Team name : 32bit

Names :

Mahmoud mohamed alsayd

Omar Khaled elsaid

Abdelkader saad

Codes :

Spi_slave :

```
module spi_slave (MOSI , MISO , SS_n , clk , rst_n , rx_data , rx_valid , tx_data , tx_valid );
parameter [2:0]IDLE = 3'b000;
parameter [2:0]READ_DATA = 3'b001;
parameter [2:0]READ_ADD = 3'b010;
parameter [2:0]CHK_CMD = 3'b011;
parameter [2:0]WRITE = 3'b100;

input MOSI  , SS_n , tx_valid, clk , rst_n;
input [7:0] tx_data;
output reg [10:0] rx_data;
output reg rx_valid , MISO=0;
(* fsm_encoding = "sequential" *)
reg [2:0] ns ,cs;
reg address_or_read=0;
reg control_bit ;
integer counter = 10;
integer counter_to_recieve = 7;

always @(posedge clk or negedge rst_n) begin
    if (~rst_n) begin
        cs <= IDLE ;
    end
    else  begin
        cs <= ns;
    end
end

//next stage
always @(cs or SS_n or control_bit ) begin
case (cs)
IDLE : begin
    if (SS_n == 0)
    ns = CHK_CMD;
    else if (SS_n == 1)
    ns = IDLE;
end

CHK_CMD : begin
    if (SS_n ==0 && control_bit==0 )
    ns = WRITE;
    else if (SS_n ==0 && control_bit ==1 && address_or_read ==0)begin
    ns = READ_ADD ;
    address_or_read =1;
    end
    else if (SS_n ==0 && control_bit ==1 && address_or_read ==1)begin
    ns = READ_DATA ;
    address_or_read =0;
```

```
    address_or_read =1;
    end
    else if (SS_n ==0 && control_bit ==1 && address_or_read ==1)begin
    ns = READ_DATA ;
    address_or_read =0;
    end
    else if (SS_n==1)
    ns = IDLE;
end


WRITE : begin
    if (SS_n == 0 )
    ns = WRITE;
    else if (SS_n == 1)
    ns = IDLE ;
    end

READ_ADD : begin
    if (SS_n ==0 )
    ns = READ_ADD;
    else if (SS_n==1)
    ns = IDLE ;

end

READ_DATA : begin
    if (SS_n == 0)
    ns = READ_DATA;
    else if (SS_n==1)
    ns = IDLE;
end
default : ns = IDLE;
endcase
end


//output logic
always @(posedge clk) begin

case (cs)
IDLE :begin
    rx_valid <=0;
    control_bit <= MOSI;//0
end

WRITE : begin
```

```
79
80  //output logic
81  always @(posedge clk) begin
82
83  case (cs)
84  IDLE :begin
85      rx_valid <=0;
86      control_bit <= MOSI;//0
87  end
88
89  WRITE : begin
90      rx_valid <=1;
91      rx_data[10] <= control_bit ;
92      rx_data [counter -1] <= MOSI;
93      counter <= counter -1;
94      if (counter == 0)
95      counter <= 10;
96  end
97
98  READ_ADD : begin
99      rx_valid <= 1;
100     rx_data [10] <= control_bit;
101     rx_data [counter-1] <= MOSI;
102     counter <= counter -1;
103     if (counter==0)
104     counter <= 10;
105 end
106
107 READ_DATA : begin
108     rx_data [10] <= control_bit;
109     rx_data [counter-1] <= MOSI;
110     counter <= counter -1;
111
112 if (counter==0)
113 counter <=10;
114
115 if (tx_valid) begin
116 MISO <= tx_data[counter_to_recieve];
117 counter_to_recieve <= counter_to_recieve -1;
118 end
119
120 if (counter_to_recieve==0)
121 counter_to_recieve <= 7;
122
123 end
124 endcase
125 end
126 endmodule
```

Ram :

```
module ram (din,clk,rst_n,rx_valid,dout,tx_valid);
parameter MEM_DEPTH = 256;
parameter ADDR_SIZE=8;
input [10:0] din;
input clk , rst_n , rx_valid;
output reg [7:0] dout;
output reg tx_valid;

reg [7:0] mem [MEM_DEPTH-1:0];
reg [ADDR_SIZE-1:0] wr_addr,rd_addr;
always @(posedge clk or negedge rst_n) begin
    if (~rst_n)begin
        dout <= 0;
        end
    else if (rx_valid) begin
        if (din[9:8] == 2'b00) begin
            wr_addr <= din[7:0];
            tx_valid <= 0;
        end


        else if (din[9:8] == 2'b01) begin
            mem[wr_addr] <= din[7:0];
            tx_valid <= 0;
        end


        else if (din[9:8] == 2'b10) begin
            rd_addr <= din[7:0];
            tx_valid <= 0;
        end
    end


    else if (din[9:8] == 2'b11) begin
            dout <= mem[rd_addr];
            tx_valid <= 1;
    end
end

endmodule
```

Wrapper :

```v
module spi_wrapper (MOSI , MISO , SS_n , clk , rst_n);

input MOSI , SS_n , clk , rst_n ;
output MISO ;

wire[10:0] rx_data ;
wire rx_valid , tx_valid ;
wire [7:0] tx_data ;

spi_slave dut1 (MOSI , MISO , SS_n , clk , rst_n , rx_data , rx_valid , tx_data , tx_valid );

ram dut2(rx_data ,clk , rst_n , rx_valid , tx_data ,tx_valid);
endmodule

```

Mem.dat:

spi_slave.v | ram_spi.v | wrapper.v | mem.dat

```
0
1
2
3
4
5
6
7
```

Testbench:

```v
module SPI_Wrapper_tb();
reg MOSI, SS_n, clk, rst_n;
wire MISO;
reg [9:0] write_addr = 10'b00_1111_0000;//address 240 in decimal

reg [9:0] write_data = 10'b01_1010_1110; // this data will write in the previos address

reg [9:0] read_addr = 10'b10_1111_0000; // address 240

reg [9:0] read_data = 10'b11_0000_0000;// it will be dummy data but we want only 11 to make ram to know that it in read state
integer i = 0;
spi_wrapper DUT(MOSI, MISO, SS_n, clk, rst_n);

initial begin
    clk =0;
    forever
    #10 clk = ~clk;
end

initial begin
    $readmemh ("mem.dat" , DUT.dut2.mem);
    rst_n = 0;
    SS_n = 1;
    #50;
    rst_n = 1;
    MOSI = 0;
    SS_n = 0;
    #50
    for (i=0; i<10 ; i=i+1)begin
        @(negedge clk)
        MOSI = write_addr [9-i];
        #5;
    end
    SS_n = 1;
    #50


    // to write data in the previos address
    MOSI = 0;
    SS_n = 0;
    #50
    for (i=0 ; i<10 ; i=i+1)begin
        @(negedge clk)
        MOSI = write_data [9-i];
        #5;
    end
    SS_n = 1;
    #50
    // to write the address that i will read from
```

```
        end
        SS_n = 1;
        #50


        // to write data in the previos address
        MOSI = 0;
        SS_n = 0;
        #50
        for (i=0 ; i<10 ; i=i+1)begin
            @(negedge clk)
            MOSI = write_data [9-i];
            #5;
        end
        SS_n = 1;
        #50
        // to write the address that i will read from
        MOSI =1;
        SS_n =0;
        #50
        for (i=0 ; i<10 ; i=i+1)begin
            @(negedge clk)
            MOSI = read_addr [9-i];
            #5;
        end
        SS_n =1;
        #50
        MOSI =1;
        SS_n =0;
        #50
        for (i=0 ; i<10 ; i=i+1)begin
            @(negedge clk)
            MOSI = read_data [9-i];
            #5;
        end

        #40 $stop;
    end

    endmodule
```

Do file :

```
vlib work

vlog ram_spi.v spi_slave.v testbench.v wrapper.v

vsim -voptargs=+acc work.SPI_Wrapper_tb

add wave *
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/dut2/mem
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/dut1/tx_valid \
sim:/SPI_Wrapper_tb/DUT/dut1/tx_data \
sim:/SPI_Wrapper_tb/DUT/dut1/rx_data \
sim:/SPI_Wrapper_tb/DUT/dut1/rx_valid

run -all
```

Tcl file :

```
create_project project_SPI_ram C:/Users/M/Desktop/projectspislave -part xc7a35ticpg236-1L -force

add_files ram_spi.v spi_slave.v wrapper.v basys_master.xdc

synth_design -rtl -top spi_wrapper > elab.log
#report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbose -max_paths 10 -input_pins -routable_nets -name tim

write_schematic elaborated_schematic.pdf -format pdf -force

launch_runs synth_1 > synth.log

wait_on_run synth_1
open_run synth_1

write_schematic synthesized_schematic.pdf -format pdf -force

write_verilog -force SPI_netlist.v

launch_runs impl_1 -to_step write_bitstream

wait_on_run impl_1
open_run impl_1

open_hw

connect_hw_server
```

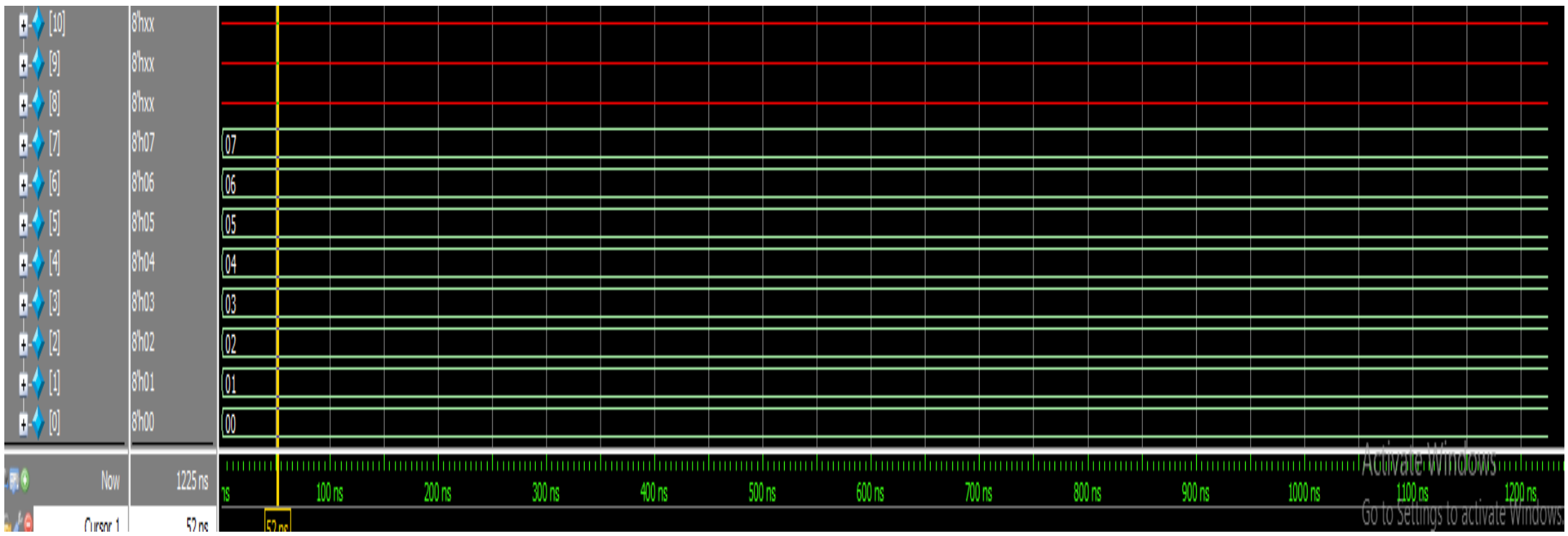
Note : here I want to make schematic after synthesis as pdf in file and write it in tcl but when run no pdf appear and no errors or critical warning appear .

Constrain file :

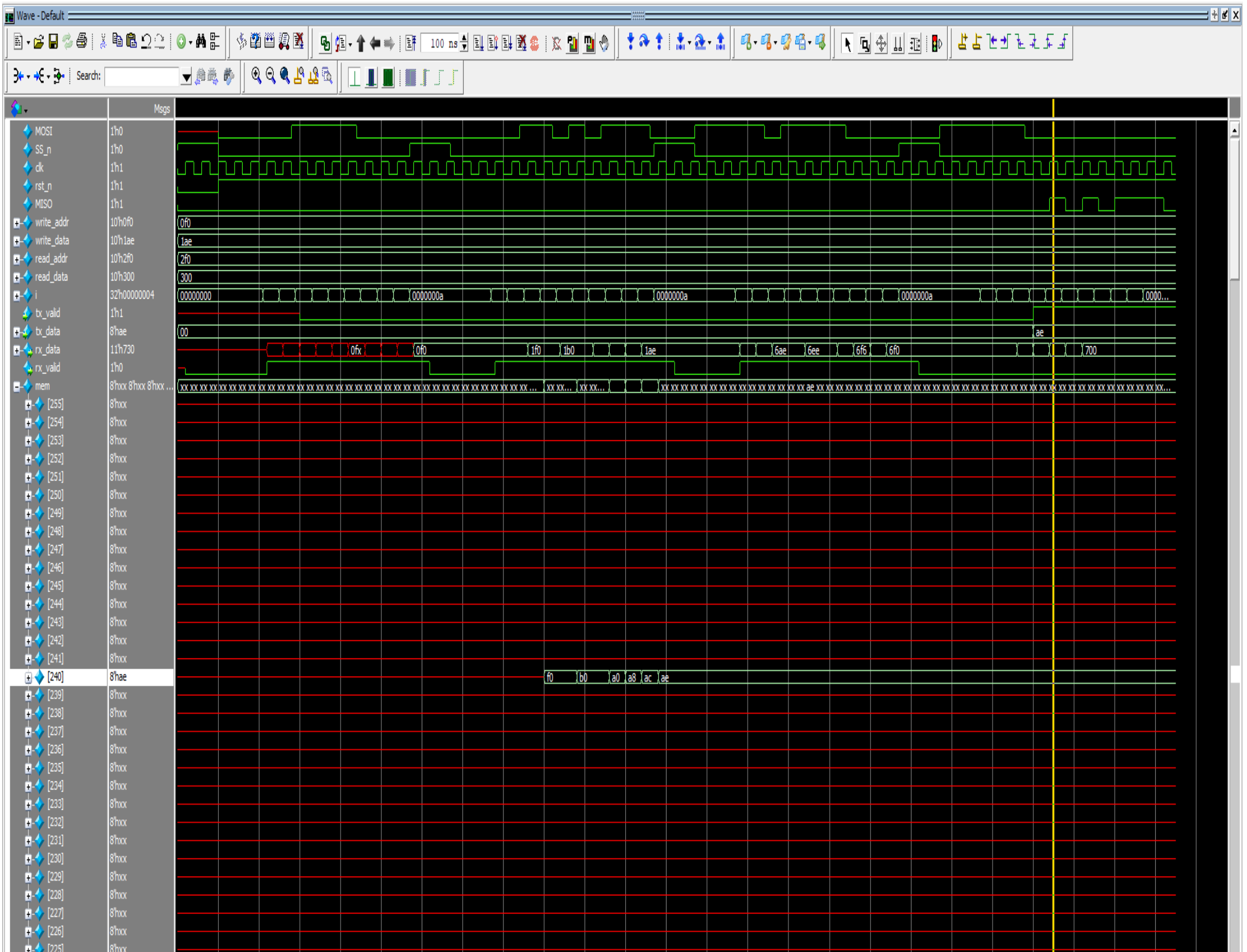
```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## 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 {rst_n}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {SS_n}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {MOSI}]
# set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {A[0]}]
# set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {A[1]}]
# set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {A[2]}]
# set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {B[0]}]
# set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {B[1]}]
# set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {B[2]}]
# set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {cin}]
# set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {red_op_A}]
# set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {red_op_B}]
# set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {bypass_A}]
# set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {bypass_B}]
# set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {direction}]
# set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {serial_in}]


## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {MISO}]
# set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {leds[1]}]
# set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {leds[2]}]
# set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {leds[3]}]
# set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {leds[4]}]
# set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {leds[5]}]
# set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {leds[6]}]
```

Simulation in questasim :

as we see the data load as in mem.dat file



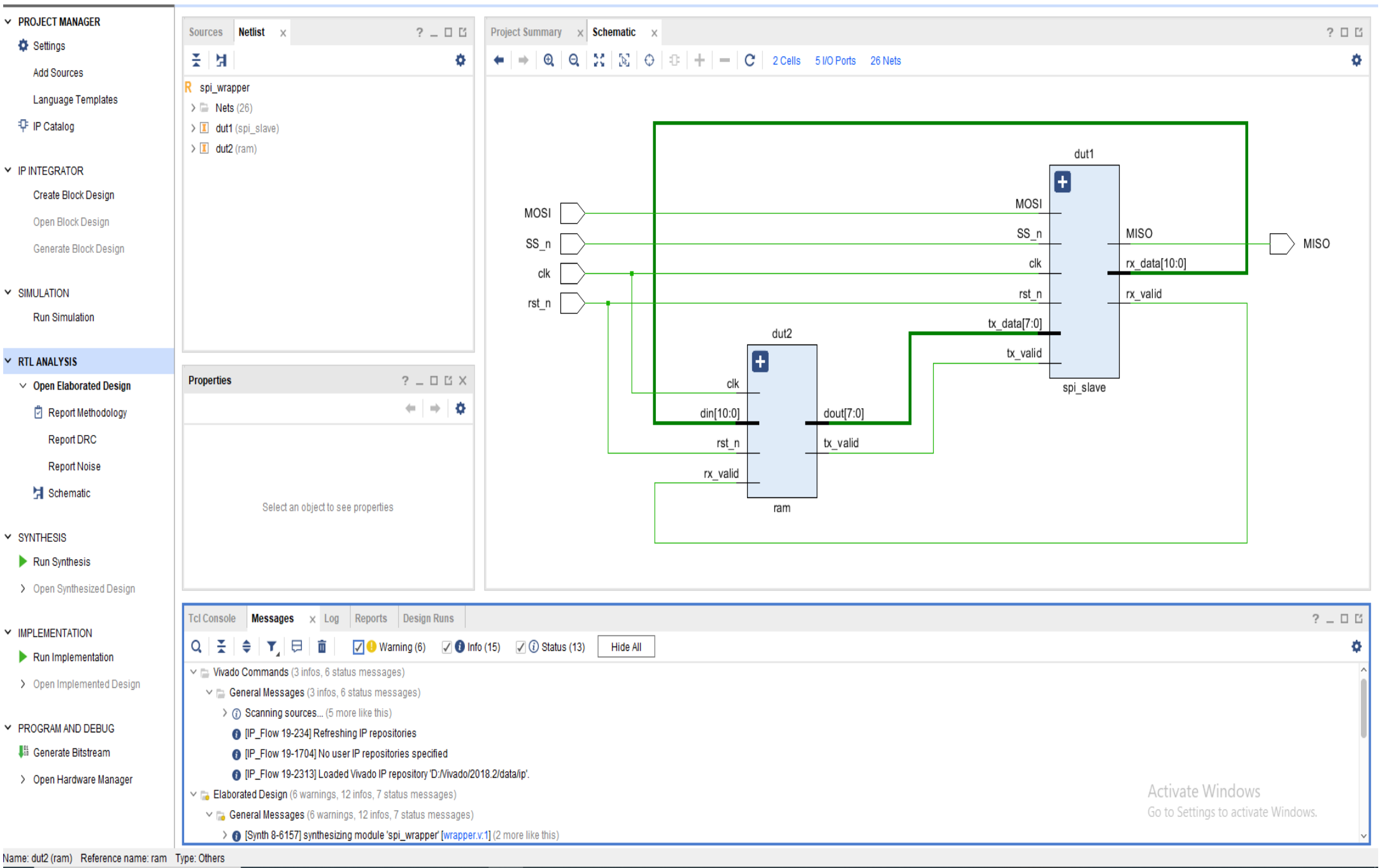
as we see the value loaded in address 240 (ae) and the value that we want in this same address will the same in miso (10101111):



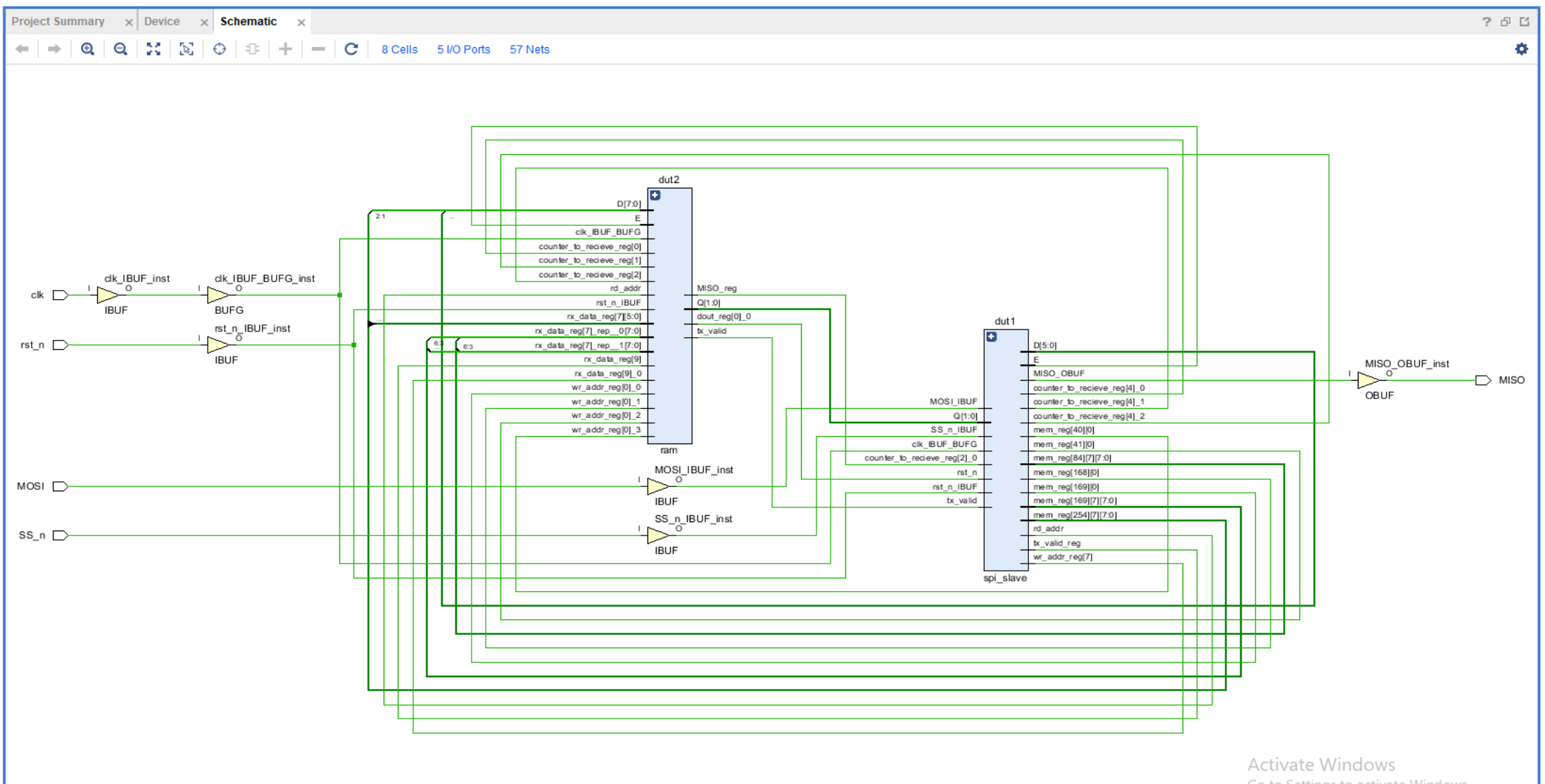
Vivado :

Gray encoding :

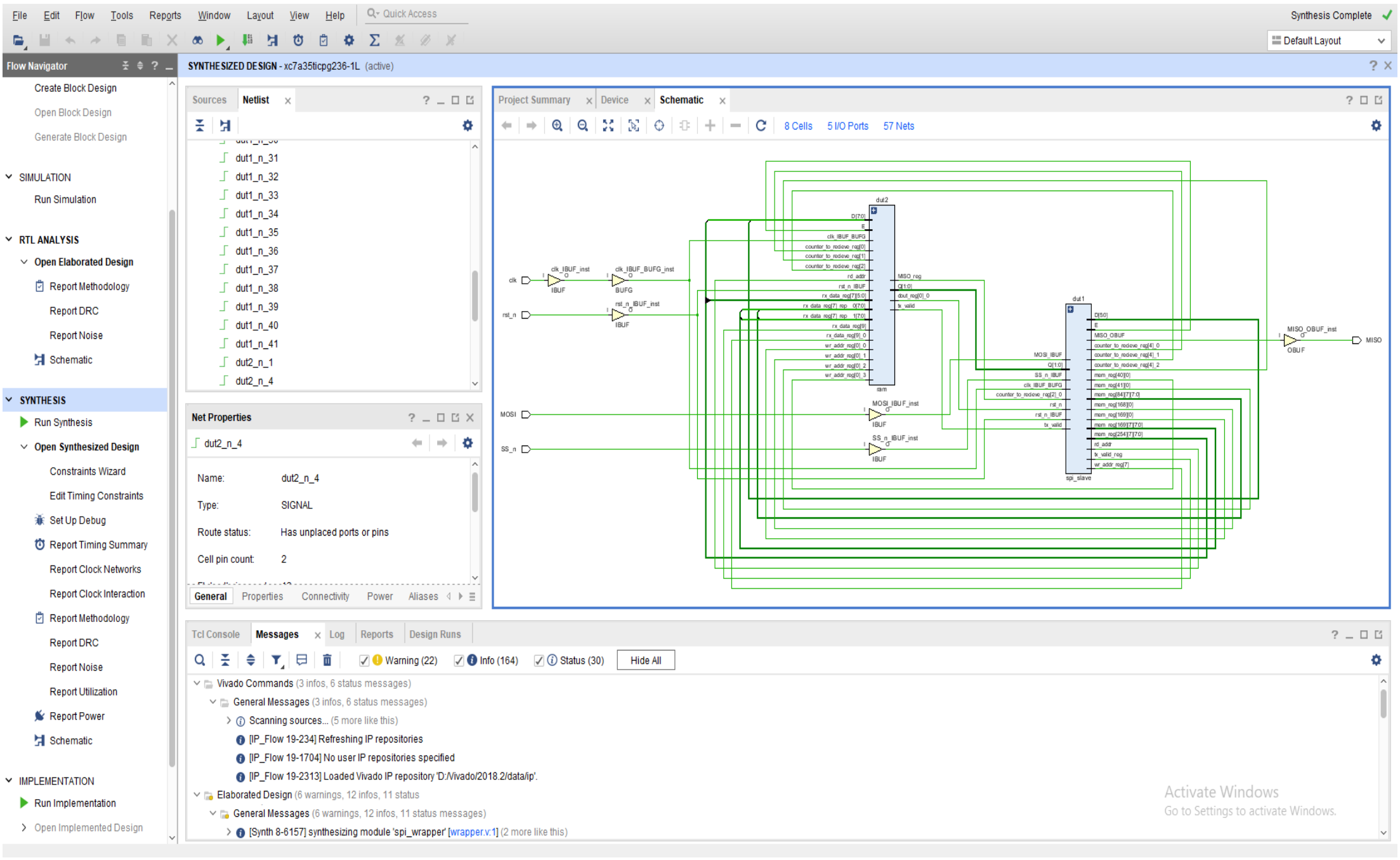
Schematic after elaborated without error :



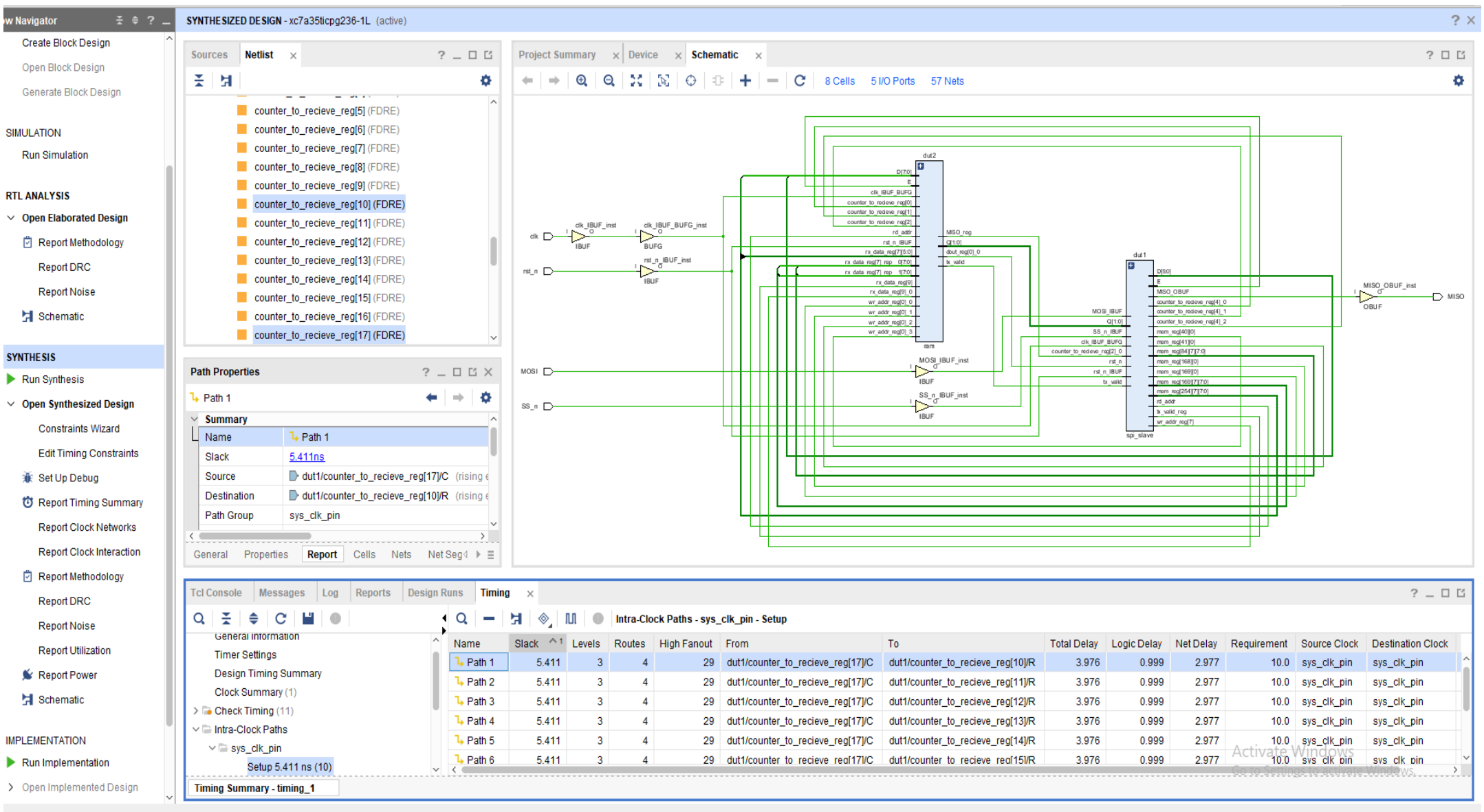
Schematic after synthesis :



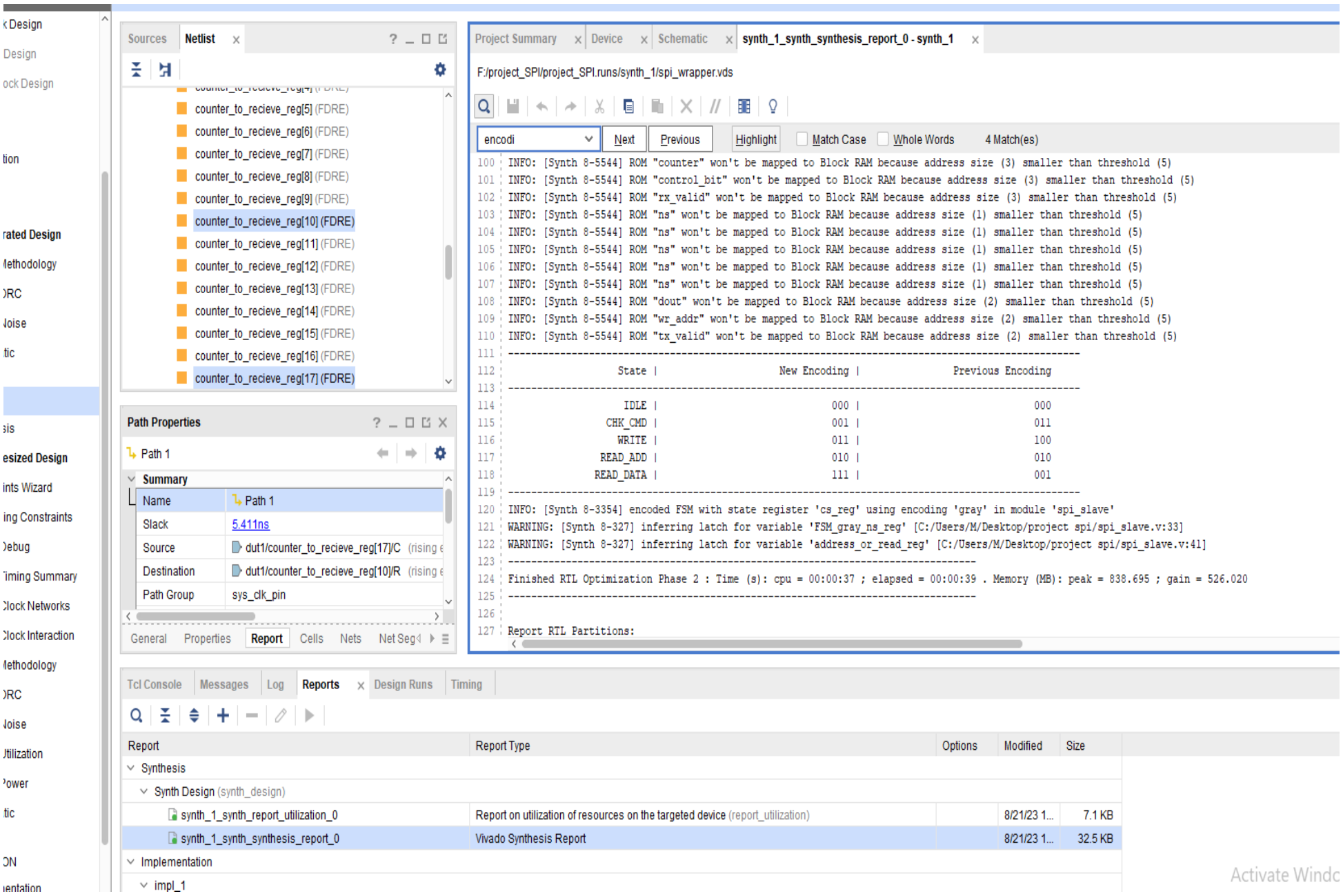
Schematic without error :



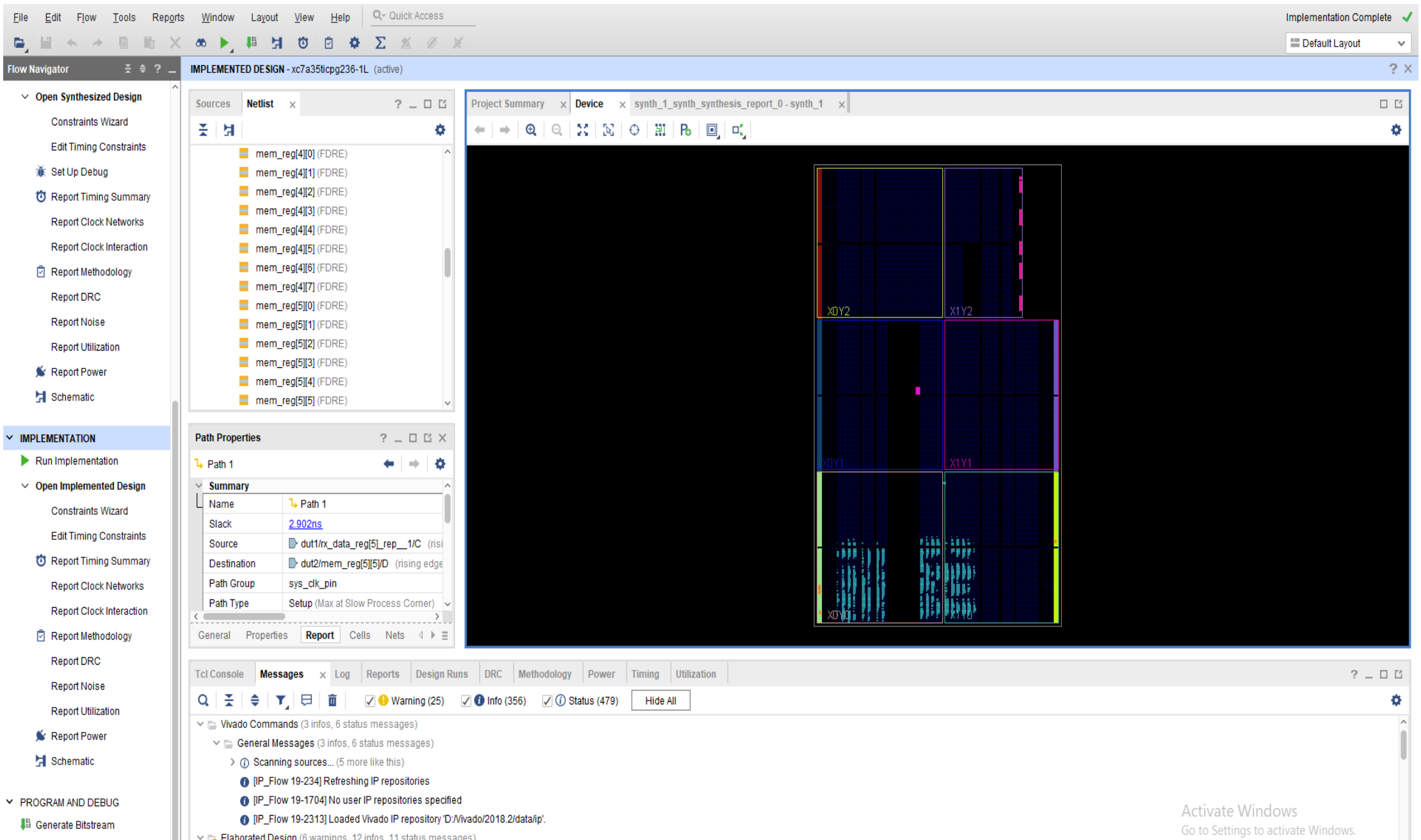
Time report :



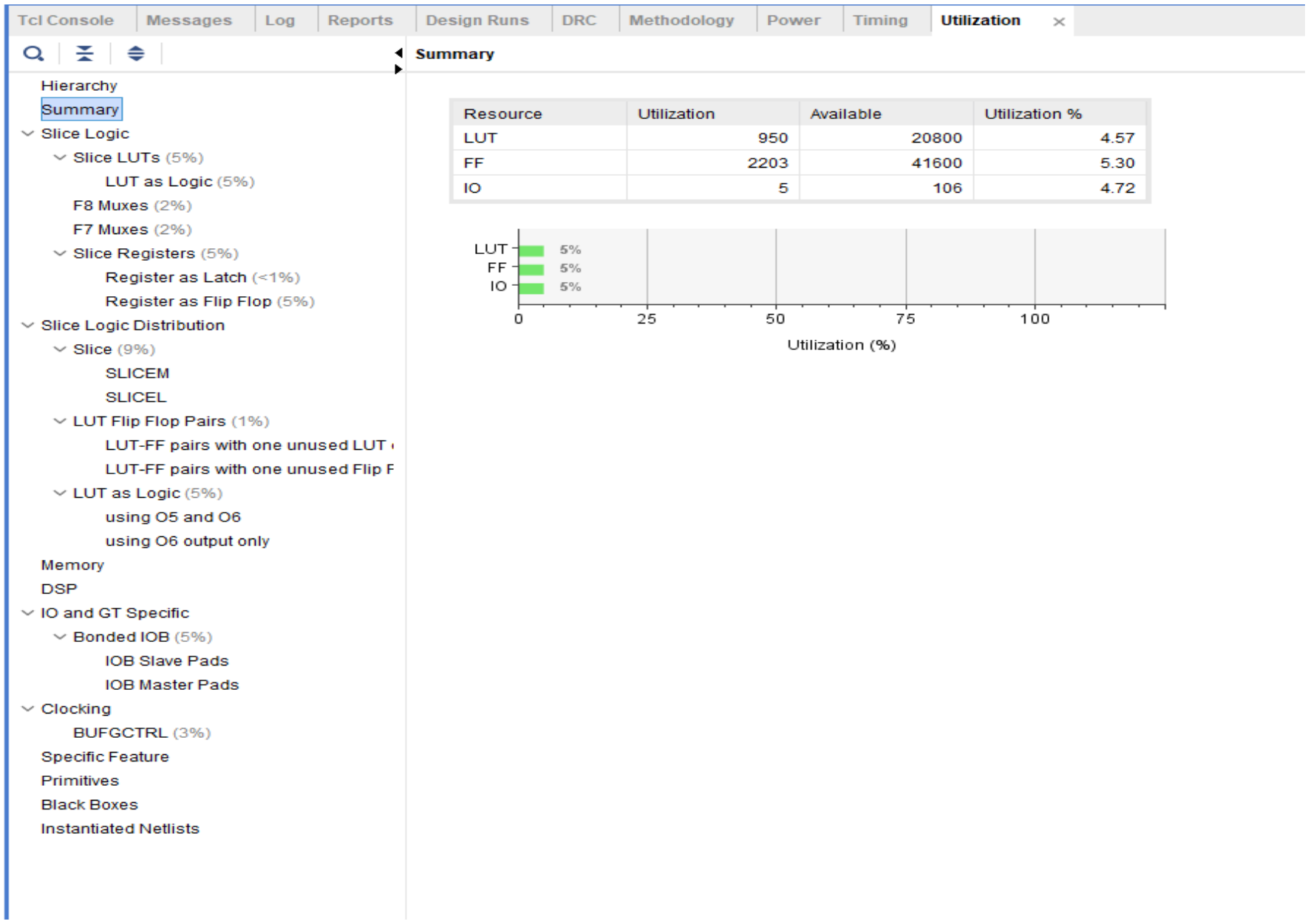
Gray encoding :



