ALU Module

SystemVerilog code for the ALU module with basic operations, the ALU can carry out 4 functions: addition, subtraction, OR and AND.

// alu.sv

// RISC-V ALU Module

// Ver: 1.1

// Date: 23/11/22

module alu #(parameter n = 32)(

    input logic [3:0] AluOp, //4 bit operation code

    input logic [n-1:0] A, B, //32 bit inputs

    output logic [n-1:0] AluOut, //32 bit outputs

    output logic zflag

    //what flags are needed??

);

always\_comb

begin

    zflag = 1'b0; //default assignments

    AluOut = A;

    case(AluOp)

    4'd0: AluOut = A + B; //ADD

    4'd1: AluOut = A + (~B + 1); //SUB

    //ALU operations

    //...

    4'd9: AluOut = A | B; //OR

    4'd10: AluOut = A & B; //AND

    default: AluOut = 0;

    endcase

    //zflag = AluOut ? 1'b1 : 1'b0;// dont think this is correct

    if(AluOut ==  '0)

        zflag = 1'b1;

end

endmodule

This is the testbench to test the ALU module

// alutestb.sv

// RISC-V ALU Test Bench Module

// Ver: 1.0

// Date: 23/11/22

module alutestb;

parameter n = 32;

logic [3:0] AluOp; //4 bit operation code

logic [n-1:0] A, B; //32 bit inputs

logic [n-1:0] AluOut; //32 bit outputs

logic zflag;

alu #(.n(n)) alu1 (.AluOp(AluOp),.A(A),.B(B),.AluOut(AluOut),.zflag(zflag));

initial

    begin A = 32'd15; B = 32'd2;

        #10 AluOp = 4'd0; //ADD

        #10 AluOp = 4'd1; //SUB

        #10 AluOp = 4'd9; //OR

        #10 AluOp = 4'd10; //AND

        #10;

    end

endmodule

This python code shows the calculates the expected outcome of the ALU testbench

a = 15

b = 2

y1 = a + b

print("A + B = " + str(y1))

y2 = a - b

print("A - B = " + str(y2))

y3 = a | b

print("A OR B = " + str(y3))

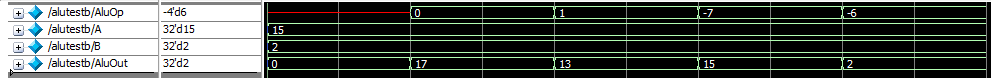
y4 = a & b

print("A AND B = " + str(y4))

This shows the output of the python code



This shows the Modelsim simulation output



The current v1.1 of the ALU is working correctly.

The ALU still needs to have the extra operations added to it.

Sll

Slt

Sltu

Xor

Srl

Sra