Lab 01: ALU Designs (Sep 19, 2019)

Submission deadlines:

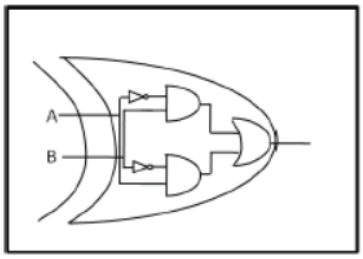
Source Code:	18:30, Sep 24, 2019
Report	23:59, Sep 29, 2019

Objective

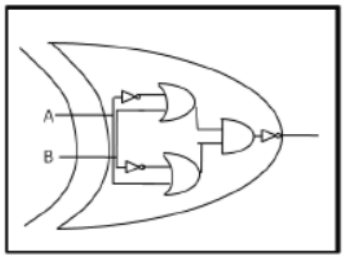
To be familiar with the structural modeling, data flow modeling, behavioral modeling, and module instantiation.

Action Items

0. (25%) Write a Verilog module (**myxor**) that models an **XOR gate** with the **gate primitives** of and, or and not gates **ONLY** (structural modeling). And test your module using the testbench myxor_t.v. Two structure examples of the XOR gate are listed as follows (you may use either of them):



Design 1



Design 2

You have to use the following template for your design.

```
module myxor (out, a, b);
    input a, b;
    output out;

    //add your design here

endmodule
```

1. (25%) Write a Verilog module that models a **1-bit ALU** with and, or, not, **myxor** as basic components. You **must** use module mux4_to_1 (in mux4_to_1.v) to realize the MUX. Test your design by using the testbench lab1_1_t.v.

   Restriction: you CANNOT use xor gate directly. Use myxor as the XOR function instead.

   You have to use the following template for your design.

```
module lab1_1 (a, b, c, aluctr, d, e);
    input a, b, c;
    input [1:0] aluctr;
    output d, e;

    // add your design here

    endmodule
```

aluctr[1]	aluctr[0]	fucntion
0	0	{e,d} = a + b + c
0	1	d = a and b e = 0
1	0	d = a nor b e = 0
1	1	d = a xor b e = 0

2. (15%) Re-write the module using **continuous assignments** (data flow modeling) and re-test your module using the testbench file lab1_1_t.v.

   You have to use the following template for your design.

```
module lab1_2 (a, b, c, aluctr, d, e);
  input a, b, c;
  input [1:0] aluctr;
  output d, e;

  // add your design here

endmodule
```

3. (15%) Re-write the module using **behavioral modeling** and re-test your module using the testbench file lab1_1_t.v.

   You have to use the following template for your design.

```
module lab1_3 (a, b, c, aluctr, d, e);
  input a, b, c;
  input [1:0] aluctr;
  output d, e;

  // add your design here

endmodule
```

4. (20%) Write a Verilog module that models a **4-bit ALU** and test your module using the testbench file lab1_4_t.v. The 4-bit ALU must be implemented by using four 1-bit ALUs (i.e., four instances of previous modules) with necessary interconnects. The addition function is designed for unsigned addition.

You have to use the following template for your design.

```
module lab1_4 (a, b, c, aluctr, d, e);
  input [3:0] a,b;
  input [1:0] aluctr;
  input c;
  output [3:0] d;
  output e;

  // add your design here

endmodule
```

Bonus

1. (10%) Change your 1-bit ALU from lab1-1, replace “a nor b” to function “cmpFunc(a,b)”. Test your module using the testbench file lab1_bonus1_t.v.

   You can use any method from gate-level, continuous assignments or behavioral modeling.

   You have to use the following template for your design.

```
module lab1_b1 (a, b, c, aluctr, d, e);
   input a, b, c;
   input [1:0] aluctr;
   output d, e;

   //  add your design here

endmodule
```

aluctr[1]	aluctr[0]	fucntion
0	0	{e,d} = a + b + c
0	1	d = a and b e = 0
1	0	d = 0 e = cmpFunc(a,b)
1	1	d = a xor b e = 0

cmpFunc():

With a,b,c are inputs, we need to compare a and b. The c represents the result of the previous bit. If there is no previous bit, c is 0.

The idea is to design a comparator that can tell whether a is bigger than b. The **pseudo code** for cmpFunc() is also listed as follows.

```
compareFunc(a, b, c):
    if a > b:
        e = 1
    else if a < b:
        e = 0
    else if a == b:
        if c == 0:  e = 0
        if c == 1:  e = 1
```

2. (5%)Implement the 4-bit ALU by using the 1-bit ALU from Bonus-1 and test your module using the testbench file lab1_bonus2_t.v

You have to use the following template for your design.

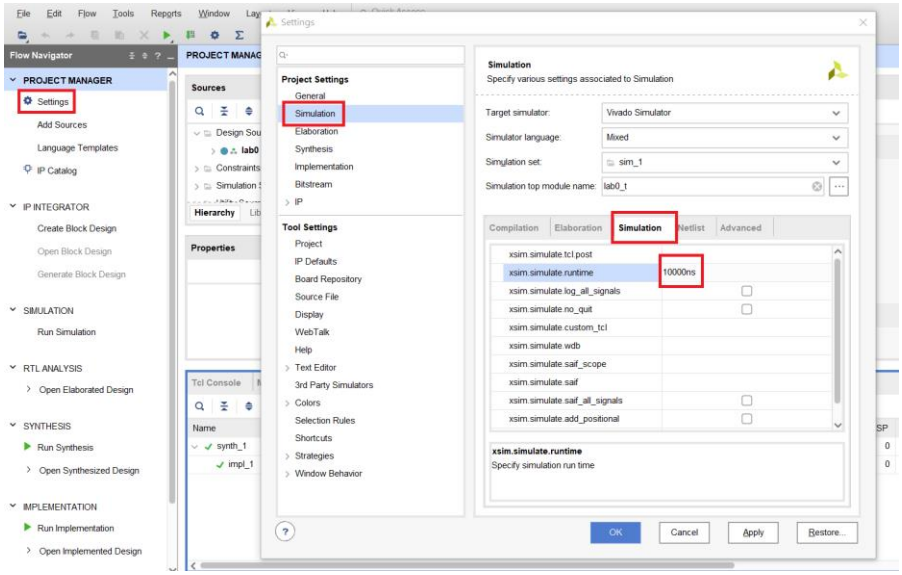
```
module lab1_b2 (a, b, c, aluctr, d, e);
    input [3:0] a,b;
    input [1:0] aluctr;
    input c;
    output [3:0] d;
    output e;

    //  add your design here

endmodule
```

Attention

✓ When you simulate lab1_4 and bonus2, you have to change your runtime to 10000ns in “Simulation Settings” before you run the simulation.



✓ You can add a **$monitor** in your testbench to show all the information of your inputs and outputs during the simulation.

✓ You should hand in all the five source file including **myxor.v, lab1_1.v, lab1_2.v, lab1_3.v and lab1_4.v.** You can also include **lab1_b1.v, lab1_b2.v** if you want extra score. (**Please do not hand in any compressed files, which will be considered as an incorrect format.**)

✓ You should also hand in your report as **lab01_report_*StudentID*.pdf** (i.e., lab01_report_107080001.pdf).

✓ You should be able to answer questions of this lab from TA during the demo.

✓ Full-Adder Reference is at below: