Report

b03902001駱定暄

1. Coding Environment:
   Operating System: Ubuntu 14.04
   Simulation Tool: Icarus Verilog
   Wave Viewer: GTKWave

2. Module Implementation:
   (1)CPU.v
   wires:
      pc_o :
         Output:
            PC.pc_o
         Input:
            Add_PC.data1_in,
            Instruction_Memory.addr_i
      instr_o:
         Output:
            Instruction_Memory.instr_o
         Input:
            [31:26] to Control.Op_i,
            [25:21] to Registers.RSaddr_i,
            [20:16] to Registers.RTaddr_i and
                  MUX_RegDst.data1_i,
            [15:11] to MUX_RegDst.data2_i,
            [15:0] to Sign_Extend.data_i,
            [5:0] to ALU_Control.funct_i
   The rest of inputs and outputs are connected in the way of OOP.
   (2)Control.v

```
assign RegDst_o = (Op_i == 6'b000000)? 1'b1:
                  (Op_i == 6'b001000)? 1'b0:
                  1'b0;

assign ALUSrc_o = (Op_i == 6'b000000)? 1'b0:
                  (Op_i == 6'b001000)? 1'b1:
                  1'b0;

assign RegWrite_o = (Op_i == 6'b000000)? 1'b1:
                    (Op_i == 6'b001000)? 1'b1:
                    1'b0;

assign ALUOp_o = (Op_i == 6'b000000)? 2'b00:
                 (Op_i == 6'b001000)? 2'b01:
                 2'b00;
```

(3)ALU_Control.v

```
always @ (*)
begin
        if (ALUOp_i == 2'b00)
        begin
                if (funct_i == 6'b100000)
                        ALUCtrl_o = `ADD;
                else if (funct_i == 6'b100010)
                        ALUCtrl_o = `SUB;
                else if (funct_i == 6'b011000)
                        ALUCtrl_o = `MUL;
                else if (funct_i == 6'b100100)
                        ALUCtrl_o = `AND;
                else if (funct_i == 6'b100101)
                        ALUCtrl_o = `OR;
                else
                        ALUCtrl_o = 3'b000;
        end
        else if (ALUOp_i == 2'b01)
                ALUCtrl_o = `ADD;
        else
                ALUCtrl_o = 3'b000;
end
```

(4)MUX5.v and (5)MUX32.v

```
assign data_o = (select_i == 0)? data1_i: data2_i;
```

(6)Sign_Extend.v

```
assign data_o = {{16{data_i[15]}},data_i[15:0]};
```

(7)ALU.v

```
always @(*)
begin
case(ALUCtrl_i)
        `ADD: data_o = data1_i + data2_i;
        `SUB: data_o = data1_i - data2_i;
        `MUL: data_o = data1_i * data2_i;
        `AND: data_o = data1_i & data2_i;
        `OR: data_o = data1_i | data2_i;
        default: data_o = data1_i;
endcase
if (data_o == 32'd0)
        Zero_o = 1'b1;
else
        Zero_o = 1'b0;
end
```

(8)Adder.v

```
assign data_o = data1_in + data2_in;
```

3. Reference:

[1]https://www.csee.umbc.edu/portal/help/VHDL/verilog/types.html
[2]https://inst.eecs.berkeley.edu/~cs150/Documents/Nets.pdf
[3]https://class.ee.washington.edu/371/peckol/doc/Always@.pdf