Lab Assignment 7

Due: 17 Nov 2020

By: Alex Oswald
EECE 2323

Verilog Code for `pdatapath_top_lab7.v`:

```verilog
`timescale 1ns / 1ps

module pdatapath_top(
        input wire clk,
        input wire rst_general,
        input wire top_pb_clk,
        output[7:0] led
    );

    wire pb_clk_debounced;

    wire [7:0] alu_1st_input, alu_2nd_input;
    wire [7:0] alu_output;
    wire [2:0] ALUOp;
    wire       alu_ovf_flag;
    wire       alu_take_branch_output;

    wire [15:0] instruction;
    //instruction fields
    wire [3:0] opcode;
    wire [1:0] rs_addr;
    wire [1:0] rt_addr;
    wire [1:0] rd_addr;
    wire [7:0] immediate;

    //control signals
    wire RegDst;
    wire RegWrite;
    wire ALUSrc1;
    wire ALUSrc2;
    wire MemWrite;
    wire MemToReg;

    wire [1:0] regfile_write_address;//destination register address
    wire [8:0] regfile_write_data;//result data
    wire [8:0] read_data1;//source register1 data
    wire [8:0] read_data2;//source register2 data

    wire [8:0] alu_result;
    wire [7:0] zero_register;
    wire [8:0] data_mem_out;

    wire [7:0] pc;

    debounce debounce_clk(
        .clk_in(clk),
        .rst_in(rst_general),
        .sig_in(top_pb_clk),
        .sig_debounced_out(pb_clk_debounced)
        );


    // ************* Add the PC logic here ************* //
    pc_logic pclog(
        .clk(pb_clk_debounced),
        .rst(rst_general),
        .count(pc));


    //*********Instantiate Your instruction memory here*********//
    instr_mem instructionmem (
        .a(pc),                      // input wire [7 : 0] a
        .spo(instruction)          // output wire [15 : 0] spo
        );


    //*********Instantiate Your instruction decoder here*********//
    decoder decode (
        .instruction(instruction),
        .opcode(opcode),
        .rs_addr(rs_addr),
        .rt_addr(rt_addr),
        .rd_addr(rd_addr),
        .immediate(immediate),
        .RegDst(RegDst),
        .RegWrite(RegWrite),
        .ALUSrc1(ALUSrc1),
        .ALUSrc2(ALUSrc2),
        .ALUOp(ALUOp),
        .MemWrite(MemWrite),
        .MemToReg(MemToReg)
        );


    //***** random stuff lol *****//
    assign zero_register = 7'b0;//ZERO constant
    assign alu_result = {alu_ovf_flag, alu_output};
    assign led = alu_output;


    //*********Instantiate Your alu-regfile here*********//
    alu_regfile ALU_RegFile (
        .ReadData1(read_data1),
        .ReadData2(read_data2),
        .ALUsrc1(ALUSrc1),
        .ALUsrc2(ALUSrc2),
        .ALUop(ALUOp),
        .Instr_i(immediate),
        .zero_register(zero_register),

        .rst(rst_general),
        .clk(pb_clk_debounced),
        .wr_en(RegWrite),
        .rd0_addr(rs_addr),
        .rd1_addr(rt_addr),
        .wr_addr(regfile_write_address),
        .wr_data(regfile_write_data),

        .input_1(alu_1st_input),
        .input_2(alu_2nd_input),
        .result(alu_output),
        .ovf(alu_ovf_flag),
        .take_branch(alu_take_branch_output)
        );

    //**********Instantiate Your data memory here**********//
    data_memory datamem (
        .rst(rst_general),
        .clk(pb_clk_debounced),
        .write_enable(MemWrite),
        .addr(alu_output),
        .write_data(read_data2),
        .read_data(data_mem_out)
        );

    //**********Mux for regfile_write_data**********//
    assign regfile_write_data = MemToReg ? data_mem_out : alu_result;

    //**********Mux for RegDST**********//
    assign regfile_write_address = RegDst ? rt_addr : rd_addr;

      //********** Instantiate the VIO here **********//
    vio_0 vio (
        .clk(clk),                              // input wire clk

        .probe_in0(alu_output),                  // [7:0] alu_out
        .probe_in1(alu_ovf_flag),              // [0:0] alu_ovf
        .probe_in2(alu_take_branch_output),      // [0:0] take_branch
        .probe_in3(read_data1),                  // [7:0] ReadData1
        .probe_in4(read_data2),                  // [7:0] ReadData2
        .probe_in5(alu_1st_input),              // [7:0] input1
        .probe_in6(alu_2nd_input),              // [7:0] input2
        .probe_in7(regfile_write_data),          // [8:0] WriteData
        .probe_in8(data_mem_out),              // [8:0] DataMemOut
        .probe_in9(opcode),                      // [3:0] opcode
        .probe_in10(rs_addr),                  // [1:0] rs_addr
        .probe_in11(rt_addr),                  // [1:0] rt_addr
        .probe_in12(rd_addr),                  // [1:0] rd_addr
        .probe_in13(immediate),                  // [7:0] immediate
        .probe_in14(RegDst),                  // [0:0] RegDst
        .probe_in15(RegWrite),                  // [0:0] RegWrite
        .probe_in16(ALUSrc1),                  // [0:0] ALUSrc1
        .probe_in17(ALUSrc2),                  // [0:0] ALUSrc2
        .probe_in18(ALUOp),                      // [2:0] ALUOp
        .probe_in19(MemWrite),                  // [0:0] MemWrite
        .probe_in20(MemToReg),                  // [0:0] MemToReg
        .probe_in21(pc),                      // [7:0] pc
        .probe_in22(instruction)              // [15:0] instruction
        );

endmodule
```

VIO Output Visualization:
Evidence is located in attached `lab7-test.mov` file. The following sequence of instructions was followed via a `mycode.coe` coefficient file.

```
clr $0               (0xD000)
clr $1               (0xD140)
clr $2               (0xD280)
clr $3               (0xD3C0)
addi $1, $0, 0x10    (0x3110)
sw $0, 0x4($1)       (0x1404)
addi $2, $0, 0x0F    (0x320F)
sw $0, 0x5($2)       (0x1805)

clr $0               (0xD000)
clr $1               (0xD140)
clr $2               (0xD280)
clr $3               (0xD3C0)
lw $1, 0x5($0)       (0x0105)
lw $2, 0x4($0)       (0x0204)
add $3, $1, $2       (0x26C0)
sw $0, 0x11($3)      (0x1C11)

clr $0               (0xD000)
clr $1               (0xD140)
clr $2               (0xD280)
clr $3               (0xD3C0)
lw $1, 0x5($0)       (0x0105)
inv $2, $1           (0x4180)
addi $3, $2, 0x01    (0x3B01)
sw $0, 0x12($3)      (0x1C12)

clr $0               (0xD000)
clr $1               (0xD140)
clr $2               (0xD280)
clr $3               (0xD3C0)
lw $1, 0x4($0)       (0x0104)
lw $2, 0x12($0)      (0x0212)
inv $1, $1           (0x4140)
addi $1, $1, 0x1     (0x3501)
add $3, $1, $2       (0x26C0)
sw $0, 0x13($0)      (0x1013)
```