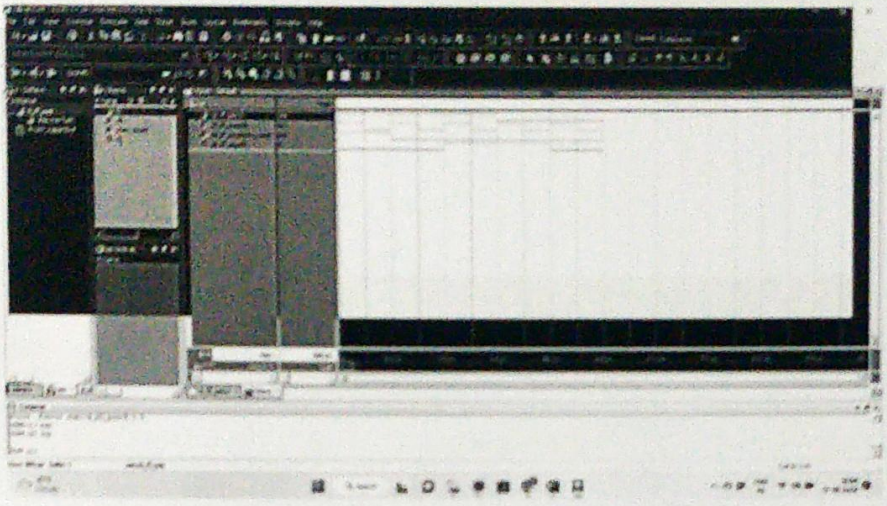
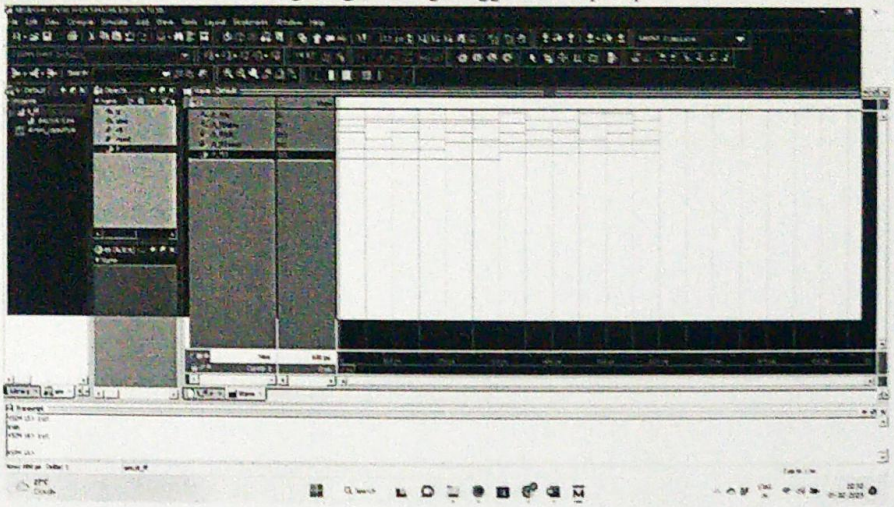
1.1) Use If statement to design positive edge triggered D flip



