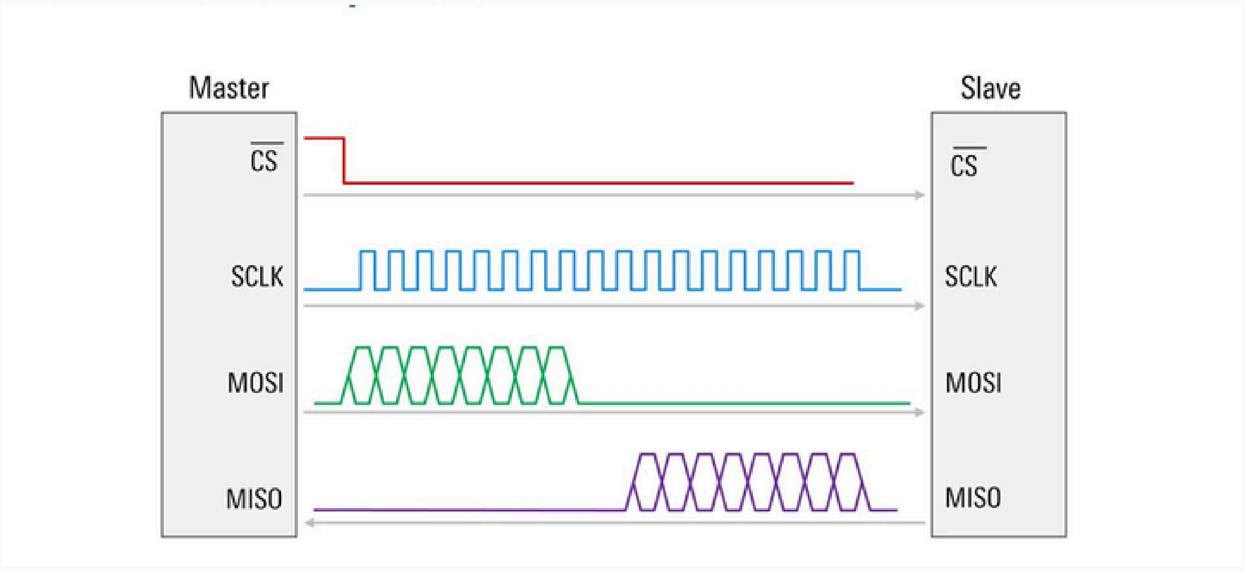
وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ ۖ وَسَتُرَدُّونَ إِلَىٰ عَالِمِ الْغَيْبِ وَالشَّهَادَةِ فَيُنَبِّئُكُم بِمَا كُنتُمْ تَعْمَلُونَ

PROJECT SPI protocol

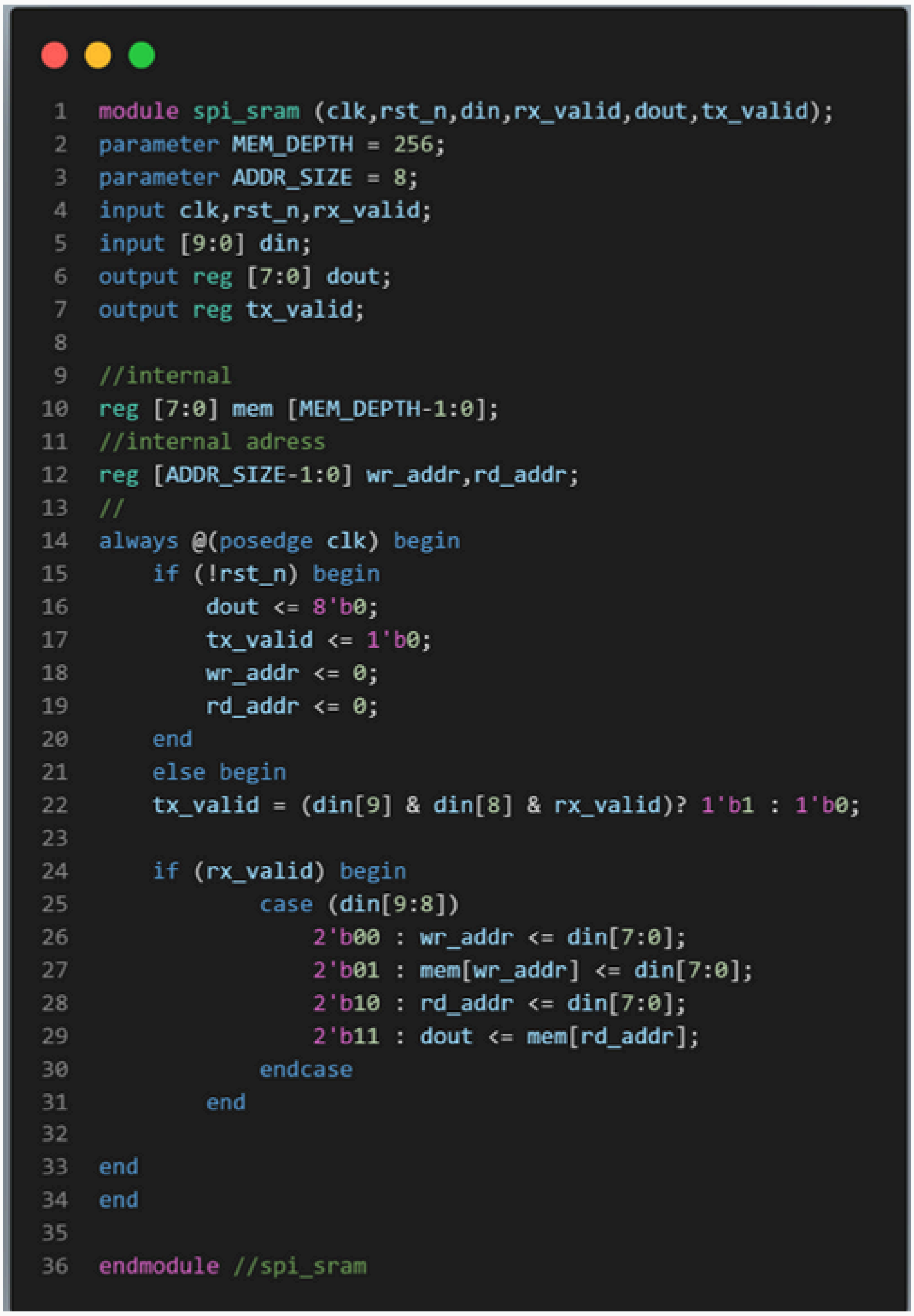
Prepared By:

Nour Elawadly

- The Serial Peripheral Interface (SPI) is a high-speed, full-duplex communication protocol widely used for short-distance data exchange between microcontrollers and peripheral devices such as sensors, memory chips, and display modules. This project focuses on the design and implementation of an SPI-based system, aiming to achieve efficient, reliable, and synchronized data transfer.

