Fall 2018 / EC311
Lab 2
Due Dates:
Section B1 - October 16, 2018
Section B2 - October 12, 2018
Section B3 - October 16, 2018
Section B4 - October 11, 2018
Section B5 - October 10, 2018

1. Goals

- Introduction to Verilog modular code design.
- Introduction to behavioral Verilog.
- Designing simple combinational circuits.

2. Overview

In Lab 2, you will design a simple 4-bit Arithmetic-Logic Unit (ALU), that will be able to perform addition and multiplication, among other functions.

2.1. ALU I/O

The ALU must have two unsigned 4-bit inputs named ‘A’ and ‘B’ as well as a 2-bit operation code named ‘S’. The ALU must output the result with an 8-bit output named ‘Y’.

2.2. ALU Components

Each module should be named after the component, not the lab or problem number.

1. **Adder** - Design a 4-bit adder module with behavioral Verilog. You cannot reuse the adder from Lab 1. The 4-bit binary adder module must have two 4-bit inputs and one 8-bit output. Simulate the adder to be sure it works. You may borrow or reuse code from the Lab 1 test bench.

2. **Multiplier** Design a 4-by-4 multiplication module, with two 4-bit inputs and one 8-bit output. This module should perform unsigned multiplication. You must use behavioral Verilog, so this module should be fairly simple. Simulate the multiplier to be sure it works.

3. **Concatenator** - Our ALU includes a concatenation functionality, which is implemented in a Concatenator module. Design a concatenator module, which includes two 4-bit inputs named ‘A’ and ‘B’ and one 8-bit output named ‘AB’. The output ‘AB’ is the concatenation of the two inputs with A in the most significant bits. Simulate the buffer module to be sure it works.

4. **Shifter** - Design a Left Shift module with inputs 4-bit inputs A and B. The module should extend the input A to 8-bits and shift the 8 bit signal left by B bits. New bits shifted in should be 0. The output should be the 8-bit shifted result. Simulate the module to be sure it works. Note that if B is larger than 7, all of the bits in A will be shifted out of AB, producing a result of 0x00.

5. **Multiplexer** - A multiplexer is a device that has N control inputs and 2^N data inputs. In Figure 1, the multiplexer is referred to as Mux, which has four 8-bit inputs (A, B, C, D), and one 2-bit control signal (S). Whenever S = 2'b00, the output Y = A; if S = 2'b01, then Y = B and so on.

   You will need to design such a multiplexer for the ALU. The easiest way to design a multiplexer in Verilog is to use a case statement. You may refer to your textbook, class notes or the internet for details how to use it. Do not forget that the case statement in Verilog has to be included in a procedural block (usually an always statement).

   You must simulate this module to be sure it works. It is acceptable to set unique constant values for each of the A, B, C, and D inputs and cycle through each value of S.

Combine all the modules you have created into a single top level module named 'alu_module'. Your top level module should instantiate five modules: Concatenator, Binary Adder, Left Shift, Multiplier, and Multiplexer. It should have two 4-bit inputs, A and B, a 2-bit control S, and an 8-bit output.
The 2-bit operation code control input is specified as follows:
00 - Concatenation
01 - Binary addition
10 - Left Shift
11 - Multiplication

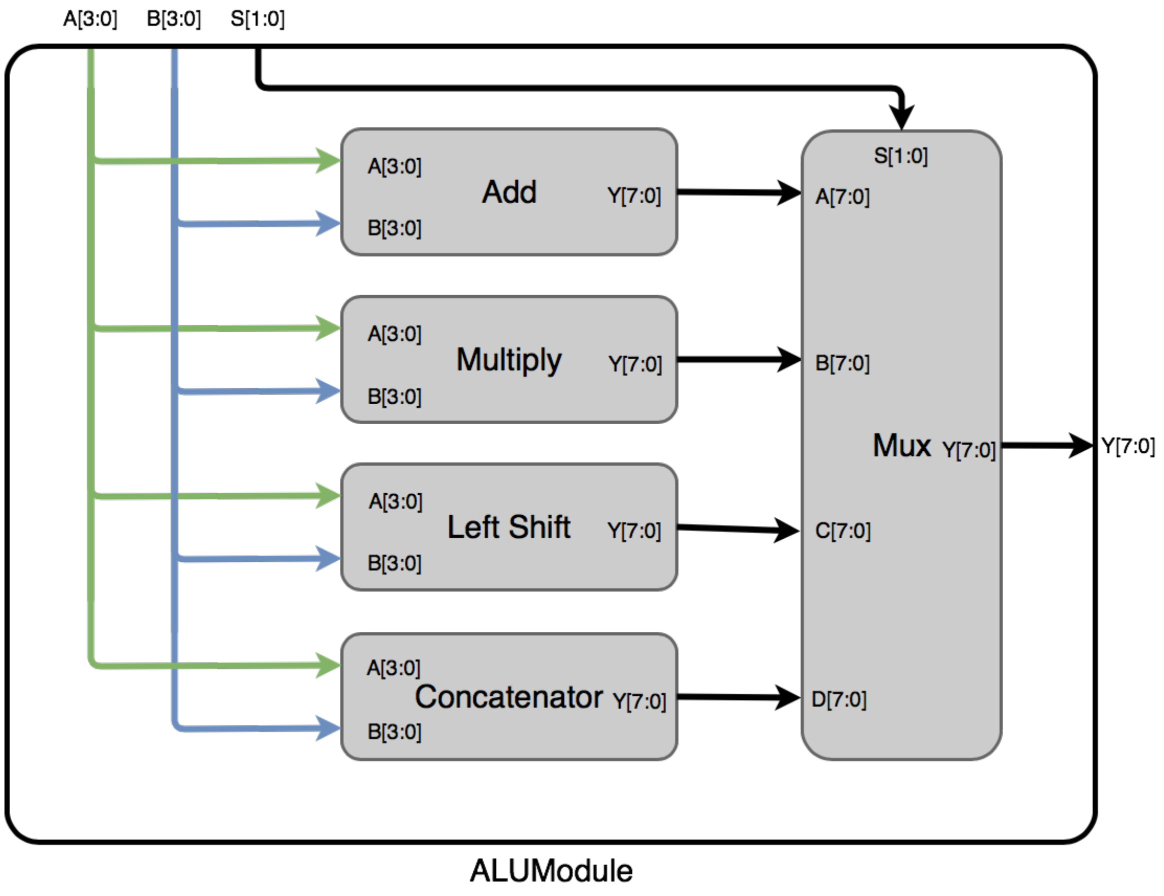

Figure 1: ALU block diagram.

3. Deliverables

You need to show your design hierarchy (under Design $\rightarrow$ Implementation window), your Verilog code, and top module simulation of the ALU to the TA. You don't have to implement your code on the FPGA board for this lab.

Your design has to abide by the following requirements. Failure to follow the instructions will result in a grade reduction.

1. Use multi-bit vector/bus/array (just different ways of naming) instead of 1-bit signals for A, B, S and Y.
2. Follow the input/output specifications.
3. For each test bench, you need to have complete test cases (except for the multiplexer). This means you need to cover all the possible combinations of input values to prove your logic is right.
4. Be hierarchical (consisting of other sub modules), use separate file for each module. Test each module and demonstrate that each works as a standalone unit. It's generally good engineering practice to test submodules before going to the next level. You must turn in a test bench for each module.

You must turn in a lab report describing what you did in this lab, how you did it and what happened when you did it. The lab report should be broken up into these sections: Objective, Methodology, Observation (can include screenshots timing diagrams, schematics, etc.) and Conclusion. Your report should contain enough detail such that any other engineer can replicate your experiments after reading your lab report. Please include all the required details in your lab report. The lab report should be 2-3 pages long with figures and diagrams.

You must submit your report in PDF form to TurnItIn. The PDF should contain your lab report and your Verilog code (in text form). Don't include screenshots of your code in the PDF.

4. Due date

Lab2 is due on the following dates:
Section B1 - October 15, 2018
Section B2 - October 12 , 2018
Section B3 - October 16, 2018
Section B4 - October 11, 2018
Section B5 - October 10, 2018