VERILOG LAB1

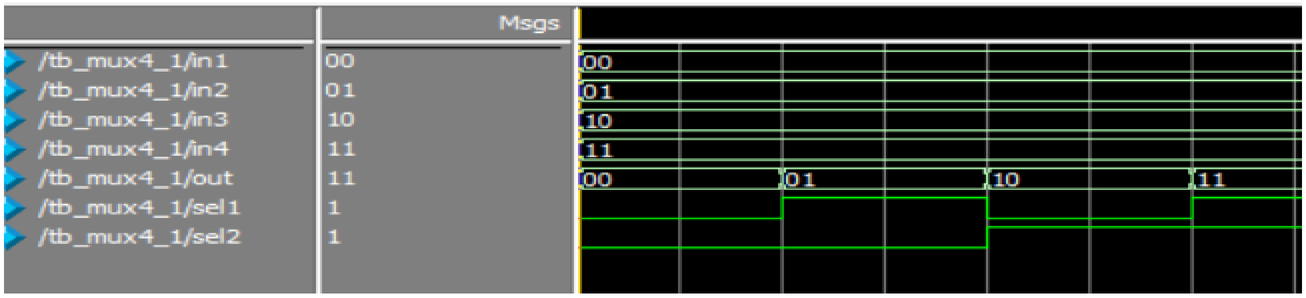
DIGITAL VERIFICATION

NAME	**OMNIA MOHAMED SELIM**

Table of Contents

Mux4-1 using mux2-1

Simulation results



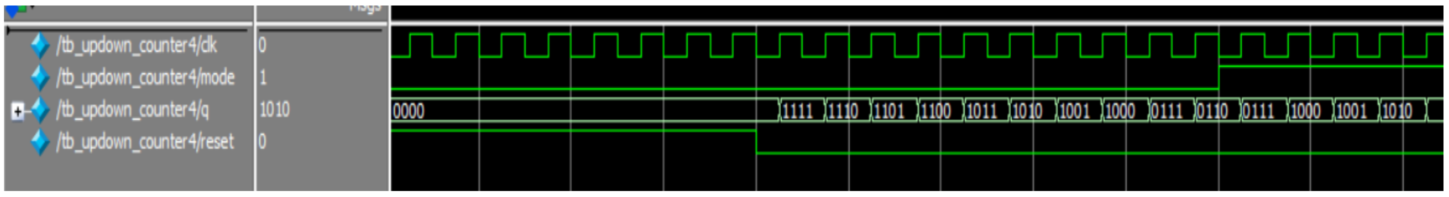
```
VSIM 126> run
# in1=00,in2=01,in3=10,in4=11 ,sel1=0,sel2=0 , out=00
# in1=00,in2=01,in3=10,in4=11 ,sel1=1,sel2=0 , out=01
# in1=00,in2=01,in3=10,in4=11 ,sel1=0,sel2=1 , out=10
# in1=00,in2=01,in3=10,in4=11 ,sel1=1,sel2=1 , out=11
```

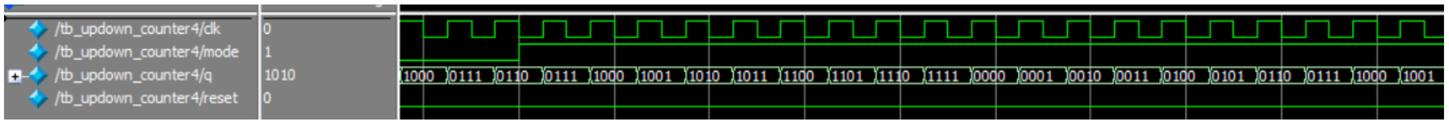
Up-down 4-bit counter

Design code

```
module updown_counter4(input mode,clk,reset,output reg [3:0] q);

always@(posedge clk) begin
 if(reset==1) q<=0;
 else begin
  if(mode==0) begin //down counter
   if(q !=0) q<=q-1;
   else q<=4'b1111;
  end
  else begin
  if  (q!=4'b1111) q<=q+1;
  else q<=0;
 end
 end
end
endmodule
```

