ASSIGNMENT 5 (EXTRA)

Abdalrahman Ali Eltaher

Q [1]

Type of errors:

1- Multiple drivers on out1

2- Possible division by zero

3- Bit-width mismatch between ns and in1

4- r1 is unused

5- Inconsistent reset logic

6- Use of 1'bx in synthesis

Q [2]

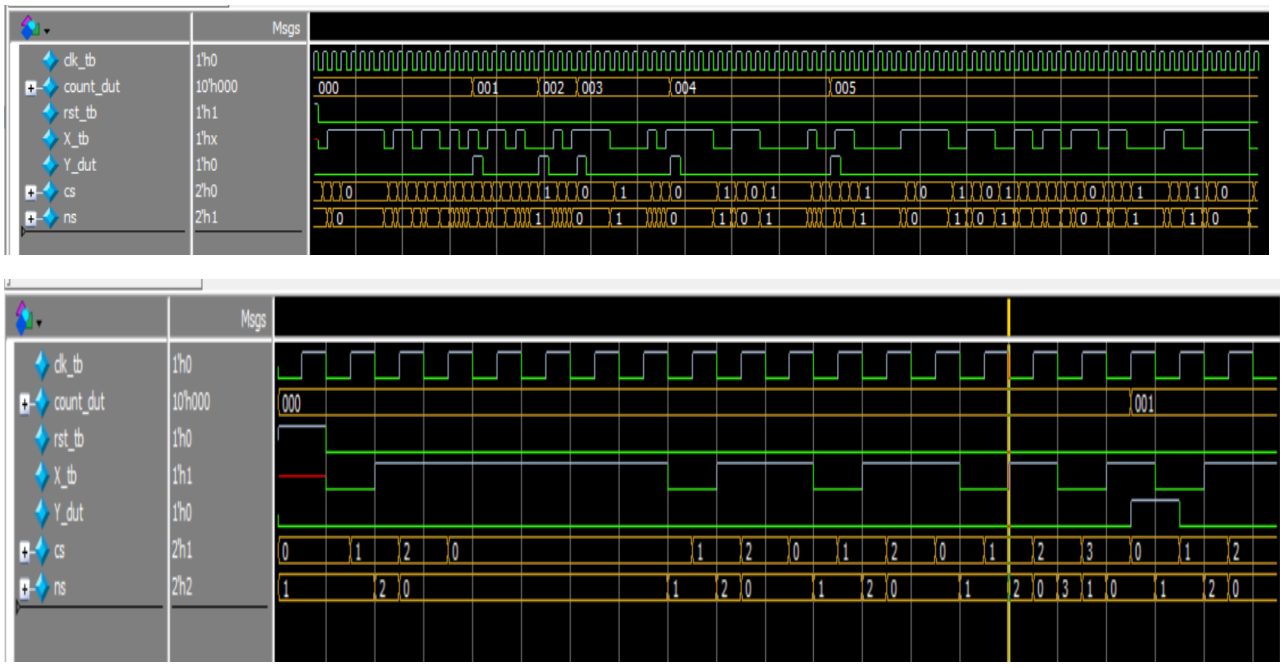
Code:

```
1   module detect_010(X,clk,rst,Y,count);
2
3   parameter IDLE =2'b00 ;
4   parameter ZERO = 2'b01 ;
5   parameter ONE = 2'b10 ;
6   parameter STORE = 2'b11 ;
7
8   input X,clk,rst;
9
10  output reg Y;
11  output reg [9:0] count;
12
13  //current state (cs) and next state(ns)
14  reg [1:0] cs , ns;
15
16  //next state
17  always @(*) begin
18      case (cs)
19          IDLE:
20              begin
21                  if(X)
22                      ns = IDLE;
23                  else
24                      ns = ZERO;
25              end
26          ZERO:
27              begin
28                  if(X)
29                      ns = ONE;
30                  else
31                      ns = ZERO;
32              end
33          ONE:
34              begin
35                  if(X)
36                      ns = IDLE;
37                  else
38                      ns = STORE;
39              end
40          STORE:
41              begin
42                  if(X)
43                      ns = IDLE;
44                  else
45                      ns = ZERO;
46              end
47          default: ns = IDLE ;
48      endcase
49  end
50
51  // state memory
52  always @(posedge clk or posedge rst ) begin
53      if(rst)
54          cs <= IDLE;
55      else
56          cs <= ns ;
57  end
58
59  // output logic
60  always @(posedge clk or posedge rst ) begin
61      if (rst)begin
62          count <= 0;
63          Y <= 0;
64      end
65      else begin
66          case (cs)
67              IDLE   : Y <= 0 ;
68              ZERO   : Y <= 0 ;
69              ONE    : Y <= 0 ;
70              STORE :
71                  begin
72                      Y <= 1 ;
73                      count <= count+1;
74                  end
75          endcase
76      end
77  end
78  endmodule
79
```

