Project (2)
SPI Slave with Single Port RAM

Abdullah Mohammed Ahmed Mahmoud
Supervising:
Eng. Kareem Wassem

August 2024

RAM Code

```verilog
module RAM #(
    parameter MEM_DEPTH = 256,
    parameter ADDR_SIZE = 8
)(
    input clk, rst_n,
    input [ADDR_SIZE+1:0] din,
    input rx_valid,

    output reg [ADDR_SIZE-1:0]dout,
    output reg tx_valid
);
    reg [ADDR_SIZE-1:0]w_addr, r_addr;
    reg [ADDR_SIZE-1:0] ram [MEM_DEPTH-1:0];

always @(posedge clk or negedge rst_n) begin
        if (~rst_n) begin
            dout <= 0;
            tx_valid <= 0;

    end

    else begin
        if (rx_valid) begin
            case (din[9:8])
                2'b00:begin
                    w_addr <= din[7:0];
                    tx_valid <=0;
                end
                2'b01:begin
                    ram[w_addr] <= din[7:0];
                    tx_valid <= 0;
                end
                2'b10: begin
                    r_addr <= din[7:0];
                    tx_valid = 0;
                end
                2'b11: begin
                    dout <= ram[r_addr];
                    tx_valid = 1;
                end
            endcase
        end
    end
end
endmodule
```

SPI Code

```verilog
module SPI (
    input clk, rst_n,
    input MOSI, SS_n,
    input tx_valid,
    input [7:0]tx_data,

    output reg MISO,
    output reg rx_valid,
    output reg [9:0]rx_data
);
    parameter IDLE = 3'b000;
    parameter CHK_CMD = 3'b001;
    parameter WRITE = 3'b010;
    parameter READ_ADD = 3'b011;
    parameter READ_DATA = 3'b100;

    reg read_trans;
    reg [4:0] cs, ns;

    always @(*) begin
        case (cs)
            IDLE:
                if (SS_n)
                    ns = IDLE;
                else
                    ns = CHK_CMD;
            CHK_CMD:
                if (SS_n)
                    ns = IDLE;
                else if (~SS_n && ~MOSI)
                    ns = WRITE;
                else if (~SS_n && MOSI && ~read_trans)
                    ns = READ_ADD;
                else
                    ns = READ_DATA;
            WRITE:
                if (~SS_n)
                    ns = WRITE;
                else
                    ns = IDLE;
            READ_ADD:
                if (~SS_n)
                    ns = READ_ADD;
                else
                    ns = IDLE;
            READ_DATA:
                if (~SS_n)
                    ns = READ_DATA;
                else
                    ns = IDLE;
            default: ns = IDLE;
        endcase
    end


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

    reg [4:0] counter;
    always @(posedge clk) begin
        if (~rst_n) begin
            counter <= 0;
            rx_data <= 0;
            MISO <= 0;
            rx_valid <= 0;
            read_trans = 0;
        end
        else if (cs != IDLE && cs != CHK_CMD) begin
                if(cs == READ_ADD)
                    read_trans = 1;
                else if(cs == READ_DATA)
                    read_trans = 0;

                if (counter<=9)
                    rx_data = {MOSI, rx_data[9:1]};

                if (counter == 9)
                    rx_valid = 1;
                else if (counter == 11 || counter == 0)
                    rx_valid <= 0;

                if (tx_valid) begin
                    if (counter==11) MISO <= tx_data[0];
                    else if (counter==12) MISO <= tx_data[1];
                    else if (counter==13) MISO <= tx_data[2];
                    else if (counter==14) MISO <= tx_data[3];
                    else if (counter==15) MISO <= tx_data[4];
                    else if (counter==16) MISO <= tx_data[5];
                    else if (counter==17) MISO <= tx_data[6];
                    else if (counter==18) MISO <= tx_data[7];
                end

                counter <= counter + 1;
        end else counter = 0;
    end
endmodule
```

Wrapper Code (Instantiating)

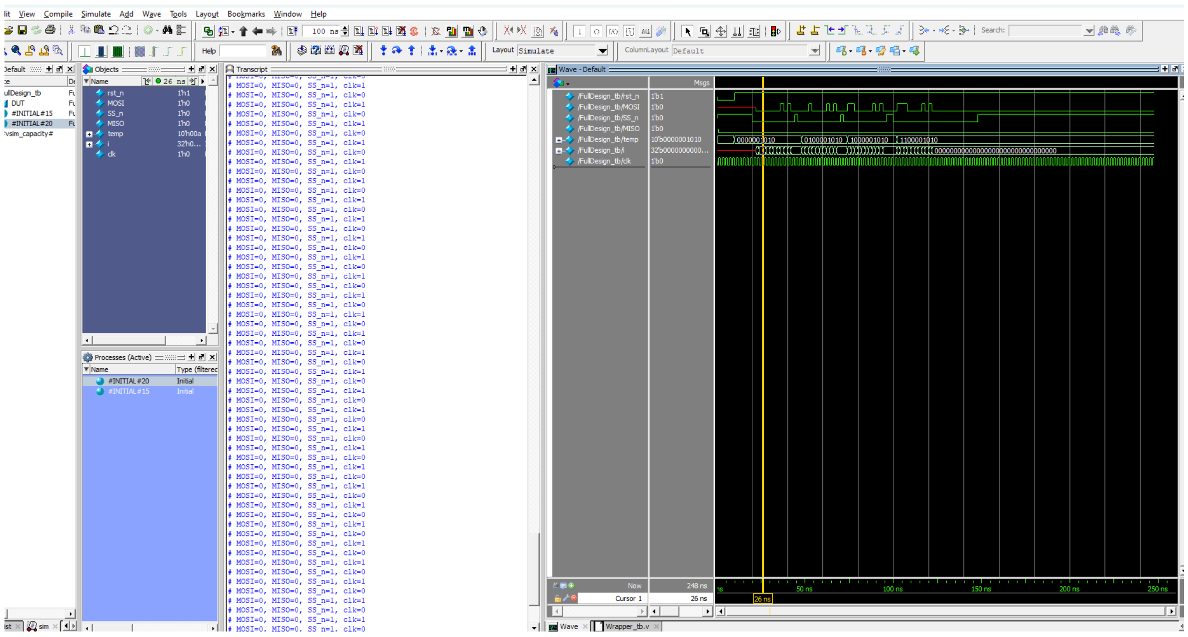
```
    module FullDesign (
    input clk, rst_n,
    input MOSI, SS_n,
    output MISO
);

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

    SPI spiBlock(
        .clk(clk), .rst_n(rst_n),
        .MOSI(MOSI), .SS_n(SS_n),
        .tx_valid(tx_valid), .tx_data(tx_data),
        .MISO(MISO),
        .rx_valid(rx_valid), .rx_data(rx_data)
    );

    RAM #(
        .MEM_DEPTH(256),
        .ADDR_SIZE(8)
    ) ramBlock (
        .clk(clk), .rst_n(rst_n),
        .din(rx_data),
        .rx_valid(rx_valid),
        .dout(tx_data),
        .tx_valid(tx_valid)
    );
endmodule
```

Wrapper TestBench

```verilog
module FullDesign_tb();
reg clk, rst_n;
reg MOSI, SS_n;
wire MISO;

FullDesign DUT(
    .clk(clk), .rst_n(rst_n),
    .MOSI(MOSI), .SS_n(SS_n),
    .MISO(MISO)
);

reg [9:0] temp;
integer i;

initial begin
    clk = 0;
    forever #1 clk = ~clk;
end

initial begin
    rst_n = 0;
    SS_n = 1;
    temp = 0;
    #10;
    temp = 10'b0000_01010;
    rst_n = 1;
    #10;
    @(negedge clk) SS_n = 0;
    @(negedge clk) MOSI = 0;
    for(i = 10; i > 0; i = i - 1) begin
        @(negedge clk) MOSI = temp[i - 1];
    end
    @(negedge clk) SS_n = 1;
    @(negedge clk) SS_n = 0;
    @(negedge clk) begin
        MOSI = 0;
        temp = 10'b01000_01010;
    end
    for(i = 10; i > 0; i = i - 1) begin
        @(negedge clk) MOSI = temp[i - 1];
    end
    @(negedge clk) SS_n = 1;
    @(negedge clk) SS_n = 0;
    @(negedge clk) begin
        MOSI = 1;
        temp = 10'b10000_01010;
    end
    for(i = 10; i > 0; i = i - 1) begin
        @(negedge clk) MOSI = temp[i - 1];
    end
    @(negedge clk) SS_n = 1;
    @(negedge clk) SS_n = 1;
    @(negedge clk) SS_n = 0;
    @(negedge clk) begin
        MOSI = 1;
        temp = 10'b11000_01010;
    end
    for(i = 10; i > 0; i = i - 1) begin
        @(negedge clk) MOSI = temp[i - 1];
    end
    #25;
    @(negedge clk) SS_n = 1;
    #100;
    $stop;
end
```

```verilog

    initial begin
        $monitor("MOSI=%b, MISO=%b, SS_n=%b, clk=%b", MOSI, MISO, SS_n, clk);
    end
endmodule
```

Simulation On Questa Sim



```
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
# MOSI=0, MISO=0, SS_n=1, clk=1
# MOSI=0, MISO=0, SS_n=1, clk=0
```

Figure 1: Simulation of the *FullDesign − tb* on Questa.

Constraint File

I used the same one we had used in the session.

```
    ## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level
    signal names in the project

## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {rst_n}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {SS_n}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {MOSI}]
#set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]


## LEDs
```

```
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {MISO}]
#set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {led[1]}]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Vivado Simulation

Elaborated Design

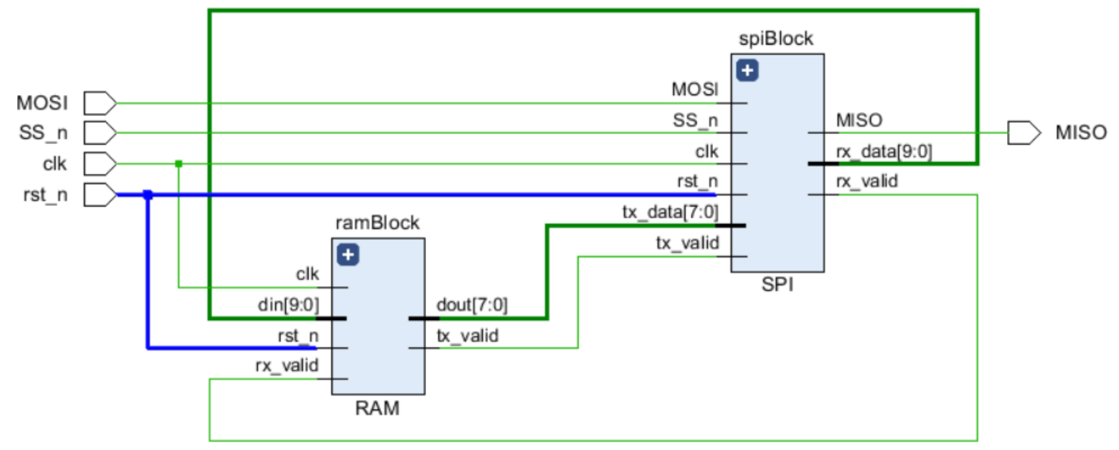


Figure 2: The Full Design

Synthesis

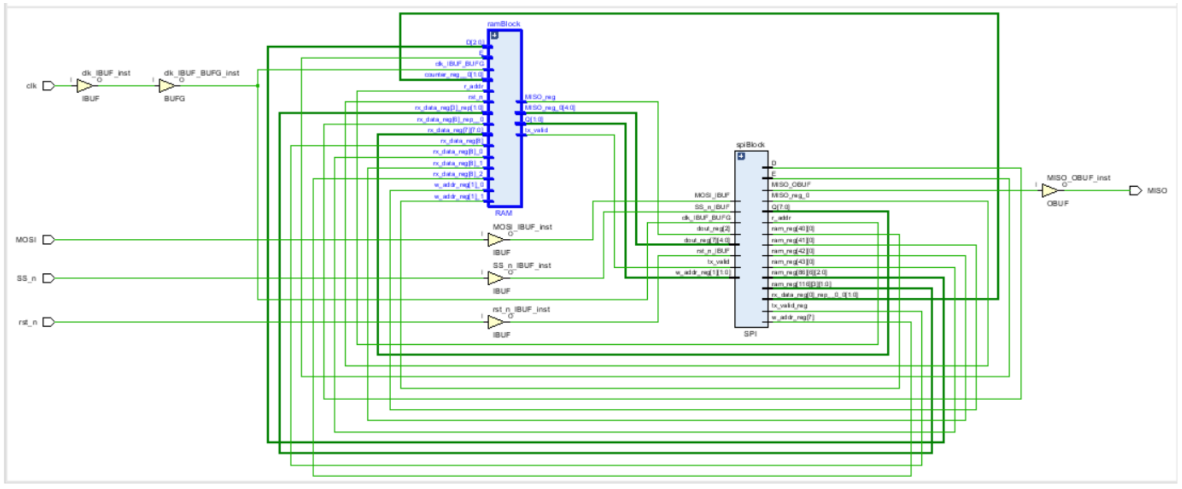
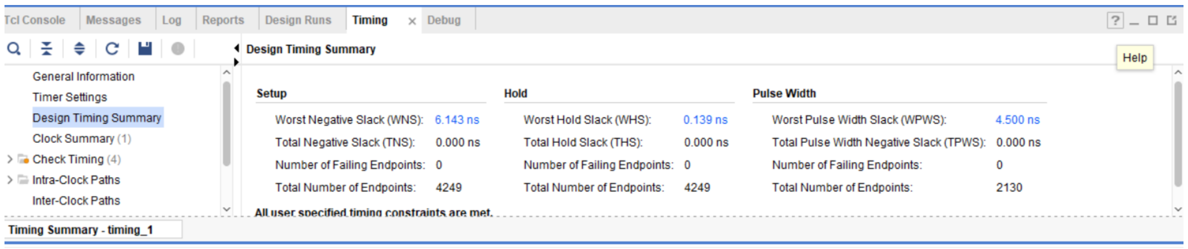


Figure 3: After "Run Synthesis"



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

Help

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (4)
Intra-Clock Paths
Inter-Clock Paths

Design Timing Summary

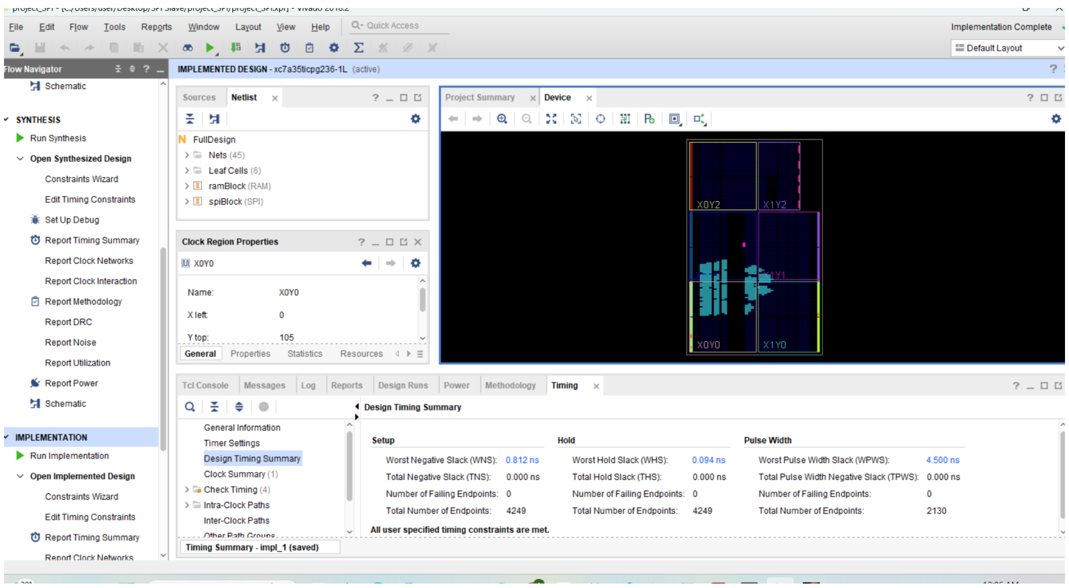
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.143 ns	Worst Hold Slack (WHS):	0.139 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4249	Total Number of Endpoints:	4249	Total Number of Endpoints:	2130

All user specified timing constraints are met.

Timing Summary - timing_1

Figure 4: Timing Report

Implementation



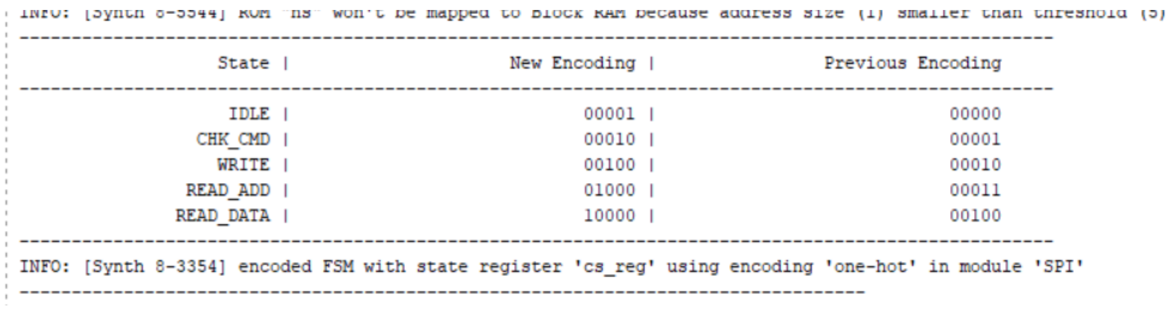
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.812 ns	Worst Hold Slack (WHS):	0.094 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:	4249	Total Number of Endpoints:	4249	Total Number of Endpoints:	2130

All user specified timing constraints are met.

Figure 5: Implementation with timing

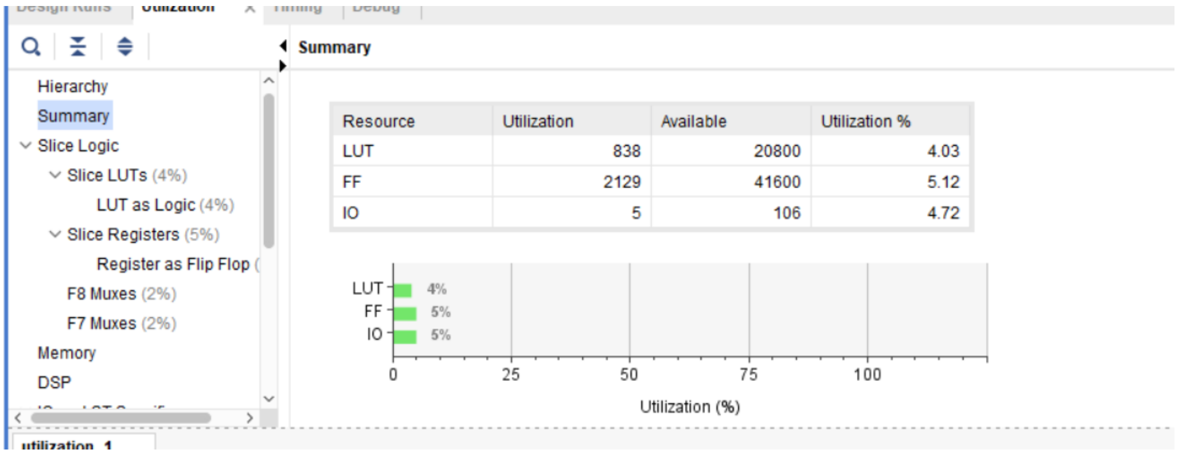
Enconding Used



```
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |                Previous Encoding
---------------------------------------------------------------------------------------------------
                    IDLE |                            00001 |                            00000
                 CHK_CMD |                            00010 |                            00001
                   WRITE |                            00100 |                            00010
                READ_ADD |                            01000 |                            00011
               READ_DATA |                            10000 |                            00100
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'SPI'
---------------------------------------------------------------------------------
```

Figure 6: One Hot Code Encoding is used.

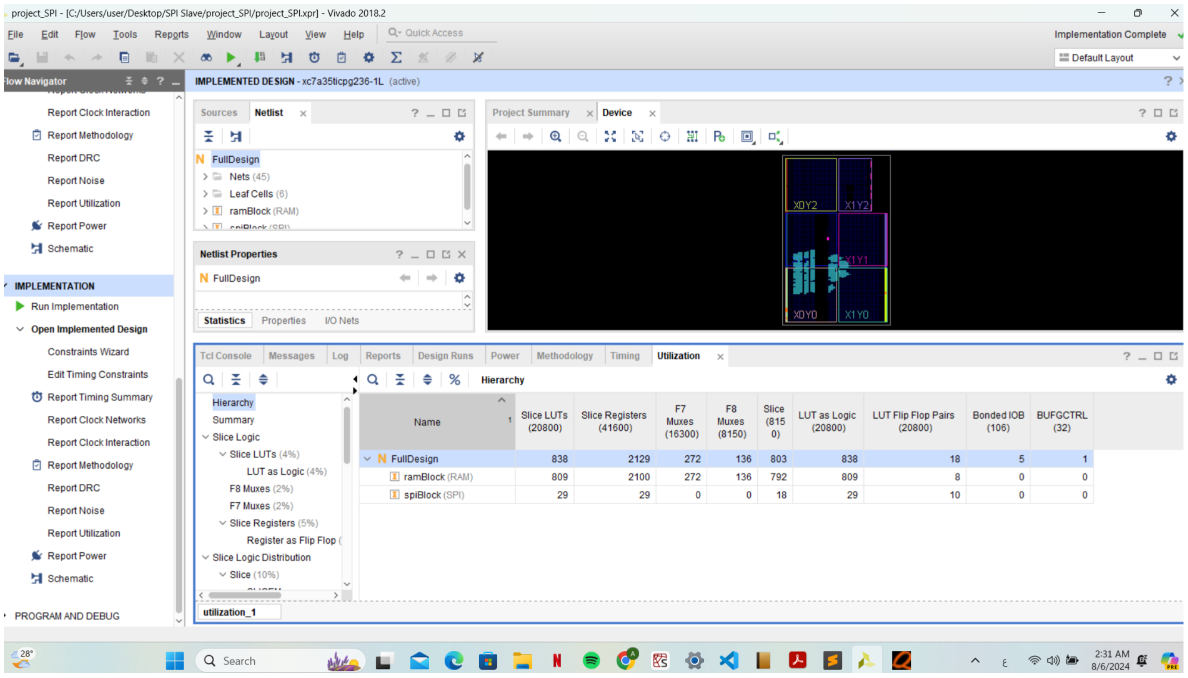
Utilization Reports



Summary

Resource	Utilization	Available	Utilization %
LUT	838	20800	4.03
FF	2129	41600	5.12
IO	5	106	4.72

Figure 7: Utilization Report of the Synthesis.



project_SPI - [C:/Users/user/Desktop/SPI Slave/project_SPI/project_SPI.xpr] - Vivado 2018.2

IMPLEMENTED DESIGN - xc7a35ticpg236-1L (active)

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
FullDesign	838	2129	272	136	803	838	18	5	1
ramBlock (RAM)	809	2100	272	136	792	809	8	0	0
spiBlock (SPI)	29	29	0	0	18	29	10	0	0

Figure 8: Utilization of the implementation.

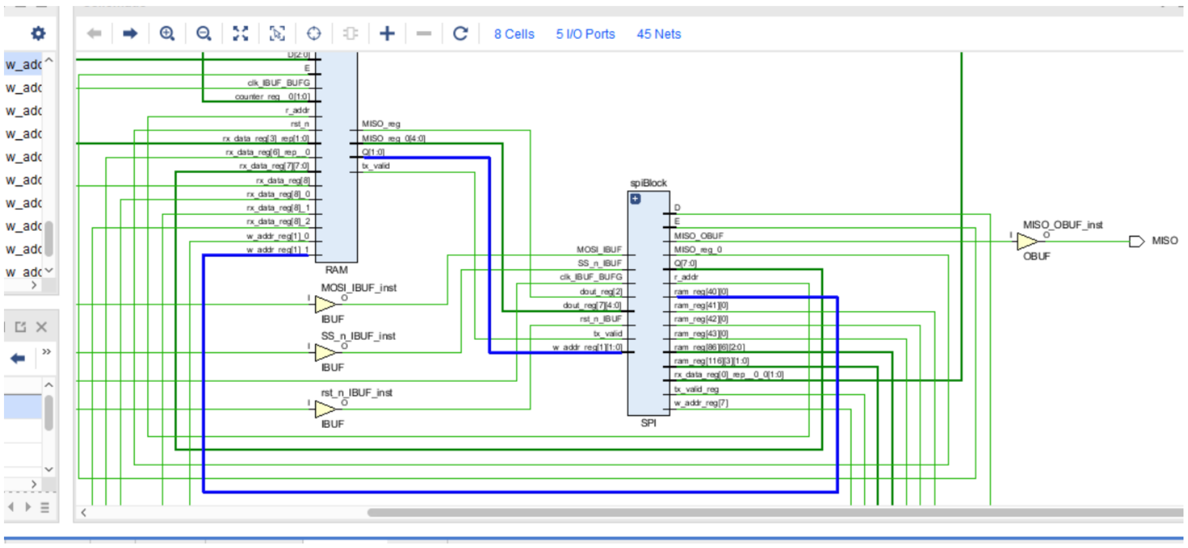


Figure 9: Critical Path

Notes

1. The do file did not work with me so I hope this will not affect the grading for the project. I searched on Reddit, youtube, and even chatGPT but I can't find any clue to the matter.
2. I hope nothing is missing and I am terribly sorry for the late submission.
3. I truly thank you for your guidance throughout the course and I can confidently say that I have learned so much from you and I hope this knowledge will put me on the right path of Digital IC Design.