LAB 5 – Raja Aadhithan

Design – Single port RAM:

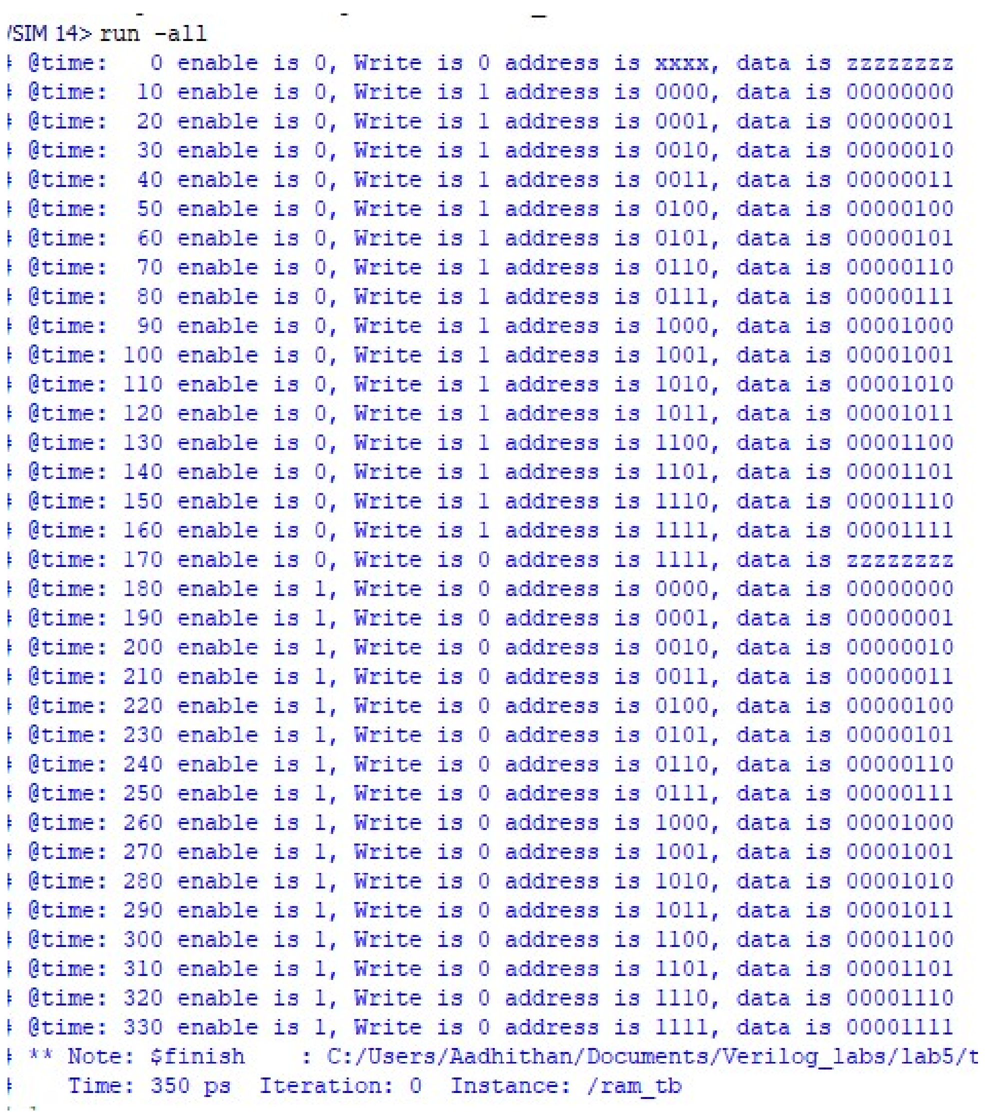
Code:

```
module ram(input we_in,enable_in,
      input [3:0]addr_in,
      inout [7:0]data);


   //Step1 : Declare a 8 bit wide memory having 16 locations.
reg [7:0] mem [15:0];
   //Understand the logic for writing data into a memory location
   always@(data,we_in,enable_in,addr_in)
      if(we_in && !enable_in)
    mem[addr_in]=data;

   //Understand the logic of reading data from a memory location
   assign data= (enable_in && !we_in) ? mem[addr_in] : 8'hzz;

endmodule
```

Testbench:

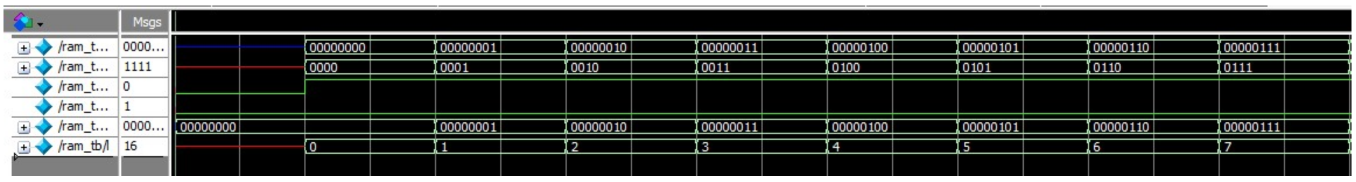
```
module ram_tb;
   wire [7:0] data;
   reg  [3:0] addr;
   reg  we,enable;
   reg  [7:0] tempd;

   integer l;

   //Step1 : Instantiate the RAM module and connect the ports
   ram dut(we,enable,addr,data);
   //Understand how the wire data acts like an input during write operation
   assign data=(we && !enable) ? tempd : 8'hzz;

    //Tasks for Initialising the inputs
   task initialize();
      begin
    we=1'b0; enable=1'b0; tempd=8'h00;
      end
   endtask

   /*Step2 : Write a task named "stimulus" to assign data into
   "addr" and "tempd" inputs through i and j variables*/
   task stimulus(input [3:0]i,
       input [7:0]j);
      begin
    addr = i;
    tempd = j;
      end
   endtask

   //Understand the various tasks used in this testbench
   task write();
      begin
    we=1'b1;
    enable=1'b0;
      end
   endtask
   task read();
      begin
    we=1'b0;
    enable=1'b1;
      end
   endtask

   task delay;
      begin
    #10;
      end
   endtask

   //Process to generate stimulus using for loop
   initial
      begin
    initialize;
    delay;
    write;
    for(l=0;l<16;l=l+1)
       begin
          stimulus(l,l);
          delay;
       end
    initialize;
    delay;
    read;
    for(l=0;l<16;l=l+1)
       begin
          stimulus(l,l);
          delay;
       end
    delay;
    $finish;
      end



endmodule
```

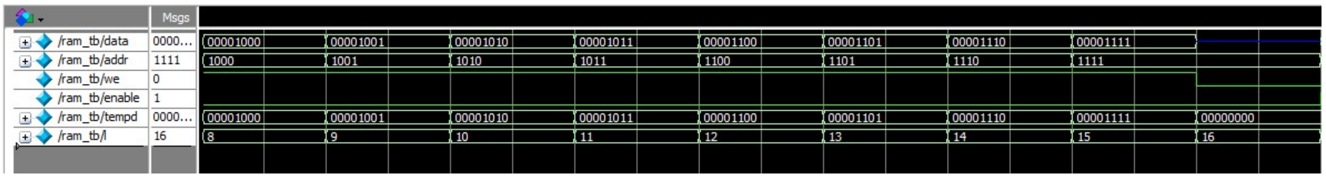
Output:

