



---

## SPI Slave with Single Port RAM

---

RTL and FPGA design



Ahmed Mohamed Abdellatif

# RTL design

- Spi slave

```
1 module SPI_Slave (mosi, ss_n, clk, rst_n, tx_valid, tx_data, miso, rx_valid, rx_data);
2
3 //State parameters
4 parameter IDLE=0;
5 parameter CHK_CMD=1;
6 parameter WRITE=2;
7 parameter READ_ADD=3;
8 parameter READ_DATA=4;
9
10 //Input ports
11 input mosi;                                //Master_out_slave_in input
12 input ss_n, clk, rst_n, tx_valid;           //Control input signals
13 input[7:0] tx_data;                         //The parallel data sent from the RAM to SPI
14
15 //Output ports
16 output reg miso;                           //Master_in_slave_out output
17 output reg rx_valid;                        //Control signal
18 output reg[9:0] rx_data;                   //The parallel data sent to RAM
19
20 //Internal signals
21 reg[3:0] counter,counter_rd_data;          //Counter for serial / parallel tranformations
22 reg[2:0] cs,ns;                            //State registers
23 reg end_counter_rd_data;                  //Internal control signals
24
25 //State memory
26 always @(posedge clk) begin
27     if(~rst_n)
28         cs<=IDLE;
29     else
30         cs<=ns;
31 end
32
33 //Next state combinational logic
34 always @(*) begin
35     case (cs)
36         IDLE: begin
37             if (ss_n == 0)
38                 ns = CHK_CMD;
39             else
40                 ns = IDLE;
41         end
42         CHK_CMD: begin
43             if ((ss_n == 0) && rx_valid && (rx_data[9] == 0))
44                 ns = WRITE;
45             else if ((ss_n == 0) && rx_valid && (rx_data[9:8] == 2'b10))
46                 ns = READ_ADD;
47             else if ((ss_n == 0) && rx_valid && (rx_data[9:8] == 2'b11))
48                 ns = READ_DATA;
49             else if(rx_valid == 0)
50                 ns = CHK_CMD;
51             else
52                 ns = IDLE;
53         end
54         WRITE: begin
55             if (ss_n == 1)
56                 ns = IDLE;
57             else
58                 ns = WRITE;
59         end
60         READ_ADD: begin
61             if (ss_n == 1)
62                 ns = IDLE;
63             else
64                 ns = READ_ADD;
65         end
66         READ_DATA: begin
67             if (ss_n == 1)
68                 ns = IDLE;
69             else
70                 ns = READ_DATA;
71         end
72     default: ns = IDLE;
73     endcase
74 end
```

```

75
76 //Serial mosi to parallel rx_data (from SPI to RAM)
77 always @(posedge clk) begin
78     if(~rst_n) begin
79         rx_data <= 0;
80         counter <= 0;
81         rx_valid <= 0;
82     end
83     else if (counter >= 10) begin
84         counter <= 0;
85         rx_valid <= 1;
86     end
87     else if((cs == CHK_CMD) && !rx_valid) begin
88         rx_data[9-counter] <= mosi;
89         counter <= counter + 1;
90     end
91     else if (cs == IDLE)
92         rx_valid <= 0;
93 end
94
95 //Parallel tx_data to serial miso (read data from RAM to SPI)
96 always @(posedge clk) begin
97     if(~rst_n) begin
98         miso <= 0;
99         counter_rd_data <= 0;
100        end_counter_rd_data <= 0;
101    end
102    else if (counter_rd_data >= 8) begin
103        counter_rd_data <= 0;
104        end_counter_rd_data <= 1;
105    end
106    else if ((cs == READ_DATA) && tx_valid && !end_counter_rd_data) begin
107        miso <= tx_data[7-counter_rd_data];
108        counter_rd_data <= counter_rd_data+1;
109    end
110    else if (cs == IDLE) begin
111        end_counter_rd_data <= 0;
112    end
113 end
114 endmodule

```

- Testbench for the SPI slave

```

1  module SPI_Slave_tb ();
2
3      reg mosi,ss_n,clk,rst_n,tx_valid;
4      reg[7:0] tx_data;
5      wire miso,rx_valid; // rx_valid show the counter ends
6      wire[9:0] rx_data;
7
8      SPI_Slave dut0 (mosi,ss_n,clk,rst_n,tx_valid,tx_data,miso,rx_valid,rx_data);
9
10     initial begin
11         clk=0;
12         forever
13             #2 clk=~clk;
14     end
15
16     initial begin
17         //1
18         rst_n=0; mosi=$random; ss_n=1; tx_valid=1; tx_data=$random;
19         @(negedge clk);
20         //2
21         rst_n=1; mosi=$random; ss_n=1; tx_valid=1; tx_data=$random;
22         @(negedge clk);
23         //3
24         ss_n=0; tx_data=$random; tx_valid=1;
25         repeat(30) begin
26             mosi=$random; ss_n=0;
27             @(negedge clk);
28         end
29         ss_n=1;
30         @(negedge clk);
31
32         //4
33         ss_n=0; tx_data=$random; tx_valid=1;
34         repeat(30) begin
35             mosi=$random; ss_n=0;
36             @(negedge clk);
37         end
38         ss_n=1;
39         @(negedge clk);
40
41         //5
42         ss_n=0; tx_data=$random; tx_valid=1;
43         repeat(30) begin
44             mosi=$random; ss_n=0;
45             @(negedge clk);
46         end
47         ss_n=1;
48         @(negedge clk);
49
50         $stop;
51     end
52
53     initial begin
54         $monitor("rst_n=%b ,ss_n=%b ,mosi=%b ,rx_valid=%b ,rx_data=%b ,tx_valid=%b ,tx_data=%b ,miso=%b ",rst_n,ss_n,mosi,rx_vali
55     end
56
57 endmodule

```

- RAM

```

9  input clk, rst_n, rx_valid;      //Control input signals
10
11 //Output ports
12 output reg [ADDR_SIZE - 1 : 0]dout; //8-bits Output data from the RAM
13 output reg tx_valid;           //Control output signal
14
15 reg [ADDR_SIZE - 1 : 0] mem [MEM_DEPTH - 1 : 0];    //Memory
16 reg [ADDR_SIZE - 1 : 0] w_addr,r_addr;           //Internal registers to store the read and write addresses
17
18 //RAM sequential logic
19 always @(posedge clk) begin
20     //Active low async reset
21     if(~rst_n) begin
22         dout <= 0;
23         tx_valid <= 0;
24         w_addr <= 0;
25         r_addr <= 0;
26     end
27     //rx_valid refers to that there is data on din ready to be read
28     else if (rx_valid) begin
29         //The MSB 2-bits are control signals
30         case (din[9:8])
31             2'b00: w_addr <= din[ADDR_SIZE-1 : 0];    //Store the Write address
32             2'b01: mem[w_addr] <= din[ADDR_SIZE-1 : 0]; //Write the data in the stored write address
33             2'b10: r_addr <= din[ADDR_SIZE-1 : 0];      //Store the Read address
34             2'b11: begin                                //Read the data stored in memory at read address
35                 dout <= mem[r_addr];
36                 tx_valid <= 1;
37             end
38         endcase
39     end
40 end
41 endmodule

```

Module: RAM, Line 41, Column 10  
Spaces: 4 SystemVerilog

- Testbench for the RAM part

```

1 module RAM_tb ();
2     reg [9:0] din;
3     reg clk,rst_n,rx_valid;
4     wire [7:0] dout;
5     wire tx_valid;
6     RAM dut_(clk,rst_n,din,rx_valid,dout,tx_valid);
7     initial begin
8         clk=0;
9         forever begin
10            #5; clk=~clk;
11        end
12    end
13    initial begin
14        $readmemh("mem.dat",dut.mem);
15        rst_n=0; rx_valid=0; din=$random();
16        @(negedge clk);
17        rst_n=1; rx_valid=1;
18        @(negedge clk);
19        repeat(1000) begin
20            din=$random();
21            @(negedge clk);
22        end
23        $stop;
24    end
25 endmodule