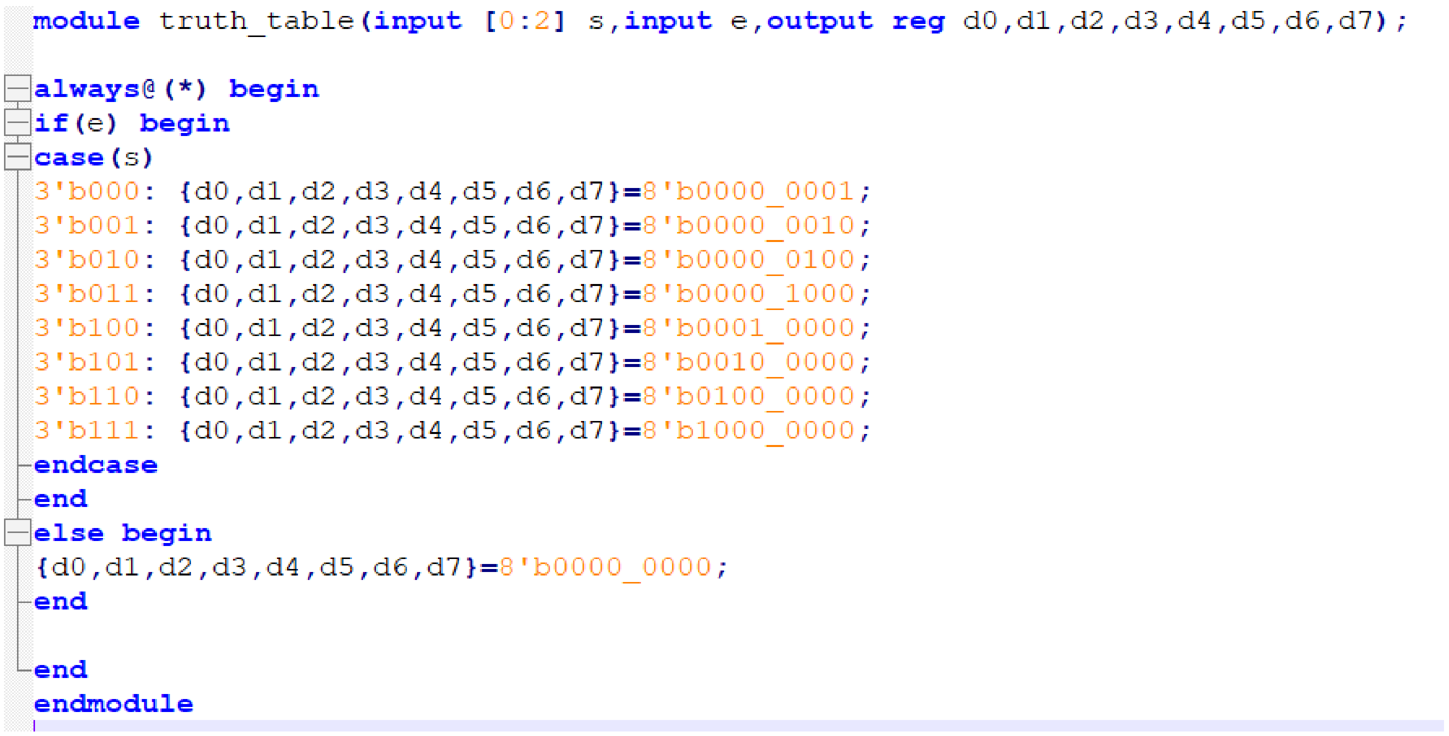
Simulation results





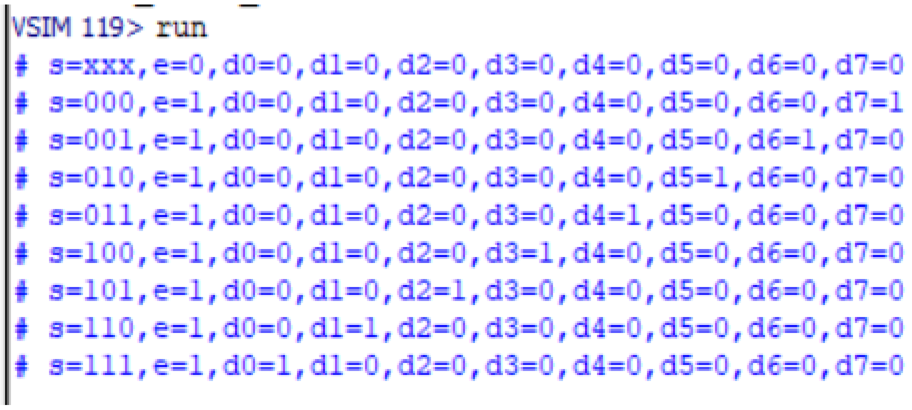
Example3

Design code

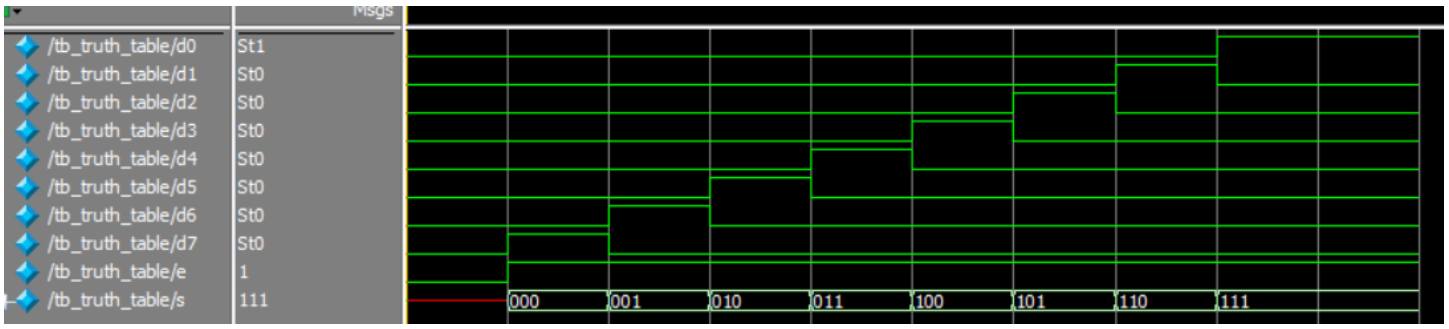


```
module truth_table(input [0:2] s,input e,output reg d0,d1,d2,d3,d4,d5,d6,d7);

always@(*) begin
if(e) begin
case(s)
3'b000: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0000_0001;
3'b001: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0000_0010;
3'b010: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0000_0100;
3'b011: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0000_1000;
3'b100: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0001_0000;
3'b101: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0010_0000;
