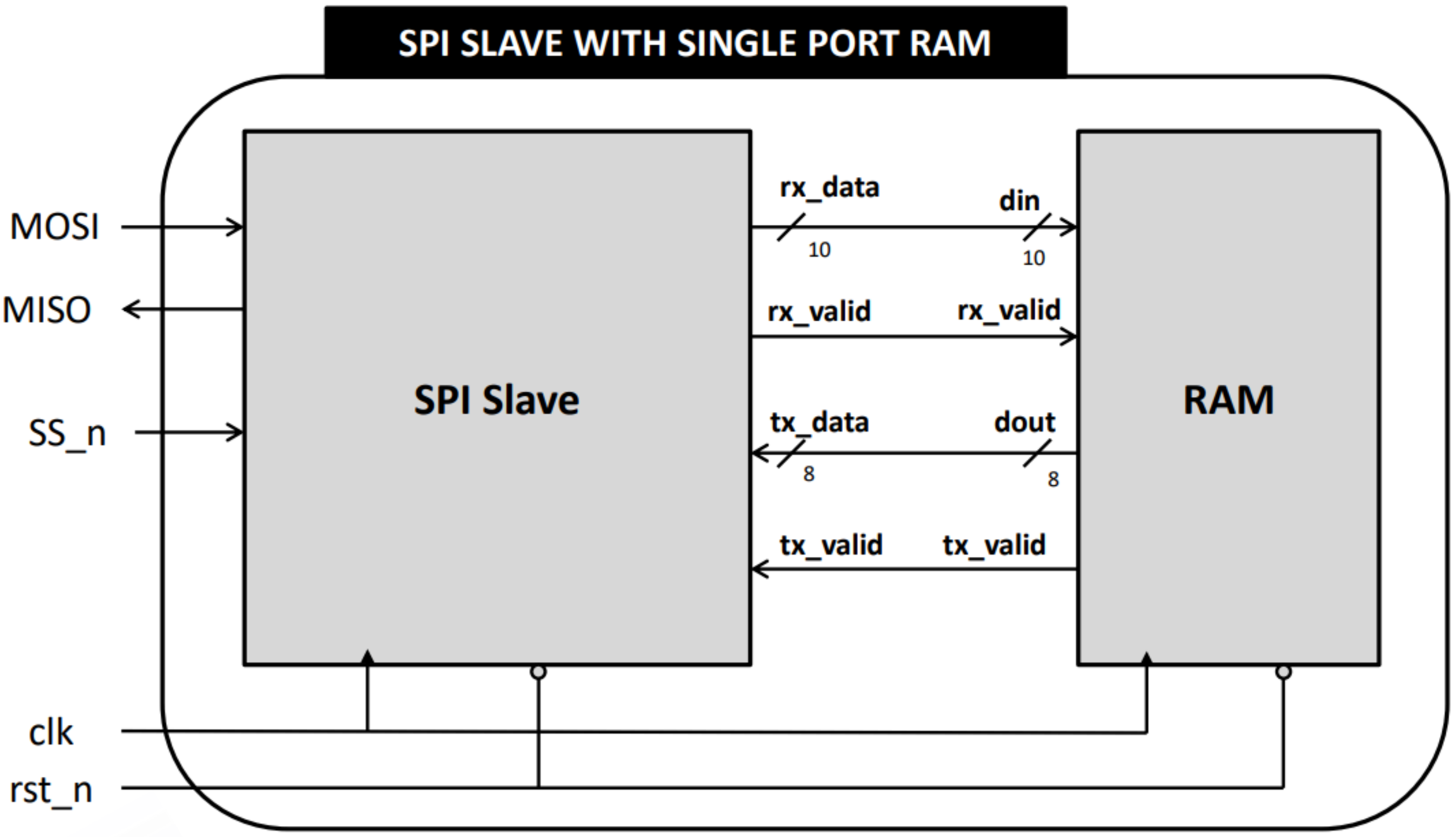
SPI REPORT



Muhammad Wael El-Sayed

Code of Design

```
module SPRAM #(parameter MEM_DEPTH = 256)(
    input clk, rst_n,
    input [9:0] D_in, // 10 Bits 2:8 , 2 -> for Control , 8 -> address Bus
    input rx_valid,
    output reg [7:0] D_out,
    output reg tx_valid
);
    reg [7:0] SP_RAM [MEM_DEPTH-1:0];

    reg [7:0] W_address;    // to Hold write address
    reg [7:0] R_address;    // to Hold read address

    always @(posedge clk) begin
        if (~rst_n) begin
            W_address <= 8'd0;
            R_address <= 8'd0;
            tx_valid <= 1'b0;
            D_out <= 8'd0;
        end
        else begin
            if (D_in[9:8] == 2'b00 && rx_valid) begin
                W_address <= D_in[7:0];
                tx_valid <= 1'b0;
            end
            else if (D_in[9:8] == 2'b01 && rx_valid) begin
                SP_RAM[W_address] <= D_in[7:0];
                tx_valid <= 1'b0;
            end
            else if (D_in[9:8] == 2'b10 && rx_valid) begin
                R_address <= D_in[7:0];
                tx_valid <= 1'b0;
            end
            else if (D_in[9:8] == 2'b11) begin
                D_out <= SP_RAM[R_address];
                tx_valid <= 1'b1;
            end
        end
    end

endmodule
```

Code of Design

```
module SPI_Slave #(
    parameter IDLE = 0,
    parameter CHK_CMD = 1,
    parameter WRITE = 2,
    parameter READ_ADD = 3,
    parameter READ_DATA = 4,
    localparam CNT_SAT = 4'd10
)(
    input clk, rst_n,
    input SS_n,
    input [7:0] tx_data,
    input tx_valid,
    input MOSI,
    output reg MISO,
    output reg [9:0] rx_data,
    output reg rx_valid
);
    (* fsm_encoding="sequential" *) reg [2:0] cs, ns;
    reg ADD_SAVED;
    reg [3:0] counter_1; // 0 -> 10
    reg [2:0] counter_2; // 0 -> 7

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

    // next state block
    always @(*) begin
        case (cs)
            IDLE: begin
                if (SS_n == 1'b0)
                    ns = CHK_CMD;
                else
                    ns = IDLE;
            end
            CHK_CMD: begin
                if (SS_n == 1'b0) begin
                    if (MOSI == 1'b1 && ADD_SAVED == 1'b0)
                        ns = READ_ADD;
                    else if (MOSI == 1'b1 && ADD_SAVED == 1'b1)
                        ns = READ_DATA;
                    else
                        ns = WRITE;
                end
                else
                    ns = IDLE;
            end
            WRITE: begin
                if (SS_n == 1'b1)
                    ns = IDLE;
                else
                    ns = WRITE;
            end
            READ_DATA: begin
                if (SS_n == 1'b1)
                    ns = IDLE;
                else
                    ns = READ_DATA;
            end
            READ_ADD: begin
                if (SS_n == 1'b1)
                    ns = IDLE;
                else
                    ns = READ_ADD;
            end
            default: ns = IDLE;
        endcase
    end

    // output block
    always @(posedge clk) begin
        if (~rst_n || counter_1 == CNT_SAT || cs == CHK_CMD || cs == IDLE)
            counter_1 <= 4'd0;
        else
            counter_1 <= counter_1 + 1'b1;
    end

    always @(posedge clk) begin
        if (~rst_n || cs != READ_DATA)
            counter_2 <= 3'd0;
        else if (tx_valid)
            counter_2 <= counter_2 + 1'b1;
    end

    always @(*) begin
        if (cs == READ_DATA && SS_n == 1'b0) begin
            case (counter_2)
                3'd0: MISO = tx_data[7];
                3'd1: MISO = tx_data[6];
                3'd2: MISO = tx_data[5];
                3'd3: MISO = tx_data[4];
                3'd4: MISO = tx_data[3];
                3'd5: MISO = tx_data[2];
                3'd6: MISO = tx_data[1];
                3'd7: MISO = tx_data[0];
            endcase
        end
        else
            MISO = 1'b0; // default value
    end

    // FF flag
    always @(posedge clk) begin
        if (~rst_n || cs == READ_DATA)
            ADD_SAVED <= 1'b0;
        else if (cs == READ_ADD)
            ADD_SAVED <= 1'b1;
    end

    always @(posedge clk) begin
        if (~rst_n || cs == CHK_CMD || cs == IDLE)
            rx_data <= 10'd0;
        else if ((cs == WRITE || cs == READ_DATA || cs == READ_ADD) && SS_n == 1'b0)
            rx_data <= {rx_data[8:0], MOSI};
    end

    always @(*) begin
        if (counter_1 == CNT_SAT && cs != IDLE && cs != CHK_CMD)
            rx_valid = 1'b1;
        else
            rx_valid = 1'b0;
    end

endmodule


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

    SPI_Slave SS1 (.clk(clk), .rst_n(rst_n), .SS_n(SS_n), .MISO(MISO), .MOSI(MOSI), .tx_data(tx_data), .tx_valid(tx_valid), .rx_data(rx_data), .rx_valid(rx_valid));

    SPRAM SRAM (.clk(clk), .rst_n(rst_n), .D_in(rx_data), .rx_valid(rx_valid), .D_out(tx_data), .tx_valid(tx_valid));

endmodule
```

Code of Test Bench

```
module SPI_Wrapper_tb();
    localparam TESTS = 20;
    localparam MEM_DEPTH = 256;
    integer i;

    reg clk, rst_n;
    reg SS_n;
    reg MOSI_tb;
    wire MISO_tb;

    reg [7:0] address_tb;       // Address wrote in W_address reg
    reg [7:0] dataWrite_tb;     // Data wrote into memory
    reg [7:0] dataRead_tb;      // Data Read from memory

    // DUT Instantiation
    SPI_Wrapper dut (.clk(clk), .rst_n(rst_n), .SS_n(SS_n), .MOSI(MOSI_tb), .MISO(MISO_tb));

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

    initial begin
        for (i = 0; i < MEM_DEPTH; i = i + 1)
            dut.SRAM.SP_RAM[i] = 0;
    end

    initial begin
        i = 0;
        address_tb = 0;
        dataWrite_tb = 0;
        dataRead_tb = 0;
        SS_n = 1;
        MOSI_tb = 0;
        rst_n = 0;
        @(negedge clk);
        rst_n = 1;

        repeat (TESTS) begin
            // Write Mode (Address)
            SS_n = 0;
            MOSI_tb = 1'b0;
            repeat(2) @(negedge clk);

            MOSI_tb = 1'b0;
            @(negedge clk);
            MOSI_tb = 1'b0;
            @(negedge clk);

            repeat(8) begin
                MOSI_tb = $random;
                address_tb = {address_tb[6:0], MOSI_tb};
                @(negedge clk);
            end

            SS_n = 1;
            @(negedge clk);

            // Write Mode (Data)
            SS_n = 0;
            @(negedge clk);
            MOSI_tb = 1'b0;
            @(negedge clk);

            MOSI_tb = 1'b0;
            @(negedge clk);
            MOSI_tb = 1'b1;
            @(negedge clk);

            repeat(8) begin
                MOSI_tb = $random;
                dataWrite_tb = {dataWrite_tb[6:0], MOSI_tb};
                @(negedge clk);
            end

            SS_n = 1;
            @(negedge clk);
```

Code of Test Bench

```
            // Read Mode (Adress)
            SS_n = 0;
            @(negedge clk);
            MOSI_tb = 1'b1;
            @(negedge clk);

            MOSI_tb = 1'b1;
            @(negedge clk);
            MOSI_tb = 1'b0;
            @(negedge clk);

            for (i = 7; i >= 0; i = i - 1) begin
                MOSI_tb = address_tb[i];
                @(negedge clk);
            end

            SS_n = 1;
            @(negedge clk);

            // Read Mode (Data)
            SS_n = 0;
            @(negedge clk);
            MOSI_tb = 1'b1;
            @(negedge clk);

            MOSI_tb = 1'b1;
            @(negedge clk);
            MOSI_tb = 1'b1;
            @(negedge clk);

            repeat(8) begin // wait for data to propagate through MISO
                MOSI_tb = $random; // Dummy bits
                @(negedge clk);
            end

            repeat(8) begin // wait for data to propagate through MISO
                dataRead_tb = {dataRead_tb[6:0], MISO_tb};
                @(negedge clk);
            end

            dataRead_tb = {dataRead_tb[6:0], MISO_tb};
            SS_n = 1;
            @(negedge clk);

            if (dataRead_tb !== dataWrite_tb) begin
                $display("[!] Error in SPI function\n");
                $stop;
            end
        end
        $display("[OK] Everything is good\n");
        $stop;
    end

endmodule
```

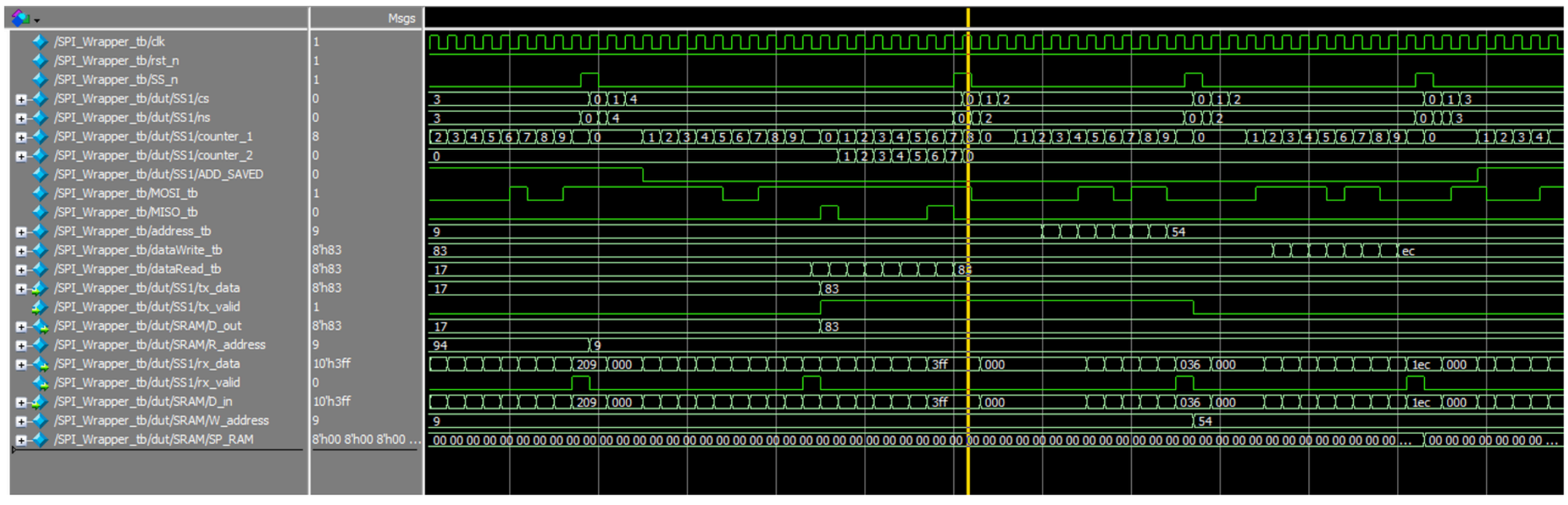
QUESTA-SIM

Do File

```
vlib work
vlog SPI.v SPI_tb.v
vsim -voptargs=+acc work.SPI_Wrapper_tb
add wave -position insertpoint \
sim:/SPI_Wrapper_tb/clk \
sim:/SPI_Wrapper_tb/rst_n \
sim:/SPI_Wrapper_tb/SS_n \
sim:/SPI_Wrapper_tb/dut/SS1/cs \
sim:/SPI_Wrapper_tb/dut/SS1/ns \
sim:/SPI_Wrapper_tb/dut/SS1/counter_1 \
sim:/SPI_Wrapper_tb/dut/SS1/counter_2 \
sim:/SPI_Wrapper_tb/dut/SS1/ADD_SAVED \
sim:/SPI_Wrapper_tb/MOSI_tb \
sim:/SPI_Wrapper_tb/MISO_tb \
sim:/SPI_Wrapper_tb/address_tb \
sim:/SPI_Wrapper_tb/dataWrite_tb \
sim:/SPI_Wrapper_tb/dataRead_tb \
sim:/SPI_Wrapper_tb/dut/SS1/tx_data \
sim:/SPI_Wrapper_tb/dut/SS1/tx_valid \
sim:/SPI_Wrapper_tb/dut/SRAM/D_out \
sim:/SPI_Wrapper_tb/dut/SRAM/R_address \
sim:/SPI_Wrapper_tb/dut/SS1/rx_data \
sim:/SPI_Wrapper_tb/dut/SS1/rx_valid \
sim:/SPI_Wrapper_tb/dut/SRAM/D_in \
sim:/SPI_Wrapper_tb/dut/SRAM/W_address \
sim:/SPI_Wrapper_tb/dut/SRAM/SP_RAM
run -all
#quit
```

QUESTA-SIM

Wave



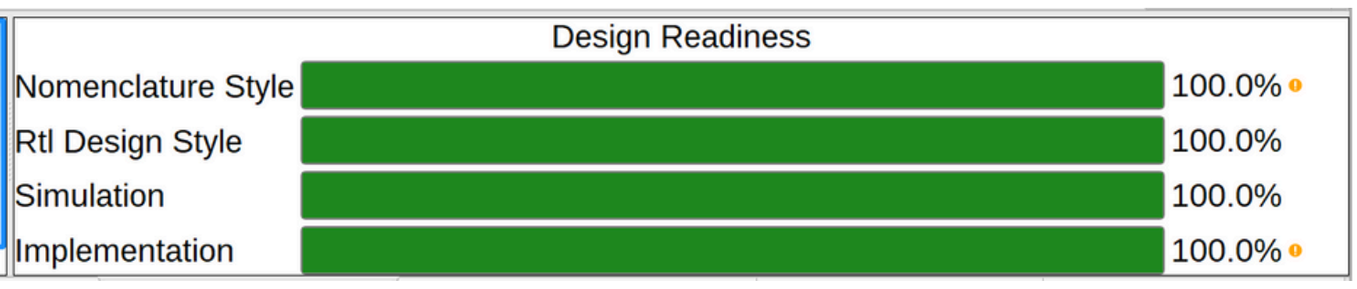
Output

```
# [OK] Everything is good
#
# ** Note: $stop    : SPI_tb.v(131)
#    Time: 12010 ns  Iteration: 1  Instance: /SPI_Wrapper_tb
# Break in Module SPI_Wrapper_tb at SPI_tb.v line 131
```

QUESTA-LINT

Lint Score

Quality Score: 100.0%



Lint Output

SPRAM module

Name	Count
Resolved(verified, fixed, waived)	1
Info	1
multi_ports_in_single_line	1

SPI Slave module

Name	Count
Resolved(verified, fixed, waived)	1
Info	1
multi_ports_in_single_line	1

SPI Wrapper module

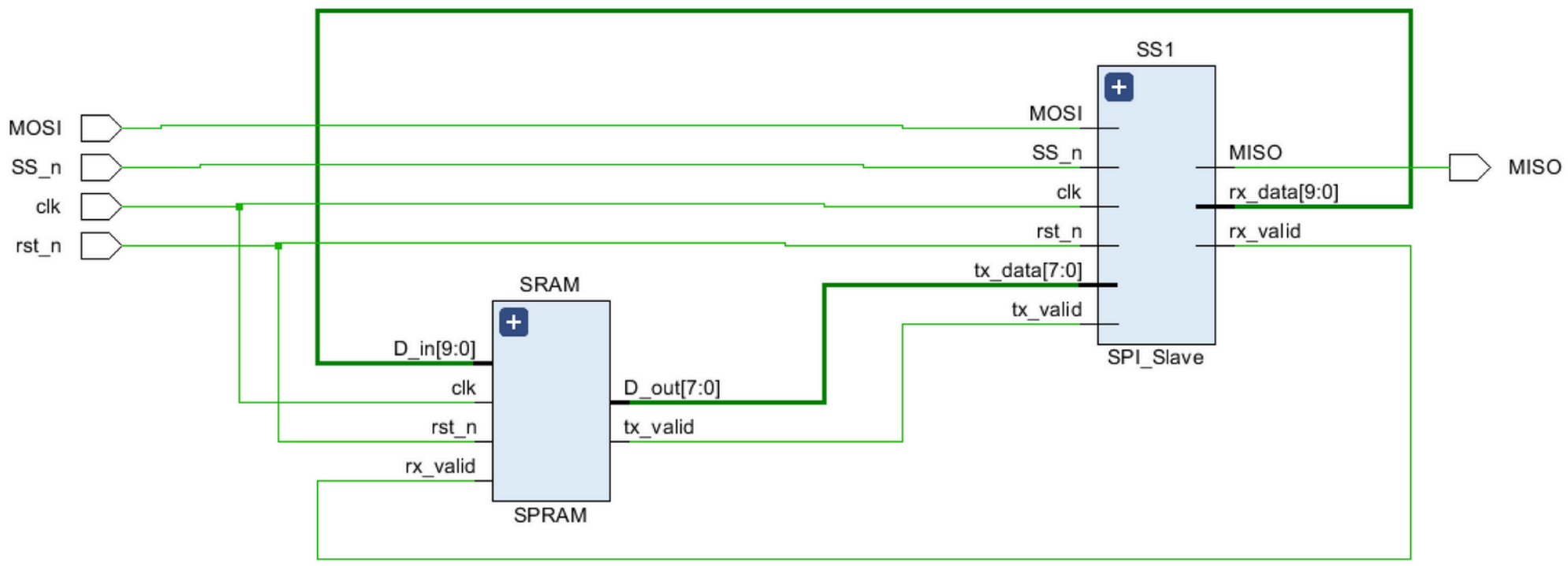
Name	Count
Resolved(verified, fixed, waived)	4
Info	4
line_char_large	1
multi_ports_in_single_line	3

VIVADO

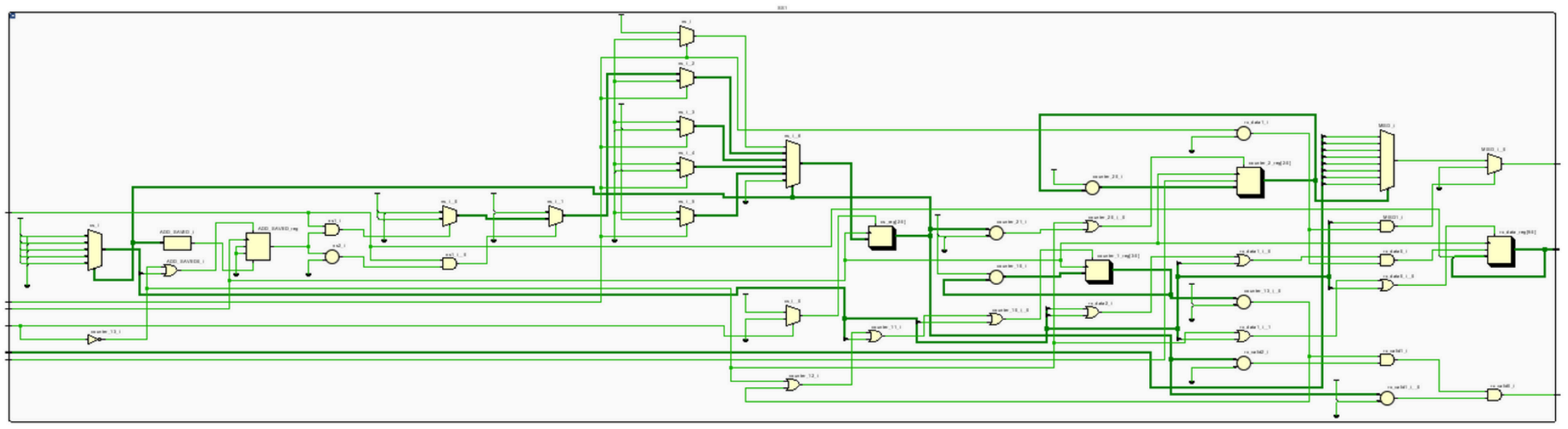
One-Hot Encoding

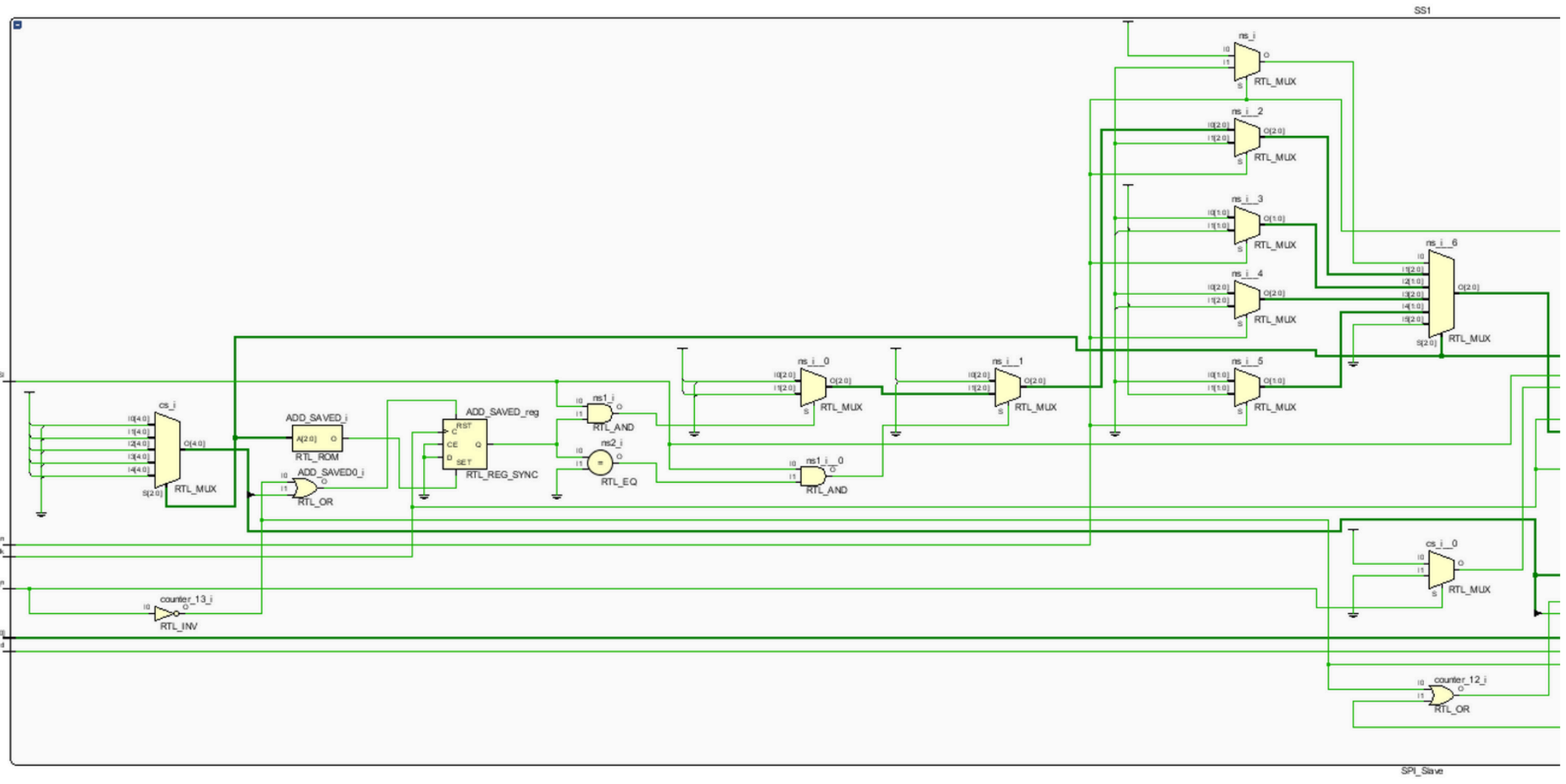
Elaborated Schematic

SPI Wrapper module



SPI Slave module



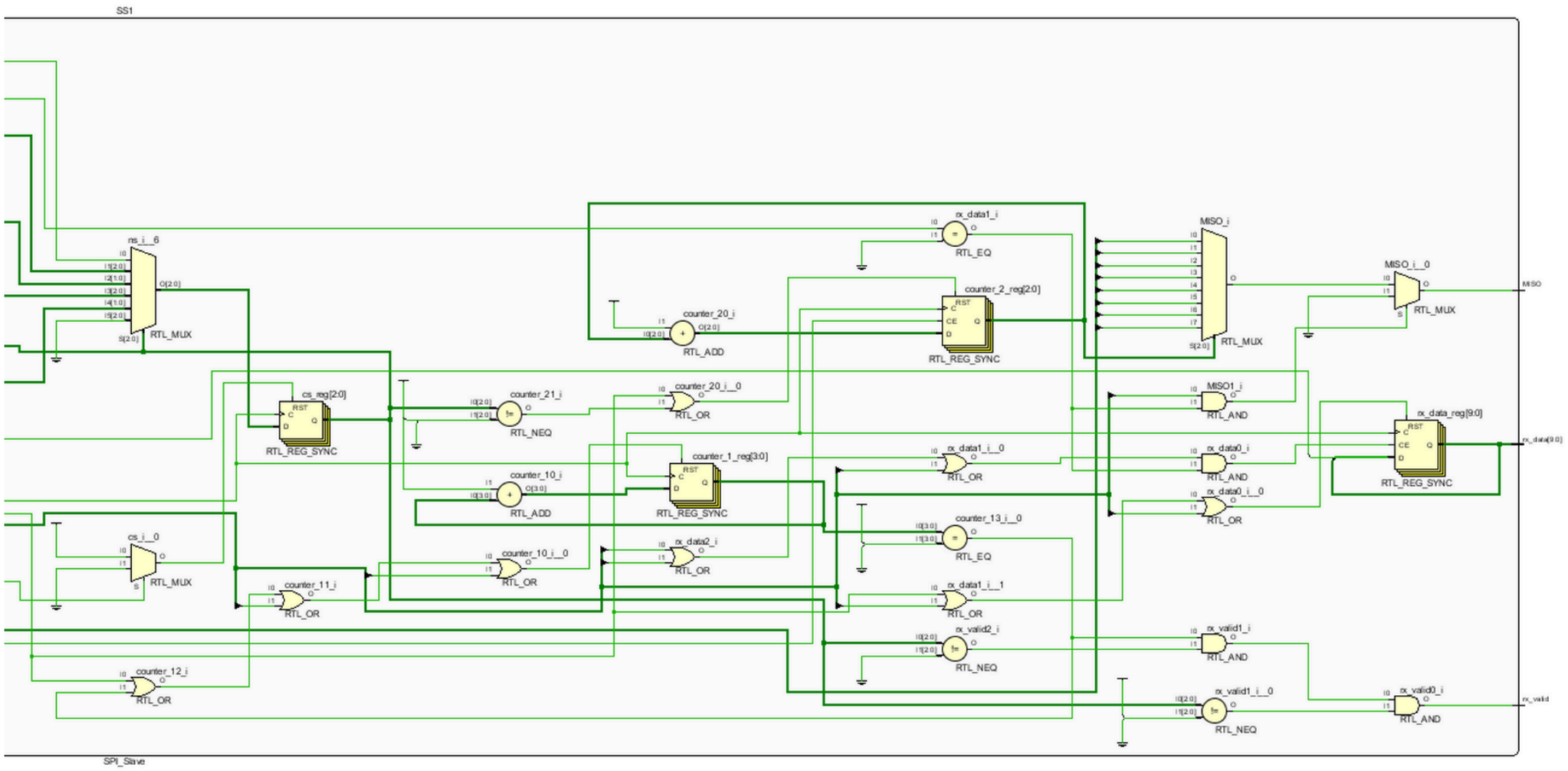


VIVADO

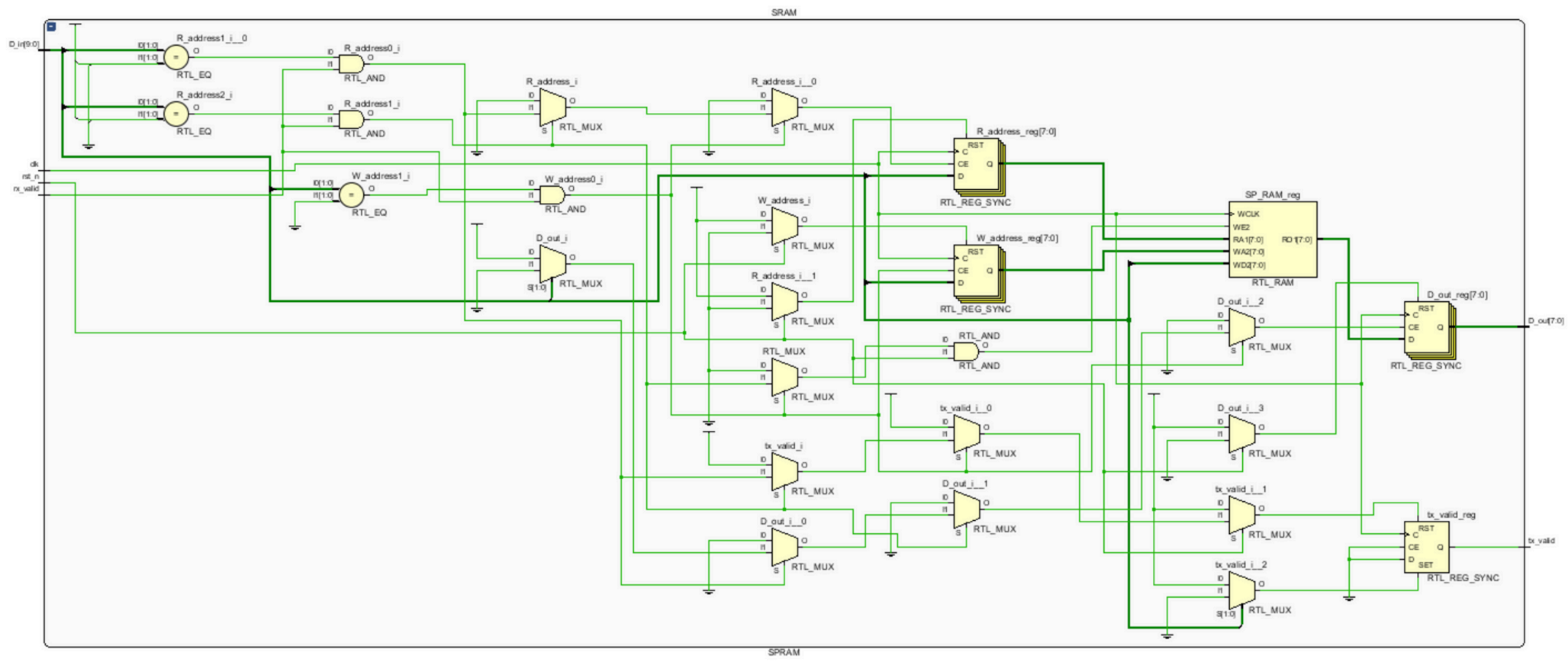
One-Hot Encoding

Elaborated Schematic

SPI Slave module



SPRAM module



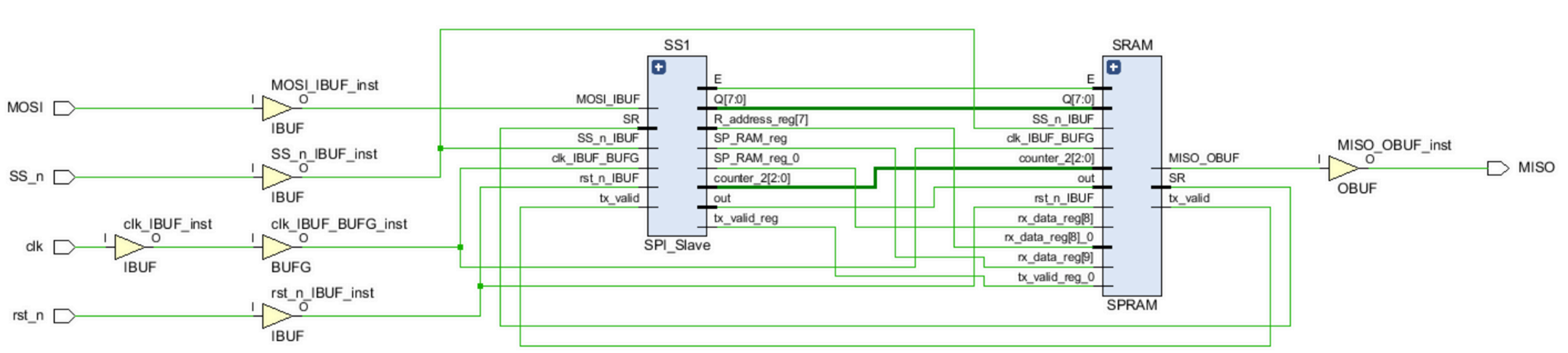
VIVADO

One-Hot Encoding

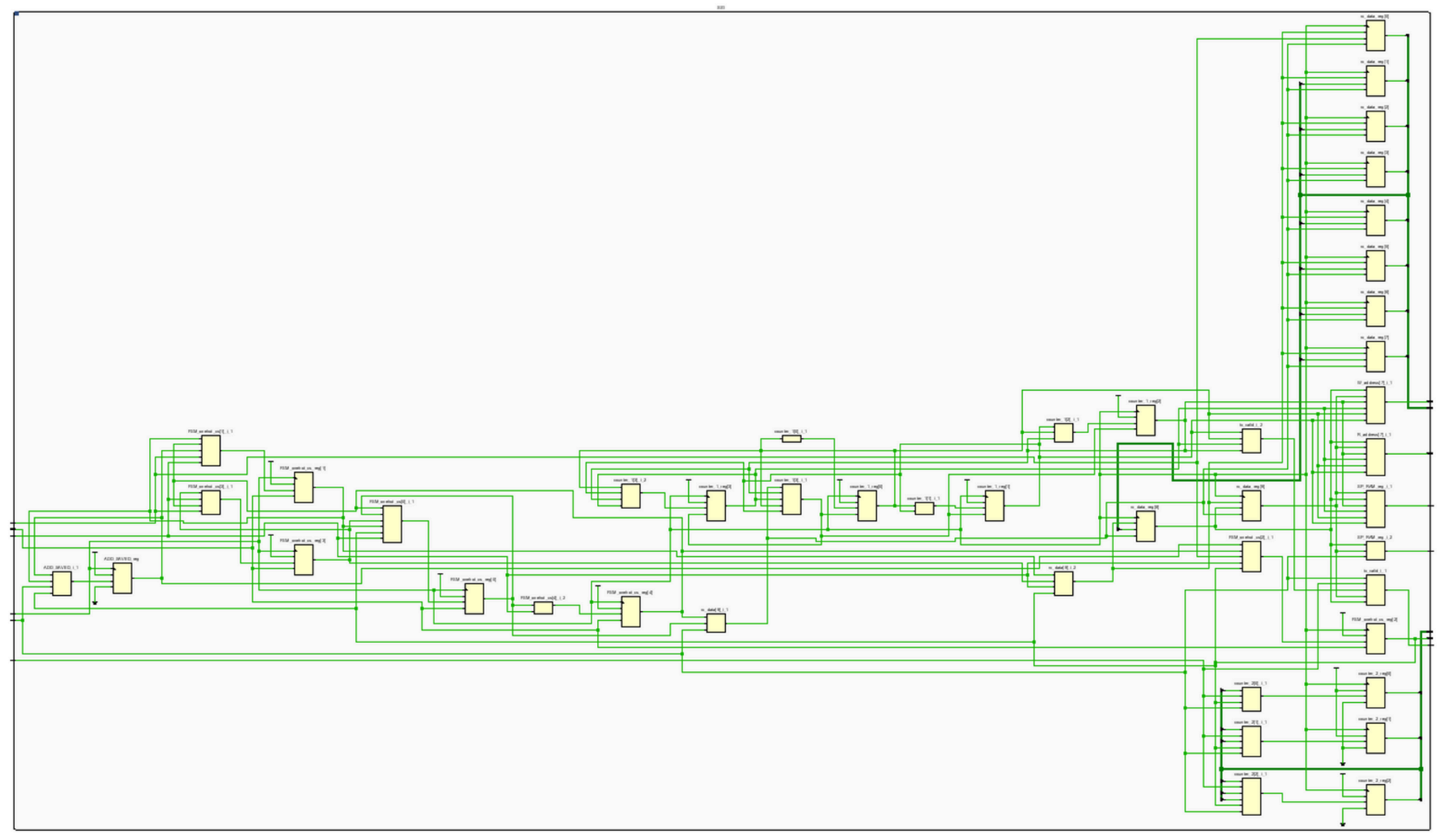
Elaborated Message

Synthesis Schematic

SPI Wrapper module

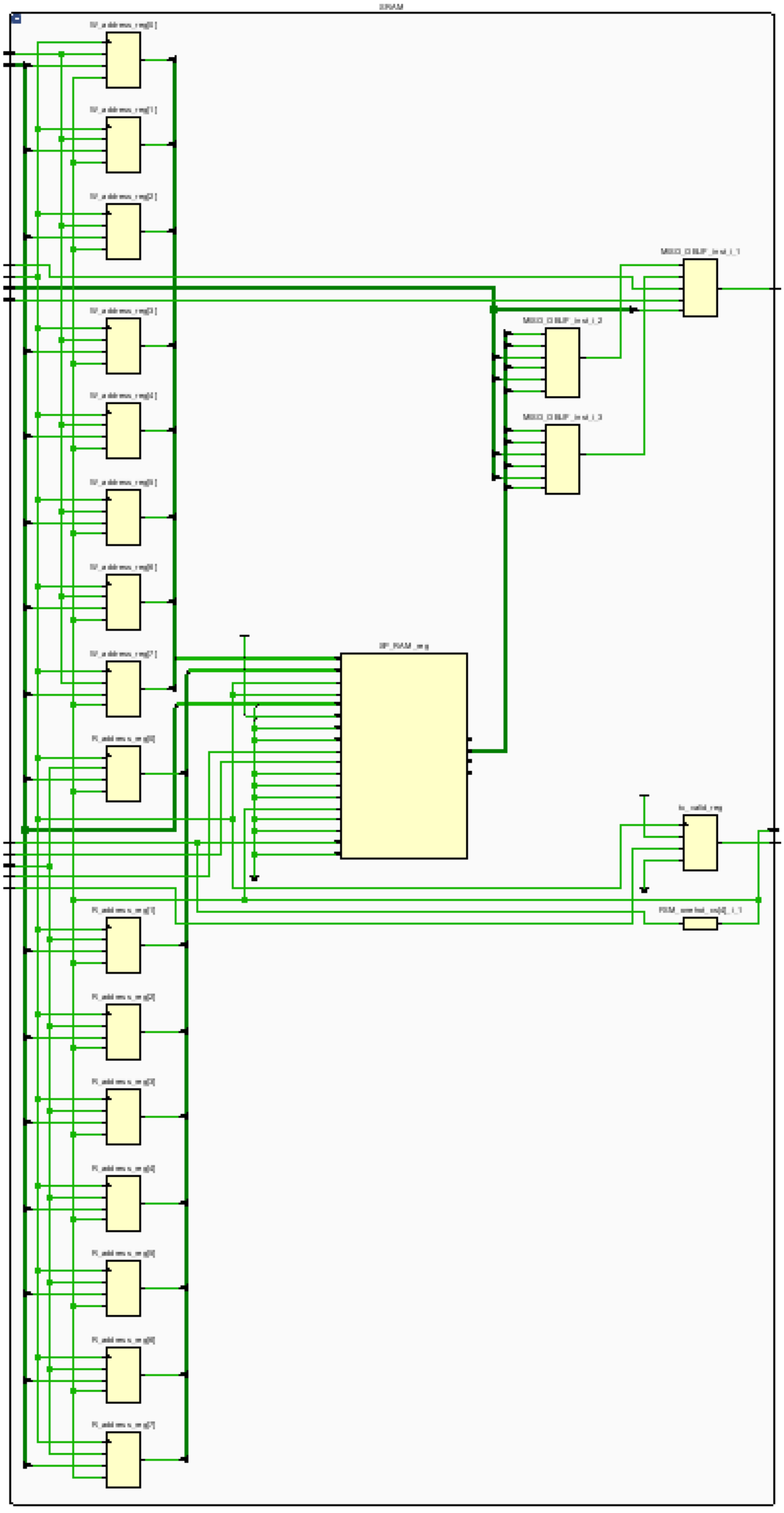


SPI Slave module



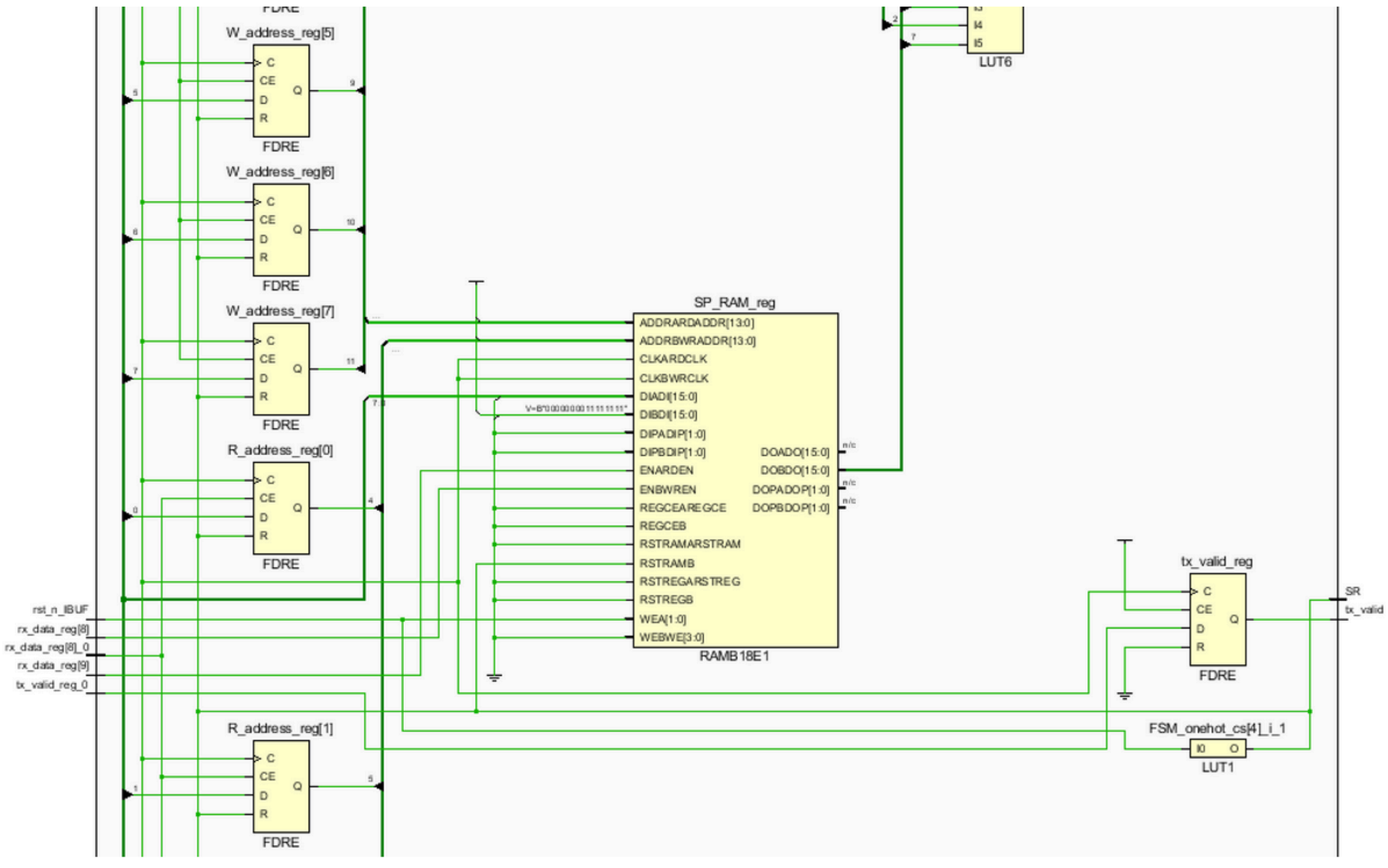
VIVADO

SPRAM module

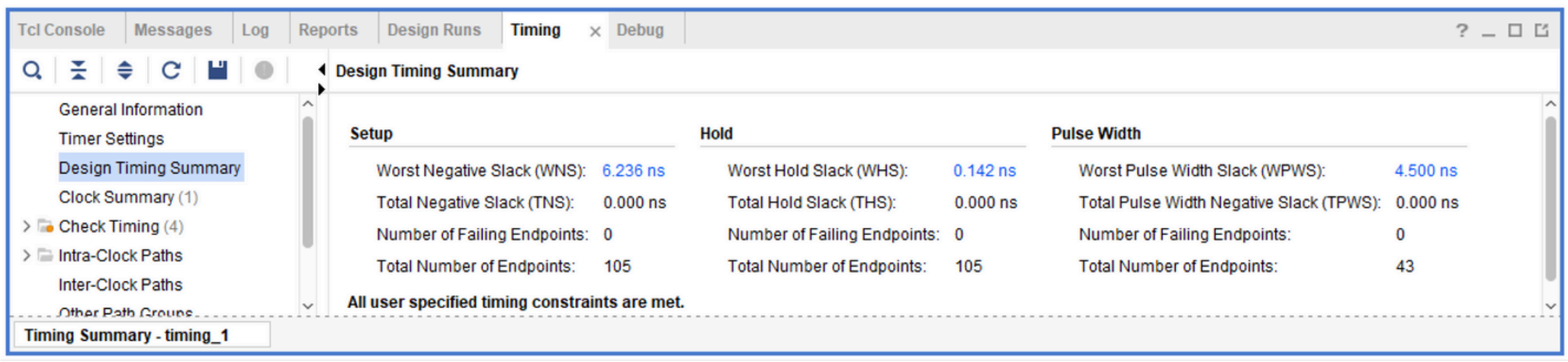


VIVADO

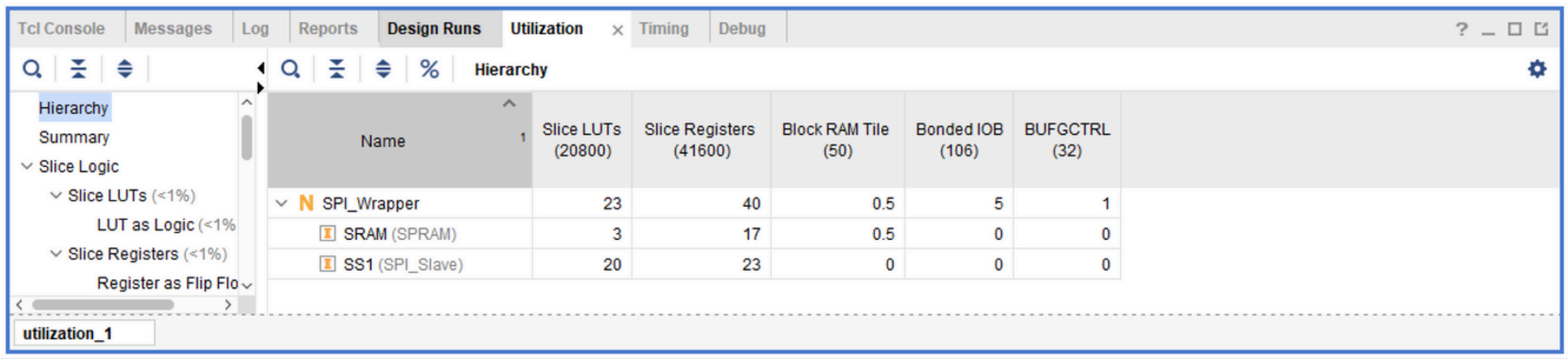
SPRAM module



Synthesis Timing Report

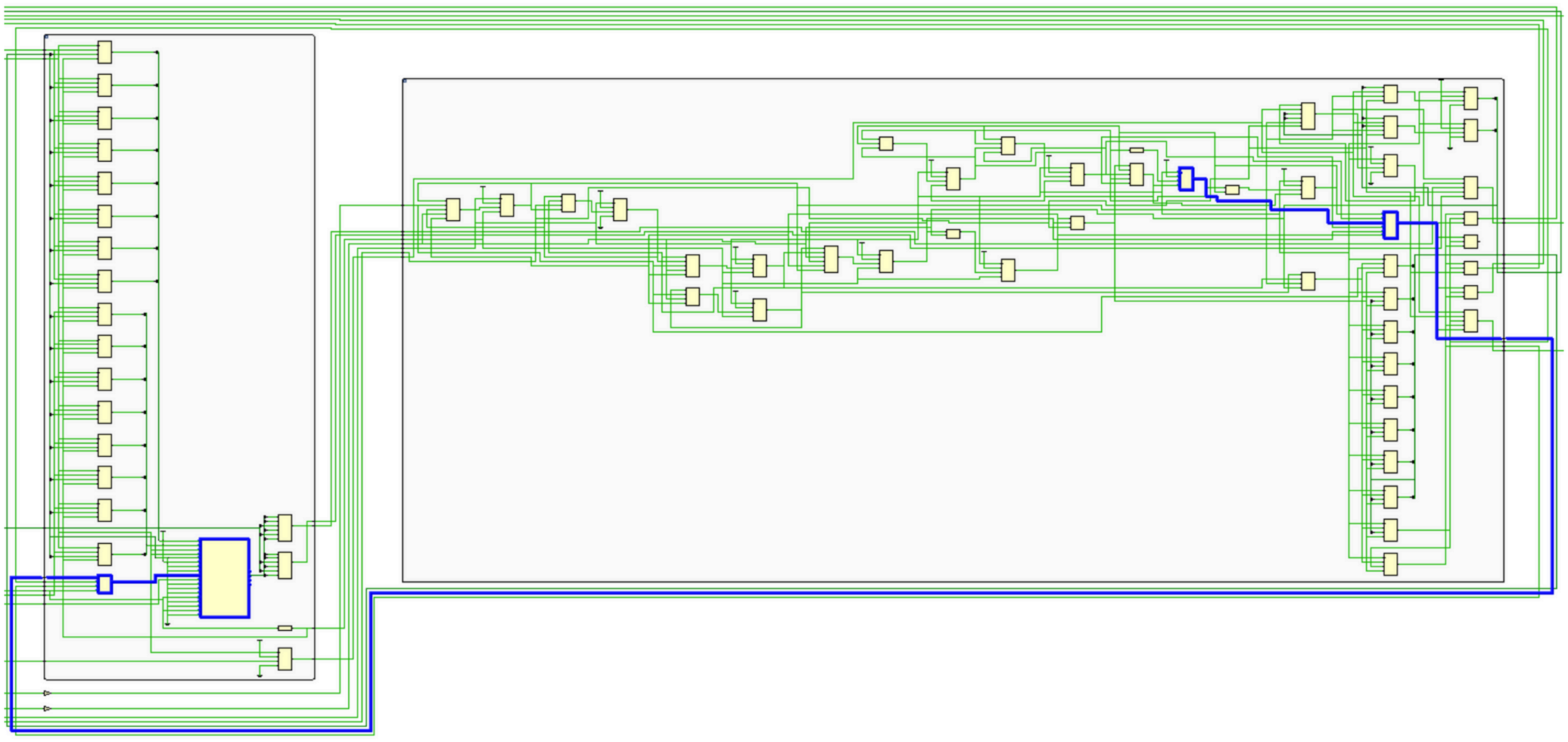


Synthesis Utilization Report



VIVADO

Critical Path



Synthesis Message

Tcl Console | Messages | Log | Reports | Design Runs | Debug

Warning (1) | Info (42) | Status (25) | Show All

Synthesis (1 warning, 33 infos)

- [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
- [Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI.v:174] (2 more like this)
- [Synth 8-5534] Detected attribute (* fsm_encoding = "one_hot" *) [SPI.v:61]
- [Synth 8-6155] done synthesizing module 'SPI_Slave' (1#1) [SPI.v:44] (2 more like this)
- [Device 21-403] Loading part xc7a35ticpg236-1L
- [Project 1-236] Implementation specific constraints were found while reading constraint file [Z:/github/DigtalDesignVerification/Project/SPI/Constraints.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [.Xil/SPI_Wrapper_propImpl.xdc].
  Resolution: To avoid this warning, move constraints listed in ['Undefined'] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.
- [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
- [Synth 8-5544] ROM "rx_valid" won't be mapped to Block RAM because address size (4) smaller than threshold (5) (6 more like this)
- [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'SPI_Slave'
- [Synth 8-4480] The timing for the instance i_0/SRAM/SP_RAM_reg (implemented as a block RAM) might be sub-optimal as no optional output register could be merged into the block ram. Providing additional output register may help in improving timing. (1 more like this)
- [Project 1-571] Translating synthesized netlist
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-570] Preparing netlist for logic optimization (1 more like this)
- [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
- [Project 1-111] Unisim Transformation Summary:
  No Unisim elements were transformed. (1 more like this)
- [Common 17-83] Releasing license: Synthesis
- [Constraints 18-5210] No constraint will be written out.
- [Common 17-1381] The checkpoint 'C:/Users/muhammadwael/Vivado/SPI_project/SPI_project.runs/synth_1/SPI_Wrapper.dcp' has been generated.
- [runtcl-4] Executing : report_utilization -file SPI_Wrapper_utilization_synth.rpt -pb SPI_Wrapper_utilization_synth.pb
- [Common 17-206] Exiting Vivado at Tue Aug 5 16:14:55 2025...

Synthesized Design (6 infos)

General Messages (6 infos)

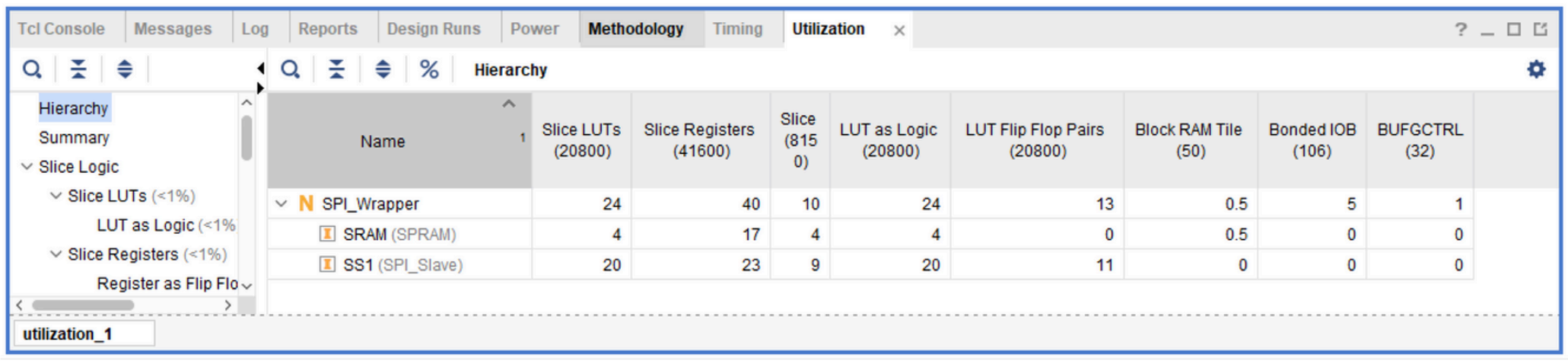
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement

VIVADO

Synthesis Report

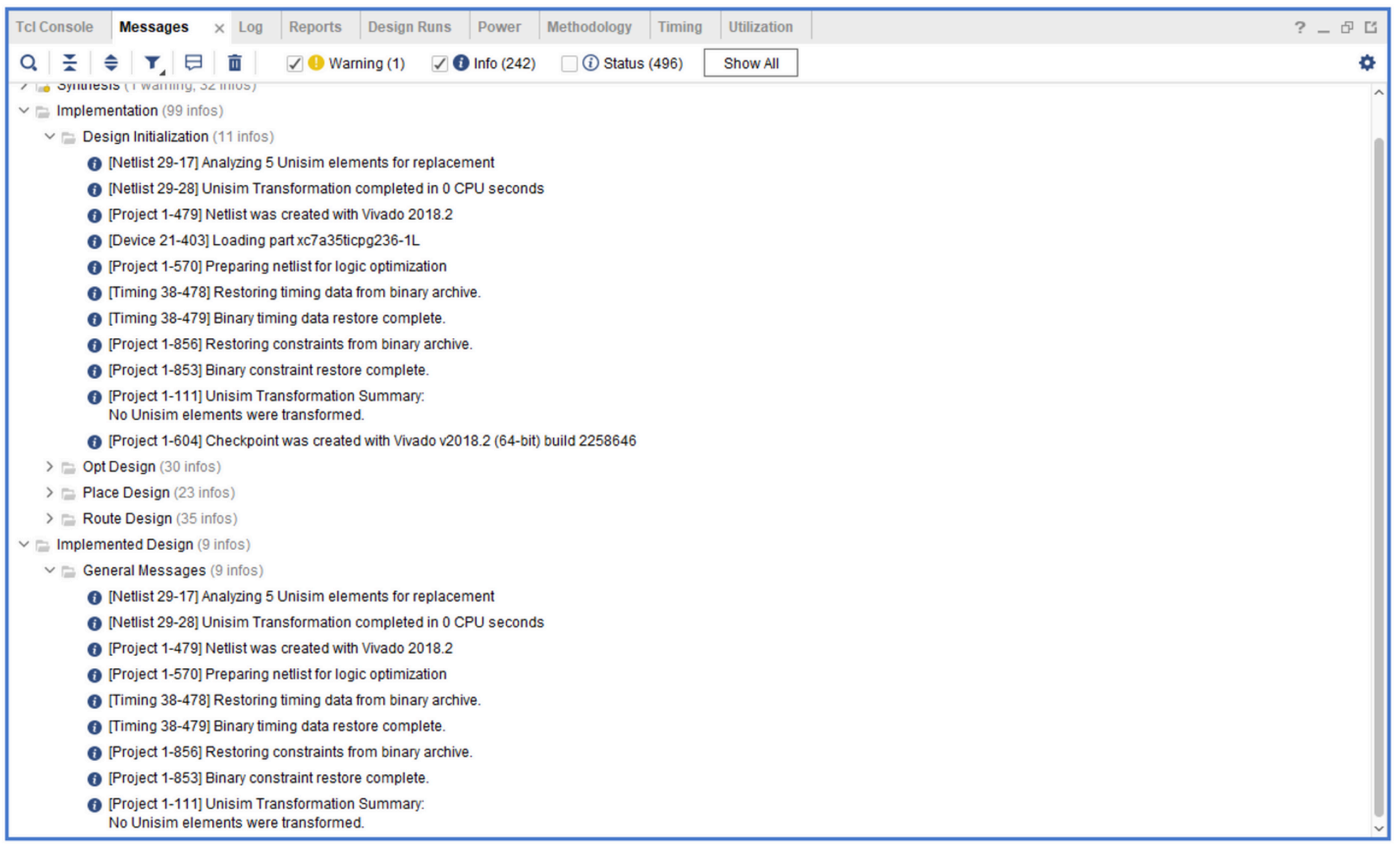
```
INFO: [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
INFO: [Synth 8-5544] ROM "ADD_SAVED" won't be mapped to Block RAM because address size (3) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "tx_valid" won't be mapped to Block RAM because address size (2) smaller than threshold (5)
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |                Previous Encoding
---------------------------------------------------------------------------------------------------
                    IDLE |                            00001 |                             000
                 CHK_CMD |                            10000 |                             001
                READ_ADD |                            00010 |                             011
               READ_DATA |                            00100 |                             100
                   WRITE |                            01000 |                             010
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'SPI_Slave'
---------------------------------------------------------------------------------------------------
```

Implementation Utilization Report



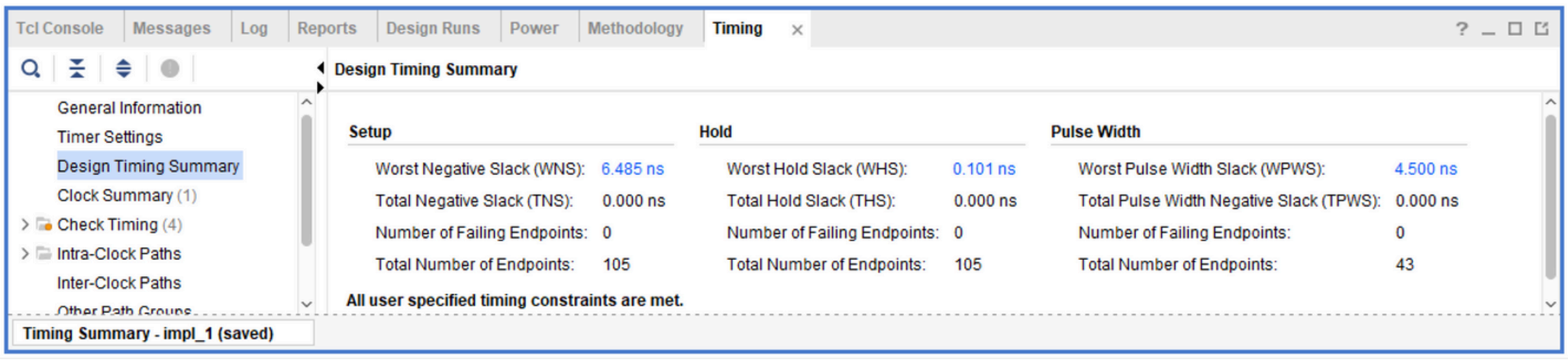
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N SPI_Wrapper	24	40	10	24	13	0.5	5	1
SRAM (SPRAM)	4	17	4	4	0	0.5	0	0
SS1 (SPI_Slave)	20	23	9	20	11	0	0	0

Implementation Message

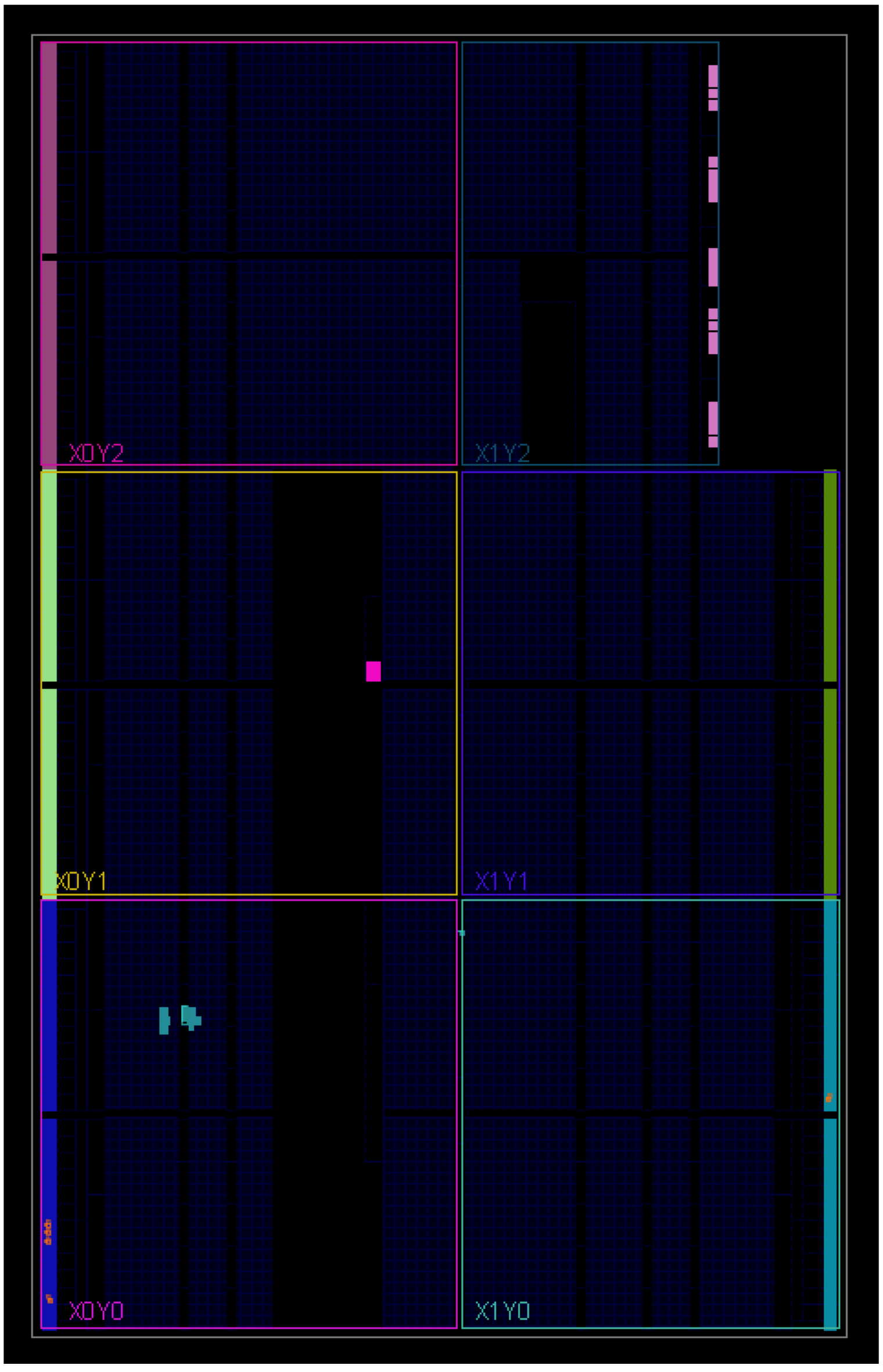


VIVADO

Implementation Timing Report



Device

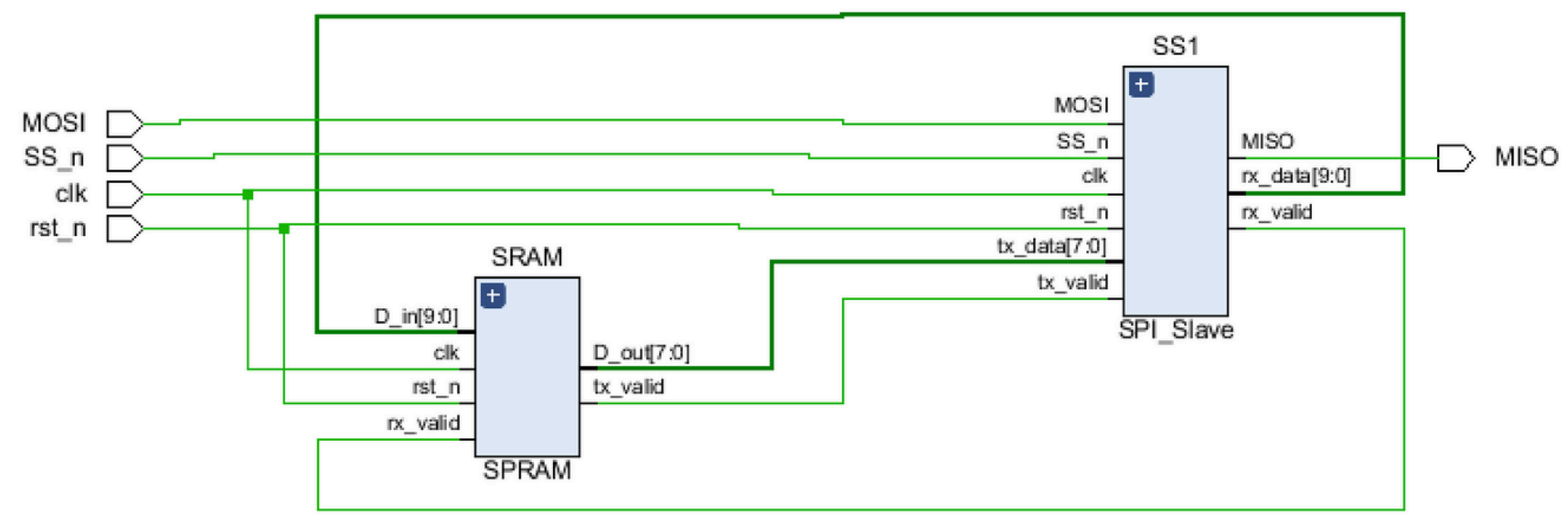


VIVADO

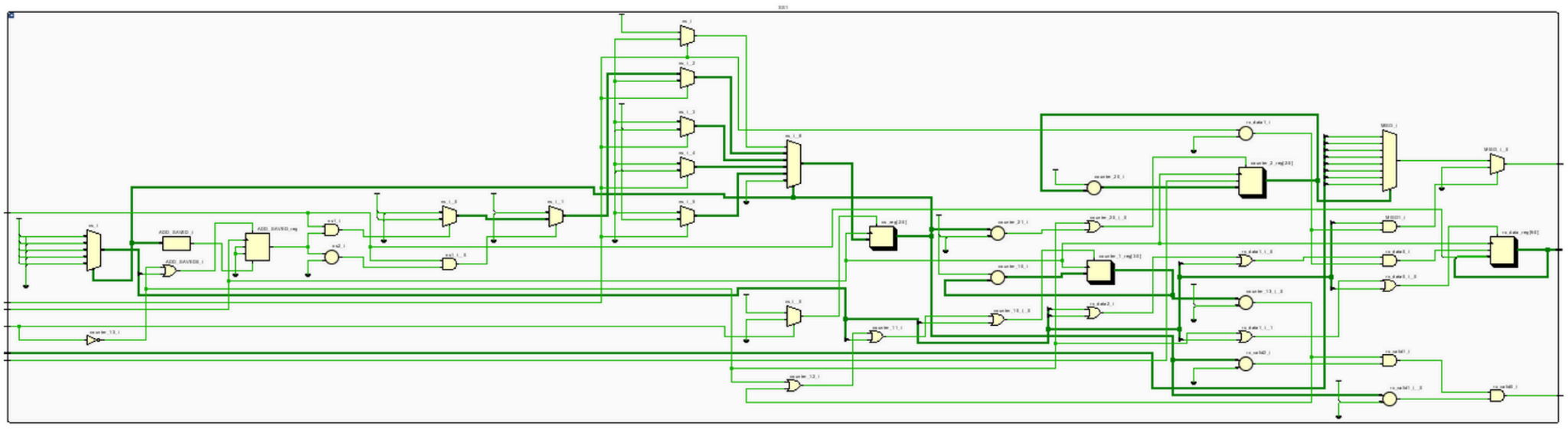
Gray Encoding

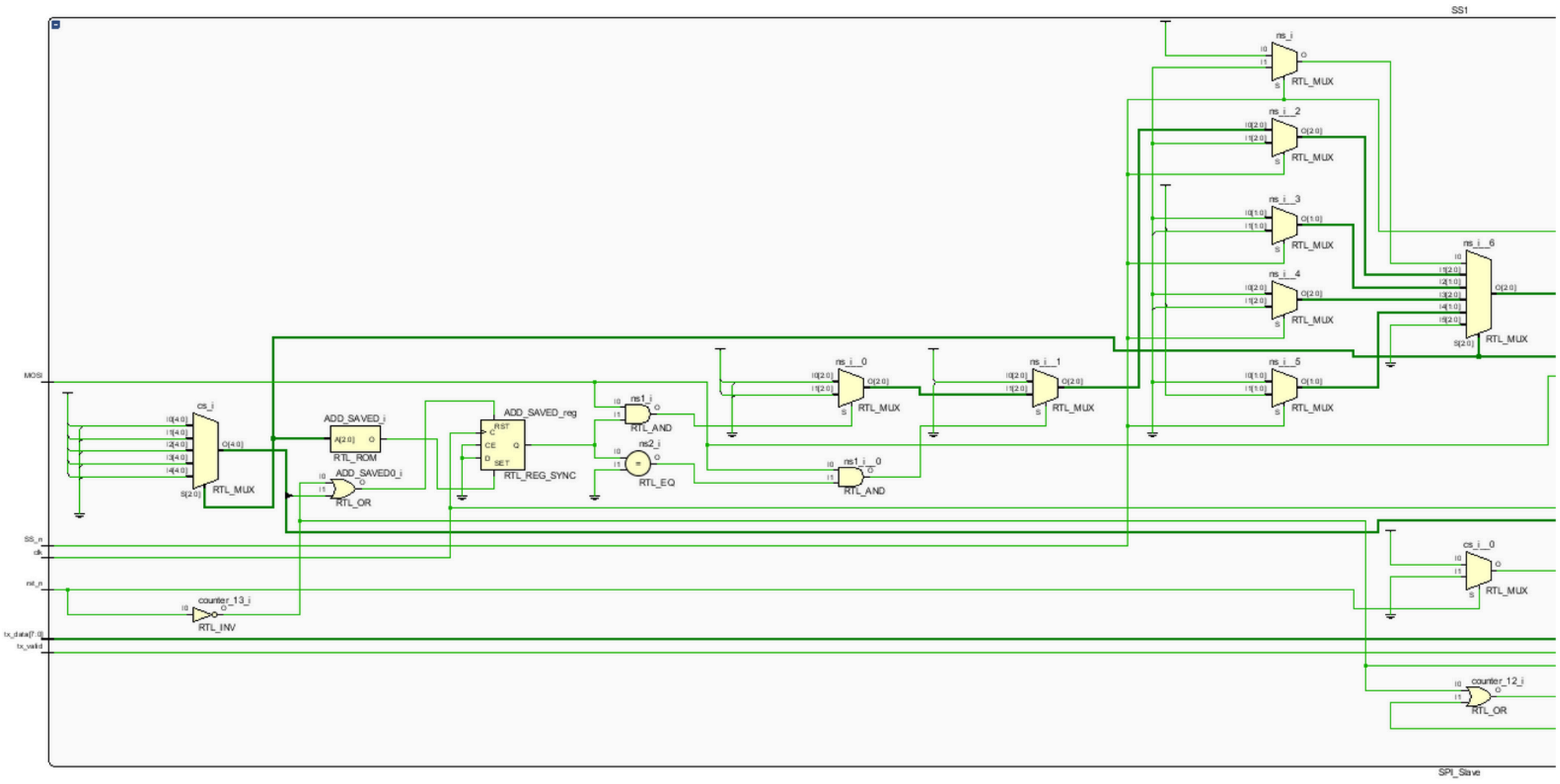
Elaborated Schematic

SPI Wrapper module



SPI Slave module



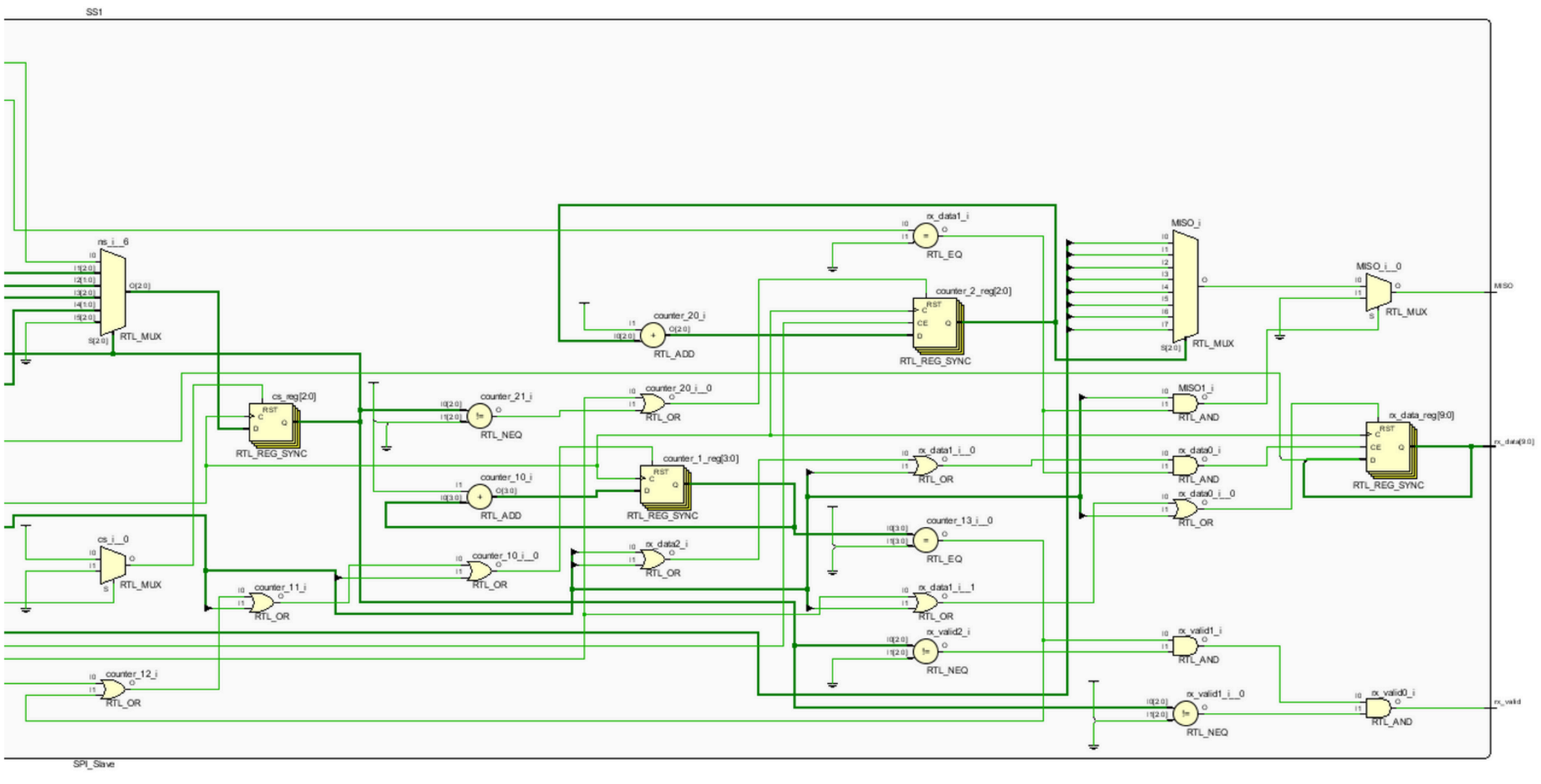


VIVADO

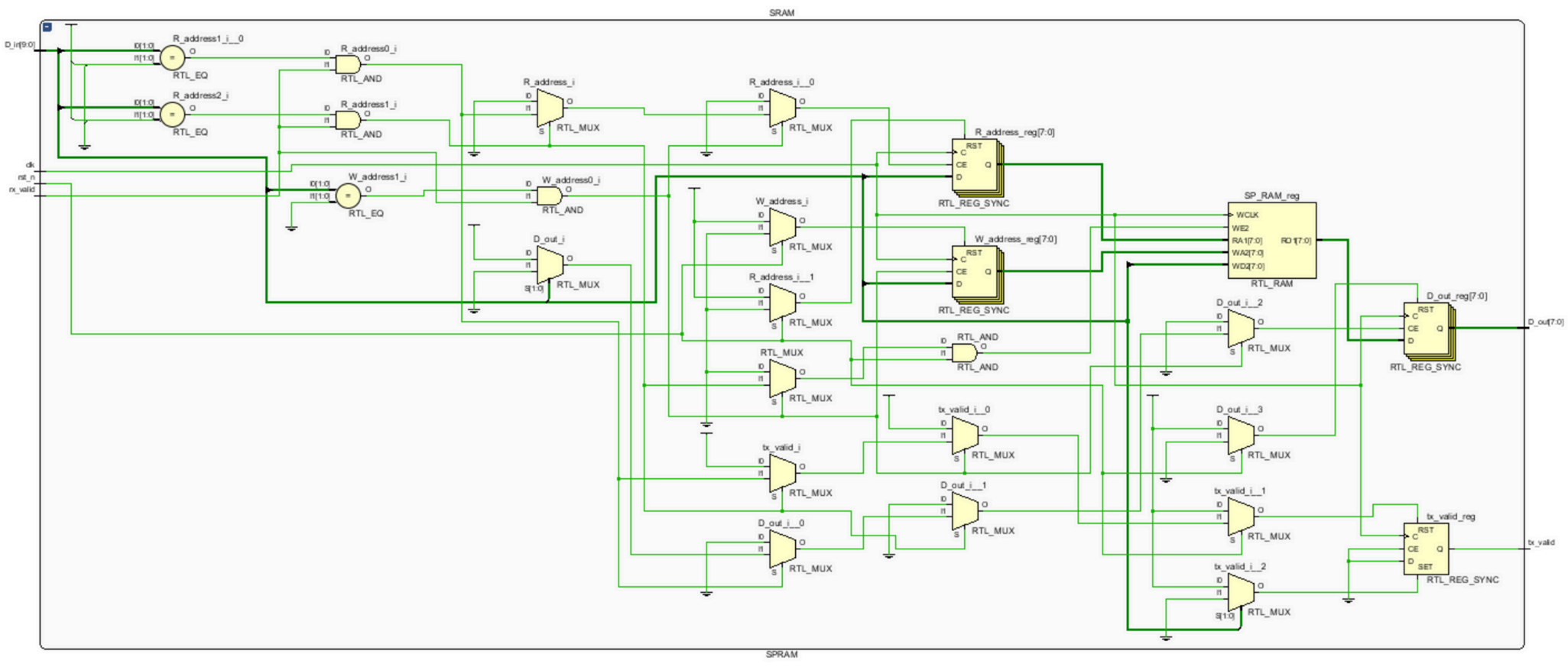
One-Hot Encoding

Elaborated Schematic

SPI Slave module



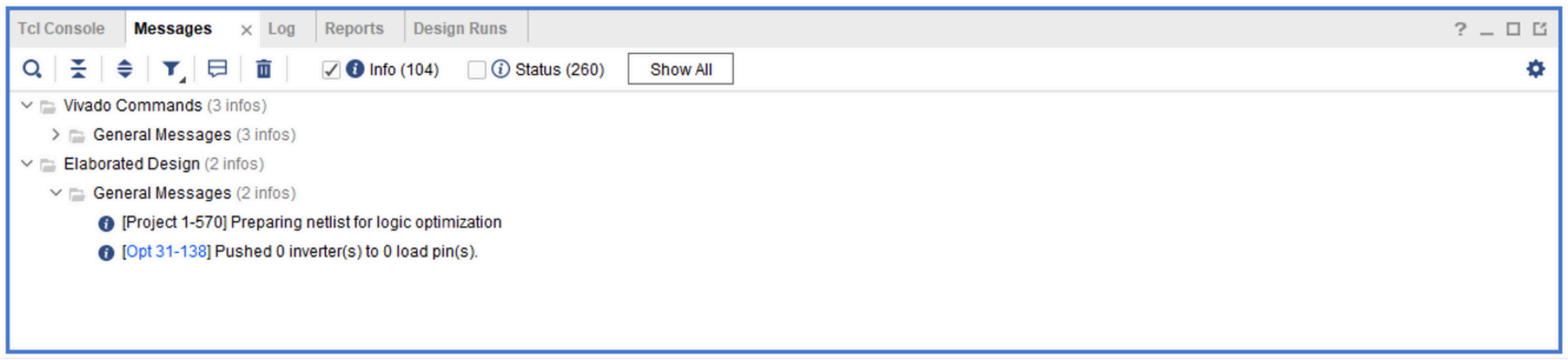
SPRAM module



VIVADO

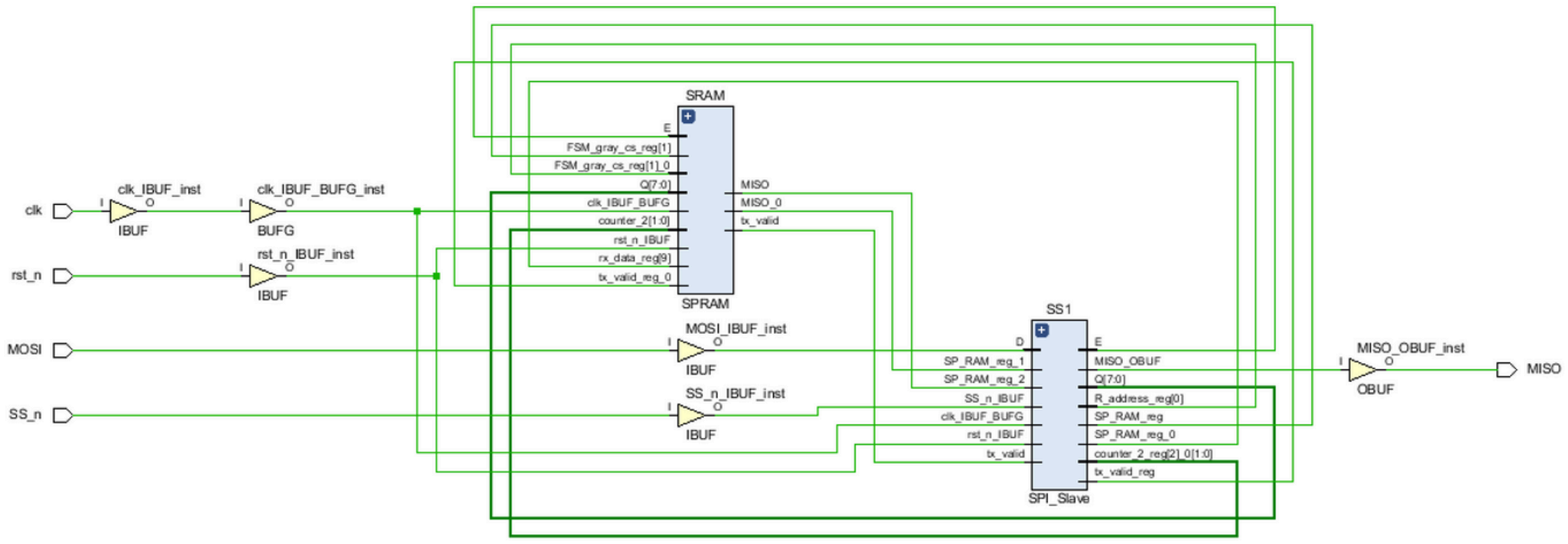
One-Hot Encoding

Elaborated Message

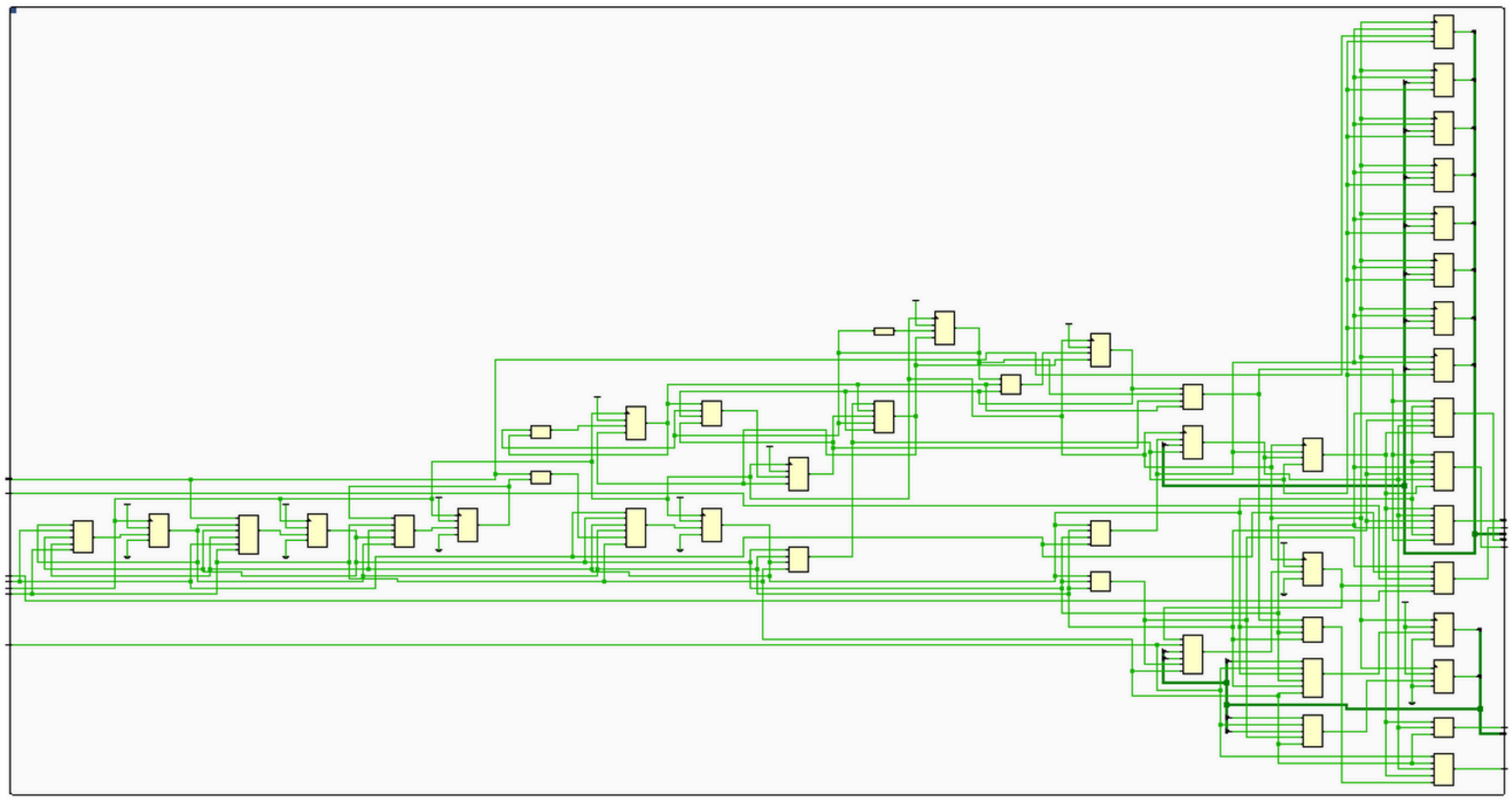


Synthesis Schematic

SPI Wrapper module

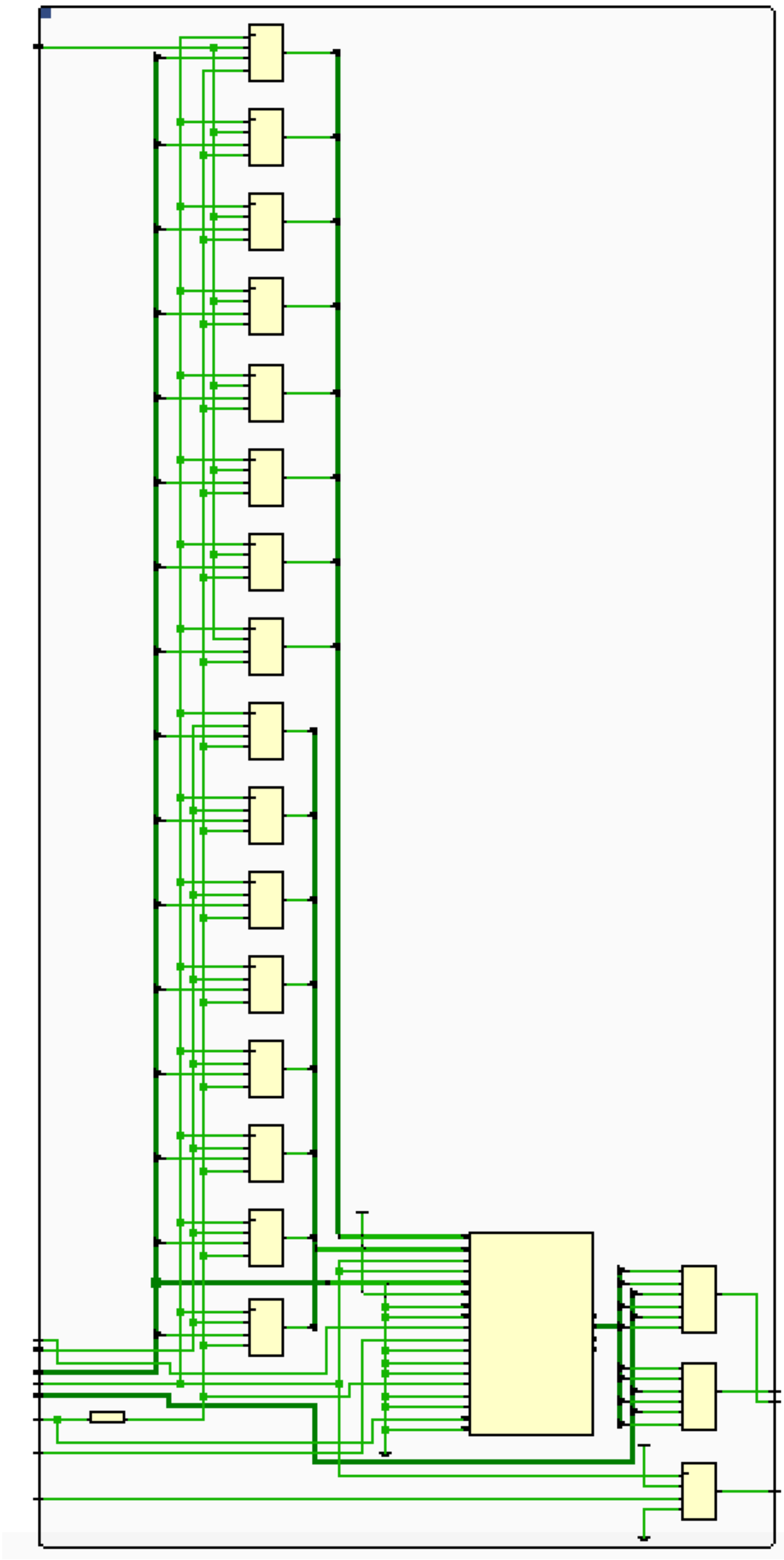


SPI Slave module



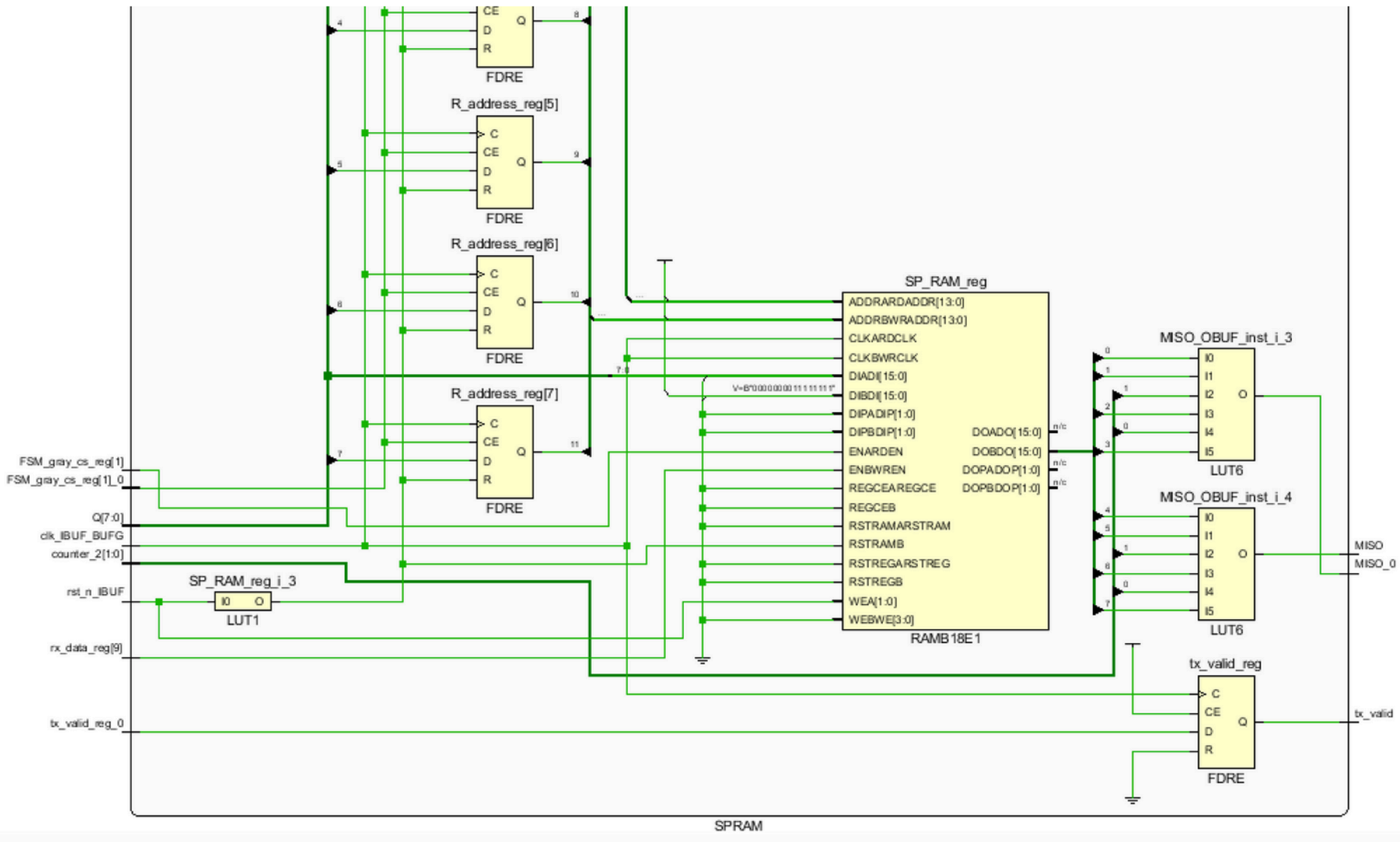
VIVADO

SPRAM module

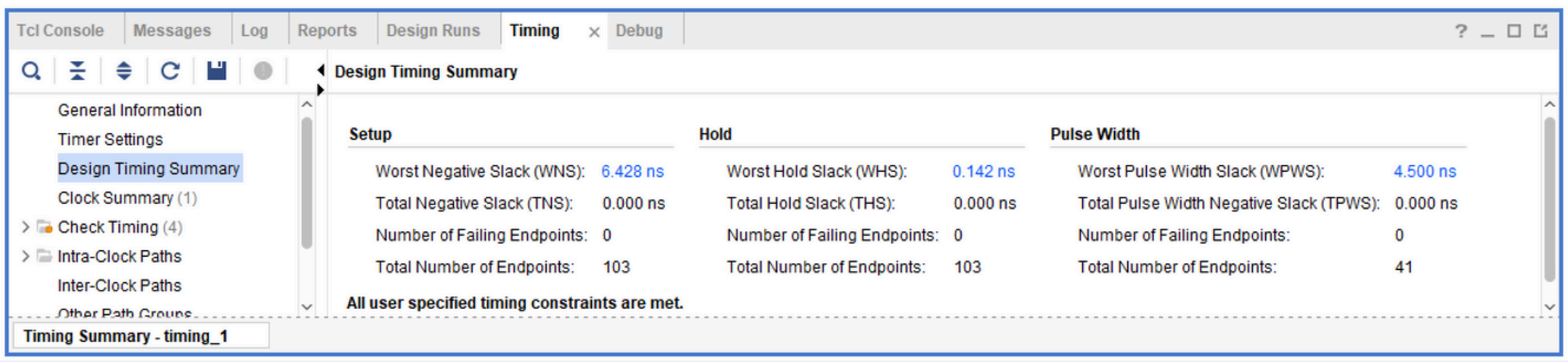


VIVADO

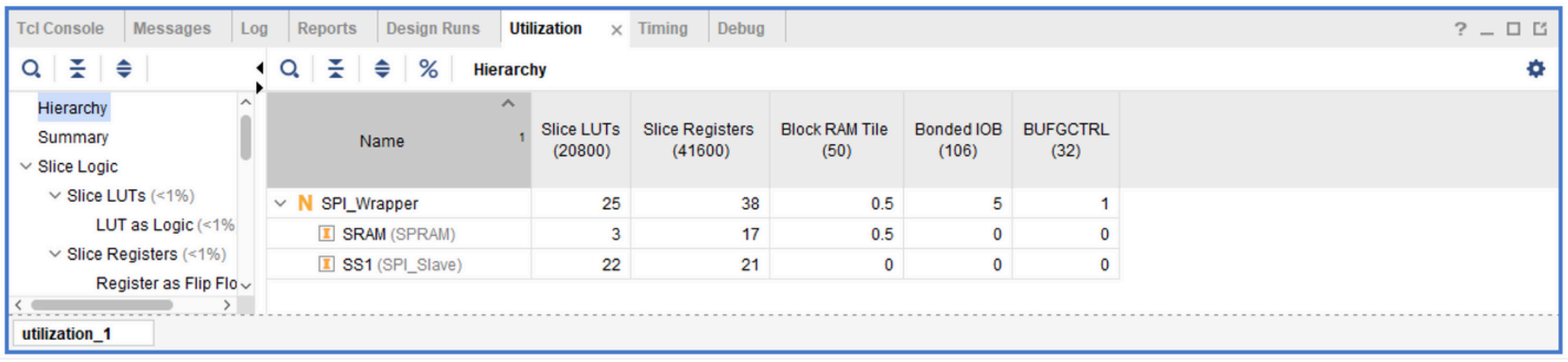
SPRAM module



Synthesis Timing Report

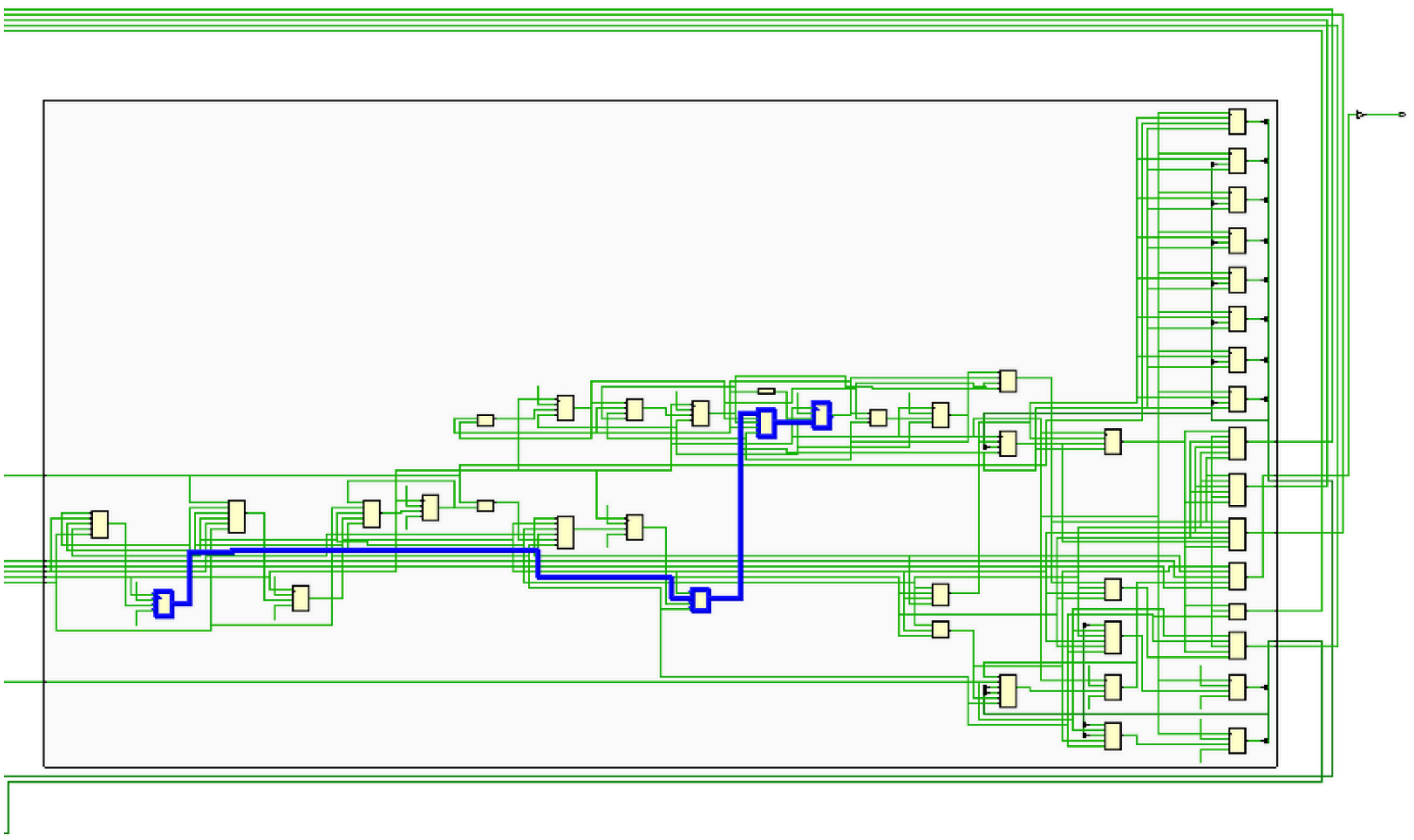


Synthesis Utilization Report

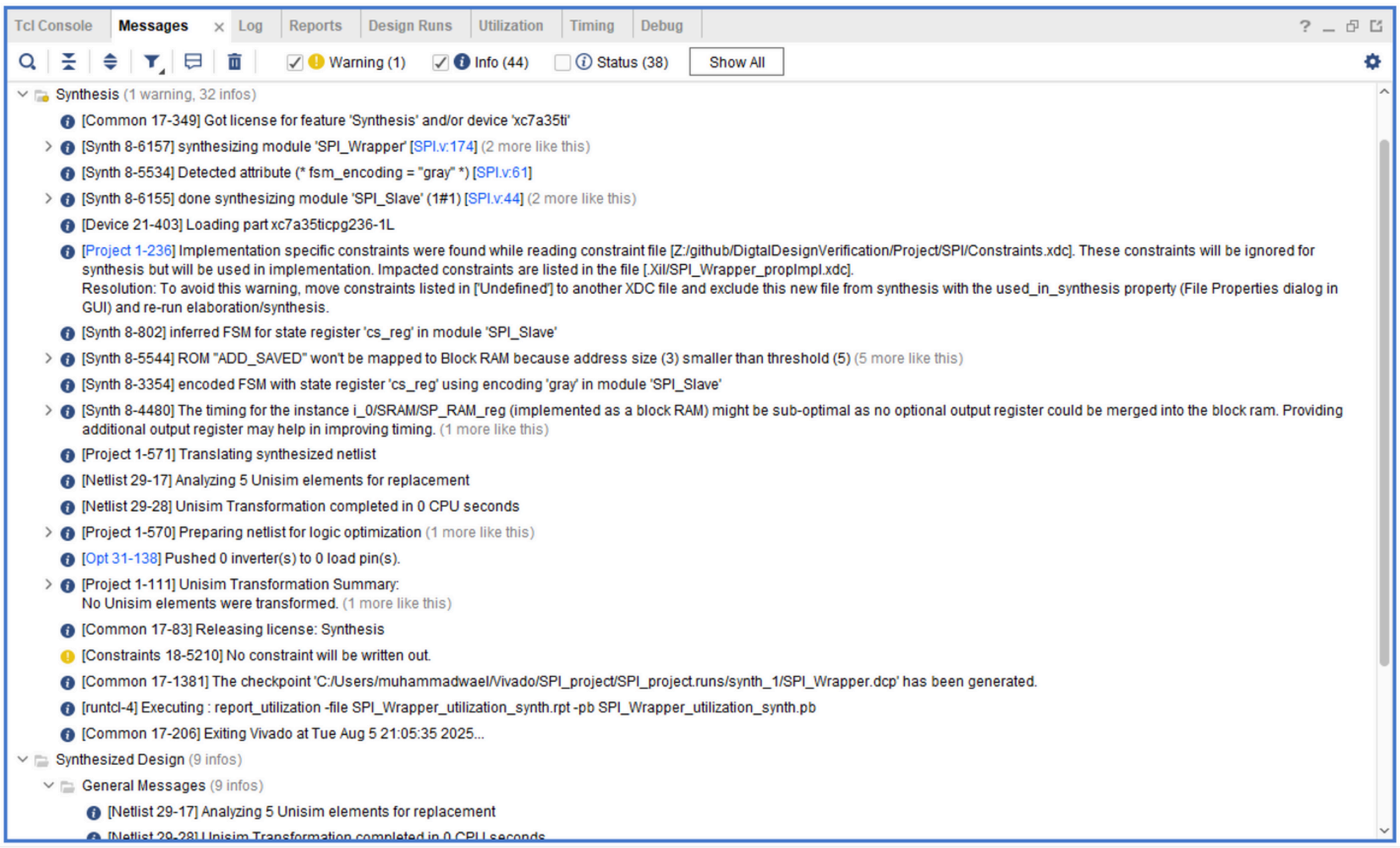


VIVADO

Critical Path



Synthesis Message

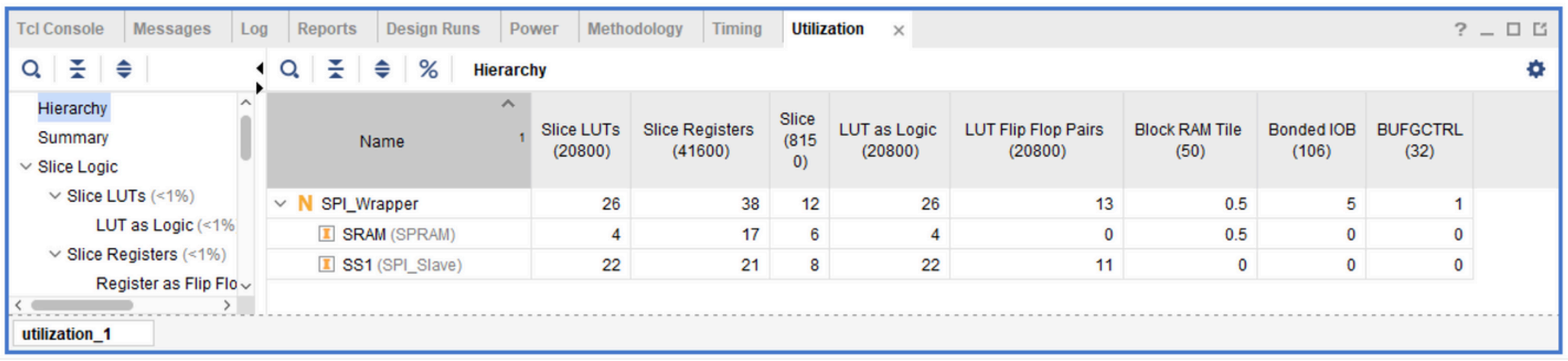


VIVADO

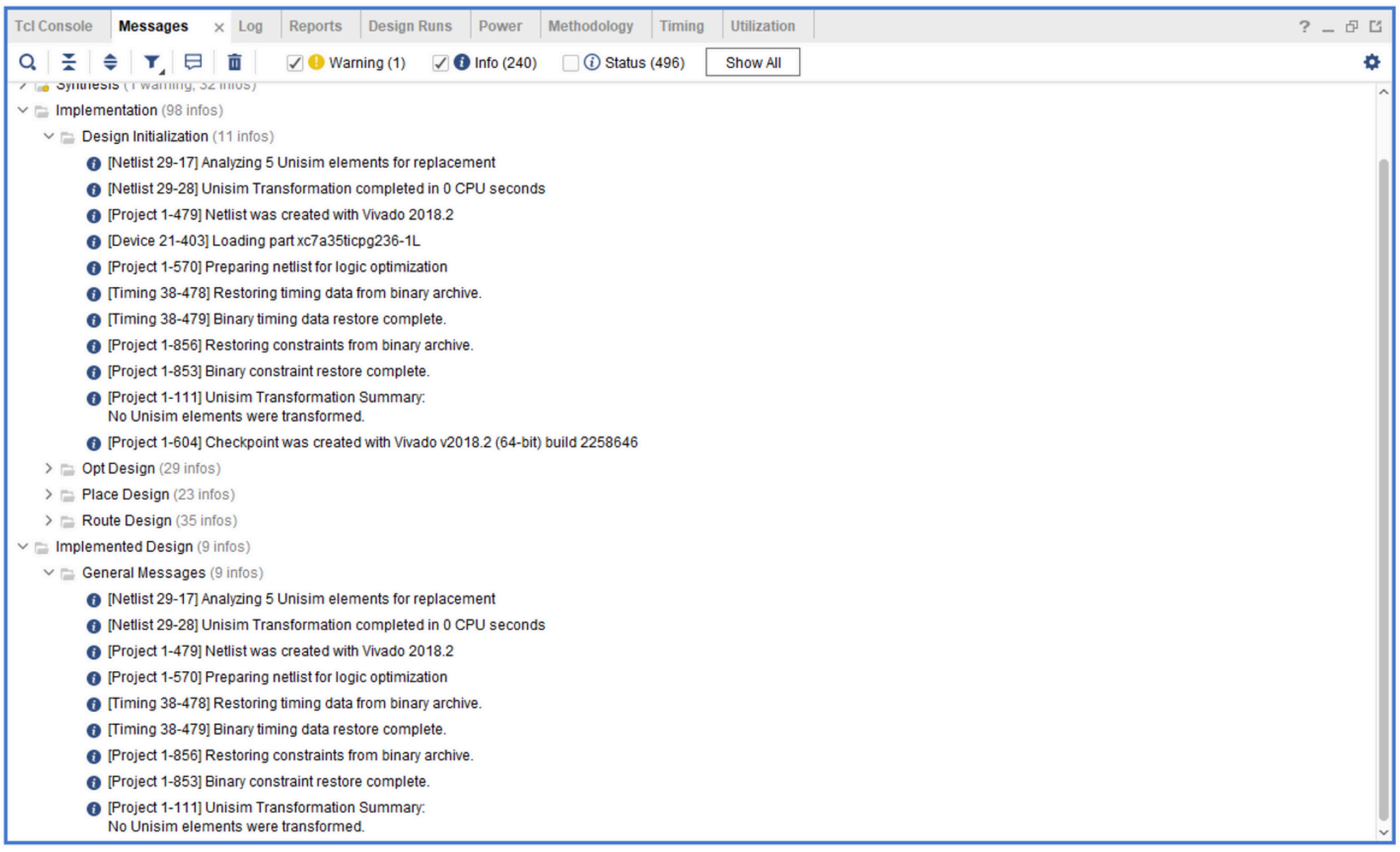
Synthesis Report

```
Finished applying 'set_property' XDC Constraints : Time (s): cpu = 00:00:31 ; elapsed = 00:00:39 . Memory (MB): peak = 791.410 ; gain = 506.445
---------------------------------------------------------------------------------
INFO: [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
INFO: [Synth 8-5544] ROM "ADD_SAVED" won't be mapped to Block RAM because address size (3) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "tx_valid" won't be mapped to Block RAM because address size (2) smaller than threshold (5)
-----------------------------------------------------------------------------------------------------
                   State |                     New Encoding |               Previous Encoding
-----------------------------------------------------------------------------------------------------
                    IDLE |                              000 |                              000
                  CHK_CMD |                              111 |                              001
                 READ_ADD |                              001 |                              011
                READ_DATA |                              011 |                              100
                   WRITE |                              010 |                              010
-----------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'gray' in module 'SPI_Slave'
```

Implementation Utilization Report

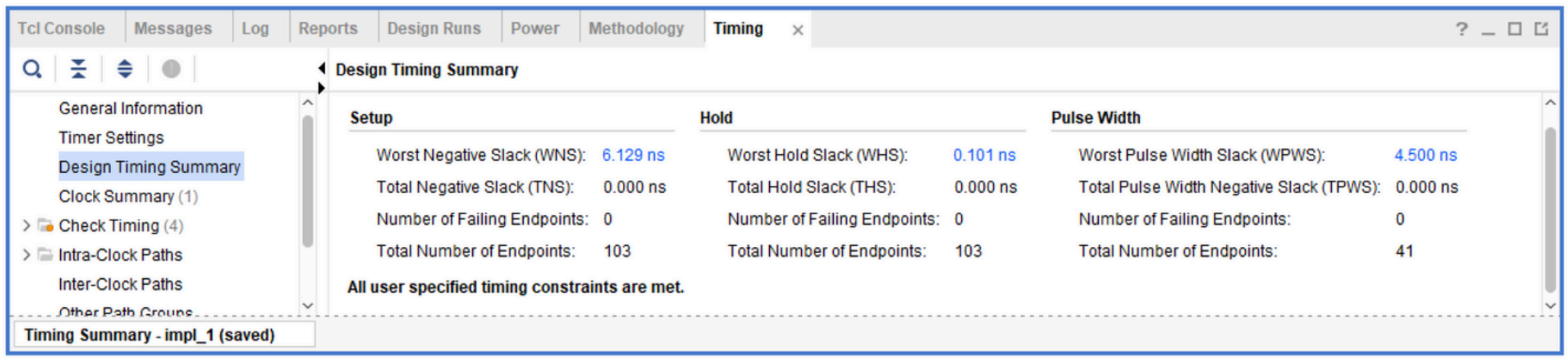


Implementation Message

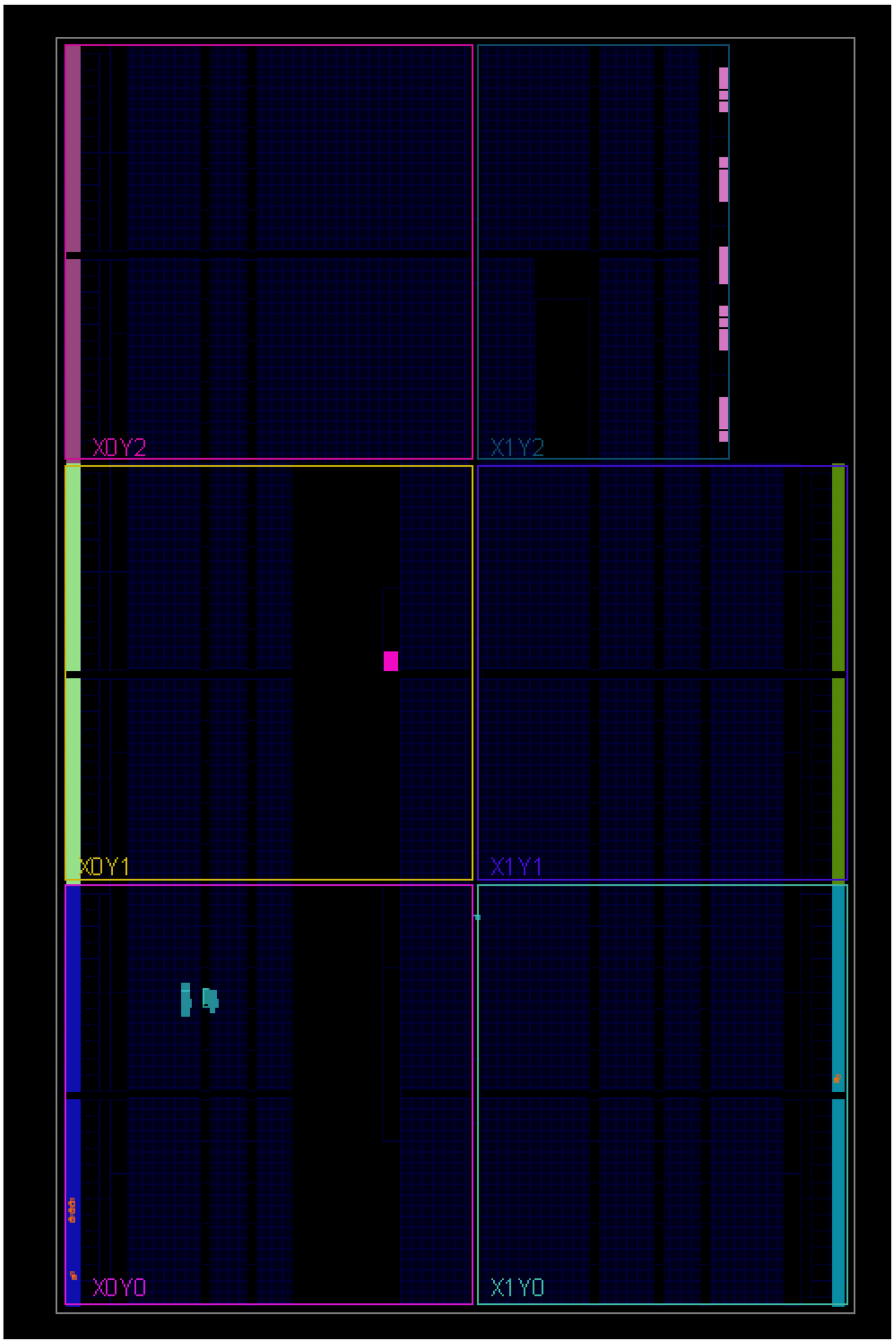


VIVADO

Implementation Timing Report



Device

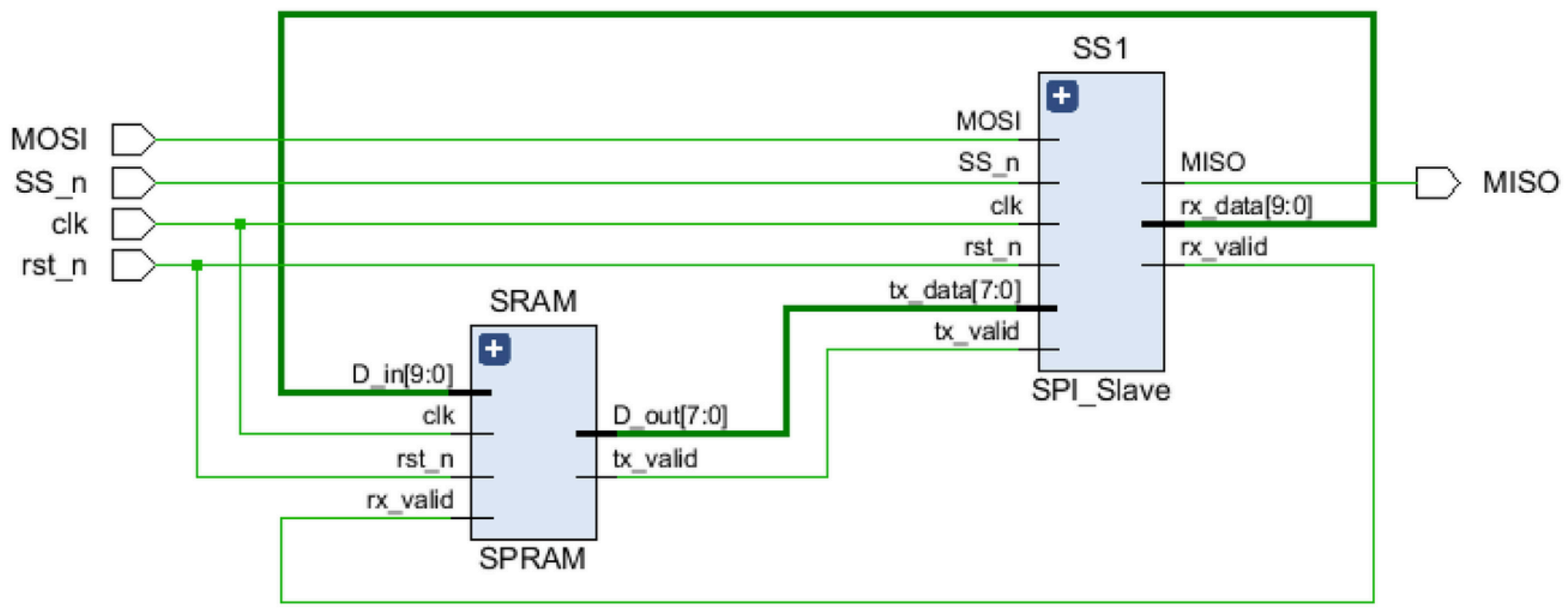


VIVADO

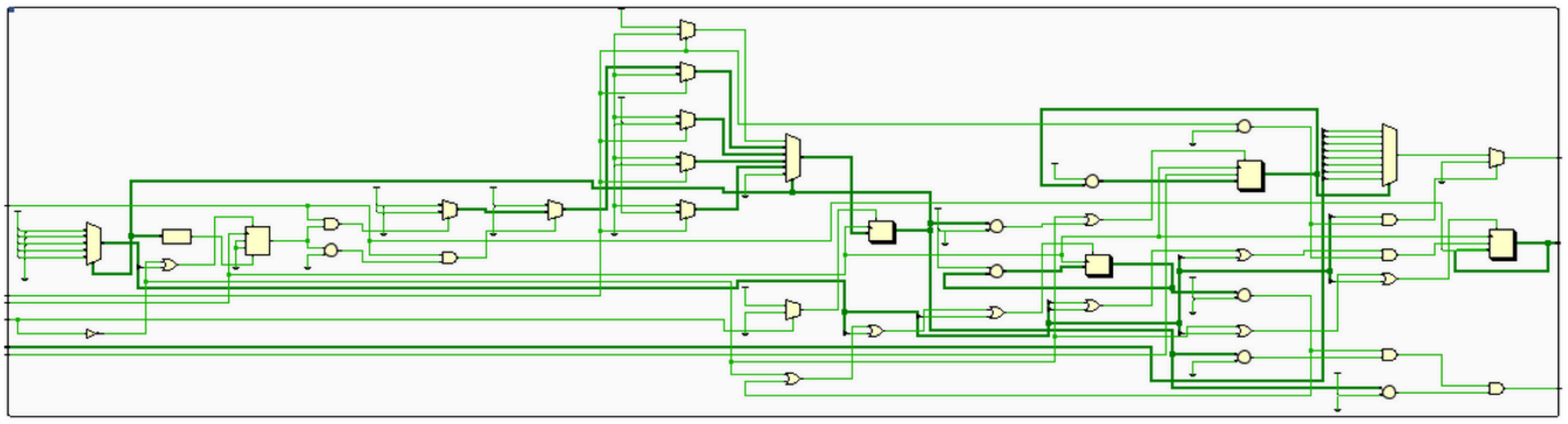
Sequntial Encoding

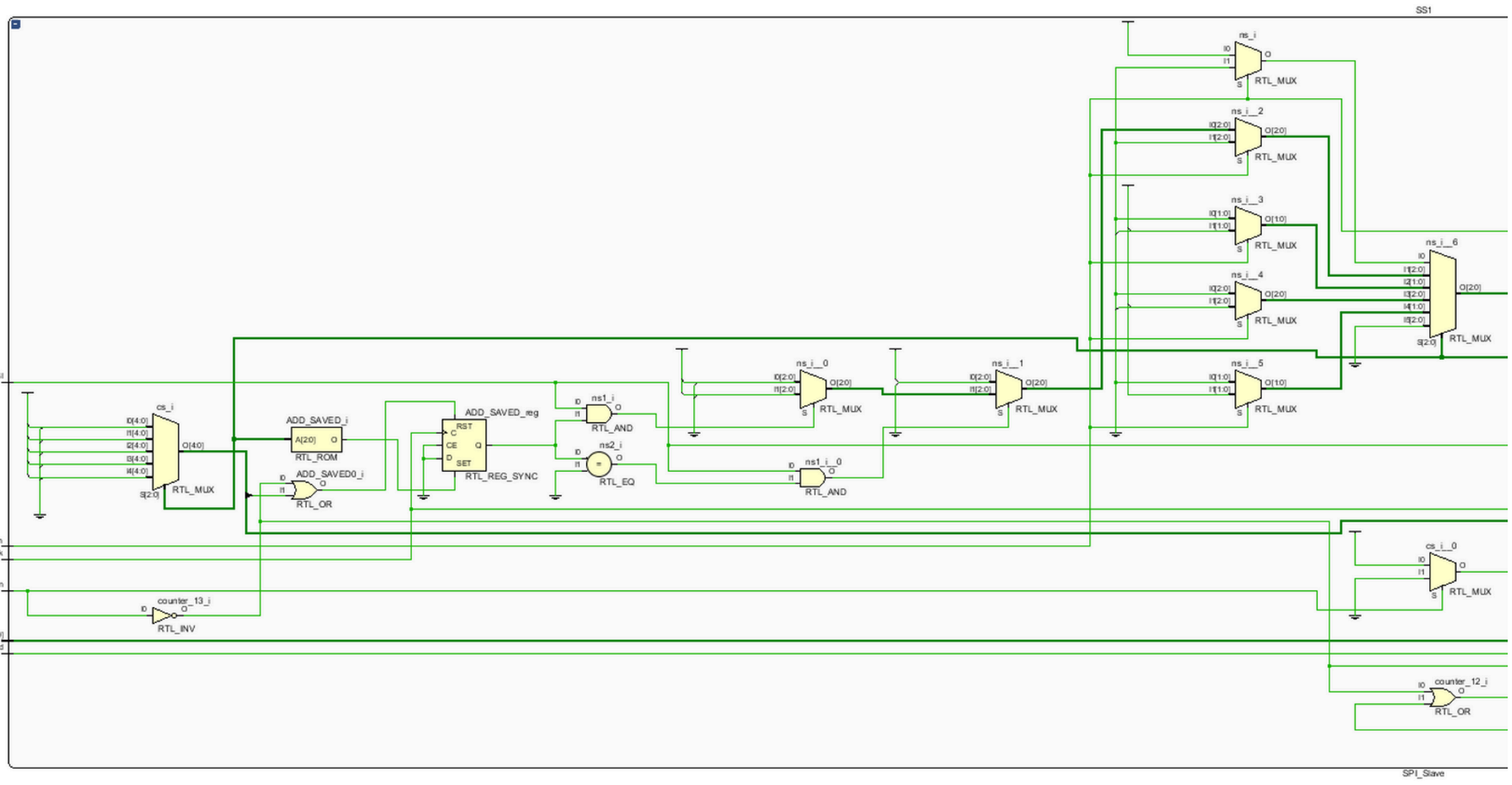
Elaborated Schematic

SPI Wrapper module



SPI Slave module



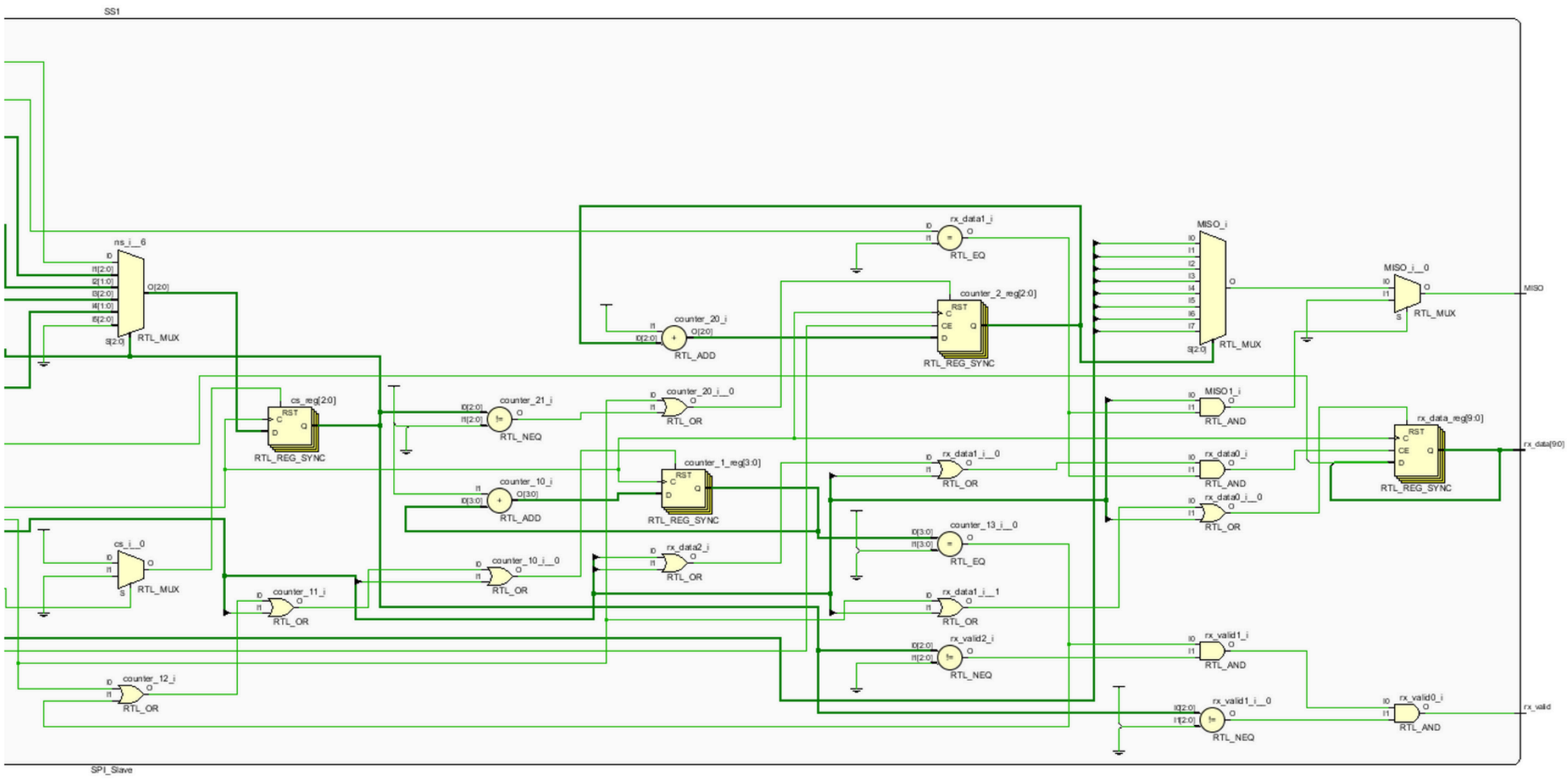


VIVADO

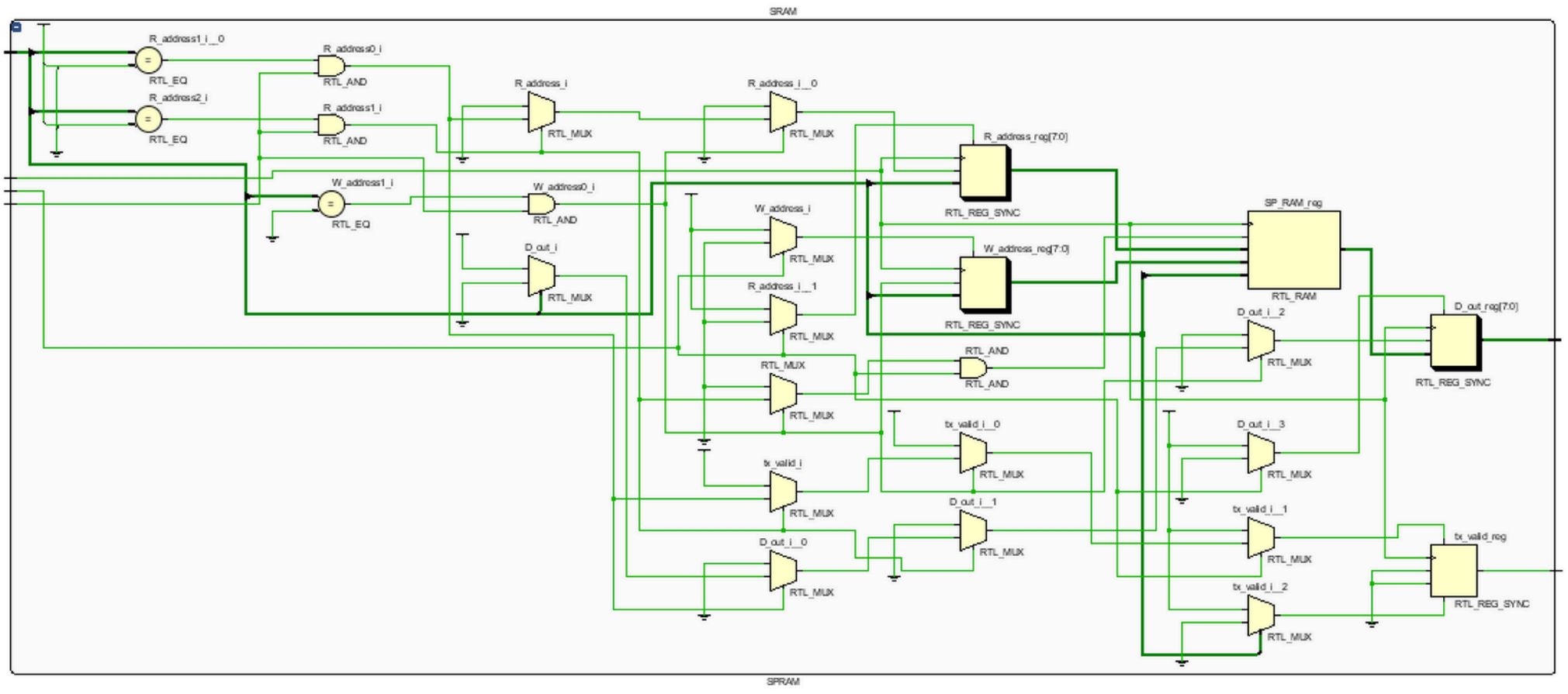
One-Hot Encoding

Elaborated Schematic

SPI Slave module



SPRAM module



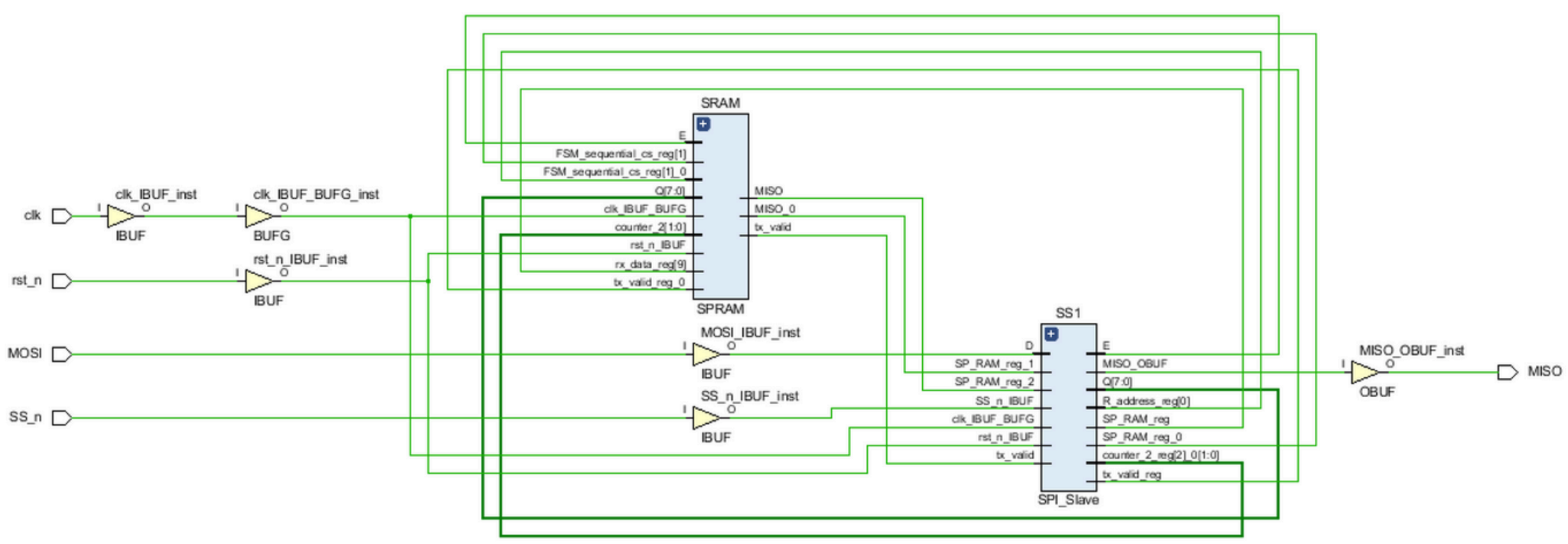
VIVADO

One-Hot Encoding

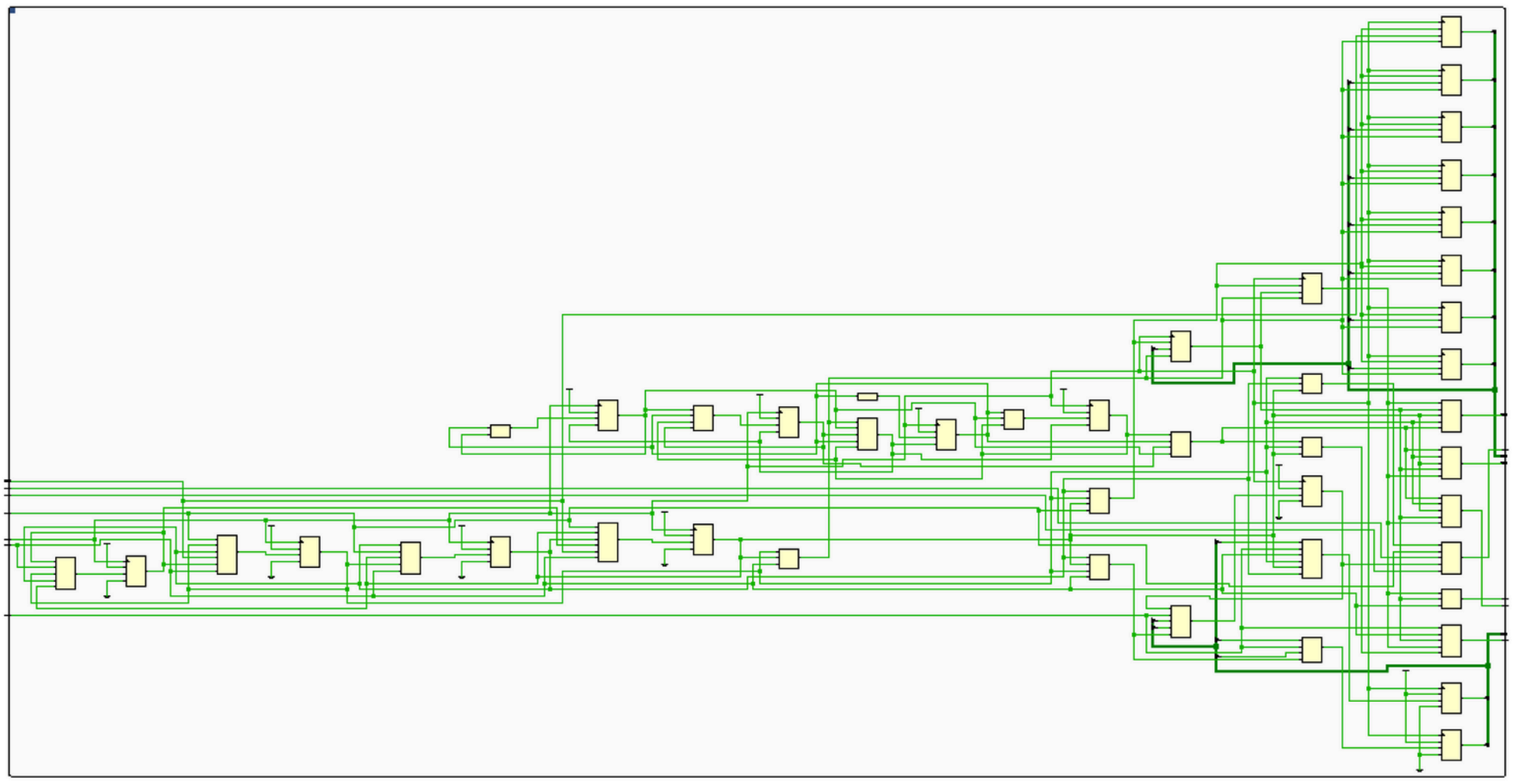
Elaborated Message

Synthesis Schematic

SPI Wrapper module

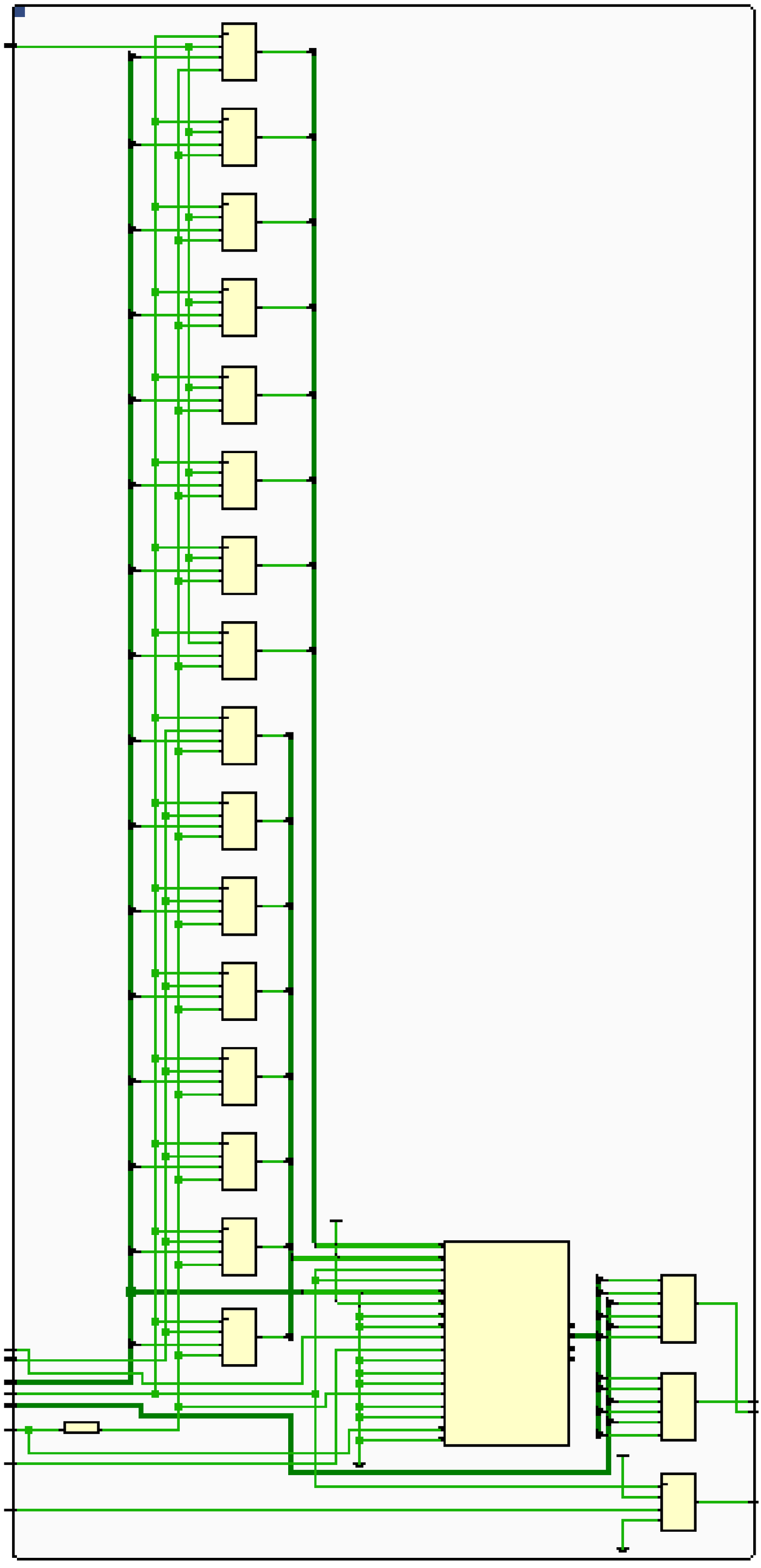


SPI Slave module



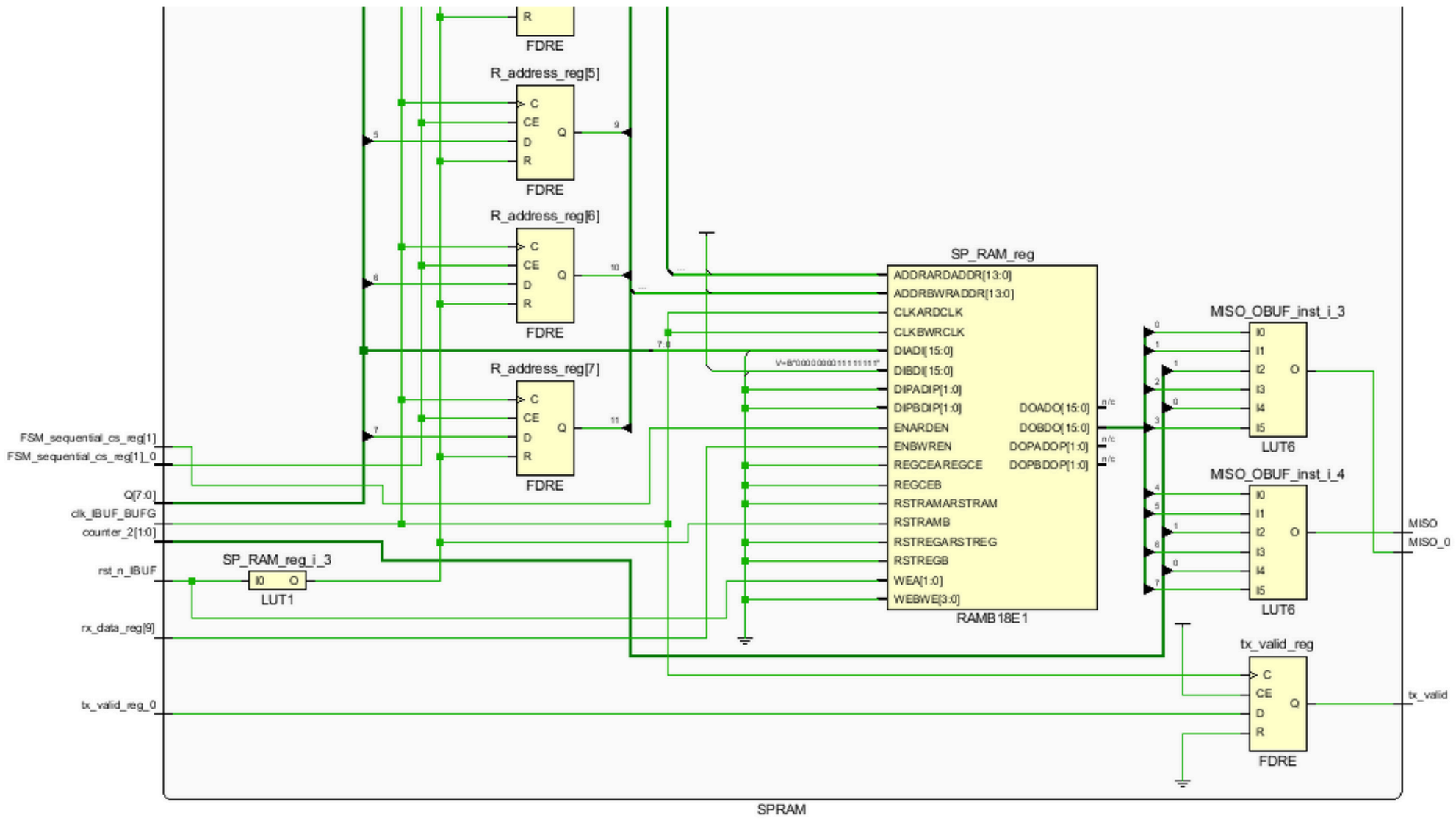
VIVADO

SPRAM module

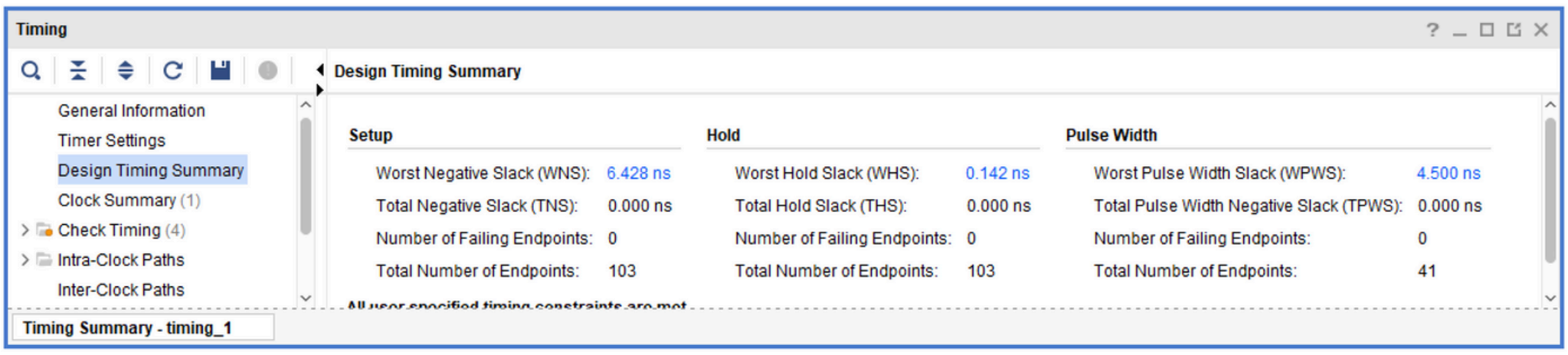


VIVADO

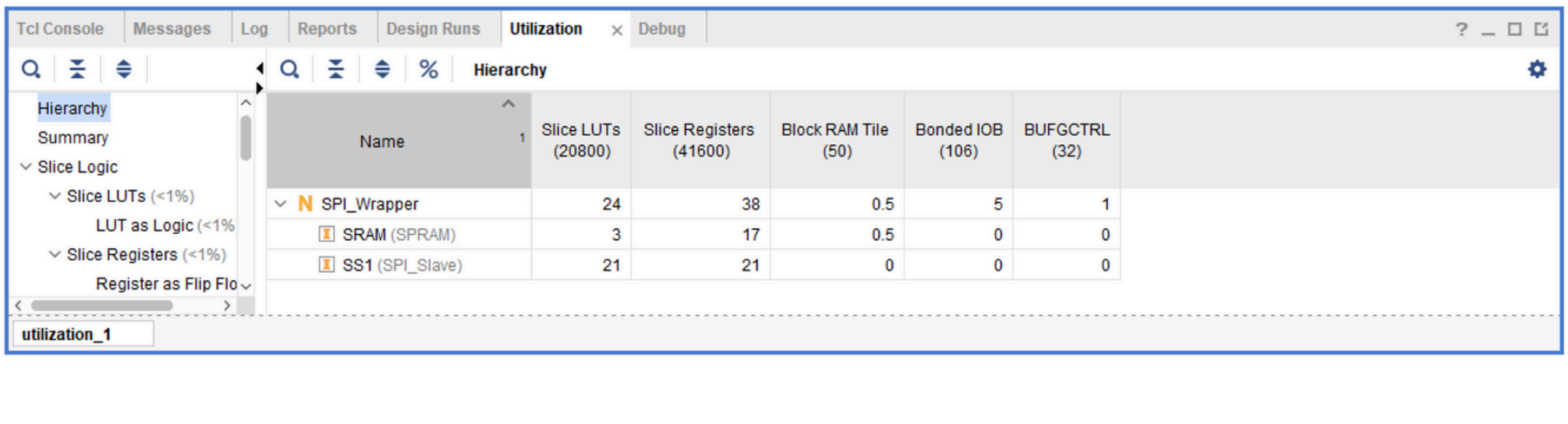
SPRAM module



Synthesis Timing Report

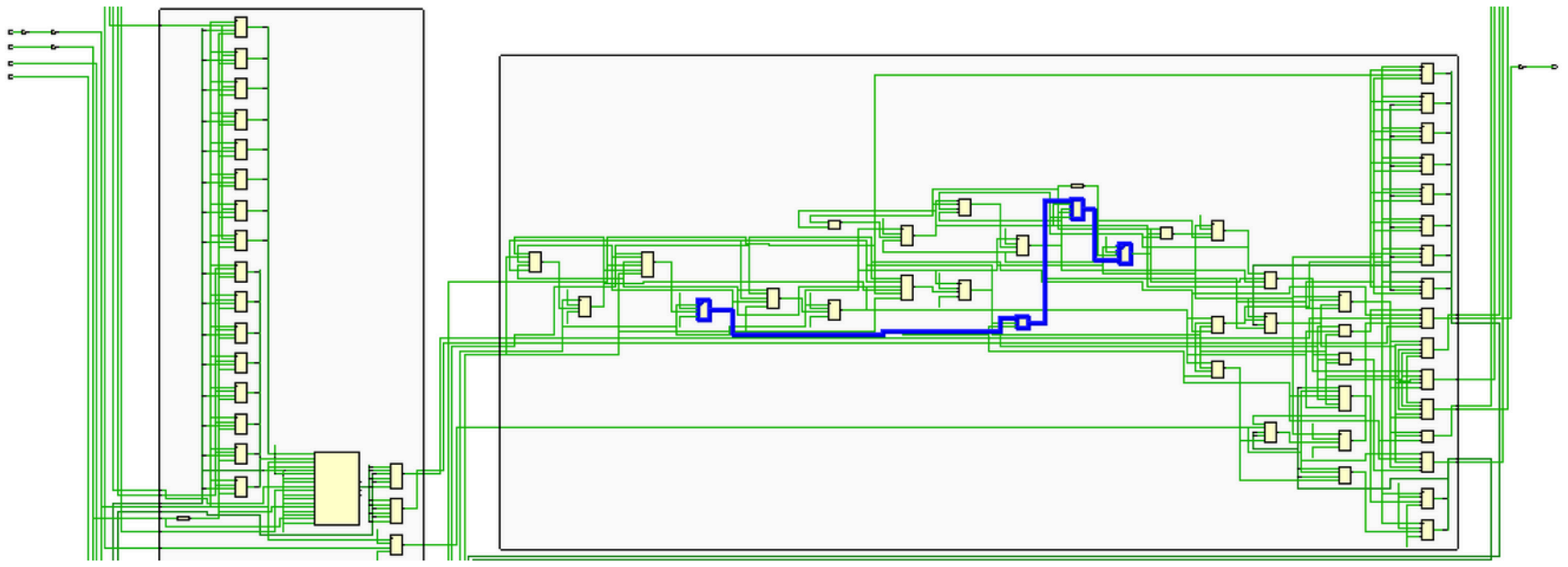


Synthesis Utilization Report



VIVADO

Critical Path



Synthesis Message

Tcl Console | **Messages** | Log | Reports | Design Runs | Utilization | Debug

Warning (1) | Info (44) | Status (44) | Show All

Synthesis (1 warning, 32 infos)

- [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
- [Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI.v:174] (2 more like this)
- [Synth 8-5534] Detected attribute (* fsm_encoding = "sequential" *) [SPI.v:61]
- [Synth 8-6155] done synthesizing module 'SPI_Slave' (1#1) [SPI.v:44] (2 more like this)
- [Device 21-403] Loading part xc7a35ticpg236-1L
- [Project 1-236] Implementation specific constraints were found while reading constraint file [Z:/github/DigtalDesignVerification/Project/SPI/Constraints.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [.Xil/SPI_Wrapper_propImpl.xdc].
  Resolution: To avoid this warning, move constraints listed in ['Undefined'] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.
- [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
- [Synth 8-5544] ROM "ADD_SAVED" won't be mapped to Block RAM because address size (3) smaller than threshold (5) (5 more like this)
- [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'sequential' in module 'SPI_Slave'
- [Synth 8-4480] The timing for the instance i_0/SRAM/SP_RAM_reg (implemented as a block RAM) might be sub-optimal as no optional output register could be merged into the block ram. Providing additional output register may help in improving timing. (1 more like this)
- [Project 1-571] Translating synthesized netlist
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-570] Preparing netlist for logic optimization (1 more like this)
- [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
- [Project 1-111] Unisim Transformation Summary:
  No Unisim elements were transformed. (1 more like this)
- [Common 17-83] Releasing license: Synthesis
- [Constraints 18-5210] No constraint will be written out.
- [Common 17-1381] The checkpoint 'C:/Users/muhammadwael/Vivado/SPI_project/SPI_project.runs/synth_1/SPI_Wrapper.dcp' has been generated.
- [runtcl-4] Executing : report_utilization -file SPI_Wrapper_utilization_synth.rpt -pb SPI_Wrapper_utilization_synth.pb
- [Common 17-206] Exiting Vivado at Tue Aug 5 22:35:43 2025...

Synthesized Design (9 infos)

General Messages (9 infos)

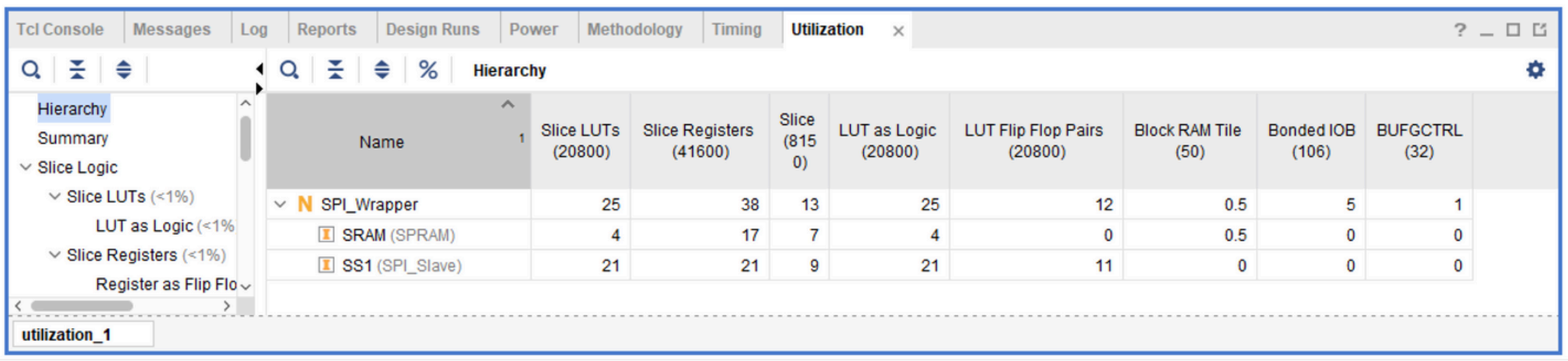
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement

VIVADO

Synthesis Report

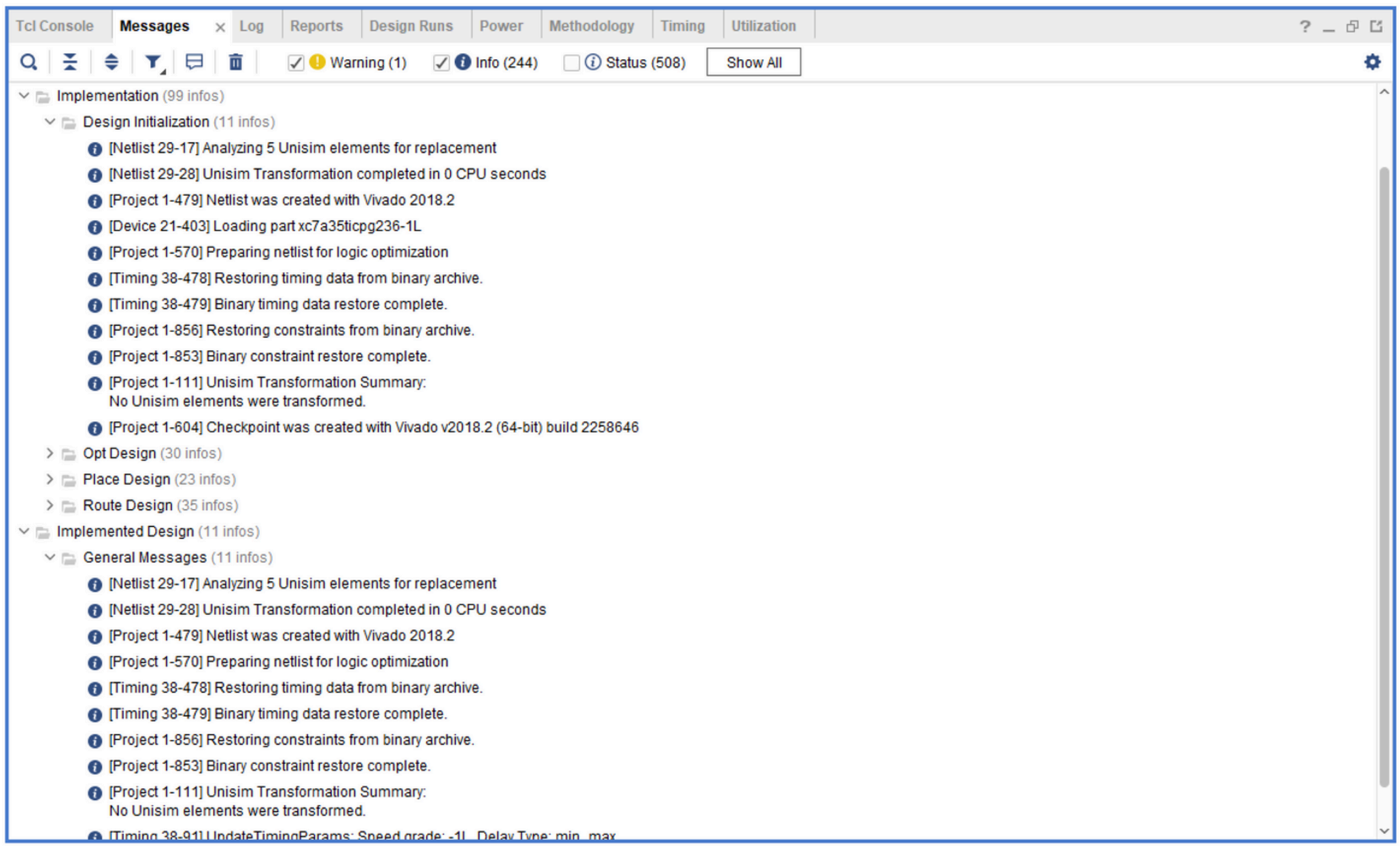
```
INFO: [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
INFO: [Synth 8-5544] ROM "ADD_SAVED" won't be mapped to Block RAM because address size (3) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "tx_valid" won't be mapped to Block RAM because address size (2) smaller than threshold (5)
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |                Previous Encoding
---------------------------------------------------------------------------------------------------
                    IDLE |                              000 |                             000
                 CHK_CMD |                              100 |                             001
                READ_ADD |                              001 |                             011
               READ_DATA |                              010 |                             100
                   WRITE |                              011 |                             010
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'sequential' in module 'SPI_Slave'
```

Implementation Utilization Report



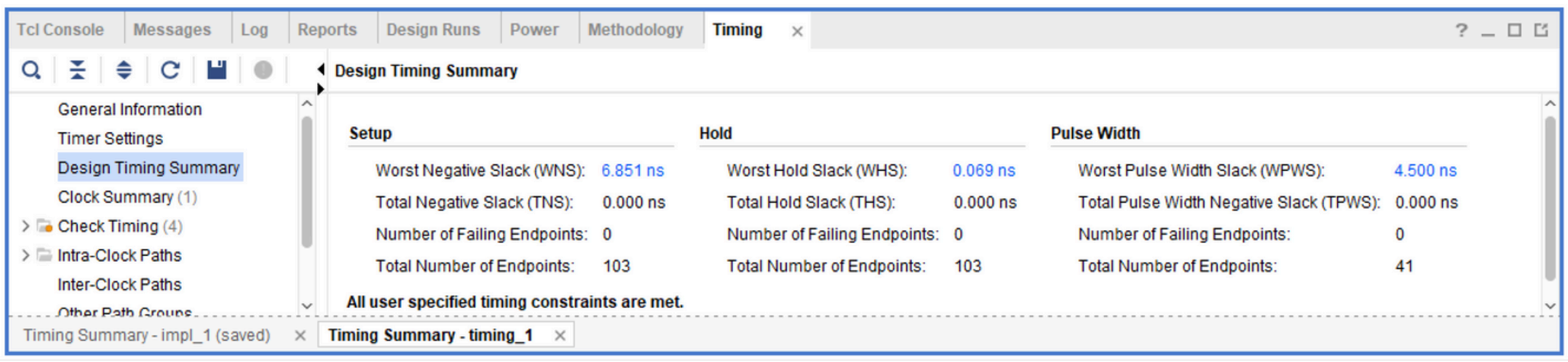
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N SPI_Wrapper	25	38	13	25	12	0.5	5	1
SRAM (SPRAM)	4	17	7	4	0	0.5	0	0
SS1 (SPI_Slave)	21	21	9	21	11	0	0	0

Implementation Message

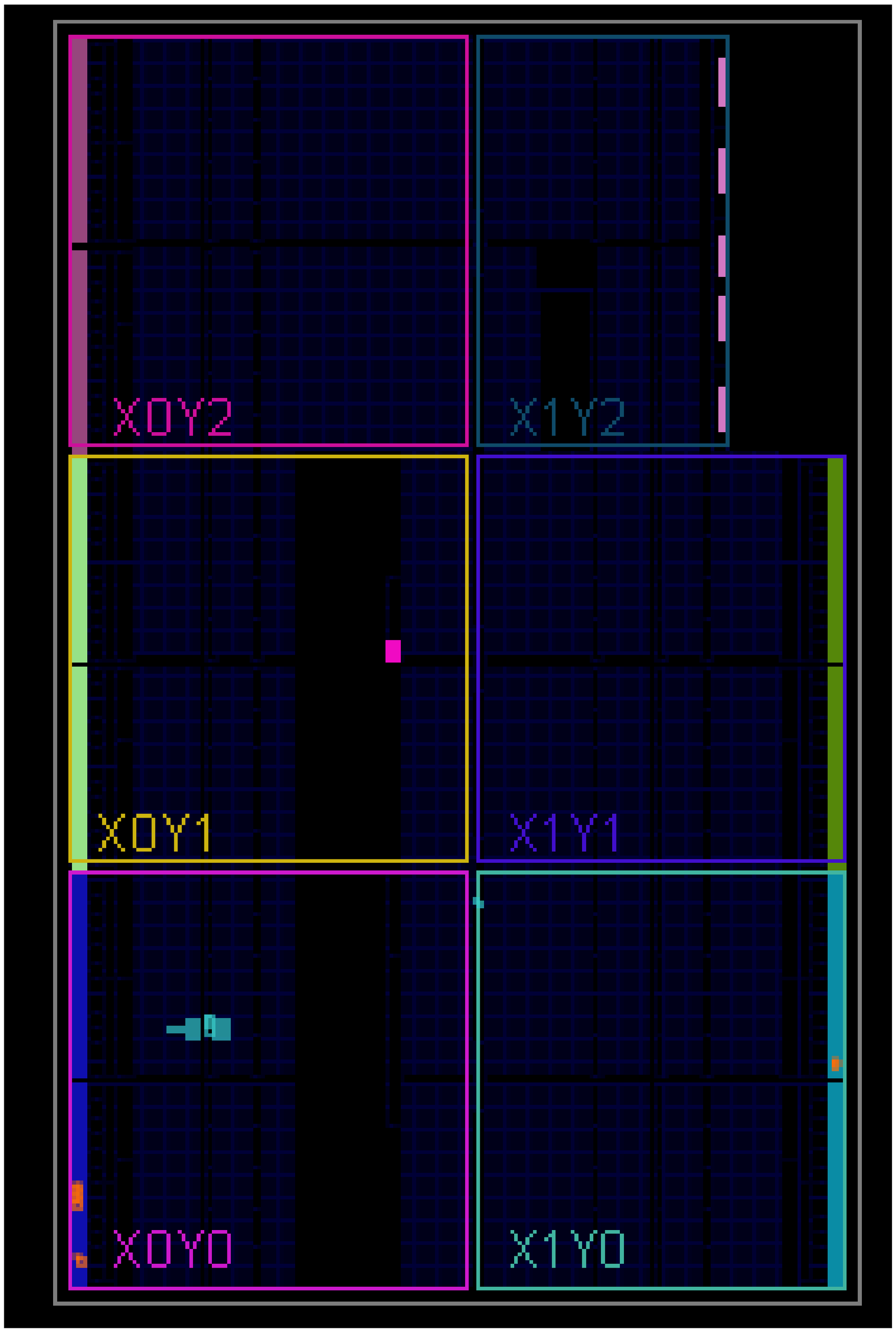


VIVADO

Implementation Timing Report

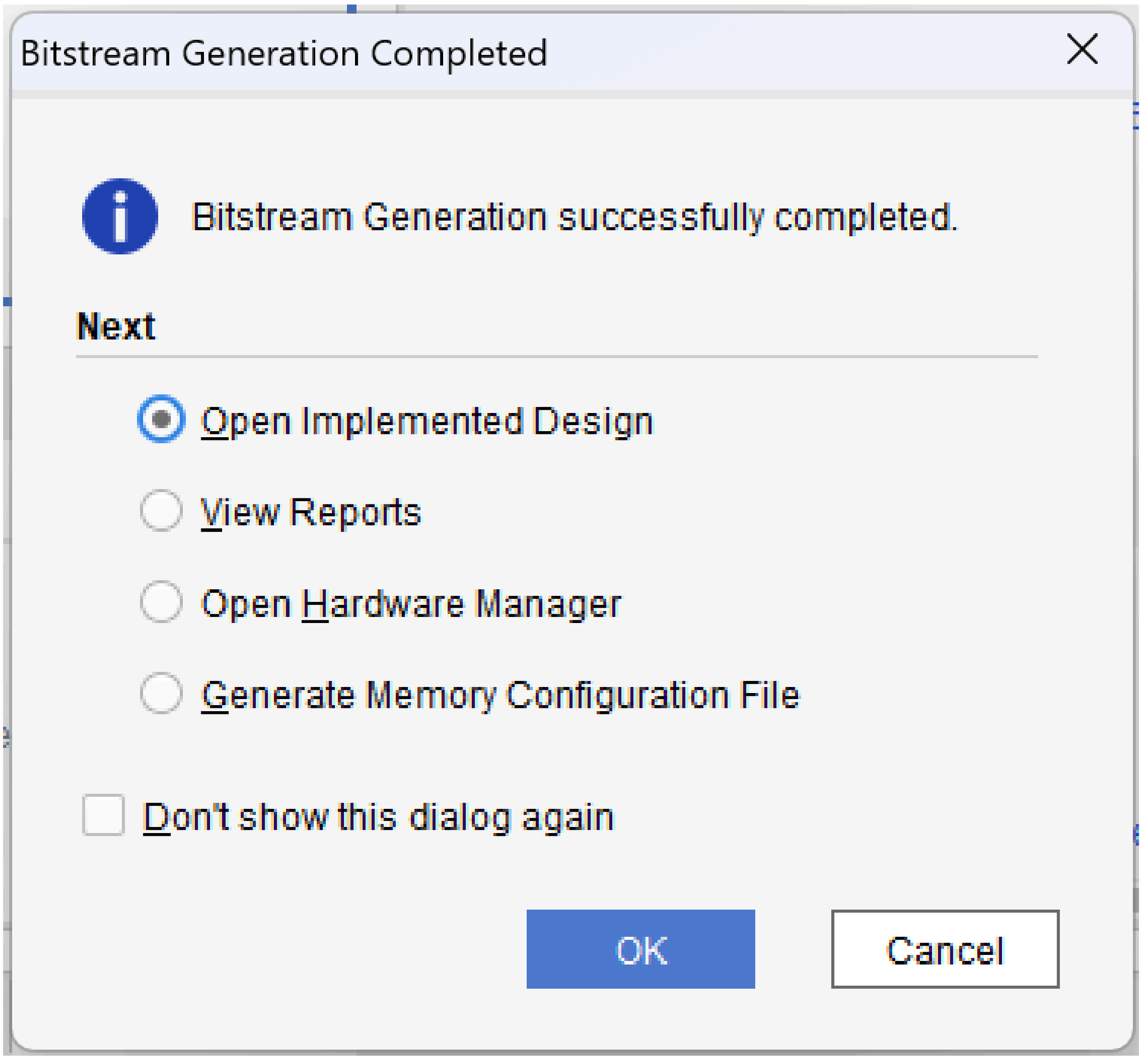


Device



VIVADO

Bit-Stream Generation



FSM Best Encoding

In order to operate at the highest frequency you should use

sequential encoding