SPI Slave with Single Port RAM

By

Karim Yehia & Amr Khaira

RTL Design:

SPI Interface:

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

parameter IDLE = 0, WRITE = 1, CHK_CMD = 2, READ_ADD = 3, READ_DATA = 4;

(* fsm_encoding = "sequential" *)
reg [2:0] cs, ns;
reg [3:0] rx_counter;
reg [2:0] tx_counter;
reg address_read;

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

```
always @(*) begin
    case (cs)
        IDLE: begin
          if(!SS_n) begin
            ns = CHK_CMD;
          end
          else begin
            ns = IDLE;
          end
        end
        CHK_CMD: begin
          if(!SS_n && !MOSI) begin
            ns = WRITE;
          end
          else if(!SS_n && MOSI) begin
            if(address_read)
              ns = READ_DATA;
            else
              ns = READ_ADD;
          end
          else
            ns = IDLE;
        end
        WRITE: begin
          if(SS_n)
            ns = IDLE;
          else
            ns = WRITE;
        end
        READ_ADD: begin
          address_read = 1;
          if(SS_n)
            ns = IDLE;
          else
            ns = READ_ADD;
        end
        READ_DATA: begin
          address_read = 0;
          if(SS_n)
            ns = IDLE;
          else
            ns = READ_DATA;
        end
    endcase
end

```

```
// output logic
always @(posedge clk) begin
    if(cs == WRITE ) begin
    rx_counter <= rx_counter + 1;
    rx_data <= {rx_data[8:0], MOSI};
    if(rx_counter == 9) begin
      rx_valid <= 1;
      rx_counter <= 0;
    end
    else begin
      rx_valid <= 0;
    end
    end
    else if(cs == READ_ADD) begin
      rx_counter <= rx_counter + 1;
      rx_data <= {rx_data[8:0], MOSI};
      if(rx_counter == 9) begin
        rx_counter <= 0;
        rx_valid <= 1;
    end
    else begin
      rx_valid <= 0;
    end
    end
    else if(cs == READ_DATA) begin
    rx_counter <= rx_counter + 1;
    rx_data <= {rx_data[8:0], MOSI};
    if(rx_counter == 9) begin
      rx_counter <= 0;
      rx_valid <= 1;
    end
    else begin
      rx_valid <= 0;
    end

    if(tx_valid) begin
      MISO <= tx_data[7 - tx_counter];
      tx_counter <= tx_counter + 1;
    end
    else begin
        MISO <= 0;
        tx_counter <= 0;
    end
end

else begin
      rx_counter <= 0;
      tx_counter <= 0;
      rx_valid <= 0;
      MISO <= 0;
end
end


endmodule
```

RAM:

```
module synch_ram(clk, rst_n, din, rx_valid, dout, tx_valid);
input clk, rst_n, rx_valid;
input [9:0] din;
output reg tx_valid;
output reg [7:0] dout;

parameter MEM_DEPTH = 256, ADDR_SIZE = 8;

reg [ADDR_SIZE-1 :0] mem [MEM_DEPTH-1 :0];

reg [ADDR_SIZE - 1 : 0] wr_address, rd_address;
always @(posedge clk) begin
    if(!rst_n) begin
      dout <= 0;
      wr_address <= 0;
      rd_address <= 0;
      tx_valid <= 0;
    end
    else begin
        if(rx_valid) begin
            case ({din[9:8]})
                2'b00: begin
                  wr_address <= din[7:0];
                  tx_valid <= 0;
                end
                2'b01: begin
                  mem[wr_address] <= din[7:0];
                  tx_valid <= 0;
                end
                2'b10: begin
                  rd_address <= din[7:0];
                  tx_valid <= 0;
                end
                2'b11: begin
                  dout <= mem[rd_address];
                  tx_valid <= 1;
                end
            endcase
        end
    end
end
endmodule
```

SPI Wrapper:

```
module spi_wrapper(MOSI, MISO, SS_n, clk, rst_n);
input MOSI, SS_n, clk, rst_n;
output MISO;

wire [9:0] rx_data;
wire [7:0] tx_data;
wire rx_valid, tx_valid;

spi_slave spi(MOSI, SS_n, clk, rst_n, tx_data, tx_valid, MISO, rx_data, rx_valid);
synch_ram ram(clk, rst_n, rx_data, rx_valid, tx_data, tx_valid);
endmodule
```

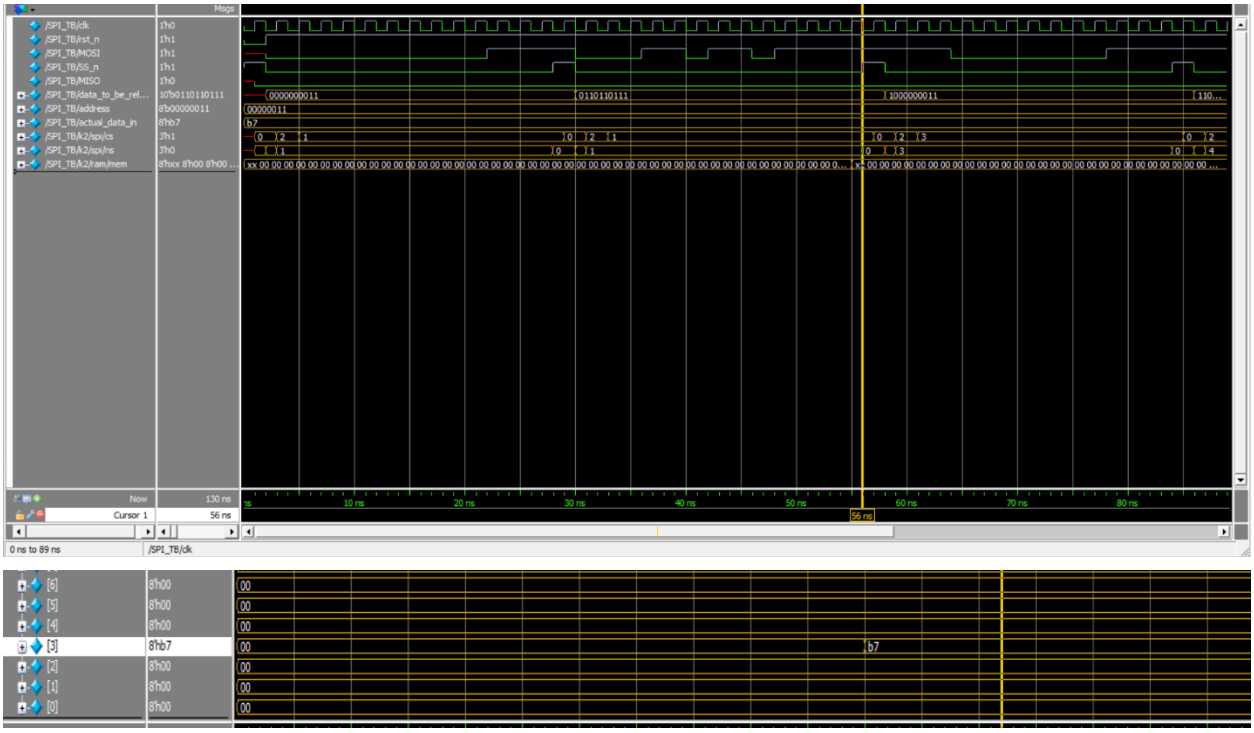
Testbench & Waveform:

Case 1 & 2: Writing data to address

```
module SPI_TB();
reg MOSI, SS_n, clk, rst_n;
wire MISO;
reg [9:0]data_to_be_relayed;//either the address or the actual data itself
reg [7:0]data_read_from_RAM;
reg [7:0]address=8'b0000011;
reg [7:0]actual_data_in=8'b10110111;
spi_wrapper k2(MOSI, MISO, SS_n, clk, rst_n);
initial begin
    clk=0;
    forever
    #1
    clk=~clk;
end
integer i;
initial begin
    /*Reset & Initiate Communication*/
    rst_n=0;
    SS_n=1;
    $readmemh("mem.dat", k2.ram.mem);
    @(negedge clk);
    rst_n=1;
    SS_n=0;
    /*Initiate Writing Address*/
    MOSI=0;
    data_to_be_relayed[9:8]=2'b00;
    data_to_be_relayed[7:0]=address;
    repeat(2) @(negedge clk); // one cycle cs goes from IDLE -> CHK_CMD, 2nd cycle: CHK_CMD -> WRITE
    for(i=9;i>=0;i=i-1)begin
        MOSI = data_to_be_relayed[i];
        @(negedge clk);
    end
    @(negedge clk);
    SS_n=1;
    @(negedge clk);
```

```
/*Initiate Writing Data*/
SS_n=0;
MOSI=0;
data_to_be_relayed[9:8]=2'b01;
data_to_be_relayed[7:0]=actual_data_in;
repeat(2) @(negedge clk); // one cycle cs goes from IDLE -> CHK_CMD, 2nd cycle: CHK_CMD -> WRITE
for(i=9;i>=0;i=i-1)begin
    MOSI = data_to_be_relayed[i];
    @(negedge clk);
end
@(negedge clk);
SS_n=1;
@(negedge clk);
```

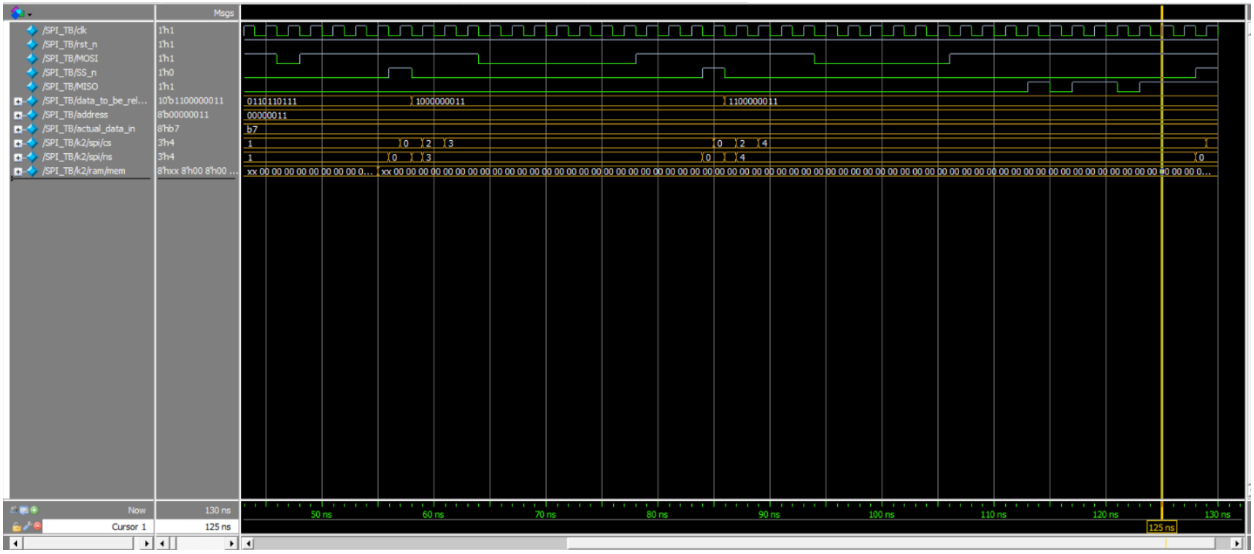
Waveform & Memory:



Case 3 & 4: Reading data from address

```
    /*Initiate Reading Address*/
    MOSI=1;
    SS_n=0;
    data_to_be_relayed[9:8]=2'b10;
    data_to_be_relayed[7:0]=address;
    repeat(2)@(negedge clk);
    for(i=9;i>=0;i=i-1)begin
        MOSI=data_to_be_relayed[i];
        @(negedge clk);
    end
    @(negedge clk);
    SS_n=1;
    @(negedge clk);
    /*Initiate Reading Data*/
    MOSI=1;
    SS_n=0;
    data_to_be_relayed[9:8]=2'b11;
    data_to_be_relayed[7:0]=address;
    repeat(2)@(negedge clk);
    for(i=9;i>=0;i=i-1)begin
        MOSI=data_to_be_relayed[i];
        @(negedge clk);
    end
    @(negedge clk);
    repeat(8) @(negedge clk);
    SS_n=1;
    @(negedge clk);
    $stop;
end
```

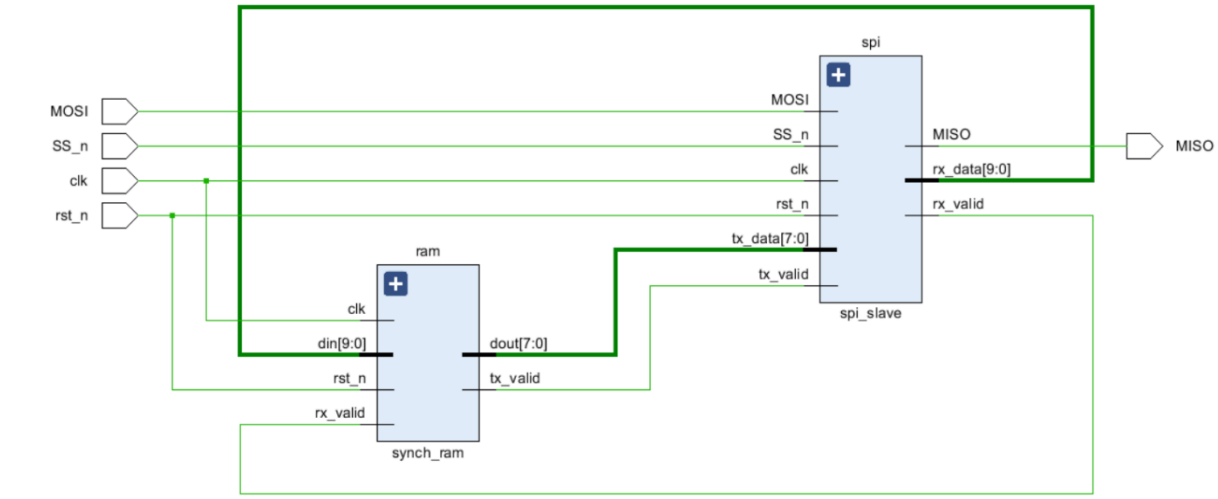
Waveform:



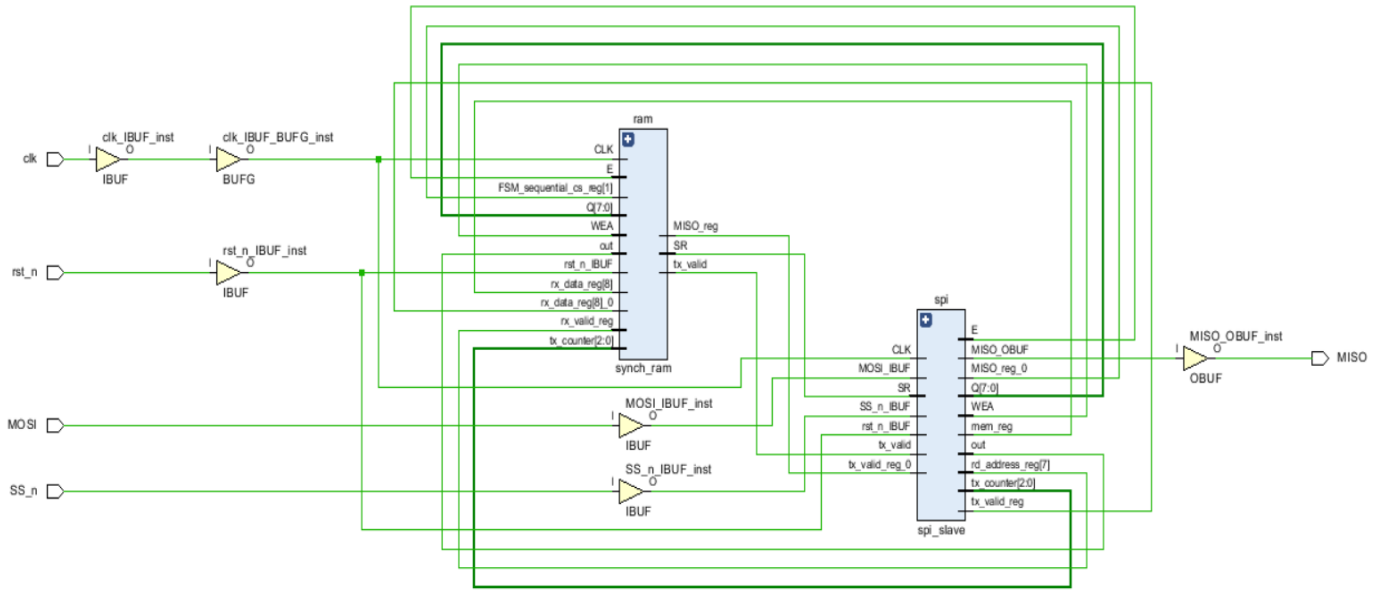
Testing Different Encoding for Best Timing

Binary Encoding:

Elaborated Design:



Synthesis Schematic:



Report Showing Encoding:

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

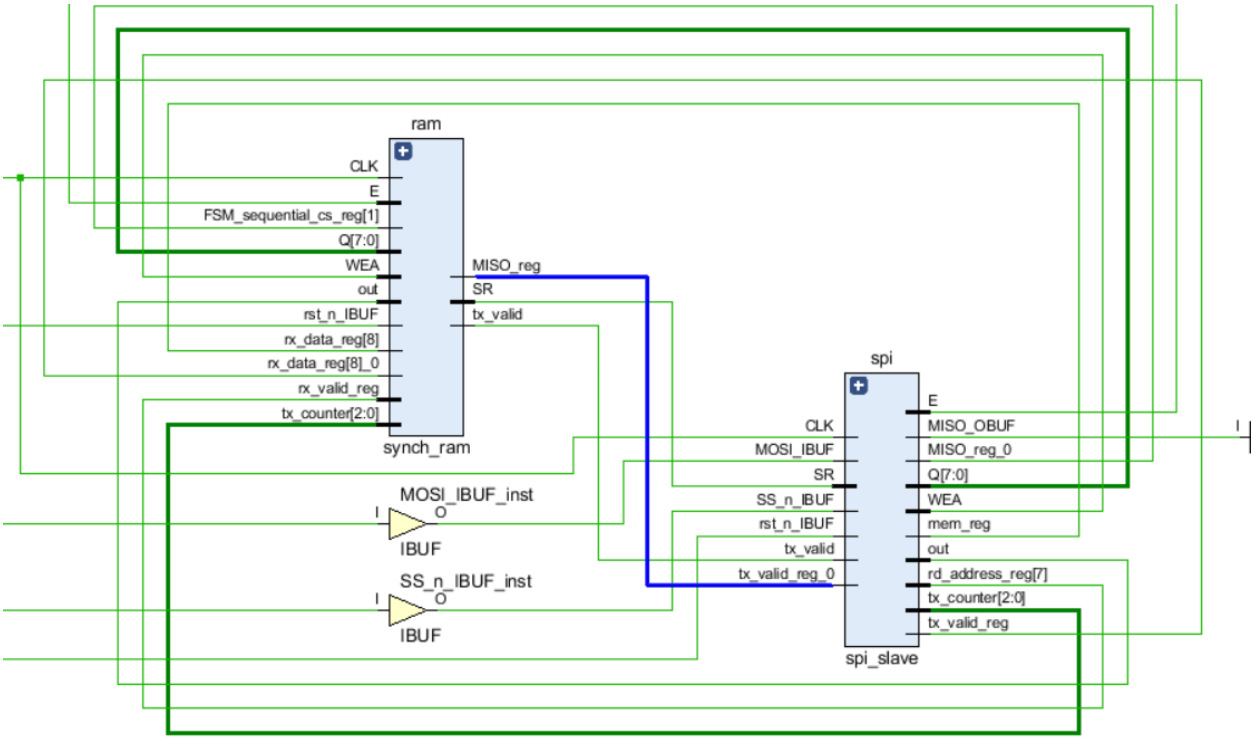
Timing Report:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.445 ns	Worst Hold Slack (WHS):	0.142 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:	93	Total Number of Endpoints:	93	Total Number of Endpoints:	42

All user specified timing constraints are met.

Critical Path



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	26	43	15	26	11	0.5	5	1
ram (synch_ram)	5	17	7	5	0	0.5	0	0
spi (spi_slave)	21	26	13	21	10	0	0	0

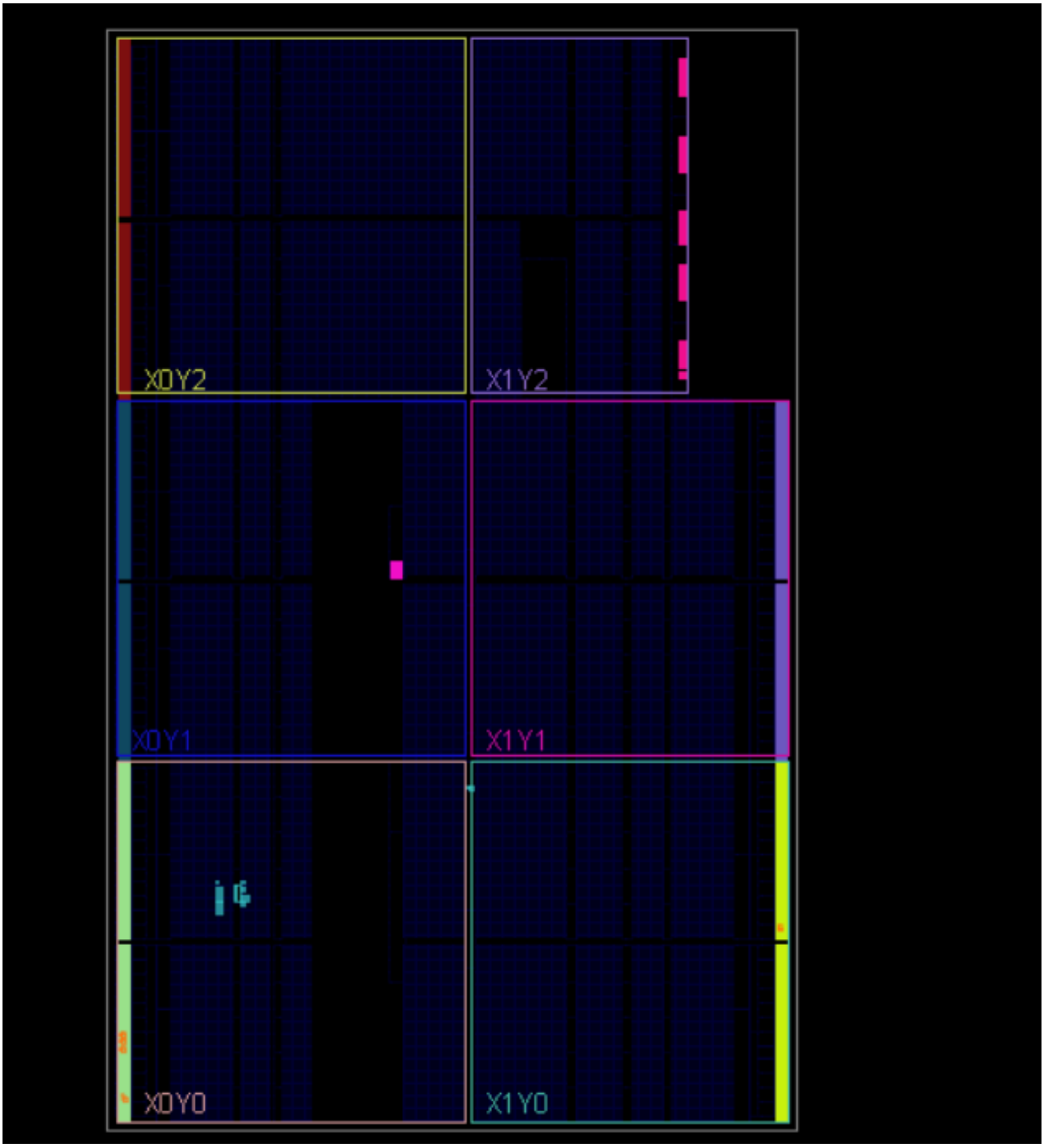
Implementation Timing Report:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.385 ns	Worst Hold Slack (WHS):	0.102 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:	94	Total Number of Endpoints:	94	Total Number of Endpoints:	42

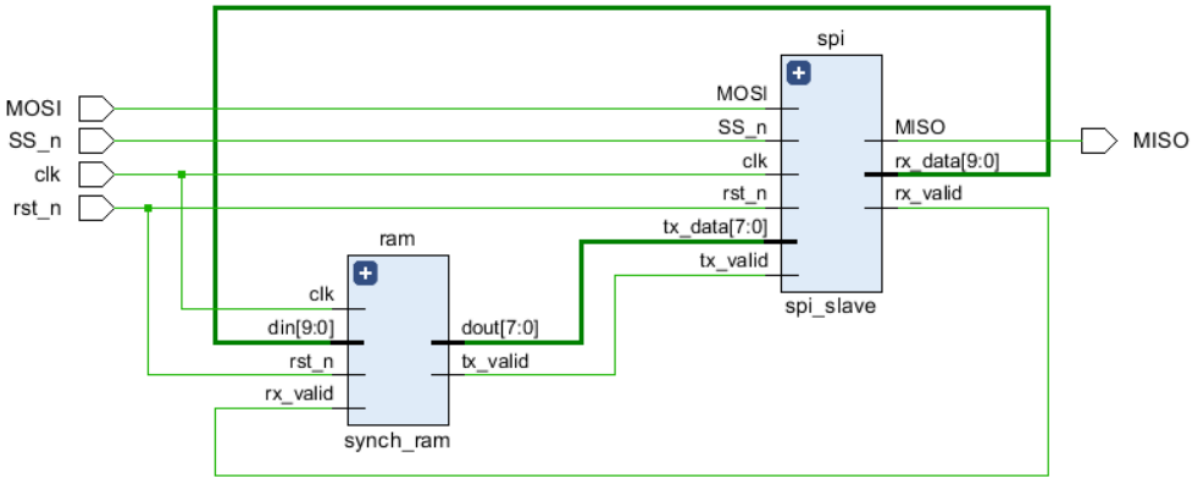
All user specified timing constraints are met.

FPGA Device:



Gray Encoding

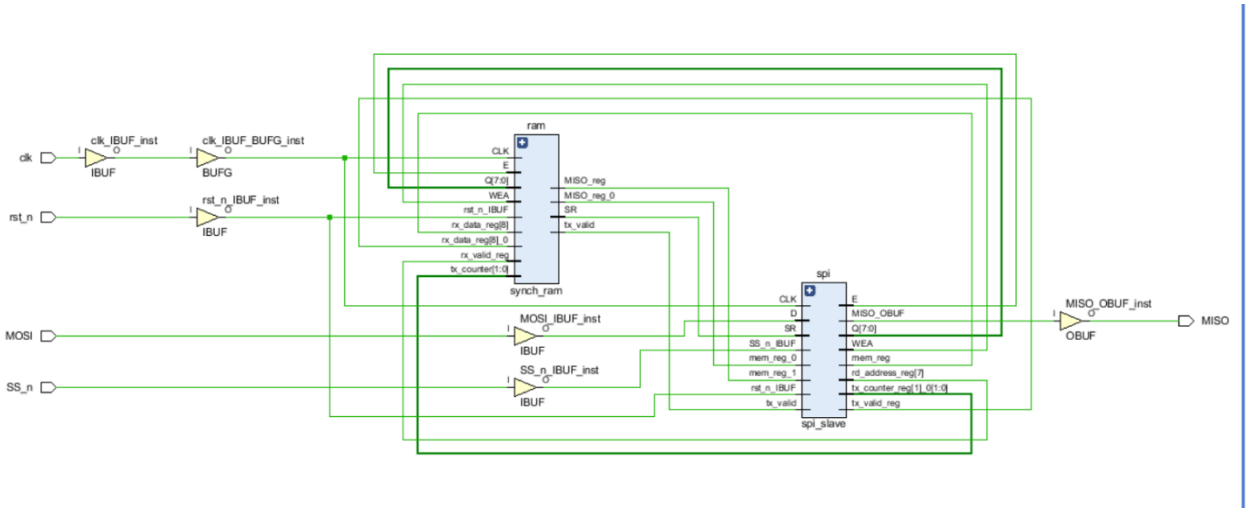
Elaborated Design:



Report Showing Encoding

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

Synthesis Schematic:



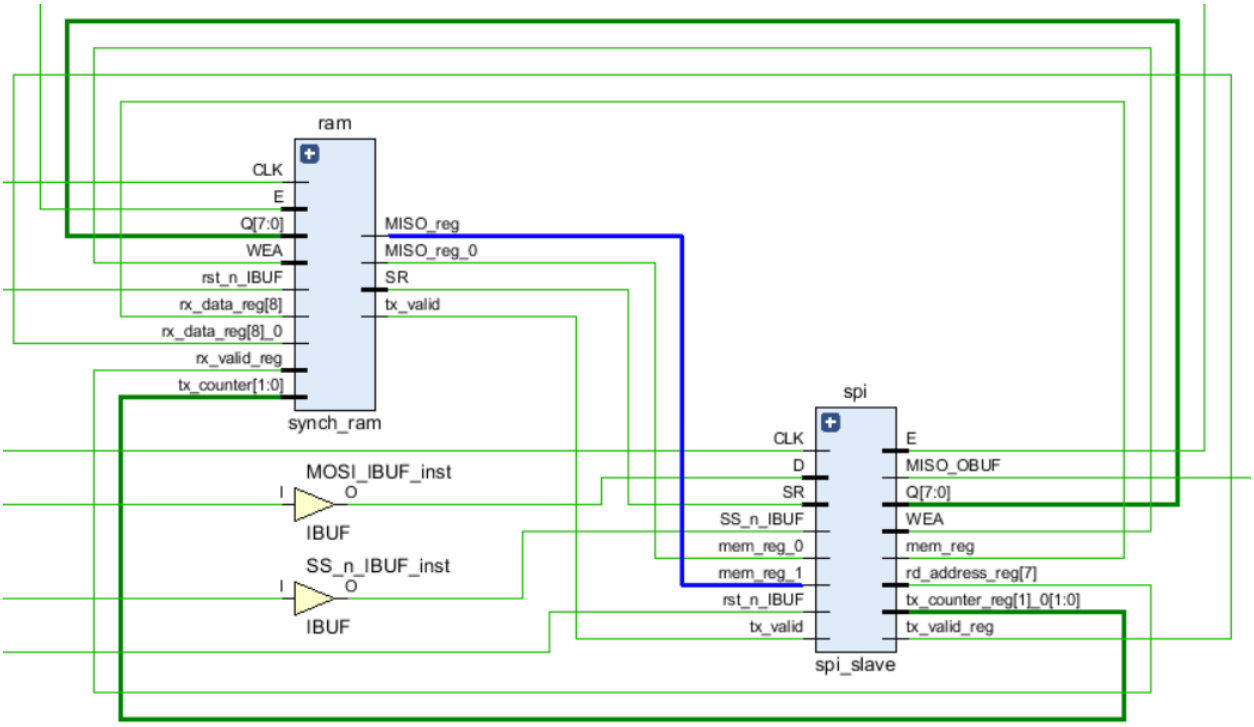
Timing Report:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.898 ns	Worst Hold Slack (WHS):	0.142 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:	93	Total Number of Endpoints:	93	Total Number of Endpoints:	42

All user specified timing constraints are met.

Critical Path:



Implementation Utilization Report:

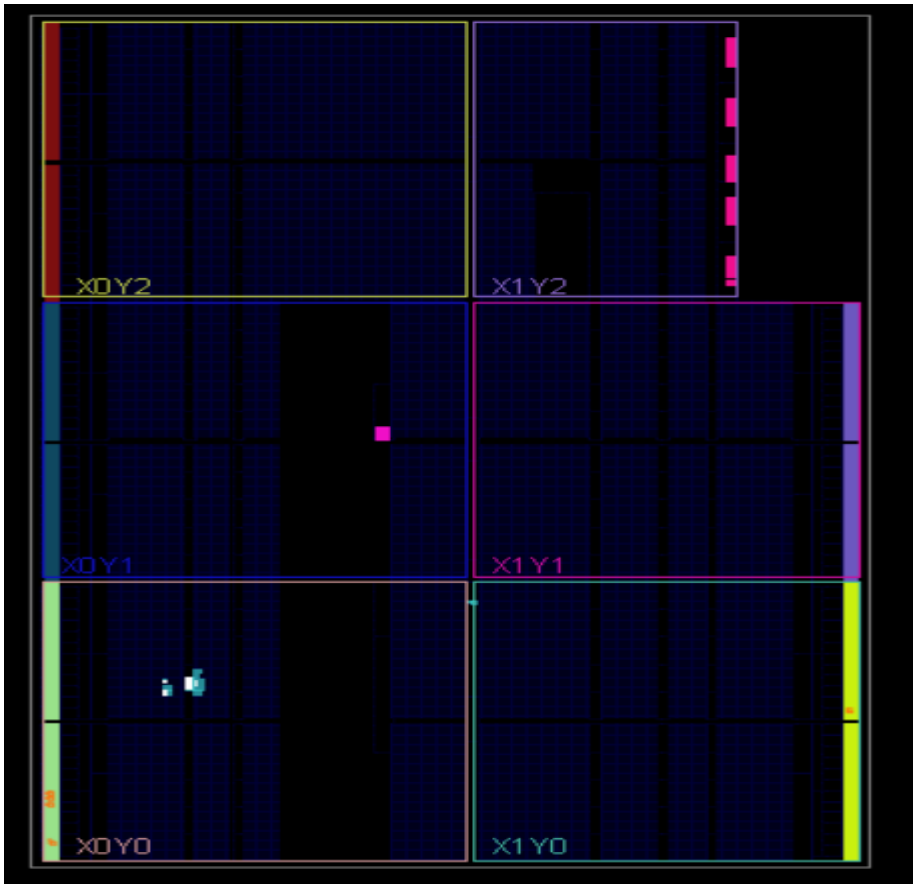
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N spi_wrapper	26	43	15	26	10	0.5	5	1
ram (synch_ram)	4	17	6	4	0	0.5	0	0
spi (spi_slave)	22	26	12	22	9	0	0	0

Implementation Timing Report:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.293 ns	Worst Hold Slack (WHS):	0.044 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:	94	Total Number of Endpoints:	94	Total Number of Endpoints:	42

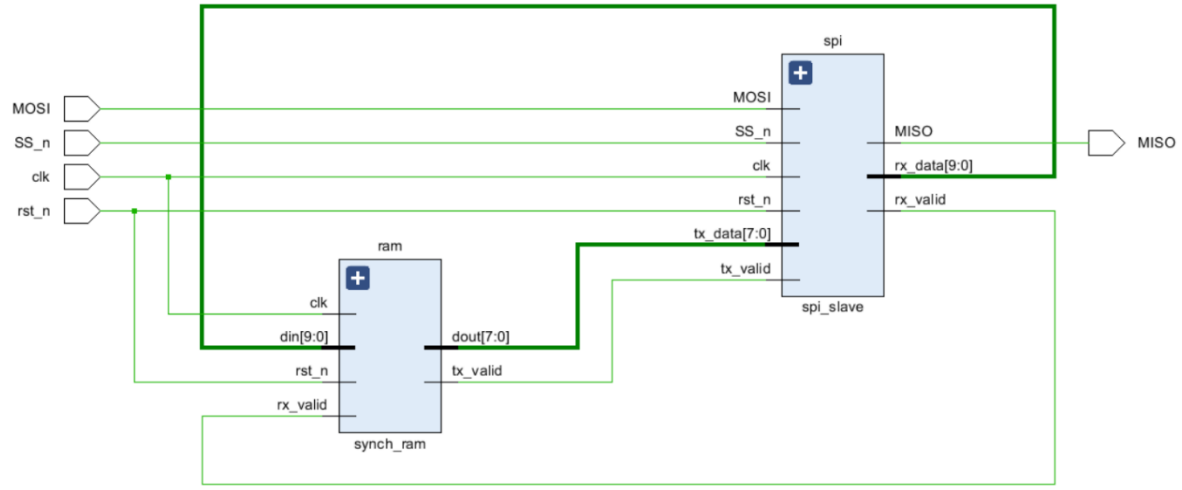
All user specified timing constraints are met.

FPGA Device:



One Hot Encoding

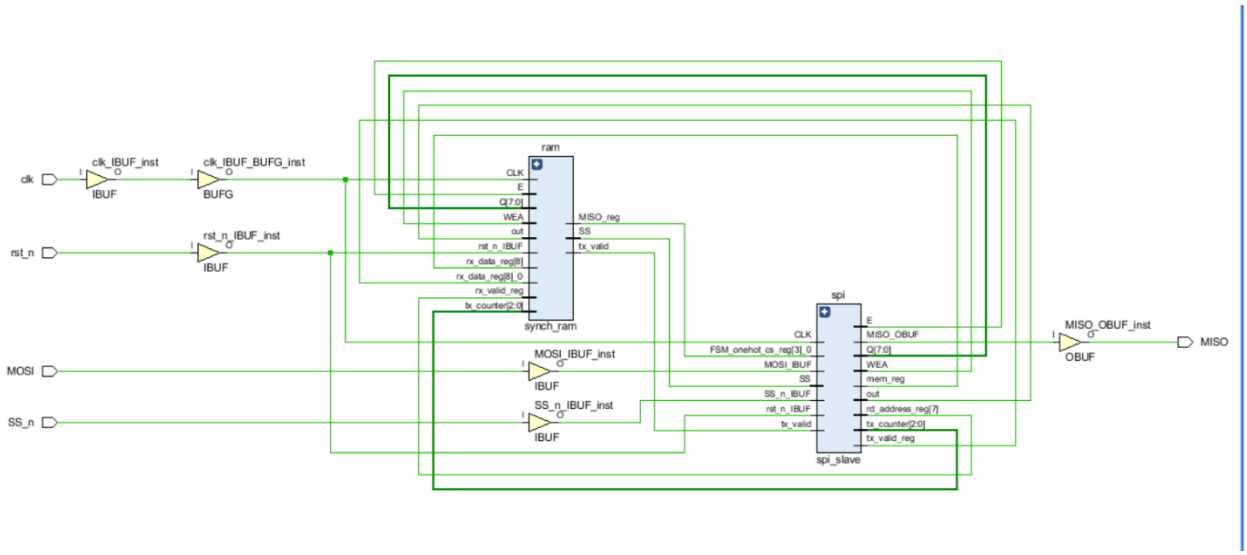
Elaborated Design:



Report Showing Encoding:

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

Synthesis Schematic:

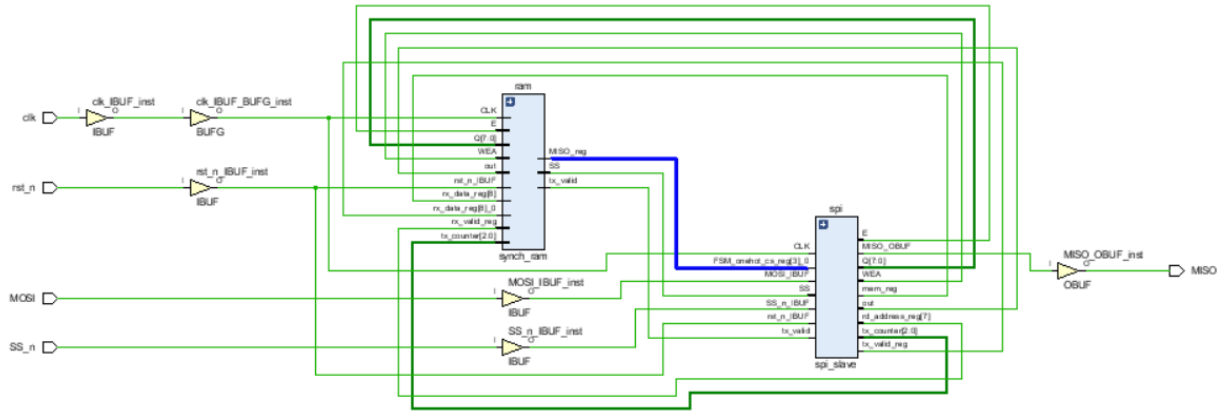


Timing Report:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.898 ns	Worst Hold Slack (WHS):	0.144 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:	93	Total Number of Endpoints:	93	Total Number of Endpoints:	44

All user specified timing constraints are met.

Critical Path:



Implementation Utilization Report:

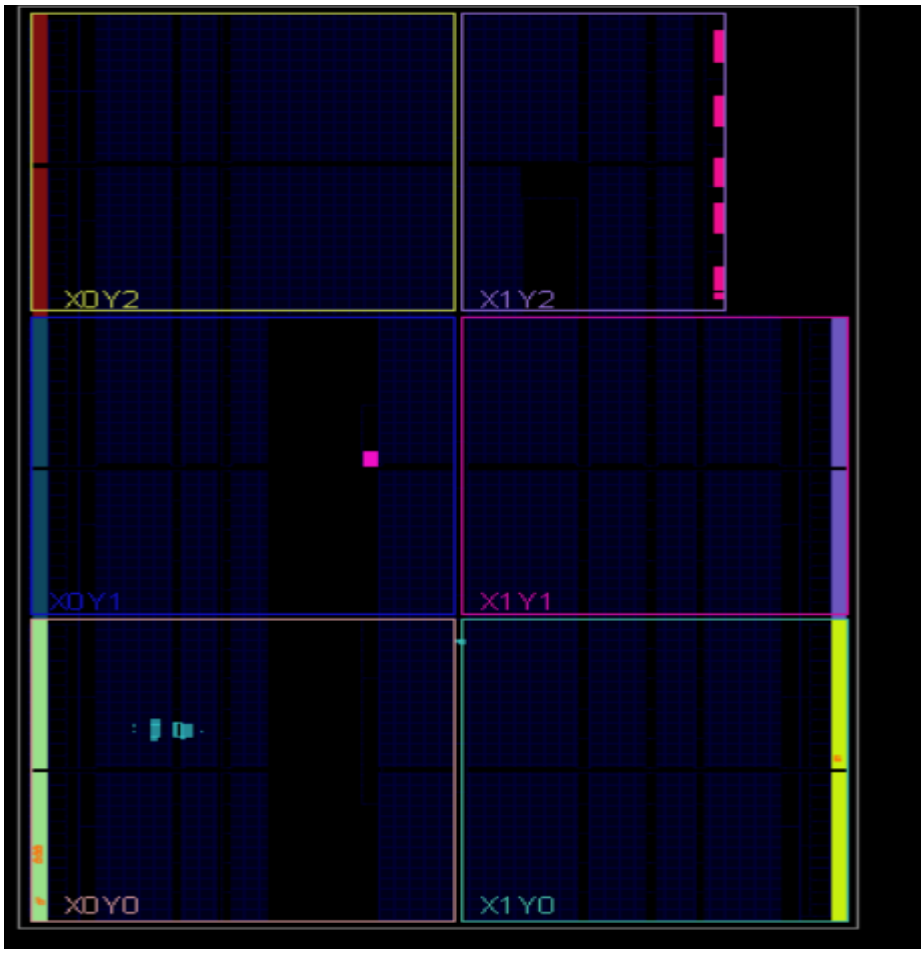
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N spi_wrapper	26	47	17	26	11	0.5	5	1
ram (synch_ram)	5	17	8	5	0	0.5	0	0
spi (spi_slave)	21	30	13	21	9	0	0	0

Implementation Timing Report:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.372 ns	Worst Hold Slack (WHS):	0.114 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:	94	Total Number of Endpoints:	94	Total Number of Endpoints:	44

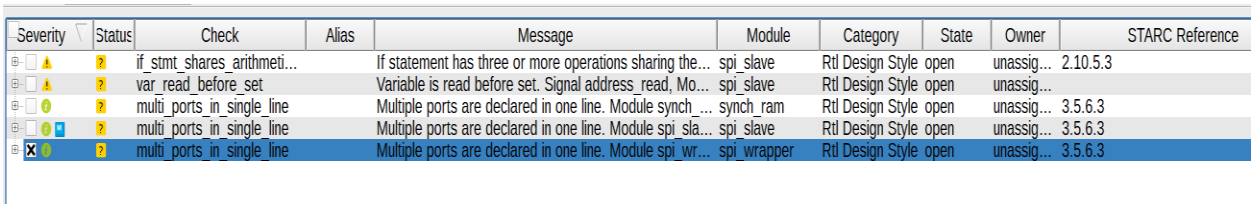
All user specified timing constraints are met.

FPGA Device:



Messages Tab:

Linting:



Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference
	?	if_stmt_shares_arithmeti...		If statement has three or more operations sharing the...	spi_slave	Rtl Design Style	open	unassig...	2.10.5.3
	?	var_read_before_set		Variable is read before set. Signal address_read, Mo...	spi_slave	Rtl Design Style	open	unassig...	
	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module synch_...	synch_ram	Rtl Design Style	open	unassig...	3.5.6.3
	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module spi_sla...	spi_slave	Rtl Design Style	open	unassig...	3.5.6.3
	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module spi_wr...	spi_wrapper	Rtl Design Style	open	unassig...	3.5.6.3

Encoding	Setup Slack	Hold Slack
Binary	5.385s	0.102s
Gray	5.293s	0.044s
One Hot	5.372s	0.114s

Binary encoding has higher setup slack while one hot encoding has higher hold time slack. We shall choose binary encoding for maximum frequency operation.