

Final Project

SPI slave with single port RAM

Reggit

DIGITAL DESIGN ENGINEER:

Mohamed Khaled Ahmed

Under the supervision of: Eng. Kareem Waseem

First: RTL design**Single port Ram RTL code:**

```

E:\Digital Design course > project_2 > codes > Single_Port_Async_RAM.v
1  module Single_Port_Async_RAM (din,clk,rst_n,rx_valid,dout,tx_valid);
2
3  parameter MEM_DEPTH = 256;
4  parameter ADDR_SIZE = 8;
5
6  input [9:0] din;
7  input clk, rx_valid, rst_n; //active low synchronous
8
9  //Vivado may not be accepting the asynchronous control signals and may not map them directly on the FPGA board as
10 //FSM or RAM. If you experience this during implementation, then change the reset to be synchronized with the clock.
11 //so used the active low synch rst
12 output reg [7:0] dout;
13 output reg tx_valid;
14
15 reg [7:0] RAM [MEM_DEPTH - 1 : 0];
16
17 // Two Address for write and read respectively
18 reg [ADDR_SIZE - 1 : 0] Wr_Addr, Rd_Addr;

```

```

19
20 // read/write----> address
21 always @(posedge clk) begin
22     if (~rst_n) begin
23         dout <= 0;
24         tx_valid <= 0;
25         Wr_Addr <= 0;
26         Rd_Addr <= 0;
27     end
28     else begin
29         //to "Read Data" tx_valid must be -> 1 in order to read data from the SPI slave
30         tx_valid <= (din[9] & din[8] & rx_valid)? 1 : 0;
31         //rx_valid -> 1 , the din[7:0] is accepted
32         if (rx_valid) begin
33             case (din[9:8])
34                 2'b00 : Wr_Addr <= din[7:0]; // Write Address
35                 2'b01 : RAM[Wr_Addr] <= din[7:0]; // Write Data
36                 2'b10 : Rd_Addr <= din[7:0]; // Read Address
37                 2'b11 : dout <= RAM[Rd_Addr]; // Read Data
38             endcase
39         end
40     end
41 end
42 endmodule

```

SPI slave RTL code:

```
E: > Digital Design course > project_2 > codes > SPI_Slave_Interface.v
1  module SPI_Slave_Interface (MOSI,MISO,SS_n,clk,rst_n,rx_data,rx_valid,tx_data,tx_valid);
2
3  parameter IDLE = 3'b000;
4  parameter CHK_CMD = 3'b001;
5  parameter WRITE = 3'b010;
6  parameter READ_ADD = 3'b011;
7  parameter READ_DATA = 3'b100;
8
9  //FSM Encoding Method ---->GRAY
10
11  (* fsm_encoding = "gray" *)
12
13  // Input Declaration
14  input MOSI, SS_n, clk, rst_n, tx_valid; //Active Low SYNC rst
15  input [7:0] tx_data;
16
17  output reg MISO, rx_valid;
18  output reg [9:0] rx_data;
19
20  //ensure that Read Address Comes 1st and Read Data Comes 2nd
21  reg rd_addr_recieved; //high ---> READ SIGNAL IS RECEIVED
22
23  reg [2:0] cs, ns; |
24  reg[3:0] counter;
25
```

```
26
27  // State Memory Logic
28  always @(posedge clk) begin
29      if (~rst_n)
30          cs <= IDLE;
31      else
32          cs <= ns;
33  end
34
```

```
35  // Next State Logic
36  always @(*) begin
37      case (cs)
38          IDLE :
39              begin
40                  if (SS_n)
41                      ns = IDLE;
42                  else
43                      ns = CHK_CMD;
44              end

```

```

45 ~      CHK_CMD :
46 ~          begin
47 ~              if (SS_n)
48 ~                  ns = IDLE;
49 ~              else if (~MOSI)
50 ~                  ns = WRITE;
51 ~              else if (~rd_addr_recieved)
52 ~                  ns = READ_ADD;
53 ~              else
54 ~                  ns = READ_DATA;
55 ~          end
56 ~      WRITE :
57 ~          begin
58 ~              if (SS_n)
59 ~                  ns = IDLE;
60 ~              else
61 ~                  ns = WRITE;
62 ~          end
63 ~      READ_ADD :
64 ~          begin
65 ~              if (SS_n)
66 ~                  ns = IDLE;
67 ~              else
68 ~                  ns = READ_ADD;
69 ~          end

```

```

69 ~          end
70 ~      READ_DATA :
71 ~          begin
72 ~              if (SS_n)
73 ~                  ns = IDLE;
74 ~              else
75 ~                  ns = READ_DATA;
76 ~          end
77 ~      default : ns = IDLE;
78 ~  endcase
79 ~ end
80

```

```

80 //output logic
81 always @(posedge clk) begin
82     if (~rst_n) begin
83         rx_data <= 0;
84         rx_valid <= 0;
85         rd_addr_recieved <= 0;
86         MISO <= 0;
87         counter <= 0;
88     end

```

```

89  ~   else begin
90  ~       case (cs)
91  ~           IDLE :
92  ~               begin
93  ~                   counter <= 0;
94  ~                   rx_valid <= 0;
95  ~                   MISO <= 0;
96  ~               end
97  ~           CHK_CMD :
98  ~               begin
99  ~                   counter <= 0;
100 ~                   rx_valid <= 0;
101 ~               end
102 ~           WRITE :
103 ~               begin
104 ~                   if (counter <= 9) begin
105 ~                       rx_data <= {rx_data[8:0],MOSI};
106 ~                       rx_valid <= 0;
107 ~                       counter <= counter + 1;
108 ~                   end
109 ~                   if (counter >= 9) begin
110 ~                       rx_valid <= 1;
111 ~                   end
112 ~               end

```

```

113 ~           READ_ADD :
114 ~               begin
115 ~                   if (counter <= 9) begin
116 ~                       rx_data <= {rx_data[8:0],MOSI};
117 ~                       rx_valid <= 0;
118 ~                       rd_addr_recieved <= 1;
119 ~                       counter <= counter + 1;
120 ~                   end
121 ~                   if (counter >= 9)
122 ~                       rx_valid <= 1;
123 ~               end
124 ~           READ_DATA :
125 ~               begin
126 ~                   if(tx_valid && counter >= 3) begin
127 ~                       MISO <= tx_data[counter - 3];
128 ~                       counter <= counter - 1;
129 ~                   end
130 ~                   else
131 ~                       if(counter <= 9) begin
132 ~                           rx_data <= {rx_data[8:0],MOSI}; // recieved bits are MSB TO LSB
133 ~                           rx_valid <= 0;
134 ~                           counter <= counter + 1;
135 ~                       end
136 ~                       if(counter >= 9) begin
137 ~                           rx_valid <= 1;
138 ~                           rd_addr_recieved <= 0;
139 ~                       end
140 ~                   end
141 ~               endcase
142 ~           end
143 ~       end
144 ~   endmodule

```

SPI Wrapper RTL code (TOP module):

```
E: > Digital Design course > project_2 > codes > SPI_Wrapper.v
1  module SPI_Wrapper (MOSI,MISO,SS_n,clk,rst_n);
2
3  parameter MEM_DEPTH = 256;
4  parameter ADDR_SIZE = 8;
5
6  input MOSI, SS_n, clk, rst_n;
7  // SS_n == 0 then, master begin Communication
8  // SS_n == 1 then, master end Communication
9  output MISO;
10
11 wire [9:0] rx_data_din;
12 wire rx_valid;
13 wire [7:0] tx_data_dout;
14 wire tx_valid;
15
16 SPI_Slave_Interface SPI_Slave_inst (MOSI,MISO,SS_n,clk,rst_n,rx_data_din,rx_valid,tx_data_dout,tx_valid);
17
18 Single_Port_Async_RAM #(MEM_DEPTH,ADDR_SIZE) RAM_inst (rx_data_din,clk,rst_n,rx_valid,tx_data_dout,tx_valid);
19
20 endmodule
```

mem.text:

//It is a 254 rows text of randomized numbers which will be used in the testbench

```
E: > Digital Design course > project_2 > codes > mem.txt
```

```
1  7B
2  E9
3  CA
4  B0
5  39
6  8E
7  13
8  C7
9  63
```

Second: Testbench

//Testbench scenario:

```
20 //TESTBENCH PLAN
21 //=====
22 //testbench will go as follow:
23 //=====
24 //rst check
25 //normal test for RAM read operation --> working on the FSM diagram
26 //normal test for RAM write operation --> working on the FSM diagram
27 //test on write address separately---> followed by write data test (self checking)
28 //test on read address separately---> followed by read data test (self checking)
29 //the verification engineer is acting as the master in giving values (self checking)
```

