```verilog
// TimeScale
`timescale 1ns/1ps

//Module Definition
module alu(
    // Inputs
    input I_clk,
    input I_en,
    input [4:0] I_aluop,
    input [15:0] I_dataA,
    input [15:0] I_dataB,
    input [7:0] I_imm,
    // Outputs
    output [15:0] O_dataResult,
    output reg O_shldBranch
);

    //Reg/Wire Declaration
    reg [17:0] int_result;
    wire op_lsb;
    wire [3:0] opcode;

    // Parameter Declaration
    localparam Add     = 0,
               Sub     = 1,
               OR      = 2,
               AND     = 3,
               XOR     = 4,
               NOT     = 5,
               Load    = 8,
               Cmp     = 9,
               SHL     = 10,
               SHR     = 11,
               JMPA    = 12,
               JMPR    = 13;


    // Initial Block
    initial begin
        int_result <= 0;
    end

    //Assigning Values
    assign op_lsb = I_aluop[0];
    assign opcode = I_aluop[4:1];
    assign O_dataResult = int_result[15:0];

    //ALU Operations
    always@(negedge I_clk) begin

        if(I_en) begin
            case(opcode)

                Add : begin
                        int_result <= (op_lsb ? ($signed(I_dataA) + $signed(I_dataB)) : (I_dataA + I_dataB));
                        O_shldBranch <= 0;
                    end

                Sub : begin
                        int_result <= (op_lsb ? ($signed(I_dataA) + $signed(I_dataB)) : (I_dataA + I_dataB));
                        O_shldBranch <= 0;
                    end

                OR :begin
                        int_result <= I_dataA | I_dataB;
                        O_shldBranch <= 0;
                   end

                AND :begin
                        int_result <= I_dataA & I_dataB;
                        O_shldBranch <= 0;
                    end

                XOR :begin
                        int_result <= I_dataA ^ I_dataB;
                        O_shldBranch <= 0;
                    end

                NOT :begin
                        int_result <= ~I_dataA;
                        O_shldBranch <= 0;
                    end

                Load : begin
                        int_result <= (op_lsb ? ({I_imm, 8'h00}) : ({8'h00, I_imm}));
                        O_shldBranch <= 0;
                    end

                Cmp : begin
                        if(op_lsb) begin
                            int_result[0] <= ($signed(I_dataA) == $signed(I_dataB)) ? 1 : 0;
                            int_result[1] <= ($signed(I_dataA) == 0) ? 1 : 0;
                            int_result[2] <= ($signed(I_dataB) == 0) ? 1 : 0;
                            int_result[3] <= ($signed(I_dataA) > $signed(I_dataB)) ? 1 : 0;
                            int_result[4] <= ($signed(I_dataA) < $signed(I_dataB)) ? 1 : 0;
                            end
                        else  begin
                            int_result[0] <= (I_dataA == I_dataB) ? 1 : 0;
                            int_result[1] <= (I_dataA == 0) ? 1 : 0;
                            int_result[2] <= (I_dataB == 0) ? 1 : 0;
                            int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
                            int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
                        end
                            O_shldBranch <= 0;
                    end

                SHL : begin
                    int_result <= I_dataA << (I_dataB[3:0]);
                    O_shldBranch <= 0;
                    end

                SHR : begin
                    int_result <= I_dataA >> (I_dataB[3:0]);
                    O_shldBranch <= 0;
                    end

                JMPA : begin
                    int_result <= (op_lsb ? I_dataA : I_imm);
                    O_shldBranch <= 1;
                    end

                JMPR : begin
                    int_result <= I_dataA;
                    O_shldBranch <= I_dataB[{op_lsb , I_imm[1:0]}];
                    end
            endcase
            end
    end
endmodule
```

```verilog
//TimeScale
`timescale 1ns / 1ps

//Module Definition
module ctrl_unit(
    // Inputs
    input I_clk,
    input I_reset,
    // Outputs
    output O_enfetch,
    output O_endec,
    output O_enrgrd,
    output O_enalu,
    output O_enrgwr,
    output O_enmem
);
    // Reg Declaration
    reg [5:0] state;

    // Initial  Block
    initial begin
        state <= 6'b000001;
    end

    //State Select Block
    always@(posedge  I_clk) begin
        if(I_reset)
            state  <= 6'b000001;
        else begin
            case(state)
                6'b000001 : state <= 6'b000010;
                6'b000010 : state <= 6'b000100;
                6'b000100 : state <= 6'b001000;
                6'b001000 : state <= 6'b010000;
                6'b010000 : state <= 6'b100000;
                default : state <= 6'b000001;
            endcase
        end
   end

    //Assignment Enable Signals
    assign O_enfetch = state[0];
    assign O_endec = state[1];
    assign O_enrgrd = state[2] | state[4];
    assign O_enalu = state[3];
    assign O_enrgwr = state[4];
    assign O_enmem = state[5];

endmodule

```

```verilog
//Timescale
`timescale 1ns / 1ps
//Module Definition
module fake_ram(
    //Inputs
    input I_clk,
    input I_we,
    input [15:0] I_addr,
    input [15:0] I_data,
    //outputs
    output reg [15:0] O_data
);
    //Memory declaration
    reg [15:0] mem [8:0];

    //Initialize registers
    initial begin

            mem[0] = 16'b1000000011111110;
            mem[1] = 16'b1000100111101101;
            mem[2] = 16'b0010001000100000;
            mem[3] = 16'b1000001100000001;
            mem[4] = 16'b1000010000000001;
            mem[5] = 16'b0000001101110000;
            mem[6] = 16'b1100000000000101;
            mem[7] = 0;
            mem[8] = 0;

            O_data = 16'b0000000000000000;
        end

        //Ram operation
        always@(negedge I_clk)begin
            if(I_we) begin
                mem[I_addr[15:0]] <= I_data;
            end
            O_data <= mem[I_addr[15:0]];
        end
endmodule
```

```verilog
//TimeScale
`timescale 1ns / 1ps

//Module Definition
module inst_dec(
    // Inputs
    input I_clk,
    input I_en,
    input [15:0] I_inst,
    // Outputs
    output reg [4:0] O_aluop,
    output reg [2:0] O_selA,
    output reg [2:0] O_selB,
    output reg [2:0] O_selD,
    output reg [15:0] O_imm,
    output reg O_regwe
);
    // Initial Block
    initial begin
        O_aluop <= 0;
        O_selA  <= 0;
        O_selB  <= 0;
        O_selD  <= 0;
        O_imm  <= 0;
        O_regwe <= 0;
    end
    always@(negedge I_clk) begin

        if(I_en) begin
            O_aluop <= I_inst[15:11];
            O_selA  <= I_inst[10:8];     //REG A
            O_selB  <= I_inst[7:5];       //REG B
            O_selD  <= I_inst[4:2];       //REG D
            O_imm   <= I_inst[7:0];       //Imm Data

            //REG Write Enable
            case(I_inst[15:12])
                4'b0111: O_regwe <= 0;
                4'b1100: O_regwe <= 0;
                4'b1101: O_regwe <= 0;
                default: O_regwe <= 1;
            endcase
        end
    end

endmodule
```

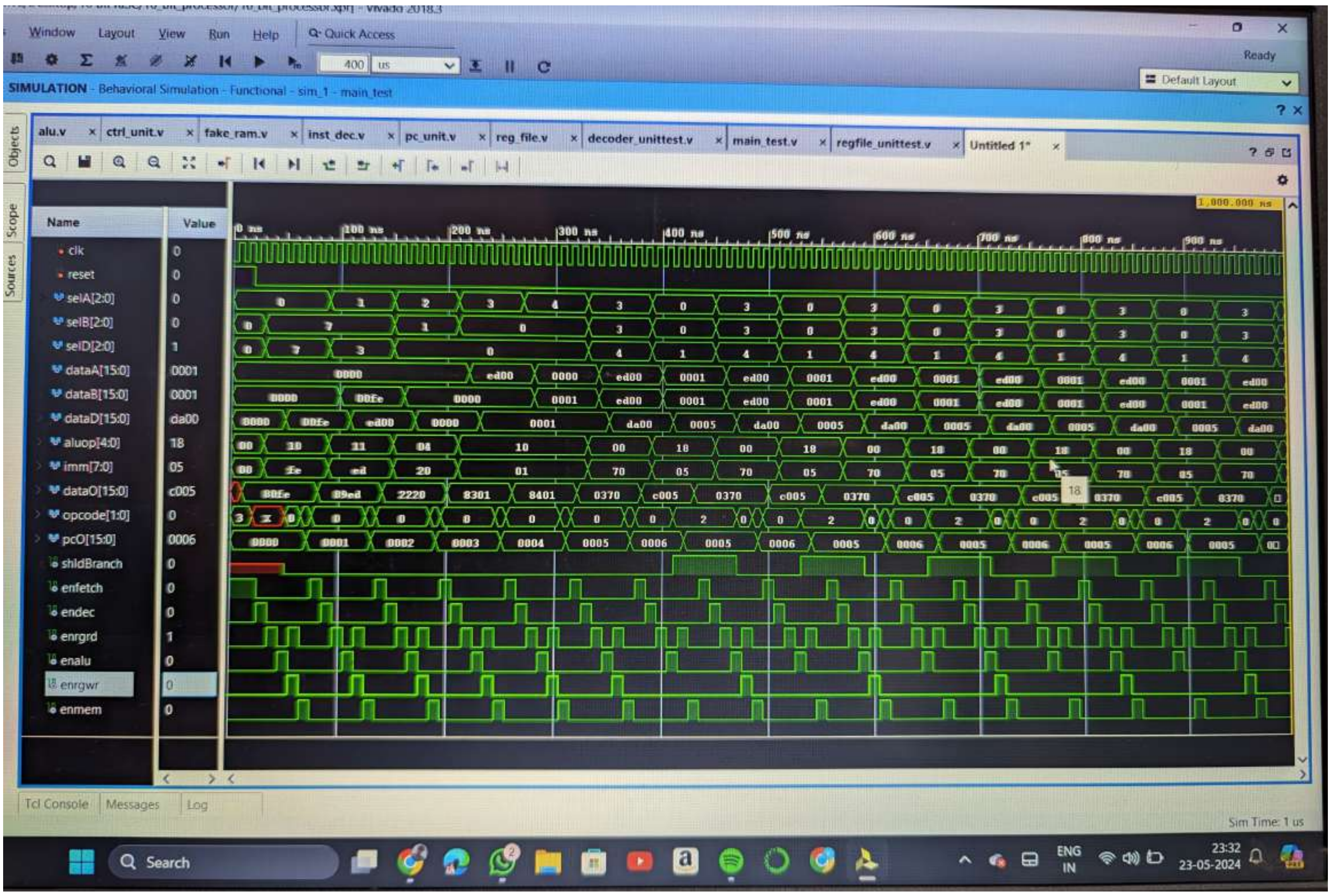
```verilog
// TimeScale
`timescale 1ns / 1ps

//Module Definition
module pc_unit(
    // Inputs
    input I_clk,
    input [1:0] I_opcode,
    input [15:0] I_pc,
    // Outputs
    output reg [15:0] O_pc
);

    // Initial Block
    initial begin
        O_pc <= 0;
    end

    // Program Counter State
    always@(negedge I_clk) begin
        case(I_opcode)
            2'b00 : O_pc <= O_pc;
            2'b01 : O_pc <= O_pc + 1;
            2'b10 : O_pc <= I_pc;
            2'b11 : O_pc <= 0;
        endcase
    end

endmodule
```

```verilog
// TimeScale
`timescale 1ns / 1ps

//Module Definition
module reg_file(
    // Inputs
    input I_clk,
    input I_en,
    input I_we,
    input [2:0] I_selA,
    input [2:0] I_selB,
    input [2:0] I_selD,
    input [15:0] I_dataD,
    // Outputs
    output reg [15:0] O_dataA,
    output reg [15:0] O_dataB
);

    // Internal register declaration
    reg [15:0] regs [7:0];

    // Loop Variable
    integer count;

    // Initialize register
    initial begin
        O_dataA = 0;
        O_dataB = 0;

        for(count = 0; count  < 8; count = count + 1) begin
            regs[count] = 0;
    end
    end

    // Assigning correct values to Op regs
    always@(negedge I_clk) begin
        if(I_en) begin
            if(I_we)
                regs[I_selD] <= I_dataD;

            O_dataA <= regs[I_selA];
            O_dataB <= regs[I_selB];
            end
    end

endmodule
```

```verilog
//Timescale
`timescale 1ns / 1ps

//Module Definition
module decoder_unittests();
    //Variable Declaration
    //Regs
    reg I_Clk;
    reg I_En;
    reg [15:0] I_Inst;
    //Wires
    wire [4:0] O_Aluop;
    wire [2:0] O_SelA;
    wire [2:0] O_SelB;
    wire [2:0] O_SelD;
    wire [15:0] O_Imm;
    wire O_Regwe;

inst_dec inst_unit(
    // Inputs
     I_Clk,
     I_En,
     I_Inst,
    // Outputs
    O_Aluop,
    O_SelA,
    O_SelB,
    O_SelD,
    O_Imm,
    O_Regwe
);

    initial begin
    //Time = 0
        I_Clk <= 0;
        I_En  <= 0;
        I_Inst <= 0;
    //Time = 10
        #10;
        I_Inst = 16'b0001011100000100;
    //TIme = 20
        #10;
        I_En = 1;
    end

    always begin
        #5;
        I_Clk = ~I_Clk;
    end
endmodule
```

```verilog
 // Timescale
`timescale 1ns / 1ps

// Module Definition
module main_test();
    // variable declaration
    //Regs
    reg clk;
    reg reset;
    reg reg_we = 0;
    reg [15:0] dataI = 0;
    //wire
    wire [2:0] selA;
    wire [2:0] selB;
    wire [2:0] selD;
    wire [15:0] dataA;
    wire [15:0] dataB;
    wire [15:0] dataD;
    wire [4:0] aluop;
    wire [7:0] imm;
    wire [15:0] dataO;
    wire [1:0] opcode;
    wire [15:0] pcO;

    wire shldBranch;
    wire enfetch;
    wire enalu;
    wire endec;
    wire enmem;
    wire enrgrd;
    wire enrgwr;
    wire regwe;
    wire update;

    //Assignments;
    assign enrgwr = regwe & update;
    assign opcode = (reset) ? 2'b11 : ((shldBranch) ? 2'b10 : ((enmem) ? 2'b01 : 2'b00));

    //instatiation
reg_file main_reg(
    // Inputs
    clk,
    enrgrd,
    enrgwr,
    selA,
    selB,
    selD,
    dataD,
    // Outputs
    dataA,
    dataB
);

inst_dec main_inst(
     // Inputs
     clk,
     endec,
     dataO,
     // outputs
     aluop,
     selA,
     selB,
     selD,
     imm,
     regwe
);

alu main_alu(
    // Inputs
    clk,
    enalu,
    aluop,
    dataA,
    dataB,
    imm,
    // Outputs
    dataD,
    shldBranch
);

ctrl_unit main_ctrl(
     // Inputs
     clk,
     reset,
     // Outputs
     enfetch,
     endec,
     enrgrd,
     enalu,
     update,
     enmem
);

pc_unit pc_main(
    // Inputs
    clk,
    opcode,
    dataD,
    // Outputs
    pcO
);

fake_ram main_ram (
     // Inputs
     clk,
     reg_we,
     pcO,
     dataI,
    // Outputs
    dataO
);

    initial begin

         clk = 0;
        reset = 1;
        #20
        reset = 0;

    end

    //Clock generation
    always begin
       #5;
       clk = ~clk;
    end


endmodule
```

```verilog
//Timescale
`timescale 1ns / 1ps

//Module Definition
module regfile_unittest();
    //Variable Declaration
    //Regs
    reg I_clk;
    reg [15:0] I_dataD;
    reg I_en;
    reg [2:0] I_selA;
    reg [2:0] I_selB;
    reg [2:0] I_selD;
    reg I_we;
    //Wires
    wire O_dataA;
    wire O_dataB;

reg_file reg_test(
    //Inputs
    I_clk,
    I_en,
    I_we,
    I_selA,
    I_selB,
    I_selD,
    I_dataD,
    //Outputs
    O_dataA,
    O_dataB
);

    initial begin
        //Reset all Inputs
        I_clk = 1'b0;
        I_dataD = 0;
        I_en = 0;
        I_selA = 0;
        I_selB = 0;
        I_selD = 0;
        I_we = 0;

        //Start Test
        //Time = 7
        #7
        I_en = 1'b1;

        I_selA = 3'b000;
        I_selB = 3'b001;
        I_selD = 3'b000;

        I_dataD = 16'hFFFF;
        I_we = 1'b1;

        //Time = 17
        I_we = 1'b0;
        I_selD = 3'b010;
        I_dataD = 16'h2222;

        //Time = 27
        #10;
        I_we = 1;

        //Time = 37
        #10;
        I_dataD = 16'h3333;

        //Time = 47
        #10;
        I_we = 0;
        I_selD = 3'b000;
        I_dataD = 16'hFEED;

        //Time = 57
        #10;
        I_selD = 3'b100;
        I_dataD = 16'h4444;

        //Time = 67
        #10;
        I_we = 1;

        //Time = 117
        #50;
        I_selA = 3'b100;
        I_selB = 3'b100;
        #20;
        $finish;

    end

    //Clock generation
    always begin
        #5;
        I_clk = ~I_clk;
    end

endmodule
```


Window Layout View Run Help
Quick Access
Ready
Default Layout
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
alu.v
ctrl_unit.v
fake_ram.v
inst_dec.v
pc_unit.v
reg_file.v
decoder_unittest.v
main_test.v
regfile_unittest.v
Untitled 1*
Name
Value
clk 0
reset 0
selA[2:0] 0
selB[2:0] 0
selD[2:0] 1
dataA[15:0] 0001
dataB[15:0] 0001
dataD[15:0] da00
aluop[4:0] 18
imm[7:0] 05
dataO[15:0] c005
opcode[1:0] 0
pcO[15:0] 0006
shldBranch 0
enfetch 0
endec 0
enrgrd 1
enalu 0
enrgwr 0
enmem 0
1,000.000 ns
Tcl Console
Messages
Log
Sim Time: 1 us
Search
ENG IN
23:32
23-05-2024