TEAM: VIPER

Project 2 - SPI

Nouran Hamdy Mohamed
Abdelrahman Mostafa Shawky Mady
Nardeen Hishmat Ageeb

Verilog Code

RAM

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

    input clk, rst_n, rx_valid;
    input [ADDR_SIZE+1:0] din;

    output reg tx_valid;
    output reg [ADDR_SIZE-1:0] dout;
    wire [ADDR_SIZE-1:0] data;
    wire [1:0] signal;

    reg [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
    reg [ADDR_SIZE-1:0] wr_addr, rd_addr;

    assign signal = din[ADDR_SIZE+1:ADDR_SIZE];
    assign data = din[ADDR_SIZE-1:0];
```

```
     always @(posedge clk) begin
         if(~rst_n) begin
             tx_valid <= 0;
             dout <= 0;
             wr_addr <= 0;
             rd_addr <= 0;
         end
         else begin
             case(signal)
             2'b00: begin
                 if(rx_valid) wr_addr <= data;
                 tx_valid <= 0;
             end
             2'b01: begin
                 if(rx_valid) mem[wr_addr] <= data;
                 tx_valid <= 0;
             end
             2'b10: begin
                 if(rx_valid) rd_addr <= data;
                 tx_valid <= 0;
             end
             2'b11: begin
                 dout <= mem[rd_addr];
                 tx_valid <= 1;
             end
             endcase
         end
     end
endmodule
```

SPI Slave

```
module SPI_SLAVE(clk, rst_n, SS_n, MOSI, MISO, rx_data, rx_valid, tx_data, tx_valid);
    parameter ADDR_SIZE = 8;

    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, MOSI, tx_valid;
    input [ADDR_SIZE-1:0] tx_data;

    output reg MISO, rx_valid;
    output reg [ADDR_SIZE+1:0] rx_data;

    (* fsm_encoding = "gray" *)
    reg [2:0] cs, ns;
    reg data_addr; //0: addr, 1: data
    reg [4:0] bit_cntr_wr, bit_cntr_rd; //counting cycles - indicates end of state

    //State Memory
    always @(posedge clk or negedge rst_n) begin
        if(~rst_n) cs <= IDLE;
        else cs <= ns;
    end

```

```
    //Next State Logic
    always @(*) begin
        case(cs)
        IDLE: ns = (SS_n) ? IDLE : CHK_CMD;
        CHK_CMD: begin
            if(SS_n) ns = IDLE;
            else begin
                if(MOSI) begin
                    if(data_addr) ns = READ_DATA; //increments on going from ADD to DATA
                    else ns = READ_ADD;
                end
                else ns = WRITE;
            end
        end
        WRITE: ns = (SS_n) ? IDLE : WRITE;
        READ_ADD: ns = (SS_n) ? IDLE : READ_ADD;
        READ_DATA: ns = (SS_n) ? IDLE : READ_DATA;
        default: ns = IDLE;
        endcase
    end

```

```
    //Output Logic
    always @(posedge clk or negedge rst_n) begin
        if(~rst_n) begin
            MISO <= 0;
            rx_valid <= 0;
            rx_data <= 0;
            bit_cntr_wr <= 0;
            bit_cntr_rd <= 0;
            data_addr <= 0;
        end
        else begin
            case(cs)
            IDLE: begin
                MISO <= 0;
                rx_valid <= 0;
                rx_data <= 0;
                bit_cntr_wr <= 0;
                bit_cntr_rd <= 0;
            end
            WRITE: begin
                rx_data <= {rx_data[ADDR_SIZE:0], MOSI}; //Shift OP (Serial to Parallel)
                bit_cntr_wr <= bit_cntr_wr + 1;
                if(bit_cntr_wr == (ADDR_SIZE+1)) begin //next bit_cntr=10, rx_data is ready
                    rx_valid <= 1;
                    bit_cntr_wr <= 0;
                end
                else rx_valid <= 0; //rx_data is not ready yet
            end
            READ_ADD: begin
                rx_data <= {rx_data[ADDR_SIZE:0], MOSI}; //Shift OP (Serial to Parallel)
                bit_cntr_wr <= bit_cntr_wr + 1;
                if(bit_cntr_wr == (ADDR_SIZE+1)) begin //next bit_cntr=10, rx_data is ready
                    rx_valid <= 1;
                    bit_cntr_wr <= 0;
                end
                else rx_valid <= 0; //rx_data is not ready yet
                if(SS_n) data_addr <= 1; //SS_n high -> end of state -> READ_DATA next
            end
```

```
            READ_DATA: begin
                if(bit_cntr_wr < (ADDR_SIZE+2)) begin //bit_cntr<10, rx_data (dummy) is being transferred
                    rx_data <= {rx_data[ADDR_SIZE:0], MOSI}; //Shift OP (Serial to Parallel)
                    if(bit_cntr_wr == (ADDR_SIZE+1)) bit_cntr_wr <= 0; //next bit_cntr=10, rx_data is ready
                    bit_cntr_wr <= bit_cntr_wr + 1;
                end
                else begin
                    if(tx_valid) begin
                        MISO <= tx_data[ADDR_SIZE-1-bit_cntr_rd]; //Parallel to Serial
                        if(bit_cntr_rd == (ADDR_SIZE-2)) bit_cntr_rd <= 0; //bit_cntr_rd range=0:7
                        if(bit_cntr_rd > (ADDR_SIZE-1)) MISO <= 0; //next bit_cntr=8, tx_data is ready
                        bit_cntr_rd <= bit_cntr_rd + 1;
                    end
                    else begin
                        MISO <= 0; //MISO=0 as long as tx_valid is low
                        bit_cntr_rd <= 0;
                    end
                end
                if(SS_n) data_addr <= 0; //SS_n high -> end of state -> READ_ADD next
            end
            default: begin
                MISO <= 0;
                rx_valid <= 0;
                rx_data <= 0;
                bit_cntr_wr <= 0;
                bit_cntr_rd <= 0;
                data_addr <= 0;
            end
            endcase
        end
    end
endmodule
```

MASTER SPI – Top Module

```
module MASTER_SPI(clk, rst_n, SS_n, MOSI, MISO);
    parameter ADDR_SIZE = 8;
    input clk, rst_n, SS_n, MOSI;
    output MISO;

    wire rx_valid, tx_valid;
    wire [ADDR_SIZE+1:0] rx_data;
    wire [ADDR_SIZE-1:0] tx_data;

    RAM #(.ADDR_SIZE(ADDR_SIZE)) RAM1(clk, rst_n, rx_valid, rx_data, tx_valid, tx_data);
    SPI_SLAVE #(.ADDR_SIZE(ADDR_SIZE)) SPI1(clk, rst_n, SS_n, MOSI, MISO, rx_data, rx_valid, tx_data, tx_valid);
endmodule
```

Test-Bench Code

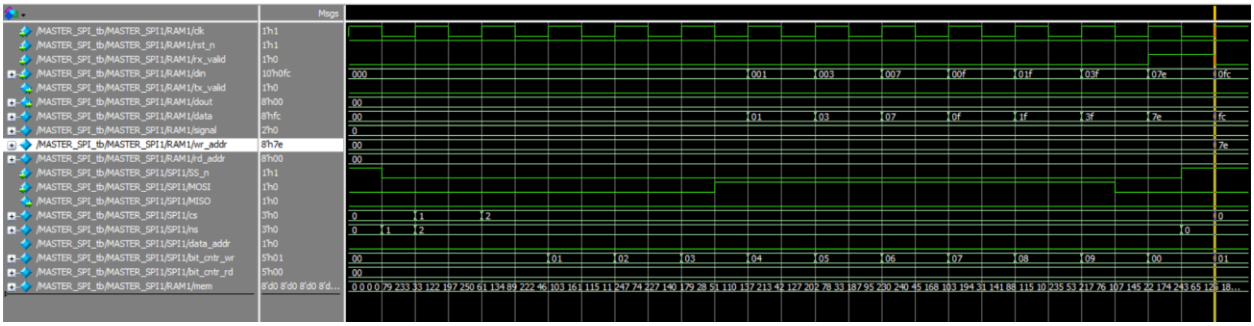
```
module MASTER_SPI_tb();
    parameter ADDR_SIZE = 8;
    reg clk, rst_n, SS_n, MOSI;
    wire MISO;

    MASTER_SPI #(.ADDR_SIZE(ADDR_SIZE)) MASTER_SPI1(clk, rst_n, SS_n, MOSI, MISO);

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

    initial begin
        $readmemh("RAM.dat.txt", MASTER_SPI1.RAM1.mem);
        rst_n = 0; SS_n = 1; MOSI = 0;
        repeat (20) @(negedge clk);

        rst_n = 1;
        @(negedge clk);
        repeat(20) begin
            SS_n = 0;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            repeat(ADDR_SIZE) begin
                MOSI = $random;
                @(negedge clk);
            end
            SS_n = 1;
            @(negedge clk);
            ////////////////////////
            ////////////////////////
```

```
            SS_n = 0;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            repeat(ADDR_SIZE) begin
                MOSI = $random;
                @(negedge clk);
            end
            SS_n = 1;
            @(negedge clk);
            ////////////////////
            ////////////////////
            SS_n = 0;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            MOSI = 0;
            @(negedge clk);
            repeat(ADDR_SIZE) begin
                MOSI = $random;
                @(negedge clk);
            end
            SS_n = 1;
            @(negedge clk);
            ////////////////////
            ////////////////////
```

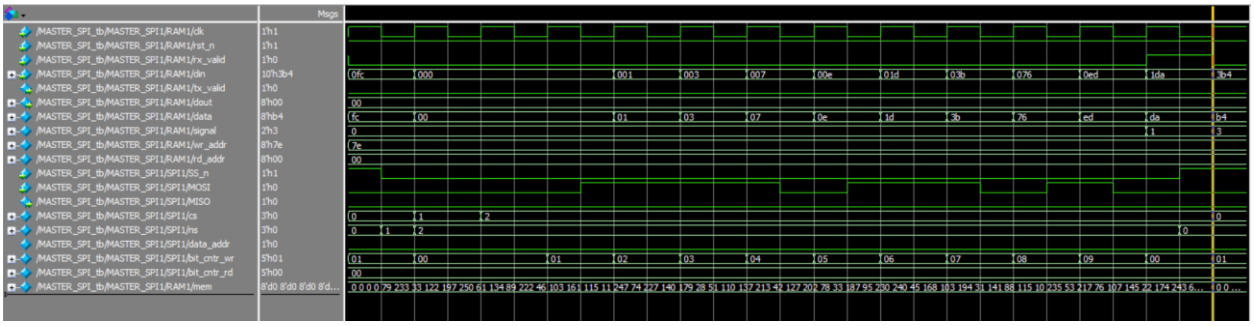
```
            SS_n = 0;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            MOSI = 1;
            @(negedge clk);
            repeat(ADDR_SIZE) begin
                MOSI = $random;
                @(negedge clk);
            end
            MOSI = 0;
            repeat(ADDR_SIZE+1) @(negedge clk);
            SS_n = 1;
            @(negedge clk);
            ////////////////////
            ////////////////////
        end
        $stop;
    end
endmodule
```

Waveform Snippets – QuestaSim

Write – Address (MOSI: 0 → 00)

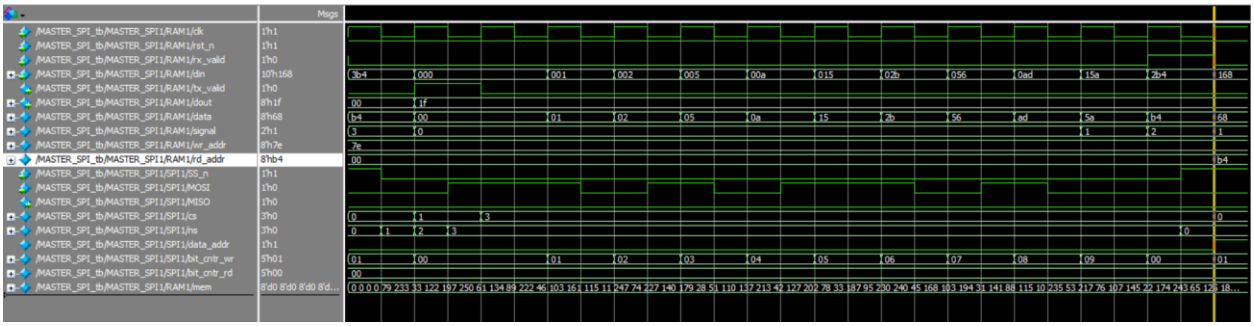


Write – Data (MOSI: 0 → 01)

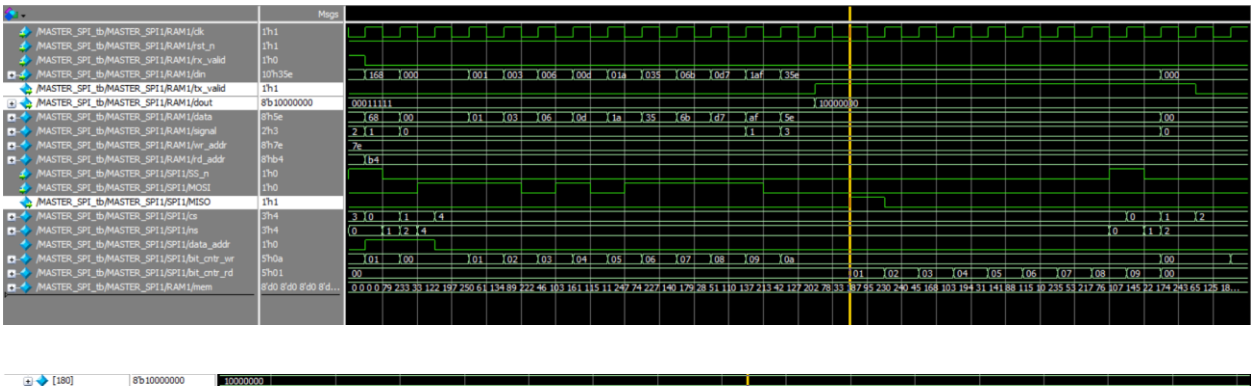


RAM Update:

Read – Address (MOSI: 1 → 10)

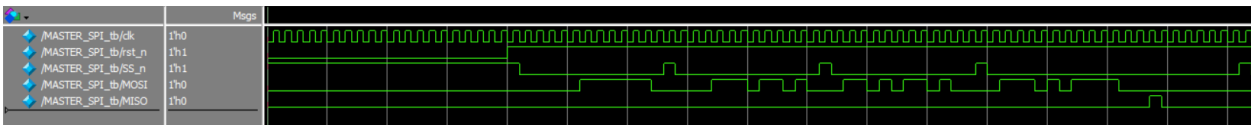


Read – Data (MOSI: 1 → 11)

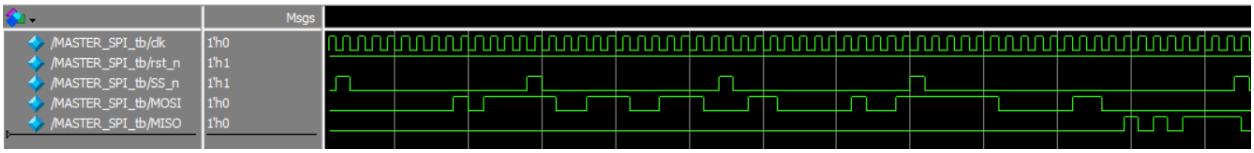


Snippets of Different Iterations

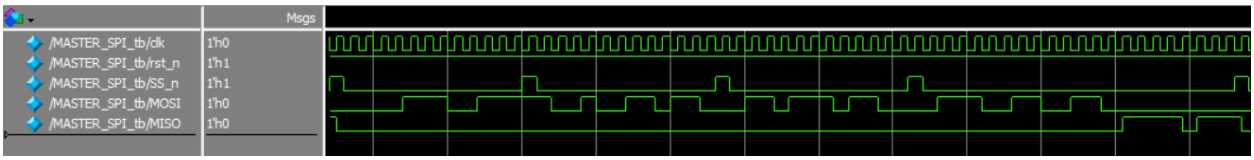
Snippet I



Snippet II



Snippet III



Picking Best Encoding

After-Implementation Timing Reports – Debug Included

Gray:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.285 ns	Worst Hold Slack (WHS):	0.049 ns	Worst Pulse Width Slack (WPWS):	3.750 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:	3707	Total Number of Endpoints:	3691	Total Number of Endpoints:	2026

All user specified timing constraints are met.

One_hot:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.280 ns	Worst Hold Slack (WHS):	0.060 ns	Worst Pulse Width Slack (WPWS):	3.750 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:	3709	Total Number of Endpoints:	3693	Total Number of Endpoints:	2028

All user specified timing constraints are met.

Seq:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.280 ns	Worst Hold Slack (WHS):	0.060 ns	Worst Pulse Width Slack (WPWS):	3.750 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:	3709	Total Number of Endpoints:	3693	Total Number of Endpoints:	2028

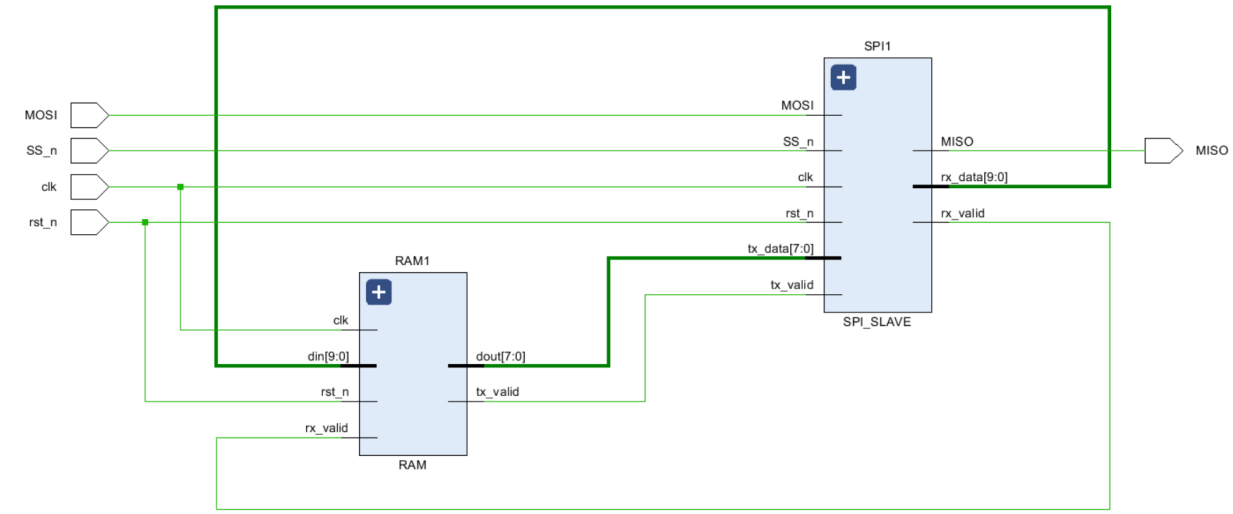
All user specified timing constraints are met.

Gray provides the best (Highest) “Setup – Worst Negative Slack”.

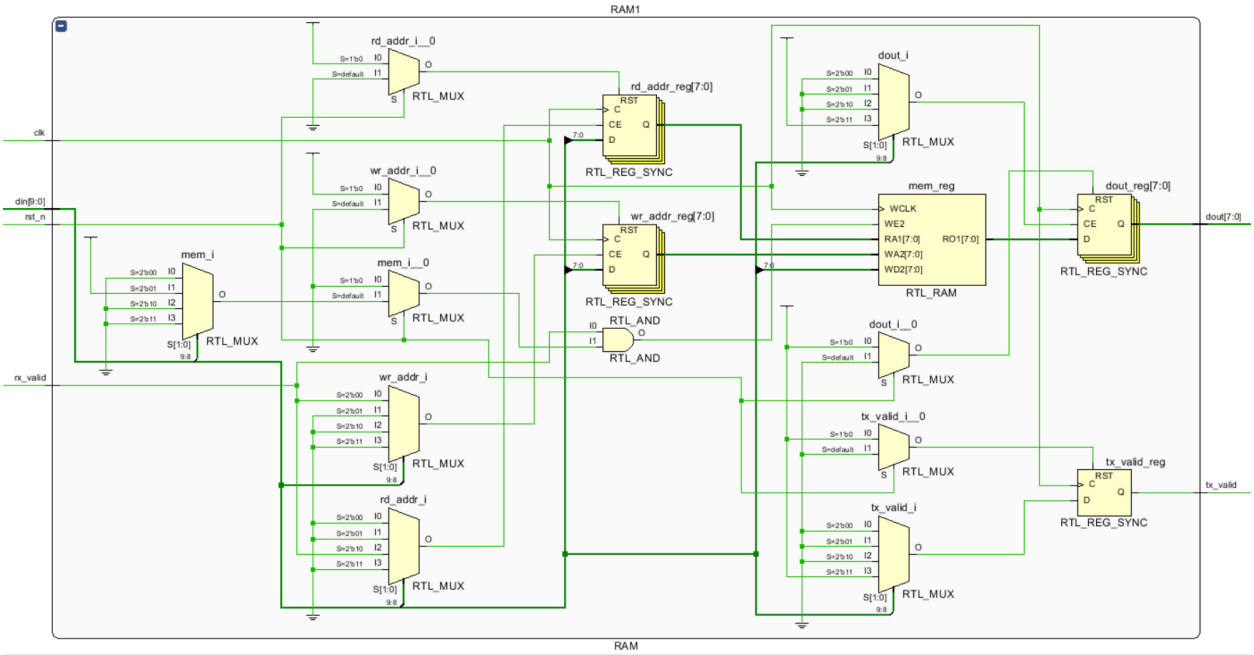
Chosen FSM Encoding: *Gray*

Schematic After Elaboration

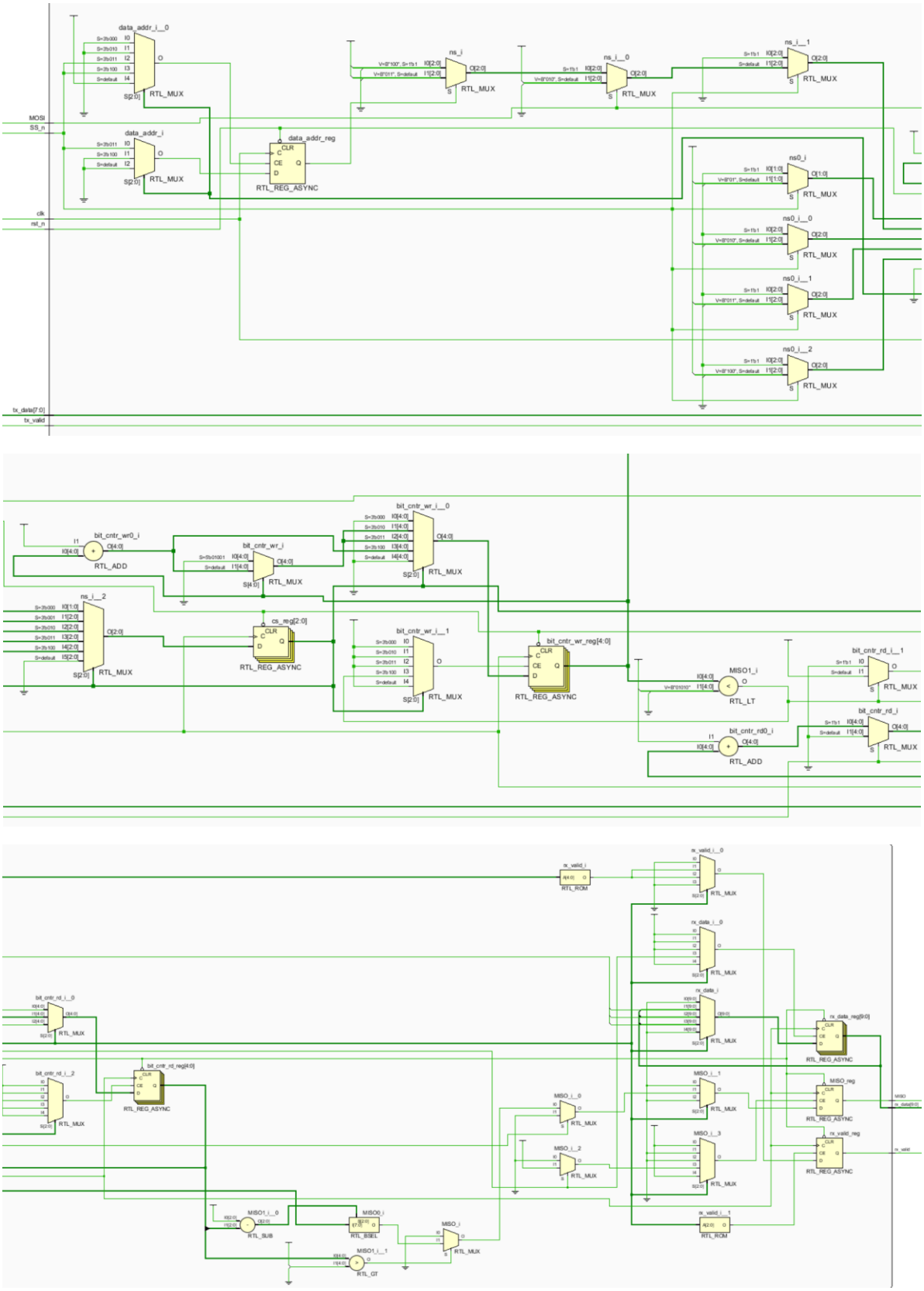
Top Module



RAM

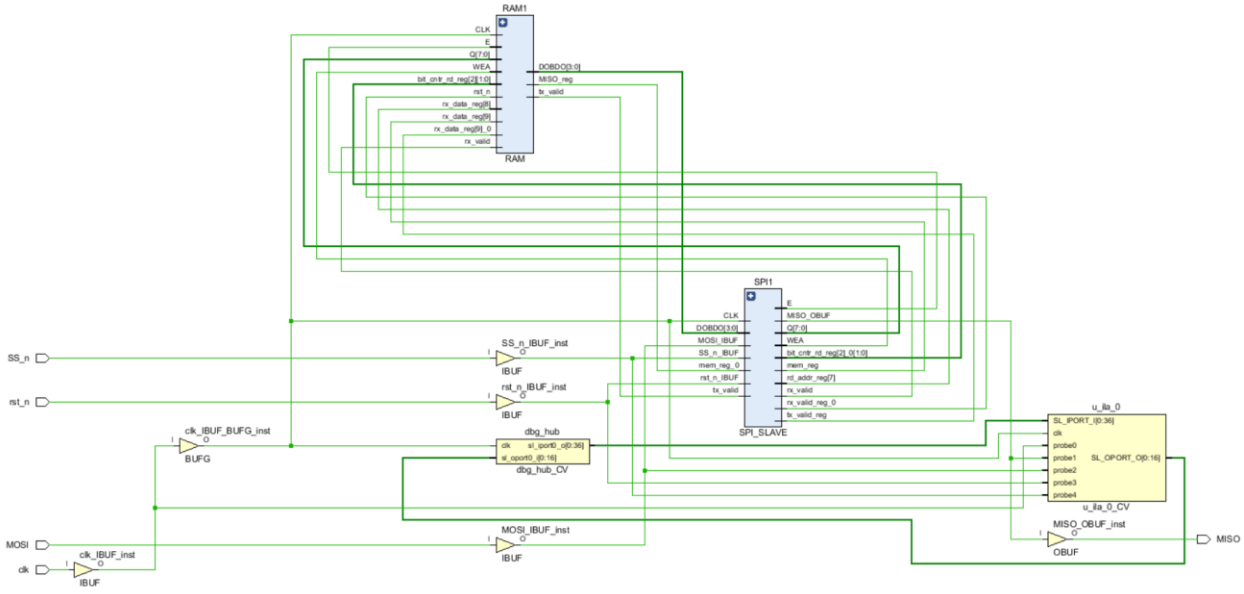


SPI Slave

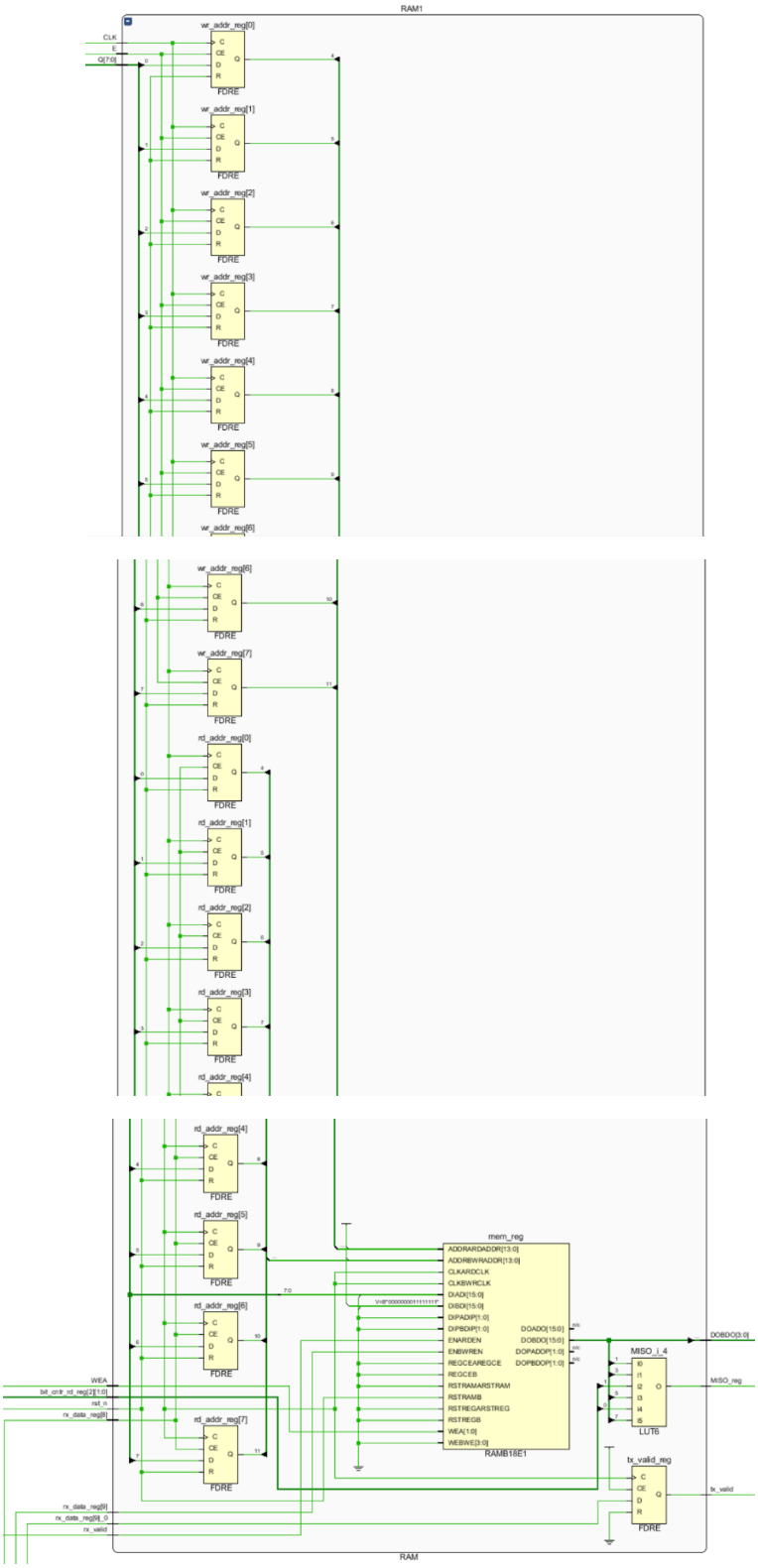

data_addr_i__0
RTL_MUX
ns_i
ns_i__0
ns_i__1
MOSI
SS_n
data_addr_i
data_addr_reg
RTL_REG_ASYNC
clk
rst_n
ns0_i
ns0_i__0
ns0_i__1
ns0_i__2
tx_data[7:0]
tx_valid
bit_cntr_wr0_i
RTL_ADD
bit_cntr_wr_i
bit_cntr_wr_i__0
bit_cntr_wr_i__1
ns_i__2
cs_reg[2:0]
bit_cntr_wr_reg[4:0]
MISO1_i
RTL_LT
bit_cntr_rd_i__1
bit_cntr_rd_i
bit_cntr_rd0_i
rx_valid_i
RTL_ROM
rx_valid_i__0
rx_data_i__0
rx_data_i
rx_data_reg[9:0]
bit_cntr_rd_i__0
bit_cntr_rd_i__2
bit_cntr_rd_reg[4:0]
MISO_i__0
MISO_i__1
MISO_i__2
MISO_i__3
MISO_reg
rx_valid_reg
MISO1_i__0
RTL_SUB
MISO0_i
RTL_BSEL
MISO_i
MISO1_i__1
RTL_GT
rx_valid_i__1

Schematic After Synthesis

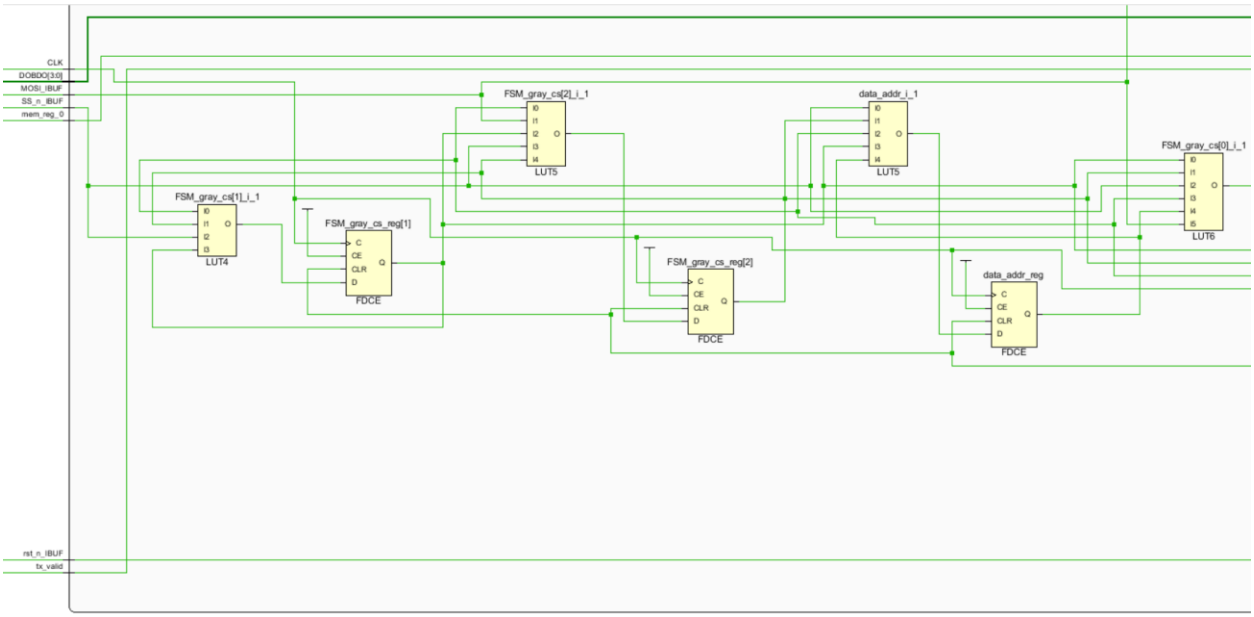
Top Module

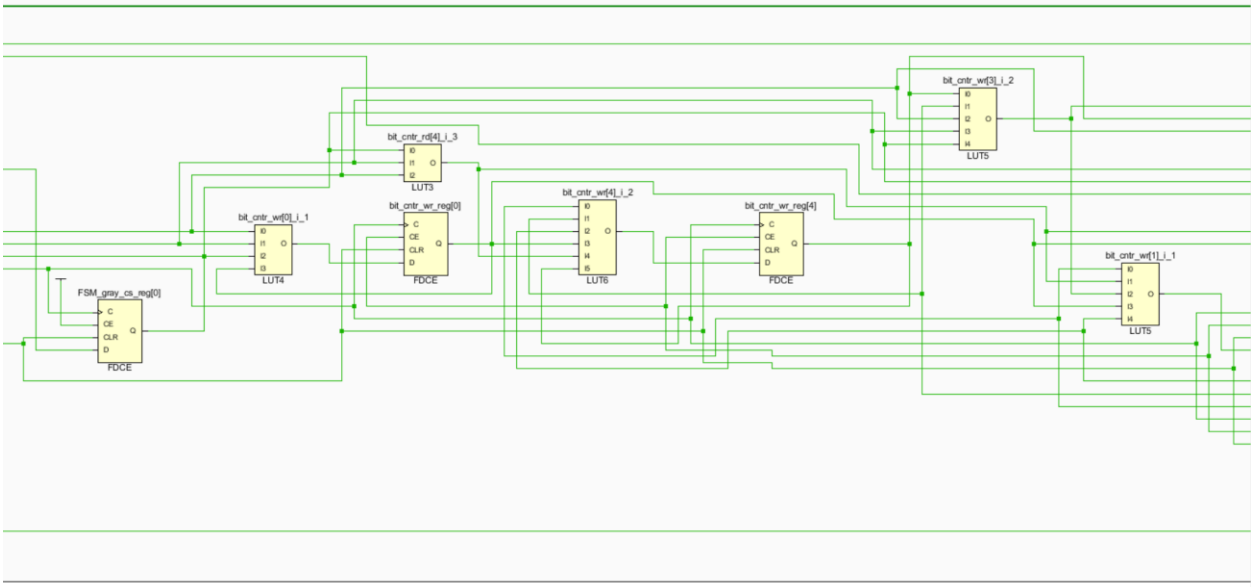


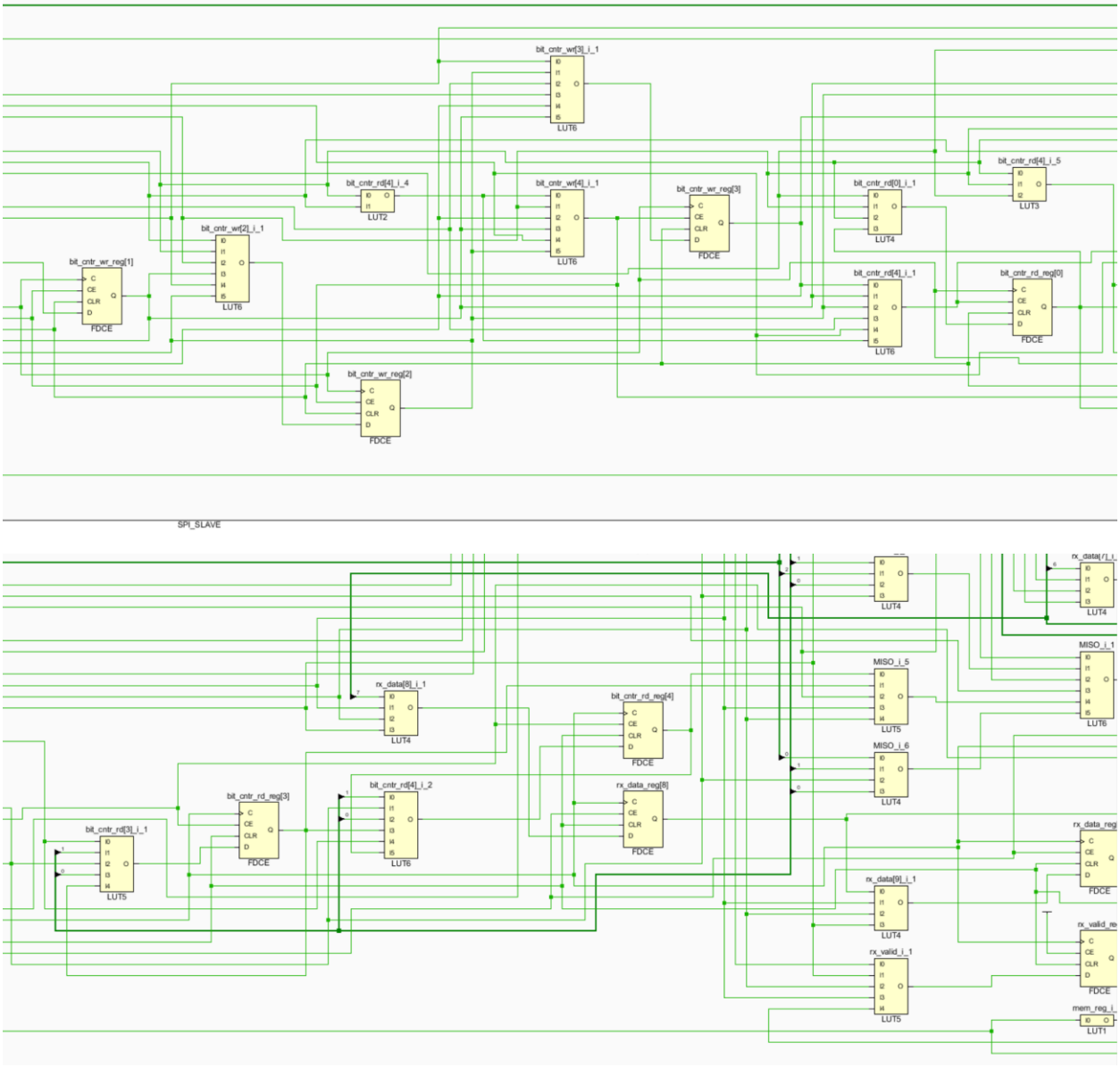
RAM


RAM1
wr_addr_reg[0]
wr_addr_reg[1]
wr_addr_reg[2]
wr_addr_reg[3]
wr_addr_reg[4]
wr_addr_reg[5]
wr_addr_reg[6]
wr_addr_reg[7]
rd_addr_reg[0]
rd_addr_reg[1]
rd_addr_reg[2]
rd_addr_reg[3]
rd_addr_reg[4]
rd_addr_reg[5]
rd_addr_reg[6]
rd_addr_reg[7]
FDRE
mem_reg
RAMB18E1
MISO_i_4
LUT6
tx_valid_reg
WEA
rst_n
RAM

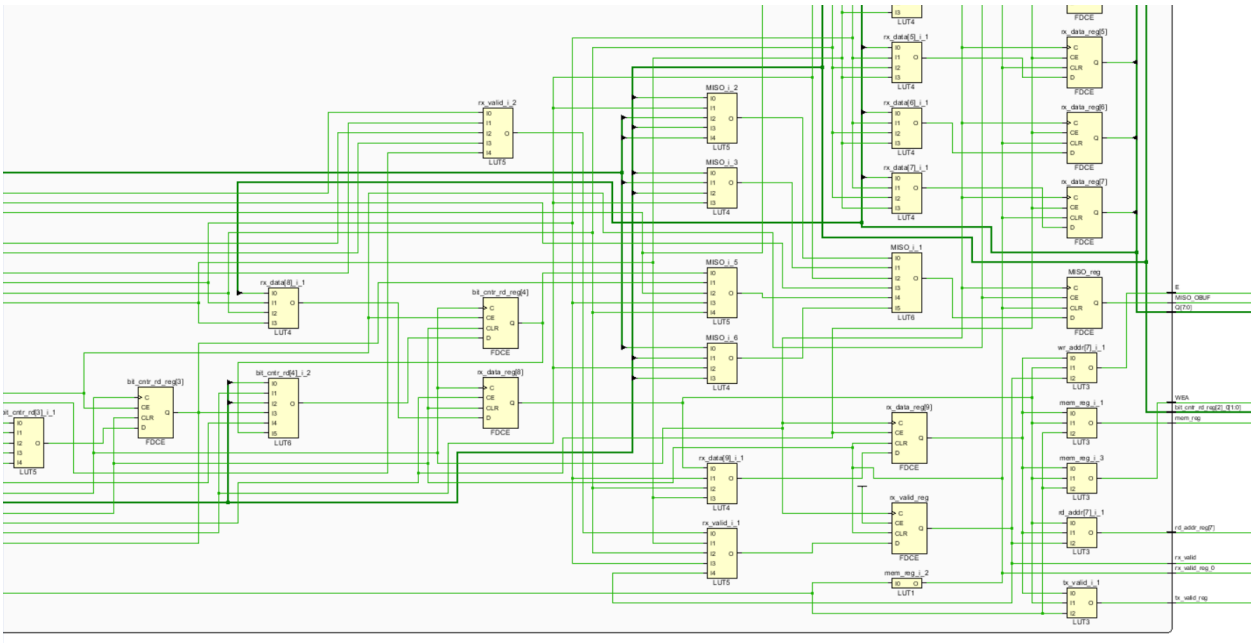
SPI Slave


CLK
DOBDO[3:0]
MOSI_IBUF
SS_n_IBUF
mem_reg_0
rst_n_IBUF
tx_valid
FSM_gray_cs[2]_i_1
LUT5
data_addr_i_1
LUT5
FSM_gray_cs[0]_i_1
LUT6
FSM_gray_cs[1]_i_1
LUT4
FSM_gray_cs_reg[1]
FDCE
FSM_gray_cs_reg[2]
FDCE
data_addr_reg
FDCE


bit_cntr_wr[3]_i_2
LUT5
bit_cntr_rd[4]_i_3
LUT3
bit_cntr_wr[4]_i_2
LUT6
bit_cntr_wr_reg[4]
FDCE
bit_cntr_wr[0]_i_1
LUT4
bit_cntr_wr_reg[0]
FDCE
FSM_gray_cs_reg[0]
FDCE
bit_cntr_wr[1]_i_1
LUT5


bit_cntr_wr[3]_i_1
LUT6
bit_cntr_rd[4]_i_4
LUT2
bit_cntr_wr[4]_i_1
LUT6
bit_cntr_wr_reg[3]
FDCE
bit_cntr_rd[0]_i_1
LUT4
bit_cntr_rd[4]_i_5
LUT3
bit_cntr_wr[2]_i_1
LUT6
bit_cntr_wr_reg[1]
FDCE
bit_cntr_rd[4]_i_1
LUT6
bit_cntr_rd_reg[0]
FDCE
bit_cntr_wr_reg[2]
FDCE
SPI_SLAVE

LUT4
rx_data[8]_i_1
LUT4
bit_cntr_rd_reg[4]
FDCE
MISO_i_5
LUT5
MISO_i_1
LUT6
MISO_i_6
LUT4
bit_cntr_rd_reg[3]
FDCE
bit_cntr_rd[4]_i_2
LUT6
rx_data_reg[8]
FDCE
bit_cntr_rd[3]_i_1
LUT5
rx_data_reg
FDCE
rx_data[9]_i_1
LUT4
rx_valid_i_1
LUT5
rx_valid_re
FDCE
mem_reg_i_
LUT1



Synthesis Report

FSM Encoding – Detected

```
---------------------------------------------------------------------------------
INFO: [Synth 8-6157] synthesizing module 'MASTER_SPI' [D:/Xilinx_Vivado/SPI/MASTER_SPI.v:1]
        Parameter ADDR_SIZE bound to: 8 - type: integer
INFO: [Synth 8-6157] synthesizing module 'RAM' [D:/Xilinx_Vivado/SPI/RAM.v:1]
        Parameter MEM_DEPTH bound to: 256 - type: integer
        Parameter ADDR_SIZE bound to: 8 - type: integer
INFO: [Synth 8-6155] done synthesizing module 'RAM' (1#1) [D:/Xilinx_Vivado/SPI/RAM.v:1]
INFO: [Synth 8-6157] synthesizing module 'SPI_SLAVE' [D:/Xilinx_Vivado/SPI/SPI_SLAVE.v:1]
        Parameter ADDR_SIZE bound to: 8 - type: integer
        Parameter IDLE bound to: 3'b000
        Parameter CHK_CMD bound to: 3'b001
        Parameter WRITE bound to: 3'b010
        Parameter READ_ADD bound to: 3'b011
        Parameter READ_DATA bound to: 3'b100
INFO: [Synth 8-5534] Detected attribute (* fsm_encoding = "gray" *) [D:/Xilinx_Vivado/SPI/SPI_SLAVE.v:17]
INFO: [Synth 8-6155] done synthesizing module 'SPI_SLAVE' (2#1) [D:/Xilinx_Vivado/SPI/SPI_SLAVE.v:1]
INFO: [Synth 8-6155] done synthesizing module 'MASTER_SPI' (3#1) [D:/Xilinx_Vivado/SPI/MASTER_SPI.v:1]
---------------------------------------------------------------------------------
```

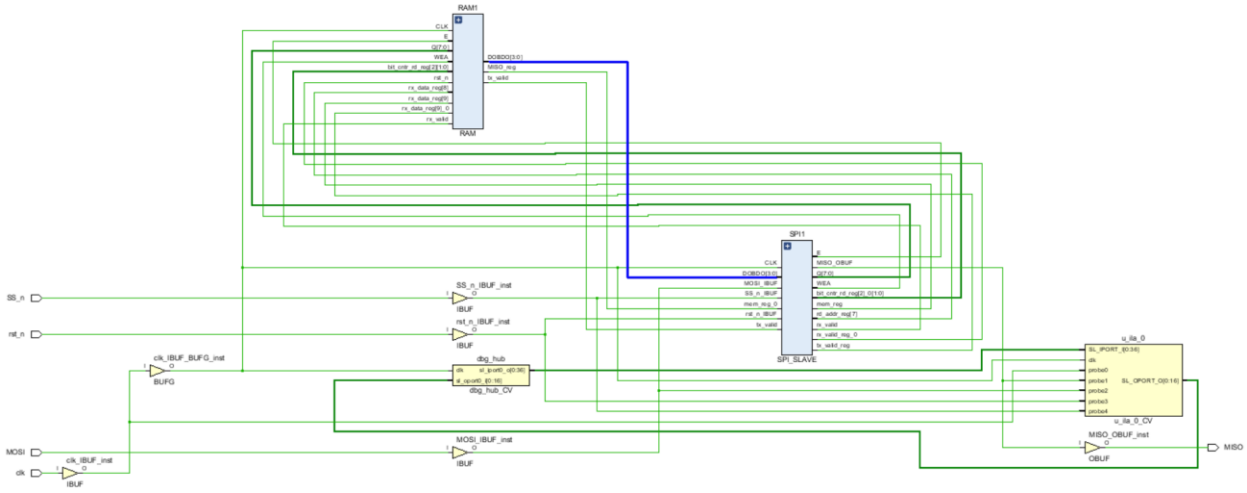
Timing Report

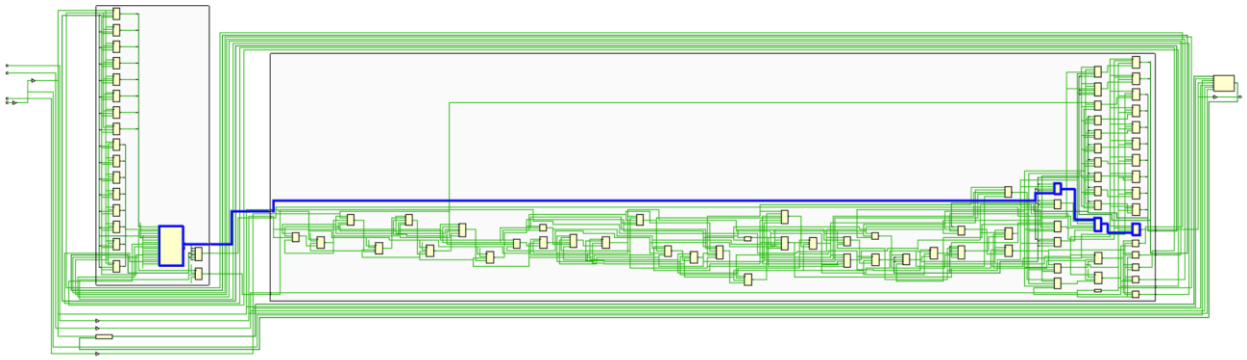
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.236 ns	Worst Hold Slack (WHS):	0.139 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:	108	Total Number of Endpoints:	108	Total Number of Endpoints:	46

All user specified timing constraints are met.

Critical Path



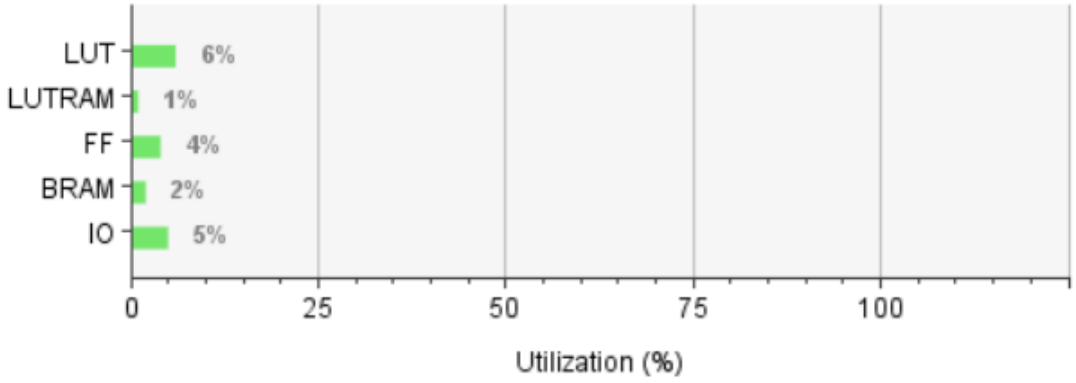


Device After Implementation

Utilization Report

Summary

Resource	Utilization	Available	Utilization %
LUT	1198	20800	5.76
LUTRAM	98	9600	1.02
FF	1858	41600	4.47
BRAM	1	50	2.00
IO	5	106	4.72



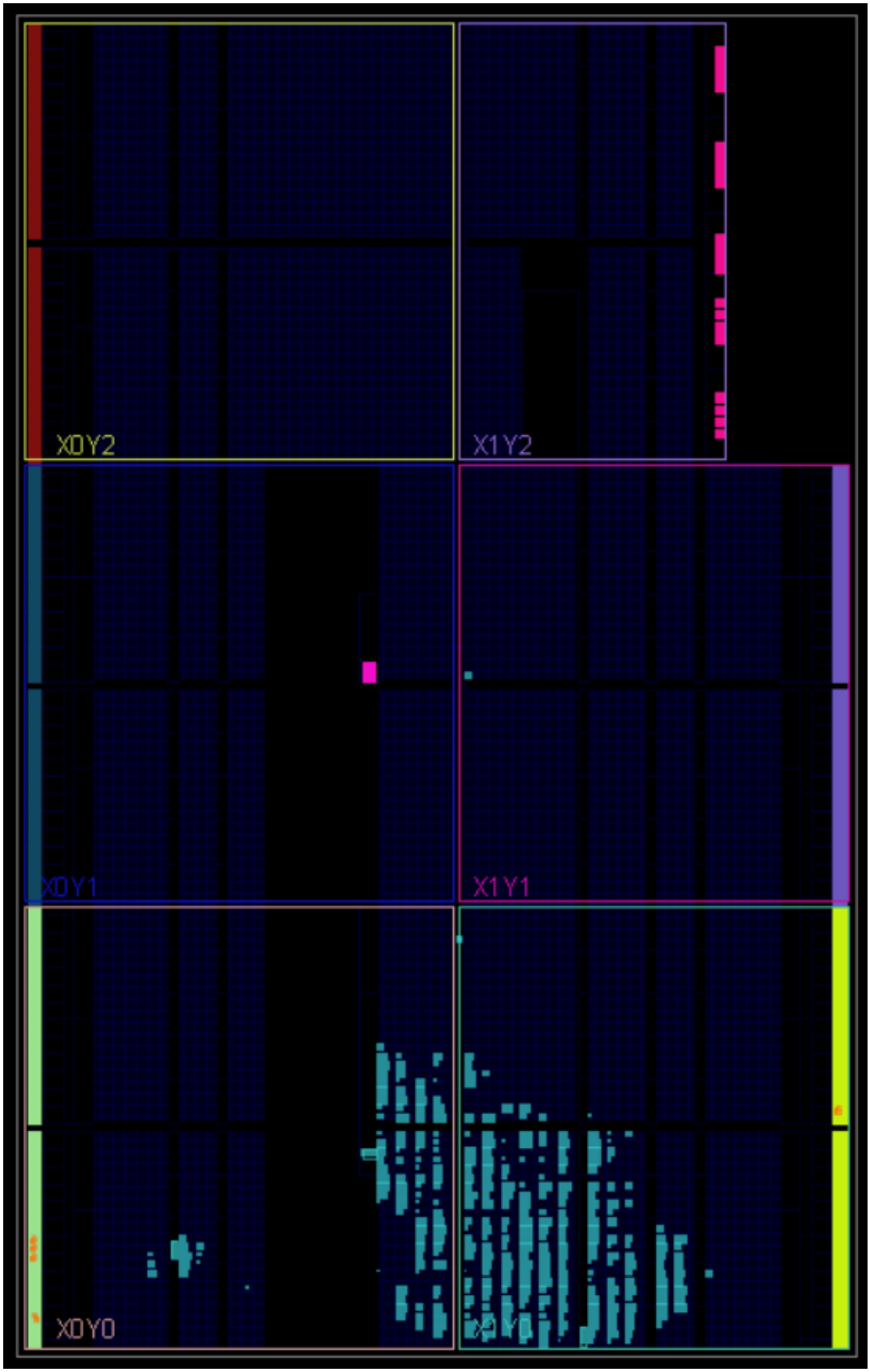
Timing Report

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.285 ns	Worst Hold Slack (WHS):	0.049 ns	Worst Pulse Width Slack (WPWS):	3.750 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:	3707	Total Number of Endpoints:	3691	Total Number of Endpoints:	2026

All user specified timing constraints are met.

FPGA Device



Messages Tab