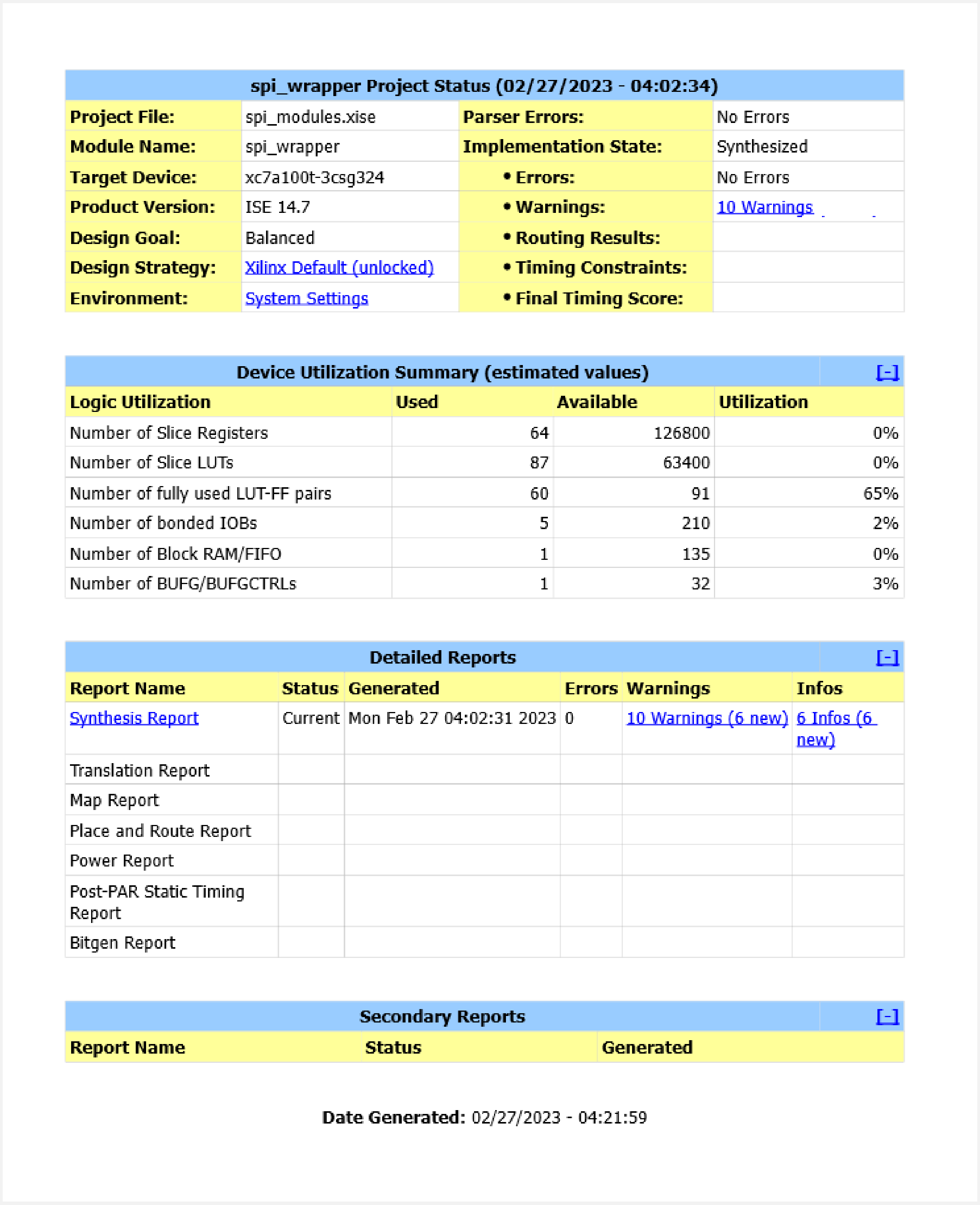
Digital Design

SPI With RAM

By: Hassan Khaled

Design Report



spi_wrapper Project Status (02/27/2023 - 04:02:34)			
Project File:	spi_modules.xise	**Parser Errors:**	No Errors
Module Name:	spi_wrapper	**Implementation State:**	Synthesized
Target Device:	xc7a100t-3csg324	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	10 Warnings
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	64	126800	0%
Number of Slice LUTs	87	63400	0%
Number of fully used LUT-FF pairs	60	91	65%
Number of bonded IOBs	5	210	2%
Number of Block RAM/FIFO	1	135	0%
Number of BUFG/BUFGCTRLs	1	32	3%

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon Feb 27 04:02:31 2023	0	10 Warnings (6 new)	6 Infos (6 new)
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

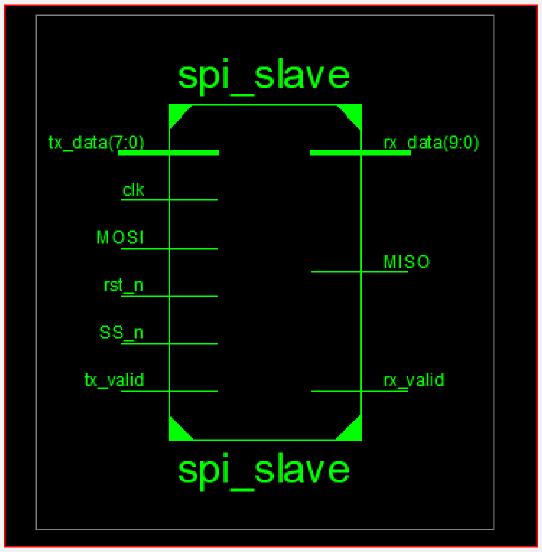
Secondary Reports [-]

Report Name	Status	Generated

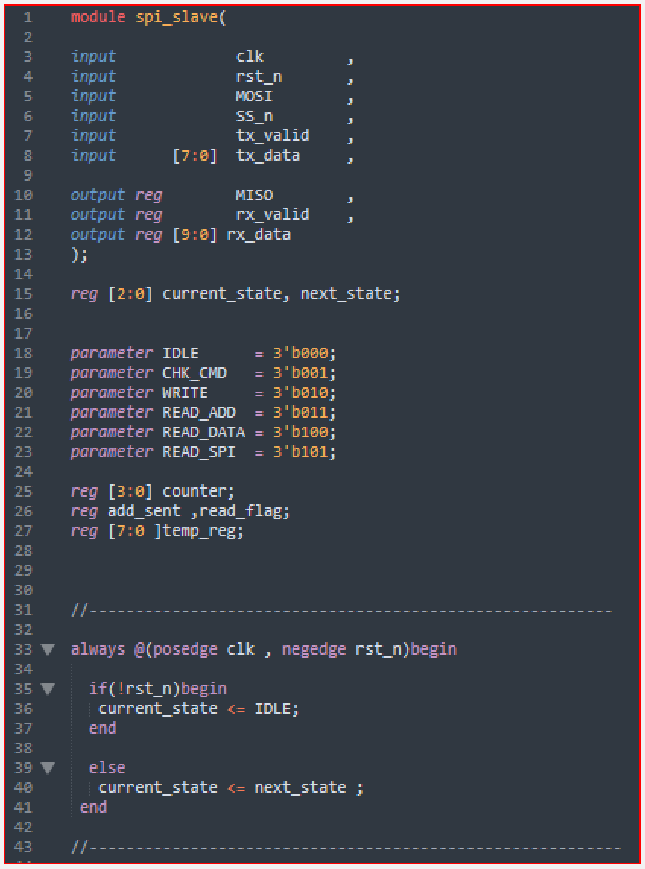
Date Generated: 02/27/2023 - 04:21:59

SPI Slave

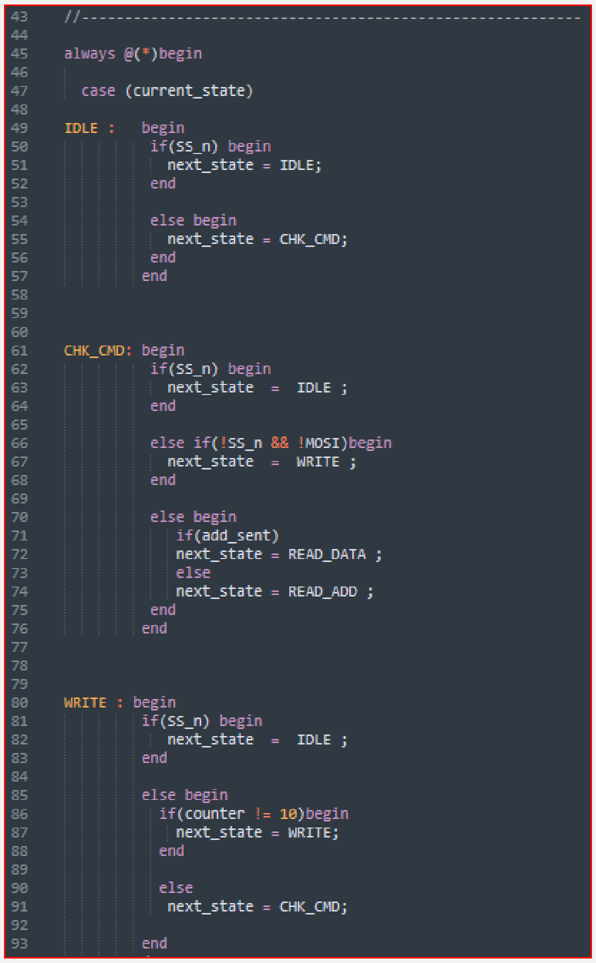
- **Block Diagram**



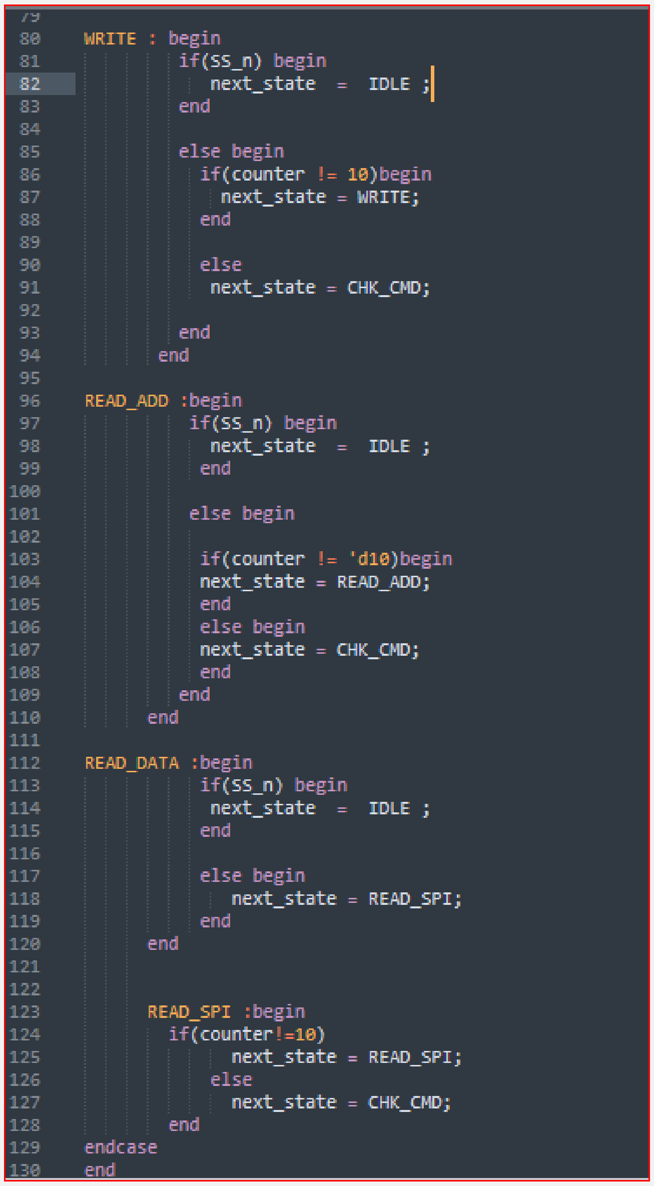
- **Design code**



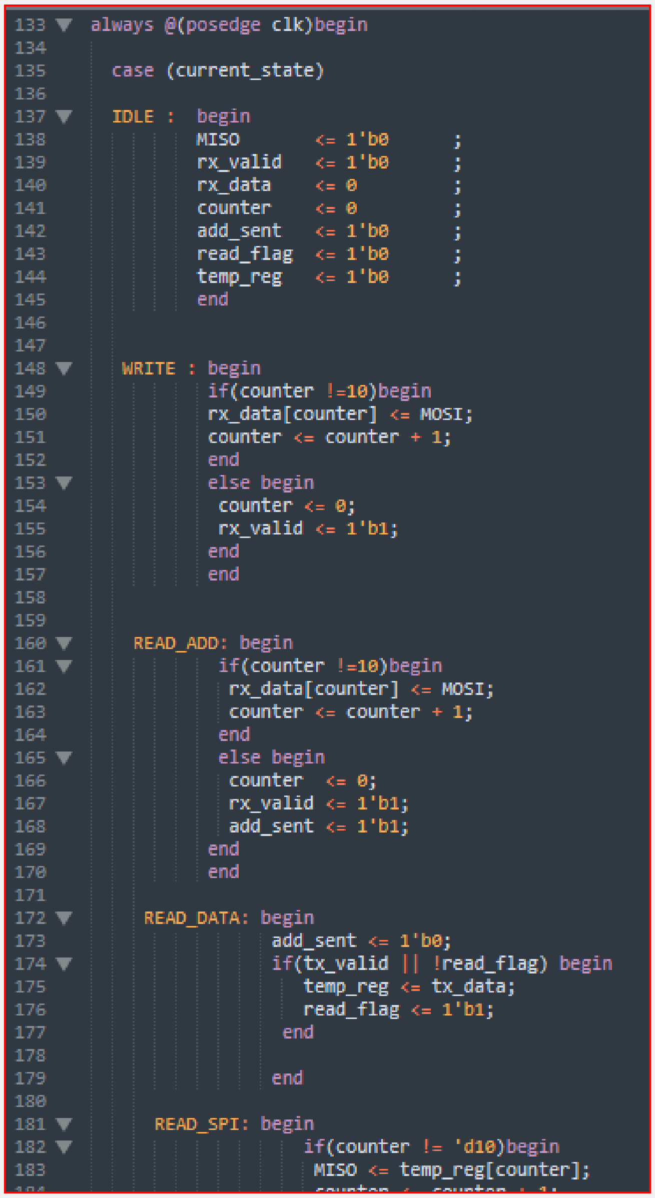
```verilog
module spi_slave(

input              clk         ,
input              rst_n       ,
input              MOSI        ,
input              SS_n        ,
input              tx_valid    ,
input      [7:0]   tx_data     ,

output reg         MISO        ,
output reg         rx_valid    ,
output reg [9:0] rx_data
);

reg [2:0] current_state, next_state;


parameter IDLE      = 3'b000;
parameter CHK_CMD   = 3'b001;
parameter WRITE     = 3'b010;
parameter READ_ADD  = 3'b011;
parameter READ_DATA = 3'b100;
parameter READ_SPI  = 3'b101;

reg [3:0] counter;
reg add_sent ,read_flag;
reg [7:0 ]temp_reg;



//-----------------------------------------------------------

always @(posedge clk , negedge rst_n)begin

  if(!rst_n)begin
   current_state <= IDLE;
  end

  else
   current_state <= next_state ;
 end

//------------------------------------------------------------
```



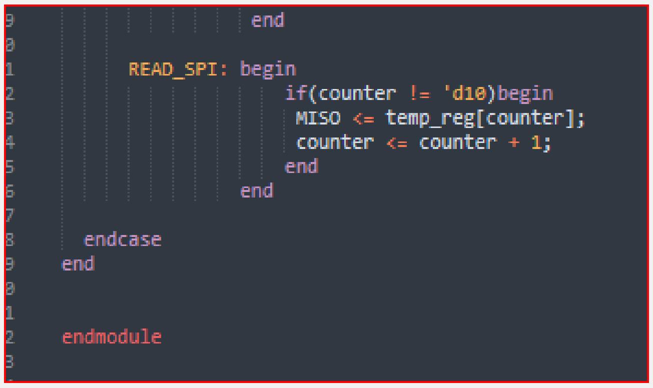
```verilog
//------------------------------------------------------------

always @(*)begin

  case (current_state)

IDLE :    begin
           if(SS_n) begin
             next_state = IDLE;
           end

           else begin
             next_state = CHK_CMD;
           end
          end



CHK_CMD: begin
           if(SS_n) begin
             next_state  =  IDLE ;
           end

           else if(!SS_n && !MOSI)begin
             next_state  =  WRITE ;
           end

           else begin
              if(add_sent)
              next_state = READ_DATA ;
              else
              next_state = READ_ADD ;
           end
          end



WRITE : begin
          if(SS_n) begin
             next_state  =  IDLE ;
          end

          else begin
            if(counter != 10)begin
              next_state = WRITE;
            end

            else
             next_state = CHK_CMD;

          end
```



```verilog
WRITE : begin
        if(SS_n) begin
           next_state  =  IDLE ;
        end

        else begin
          if(counter != 10)begin
            next_state = WRITE;
          end

          else
           next_state = CHK_CMD;

        end
      end

READ_ADD :begin
          if(SS_n) begin
            next_state  =  IDLE ;
          end

          else begin

           if(counter != 'd10)begin
           next_state = READ_ADD;
           end
           else begin
           next_state = CHK_CMD;
           end
         end
      end

READ_DATA :begin
           if(SS_n) begin
            next_state  =  IDLE ;
           end

           else begin
              next_state = READ_SPI;
           end
      end


      READ_SPI :begin
        if(counter!=10)
              next_state = READ_SPI;
            else
              next_state = CHK_CMD;
        end
endcase
end
```



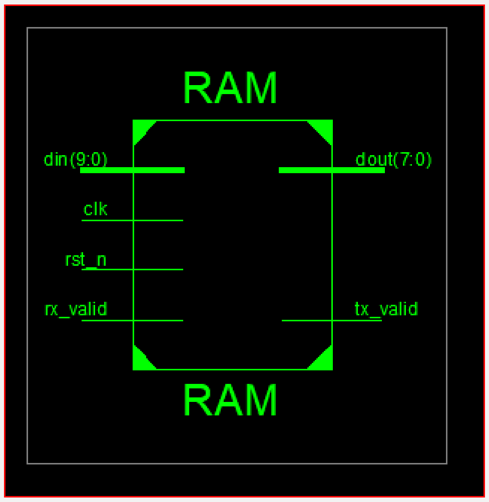
```verilog
always @(posedge clk)begin

  case (current_state)

  IDLE :  begin
          MISO        <= 1'b0      ;
          rx_valid    <= 1'b0      ;
          rx_data     <= 0         ;
          counter     <= 0         ;
          add_sent    <= 1'b0      ;
          read_flag   <= 1'b0      ;
          temp_reg    <= 1'b0      ;
          end


   WRITE : begin
           if(counter !=10)begin
           rx_data[counter] <= MOSI;
           counter <= counter + 1;
           end
           else begin
            counter <= 0;
            rx_valid <= 1'b1;
           end
           end


    READ_ADD: begin
            if(counter !=10)begin
             rx_data[counter] <= MOSI;
             counter <= counter + 1;
            end
            else begin
             counter  <= 0;
             rx_valid <= 1'b1;
             add_sent <= 1'b1;
           end
           end

     READ_DATA: begin
                add_sent <= 1'b0;
                if(tx_valid || !read_flag) begin
                   temp_reg <= tx_data;
                   read_flag <= 1'b1;
                 end

                end

      READ_SPI: begin
                    if(counter != 'd10)begin
                     MISO <= temp_reg[counter];
```



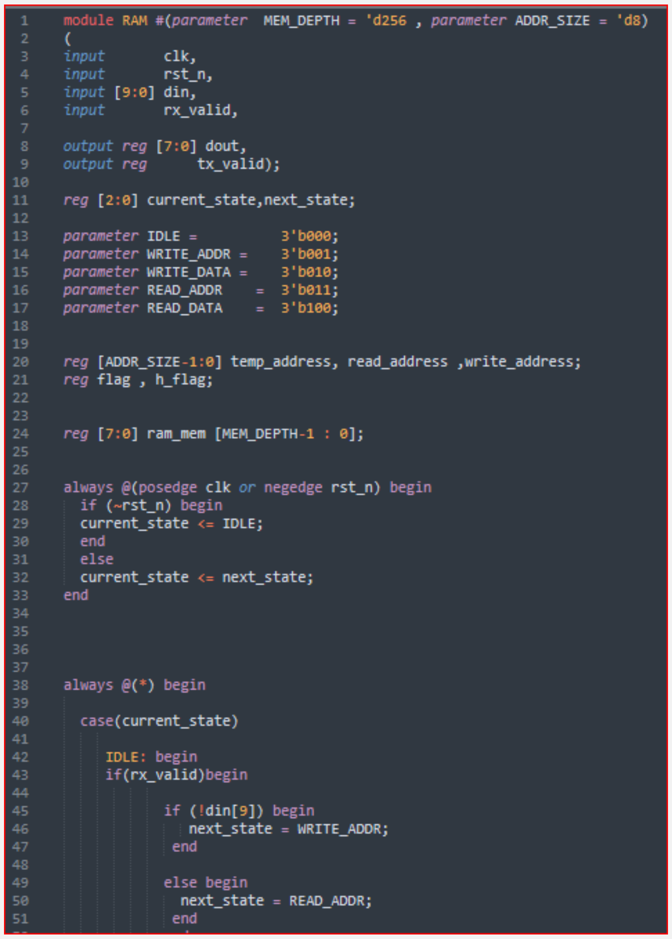
```verilog
                    end

      READ_SPI: begin
                    if(counter != 'd10)begin
                     MISO <= temp_reg[counter];
                     counter <= counter + 1;
                    end
                end

    endcase
  end


  endmodule
```

RAM

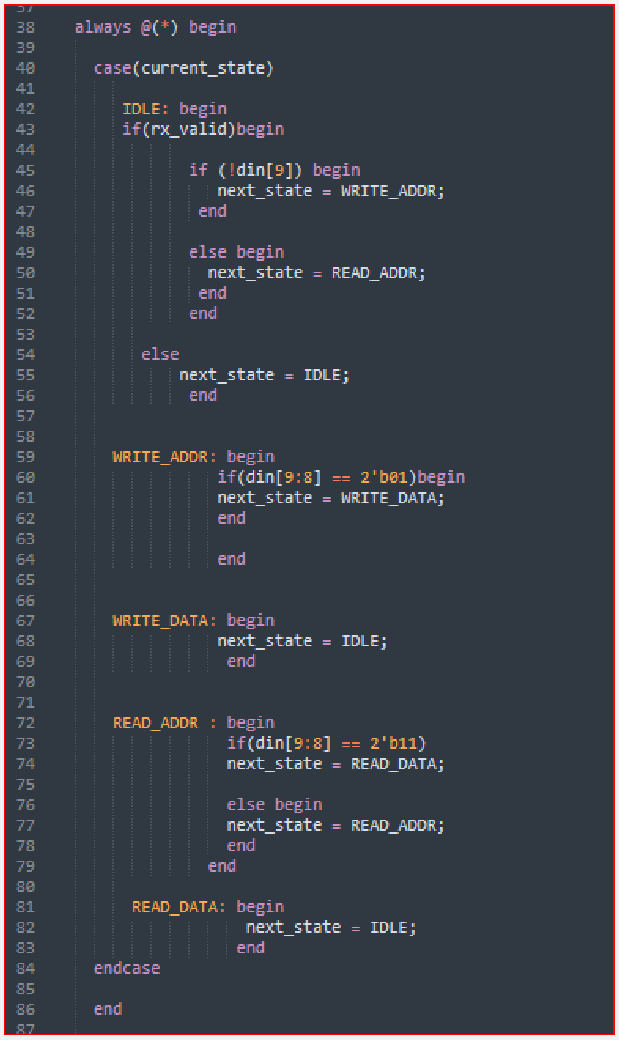
- **Block Diagram**



- **Design Code**

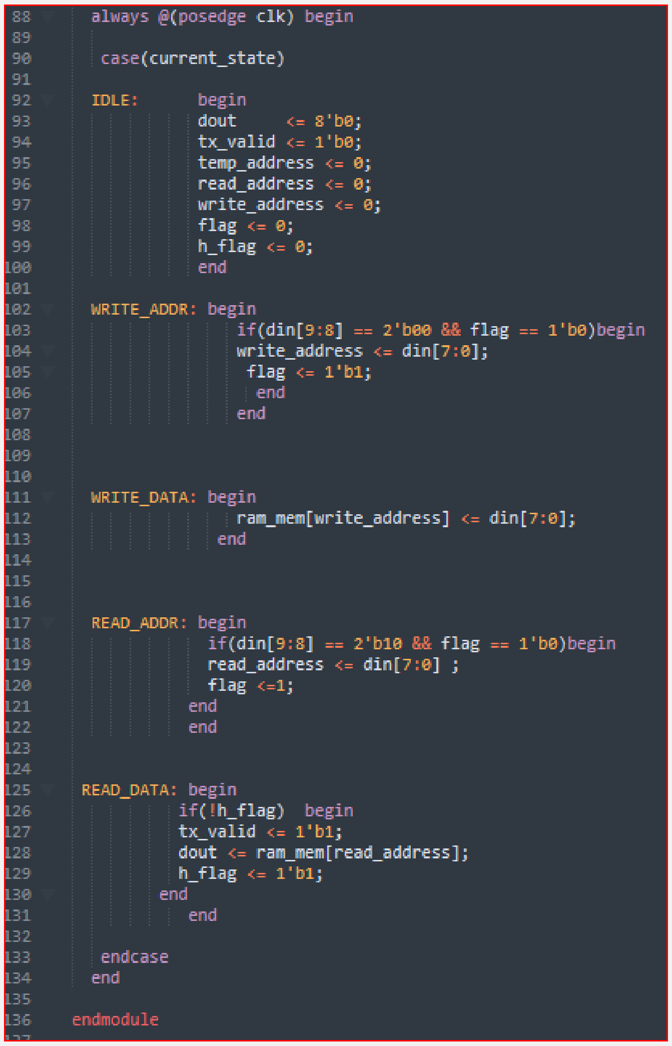


```verilog
module RAM #(parameter  MEM_DEPTH = 'd256 , parameter ADDR_SIZE = 'd8)
(
input       clk,
input       rst_n,
input [9:0] din,
input       rx_valid,

output reg [7:0] dout,
output reg      tx_valid);

reg [2:0] current_state,next_state;

parameter IDLE =          3'b000;
parameter WRITE_ADDR =    3'b001;
parameter WRITE_DATA =    3'b010;
parameter READ_ADDR    =  3'b011;
parameter READ_DATA    =  3'b100;


reg [ADDR_SIZE-1:0] temp_address, read_address ,write_address;
reg flag , h_flag;


reg [7:0] ram_mem [MEM_DEPTH-1 : 0];


always @(posedge clk or negedge rst_n) begin
  if (~rst_n) begin
  current_state <= IDLE;
  end
  else
  current_state <= next_state;
end




always @(*) begin

  case(current_state)

     IDLE: begin
     if(rx_valid)begin

            if (!din[9]) begin
               next_state = WRITE_ADDR;
             end

            else begin
              next_state = READ_ADDR;
             end
```



```verilog
always @(*) begin

  case(current_state)

     IDLE: begin
     if(rx_valid)begin

            if (!din[9]) begin
               next_state = WRITE_ADDR;
             end

            else begin
              next_state = READ_ADDR;
             end
```

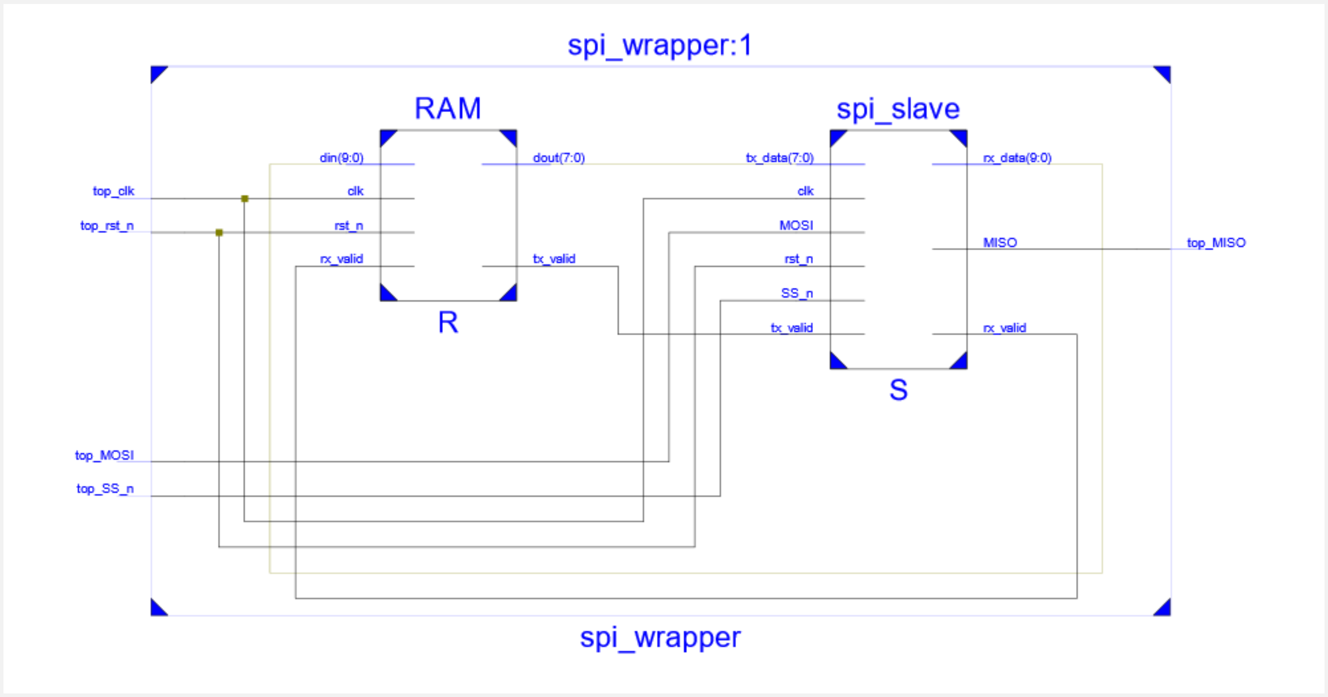
```verilog
            end

        else
            next_state = IDLE;
            end


    WRITE_ADDR: begin
               if(din[9:8] == 2'b01)begin
               next_state = WRITE_DATA;
               end

               end


    WRITE_DATA: begin
               next_state = IDLE;
                end


    READ_ADDR : begin
                if(din[9:8] == 2'b11)
                next_state = READ_DATA;

                else begin
                next_state = READ_ADDR;
                end
              end

      READ_DATA: begin
                  next_state = IDLE;
                 end
  endcase

  end
```



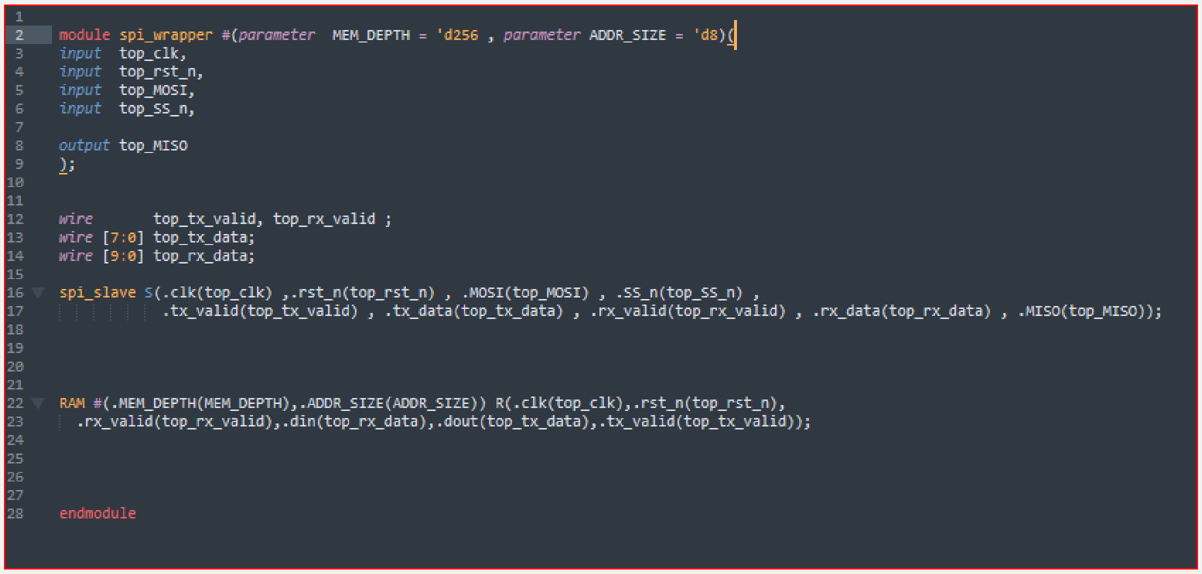
```
  always @(posedge clk) begin

   case(current_state)

  IDLE:      begin
             dout     <= 8'b0;
             tx_valid <= 1'b0;
             temp_address <= 0;
             read_address <= 0;
             write_address <= 0;
             flag <= 0;
             h_flag <= 0;
             end

  WRITE_ADDR: begin
                 if(din[9:8] == 2'b00 && flag == 1'b0)begin
                 write_address <= din[7:0];
                  flag <= 1'b1;
                   end
                 end



  WRITE_DATA: begin
                 ram_mem[write_address] <= din[7:0];
               end



  READ_ADDR: begin
              if(din[9:8] == 2'b10 && flag == 1'b0)begin
              read_address <= din[7:0] ;
              flag <=1;
             end
             end


 READ_DATA: begin
            if(!h_flag)  begin
            tx_valid <= 1'b1;
            dout <= ram_mem[read_address];
            h_flag <= 1'b1;
          end
             end

   endcase
  end

endmodule
```

SPI_WRAPPER

• Block Diagram

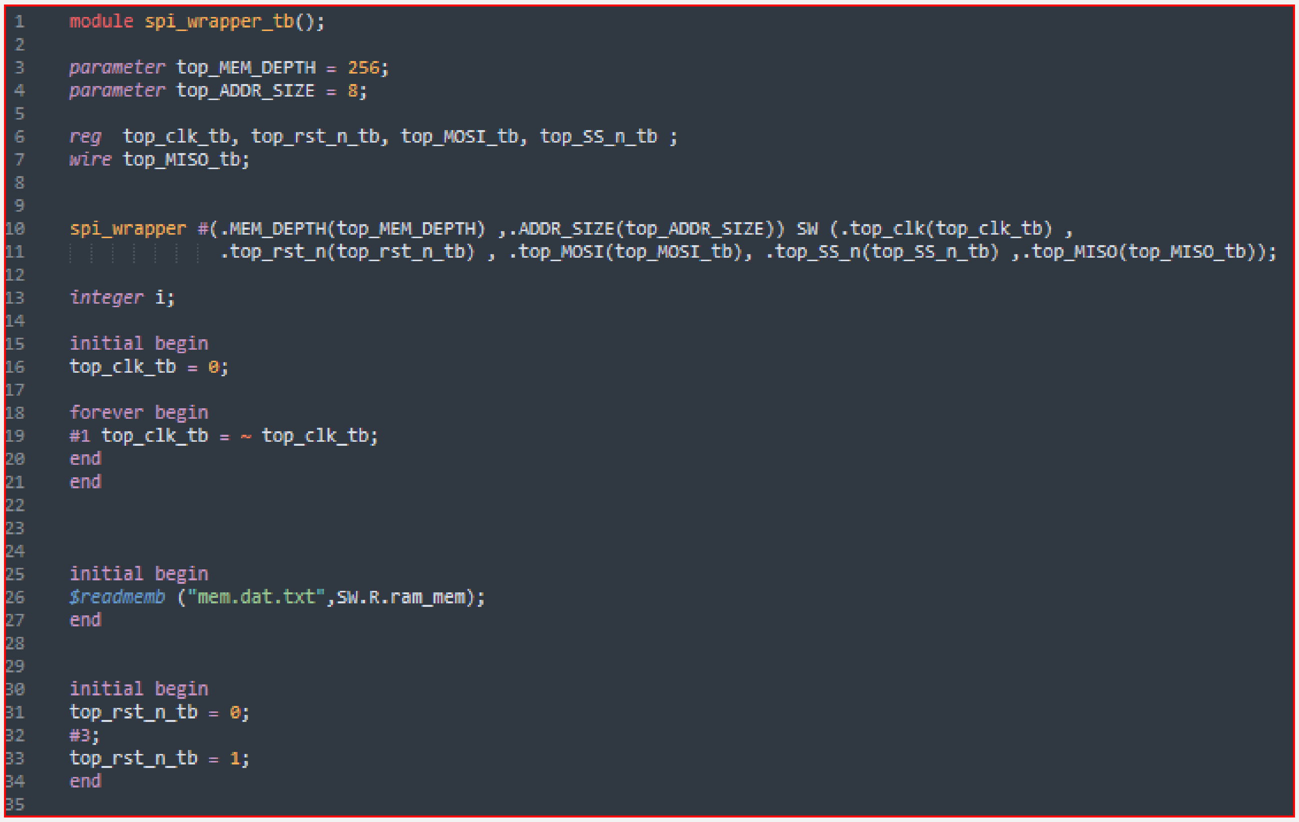


• Design Code

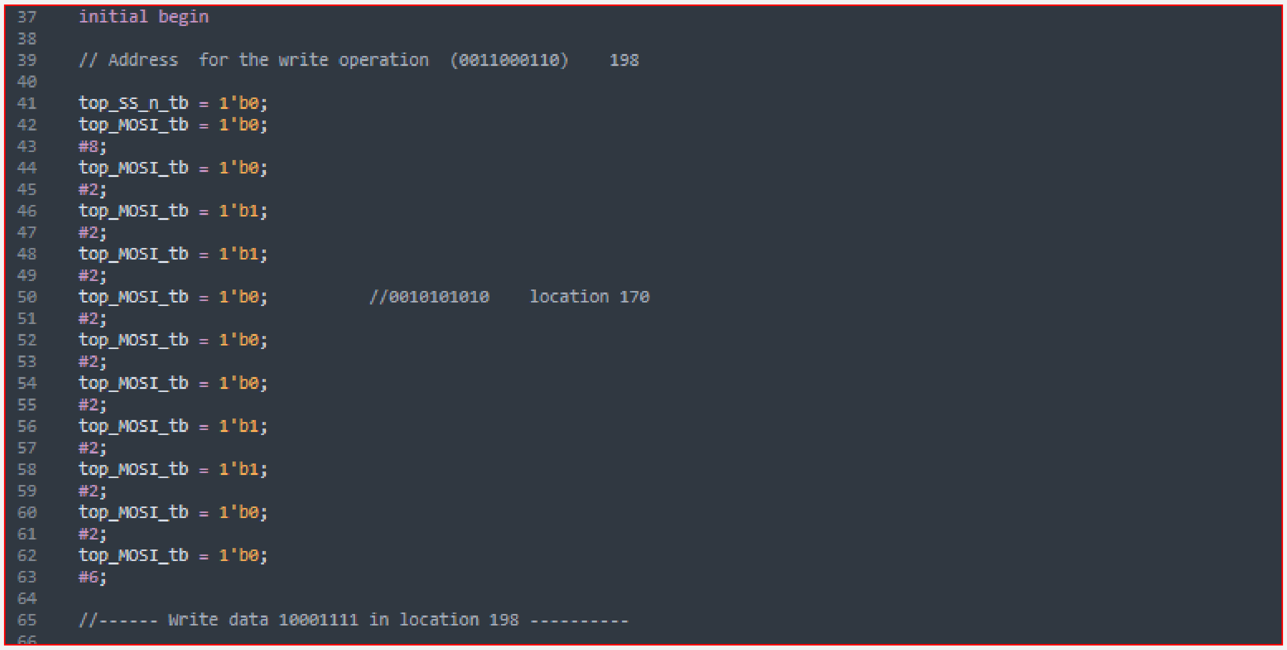


```verilog

module spi_wrapper #(parameter  MEM_DEPTH = 'd256 , parameter ADDR_SIZE = 'd8)(
input  top_clk,
input  top_rst_n,
input  top_MOSI,
input  top_SS_n,

output top_MISO
);


wire       top_tx_valid, top_rx_valid ;
wire [7:0] top_tx_data;
wire [9:0] top_rx_data;

spi_slave S(.clk(top_clk) ,.rst_n(top_rst_n) , .MOSI(top_MOSI) , .SS_n(top_SS_n) ,
            .tx_valid(top_tx_valid) , .tx_data(top_tx_data) , .rx_valid(top_rx_valid) , .rx_data(top_rx_data) , .MISO(top_MISO));




RAM #(.MEM_DEPTH(MEM_DEPTH),.ADDR_SIZE(ADDR_SIZE)) R(.clk(top_clk),.rst_n(top_rst_n),
  .rx_valid(top_rx_valid),.din(top_rx_data),.dout(top_tx_data),.tx_valid(top_tx_valid));




endmodule
```

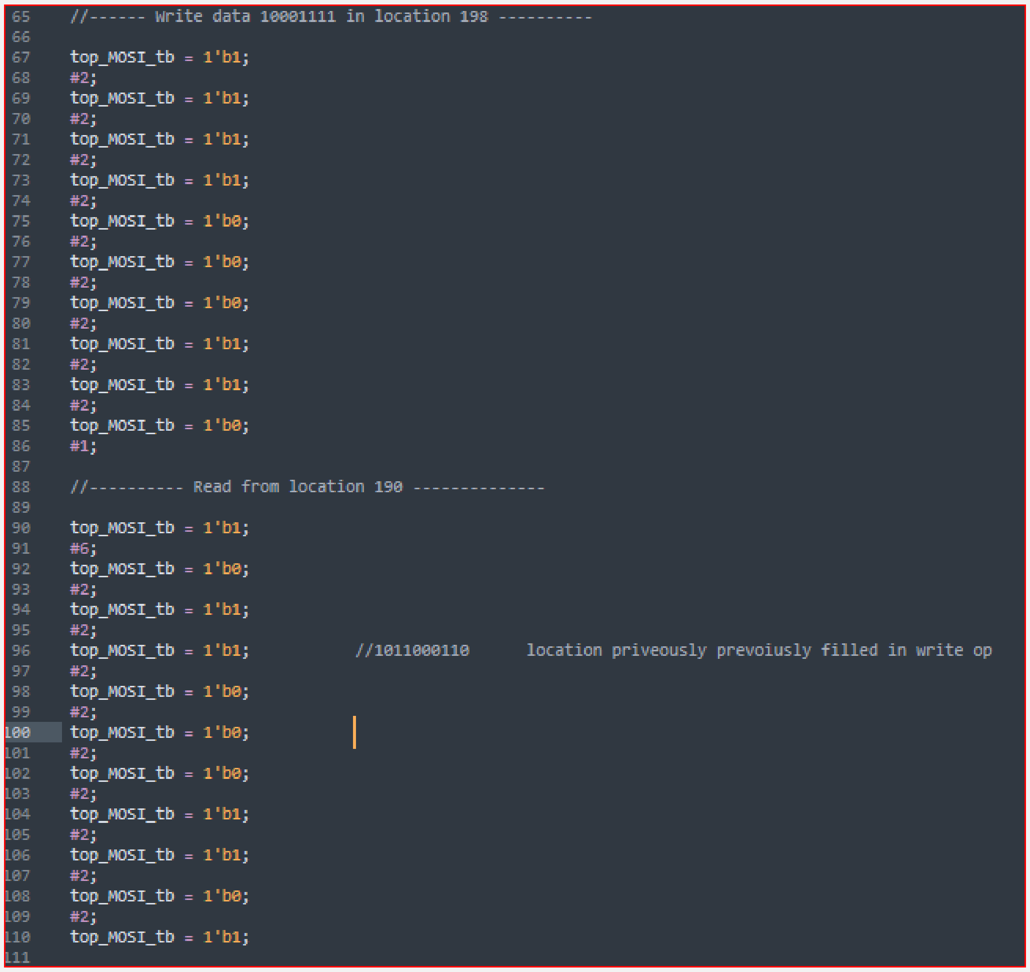
• Testbench Code



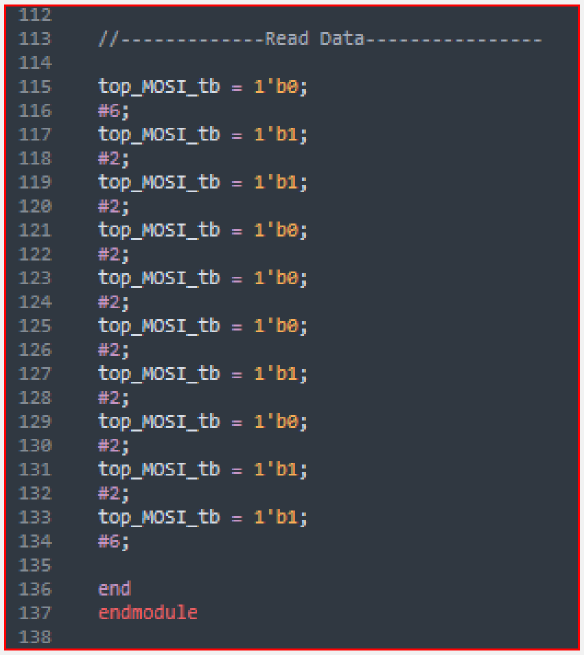
```verilog
module spi_wrapper_tb();

parameter top_MEM_DEPTH = 256;
parameter top_ADDR_SIZE = 8;

reg  top_clk_tb, top_rst_n_tb, top_MOSI_tb, top_SS_n_tb ;
wire top_MISO_tb;


spi_wrapper #(.MEM_DEPTH(top_MEM_DEPTH) ,.ADDR_SIZE(top_ADDR_SIZE)) SW (.top_clk(top_clk_tb) ,
              .top_rst_n(top_rst_n_tb) , .top_MOSI(top_MOSI_tb), .top_SS_n(top_SS_n_tb) ,.top_MISO(top_MISO_tb));

integer i;

initial begin
top_clk_tb = 0;

forever begin
#1 top_clk_tb = ~ top_clk_tb;
end
end



initial begin
$readmemb ("mem.dat.txt",SW.R.ram_mem);
end


initial begin
top_rst_n_tb = 0;
#3;
top_rst_n_tb = 1;
end

```



```verilog
initial begin

// Address  for the write operation  (0011000110)    198

top_SS_n_tb = 1'b0;
top_MOSI_tb = 1'b0;
#8;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;          //0010101010    location 170
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#6;

//------ Write data 10001111 in location 198 ----------
```



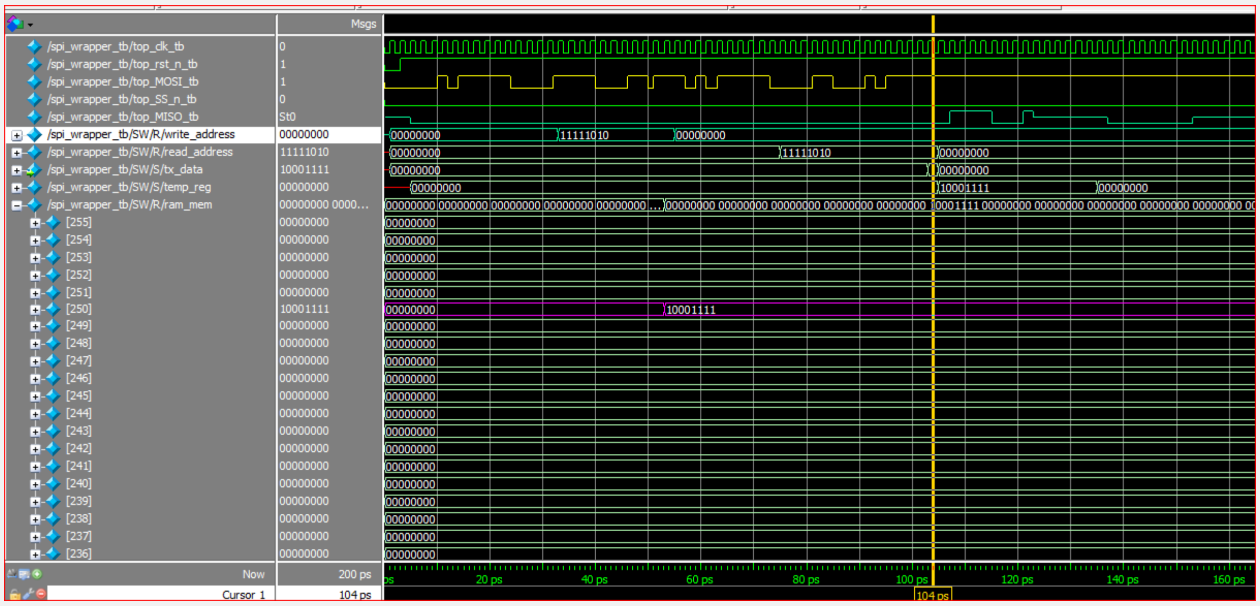
```
//------ Write data 10001111 in location 198 ----------

top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#1;

//---------- Read from location 190 -------------

top_MOSI_tb = 1'b1;
#6;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;           //1011000110      location priveously prevoiusly filled in write op
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;

```



```

//-------------Read Data----------------

top_MOSI_tb = 1'b0;
#6;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b0;
#2;
top_MOSI_tb = 1'b1;
#2;
top_MOSI_tb = 1'b1;
#6;

end
endmodule

```

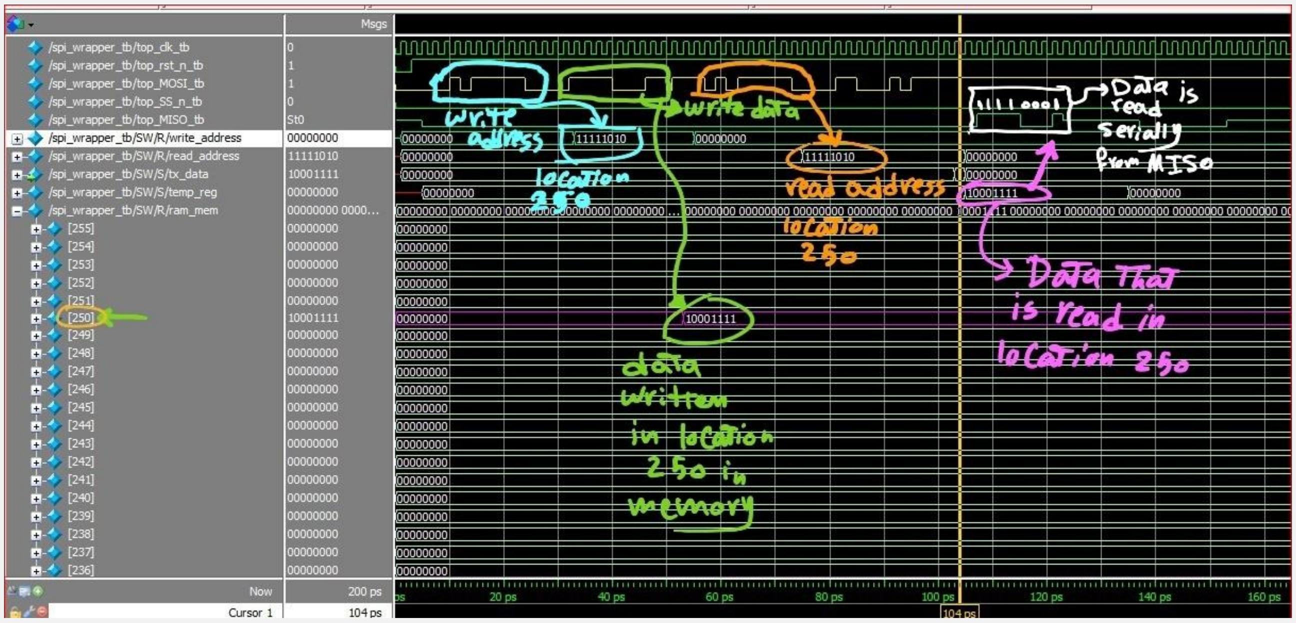
- **Waveform**

  - **Test Case 1 -> Write in location 250 then read from the same location**



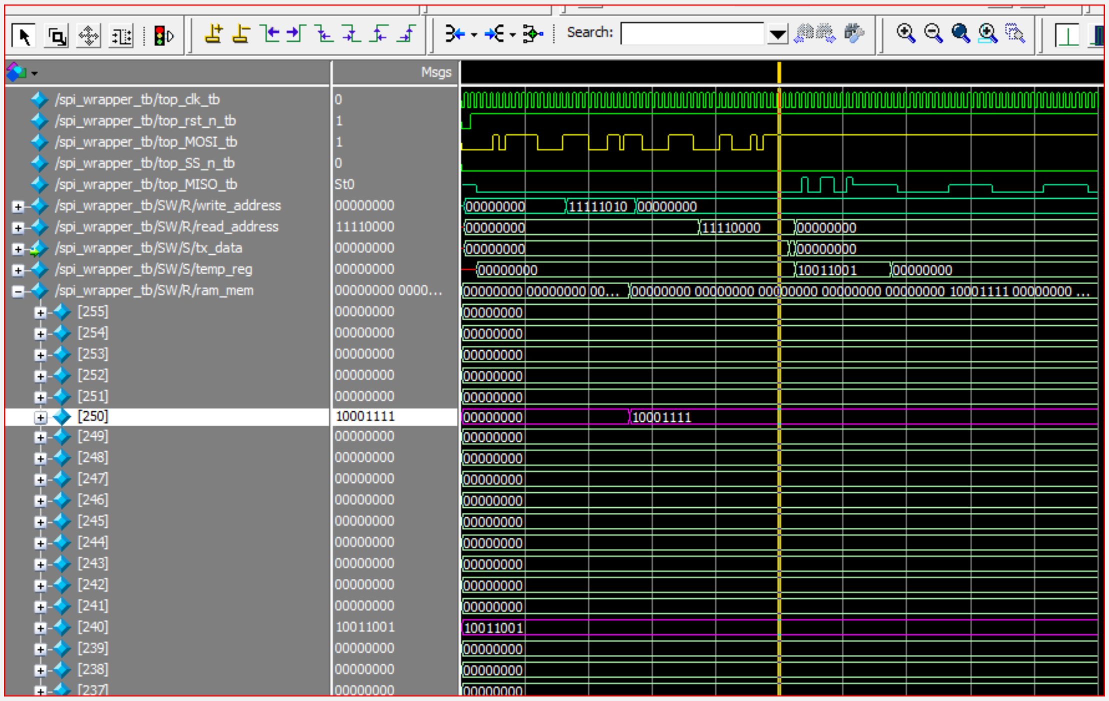
	Msgs
/spi_wrapper_tb/top_clk_tb	0
/spi_wrapper_tb/top_rst_n_tb	1
/spi_wrapper_tb/top_MOSI_tb	1
/spi_wrapper_tb/top_SS_n_tb	0
/spi_wrapper_tb/top_MISO_tb	St0
/spi_wrapper_tb/SW/R/write_address	00000000
/spi_wrapper_tb/SW/R/read_address	11111010
/spi_wrapper_tb/SW/S/tx_data	10001111
/spi_wrapper_tb/SW/S/temp_reg	00000000
/spi_wrapper_tb/SW/R/ram_mem	00000000 0000...
[255]	00000000
[254]	00000000
[253]	00000000
[252]	00000000
[251]	00000000
[250]	10001111
[249]	00000000
[248]	00000000
[247]	00000000
[246]	00000000
[245]	00000000
[244]	00000000
[243]	00000000
[242]	00000000
[241]	00000000
[240]	00000000
[239]	00000000
[238]	00000000
[237]	00000000
[236]	00000000
Now	200 ps
Cursor 1	104 ps

Explained Waveform



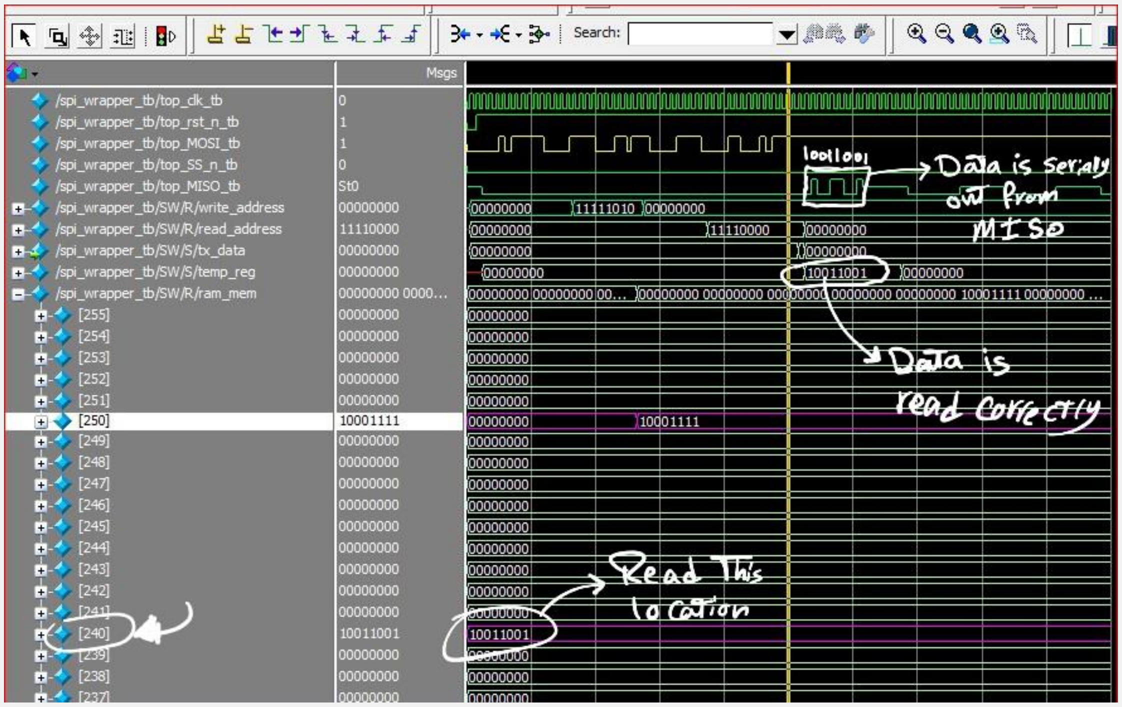
write address
location 250
write data
read address
location 250
11110001
Data is read serially from MISO
Data That is read in location 250
data written in location 250 in memory

- Test Case 2 -> Write in location 250 then read from different location 240 that I have previously set manually during initialization.

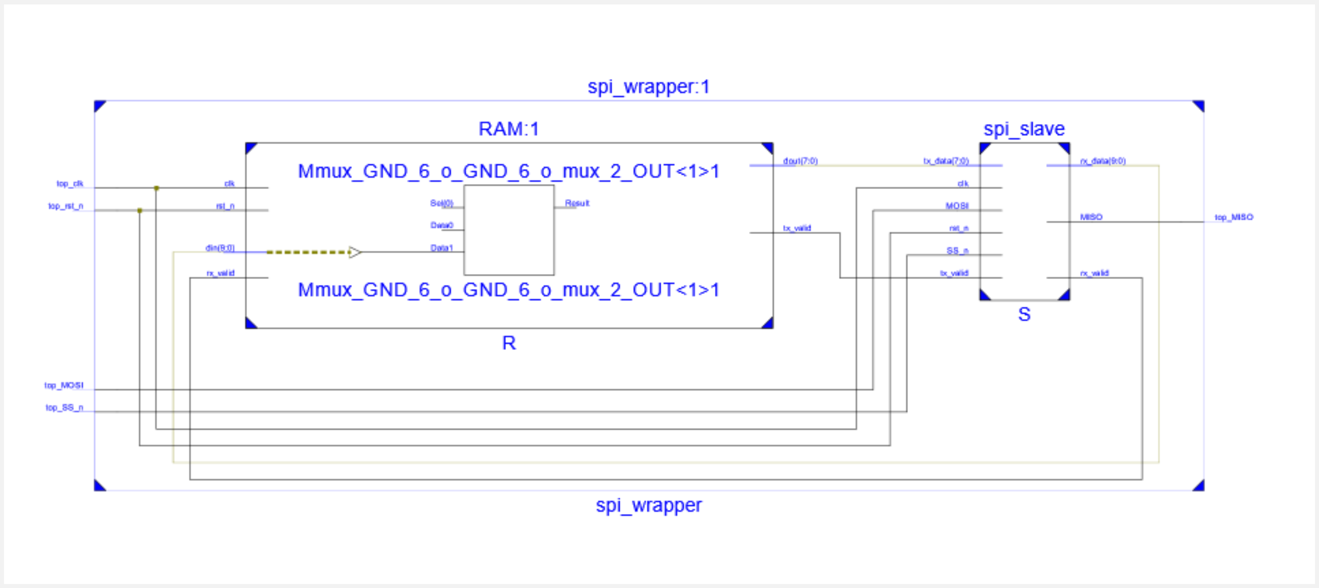


```
/spi_wrapper_tb/top_clk_tb            0
/spi_wrapper_tb/top_rst_n_tb          1
/spi_wrapper_tb/top_MOSI_tb           1
/spi_wrapper_tb/top_SS_n_tb           0
/spi_wrapper_tb/top_MISO_tb           St0
/spi_wrapper_tb/SW/R/write_address    00000000
/spi_wrapper_tb/SW/R/read_address     11110000
/spi_wrapper_tb/SW/S/tx_data          00000000
/spi_wrapper_tb/SW/S/temp_reg         00000000
/spi_wrapper_tb/SW/R/ram_mem          00000000 0000...
[255]                                 00000000
[254]                                 00000000
[253]                                 00000000
[252]                                 00000000
[251]                                 00000000
[250]                                 10001111
[249]                                 00000000
[248]                                 00000000
[247]                                 00000000
[246]                                 00000000
[245]                                 00000000
[244]                                 00000000
[243]                                 00000000
[242]                                 00000000
[241]                                 00000000
[240]                                 10011001
[239]                                 00000000
[238]                                 00000000
[237]                                 00000000
```

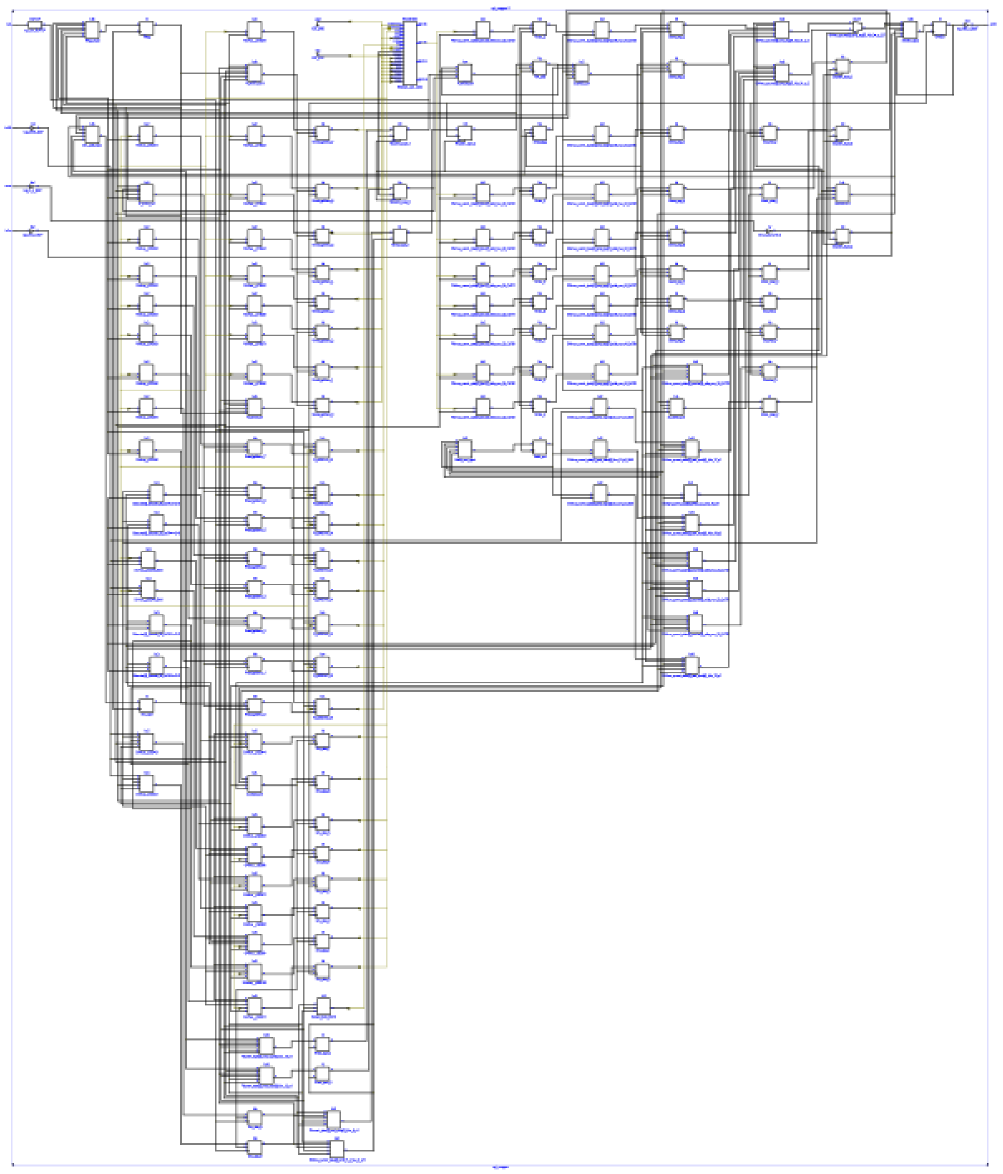
Explained Waveform



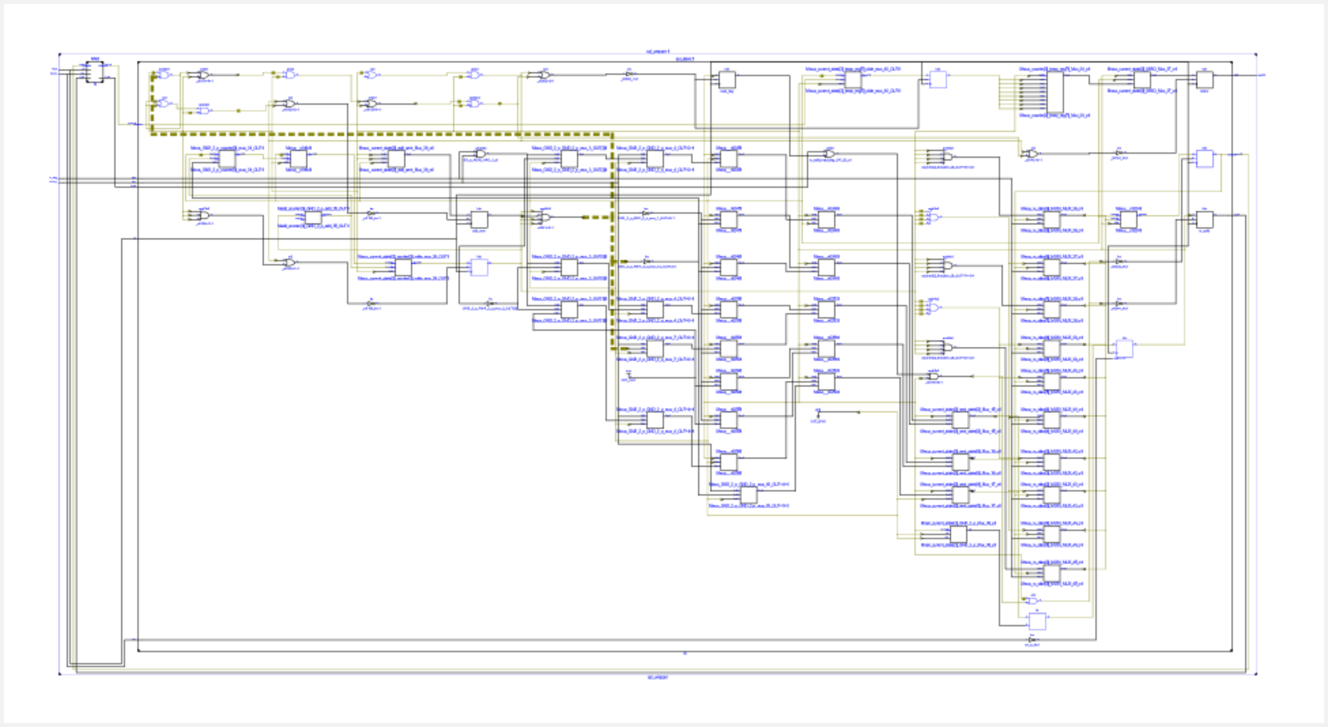
SPI_WRAPPER RTL Schematic



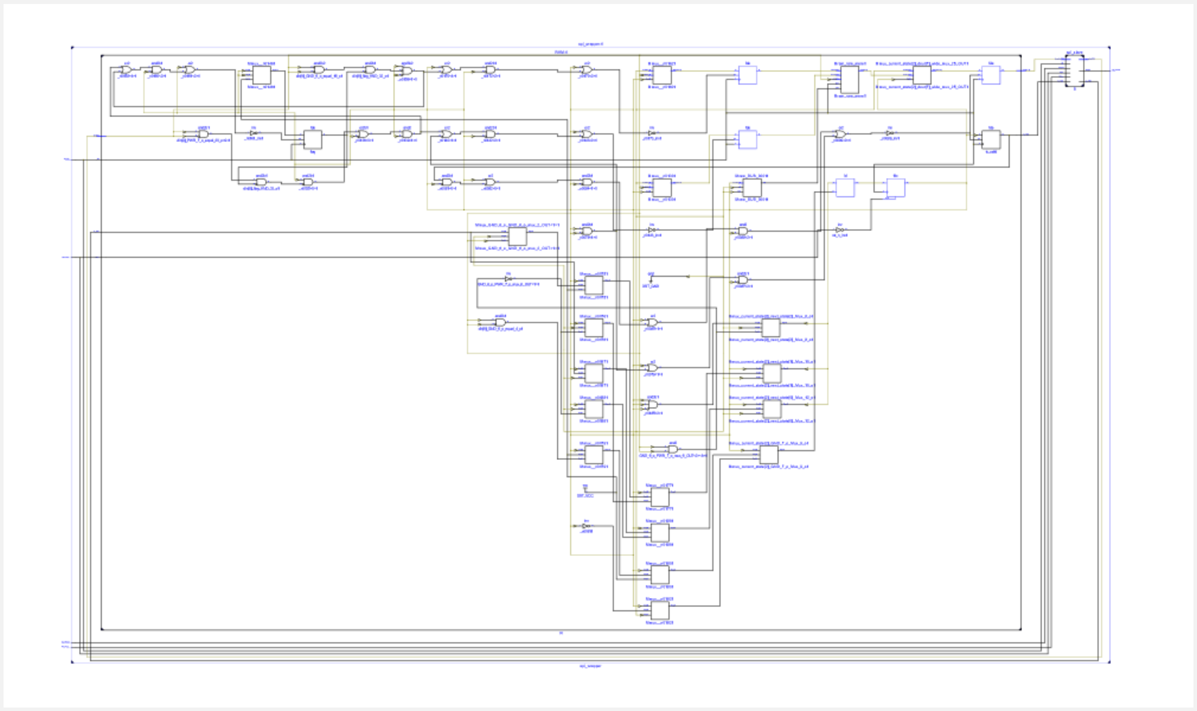
SPI_WRAPPER
Technology Schematic



SPI_SLAVE RTL Schematic



RAM RTL Schematic



- **Design GitHub Link :** HassanKhaled11/SPI_project (github.com)
- **LinkedIn :** Hassan Khaled | LinkedIn
- **Facebook :** https://m.facebook.com/100087870183210/