Implementation device and no error :

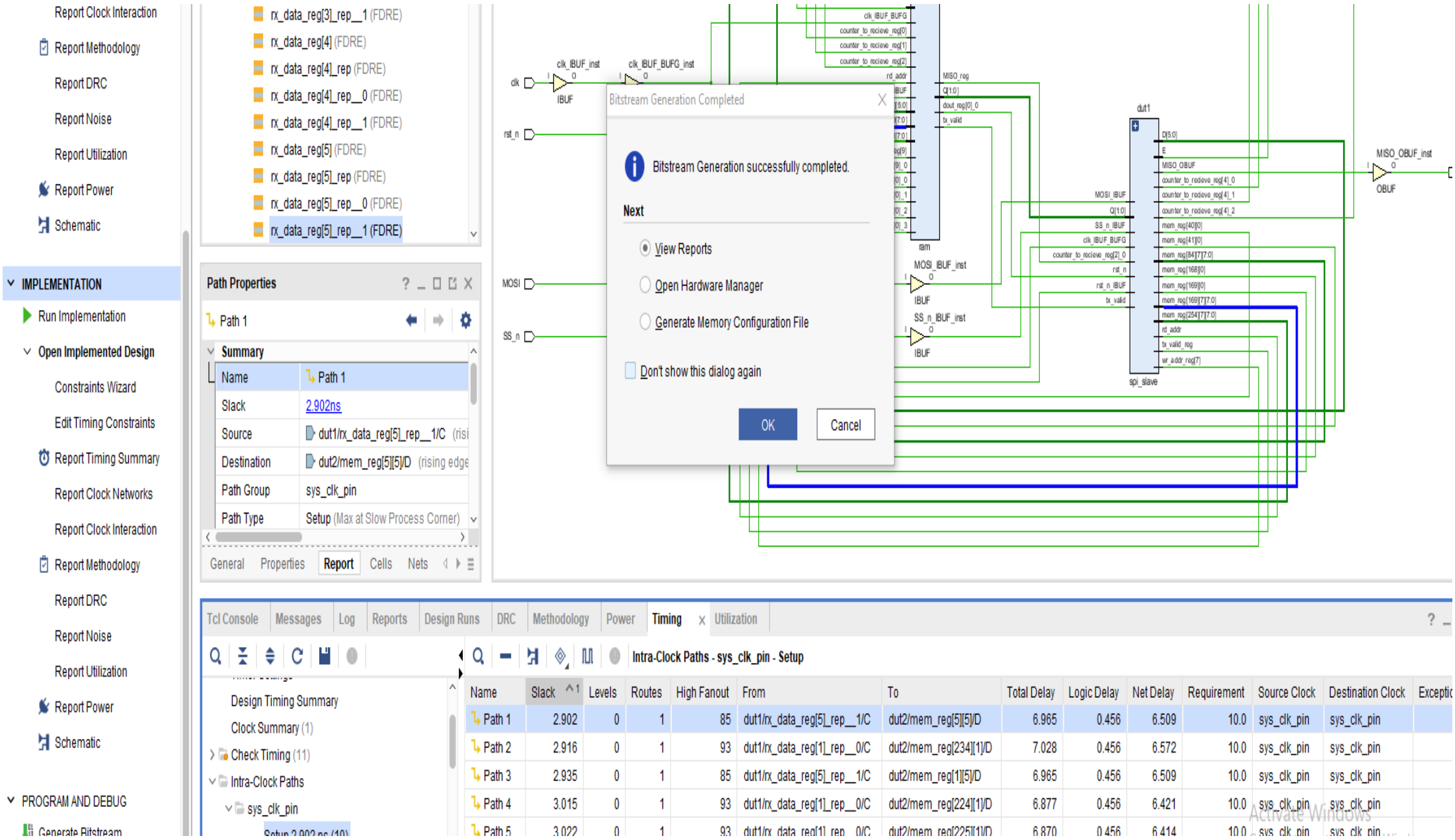


Utilization report :



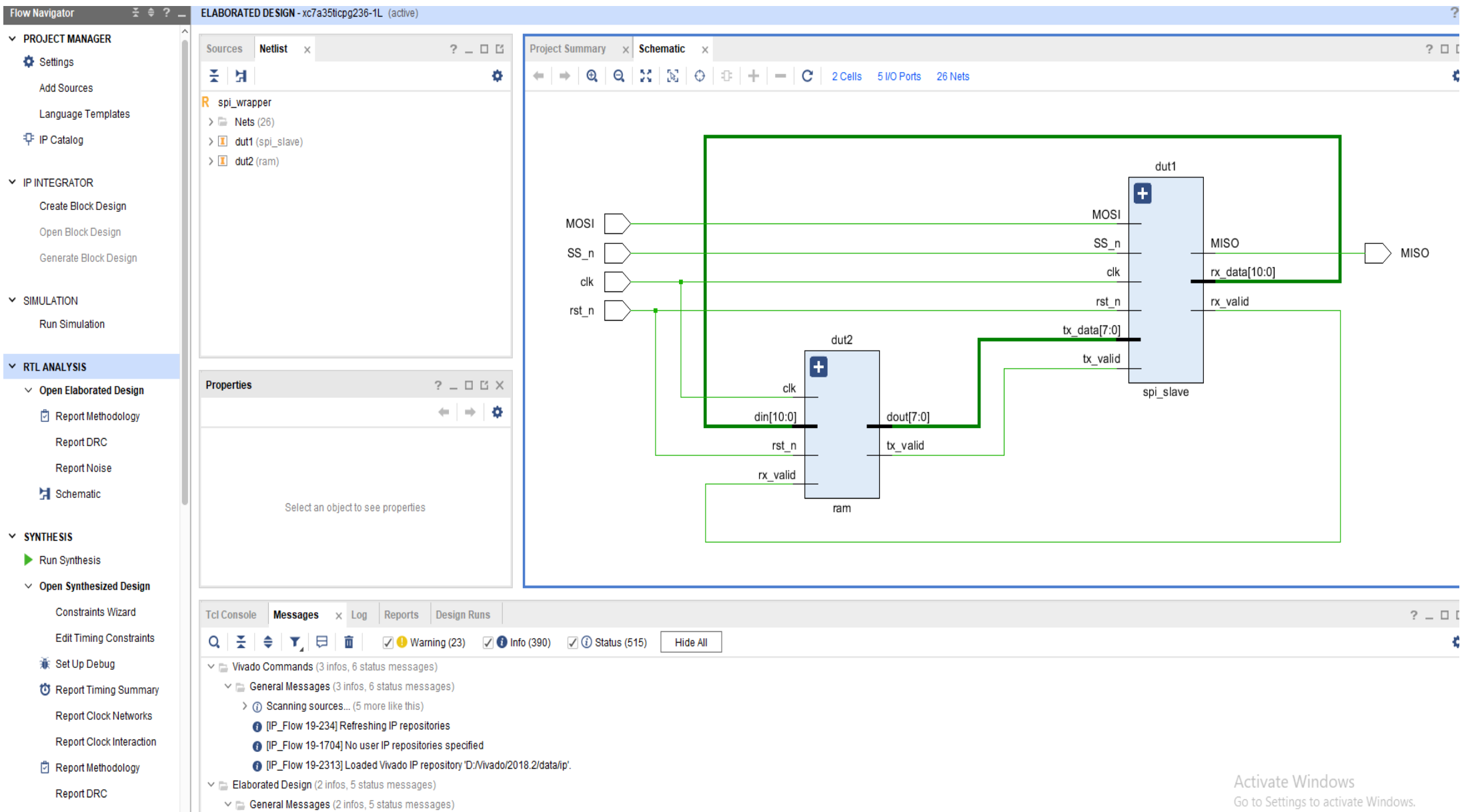
worst_slack in implementaion_gray:

Successful generate bit:

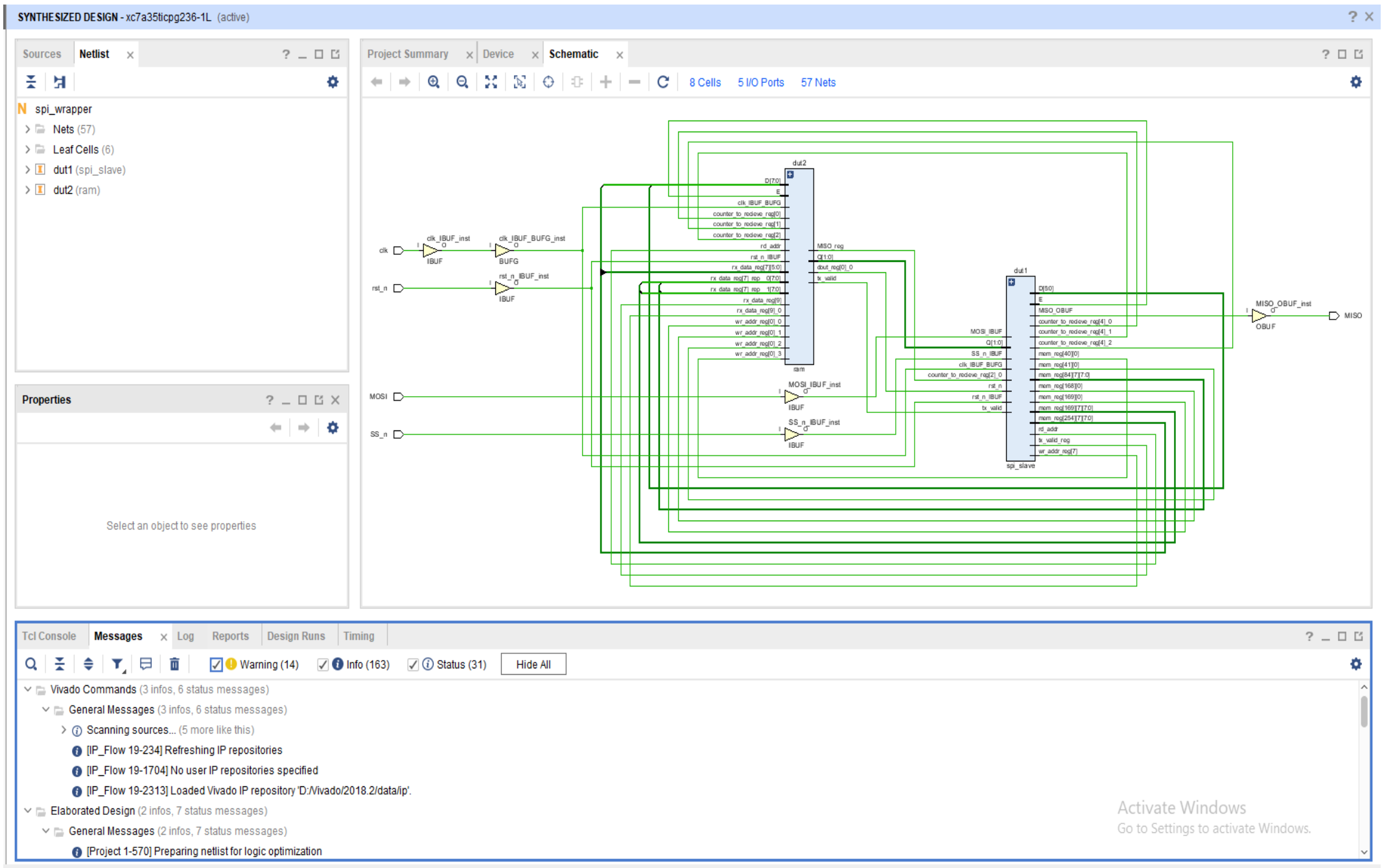


One_hot_encoding :

