conduction (CO2-10M), Viva-Voce (CO1-5M).
4. Average of (Lab Manual Marks + Lab Record Marks) and Internal Assessment will be considered for final IA marks.

Conduct of Practical Examination

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script for breakup of marks.
4. Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero.

Contents

THEORY ASPECT OF HARDWARE DESCRIPTION LANGUAGE(VERILOG)

Hardware description languages such as verilog, differ from software programming languages because they include ways of describing the propagation of time and signal dependencies(sensitivity).Their are two assignment operator blocking assingment(=)and an non blocking assignment(<=).The non blocking assignment allows designers to describe a state-machine update without needing to declare and use temporary storage variables. Since these concepts are part of verilog language semantics, designers could quickly write descriptions of large circuits, in a relatively compact and concise form .At the time of verilog introduction (1984) .Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits.

The designers of verilog wanted a language with syntax similar to C programming language, which was already widely used in engineering software development. A verilog is case sensitive, has a basic preprocessor and equivalent control flow keywords (ifelse,while,for,case etc) and compatible operator precedence. Syntactic differences include variable declaration; verilog requires bit-widths on net/reg types, demarcation of procedural blocks such as begin/ end and many other minor differences.

A verilog design consist of a hierarchy of modules .Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output and in out ports. Internally a module can contain any combination of the following-Net/variable declaration such as wire, register, integer etc, concurrent and sequential statement blocks and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order with in the block. But the block themselves are executed concurrently, qualifing verilog as a dataflow language.

Verilog concept “wire” consists of both signal values (4-state:”1,0,floating,undefined”) and strengths(strong, weak, etc.).This system allows abstract modeling of shared signal-lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire’s readable value is resolved by a function of the source drivers and their strengths.

A subset of statements in the verilog language is synthesizable .Verilog modules that conform to a synthesizable coding style known as RTL can be physically realized by synthesis software. This software algorithmically transforms the abstract verilog to a netlist a logically-equivalent description consisting only elementary logic primitives such as AND, OR, NOT,FLIPFLOPS etc. That are available in a specific FPGA or VLSI technology further manipulations to the netlist ultimately lead to a circuit fabrication blueprint such as a photo mask set for an ASIC, or a bit stream file for an FPGA.

OPERATORS

OPERATOR TYPE	OPERATOR SYMBOLS	OPERATION PERFORMED
Bitwise	**~**	1'compliment
	&	Bitwise AND
	|	Bitwise OR
	^	Bitwise XOR
	~^ or ^~	Bitwise XNOR
Logical	**!**	NOT
	&&	AND
	||	OR
Reduction	**&**	Reduction AND
	~&	Reduction NAND
	|	Reduction OR
	~|	Reduction NOR
	^	Reduction XOR
	~^ or ^~	Reduction XNOR

Arithmetic	+	Addition
	-	Subtraction
	-	2’s compliment
	*	Multiplication
	/	Division
	**	Exponent
Relational	>	Greater than
	<	Lesser than
	>=	Greater than or equal to
	<=	Lesser than or equal to
	==	Logical equality
	!=	Logical inequality
	===	4-state logical equality
	!==	4-state logical inequality
Shift	>>	Logical right shift
	<<	Logical left shift
	>>>	Arithmetic right shift
	<<<	Arithmetic left shift
Concatenation	{,}	Concatenation
Replication	{n{m}}	Replicate value m for n times
Conditional	?:	Conditional

PROCEDURE:

The Procedure to be followed for Software and Hardware Programs are as follows:

Step 1: Go to Start Menu $\longrightarrow$ All Programs $\longrightarrow$ Xilinx ISE 9.1i and Select Project Navigator.

Step 2: Go to File Menu and Close any previously opened project if any, and then Select New Project.

Step 3: Enter the Project name and location and Select the Top level module type as HDL.

Step 4: Select the product category as General purpose, Device family and Device name as Spartan3 and XC3S50, pin density PQ208, speed as -5 for FPGA.

Step 5: Right click on the source file and select New source speed followed by Verilog module and Give the file name same as the name of the entity.

Step 6: Define the ports used and their respective directions in the next window that opens.

Step 7: Write the both Verilog code & test bench code in the work space that opens and save the file after completion of editing.

Step 8: Go to the Process view window and right click on the Synthesize - XST and then click on check syntax. Correct the errors if any.

Step 9: Make the alterations in the Clock information and initial length of the test bench if needed.

Step 10: Go to Process view and under Xilinx ISE Simulator Right click on the Simulate Behavioral model to see the output for the input conditions.

PART A

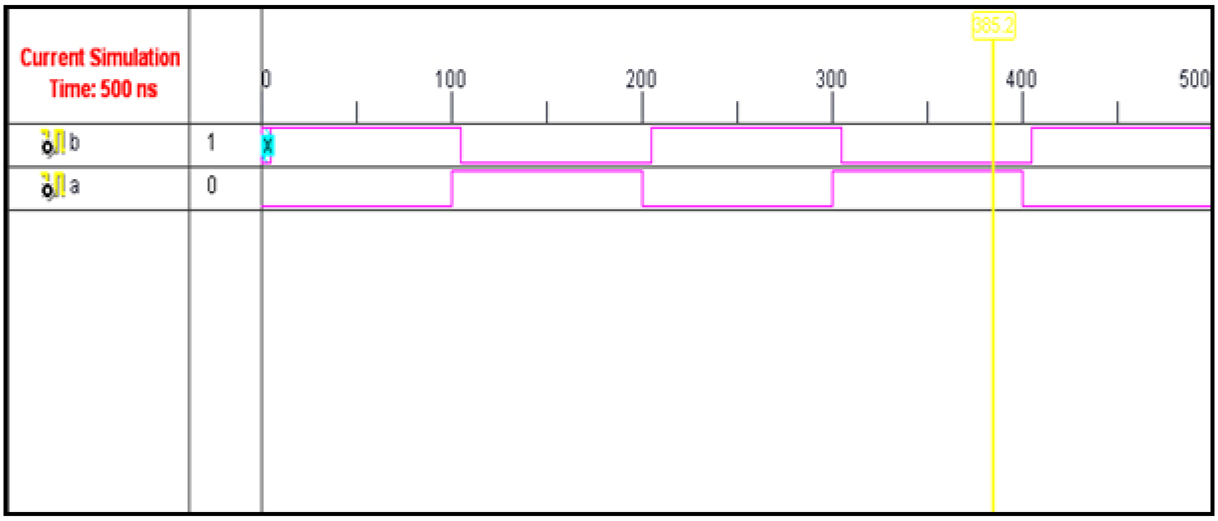
Digital Design

1.Inverter-Verilog code and Testbench

```
module inv(a, b);
 input a;
output b;
assign b=~a;
endmodule
```

//TESTBENCH

```
module inv_tb();
reg t_a;
wire t_b;
inv mygate(.a(t_a),.b(t_b));
initial
begin
$monitor (t_a,t_b);
t_a=1'b0;
#5 t_a=1'b1;
#5 t_a=1'b0;
#5 t_a=0'b1;
#5 t_a=0'b0;
end
endmodule
```

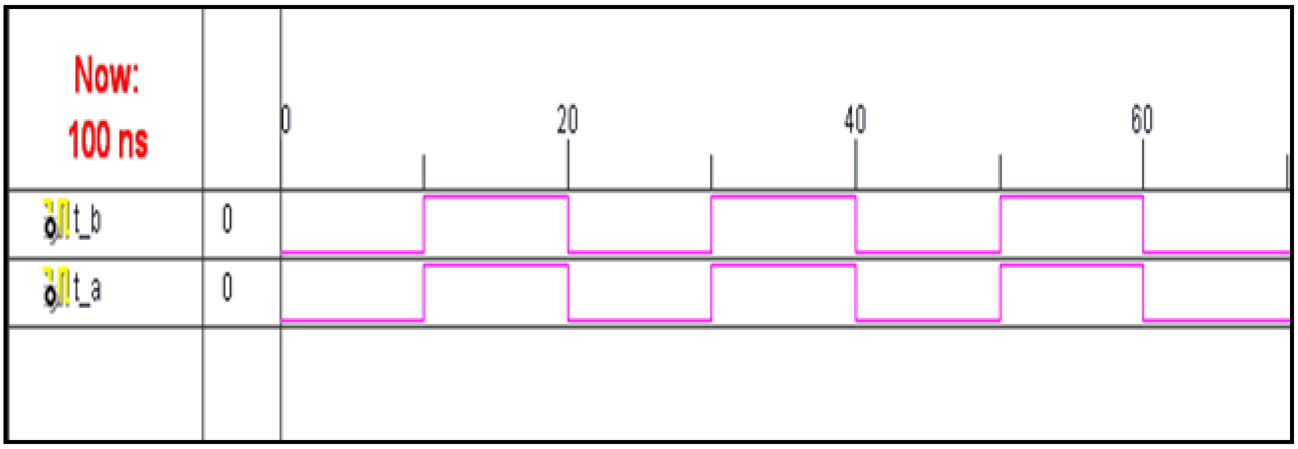


2.Buffer-verilog code and Test bench

```
module buff(a, b);
 input a;
 output b;
assign b=a;
endmodule
```

//TESTBENCH

```
module buff_tb();
reg t_a;
wire t_b;
buff mygate (.a(t_a),.b(t_b));
initial
begin
$monitor (t_a,t_b);
t_a=1'b0;
#10 t_a=1'b1;
#10 t_a=0'b0;
#10 t_a=1'b1;
#10 t_a=0'b0;
#10 t_a=1'b1;
#10 t_a=0'b0;
end
endmodule
```

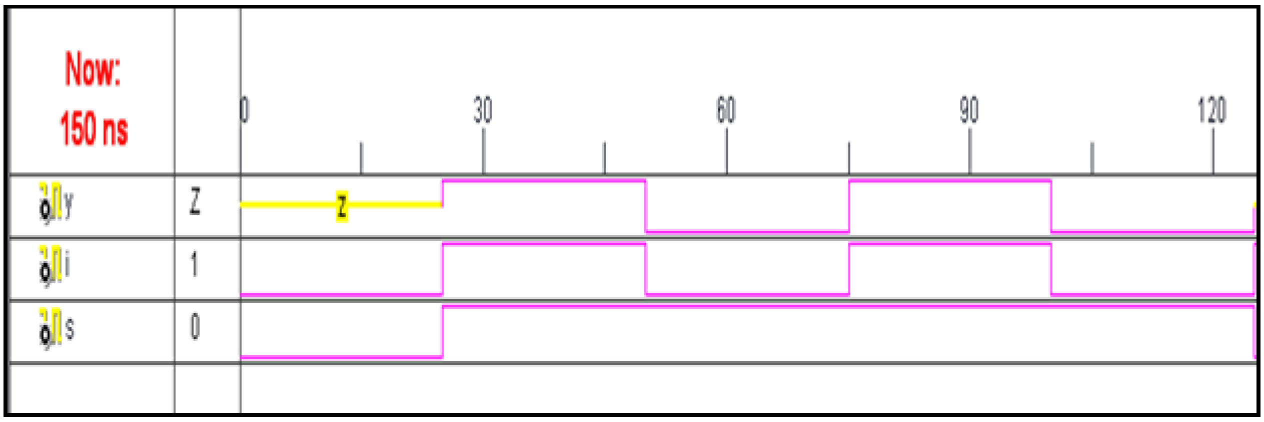


3.Transmission gate--verilog code and Test bench

```
module tr(i,s, y);
 input i,s;
 output y;
reg y;
always@(i,s)
begin
if(s==1)
y=i;
else
y=1'bz;
end
endmodule
```

//TESTBENCH

```
module tr_v;
reg i;
reg s;
wire y;
tr uut(.i(i),.s(s),.y(y));
initial
begin
i=0; s=0; #25;
i=1; s=1; #25;
i=0; s=1; #25;
i=1; s=1; #25;
i=0; s=1; #25;
i=1; s=0; #25;
end
endmodule
```



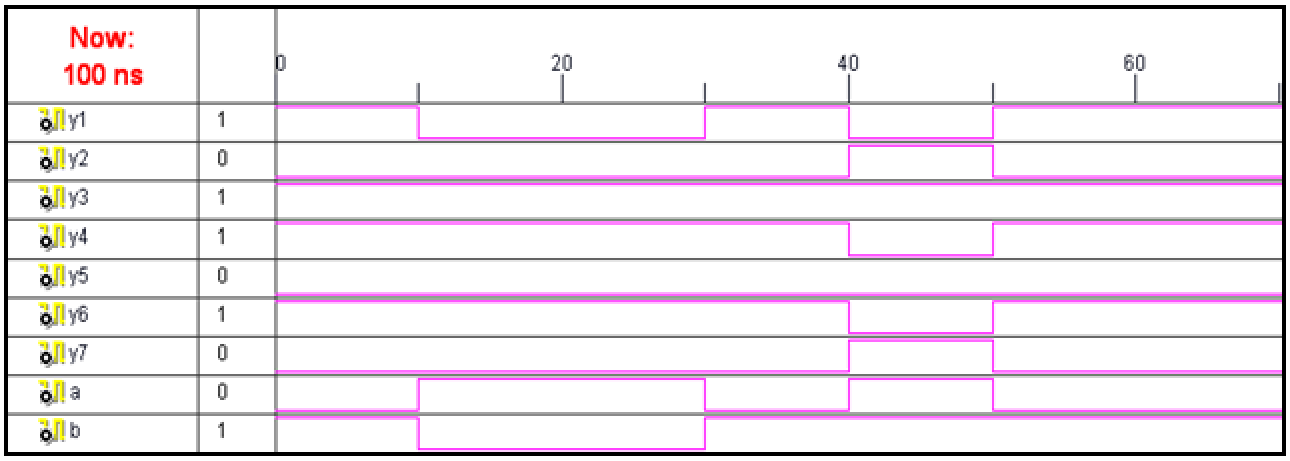
4. Logic gates-verilog code and Test bench

```
module gates(a,b, y1,y2,y3,y4,y5,y6,y7);
    input a,b;
    output y1,y2,y3,y4,y5,y6,y7;
              assign y1=~a;
              assign y2=a&b;
              assign y3=a|b;
              assign y4=~(a&b);
              assign y5=~(a|b);
              assign y6 =a^b;
              assign y7=~(a^b);
endmodule
```

//TESTBENCH

```
module gates_v;
reg a;
reg b;
wire y1;wire y2;wire y3;wire y4;wire y5;wire y6;wire y7;
gates uut(.a(a),.b(b),.y1(y1),.y2(y2),.y3(y3),.y4(y4),.y5(y5),.y6(y6),.y7(y7))
initial
begin

a=0; b=0;# 10;            a=0; b=0;# 10;
a=0; b=0;# 10;            a=0; b=0;# 10;
a=0; b=0;# 10;            a=0; b=0;# 10;
end
endmodule
```

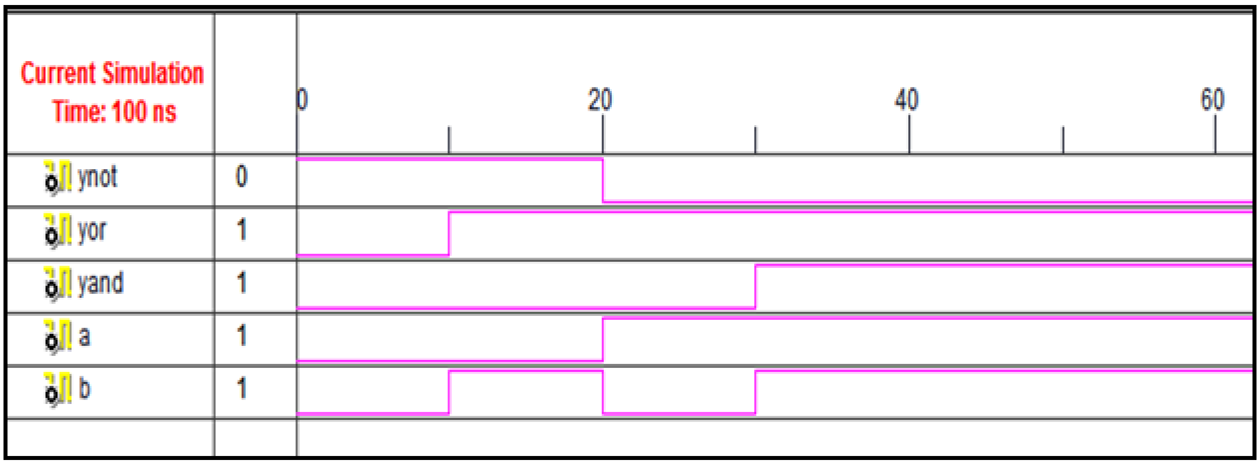


5. Realization of basic gates using Nand gates-verilog code and Testbench

```
module bgnd(a,b, yand,yor,ynot, anot,bnot,x);
    input a,b;
    output yand,yor,ynot;
    inout anot,bnot,x;
assign ynot=~(a&a);
assign anot=~(a&a);
assign bnot=~(b&b);
assign x=~(a&b);
assign yand =~(x&x);
assign yor =~(anot & bnot);
endmodule
```

//Testbench

```
module bgnd_tb();
reg a,b;
wire yand,yor,ynot;
bgnd uut(.a(a),.b(b),.yand(yand),.yor(yor),.ynot(ynot));
initial
begin
a=0; b=0;#10;
a=0; b=1;#10;
a=1; b=0;#10;
a=1; b=1;#10;
end
endmodule
```

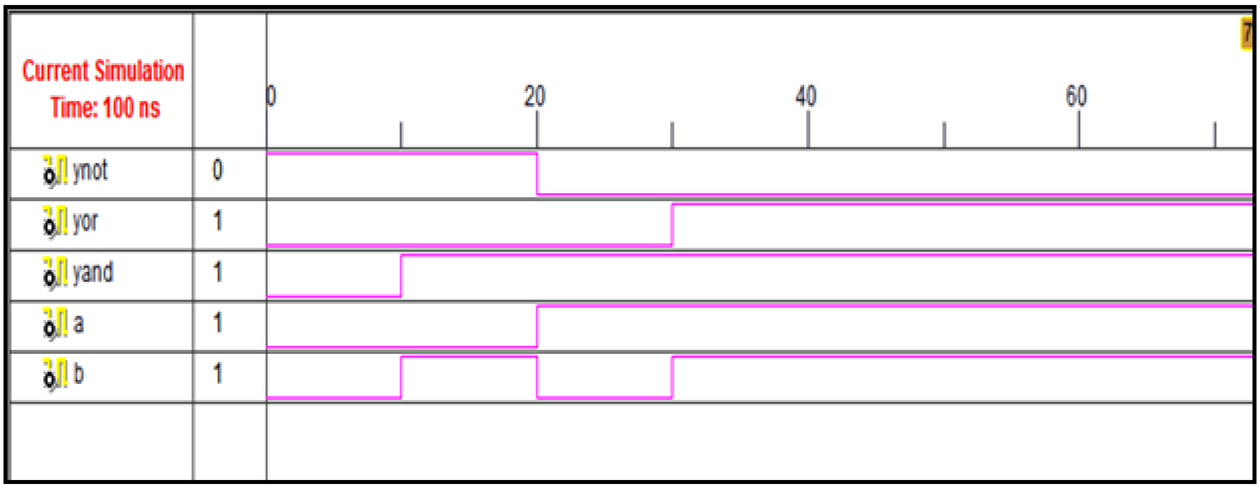


6. Realization of basic gates using Nor gates-Verilog code and Testbench

```
module bgnr(a,b, yand,yor,ynot, anot,bnot,x);
    input a,b;
    output yand,yor,ynot;
    inout anot,bnot,x;
assign ynot=~(a|a);
assign anot=~(a|a);
assign bnot=~(b|b);
assign x=~(a|b);
assign yand =~(x|x);
assign yor =~(anot | bnot);
endmodule
```

Testbench

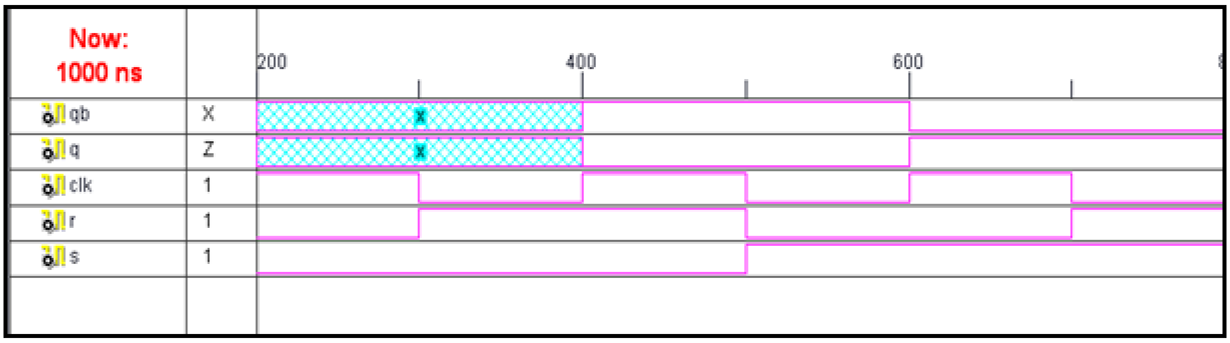
```
module bgnr_tb();
reg a,b;
wire yand,yor,ynot;
bgnd uut(.a(a),.b(b),.yand(yand),.yor(yor),.ynot(ynot));
initial
begin
a=0; b=0;#10;
a=0; b=1;#10;
a=1; b=0;#10;
a=1; b=1;#10;
end
endmodule
```



7. SR FLIP FLOP- Verilog code and Testbench

```
module srf(s,r,clk, q,qb);
   input s,r,clk;
   output q,qb;
            reg q,qb;
            always @(posedge clk)
            begin
            if(s==0 & r==1)
            q=0;
            else if(s==1 &r==0)
            q=1;
            else if(s==0 & r==0)
            q=q;
            else if (s==1 &r==1)
            q=1'bz;
            qb=~q;
            end
            endmodule

            //TESTBENCH
            module srf_v;
            reg s;reg r;reg clk;
            wire q,qb;
            srf  uut (.s(s),.r(r),.clk(clk),.q(q),.qb(qb));
            initial
            begin
            #100  s=0; r=0; clk=0;
            #100  s=0; r=0; clk=1;
            #100  s=0; r=1; clk=0;
            #100  s=0; r=1; clk=1;
            #100  s=1; r=0; clk=0;
            #100  s=1; r=0; clk=1;
            #100  s=1; r=1; clk=0;
            #100  s=1; r=1; clk=1;
end
endmodule
```

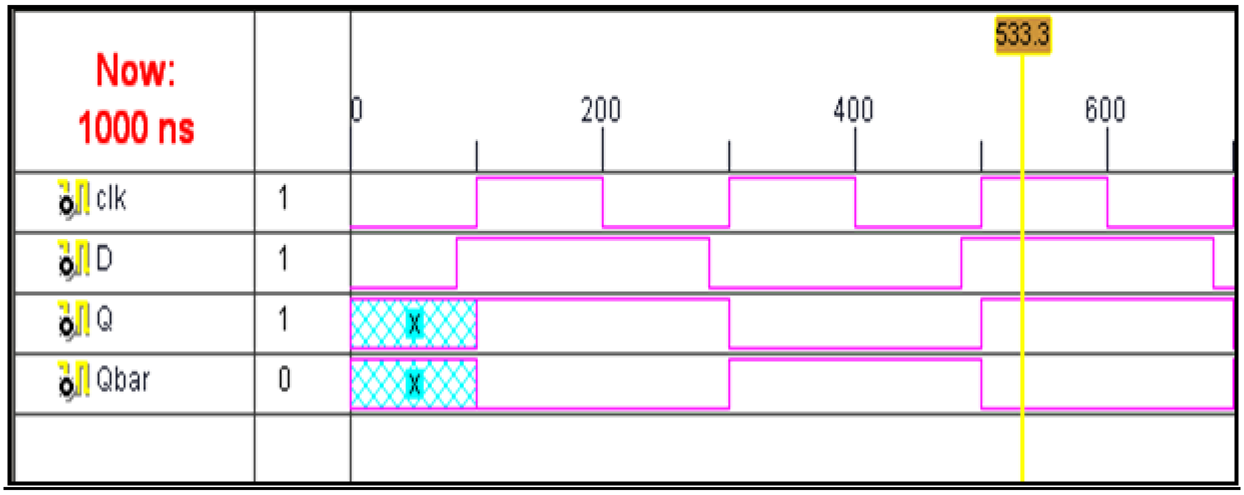


8. D flipflop- - Verilog code and Testbench

```
module dff(clk,d,q,qbar);
input clk;
input d;
output q,qbar;
reg temp,q,qbar;
always @(posedge clk)
begin
if (d==0)
temp = 1'b0;
else
temp = 1'b1;
end
assign q =  temp;
assign qbar = ~q;
endmodule
```

//TEST BENCH

```
module dff_tb ();
reg clk;
reg d;
wire q,qbar;
dff uut (.clk(clk),.d(d),.q(q),.qbar(qbar));
initial
begin
clk=0;d=0; #100;
clk=1;d=0; #100;
clk=0;d=0; #100;
clk=1;d=0; #100;
clk=0;d=0; #100;
end
endmodule
```

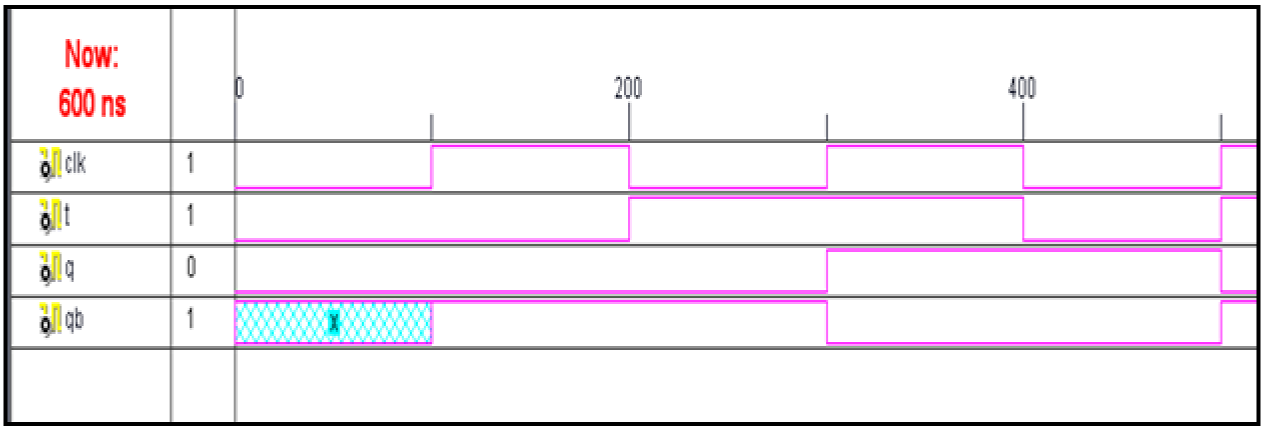


9. T flipflop- Verilog and code and Testbench

```
module tff(t,clk, q,qb);
   input t,clk;
  output q,qb;
 reg qb;
reg q;
initial q=0;
always@(posedge clk)
begin
if(t==0)
q=q;
else
q=~q;
qb=~q;
end
endmodule
```

//TESTBENCH

```
module tff_v;
reg t;
reg clk;
wire q;
wire qb;
tff uut(.t(t),.clk(clk),.q(q),.qb(qb));
initial begin
t=0;  clk=0; #100;
t=0;  clk=1; #100;
t=1;  clk=0; #100;
t=1;  clk=1; #100;
t=0;  clk=0; #100;
t=1;  clk=1; #100;
end
endmodule
```

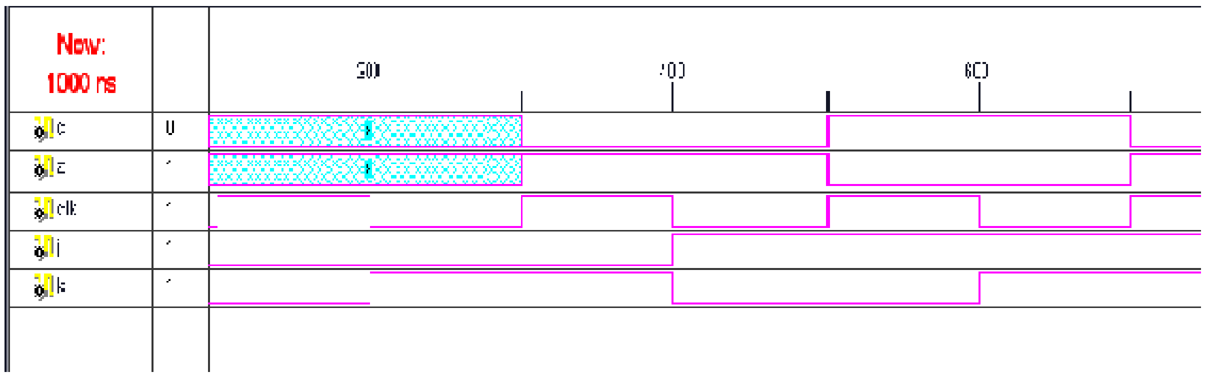


10. JK FLIPFLOP- verilog code and Testbench

```
module jkflip(j, k, clk, q, z);
   input j;
   input k;
   input clk;
   output q;
   output z;
          reg q,z;
          always@(posedge clk)
          begin
          if(j==0 & k==1)
          q=0;
          else if(j==1 & k==0)
          q=1;
          else if(j==0 & k==0)
          q=q;
          else if(j==1 & k==1)
          q=~q;
          z=~q;
          end
endmodule
```

//TESTBENCH

```
module jkflip_v;
reg j; reg k ;
reg clk ;
wire q,z;
jkflip uut(.j(j), .k(k), .clk(clk),.q(q),.z(z));
initial
begin
j=0; k=0; clk=0;
#100; j=0; k=0; clk=1;
#100; j=0; k=1; clk=0;
#100; j=0; k=1; clk=1;
#100; j=1; k=0; clk=0;
#100; j=1; k=0; clk=1;
#100; j=1; k=1; clk=0;
#100; j=1; k=1; clk=1;
end
endmodule
```

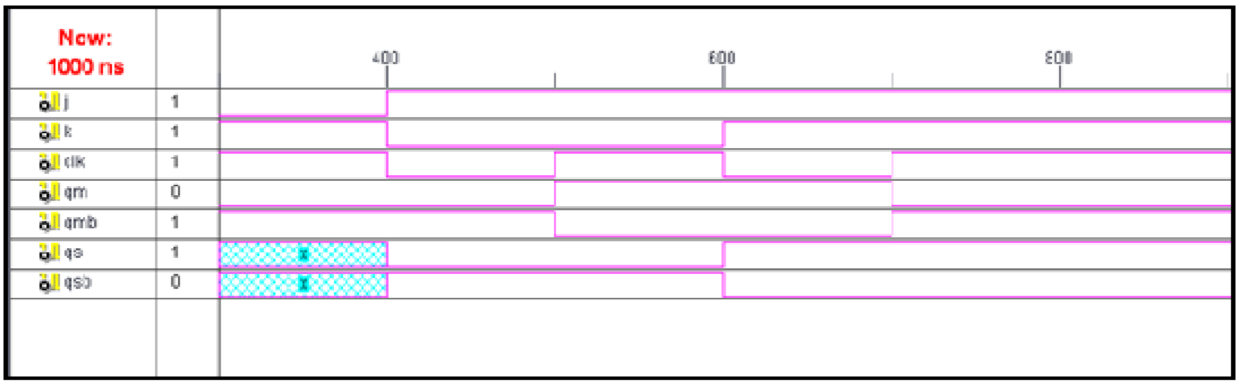


11. Master slave JK-verilog code and Testbench

```
module msjk(j,k,clk, qm,qmb,qs,qsb);
   input j,k,clk;
   output qm,qmb,qs,qsb;

reg qm,qmb ,qs,qsb;
always@(posedge clk)
begin
if(j==0 &k==1)
qm=0;
else if(j==1 &k==0)
qm=1;
else if (j==0 &k==0)
qm=qm;
else if (j==1 &k==1)
qm=~qm;
qmb=~qm;
end
always @(negedge clk)
begin
qs=qm;
qsb=~qs;
end
endmodule
```

```
//TESTBENCH
module msjkf();
reg j,k,clk;
wire qm,qmb,qs,qsb;
msjk uut(.j(j),.k(k),.clk(clk),.qs(qs),.qsb(qsb),.qm(qm),.qmb(qmb));
initial
begin
j=0; k=0; clk=0;
#100; j=0; k=0; clk=1;
#100; j=0; k=1; clk=0;
#100; j=0; k=1; clk=1;
#100; j=1; k=0; clk=0;
#100; j=1; k=0; clk=1;
#100; j=1; k=1; clk=0;
#100; j=1; k=1; clk=1;
end
endmodule
```

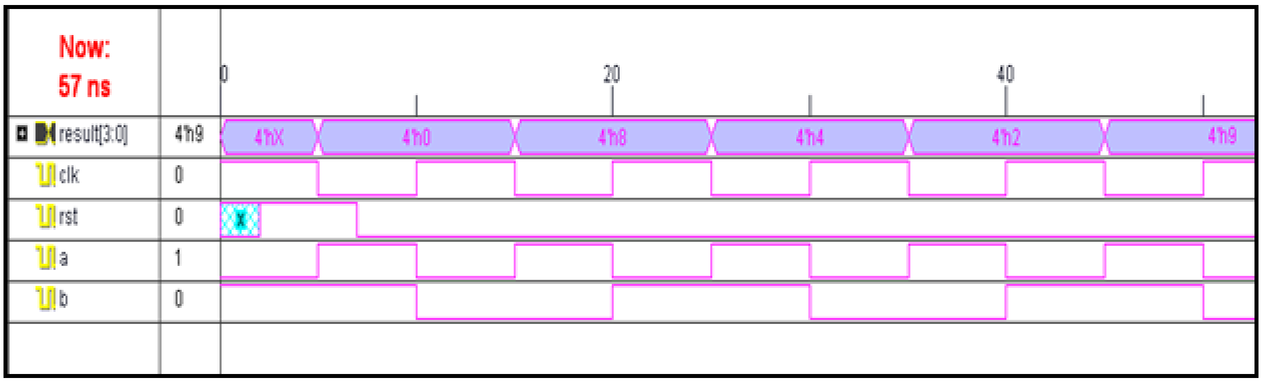


12. Serial adder –verilog code and Testbench

```
module sradd(a,b,rst,clk, result);
    input a,b,rst,clk;
    output [3:0] result;
             reg[3:0] result;
reg sum,carry;
integer count;
initial count=4;
always @(negedge clk)
begin
if(rst)
begin
count=0;carry=0;result=0;
end
else
begin
if(count<4)
begin
count=count+1;
sum=a^b^carry;
carry=((a&b)|(a&carry)|(b&carry));
result={sum,result[3:1]};
end
end
end
endmodule
```

//TESTBENCH

```
module sradd_tb();
reg a,b,rst,clk;
wire[3:0] result;
sradd u1(.a(a),.b(b),.rst(rst),.clk(clk),.result(result));
initial
begin
$monitor (a,b,rst,clk,result);
clk =1'b1;
a=1'b0;
b=1'b1;
#2 rst=1'b1;
#5 rst=1'b0;
#50 $finish;
end
always #5 clk=~clk;
always #5 a=~a;
always #10 b=~b;
endmodule
```

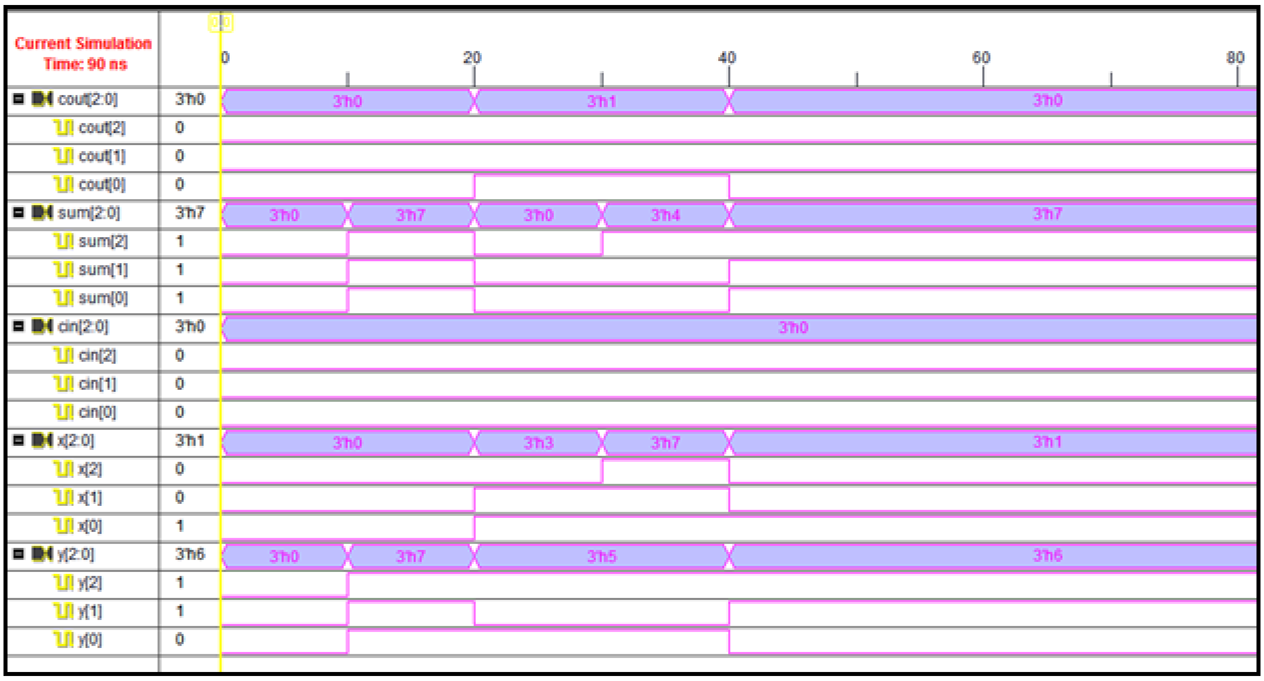

Now:
57 ns
0
20
40
result[3:0]
4'h9
4'hX
4'h0
4'h8
4'h4
4'h2
4'h9
clk
0
rst
0
X
a
1
b
0

13. Parallel Adder- Verilog code and Test benc

```
module pd(x,y, cin, sum,cout);
   input [2:0] x,y;
   input cin;
   output [2:0] sum,cout;
wire c0,c1;
          assign sum [0]=x[0]^y[0]^cin;
                   assign sum [1]=x[1]^y[1]^c0;
                            assign sum[2]=x[2]^y[2]^c1;
          assign c0=(x[0]&y[0])|(x[0]&cin)|(y[0]&cin);
          assign c1= (x[1]&y[1])|(x[0]&c0)|(y[0]&c0);
          assign cout=(x[2]&y[2])|(x[2]&c1)|(y[2]&c1);
endmodule
```

//Test Bench

```
module pd_tb();
reg [2:0]x,y,cin;
wire [2:0]sum,cout;
initial
begin
$monitor (x,y,cin,sum,cout);
x=3'b000;
y=3'b0000;
cin=0;
#10 x=3'b000;
y=3'b111;
#10 x=3'b011;
y=3'b101;
#10 x=3'b111;
y=3'b101;
#10 x=3'b001;
y=3'b110;
#50 $finish;
end
pd uut(.x(x),.y(y),.cin(cin),.sum(sum),.cout(cout));
endmodule
```


Current Simulation Time: 90 ns
0
20
40
60
80
cout[2:0] 3'h0
cout[2] 0
cout[1] 0
cout[0] 0
sum[2:0] 3'h7
sum[2] 1
sum[1] 1
sum[0] 1
cin[2:0] 3'h0
cin[2] 0
cin[1] 0
cin[0] 0
x[2:0] 3'h1
x[2] 0
x[1] 0
x[0] 1
y[2:0] 3'h6
y[2] 1
y[1] 1
y[0] 0
3'h0
3'h1
3'h0
3'h0
3'h7
3'h0
3'h4
3'h7
3'h0
3'h0
3'h3
3'h7
3'h1
3'h0
3'h7
3'h5
3'h6

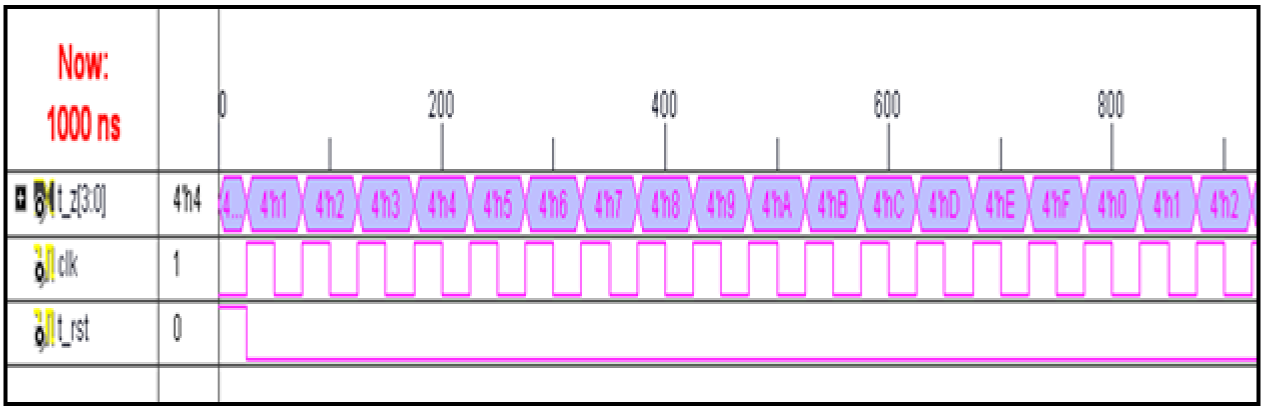
14.Asynchronous upcounter-verilog code and testbench

```
module ask(clk,rst, z);
   input clk,rst;
   output [3:0] z;
            reg z0,z1,z2,z3;
always @(posedge clk or posedge rst)
begin
if(rst)
z0=1'b0;
else
z0=~z0;
end
always @(negedge  z0 or posedge rst)
begin
if(rst)
z1=1'b0;
else
z1=~z1;
end
always @(negedge z1 or posedge rst)
begin
if(rst)
z2=1'b0;
else
z2=~z2;
end
always @(negedge z2 or posedge rst)
begin
if(rst)
z3=1'b0;
else
z3=~z3;
end
assign z={z3,z2,z1,z0};
endmodule
```

//TESTBENCH

```
module ask_tb;
reg  clk,t_rst;
wire[3:0]t_z;
initial
begin
$monitor (clk,t_z,t_rst);
t_rst=1;
clk=0;
#25 t_rst=0;
end
always
begin
#25 clk=~clk;
end
ask(.clk(clk),.z(t_z),.rst(t_rst));
```

```
endmodule
```


Now:
1000 ns
0
200
400
600
800
t_z[3:0]
4'h4
clk
1
t_rst
0
4'h1
4'h2
4'h3
4'h4
4'h5
4'h6
4'h7
4'h8
4'h9
4'hA
4'hB
4'hC
4'hD
4'hE
4'hF
4'h0
4'h1
4'h2

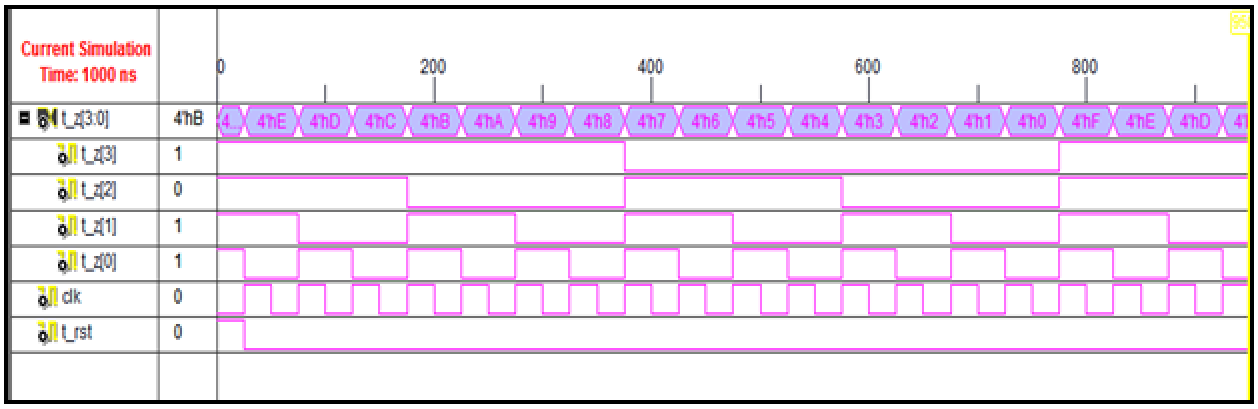
15. Asynchronous down-verilog and testbench

```
module asd(clk,rst, z);
   input clk,rst;
   output [3:0] z;
          reg z0,z1,z2,z3;
always@(posedge clk or posedge rst)
          begin if (rst)
z0=1'b1;
else
z0=~z0;
end
always@(posedge z0 or posedge rst)
          begin if (rst)
z1=1'b1;
else
z1=~z1;
end
always@(posedge z1 or posedge rst)
          begin if (rst)
z2=1'b1;
else
z2=~z2;
end
always@(posedge z2 or posedge rst)
          begin if (rst)
z3=1'b1;
else
z3=~z3;
end
assign z={z3,z2,z1,z0};

endmodule
```

//Testbench

```
module asd_tb();
reg clk,t_rst;
wire[3:0]t_z;
initial
begin
$monitor(clk,t_rst,t_z);
t_rst=1;
clk=0;
#25t_rst=0;
end
always
begin
#25 clk=~clk;
end
asd uut(.clk(clk),.z(t_z),.rst(t_rst));
endmodule
```

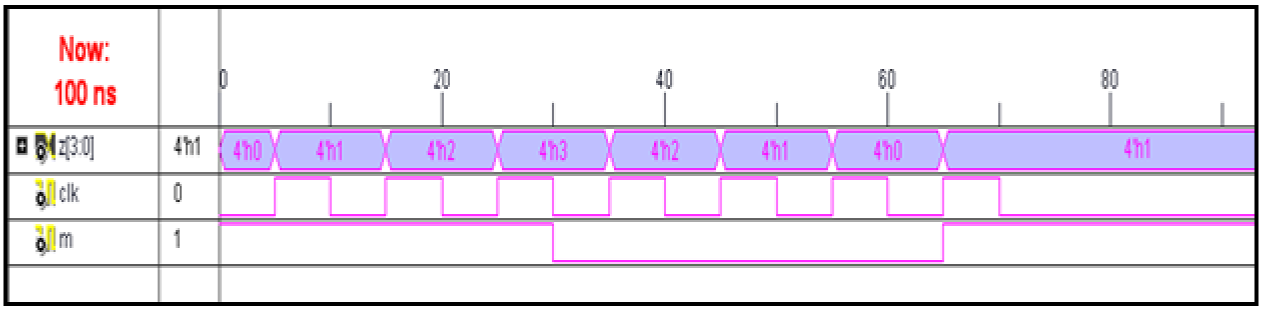

Current Simulation
Time: 1000 ns
0
200
400
600
800
t_z[3:0] 4'hB
t_z[3] 1
t_z[2] 0
t_z[1] 1
t_z[0] 1
clk 0
t_rst 0
4...
4'hE
4'hD
4'hC
4'hB
4'hA
4'h9
4'h8
4'h7
4'h6
4'h5
4'h4
4'h3
4'h2
4'h1
4'h0
4'hF
4'hE
4'hD

16.Synchronous up-down counter- verilog code and Testbench

```
module syn(m,clk, z);
    input m,clk;
    output [3:0] z;
             reg[3:0] z;
             initial
begin
z=4'b0000;
end
always@(posedge clk)
begin
if(m==1)
z=z+1;
else
z=z-1;
end
endmodule
```

//TESTBENCH

```
module syn_v;
reg clk;
reg m;
wire[3:0]z;
syn uut(.z(z),.clk(clk),.m(m));
initial
begin
clk=0; m=1; #5;
clk=1; m=1; #5;
clk=0; m=1;  #5;
clk=1; m=1; #5;
clk=0; m=1; #5;
clk=1; m=1; #5;
clk=0; m=0; #5;
clk=1; m=0; #5;
clk=0; m=0; #5;
clk=1; m=0; #5;
clk=0; m=0; #5;
clk=1; m=0; #5;
clk=0; m=0; #5;
clk=1; m=1; #5;
clk=0; m=1; #5;
end
endmodule
```


Now:
100 ns
0
20
40
60
80
z[3:0]
4'h1
clk
0
m
1
4'h0
4'h1
4'h2
4'h3
4'h2
4'h1
4'h0
4'h1

Probable Viva questions

1. Give expansion for the following,
   VHDL, FPGA, CPLD, ASIC’s, ISP, IEEE, PLD, and JTAG.
2. Which are the two IEEE standard HDL languages?
3. Which IEEE standard used to write HDL?
4. What is synthesis? Which synthesis tool is used in the lab?
5. What is simulator? Which simulator tool is used in the lab?
6. Define signal, variable, constant.
7. What are the different types of assignment statements?
8. What do you mean by keywords? List some of the key words of VHDL and Verilog.
9. Define Comparator.
10. What is the application of comparator?
11. Why we need to use comparator?
12. What is the difference between analog comparator to digital comparator?
13. In which field comparator plays an important role?
14. What is key difference between sequential and combinational circuits?
15. What are applications of multiplexers?
16. Write short note on transport delay.
17. Write about behavioral models.
18. Write short note on delta delay.
19. What are hardware description languages?
20. Write short note on inertial delay.
21. Mention the applications of comparators, encoders, Mux, Demux, decoders.
22. How do we choose the appropriate type of description for the given program?
23. Why we need covertion techniques?
24. Application of gray to binary and binary to gray conversion.
25. Define MUX with example.
26. Define DeMUX with example.
27. Why we need to convert from binary to gray code and vice varsa.
28. What is the difference between the adders constructing using IC and basic gates?
29. What is the advantage of using IC’s to construct the adders and subtractor.
30. Define parallel adder and parallel subtractor

31. What is the difference between parallel adder and parallel constructor?
32. What is the need of code conversion?

33. Define adders and subtractor.
34. What are different types of ROM,s.
35. What are applications of flip flops.
36. Explain resolution functions.
37. How does look-a head carry adder speed up the addition process.
38. What is parity bit generator.
39. What are steps involved in implementation and anlysis of digital systems.
40. What are capabilities of VHDL.
41. Explain the significance of conditional signal assignment statement and selected
42. signal assignment statement.
43. How will you compare component declaration and component instantiation.
44. Write down the configuration specification for full adder circuit.
45. Write down the VHDL code of following:
    (a)D-flip flop (b) T-flip flop
46. Write down the VHDL code of S-R flip flop.
47. What are generics?
48. Explain with example that how a component can be made more general using generics.
49. What are sequential statements? Write down its syntax.
50. Discuss process and wait statements.
51. How are sequential statements different from concurrent statements?
52. Write a short note on package and library.
53. Write short note on subprogram.
54. What are aliases. Explain with example.
55. Design and implement counter using VHDL which counts upto 9 and down counts
56. again from 9 to 0.
57. Differentiate between a process and wait statement. Can they be used simultaneously in a program?
58. What is the difference between variable and signal.
59. Write down the truth table and VHDL code for the 4-bit left to right shift register.
    Also draw the circuit and output waveforms.
60. Write down the truth table and VHDL code for the 4-bit up/down counter. Also

draw the circuit and output waveforms

61. What are the basic components of a micro computer? Explain briefly.
62. Describe microcomputer implementation in VHDL.

63. How can a ROM be used as a PLA') Write down its advantages.
64. Discuss briefly 22 V /0 PLD.
65. Write short note on PAL 16L8.
66. Write short note on Various Loops in VHDL
67. Write short note on Packages.
68. What does VHDL stands for?
69. Which IEEE standard describes the VHDL language?
70. List the three popular Hardware languages.
71. Which are the different levels of abstraction that can be specified using VHDL?
72. List the different design units of VHDL.
73. Which are the mandatory design units to write VHDL code?
74. Which are the different modes of port declaration?
75. Which are the valid characters for identifier declaration?
76. Which are the different classes of operators?
77. Where do you write the concurrent statements?
78. Where do you write the sequential statement?
79. In which model process statement appears?
80. What is the importance of sensitivity list in process statement?
81. Is VHDL Case sensitive?
82. Does VHDL support multi dimensional arrays?
83. Can combinational circuits be coded inside the process?
84. Does VHDL support operator overloading?
85. Is it possible to write multiple entities for a single architecture?
86. Is it possible to write multiple architectures for a single entity?
87. Where we declare the variable?
88. Write device configuration for CPLD and FPGA Used in your Lab.
89. Expand CPLD and FPGA.
90. Differentiate sequential and concurrent statement.
91. List the different types of wait statements.
92. How you model your program using wait statement?
93. What are the different modeling styles in VHDL?

94. What is the difference between the bit and std_logic?
95. What is the difference between the variable and signals?
96. Name the different VHDL objects.
97. Name the different data types used in VHDL.

98. Explain the VHDL term (i) Entity, (ii) Architecture, (iii) Configuration, (iv) Package, (v) Driver, (vi) Process, (vii) Attribute, (viii) Generic and (ix) Bus.
99. Write the general syntax for Case, LOOP, Architecture Configuration, package, Process, Exit.
100. Differentiate between Procedure and Function
101. Explain attribute, event, range
102. How to detect signal edge using attribute?
103. What is synthesis?
104. What is simulation?
105. Differentiate between syntax error and semantic error.
106. Is other clause necessary in VHDL case statement? why?
107. What are the inputs required for synthesis?
108. Which architecture description you preferred? Why?
109. Which Tool you used for simulation?
110. Which Tool you used for synthesis?
111. What is the difference between CPLD and FPGA?
112. What is the difference between synchronous and asynchronous reset?
113. What is the basic element of memory?
114. What do you mean by latch?
115. How you model latch in VHDL?
116. What is the difference between synchronous and asynchronous counter?
117. What is the difference between backend and front-end?
118. Expand ASIC.
119. Expand JTAG.
120. Expand ISE
121. Which IEEE standard supports JTAG?
122. What information is present in .Bit File?
123. What do you mean by configuration?
124. Which file used to configure the CPLD?
125. Which file used to configure the FPGA?
126. Explain the fabrication steps of NMOS transistor?

127. Explain the fabrication steps of pmos transistor?
128. What is BICMOS?
129. Differentiate between CMOS and bipolar transistors.
130. What is dynamic CMOS Transistor?

131. Write a verilog code for SR FLIPFLOP in structural description?
132. Write a verilog code for T FLIPFLOP in structural description?
133. Design a 4:1 multiplexer?
134. Design a parity generator?
135. write a verilog code for parity generator?

Procedure for layout design by using Microwind

1. Open Microwind.
2. Open file tab-Open select foundry-select CMOS 0.12µm technology.
3. Construct layout by picking the layers from the pallet.
4. Save and simulate the layout.
5. Check DC and Transient analysis.
6. Calculate Gain of layout and verify with the practical result.
7. Extract RC values.
8. Note all the results- save and quit.

Procedure for schematic design by using DSCH

1. Open DSCH 3.1.
2. File tab -generate spice file.
3. To set path for spice file, select option for Browse- select the drive.
4. Construct the schematic circuit by dragging the components from the pallet.
5. Save and run.
6. Go to program files.
7. Select Microwind 3.1 folder.

7. Select client folder-select DSCH3.1 folder-select system folder.

8. Select spice library file-close the window-Enable the ADD node list.

9. Check on update spice file.

10. Change to n.

11. Save the spice file and click ok.

Procedure for schematic design using DSCH

1. File – convert into- spice ntelist- select required format
2. Go to models-parameters and enable-empherical level 3-ok
3. Now click on extract to extract RC values and close.

1. Design an **<u>Inverter</u>** with given specifications*, completing the design flow mentioned below:

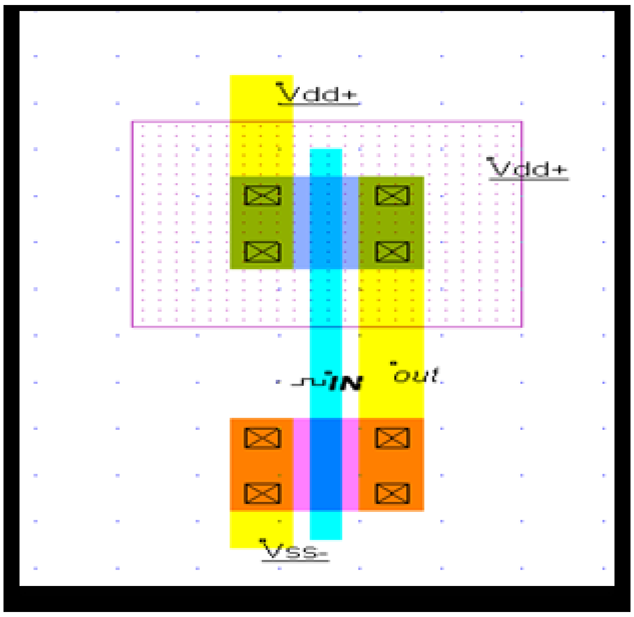
a. Draw the schematic and verify the following

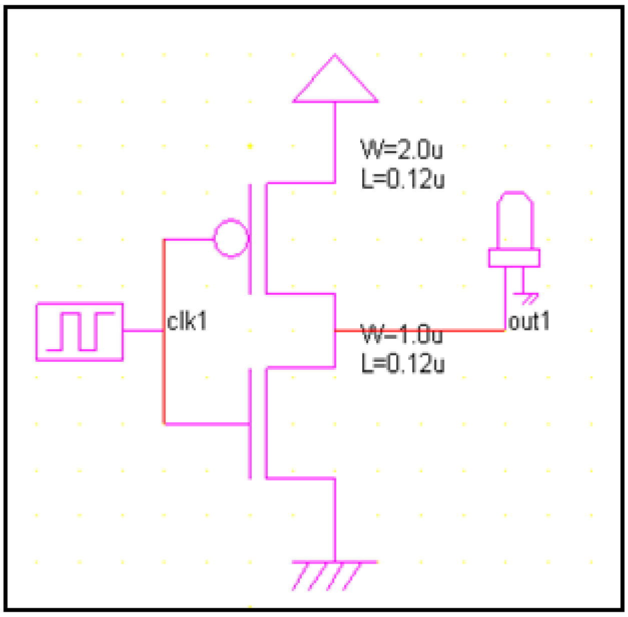
i) DC Analysis

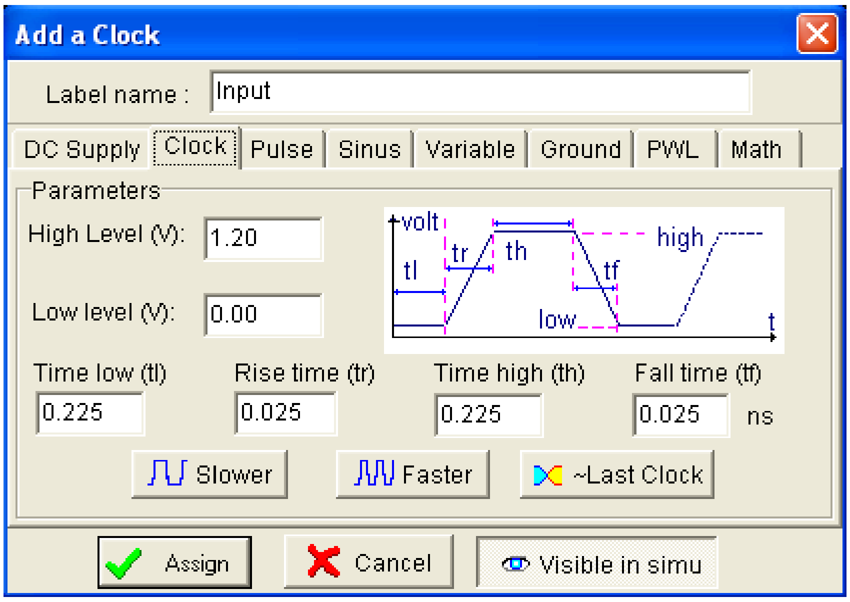
ii) Transient Analysis

b. Draw the Layout and verify the DRC

(i) CMOS Inverter







Add a Clock

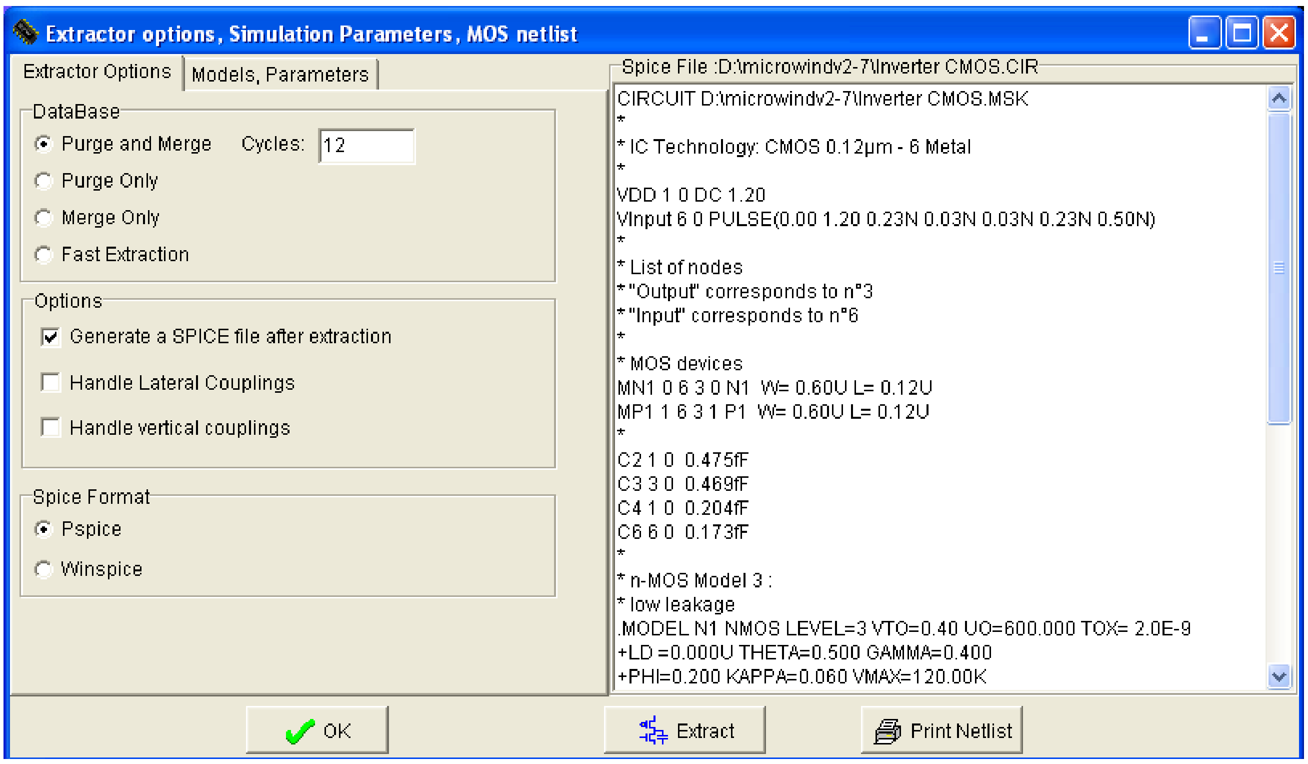
Label name : Input

Parameters

High Level (V): 1.20

Low level (V): 0.00

Time low (tl)	Rise time (tr)	Time high (th)	Fall time (tf)	
0.225	0.025	0.225	0.025	ns



Extractor options , Simulation Parameters , MOS netlist

Extractor Options | Models, Parameters

DataBase
(● Purge and Merge) Cycles: 12
○ Purge Only
○ Merge Only
○ Fast Extraction

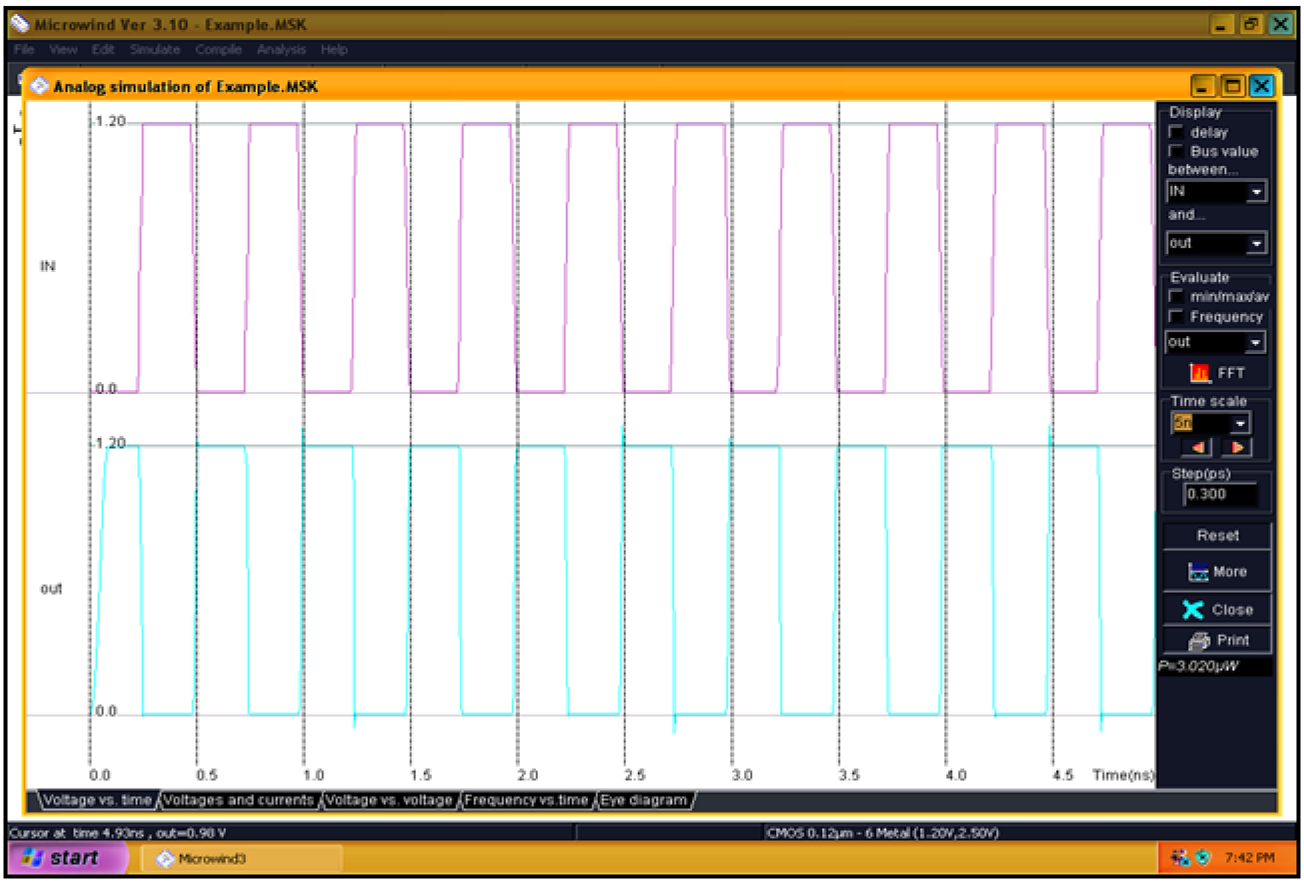
Options
[✓] Generate a SPICE file after extraction
[ ] Handle Lateral Couplings
[ ] Handle vertical couplings

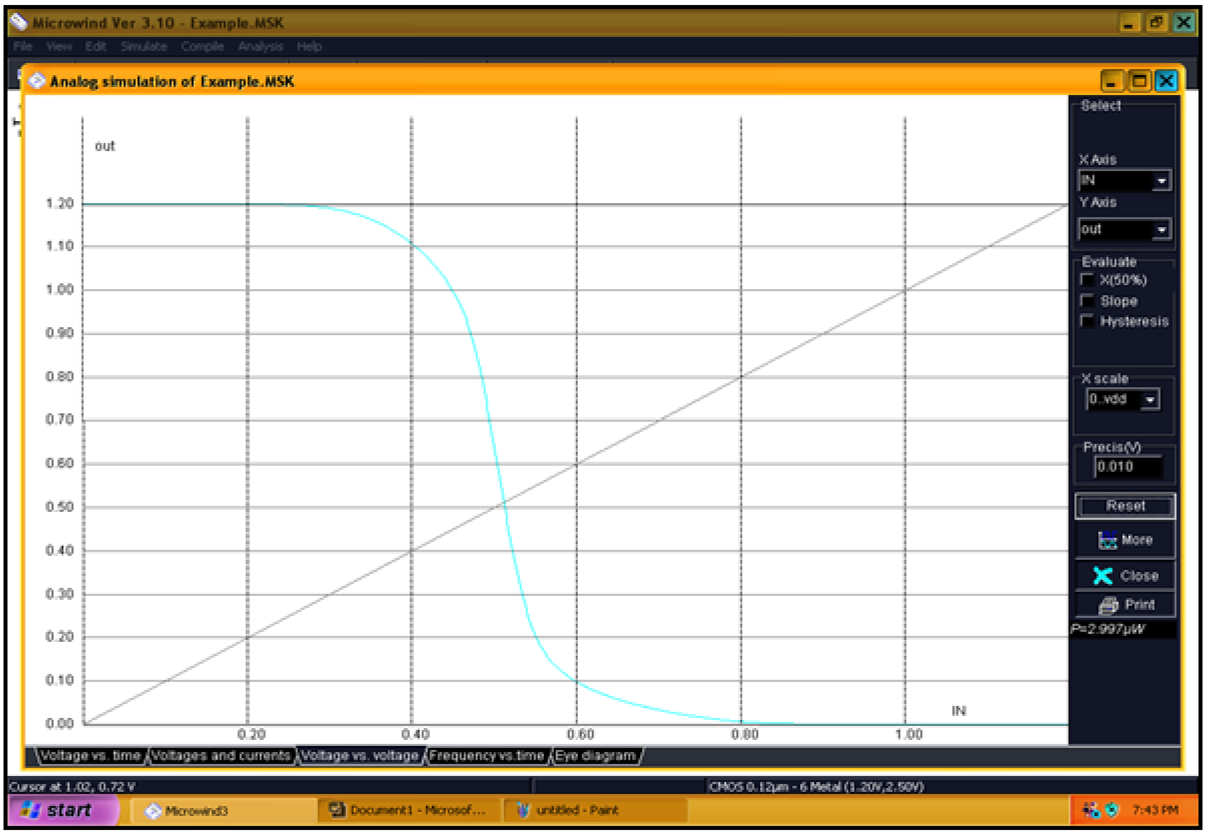
Spice Format
(● Pspice)
○ Winspice

Spice File :D:\microwindv2-7\Inverter CMOS.CIR

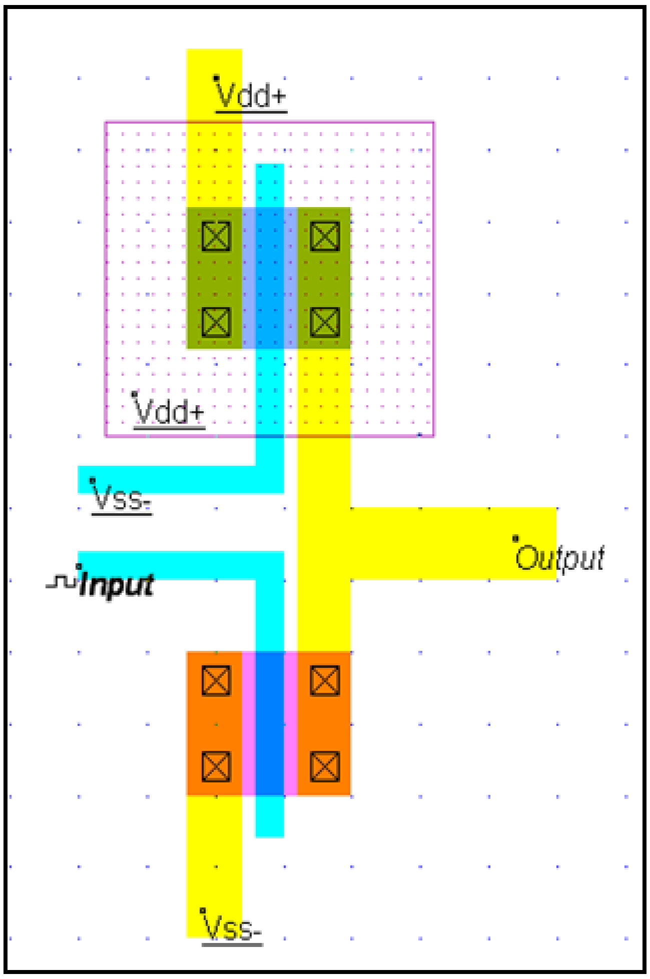
```
CIRCUIT D:\microwindv2-7\Inverter CMOS.MSK
*
* IC Technology: CMOS 0.12µm - 6 Metal
*
VDD 1 0 DC 1.20
VInput 6 0 PULSE(0.00 1.20 0.23N 0.03N 0.03N 0.23N 0.50N)
*
* List of nodes
* "Output" corresponds to n°3
* "Input" corresponds to n°6
*
* MOS devices
MN1 0 6 3 0 N1  W= 0.60U L= 0.12U
MP1 1 6 3 1 P1  W= 0.60U L= 0.12U
*
C2 1 0  0.475fF
C3 3 0  0.469fF
C4 1 0  0.204fF
C6 6 0  0.173fF
*
* n-MOS Model 3 :
* low leakage
.MODEL N1 NMOS LEVEL=3 VTO=0.40 UO=600.000 TOX= 2.0E-9
+LD =0.000U THETA=0.500 GAMMA=0.400
+PHI=0.200 KAPPA=0.060 VMAX=120.00K
```

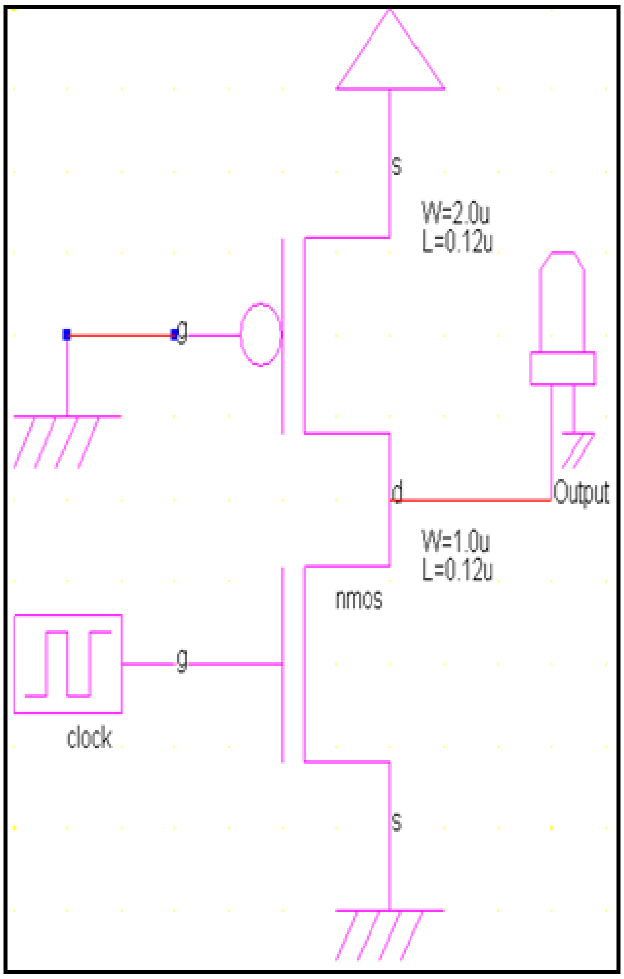
OK | Extract | Print Netlist

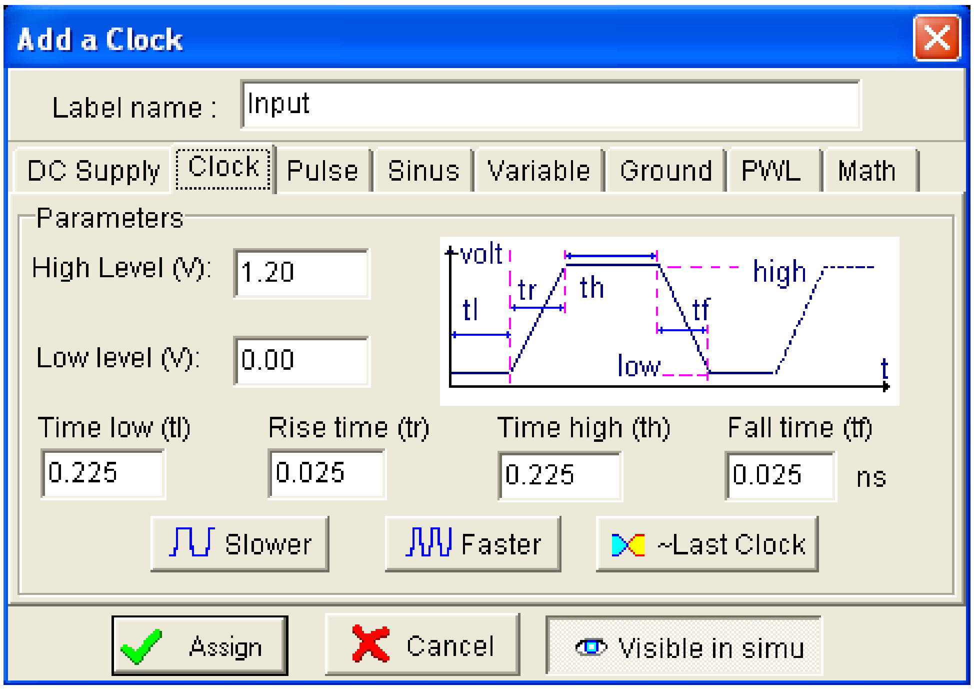



Microwind Ver 3.10 - Example.MSK
File View Edit Simulate Compile Analysis Help
Analog simulation of Example.MSK
out
IN
Select
X Axis
IN
Y Axis
out
Evaluate
X(50%)
Slope
Hysteresis
X scale
0..vdd
Precis(V)
0.010
Reset
More
Close
Print
P=2.997µW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram
Cursor at 1.02, 0.72 V
CMOS 0.12µm - 6 Metal (1.20V,2.50V)
start
Microwind3
Document1 - Microsof...
untitled - Paint
7:43 PM

(ii) Pseudo nmos Inverter







Label name : Input

Parameters

High Level (V): 1.20

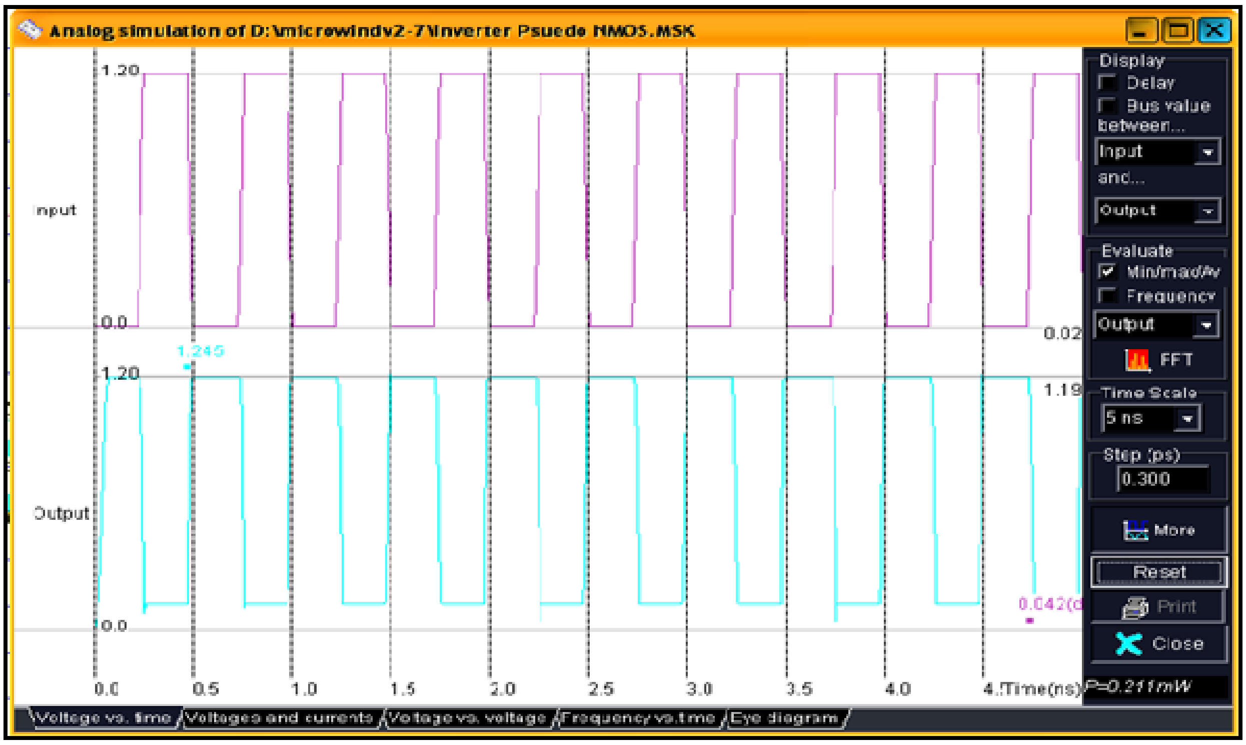
Low level (V): 0.00

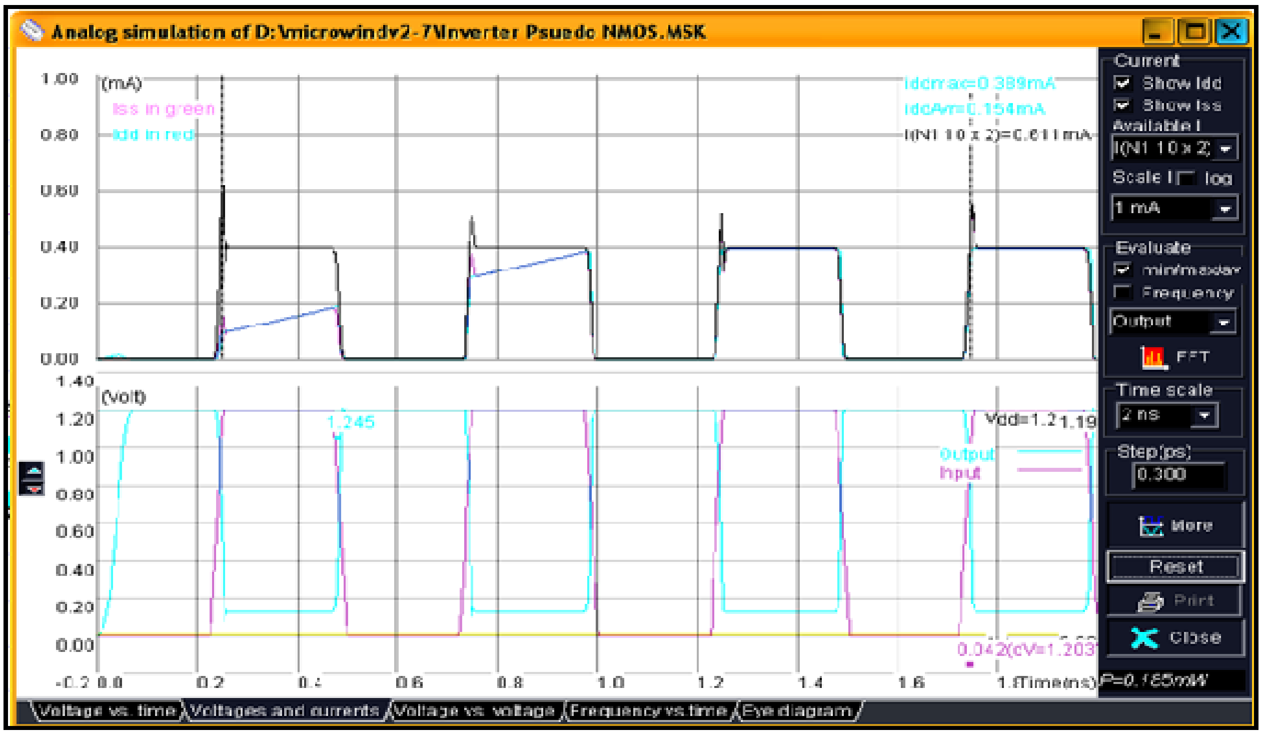
Time low (tl) 0.225

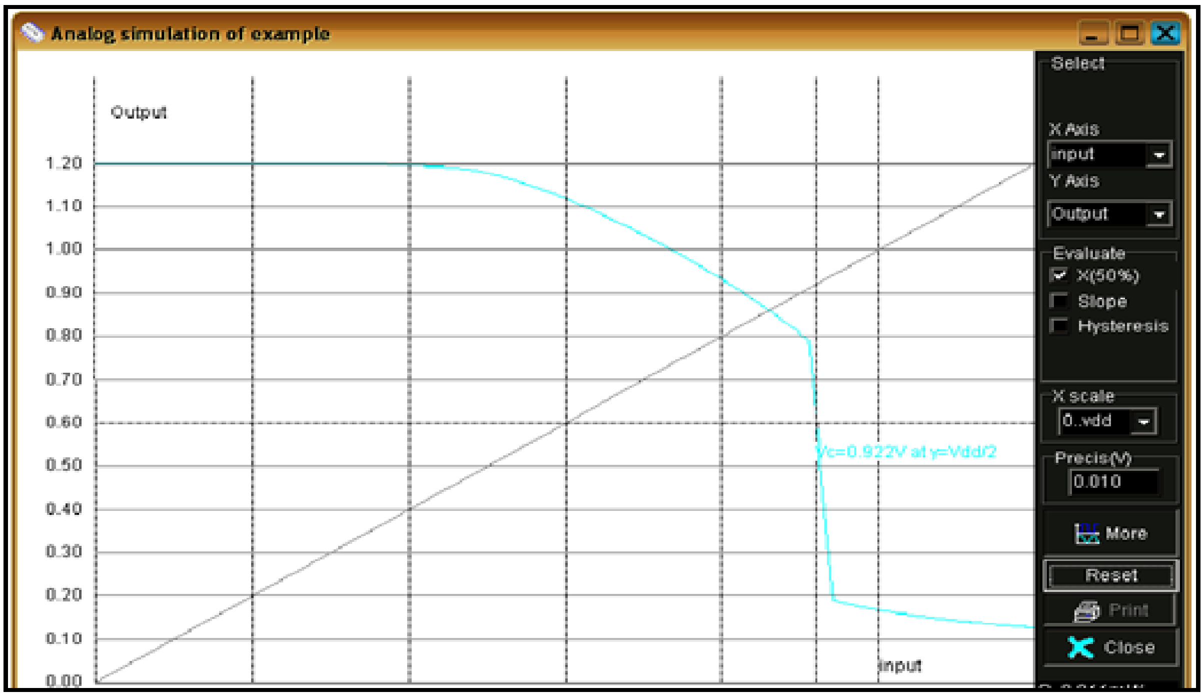
Rise time (tr) 0.025

Time high (th) 0.225

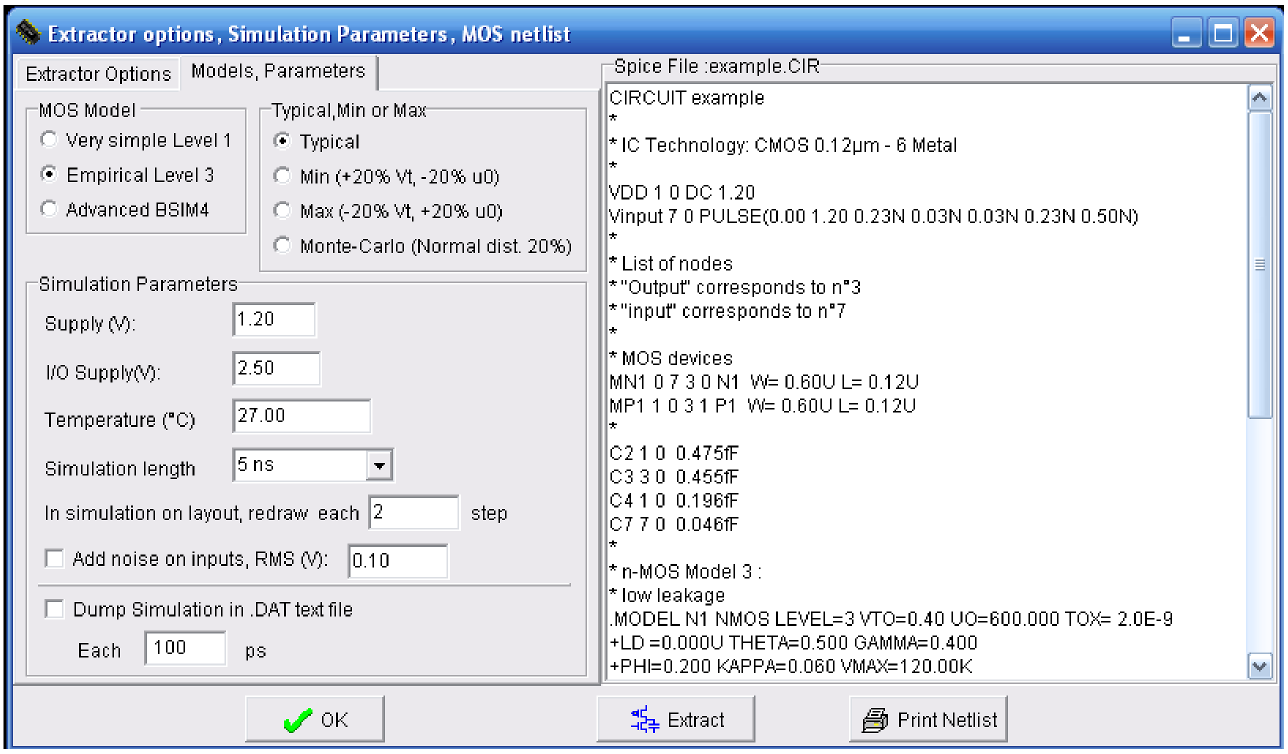
Fall time (tf) 0.025 ns


Analog simulation of D:\microwindv2-7\Inverter Psuedo NMOS.MSK
Input
Output
1.20
0.0
1.245
0.02
1.19
0.042
Time(ns)
P=0.211mW
Display
Delay
Bus value between...
Input
and...
Output
Evaluate
Min/max/Av
Frequency
FFT
Time Scale
5 ns
Step (ps)
0.300
More
Reset
Print
Close
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram


Analog simulation of D:\microwindv2-7\Inverter Psuedo NMOS.MSK
(mA)
Iss in green
Idd in red
Iddmax=0.389mA
IddAvr=0.154mA
I(N1 10 x 2)=0.611mA
(volt)
1.245
Vdd=1.2
1.19
Output
Input
0.042(dV=1.203
Time(ns)
P=0.165mW
Current
Show Idd
Show Iss
Available I
I(N1 10 x 2)
Scale I
log
1 mA
Evaluate
min/max/av
Frequency
Output
FFT
Time scale
2 ns
Step(ps)
0.300
More
Reset
Print
Close
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram



X Axis: input
Y Axis: Output
Evaluate: [✓ X(50%)] Slope Hysteresis
X scale: 0..vdd
Precis(V): 0.010



MOS Model: Very simple Level 1 [✓ Empirical Level 3] Advanced BSIM4

Typical,Min or Max: [✓ Typical] Min (+20% Vt, -20% u0) Max (-20% Vt, +20% u0) Monte-Carlo (Normal dist. 20%)

Simulation Parameters

Supply (V): 1.20

I/O Supply(V): 2.50

Temperature (°C): 27.00

Simulation length: 5 ns

In simulation on layout, redraw each 2 step

Add noise on inputs, RMS (V): 0.10

Dump Simulation in .DAT text file

Each 100 ps

Spice File :example.CIR

```
CIRCUIT example
*
* IC Technology: CMOS 0.12µm - 6 Metal
*
VDD 1 0 DC 1.20
Vinput 7 0 PULSE(0.00 1.20 0.23N 0.03N 0.03N 0.23N 0.50N)
*
* List of nodes
* "Output" corresponds to n°3
* "input" corresponds to n°7
*
* MOS devices
MN1 0 7 3 0 N1  W= 0.60U L= 0.12U
MP1 1 0 3 1 P1  W= 0.60U L= 0.12U
*
C2 1 0  0.475fF
C3 3 0  0.455fF
C4 1 0  0.196fF
C7 7 0  0.046fF
*
* n-MOS Model 3 :
* low leakage
.MODEL N1 NMOS LEVEL=3 VTO=0.40 UO=600.000 TOX= 2.0E-9
+LD =0.000U THETA=0.500 GAMMA=0.400
+PHI=0.200 KAPPA=0.060 VMAX=120.00K
```

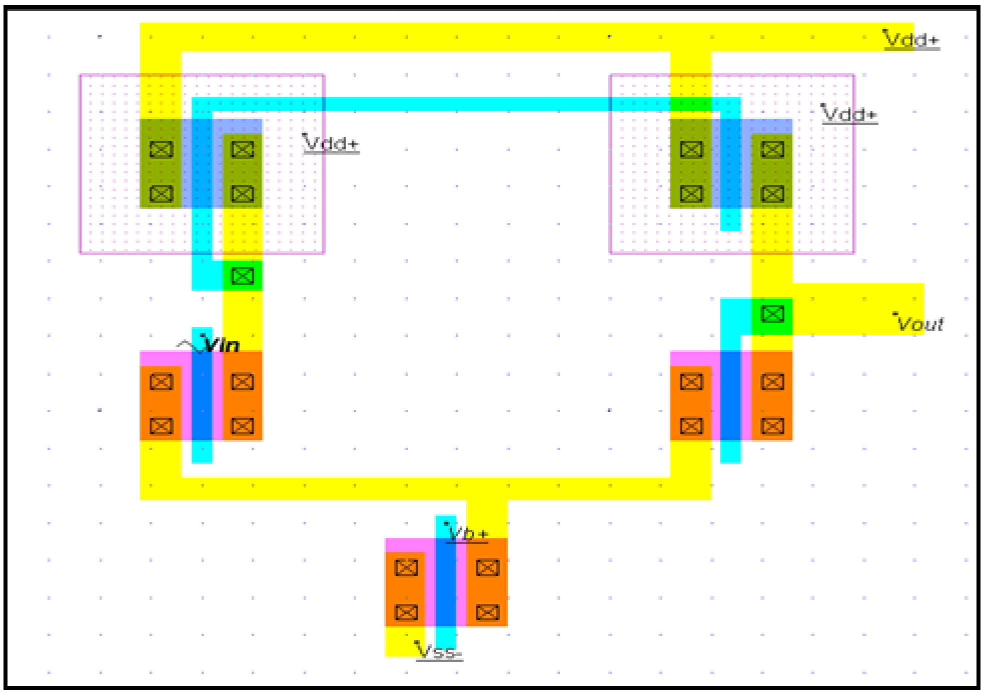
2. Design the following circuits with given specifications*, completing the design flow mentioned below:

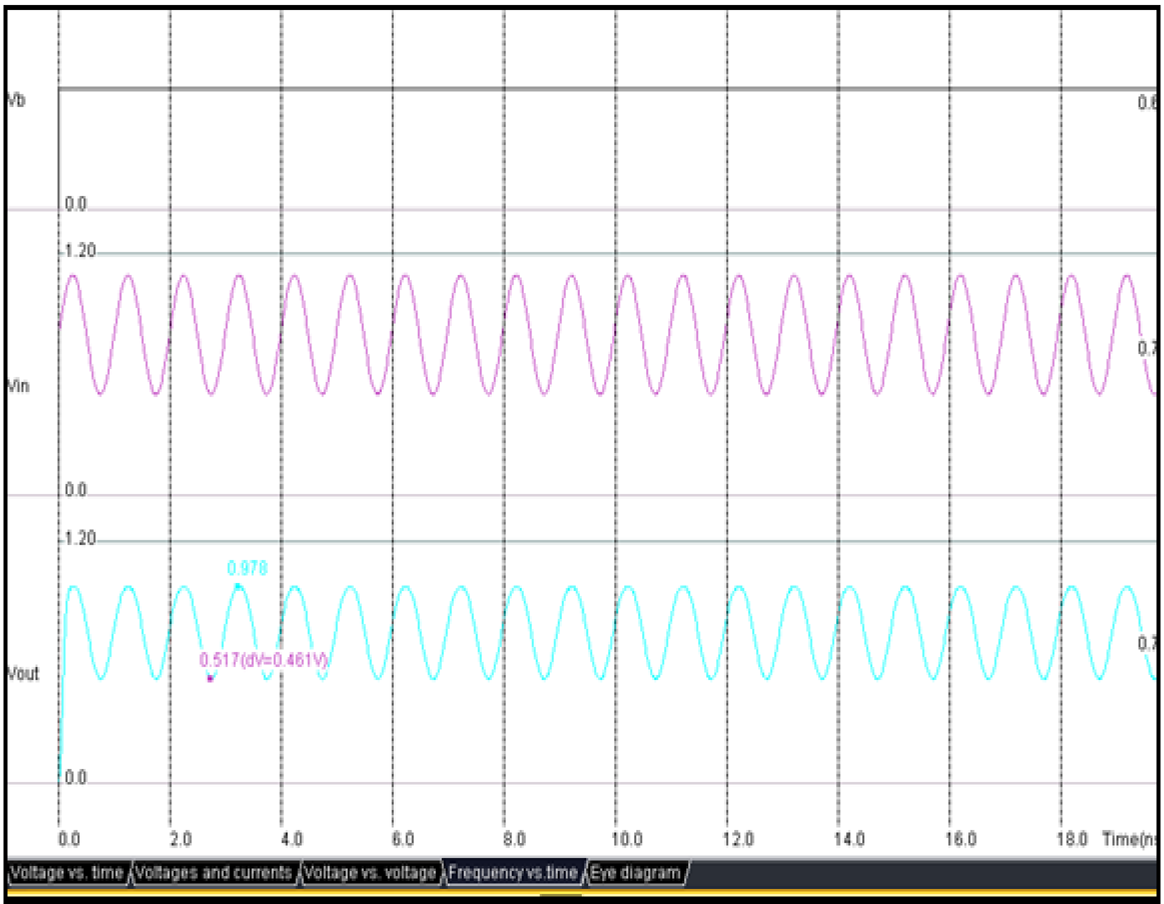
a. Draw the schematic and verify the following

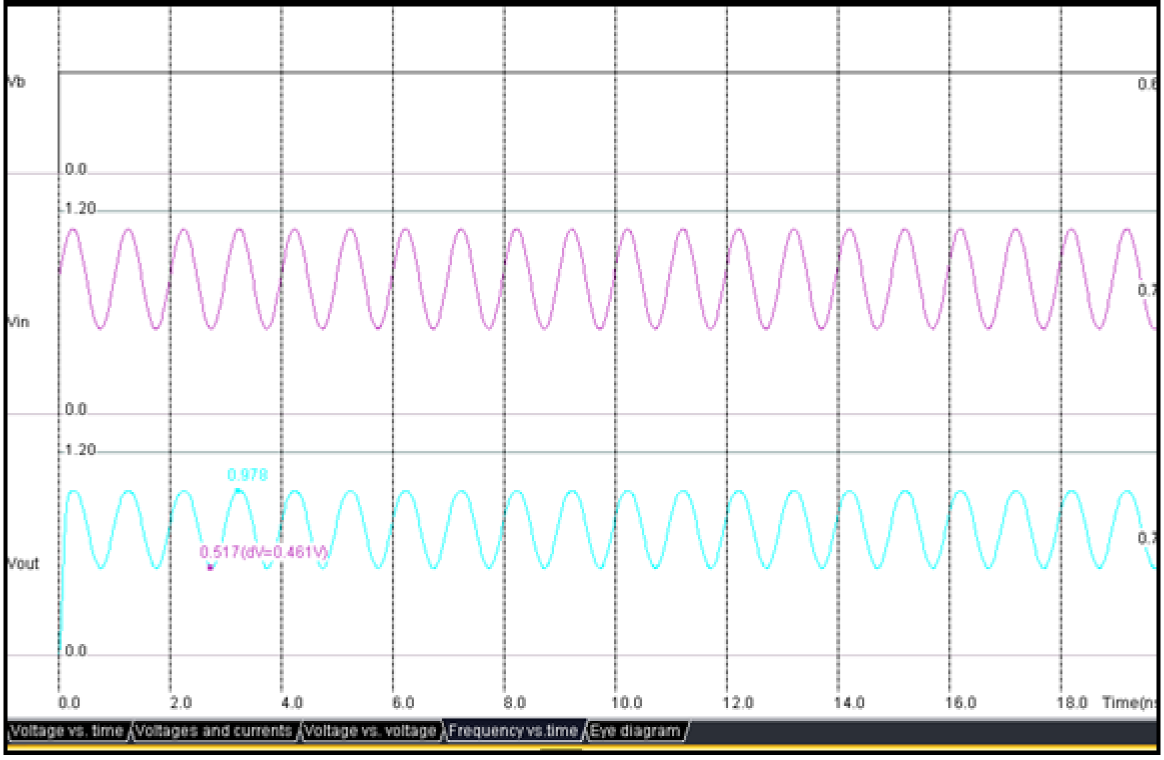
i) DC Analysis ii) AC Analysis iii) Transient Analysis

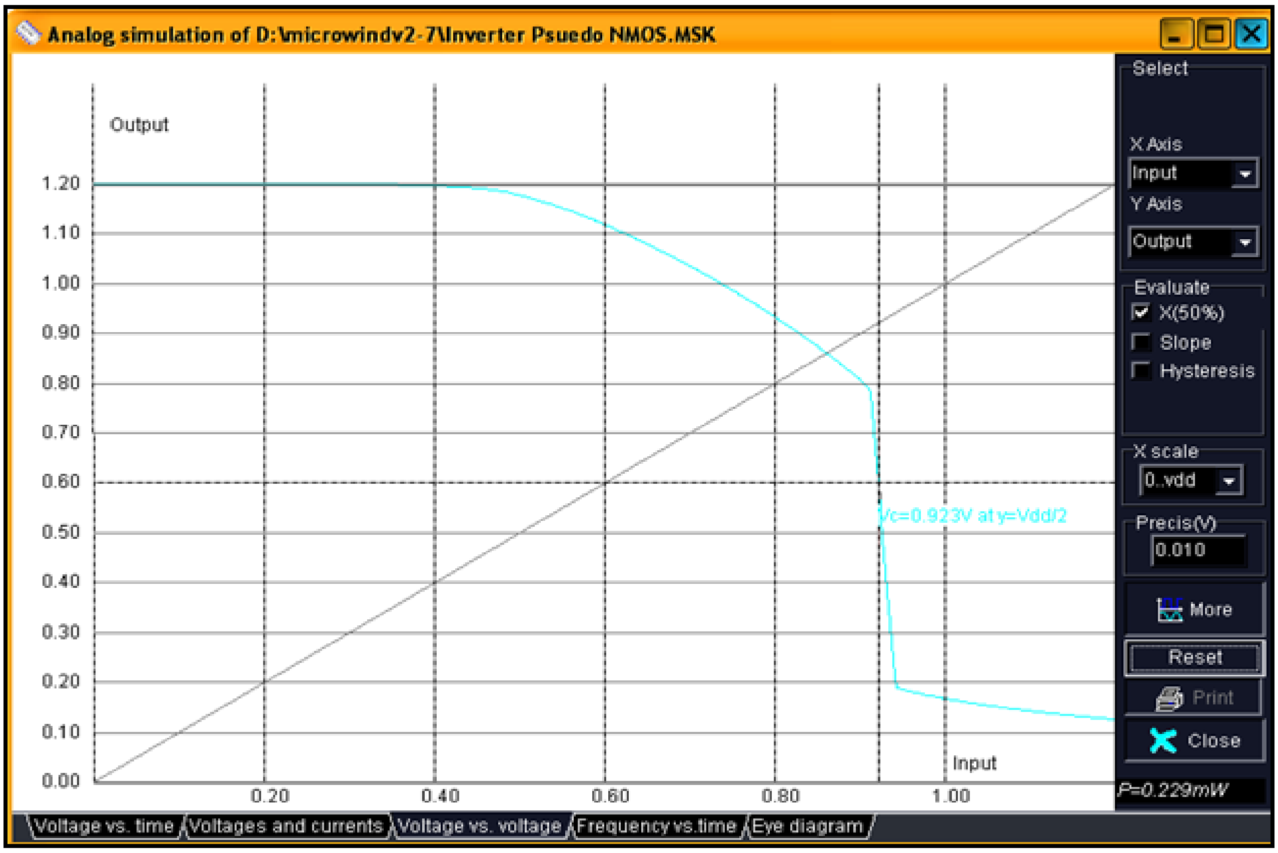
b. Draw the Layout and verify the DRC.

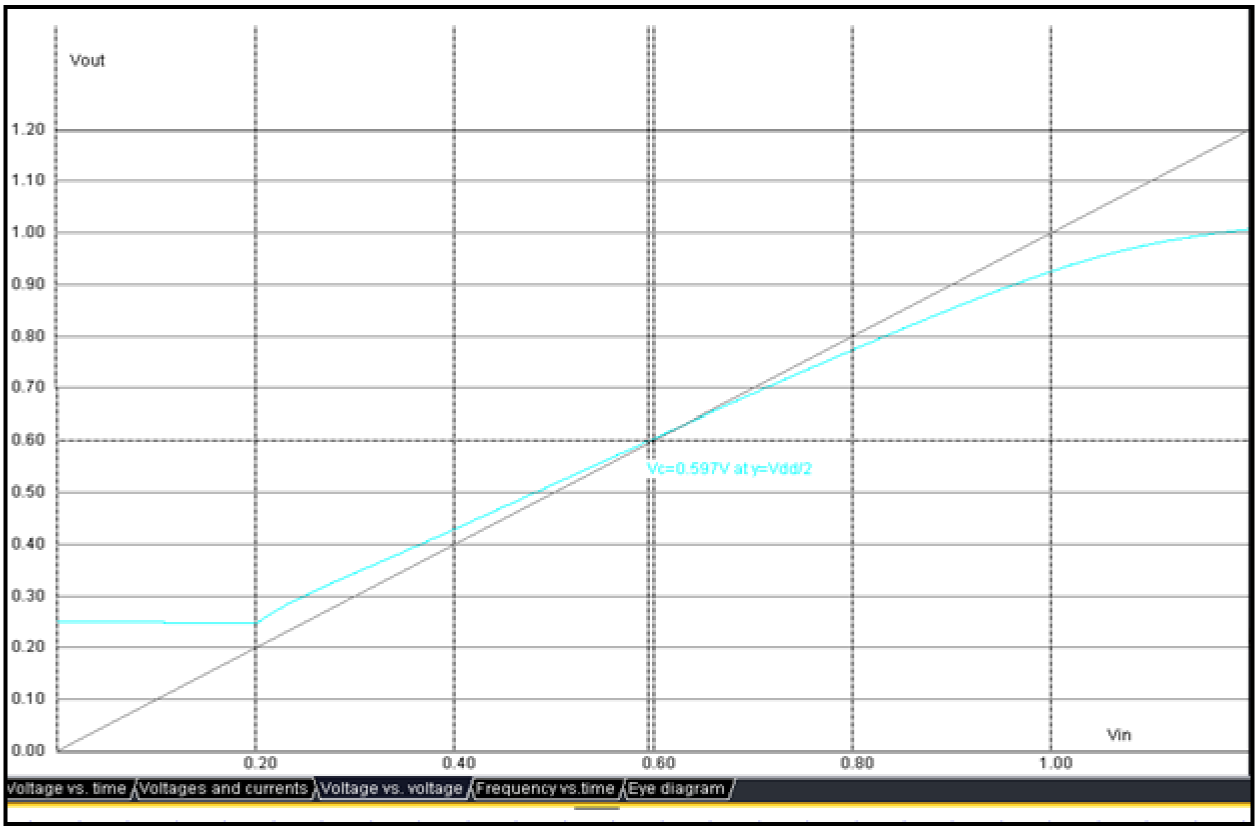
i) A Single Stage differential amplifier

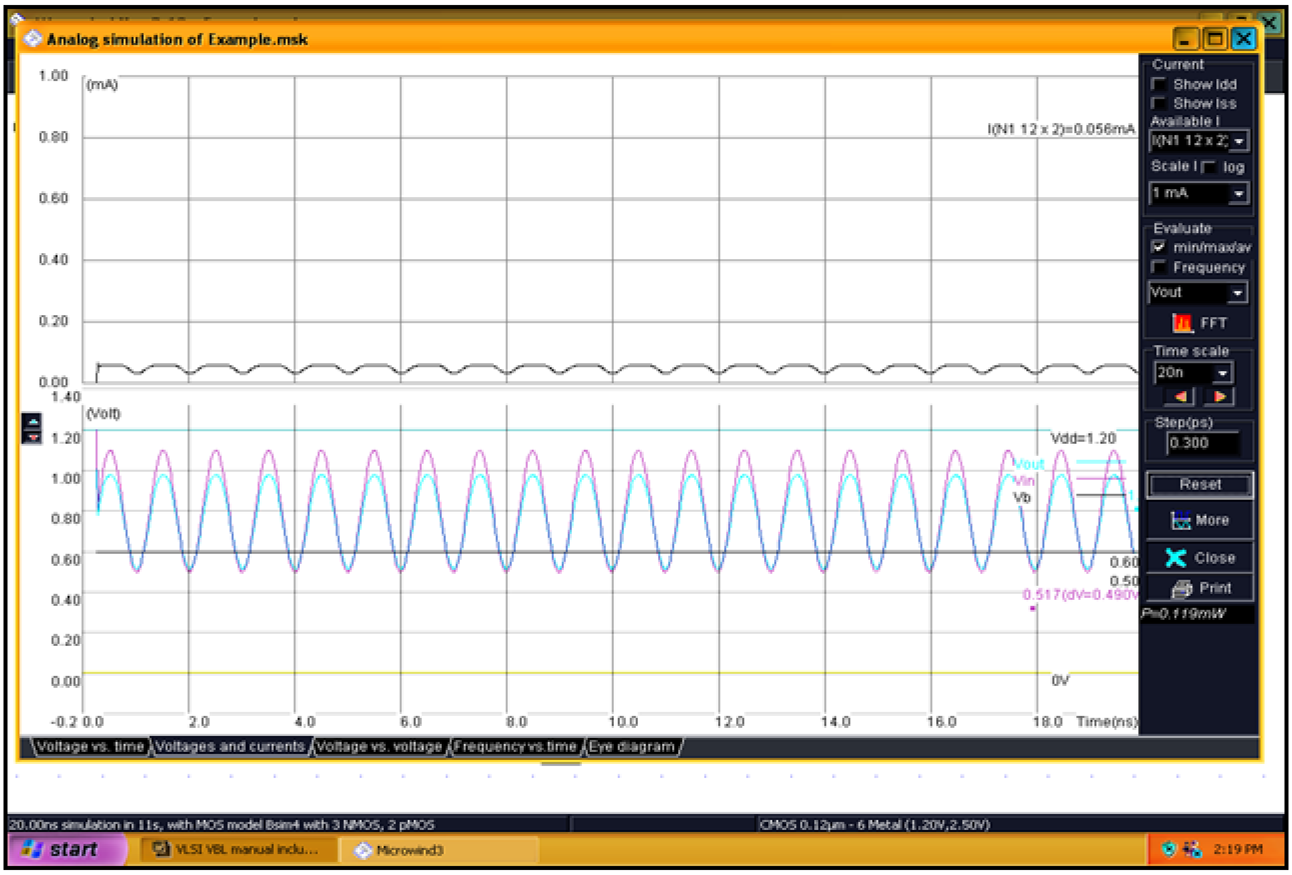



Vb
0.0
1.20
Vin
0.0
1.20
0.978
0.517(dV=0.461V)
Vout
0.0
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
Time(ns
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram


Vb
0.0
1.20
Vin
0.0
1.20
0.978
0.517(dV=0.461V)
Vout
0.0
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
Time(ns
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram


Analog simulation of D:\microwindv2-7\Inverter Psuedo NMOS.MSK
Output
Input
Vc=0.923V at y=Vdd/2
Select
X Axis
Input
Y Axis
Output
Evaluate
X(50%)
Slope
Hysteresis
X scale
0..vdd
Precis(V)
0.010
More
Reset
Print
Close
P=0.229mW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

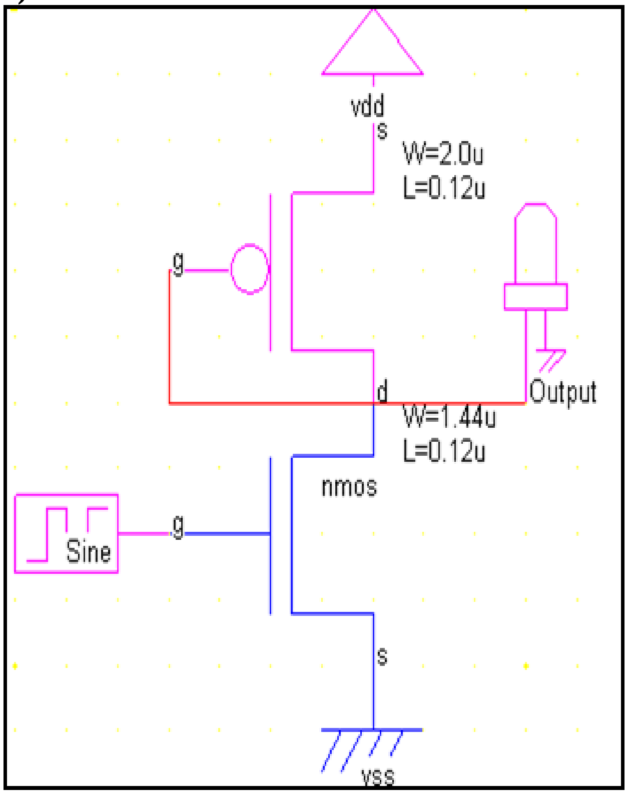

Vout
Vin
Vc=0.597V at y=Vdd/2
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

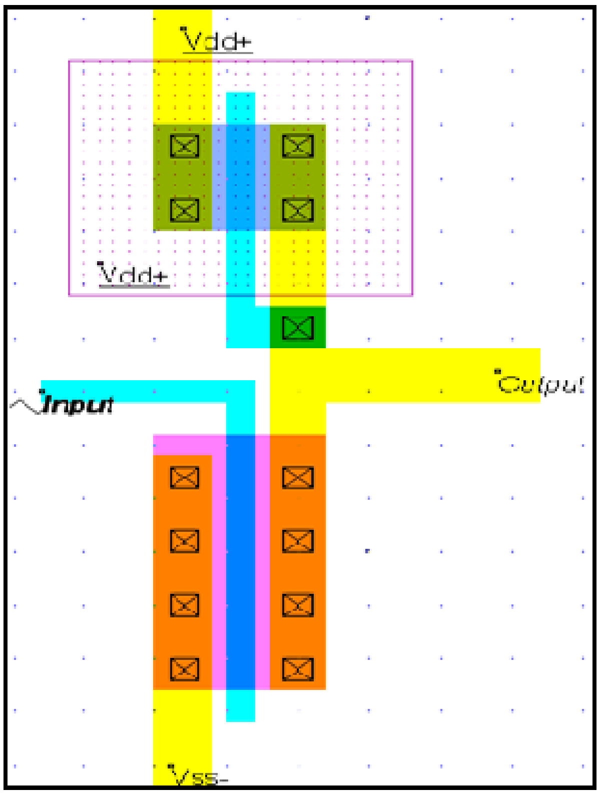


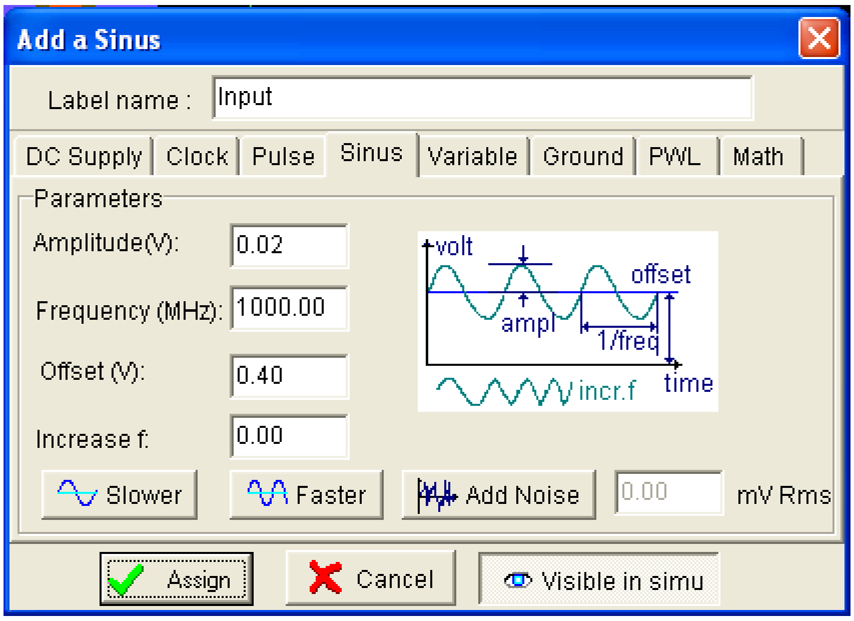
20.00ns simulation in 11s, with MOS model Bsim4 with 3 NMOS, 2 pMOS

CMOS 0.12µm - 6 Metal (1.20V,2.50V)

ii) Common source







Add a Sinus

Label name : Input

DC Supply | Clock | Pulse | Sinus | Variable | Ground | PWL | Math

Parameters

Amplitude(V): 0.02

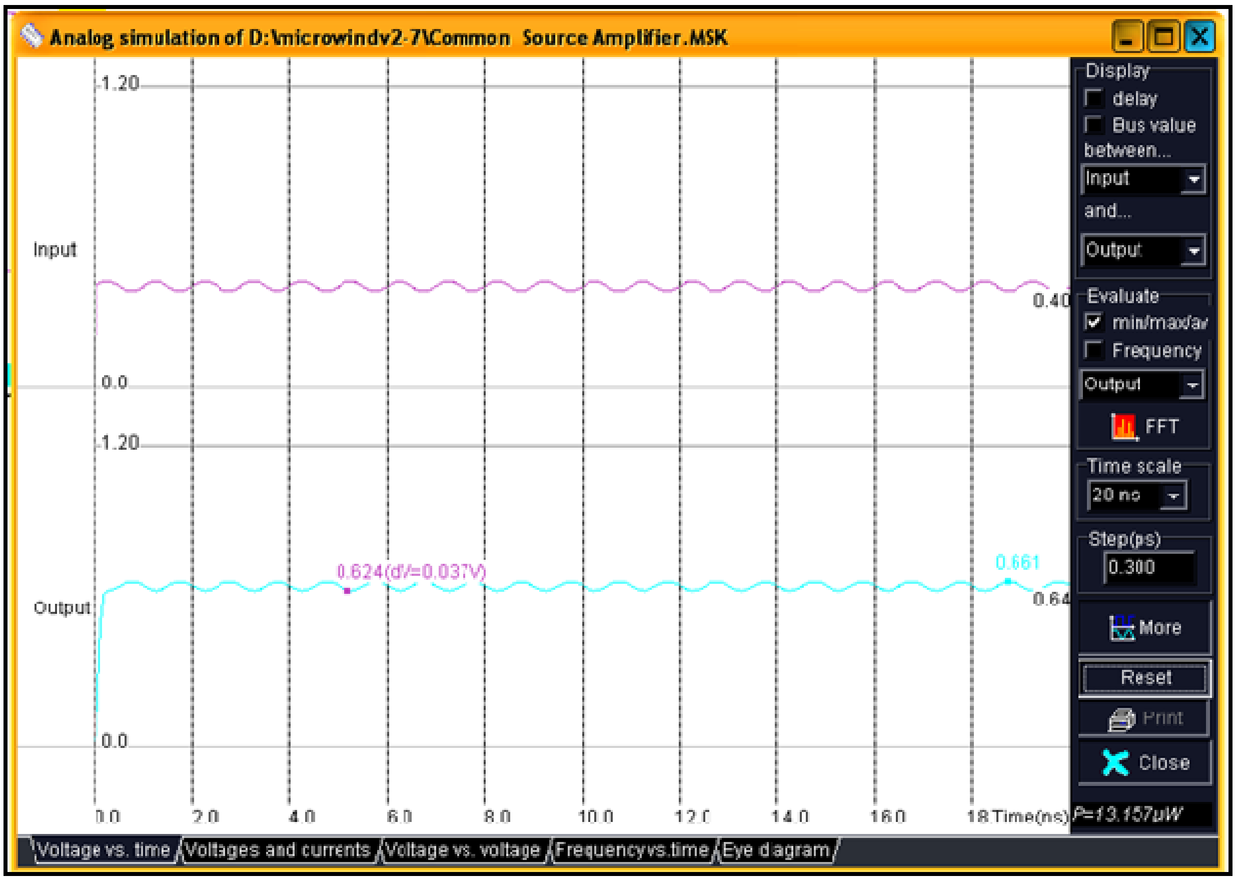
Frequency (MHz): 1000.00

Offset (V): 0.40

Increase f: 0.00

Slower | Faster | Add Noise | 0.00 mV Rms

Assign | Cancel | Visible in simu



Analog simulation of D:\microwindv2-7\Common Source Amplifier.MSK

Display: ☐ delay, ☐ Bus value between... Input and... Output

Evaluate: ☑ min/max/av, ☐ Frequency, Output, FFT

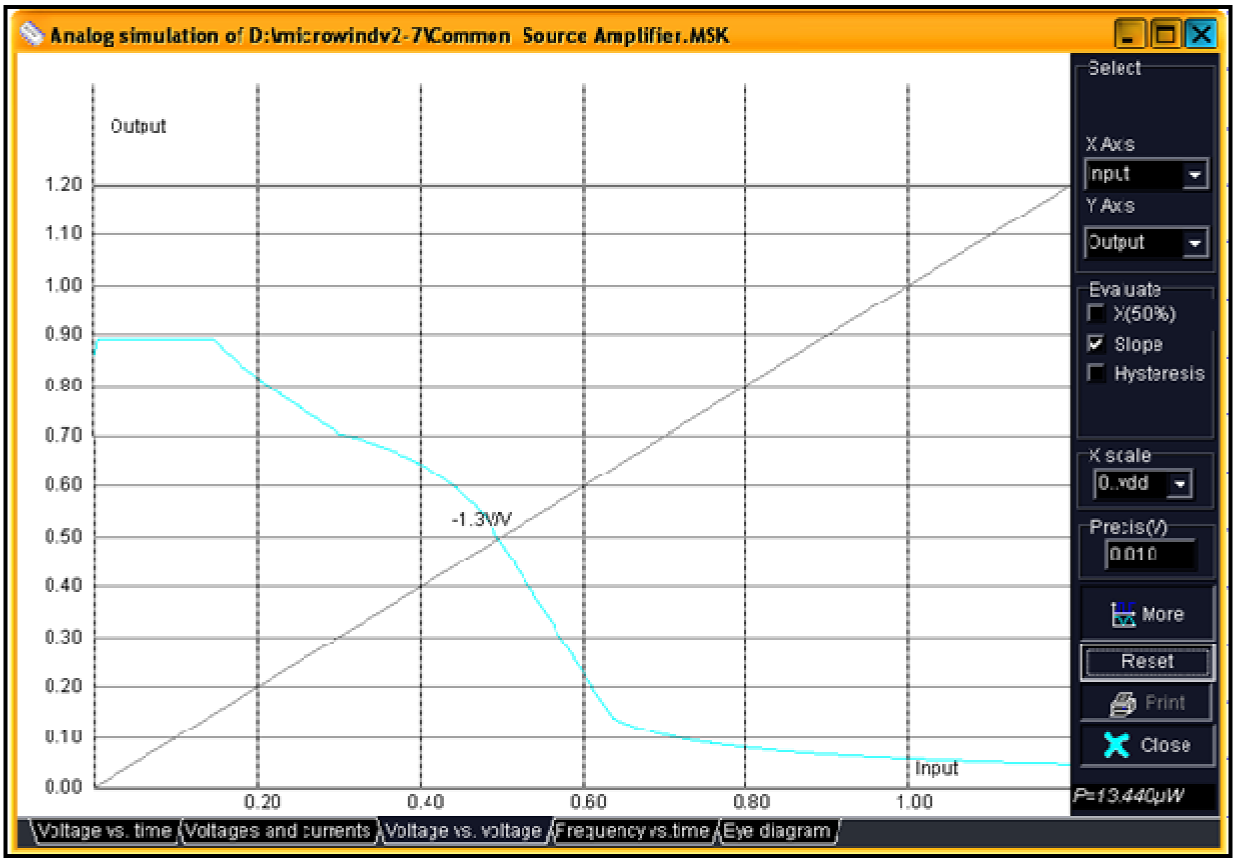
Time scale: 20 ns

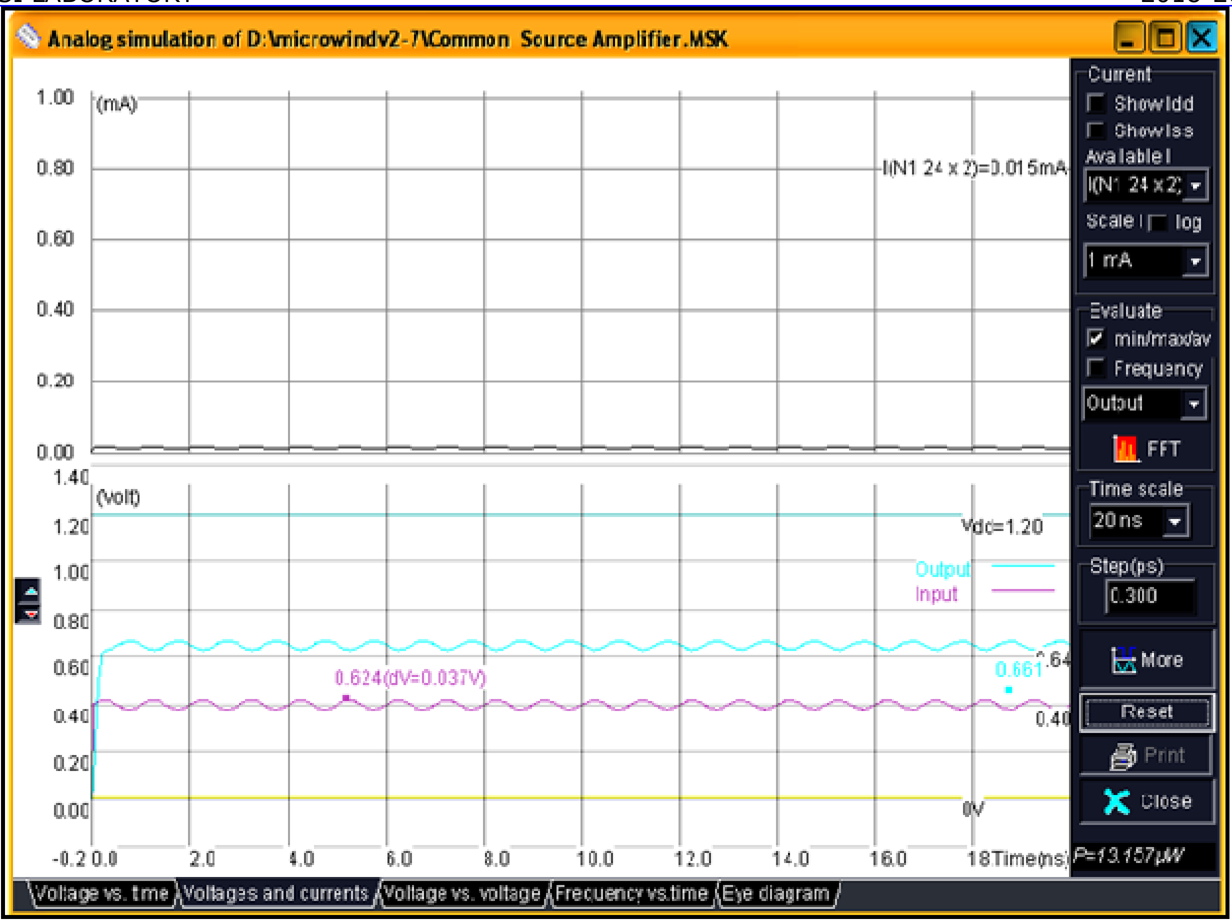
Step(ps): 0.300

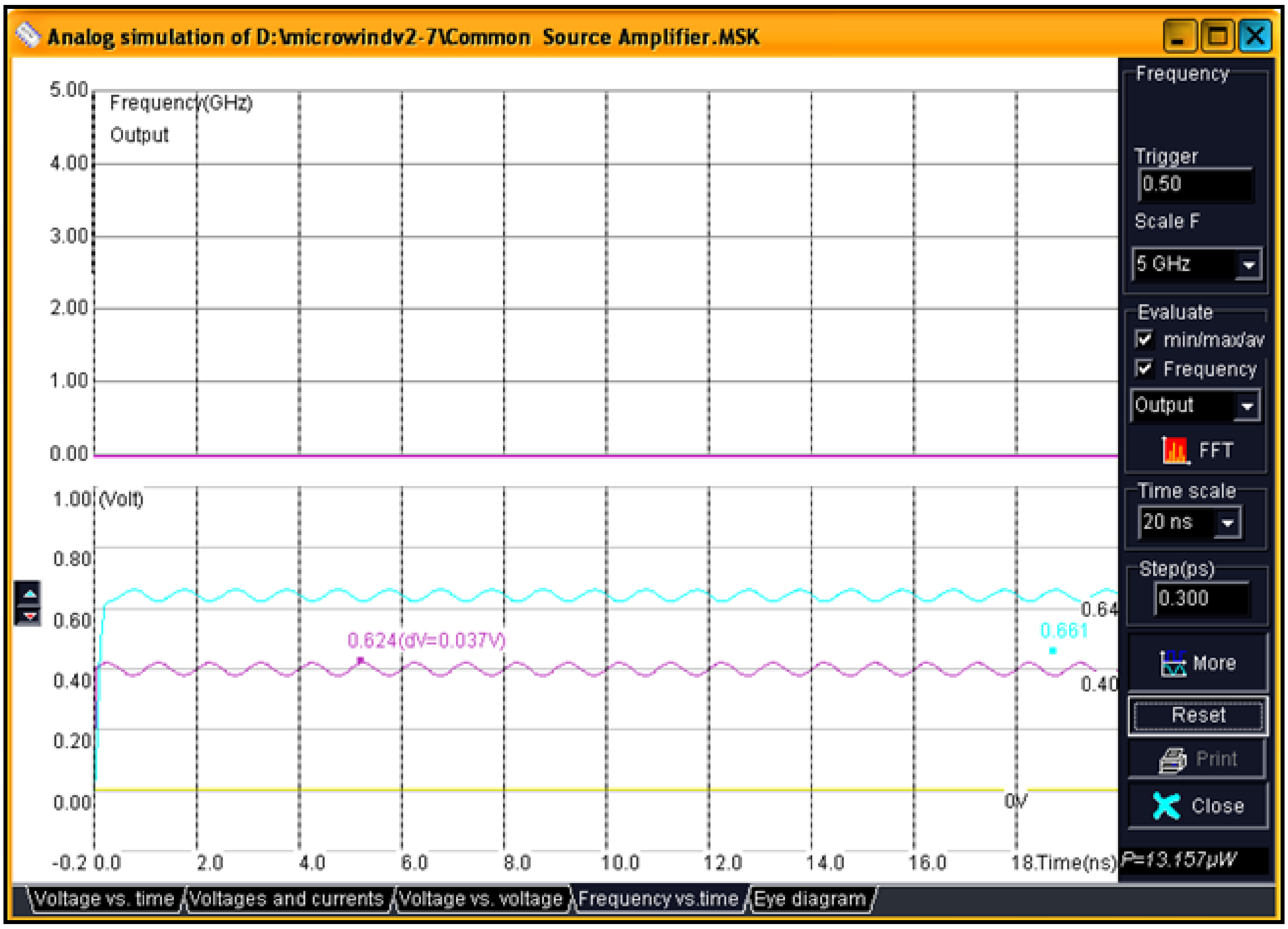
More | Reset | Print | Close

P=13.157uW

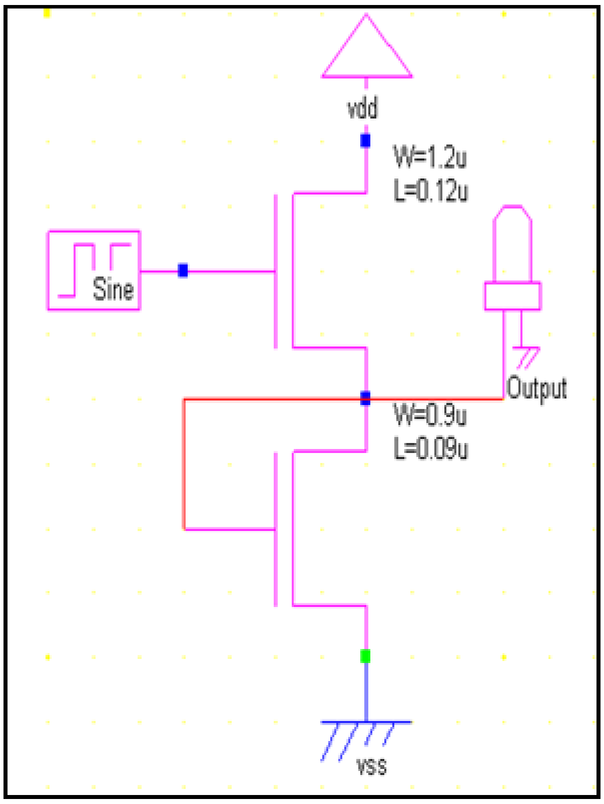
Voltage vs. time | Voltages and currents | Voltage vs. voltage | Frequency vs.time | Eye diagram

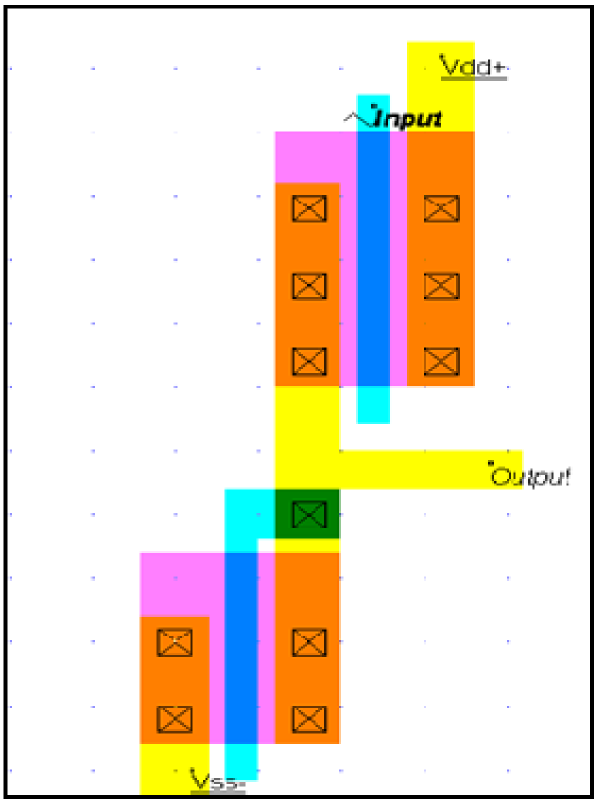

Analog simulation of D:\microwindv2-7\Common Source Amplifier.MSK
Output
Input
-1.3V/V
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.20
0.40
0.60
0.80
1.00
Select
X Axis
Input
Y Axis
Output
Evaluate
X(50%)
Slope
Hysteresis
X scale
0..vdd
Precis(V)
0.010
More
Reset
Print
Close
P=13.440µW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

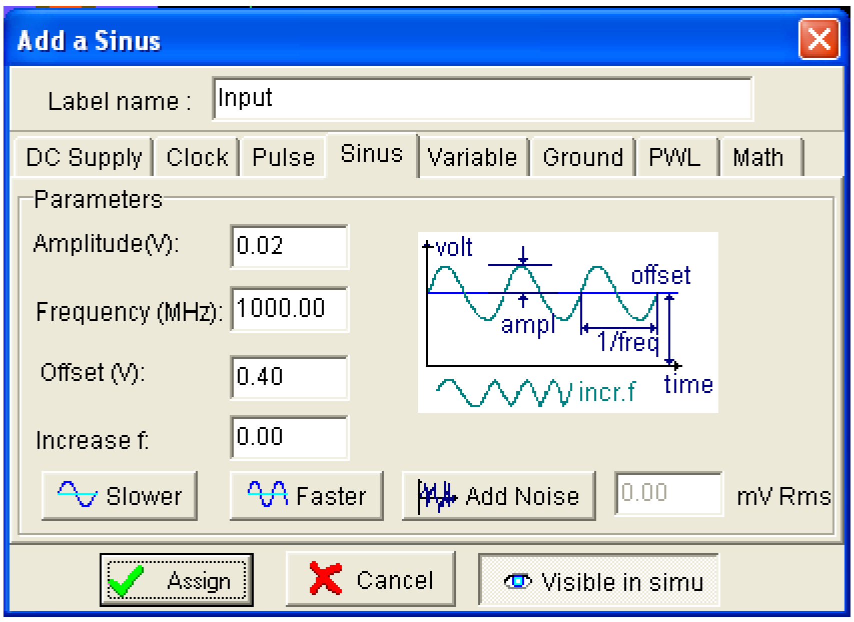

Analog simulation of D:\microwindv2-7\Common Source Amplifier.MSK
(mA)
I(N1 24 x 2)=0.015mA
(volt)
Vdd=1.20
Output
Input
0.624(dV=0.037V)
0.661
0.64
0.40
0V
Time(ns)
Current
ShowIdd
ShowIss
Available I
Scale I
log
1 mA
Evaluate
min/max/av
Frequency
Output
FFT
Time scale
20 ns
Step(ps)
0.300
More
Reset
Print
Close
P=13.157µW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram


Analog simulation of D:\microwindv2-7\Common Source Amplifier.MSK
Frequency(GHz)
Output
(Volt)
0.624(dV=0.037V)
0.661
0.64
0.40
0V
Time(ns)
Frequency
Trigger
0.50
Scale F
5 GHz
Evaluate
min/max/av
Frequency
Output
FFT
Time scale
20 ns
Step(ps)
0.300
More
Reset
Print
Close
P=13.157µW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

ii)Common Drain amplifier







Label name : Input

Parameters

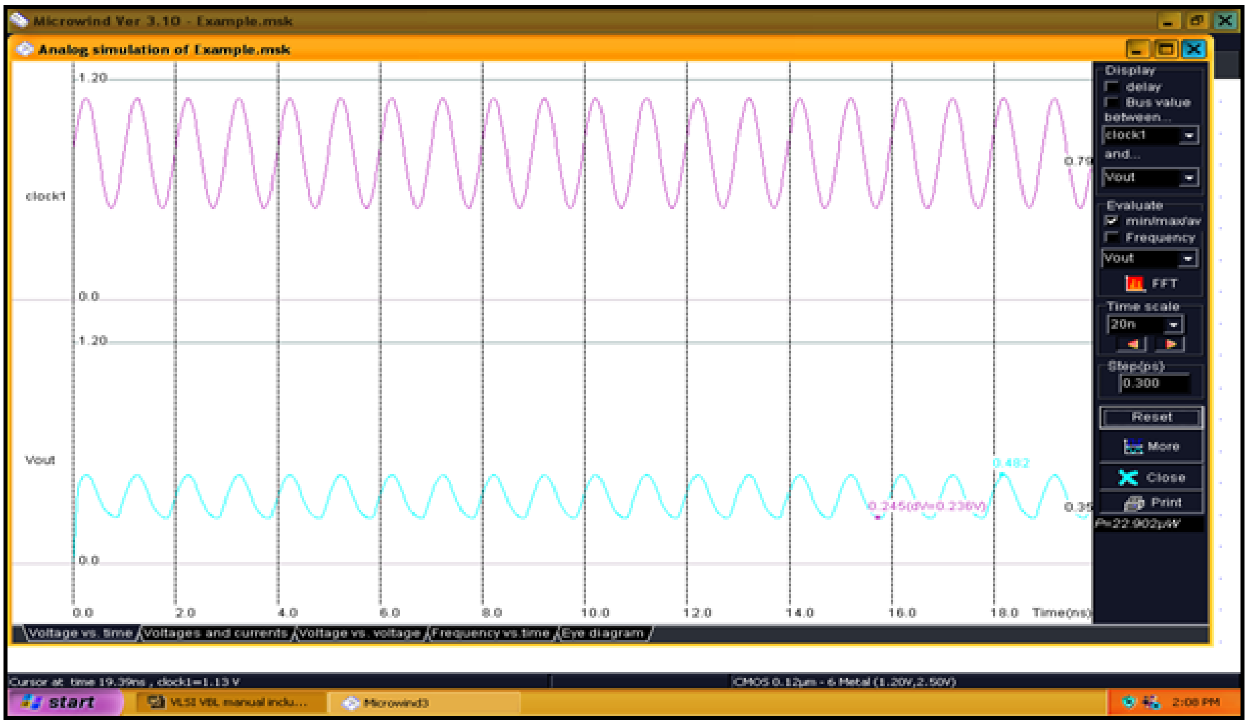
Amplitude(V): 0.02

Frequency (MHz): 1000.00

Offset (V): 0.40

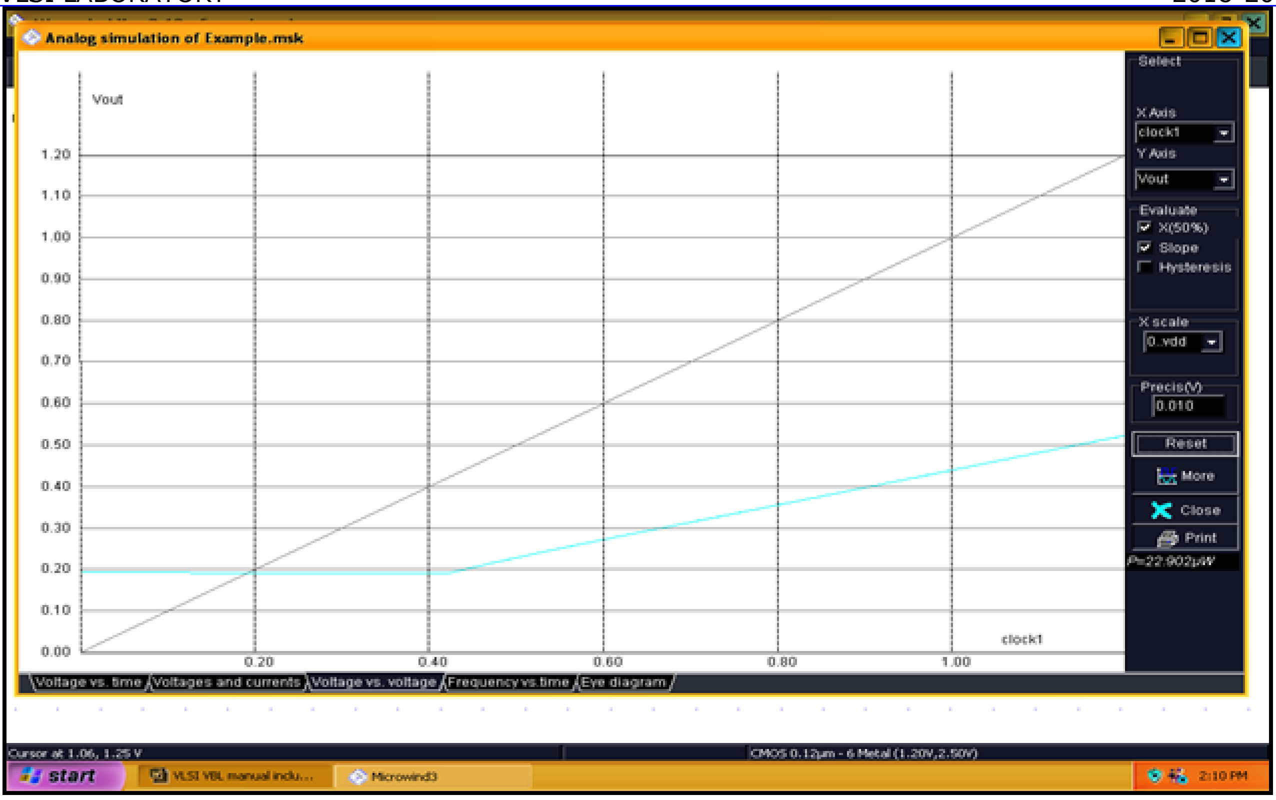
Increase f: 0.00

0.00 mV Rms

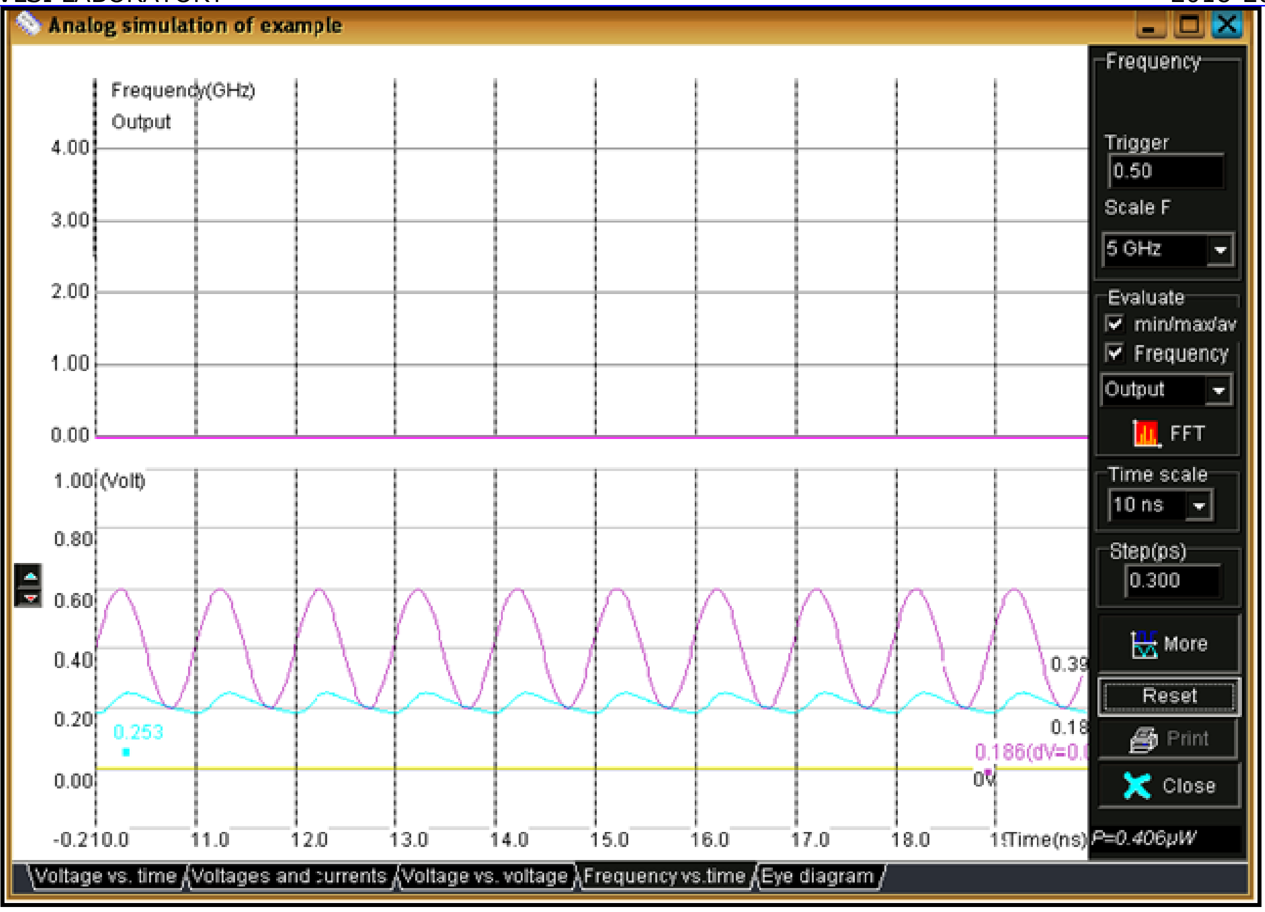


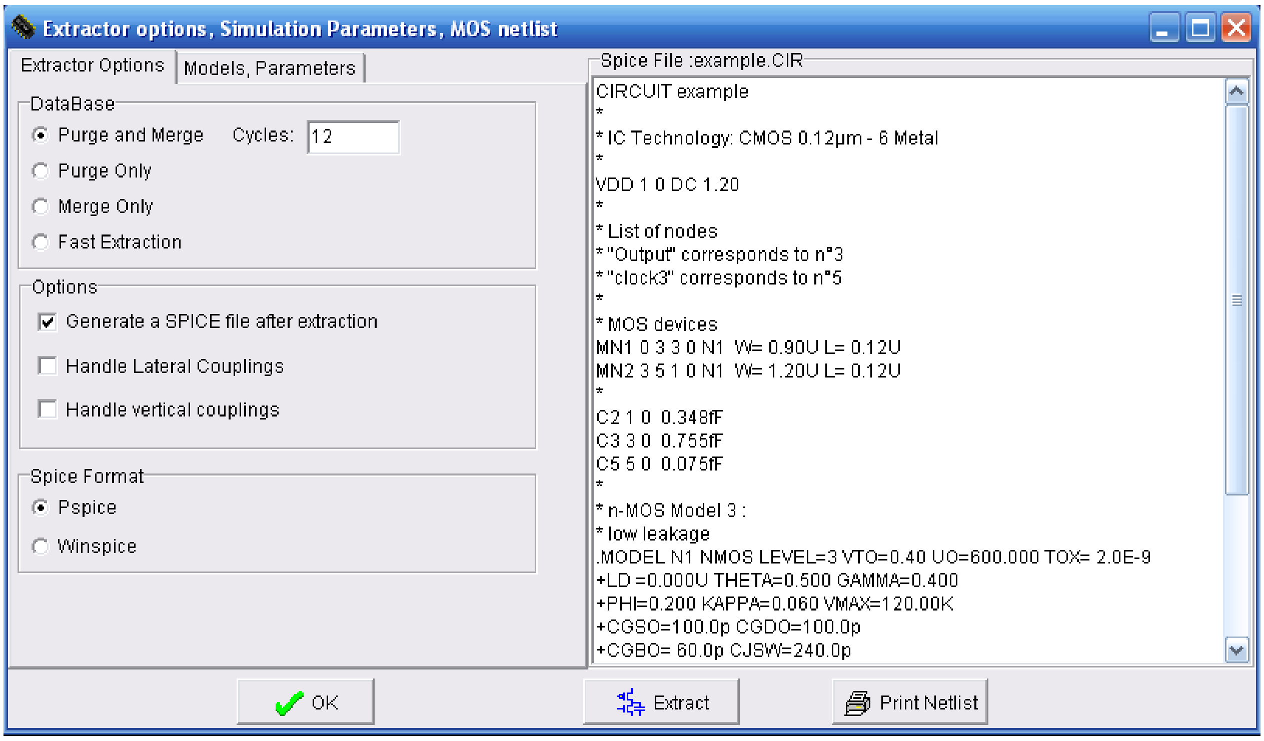
Cursor at time 19.39ns , clock1=1.13 V

CMOS 0.12µm - 6 Metal (1.20V,2.50V)



X Axis: clock1
Y Axis: Vout
X scale: 0..vdd
Precis(V): 0.010
P=22.902µW
Cursor at 1.06, 1.25 V
CMOS 0.12µm - 6 Metal (1.20V,2.50V)





```
CIRCUIT example
*
* IC Technology: CMOS 0.12µm - 6 Metal
*
VDD 1 0 DC 1.20
*
* List of nodes
* "Output" corresponds to n°3
* "clock3" corresponds to n°5
*
* MOS devices
MN1 0 3 3 0 N1  W= 0.90U L= 0.12U
MN2 3 5 1 0 N1  W= 1.20U L= 0.12U
*
C2 1 0  0.348fF
C3 3 0  0.755fF
C5 5 0  0.075fF
*
* n-MOS Model 3 :
* low leakage
.MODEL N1 NMOS LEVEL=3 VTO=0.40 UO=600.000 TOX= 2.0E-9
+LD =0.000U THETA=0.500 GAMMA=0.400
+PHI=0.200 KAPPA=0.060 VMAX=120.00K
+CGSO=100.0p CGDO=100.0p
+CGBO= 60.0p CJSW=240.0p
```

3. Design an **op-amp** with given specification* using given differential amplifier Common source and Common Drain amplifier in library** and completing the design flow mentioned below:

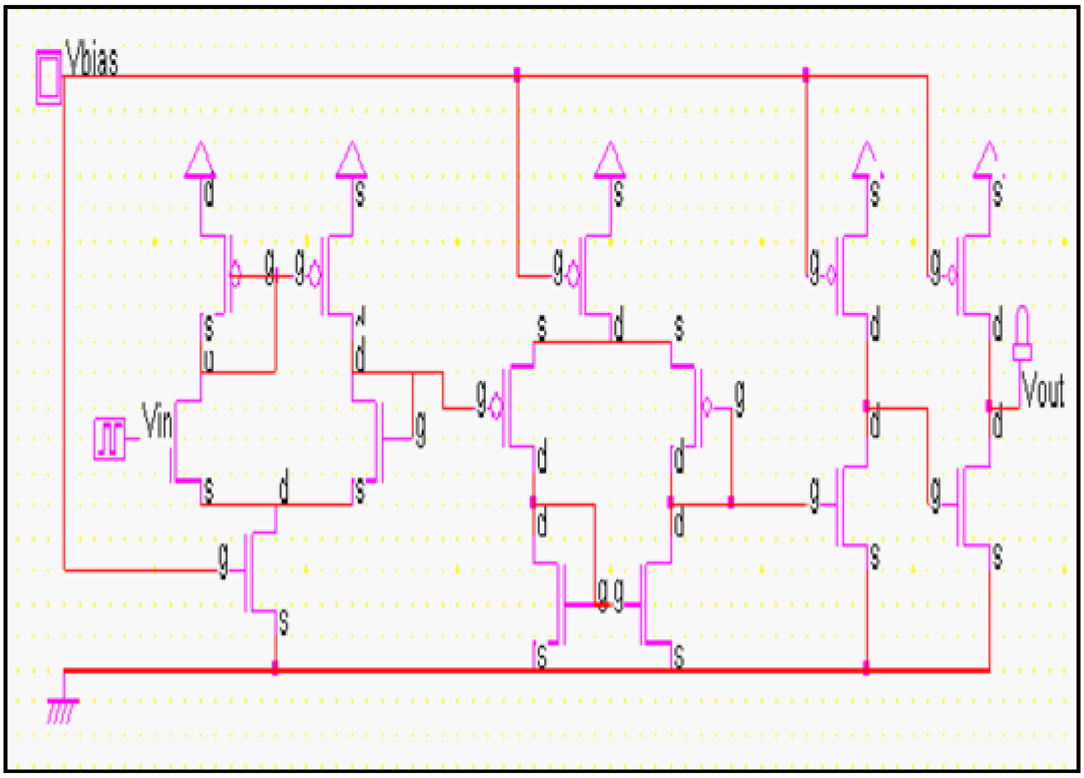
a. Draw the schematic and verify the following

