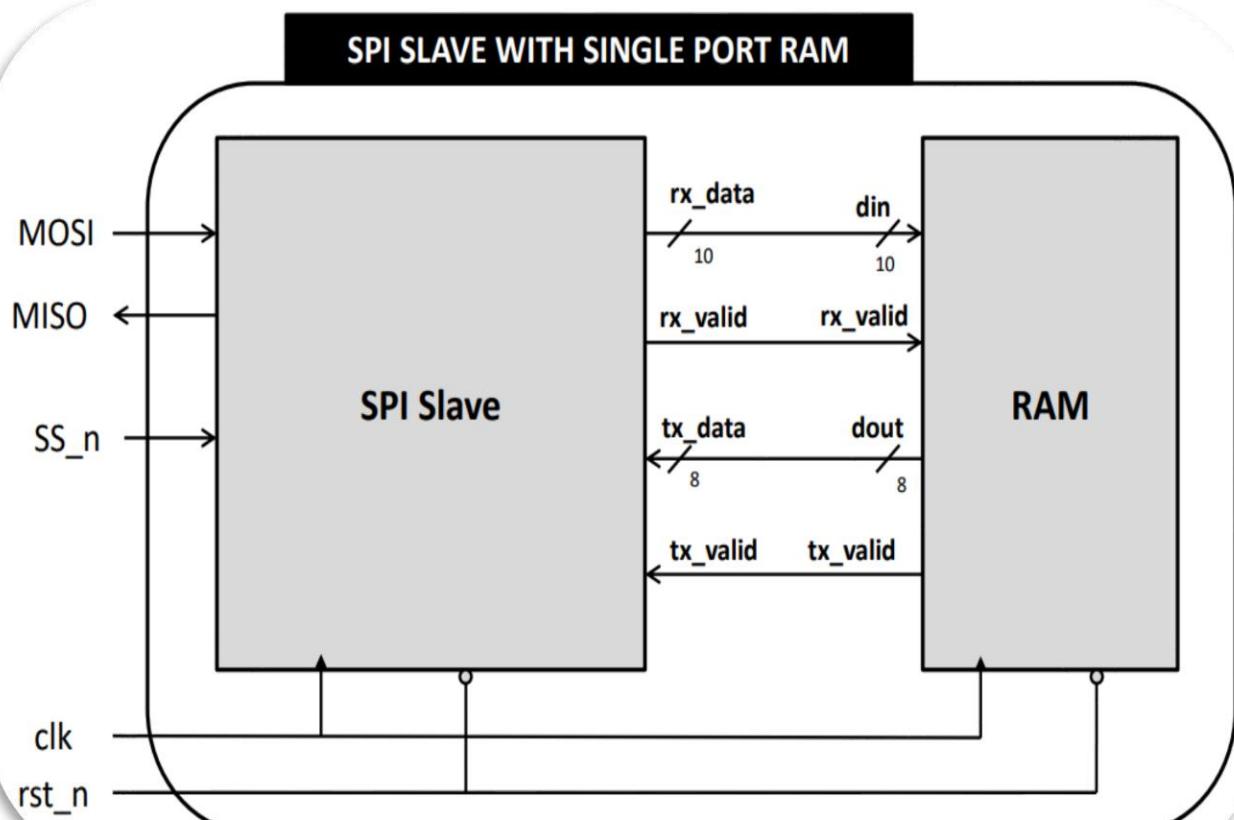


SPI Slave with Single Port RAM



Prepared by: Mohamed Shaban Moussa

Submitted to: Eng. Kareem Waseem

Contact

- Email: mohamedmouse066@gmail.com
- Linkedin: [Press Here](#)

Project Sources: [Press here](#)

SPI Wrapper RTL code

```
④ SPI_Wrapper.v > SPI_Wrapper > MISO
1  module SPI_Wrapper (MOSI,SS_n,clk,rst_n,MISO); /// top module
2  input MOSI,SS_n,clk,rst_n;
3  output MISO;
4  wire [9:0] rx_data;
5  wire [7:0] tx_data;
6  wire tx_valid,rx_valid;
7  SPI SPI_MS(MOSI,SS_n,clk,rst_n,tx_data,tx_valid,rx_data,rx_valid,MISO);
8  Single_Port_RAM MEM(rx_data,rx_valid,clk,rst_n,tx_data,tx_valid);
9
10 endmodule //SPI_Wrapper
```

Single Port RAM RTL code

```
④ Single_Port_RAM.v > Single_Port_RAM
1  module Single_Port_RAM (din,rx_valid,clk,rst_n,dout,tx_valid);
2  parameter MEM_DEPTH = 256;
3  parameter ADDR_SIZE = 8 ;
4  input [9:0] din;
5  input rx_valid,clk,rst_n ;
6  output reg [7:0] dout ;
7  output reg tx_valid;
8  reg [7:0] mem [MEM_DEPTH-1:0]; // memory
9  reg [ADDR_SIZE-1:0] wr_Addr,rd_Addr; //Separated two addresses for write and read operations
10 reg [3:0] tx_valid_counter; // counter to deassert tx_valid after 8 cycles [0 to 7]
11 always @ (posedge clk) begin
12   if (~rst_n) begin
13     dout <= 0;
14     wr_Addr <= 0;
15     rd_Addr <= 0;
16     tx_valid <= 0;
17     tx_valid_counter <= 0;
18   end
19   else if (tx_valid_counter == 7 && tx_valid) begin // Check if 8 clock cycles have been reached to deassert tx_valid.
20     tx_valid_counter <= 0;
21     tx_valid <= 0;
22   end
23   else if (tx_valid) tx_valid_counter <= tx_valid_counter + 1;
24   else if (rx_valid) begin
25     case(din[9:8])
26       2'b00: wr_Addr <= din[7:0];
27       2'b01: mem[wr_Addr] <= din[7:0];
28       2'b10: rd_Addr <= din[7:0];
29       2'b11: begin // forced to wait rx_data after 10 clk
30         dout <= mem[rd_Addr]; // tx_data
31         tx_valid <= 1;
32       end
33     endcase
34   end
35 end
36 endmodule
```

SPI Protocol RTL code

States Definitions And State Memory Block

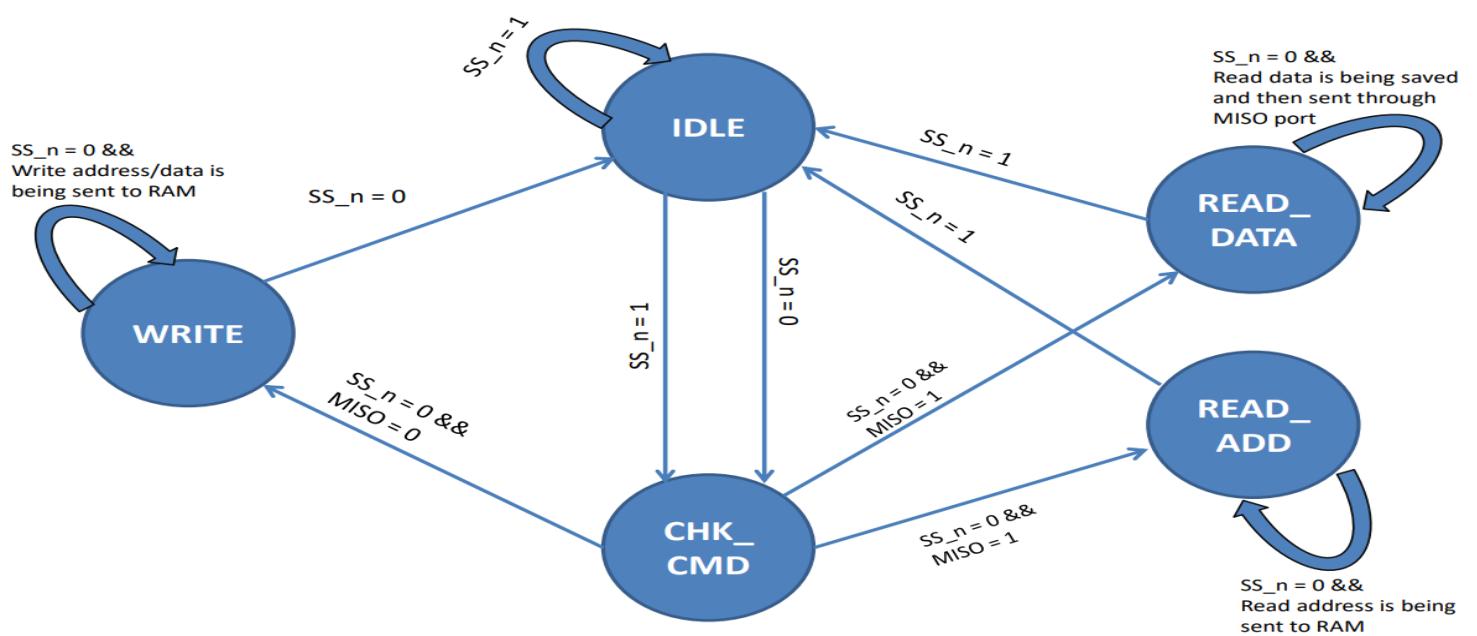
SPI.v > SPI

```
1  module SPI (MOSI,SS_n,clk,rst_n,tx_data,tx_valid,rx_data,rx_valid,MISO);
2  // Declaration of FSM states
3  parameter IDLE =0;
4  parameter CHK_CMD=1;
5  parameter WRITE =2;
6  parameter READ_ADD =3 ;
7  parameter READ_DATA =4;
8  (*fsm_encoding="gray")reg [2:0] ns,cs;
9  reg rd_data; //// wire control read state IF 1 (DATA) IF 0 (ADD)
10 input MOSI,SS_n,tx_valid,clk,rst_n;
11 input [7:0] tx_data;
12 output reg rx_valid;
13 output MISO;
14 output reg [9:0] rx_data;
15 reg [3:0] clk_count; // Controls the number of clock cycles needed for rx_data and MISO
16 //////// state memory ///////
17 always @(posedge clk) begin
18   if(~rst_n) begin
19     cs<=IDLE;
20   end
21   else cs<=ns;
22 end
```

- **rd_data:** Control signal that determines if reading data (1) or address (0)
- **clk_count:** 4-bit counter to track clock cycles for rx_data and MISO operations

Next State Logic Block

```
////////// Next State Logic //////////
always @(*) begin
    case(cs)
        IDLE:begin  if(!SS_n) ns=CHK_CMD;
        | | | | | | | else ns=IDLE;
    end
        CHK_CMD: begin
            if(SS_n) ns=IDLE;
            else if(!MOSI) ns=WRITE;
            else if(MOSI) begin
                if(!rd_data)
                    ns=READ_ADD;
                else ns=READ_DATA;
            end
        end
        end
        WRITE:begin if(SS_n) ns=IDLE;
        | | | | | | | else ns=WRITE;
    end
        READ_ADD:begin if(SS_n) ns=IDLE;
            else begin
                ns=READ_ADD;
            end
        end
        end
        READ_DATA:begin if(SS_n) ns=IDLE;
            else begin
                ns=READ_DATA;
            end
        end
    end
endcase
end
```



Output Logic Block

```
55 /////////////// Output Logic ///////////////////
56 always @(posedge clk) begin
57     if(!rst_n) begin
58         rx_data<=0;
59         rx_valid<=0;
60         clk_count<=0;
61         rd_data<=0;
62     end
63     else begin
64         if(~SS_n) begin
65             rx_valid<=0;
66             if(cs==WRITE || cs==READ_ADD) begin
67                 rx_data<={rx_data[8:0],MOSI}; // Serial to parallel Conversion (insert before checking)
68                 if(clk_count==9) begin
69                     rx_valid<=1; // After 10 clock cycles rx_valid should be asserted to communicate with RAM
70                     clk_count<=0;
71                     if(cs==READ_ADD) rd_data<=1;//rd_data must be high to trigger next transition to the READ_DATA state
72                 end
73                 else begin
74                     clk_count<=clk_count+1;
75                 end
76             end
77             else if (cs==READ_DATA) begin
78                 if (!tx_valid) begin //if true Send rx_data (with dummy bits) and assert rx_valid to receive tx_valid and tx_data (10 clk)
79                     rx_data<={rx_data[8:0],MOSI}; // Serial to parallel Conversion
80                     if(clk_count==9) begin
81                         rx_valid<=1; // After 10 clock cycles rx_valid should be asserted to communicate with RAM
82                         clk_count<=0;
83                     end
84                     else clk_count<=clk_count+1;
85                 end
86                 else
87                     begin /// if tx_valid HIGH
88                     if(clk_count>7) begin // When clock count reaches 8 reset it to zero in the next cycle
89                         clk_count<=0;
90                         rd_data<=0;
91                     end
92                     else clk_count<=clk_count+1;
93                 end
94             end
95             end
96             else rx_valid<=0;
97         end
98         end
99         assign MISO=(tx_valid&&cs==READ_DATA)?tx_data[8-clk_count]:0;
100        //Due to the fetch cycle between RAM and SPI, tx_valid is asserted one cycle after rx_valid goes high
101        // tx_data[8 - clk_count] ensures transmission of tx7 to tx0, and stops when tx_valid goes low
102    endmodule //SPI
```

SPI Wrapper Testbench code

Clock Generation And Apply rst_n

```
④ SPI_Wrapper_Tb.v > ⑤ SPI_Wrapper_Tb
1  module SPI_Wrapper_Tb ();
2  reg MOSI,SS_n,clk,rst_n;
3  reg [7:0] addr;
4  reg [7:0] data;
5  wire MISO;
6  SPI_Wrapper TEST(MOSI,SS_n,clk,rst_n,MISO);
7  initial begin
8    clk=0;
9    forever begin
10      #1 clk=~clk;
11    end
12  end
13  integer i;
14  initial begin
15    {rst_n,SS_n,MOSI,addr,data}=0;
16    @(negedge clk); // to deassert all output signals
17    rst_n=1;
```

Briefly describe the code flow and what data and addr do

Phase 1: Send Address (Write Address Command)

1. SS_n = 0 → Start communication.
2. MOSI = 0, wait 2 falling edges of clk → To reach WRITE state (1 for CHK_CMD, 1 for WRITE).
3. Wait 2 more falling edges → Set rx_data[9:8] = 2'b00 → This is "Write Address" command.
4. Loop 8 cycles:
 - Send 8-bit address bit-by-bit on MOSI.
 - Bits are stored in addr[7:0].
5. SS_n = 1 → End communication. Wait 1 clock cycle → Slave fetches addr into internal memory.

Phase 2: Send Data (Write Data Command)

1. SS_n = 0, MOSI = 0 → Start communication again.
2. Wait 2 clock cycles → To reach WRITE state.
3. Send rx_data[9]=0 and rx_data[8]=1 → Command becomes 2'b01, which is "Write Data".
4. Loop 8 cycles:
 - Send 8-bit data on MOSI.
 - Bits are stored in data[7:0].
5. SS_n = 1 → End communication, Wait 1 clock cycle → Slave stores data into memory at addr.

Phase 3: Read by Address (Read Address Command)

1. **SS_n = 0, MOSI = 1** → Start communication again.
2. Wait 2 clock cycles → To reach READ_ADDR state.
3. Send rx_data[9]=1 and rx_data[8]=0 → Command is 2'b10, "Read Address".
4. Loop 8 cycles:
 - Send back stored addr[7:0] on MOSI.
 - Slave receives it and uses it as read address.
5. **SS_n = 1** → End communication.
6. Wait 1 clock → Slave fetches data from memory at the given address.

Phase 4: Receive Data (Read Data Command)

1. **SS_n = 0, MOSI = 1**, wait 2 clocks → Enter READ_DATA state.
2. Send rx_data[9:8] = 2'b11 → Read Data command.
3. Send 8 dummy bits on MOSI just to allow clock edges.
4. Wait 1 clock → Slave loads tx_data with stored data.
5. Loop 8 cycles:
 - Master reads MISO bit-by-bit.
 - **Each MISO bit is compared to data[7:0].**
 - **If mismatch → Display error and stop simulation.**
6. **SS_n = 1** → End communication.
7. Wait 1 clock.
8. After loop ends → \$stop is called to finish simulation.

The code is repeated 2000 times to ensure thorough testing of all address and data combinations through multiple SPI transactions.

Self-Checking Testbench (Repeat 2000 times)

```
18 repeat(2000) begin
19   SS_n=0;
20   MOSI=0;
21   repeat(2) @(negedge clk); /// to reach WRITE state ( 1 for CHK_CMD and 1 for WRITE )
22   repeat(2) @(negedge clk); /// to make rx_data[9:8] = 2'b00
23   for(i=0;i<8;i=i+1) begin
24     MOSI=$random;
25     addr[7-i]=MOSI; /// to store addr
26     @(negedge clk);
27   end
28   SS_n=1; /// to end Comm
29   @(negedge clk); /// to fetch from RAM and SPI
30   SS_n=0;/// to start Comm
31   MOSI=0;
32   repeat(2) @(negedge clk); /// to reach WRITE state ( 1 for CHK_CMD and 1 for WRITE )
33   MOSI=0; @(negedge clk) ; // rx_data[9]=0
34   MOSI=1; @(negedge clk) ; // rx_data[8]=1
35   for(i=0;i<8;i=i+1) begin
36     MOSI=$random;
37     data[7-i]=MOSI;
38     @(negedge clk);
39   end
40   SS_n=1; // to end comm
41   @(negedge clk);
42   SS_n=0; // to start comm
43   MOSI=1; // towards READ add
44   repeat(2) @(negedge clk); // to reach READ_ADD
45   @(negedge clk) // rx_data[9]=1;
46   MOSI=0; @(negedge clk); // rx_data[8]=0; // 10
47   for(i=0;i<8;i=i+1) begin
48     MOSI=addr[7-i];
49     @(negedge clk); // make rd_add = ff
50   end
51   SS_n=1;
52   @(negedge clk); // fetch memory
53   SS_n=0;
54   MOSI=1; repeat(2) @(negedge clk); //reach read data
55   repeat(2) @(negedge clk); // rx_data[9:8]=2'b11
56   repeat(8) begin
57     MOSI=$random; // dumy inputs
58     @(negedge clk);
59   end
60   @(negedge clk) // to fetch from memory
61   for(i=0;i<8;i=i+1) begin // to convert tx_data to serial MISO
62     if(MISO!=data[7-i]) begin
63       $display("Error In SPI Master Slave");
64       $stop;
65     end
66     @(negedge clk);
67   end
68   SS_n=1; // to end comm
69   @(negedge clk);
70   end // tx_valid should be converted to zero
71   $stop;
72 end
73 endmodule //SPI_Wrapper_Tb
```

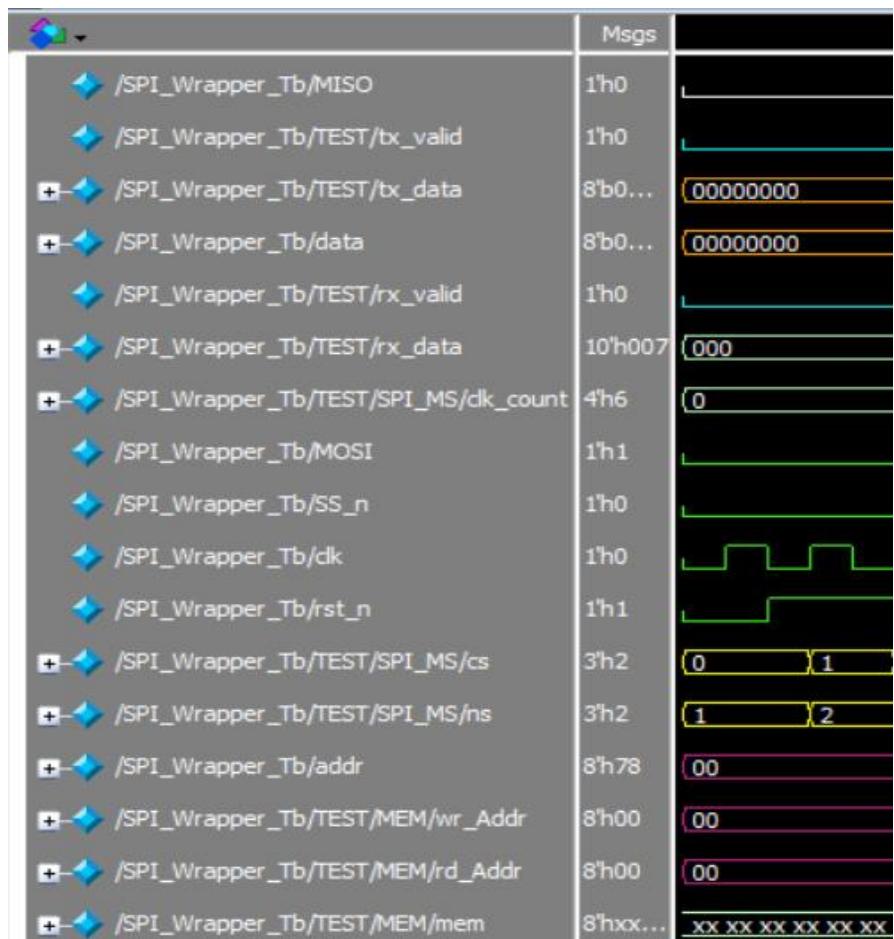
SPI Wrapper do File

```

≡ SPI_Wrapper.do
1   vlib work
2   vlog SPI_Wrapper.v SPI_Wrapper_Tb.v
3   vsim -voptargs=+acc work.SPI_Wrapper_Tb
4   add wave -color white -height 27 sim:/SPI_Wrapper_Tb/MISO
5   add wave -color cyan -height 27 sim:/SPI_Wrapper_Tb/TEST/tx_valid
6   add wave -radix binary -color orange -height 27 sim:/SPI_Wrapper_Tb/TEST/tx_data
7   add wave -radix binary -color orange -height 27 sim:/SPI_Wrapper_Tb/data
8   add wave -color cyan -height 27 sim:/SPI_Wrapper_Tb/TEST/rx_valid
9   add wave -height 27 sim:/SPI_Wrapper_Tb/TEST/rx_data
10  add wave -height 27 sim:/SPI_Wrapper_Tb/TEST/SPI_MS/clk_count
11  add wave -height 27 sim:/SPI_Wrapper_Tb/MOSI
12  add wave -height 27 sim:/SPI_Wrapper_Tb/SS_n
13  add wave -height 27 sim:/SPI_Wrapper_Tb/clk
14  add wave -height 27 sim:/SPI_Wrapper_Tb/rst_n
15  add wave -color yellow -height 27 sim:/SPI_Wrapper_Tb/TEST/SPI_MS/cs
16  add wave -color yellow -height 27 sim:/SPI_Wrapper_Tb/TEST/SPI_MS/ns
17  add wave -color violetred -height 27 sim:/SPI_Wrapper_Tb/addr
18  add wave -color violetred -height 27 sim:/SPI_Wrapper_Tb/TEST/MEM/wr_Addr
19  add wave -color violetred -height 27 sim:/SPI_Wrapper_Tb/TEST/MEM/rd_Addr
20  add wave -height 27 sim:/SPI_Wrapper_Tb/TEST/MEM/mem
21  run -all

```

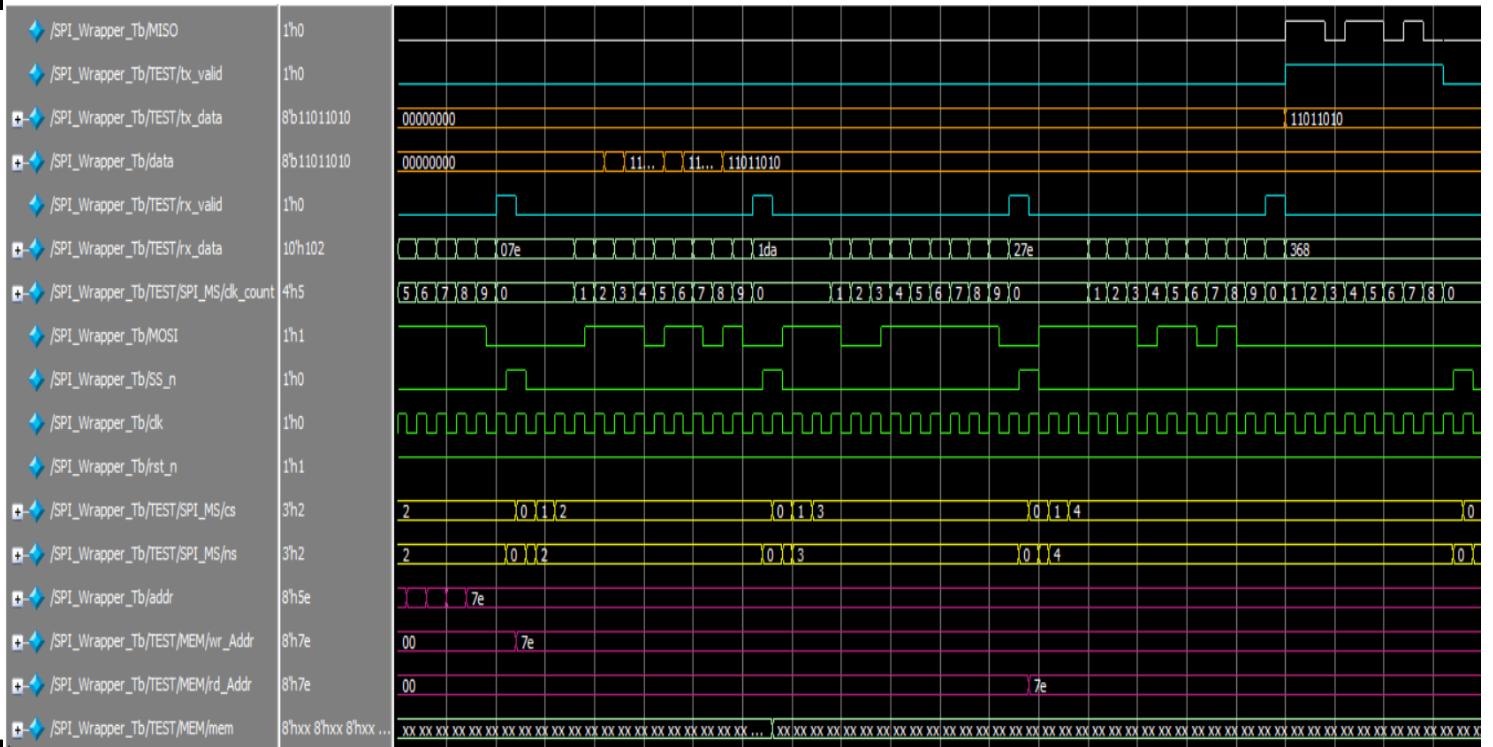
Signal order	Signal Color
MISO	WHITE
tx_valid	cyan
tx_data	orange
data	orange
rx_valid	cyan
rx_data	Normal
clk_count	Normal
MOSI	Normal
SS_n	Normal
clk	Normal
rst_n	Normal
cs	yellow
ns	yellow
addr	violet
wr_Addr	violet
rd_Addr	violet
mem	Normal



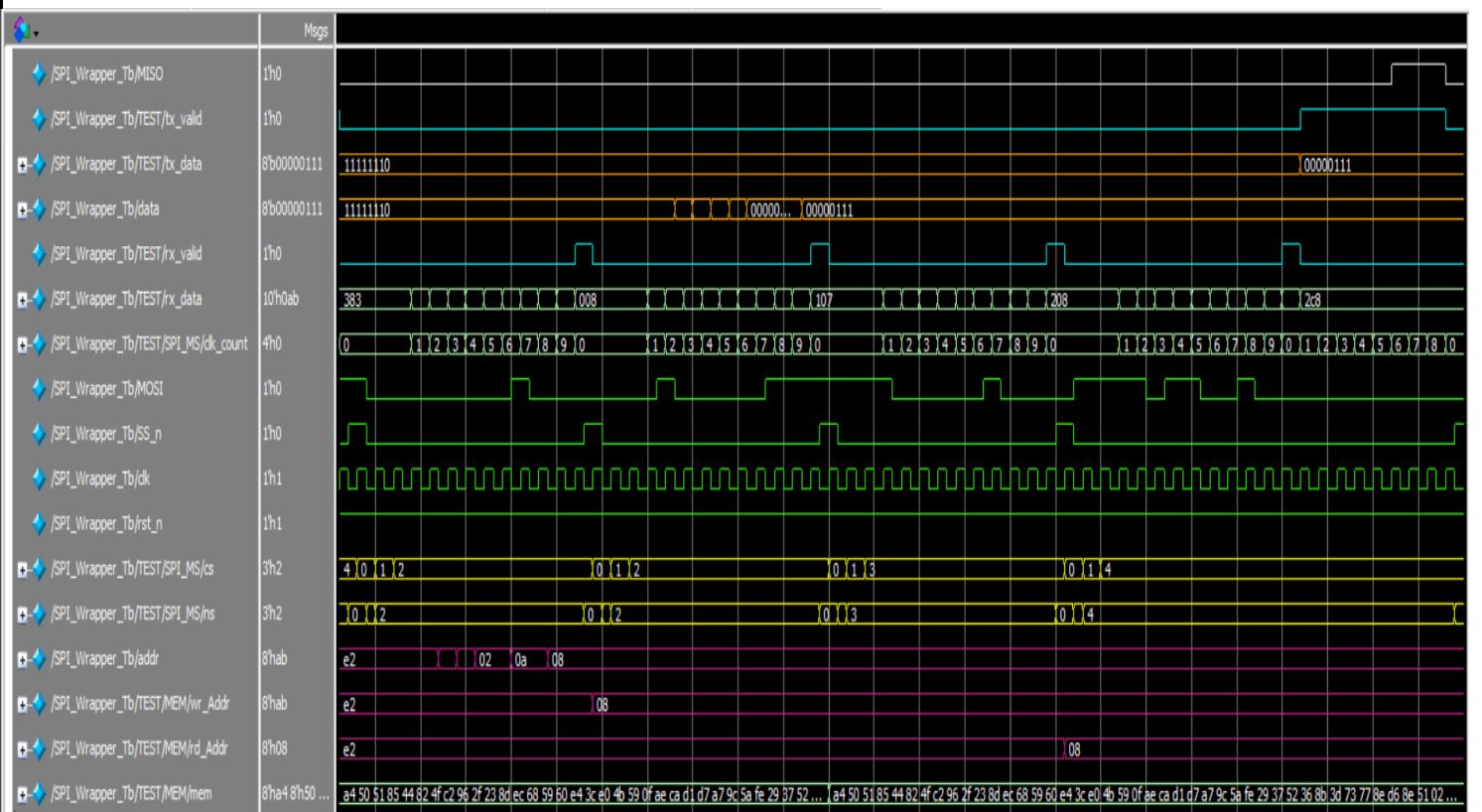
SPI Wrapper Wave Forms

Note: I implemented a tx_valid_counter alongside a single-port RAM to ensure that tx_valid is asserted only for 8 clock cycles during transmission as we see

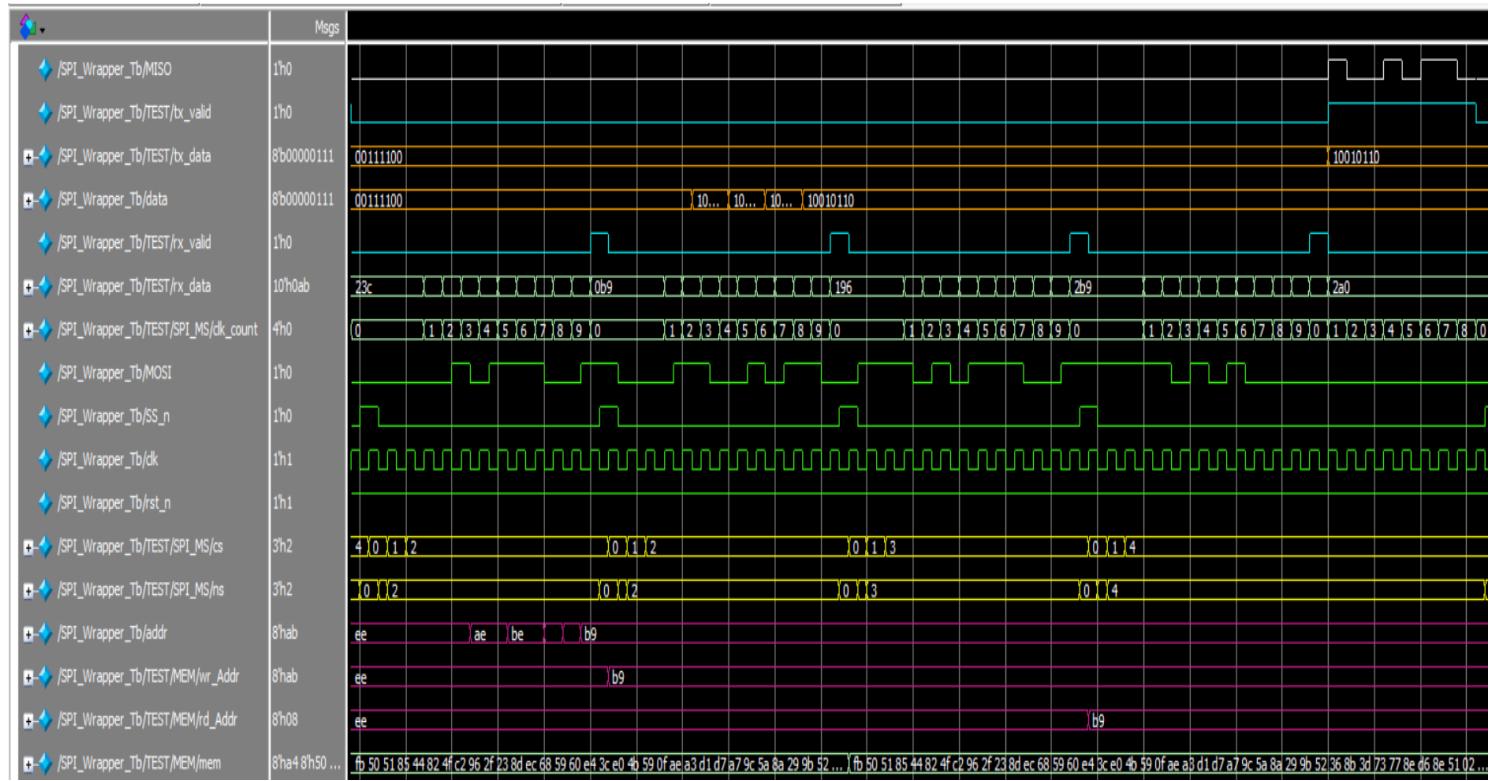
First Check



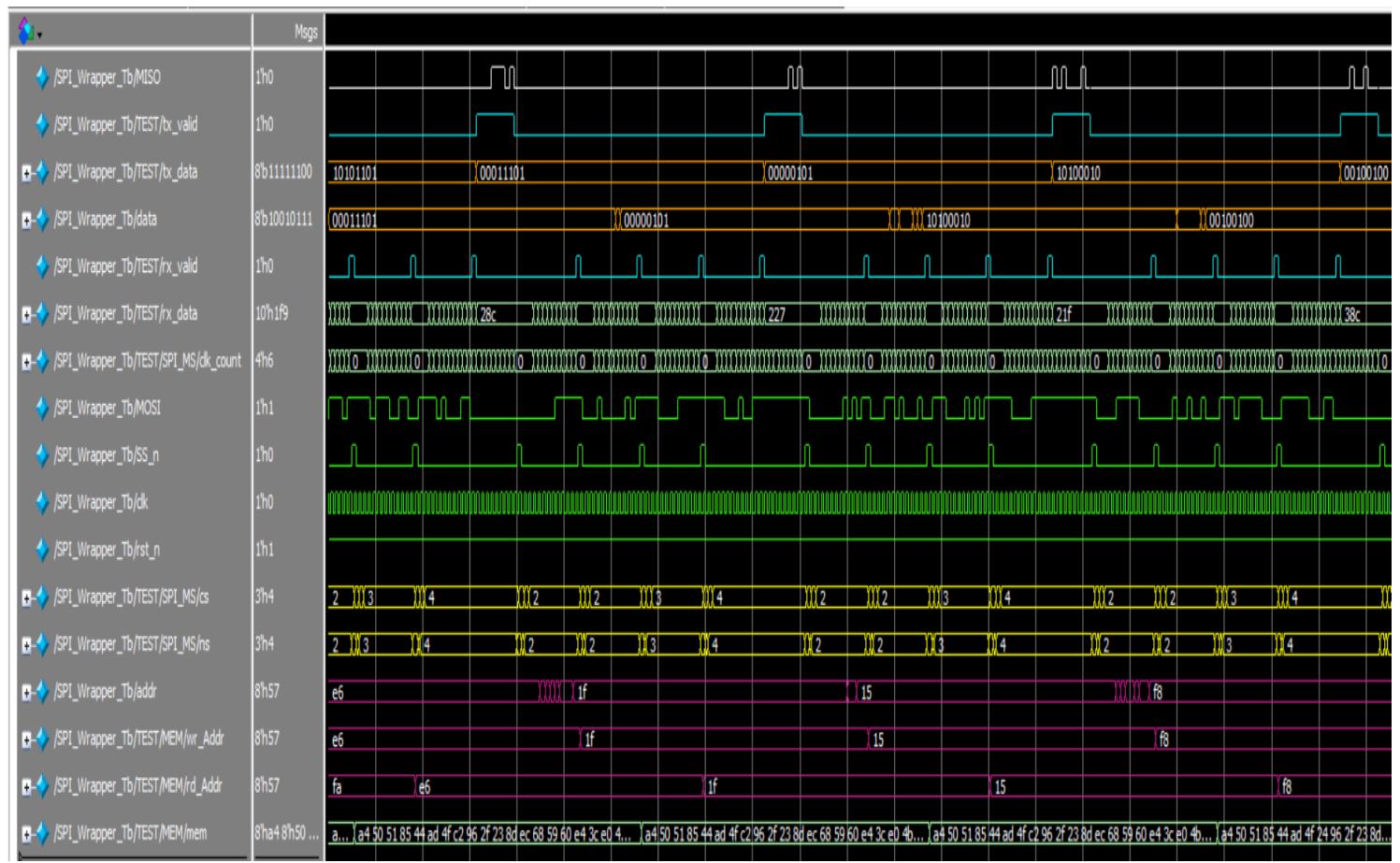
Second Check



Third Check



Multiple Checks



Memory Contents After 2000 Operation

```
sim:/SPI_Wrapper_Tb/TEST/MEM/mem @ 243962 ns
255 : 8'h44 8'h50 8'h51 8'h85 8'h44 8'had 8'h4f 8'h24 8'h73 8'h2f 8'h23 8'h8d 8'hec 8'h68 8'h59 8'h60
239 : 8'he4 8'h3c 8'he0 8'h4b 8'h59 8'h0f 8'hae 8'hca 8'hd1 8'h40 8'ha7 8'h9c 8'h5a 8'hfe 8'h29 8'h37
223 : 8'h52 8'h36 8'h8b 8'h4c 8'h73 8'h77 8'h8e 8'hd6 8'h8e 8'h51 8'h43 8'hba 8'h3e 8'h71 8'he4 8'h08
207 : 8'h3b 8'heb 8'h8e 8'h73 8'h39 8'h5b 8'h92 8'h0b 8'ha5 8'hfd 8'h75 8'h80 8'hfe 8'h69 8'h68 8'hed
191 : 8'h3f 8'h89 8'h12 8'h94 8'h65 8'h23 8'h96 8'ha4 8'hc0 8'h81 8'h67 8'h74 8'hd6 8'ha2 8'he1 8'h97
175 : 8'h72 8'hbb 8'h52 8'h87 8'hd9 8'hfd 8'he2 8'h85 8'h06 8'h3c 8'had 8'h54 8'hb4 8'h16 8'h7b 8'h79
159 : 8'hcc 8'hdc 8'h5f 8'h73 8'h3d 8'h55 8'h32 8'h40 8'h9c 8'hbc 8'hc5 8'h50 8'hc7 8'h8b 8'h2e 8'h37
143 : 8'hcd 8'hc2 8'h59 8'h6f 8'h01 8'h15 8'hlf 8'h79 8'he6 8'he7 8'hb5 8'h7c 8'h78 8'h5a 8'h1a 8'h95
127 : 8'h37 8'h96 8'h9c 8'hda 8'h04 8'h68 8'hc4 8'h2e 8'he7 8'h2f 8'ha7 8'hc2 8'h54 8'h90 8'h2d 8'hae
111 : 8'h96 8'hd6 8'h6a 8'hda 8'h32 8'h89 8'h8d 8'h8a 8'h28 8'h73 8'hf6 8'ha4 8'h6b 8'hb7 8'hfd 8'hcd
95 : 8'h16 8'he2 8'h64 8'hd5 8'ha3 8'h4b 8'h18 8'h8d 8'h97 8'h28 8'h29 8'h7b 8'h01 8'h1c 8'h5e 8'h12
79 : 8'h06 8'hf1 8'hc4 8'h30 8'h56 8'h0e 8'h80 8'hdc 8'hf1 8'hd0 8'h2e 8'h3c 8'h02 8'h9d 8'h15 8'hc7
63 : 8'h0e 8'h23 8'h45 8'hc9 8'ha5 8'hec 8'h21 8'h82 8'hf6 8'hda 8'hdc 8'h7e 8'h43 8'h6e 8'h60 8'h7e
47 : 8'h40 8'hb3 8'h3a 8'h63 8'hbc 8'h84 8'ha4 8'h65 8'hla 8'h24 8'h54 8'h88 8'hd8 8'h86 8'h49 8'hfc
31 : 8'h05 8'hf4 8'h5e 8'h96 8'hd7 8'h9e 8'h66 8'hd8 8'h59 8'hca 8'ha2 8'hef 8'hc3 8'h76 8'h09 8'hba
15 : 8'h33 8'hfd 8'he9 8'h97 8'h07 8'h70 8'h0d 8'h07 8'h9b 8'h06 8'h3c 8'h6f 8'h8d 8'hfc 8'h17 8'ha4
```

Qlint Snippet (No Errors)

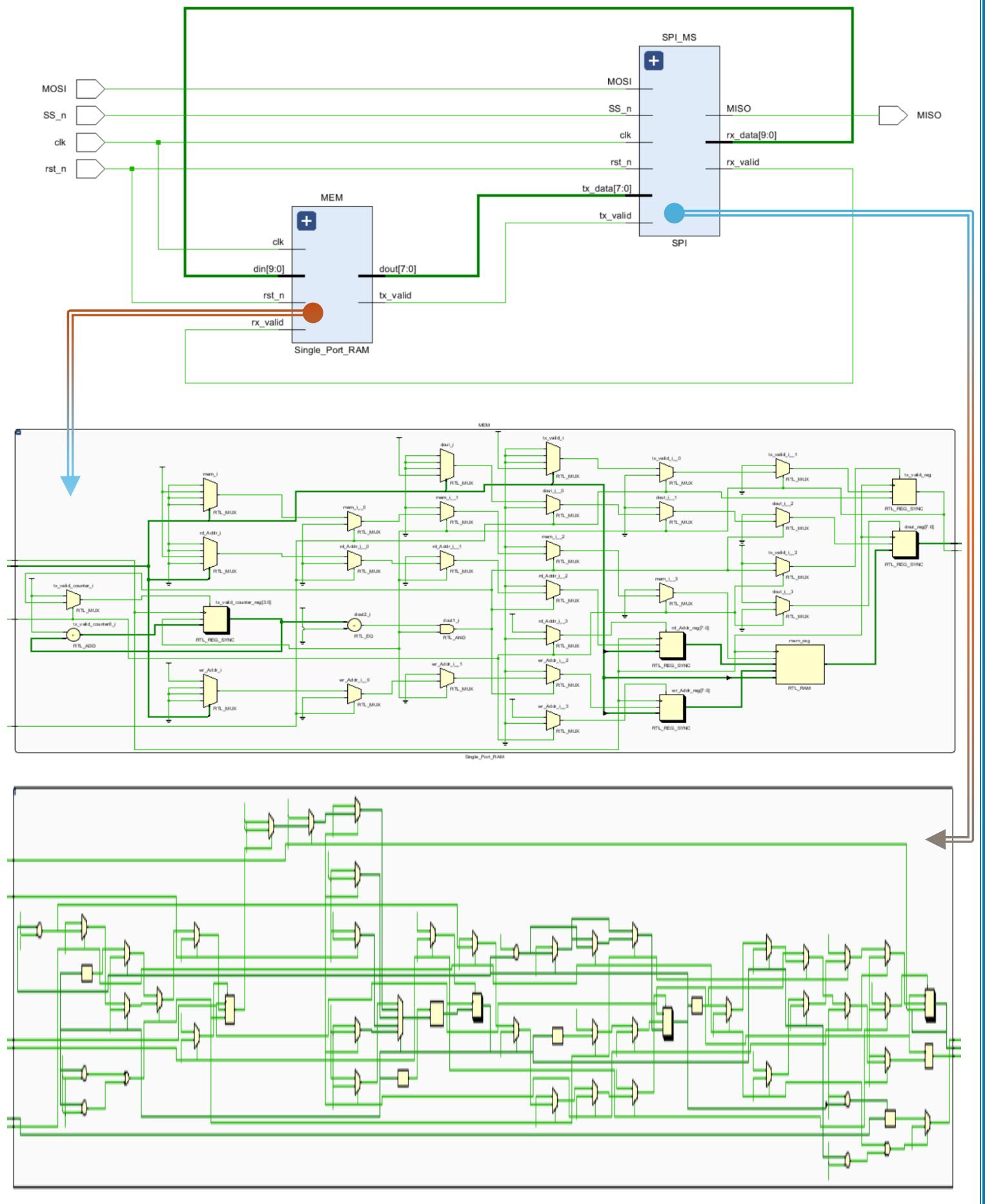
The screenshot shows the Qlint tool interface. The main window displays a Verilog code snippet for a SPI wrapper module. The code includes declarations for MOSI, SS_n, CLK, RST_n, and MISO pins, as well as rx_data and tx_data wires. It also includes SPI and RAM interface declarations. The code is annotated with line numbers from 1 to 10.

```
1 module SPI_Wrapper (MOSI,SS_n,clk,rst_n,MISO); // top module
2   input MOSI,SS_n,clk,rst_n;
3   output MISO;
4   wire [9:0] rx_data;
5   wire [7:0] tx_data;
6   wire tx_valid,rx_valid;
7   SPI SPI_MS(MOSI,SS_n,clk,rst_n,tx_data,tx_valid,rx_data,rx_valid,MISO);
8   Single_Port_RAM MEM(rx_data,rx_valid,clk,rst_n,tx_data,tx_valid);
9
10 endmodule //SPI_Wrapper
```

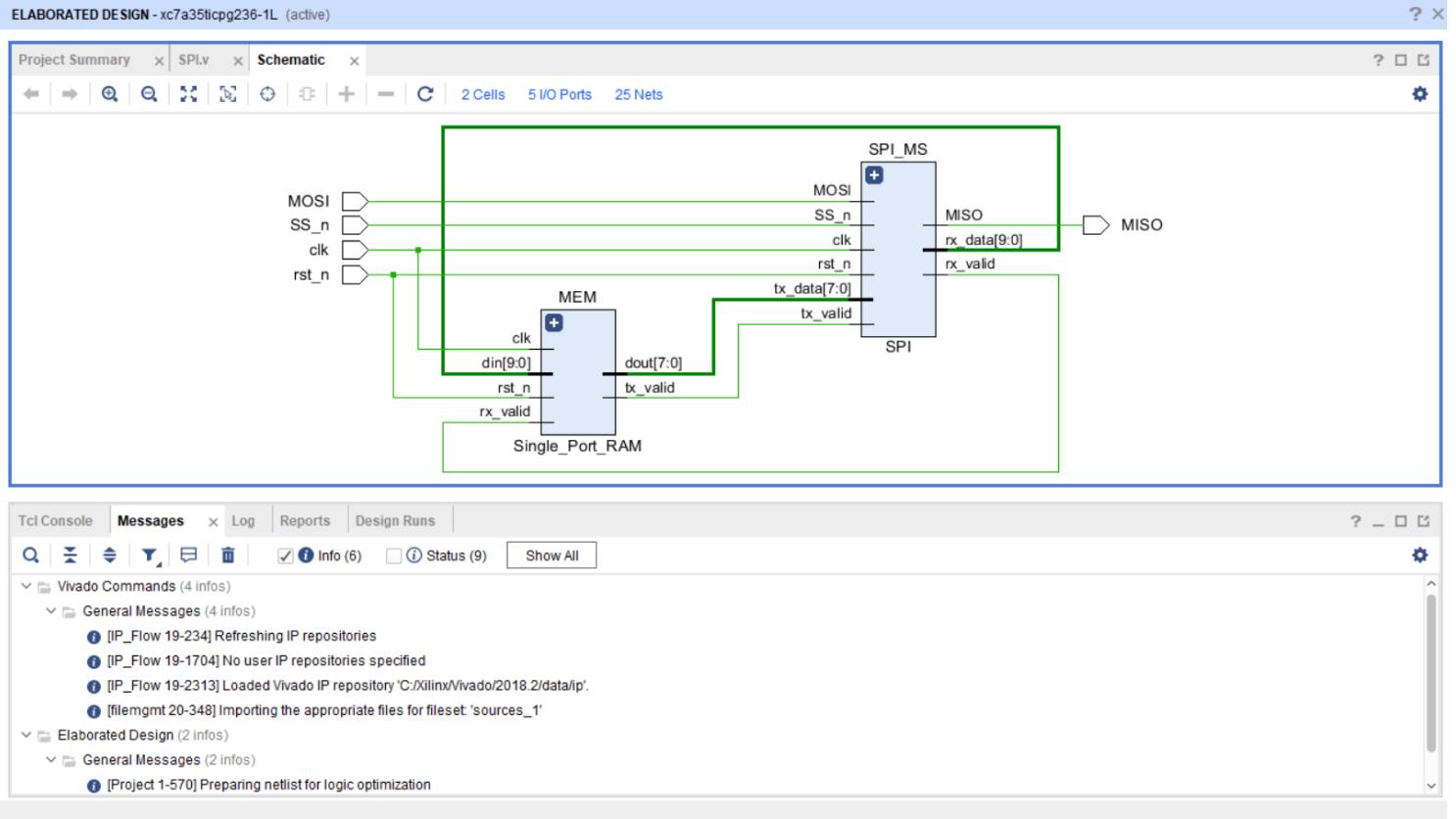
Below the code editor, there is a toolbar with icons for Flow Navi..., D..., SPI_Wrapper.v, and Single_Port_RAM.v. A Lint Checks section shows a green checkmark and a filter field. At the bottom, a table provides a detailed view of the lint results, with columns for Severity, Status, Check, Alias, Message, Module, Category, State, Owner, and STARC Reference.

Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference

RTL Schematic



RTL Schematic Messages



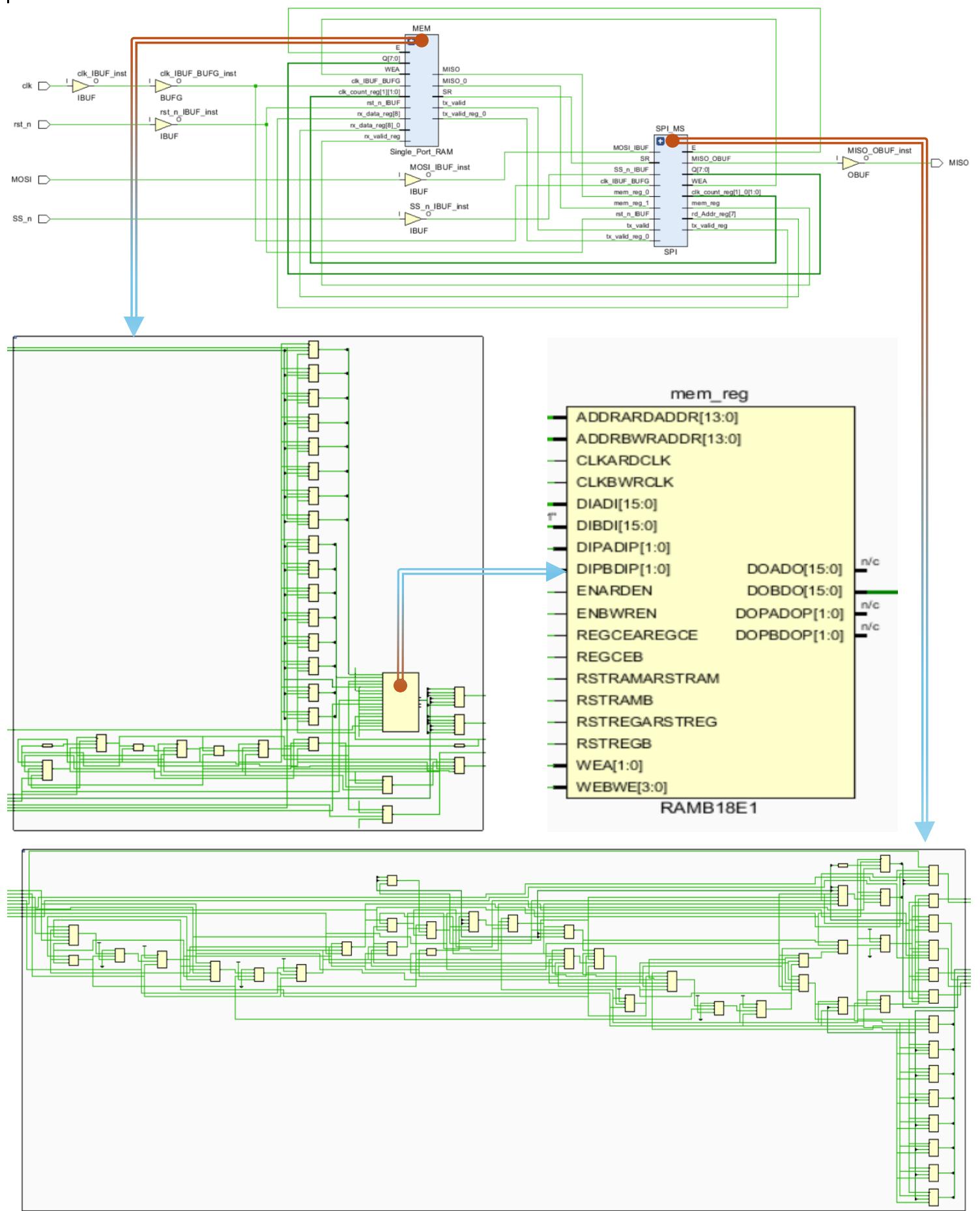
Note :RTL schematic is the same for three different encoding methods

Sequential Encoding

Encoding Report

State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	010	010
READ_ADD	011	011
READ_DATA	100	100

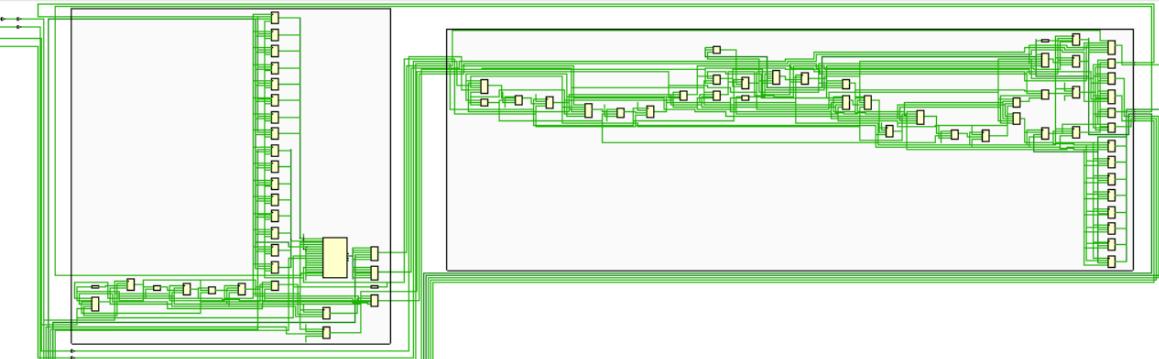
Synthesis Schematic



Synthesis Messages Tab

SYNTHESIZED DESIGN - xc7a35tciplg236-1L (active)

SPI.v Schematic 85 Cells 5 I/O Ports 147 Nets



Tcl Console Messages Log Reports Design Runs Debug

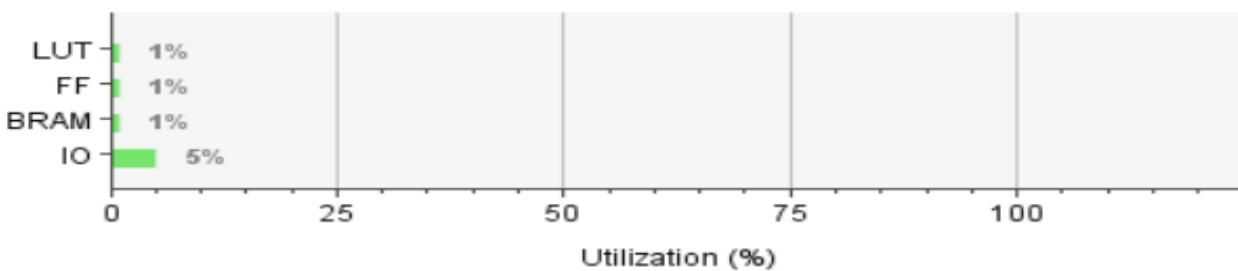
Q ? _ □ □ Show All

Warning (2) Info (56) Status (30)

Vivado Commands (4 infos)
General Messages (4 infos)
[IP_Flow 19-234] Refreshing IP repositories
[IP_Flow 19-1704] No user IP repositories specified
[IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.2/data/ip'.
[filemgmt 20-348] Importing the appropriate files for fileset: 'sources_1'
Elaborated Design (11 infos)
General Messages (11 infos)
[Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI_Wrapper.v:1] (2 more like this)

Synthesis Utilization & Timing Reports

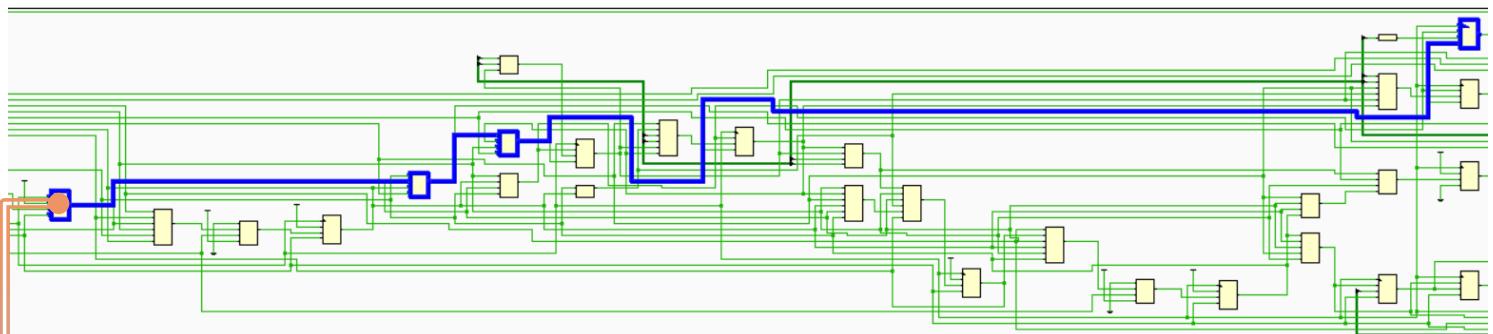
Resource	Utilization	Available	Utilization %
LUT	29	20800	0.14
FF	43	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.468 ns	Worst Hold Slack (WHS): 0.146 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: 105	Total Number of Endpoints: 105	Total Number of Endpoints: 43

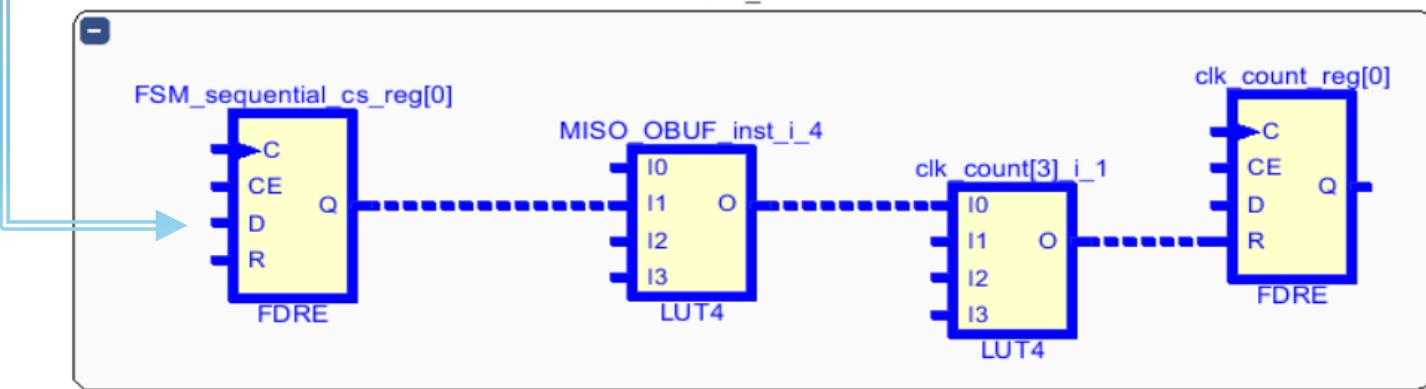
All user specified timing constraints are met.

Synthesis Critical Path



Timing							
Intra-Clock Paths - sys_clk_pin - Setup							
Name	Slack	^1	Levels	High Fanout	From	To	Total Delay
Path 1	6.468		2	10	SPI_MS/FSM_seq...l_cs_reg[0]/C	SPI_MS/clk_count_reg[0]/R	2.919
Path 2	6.468		2	10	SPI_MS/FSM_seq...l_cs_reg[0]/C	SPI_MS/clk_count_reg[1]/R	2.919

SPI_MS



Implementation Messages Tab

IMPLEMENTED DESIGN - xc7a35tclcp236-1L (active)

Project Summary Device SPIv synth_1_synth_synthesis_report_0 - synth_1

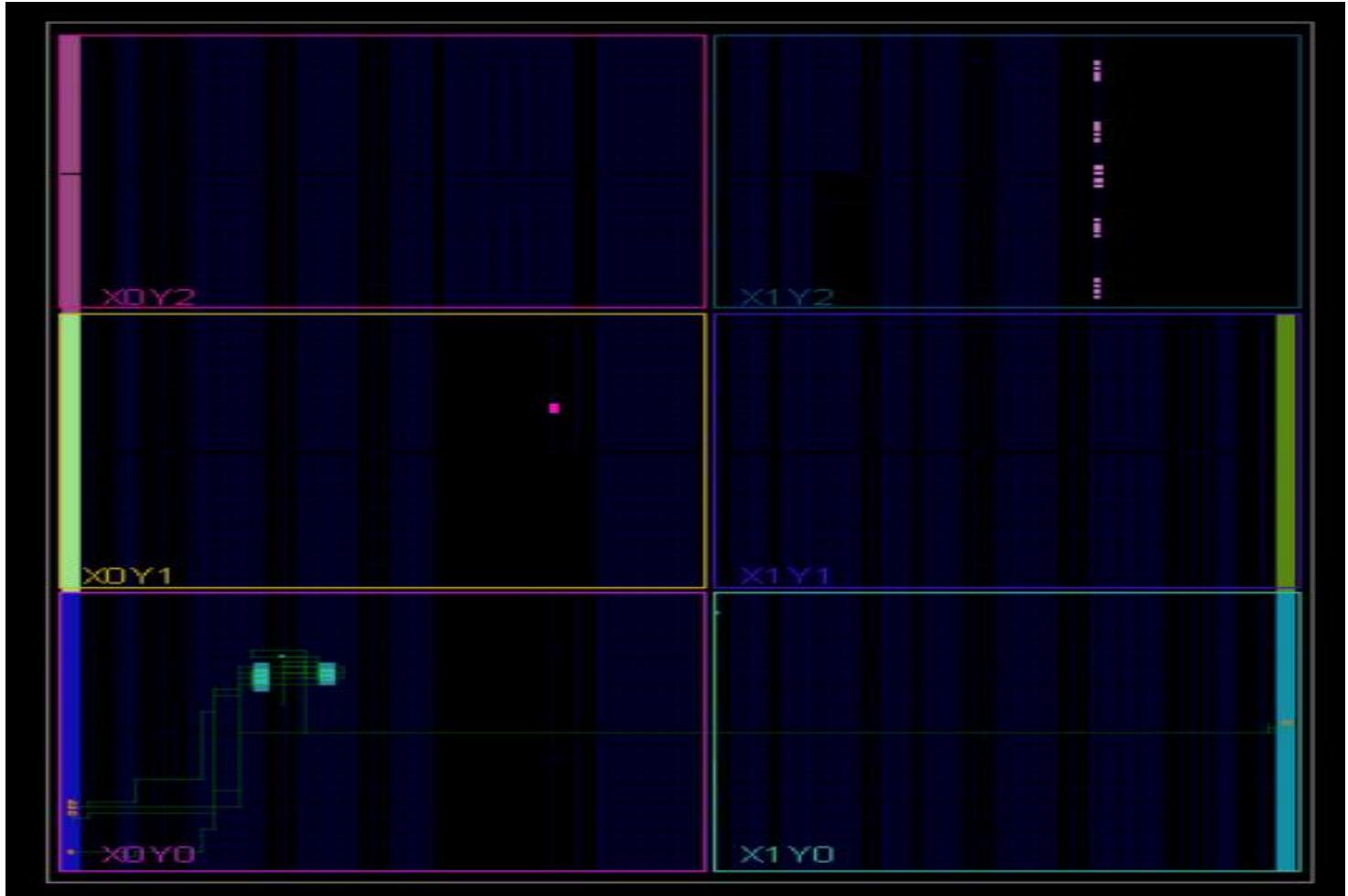
Netlist Cell Properties

Tcl Console Messages Log Reports Design Runs Power DRC Methodology Timing

Warning (2) Info (269) Status (503) Show All

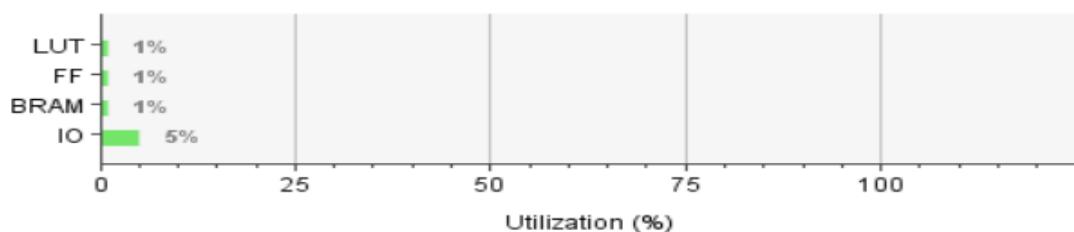
[Netlist 29-1] Analyzing 5 Unisim elements for replacement
[Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
[Project 1-479] Netlist was created with Vivado 2018.2
[Project 1-570] Preparing netlist for logic optimization
[Timing 38-478] Restoring timing data from binary archive.
[Timing 38-479] Binary timing data restore complete.
[Project 1-856] Restoring constraints from binary archive.
[Project 1-853] Binary constraint restore complete.
[Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

Implementation Device



Implementation Utilization & Timing Reports

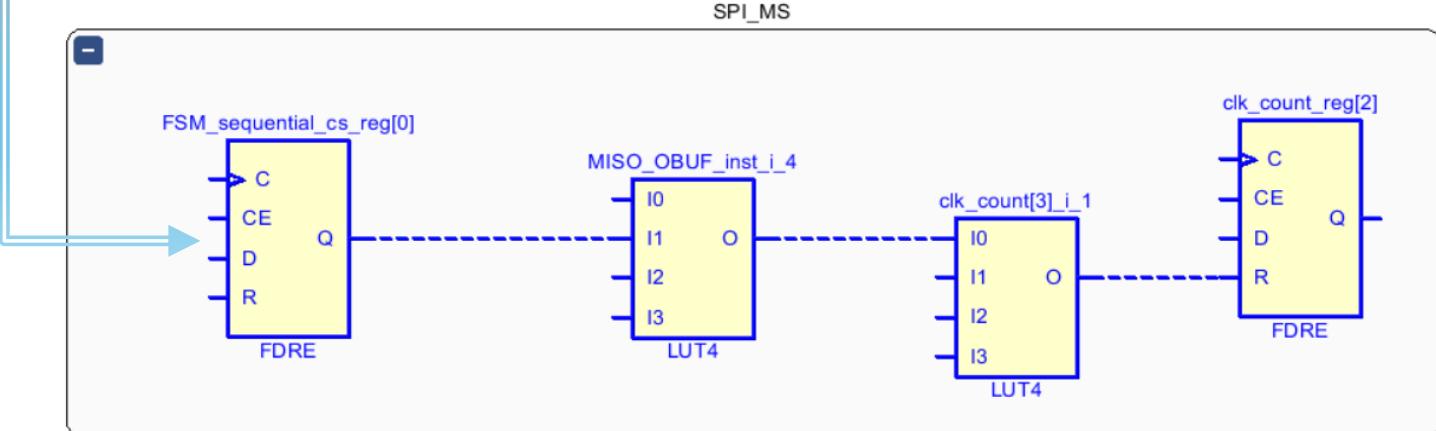
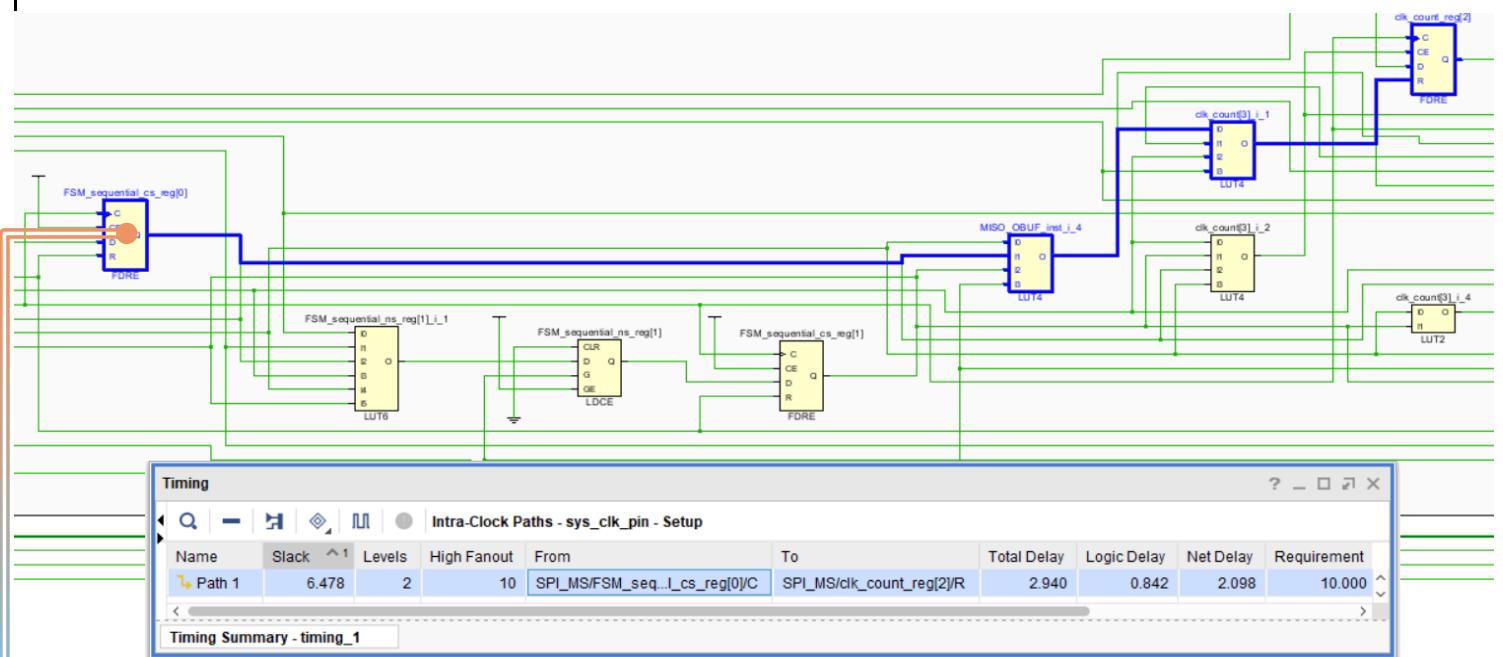
Resource	Utilization	Available	Utilization %
LUT	30	20800	0.14
FF	43	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.478 ns	Worst Hold Slack (WHS): 0.101 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: 106	Total Number of Endpoints: 106	Total Number of Endpoints: 43

All user specified timing constraints are met.

Implementation Critical Path

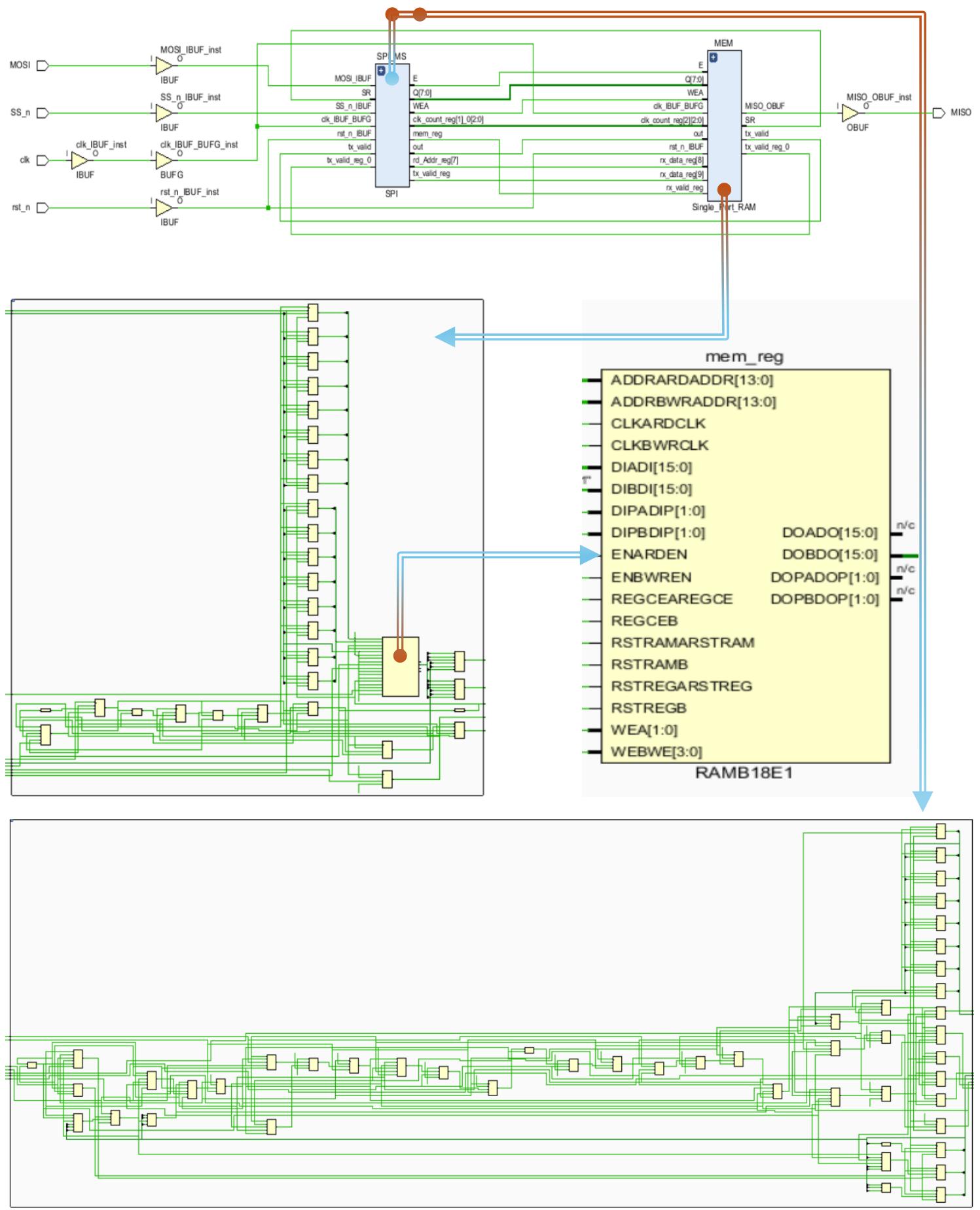


One hot Encoding

Encoding Report

State	New Encoding	Previous Encoding
IDLE	00001	000
CHK_CMD	00010	001
WRITE	00100	010
READ_ADD	01000	011
READ_DATA	10000	100

Synthesis Schematic



Synthesis Messages Tab

SYNTHESIZED DESIGN - xc7a35ticpg236-1L (active)

Schematic

90 Cells 5 I/O Ports 152 Nets

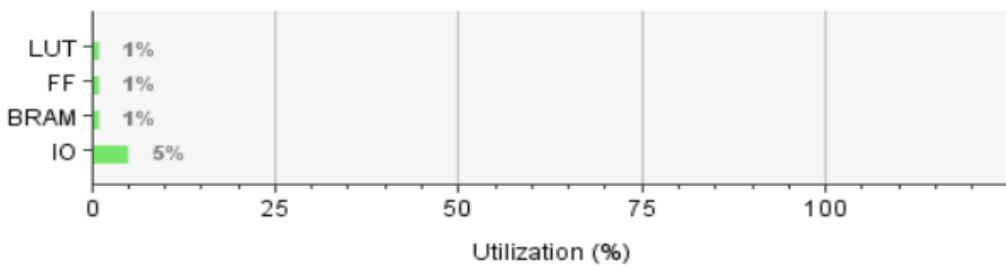
Tcl Console | Messages | Log | Reports | Design Runs | Debug | ? □ ×

General Messages (12 infos)

- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Project 1-570] Preparing netlist for logic optimization
- [Timing 38-478] Restoring timing data from binary archive.
- [Timing 38-479] Binary timing data restore complete.
- [Project 1-856] Restoring constraints from binary archive.
- [Project 1-853] Binary constraint restore complete.
- > [Timing 38-91] UpdateTimingParams: Speed grade: -1L, Delay Type: min_max. (1 more like this)
- > [Timing 38-191] Multithreading enabled for timing update using a maximum of 2 CPUs (1 more like this)

Synthesis Utilization & Timing Reports

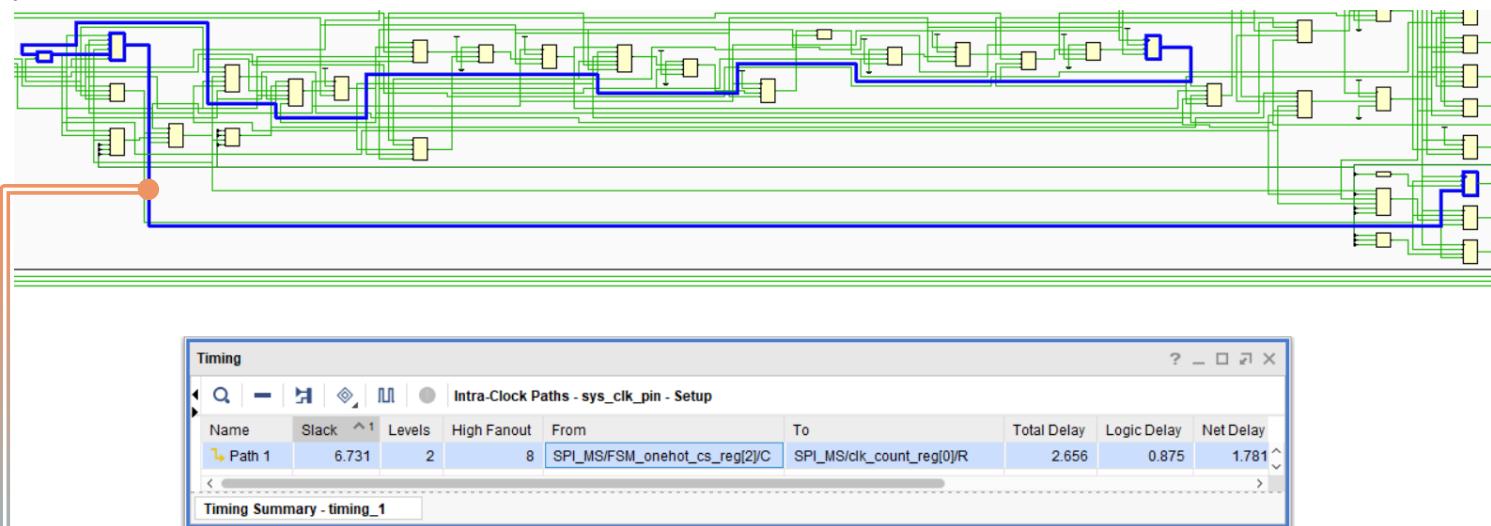
Resource	Utilization	Available	Utilization %
LUT	30	20800	0.14
FF	47	41600	0.11
BRAM	0.50	50	1.00
IO	5	106	4.72



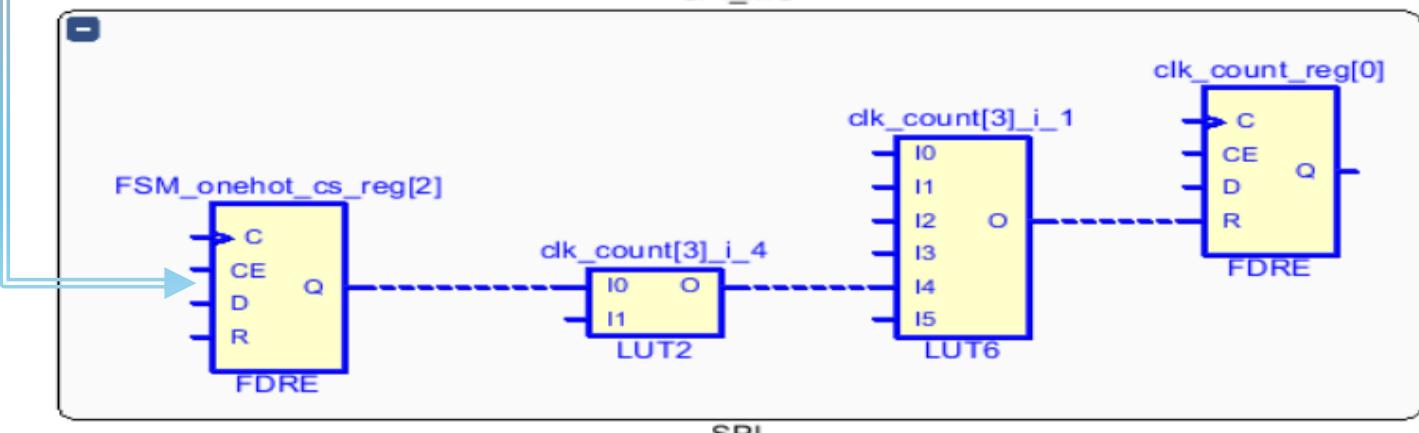
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.731 ns	Worst Hold Slack (WHS): 0.145 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 105	Total Number of Endpoints: 105	Total Number of Endpoints: 45

All user specified timing constraints are met.

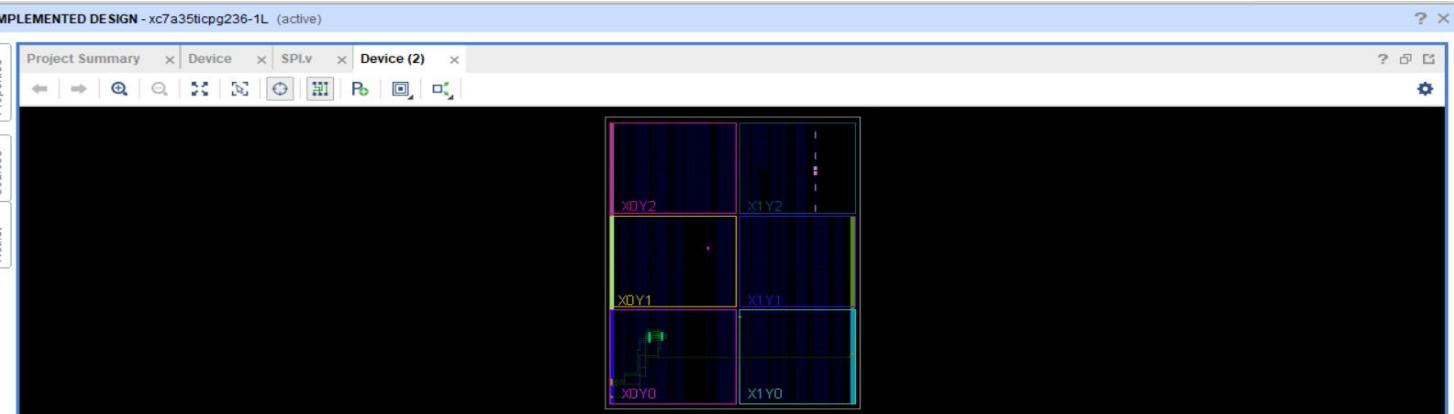
Synthesis Critical Path



SPI_MS

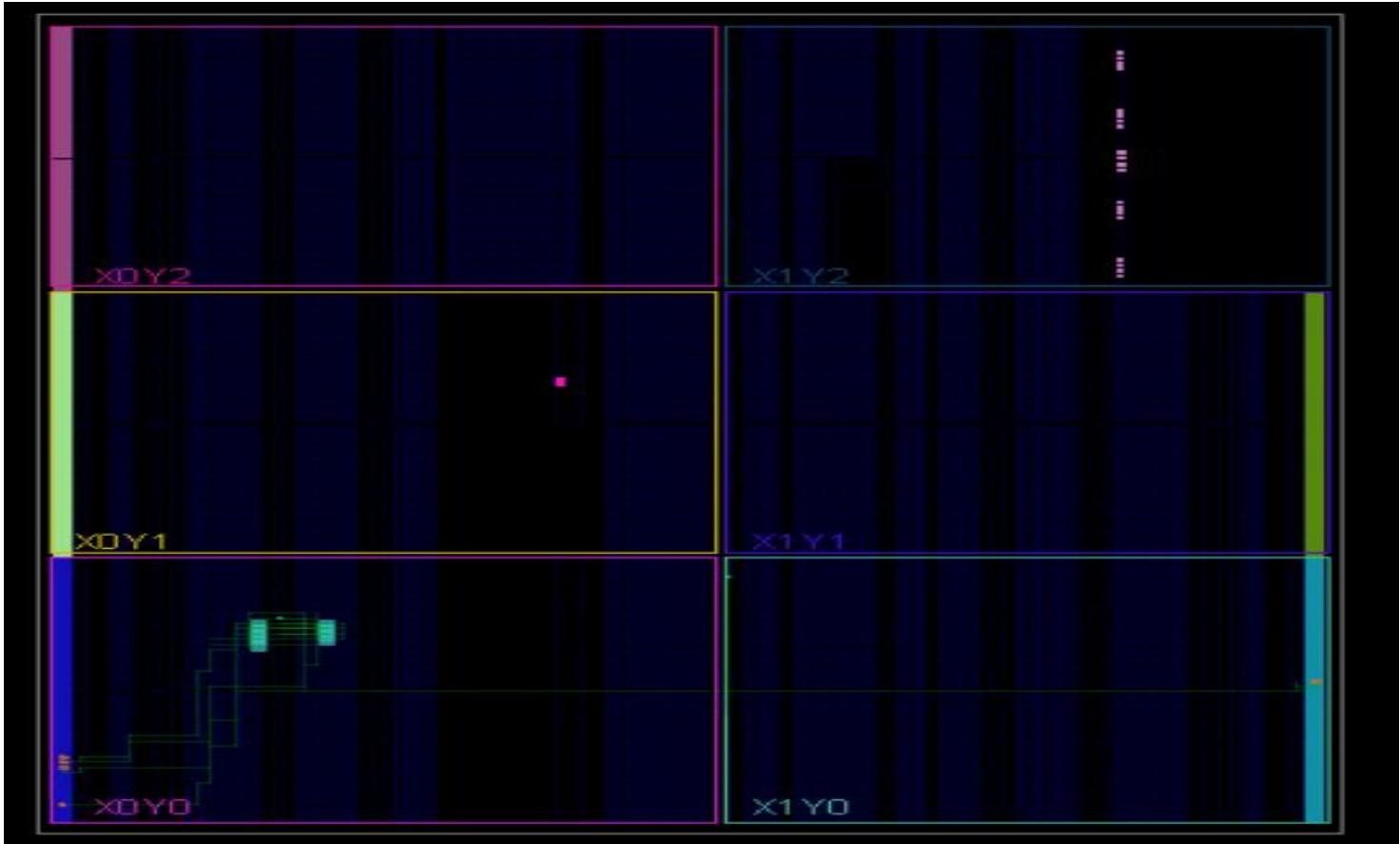


Implementation Messages Tab



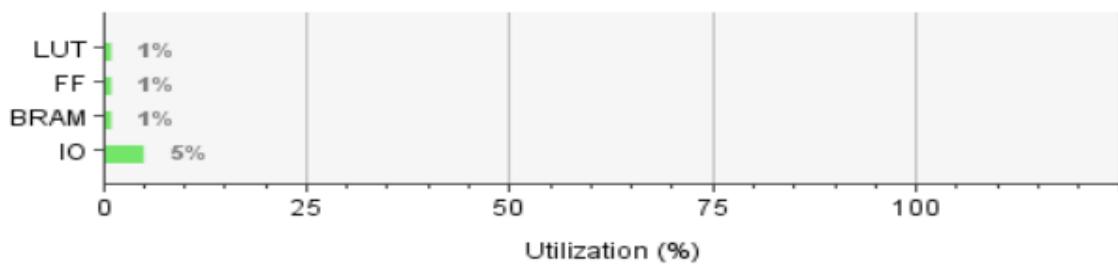
Tcl Console						Messages	Log	Reports	Methodology	Power	DRC
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Warning (2)	<input checked="" type="checkbox"/> Info (257)	<input type="checkbox"/>	<input checked="" type="checkbox"/> Status (501)	Show All			
① [Netlist 29-17] Analyzing 5 Unisim elements for replacement	① [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds	① [Project 1-479] Netlist was created with Vivado 2018.2	① [Device 21-403] Loading part xc7a35tcpg236-1L	① [Project 1-570] Preparing netlist for logic optimization	① [Timing 38-478] Restoring timing data from binary archive.	① [Timing 38-479] Binary timing data restore complete.	① [Project 1-856] Restoring constraints from binary archive.	① [Project 1-853] Binary constraint restore complete.			

Implementation Device



Implementation Utilization & Timing Reports

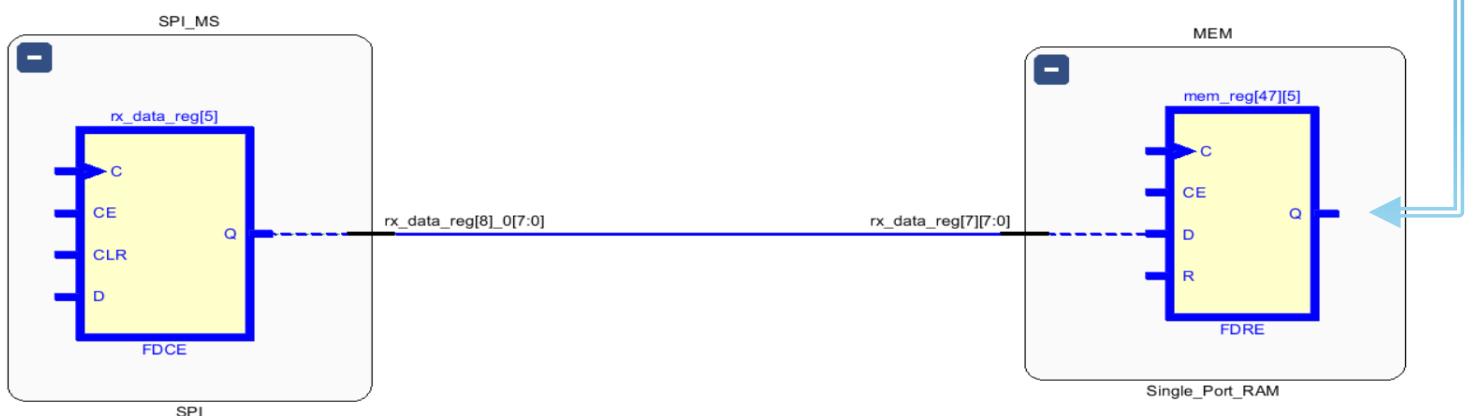
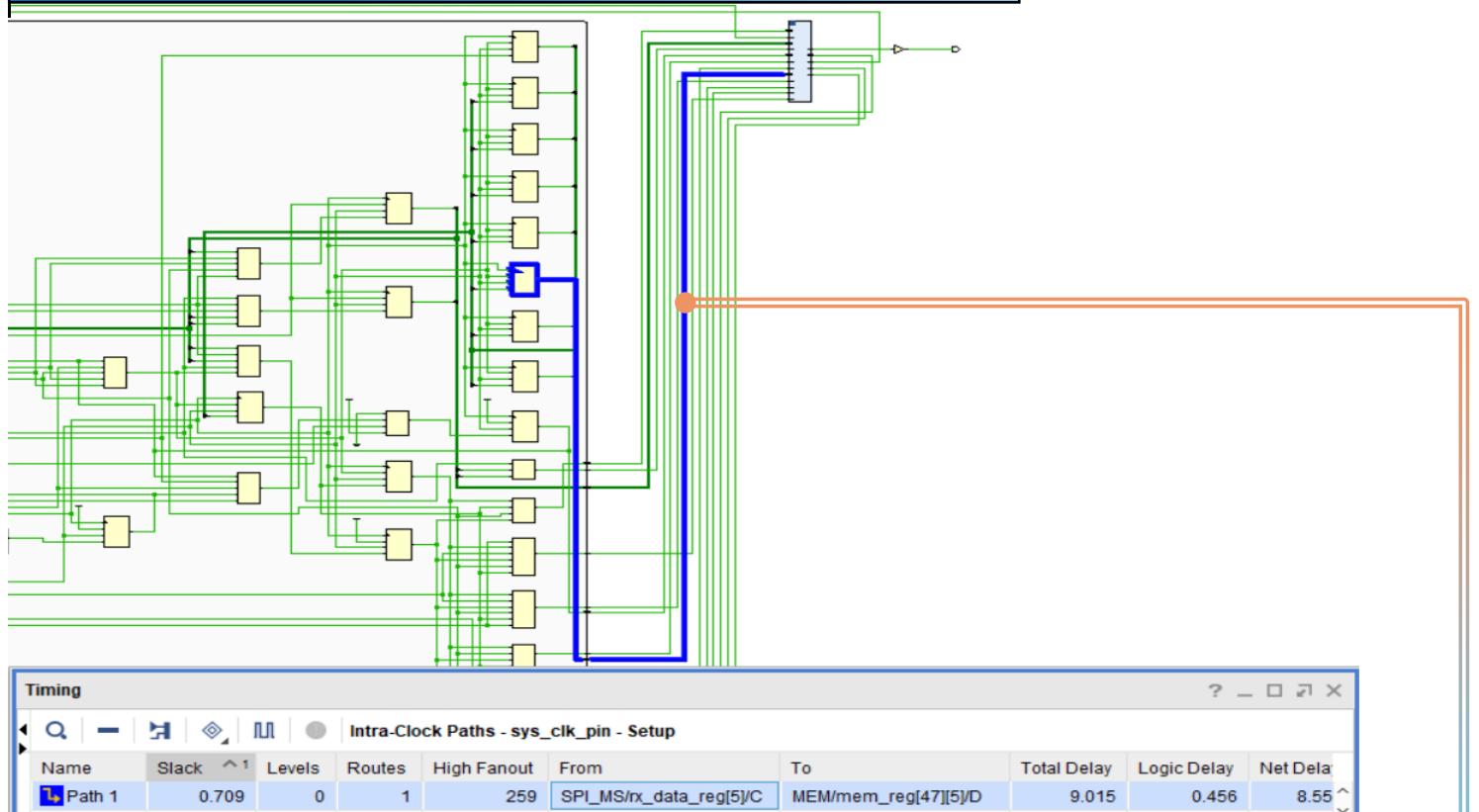
Resource	Utilization	Available	Utilization %
LUT	31	20800	0.15
FF	47	41600	0.11
BRAM	0.50	50	1.00
IO	5	106	4.72



Setup	Hold	Pulse Width	
Worst Negative Slack (WNS):	6.822 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	106	Total Number of Endpoints:	45

All user specified timing constraints are met.

Implementation Critical Path

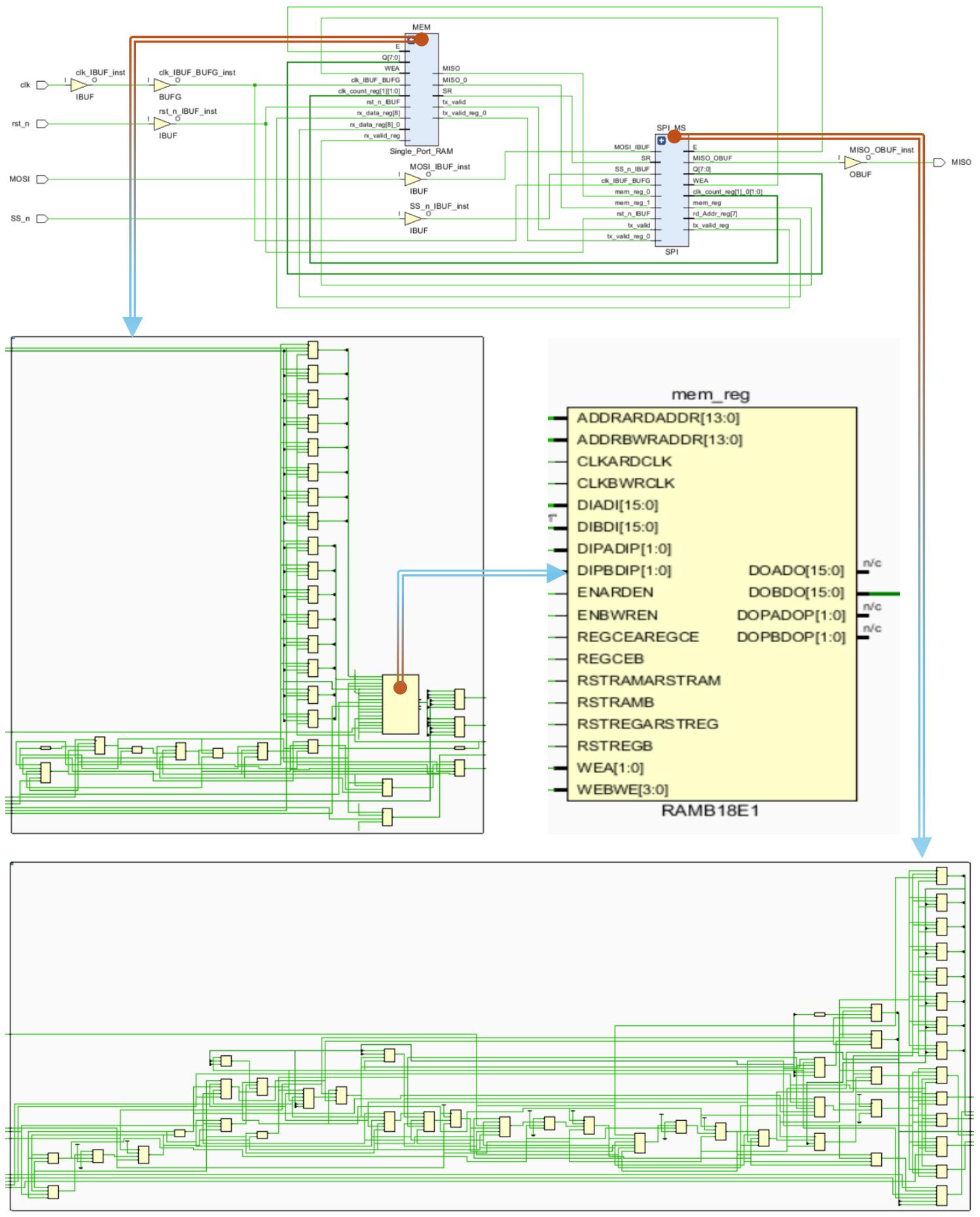


Gray Encoding

Encoding Report

State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	011	010
READ_ADD	010	011
READ_DATA	111	100

Synthesis Schematic



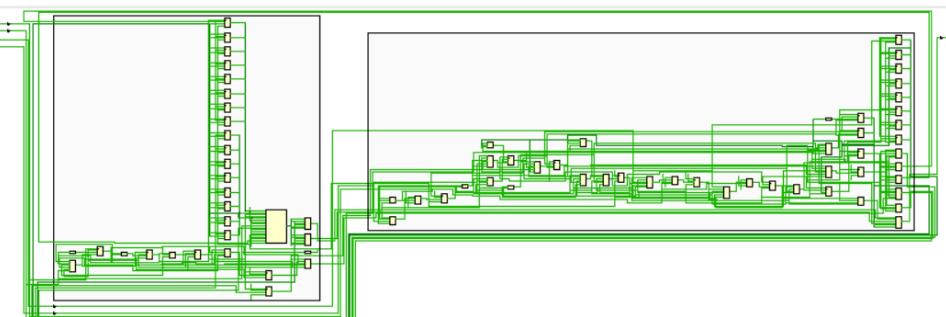
Synthesis Messages Tab

SYNTHESIZED DESIGN - xc7a35ticpg236-1L (active)

SPI.v × synth_1_synth_synthesis_report_0 - synth_1 × Schematic ×

Sources Netlist Cell Properties

85 Cells 5 I/O Ports 147 Nets



Tcl Console Messages Log Reports Design Runs Timing Debug

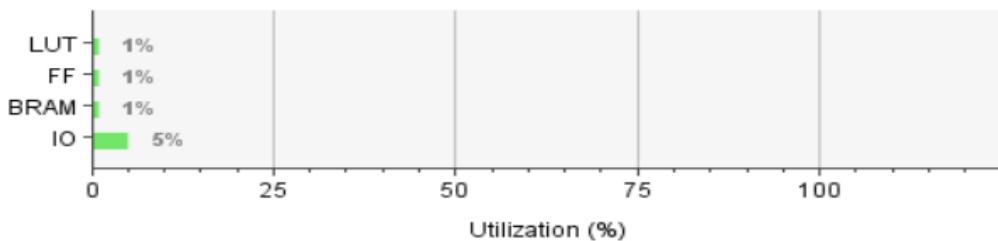
Q ? - □ □ Show All

Warning (2) Info (69) Status (39)

- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Project 1-570] Preparing netlist for logic optimization
- [Timing 38-478] Restoring timing data from binary archive.
- [Timing 38-479] Binary timing data restore complete.
- [Project 1-856] Restoring constraints from binary archive.
- [Project 1-853] Binary constraint restore complete.
- [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.
- > [Timing 38-91] UpdateTimingParams: Speed grade: -1L, Delay Type: min_max. (1 more like this)
- > [Timing 38-191] Multithreading enabled for timing update using a maximum of 2 CPUs (1 more like this)

Synthesis Utilization & Timing Reports

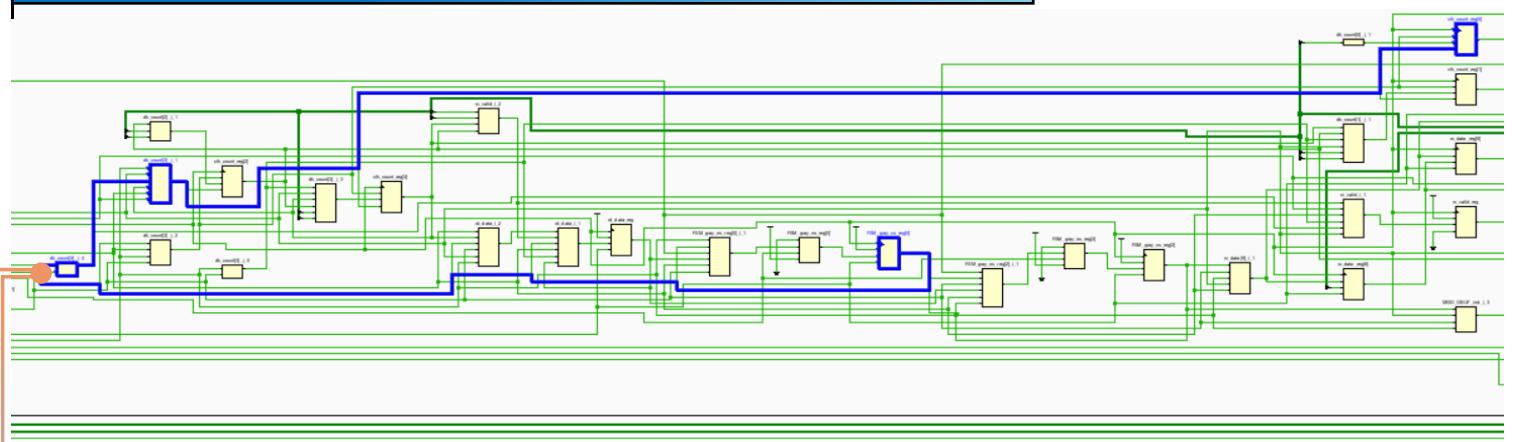
Resource	Utilization	Available	Utilization %
LUT	29	20800	0.14
FF	43	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



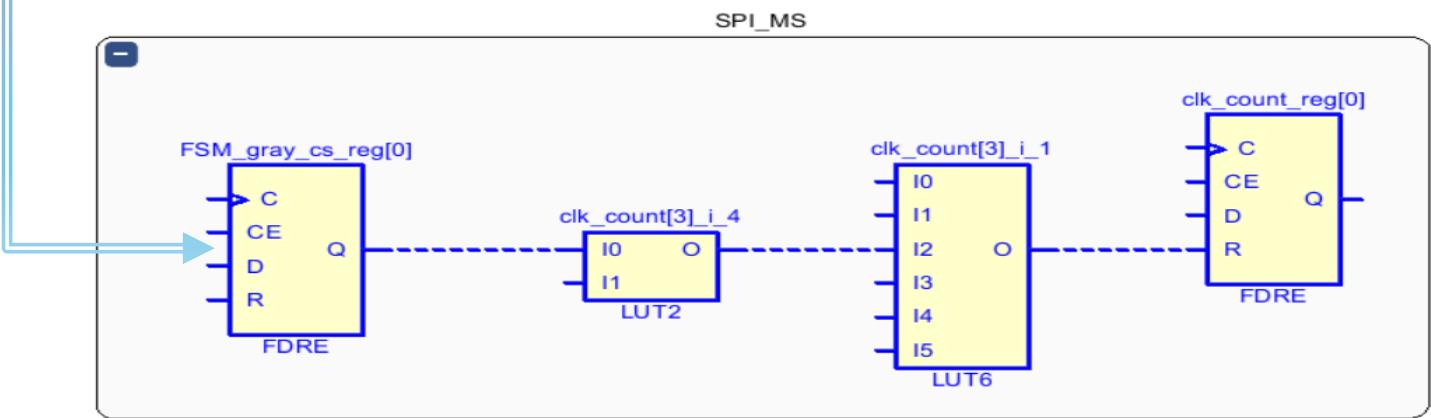
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.739 ns	Worst Hold Slack (WHS): 0.145 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 105	Total Number of Endpoints: 105	Total Number of Endpoints: 43

All user specified timing constraints are met.

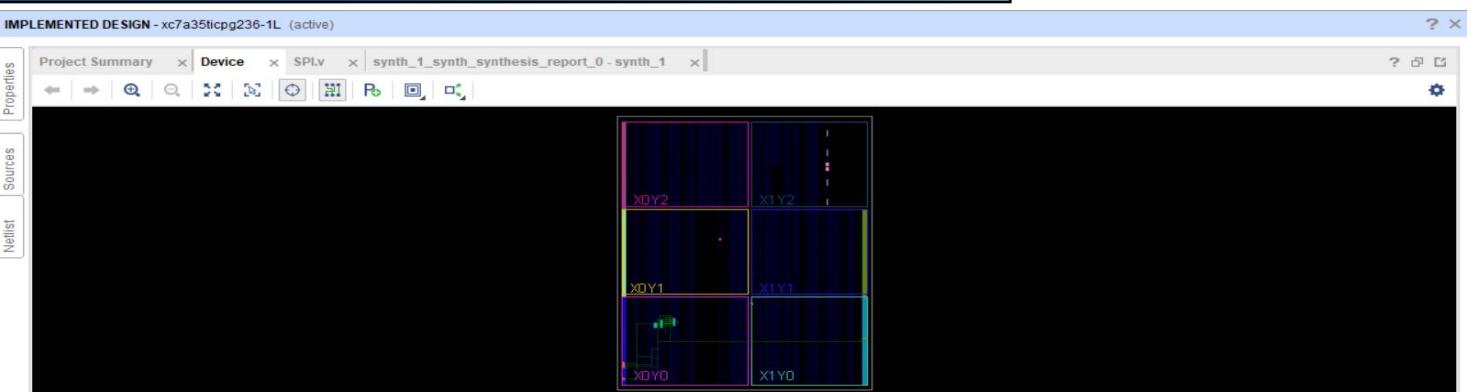
Synthesis Critical Path



Timing						
Intra-Clock Paths - sys_clk_pin - Setup						
Name	Slack	^1 Levels	High Fanout	From	To	Total Delay
Path 1	6.739	2	10	SPI_MS/FSM_gray_cs_reg[0]/C	SPI_MS/clk_count_reg[0]/R	2.648
Timing Summary - timing_1						

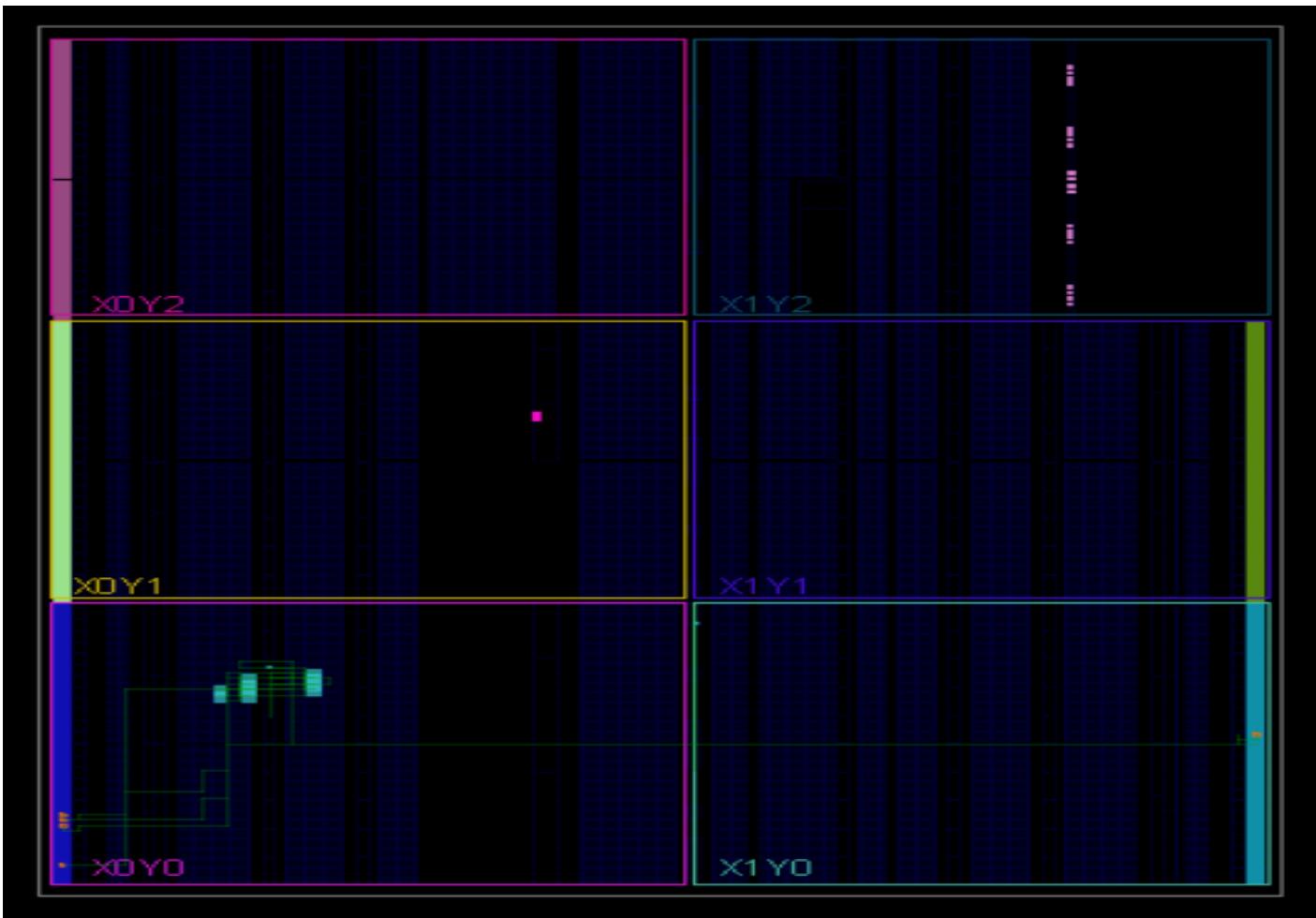


Implementation Messages Tab



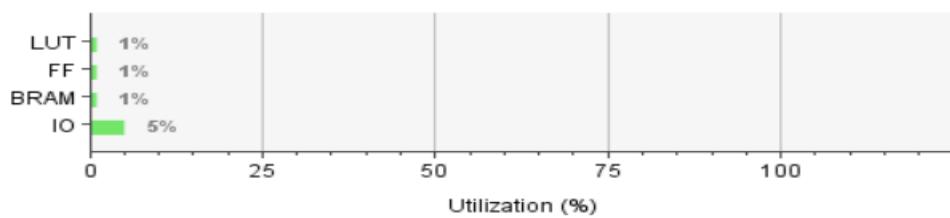
Tcl Console	Messages	x	Log	Reports	Design Runs	Methodology	Timing	Power	DRC
Netlist	Properties	Sources							
General Messages (8 Infos)									
<ul style="list-style-type: none"> ① [Netlist 29-17] Analyzing 5 Unisim elements for replacement ① [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds ① [Project 1-479] Netlist was created with Vivado 2018.2 ① [Project 1-570] Preparing netlist for logic optimization ① [Timing 38-478] Restoring timing data from binary archive. ① [Timing 38-479] Binary timing data restore complete. ① [Project 1-856] Restoring constraints from binary archive. ① [Project 1-853] Binary constraint restore complete. 									

Implementation Device



Implementation Utilization & Timing Reports

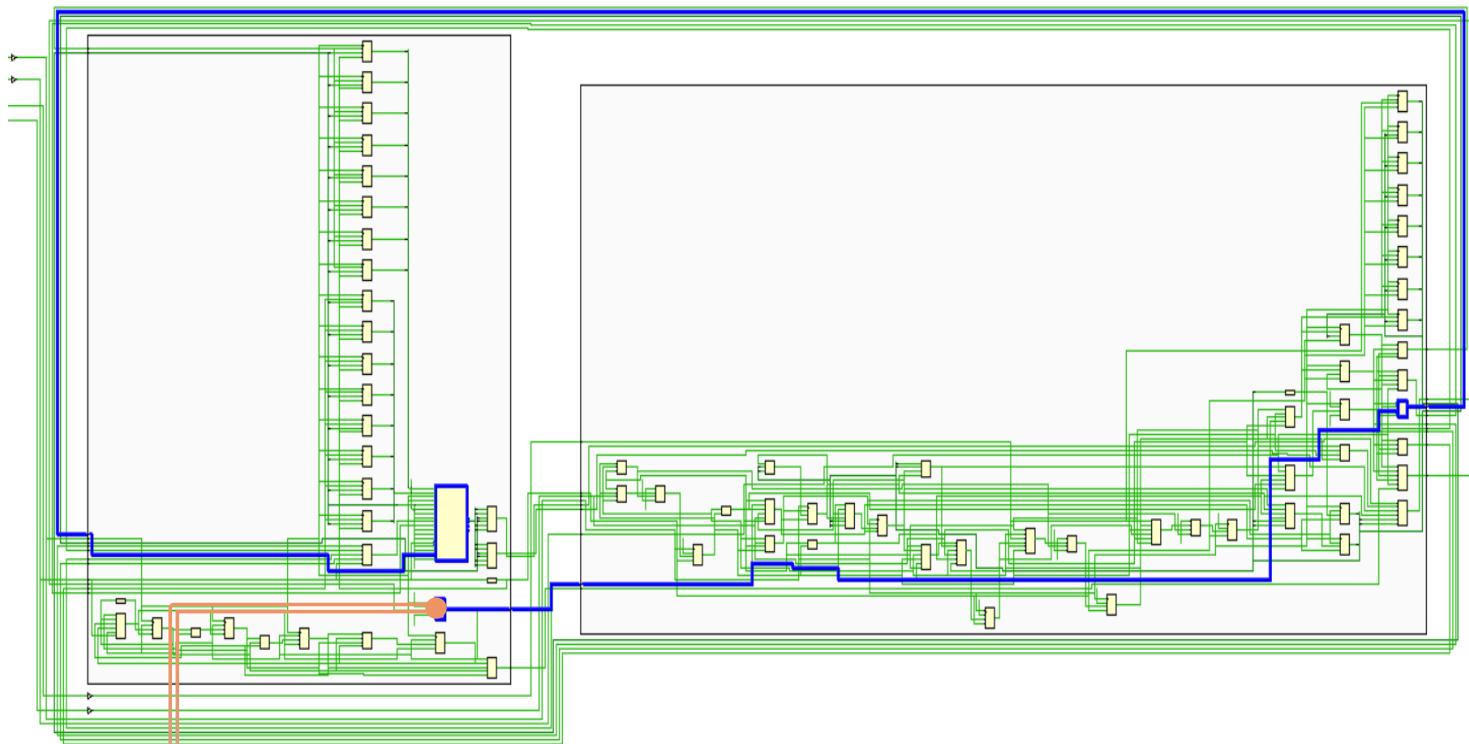
Resource	Utilization	Available	Utilization %
LUT	30	20800	0.14
FF	43	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



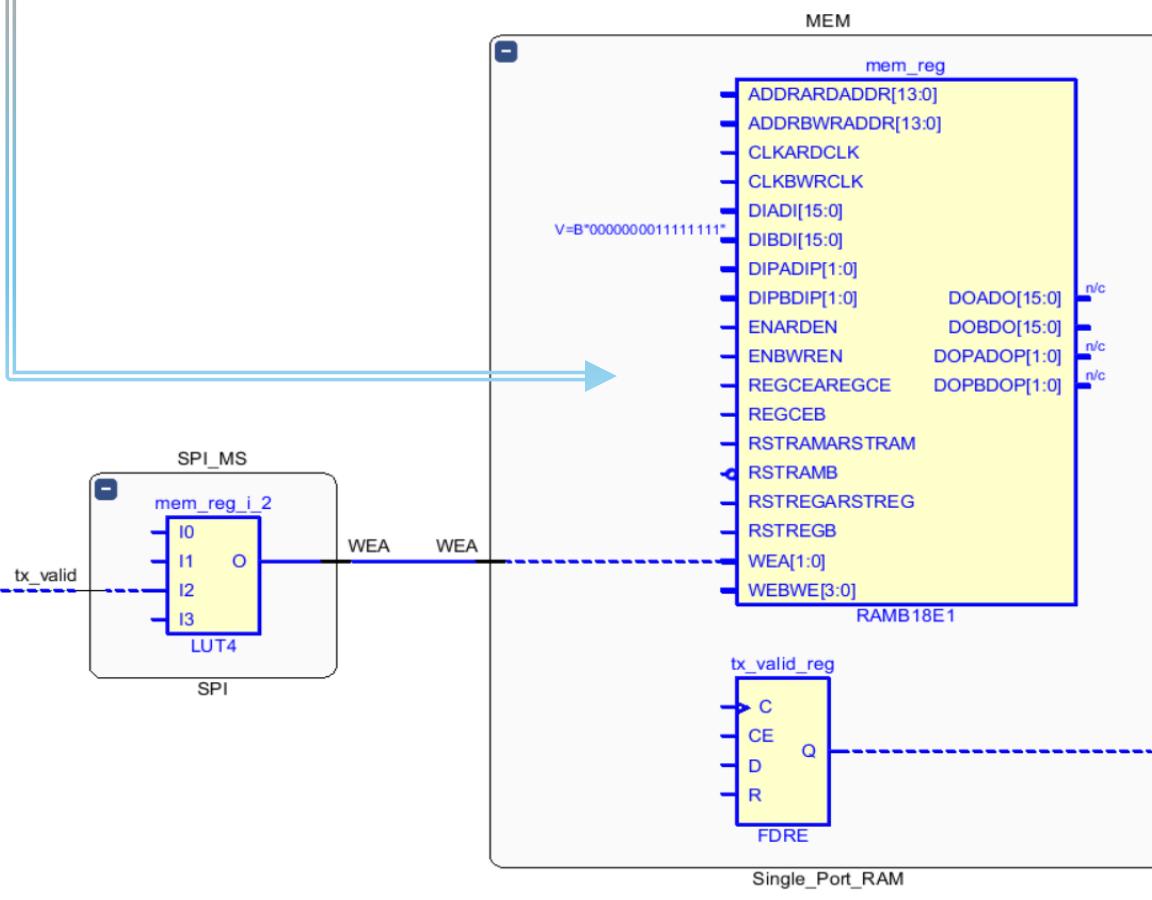
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS):	6.973 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Pulse Width Negative Slack (TPWNS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	106	Total Number of Endpoints:	43

All user specified timing constraints are met.

Implementation Critical Path



Timing						
Name	Slack	^1	Levels	High Fanout	From	To
Path 1	6.973		1	19	MEM/tx_valid_reg/C	MEM/mem_reg/WEA[1]
						Total Delay: 2.399, Logic Delay: 0.642, Net Delay: 1.757



Timing comparison of three different encoding implementation methods

Method	Setup WNS (ns)	Hold WHS (ns)	Selected
sequential	6.478	0.101	✗
One hot	6.822	0.055	✗
Gray	6.973	0.101	✓

- **Setup WNS = 6.973 ns → Best (means most margin before violating setup constraint → allows highest clock frequency).**
- **Hold WHS = 0.101 ns → Higher than One Hot**

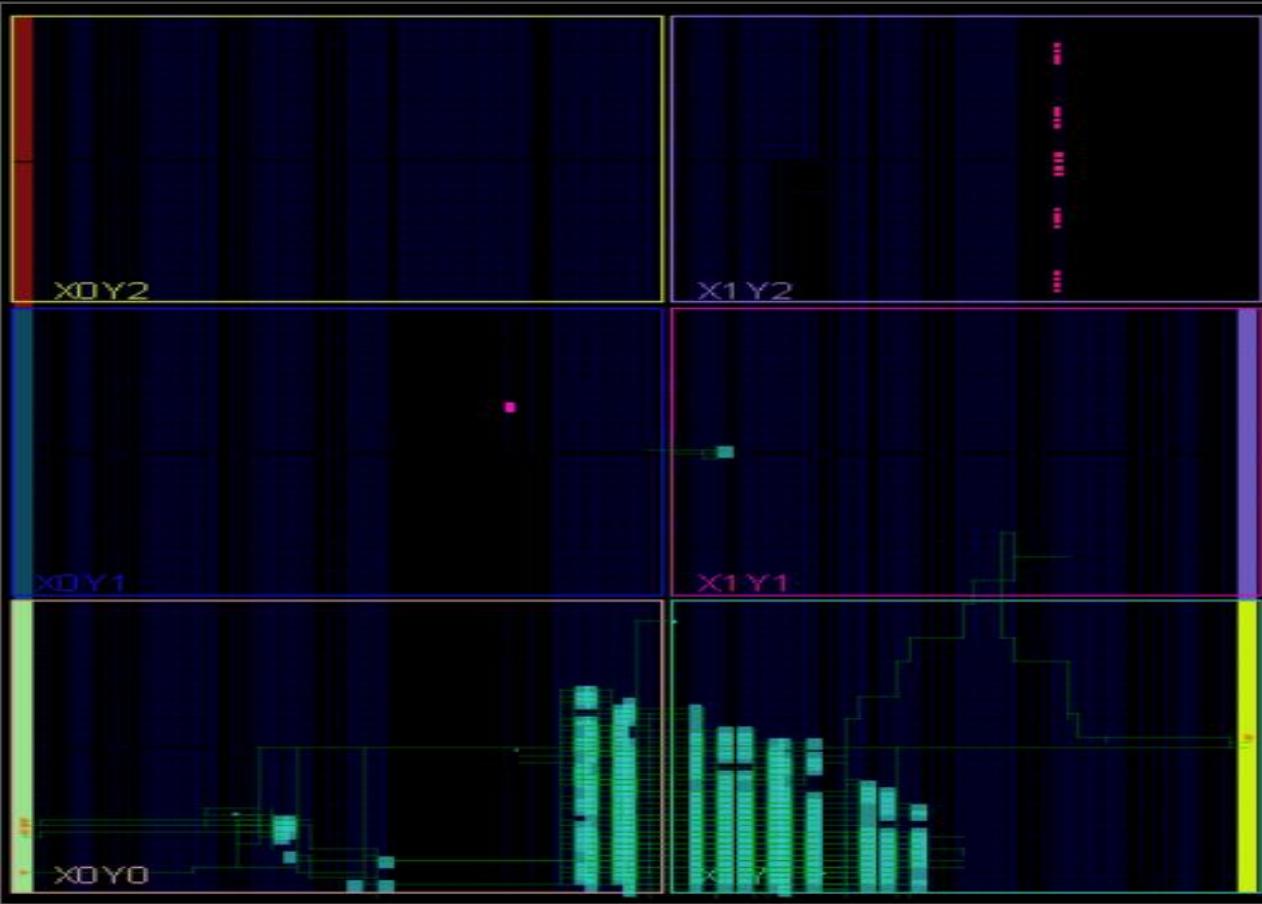
Conclusion: chose **Gray** → It gives the best timing margin for maximum frequency operation while meeting all constraints.

Implementation Message Tab After Debug Setup

The screenshot shows the Vivado IDE interface after a debug setup. The top half displays a logic diagram with four quadrants labeled X0Y2, X1Y2, X0Y1, X1Y1, and X0Y0. The bottom half shows the 'Messages' tab, which is active. The message list includes:

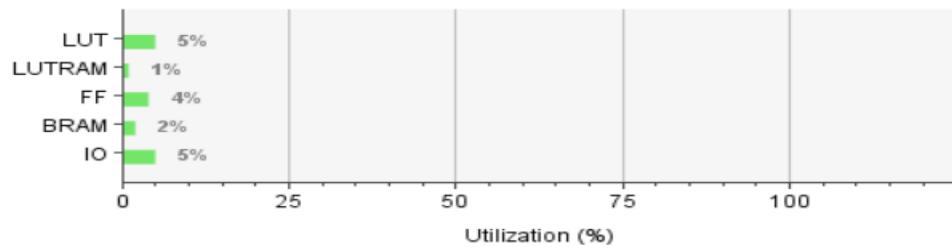
- Vivado Commands (3 infos):
 - [IP_Flow 19-234] Refreshing IP repositories
 - [IP_Flow 19-1704] No user IP repositories specified
 - [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.2/data/ip'.
- Synthesis (2 warnings, 35 infos):
 - [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
 - [Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI_Wrapper.v1] (2 more like this)
 - [Synth 8-5534] Detected attribute (* fsm_encoding = "gray") [SPI.v8]
 - [Synth 8-155] case statement is not full and has no default [SPI.v8]

New Device After Debug Setup



Implementation Utilization & Timing Reports After Debug Setup

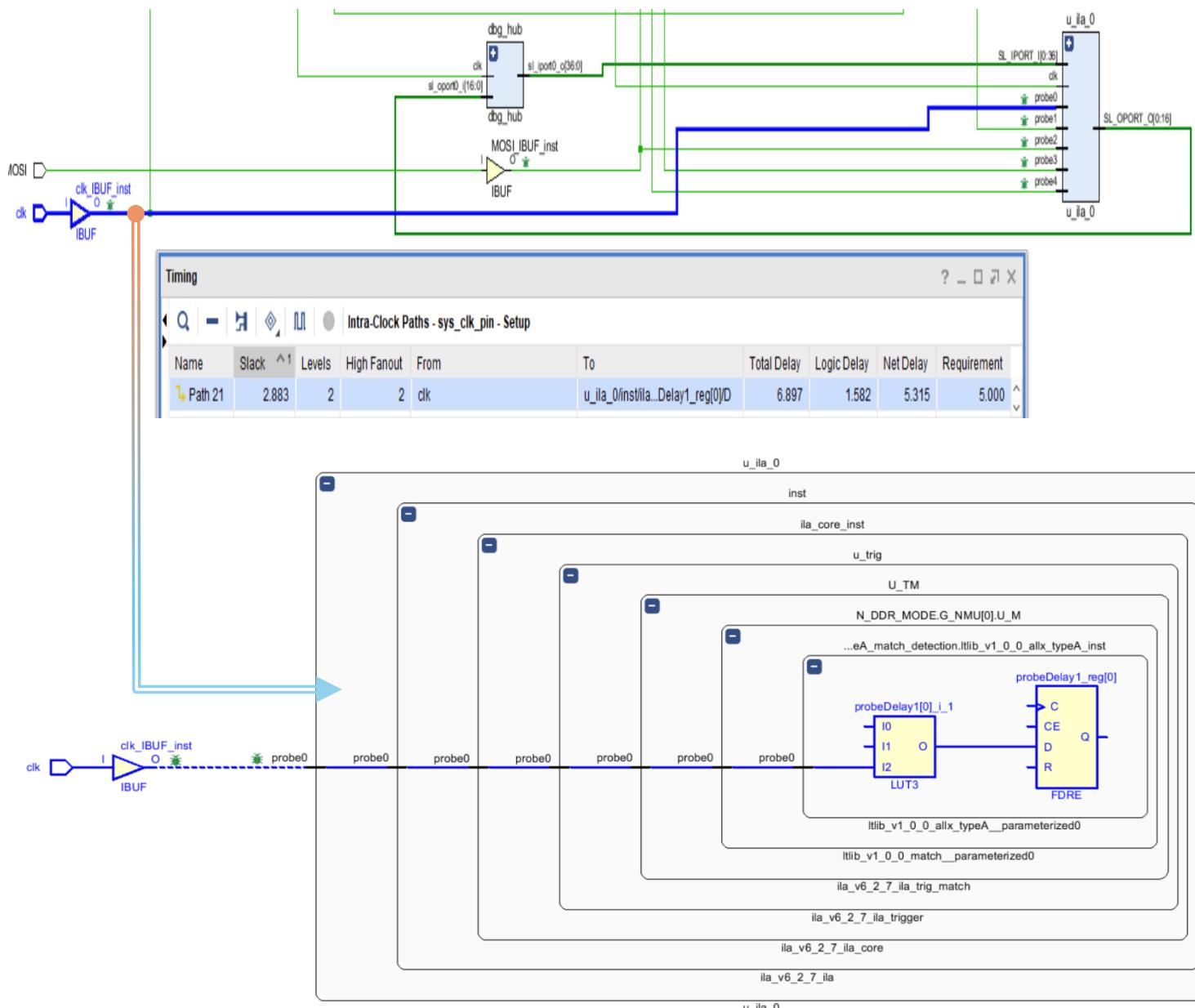
Resource	Utilization	Available	Utilization %
LUT	1101	20800	5.29
LUTRAM	88	9600	0.92
FF	1758	41600	4.23
BRAM	1	50	2.00
IO	5	106	4.72



Setup	Hold	Pulse Width	
Worst Negative Slack (WNS):	2.883 ns	Worst Hold Slack (WHS):	0.015 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3543	Total Number of Endpoints:	3527
		Number of Failing Endpoints:	0
		Total Number of Endpoints:	1905

All user specified timing constraints are met.

Implementation Critical Path After Debug Setup



Constraints File

```

6 ## Clock signal
7 set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMS33} [get_ports clk]
8 create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
9
10
11 ## Switches
12 set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMS33} [get_ports rst_n]
13 set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMS33} [get_ports SS_n]
14 set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMS33} [get_ports MOSI]
29
30 ## LEDs
31 set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMS33} [get_ports MISO]

```

Debug Core Added To Constraints File

```
152 ## Configuration options, can be used for all designs
153 set_property CONFIG_VOLTAGE 3.3 [current_design]
154 set_property CFGBVS VCCO [current_design]
155
156 ## SPI configuration mode options for QSPI boot, can be used for all designs
157 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
158 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
159 set_property CONFIG_MODE SPIx4 [current_design]
160
161 set_property MARK_DEBUG true [get_nets clk_IBUF]
162 set_property MARK_DEBUG true [get_nets MISO_OBUF]
163 set_property MARK_DEBUG true [get_nets MOSI_IBUF]
164 set_property MARK_DEBUG true [get_nets rst_n_IBUF]
165 set_property MARK_DEBUG true [get_nets SS_n_IBUF]
166 create_debug_core u_ila_0 ila
167 set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
168 set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
169 set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
170 set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
171 set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
172 set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
173 set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
174 set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
175 set_property port_width 1 [get_debug_ports u_ila_0/clk]
176 connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
177 set_property PROBE_TYPE TRIGGER [get_debug_ports u_ila_0/probe0]
178 set_property port_width 1 [get_debug_ports u_ila_0/probe0]
179 connect_debug_port u_ila_0/probe0 [get_nets [list clk_IBUF]]
180 create_debug_port u_ila_0 probe
181 set_property PROBE_TYPE DATA [get_debug_ports u_ila_0/probe1]
182 set_property port_width 1 [get_debug_ports u_ila_0/probe1]
183 connect_debug_port u_ila_0/probe1 [get_nets [list MISO_OBUF]]
184 create_debug_port u_ila_0 probe
185 set_property PROBE_TYPE DATA [get_debug_ports u_ila_0/probe2]
186 set_property port_width 1 [get_debug_ports u_ila_0/probe2]
187 connect_debug_port u_ila_0/probe2 [get_nets [list MOSI_IBUF]]
188 create_debug_port u_ila_0 probe
189 set_property PROBE_TYPE DATA [get_debug_ports u_ila_0/probe3]
190 set_property port_width 1 [get_debug_ports u_ila_0/probe3]
191 connect_debug_port u_ila_0/probe3 [get_nets [list rst_n_IBUF]]
192 create_debug_port u_ila_0 probe
193 set_property PROBE_TYPE DATA [get_debug_ports u_ila_0/probe4]
194 set_property port_width 1 [get_debug_ports u_ila_0/probe4]
195 connect_debug_port u_ila_0/probe4 [get_nets [list SS_n_IBUF]]
196 set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
197 set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
198 set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
199 connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
200
```

Successful Bitstream Generation

