Lab_assignment2

Digital_Design_2023_fallterm@cse.sustech

P1. use **Verilog** to describe the following circuit: the module name is **lab_a2_p1**, with 2 input port **a** and **b**, the bit-width of both **a** and **b** is **3**, the circuit treat **a** and **b** as **signed number** and do the **signed addition operation** on them, there are 2 **output** ports of lab_a2_p1: **sum_flag** and **sum_absolut**. The bit-width of **sum_flag** is 1. If the sum of a and b is **>= 0**, then **sum_flag is 1'b0**, otherwise, **sum_flag is 1'b1**. The bit-width of **sum_absolut** is 3, it is the absolute value of the sum of a and b.(10points)

Tips: while run the simulation on the testbench to test the circuit, the expected waveform is like:

P2. **Build** a testbench to simulate and test the reference circuits, and the modules being tested is named **lab_a2_p1**. (10points)

- **lab_a2_p1** have 2 input ports **a**, **b** and two output ports **sum_flag** and **sum_absolut**. The bit widths of a,b and sum_absolut are all 3, The bit width of sum_flag is 1.
- **a** in the testbench connects to port **a** of **lab_a2_p1**;
- **b** in the testbench connects to port **b** of **lab_a2_p1;**
- **sum_flag** in the testbench connects to port **sum_flag** of **lab_a2_p1;**
- **sum_absolut** in the testbench connects to port **sum_absolut** of **lab_a2_p1;**

NOTE: Before submitting your code to Verilog-OJ, please make sure to use the **monitor** statement only once in `initial` block (there is only one `initial` block). For example:

```
Initial begin    // just for reference
$monitor ("%d %d %d %d", a, b, sum_flag, sum_absolut);
 {a,b} = 6'b000_000;
repeat( /*to be complete*/)  #10  {a,b} = {a,b} + 1;
#10 $finish;
end
```

Tips: The expected waveform are like the pictures of P1 while running the simulation on the testbench,

P3. Design a circuit which get the addition of two three-bit signed numbers, and display the decimal

representation of the result on the two 7-segment tubes, assuming the two 7-segment tubes' name are: **tub_sflag** and **tub_sabsolut**. (10points)

The name of 7-segment tubes	It's **tub_select**(1bit) While it's 1 means tube work, otherwise means not wok	It's **tub_control**(8bit)
tub_sflag	**tub_sflag_sel**	**tub_sflag_control**
tub_sabsolut	**tub_sabsolut_sel**	**tub_sabsolut_control**

The module name is **lab_a2_p3**, with 2 input ports **a** and **b**, 4 output ports **tub_sflag_sel**, **tub_sabsolut_sel**, **tub_sflag_control** and **tub_sabsolut_control**. The bit-width of **a** and **b** is 3, the bit-width of **tub_sflag_sel** and **tub_sabsolut_sel** is 1, the bit-width of **tub_sflag_control** and **tub_sabsolut_control** is 8.

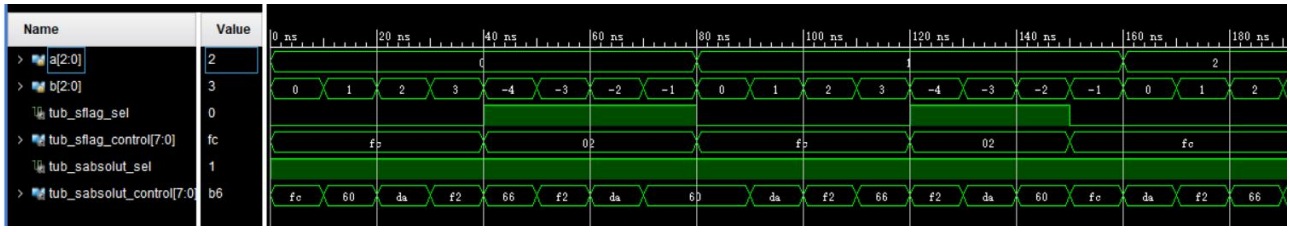
- ✓ **If the sum of a and b is negative**, both two 7-segment tube work, that means the value of both **tub_sflag_sel and tub_sabsolut_sel are 1'b1**, **tub_sflag** display the charactor "-", **tub_sabsolut** display the **absolute value of sum in decimal**.
- ✓ **If the sum of a and b is not negative**, tub_sflag don't work, that means **tub_sflag_sel is 1'b0**, only **tub_sabsolut** work which means **tub_sabsolut_sel is 1'b1, tub_sabsolut** display the **absolute value of sum in decimal.**

NOTE: It is asked to use module lab_a2_p1 and **lab_a2_p3_tub_display** to do the circuit design in **structural manner**. Both these two modules are provided on the OJ, the definition of lab_a2_p1 is exactly same as the module **lab_a2_p1** of this assignment, the definition of **lab_a2_p3_tub_display** is as the content in the following table.

```
module lab_a2_p3_tub_display(
input [3:0] in_b4,  input tub_sel_in,  output tub_sel_out,  output reg [7:0]  tub_control );

   assign tub_sel_out = tub_sel_in;
//tub_control : = {a,b,c,b,e,f,g,"dot"}
   always @ * begin
      case(in_b4)
         4'b0000: tub_control = 8'b1111_1100;    //"0" : abcdef_ _
         4'b0001: tub_control = 8'b0110_0000;    //"1":  _bc_ _ _ _ _ _
         4'b0010: tub_control = 8'b1101_1010;    //"2": ab_de_g_
         4'b0011: tub_control = 8'b1111_0010;    //"3":  abcd_ _ g _
         4'b0100: tub_control = 8'b0110_0110;    //"4": _bc _ _fg_
         4'b0101: tub_control = 8'b1011_0110;    //"5": a_cd_fg_
         4'b0110: tub_control = 8'b1011_1110;    //"6": a_cdefg_
         4'b0111: tub_control = 8'b1110_0000;    //"7": abc_ _ _ _ _
         4'b1000: tub_control = 8'b1111_1110;    //"8": abcdefg_
         4'b1001: tub_control = 8'b1110_0110;    //"9": abc_ _ fg_
         4'b1111: tub_control = 8'b0000_0010;    //"-": _ _ _ _ _ _ g _
         default:
            tub_control = 8'b1001_1110;          //"E": a_ _ defg_
      endcase
   end
endmodule
```

Tips: while run the simulation on the testbench, the expected waveform is like:



P4. Build a testbench to simulate and test the design you make in P3, and the modules being tested is named **lab_a2_p3**. (10points)

- **lab_a2_p3** have 2 input ports **a**, **b** and 4 output ports **tub_sflag_sel, tub_sflag_control, tub_sabosolut_sel, tub_sabsolut_control**. The bit widths of a,b are all 3. The bit width of **tub_sflag_sel, tub_sabosolut_sel** are all 1. The bit width of **tub_sflag_control, tub_sabsolut_control** are all 8**.**
- **a** in the testbench connects to port **a** of **lab_a2_p3**;
- **b** in the testbench connects to port **b** of **lab_a2_p3;**
- **tub_sflag_sel** in the testbench connects to port **tub_sflag_sel** of **lab_a2_p3;**
- **tub_sflag_control** in the testbench connects to port **tub_sflag_control** of **lab_a2_p3;**
- **tub_sabosolut_sel** in the testbench connects to port **tub_sabosolut_sel** of **lab_a2_p3;**
- **tub_sabsolut_control** in the testbench connects to port **tub_sabsolut_control** of **lab_a2_p3;**

NOTE: Before submitting your code to Verilog-OJ, make sure to use the **monitor** statement only once in initial block.

Tips: while run the simulation on the testbench, the expected waveform are like the pictures of P3.

P5. Design a circuit which calculates the number of 1 in the input signal. If the number of 1 is greater than the number of 0, output 1; otherwise, output 0. (10points)

The module name is **lab_a2_p5**, with 4 input port **a,b,c,d**. The bit-width of both a,b,c and d is **1**. There are **2** output ports which are named **more1_somin** and **more1_pomax**, both of them are **1** bit. While using **Sum of Minterm** to implement the circuit, assign the result to **more1_somin,** while using **Product of Maxterm** to implement the circuit, assign the result to **more1_pomax.**

For example:

more1_somin more1_pomax	**number of 1 in the input:**	**a,b,c,d**					
1	**4**	**4'b1111**	**..**				
1	**3**	**4'b0111**	**4'b1011**	**4'b1101**	**4'b1110**	**..**	
0	**2**	**4'b0011**	**4'b0101**	**4'b1001**	**4'b1100**	**4'b1010**	**..**
0	**1**	**4'b0001**	**4'b0010**	**4'b0100**	**4'b1000**	**..**	
0	**0**	**4'b0000**	**..**				

NOTE: please using "assign" to implement the circuit, gates and "always" are not allowed in this assignment

P6. Design a circuit which calculate the number of 1 in the input signal. If the number of 1 is greater than the number of 0, output 1. Implement it using **behavior modeling method**. The module name is **lab_a2_p6**, with 1 input port **in** and 1 output port **more1**. The bit-width of **in** is **4**, the bit-width of **more1** is **1**. (10points)
NOTE: "always" block is expected, don't use any gates or keyword "assign"

P7. Re-implement the design of P6 using **module MUX_16_1 and** design in **structural manner**. The module name is **lab_a2_p7**, with 1 input port **in** and 1 output port **more1**. (10points)
NOTE:
keyword "**always"and any gates are not allowed in this module, "MUX_16_1" MUST be used in your code, the output port more1 MUST connect with the output of the "MUX_16_1".**
TIPS: The module **MUX_16_1** would be provided on the OJ, the defination of **MUX_16_1** is as the content in the following table.

```
module MUX_16_1(input [3:0] sel, input [15:0] data_in, output reg data_out );
  always @ * begin
  case(sel)
       4'b0000: data_out = data_in[0];
       4'b0001: data_out = data_in[1];
       4'b0010: data_out = data_in[2];
       4'b0011: data_out = data_in[3];
       4'b0100: data_out = data_in[4];
       4'b0101: data_out = data_in[5];
       4'b0110: data_out = data_in[6];
       4'b0111: data_out = data_in[7];
       4'b1000: data_out = data_in[8];
       4'b1001: data_out = data_in[9];
       4'b1010: data_out = data_in[10];
       4'b1011: data_out = data_in[11];
       4'b1100: data_out = data_in[12];
       4'b1101: data_out = data_in[13];
       4'b1110: data_out = data_in[14];
       4'b1111: data_out = data_in[15];
  endcase
  end
endmodule
```

P8. Re-implement the design of P6 using **module MUX_8_1 and** design in **structural manner**. The module name is **lab_a2_p8**, with 1 input port **in** and 1 output port **more1**. (10points)
NOTE: keyword "**always" and any gates are not allowed in this module, "MUX_8_1" MUST be used in your code, the output port more1 MUST connect with the output of the "MUX_8_1".**
TIPS: The module **MUX_8_1** would be provided on the OJ, the defination of **MUX_8_1** is as the content in the following table.

```
module MUX_8_1( input [2:0] sel, input [7:0] data_in, output reg data_out );
  always@* begin
  case(sel)
     3'b000: data_out = data_in[0];
     3'b001: data_out = data_in[1];
     3'b010: data_out = data_in[2];
     3'b011: data_out = data_in[3];
     3'b100: data_out = data_in[4];
```

```
      3'b101: data_out = data_in[5];
      3'b110: data_out = data_in[6];
      3'b111: data_out = data_in[7];
    endcase
    end
endmodule
```

P9: Re-implement the design of P6 using module **Decoder_4_16** and design in **structural manner** The module name is **lab_a2_p9**, with 1 input port **in** and 1 output port **more1**. (10points)

NOTE: keyword "**always", "assign" are NOT allowed in this module. "Decoder_4_16" and "or"gate MUST be used in your code, the output port more1 MUST connect with the output of the "or" gate.**

TIPS: The module **Decoder_4_16** would be provided on the OJ, the definition of **Decoder_4_16** is as the content in the following table.

```
module Decoder_4_16( input [3:0] in, output reg [15:0] out );
  always@* begin
    case(in)
        4'b0000: out = 16'h0001;
        4'b0001: out = 16'h0002;
        4'b0010: out = 16'h0004;
        4'b0011: out = 16'h0008;
        4'b0100: out = 16'h0010;
        4'b0101: out = 16'h0020;
        4'b0110: out = 16'h0040;
        4'b0111: out = 16'h0080;
        4'b1000: out = 16'h0100;
        4'b1001: out = 16'h0200;
        4'b1010: out = 16'h0400;
        4'b1011: out = 16'h0800;
        4'b1100: out = 16'h1000;
        4'b1101: out = 16'h2000;
        4'b1110: out = 16'h4000;
        4'b1111: out = 16'h8000;
    endcase
  end
endmodule
```

P10: Build a testbench to simulate and test the circuits from p5 to p9, and the modules being tested is named **lab_a2_p5**, **lab_a2_p6, lab_a2_p7**, **lab_a2_p8**, **lab_a2_p9**. (10points)

- The bit-width of the input port of both **lab_a2_p5**, **lab_a2_p6, lab_a2_p7**, **lab_a2_p8 and lab_a2_p9** are 4, The bit-width of output port **more1 of both lab_a2_p6, lab_a2_p7**, **lab_a2_p8 and lab_a2_p9** are 1 bit, the bit-width of output port **a2_p5_somin and a2_p5_pomax** are 1.
- **in** in the testbench connects to port **a,b,c,d** of **lab_a2_p5**, **lab_a2_p6,lab_a2_p7**, **lab_a2_p8**, **lab_a2_p9 ;**
- **a2_p5_somin** in the testbench connects to port **more1_somin** of **lab_a2_p5;**
- **a2_p5_pomax** in the testbench connects to port **more1_pomax** of **lab_a2_p5;**
- **a2_p6_more1** in the testbench connects to port **more1** of **lab_a2_p6;**
- **a2_p7_more1** in the testbench connects to port **more1** of **lab_a2_p7;**
- **a2_p8_more1** in the testbench connects to port **more1** of **lab_a2_p8;**
- **a2_p9_more1** in the testbench connects to port **more1** of **lab_a2_p9;**

NOTE: Before submitting your code to Verilog-OJ, please make sure to use the **monitor** statement once in `initial` block (there is only one `initial` block).

Tips: while run the simulation on the testbench, the expected waveform is like: