Abdelrahman Ahmed Sayed

TASK 1

Design:

```
module ALSU(
    input [2:0] A_, B_,
    input [2:0] opcode_,
    input cin_,
    input serial_in_,
    input direction_, // 1 is left
    input red_op_A_, red_op_B_, bypass_A_, bypass_B_,
    input clk, rst,
    output reg [5:0] out,
    output reg [15:0] leds
);

parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";

reg [2:0] A, B;
reg [2:0] opcode;
reg cin;
reg serial_in;
reg direction; // 1 is left
reg red_op_A, red_op_B, bypass_A, bypass_B;

always @(posedge clk or posedge rst) begin
    if (rst) begin
        A <= 0;
        B <= 0;
        opcode <= 0;
        cin <= 0;
        serial_in <= 0;
        direction <= 0;
        red_op_A <= 0;
        red_op_B <= 0;
        bypass_A <= 0;
        bypass_B <= 0;
    end
    else begin
        A <= A_;
        B <= B_;
        opcode <= opcode_;
```

objects Ln 13, Col 32 Spaces: 4 UTF-8 CRLF

Task_1.v > ALSU

```
module ALSU(
        cin <= cin_;
        serial_in <= serial_in_;
        direction <= direction_;
        red_op_A <= red_op_A_;
        red_op_B <= red_op_B_;
        bypass_A <= bypass_A_;
        bypass_B <= bypass_B_;
    end
end

always @(posedge clk or posedge rst) begin
    if (rst) begin
        leds <= 16'b0;
        out <= 6'b0;
    end
    else begin
        ////////////// Handling  Invalid Cases //////////////////
        if (opcode == 3'b111 || opcode == 3'b110 || (((opcode != 3'b000) || (opcode != 3'b001)) && ((red_op_A) || (red_op_B))))begin
            leds <= ~leds;
            if (bypass_A && bypass_B) begin
                if(INPUT_PRIORITY == "A")
                    out <= A;
                else if (INPUT_PRIORITY == "B")
                    out <= B;
                else
                    out <= 6'b0;
            end
            else if (bypass_A)
                out <= A;
            else if (bypass_B)
                out <= B;
            else
                out <= 0;
        end

        ///////// IF all valid but the bypass is high we should Ignore the OUT ////////////
        else if (bypass_A && bypass_B) begin
            if(INPUT_PRIORITY == "A")
```

ed objects Ln 13, Col 32 Spaces: 4 UTF-8 CRLF {} Verilo

```
module ALSU(
                if(INPUT_PRIORITY == "A")
                    out <= A;
                else if (INPUT_PRIORITY == "B")
                    out <= B;
                else
                    out <= 6'b0;
            end
            else if (bypass_A)
                out <= A;
            else if (bypass_B)
                out <= B;



        ////////////// IF YOU reatched here so the op is valid and now you are a ALU ///////////
        else begin
            case (opcode)
                0:  begin // AND operation
                        if (red_op_A && red_op_B) begin

                            out <= {5'b0, &A | &B};
                        end
                        else if (red_op_A) begin

                            out <= {5'b0, &A};
                        end
                        else if (red_op_B) begin
                            out <= {5'b0, &B};
                        end
                        else begin
                            out <= {3'b0, A & B};
                        end
                    end

                1:  begin
                    if (red_op_A && red_op_B) begin
                        out <= {5'b0, ^A ^ ^B};
                    end
```

ed objects Ln 13, Col 32 Spaces: 4 UTF-8 CRLF

```
                1:  begin
                    if (red_op_A && red_op_B) begin
                        out <= {5'b0, ^A ^ ^B};
                    end
                    else if (red_op_A) begin
                        out <= {5'b0, ^A};
                    end
                    else if (red_op_B) begin
                        out <= {5'b0, ^B};
                    end
                    else begin
                        out <= {3'b0, A ^ B};
                    end
                end

                2:  begin
                        if (FULL_ADDER == "ON")
                            out <= A + B + cin;
                        else if (FULL_ADDER == "OFF")
                            out <= A + B;
                        else
                            out <= 0;
                    end
                3:  out <= A * B;
                4:  out <= (direction) ? ({A, B} << 1 | serial_in) : ({serial_in, A, B} >> 1);
                5:  begin // Rotate
                    if (direction)
                        out <= {out[4:0], out[5]}; // Left rotate
                    else
                        out <= {out[0], out[5:1]}; // Right rotate
                end
                default: out <= 6'b0;
            endcase
        end
    end
end

endmodule
```

Testbench:

```
tb_alsu.v > ALSU_tb
module ALSU_tb();
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";

reg [2:0] A, B;
reg [2:0] opcode;
reg cin, serial_in;
reg direction; // 1 is left
reg red_op_A, red_op_B, bypass_A, bypass_B;
reg clk, rst;
wire [5:0] out;
wire [15:0] leds;

ALSU alsu_1 (A, B, opcode, cin, serial_in, direction, red_op_A, red_op_B, bypass_A, bypass_B, clk, rst, out, leds);

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

initial begin


    //////// 2.1 ////////////////
    rst = 1;
    repeat(100) begin
        A = 0;
        B = 0;
        opcode = 0;
        cin = 0;
        serial_in = 0;
        direction = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;
        @(negedge clk);
```

```
        @(negedge clk);
        if (leds != 0 || out != 0)begin
            $display("===> Error while rst = 1,  leds or out is not 0 :( ");
            $stop;
        end
    end


    /////// 2.2 ///////////////////
    rst = 0;
    repeat(100) begin
        A = $random;
        B = $random;
        opcode = $urandom_range(0,5);
        cin = 0;
        serial_in = 0;
        direction = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 1;
        bypass_B = 1;
        repeat(2) @(negedge clk);
        if (out != A)begin
            $display("===> Error while rst = 0, the INPUT_PRIORITY default A and something wrong happened");
            $stop;
        end
    end

    /// 2.3 ////////////////////////
    repeat(100) begin
        A = $random;
        B = $random;
        opcode = 0;
        cin = 0;
        serial_in = 0;
        direction = 0;
        red_op_A = $random;
        red_op_B = $random;
```

d objects Ln 70, Col 21 Spaces: 4 UTF-8 CRLF

```
        red_op_B = $random;
        bypass_A = 0;
        bypass_B = 0;
        repeat(2) @(negedge clk);
            if (red_op_A && red_op_B) begin
                if (out !== {5'b0, &A | &B}) begin
                    $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
                            red_op_A, red_op_B, A, B, out, {5'b0, &A | &B});
                    $stop;
                end
            end
            else if (red_op_A) begin
                if (out !== {5'b0, &A}) begin
                    $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
                            red_op_A, red_op_B, A, B, out, {5'b0, &A});
                    $stop;
                end
            end
            else if (red_op_B) begin
                if (out !== {5'b0, &B}) begin
                    $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
                            red_op_A, red_op_B, A, B, out, {5'b0, &B});
                    $stop;
                end
            end
            else begin
                if (out !== {3'b0, A & B}) begin
                    $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
                            red_op_A, red_op_B, A, B, out, {3'b0, A & B});
                    $stop;
                end
            end
    end

    ///// 2.5 ////////////////////////

    repeat(100) begin
```

objects Ln 70, Col 21 Spaces: 4 UTF-8 CRLF

```
    ///// 2.5 ////////////////////

    repeat(100) begin
        A = $random;
        B = $random;
        opcode = 2;
        cin = $random;
        serial_in = 0;
        direction = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;
        repeat(2) @(negedge clk);
        if (FULL_ADDER == "ON")begin
            if (out != A + B + cin)begin
              $display("Error in 2.5 p1");
              $stop;
            end
        end
        else if (FULL_ADDER == "OFF")begin
            if (out != A + B)begin
              $display("Error in 2.5 p2");
              $stop;
            end
        end
    end

    ///// 2.6 ////////////////////
    repeat(100) begin
        A = $random;
        B = $random;
        opcode = 3;
        cin = 0;
        repeat(2) @(negedge clk);
        if (out != A * B)begin
            $display("ERROR");
```

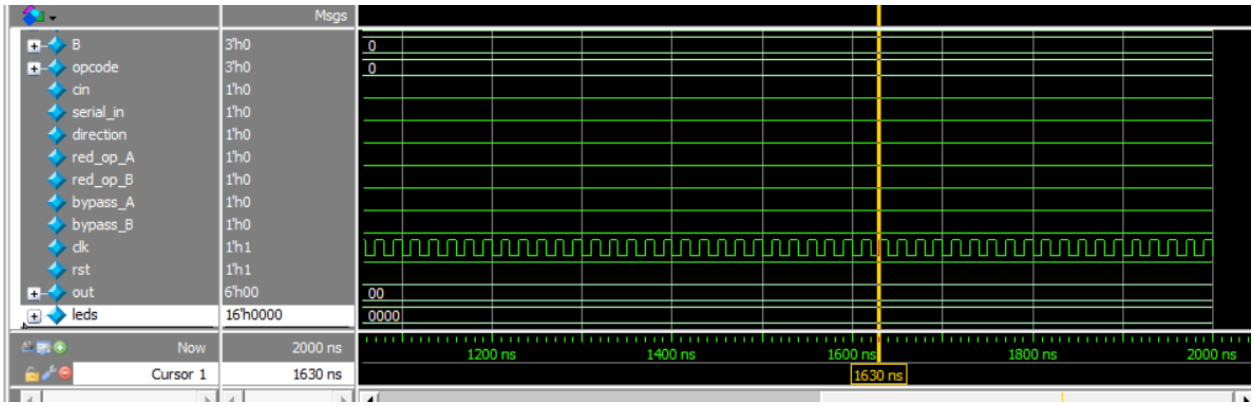
ed objects Ln 70, Col 21 Spaces: 4 UTF-8 CRLF

```
            $display("ERROR");
            $stop;
        end
    end

    ///// 2.7 ////////////////
    repeat(100) begin
        A = $random;
        B = $random;
        serial_in = $random;
        direction = $random;
        opcode = 4;
        repeat(2) @(negedge clk);
        if (direction) begin
            if (out !== ({A, B} << 1 | serial_in)) begin
                $display("Error: A = %b, B = %b, serial_in = %b, direction = %b, out = %b (expected %b)",
                          A, B, serial_in, direction, out, ({A, B} << 1 | serial_in));
                $stop;
            end
        end
        else begin
            if (out !== ({serial_in, A, B} >> 1)) begin
                $display("Error: A = %b, B = %b, serial_in = %b, direction = %b, out = %b (expected %b)",
                          A, B, serial_in, direction, out, ({serial_in, A, B} >> 1));
                $stop;
            end
        end
    end

    ///// 2.8 ///

    repeat (20) begin
            A = $random;
            B = $random;
            direction = $random;
            opcode = 5;
            repeat (2) @(negedge clk);
```

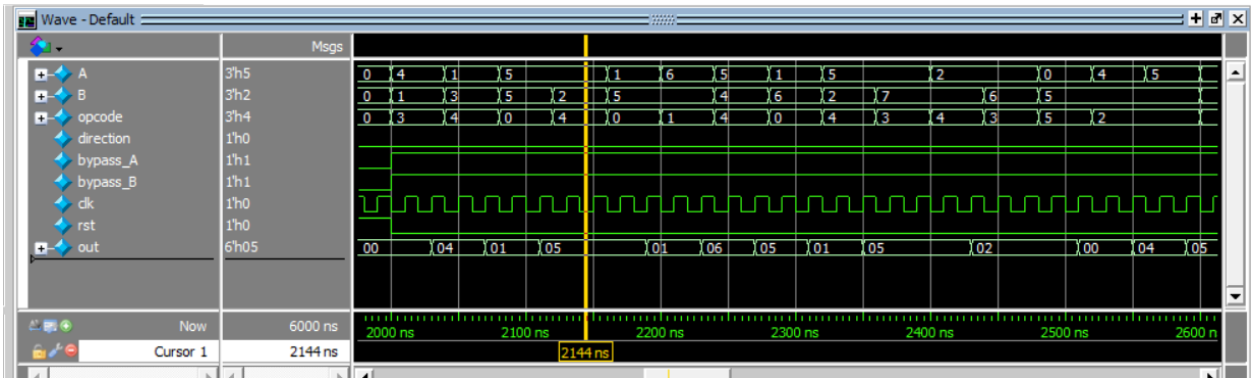
```
181                 opcode = 5;
182                 repeat (2) @(negedge clk);
183                 if (direction) begin
184                     if (out !== {out[4:0], out[5]}) begin
185                         $display("Error: A = %b, B = %b, direction = %b, out = %b (expected %b)",
186                                  A, B, direction, out, {out[4:0], out[5]});
187                         $stop;
188                     end
189                 end
190                 else begin
191                     if (out !== {out[0], out[5:1]}) begin
192                         $display("Error: A = %b, B = %b, direction = %b, out = %b (expected %b)",
193                                  A, B, direction, out, {out[0], out[5:1]});
194                         $stop;
195                     end
196                 end
197             end
198
199         $display("All Tests passed");
200         $stop;
201     end
202
203
204
205
206     endmodule
```

Wave forms:

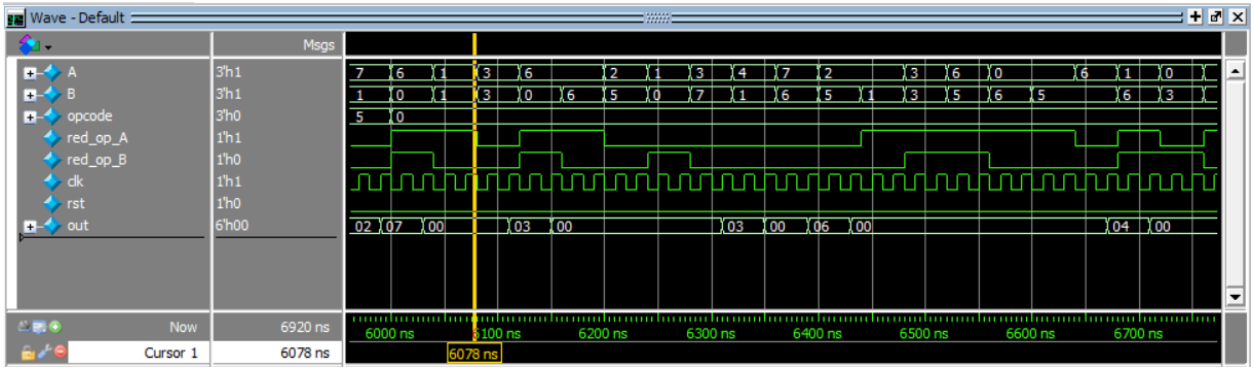
2.1) when Drive rst = 1;



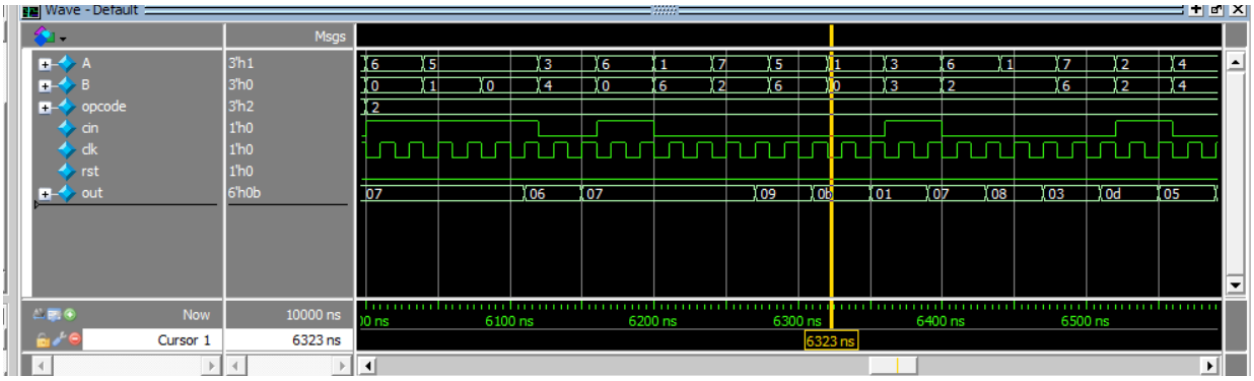
2.2) Drive bypass_A and bypass_B to 1 and deactivate the rst



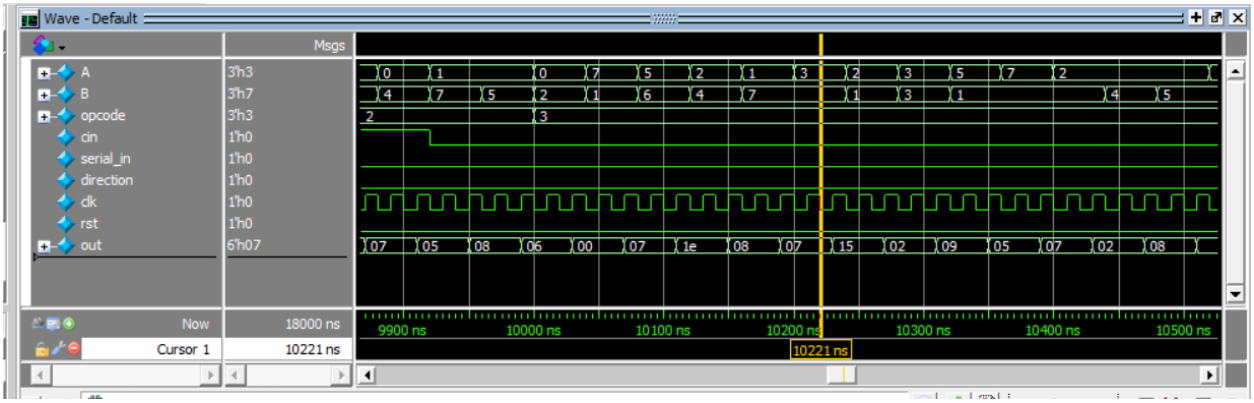
2.3) Verify opcode 0 Functionality



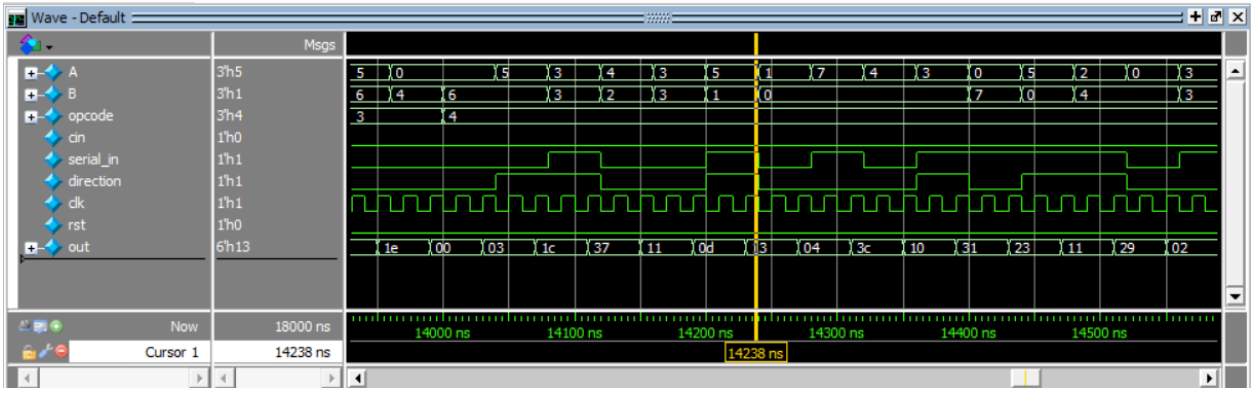
2.5 Verify opcode 2 Functionality



2.6) Verify opcode 3 Functionality



2.7 Verify opcode 4 Functionality



Constraints_basys3.xdc

```
## 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 {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 {serial_in_}]
set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {direction_}]
set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {red_op_A_}]
set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {red_op_B_}]
set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {bypass_A_}]
set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {bypass_B_}]
#

## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {out[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {out[1]}]
set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {out[2]}]
set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {out[3]}]
set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {out[4]}]
set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {out[5]}]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {led[6]}]


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

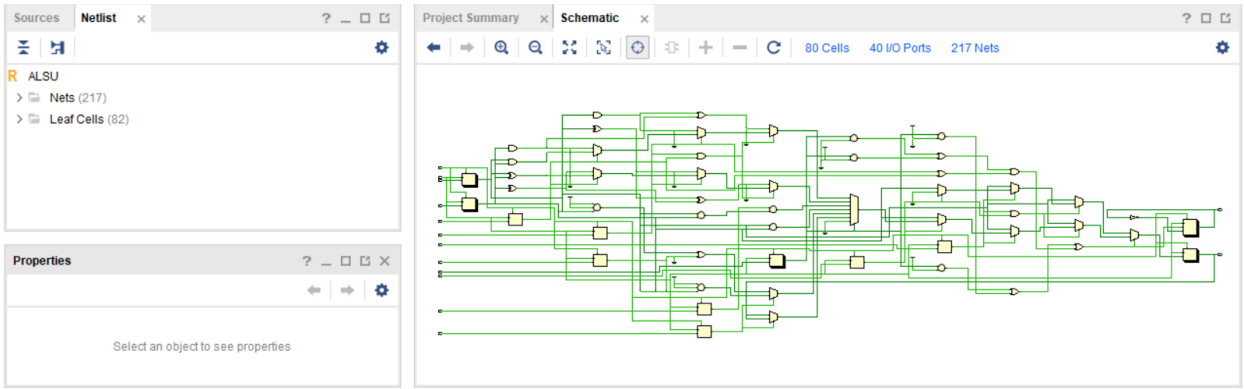
Do file

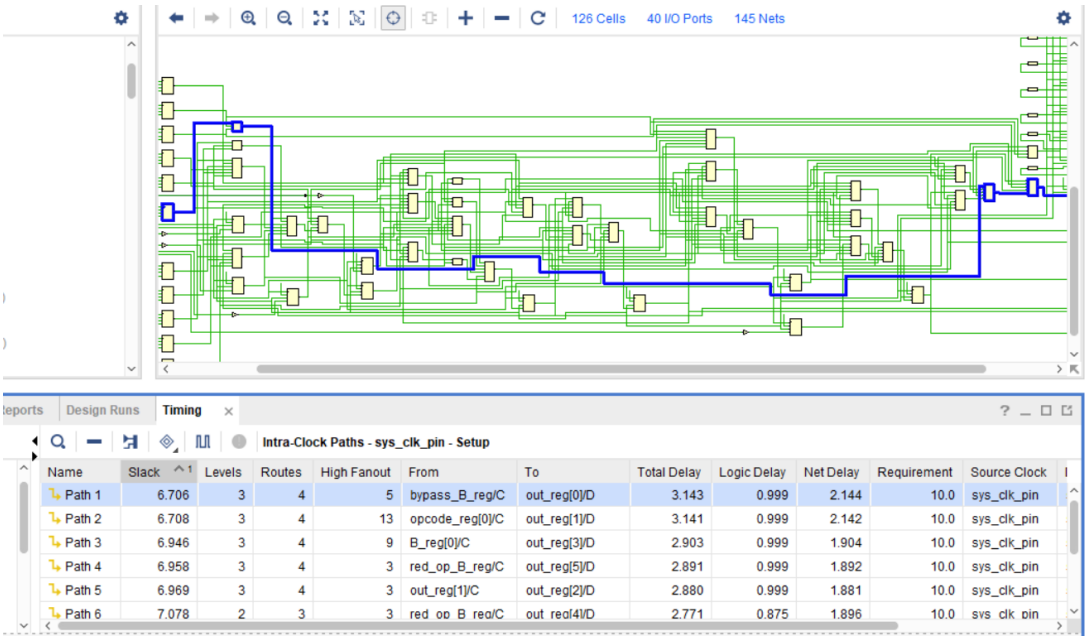
C: > Users > ABDOU > Desktop > Digital World > training_verilog > Section 3 > files > run_dff.do

```
vlib work
vlog Task_1.v tb_alsu.v
vsim -voptargs=+acc work.tb_alsu
add wave *
run -all
#quit -sim
```

VIVADOOOO:

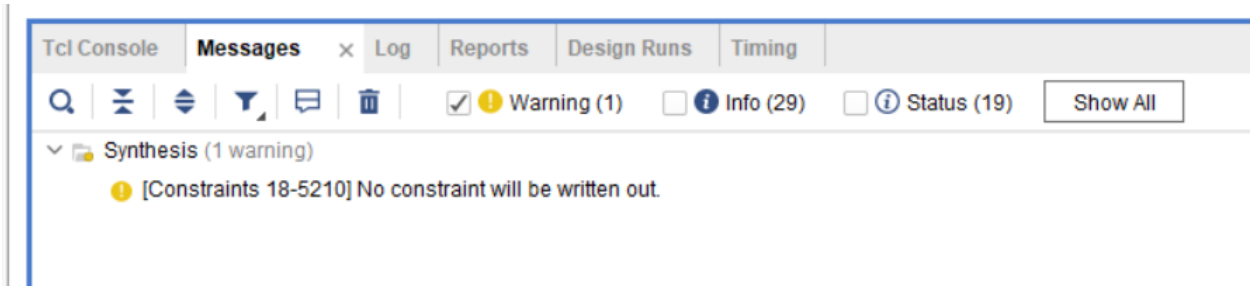
No warning :





Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	6.706	3	4	5	bypass_B_reg/C	out_reg[0]/D	3.143	0.999	2.144	10.0	sys_clk_pin
Path 2	6.708	3	4	13	opcode_reg[0]/C	out_reg[1]/D	3.141	0.999	2.142	10.0	sys_clk_pin
Path 3	6.946	3	4	9	B_reg[0]/C	out_reg[3]/D	2.903	0.999	1.904	10.0	sys_clk_pin
Path 4	6.958	3	4	3	red_op_B_reg/C	out_reg[5]/D	2.891	0.999	1.892	10.0	sys_clk_pin
Path 5	6.969	3	4	3	out_reg[1]/C	out_reg[2]/D	2.880	0.999	1.881	10.0	sys_clk_pin
Path 6	7.078	2	3	3	red_op_B_reg/C	out_reg[4]/D	2.771	0.875	1.896	10.0	sys_clk_pin

I don’t know what is this



TASK 2

Task_2.v | tb_dsp.v

Task_2.v > DSP48A1

```v
module DSP48A1 (
    input  [17:0] A,
    input  [17:0] B,
    input  [47:0] C,
    input  [17:0] D,
    input  clk,
    input  rst_n,
    output reg [47:0] P
);
    parameter OPERATION = "ADD"; // Default operation is ADD

    reg [17:0] A_reg, B_reg, D_reg;
    reg [47:0] C_reg;
    reg [47:0] P_reg;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            A_reg <= 18'b0;
            B_reg <= 18'b0;
            C_reg <= 48'b0;
            D_reg <= 18'b0;
        end
        else begin
            A_reg <= A;
            B_reg <= B;
            C_reg <= C;
            D_reg <= D;
        end
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            P_reg <= 48'b0;
        end
        else begin
            if (OPERATION == "ADD")
                P_reg <= C_reg + (A_reg * B_reg) + D_reg;
            else if (OPERATION == "SUBTRACT")
                P_reg <= C_reg - (A_reg * B_reg) - D_reg;
            else
                P_reg <= 48'b0;
```

```
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            P_reg <= 48'b0;
        end
        else begin
            if (OPERATION == "ADD")
                P_reg <= C_reg + (A_reg * B_reg) + D_reg;
            else if (OPERATION == "SUBTRACT")
                P_reg <= C_reg - (A_reg * B_reg) - D_reg;
            else
                P_reg <= 48'b0;
        end
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            P <= 48'b0;
        end
        else begin
            P <= P_reg;
        end
    end

endmodule
```

TEST:

tb_dsp.v > DSP48A1_tb

```v
module DSP48A1_tb;

    // Inputs
    reg [17:0] A;
    reg [17:0] B;
    reg [47:0] C;
    reg [17:0] D;
    reg clk;
    reg rst_n;

    wire [47:0] P;

    DSP48A1 #(.OPERATION("ADD")) dut_add (
        .A(A),
        .B(B),
        .C(C),
        .D(D),
        .clk(clk),
        .rst_n(rst_n),
        .P(P)
    );

    DSP48A1 #(.OPERATION("SUBTRACT")) dut_sub (
        .A(A),
        .B(B),
        .C(C),
        .D(D),
        .clk(clk),
        .rst_n(rst_n),
        .P(P)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
```

```v
    initial begin
        A = 18'b0;
        B = 18'b0;
        C = 48'b0;
        D = 18'b0;
        rst_n = 0;

        #20;
        rst_n = 1;

        A = 18'h1; B = 18'h2; C = 48'h3; D = 18'h4;
        #20;
        if (P !== 48'h9) begin
            $display("Error: Test case 1 failed. P = %h (expected 9)", P);
            $stop;
        end

        A = 18'h2; B = 18'h3; C = 48'h10; D = 18'h1;
        #20;
        if (P !== 48'h9) begin
            $display("Error: Test case 2 failed. P = %h (expected 9)", P);
            $stop;
        end

        A = 18'b0; B = 18'b0; C = 48'b0; D = 18'b0;
        #20;
        if (P !== 48'b0) begin
            $display("Error: Test case 3 failed. P = %h (expected 0)", P);
            $stop;
        end

        A = 18'h3FFF; B = 18'h3FFF; C = 48'hFFFFFFFF; D = 18'h3FFF;
        #20;
        if (P !== 48'hFFFFFFFF + (18'h3FFF * 18'h3FFF) + 18'h3FFF) begin
            $display("Error: Test case 4 failed. P = %h", P);
            $stop;
        end

        $display("All test cases passed!");
```

DO FILE:

```
vlib work
vlog Task_2.v tb_dsp.v
vsim -voptargs=+acc work.tb_dsp
add wave *
run -all
#quit -sim
```

TASK 3

Design:

Task_3.v > TDM

```v
module TDM (
    input  clk,
    input  rst,
    input  [1:0] in0,
    input  [1:0] in1,
    input  [1:0] in2,
    input  [1:0] in3,
    output reg [1:0] out
);

    reg [1:0] counter;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            counter <= 2'b00;
            out <= in0;
        end
        else begin
            counter <= counter + 1;

            case (counter)
                2'b00: out <= in0;
                2'b01: out <= in1;
                2'b10: out <= in2;
                2'b11: out <= in3;
            endcase
        end
    end

endmodule
```

Test:

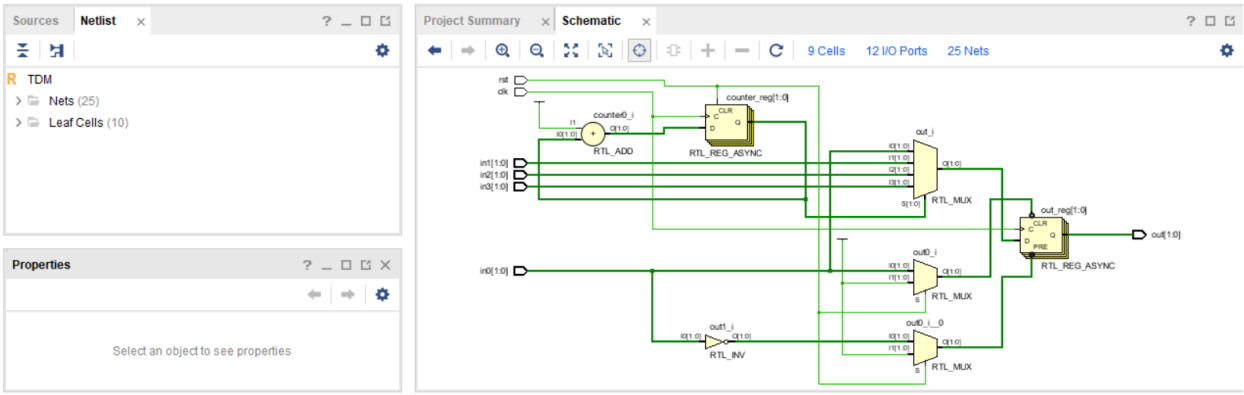
```
module TDM_tb;

    reg clk;
    reg rst;
    reg [1:0] in0, in1, in2, in3;

    wire [1:0] out;

    TDM dut (
        .clk(clk),
        .rst(rst),
        .in0(in0),
        .in1(in1),
        .in2(in2),
        .in3(in3),
        .out(out)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        rst = 1;
        in0 = 2'b00; in1 = 2'b01; in2 = 2'b10; in3 = 2'b11;

        #20;
        rst = 0;

        #10;
        if (out !== in0) begin
            $display("Error");
            $stop;
        end

        #10;
```

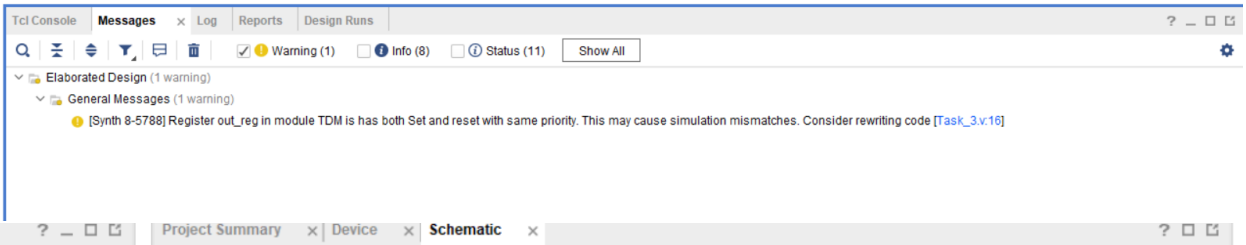
```
        if (out !== in0) begin
            $display("Error");
            $stop;
        end

        #10;
        if (out !== in1) begin
            $display("Error");
            $stop;
        end

        #10;
        if (out !== in2) begin
            $display("Error");
            $stop;
        end

        #10;
        if (out !== in3) begin
            $display("Error");
            $stop;
        end

        rst = 1;
        #10;
        if (out !== in0) begin
            $display("Error");
            $stop;
        end

        $display("All test cases passed!");
        $stop;
    end

endmodule
```

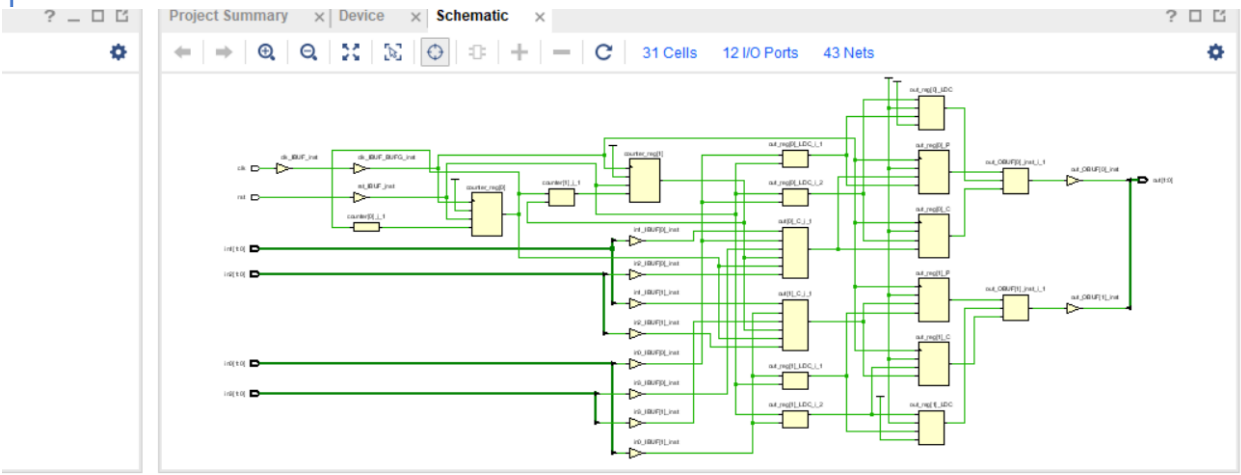
Constrain:

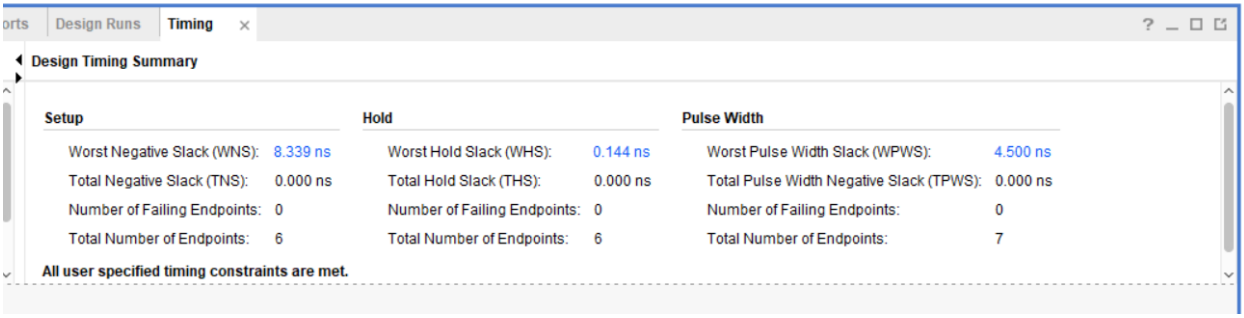
constrain_3.xdc

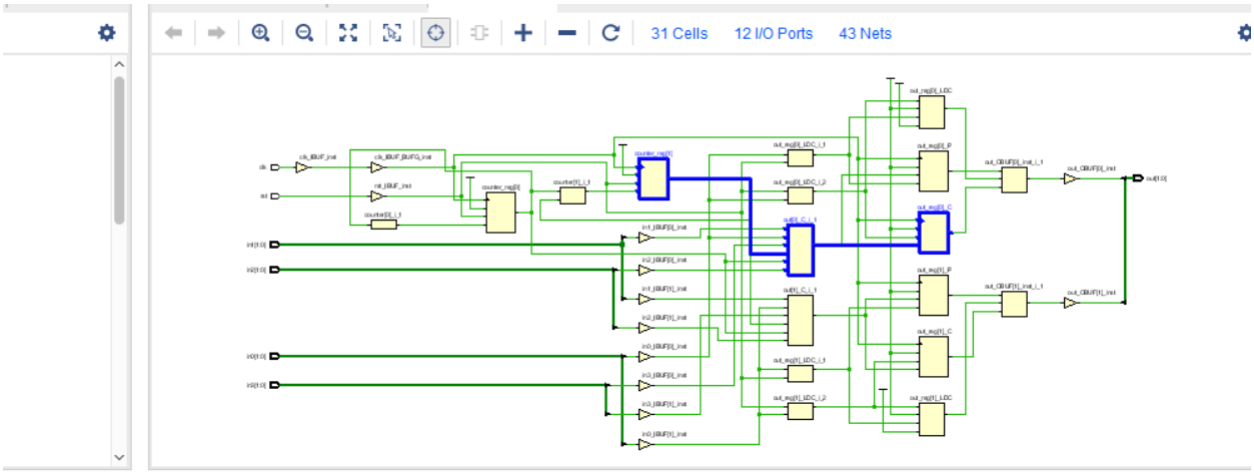
```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {in0[0]}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {in0[1]}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {in1[0]}]
set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {in1[1]}]
set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {in2[0]}]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {in2[1]}]
set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {in3[0]}]
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {in3[1]}]

## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {out[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {out[1]}]

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


## Configuration options
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```


Sources
Netlist
TDM
Nets (25)
Leaf Cells (10)
Properties
Select an object to see properties
Project Summary
Schematic
9 Cells
12 I/O Ports
25 Nets
RTL_ADD
RTL_REG_ASYNC
RTL_MUX
RTL_INV


Tcl Console
Messages
Log
Reports
Design Runs
Warning (1)
Info (8)
Status (11)
Show All
Elaborated Design (1 warning)
General Messages (1 warning)
[Synth 8-5788] Register out_reg in module TDM is has both Set and reset with same priority. This may cause simulation mismatches. Consider rewriting code [Task_3.v:16]


Project Summary
Device
Schematic
31 Cells
12 I/O Ports
43 Nets


Design Runs
Timing
Design Timing Summary
Setup
Worst Negative Slack (WNS): 8.339 ns
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 6
Hold
Worst Hold Slack (WHS): 0.144 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 6
Pulse Width
Worst Pulse Width Slack (WPWS): 4.500 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 7
All user specified timing constraints are met.



Design Runs | Timing

Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source Clock	D
Path 1	8.339	1	2	3	counter_reg[1]/C	out_reg[0]_C/D	1.510	0.751	0.759	10.0	sys_clk_pin	s
Path 2	8.339	1	2	3	counter_reg[1]/C	out_reg[0]_P/D	1.510	0.751	0.759	10.0	sys_clk_pin	s
Path 3	8.339	1	2	3	counter_reg[1]/C	out_reg[1]_C/D	1.510	0.751	0.759	10.0	sys_clk_pin	s
Path 4	8.339	1	2	3	counter_reg[1]/C	out_reg[1]_P/D	1.510	0.751	0.759	10.0	sys_clk_pin	s
Path 5	8.578	1	2	4	counter_reg[0]/C	counter_reg[1]/D	1.271	0.777	0.494	10.0	sys_clk_pin	s
Path 6	8.763	1	2	4	counter_reg[0]/C	counter_reg[0]/D	1.086	0.751	0.335	10.0	sys_clk_pin	s