

TASK 1

Design:

```
1  module ALSU(  
2      input [2:0] A_, B_,  
3      input [2:0] opcode_,  
4      input cin_,  
5      input serial_in_,  
6      input direction_, // 1 is left  
7      input red_op_A_, red_op_B_, bypass_A_, bypass_B_,  
8      input clk, rst,  
9      output reg [5:0] out,  
10     output reg [15:0] leds  
11 );  
12  
13 parameter INPUT_PRIORITY = "A";  
14 parameter FULL_ADDER = "ON";  
15  
16 reg [2:0] A, B;  
17 reg [2:0] opcode;  
18 reg cin;  
19 reg serial_in;  
20 reg direction; // 1 is left  
21 reg red_op_A, red_op_B, bypass_A, bypass_B;  
22  
23 always @(posedge clk or posedge rst) begin  
24     if (rst) begin  
25         A <= 0;  
26         B <= 0;  
27         opcode <= 0;  
28         cin <= 0;  
29         serial_in <= 0;  
30         direction <= 0;  
31         red_op_A <= 0;  
32         red_op_B <= 0;  
33         bypass_A <= 0;  
34         bypass_B <= 0;  
35     end  
36     else begin  
37         A <= A_;  
38         B <= B_;  
39         opcode <= opcode_;
```

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

```

V Task_1.v > ALSU
1  module ALSU(
40      cin <= cin_;
41      serial_in <= serial_in_;
42      direction <= direction_;
43      red_op_A <= red_op_A_;
44      red_op_B <= red_op_B_;
45      bypass_A <= bypass_A_;
46      bypass_B <= bypass_B_;
47  end
48  end
49
50  always @(posedge clk or posedge rst) begin
51      if (rst) begin
52          leds <= 16'b0;
53          out <= 6'b0;
54      end
55      else begin
56          /////////////// Handling Invalid Cases ///////////////////
57          if (opcode == 3'b111 || opcode == 3'b110 || (((opcode != 3'b000) || (opcode != 3'b001)) && ((red_op_A) || (red_op_B))))begin
58              leds <= ~leds;
59              if (bypass_A && bypass_B) begin
60                  if (INPUT_PRIORITY == "A")
61                      out <= A;
62                  else if (INPUT_PRIORITY == "B")
63                      out <= B;
64                  else
65                      out <= 6'b0;
66              end
67              else if (bypass_A)
68                  out <= A;
69              else if (bypass_B)
70                  out <= B;
71              else
72                  out <= 0;
73          end
74
75          /////////////// IF all valid but the bypass is high we should Ignore the OUT ///////////////////
76          else if (bypass_A && bypass_B) begin
77              if (INPUT_PRIORITY == "A")

```

Ln 13, Col 32 Spaces: 4 UTF-8 CRLF { } Verilog

```

1  module ALSU(
77      if (INPUT_PRIORITY == "A")
78          out <= A;
79      else if (INPUT_PRIORITY == "B")
80          out <= B;
81      else
82          out <= 6'b0;
83  end
84  else if (bypass_A)
85      out <= A;
86  else if (bypass_B)
87      out <= B;
88
89
90
91  /////////////// IF YOU reached here so the op is valid and now you are a ALU ///////////////////
92  else begin
93      case (opcode)
94          0: begin // AND operation
95              if (red_op_A && red_op_B) begin
96                  out <= {5'b0, &A | &B};
97              end
98              else if (red_op_A) begin
99                  out <= {5'b0, &A};
100              end
101              else if (red_op_B) begin
102                  out <= {5'b0, &B};
103              end
104              else begin
105                  out <= {3'b0, A & B};
106              end
107          end
108      end
109  end
110
111      1: begin
112          if (red_op_A && red_op_B) begin
113              out <= {5'b0, ^A ^ ^B};
114          end

```

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

```

111         1: begin
112             if (red_op_A && red_op_B) begin
113                 out <= {5'b0, ^A ^ ^B};
114             end
115             else if (red_op_A) begin
116                 out <= {5'b0, ^A};
117             end
118             else if (red_op_B) begin
119                 out <= {5'b0, ^B};
120             end
121             else begin
122                 out <= {3'b0, A ^ B};
123             end
124         end
125
126         2: begin
127             if (FULL_ADDER == "ON")
128                 out <= A + B + cin;
129             else if (FULL_ADDER == "OFF")
130                 out <= A + B;
131             else
132                 out <= 0;
133             end
134         3: out <= A * B;
135         4: out <= (direction) ? ({A, B} << 1 | serial_in) : ({serial_in, A, B} >> 1);
136         5: begin // Rotate
137             if (direction)
138                 out <= {out[4:0], out[5]}; // Left rotate
139             else
140                 out <= {out[0], out[5:1]}; // Right rotate
141             end
142             default: out <= 6'b0;
143         endcase
144     end
145 end
146 end
147
148 endmodule

```

Testbench:

```

V tb_alsu.v > ALSU_tb
1  module ALSU_tb();
2  parameter INPUT_PRIORITY = "A";
3  parameter FULL_ADDER = "ON";
4
5  reg [2:0] A, B;
6  reg [2:0] opcode;
7  reg cin, serial_in;
8  reg direction; // 1 is left
9  reg red_op_A, red_op_B, bypass_A, bypass_B;
10 reg clk, rst;
11 wire [5:0] out;
12 wire [15:0] leds;
13
14 ALSU alsu_1 (A, B, opcode, cin, serial_in, direction, red_op_A, red_op_B, bypass_A, bypass_B, clk, rst, out, leds);
15
16 initial begin
17     clk = 0;
18     forever begin
19         #10 clk = ~clk;
20     end
21 end
22
23 initial begin
24
25     ////////// 2.1 //////////
26     rst = 1;
27     repeat(100) begin
28         A = 0;
29         B = 0;
30         opcode = 0;
31         cin = 0;
32         serial_in = 0;
33         direction = 0;
34         red_op_A = 0;
35         red_op_B = 0;
36         bypass_A = 0;
37         bypass_B = 0;
38         @(negedge clk);
39

```

```

39     @(negedge clk);
40     if (leds != 0 || out != 0)begin
41         $display("==> Error while rst = 1, leds or out is not 0 :( ");
42         $stop;
43     end
44 end
45
46
47 // 2.2 //
48 rst = 0;
49 repeat(100) begin
50     A = $random;
51     B = $random;
52     opcode = $urandom_range(0,5);
53     cin = 0;
54     serial_in = 0;
55     direction = 0;
56     red_op_A = 0;
57     red_op_B = 0;
58     bypass_A = 1;
59     bypass_B = 1;
60     repeat(2) @(negedge clk);
61     if (out != A)begin
62         $display("==> Error while rst = 0, the INPUT_PRIORITY default A and something wrong happened");
63         $stop;
64     end
65 end
66
67 // 2.3 //
68 repeat(100) begin
69     A = $random;
70     B = $random;
71     opcode = 0;
72     cin = 0;
73     serial_in = 0;
74     direction = 0;
75     red_op_A = $random;
76     red_op_B = $random;

```

objects

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

```

76     red_op_B = $random;
77     bypass_A = 0;
78     bypass_B = 0;
79     repeat(2) @(negedge clk);
80     if (red_op_A && red_op_B) begin
81         if (out != {5'b0, &A | &B}) begin
82             $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
83                 red_op_A, red_op_B, A, B, out, {5'b0, &A | &B});
84             $stop;
85         end
86     end
87     else if (red_op_A) begin
88         if (out != {5'b0, &A}) begin
89             $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
90                 red_op_A, red_op_B, A, B, out, {5'b0, &A});
91             $stop;
92         end
93     end
94     else if (red_op_B) begin
95         if (out != {5'b0, &B}) begin
96             $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
97                 red_op_A, red_op_B, A, B, out, {5'b0, &B});
98             $stop;
99         end
100     end
101     else begin
102         if (out != {3'b0, A & B}) begin
103             $display("Error: red_op_A = %b, red_op_B = %b, A = %b, B = %b, out = %b (expected %b)",
104                 red_op_A, red_op_B, A, B, out, {3'b0, A & B});
105             $stop;
106         end
107     end
108 end
109
110 // 2.5 //
111
112 repeat(100) begin

```

objects

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

```

109
110
111
112 repeat(100) begin
113     A = $random;
114     B = $random;
115     opcode = 2;
116     cin = $random;
117     serial_in = 0;
118     direction = 0;
119     red_op_A = 0;
120     red_op_B = 0;
121     bypass_A = 0;
122     bypass_B = 0;
123     repeat(2) @(negedge clk);
124     if (FULL_ADDER == "ON")begin
125         if (out != A + B + cin)begin
126             $display("Error in 2.5 p1");
127             $stop;
128         end
129     end
130     else if (FULL_ADDER == "OFF")begin
131         if (out != A + B)begin
132             $display("Error in 2.5 p2");
133             $stop;
134         end
135     end
136 end
137
138
139 repeat(100) begin
140     A = $random;
141     B = $random;
142     opcode = 3;
143     cin = 0;
144     repeat(2) @(negedge clk);
145     if (out != A * B)begin
146         $display("ERROR");
147     end
148 end

```

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

```

146     $display("ERROR");
147     $stop;
148 end
149
150
151
152 repeat(100) begin
153     A = $random;
154     B = $random;
155     serial_in = $random;
156     direction = $random;
157     opcode = 4;
158     repeat(2) @(negedge clk);
159     if (direction) begin
160         if (out != ((A, B) << 1 | serial_in)) begin
161             $display("Error: A = %b, B = %b, serial_in = %b, direction = %b, out = %b (expected %b)",
162                 A, B, serial_in, direction, out, ((A, B) << 1 | serial_in));
163             $stop;
164         end
165     end
166     else begin
167         if (out != ({serial_in, A, B} >> 1)) begin
168             $display("Error: A = %b, B = %b, serial_in = %b, direction = %b, out = %b (expected %b)",
169                 A, B, serial_in, direction, out, ({serial_in, A, B} >> 1));
170             $stop;
171         end
172     end
173 end
174
175
176
177 repeat (20) begin
178     A = $random;
179     B = $random;
180     direction = $random;
181     opcode = 5;
182     repeat (2) @(negedge clk);
183     if (out != A + B + direction) begin
184         $display("Error: A = %b, B = %b, direction = %b, out = %b (expected %b)",
185             A, B, direction, out, A + B + direction);
186         $stop;
187     end
188 end

```

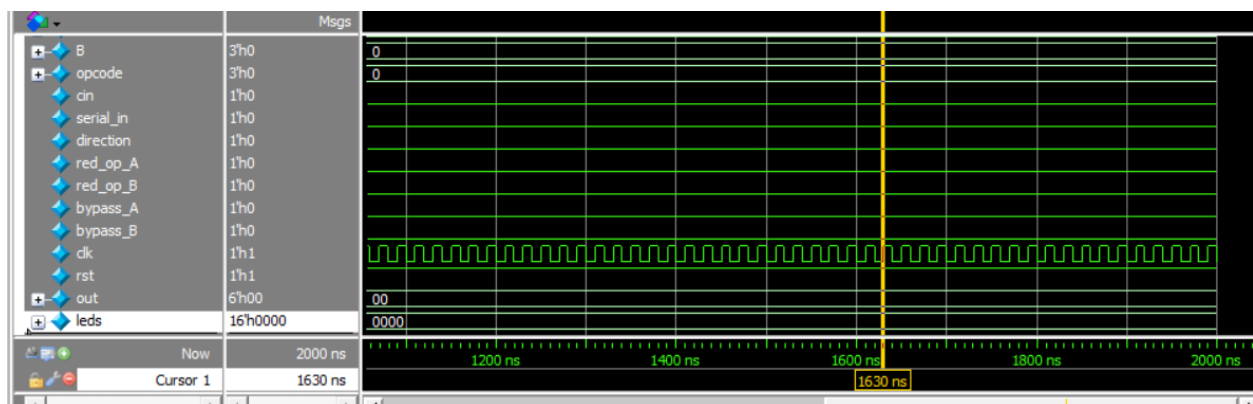
```

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

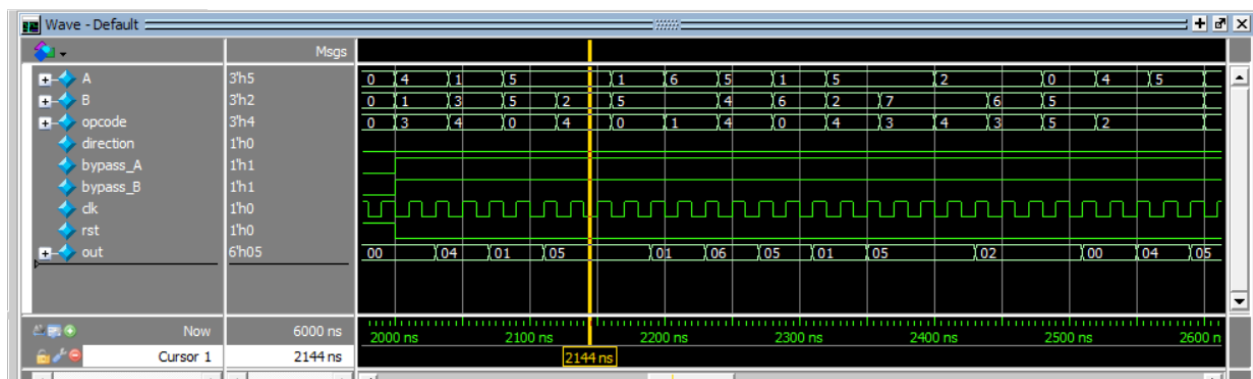
```

