Lab#2: Basma Gamal Fawzy

Problem 1:

a) Implement a Verilog code for 4-bit PISO.

b) Implement a Verilog code for 4-bit SIPO.

PISO

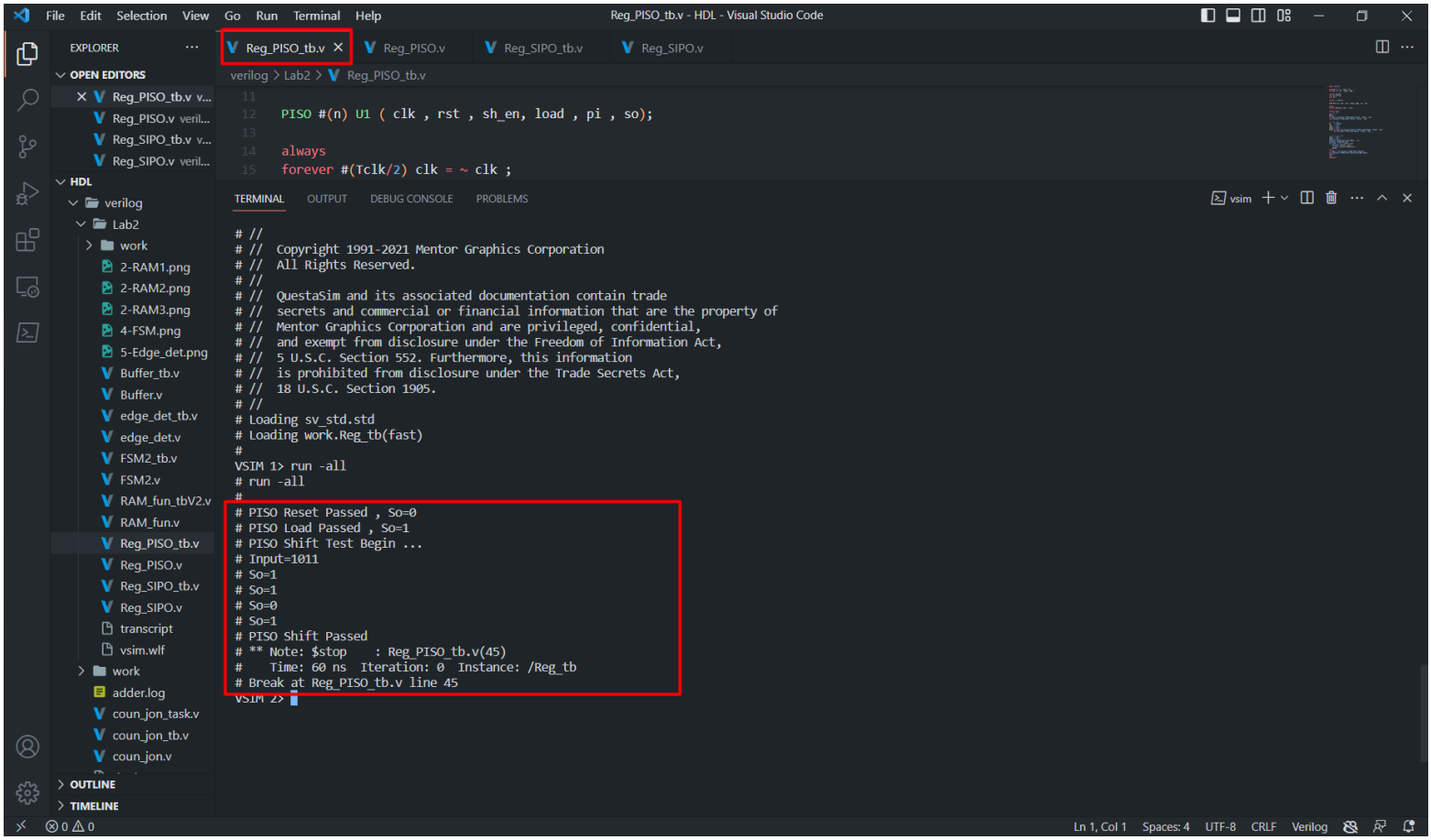
```
module PISO #(parameter n = 4) (input clk , rst , sh_en, load , [n-1:0] pi ,
output reg so);
reg [n-1:0] q ;
assign so = q[0] ;
always @(posedge clk ) begin
   if (rst==1) q = 0 ;
   else if (load==1)    q = pi ;
   else if (sh_en==1)   q = {1'b0 , q[n-1:1] };
end
endmodule
```

PISO Testbench

```
module Reg_tb;
parameter n = 4 , Tclk = 10;
reg clk = 0, rst , sh_en, load ;
reg si, so_old;
reg  [n-1:0] pi; wire so ;
integer i ,fail=0;
PISO #(n) U1 ( clk , rst , sh_en, load , pi , so);

always
forever #(Tclk/2) clk = ~ clk ;
initial begin
// test rest
rst = 1 ;
#Tclk
if (so==0) $display ("PISO Reset Passed , So=%b" , so);
else $display ("PISO Reset Failed , So=%b" , so);

// test load
pi = 4'b1011 ; rst= 1'b0 ; load= 1'b1 ; sh_en = 1'b0 ;
#Tclk if (so == pi [n-1]) $display ("PISO Load Passed , So=%b" , so);
      else $display ("PISO Load Failed , So=%b" , so);
```

```
// test shift PISO
load  = 1'b0 ;
sh_en = 1'b1 ;
$display ("PISO Shift Test Begin ...");
$display ("Input=%b",pi);
for (i=0 ; i < n ; i=i+1) begin
    $display ("So=%b ", so);
    if (so != pi [i]) fail=fail+1 ;
    #Tclk ;
end
if (fail == 0) $display ("PISO Shift Passed");
else $display ("PISO Shift Failed Fail=%0d",fail);
$stop ;
end
endmodule
```

PISO Output



SIPO

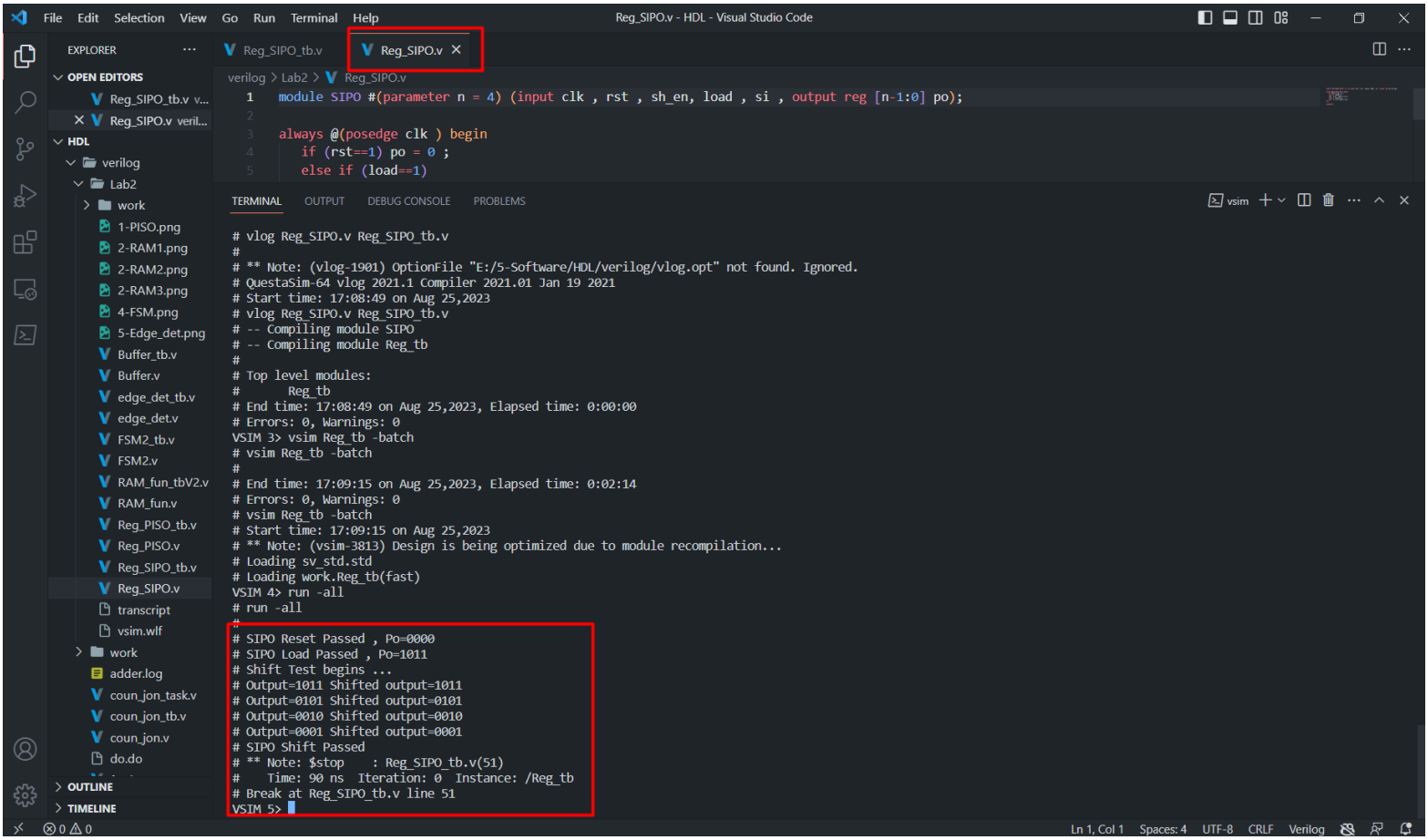
```
module SIPO #(parameter n = 4) (input clk , rst , sh_en, load , si , output reg
[n-1:0] po);
always @(posedge clk ) begin
   if (rst==1) po = 0 ;
   else if (load==1)    po = { si , po[n-1:1] };
   else if (sh_en==1)   po = { 1'b0, po[n-1:1]};
end
endmodule
```

SIPO Testbench

```
module Reg_tb;
parameter n = 4 , Tclk = 10;
reg clk = 0, rst , sh_en, load, si;
reg  [n-1:0] inputs , po_sh;
wire [n-1:0] po  ;
integer i ,fail=0;
SIPO #(n) U2 ( clk , rst , sh_en, load , si , po);
always
forever #(Tclk/2) clk = ~ clk ;

initial begin
// test rest
rst = 1 ;
#Tclk
if (po==0) $display ("SIPO Reset Passed , Po=%b",po);
else $display ("SIPO Reset Failed , Po=%b",po);

// test load
inputs = 4'b1011 ; rst = 1'b0 ; load = 1'b1 ; sh_en  = 1'b0 ;
for (i=0 ; i < n ;i=i+1) begin
si= inputs[i] ; #Tclk ;
end
if (po == inputs) $display ("SIPO Load Passed , Po=%b",po);
else $display ("SIPO Load Failed , Po=%b",po);

// test shift SIPO
load  = 1'b0 ;
sh_en = 1'b1 ;
po_sh = po;
$display ("Shift Test begins ...");
```

```
for (i=0 ; i < n ;i=i+1) begin
$display ("Output=%b Shifted output=%b",po, po_sh);
po_sh = {1'b0 , po_sh[n-1:1]} ;
#Tclk if (po != po_sh) fail=fail+1 ;
end

if (fail == 0) $display  ("SIPO Shift Passed");
else $display  ("SIPO Shift Failed");
$stop ;
end
endmodule
```

SIPO Output



2- Write Verilog to model ram block and write a function in ram module to translate the input address ranges from 16'h8000- 16'h800f to 8'h00-8'h0f as the address of the ram is 8bits. The address port of the block is 16-bit. The data_in and data_out is 8-bit.

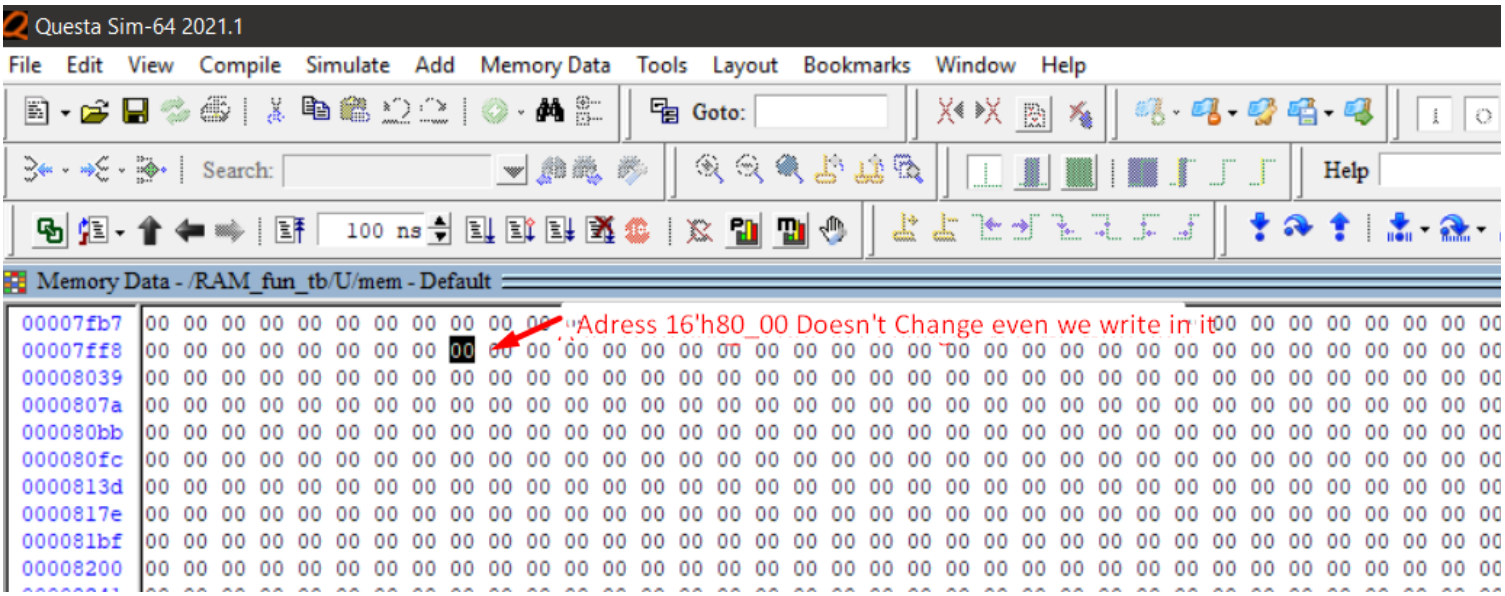
```
module RAM_fun #(parameter n = 16 ,  d = 8)
            (input rst , clk , en , wr , [d-1:0] data_in , [n-1:0] adr,
             output reg [d-1:0] data_out);

reg [d-1:0] mem [0: (2**n)-1];
reg [7:0] new_adr ;
integer i ;

function [7:0] adr_conv;
input [n-1:0] adr ;
adr_conv=  adr[7:0] ;
endfunction


always @(posedge(clk)) begin

// // Initialize
if (rst==1)
   for (i=0 ; i< 2**n ; i=i+1)
    mem[i]=0 ;

// Synchronous Read
else if (en==1)
    if (wr==1) mem[new_adr] = data_in ;
end

// asynchrounous Read
assign data_out = (en & ~ wr) ? mem[new_adr] : 0 ;

// addres convert
assign new_adr= adr_conv(adr);
endmodule
```

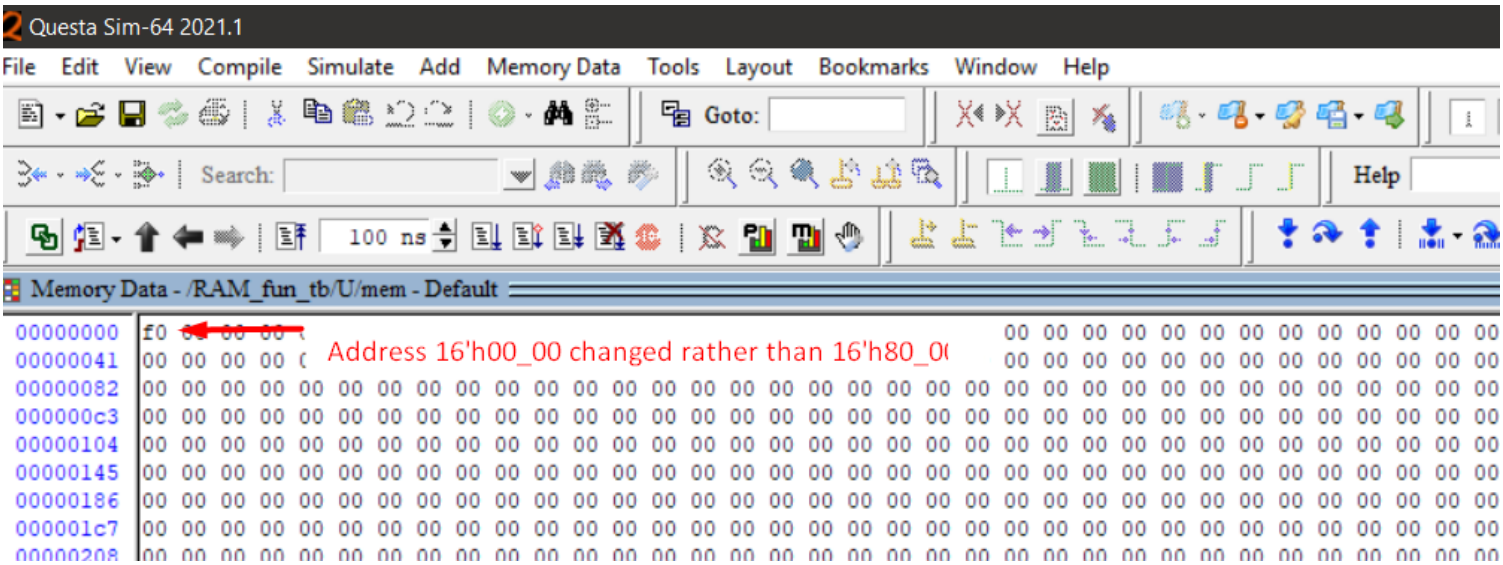
Simple Testbench

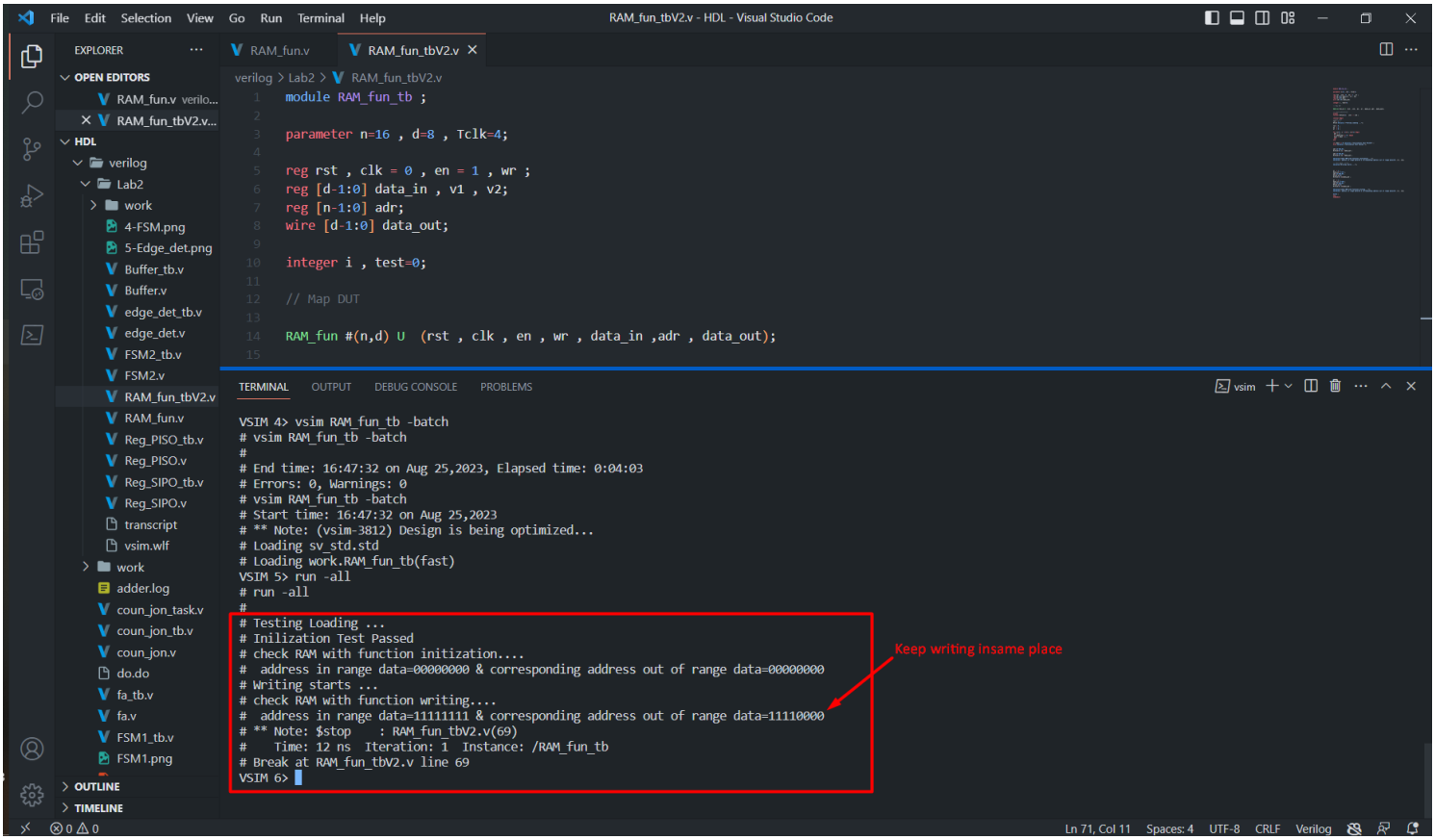
```
module RAM_fun_tb ;

parameter n=16 , d=8 , Tclk=4;

reg rst , clk = 0 , en = 1 , wr ;
reg [d-1:0] data_in , v1 , v2;
reg [n-1:0] adr;
wire [d-1:0] data_out;

integer i , test=0;

// Map DUT
RAM_fun #(n,d) U  (rst , clk , en , wr , data_in ,adr , data_out);

// clock generation
always
forever #(Tclk/2)   clk = ~ clk ;

initial begin
//test rest
rst = 1 ;
#Tclk $display ("Testing Loading ...");
rst = 0 ; en  = 1 ; wr  = 0 ;

for (i=0 ; i < 2**n ; i=i+1) begin
 adr = i ;
 if (data_out != 0 ) begin
 test = test + 1 ;
 end
end

if (test == 0) $display ("Inilization Test Passed") ;
else $display ("Inilization Test Failed ");

adr=16'h00_00 ;
#(Tclk/8) v1 = data_out ;

adr=16'h80_00 ;
#(Tclk/8) v2 = data_out ;

$display("check RAM with function initization....");
```

```
$display(" address in range data=%b & corresponding address out of range
data=%b", v1 , v2);

// test Read + write
$display("Writing starts ...");
wr   = 1 ;
data_in= 8'hff ;
adr=16'h00_00 ;
#Tclk wr=0;
#(Tclk/8) v1=data_out ;
wr   = 1 ;
data_in= 8'hf0 ;
adr=16'h80_00 ;
#Tclk wr=0;
#(Tclk/8) v2=data_out ;
$display("check RAM with function writing....");
$display(" address in range data=%b & corresponding address out of range
data=%b", v1 , v2);
$stop ;
end
endmodule
```

--Result




Questa Sim-64 2021.1
File Edit View Compile Simulate Add Memory Data Tools Layout Bookmarks Window Help
Goto:
Search:
Help
100 ns
Memory Data - /RAM_fun_tb/U/mem - Default
Address 16'h00_00 changed rather than 16'h80_0(


File Edit Selection View Go Run Terminal Help
RAM_fun_tbV2.v - HDL - Visual Studio Code
EXPLORER
OPEN EDITORS
RAM_fun.v verilo...
RAM_fun_tbV2.v...
HDL
verilog
Lab2
work
4-FSM.png
5-Edge_det.png
Buffer_tb.v
Buffer.v
edge_det_tb.v
edge_det.v
FSM2_tb.v
FSM2.v
RAM_fun_tbV2.v
RAM_fun.v
Reg_PISO_tb.v
Reg_PISO.v
Reg_SIPO_tb.v
Reg_SIPO.v
transcript
vsim.wlf
work
adder.log
coun_jon_task.v
coun_jon_tb.v
coun_jon.v
do.do
fa_tb.v
fa.v
FSM1_tb.v
FSM1.png
OUTLINE
TIMELINE
RAM_fun.v
RAM_fun_tbV2.v
verilog > Lab2 > RAM_fun_tbV2.v
module RAM_fun_tb ;
parameter n=16 , d=8 , Tclk=4;
reg rst , clk = 0 , en = 1 , wr ;
reg [d-1:0] data_in , v1 , v2;
reg [n-1:0] adr;
wire [d-1:0] data_out;
integer i , test=0;
// Map DUT
RAM_fun #(n,d) U (rst , clk , en , wr , data_in ,adr , data_out);
TERMINAL OUTPUT DEBUG CONSOLE PROBLEMS
vsim
VSIM 4> vsim RAM_fun_tb -batch
vsim RAM_fun_tb -batch
#
End time: 16:47:32 on Aug 25,2023, Elapsed time: 0:04:03
Errors: 0, Warnings: 0
vsim RAM_fun_tb -batch
Start time: 16:47:32 on Aug 25,2023
** Note: (vsim-3812) Design is being optimized...
Loading sv_std.std
Loading work.RAM_fun_tb(fast)
VSIM 5> run -all
run -all
#
Testing Loading ...
Inilization Test Passed
check RAM with function initization....
address in range data=00000000 & corresponding address out of range data=00000000
Writing starts ...
check RAM with function writing....
address in range data=11111111 & corresponding address out of range data=11110000
** Note: $stop : RAM_fun_tbV2.v(69)
Time: 12 ns Iteration: 1 Instance: /RAM_fun_tb
Break at RAM_fun_tbV2.v line 69
VSIM 6>
Keep writing insame place
Ln 71, Col 11 Spaces: 4 UTF-8 CRLF Verilog

3- Write a Verilog code to model simple memory buffer:

- **The input signals: clk, rst, w, r, 16-bit data_in.**
- **The output signals: 16-bit data_out.**
- **The internal writer pointer and read pointer of the memory buffer should wrap.**

Code

```
module FIFO #(parameter depth = 4 , width = 16) (input clk , rst , w , r ,
[width-1:0] data_in , output reg [width-1:0] data_out);

parameter MSB = $clog2(depth)-1;
reg  [width-1:0] stack [depth -1 : 0 ] ;
reg [MSB: 0] r_p = 0 , w_p = 0 ;
wire full, empty ;
reg overflow =0;
integer ocup = 0;

assign empty = (ocup ==  0)   ? 1'b1 : 1'b0 ;
assign full  = (ocup == depth)? 1'b1 : 1'b0 ;

task push () ;
begin
stack[w_p] = data_in ;
ocup = ocup + 1 ;
if (w_p==(depth-1)) begin
overflow = 1'b1;
w_p = 0 ;
end
else w_p =  w_p + 1 ;
end
endtask


task pop ();
begin
data_out = stack[r_p] ;
ocup = ocup-1 ;
if (r_p==(depth-1)) begin
r_p = 0 ; overflow =1'b0;
end
else r_p = r_p + 1 ;
end
endtask
```

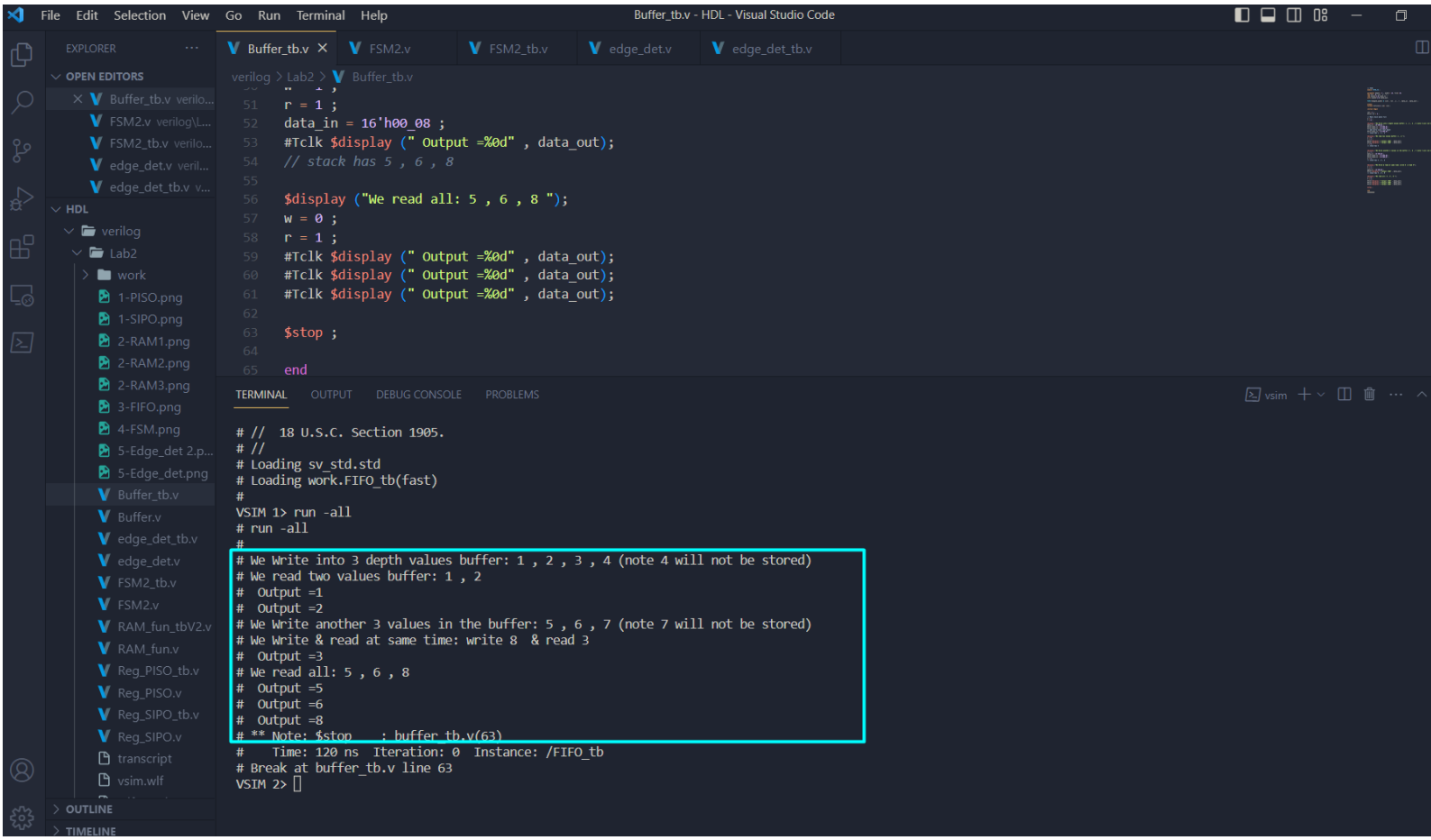
```
always @(posedge clk ) begin
if (rst == 1) begin
r_p  = 0 ;
w_p = 0 ;
ocup = 0 ;
end
else if (w==1 & r==0) begin
//  Not Full & NO Overflow or less than read at the Overflow
 if ((!full && !overflow) || (overflow && (w_p < r_p)) )
     push ;   end

else if (w==0 & r==1) begin
 if ((!overflow && (r_p < w_p)) || (!empty && overflow) )
     pop ; end

else if (w==1 & r==1) begin
  if (empty)
  data_out = data_in ;
  else
  begin
  pop; push;
  end
end
end
endmodule
```

Testbench

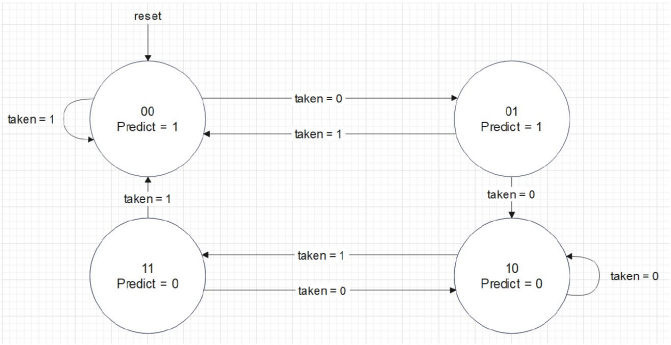
```
// FIFO
module FIFO_tb ;

parameter depth = 3 , width = 16, Tclk= 10;
reg clk = 0, rst , w , r ;
reg [width-1:0] data_in ;
wire [width-1:0] data_out;

FIFO #(depth,width) U (clk , rst , w , r , data_in , data_out) ;

always
forever #(Tclk/2) clk= ~clk ;

initial begin

rst = 1 ;
```

```
#Tclk rst = 0 ;

// Make stack above full
w = 1 ; r = 0 ;
$display ("We Write into 3 depth values buffer: 1 , 2 , 3 , 4 (note 4 will not be
stored)");
data_in = 16'h00_01 ;
#Tclk data_in = 16'h00_02 ;
#Tclk data_in = 16'h00_03 ;
// add extra value than depth
#Tclk data_in = 16'h00_04 ;
// stack has 1 , 2 , 3

$display ("We read two values buffer: 1 , 2 ");
w = 0 ;
r = 1 ;
#Tclk $display (" Output =%0d" , data_out);
#Tclk $display (" Output =%0d" , data_out);
r = 0 ;
// stack has 3
$display ("We Write another 3 values in the buffer: 5 , 6 , 7 (note 7 will not be
stored)");
w = 1 ;
data_in = 16'h00_05 ;
#Tclk data_in = 16'h00_06 ;
#Tclk data_in = 16'h00_07 ;
w = 0 ;
// stack has 3 , 5 , 6
$display ("We Write & read at same time: write 8  & read 3");
w = 1 ;
r = 1 ;
data_in = 16'h00_08 ;
#Tclk $display (" Output =%0d" , data_out);
// stack has 5 , 6 , 8
$display ("We read all: 5 , 6 , 8 ");
w = 0 ;
r = 1 ;
#Tclk $display (" Output =%0d" , data_out);
#Tclk $display (" Output =%0d" , data_out);
#Tclk $display (" Output =%0d" , data_out);
$stop ;
end
endmodule
```

Output


File Edit Selection View Go Run Terminal Help
Buffer_tb.v - HDL - Visual Studio Code
EXPLORER
OPEN EDITORS
Buffer_tb.v verilo...
FSM2.v verilog\L...
FSM2_tb.v verilo...
edge_det.v veril...
edge_det_tb.v v...
HDL
verilog
Lab2
work
1-PISO.png
1-SIPO.png
2-RAM1.png
2-RAM2.png
2-RAM3.png
3-FIFO.png
4-FSM.png
5-Edge_det 2.p...
5-Edge_det.png
Buffer_tb.v
Buffer.v
edge_det_tb.v
edge_det.v
FSM2_tb.v
FSM2.v
RAM_fun_tbV2.v
RAM_fun.v
Reg_PISO_tb.v
Reg_PISO.v
Reg_SIPO_tb.v
Reg_SIPO.v
transcript
vsim.wlf
OUTLINE
TIMELINE
Buffer_tb.v
FSM2.v
FSM2_tb.v
edge_det.v
edge_det_tb.v
verilog > Lab2 > Buffer_tb.v
51 r = 1 ;
52 data_in = 16'h00_08 ;
53 #Tclk $display (" Output =%0d" , data_out);
54 // stack has 5 , 6 , 8
55
56 $display ("We read all: 5 , 6 , 8 ");
57 w = 0 ;
58 r = 1 ;
59 #Tclk $display (" Output =%0d" , data_out);
60 #Tclk $display (" Output =%0d" , data_out);
61 #Tclk $display (" Output =%0d" , data_out);
62
63 $stop ;
64
65 end
TERMINAL
OUTPUT
DEBUG CONSOLE
PROBLEMS
vsim
// 18 U.S.C. Section 1905.
//
Loading sv_std.std
Loading work.FIFO_tb(fast)
#
VSIM 1> run -all
run -all
#
We Write into 3 depth values buffer: 1 , 2 , 3 , 4 (note 4 will not be stored)
We read two values buffer: 1 , 2
Output =1
Output =2
We Write another 3 values in the buffer: 5 , 6 , 7 (note 7 will not be stored)
We Write & read at same time: write 8 & read 3
Output =3
We read all: 5 , 6 , 8
Output =5
Output =6
Output =8
** Note: $stop : buffer_tb.v(63)
Time: 120 ns Iteration: 0 Instance: /FIFO_tb
Break at buffer_tb.v line 63
VSIM 2>

4-Write a Verilog code that implement this state machine. Using Moore.

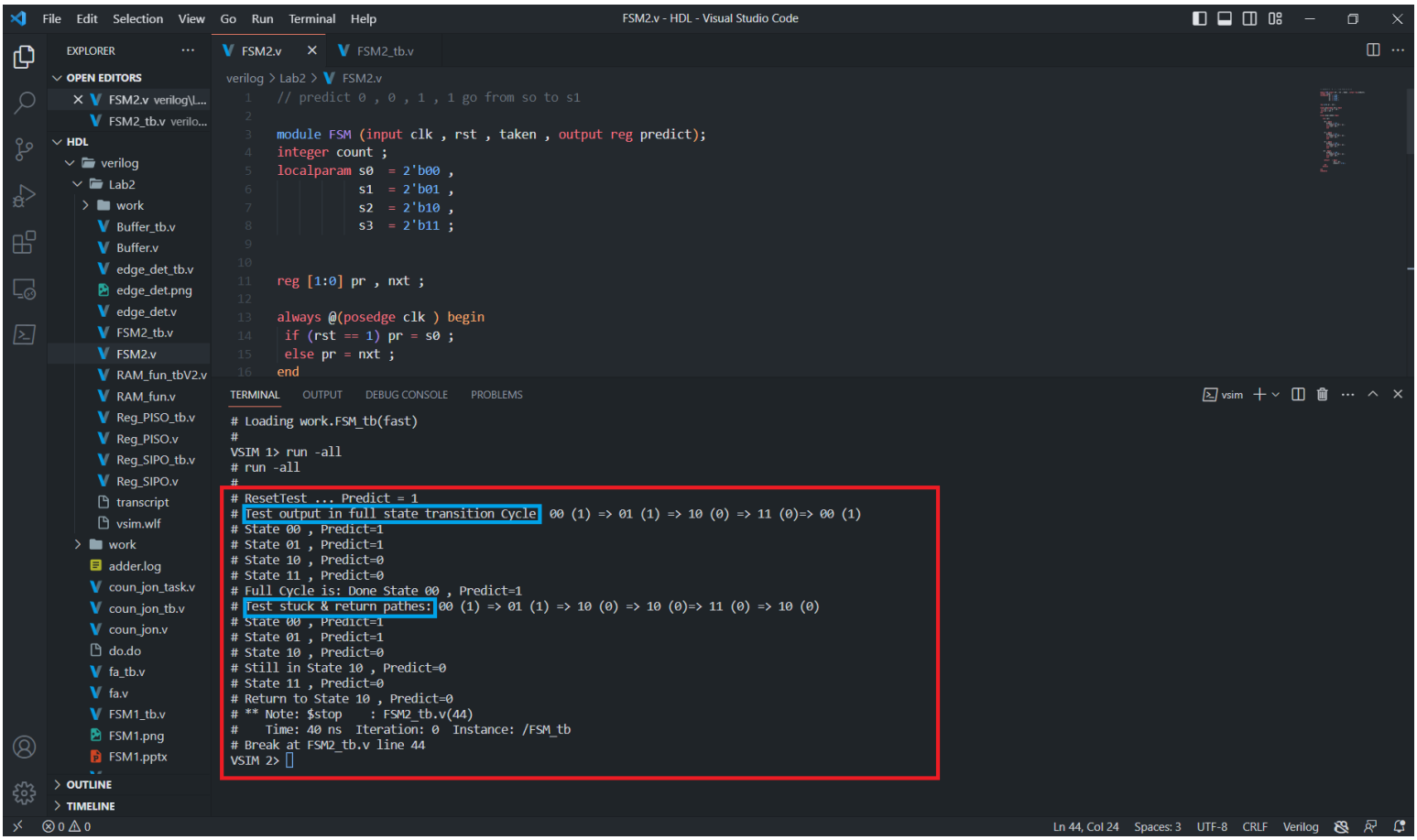


```
module FSM (input clk , rst , taken , output reg predict);
integer count ;
localparam s0  = 2'b00 , s1  = 2'b01 , s2  = 2'b10 , s3  = 2'b11 ;
reg [1:0] pr , nxt ;
always @(posedge clk ) begin
 if (rst == 1) pr = s0 ;
 else pr = nxt ; end
always @(pr,taken) begin
   case (pr)
   s0 : begin
        predict = 1'b1 ;
        if (taken==0) nxt = s1 ;
        else nxt = s0 ; end
    s1 : begin
        predict = 1'b1 ;
        if (taken==0) nxt = s2 ;
        else nxt = s0 ; end
     s2 : begin
        predict = 1'b0 ;
        if (taken==0) nxt = s2 ;
        else nxt = s3 ; end
    s3 : begin
        predict = 1'b0 ;
        if (taken==0) nxt = s2 ;
        else nxt = s0 ; end
     default : begin
               nxt = s0 ; predict = 1'b1 ; end
   endcase
end
endmodule
```

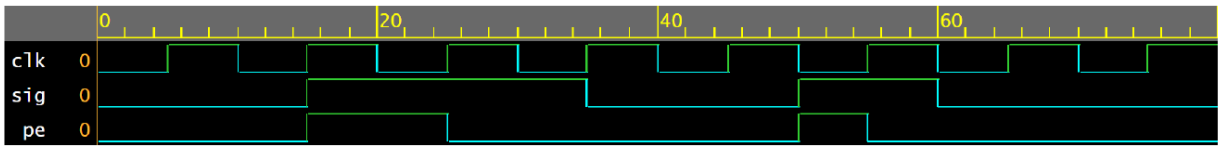
Test bench

```
module FSM_tb;
reg clk = 0 , rst , taken ; wire predict ; parameter Tclk = 4 ;
FSM U (clk , rst , taken , predict) ;

always
forever #(Tclk/2) clk = ~ clk ;

initial begin

rst = 1 ; #Tclk $display ("ResetTest ... Predict = %b", predict) ;

rst = 0 ;

$display ("Test output in full state transition Cycle: 00 (1) => 01 (1) => 10 (0)
=> 11 (0)=> 00 (1) ") ;
$display ("State 00 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("State 01 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("State 10 , Predict=%b" , predict) ;
taken = 1'b1;
#Tclk $display ("State 11 , Predict=%b" , predict) ;
taken = 1'b1;
#Tclk $display ("Full Cycle is: Done State 00 , Predict=%b" , predict) ;

$display ("Test stuck & return pathes: 00 (1) => 01 (1) => 10 (0) => 10 (0)=> 11
(0) => 10 (0)") ;
$display ("State 00 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("State 01 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("State 10 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("Still in State 10 , Predict=%b" , predict) ;
taken = 1'b1;
#Tclk $display ("State 11 , Predict=%b" , predict) ;
taken = 1'b0;
#Tclk $display ("Return to State 10 , Predict=%b" , predict) ;
$stop ;
end
endmodule
```

Result


File Edit Selection View Go Run Terminal Help
FSM2.v - HDL - Visual Studio Code
EXPLORER
OPEN EDITORS
FSM2.v verilog\L...
FSM2_tb.v verilo...
HDL
verilog
Lab2
work
Buffer_tb.v
Buffer.v
edge_det_tb.v
edge_det.png
edge_det.v
FSM2_tb.v
FSM2.v
RAM_fun_tbV2.v
RAM_fun.v
Reg_PISO_tb.v
Reg_PISO.v
Reg_SIPO_tb.v
Reg_SIPO.v
transcript
vsim.wlf
work
adder.log
coun_jon_task.v
coun_jon_tb.v
coun_jon.v
do.do
fa_tb.v
fa.v
FSM1_tb.v
FSM1.png
FSM1.pptx
OUTLINE
TIMELINE
FSM2.v
FSM2_tb.v
verilog > Lab2 > FSM2.v
1 // predict 0 , 0 , 1 , 1 go from so to s1
2
3 module FSM (input clk , rst , taken , output reg predict);
4 integer count ;
5 localparam s0 = 2'b00 ,
6 s1 = 2'b01 ,
7 s2 = 2'b10 ,
8 s3 = 2'b11 ;
9
10
11 reg [1:0] pr , nxt ;
12
13 always @(posedge clk) begin
14 if (rst == 1) pr = s0 ;
15 else pr = nxt ;
16 end
TERMINAL OUTPUT DEBUG CONSOLE PROBLEMS
vsim
Loading work.FSM_tb(fast)
#
VSIM 1> run -all
run -all
#
ResetTest ... Predict = 1
Test output in full state transition Cycle 00 (1) => 01 (1) => 10 (0) => 11 (0)=> 00 (1)
State 00 , Predict=1
State 01 , Predict=1
State 10 , Predict=0
State 11 , Predict=0
Full Cycle is: Done State 00 , Predict=1
Test stuck & return pathes: 00 (1) => 01 (1) => 10 (0) => 10 (0)=> 11 (0) => 10 (0)
State 00 , Predict=1
State 01 , Predict=1
State 10 , Predict=0
Still in State 10 , Predict=0
State 11 , Predict=0
Return to State 10 , Predict=0
** Note: $stop : FSM2_tb.v(44)
Time: 40 ns Iteration: 0 Instance: /FSM_tb
Break at FSM2_tb.v line 44
VSIM 2>
Ln 44, Col 24 Spaces: 3 UTF-8 CRLF Verilog

5-Write a Verilog code that implements a positive edge detector.

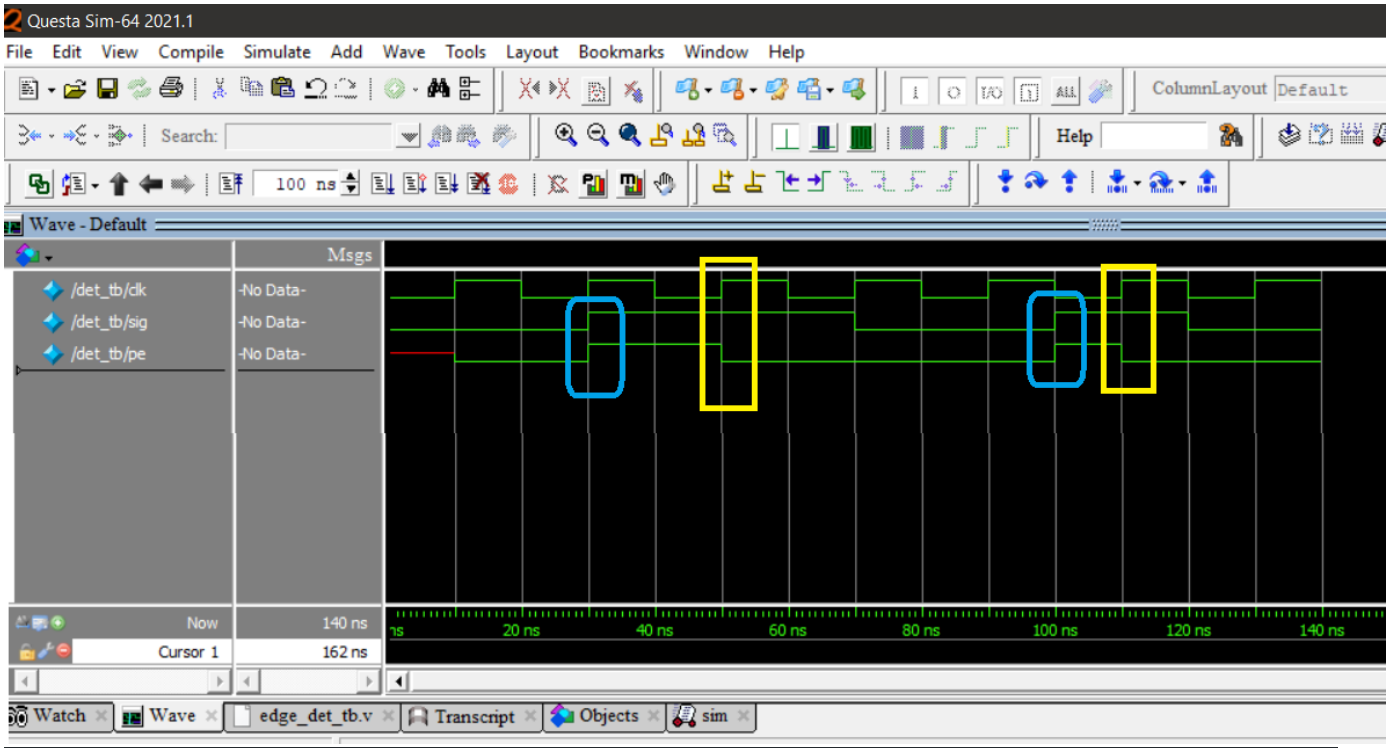


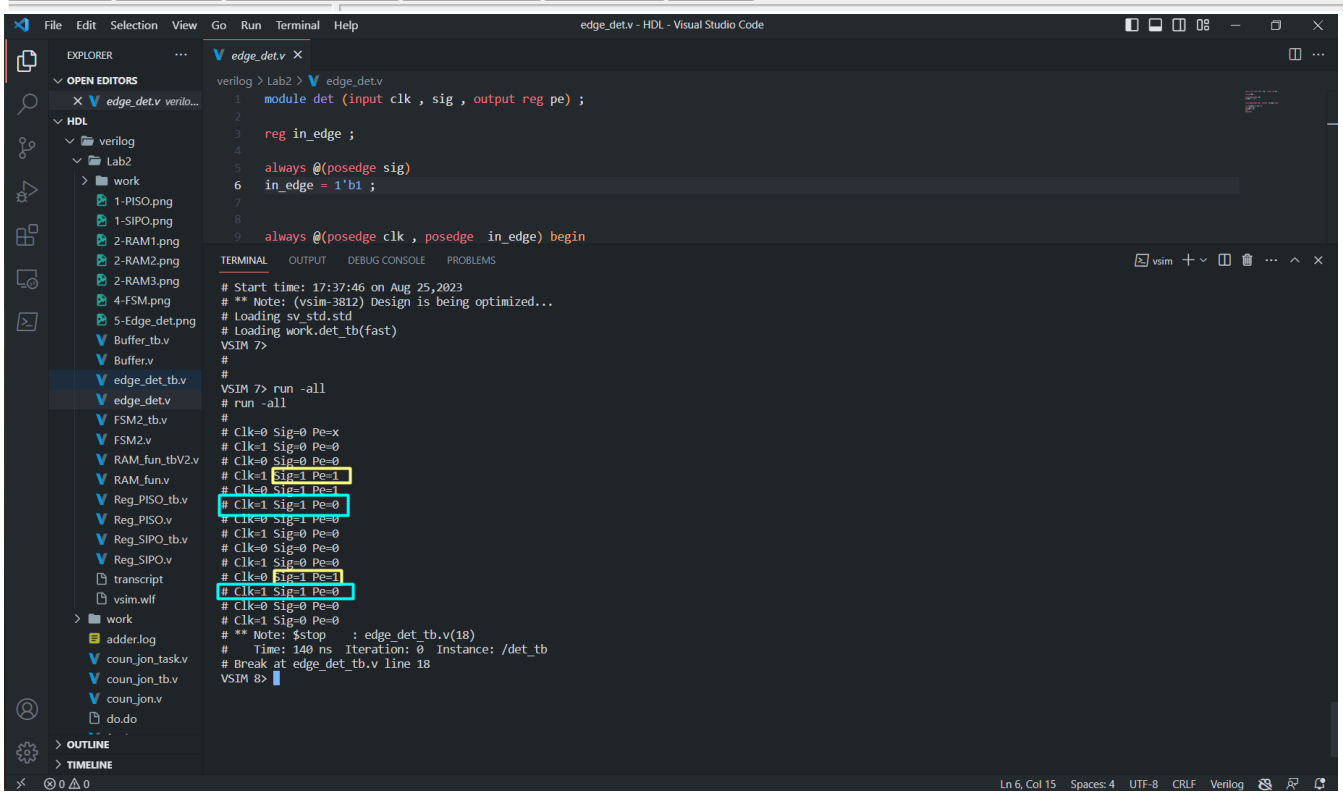
```
module det (input clk , sig , output reg pe) ;
reg in_edge ;
always @(posedge sig)
in_edge = 1'b1 ;
always @(posedge clk , posedge  in_edge) begin
if (in_edge==1) pe=1'b1;
else pe=1'b0 ;
in_edge=1'b0;
end
endmodule
```

Test bench

```
module det_tb ;
reg clk = 0 , sig =0;
wire  pe ;
parameter T = 10 ;

det U (clk , sig , pe) ;

always
forever #(T) clk = ~clk ;

initial begin
$monitor ("Clk=%b Sig=%b Pe=%b", clk , sig , pe);
#(3*T) sig = 1'b1 ;
#(4*T) sig = 1'b0 ;
#(3*T) sig = 1'b1 ;
#(2*T) sig = 1'b0 ;
#(2*T) $stop ;
end
endmodule
```

--Result


Questa Sim-64 2021.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
/det_tb/clk
/det_tb/sig
/det_tb/pe
-No Data-
-No Data-
-No Data-
Now
140 ns
Cursor 1
162 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
Watch
Wave
edge_det_tb.v
Transcript
Objects
sim


edge_det.v - HDL - Visual Studio Code
EXPLORER
OPEN EDITORS
HDL
verilog
Lab2
work
1-PISO.png
1-SIPO.png
2-RAM1.png
2-RAM2.png
2-RAM3.png
4-FSM.png
5-Edge_det.png
Buffer_tb.v
Buffer.v
edge_det_tb.v
edge_det.v
FSM2_tb.v
FSM2.v
RAM_fun_tbV2.v
RAM_fun.v
Reg_PISO_tb.v
Reg_PISO.v
Reg_SIPO_tb.v
Reg_SIPO.v
transcript
vsim.wlf
work
adder.log
coun_jon_task.v
coun_jon_tb.v
coun_jon.v
do.do
OUTLINE
TIMELINE
verilog > Lab2 > edge_det.v
module det (input clk , sig , output reg pe) ;
reg in_edge ;
always @(posedge sig)
in_edge = 1'b1 ;
always @(posedge clk , posedge in_edge) begin
TERMINAL
OUTPUT
DEBUG CONSOLE
PROBLEMS
Start time: 17:37:46 on Aug 25,2023
** Note: (vsim-3812) Design is being optimized...
Loading sv_std.std
Loading work.det_tb(fast)
VSIM 7>
VSIM 7> run -all
run -all
Clk=0 Sig=0 Pe=x
Clk=1 Sig=0 Pe=0
Clk=0 Sig=0 Pe=0
Clk=1 Sig=1 Pe=1
Clk=0 Sig=1 Pe=1
Clk=1 Sig=1 Pe=0
Clk=0 Sig=1 Pe=0
Clk=1 Sig=0 Pe=0
Clk=0 Sig=0 Pe=0
Clk=1 Sig=0 Pe=0
Clk=0 Sig=1 Pe=1
Clk=1 Sig=1 Pe=0
Clk=0 Sig=0 Pe=0
Clk=1 Sig=0 Pe=0
** Note: $stop : edge_det_tb.v(18)
Time: 140 ns Iteration: 0 Instance: /det_tb
Break at edge_det_tb.v line 18
VSIM 8>
Ln 6, Col 15
Spaces: 4
UTF-8
CRLF
Verilog