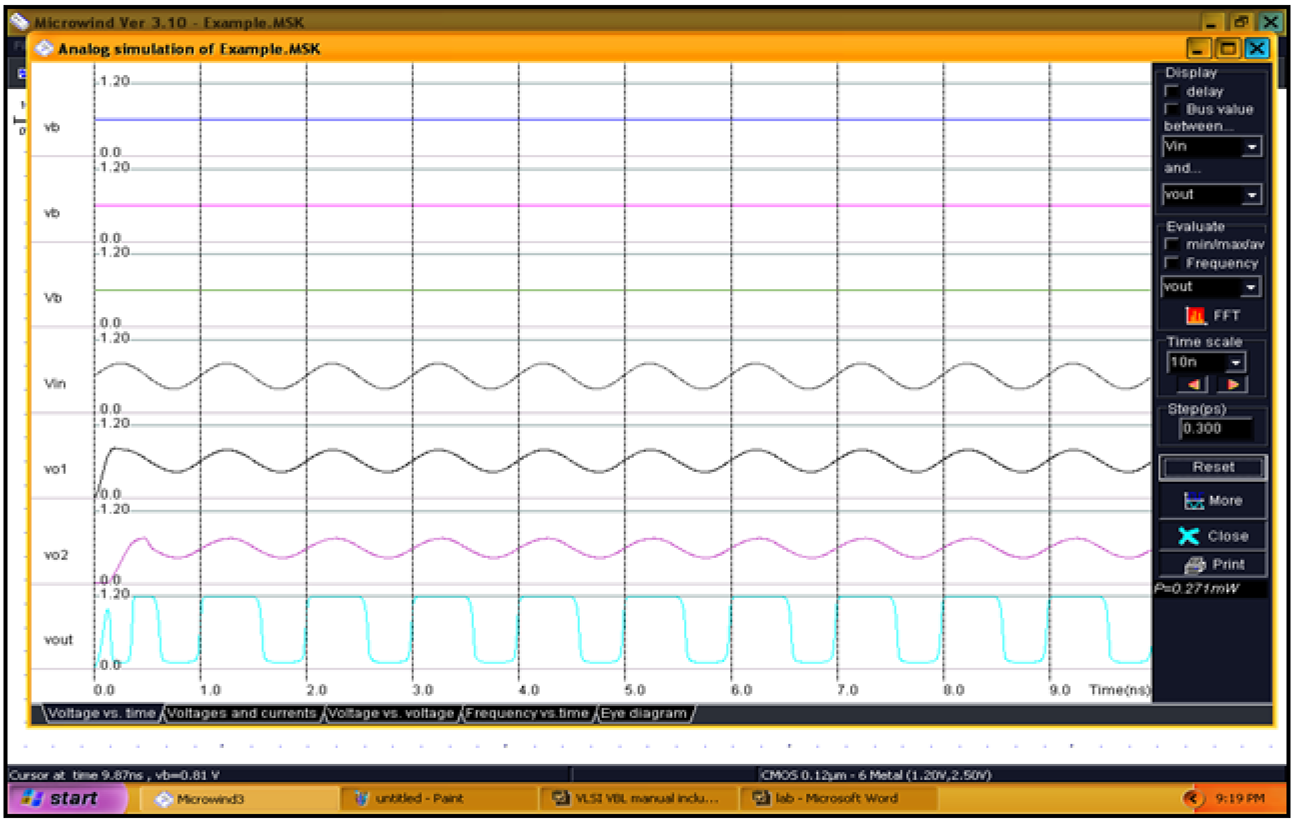
i) DC Analysis

ii). AC Analysis

iii) Transient Analysis

b. Draw the Layout and verify the DRC,





between... Vin and... vout

vout

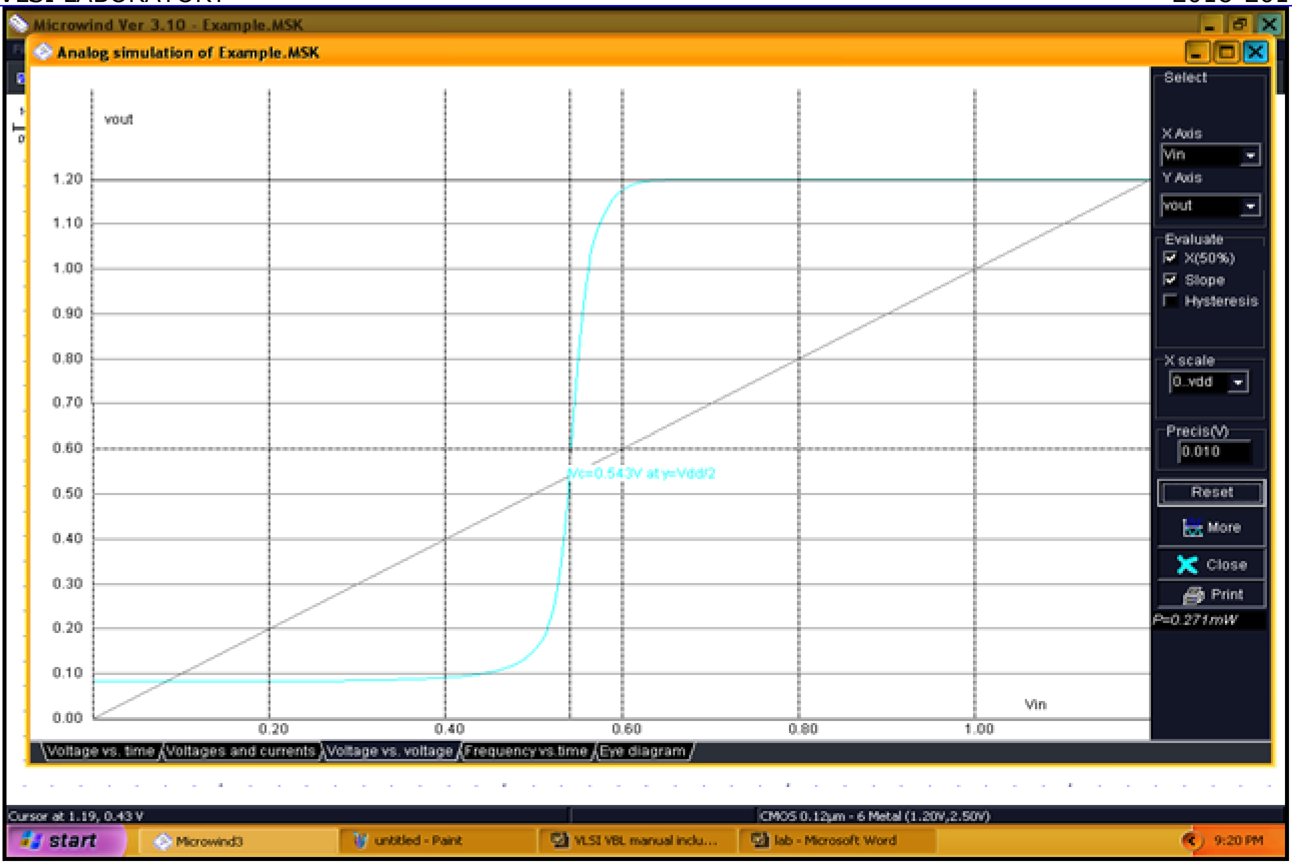
Time scale 10n

Step(ps) 0.300

P=0.271mW

Cursor at time 9.87ns , vb=0.81 V

CMOS 0.12µm - 6 Metal (1.20V,2.50V)



X Axis: Vin
Y Axis: vout
☑ X(50%)
☑ Slope
☐ Hysteresis
X scale: 0..vdd
Precis(V): 0.010
Vc=0.543V at y=Vdd/2
P=0.271mW
Cursor at 1.19, 0.43 V
CMOS 0.12µm - 6 Metal (1.20V,2.50V)

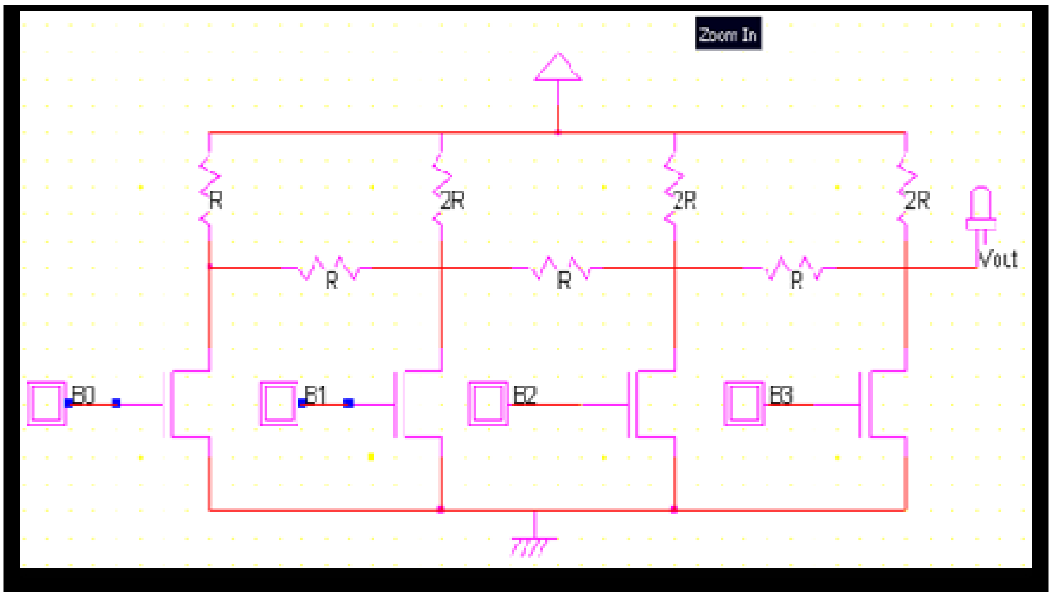
4. Design a **4 bit R-2R based DAC** for the given specification and completing the design flow mentioned using given op-amp in the library**.

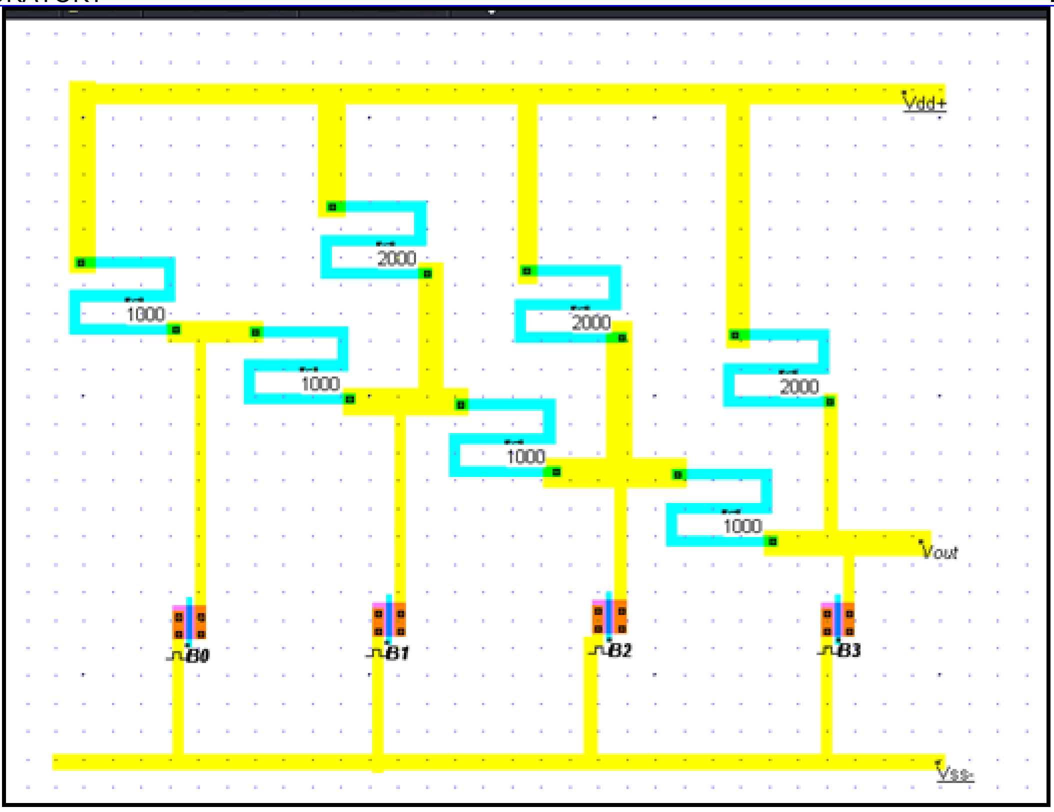
a. Draw the schematic and verify the following

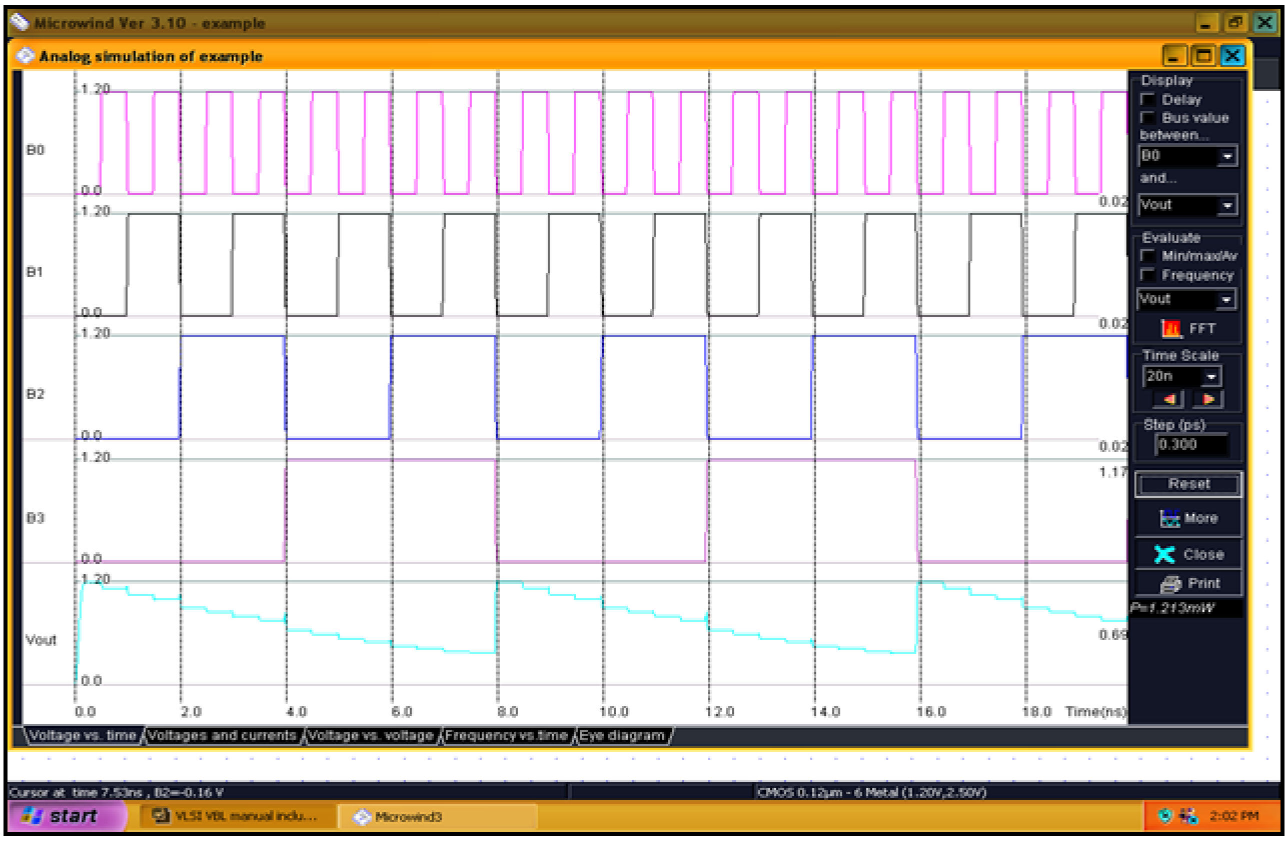
i) DC Analysis

ii) AC Analysis

iii) Transient Analysis

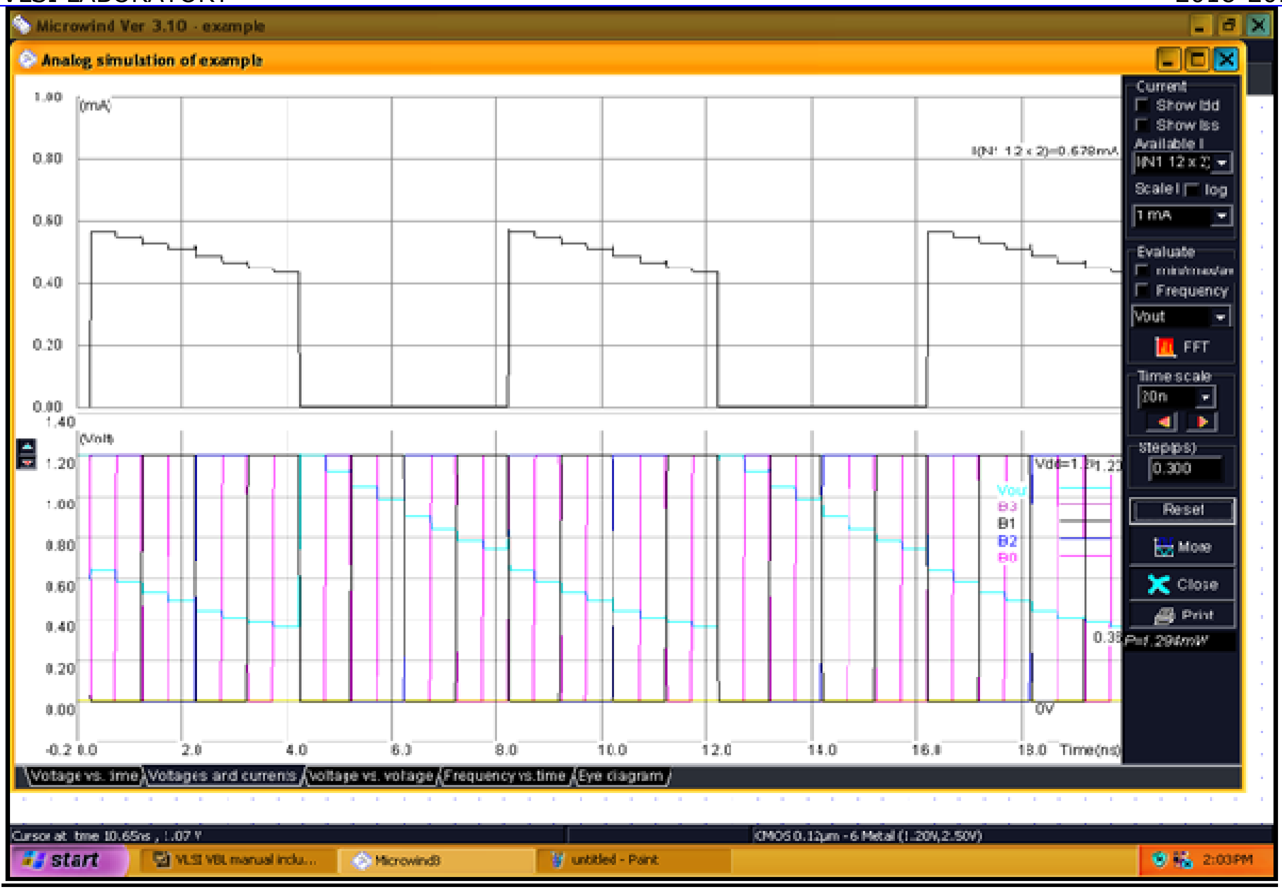






Cursor at time 7.53ns , B2=-0.16 V

CMOS 0.12µm - 6 Metal (1.20V,2.50V)



Probable viva questions

1. What is a latch

2. Why is NAND gate preferred over NOR gate for fabrication?

3) What is Noise Margin? Explain the procedure to determine Noise Margin

4) Explain sizing of the inverter?

5) How do you size NMOS and PMOS transistors to increase the threshold voltage?

6) What is Noise Margin? Explain the procedure to determine Noise Margin?

7) What happens to delay if you increase load capacitance?

8) What happens to delay if we include a resistance at the output of a CMOS circuit?

9) What are the limitations in increasing the power supply to reduce delay?

10) How does Resistance of the metal lines vary with increasing thickness and increasing length?

11) For CMOS logic, give the various techniques you know to minimize power consumption?

12) What is Charge Sharing? Explain the Charge Sharing problem while sampling data from a Bus?

13) Why do we gradually increase the size of inverters in buffer design? Why not give the output of a circuit to one large inverter?

14) What is Latch Up? Explain Latch Up with cross section of a CMOS Inverter. How do you avoid Latch Up?

15) Give the expression for CMOS switching power dissipation?

16) What is Body Effect?

17) Why is the substrate in NMOS connected to Ground and in PMOS to VDD?

18) What is the fundamental difference between a MOSFET and BJT ?

19) Which transistor has higher gain. BJT or MOS and why?

20) Why do we gradually increase the size of inverters in buffer design when trying to drive a high capacitive load? Why not give the output of a circuit to one large inverter?

21) In CMOS technology, in digital design, why do we design the size of pmos to be higher than the nmos.What determines the size of pmos wrt nmos. Though this is a simple question try to list all the reasons possible?

22) Why PMOS and NMOS are sized equally in a Transmission Gates?

23) All of us know how an inverter works. What happens when the PMOS and NMOS are interchanged with one another in an inverter?

24)A good question on Layouts. Give 5 important Design techniques you would follow when doing a Layout for Digital Circuits?

25) What is metastability? When/why it will occur?Different ways to avoid this?

26) Let A and B be two inputs of the NAND gate. Say signal A arrives at the NAND gate later than signal B. To optimize delay of the two series NMOS inputs A and B which one would you place near to the output?

27) Explain about setup time and hold time, what will happen if there is setup time and hold tine violation, how to overcome this problem?

28) What is skew, what are problems associated with it and how to minimize it?

29) What is slack?

30) What is glitch? What causes it (explain with waveform)? How to overcome it?

31) Given only two xor gates one must function as buffer and another as inverter?

32) What is difference between latch and flipflop?

33) Build a 4:1 mux using only 2:1 mux?

34) Difference between heap and stack?

35) Difference between mealy and moore state machine?

36) What are different ways to synchronize between two clock domains?

37) How to calculate maximum operating frequency?

38) How to find out longest path?

49) Tell some of applications of buffer?

40).How can you convert an SR Flip-flop to a JK Flip-flop?

41) How can you convert the JK Flip-flop to a D Flip-flop?

42) What is false path? How it determine in circuit? What the effect of false path in circuit?

43)Design s 2 input NAND gate?

44) Design 2 input NOR gate?

45) Design 3 input NAND and NOR gate?

46) Design 2input xnor gate?

47) Design 2 input xor gate?

48) Design 3 input pseudonmos gate?

49) Design 3 input CMOS DOMINO gate?

50) Design 3 input CMOS DYNAMIC gate?