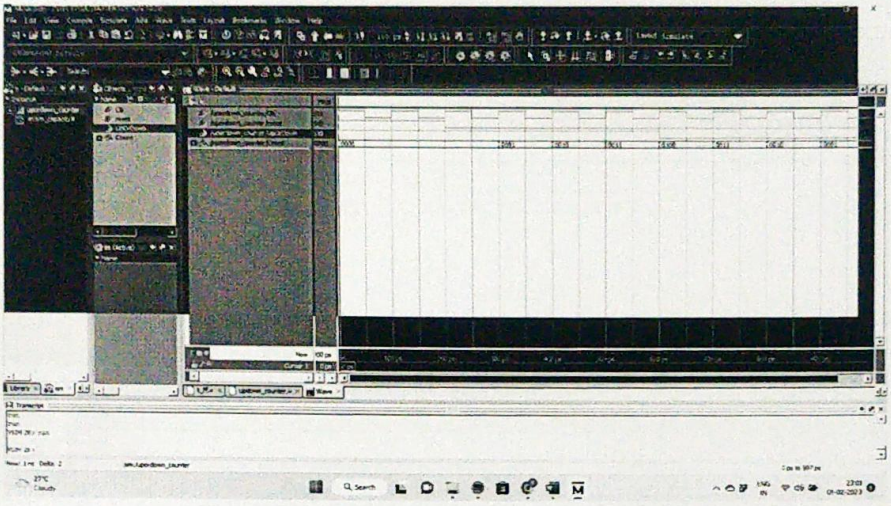
```
module
RisingEdge_DFlipFlop_SyncReset(D,clk,sync_reset,Q);
input D;
input clk;
input sync_reset;
output reg Q; //
always @(posedge clk)
begin
 if(sync_reset==1'b1)
  Q <= 1'b0;
 else
  Q <= D;
end
endmodule
```

1.2) Use case statement to design negative edge triggered T flip flop



```
module t_ff(q,qbar,clk,reset,t);

output reg q,qbar;

input clk,reset,t;

always @(negedge clk,reset,t)

begin

case({reset,t})

2'b00:q<=1'b0;

2'b01:q<=1'b0;

2'b10:q<=q;

2'b11:q<=~q;

endcase

qbar<=~q;

end

endmodule
```

2.1)



```
module upordown_counter(
  Clk,
  reset,
  UpOrDown,  //high for UP counter and low for Down counter
  Count
  );


  //input ports and their sizes
  input Clk,reset,UpOrDown;
  //output ports and their size
  output [3 : 0] Count;
  //Internal variables
  reg [3 : 0] Count = 0;

  always @(posedge(Clk) or posedge(reset))
  begin
    if(reset == 1)
      Count <= 0;
    else
      if(UpOrDown == 1)  //Up mode selected
        if(Count == 15)
          Count <= 0;
        else
          Count <= Count + 1; //Incremend Counter
      else  //Down mode selected
        if(Count == 0)
          Count <= 15;
        else
          Count <= Count - 1; //Decrement counter
  end
```

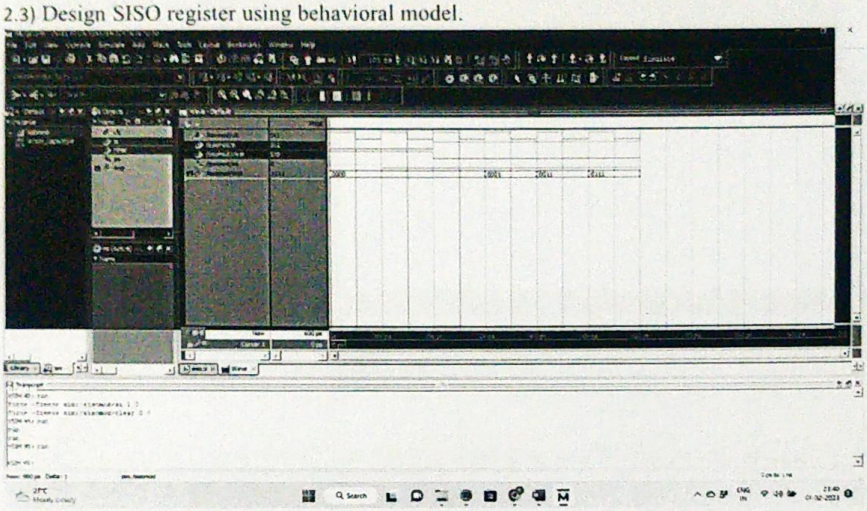
51

o/p verified

14/2/23

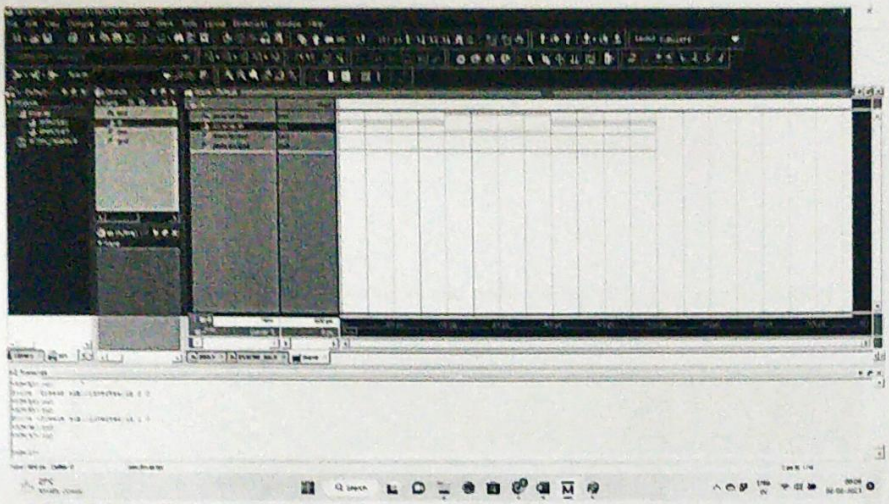
endmodule

2.3) Design SISO register using behavioral model.



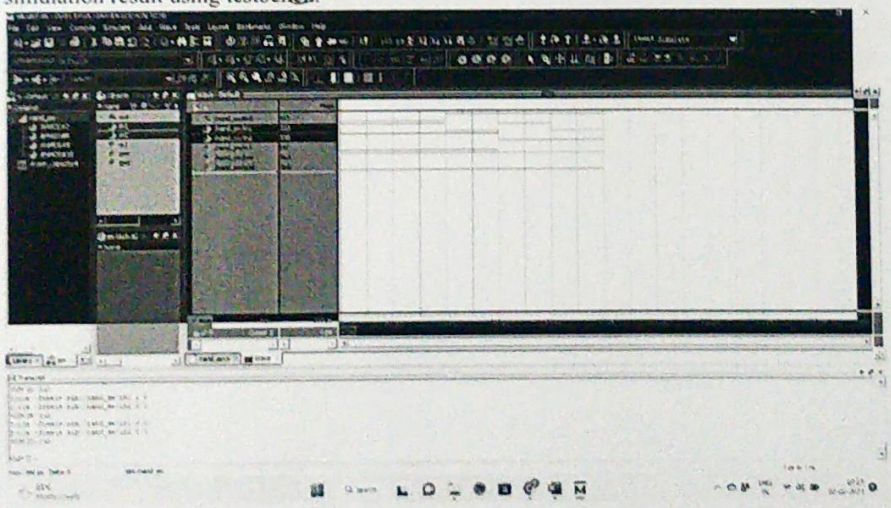
```
module sisomod(clk,clear,si,so);

input clk,si,clear;

output so;

reg so;

reg [3:0] tmp;

always @(posedge clk )

begin

if (clear)

tmp <= 4'b0000;

else

tmp <= tmp << 1;

tmp[0] <= si;

so = tmp[3];

end

endmodule
```

3.1) Design inverter logic using verilog switch level modeling and verify the simulation result using test bench



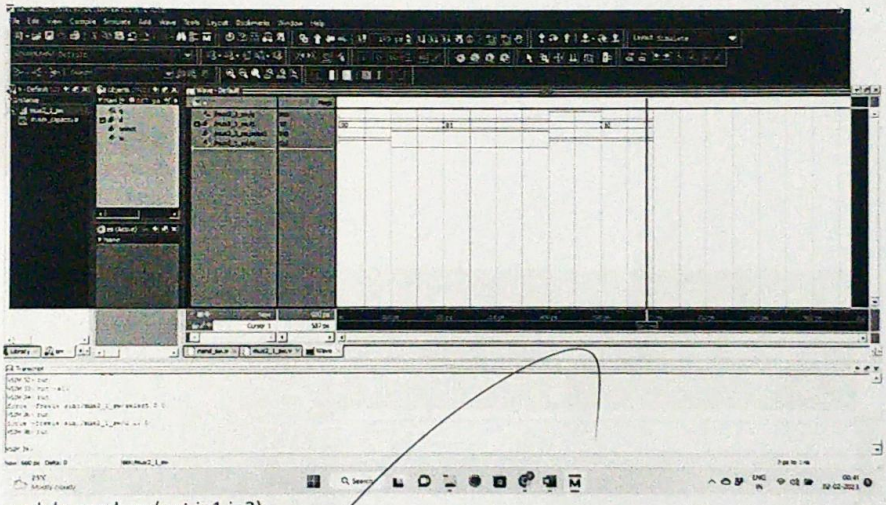
```
module inverter(out,in);
output out;
input in;
supply1 pwr;
supply0 gnd;
pmos(out,pwr,in);
nmos(out,gnd,in);
endmodule
```

3.2) Design two input CMOS NAND logic using verilog switch level modeling and verify the simulation result using testbench.



```
module nand_sw(out,in1,in2);
output out;
input in1,in2;
wire w1;
supply1 pwr;
supply0 gnd;
pmos(out,pwr,in1);
pmos(out,pwr,in2);
nmos(out,w1,in1);
nmos(w1,gnd,in2);
endmodule
```

3.3) Design 2:1 Mux using CMOS switches and write verilog coding using switch level modeling and verify the simulation result.



```
module nand_sw(out,in1,in2);
output out;
input in1,in2;
wire w1;
supply1 pwr;
supply0 gnd;
pmos(out,pwr,in1);
pmos(out,pwr,in2);
nmos(out,w1,in1);
nmos(w1,gnd,in2);
endmodule
```

Post lab

1) module sipo(clk,clear, si, po);

input clk, si,clear;

output [3:0] po;

reg [3:0] tmp;

reg [3:0] po;

always @(posedge clk)

begin

if (clear)

tmp <= 4'b0000;

else

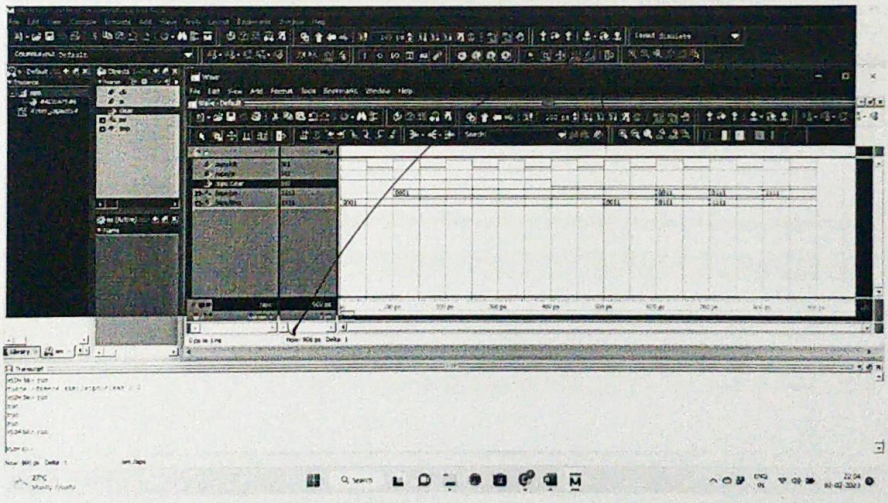
tmp <= tmp << 1;

tmp[0] <= si;

po = tmp;

end

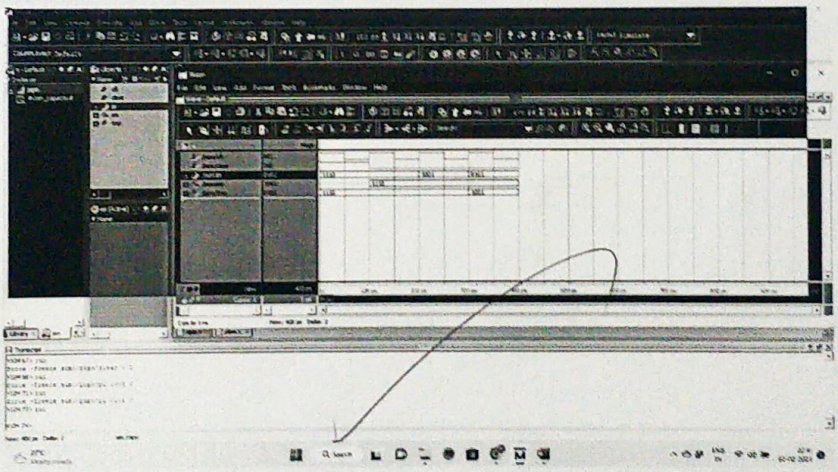
endmodule



2) module pipo(clk,clear, pi, po);

input clk,clear;

```
input [3:0]pi;
output [3:0] po;
reg [3:0] tmp;
reg [3:0] po;
always @(posedge clk)
begin
if (clear)
tmp <= 4'b0000;
else
tmp <= tmp << 1;
tmp <= pi;
po = tmp;
end
endmodule
```



VLSI Lab

Experiment - 2

I Pre-Lab Questions:

1. Write the difference b/w initial and always blocks

Soln. Initial:

(i) begin - imperative statement end
(ii) Runs from simulation starts
(iii) Good for providing stimulus

Always:

(i) begin - imperative statement end
(ii) Runs when simulation starts
(iii) Good for providing, specific hardware and modelling

2. List the reduction and logical operation

Soln. Reduction operators

& : AND
| : OR
~& : NAND
~| : NOR
^ : XOR

Logical operators

&& : AND
|| : OR
! : NOT

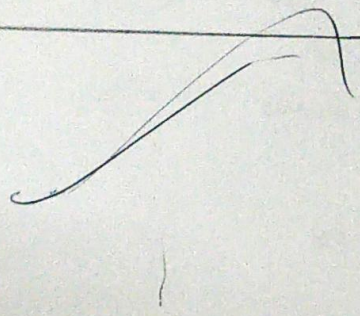
3. Give the use of blocking and non-blocking statements

Soln. Blocking statement will not block the execution of statement that are in parallel block, while non-blocking statement allow scheduling of assignment that are executed in sequential block.

4. Differentiate case, casex and casez statements

Soln. Case statement Consider x or z as it is so a

Case expression contains x or z at corresponding its positions if no case item matches then default item is executed.



II Post-Lab Questions

1 Write Verilog HDL Code to implement a SIPO and PIPO Shift registers

Soln PIPO Shift Register

```
module PIPO_register (clk, rst, a, q);
    input clk,
    input rst?
    input [3:0] a;
    output [3:0] q;
    reg [3:0] q;
    always @ (posedge clk, posedge rst)
    begin if (rst == 1'b1)
     q <= 4'b0000;
     else
     q <= a; end
     endmodule
```

SIPO Shift Register

```
Module SIPO_ register (a, clk, rst, q);
    input a, clk, rst;
    output [3:0] q;
    wire [3:0] q;
    reg [3:0] temp;
    always @ (posedge clk, posedge rst)
     begin if (rst == 1'b1)
     temp <= 4'b000; else
     begin
     temp <= temp << 1'b1;
```

temp [0] <= a; end
end assign q = temp;
endmodule.

RESULT

Realization of Digital Circuits using behavioral and Switch level modeling using Modelsim is done and output is verified.

14/2/23