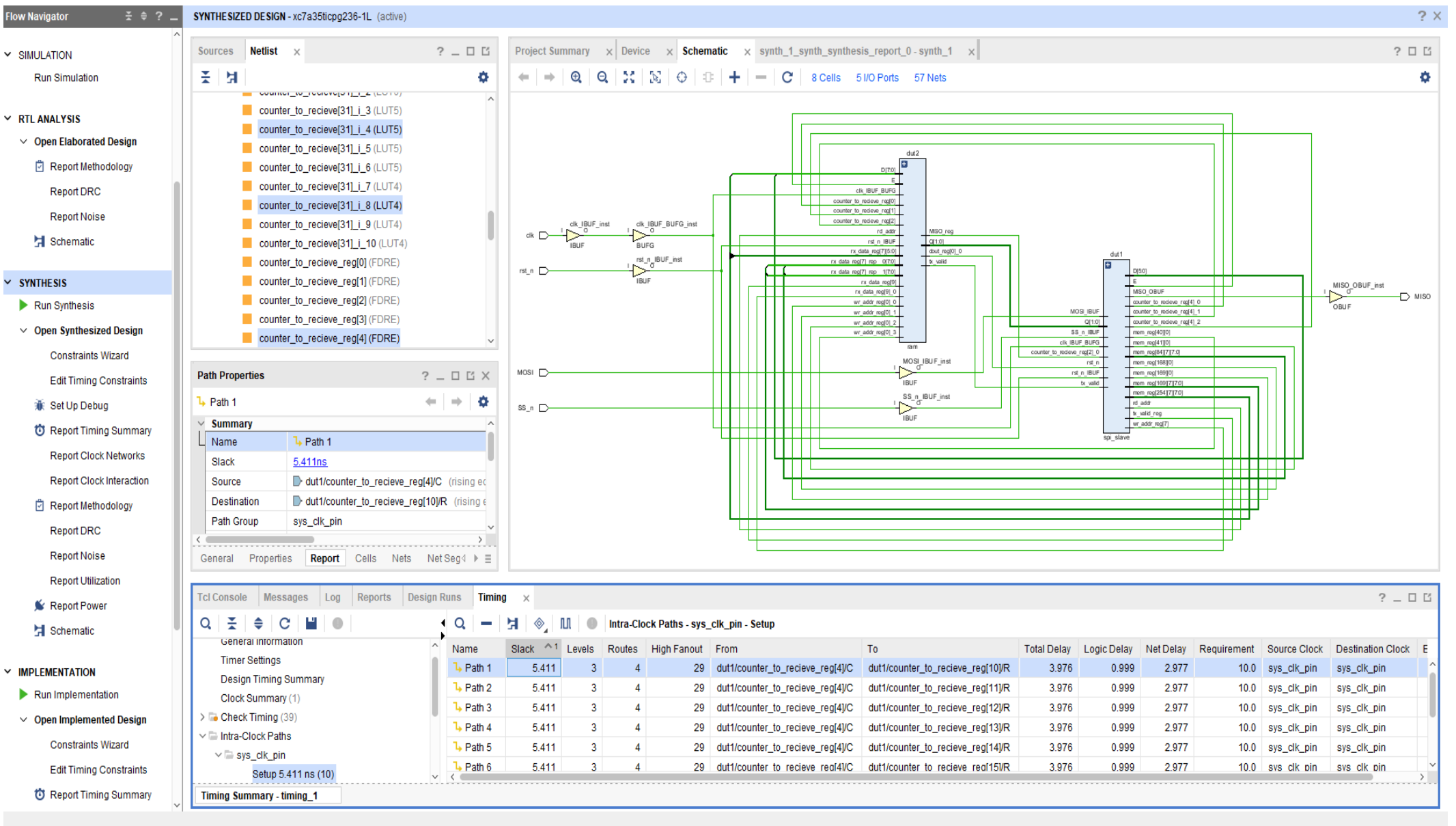
schematic after elaborated without error :



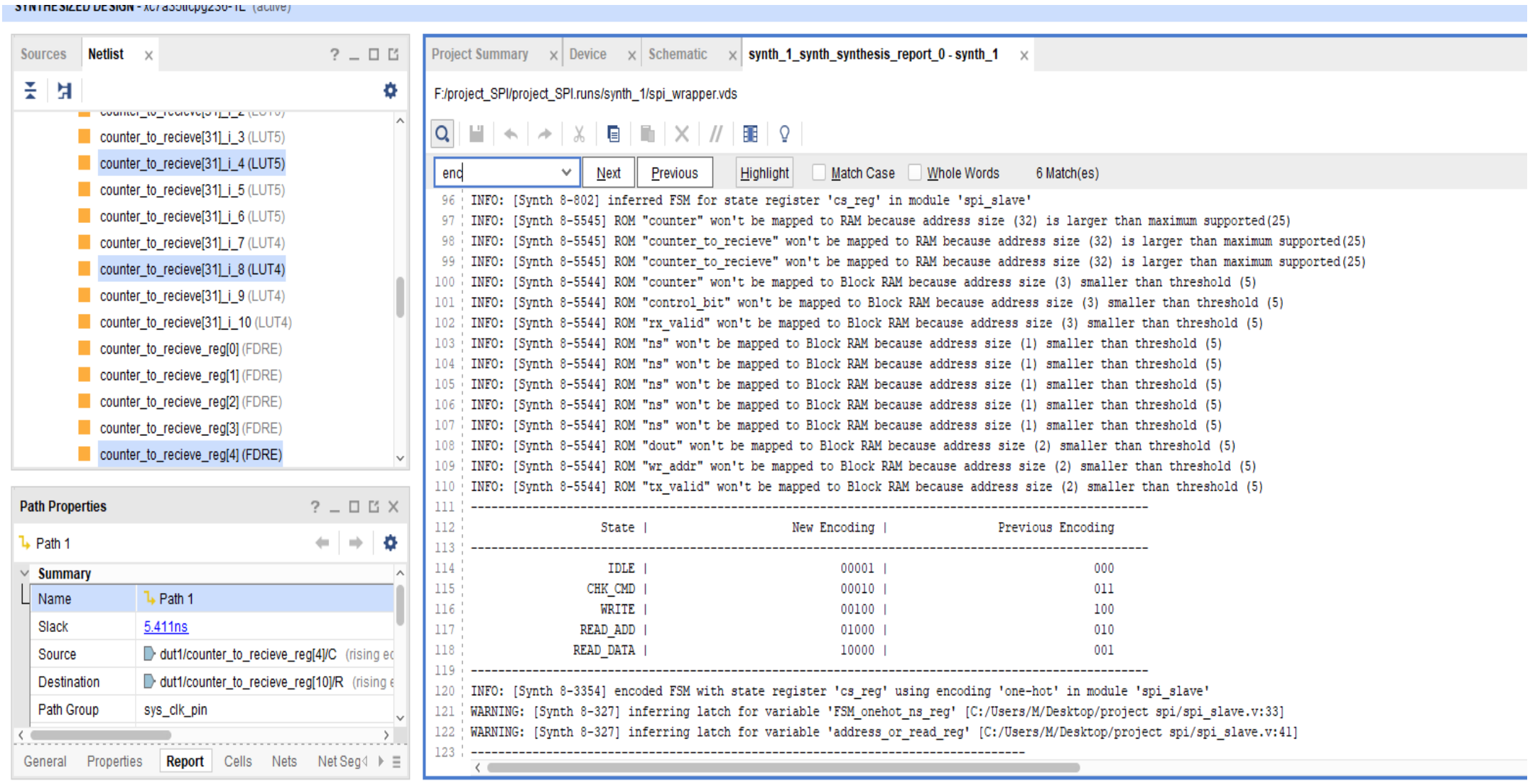
Schematic after synthesis with no error:



Time report after synthesis the same:

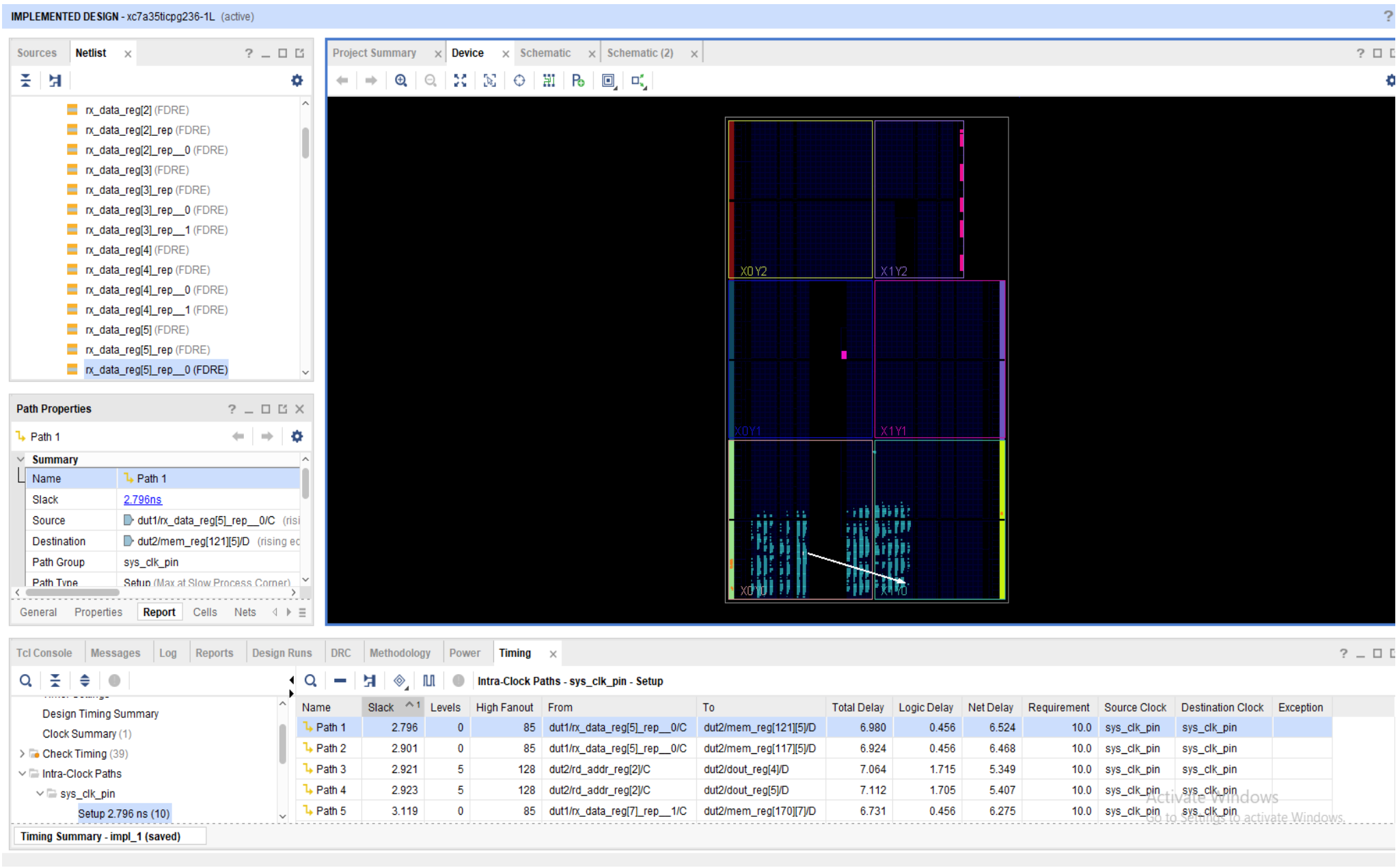


One_hot encoding :



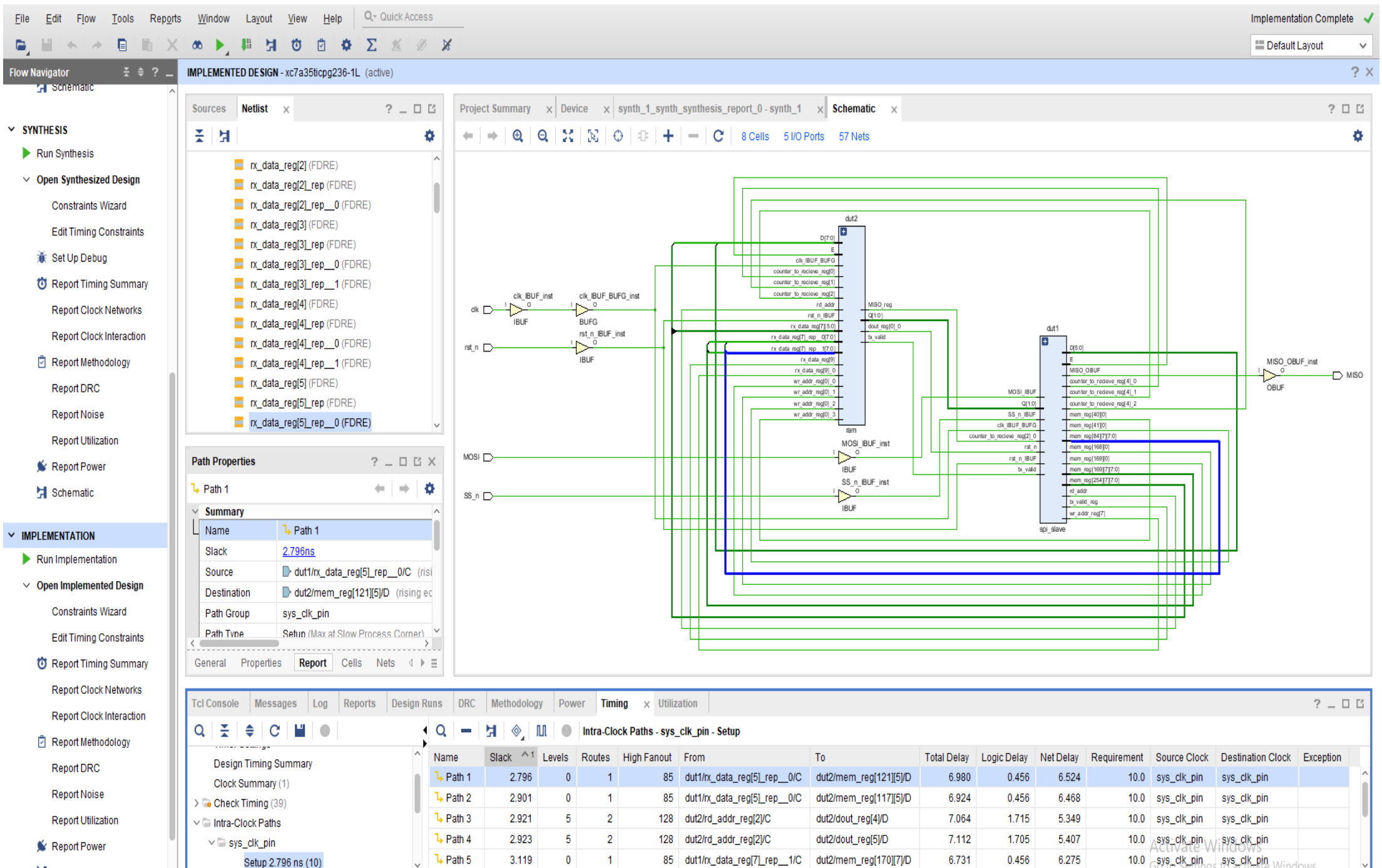
Schematic after implementation without errors :

Device :



Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	2.796	0	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[121][5]/D	6.980	0.456	6.524	10.0	sys_clk_pin	sys_clk_pin	
Path 2	2.901	0	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[117][5]/D	6.924	0.456	6.468	10.0	sys_clk_pin	sys_clk_pin	
Path 3	2.921	5	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[4]/D	7.064	1.715	5.349	10.0	sys_clk_pin	sys_clk_pin	
Path 4	2.923	5	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[5]/D	7.112	1.705	5.407	10.0	sys_clk_pin	sys_clk_pin	
Path 5	3.119	0	85	dut1/rx_data_reg[7]_rep__1/C	dut2/mem_reg[170][7]/D	6.731	0.456	6.275	10.0	sys_clk_pin	sys_clk_pin	

Worst slack after implementation :



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	2.796	0	1	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[121][5]/D	6.980	0.456	6.524	10.0	sys_clk_pin	sys_clk_pin	
Path 2	2.901	0	1	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[117][5]/D	6.924	0.456	6.468	10.0	sys_clk_pin	sys_clk_pin	
Path 3	2.921	5	2	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[4]/D	7.064	1.715	5.349	10.0	sys_clk_pin	sys_clk_pin	
Path 4	2.923	5	2	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[5]/D	7.112	1.705	5.407	10.0	sys_clk_pin	sys_clk_pin	
Path 5	3.119	0	1	85	dut1/rx_data_reg[7]_rep__1/C	dut2/mem_reg[170][7]/D	6.731	0.456	6.275	10.0	sys_clk_pin	sys_clk_pin	

Utilization report :

Tcl Console | Messages | Log | Reports | Design Runs | DRC | Methodology | Power | Timing | **Utilization**

Hierarchy
Summary
Slice Logic
- Slice LUTs (5%)
  - LUT as Logic (5%)
- F8 Muxes (2%)
- F7 Muxes (2%)
- Slice Registers (5%)
  - Register as Latch (<1%)
  - Register as Flip Flop (5%)

Slice Logic Distribution
- Slice (10%)
  - SLICEM
  - SLICEL
- LUT Flip Flop Pairs (1%)
  - LUT-FF pairs with one unused LUT
  - LUT-FF pairs with one unused Flip F
- LUT as Logic (5%)
  - using O5 and O6
  - using O6 output only

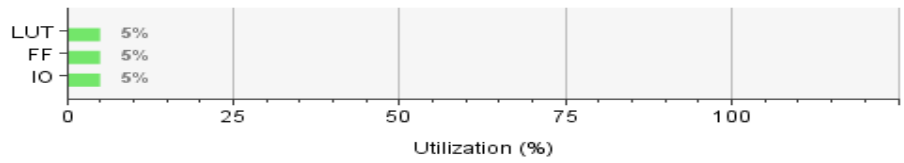
Memory
DSP
IO and GT Specific
- Bonded IOB (5%)
  - IOB Slave Pads
  - IOB Master Pads

Clocking
- BUFGCTRL (3%)

Specific Feature
Primitives
Black Boxes
Instantiated Netlists

Summary

Resource	Utilization	Available	Utilization %
LUT	951	20800	4.57
FF	2210	41600	5.31
IO	5	106	4.72



Generate bit stream successful :

IMPLEMENTED DESIGN - xc7a35ticpg236-1L (active)

Sources | **Netlist**

- rx_data_reg[2] (FDRE)
- rx_data_reg[2]_rep (FDRE)
- rx_data_reg[2]_rep__0 (FDRE)
- rx_data_reg[3] (FDRE)
- rx_data_reg[3]_rep (FDRE)
- rx_data_reg[3]_rep__0 (FDRE)
- rx_data_reg[3]_rep__1 (FDRE)
- rx_data_reg[4] (FDRE)
- rx_data_reg[4]_rep (FDRE)
- rx_data_reg[4]_rep__0 (FDRE)
- rx_data_reg[4]_rep__1 (FDRE)
- rx_data_reg[5] (FDRE)
- rx_data_reg[5]_rep (FDRE)
- rx_data_reg[5]_rep__0 (FDRE)

Path Properties

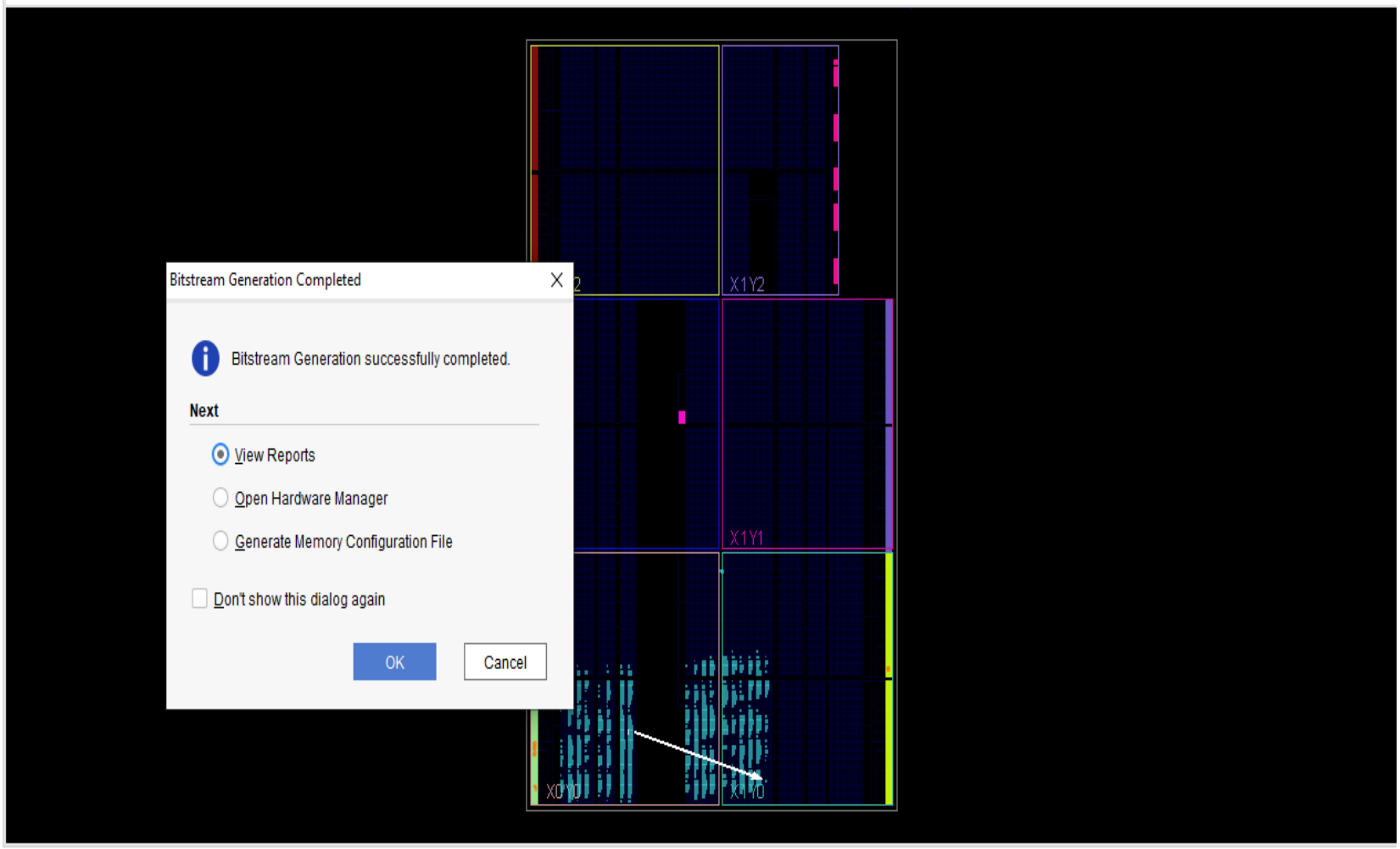
Path 1

Summary

Name	Path 1
Slack	2.796ns
Source	dut1/rx_data_reg[5]_rep__0/C (risi
Destination	dut2/mem_reg[121][5]/D (rising ec
Path Group	sys_clk_pin
Path Type	Setup (Max at Slow Process Corner)

General | Properties | **Report** | Cells | Nets

Project Summary | **Device** | Schematic | Schematic (2)



Tcl Console | Messages | Log | Reports | Design Runs | DRC | Methodology | Power | **Timing**

Design Timing Summary
Clock Summary (1)
Check Timing (39)
Intra-Clock Paths
- sys_clk_pin
  - Setup 2.796 ns (10)

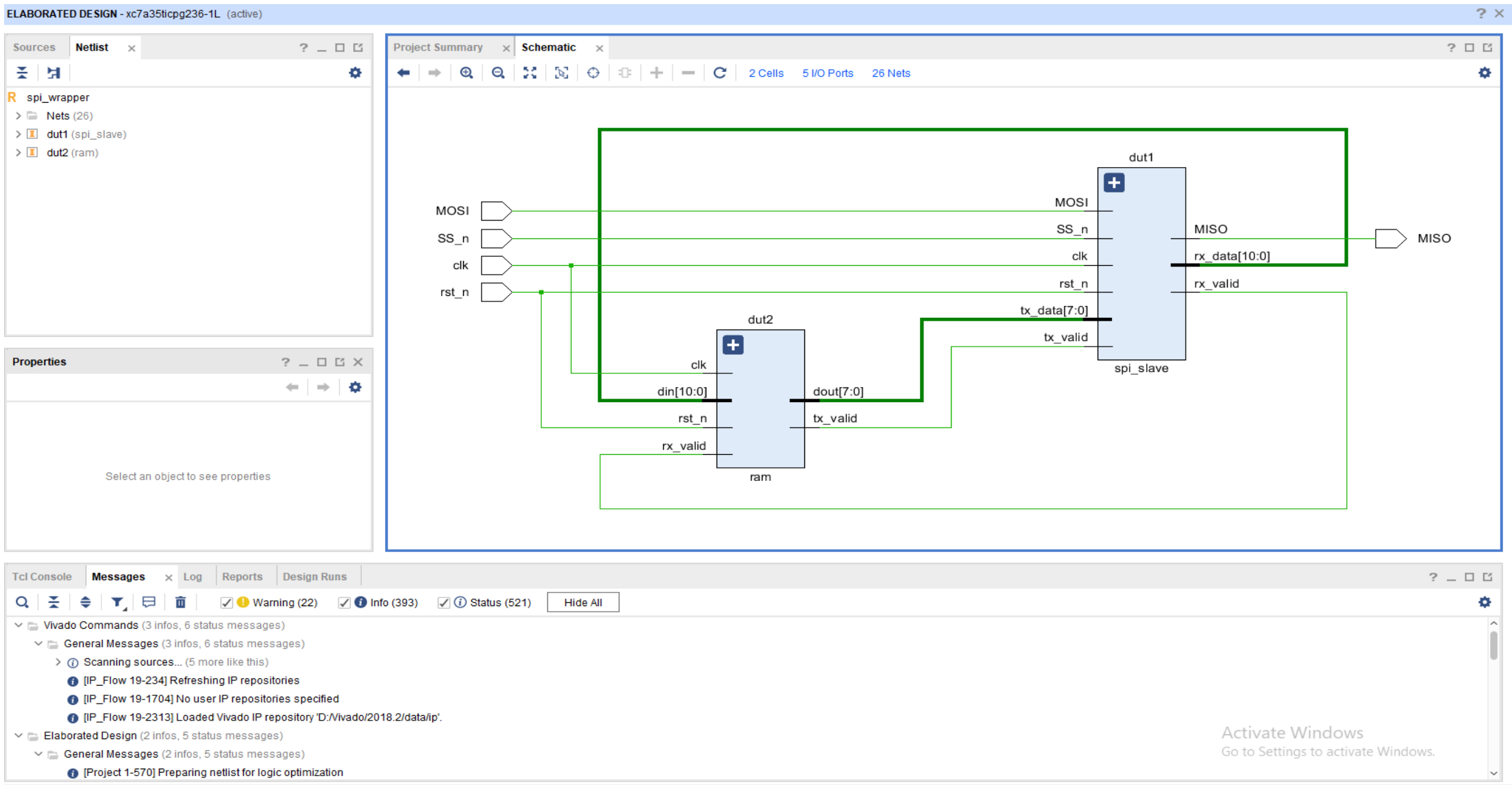
Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	2.796	0	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[121][5]/D	6.980	0.456	6.524	10.0	sys_clk_pin	sys_clk_pin	
Path 2	2.901	0	85	dut1/rx_data_reg[5]_rep__0/C	dut2/mem_reg[117][5]/D	6.924	0.456	6.468	10.0	sys_clk_pin	sys_clk_pin	
Path 3	2.921	5	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[4]/D	7.064	1.715	5.349	10.0	sys_clk_pin	sys_clk_pin	
Path 4	2.923	5	128	dut2/rd_addr_reg[2]/C	dut2/dout_reg[5]/D	7.112	1.705	5.407	10.0	sys_clk_pin	sys_clk_pin	
Path 5	3.119	0	85	dut1/rx_data_reg[7]_rep__1/C	dut2/mem_reg[170][7]/D	6.731	0.456	6.275	10.0	sys_clk_pin	sys_clk_pin	

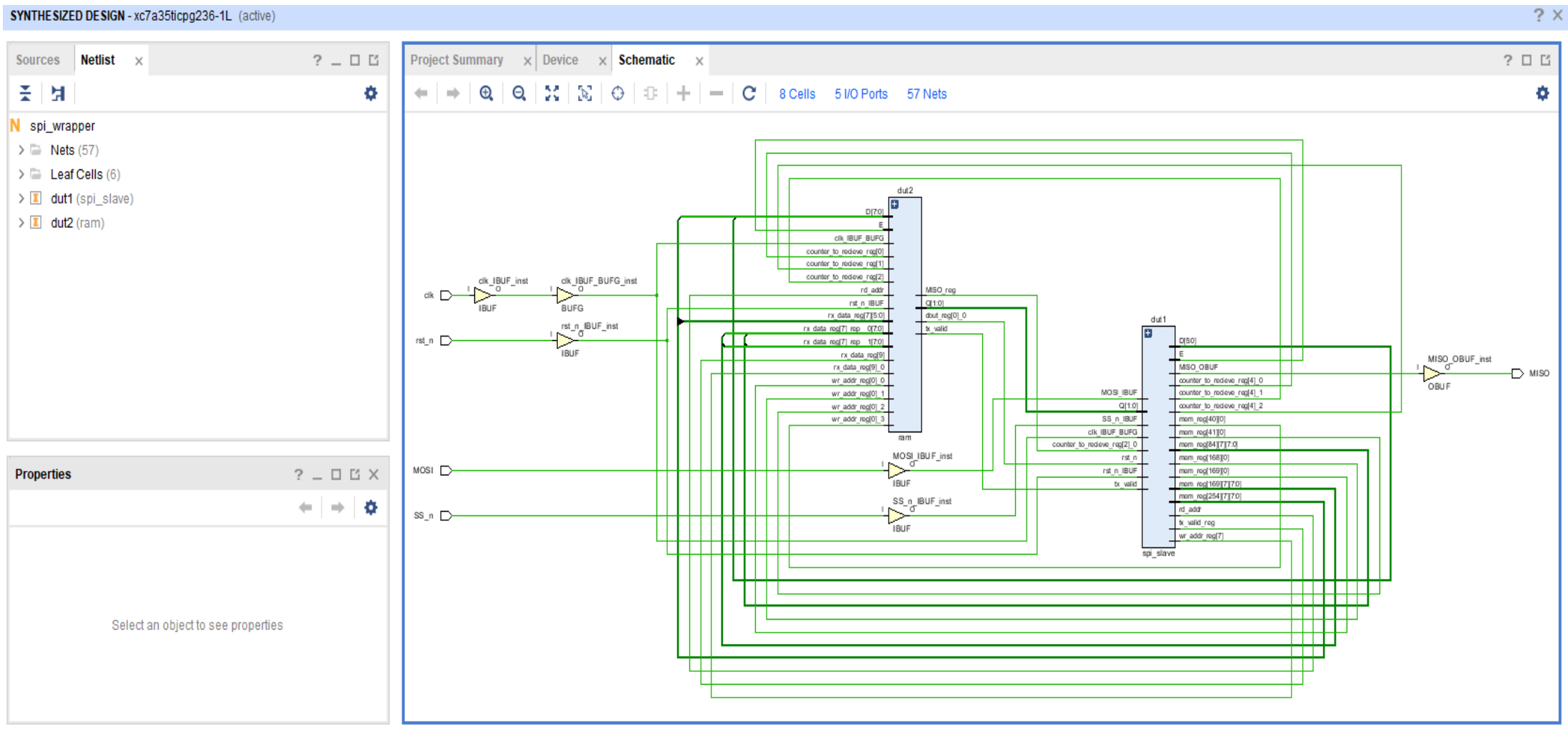
Timing Summary - impl_1 (saved)

Sequential :

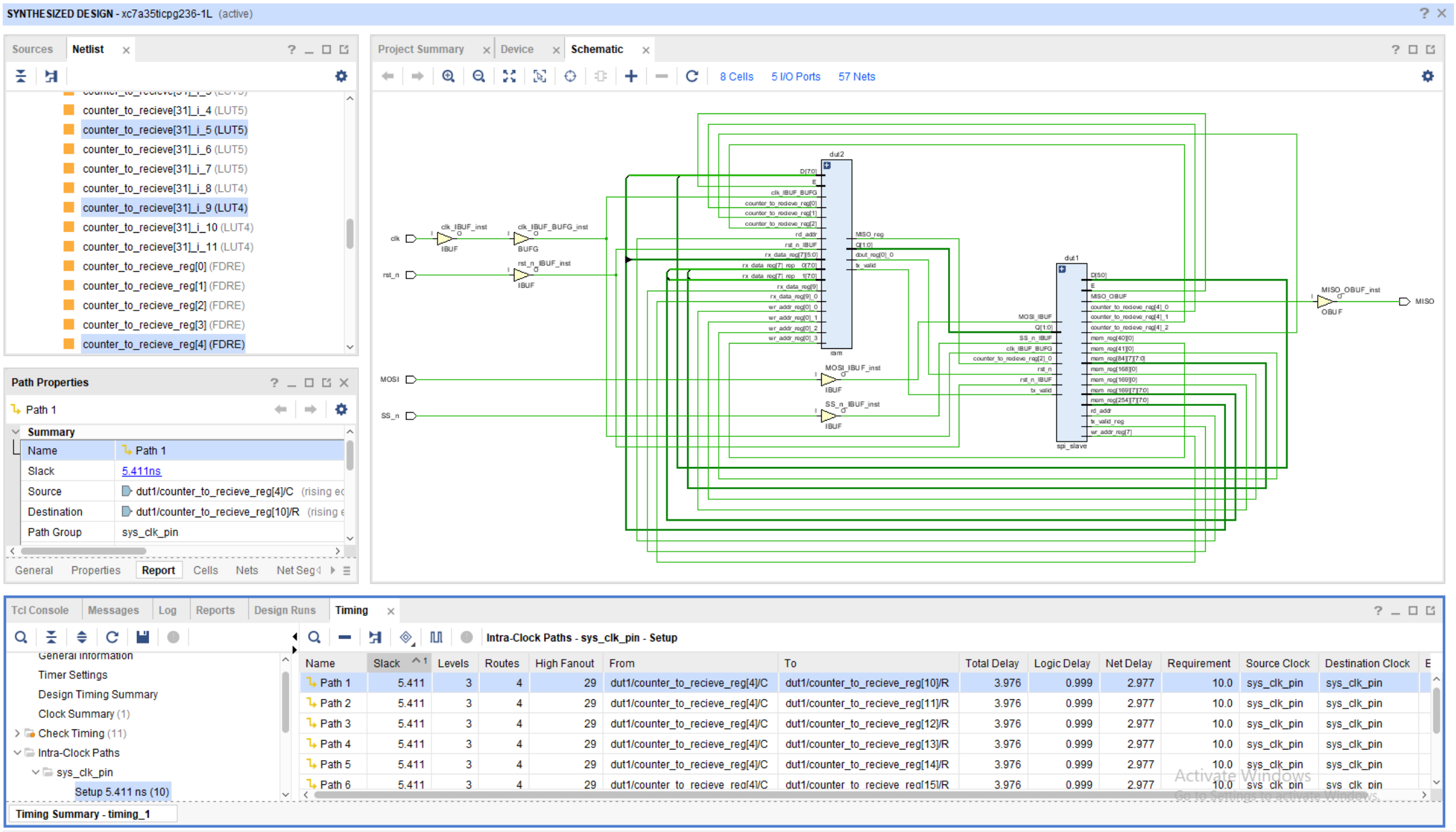
schematic after elaborated without error :



Schematic after synthesis with no error :

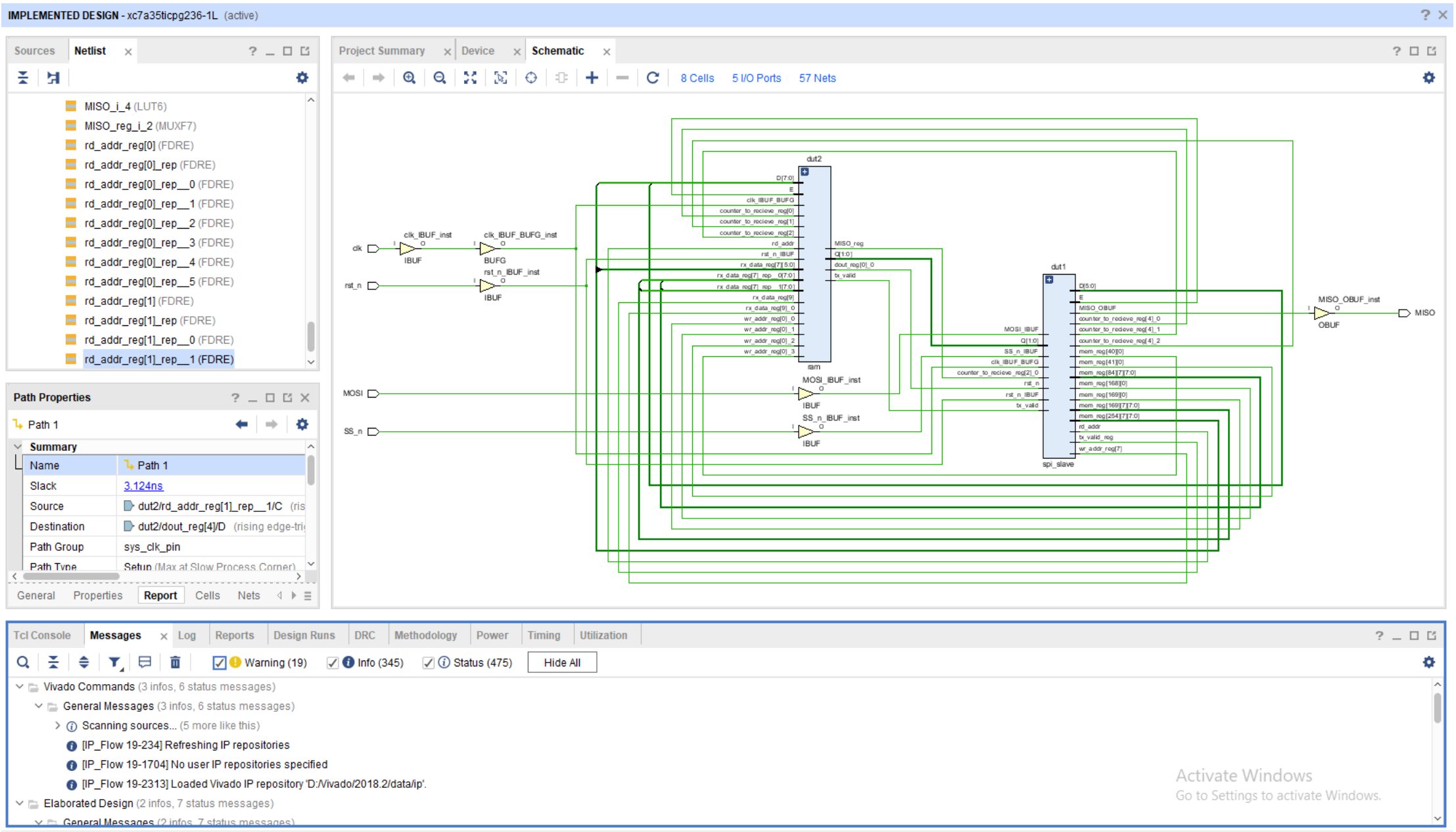


Time report after synthesis is same for all encoding :

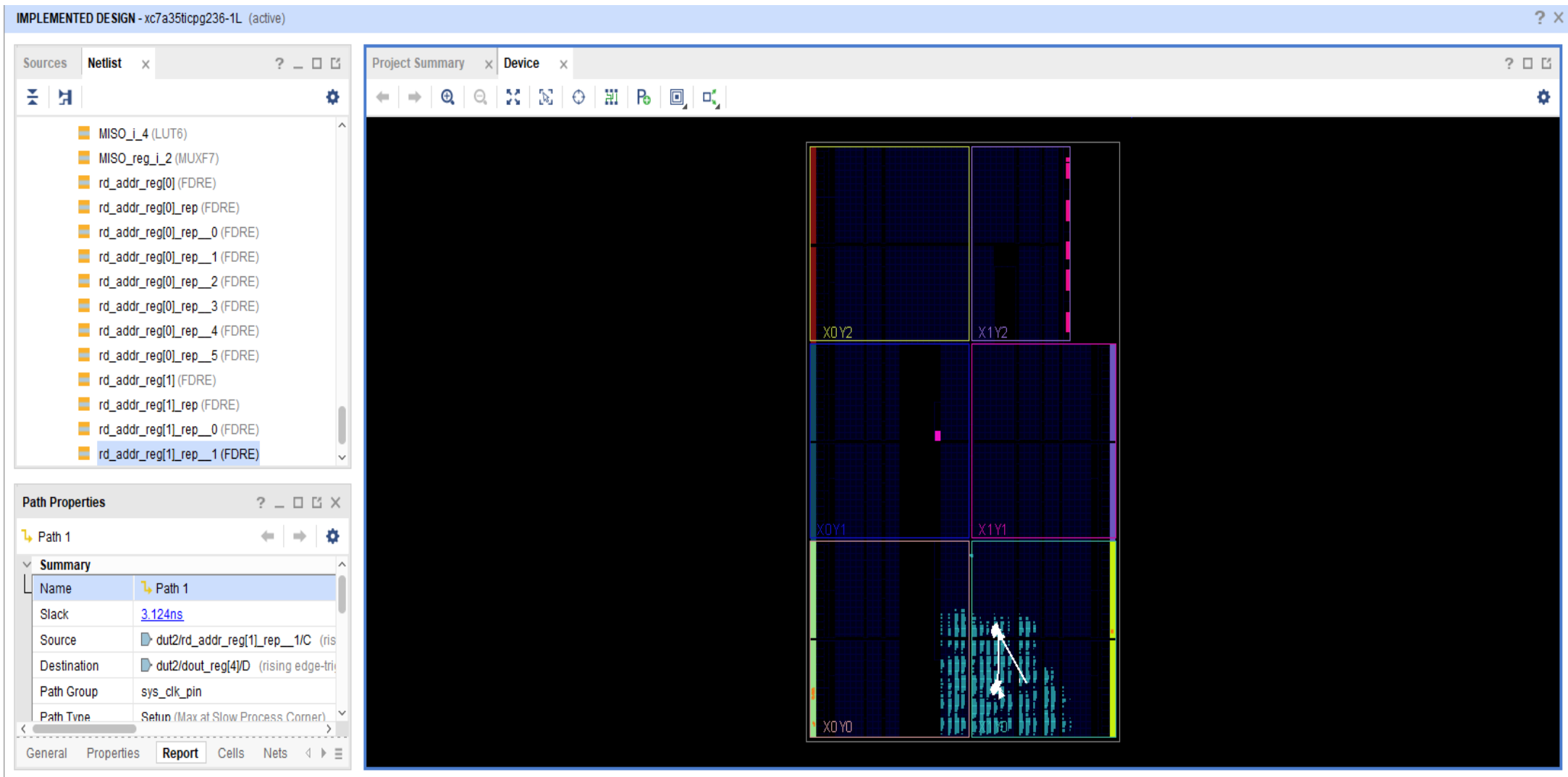


Sequential encoding :

Schematic after implementation without error :

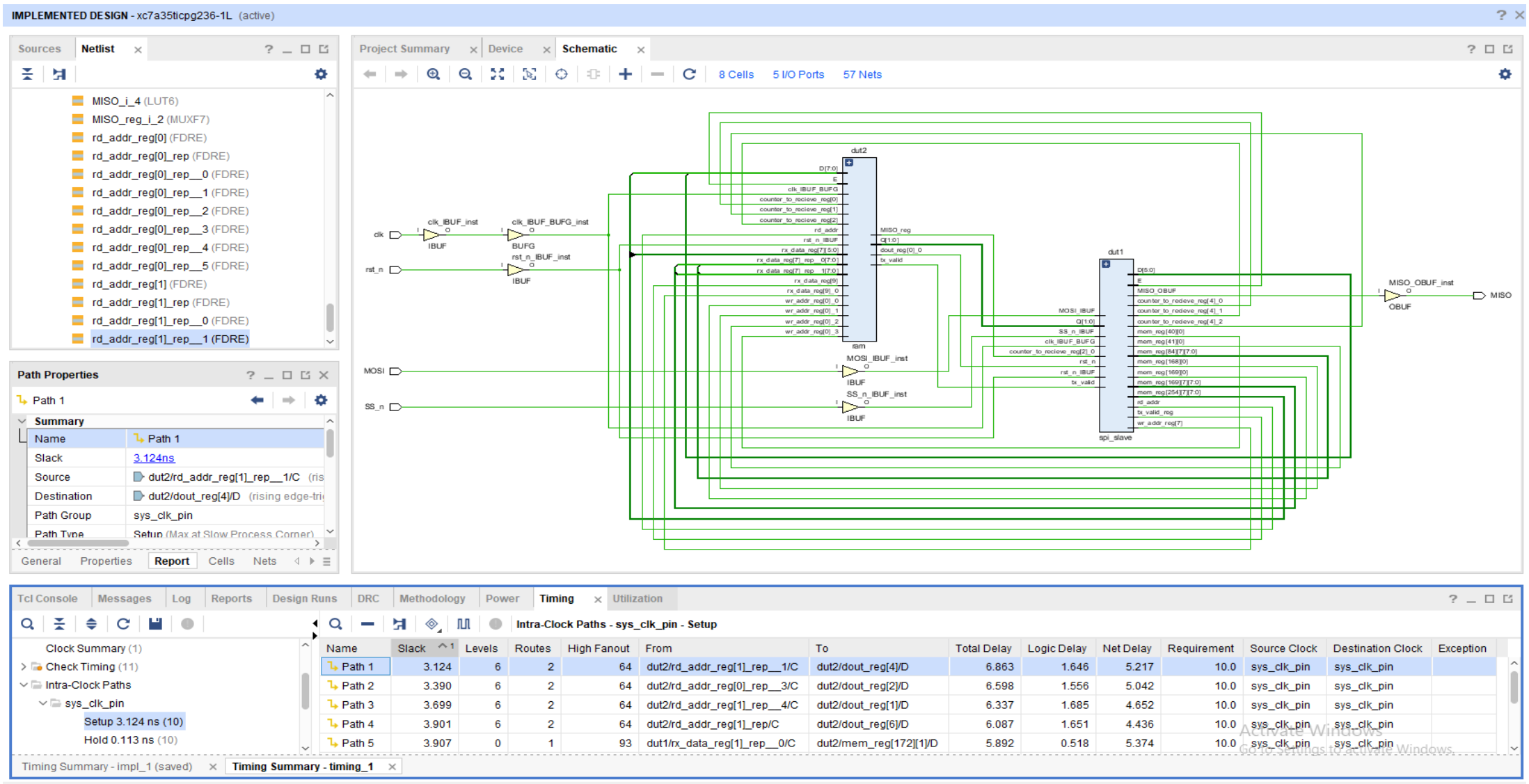


Device :

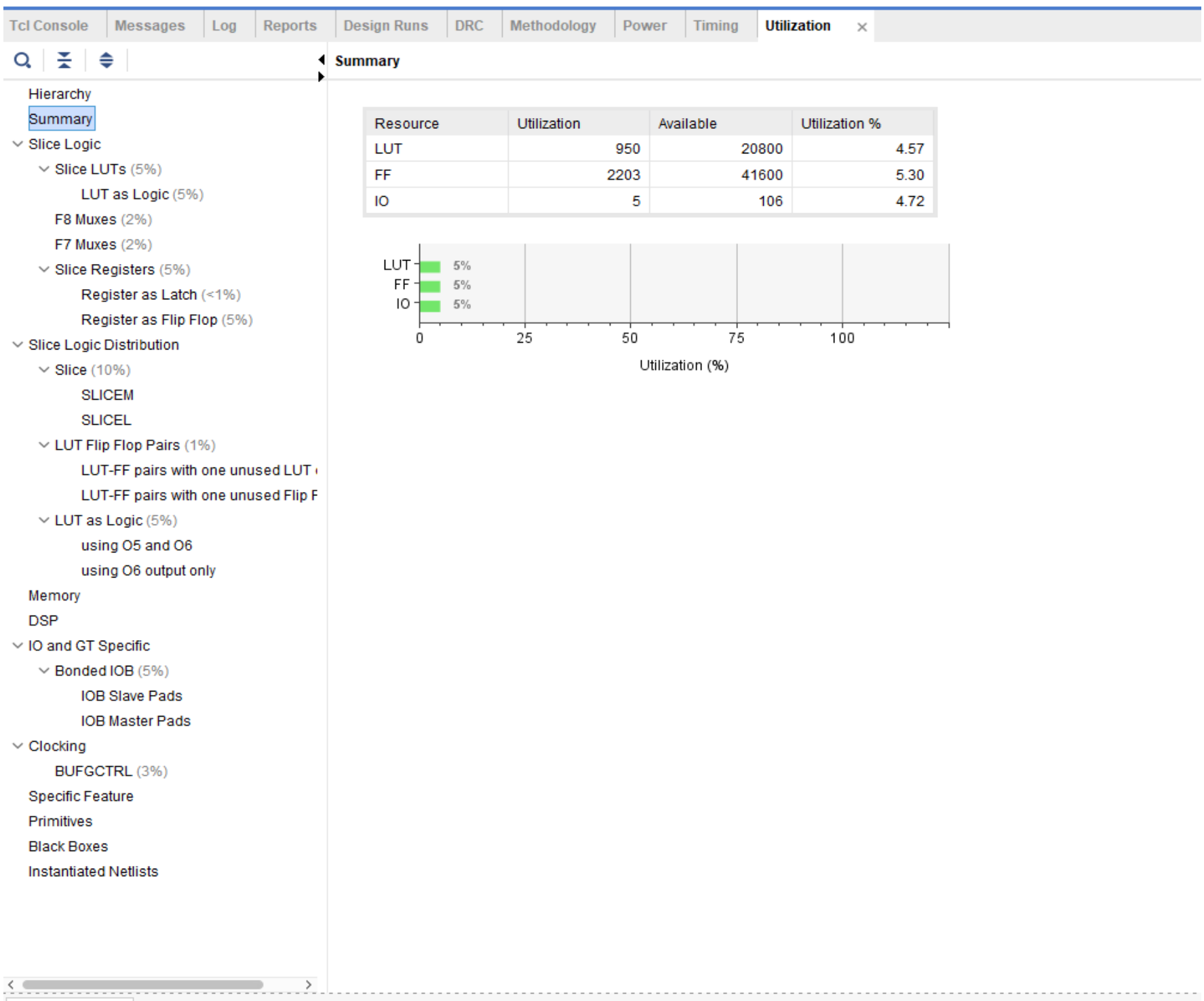


Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	3.124	6	64	dut2/rd_addr_reg[1]_rep__1/C	dut2/dout_reg[4]/D	6.863	1.646	5.217	10.0	sys_clk_pin	sys_clk_pin	
Path 2	3.390	6	64	dut2/rd_addr_reg[0]_rep__3/C	dut2/dout_reg[2]/D	6.598	1.556	5.042	10.0	sys_clk_pin	sys_clk_pin	
Path 3	3.699	6	64	dut2/rd_addr_reg[1]_rep__4/C	dut2/dout_reg[1]/D	6.337	1.685	4.652	10.0	sys_clk_pin	sys_clk_pin	
Path 4	3.901	6	64	dut2/rd_addr_reg[1]_rep/C	dut2/dout_reg[6]/D	6.087	1.651	4.436	10.0	sys_clk_pin	sys_clk_pin	
Path 5	3.907	0	93	dut1/rx_data_reg[1]_rep__0/C	dut2/mem_reg[172][1]/D	5.892	0.518	5.374	10.0	sys_clk_pin	sys_clk_pin	

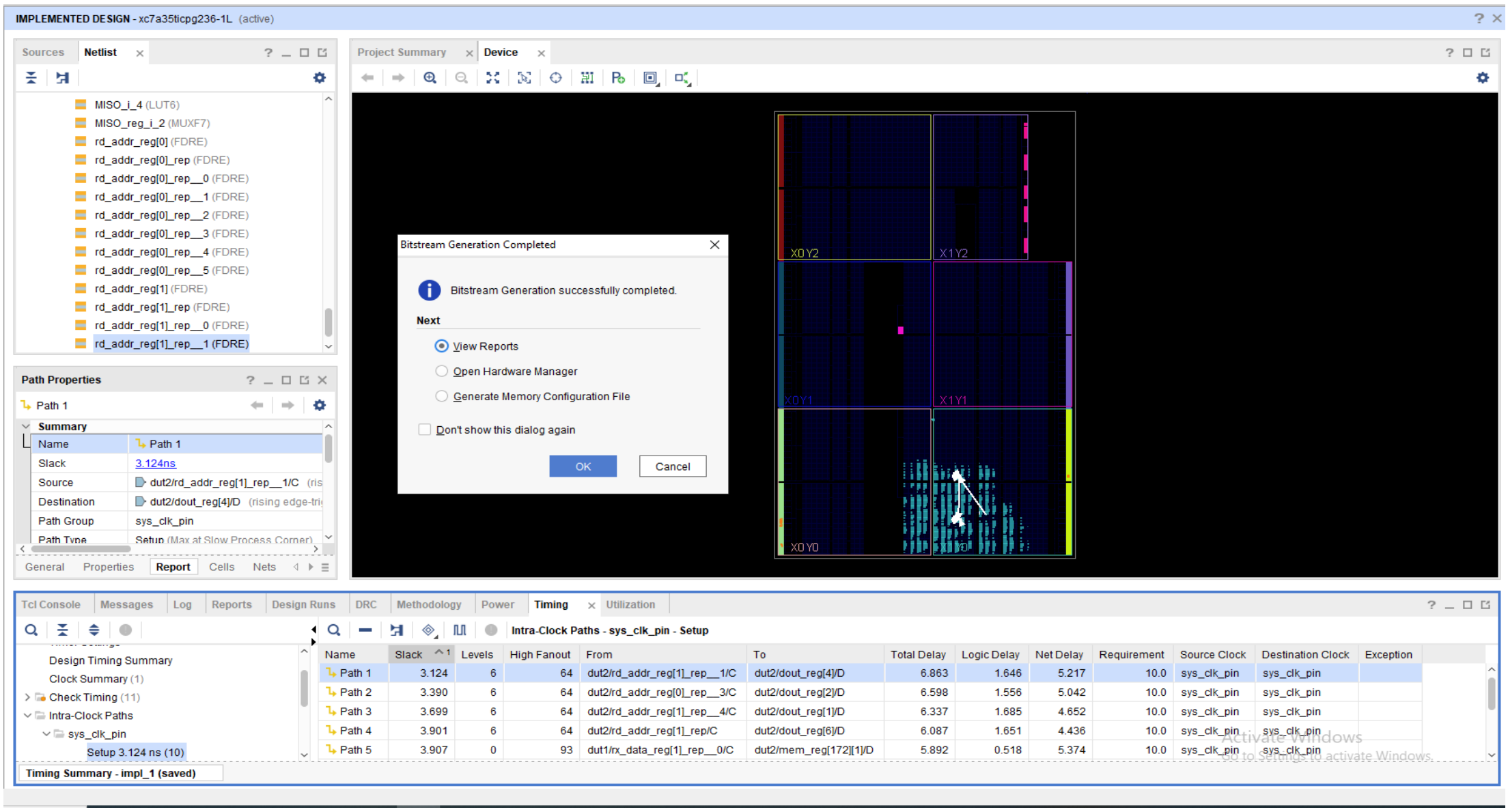
Worst slack for sequential :



Utilization report :



Generate bit successful :



Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	3.124	6	64	dut2/rd_addr_reg[1]_rep__1/C	dut2/dout_reg[4]/D	6.863	1.646	5.217	10.0	sys_clk_pin	sys_clk_pin	
Path 2	3.390	6	64	dut2/rd_addr_reg[0]_rep__3/C	dut2/dout_reg[2]/D	6.598	1.556	5.042	10.0	sys_clk_pin	sys_clk_pin	
Path 3	3.699	6	64	dut2/rd_addr_reg[1]_rep__4/C	dut2/dout_reg[1]/D	6.337	1.685	4.652	10.0	sys_clk_pin	sys_clk_pin	
Path 4	3.901	6	64	dut2/rd_addr_reg[1]_rep/C	dut2/dout_reg[6]/D	6.087	1.651	4.436	10.0	sys_clk_pin	sys_clk_pin	
Path 5	3.907	0	93	dut1/rx_data_reg[1]_rep__0/C	dut2/mem_reg[172][1]/D	5.892	0.518	5.374	10.0	sys_clk_pin	sys_clk_pin	