Lab1: ALULab

EE312 Computer Architecture

Professor: Minsoo Rhu

Student names: Maro Han, Sanghyun Kim

Student Numbers: 20150912, 20150146

Introduction:

We believed the lab was aimed to make us get familiar with the basics of Verilog. This includes learning the modeling concept and understanding sequential and combinational logic. Moreover, we learned when and how we should use continuous assignment (using wire connections) and procedural assignments (using registers).

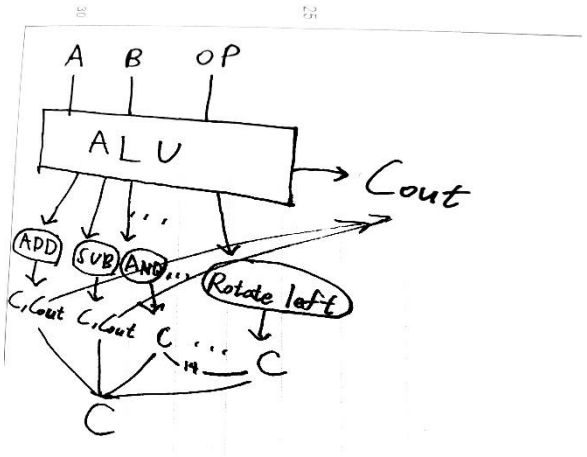
The design of ALU was behavioral, and used combinational logic. For assignment we used procedural assignment to store values in registers.

Design:

We managed to complete all the tests successfully by using one model. Since, we did not repeat lines of code, we didn't find any need to implement separate models.

In the model we used 16 if statements to handle the 16 operations our ALU was supposed to handle. The code is able to distinguish the operations by reading the input OP code and comparing it to the predefined OP code.

For add operation we detected overflow by detecting two cases. One, when both inputs are positive and output is negative. Two, when both inputs are negative and output is positive. For sub operation we used the same logic after inverting the sign of the subtracting input (if A-B, A is the subtracted input, by is the subtracting input).



Above figure is a brief illustration of our design.

Implementation:

Before module we defined different OP codes for different operations (etc. `define OP_ADD 4'b0000). This was compared to the input OP code to determine what operations to execute.

We stored the output and Cout (overflow flag) as registers. Different variables we used inside the code was also stored as registers as we were using the procedural assignment. As briefly mentioned before in the introduction we used 16 if statements to distinguish between the 16 operations. All the if statements were nested inside an always procedure coupled with blocking assignments keeping the combinational logic consistently through the code.

For all operations we used predefined Verilog operators to carry out the operations except for NAND, NOR and arithmetic right shift because there were no predefined Verilog operators for said operations. To solve this problem for NAND and NOR operations, we simply inverted the respective counterparts of the operations (ex: NAND operation = ~(A&B)).

During debugging & testing, we found out that the predefined operator for arithmetic right shift operator functioned just like the logical right shift operator. Hence, we were forced to implement the arithmetic right shift operator on our own. First, temporarily stored the MSB of the input into a single-bit variable "a" which was stored as a register in our code. Second, we did a logical right shift on the input. Finally, we amended the MSB of the result with "a" to obtain the output.

For Add and Sub operations we had to detect overflow. For add operation we read the MSB of both the inputs and the sum of inputs (output). The overflow flag is trigged in two cases. The first case is when the MSBs of both inputs are 1, and the MSB of the output is 0. The second case is when the MSBs of both inputs are 0, and the MSB of the output is 1. For the sub operation we used the same logic as the add operation, using 2's complement.

Evaluation:

We passed all 50 tests in ALU_TB.v. We believe our code design was intuitive, and out implementation was clear and simple.

Discussion:

Because we passed all 50 tests there are no problems that we experienced running our code. However, in hindsight more explanation between using combinational and sequential logic. We had to teach ourselves when to used assign, and always operations and how to couple these with wire and register assignments. With that said, we thought the ALULab description pdf was clearly written and easy to understand.

Conclusion:

I assume because it was the first lab assignment, the assignment was focused more on getting familiar with Verilog language and using modelsim. For us, the assignment met that goal and we are looking forward to using the acquired knowledge and designing other programing assignments.