



DIGITAL DESIGN PROJECT



Team Members:

Mohammad Wael Monir
Mostapha Sayed Abdelaal Saad-Eddin
Seif Allah Magdy Gad Saied

SPI Project

- **RTL Design Code:**

```
module RAM(  
    input [9:0] din,  
    input clk,  
    input rst_n,  
    input rx_valid,  
    output [7:0] dout,  
    output tx_valid  
);  
  
    parameter MEM_DEPTH = 256;  
    parameter ADDR_SIZE = 8;  
  
    localparam WRITE_ADD=2'b00,WRITE_DATA=2'b01,READ_ADD=2'b10,READ_DATA=2'b11;  
  
    (* RAM_STYLE = "block" *)  
    reg [7:0] memory [0:MEM_DEPTH-1];  
    reg [ADDR_SIZE-1:0] wr_addr;  
    reg [ADDR_SIZE-1:0] rd_addr;  
  
    always @(posedge clk or negedge rst_n) begin  
        if (!rst_n) begin  
            wr_addr <= 0;  
            rd_addr <= 0;  
        end else if (rx_valid) begin  
            case (din[9:8])  
                WRITE_ADD: wr_addr <= din[7:0];  
                WRITE_DATA: memory[wr_addr] <= din[7:0];  
                READ_ADD: rd_addr <= din[7:0];  
            endcase  
        end  
    end  
  
    assign dout = memory[rd_addr];  
    assign tx_valid = (din[9:8] == READ_DATA);  
  
endmodule
```

```

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

    parameter IDLE = 3'b000;
    parameter CHK_CMD = 3'b001;
    parameter WRITE = 3'b010;
    parameter READ_ADD = 3'b011;
    parameter READ_DATA = 3'b100;

    (*fsm_encoding="one_hot"*)
    reg [2:0] cs, ns;
    reg [9:0] rx_data,tx_data;
    reg [3:0] bit_count;
    reg [7:0] read_reg;
    wire flag;

    always@(posedge clk or negedge rst_n) begin

        if(!rst_n)
            read_reg<=0;

        else
            read_reg<=dout;

    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            cs <= IDLE;
        end else begin
            cs <= ns;
        end
    end
end

```

```

always @(posedge clk or negedge rst_n) begin

    if(!rst_n)
        bit_count<=0;

    else if ((cs==WRITE || cs==READ_ADD)&&(bit_count<10)) begin

        rx_data[9-bit_count] <= MOSI;
        bit_count <= bit_count + 1;

    end

    else if (flag) begin
        bit_count<=bit_count+1;
    end

    else
        bit_count<=0;
end

always @(*) begin
    case (cs)
        IDLE: begin
            if (SS_n) begin
                ns = IDLE;
            end else begin
                ns = CHK_CMD;
            end
        end

        CHK_CMD: begin
            if(SS_n)
                ns=IDLE;

            else if (MOSI == 1'b0) begin
                ns = WRITE;
            end
            else begin
                ns = READ_ADD;
            end
        end

    end

    WRITE: begin

```

```

        if (SS_n) begin
            ns = IDLE;
        end
        else begin
            ns = WRITE;
        end
    end
end

READ_ADD: begin
    if (SS_n) begin
        ns = IDLE;
    end

    else if((rx_data[8]==1)&&(bit_count==4'd10))
        ns=READ_DATA;

    else begin
        ns = READ_ADD;
    end
end

READ_DATA: begin
    if (SS_n) begin
        ns = IDLE;
    end else begin
        ns = READ_DATA;
    end
end

default:ns=IDLE;

endcase
end

always @(*) begin
    case (cs)
        WRITE,READ_ADD: begin
            din = rx_data;
            rx_valid = (bit_count==4'd10)?1:0;
            MISO = 1'b0;
        end

        READ_DATA: begin
            din = rx_data;
            rx_valid = (bit_count==4'd10)?1:0;

```

```

        MISO=(flag)?read_reg[7-bit_count]:0;
    end

    default:begin
        din=0;
        rx_valid = 1'b0;
        MISO = 1'b0;
    end

endcase
end

assign flag=((cs==READ_DATA)&&(tx_valid==1)&&(bit_count<8))?1:0;

endmodule

module SPI_Wrapper(clk,rst_n,SS_n,MOSI,MISO);

    input clk,rst_n,SS_n,MOSI;
    output MISO;

    wire [9:0] din;
    wire [7:0] dout;
    wire tx_valid,rx_valid;

    spi_slave spi_slave0(clk,rst_n,SS_n,MOSI,MISO,din,rx_valid,dout,tx_valid);
    RAM
    RAM0(.din(din),.clk(clk),.rst_n(rst_n),.rx_valid(rx_valid),.dout(dout),.tx_valid(
tx_valid));

endmodule

```

• Testbench Code:

```

module SPI_Wrapper_tb;

    reg clk,rst_n,MOSI,SS_n;
    wire MISO;

    SPI_Wrapper DUT(clk,rst_n,SS_n,MOSI,MISO);

    initial begin
        clk=0;
    end

```

```
forever
#1 clk=~clk;
end

integer signed i,j;

initial begin

    rst_n=0;
    repeat(30) begin

        MOSI=$random;
        SS_n=$random;
        @(negedge clk);

    end

    rst_n=1;
    #2;

    for(i=0;i<40;i=i+1) begin

        SS_n=0;
        #2;
        MOSI=0;
        #6;

        for(j=7;j>=0;j=j-1) begin

            MOSI=i[j];
            #2;

        end

        SS_n=1;
        #2;

        SS_n=0;
        #2;

        MOSI=0;
        #4;

        MOSI=1;
```

```
#2;

for(j=0;j<8;j=j+1) begin

    MOSI=$random;
    #2;

end

SS_n=1;
#2;

end

for(i=0;i<40;i=i+1) begin

    SS_n=0;
    #2;
    MOSI=1;
    #4;

    MOSI=0;
    #2;

    for(j=7;j>=0;j=j-1) begin

        MOSI=i[j];
        #2;

    end

    SS_n=1;
    #2;

    SS_n=0;
    #2;

    MOSI=1;
    #6;

    MOSI=$random;
    #32;

    SS_n=1;
    #2;
```



```

        end

    $stop;
    end

endmodule

```

• Do File:

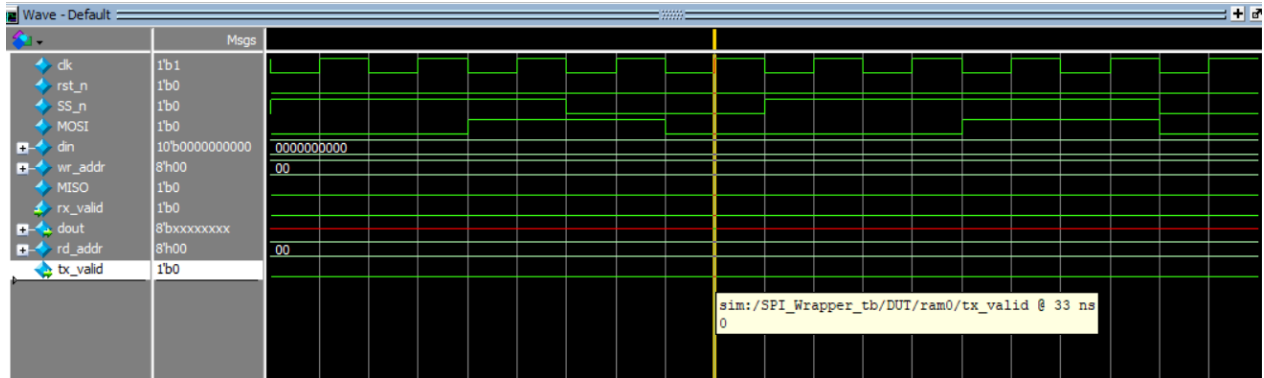
```

vlib work
vlog SPI_Wrapper_tb.v
vsim -voptargs=+acc work.SPI_Wrapper_tb
add wave *
run -all

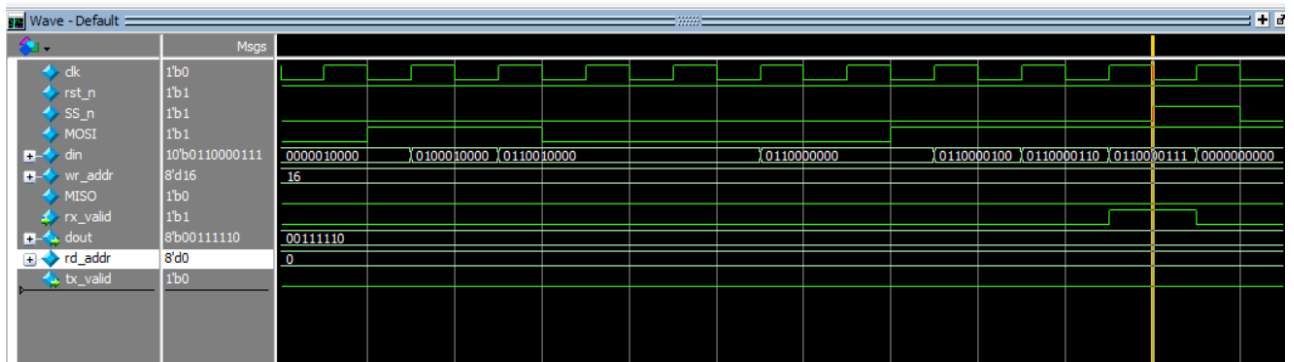
```

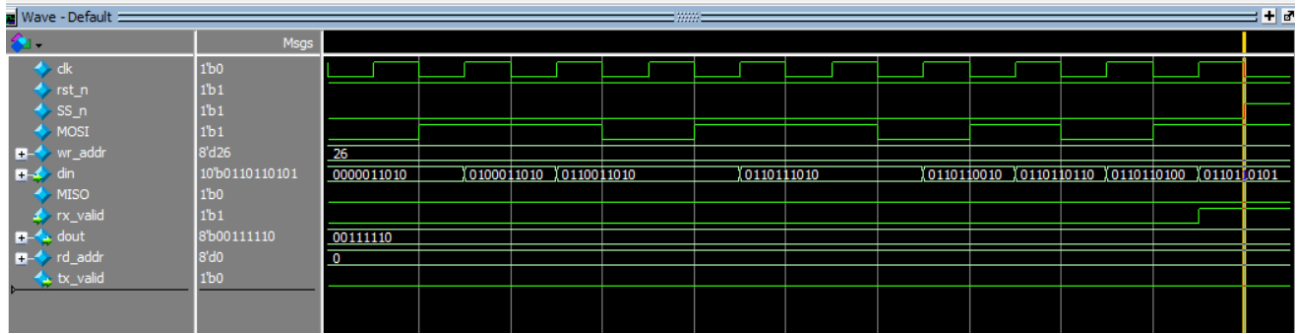
• Questa-Sim Snippets:

Reset:

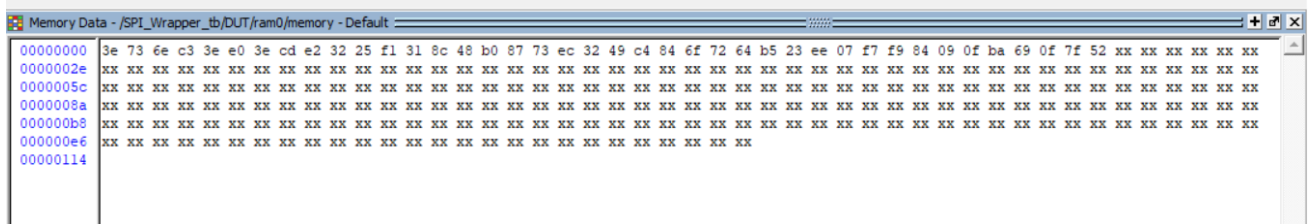


Writing examples:

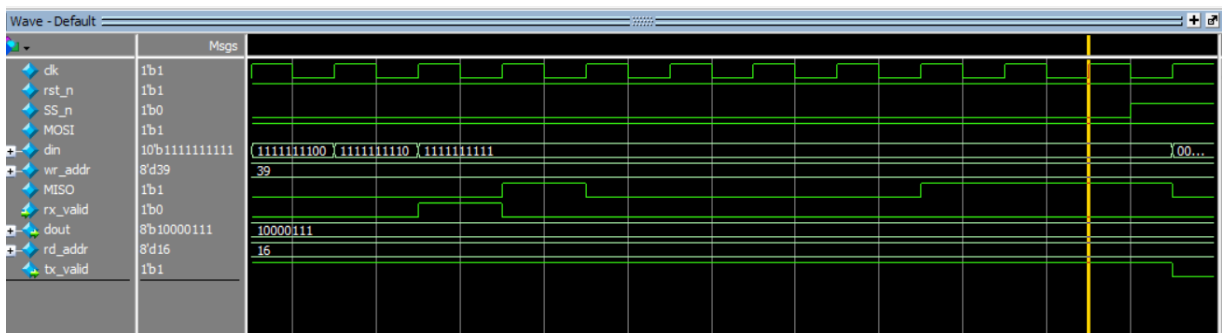
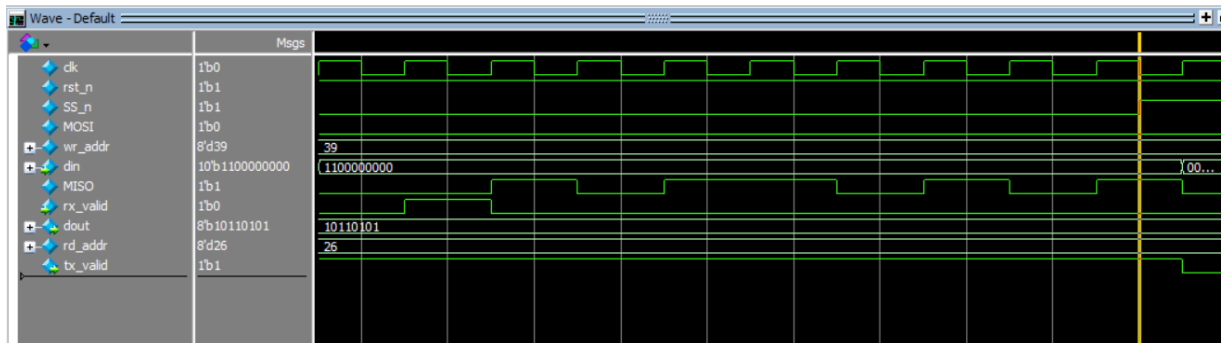




Memory after writing on 40 addresses:

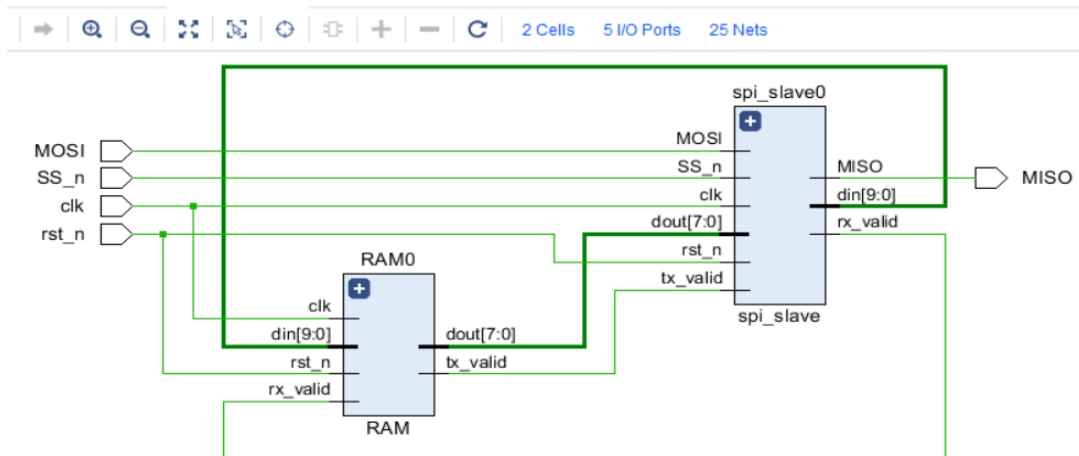


Reading examples:

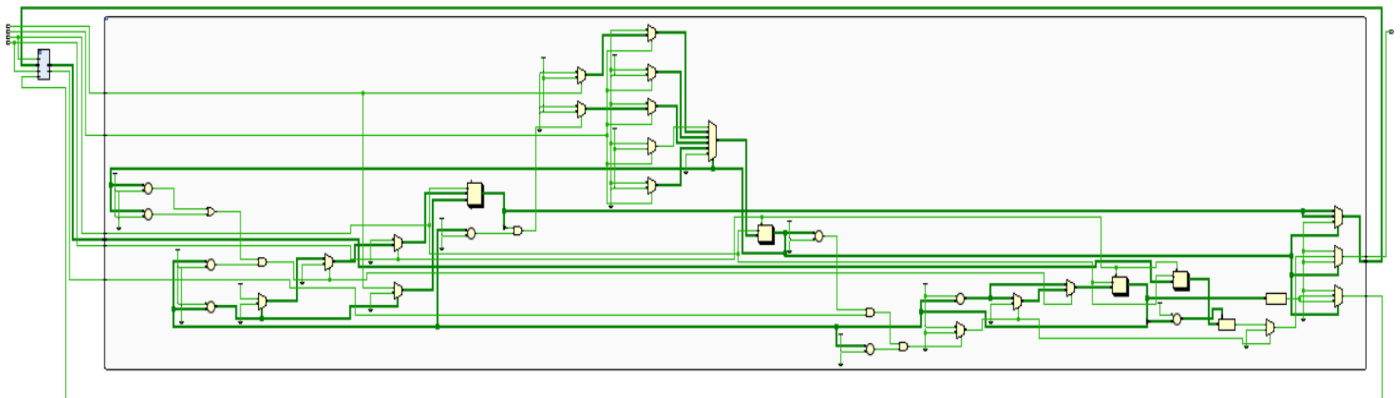


- **Elaboration:**

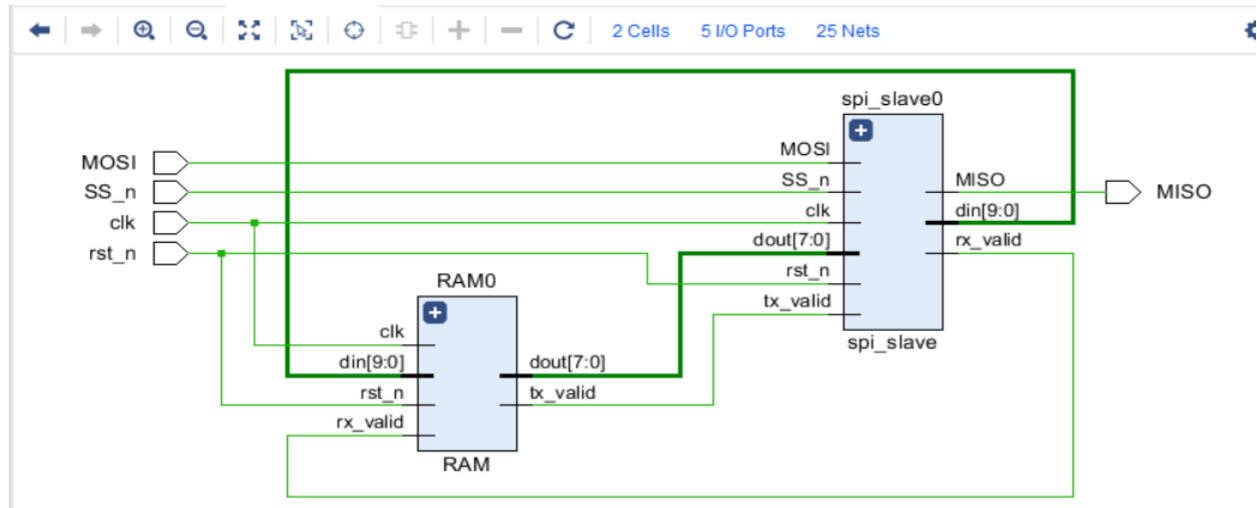
- 1. Gray Encoding:



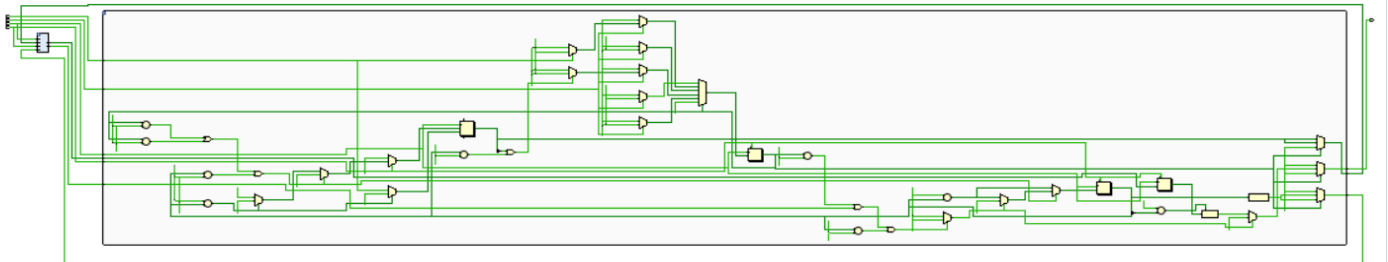
SPI slave elaboration:



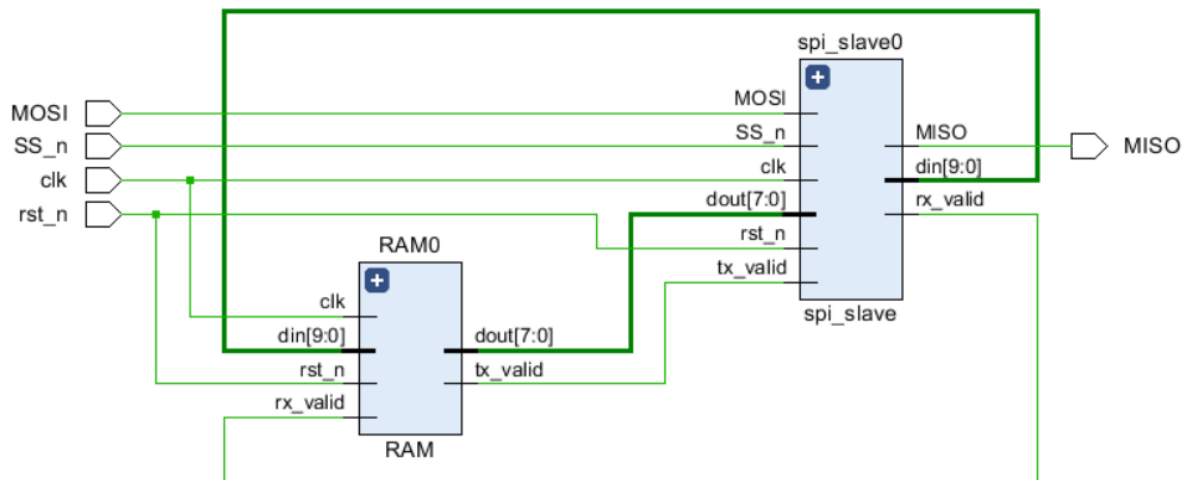
2. Sequential Encoding:



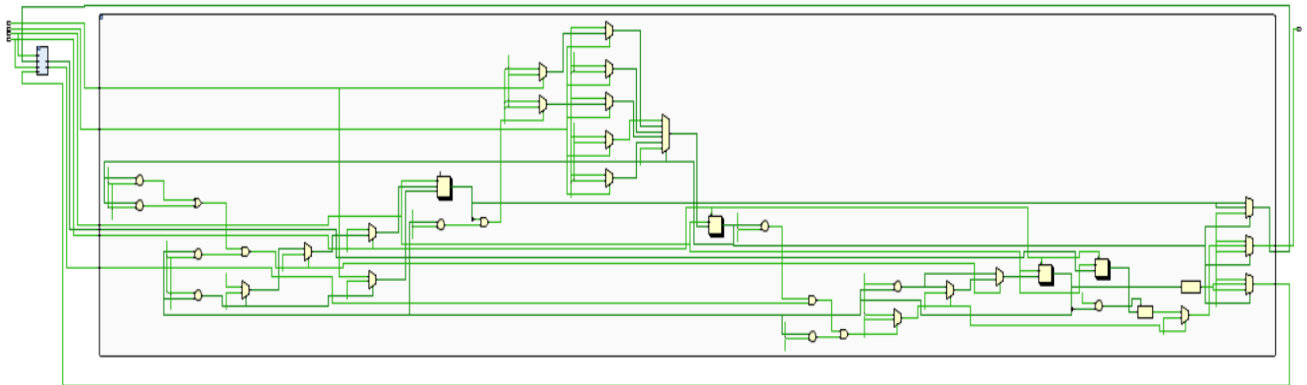
SPI slave elaboration:



1. One-hot Encoding:

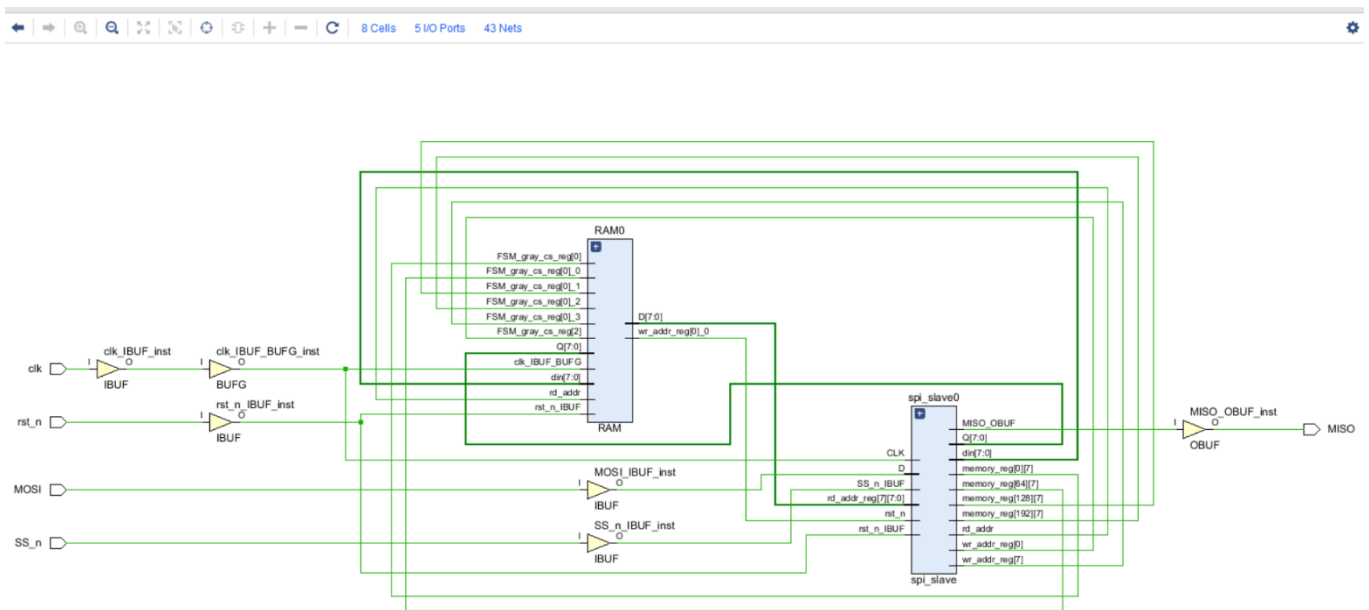


SPI slave elaboration:

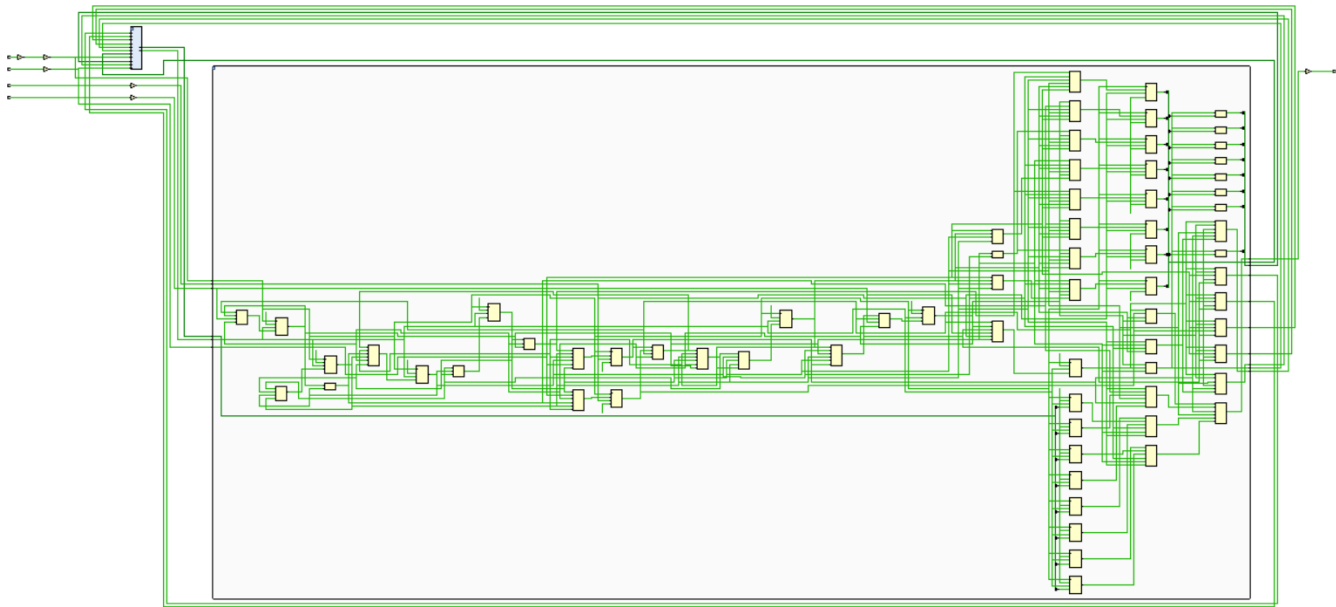


- **Synthesis:**

1. Gray Encoding:



SPI slave Synthesis:

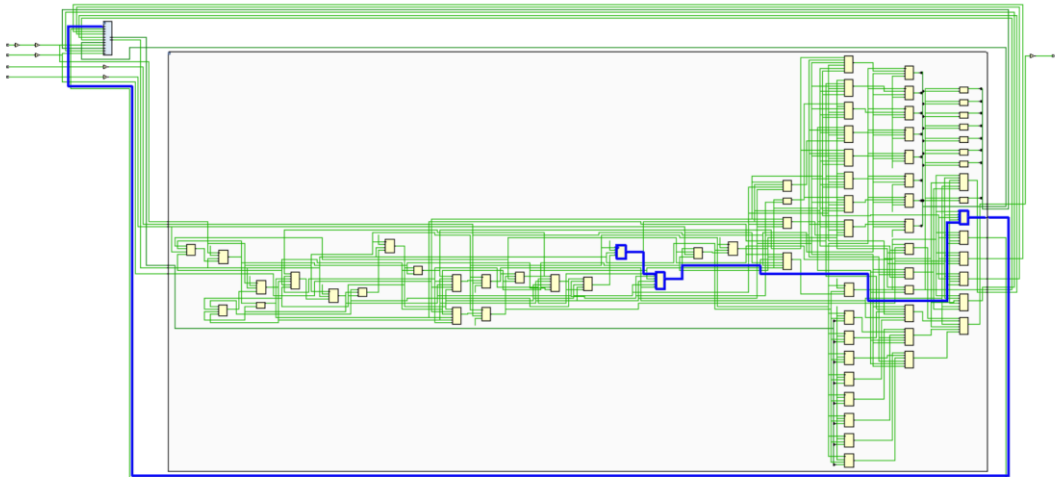


Timing report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.516 ns	Worst Hold Slack (WHS): 0.160 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: 6255	Total Number of Endpoints: 6255	Total Number of Endpoints: 2117

All user specified timing constraints are met.

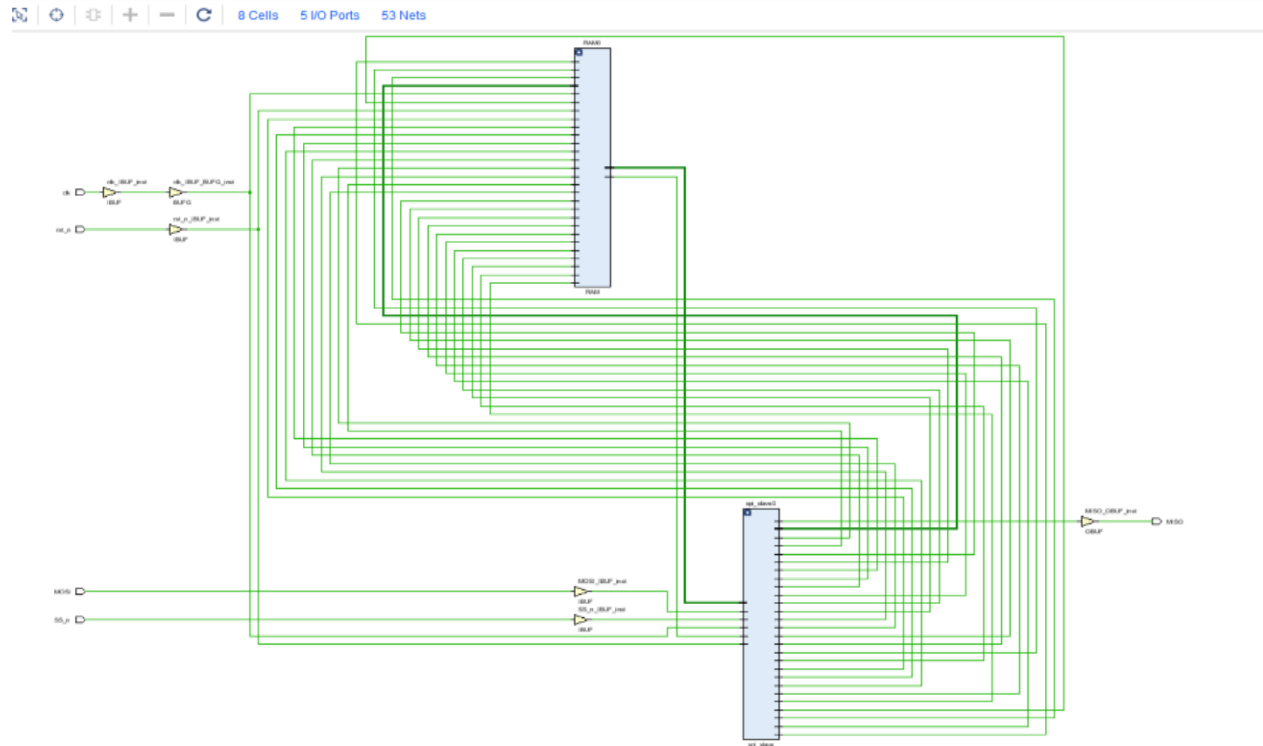
Critical Path:



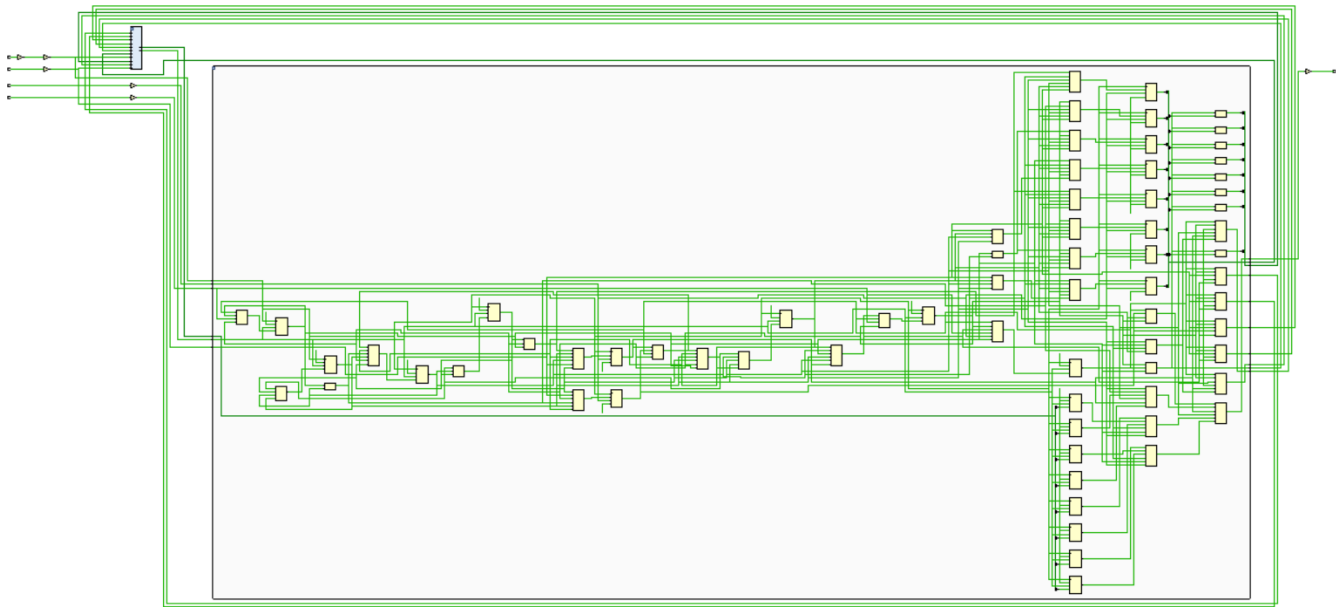
Synthesis report:

```
100 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
101 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
102 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
103 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
104 -----
105 State | New Encoding | Previous Encoding
106 -----
107 IDLE | 000 | 000
108 CHK_CMD | 001 | 001
109 WRITE | 011 | 010
110 READ_ADD | 010 | 011
111 READ_DATA | 111 | 100
112 -----
113 INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'gray' in module 'spi_slave'
114 -----
```

2. Sequential Encoding:



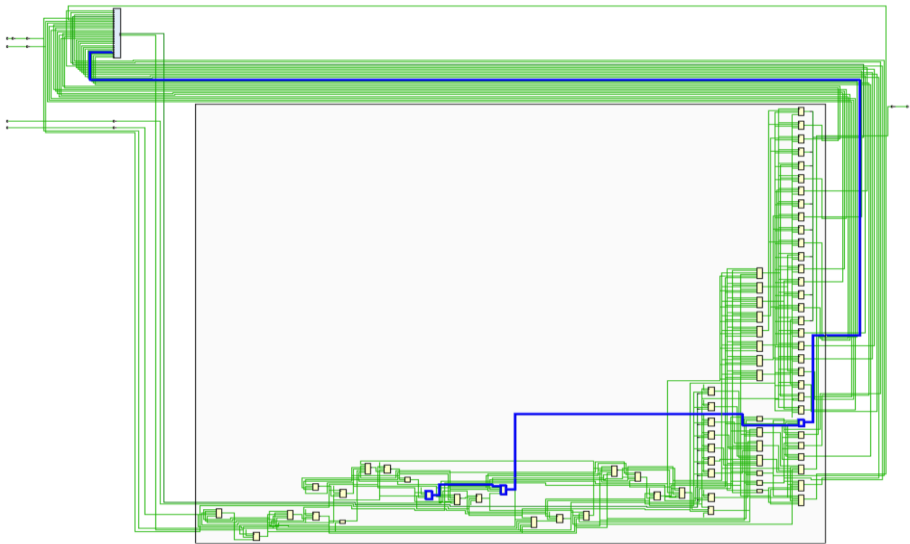
SPI slave Synthesis:



Timing report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.670 ns	Worst Hold Slack (WHS): 0.181 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: 6271	Total Number of Endpoints: 6271	Total Number of Endpoints: 2133

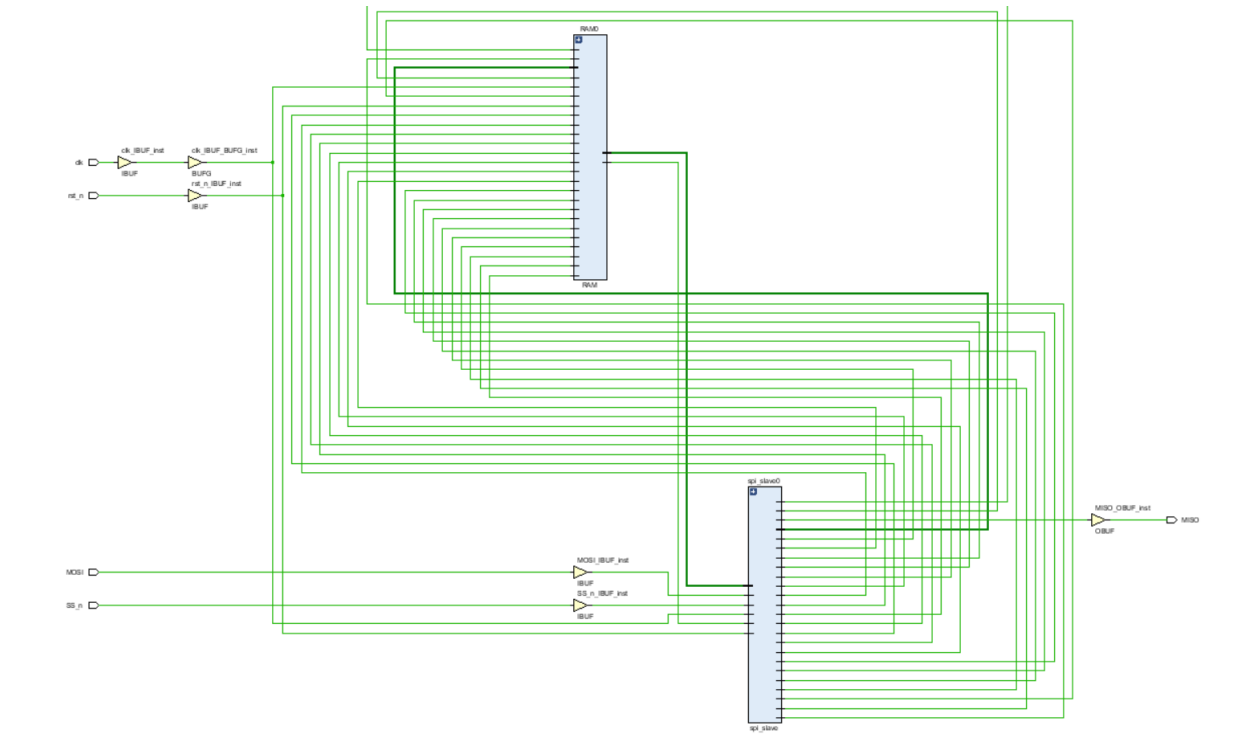
Critical Path:



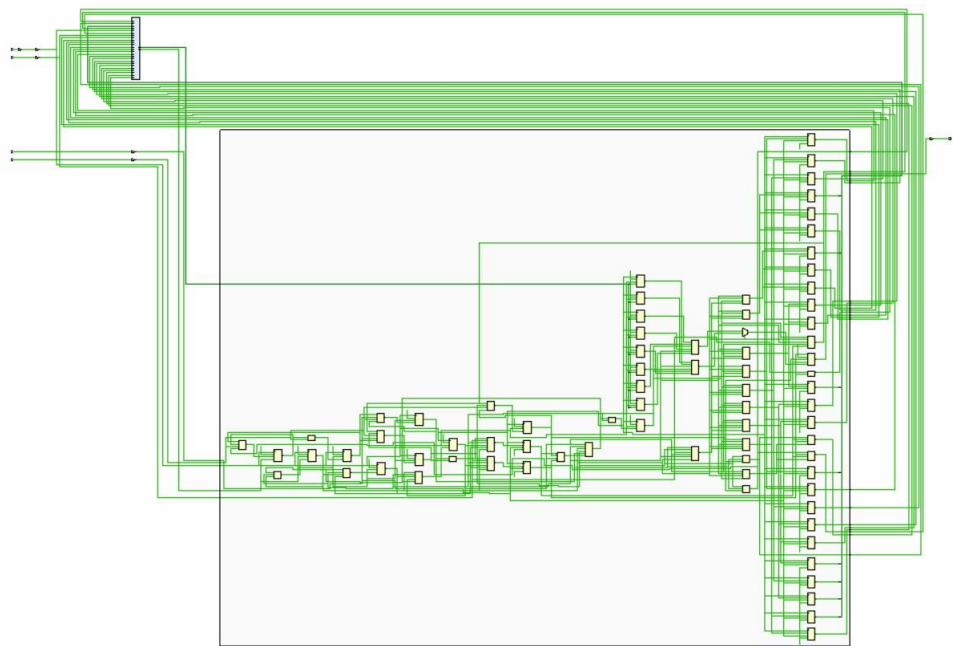
Synthesis report:

```
100 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
101 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
102 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
103 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
104 -----
105 State | New Encoding | Previous Encoding
106 -----
107 IDLE | 000 | 000
108 CHK_CMD | 001 | 001
109 WRITE | 010 | 010
110 READ_ADD | 011 | 011
111 READ_DATA | 100 | 100
112 -----
113 INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'sequential' in module 'spi_slave'
114 -----
```

3. One-hot Encoding:



SPI slave Synthesis:

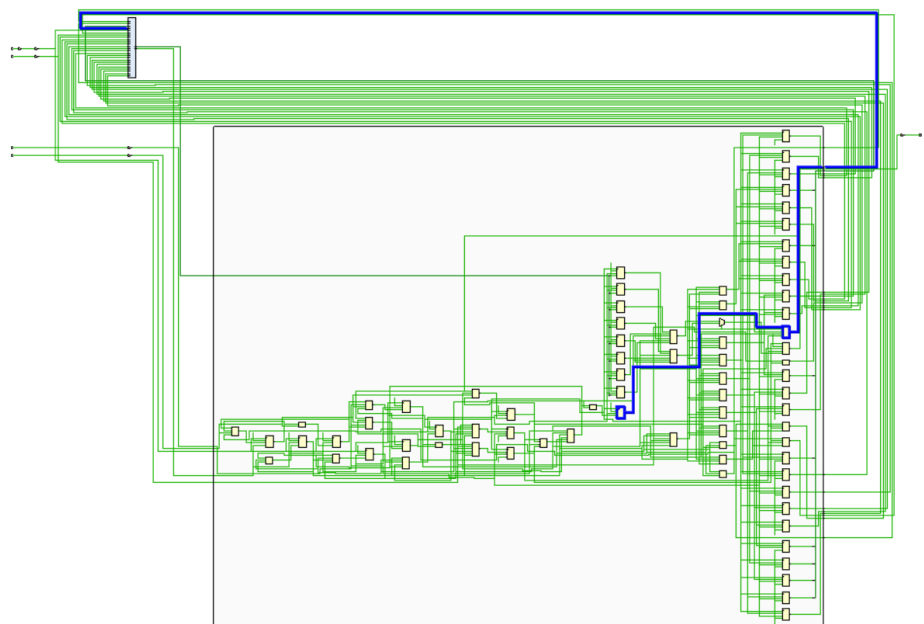


Timing report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.469 ns	Worst Hold Slack (WHS): 0.148 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: 6273	Total Number of Endpoints: 6273	Total Number of Endpoints: 2135

All user specified timing constraints are met.

Critical Path:



Synthesis report:

```
100 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
101 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
102 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
103 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
104 -----
105 State | New Encoding | Previous Encoding
106 -----
107 IDLE | 00001 | 000
108 CHK_CMD | 00010 | 001
109 WRITE | 00100 | 010
110 READ_ADD | 01000 | 011
111 READ_DATA | 10000 | 100
112 -----
113 INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'spi_slave'
114 -----
```

• Implementation:

1. Gray Encoding:

Device:



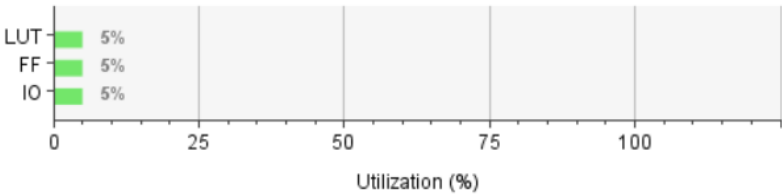
Timing Report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.708 ns	Worst Hold Slack (WHS): 0.187 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: 6255	Total Number of Endpoints: 6255	Total Number of Endpoints: 2117

All user specified timing constraints are met.

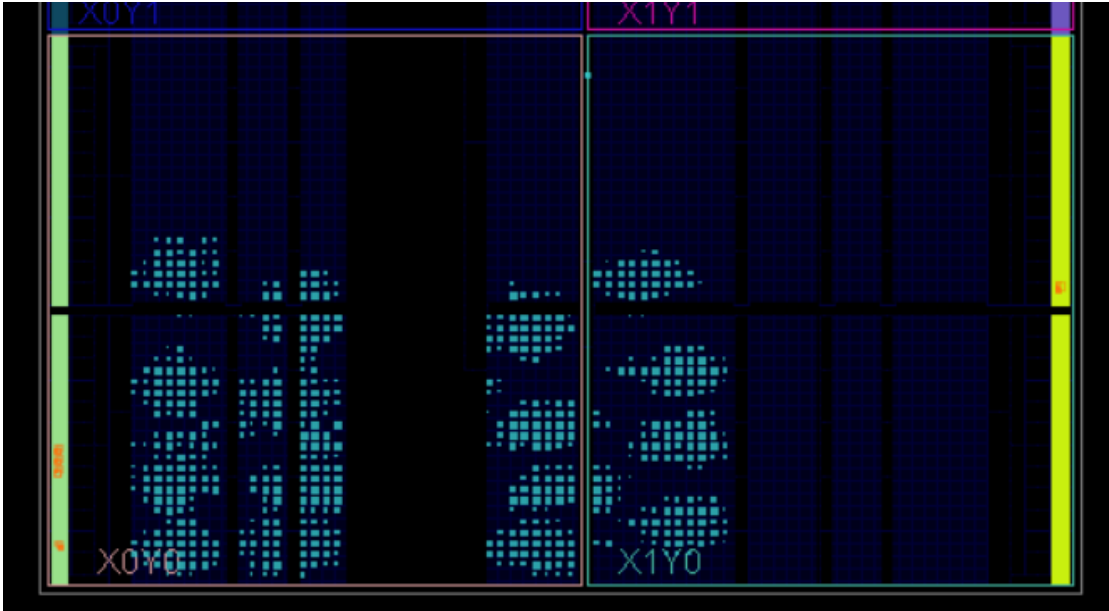
Utilization:

Resource	Utilization	Available	Utilization %
LUT	1117	20800	5.37
FF	2116	41600	5.09
IO	5	106	4.72



2. Sequential:

Device:

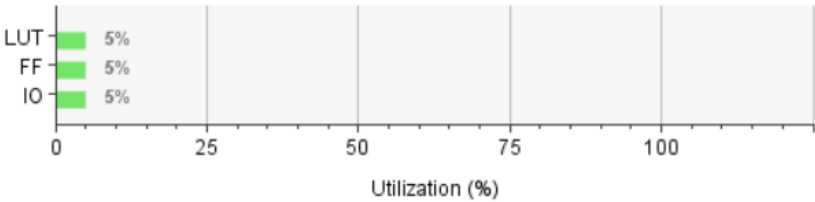


Timing Report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 1.421 ns	Worst Hold Slack (WHS): 0.209 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: 6271	Total Number of Endpoints: 6271	Total Number of Endpoints: 2133
All user specified timing constraints are met.		

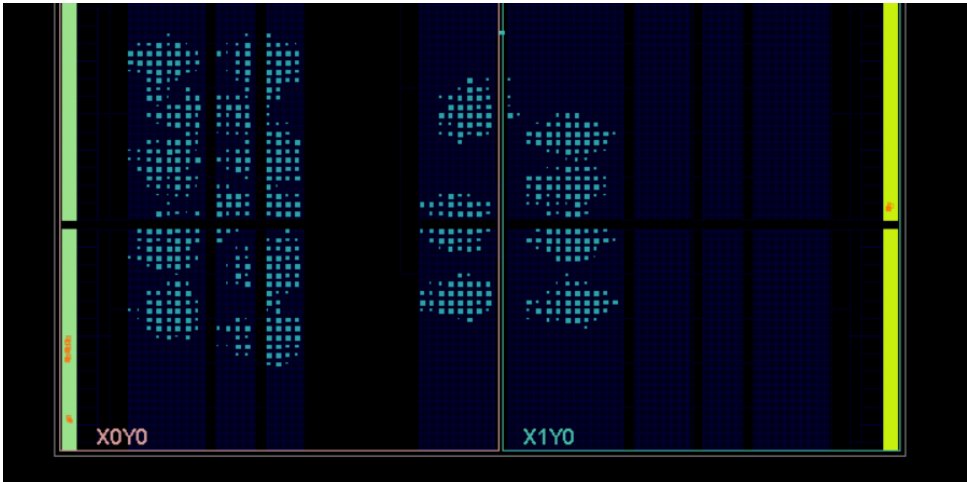
Utilization:

Resource	Utilization	Available	Utilization %
LUT	1110	20800	5.34
FF	2132	41600	5.13
IO	5	106	4.72



3. One-hot Encoding:

Device:



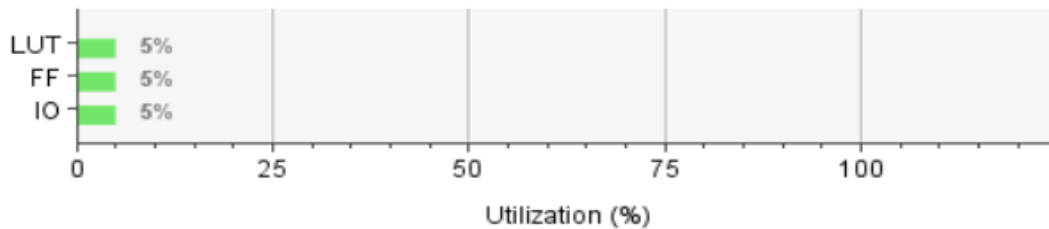
Timing Report:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 2.002 ns	Worst Hold Slack (WHS): 0.176 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: 6273	Total Number of Endpoints: 6273	Total Number of Endpoints: 2135

All user specified timing constraints are met.

Utilization:

Resource	Utilization	Available	Utilization %
LUT	1104	20800	5.31
FF	2134	41600	5.13
IO	5	106	4.72



One-hot has the highest slack in the implementation so it'll be used.

• Successful bitstream generation:

The screenshot shows two windows from a digital design tool. On the left, a 'Bitstream Generation Completed' dialog box is open, displaying a message: 'Bitstream Generation successfully completed.' Below the message, there are three radio buttons under the 'Next' section: 'View Reports', 'Open Hardware Manager' (which is selected), and 'Generate Memory Configuration File'. At the bottom of the dialog, there is a checkbox labeled 'Don't show this dialog again' and two buttons: 'OK' and 'Cancel'. On the right, the 'Project Summary' window is visible, showing the file path 'D:/C/Digital_Design_Kareem_Waseem/Project2/Project_2_runs/synth_1/SPI_Wrapper.vds'. Below the path, there is a search bar and a list of log messages. The messages include information about ROM mapping and FSM encoding. A table within the log shows the state transitions for the FSM, with columns for 'State', 'New Encoding', and 'Previous Encoding'. The table lists states: IDLE, CHK_CMD, WRITE, READ_ADD, and READ_DATA, along with their corresponding new and previous encodings. At the bottom of the log, it states 'Finished RTL Optimization Phase 2 : Time (s): cpu = 00:01:03 ; elapsed = 00:01:40 . Memory (MB): peak = 837.418 ; gain'.

Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

☐ View Reports

☒ Open Hardware Manager

☐ Generate Memory Configuration File

☐ Don't show this dialog again

OK Cancel

Project Summary synth_1_synth_synthesis_report_0 - synth_1

D:/C/Digital_Design_Kareem_Waseem/Project2/Project_2_runs/synth_1/SPI_Wrapper.vds

100 INFO: [Synth 8-5544] ROM "na" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

101 INFO: [Synth 8-5544] ROM "na" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

102 INFO: [Synth 8-5544] ROM "na" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

103 INFO: [Synth 8-5544] ROM "na" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

104

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

112

113 INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'spi_slave'

114

115 Finished RTL Optimization Phase 2 : Time (s): cpu = 00:01:03 ; elapsed = 00:01:40 . Memory (MB): peak = 837.418 ; gain

- No errors screenshot:

