ANNA UNIVERSITY, CHENNAI

(R 2008)

EC2357 - VLSI DESIGN LABORATORY

SYLLABUS

1. Design Entry and simulation of combinational logic circuits (8 bit adders, 4 bit multipliers,address decoders, multiplexers), Test bench creation, functional verification, andconcepts of concurrent and sequential execution to be highlighted.

2. Design Entry and simulation of sequential logic circuits (counters, PRBS generators,accumulators). Test bench creation, functional verification, and concepts of concurrent and sequential execution to be highlighted.

3. Synthesis, P&R and Post P&R simulation for all the blocks/codes developed in Expt.No. 1 and No. 2 given above. Concepts of FPGA floor plan, critical path, design gate count, I/O configuration and pin assignment to be taught in this experiment.

4. Generation of configuration/fuse files for all the blocks/codes developed as part of Expt.1. and Expt. 2. FPGA devices must be configured and hardware tested for the blocks/codes developed as part of Expt. 1. and Expt. 2. The correctness of the inputs and outputs for each of the blocks must be demonstrated atleast on oscilloscopes (logic analyzer preferred).

5. Schematic Entry and SPICE simulation of MOS differential amplifier. Determination of gain, bandwidth, output impedance and CMRR.

6. Layout of a simple CMOS inverter, parasitic extraction and simulation.

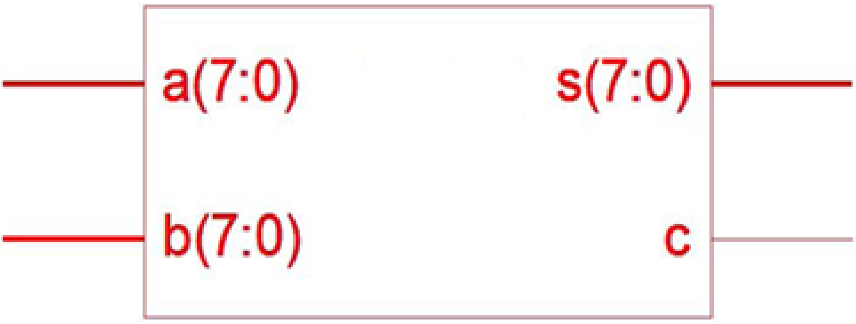
7. Design of a 10 bit number controlled oscillator using standard cell approach, simulation followed by study of synthesis reports.

8. Automatic layout generation followed by post layout extraction and simulation of the circuit studied in Expt. No.7

LIST OF EXPERIMENTS

EXP. NO	NAME OF THE EXPERIMENT	PAGE. NO
	CYCLE I	
1.	Simulation and Implementation of 8 Bit Adder	
2.	Simulation and Implementation of 4 bit multiplier	
3.	Simulation and Implementation of 2x4 Decoder	
4.	Simulation and Implementation of 4x1 multiplexer	
5.	Simulation and Implementation of 2- Bit Counter	
6.	Simulation and Implementation of accumulator	
	CYCLE II	
7.	Simulation and implementation of PRBS generator.	
8.	Simulation of CMOS Inverter Using	
9.	Simulation of MOS Differential Amplifier	
10.	Simulation of Simple CMOS Inverter Using	
11.	Design and Simulation Of 10 Bit Number Controlled Oscillator Using	

8 BIT ADDER:



EX NO: 1 SIMULATION AND IMPLEMENTATION OF 8 BIT ADDER

DATE:

AIM:

To design and simulate the 8-bit adder using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

PROGRAM:

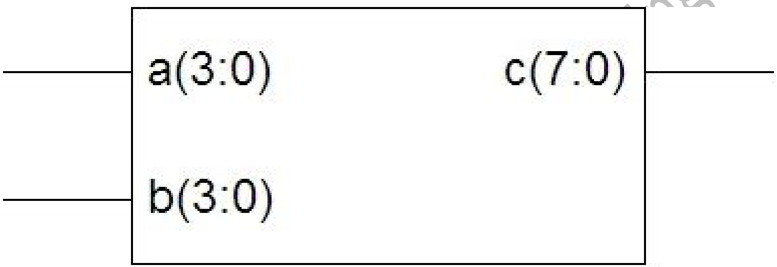
```
module adder(a,b, s,c);
input [7:0] a,b;
output [7:0] s,c;
assign {c,s} = a + b;
endmodule
```

TESTBENCH:

```
module testbenadder;
//Inputs
reg [7:0] a,b;
//outputs
wire [7:0] s;
wire c;
adder add(.s(s),.c(c),.a(a),.b(b));
initial
begin
```

```
//initialize input
a=8'd0; b=8'd0;
//wait 100ns for global reset to finish
#100; a=8'd1; b=8'd0;
#100; a=8'd9; b=8'd5;
#100; a=8'd5; b=8'd7;
end
endmodule
```

4 BIT MULTIPLIER:



EX NO: 2 **SIMULATION AND IMPLEMENTATION OF 4 BIT**
DATE: **MULTIPLIER**

AIM:

To design and simulate the 4 bit multiplier using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator, create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

PROGRAM

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

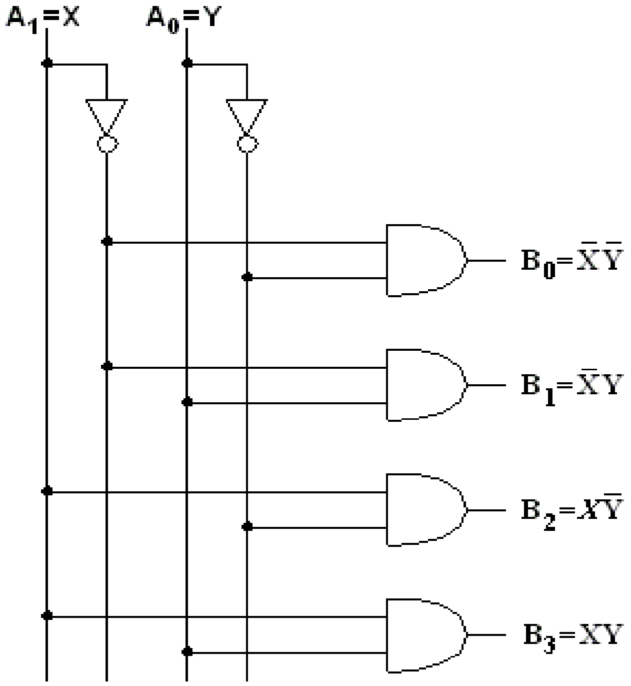
TESH BENCH:

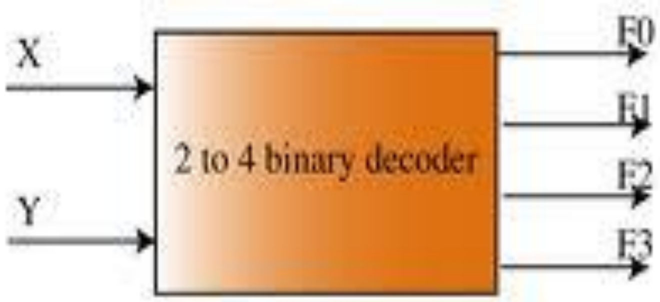
```
module testbenchmulti ;
//inputs
reg [3:0] a,b;
//outputs
wire [7:0] c;
multi multipl(.c(c),.a(a),.b(b));
initial
begin
a=4'b0; b=4'b0;
//wait 100ns for global reset to finish
#100; a=4'd3; b=4'd4;
#100; a=4'd3; b=4'd5;
#100; a=4'd2; b=4'd4;
end
endmodule
```

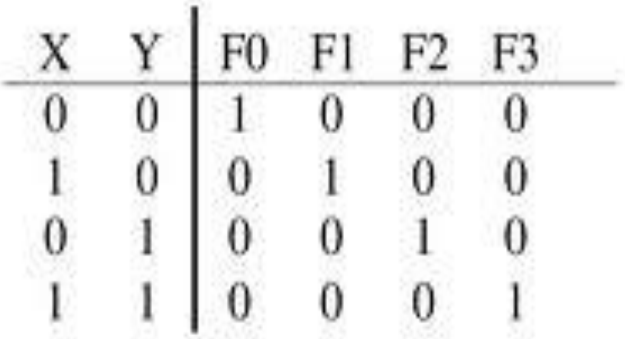
RESULT:

Thus the 4 bit multiplier is simulated using Xilinx ISE 8.2 simulator and the output was verified and also implemented in FPGA kit.

2X4 DECODER







X	Y	F0	F1	F2	F3
0	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

EX NO: 3 SIMULATION AND IMPLEMENTATION OF 2X4

DATE: DECODER

AIM:

To design and simulate the 2x4 decoder using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

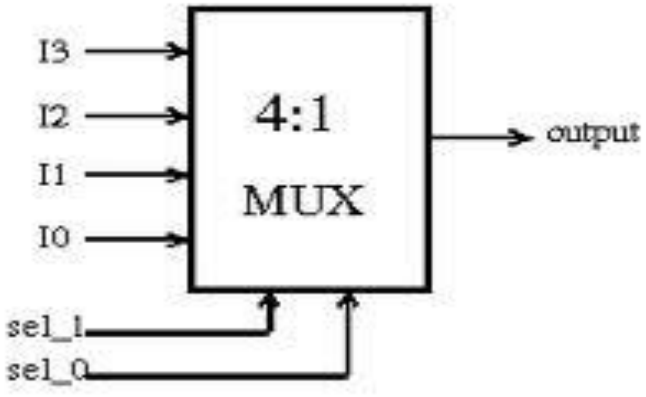
PROGRAM

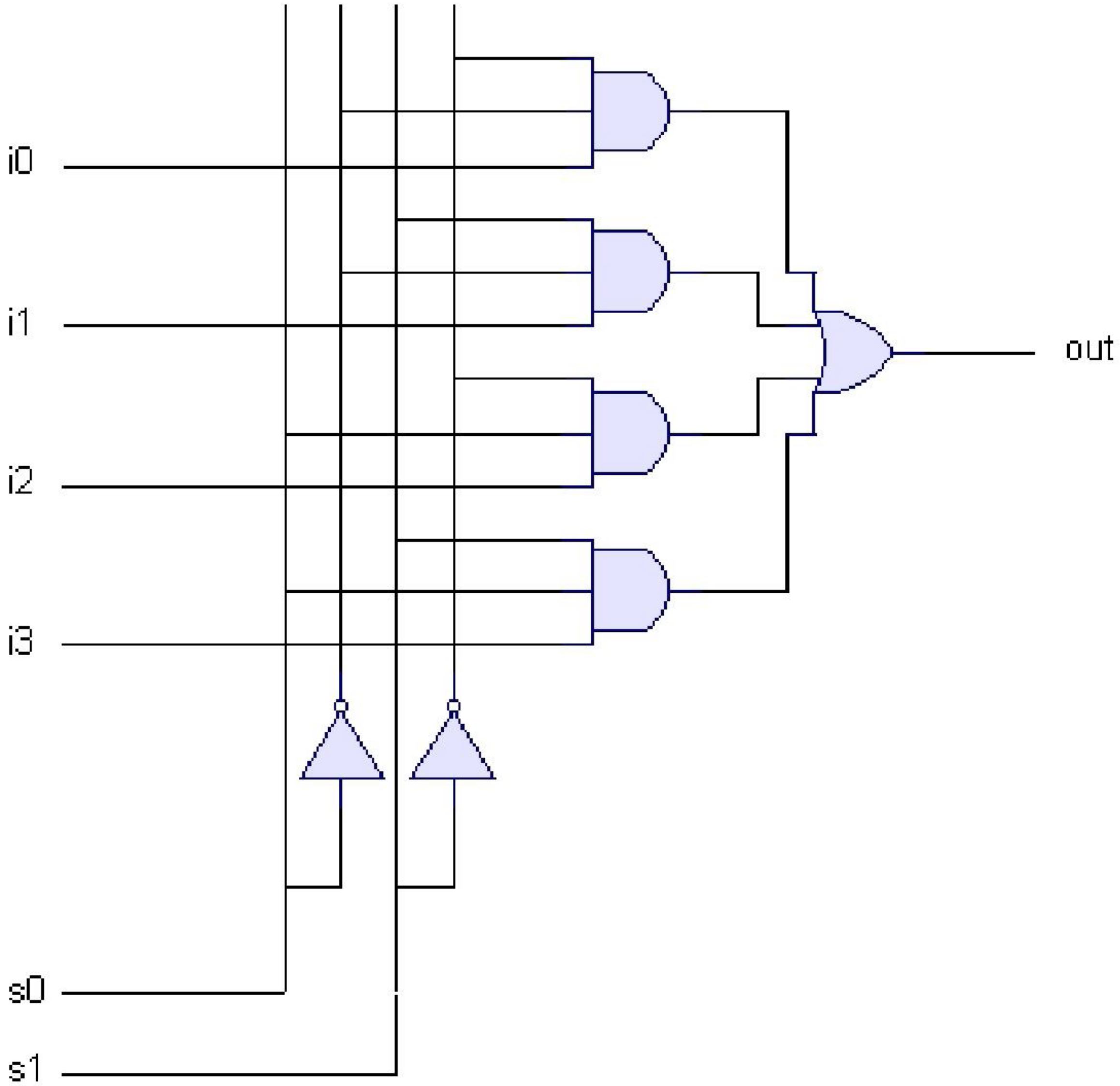
```
module Decd2to4(i0, i1, out0, out1, out2, out3);
input i0, i1;
output out0, out1, out2, out3;
reg
out0,out1,out2,out3;
        always@(i0,i1)
                case({i0,i1})
                2'b00:
                {out0,out1,out2,out3}=4'b1000;
                2'b01:
                {out0,out1,out2,out3}=4'b0100;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

TESH BENCH:

```
module Stimulus_v;
// Inputs
reg i0, i1;
// Outputs
wire out0, out1, out2, out3;
// Instantiate the Unit Under Test (UUT)
Decd2to4 uut (.i0(i0),.i1(i1),.out0(out0),.out1(out1),.out2(out2),.out3(out3));
initial
begin
$display("\t\t 2to4 Decoder");
$display("\t\t ---------------------------");
$display("\t\t Input \t\t Output");
$display("\t\t ---------------------------");
$monitor("\t\t %b%b\t\t\t %b%b%b %b",i0,i1,out0,out1,out2,out3);
#4 $display("\t\t --------------------------");
end
initial
begin
i0=0;i1=0;
#1 i0=0;i1=1;
#1 i0=1;i1=0;
#1 i0=1;i1=1;
#1 $stop;
end
endmodule
```

4x1 MULTIPLEXER:





EX NO: 4 SIMULATION AND IMPLEMENTATION OF 4x1 MULTIPLEXER

DATE:

AIM:

To design and simulate the 4x1 multiplexer using VerilogHDL code Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

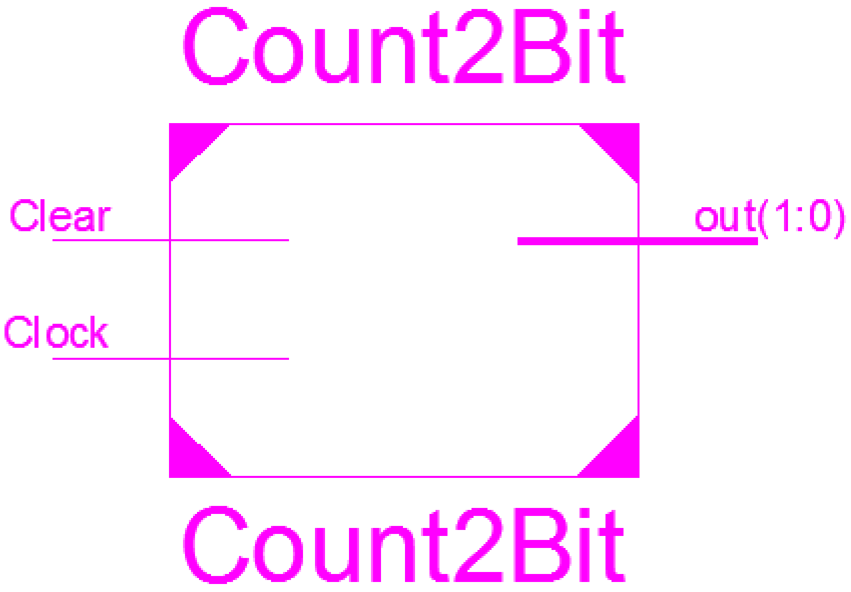
PROGRAM:

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

TESH BENCH:

```
module Stimulus_v;
// Inputs
reg i0, i1, i2,,i3, s0, s1; //
Outputs
wire out;
Instantiate the Unit Under Test (UUT)
Mux4to1 uut (.i0(i0),.i1(i1),.i2(i2),.i3(i3),.s0(s0),.s1(s1),.out(out));
initial
begin
$display("\t\t\t4to1 Multiplexer");
$display("\t\t-------------------------");
#1 $display("\t\t\t Input=%b%b%b%b",i0,i1,i2,i3);
$display("\t\t----------------------------------");
$display("\t\tSelector \t\t\t\t Output");
$display("\t\t----------------------------------");
$monitor("\t\t{%b,%b}\t\t\t\t%b",s0,s1,out);
#4 $display("\t\t------------------------------------------------");
end
initial begin
i0=1; i1=0; i2=1; i3=1; #1
s0=0; s1=0;
#1 s0=1; s1=0;
#1 s0=0; s1=1;
#1 s0=1; s1=1;
#1 $stop;
end
Endmodule
```

COUNTER:



EX NO: 5 SIMULATION AND IMPLEMENTATION OF 2- BIT COUNTER

DATE:

AIM:

To design and simulate the 2x4 decoder using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

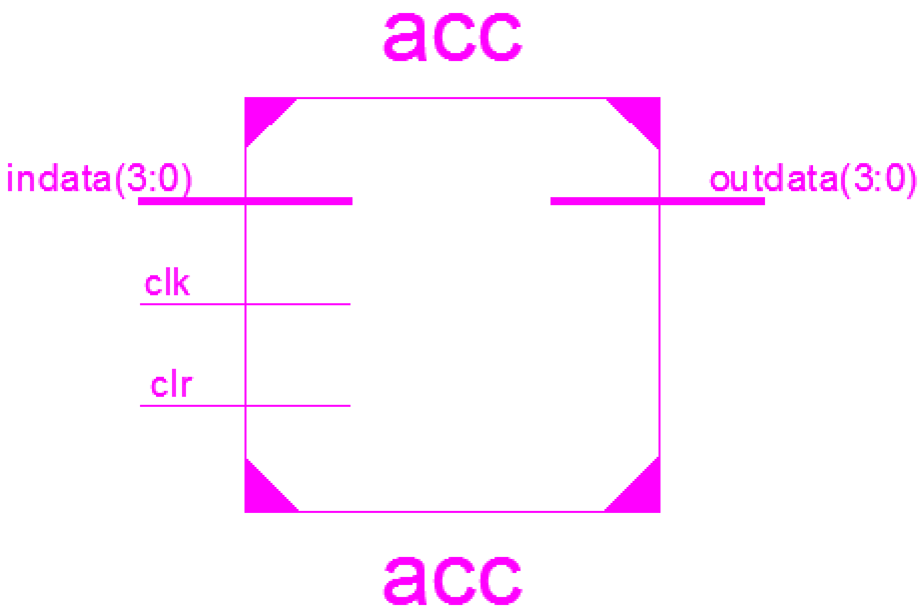
PROGRAM:

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

TESH BENCH:

```
module Stimulus_v;
// Inputs
reg Clock, Clear; //
Outputs
wire [1:0] out;
// Instantiate the Unit Under Test (UUT)
Count2Bit uut (.Clock(Clock),.Clear(Clear),.out(out));
initial
begin
$display("\t\t\t 2 Bit Counter");
$display("\t\t------------------------------------------");
$display("\t\tClock\t\tClear\t\tOutput[2]");
$display("\t\t------------------------------------------");
$monitor("\t\t %b\t\t %b \t\t %b ",Clock,Clear,out);
#28 $display("\t\t------------------------------------------");
end
always
#1 Clock=~Clock;
initial
begin
Clock=0; Clear=0;
#10 Clear=1;
#18 Clear=0;
#2 $stop;
end
endmodule
```

ACCUMULATOR:



EX NO: 6 SIMULATION AND IMPLEMENTATION OF

DATE: ACCUMULATOR

AIM:

To design and simulate the accumulator using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

PROGRAM:

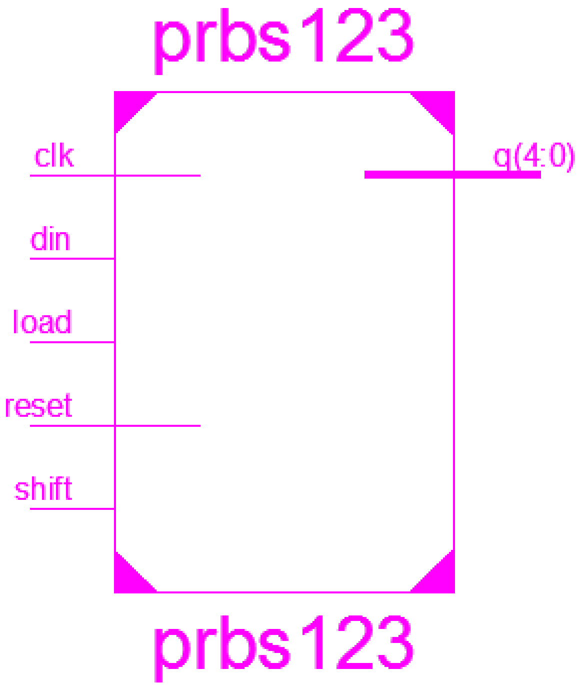
```
module acc(indata, clk,clr, outdata);
input [3:0] indata;
input clk,clr;
output [3:0] outdata;
reg [3:0] outdata;
always@(posedge clk or posedge clr)
begin
if(clr)
outdata <= 4'd0;
else
outdata <= indata;
end
endmodule
```

STIMULUS

```
module testbenchacc;
//input [3:0] indata;
//input clk,clr;
reg [3:0] indata;
reg clk,clr;

//output [7:0] outdata;
wire [3:0] outdata;
//Instantiate unit under test (uut)
acc uut (.indata(indata),.clk(clk),.clr(clr),.outdata(outdata));
initial
begin
//Initialize Inputs
indata = 4'd0;
clk = 1'b0;
clr = 1'b1;
#50; indata = 4'd4; clk = 1'b1; clr = 1'b0;
#50; indata = 4'd5; clk = 1'b0; clr = 1'b0;
#50; indata = 4'd3; clk = 1'b1; clr = 1'b0;
#50; indata = 4'd3; clk = 1'b0; clr = 1'b0;
#50; indata = 4'd1; clk = 1'b1; clr = 1'b0;
#50; indata = 4'd7; clk = 1'b0; clr = 1'b0;
#50; indata = 4'd1; clk = 1'b1; clr = 1'b0;
#50; indata = 4'd2; clk = 1'b0; clr = 1'b1;
#50; indata = 4'd9; clk = 1'b1; clr = 1'b0;
end
endmodule
```

PRBS GENERATOR :



EX NO: 7 SIMULATION AND IMPLEMENTATION OF
DATE: PRBS GENERATOR

AIM:

To design and simulate the PRBS GENERATOR using VerilogHDL code in Xilinx ISE design suite 8.2and also implement in FPGA kit.

SOFTWARE REQUIRED:

Xilinx ISE design suite 8.2, FPGA kit.

PROCEDURE:

1. Start the Xilinx project navigator

2.Create a new source.

3.Create a new verilog file assign required.

4. Write verilog code in test verilog editor simulator,create the test bench file.

5. Synthesize the design.

6. Assign the package pins for input and output port and verify the output.

PROGRAM:

```
module prbs123( clk,reset,din,load,shift,q );
input clk,reset;
input din;
input load;
input shift;
output[4:0]q;
parameter N=5;
parameter tap0=0;
parameter tap1=2;
parameter tap2=3;
parameter tap3=4;
reg[N-1:0]q;
reg[N-1:0]q_N;
wire TAPS;
```

```
always@(posedge clk or posedge reset)
begin
if(reset)
q<=0-1;
else
q<=q-N;
end
assign TAPS=q[tap0]^q[tap1]^q[tap2]^q[tap3];
always @(q or shift or load or din or TAPS)
begin
q_N=q;
if(load)
q_N={din,q[N-1:1]};
else
if(shift)
q_N={TAPS,q[N-1:1]};
end
endmodule
```

TEST BENCH:

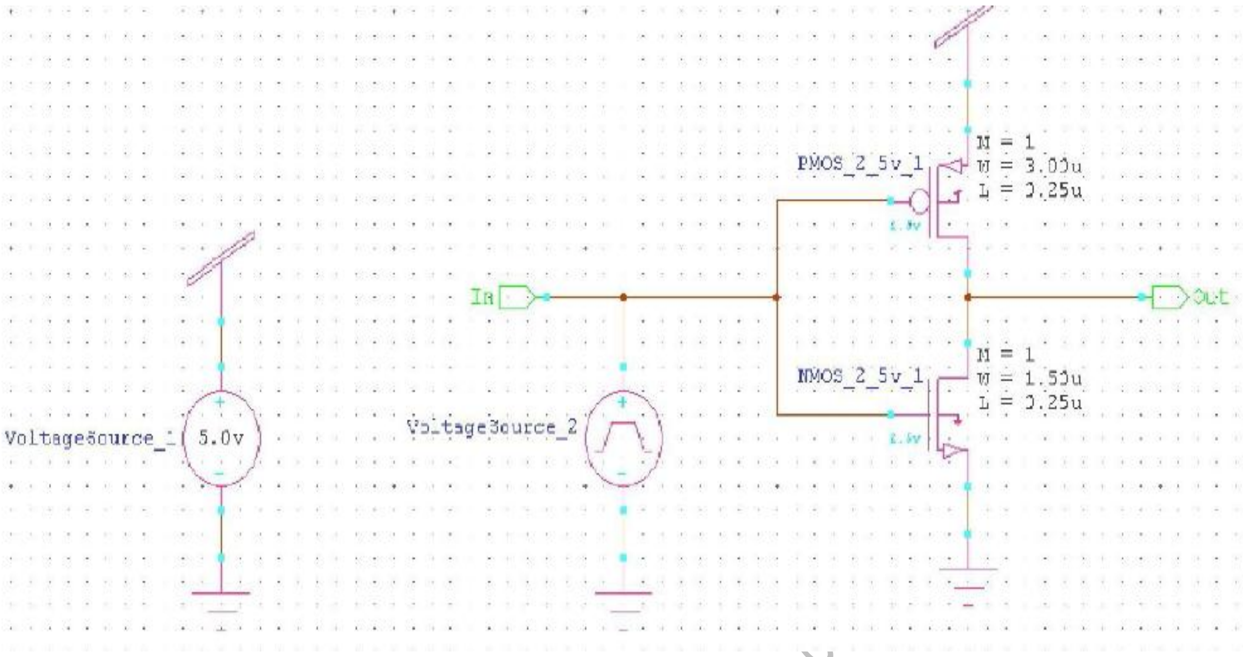
```
module prbs123( clk,reset,din,load,shift,q );
input clk,reset;
input din;
input load;
input shift;
output[4:0]q;
parameter N=5;
parameter tap0=0;
parameter tap1=2;
parameter tap2=3;
parameter tap3=4;
reg[N-1:0]q;
reg[N-1:0]q_N;
wire TAPS;
always@(posedge clk or posedge reset)
begin
if(reset)
q<=0-1;
else
```

```
q<=q-N;
end
assign TAPS=q[tap0]^q[tap1]^q[tap2]^q[tap3];
always @(q or shift or load or din or TAPS)
begin
q_N=q;
if(load)
q_N={din,q[N-1:1]};
else
if(shift)
q_N={TAPS,q[N-1:1]};
end
Endmodule
```

RESULT:

Thus the PRBS GENERATOR is simulated using Xilinx ISE 8.2 simulator and the output was verified and also implemented in FPGA kit.

CMOS INVERTER:



EX NO: 8 SIMULATION OF CMOS INVERTER

DATE:

AIM:

To simulate CMOS inverter using D-Schematic

APPARATUS REQUIRED:

D-Schematic

PROCEDURE :

1. Open D-Schematic software.

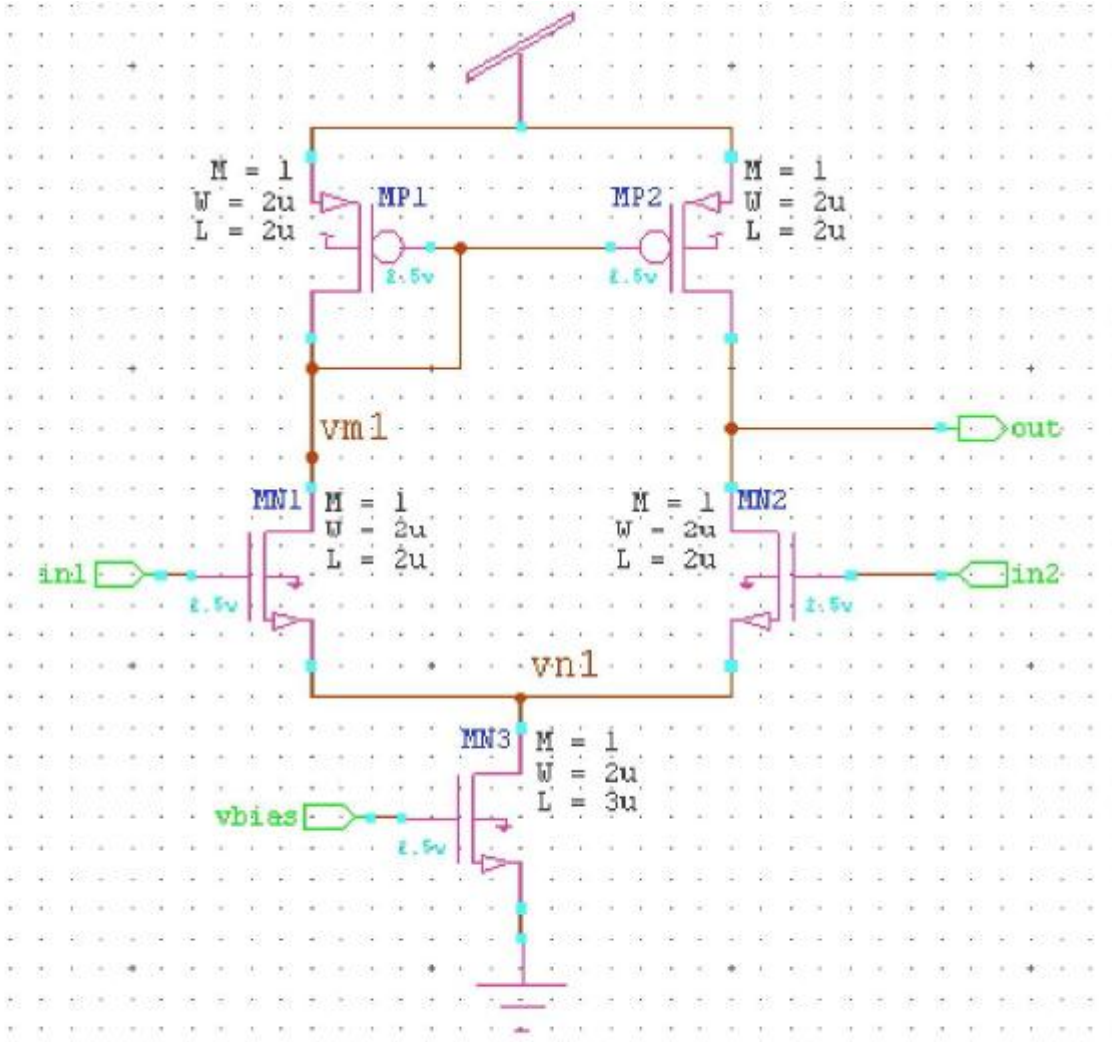
2. Create a new file.

3. Draw the schematic of inverter

4.Click port on the tool bar and draw OUT,IN ports.

5.Launch run to simulation.

DIFFERENTIAL AMPLIFIER :



EX NO: 9 SIMULATION OF MOS DIFFERENTIAL AMPLIFIER

DATE:

AIM:

To simulate MOS differential amplifier using D-Schematic.

APPARATUS REQUIRED:

D-Schematic

PROCEDURE:

1. Open D-Schematic software.

2. Create a new file.

3. Draw the schematic of Differential amplifier

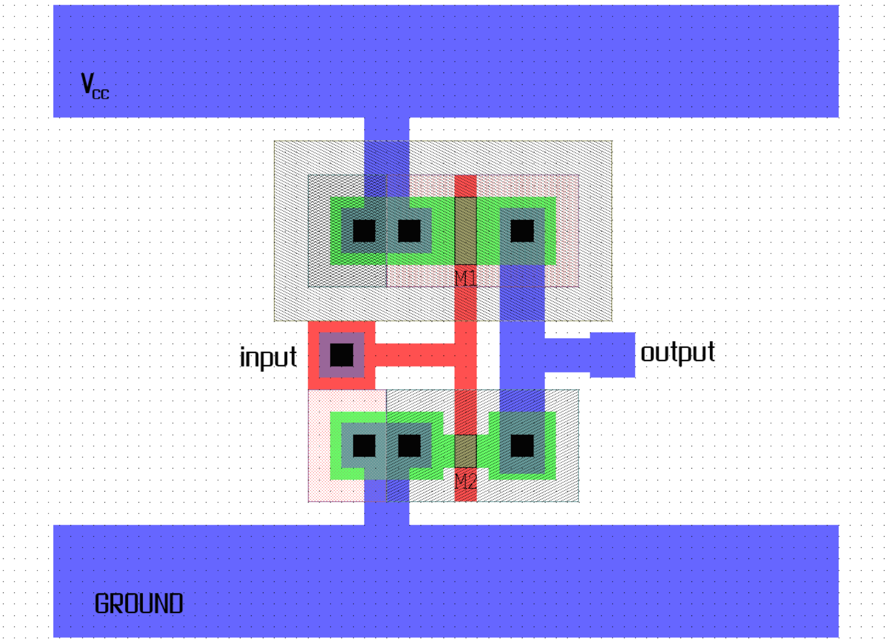
4.Click port on the tool bar and draw OUT,IN ports.

5.Launch run to simulation.

RESULT :

