

SPI Slave with Single Port RAM

Project_2

Name: Ahmed Nour

Email: ahmedmohamednoour@gmail.com

Table of Contents

1. RTL code	3
1. TOP_Module	3
2. SPI_MODULE	4
3. RAM_MODULE	7
2. Testbench code	9
3. Do file	11
4. QuestaSim Snippets	12
1. Messages Snippet	12
2. Waveform Snippet	12
5. Constraint File	13
6. Elaboration	14
1. Message tab	14
2. Schematic	14
7. Synthesis	15
1. Message Tab	15
2. Utilization report	15
3. Timing report	15
4. Schematic	16
5. Netlist	17
8. Implementation	18
1. Message Tab	18
2. Utilization report	18
3. Timing report	18
4. Device	19
5. Bitstream	19
9. Linting	20
1. Lint snippet	20
2. Lint Schematic	20
10. Comment	21

1. RTL code

1. TOP_Module

```
1 module SPI_SLAVE_WITH_SINGLE_PORT_RAM (MISO, MOSI, clk, rst_n, SS_n);
2     /*=====*/
3     //Input Ports of SPI_SLAVE_WITH_SINGLE_PORT_RAM
4     /*=====*/
5     //Data ports
6     input MOSI ;
7
8     //clock ports
9     input clk ;
10
11     //control ports
12     input SS_n ;
13     input rst_n ;
14
15     /*=====*/
16     //Output Ports of SPI_SLAVE_WITH_SINGLE_PORT_RAM
17     /*=====*/
18     //Data ports
19     output MISO ;
20
21     /*=====*/
22     //Internal Ports of SPI_SLAVE_WITH_SINGLE_PORT_RAM
23     /*=====*/
24     //Internal Data ports
25     wire [9:0] RAM_IN ;
26     wire [7:0] RAM_OUT ;
27
28     wire rx_valid ;
29     wire tx_valid ;
30
31     /*=====*/
32     // Instantiation of Modules
33     /*=====*/
34     //SPI Slave
35     SPI_Slave SPI1 (.clk(clk), .rst_n(rst_n), .SS_n(SS_n), .tx_valid(tx_valid), .MOSI(MOSI),
36                    .MISO(MISO), .tx_data(RAM_OUT), .rx_valid(rx_valid), .rx_data(RAM_IN));
37
38     //Single Port RAM
39     SINGLE_PORT_RAM RAM1 (.clk(clk), .rst_n(rst_n), .din(RAM_IN),
40                           .rx_valid(rx_valid), .dout(RAM_OUT), .tx_valid(tx_valid));
41
42 endmodule
```

2. SPI_MODULE

```
1 module SPI_Slave (clk, rst_n, SS_n, tx_valid, MOSI, MISO, tx_data, rx_valid, rx_data);
2 /*=====*/
3 // states
4 /*=====*/
5 parameter IDLE_STATE    = 3'b000;
6 parameter CMD_CHECK     = 3'b001;
7 parameter WRITE_MODE    = 3'b010;
8 parameter READ_ADDRESS  = 3'b011;
9 parameter READ_DATA     = 3'b100;
10 /*=====*/
11 // Input Ports
12 /*=====*/
13 input      clk      ;
14 input      rst_n    ;
15 input      SS_n     ;
16 input      tx_valid ;
17 input      MOSI     ;
18 input [7:0] tx_data ;
19 /*=====*/
20 // Output Ports
21 /*=====*/
22 output      MISO     ;
23 output reg   rx_valid ;
24 output reg [9:0] rx_data ;
25 /*=====*/
26 // Internals
27 /*=====*/
28 reg [2:0] CS      ;
29 reg [2:0] NS      ;
30 reg      read_flag ;
31 reg [3:0] counter ;
```

```
1 /*=====*/
2 // State Transition
3 /*=====*/
4 always @(posedge clk or negedge rst_n) begin
5     if (!rst_n)
6         CS <= IDLE_STATE;
7     else
8         CS <= NS;
9 end
```

```

1  /*=====*/
2  // Next State Logic
3  /*=====*/
4  always @(*) begin
5      case (CS)
6          IDLE_STATE: begin
7              if (!SS_n)
8                  NS = CMD_CHECK;
9              else
10                 NS = IDLE_STATE;
11          end
12
13          CMD_CHECK: begin
14              if (SS_n)
15                  NS = IDLE_STATE;
16              else if (MOSI == 0)
17                  NS = WRITE_MODE;
18              else if (MOSI == 1 && !read_flag)
19                  NS = READ_ADDRESS;
20              else
21                  NS = READ_DATA;
22          end
23
24          WRITE_MODE: begin
25              if (SS_n)
26                  NS = IDLE_STATE;
27              else
28                  NS = WRITE_MODE;
29          end
30
31          READ_ADDRESS: begin
32              if (SS_n)
33                  NS = IDLE_STATE;
34              else
35                  NS = READ_ADDRESS;
36          end
37
38          READ_DATA: begin
39              if (SS_n)
40                  NS = IDLE_STATE;
41              else
42                  NS = READ_DATA;
43          end
44
45          default: NS = IDLE_STATE;
46      endcase
47  end

```

```

1  /*=====*/
2  // Output Logic
3  /*=====*/
4  always @(posedge clk or negedge rst_n) begin
5      if (!rst_n) begin
6          rx_data    <= 10'b0;
7          rx_valid   <= 1'b0;
8          counter    <= 4'b0;
9          read_flag  <= 1'b0;
10     end
11     else begin
12         if (!SS_n) begin
13             rx_valid <= 1'b0;
14             if (CS == WRITE_MODE || CS == READ_ADDRESS) begin
15                 rx_data <= {rx_data[8:0], MOSI};
16                 if (counter == 9) begin
17                     rx_valid <= 1'b1;
18                     counter <= 4'b0;
19                     if (CS == READ_ADDRESS)
20                         read_flag <= 1'b1;
21                 end
22                 else begin
23                     counter <= counter + 1;
24                 end
25             end
26             else if (CS == READ_DATA) begin
27                 if (!tx_valid) begin
28                     rx_data <= {rx_data[8:0], MOSI};
29                     if (counter == 9) begin
30                         rx_valid <= 1'b1;
31                         counter <= 4'b0;
32                     end
33                     else begin
34                         counter <= counter + 1;
35                     end
36                 end
37                 else begin
38                     if (counter > 7) begin
39                         counter <= 4'b0;
40                         read_flag <= 1'b0;
41                     end
42                     else begin
43                         counter <= counter + 1;
44                     end
45                 end
46             end
47         end
48         else begin
49             rx_valid <= 1'b0;
50         end
51     end
52 end
53
54 assign MISO = (tx_valid && CS == READ_DATA) ? tx_data[8 - counter] : 0;
55
56 endmodule

```

3. RAM_MODULE

```
1 module SINGLE_PORT_RAM (din,rx_valid,clk,rst_n,dout,tx_valid);
2 /*=====*/
3 // Input Ports
4 /*=====*/
5 input  [9:0] din;
6 input  clk;
7 input  rst_n;
8 input  rx_valid;
9
10 /*=====*/
11 // Output Ports
12 /*=====*/
13 output reg [7:0] dout;
14 output reg tx_valid;
15
16 /*=====*/
17 // Parameters
18 /*=====*/
19 parameter MEM_DEPTH = 256;
20 parameter WIDTH = 8;
21
22 /*=====*/
23 // Internal Registers
24 /*=====*/
25 reg [WIDTH-1:0] WRITE;
26 reg [WIDTH-1:0] READ;
27 reg [7:0] mem [0:MEM_DEPTH-1];
```

```
1  /*=====*/
2  // Sequential Logic
3  /*=====*/
4  always @(posedge clk) begin
5      if (!rst_n) begin
6          dout <= 8'd0;
7          tx_valid <= 1'b0;
8          WRITE <= 0;
9          READ <= 0;
10     end
11
12     else begin
13         if (rx_valid) begin
14             case (din[9:8])
15                 2'b00: begin
16                     WRITE <= din[7:0];
17                     tx_valid <= 0;
18                 end
19
20                 2'b01: begin
21                     mem[WRITE] <= din[7:0];
22                     tx_valid <= 0;
23                 end
24
25                 2'b10: begin
26                     READ <= din[7:0];
27                     tx_valid <= 0;
28                 end
29
30                 2'b11: begin
31                     dout <= mem[READ];
32                     tx_valid <= 1;
33                 end
34             endcase
35         end
36     end
37 end
38
39 endmodule
```


2. Testbench code

```
1 module SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb();
2 /*=====*/
3 // Internal Signals
4 /*=====*/
5 reg mosi_tb;
6 reg clk_tb;
7 reg ss_n_tb;
8 reg rst_n_tb;
9
10 wire miso_tb;
11
12 integer i;
13
14 /*=====*/
15 // DUT Instantiation
16 /*=====*/
17 SPI_SLAVE_WITH_SINGLE_PORT_RAM DUT (.MISO(miso_tb), .MOSI(mosi_tb), .clk(clk_tb),
18                                     .rst_n(rst_n_tb), .SS_n(ss_n_tb));
19
20 /*=====*/
21 // Clock Generation
22 /*=====*/
23 initial begin
24     clk_tb = 0;
25     forever #1 clk_tb = ~clk_tb;
26 end
27 /*=====*/
28 // Stimulus
29 /*=====*/
30 initial begin
31 /*=====*/
32 // Reset
33 /*=====*/
34     $readmemh("mem.dat", DUT.RAM1.mem);
35     for (i = 0; i < 256; i = i + 1)
36         DUT.RAM1.mem[i] <= 8'd0;
37
38     rst_n_tb = 0;
39     mosi_tb = 1;
40     ss_n_tb = 1;
41     @(negedge clk_tb);@(negedge clk_tb);
```

```

1  /*=====*/
2  // Write Address
3  /*=====*/
4      rst_n_tb = 1; @(negedge clk_tb);
5      ss_n_tb = 0;
6      @(negedge clk_tb); @(negedge clk_tb);
7
8      //00111_11111
9      mosi_tb = 0; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
10     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
11     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
12     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
13     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
14
15
16     ss_n_tb = 1;
17     @(negedge clk_tb);
18     @(negedge clk_tb);
19
20 /*=====*/
21 // Write Data
22 /*=====*/
23     ss_n_tb = 0;
24     @(negedge clk_tb); @(negedge clk_tb);
25     //01101_01010
26     mosi_tb = 0; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
27     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
28     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
29     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
30     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
31
32
33     ss_n_tb = 1;
34     @(negedge clk_tb);
35     @(negedge clk_tb);
36
37 /*=====*/
38 // Read Address
39 /*=====*/
40     ss_n_tb = 0;
41     @(negedge clk_tb); @(negedge clk_tb);
42     //1010101101
43     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 0; @(negedge clk_tb);
44     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
45     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
46     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
47     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
48
49
50
51     ss_n_tb = 1;
52     @(negedge clk_tb);
53     @(negedge clk_tb);
54
55 /*=====*/
56 // Read Data
57 /*=====*/
58     ss_n_tb = 0;
59     @(negedge clk_tb); @(negedge clk_tb);
60
61     mosi_tb = 1; @(negedge clk_tb); mosi_tb = 1; @(negedge clk_tb);
62
63     repeat (16)begin
64         @(negedge clk_tb);
65     end
66
67     ss_n_tb = 1;
68     @(negedge clk_tb);
69     @(negedge clk_tb);
70
71     $display("Test Done");
72     $finish;
73 end
74
75 endmodule

```

3. Do file

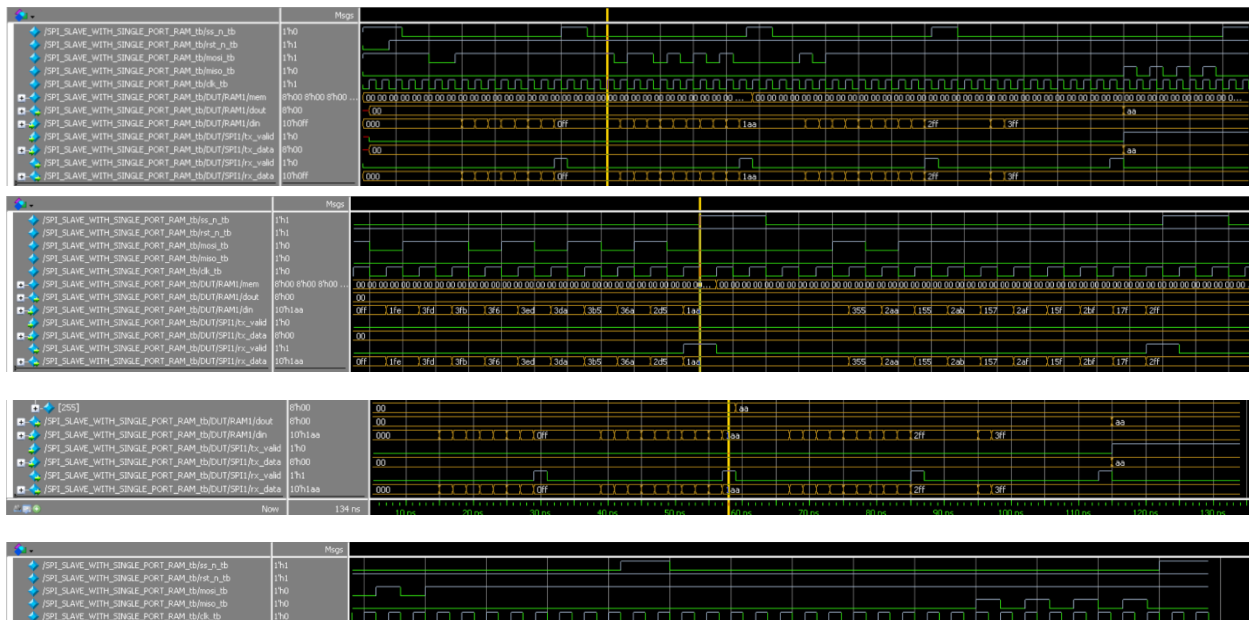
```
1  vlib work
2  vlog SPI_Slave.v SINGLE_PORT_RAM.v SPI_SLAVE_WITH_SINGLE_PORT_RAM.v SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb.v
3
4  vsim -voptargs=+acc work.SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb
5
6  add wave -position insertpoint \
7      sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/ss_n_tb \
8      sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/rst_n_tb \
9      sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/mosi_tb \
10     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/miso_tb \
11     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/clk_tb
12
13  add wave -position insertpoint \
14     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/RAM1/mem \
15     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/RAM1/dout \
16     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/RAM1/din
17
18  add wave -position insertpoint \
19     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/SPI1/tx_valid \
20     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/SPI1/tx_data \
21     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/SPI1/rx_valid \
22     sim:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb/DUT/SPI1/rx_data
23
24  run -all
25  #quit -sim
26
```

1. Messages Snippet

1. Messages Snippet

```
# Top level modules:
# SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb
# End time: 17:19:51 on Aug 20,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb
# Start time: 17:19:51 on Aug 20,2025
# ** Note: (vsim-3812) Design is being optimized...
# ** Note: (vopt-143) Recognized 1 FSM in module "SPI_Slave(fast)".
# Loading work.SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb(fast)
# Loading work.SPI_SLAVE_WITH_SINGLE_PORT_RAM(fast)
# Loading work.SPI_Slave(fast)
# Loading work.SINGLE_PORT_RAM(fast)
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
# File in use by: nour Hostname: NOUR ProcessID: 29948
# Attempting to use alternate WLF file "./wlf0r18k7".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
# Using alternate file: ./wlf0r18k7
# ** Warning: (vsim-7) Failed to open readmem file "mem.dat" in read mode.
# No such file or directory. (errno = ENOENT) : SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb.v(34)
# Time: 0 ns Iteration: 0 Instance: /SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb
# Test Done
# ** Note: #finish : SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb.v(114)
# Time: 134 ns Iteration: 1 Instance: /SPI_SLAVE_WITH_SINGLE_PORT_RAM_tb
```

2. Waveform Snippet



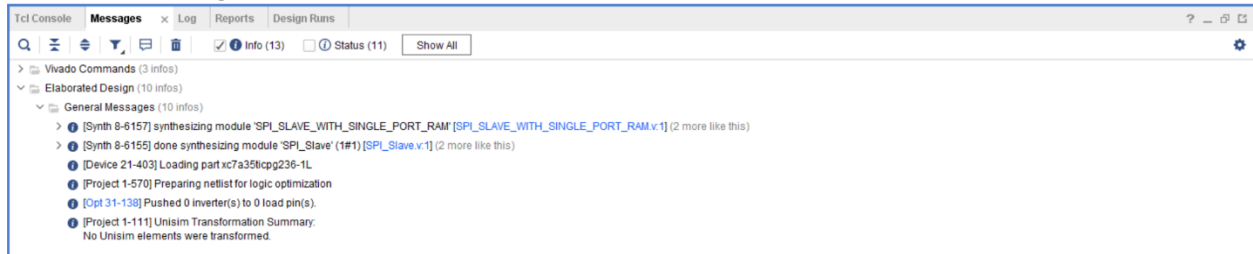
5. Constraint File

```
5
6  ## Clock signal
7  set_property -dict { PACKAGE_PIN W5      IOSTANDARD LVCMOS33 } [get_ports clk]
8  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
9
10
11 # Switches
12 set_property -dict { PACKAGE_PIN V17      IOSTANDARD LVCMOS33 } [get_ports {rst_n}]
13 set_property -dict { PACKAGE_PIN V16      IOSTANDARD LVCMOS33 } [get_ports {SS_n}]
14 set_property -dict { PACKAGE_PIN W16      IOSTANDARD LVCMOS33 } [get_ports {MOSI}]
15 #set_property -dict { PACKAGE_PIN W17      IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
16 #set_property -dict { PACKAGE_PIN W15      IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
17 #set_property -dict { PACKAGE_PIN V15      IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
18 #set_property -dict { PACKAGE_PIN W14      IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
19 #set_property -dict { PACKAGE_PIN W13      IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
20 #set_property -dict { PACKAGE_PIN V2       IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
21 #set_property -dict { PACKAGE_PIN T3       IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
22 #set_property -dict { PACKAGE_PIN T2       IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
23 #set_property -dict { PACKAGE_PIN R3       IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
24 #set_property -dict { PACKAGE_PIN W2       IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
25 #set_property -dict { PACKAGE_PIN U1       IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
26 #set_property -dict { PACKAGE_PIN T1       IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
27 #set_property -dict { PACKAGE_PIN R2       IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]
28
29
30 ## LEDs
31 set_property -dict { PACKAGE_PIN U16      IOSTANDARD LVCMOS33 } [get_ports {MISO}]
32 #set_property -dict { PACKAGE_PIN E19      IOSTANDARD LVCMOS33 } [get_ports {Led[1]}]
33 #set_property -dict { PACKAGE_PIN U19      IOSTANDARD LVCMOS33 } [get_ports {Led[2]}]
```

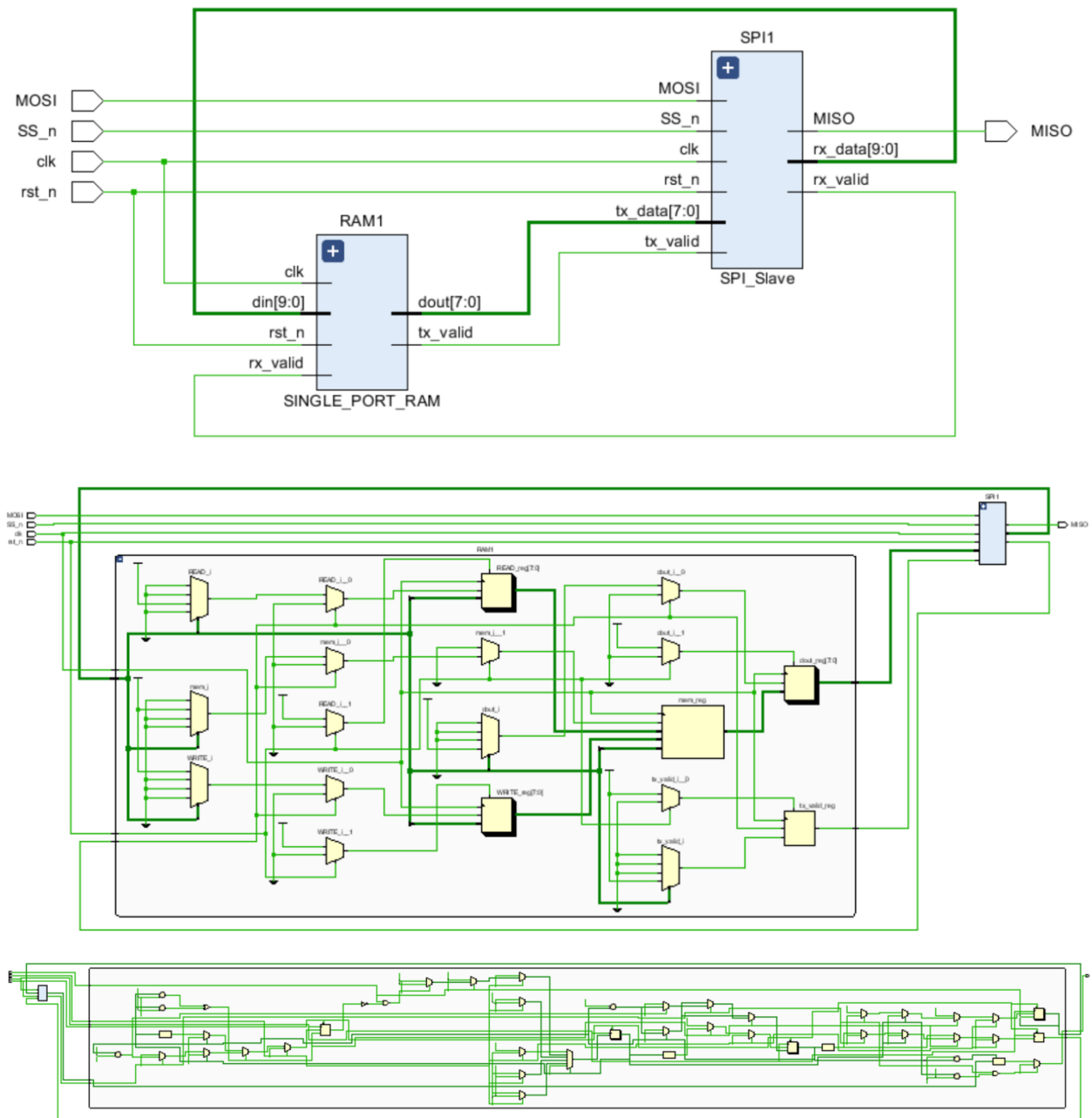
```
156 ## SPI configuration mode options for QSPI boot, can be used for all designs
157 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
158 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
159 set_property CONFIG_MODE SPIx4 [current_design]
160
161
162
163 create_debug_core u_ila_0 ila
164 set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
165 set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
166 set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
167 set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
168 set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
169 set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
170 set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
171 set_property C_TRIGGER_OUT false [get_debug_cores u_ila_0]
172 set_property port_width 1 [get_debug_ports u_ila_0/clk]
173 connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
174 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
175 set_property port_width 1 [get_debug_ports u_ila_0/probe0]
176 connect_debug_port u_ila_0/probe0 [get_nets [list clk_IBUF]]
177 create_debug_port u_ila_0 probe
178 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
179 set_property port_width 1 [get_debug_ports u_ila_0/probe1]
180 connect_debug_port u_ila_0/probe1 [get_nets [list MISO_0BUF]]
181 create_debug_port u_ila_0 probe
182 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
183 set_property port_width 1 [get_debug_ports u_ila_0/probe2]
184 connect_debug_port u_ila_0/probe2 [get_nets [list MOSI_IBUF]]
185 create_debug_port u_ila_0 probe
186 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
187 set_property port_width 1 [get_debug_ports u_ila_0/probe3]
188 connect_debug_port u_ila_0/probe3 [get_nets [list rst_n_IBUF]]
189 create_debug_port u_ila_0 probe
190 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
191 set_property port_width 1 [get_debug_ports u_ila_0/probe4]
192 connect_debug_port u_ila_0/probe4 [get_nets [list SS_n_IBUF]]
193 set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
194 set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
195 set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
196 connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
```

6. Elaboration

1. Message tab



2. Schematic



7. Synthesis

1. Message Tab

The Messages tab displays various messages related to the synthesis process. The messages are categorized into 'Vivado Commands', 'General Messages', and 'Synthesis'. The 'Synthesis' section contains several warnings and information messages, including:

- [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
- [Synth 8-6157] synthesizing module 'SPI_SLAVE_WITH_SINGLE_PORT_RAM' [SPI_SLAVE_WITH_SINGLE_PORT_RAM.v] (2 more like this)
- [Synth 8-6155] done synthesizing module 'SPI_Slave' (1#1) [SPI_Slave.v] (2 more like this)
- [Device 21-403] Loading part xc7a35tcp236-1L
- [Project 1-236] Implementation specific constraints were found while reading constraint file [D:/other/Courses/Kareem_waseem_Digital_Design/Projects/Fantastic_Three_Project_2/Synthesis/Constraint/Constraints_baays3 - Copy.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [X:/SPI_SLAVE_WITH_SINGLE_PORT_RAM_propmpt.xdc]. Resolution: To avoid this warning, move constraints listed in [Undefined] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.
- [Synth 8-802] inferred FSM for state register 'CS_reg' in module 'SPI_Slave'
- [Synth 8-5544] ROM 'm_valid' won't be mapped to Block RAM because address size (4) smaller than threshold (5) (6 more like this)
- [Synth 8-3354] encoded FSM with state register 'CS_reg' using encoding 'one-hot' in module 'SPI_Slave'
- [Synth 8-4480] The timing for the instance L5RAM1/mem_reg (implemented as a block RAM) might be sub-optimal as no optional output register could be merged into the block ram. Providing additional output register may help in improving timing. (1 more like this)
- [Project 1-571] Translating synthesized netlist
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 1 CPU seconds
- [Project 1-570] Preparing netlist for logic optimization (1 more like this)
- [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
- [Project 1-111] Unisim Transformation Summary: No Unisim elements were transformed. (1 more like this)
- [Common 17-83] Releasing license: Synthesis
- [Constraints 18-5210] No constraint will be written out.
- [Common 17-1381] The checkpoint D:/other/Courses/Kareem_waseem_Digital_Design/Projects/Fantastic_Three_Project_2/Synthesis/SPI/SPI_runs/synth_1/SPI_SLAVE_WITH_SINGLE_PORT_RAM.dcp has been generated.
- [nuntc-4] Executing : report_utilization -file SPI_SLAVE_WITH_SINGLE_PORT_RAM_utilization_synth.rpt -pb SPI_SLAVE_WITH_SINGLE_PORT_RAM_utilization_synth.pb
- [Common 17-206] Exiting Vivado at Wed Aug 20 17:40:23 2025...

The 'Synthesized Design' section shows general messages related to the netlist and timing analysis.

2. Utilization report

The Utilization report shows the resource usage of the design. The table below summarizes the key metrics:

Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_SLAVE_WITH_SING...	25	38	0.5	5	1
RAM1 (SINGLE_PORT...	3	17	0.5	0	0
SPI1 (SPI_Slave)	22	21	0	0	0

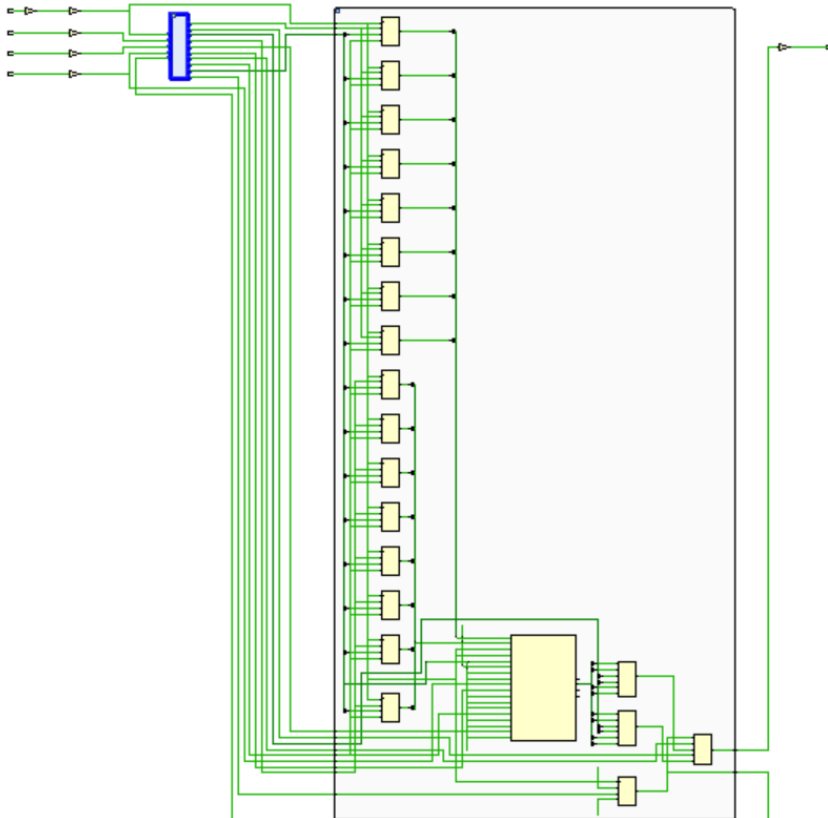
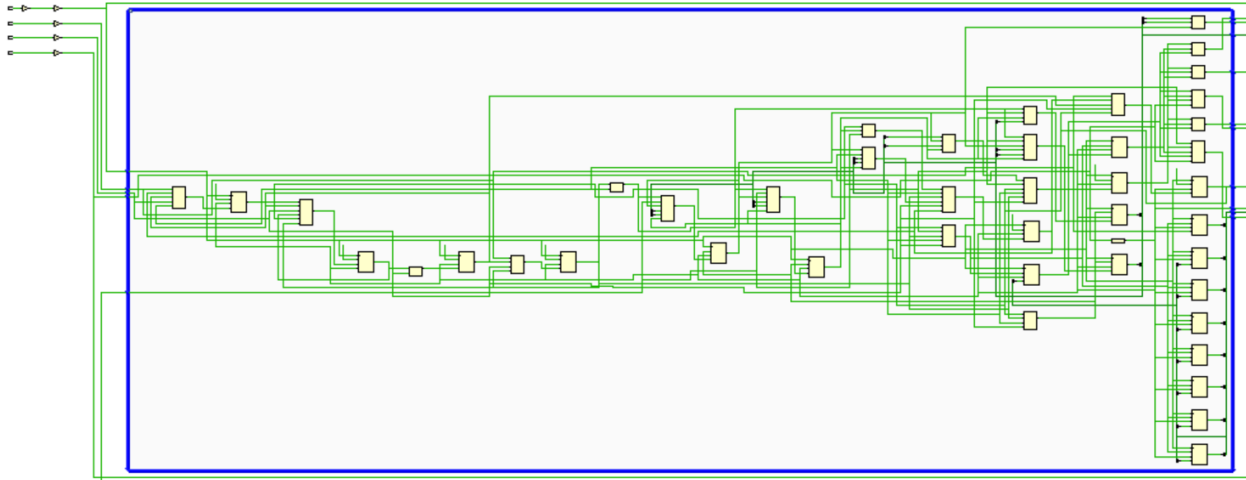
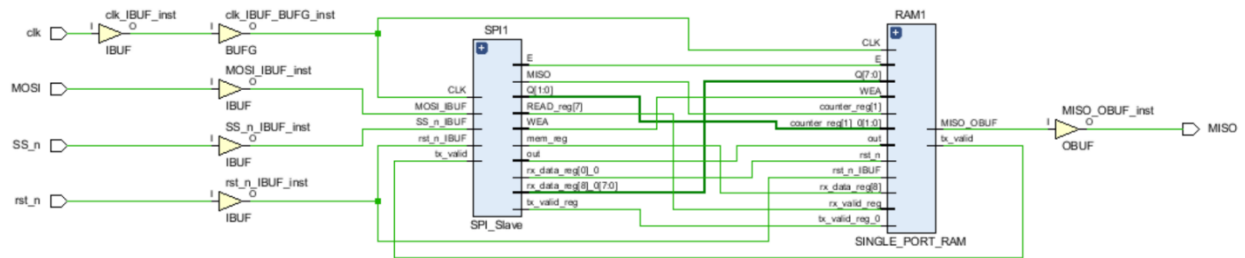
3. Timing report

The Timing report provides a summary of the design's timing performance. The key metrics are as follows:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.968 ns	Worst Hold Slack (WHS): 0.149 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: 41

All user specified timing constraints are met.

4. Schematic



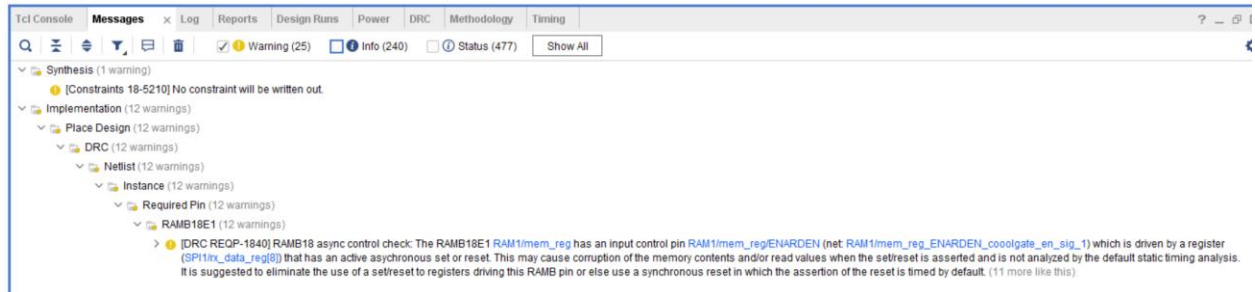
5. Netlist

```
14 `timescale 1 ps / 1 ps
15
16 module SINGLE_PORT_RAM
17 (tx_valid,
18  MISO_OBUF,
19  CLK,
20  rst_n_IBUF,
21  \rx_data_reg[8] ,
22  rst_n,
23  Q,
24  WEA,
25  tx_valid_reg_0,
26  out,
27  \counter_reg[1] ,
28  \counter_reg[1]_0 ,
29  E,
30  rx_valid_reg);
31 output tx_valid;
32 output MISO_OBUF;
33 input CLK;
34 input rst_n_IBUF;
35 input \rx_data_reg[8] ;
36 input rst_n;
37 input [7:0]Q;
38 input [0:0]WEA;
39 input tx_valid_reg_0;
40 input [0:0]out;
```

```
868 .INIT(64'h00000000AAAAA20))
869 rx_valid_i_1
870 (.I0(rx_valid2_in),
871  .I1(tx_valid),
872  .I2(out),
873  .I3(\FSM_onehot_CS_reg_n_0_[3] ),
874  .I4(\FSM_onehot_CS_reg_n_0_[2] ),
875  .I5(SS_n_IBUF),
876  .O(rx_valid_i_1_n_0));
877 (* SOFT_HLUTNM = "soft_Lutpair1" *)
878 LUT4 #(
879  .INIT(16'h1000))
880 rx_valid_i_2
881 (.I0(Q[1]),
882  .I1(\counter_reg_n_0_[2] ),
883  .I2(Q[0]),
884  .I3(counter1),
885  .O(rx_valid2_in));
886 FDCE #(
887  .INIT(1'b0))
888 rx_valid_reg
889 (.C(CLK),
890  .CE(\<const1> ),
891  .CLR(\rx_data_reg[0]_0 ),
892  .D(rx_valid_i_1_n_0),
893  .Q(rx_valid));
894 (* SOFT_HLUTNM = "soft_Lutpair0" *)
895 LUT5 #(
896  .INIT(32'hE2220000))
897 tx_valid_i_1
898 (.I0(tx_valid),
899  .I1(rx_valid),
900  .I2(RAM_IN[9]),
901  .I3(RAM_IN[8]),
902  .I4(rst_n_IBUF),
903  .O(tx_valid_reg));
904 endmodule
```

8. Implementation

1. Message Tab



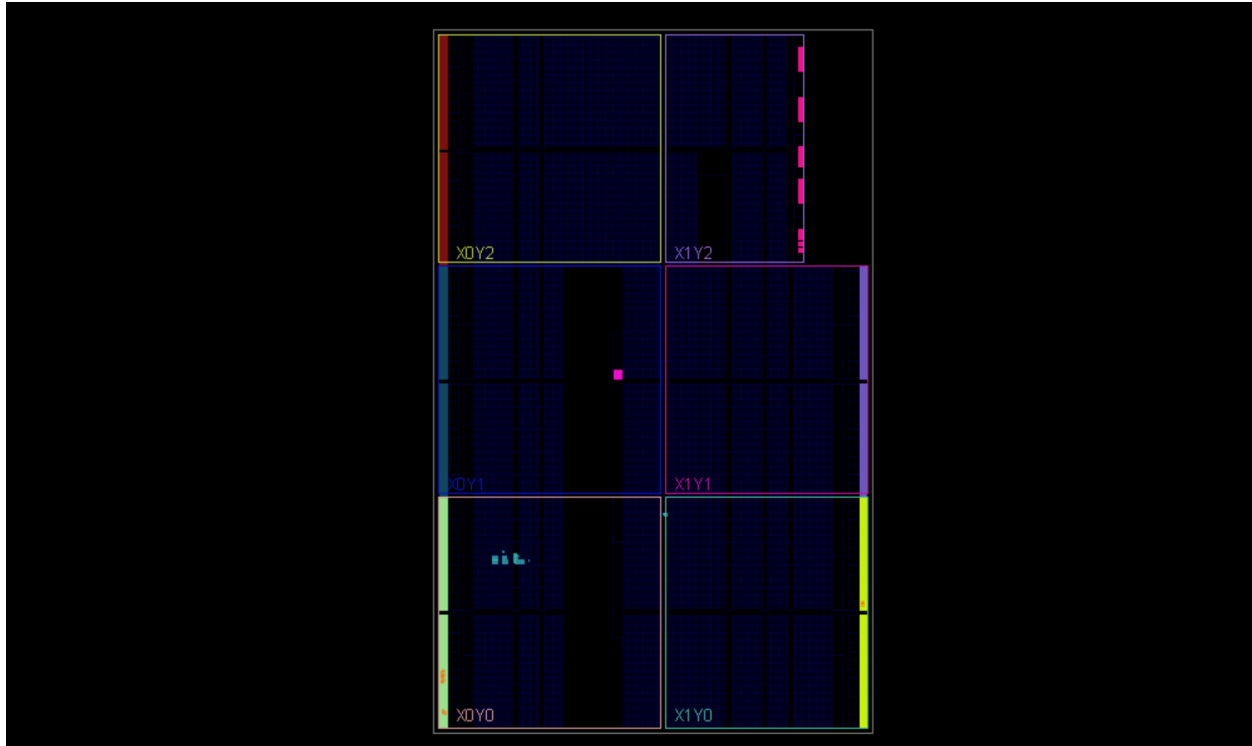
2. Utilization report

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (815 0)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_SLAVE_WITH_SING...	26	38	13	26	11	0.5	5	1
RAM1 (SINGLE_PORT...	4	17	7	4	0	0.5	0	0
SPI1 (SPI_Slave)	22	21	8	22	11	0	0	0

3. Timing report

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 7.271 ns	Worst Hold Slack (WHS): 0.072 ns	Worst Pulse Width Slack (VPWS): 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: 95	Total Number of Endpoints: 95	Total Number of Endpoints: 41

4. Device



5. Bitstream

lets (33)
Leaf Cells (6)
RAM1 (SINGLE_PORT_RAM)
IP11 (SPI_Slave)

Select an object to see properties

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

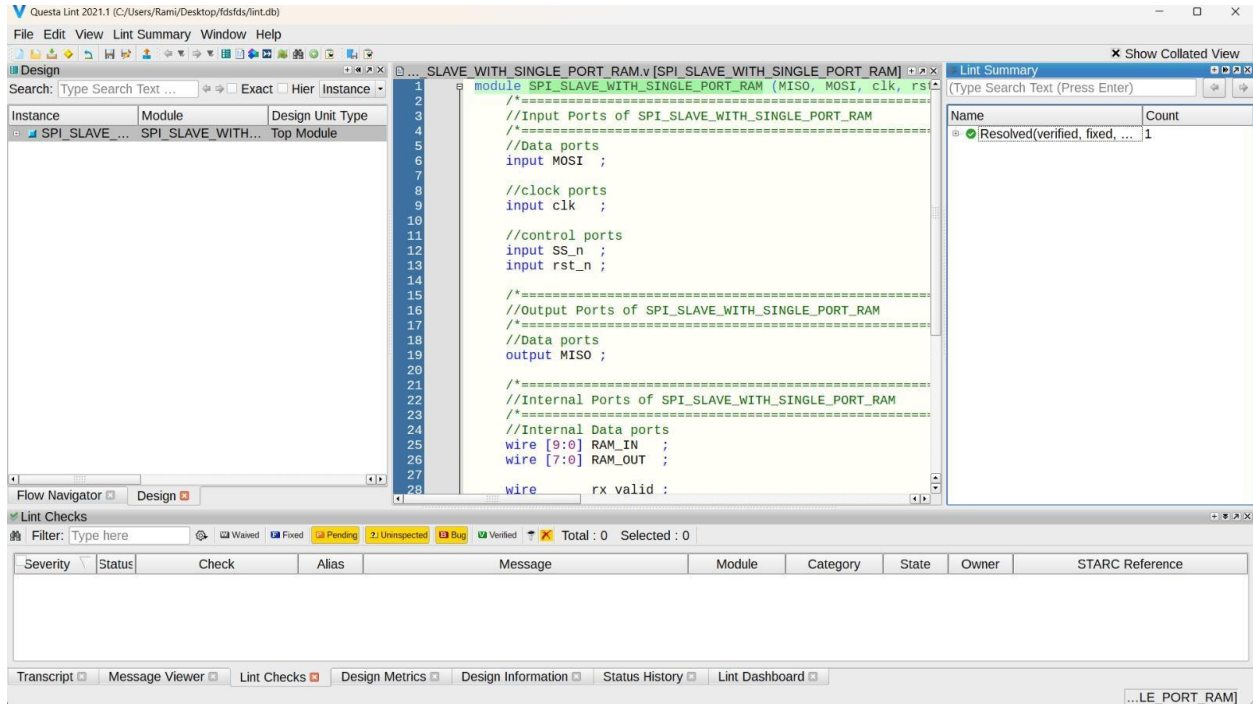
sole Messages Log Reports Design Runs Power DRC Methodology Timing x Utilization

Design Timing Summary

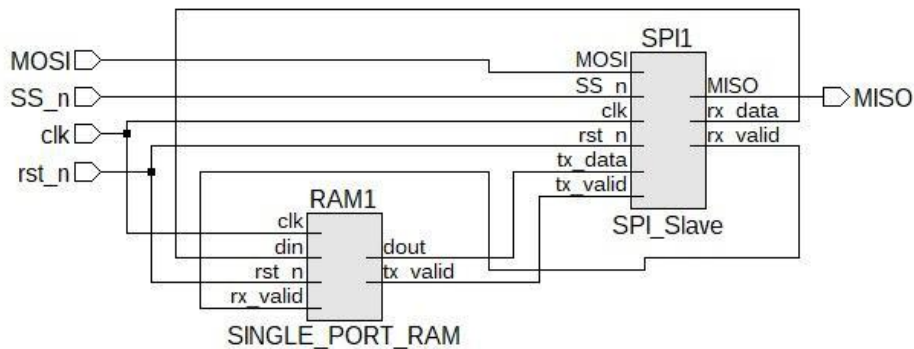
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 7.271 ns	Worst Hold Slack (WHS): 0.072 ns	Worst Pulse Width Slack (WPWS): 4.500 ns

9. Linting

1. Lint snippet

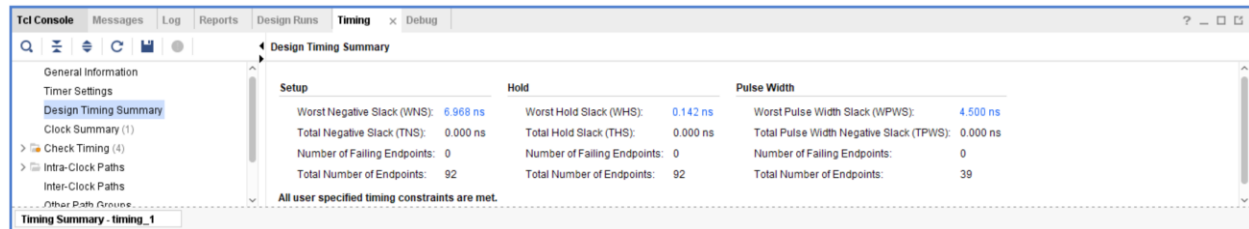


2. Lint Schematic



10. Comment

Gray encoding:

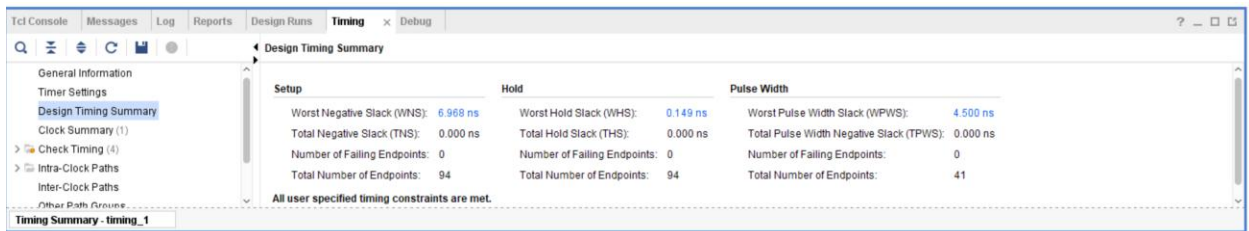


The screenshot shows the 'Design Timing Summary' window for a Gray encoding design. The left sidebar lists 'General Information', 'Timer Settings', 'Design Timing Summary' (selected), 'Clock Summary (1)', 'Check Timing (4)', 'Intra-Clock Paths', and 'Inter-Clock Paths'. The main area displays a table with three columns: Setup, Hold, and Pulse Width. The data is as follows:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.968 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: 92	Total Number of Endpoints: 92	Total Number of Endpoints: 39

At the bottom, it states: 'All user specified timing constraints are met.'

One_hot encoding:



The screenshot shows the 'Design Timing Summary' window for a One_hot encoding design. The left sidebar lists 'General Information', 'Timer Settings', 'Design Timing Summary' (selected), 'Clock Summary (1)', 'Check Timing (4)', 'Intra-Clock Paths', and 'Inter-Clock Paths'. The main area displays a table with three columns: Setup, Hold, and Pulse Width. The data is as follows:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.968 ns	Worst Hold Slack (WHS): 0.149 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: 41

At the bottom, it states: 'All user specified timing constraints are met.'

So, in terms of the highest frequency possible it would be gray encoding.