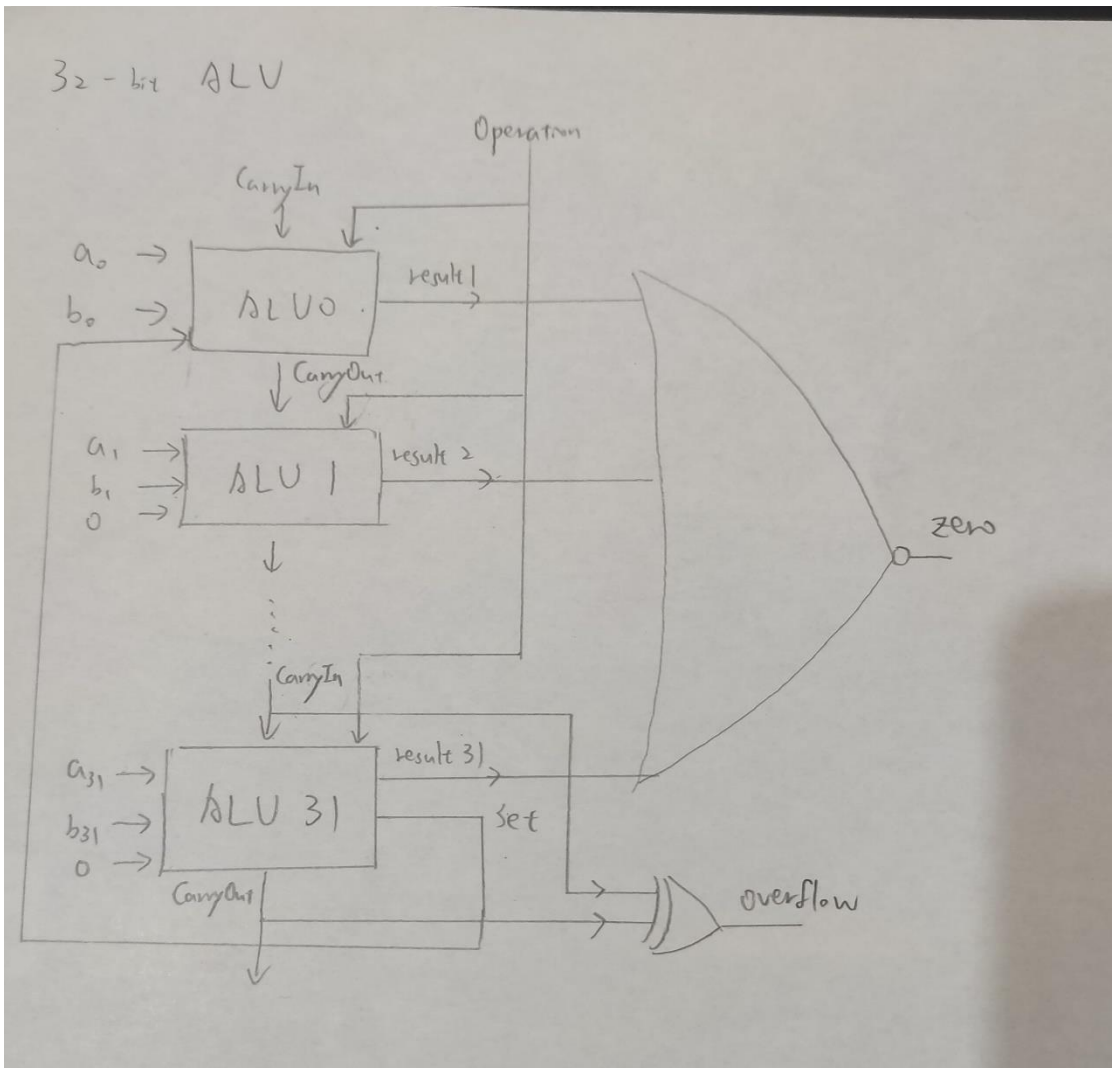
Computer Organization

Architecture diagrams:



Hardware module analysis:

首先，用 ALU_control 來判斷現在要做甚麼，藉由這樣的設計，可以輕鬆的執行出想要得到的結果。在開始之前，會先有一個 negative reset，這時候我先初始化 result, zero. Cout 和 overflow。接著，我用 case 來判斷是哪種 operation(and, or,...)，如果是 and 或 or，就直白的用邏輯式來 assign result 的值，並把 carryout、overflow 都設成 0(方便理解 code，其實這裡也可以不用)；而 addition 的部分，我用 32 的 cout 來存每個 ALU 的 carryout，因為每個 ALU 的 carryout 都會影響到後面的運算，最後在用教授講義上的判斷方法來判斷是否有 overflow。

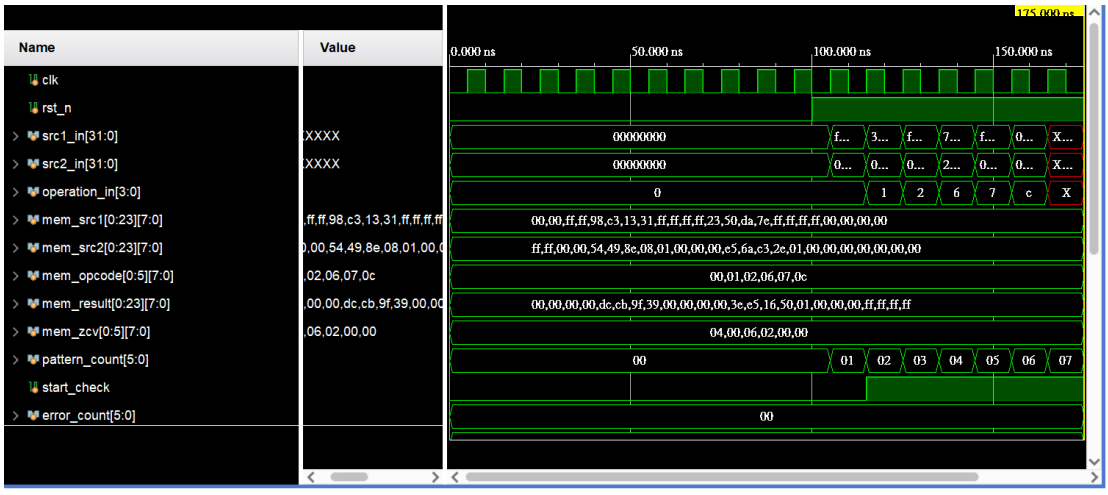
- Overflow if result out of range
  - Adding +ve and –ve operands, no overflow
  - Adding two +ve operands
    - Overflow if result sign is 1
  - Adding two –ve operands
    - Overflow if result sign is 0

至於 subtraction，和 addition 也很類似，差別只在把運算的+改成-，並稍微修改一下 overflow 的判斷式，這裡用的也是教授講義裡所寫的方式

- Overflow if result out of range
  - Subtracting two +ve or two –ve operands, no overflow
  - Subtracting +ve from –ve operand (ex. (-b) – a)
    - Overflow if result sign is 0
  - Subtracting –ve from +ve operand (ex. b – (-a))
    - Overflow if result sign is 1

到了最後的 set less than，首先我判斷 src1 和 src2 的正負號，也就是第 32 個 bit(src[31])，如果正負號不一樣，那就直接藉由正負號判斷；如果一樣，則比較他們兩個的值。至於結果，假如 src1<src2，result 就會是 1，反之，若是 src1>=src2，則 result 就會是 0。

Experiment result:



符合

Problems you met and solutions:

First, before this semester, I rarely use Verilog, so I need to google for a lot of things, even some basic functions (if, nor, etc). Next, I don't know how to handle carryout, because I need to consider from ALU0 to ALU31. After I google some website, and scroll through professor Li's handout, I finally figure out how to handle it correctly. Last, in the beginning, I have no idea how do detect an overflow. Although I know how an actual ALU detects it, I just don't know how to implement the data flow using Verilog. Nonetheless, thanks to my roommates, they help me at some tiny, but crucial points, which allows me to finish this lab successfully.

Summary:

In this lab, I learn how a dataflow diagram is like in ALU. Also, I know more details in ALU and get more familiar with verilog. ALU is a very basic and common module in any other design, so understanding this lab is very important. I hope I can keep learning more about computer organization.