يوسف أحمد محمد ابراهيم

Project 2

SPI slave interface

- **RTL code :**
  - top module :

```
1   module SPI_slave_interface(
2       input   wire        MOSI,
3       input   wire        SS_n,
4       input   wire        clk,
5       input   wire        rst_n,
6
7       output  wire         MISO
8   );
9       wire [9:0] din_rx_data;
10      wire       rx_valid;
11      wire [7:0] tx_data_dout;
12      wire       tx_valid;
13
14      RAM ram(din_rx_data, clk, rst_n, rx_valid,
15          tx_data_dout, tx_valid);
16      slave slave(MOSI, SS_n, clk, rst_n, tx_valid,
17          tx_data_dout, MISO, rx_valid, din_rx_data);
18  endmodule
```

- **RTL code :**
  - slave

```
22  module slave(
23      input   wire        MOSI,
24      input   wire        SS_n,
25      input   wire        clk,
26      input   wire        rst_n,
27      input   wire        tx_valid,
28      input   wire [7:0]  tx_data,
29
30      output  reg         MISO,
31      output  reg         rx_valid,
32      output  reg  [9:0]  rx_data
33  );
34
35      localparam IDLE         = 3'b000;
36      localparam CHK_CMD      = 3'b001;
37      localparam WRITE        = 3'b010;
38      localparam READ_ADD     = 3'b011;
39      localparam READ_DATA    = 3'b100;
40
41      reg [2:0]   cs, ns;
42      reg [3:0]   count;
43      reg [9:0]   parallel_data;
44      reg         add_data;
45      reg [7:0]   received_data;
46      reg         send_MISO;
```

- **RTL code :**
  - slave

```
48      //output logic
49      always @(posedge clk, negedge rst_n) begin
50          if(~rst_n) begin
51              add_data <= 0;
52              MISO <= 0;
53              rx_valid <= 0;
54              rx_data <= 0;
55              count <= 0;
56              parallel_data <= 0;
57              received_data <= 0;
58              send_MISO <= 0;
59          end
60          else begin
61              case (cs)
62                  IDLE: begin
63                      MISO <= 0;
64                      rx_valid <= 0;
65                      rx_data <= 0;
66                      count <= 0;
67                      parallel_data <= 0;
68                      received_data <= 0;
69                      send_MISO <= 0;
70                  end
71                  WRITE: begin
72                      if(count > 9) begin
73                          rx_data <= parallel_data;
74                          rx_valid <= 1;
75                      end
76                      else begin
77                          parallel_data[9:1] <= parallel_data[8:0];
78                          parallel_data[0] <= MOSI;
79                          count <= count + 1;
80                      end
81                  end
```

- **RTL code :**
  - slave

```
82                  READ_ADD: begin
83                      if(count > 9) begin
84                          rx_data <= parallel_data;
85                          rx_valid <= 1;
86                          add_data <= 1;
87                      end
88                      else begin
89                          parallel_data[9:1] <= parallel_data[8:0];
90                          parallel_data[0] <= MOSI;
91                          count <= count + 1;
92                      end
93                  end
94                  READ_DATA: begin
95                      if (send_MISO && count < 8)
96                          {MISO, received_data} <= {received_data, 1'b0};
97                      else if (tx_valid) begin
98                          received_data <= tx_data;
99                          count <= 0;
100                         send_MISO <= 1;
101                     end
102                     else if(count > 9) begin
103                         rx_data <= parallel_data;
104                         rx_valid <= 1;
105                         add_data <= 0;
106                     end
107                     else if (count < 10) begin
108                         parallel_data[0] <= MOSI;
109                         parallel_data[9:1] <= parallel_data[8:0];
110                         count <= count + 1;
111                     end
112                 end
113             endcase
114         end
115     end
```

- **RTL code :**
  - slave

```
117     //next state logic
118     always @(*) begin
119         case (cs)
120             IDLE : begin
121                 if(SS_n)
122                     ns = IDLE;
123                 else
124                     ns = CHK_CMD;
125             end
126             CHK_CMD : begin
127                 if(SS_n)
128                     ns = IDLE;
129                 else begin
130                     if(MOSI)
131                         if(add_data)
132                             ns = READ_DATA;
133                         else
134                             ns = READ_ADD;
135                     else
136                         ns = WRITE;
137                 end
138             end
139             WRITE: begin
140                 if (SS_n)
141                     ns = IDLE;
142                 else
143                     ns = WRITE;
144             end
```

```
139             WRITE: begin
140                 if (SS_n)
141                     ns = IDLE;
142                 else
143                     ns = WRITE;
144             end
145             READ_ADD: begin
146                 if (SS_n)
147                     ns = IDLE;
148                 else
149                     ns = READ_ADD;
150             end
151             READ_DATA: begin
152                 if (SS_n)
153                     ns = IDLE;
154                 else
155                     ns = READ_DATA;
156             end
157             default: ns = IDLE;
158         endcase
159     end
```

```
//state memory
always @(posedge clk, negedge rst_n) begin
    if(~rst_n)
        cs <= IDLE;
    else
        cs <= ns;
end
```

- **RTL code :**
  - memory

```
172  module RAM #(
173      parameter MEM_DEPTH = 256,
174      parameter ADDR_SIZE = 8
175  )(
176      input       [9:0]   din,
177      input               clk,
178      input               rst_n,
179      input               rx_valid,
180      output reg  [7:0]   dout,
181      output reg          tx_valid
182  );
183      reg [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
184
185      reg [ADDR_SIZE-1:0] write_address;
186      reg [ADDR_SIZE-1:0] read_address;
187
188      always @(posedge clk, negedge rst_n) begin
189          if(~rst_n) begin
190              dout            <= 0;
191              tx_valid        <= 0;
192              write_address   <= 0;
193              read_address    <= 0;
194          end
195          else begin
196              tx_valid <= 0;
197              if(rx_valid) begin
198                  case (din[9:8])
199                      2'b00 : write_address <= din[7:0];
200                      2'b01 : mem[write_address] <= din [7:0];
201                      2'b10 : read_address <= din[7:0];
202                      2'b11 : begin
203                          tx_valid <= 1;
204                          dout <= mem[read_address];
205                      end
206                  endcase
207              end
208          end
209      end
210  endmodule
```

- **Testbench code:**

```
module SPI_slave_interface_tb();
    reg         MOSI;
    reg         SS_n;
    reg         clk;
    reg         rst_n;
    wire        MISO;
    reg         MISO_exp;

    reg [9:0]   MOSI_data;

    SPI_slave_interface dut(MOSI, SS_n, clk, rst_n, MISO);

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end
    integer i;
    initial begin
        //check rest functionality
        $readmemb ("mem.dat", dut.ram.mem);
        rst_n = 0;
        SS_n = 1;
        MISO_exp = 0;
        MOSI = 0;
        MOSI_data = 0;
        repeat(5)@ (negedge clk);
        if (MISO != MISO_exp) begin
            $display("*** RESET CHECK FAILS! ***");
        end
        $display("*** RESET FUNCTION TEST PASS! ***");
        rst_n = 1;
        $stop;
```

- **Testbench code:**

```
35          //check write fucntionality
36          SS_n = 0;
37          MOSI = 0;
38          @(negedge clk);
39          @(negedge clk);
40          MOSI_data = 10'b0000000011;
41          for (i=0; i<10; i = i+1) begin
42              MOSI = MOSI_data[9 - i];
43              @(negedge clk);
44          end
45          SS_n = 1;
46          @(negedge clk);
47          MOSI = 0;
48          SS_n = 0;
49          @(negedge clk);
50          @(negedge clk);
51          MOSI_data = 10'b0101010101;
52          for(i = 0; i<10; i = i+ 1) begin
53              MOSI = MOSI_data[9 - i];
54              @(negedge clk);
55          end
56          SS_n = 1;
57          @(negedge clk);
58          @(negedge clk);
59          if (dut.ram.mem[3] != 10'b01010101) begin
60              $display("*** WRITE CHECK FAILS! ***");
61              $stop;
62          end
63          $display("*** WRITE CHECK PASS! ***");
64          $stop;
```

- **Testbench code:**

```
        //check read fucntionality
        SS_n = 0;
        MOSI = 1;
        @(negedge clk);
        @(negedge clk);
        MOSI_data = 10'b1000000011;
        for (i=0; i<10; i = i+1) begin
            MOSI = MOSI_data[9 - i];
            @(negedge clk);
        end
        SS_n = 1;
        @(negedge clk);
        MOSI = 1;
        SS_n = 0;
        @(negedge clk);
        @(negedge clk);
        MOSI_data = 10'b1101010101;
        for(i = 0; i<10; i = i+1) begin
            MOSI = MOSI_data[9 - i];
            @(negedge clk);
        end
        @(negedge clk);
        @(negedge clk);
        repeat(8) begin
            @(negedge clk);
            if(MOSI != MOSI_data[7 - i]) begin
                $display ("*** READ TEST FAILS! ***");
                $display ("\n*** MOSI = %d, expected: %d ***", MOSI, MOSI_data);
                $stop;
            end
        end
        SS_n = 1;
        @(negedge clk);
        @(negedge clk);
        $display("*** READ TEST PASS! ***");
        $stop;
    end

    initial begin
        $monitor("Time=%0t | clk=%b | rst_n=%b | SS_n=%b | MOSI=%b | MISO=%b",
                 $time, clk, rst_n, SS_n, MOSI, MISO);
    end
endmodule
```

- **Do file :**

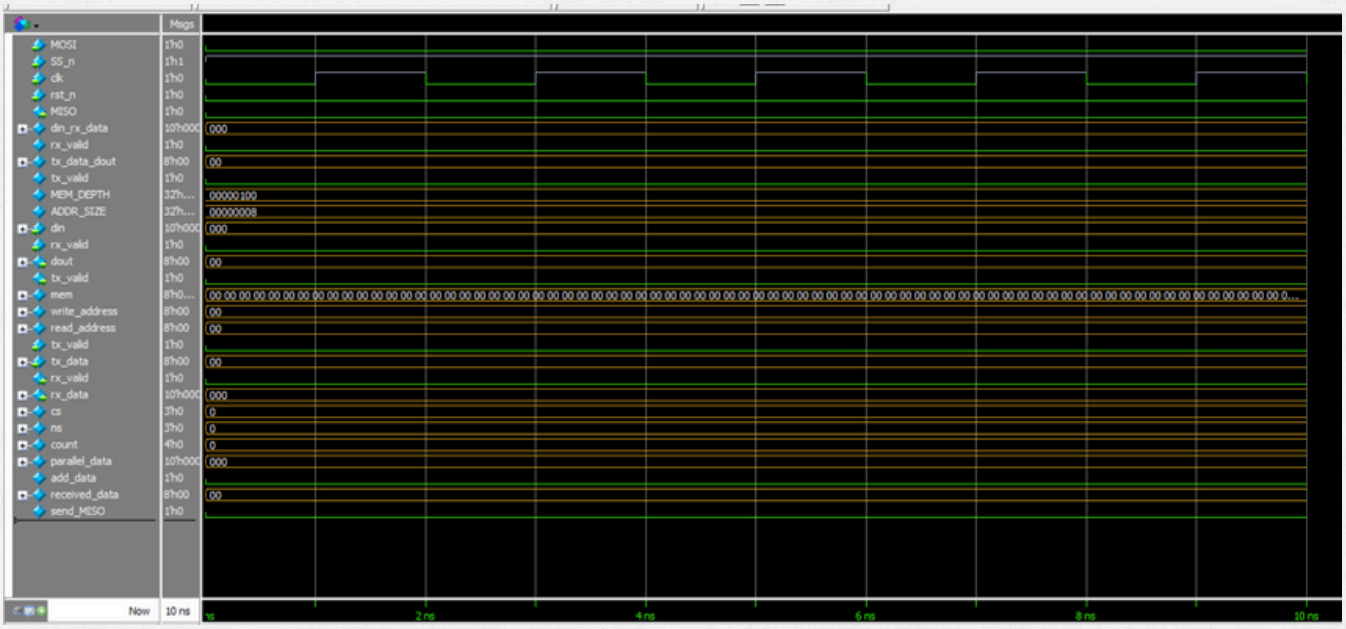
```
vlib work
vlog SPI.v SPI_tb.v
vsim -voptargs=+acc work.SPI_slave_interface_tb
add wave -position insertpoint  \
sim:/SPI_slave_interface_tb/dut/MOSI \
sim:/SPI_slave_interface_tb/dut/SS_n \
sim:/SPI_slave_interface_tb/dut/clk \
sim:/SPI_slave_interface_tb/dut/rst_n \
sim:/SPI_slave_interface_tb/dut/MISO \
sim:/SPI_slave_interface_tb/dut/din_rx_data \
sim:/SPI_slave_interface_tb/dut/rx_valid \
sim:/SPI_slave_interface_tb/dut/tx_data_dout \
sim:/SPI_slave_interface_tb/dut/tx_valid
add wave -position insertpoint  \
sim:/SPI_slave_interface_tb/dut/ram/MEM_DEPTH \
sim:/SPI_slave_interface_tb/dut/ram/ADDR_SIZE \
sim:/SPI_slave_interface_tb/dut/ram/din \
sim:/SPI_slave_interface_tb/dut/ram/clk \
sim:/SPI_slave_interface_tb/dut/ram/rst_n \
sim:/SPI_slave_interface_tb/dut/ram/rx_valid \
sim:/SPI_slave_interface_tb/dut/ram/dout \
sim:/SPI_slave_interface_tb/dut/ram/tx_valid \
sim:/SPI_slave_interface_tb/dut/ram/mem \
sim:/SPI_slave_interface_tb/dut/ram/write_address \
sim:/SPI_slave_interface_tb/dut/ram/read_address
add wave -position insertpoint  \
sim:/SPI_slave_interface_tb/dut/slave/MOSI \
sim:/SPI_slave_interface_tb/dut/slave/SS_n \
sim:/SPI_slave_interface_tb/dut/slave/clk \
sim:/SPI_slave_interface_tb/dut/slave/rst_n \
sim:/SPI_slave_interface_tb/dut/slave/tx_valid \
sim:/SPI_slave_interface_tb/dut/slave/tx_data \
sim:/SPI_slave_interface_tb/dut/slave/MISO \
sim:/SPI_slave_interface_tb/dut/slave/rx_valid \
sim:/SPI_slave_interface_tb/dut/slave/rx_data \
sim:/SPI_slave_interface_tb/dut/slave/cs \
sim:/SPI_slave_interface_tb/dut/slave/ns \
sim:/SPI_slave_interface_tb/dut/slave/count \
sim:/SPI_slave_interface_tb/dut/slave/parallel_data \
sim:/SPI_slave_interface_tb/dut/slave/add_data \
sim:/SPI_slave_interface_tb/dut/slave/received_data \
sim:/SPI_slave_interface_tb/dut/slave/send_MISO

run -all
#quit -sim
```

- **Simulation snippets:**

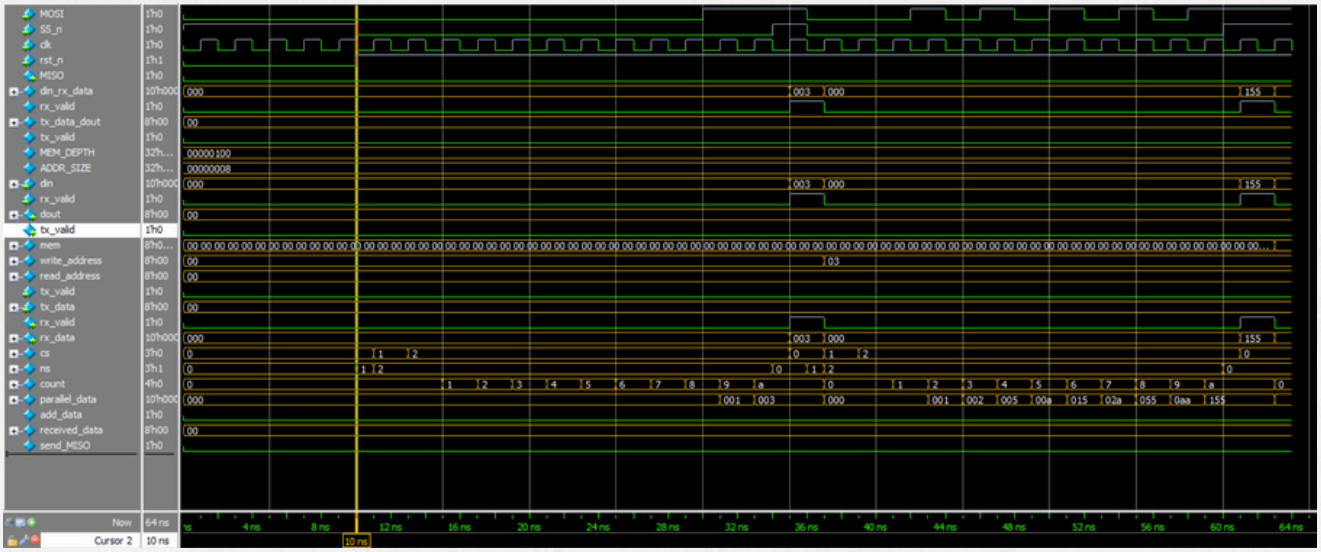
  (questasim)

  - reset fucntion test:




```
# Time=0 | clk=0 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=1 | clk=1 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=2 | clk=0 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=3 | clk=1 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=4 | clk=0 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=5 | clk=1 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=6 | clk=0 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=7 | clk=1 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=8 | clk=0 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# Time=9 | clk=1 | rst_n=0 | SS_n=1 | MOSI=0 | MISO=0
# *** RESET FUNCTION TEST PASS! ***
# ** Note: $stop    : SPI_tb.v(33)
#    Time: 10 ns  Iteration: 1  Instance: /SPI_slave_interface_tb
```

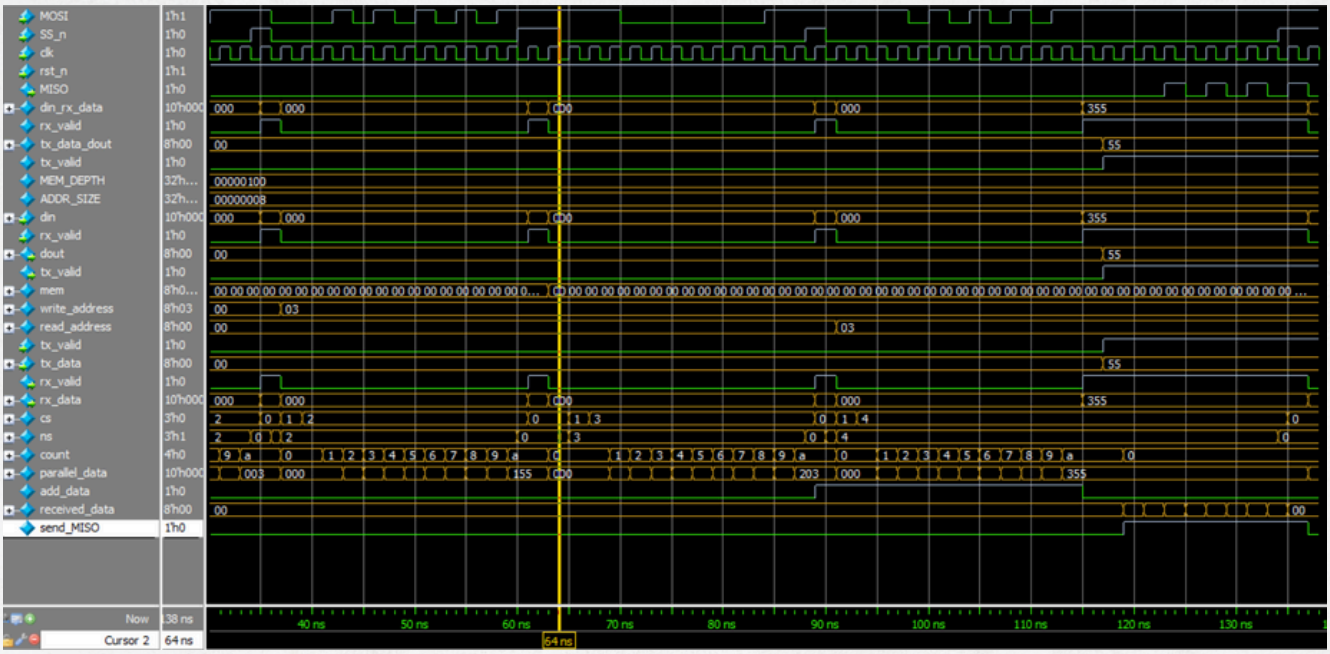
- Write fucntion test:




```
# Time=10 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=11 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=12 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=13 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=14 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=15 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=16 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=17 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=18 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=19 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=20 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=21 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=22 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=23 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=24 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=25 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=26 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=27 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=28 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=29 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=30 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=31 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=32 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=33 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=34 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=35 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=36 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=37 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=38 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=39 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=40 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
```

```
# Time=40 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=41 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=42 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=43 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=44 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=45 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=46 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=47 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=48 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=49 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=50 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=51 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=52 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=53 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=54 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=55 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=56 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=57 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=58 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=59 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=60 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=61 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=62 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=63 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# *** WRITE CHECK PASS! ***
# ** Note: $stop    : SPI_tb.v(64)
#    Time: 64 ns  Iteration: 1  Instance: /SPI_slave_interface_tb
# Break in Module SPI_slave_interface_tb at SPI_tb.v line 64
```

- read fucntion test:




```
# Time=64 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=65 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=66 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=67 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=68 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=69 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=70 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=71 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=72 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=73 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=74 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=75 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=76 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=77 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=78 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=79 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=80 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=81 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=82 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=83 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=84 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=85 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=86 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=87 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=88 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=89 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=90 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=91 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=92 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=93 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=94 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=95 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=96 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=97 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=98 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=99 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
```

```
# Time=100 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=101 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=102 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=103 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=104 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=105 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=106 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=107 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=108 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=109 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=110 | clk=0 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=111 | clk=1 | rst_n=1 | SS_n=0 | MOSI=0 | MISO=0
# Time=112 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=113 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=114 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=115 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=116 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=117 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=118 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=119 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=120 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=121 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=122 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=123 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=124 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=125 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=126 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=127 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=128 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=129 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=130 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=131 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=132 | clk=0 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=1
# Time=133 | clk=1 | rst_n=1 | SS_n=0 | MOSI=1 | MISO=0
# Time=134 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
# Time=135 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=1
# Time=136 | clk=0 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=1
# Time=137 | clk=1 | rst_n=1 | SS_n=1 | MOSI=1 | MISO=0
```

```
# *** READ TEST PASS! ***
# ** Note: $stop    : SPI_tb.v(101)
#    Time: 138 ns  Iteration: 1  Instance: /SPI_slave_interface_tb
# Break in Module SPI_slave_interface_tb at SPI_tb.v line 101
```

- **Constraints file:**

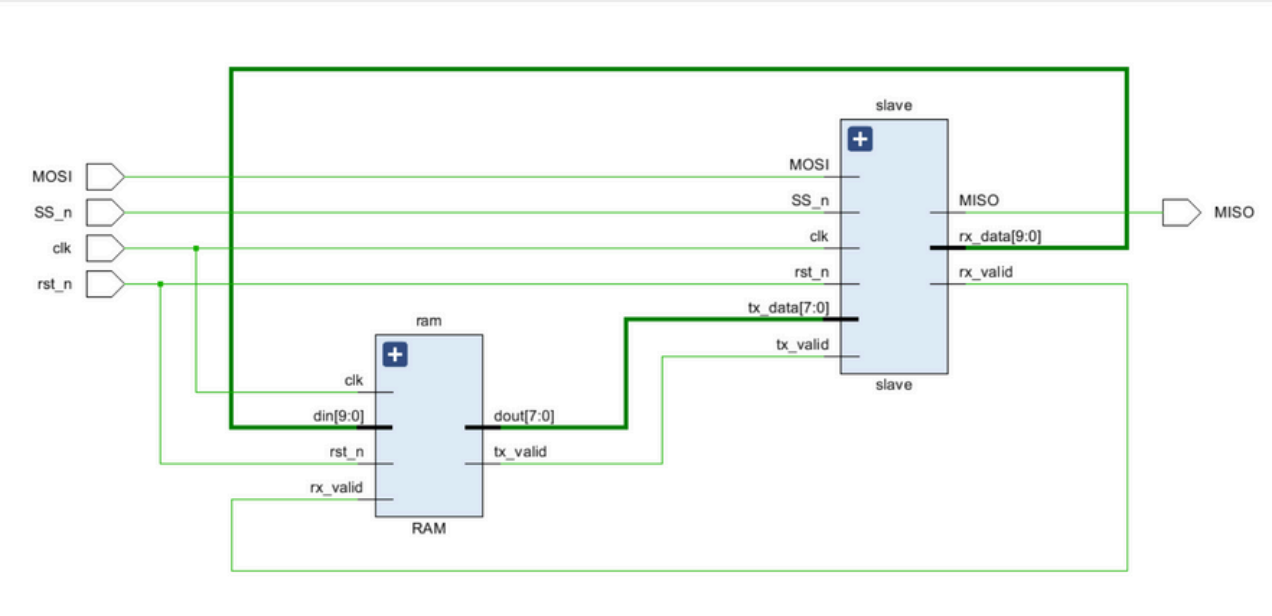
```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {MOSI}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {SS_n}]
```

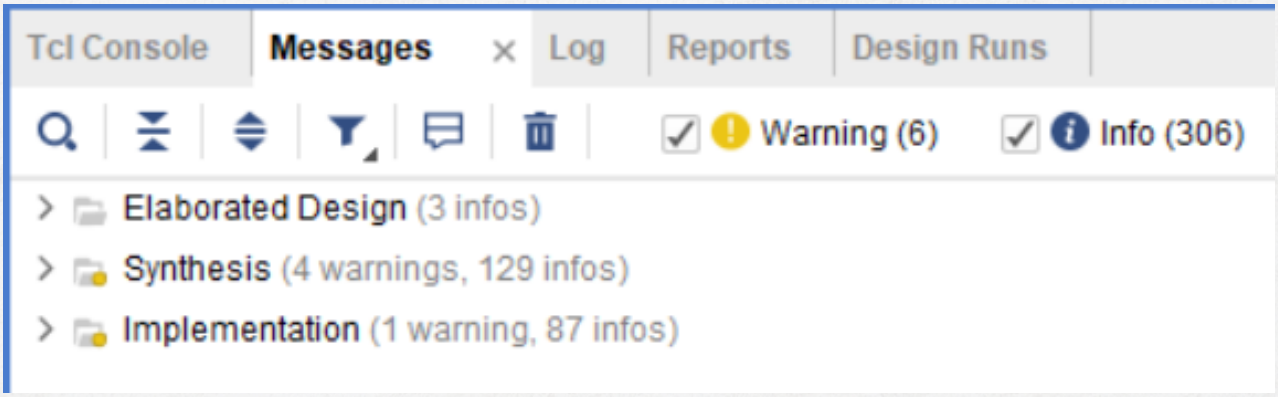
```
## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {MISO}]
```

```
##Buttons
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports rst_n]
```

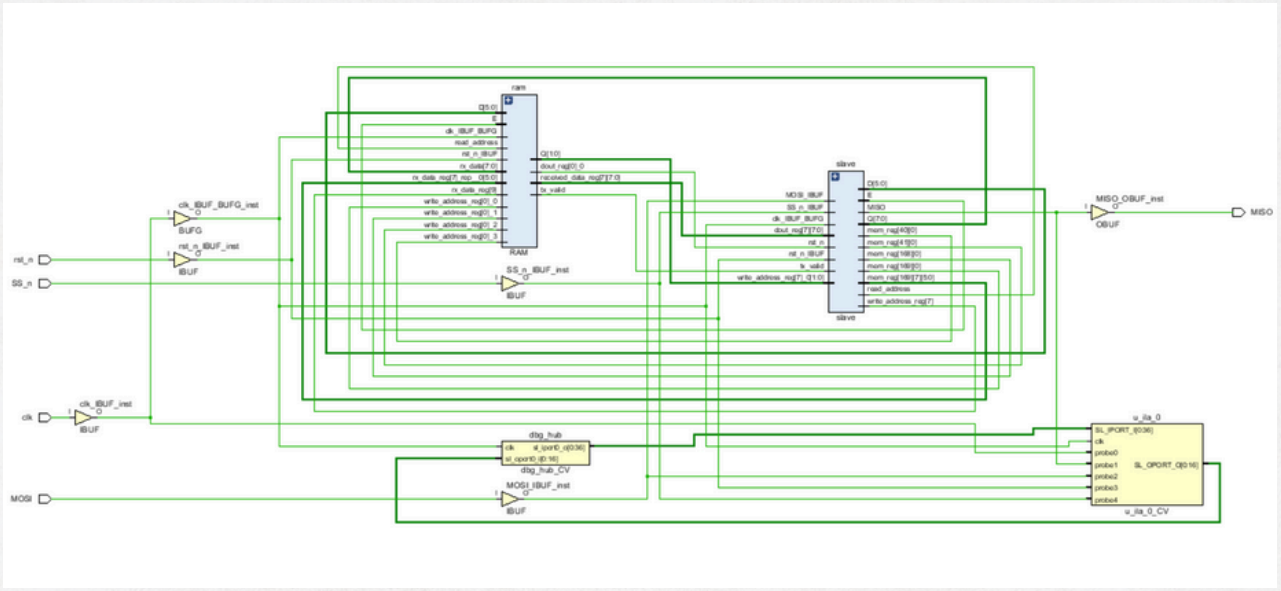
- **Elaporation stage :**
  - schematic :



  - messages (all stages) :



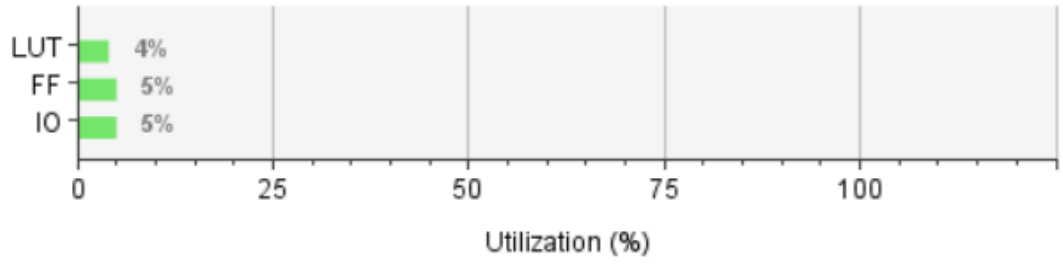
- **Synthesis stage :**
  - schematic :



  - utilization report :

Summary

Resource	Utilization	Available	Utilization %
LUT	851	20800	4.09
FF	2151	41600	5.17
IO	5	106	4.72



- **Synthesis stage :**

  - timing report : (Gray encoding) freq = 100MHz

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.502 ns	Worst Hold Slack (WHS):	0.140 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:	4296	Total Number of Endpoints:	4296	Total Number of Endpoints:	2152

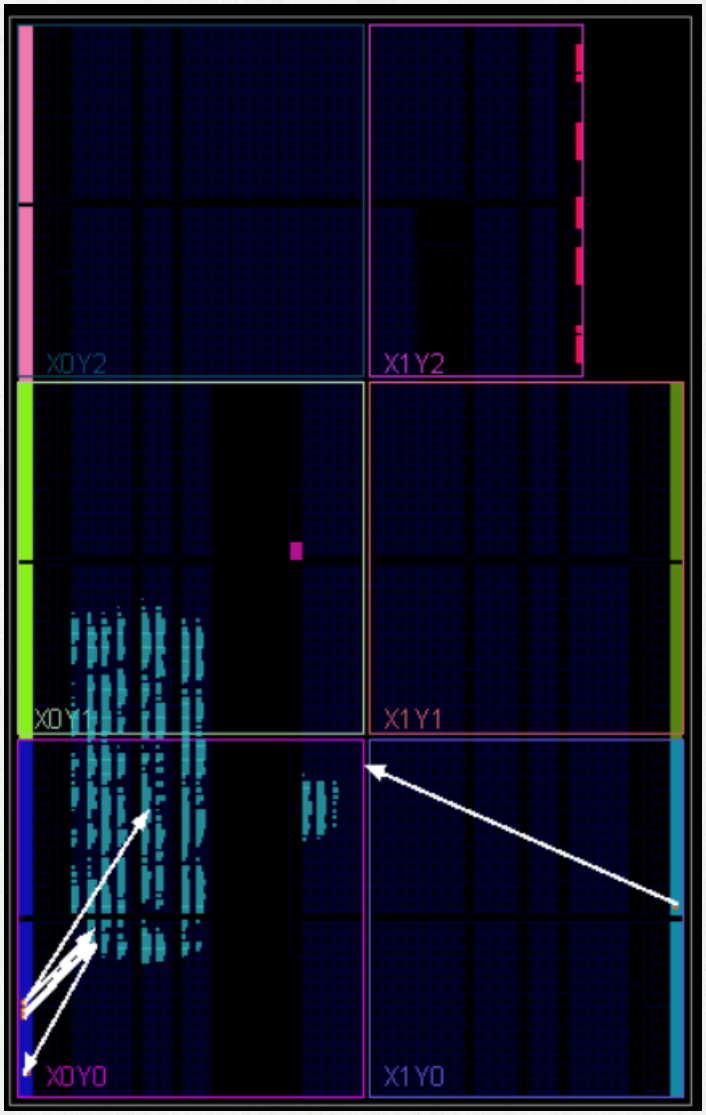
All user specified timing constraints are met.

  - synthesis report (to show the encoding type:)

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

- **Implementation stage**

  - device snippet :

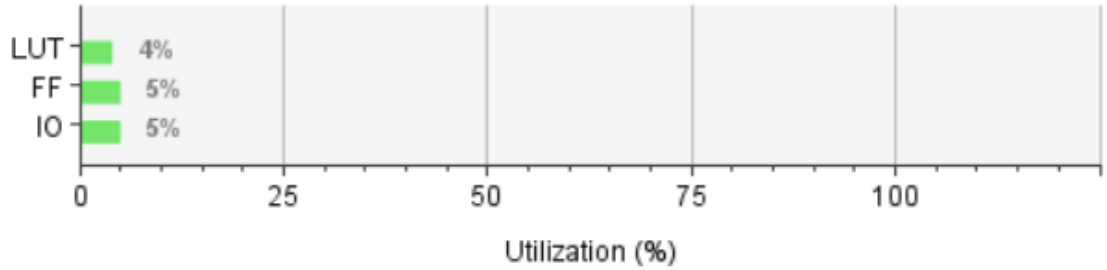


- **Implementation stage :**

  - utilization report :

Summary

Resource	Utilization	Available	Utilization %
LUT	851	20800	4.09
FF	2151	41600	5.17
IO	5	106	4.72



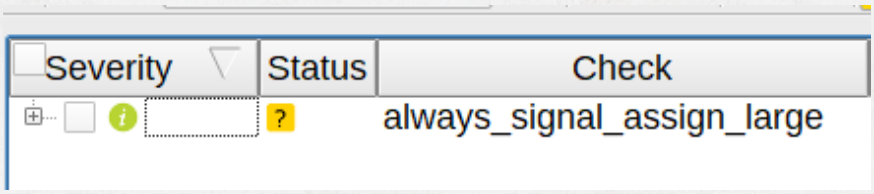
  - timing report :

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.025 ns	Worst Hold Slack (WHS):	0.109 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:	4296	Total Number of Endpoints:	4296	Total Number of Endpoints:	2152

All user specified timing constraints are met.

- **Lint result :**



Note:

changing the encoding type had almost no effect on the timing report, however gray encoding was more effective than one-hot encoding in the utilization report.
this was ecpected because one-hot encoding uses more resources.