Testbench:

```
1   module detect_010_tb();
2
3   reg X_tb,clk_tb,rst_tb;
4
5   wire Y_dut;
6   wire [9:0] count_dut;
7
8   detect_010 dut(X_tb,clk_tb,rst_tb,Y_dut,count_dut);
9
10  //clock generation.
11  initial begin
12      clk_tb=0;
13      forever
14          #1 clk_tb = ~ clk_tb;
15  end
16  initial begin
17      //reset
18      rst_tb = 1 ;
19      @(negedge clk_tb);
20      rst_tb = 0 ;
21      //x = 0
22      repeat(100)begin
23          X_tb = $random;
24          @(negedge clk_tb);
25      end
26      $stop;
27  end
28  endmodule
```

Wave:



Q [3]

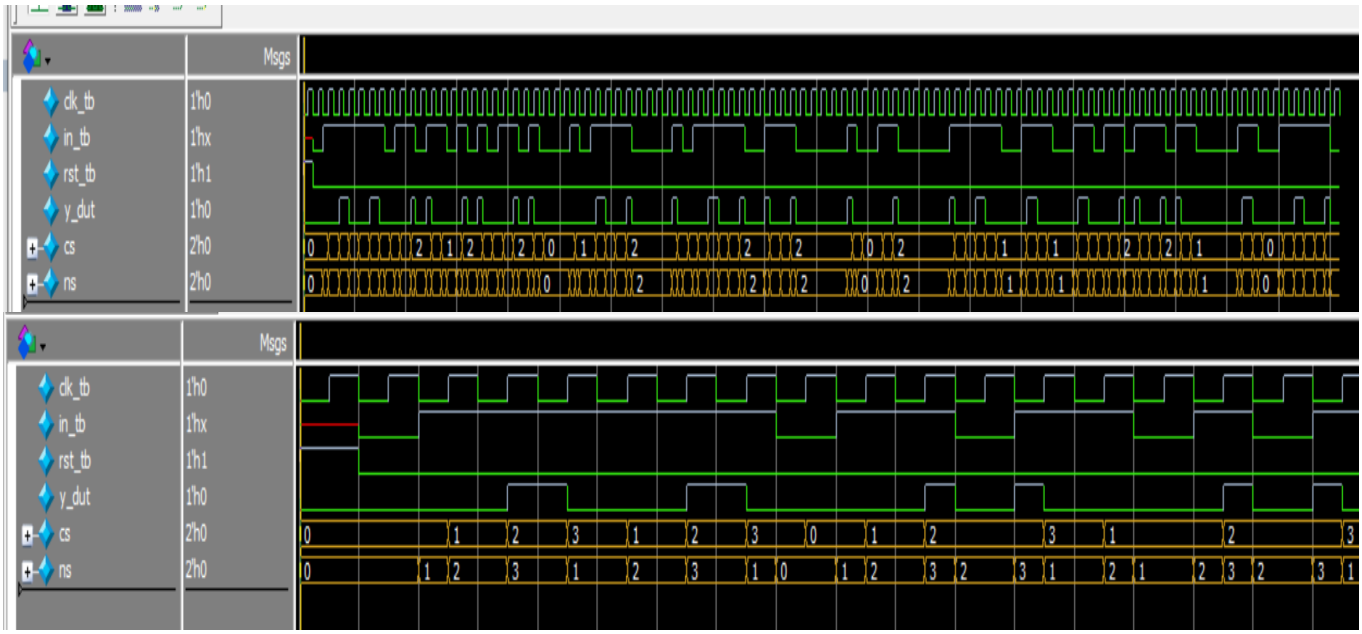
Code:

```
module seq_detector(clk,rst,in,y);

    parameter  S0 = 2'b00 ;
    parameter  S1 = 2'b01 ;
    parameter  S2 = 2'b10 ;
    parameter  S3 = 2'b11 ;

    input clk,rst,in;

    output y;

    //current state (cs) and next state (ns)
    reg [1:0] cs,ns;

    //next state
    always @(*) begin
        case (cs)
            S0:
                begin
                    if(in)
                        ns = S1;
                    else
                        ns = S0;
                end
            S1:
                begin
                    if(in)
                        ns = S2;
                    else
                        ns = S1;
                end
            S2:
                begin
                    if(in)
                        ns = S3;
                    else
                        ns = S2;
                end
            S3:
                begin
                    if(in)
                        ns = S1;
                    else
                        ns = S0;
                end
            default: ns =S0;
        endcase
    end

    //state memory.
    always @(posedge clk or posedge rst) begin
        if(rst)
            cs<=S0;
        else
            cs<=ns;
    end

    //output logic.
    assign y = (cs ==S2 && in )? 1:0;

endmodule
```