3'b110: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b0100_0000;
3'b111: {d0,d1,d2,d3,d4,d5,d6,d7}=8'b1000_0000;
endcase
end
else begin
{d0,d1,d2,d3,d4,d5,d6,d7}=8'b0000_0000;
end

end
endmodule
```

Simulation results



```
VSIM 119> run
# s=xxx,e=0,d0=0,d1=0,d2=0,d3=0,d4=0,d5=0,d6=0,d7=0
# s=000,e=1,d0=0,d1=0,d2=0,d3=0,d4=0,d5=0,d6=0,d7=1
# s=001,e=1,d0=0,d1=0,d2=0,d3=0,d4=0,d5=0,d6=1,d7=0
# s=010,e=1,d0=0,d1=0,d2=0,d3=0,d4=0,d5=1,d6=0,d7=0
# s=011,e=1,d0=0,d1=0,d2=0,d3=0,d4=1,d5=0,d6=0,d7=0
# s=100,e=1,d0=0,d1=0,d2=0,d3=1,d4=0,d5=0,d6=0,d7=0
# s=101,e=1,d0=0,d1=0,d2=1,d3=0,d4=0,d5=0,d6=0,d7=0
# s=110,e=1,d0=0,d1=1,d2=0,d3=0,d4=0,d5=0,d6=0,d7=0
# s=111,e=1,d0=1,d1=0,d2=0,d3=0,d4=0,d5=0,d6=0,d7=0
```

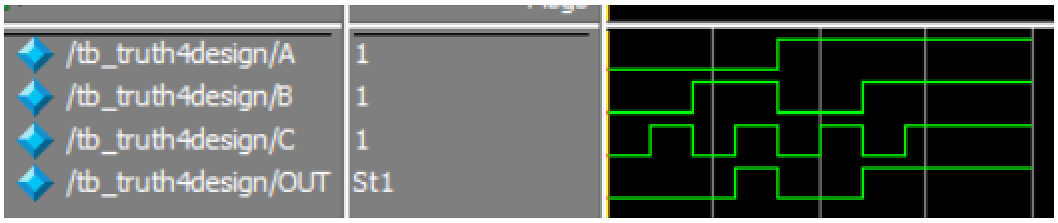


Example4: design logic circuit for the truth table

Design code

```
module truth4design (input A,B,C,output OUT);
assign OUT=(A&B) | (C&B);

endmodule
```

Simulation results



```
/tb_truth4design/A     1
/tb_truth4design/B     1
/tb_truth4design/C     1
/tb_truth4design/OUT   St1
```

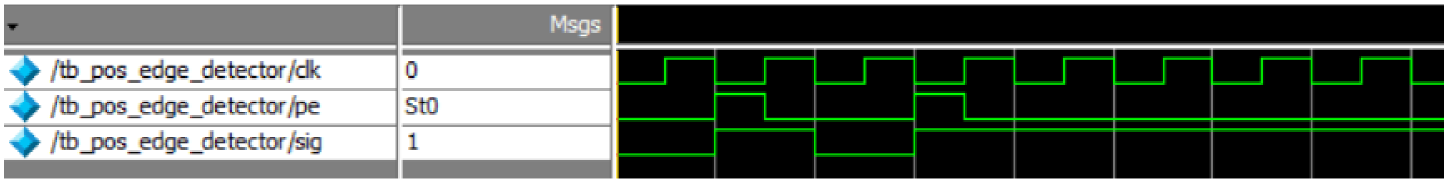
```
# A=0,B=0,C=0,OUT=0
# A=0,B=0,C=1,OUT=0
# A=0,B=1,C=0,OUT=0
# A=0,B=1,C=1,OUT=1
# A=1,B=0,C=0,OUT=0
# A=1,B=0,C=1,OUT=0
# A=1,B=1,C=0,OUT=1
# A=1,B=1,C=1,OUT=1
```

+ve edge detector

Design code

```
module pos_edge_detector(input clk,sig,output pe);
reg delayed_sig;
always@(posedge clk) begin
delayed_sig<=sig;
end

assign pe=sig & ~delayed_sig;
endmodule
```

Simulation results



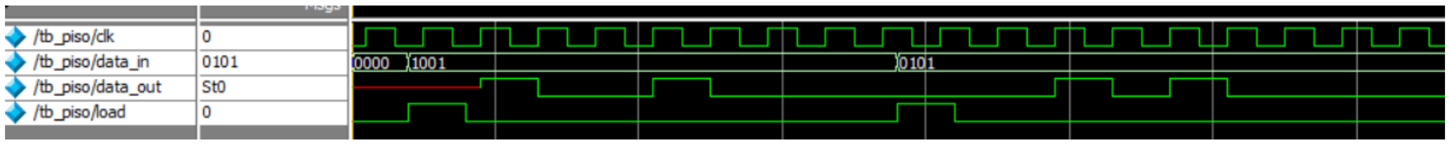
```
                          Msgs
/tb_pos_edge_detector/clk   0
/tb_pos_edge_detector/pe    St0
/tb_pos_edge_detector/sig   1
```

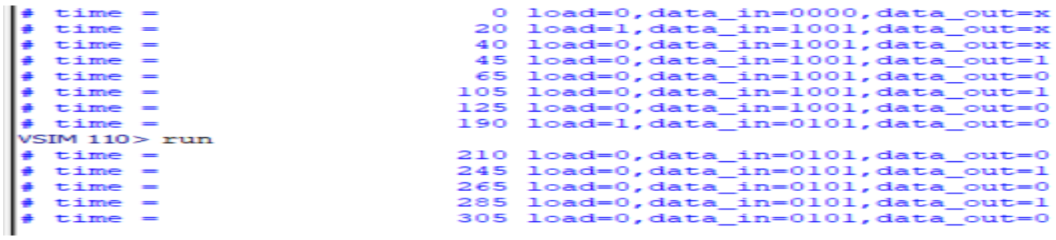
4-bit PISO

Design code

```
module piso(input clk,load,[3:0] data_in,output reg data_out);

reg [3:0] temp;
always @(posedge clk) begin
if (load) temp<=data_in;
else begin
data_out<=temp[3];
temp <= {temp[2:0],1'b0};
end
end
endmodule
```

Simulation results





```
# time =                    0 load=0,data_in=0000,data_out=x
# time =                   20 load=1,data_in=1001,data_out=x
# time =                   40 load=0,data_in=1001,data_out=x
# time =                   45 load=0,data_in=1001,data_out=1
# time =                   65 load=0,data_in=1001,data_out=0
# time =                  105 load=0,data_in=1001,data_out=1
# time =                  125 load=0,data_in=1001,data_out=0
# time =                  190 load=1,data_in=0101,data_out=0
VSIM 110> run
# time =                  210 load=0,data_in=0101,data_out=0
# time =                  245 load=0,data_in=0101,data_out=1
# time =                  265 load=0,data_in=0101,data_out=0
# time =                  285 load=0,data_in=0101,data_out=1
# time =                  305 load=0,data_in=0101,data_out=0
```

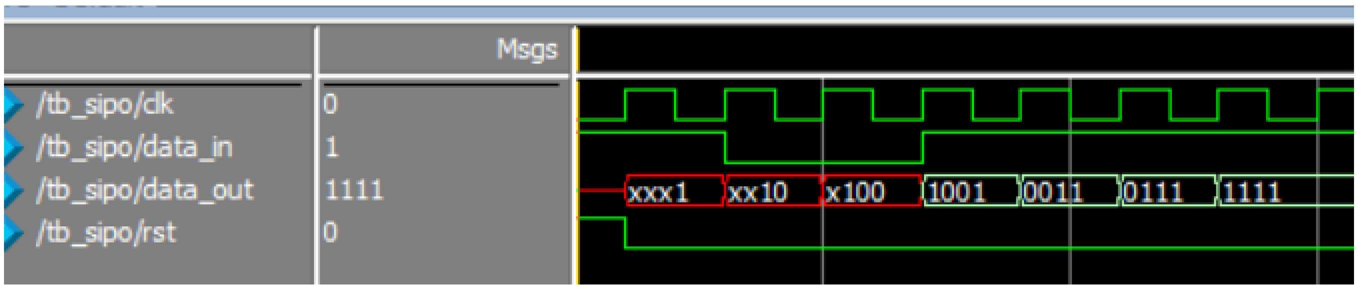
4-bit SIPO

Design code

```
module sipo(input clk,rst,data_in,output [3:0] data_out);
reg[3:0]temp;
always@(posedge clk)begin
if(rst) temp<=4'b0000;
else begin
 temp<={temp[2:0],data_in};
end
end
assign data_out=temp;


endmodule
```

Simulation results

```
# data_in=1,data_out=xxx1
# data_in=0,data_out=xx10
# data_in=0,data_out=x100
# data_in=1,data_out=1001
# data_in=1,data_out=0011
# data_in=1,data_out=0111
# data_in=1,data_out=1111
```

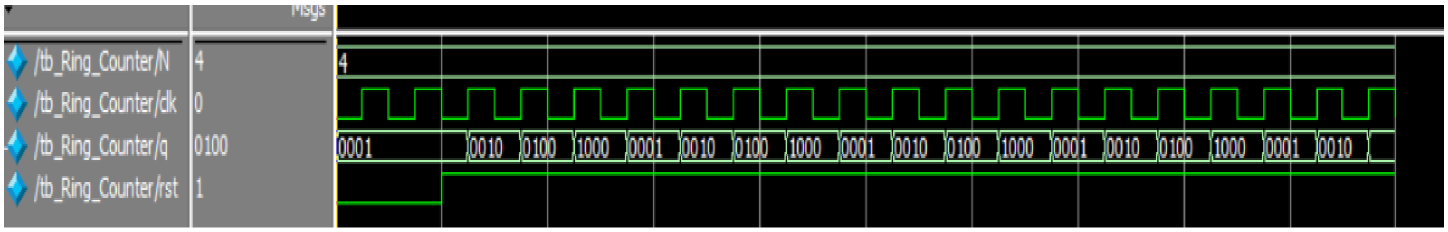


N-BIT RING COUNTER

Design code

```
module D_FF_clr (output reg q,input  clk,clr, d);
always @(posedge clk or negedge clr)
   if (~clr)
      q<= 1'b0;
   else
      q <= d;

endmodule
module D_FF_pr (output reg q,input  clk,pr, d);
always @(posedge clk or negedge pr)
   if (~pr)
      q<= 1'b1;
   else
      q <= d;

endmodule
module Ring_Counter#(parameter N = 4)(
   input  clk,rst,output  [N-1:0] q_out );

wire q1;
wire [N-1:0] d,q;

generate
   genvar i;
   for (i = 0; i < N; i = i + 1) begin
      if (i == 0) begin
        D_FF_pr u1(q[0],clk,rst,q1);
      end
      else begin
      D_FF_clr u2(q[i],clk,rst,q[i-1]);
      end
   end
endgenerate
assign q1 = q[N-1];
assign q_out = q;

endmodule
```

Simulation results

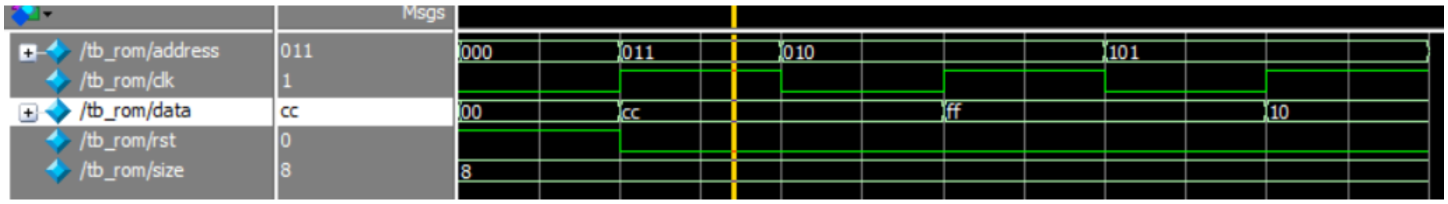


ROM

Design code

```
module rom #(parameter size=8,parameter filename="rom.txt")
(input clk,rst,[2:0]address,output reg[size-1:0] data);
reg [7:0] rom[0:size-1];
always@(posedge clk or posedge rst) begin
if(rst) begin
$readmemh(filename,rom);
data<='b0;
end
else data<=rom[address];
end
endmodule
```

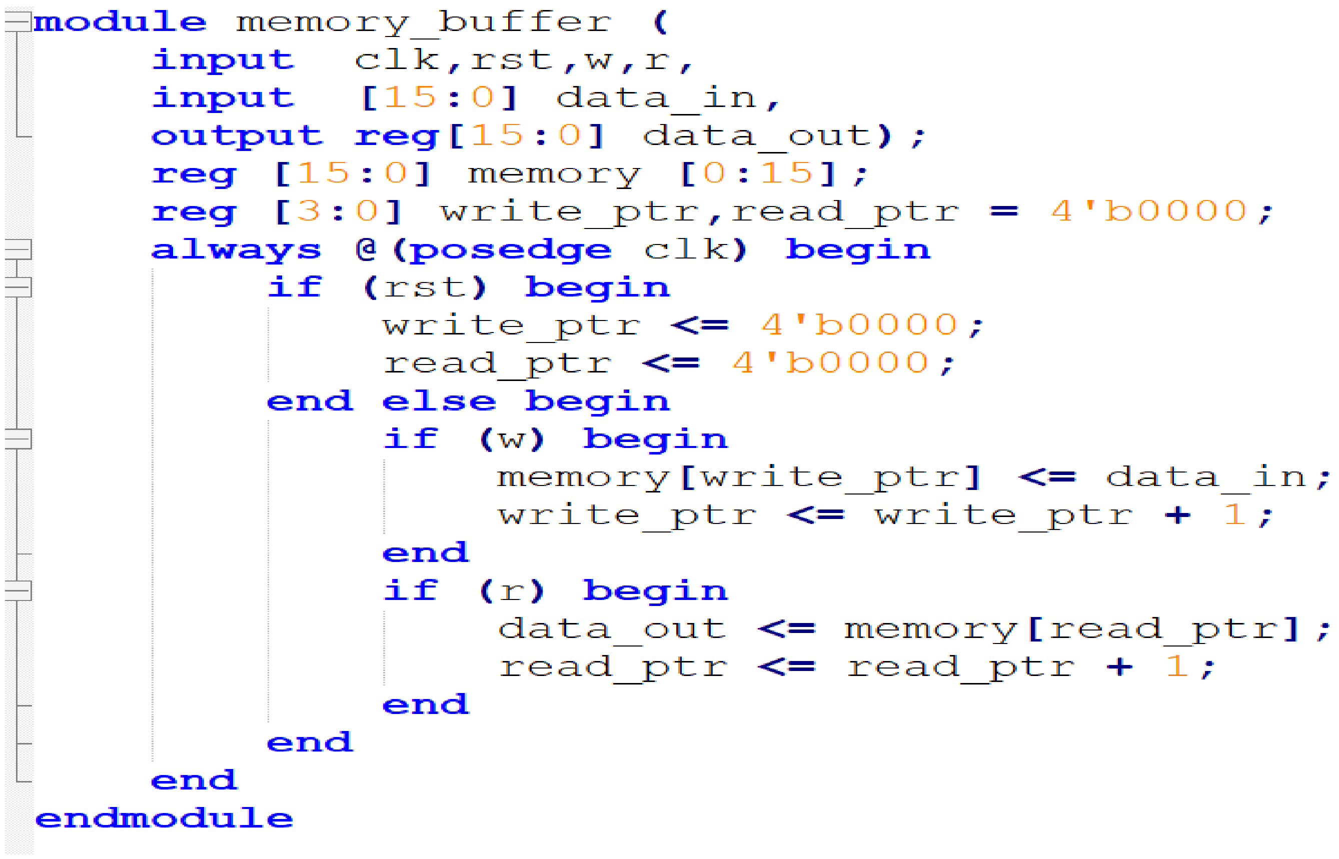
Simulation results



```
VSIM 97> run
# clk=0 ,rst=1address = 0,data = 00
# clk=1 ,rst=0address = 3,data = cc
# clk=0 ,rst=0address = 2,data = cc
# clk=1 ,rst=0address = 2,data = ff
# clk=0 ,rst=0address = 5,data = ff
# clk=1 ,rst=0address = 5,data = 10
```

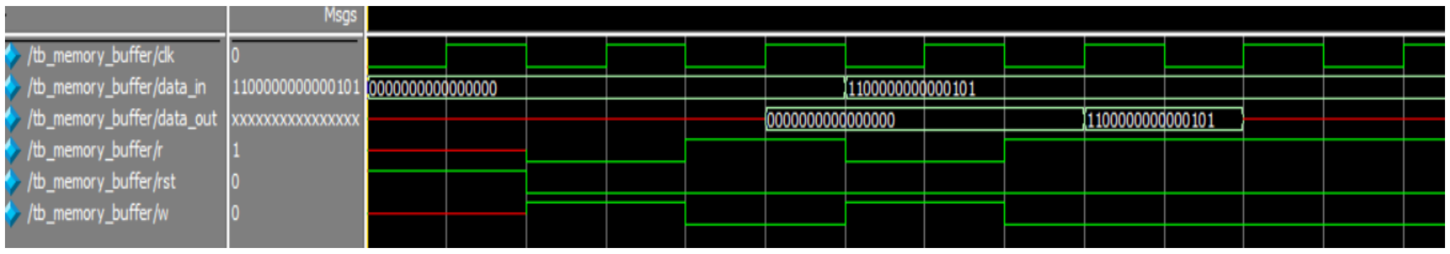
Memory buffer

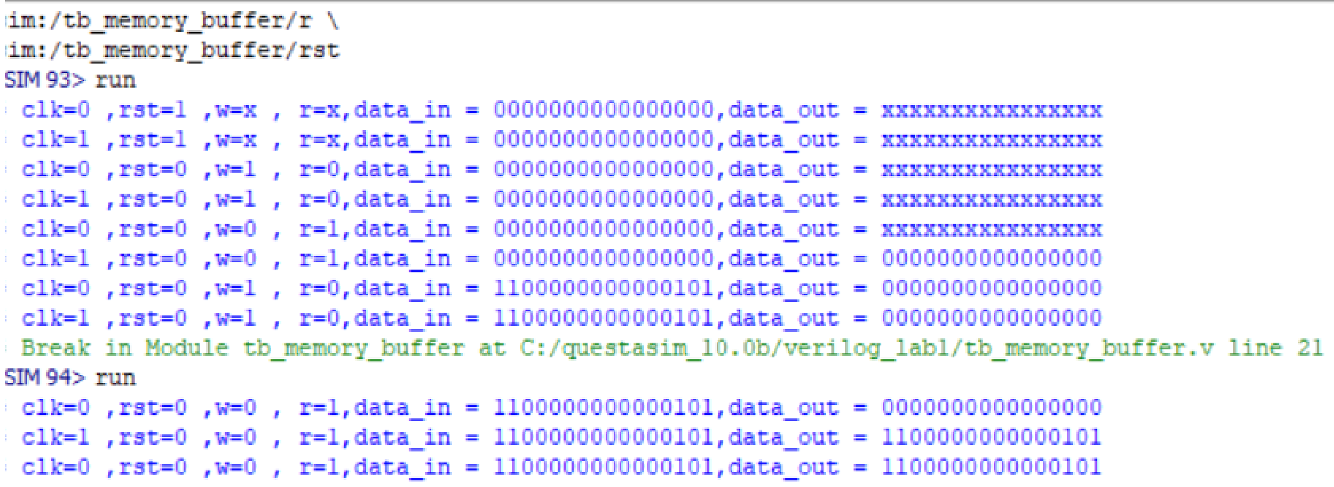
Design code



```verilog
module memory_buffer (
      input   clk,rst,w,r,
      input   [15:0] data_in,
      output reg[15:0] data_out);
      reg [15:0] memory [0:15];
      reg [3:0] write_ptr,read_ptr = 4'b0000;
      always @(posedge clk) begin
          if (rst) begin
              write_ptr <= 4'b0000;
              read_ptr <= 4'b0000;
          end else begin
              if (w) begin
                  memory[write_ptr] <= data_in;
                  write_ptr <= write_ptr + 1;
              end
              if (r) begin
                  data_out <= memory[read_ptr];
                  read_ptr <= read_ptr + 1;
              end
          end
      end
 endmodule
```

Simulation results





```
im:/tb_memory_buffer/r \
im:/tb_memory_buffer/rst
SIM 93> run
 clk=0 ,rst=1 ,w=x , r=x,data_in = 0000000000000000,data_out = xxxxxxxxxxxxxxxx
 clk=1 ,rst=1 ,w=x , r=x,data_in = 0000000000000000,data_out = xxxxxxxxxxxxxxxx
 clk=0 ,rst=0 ,w=1 , r=0,data_in = 0000000000000000,data_out = xxxxxxxxxxxxxxxx
 clk=1 ,rst=0 ,w=1 , r=0,data_in = 0000000000000000,data_out = xxxxxxxxxxxxxxxx
 clk=0 ,rst=0 ,w=0 , r=1,data_in = 0000000000000000,data_out = xxxxxxxxxxxxxxxx
 clk=1 ,rst=0 ,w=0 , r=1,data_in = 0000000000000000,data_out = 0000000000000000
 clk=0 ,rst=0 ,w=1 , r=0,data_in = 1100000000000101,data_out = 0000000000000000
 clk=1 ,rst=0 ,w=1 , r=0,data_in = 1100000000000101,data_out = 0000000000000000
 Break in Module tb_memory_buffer at C:/questasim_10.0b/verilog_lab1/tb_memory_buffer.v line 21
SIM 94> run
 clk=0 ,rst=0 ,w=0 , r=1,data_in = 1100000000000101,data_out = 0000000000000000
 clk=1 ,rst=0 ,w=0 , r=1,data_in = 1100000000000101,data_out = 1100000000000101
 clk=0 ,rst=0 ,w=0 , r=1,data_in = 1100000000000101,data_out = 1100000000000101
```