5. Question 5

5.1. Snippet after instantiation of IP modules

```
wire [5:0]adder_out,multiplier_out;
    /* Adder IP Instantiotion */
    generate
        if(FULL_ADDER = "ON")
           c_addsub_0 adder_mod (
             .A(A_r),             // input wire [2 : 0] A
             .B(B_r),             // input wire [2 : 0] B
             .C_IN(cin_r),        // input wire C_IN
             .S(adder_out)        // output wire [3 : 0] S
            );
        else if(FULL_ADDER = "OFF")
           c_addsub_0 adder_mod (
             .A(A_r),             // input wire [2 : 0] A
             .B(B_r),             // input wire [2 : 0] B
             .C_IN(cin_r),        // input wire C_IN
             .S(adder_out)        // output wire [3 : 0] S
            );
    endgenerate

    /* Multiplication IP Instantiotion */
    mult_gen_0 your_instance_name (
        .A(A_r),  // input wire [2 : 0] A
        .B(B_r),  // input wire [2 : 0] B
        .P(multiplier_out)  // output wire [5 : 0] P
    );

    /* always block for storing the inputs value to D-FF with rising edge of clock if rst is LOW */
    always @(posedge clk or posedge rst) begin : INPUTS_REGISTERATION
        if(rst)begin
            A_r <= 0;
            B_r <= 0;
            opcode_r <=0;
            cin_r    <= 0;
            serial_in_r <= 0;
            red_op_A_r  <= 0;
            red_op_B_r  <= 0;
            bypass_A_r  <= 0;
            bypass_B_r  <= 0;
            direction_r <= 0;
            out_r <= 0;
        end
        else begin
            A_r <= A;
            B_r <= B;
            opcode_r <= opcode;
            cin_r    <= cin;
            serial_in_r <= serial_in;
            red_op_A_r  <= red_op_A;
            red_op_B_r  <= red_op_B;
            bypass_A_r  <= bypass_A;
            bypass_B_r  <= bypass_B;
            direction_r <= direction;

        if(bypass_A_r && bypass_B_r)
            out_r <= First_priority;
        else if (bypass_A_r)
            out_r <= A_r;
        else if (bypass_B_r)
            out_r <= B_r;
        else if(~invalid_case)begin
        case (opcode_r)
            3'b000 : begin
                /* AND operation */
                if(red_op_A_r && red_op_B_r)    out_r <= &First_priority;
                else if(red_op_A_r)     out_r <= & A_r;
                else if(red_op_B_r)     out_r <= & B_r;
                else    out_r <= A_r & B_r;
            end
            3'b001 : begin
                /* XOR operation */
                if(red_op_A_r && red_op_B_r)    out_r <= ^First_priority;
                else if(red_op_A_r)     out_r <= ^A_r;
                else if(red_op_B_r)     out_r <= ^B_r;
                else out_r <= A_r ^ B_r;
            end
            3'b010 :begin
                /* addition */
                 out_r <= adder_out;
            end
```

```
            3'b011 : begin
                /* multiplication */
                out_r <= multiplier_out;
            end
            3'b100 : begin
                /* shift output by 1 */
                if(direction_r) begin
                    /* left shift */
                    out_r <= {out_r[4:0],serial_in_r};
                end
                else begin
                        /* right shift */
                    out_r <= {serial_in_r,out_r[5:1]};
                end
            end
            3'b101 :
                /* Rotate output by one */
                if(direction_r) begin
                    /* left rotation */
                    out_r <= {out_r[4:0],out_r[5]};
                end
                else begin
                    /* right rotation */
                    out_r <= {out_r[0],out_r[5:1]};
                end
        endcase
        end
        end

    end
```

5.2. 7-Segment Code

```
reg [3:0]  nibble1, nibble0; // divid the out_r[5:0] into two internal signal
 reg [1:0]  digit_select;     // first segment or second or ...
 reg [18:0] refresh_counter; // 19-bit counter for refreshing the display

 // Clock divider for refreshing the display at ~1kHz (assuming 100MHz input clock)
 always @(posedge clk or posedge rst) begin
    if (rst) begin
        refresh_counter <= 0;
        digit_select <= 0;

    end else begin
        refresh_counter <= refresh_counter + 1;
        digit_select <= refresh_counter[18:17]; // Digit select based on the refresh counter
    end
 end

 always @(*) begin : Seven_Segment
    nibble0 =out_r[3:0];
    nibble1 ={2'b00,out_r[5:4]};
    if(invalid_case)begin
        case(digit_select)
            2'b00:
                anode = 4'b1110;
                // activate LED1 and Deactivate LED2, LED3, LED4
                cathode = 7'b1001100; // "4"
            2'b01:
                anode = 4'b1101;
                // activate LED2 and Deactivate LED1, LED3, LED4
                cathode = 7'b0000001; // "0"
            2'b10: begin
                anode = 4'b1011;
                // activate LED3 and Deactivate LED2, LED1, LED4
                cathode = 7'b1001100; // "4"
            end
            2'b11: begin
                anode = 4'b0111;
                // activate LED4 and Deactivate LED2, LED3, LED1
                cathode = 7'b0110000; // "E"
            end
        endcase
```

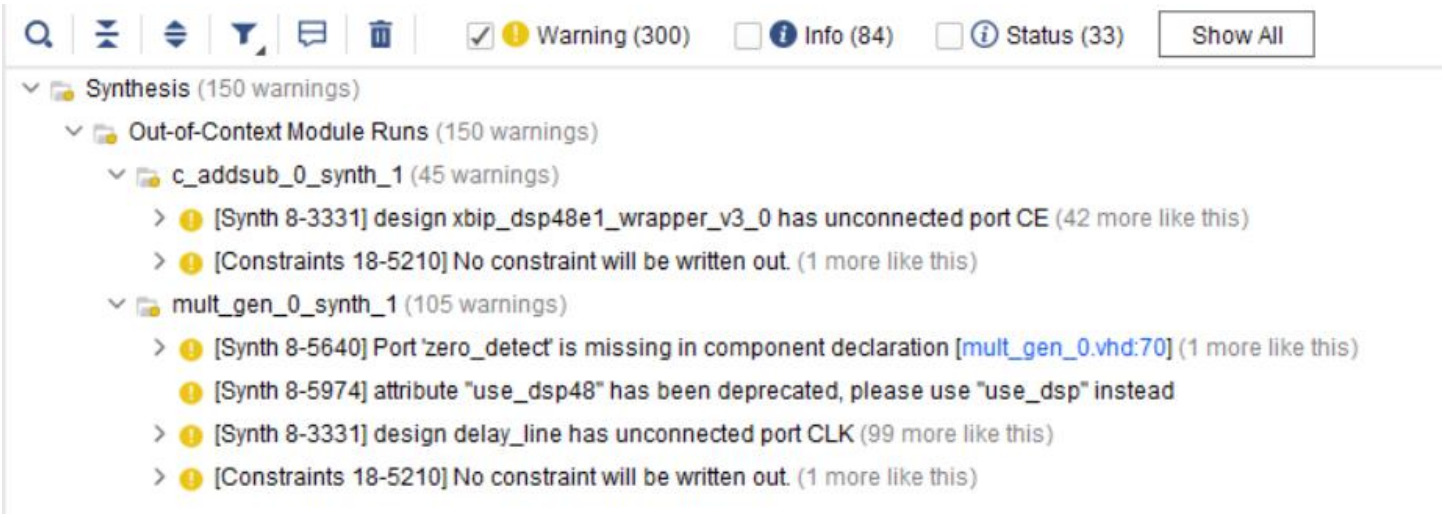
```
    end else begin
        case(digit_select)
            2'b00:
                anode = 4'b1110;
                // activate LED1 and Deactivate LED2, LED3, LED4
                case(nibble0)
                    4'b0000: cathode = 7'b0000001; // "0"
                    4'b0001: cathode = 7'b1001111; // "1"
                    4'b0010: cathode = 7'b0010010; // "2"
                    4'b0011: cathode = 7'b0000110; // "3"
                    4'b0100: cathode = 7'b1001100; // "4"
                    4'b0101: cathode = 7'b0100100; // "5"
                    4'b0110: cathode = 7'b0100000; // "6"
                    4'b0111: cathode = 7'b0001111; // "7"
                    4'b1000: cathode = 7'b0000000; // "8"
                    4'b1001: cathode = 7'b0000100; // "9"
                    4'b1010: cathode = 7'b0001000; // "A"
                    4'b1011: cathode = 7'b0000000; // "B"
                    4'b1100: cathode = 7'b0110001; // "C"
                    4'b1101: cathode = 7'b1000010; // "d"
                    4'b1110: cathode = 7'b0110000; // "E"
                    4'b1111: cathode = 7'b0111000; // "F"
                    default: cathode = 7'b1111110; // "-"
                endcase
            2'b01:
                anode = 4'b1101;
                // activate LED2 and Deactivate LED1, LED3, LED4
                case(nibble1)
                    4'b0000: cathode = 7'b0000001; // "0"
                    4'b0001: cathode = 7'b1001111; // "1"
                    4'b0010: cathode = 7'b0010010; // "2"
                    4'b0011: cathode = 7'b0000110; // "3"
                    4'b0100: cathode = 7'b1001100; // "4"
                    4'b0101: cathode = 7'b0100100; // "5"
                    4'b0110: cathode = 7'b0100000; // "6"
                    4'b0111: cathode = 7'b0001111; // "7"
                    4'b1000: cathode = 7'b0000000; // "8"
                    4'b1001: cathode = 7'b0000100; // "9"
                    4'b1010: cathode = 7'b0001000; // "A"
                    4'b1011: cathode = 7'b0000000; // "B"
                    4'b1100: cathode = 7'b0110001; // "C"
                    4'b1101: cathode = 7'b1000010; // "d"
                    4'b1110: cathode = 7'b0110000; // "E"
                    4'b1111: cathode = 7'b0111000; // "F"
                    default: cathode = 7'b1111110; // "-"
                endcase
            2'b10: begin
                anode = 4'b1011;
                // activate LED3 and Deactivate LED2, LED1, LED4
                cathode = 7'b1111110; // "-"
            end
            2'b11: begin
                anode = 4'b0111;
                // activate LED4 and Deactivate LED2, LED3, LED1
                cathode = 7'b1111110; // "-"
            end
        endcase
    end
 end
endmodule
```

5.3. Constrains File

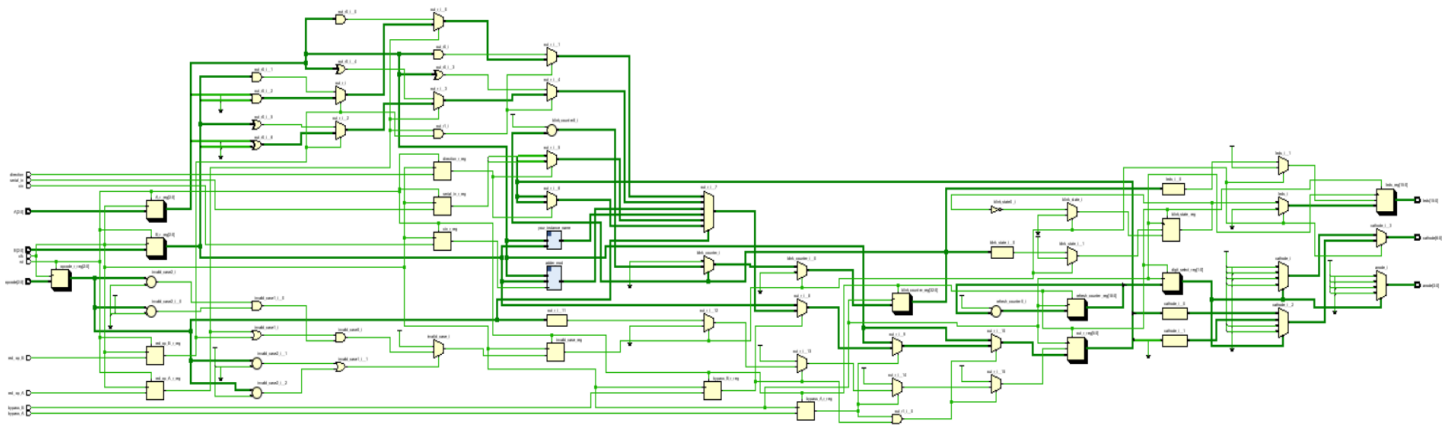
```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

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

## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {opcode[0]}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {opcode[1]}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {opcode[2]}]
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 {leds[0]}]
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]}]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {leds[7]}]
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {leds[8]}]
set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports {leds[9]}]
set_property -dict { PACKAGE_PIN W3    IOSTANDARD LVCMOS33 } [get_ports {leds[10]}]
set_property -dict { PACKAGE_PIN U3    IOSTANDARD LVCMOS33 } [get_ports {leds[11]}]
set_property -dict { PACKAGE_PIN P3    IOSTANDARD LVCMOS33 } [get_ports {leds[12]}]
set_property -dict { PACKAGE_PIN N3    IOSTANDARD LVCMOS33 } [get_ports {leds[13]}]
set_property -dict { PACKAGE_PIN P1    IOSTANDARD LVCMOS33 } [get_ports {leds[14]}]
set_property -dict { PACKAGE_PIN L1    IOSTANDARD LVCMOS33 } [get_ports {leds[15]}]


##7 Segment Display
set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {cathode[0]}]
set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {cathode[1]}]
set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {cathode[2]}]
set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {cathode[3]}]
set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {cathode[4]}]
set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {cathode[5]}]
set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {cathode[6]}]

#set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports dp]

set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {anode[0]}]
set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {anode[1]}]
set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {anode[2]}]
set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {anode[3]}]


##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]
```

5.4. Elaboration

5.4.1. Messages Tab

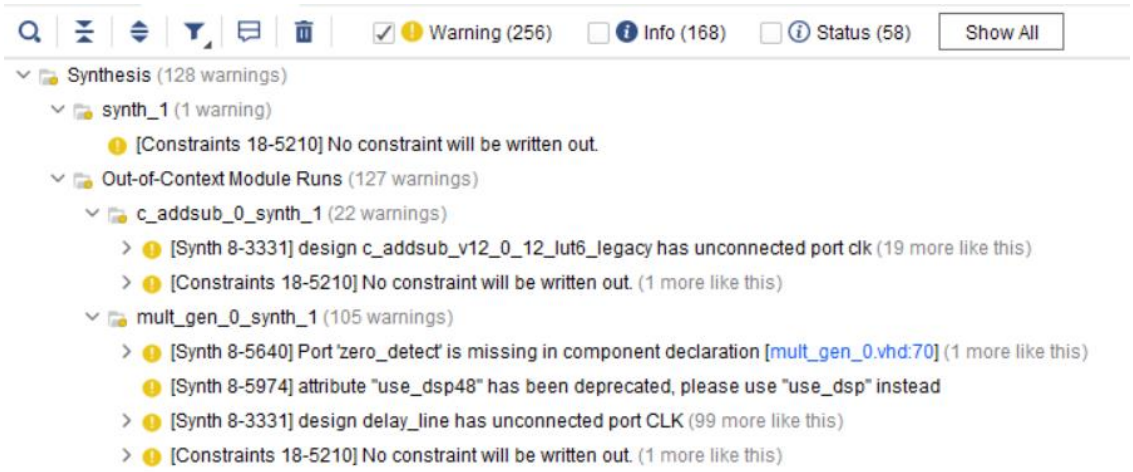


5.4.2. Schematic

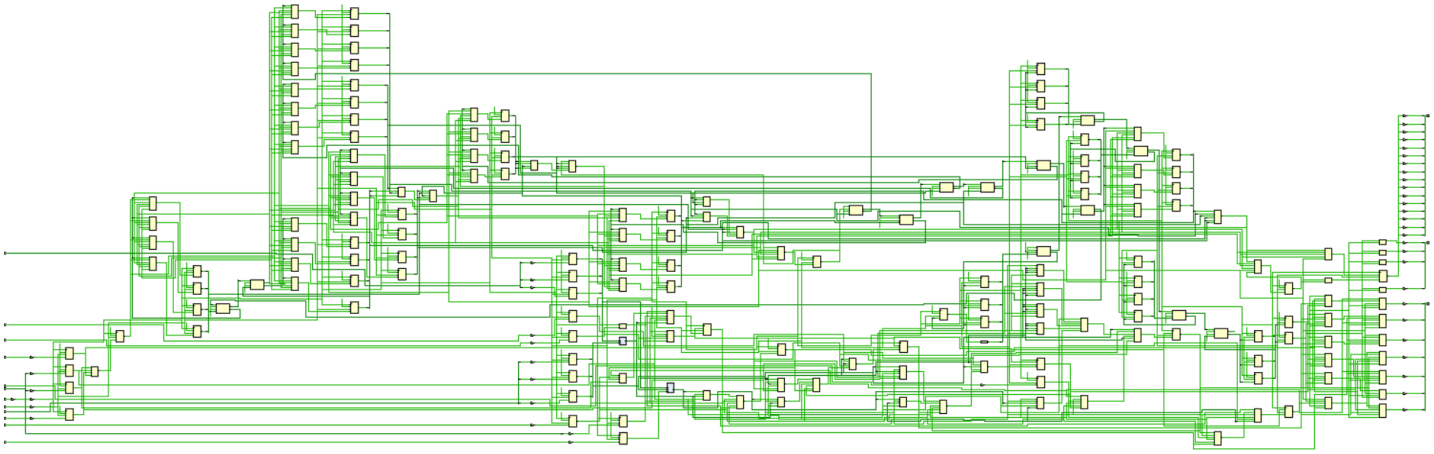


5.5. Synthesis

5.5.1. Messages Tab



5.5.2. Schematic



5.5.3. Report Timing Summary

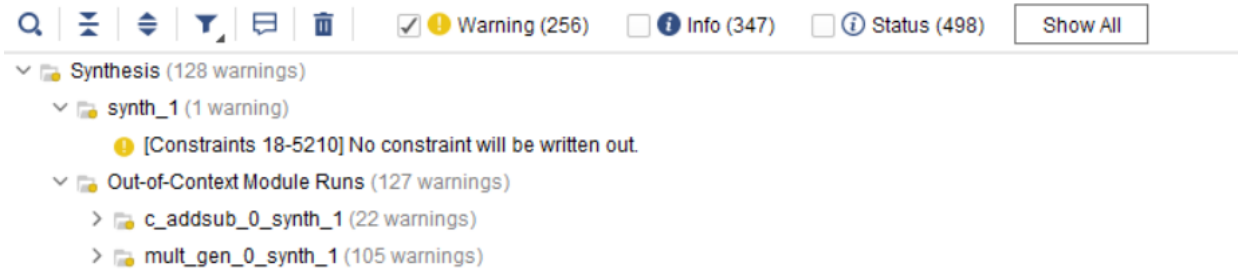
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.507 ns	Worst Hold Slack (WHS):	0.131 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:	70	Total Number of Endpoints:	70	Total Number of Endpoints:	80

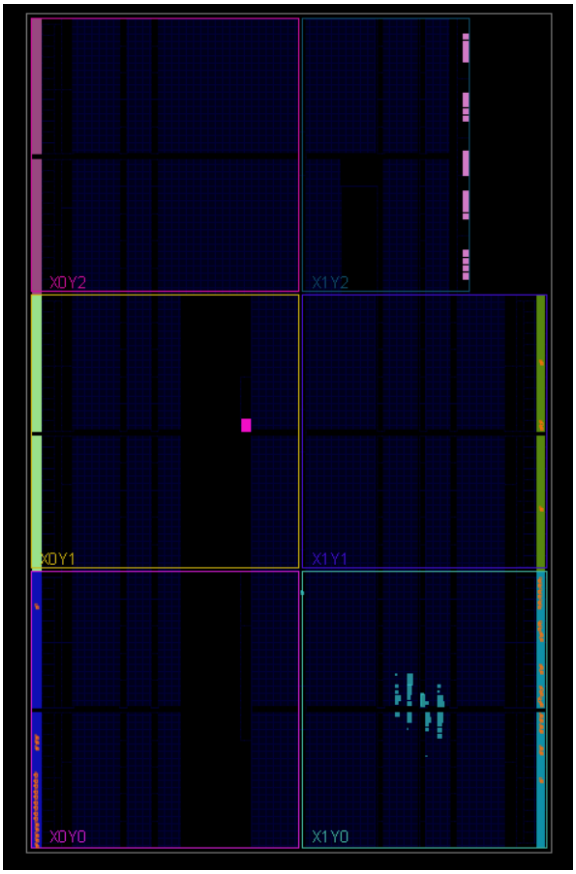
All user specified timing constraints are met.

5.6. Implementation

5.6.1. Messages Tab



5.6.2. Device



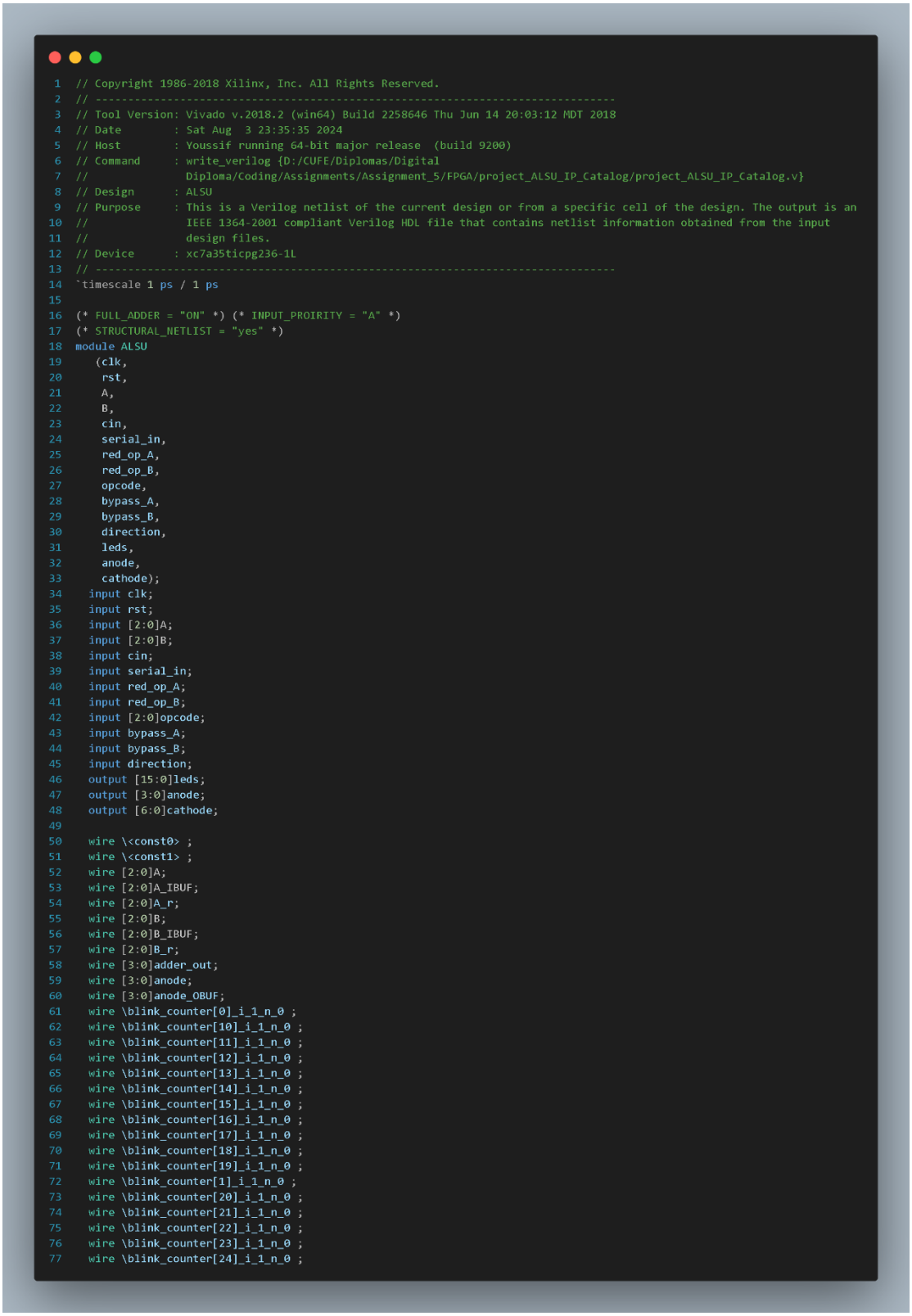
5.6.3. Report Timing Summary

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.112 ns	Worst Hold Slack (WHS):	0.197 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:	100	Total Number of Endpoints:	100	Total Number of Endpoints:	95

All user specified timing constraints are met.

5.7. Snippet from the Netlist file



```
// Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
// ----------------------------------------------------------------------------
// Tool Version: Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14 20:03:12 MDT 2018
// Date        : Sat Aug  3 23:35:35 2024
// Host        : Youssif running 64-bit major release  (build 9200)
// Command     : write_verilog {D:/CUFE/Diplomas/Digital
//               Diploma/Coding/Assignments/Assignment_5/FPGA/project_ALSU_IP_Catalog/project_ALSU_IP_Catalog.v}
// Design      : ALSU
// Purpose     : This is a Verilog netlist of the current design or from a specific cell of the design. The output is an
//               IEEE 1364-2001 compliant Verilog HDL file that contains netlist information obtained from the input
//               design files.
// Device      : xc7a35ticpg236-1L
// ----------------------------------------------------------------------------
`timescale 1 ps / 1 ps

(* FULL_ADDER = "ON" *) (* INPUT_PROIRITY = "A" *)
(* STRUCTURAL_NETLIST = "yes" *)
module ALSU
   (clk,
    rst,
    A,
    B,
    cin,
    serial_in,
    red_op_A,
    red_op_B,
    opcode,
    bypass_A,
    bypass_B,
    direction,
    leds,
    anode,
    cathode);
  input clk;
  input rst;
  input [2:0]A;
  input [2:0]B;
  input cin;
  input serial_in;
  input red_op_A;
  input red_op_B;
  input [2:0]opcode;
  input bypass_A;
  input bypass_B;
  input direction;
  output [15:0]leds;
  output [3:0]anode;
  output [6:0]cathode;

  wire \<const0> ;
  wire \<const1> ;
  wire [2:0]A;
  wire [2:0]A_IBUF;
  wire [2:0]A_r;
  wire [2:0]B;
  wire [2:0]B_IBUF;
  wire [2:0]B_r;
  wire [3:0]adder_out;
  wire [3:0]anode;
  wire [3:0]anode_OBUF;
  wire \blink_counter[0]_i_1_n_0 ;
  wire \blink_counter[10]_i_1_n_0 ;
  wire \blink_counter[11]_i_1_n_0 ;
  wire \blink_counter[12]_i_1_n_0 ;
  wire \blink_counter[13]_i_1_n_0 ;
  wire \blink_counter[14]_i_1_n_0 ;
  wire \blink_counter[15]_i_1_n_0 ;
  wire \blink_counter[16]_i_1_n_0 ;
  wire \blink_counter[17]_i_1_n_0 ;
  wire \blink_counter[18]_i_1_n_0 ;
  wire \blink_counter[19]_i_1_n_0 ;
  wire \blink_counter[1]_i_1_n_0 ;
  wire \blink_counter[20]_i_1_n_0 ;
  wire \blink_counter[21]_i_1_n_0 ;
  wire \blink_counter[22]_i_1_n_0 ;
  wire \blink_counter[23]_i_1_n_0 ;
  wire \blink_counter[24]_i_1_n_0 ;
```

5.8. Utilization Report summary

5.8.1. Hierarchy

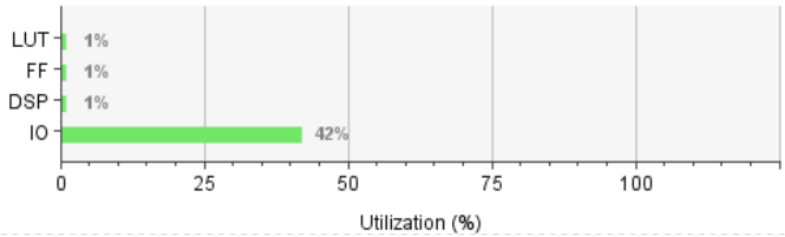
Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	DSPs (90)	Bonded IOB (106)	BUFGCTRL (32)
ALSU	83	94	47	83	42	1	45	1
adder_mod (c_addsub...	3	0	1	3	0	0	0	0
mult_mod (mult_gen_0)	0	0	0	0	0	1	0	0

5.8.2. Summary

Summary

Resource	Utilization	Available	Utilization %
LUT	83	20800	0.40
FF	94	41600	0.23
DSP	1	90	1.11
IO	45	106	42.45



5.9. Bitstream generation

Write Bitstream (write_bitstream)				
impl_1_bitstream_report_webtalk_0	Webtalk Report		8/3/24 10:33 PM	25.6 KB
impl_1_bitstream_implementation_log_0	Vivado Implementation Log		8/3/24 10:34 PM	29.4 KB