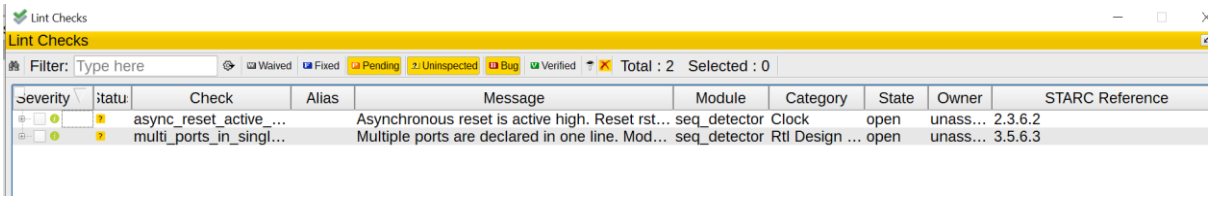
Testbench:

```
1   module seq_detector_tb();
2
3   reg clk_tb,rst_tb,in_tb;
4   wire  y_dut;
5   seq_detector dut (clk_tb,rst_tb,in_tb,y_dut);
6   //generate clock
7   initial begin
8       clk_tb=0;
9       forever
10      #1 clk_tb=~clk_tb;
11  end
12  initial begin
13      rst_tb = 1;
14      @(negedge clk_tb);
15      rst_tb = 0 ;
16      repeat (100)begin
17          in_tb = $random;
18          @(negedge clk_tb);
19      end
20      $stop;
21  end
22  endmodule
```

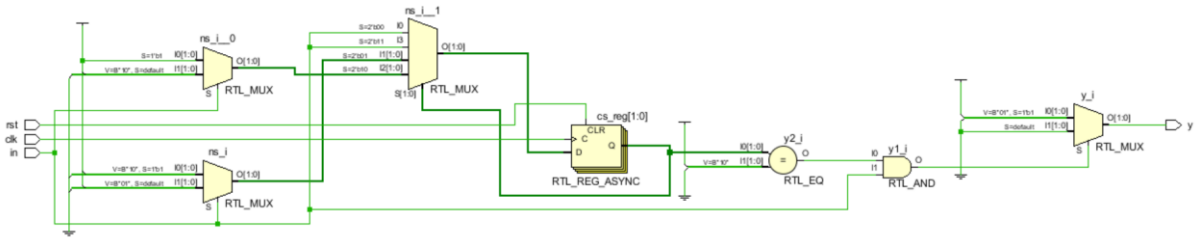
Wave:



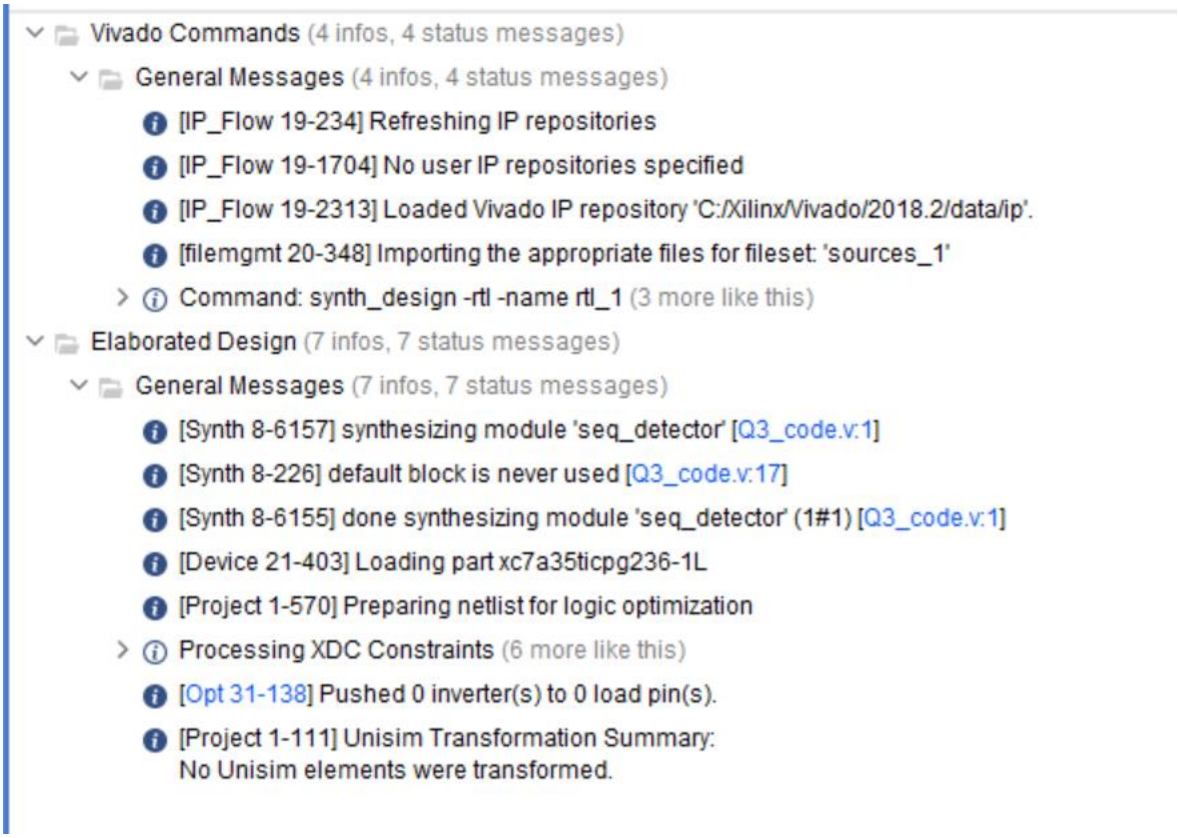
Linting snippet free from warnings or errors:



Snippets from the schematic after the elaboration:



Snippets from the message after the elaboration:



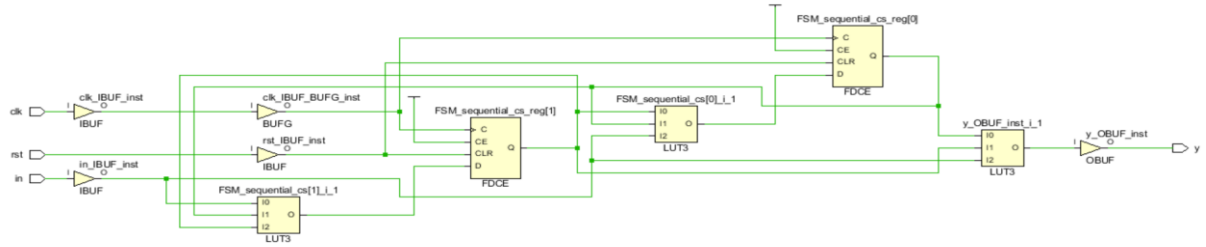
Snippets from the message after the synthesis:

Messages

Warning (1) Info (32) Status (19) Hide All

- Vivado Commands (4 infos, 6 status messages)
- Synthesis (1 warning, 22 infos, 11 status messages)
  - Command: synth_design -top seq_detector -part xc7a35ticpg236-1L (10 more like this)
  - [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
  - [Synth 8-6157] synthesizing module 'seq_detector' [Q3_code.v:1]
  - [Synth 8-226] default block is never used [Q3_code.v:17]
  - [Synth 8-6155] done synthesizing module 'seq_detector' (1#1) [Q3_code.v:1]
  - [Device 21-403] Loading part xc7a35ticpg236-1L
  - [Project 1-236] Implementation specific constraints were found while reading constraint file [E:/Vivado_session/project_6/project_6.srcs/constrs_1/imports/Assignment_5_Extra/const_Q3.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [.Xil/seq_detector_propImpl.xdc].
    Resolution: To avoid this warning, move constraints listed in ['Undefined'] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.
  - [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'seq_detector'
  - [Synth 8-5544] ROM "ns" won't be mapped to Block RAM because address size (1) smaller than threshold (5) (1 more like this)
  - [Synth 8-3354] encoded FSM with state register 'cs_reg' using encoding 'sequential' in module 'seq_detector'
  - [Project 1-571] Translating synthesized netlist
  - [Netlist 29-17] Analyzing 3 Unisim elements for replacement
  - [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
  - [Project 1-570] Preparing netlist for logic optimization (1 more like this)
  - [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
  - [Project 1-111] Unisim Transformation Summary:
    No Unisim elements were transformed. (1 more like this)
  - [Common 17-83] Releasing license: Synthesis
  - [Constraints 18-5210] No constraint will be written out.
  - [Common 17-1381] The checkpoint 'E:/Vivado_session/project_6/project_6.runs/synth_1/seq_detector.dcp' has been generated.
  - [runtcl-4] Executing : report_utilization -file seq_detector_utilization_synth.rpt -pb seq_detector_utilization_synth.pb
  - [Common 17-206] Exiting Vivado at Tue Aug 12 00:08:29 2025...
- Synthesized Design (6 infos, 2 status messages)

Snippets from the schematic after the synthesis:



Snippets from the utilization after the synthesis:

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
seq_detector	3	2	4	1

Snippets from the timing after the synthesis:

Design Timing Summary

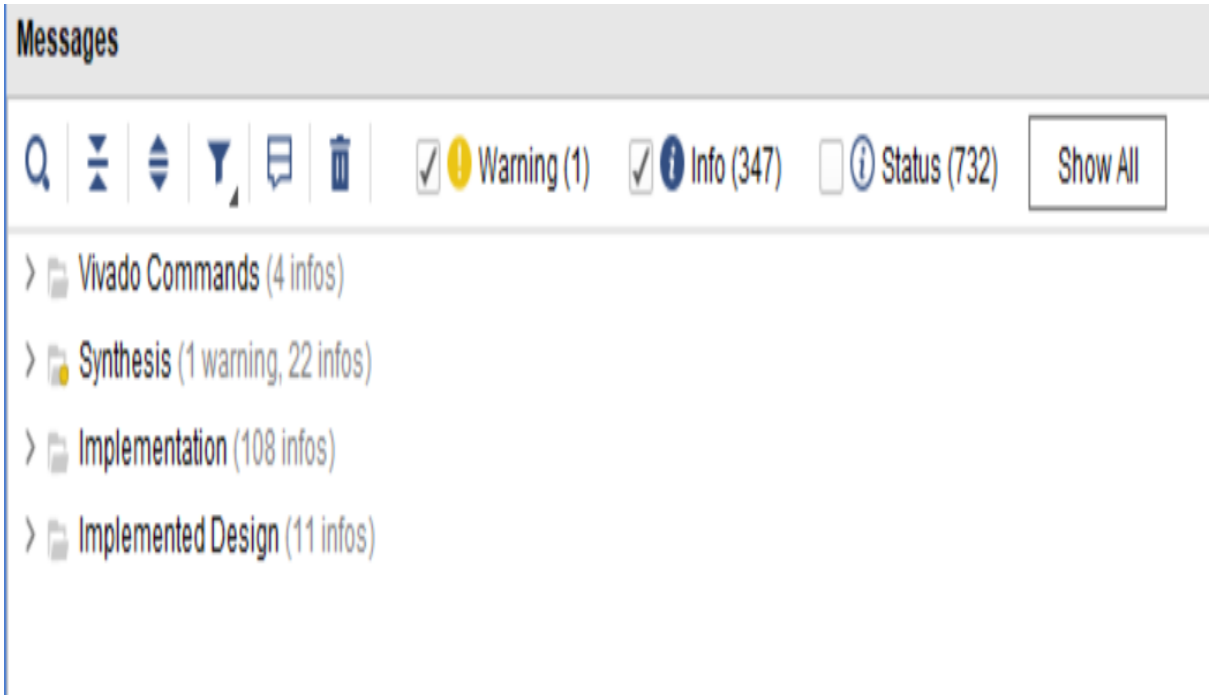
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.339 ns	Worst Hold Slack (WHS):	0.142 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	2	Total Number of Endpoints:	2	Total Number of Endpoints:	3

All user specified timing constraints are met.

Constraint 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 in]

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

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

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Snippet from the message after implementation:



Snippet from the utilization after implementation:

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
N seq_detector	3	2	1	3	2	4	1

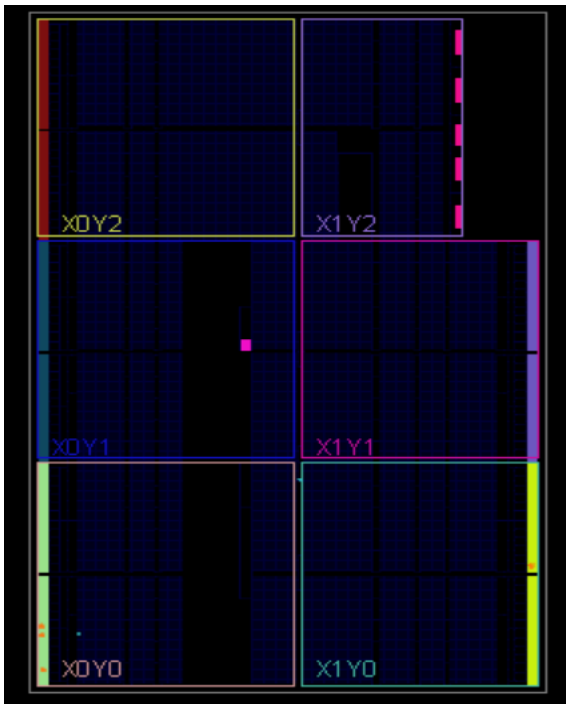
Snippet from the timing after implementation:

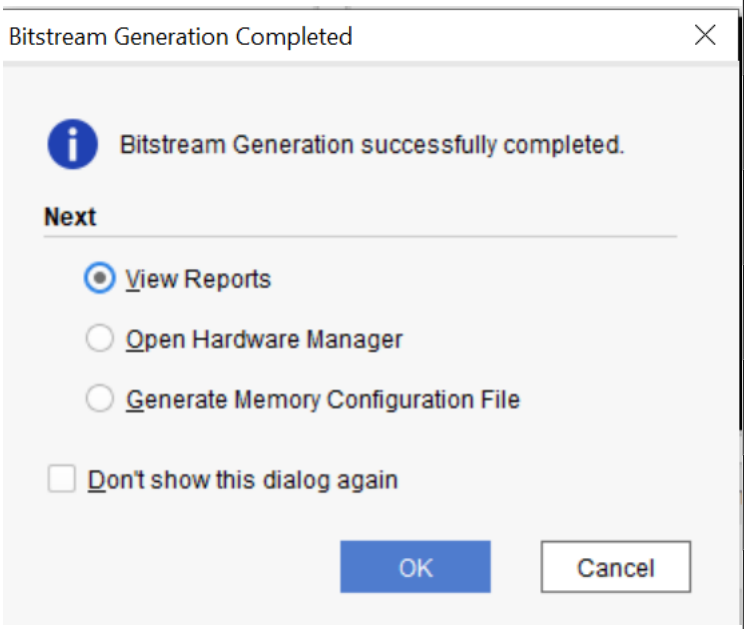
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.733 ns	Worst Hold Slack (WHS):	0.279 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:	2	Total Number of Endpoints:	2	Total Number of Endpoints:	3

All user specified timing constraints are met.

Snippet from the device after implementation:





Q [4]

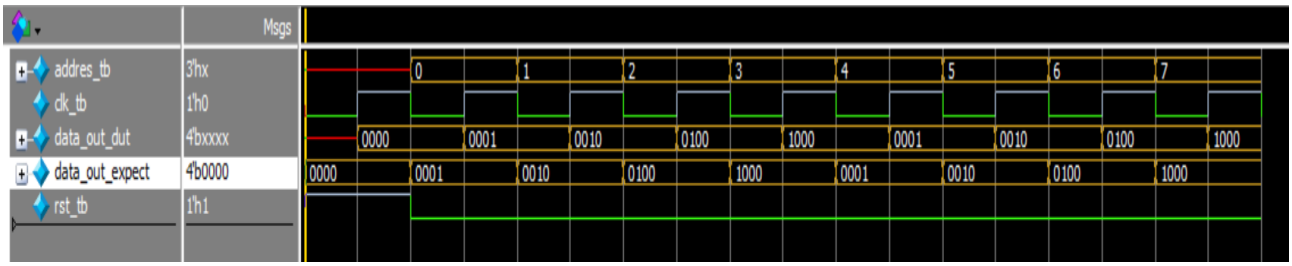
Code:

```
1  module rom_8x4_sync(clk,rst,addres,data_out);
2
3      input clk,rst;
4      input [2:0]addres ;
5
6      output reg [3:0]data_out;
7
8      always @(posedge clk ) begin
9          if(rst)
10             data_out<=0;
11         else begin
12             case (addres)
13                 0:data_out <= 4'b0001;
14                 1:data_out <= 4'b0010;
15                 2:data_out <= 4'b0100;
16                 3:data_out <= 4'b1000;
17                 4:data_out <= 4'b0001;
18                 5:data_out <= 4'b0010;
19                 6:data_out <= 4'b0100;
20                 7:data_out <= 4'b1000;
21                 default: data_out = 4'b0000;
22             endcase
23         end
24     end
25 endmodule
26
```

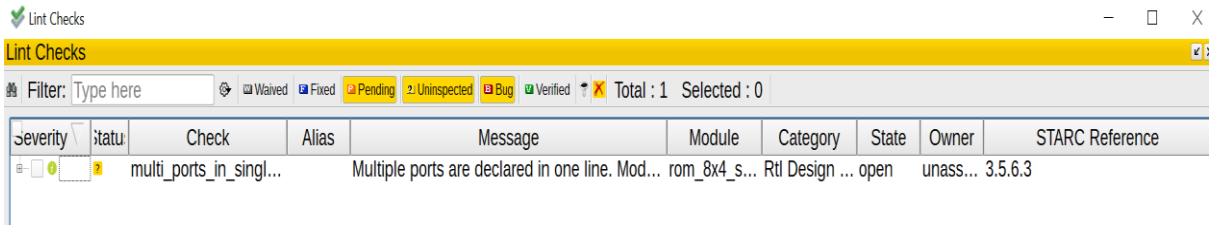
Testbench:

```
module rom_8x4_sync_tb();

    reg clk_tb,rst_tb;
    reg [2:0]addres_tb ;
    reg  [3:0]data_out_expect;

    wire  [3:0]data_out_dut;

    rom_8x4_sync dut(clk_tb,rst_tb,addres_tb,data_out_dut);
    initial begin
        clk_tb=0;
        forever
            #1 clk_tb=~clk_tb;
    end
    initial begin
        rst_tb = 1;data_out_expect=0;
        @(negedge clk_tb);
        rst_tb =0;

        //read from addres 0
        addres_tb = 0 ; data_out_expect = 4'b0001;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 1
        addres_tb = 1 ; data_out_expect = 4'b0010;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 2
        addres_tb = 2 ; data_out_expect = 4'b0100;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 3
        addres_tb = 3 ; data_out_expect = 4'b1000;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 4
        addres_tb = 4 ; data_out_expect = 4'b0001;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 5
        addres_tb = 5 ; data_out_expect = 4'b0010;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 6
        addres_tb = 6 ; data_out_expect = 4'b0100;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end

        //read from addres 7
        addres_tb = 7 ; data_out_expect = 4'b1000;
        @(negedge clk_tb )
        if(data_out_dut != data_out_expect)begin
            $display ("error : data_out_dut = %h,data_out_expect=%h",data_out_dut,data_out_expect);
            $stop;
        end
        $display ("testbench is correct ^_^");
        $stop;

    end
endmodule
```

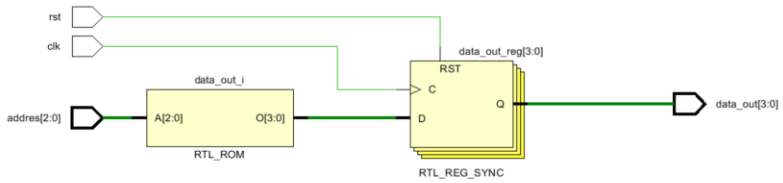
wave:



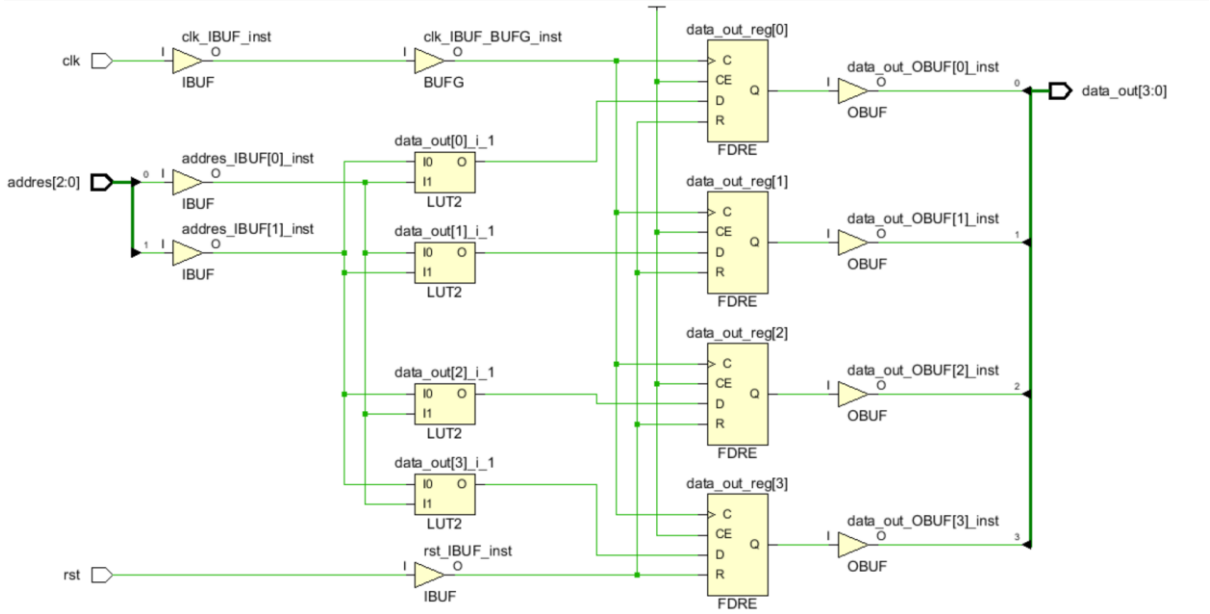
Linting snippet free from warnings or errors:



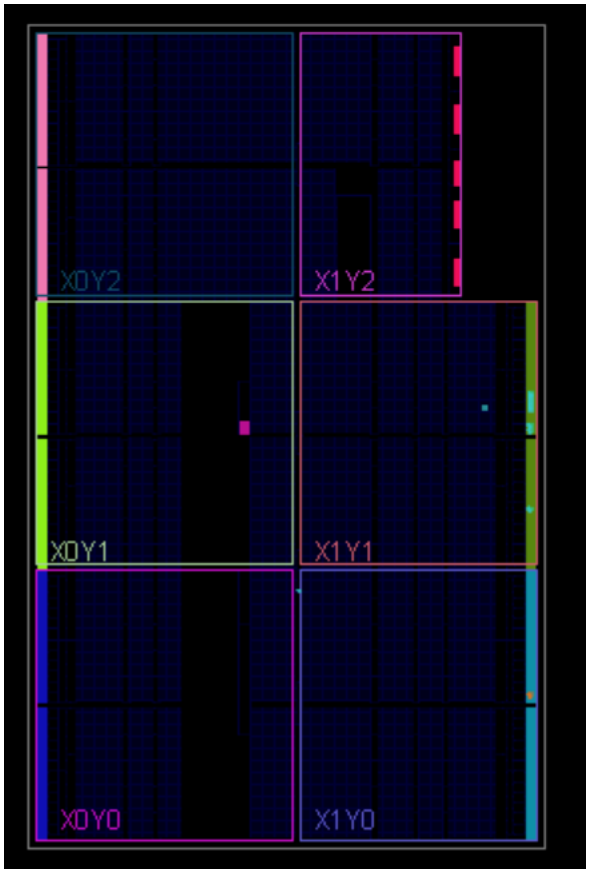
Snippets from the schematic after the synthesis:



Snippets from the schematic after the synthesis:



Snippet from the device after implementation:



Q [5]

Code:

```
1  module CRC_detect(clk,reset_n,data_in,data_valid,crc);
2
3      input clk,reset_n,data_in,data_valid;
4
5      output reg [3:0]  crc;
6
7      wire xor_out_1,xor_out_2;
8
9      assign xor_out_1 = data_in^crc[3];
10     assign xor_out_2 = crc[0]^crc[3] ;
11
12     always @(posedge clk or negedge reset_n) begin
13         if(~reset_n)
14             crc<=0;
15         else begin
16             if(data_valid) begin
17                 crc[0] <= xor_out_1 ;
18                 crc[1] <= xor_out_2 ;
19                 crc[2] <= crc[1];
20                 crc[3] <= crc[2];
21             end
22         end
23     end
24 endmodule
25
```

Testbench:

```
1   module CRC_detect_tb();
2
3   reg clk,reset_n,data_in,data_valid;
4   wire [3:0]crc_dut;
5   CRC_detect dut (clk,reset_n,data_in,data_valid,crc_dut);
6   initial begin
7       clk = 0;
8       forever
9           #1 clk = ~clk;
10  end
11  initial begin
12      //reset_n
13      reset_n = 0 ; data_valid = 1;
14      @(negedge clk);
15      reset_n = 1 ;
16
17      //insert message.
18     data_in = 1 ; @(negedge clk);
19     data_in = 0 ; @(negedge clk);
20     data_in = 1 ; @(negedge clk);
21     data_in = 1 ; @(negedge clk);
22     data_in = 0 ; @(negedge clk);
23     data_in = 0 ; @(negedge clk);
24     data_in = 1 ; @(negedge clk);
25     data_in = 0 ; @(negedge clk);
26     data_in = 0 ; @(negedge clk);
27     data_in = 0 ; @(negedge clk);
28     data_in = 0 ; @(negedge clk);
29     if (crc_dut != 4'b1010)begin
30         $display("error : crc_dut =%h",crc_dut);
31         $stop ;
32     end
33      $display("testbench is correct ^_^ ");
34      $stop ;
35  end
36  endmodule
```

Wave: