title

your name April 8, 2013

# Introduction

Introduction with problem overview, your design procedure, and rationale.

# Interface

This section explains the input and output relationships of the design, so it can be treated as a black box. I have essentially provided you the inputs, outputs, and inouts, but I want to know you understand why. What inputs are used and how? What outputs are used and how? What sequence of commands need to be performed in order to operate the module? What form does the data get sent in (binary, two’s compliment, excess code, etc.)? How are they grouped?

**Adder**:

The *inputs* include the *add\_in1* and *add\_in2* which take the 32-bit data register outputs from the Instruction Decoder Extend module. The *add\_out* *output* essentially adds the 2 inputs together and passes them to the Instruction Fetch Multiplexer.

**ALU Control:**

The *input funct* (function bits) is a 6-bit instruction set from the Instruction Decoder Extend module. The *ALUop* (ALU operation type) *input* is a 2-bit that determines whether it is add, subtract, etc. *Select* is a 3-bit *output* signal that is created from both the funct and ALUop inputs. This signal dictates what operation the ALU should do.

**Multiplexer:**

Put something here

**ALU:**

There are 3 *inputs* of *a, b,* and *control*; 2 outputs of result and zero. Input a is supplied by the Instruction Decoder Extend. Input b is from the multiplex. Both a and b are 32-bit. Input control is a 3-bit output from the ALU Control. *Output* *zero* (1-bit) goes to the MEM branch while *result* (32-bit) goes to the MEM data memory and MEM/WB latch.

**Execution Memory:**

This acts as a pipeline and handles the inputs and outputs from the 5 modules above.

# Design

This is the internal design of the item. Design description and explanation, including any pictures, charts, etc.

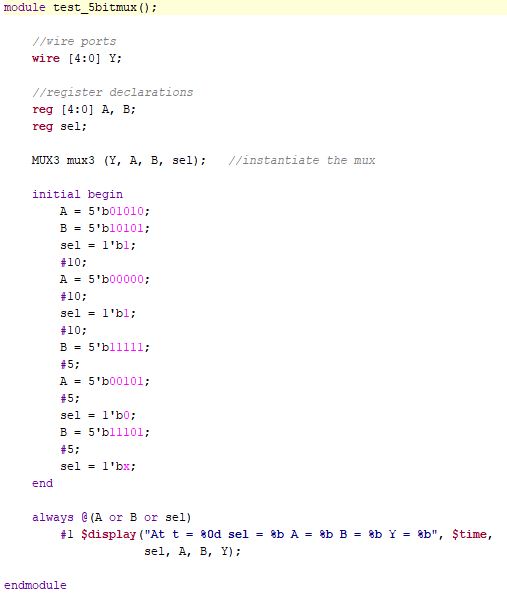
# Implementation

The Verilog code and explanations of why you implemented this way. There are many ways to implement a given design in verilog. For instance why choose a case statement or ifs? Why did you trigger on a negedge verses any signal change?

# Test Bench Design

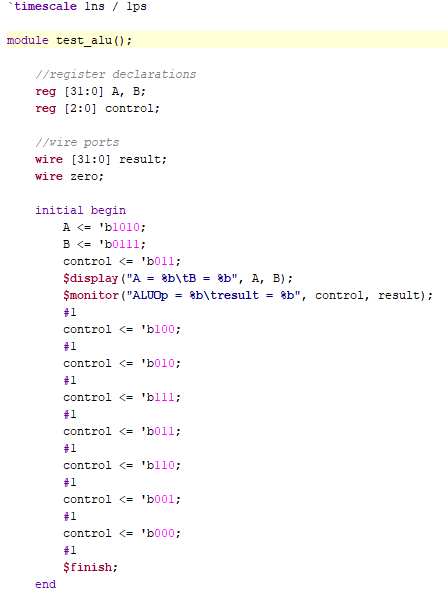
This is where you discuss the test benches you wrote, and what they were designed to test. You should discuss expected errors as well as random errors. Be sure to include your Verilog code.

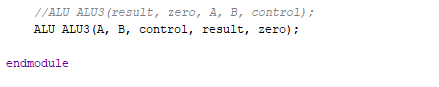
test\_5bitmux.v



Put something here for the mux.

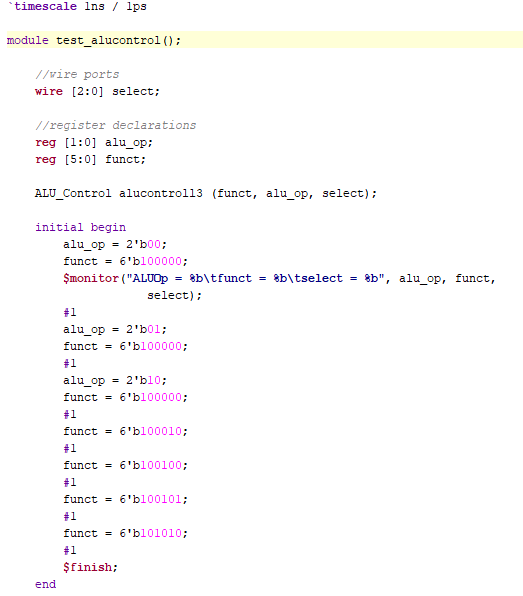
test\_alu.v





In the ALU test bench, the control input is filled with signal instructions that dictates what operation the ALU should perform. A and B are just values to be operated by the ALU, depending on what the control is.

test\_alucontrol.v



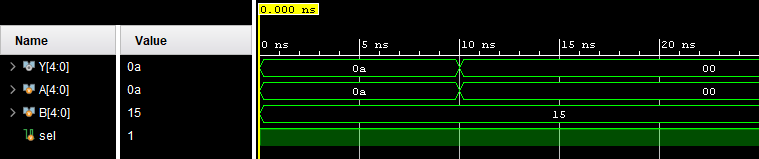


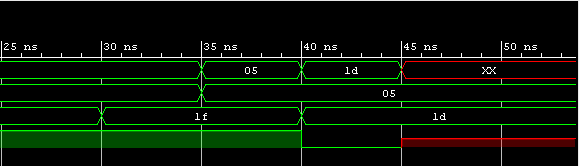
In the ALU Control, alu\_op is initialized with the operation type (could be add, sub, etc.) and the funct is an instruction (should be from the ID/EX latch but we initialized it here).

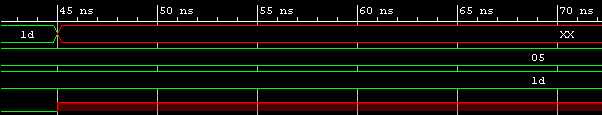
# Simulation

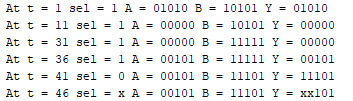
In this section you should show the results of your simulation, such as timing diagrams and explain any design issues you had to deal with.

MUX3:

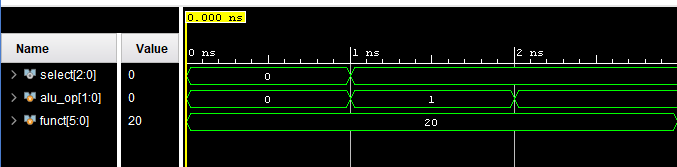


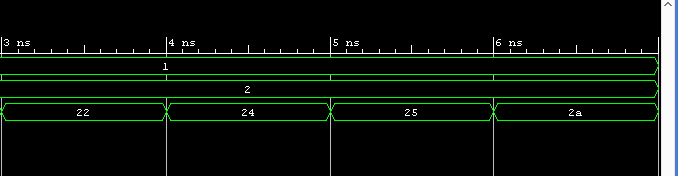


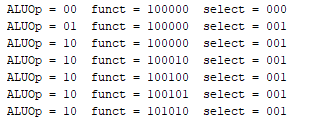




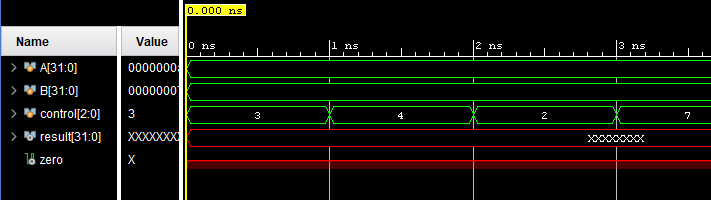
ALU\_Control:

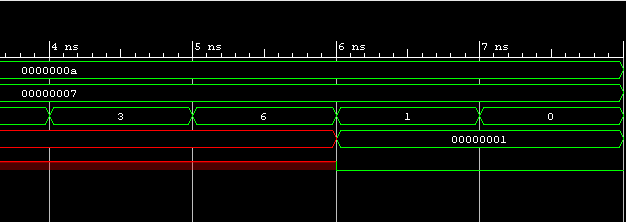


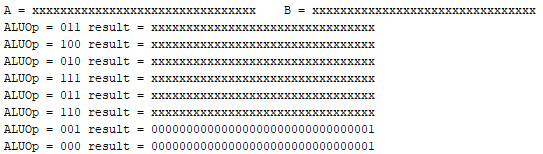




ALU:







# Conclusions

Overview the main points you want to stick in people’s minds and answer key questions you want to stick in people’s minds. Did it work? How well? What would you have done differently? What did you learn?