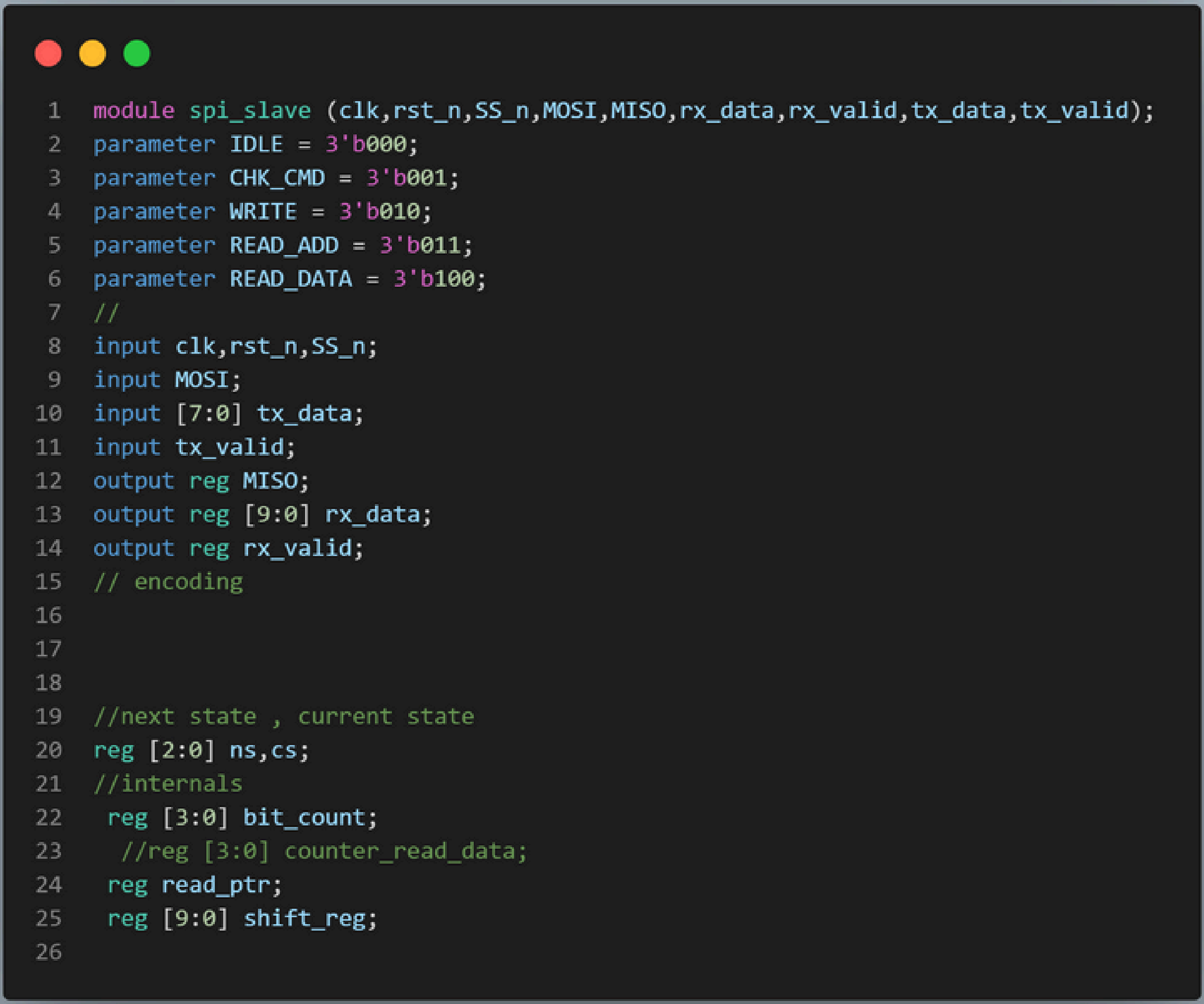


RAM Design



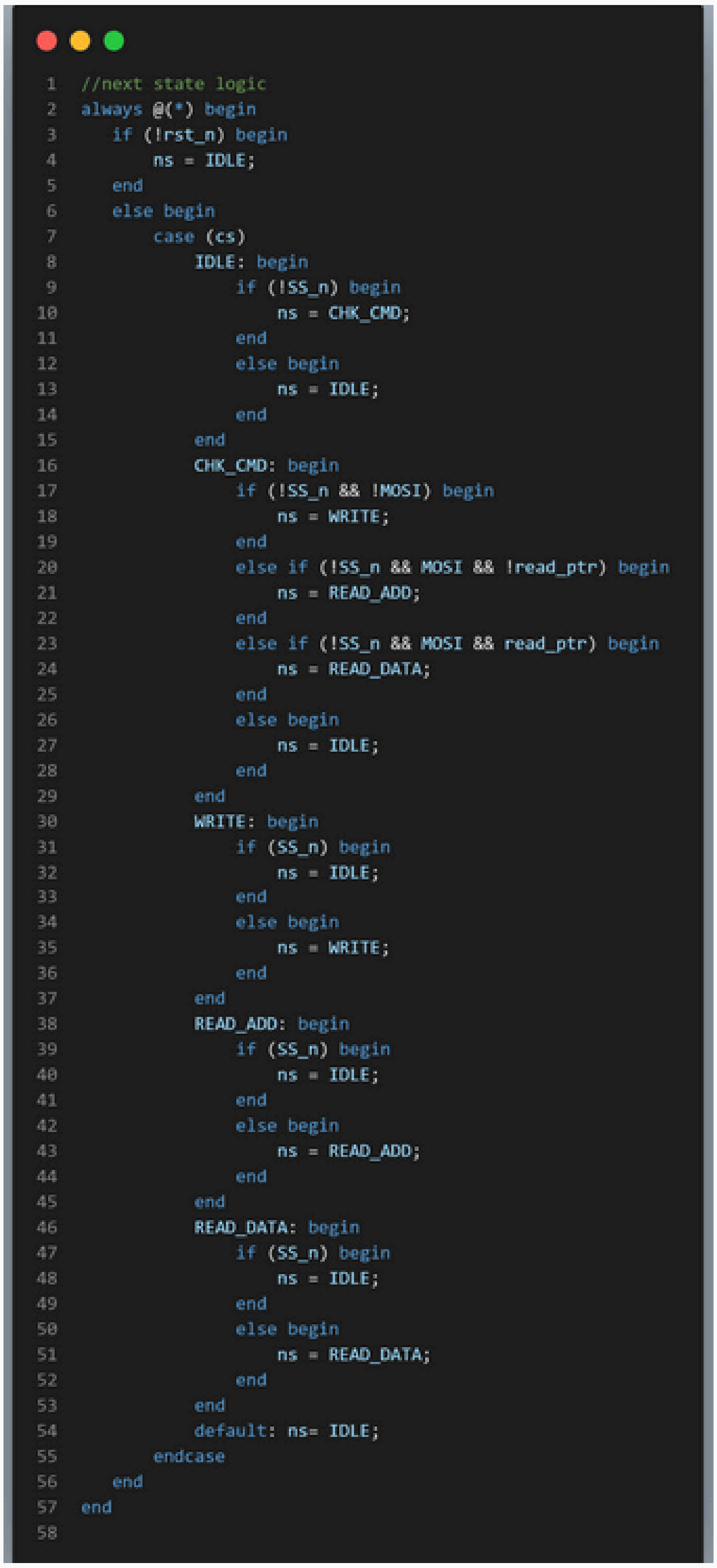
```verilog
module spi_sram (clk,rst_n,din,rx_valid,dout,tx_valid);
parameter MEM_DEPTH = 256;
parameter ADDR_SIZE = 8;
input clk,rst_n,rx_valid;
input [9:0] din;
output reg [7:0] dout;
output reg tx_valid;

//internal
reg [7:0] mem [MEM_DEPTH-1:0];
//internal adress
reg [ADDR_SIZE-1:0] wr_addr,rd_addr;
//
always @(posedge clk) begin
    if (!rst_n) begin
        dout <= 8'b0;
        tx_valid <= 1'b0;
        wr_addr <= 0;
        rd_addr <= 0;
    end
    else begin
    tx_valid = (din[9] & din[8] & rx_valid)? 1'b1 : 1'b0;

    if (rx_valid) begin
            case (din[9:8])
                2'b00 : wr_addr <= din[7:0];
                2'b01 : mem[wr_addr] <= din[7:0];
                2'b10 : rd_addr <= din[7:0];
                2'b11 : dout <= mem[rd_addr];
            endcase
        end

end
end

endmodule //spi_sram
```