Wave forms:

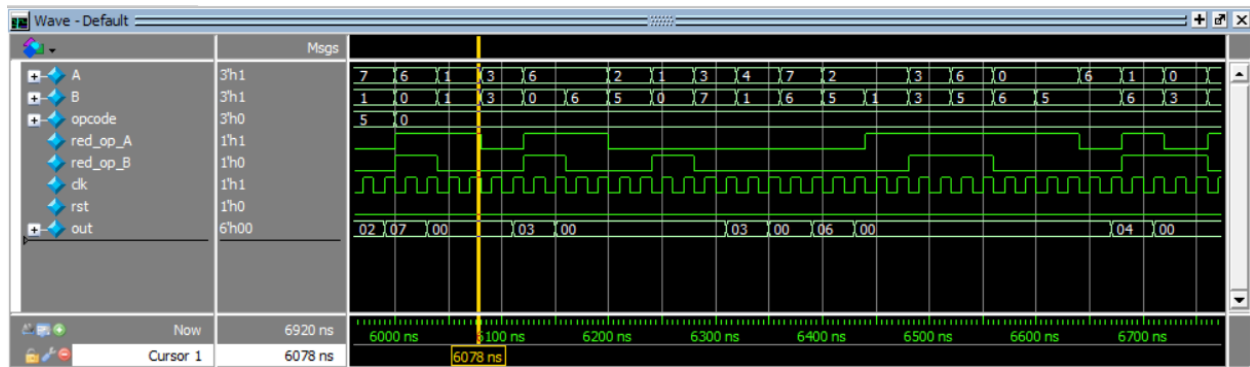
2.1) when Drive rst = 1;



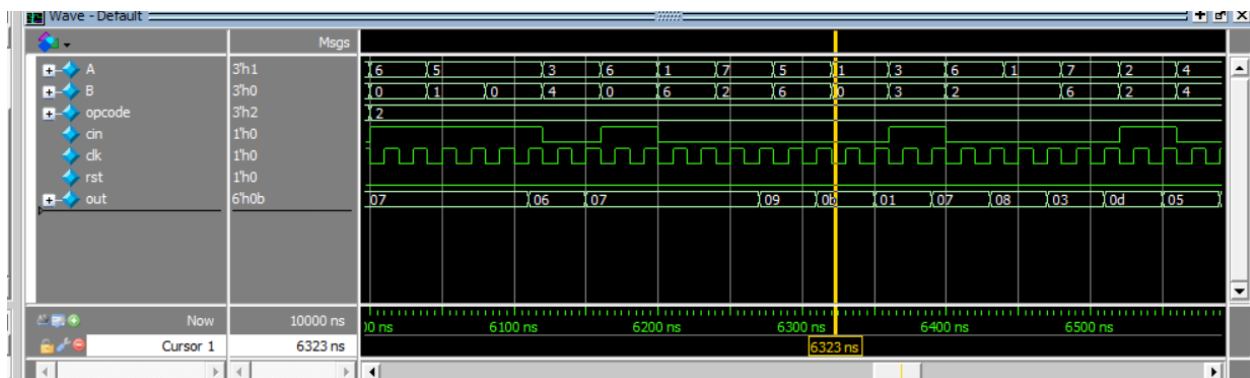
2.2) Drive bypass_A and bypass_B to 1 and deactivate the rst



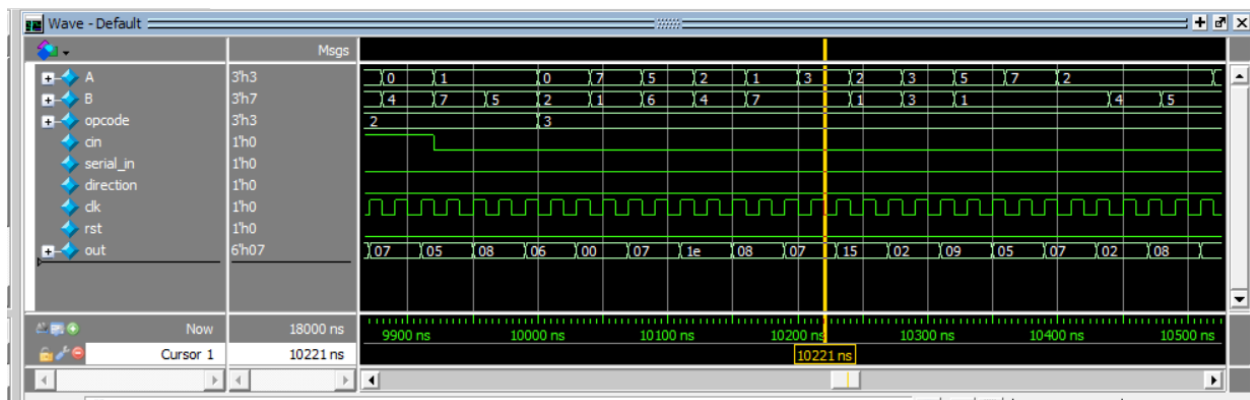
2.3) Verify opcode 0 Functionality



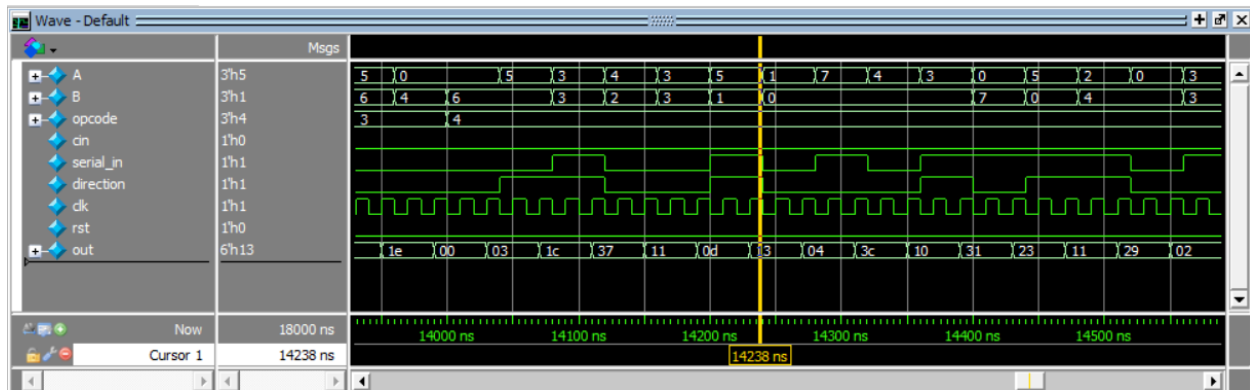
2.5 Verify opcode 2 Functionality



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

1 vlib work
2 vlog Task_1.v tb_alu.v
3 vsim -voptargs=+acc work.tb_alu
4 add wave *
5 run -all
6 #quit -sim
```

VIVADOOOOO:

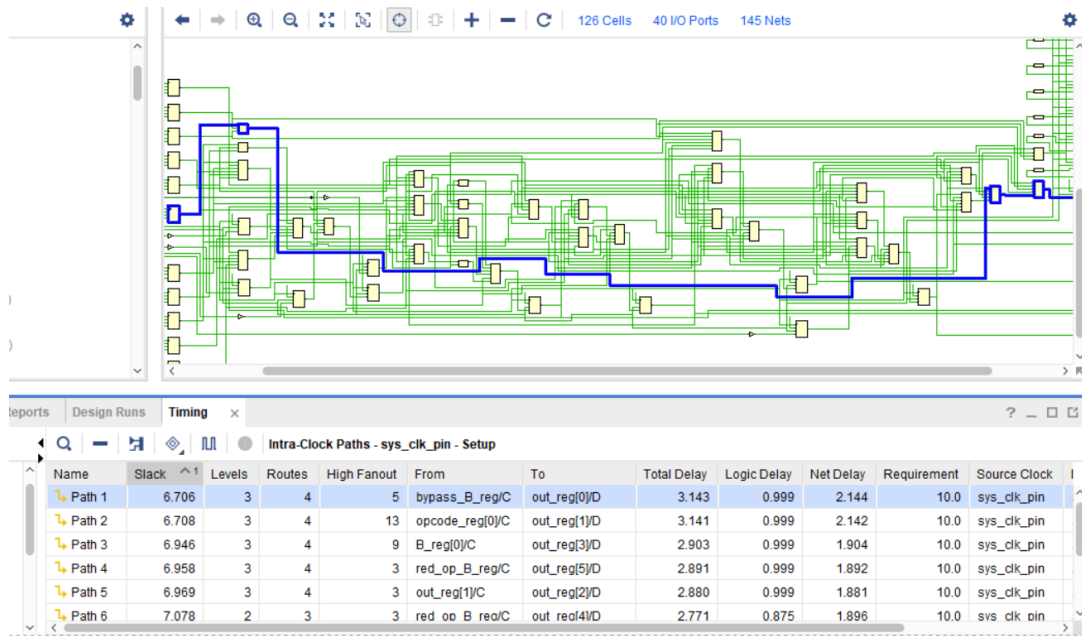
No warning :

The screenshot displays the Vivado IDE interface. The top toolbar includes tabs for 'Tcl Console', 'Messages', 'Log', 'Reports', and 'Design Runs'. The 'Messages' window is active, showing three general messages:

- [IP_Flow 19-234] Refreshing IP repositories
- [IP_Flow 19-1704] No user IP repositories specified
- [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.2/data/ip'.

Below the messages, the 'Sources' window shows a project structure with 'ALU' and 'Nets (217)'. The 'Properties' window is empty. The 'Schematic' window displays a logic diagram with 80 cells, 40 I/O ports, and 217 nets. The bottom 'Design Runs' window shows a table of design runs:

| Name | Constraints | Status | WNS | TNS | WHS | THS | TPWS | Total Power | Failed Routes | LUT | FF | BRAMs | URAM | DSP | Start | Elapsed | Run Strategy |
|---------|-------------|-------------|-----|-----|-----|-----|------|-------------|---------------|-----|----|-------|------|-----|-------|---------|--|
| synth_1 | constrs_1 | Not started | | | | | | | | | | | | | | | Vivado Synthesis Defaults (Vivado Synthesis) |
| impl_1 | constrs_1 | Not started | | | | | | | | | | | | | | | Vivado Implementation Defaults (Vivado Impl) |



I don't know what is this

Messages window showing a warning message from Synthesis.

Synthesis (1 warning)

[Constraints 18-5210] No constraint will be written out.

