Project 3 Report

I. Project Introduction

Project 3 is about building a ALU module using verilog HDL. This is a relatively simpler project since more time is given for us to get more familiar with this HDL. Therefore, the report will be divided into two parts. One is for project requirements, while the other will focus more on few understanding of the language.

II. Main work

To simulate the ALU module and its control unit, there are three files created, namely ALU.v, ALU_control.v, test_ALU.v. For testing the functioning of the project, one can easily copy the instructions and data into the given places in test_ALU.v file. The output will be display in the terminal and clearly separated with instruction labels. The following is the logic flow of implementation and my own test case output display.

A. ALU_control

First is the ALU control part. This part is used to complete some data transfers between the fetched instructions and ALU module where the data is finally processed. Since it is not required to fetch data from register files, the inputs of ALU_control module are the following: i_dataIn, gr1 and gr2. They stands for the instruction fetched, value in rs, value in rt (if any). The outputs are ALUCr, reg_a and reg_b, meaning the control bits generated and final values to be used in operations of ALU module.

According to the project instruction, the ALU module should be able to handle 32 instructions. However, many of them may share the same basic function of ALU module, resulting in the fact that the ALUCr generated only have 11 cases. The cases are all shown in the parameters declared in ALU_control file. ALU_control module will generate ALU control bits according to the instruction code given. ALU_control module will also finish data modification like sign or zero extension. Finally output the data ready for ALU module to operate as reg_a and reg_b.

B. ALU

With given ALU control bits from ALU_control module, work for ALU is simple and straight forward. For this module, the inputs are ALUCr, reg_a, reg_b, while the outputs are c, zero, overflow, negative. There are some other inner registers which are, result, reg_hilo, zf, nf and of. They are used to store the process results and assign them to the output. Reg_hilo is for special handling of mult and div type instructions. Also note that result is of 32 bits wide, which simplifies the overflow detection logic greatly.

For mult, multi, div, divu, as it is required in the tutorial, I simply used * and / to implement and created a reg_hilo inside the ALU module. In real cases, since the result of these instructions are not of the same length as other instructions, the high and low registers are in different place and the multiplication and division operation are also done in different place. In multiplication instruction, the multiplicand is placed in a 32 bit register while the multiplier is placed at the lower 32 bits inside the hilo register. The whole multiplication process is done be multiply the right most bit in hilo register with multiplicand, plus it to the left 32 bits of itself, and then do a left shift until the multiplier is all gone. For division, the initial placement is the same. Multiplicand is now divisor and multiplier is now dividend. The main difference is now dividend is subtracted from the upper 32 bits and the shifting direction becomes left.

Other basic function of ALU module is implemented with operators. For required flags, zero flag is detected by comparing the result to 0. Negative flag is detected by. Directly checking the most significant bit of the result. Overflow flag is detected by checking whether the carry out of the most significant bit is equal to the carry in of the most significant bit.

C. test_ALU

This is a simple test banch file adapted from given example file. It contains all my test cases.

D. Test cases

(1) sll

(2) sllv

(3) srl

(4) srlv

(5) sra

(6) srav

(7) add

(8) addu

(9) sub

(10)subu

(11)mult

(12)multu

(13)div

(14)divu

(15)and

(16)nor

(17)or

(18)xor

(19)slt

(20)sltu

(21)addi

(22)addiu

(23)andi

(24)ori

(25)xori

(26)slti

(27)sltiu

(28)beq

(29)bne

(30)lw

(31)sw

III. Reflection