SLAVE Design

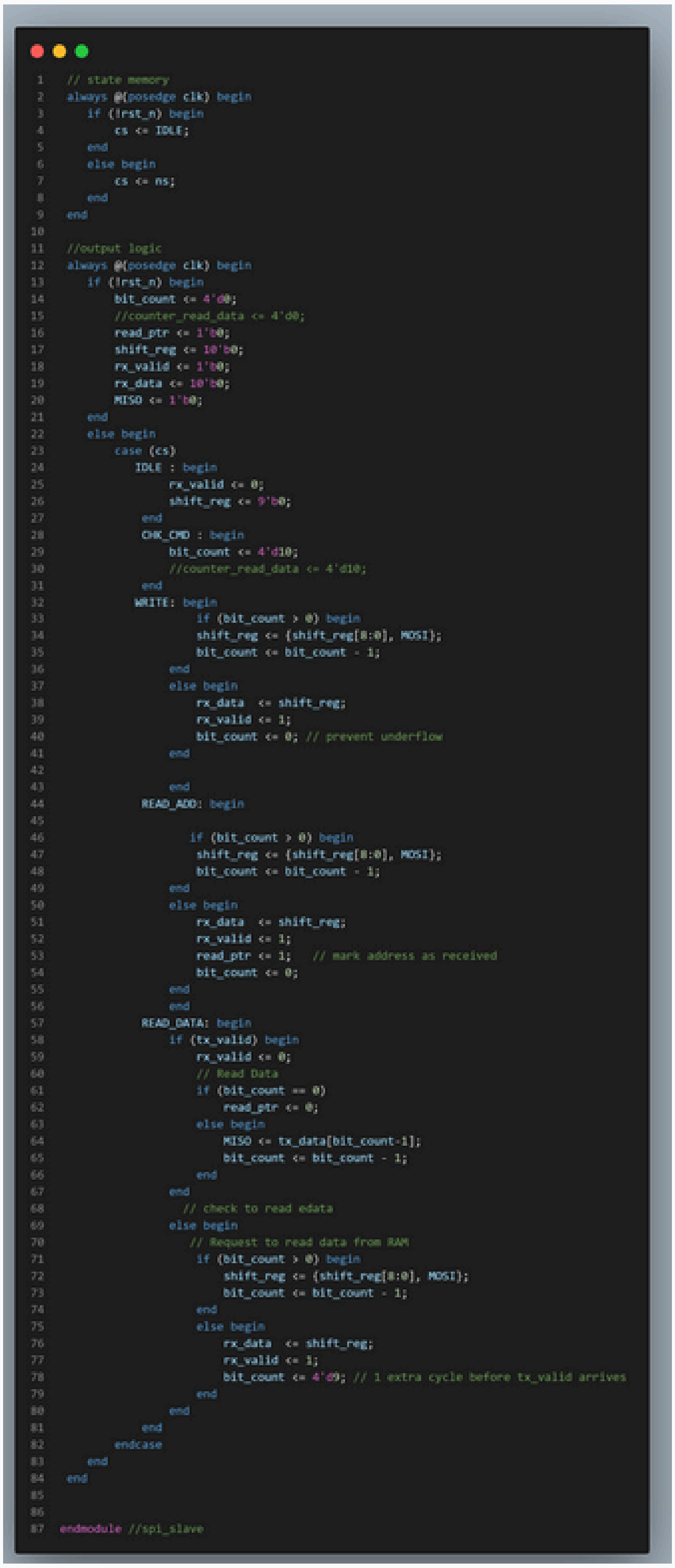


```
module spi_slave (clk,rst_n,SS_n,MOSI,MISO,rx_data,rx_valid,tx_data,tx_valid);
parameter IDLE = 3'b000;
parameter CHK_CMD = 3'b001;
parameter WRITE = 3'b010;
parameter READ_ADD = 3'b011;
parameter READ_DATA = 3'b100;
//
input clk,rst_n,SS_n;
input MOSI;
input [7:0] tx_data;
input tx_valid;
output reg MISO;
output reg [9:0] rx_data;
output reg rx_valid;
// encoding



//next state , current state
reg [2:0] ns,cs;
//internals
 reg [3:0] bit_count;
  //reg [3:0] counter_read_data;
 reg read_ptr;
 reg [9:0] shift_reg;

```

FSM Algorithm

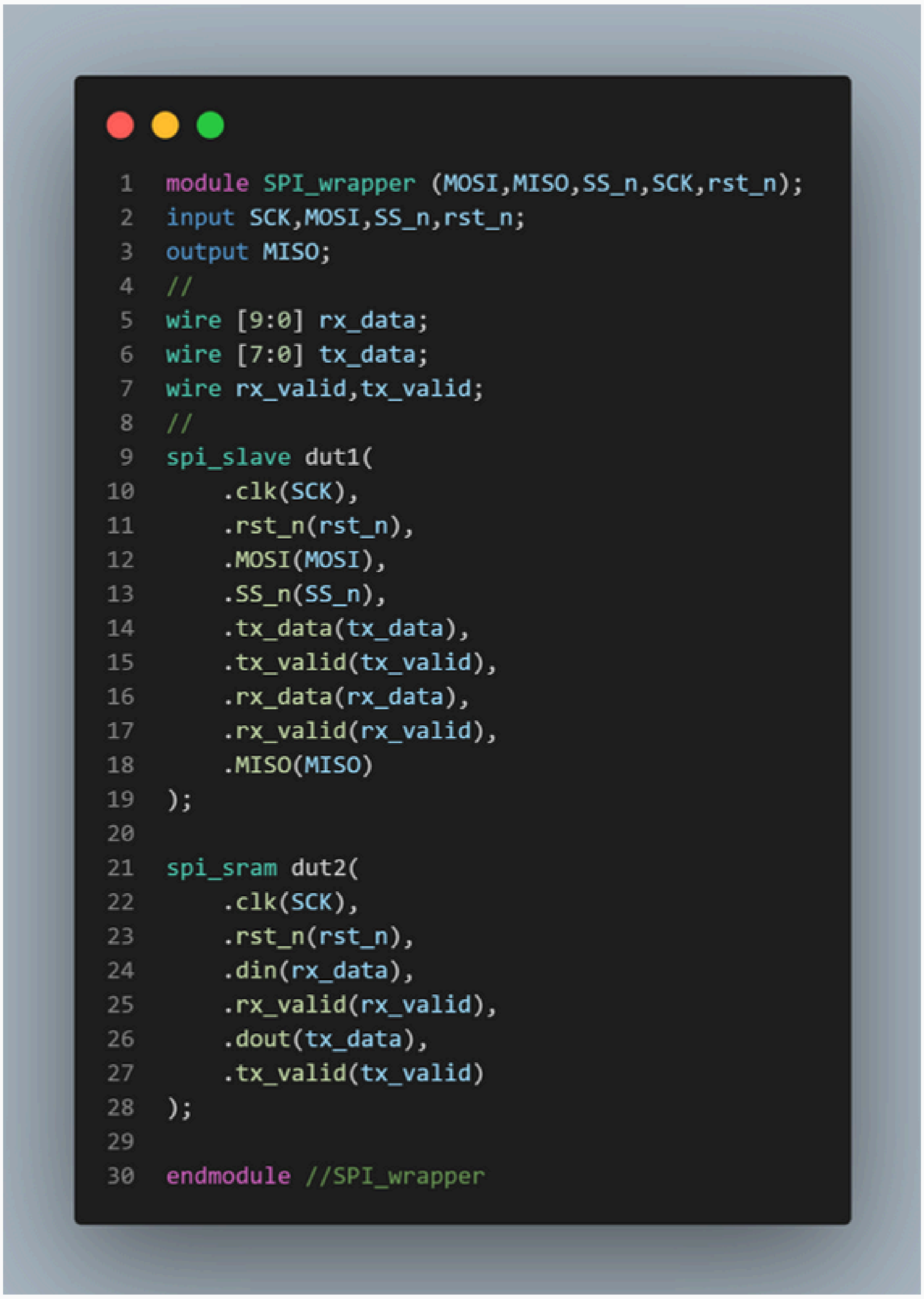


```
//next state logic
always @(*) begin
    if (!rst_n) begin
        ns = IDLE;
    end
    else begin
        case (cs)
            IDLE: begin
                if (!SS_n) begin
                    ns = CHK_CMD;
                end
                else begin
                    ns = IDLE;
                end
            end
            CHK_CMD: begin
                if (!SS_n && !MOSI) begin
                    ns = WRITE;
                end
                else if (!SS_n && MOSI && !read_ptr) begin
                    ns = READ_ADD;
                end
                else if (!SS_n && MOSI && read_ptr) begin
                    ns = READ_DATA;
                end
                else begin
                    ns = IDLE;
                end
            end
            WRITE: begin
                if (SS_n) begin
                    ns = IDLE;
                end
                else begin
                    ns = WRITE;
                end
            end
            READ_ADD: begin
                if (SS_n) begin
                    ns = IDLE;
                end
                else begin
                    ns = READ_ADD;
                end
            end
            READ_DATA: begin
                if (SS_n) begin
                    ns = IDLE;
                end
                else begin
                    ns = READ_DATA;
                end
            end
            default: ns= IDLE;
        endcase
    end
end
```



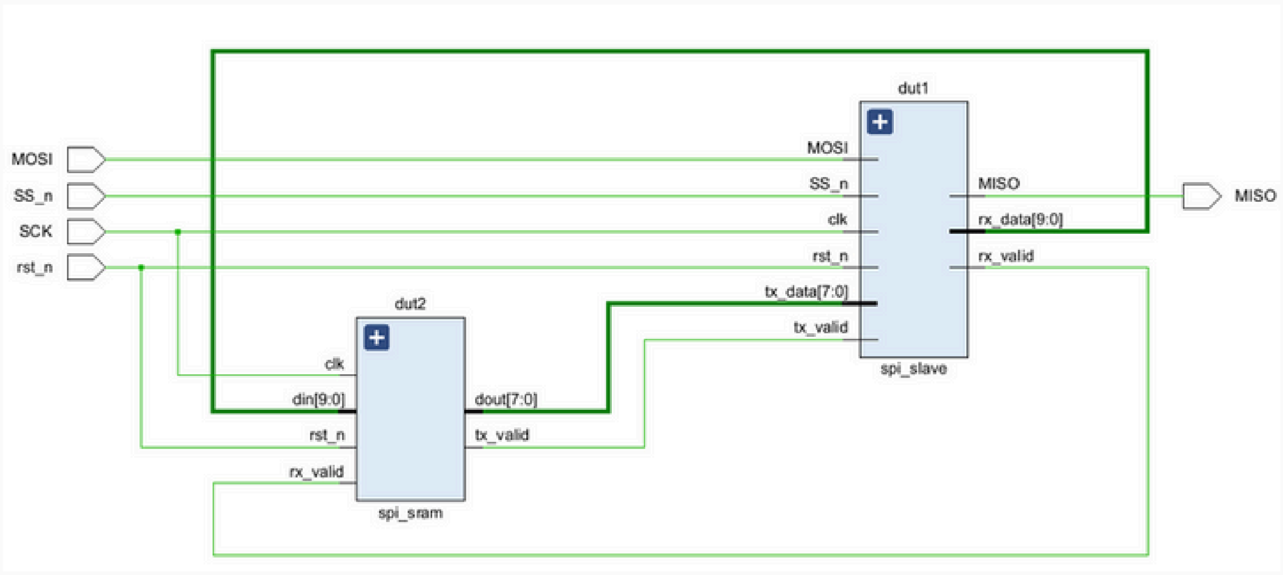
```verilog
// state memory
always @(posedge clk) begin
   if (!rst_n) begin
       cs <= IDLE;
   end
   else begin
       cs <= ns;
   end
end

//output logic
always @(posedge clk) begin
   if (!rst_n) begin
       bit_count <= 4'd0;
       //counter_read_data <= 4'd0;
       read_ptr <= 1'b0;
       shift_reg <= 10'b0;
       rx_valid <= 1'b0;
       rx_data <= 10'b0;
       MISO <= 1'b0;
   end
   else begin
       case (cs)
          IDLE : begin
               rx_valid <= 0;
               shift_reg <= 9'b0;
           end
           CHK_CMD : begin
               bit_count <= 4'd10;
               //counter_read_data <= 4'd10;
           end
          WRITE: begin
                   if (bit_count > 0) begin
                   shift_reg <= {shift_reg[8:0], MOSI};
                   bit_count <= bit_count - 1;
               end
               else begin
                   rx_data  <= shift_reg;
                   rx_valid <= 1;
                   bit_count <= 0; // prevent underflow
               end

               end
           READ_ADD: begin

                  if (bit_count > 0) begin
                   shift_reg <= {shift_reg[8:0], MOSI};
                   bit_count <= bit_count - 1;
               end
               else begin
                   rx_data  <= shift_reg;
                   rx_valid <= 1;
                   read_ptr <= 1;   // mark address as received
                   bit_count <= 0;
               end
               end
           READ_DATA: begin
               if (tx_valid) begin
                   rx_valid <= 0;
                   // Read Data
                   if (bit_count == 0)
                       read_ptr <= 0;
                   else begin
                       MISO <= tx_data[bit_count-1];
                       bit_count <= bit_count - 1;
                   end
               end
                 // check to read edata
               else begin
                  // Request to read data from RAM
                   if (bit_count > 0) begin
                       shift_reg <= {shift_reg[8:0], MOSI};
                       bit_count <= bit_count - 1;
                   end
                   else begin
                       rx_data  <= shift_reg;
                       rx_valid <= 1;
                       bit_count <= 4'd9; // 1 extra cycle before tx_valid arrives
                   end
               end
           end
       endcase
   end
end


endmodule //spi_slave
```

SPI wrapper

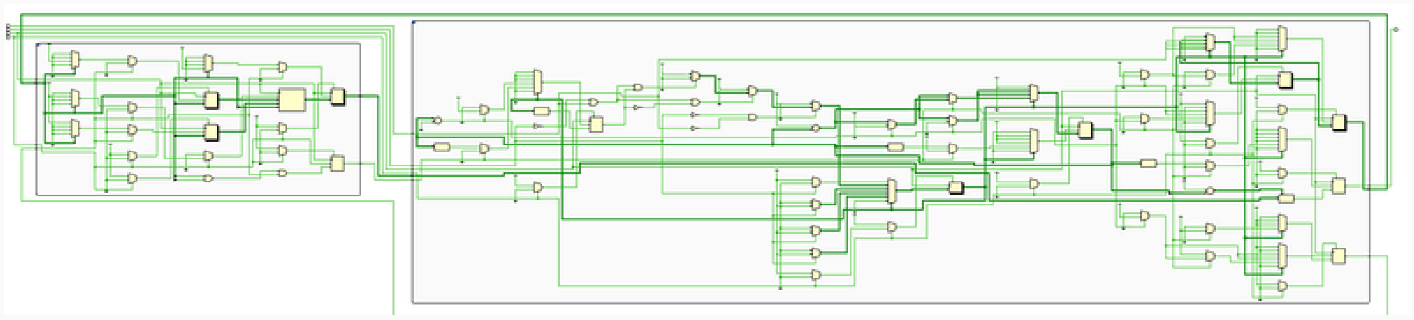


```
module SPI_wrapper (MOSI,MISO,SS_n,SCK,rst_n);
input SCK,MOSI,SS_n,rst_n;
output MISO;
//
wire [9:0] rx_data;
wire [7:0] tx_data;
wire rx_valid,tx_valid;
//
spi_slave dut1(
    .clk(SCK),
    .rst_n(rst_n),
    .MOSI(MOSI),
    .SS_n(SS_n),
    .tx_data(tx_data),
    .tx_valid(tx_valid),
    .rx_data(rx_data),
    .rx_valid(rx_valid),
    .MISO(MISO)
);

spi_sram dut2(
    .clk(SCK),
    .rst_n(rst_n),
    .din(rx_data),
    .rx_valid(rx_valid),
    .dout(tx_data),
    .tx_valid(tx_valid)
);

endmodule //SPI_wrapper
```

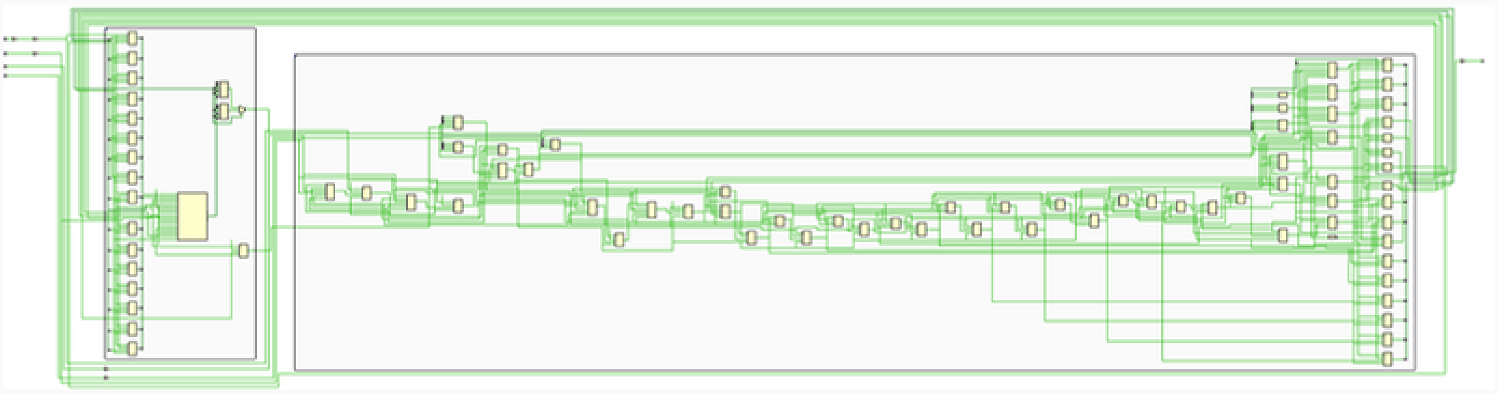
Elaborated



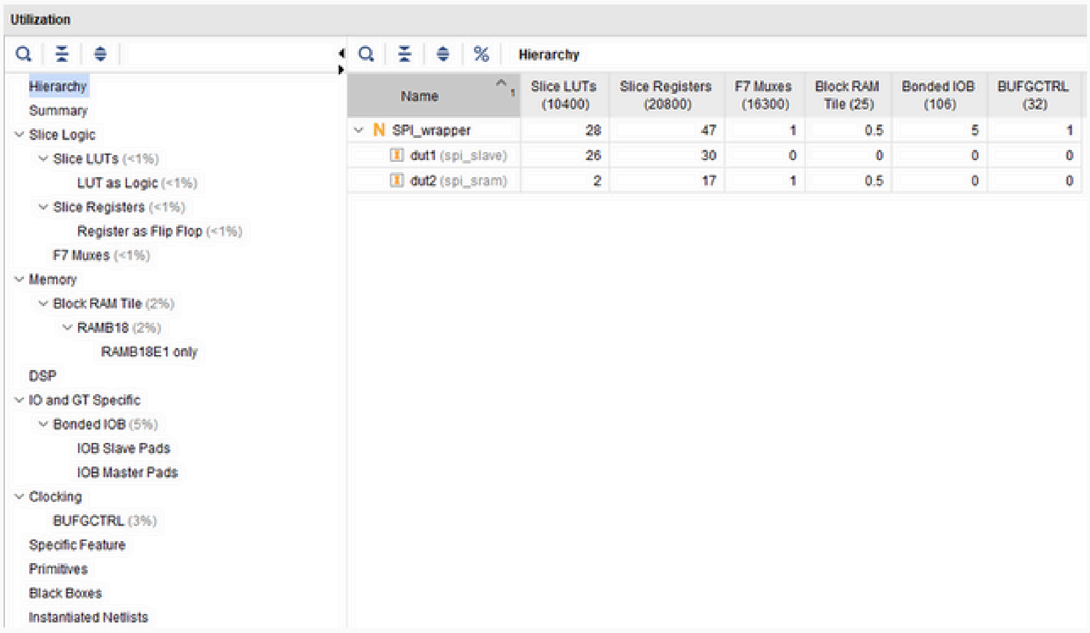
dut1
MOSI
SS_n
SCK
rst_n
MOSI
SS_n
clk
rst_n
tx_data[7:0]
tx_valid
spi_slave
MISO
rx_data[9:0]
rx_valid
MISO
dut2
clk
din[9:0]
rst_n
rx_valid
spi_sram
dout[7:0]
tx_valid



Synthesis



Utilization



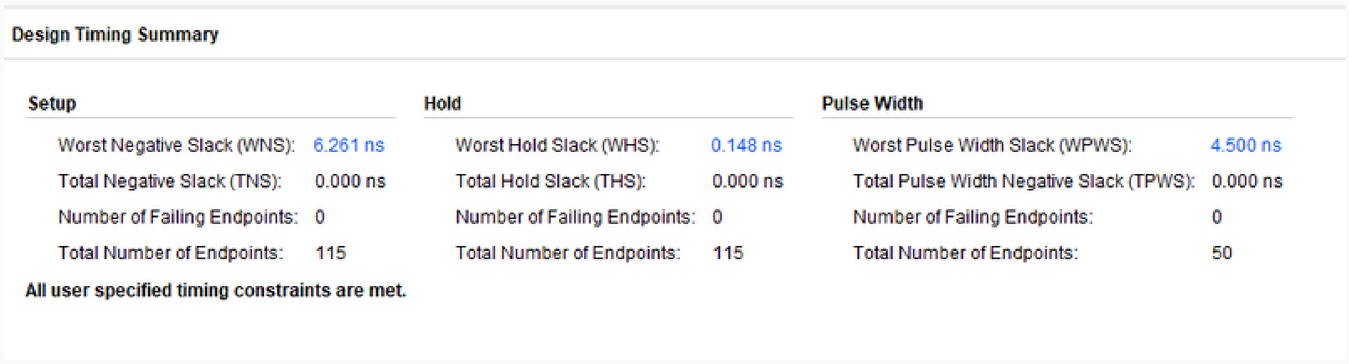
Utilization

Hierarchy

- Hierarchy
- Summary
- Slice Logic
  - Slice LUTs (<1%)
    - LUT as Logic (<1%)
  - Slice Registers (<1%)
    - Register as Flip Flop (<1%)
  - F7 Muxes (<1%)
- Memory
  - Block RAM Tile (2%)
    - RAMB18 (2%)
      - RAMB18E1 only
- DSP
- IO and GT Specific
  - Bonded IOB (5%)
    - IOB Slave Pads
    - IOB Master Pads
- Clocking
  - BUFGCTRL (3%)
- Specific Feature
- Primitives
- Black Boxes
- Instantiated Netlists

Name	Slice LUTs (10400)	Slice Registers (20800)	F7 Muxes (16300)	Block RAM Tile (25)	Bonded IOB (106)	BUFGCTRL (32)
N SPI_wrapper	28	47	1	0.5	5	1
dut1 (spi_slave)	26	30	0	0	0	0
dut2 (spi_sram)	2	17	1	0.5	0	0

Timing Summary

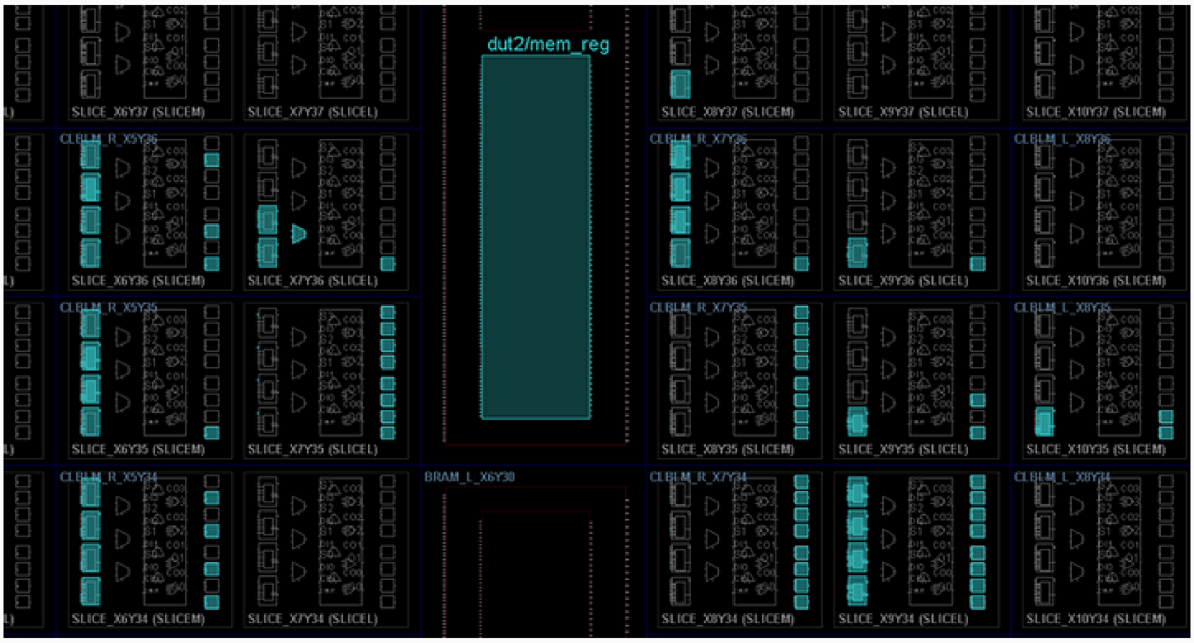


Design Timing Summary

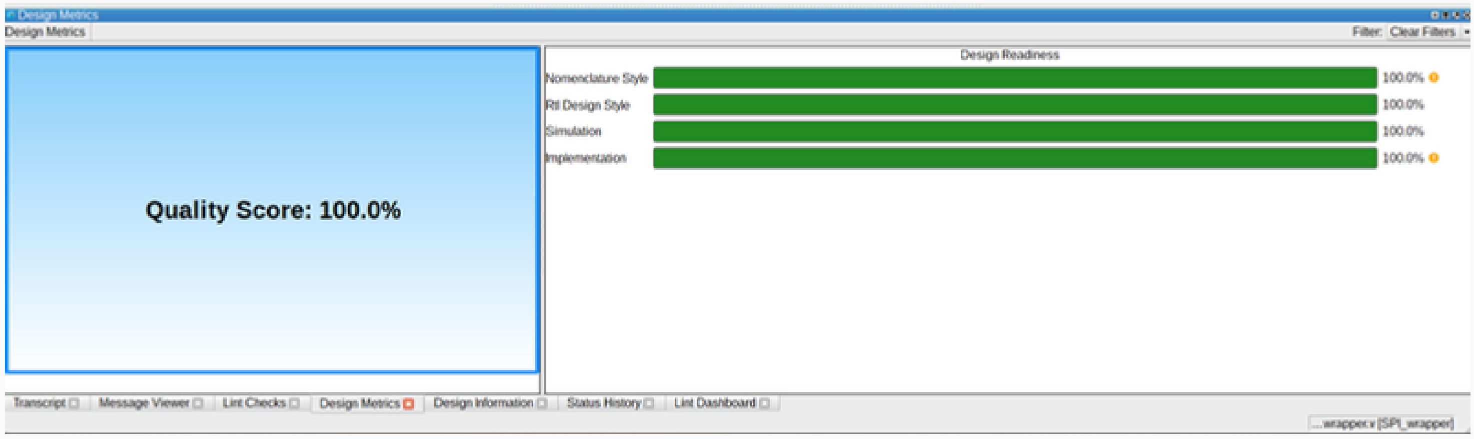
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.261 ns	Worst Hold Slack (WHS):	0.148 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	115	Total Number of Endpoints:	115	Total Number of Endpoints:	50

All user specified timing constraints are met.

Implementation



Linting results



Design Metrics

Design Metrics — Filter: Clear Filters

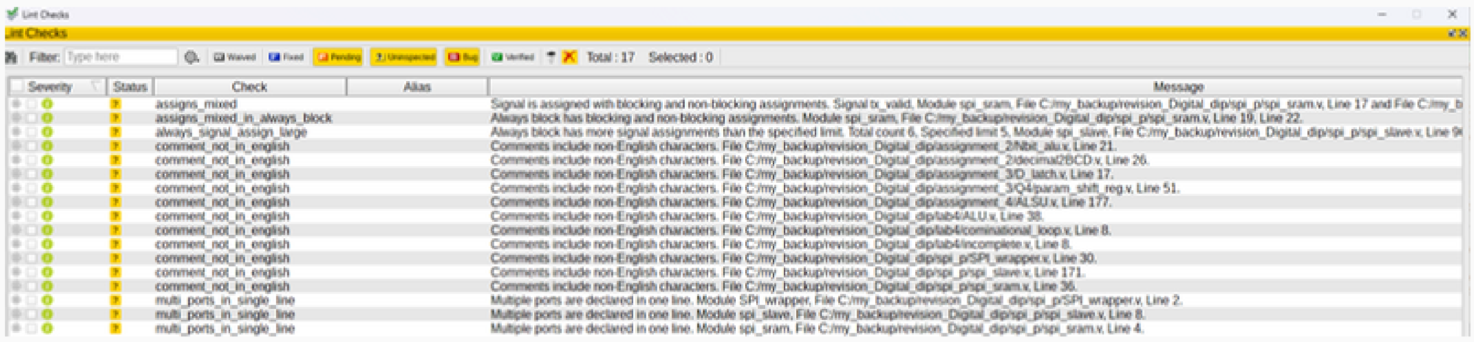
Quality Score: 100.0%

Design Readiness

Nomenclature Style	🟩	100.0%
Rtl Design Style	🟩	100.0%
Simulation	🟩	100.0%
Implementation	🟩	100.0%

Transcript | Message Viewer | Lint Checks | Design Metrics | Design Information | Status History | Lint Dashboard

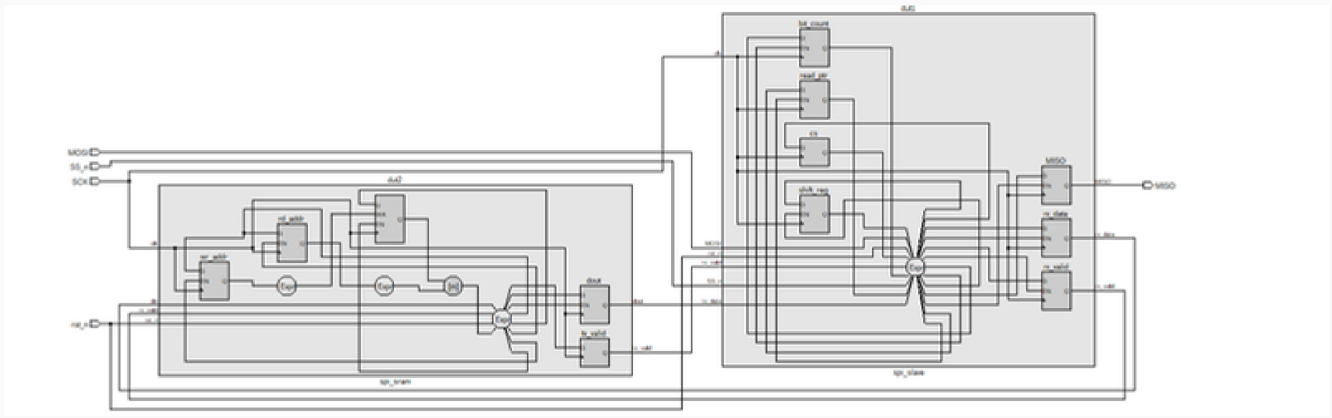
...wrapper.v [SPI_wrapper]



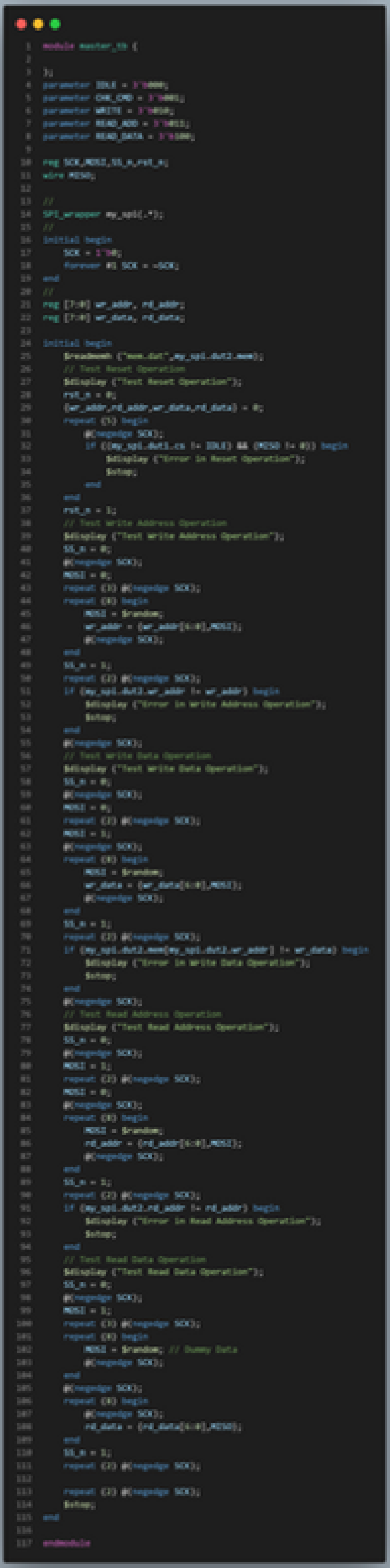
Lint Checks

Filter: Type here — Waived | Fixed | Pending | Uninspected | Bug | Verified — Total : 17 Selected : 0

Severity	Status	Check	Alias	Message
●	?	assigns_mixed		Signal is assigned with blocking and non-blocking assignments. Signal tx_valid, Module spi_sram, File C:/my_backup/revision_Digital_dip/spi_p/spi_sram.v, Line 17 and File C:/my_b
●	?	assigns_mixed_in_always_block		Always block has blocking and non-blocking assignments. Module spi_sram, File C:/my_backup/revision_Digital_dip/spi_p/spi_sram.v, Line 19, Line 22.
●	?	always_signal_assign_large		Always block has more signal assignments than the specified limit. Total count 6, Specified limit 5, Module spi_slave, File C:/my_backup/revision_Digital_dip/spi_p/spi_slave.v, Line 9
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/assignment_2/Nbit_alu.v, Line 21.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/assignment_2/decimal2BCD.v, Line 26.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/assignment_3/D_latch.v, Line 17.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/assignment_3/Q4/param_shift_reg.v, Line 51.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/assignment_4/ALSU.v, Line 177.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/lab4/ALU.v, Line 38.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/lab4/cominational_loop.v, Line 8.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/lab4/incomplete.v, Line 8.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/spi_p/SPI_wrapper.v, Line 30.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/spi_p/spi_slave.v, Line 171.
●	?	comment_not_in_english		Comments include non-English characters. File C:/my_backup/revision_Digital_dip/spi_p/spi_sram.v, Line 36.
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module SPI_wrapper, File C:/my_backup/revision_Digital_dip/spi_p/SPI_wrapper.v, Line 2.
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module spi_slave, File C:/my_backup/revision_Digital_dip/spi_p/spi_slave.v, Line 8.
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module spi_sram, File C:/my_backup/revision_Digital_dip/spi_p/spi_sram.v, Line 4.

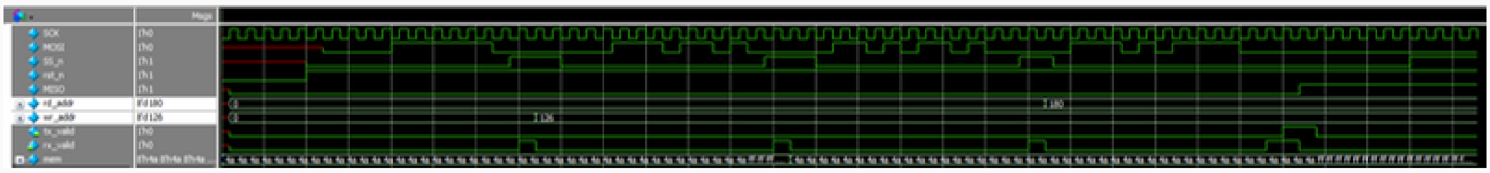


MASTER(tb)

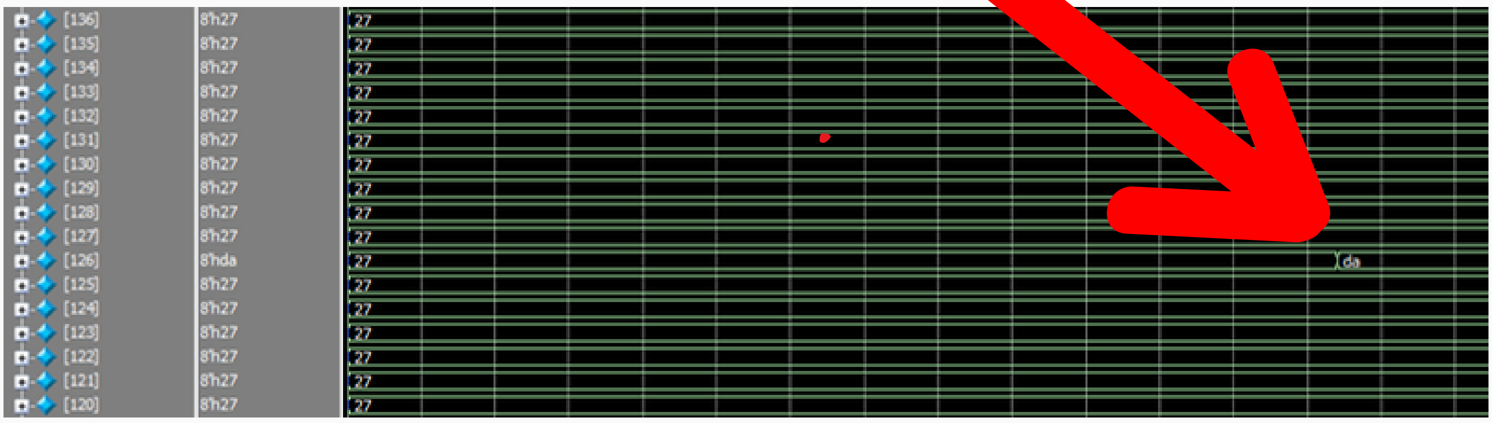


```verilog
module master_tb (

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

reg SCK,MOSI,SS_n,rst_n;
wire MISO;

//
SPI_wrapper my_spi(.*);
//
initial begin
    SCK = 1'b0;
    forever #1 SCK = ~SCK;
end
//
reg [7:0] wr_addr, rd_addr;
reg [7:0] wr_data, rd_data;

initial begin
    $readmemh ("mem.dat",my_spi.dut2.mem);
    // Test Reset Operation
    $display ("Test Reset Operation");
    rst_n = 0;
    {wr_addr,rd_addr,wr_data,rd_data} = 0;
    repeat (5) begin
        @(negedge SCK);
        if ((my_spi.dut1.cs != IDLE) && (MISO != 0)) begin
            $display ("Error in Reset Operation");
            $stop;
        end
    end
    rst_n = 1;
    // Test Write Address Operation
    $display ("Test Write Address Operation");
    SS_n = 0;
    @(negedge SCK);
    MOSI = 0;
    repeat (3) @(negedge SCK);
    repeat (8) begin
        MOSI = $random;
        wr_addr = {wr_addr[6:0],MOSI};
        @(negedge SCK);
    end
    SS_n = 1;
    repeat (2) @(negedge SCK);
    if (my_spi.dut2.wr_addr != wr_addr) begin
        $display ("Error in Write Address Operation");
        $stop;
    end
    @(negedge SCK);
    // Test Write Data Operation
    $display ("Test Write Data Operation");
    SS_n = 0;
    @(negedge SCK);
    MOSI = 0;
    repeat (2) @(negedge SCK);
    MOSI = 1;
    @(negedge SCK);
    repeat (8) begin
        MOSI = $random;
        wr_data = {wr_data[6:0],MOSI};
        @(negedge SCK);
    end
    SS_n = 1;
    repeat (2) @(negedge SCK);
    if (my_spi.dut2.mem[my_spi.dut2.wr_addr] != wr_data) begin
        $display ("Error in Write Data Operation");
        $stop;
    end
    @(negedge SCK);
    // Test Read Address Operation
    $display ("Test Read Address Operation");
    SS_n = 0;
    @(negedge SCK);
    MOSI = 1;
    repeat (2) @(negedge SCK);
    MOSI = 0;
    @(negedge SCK);
    repeat (8) begin
        MOSI = $random;
        rd_addr = {rd_addr[6:0],MOSI};
        @(negedge SCK);
    end
    SS_n = 1;
    repeat (2) @(negedge SCK);
    if (my_spi.dut2.rd_addr != rd_addr) begin
        $display ("Error in Read Address Operation");
        $stop;
    end
    // Test Read Data Operation
    $display ("Test Read Data Operation");
    SS_n = 0;
    @(negedge SCK);
    MOSI = 1;
    repeat (3) @(negedge SCK);
    repeat (8) begin
        MOSI = $random; // Dummy Data
        @(negedge SCK);
    end
    @(negedge SCK);
    repeat (8) begin
        @(negedge SCK);
        rd_data = {rd_data[6:0],MISO};
    end
    SS_n = 1;
    repeat (2) @(negedge SCK);

    repeat (2) @(negedge SCK);
    $stop;
end

endmodule
```

Simulation result



write in addr(126)



```
[136]  8'h27  27
[135]  8'h27  27
[134]  8'h27  27
[133]  8'h27  27
[132]  8'h27  27
[131]  8'h27  27
[130]  8'h27  27
[129]  8'h27  27
[128]  8'h27  27
[127]  8'h27  27
[126]  8'hda  27  da
[125]  8'h27  27
[124]  8'h27  27
[123]  8'h27  27
[122]  8'h27  27
[121]  8'h27  27
[120]  8'h27  27
```