TASK 2

```
Task_2.v x tb_dsp.v
Task_2.v > DSP48A1
1 module DSP48A1 (
2     input [17:0] A,
3     input [17:0] B,
4     input [47:0] C,
5     input [17:0] D,
6     input clk,
7     input rst_n,
8     output reg [47:0] P
9 );
10 parameter OPERATION = "ADD"; // Default operation is ADD
11
12 reg [17:0] A_reg, B_reg, D_reg;
13 reg [47:0] C_reg;
14 reg [47:0] P_reg;
15
16 always @(posedge clk or negedge rst_n) begin
17     if (!rst_n) begin
18         A_reg <= 18'b0;
19         B_reg <= 18'b0;
20         C_reg <= 48'b0;
21         D_reg <= 18'b0;
22     end
23     else begin
24         A_reg <= A;
25         B_reg <= B;
26         C_reg <= C;
27         D_reg <= D;
28     end
29 end
30
31 always @(posedge clk or negedge rst_n) begin
32     if (!rst_n) begin
33         P_reg <= 48'b0;
34     end
35     else begin
36         if (OPERATION == "ADD")
37             P_reg <= C_reg + (A_reg * B_reg) + D_reg;
38         else if (OPERATION == "SUBTRACT")
39             P_reg <= C_reg - (A_reg * B_reg) - D_reg;
40         else
41             P_reg <= 48'b0;
```

```
30
31     always @(posedge clk or negedge rst_n) begin
32         if (!rst_n) begin
33             P_reg <= 48'b0;
34         end
35         else begin
36             if (OPERATION == "ADD")
37                 P_reg <= C_reg + (A_reg * B_reg) + D_reg;
38             else if (OPERATION == "SUBTRACT")
39                 P_reg <= C_reg - (A_reg * B_reg) - D_reg;
40             else
41                 P_reg <= 48'b0;
42         end
43     end
44
45     always @(posedge clk or negedge rst_n) begin
46         if (!rst_n) begin
47             P <= 48'b0;
48         end
49         else begin
50             P <= P_reg;
51         end
52     end
53
54 endmodule
```

TEST:

```
tb_dsp.v > DSP48A1_tb
1  module DSP48A1_tb;
2
3      // Inputs
4      reg [17:0] A;
5      reg [17:0] B;
6      reg [47:0] C;
7      reg [17:0] D;
8      reg clk;
9      reg rst_n;
10
11      wire [47:0] P;
12
13      DSP48A1 #(.OPERATION("ADD")) dut_add (
14          .A(A),
15          .B(B),
16          .C(C),
17          .D(D),
18          .clk(clk),
19          .rst_n(rst_n),
20          .P(P)
21      );
22
23      DSP48A1 #(.OPERATION("SUBTRACT")) dut_sub (
24          .A(A),
25          .B(B),
26          .C(C),
27          .D(D),
28          .clk(clk),
29          .rst_n(rst_n),
30          .P(P)
31      );
32
33      initial begin
34          clk = 0;
35          forever #5 clk = ~clk;
36      end
37
```

```
37
38      initial begin
39          A = 18'b0;
40          B = 18'b0;
41          C = 48'b0;
42          D = 18'b0;
43          rst_n = 0;
44
45          #20;
46          rst_n = 1;
47
48          A = 18'h1; B = 18'h2; C = 48'h3; D = 18'h4;
49          #20;
50          if (P !== 48'h9) begin
51              $display("Error: Test case 1 failed. P = %h (expected 9)", P);
52              $stop;
53          end
54
55          A = 18'h2; B = 18'h3; C = 48'h10; D = 18'h1;
56          #20;
57          if (P !== 48'h9) begin
58              $display("Error: Test case 2 failed. P = %h (expected 9)", P);
59              $stop;
60          end
61
62          A = 18'b0; B = 18'b0; C = 48'b0; D = 18'b0;
63          #20;
64          if (P !== 48'b0) begin
65              $display("Error: Test case 3 failed. P = %h (expected 0)", P);
66              $stop;
67          end
68
69          A = 18'h3FFF; B = 18'h3FFF; C = 48'hFFFFFFFF; D = 18'h3FFF;
70          #20;
71          if (P !== 48'hFFFFFFFF + (18'h3FFF * 18'h3FFF) + 18'h3FFF) begin
72              $display("Error: Test case 4 failed. P = %h", P);
73              $stop;
74          end
75
76          $display("All test cases passed!");
77      end
78
```

DO FILE:

```
1  vlib work
2  vlog Task_2.v tb_dsp.v
3  vsim -voptargs=+acc work.tb_dsp
4  add wave *
5  run -all
6  #quit -sim
```

TASK 3

Design:

```
V Task_3.v > TDM
1  module TDM (
2      input  clk,
3      input  rst,
4      input  [1:0] in0,
5      input  [1:0] in1,
6      input  [1:0] in2,
7      input  [1:0] in3,
8      output reg [1:0] out
9  );
10
11      reg [1:0] counter;
12
13      always @(posedge clk or posedge rst) begin
14          if (rst) begin
15              counter <= 2'b00;
16              out <= in0;
17          end
18          else begin
19              counter <= counter + 1;
20
21              case (counter)
22                  2'b00: out <= in0;
23                  2'b01: out <= in1;
24                  2'b10: out <= in2;
25                  2'b11: out <= in3;
26              endcase
27          end
28      end
29
30  endmodule
```

Test:

```

1  module TDM_tb;
2
3      reg clk;
4      reg rst;
5      reg [1:0] in0, in1, in2, in3;
6
7      wire [1:0] out;
8
9      TDM dut (
10         .clk(clk),
11         .rst(rst),
12         .in0(in0),
13         .in1(in1),
14         .in2(in2),
15         .in3(in3),
16         .out(out)
17     );
18
19     initial begin
20         clk = 0;
21         forever #5 clk = ~clk;
22     end
23
24     initial begin
25         rst = 1;
26         in0 = 2'b00; in1 = 2'b01; in2 = 2'b10; in3 = 2'b11;
27
28         #20;
29         rst = 0;
30
31         #10;
32         if (out != in0) begin
33             $display("Error");
34             $stop;
35         end
36
37         #10;

```

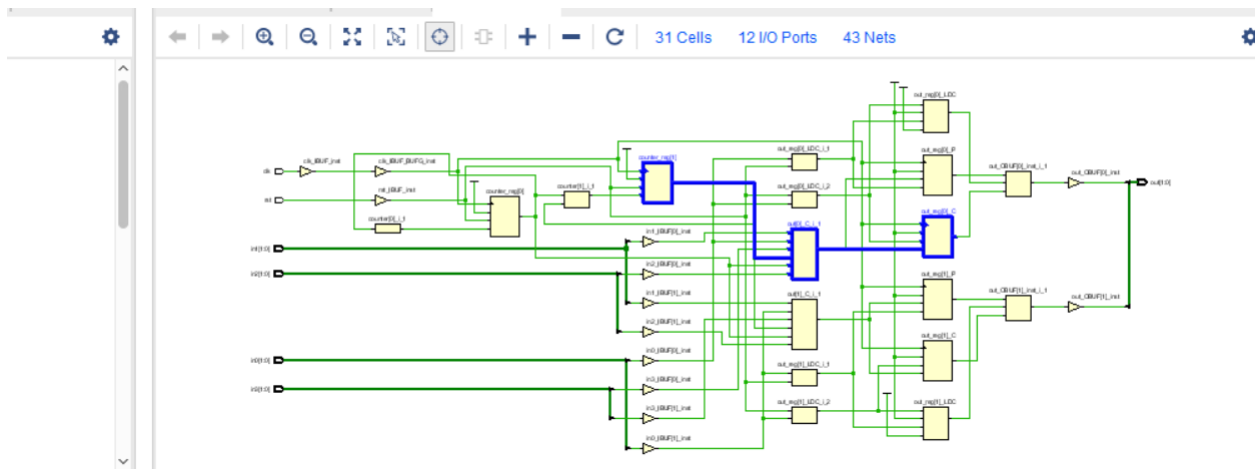
```

32         if (out != in0) begin
33             $display("Error");
34             $stop;
35         end
36
37         #10;
38         if (out != in1) begin
39             $display("Error");
40             $stop;
41         end
42
43         #10;
44         if (out != in2) begin
45             $display("Error");
46             $stop;
47         end
48
49         #10;
50         if (out != in3) begin
51             $display("Error");
52             $stop;
53         end
54
55         rst = 1;
56         #10;
57         if (out != in0) begin
58             $display("Error");
59             $stop;
60         end
61
62         $display("All test cases passed!");
63         $stop;
64     end
65
66 endmodule

```


Constrain:

```
constrain_3.xdc
1  ## Clock signal
2  set_property -dict { PACKAGE_PIN W5    IOSTANDARD LVCMOS33 } [get_ports clk]
3  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
4
5
6  ## Switches
7  set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVCMOS33 } [get_ports {in0[0]}]
8  set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMOS33 } [get_ports {in0[1]}]
9  set_property -dict { PACKAGE_PIN W16    IOSTANDARD LVCMOS33 } [get_ports {in1[0]}]
10 set_property -dict { PACKAGE_PIN W17    IOSTANDARD LVCMOS33 } [get_ports {in1[1]}]
11 set_property -dict { PACKAGE_PIN W15    IOSTANDARD LVCMOS33 } [get_ports {in2[0]}]
12 set_property -dict { PACKAGE_PIN V15    IOSTANDARD LVCMOS33 } [get_ports {in2[1]}]
13 set_property -dict { PACKAGE_PIN W14    IOSTANDARD LVCMOS33 } [get_ports {in3[0]}]
14 set_property -dict { PACKAGE_PIN W13    IOSTANDARD LVCMOS33 } [get_ports {in3[1]}]
15
16 ## LEDs
17 set_property -dict { PACKAGE_PIN U16    IOSTANDARD LVCMOS33 } [get_ports {out[0]}]
18 set_property -dict { PACKAGE_PIN E19    IOSTANDARD LVCMOS33 } [get_ports {out[1]}]
19
20 ## Reset button
21 set_property -dict { PACKAGE_PIN U18    IOSTANDARD LVCMOS33 } [get_ports rst]
22
23
24 ## Configuration options
25 set_property CONFIG_VOLTAGE 3.3 [current_design]
26 set_property CFGBVS VCC0 [current_design]
27 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
28 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
29 set_property CONFIG_MODE SPIx4 [current_design]
```

| Timing | | | | | | | | | | | | | |
|---|-------|--------|--------|-------------|------------------|------------------|-------------|-------------|-----------|-------------|--------------|---|--|
| Intra-Clock Paths - sys_clk_pin - Setup | | | | | | | | | | | | | |
| Name | Slack | Levels | Routes | High Fanout | From | To | Total ... 1 | 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_real0/C | counter_real0/D | 1.086 | 0.751 | 0.335 | 10.0 | sys_clk_pin | s | |