```

## Top module

```

1 module SPI_top_module (mosi, ss_n, clk, rst_n, miso);
2
3 //Input ports
4 input clk, rst_n, ss_n; //Control signals
5 input mosi;           //Master_out_slave_in input
6
7 //Output ports
8 output miso;          //Master_in_slave_out output
9
10 //Internal signals
11 wire[9:0] rx_data;    //The parallel data sent to RAM
12 wire rx_valid, tx_valid; //Control signals for the read and write
13 wire[7:0] tx_data;    //The parallel data sent from the RAM to SPI
14
15 //SPI module
16 SPI_Slave dut0 (mosi, ss_n, clk,rst_n, tx_valid, tx_data, miso, rx_valid, rx_data);
17
18 //RAM module
19 RAM dut1 (.clk(clk), .rst_n(rst_n), .rx_valid(rx_valid), .tx_valid(tx_valid), .din(rx_data), .dout(tx_data));
20 endmodule

```

## ● Testbench

```

1  module SPI_top_module_tb ();
2
3  //Ports
4  reg mosi, ss_n, clk,rst_n;
5  wire miso;
6
7  //The SPI module
8  SPI_top_module dut (mosi, ss_n, clk, rst_n, miso);
9
10 //Initiate the clock
11 initial begin
12     clk=0;
13     forever
14         #2 clk=~clk;
15 end
16
18 integer i;
19 initial begin
20
21     //Test reset & intiate signals
22     mosi = 0;    ss_n = 1;
23     rst_n = 0;
24     @(negedge clk);
25     if (miso != 0) begin
26         $display("there was an error");
27         $stop;
28     end
29
30     rst_n = 1;  //Release the reset button
31
32     //Test write & its address
33     repeat (1000) begin
34         ss_n = 0;                      //Release the ss_n to start communication
35         @(negedge clk);
36
37         //Sends the Write address
38         for (i = 0; i < 10; i = i + 1) begin
39             if (i < 2) begin
40                 mosi = 0;                  //first 2 bits are 00 to intiate write address
41             end
42             else
43                 mosi = $random;        //sends 8 bits to be stored in the write address
44             @(negedge clk);
45         end
46
47         repeat(2) @(negedge clk);      //Wait for the write address to read din in 2 clocks
48
49         ss_n = 1;                    //Set ss_n to stop the communication and start new one
50         @(negedge clk);
51
52         ss_n = 0;                    //Release the ss_n to start new task
53         @(negedge clk);
54
55         //Sends the data to be stored
56         for (i = 0; i < 10; i = i + 1) begin
57             if (i == 0) begin
58                 mosi = 0;              //first 2 bits are 01 to intiate write data
59             end
60             else if (i == 1) begin
61                 mosi = 1;
62             end
63             else
64                 mosi = $random;       //sends 8 bits to be stored in the memory
65             @(negedge clk);
66         end
67
68         repeat(2) @(negedge clk);      //Wait for the RAM to store din in 2 clocks
69
70         ss_n = 1;                    //Set ss_n to stop the communication and start new one
71         @(negedge clk);
72     end
73     $stop;                         //Pause the sim after writing the memory
74

```

```

4      //test Read & its address
5      repeat (1000) begin
6          ss_n = 0;                                //Release the ss_n to start communication
7          @(negedge clk);
8
9          // sends the Read address
10         for (i = 0; i < 10; i = i + 1) begin
11             if (i == 0) begin
12                 mosi = 1;                      //first 2 bits are 10 to intiate Read address
13             end
14             else if (i == 1) begin
15                 mosi = 0;
16             end
17             else
18                 mosi = $random;           //sends 8 bits to be stored in the Read address
19             @(negedge clk);
20         end
21
22         repeat(2) @(negedge clk);           //Wait for the read address to store din in 2 clocks
23
24         ss_n = 1;                          //Set ss_n to stop the communication and start new one
25         @(negedge clk);
26
27         ss_n = 0;                          //Release the ss_n to start a new task
28         @(negedge clk);
29
30
31         //Receives the read data
32         for (i = 0; i < 10; i = i + 1) begin
33             if (i < 2) begin
34                 mosi = 1;                  //first 2 bits are 11 to intiate read data
35             end
36             else
37                 mosi = $random;       //the rest of the 8 bits are dummy data
38             @(negedge clk);
39         end
40
41         repeat(2) @(negedge clk);           //Wait for the RAM to send the data on TX_data from mem read address stored
42
43         repeat(9) @(negedge clk);          //Wait for the SPI slave to transfer the TX_data from parallel to series on the mis
44
45         ss_n = 1;                          //Set ss_n to stop the communication and start new one
46         @(negedge clk);
47     end
48     $stop;                            //End of simulation
49 end
50 endmodule

```

- DO file

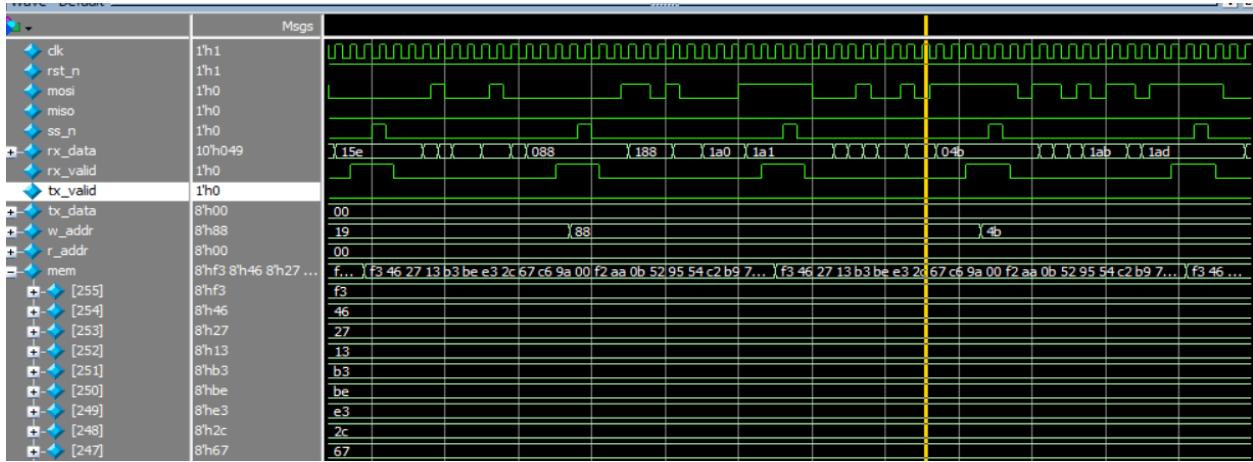
```

vlib work
vlog RAM.v SPI_Slave.v SPI_top_module.v SPI_tb.v
vsim -voptargs+=acc work.SPI_top_module_tb
add wave *
add wave -position insertpoint \
sim:/SPI_top_module_tb/dut/rx_data \
sim:/SPI_top_module_tb/dut/rx_valid \
sim:/SPI_top_module_tb/dut/tx_valid \
sim:/SPI_top_module_tb/dut/tx_data
add wave -position insertpoint \
sim:/SPI_top_module_tb/dut/dut1/mem \
sim:/SPI_top_module_tb/dut/dut1/w_addr \
sim:/SPI_top_module_tb/dut/dut1/r_addr
run -all
#quit -sim

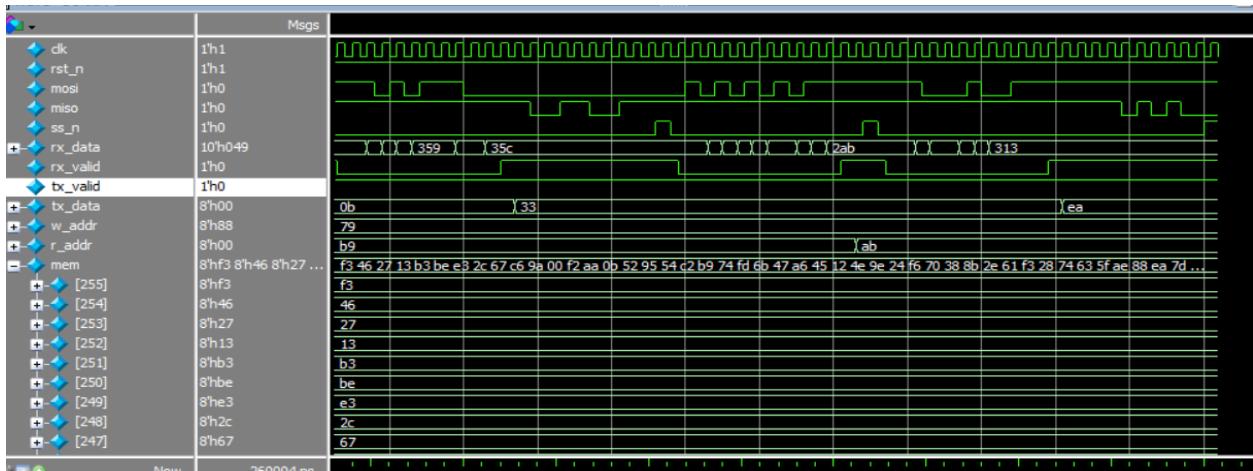
```

# Simulation

- Write operation



- Read operation



# Constrain file

```
## Clock signal
set_property -dict { PACKAGE_PIN W5    IOSTANDARD LVC MOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

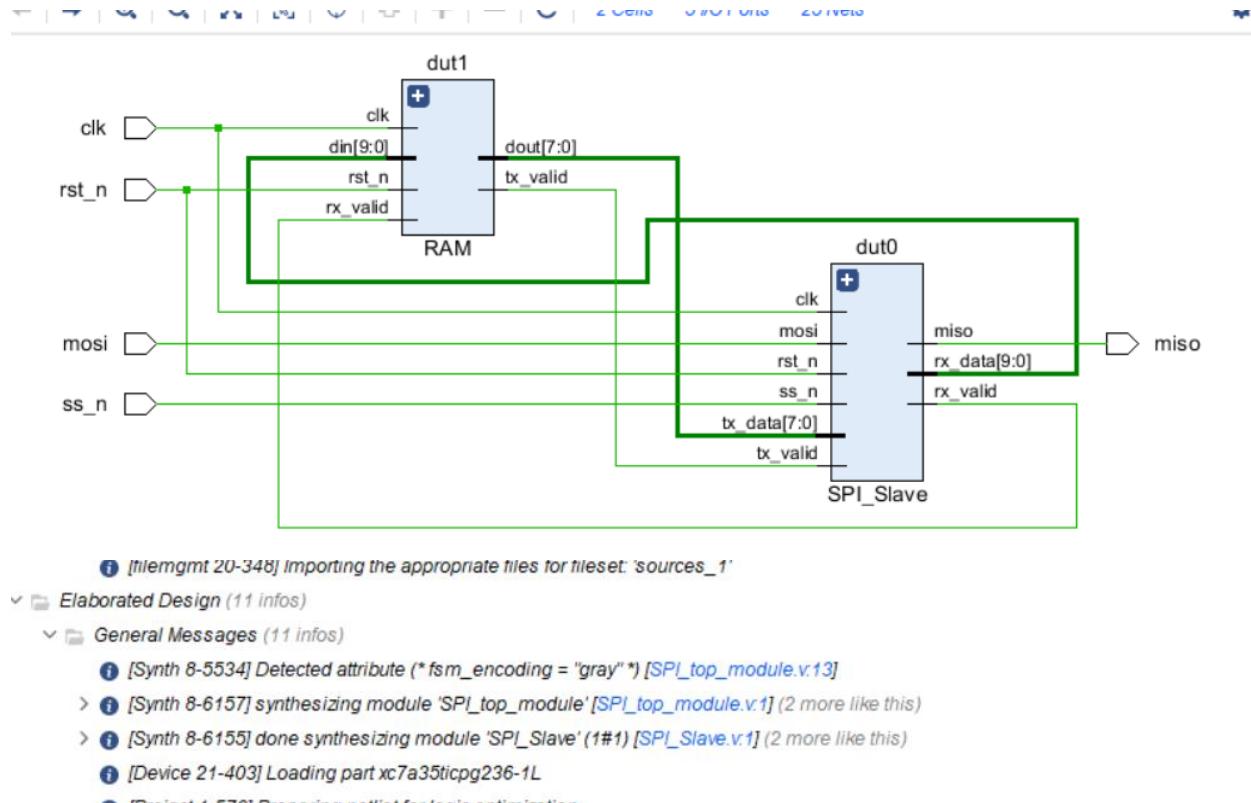
10
11 ## Switches
12 set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVC MOS33 } [get_ports {rst_n}]
13 set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVC MOS33 } [get_ports {ss_n}]
14 set_property -dict { PACKAGE_PIN W16    IOSTANDARD LVC MOS33 } [get_ports {mosi}]
15 #set_property -dict { PACKAGE_PIN W17    IOSTANDARD LVC MOS33 } [get_ports {sw[3]}]
16 #set_property -dict { PACKAGE_PIN W15    IOSTANDARD LVC MOS33 } [get_ports {sw[4]}]
17 #set_property -dict { PACKAGE_PIN V15    IOSTANDARD LVC MOS33 } [get_ports {sw[5]}]
18 #set_property -dict { PACKAGE_PIN W14    IOSTANDARD LVC MOS33 } [get_ports {sw[6]}]

9
0 ## LEDs
1 set_property -dict { PACKAGE_PIN U16    IOSTANDARD LVC MOS33 } [get_ports {miso}]
2 #set_property -dict { PACKAGE_PIN E19    IOSTANDARD LVC MOS33 } [get_ports {led[1]}]
```

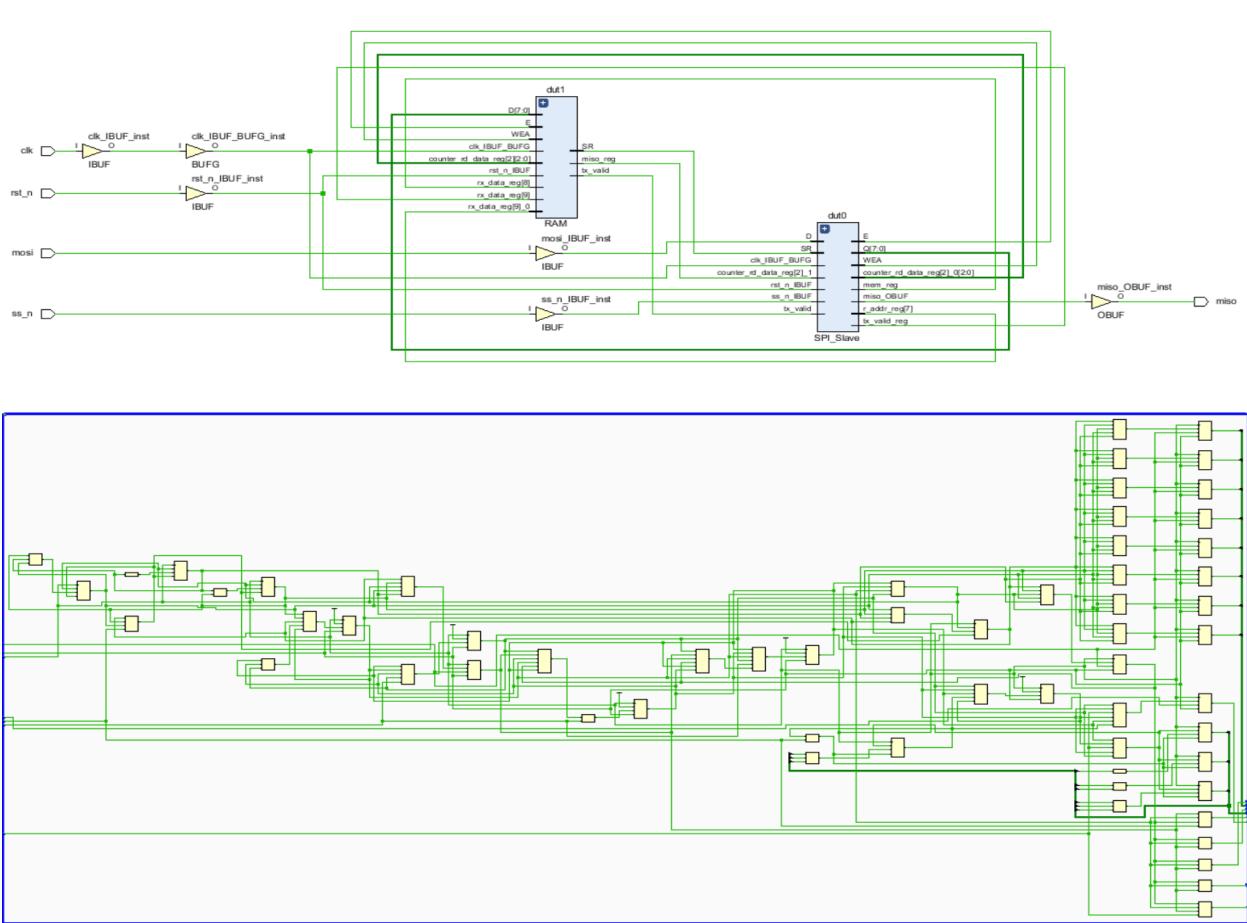
We will try Elaboration, Synthesis and Implementation using different encoding techniques.

### 1- Gray encoding:-

- Elaboration



## • Synthesis



### - Messages

```

- Synthesis (30 infos)
  ① [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
  ① [Synth 8-6157] synthesizing module 'SPI_top_module'[SPI_top_module.v:1] (2 more like this)
  ① [Synth 8-5534] Detected attribute (* fsm_encoding = "gray") [SPI_Slave.v:23]
  ① [Synth 8-6155] done synthesizing module 'SPI_Slave'(#1) [SPI_Slave.v:1] (2 more like this)
  ① [Device 21-403] Loading part xc7a35t/cpg236-1L
  ① [Project 1-236] Implementation specific constraints were found while reading constraint file [C:/Users/elnoor/Desktop/SPI/project_1/srcs/constrs_1/imports/SPI/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_top_module_prop/impl.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.
  ① [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
  ① [Synth 8-5544] ROM 'rx_valid' won't be mapped to Block RAM because address size (3) smaller than threshold (5) (3 more like this)
  ① [Synth 8-3354] encoded FSM with state register cs_reg using encoding 'gray' in module 'SPI_Slave'

```

### - Utilization

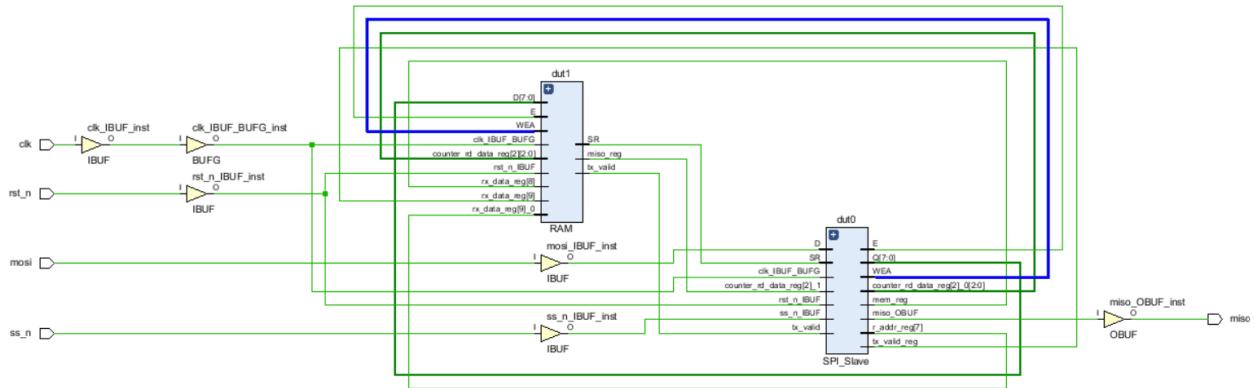
Name	1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
④ SPI_top_module		34	41	1	0.5	5	1
④ dut0 (SPI_Slave)		31	24	0	0	0	0
④ dut1 (RAM)		3	17	1	0.5	0	0

- Timing

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.261 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: 101	Total Number of Endpoints: 101	Total Number of Endpoints: 44

All user specified timing constraints are met.

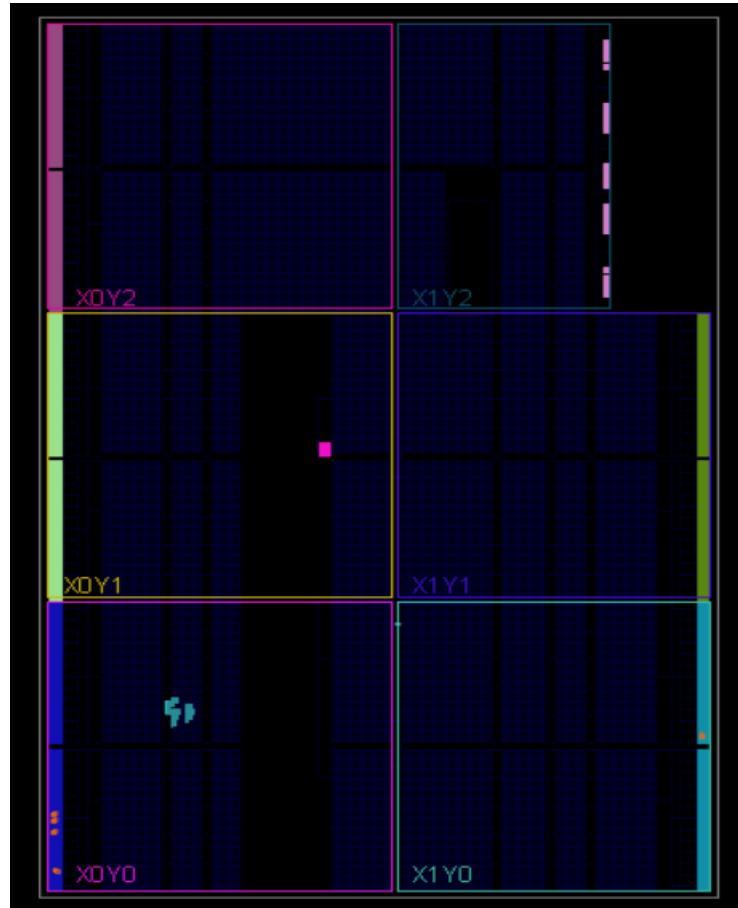
- Critical path



- Synthesis summary (encoding technique)

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

- Implementation



- Utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (815 0)	LUT as Logic (20800)	LUTFlip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	
▀ N SPI_top_module	35	41	1	17	35		10	0.5	5	1
▀ dut0 (SPI_Slave)	31	24	0	16	31		9	0	0	0
▀ dut1 (RAM)	4	17	1	6	4		0	0.5	0	0

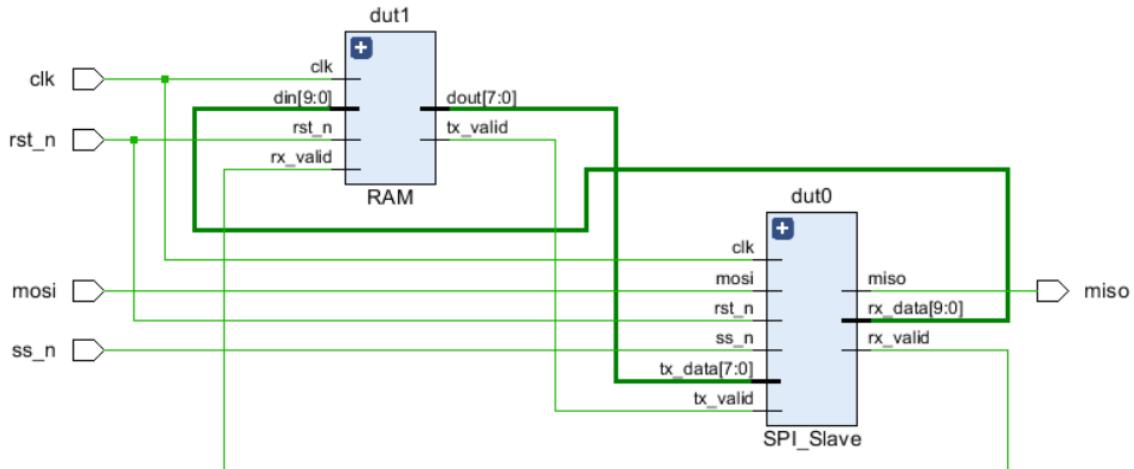
- Timing

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.049 ns	Worst Hold Slack (WHS): 0.110 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: 102	Total Number of Endpoints: 102	Total Number of Endpoints: 44

All user specified timing constraints are met.

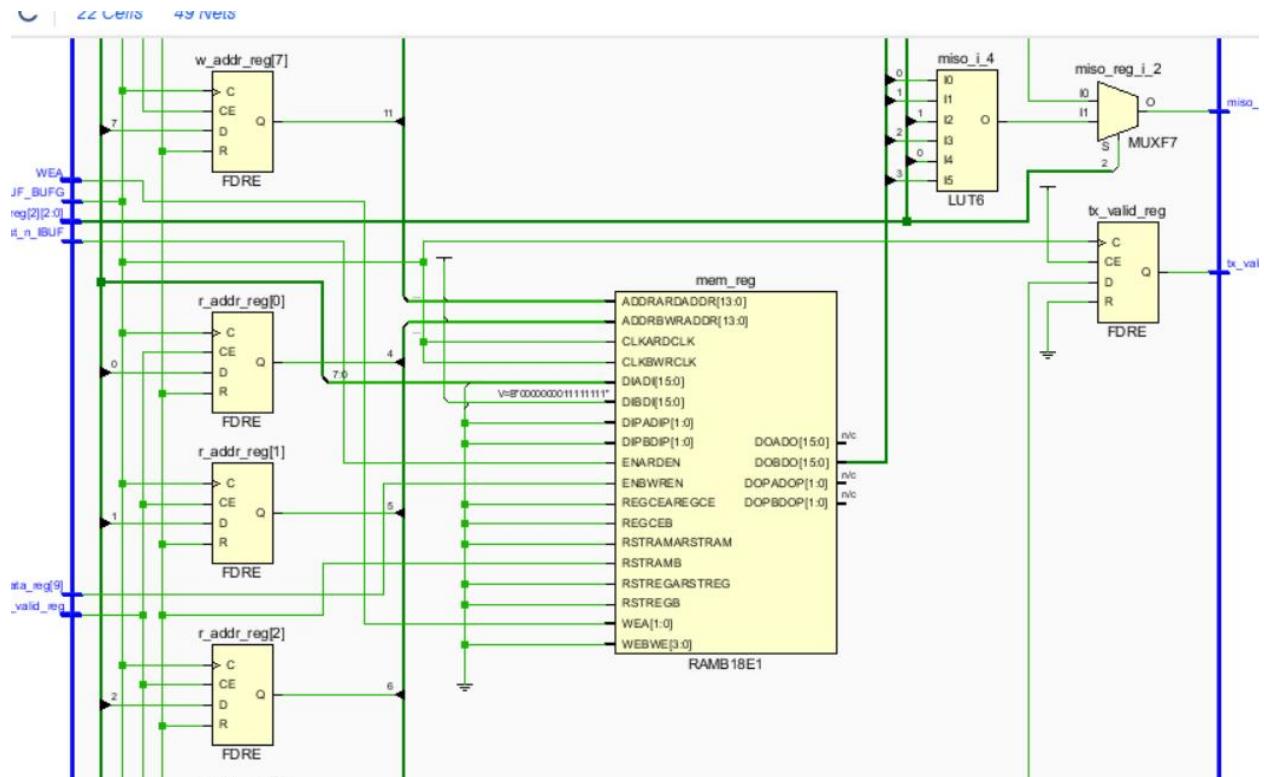
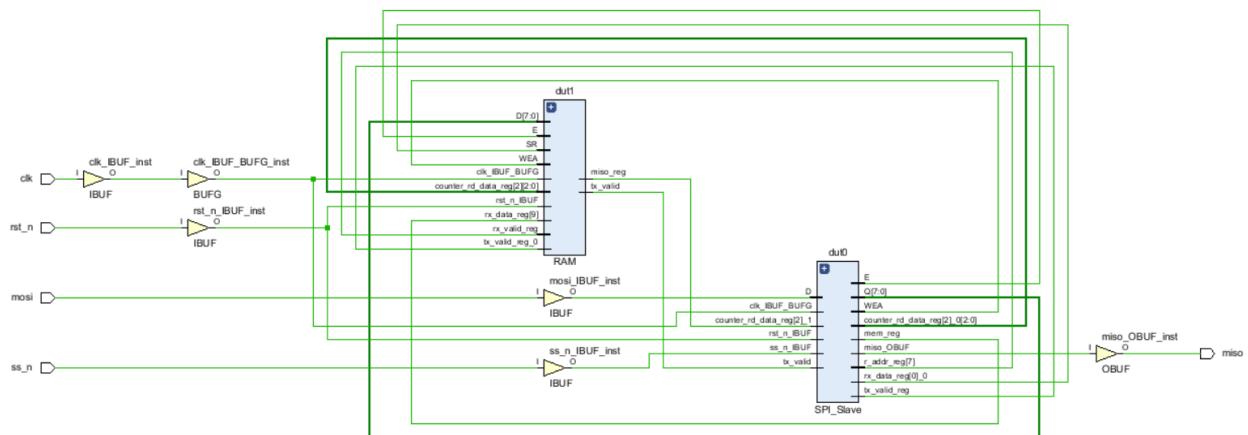
## 2- Seq encoding:-

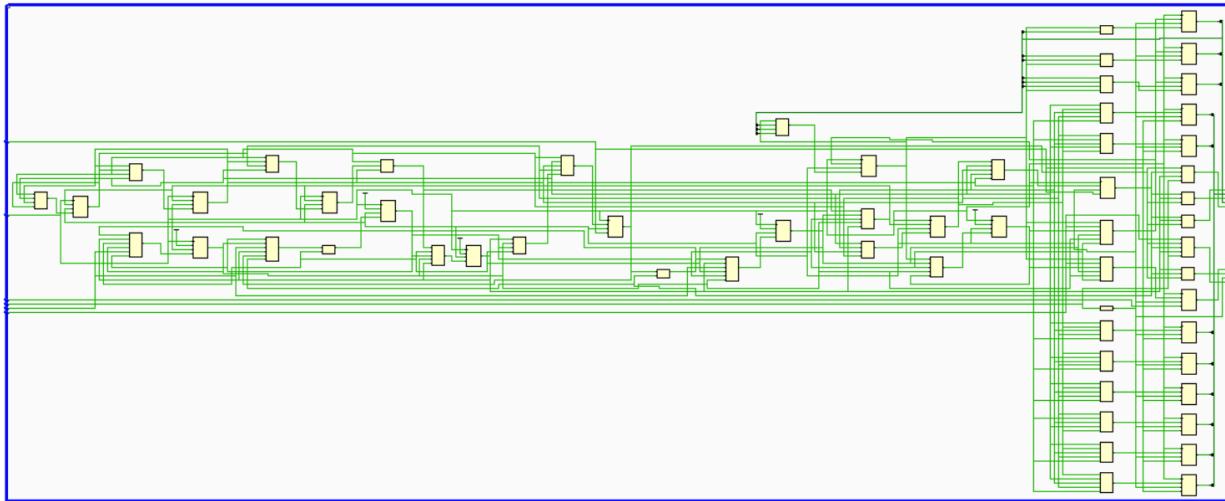
- Elaboration



```
✓ Elaborated Design (10 infos)
  ✓ General Messages (10 infos)
    ⓘ [Synth 8-5534] Detected attribute (* fsm_encoding = "seq" *) [SPI_top_module.v:13]
    > ⓘ [Synth 8-6157] synthesizing module 'SPI_top_module' [SPI_top_module.v:1] (2 more like this)
    > ⓘ [Synth 8-6155] done synthesizing module 'SPI_Slave'(1#1) [SPI_Slave.v:1] (2 more like this)
    ⓘ [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.
```

- Synthesis





## - Messages

✓ Synthesized Design (6 infos)

✓ General Messages (6 infos)

- ⓘ [Netlist 29-17] Analyzing 6 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.

## - Utilization

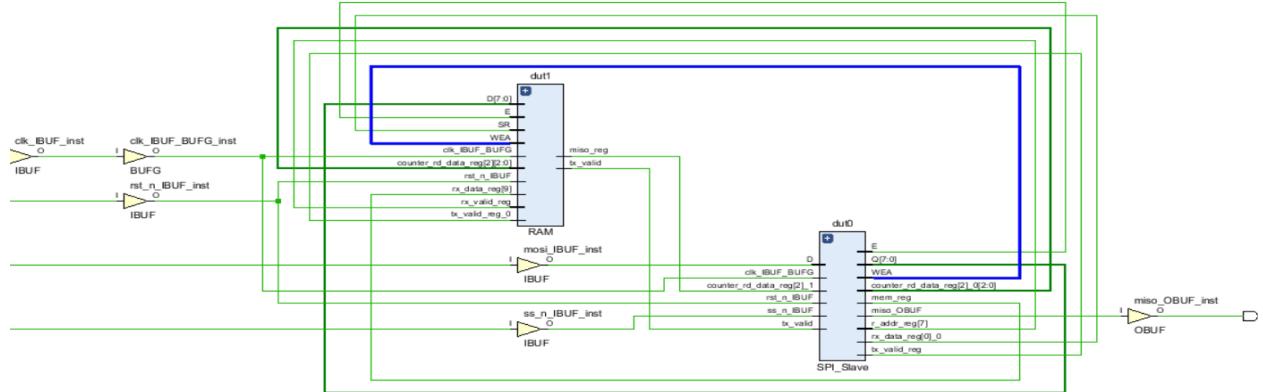
	Name	1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	Bonded I/OB (106)	BUFGCTRL (32)	
'<1%									
Logic (<1%)	✓ N SPI_top_module	32	41	1	0.5	5	1		
1%)	dbg_hub (dbg_hub_CV)	0	0	0	0	0	0	0	
ers (<1%)	dut0 (SPI_Slave)	30	24	0	0	0	0	0	
	dut1 (RAM)	2	17	1	0.5	0	0	0	

## - Timing

Design Timing Summary					
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.261 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:	93	Total Number of Endpoints:	93	Total Number of Endpoints:	44

All user specified timing constraints are met

- Critical path



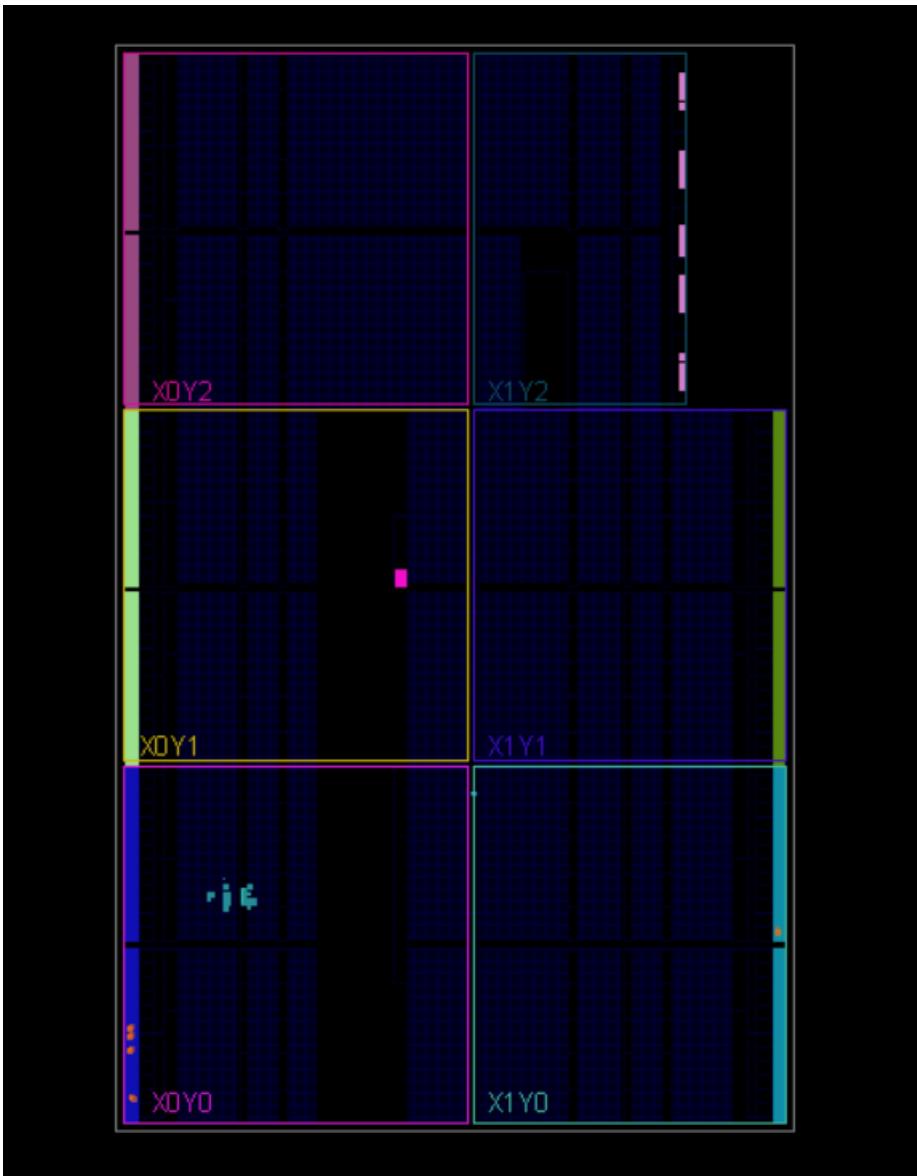
- Synthesis summary (encoding technique)

```
INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
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	000	000
CHK_CMD	001	001
WRITE	010	010
READ_ADD	011	011
READ_DATA	100	100

```
INFO: [Svnth 8-33541] encoded FSM with state register 'cs reg' using encoding 'sequential' in module 'SPI Slave'
```

- Implementation



✓ Implemented Design (9 infos)

  ✓ General Messages (9 infos)

- [Netlist 29-17] Analyzing 6 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*
- [Timing 38-478] Restoring timing data from binary archive.*
- [Timing 38-479] Binary timing data restore complete.*
- [Project 1-856] Restoring constraints from binary archive.*

- Utilization

Q | Hierarchy

Name	1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	
N SPI_top_module		32	41	1	0.5	5	1	
dut0 (SPI_Slave)		30	24	0	0	0	0	
dut1 (RAM)		2	17	1	0.5	0	0	

- Timing

Ports | Design Runs | Power | DRC | Methodology | **Timing** x ? — □

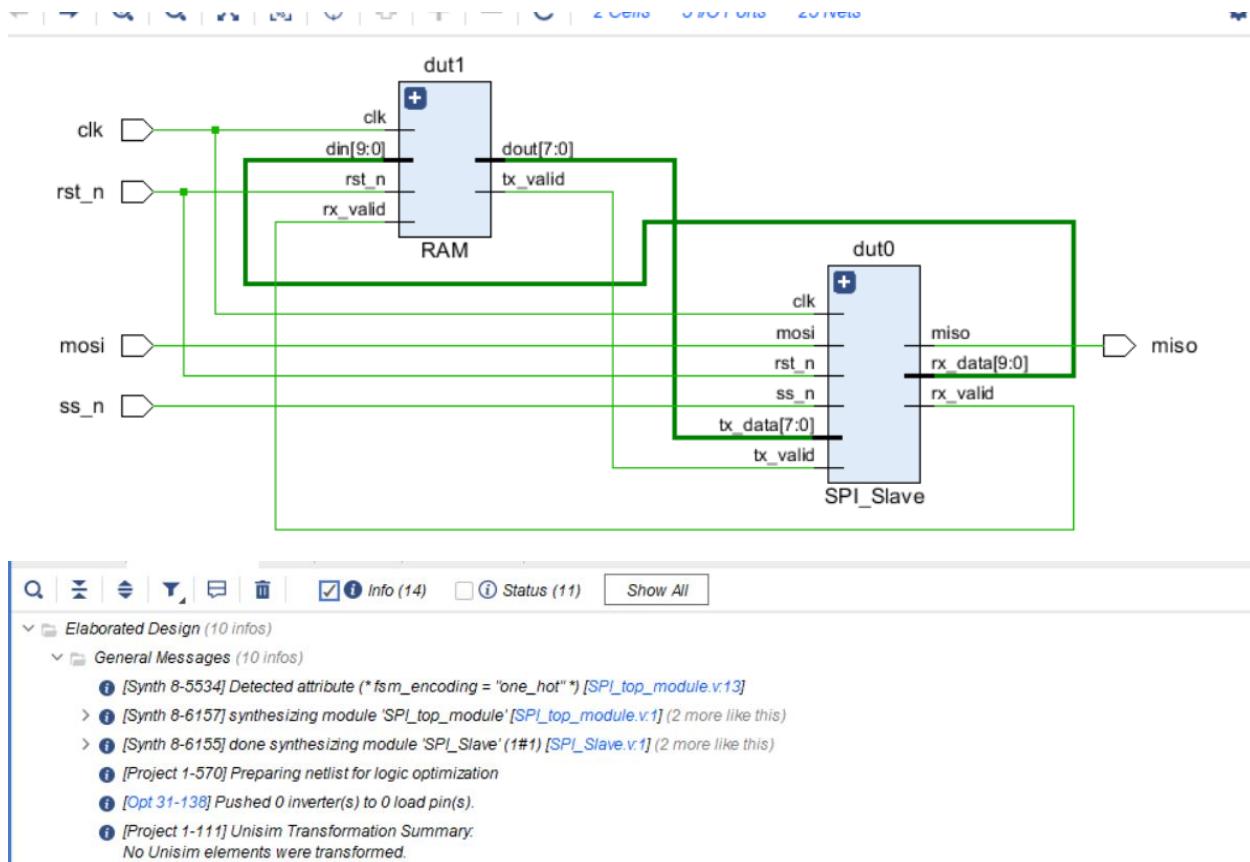
◀ Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): <b>6.052 ns</b>	Worst Hold Slack (WHS): <b>0.102 ns</b>	Worst Pulse Width Slack (WPWS): <b>4.500 ns</b>
Total Negative Slack (TNS): <b>0.000 ns</b>	Total Hold Slack (THS): <b>0.000 ns</b>	Total Pulse Width Negative Slack (TPWS): <b>0.000 ns</b>
Number of Failing Endpoints: <b>0</b>	Number of Failing Endpoints: <b>0</b>	Number of Failing Endpoints: <b>0</b>
Total Number of Endpoints: <b>94</b>	Total Number of Endpoints: <b>94</b>	Total Number of Endpoints: <b>44</b>

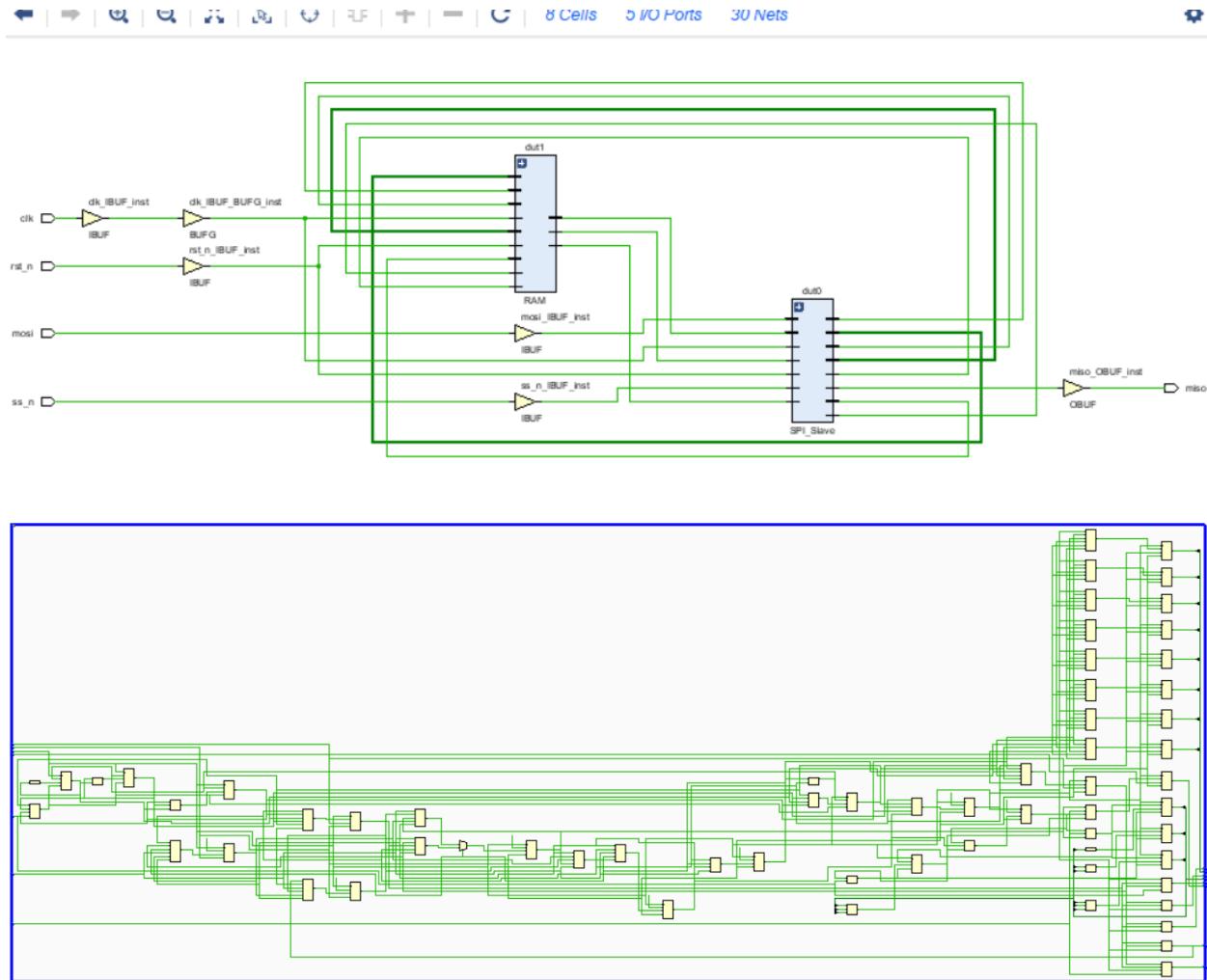
All user specified timing constraints are met.

### 3- One\_hot encoding:-

- Elaboration



## ● Synthesis



### - Messages

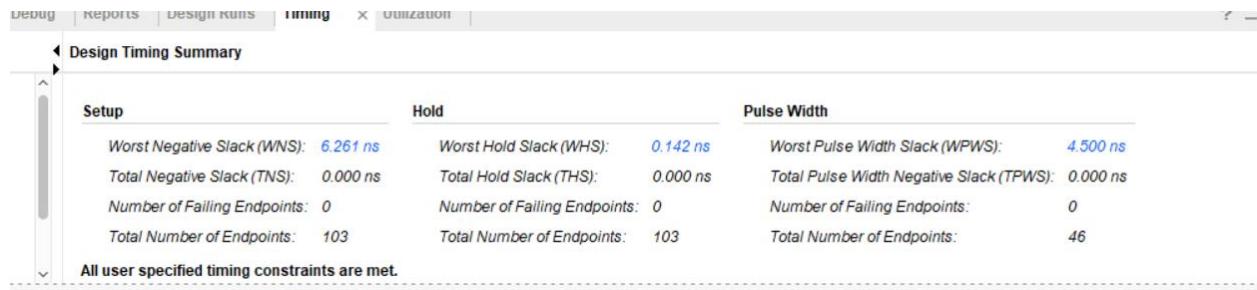
```

    ⓘ [filemgmt 20-348] Importing the appropriate files for fileset: 'sources_1'
    ⓐ [Synthesis (1 warning, 31 infos)
      ⓘ [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
      > ⓘ [Synth 8-6157] synthesizing module 'SPI_top_module'[SPI_top_module.v1] (2 more like this)
      > ⓘ [Synth 8-5534] Detected attribute (* fsm_encoding = "one_hot"*) [SPI_top_module.v1] (1 more like this)
      > ⓘ [Synth 8-6155] done synthesizing module 'SPI_Slave'(1#1) [SPI_Slave.v1] (2 more like this)
      ⓘ [Device 21-403] Loading part xc7a35ticpg236-1L
      ⓘ [Project 1-236] Implementation specific constraints were found while reading constraint file [C:/Users/einoor/Desktop/SPI/project_2/srcs/constrs_1/imports/SPI/Constraints_basics3.xdc]. 1
      
```

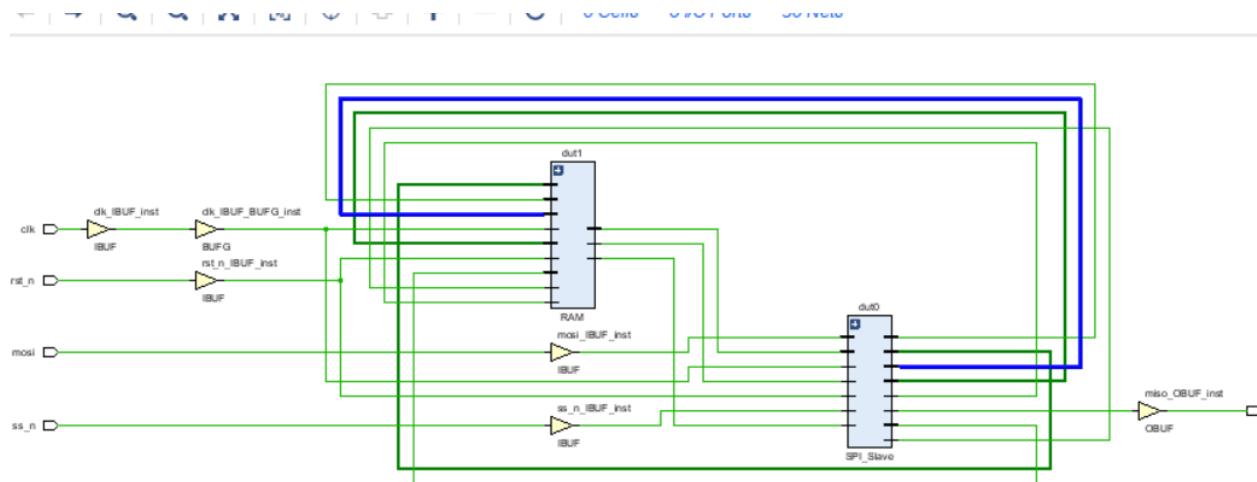
### - Utilization

Name	1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
▀ N SPI_top_module		35	43	2	0.5	5	1
▀ dut0 (SPI_Slave)		32	26	1	0	0	0
▀ dut1 (RAM)		3	17	1	0.5	0	0

- Timing



- Critical path



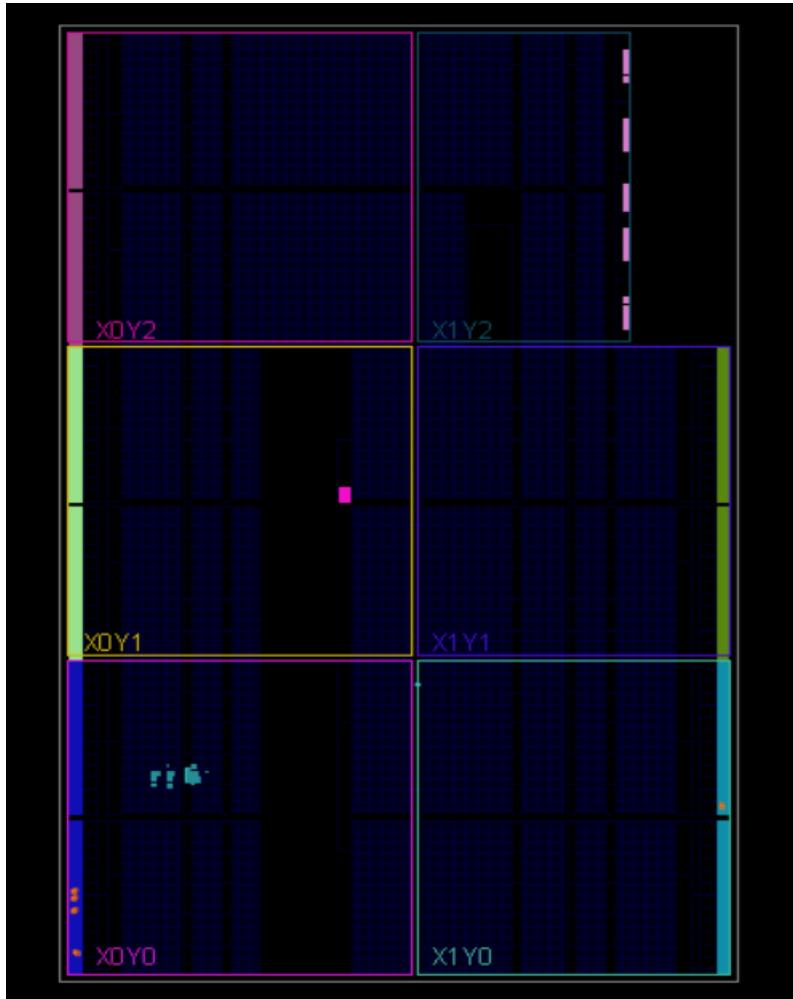
- Synthesis summary (encoding technique)

```

1 INFO: [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
2 -----
3      State |           New Encoding |           Previous Encoding
4 -----
5          IDLE |           00001 |           000
6          CHK_CMD |           00010 |           001
7          WRITE |           00100 |           010
8          READ_ADD |           01000 |           011
9          READ_DATA |           10000 |           100
10 -----
11 INFO: [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'one-hot' in module 'SPI_Slave'
12 -----
13 Finished RTL Optimization Phase 2 : Time (s): cpu = 00:00:32 ; elapsed = 00:00:35 . Memory (MB): peak = 765.305 ; ga
14 -----

```

- Implementation



✓ *Implemented Design (10 infos)*

✓ *General Messages (10 infos)*

- [Netlist 29-17] Analyzing 7 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Device 21-403] Loading part xc7a35ticpg236-1L
- [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.
- [Project 1-856] Restoring constraints from binary archive.
- [Project 1-853] Binary constraint restore complete.
- [Project 1-111] Unisim Transformation Summary:  
No Unisim elements were transformed.

- Utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (815 0)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	
▀ N SPI_top_module	36	43	2	19	36		12	0.5	5	1
█ dut0 (SPI_Slave)	32	26	1	16	32		11	0	0	0
█ dut1 (RAM)	4	17	1	6	4		0	0.5	0	0

- Timing

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.212 ns	Worst Hold Slack (WHS): 0.070 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: 104	Total Number of Endpoints: 104	Total Number of Endpoints: 46

All user specified timing constraints are met.

## 4- Comparison between techniques

Technique	Setup slack	Hold slack
Gray	6.049 ns	0.110 ns
Seq	6.052 ns	0.102 ns
One_hot	6.212 ns	0.070 ns

So as we considered gray encoding gives the best in the worst hold slack and one\_hot encoding gives the best in the worst setup slack.

## 5- Bitstream generation

