Team Project

Digital Design

Team Name: Veri-cooked

Members : Ahmed Mahmoud Fathy

Omar Ahmed Gamal

Mohamed Abdou Ahmed

RTL Code:

SPI_Slave Code:

```
module SPI_Slave (MOSI,MISO,SS_n,clk,rst_n,rx_data,tx_data,rx_valid,tx_valid);
    parameter IDLE = 0;
    parameter CHK_CMD = 1;
    parameter WRITE = 2;
    parameter READ_DATA = 3;
    parameter READ_ADD = 4;
    reg [2:0] CS, NS;

    input MOSI;
    output reg MISO;
    input SS_n,clk,rst_n;
    input [7:0] tx_data;
    output reg [9:0] rx_data;
    output reg rx_valid;
    input tx_valid;
    reg [3:0] counter = 0;
    reg wr_or_rd;
    reg [1:0] rx_data_MSB;
    reg [9:0] rx_data_in;
    reg [7:0] tx_data_out;
    reg shifter_loaded;

    always @(posedge clk) begin
        if (~rst_n) begin
            rx_data_in <= 10'b0;
            rx_valid <= 1'b0;
            counter <= 0;
            rx_data <= 10'b0;
            MISO <= 1'b0;
            tx_data_out <= 8'b0;
            shifter_loaded <= 1'b0;
        end else if (~SS_n) begin
            if (counter == 0)
                wr_or_rd <= MOSI;
            else if (counter == 3) begin
                rx_data_MSB <= rx_data_in[1:0];
                rx_data_in <= {rx_data_in[8:0], MOSI};
```

```
            end else
                rx_data_in <= {rx_data_in[8:0], MOSI};

            counter <= counter + 1;

            if (CS == READ_DATA && tx_valid) begin
                if (!shifter_loaded) begin
                    tx_data_out <= tx_data;
                    MISO <= tx_data[7];
                    shifter_loaded <= 1'b1;
                end else begin
                    tx_data_out <= {tx_data_out[6:0], 1'b0};
                    MISO <= tx_data_out[7];
                end
            end else if (CS != READ_DATA) begin
                MISO <= 1'b0;
            end

        end else begin
            rx_valid <= 1'b0;
            counter <= 0;
            shifter_loaded <= 1'b0;
        end
        if (counter == 11) begin
            counter <= 0;
            if (CS == WRITE || CS == READ_ADD) begin
                rx_data <= rx_data_in;
                rx_valid <= 1'b1;
            end
        end
    end
```

```
// Current state logic
always @(posedge clk) begin
    if (~rst_n) begin
        CS <= IDLE;
    end else begin
        CS <= NS;
    end
end

// Next state logic
always @ (*) begin
    NS = CS;
    case (CS)
    IDLE:
        if(~SS_n)
            NS = CHK_CMD;
        else
            NS = IDLE;
    CHK_CMD:
        if (SS_n)
            NS = IDLE;
        else if (counter > 3) begin
            if (wr_or_rd == 1'b0)
                NS = WRITE;
            else if (wr_or_rd == 1'b1 && rx_data_MSB == 2'b10)
                NS = READ_ADD;
            else if (wr_or_rd == 1'b1 && rx_data_MSB == 2'b11)
                NS = READ_DATA;
            else
                NS = IDLE;
        end
        else
            NS = CHK_CMD;
```

```
            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

endmodule
```

RAM code:

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

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

always @(posedge clk) begin
    if (~rst_n) begin
        dout <= 0;
        wr_addr <=0;
        rd_addr <=0;
    end else if (rx_valid) begin
        case (din[9])
            1'b0: begin
                if (din[8] == 0)
                    wr_addr <= din[7:0];
                else
                    mem [wr_addr] <= din[7:0];
            end
            1'b1: begin
               if (din[8] == 0)
                    rd_addr <= din[7:0];
                else
                    dout <= mem[rd_addr];
                    tx_valid <= 1'b1;
            end

        endcase
    end
end
endmodule
```

SPI_RAM_Wrapper code:

```
module SPI_RAM_wrapper (
    input  MOSI,
    output MISO,
    input  SS_n,
    input  clk,
    input  rst_n
);
    wire [9:0] rx_data;
    wire       rx_valid;
    wire [7:0] tx_data;
    wire       tx_valid;

    // Instantiate SPI Slave
    SPI_Slave spi_slave_inst (
        .MOSI(MOSI),
        .MISO(MISO),
        .SS_n(SS_n),
        .clk(clk),
        .rst_n(rst_n),
        .rx_data(rx_data),
        .tx_data(tx_data),
        .rx_valid(rx_valid),
        .tx_valid(tx_valid)
    );
    RAM ram_inst (
        .din(rx_data),
        .clk(clk),
        .rst_n(rst_n),
        .rx_valid(rx_valid),
        .dout(tx_data),
        .tx_valid(tx_valid)
    );
endmodule
```

Testbench:

```
module SPI_Slave_tb;
    reg MOSI;
    reg SS_n;
    reg clk;
    reg rst_n;
    reg [7:0] tx_data;
    reg tx_valid;

    wire MISO;
    wire [9:0] rx_data;
    wire rx_valid;

    SPI_Slave uut (
        .MOSI(MOSI),
        .MISO(MISO),
        .SS_n(SS_n),
        .clk(clk),
        .rst_n(rst_n),
        .rx_data(rx_data),
        .tx_data(tx_data),
        .rx_valid(rx_valid),
        .tx_valid(tx_valid)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        MOSI = 0;
        SS_n = 1;
        rst_n = 0;
        tx_data = 8'hBE;
        tx_valid = 0;
```

```
$display("--- Starting SPI Slave Test Bench (Original Code) ---");

#10 rst_n = 1;
$display("Time: %0t, Reset de-asserted. Current State (CS): %0d", $time, uut.CS);
#20;

// 2. Write Address Command

$display("\n--- Test Case: Write Address (0x0A) ---");
SS_n = 0;
$display("Time: %0t, SS_n asserted. Starting Write Address command.", $time);
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
#10 SS_n = 1;
$display("Time: %0t, SS_n de-asserted. Write Address Command finished.", $time);
$display("Time: %0t, rx_data: %b (Expected: 10'b0000010100), rx_valid: %b", $time, rx_data, rx_valid);
#20;
```

```
// 3. Write Data Command

$display("\n--- Test Case: Write Data (0x6D) ---");
SS_n = 0;
$display("Time: %0t, SS_n asserted. Starting Write Data command.", $time);
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
#10 SS_n = 1;
$display("Time: %0t, SS_n de-asserted. Write Data Command finished.", $time);
$display("Time: %0t, rx_data: %b (Expected: 10'b0110110101), rx_valid: %b", $time, rx_data, rx_valid);
#20;
```

```
// 4. Read Address Command

$display("\n--- Test Case: Read Address (0x2B) ---");
SS_n = 0;
$display("Time: %0t, SS_n asserted. Starting Read Address command.", $time);
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
#10 SS_n = 1;
$display("Time: %0t, SS_n de-asserted. Read Address Command finished.", $time);
$display("Time: %0t, rx_data: %b (Expected: 10'b0010101110), rx_valid: %b", $time, rx_data, rx_valid);
#20;
```

```
// 5. Read Data Command

$display("\n--- Test Case: Read Data (Slave outputs 0xBE) ---");
SS_n = 0;
$display("Time: %0t, SS_n asserted. Starting Read Data command.", $time);
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 0;
@(negedge clk) MOSI = 1;
@(negedge clk) MOSI = 1;
#5;
tx_valid = 1;
$display("Time: %0t, tx_valid asserted. Slave should start sending data on MISO.", $time);
repeat (8) @(negedge clk) $display("Time: %0t, MISO: %b", $time, MISO);
#10 tx_valid = 0;
#10 SS_n = 1;
$display("Time: %0t, SS_n de-asserted. Read Data Command finished.", $time);
$display("Time: %0t, rx_data: %b, rx_valid: %b", $time, rx_data, rx_valid);
#20;

$display("\n--- Test Bench Finished ---");
$stop;
```

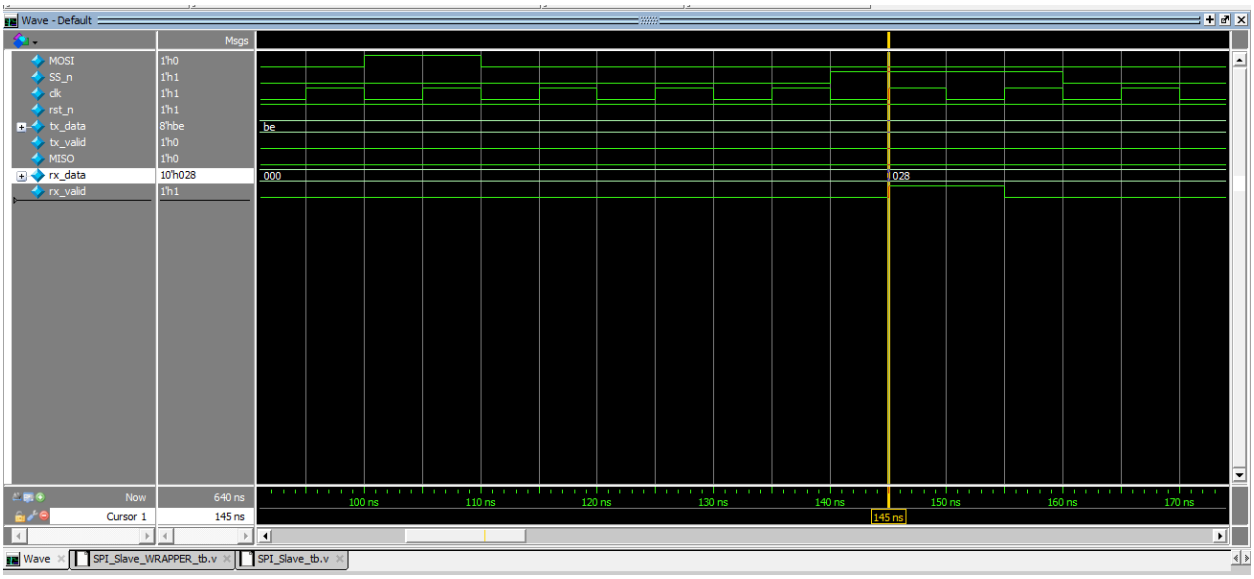
```
initial begin
    $monitor("Time: %0t, CS: %0d, NS: %0d, MOSI: %b, MISO: %b, SS_n: %b, rst_n: %b, counter: %0d, wr_or_rd: %b, rx_data_MSB: %b, rx_data_in: %b, rx_data: %b, rx_valid: %b, tx_data: %h, tx_valid: %b",
             $time, uut.CS, uut.NS, MOSI, MISO, SS_n, rst_n, uut.counter, uut.wr_or_rd, uut.rx_data_MSB, uut.rx_data_in, rx_data, rx_valid, tx_data, tx_valid);
end
endmodule
```

DO File:

```
vlib work
vlog SPI_Slave.v SPI_Slave_tb.v
vsim -voptargs=+acc work.SPI_Slave_tb
add wave *
run -all
#quit -sim
```

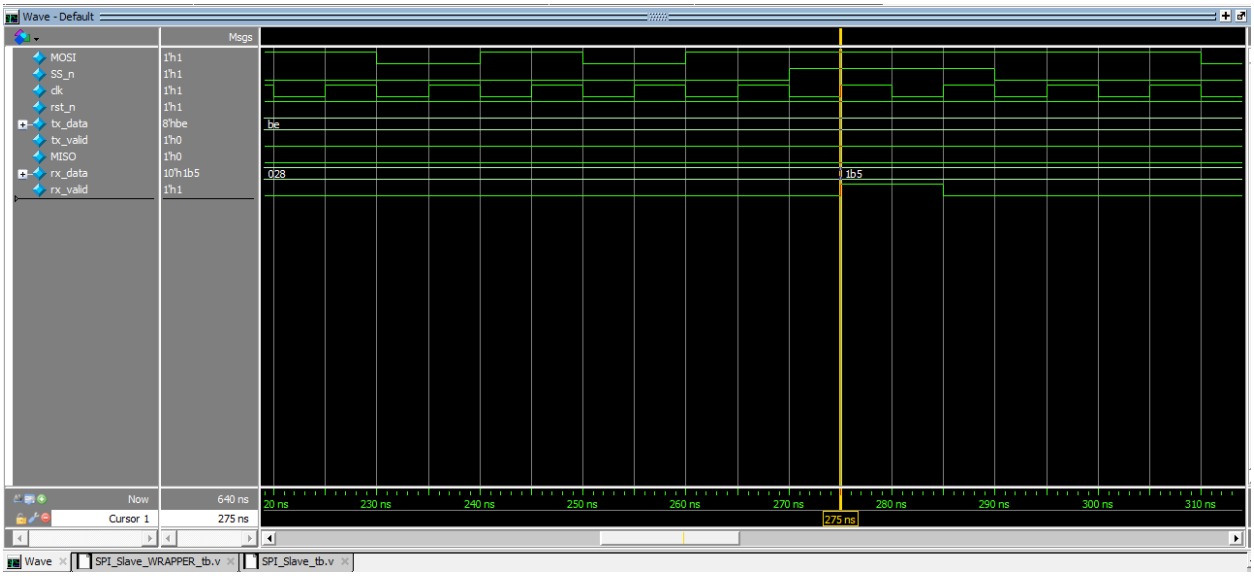
Test 1:

Case1: write address



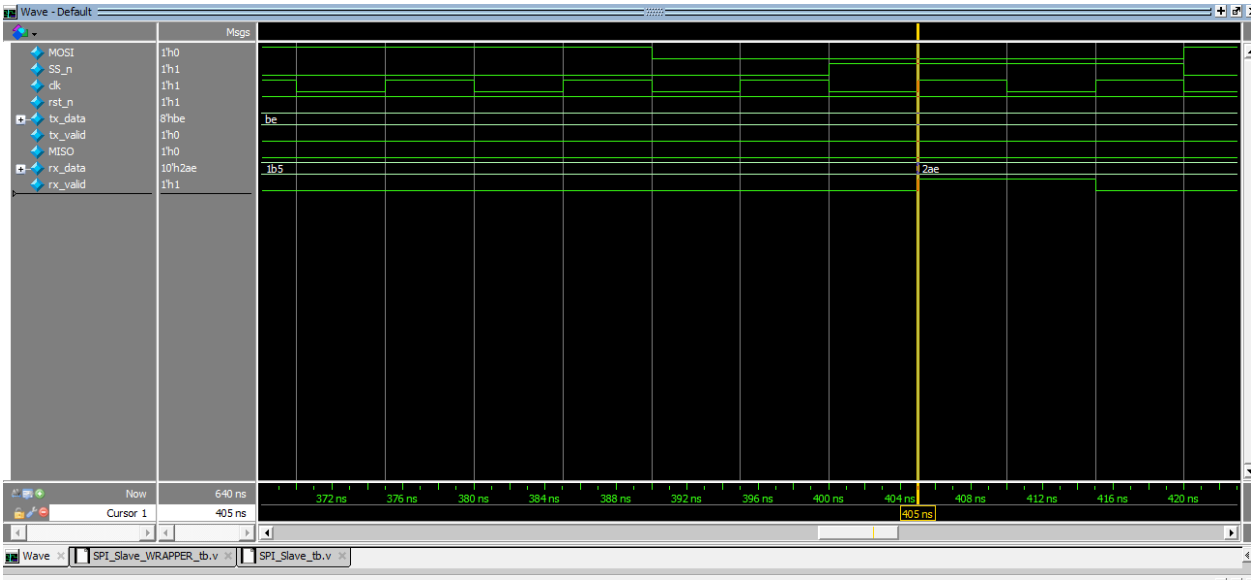

```
#
# --- Test Case: Write Address (0x0A) ---
# Time: 30, SS_n asserted. Starting Write Address command.
# Time: 30, CS: 0, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 0, wr_or_rd: x, rx_data_MSB: xx, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 35, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 1, wr_or_rd: 0, rx_data_MSB: xx, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 45, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 0, rx_data_MSB: xx, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 55, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 0, rx_data_MSB: xx, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 65, CS: 1, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 75, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 80, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 85, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000001, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 90, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000001, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 95, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000010, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 100, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000010, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 105, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000101, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 110, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000000101, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 115, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000001010, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 125, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000010100, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 135, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 11, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 140, SS_n de-asserted. Write Address Command finished.
# Time: 140, rx_data: 0000000000 (Expected: 10'b0000010100), rx_valid: 0
# Time: 140, CS: 2, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 11, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000000000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 145, CS: 0, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000101000, rx_valid: 1, tx_data: be, tx_valid: 0
# Time: 155, CS: 0, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
```

Case2: write data



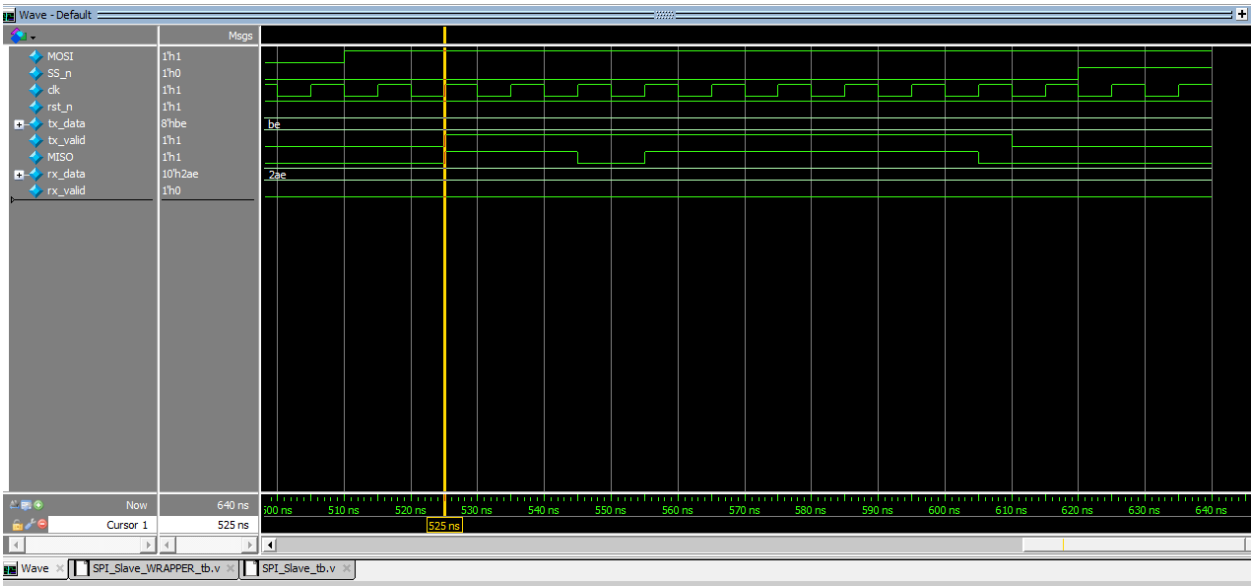

```
# --- Test Case: Write Data (0x6D) ---
# Time: 160, SS_n asserted. Starting Write Data command.
# Time: 160, CS: 0, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 165, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 1, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0000101000, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 175, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0001010000, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 180, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0001010000, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 185, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 0, rx_data_MSB: 00, rx_data_in: 0010100001, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 195, CS: 1, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0101000011, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 200, CS: 1, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0101000011, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 205, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 1010000110, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 210, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 1010000110, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 215, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0100001101, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 225, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 1000011011, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 230, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 1000011011, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 235, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0000110110, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 240, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0000110110, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 245, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0001101101, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 250, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0001101101, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 255, CS: 2, NS: 2, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0011011010, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 260, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0011011010, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 265, CS: 2, NS: 2, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 11, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 270, SS_n de-asserted. Write Data Command finished.
# Time: 270, rx_data: 0000101000 (Expected: 10'b0110110101), rx_valid: 0
# Time: 270, CS: 2, NS: 0, MOSI: 1, MISO: 0, SS_n: 1, rst_n: 1, counter: 11, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0000101000, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 275, CS: 0, NS: 0, MOSI: 1, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0110110101, rx_valid: 1, tx_data: be, tx_valid: 0
# Time: 285, CS: 0, NS: 0, MOSI: 1, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
```

Case3 : read address




```
# --- Test Case: Read Address (0x2B) ---
# Time: 290, SS_n asserted. Starting Read Address command.
# Time: 290, CS: 0, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 0, wr_or_rd: 0, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 295, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 1, wr_or_rd: 1, rx_data_MSB: 01, rx_data_in: 0110110101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 305, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 1, rx_data_MSB: 01, rx_data_in: 1101101011, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 310, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 1, rx_data_MSB: 01, rx_data_in: 1101101011, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 315, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 1, rx_data_MSB: 01, rx_data_in: 1011010110, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 320, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 1, rx_data_MSB: 01, rx_data_in: 1011010110, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 325, CS: 1, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0110101101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 330, CS: 1, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0110101101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 335, CS: 4, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1101011010, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 340, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1101011010, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 345, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010110101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 350, CS: 4, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010110101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 355, CS: 4, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0101101010, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 360, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0101101010, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 365, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1011010101, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 375, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0110101011, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 385, CS: 4, NS: 4, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1101010111, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 390, CS: 4, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1101010111, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 395, CS: 4, NS: 4, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 11, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 400, SS_n de-asserted. Read Address Command finished.
# Time: 400, rx_data: 0110110101 (Expected: 10'b0010101110), rx_valid: 0
# Time: 400, CS: 4, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 11, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 0110110101, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 405, CS: 0, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 1010101110, rx_valid: 1, tx_data: be, tx_valid: 0
# Time: 415, CS: 0, NS: 0, MOSI: 0, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
#
```

Case4: read data

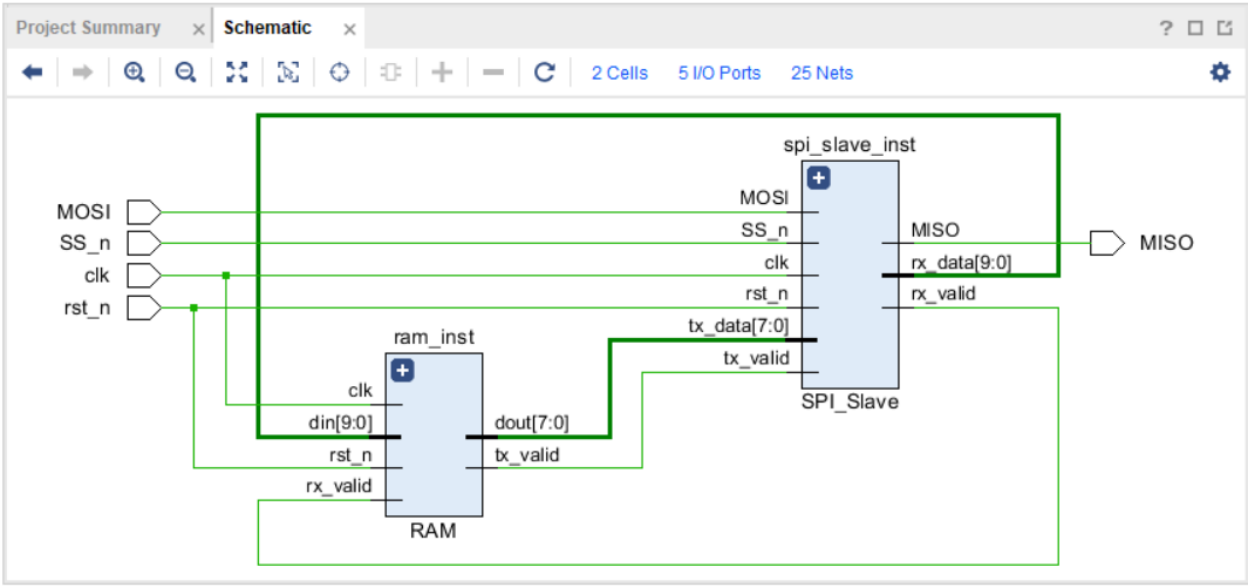


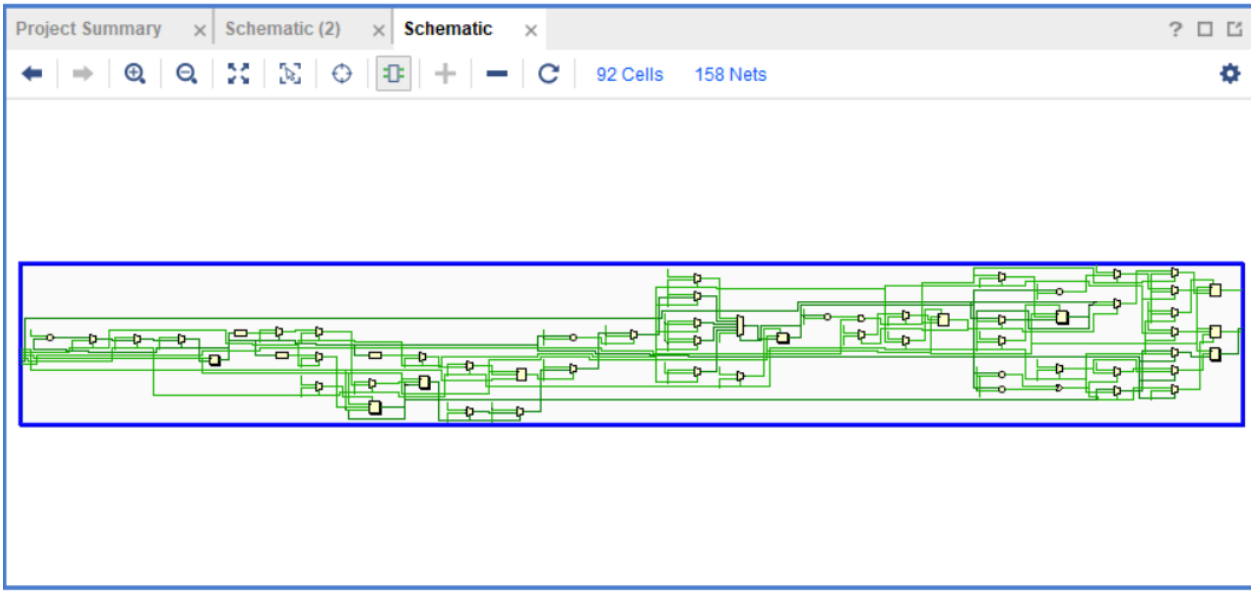

```
# --- Test Case: Read Data (Slave outputs 0xBE) ---
# Time: 420, SS_n asserted. Starting Read Data command.
# Time: 420, CS: 0, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 0, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 425, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 1, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010101110, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 435, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 0101011101, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 445, CS: 1, NS: 1, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010111011, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 450, CS: 1, NS: 1, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 1, rx_data_MSB: 10, rx_data_in: 1010111011, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 455, CS: 1, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0101110110, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 465, CS: 3, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1011101100, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 475, CS: 3, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0111011000, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 485, CS: 3, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1110110000, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 495, CS: 3, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1101100000, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 505, CS: 3, NS: 3, MOSI: 0, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1011000000, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 510, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 9, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1011000000, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 515, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 10, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0110000001, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 525, tx_valid asserted. Slave should start sending data on MISO.
# Time: 525, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 11, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1100000011, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 530, MISO: 1
# Time: 535, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 0, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1000000111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 540, MISO: 1
# Time: 545, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 1, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1000000111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 550, MISO: 0
# Time: 555, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 2, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0000001111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 560, MISO: 1
# Time: 565, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 3, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0000011111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 570, MISO: 1
# Time: 575, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 4, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0000111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 580, MISO: 1
# Time: 585, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 5, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0001111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 590, MISO: 1
# Time: 595, CS: 3, NS: 3, MOSI: 1, MISO: 1, SS_n: 0, rst_n: 1, counter: 6, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0011111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 600, MISO: 1
# Time: 605, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0111111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 1
# Time: 610, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 7, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 0111111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 615, CS: 3, NS: 3, MOSI: 1, MISO: 0, SS_n: 0, rst_n: 1, counter: 8, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1111111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 620, SS_n de-asserted. Read Data Command finished.
# Time: 620, rx_data: 1010101110, rx_valid: 0
# Time: 620, CS: 3, NS: 0, MOSI: 1, MISO: 0, SS_n: 1, rst_n: 1, counter: 8, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1111111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
# Time: 625, CS: 0, NS: 0, MOSI: 1, MISO: 0, SS_n: 1, rst_n: 1, counter: 0, wr_or_rd: 1, rx_data_MSB: 11, rx_data_in: 1111111111, rx_data: 1010101110, rx_valid: 0, tx_data: be, tx_valid: 0
#
# --- Test Bench Finished ---
```

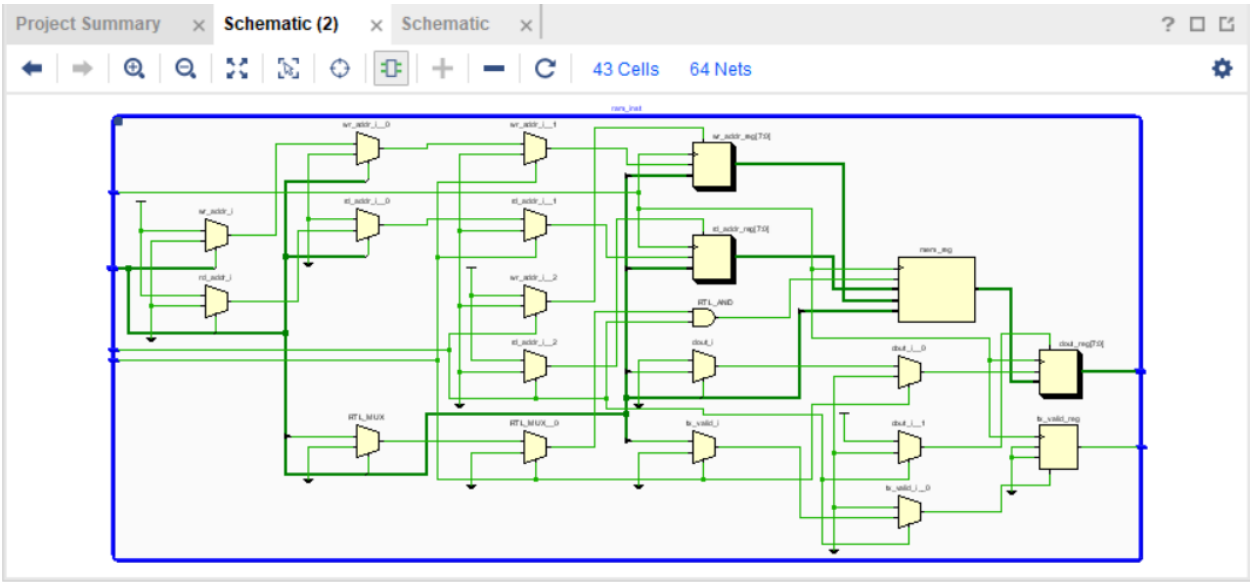
Constraint file

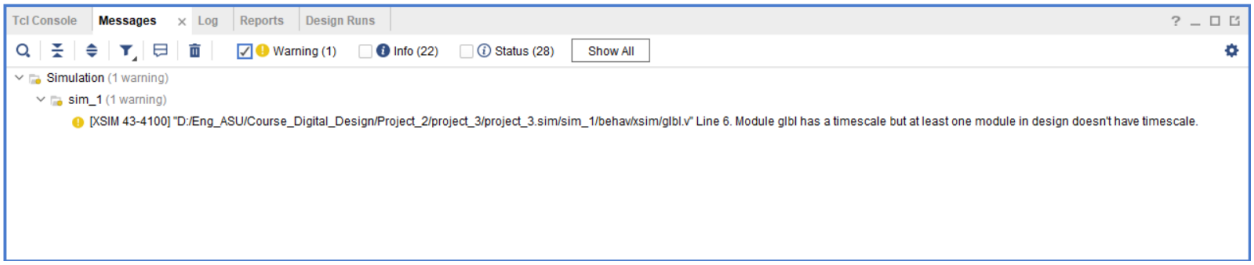
```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports rst_n]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports SS_n]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports MOSI]
#set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
#set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
#set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
#set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
#set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
#set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
#set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
#set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
#set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]


## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {MISO}]
#set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {led[1]}]
#set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {led[2]}]
#set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {led[3]}]
#set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {led[4]}]
#set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {led[5]}]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {led[6]}]
#set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {led[7]}]
#set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {led[8]}]
#set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports {led[9]}]
```

Elaboration:

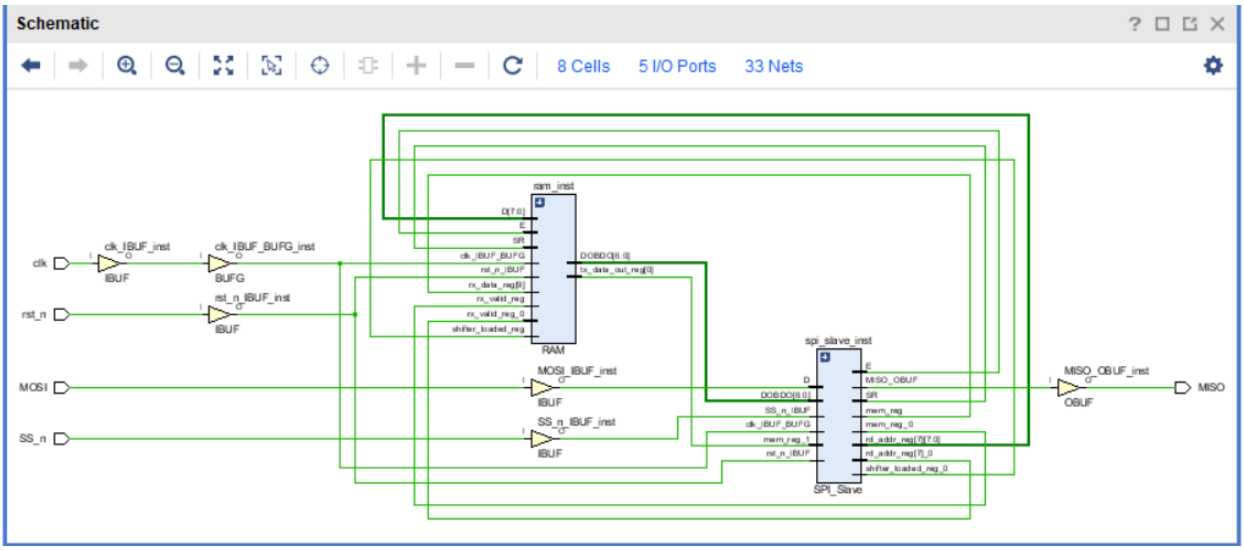


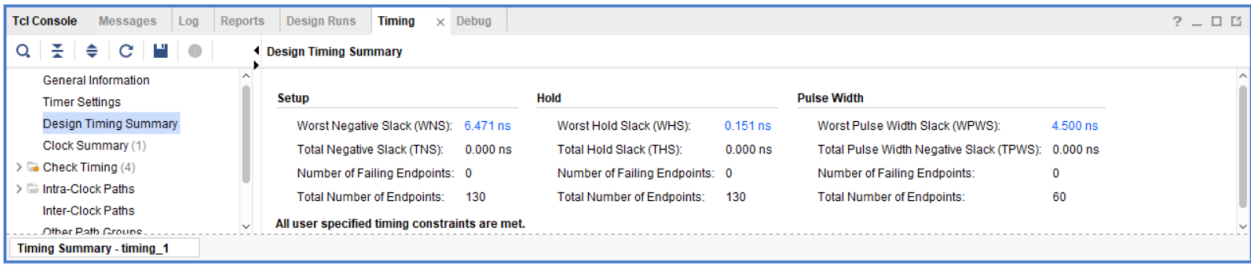


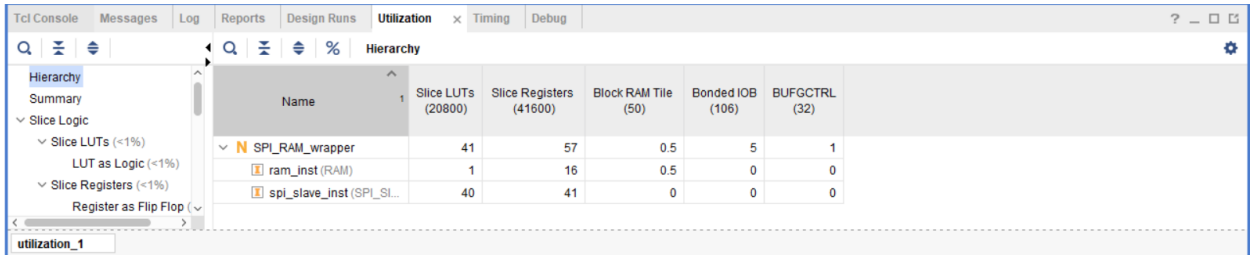

Project Summary
Schematic (2)
Schematic
43 Cells
64 Nets

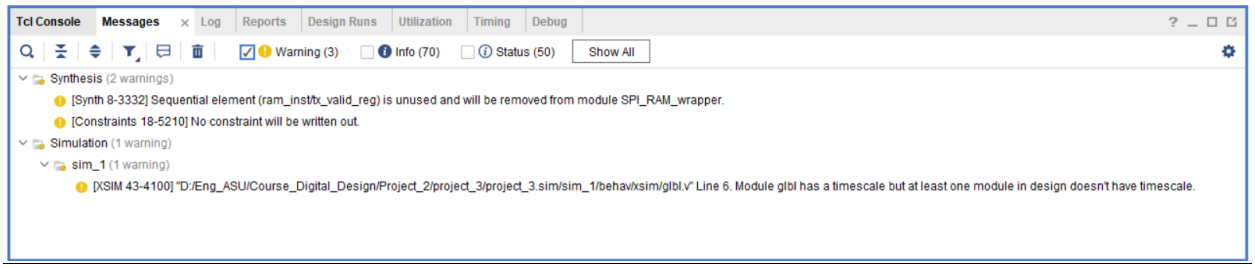

Tcl Console
Messages
Log
Reports
Design Runs
Warning (1)
Info (22)
Status (28)
Show All
Simulation (1 warning)
sim_1 (1 warning)
[XSIM 43-4100] "D:/Eng_ASU/Course_Digital_Design/Project_2/project_3/project_3.sim/sim_1/behav/xsim/glbl.v" Line 6. Module glbl has a timescale but at least one module in design doesn't have timescale.

Synthesis:

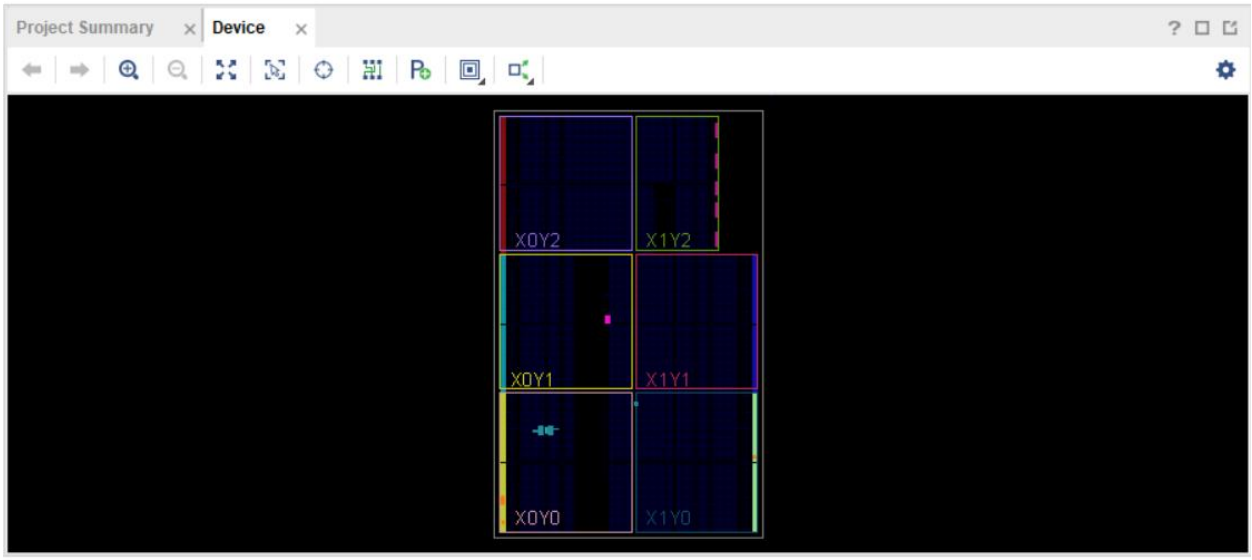

Schematic
8 Cells
5 I/O Ports
33 Nets
clk
rst_n
MOSI
SS_n
MISO
clk_IBUF_inst
clk_IBUF_BUFG_inst
rst_n_IBUF_inst
MOSI_IBUF_inst
SS_n_IBUF_inst
MISO_OBUF_inst
IBUF
BUFG
OBUF
ram_inst
RAM
spi_slave_inst
SPI_Slave

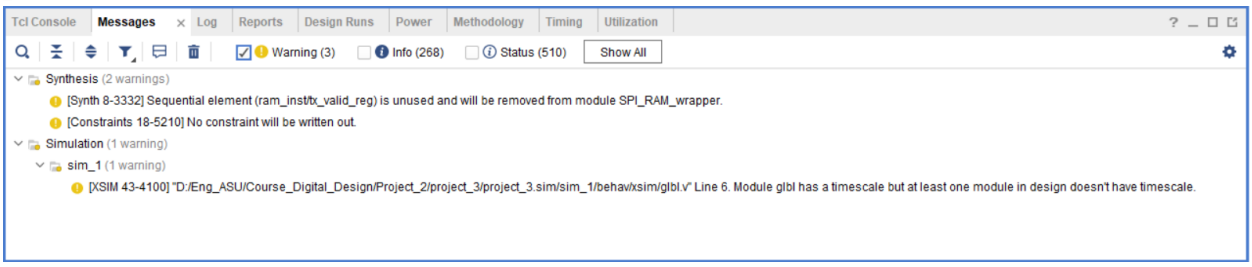






Implementation:





Tcl Console | Messages | Log | Reports | Design Runs | Power | Methodology | Timing

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (4)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.329 ns	Worst Hold Slack (WHS):	0.068 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:	130	Total Number of Endpoints:	130	Total Number of Endpoints:	60

All user specified timing constraints are met.

Timing Summary - impl_1 (saved)

Tcl Console | Messages | Log | Reports | Design Runs | Power | Methodology | Timing | Utilization

Hierarchy
Summary
Slice Logic
Slice LUTs (<1%)
LUT as Logic (<1%)
Slice Registers (<1%)
Register as Flip Flop (

Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_RAM_wrapper	41	57	17	41	27	0.5	5	1
ram_inst (RAM)	2	16	4	2	0	0.5	0	0
spi_slave_inst (SPI_SI...	39	41	15	39	25	0	0	0

utilization_1