//tb module:

```
E: > Digital Design course > project_2 > codes > SPI_Master_tb.v
1  module SPI_Master_tb ();
2
3  parameter MEM_DEPTH = 256 ;
4  parameter ADDR_SIZE = 8 ;
5  reg clk , rst_n , MOSI , SS_n;
6  wire MISO;
7
8  SPI_Wrapper #(MEM_DEPTH,ADDR_SIZE) DUT (MOSI,MISO,SS_n,clk,rst_n);
9
10 // Clock Generation
11 initial begin
12     clk = 0;
13     forever
14         #5 clk = ~clk; // then negedge sequence will be 0, 10, 20, .... SO,CLK Period = 10
15 end
16
17 reg [7:0] wr_add, rd_add;
18 reg [7:0] wr_data, rd_data;
19
```

//RESET check:


```

31 // Signal to Check MOSI Sequence in Read Operation
32 reg read_sequence;
33
34
35 ~ initial begin
36
37     $readmemh ("mem.txt",DUT.RAM_inst.RAM);
38
39     $display("Test Reset Operation");
40     rst_n = 0;
41     {wr_add,rd_add,wr_data,rd_data} = 4'b0000;
42 ~ repeat(5) begin
43     MOSI = $random;
44     SS_n = $random;
45     @(negedge clk);
46 end
47 rst_n = 1;

```

//normal read operation of RAM:

```

48
49     read_sequence = 1; // Initially Ensure That The Read Sequence is Correct
50
51     $display("Test Normal RAM Read Operation");
52 ~ repeat(4) begin
53     SS_n = 0; // Start Communication
54     MOSI = $random;
55     @(negedge clk);
56     MOSI = 1; // Read Operation
57     read_sequence = ~read_sequence;
58
59     @(negedge clk); // din[9] = 1'b1
60     @(negedge clk); // din[9:8] = 2'b11
61     MOSI = read_sequence;
62 ~ repeat(10) begin
63     @(negedge clk);
64     MOSI = $random;
65 end
66 ~ if (~read_sequence)
67     SS_n = 1; // End Communication
68 ~ else begin
69     repeat(8) @(negedge clk);
70 end
71     SS_n = 1;
72     repeat(3) @(negedge clk);
73 end
74

```

//normal write operation of RAM

```

76    $display("Test Normal RAM Write Operation");
77    // Second Test Write Operation
78    repeat(4) begin
79        SS_n = 0;
80        MOSI = $random;
81        @(negedge clk);
82        MOSI = 0; // Write Operation
83        @(negedge clk); // din[9] = 1'b0;
84        @(negedge clk); // din[9:8] = 2'b00;
85        MOSI = $random;
86        repeat(10) begin // Data to be Written
87            @(negedge clk);
88            MOSI = $random;
89        end
90        SS_n = 1;
91        repeat(3) @(negedge clk);
92    end

```

//write address operation test:

```

94    //testing for each operation separately
95    $display("Write Address Operation");
96    SS_n = 0;
97    @(negedge clk);
98    MOSI = 0;
99    repeat(3) @(negedge clk);
100
101    repeat (8) begin
102        MOSI = $random;
103        wr_add = {wr_add[6:0],MOSI};
104        @(negedge clk);
105    end
106    SS_n = 1;
107    @(negedge clk);
108    // Check for Write Address Value
109    if (DUT.RAM_inst.Wr_Addr == wr_add)
110        $display("Write Address is Done Correctly");
111    else begin
112        $display("Error in Write Address");
113        $stop;
114    end
115    @(negedge clk);
116

```

//write data operation test:

```

117     $display("Write Data Operation");
118     SS_n = 0;
119     @(negedge clk);
120
121     MOSI = 0;
122     repeat(2) @(negedge clk);
123     MOSI = 1;
124     @(negedge clk);
125
126     repeat(8) begin
127         MOSI = $random;
128         wr_data = {wr_data[6:0],MOSI};
129         @(negedge clk);
130     end
131     SS_n = 1;
132     @(negedge clk);
133     // Check for Write Data Value
134     if (DUT.RAM_inst.RAM[DUT.RAM_inst.Wr_Addr] == wr_data)
135         $display("Write Data is Done Correctly");
136     else begin
137         $display("Error in Write Data");
138         $stop;
139     end
140     @(negedge clk);

```

//read address operation test:

```

141
142     $display("Read Address Operation");
143     SS_n = 0;
144     @(negedge clk);
145     MOSI = 1;
146     repeat(2) @(negedge clk);
147     MOSI = 0;
148     @(negedge clk);
149
150     repeat(8) begin
151         MOSI = $random;
152         rd_add = {rd_add[6:0],MOSI};
153         @(negedge clk);
154     end
155     SS_n = 1;
156     @(negedge clk);
157     // Check for Read Address
158     if (DUT.RAM_inst.Rd_Addr == rd_add)
159         $display("Read Address is Done Correctly");
160     else begin
161         $display("Error in Read Address");
162         $stop;
163     end
164     @(negedge clk);

```

//read data operation test:

Reggit

```
166     $display("Read Data Operation");
167     SS_n = 0;
168     @(negedge clk);
169     MOSI = 1;
170     repeat(3) @(negedge clk);
171
172     repeat(8) begin
173         MOSI = $random; //----->dummy values
174         @(negedge clk);
175     end
176     @(negedge clk);
177
178     repeat(8) begin
179         @(negedge clk);
180         rd_data = {rd_data[6:0],MISO};
181     end
182     SS_n = 1;
183     @(negedge clk);
184     // Check for Read Addresss
185     if (DUT.RAM_inst.RAM[DUT.RAM_inst.Rd_Addr] == rd_data)
186         $display("Read Data is Done Correctly");
187     else begin
188         $display("Error in Read Data");
189         $stop;
190     end
191     repeat(2) @(negedge clk);
192
193     $stop;
194 end
```

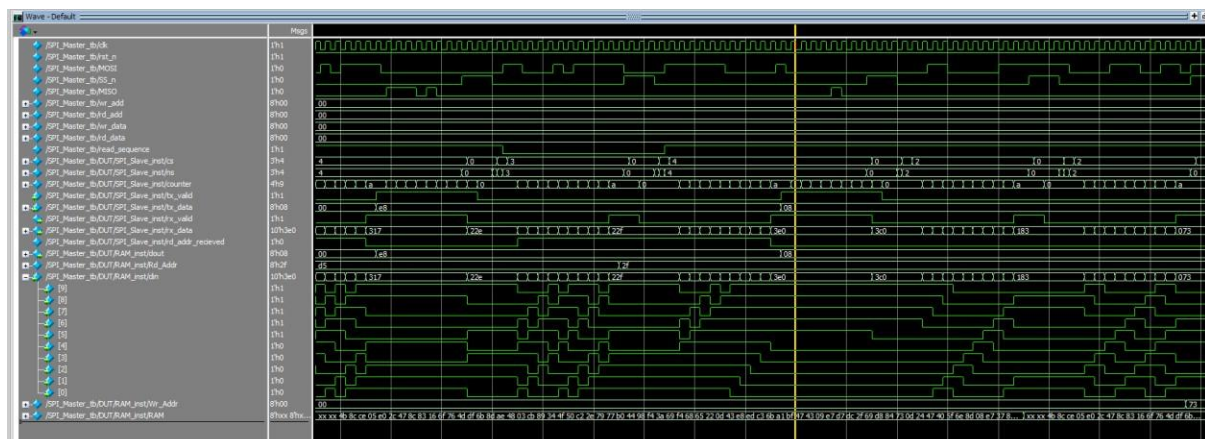
//monitoring

```
195
196 initial
197 $monitor("%MOSI = %b, MISO = %b, SS_n = %b, clk = %b, rst_n = %b, rx_data = %d, rx_valid = %b, tx_data = %d, tx_valid = %b, TIME= %t",
198         MOSI,MISO,SS_n,clk,rst_n,DUT.rx_data_din,DUT.rx_valid,DUT.tx_data_dout,DUT.tx_valid, $time);
199
200 endmodule
```

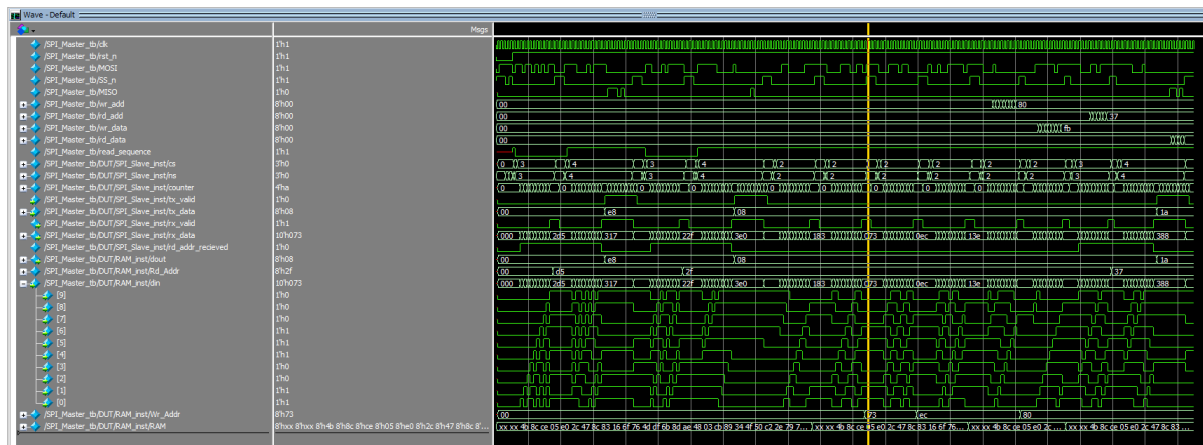
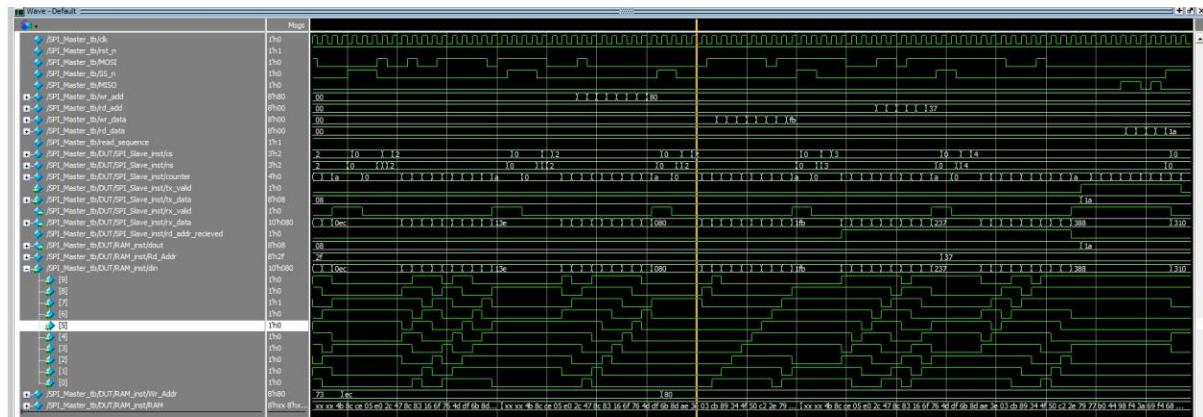
Third: DO file

```
E: > Digital Design course > project_2 > codes > spi.do.txt
1  vlib work
2  vlog SPI_Slave_Interface.v Single_Port_Async_RAM.v SPI_Wrapper.v SPI_Master_tb.v
3  vsim -voptargs=+acc work.SPI_Master_tb
4  add wave *
5  add wave -position insertpoint \
6  sim:/SPI_Master_tb/DUT/SPI_Slave_inst/cs \
7  sim:/SPI_Master_tb/DUT/SPI_Slave_inst/ns \
8  sim:/SPI_Master_tb/DUT/SPI_Slave_inst/counter \
9  sim:/SPI_Master_tb/DUT/SPI_Slave_inst/tx_valid \
10 sim:/SPI_Master_tb/DUT/SPI_Slave_inst/tx_data \
11 sim:/SPI_Master_tb/DUT/SPI_Slave_inst/rx_valid \
12 sim:/SPI_Master_tb/DUT/SPI_Slave_inst/rx_data \
13 sim:/SPI_Master_tb/DUT/SPI_Slave_inst/rd_addr_recieved \
14 sim:/SPI_Master_tb/DUT/RAM_inst/dout \
15 sim:/SPI_Master_tb/DUT/RAM_inst/Rd_Addr \
16 sim:/SPI_Master_tb/DUT/RAM_inst/din \
17 sim:/SPI_Master_tb/DUT/RAM_inst/Wr_Addr \
18 sim:/SPI_Master_tb/DUT/RAM_inst/RAM
19 run -all
```

Fourth: Questa snippets



Reggit



#	Test Reset Operation	
#	MOSI = 0, MISO = x, SS_n = 1, clk = 0, rst_n = 0, rx_data = x, rx_valid = x, tx_data = x, tx_valid = x, TIME=	0
#	MOSI = 0, MISO = 0, SS_n = 1, clk = 1, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	5
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 0, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	10
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 1, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	15
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 0, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	20
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 1, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	25
#	MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	30
#	MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	35
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 0, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	40
#	MOSI = 1, MISO = 0, SS_n = 1, clk = 1, rst_n = 0, rx_data = 0, rx_valid = 0, tx_data = 0, tx_valid = 0, TIME=	45

[illegible]

Reggit

[illegible]

[illegible]

Reggit

[illegible]

```
# Read Data Operation
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 567, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1910
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 567, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1915
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 567, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1920
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 567, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1925
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 567, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1930
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 111, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1935
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 111, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1940
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 223, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1945
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 223, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1950
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 447, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1955
# MOSI = 1, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 447, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1960
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 894, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1965
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 894, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1970
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 764, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1975
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 764, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1980
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 504, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1985
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 504, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1990
# MOSI = 1, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 1009, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 1995
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 1009, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 2000
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 994, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 2005
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 994, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 2010
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 964, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 2015
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 964, rx_valid = 0, tx_data = 8, tx_valid = 0, TIME= 2020
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 8, tx_valid = 0, TIME= 2025
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 8, tx_valid = 0, TIME= 2030
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2035
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2040
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2045
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2050
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2055
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2060
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2065
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2070
# MOSI = 0, MISO = 1, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2075
# MOSI = 0, MISO = 1, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2080
# MOSI = 0, MISO = 1, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2085
# MOSI = 0, MISO = 1, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2090
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2095
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2100
# MOSI = 0, MISO = 1, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2105
# MOSI = 0, MISO = 1, SS_n = 0, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2110
# MOSI = 0, MISO = 0, SS_n = 0, clk = 1, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2115
# MOSI = 0, MISO = 0, SS_n = 1, clk = 0, rst_n = 1, rx_data = 904, rx_valid = 1, tx_data = 26, tx_valid = 1, TIME= 2120
# MOSI = 0, MISO = 0, SS_n = 1, clk = 1, rst_n = 1, rx_data = 784, rx_valid = 0, tx_data = 26, tx_valid = 1, TIME= 2125
# Read Data is Done Correctly
# MOSI = 0, MISO = 0, SS_n = 1, clk = 0, rst_n = 1, rx_data = 784, rx_valid = 0, tx_data = 26, tx_valid = 1, TIME= 2130
# MOSI = 0, MISO = 0, SS_n = 1, clk = 1, rst_n = 1, rx_data = 784, rx_valid = 0, tx_data = 26, tx_valid = 0, TIME= 2135
# MOSI = 0, MISO = 0, SS_n = 0, clk = 0, rst_n = 1, rx_data = 784, rx_valid = 0, tx_data = 26, tx_valid = 0, TIME= 2140
# MOSI = 0, MISO = 0, SS_n = 1, clk = 1, rst_n = 1, rx_data = 784, rx_valid = 0, tx_data = 26, tx_valid = 0, TIME= 2145
# ** Note: $stop : SPI_Master_tb.v(197)
# Time: 2150 ns Iteration: 1 Instance: /SPI_Master_tb
```

Fifth: Constraints file

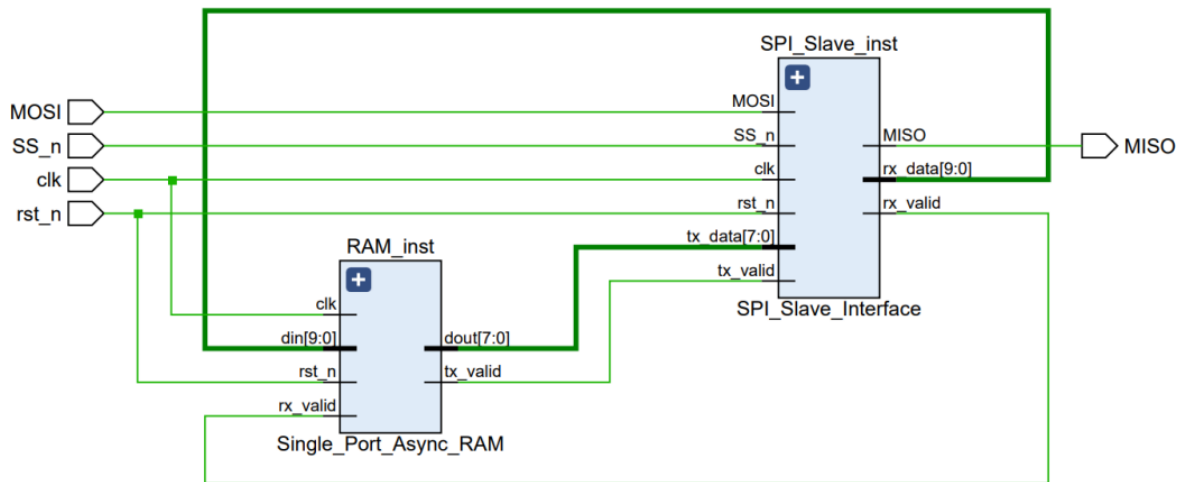
```
E> Digital Design course > project_2 > codes > basys_master.xdc
1  ## This file is a general .xdc for the Basys3 rev B board
2  ## To use it in a project:
3  ## - uncomment the lines corresponding to used pins
4  ## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
5
6  ## Clock signal
7  set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
8  create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
9
10 ## Switches
11 set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports rst_n]
12 set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports SS_n]
13 set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports MOSI]
14
15 ## LEDs
16 set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports MISO]
17
18 ## Configuration options, can be used for all designs
19 set_property CONFIG_VOLTAGE 3.3 [current_design]
20 set_property CFGBVS VCC0 [current_design]
21
22 ## SPI configuration mode options for QSPI boot, can be used for all designs
23 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
24 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
25 set_property CONFIG_MODE SPIx4 [current_design]
```

Reggit

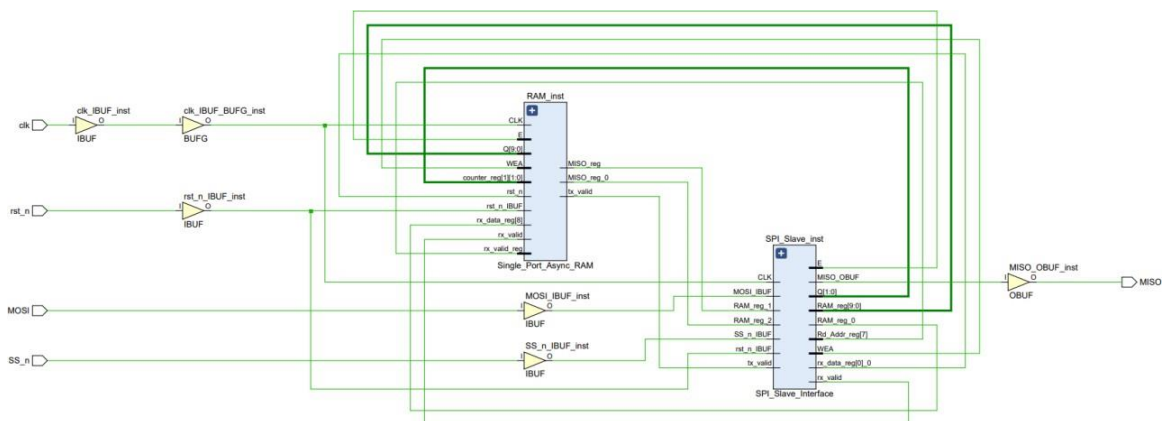
//Three FSM encoding (Gray, One hot, Seq)

Sixth: Gray encoding

Schematic after elaboration:



Schematic after synthesis:



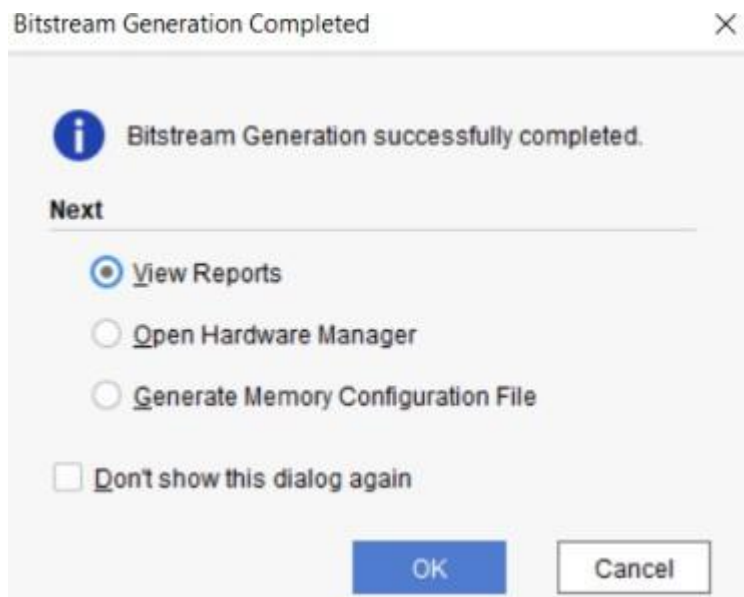
Reggit

Utilization:

Summary

Resource	Utilization	Available	Utilization %
LUT	26	20800	0.13
FF	37	41600	0.09
BRAM	0.50	50	1.00
IO	5	106	4.72

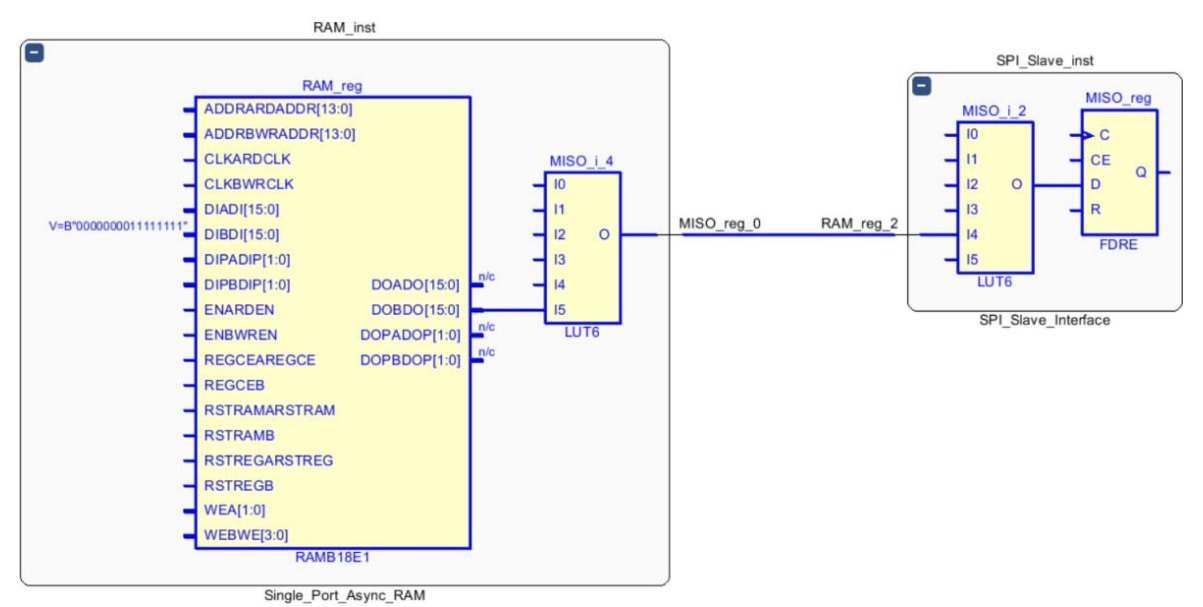
Bitstream:



Timing summary:

Design Timing Summary			
Setup		Hold	Pulse Width
Worst Negative Slack (WNS):		Worst Hold Slack (WHS):	Worst Pulse Width Slack (WPWS):
Total Negative Slack (TNS):		Total Hold Slack (THS):	Total Pulse Width Negative Slack (TPWS):
Number of Failing Endpoints:		Number of Failing Endpoints:	Number of Failing Endpoints:
Total Number of Endpoints:		Total Number of Endpoints:	Total Number of Endpoints:

Critical path:

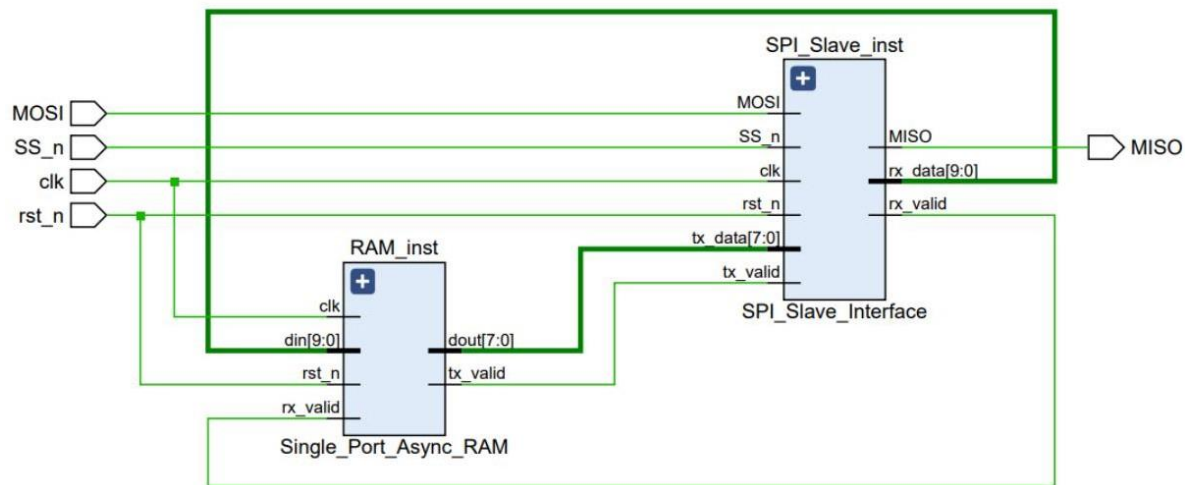


FSM encoding:

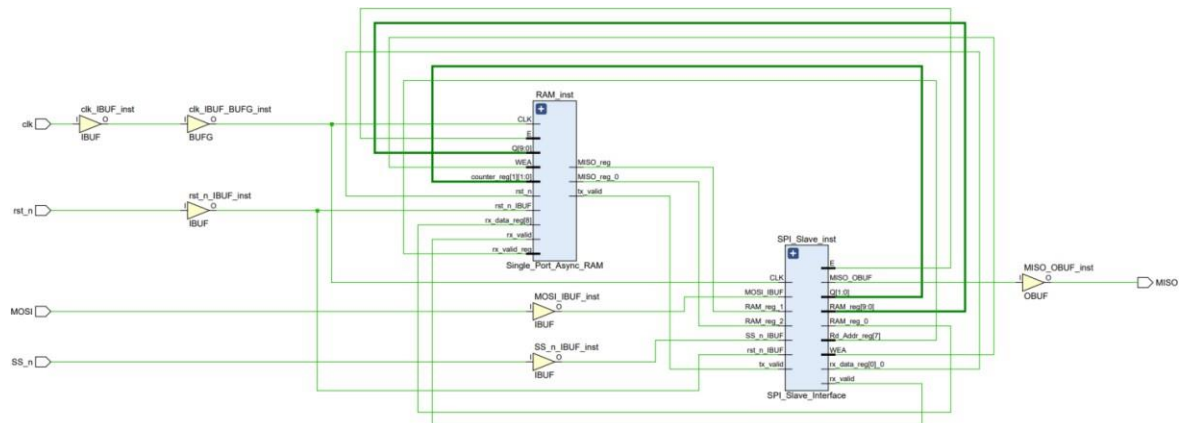
State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	011	010
READ_ADD	010	011
READ_DATA	111	100

Seventh: One Hot encoding

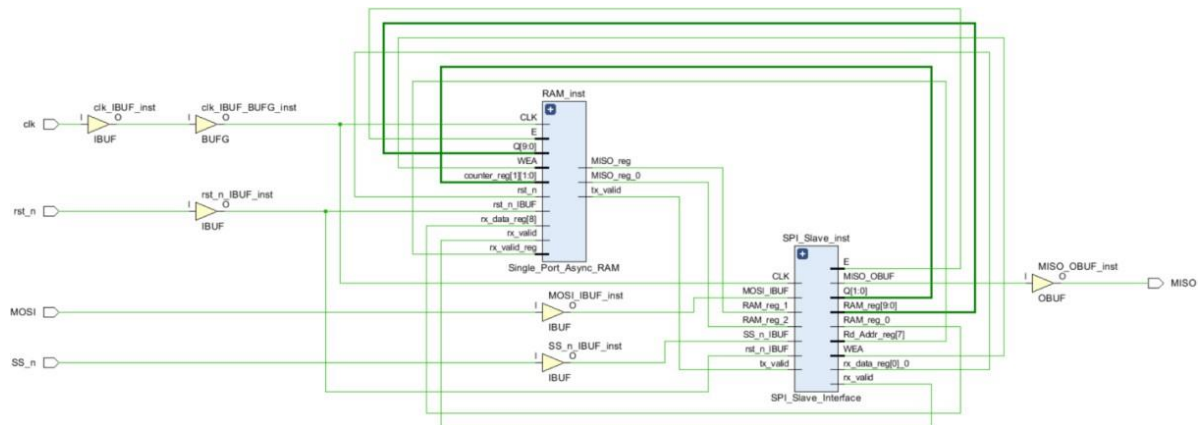
Schematic after elaboration:



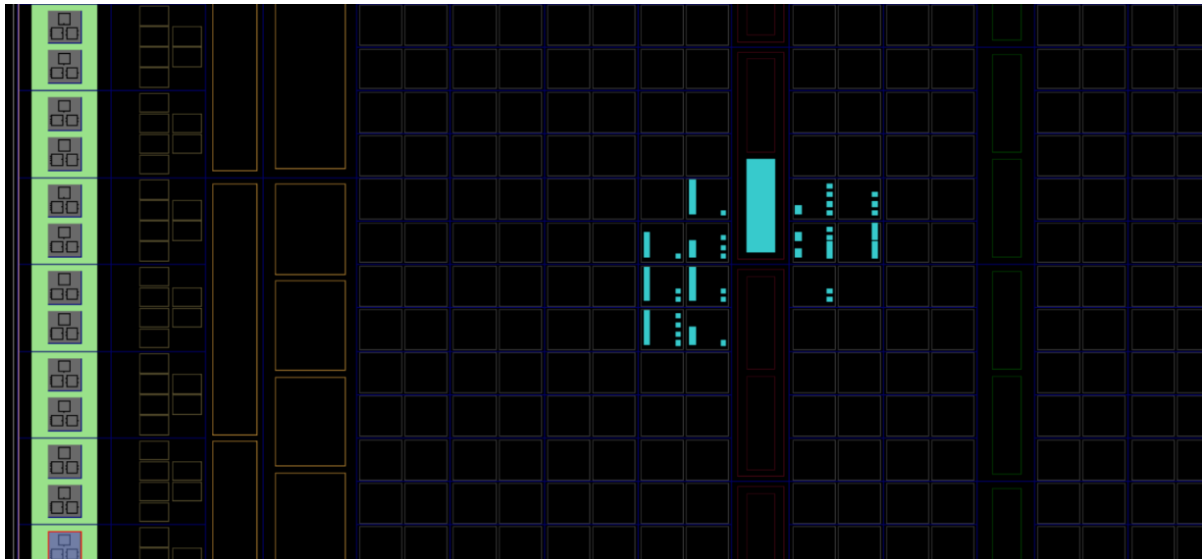
Schematic after Synthesis:



Schematic after Implementation:



Device after Implementation:



Messages tab:

Tcl Console | Log | Reports | Design Runs | Debug | **Messages** x

Search: [] Filter: [] Critical warning (2) Warning (10) Info (289) Status (585) Show All

- Synthesis (1 warning)
 - [Constraints 19-52:10] No constraint will be written out.
- Synthesized Design (1 warning)
 - General Messages (1 warning)
 - [Project 1-478] Design 'design_1' is stale and will not be used when launching 'impl_1'
- Implementation (1 critical warning, 4 warnings)
 - Route Design (1 critical warning, 3 warnings)
 - [Route 35-455] Router was unable to fix hold violation on 1 pins. This could be due to a combination of congestion, blockages and run-time limitations. Such pins are: u_ila_0instila_core_instlv_trigU_TMN_DDR_MODE_G_NMU[0]U_M/allx_typeA_match_detection[lib_v1_0_0_allx_typeA_instprobeDelay10]_L_1/2
 - Resolution: You may try high effort hold fixing by turning on param route.enableGlobalHoldtler.
 - [Route 35-328] Router estimated timing not met.
 - Resolution: For a complete and accurate timing signoff, report_timing_summary must be run after route_design. Alternatively, route_design can be run with the -timing_summary option to enable a complete timing signoff at the end of route_design.
 - [Timing 38-282] The design failed to meet the timing requirements. Please see the timing summary report for details on the timing violations.
 - [Timing 38-436] There are set_bus_skew constraint(s) in this design. Please run report_bus_skew to ensure that bus skew requirements are met.
 - Write Bitstream (1 warning)

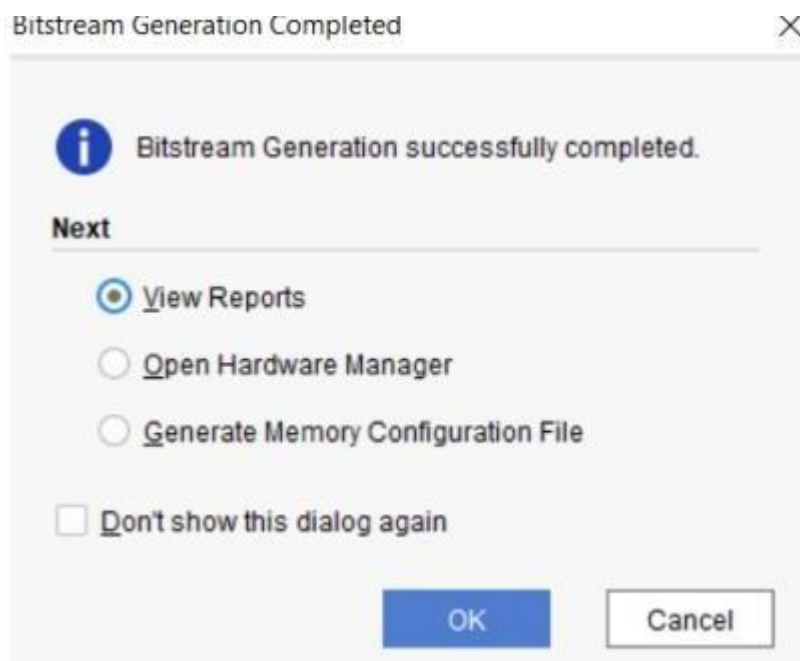
Reggit

Utilization:

Summary

Resource	Utilization	Available	Utilization %
LUT	26	20800	0.13
FF	37	41600	0.09
BRAM	0.50	50	1.00
IO	5	106	4.72

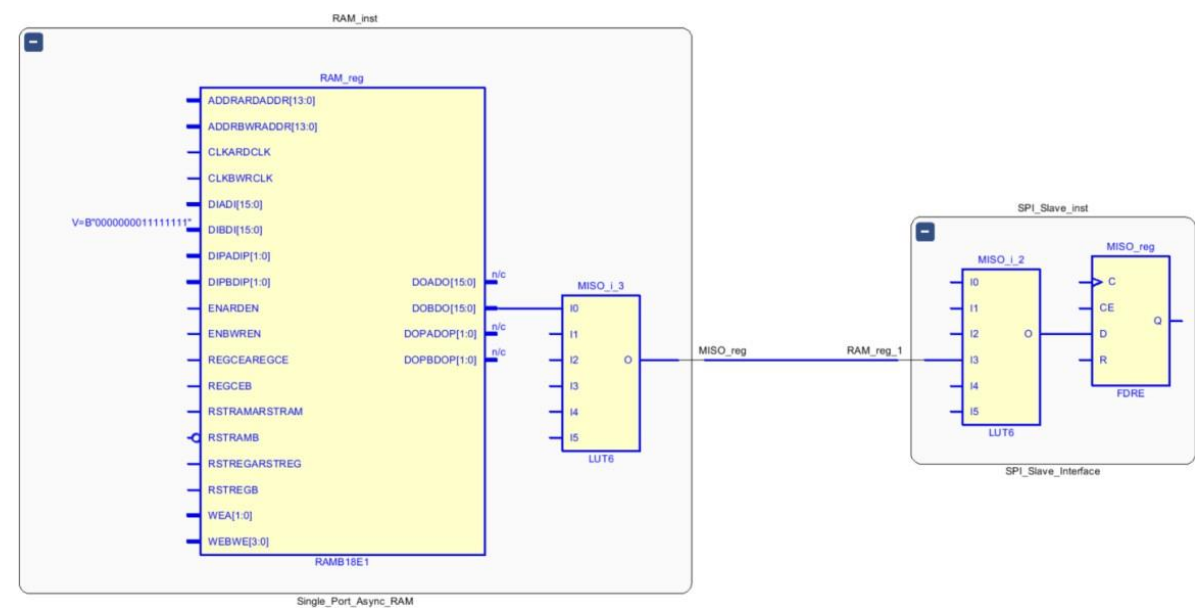
Bitstream:



Timing Summary:

Design Timing Summary		
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.352 ns	Worst Hold Slack (WHS): 0.073 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: 113	Total Number of Endpoints: 113	Total Number of Endpoints: 52

Critical Path:

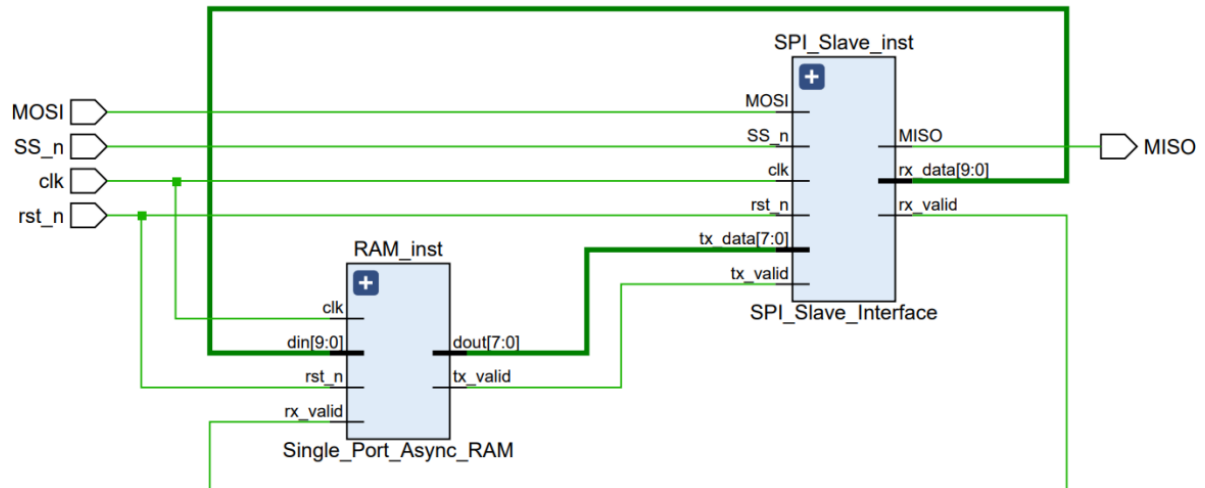


FSM encoding:

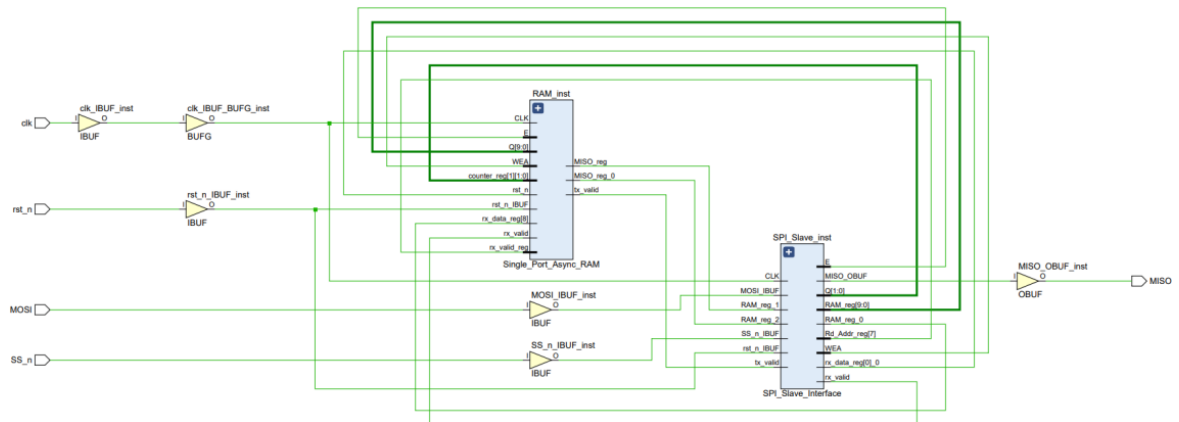
State	New Encoding	Previous Encoding
IDLE	00001	000
CHK_CMD	00010	001
WRITE	00100	010
READ_ADD	01000	011
READ_DATA	10000	100

Eighth: SEQ encoding

Schematic after elaboration:

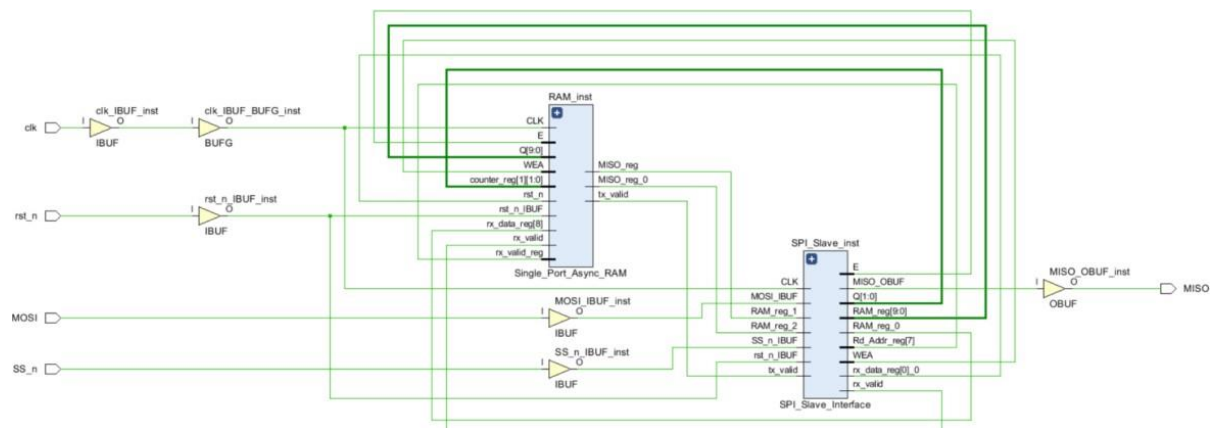


Schematic after Synthesis:

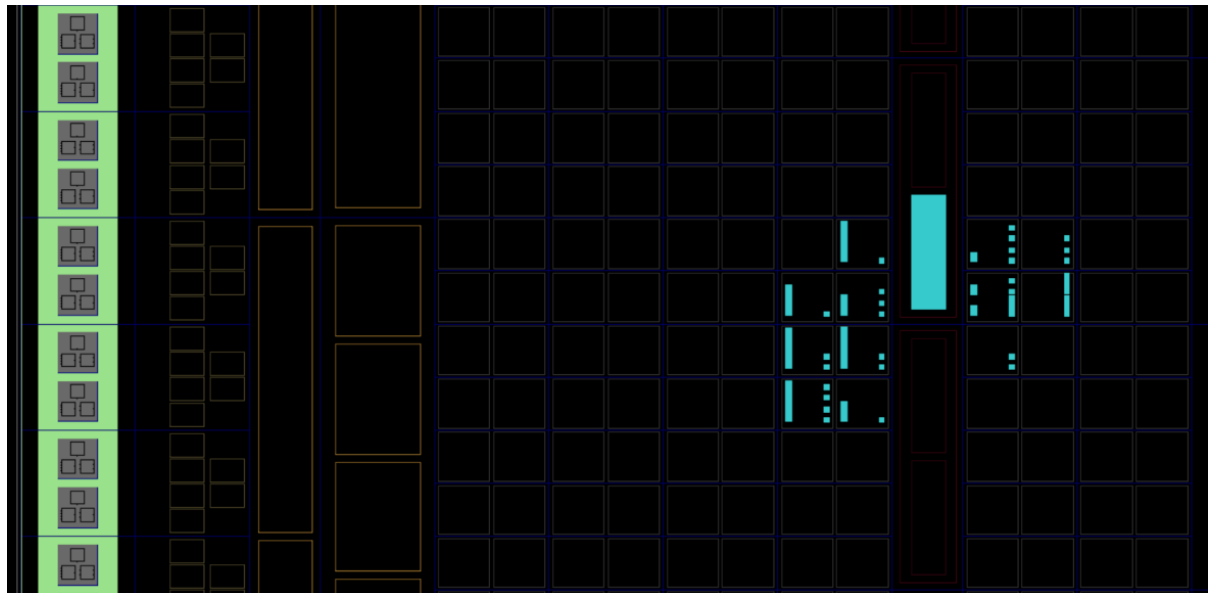


Reggit

Schematic after Implementation:



Device after Implementation:



Messages tab:

Tcl Console
Messages
Log
Reports
Design Runs
Debug

Critical warning (2)
 Warning (10)
 Info (290)
 Status (584)
Show All

Synthesis (1 warning)

- [Constraints 18-5210] No constraint will be written out.

Synthesized Design (1 warning)

- General Messages (1 warning)
 - [Project 1-478] Design 'design_tf' is stale and will not be used when launching 'impl_tf'

Implementation (1 critical warning, 4 warnings)

- Route Design (1 critical warning, 3 warnings)
 - [Route 35-459] Router was unable to fix hold violation on 1 pins. This could be due to a combination of congestion, blockages and run-time limitations. Such pins are: u_ila_0instlla_core_instsig_trigU_TIMN_DDR_MODE_G_NMU0[U_M'alix_typeA_match_detection.libb_v1_0_0_alix_typeA_inst]probeDelay[0]_I12

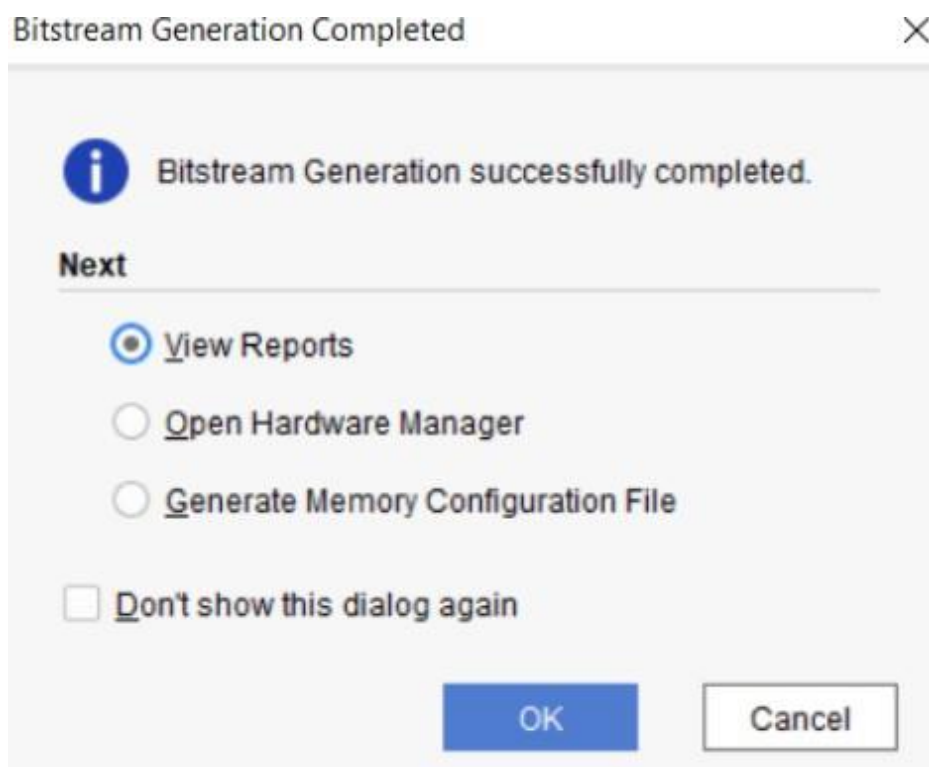
Resolution: You may try high effort hold fixing by turning on param route.enableGlobalHoldfilter.

Utilization:

Summary

Resource	Utilization	Available	Utilization %
LUT	26	20800	0.13
FF	37	41600	0.09
BRAM	0.50	50	1.00
IO	5	106	4.72

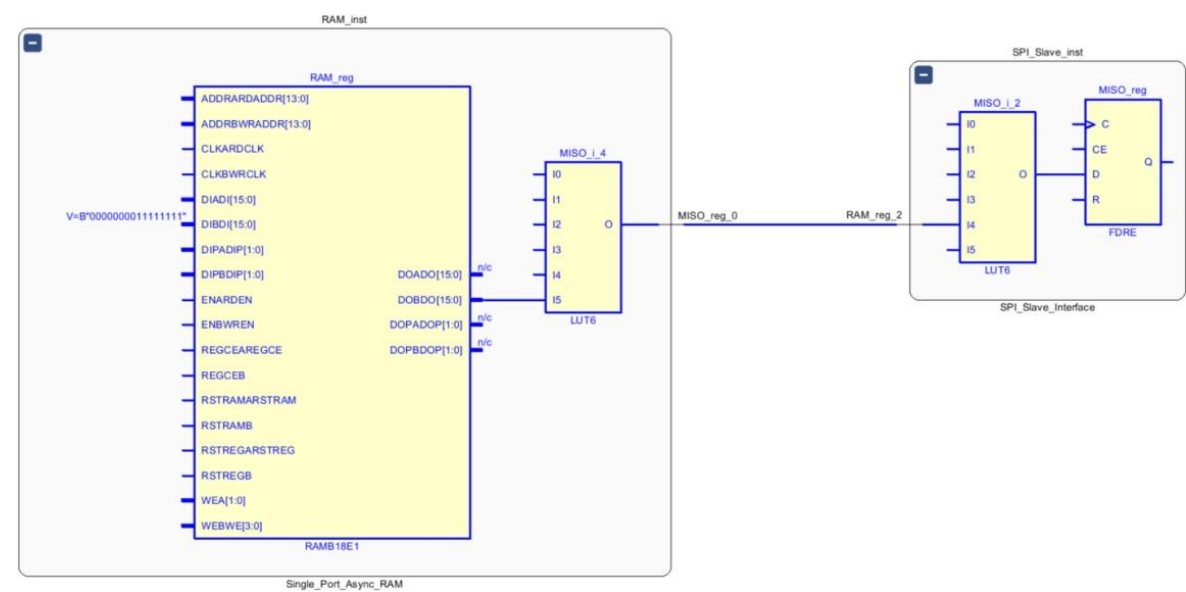
Bitstream:



Timing Summary:

Design Timing Summary		
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 4.895 ns	Worst Hold Slack (WHS): 0.067 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: 111	Total Number of Endpoints: 111	Total Number of Endpoints: 50

Critical path:



FSM encoding:

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

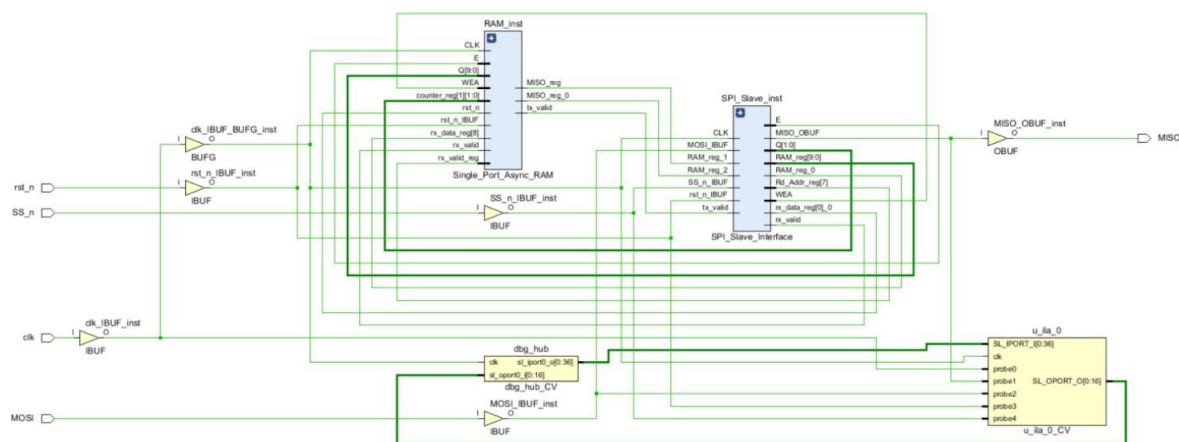
Ninth: best encoding

//Specs wise → it is required to operate at the highest frequency, then it is obvious that the //GRAY encoding is relatively the best as it has the highest setup slack time = 5.490 ns, which //was shown in the GRAY FSM encoding timing summary.

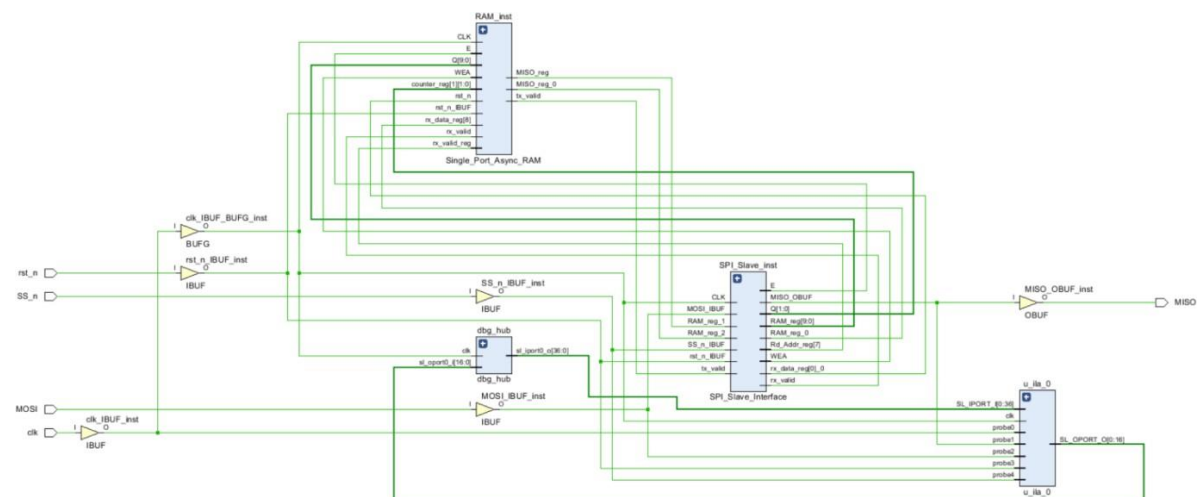
//Then, Gray encoding is chosen.

//Debug core is added in the following schematics.

Synthesis Schematic:

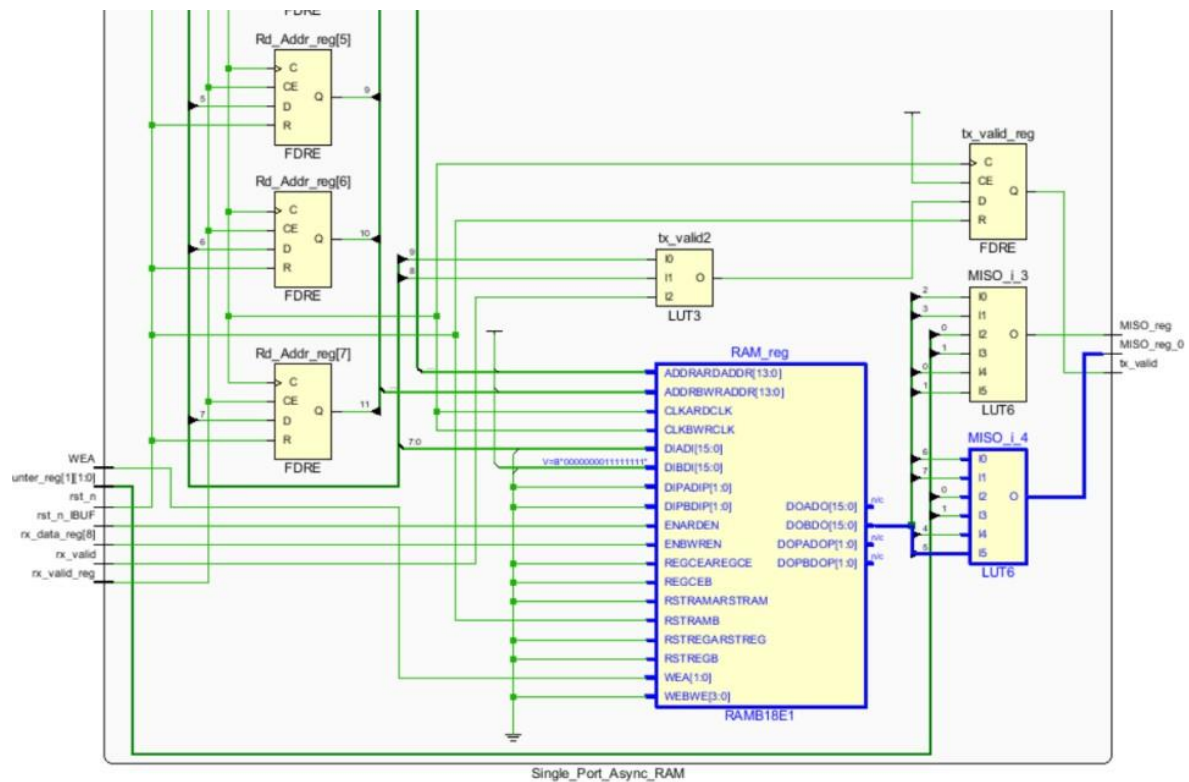
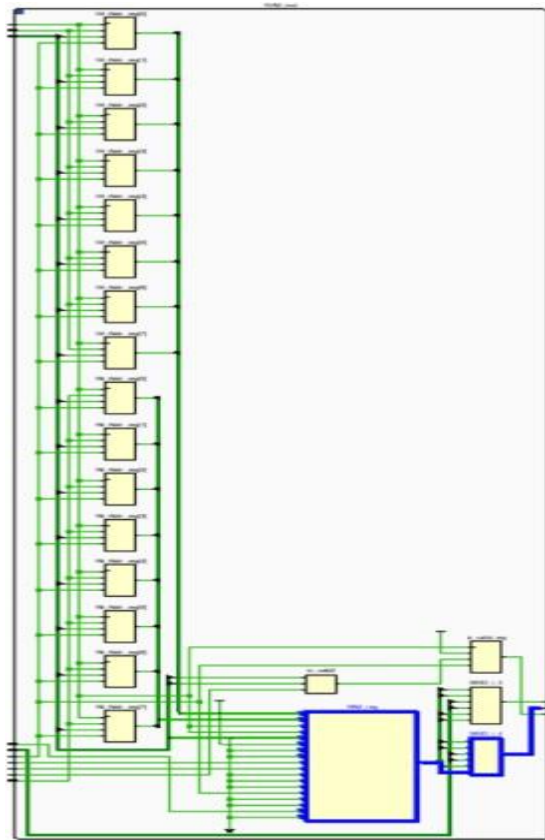


Implementation Schematic:



Tenth: Critical path

RAM:



Only RAMS

SPI:

