**CO project 2 – Simple Single Cycle CPU**

*ID: 0340249*

*Name: 孫聖*

1. **Introduction**

In this lab, we are required to implement a simple single cycle CPU, consisting of two adders, one ALU, one shift-left-two unit, one sign-extend unit, one decoder, one ALU-control unit and three multiplexers.

1. **Procedure**

I start from the basic components. For each component, I write a module to test its functionality. Then I move to the more complicated units(decoder and ALU-control).

In the implementation of decoder, I add two more output: isOri\_o and isBne\_o to recognize operation ori and bne respectively. I divide the instructions into 2 groups, R-type and the rest. For R-type, I set the ALUOp to 010 while for others I randomly choose ALUop for them.

As for ALU-control unit, First I decide which operation should be executed according to the ALUOp and function code, then I map it to the ALU control number using CASE statements. At this stage, I change some ALU control to meet the operations required in this lab, e.g. SLL, SLRV, LUI.

Lastly, I fill in the ports in Simple Single CPU to connect all the components together.

1. **Results**

The results of running three test cases are correct. But when I try one of my own test case, it doesn’t work as I expect, so now I am trying to solve this problem.

1. **Conclusion**

With the help of Lab1, I have deeper understandings of Verilog and I don’t need to worry about the basic syntax. But I still make some silly mistakes, for example, when I try to assign a binary value, I forget to add 4’b in front of the values, resulting in the misinterpretation of them as decimals. During the implementation, I want to figure out the relationship among Opcode, ALUOp and ALU-control, but I fail to, so I just resort to using CASE statements.