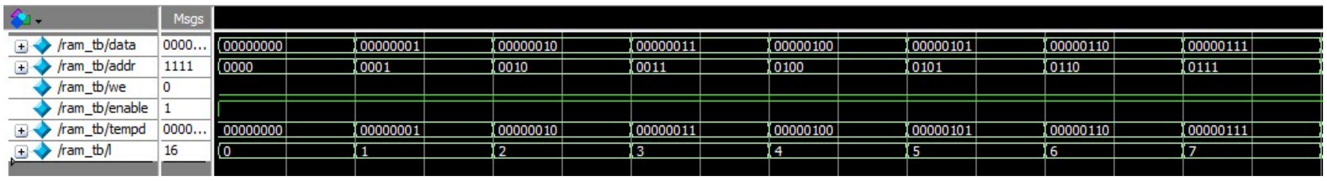


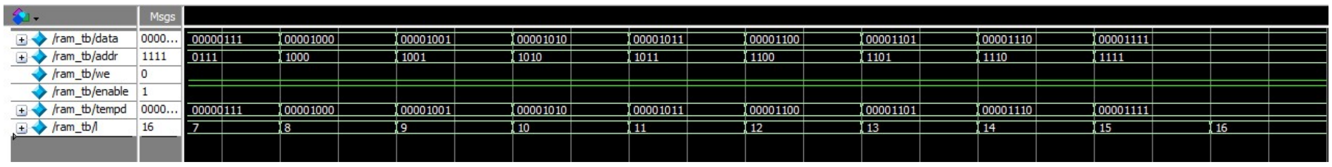
```
/SIM 14> run -all
# @time:   0 enable is 0, Write is 0 address is xxxx, data is zzzzzzzz
# @time:  10 enable is 0, Write is 1 address is 0000, data is 00000000
# @time:  20 enable is 0, Write is 1 address is 0001, data is 00000001
# @time:  30 enable is 0, Write is 1 address is 0010, data is 00000010
# @time:  40 enable is 0, Write is 1 address is 0011, data is 00000011
# @time:  50 enable is 0, Write is 1 address is 0100, data is 00000100
# @time:  60 enable is 0, Write is 1 address is 0101, data is 00000101
# @time:  70 enable is 0, Write is 1 address is 0110, data is 00000110
# @time:  80 enable is 0, Write is 1 address is 0111, data is 00000111
# @time:  90 enable is 0, Write is 1 address is 1000, data is 00001000
# @time: 100 enable is 0, Write is 1 address is 1001, data is 00001001
# @time: 110 enable is 0, Write is 1 address is 1010, data is 00001010
# @time: 120 enable is 0, Write is 1 address is 1011, data is 00001011
# @time: 130 enable is 0, Write is 1 address is 1100, data is 00001100
# @time: 140 enable is 0, Write is 1 address is 1101, data is 00001101
# @time: 150 enable is 0, Write is 1 address is 1110, data is 00001110
# @time: 160 enable is 0, Write is 1 address is 1111, data is 00001111
# @time: 170 enable is 0, Write is 0 address is 1111, data is zzzzzzzz
# @time: 180 enable is 1, Write is 0 address is 0000, data is 00000000
# @time: 190 enable is 1, Write is 0 address is 0001, data is 00000001
# @time: 200 enable is 1, Write is 0 address is 0010, data is 00000010
# @time: 210 enable is 1, Write is 0 address is 0011, data is 00000011
# @time: 220 enable is 1, Write is 0 address is 0100, data is 00000100
# @time: 230 enable is 1, Write is 0 address is 0101, data is 00000101
# @time: 240 enable is 1, Write is 0 address is 0110, data is 00000110
# @time: 250 enable is 1, Write is 0 address is 0111, data is 00000111
# @time: 260 enable is 1, Write is 0 address is 1000, data is 00001000
# @time: 270 enable is 1, Write is 0 address is 1001, data is 00001001
# @time: 280 enable is 1, Write is 0 address is 1010, data is 00001010
# @time: 290 enable is 1, Write is 0 address is 1011, data is 00001011
# @time: 300 enable is 1, Write is 0 address is 1100, data is 00001100
# @time: 310 enable is 1, Write is 0 address is 1101, data is 00001101
# @time: 320 enable is 1, Write is 0 address is 1110, data is 00001110
# @time: 330 enable is 1, Write is 0 address is 1111, data is 00001111
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/lab5/t
#    Time: 350 ps  Iteration: 0  Instance: /ram_tb
```

Wave:









RTL:

Too lengthy to crop.

Exercise:

Design : FIFO:

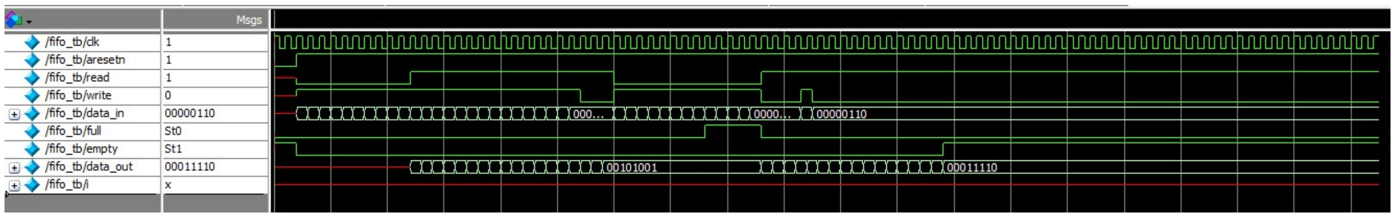
Code:

```
module fifo (clk,reset,read,write,data_in,full,empty,data_out);

parameter WIDTH = 8;
parameter DEPTH = 16;

output reg [WIDTH-1 : 0] data_out;
output full;
output empty;

input [WIDTH-1 : 0] data_in;
input clk;
input reset,read,write;

reg [WIDTH-1 : 0] mem [DEPTH-1 : 0];
reg [3 : 0] rd_pointer;
reg [3 : 0] wr_pointer;

assign empty = ((wr_pointer - rd_pointer) == 0) ? 1'b1 : 1'b0;
assign full  = ((wr_pointer - rd_pointer) == DEPTH) ? 1'b1 : 1'b0;

always @(posedge clk or negedge reset) begin
    if (!reset) begin
        // reset
        wr_pointer <= 0;
        rd_pointer <= 0;
    end
    else begin
        if (full == 1'b0 & write) begin
            mem[wr_pointer] <= data_in;
            wr_pointer <= wr_pointer + 1;
        end
        if (empty == 1'b0 & read) begin
            data_out <= mem[rd_pointer];
            rd_pointer <= rd_pointer + 1;
        end
    end
end
endmodule
```

Testbench:

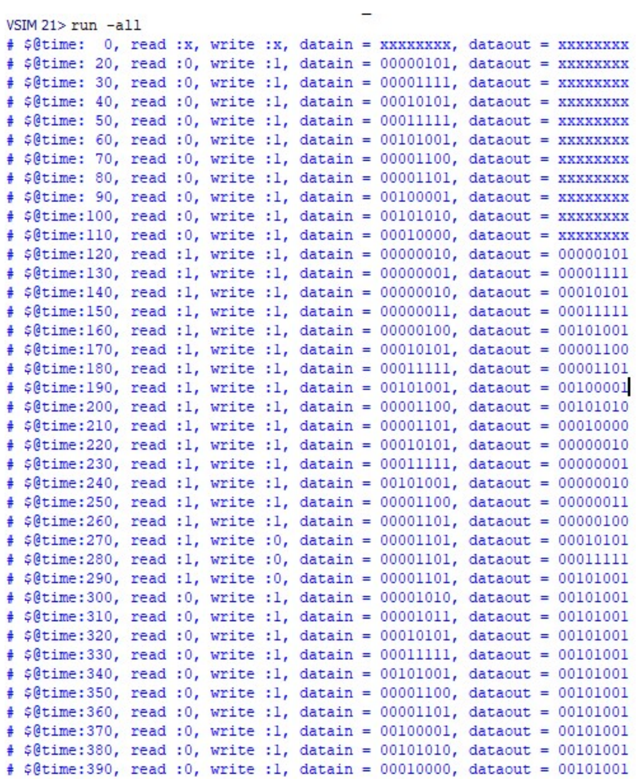
```
module fifo_tb();
reg clk,aresetn,read,write;
reg [7:0]data_in;
wire full,empty;
wire [7:0]data_out;
integer i;

fifo dut(clk,aresetn,read,write,data_in,full,empty,data_out);
initial begin
    clk = 1'b1;
    forever #5 clk = ~clk;
end

initial begin
    $monitor("$@time:%3d, read :%b, write :%b, datain = %b, dataout = %b",$time,read,write,data_in,data_out);
    aresetn = 0;
    #20;
    aresetn = 1;
    write = 1;
    read = 0;
    data_in = 8'd5;
    #10;
    data_in = 8'd15;
    #10;
    data_in = 8'd21;
    #10;
    data_in = 8'd31;
    #10;
    data_in = 8'd41;
    #10;
    data_in = 8'd12;
    #10;
    data_in = 8'd13;
    #10;
    data_in = 8'd33;
    #10;
    data_in = 8'd42;
    #10;
    data_in = 8'd16;
    #10;
    read = 1;
    data_in = 8'd2;
    #10;
    data_in = 8'd1;
    #10;
    data_in = 8'd2;
    #10;
    data_in = 8'd3;
    #10;
    data_in = 8'd4;
    #10;
    data_in = 8'd21;
    #10;
    data_in = 8'd31;
    #10;
    data_in = 8'd41;
    #10;
    data_in = 8'd12;
    #10;
    data_in = 8'd13;
    #10;
    data_in = 8'd21;
    #10;
    data_in = 8'd31;
    #10;
    data_in = 8'd41;
    #10;
    data_in = 8'd12;
    #10;
    data_in = 8'd13;
    #10;
    write = 0;
    #30;
    read = 0;
    write =1;
    data_in = 8'd10;
    #10;
    data_in = 8'd11;
    #10;
    data_in = 8'd21;
    #10;
    data_in = 8'd31;
    #10;
    data_in = 8'd41;
    #10;
    data_in = 8'd12;
    #10;
    data_in = 8'd13;
    #10;
    data_in = 8'd33;
    #10;
    data_in = 8'd42;
    #10;
    data_in = 8'd16;
    #10;
    data_in = 8'd22;
    #10;
    data_in = 8'd30;
    #10;
    data_in = 8'd06;
    #10;
    read = 1;
    write = 0;
    #35;
    write =1;
    data_in = 8'd30;
    #10;
    data_in = 8'd06;
    write =0;
    #500;
    $finish;
end
endmodule
```

Wave: Zoomed out version

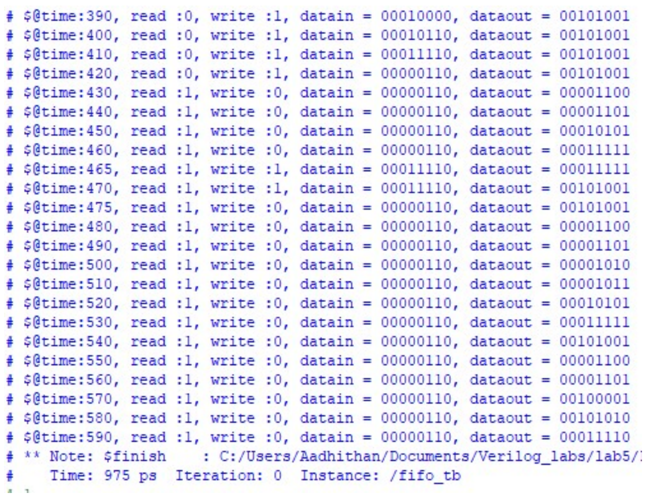


	Msgs
/fifo_tb/clk	1
/fifo_tb/aresetn	1
/fifo_tb/read	1
/fifo_tb/write	0
/fifo_tb/data_in	00000110
/fifo_tb/full	St0
/fifo_tb/empty	St1
/fifo_tb/data_out	00011110
/fifo_tb/i	x

Output:

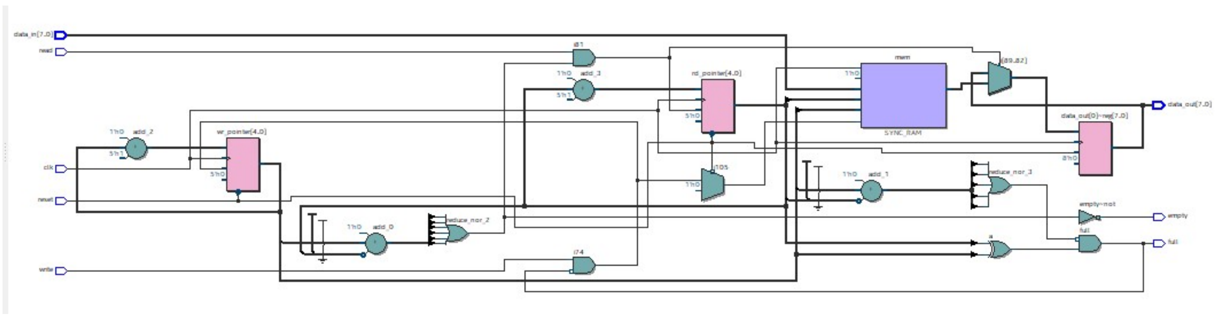


```
VSIM 21> run -all
# $@time:  0, read :x, write :x, datain = xxxxxxxx, dataout = xxxxxxxx
# $@time: 20, read :0, write :1, datain = 00000101, dataout = xxxxxxxx
# $@time: 30, read :0, write :1, datain = 00001111, dataout = xxxxxxxx
# $@time: 40, read :0, write :1, datain = 00010101, dataout = xxxxxxxx
# $@time: 50, read :0, write :1, datain = 00011111, dataout = xxxxxxxx
# $@time: 60, read :0, write :1, datain = 00101001, dataout = xxxxxxxx
# $@time: 70, read :0, write :1, datain = 00001100, dataout = xxxxxxxx
# $@time: 80, read :0, write :1, datain = 00001101, dataout = xxxxxxxx
# $@time: 90, read :0, write :1, datain = 00100001, dataout = xxxxxxxx
# $@time:100, read :0, write :1, datain = 00101010, dataout = xxxxxxxx
# $@time:110, read :0, write :1, datain = 00010000, dataout = xxxxxxxx
# $@time:120, read :1, write :1, datain = 00000010, dataout = 00000101
# $@time:130, read :1, write :1, datain = 00000001, dataout = 00001111
# $@time:140, read :1, write :1, datain = 00000010, dataout = 00010101
# $@time:150, read :1, write :1, datain = 00000011, dataout = 00011111
# $@time:160, read :1, write :1, datain = 00000100, dataout = 00101001
# $@time:170, read :1, write :1, datain = 00010101, dataout = 00001100
# $@time:180, read :1, write :1, datain = 00011111, dataout = 00001101
# $@time:190, read :1, write :1, datain = 00101001, dataout = 00100001
# $@time:200, read :1, write :1, datain = 00001100, dataout = 00101010
# $@time:210, read :1, write :1, datain = 00001101, dataout = 00010000
# $@time:220, read :1, write :1, datain = 00010101, dataout = 00000010
# $@time:230, read :1, write :1, datain = 00011111, dataout = 00000001
# $@time:240, read :1, write :1, datain = 00101001, dataout = 00000010
# $@time:250, read :1, write :1, datain = 00001100, dataout = 00000011
# $@time:260, read :1, write :1, datain = 00001101, dataout = 00000100
# $@time:270, read :1, write :0, datain = 00001101, dataout = 00010101
# $@time:280, read :1, write :0, datain = 00001101, dataout = 00011111
# $@time:290, read :1, write :0, datain = 00001101, dataout = 00101001
# $@time:300, read :0, write :1, datain = 00001010, dataout = 00101001
# $@time:310, read :0, write :1, datain = 00001011, dataout = 00101001
# $@time:320, read :0, write :1, datain = 00010101, dataout = 00101001
# $@time:330, read :0, write :1, datain = 00011111, dataout = 00101001
# $@time:340, read :0, write :1, datain = 00101001, dataout = 00101001
# $@time:350, read :0, write :1, datain = 00001100, dataout = 00101001
# $@time:360, read :0, write :1, datain = 00001101, dataout = 00101001
# $@time:370, read :0, write :1, datain = 00100001, dataout = 00101001
# $@time:380, read :0, write :1, datain = 00101010, dataout = 00101001
# $@time:390, read :0, write :1, datain = 00010000, dataout = 00101001
```



```
# $@time:390, read :0, write :1, datain = 00010000, dataout = 00101001
# $@time:400, read :0, write :1, datain = 00010110, dataout = 00101001
# $@time:410, read :0, write :1, datain = 00011110, dataout = 00101001
# $@time:420, read :0, write :1, datain = 00000110, dataout = 00101001
# $@time:430, read :1, write :0, datain = 00000110, dataout = 00001100
# $@time:440, read :1, write :0, datain = 00000110, dataout = 00001101
# $@time:450, read :1, write :0, datain = 00000110, dataout = 00010101
# $@time:460, read :1, write :0, datain = 00000110, dataout = 00011111
# $@time:465, read :1, write :1, datain = 00011110, dataout = 00011111
# $@time:470, read :1, write :1, datain = 00011110, dataout = 00101001
# $@time:475, read :1, write :0, datain = 00000110, dataout = 00101001
# $@time:480, read :1, write :0, datain = 00000110, dataout = 00001100
# $@time:490, read :1, write :0, datain = 00000110, dataout = 00001101
# $@time:500, read :1, write :0, datain = 00000110, dataout = 00001010
# $@time:510, read :1, write :0, datain = 00000110, dataout = 00001011
# $@time:520, read :1, write :0, datain = 00000110, dataout = 00010101
# $@time:530, read :1, write :0, datain = 00000110, dataout = 00011111
# $@time:540, read :1, write :0, datain = 00000110, dataout = 00101001
# $@time:550, read :1, write :0, datain = 00000110, dataout = 00001100
# $@time:560, read :1, write :0, datain = 00000110, dataout = 00001101
# $@time:570, read :1, write :0, datain = 00000110, dataout = 00100001
# $@time:580, read :1, write :0, datain = 00000110, dataout = 00101010
# $@time:590, read :1, write :0, datain = 00000110, dataout = 00011110
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/lab5/
#    Time: 975 ps  Iteration: 0  Instance: /fifo_tb
```

RTL:



Assignments:

Design : 16x8 dual port Ram

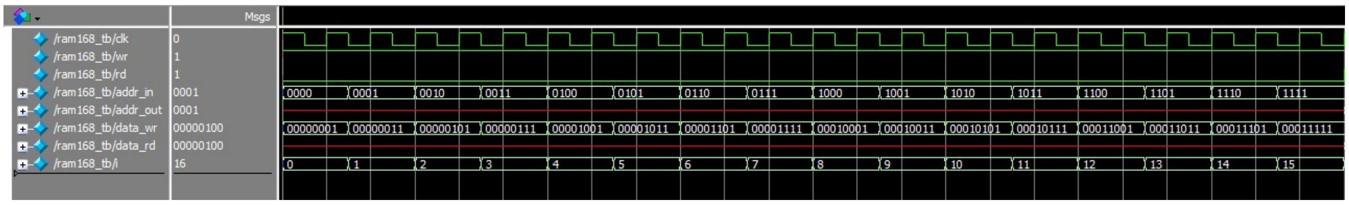
Code:

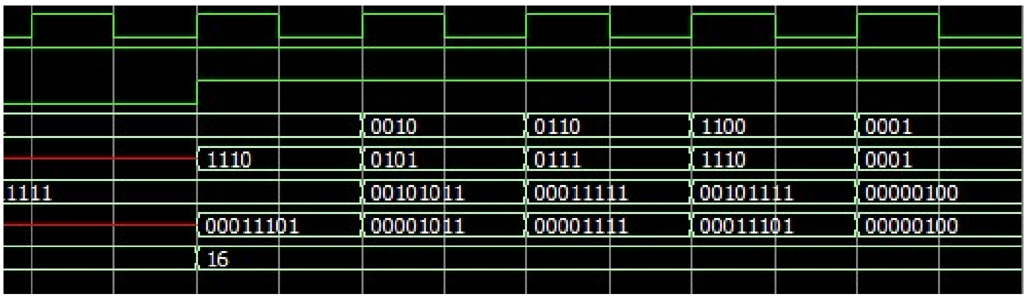
```
module ram168(input clk,wr_in,rd_in,
      input [3:0]addr_in,addr_out,
      input [7:0]data_wr, output [7:0] data_rd);
   reg [7:0]out;

reg [7:0] mem [15:0];

   always@(posedge clk)begin
      if(wr_in) mem[addr_in]=data_wr;
      if(rd_in) out = (rd_in) ? mem[addr_out] : 8'hzz;
   end
   assign data_rd = out;
endmodule
```

Testbench:

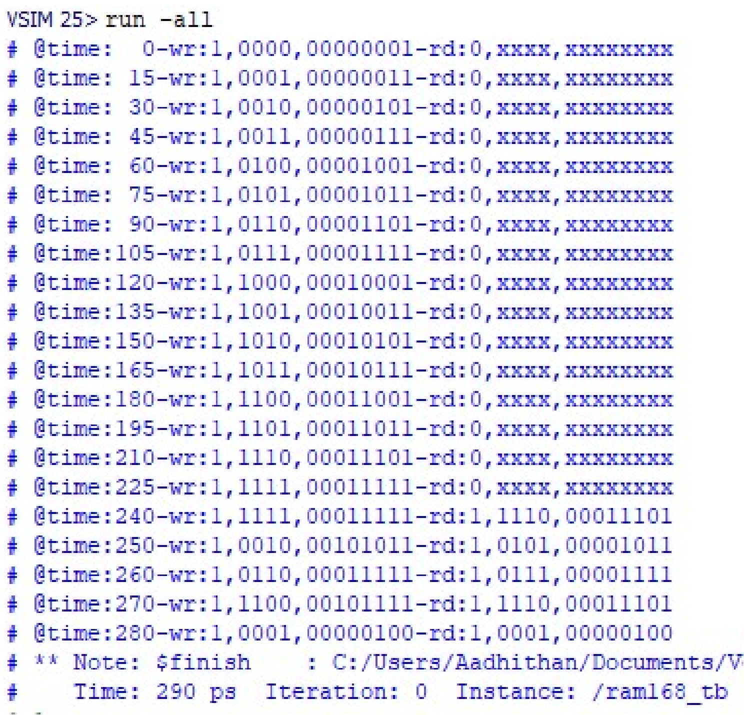
```
module ram168_tb();
reg clk,wr,rd;
reg [3:0] addr_in,addr_out;
reg [7:0]data_wr;
wire [7:0] data_rd;
integer i;
ram168 dut(clk,wr,rd,addr_in,addr_out,data_wr,data_rd);
initial begin
    clk =1;
    forever #5 clk = !clk;
end
initial begin
    wr = 1;
    rd = 0;
    $monitor("@time:%3d-wr:%b,%b,%b-rd:%b,%b,%b",$time,wr,addr_in,data_wr,rd,addr_out,data_rd);
    for (i=0;i<16;i=i+1)begin
        addr_in = i[3:0];
        data_wr = 2*i+1;
        #15;
    end
    rd = 1;
    addr_out = 14;
    #10;
    addr_in = 4'd2;
    data_wr = 8'd43;
    addr_out = 4'd5;
    #10;
    addr_in = 4'd6;
    data_wr = 8'd31;
    addr_out = 4'd7;
    #10;
    addr_in = 4'd12;
    data_wr = 8'd47;
    addr_out = 4'd14;
    #10;
    addr_in = 4'd1;
    data_wr = 8'd4;
    addr_out = 4'd1;
    #10;

$finish;
end
endmodule
```

Wave:



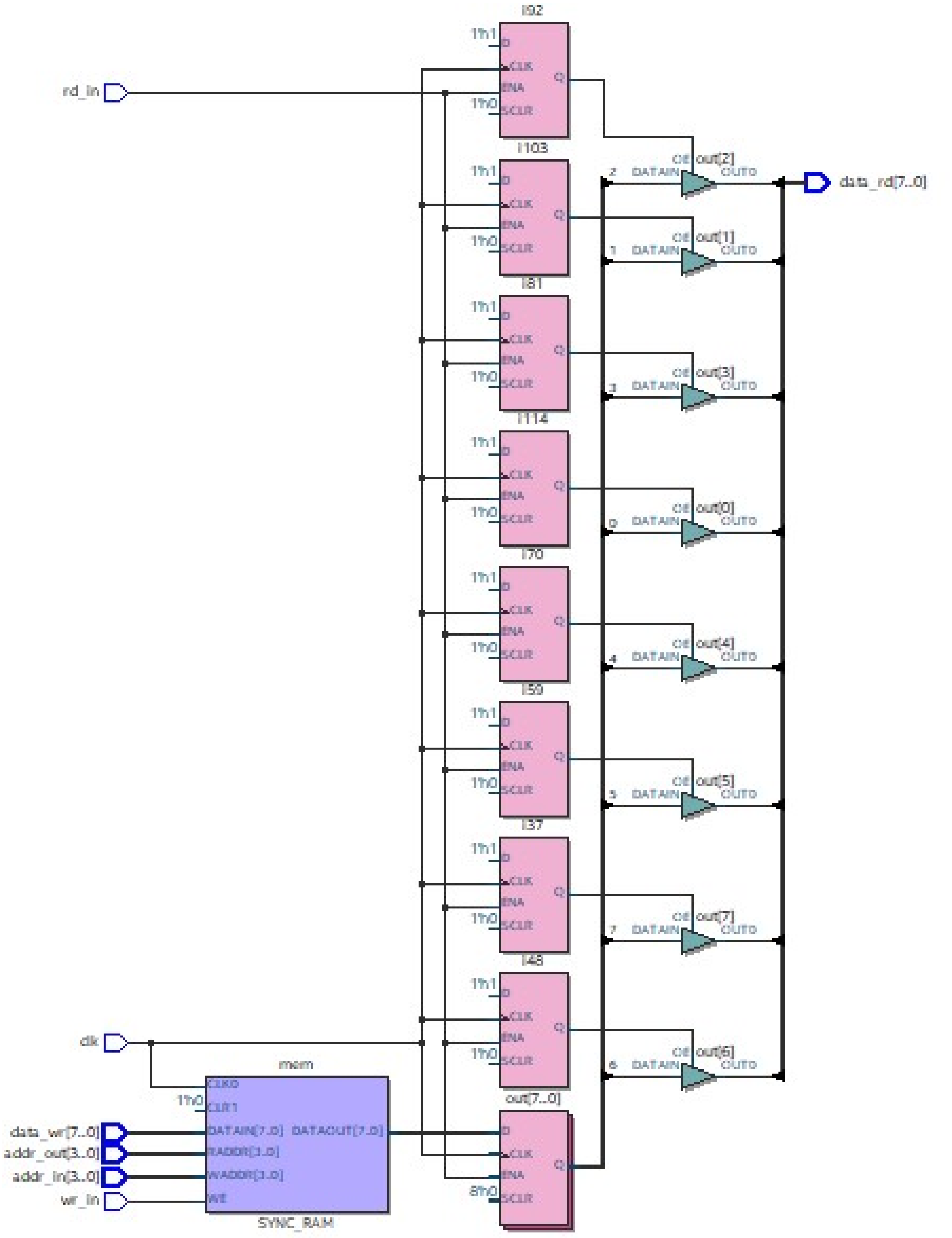


Output:



```
VSIM 25> run -all
# @time:  0-wr:1,0000,00000001-rd:0,xxxx,xxxxxxxx
# @time: 15-wr:1,0001,00000011-rd:0,xxxx,xxxxxxxx
# @time: 30-wr:1,0010,00000101-rd:0,xxxx,xxxxxxxx
# @time: 45-wr:1,0011,00000111-rd:0,xxxx,xxxxxxxx
# @time: 60-wr:1,0100,00001001-rd:0,xxxx,xxxxxxxx
# @time: 75-wr:1,0101,00001011-rd:0,xxxx,xxxxxxxx
# @time: 90-wr:1,0110,00001101-rd:0,xxxx,xxxxxxxx
# @time:105-wr:1,0111,00001111-rd:0,xxxx,xxxxxxxx
# @time:120-wr:1,1000,00010001-rd:0,xxxx,xxxxxxxx
# @time:135-wr:1,1001,00010011-rd:0,xxxx,xxxxxxxx
# @time:150-wr:1,1010,00010101-rd:0,xxxx,xxxxxxxx
# @time:165-wr:1,1011,00010111-rd:0,xxxx,xxxxxxxx
# @time:180-wr:1,1100,00011001-rd:0,xxxx,xxxxxxxx
# @time:195-wr:1,1101,00011011-rd:0,xxxx,xxxxxxxx
# @time:210-wr:1,1110,00011101-rd:0,xxxx,xxxxxxxx
# @time:225-wr:1,1111,00011111-rd:0,xxxx,xxxxxxxx
# @time:240-wr:1,1111,00011111-rd:1,1110,00011101
# @time:250-wr:1,0010,00101011-rd:1,0101,00001011
# @time:260-wr:1,0110,00011111-rd:1,0111,00001111
# @time:270-wr:1,1100,00101111-rd:1,1110,00011101
# @time:280-wr:1,0001,00000100-rd:1,0001,00000100
# ** Note: $finish    : C:/Users/Aadhithan/Documents/V
#    Time: 290 ps  Iteration: 0  Instance: /ram168_tb
```

RTL:



I92
I103
I81
I114
I70
I59
I37
I48
out[7..0]
1'h1
1'h0
8'h0
D
CLK
ENA
SCLR
Q
rd_lin
clk
data_wr[7..0]
addr_out[3..0]
addr_in[3..0]
wr_lin
mem
CLK0
CLK1
DATAIN[7..0]
DATAOUT[7..0]
RADDR[3..0]
WADDR[3..0]
WE
SYNC_RAM
OE
DATAIN
OUT0
out[2]
out[1]
out[3]
out[0]
out[4]
out[5]
out[7]
out[6]
data_rd[7..0]

Design : 8x16 Ram:

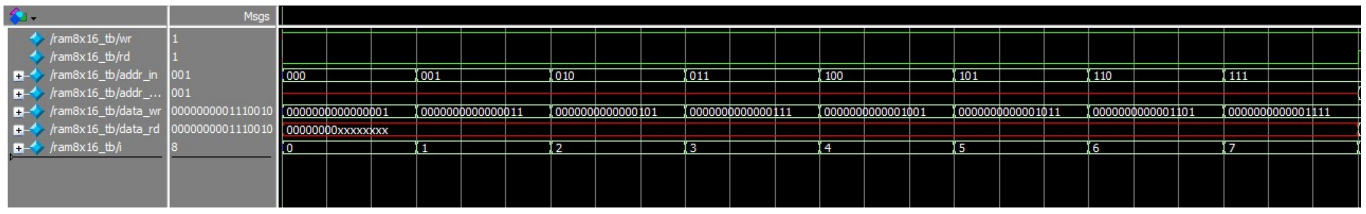
Code:

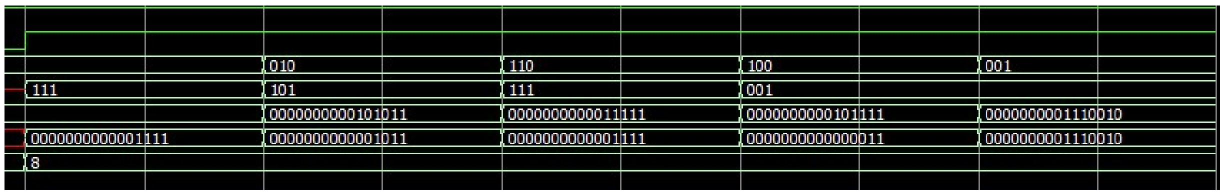
```
module aram8x16(input wr_in,rd_in,
      input [2:0]addr_in,addr_out,
      input [15:0]data_wr, output [15:0] data_rd);
   reg [7:0]out;

reg [15:0] mem [7:0];

   always@(*)begin
      if(wr_in) mem[addr_in]=data_wr;
      if(rd_in) out = (rd_in) ? mem[addr_out] : 16'hzz;
   end
   assign data_rd = out;
endmodule
```

Testbench:

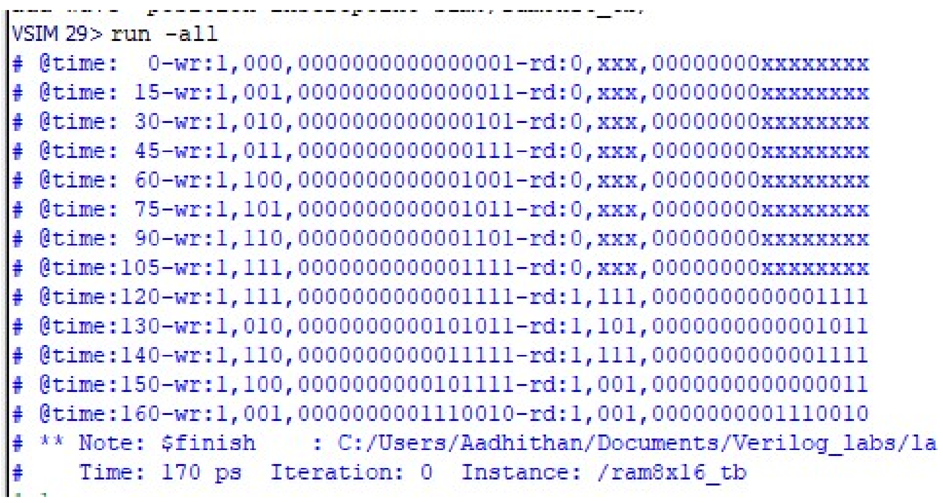
```
module ram8x16_tb();
reg wr,rd;
reg [2:0] addr_in,addr_out;
reg [15:0]data_wr;
wire [15:0] data_rd;
integer i;
aram8x16 dut(wr,rd,addr_in,addr_out,data_wr,data_rd);
initial begin
    wr = 1;
    rd = 0;
    $monitor("@time:%3d-wr:%b,%b,%b-rd:%b,%b,%b",$time,wr,addr_in,data_wr,rd,addr_out,data_rd);
    for (i=0;i<8;i=i+1)begin
        addr_in = i[2:0];
        data_wr = 2*i+1;
        #15;
    end
    rd = 1;
    addr_out = 7;
    #10;
    addr_in = 3'd2;
    data_wr = 16'd43;
    addr_out = 3'd5;
    #10;
    addr_in = 3'd6;
    data_wr = 16'd31;
    addr_out = 3'd7;
    #10;
    addr_in = 3'd4;
    data_wr = 16'd47;
    addr_out = 3'd1;
    #10;
    addr_in = 3'd1;
    data_wr = 16'd114;
    addr_out = 3'd1;
    #10;

$finish;
end
endmodule
```

Wave:





Output:


```
VSIM 29> run -all
# @time:  0-wr:1,000,0000000000000001-rd:0,xxx,00000000xxxxxxxx
# @time: 15-wr:1,001,0000000000000011-rd:0,xxx,00000000xxxxxxxx
# @time: 30-wr:1,010,0000000000000101-rd:0,xxx,00000000xxxxxxxx
# @time: 45-wr:1,011,0000000000000111-rd:0,xxx,00000000xxxxxxxx
# @time: 60-wr:1,100,0000000000001001-rd:0,xxx,00000000xxxxxxxx
# @time: 75-wr:1,101,0000000000001011-rd:0,xxx,00000000xxxxxxxx
# @time: 90-wr:1,110,0000000000001101-rd:0,xxx,00000000xxxxxxxx
# @time:105-wr:1,111,0000000000001111-rd:0,xxx,00000000xxxxxxxx
# @time:120-wr:1,111,0000000000001111-rd:1,111,0000000000001111
# @time:130-wr:1,010,0000000000101011-rd:1,101,0000000000001011
# @time:140-wr:1,110,0000000000011111-rd:1,111,0000000000001111
# @time:150-wr:1,100,0000000000101111-rd:1,001,0000000000000011
# @time:160-wr:1,001,0000000001110010-rd:1,001,0000000001110010
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/la
#    Time: 170 ps  Iteration: 0  Instance: /ram8x16_tb
```

RTL:

Too length to attach.

Design : 4 bit SISO

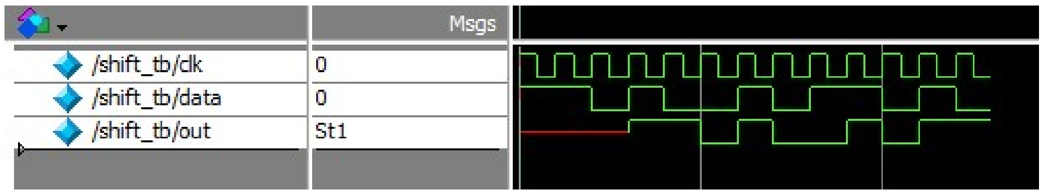
Code:

```
module shift(input clk,data, output out);
reg [3:0] regis;
always@(posedge clk)begin
    regis[3] <= data;
    regis[2] <= regis[3];
    regis[1] <= regis[2];
    regis[0] <= regis[1];
end
assign out = regis[0];
endmodule
```

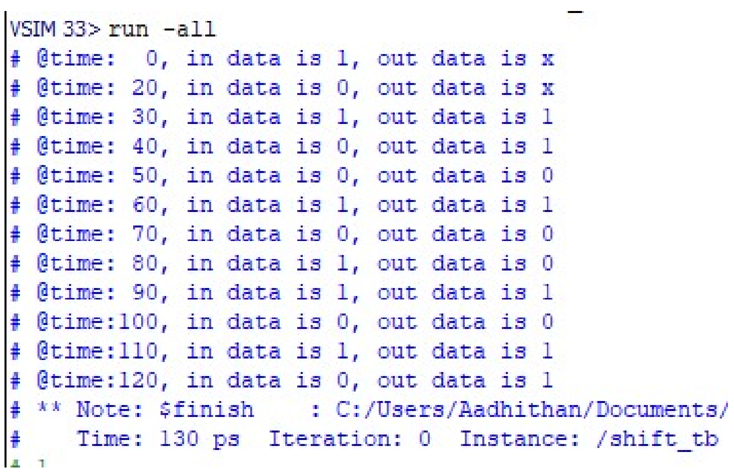
Testbench:

```
module shift_tb();
reg clk,data;
wire out;
shift dut(clk,data,out);
initial begin
    clk = 1;
    forever #5 clk = ~clk;
end
initial begin
    $monitor("@time:%3d, in data is %b, out data is %b",$time,data,out);
    data = 1;
    #10;
    data = 1;
    #10;
    data = 0;
    #10;
    data = 1;
    #10;
    data = 0;
    #10;
    data = 0;
    #10;
    data = 1;
    #10;
    data = 0;
    #10;
    data = 1;
    #10;
    data = 1;
    #10;
    data = 0;
    #10;
    data = 1;
    #10;
    data = 0;
    #10;
$finish;
end
endmodule
```

Wave:

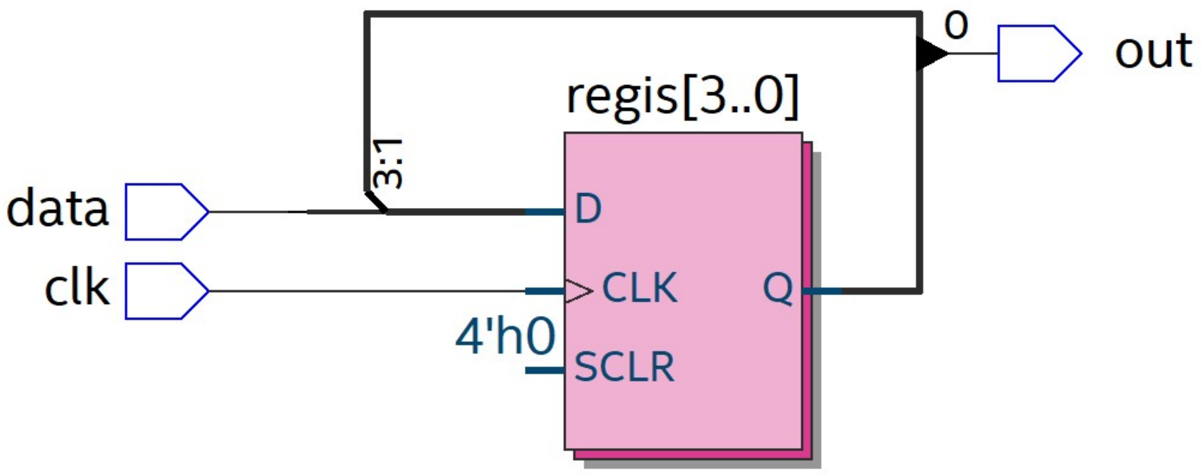


Output:



```
VSIM 33> run -all
# @time:  0, in data is 1, out data is x
# @time: 20, in data is 0, out data is x
# @time: 30, in data is 1, out data is 1
# @time: 40, in data is 0, out data is 1
# @time: 50, in data is 0, out data is 0
# @time: 60, in data is 1, out data is 1
# @time: 70, in data is 0, out data is 0
# @time: 80, in data is 1, out data is 0
# @time: 90, in data is 1, out data is 1
# @time:100, in data is 0, out data is 0
# @time:110, in data is 1, out data is 1
# @time:120, in data is 0, out data is 1
# ** Note: $finish    : C:/Users/Aadhithan/Documents/
#    Time: 130 ps  Iteration: 0  Instance: /shift_tb
```

RTL:



Design : Clock buffer:

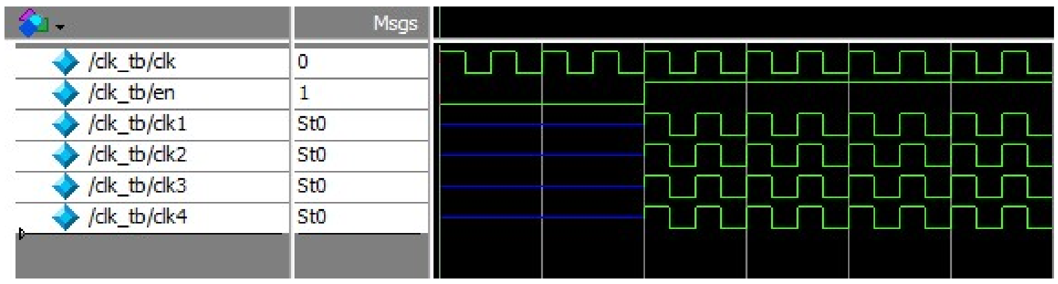
Code:

```
module clockbuf(input clk,enb,output clk1,clk2,clk3,clk4);
wire x;
buf(x,clk);
bufif1(clk1,x,enb);
bufif1(clk2,x,enb);
bufif1(clk3,x,enb);
bufif1(clk4,x,enb);
endmodule
```

Testbench:

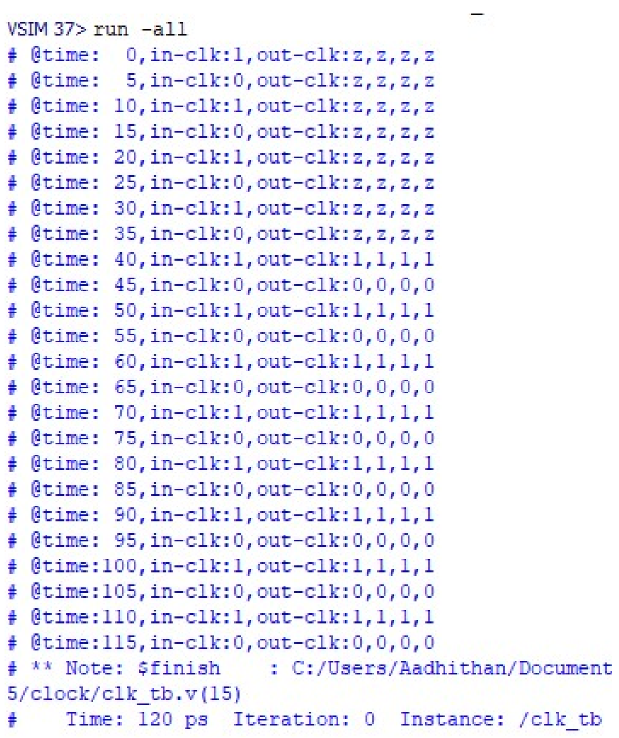
```
module clk_tb();
reg clk,en;
wire clk1,clk2,clk3,clk4;
clockbuf dut(clk,en,clk1,clk2,clk3,clk4);
initial begin
    clk =1;
    forever #5 clk = ~clk;
end
initial begin
    $monitor("@time:%3d,in-clk:%b,out-clk:%b,%b,%b,%b",$time,clk,clk1,clk2,clk3,clk4);
    en = 0 ;
    #40;
    en = 1 ;
    #80;
    $finish;
end
endmodule
```

Wave:



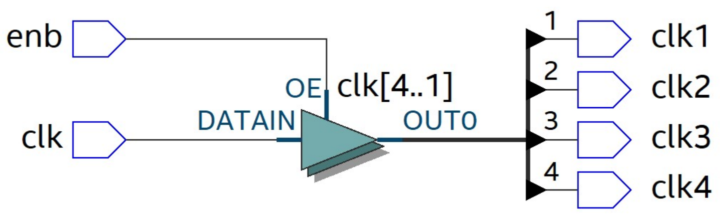
	Msgs
/clk_tb/clk	0
/clk_tb/en	1
/clk_tb/clk1	St0
/clk_tb/clk2	St0
/clk_tb/clk3	St0
/clk_tb/clk4	St0

Output:



```
VSIM 37> run -all
# @time:  0,in-clk:1,out-clk:z,z,z,z
# @time:  5,in-clk:0,out-clk:z,z,z,z
# @time: 10,in-clk:1,out-clk:z,z,z,z
# @time: 15,in-clk:0,out-clk:z,z,z,z
# @time: 20,in-clk:1,out-clk:z,z,z,z
# @time: 25,in-clk:0,out-clk:z,z,z,z
# @time: 30,in-clk:1,out-clk:z,z,z,z
# @time: 35,in-clk:0,out-clk:z,z,z,z
# @time: 40,in-clk:1,out-clk:1,1,1,1
# @time: 45,in-clk:0,out-clk:0,0,0,0
# @time: 50,in-clk:1,out-clk:1,1,1,1
# @time: 55,in-clk:0,out-clk:0,0,0,0
# @time: 60,in-clk:1,out-clk:1,1,1,1
# @time: 65,in-clk:0,out-clk:0,0,0,0
# @time: 70,in-clk:1,out-clk:1,1,1,1
# @time: 75,in-clk:0,out-clk:0,0,0,0
# @time: 80,in-clk:1,out-clk:1,1,1,1
# @time: 85,in-clk:0,out-clk:0,0,0,0
# @time: 90,in-clk:1,out-clk:1,1,1,1
# @time: 95,in-clk:0,out-clk:0,0,0,0
# @time:100,in-clk:1,out-clk:1,1,1,1
# @time:105,in-clk:0,out-clk:0,0,0,0
# @time:110,in-clk:1,out-clk:1,1,1,1
# @time:115,in-clk:0,out-clk:0,0,0,0
# ** Note: $finish    : C:/Users/Aadhithan/Document
5/clock/clk_tb.v(15)
#    Time: 120 ps  Iteration: 0  Instance: /clk_tb
```

RTL:


enb
clk
OE
DATAIN
clk[4..1]
OUT0
1
2
3
4
clk1
clk2
clk3
clk4