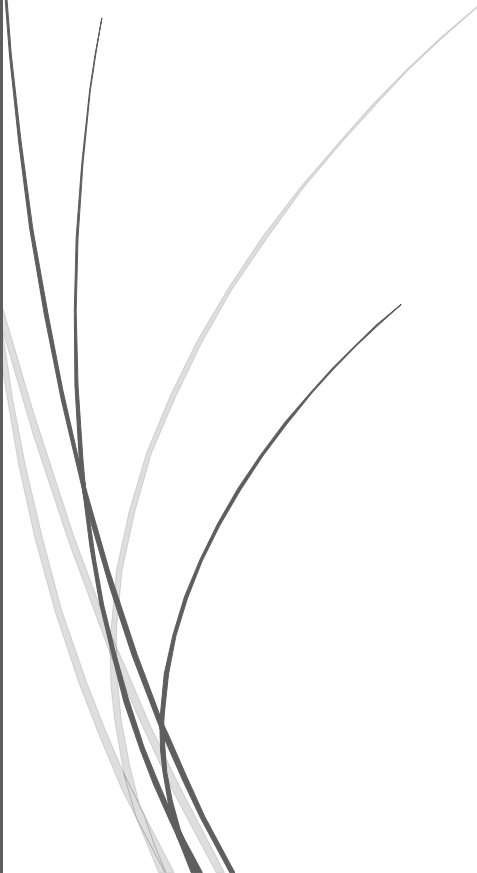




2025

# **SPI Slave with Single Port RAM**

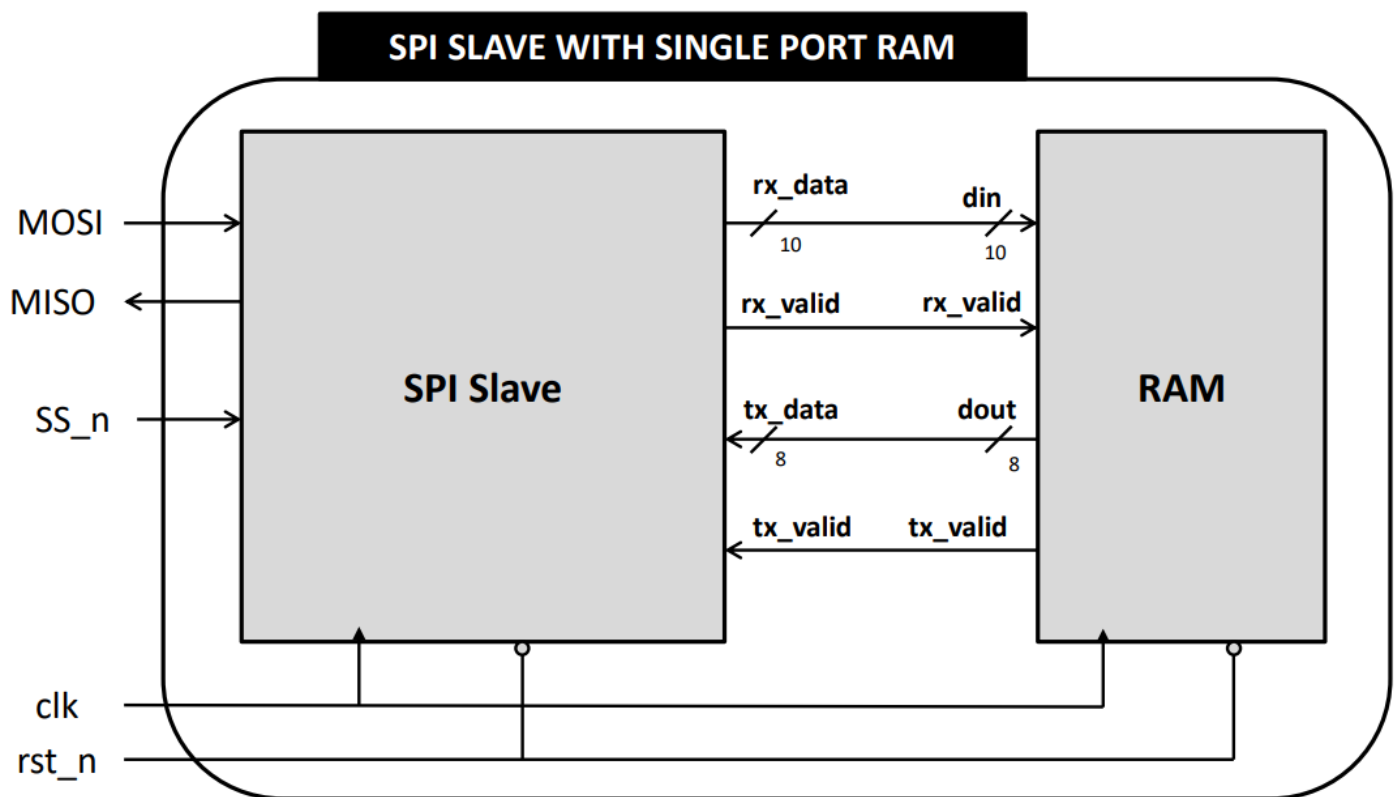


# CONTENTS

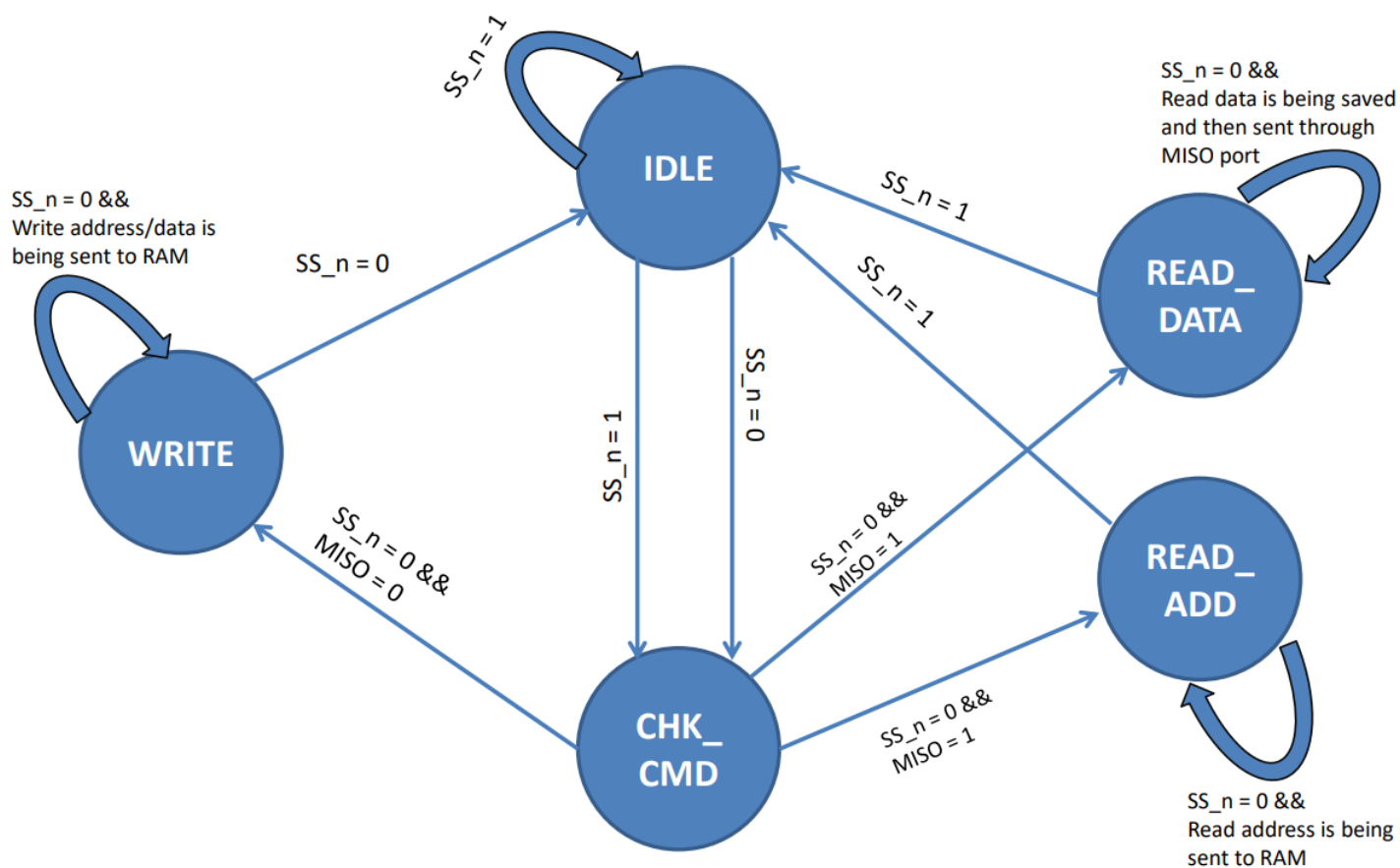
Project Description.....	2
SPI Transition Diagram.....	3
Single Port RAM.....	4
SPI slave design.....	5
SPI Wrapper .....	8
QuestaSim Snippets .....	9
QuestaLint Snippets.....	13
Vivado Snippets .....	14
Gray fsm .....	15
Implementation (gray) .....	17
One_hot fsm .....	18
Implementaion (one_hot) .....	20
Seq fsm .....	21
Implementaion (seq fsm).....	22
Johnson fsm .....	23
Implementaion (Johnson) .....	25

# SPI Slave with Single Port RAM

- Create a constraint file where the rst\_n , SS\_n & MOSI are connected to 3 switches, and the MISO to a led.
- The SPI slave implementation is done using FSM, we shall try out three different encoding (gray, one\_hot or seq)
- We wish to operate at the highest frequency possible and so you shall choose the encoding based on the best timing report that gives the highest setup/hold slack after implementation.
- After choosing the best encoding, add a debug core such that all internals (MISO, MOSI, SS\_n, rst\_n & clk) can be analyzed and then generate a bitstream file.



## SPI slave Transition Diagram.



# Single Port Ram

```
module RAM(din, clk, rst_n, rx_vaild, dout, tx_valid);
```

```
parameter MEM_DEPTH = 256;
```

```
parameter ADDR_SIZE = 8;
```

```
input [9:0] din;
```

```
input clk, rst_n, rx_vaild;
```

```
output reg [7:0] dout;
```

```
output reg tx_valid;
```

```
reg [7:0] mem [MEM_DEPTH-1:0];
```

```
reg [7:0] address;
```

```
always @(posedge clk) begin
```

```
    if (!rst_n)begin
```

```
        dout <= 0;
```

```
        tx_valid <= 0;
```

```
        address<=0;
```

```
    end
```

```
    else begin
```

```
        case(din[9:8])
```

```
            2'b00: begin
```

```
                if(rx_vaild)begin
```

```
                    address <= din[7:0];
```

```
                    tx_valid <= 0;
```

```
                end
```

```
            end
```

```
            2'b01: begin
```

```
                if(rx_vaild)begin
```

```
                    mem[address] <= din[7:0];
```

```
                    tx_valid <= 0;
```

```
                end
```

```
            end
```

```
            2'b10:begin
```

```
                if(rx_vaild)begin
```

```
                    address <= din[7:0];
```

```
                end
```

```
                tx_valid <= 1;
```

```
            end
```

```
            2'b11: begin
```

```
                dout <= mem[address];
```

```
                tx_valid <= 1;
```

```
            end
```

```
        endcase
```

```
    end
```

```
end
```

```
endmodule
```

## SPI slave

```
module SPI_slave(  
    input [7:0] tx_data,  
    input ss_n, rst_n, clk, tx_valid, MOSI,  
    output reg [9:0] rx_data,  
    output reg rx_valid, MISO  
);  
    reg check_read=0;  
  
    //counter declaration  
    reg[3:0] counter;  
  
    //flip_flops  
    reg[2:0] cs, ns;  
  
    //states assignment  
    parameter IDLE      = 3'b000,  
               CHK_CMD   = 3'b001,  
               WRITE     = 3'b010,  
               READ_DATA  = 3'b011,  
               READ_ADD   = 3'b100;  
  
    //Dff model  
    always @(posedge clk)  
        if(!rst_n)begin  
            cs <= IDLE;  
  
        end  
        else begin  
            cs <= ns;  
        end  
  
    //next state logic  
    always@(*)begin  
        case(cs)  
            IDLE: begin  
                if(ss_n == 1)  
                    ns = IDLE;  
                else  
                    ns = CHK_CMD;  
            end  
            CHK_CMD: begin  
                case({check_read, MOSI})  
                    2'b00,2'b10:  
                        ns = WRITE;  
                    2'b01:  

```

```

        ns = READ_ADD;
    2'b11:
        ns = READ_DATA;
    endcase
end
WRITE: begin
    if(ss_n==0)
        ns = WRITE;
    else
        ns = IDLE;
    end
READ_ADD: begin
    if(ss_n==0)
        ns = READ_ADD;
    else
        ns = IDLE;
    end
READ_DATA: begin
    if(ss_n==0)
        ns = READ_DATA;
    else
        ns = IDLE;
    end
default: ns = IDLE;
endcase
end

```

```

//output logic
always@(posedge clk)begin
    if (!rst_n)begin
        rx_data <= 0;
        rx_valid <= 0;
        check_read <= 0;
        MISO <= 0;
        counter <= 0;
    end
    else begin
        case(cs)
            IDLE:
                begin
                    rx_data <= 0;
                    rx_valid<=0;

```

```

        rx_valid<=0;
        MISO <= 0;
    end

    CHK_CMD:
    begin
        rx_data <= 0;
        rx_valid <= 0;
        MISO <= 0;
    end

    WRITE: begin
        rx_data <= {rx_data[8:0],MOSI};
        counter <= counter+1;
        if(counter == 10)begin
            rx_valid <= 1;
            counter <= 0;
        end
        else
            rx_valid <= 0;
        end
    end

    READ_ADD: begin
        rx_data <= {rx_data[8:0],MOSI};
        counter <= counter+1;
        if(counter == 10)begin
            rx_valid <= 1;
            counter <= 0;
            check_read <= 1; /// flag to know if it's ( address or data )
        end
        else
            rx_valid <= 0;
        end
    end

    READ_DATA: begin
        rx_data <= {rx_data[8:0],MOSI};
        counter <= counter+1;
        if(counter == 10)begin
            rx_valid <= 1;
            counter <= 0;
            check_read <= 1; /// flag to know if it's ( address or data )
        end
        else
            rx_valid <= 0;
        end

        if((tx_valid==1)&&(counter<7||counter==7))
            MISO <= tx_data[7-counter];
        else
            MISO<=0;
        end
    end
endcase
end
end

endmodule

```

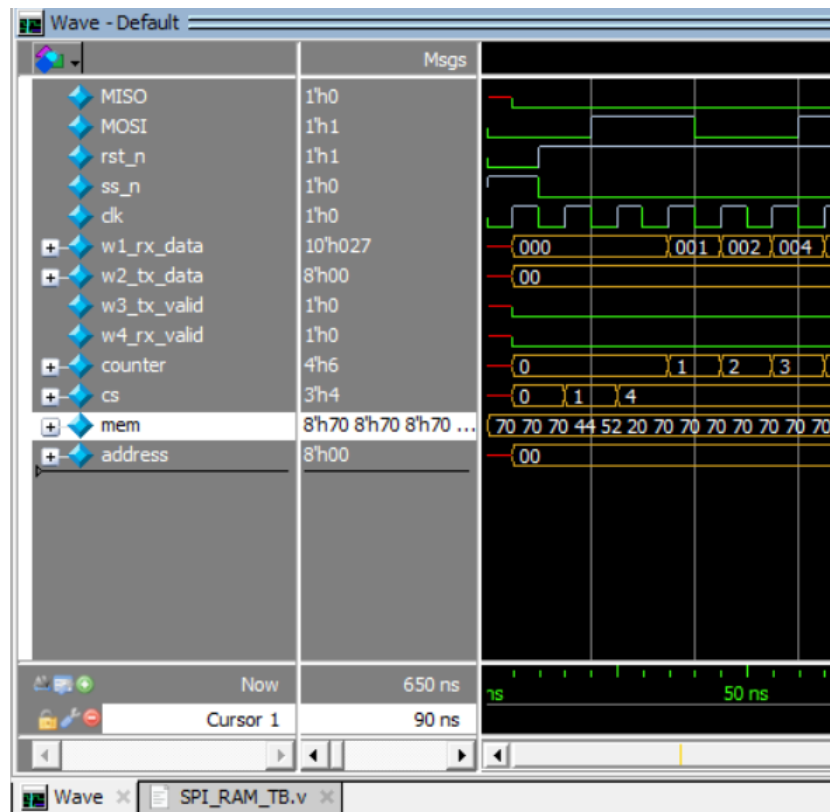


## SPI Wrapper

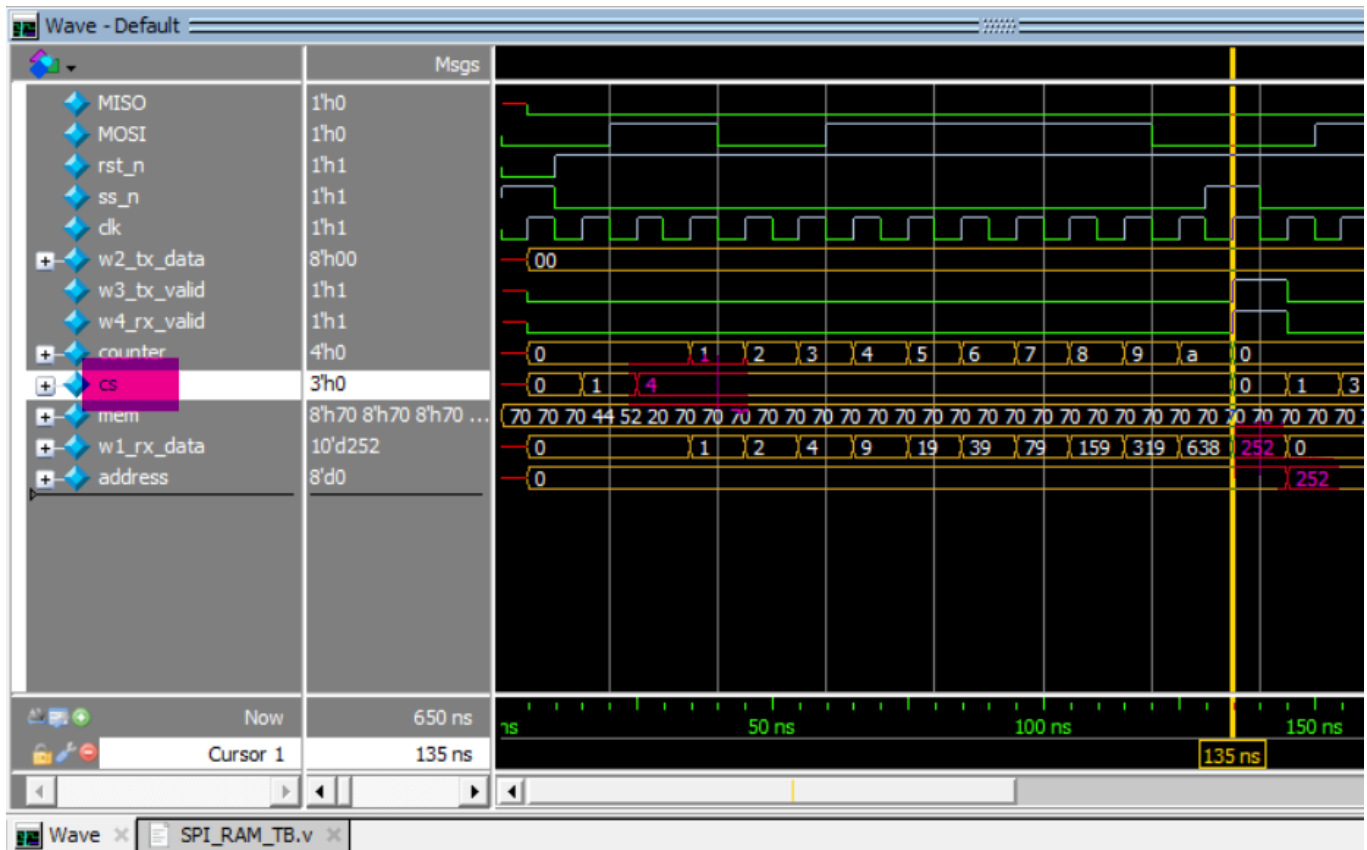
```
1 module SPI_w_RAM #(
2     parameter MEM_DEPTH = 256,
3     parameter ADDR_SIZE = 8
4 )
5     output MISO,
6     input MOSI,
7     input rst_n , ss_n , clk
8 );
9
10 wire[9:0]w1_rx_data;
11 wire[7:0]w2_tx_data;
12 wire w3_tx_valid , w4_rx_valid;
13
14 //connecting modules
15
16 SPI_slave S1(
17     .tx_data(w2_tx_data),
18     .ss_n(ss_n),
19     .rst_n(rst_n),
20     .clk(clk),
21     .tx_valid(w3_tx_valid),
22     .MOSI(MOSI),
23     .rx_data(w1_rx_data),
24     .rx_valid(w4_rx_valid),
25     .MISO(MISO)
26 );
27
28 RAM #(
29     .MEM_DEPTH(MEM_DEPTH),
30     .ADDR_SIZE(ADDR_SIZE)
31 )R1(
32     .din(w1_rx_data),
33     .clk(clk),
34     .rst_n(rst_n),
35     .rx_vaild(w4_rx_valid),
36     .dout(w2_tx_data),
37     .tx_valid(w3_tx_valid)
38 );
39
40 endmodule
```

# Snippets from the waveforms captured from QuestaSim

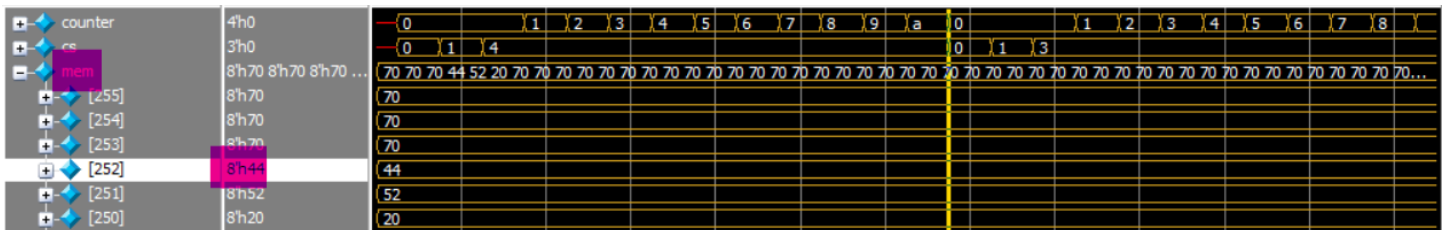
## 1\_testing synchronous reset



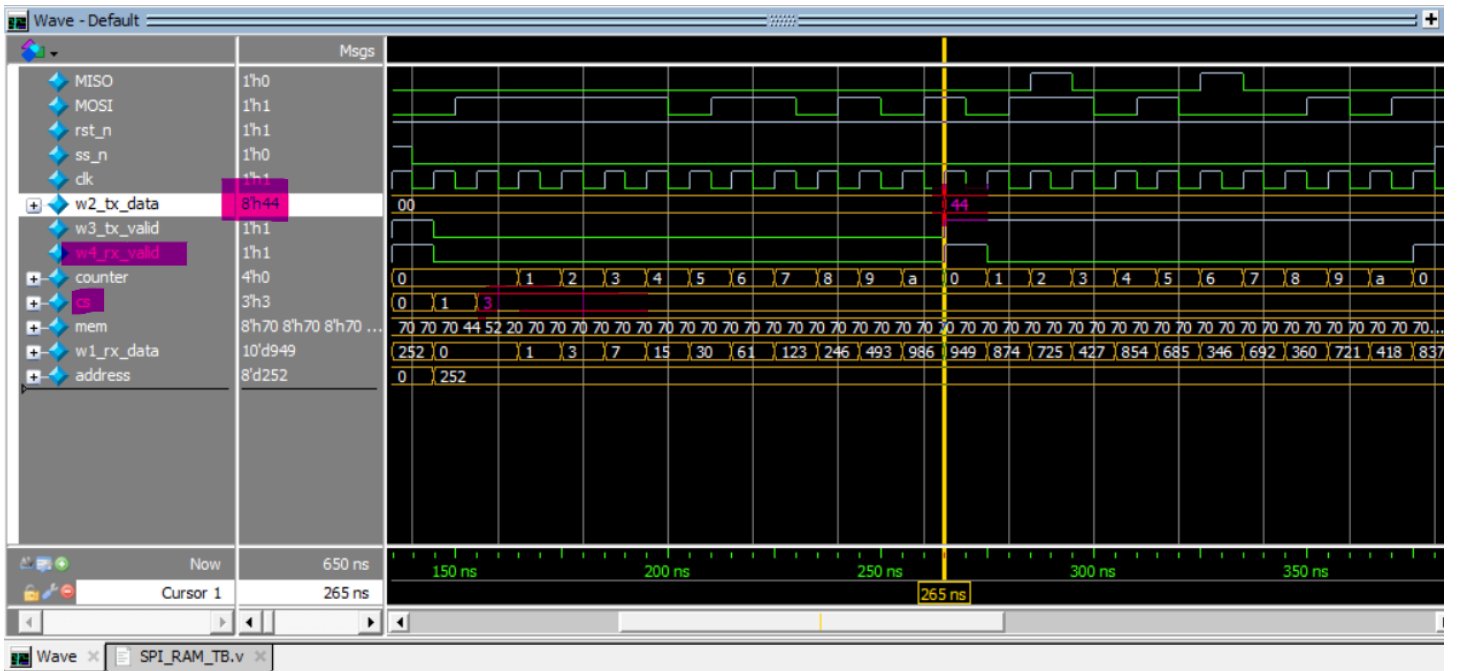
## 2\_testing read address



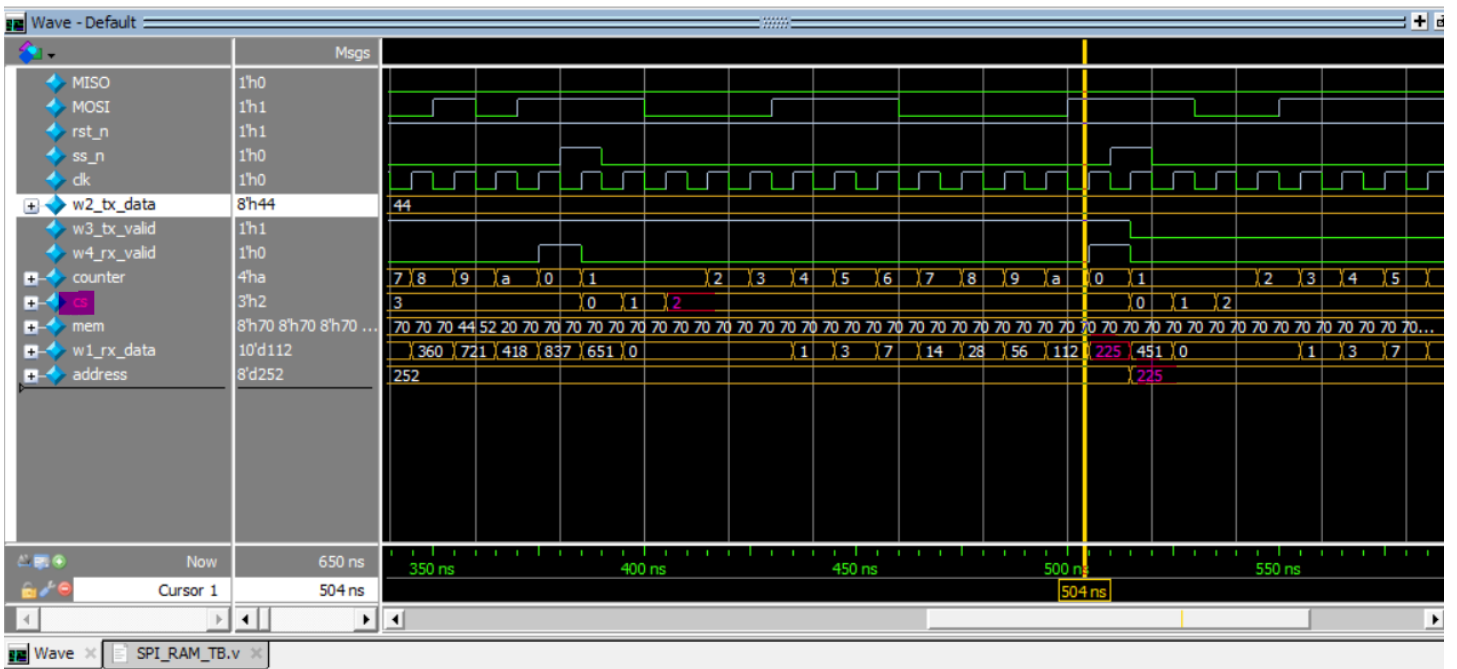
## 3\_what was in the address



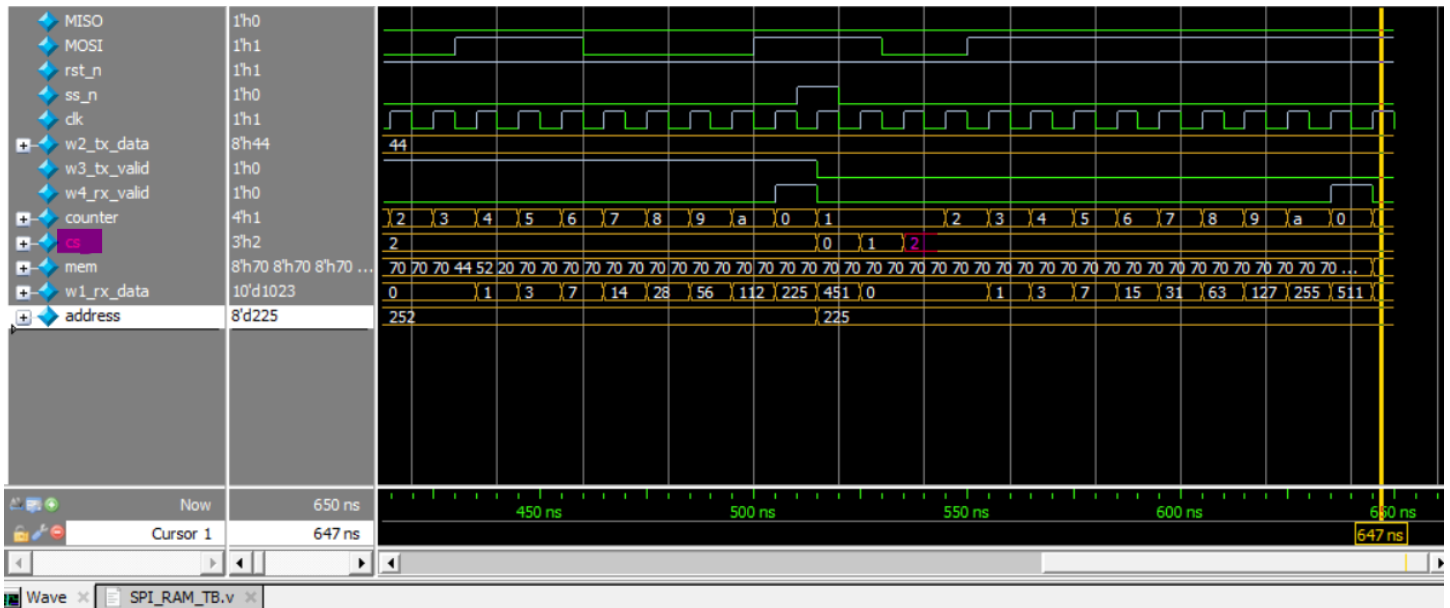
#### 4\_testing read data



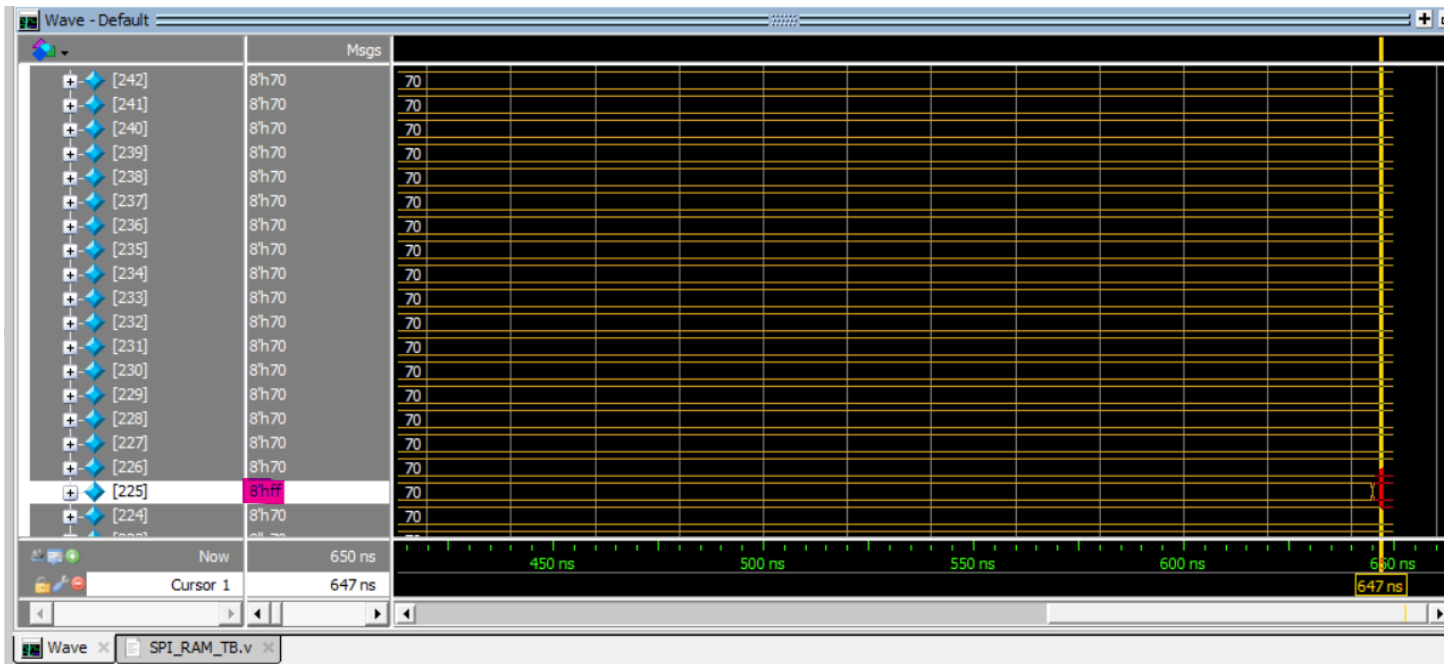
## 5\_test write address



## 6\_testing write data



## 7\_what we wrote in TB is written in the address we determined



## Questa Lint

The screenshot displays the Questa Lint 2021.1 interface. The main window shows a Verilog code file named `SPI_w_RAM.v` with the following code:

```

8
9
10
11 wire[9:0]w1_rx_data;
12 wire[7:0]w2_tx_data;
13 wire w3_tx_valid , w4_rx_valid;
14
15 //connecting modules
16
17 SPI_slave S1(
18     .tx_data(w2_tx_data),
19     .ss_n(ss_n),
20     .rst_n(rst_n),
21     .clk(clk),
22     .tx_valid(w3_tx_valid),
23     .MOSI(MOSI),
24     .rx_data(w1_rx_data),
25     .rx_valid(w4_rx_valid),
26     .MISO(MISO)
27 );
28
29 RAM #(
30     .MEM_DEPTH(MEM_DEPTH),
31     .ADDR_SIZE(ADDR_SIZE)
32 )R1(
33     .din(w1_rx_data),
34     .clk(clk),
35     .rst_n(rst_n),
36     .rx_vaild(w4_rx_valid),
37     .dout(w2_tx_data),
38     .tx_valid(w3_tx_valid)
39 );
40
41 endmodule

```

The Lint Summary window on the right shows the following data:

Name	Count
Open(uninspected, pending, bug)	5 (6)
Info	5 (6)

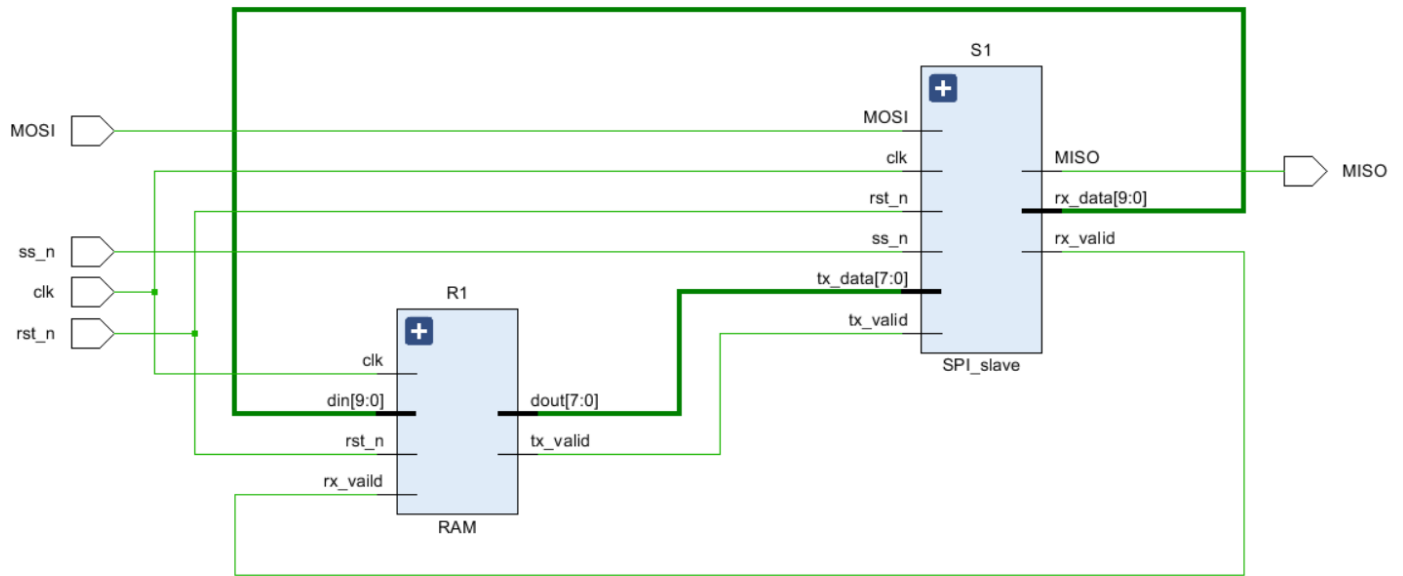
The Lint Checks window at the bottom shows a list of lint checks with the following columns: Severity, Status, Check, Alias, Message, Module, and Category.

Severity	Status	Check	Alias	Message	Module	Category
Warning	Fixed	parameter_name_duplicate		Same parameter name is used in more than one module. Parameter MEM_DEPTH, Total count 2. First module: Module RAM, File C:/Users/ABDOU/Desktop/Que...	RAM	Nomenclature Style
Warning	Fixed	parameter_name_duplicate		Same parameter name is used in more than one module. Parameter ADDR_SIZE, Total count 2. First module: Module RAM, File C:/Users/ABDOU/Desktop/Quest...	RAM	Nomenclature Style
Warning	Fixed	multi_ports_in_single_line		Multiple ports are declared in one line. Module SPI_slave, File C:/Users/ABDOU/Desktop/Questa-20250315T193333Z-001/Questa/SPI_slave.v, Line 3.	SPI_slave	Rtl Design Style
Warning	Fixed	multi_ports_in_single_line		Multiple ports are declared in one line. Module SPI_w_RAM, File C:/Users/ABDOU/Desktop/Questa-20250315T193333Z-001/Questa/SPI_w_RAM.v, Line 7.	SPI_w_RAM	Rtl Design Style
Warning	Fixed	multi_ports_in_single_line		Multiple ports are declared in one line. Module RAM, File C:/Users/ABDOU/Desktop/Questa-20250315T193333Z-001/Questa/memory.v, Line 7.	RAM	Rtl Design Style

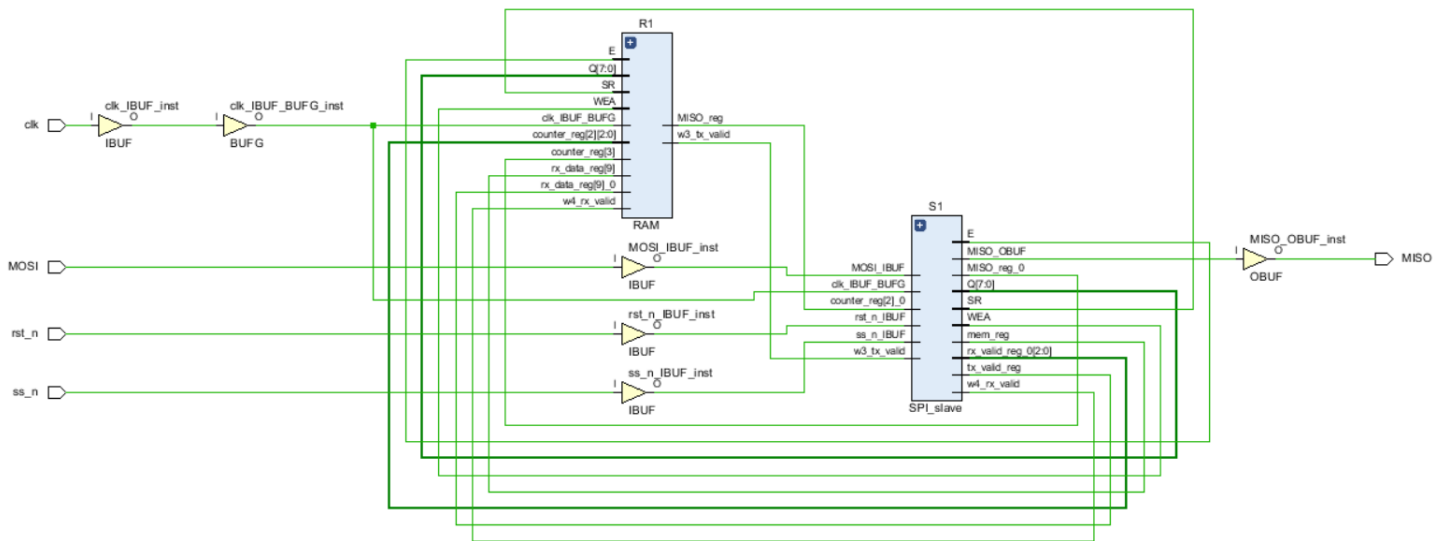
## VIVADO

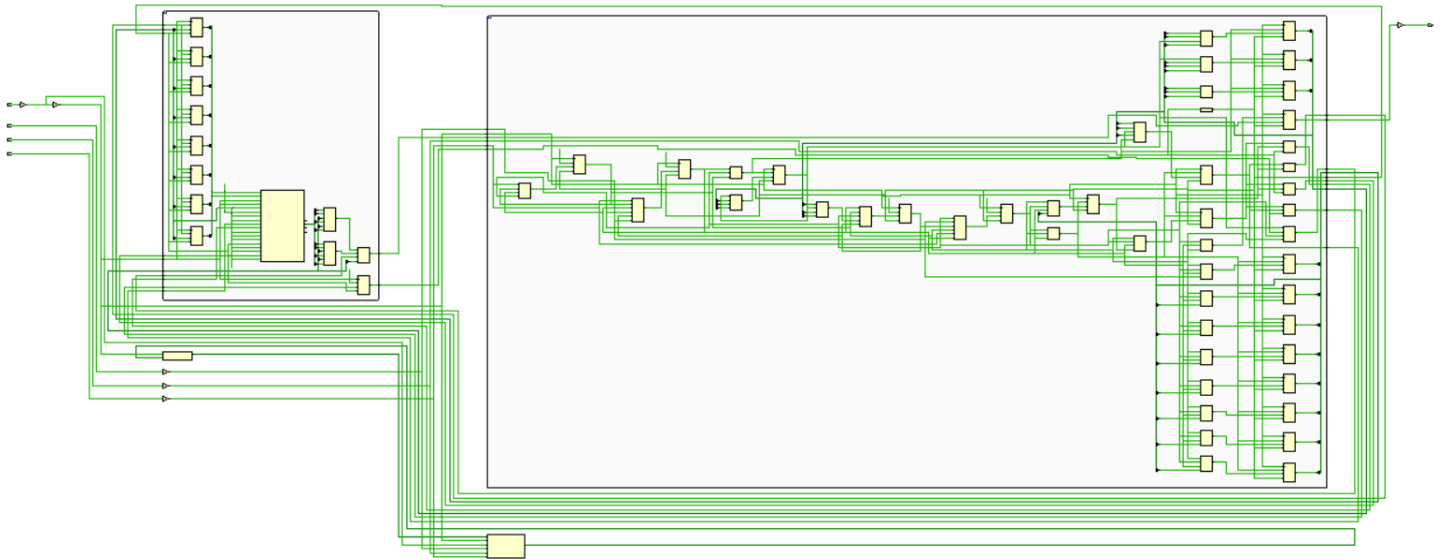
(\* fsm\_encoding = "gray" \*)

## Schematic elaboration



## Schematic Synthesis





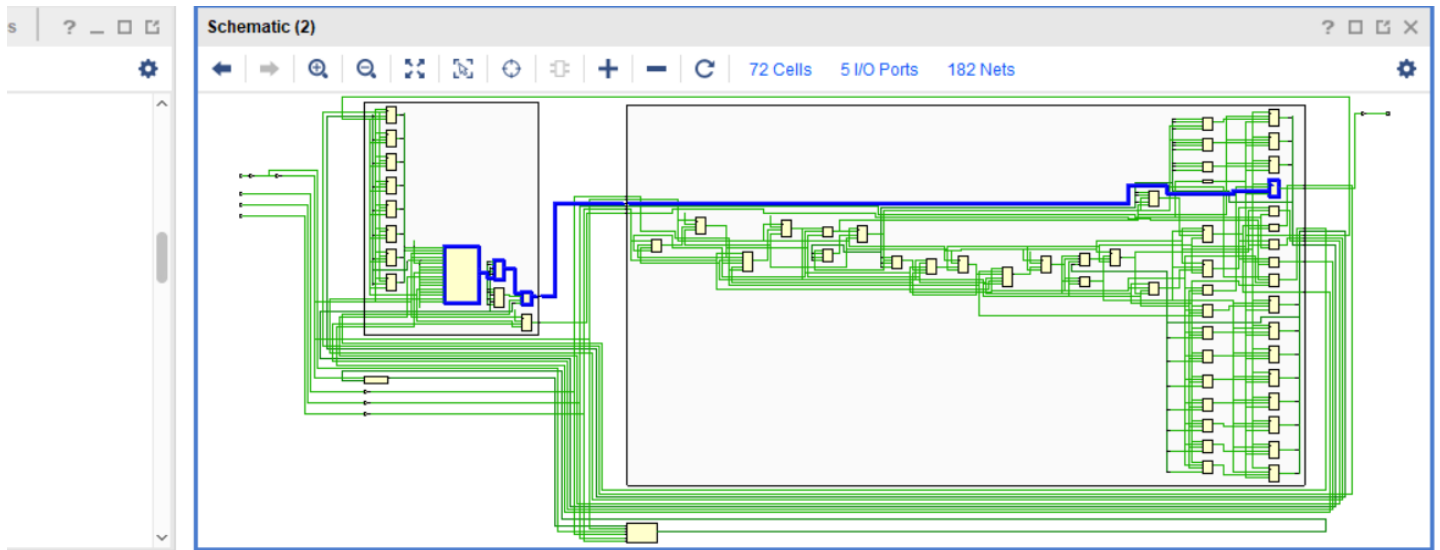
### Encoding from report

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

### Timing report

Design Runs			Timing		
Design Timing Summary					
Setup		Hold	Pulse Width		
Worst Negative Slack (WNS):		Worst Hold Slack (WHS):	Worst Pulse Width Slack (WPWS):		
Total Negative Slack (TNS):		Total Hold Slack (THS):	Total Pulse Width Negative Slack (TPWS):		
Number of Failing Endpoints:		Number of Failing Endpoints:	Number of Failing Endpoints:		
5.445 ns		0.142 ns	4.500 ns		
0.000 ns		0.000 ns	0.000 ns		
0		0	0		

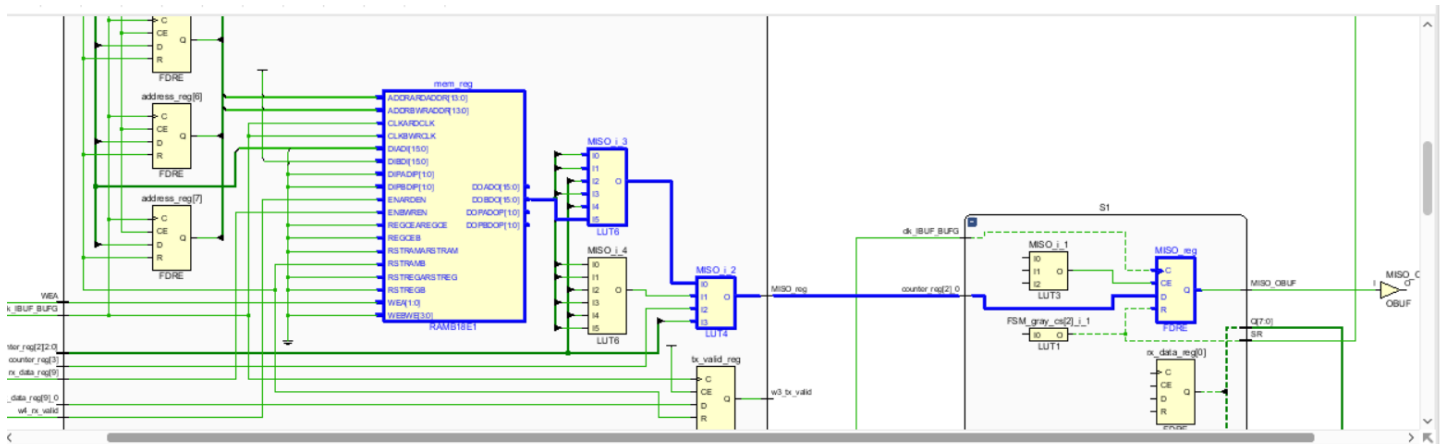




ports Design Runs Timing x

Intra-Clock Paths - sys\_clk\_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	5.445	2	3	1	R1/mem_reg/CLKBWRCLK	S1/MISO_reg/D	4.404	2.702	1.702	10.0



Tcl Console Messages Log Reports Design Runs Timing x

Intra-Clock Paths - sys\_clk\_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	5.445	2	3	1	R1/mem_reg/CLKBWRCLK	S1/MISO_reg/D	4.404	2.702	1.702	
Path 2	6.978	1	2	3	S1/rx_data_reg[9]/C	R1/mem_reg/WEA[0]	2.310	0.751	1.559	

## Implementation

Hierarchy									
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	
▼ SPI_w_RAM	28	29	9	28	17	0.5	5	1	
R1 (RAM)	3	9	3	3	0	0.5	0	0	
S1 (SPI_slave)	25	20	9	25	16	0	0	0	

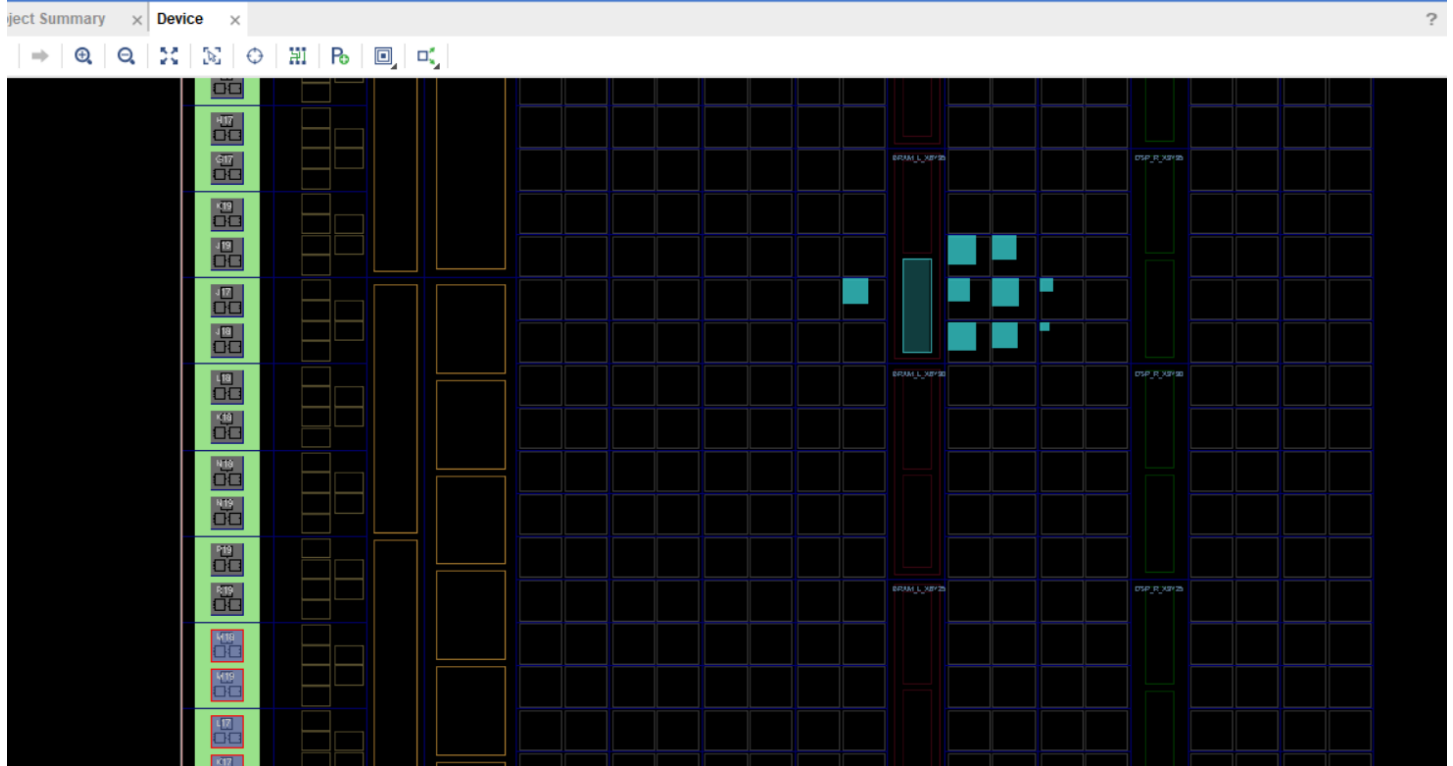
#### Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.578 ns	Worst Hold Slack (WHS): 0.055 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: 81	Total Number of Endpoints: 81	Total Number of Endpoints: 32

All user specified timing constraints are met.

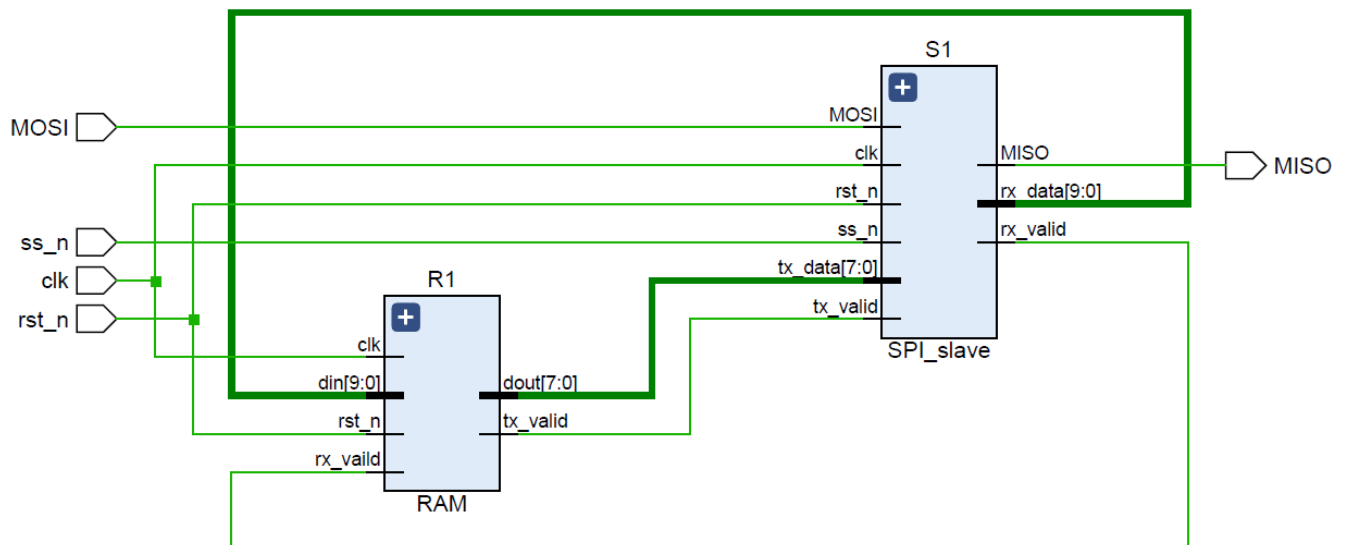
## Device:

INTED DESIGN - xc7a35t1cp236-1L (active)

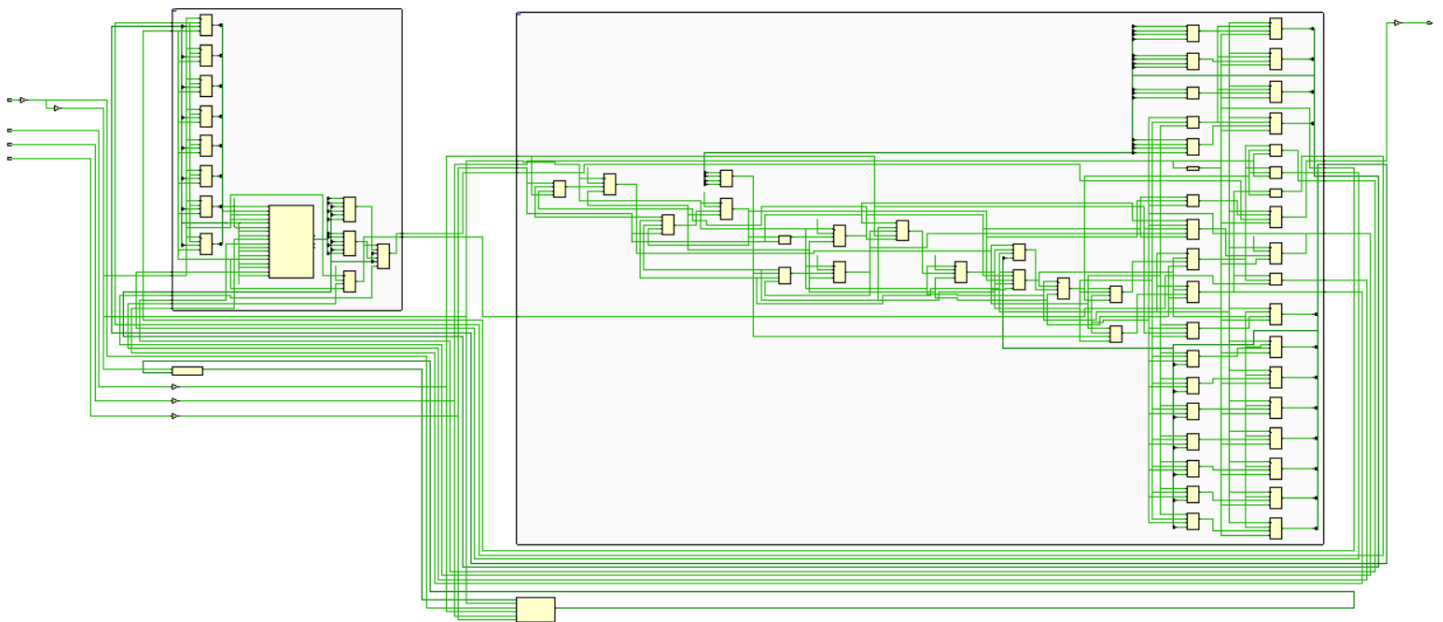


(\* fsm\_encoding = "one\_hot" \*)

## Elaborator



## Schematic Synthesis:



## Encoding Report:

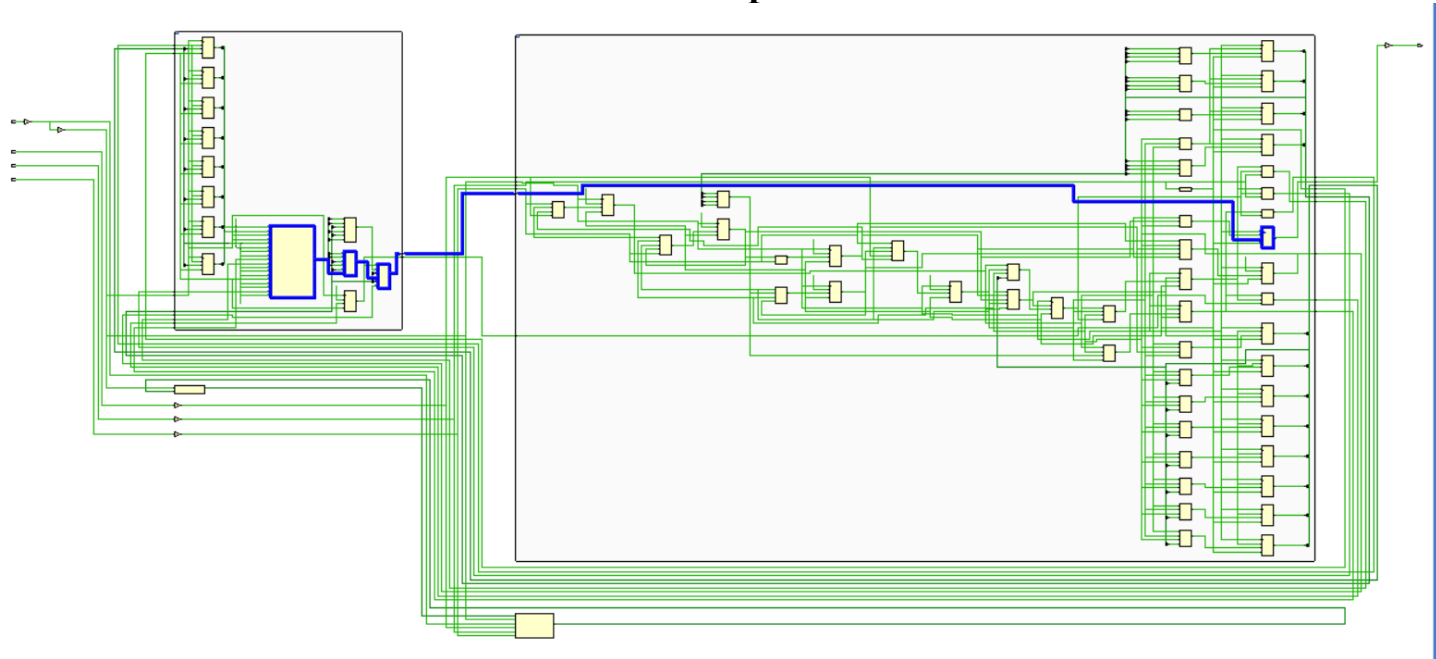
State	New Encoding	Previous Encoding
IDLE	00001	000
CHK_CMD	00010	001
WRITE	00100	010
READ_ADD	01000	100
READ_DATA	10000	011

### 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

### Critical path

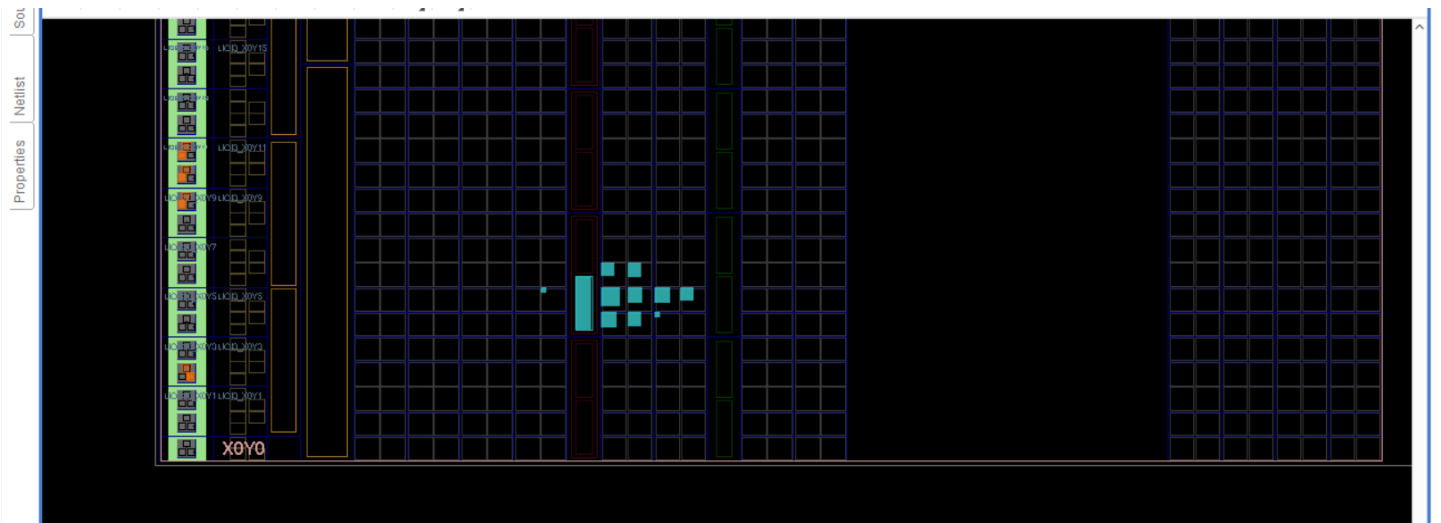


Implementation:

Reports	Design Runs	Power	Methodology	Timing	Utilization				
					Hierarchy				
Name	1	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
▼  SPI_w_RAM		30	31	10	30	20	0.5	5	1
R1 (RAM)		3	9	4	3	0	0.5	0	0
S1 (SPI_slave)		27	22	9	27	19	0	0	0

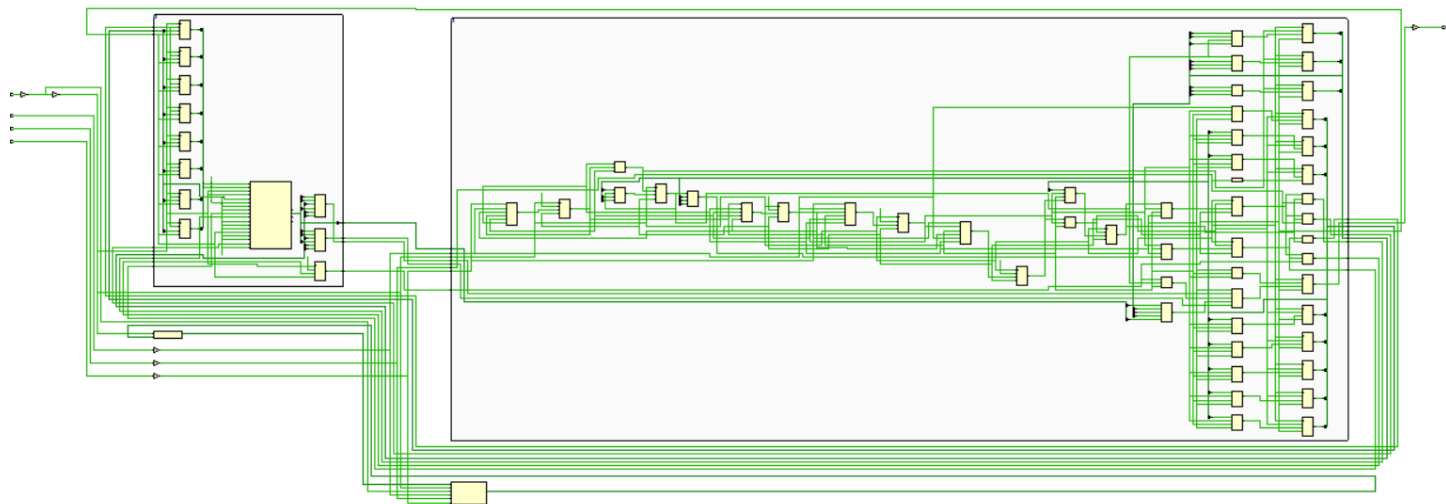
Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.414 ns	Worst Hold Slack (WHS): 0.047 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



(\* fsm\_encoding = "sequential" \*)

Synthesis Schema:

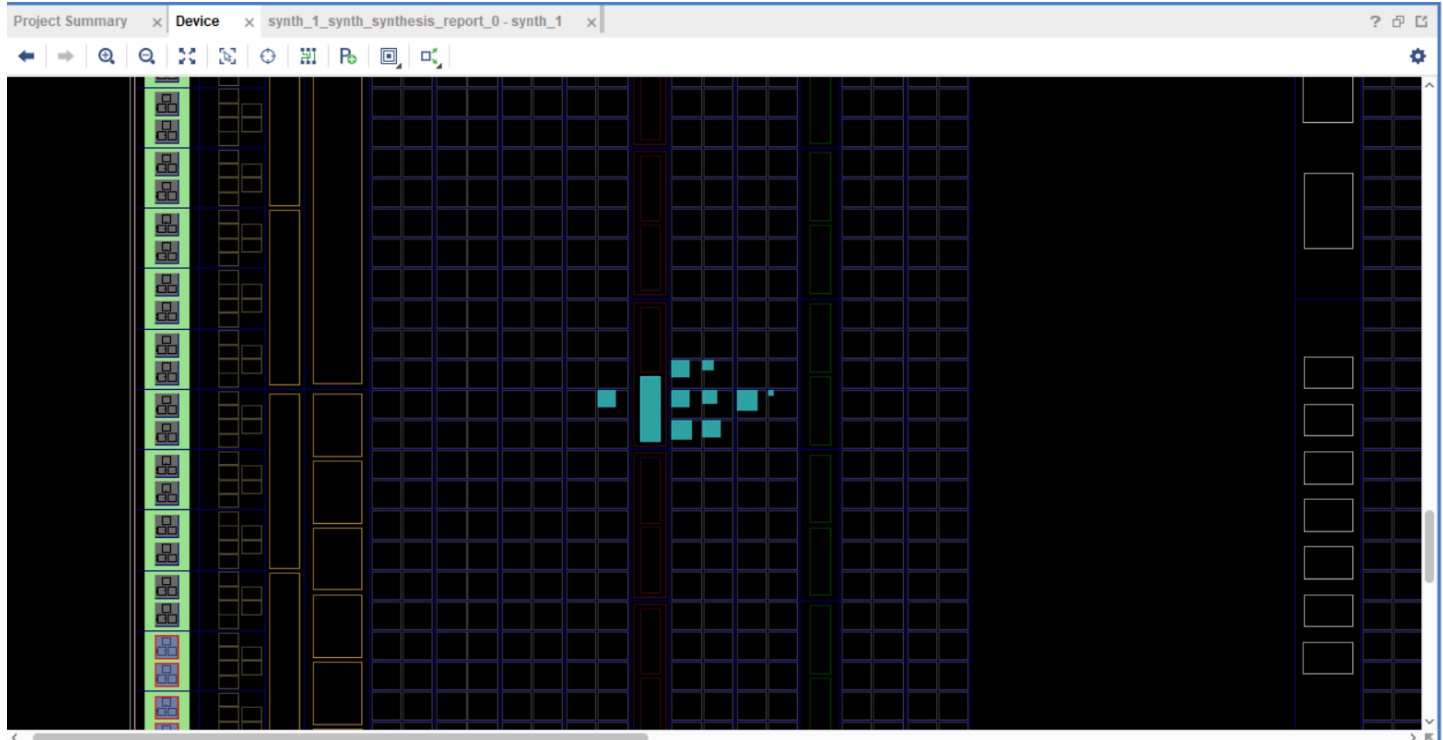


Encoding report

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

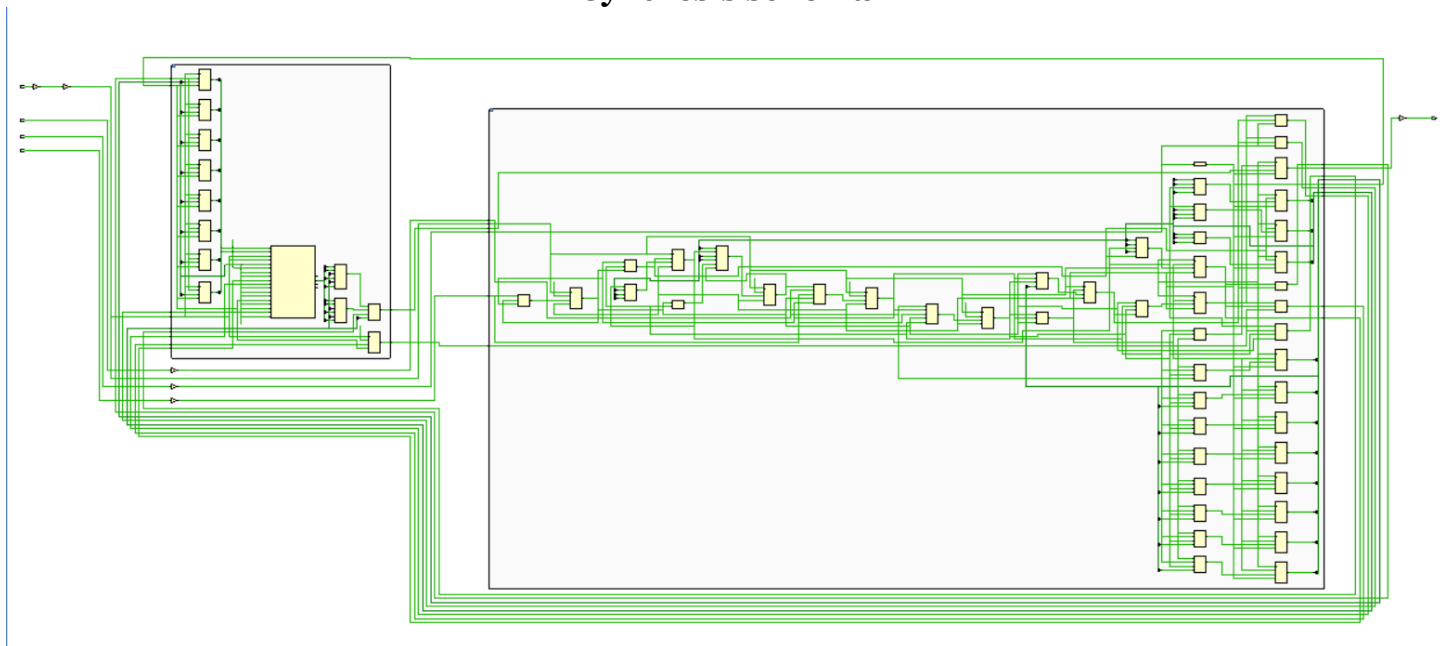
Timing report:





(\* fsm\_encoding = "johnson" \*)

### Synthesis schema





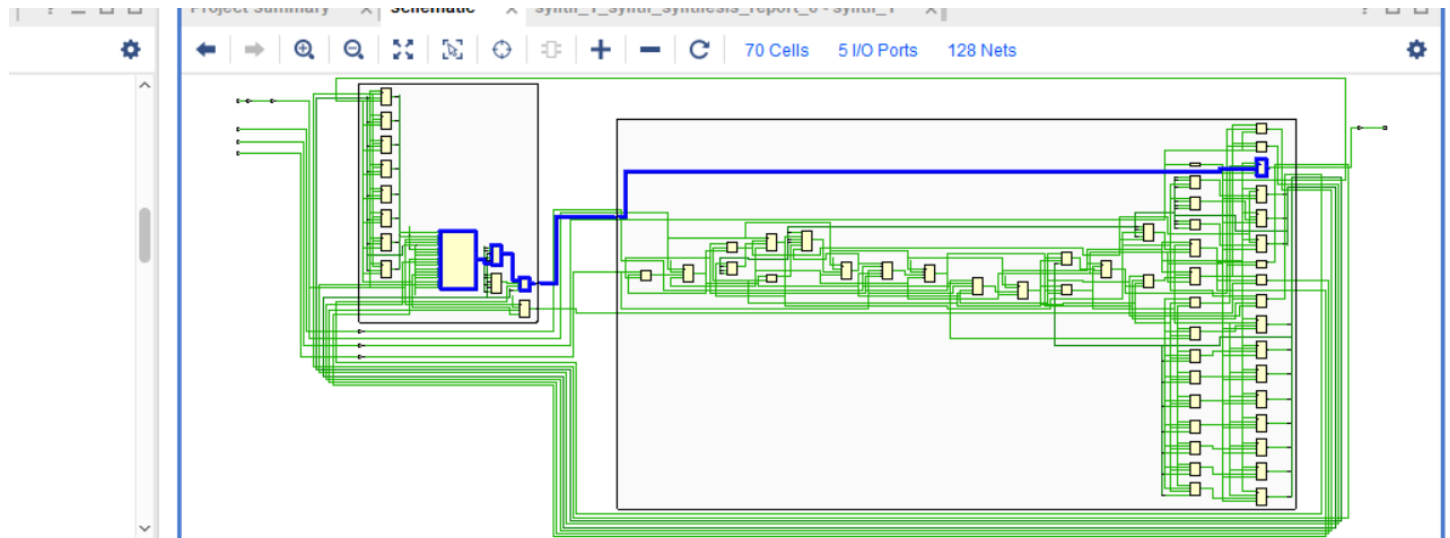
## Encoding Report:

State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	100	001
WRITE	110	010
READ_ADD	111	100
READ_DATA	011	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



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	5.445	2	3	1	R1/mem_reg/CLKBWRCLK	S1/MISO_reg/D	4.404	2.702	1.702	1000 ps
Path 2	6.978	1	2	3	S1/rx_data_reg[9]/C	R1/mem_reg/WEA[0]	2.310	0.751	1.559	1000 ps

## Implementation

Reports

Design Runs

Power

Methodology

Timing

Utilization

?

Q

≡

⚙

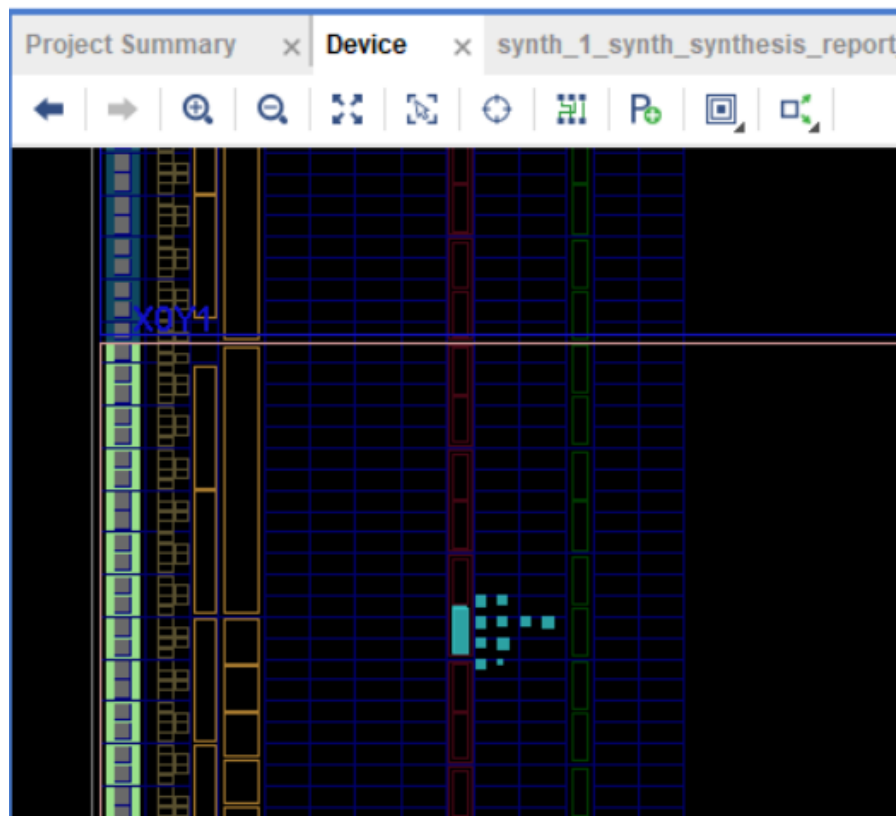
%

Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
▼ N SPI_w_RAM	29	29	10	29	18	0.5	5	1
R1 (RAM)	3	9	4	3	0	0.5	0	0
S1 (SPI_slave)	26	20	9	26	17	0	0	0

#### Design Timing Summary

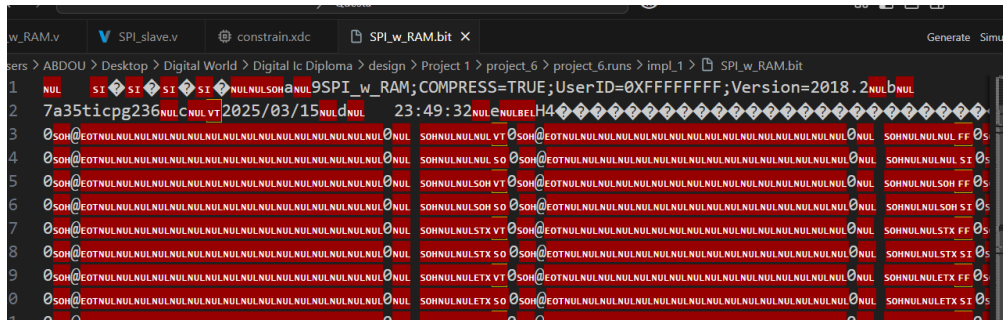
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.635 ns	Worst Hold Slack (WHS): 0.044 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns



#### Messages:

Tcl Console	Messages	Log	Reports	Design Runs	Power	Methodology	Timing	Utilization
<div> <input type="checkbox"/> Warning (1)           <input type="checkbox"/> Info (239)           <input type="checkbox"/> Status (490)           <button>Show All</button> </div>								
▼ Synthesis (1 warning) <div> <input type="checkbox"/> [Constraints 18-5210] No constraint will be written out.           </div>								

write\_bitstream Complete ✓



THANK YOU