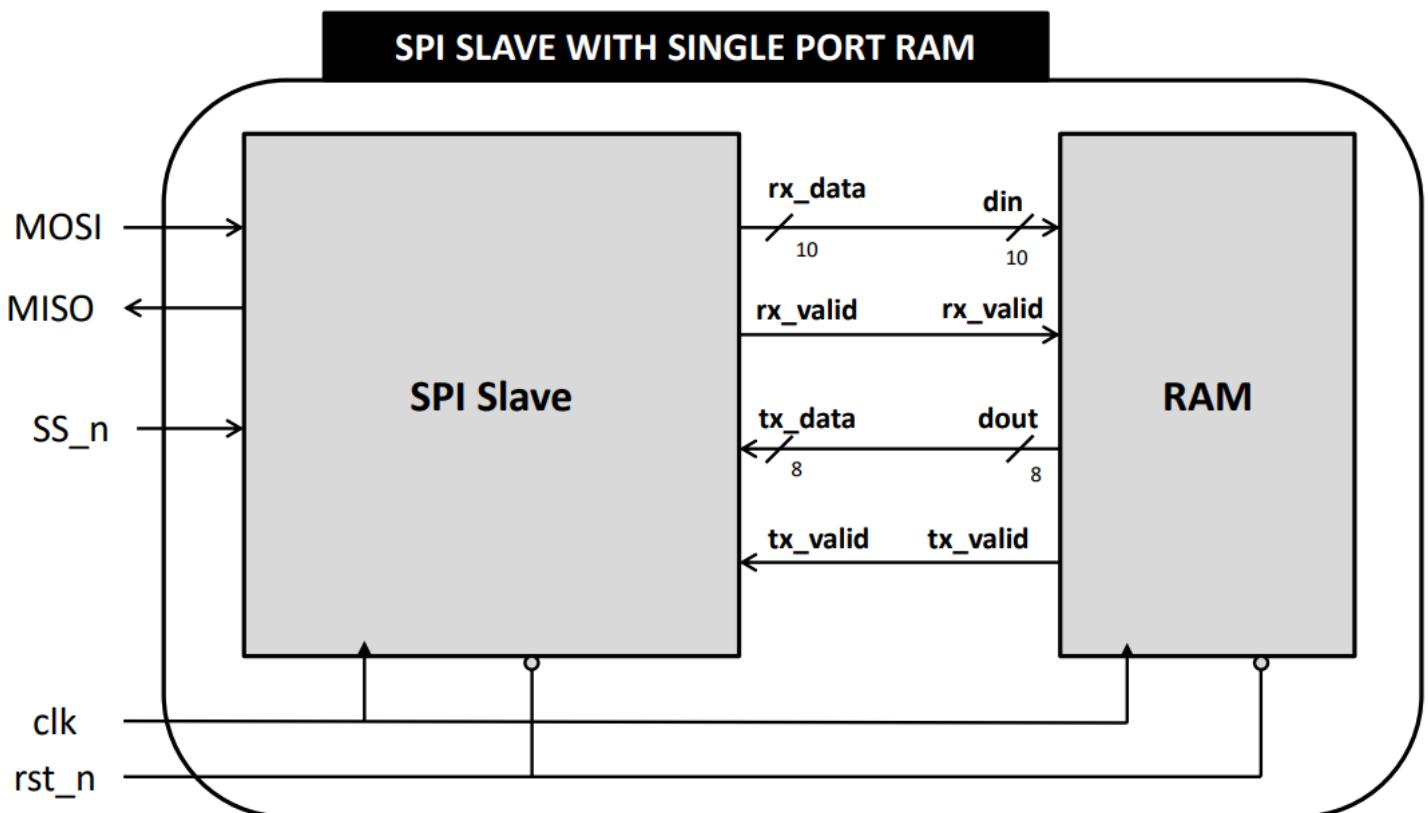


FINAL PROJECT

SPI SLAVE Interface

By: Khaled Ahmed Hamed

Under Supervision of Eng. Kareem Waseem



Introduction

The "SPI Slave with Single Port RAM" project implements an SPI slave module that communicates with a single-port RAM. The SPI slave receives data from a master device, storing and retrieving data to and from the RAM.

For more details, visit the [GitHub Repo](#)

Single Port Async. RAM

The single port asynchronous RAM module implements a memory block with a single data port.

It has the following parameters:

- `MEM_DEPTH` (default: 256): Depth of the memory.
- `ADDR_SIZE` (default: 8): Size of the memory address.

Ports:

Name	Type	Size	Description
clk	Input	1 bit	Clock signal
rst_n	Input	1 bit	Active low reset signal
din	Input	10 bit	Data input
rx_valid	Input	1 bit	If HIGH, accepts din[7:0] to save the write/read address internally or writes a memory word depending on the most significant 2 bits din[9:8]
dout	Output	8 bit	Data output
tx_valid	Output	1 bit	Whenever the command is a memory read, tx_valid should be HIGH

The most significant bits of din (din[9:8]) determine the operation to be performed:

Port	din[9:8]	Command	Description
din	00	Write	Holds din[7:0] internally as a write address
din	01	Write	Writes din[7:0] to the memory with the write address held previously
din	10	Read	Holds din[7:0] internally as a read address
din	11	Read	Reads the memory with the read address held previously. tx_valid should be HIGH, and dout holds the word read from the memory. din[7:0] is ignored

RTL Code Snippet:

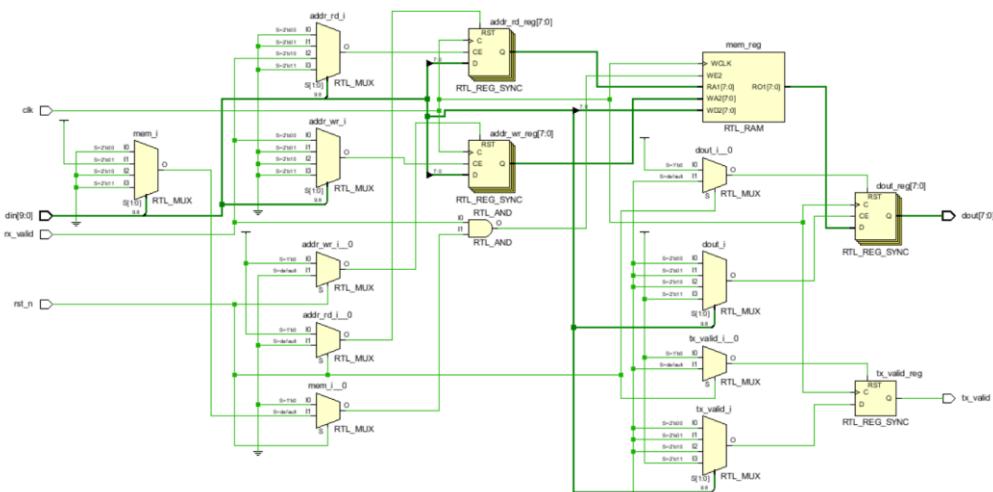
```

1  module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);
2  //Ports
3  input [9:0] din ;
4  input clk,rst_n,rx_valid ;
5  output reg [7:0] dout ;
6  output reg tx_valid ;
7  //Parameters
8  parameter MEM_DEPTH = 256 ;
9  parameter ADDR_SIZE = 8 ;
10 //Addresses
11 reg [ADDR_SIZE-1:0] addr_rd , addr_wr ;
12 //Memory , Memory Width >> 8 Bits as ADDR_SIZE & din 10 bits & 1st 2 bits are for protocol only
13 reg [7:0] mem [MEM_DEPTH-1:0] ;
14 //RAM Functionality , Active Low Async. rst , but in vivado async. signals make problems in synthesis so its sync. here
15 always @(posedge clk) begin
16     if (~rst_n) begin
17         dout<=0;
18         tx_valid<=0;
19         addr_rd<=0;
20         addr_wr<=0;
21     end
22     else begin
23         case(din[9:8])
24             2'b00: begin
25                 //Hold Write Address
26                 tx_valid<=0;
27                 if (rx_valid) begin
28                     addr_wr<=din[7:0];
29                 end
30             end
31             2'b01: begin
32                 //Write in the memory with address held previously
33                 tx_valid<=0;
34                 if (rx_valid) begin
35                     mem[addr_wr]<=din[7:0];
36                 end
37             end
38             2'b10: begin
39                 //Hold Read Address
40                 tx_valid<=0;
41                 if (rx_valid) begin
42                     addr_rd=din[7:0];
43                 end
44             end
45             2'b11: begin
46                 //Readt the memory with address held previously , din[7:0] Is dummy & Raise tx_valid high
47                 dout<=mem[addr_rd];
48                 tx_valid<=1;
49             end
50         endcase
51     end
52 end
53 //tx_valid could be also calculated as : assign tx_valid = din[9] & din[8] , Out of the always block
54 endmodule

```

Elaboration:

Schematic:

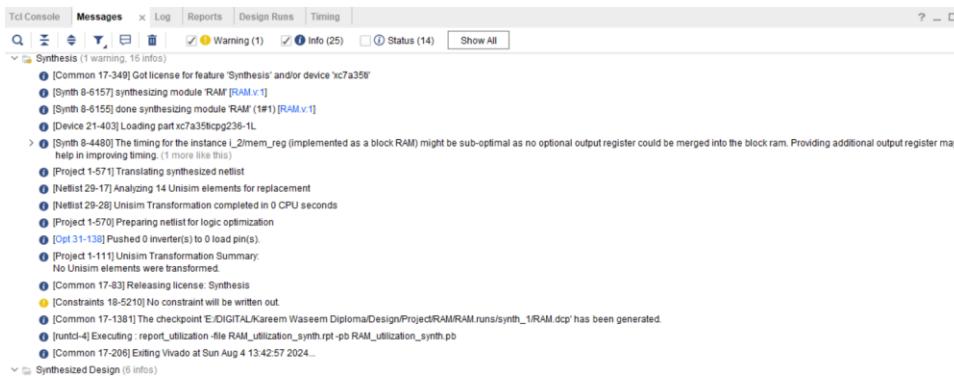


Messages Tab:

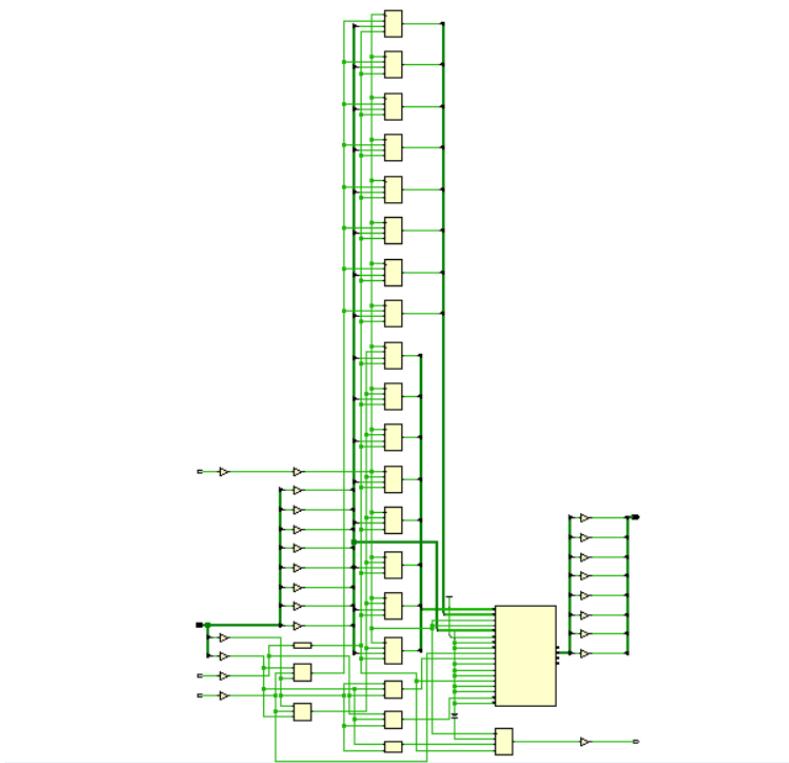


Synthesis:

Messages Tab:



Schematic:



Constraints File:

```
1  # Clock signal
2  set_property -dict { PACKAGE_PIN W5    IO_STANDARD LVCMS33 } [get_ports clk]
3  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
4
5  ##Buttons
6  set_property -dict { PACKAGE_PIN U18    IO_STANDARD LVCMS33 } [get_ports rst_n]
7
8  ## Configuration options, can be used for all designs
9  set_property CONFIG_VOLTAGE 3.3 [current_design]
10 set_property CFGVB VCCO [current_design]
11
12 ## SPI configuration mode options for QSPI boot, can be used for all designs
13 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
14 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
15 set_property CONFIG_MODE SPIx4 [current_design]
```

Timing Report Summary:

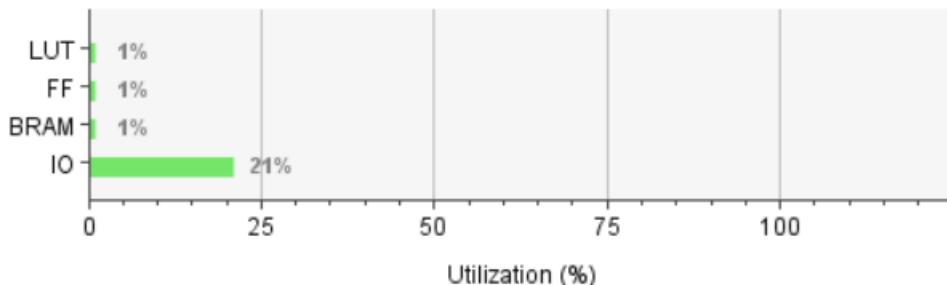
Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 7.823 ns	Worst Hold Slack (WHS): 0.204 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: 16	Total Number of Endpoints: 16	Total Number of Endpoints: 20

All user specified timing constraints are met.

Utilization Report:

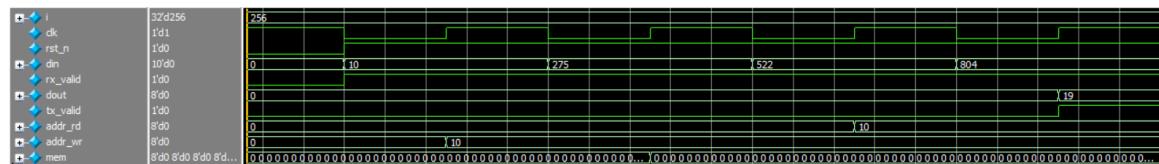
Resource	Utilization	Available	Utilization %
LUT	6	20800	0.03
FF	17	41600	0.04
BRAM	0.50	50	1.00
IO	22	106	20.75



Testbench Code Snippet:

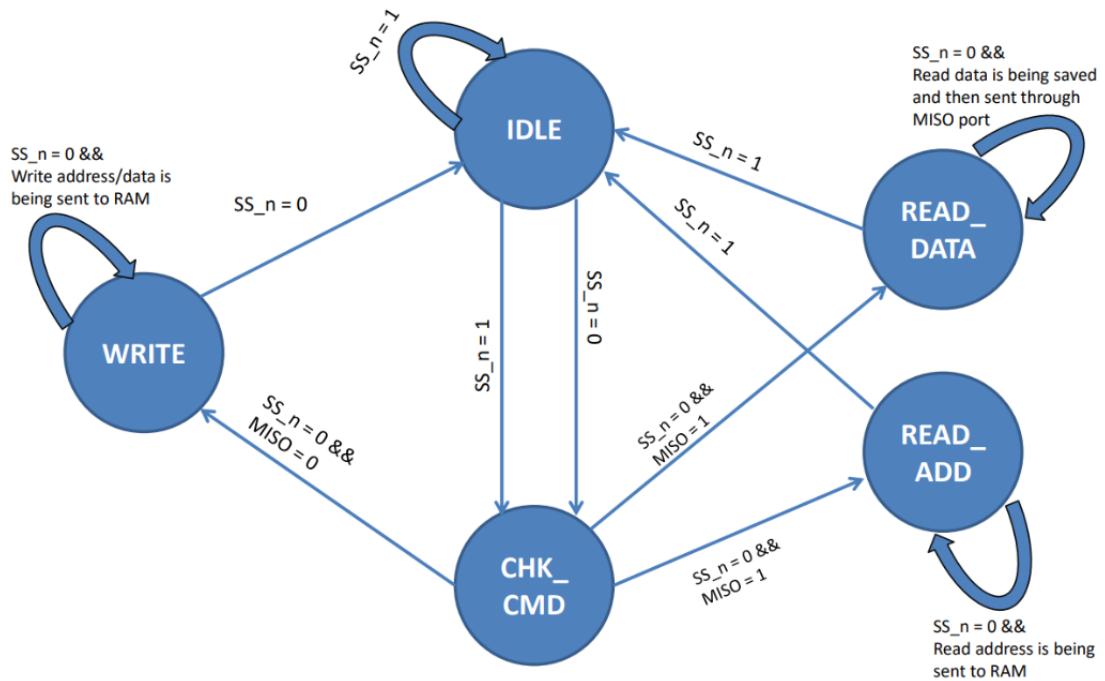
```
1  `timescale 1ns/1ps
2  module RAM_tb ();
3  //Signals Declaration
4  reg [9:0] din ;
5  reg clk,rst_n,rx_valid ;
6  wire [7:0] dout ;
7  wire tx_valid ;
8  integer i = 0;
9  //Clock Generation
10 initial begin
11   clk=0;
12   forever #5 clk=~clk;//Period=10ns
13 end
14 //DUT Instantiation
15 RAM #( .MEM_DEPTH(256), .ADDR_SIZE(8) ) DUT (.din(din),.clk(clk),.rst_n(rst_n),.rx_valid(rx_valid),.dout(dout),.tx_valid(tx_valid));
16 //Test Stimulus Generator
17 initial begin
18   //Initializing Memory
19   for(i=0;i<256;i=i+1)
20     DUT.mem[i]=0;
21   //Activate rst_n & Initialize all signals
22   rst_n = 0;
23   din=0;
24   rx_valid=0;
25   @(negedge clk);
26   //De-Activate rst_n & Start Testing
27   rst_n=1;
28   //Test Holding din[7:0] as a Write Address
29   rx_valid=1;
30   din[9:8]=2'b00;
31   din[7:0]=8'd10;//Memory Location Where data goes to
32   @(negedge clk);
33   //Test Writing in the memory with address held previously
34   rx_valid=1;
35   din[9:8]=2'b01;
36   din[7:0]=8'd19;//Data which will bw stored in memory
37   @(negedge clk);
38   //Test Holding din[7:0] as a Read Address
39   rx_valid=1;
40   din[9:8]=2'b10;
41   din[7:0]=8'd10;//Read data from this memory location
42   @(negedge clk);
43   //Test Writing in the memory with address held previously & Raising tx_valid high
44   /*rx_valid Will not affect neither is 0 nor 1*/
45   /*Expected Output is 19 in location 10*/
46   din[9:8]=2'b11;
47   din[7:0]=\$random;//Dummy Data
48   @(negedge clk);
49   $stop;
50 end
51 //Test Monitor & Results
52 initial begin
53   $monitor(`clk=%d,rst_n=%d,rx_valid=%d,din=%d,dout=%d,tx_valid=%d",clk,rst_n,rx_valid,din,dout,tx_valid);
54 end
55 endmodule
```

Simulation Waveform Snippet:



SPI SLAVE

SPI SLAVE State Transition Diagram



Introduction to SPI SLAVE Module:

The SPI slave module effectively manages data transfer between a master and a slave device. It uses state machines to control the flow of operations, ensuring proper read and write transactions based on the SPI protocol. The counters and flags aid in converting data formats and ensuring synchronization between the master and slave devices.

RTL Code Snippets:

```

1  module SPI_Slave (MOSI,MISO,SS_n,clk,rst_n,rx_data,rx_valid,tx_data,tx_valid);
2  //Ports
3  input MOSI,SS_n,clk,rst_n,rx_data,rx_valid;
4  input [7:0] tx_data;
5  output reg [9:0] rx_data;
6  output reg rx_valid,MISO;
7  //Internal Signals
8  reg [3:0] serial_to_parallel_counter;//Data Converted from serial to parallel in 10 clock cycles
9  reg [2:0] parallel_to_serial_counter;//Data Converted from parallel to serial in 8 clock cycles
10 reg addr_read_rec;//Address Read Received >> Ensures that reading address is done before reading data
11 //CS & NS
12 (* fsm_encoding="sequential")*
13 reg [2:0] cs ,ns;
14 //States
15 parameter IDLE = 3'b000;
16 parameter CHK_CMD = 3'b001;
17 parameter WRITE = 3'b010;
18 parameter READ_ADD = 3'b011;
19 parameter READ_DATA = 3'b100;
20 //Next State Logic (Comb)
21 always @(*) begin//always@(cs,MOSI,tx_data,tx_valid)
22     case(cs)
23         IDLE:begin
24             if (~SS_n) begin
25                 ns=CHK_CMD;
26             end
27             else begin
28                 ns=IDLE;
29             end
30         end
31         CHK_CMD:begin
32             if (SS_n) begin
33                 ns=IDLE;
34             end
35             else begin
36                 if (~MOSI) begin
37                     ns=WRITE;
38                 end
39                 else begin
40                     if (~addr_read_rec) begin
41                         ns=READ_ADD;
42                     end
43                     else begin
44                         ns=READ_DATA;
45                     end
46                 end
47             end
48         end

```

```

49         WRITE:begin
50             if (SS_n) begin
51                 ns=IDLE;
52             end
53             else begin
54                 ns=WRITE;
55             end
56         end
57         READ_ADD:begin
58             if (SS_n) begin
59                 ns=IDLE;
60             end
61             else begin
62                 ns=READ_ADD;
63             end
64         end
65         READ_DATA:begin
66             if (SS_n) begin
67                 ns=IDLE;
68             end
69             else begin
70                 ns=READ_DATA;
71             end
72         end
73         default : ns = IDLE;
74     endcase
75 end
76 //State Memory (Seq)
77 always @(posedge clk) begin
78     if (~rst_n) begin
79         cs<=IDLE;
80     end
81     else begin
82         cs<=ns;
83     end
84 end
85 //Output Logic (Seq)
86 always @(posedge clk) begin
87     if (~rst_n) begin
88         rx_data<=0;
89         rx_valid<=0;
90         MISO<=0;
91         addr_read_rec<=0;
92         serial_to_parallel_counter<=0;
93         parallel_to_serial_counter<=0;
94     end
95     else begin
96         case(cs)
97             IDLE:begin
98                 rx_valid<=0;
99                 MISO<=0;
100                serial_to_parallel_counter<=0;
101                parallel_to_serial_counter<=0;
102            end

```

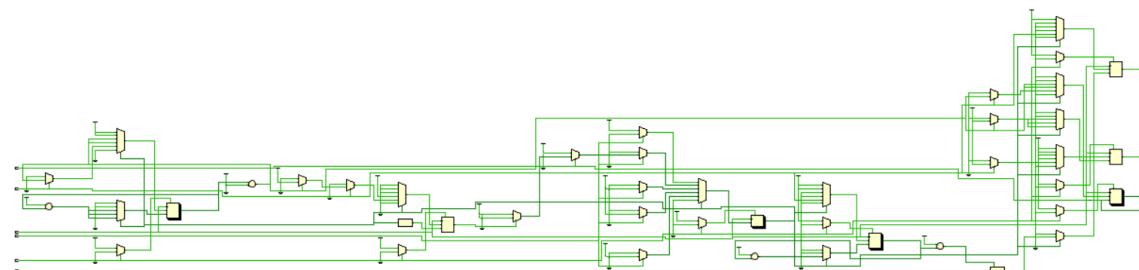
```

103      CHK_CWD;
104      begin
105          rx_valid<=0;
106          serial_to_parallel_counter<=0;
107          parallel_to_serial_counter<=0;
108      end
109      WRITE:begin
110          if (serial_to_parallel_counter<10) begin
111              rx_data <= {rx_data[8:0],MOSI};//Could be calculated as shift register or Using Up Counter : rx_data[9-serial_to_parallel_counter]<=MOSI;
112              serial_to_parallel_counter<=serial_to_parallel_counter+1;
113              rx_valid<=0;
114          end
115          else begin
116              rx_valid<=1;//Conversion is done
117          end
118      end
119      READ_ADD:begin
120          addr_read_rec<-1;
121          if (serial_to_parallel_counter<10) begin
122              rx_data <= {rx_data[8:0],MOSI};//Could be calculated as shift register or Using Up Counter : rx_data[9-serial_to_parallel_counter]<=MOSI;
123              serial_to_parallel_counter<=serial_to_parallel_counter+1;
124              rx_valid<=0;
125          end
126          else begin
127              rx_valid<=1;//Conversion is done
128          end
129      end
130      READ_DATA:begin
131          if (tx.valid && parallel_to_serial_counter<8) begin
132              MISO <= tx.data[7 - parallel_to_serial_counter];
133              parallel_to_serial_counter <= parallel_to_serial_counter + 1;
134              /*could be calculated as shift register or Using Up Counter : MISO <= tx.data[7];
135              tx_data <= tx.data << 1; */
136          end
137          else begin
138              if (serial_to_parallel_counter<10) begin
139                  rx_data <= {rx_data[8:0],MOSI};//Could be calculated as shift register or Using Up Counter : rx_data[9-serial_to_parallel_counter]<=MOSI;
140                  serial_to_parallel_counter<=serial_to_parallel_counter+1;
141                  rx_valid<=0;
142              end
143              else begin
144                  rx_valid<=1;//Conversion is done
145              end
146          end
147      end
148  end
149 endcase
150 end
151 end
152 endmodule

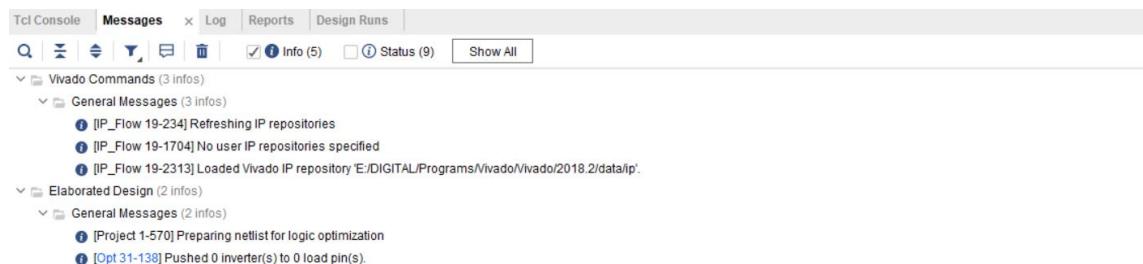
```

Elaboration:

Schematic:



Messages Tab:



Synthesis:

Encoding:

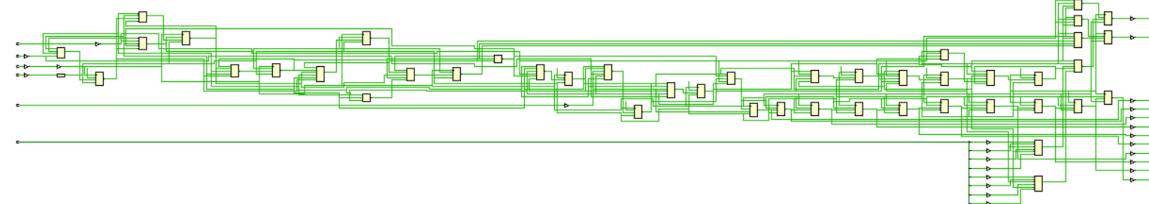
State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	011	010
READ_ADD	010	011
READ_DATA	111	100

Messages Tab:

The screenshot shows the Vivado IDE's Messages tab. The log area contains the following messages:

- > Vivado Commands (3 infos)
- > Synthesis (1 warning, 25 infos)
- > Synthesized Design (6 infos)
 - General Messages (6 infos)
 - [Netlist 29-17] Analyzing 13 Unisim elements for replacement
 - [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
 - [Project 1-479] Netlist was created with Vivado 2018.2
 - [Project 1-570] Preparing netlist for logic optimization
 - [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
 - [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

Schematic:



Timing Report Summary:

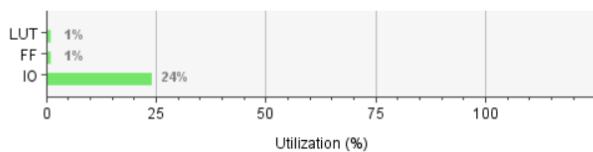
Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.975 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: 38	Total Number of Endpoints: 38	Total Number of Endpoints: 24

All user specified timing constraints are met.

Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	22	20800	0.11
FF	23	41600	0.06
IO	25	106	23.58



Testbench Code Snippets:

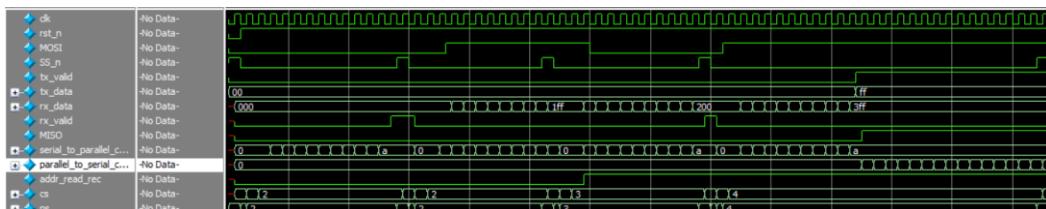
```

1  module SPI_Slave_tb();
2  //Signals Declaration
3  reg MOSI,SS_n,clk,rst_n,tx_valid;
4  reg [7:0] tx_data;
5  wire [9:0] rx_data;
6  wire rx_valid,MISO;
7  //Clock Generation
8  initial begin
9    clk=0;
10   forever
11   #5 clk=~clk;//Period=10ns
12 end
13 //DUT Instantiation
14 SPI_Slave DUT (.MOSI(MOSI),.MISO(MISO),.SS_n(SS_n),.clk(clk),.rst_n(rst_n),.rx_data(rx_data),.rx_valid(rx_valid),.tx_data(tx_data),.tx_valid(tx_valid));
15 //Test Stimulus Generator
16 initial begin
17   //Activate Reset ,Un-Enable SS_n & Initialize all signals
18   rst_n=0;
19   MOSI=0;
20   SS_n=1;
21   tx_data=0;
22   tx_valid=0;
23   @(negedge clk);
24   //De-activate Reset ,Enable SS_n & Start testing
25   rst_n=1;
26   SS_n=0;
27   @(negedge clk);
28   //Test Writing Address
29   SS_n=0;
30   @(negedge clk);
31   MOSI=0;/Write
32   @(negedge clk);
33   repeat(2) begin//2'b00 >> To write address
34     MOSI=0;
35     @(negedge clk);
36   end
37   repeat(8) begin//8'b00000000 >> Address
38     MOSI=0;
39     @(negedge clk);
40   end
41   SS_n=1;
42   @(negedge clk);

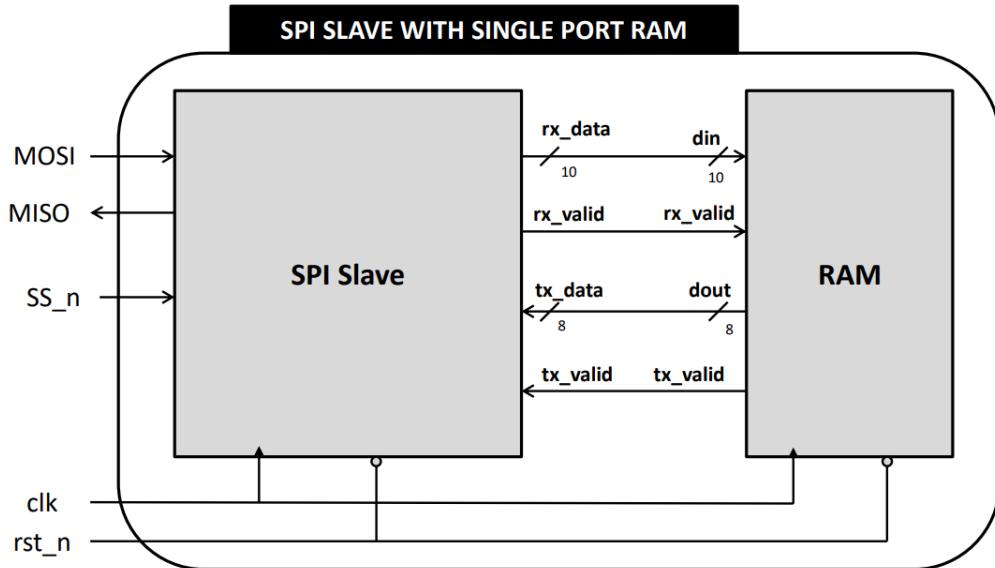
43   //Test Writing Data
44   SS_n=0;
45   @(negedge clk);
46   MOST=0;/Write
47   @(negedge clk);
48   MOST=0;
49   @(negedge clk);
50   MOST=1;/2'b01 >> To write Data
51   repeat(8) begin//8'b11111111 >> Data
52     MOST=1;
53     @(negedge clk);
54   end
55   SS_n=1;
56   @(negedge clk);
57   //Test Reading Address
58   SS_n=0;
59   @(negedge clk);
60   MOST=1;/Read
61   @(negedge clk);
62   MOST=1;
63   @(negedge clk);
64   MOST=0;/2'b10 >> To Read address
65   @(negedge clk);
66   repeat(8) begin//8'b00000000 >> Address
67     MOST=0;
68     @(negedge clk);
69   end
70   SS_n=1;
71   @(negedge clk);
72   //Test Reading Data
73   SS_n=0;
74   @(negedge clk);
75   MOST=1;/Read
76   @(negedge clk);
77   repeat(2) begin//2'b11 >> To Read Data
78     MOST=1;
79     @(negedge clk);
80   end
81   repeat(8) begin//8'b00000000 >> Dummy Data
82     MOST=1;
83     @(negedge clk);
84   end
85   tx_valid=1;
86   tx_data=8'b11111111;//Data to be transmitted to MISO
87   repeat(15) @(negedge clk); //Wait To Read Data
88   SS_n=1;
89   @(negedge clk);
90 $stop;
91 end
92 //Test Monitor & Results
93 initial begin
94   $monitor("MOSI = %b, MISO = %b, SS_n = %b, rx_data = %b, rx_valid = %b",MOSI, MISO, SS_n, rx_data, rx_valid);
95 end
96 endmodule

```

Simulation Waveform Snippet:



SPI Wrapper



Introduction to Wrapper:

The Wrapper module integrates the Slave and RAM modules to enable SPI communication with a memory block. The Slave manages the SPI protocol and data transfer with the master device, while the RAM module handles data storage and retrieval. This setup allows the master device to read from and write to the memory block via the SPI interface, with the Slave ensuring proper data transfer and the RAM maintaining the data integrity.

Main Wires between RAM & Slave

1. rx_data in the SPI slave module is connected to the din port in the RAM module.
2. rx_valid in the SPI slave module is connected to rx_valid in the RAM module.
3. dout in the RAM module is connected to tx_data in the SPI slave module.
4. tx_valid in the RAM module is connected to tx_valid in the SPI slave module.

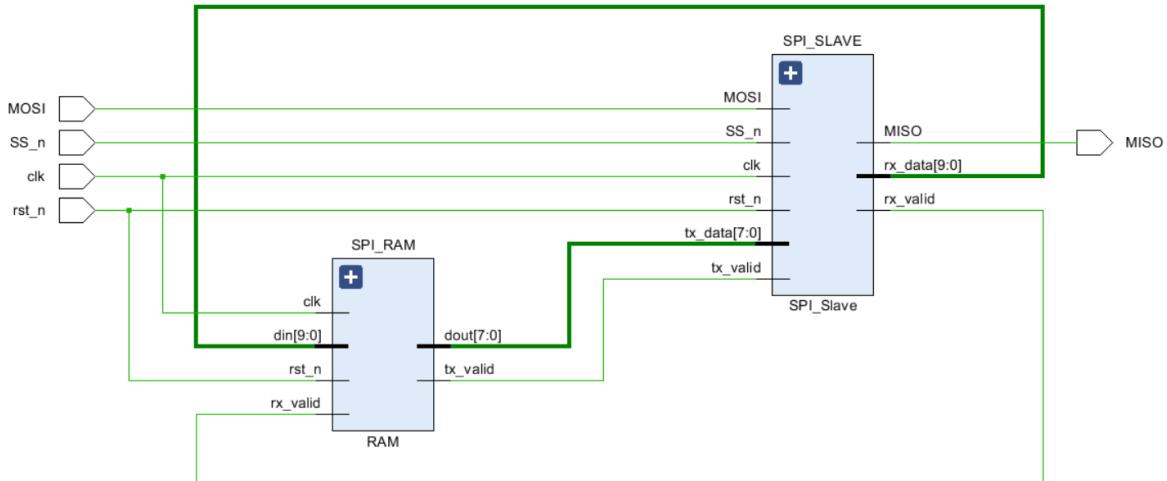
RTL Code Snippet:

```
1  module SPI_Wrapper(MOSI, MISO, SS_n, clk, rst_n);
2  // Ports
3  input MOSI, clk, rst_n, SS_n;
4  output MISO;
5
6  // Internal Signals
7  wire [9:0] rx_data;
8  wire rx_valid;
9  wire [7:0] tx_data;
10 wire tx_valid;
11
12 // Instantiations
13 SPI_Slave SPI_SLAVE (.MOSI(MOSI), .MISO(MISO), .SS_n(SS_n), .clk(clk), .rst_n(rst_n), .rx_data(rx_data), .rx_valid(rx_valid), .tx_valid(tx_valid), .tx_data(tx_data));
14 RAM #(ADDR_SIZE(8), .MEM_DEPTH(256)) SPI_RAM (.din(rx_data), .rx_valid(rx_valid), .clk(clk), .rst_n(rst_n), .dout(tx_data), .tx_valid(tx_valid));
15 endmodule
```

Gray Encoding

Elaboration:

Schematic:



Messages Tab:

Tcl Console Messages Log Reports Design Rules

Vivado Commands (3 infos)

- IP_Flow 19-234 Refreshing IP repositories
- IP_Flow 19-1704 No user IP repositories specified
- IP_Flow 19-2313 Loaded Vivado IP repository E:/DIGITAL/Programs/Vivado/Vivado/2018.2/data/ip

General Messages (111 infos)

- Synth 8-157 Synthesizing module SPI_Wrapper [SPI_Wrapper v 1] (2 more like this)
- Synth 8-5534 Detected attribute (* fsm_encoding = "gray") [SPI_Slave v 1]
- Synth 8-155 case statement is not full and has no default [SPI_Slave v 9]
- Synth 8-155 done synthesizing module SPI_Slave (1#1) [SPI_Slave v 1] (2 more like this)
- Project 1-570 Preparing netlist for logic optimization
- [Op 31-138] Pushed 0 inverter(s) to 0 load pin(s).
- Project 1-111 Unisim Transformation Summary:
No Unisim elements were transformed.

Synthesis:

Encoding:

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

Messages Tab:

Tcl Console Messages Log Reports Design Rules Timing

Vivado Commands (3 infos)

- IP_Flow 19-234 Refreshing IP repositories
- IP_Flow 19-1704 No user IP repositories specified
- IP_Flow 19-2313 Loaded Vivado IP repository E:/DIGITAL/Programs/Vivado/Vivado/2018.2/data/ip

General Messages (1 info)

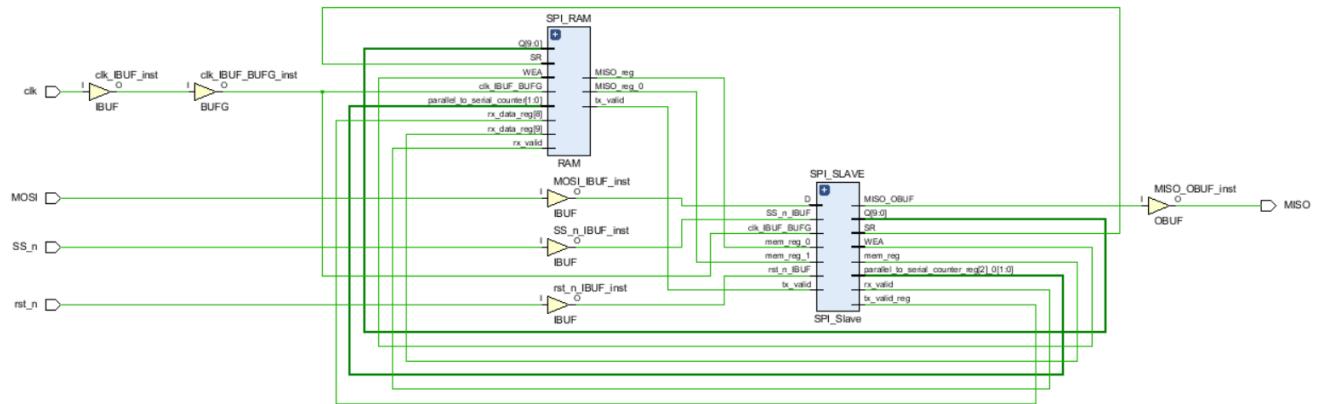
- Synth 29-17 Analyzing 5 Unisim elements for replacement

Synthesis (1 warning, 21 infos)

Synthesized Design (6 infos)

- Netlist 29-17 Netlist was created with Vivado 2018.2
- Project 1-479 Netlist was created with Vivado 2018.2
- Project 1-570 Preparing netlist for logic optimization
- [Op 31-138] Pushed 0 inverter(s) to 0 load pin(s).
- Project 1-111 Unisim Transformation Summary:
No Unisim elements were transformed.

Schematic:



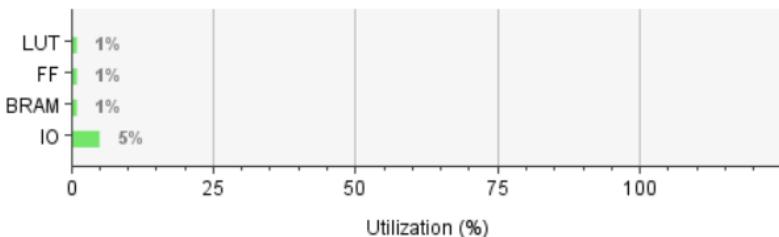
Timing Report Summary:

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: 98	Total Number of Endpoints: 98	Total Number of Endpoints: 43

All user specified timing constraints are met.

Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	26	20800	0.13
FF	40	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



Implementation:

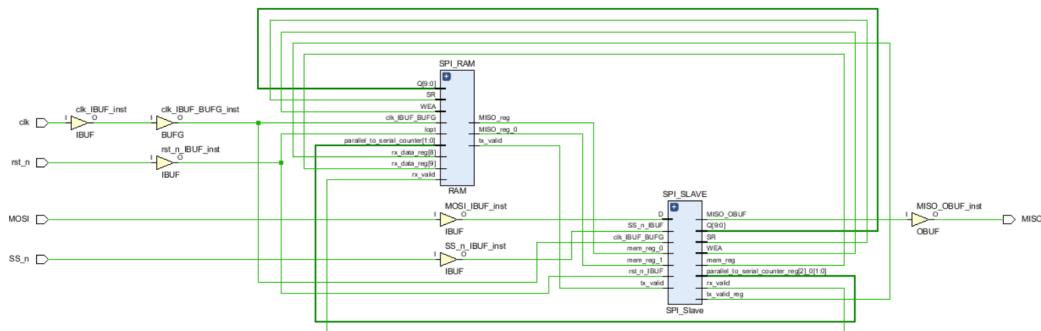
Messages Tab:

Tcl Console Messages Log Reports Design Runs Power Methodology Timing ?

Q | X | S | T | B | I | W Warning (1) Info (235) Status (477) Show All

- Implementation (9 infos)
 - > Design Initialization (11 infos)
 - > Opt Design (29 infos)
 - > Place Design (21 infos)
 - > Route Design (35 infos)
- Implemented Design (9 infos)
 - > General Messages (9 infos)
 - Info [Netlist 29-17] Analyzing 5 Unisim elements for replacement
 - Info [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
 - Info [Project 1-478] Netlist was created with Vivado 2018.2
 - Info [Project 1-570] Preparing netlist for logic optimization
 - Timing [Timing 38-478] Restoring timing data from binary archive.
 - Timing [Timing 38-478] Binary timing data restore complete.

Schematic:



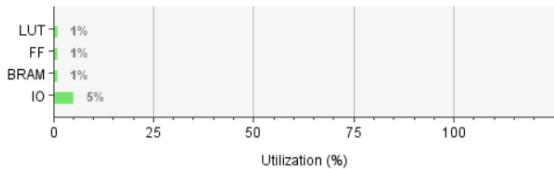
Timing Report Summary:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.375 ns	Worst Hold Slack (WHS): 0.101 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: 98	Total Number of Endpoints: 98	Total Number of Endpoints: 43

All user specified timing constraints are met.

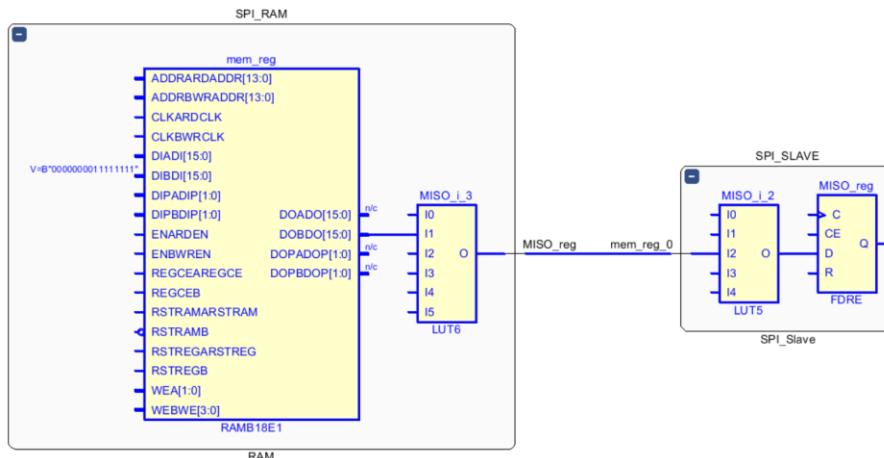
Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	26	20800	0.13
FF	40	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72

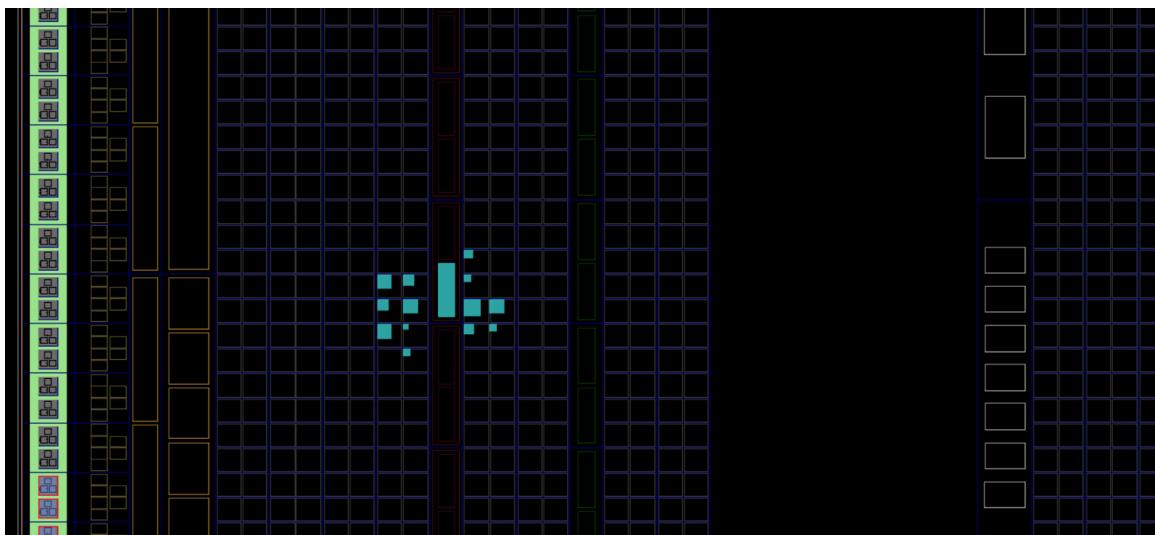
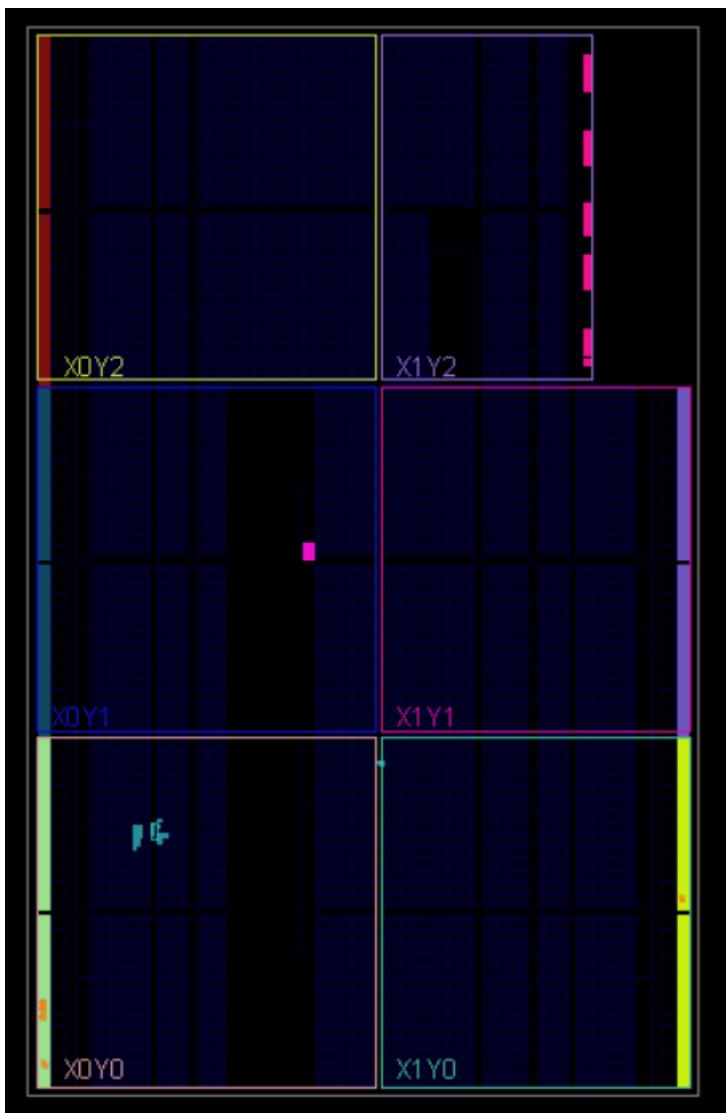


Critical Path:

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	5.375	2	1	SPI_RAM/mem.../CLKBWRCLK	SPI_SLAVE/MISO_reg/D	4.652	2.702	1.950



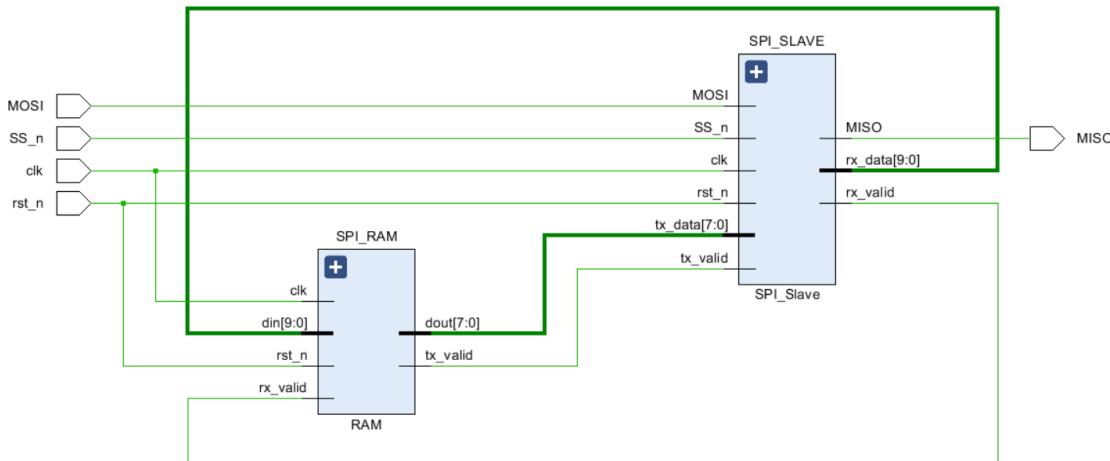
Device:



Sequential Encoding

Elaboration:

Schematic:



Messages Tab:

Tcl Console Messages Log Reports Design Runs ? -

Q | Info (14) Status (11) Show All

- Vivado Commands (3 info)
 - IP_Flow 19-234 Refreshing IP repositories
 - IP_Flow 19-1704 No user IP repositories specified
 - IP_Flow 19-2313 Loaded Vivado IP repository E:/DIGITAL/Programs/Vivado/Vivado2018.2/dataip/
- Elaborated Design (11 info)
 - ynth 8-157 synthesizing module SPI_Wrapper [SPI_Wrapper v 1] (2 more like this)
 - ynth 8-6534 Detected attribute (* fsm_encoding = "gray") [SPI_Slave v 1]
 - ynth 8-155 case statement is not full and has no default [SPI_Slave v 9]
 - ynth 8-155 case statement is not full and has no default [SPI_Slave v 1] (2 more like this)
 - Project 1-570 Preparing retiming for logic optimization
 - Dat 31-130 Pushed 9 inverters to 9 load pins.
 - Project 1-111 Usain Transformation Summary: No Usain elements were transformed.

Synthesis:

Encoding:

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

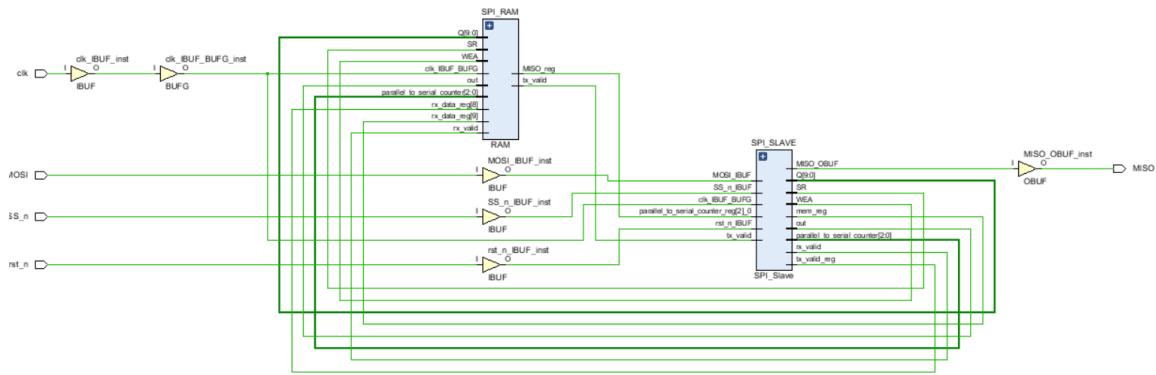
Messages Tab:

Tcl Console Messages Log Reports Design Runs Timing ? -

Q | Warning (1) Info (40) Status (21) Show All

- Synthesis (1 warning, 31 infos)
 - [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
 - [Synth 8-157] synthesizing module SPI_Wrapper [SPI_Wrapper v 1] (2 more like this)
 - [Synth 8-6534] Detected attribute (* fsm_encoding = "sequential") [SPI_Slave v 13]
 - [Synth 8-155] case statement is not full and has no default [SPI_Slave v 9]
 - [Synth 8-155] case statement is not full and has no default [SPI_Slave v 1] (2 more like this)
 - [Synth 8-155] done synthesizing module SPI_Slave (1#1) [SPI_Slave v 1] (2 more like this)
 - [Device 21-403] Loading part xc7a35tcpg236-1L
 - [Project 1-238] Implementation specific constraints were found while reading constraint file [E:/DIGITAL/Kareem Waseem Diploma/Design/Project/Constraints_basys3.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [Xil/SPI_Wrapper_propImpl.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

Schematic:



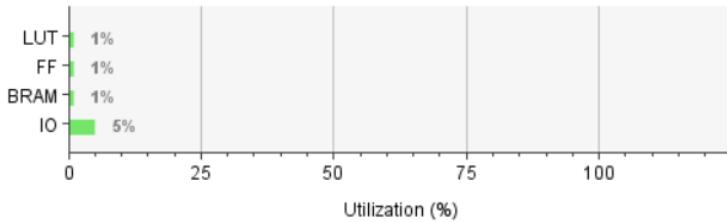
Timing Report Summary:

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: 98	Total Number of Endpoints: 98	Total Number of Endpoints: 43

All user specified timing constraints are met.

Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	27	20800	0.13
FF	40	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72



Implementation:

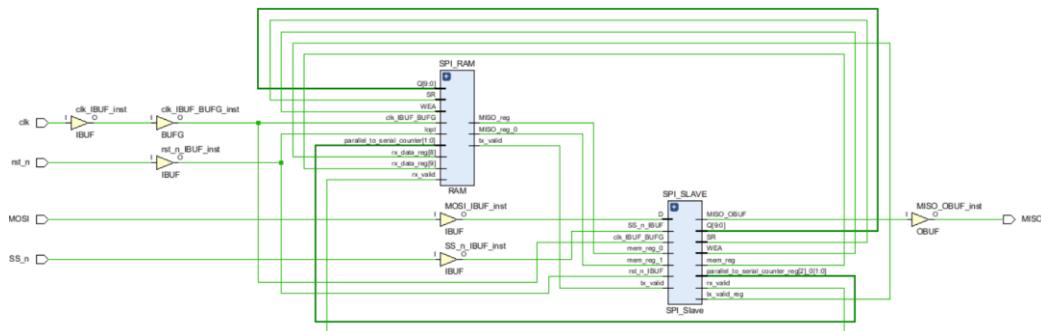
Messages Tab:

Tcl Console Messages Log Reports Design Runs Power Methodology Timing ?

Q | Z | ⌂ | T | E | B | ⓘ | ⓘ Warning (1) ⓘ Info (235) ⓘ Status (477) Show All

- Implementation (9 infos)
 - > Design Initialization (11 infos)
 - > Opt Design (29 infos)
 - > Place Design (21 infos)
 - > Route Design (35 infos)
- Implemented Design (9 infos)
 - > General Messages (9 infos)
 - ⓘ [Netlist 29-17] Analyzing 5 Unisim elements for replacement
 - ⓘ [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
 - ⓘ [Project 1-478] Netlist was created with Vivado 2018.2
 - ⓘ [Project 1-570] Preparing netlist for logic optimization
 - ⓘ [Timing 38-478] Restoring timing data from binary archive.
 - ⓘ [Timing 38-479] Binary timing data restore complete.

Schematic:



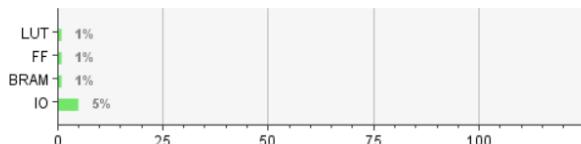
Timing Report Summary:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.639 ns	Worst Hold Slack (WHS): 0.068 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: 98	Total Number of Endpoints: 98	Total Number of Endpoints: 43

All user specified timing constraints are met.

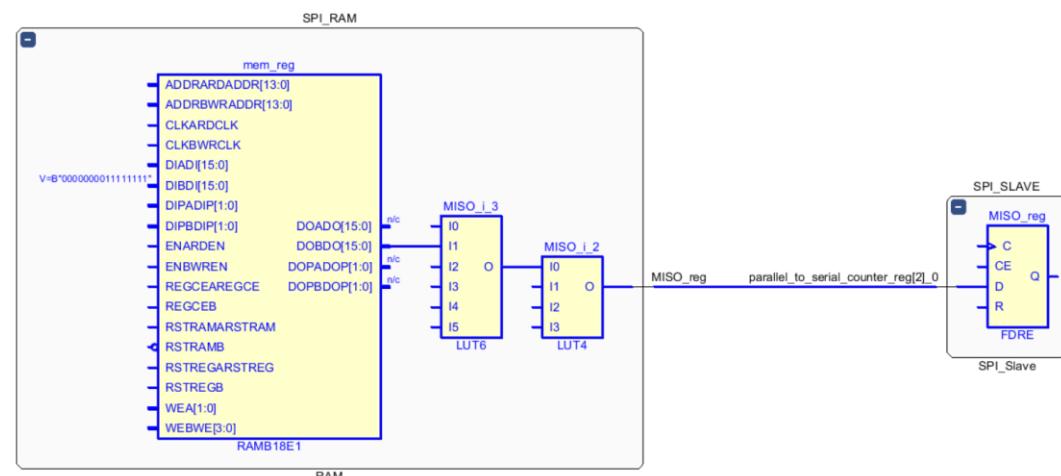
Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	27	20800	0.13
FF	40	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72

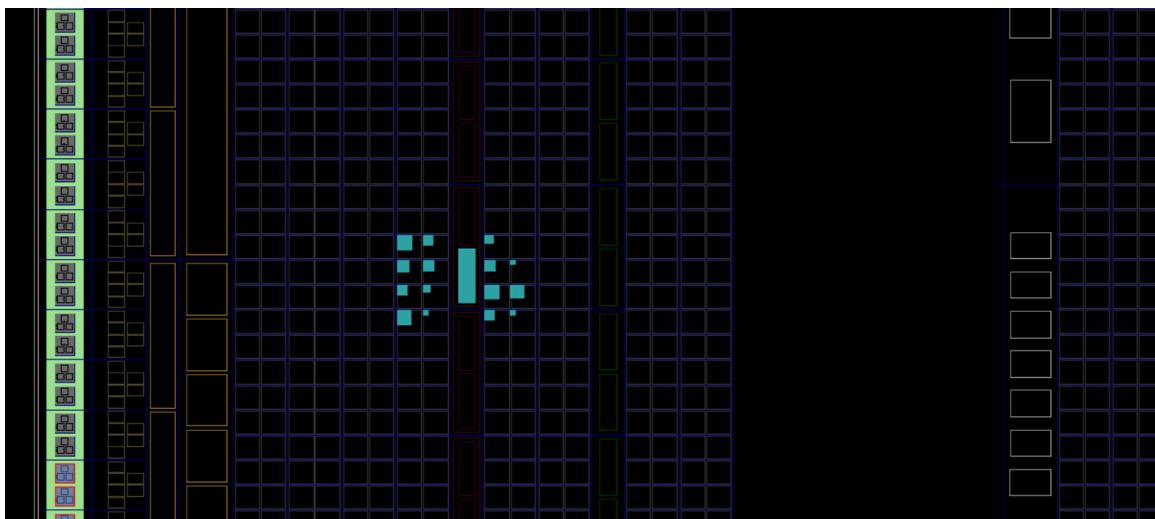
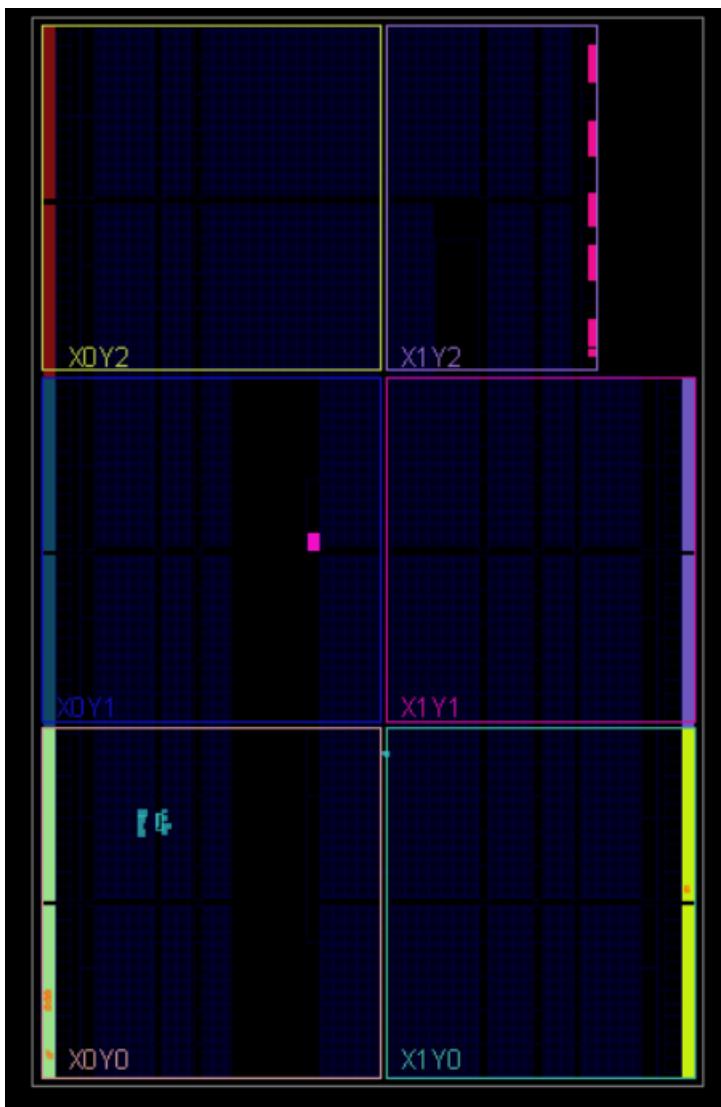


Critical Path:

Name	Slack	^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	5.639		2	1	SPI_RAM/mem.../CLKBWRCLK	SPI_SLAVE/MISO_reg/D	4.388	2.702	1.686



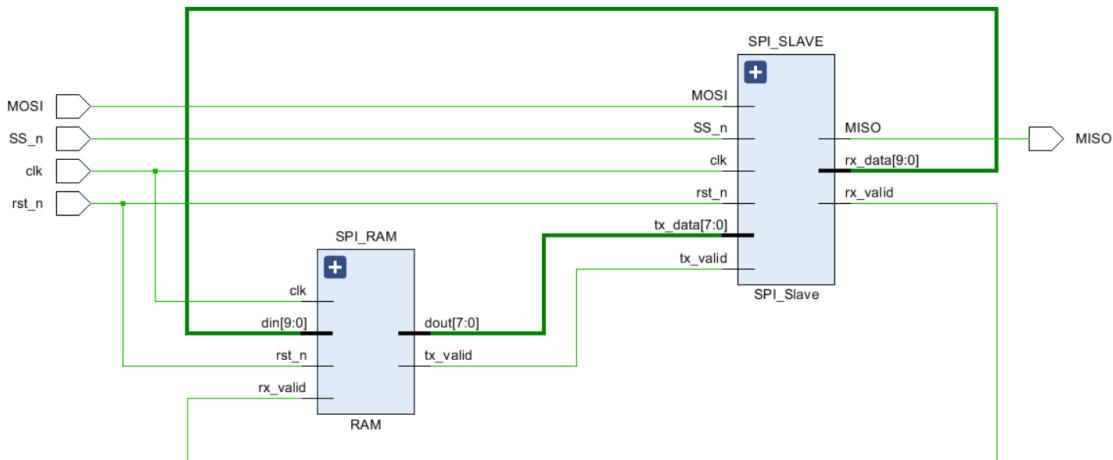
Device:



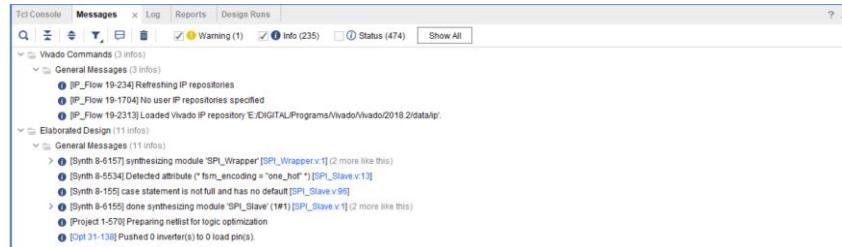
One Hot Encoding

Elaboration:

Schematic:



Messages Tab:

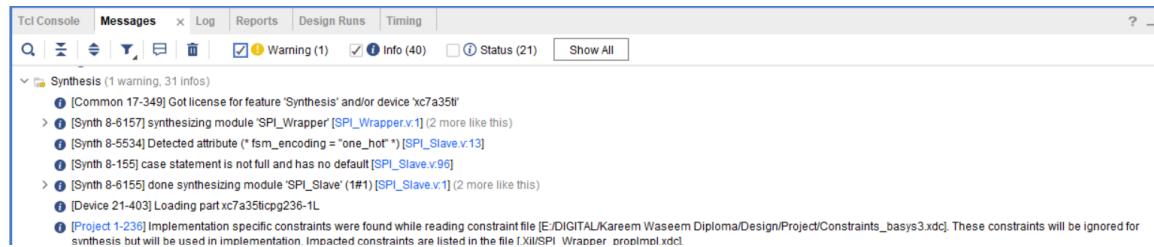


Synthesis:

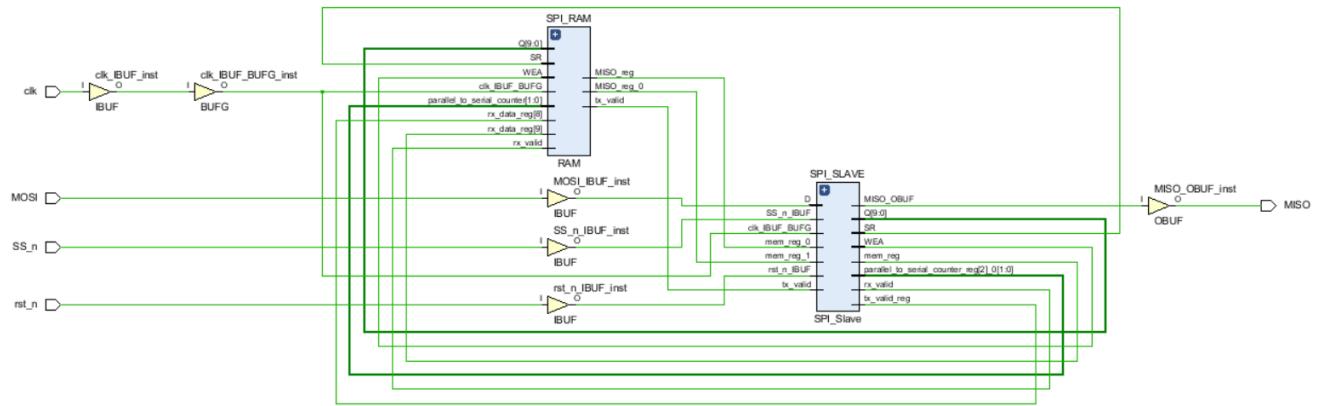
Encoding:

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

Messages Tab:



Schematic:



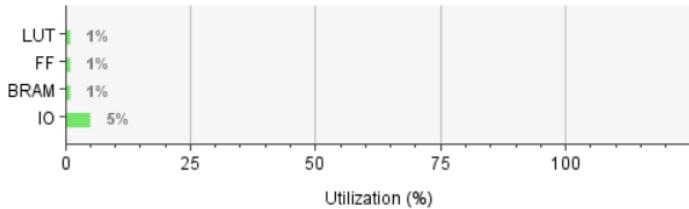
Timing Report Summary:

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:	100	Total Number of Endpoints:	100	Total Number of Endpoints:	45

All user specified timing constraints are met.

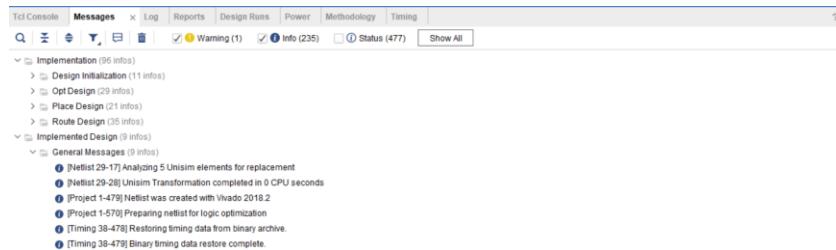
Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	29	20800	0.14
FF	42	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72

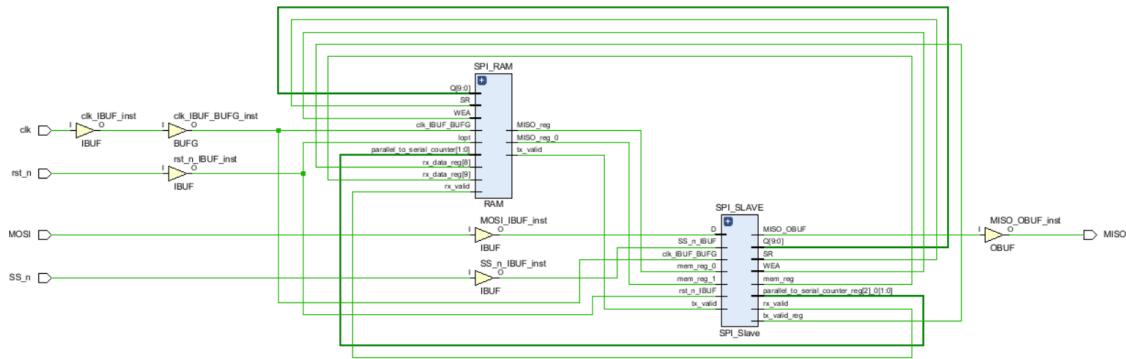


Implementation:

Messages Tab:



Schematic:



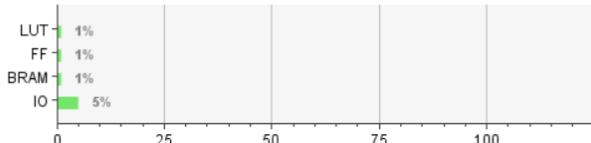
Timing Report Summary:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.806 ns	Worst Hold Slack (WHS): 0.075 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: 100	Total Number of Endpoints: 100	Total Number of Endpoints: 45

All user specified timing constraints are met.

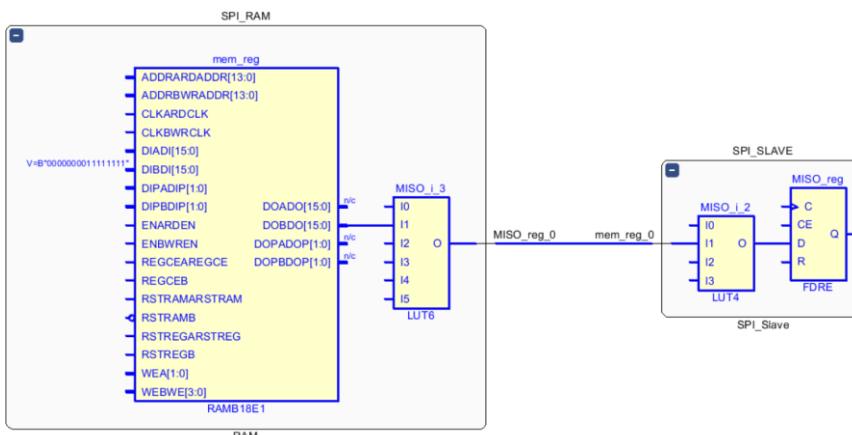
Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	29	20800	0.14
FF	42	41600	0.10
BRAM	0.50	50	1.00
IO	5	106	4.72

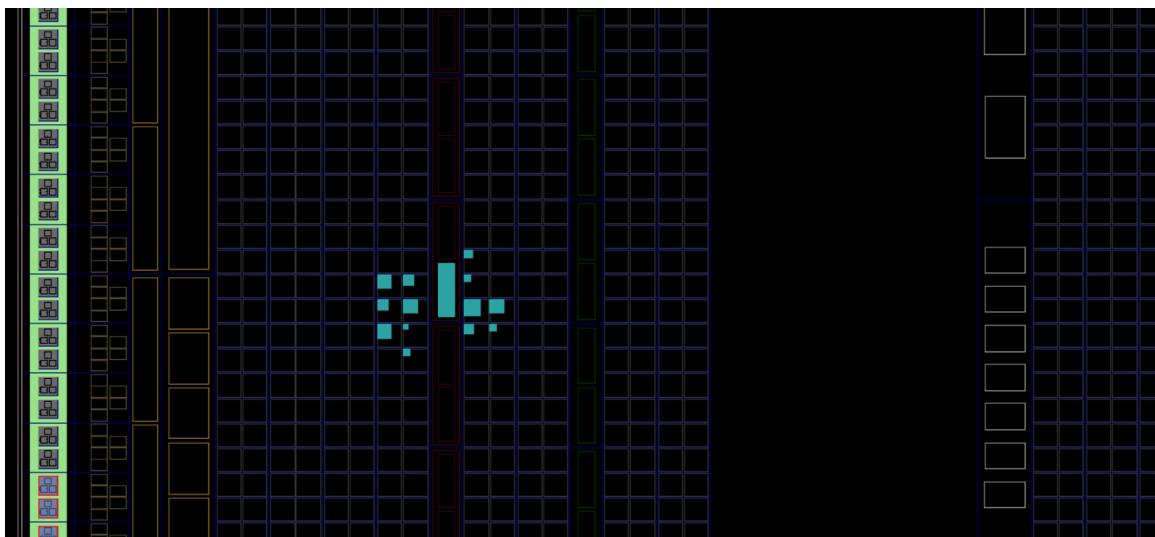
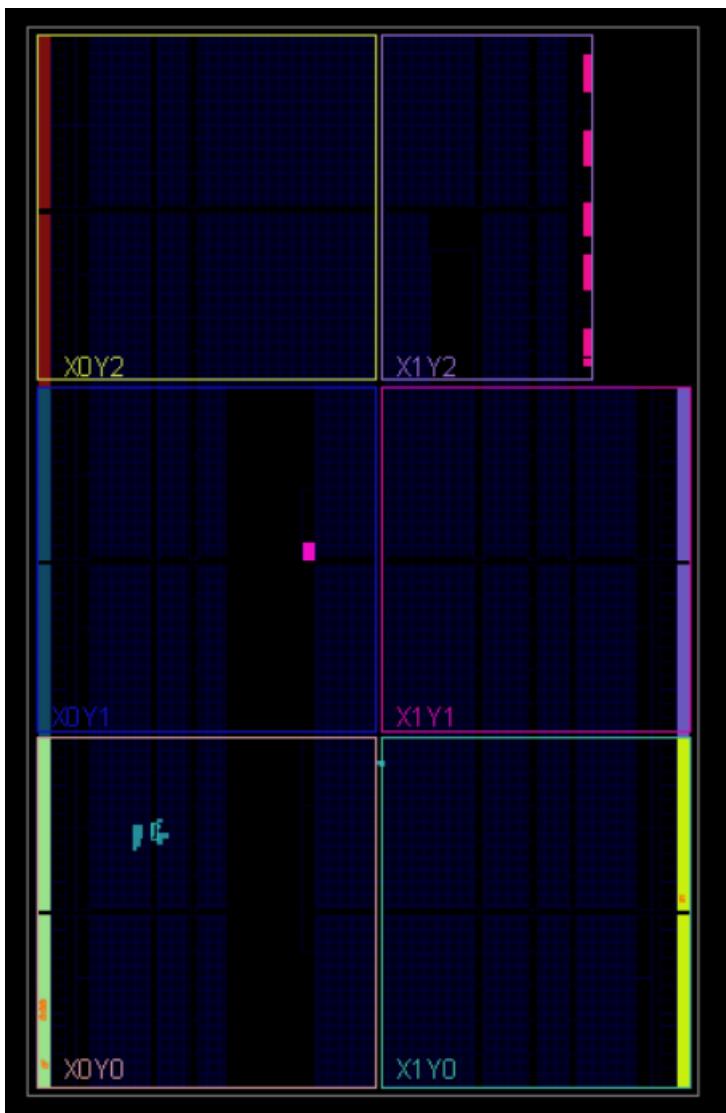


Critical Path:

Name	Slack	^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	5.806		2	1	SPI_RAM/mem.../CLKBWRCLK	SPI_SLAVE/MISO_reg/D	4.168	2.702	1.466



Device:



The Best Encoding

based on the best timing report that gives the high setup/hold slack after implementation.

Gray Encoding:

- **Worst Negative Setup Slack (WNSS):** 5.376 ns
- **Worst Hold Slack (WHS):** 0.101 ns
- **Worst Pulse Width Slack (WPWS):** 4.500 ns

Sequential Encoding:

- **Worst Negative Setup Slack (WNSS):** 5.693 ns
- **Worst Hold Slack (WHS):** 0.068 ns
- **Worst Pulse Width Slack (WPWS):** 4.500 ns

One-Hot Encoding:

- **Worst Negative Setup Slack (WNSS):** 5.806 ns
- **Worst Hold Slack (WHS):** 0.144 ns
- **Worst Pulse Width Slack (WPWS):** 4.500 ns

Analysis

- **Setup Slack:** One-Hot encoding has the highest worst negative setup slack (5.806 ns), indicating it has the most timing margin for setup constraints.
- **Hold Slack:** One-Hot encoding has the best worst hold slack (0.144 ns), providing the highest timing margin for hold constraints.
- **Pulse Width Slack:** All three encodings have the same worst pulse width slack (4.500 ns).

Comparison:

- **Setup Slack (Higher is better):**
 - Gray: 5.376 ns
 - Sequential: 5.693 ns
 - One-Hot: 5.806 ns
- **Hold Slack (Higher is better):**
 - Gray: 0.101 ns
 - Sequential: 0.068 ns
 - One-Hot: 0.144 ns

Conclusion

Given that One-Hot encoding has the highest setup and hold slack values, it provides the best timing margin overall. Therefore, to operate at the highest frequency possible, you should choose **One-Hot encoding**.

Constraints File:

```

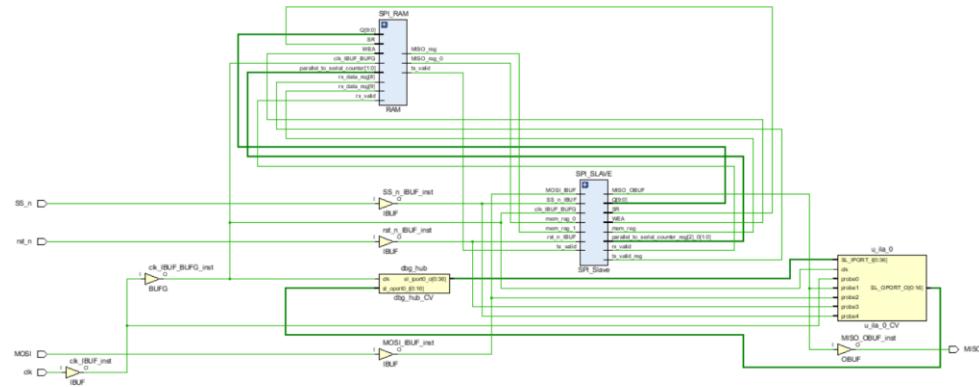
1  ## Clock signal
2  set_property -dict { PACKAGE_PIN W5    IOSTANDARD LVCMS33 } [get_ports clk]
3  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
4
5  ## Switches
6  set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVCMS33 } [get_ports {rst_n}]
7  set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMS33 } [get_ports {SS_n}]
8  set_property -dict { PACKAGE_PIN W16    IOSTANDARD LVCMS33 } [get_ports {MOSI}]
9
10 ## LEDs
11 set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMS33 } [get_ports {MISO}]
12
13 ## Configuration options, can be used for all designs
14 set_property CONFIG_VOLTAGE 3.3 [current_design]
15 set_property CFGBVS VCCO [current_design]
16
17 ## SPI configuration mode options for QSPI boot, can be used for all designs
18 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
19 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
20 set_property CONFIG_MODE SPIx4 [current_design]

```

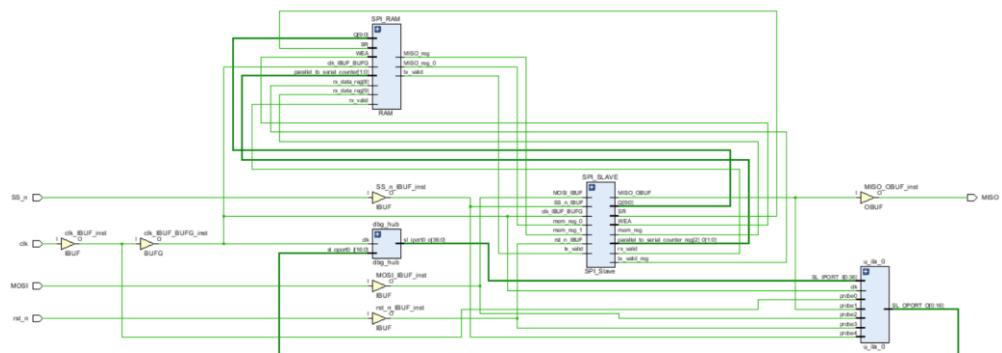
Debug Core:

Debug Core is added to One Hot Encoded Design

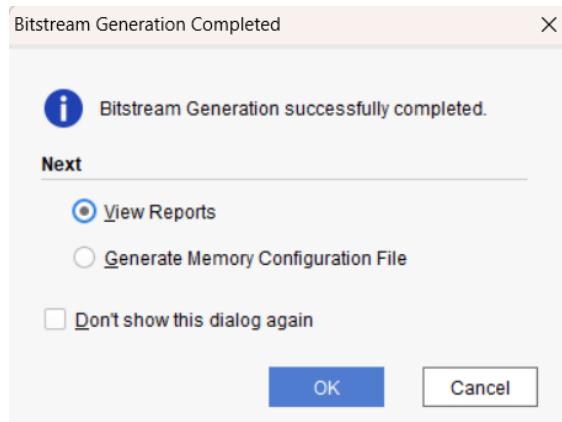
Synthesis Schematic:



Implementation Schematic:



Bitstream Generation Success:



```

1 0009 0ff0 0ff0 0ff0 0000 0161 003b
2 5350 495f 5772 6170 7065 723b 434f 4d50
3 5245 5353 3d54 5255 453b 5573 6572 4944
4 3d30 5846 4646 4646 4646 463b 5665 7273
5 696f 6e3d 3230 3138 2e32 0062 000d 3761
6 3335 7469 6370 6732 3336 0063 000b 3230
7 3234 2f30 382f 3034 0064 0009 3137 3a33
8 343a 3437 0065 0007 93c4 ffff ffff
9 ffff ffff ffff ffff ffff ffff ffff ffff
10 ffff ffff ffff ffff ffff 0000 00bb 1122
11 0044 ffff ffff ffff ffff aa99 5566 2000
12 0000 3003 e001 0000 026b 3000 8001 0000
13 0012 2000 0000 3002 2001 0000 0000 3002
14 0001 0000 0000 3000 8001 0000 0000 2000
15 0000 3000 8001 0000 0007 2000 0000 2000

```

Constraints file after adding debug core:

```

1 ## Clock signal
2 set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMS33} [get_ports clk]
3 create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
4
5 ## Switches
6 set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMS33} [get_ports rst_n]
7 set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMS33} [get_ports SS_n]
8 set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMS33} [get_ports MOSI]
9 ## LEDs
10 set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMS33} [get_ports MISO]
11
12 ## Configuration options, can be used for all designs
13 set_property CONFIG_VOLTAGE 3.3 [current_design]
14 set_property CFGBVS VCCO [current_design]
15
16 ## SPI configuration mode options for QSPI boot, can be used for all designs
17 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
18 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
19 set_property CONFIG_MODE SPIx4 [current_design]
20
21 create_debug_core u_ila_0 ila
22 set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
23 set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
24 set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
25 set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
26 set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
27 set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
28 set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
29 set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
30 set_property port_width 1 [get_debug_ports u_ila_0/clk]
31 connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
32 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
33 set_property port_width 1 [get_debug_ports u_ila_0/probe0]
34 connect_debug_port u_ila_0/probe0 [get_nets [list clk_IBUF]]
35 create_debug_port u_ila_0 probe
36 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
37 set_property port_width 1 [get_debug_ports u_ila_0/probe1]
38 connect_debug_port u_ila_0/probe1 [get_nets [list MISO_OBUF]]
39 create_debug_port u_ila_0 probe
40 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
41 set_property port_width 1 [get_debug_ports u_ila_0/probe2]
42 connect_debug_port u_ila_0/probe2 [get_nets [list MOSI_IBUF]]
43 create_debug_port u_ila_0 probe
44 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
45 set_property port_width 1 [get_debug_ports u_ila_0/probe3]
46 connect_debug_port u_ila_0/probe3 [get_nets [list rst_n_IBUF]]
47 create_debug_port u_ila_0 probe
48 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
49 set_property port_width 1 [get_debug_ports u_ila_0/probe4]
50 connect_debug_port u_ila_0/probe4 [get_nets [list SS_n_IBUF]]
51 set_property C_CLK_INPUT_FREQ_HZ 30000000 [get_debug_cores dbg_hub]
52 set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
53 set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
54 connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
55

```

Master (SPI SLAVE Interface TB) RTL Code Snippet:

```
1  module MASTER();
2  //Signals Declaration
3  reg MOSI;
4  reg SS_n;
5  reg clk;
6  reg rst_n;
7  wire MISO;
8  //DUT Instantiation
9  SPI_Wrapper DUT(.MOSI(MOSI),.MISO(MISO),.SS_n(SS_n),.clk(clk),.rst_n(rst_n));
10 //Clock Generation
11 initial begin
12   clk = 0;
13   forever
14     #5 clk = ~clk;
15 end
16 //Test Stimulus Generator
17 initial begin
18   //Activate Reset ,Un-Enable SS_n& Initialize all signals
19   rst_n = 0;
20   SS_n = 1;
21   MOSI = 0;
22   @(negedge clk);
23   //De-Activate Reset& Start testing
24   rst_n = 1;
25
26   //Test Writing Address
27   SS_n = 0;
28   @(negedge clk);
29
30   MOSI = 0;//Write
31   @(negedge clk);
32
33   MOSI = 0;
34   @(negedge clk);
35   MOSI = 0;
36   @(negedge clk);
37   //2'b00 >> To write address
38   repeat(3) begin
39     MOSI = 0;
40     @(negedge clk);
41     MOSI = 1;
42     @(negedge clk);
43     MOSI = 1;
44     @(negedge clk);
45     MOSI = 1;
46     @(negedge clk);
47   end
48   //Address : 8'b01110111
49   SS_n = 1;
50   @(negedge clk);
51
```

```
52   //Test Writing Data
53   SS_n = 0;
54   @(negedge clk);
55   MOSI = 0;//Wtite
56   @(negedge clk);
57
58   MOSI = 0;
59   @(negedge clk);
60   MOSI = 1;
61   @(negedge clk);
62   //2'b01 >> To write Data
63   MOSI = 1;
64   @(negedge clk);
65   MOSI = 0;
66   @(negedge clk);
67   MOSI = 1;
68   @(negedge clk);
69   MOSI = 0;
70   @(negedge clk);
71   MOSI = 1;
72   @(negedge clk);
73   MOSI = 0;
74   @(negedge clk);
75   MOSI = 1;
76   @(negedge clk);
77   MOSI = 0;
78   @(negedge clk);
79   //Data:8'b10101010
80   SS_n = 1;
81   @(negedge clk);
82
83   //Test Reading Address
84   SS_n = 0;
85   @(negedge clk);
86
87   MOSI = 1;//Read
88   @(negedge clk);
89
90   MOSI = 1;
91   @(negedge clk);
92   MOSI = 0;
93   @(negedge clk);
```

```

93      @(negedge clk);
94      //2'b10 >> To Read address
95      repeat(3) begin
96          MOSI = 0;
97          @(negedge clk);
98          MOSI = 1;
99          @(negedge clk);
100         MOSI = 1;
101         @(negedge clk);
102         MOSI = 1;
103         @(negedge clk);
104     end
105     //Address : 8'b01110111
106     SS_n = 1;
107     @(negedge clk); | //Test Reading Data
108     SS_n = 0;
109     @(negedge clk);
110
111     MOSI = 1;//Read
112     @(negedge clk);
113
114     MOSI = 1;
115     @(negedge clk);
116     MOSI = 1;
117     @(negedge clk);
118     //2'b11 >> To Read Data
119     MOSI = 0;
120     @(negedge clk);
121     MOSI = 1;
122     @(negedge clk);
123     MOSI = 1;
124     @(negedge clk);
125     MOSI = 0;
126     @(negedge clk);
127     MOSI = 1;
128     @(negedge clk);
129     MOSI = 0;
130     @(negedge clk);
131     MOSI = 0;
132     @(negedge clk);
133     MOSI = 0;
134     @(negedge clk);
135     @(negedge clk);
136     //Dummy Data
137     repeat(8) @(negedge clk);
138     SS_n = 1;
139     $stop;
140 end
141 //Test Monitor & Results
142 initial begin
143     $monitor("MOSI = %b, MISO = %b, SS_n = %b",MOSI, MISO, SS_n);
144 end
145 endmodule

```

Simulation Waveform Snippet:



Do File:

```

vlib work
vlog RAM.v SPI_Slave.v SPI_Wrapper.v MASTER.v
vsim -voptargs=+acc work.MASTER
add wave *
run -all
#quit -sim

```

تم بحمد الله

سورة البقرة

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

وَأَن لَّيْسَ لِلْإِنْسَنِ إِلَّا مَا سَعَى

(٢٩)

سورة التوبة

وَقُلْ أَعْمَلُوا فَسِيرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ

وَسَرُّدُونَ إِلَى عَلِيهِ الْغَيْبِ وَالشَّهَدَةِ فَيُنَبِّئُكُمْ بِمَا كُنْتُمْ

تَعْمَلُونَ

(١٥)