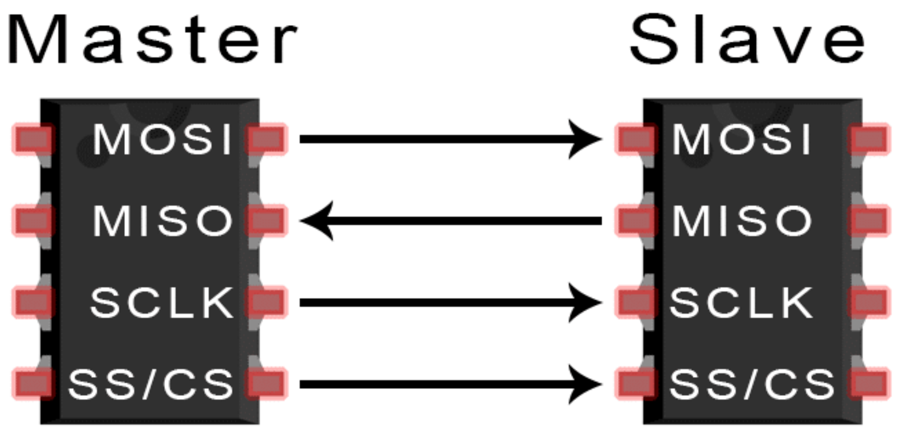
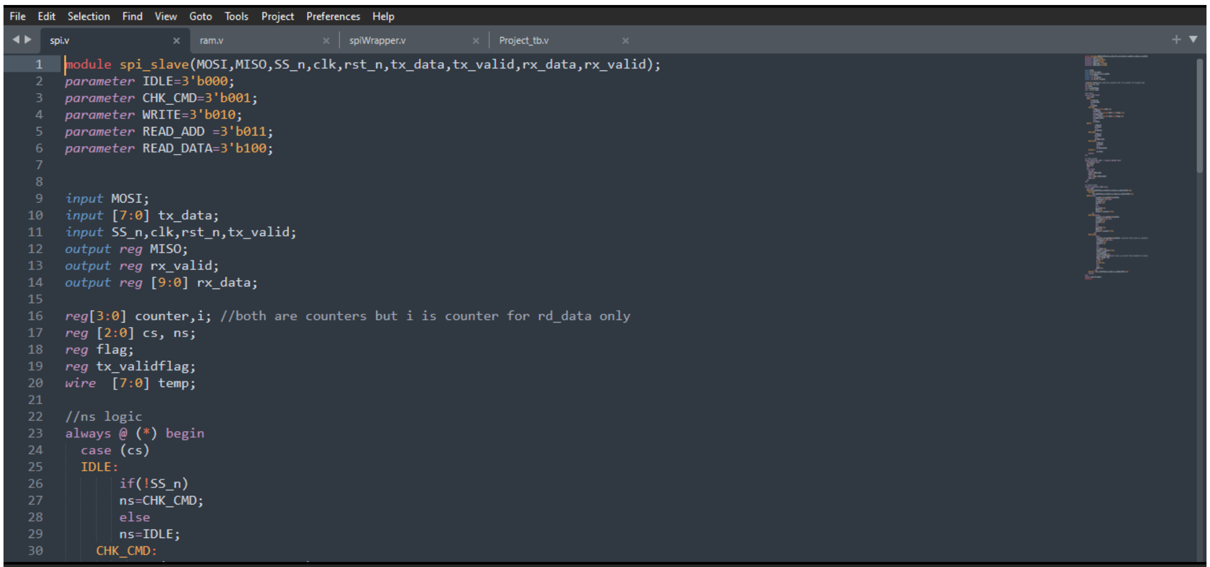
SPI PROJECT



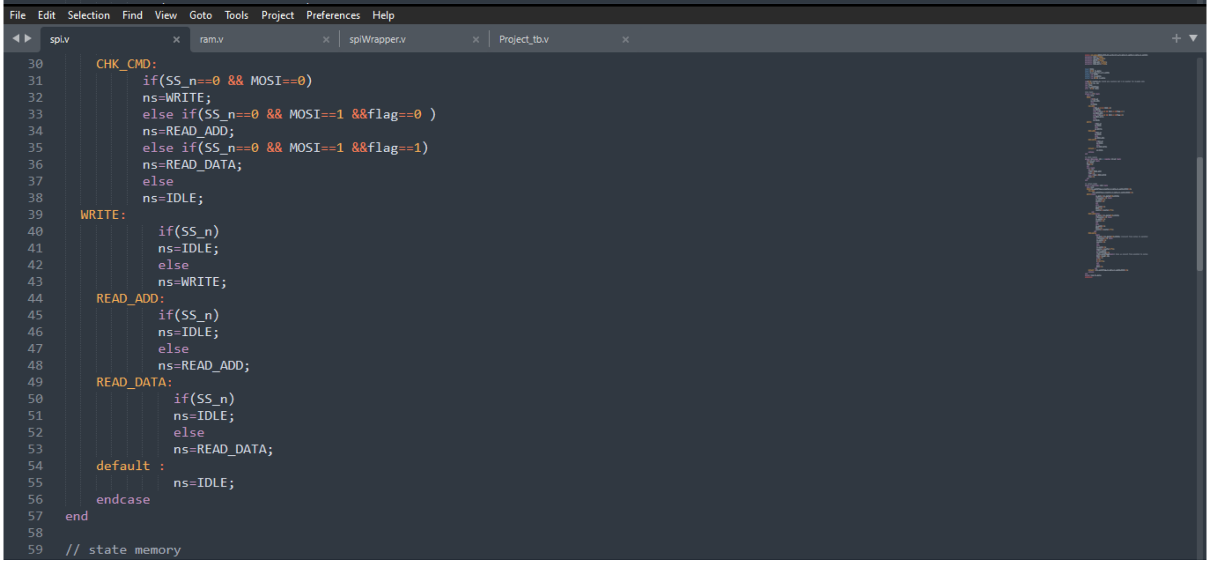
OCTOBER 16, 2022

KAREEM YAHIA MOHAMED MOHAMED

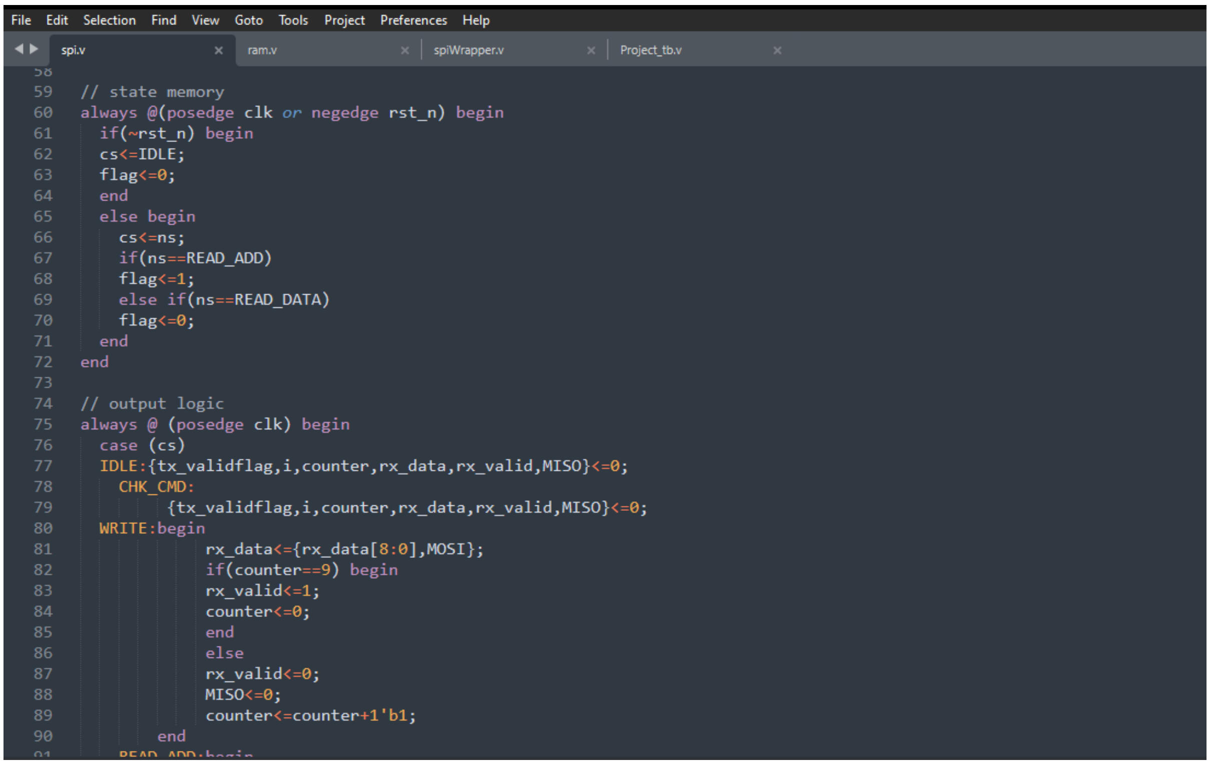
1-SPI CODE


File Edit Selection Find View Goto Tools Project Preferences Help
spi.v
ram.v
spiWrapper.v
Project_tb.v

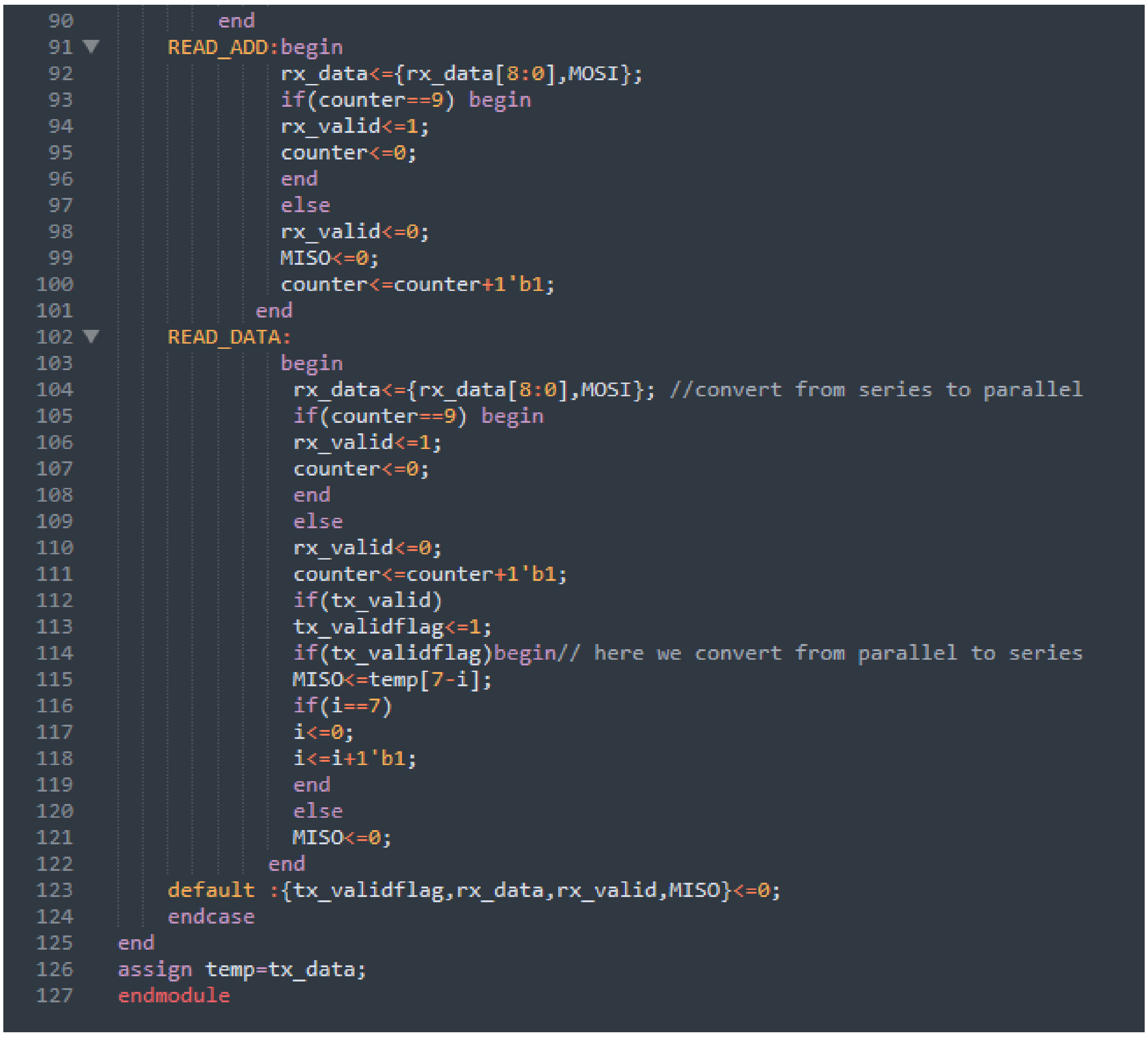
```
module spi_slave(MOSI,MISO,SS_n,clk,rst_n,tx_data,tx_valid,rx_data,rx_valid);
parameter IDLE=3'b000;
parameter CHK_CMD=3'b001;
parameter WRITE=3'b010;
parameter READ_ADD =3'b011;
parameter READ_DATA=3'b100;


input MOSI;
input [7:0] tx_data;
input SS_n,clk,rst_n,tx_valid;
output reg MISO;
output reg rx_valid;
output reg [9:0] rx_data;

reg[3:0] counter,i; //both are counters but i is counter for rd_data only
reg [2:0] cs, ns;
reg flag;
reg tx_validflag;
wire  [7:0] temp;

//ns logic
always @ (*) begin
  case (cs)
  IDLE:
       if(!SS_n)
       ns=CHK_CMD;
       else
       ns=IDLE;
    CHK_CMD:
```


File Edit Selection Find View Goto Tools Project Preferences Help
spi.v
ram.v
spiWrapper.v
Project_tb.v

```
    CHK_CMD:
          if(SS_n==0 && MOSI==0)
          ns=WRITE;
          else if(SS_n==0 && MOSI==1 &&flag==0 )
          ns=READ_ADD;
          else if(SS_n==0 && MOSI==1 &&flag==1)
          ns=READ_DATA;
          else
          ns=IDLE;
  WRITE:
            if(SS_n)
            ns=IDLE;
            else
            ns=WRITE;
    READ_ADD:
            if(SS_n)
            ns=IDLE;
            else
            ns=READ_ADD;
    READ_DATA:
              if(SS_n)
              ns=IDLE;
              else
              ns=READ_DATA;
    default :
              ns=IDLE;
    endcase
end

// state memory
```

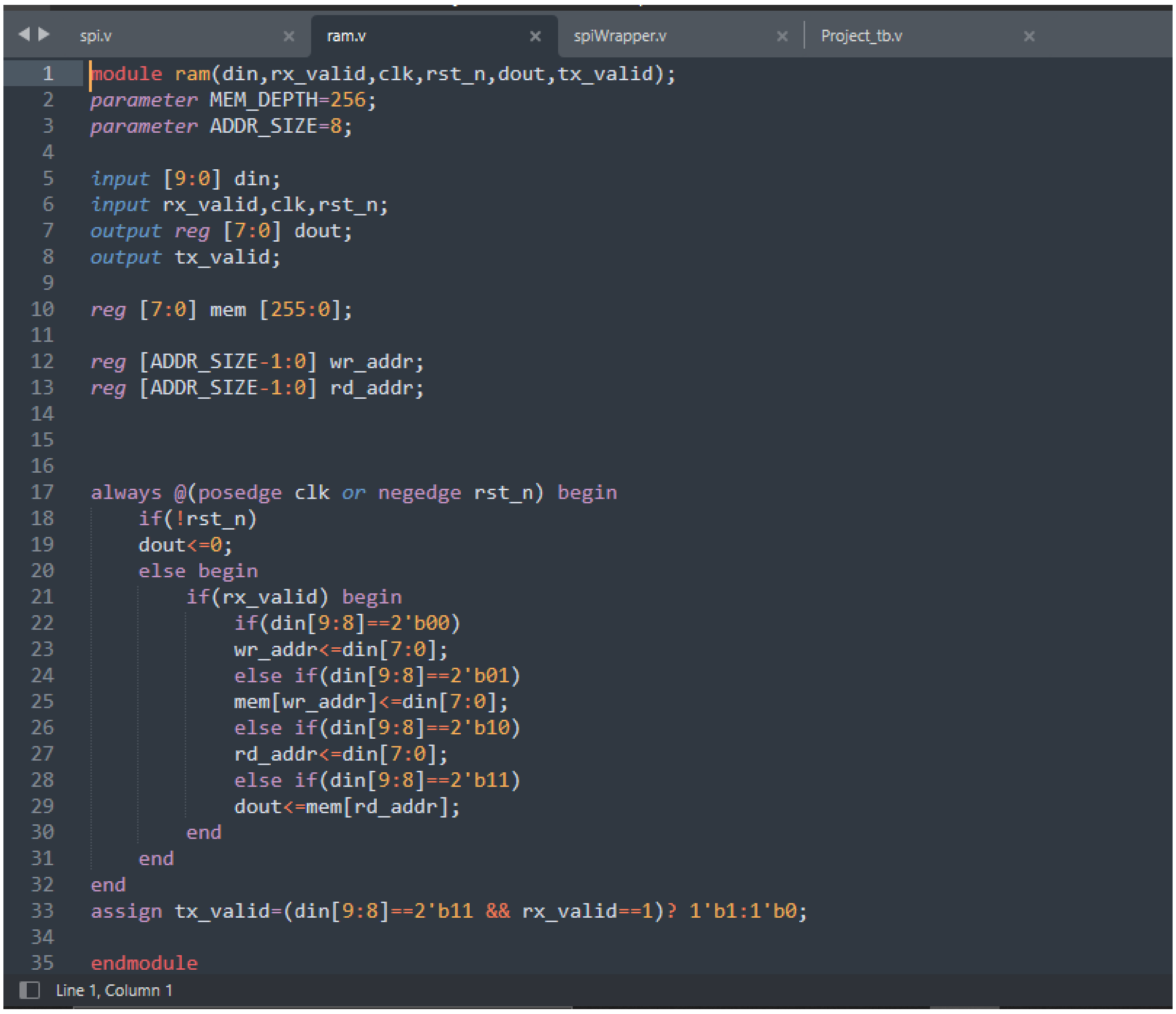


```
// state memory
always @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
  cs<=IDLE;
  flag<=0;
  end
  else begin
    cs<=ns;
    if(ns==READ_ADD)
    flag<=1;
    else if(ns==READ_DATA)
    flag<=0;
  end
end

// output logic
always @ (posedge clk) begin
  case (cs)
  IDLE:{tx_validflag,i,counter,rx_data,rx_valid,MISO}<=0;
    CHK_CMD:
         {tx_validflag,i,counter,rx_data,rx_valid,MISO}<=0;
  WRITE:begin
             rx_data<={rx_data[8:0],MOSI};
             if(counter==9) begin
             rx_valid<=1;
             counter<=0;
             end
             else
             rx_valid<=0;
             MISO<=0;
             counter<=counter+1'b1;
        end
```



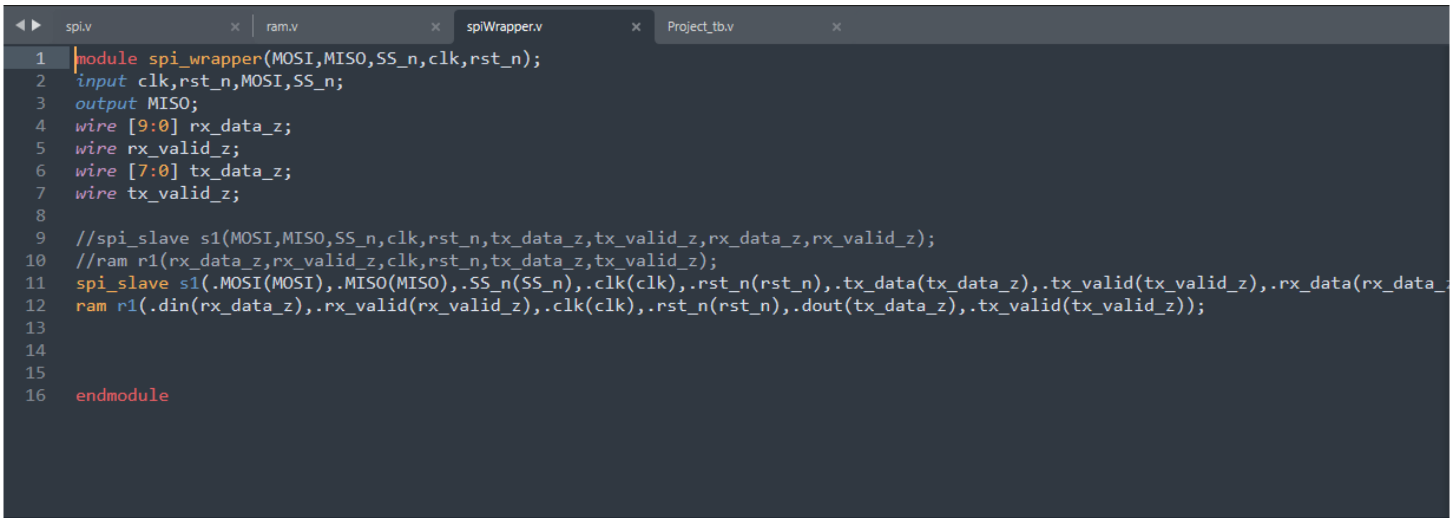
```
           end
    READ_ADD:begin
             rx_data<={rx_data[8:0],MOSI};
             if(counter==9) begin
             rx_valid<=1;
             counter<=0;
             end
             else
             rx_valid<=0;
             MISO<=0;
             counter<=counter+1'b1;
           end
    READ_DATA:
             begin
              rx_data<={rx_data[8:0],MOSI}; //convert from series to parallel
              if(counter==9) begin
              rx_valid<=1;
              counter<=0;
              end
              else
              rx_valid<=0;
              counter<=counter+1'b1;
              if(tx_valid)
              tx_validflag<=1;
              if(tx_validflag)begin// here we convert from parallel to series
              MISO<=temp[7-i];
              if(i==7)
              i<=0;
              i<=i+1'b1;
              end
              else
              MISO<=0;
            end
    default :{tx_validflag,rx_data,rx_valid,MISO}<=0;
    endcase
end
assign temp=tx_data;
endmodule
```

2-RAM CODE



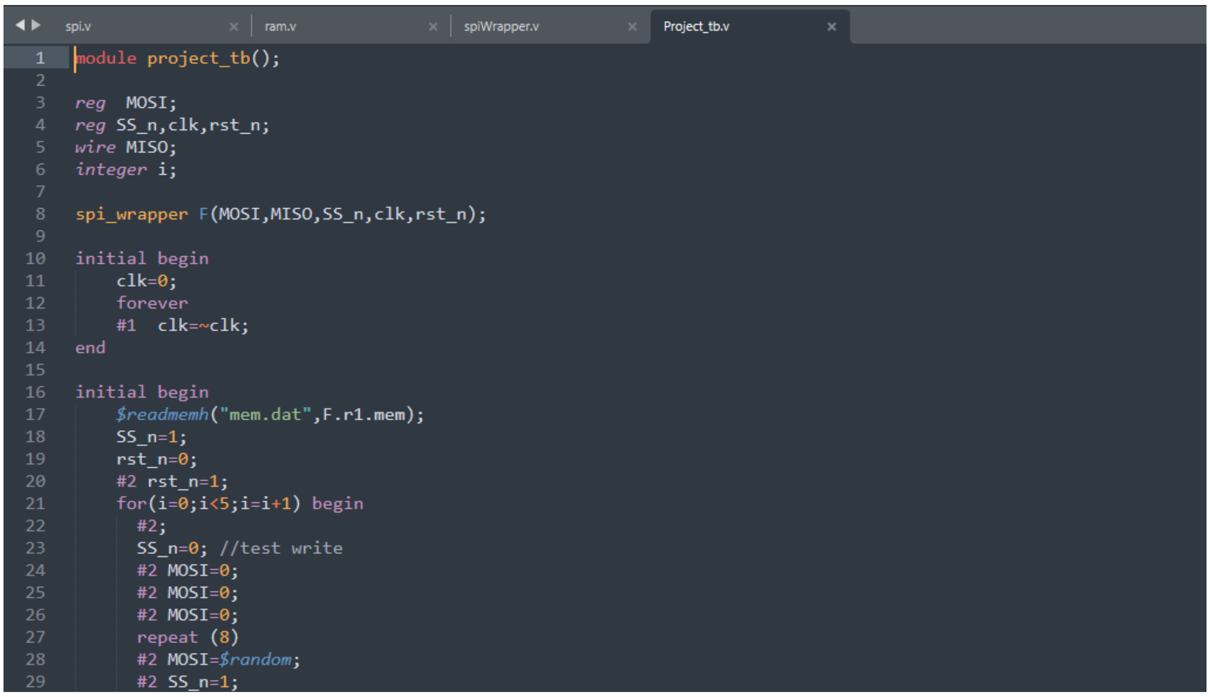
```verilog
module ram(din,rx_valid,clk,rst_n,dout,tx_valid);
parameter MEM_DEPTH=256;
parameter ADDR_SIZE=8;

input [9:0] din;
input rx_valid,clk,rst_n;
output reg [7:0] dout;
output tx_valid;

reg [7:0] mem [255:0];

reg [ADDR_SIZE-1:0] wr_addr;
reg [ADDR_SIZE-1:0] rd_addr;



always @(posedge clk or negedge rst_n) begin
    if(!rst_n)
    dout<=0;
    else begin
        if(rx_valid) begin
            if(din[9:8]==2'b00)
            wr_addr<=din[7:0];
            else if(din[9:8]==2'b01)
            mem[wr_addr]<=din[7:0];
            else if(din[9:8]==2'b10)
            rd_addr<=din[7:0];
            else if(din[9:8]==2'b11)
            dout<=mem[rd_addr];
        end
    end
end
assign tx_valid=(din[9:8]==2'b11 && rx_valid==1)? 1'b1:1'b0;

endmodule
```

3-SPI Wrapper



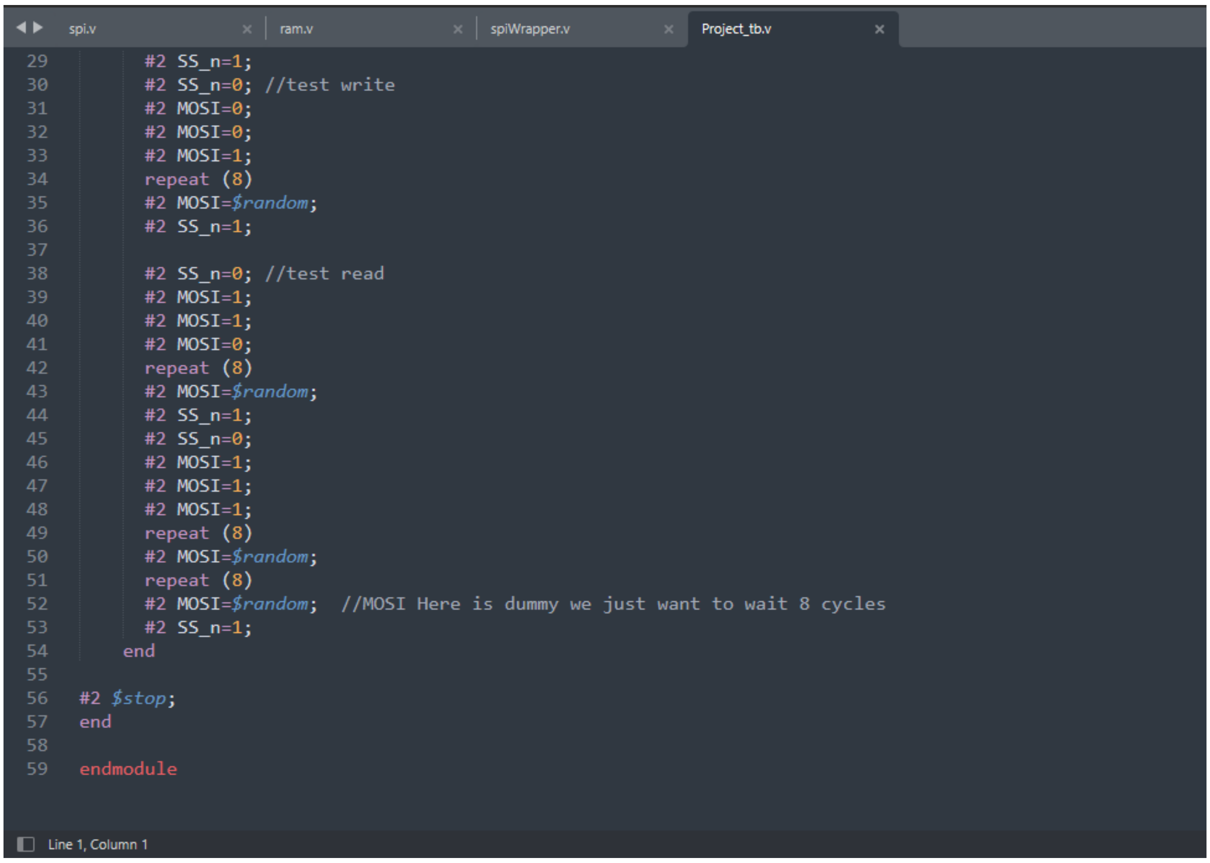
```verilog
module spi_wrapper(MOSI,MISO,SS_n,clk,rst_n);
input clk,rst_n,MOSI,SS_n;
output MISO;
wire [9:0] rx_data_z;
wire rx_valid_z;
wire [7:0] tx_data_z;
wire tx_valid_z;

//spi_slave s1(MOSI,MISO,SS_n,clk,rst_n,tx_data_z,tx_valid_z,rx_data_z,rx_valid_z);
//ram r1(rx_data_z,rx_valid_z,clk,rst_n,tx_data_z,tx_valid_z);
spi_slave s1(.MOSI(MOSI),.MISO(MISO),.SS_n(SS_n),.clk(clk),.rst_n(rst_n),.tx_data(tx_data_z),.tx_valid(tx_valid_z),.rx_data(rx_data_
ram r1(.din(rx_data_z),.rx_valid(rx_valid_z),.clk(clk),.rst_n(rst_n),.dout(tx_data_z),.tx_valid(tx_valid_z));



endmodule
```

4-Project testbench



spi.v ram.v spiWrapper.v Project_tb.v

```verilog
module project_tb();

reg  MOSI;
reg SS_n,clk,rst_n;
wire MISO;
integer i;

spi_wrapper F(MOSI,MISO,SS_n,clk,rst_n);

initial begin
    clk=0;
    forever
    #1  clk=~clk;
end

initial begin
    $readmemh("mem.dat",F.r1.mem);
    SS_n=1;
    rst_n=0;
    #2 rst_n=1;
    for(i=0;i<5;i=i+1) begin
      #2;
      SS_n=0; //test write
      #2 MOSI=0;
      #2 MOSI=0;
      #2 MOSI=0;
      repeat (8)
      #2 MOSI=$random;
      #2 SS_n=1;
```

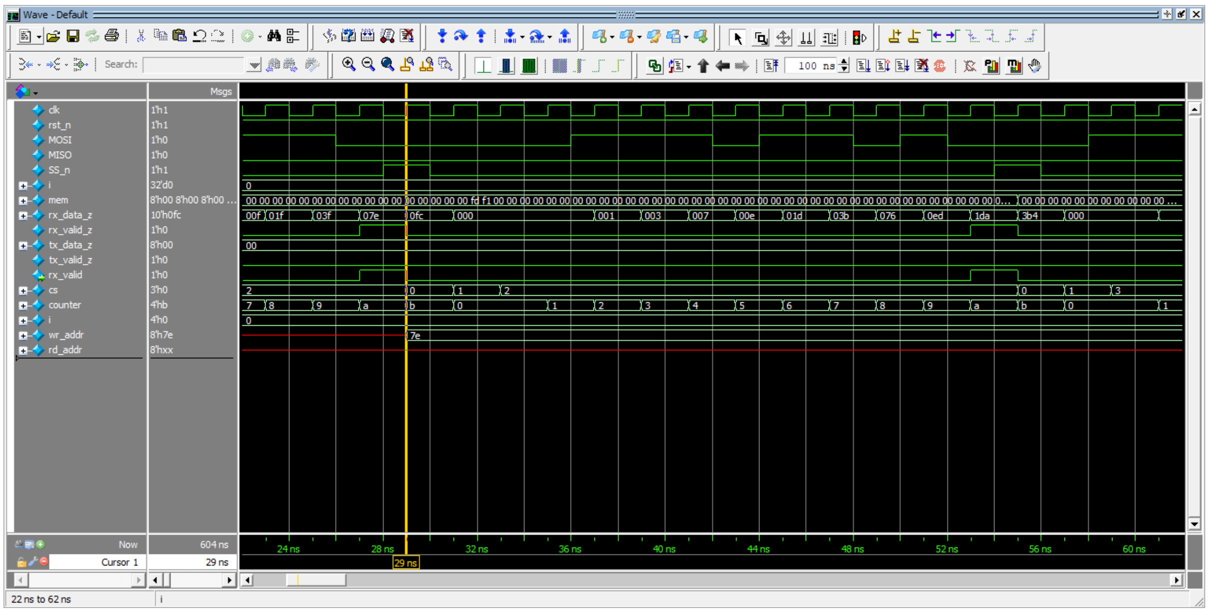


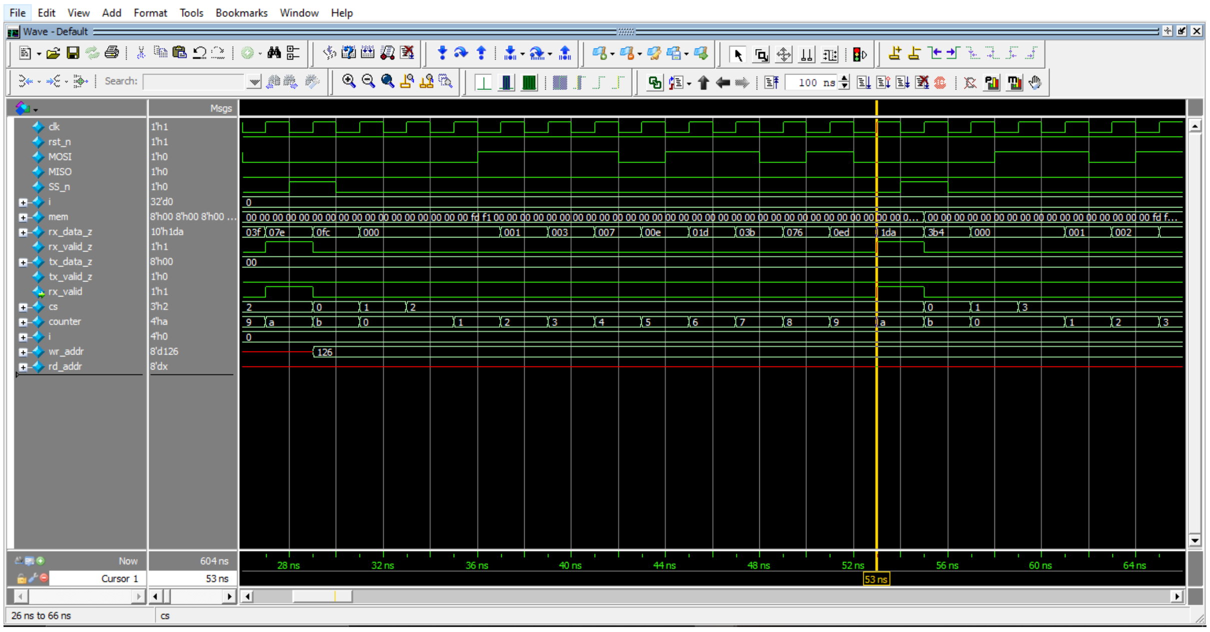
spi.v ram.v spiWrapper.v Project_tb.v

```verilog
      #2 SS_n=1;
      #2 SS_n=0; //test write
      #2 MOSI=0;
      #2 MOSI=0;
      #2 MOSI=1;
      repeat (8)
      #2 MOSI=$random;
      #2 SS_n=1;

      #2 SS_n=0; //test read
      #2 MOSI=1;
      #2 MOSI=1;
      #2 MOSI=0;
      repeat (8)
      #2 MOSI=$random;
      #2 SS_n=1;
      #2 SS_n=0;
      #2 MOSI=1;
      #2 MOSI=1;
      #2 MOSI=1;
      repeat (8)
      #2 MOSI=$random;
      repeat (8)
      #2 MOSI=$random;  //MOSI Here is dummy we just want to wait 8 cycles
      #2 SS_n=1;
    end

  #2 $stop;
  end

  endmodule
```

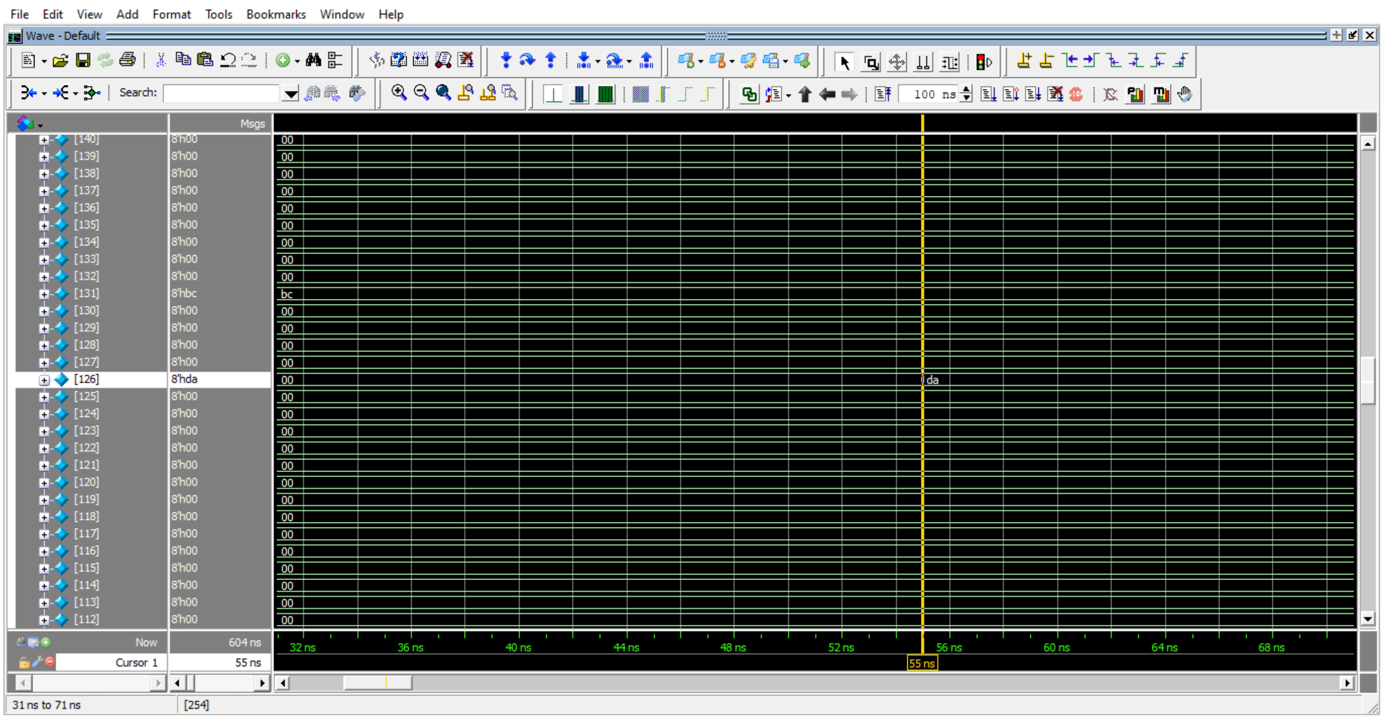
Line 1, Column 1

Snippets for write operation





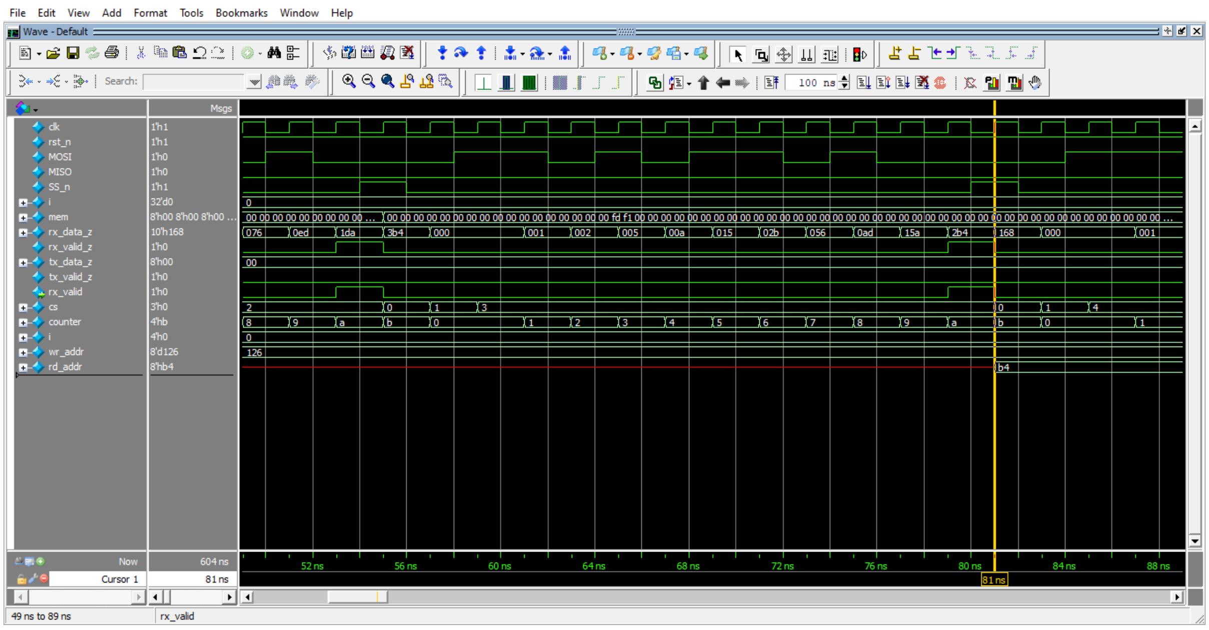
- Mem[126]=da;
- Write operation is done;



	Msgs	
[140]	8'h00	00
[139]	8'h00	00
[138]	8'h00	00
[137]	8'h00	00
[136]	8'h00	00
[135]	8'h00	00
[134]	8'h00	00
[133]	8'h00	00
[132]	8'h00	00
[131]	8'hbc	bc
[130]	8'h00	00
[129]	8'h00	00
[128]	8'h00	00
[127]	8'h00	00
[126]	8'hda	00
[125]	8'h00	00
[124]	8'h00	00
[123]	8'h00	00
[122]	8'h00	00
[121]	8'h00	00
[120]	8'h00	00
[119]	8'h00	00
[118]	8'h00	00
[117]	8'h00	00
[116]	8'h00	00
[115]	8'h00	00
[114]	8'h00	00
[113]	8'h00	00
[112]	8'h00	00
Now	604 ns	
Cursor 1	55 ns	

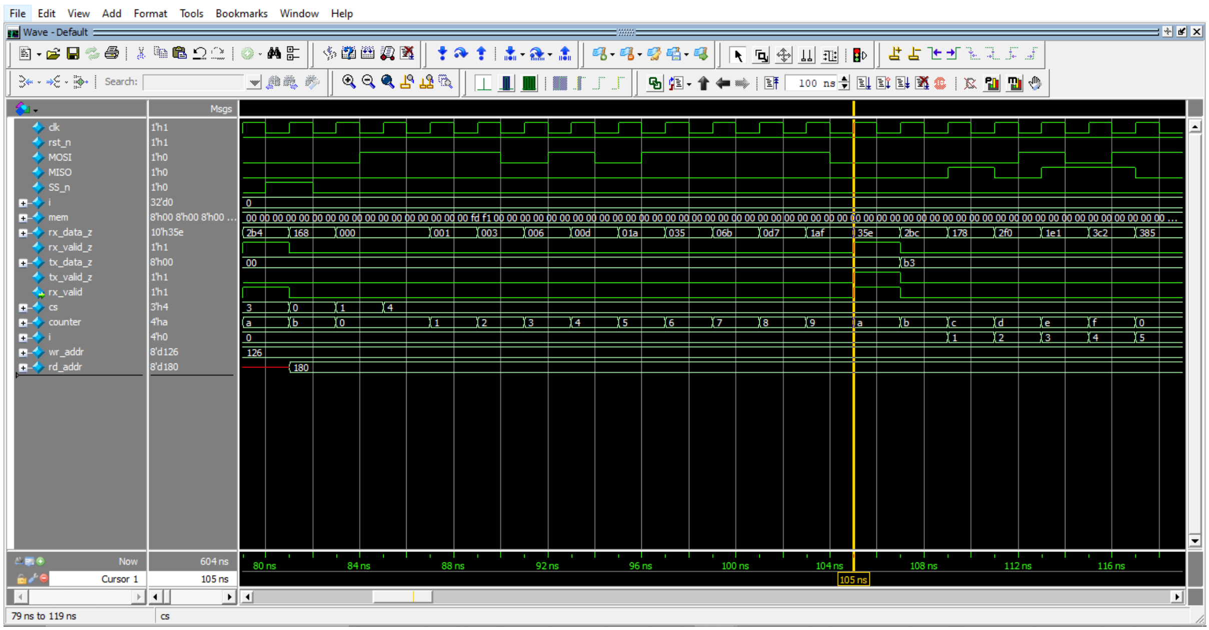
31 ns to 71 ns [254]

Snippets for read operation

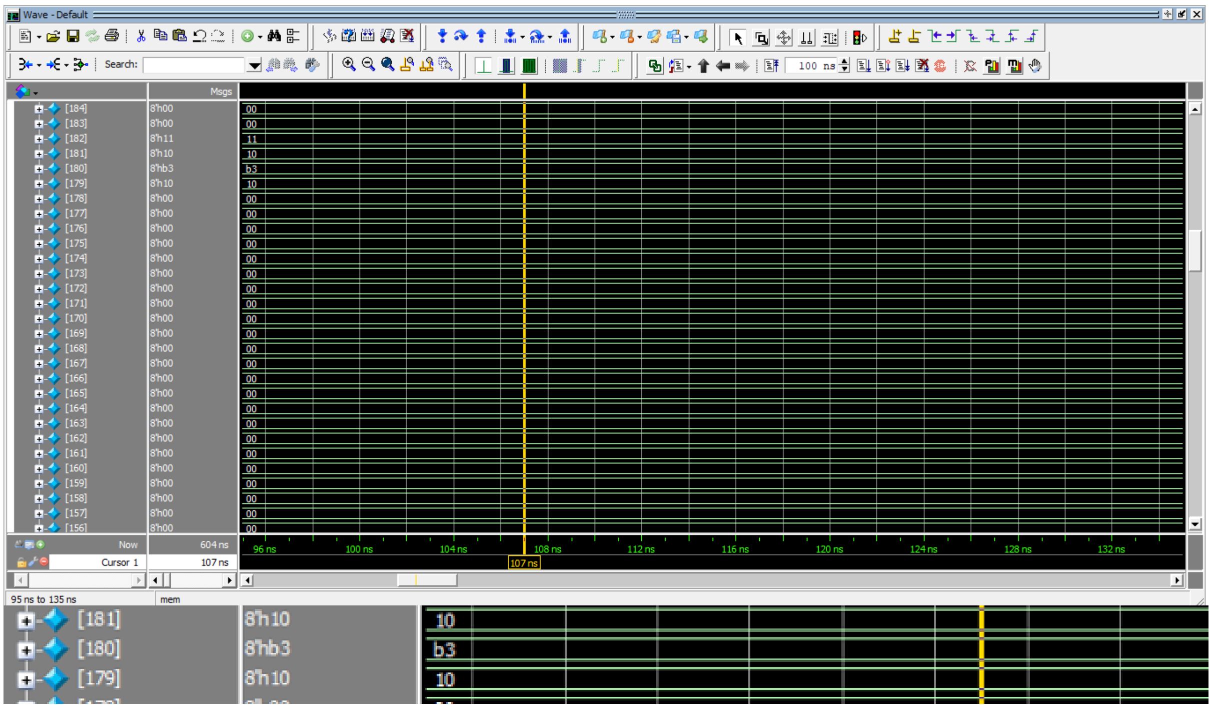


```
Wave - Default                    Msgs
clk                               1'h1
rst_n                             1'h1
MOSI                              1'h0
MISO                              1'h0
SS_n                              1'h1
i                                 32'd0
mem                               8'h00 8'h00 8'h00 ...
rx_data_z                         10'h168
rx_valid_z                        1'h0
tx_data_z                         8'h00
tx_valid_z                        1'h0
rx_valid                          1'h0
cs                                3'h0
counter                           4'hb
i                                 4'h0
wr_addr                           8'd126
rd_addr                           8'hb4
Now                               604 ns
Cursor 1                          81 ns
49 ns to 89 ns                    rx_valid
```

address read=8’hb4=8’d180



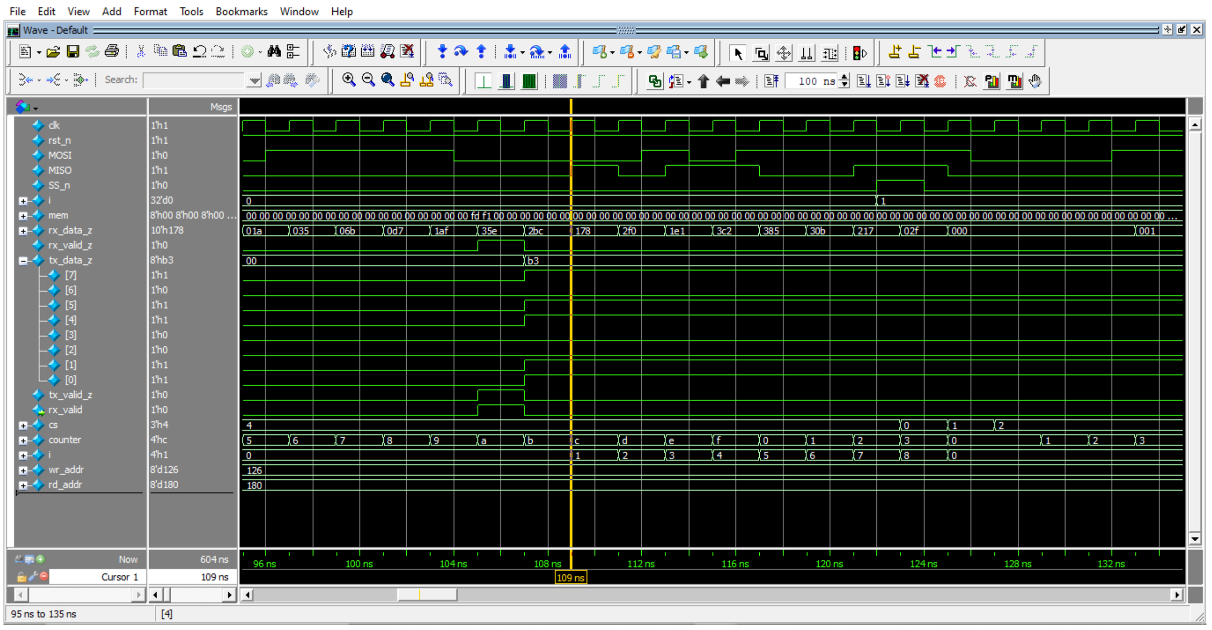
```
Wave - Default                    Msgs
clk                               1'h1
rst_n                             1'h1
MOSI                              1'h0
MISO                              1'h0
SS_n                              1'h0
i                                 32'd0
mem                               8'h00 8'h00 8'h00 ...
rx_data_z                         10'h35e
rx_valid_z                        1'h1
tx_data_z                         8'h00
tx_valid_z                        1'h1
rx_valid                          1'h1
cs                                3'h4
counter                           4'ha
i                                 4'h0
wr_addr                           8'd126
rd_addr                           8'd180
Now                               604 ns
Cursor 1                          105 ns
79 ns to 119 ns                   cs
```


Wave - Default

	Msgs
[184]	8'h00
[183]	8'h00
[182]	8'h11
[181]	8'h10
[180]	8'hb3
[179]	8'h10
[178]	8'h00
[177]	8'h00
[176]	8'h00
[175]	8'h00
[174]	8'h00
[173]	8'h00
[172]	8'h00
[171]	8'h00
[170]	8'h00
[169]	8'h00
[168]	8'h00
[167]	8'h00
[166]	8'h00
[165]	8'h00
[164]	8'h00
[163]	8'h00
[162]	8'h00
[161]	8'h00
[160]	8'h00
[159]	8'h00
[158]	8'h00
[157]	8'h00
[156]	8'h00
Now	604 ns
Cursor 1	107 ns

95 ns to 135 ns mem

[181]	8'h10	10
[180]	8'hb3	b3
[179]	8'h10	10


File Edit View Add Format Tools Bookmarks Window Help

Wave - Default

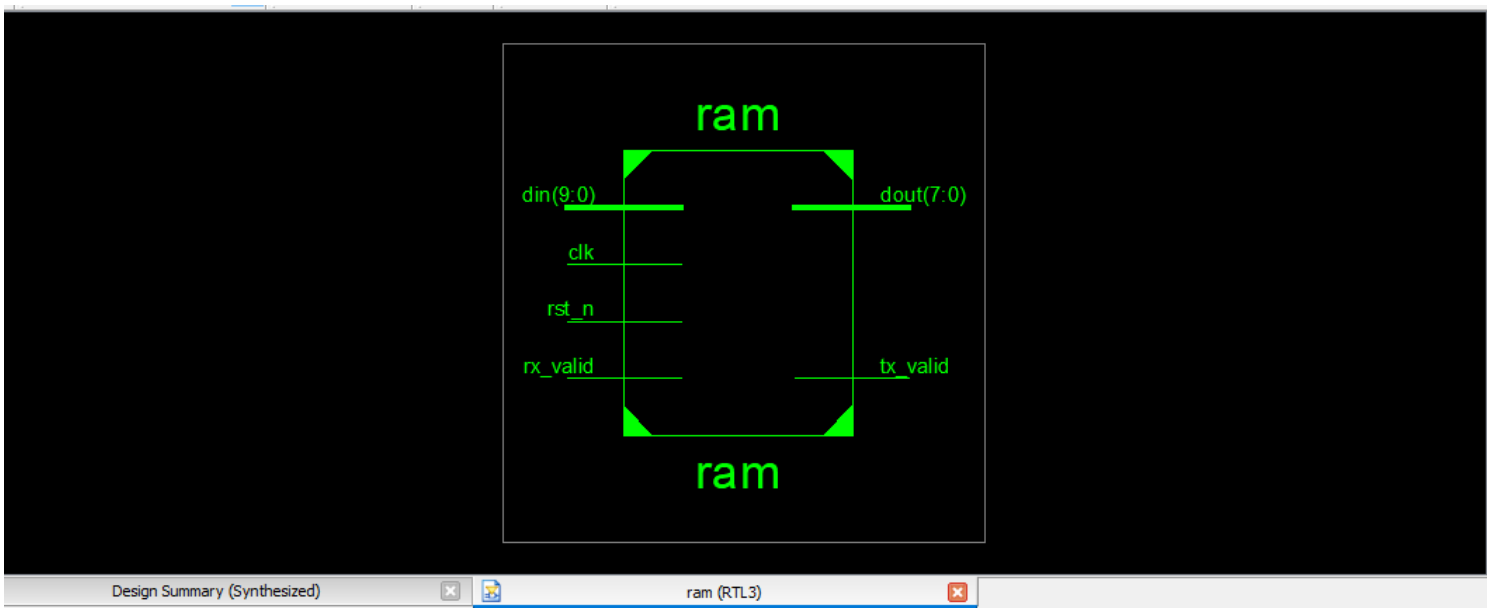
	Msgs
clk	1'h1
rst_n	1'h1
MOSI	1'h0
MISO	1'h1
SS_n	1'h0
i	32'd0
mem	8'h00 8'h00 8'h00 ...
rx_data_z	10'h178
rx_valid_z	1'h0
tx_data_z	8'hb3
[7]	1'h1
[6]	1'h0
[5]	1'h1
[4]	1'h1
[3]	1'h0
[2]	1'h0
[1]	1'h1
[0]	1'h1
tx_valid_z	1'h0
rx_valid	1'h0
cs	3'h4
counter	4'hc
i	4'h1
wr_addr	8'd126
rd_addr	8'd180
Now	604 ns
Cursor 1	109 ns

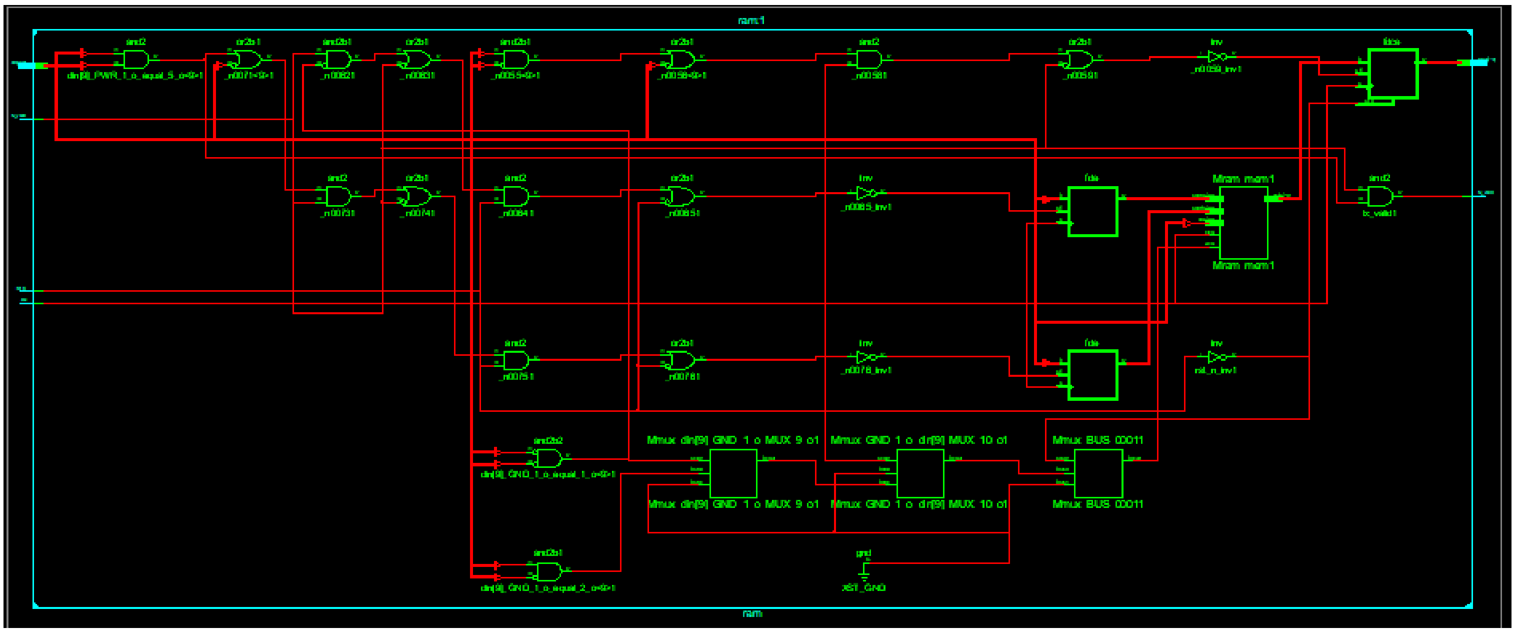
95 ns to 135 ns [4]

01a 035 06b 0d7 1af 35e 2bc 178 2f0 1e1 3c2 385 30b 217 02f 000 001
00 b3
5 6 7 8 9 a b c d e f 0 1 2 3 0 1 2 3
126
180
96 ns
100 ns
104 ns
108 ns
112 ns
116 ns
120 ns
124 ns
128 ns
132 ns
109 ns

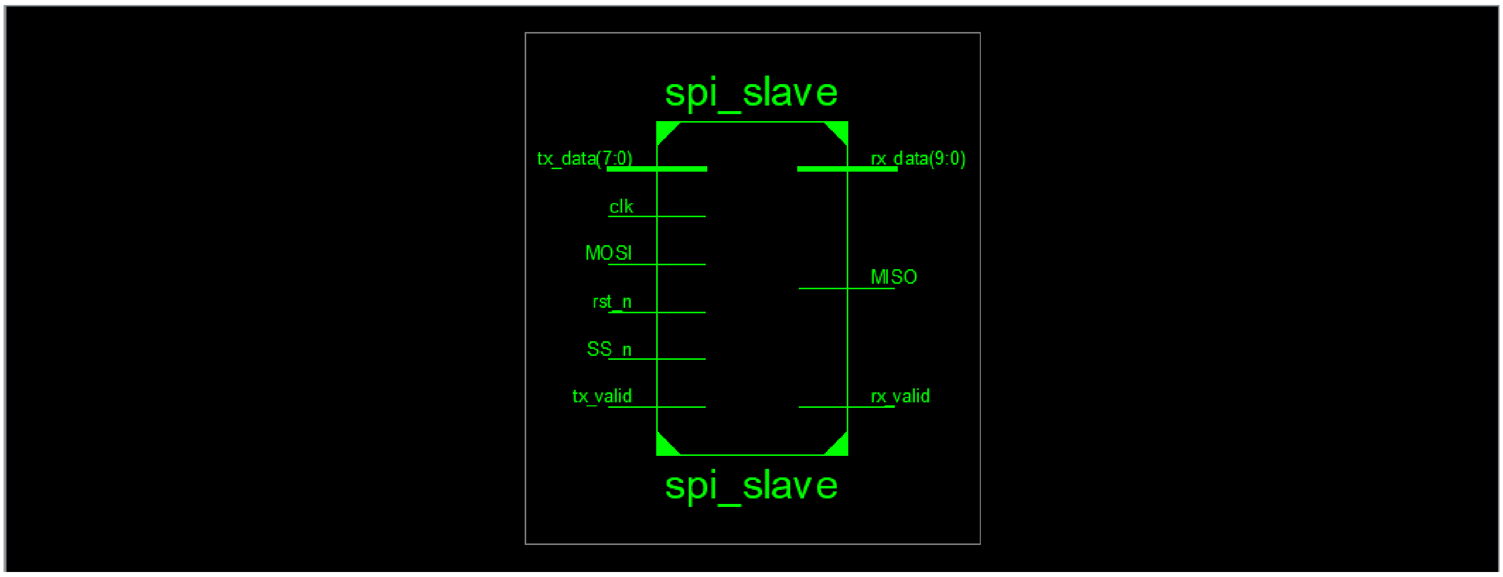
ISE Snippets

1-RAM





2-SPI


spi_slave
tx_data(7:0)
clk
MOSI
rst_n
SS_n
tx_valid
rx_data(9:0)
MISO
rx_valid
spi_slave