Thus the simulation of MOS differential amplifier using D-Schematic is performed and the output was verified.

CMOS INVERTER:



EX NO:10 SIMULATION OF SIMPLE CMOS INVERTER

DATE:

AIM:

To design a layout and simulation of simple CMOS inverter.

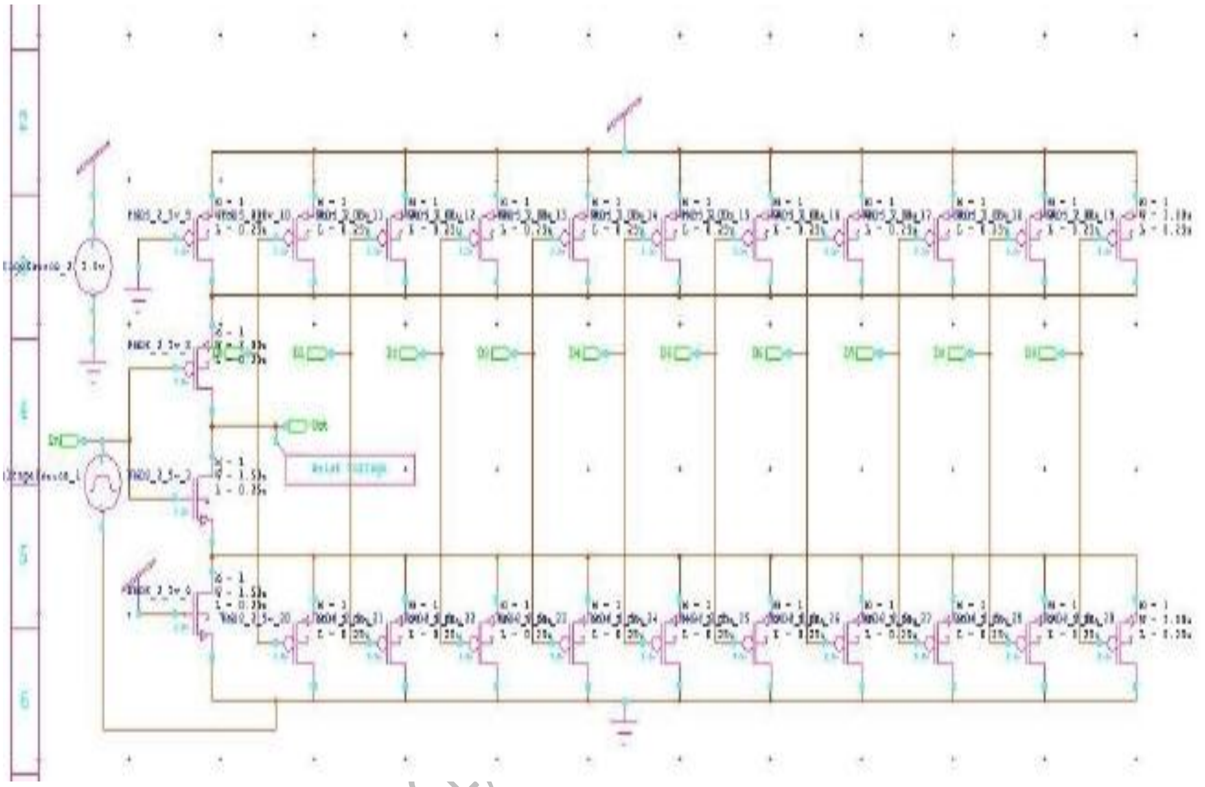
SOFTWARE REQUIRED:

Microwind

PROCEDURE:

1. Open Microwind software
2. Create a new layout file.
3. Draw the layout of inverter
4. Click port ON the toolbar and draw OUT,VDD,GND port
5. Check for design rule violation at each stage.
6. If no DRC error,extract the file spice file and simulate.

10 BIT NUMBER CONTROLLED OSCILLATOR:



EX NO:11 DESIGN AND SIMULATION OF 10 BIT NUMBER

DATE: CONTROLLED OSCILLATOR

AIM:

To design and simulate 10 BIT number controller oscillator using D-Schematic

APPARATUS REQUIRED:

D-Schematic

PROCEDURE :

1. Open D-Schematic software.

2. Create a new file.

3. Draw the circuit diagram of oscillator controller

4.Click port on the tool bar and draw OUT,IN ports.

5.Launch run to simulation.

RESULT :

Thus design the simulation of 10 bit number controlled Oscillator is performed and the output was verified.