Please attach this cover sheet to your completed homework

Homework3 (take-home Quiz)
SystemVerilog HDL combinational logic modeling and simulation

Name: _____

Since this is a take-home quiz:
please work alone – do not collaborate!

Exercise	Course outcome	Grade
Homework3	2.a, 7.b	/30

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.

7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction: In this assignment we are called upon to implement a very basic version of an arithmetic logic unit capable of four simple operations. Each of these operations is specified by an opcode, 0-3 and correspond to addition, two bit left shift, bit wise and as well as ignoring the inputs and passing out 1. In order to do this we will utilize case statements to apply the proper operation depending on the provided op code. Continuus assigns however will not function here as the output must change regularly so the always assign statements are used here. Additionally, in the test bench, looping will be used to quickly pass every possible option to every possible

Implementation: The implementation of the ALU in this case is incredibly simple. It only implements four simple arithmetic operations using syntax that would be familiar to any programmer. The complication comes in in the test bench.

Module: The novel concept introduced in this lab is the procedural assignment and case statements. The case statements are used to perform an operation based on the provided op code. These can be seen in basic_alu.sv lines 9-12.

Test bench: The test bench in this case contains more complexity. It contains nested four loops to produce an enormous variety of inputs (every possible one in fact). The four loops complete one loop through the fifteen options for b before incrementing a and then looping through the possibilities of b again. Eventually it makes it way through all combinations of both and then increments the opcode.

Proof of success:

Success is demonstrated by the figures captured below.

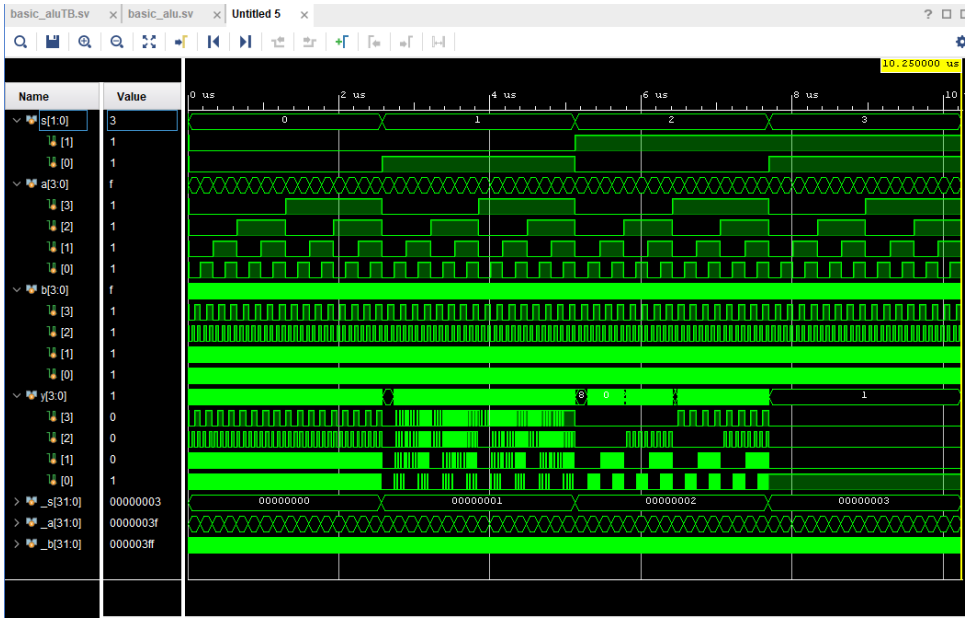

Figure 1: All possible combinations of s, a and b are demonstrated here as well as output y.

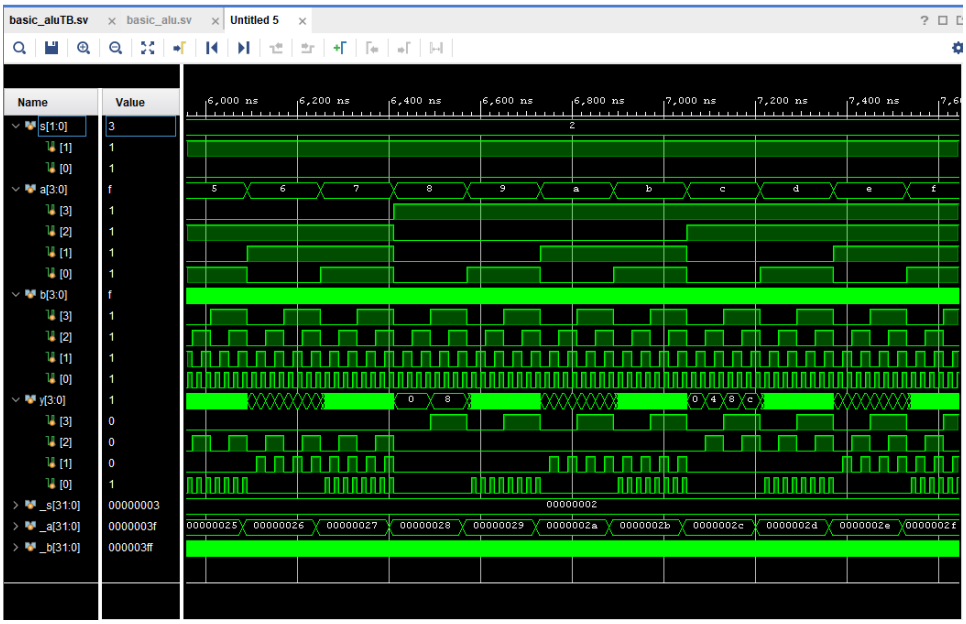

Figure 2: a zoomed in capture of the above diagram for clarity in demonstrating success.

Additionally, to demonstrate functionality if we take the function along the 6.600ns as an example

S = 2, a = 9, b = 3 y = 1. This is what we would expect from a logical shift right and confirms the functionality of the code.

Finally, errors in the code were caught by an assert statement when the code is sabotaged. When opcode 00 was changed to implement a-b the assert statement on lines 22-23 in basic_aluTB.sv produced an alert saying that the code was not performing to specifications.

Summary: In this assignment, case statements, continuous assignments, loops and assert statements were successfully leveraged to create a simple alu and a test bench which evaluates all possible op codes and values it could operate on.