Lab2

PB22030892 刘铠瑜

任务 1：寄存器堆设计

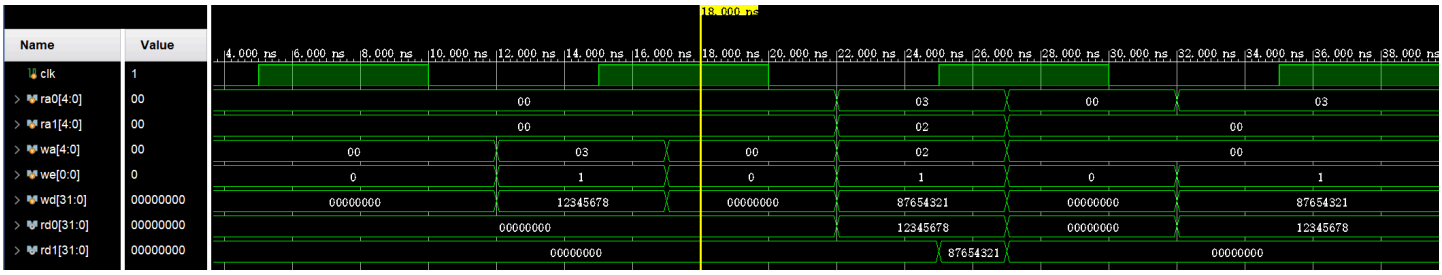
REG_FILE.v

```
module REG_FILE (
    input                   [ 0 : 0]        clk,

    input                   [ 4 : 0]        rf_ra0,     // 读寄存器地址
    input                   [ 4 : 0]        rf_ra1,
    input                   [ 4 : 0]        rf_wa,      // 写寄存器地址
    input                   [ 0 : 0]        rf_we,      // 写使能信号
    input                   [31 : 0]        rf_wd,      // 写数据

    output                  [31 : 0]        rf_rd0,     // 读数据
    output                  [31 : 0]        rf_rd1
);

    reg [31 : 0] reg_file [0 : 31];

    // 用于初始化寄存器
    integer i;
    initial begin
        for (i = 0; i < 32; i = i + 1)
            reg_file[i] = 0;
    end

    always @(posedge clk) begin             // 在时钟的上升沿进行判断
        // 在写使能信号有效的同时，寄存器地址不为0才能写入，保证0号寄存器始终为0
        if (rf_we && (rf_wa != 0)) begin
            reg_file[rf_wa] <= rf_wd;
        end
    end

    // 读操作时钟异步
    assign rf_rd0 = reg_file[rf_ra0];
    assign rf_rd1 = reg_file[rf_ra1];

endmodule
```

REG_FILE_tb.v

```
module REG_FILE_tb ();
    reg                 clk;
    reg     [ 4 : 0]    ra0, ra1, wa;
    reg     [ 0 : 0]    we;
    reg     [31 : 0]    wd;
    wire    [31 : 0]    rd0;
    wire    [31 : 0]    rd1;

    REG_FILE regfile (
        .clk    (clk),
        .rf_ra0    (ra0),
        .rf_ra1    (ra1),
        .rf_wa     (wa),
        .rf_we     (we),
        .rf_wd     (wd),
        .rf_rd0    (rd0),
        .rf_rd1    (rd1)
    );

    initial begin
        clk = 0;
        ra0 = 5'H0; ra1 = 5'H0; wa = 5'H0; we = 1'H0; wd = 32'H0;

        #12
        ra0 = 5'H0; ra1 = 5'H0; wa = 5'H3; we = 1'H1; wd = 32'H12345678;

        #5
        ra0 = 5'H0; ra1 = 5'H0; wa = 5'H0; we = 1'H0; wd = 32'H0;

        #5
        ra0 = 5'H3; ra1 = 5'H2; wa = 5'H2; we = 1'H1; wd = 32'H87654321;

        #5
        ra0 = 5'H0; ra1 = 5'H0; wa = 5'H0; we = 1'H0; wd = 32'H0;

        #5
        ra0 = 5'H3; ra1 = 5'H0; wa = 5'H0; we = 1'H1; wd = 32'H87654321;

        #10
        $finish;
    end
    always #5 clk = ~clk;
endmodule
```

波形图



任务 2：ALU 设计

ALU.v

```
module ALU (
    input                   [31 : 0]            alu_src0,
    input                   [31 : 0]            alu_src1,
    input                   [ 4 : 0]            alu_op,

    output      reg         [31 : 0]            alu_res
);

    `define ADD                 5'B00000
    `define SUB                 5'B00010
    `define SLT                 5'B00100
    `define SLTU                5'B00101
    `define AND                 5'B01001
    `define OR                  5'B01010
    `define XOR                 5'B01011
    `define SLL                 5'B01110
    `define SRL                 5'B01111
    `define SRA                 5'B10000
    `define SRC0                5'B10001
    `define SRC1                5'B10010

    always @(*) begin
        case(alu_op)
            `ADD  :
                alu_res = alu_src0 + alu_src1;
            `SUB  :
                alu_res = alu_src0 - alu_src1;
            `SLT  :
                alu_res = ($signed(alu_src0) < $signed(alu_src1)) ? 32'H1 : 32'H0;
            `SLTU :
                alu_res = (alu_src0 < alu_src1) ? 32'H1 : 32'H0;
            `AND  :
                alu_res = alu_src0 & alu_src1;
            `OR   :
                alu_res = alu_src0 | alu_src1;
            `XOR  :
                alu_res = alu_src0 ^ alu_src1;
            `SLL  :
                alu_res = alu_src0 << alu_src1[4:0];
            `SRL  :
                alu_res = alu_src0 >> alu_src1[4:0];
            `SRA  :
                alu_res = $signed(alu_src0) >>> alu_src1[4:0];
            `SRC0 :
                alu_res = alu_src0;
            `SRC1 :
```

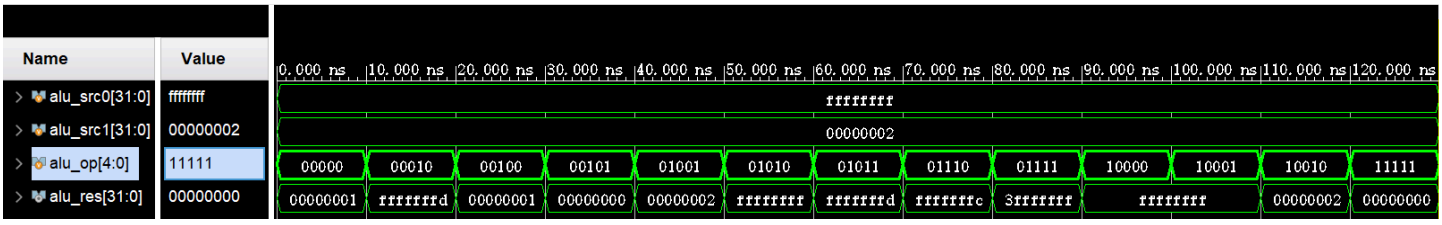
```
                alu_res = alu_src1;
            default :
                alu_res = 32'H0;
        endcase
    end
endmodule
```

ALU_tb.v

```
`timescale 1ns / 1ps

module ALU_tb;

    // Inputs
    reg [31:0] alu_src0;
    reg [31:0] alu_src1;
    reg [4:0] alu_op;

    // Outputs
    wire [31:0] alu_res;

    ALU uut (
        .alu_src0(alu_src0),
        .alu_src1(alu_src1),
        .alu_op(alu_op),
        .alu_res(alu_res)
    );

    initial begin
        // Initialize inputs
        alu_src0 = 0;
        alu_src1 = 0;
        alu_op = 0;

        // 测试数据为 -1 和 2
        alu_src0 = 32'hFFFFFFFF;
        alu_src1 = 32'h00000002;
        alu_op = 5'b00000; // ADD
        #10;

        alu_op = 5'b00010; // SUB
        #10;

        alu_op = 5'b00100; // SLT
        #10;

        alu_op = 5'b00101; // SLTU
        #10;

        alu_op = 5'b01001; // AND
        #10;

        alu_op = 5'b01010; // OR
        #10;
```

```
        alu_op = 5'b01011; // XOR
        #10;

        alu_op = 5'b01110; // SLL
        #10;

        alu_op = 5'b01111; // SRL
        #10;

        alu_op = 5'b10000; // SRA
        #10;

        alu_op = 5'b10001; // SRC0
        #10;

        alu_op = 5'b10010; // SRC1
        #10;

        alu_op = 5'b11111; // Invalid operation
        #10;

        $finish;
    end

endmodule
```

波形图



任务 3：在线计算器

Top.v

```
module TOP (
    input                   [ 0 : 0]            clk,
    input                   [ 0 : 0]            rst,

    input                   [ 0 : 0]            enable,
    input                   [ 4 : 0]            in,
    input                   [ 1 : 0]            ctrl,

    output                  [ 3 : 0]            seg_data,
    output                  [ 2 : 0]            seg_an
);

    reg [31:0] alu_src0;
    reg [31:0] alu_src1;
    reg [4:0] alu_op;
    wire [31:0] alu_res;
    reg [31:0] output_data;

    ALU alu (
        .alu_src0(alu_src0),
        .alu_src1(alu_src1),
        .alu_op(alu_op),
        .alu_res(alu_res)
    );

    Segment segment (
        .clk(clk),
        .rst(rst),
        .output_data(output_data),
        .seg_data(seg_data),
        .seg_an(seg_an)
    );

    always @(posedge clk or posedge rst) begin      // 异步复位
        if (rst) begin
            alu_src0 <= 32'b0;
            alu_src1 <= 32'b0;
            alu_op <= 5'b0;
            output_data <= 32'b0;
        end else if (enable) begin
            case (ctrl)
                2'b00: begin
                    alu_op <= in;                   // 输入操作码
                end
                2'b01: begin
                    alu_src0 <= {{27{in[4]}}, in};  // 输入第一个操作数，符号扩展为32位
```

```
                end
                2'b10: begin
                    alu_src1 <= {{27{in[4]}}, in};  // 输入第二个操作数，符号扩展为32位
                end
                2'b11: begin
                    output_data <= alu_res;         // 输出结果
                end
            endcase
        end
    end

endmodule
```

任务 4：初始化存储器

dist_mem_gen_0.veo

```
dist_mem_gen_0 your_instance_name (
  .a(a),      // input wire [5 : 0] a
  .d(d),      // input wire [31 : 0] d
  .clk(clk),  // input wire clk
  .we(we),    // input wire we
  .spo(spo)  // output wire [31 : 0] spo
);
```

top.v

```
module top (
    input wire [5:0] a,        // 地址输入
    input wire [31:0] d,       // 数据输入
    input wire clk,            // 时钟信号
    input wire we,             // 写使能信号
    output wire [31:0] spo     // 数据输出
);

    // 实例化 dist_mem_gen_0
    dist_mem_gen_0 u_dist_mem_gen_0 (
        .a(a),
        .d(d),
        .clk(clk),
        .we(we),
        .spo(spo)
    );

endmodule
```